INTELLIGENT MOBILE PRIVATE TUTOR FINDERS APPLICATION

LEONG KAM GA

UNIVERSITI TUNKU ABDUL RAHMAN

INTELLIGENT MOBILE PRIVATE TUTOR FINDERS APPLICATION

LEONG KAM GA

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2022

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	Leong Kam Ga
ID No.	:	18UEB02038
Date	:	9 May 2022

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“INTELLIGENT MOBILE PRIVATE TUTOR FINDERS APPLICATION”** was prepared by **LEONG KAM GA** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Hons) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Too Chian Wen
Date	:	10 May 2022

Signature	:	
Co-Supervisor	:	
Date	:	

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2022, Leong Kam Ga. All right reserved.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Too Chian Wen for her invaluable advice, guidance, and her enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement and advice during the process of conducting this project.

ABSTRACT

According to the official statistic from the Department of Statistics Malaysia (DOSM), there are 1899 tuition centers was established in Malaysia in 2010. The number of registered private tuition centers had increased to 3107 in Malaysia in 2013. The demand for private tutoring services was elevated day by day due to the changes in the Malaysia examination and assessment system. In addition, it is a challenge to find a private tutor during this COVID-19 pandemic because it has a high risk to meet with each of the tutors for choosing the most matched private tutor. However, there are only limited online platforms available in Malaysia while existing platforms only provide simple filtering criteria in the searching tutor function and only apply traditional searching methods (SQL exact matching).

Therefore, an intelligent mobile private tutor application is developed to help the tutor seeker in finding the ideal tutor that is highly matched their expectation at their fingertips. By putting aside the traditional searching method, this application applied similarity measures (Euclidean Distance, Manhattan Distance, Minkowski Distance, Jaccard Similarity Coefficient, and Cosine Similarity) to compare the similarity between one tutor with the ideal tutor of the tutor seeker. The similarity percentage is considered as a metric for tutor seekers to choose the ideal tutor while not only relying on intuition. In addition, there are chat functions, demo classes, and other functions provided to tutor seekers to have a formal channel to interact and communicate with the tutor before coming to the final decision of having a formal class. After conducting the unit testing, API testing, usability testing, and User Acceptance Testing (UAT), the final product only applied the top three similarity measure methods in the search function. The final product of the project enables tutor seekers to search for their ideal tutor while also enabling tutors to expose themselves to more opportunities.

TABLE OF CONTENTS

DECLARATION	**iii**
APPROVAL FOR SUBMISSION	**iv**
ACKNOWLEDGEMENTS	**vi**
ABSTRACT	**vii**
TABLE OF CONTENTS	**viii**
LIST OF TABLES	**xiii**
LIST OF FIGURES	**xvii**
LIST OF SYMBOLS / ABBREVIATIONS	**xxviii**
LIST OF APPENDICES	**xxix**

CHAPTER

1	**INTRODUCTION**			**1**
	1.1	Introduction		1
	1.2	Problem statement		2
	1.3	Project Objectives		4
	1.4	Project Solution		4
	1.5	Project Approach		6
	1.6	Scope of the Project		7
		1.6.1	Targeted Users	7
		1.6.2	End Users Involved	7
		1.6.3	Modules Covered	8
2	**LITERATURE REVIEW**			**9**
	2.1	Introduction		9
	2.2	Study of Existing Similar Application		9
		2.2.1	MyPrivateTutor	9
		2.2.2	TeachMe	12

2.2.3 Tuition Hero Malaysia 16
2.2.4 Stapps 19
2.2.5 Zhang Men 25
2.2.6 Comparison of Existing Similar Application 32
2.3 Searching Approaches 33
2.3.1 Exact searching by using SQL query 33
2.3.2 Similarity Measures 35
2.3.3 K-Nearest Neighbours (KNN) Algorithm 42
2.3.4 Summary of Searching Approaches 43
2.4 Software Development Methodology 43
2.4.1 Waterfall Model 44
2.4.2 Prototyping Model 45
2.4.3 Agile Software Development Model 47
2.4.4 Rapid Application Development (RAD) Model 48
2.4.5 Comparison of Software Development Methodologies 50
2.5 Usability Testing 51
2.5.1 Overview of Usability Testing 51
2.5.2 Usability Testing Methods 55
2.5.3 Summary of Usability Testing 56

3 METHODOLOGY AND WORK PLAN 57
3.1 Introduction 57
3.2 Software Development Methodology 57
3.2.1 Initial Requirements 58
3.2.2 Prototyping Iteration 58
3.2.3 Development 61
3.2.4 Test 61
3.2.5 Deployment and Maintenance 62
3.3 Research Methodology 62
3.3.1 Questionnaire 62
3.3.2 Literature Reviews 63
3.4 Project Planning 64

		3.4.1 Work Break Down Structure (WBS)	64
		3.4.2 Project Duration Plan	67
		3.4.3 Gantt Chart	69
	3.5	Development Tools	69
		3.5.1 React Native	69
		3.5.2 Firebase	70
		3.5.3 Visual Studio Code	70
		3.5.4 Express.js	70
		3.5.5 Postman	71
		3.5.6 AnyDesk	71
4	**PROJECT INITIAL SPECIFICATION**		**72**
	4.1	Introduction	72
	4.2	Use Case Modelling	73
		4.2.1 Application’s Use Case Diagram	73
		4.2.2 Use Case Description	74
	4.3	System Requirements	91
		4.3.1 Functional Requirements	91
		4.3.2 Non-functional Requirements	92
	4.4	Fact Findings	92
		4.4.1 Overview of Responses	92
		4.4.2 Data Analysis	95
5	**SYSTEM DESIGN**		**98**
	5.1	Introduction	98
	5.2	System Architecture Design	98
		5.2.1 Three-tier Architecture	98
		5.2.2 System Architecture Diagram	100
		5.2.3 Context Diagram	103
		5.2.4 Data Flow Diagram	104
		5.2.5 Activity Diagram	105
	5.3	Data Model Design	120
		5.3.1 Logical Data Model (LDM)	120

		5.3.2 Physical Data Model (PDM)	121
		5.3.3 Data Dictionary	123
	5.4	User Interface Design	127
		5.4.1 Navigation Model	127
		5.4.2 User Interface Display	130
6	**SYSTEM IMPLEMENTATION**		**150**
	6.1	Introduction	150
	6.2	Application Programming Interface (API) Endpoint	150
	6.3	Application Implementation	155
		6.3.1 Sign Up Activity	155
		6.3.2 Sign In Activity	158
		6.3.3 Registered Tutor Seeker Side	160
		6.3.4 Registered Tutor Side	188
		6.3.5 Admin Side	197
7	**SYSTEM TESTING**		**204**
	7.1	Introduction	204
	7.2	Unit Testing	204
	7.3	Application Programming Interface (API) Testing	214
	7.4	User Acceptance Testing (UAT)	225
		7.4.1 Tutor Seeker Side	225
		7.4.2 Registered Tutor Side	227
		7.4.3 Admin Side	228
	7.5	Usability Testing	229
		7.5.1 Usability Testing Scenario	229
		7.5.2 User Satisfaction Survey Form	233
		7.5.3 Usability Test Result Analysis	236
8	**CONCLUSIONS AND RECOMMENDATIONS**		**238**
	8.1	Conclusions	238
	8.2	Limitations and Future Enhancement	239

REFERENCES **240**

APPENDICES **247**

LIST OF TABLES

Table 2.1: Comparison of Existing Similar Application 32

Table 2.2: Strengths and Limitations of SQL Query 34

Table 2.3: Strengths and Limitations of Euclidean Distance 36

Table 2.4: Strengths and Limitations of Manhattan Distance 37

Table 2.5: Strengths and Limitations of Minkowski Distance 39

Table 2.6: Strengths and Limitations of Cosine Similarity 40

Table 2.7: Strengths and Limitations of Jaccard Similarity Coefficient 41

Table 2.8: Strengths and Limitations of KNN Algorithm 42

Table 2.9: Pros and Cons of Waterfall Model 45

Table 2.10: Pros and Cons of Prototyping Model 46

Table 2.11: Pros and Cons of Agile Software Development Model 48

Table 2.12: Pros and Cons of Rapid Application Development (RAD) Model 49

Table 2.13: Comparison of Software Development Methodologies 50

Table 2.14: Moderated and Unmoderated Usability Testing 54

Table 2.15: Remote and In-person Usability Testing 54

Table 2.16: Explorative and Comparative Usability Testing 54

Table 3.1: Project Duration Plan 67

Table 4.1: Description of Use Case - Register New Account 74

Table 4.2: Description of Use Case - Login Account 75

Table 4.3: Description of Use Case - Edit Profile 76

Table 4.4: Description of Use Case - Search Tutor 77

Table 4.5: Description of Use Case - View Tutor’s Profile 78

Table 4.6: Description of Use Case – Initiate chat 80

Table 4.7: Description of Use Case - View Class Detail 81

Table 4.8: Description of Use Case - Manage Demo Class Request 82

Table 4.9: Description of Use Case - View Formal Class Booking 83

Table 4.10: Description of Use Case - View Chat 84

Table 4.11: Description of Use Case – Reply Message 85

Table 4.12: Description of Use Case – Notify User 86

Table 4.13: Description of Use Case - Create Tutor’s Account 87

Table 4.14: Description of Use Case – Update Tutor’s Profile 88

Table 4.15: Description of Use Case - View User List 89

Table 4.16: Description of Use Case – Delete Account 90

Table 4.17: Count of Rating in Tutor Criteria Selection 94

Table 5.1: Data Dictionary (Users Collection) 123

Table 5.2: Data Dictionary (Tutor Info Collection) 123

Table 5.3: Data Dictionary (Demo Class Registration Collection) 124

Table 5.4: Data Dictionary (Formal Class Registration Collection) 125

Table 5.5: Data Dictionary (Rate Review Collection) 125

Table 5.6: Data Dictionary (Chat Collection) 126

Table 6.1: API Endpoint List 151

Table 7.1: Unit Testing – Log In Account 205

Table 7.2: Unit Testing – Sign Up Account 206

Table 7.3: Unit Testing – Search Tutor 207

Table 7.4: Unit Testing – Send Message 207

Table 7.5: Unit Testing – Send Demo Class Request 208

Table 7.6: Unit Testing – Book Formal Class 209

Table 7.7: Unit Testing – Rate and Review 210

Table 7.8: Unit Testing – Delete Account 210

Table 7.9: Unit Testing – Edit Profile 211

Table 7.10: Unit Testing – Manage Demo Class 211

Table 7.11: Unit Testing – View Formal Class 211

Table 7.12: Unit Testing – Add Tutor 212

Table 7.13: Unit Testing – Update Tutor's Professional Detail 213

Table 7.14: Unit Testing – View Tutor Seeker List 213

Table 7.15: Testing Information Numbering 214

Table 7.16: UAT – Create and Login Account 225

Table 7.17: UAT – Search and View Ideal Tutor 226

Table 7.18: UAT – Chat With Tutor 226

Table 7.19: UAT – Register and View Classes 226

Table 7.20: UAT – View and Edit Profile (Tutor Seeker) 227

Table 7.21: UAT – Log In Account (Registered Tutor) 227

Table 7.22: UAT – View and Edit Profile (Registered Tutor) 227

Table 7.23: UAT – Manage Class 227

Table 7.24: UAT – Chat With Tutor Seeker 228

Table 7.25: UAT – Log In Account (Admin) 228

Table 7.26: UAT – Manager Tutor 228

Table 7.27: UAT – View Tutor Seeker 229

Table 7.28: Usability Testing Scenario – Tutor Seeker 229

Table 7.29: Usability Testing Scenario – Registered Tutor 231

Table 7.30: Usability Testing Scenario – Admin 232

Table 7.31: Matching Rate of Similarity Measures' Search Results 237

LIST OF FIGURES

Figure 1.1: Overview of Application Architecture 6

Figure 1.2: Prototyping Methodology 6

Figure 2.1: Filter Result of MyPrivateTutor 10

Figure 2.2: Tutor Profile of MyPrivateTutor 10

Figure 2.3: Tutor Request of MyPrivateTutor 11

Figure 2.4: Tutor Suggestion of MyPrivateTutor 11

Figure 2.5: Job Applying of MyPrivateTutor 12

Figure 2.6: Filter Result of TeachMe 12

Figure 2.7: Tutors Profile of TeachMe 13

Figure 2.8: Online Tutoring Class Demo of TeachMe 14

Figure 2.9: Tutor Availability of TeachMe 14

Figure 2.10: Tutor Lessons of TeachMe 15

Figure 2.11: Tutor Add Subjects of TeachMe 15

Figure 2.12: Tutor Teaching Materials of TeachMe 15

Figure 2.13: Tutor Request Form of Tuition Hero Malaysia 17

Figure 2.14: Home Tuition Fee Calculator of Tuition Hero Malaysia 18

Figure 2.15: Job Applying of Tuition Hero Malaysia (Web Application) 18

Figure 2.16: Job Applying of Tuition Hero Malaysia (Mobile Application) 19

Figure 2.17: Tutor Filtering of Stapps 20

Figure 2.18: Tutor Sorting of Stapps 20

Figure 2.19: Tutor Requesting of Stapps 21

Figure 2.20: Tutors Suggestion of Stapps 22

Figure 2.21: Learning Material Sharing of Stapps 22

Figure 2.22: Tutors' Profile of Stapps 23

Figure 2.23: Job Filtering of Stapps 24

Figure 2.24: Job Sorting of Stapps 24

Figure 2.25: Trial Tutoring Class of Zhang Men (Web Application) 25

Figure 2.26: Trial Tutoring Class of Zhang Men (Mobile Application) 26

Figure 2.27: Tutor Teaching Style of Trial Tutoring Class of Zhang Men (Mobile Application) 26

Figure 2.28: Instrument Testing of Zhang Men (Mobile Application) 27

Figure 2.29: Class Schedular of Zhang Men (Web Application) 27

Figure 2.30: Class Schedular of Zhang Men (Mobile Application) 28

Figure 2.31: Revision Schedular of Zhang Men (Mobile Application) 28

Figure 2.32: Learning Materials of Zhang Men 29

Figure 2.33: Public Class of Zhang Men (Web Application) 29

Figure 2.34: Public Class of Zhang Men (Mobile Application) 30

Figure 2.35: AI English Level Test of Zhang Men 31

Figure 2.36: Learning Ability Test of Zhang Men 31

Figure 2.37: Basic Structure of SQL Query 34

Figure 2.38: Graphical Representation of Euclidean Distance 35

Figure 2.39: Sample Python Code by Applying Euclidean Distance 36

Figure 2.40: Graphical Representation of Manhattan Distance 37

Figure 2.41: Sample Python Code by Applying Manhattan Distance 37

Figure 2.42: Graphical Representation of Manhattan Distance 38

Figure 2.43: Sample Python Code by Applying Minkowski Distance 38

Figure 2.44: Graphical Representation of Cosine Similarity 39

Figure 2.45: Sample Python Code by Applying Cosine Similarity 40

Figure 2.46: Graphical Representation of Jaccard Similarity Coefficient 41

Figure 2.47: Sample Python Code by Applying Jaccard Similarity Coefficient 41

Figure 2.48: Graphical Representation of KNN Algorithm 42

Figure 2.49: Waterfall Model 44

Figure 2.50: Prototyping Model 45

Figure 2.51: Agile Software Development Model 47

Figure 2.52: Rapid Application Development (RAD) Model 48

Figure 2.53: Usability Testing Flow of Information 52

Figure 2.54: Sample Template of SUS 53

Figure 2.55: Usability Testing Models 55

Figure 3.1: Prototyping Model 57

Figure 4.1: Use Case Diagram 73

Figure 4.2: Distribution of Parents and Students 92

Figure 4.3: Tutor Selection Criteria (Tutors’ Criteria) 93

Figure 4.4: Tutor Selection Criteria (External Criteria) 93

Figure 4.5: Tutor Selection Criteria (External Criteria) 95

Figure 4.6: Overall Average Score of Tutor Selection Criteria 95

Figure 4.7: Average Score of Tutor Selection Criteria (Parents) 96

Figure 4.8: Average Score of Tutor Selection Criteria (Students) 97

Figure 5.1: Three-tier Architecture Diagram 98

Figure 5.2: System Architecture Diagram 100

Figure 5.3: Firestore Access Rules 102

Figure 5.4: Context Diagram 103

Figure 5.5: DFD - Level 0 104

Figure 5.6: Activity Diagram – Register New Account 105

Figure 5.7: Activity Diagram – Login Account 106

Figure 5.8: Activity Diagram – Edit Profile 107

Figure 5.9: Activity Diagram – Search Tutor 108

Figure 5.10: Activity Diagram – View Tutor’s Profile 109

Figure 5.11: Activity Diagram – Initiate Chat 110

Figure 5.12: Activity Diagram – View Class Detail 111

Figure 5.13: Activity Diagram – Manage Demo Class Request 112

Figure 5.14: Activity Diagram – View Formal Class Booking 113

Figure 5.15: Activity Diagram – View Chat 114

Figure 5.16: Activity Diagram – Reply Message 115

Figure 5.17: Activity Diagram – Notify User 116

Figure 5.18: Activity Diagram – Create Tutor’s Account 117

Figure 5.19: Activity Diagram – Update Tutor’s Profile 118

Figure 5.20: Activity Diagram – View User List 119

Figure 5.21: Logical Data Model Diagram 120

Figure 5.22: Physical Data Model Diagram 121

Figure 5.23: Data Schema 122

Figure 5.24: Navigation Model Diagram (Overview) 127

Figure 5.25: Navigation Model Diagram (Student Interface) 128

Figure 5.26: Navigation Model Diagram (Tutor Interface) 129

Figure 5.27: Navigation Model Diagram (Admin Interface) 129

Figure 5.28: UI – Sign in 130

Figure 5.29: UI – Sign up 130

Figure 5.30: UI – Tutor Seeker Home 131

Figure 5.31: UI – Tutor Seeker Class Requests History 131

Figure 5.32: UI – Tutor Seeker Rate and Review 132

Figure 5.33: UI – Tutor Seeker Chat History 132

Figure 5.34: UI – Tutor Seeker Chat Box 133

Figure 5.35: UI – Tutor Seeker Push Notification (From Tutor) 133

Figure 5.36: UI – Tutor Seeker Search Tutor 134

Figure 5.37: UI – Tutor Seeker Search Result (Manhattan Distance) 134

Figure 5.38: UI – Tutor Seeker Search Result (Euclidean Distance) 135

Figure 5.39: UI – Tutor Seeker Search Result (Minkowski Distance) 135

Figure 5.40: UI – Tutor Seeker Search Result (Jaccard Similarity Coefficient) 136

Figure 5.41: UI – Tutor Seeker Search Result (Cosine Similarity) 136

Figure 5.42: UI – Tutor Seeker Tutor Profile 137

Figure 5.43: UI – Tutor Seeker Demo and Formal Class Registration 137

Figure 5.44: UI – Tutor Seeker Profile and Profile Edit 138

Figure 5.45: UI – Tutor Home 138

Figure 5.46: UI – Tutor Demo Class Manage 139

Figure 5.47: UI – Tutor Formal Class Manage 139

Figure 5.48: UI – Tutor Chat History 140

Figure 5.49: UI – Tutor Chat Box 140

Figure 5.50: UI – Tutor Push Notification (From Tutor Seeker) 141

Figure 5.51: UI – Tutor Profile 141

Figure 5.52: UI – Tutor Profile Edit 142

Figure 5.53: UI – Admin Home 143

Figure 5.54: UI – Admin Add Tutor 144

Figure 5.55: UI – Admin Update Tutor 145

Figure 5.56: UI – Admin Student List 145

Figure 5.57: IFD – Helper Text (Input Error Handling) 146

Figure 5.58: IFD – Activity Indicator 147

Figure 5.59: IFD – Password Visibility Toggle 147

Figure 5.60: IFD – Empty Record Display 148

Figure 5.61: IFD – Alert Messages 149

Figure 6.1: Upload Profile Picture 155

Figure 6.2: Sign Up Tutor Seeker Account 156

Figure 6.3: Section Code (Front-end) – Sign Up Tutor Seeker Account 157

Figure 6.4: Section Code (Server API) – Sign Up Tutor Seeker Account 157

Figure 6.5: Sign In Account 158

Figure 6.6: Section Code (Front-end) – Sign In Account 159

Figure 6.7: Section Code (Server API) – Retrieve Basic Profile 159

Figure 6.8: Home (Registered Tutor Seeker) 160

Figure 6.9: Section Code (Server API) – Retrieve Popular Tutor Part 1 161

Figure 6.10: Section Code (Server API) – Retrieve Popular Tutor Part 2 161

Figure 6.11: Section Code (Front-end) – Home AsynStorage Activity 162

Figure 6.12: Push Notification 162

Figure 6.13: Section Code (Front-end) – Home Push Notification Activity 163

Figure 6.14: Section Code (Server API) – Update FCM Token 163

Figure 6.15: History (Demo Class and Formal Class) 164

Figure 6.16: Section Code (Front-end) – History Tab View 164

Figure 6.17: Section Code (Server API) – Retrieve Class Part 1 165

Figure 6.18: Section Code (Server API) – Retrieve Class Part 2 166

Figure 6.19: History (Rate and Review) 167

Figure 6.20: Section Code (Server API) – Rate Review 167

Figure 6.21: Chat (Chat History List) 168

Figure 6.22: Section Code (Server API) – Retrieve Chat User List 169

Figure 6.23: Section Code (Front-end) – Retrieve Chat List Activity 170

Figure 6.24: Chat (Chat Box) 171

Figure 6.25: Section Code (Front-end) – Chat Box Activities Gifted Chat 171

Figure 6.26: Section Code (Front-end) – Retrieve Chat Activity 172

Figure 6.27: Section Code (Front-end) – Send Notification Activity 172

Figure 6.28: Search (Input and Result) 173

Figure 6.29: Section Code (Server API) – Similarity Measure Part 1 174

Figure 6.30: Section Code (Server API) – Similarity Measure Part 2 175

Figure 6.31: Section Code (Server API) – Similarity Measure Part 3 176

Figure 6.32: Section Code (Server API) – Similarity Measure Part 4 176

Figure 6.33: Section Code (Server API) – Similarity Measure Part 5 177

Figure 6.34: Section Code (Server API) – Similarity Measure Part 6 177

Figure 6.35: Section Code (Server API) – Similarity Measure Part 7 178

Figure 6.36: Section Code (Server API) – Similarity Measure Part 8 178

Figure 6.37: Section Code (Server API) – Similarity Measure Part 9 179

Figure 6.38: Section Code (Server API) – Similarity Measure Part 10 180

Figure 6.39: Search (View Tutor Profile) 181

Figure 6.40: Section Code (Server API) – Retrieve Rate Review 182

Figure 6.41: Section Code (Server API) – Retrieve Profile Information 183

Figure 6.42: Search (Demo and Formal Class Request Status) 184

Figure 6.43: Section Code (Server API) – Register Demo Class Part 1 185

Figure 6.44: Section Code (Server API) – Register Formal Class 186

Figure 6.45: Profile (Display Profile, Edit Profile, Sign Out Account) 187

Figure 6.46: Section Code (Server API) – Update Basic Profile 187

Figure 6.47: Section Code (Front-end) – Sign Out Account 187

Figure 6.48: Section Code (Server API) – Delete Account 188

Figure 6.49: Section Code (Front-end) – Delete Account 188

Figure 6.50: Home (Registered Tutor) 189

Figure 6.51: Demo Class (Display and Manage Demo Class Requests) 190

Figure 6.52: Section Code (Server API) – Update Demo Class Status 190

Figure 6.53: Section Code (Server API) – Retrieve Demo Class 191

Figure 6.54: Formal Class (Display Formal Class Requests) 192

Figure 6.55: Section Code (Server API) – Retrieve Formal Class 193

Figure 6.56: Chat (Registered Tutor) 194

Figure 6.57: Registered Tutor Profile (Profile Display and Edit) 194

Figure 6.58: Section Code (Front-end) – Edit Tutor Profile Part 1 195

Figure 6.59: Section Code (Front-end) – Edit Tutor Profile Part 2 195

Figure 6.60: Section Code (Front-end) – Edit Tutor Profile Part 3 196

Figure 6.61: Section Code (Server API) – Update Tutor Basic Information 196

Figure 6.62: Home (Admin) 197

Figure 6.63: Add Tutor 198

Figure 6.64: Section Code (Front-end) – Create Tutor Information 199

Figure 6.65: Section Code (Server API) – Create Tutor Information 200

Figure 6.66: Update Tutor 201

Figure 6.67: Section Code (Server API) – Retrieve User List 202

Figure 6.68: Section Code (Server API) – Update Professional Tutor Info 202

Figure 6.69: User List 203

Figure 7.1: API Testing – Retrieve User Basic Profile 214

Figure 7.2: API Testing – Retrieve Profile Info 215

Figure 7.3: API Testing – Retrieve Class 216

Figure 7.4: API Testing – Retrieve Popular Tutor 216

Figure 7.5: API Testing – Retrieve Rate Review 217

Figure 7.6: API Testing – Retrieve Tutor Formal Class 217

Figure 7.7: API Testing – Retrieve Tutor Demo Class 217

Figure 7.8: API Testing – Retrieve User List 218

Figure 7.9: API Testing – Retrieve Chat User List 218

Figure 7.10: API Testing – Delete Account 219

Figure 7.11: API Testing – Similarity Measure (Manhattan) 219

Figure 7.12: API Testing – Similarity Measure (Euclidean) 220

Figure 7.13: API Testing – Similarity Measure (Minkowski) 220

Figure 7.14: API Testing – Similarity Measure (Jaccard) 221

Figure 7.15: API Testing – Similarity Measure (Cosine) 221

Figure 7.16: API Testing – Register Demo Class 222

Figure 7.17: API Testing – Register Formal Class 222

Figure 7.18: API Testing – Rate Review 222

Figure 7.19: API Testing – Register Basic Account 223

Figure 7.20: API Testing – Update Basic Profile 223

Figure 7.21: API Testing – Update Demo Class Status 223

Figure 7.22: API Testing – Update FCM Token 224

Figure 7.23: API Testing – Update Basic Tutor Information 224

Figure 7.24: API Testing – Update Professional Tutor Information 224

Figure 7.25: API Testing – Create Tutor Information 225

Figure 7.26: User Satisfaction Survey Form – Tutor Seeker (Part 1) 233

Figure 7.27: User Satisfaction Survey Form – Tutor Seeker (Part 2) 234

Figure 7.28: User Satisfaction Survey Form – Admin and Registered Tutor 235

Figure 7.29: SUS of Participant on Application 236

Figure 7.30: SUS of Participant on Application 236

LIST OF SYMBOLS / ABBREVIATIONS

API	Application Programming Interface
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
KNN	K-Nearest Neighbours
RAD	Rapid Application Development
SDLC	Software Development Life Cycle
SQL	Structured Query Language
SUS	System Usability Scale
UAT	User Acceptance Testing

LIST OF APPENDICES

APPENDIX A: Questionnaire 247

APPENDIX B: Gantt Chart 252

APPENDIX C: Usability Test Result 256

APPENDIX D: User Acceptance Testing Results 268

CHAPTER 1

INTRODUCTION

1.1 Introduction

In this project, an intelligent mobile private tutor finder's application is developed to help the students and parents in finding their ideal tutors within a short period. Prototyping methodology is applied as the software development methodology while React Native, Express.js, Firebase as the main development tools in this project.

This chapter provides an overview of the intelligent mobile private tutor finders application project that included the project background, project objectives, project problem statements, project solution, project approach, and the scope of the project.

Private tutoring can be defined as fee-charging services that provide educational assistance outside the schooling system in teaching academics subjects. It could be performed via various forms such as one-to-one tutoring in-home, small or large groups tutoring in a class, online tutoring via the internet, and so on (Bray, 1999). According to the official statistic from the Department of Statistics Malaysia (DOSM), there are 1899 tuition centers was established in Malaysia in 2010 (Department of Statistics Malaysia, 2021). Besides, the number of registered private tuition centers had increased to 3107 in Malaysia during 2013 (Kenayathulla and Ubbudari, 2017). This shows that the demand for the private tutoring service is increasing. Besides, the demand for private tutoring services will be elevated day by day due to the changes in the Malaysia examination and assessment system. For example, the Pentaksiran Tingkatan Tiga (PT3) and Pentaksiran Berasaskan Sekolah (PBS) that was introduced a few years ago to replace the old examination and assessment system. Both systems not only focus on the academics of the student as the old system but it focuses on both academic and non-academic skills such as oral communication skills and critical thinking skills (The Government of Malaysia, 2019). Therefore, it is a new challenge for the student to fit themselves into a new examination and assessment skills within a short period. Private tutors played the main role in helping the student to cope with this challenge. This is because schoolteachers are always responsible to teach many classes and subjects instead of focusing on a particular group of students.

Student and parent in Malaysia commonly find their private tutor through the introduction of their friends or relatives due to the lack of a user-friendly platform to find a suitable private tutor. In addition, it is a challenge to find a private tutor during this COVID-19 pandemic because it has a high risk to meet with each of the tutors for choosing the most matched private tutor (OECD, 2020). However, there are fewer private tutor web and mobile applications with intelligent matching functions are available in Malaysia cause students and parents spend a long time finding the best private tutor.

Nowadays, the mobile application is widely used in different areas such as business, individual and social areas due to its user-friendliness, accessible anytime and anywhere (Islam, Islam and Mazumder, 2010). Therefore, a user-friendly intelligent private tutor finder mobile application is needed instead of a web application to tackle this issue. The mobile application should consist of an intelligent matching feature that helps the student find the most matched tutors within a short period and is easy to be used. This application is built and implemented on only the Android platform. Besides, more criteria are needed in the filtering function instead of only basic criteria that may help the user to filter the private tutor in a more precise and accurate way.

1.2 Problem statement

i. Lack of private tutor finder mobile application in Malaysia

Nowadays, the mobile application is widely used in different areas such as business, individual and social areas due to its user-friendliness, accessible at anytime and anywhere. The use of mobile applications is increasing day by day corresponding to desktop applications (Islam, Islam and Mazumder, 2010). However, there is very limited private tutor finder mobile application that is applicable for the Malaysia area are available at Play Store and App Store. Most of the private tutor finder platforms in Malaysia such as MyPrivateTutor, TeachMe and Tuition Hero Malaysia (tutor seeker side) remain as web applications while some students find their private tutor via their friends or relatives. Hence, these will cause students and parents to spend more time finding a suitable private tutor. In contrast, there are many popular private tutor finder mobile applications available in other countries such as Stapps from Hong Kong and Zhang Men in China. Finding a private tutor by using a mobile application became a norm in their countries.

ii. Users hard in finding a suitable private tutor with limited features filter functioning

By referring to some of the existing private tutor finder applications in Malaysia that are shown in chapter 2, most of the applications filter the tutor by just using a few basic criteria such as subject, standard, and area. For example, TeachMe only provides teaching subjects and teaching standards in filtering and searching the tutors. These basic criteria are not the best way in choosing a best-matched tutor for a student. There are some important criteria that should be considered while finding a suitable private tutor such as gender, race, teaching languages, teaching experiences, and so on. For example, teaching a student with his or her mother tongue will cause the student to learn more effectively compare to use the language that the student is not familiar with (Nishanthi, 2020). Besides the student capable of fully understanding the explanation of tutors, this will also contribute to the student can express themselves clearly when facing some doubts (Nishanthi, 2020). In contrast, there are more features applied in the filtering function in Stapps (tutor finder mobile application in Hong Kong) and it provides an AI tutor matching function to help the student find the best matching tutor.

iii. Limitation of rules-based SQL query

Structured Query Langue (SQL) Query is widely applied to access and manage the database that is connected with the application or system (Myalapalli and Teja, 2015). It is also applicable to the simple searching function of a system. However, the limitation of SQL query is a problem in the tutor searching function. A rules-based SQL query only perform an “IF” “THEN” logic that is instructed by developers which is not applicable in handling high volume of complicated rules (Carew, 2020). For example, query retrieves the data according to the “WHERE” clause with the condition while the condition could be the “AND” or “OR” clause. By applying all “AND” clauses in the conditions, it may cause the issue of getting less or no result return due to no tutors to fulfill all the selected criteria. By applying all “OR” clauses in the condition, it may cause the issue of getting irrelevant result returns due to tutors that only fulfill fewer criteria being considered as a matched result. The using od “AND” and “OR” clauses are hard to be measured to get a high accuracy to result in the tutor searching function. In addition, large queries will consume many memories that affect the application's performance (Elastic NV, 2021). Therefore, SQL queries limit the searching function of tutor finder applications and become a problem that should be solved by applying other algorithms.

1.3 Project Objectives

1. To develop a mobile application in helping students and parents find their ideal tutors according to their learning preferences.
2. To provide a better platform for finding ideal tutors by applying similarity measures according to the learning preferences of students and parents.
3. To evaluate the usability of the private tutor finder mobile application by using usability testing in System Usability Scale (SUS).

1.4 Project Solution

As the problem statement stated above, there is a lack of mobile tutor searching platform and incompleteness of tutor searching features due to few searching criteria and limitation of SQL query in existing Malaysia mobile tutor searching platform. These cause students and parents that could not find their ideal private tutors, especially during this Covid-19 pandemic period.

This intelligent mobile private tutor finders application provides a new mobile platform to users while also allowing the students or parents to search for their ideal tutors according to more criteria such as teaching style, subjects, area, and so on. The similarity measures are applied in this project as the algorithm to measure the similarity between the selected criteria and the criteria of registered tutors such as Euclidean distance, Manhattan distance, Minkowski distance, Cosine similarity, and Jaccard similarity coefficient. The similarity measures algorithm could tackle the limitation of SQL query that can display tutors according to similarity instead of exact matching (Polamuri, 2015).

In this project, React Native is applied to develop the user interface as the frontend displays to the user. React Native is a JavaScript framework that runs on an open-source library called React and it is a popular open-source framework that is widely applied in developing a native-rendered mobile application (Netguru, 2021). The frontend sends tutor selection criteria to the server (Express.js is used to create the API within a server in supporting the backend logic and processing such as similarity measures) via the HTTP request to prevent direct accessing the backend processing and database table via the frontend and prevent task overloading at the frontend. The server retrieves the tutors' details from the cloud database (Firestore). The similarity measure is performed in a back-end calculation by calling the API. API returns the top 20 tutors' information with the highest similarity percentage in JSON form to the frontend. Then, the list of tutors will be displayed for users as the search result. The algorithm will increase the accuracy in finding an ideal tutor for the students or parents. Students or parents could look through the profiles of tutors and chat with tutors to discuss the details such as price or time. make a demo class request or book the tutor's formal class after attending the demo class.

The application shows the demo class requests on the user interface of tutors. Tutors could discuss details of class with students or parents by using the chat function in the application. Then, the tutor could decide to accept or decline the demo class requests. The application displays the result of the request to students or parents after the tutors decide. This module helps the student to more understand the teaching style of the tutors before hiring the tutors as their private tutors.

The application only allows the admin to add new tutors and update their details to the database after verification to ensure the quality of tutors. Admin could view the user list by using the application without accessing the back-end database.

The overview architecture of this application is shown in Figure 1.1. As mentioned above, React Native is applied as the frontend development tool while Express.js is used to create APIs within the server (laptop) to process the backend logic actions. Firestore is linked to Express.js for CRU actions and as a cloud database for storing application data such as tutors' information, tutor seekers' information, rate and reviews, chats, and so on. Besides, the Firebase Cloud Messaging service is linked with the application in order to perform push notification action. Firebase Authentication is applied in handling the sign-in and sign-up actions in this application while Firebase Storage is used as the photo storage to store and upload the users' profile pictures.

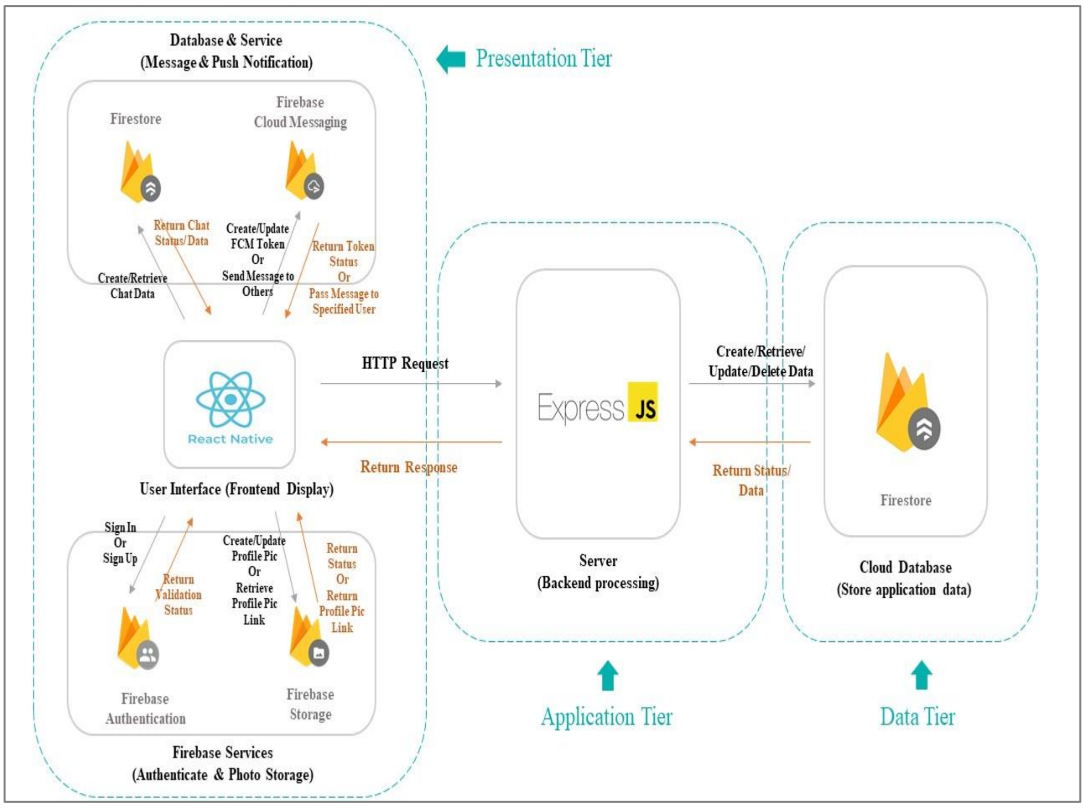


Figure 1.1: Overview of Application Architecture

1.5 Project Approach

Prototyping methodology is chosen as the development methodology in this project to develop the intelligent private tutor finder mobile application. Figure 1.2 shows the phases in the prototyping methodology model.

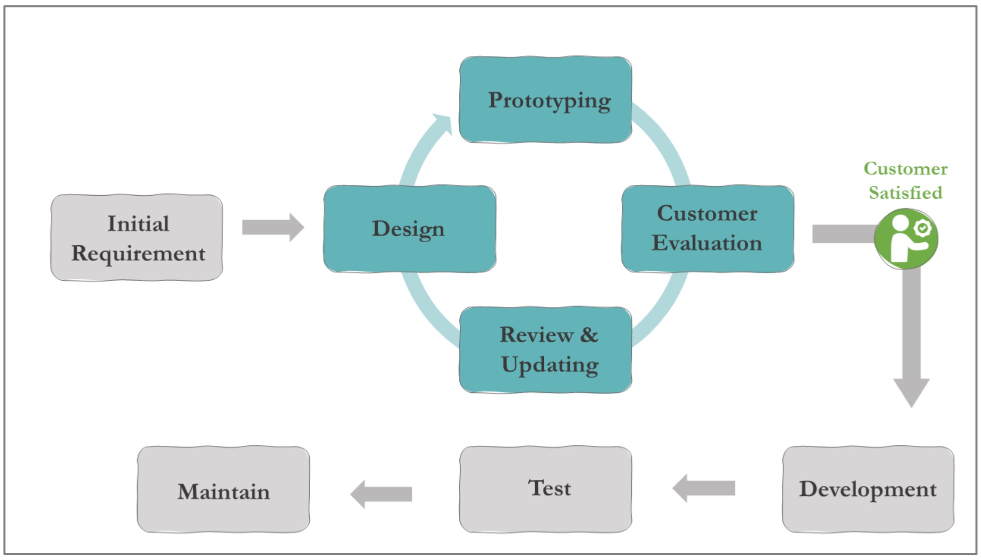


Figure 1.2: Prototyping Methodology

Prototyping methodology is widely applied in real software development projects because it helps developers to refine the software products. It needs minimum initial specifications in building the first prototype with essential functions quickly. Next, clients give feedback on the prototype and keep refining the prototypes. The iteration of the planning, design, prototype, and feedback phases will be stopped and enter the development phase when the client is satisfied with the prototype. It helps the users to identify their actual requirements throughout the iteration and build their ideal software (Susanto and Meiryani, 2019). This methodology is suitable to be applied in the projects that lack the previous example exists (Despa, 2014).

There is a lack of existing similar mobile applications available in PlayStore and AppStore. This caused the specifications of the application could be hardly defined. Prototyping methodology helps to refine the specifications and requirements after each iteration. Besides, the algorithms should be tested frequently to get the best result of matching. Therefore, space for improvement could be achieved in each version of the prototype to build an intelligent private tutor finder mobile application that helps the users in finding their ideal tutors with high accuracy.

1.6 Scope of the Project

1.6.1 Targeted Users

The targeted users of this intelligent private tutor finder mobile application are students who study in primary and secondary schools in the Klang Valley area. Students between 7 to 17 years old may lack experience in finding the best way in studying and learning, they need more guidance and assistance in their learning journey compared to the students who stepped into university. Besides, parents that help their children in finding a private tutor also be considered the main targeted user as well.

1.6.2 End Users Involved

Students, Parents, Tutors, and Admins are the end-users involved in this application. Students or parents could find their ideal tutors while tutors could get their jobs from this application. Admins are responsible to manage and update the registered tutors to this application.

1.6.3 Modules Covered

i. Searching tutors according preferences

The application allows the users (students or parents) to choose their preference criteria of tutors, the application will display the matched tutors in a list. Users could read the tutor's profile before proceeding to request demo classes and formal classes.

ii. Requesting demo classes

Tutor seekers (students or parents) could request a free demo class before applying for a formal class to more understand the teaching style of the tutor. Tutors could choose to accept or reject the demo class request by using the application.

iii. Rating and reviewing tutors

Tutor seekers (students or parents) could rate and give a review to the tutors by using the application. The rate and review will be displayed in the tutor's profile as a reference for the other tutor seekers to find their ideal tutors.

iv. Booking formal classes

Tutor seekers (students or parents) could book a formal class after attending the demo class and feel satisfied with the tutors' teaching style and method by using the application. The tutor will receive the booking by using the application.

v. Chatting between tutor seekers and tutors

Tutor seekers (students or parents) could chat with the tutors to discuss detailed information that is not listed in the tutor profile before and after raising the demo classes and formal classes requests. A tutor could discuss the class details with tutor seekers after the tutor seekers raise the demo classes or formal classes requests.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Five existing similar applications have been reviewed and compared about their functions. Some functions or features of similar applications could be the reference of this project. Next, searching approaches are reviewed and compared to identify the best approaches that could be applied in the searching function in this project. Besides, software development methodologies are studied and compared to apply the most suitable methodology in this project that may improve the development process and quality of products. Last but not least, the usability testing is reviewed to have a better understanding of the workflow of usability testing.

2.2 Study of Existing Similar Application

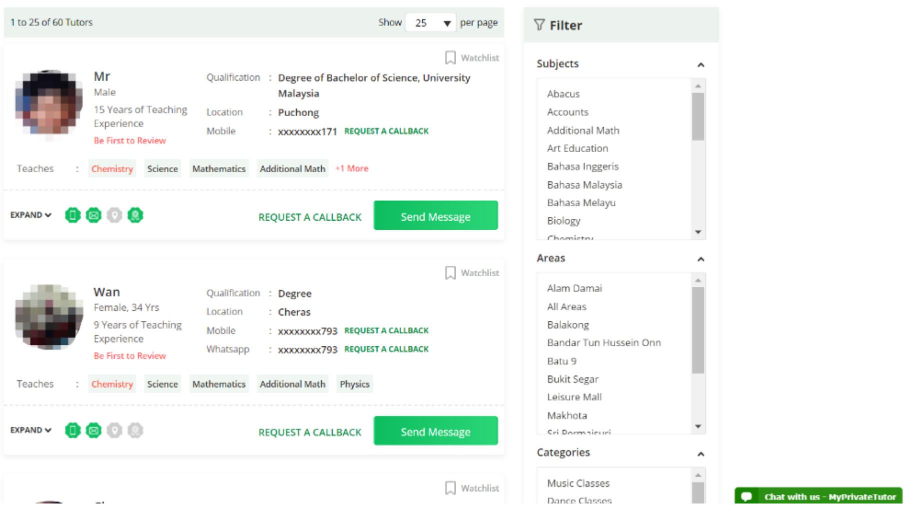
There are 5 existing similar applications that have been reviewed about the functions and services provided. MyPrivateTutor and TeachMe are web applications while TuitionHero is a cross-platform application (mobile and web platform) that is available in Malaysia. At the same time, there are 2 similar applications that are popular in Hong Kong and China which are Stapps (mobile application) and ZhangMen (cross-platform application) respectively.

2.2.1 MyPrivateTutor

MyPrivateTutor provides services to 3 types of end-users which are (parents and students), (tutors and trainers), and (center and training institute). MyPrivateTutor is a web application that provides the service of tutors and institutes finding to students and parents while job finding service to the tutors and institutes over 25 countries such as Malaysia, Thailand, Taiwan, Vietnam, United States (US), United Kingdom (UK) and so on. It is marketed by Softz Solutions & Co Pvt Ltd since 2009 and is available at https://www.myprivatetutor.my/ (Malaysia Area).

i. Tutors or Institutes Filtering

MyPrivateTutor allows the students or parents to find their ideal tutors or institutes by using browsing and filtering functions according to the simple criteria that have been selected by students and parents. For example, subjects, areas, and categories.



1 to 25 of 60 Tutors — Show 25 per page

Watchlist

Mr
Male
15 Years of Teaching Experience
Be First to Review

Qualification : Degree of Bachelor of Science, University Malaysia
Location : Puchong
Mobile : xxxxxxxx171 REQUEST A CALLBACK

Teaches : Chemistry Science Mathematics Additional Math +1 More

EXPAND — REQUEST A CALLBACK — Send Message

Watchlist

Wan
Female, 34 Yrs
9 Years of Teaching Experience
Be First to Review

Qualification : Degree
Location : Cheras
Mobile : xxxxxxxx793 REQUEST A CALLBACK
Whatsapp : xxxxxxxx793 REQUEST A CALLBACK

Teaches : Chemistry Science Mathematics Additional Math Physics

EXPAND — REQUEST A CALLBACK — Send Message

Watchlist

Filter

Subjects: Abacus, Accounts, Additional Math, Art Education, Bahasa Inggeris, Bahasa Malaysia, Bahasa Melayu, Biology

Areas: Alam Damai, All Areas, Balakong, Bandar Tun Hussein Onn, Batu 9, Bukit Segar, Leisure Mall, Makhota

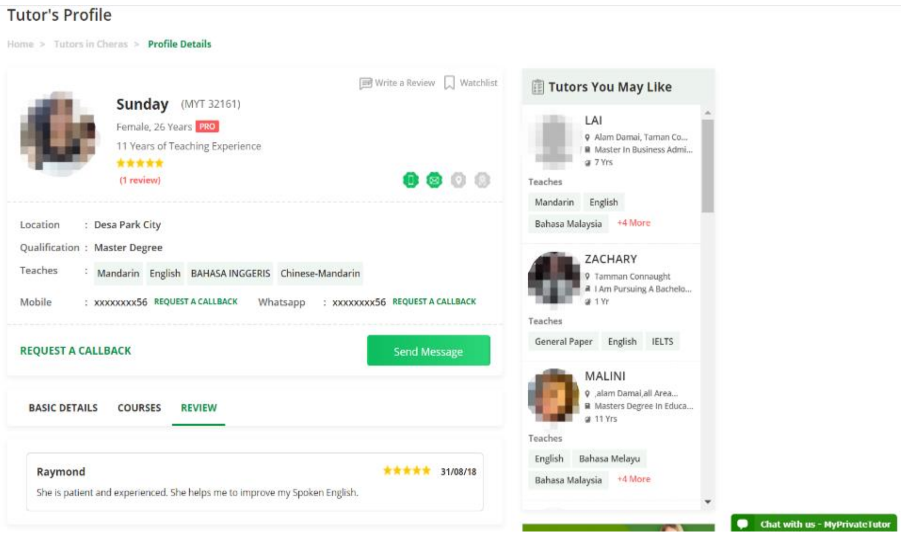
Categories: Music Classes

Chat with us - MyPrivateTutor

Figure 2.1: Filter Result of MyPrivateTutor

ii. Tutors Profile and Reviews

Students and parents could give reviews and rates to tutors and institutes after attending class. The reviews and rates will be displayed in the profile of the tutors and institute with their basic information and details.



Tutor's Profile

Home > Tutors in Cheras > Profile Details

Write a Review — Watchlist

Sunday (MYT 32161)
Female, 26 Years PRO
11 Years of Teaching Experience
★★★★★
(1 review)

Location : Desa Park City
Qualification : Master Degree
Teaches : Mandarin English BAHASA INGGERIS Chinese-Mandarin
Mobile : xxxxxxxx56 REQUEST A CALLBACK — Whatsapp : xxxxxxxx56 REQUEST A CALLBACK

REQUEST A CALLBACK — Send Message

BASIC DETAILS — COURSES — REVIEW

Raymond ★★★★★ 31/08/18
She is patient and experienced. She helps me to improve my Spoken English.

Tutors You May Like

LAI
Alam Damai, Taman Co...
Master In Business Admi...
7 Yrs
Teaches: Mandarin English Bahasa Malaysia +4 More

ZACHARY
Tamman Connaught
I Am Pursuing A Bachelo...
1 Yr
Teaches: General Paper English IELTS

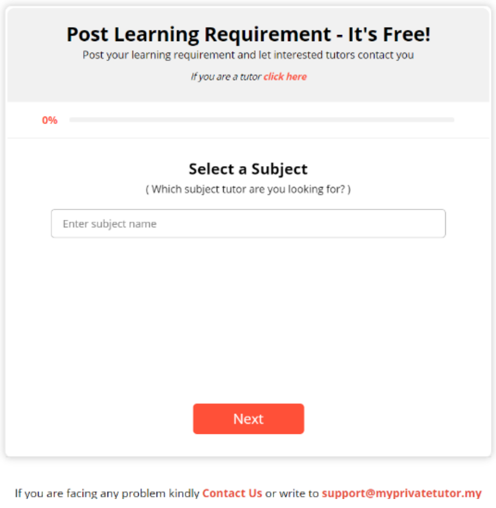
MALINI
,alam Damai,all Area...
Masters Degree In Educa...
11 Yrs
Teaches: English Bahasa Melayu Bahasa Malaysia +4 More

Chat with us - MyPrivateTutor

Figure 2.2: Tutor Profile of MyPrivateTutor

iii. Tutors Requesting

Students and parents can make a tutor request on MyPrivateTutor by listing the learning requirements and waiting for the application of tutors. The learning requirements are basic criteria such as subject, area, type of tutors (online tutor, private tutor, tuition center, and so on), introduction (what have learned), and basic information (name, email address, mobile number)



Post Learning Requirement - It's Free!

Post your learning requirement and let interested tutors contact you

If you are a tutor click here

0%

Select a Subject

(Which subject tutor are you looking for?)

Enter subject name

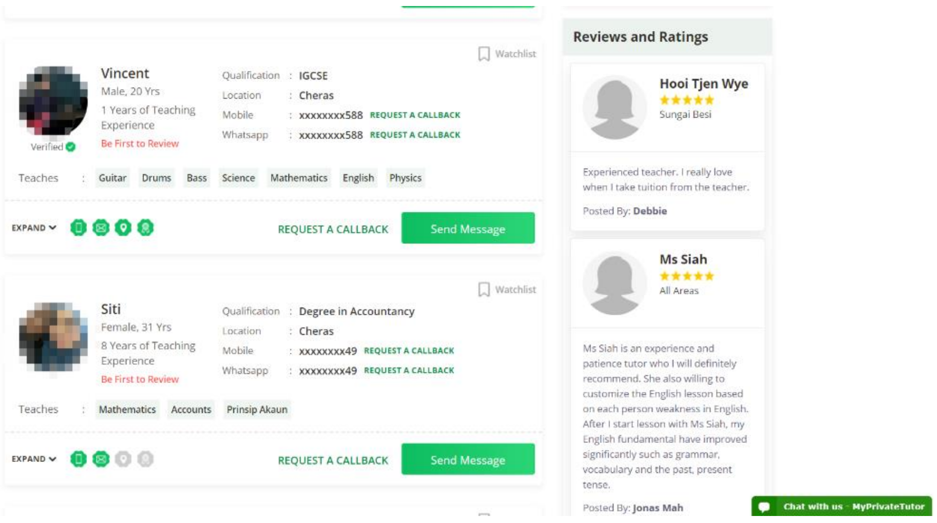
Next

If you are facing any problem kindly Contact Us or write to support@myprivatetutor.my

Figure 2.3: Tutor Request of MyPrivateTutor

iv. Tutors and Institutes Suggestion

MyPrivateTutor suggests the relevant tutors as the “Reviews and Ratings” column on the filter result page (Figure 2.4) and as “Tutors You May Like” on the tutor profile page (Figure 2.2).



Watchlist

Vincent
Male, 20 Yrs
1 Years of Teaching Experience
Verified
Be First to Review

Qualification	:	IGCSE
Location	:	Cheras
Mobile	:	xxxxxxxx588 REQUEST A CALLBACK
Whatsapp	:	xxxxxxxx588 REQUEST A CALLBACK

Teaches : Guitar Drums Bass Science Mathematics English Physics

EXPAND REQUEST A CALLBACK Send Message

Watchlist

Siti
Female, 31 Yrs
8 Years of Teaching Experience
Be First to Review

Qualification	:	Degree in Accountancy
Location	:	Cheras
Mobile	:	xxxxxxxx49 REQUEST A CALLBACK
Whatsapp	:	xxxxxxxx49 REQUEST A CALLBACK

Teaches : Mathematics Accounts Prinsip Akaun

EXPAND REQUEST A CALLBACK Send Message

Reviews and Ratings

Hooi Tjen Wye
★★★★★
Sungai Besi

Experienced teacher. I really love when I take tuition from the teacher.

Posted By: **Debbie**

Ms Siah
★★★★★
All Areas

Ms Siah is an experience and patience tutor who I will definitely recommend. She also willing to customize the English lesson based on each person weakness in English. After I start lesson with Ms Siah, my English fundamental have improved significantly such as grammar, vocabulary and the past, present tense.

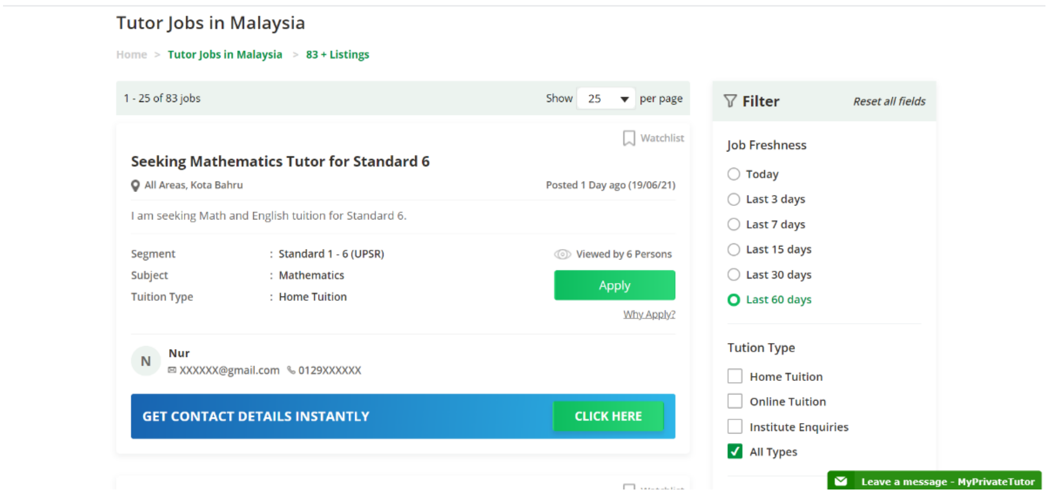
Posted By: **Jonas Mah**

Chat with us · MyPrivateTutor

Figure 2.4: Tutor Suggestion of MyPrivateTutor

v. Jobs Applying

Besides waiting for the job requests from students or parents, tutors can also apply for the tutoring jobs that were posted on MyPrivateTutor and wait for the reply of the students and parents.


Tutor Jobs in Malaysia

Home > Tutor Jobs in Malaysia > 83 + Listings

1 - 25 of 83 jobs Show 25 per page

Watchlist

Seeking Mathematics Tutor for Standard 6

All Areas, Kota Bahru Posted 1 Day ago (19/06/21)

I am seeking Math and English tuition for Standard 6.

Segment : Standard 1 - 6 (UPSR)
Subject : Mathematics
Tuition Type : Home Tuition

Viewed by 6 Persons

Apply

Why Apply?

N Nur
XXXXXX@gmail.com 0129XXXXXX

GET CONTACT DETAILS INSTANTLY CLICK HERE

Filter Reset all fields

Job Freshness
- Today
- Last 3 days
- Last 7 days
- Last 15 days
- Last 30 days
- [✓ Last 60 days]

Tution Type
- Home Tuition
- Online Tuition
- Institute Enquiries
- [✓ All Types]

Leave a message - MyPrivateTutor

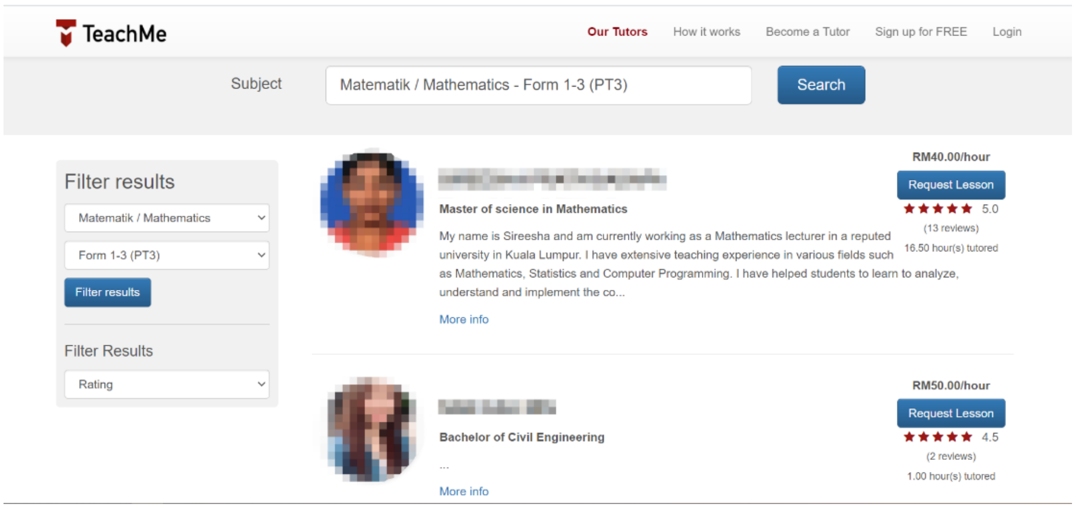
Figure 2.5: Job Applying of MyPrivateTutor

2.2.2 TeachMe

TeachMe provides services to 2 types of end-users which are (parents and students) and tutors. TeachMe is a web application that provides tutors finding and online tutoring services to students and parents while class managing service to the tutors in Malaysia. It is released in Malaysia in 2017 and available at https://teachme.com.my/

i. Tutors Filtering

TeachMe allows the students and parents to find their ideal tutors by using searching and filtering functions according to the basic criteria that have been searched and selected by students and parents which are subjects and grades. The tutor list is displayed either in a name or rating order.


TeachMe Our Tutors How it works Become a Tutor Sign up for FREE Login

Subject Matematik / Mathematics - Form 1-3 (PT3) Search

Filter results
Matematik / Mathematics
Form 1-3 (PT3)
Filter results

Filter Results
Rating

[redacted]
Master of science in Mathematics
My name is Sireesha and am currently working as a Mathematics lecturer in a reputed university in Kuala Lumpur. I have extensive teaching experience in various fields such as Mathematics, Statistics and Computer Programming. I have helped students to learn to analyze, understand and implement the co...
More info

RM40.00/hour
Request Lesson
★★★★★ 5.0
(13 reviews)
16.50 hour(s) tutored

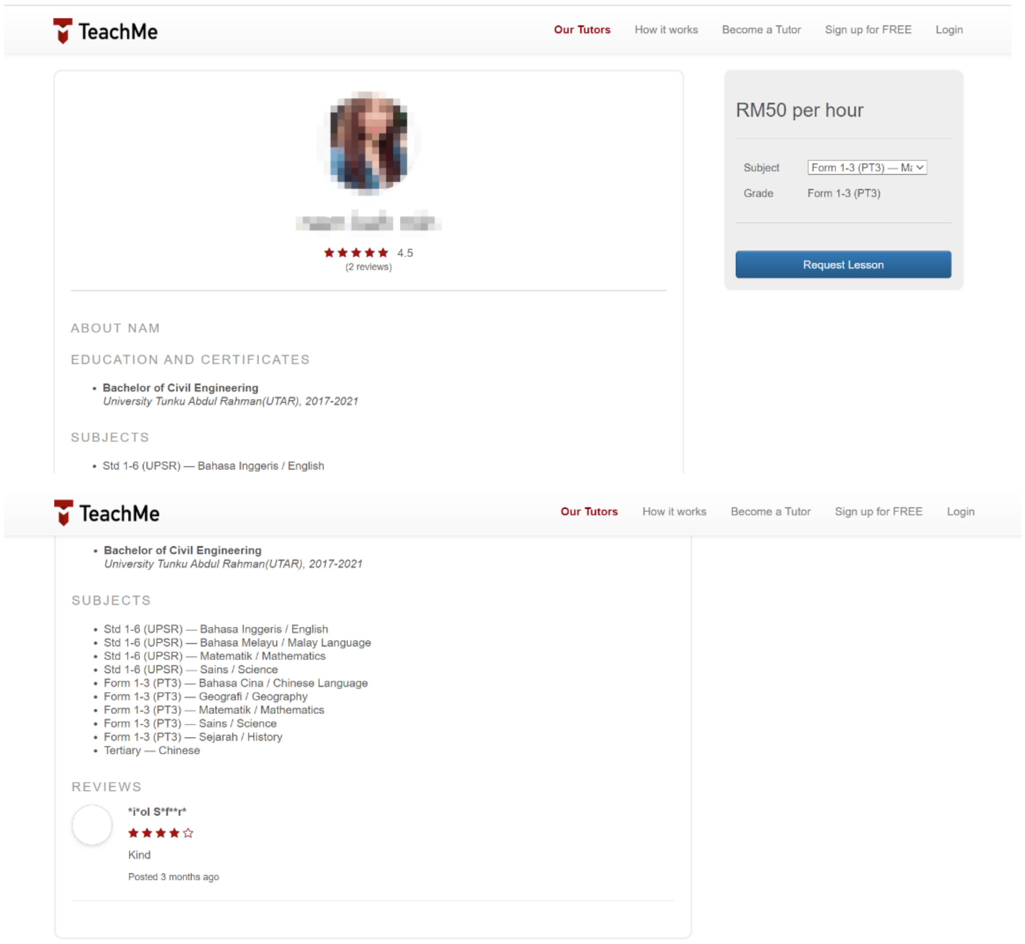
[redacted]
Bachelor of Civil Engineering
...
More info

RM50.00/hour
Request Lesson
★★★★★ 4.5
(2 reviews)
1.00 hour(s) tutored

Figure 2.6: Filter Result of TeachMe

ii. Tutors Profile and Reviews

Students and parents could give reviews and rates to tutors after attending the online class. There are basic information, reviews, and rates are displayed in the tutors' profile.



TeachMe — Our Tutors | How it works | Become a Tutor | Sign up for FREE | Login

[redacted]

★★★★★ 4.5
(2 reviews)

RM50 per hour

Subject: Form 1-3 (PT3) — Ma
Grade: Form 1-3 (PT3)

Request Lesson

ABOUT NAM

EDUCATION AND CERTIFICATES

- **Bachelor of Civil Engineering**
  University Tunku Abdul Rahman(UTAR), 2017-2021

SUBJECTS

- Std 1-6 (UPSR) — Bahasa Inggeris / English
- Std 1-6 (UPSR) — Bahasa Melayu / Malay Language
- Std 1-6 (UPSR) — Matematik / Mathematics
- Std 1-6 (UPSR) — Sains / Science
- Form 1-3 (PT3) — Bahasa Cina / Chinese Language
- Form 1-3 (PT3) — Geografi / Geography
- Form 1-3 (PT3) — Matematik / Mathematics
- Form 1-3 (PT3) — Sains / Science
- Form 1-3 (PT3) — Sejarah / History
- Tertiary — Chinese

REVIEWS

*i*ol S*f**r*
★★★★☆
Kind
Posted 3 months ago

Figure 2.7: Tutors Profile of TeachMe

iii. Online Class Tutoring

Besides the tutor finding function, TeachMe provides the service of online tutoring by using their platform instead of third party's platforms such as google meet, Microsoft team, and so on. The tutor's video is displayed on the left side while the whiteboard and tools are displayed on the right side. Both students and tutors share documents, edit the whiteboard, and export it as a note in a pdf format.

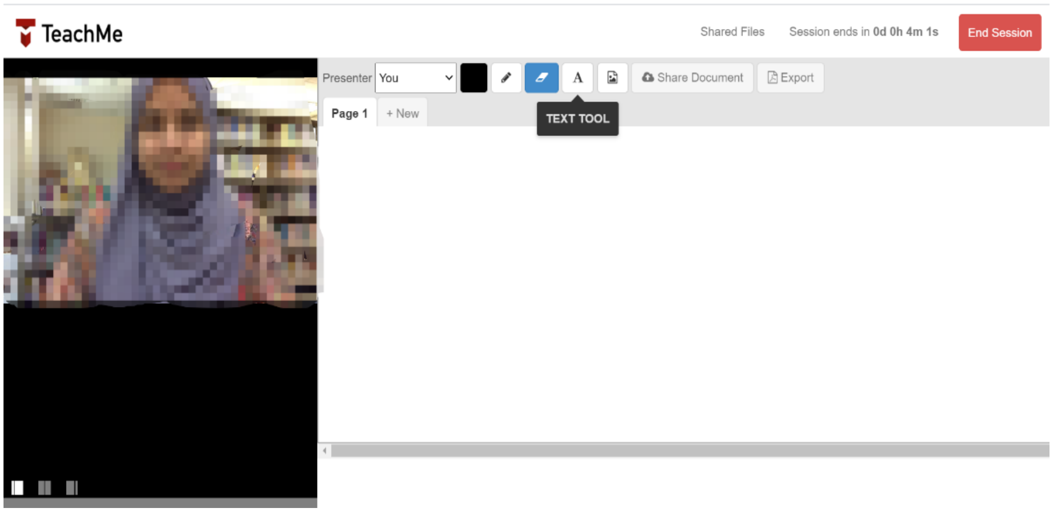


Figure 2.8: Online Tutoring Class Demo of TeachMe

iv. Class Managing

TeachMe provides some functions to help tutors in managing their tutorial classes and teaching materials. For example, “Availability” and “Lessons” is used to manage the class schedule of tutors and check the history and status of the upcoming, ended, canceled, and declined class respectively.

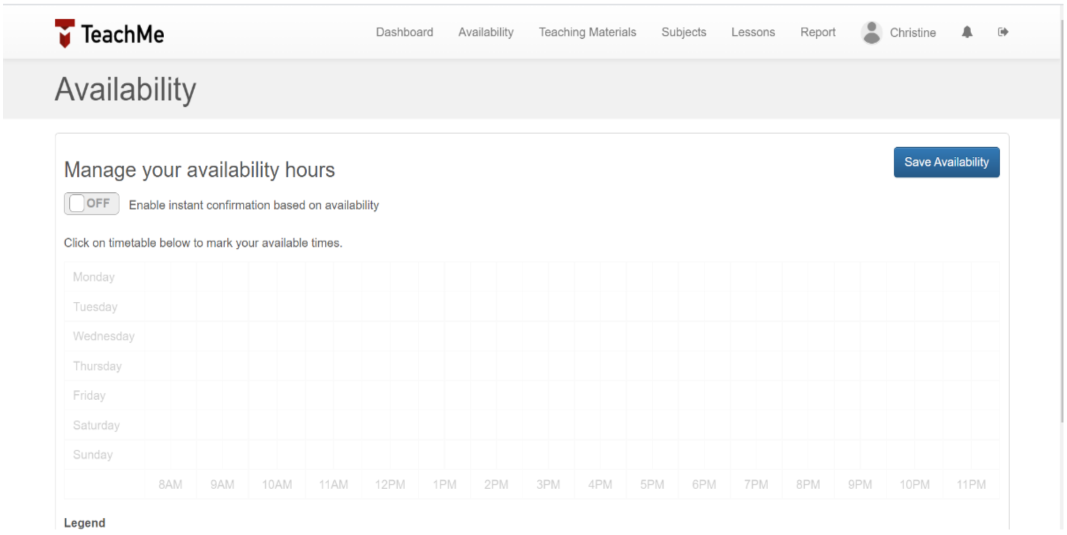
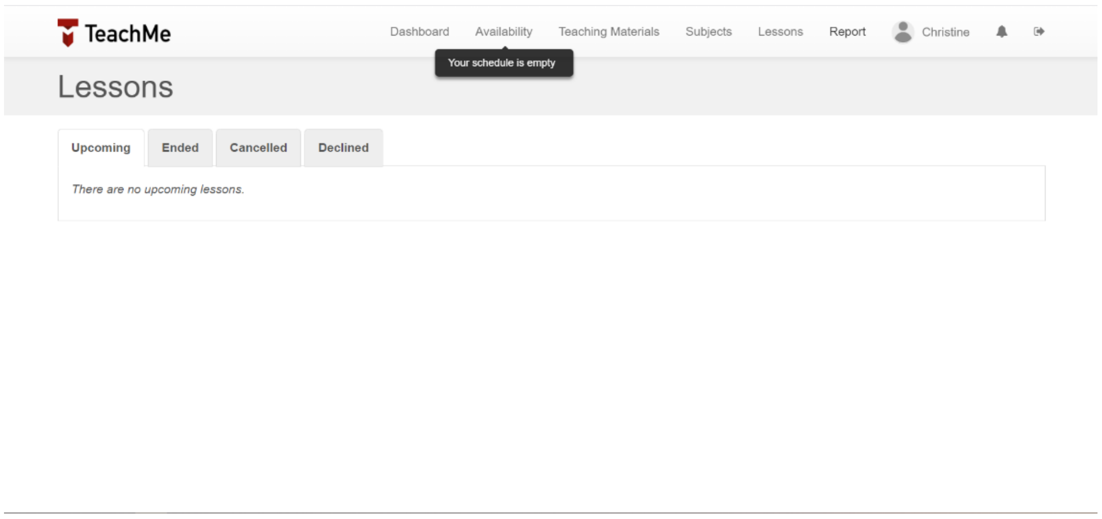
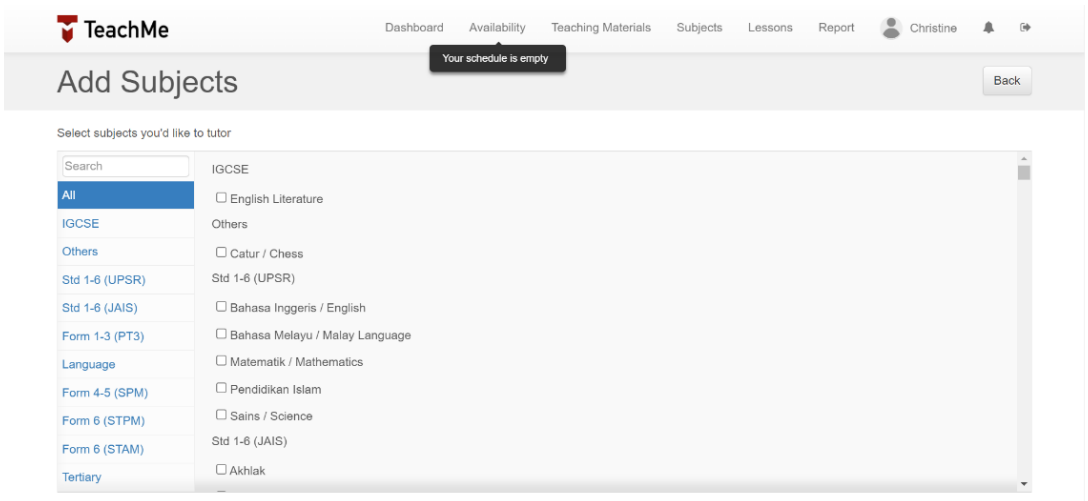


Figure 2.9: Tutor Availability of TeachMe



There are no upcoming lessons.

Figure 2.10: Tutor Lessons of TeachMe



Select subjects you'd like to tutor

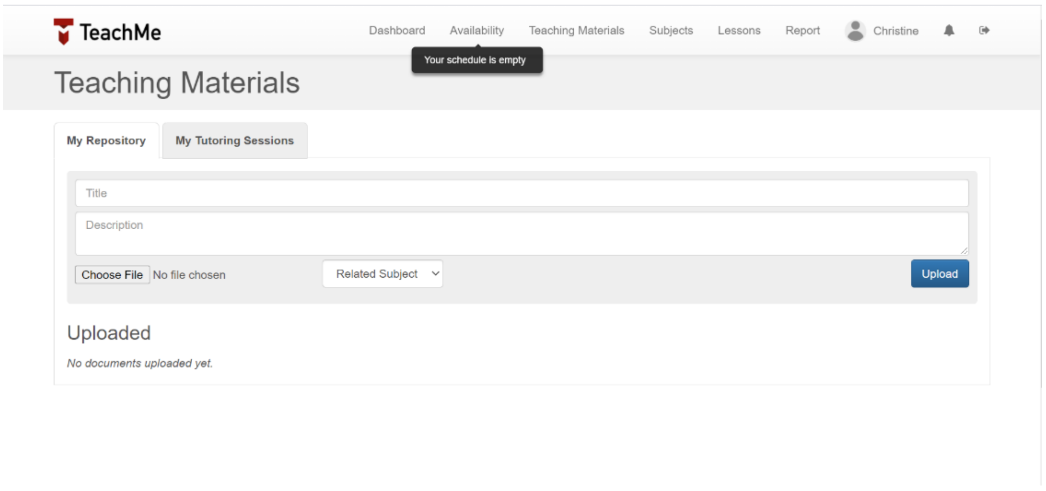
IGCSE
- ☐ English Literature

Others
- ☐ Catur / Chess

Std 1-6 (UPSR)
- ☐ Bahasa Inggeris / English
- ☐ Bahasa Melayu / Malay Language
- ☐ Matematik / Mathematics
- ☐ Pendidikan Islam
- ☐ Sains / Science

Std 1-6 (JAIS)
- ☐ Akhlak

Figure 2.11: Tutor Add Subjects of TeachMe



No documents uploaded yet.

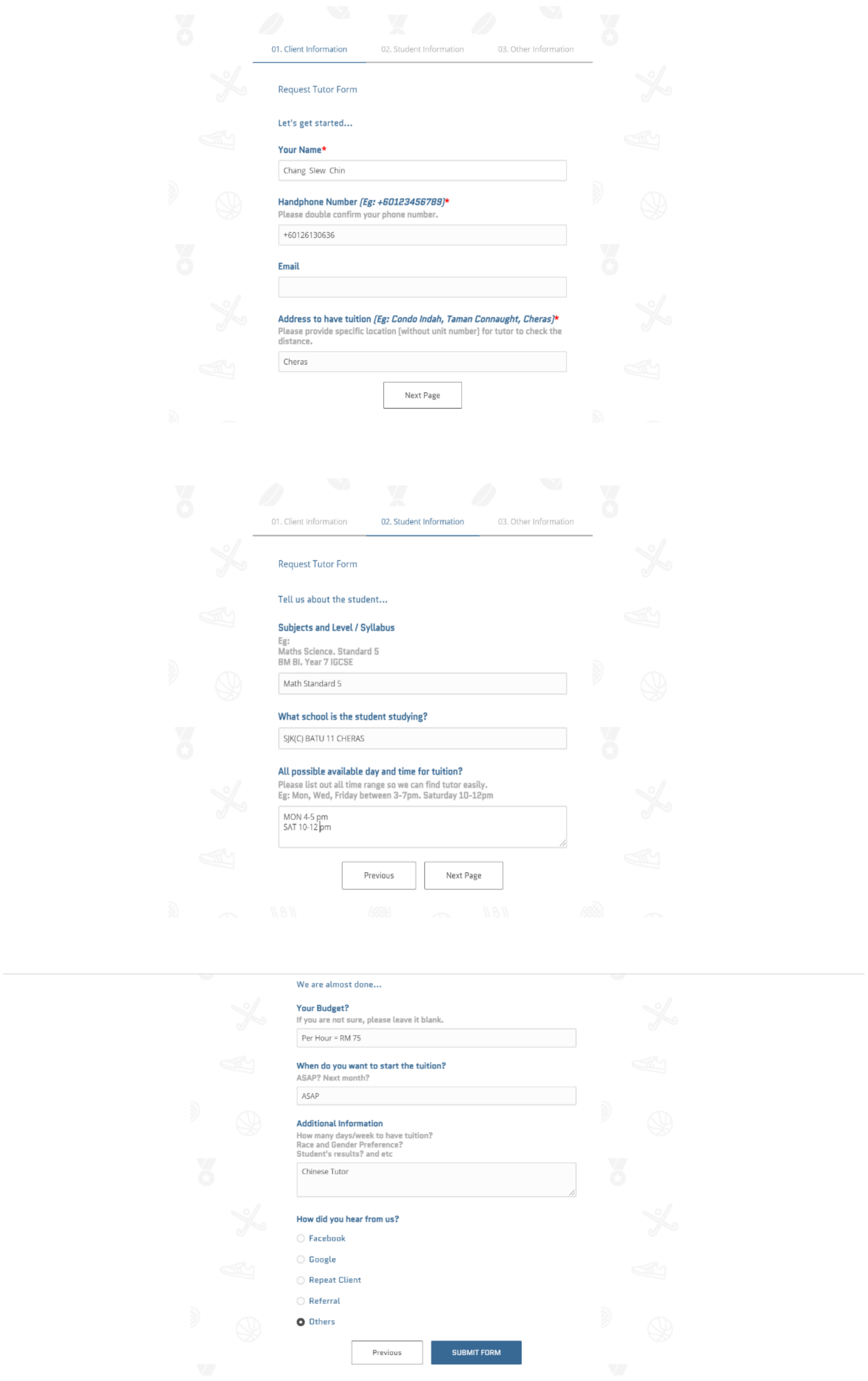
Figure 2.12: Tutor Teaching Materials of TeachMe

2.2.3 Tuition Hero Malaysia

Tuition Hero Malaysia provides services to 2 types of end-users which is (parents and students) and tutors. Tuition Hero Malaysia is a cross-platform application (web and mobile platform) that is owned by Hero Education Sdn Bhd which provide tutor matching service in Kuala Lumpur, Selangor, Seremban, Penang, and Johor only. Tuition Hero Malaysia mobile application is only served the job finding service to the tutors. At the same time, the Tuition Hero Malaysia web application provides both services of tutors finding to students and parents and job finding service to the tutors. The web application is available at https://tuitionhero.my/ since 2015 while the mobile application is available in Malaysia Play Store and AppStore since 2018.

i. Tutors Requesting

Students and parents can make a tutor request on Tuition Hero Malaysia by contacting the person in charge with WhatsApp or submitting the form that contains some basic information of client, student, and additional information such as budget, preference, and tuition start date. The requests will be displayed on the job list of the tutors' interface for applying.



Request Tutor Form

Let's get started...

Your Name*

Chang Siew Chin

Handphone Number *(Eg: +60123456789)**

Please double confirm your phone number.

+60126130636

Email

Address to have tuition *(Eg: Condo Indah, Taman Connaught, Cheras)**

Please provide specific location [without unit number] for tutor to check the distance.

Cheras

Request Tutor Form

Tell us about the student...

Subjects and Level / Syllabus

Eg:
Maths Science. Standard 5
BM BI. Year 7 IGCSE

Math Standard 5

What school is the student studying?

SJK(C) BATU 11 CHERAS

All possible available day and time for tuition?

Please list out all time range so we can find tutor easily.
Eg: Mon, Wed, Friday between 3-7pm. Saturday 10-12pm

MON 4-5 pm
SAT 10-12 pm

We are almost done...

Your Budget?

If you are not sure, please leave it blank.

Per Hour = RM 75

When do you want to start the tuition?

ASAP? Next month?

ASAP

Additional Information

How many days/week to have tuition?
Race and Gender Preference?
Student's results? and etc

Chinese Tutor

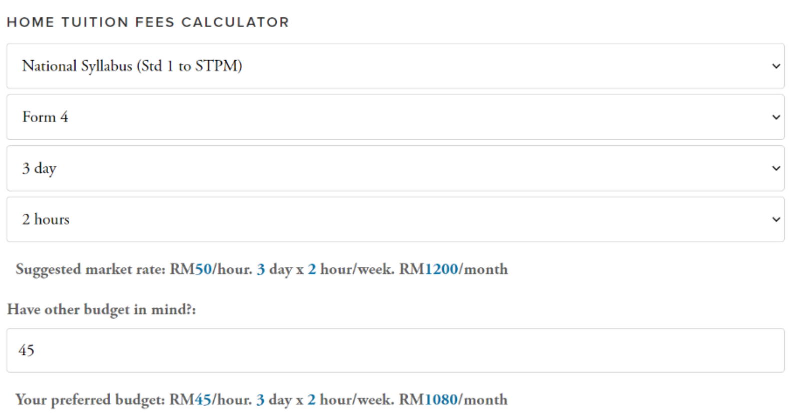
How did you hear from us?

- Facebook
- Google
- Repeat Client
- Referral
- [✓ Others]

Figure 2.13: Tutor Request Form of Tuition Hero Malaysia

ii. Home Tuition Fee Calculating

Tuition Hero Malaysia web application provides the function of calculating the suggested market rate of home tuition according to the syllabus standard, days per week, and the hours per day. Besides, it also calculates the home tuition fee according to the preferred budget that was provided.



HOME TUITION FEES CALCULATOR

National Syllabus (Std 1 to STPM)

Form 4

3 day

2 hours

Suggested market rate: RM50/hour. 3 day x 2 hour/week. RM1200/month

Have other budget in mind?:

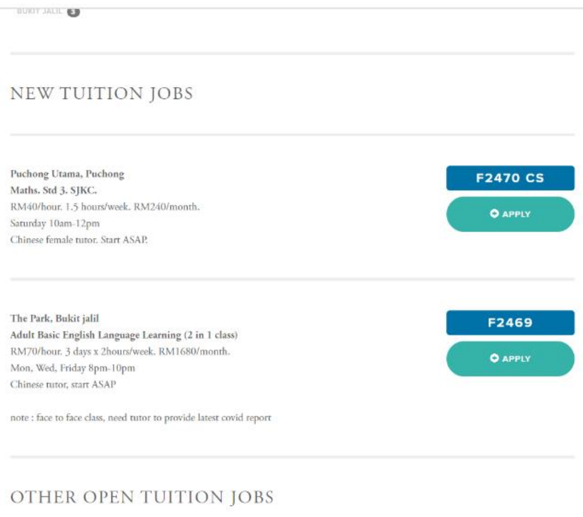
45

Your preferred budget: RM45/hour. 3 day x 2 hour/week. RM1080/month

Figure 2.14: Home Tuition Fee Calculator of Tuition Hero Malaysia

iii. Jobs Applying

Tutors can apply for the tutoring jobs that were posted on Tuition Hero Malaysia via the web application or mobile application and wait for the reply of the students and parents. Tutors can only filter the jobs by using the area.



NEW TUITION JOBS

Puchong Utama, Puchong
Maths. Std 3. SJKC.
RM40/hour. 1.5 hours/week. RM240/month.
Saturday 10am-12pm
Chinese female tutor. Start ASAP.

F2470 CS

APPLY

The Park, Bukit jalil
Adult Basic English Language Learning (2 in 1 class)
RM70/hour. 3 days x 2hours/week. RM1680/month.
Mon, Wed, Friday 8pm-10pm
Chinese tutor, start ASAP

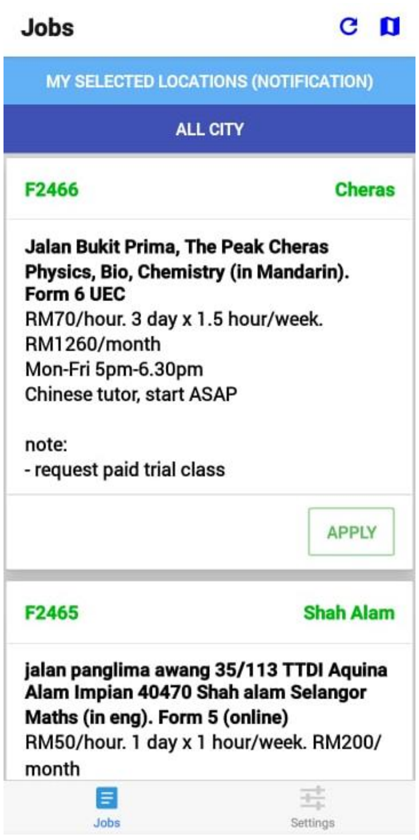
note : face to face class, need tutor to provide latest covid report

F2469

APPLY

OTHER OPEN TUITION JOBS

Figure 2.15: Job Applying of Tuition Hero Malaysia (Web Application)


Jobs

MY SELECTED LOCATIONS (NOTIFICATION)

ALL CITY

F2466 Cheras

Jalan Bukit Prima, The Peak Cheras
Physics, Bio, Chemistry (in Mandarin).
Form 6 UEC
RM70/hour. 3 day x 1.5 hour/week.
RM1260/month
Mon-Fri 5pm-6.30pm
Chinese tutor, start ASAP

note:
- request paid trial class

APPLY

F2465 Shah Alam

jalan panglima awang 35/113 TTDI Aquina
Alam Impian 40470 Shah alam Selangor
Maths (in eng). Form 5 (online)
RM50/hour. 1 day x 1 hour/week. RM200/
month

Figure 2.16: Job Applying of Tuition Hero Malaysia (Mobile Application)

2.2.4 Stapps

Stapps provides services to 2 types of end-users which are (parents and students) and tutors. Stapps is a mobile application owned and released by Stapps Limited in 2019 on Play Store and AppStore. It also can be downloaded via their official website and available at: https://www.stapps.hk/. Stapps provides tutor matching services in Hong Kong only. Students and parents could find their ideal tutors while tutors could apply for tutoring jobs by using this mobile application.

i. Tutors Filtering

Stapps allows the students and parents to find their ideal tutors by using sorting and filtering functions according to the criteria that have been selected by students and parents such as school type, subject, level, area, budget, tutor available time, tutor institute, tutor gender, and tutor teaching style. The tutor list is sorted and displayed in different order such as rates, academic performance, number of students, nearby area, and tutoring fee.

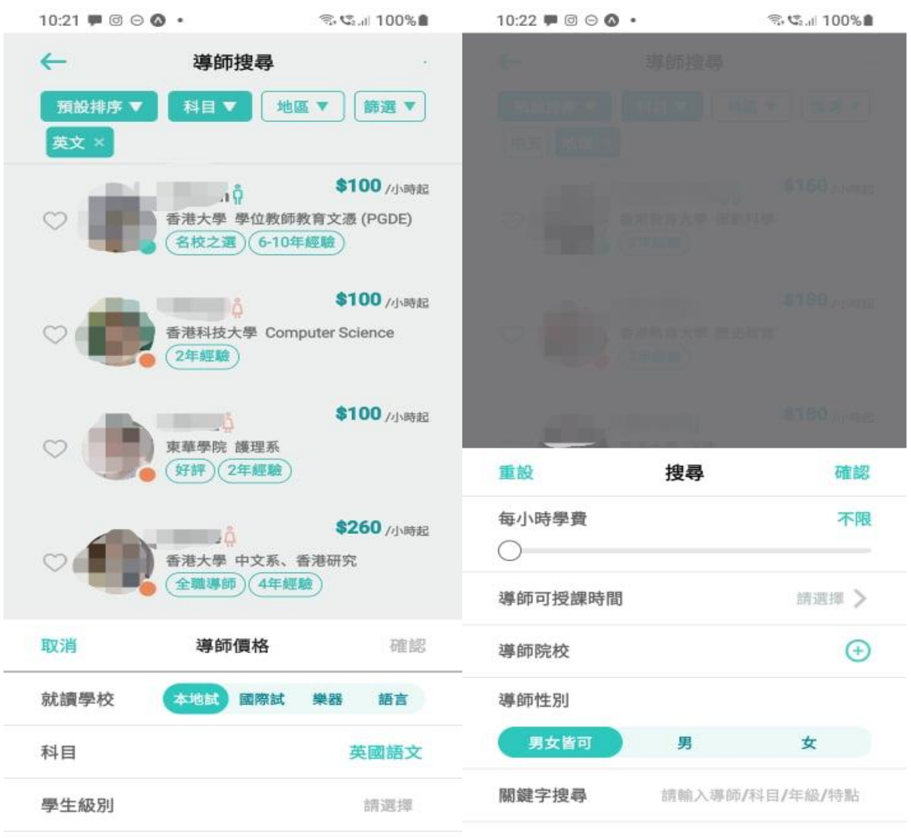


Figure 2.17: Tutor Filtering of Stapps

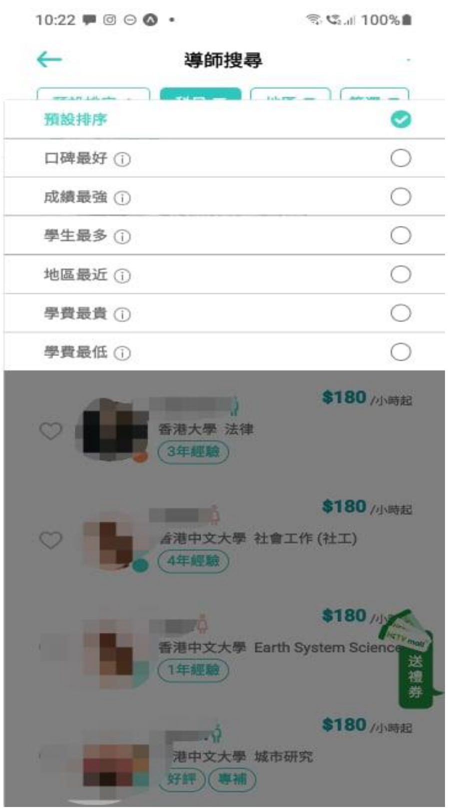
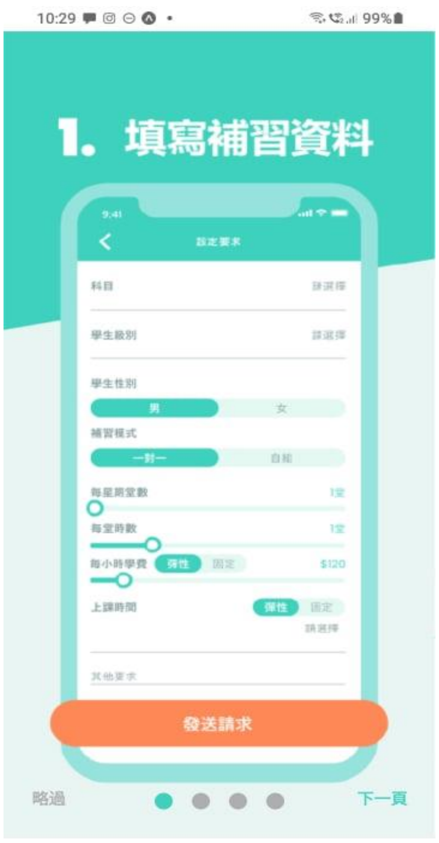


Figure 2.18: Tutor Sorting of Stapps

ii. Tutors Requesting

Students and parents can make a tutor request on Stapps by filling in the criteria of tutors and basic information of students such as subject, student level, student gender, tutoring type (one to one or group), number of classes per week, number of hours per class, tutoring fee per hour, class time slot and additional requirement. The requests will be displayed on the job list of the tutors' interface for applying.


10:29 99%

1. 填寫補習資料

9:41

設定要求

科目 請選擇

學生級別 請選擇

學生性別 [✓ 男] 女

補習模式 [✓ 一對一] 自組

每星期堂數 1堂

每堂時數 1堂

每小時學費 [✓ 彈性] 固定 $120

上課時間 [✓ 彈性] 固定 請選擇

其他要求

發送請求

略過 下一頁

Figure 2.19: Tutor Requesting of Stapps

iii. Tutors Suggestion

Stapps suggests the relevant tutors at the section below the tutor filtering and requesting section on the main page.

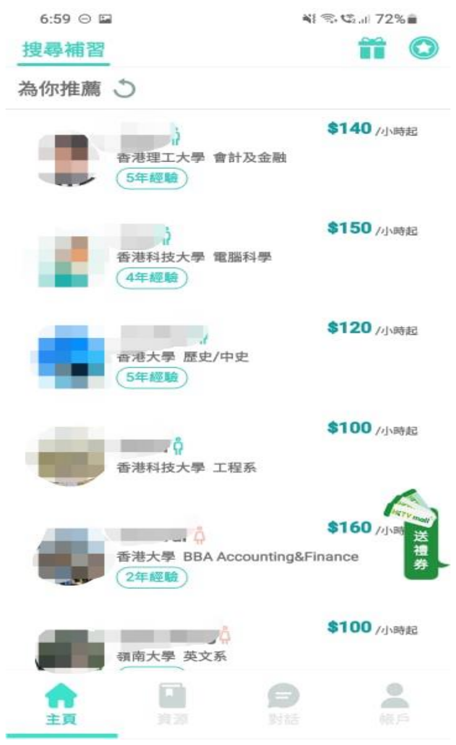

6:59 72%
搜尋補習
為你推薦
$140 /小時起
香港理工大學 會計及金融
5年經驗
$150 /小時起
香港科技大學 電腦科學
4年經驗
$120 /小時起
香港大學 歷史/中史
5年經驗
$100 /小時起
香港科技大學 工程系
$160 /小時起
香港大學 BBA Accounting&Finance
2年經驗
送禮券
$100 /小時起
嶺南大學 英文系
主頁 資源 對話 帳戶

Figure 2.20: Tutors Suggestion of Stapps

iv. Learning Materials Sharing

There is a source sharing function in Stapps that allows the tutors to upload free learning material such as tutoring videos, exercise papers, notes of different subjects.

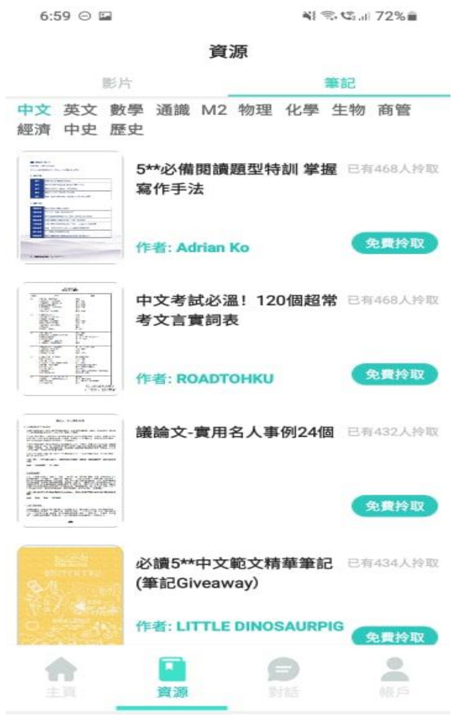
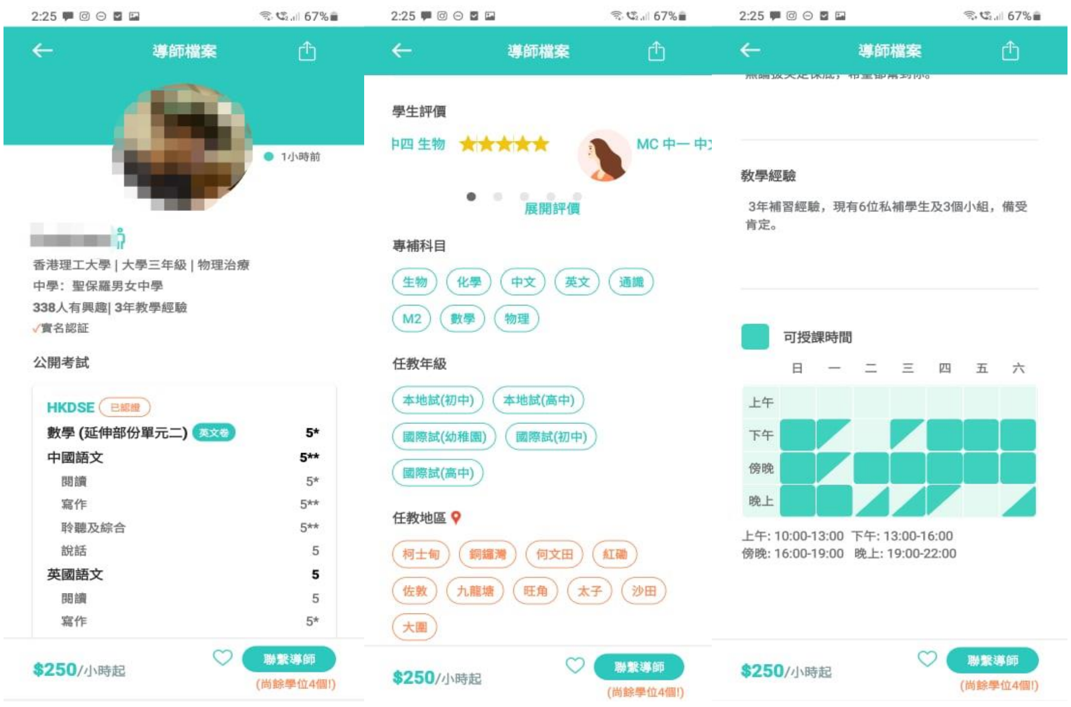

6:59 72%
資源
影片 筆記
中文 英文 數學 通識 M2 物理 化學 生物 商管 經濟 中史 歷史
5**必備閱讀題型特訓 掌握寫作手法 已有468人拎取
作者: Adrian Ko 免費拎取
中文考試必溫！120個超常考文言實詞表 已有468人拎取
作者: ROADTOHKU 免費拎取
議論文-實用名人事例24個 已有432人拎取
免費拎取
必讀5**中文範文精華筆記(筆記Giveaway) 已有434人拎取
作者: LITTLE DINOSAURPIG 免費拎取
主頁 資源 對話 帳戶

Figure 2.21: Learning Material Sharing of Stapps

v. Tutors Profile and Reviews

Students and parents could give reviews and rates to tutors after attending the class. There are reviews and rates, tutors' basic information, and additional information are displayed in the tutors' profile. The additional information is including grades of public examination of tutors that have been verified and available time slots that are rarely displayed in tutors' profiles compared with other similar applications. After looking through the tutors' profiles, students and parents can add their favorite tutors to their favorite list.



香港理工大學 | 大學三年級 | 物理治療
中學：聖保羅男女中學
338人有興趣| 3年教學經驗
✓實名認証

公開考試

HKDSE 已認證

數學 (延伸部份單元二) 英文卷	5*
中國語文	5**
閱讀	5*
寫作	5**
聆聽及綜合	5**
說話	5
英國語文	5
閱讀	5
寫作	5*

學生評價
中四 生物 ★★★★★

專補科目
生物 化學 中文 英文 通識 M2 數學 物理

任教年級
本地試(初中) 本地試(高中) 國際試(幼稚園) 國際試(初中) 國際試(高中)

任教地區
柯士甸 銅鑼灣 何文田 紅磡 佐敦 九龍塘 旺角 太子 沙田 大圍

教學經驗
3年補習經驗，現有6位私補學生及3個小組，備受肯定。

🟩 可授課時間

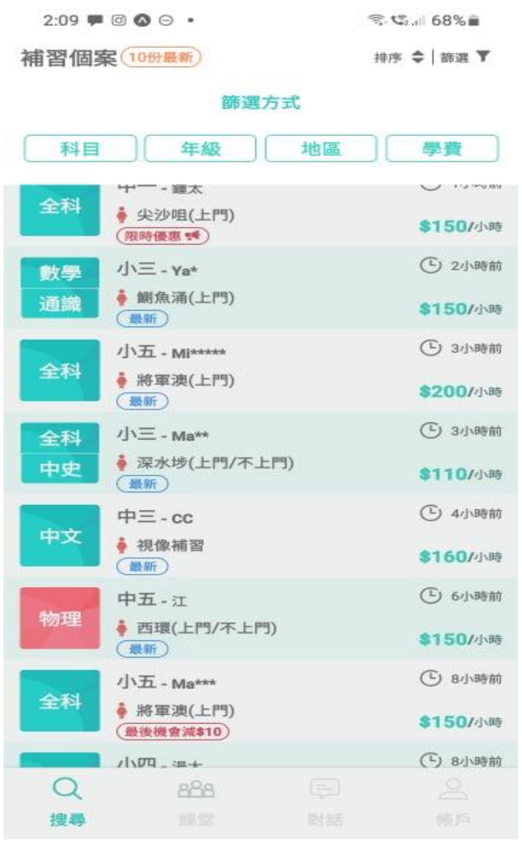
	日	一	二	三	四	五	六
上午							
下午	🟩	◩		◩	🟩	🟩	🟩
傍晚	🟩	◩	🟩	🟩	🟩	🟩	🟩
晚上	🟩	🟩	◪	◪	◩		◪

上午: 10:00-13:00 下午: 13:00-16:00
傍晚: 16:00-19:00 晚上: 19:00-22:00

Figure 2.22: Tutors' Profile of Stapps

vi. Jobs Applying

Tutors can apply for the tutoring jobs that were posted on Stapps wait for the reply of the students and parents. Tutors could filter the jobs by using subject, area, level, and tutoring fee per hour. Besides, tutors can sort the job request according to the tutoring fee and the time of the request.


2:09 68%

補習個案 10份最新 排序 | 篩選

篩選方式

科目 | 年級 | 地區 | 學費

全科 | 尖沙咀(上門) | 限時優惠 | $150/小時

數學 通識 | 小三 - Ya* | 鰂魚涌(上門) | 最新 | 2小時前 | $150/小時

全科 | 小五 - Mi***** | 將軍澳(上門) | 最新 | 3小時前 | $200/小時

全科 中史 | 小三 - Ma** | 深水埗(上門/不上門) | 最新 | 3小時前 | $110/小時

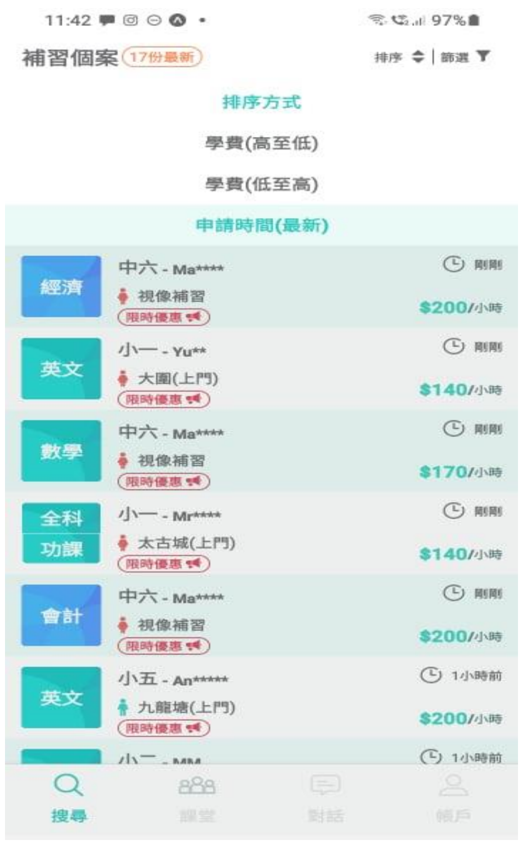
中文 | 中三 - cc | 視像補習 | 最新 | 4小時前 | $160/小時

物理 | 中五 - 江 | 西環(上門/不上門) | 最新 | 6小時前 | $150/小時

全科 | 小五 - Ma*** | 將軍澳(上門) | 最後機會減$10 | 8小時前 | $150/小時

8小時前

Figure 2.23: Job Filtering of Stapps


11:42 97%

補習個案 17份最新 排序 | 篩選

排序方式

學費(高至低)

學費(低至高)

申請時間(最新)

經濟 | 中六 - Ma**** | 視像補習 | 限時優惠 | 剛剛 | $200/小時

英文 | 小一 - Yu** | 大圍(上門) | 限時優惠 | 剛剛 | $140/小時

數學 | 中六 - Ma**** | 視像補習 | 限時優惠 | 剛剛 | $170/小時

全科 功課 | 小一 - Mr**** | 太古城(上門) | 限時優惠 | 剛剛 | $140/小時

會計 | 中六 - Ma**** | 視像補習 | 限時優惠 | 剛剛 | $200/小時

英文 | 小五 - An***** | 九龍塘(上門) | 限時優惠 | 1小時前 | $200/小時

1小時前

Figure 2.24: Job Sorting of Stapps

2.2.5 Zhang Men

Zhang Men only provides services to 1 type of end-users which is (parents and students). Zhang Men is a cross-platform (web and mobile platform) online tutoring application in China that allows parents and students to find their ideal private tutors and conduct online classes. At the same time, it provides the many sub-functions such as class recording that enable the student to replay their class, homework checking function, and so on. More sub-functions will be available on the mobile platform while only main functions will be provided on the web platform. It is considered as a combination of tutor finder and online tutoring application that is complete and well developed. Zhang Men is first released in 2014 by Shanghai Zhang XiaoMen Education Technology Co., Ltd. This mobile application is only available in China AppStore or downloads via their official website at: https://www.zhangmen.com/.

i. Trial Tutoring Class and Tutor Matching

Zhang Men allows the students or parents to attend one free trial class before purchasing the tutoring class. Students or parents should fill in the name, level, subject, available time slot (only in the mobile platform) and select the preferred tutor teaching style (only in mobile flatform). Next, the education consultant will contact the students or parents to more understand their learning requirements of them and arrange for the trial class. Once, students and parents are satisfied with the trial class and the tutor, the matching is considered a success and the class will be conducted after payment.

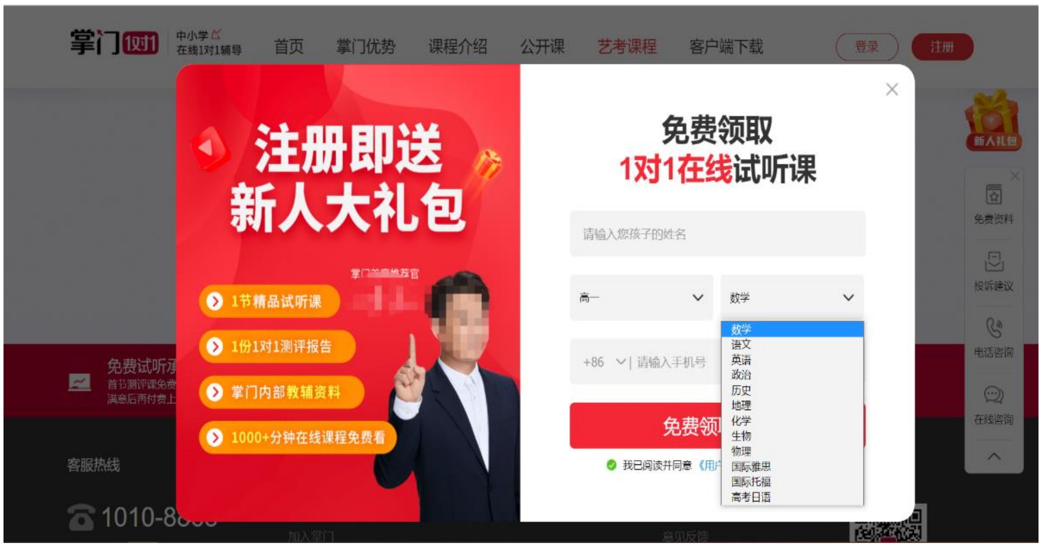


Figure 2.25: Trial Tutoring Class of Zhang Men (Web Application)

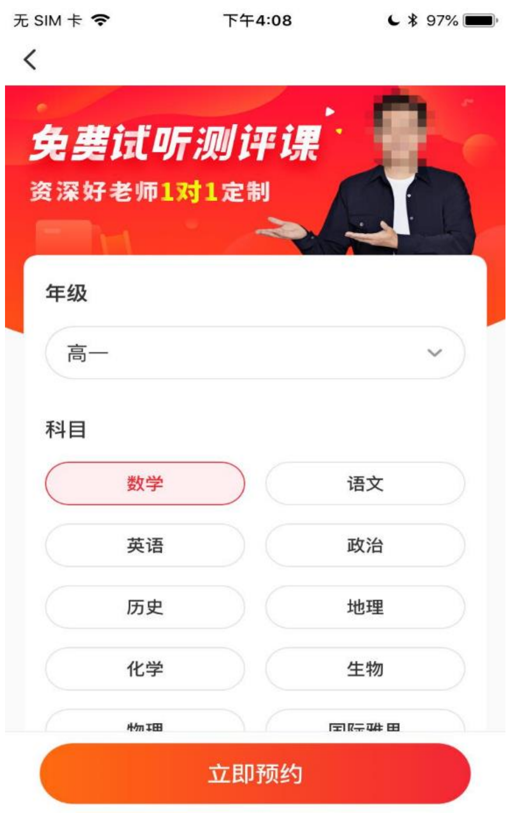


Figure 2.26: Trial Tutoring Class of Zhang Men (Mobile Application)

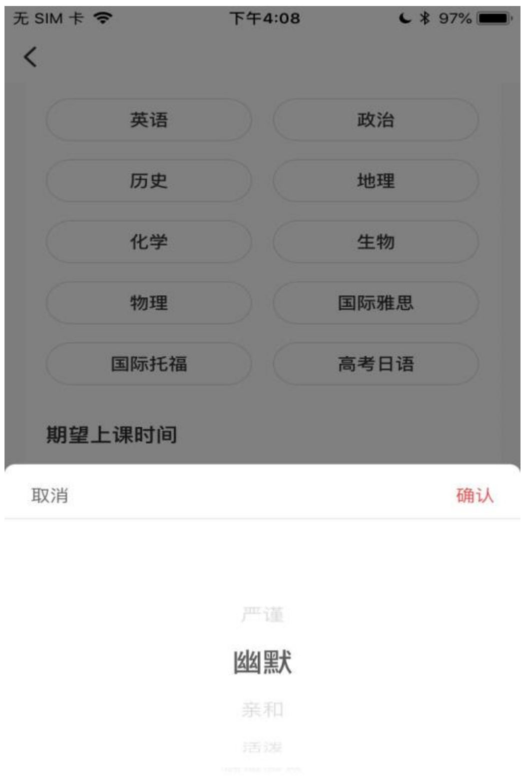


Figure 2.27: Tutor Teaching Style of Trial Tutoring Class of Zhang Men (Mobile Application)

ii. Online Class Tutoring

Zhang Men provides the service of online tutoring by using their platform instead of third-party's platforms. The class will be recorded and can be replayed after the class for revision purposes. Besides, parents can monitor their child's study condition with the application. In addition, Zhang Men AI will guide the student to set up their instrument such as cameras, mic, wi-fi in a stable condition before the class.

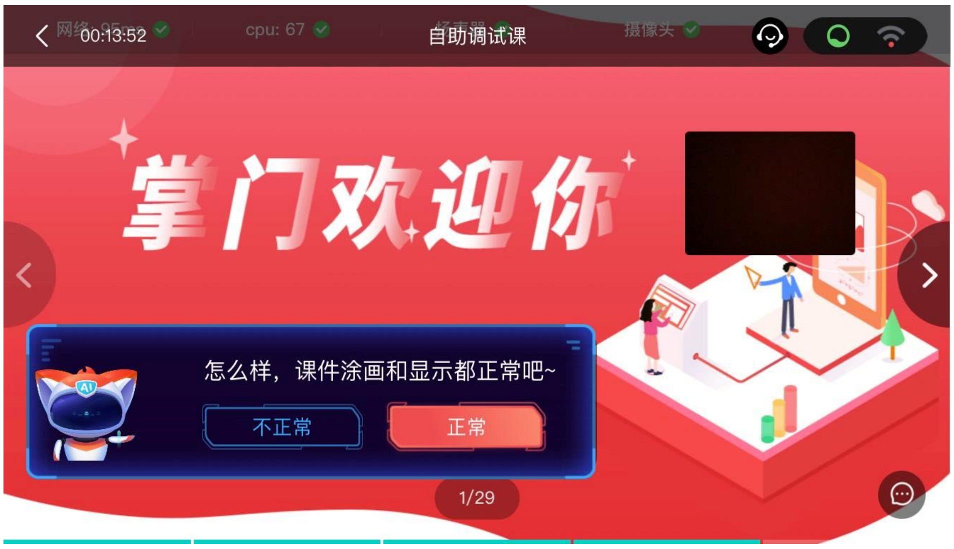


Figure 2.28: Instrument Testing of Zhang Men (Mobile Application)

iii. Schedule Management

There is a scheduler function in Zhang Men for the students to arrange their class and self-learning time wisely to prevent the booked classes are clash with each other's, remind the student to attend class on time and arrange their self-learning time.

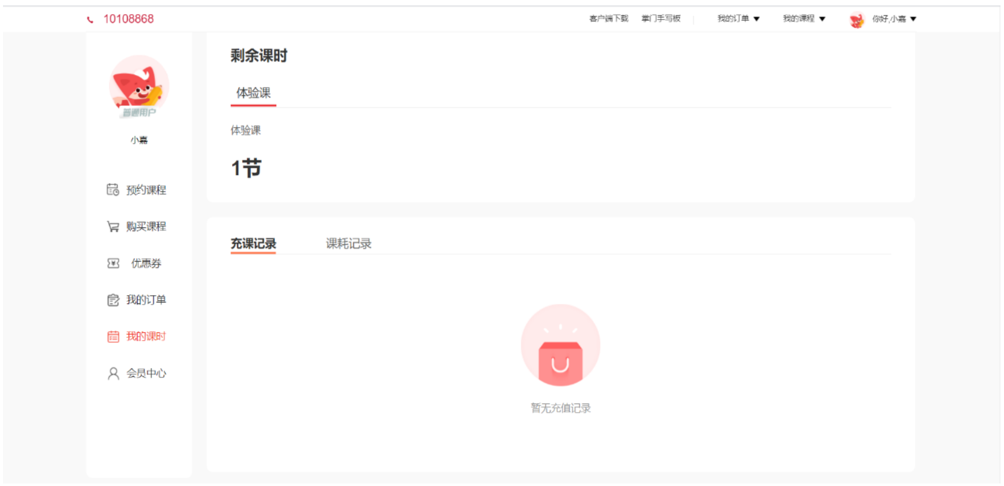


Figure 2.29: Class Schedular of Zhang Men (Web Application)

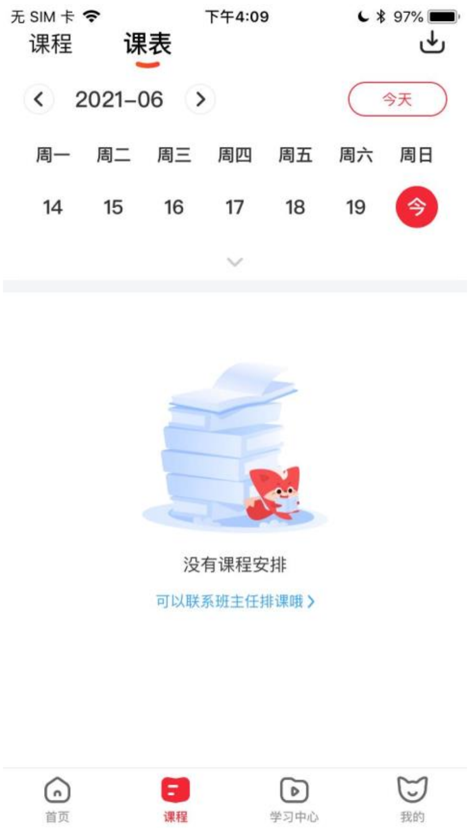


Figure 2.30: Class Schedular of Zhang Men (Mobile Application)

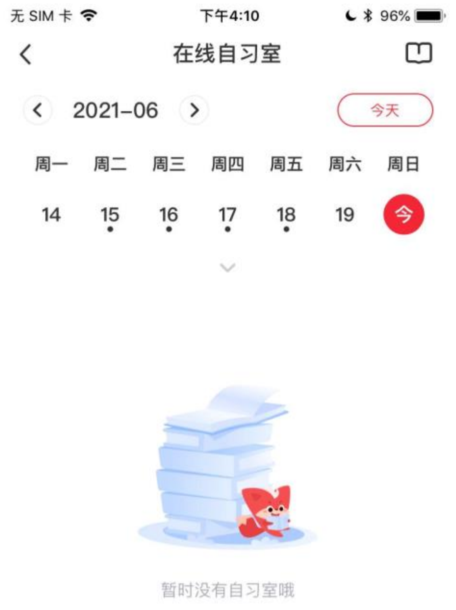


Figure 2.31: Revision Schedular of Zhang Men (Mobile Application)

v. Learning Materials Offering

Zhang Men provides various notes and module papers of different subjects for the student with no extra charge (only on a mobile platform) to train encourage the self-learning of students.

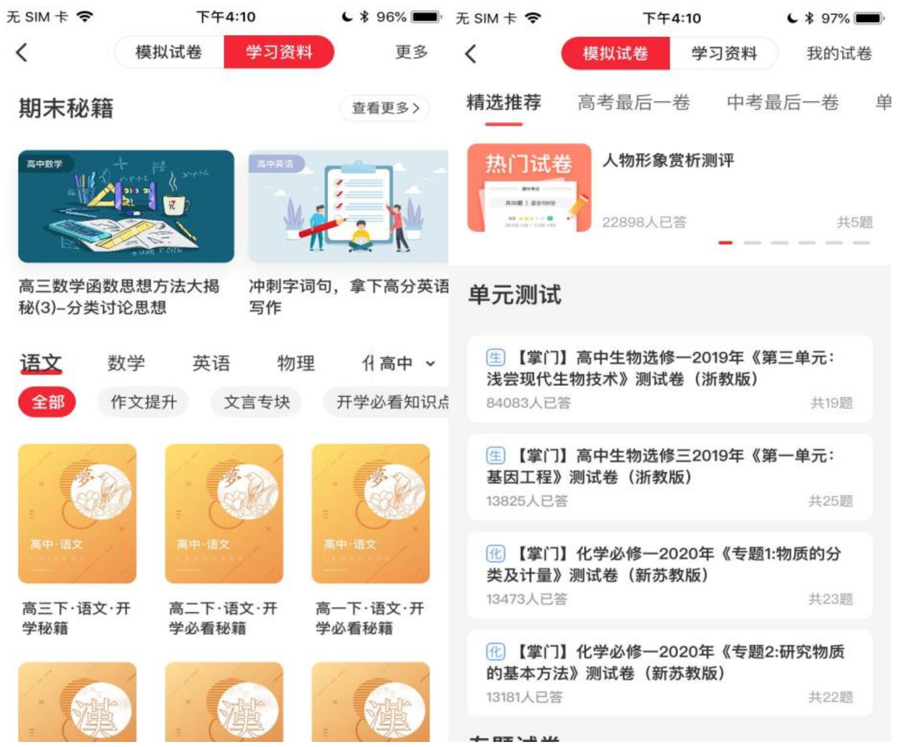


Figure 2.32: Learning Materials of Zhang Men

vi. Extra Public Classes Redemption

Zhang Men provides some public classes that are tutored by some popular tutors that can be redeemed after the student achieve the minimum class attendance without extra fee-charging.

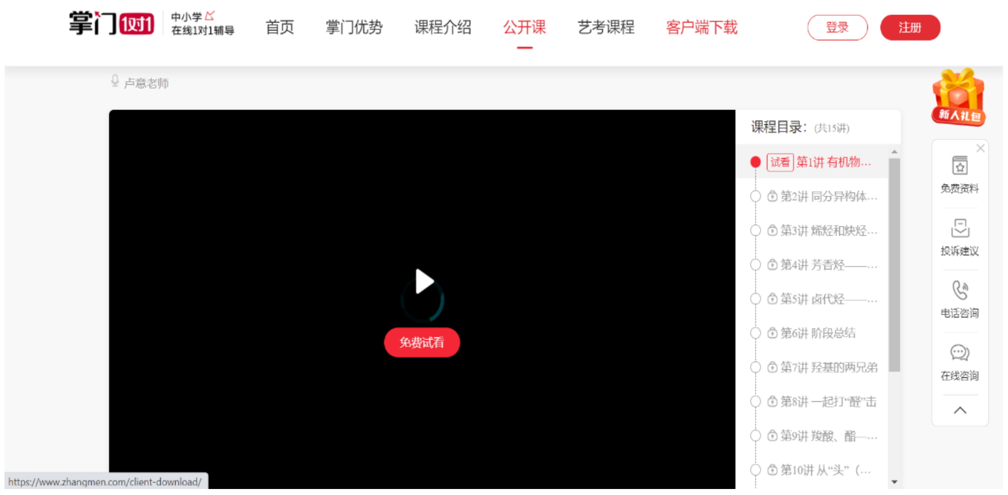


Figure 2.33: Public Class of Zhang Men (Web Application)

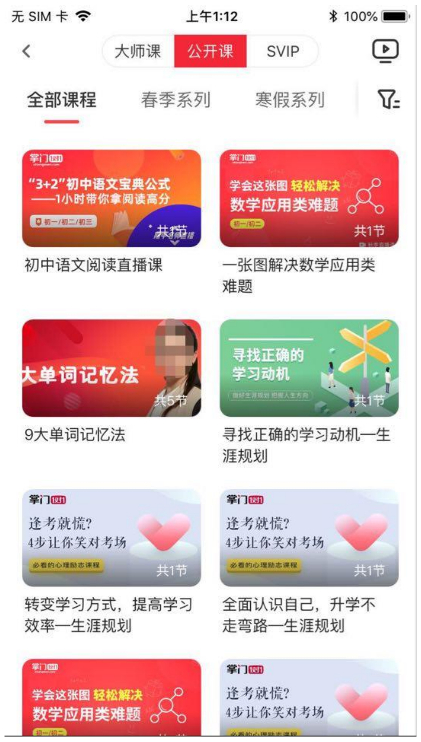
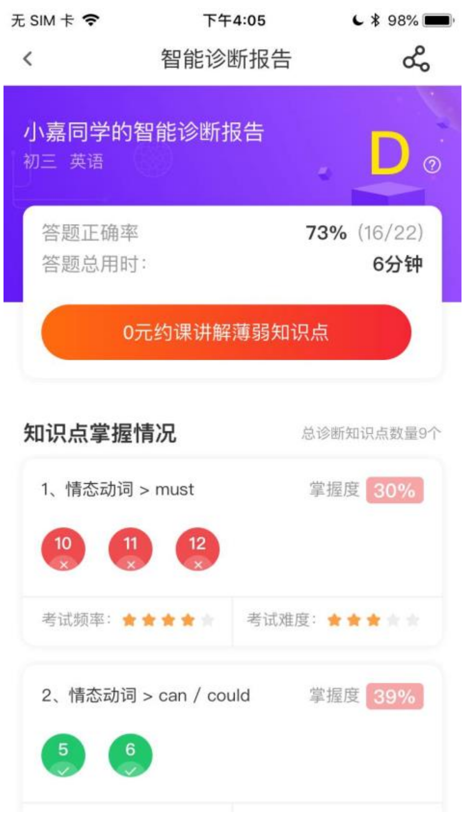


Figure 2.34: Public Class of Zhang Men (Mobile Application)

vii. AI Level and Learning Ability Testing

There are some tests are provided to students to understand their level in some subjects and their learning ability in the Zhang Men mobile application. This also helps the tutors to know the level of students and make a suitable teaching plan for them.


无 SIM 卡 下午4:05 98%

智能诊断报告

小嘉同学的智能诊断报告

初三 英语

D

答题正确率 73% (16/22)

答题总用时: 6分钟

0元约课讲解薄弱知识点

知识点掌握情况 总诊断知识点数量9个

1、情态动词 > must 掌握度 30%

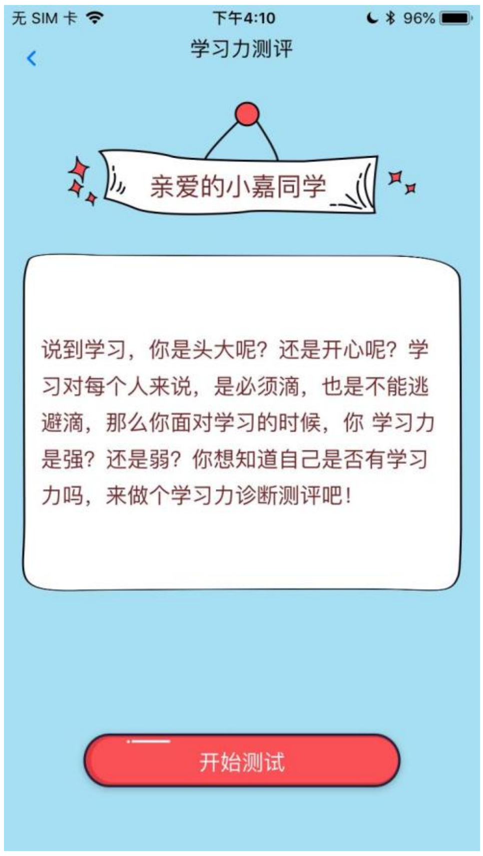
10 11 12

考试频率: ★★★★☆ 考试难度: ★★★☆☆

2、情态动词 > can / could 掌握度 39%

5 6

Figure 2.35: AI English Level Test of Zhang Men


无 SIM 卡 下午4:10 96%

学习力测评

亲爱的小嘉同学

说到学习，你是头大呢？还是开心呢？学习对每个人来说，是必须滴，也是不能逃避滴，那么你面对学习的时候，你 学习力是强？还是弱？你想知道自己是否有学习力吗，来做个学习力诊断测评吧！

开始测试

Figure 2.36: Learning Ability Test of Zhang Men

2.2.6 Comparison of Existing Similar Application

Table 2.1: Comparison of Existing Similar Application

Application Name / Functions	My Private Tutor	TeachMe	Tuition Hero Malaysia	Stapps	Zhang Men
Tutors Filtering	Yes (3 Criteria)	Yes (2 Criteria)	No	Yes (8 Criteria)	Yes (Education Consultant)
Tutors Requesting	Yes	No	Yes	Yes	Yes
Tutors Suggesting	Yes	No	No	Yes	No
Tutors Profile	Yes	Yes	No	Yes	No
Tutors Rating and Reviewing	Yes	Yes	No	Yes	No
Add to Favourite	No	No	No	Yes	No
Learning Materials Providing (Open Source)	No	No	No	Yes	Yes
Trial Class Providing	No	No	No	No	Yes
Online Tutoring Platform	No	Yes	No	No	Yes
AI Level Testing	No	No	No	No	Yes
Direct Chatting with Tutors	Yes	Yes	Yes	Yes	No
Class Schedule	No	No	No	No	Yes
Job Applying (Tutors/Trainers/Institute)	Yes	No	Yes	Yes	No
Class Managing (Tutors/Trainers/Institute)	No	Yes	No	Yes	No

In conclusion, there are various functions and services are provided by the applications to achieve the goal of matching an ideal tutor to a student or tutor online. Some of the applications major in the tutors finding service such as MyPrivateTutor, Tuition Hero Malaysia, and Stapps while TeachMe and ZhangMen major in online tutoring and teaching services. Each of the applications possesses their strength and weakness while the strength of the applications can be referred to and combined in developing a more intelligent and user-friendly private tutor finder mobile application. There are 4 functions (tutor filtering, tutor profiles, tutor rating and reviewing, direct chatting with tutors) are selected as the references which is sufficient to achieve the main goals of helping students and parents to find their ideal tutor while easier for the tutors in getting their jobs.

i. **Tutor Filtering**

   Students and parents could search their ideal tutors by selecting the required criteria in the filter list, the tutor list will be displayed.

ii. **Demo Class Requesting (Trial Class)**

   Students and parents can send the demo class request (trial class) to the tutors before hiring the tutor to ensure the tutor is suitable for the student.

iii. **Rate and Review**

   Students and parents could give rates and reviews to the tutors after attending their demo or formal classes and act as a reference to other students and parents who are looking for a tutor.

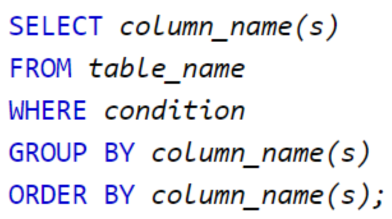
iv. **Tutors' Profile Viewing**

   The tutors' profile displays their basic information (name, education level, subject, teaching style, and so on), rate, and reviews that easier for the students and parents to filter and find their ideal tutor.

2.3 Searching Approaches

2.3.1 Exact searching by using SQL query

Structured Query Language (SQL) is a language that is used to access and manage the underlying database of the application or software (Myalapalli and Teja, 2015). SQL is categorized into two standards which are American National Standards Institute (ANSI) and International Organization for Standardization (ISO). Although there are various versions of SQL language, some of the major commands in different versions are similar due to the ANSI or ISO standard. For example, SELECT, WHERE INSERT, DELETE, and UPDATE commands (W3Schools, 2021a). SQL queries could be applied in retrieving the matched-criteria tutors' information from the database and displayed as a list. Figure 2.37 shows the basic structure of SQL query in search results from the database.



```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

Figure 2.37: Basic Structure of SQL Query (W3Schools, 2021b)

The "SELECT" must come with the "FROM" in retrieving the specified attributes of the records. At the same time, the "WHERE" statement is applied to filter the displayed records that fulfilled the required conditions or ranges. The "GROUP BY" statement helps in combining the same values into summary rows while the "ORDER BY" statement arranges the result records in ascending or descending order according to the specified attributes (column). There are some strengths and limitations of SQL query, and it may affect the searching result and performance in this application.

Table 2.2: Strengths and Limitations of SQL Query

Strength	Limitation
- Easy to learn and understand - Portable (can be embedded with different applications according to developers' needs and requirements)	- Large queries consume many memories and cause Parsing Exception (Elastic NV, 2021). - Suppose to produce consistent results based on unchanging input data (Helland, 2016). - Possible to get no result due to incorrect condition settings (exact matching all the required criteria from users may cause no result) - Sorting by aggregation cause reduce accuracy of matched results (usage "OR" clause in

	filtering the tutors cause many less relevant tutors are considered as “matched result “)

2.3.2 Similarity Measures

Similarity measures are approaches that measure the similarity between two objects by calculating their distance in terms of features similarity. The longer the distance between two objects, the bigger the differences between the two objects and vice versa. The similarity between two objects could be measured in the range of 0 to 1, 1 indicates the two objects are the same while 0 indicates the two objects are completely different. The more the similarity score closer to 1 the higher the degree of similarity between the two objects. A similarity measure is a basic block that is widely applied in many machine learning and data mining such as classification and clustering (Polamuri, 2015). There are five popular similarity measures always be applied in the searching function.

i. Euclidean Distance

Euclidean distance is one of the common measures that is applied to calculate the length of the straight connecting path of the two spatial points (Lu et al., 2020). Euclidean distance is calculated based on the Pythagorean theorem (Polamuri, 2015). In simple word, Euclidean distance measure the straight distance between two vectors. The bigger the Euclidean distance value, the smaller the similarity between two objects. The formula for calculating Euclidean distance is shown in equation 2.1.

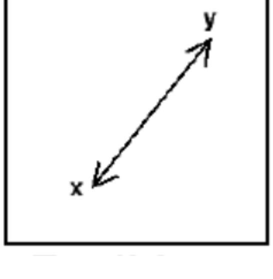
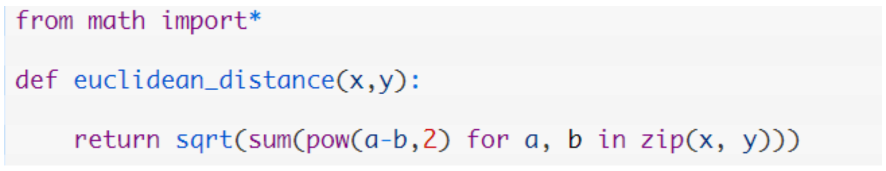


Figure 2.38: Graphical Representation of Euclidean Distance (Sharma, 2019)

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$



```
from math import*

def euclidean_distance(x,y):

    return sqrt(sum(pow(a-b,2) for a, b in zip(x, y)))
```

Figure 2.39: Sample Python Code by Applying Euclidean Distance (Polamuri, 2015)

Table 2.3: Strengths and Limitations of Euclidean Distance

Strength	Limitation
- Perform well in compacted or isolated clusters dataset (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015). - Additional new objects in the analysis will not affect the distance of the 2 objects (Bora and Gupta, 2014).	- The distance is greatly affected by the scale differences (Bora and Gupta, 2014). - The largest scale feature will dominate the other features (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015). - The two objects that have the same attribute value get a bigger distance value than the objects pair that do not have the same attribute values (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015).

ii. Manhattan Distance

Manhattan distance is also known as Taxicab distance or the City Block distance that calculates the sum of the absolute differences between its cartesian coordinates (x-coordinates and y-coordinates) (Brownlee, 2020). In simple words, Manhattan distance calculates the total differences of x-coordinate and y-coordinate of two objects. A mathematical way to explain Manhattan distance is by calculating the distances of the x-axis and y-axis of two objects at the right angle (Polamuri, 2015). Equation 2.2 shows the formula for calculating Manhattan distance.

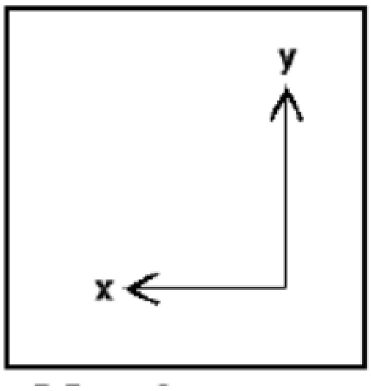
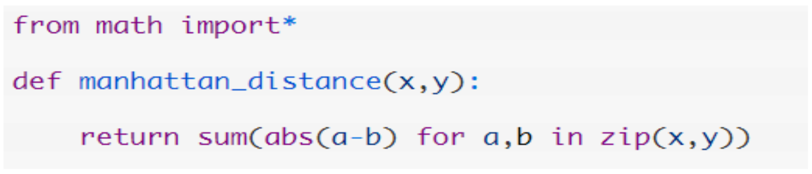


Figure 2.40: Graphical Representation of Manhattan Distance (Sharma, 2019)

$$d = \sum_{i=1}^{n} |x_i - y_i| \tag{2.2}$$



```python
from math import*

def manhattan_distance(x,y):

    return sum(abs(a-b) for a,b in zip(x,y))
```

Figure 2.41: Sample Python Code by Applying Manhattan Distance (Polamuri, 2015)

Table 2.4: Strengths and Limitations of Manhattan Distance

Strength	Limitation
- Perform well in compacted or isolated clusters dataset (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015). - Perform well in the high-dimensional data mining process. (Better than Euclidean distance in a many-feature dataset) (Aggarwal, Hinneburg and Keim, 2001)	- Sensitive to outliers (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015).

iii. Minkowski Distance

The Minkowski Distance is a generalization distance metric of Euclidean distance and Manhattan distance (Polamuri, 2015). The formula of the metric can be modified to calculate the distance differently. Equation 2.3 shows the mathematical formula for calculating Minkowski distance. The p-value can be changed to 1,2 or ∞ to calculate the Manhattan distance, Euclidean distance, and Chebychev distance respectively (Sharma, 2019).

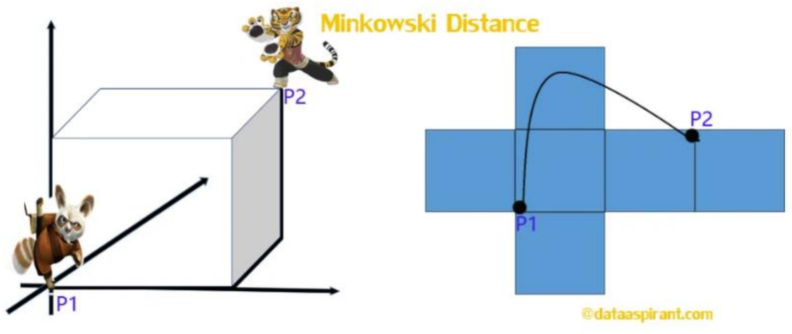
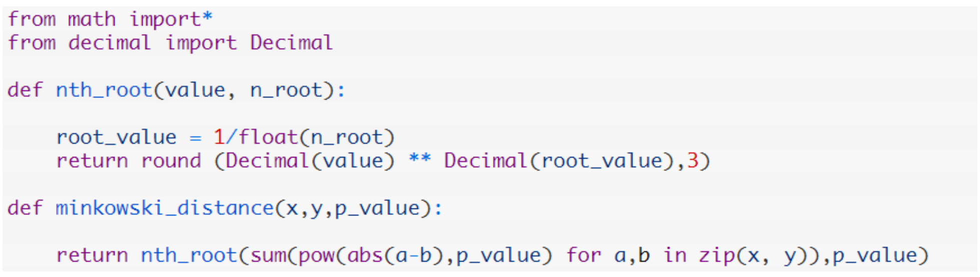


Figure 2.42: Graphical Representation of Manhattan Distance (Polamuri, 2015)

$$\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p} \tag{2.3}$$

where p = Minkowski metric (1,2 or ∞)



```
from math import*
from decimal import Decimal

def nth_root(value, n_root):

    root_value = 1/float(n_root)
    return round (Decimal(value) ** Decimal(root_value),3)

def minkowski_distance(x,y,p_value):

    return nth_root(sum(pow(abs(a-b),p_value) for a,b in zip(x, y)),p_value)
```

Figure 2.43: Sample Python Code by Applying Minkowski Distance (Polamuri, 2015)

Table 2.5: Strengths and Limitations of Minkowski Distance

Strength	Limitation
- Perform well in compacted or isolated clusters dataset (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015).	- The largest scale feature will dominate the other features (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015). - The same limitations as Manhattan distance and Euclidean distance, cause trouble when choosing a wrong p-value (Grootendorst, 2021).

iv. Cosine Similarity

Cosine similarity measures the inner product space similarity between two objects or vectors by finding the cosine angle between them while also identifying the two vectors are pointing in the same direction or different directions (Han, Kamber and Pei, 2012). The range of cosine similarity is from 0 to 1, 1 indicates there is a 0° angle difference between two vectors while 0 indicates there is a 90° angle difference between two vectors (Polamuri, 2015). The closer to 1 cosine similarity, the higher the similarity between two objects (smaller angle difference between two vectors) and vice versa. Equation 2.4 shows the formula for calculating Cosine similarity.

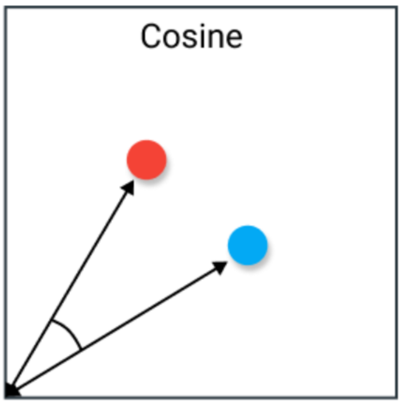
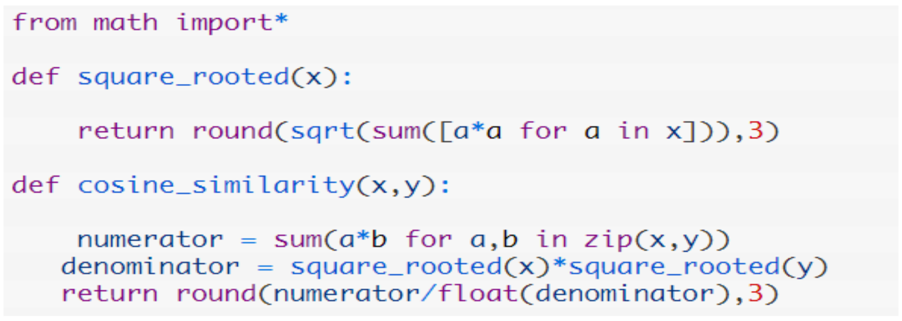


Figure 2.44: Graphical Representation of Cosine Similarity (Grootendorst, 2021)

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{2.4}$$

where θ = angle difference between two vectors



```
from math import*

def square_rooted(x):

    return round(sqrt(sum([a*a for a in x])),3)

def cosine_similarity(x,y):

    numerator = sum(a*b for a,b in zip(x,y))
   denominator = square_rooted(x)*square_rooted(y)
   return round(numerator/float(denominator),3)
```

Figure 2.45: Sample Python Code by Applying Cosine Similarity (Polamuri, 2015)

Table 2.6: Strengths and Limitations of Cosine Similarity

Strength	Limitation
- Could be varied when comes to linear transformation (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015). - Not dependent on vectors' length (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015).	- Could not be varied when comes to rotation (Shirkhorshidi, Aghabozorgi and Ying Wah, 2015). - The magnitude of vectors' direction is not been taken into account (Grootendorst, 2021).

v. Jaccard Similarity Coefficient/Index

Jaccard similarity coefficient is used to calculate the diversity and similarity between different sample sets. Jaccard similarity coefficient is calculated by the intersection cardinality of sets divided by the union cardinality of the sample sets. In simple words, the Jaccard similarity coefficient is calculated by the number of similar elements in sets divided by the total number of elements in sets (Grootendorst, 2021). The range of the Jaccard similarity coefficient is from 0 to 1 and could be converted as a percentage by multiplying the division result by 100. The more the Jaccard similarity coefficient closer to 1 or 100%, the higher the similarity between sets and vice versa (DeepAI, 2016). Equation 2.5 shows the formula to calculate the Jaccard similarity coefficient.

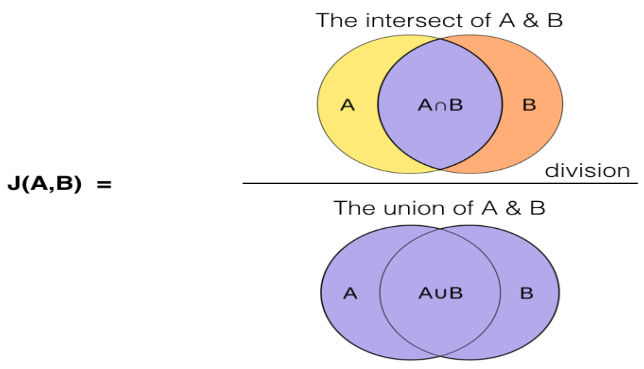
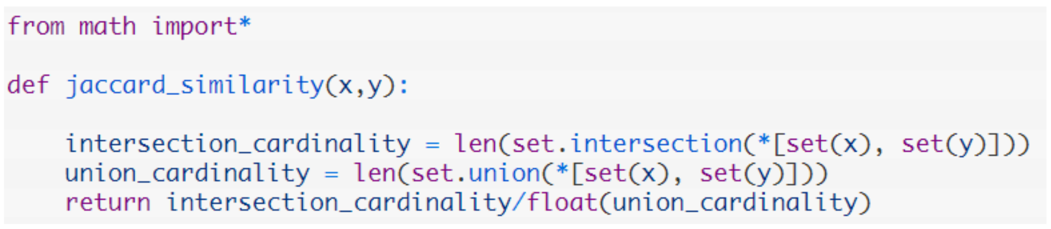


Figure 2.46: Graphical Representation of Jaccard Similarity Coefficient (Uniqtech, 2020)

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.5}$$

where ∪ = union, ∩ = intersection



```
from math import*

def jaccard_similarity(x,y):

    intersection_cardinality = len(set.intersection(*[set(x), set(y)]))
    union_cardinality = len(set.union(*[set(x), set(y)]))
    return intersection_cardinality/float(union_cardinality)
```

Figure 2.47: Sample Python Code by Applying Jaccard Similarity Coefficient (Polamuri, 2015)

Table 2.7: Strengths and Limitations of Jaccard Similarity Coefficient

Strength	Limitation
- Perform well in word similarity measuring (Niwattanakul et al., 2013).	- Highly influenced by the size of the dataset (large dataset will increase the number of union sets) (Grootendorst, 2021).

2.3.3 K-Nearest Neighbours (KNN) Algorithm

K-Nearest Neighbours (KNN) Algorithm is a supervised machine learning algorithm that is widely applied in solving the classification problem and predict the result. KNN algorithm assumes similar objects near to each other and classifies them by calculating their distances in order to predict the target result (Harrison, 2018). The distance in this algorithm could be calculated by Euclidean Distance, Manhattan Distance, Chebyshev Distance, Minkowski Distance, WMinkowski Distance, SEuclidean Distance, or MahalanobisDistance (scikit-learn, 2020). This algorithm could be applied in classification to classify the tutors into ideal tutor groups and non-ideal tutor groups by using the searching history of the users as the train data.

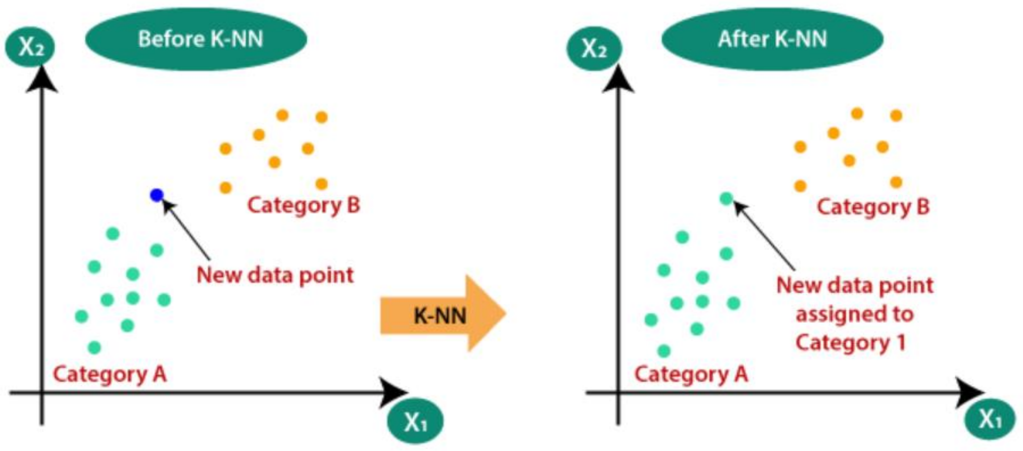


Figure 2.48: Graphical Representation of KNN Algorithm (Javatpoint, 2021)

Table 2.8: Strengths and Limitations of KNN Algorithm (Javatpoint, 2021)

Strength	Limitation
- Simple to implement. - Robust to noisy training data.	- Hard to identify the most suitable K value. - Need many training data to get higher accuracy results.

2.3.4 Summary of Searching Approaches

In conclusion, the approaches have their strengths and limitations that might affect the performance of searching features in this application. However, the similarity measures are more suitable to be applied in the searching and matching function. This is because the similarity measures could calculate the similarity value between the two objects and the higher similarity objects could be displayed as the searched or matched result to the users. In contrast, the exact matching by using SQL query only displays the objects that exactly fulfill the required criteria of users that may cause fewer choices for the users or even no result be displayed. On the other hand, the KNN algorithm needs a large number of searching histories data of users to predict the group of the objects that prohibit the application of the algorithm in the searching or matching function (Javatpoint, 2021). KNN algorithm is more suitable to be applied in product categorizing function compared to searching or matching functions.

Similarity measures are more suitable to be applied in the searching and matching function in this project. The limitation of SQL may provide low accuracy results while the KNN algorithm needs a large number of users' history to predict a high accuracy result. There is a lack of users’ history in selecting tutors to train the KNN algorithm since it is a new application, KNN algorithm is more suitable to be applied in a feature-improving project in the future.

2.4 Software Development Methodology

The process in developing a successful software is not only “code and fix”. There are other processes such as planning, design, and maintain should not be neglected to produce a successful software under the limitation of time, cost, resources, and so on. Therefore, software development methodology is needed to ensure the software development is conducted more efficiently and predictable by imposing some disciplined process upon it (Awad, 2005). In simple words, software development methodology is a framework that is applied in software development to plan structure and control the process of development (Simelane and Zuva, 2019). There are various types of software development methodologies or models that can be applied in developing software, the Waterfall Model, Prototyping Model, Agile Software Development Model and Rapid Application Development (RAD) Model will be discussed in the below section.

2.4.1 Waterfall Model

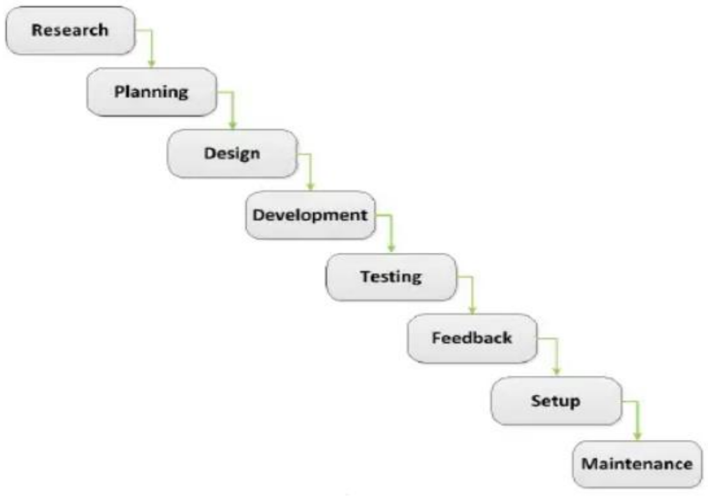


Figure 2.49: Waterfall Model (Despa, 2014)

Waterfall model is the oldest Software Development Life Cycle (SDLC) model that is widely applied in many software developments projects (Alshamrani and Bahattab, 2015). Waterfall model is a linear-sequential development model that only proceeds to the next phase after all activities in the current phase are completed. The required deliverables and requirements of the next phase are clearly defined before proceeding to the next phases. It ensures each development phase could be completed orderly without overlapping with each other within the fixed period. Besides, this causes each phase to be highly dependent on their previous phase and hard to be reversed once proceed to the next phase. In addition, the testing team only be involved in the testing phase. That causes the defects to be later found, high cost and time-consuming in solving the defects (Balaji and Murugaiyan, 2012). On the other hand, the Waterfall model that more concentrate on documentation and planning activities contributes to the quality controlling of a project. Therefore, it is more suitable for small-scale software development projects with clear requirements (Despa, 2014). There are some pros and cons to applying the Waterfall model in a software development project.

Table 2.9: Pros and Cons of Waterfall Model

Pros	Cons
Requirements are clearly defined before starting the development.	Not flexible in facing the requirements changing of the client.
Linear models cause easier to be implemented.	Working software only be produced at the late phase of the development cycle.
Ensure each phase can be completed within the given period.	High risk and uncertainty.
Provide a structured guide for an inexperienced development team.	Difficult to go back to the previous phase to solve the mistakes.
Proper documentation in each phase helps in quality control.	Poor performance in complex and object-oriented projects (Ragunath et al., 2010).
Perform well in small and well-understood requirements projects (Ragunath et al., 2010).	Poor performance in long and ongoing projects (Ragunath et al., 2010).

2.4.2 Prototyping Model

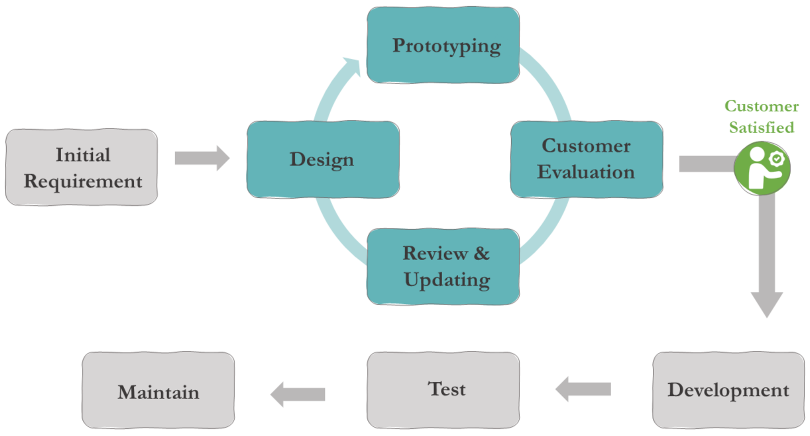


Figure 2.50: Prototyping Model

Prototyping model is a process that builds prototype rapidly with rapid planning and making improvement of the prototype iteratively according to the feedback from the customer until fulfil the requirements of the customer (Despa, 2014). In order to achieve customer's ideal software, Prototyping model allows users who only know their basic requirements to have an initial prototype and add on details during the iterative prototyping process. The planning phase is conducted rapidly to collect the basic needs and requirements of customers. By undergoing a fast analysis, design, and implementation phase, the initial prototype will be produced. Prototypes are evaluated by customers to get feedback and add more features to the software. This process is conducted iteratively until both the development team and customers satisfy with the prototype to undergo a software deployment and development. The iterative process in building prototype will not only improve the specified features of the software while helps customers to clarify their requirements (Susanto and Meiryani, 2019). However, an experienced development team or developer is needed to cope with the fast pace in developing prototypes. It might cause a longer time consuming due to the inexperience team not familiar with handling the customers and prototype building (Chandra, 2015). There are some pros and cons to apply Prototyping model in a software development project.

Table 2.10: Pros and Cons of Prototyping Model

Pros	Cons
Flexible in adapting customers' requirements change.	Development teams always neglect the best practice of software development to build a prototype quickly.
Easier to identify customers' detail needs through the iterative process.	Focus on quick developing reduce the quality of software.
Helps customers who unclear about their requirements to have an initial picture of the software.	Lack of consideration for the long-time maintenance.
High customers involvement improve communication between customers and development teams.	Prototyping causes the customer may neglect the overall quality of the software and more focus on specific features.
High performance in new unique software development projects that lack of previous example (Despa, 2014).	Poor performance in large software development projects (Chandra, 2015).

2.4.3 Agile Software Development Model

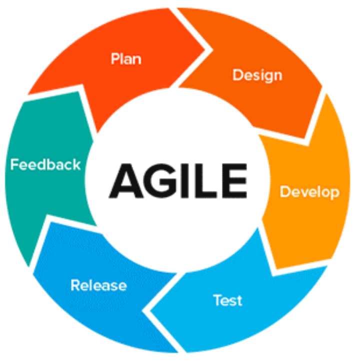


Figure 2.51: Agile Software Development Model (Accurate Reviews, 2020)

Agile software development model is primarily designed for an adaptive development team in responding to the changing requirements of customers and users (Balaji and Murugaiyan, 2012). The small software deliverables are produced quickly and continuously within few weeks instead of months. Agile mode pays attention to customer collaboration that involves customer interaction continuously from the initial to the late in development while less relying on documentation. Therefore, the development team keeps the initial phases as simple as possible such as the planning phase. The working software is built and release to the customers. Then, the customers provide feedback on the software. The development team responds to the feedback quickly and improves the software base on the feedback and requirements. Agile model always welcomes the change in requirements even in the late phase of development. The iteration of software development phases as a “loop” until the software met the requirements of customers. However, the fast pace in requirement changing and software releasing (weekly or every two weeks) give a high pressure on the development team (Dima and Maassen, 2018; Dubey, Jain and Mantri, 2015). There are some pros and cons of Agile software development model.

Table 2.11: Pros and Cons of Agile Software Development Model

Pross	Cons
Project can be delivered quickly.	High individual dependency due to less documentation. New developers or development team hard to take over the project.
Rapid respond to changing requirements of customers.	High pressures on development teams and developers to release software weekly or every two weeks.
Minimal of documentation reduces total project development time.	Scope of project increase continuously without limit if a clear-end goal is not be set well.
High involvement of customers in software development process reduces the misunderstanding of requirements within the development process.	The adding of new features and requirements cause some of the function's implementation violate the good design of development. Extra works are needed in long run to maintain and sustain the software (Gallagher, Dunleavy and Reeves, 2019; Dubey, Jain and Mantri, 2015).
Perform well in small and medium software development projects (Rahim et al., 2018; Balaji and Murugaiyan, 2012).	Poor performance in large software development projects (Rahim et al., 2018; Balaji and Murugaiyan, 2012).

2.4.4 Rapid Application Development (RAD) Model

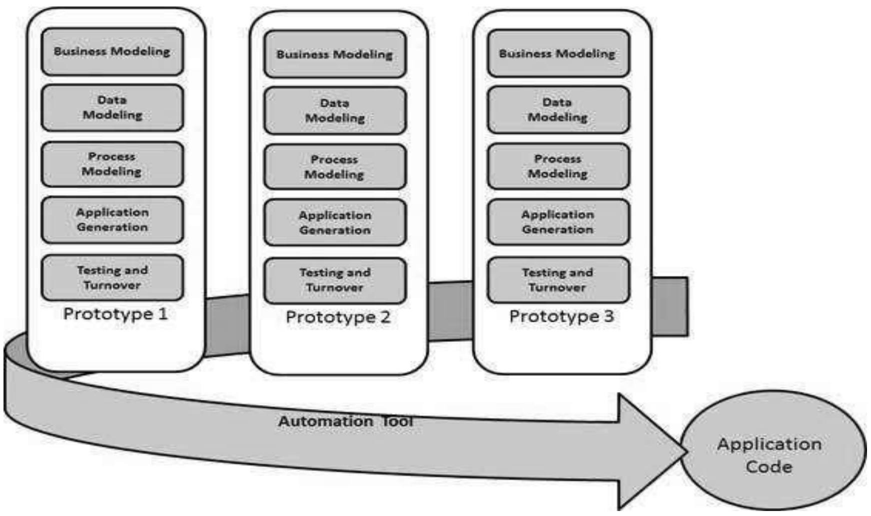


Figure 2.52: Rapid Application Development (RAD) Model (tutorialspoint, 2021)

Rapid Application Development (RAD) model is originated from rapid prototyping model and first released by James Martin in 1991(Berger, Beynon-Davies and Cleary, 2004). RAD is a linear sequential software development model that applied iterative methods to develop software in a very short time. It is more emphasis on development instead of planning tasks. The software or application is broken down into different modules with different features and developed parallelly. After going through the business modelling (business analysis), data modelling (research and information gathering), and process modelling (planning and design) quickly, the main focus is on application generation, testing and turnover. The process and data models are be converted to actual prototypes. The prototypes are refined and tested base on the customer's feedback. After customers satisfying with the prototypes, the actual application is built by integrating the prototypes with an automated tool. These processes are completed within a short time and it is time constraint is one of the limitations of this model (Fatimah, Supriatna and Kurniawati, 2018; Despa, 2014). There are some pros and cons of RAD model in software development.

Table 2.12: Pros and Cons of Rapid Application Development (RAD) Model

Pros	Cons
Application can be produced within a very short period.	Project scope less scalable (the scope of projects are limited and not similar as Agile and Prototyping models that can keep adding more and more features)
Adaptable to customers' requirement changes.	Poor documentation cause new developers or programmers hard to follow up the previous work.
Code and components can be reused.	Code integration issues always happen when combining the modules as a real application.
Perform well in small and medium software development project (Despa, 2014).	Weak performance in large software development project (Chandra, 2015).

2.4.5 Comparison of Software Development Methodologies

Table 2.13: Comparison of Software Development Methodologies

Methodology / Characteristic	Waterfall Model	Prototyping Model	Agile Software Development Model	Rapid Application Development (RAD) Model
Adaptability to Changes	Low	High	Very High	High
User Involvement	Low (Only involve in initial phases)	High (Involve in all phases but more focus on specific features of prototype)	Very High (Involve in all phases, focus the real application)	High (Involve in all phases but more focus on specific features of prototypes or modules)
Development Cost	Low	Intermediate	Very High	Very High
Targeted Users	Users have clear goals and requirements detail.	Users have only basic requirements and not clear with requirement details.	Users have only basic requirements and not clear with requirement details.	Users have only basic requirements and not clear with requirement details.
Experts Required	Little	Less	Many	Many
Performance in Short Time Schedule Projects	Very poor	Good	Very Good	Good
Suitable Project	Small and well-understood requirements projects	Small and new unique software development projects that lack of previous example	Small and medium software development projects	small and medium software development projects.

In conclusion, there are strengths and weaknesses in each software development methodology. If the methodology or model is chosen and applied in the right way, their strengths can be fully brought into a software development project to develop software efficiently and effectively. According to the comparison table above, all four models are suitable to be applied in small to medium-scoped projects. However, the only Waterfall model performs poorly in adapting the user requirement changes and the documentation consumes more total software development time compared to the other models. Although both Agile and RAD models adapt to the changes of users' requirements, it needs many experts that familiar with the specific technology field in releasing software or modules within short periods. This needs a very high cost to employ experts and senior developers. Therefore, the prototyping model is more suitable to be applied in this project. As the background research stated before, there are fewer similar mobile applications available that could be referred to as an example. Besides, the targeted users such as parents and students in Malaysia that less exposed to the similar application do not have a clear details requirement towards a tutor finder application. At the same time, this project is a short time schedule that only consists of three months in doing research and three months in developing the small-scoped mobile application.

2.5 Usability Testing

2.5.1 Overview of Usability Testing

Testing can be subdivided into two types which is formative testing and summative testing. Formative testing is conducted repeatedly while the software is in development to diagnose and fix the problem of the software. Summative testing is conducted after the software is nearly finished or finished to validate the software achieves the requirements before releasing (M.Barnum, 2021). Usability testing is one of the summative testing that can be defined as an evaluation performed by representative users in testing the software or products. The main objective of usability testing is to determine the satisfaction level of the users toward to software by collecting the qualitative and quantitative data and identifying usability problems of the software (usability.gov, 2021b; Gatsou, Politis and Zevgolis, 2013).

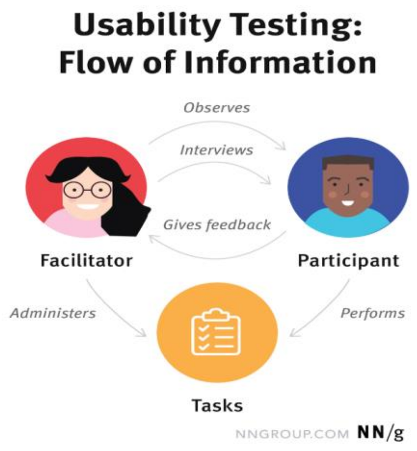
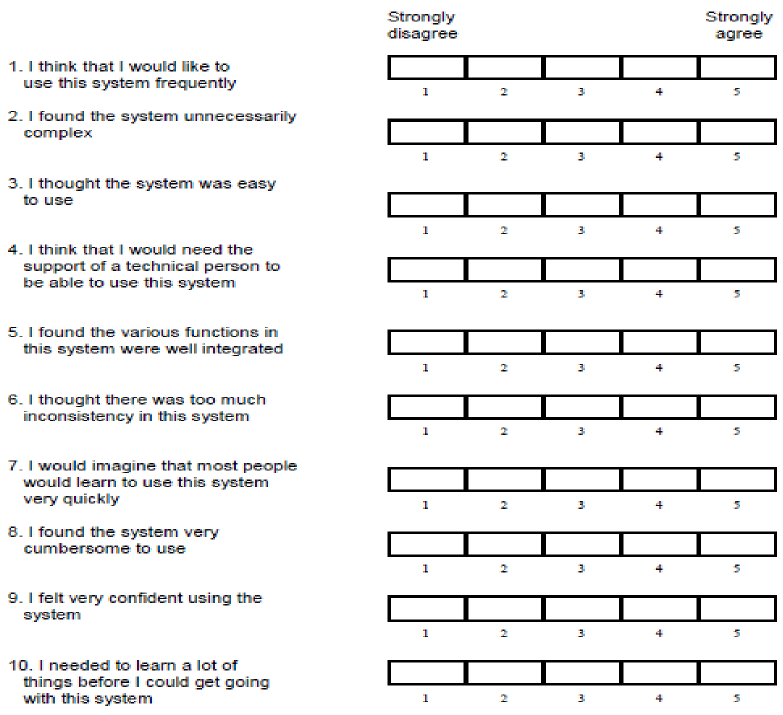


Figure 2.53: Usability Testing Flow of Information (Moran, 2019)

There are some main roles in usability which are participant, facilitator, and note-takers. Sometimes, the facilitator could be the note-takers as well. There are four main elements play important roles to conduct a small usability test (M.Barnum, 2021). First, establishing the user profiles by grouping the intended participants. A small usability testing usually involves 5- 10 participants due to budget limitations. Next, creating the task-based scenarios for the participants. Participants should perform a specific task that was included in the task-based scenario after the facilitator giving a brief description (not direct instruction). The facilitator or note-takers should observe the operation of participants in performing the tasks. Third, facilitators applied the think-aloud protocol in encouraging the participants to speak out their thoughts while using the software. This easier for the facilitators and note-takers in understanding what the participants are thinking at the moment. This helps facilitators and note-takers in evaluating the software and writing the testing report. After the test, the facilitator will give a questionnaire to participants or conduct a post-test interview to more understand the issues encountered by the participants in the test. Data are analysed and summarized as a test report. Forth, the software is changed based on the problems found in the usability test and conduct a usability test again.

Besides, an usability scale can be applied in determining the usability of a system which is System Usability Scale (SUS). Participants rate from 1 to 5 according to 10 questions, 1 indicate strongly disagree and 5 indicate strongly agree. There are two steps in calculating SUS which is sum up the score and multiply by 2.5. The score should be processed before summing up. The odd-numbered questions' score is deducted by 1 while deduct 5 by the even-numbered questions' score. The SUS will be calculated according to the rate from participants, if most of the participants score more than 68 means that the usability of the system is above average, the system performs well and user-friendly. In contrast, if majority of the participants score less than 68 means the usability of the system is below average, there might have some issues within the system have to be fixed before releasing (usability.gov, 2021a; Sauro, 2011; Brooke, 1995).



	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Figure 2.54: Sample Template of SUS (Brooke, 1995)

There are various type of usability testing methods and could be grouped as overall 3 main testing type (hotjar, 2021).

i. Moderated or Unmoderated Usability Testing

Table 2.14: Moderated and Unmoderated Usability Testing (hotjar, 2021; playbook ux, 2021)

Moderated Usability Testing	Unmoderated Usability Testing
Testing process must be administered by at least one moderator (facilitator or note-takers). Moderators give a brief introduction of the test to the participants and ask follow-up questions to the participants after the testing process. It is applied to more understand the user behaviour in finding the usability issues.	Testing process can be conducted without direct supervision. Participants are likely are in their own spaces and using their own devices to test the software. It is applied to analyse the user behaviour pattern in improving the software.

ii. Remote or In-person Usability Testing

Table 2.15: Remote and In-person Usability Testing (hotjar, 2021)

Remote Usability Testing	In-person Usability Testing
Testing process is conducted over the internet or by phone, face-to-face meeting with a moderator is not required. Remote usability testing main in getting large numbers of brief data instead of conducting a deep analysis in participants as in-person usability testing.	Testing process is conducted in the physical presence, a face-to-face meeting with the moderator is required. Moderators can observe the facial expression and body language of participants during the testing process. Therefore, the moderator can get a deeper analysis compared to remote usability testing.

iii. Explorative or Comparative Usability Testing

Table 2.16: Explorative and Comparative Usability Testing (hotjar, 2021)

Explorative Usability Testing	Comparative Usability Testing
Explorative usability testing is an open-ended test that allows participants to give opinions, brainstorm, and voice out their ideas towards the software. This	Comparative usability testing is a close-ended test that allows participants to choose theirs prefer solutions. This testing is usually applied in comparison

helps the development team in collecting the data in identifying potential new features of the software.	of software between competitors.

2.5.2 Usability Testing Methods

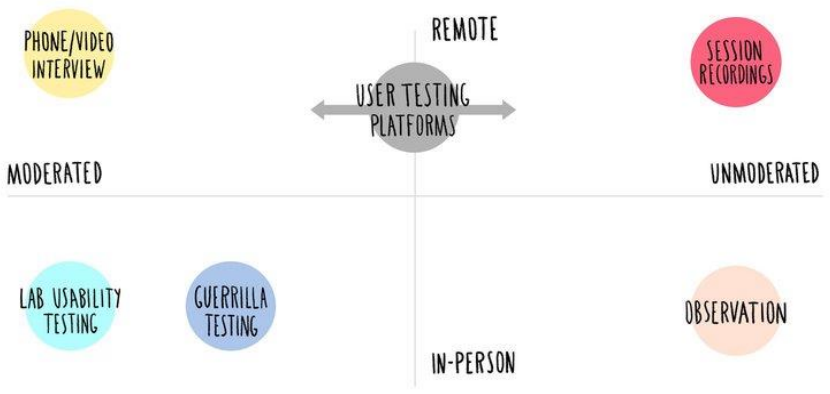


Figure 2.55: Usability Testing Models (hotjar, 2021)

There are some usability testing methods that are created based on the two main concepts above such as video interview, session recording, lab usability testing, Guerrilla testing, and observation. Video interview usability testing, lab usability testing, and Guerrilla usability testing is the top popular usability methods that widely applied in many systems testing processes.

i. Video Interview Usability Testing

Video interview usability testing method applied the concept of moderated and remote usability testing. In video interview usability testing, the moderator will give the task-based scenario to the participants and observe the performance of the participants remotely. Participants should share their screen and switch on their camera that enables moderator to catch every single facial expression of participants and task performed by the participants.

ii. Lab Usability Testing

Lab usability testing method applied the concept of moderated and in-person usability testing. In lab usability testing, the test is conducted a testing lab that consists of a one-way mirror. Participants perform the task while the moderators observe the participants' behaviour and ask them some related questions. At the same time, the stakeholders or development team members can observe the participants behind the one-way mirror (playbook ux, 2021). It is suitable to be applied in investigating the dept information of user-software interaction and get more qualitative information compared to video interview usability testing. However, it needs high cost to fulfil the requirement of standardised environments such as computers and a testing lab (hotjar, 2021; Babich, 2019).

iii. Guerrilla Usability Testing

Guerrilla usability testing method applied the concept of moderated and in-person usability testing. Guerrilla usability testing chooses to invite the participants randomly with a small incentive at public areas instead of recruit and pay the formal participants. It is a good testing method in collecting a large number of personal opinions toward a specific element in the software with a minimal cost. However, it is not suitable to be applied to get deep information due to the time limitation, the participants usually reluctant to give more than 15 minutes in doing software testing in a public area (Babich, 2019; hotjar, 2021).

2.5.3 Summary of Usability Testing

In conclusion, each usability testing method possesses its main specification in getting a test result. Different usability testing methods should be applied accordingly based on the purpose of the development team such as getting a quality or quantity test result. Video interview testing method is one of the suitable methods that can be applied in the testing stage of this project. During this Covid-19 pandemic, it could reduce physical interaction between participants and moderator while it allows the moderator to observe the facial expression and performance of participants remotely. Besides, it is a live session testing that provides a live communication platform for moderator and participants instead of session recording that only record the actions of participant without communication. In addition, the SUS could increase the accuracy in evaluating the application by showing a score instead of just get a generic answer from the questionnaires.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The prototyping model is applied as the software development methodology in this project. The methodology and workplan of the project are described according to the prototyping methodology. The time duration planning is prepared and explained in detail. The time duration planning and task distribution are to ensure the project can be completed on time and follow the plan without running in the wrong direction. The main development tools are identified to develop the application such as React Native framework, Firebase, and Visio Studio Code.

3.2 Software Development Methodology

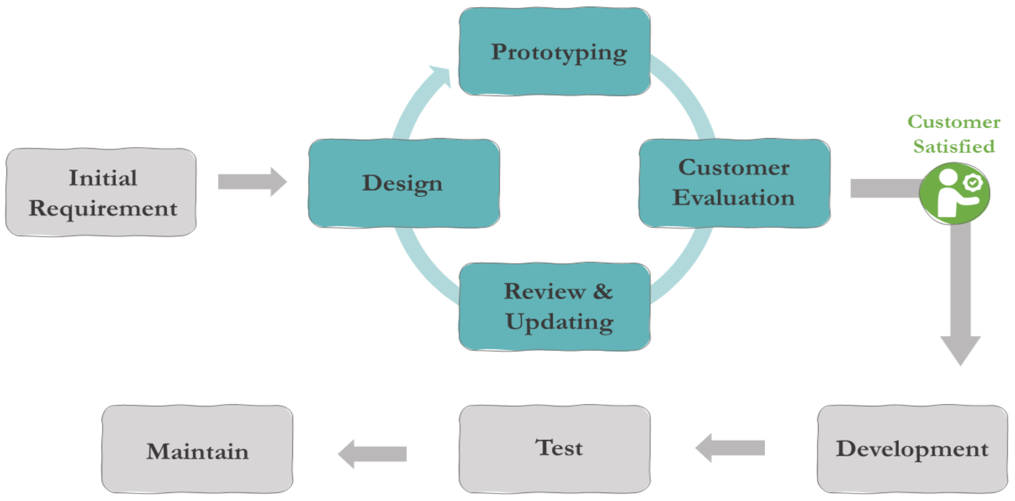


Figure 3.1: Prototyping Model

According to the study in CHAPTER 2, prototyping model is chosen as the software development methodology in developing this intelligent mobile private tutor finders application. Prototyping model helps users in identifying their real needs and requirements during the iteration of prototyping implementation. Besides, there are other reasons this model is chosen are clarified in CHAPTER 2. There are 5 main parts of prototype model are shown in figure 3.1 which include initial requirement, prototyping iteration, development, testing, deployment and maintenance phases.

3.2.1 Initial Requirements

The initial requirement phase is a phase that collects related information of the project to identify the initial requirements of the project such as system requirements, user requirements, resources required, and so on. In this project, the initial requirement phase is implemented by conducting various studies. A general background study is conducted to have a better understanding of the current status and market of private tutors in Malaysia. The study notifies the problems of the lack of similar mobile applications are available in Malaysia compared to China and Hong Kong. The limited tutor finder platforms that are available in Malaysia lack the features of "intelligent" in searching for a tutor. Hence, the scopes and objectives of this project are roughly drafted out after going through a general background study. At the same time, five similar existing applications are reviewed and compared base on their features and functionality. The strengths of each application can be considered as a reference in determining the functional requirements in this application. Besides, quantitative research is conducted to more clarify the tutor selection criteria of the intended users which are parents and students. The data is collected and analysed to identify criteria that are important for students and parents in searching for an ideal tutor. Therefore, the functional and non-functional requirements are determined while the use case modelling can be constructed accordingly. In order to ensure the development of the application can be completed within 24 weeks, project planning is implemented by constructing the work breakdown structure, project duration plan, and Gantt chart.

3.2.2 Prototyping Iteration

After the initial requirements are collected, prototype iteration phase is started to develop different versions of the prototype base on the reviews of customers. There are no customers in this project while users play the role of reviewing the prototype. There are four main stages in the prototyping iteration phase which include design, prototyping, user evaluation, review, and update stages. A total of three prototyping iteration will be performed in this project to improve and enhance the main functions of the prototype according to the feedbacks from users.

3.2.2.1 First Iteration

The main focus of the first iteration is to perform the main function of the application in searching a tutor by using different similarity measures. This main function enables the users to understand that the main focus of the application is to search for an ideal tutor by applying different similarity measures as the “intelligent” features in this application. Besides, the sample tutor’s data are prepared for the purpose of performing the search function.

i. Design

Drafting out the prototype layouts that enable a clearer picture of the application. Preparing the sample tutors’ data for the search function purpose. Making an outline of the implementation of similarity measures to the search function.

ii. Prototyping

Creating a primary prototype with sample data by using React Native framework. Implementing the search tutor function with three similarity measures which include Euclidean distance, Manhattan distance, and Minkowski distance.

iii. User Evaluation

Selected users perform a simple survey on the first prototype to evaluate the search function with the application of the three different similarity measures of the application. Selected users provide feedbacks and comments that help in the further refinement of the prototype.

iv. Review and Update

The collected feedbacks and comments are reviewed and analysed to improve and update the features of the prototype. Search function with the application of the three similarity measures is refined to achieve the satisfaction of users before proceeding to the second iteration.

3.2.2.2 Second Iteration

After achieving success in implementing the search function with the application of the first three similarity measures, the rest of the two similarity measures are implemented in the prototype. Besides that, additional functions are added to the prototype in the second iteration which is added to raise demo class function, rate and review tutor function.

i. Design

The layout of the application is drafted according to the addition of new two functions. Outlines of the additional functions are prepared to avoid the conflict of functions and interfaces.

ii. Prototyping

Implementing the other two similarity measures (Cosine similarity and Jaccard similarity coefficient) to the search function according to the suggestion that is provided by the users. Adding two more new functions to the prototype which include raise demo class function, book formal class function, rate and review tutors' function.

iii. User Evaluation

After all similarity measures are applied to the search function, the users evaluate the prototype again and provide feedbacks and comparison about the accuracy of the different similarity measures. Besides, the add to raise demo class function, book formal class function, rate and review function are checked by users to ensure all are workable.

iv. Review and Update

Analysing the effect of similarity measures on search function according to the comparison feedbacks from users and make enhancement on the search function. Review and refine the raise demo class, book formal class function, rate and review tutors' function according to the review of users before proceeding to the third iteration.

3.2.2.3 Third Iteration

The main focus of the prototype in the third iteration is the subfunctions of the application which include manage demo class, mana tutors' detail, edit profile function, register and log in functions. The produced prototype in the third iteration provides a very clear picture of the application workflow to the users.

i. Design

There are many functions in the third-iteration prototype that are confusing and cause conflict easily. Therefore, outline the interfaces and functions linkage to have a clear picture of the relationship of different interfaces and functions especially the newly added functions such as manage demo class, manage tutors' detail, edit profile function, register and log in functions.

ii. Prototyping

Implementing the raise demo class, manage demo class, mana tutors' detail, edit profile function, register and log in functions wisely to avoid the conflict between functions. The third-iteration prototype is considered as a "semi-final" product of the project that consists of all functions of the private tutors' finder's mobile application.

The complete functions are shown in the prototype that enables the user to completely understand the workflow in using the application.

iii. User Evaluation

Selected users' feedbacks are expected to be more distributed to different aspects of the prototype compared to the previous two iterations due to the implementation of full functions in this prototype. Different selected users may provide feedbacks on different functions base on their preference. Variety type of suggestions are provided to the prototype.

iv. Review and Update

The collection of reviews and feedbacks from selected users are analysed and applied to enhance the features of the prototype until the selected users satisfy with the final prototype before proceeding to the development phase.

3.2.3 Development

The final prototype is equipped with full functionality and can be implemented in a real application. At the same time, the minor view functions that required minimal user-application interaction will be developed in this development phase instead of the prototyping phase. This is due to the main aims of prototyping is to identify the needs of users in using the application (functions that need user-application interaction) while the minor view functions are considered as a minor part of functions that provide convenience to users instead of defining the needs of users. The features and functions in the final prototype are considered as a reference to the real application. Therefore, the codes and components can be refined reused, and became the reference in developing the real application. Besides, the sample data is not enough to cover the requirement of users in finding tutors, the real application is connected to the real-time database with the insertion of more tutors' data. The improvement and modification of user interface design are performed when the database connection and functions of the application are well-prepared. Last but not least, the comments are added as the internal documentation while the codes are tidied to make it simpler and clearer.

3.2.4 Test

After the application is well-prepared and the environment is set up, different types of tests are conducted to ensure the functional and non-functional requirements are achieved in this application. The tests are including unit tests, integration tests, system tests, usability tests, and user acceptance tests. Developers play the role in conducting the unit, integration test and system test while there are intended users who are invited as the tester to perform usability tests and usability testing. The main objectives of the unit test, integration test and system test are testing the functionality of applications (whether work or not) while the usability and user acceptance testing involve both functional and non-functional requirements (quality) of the application. The usability test is evaluated by using System Usability Scale (SUS). Once the SUS of this application is above 68 that indicates this application is ready to be deployed without big issue. The bugs or failures that are found in tests are fixed to ensure all the functions perform successfully. The test phase is ended while the test exit criteria are achieved and continue proceeding to the deployment phase.

3.2.5 Deployment and Maintenance

The last part of the application development process is the deployment and maintenance phase that indicates that the application is ready to be released and used by the users. In the deployment phase, the application is installed in an Android phone as the final product. The documentation work is completed by preparing a report that consists of project details such as project objectives, scopes, workflow, development process, and so on. The report clearly listed out the details of developing the application from the beginning to the end that enables an outsider can clearly understand the project. Then, a presentation is conducted to convey the overview of the project and show the final product (application).

3.3 Research Methodology

In this project, there are two types of research methodologies are applied which are questionnaire and literature review methodology. Literature review research methodology provides a better understanding of the related knowledge of this project while questionnaire research methodology clarifies the needs of potential end-users towards this private tutors' finder's mobile application.

3.3.1 Questionnaire

Questionnaire is one of the quantitative research methodologies that is applied to determine whether a hypothesis is approved or rejected. The criteria that are concerned by each end-user in selecting their ideal tutors are different from one to another, this became one of the challenges to determine the tutor searching criteria in this project. Therefore, a questionnaire is suitable to be applied in clarifying the real needs of different end-users in selecting their ideal tutors. There are 50 potential end-users are involved themselves as respondents in this questionnaire research via a google form. The respondents are different ages (from 19-55) and marital status (as students or parents). The questionnaire contains two main parts which are personal details and tutor selection criteria. There are mixed open-ended and close-ended questions in both sections. Respondents have to rate the importance of criteria in selecting a tutor and provide extra criteria that are important in selecting a tutor in the questionnaire. The results of questionnaire research are collected and analysed in the CHAPTER 4 Fact Findings section.

3.3.2 Literature Reviews

Literature reviews are neither qualitative nor quantitative research methodologies. However, literature review is considered as a research methodology that is used to offer an overview of different types of reviews, compare and evaluate different types of reviews (Snyder, 2019). It can provide a basic concept of a theory to the reviewers to have a better understanding of theories or technical skills that they are never exposed to before.

In this project, five existing similar applications' features have been reviewed and compared to have a better understanding of the main features of a tutor finder application. The five existing similar applications are MyPrivateTutor, TeachMe, TuitionHero Malaysia, Stapps, and Zhang Men. Next, different searching approaches are reviews and compared to identify the best approaches to be applied in the searching function to display a list of tutors that matches the expectation of the students and parents. The searching approaches that are studied are exact SQL query searching, similarity measures, and K-Nearest Neighbors (KNN) Algorithm. Next, there are four software development methodologies are studied to identify the methodology that is the most suitable to be applied in this project in developing a high-quality application within 24 weeks. Last but not least, the usability testing is studied to more understanding on the workflow and metrics can be applied in measuring the usability of this application.

3.4 Project Planning

3.4.1 Work Break Down Structure (WBS)

0.0 Intelligent Mobile Private Tutors Finders Application

1.0 Preliminary Planning

- 1.1 Conduct background study
  - 1.1.1 Gather general information of project
- 1.2 Identify problem statement
- 1.3 Identify project objective
- 1.4 Propose project solution
- 1.5 Propose project approach
- 1.6 Determine project scope
  - 1.6.1 Identify targeted users
  - 1.6.2 Identify modules covered

2.0 Project Analysis

- 2.1 Conduct literature review
  - 2.1.1 Review on similar existing application
    - 2.1.1.1 Study features of MyPrivateTutor
    - 2.1.1.2 Study features of TeachMe
    - 2.1.1.3 Study features of Tuition Hero Malaysia
    - 2.1.1.4 Study features of Stapps
    - 2.1.1.5 Study features of Zhang Men
    - 2.1.1.6 Compare and contrast features of applications
  - 2.1.2 Review on searching approach
    - 2.1.2.1 Research on SQL exact matching
    - 2.1.2.2 Research on similarity measures
    - 2.1.2.3 Research on K-Nearest Neighbors (KNN) algorithm
    - 2.1.2.4 List out strengths and limitations of approaches
  - 2.1.3 Review on software development methodology
    - 2.1.3.1 Study features of Waterfall model
    - 2.1.3.2 Study features of Prototyping model
    - 2.1.3.3 Study features of Agile model
    - 2.1.3.4 Study features of Rapid Application Development (RAD) model

2.1.3.5 Compare and contrast features of software development models

2.1.4 Review on usability testing

2.1.4.1 Research on usability testing methods

2.1.4.2 Research on System Usability Scale

2.2 Requirement analysis

2.2.1 Conduct survey

2.2.1.1 Design questionnaire

2.2.1.2 Collect data from intended users

2.2.2 Data analysis

3.0 Project Initial Specification

3.1 Time management

3.1.1 Identify tasks and activities

3.1.1.1 Construct works break down structure (WBS)

3.1.2 Create project duration plan

3.1.3 Construct Gantt chart

3.2 Tools management

3.2.1 Determine development tools

3.3 Requirement specification

3.3.1 Identify functional requirements

3.3.2 Identify non-functional requirements

3.4 Use case modelling

3.4.1 Construct use case diagram

3.4.2 Prepare use case description

4.0 Prototype Implementation

4.1 First Iteration

4.1.1 Design prototype with main functions

4.1.1.1 Set up sample data for developing search function purpose

4.1.1.2 Search function with Euclidean distance

4.1.1.3 Search function with Manhattan distance

4.1.1.4 Search function with Minkowski distance

4.1.2 User survey

4.1.3 Improve prototype based on user’s review

4.2 Second Iteration

- 4.2.1 Design Prototype with main functions
  - 4.2.1.1 Search function with Cosine similarity
  - 4.2.1.2 Search function with Jaccard similarity coefficient
  - 4.2.1.3 Rate and review function
  - 4.2.1.4 Raise demo class function
  - 4.2.1.5 Book formal class function
- 4.2.2 User survey
- 4.2.3 Improve prototype based on review

4.3 Third Iteration

- 4.3.1 Design Prototype with main functions
  - 4.3.1.1 Manage demo class function
  - 4.3.1.2 Manage tutors’ detail function
  - 4.3.1.3 Register and log in function
  - 4.3.1.4 Edit profile function
- 4.3.2 User survey
- 4.3.3 Improve prototype based on review

5.0 Development

- 5.1 Develop all functions
  - 5.1.1 Main functions with proper interface linkage
  - 5.1.2 Minor view functions with proper interface linkage
- 5.2 Set up real-time database
- 5.3 Insert data into database
- 5.4 Improve user interface design
- 5.5 Internal documentation
  - 5.5.1 Clean code
  - 5.5.2 Add comments for explanation

6.0 Testing

- 6.1 Conduct unit testing
- 6.2 Conduct integration testing
- 6.3 Conduct system testing
- 6.4 Conduct usability testing
- 6.5 Conduct user acceptance testing

7.0 Deployment

- 7.1 Deploy production

7.2 Documentation

7.3 Presentation

3.4.2 Project Duration Plan

Table 3.1: Project Duration Plan

Task Name	Duration	Start Date	Finish Date
Intelligent Mobile Private Tutors Finders Application	**339 days**	**Mon 7/6/21**	**Tue 19/4/22**
1.0 Preliminary Planning	**14 days**	**Mon 7/6/21**	**Sun 20/6/21**
1.1 Conduct background study	**2 days**	**Mon 7/6/21**	**Tue 8/6/21**
1.1.1 Gather general information of project	2 days	Mon 7/6/21	Tue 8/6/21
1.2 Identify problem statement	2 days	Wed 9/6/21	Thu 10/6/21
1.3 Identify project objective	2 days	Fri 11/6/21	Sat 12/6/21
1.4 Propose project solution	2 days	Sat 12/6/21	Mon 14/6/21
1.5 Propose project approach	2 days	Mon 14/6/21	Wed 16/6/21
1.6 Determine project scope	**4 days**	**Wed 16/6/21**	**Sun 20/6/21**
1.6.1 Identify targeted users	2 days	Wed 16/6/21	Fri 18/6/21
1.6.2 Identify modules covered	2 days	Fri 18/6/21	Sun 20/6/21
2.0 Project Analysis	**46 days**	**Sun 20/6/21**	**Sun 1/8/21**
2.1 Conduct literature review	**36 days**	**Sun 20/6/21**	**Fri 23/7/21**
2.1.1 Review on similar existing application	**12 days**	**Sun 20/6/21**	**Thu 1/7/21**
2.1.1.1 Study features of MyPrivateTutor	2 days	Sun 20/6/21	Mon 21/6/21
2.1.1.2 Study features of TeachMe	2 days	Tue 22/6/21	Wed 23/6/21
2.1.1.3 Study features of Tuition Hero Malaysia	2 days	Thu 24/6/21	Fri 25/6/21
2.1.1.4 Study features of Stapps	2 days	Sat 26/6/21	Sun 27/6/21
2.1.1.5 Study features of Zhang Men	2 days	Sun 27/6/21	Tue 29/6/21
2.1.1.6 Compare and contrast features of application	2 days	Tue 29/6/21	Thu 1/7/21
2.1.2 Review on searching approach	**10 days**	**Thu 1/7/21**	**Sat 10/7/21**
2.1.2.1 Research on SQL exact matching	2 days	Thu 1/7/21	Sat 3/7/21
2.1.2.2 Research on similarity measures	3 days	Sat 3/7/21	Mon 5/7/21
2.1.2.3 Research on K-Nearest Neighbors (KNN) algorithm	2 days	Tue 6/7/21	Wed 7/7/21
2.1.2.4 List out strengths and limitations of approaches	3 days	Thu 8/7/21	Sat 10/7/21
2.1.3 Review on software development methodology	**10 days**	**Sat 10/7/21**	**Mon 19/7/21**
2.1.3.1 Study features of Waterfall model	2 days	Sat 10/7/21	Mon 12/7/21
2.1.3.2 Study features of Prototyping model	2 days	Mon 12/7/21	Wed 14/7/21
2.1.3.3 Study features of Agile model	2 days	Wed 14/7/21	Fri 16/7/21
2.1.3.4 Study features of Rapid Application Development (RAD) model	2 days	Fri 16/7/21	Sun 18/7/21

2.1.3.5 Compare and contrast features of software development models	2 days	Sun 18/7/21	Mon 19/7/21
2.1.4 Review on usability testing	**4 days**	**Tue 20/7/21**	**Fri 23/7/21**
2.1.4.1 Research on usability testing methods	2 days	Tue 20/7/21	Wed 21/7/21
2.1.4.2 Research on System Usability Scale	2 days	Thu 22/7/21	Fri 23/7/21
2.2 Requirement analysis	**10 days**	**Sat 24/7/21**	**Sun 1/8/21**
2.2.1 Conduct survey	**7 days**	**Sat 24/7/21**	**Fri 30/7/21**
2.2.1.1 Design questionnaire	2 days	Sat 24/7/21	Sun 25/7/21
2.2.1.2 Collect data from intended users	5 days	Sun 25/7/21	Fri 30/7/21
2.2.2 Data analysis	3 days	Fri 30/7/21	Sun 1/8/21
3.0 Project Initial Specification	**20 days**	**Mon 2/8/21**	**Fri 20/8/21**
3.1 Time management	**8 days**	**Mon 2/8/21**	**Mon 9/8/21**
3.1.1 Identify tasks and activities	**3 days**	**Mon 2/8/21**	**Wed 4/8/21**
3.1.1.1 Construct works break down structure (WBS)	3 days	Mon 2/8/21	Wed 4/8/21
3.1.2 Create project duration plan	3 days	Thu 5/8/21	Sat 7/8/21
3.1.3 Construct Gantt chart	2 days	Sat 7/8/21	Mon 9/8/21
3.2 Tool management	**3 days**	**Mon 9/8/21**	**Thu 12/8/21**
3.2.1 Determine development tools	3 days	Mon 9/8/21	Thu 12/8/21
3.3 Requirement specification	**4 days**	**Thu 12/8/21**	**Sun 15/8/21**
3.3.1 Identify functional requirements	2 days	Thu 12/8/21	Sat 14/8/21
3.3.2 Identify non-functional requirements	2 days	Sat 14/8/21	Sun 15/8/21
3.4 Use case modelling	**5 days**	**Mon 16/8/21**	**Fri 20/8/21**
3.4.1 Construct use case diagram	2 days	Mon 16/8/21	Tue 17/8/21
3.4.2 Prepare use case description	3 days	Wed 18/8/21	Fri 20/8/21
4.0 Prototype Implementation	**34 days**	**Mon 24/1/22**	**Thu 24/2/22**
4.1 First Iteration	**12 days**	**Mon 24/1/22**	**Fri 4/2/22**
4.1.1 Design prototype with main function	**8 days**	**Mon 24/1/22**	**Mon 31/1/22**
4.1.1.1 Set up sample data for developing search function purpose	2 days	Mon 24/1/22	Tue 25/1/22
4.1.1.2 Search function with Euclidean distance	2 days	Wed 26/1/22	Thu 27/1/22
4.1.1.3 Search function with Manhattan distance	2 days	Fri 28/1/22	Sat 29/1/22
4.1.1.4 Search function with Minkowski distance	2 days	Sat 29/1/22	Mon 31/1/22
4.1.2 User survey	2 days	Mon 31/1/22	Wed 2/2/22
4.1.3 Improve prototype based on user's review	2 days	Wed 2/2/22	Fri 4/2/22
4.2 Second Iteration	**14 days**	**Fri 4/2/22**	**Thu 17/2/22**
4.2.1 Design Prototype with main functions	**10 days**	**Fri 4/2/22**	**Sun 13/2/22**
4.2.1.1 Search function with Cosine similarity	2 days	Fri 4/2/22	Sun 6/2/22
4.2.1.2 Search function with Jaccard similarity coefficient	2 days	Sun 6/2/22	Mon 7/2/22
4.2.1.3 Rate and review function	2 days	Tue 8/2/22	Wed 9/2/22
4.2.1.4 Raise demo class function	2 days	Thu 10/2/22	Fri 11/2/22

4.2.1.5 Book formal class function	2 days	Sat 12/2/22	Sun 13/2/22
4.2.2 User survey	2 days	Sun 13/2/22	Tue 15/2/22
4.2.3 Improve prototype based on review	2 days	Tue 15/2/22	Thu 17/2/22
4.3 Third Iteration	**10 days**	**Tue 15/2/22**	**Thu 24/2/22**
4.3.1 Design Prototype with main functions	**6 days**	**Tue 15/2/22**	**Sun 20/2/22**
4.3.1.1 Manage demo class function	2 days	Tue 15/2/22	Thu 17/2/22
4.3.1.2 Manage tutors' detail function	2 days	Thu 17/2/22	Sat 19/2/22
4.3.1.3 Register and log in function	1 day	Sat 19/2/22	Sun 20/2/22
4.3.1.4 Edit profile function	1 day	Sun 20/2/22	Sun 20/2/22
4.3.2 User survey	2 days	Mon 21/2/22	Tue 22/2/22
4.3.3 Improve prototype based on review	2 days	Wed 23/2/22	Thu 24/2/22
5.0 Development	**33 days**	**Fri 25/2/22**	**Sun 27/3/22**
5.1 Develop all functions	**14 days**	**Fri 25/2/22**	**Wed 9/3/22**
5.1.1 Main functions with proper interface linkage	7 days	Fri 25/2/22	Thu 3/3/22
5.1.2 Minor view functions with proper interface linkage	7 days	Thu 3/3/22	Wed 9/3/22
5.2 Set up real-time database	5 days	Thu 10/3/22	Mon 14/3/22
5.3 Insert data into database	5 days	Mon 14/3/22	Sat 19/3/22
5.4 Improve user interface design	5 days	Sat 19/3/22	Wed 23/3/22
5.5 Internal documentation	**4 days**	**Thu 24/3/22**	**Sun 27/3/22**
5.5.1 Clean code	4 days	Thu 24/3/22	Sun 27/3/22
6.0 Testing	**13 days**	**Sun 27/3/22**	**Fri 8/4/22**
6.1 Conduct unit testing	4 days	Sun 27/3/22	Thu 31/3/22
6.2 Conduct integration testing	3 days	Thu 31/3/22	Sun 3/4/22
6.3 Conduct system testing	2 days	Sun 3/4/22	Mon 4/4/22
6.4 Conduct usability testing	2 days	Tue 5/4/22	Wed 6/4/22
6.5 Conduct user acceptance testing	2 days	Thu 7/4/22	Fri 8/4/22
7.0 Deployment	**11 days**	**Sat 9/4/22**	**Mon 18/4/22**
7.1 Deploy production	3 days	Sat 9/4/22	Mon 11/4/22
7.2 Documentation	7 days	Mon 11/4/22	Sun 17/4/22
7.3 Presentation	1 day	Mon 18/4/22	Mon 18/4/22

3.4.3 Gantt Chart

The Gantt chart is attached in the Appendices section.

3.5 Development Tools

3.5.1 React Native

Netguru (2021) stated React Native is a JavaScript framework that runs on an open-source library called React. React Native application is written by using JavaScript and JXL (special markup code). It is widely applied in developing a native-rendered mobile application. This is because of the high code reusability in React Native, the code can be applied in powering both iOS and Android. Besides the UI native components, there are ready-to-use UI kits in React Native that enable the developers to enhance their user interface design. In this project, React Native acts as the main tool in developing the user interface of the mobile application.

3.5.2 Firebase

Firebase is a Backend as a Services (BaaS) that supports the real-time NoSQL database for React Native that provides a real-time data synchronization function. The data is stored in JSON format (Firebase, 2021a). Firebase could be a database, servers, and API that need less management of the developers (Esplin, 2016). It provides a cross-platform API with minimal setup when it is working with React Native in building an application. Besides, Firebase also acts as an authentication system that helps users sign up and authentication such as creating new user records when new users sign up (Firebase, 2021b). By applying Firebase as the database in this application, it enables the data can be synchronized and provide simple authentication of users. Firebase provides various kinds of services such as Firestore, Firebase Storage, Firebase Authentication, Firebase Cloud Messaging, and so on. In this project, Firestore is responsible as a cloud database to store the needed data in this application, the create, update, retrieve actions (CRU actions) are triggered by the server. Firebase Authentication is used to handle the sign-in and sign-up actions while Firebase Storage is used to upload users' profile pictures and retrieve profile picture links. Besides, Firebase Cloud Messaging played an important role in handling the message push notification actions when users send messages by using the application.

3.5.3 Visual Studio Code

Visual Studio Code is a popular code editor that supports different development operations such as version control, task running, and debugging. Visual Studio Code is chosen due to the extensive features that provide a development environment for React Native. The code can be debugged easily by applying the extensions.

3.5.4 Express.js

Express.js is a framework for Node.js that allows the developers in creating an API in a more convenient way and with a minimal line of codes. It is responsible as a library of a server in accepting and responding to HTTP requests from the client-side. Express.js is used to design an API in a server to manage the backend logic such as similarity measures in this application to reduce the client-side application workload while preventing the data can be retrieved easily by accessing the front-end interface.

3.5.5 Postman

Postman is a platform that provides a simple Graphical User Interface (GUI) for testing the API. The HTTP requests with parameters or raw body could be saved and reused by using the GUI without memorizing the requested link and testing with the browser. Postman is applied in API testing of this project in order to test the functionality of each API.

3.5.6 AnyDesk

AnyDesk is a remote desktop application that is closed source. This application is applied in conducting usability testing and User Acceptance Testing (UAT) by controlling the tutor finders application with a laptop or another smartphone. Besides, this application is used in doing product demonstrations by showing the phone's interface on the laptop's screen.

CHAPTER 4

PROJECT INITIAL SPECIFICATION

4.1 Introduction

The data collected from the respondent is analyzed as a fact-finding of this project. Use case modeling such as use case diagram and use case descriptions are constructed according to the fact-finding and literature review in CHAPTER 2 to formalize the function overview of this application. The functional and non-functional requirements are listed to clearly identify the basic requirements of users while the detailed requirements could be defined during the prototype iteration process.

4.2 Use Case Modelling

4.2.1 Application's Use Case Diagram

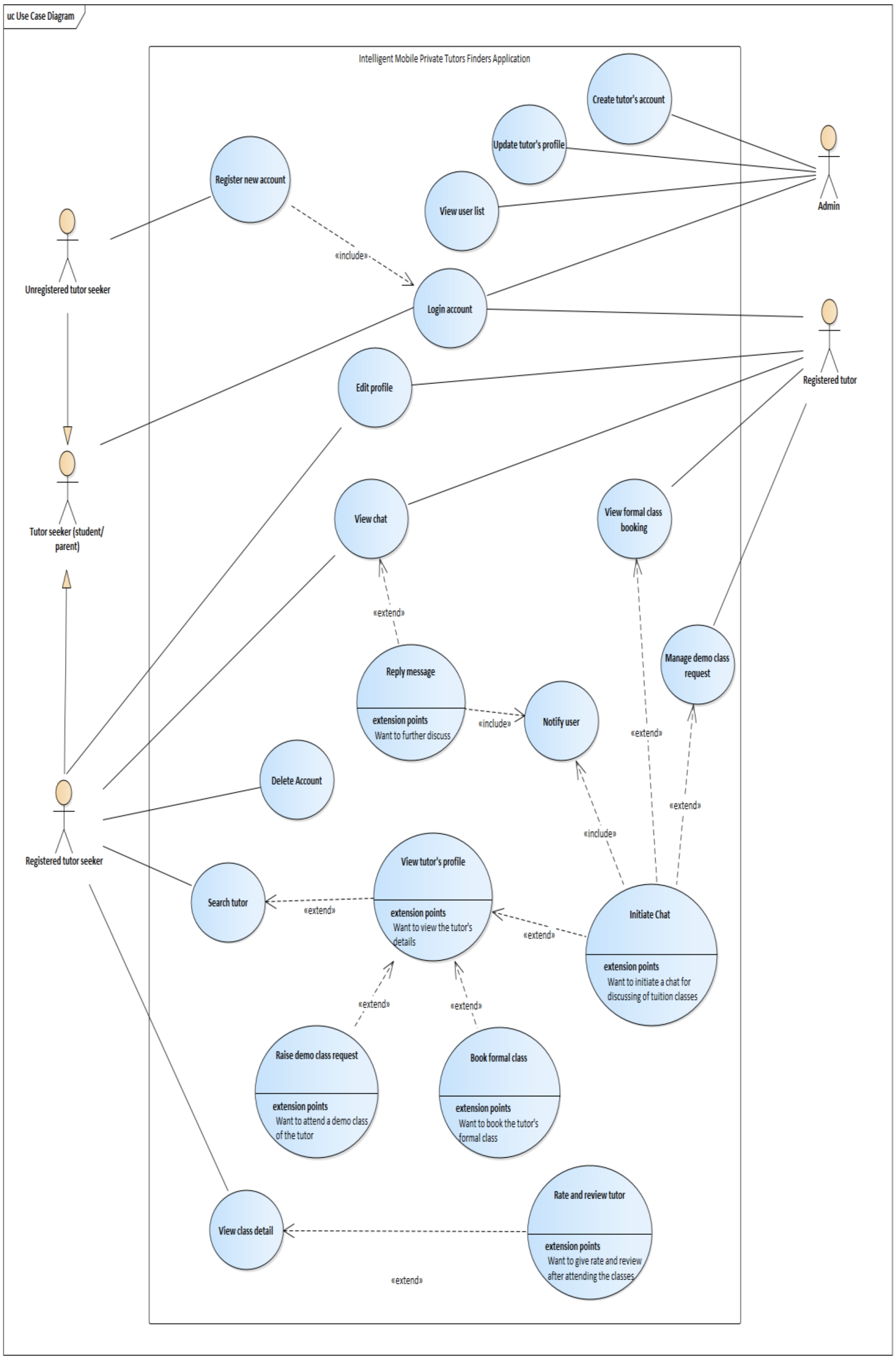


Figure 4.1: Use Case Diagram

4.2.2 Use Case Description

Table 4.1: Description of Use Case - Register New Account

Use Case: Register new account	Use Case ID: UCD-1
Related Stakeholders: Unregistered tutor seeker – Student or parent that wants to create a new account in finding a tutor.	
General Use Case Description: Describe how an unregistered tutor seeker creates a new account before starting to find a tutor by using the application.	
Triggering Activity: A unregistered tutor seeker wants to create an account to find an ideal tutor by using the application.	
Relationships between Use Case / Stakeholders: Association: Unregistered tutor seeker Include: Login account	
Basic Flow of Use Case: 1. The unregistered tutor seeker downloads the intelligent mobile private tutor finder's application. 2. The unregistered tutor seeker wants to create a new account. 3. The unregistered tutor seeker is required to enter their basic personal details, email, username, and password. If the email is present in the database Perform exceptional flow E-1. 4. A new account is created successfully, account information is created in the database. 5. Perform Login account use case.	
Exceptional Flow of Use Case: E-1 1. Unregistered tutor seeker re-enters the email and presses the "create" button again.	

Table 4.2: Description of Use Case - Login Account

Use Case: Login account	Use Case ID: UCD-2
Related Stakeholders: Tutor seeker – Student or parent that want to login a registered account in finding a tutor. Admin – Admin that wants to log in to a registered account before starting to manage tutors' details. Registered tutor – Registered tutor that wants to log in to a registered account before starting to manage the demo class requests.	
General Use Case Description: Describe how the registered tutor seeker, admin, and registered tutor to log in their registered account to achieve different purposes.	
Triggering Activity: A registered tutor seeker, admin, or registered tutor wants to log in before performing different tasks by using the application.	
Relationships between Use Case / Stakeholders: Association: Tutor seeker, Admin, Registered tutor	
Basic Flow of Use Case: 1. The registered tutor seeker, admin, or registered tutor wants to log in to their account. 2. The registered tutor seeker, admin, or registered tutor is required to enter their email and password. If the email or password are not validate Perform exceptional flow E-1. 3. The registered tutor seeker, admin, or registered tutor login successfully and be directed to their home page respectively.	
Exceptional Flow of Use Case: E-1 1. The tutor seeker, admin, or tutor re-enter the email and password.	

Table 4.3: Description of Use Case - Edit Profile

Use Case: Edit profile	Use Case ID: UCD-3
Related Stakeholders: Registered tutor seeker – Tutor seeker that registered an account manages to change his/her profile detail. Registered tutor – Tutor that registered an account manages to change his/her profile detail.	
General Use Case Description: This use case describes how the registered tutor seeker or registered tutor modifies their basic profile details.	
Triggering Activity: A registered tutor seeker or registered tutor wants to change their basic information that is outdated or incorrect.	
Relationships between Use Case / Stakeholders: Association: Registered tutor seeker, Registered tutor	
Basic Flow of Use Case: 1. The registered tutor seeker or registered tutor wants to modify his/her basic personal information in his/her profile. 2. The registered user or registered tutor press the “profile” option to view their profile details. 3. The registered user or registered tutor press the “edit” option to edit their basic personal information. 4. The registered user or registered tutor enters the new basic personal information such as contact number, username, and so on. 5. The registered tutor seeker or registered tutor presses the “save” option after changing the details. If invalid fields exist and an alert message is shown by the system. Perform exceptional flow E-1. If all fields are valid and correct. The system update the profile details to database.	
Exceptional Flows of Use Case: E-1 1. The registered tutor seeker or registered tutor reenters the invalid fields and presses on “save” button again.	

Table 4.4: Description of Use Case - Search Tutor

Use Case: Search tutor	Use Case ID: UCD-4
Related Stakeholders: Registered tutor seeker – Tutor seeker that registered an account to search their ideal tutor according to their preference.	
General Use Case Description: This use case describes how the registered tutor seeker searches their preference tutor based on their learning preference.	
Triggering Activity: A registered tutor seeker wants to search for a tutor that matches their learning preference within a short period.	
Relationships between Use Case / Stakeholders: Association: Registered user Extend: View tutor's profile	
Basic Flow of Use Case: 1. The registered tutor seeker presses the 'search' option to search for a private tutor. 2. The registered tutor seeker enters the details of their preference tutor's criteria such as teaching experiences, teaching styles, teaching languages, and so on. 3. The system displays a list of tutors that are matched the criteria selected by the registered tutor seeker in descending order of similarity (from highest similarity to lowest similarity). 4. The registered tutor seeker browses through the list. If registered tutor seeker press into a tutor's profile Perform View tutor's profile use case	

Table 4.5: Description of Use Case - View Tutor’s Profile

Use Case: View tutor’s profile	Use Case ID: UCD-5
Related Stakeholders: Registered tutor seeker – Tutor seeker that registered an account views the profile details of a tutor and performs different functions.	
General Use Case Description: This use case describes how the registered tutor seeker views the profile details of a tutor they are interested in.	
Triggering Activity: A registered tutor seeker wants to view the profile details of a tutor from the tutor list after performing the search tutor use case.	
Relationships between Use Case / Stakeholders: Extend: Raise demo class request, Book demo class, Initiate chat	
Basic Flow of Use Case: 1. The registered tutor seeker presses on the tutor profile he/her interested in from the tutor list that is displayed. 2. The system displays the profile details of the tutor such as name, education level, teaching subjects, and so on. 3. The registered user browses through the profile of the tutor. 4. If the registered user presses the “raise demo class” option Perform exceptional flow S-1. 5. If the registered user presses the “book formal class” option Perform exceptional flow S-2. 6. If the registered user presses the “chat with tutor” option Perform Initiate chat use case	
Sub Flow of Use Case: S-1 1. The system displays a registration column to the registered tutor seeker for filling in the demo class details. 2. The registered tutor seeker enters his/her preferred subject, syllabus, and approach of the demo class. 3. The registered tutor seeker presses on the “submit” option and waits for approval from the tutor. If invalid fields exist and an alert message is shown by the system. Perform exceptional flow E-1. If same class record exisit or more than 2 demo classes is raised from the the same registered tutor seeker to the registered tutor. Perform exceptional flow E-2. If all fields and validation are passed. The system create record in database.	

S-2

1. The system displays a registration column to the registered user for filling in the formal class details.
2. The registered tutor seeker enters his/her preferred syllabus, subject, approach, and the ideal fee of a formal class.
3. The registered tutor seeker presses on the “submit” option and waits for the arrangement of the tutor.
   If invalid fields exist and an alert message is shown by the system.
      Perform exceptional flow E-1.
   If same class record exisit in database.
      Perform exceptional flow E-2.
   If all fields and validation are passed.
      The system create record in database.

Exceptional Flows of Use Case:

E-1

1. The registered tutor seeker reenters the invalid fields and presses on “submit” button again.

E-2

1. The system displays related alert message to the registered tutor seeker.

Table 4.6: Description of Use Case – Initiate chat

Use Case: Initiate chat	Use Case ID: UCD-6
Related Stakeholders: Registered tutor seeker – Tutor seeker who initiates the chat with the tutor he/her interested in. Registered tutor – Registered tutor who initiate the chat with the tutor seeker who raised the classes requests.	
General Use Case Description: This use case describes how the registered tutor seeker or registered tutor initiates the chat between one to another.	
Triggering Activity: A registered tutor seeker wants to discuss the class details or tutor's teaching details with the tutor he/her interested in. A registered tutor wants to discuss the class details with the registered tutor seeker who raised the demo class or formal class request.	
Relationships between Use Case / Stakeholders: Include: Notify user	
Basic Flow of Use Case: 1. The system directs the registered tutor seeker or the registered tutor to a chat box between themselves with the selected user. 2. The registered tutor seeker or registered tutor enters a message. 3. The registered tutor seeker or registered tutor presses on the send icon to initiate the chat between one to another. 4. Perform Notify user use case.	

Table 4.7: Description of Use Case - View Class Detail

Use Case: View class detail	USE CASE ID: UCD-7
Related Stakeholders: Registered tutor seeker – Tutor seeker that registered an account to view the class details.	
General Use Case Description: This use case describes how the registered tutor seeker to view the details of the class he/her requested.	
Triggering Activity: A registered tutor seeker wants to view the status or details of the requested demo class or details of the booked formal class.	
Relationships between Use Case / Stakeholders: Association: Registered tutor seeker Extend: Rate and review tutor	
Basic Flow of Use Case: 1. The registered tutor seeker presses the 'history' option to check the classes' details. 2. The system displays the options of "demo class" and "formal class" for user selection. If registered tutor seeker swipe to "demo class" 2.1 The system displays the list of demo class requests that are raised by the registered tutor seeker with class details such as status (approved/rejected/pending), tutor's name, subject, syllabus, and so on. If registered tutor seeker presses on "rate and review" option Perform exceptional flow S-1. If registered tutor seeker swipe to "formal class" 2.2 The system displays the list of formal class bookings that are done by the registered tutor seeker with class details such as syllabus, subject, tutor's name, and so on. If registered tutor seeker presses on "rate and review" option Perform exceptional flow S-1.	
Sub Flow of Use Case: S-1 1. The system displays the tutor's profile picture, name, rate column, and review column. 2. The registered tutor seeker gives a rate and review to the chosen tutor. 3. The registered tutor seeker presses the "submit" button to upload the rate and reviews on the tutor's profile. If invalid fields exist and an alert message is shown by the system. Perform exceptional flow E-1. If all fields are valid and correct. Create or update rate and review records in the database.	
Exceptional Flows of Use Case: E-1 1. The registered tutor seeker reenters the invalid fields and presses on "submit" button again.	

Table 4.8: Description of Use Case - Manage Demo Class Request

Use Case: Manage demo class request	USE CASE ID: UCD-8
Related Stakeholders: Registered tutor – Registered tutor to manage the demo class request.	
General Use Case Description: This use case describes how the registered tutor manages the demo class requests that are raised by the registered tutor seeker.	
Triggering Activity: A registered tutor wants to manage the demo class request by rejecting or accepting it.	
Relationships between Use Case / Stakeholders: Association: Registered tutor Extend: Initiate chat	
Basic Flow of Use Case: 1. The registered tutor presses the ‘demo class’ option to check the raised demo class details and manage the requests. 2. The system displays the list of demo class requests that are raised by the registered tutor seeker. 3. The registered tutor browses through the demo class requests with basic information such as syllabus, subject, and so on. 4. The registered tutor proceeds to read the demo class details such as the registered user’s details and decides to accept, reject the requests or chat with the registered tutor seeker. If the registered tutor decides to accept the demo class request 4.1 Registered tutor presses the “accept” option. If the registered tutor decides to reject the demo class request. 4.2 Registered tutor presses the “reject” option. If the registered tutor decides to chat with the registered tutor seeker. 4.3 Perform Initiate chat use case.	

Table 4.9: Description of Use Case - View Formal Class Booking

Use Case: View formal class booking	USE CASE ID: UCD-9
Related Stakeholders: Registered tutor – Registered tutor to view the formal class booking.	
General Use Case Description: This use case describes how the registered tutor views the formal class booking that is done by the registered tutor seeker.	
Triggering Activity: A registered tutor wants to view the formal class booking that is done by the registered tutor seeker.	
Relationships between Use Case / Stakeholders: Association: Registered tutor Extend: Initiate chat	
Basic Flow of Use Case: 1. The registered user presses the ‘formal class’ option to check the formal class booking details. 2. The system displays the list of formal class booking that is raised by the registered tutor seeker. 3. The registered tutor browses through the formal class booking with basic information such as syllabus, subject, and registered user’s information. If the registered tutor decides to chat with the registered tutor seeker. 3.1 Perform Initiate chat use case.	

Table 4.10: Description of Use Case - View Chat

Use Case: View chat	USE CASE ID: UCD-10
Related Stakeholders: Registered tutor seeker – Registered tutor seeker to view the chat list. Registered tutor – Registered tutor to view the chat list.	
General Use Case Description: This use case describes how the registered tutor seeker or the registered tutor views his/her chat list with registered tutors or registered tutor seekers respectively.	
Triggering Activity: A registered tutor seeker or registered tutor wants to check their chat list and reply to the new messages after being acknowledged by the push notification.	
Relationships between Use Case / Stakeholders: Association: Registered tutor seeker, Registered tutor Extend: Reply message	
Basic Flow of Use Case: 1. The registered tutor seeker or the registered tutor presses the 'chat' option to check the recent chat lists. 2. The system displays the list of chats that are arranged by message sending time in descending order (from the latest to the oldest). 3. The registered tutor seeker or the registered tutor browses through the chat list. If the registered tutor seeker or registered tutor decides to search specified user chatbox by name. 3.1 The registered tutor seeker or registered tutor enters the name of the specified user. 3.2 The system displays the specified user chat column on the chat list. 3.3 The registered tutor seeker or registered tutor selects the chat column. 3.4 Perform Reply message use case If the registered tutor seeker or registered tutor found a specified user's message. 3.5 The registered tutor seeker or registered tutor selects the chat column. 3.6 Perform Reply message use case.	

Table 4.11: Description of Use Case – Reply Message

Use Case: Reply message	USE CASE ID: UCD-11
Related Stakeholders: Registered tutor seeker – Registered tutor seeker to reply to the chat from other users. Registered tutor – Registered tutor to reply to the chat from other users.	
General Use Case Description: This use case describes how the registered tutor seeker or the registered tutor reply his/her chat from other users.	
Triggering Activity: A registered tutor seeker or registered tutor wants to reply to the new messages after being acknowledged by the push notification.	
Relationships between Use Case / Stakeholders: Include: Notify user	
Basic Flow of Use Case: 1. The system directs the registered tutor seeker or the registered tutor to the chatbox of himself/herself with the selected user. 2. The registered tutor seeker or the registered tutor reads the messages from the respective user in the chatbox. 3. The registered tutor seeker or the registered tutor enters the reply messages and presses send icon to reply to the respective user. 4. Perform Notify user use case.	

Table 4.12: Description of Use Case – Notify User

Use Case: Notify user	USE CASE ID: UCD-12
Related Stakeholders: Registered tutor seeker – Registered tutor seeker receives the push notification from the application due to incoming messages. Registered tutor – Registered tutor receives the push notification from the application due to incoming messages.	
General Use Case Description: This use case describes how the registered tutor seeker or the registered tutor receives their push notification on the phone when a message is sent to his/her account.	
Triggering Activity: A message is sent to the registered tutor seeker or registered tutor by the registered tutor or the registered tutor seeker respectively via the application.	
Relationships between Use Case / Stakeholders:	
Basic Flow of Use Case: 1. The system sends a message to the registered tutor seeker or registered tutor. 2. The system shows a push notification on the registered tutor seeker's or the registered tutor's phone with the message and sender's name displayed.	

Table 4.13: Description of Use Case - Create Tutor’s Account

Use Case: Create tutor’s account	USE CASE ID: UCD-13
Related Stakeholders: Admin – Admin is responsibles to create an account for the new tutor.	
General Use Case Description: This use case describes how the admin creates a tutor account.	
Triggering Activity: An admin wants to create a tutor account for the newly participated tutor on this platform after verifying their professional and teaching information.	
Relationships between Use Case / Stakeholders: Association: Admin	
Basic Flow of Use Case: 1. The admin presses the “add tutor” option to create an account for the tutor. 2. The admin uploads the profile picture that is submitted by the new tutor. 3. The admin enters the registered tutor’s information which is verified accordingly. 4. The admin presses the “create” button to create a new account for the registered tutor. If invalid fields exists and an alert message is shown by the system. Perform exceptional flow E-1. If all fields are valid and correct. The record is created in the database.	
Exceptional Flows of Use Case: E-1 1. The admin reenters the invalid fields and presses on “create” button again.	

Table 4.14: Description of Use Case – Update Tutor's Profile

Use Case: Manage tutor's profile	USE CASE ID: UCD-14
Related Stakeholders: Admin – Admin is responsibles to update the registered tutor's profile.	
General Use Case Description: This use case describes how the admin updates the registered tutor's profile.	
Triggering Activity: An admin wants to update the professional information of the registered tutor in their profiles after the information is verified.	
Relationships between Use Case / Stakeholders: Association: Admin	
Basic Flow of Use Case: 1. The admin presses the "update tutor" option to update the registered tutor profile. 2. The system displays a list of registered tutors with some basic information as a profile column such as name, email, contact number, and so on. 3. The admin searches the specified registered tutor by entering the tutor's name. 4. The system shows the matched tutor profile columns as a list. 5. The admin presses on the profile column. 6. The system directs to the screen that shows the existing professional information of the chosen tutor. 7. The admin enters the updated information. 8. The admin presses the "save" button to update the professional information of the tutor on his/her tutor profile. If invalid fields exists and an alert message is shown by the system. Perform exceptional flow E-1. If all fields are valid and correct. The record is created in the database.	
Exceptional Flows of Use Case: E-1 1. The admin reenters the invalid fields and presses on "save" button again.	

Table 4.15: Description of Use Case - View User List

Use Case: View user list	USE CASE ID: UCD-15
Related Stakeholders: Admin – Admin responsible to manage and maintain the system by viewing the user's information during customer feedback or support process.	
General Use Case Description: This use case describes how the admin views the registered tutor seekers' information.	
Triggering Activity: A registered tutor wants to retrieve the information of a registered user to conduct a customer support process.	
Relationships between Use Case / Stakeholders: Association: Admin	
Basic Flow of Use Case: 1. The admin presses the "student list" option to check the registered tutor seeker's information. 2. The system displays the search bar and list of registered tutor seekers with personal information such as name, id, contact number, and so on. 3. The admin browses through the list. If the admin wants to search a specified registered user 3.1 The admin enters a "search key" in the search bar which is the name. 3.2 The system displays the matched registered user details in a list.	

Table 4.16: Description of Use Case – Delete Account

Use Case: Delete Account	USE CASE ID: UCD-16
Related Stakeholders: Registered Tutor Seeker – Registered tutor seeker who does not want to use this application anymore.	
General Use Case Description: This use case describes how the registered tutor seeker deletes their existing account on this application.	
Triggering Activity: A registered tutor wants to delete an existing account.	
Relationships between Use Case / Stakeholders: Association: Registered tutor seeker	
Basic Flow of Use Case: 1. The registered tutor seeker presses the "profile" option to view their profile details. 2. The registered tutor seeker presses the "delete account" button to delete the existing account 3. The system displays an alert message to make double confirmation with the registered tutor seeker. 4. The registered tutor seeker presses the "Yes" button. 5. The system deletes the existing user in the database and authentication account list. 6. The system displays an alert message "your account has been deleted" and directs the tutor seeker to the login screen.	

4.3 System Requirements

4.3.1 Functional Requirements

1) The system shall allow the registered tutor seeker (student and parent) to search the tutors according to his/her preference such as teaching experiences, teaching styles, teaching languages, education level, teaching approach, and so on.
2) The system shall allow the registered tutor seeker to view the tutor profiles' details such as rates, reviews, basic personal information, and so on.
3) The system shall allow the registered tutor seeker to raise a demo class request to the tutor he/her interested in.
4) The system shall allow the registered tutor seeker to book a formal class.
5) The system shall allow the registered tutor seeker to view the class details which are the demo class requests' information and formal class bookings' information.
6) The system shall allow the registered tutor seeker to post a rate and review of a registered tutor after attending the class.
7) The system shall allow the registered tutor to accept or reject a demo class request that is raised by the registered tutor seeker.
8) The system shall allow the registered tutor to view the formal class booking that is requested by the registered tutor seeker.
9) The system shall allow the admin to view the registered user list.
10) The system shall allow the admin to manage and update the registered tutors.
11) The system shall allow the registered tutor seeker and registered tutor to view and edit their basic personal profile details.
12) The system shall allow the registered tutor seeker and registered tutor to chat to discuss their class details such as price, time, and so on.
13) The system shall send a push notification to the phone of the registered tutor seeker or registered tutor once the message is sent from the others to him/her.
14) The system shall allow first-time tutors seeker to register a new account.
15) The system shall allow users to log in with their username and password.

4.3.2 Non-functional Requirements

1) The system shall be able to be used by the users easily without conducting a training session.
2) The system shall be accessible 24 hours a day and 7 days a week with a stable internet connection.
3) The system shall protect the users' information under the privacy policy and not leaked to unrelated parties without users' consent.

4.4 Fact Findings

4.4.1 Overview of Responses

A survey is conducted to investigate and analyze the main concern criteria of users in choosing an ideal private tutor. The questionnaire had been sent to intended users (students and parents) with different backgrounds such as different ages, education levels, marital status, and so on. There are 50 respondents are involved in this survey. The overview results of some main questions are briefly explained below.

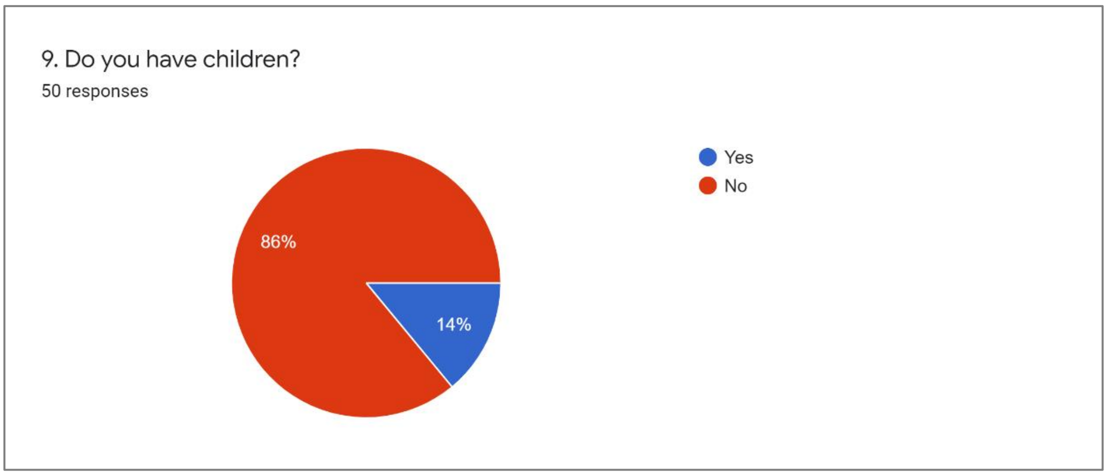


Figure 4.2: Distribution of Parents and Students

The main goal of this application is to help the students in finding the ideal tutors that are more concerned with the students' preferences instead of parents. Therefore, 86% of the respondents is a student while 14% of the respondents is a parent that has children.

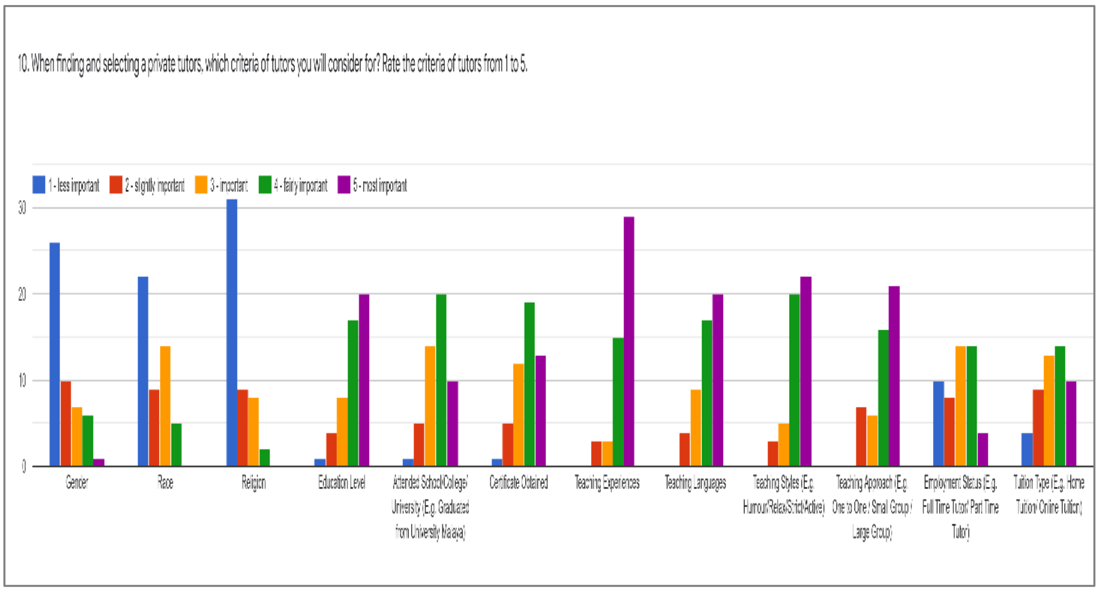

10. When finding and selecting a private tutors, which criteria of tutors you will consider for? Rate the criteria of tutors from 1 to 5.

Figure 4.3: Tutor Selection Criteria (Tutors' Criteria)

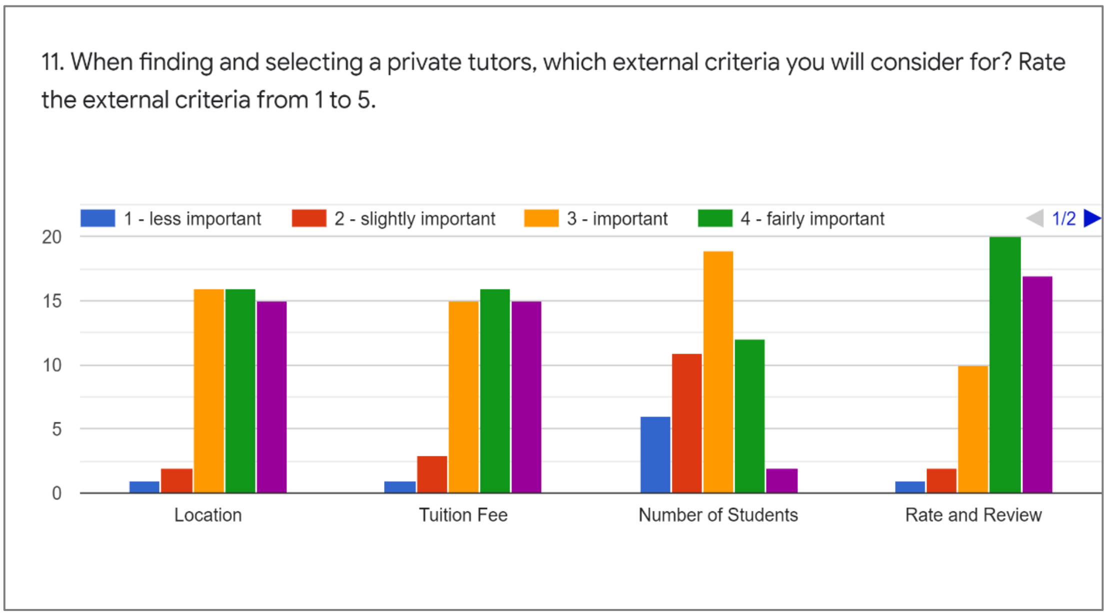

11. When finding and selecting a private tutors, which external criteria you will consider for? Rate the external criteria from 1 to 5.

Figure 4.4: Tutor Selection Criteria (External Criteria)

Table 4.17: Count of Rating in Tutor Criteria Selection

Criteria	Count of Rating (Number of Respondents)				
	1 Less Importan t	2 Slightly importan t	3 Importan t	4 Fairly Importan t	5 Most Importan t
Gender	26	10	7	6	1
Race	22	9	14	5	0
Religion	31	9	8	2	0
Education Level	1	4	8	17	20
Attended School/College/Universit y	1	5	14	20	10
Certificate Obtained	1	5	12	19	13
Teaching Experiences	0	3	3	15	29
Teaching Languages	0	4	9	17	20
Teaching Styles	0	3	5	20	22
Teaching Approach	0	7	6	16	21
Employment Status	1	8	14	14	4
Tuition Type	4	9	13	14	10
Location	1	2	16	16	15
Tuition Fee	1	3	15	16	15
Number of Students	6	11	19	12	2
Rate and Review	1	2	10	20	17

From the count of rating of different criteria by the respondents, most of the respondents treat gender, race, and religion as the less important criteria that they will not consider in selecting their ideal tutors. In contrast, the teaching experiences, teaching languages, teaching styles, teaching approaches, and the rate and review are the main concern of respondents while selecting their ideal tutors.

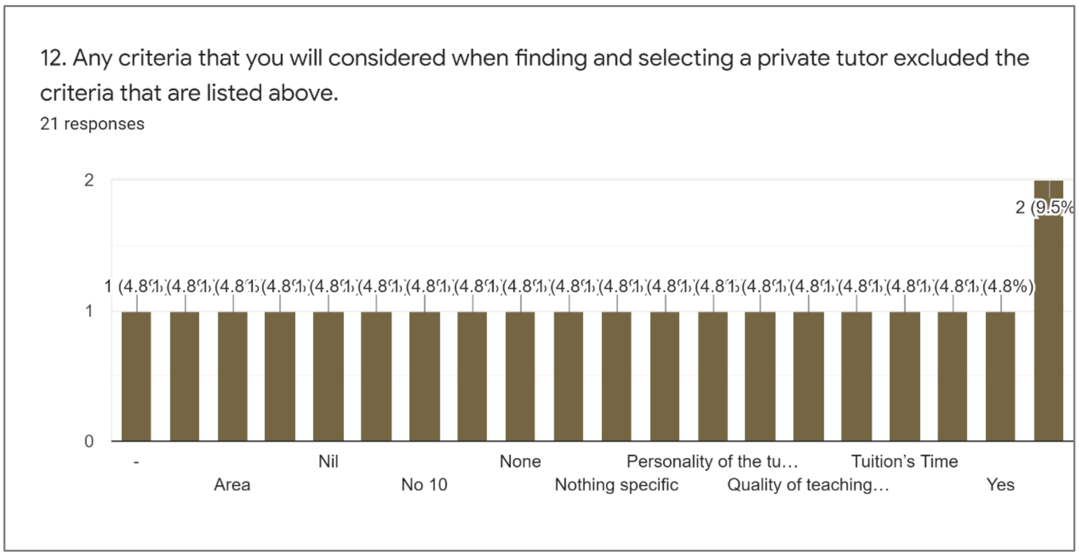


Figure 4.5: Tutor Selection Criteria (External Criteria)

There are some new criteria (excluded from the list) are suggested by the respondents that they always consider while selecting a tutor are as follow:

i. Environment condition (clean and comfortable environment)
ii. Patience of tutors
iii. Personality and reliability of tutors
iv. Quality of teaching material that is provided by the tutors
v. Tuition time match the schedule of student

4.4.2 Data Analysis

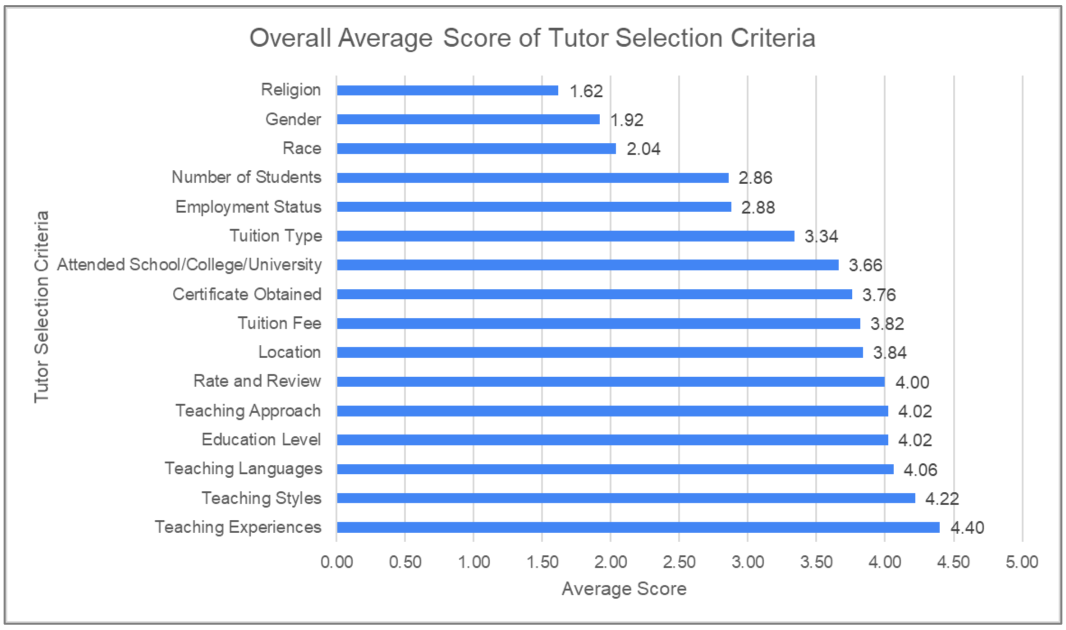


Figure 4.6: Overall Average Score of Tutor Selection Criteria

The average score (both parents and students respondents) of each criterion is calculated and displayed in a bar chart in ascending order. According to Figure 4.5, teaching experiences get the highest average score of 4.40 which indicates that most of the respondents treat the teaching experiences criteria as the fairly important criteria in selecting the ideal tutors. There are 6 criteria that get an average score of more than 4.0, most of the criteria are related to the teaching aspects of the tutors such as teaching language, teaching approaches, and teaching styles. This indicates that most of the respondents treat the teaching aspect as the main concern in finding a tutor instead of the background of the tutors and the external factors.

Besides that, an analysis of tutor selection criteria is conducted by separating the average score of parents and students to investigate the main concern of different roles in selecting the ideal tutors. Sometimes, the main concern of parents may be different with the students (children) in selecting an ideal tutor.

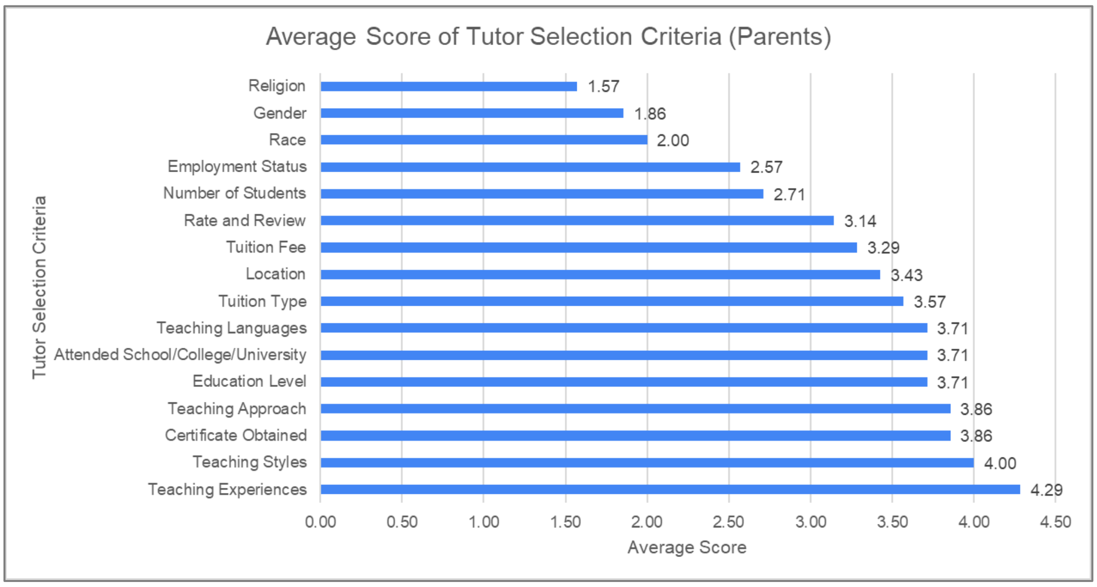


Figure 4.7: Average Score of Tutor Selection Criteria (Parents)

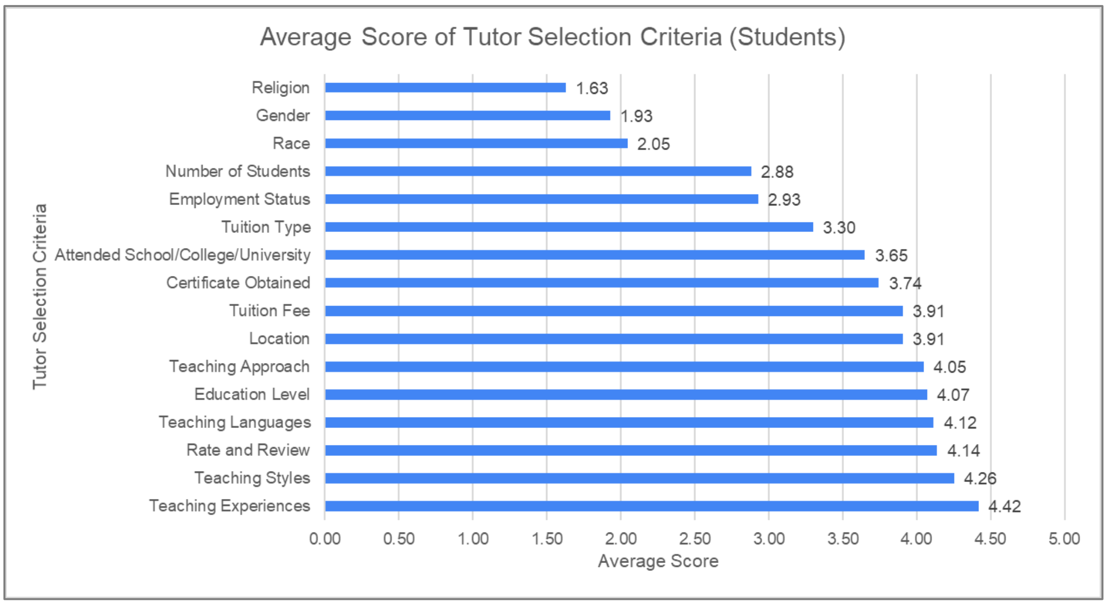


Figure 4.8: Average Score of Tutor Selection Criteria (Students)

As a comparison between the top 6 criteria of students and parents in selecting an ideal tutor, students tend to concern about the teaching aspect of the tutors. However, parents are not only concern about the teaching aspect but also the educational background of the tutors such as certificate obtained, education level, and attended institution. The criteria od certificate obtained and attended institution are located at the 9th and 10th places from the students' perspectives respectively while it located at the 3rd and 6th places from the parent's perspective.

In conclusion, the top 11 overall criteria will be selected and implemented in the searching function of this application to fulfill both requirements of parents and students in finding their ideal tutors. The selected criteria are teaching experiences, teaching styles, teaching languages, education level, teaching approach, rate and review, location, tuition fee, certificate obtained, attended school/college/university, and tuition type.

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

The system design is shown and explained in this chapter from frontend to backend. The first part includes the system architecture design to explain the flow of data and activity between one to another in the system while followed by the explanation of data model design that describes the data structure and data type in the database. The last part involves the displays of the user interface which is the frontend design of this application.

5.2 System Architecture Design

5.2.1 Three-tier Architecture

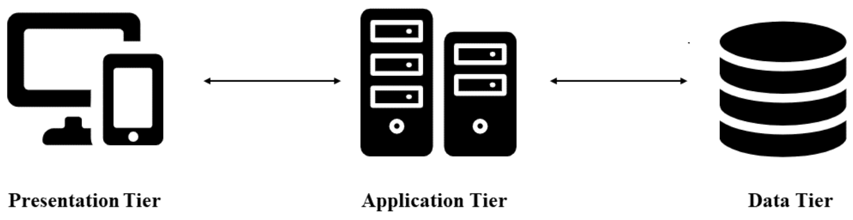


Figure 5.1: Three-tier Architecture Diagram

A three-tier architecture design is applied in this application in performing the activities. Three-tier architecture is a software architecture that distributes the application into three physical and logical computing tiers which are presentation tier, application tier, and data tier. The presentation tier is considered as the communication layer and interaction layer between the users and system which is the user interface of the application. The presentation tier is significant in displaying the information to users while also collecting the information (user input) to the application and data tier. The application tier is being called the logic or middle tier which is the main tier in processing the collected information from the presentation tier by applying the business logic. At the same time, the data create, retrieve, update, delete (CRUD) actions in the data tier also be called in this tier. Therefore, the application tier is called the heart of the application. Next, the data tier is also known as the database tier or backend which is important in storing and managing the processed information from the application tier (IBM Cloud Education, 2020).

The main objectives in applying three-tier architecture in this application are security improvement and performance improvement. Three-tier architecture improved the security of this application by preventing the direct commutation between the user interface with the database in order to eliminate SQL injections and other malicious exploits. The similarity measure calculation API compares the similarity of the ideal tutor of the user with all tutors in the database. If this action is performed in the presentation tier will have a high risk of letting malicious hackers get all the tutor's information on this platform. Besides, the three-tier architecture improved the application's performance by reducing the workloads of users' phones. The similarity measure calculation involves many processes in calculating the similarity metrics of all tutors in the database, the heavy workload will be increased with the increase of registered tutors on this platform which may cause low performance in application in the future such as low speed in getting a searching result.

5.2.2 System Architecture Diagram

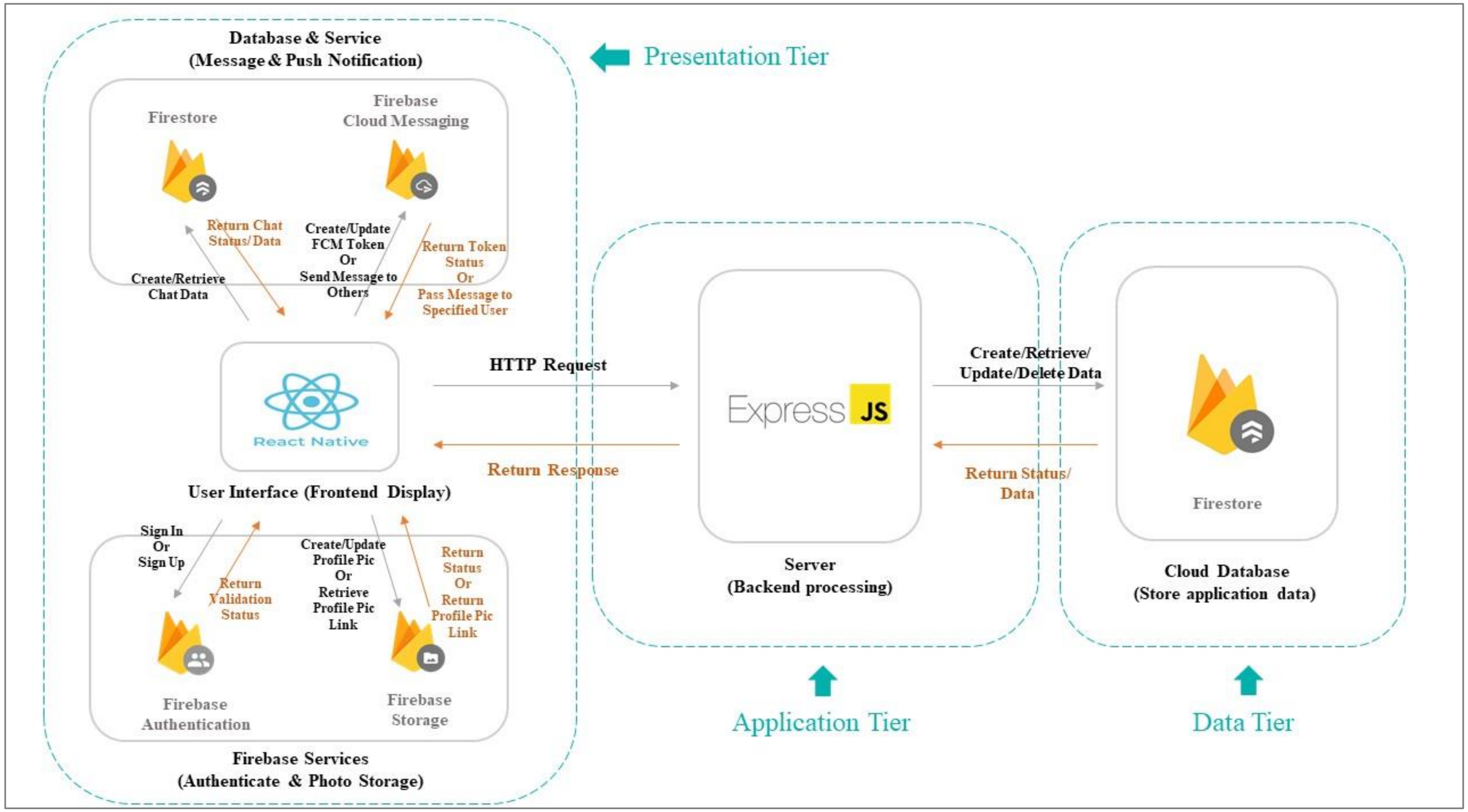


Figure 5.2: System Architecture Diagram

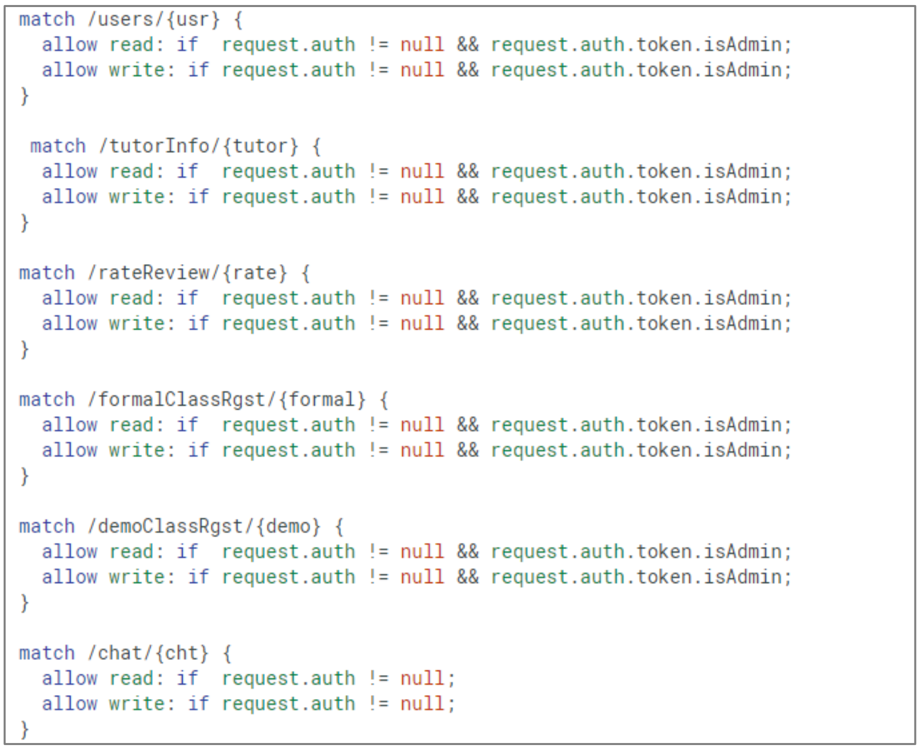
This application applied the three-tier architecture in performing the activities. React Native is used to develop a user interface which is the presentation tier in this application. Express.js is implemented as the application tier which performs the business logic in triggering the create, retrieve and update (CRU) actions while also applying algorithms in calculating the similarity measure calculation. Firestore is chosen as the data tier in storing and managing the application's data such as users', tutors, classes' information that will be listed in the following sections.

Firebase authentication serves as a third-party service in performing the account validation action in this application. Firebase authentication receives the email and password from the presentation tier and checks whether the email and password are valid or invalid. The validation status will be sent back to the presentation tier. Firebase also provides Firebase Storage as a service in storing photos instead of storing the photo in Firestore. Therefore, the functions that involve photo uploading use this service at the presentation tier to upload the photo and get the photo as a link such as the sign-up and edit profile function in this application.

Users who are authenticated could perform their activities via the presentation tier. The HTTP requests are sent to the application tier (server) to call the Application Programming Interface (API) in performing the real actions. The server could perform CRU actions toward the data tier while the data tier returns the required data or CRU status back to the server to indicate whether the action is completed or errors are raised. Last, the server returns the result as a response in JSON format back to the presentation tier which can be displayed as an action's result to the users.

For the chat and push notification functions in this application, which involves the applying of Firestore and Firebase Cloud Messaging (FCM) service. The presentation tier could only access the chat collection in Firestore to create and retrieve chat documents. The presentation tier could trigger the creation and update FCM token actions while also sending and receiving a message by using the FCM service. Therefore, FCM can be used to receive a message while Notifee (a library) displays the push notification on the receiver's phone. The details of chat implementation will be explained in chapter 6.

The third-party services are linked with the presentation tier directly without going through the application tier which could improve the performance of the services and application while security issues are handled by the third-party service without direct accessing our data tier. However, the chat collection in the data tier is directly linked with the presentation tier in creating and retrieving the chat data due to the efficiency of the application. The listener in the presentation tier should keep listening to the oncoming chat and reload the chatbox displays cause the frequent accessing the chat collection in the database. Therefore, it is not efficient to call API continuously to listen and display the ongoing chatting data. In order to keep the efficiency and security of the application, the firestorm rules are identified to deny any read and write from the presentation tier towards all collections except chat collection.



```
match /users/{usr} {
  allow read: if  request.auth != null && request.auth.token.isAdmin;
  allow write: if request.auth != null && request.auth.token.isAdmin;
}

 match /tutorInfo/{tutor} {
  allow read: if  request.auth != null && request.auth.token.isAdmin;
  allow write: if request.auth != null && request.auth.token.isAdmin;
}

match /rateReview/{rate} {
  allow read: if  request.auth != null && request.auth.token.isAdmin;
  allow write: if request.auth != null && request.auth.token.isAdmin;
}

match /formalClassRgst/{formal} {
  allow read: if  request.auth != null && request.auth.token.isAdmin;
  allow write: if request.auth != null && request.auth.token.isAdmin;
}

match /demoClassRgst/{demo} {
  allow read: if  request.auth != null && request.auth.token.isAdmin;
  allow write: if request.auth != null && request.auth.token.isAdmin;
}

match /chat/{cht} {
  allow read: if  request.auth != null;
  allow write: if request.auth != null;
}
```

Figure 5.3: Firestore Access Rules

Authenticated users from the presentation tier only can access chat collection in Firestore while using the chat function. Admin right is needed in accessing other collections while admin right only is assigned in the server. In simple words, authenticated users from the presentation tier do not have the right in accessing directly the other collections' data while also the efficiency in creating and retrieving chat data can be conserved.

5.2.3 Context Diagram

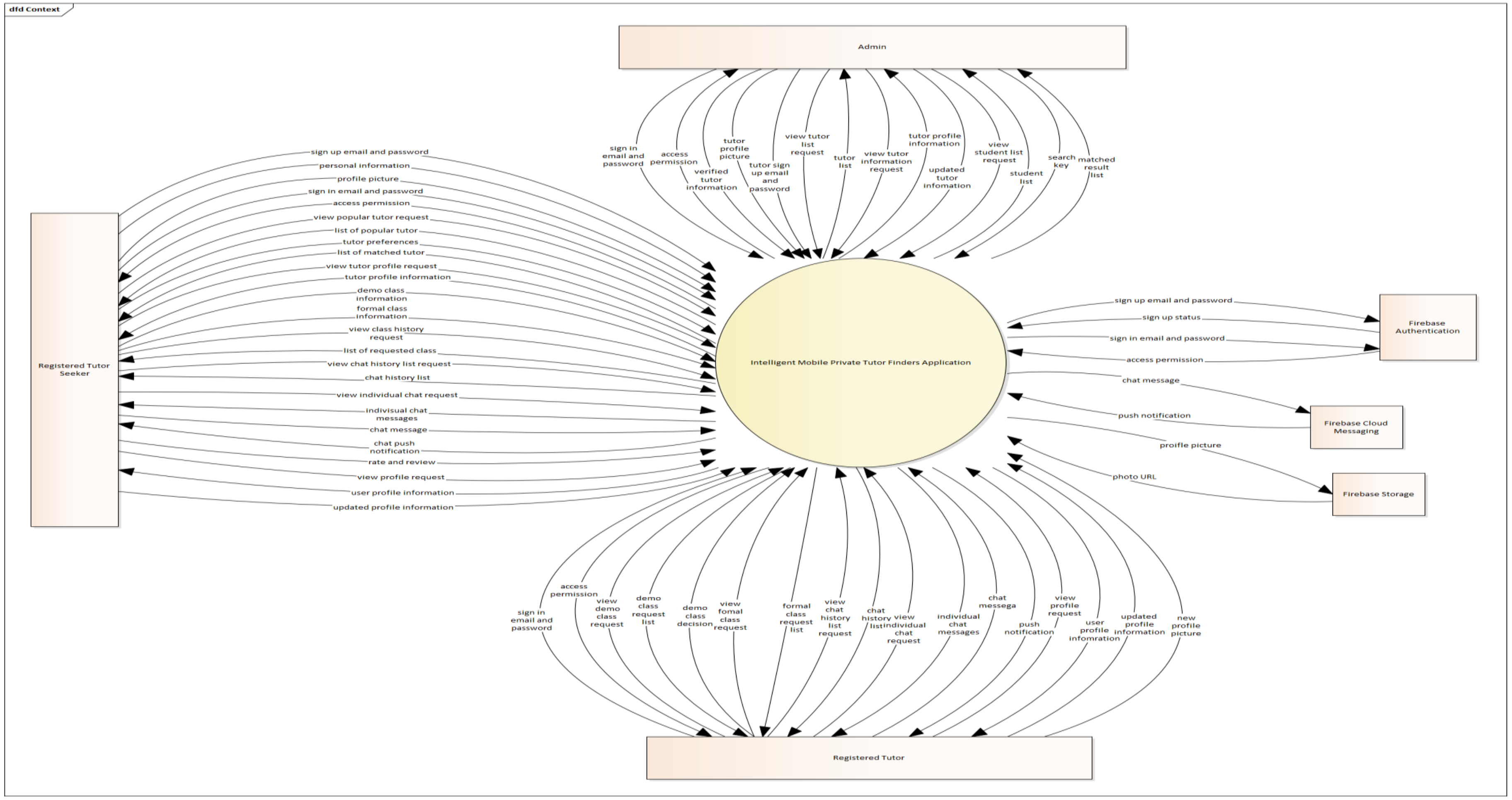


Figure 5.4: Context Diagram

5.2.4 Data Flow Diagram

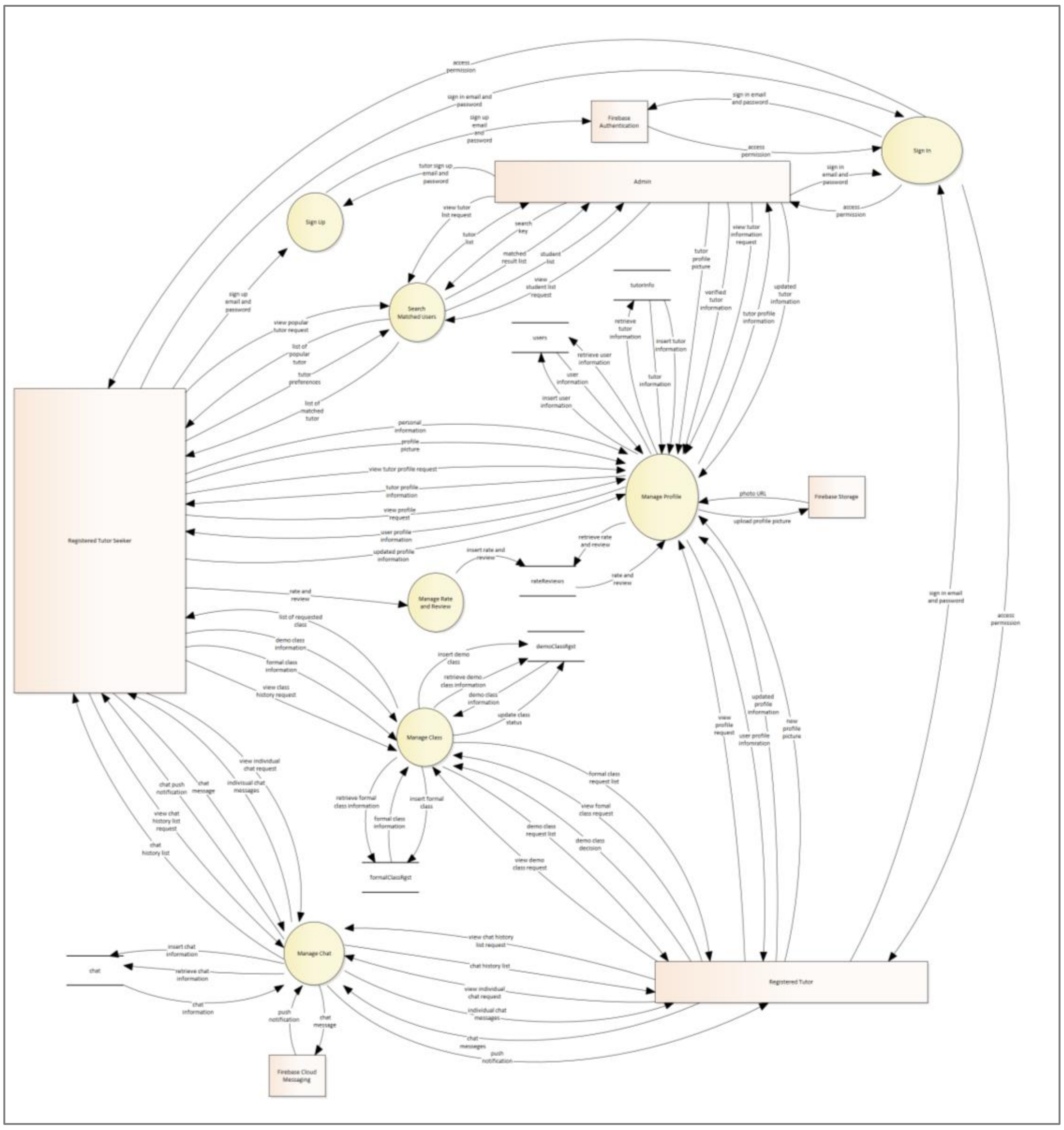


Figure 5.5: DFD - Level 0

5.2.5 Activity Diagram

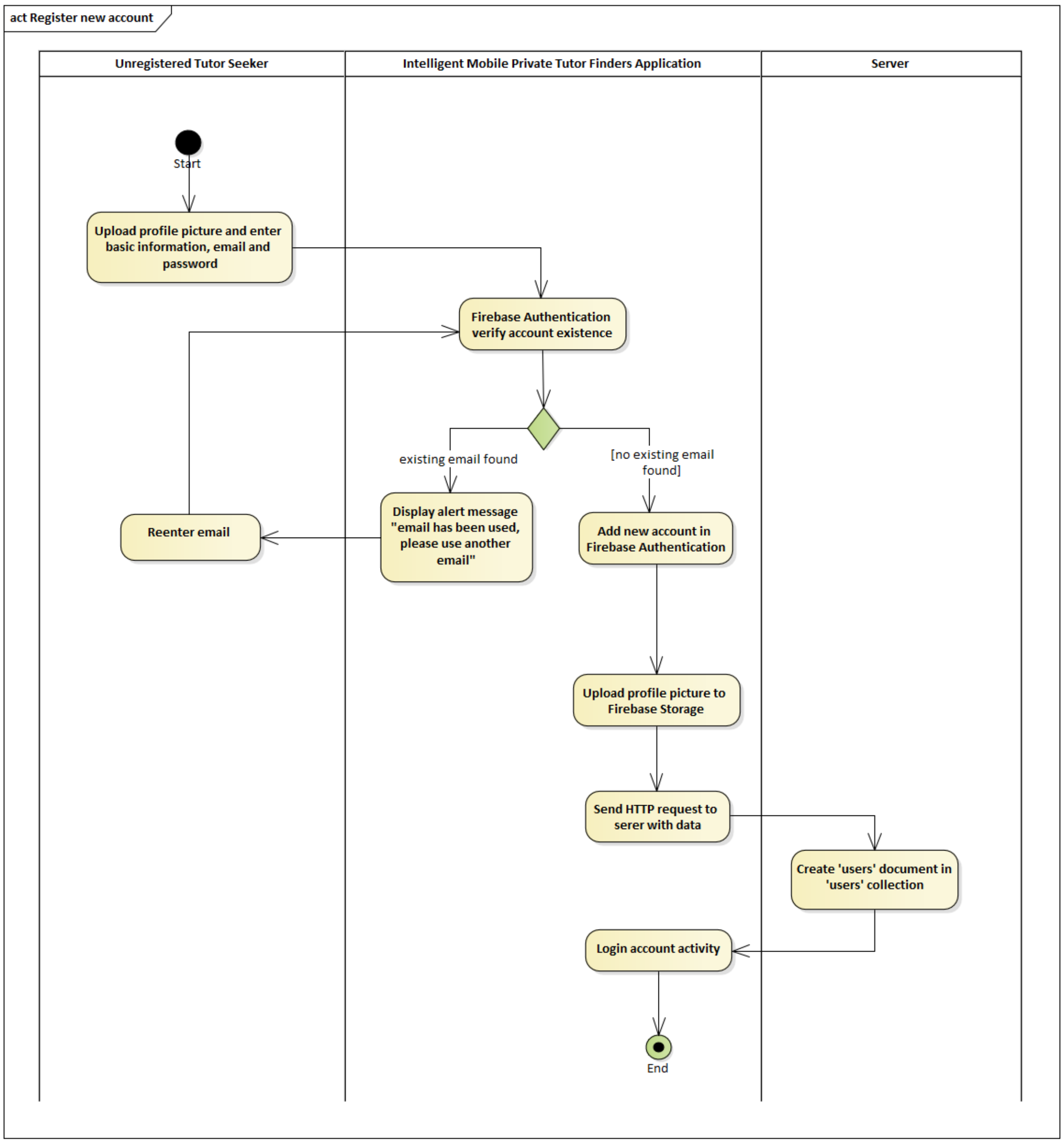

act Register new account
Unregistered Tutor Seeker
Intelligent Mobile Private Tutor Finders Application
Server
Start
Upload profile picture and enter basic information, email and password
Firebase Authentication verify account existence
existing email found
[no existing email found]
Reenter email
Display alert message "email has been used, please use another email"
Add new account in Firebase Authentication
Upload profile picture to Firebase Storage
Send HTTP request to serer with data
Create 'users' document in 'users' collection
Login account activity
End

Figure 5.6: Activity Diagram – Register New Account

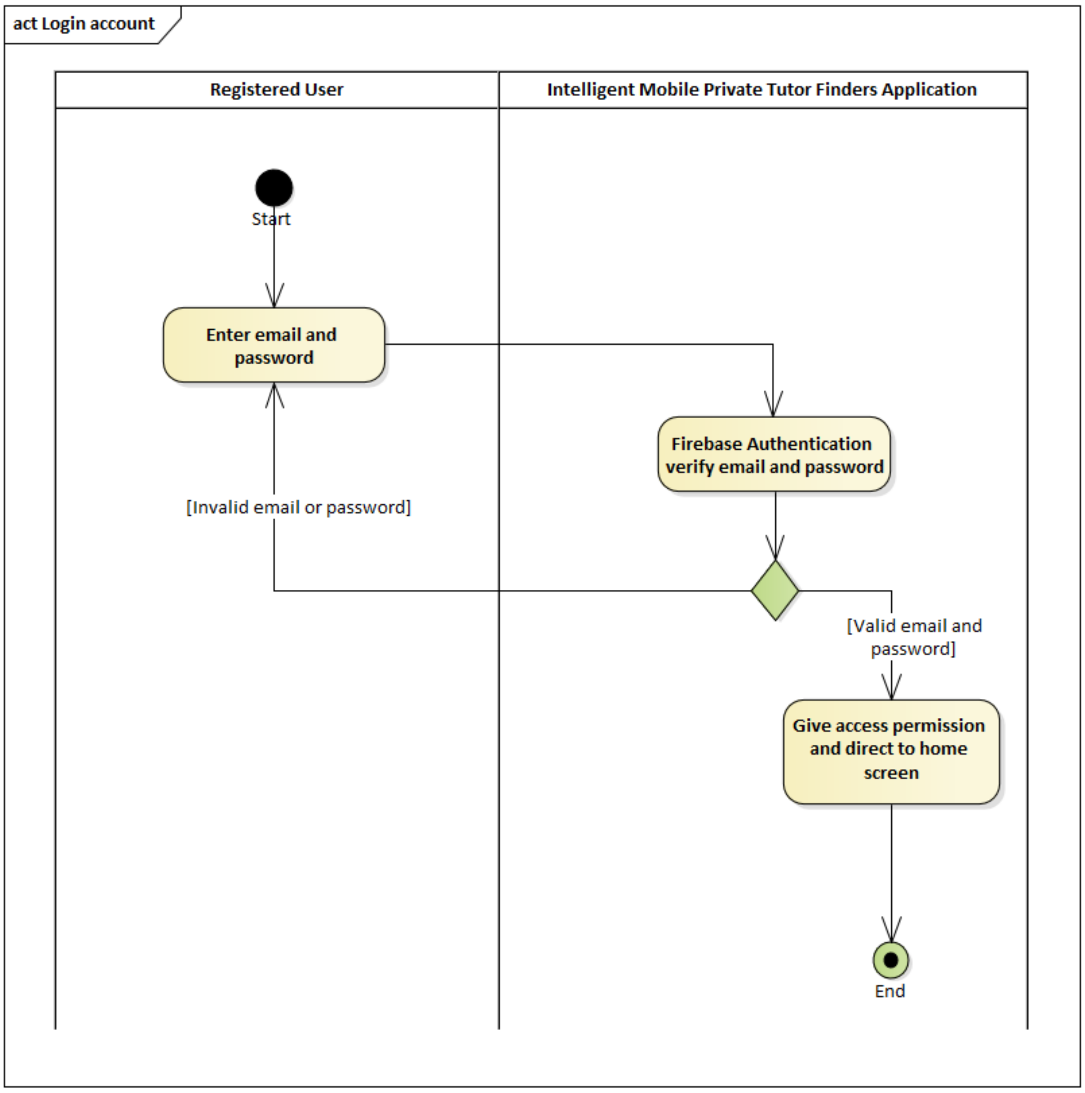

act Login account
Registered User
Intelligent Mobile Private Tutor Finders Application
Start
Enter email and password
Firebase Authentication verify email and password
[Invalid email or password]
[Valid email and password]
Give access permission and direct to home screen
End

Figure 5.7: Activity Diagram – Login Account

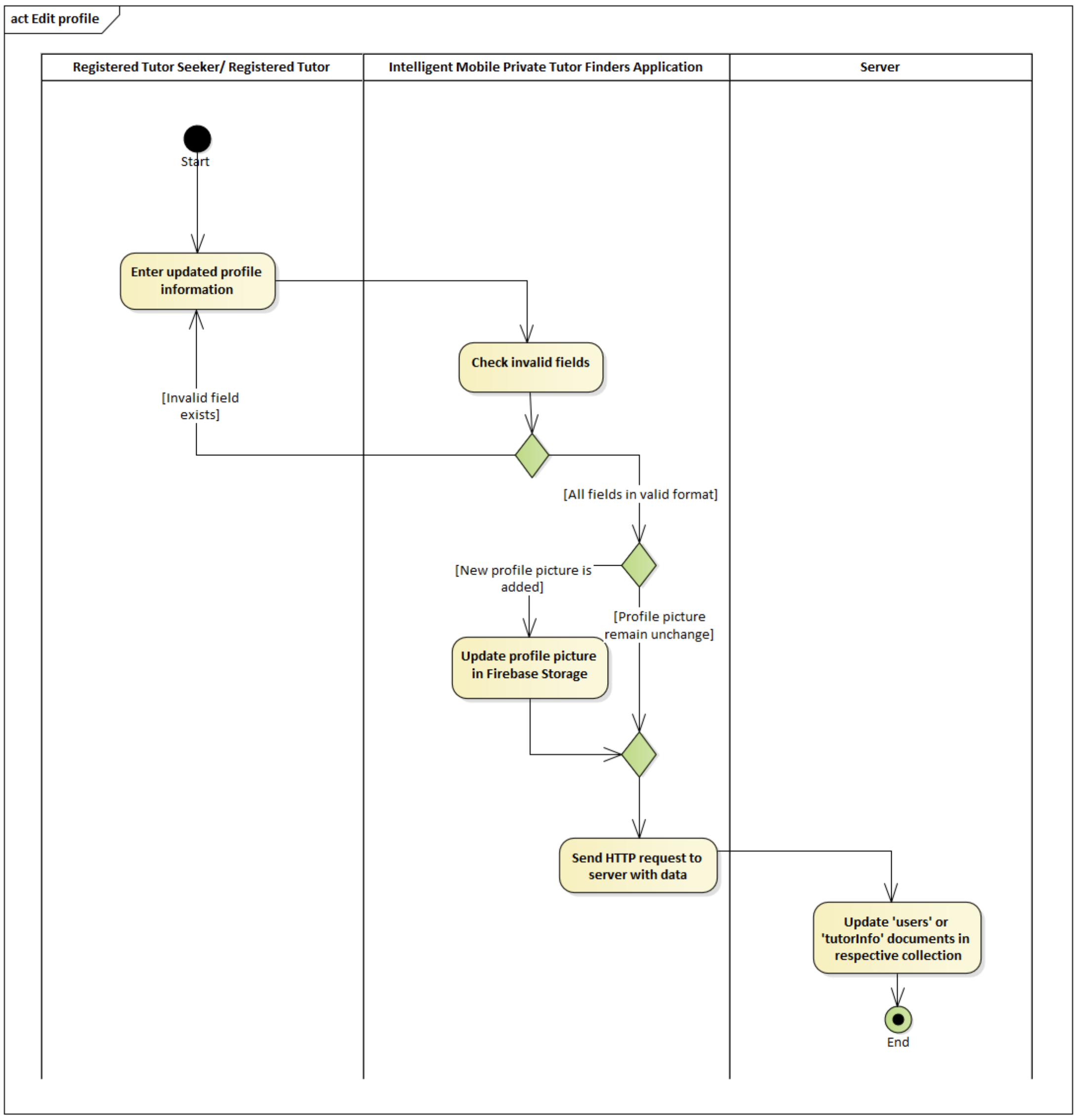

act Edit profile
Registered Tutor Seeker/ Registered Tutor
Intelligent Mobile Private Tutor Finders Application
Server
Start
Enter updated profile information
Check invalid fields
[Invalid field exists]
[All fields in valid format]
[New profile picture is added]
[Profile picture remain unchange]
Update profile picture in Firebase Storage
Send HTTP request to server with data
Update 'users' or 'tutorInfo' documents in respective collection
End

Figure 5.8: Activity Diagram – Edit Profile

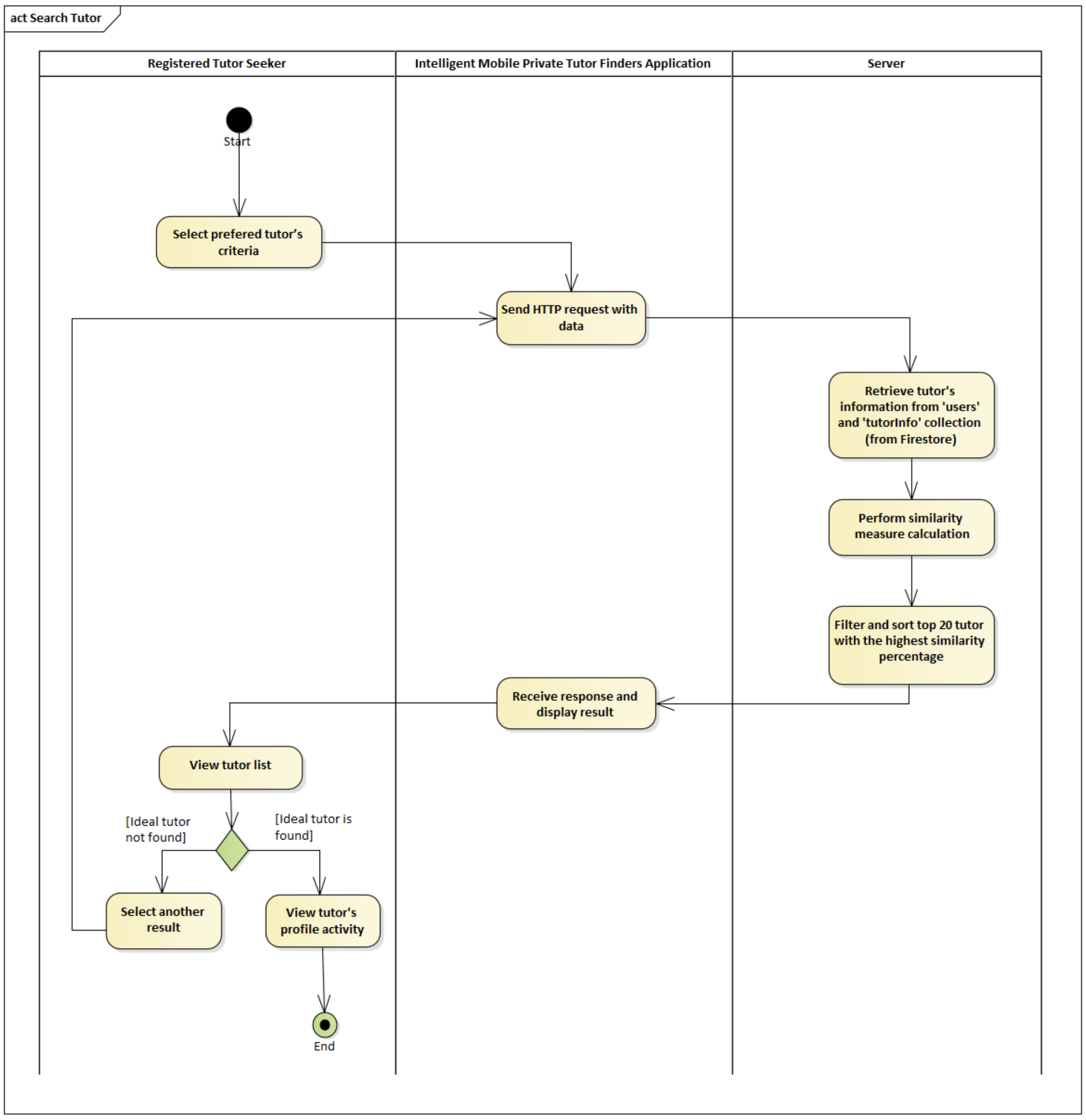

act Search Tutor
Registered Tutor Seeker
Intelligent Mobile Private Tutor Finders Application
Server
Start
Select prefered tutor's criteria
Send HTTP request with data
Retrieve tutor's information from 'users' and 'tutorInfo' collection (from Firestore)
Perform similarity measure calculation
Filter and sort top 20 tutor with the highest similarity percentage
Receive response and display result
View tutor list
[Ideal tutor not found]
[Ideal tutor is found]
Select another result
View tutor's profile activity
End

Figure 5.9: Activity Diagram – Search Tutor

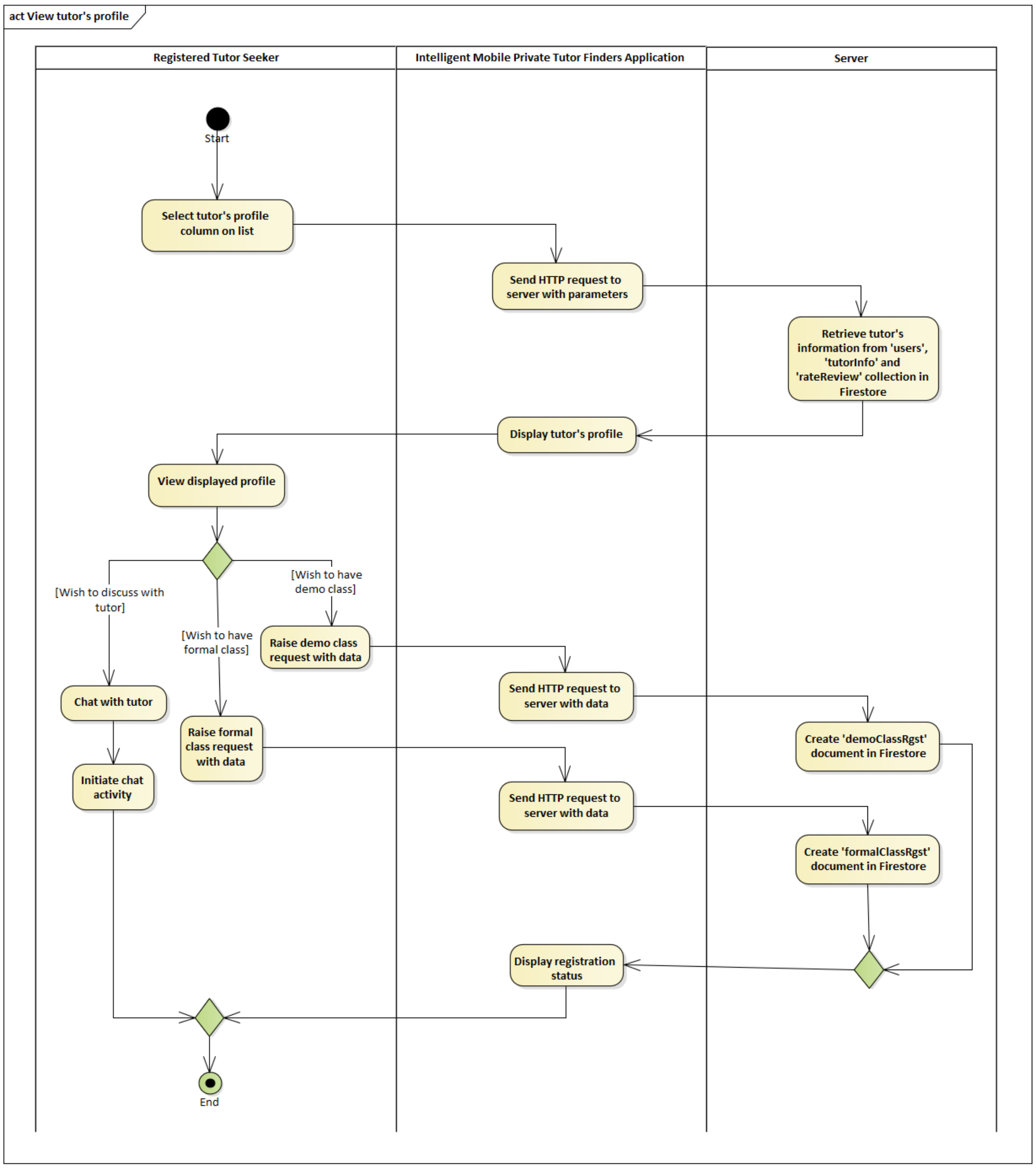

act View tutor's profile
Registered Tutor Seeker
Intelligent Mobile Private Tutor Finders Application
Server
Start
Select tutor's profile column on list
Send HTTP request to server with parameters
Retrieve tutor's information from 'users', 'tutorInfo' and 'rateReview' collection in Firestore
Display tutor's profile
View displayed profile
[Wish to discuss with tutor]
[Wish to have demo class]
[Wish to have formal class]
Raise demo class request with data
Chat with tutor
Send HTTP request to server with data
Create 'demoClassRgst' document in Firestore
Raise formal class request with data
Initiate chat activity
Send HTTP request to server with data
Create 'formalClassRgst' document in Firestore
Display registration status
End

Figure 5.10: Activity Diagram – View Tutor’s Profile

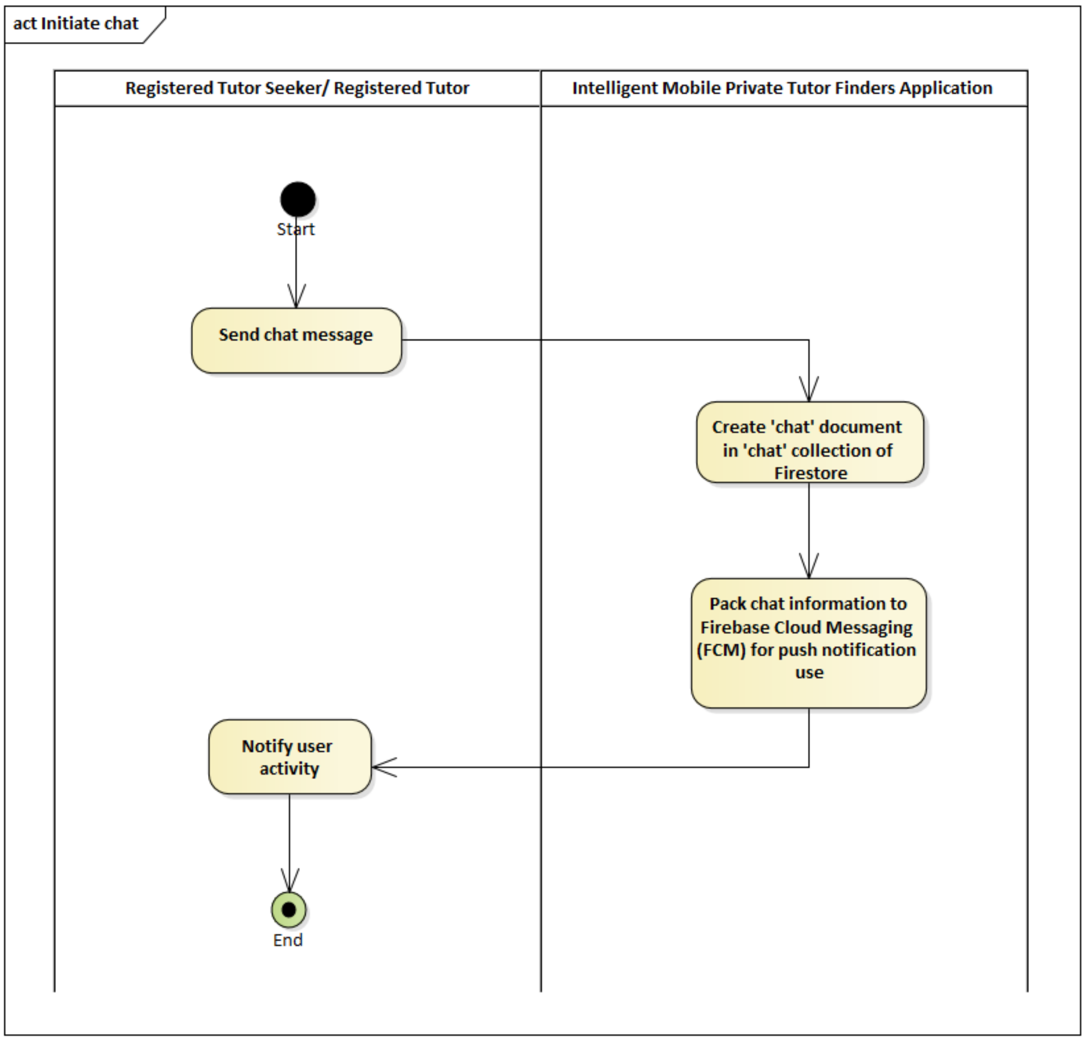

act Initiate chat
Registered Tutor Seeker/ Registered Tutor
Intelligent Mobile Private Tutor Finders Application
Start
Send chat message
Create 'chat' document in 'chat' collection of Firestore
Pack chat information to Firebase Cloud Messaging (FCM) for push notification use
Notify user activity
End

Figure 5.11: Activity Diagram – Initiate Chat

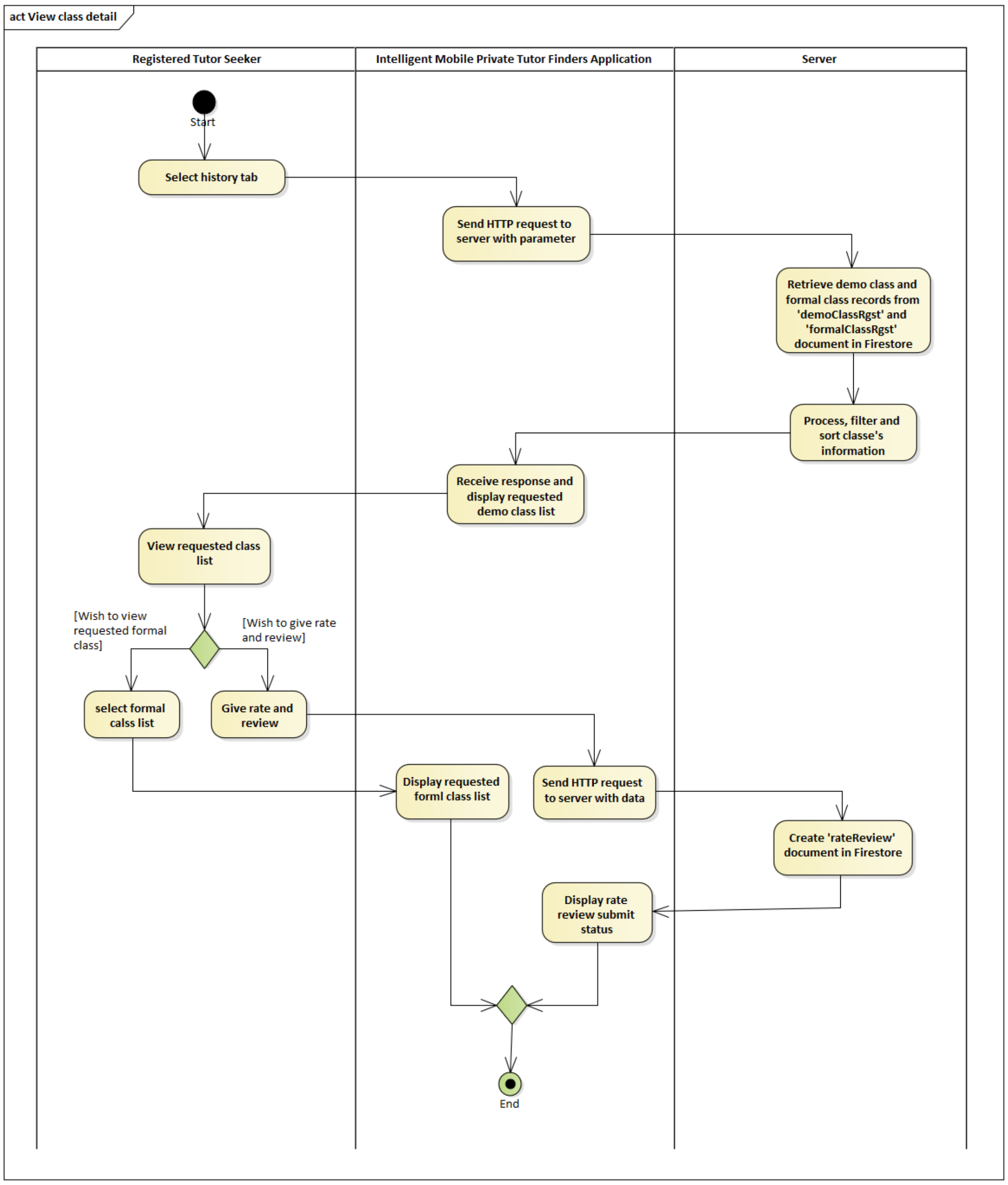

act View class detail
Registered Tutor Seeker
Intelligent Mobile Private Tutor Finders Application
Server
Start
Select history tab
Send HTTP request to server with parameter
Retrieve demo class and formal class records from 'demoClassRgst' and 'formalClassRgst' document in Firestore
Process, filter and sort classe's information
Receive response and display requested demo class list
View requested class list
[Wish to view requested formal class]
[Wish to give rate and review]
select formal calss list
Give rate and review
Display requested forml class list
Send HTTP request to server with data
Create 'rateReview' document in Firestore
Display rate review submit status
End

Figure 5.12: Activity Diagram – View Class Detail

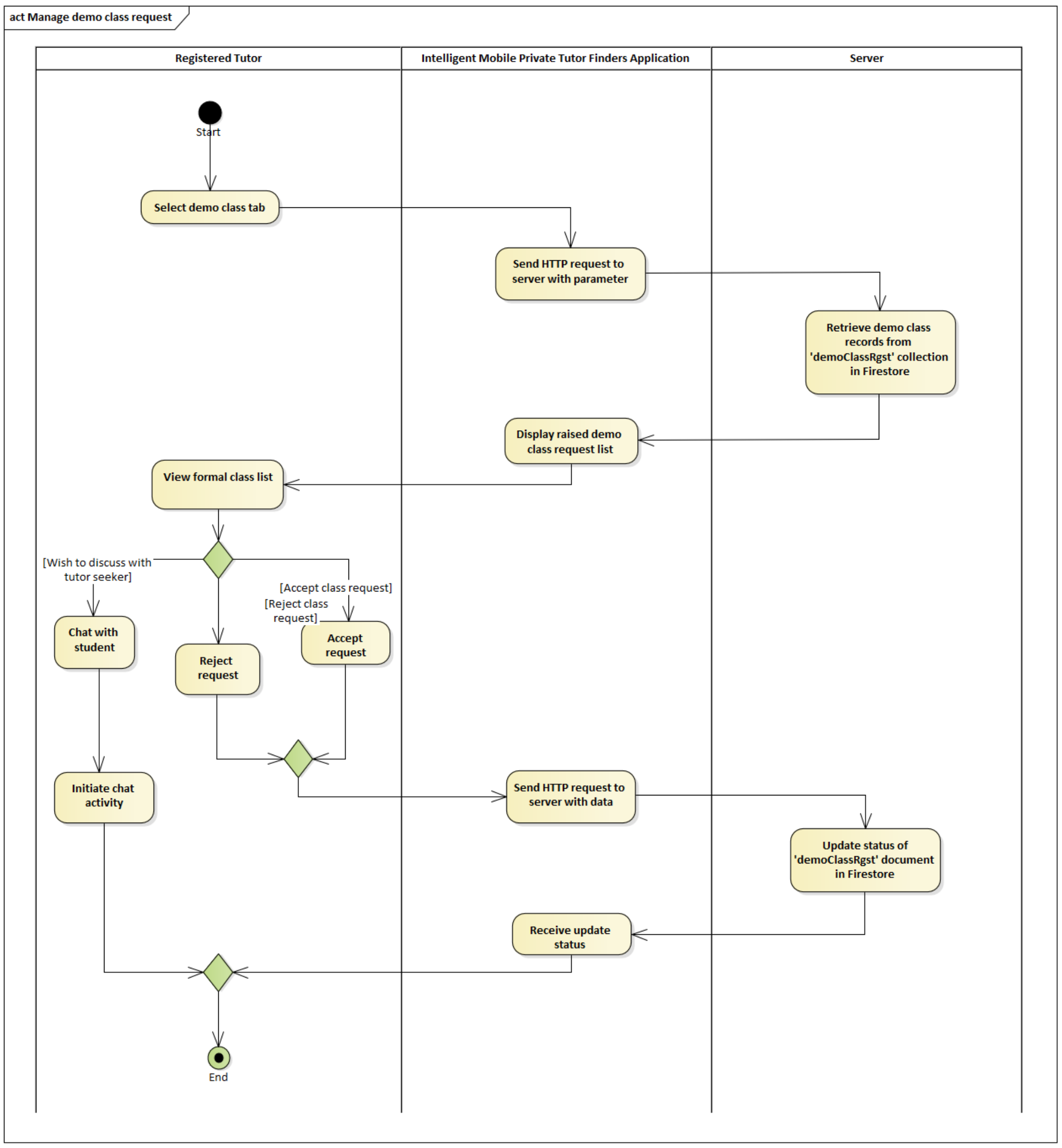

act Manage demo class request
Registered Tutor
Intelligent Mobile Private Tutor Finders Application
Server
Start
Select demo class tab
Send HTTP request to server with parameter
Retrieve demo class records from 'demoClassRgst' collection in Firestore
Display raised demo class request list
View formal class list
[Wish to discuss with tutor seeker]
[Accept class request]
[Reject class request]
Chat with student
Reject request
Accept request
Initiate chat activity
Send HTTP request to server with data
Update status of 'demoClassRgst' document in Firestore
Receive update status
End

Figure 5.13: Activity Diagram – Manage Demo Class Request

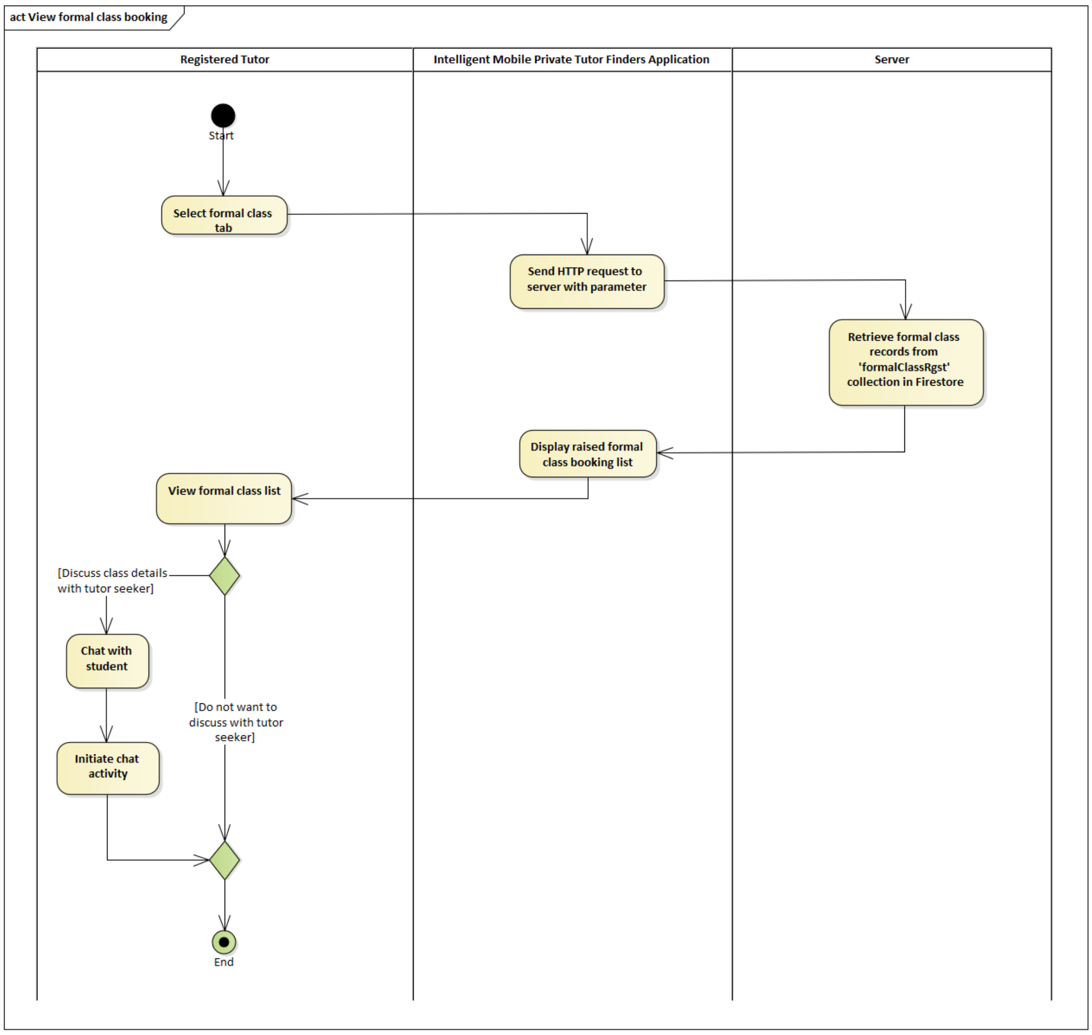

act View formal class booking
Registered Tutor
Intelligent Mobile Private Tutor Finders Application
Server
Start
Select formal class tab
Send HTTP request to server with parameter
Retrieve formal class records from 'formalClassRgst' collection in Firestore
Display raised formal class booking list
View formal class list
[Discuss class details with tutor seeker]
Chat with student
[Do not want to discuss with tutor seeker]
Initiate chat activity
End

Figure 5.14: Activity Diagram – View Formal Class Booking

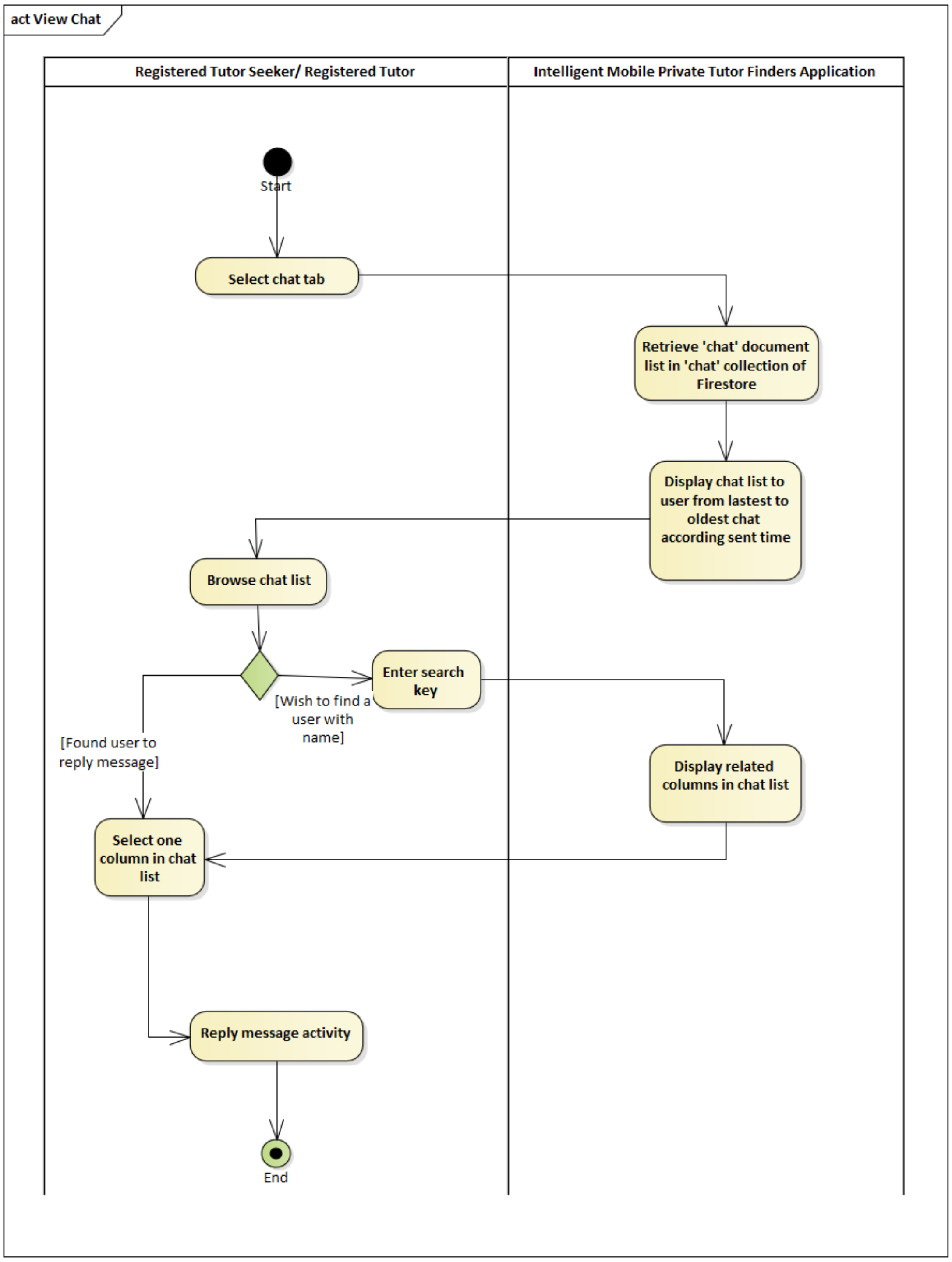

act View Chat
Registered Tutor Seeker/ Registered Tutor
Intelligent Mobile Private Tutor Finders Application
Start
Select chat tab
Retrieve 'chat' document list in 'chat' collection of Firestore
Display chat list to user from lastest to oldest chat according sent time
Browse chat list
Enter search key
[Wish to find a user with name]
[Found user to reply message]
Display related columns in chat list
Select one column in chat list
Reply message activity
End

Figure 5.15: Activity Diagram – View Chat

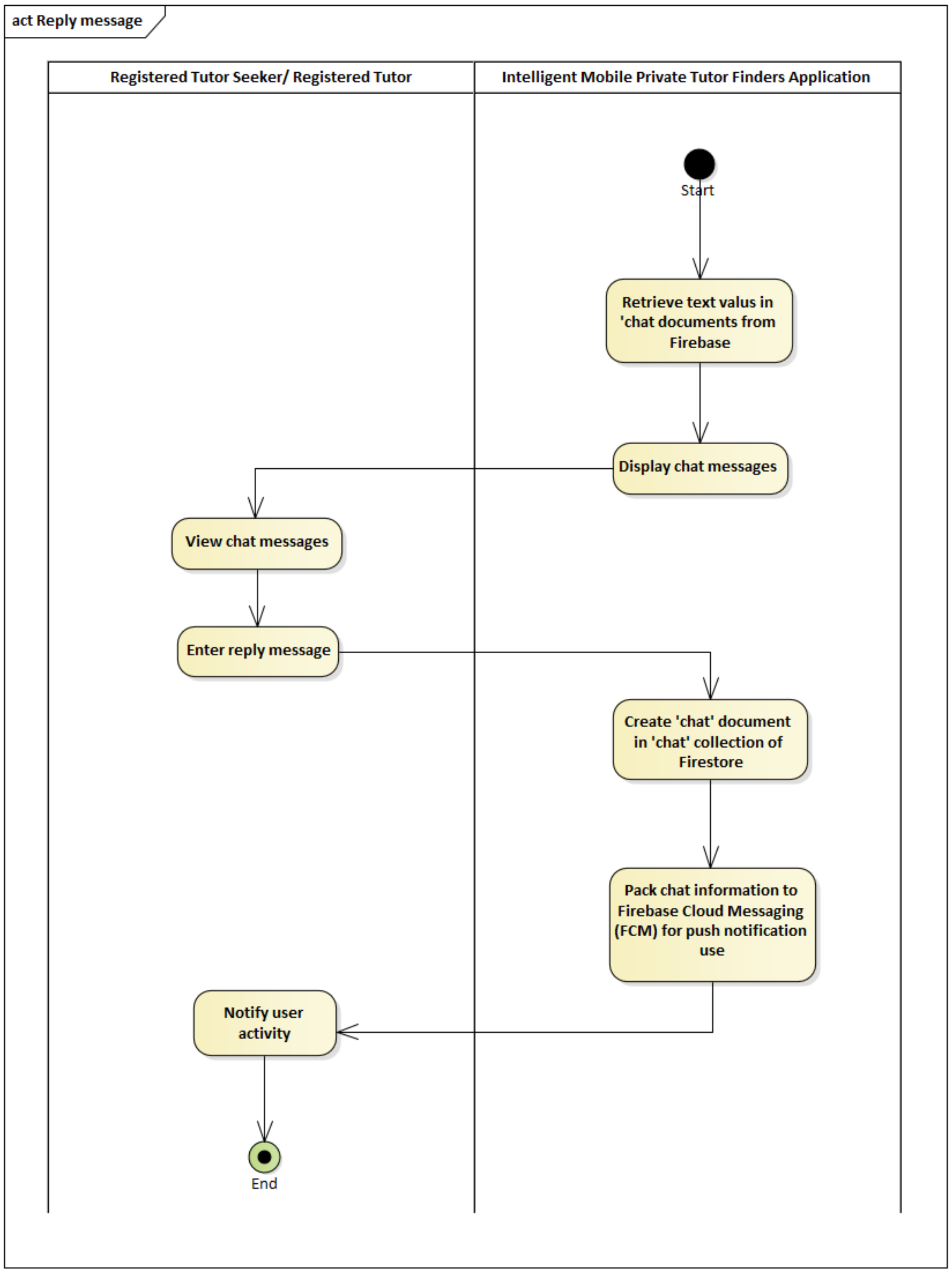

act Reply message
Registered Tutor Seeker/ Registered Tutor
Intelligent Mobile Private Tutor Finders Application
Start
Retrieve text valus in 'chat documents from Firebase
Display chat messages
View chat messages
Enter reply message
Create 'chat' document in 'chat' collection of Firestore
Pack chat information to Firebase Cloud Messaging (FCM) for push notification use
Notify user activity
End

Figure 5.16: Activity Diagram – Reply Message

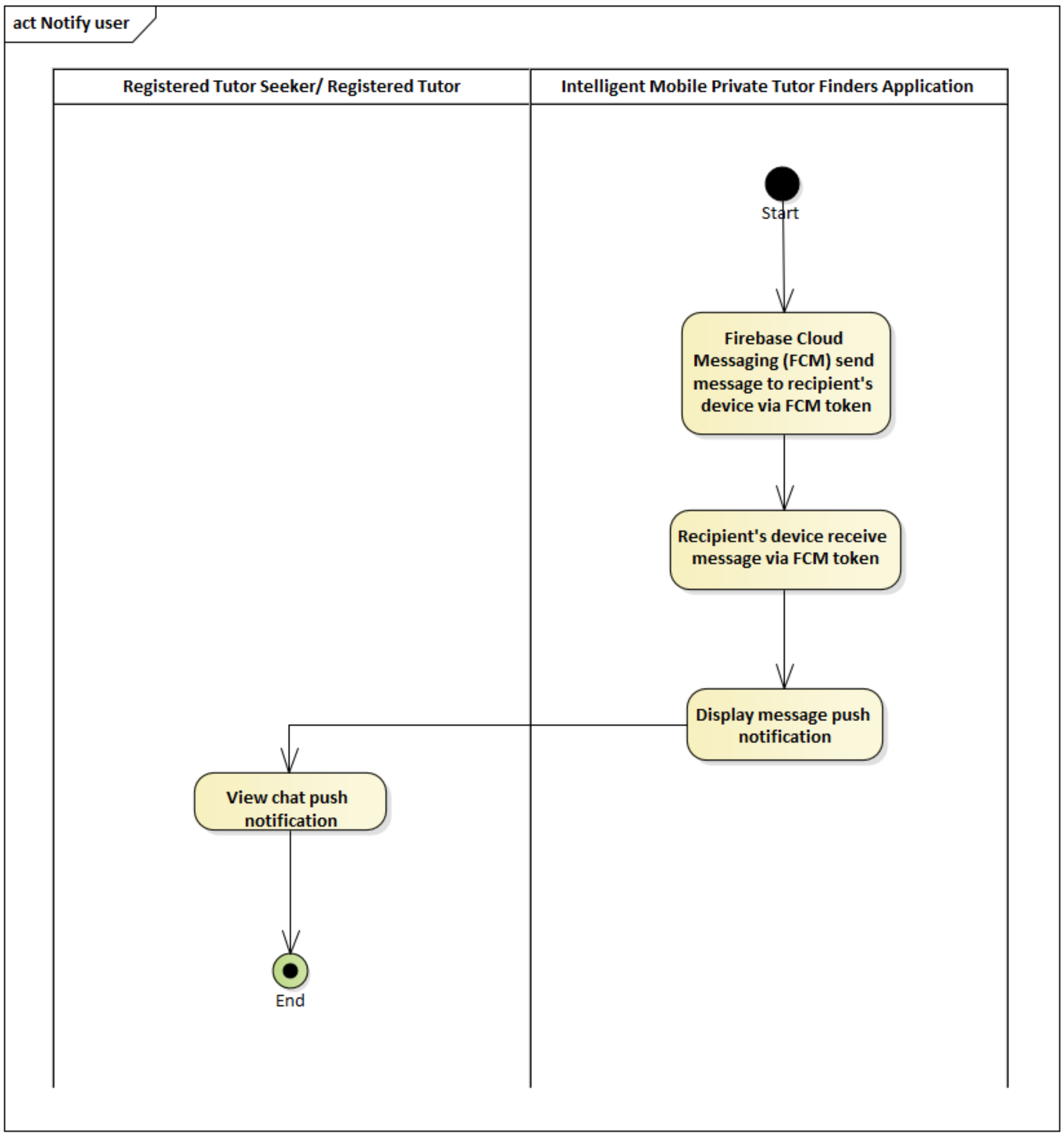

act Notify user
Registered Tutor Seeker/ Registered Tutor
Intelligent Mobile Private Tutor Finders Application
Start
Firebase Cloud Messaging (FCM) send message to recipient's device via FCM token
Recipient's device receive message via FCM token
Display message push notification
View chat push notification
End

Figure 5.17: Activity Diagram – Notify User

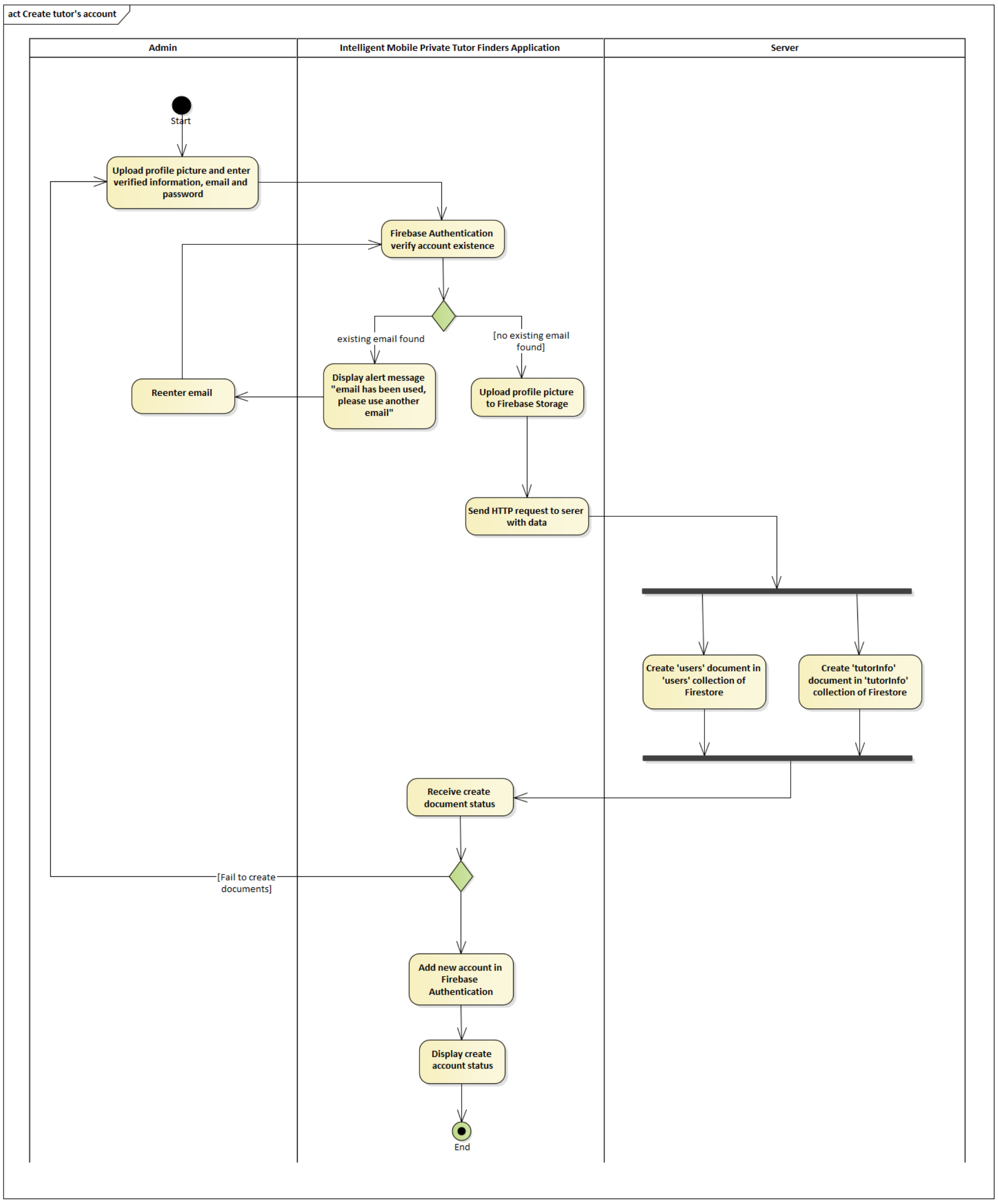

act Create tutor's account
Admin
Intelligent Mobile Private Tutor Finders Application
Server
Start
Upload profile picture and enter verified information, email and password
Firebase Authentication verify account existence
existing email found
[no existing email found]
Display alert message "email has been used, please use another email"
Reenter email
Upload profile picture to Firebase Storage
Send HTTP request to serer with data
Create 'users' document in 'users' collection of Firestore
Create 'tutorInfo' document in 'tutorInfo' collection of Firestore
Receive create document status
[Fail to create documents]
Add new account in Firebase Authentication
Display create account status
End

Figure 5.18: Activity Diagram – Create Tutor’s Account

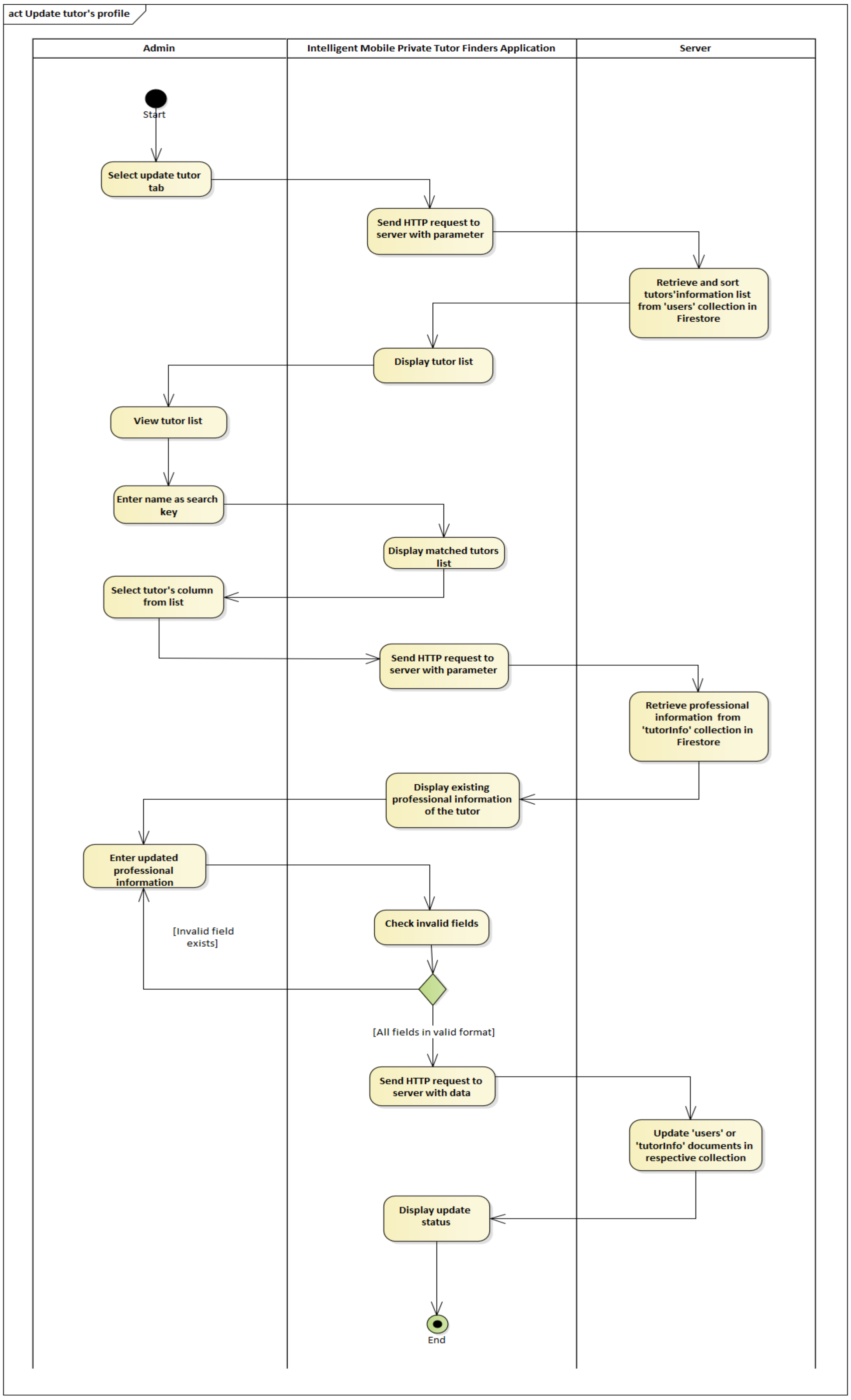

act Update tutor's profile
Admin
Intelligent Mobile Private Tutor Finders Application
Server
Start
Select update tutor tab
Send HTTP request to server with parameter
Retrieve and sort tutors'information list from 'users' collection in Firestore
Display tutor list
View tutor list
Enter name as search key
Display matched tutors list
Select tutor's column from list
Send HTTP request to server with parameter
Retrieve professional information from 'tutorInfo' collection in Firestore
Display existing professional information of the tutor
Enter updated professional information
Check invalid fields
[Invalid field exists]
[All fields in valid format]
Send HTTP request to server with data
Update 'users' or 'tutorInfo' documents in respective collection
Display update status
End

Figure 5.19: Activity Diagram – Update Tutor’s Profile

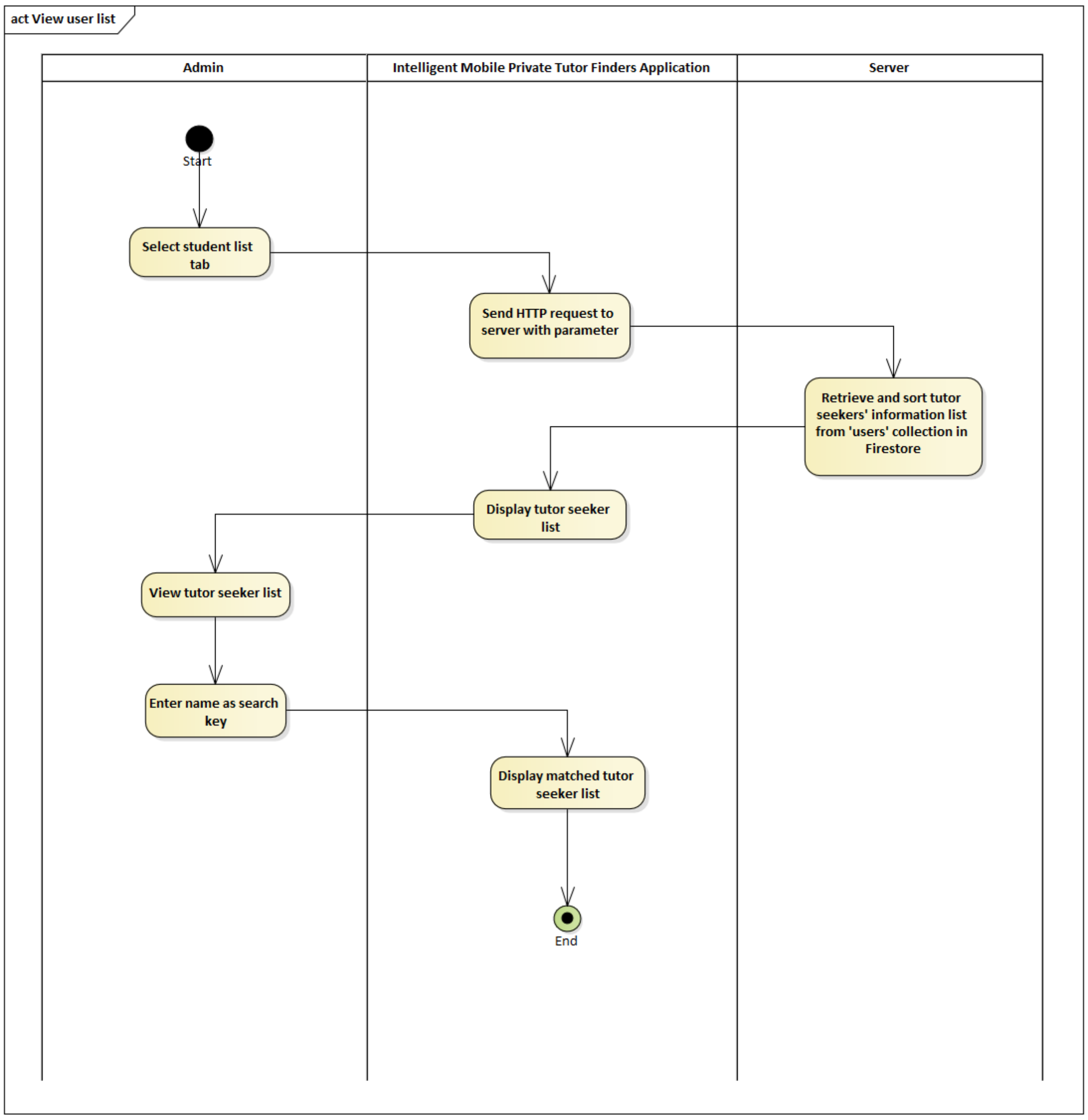

act View user list
Admin
Intelligent Mobile Private Tutor Finders Application
Server
Start
Select student list tab
Send HTTP request to server with parameter
Retrieve and sort tutor seekers' information list from 'users' collection in Firestore
Display tutor seeker list
View tutor seeker list
Enter name as search key
Display matched tutor seeker list
End

Figure 5.20: Activity Diagram – View User List

5.3 Data Model Design

Firestore is used as the cloud database in this application which applied NoSQL to store the application's data. There are 6 collections and each collection consists of many documents which are the data records in this application. The data model design's explanation is shown in the logical data model (LDM), physical data model (PDM), data schema, and data dictionary.

5.3.1 Logical Data Model (LDM)

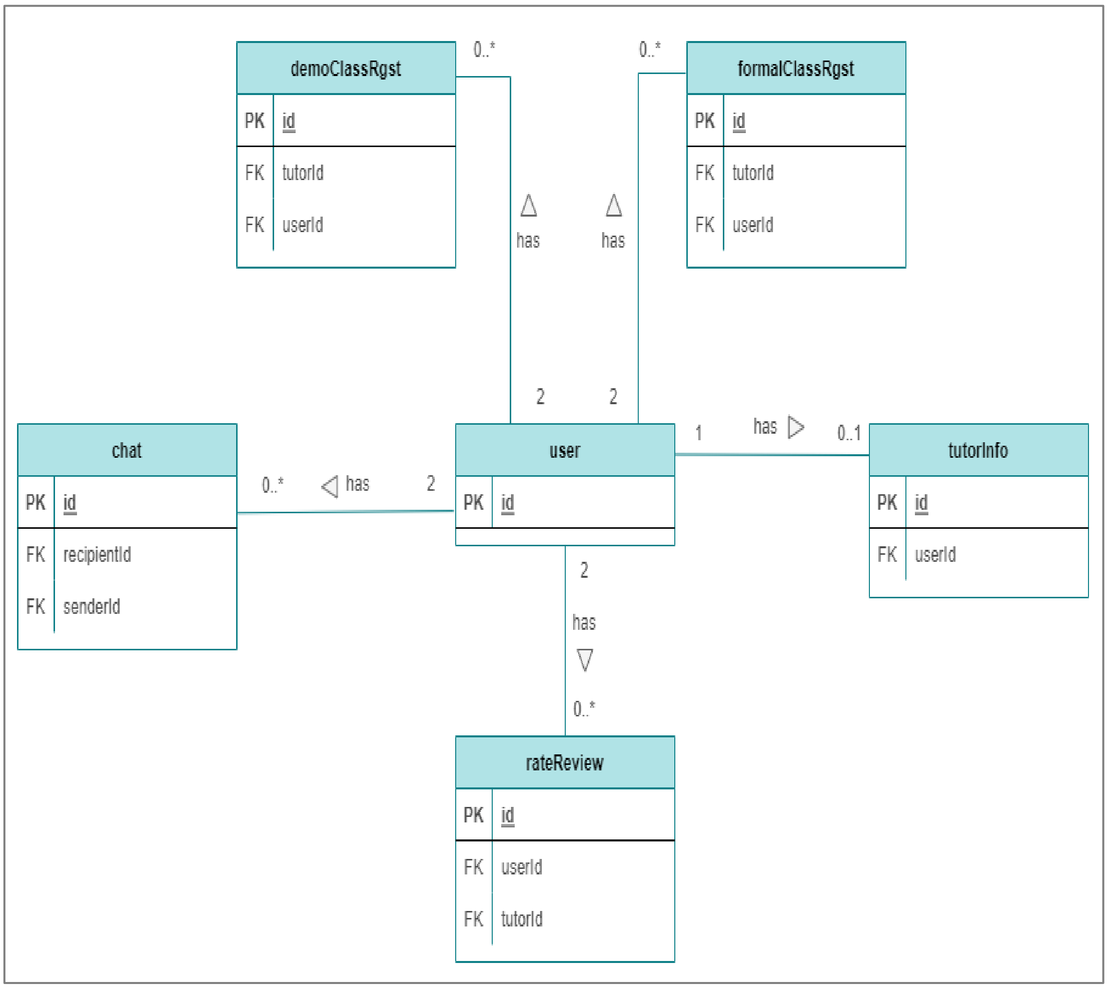


Figure 5.21: Logical Data Model Diagram

5.3.2 Physical Data Model (PDM)

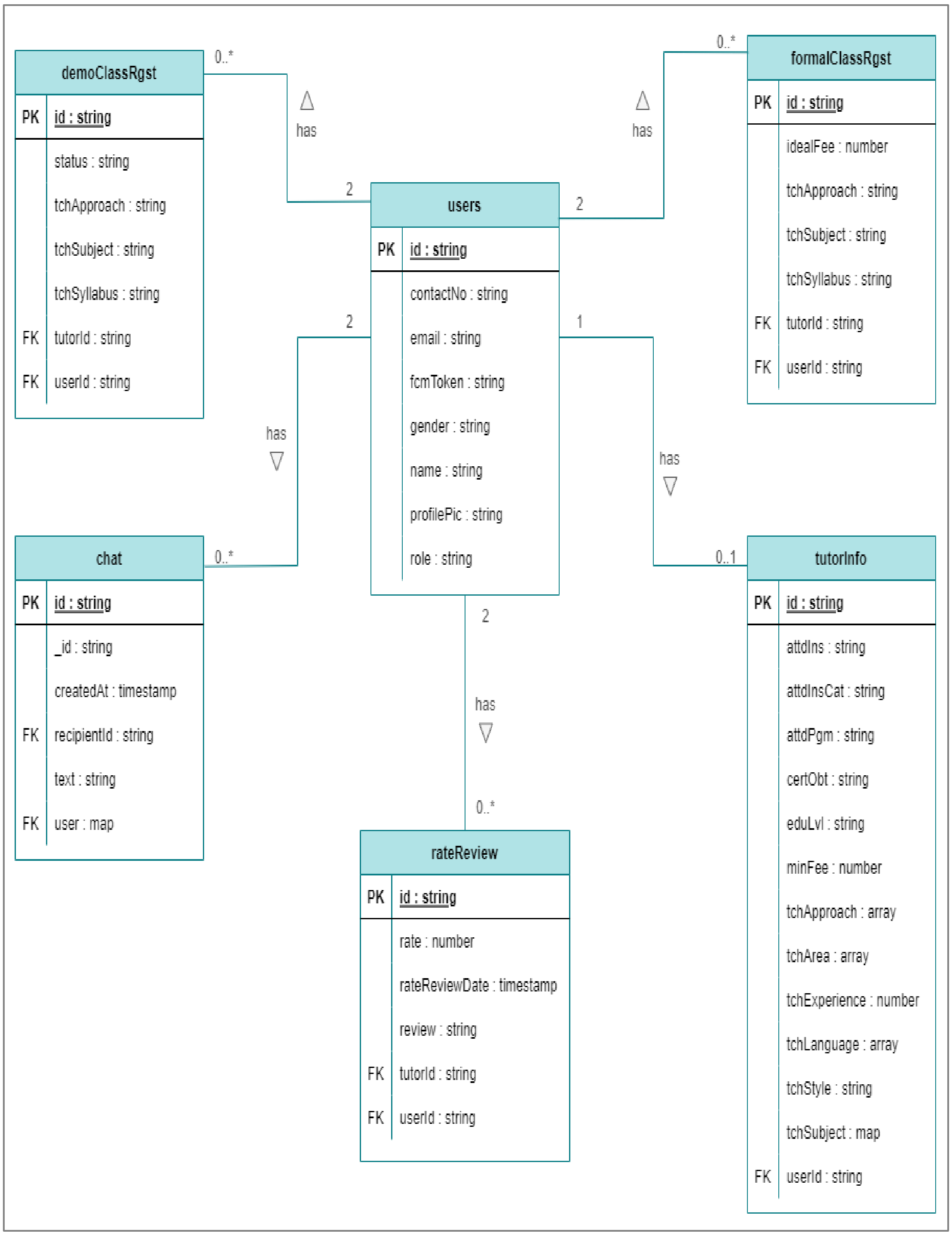
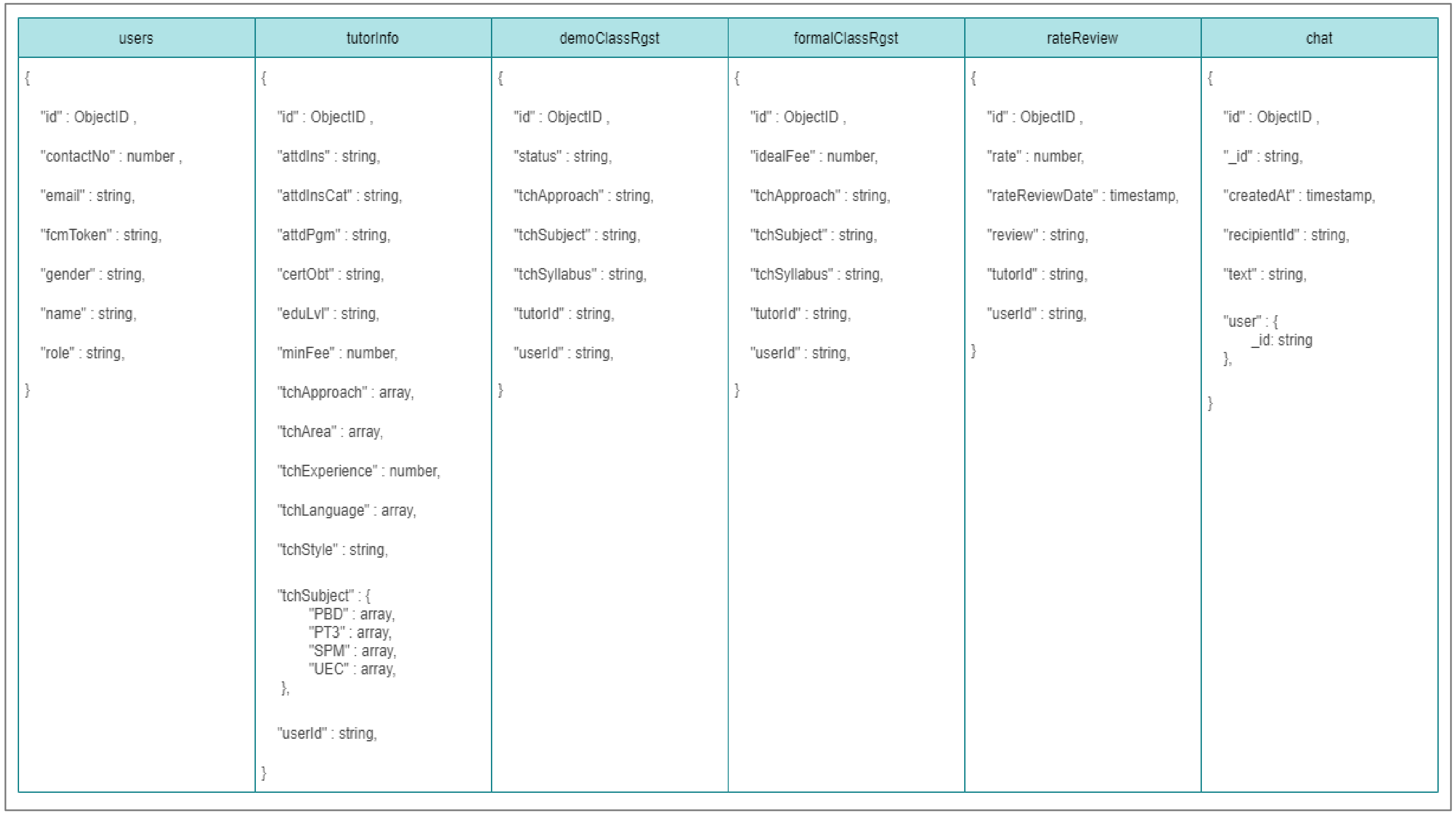


Figure 5.22: Physical Data Model Diagram



users
```
{
    "id" : ObjectID ,
    "contactNo" : number ,
    "email" : string,
    "fcmToken" : string,
    "gender" : string,
    "name" : string,
    "role" : string,
}
```

tutorInfo
```
{
    "id" : ObjectID ,
    "attdIns" : string,
    "attdInsCat" : string,
    "attdPgm" : string,
    "certObt" : string,
    "eduLvl" : string,
    "minFee" : number,
    "tchApproach" : array,
    "tchArea" : array,
    "tchExperience" : number,
    "tchLanguage" : array,
    "tchStyle" : string,
    "tchSubject" : {
          "PBD" : array,
          "PT3" : array,
          "SPM" : array,
          "UEC" : array,
     },
    "userId" : string,
}
```

demoClassRgst
```
{
    "id" : ObjectID ,
    "status" : string,
    "tchApproach" : string,
    "tchSubject" : string,
    "tchSyllabus" : string,
    "tutorId" : string,
    "userId" : string,
}
```

formalClassRgst
```
{
    "id" : ObjectID ,
    "idealFee" : number,
    "tchApproach" : string,
    "tchSubject" : string,
    "tchSyllabus" : string,
    "tutorId" : string,
    "userId" : string,
}
```

rateReview
```
{
    "id" : ObjectID ,
    "rate" : number,
    "rateReviewDate" : timestamp,
    "review" : string,
    "tutorId" : string,
    "userId" : string,
}
```

chat
```
{
    "id" : ObjectID ,
    "_id" : string,
    "createdAt" : timestamp,
    "recipientId" : string,
    "text" : string,
    "user" : {
         _id: string
    },
}
```

Figure 5.23: Data Schema

5.3.3 Data Dictionary

Table 5.1: Data Dictionary (Users Collection)

Field Name	**Data Type**	**Description**	**Key (PK/ FK)**	**FK Reference Collection**	**Null able**
id	string	Document id which is unique for each user	PK	-	No
contactNo	string	Contact number of user	-	-	No
email	string	Email address of user	-	-	No
fcmToken	string	Firebase Cloud Messaging Token of user (unique in each device's application)	-	-	No
gender	string	Gender of user	-	-	No
name	string	Name of user	-	-	No
role	string	Application role of user	-	-	No

Table 5.2: Data Dictionary (Tutor Info Collection)

Field Name	**Data Type**	**Description**	**Key (PK/ FK)**	**FK Refere nce Collecti on**	**Nullable**
id	string	Document id which is unique for each tutor information	PK	-	No
attdIns	string	Attended institution of tutor	-	-	No
attdInsCat	string	Attended institution category of tutor (local or oversea)	-	-	No
attdPgm	string	Attended program of tutor in his/her attended institution	-	-	No
certObt	string	Professional education certificate obtained of tutor	-	-	Yes
eduLvl	string	Highest education achievement of tutor	-	-	No

minFee	number	Minimum tuition fee per hour of the tutor (Malaysia Ringgit)	-	-	No
tchApproach	Array of string	Teaching approaches of tutor	-	-	No
tchArea	Array of string	Teaching areas of tutor	-	-	No
tchExperience	number	Teaching experience of tutor (in years)	-	-	No
tchLanguage	Array of string	Teaching languages of tutor	-	-	No
tchStyle	string	Teaching style of tutor	-	-	No
tchSubject	map of array	Teaching subjects of tutor according to syllabus: - "PBD": array of subjects in string - "PT3": array of subjects in string - "SPM": array of subjects in string - "UEC": array of subjects in string	-	-	No (If no string in the key, the key will not exist. E.g., tutor only teach SPM only have SPM key in the map)
userId	string	User's collection's document id which is respective to the tutor information	FK	users	No

Table 5.3: Data Dictionary (Demo Class Registration Collection)

Field Name	**Data Type**	**Description**	**Key (PK/ FK)**	**FK Reference Collection**	**Null able**
id	string	Document id which is unique for each demo class registration	PK	-	No
status	string	Status of demo class registration	-	-	No
tchApproac h	string	Teaching approach of demo class that is selected by tutor seeker	-	-	No

tchSubject	string	Teaching subject of demo class that is selected by tutor seeker	-	-	No
tchSyllabus	string	Teaching syllabus of demo class that is selected by tutor seeker	-	-	No
tutorId	string	User’s collection’s document id of the tutor	FK	users	No
userId	string	User’s collection’s document id of the tutor seeker	FK	users	No

Table 5.4: Data Dictionary (Formal Class Registration Collection)

Field Name	**Data Type**	**Description**	**Key (PK/ FK)**	**FK Reference Collection**	**Null able**
id	string	Document id which is unique for each demo class registration	PK	-	No
IdealFee	number	Ideal tuition fee of tutor seeker to pay to the tutor for per hour class (in Malaysia Ringgit)	-	-	No
tchApproac h	string	Teaching approach of demo class that is selected by tutor seeker	-	-	No
tchSubject	string	Teaching subject of demo class that is selected by tutor seeker	-	-	No
tchSyllabus	string	Teaching syllabus of demo class that is selected by tutor seeker	-	-	No
tutorId	string	User’s collection’s document id of the tutor	FK	users	No
userId	string	User’s collection’s document id of the tutor seeker	FK	users	No

Table 5.5: Data Dictionary (Rate Review Collection)

Field Name	**Data Type**	**Description**	**Key (PK/ FK)**	**FK Reference Collection**	**Null able**
id	string	Document id which is unique for each demo class registration	PK	-	No

rate	number	Rate that is given by the tutor seeker to the tutor (from 1 to 5)	-	-	No
rateReview Date	timestamp	Create time of the rate review document	-	-	No
review	string	Review that is given by the tutor seeker to tutor	-	-	Yes
tutorId	string	User’s collection’s document id of the tutor	FK	users	No
userId	string	User’s collection’s document id of the tutor seeker	FK	users	No

Table 5.6: Data Dictionary (Chat Collection)

Field Name	**Data Type**	**Description**	**Key (PK /FK)**	**FK Reference Collection**	**Null able**
id	string	Document id which is unique for each chat	PK	-	No
_id	string	A unique id that is used in react-native-gifted-chat to identify each chat message	-	-	No
createdAt	timestamp	Create time for the chat message	-	-	No
recipientId	string	User’s collection’s document id of the receiving user	-	-	Yes
text	string	Chat text in the message	-	-	No
user	Map of string	Map to store sender’s information: - “_id”: user’s id string	FK	users	No

5.4 User Interface Design

5.4.1 Navigation Model

The navigation diagrams show the navigation between screens and stacks. The switch navigator in App.js is the main controller to handle the navigation in the application. The student interface, tutor interface, and admin interface indicate tutor seeker user interface, tutor user interface, and admin interface respectively. There are separated into different tab that includes different stacks that consist of the different screens for user interface displays. The initial stack indicates the first accessing stack while the initial screen is the first display screen when accessing the stack. The bidirectional arrow indicates the navigation between stack and screen is allowed (as an accessing gate) in this application.

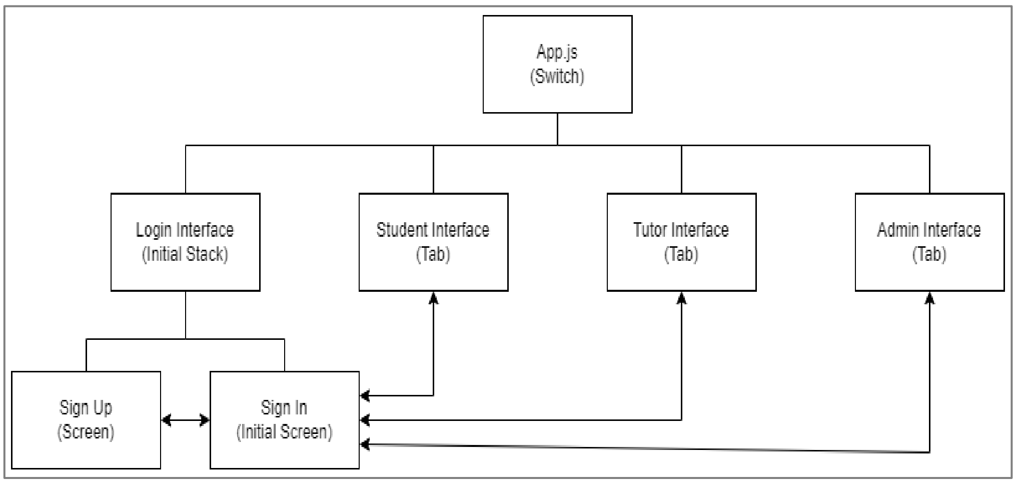


Figure 5.24: Navigation Model Diagram (Overview)

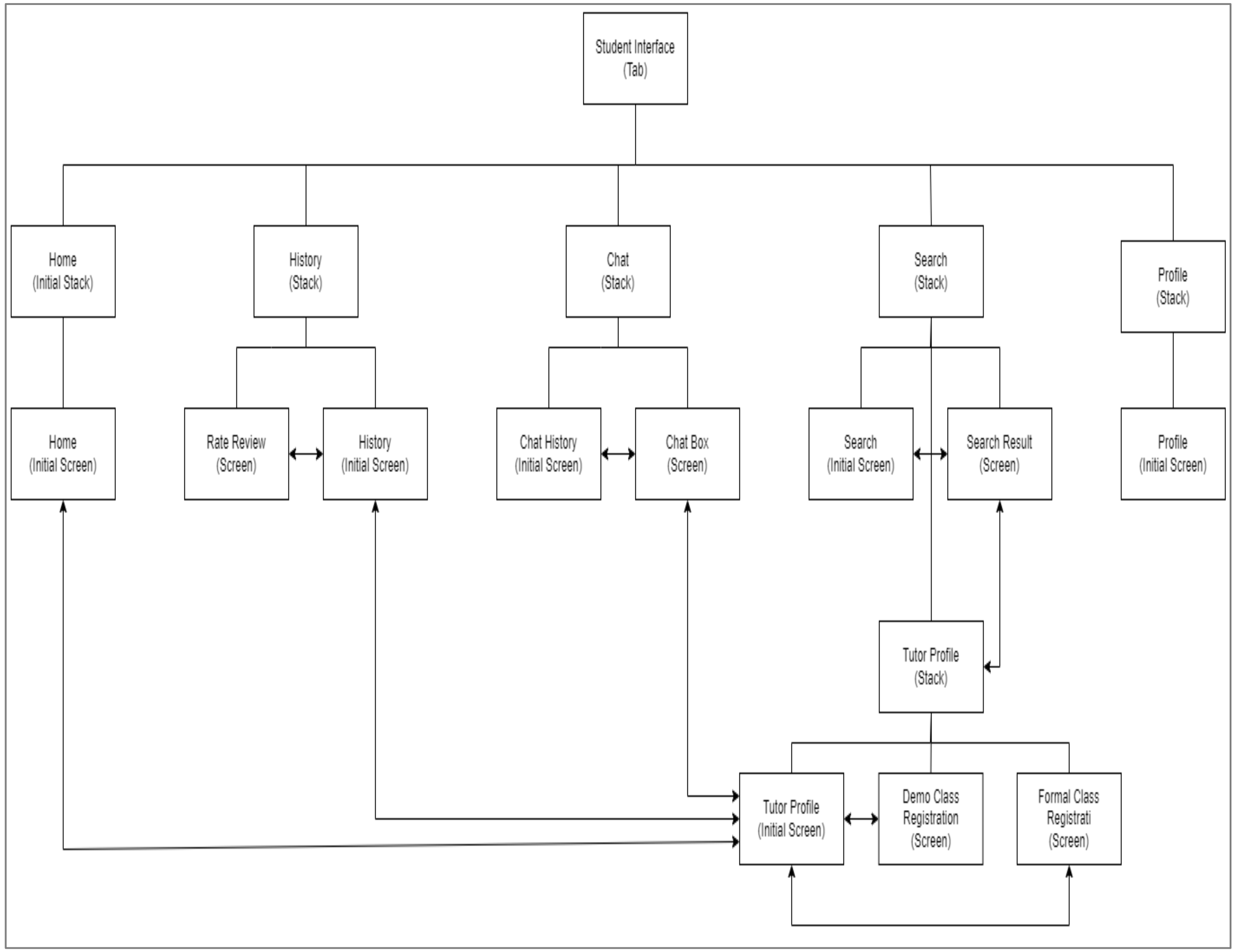


Figure 5.25: Navigation Model Diagram (Student Interface)

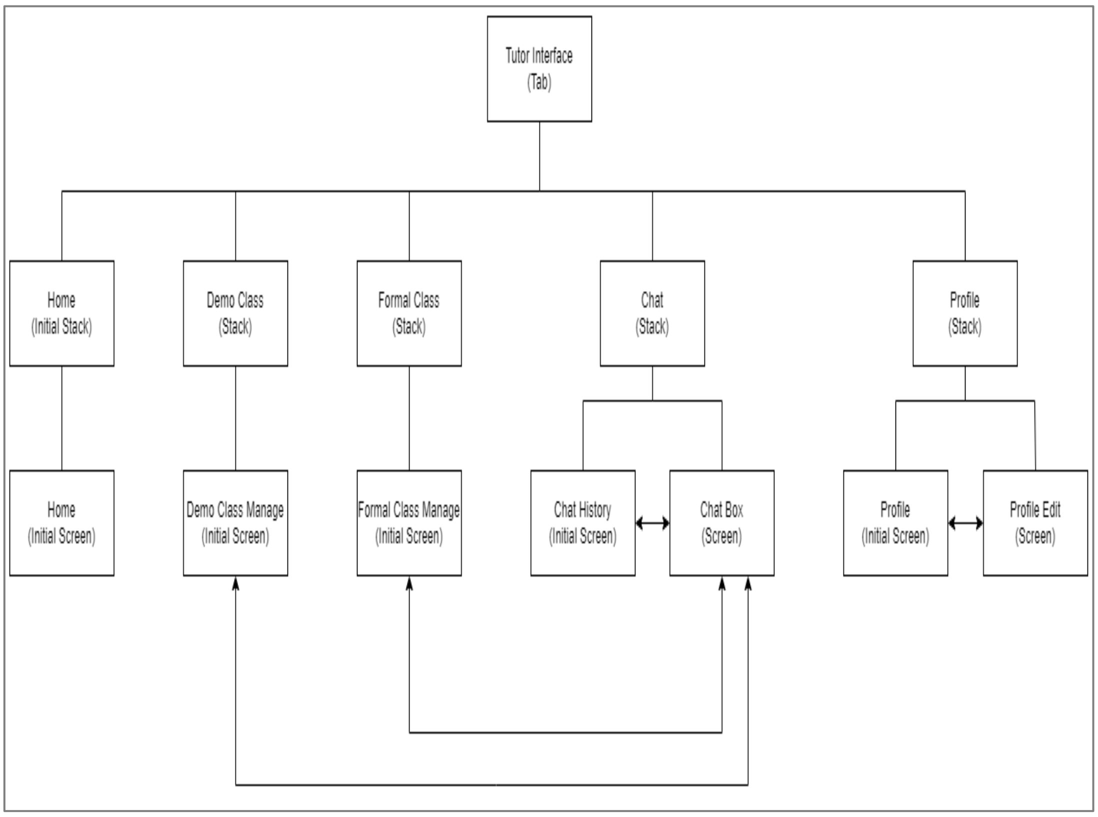


Figure 5.26: Navigation Model Diagram (Tutor Interface)

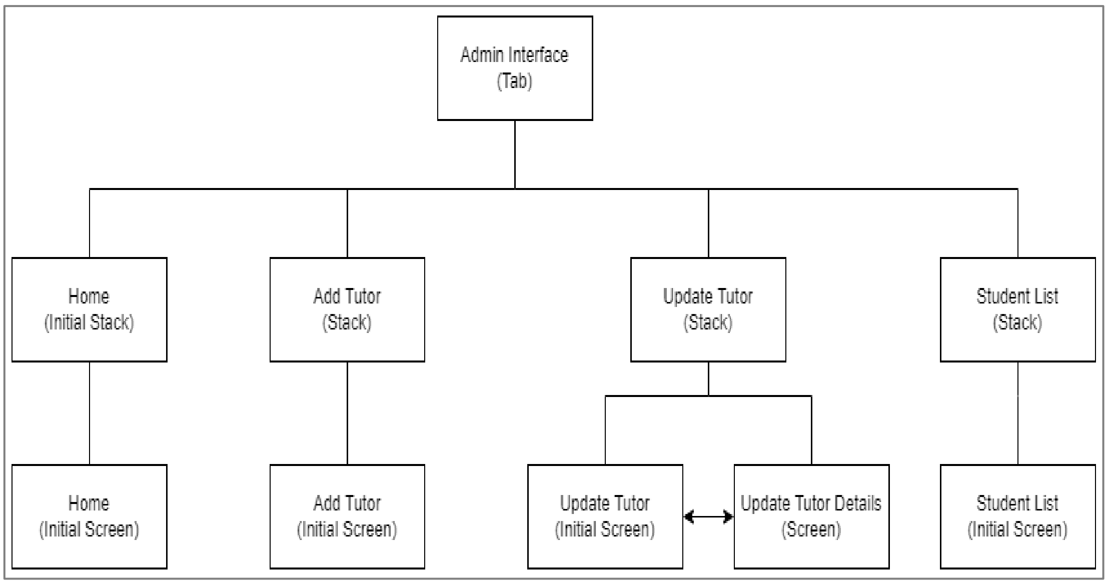
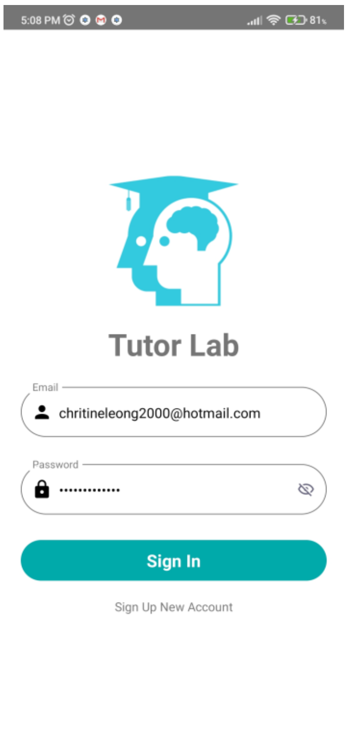


Figure 5.27: Navigation Model Diagram (Admin Interface)

5.4.2 User Interface Display

5.4.2.1 Login Interface

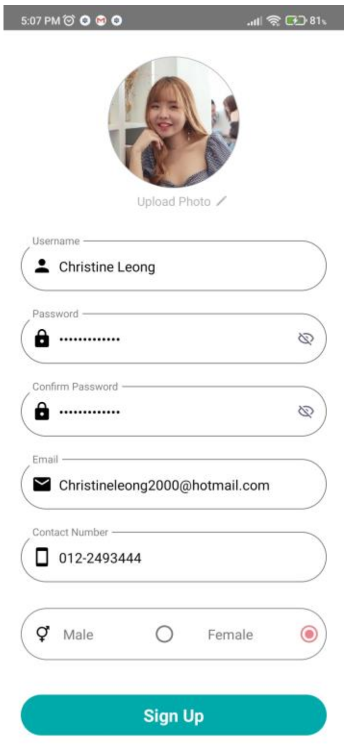

5:08 PM

Tutor Lab

Email: chritineleong2000@hotmail.com

Password: ••••••••••••••

Figure 5.28: UI – Sign in


5:07 PM

Username: Christine Leong

Password: ••••••••••••••

Confirm Password: ••••••••••••••

Email: Christineleong2000@hotmail.com

Contact Number: 012-2493444

Male Female [✓]

Figure 5.29: UI – Sign up

5.4.2.2 Student Interface

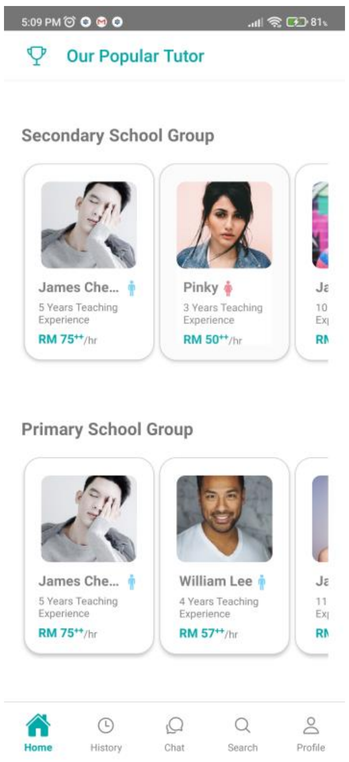


Figure 5.30: UI – Tutor Seeker Home

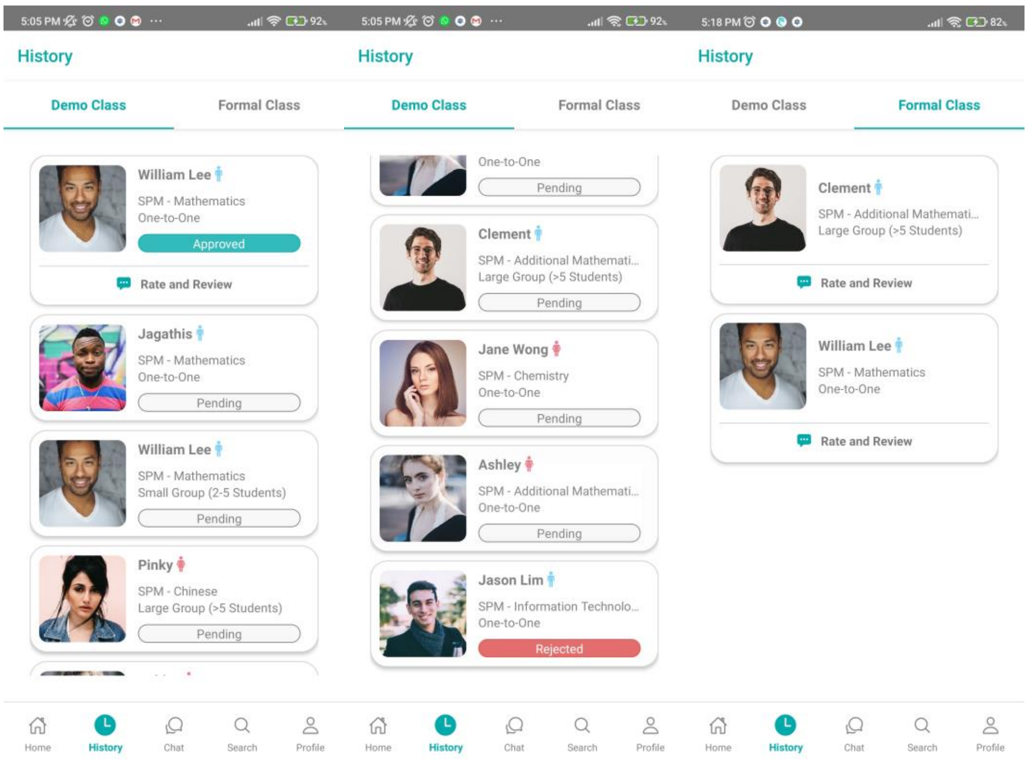
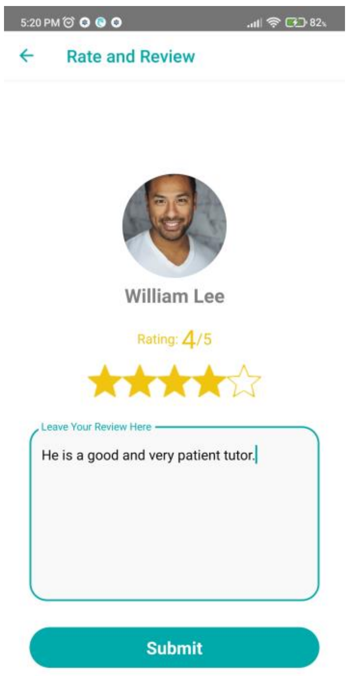


Figure 5.31: UI – Tutor Seeker Class Requests History



He is a good and very patient tutor.

Figure 5.32: UI – Tutor Seeker Rate and Review

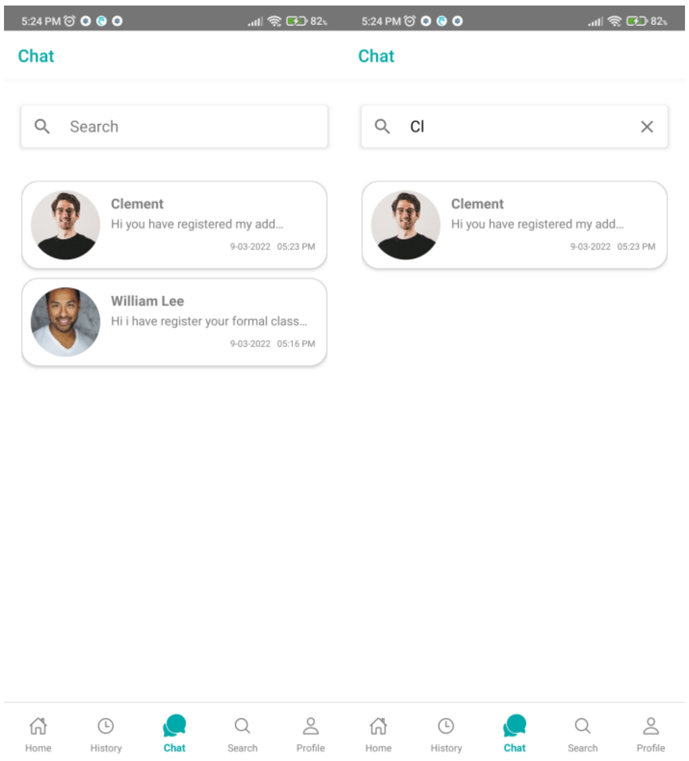
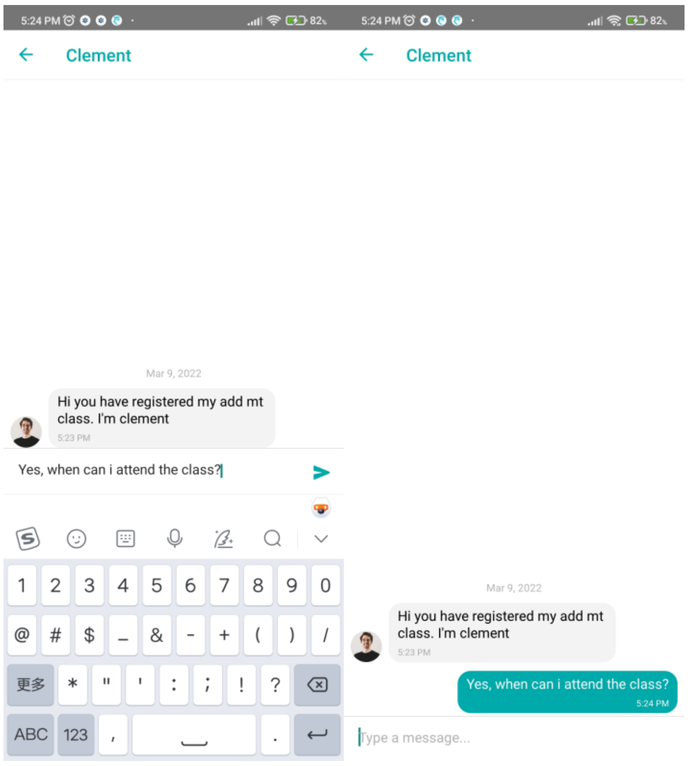


Figure 5.33: UI – Tutor Seeker Chat History



Clement

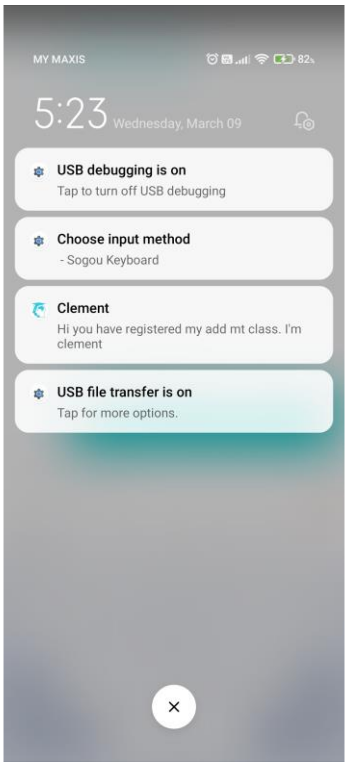
Mar 9, 2022

Hi you have registered my add mt class. I'm clement
5:23 PM

Yes, when can i attend the class?
5:24 PM

Type a message...

Figure 5.34: UI – Tutor Seeker Chat Box



MY MAXIS

5:23 Wednesday, March 09

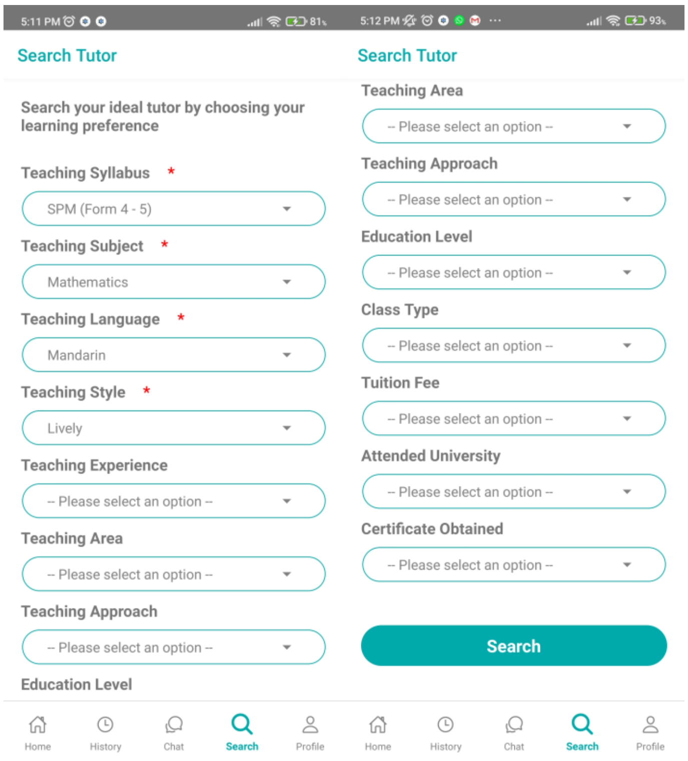
USB debugging is on
Tap to turn off USB debugging

Choose input method
- Sogou Keyboard

Clement
Hi you have registered my add mt class. I'm clement

USB file transfer is on
Tap for more options.

Figure 5.35: UI – Tutor Seeker Push Notification (From Tutor)

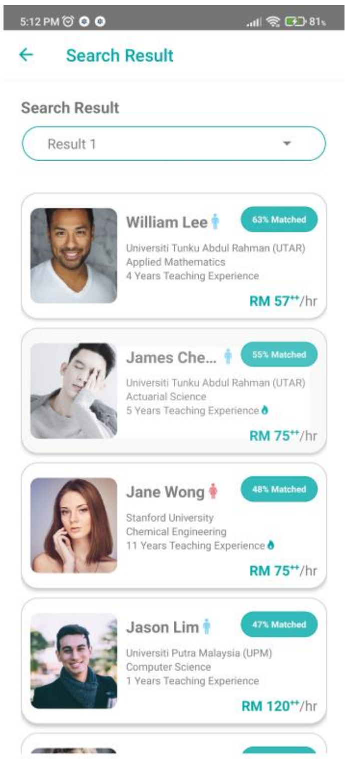


Search Tutor

Search your ideal tutor by choosing your learning preference

Teaching Syllabus *	SPM (Form 4 - 5)
Teaching Subject *	Mathematics
Teaching Language *	Mandarin
Teaching Style *	Lively
Teaching Experience	-- Please select an option --
Teaching Area	-- Please select an option --
Teaching Approach	-- Please select an option --
Education Level	-- Please select an option --
Class Type	-- Please select an option --
Tuition Fee	-- Please select an option --
Attended University	-- Please select an option --
Certificate Obtained	-- Please select an option --

Figure 5.36: UI – Tutor Seeker Search Tutor



Search Result

Search Result: Result 1

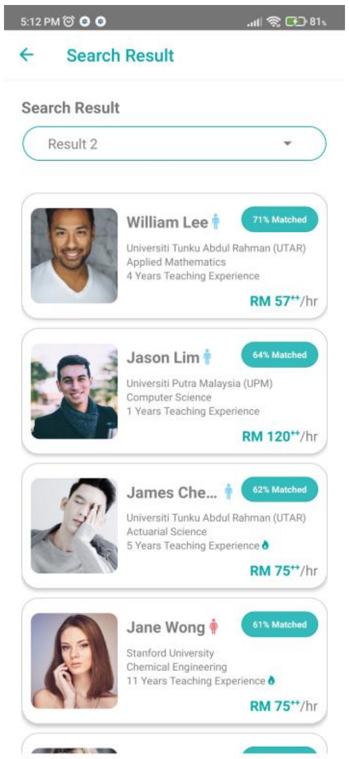
William Lee — 63% Matched
Universiti Tunku Abdul Rahman (UTAR)
Applied Mathematics
4 Years Teaching Experience
RM 57⁺⁺/hr

James Che... — 55% Matched
Universiti Tunku Abdul Rahman (UTAR)
Actuarial Science
5 Years Teaching Experience
RM 75⁺⁺/hr

Jane Wong — 48% Matched
Stanford University
Chemical Engineering
11 Years Teaching Experience
RM 75⁺⁺/hr

Jason Lim — 47% Matched
Universiti Putra Malaysia (UPM)
Computer Science
1 Years Teaching Experience
RM 120⁺⁺/hr

Figure 5.37: UI – Tutor Seeker Search Result (Manhattan Distance)



Search Result

Search Result

Result 2

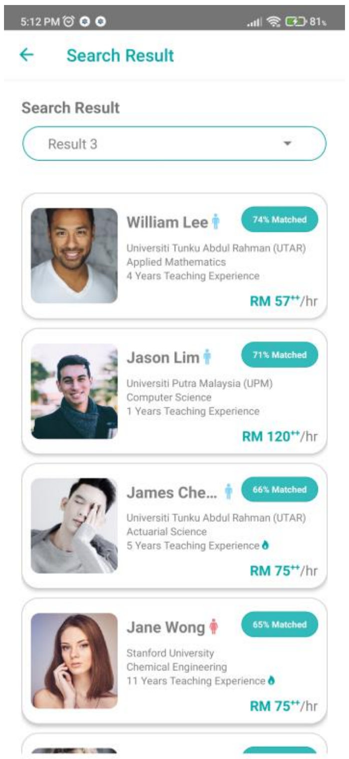
William Lee
71% Matched
Universiti Tunku Abdul Rahman (UTAR)
Applied Mathematics
4 Years Teaching Experience
RM 57^{++}/hr

Jason Lim
64% Matched
Universiti Putra Malaysia (UPM)
Computer Science
1 Years Teaching Experience
RM 120^{++}/hr

James Che...
62% Matched
Universiti Tunku Abdul Rahman (UTAR)
Actuarial Science
5 Years Teaching Experience
RM 75^{++}/hr

Jane Wong
61% Matched
Stanford University
Chemical Engineering
11 Years Teaching Experience
RM 75^{++}/hr

Figure 5.38: UI – Tutor Seeker Search Result (Euclidean Distance)



Search Result

Search Result

Result 3

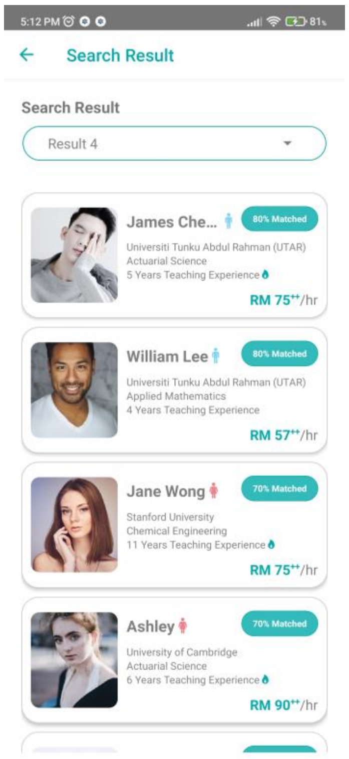
William Lee
74% Matched
Universiti Tunku Abdul Rahman (UTAR)
Applied Mathematics
4 Years Teaching Experience
RM 57^{++}/hr

Jason Lim
71% Matched
Universiti Putra Malaysia (UPM)
Computer Science
1 Years Teaching Experience
RM 120^{++}/hr

James Che...
66% Matched
Universiti Tunku Abdul Rahman (UTAR)
Actuarial Science
5 Years Teaching Experience
RM 75^{++}/hr

Jane Wong
65% Matched
Stanford University
Chemical Engineering
11 Years Teaching Experience
RM 75^{++}/hr

Figure 5.39: UI – Tutor Seeker Search Result (Minkowski Distance)



Search Result

Result 4

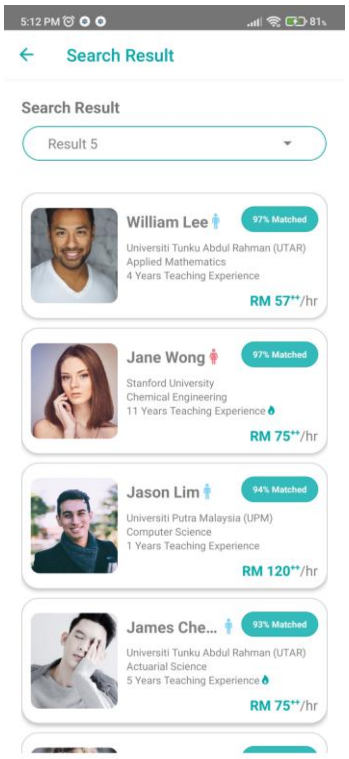
James Che... 80% Matched
Universiti Tunku Abdul Rahman (UTAR)
Actuarial Science
5 Years Teaching Experience
RM 75++/hr

William Lee 80% Matched
Universiti Tunku Abdul Rahman (UTAR)
Applied Mathematics
4 Years Teaching Experience
RM 57++/hr

Jane Wong 70% Matched
Stanford University
Chemical Engineering
11 Years Teaching Experience
RM 75++/hr

Ashley 70% Matched
University of Cambridge
Actuarial Science
6 Years Teaching Experience
RM 90++/hr

Figure 5.40: UI – Tutor Seeker Search Result (Jaccard Similarity Coefficient)



Search Result

Result 5

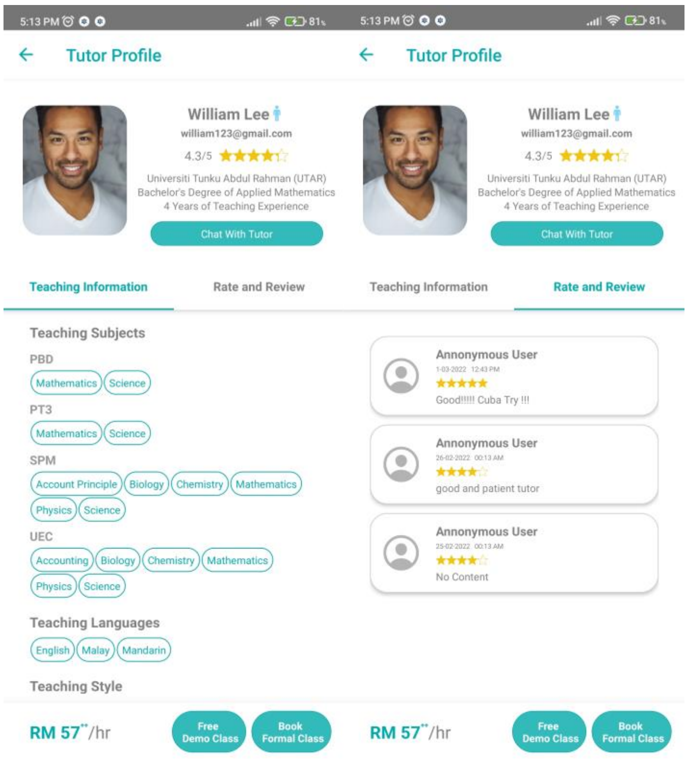
William Lee 97% Matched
Universiti Tunku Abdul Rahman (UTAR)
Applied Mathematics
4 Years Teaching Experience
RM 57++/hr

Jane Wong 97% Matched
Stanford University
Chemical Engineering
11 Years Teaching Experience
RM 75++/hr

Jason Lim 94% Matched
Universiti Putra Malaysia (UPM)
Computer Science
1 Years Teaching Experience
RM 120++/hr

James Che... 93% Matched
Universiti Tunku Abdul Rahman (UTAR)
Actuarial Science
5 Years Teaching Experience
RM 75++/hr

Figure 5.41: UI – Tutor Seeker Search Result (Cosine Similarity)


5:13 PM 81%

Tutor Profile

William Lee
william123@gmail.com
4.3/5
Universiti Tunku Abdul Rahman (UTAR)
Bachelor's Degree of Applied Mathematics
4 Years of Teaching Experience
Chat With Tutor

Teaching Information | Rate and Review

Teaching Subjects
PBD
Mathematics | Science
PT3
Mathematics | Science
SPM
Account Principle | Biology | Chemistry | Mathematics | Physics | Science
UEC
Accounting | Biology | Chemistry | Mathematics | Physics | Science

Teaching Languages
English | Malay | Mandarin

Teaching Style

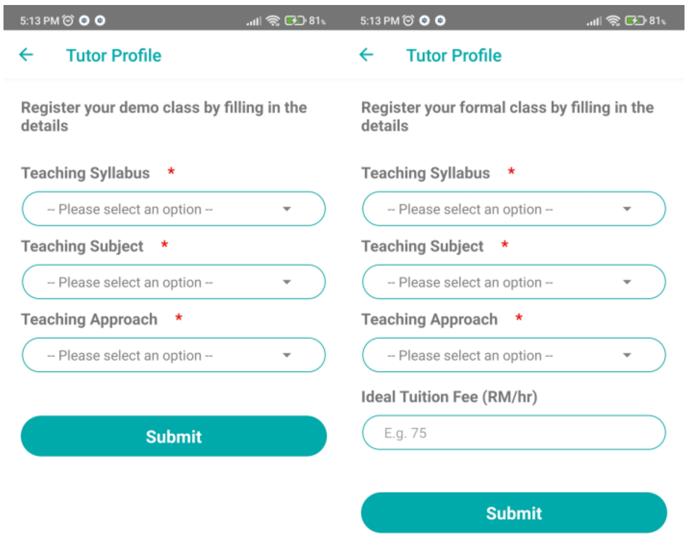
RM 57**/hr
Free Demo Class | Book Formal Class

Annonymous User
1-03-2022 12:43 PM
★★★★★
Good!!!!! Cuba Try !!!

Annonymous User
26-02-2022 00:13 AM
★★★★☆
good and patient tutor

Annonymous User
25-02-2022 00:13 AM
★★★★☆
No Content

Figure 5.42: UI – Tutor Seeker Tutor Profile


5:13 PM 81%

Tutor Profile

Register your demo class by filling in the details

Teaching Syllabus *
-- Please select an option --
Teaching Subject *
-- Please select an option --
Teaching Approach *
-- Please select an option --

Submit

Register your formal class by filling in the details

Teaching Syllabus *
-- Please select an option --
Teaching Subject *
-- Please select an option --
Teaching Approach *
-- Please select an option --
Ideal Tuition Fee (RM/hr)
E.g. 75

Submit

Figure 5.43: UI – Tutor Seeker Demo and Formal Class Registration

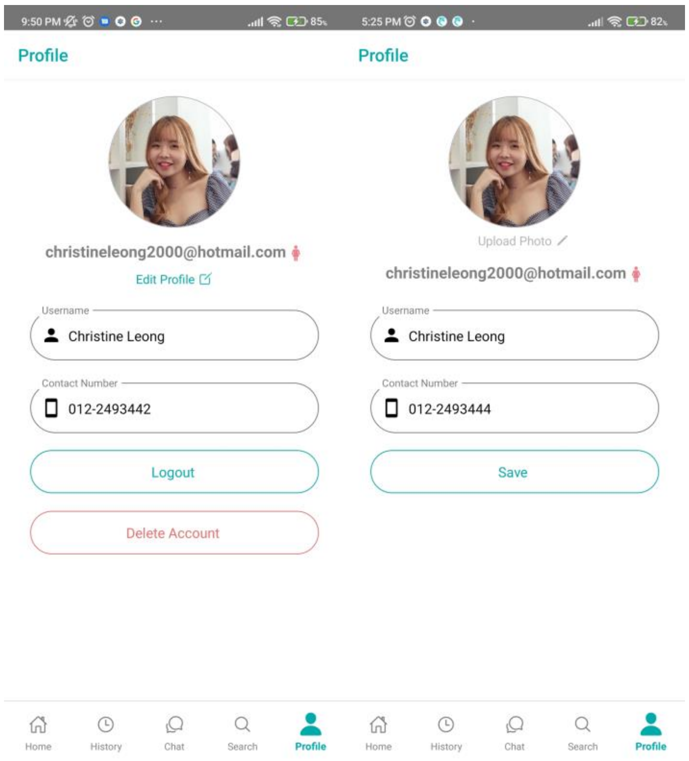
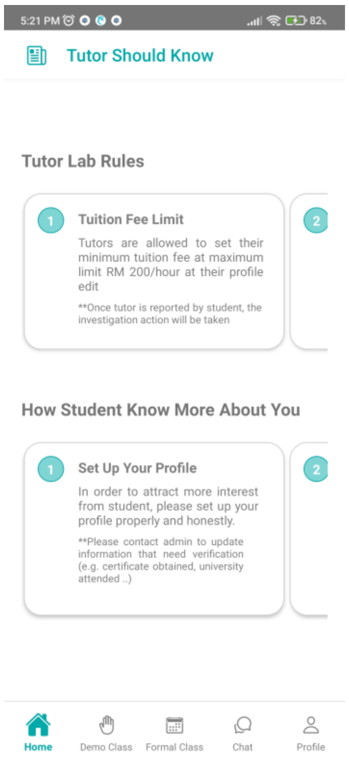


Figure 5.44: UI – Tutor Seeker Profile and Profile Edit

5.4.2.3 Tutor Interface



Tutor Lab Rules

1 Tuition Fee Limit

Tutors are allowed to set their minimum tuition fee at maximum limit RM 200/hour at their profile edit

**Once tutor is reported by student, the investigation action will be taken

2

How Student Know More About You

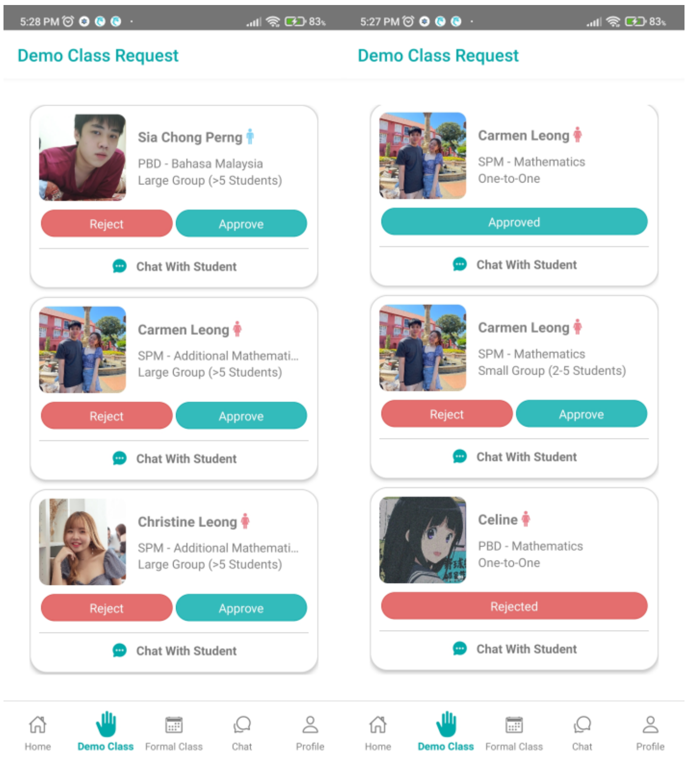
1 Set Up Your Profile

In order to attract more interest from student, please set up your profile properly and honestly.

**Please contact admin to update information that need verification (e.g. certificate obtained, university attended ..)

2

Figure 5.45: UI – Tutor Home



5:28 PM ⏰ ● ● ● · 📶 83%

5:27 PM ⏰ ● ● ● · 📶 83%

Demo Class Request

Demo Class Request

Sia Chong Perng
PBD - Bahasa Malaysia
Large Group (>5 Students)
Reject | Approve
Chat With Student

Carmen Leong
SPM - Additional Mathemati...
Large Group (>5 Students)
Reject | Approve
Chat With Student

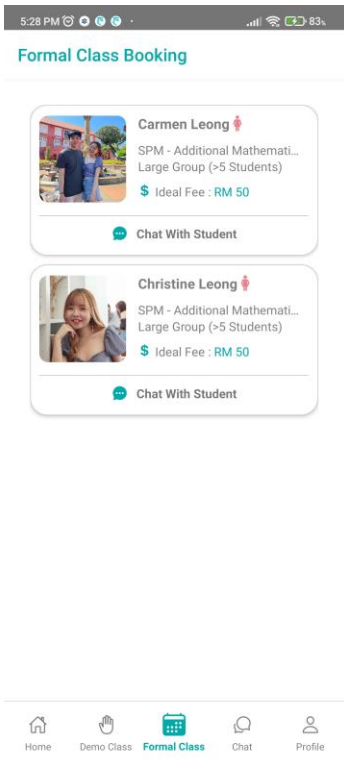
Christine Leong
SPM - Additional Mathemati...
Large Group (>5 Students)
Reject | Approve
Chat With Student

Carmen Leong
SPM - Mathematics
One-to-One
Approved
Chat With Student

Carmen Leong
SPM - Mathematics
Small Group (2-5 Students)
Reject | Approve
Chat With Student

Celine
PBD - Mathematics
One-to-One
Rejected
Chat With Student

Figure 5.46: UI – Tutor Demo Class Manage



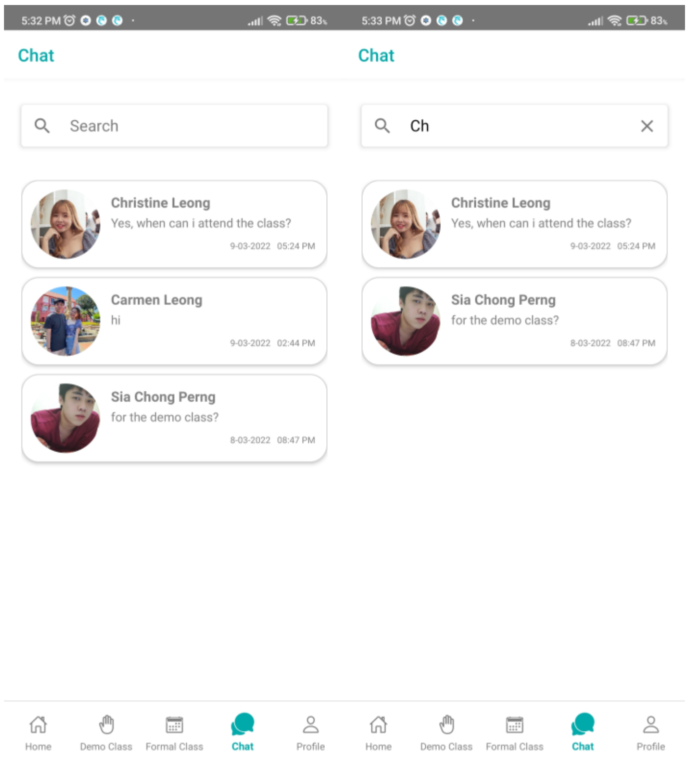
5:28 PM ⏰ ● ● ● · 📶 83%

Formal Class Booking

Carmen Leong
SPM - Additional Mathemati...
Large Group (>5 Students)
$ Ideal Fee : RM 50
Chat With Student

Christine Leong
SPM - Additional Mathemati...
Large Group (>5 Students)
$ Ideal Fee : RM 50
Chat With Student

Figure 5.47: UI – Tutor Formal Class Manage



5:32 PM · 83%

Chat

Search

Christine Leong
Yes, when can i attend the class?
9-03-2022 05:24 PM

Carmen Leong
hi
9-03-2022 02:44 PM

Sia Chong Perng
for the demo class?
8-03-2022 08:47 PM

5:33 PM · 83%

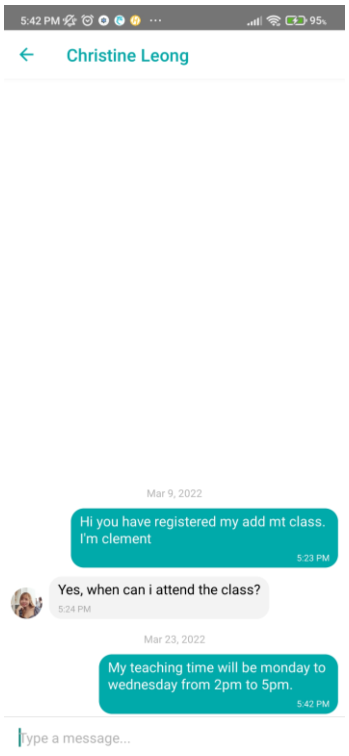
Chat

Ch

Christine Leong
Yes, when can i attend the class?
9-03-2022 05:24 PM

Sia Chong Perng
for the demo class?
8-03-2022 08:47 PM

Figure 5.48: UI – Tutor Chat History



5:42 PM · 95%

Christine Leong

Mar 9, 2022

Hi you have registered my add mt class. I'm clement
5:23 PM

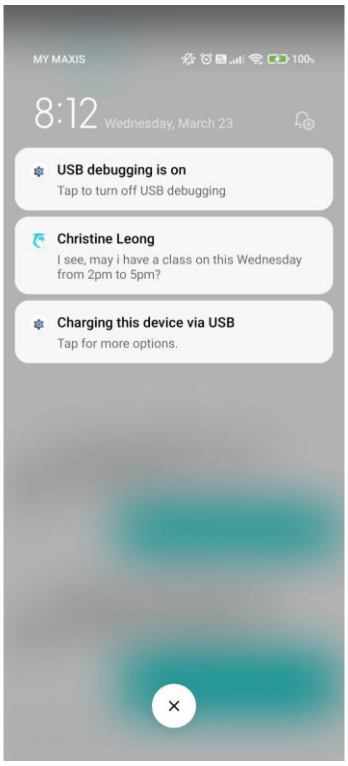
Yes, when can i attend the class?
5:24 PM

Mar 23, 2022

My teaching time will be monday to wednesday from 2pm to 5pm.
5:42 PM

Type a message...

Figure 5.49: UI – Tutor Chat Box


MY MAXIS

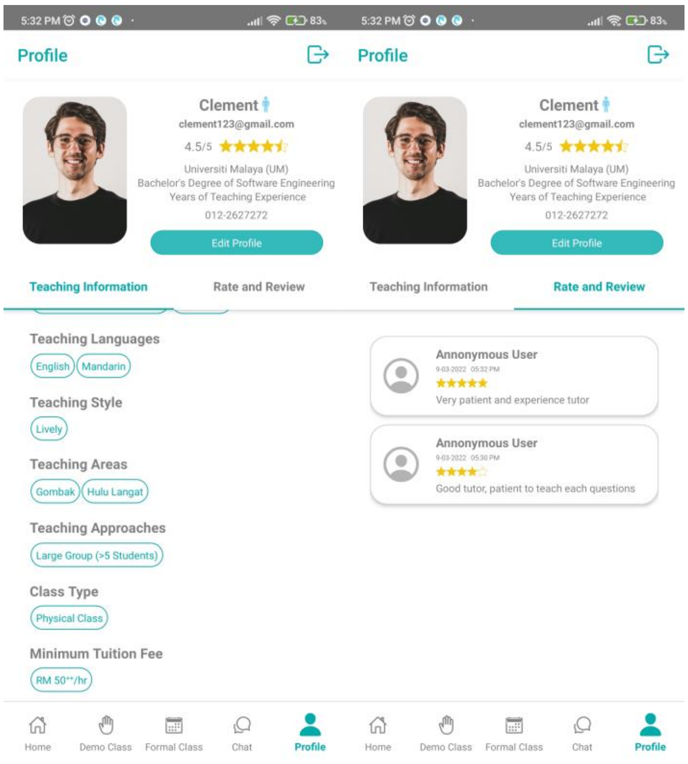
8:12 Wednesday, March 23

USB debugging is on
Tap to turn off USB debugging

Christine Leong
I see, may i have a class on this Wednesday from 2pm to 5pm?

Charging this device via USB
Tap for more options.

Figure 5.50: UI – Tutor Push Notification (From Tutor Seeker)


Profile

Clement
clement123@gmail.com
4.5/5
Universiti Malaya (UM)
Bachelor's Degree of Software Engineering
Years of Teaching Experience
012-2627272

Edit Profile

Teaching Information | Rate and Review

Teaching Languages
English, Mandarin

Teaching Style
Lively

Teaching Areas
Gombak, Hulu Langat

Teaching Approaches
Large Group (>5 Students)

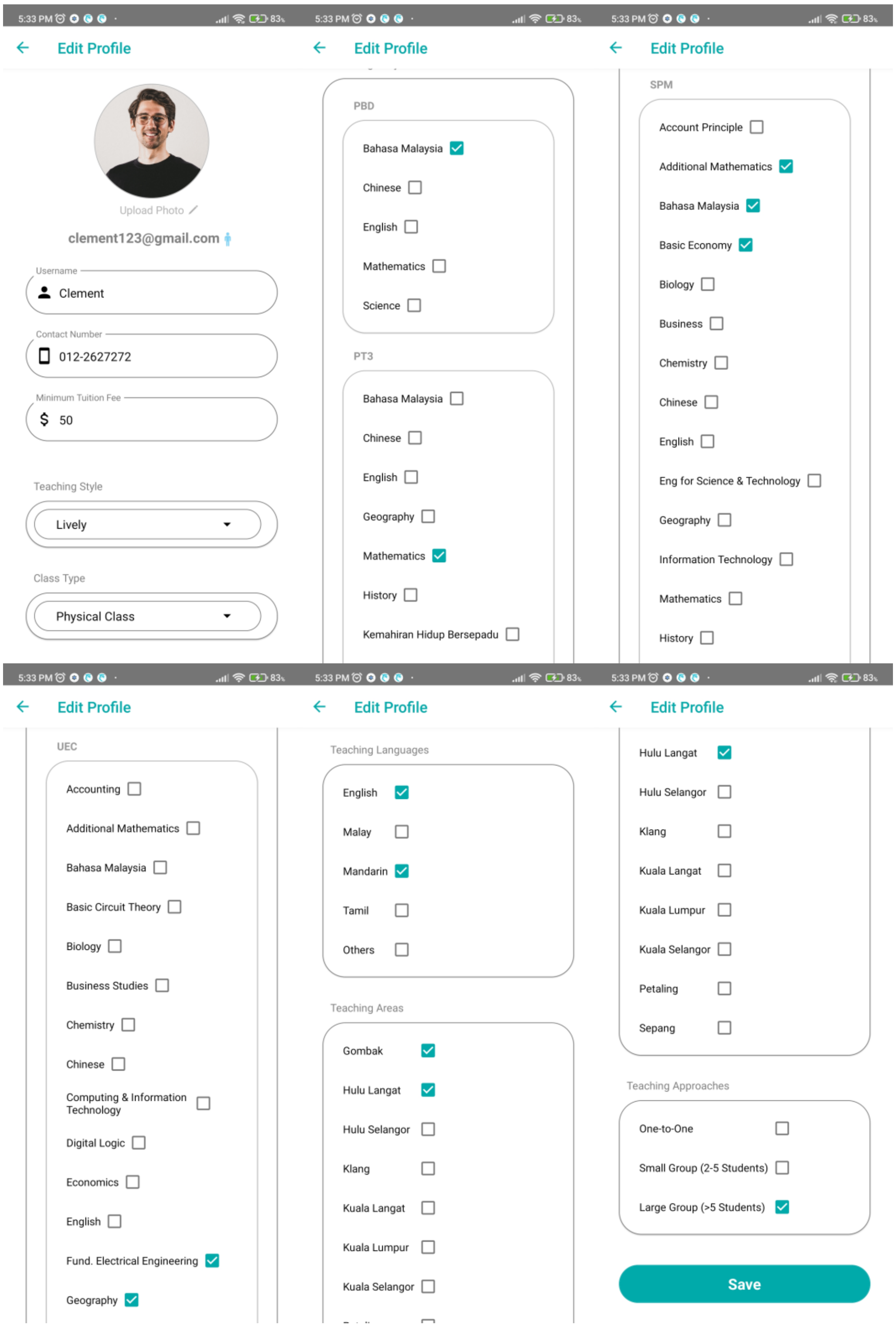
Class Type
Physical Class

Minimum Tuition Fee
RM 50**/hr

Annonymous User
9-03-2022 05:32 PM
Very patient and experience tutor

Annonymous User
9-03-2022 05:30 PM
Good tutor, patient to teach each questions

Figure 5.51: UI – Tutor Profile



5:33 PM 83%

Edit Profile

Upload Photo

clement123@gmail.com

Username	Clement
Contact Number	012-2627272
Minimum Tuition Fee	$ 50
Teaching Style	Lively
Class Type	Physical Class

5:33 PM 83%

Edit Profile

PBD

Bahasa Malaysia	✓
Chinese	☐
English	☐
Mathematics	☐
Science	☐

PT3

Bahasa Malaysia	☐
Chinese	☐
English	☐
Geography	☐
Mathematics	✓
History	☐
Kemahiran Hidup Bersepadu	☐

5:33 PM 83%

Edit Profile

SPM

Account Principle	☐
Additional Mathematics	✓
Bahasa Malaysia	✓
Basic Economy	✓
Biology	☐
Business	☐
Chemistry	☐
Chinese	☐
English	☐
Eng for Science & Technology	☐
Geography	☐
Information Technology	☐
Mathematics	☐
History	☐

5:33 PM 83%

Edit Profile

UEC

Accounting	☐
Additional Mathematics	☐
Bahasa Malaysia	☐
Basic Circuit Theory	☐
Biology	☐
Business Studies	☐
Chemistry	☐
Chinese	☐
Computing & Information Technology	☐
Digital Logic	☐
Economics	☐
English	☐
Fund. Electrical Engineering	✓
Geography	✓

5:33 PM 83%

Edit Profile

Teaching Languages

English	✓
Malay	☐
Mandarin	✓
Tamil	☐
Others	☐

Teaching Areas

Gombak	✓
Hulu Langat	✓
Hulu Selangor	☐
Klang	☐
Kuala Langat	☐
Kuala Lumpur	☐
Kuala Selangor	☐

5:33 PM 83%

Edit Profile

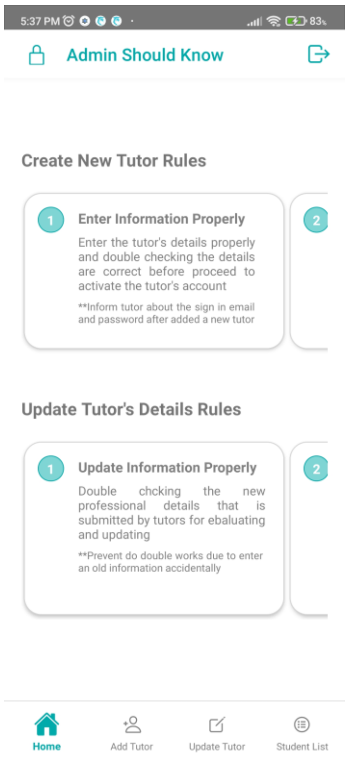
Hulu Langat	✓
Hulu Selangor	☐
Klang	☐
Kuala Langat	☐
Kuala Lumpur	☐
Kuala Selangor	☐
Petaling	☐
Sepang	☐

Teaching Approaches

One-to-One	☐
Small Group (2-5 Students)	☐
Large Group (>5 Students)	✓

Figure 5.52: UI – Tutor Profile Edit

5.4.2.4 Admin Interface



Create New Tutor Rules

1 Enter Information Properly

Enter the tutor's details properly and double checking the details are correct before proceed to activate the tutor's account

**Inform tutor about the sign in email and password after added a new tutor

2

Update Tutor's Details Rules

1 Update Information Properly

Double chcking the new professional details that is submitted by tutors for ebaluating and updating

**Prevent do double works due to enter an old information accidentally

2

Figure 5.53: UI – Admin Home

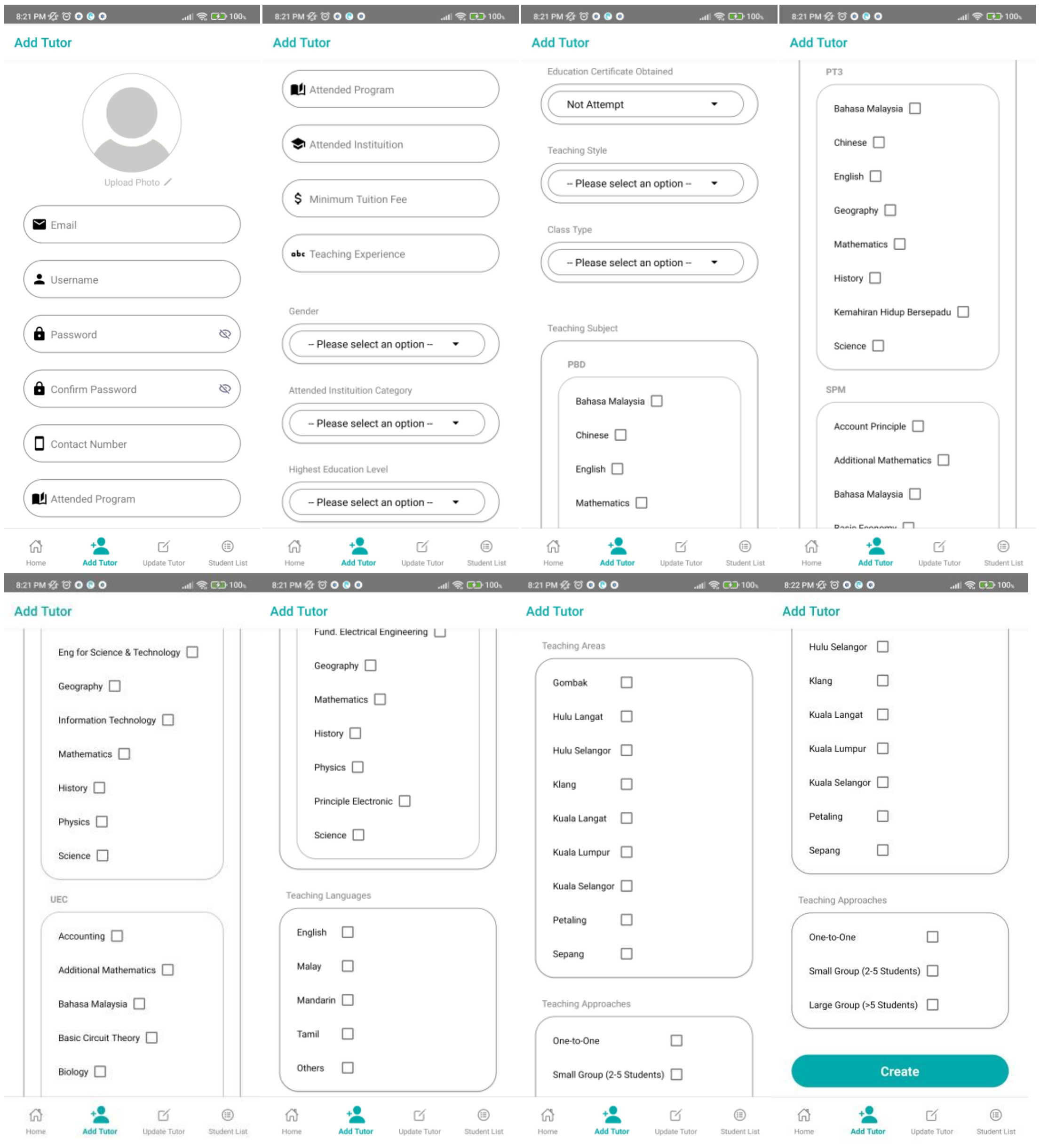
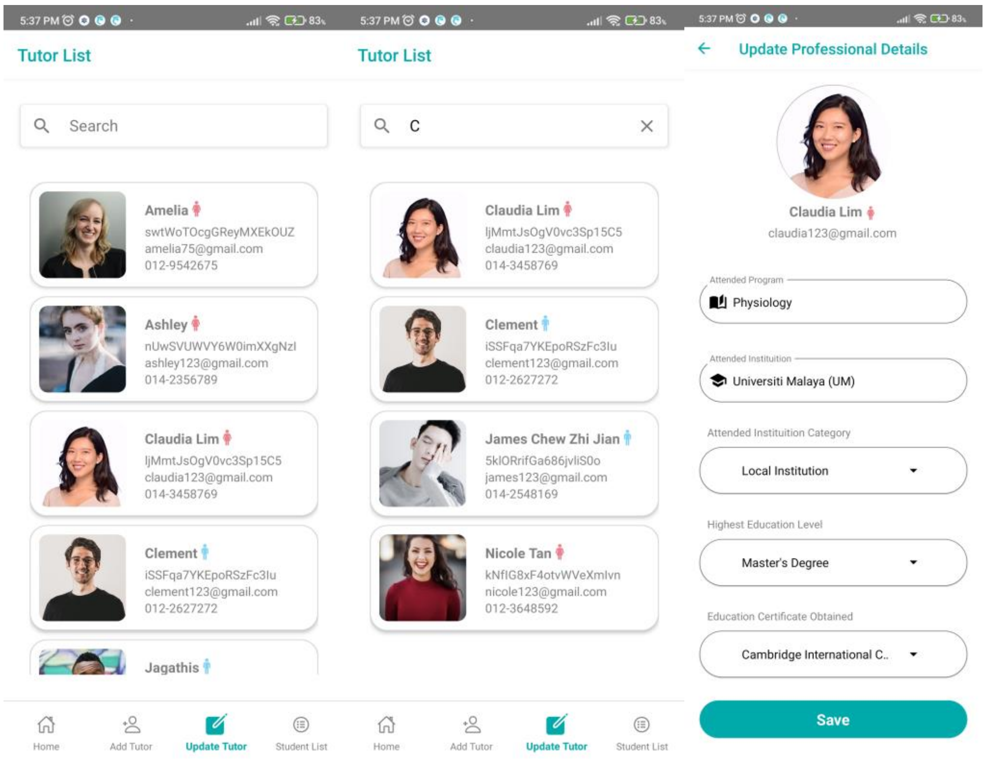


Figure 5.54: UI – Admin Add Tutor


Tutor List

Search

Amelia
swtWoTOcgGReyMXEkOUZ
amelia75@gmail.com
012-9542675

Ashley
nUwSVUWVY6W0imXXgNzl
ashley123@gmail.com
014-2356789

Claudia Lim
ljMmtJsOgV0vc3Sp15C5
claudia123@gmail.com
014-3458769

Clement
iSSFqa7YKEpoRSzFc3lu
clement123@gmail.com
012-2627272

Jagathis

Tutor List

C

Claudia Lim
ljMmtJsOgV0vc3Sp15C5
claudia123@gmail.com
014-3458769

Clement
iSSFqa7YKEpoRSzFc3lu
clement123@gmail.com
012-2627272

James Chew Zhi Jian
5kIORrifGa686jvliS0o
james123@gmail.com
014-2548169

Nicole Tan
kNfIG8xF4otvWVeXmlvn
nicole123@gmail.com
012-3648592

Update Professional Details

Claudia Lim
claudia123@gmail.com

Attended Program: Physiology

Attended Institution: Universiti Malaya (UM)

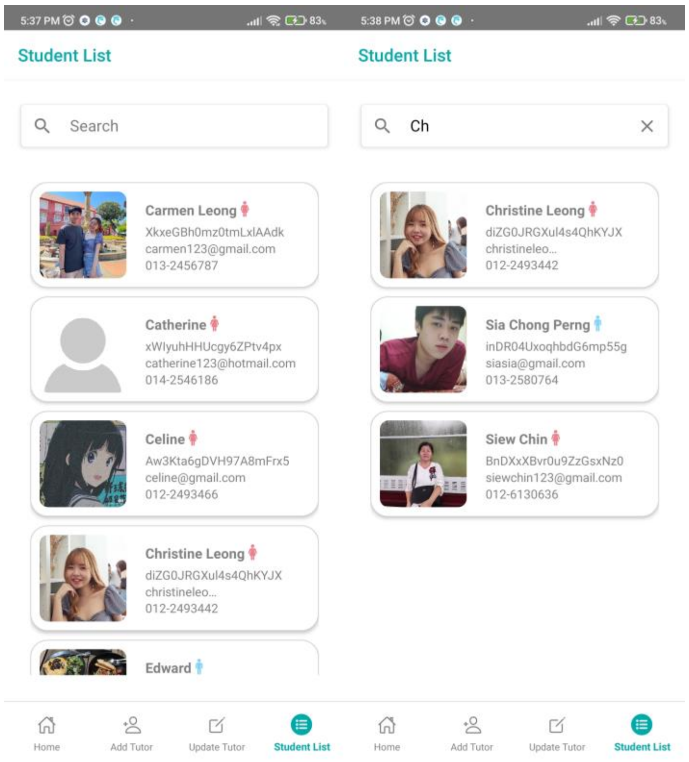
Attended Institution Category: Local Institution

Highest Education Level: Master's Degree

Education Certificate Obtained: Cambridge International C..

Save

Figure 5.55: UI – Admin Update Tutor


Student List

Search

Carmen Leong
XxkeGBh0mz0tmLxlAAdk
carmen123@gmail.com
013-2456787

Catherine
xWlyuhHHUcgy6ZPtv4px
catherine123@hotmail.com
014-2546186

Celine
Aw3Kta6gDVH97A8mFrx5
celine@gmail.com
012-2493466

Christine Leong
diZG0JRGXul4s4QhKYJX
christineleo...
012-2493442

Edward

Student List

Ch

Christine Leong
diZG0JRGXul4s4QhKYJX
christineleo...
012-2493442

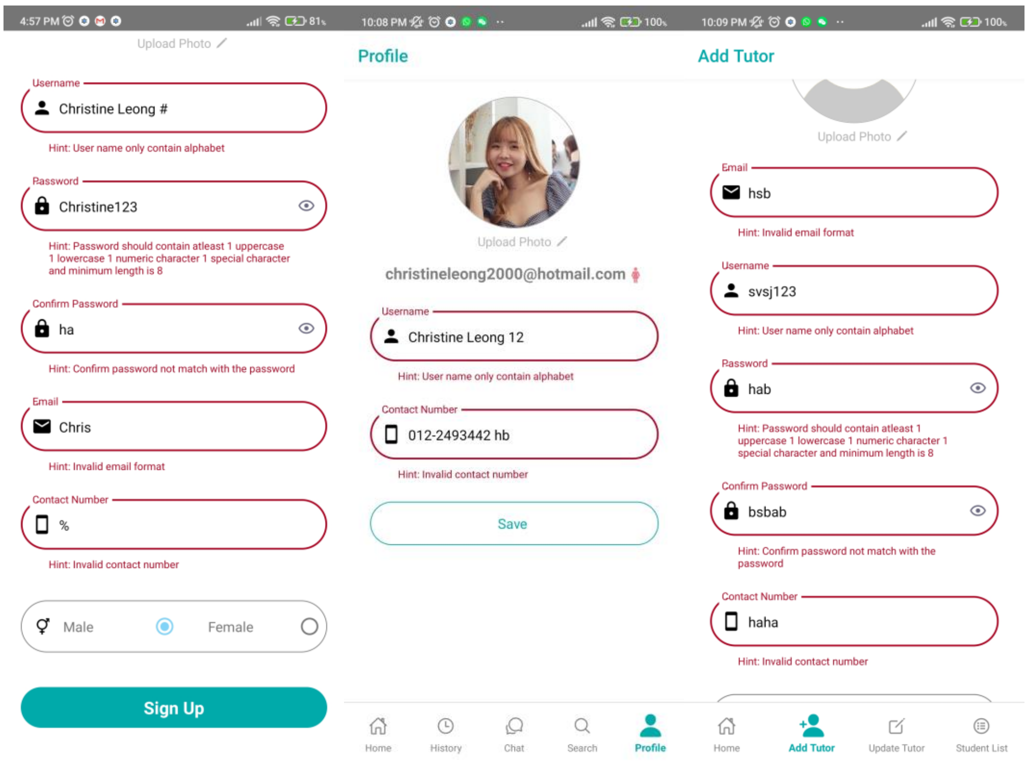
Sia Chong Perng
inDR04UxoqhbdG6mp55g
siasia@gmail.com
013-2580764

Siew Chin
BnDXxXBvr0u9ZzGsxNz0
siewchin123@gmail.com
012-6130636

Figure 5.56: UI – Admin Student List

5.4.2.5 Informative Feedback Design (IFD)

There are some informative feedback designs are applied in the user interface as a hint for users in improving the user friendliness of this application. For example, text helpers (let user know error input existing), activity indicator (while loading for data to let user waiting for result), password visibility toggle (easier to identify when enter wrong password), empty record display (improved new user experience), alert messages (informing users what they have perform incorrect or invalid) and so on. The figures below shown part of the informative feedback design in this application.



4:57 PM 81%

Upload Photo

Username: Christine Leong #

Hint: User name only contain alphabet

Password: Christine123

Hint: Password should contain atleast 1 uppercase 1 lowercase 1 numeric character 1 special character and minimum length is 8

Confirm Password: ha

Hint: Confirm password not match with the password

Email: Chris

Hint: Invalid email format

Contact Number: %

Hint: Invalid contact number

Male Female

Sign Up

10:08 PM 100%

Profile

Upload Photo

christineleong2000@hotmail.com

Username: Christine Leong 12

Hint: User name only contain alphabet

Contact Number: 012-2493442 hb

Hint: Invalid contact number

Save

Home History Chat Search Profile

10:09 PM 100%

Add Tutor

Upload Photo

Email: hsb

Hint: Invalid email format

Username: svsj123

Hint: User name only contain alphabet

Password: hab

Hint: Password should contain atleast 1 uppercase 1 lowercase 1 numeric character 1 special character and minimum length is 8

Confirm Password: bsbab

Hint: Confirm password not match with the password

Contact Number: haha

Hint: Invalid contact number

Home Add Tutor Update Tutor Student List

Figure 5.57: IFD – Helper Text (Input Error Handling)

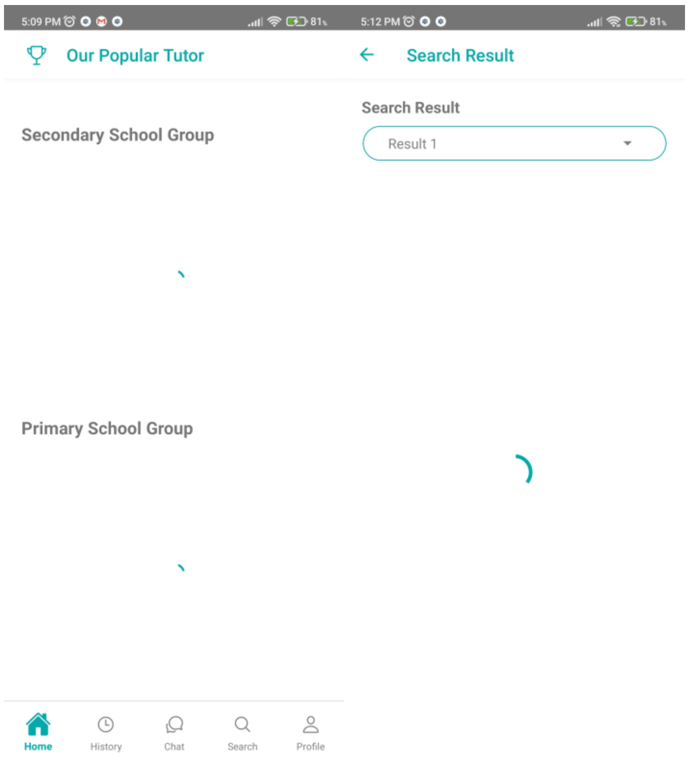


Figure 5.58: IFD – Activity Indicator

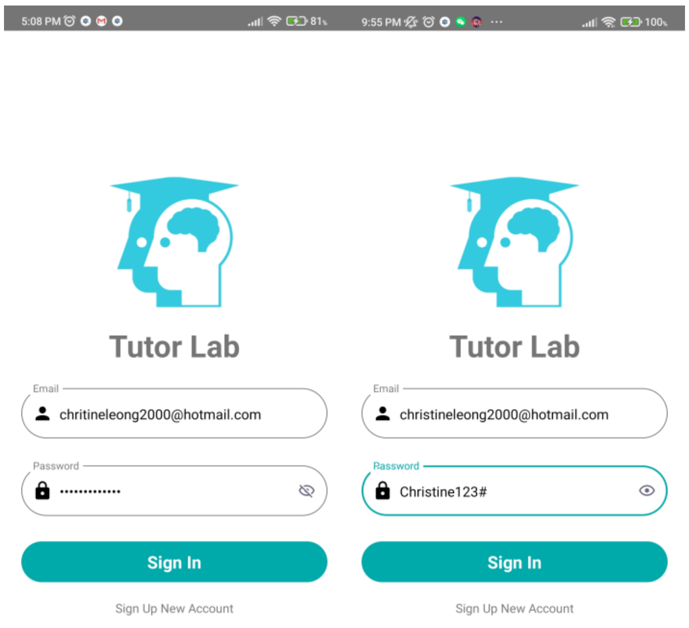


Figure 5.59: IFD – Password Visibility Toggle

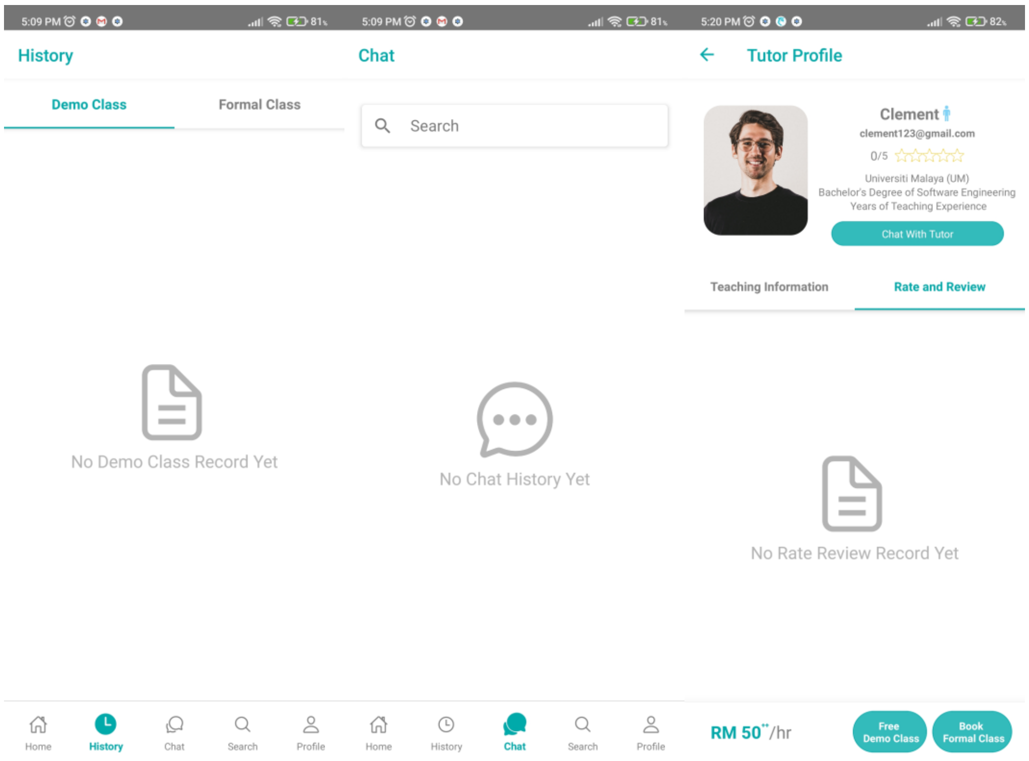
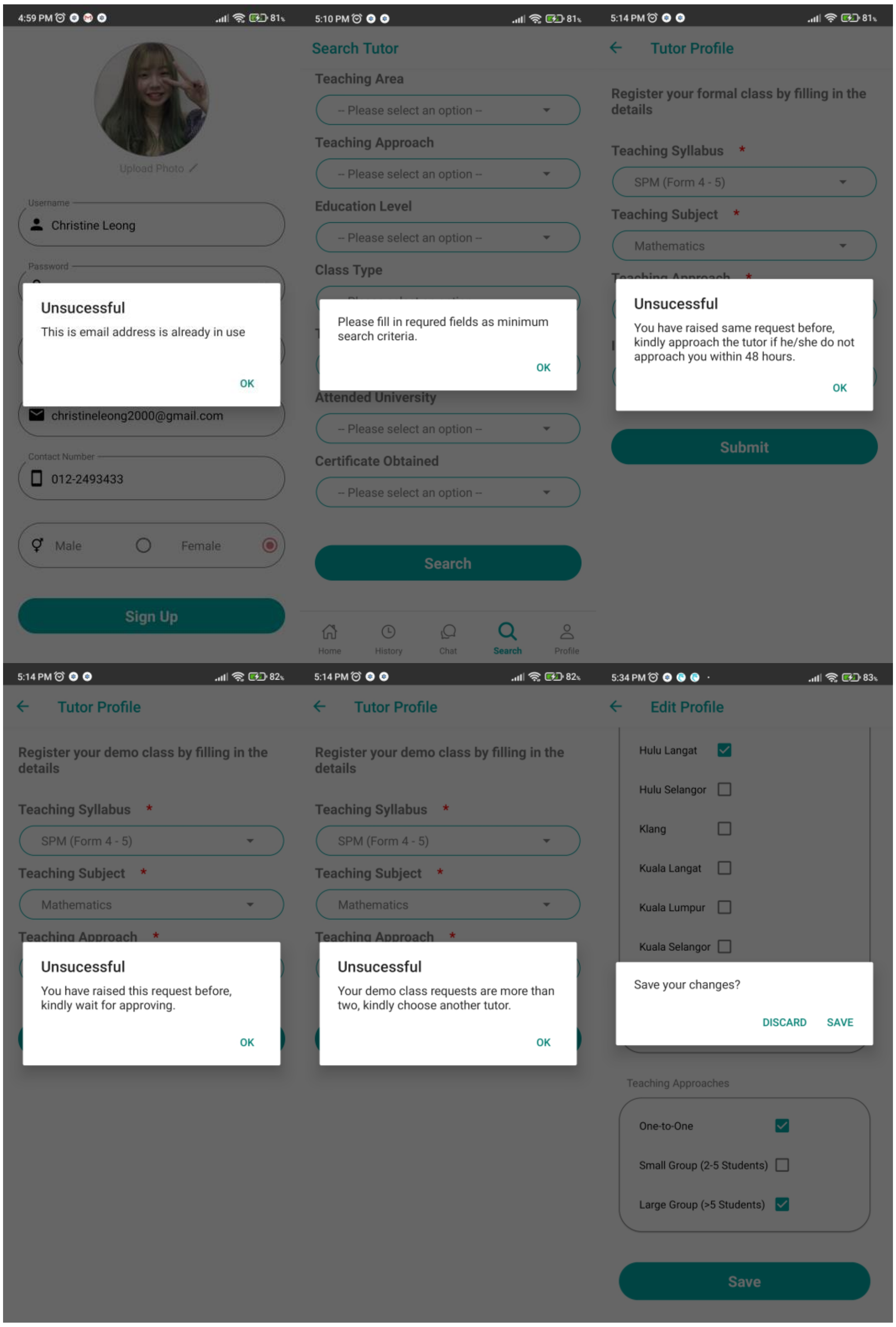


Figure 5.60: IFD – Empty Record Display


4:59 PM

Upload Photo

Username
Christine Leong

Password

Unsucessful
This is email address is already in use
OK

christineleong2000@gmail.com

Contact Number
012-2493433

Male ○ Female [✓ ●]

Sign Up

5:10 PM

Search Tutor

Teaching Area
-- Please select an option --

Teaching Approach
-- Please select an option --

Education Level
-- Please select an option --

Class Type

Please fill in requred fields as minimum search criteria.
OK

Attended University
-- Please select an option --

Certificate Obtained
-- Please select an option --

Search

Home History Chat Search Profile

5:14 PM

Tutor Profile

Register your formal class by filling in the details

Teaching Syllabus *
SPM (Form 4 - 5)

Teaching Subject *
Mathematics

Teaching Approach *

Unsucessful
You have raised same request before, kindly approach the tutor if he/she do not approach you within 48 hours.
OK

Submit

5:14 PM

Tutor Profile

Register your demo class by filling in the details

Teaching Syllabus *
SPM (Form 4 - 5)

Teaching Subject *
Mathematics

Teaching Approach *

Unsucessful
You have raised this request before, kindly wait for approving.
OK

5:14 PM

Tutor Profile

Register your demo class by filling in the details

Teaching Syllabus *
SPM (Form 4 - 5)

Teaching Subject *
Mathematics

Teaching Approach *

Unsucessful
Your demo class requests are more than two, kindly choose another tutor.
OK

5:34 PM

Edit Profile

Hulu Langat ✓
Hulu Selangor ☐
Klang ☐
Kuala Langat ☐
Kuala Lumpur ☐
Kuala Selangor ☐

Save your changes?
DISCARD SAVE

Teaching Approaches
One-to-One ✓
Small Group (2-5 Students) ☐
Large Group (>5 Students) ✓

Save

Figure 5.61: IFD – Alert Messages

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 Introduction

The system implementation is explained in this chapter which includes the API endpoint description, functions, and third parties service implementation. The implementations are explained with section codes to have a deeper understanding of the working of the functions. The implemented functions are categorized according to the different users' roles which are registered tutor seeker side, registered tutor side, and admin side. Last,

6.2 Application Programming Interface (API) Endpoint

There are 21 API endpoints is created in the server (Express.js) while the APIs are called by the presentation tier which is the user interface action (React Native). Some APIs could be reused in performing similar activities in different modules or functions while not only being used in performing only one function or module. The APIs are created for this application usage for performance and security purpose while not a well developed API from external developer or organisation.

Table 6.1: API Endpoint List

API Route	Method	Parameters / Raw Body	Description
/api/retrieveProfileInfo/:userId/:role	GET	- userId - role	- Retrieve tutor teaching and basic information. (Confidential information only be returned when admin or tutor side call this API)
/api/retrieveClass/:userId	GET	- userId	- Retrieve requested demo classes and formal classes information with the tutors' basic information.
/api/retrievePopularTutor/:category	GET	- category	- Retrieve basic information of popular tutors according to school categories.
/api/retrieveRateReview/:tutorId	GET	- tutorId	- Retrieve rates and reviews of the tutor. - Calculate and return the average rates of the tutor for displaying.
/api/retrieveTutorFormalClass/:userId	GET	- userId	- Retrieve requested formal classes information with the tutor seekers' basic information.
/api/retrieveTutorDemoClass/:userId	GET	- userId	- Retrieve requested demo classes information with the tutor seekers' basic information.
/api/retrieveUserList/:role	GET	- role	- Retrieve users' basic information according to the user role.
/api/retrieveChatUserList/:userId	GET	- userId	- Retrieve chat users' basic information for the chat history list displaying.
/api/deleteAccount/:userId	GET	- userId	- Delete the existing account in the 'users' collection.
/api/similarityMeasure	POST	- similarityMtd	- Calculate the similarity percentage of all tutors in

		- tchSyllabus - tchSubject - tchArea - tchLanguage - tchStyle - tchExperience - tchApproach - eduLvl - classType - minFee - attdInsCat - certObt	the database with the ideal tutor of tutor seeker. - Calculation applied Euclidean distance, Minkowski distance, Manhattan distance, Cosine Similarity, and Jaccard Similarity Coefficient algorithms. - Retrieve 20 tutors' basic information with the highest similarity percentage.
/api/registerDemoClass	POST	- userId - tutorId - tchSyllabus - tchSubject - tchApproach	- Create demo class registration document.
/api/registerFormalClass	POST	- userId - tutorId - tchSyllabus - tchSubject - tchApproach	- Create formal class registration document.
/api/rateReview	POST	- userId - tutorId - rate	- Create a rate review document (for new record) - Update rate review document (for existing record)

		- review	
/api/registerBasicAcc	POST	- name - email - contactNo - gender - profilePic - role - fcmToken	- Create a users document with student role (tutor seeker role)
/api/updateBasicProfile	POST	- userId - name - contactNo - profilePic	- Update basic information of users document.
/api/updateBasicTutorInfo	POST	- tutorDocId - tchSubject - tchLanguage - tchArea - tchApproach - classType - tchStyle - minFee	- Update tutor information document (exclude professional information, professional information updating is handled in another API)
/api/updateProfessionalTutorInfo	POST	- tutorDocId - attdIns - attdInsCat - attdPgm - eduLvl	- Update professional information of tutor information document.

		- certObt	
/api/createTutorInfo	POST	- userId - attdIns - attdPgm - attdInsCat - certObt - eduLvl - tchSubject - tchLanguage - tchStyle - tchArea - tchApproach - classType - minFee - tchExperience	- Create tutor information document.
/api/updateFCMToken	POST	- userId - fcmToken	- Update fcmToken value users document.

6.3 Application Implementation

The implementation of the application is explained according to activities in different screens while the activities are grouped by the role of users which is tutor seeker side, tutor side, and admin side. The section codes are inserted as figures with the display result in explaining the implementation of API, library, third-parties services, and so on.

6.3.1 Sign Up Activity

Tutor seeker is allowed to sign up for an account with his/her email address while the tutor's account only can be created by the admin which will be explained in the following part.

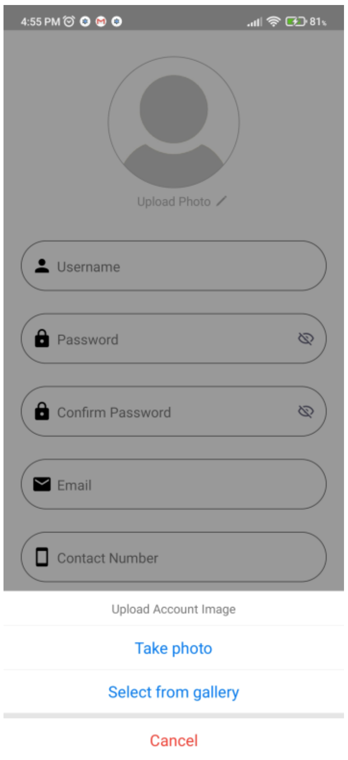
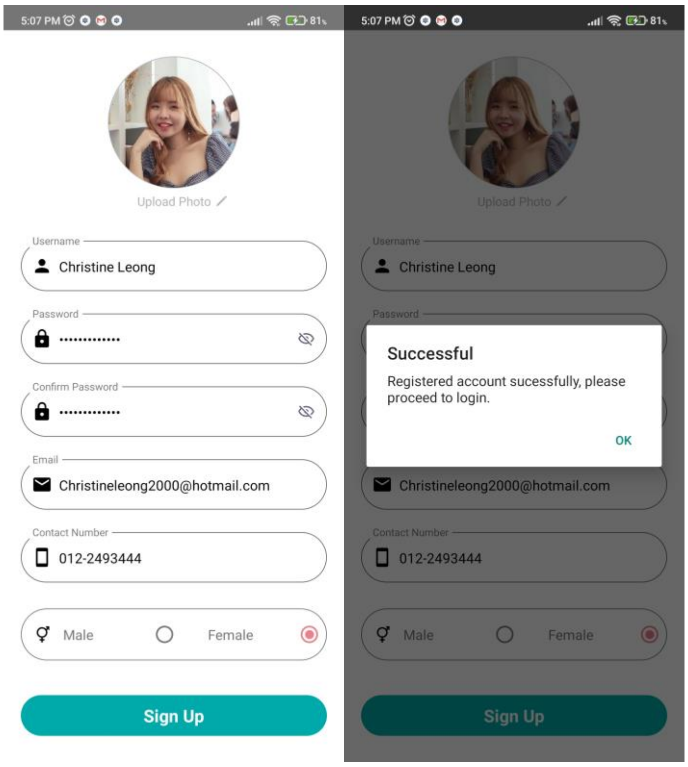


Figure 6.1: Upload Profile Picture



5:07 PM

Username: Christine Leong

Password: ••••••••••••••

Confirm Password: ••••••••••••••

Email: Christineleong2000@hotmail.com

Contact Number: 012-2493444

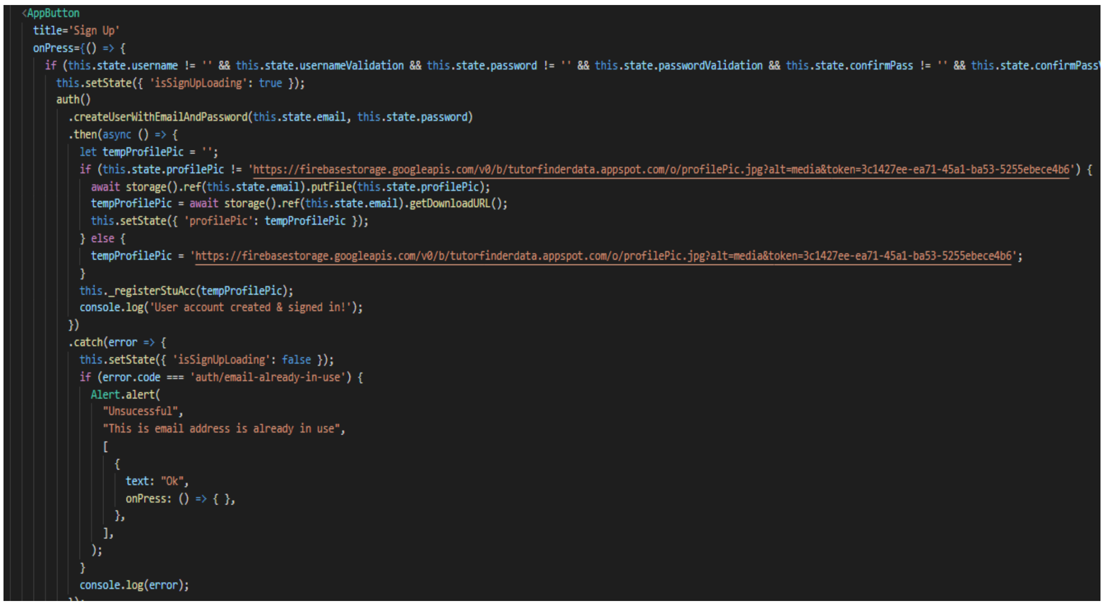
Male ○ Female ◉

Successful

Registered account sucessfully, please proceed to login.

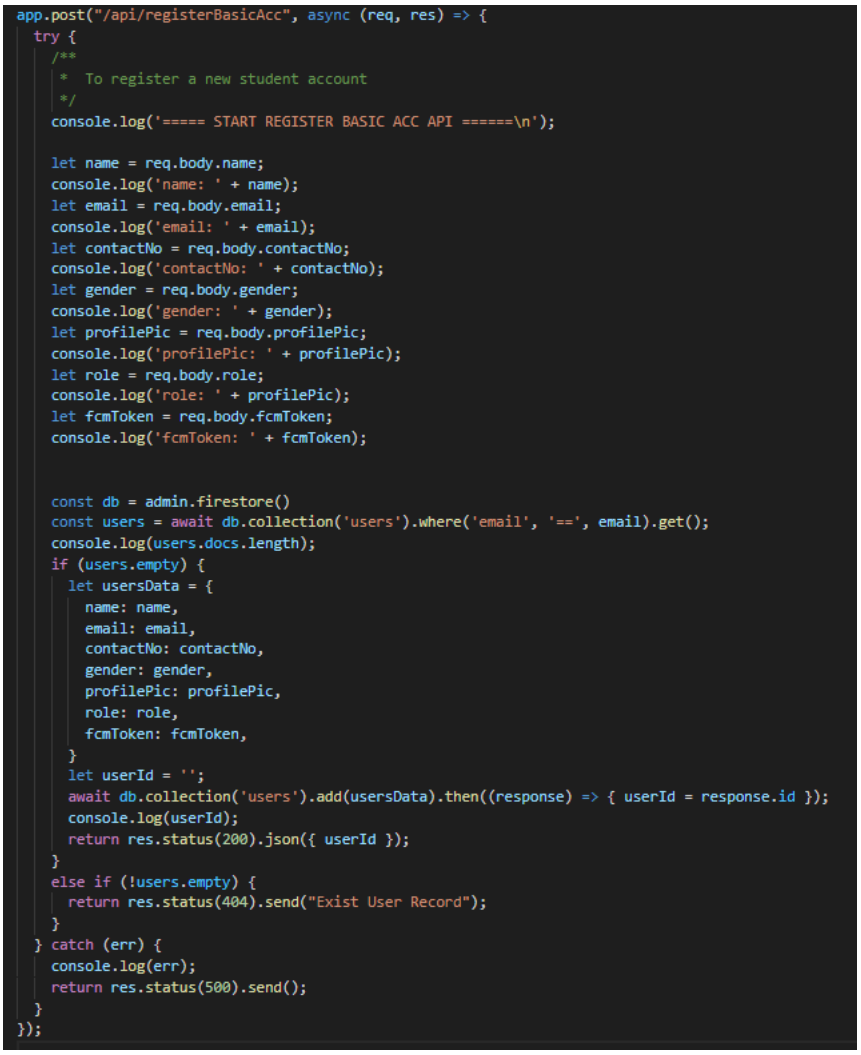
Figure 6.2: Sign Up Tutor Seeker Account

There is an upload image activity in the signup function that involves the implementation of external libraries, API, and third-party services before proceeding to sign up for an account. The external library 'react-native-actionsheet' is imported, the upload account image action sheet is displayed to tutor seeker in order to take their profile picture by using the phone’s camera or select from their gallery. The external library 'react-native-image-picker' enables the application to access the phone’s camera and gallery to get the photo. The input validation is handled and will be explained in the flowing section.



```
<AppButton
  title='Sign Up'
  onPress={() => {
    if (this.state.username != '' && this.state.usernameValidation && this.state.password != '' && this.state.passwordValidation && this.state.confirmPass != '' && this.state.confirmPass
      this.setState({ 'isSignUpLoading': true });
      auth()
        .createUserWithEmailAndPassword(this.state.email, this.state.password)
        .then(async () => {
          let tempProfilePic = '';
          if (this.state.profilePic != 'https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/o/profilePic.jpg?alt=media&token=3c1427ee-ea71-45a1-ba53-5255ebece4b6') {
            await storage().ref(this.state.email).putFile(this.state.profilePic);
            tempProfilePic = await storage().ref(this.state.email).getDownloadURL();
            this.setState({ 'profilePic': tempProfilePic });
          } else {
            tempProfilePic = 'https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/o/profilePic.jpg?alt=media&token=3c1427ee-ea71-45a1-ba53-5255ebece4b6';
          }
          this._registerStuAcc(tempProfilePic);
          console.log('User account created & signed in!');
        })
        .catch(error => {
          this.setState({ 'isSignUpLoading': false });
          if (error.code === 'auth/email-already-in-use') {
            Alert.alert(
              "Unsucessful",
              "This is email address is already in use",
              [
                {
                  text: "Ok",
                  onPress: () => { },
                },
              ],
            );
          }
          console.log(error);
```

Figure 6.3: Section Code (Front-end) – Sign Up Tutor Seeker Account



```
app.post("/api/registerBasicAcc", async (req, res) => {
  try {
    /**
     *  To register a new student account
     */
    console.log('===== START REGISTER BASIC ACC API ======\n');

    let name = req.body.name;
    console.log('name: ' + name);
    let email = req.body.email;
    console.log('email: ' + email);
    let contactNo = req.body.contactNo;
    console.log('contactNo: ' + contactNo);
    let gender = req.body.gender;
    console.log('gender: ' + gender);
    let profilePic = req.body.profilePic;
    console.log('profilePic: ' + profilePic);
    let role = req.body.role;
    console.log('role: ' + profilePic);
    let fcmToken = req.body.fcmToken;
    console.log('fcmToken: ' + fcmToken);


    const db = admin.firestore()
    const users = await db.collection('users').where('email', '==', email).get();
    console.log(users.docs.length);
    if (users.empty) {
      let usersData = {
        name: name,
        email: email,
        contactNo: contactNo,
        gender: gender,
        profilePic: profilePic,
        role: role,
        fcmToken: fcmToken,
      }
      let userId = '';
      await db.collection('users').add(usersData).then((response) => { userId = response.id });
      console.log(userId);
      return res.status(200).json({ userId });
    }
    else if (!users.empty) {
      return res.status(404).send("Exist User Record");
    }
  } catch (err) {
    console.log(err);
    return res.status(500).send();
  }
});
```

Figure 6.4: Section Code (Server API) – Sign Up Tutor Seeker Account

The upload photo action, create account action will be triggered when the "Sign Up" button is pressed. As the figure above shown, auth() – createUserWithEmailAndPassword function is called from the third party service (Firebase Authentication) to create an account record in the user list of the linked Firebase Authentication. Then, the uploaded new photo from the phone is stored to Firebase Storage while the filename is the email address of the tutor seeker (default profile picture is used if no photo is uploaded by the tutor seeker). Next, the "/api/registerBasicAcc" API is called to insert the new tutor seeker's information into the 'users' collection as a new document. Redirecting the tutor seeker to sign up screen after the server created the 'users' document and return a 200 status back to the front-end.

6.3.2 Sign In Activity

A registered tutor seeker, registered tutor, and admin are allowed to sign in to the application via the same sign-in screen.

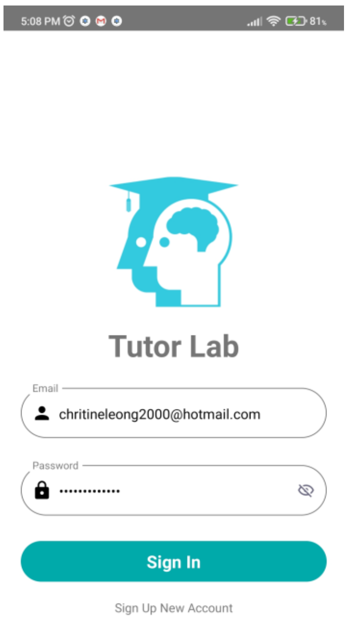
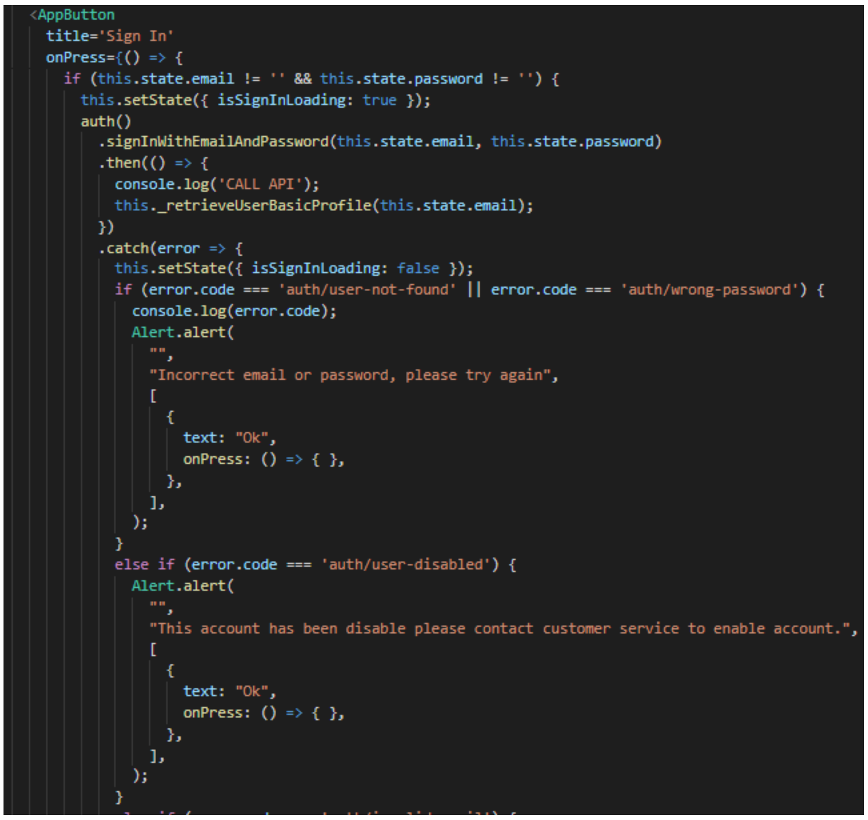
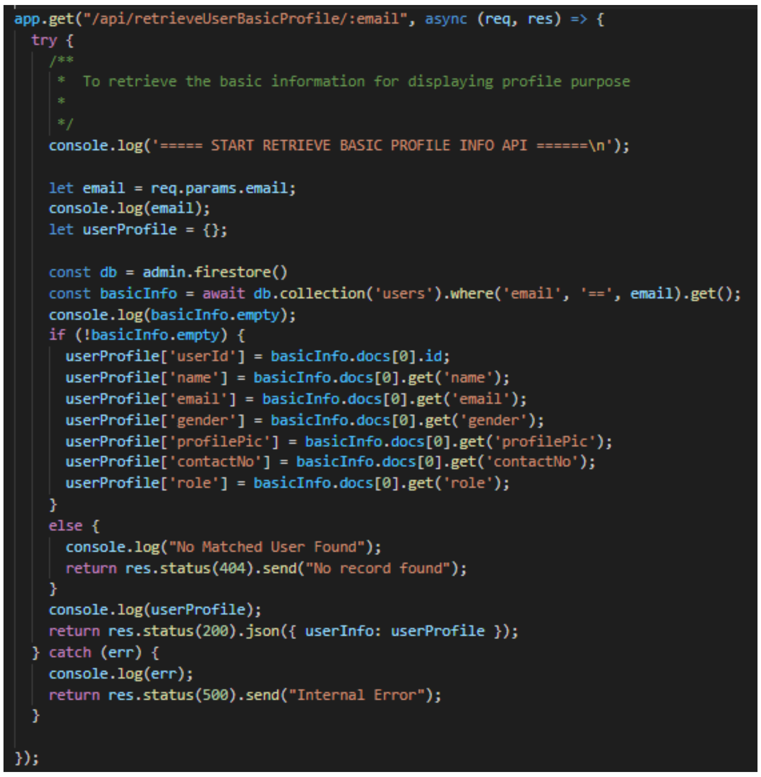


Figure 6.5: Sign In Account



```
<AppButton
  title='Sign In'
  onPress={() => {
    if (this.state.email != '' && this.state.password != '') {
      this.setState({ isSignInLoading: true });
      auth()
        .signInWithEmailAndPassword(this.state.email, this.state.password)
        .then(() => {
          console.log('CALL API');
          this._retrieveUserBasicProfile(this.state.email);
        })
        .catch(error => {
          this.setState({ isSignInLoading: false });
          if (error.code === 'auth/user-not-found' || error.code === 'auth/wrong-password') {
            console.log(error.code);
            Alert.alert(
              "",
              "Incorrect email or password, please try again",
              [
                {
                  text: "Ok",
                  onPress: () => { },
                },
              ],
            );
          }
          else if (error.code === 'auth/user-disabled') {
            Alert.alert(
              "",
              "This account has been disable please contact customer service to enable account.",
              [
                {
                  text: "Ok",
                  onPress: () => { },
                },
              ],
            );
          }
```

Figure 6.6: Section Code (Front-end) – Sign In Account



```
app.get("/api/retrieveUserBasicProfile/:email", async (req, res) => {
  try {
    /**
     *  To retrieve the basic information for displaying profile purpose
     *
     */
    console.log('===== START RETRIEVE BASIC PROFILE INFO API ======\n');

    let email = req.params.email;
    console.log(email);
    let userProfile = {};

    const db = admin.firestore()
    const basicInfo = await db.collection('users').where('email', '==', email).get();
    console.log(basicInfo.empty);
    if (!basicInfo.empty) {
      userProfile['userId'] = basicInfo.docs[0].id;
      userProfile['name'] = basicInfo.docs[0].get('name');
      userProfile['email'] = basicInfo.docs[0].get('email');
      userProfile['gender'] = basicInfo.docs[0].get('gender');
      userProfile['profilePic'] = basicInfo.docs[0].get('profilePic');
      userProfile['contactNo'] = basicInfo.docs[0].get('contactNo');
      userProfile['role'] = basicInfo.docs[0].get('role');
    }
    else {
      console.log("No Matched User Found");
      return res.status(404).send("No record found");
    }
    console.log(userProfile);
    return res.status(200).json({ userInfo: userProfile });
  } catch (err) {
    console.log(err);
    return res.status(500).send("Internal Error");
  }

});
```

Figure 6.7: Section Code (Server API) – Retrieve Basic Profile

A third-party service – Firebase Authentication is applied to check the authentication of the input email address. The "/api/retrieveUserBasicProfile/:email" API is called after validating the email is authenticated. API is used to retrieve the basic information of the authenticated user from the 'users' collection in Firestore according to the email. Then, data is returned to the front-end in JSON format with status. The returned information is passed to the home screen for the following function. Besides, the role data that is returned from the server identifies the respective redirected tab which is the student, tutor, or admin tab that is shown in CHAPTER 5.

6.3.3 Registered Tutor Seeker Side

6.3.3.1 Home Activities

The home stack on the registered tutor seeker side displays the popular tutor according to a primary and secondary school group. At the same time, the home screen is responsible for handling AsyncStorage, Firebase Cloud Messaging (FCM) token and push notification display settings.

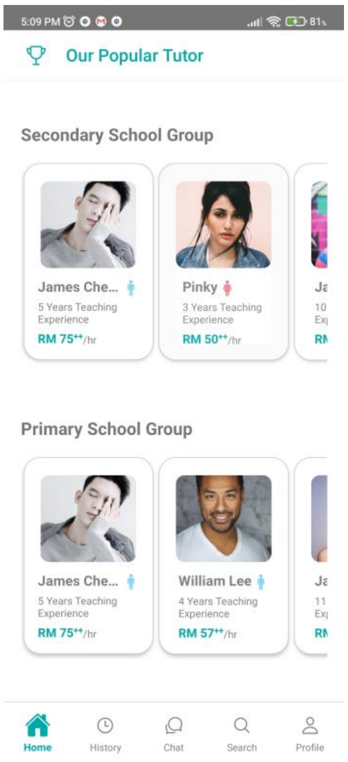
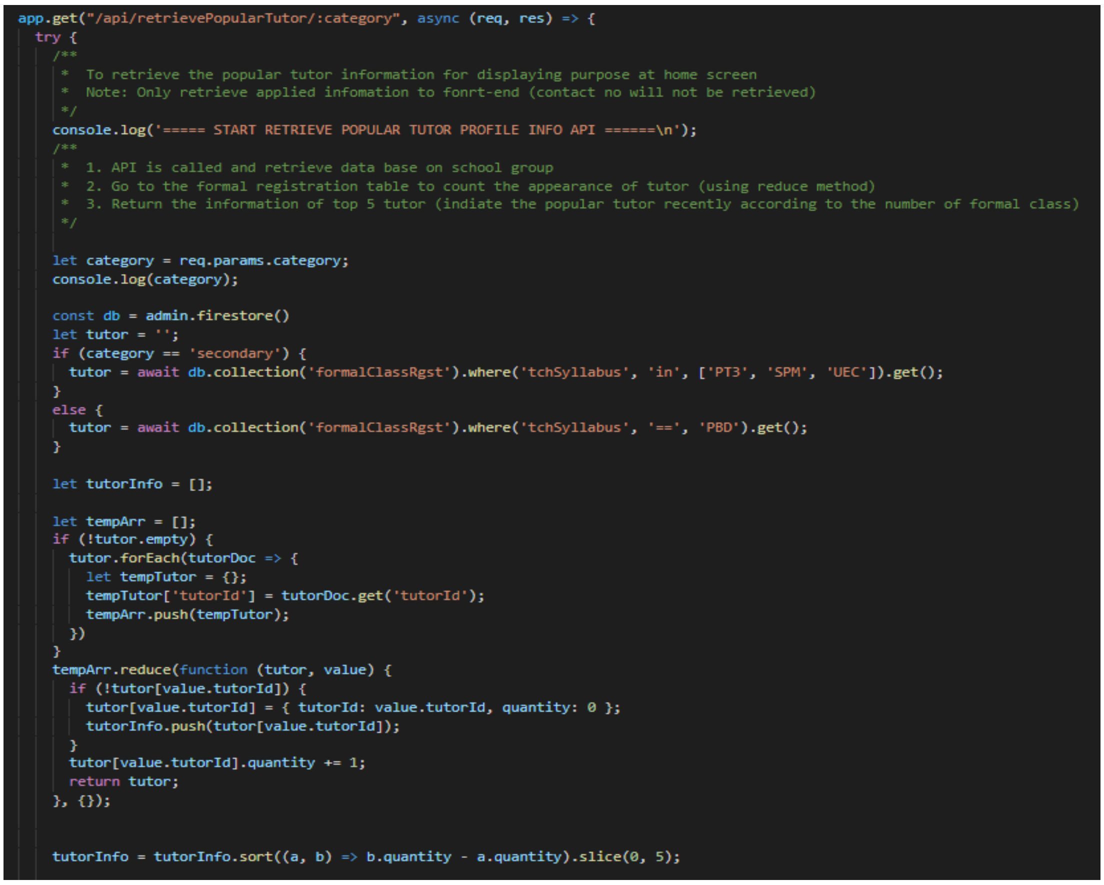
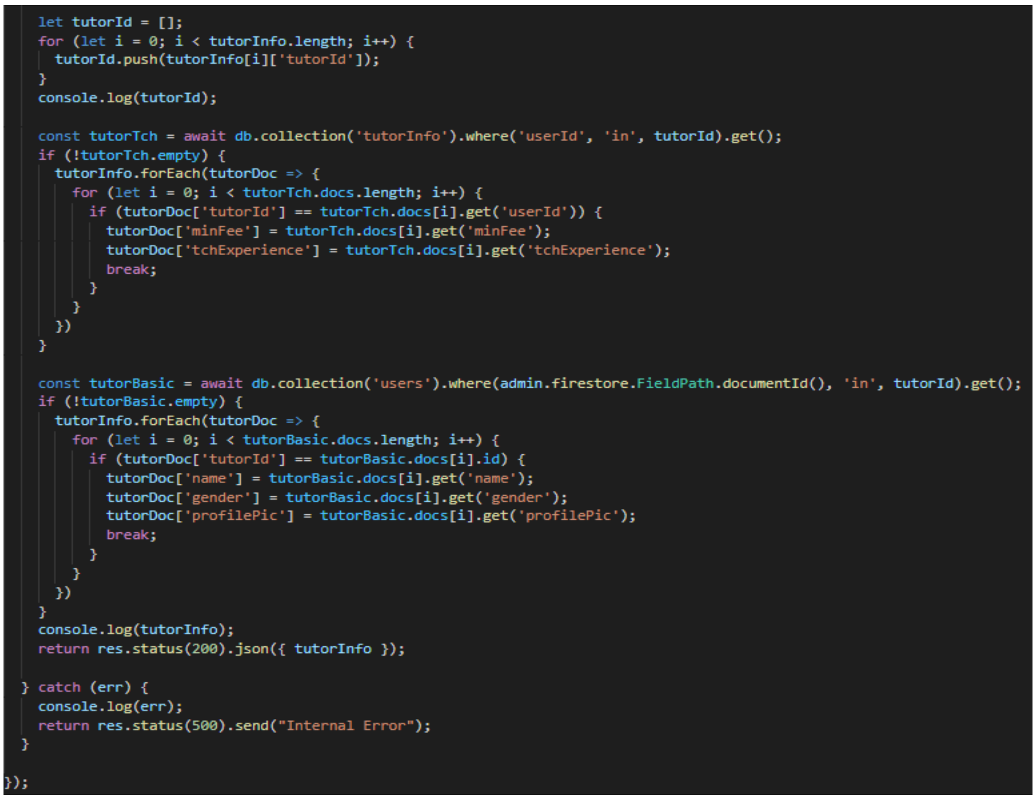


Figure 6.8: Home (Registered Tutor Seeker)



```
app.get("/api/retrievePopularTutor/:category", async (req, res) => {
  try {
    /**
     *  To retrieve the popular tutor information for displaying purpose at home screen
     *  Note: Only retrieve applied infomation to fonrt-end (contact no will not be retrieved)
     */
    console.log('===== START RETRIEVE POPULAR TUTOR PROFILE INFO API ======\n');
    /**
     *  1. API is called and retrieve data base on school group
     *  2. Go to the formal registration table to count the appearance of tutor (using reduce method)
     *  3. Return the information of top 5 tutor (indiate the popular tutor recently according to the number of formal class)
     */

    let category = req.params.category;
    console.log(category);

    const db = admin.firestore()
    let tutor = '';
    if (category == 'secondary') {
      tutor = await db.collection('formalClassRgst').where('tchSyllabus', 'in', ['PT3', 'SPM', 'UEC']).get();
    }
    else {
      tutor = await db.collection('formalClassRgst').where('tchSyllabus', '==', 'PBD').get();
    }

    let tutorInfo = [];

    let tempArr = [];
    if (!tutor.empty) {
      tutor.forEach(tutorDoc => {
        let tempTutor = {};
        tempTutor['tutorId'] = tutorDoc.get('tutorId');
        tempArr.push(tempTutor);
      })
    }
    tempArr.reduce(function (tutor, value) {
      if (!tutor[value.tutorId]) {
        tutor[value.tutorId] = { tutorId: value.tutorId, quantity: 0 };
        tutorInfo.push(tutor[value.tutorId]);
      }
      tutor[value.tutorId].quantity += 1;
      return tutor;
    }, {});


    tutorInfo = tutorInfo.sort((a, b) => b.quantity - a.quantity).slice(0, 5);
```

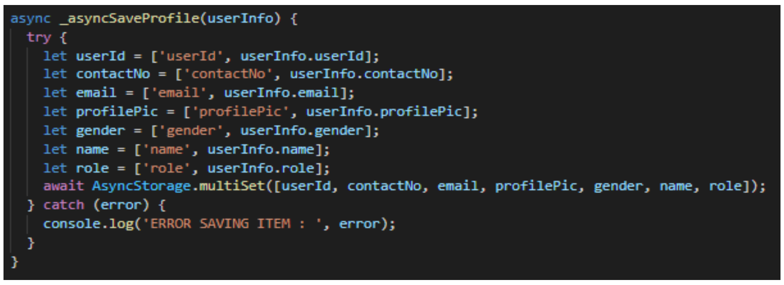
Figure 6.9: Section Code (Server API) – Retrieve Popular Tutor Part 1



```
    let tutorId = [];
    for (let i = 0; i < tutorInfo.length; i++) {
      tutorId.push(tutorInfo[i]['tutorId']);
    }
    console.log(tutorId);

    const tutorTch = await db.collection('tutorInfo').where('userId', 'in', tutorId).get();
    if (!tutorTch.empty) {
      tutorInfo.forEach(tutorDoc => {
        for (let i = 0; i < tutorTch.docs.length; i++) {
          if (tutorDoc['tutorId'] == tutorTch.docs[i].get('userId')) {
            tutorDoc['minFee'] = tutorTch.docs[i].get('minFee');
            tutorDoc['tchExperience'] = tutorTch.docs[i].get('tchExperience');
            break;
          }
        }
      })
    }

    const tutorBasic = await db.collection('users').where(admin.firestore.FieldPath.documentId(), 'in', tutorId).get();
    if (!tutorBasic.empty) {
      tutorInfo.forEach(tutorDoc => {
        for (let i = 0; i < tutorBasic.docs.length; i++) {
          if (tutorDoc['tutorId'] == tutorBasic.docs[i].id) {
            tutorDoc['name'] = tutorBasic.docs[i].get('name');
            tutorDoc['gender'] = tutorBasic.docs[i].get('gender');
            tutorDoc['profilePic'] = tutorBasic.docs[i].get('profilePic');
            break;
          }
        }
      })
    }
    console.log(tutorInfo);
    return res.status(200).json({ tutorInfo });

  } catch (err) {
    console.log(err);
    return res.status(500).send("Internal Error");
  }

});
```

Figure 6.10: Section Code (Server API) – Retrieve Popular Tutor Part 2

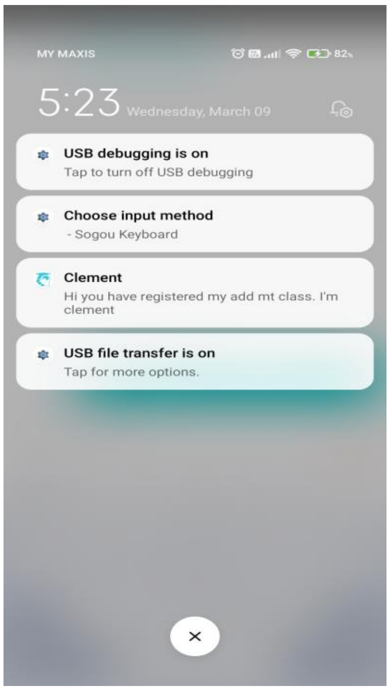
For popular tutor displaying, the "/api/retrievePopularTutor/:category" API is called every didFocus is invoked (focus on the home screen). Each invokes calls the API with ‘primary’ and ‘secondary’ category values. The data is retrieved from ‘formalClassRgst’ collection according to the school category which is the teaching syllabus in the collection. Then, applying the reduce method to count the number of the registered classes of the tutor in the collection to identify the popular tutor (the higher count indicates the more popular). The server returns the information of the top 5 tutors from the highest registered class count to the lowest count.



```
async _asyncSaveProfile(userInfo) {
  try {
    let userId = ['userId', userInfo.userId];
    let contactNo = ['contactNo', userInfo.contactNo];
    let email = ['email', userInfo.email];
    let profilePic = ['profilePic', userInfo.profilePic];
    let gender = ['gender', userInfo.gender];
    let name = ['name', userInfo.name];
    let role = ['role', userInfo.role];
    await AsyncStorage.multiSet([userId, contactNo, email, profilePic, gender, name, role]);
  } catch (error) {
    console.log('ERROR SAVING ITEM : ', error);
  }
}
```

Figure 6.11: Section Code (Front-end) – Home AsynStorage Activity

For AsyncStorage handling, the user’s information that is passed from the sign-in screen is stored by using the AsyncStorage feature in React-Native, the Async data is retrieved in different screens to perform their functions which are explained in the following section.



MY MAXIS 82%

5:23 Wednesday, March 09

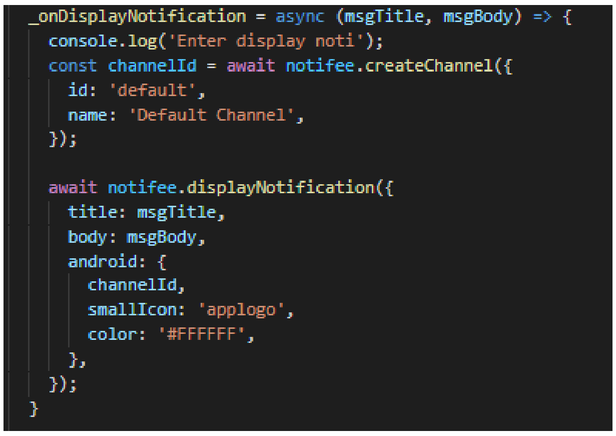
USB debugging is on
Tap to turn off USB debugging

Choose input method
- Sogou Keyboard

Clement
Hi you have registered my add mt class. I'm clement

USB file transfer is on
Tap for more options.

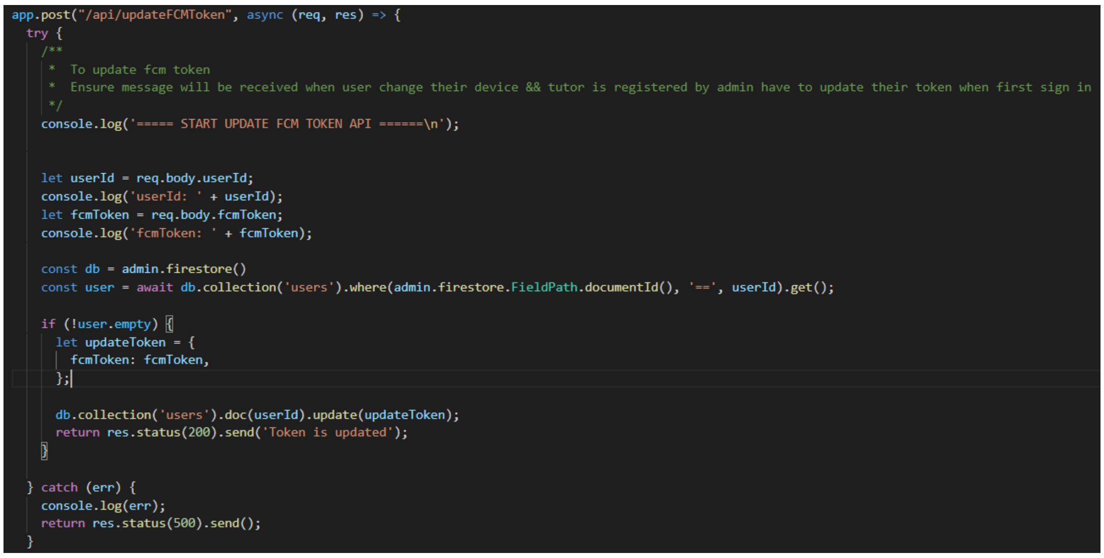
Figure 6.12: Push Notification



```
_onDisplayNotification = async (msgTitle, msgBody) => {
  console.log('Enter display noti');
  const channelId = await notifee.createChannel({
    id: 'default',
    name: 'Default Channel',
  });

  await notifee.displayNotification({
    title: msgTitle,
    body: msgBody,
    android: {
      channelId,
      smallIcon: 'applogo',
      color: '#FFFFFF',
    },
  });
}
```

Figure 6.13: Section Code (Front-end) – Home Push Notification Activity

For push notification handling, the external library called Notifee is applied to display the push notification on the phone when the FCM received a message from the other user by using the onMessage function.



```
app.post("/api/updateFCMToken", async (req, res) => {
  try {
    /**
     *  To update fcm token
     *  Ensure message will be received when user change their device && tutor is registered by admin have to update their token when first sign in
     */
    console.log('===== START UPDATE FCM TOKEN API ======\n');


    let userId = req.body.userId;
    console.log('userId: ' + userId);
    let fcmToken = req.body.fcmToken;
    console.log('fcmToken: ' + fcmToken);

    const db = admin.firestore()
    const user = await db.collection('users').where(admin.firestore.FieldPath.documentId(), '==', userId).get();

    if (!user.empty) {
      let updateToken = {
        fcmToken: fcmToken,
      };

      db.collection('users').doc(userId).update(updateToken);
      return res.status(200).send('Token is updated');
    }

  } catch (err) {
    console.log(err);
    return res.status(500).send();
  }
```

Figure 6.14: Section Code (Server API) – Update FCM Token

For updating the FCM token, the "/api/updateFCMToken" API is called to update the fcmToken value in the ‘users’ document when the onTokenRefresh function detects there is an update on the FCM token (FCM token will be updated when the application is restored on a new device, the application is uninstalled and reinstalled, and the application’s data is cleared). Besides, this API enables registered tutors could receive their message on their device after they first sign in to the application with their device due to their account being created by the admin after the information has been verified.

6.3.3.2 History Activities

The main focus of history stack is to display demo class requests and formal class booking that is raised by the registered tutor seeker to check the requested demo classes' status and the requested classes' information. Besides, history stack also handles the other related activities such as rate and review activity.

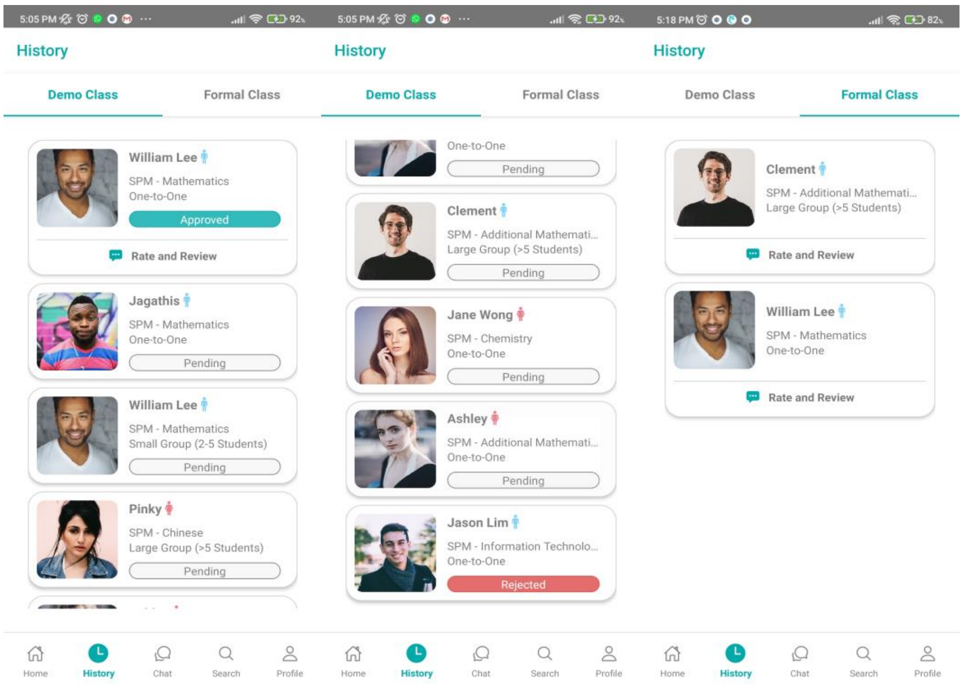
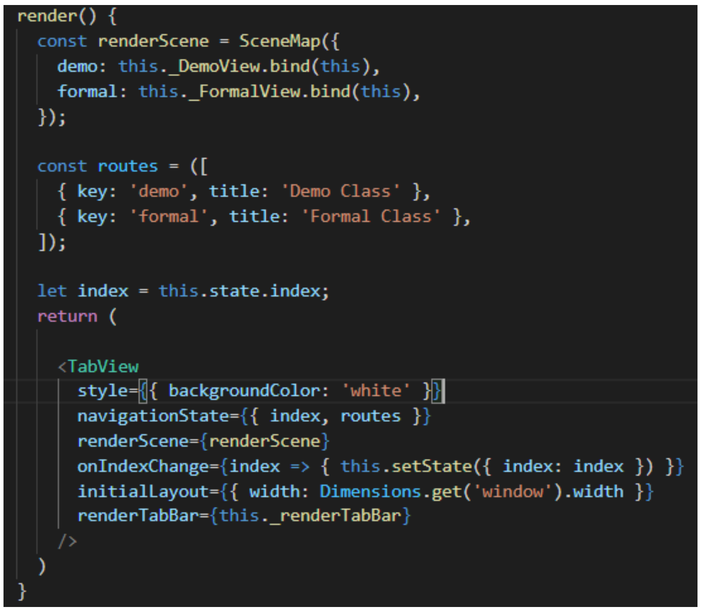


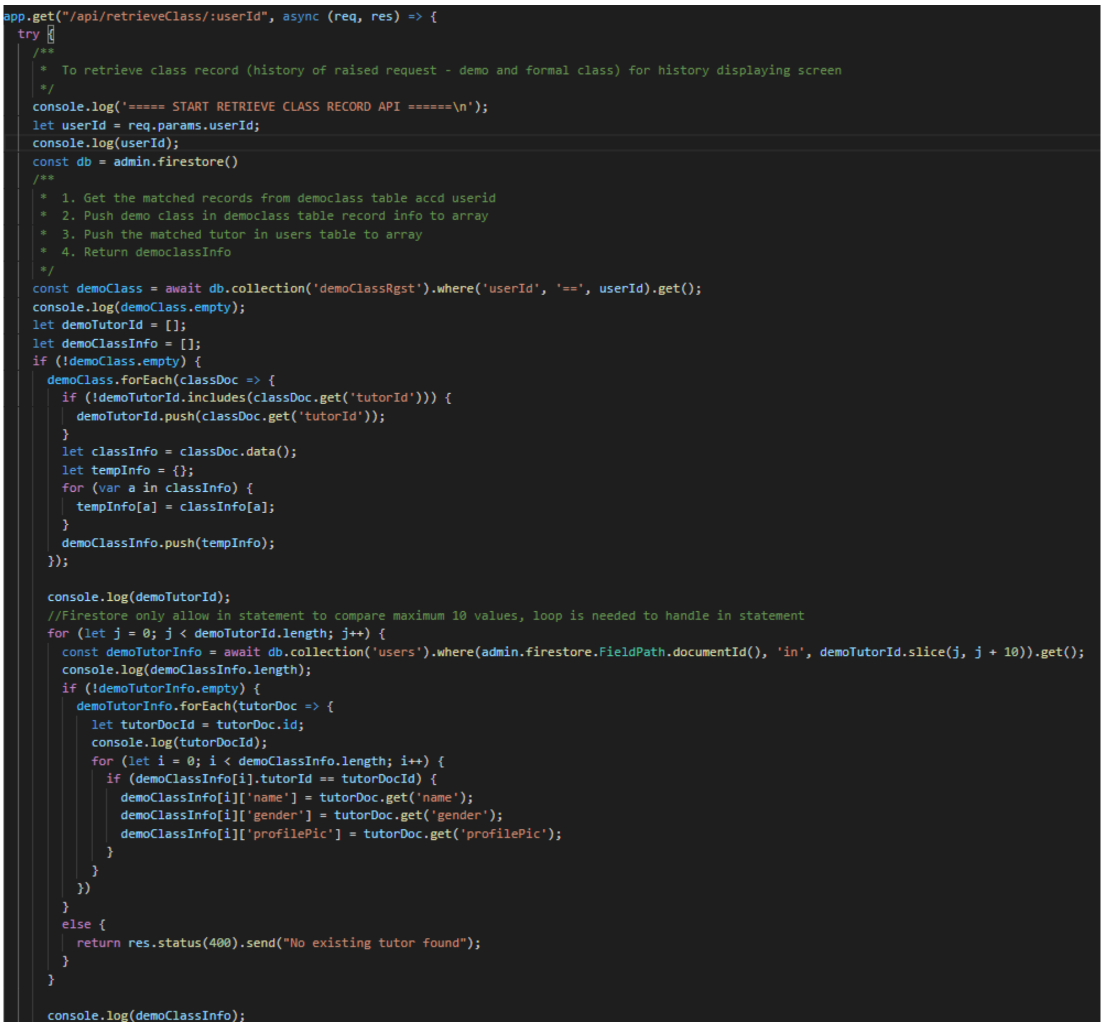
Figure 6.15: History (Demo Class and Formal Class)



```
render() {
  const renderScene = SceneMap({
    demo: this._DemoView.bind(this),
    formal: this._FormalView.bind(this),
  });

  const routes = ([
    { key: 'demo', title: 'Demo Class' },
    { key: 'formal', title: 'Formal Class' },
  ]);

  let index = this.state.index;
  return (

    <TabView
      style={{ backgroundColor: 'white' }}
      navigationState={{ index, routes }}
      renderScene={renderScene}
      onIndexChange={index => { this.setState({ index: index }) }}
      initialLayout={{ width: Dimensions.get('window').width }}
      renderTabBar={this._renderTabBar}
    />
  )
}
```

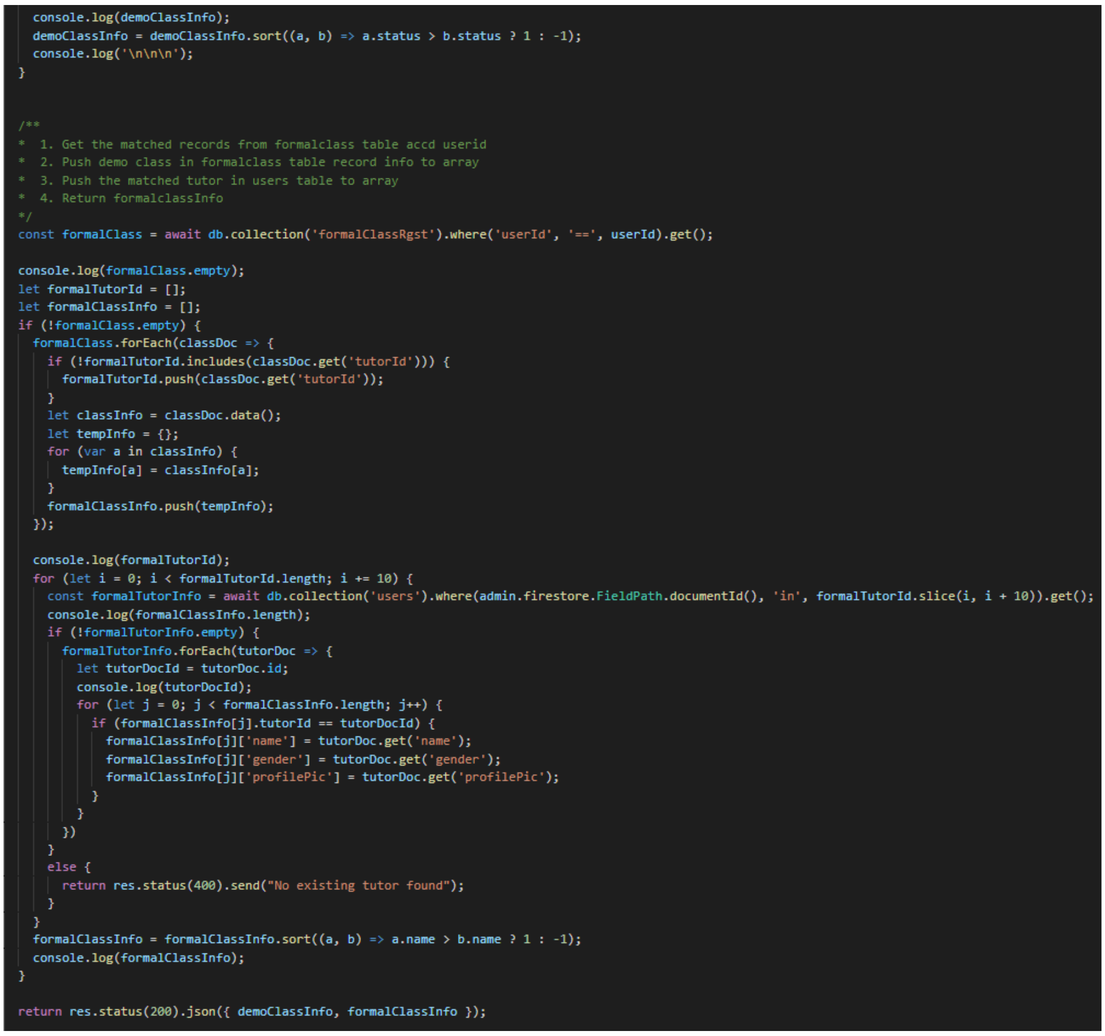
Figure 6.16: Section Code (Front-end) – History Tab View

The external library called 'react-native-tab-view' is imported to generate tab view in displaying the demo class requests and formal class requests that are raised by the registered tutor seeker in demo view tab and formal view tab respectively. Registered tutor seeker could swipe left or right to interchange the demo and formal view.



```
app.get("/api/retrieveClass/:userId", async (req, res) => {
  try {
    /**
     *  To retrieve class record (history of raised request - demo and formal class) for history displaying screen
     */
    console.log('===== START RETRIEVE CLASS RECORD API ======\n');
    let userId = req.params.userId;
    console.log(userId);
    const db = admin.firestore()
    /**
     *  1. Get the matched records from democlass table accd userid
     *  2. Push demo class in democlass table record info to array
     *  3. Push the matched tutor in users table to array
     *  4. Return democlassInfo
     */
    const demoClass = await db.collection('demoClassRgst').where('userId', '==', userId).get();
    console.log(demoClass.empty);
    let demoTutorId = [];
    let demoClassInfo = [];
    if (!demoClass.empty) {
      demoClass.forEach(classDoc => {
        if (!demoTutorId.includes(classDoc.get('tutorId'))) {
          demoTutorId.push(classDoc.get('tutorId'));
        }
        let classInfo = classDoc.data();
        let tempInfo = {};
        for (var a in classInfo) {
          tempInfo[a] = classInfo[a];
        }
        demoClassInfo.push(tempInfo);
      });

      console.log(demoTutorId);
      //Firestore only allow in statement to compare maximum 10 values, loop is needed to handle in statement
      for (let j = 0; j < demoTutorId.length; j++) {
        const demoTutorInfo = await db.collection('users').where(admin.firestore.FieldPath.documentId(), 'in', demoTutorId.slice(j, j + 10)).get();
        console.log(demoClassInfo.length);
        if (!demoTutorInfo.empty) {
          demoTutorInfo.forEach(tutorDoc => {
            let tutorDocId = tutorDoc.id;
            console.log(tutorDocId);
            for (let i = 0; i < demoClassInfo.length; i++) {
              if (demoClassInfo[i].tutorId == tutorDocId) {
                demoClassInfo[i]['name'] = tutorDoc.get('name');
                demoClassInfo[i]['gender'] = tutorDoc.get('gender');
                demoClassInfo[i]['profilePic'] = tutorDoc.get('profilePic');
              }
            }
          })
        }
        else {
          return res.status(400).send("No existing tutor found");
        }
      }

      console.log(demoClassInfo);
```

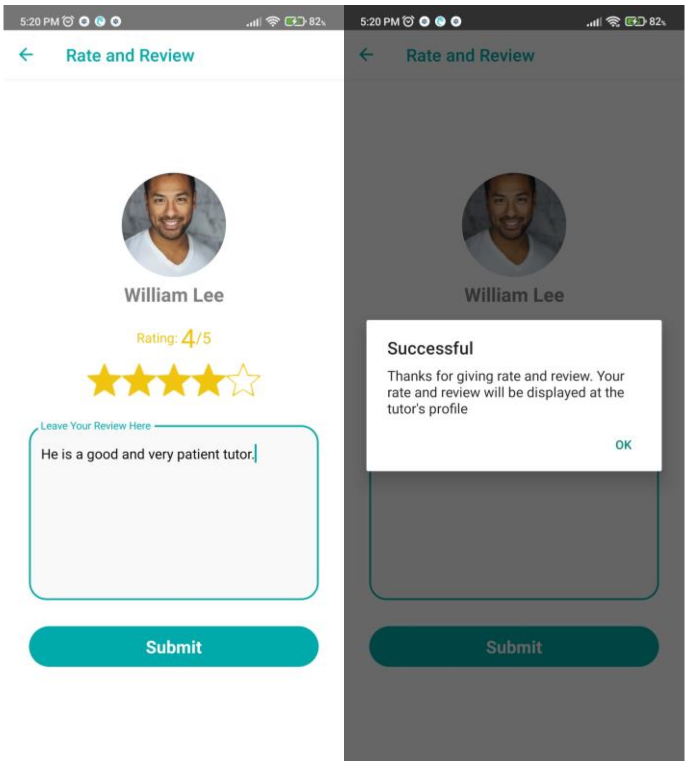
Figure 6.17: Section Code (Server API) – Retrieve Class Part 1



```
  console.log(demoClassInfo);
  demoClassInfo = demoClassInfo.sort((a, b) => a.status > b.status ? 1 : -1);
  console.log('\n\n\n');
}


/**
*  1. Get the matched records from formalclass table accd userid
*  2. Push demo class in formalclass table record info to array
*  3. Push the matched tutor in users table to array
*  4. Return formalclassInfo
*/
const formalClass = await db.collection('formalClassRgst').where('userId', '==', userId).get();

console.log(formalClass.empty);
let formalTutorId = [];
let formalClassInfo = [];
if (!formalClass.empty) {
  formalClass.forEach(classDoc => {
    if (!formalTutorId.includes(classDoc.get('tutorId'))) {
      formalTutorId.push(classDoc.get('tutorId'));
    }
    let classInfo = classDoc.data();
    let tempInfo = {};
    for (var a in classInfo) {
      tempInfo[a] = classInfo[a];
    }
    formalClassInfo.push(tempInfo);
  });

  console.log(formalTutorId);
  for (let i = 0; i < formalTutorId.length; i += 10) {
    const formalTutorInfo = await db.collection('users').where(admin.firestore.FieldPath.documentId(), 'in', formalTutorId.slice(i, i + 10)).get();
    console.log(formalClassInfo.length);
    if (!formalTutorInfo.empty) {
      formalTutorInfo.forEach(tutorDoc => {
        let tutorDocId = tutorDoc.id;
        console.log(tutorDocId);
        for (let j = 0; j < formalClassInfo.length; j++) {
          if (formalClassInfo[j].tutorId == tutorDocId) {
            formalClassInfo[j]['name'] = tutorDoc.get('name');
            formalClassInfo[j]['gender'] = tutorDoc.get('gender');
            formalClassInfo[j]['profilePic'] = tutorDoc.get('profilePic');
          }
        }
      })
    }
    else {
      return res.status(400).send("No existing tutor found");
    }
  }
  formalClassInfo = formalClassInfo.sort((a, b) => a.name > b.name ? 1 : -1);
  console.log(formalClassInfo);
}

return res.status(200).json({ demoClassInfo, formalClassInfo });
```

Figure 6.18: Section Code (Server API) – Retrieve Class Part 2

For displaying the demo and formal class requests' information, the userId is get by applying the AsynStorage and the userId is used as parameter in calling the "/api/retrieveClass/:userId" API in server. The API get data from 'demoClassRgst' and 'formalClassRgst' according to the userId parameter. Then, the basic information of the tutor of each matched class is pushed into the array while the arrays are sorted according to the status (demo class) or name (formal class). Last, the arrays are returned to the front-end in JSON format.

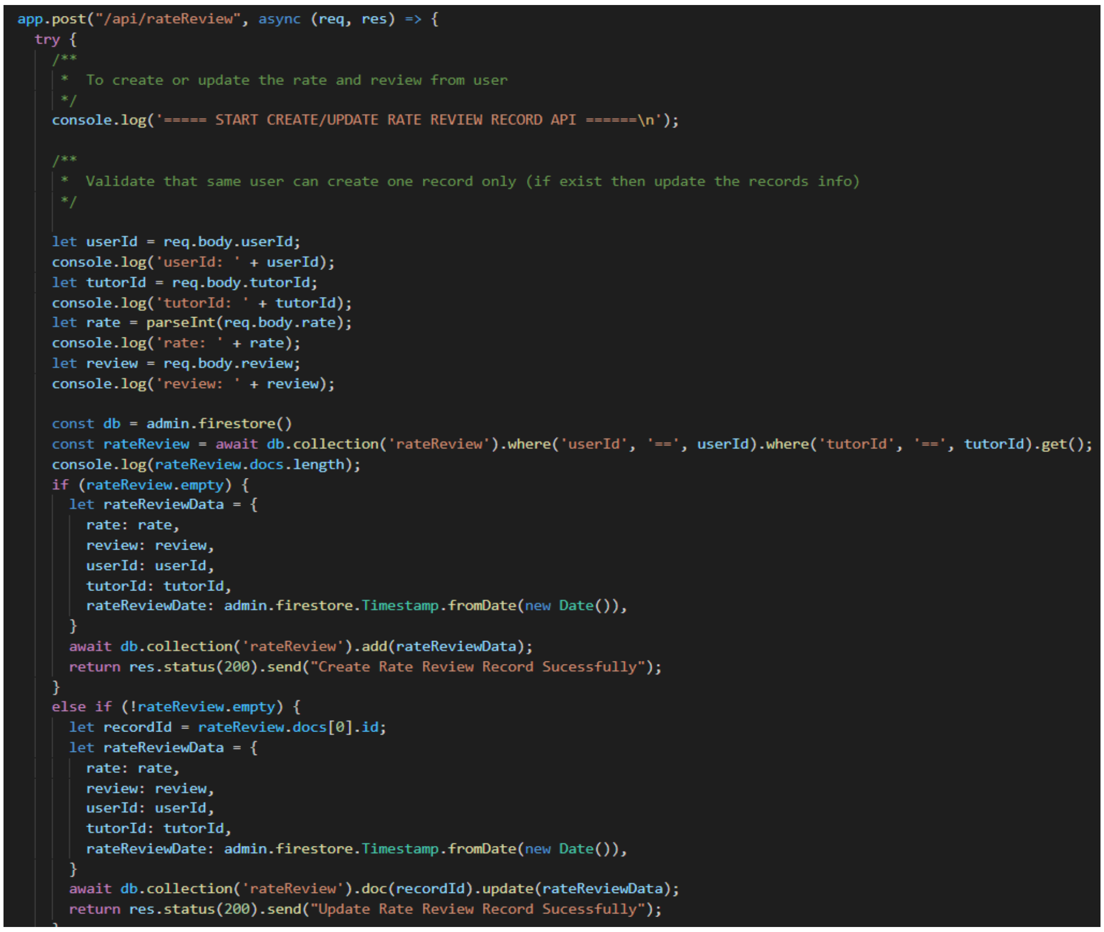


Successful

Thanks for giving rate and review. Your rate and review will be displayed at the tutor's profile

OK

Figure 6.19: History (Rate and Review)



```
app.post("/api/rateReview", async (req, res) => {
  try {
    /**
     *  To create or update the rate and review from user
     */
    console.log('===== START CREATE/UPDATE RATE REVIEW RECORD API ======\n');

    /**
     *  Validate that same user can create one record only (if exist then update the records info)
     */

    let userId = req.body.userId;
    console.log('userId: ' + userId);
    let tutorId = req.body.tutorId;
    console.log('tutorId: ' + tutorId);
    let rate = parseInt(req.body.rate);
    console.log('rate: ' + rate);
    let review = req.body.review;
    console.log('review: ' + review);

    const db = admin.firestore()
    const rateReview = await db.collection('rateReview').where('userId', '==', userId).where('tutorId', '==', tutorId).get();
    console.log(rateReview.docs.length);
    if (rateReview.empty) {
      let rateReviewData = {
        rate: rate,
        review: review,
        userId: userId,
        tutorId: tutorId,
        rateReviewDate: admin.firestore.Timestamp.fromDate(new Date()),
      }
      await db.collection('rateReview').add(rateReviewData);
      return res.status(200).send("Create Rate Review Record Sucessfully");
    }
    else if (!rateReview.empty) {
      let recordId = rateReview.docs[0].id;
      let rateReviewData = {
        rate: rate,
        review: review,
        userId: userId,
        tutorId: tutorId,
        rateReviewDate: admin.firestore.Timestamp.fromDate(new Date()),
      }
      await db.collection('rateReview').doc(recordId).update(rateReviewData);
      return res.status(200).send("Update Rate Review Record Sucessfully");
```

Figure 6.20: Section Code (Server API) – Rate Review

The rate and review activity only can be performed by the registered tutor seeker to the registered tutor after the formal class is raised (assumed the formal class is conducted) or the demo class request is approved (assumed the demo class is conducted). The profile picture and name of the tutor are passed to the rate and review screen while the "/api/rateReview" API is triggered when the registered tutor seeker presses the “Submit” button. The API checks the existence of the same record (same userId and tutorId) in ‘rateReview’ collection. Then, creating a new document if no same record is found or updating the existing document if the same record is found.

6.3.3.3 Chat Activities

The significant role of chat stack is to handle the chat between the registered tutor seeker and registered tutor. Push notification is displayed when receive a new message from the other users (it is explained in home activities). The chat activities can be conducted by accessing the “chat with tutor” button on the tutor profile screen

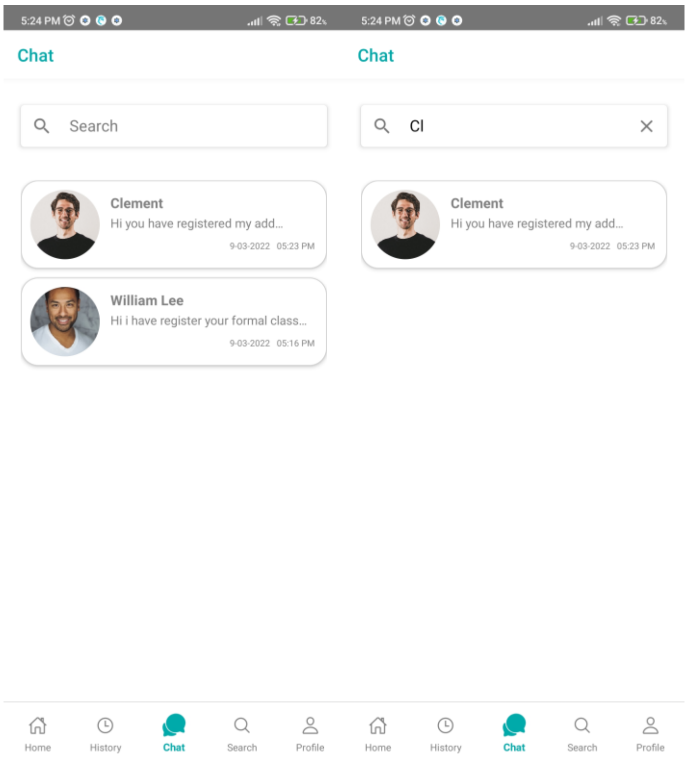
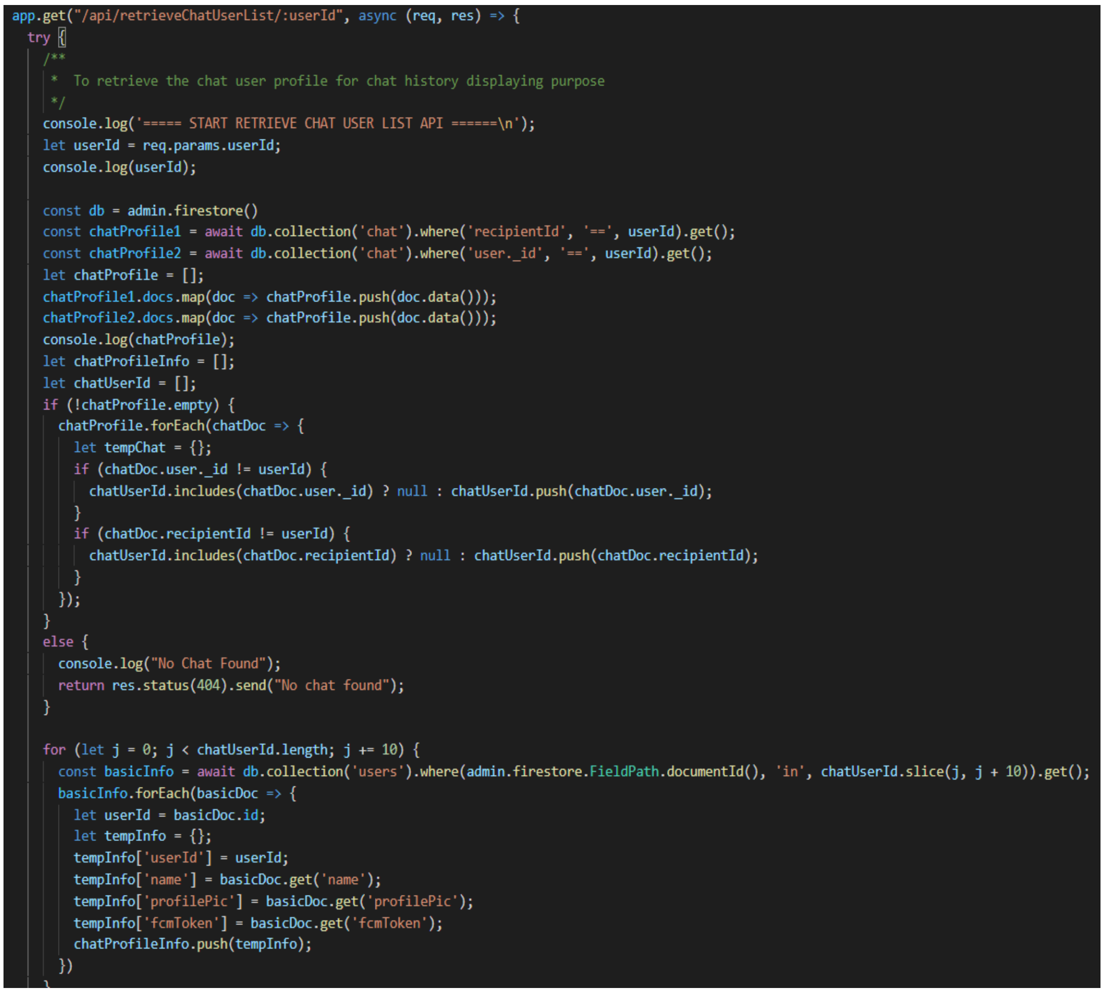


Figure 6.21: Chat (Chat History List)

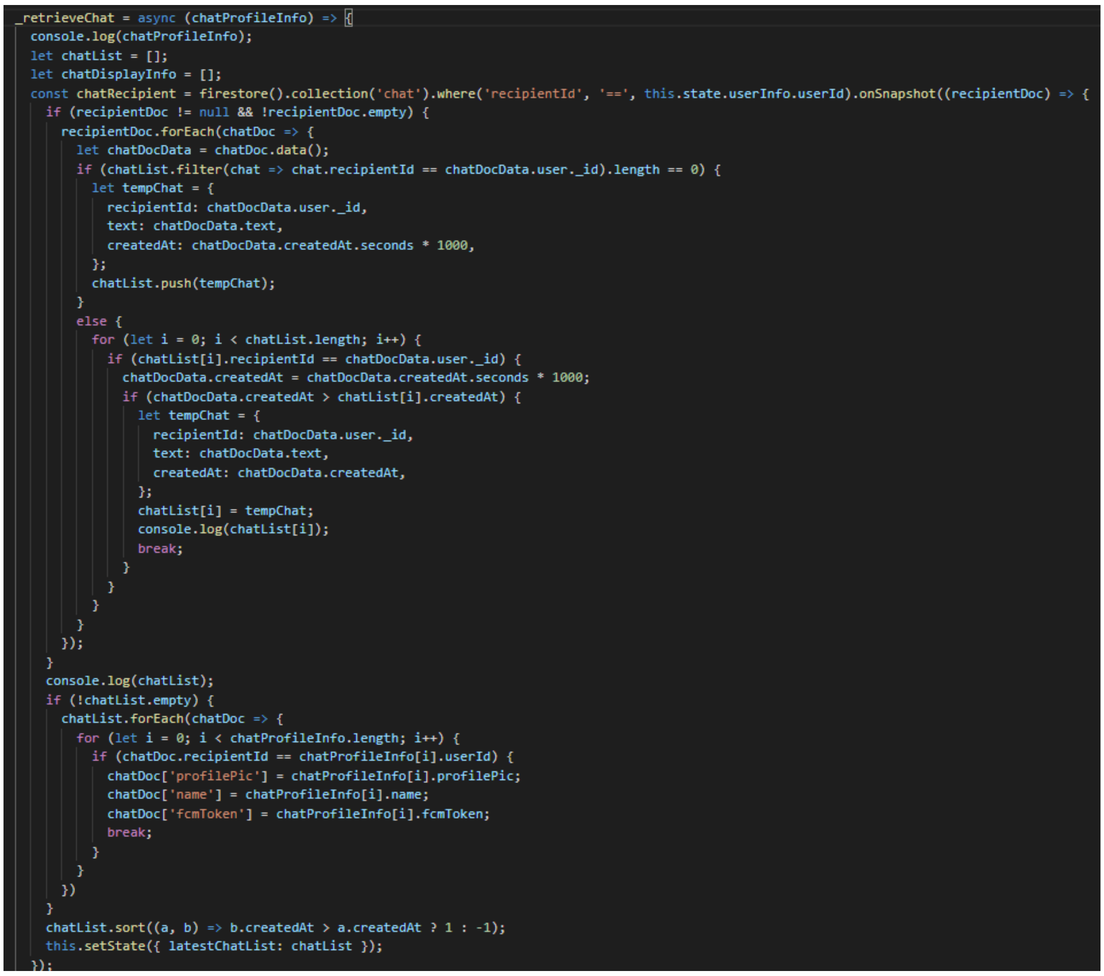
Every time didFocus is triggered invoked the readProfile function which is AsyncStorage in getting the user’s information. The "/api/retrieveChatUserList/:userId" API is called directly after getting the user’s information from AsyncStorage while the response from API also be returned as a parameter to call retrieveChat function at front-end. The retrieveChat is a function that accesses the ‘chat’ collection in getting chat data. The concerns of direct linking front-end to the database is explained in CHAPTER 5.



```
app.get("/api/retrieveChatUserList/:userId", async (req, res) => {
  try {
    /**
     *  To retrieve the chat user profile for chat history displaying purpose
     */
    console.log('===== START RETRIEVE CHAT USER LIST API ======\n');
    let userId = req.params.userId;
    console.log(userId);

    const db = admin.firestore()
    const chatProfile1 = await db.collection('chat').where('recipientId', '==', userId).get();
    const chatProfile2 = await db.collection('chat').where('user._id', '==', userId).get();
    let chatProfile = [];
    chatProfile1.docs.map(doc => chatProfile.push(doc.data()));
    chatProfile2.docs.map(doc => chatProfile.push(doc.data()));
    console.log(chatProfile);
    let chatProfileInfo = [];
    let chatUserId = [];
    if (!chatProfile.empty) {
      chatProfile.forEach(chatDoc => {
        let tempChat = {};
        if (chatDoc.user._id != userId) {
          chatUserId.includes(chatDoc.user._id) ? null : chatUserId.push(chatDoc.user._id);
        }
        if (chatDoc.recipientId != userId) {
          chatUserId.includes(chatDoc.recipientId) ? null : chatUserId.push(chatDoc.recipientId);
        }
      });
    }
    else {
      console.log("No Chat Found");
      return res.status(404).send("No chat found");
    }

    for (let j = 0; j < chatUserId.length; j += 10) {
      const basicInfo = await db.collection('users').where(admin.firestore.FieldPath.documentId(), 'in', chatUserId.slice(j, j + 10)).get();
      basicInfo.forEach(basicDoc => {
        let userId = basicDoc.id;
        let tempInfo = {};
        tempInfo['userId'] = userId;
        tempInfo['name'] = basicDoc.get('name');
        tempInfo['profilePic'] = basicDoc.get('profilePic');
        tempInfo['fcmToken'] = basicDoc.get('fcmToken');
        chatProfileInfo.push(tempInfo);
      })
```

Figure 6.22: Section Code (Server API) – Retrieve Chat User List

The API retrieve related chat document from database according to the userId and push the sender or recipient id (except id that is same with userId) into an array to retrieve the basic information in ‘users’ collection and return to the front-end.



```
_retrieveChat = async (chatProfileInfo) => {
  console.log(chatProfileInfo);
  let chatList = [];
  let chatDisplayInfo = [];
  const chatRecipient = firestore().collection('chat').where('recipientId', '==', this.state.userInfo.userId).onSnapshot((recipientDoc) => {
    if (recipientDoc != null && !recipientDoc.empty) {
      recipientDoc.forEach(chatDoc => {
        let chatDocData = chatDoc.data();
        if (chatList.filter(chat => chat.recipientId == chatDocData.user._id).length == 0) {
          let tempChat = {
            recipientId: chatDocData.user._id,
            text: chatDocData.text,
            createdAt: chatDocData.createdAt.seconds * 1000,
          };
          chatList.push(tempChat);
        }
        else {
          for (let i = 0; i < chatList.length; i++) {
            if (chatList[i].recipientId == chatDocData.user._id) {
              chatDocData.createdAt = chatDocData.createdAt.seconds * 1000;
              if (chatDocData.createdAt > chatList[i].createdAt) {
                let tempChat = {
                  recipientId: chatDocData.user._id,
                  text: chatDocData.text,
                  createdAt: chatDocData.createdAt,
                };
                chatList[i] = tempChat;
                console.log(chatList[i]);
                break;
              }
            }
          }
        }
      });
    }
    console.log(chatList);
    if (!chatList.empty) {
      chatList.forEach(chatDoc => {
        for (let i = 0; i < chatProfileInfo.length; i++) {
          if (chatDoc.recipientId == chatProfileInfo[i].userId) {
            chatDoc['profilePic'] = chatProfileInfo[i].profilePic;
            chatDoc['name'] = chatProfileInfo[i].name;
            chatDoc['fcmToken'] = chatProfileInfo[i].fcmToken;
            break;
          }
        }
      })
    }
    chatList.sort((a, b) => b.createdAt > a.createdAt ? 1 : -1);
    this.setState({ latestChatList: chatList });
  });
```

Figure 6.23: Section Code (Front-end) – Retrieve Chat List Activity

The basic information from the server is a parameter in the retrieve chat function at the front-end. In figure 6.23, the onSnapshot function in Firestore enables the application to keep listening to the ‘chat’ collection in querying the latest chat document. Then, push the latest chat history into the chatList. In figure 6.28, the matched user’s profile information is inserted into the respective object in chatList. Last, sort the chatList with creating time from the latest to the oldest. The same implementation is applied in retrieving the chat history when the user is in a sender role. All latest chat data is pushed into chatList and the chat list is displayed from the latest to oldest in the chat history screen. The user is directed to the respective chatbox screen when the user press the respective chat history column.

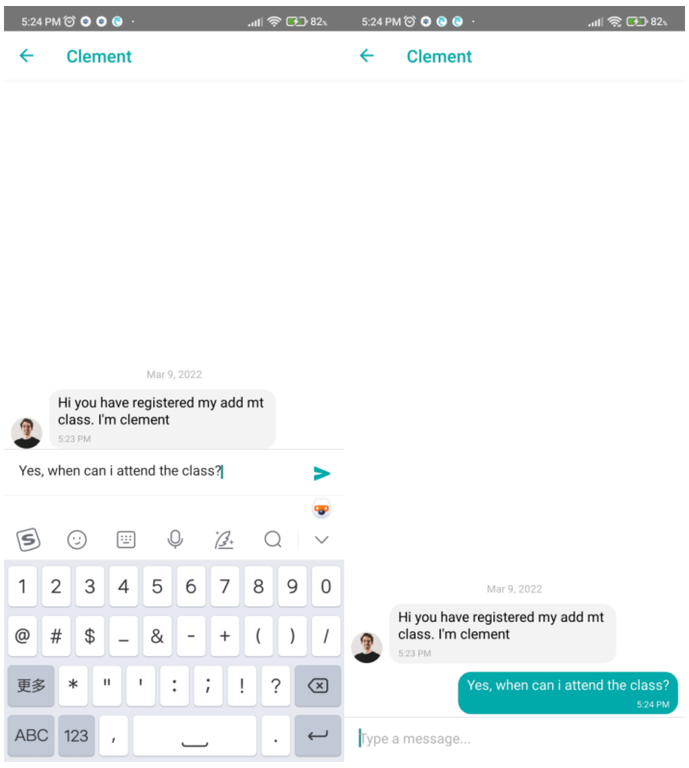
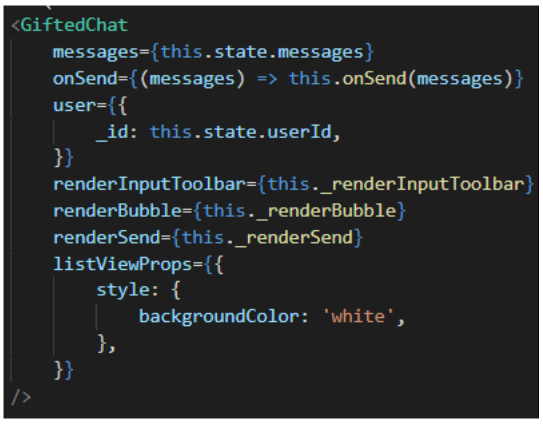
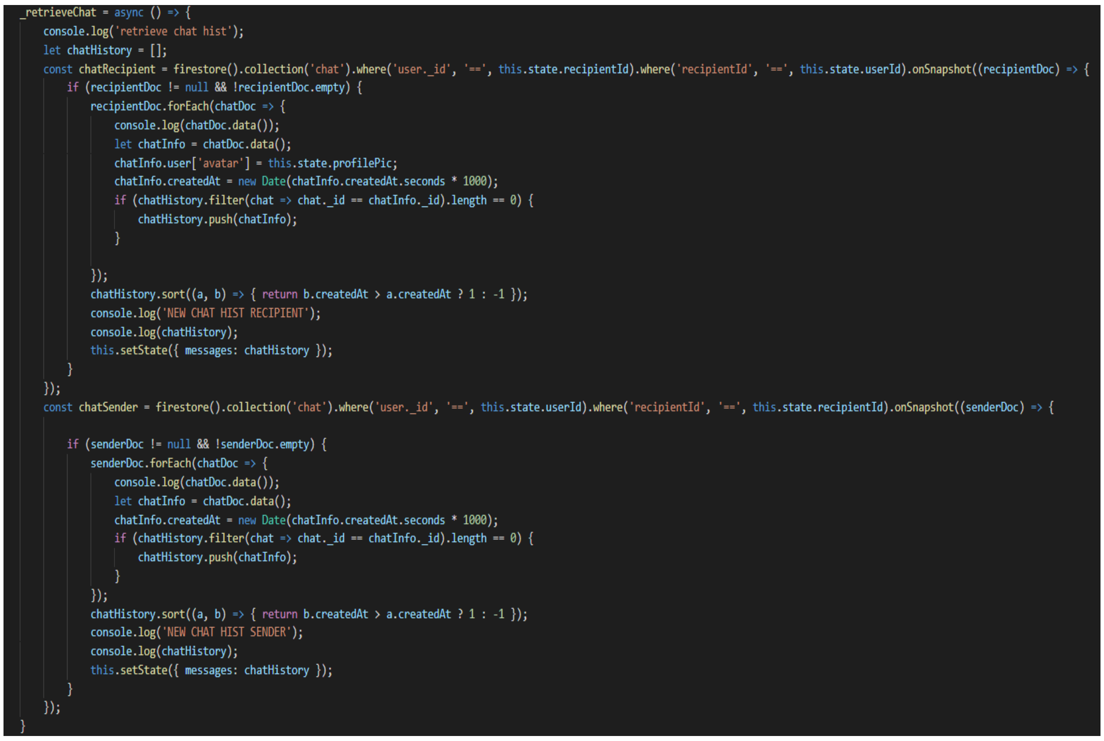

5:24 PM
Clement
Mar 9, 2022
Hi you have registered my add mt class. I'm clement
5:23 PM
Yes, when can i attend the class?
5:24 PM
Type a message...

Figure 6.24: Chat (Chat Box)



```
<GiftedChat
    messages={this.state.messages}
    onSend={(messages) => this.onSend(messages)}
    user={{
        _id: this.state.userId,
    }}
    renderInputToolbar={this._renderInputToolbar}
    renderBubble={this._renderBubble}
    renderSend={this._renderSend}
    listViewProps={{
        style: {
            backgroundColor: 'white',
        },
    }}
/>
```

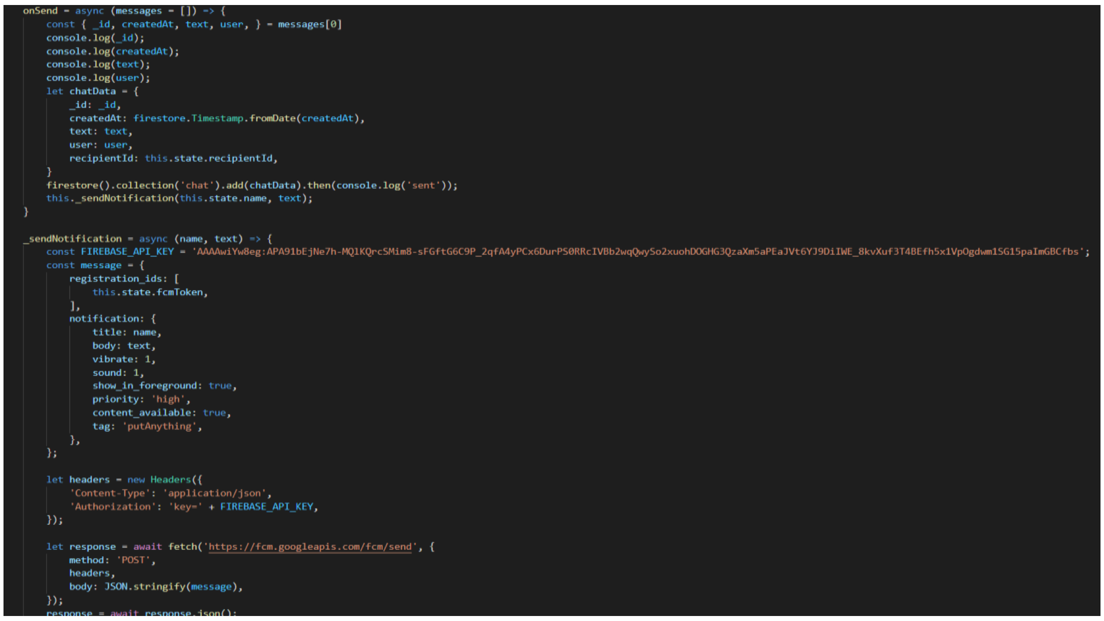
Figure 6.25: Section Code (Front-end) – Chat Box Activities Gifted Chat



```
_retrieveChat = async () => {
    console.log('retrieve chat hist');
    let chatHistory = [];
    const chatRecipient = firestore().collection('chat').where('user._id', '==', this.state.recipientId).where('recipientId', '==', this.state.userId).onSnapshot((recipientDoc) => {
        if (recipientDoc != null && !recipientDoc.empty) {
            recipientDoc.forEach(chatDoc => {
                console.log(chatDoc.data());
                let chatInfo = chatDoc.data();
                chatInfo.user['avatar'] = this.state.profilePic;
                chatInfo.createdAt = new Date(chatInfo.createdAt.seconds * 1000);
                if (chatHistory.filter(chat => chat._id == chatInfo._id).length == 0) {
                    chatHistory.push(chatInfo);
                }

            });
            chatHistory.sort((a, b) => { return b.createdAt > a.createdAt ? 1 : -1 });
            console.log('NEW CHAT HIST RECIPIENT');
            console.log(chatHistory);
            this.setState({ messages: chatHistory });
        }
    });
    const chatSender = firestore().collection('chat').where('user._id', '==', this.state.userId).where('recipientId', '==', this.state.recipientId).onSnapshot((senderDoc) => {

        if (senderDoc != null && !senderDoc.empty) {
            senderDoc.forEach(chatDoc => {
                console.log(chatDoc.data());
                let chatInfo = chatDoc.data();
                chatInfo.createdAt = new Date(chatInfo.createdAt.seconds * 1000);
                if (chatHistory.filter(chat => chat._id == chatInfo._id).length == 0) {
                    chatHistory.push(chatInfo);
                }
            });
            chatHistory.sort((a, b) => { return b.createdAt > a.createdAt ? 1 : -1 });
            console.log('NEW CHAT HIST SENDER');
            console.log(chatHistory);
            this.setState({ messages: chatHistory });
        }
    });
}
```

Figure 6.26: Section Code (Front-end) – Retrieve Chat Activity

The AsyncStorage and retrieve chat function when entering the chatbox screen. The retrieve chat function is similar to the function applied in the chat history screen to retrieve the chat text in the ‘chat’ collection and display it according to the created time. The difference between the function applied in the chat history screen and chatbox screen is only retrieving the chat between the user with the respective tutor in the chatbox screen instead of all tutors in the chat history screen.



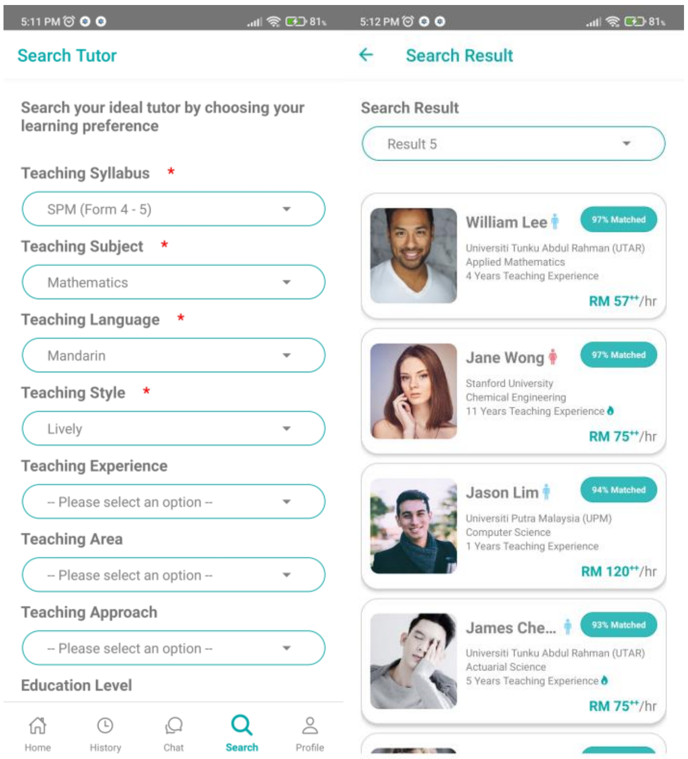
```
onSend = async (messages = []) => {
    const { _id, createdAt, text, user, } = messages[0]
    console.log(_id);
    console.log(createdAt);
    console.log(text);
    console.log(user);
    let chatData = {
        _id: _id,
        createdAt: firestore.Timestamp.fromDate(createdAt),
        text: text,
        user: user,
        recipientId: this.state.recipientId,
    }
    firestore().collection('chat').add(chatData).then(console.log('sent'));
    this._sendNotification(this.state.name, text);
}

_sendNotification = async (name, text) => {
    const FIREBASE_API_KEY = 'AAAAwiYw8eg:APA91bEjNe7h-MQ1KQrcSMim8-sFGftG6C9P_2qfA4yPCx6DurPS0RRcIVBb2wqQwySo2xuohDOGHG3QzaXm5aPEaJVt6YJ9DiIWE_8kvXuf3T4BEfh5x1VpOgdwm1SG15paImGBCfbs';
    const message = {
        registration_ids: [
            this.state.fcmToken,
        ],
        notification: {
            title: name,
            body: text,
            vibrate: 1,
            sound: 1,
            show_in_foreground: true,
            priority: 'high',
            content_available: true,
            tag: 'putAnything',
        },
    };

    let headers = new Headers({
        'Content-Type': 'application/json',
        'Authorization': 'key=' + FIREBASE_API_KEY,
    });

    let response = await fetch('https://fcm.googleapis.com/fcm/send', {
        method: 'POST',
        headers,
        body: JSON.stringify(message),
    });
    response = await response.json();
```

Figure 6.27: Section Code (Front-end) – Send Notification Activity

For sending a message, an external library called 'react-native-gifted-chat' is imported to handle the user interface design of the chatbox while the onSend function is triggered to create a chat document into the database when the user press the send icon. Then, the sendNotification function is invoked to trigger the FCM in sending a message and push notification on the recipient’s device. The FCM will send the message to the recipient and a push notification will be popped out according to the FCM token that is passed from the chat history screen (passed from the server).

6.3.3.4 Search Activities

The main focus of this application is to help the tutor seekers in finding their ideal tutors which is the main role of the search stack. The search stack is a nested stack (shown in CHAPTER 5). Search stack handles the search activity with the application of similarity measures, view tutor’s profile activity, raising demo class activity, and book formal class activity.



Search Tutor

Search your ideal tutor by choosing your learning preference

Teaching Syllabus * SPM (Form 4 - 5)
Teaching Subject * Mathematics
Teaching Language * Mandarin
Teaching Style * Lively
Teaching Experience -- Please select an option --
Teaching Area -- Please select an option --
Teaching Approach -- Please select an option --
Education Level

Search Result

Search Result: Result 5

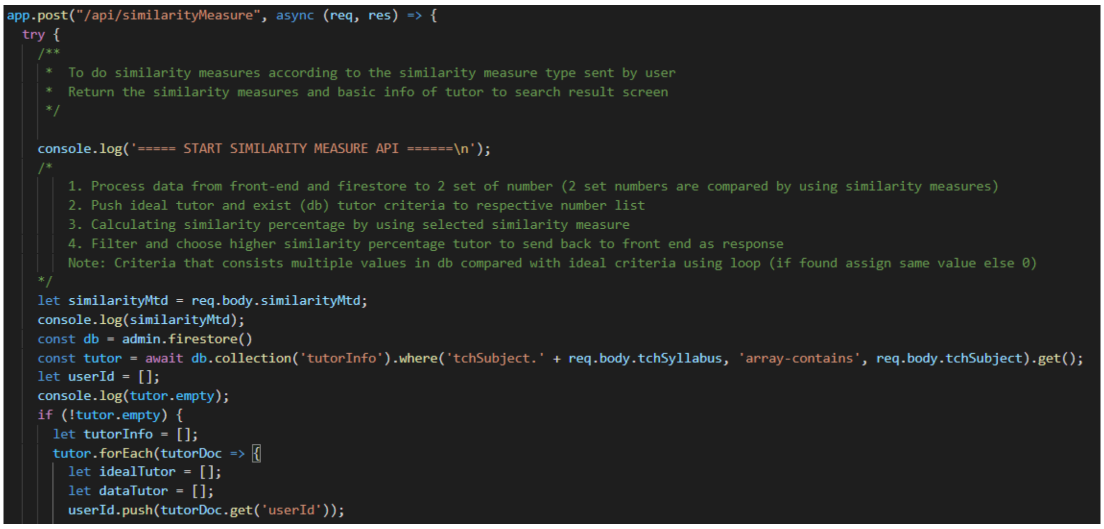
William Lee — 97% Matched
Universiti Tunku Abdul Rahman (UTAR)
Applied Mathematics
4 Years Teaching Experience
RM 57⁺⁺/hr

Jane Wong — 97% Matched
Stanford University
Chemical Engineering
11 Years Teaching Experience
RM 75⁺⁺/hr

Jason Lim — 94% Matched
Universiti Putra Malaysia (UPM)
Computer Science
1 Years Teaching Experience
RM 120⁺⁺/hr

James Che... — 93% Matched
Universiti Tunku Abdul Rahman (UTAR)
Actuarial Science
5 Years Teaching Experience
RM 75⁺⁺/hr

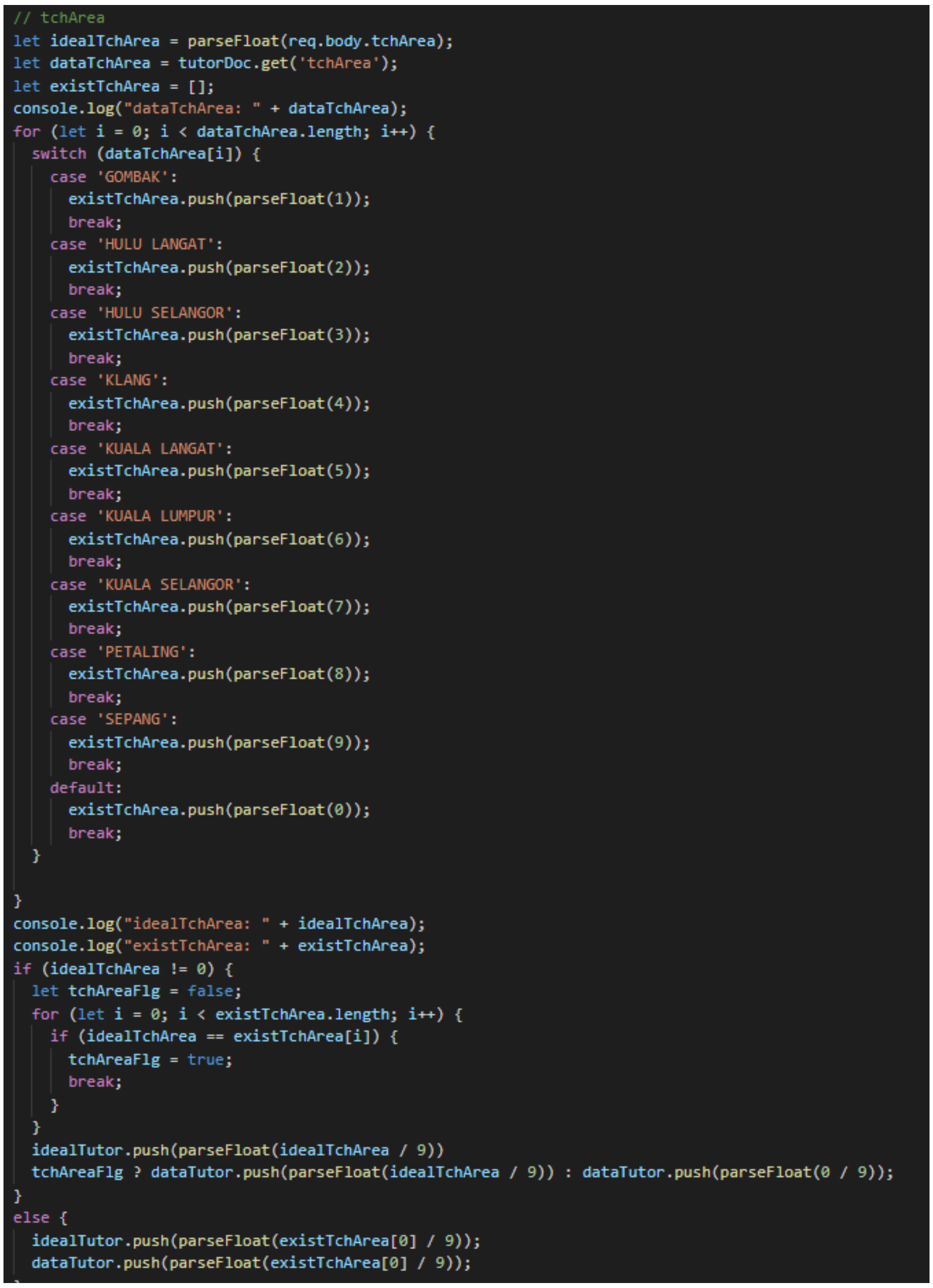
Figure 6.28: Search (Input and Result)



```
app.post("/api/similarityMeasure", async (req, res) => {
  try {
    /**
     *  To do similarity measures according to the similarity measure type sent by user
     *  Return the similarity measures and basic info of tutor to search result screen
     */

    console.log('===== START SIMILARITY MEASURE API ======\n');
    /*
        1. Process data from front-end and firestore to 2 set of number (2 set numbers are compared by using similarity measures)
        2. Push ideal tutor and exist (db) tutor criteria to respective number list
        3. Calculating similarity percentage by using selected similarity measure
        4. Filter and choose higher similarity percentage tutor to send back to front end as response
        Note: Criteria that consists multiple values in db compared with ideal criteria using loop (if found assign same value else 0)
    */
    let similarityMtd = req.body.similarityMtd;
    console.log(similarityMtd);
    const db = admin.firestore()
    const tutor = await db.collection('tutorInfo').where('tchSubject.' + req.body.tchSyllabus, 'array-contains', req.body.tchSubject).get();
    let userId = [];
    console.log(tutor.empty);
    if (!tutor.empty) {
      let tutorInfo = [];
      tutor.forEach(tutorDoc => {
        let idealTutor = [];
        let dataTutor = [];
        userId.push(tutorDoc.get('userId'));
```

Figure 6.29: Section Code (Server API) – Similarity Measure Part 1

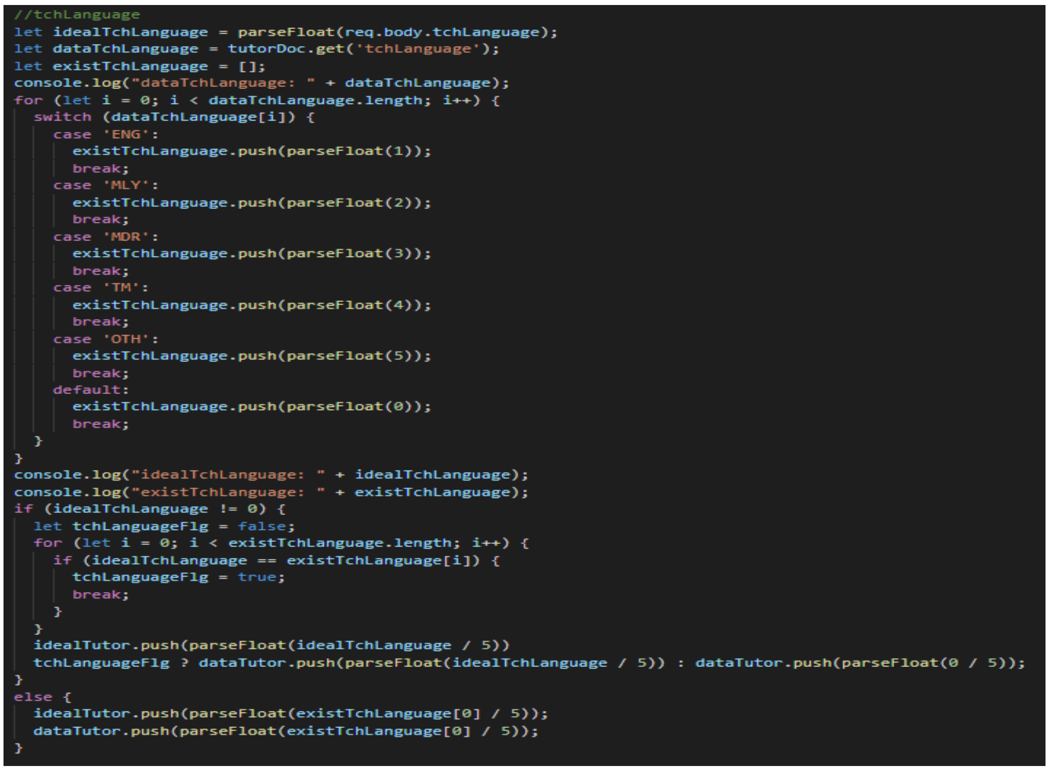
The "/api/similarityMeasure" API is called after the registered seeker press the “Search” button and navigates to the search result screen. The flow overview of this API is shown in figure 6.37. First, process the data from the front-end and database by converting the criteria into 2 set arrays of integers. Then, calculating the similarity percentages between each tutor in the database that fulfilled the minimum requirement (matched syllabus and subject) with the ideal tutor of the registered tutor seeker. Last, filter and choose the top 20 tutor with the highest similarity percentage as result in responding to the front-end.



```
// tchArea
let idealTchArea = parseFloat(req.body.tchArea);
let dataTchArea = tutorDoc.get('tchArea');
let existTchArea = [];
console.log("dataTchArea: " + dataTchArea);
for (let i = 0; i < dataTchArea.length; i++) {
  switch (dataTchArea[i]) {
    case 'GOMBAK':
      existTchArea.push(parseFloat(1));
      break;
    case 'HULU LANGAT':
      existTchArea.push(parseFloat(2));
      break;
    case 'HULU SELANGOR':
      existTchArea.push(parseFloat(3));
      break;
    case 'KLANG':
      existTchArea.push(parseFloat(4));
      break;
    case 'KUALA LANGAT':
      existTchArea.push(parseFloat(5));
      break;
    case 'KUALA LUMPUR':
      existTchArea.push(parseFloat(6));
      break;
    case 'KUALA SELANGOR':
      existTchArea.push(parseFloat(7));
      break;
    case 'PETALING':
      existTchArea.push(parseFloat(8));
      break;
    case 'SEPANG':
      existTchArea.push(parseFloat(9));
      break;
    default:
      existTchArea.push(parseFloat(0));
      break;
  }

}
console.log("idealTchArea: " + idealTchArea);
console.log("existTchArea: " + existTchArea);
if (idealTchArea != 0) {
  let tchAreaFlg = false;
  for (let i = 0; i < existTchArea.length; i++) {
    if (idealTchArea == existTchArea[i]) {
      tchAreaFlg = true;
      break;
    }
  }
  idealTutor.push(parseFloat(idealTchArea / 9))
  tchAreaFlg ? dataTutor.push(parseFloat(idealTchArea / 9)) : dataTutor.push(parseFloat(0 / 9));
}
else {
  idealTutor.push(parseFloat(existTchArea[0] / 9));
  dataTutor.push(parseFloat(existTchArea[0] / 9));
```

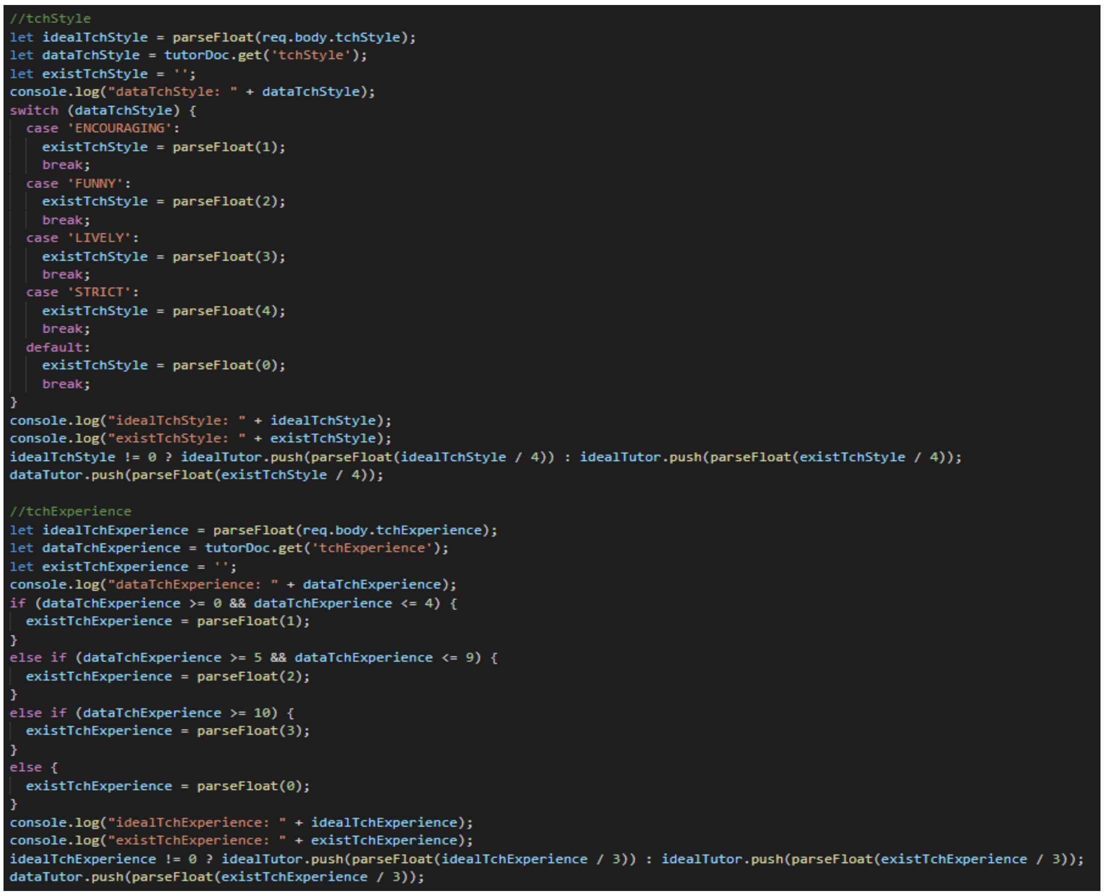
Figure 6.30: Section Code (Server API) – Similarity Measure Part 2

Figure 6.38 to figure 6.44 show the flow in processing the data from Firestore into integers and insert the integers into the dataTutor array while processing the data from front-end and inserting the data into the idealTutor array. Different fields in the ‘tutorInfo’ document are processed in different methods.



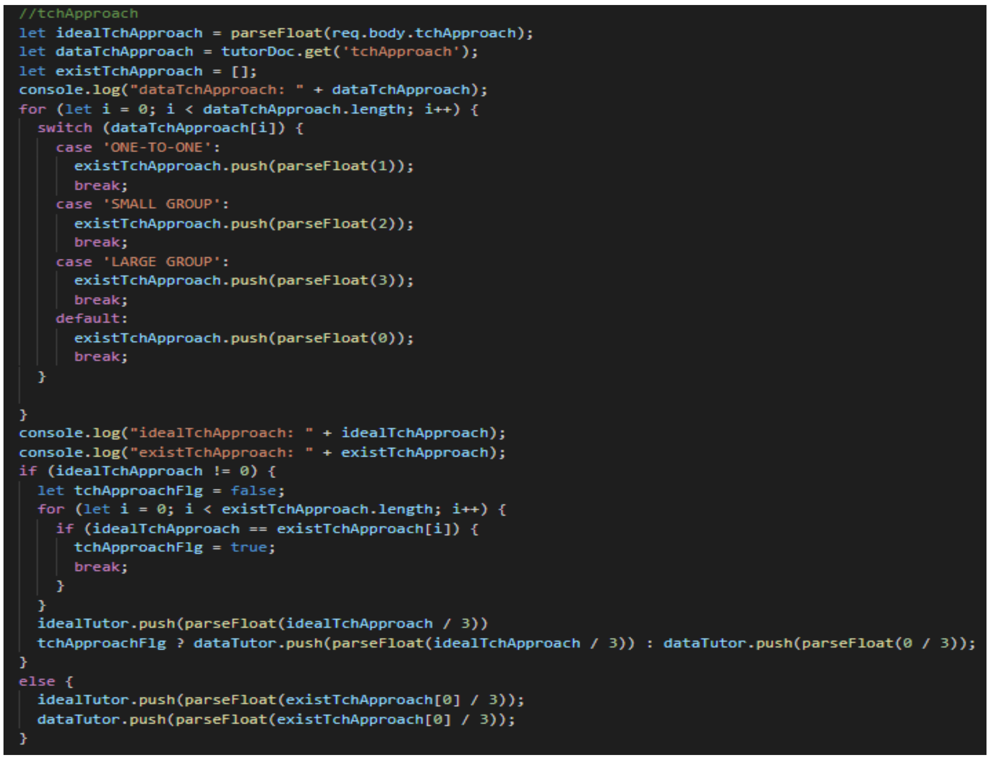
```
//tchLanguage
let idealTchLanguage = parseFloat(req.body.tchLanguage);
let dataTchLanguage = tutorDoc.get('tchLanguage');
let existTchLanguage = [];
console.log("dataTchLanguage: " + dataTchLanguage);
for (let i = 0; i < dataTchLanguage.length; i++) {
  switch (dataTchLanguage[i]) {
    case 'ENG':
      existTchLanguage.push(parseFloat(1));
      break;
    case 'MLY':
      existTchLanguage.push(parseFloat(2));
      break;
    case 'MDR':
      existTchLanguage.push(parseFloat(3));
      break;
    case 'TM':
      existTchLanguage.push(parseFloat(4));
      break;
    case 'OTH':
      existTchLanguage.push(parseFloat(5));
      break;
    default:
      existTchLanguage.push(parseFloat(0));
      break;
  }
}
console.log("idealTchLanguage: " + idealTchLanguage);
console.log("existTchLanguage: " + existTchLanguage);
if (idealTchLanguage != 0) {
  let tchLanguageFlg = false;
  for (let i = 0; i < existTchLanguage.length; i++) {
    if (idealTchLanguage == existTchLanguage[i]) {
      tchLanguageFlg = true;
      break;
    }
  }
  idealTutor.push(parseFloat(idealTchLanguage / 5))
  tchLanguageFlg ? dataTutor.push(parseFloat(idealTchLanguage / 5)) : dataTutor.push(parseFloat(0 / 5));
}
else {
  idealTutor.push(parseFloat(existTchLanguage[0] / 5));
  dataTutor.push(parseFloat(existTchLanguage[0] / 5));
}
```

Figure 6.31: Section Code (Server API) – Similarity Measure Part 3



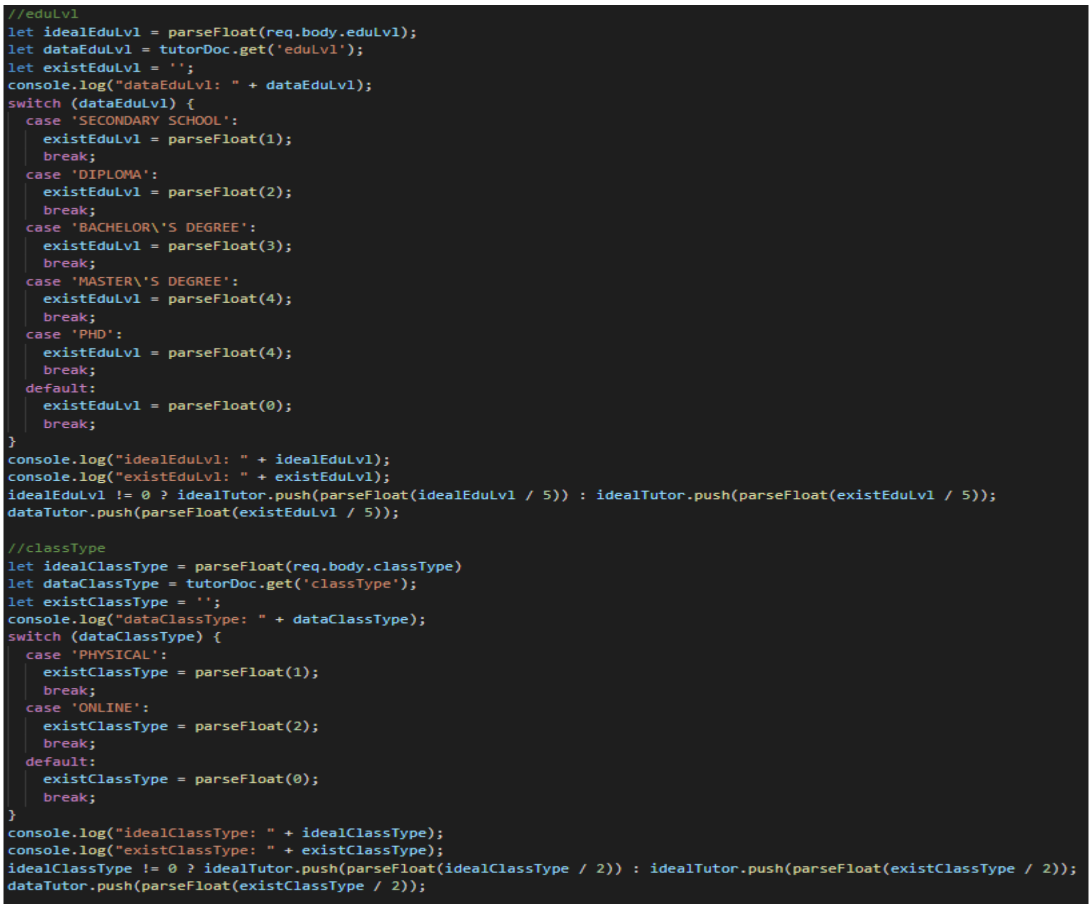
```
//tchStyle
let idealTchStyle = parseFloat(req.body.tchStyle);
let dataTchStyle = tutorDoc.get('tchStyle');
let existTchStyle = '';
console.log("dataTchStyle: " + dataTchStyle);
switch (dataTchStyle) {
  case 'ENCOURAGING':
    existTchStyle = parseFloat(1);
    break;
  case 'FUNNY':
    existTchStyle = parseFloat(2);
    break;
  case 'LIVELY':
    existTchStyle = parseFloat(3);
    break;
  case 'STRICT':
    existTchStyle = parseFloat(4);
    break;
  default:
    existTchStyle = parseFloat(0);
    break;
}
console.log("idealTchStyle: " + idealTchStyle);
console.log("existTchStyle: " + existTchStyle);
idealTchStyle != 0 ? idealTutor.push(parseFloat(idealTchStyle / 4)) : idealTutor.push(parseFloat(existTchStyle / 4));
dataTutor.push(parseFloat(existTchStyle / 4));

//tchExperience
let idealTchExperience = parseFloat(req.body.tchExperience);
let dataTchExperience = tutorDoc.get('tchExperience');
let existTchExperience = '';
console.log("dataTchExperience: " + dataTchExperience);
if (dataTchExperience >= 0 && dataTchExperience <= 4) {
  existTchExperience = parseFloat(1);
}
else if (dataTchExperience >= 5 && dataTchExperience <= 9) {
  existTchExperience = parseFloat(2);
}
else if (dataTchExperience >= 10) {
  existTchExperience = parseFloat(3);
}
else {
  existTchExperience = parseFloat(0);
}
console.log("idealTchExperience: " + idealTchExperience);
console.log("existTchExperience: " + existTchExperience);
idealTchExperience != 0 ? idealTutor.push(parseFloat(idealTchExperience / 3)) : idealTutor.push(parseFloat(existTchExperience / 3));
dataTutor.push(parseFloat(existTchExperience / 3));
```

Figure 6.32: Section Code (Server API) – Similarity Measure Part 4



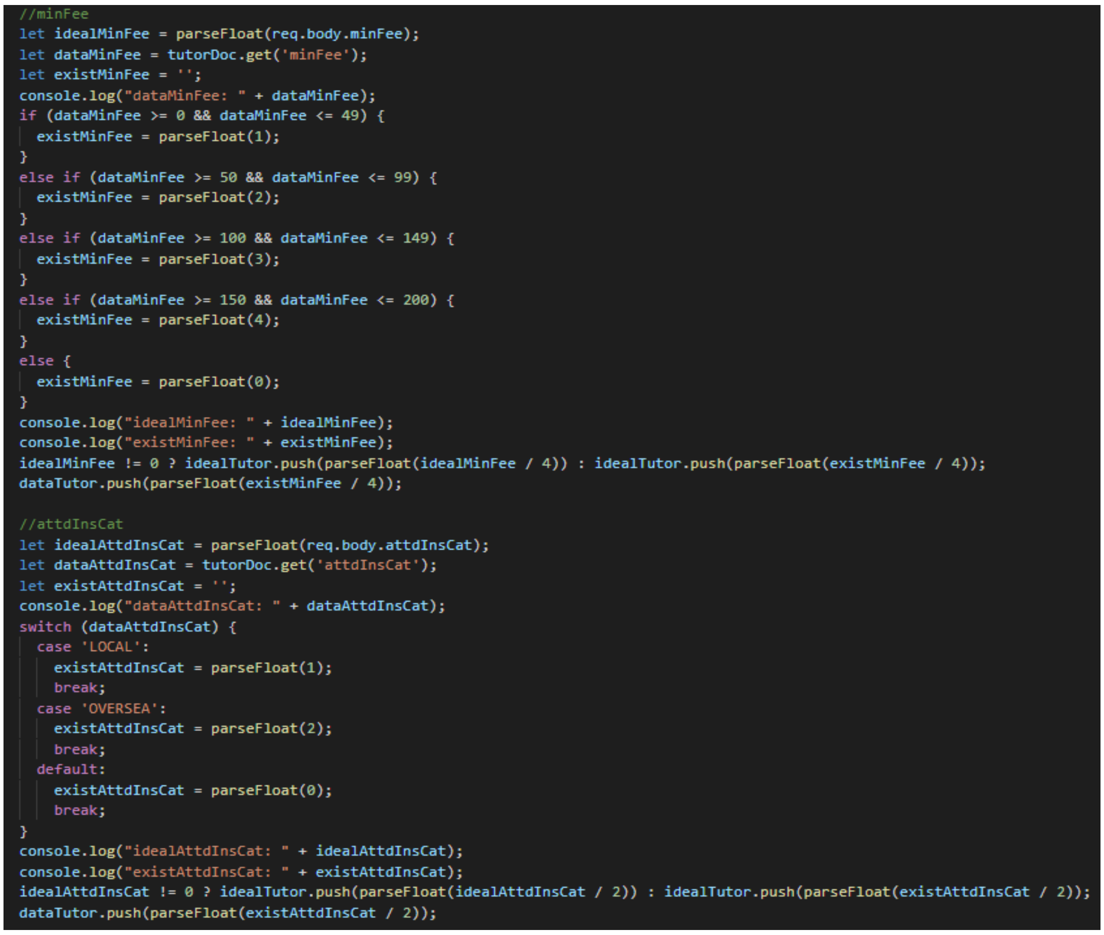
```
//tchApproach
let idealTchApproach = parseFloat(req.body.tchApproach);
let dataTchApproach = tutorDoc.get('tchApproach');
let existTchApproach = [];
console.log("dataTchApproach: " + dataTchApproach);
for (let i = 0; i < dataTchApproach.length; i++) {
  switch (dataTchApproach[i]) {
    case 'ONE-TO-ONE':
      existTchApproach.push(parseFloat(1));
      break;
    case 'SMALL GROUP':
      existTchApproach.push(parseFloat(2));
      break;
    case 'LARGE GROUP':
      existTchApproach.push(parseFloat(3));
      break;
    default:
      existTchApproach.push(parseFloat(0));
      break;
  }

}
console.log("idealTchApproach: " + idealTchApproach);
console.log("existTchApproach: " + existTchApproach);
if (idealTchApproach != 0) {
  let tchApproachFlg = false;
  for (let i = 0; i < existTchApproach.length; i++) {
    if (idealTchApproach == existTchApproach[i]) {
      tchApproachFlg = true;
      break;
    }
  }
  idealTutor.push(parseFloat(idealTchApproach / 3))
  tchApproachFlg ? dataTutor.push(parseFloat(idealTchApproach / 3)) : dataTutor.push(parseFloat(0 / 3));
}
else {
  idealTutor.push(parseFloat(existTchApproach[0] / 3));
  dataTutor.push(parseFloat(existTchApproach[0] / 3));
}
```

Figure 6.33: Section Code (Server API) – Similarity Measure Part 5



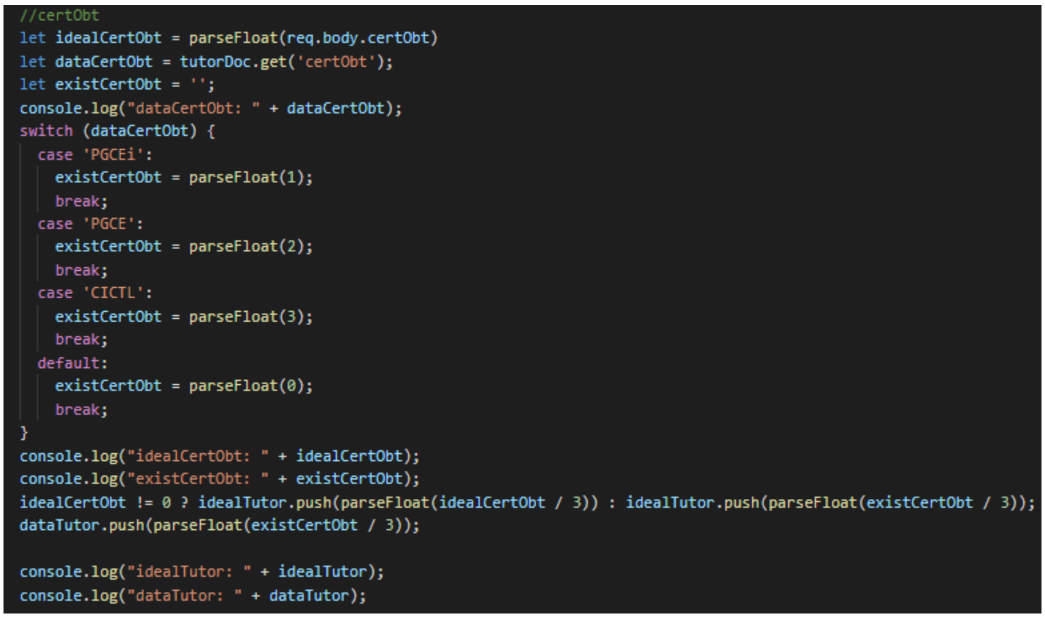
```
//eduLvl
let idealEduLvl = parseFloat(req.body.eduLvl);
let dataEduLvl = tutorDoc.get('eduLvl');
let existEduLvl = '';
console.log("dataEduLvl: " + dataEduLvl);
switch (dataEduLvl) {
  case 'SECONDARY SCHOOL':
    existEduLvl = parseFloat(1);
    break;
  case 'DIPLOMA':
    existEduLvl = parseFloat(2);
    break;
  case 'BACHELOR\'S DEGREE':
    existEduLvl = parseFloat(3);
    break;
  case 'MASTER\'S DEGREE':
    existEduLvl = parseFloat(4);
    break;
  case 'PHD':
    existEduLvl = parseFloat(4);
    break;
  default:
    existEduLvl = parseFloat(0);
    break;
}
console.log("idealEduLvl: " + idealEduLvl);
console.log("existEduLvl: " + existEduLvl);
idealEduLvl != 0 ? idealTutor.push(parseFloat(idealEduLvl / 5)) : idealTutor.push(parseFloat(existEduLvl / 5));
dataTutor.push(parseFloat(existEduLvl / 5));

//classType
let idealClassType = parseFloat(req.body.classType)
let dataClassType = tutorDoc.get('classType');
let existClassType = '';
console.log("dataClassType: " + dataClassType);
switch (dataClassType) {
  case 'PHYSICAL':
    existClassType = parseFloat(1);
    break;
  case 'ONLINE':
    existClassType = parseFloat(2);
    break;
  default:
    existClassType = parseFloat(0);
    break;
}
console.log("idealClassType: " + idealClassType);
console.log("existClassType: " + existClassType);
idealClassType != 0 ? idealTutor.push(parseFloat(idealClassType / 2)) : idealTutor.push(parseFloat(existClassType / 2));
dataTutor.push(parseFloat(existClassType / 2));
```

Figure 6.34: Section Code (Server API) – Similarity Measure Part 6



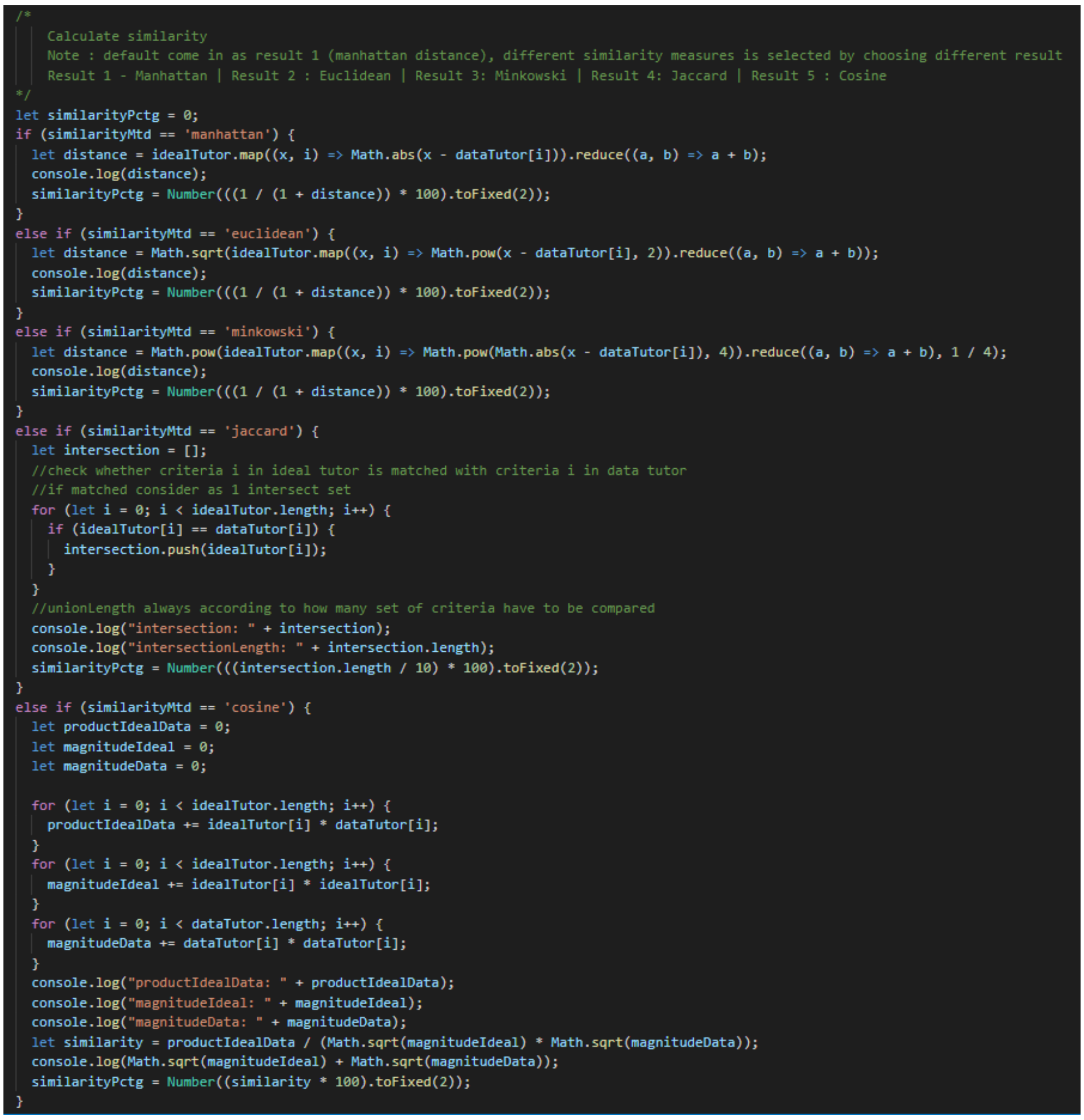
```javascript
//minFee
let idealMinFee = parseFloat(req.body.minFee);
let dataMinFee = tutorDoc.get('minFee');
let existMinFee = '';
console.log("dataMinFee: " + dataMinFee);
if (dataMinFee >= 0 && dataMinFee <= 49) {
  existMinFee = parseFloat(1);
}
else if (dataMinFee >= 50 && dataMinFee <= 99) {
  existMinFee = parseFloat(2);
}
else if (dataMinFee >= 100 && dataMinFee <= 149) {
  existMinFee = parseFloat(3);
}
else if (dataMinFee >= 150 && dataMinFee <= 200) {
  existMinFee = parseFloat(4);
}
else {
  existMinFee = parseFloat(0);
}
console.log("idealMinFee: " + idealMinFee);
console.log("existMinFee: " + existMinFee);
idealMinFee != 0 ? idealTutor.push(parseFloat(idealMinFee / 4)) : idealTutor.push(parseFloat(existMinFee / 4));
dataTutor.push(parseFloat(existMinFee / 4));

//attdInsCat
let idealAttdInsCat = parseFloat(req.body.attdInsCat);
let dataAttdInsCat = tutorDoc.get('attdInsCat');
let existAttdInsCat = '';
console.log("dataAttdInsCat: " + dataAttdInsCat);
switch (dataAttdInsCat) {
  case 'LOCAL':
    existAttdInsCat = parseFloat(1);
    break;
  case 'OVERSEA':
    existAttdInsCat = parseFloat(2);
    break;
  default:
    existAttdInsCat = parseFloat(0);
    break;
}
console.log("idealAttdInsCat: " + idealAttdInsCat);
console.log("existAttdInsCat: " + existAttdInsCat);
idealAttdInsCat != 0 ? idealTutor.push(parseFloat(idealAttdInsCat / 2)) : idealTutor.push(parseFloat(existAttdInsCat / 2));
dataTutor.push(parseFloat(existAttdInsCat / 2));
```

Figure 6.35: Section Code (Server API) – Similarity Measure Part 7



```javascript
//certObt
let idealCertObt = parseFloat(req.body.certObt)
let dataCertObt = tutorDoc.get('certObt');
let existCertObt = '';
console.log("dataCertObt: " + dataCertObt);
switch (dataCertObt) {
  case 'PGCEi':
    existCertObt = parseFloat(1);
    break;
  case 'PGCE':
    existCertObt = parseFloat(2);
    break;
  case 'CICTL':
    existCertObt = parseFloat(3);
    break;
  default:
    existCertObt = parseFloat(0);
    break;
}
console.log("idealCertObt: " + idealCertObt);
console.log("existCertObt: " + existCertObt);
idealCertObt != 0 ? idealTutor.push(parseFloat(idealCertObt / 3)) : idealTutor.push(parseFloat(existCertObt / 3));
dataTutor.push(parseFloat(existCertObt / 3));

console.log("idealTutor: " + idealTutor);
console.log("dataTutor: " + dataTutor);
```

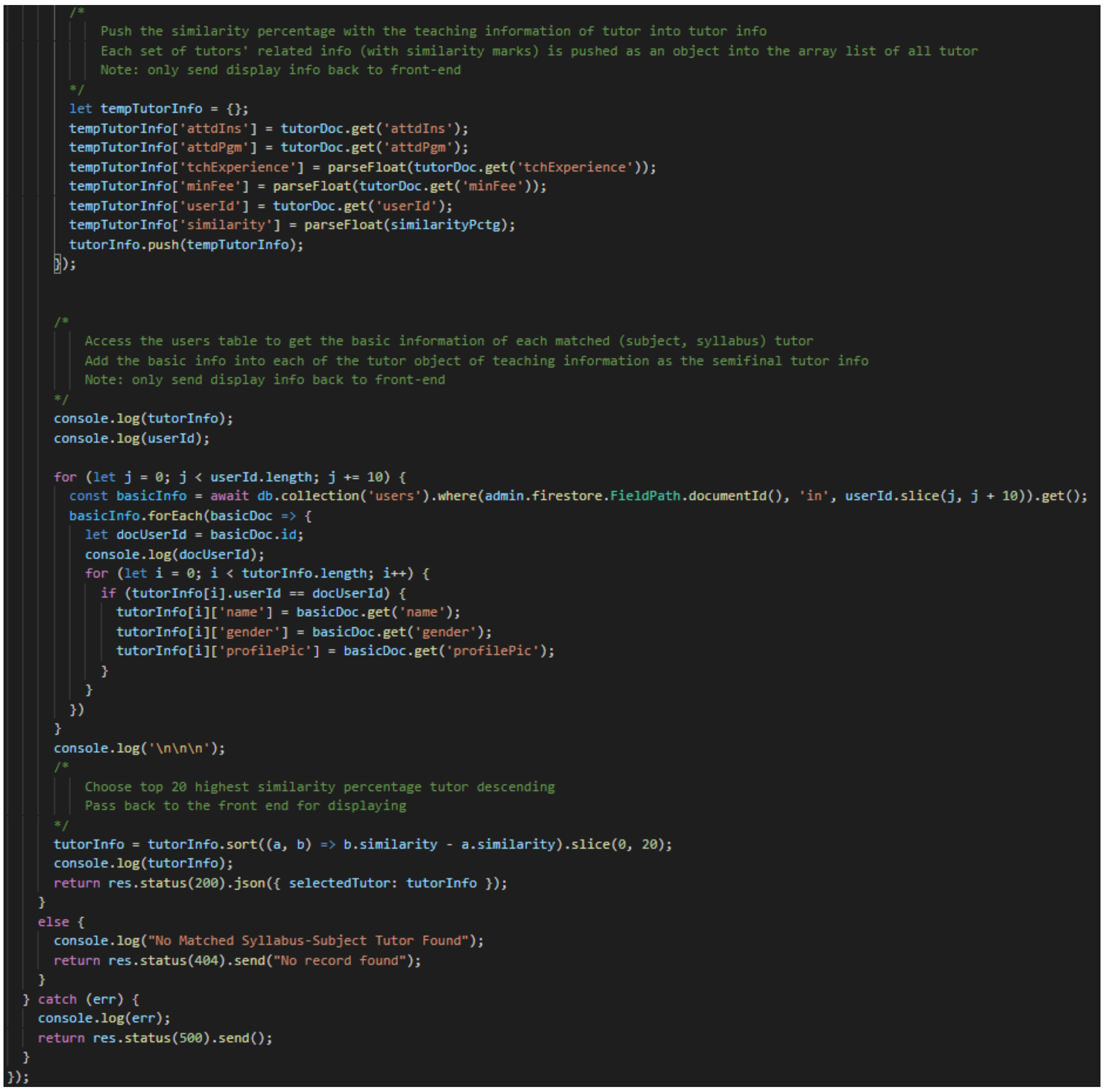
Figure 6.36: Section Code (Server API) – Similarity Measure Part 8



```
/*
    Calculate similarity
    Note : default come in as result 1 (manhattan distance), different similarity measures is selected by choosing different result
    Result 1 - Manhattan | Result 2 : Euclidean | Result 3: Minkowski | Result 4: Jaccard | Result 5 : Cosine
*/
let similarityPctg = 0;
if (similarityMtd == 'manhattan') {
  let distance = idealTutor.map((x, i) => Math.abs(x - dataTutor[i])).reduce((a, b) => a + b);
  console.log(distance);
  similarityPctg = Number(((1 / (1 + distance)) * 100).toFixed(2));
}
else if (similarityMtd == 'euclidean') {
  let distance = Math.sqrt(idealTutor.map((x, i) => Math.pow(x - dataTutor[i], 2)).reduce((a, b) => a + b));
  console.log(distance);
  similarityPctg = Number(((1 / (1 + distance)) * 100).toFixed(2));
}
else if (similarityMtd == 'minkowski') {
  let distance = Math.pow(idealTutor.map((x, i) => Math.pow(Math.abs(x - dataTutor[i]), 4)).reduce((a, b) => a + b), 1 / 4);
  console.log(distance);
  similarityPctg = Number(((1 / (1 + distance)) * 100).toFixed(2));
}
else if (similarityMtd == 'jaccard') {
  let intersection = [];
  //check whether criteria i in ideal tutor is matched with criteria i in data tutor
  //if matched consider as 1 intersect set
  for (let i = 0; i < idealTutor.length; i++) {
    if (idealTutor[i] == dataTutor[i]) {
      intersection.push(idealTutor[i]);
    }
  }
  //unionLength always according to how many set of criteria have to be compared
  console.log("intersection: " + intersection);
  console.log("intersectionLength: " + intersection.length);
  similarityPctg = Number(((intersection.length / 10) * 100).toFixed(2));
}
else if (similarityMtd == 'cosine') {
  let productIdealData = 0;
  let magnitudeIdeal = 0;
  let magnitudeData = 0;

  for (let i = 0; i < idealTutor.length; i++) {
    productIdealData += idealTutor[i] * dataTutor[i];
  }
  for (let i = 0; i < idealTutor.length; i++) {
    magnitudeIdeal += idealTutor[i] * idealTutor[i];
  }
  for (let i = 0; i < dataTutor.length; i++) {
    magnitudeData += dataTutor[i] * dataTutor[i];
  }
  console.log("productIdealData: " + productIdealData);
  console.log("magnitudeIdeal: " + magnitudeIdeal);
  console.log("magnitudeData: " + magnitudeData);
  let similarity = productIdealData / (Math.sqrt(magnitudeIdeal) * Math.sqrt(magnitudeData));
  console.log(Math.sqrt(magnitudeIdeal) + Math.sqrt(magnitudeData));
  similarityPctg = Number((similarity * 100).toFixed(2));
}
```

Figure 6.37: Section Code (Server API) – Similarity Measure Part 9

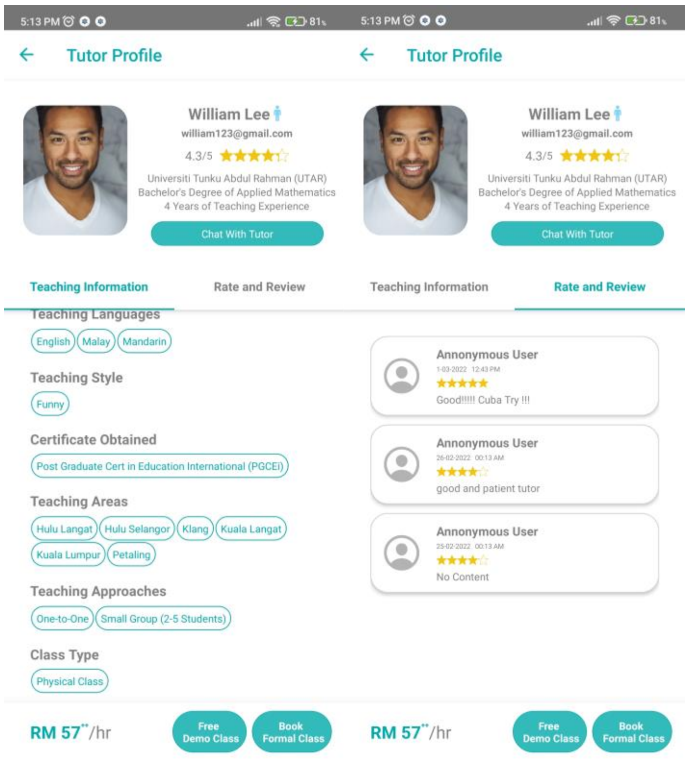
The similarity between 2 sets of numbers is calculated by applying different similarity measures according to the user result selection in the front-end. The similarity measures include Manhattan distance, Euclidean distance, Minkowski distance, Jaccard Similarity Coefficient, and Cosine similarity. In order to get a more accurate result display, the similarity measures method will be reduced to three or even the only one similarity measure after conducting the user acceptance testing (UAT) and receiving the rates from users.



```
        /*
            Push the similarity percentage with the teaching information of tutor into tutor info
            Each set of tutors' related info (with similarity marks) is pushed as an object into the array list of all tutor
            Note: only send display info back to front-end
        */
        let tempTutorInfo = {};
        tempTutorInfo['attdIns'] = tutorDoc.get('attdIns');
        tempTutorInfo['attdPgm'] = tutorDoc.get('attdPgm');
        tempTutorInfo['tchExperience'] = parseFloat(tutorDoc.get('tchExperience'));
        tempTutorInfo['minFee'] = parseFloat(tutorDoc.get('minFee'));
        tempTutorInfo['userId'] = tutorDoc.get('userId');
        tempTutorInfo['similarity'] = parseFloat(similarityPctg);
        tutorInfo.push(tempTutorInfo);
      });


      /*
          Access the users table to get the basic information of each matched (subject, syllabus) tutor
          Add the basic info into each of the tutor object of teaching information as the semifinal tutor info
          Note: only send display info back to front-end
      */
      console.log(tutorInfo);
      console.log(userId);

      for (let j = 0; j < userId.length; j += 10) {
        const basicInfo = await db.collection('users').where(admin.firestore.FieldPath.documentId(), 'in', userId.slice(j, j + 10)).get();
        basicInfo.forEach(basicDoc => {
          let docUserId = basicDoc.id;
          console.log(docUserId);
          for (let i = 0; i < tutorInfo.length; i++) {
            if (tutorInfo[i].userId == docUserId) {
              tutorInfo[i]['name'] = basicDoc.get('name');
              tutorInfo[i]['gender'] = basicDoc.get('gender');
              tutorInfo[i]['profilePic'] = basicDoc.get('profilePic');
            }
          }
        })
      }
      console.log('\n\n\n');
      /*
          Choose top 20 highest similarity percentage tutor descending
          Pass back to the front end for displaying
      */
      tutorInfo = tutorInfo.sort((a, b) => b.similarity - a.similarity).slice(0, 20);
      console.log(tutorInfo);
      return res.status(200).json({ selectedTutor: tutorInfo });
    }
    else {
      console.log("No Matched Syllabus-Subject Tutor Found");
      return res.status(404).send("No record found");
    }
  } catch (err) {
    console.log(err);
    return res.status(500).send();
  }
});
```

Figure 6.38: Section Code (Server API) – Similarity Measure Part 10

Last, insert the tutor’s information from the ‘users’ and ‘tutorInfo’ collection which is not included in the similarity calculation. The array is sorted according to the similarity percentage from the highest to the lowest and the top 20 tutor’s information is returned in JSON format to the front-end as the search result list displayed in figure 6.36.



William Lee
william123@gmail.com
4.3/5
Universiti Tunku Abdul Rahman (UTAR)
Bachelor's Degree of Applied Mathematics
4 Years of Teaching Experience

Teaching Languages
English | Malay | Mandarin

Teaching Style
Funny

Certificate Obtained
Post Graduate Cert in Education International (PGCEi)

Teaching Areas
Hulu Langat | Hulu Selangor | Klang | Kuala Langat | Kuala Lumpur | Petaling

Teaching Approaches
One-to-One | Small Group (2-5 Students)

Class Type
Physical Class

Annonymous User
1-03-2022 12:43 PM
Good!!!!! Cuba Try !!!

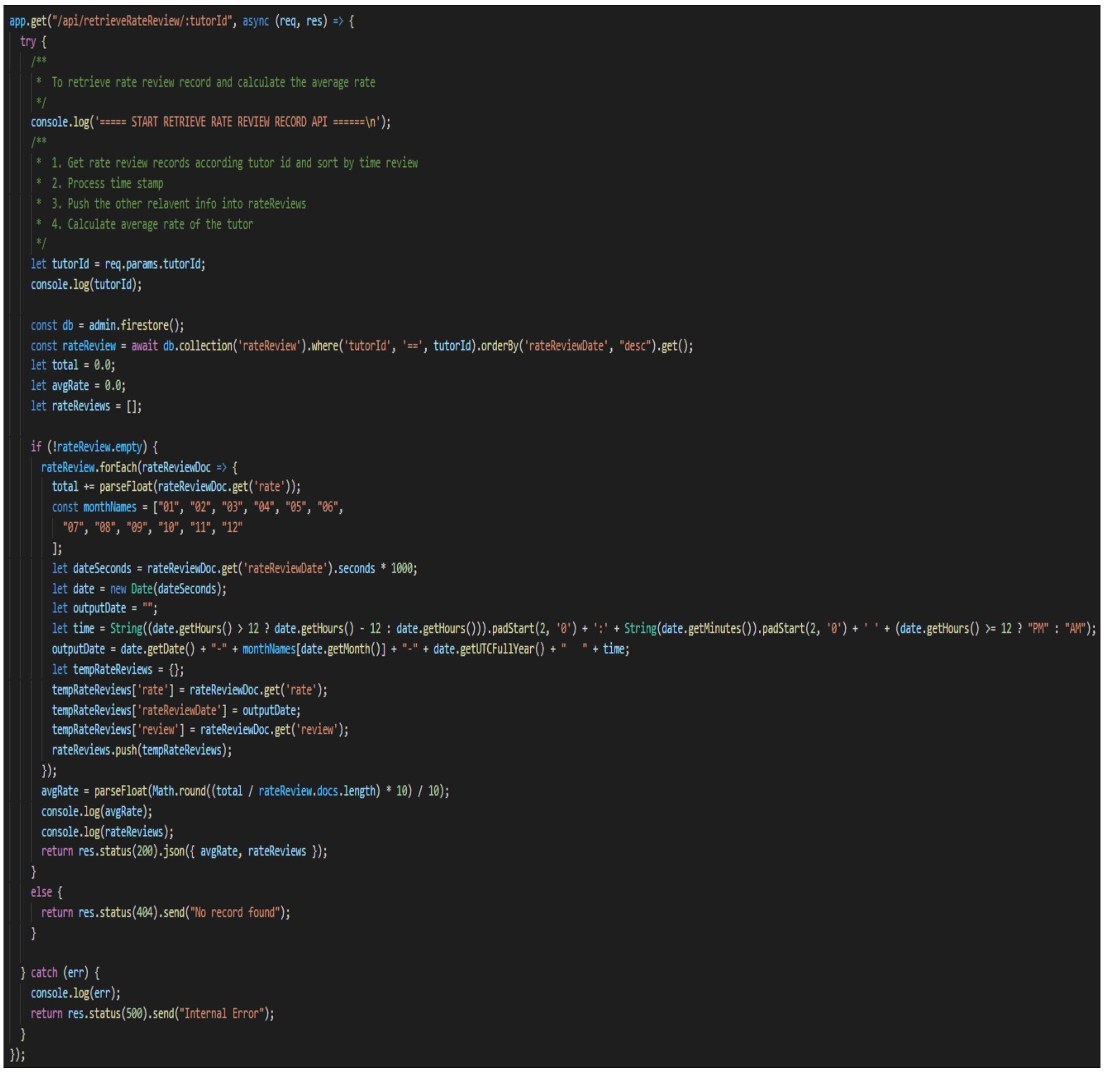
Annonymous User
26-02-2022 00:13 AM
good and patient tutor

Annonymous User
25-02-2022 00:13 AM
No Content

RM 57**/hr

Figure 6.39: Search (View Tutor Profile)

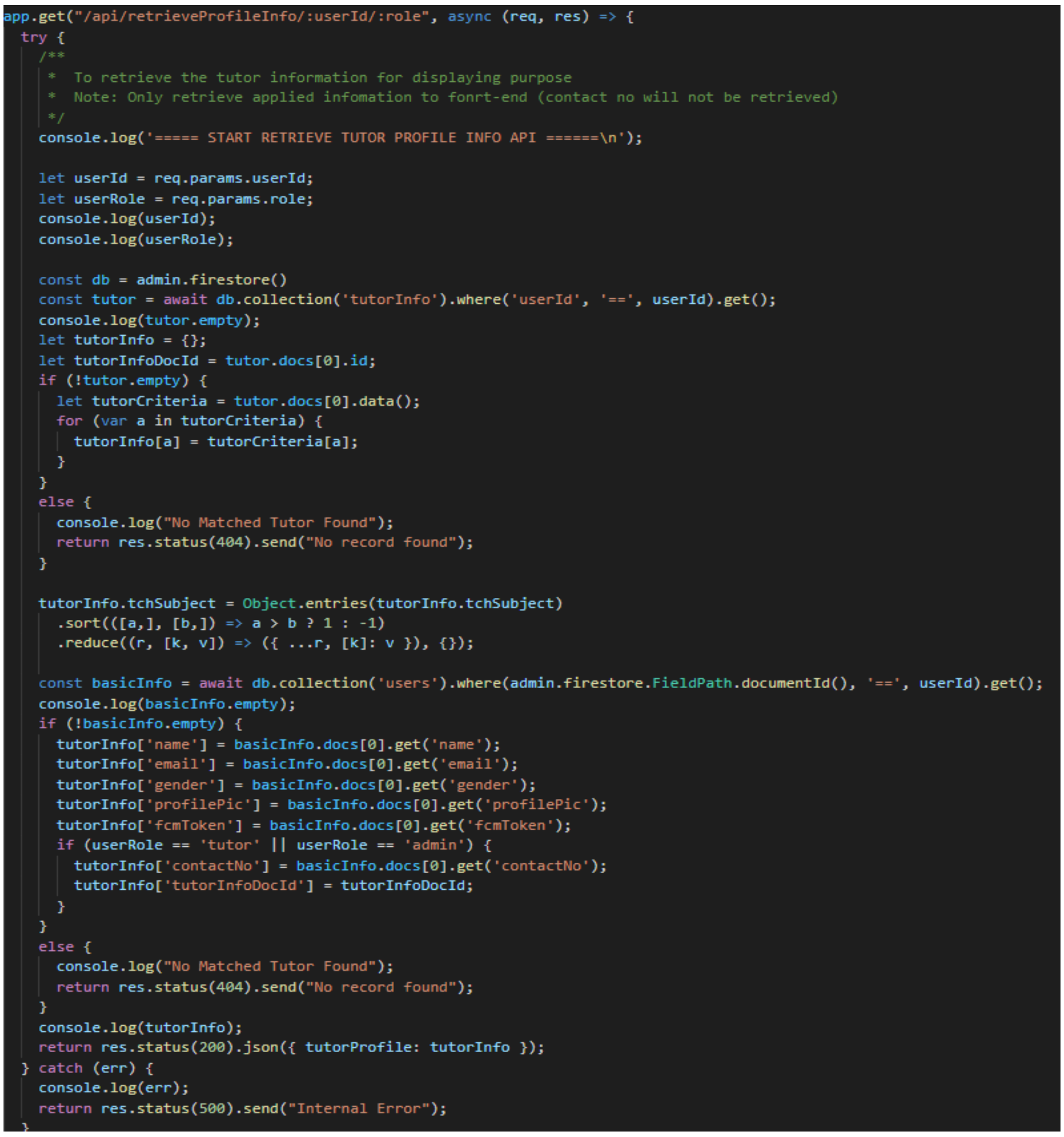
The "/api/retrieveProfileInfo/:userId/:role" API and "/api/retrieveRateReview/:tutorId" API are called when entering the tutor profile screen to retrieve the tutor’s information and tutor’s rate and review respectively. The view tutor profile activity can be conducted from the home screen and history screen as the navigation diagram in CHAPTER 5.



```
app.get("/api/retrieveRateReview/:tutorId", async (req, res) => {
  try {
    /**
     *  To retrieve rate review record and calculate the average rate
     */
    console.log('===== START RETRIEVE RATE REVIEW RECORD API ======\n');
    /**
     *  1. Get rate review records according tutor id and sort by time review
     *  2. Process time stamp
     *  3. Push the other relavent info into rateReviews
     *  4. Calculate average rate of the tutor
     */
    let tutorId = req.params.tutorId;
    console.log(tutorId);

    const db = admin.firestore();
    const rateReview = await db.collection('rateReview').where('tutorId', '==', tutorId).orderBy('rateReviewDate', "desc").get();
    let total = 0.0;
    let avgRate = 0.0;
    let rateReviews = [];

    if (!rateReview.empty) {
      rateReview.forEach(rateReviewDoc => {
        total += parseFloat(rateReviewDoc.get('rate'));
        const monthNames = ["01", "02", "03", "04", "05", "06",
          "07", "08", "09", "10", "11", "12"
        ];
        let dateSeconds = rateReviewDoc.get('rateReviewDate').seconds * 1000;
        let date = new Date(dateSeconds);
        let outputDate = "";
        let time = String((date.getHours() > 12 ? date.getHours() - 12 : date.getHours())).padStart(2, '0') + ':' + String(date.getMinutes()).padStart(2, '0') + ' ' + (date.getHours() >= 12 ? "PM" : "AM");
        outputDate = date.getDate() + "-" + monthNames[date.getMonth()] + "-" + date.getUTCFullYear() + "   " + time;
        let tempRateReviews = {};
        tempRateReviews['rate'] = rateReviewDoc.get('rate');
        tempRateReviews['rateReviewDate'] = outputDate;
        tempRateReviews['review'] = rateReviewDoc.get('review');
        rateReviews.push(tempRateReviews);
      });
      avgRate = parseFloat(Math.round((total / rateReview.docs.length) * 10) / 10);
      console.log(avgRate);
      console.log(rateReviews);
      return res.status(200).json({ avgRate, rateReviews });
    }
    else {
      return res.status(404).send("No record found");
    }

  } catch (err) {
    console.log(err);
    return res.status(500).send("Internal Error");
  }
});
```

Figure 6.40: Section Code (Server API) – Retrieve Rate Review

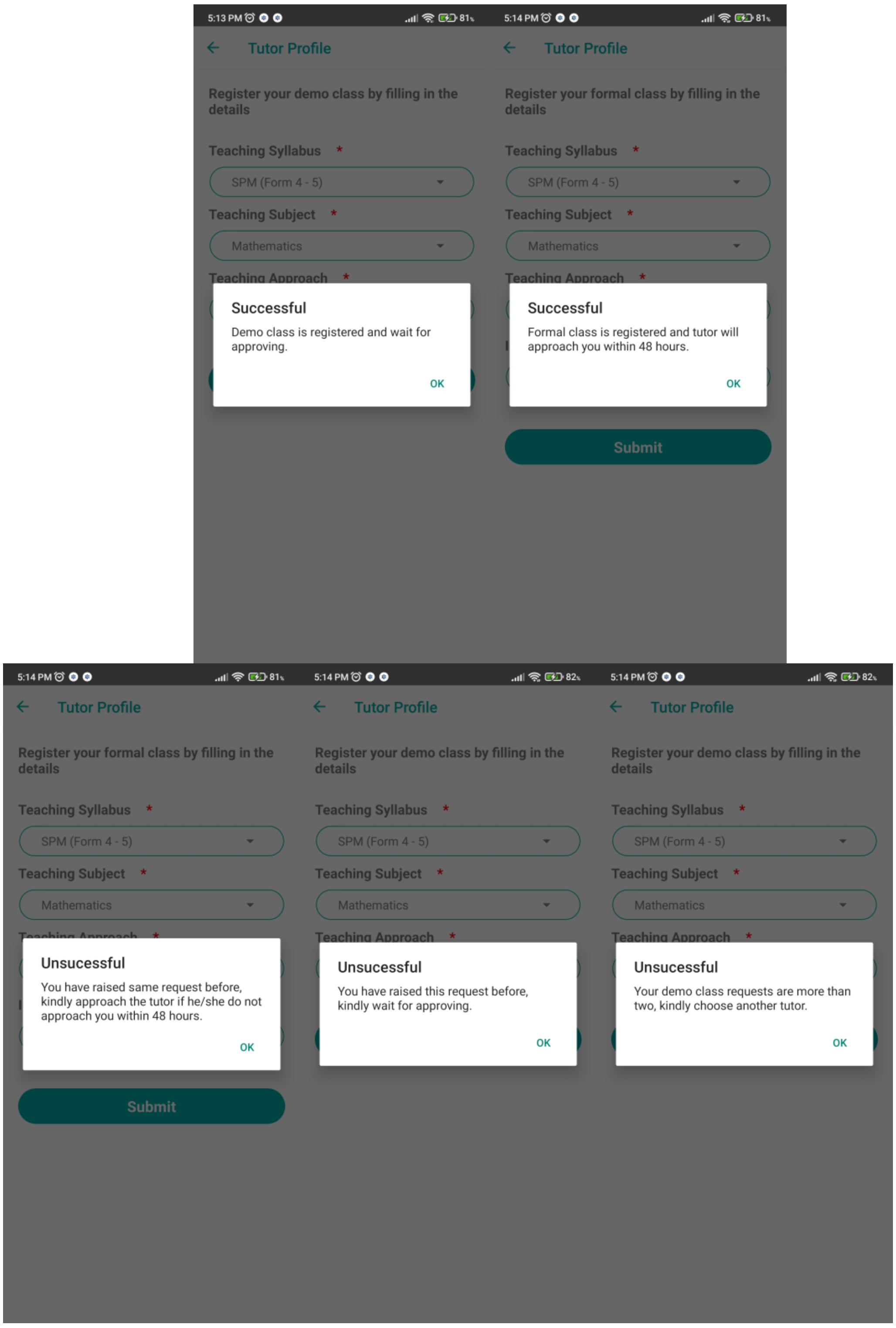
The retrieve rate review API is responsible in returning the related rate and reviews and average rate of the tutor to the front-end after the calculation. The data is retrieved with the descending order of time and the time is converted into display string. Therefore, the array list that is returned to the front-end is ready to be used without requiring the process of sorting and processing to display the rate and reviews according to descending order.



```
app.get("/api/retrieveProfileInfo/:userId/:role", async (req, res) => {
  try {
    /**
     *  To retrieve the tutor information for displaying purpose
     *  Note: Only retrieve applied infomation to fonrt-end (contact no will not be retrieved)
     */
    console.log('===== START RETRIEVE TUTOR PROFILE INFO API ======\n');

    let userId = req.params.userId;
    let userRole = req.params.role;
    console.log(userId);
    console.log(userRole);

    const db = admin.firestore()
    const tutor = await db.collection('tutorInfo').where('userId', '==', userId).get();
    console.log(tutor.empty);
    let tutorInfo = {};
    let tutorInfoDocId = tutor.docs[0].id;
    if (!tutor.empty) {
      let tutorCriteria = tutor.docs[0].data();
      for (var a in tutorCriteria) {
        tutorInfo[a] = tutorCriteria[a];
      }
    }
    else {
      console.log("No Matched Tutor Found");
      return res.status(404).send("No record found");
    }

    tutorInfo.tchSubject = Object.entries(tutorInfo.tchSubject)
      .sort(([a,], [b,]) => a > b ? 1 : -1)
      .reduce((r, [k, v]) => ({ ...r, [k]: v }), {});

    const basicInfo = await db.collection('users').where(admin.firestore.FieldPath.documentId(), '==', userId).get();
    console.log(basicInfo.empty);
    if (!basicInfo.empty) {
      tutorInfo['name'] = basicInfo.docs[0].get('name');
      tutorInfo['email'] = basicInfo.docs[0].get('email');
      tutorInfo['gender'] = basicInfo.docs[0].get('gender');
      tutorInfo['profilePic'] = basicInfo.docs[0].get('profilePic');
      tutorInfo['fcmToken'] = basicInfo.docs[0].get('fcmToken');
      if (userRole == 'tutor' || userRole == 'admin') {
        tutorInfo['contactNo'] = basicInfo.docs[0].get('contactNo');
        tutorInfo['tutorInfoDocId'] = tutorInfoDocId;
      }
    }
    else {
      console.log("No Matched Tutor Found");
      return res.status(404).send("No record found");
    }
    console.log(tutorInfo);
    return res.status(200).json({ tutorProfile: tutorInfo });
  } catch (err) {
    console.log(err);
    return res.status(500).send("Internal Error");
```

Figure 6.41: Section Code (Server API) – Retrieve Profile Information

The retrieve profile information API is used to retrieve the tutor’s information for displaying purposes which is similar to the basic profile retrieval API. The difference is a different role in accessing this API will cause API to return different information. As the figure above shown registered tutor and admin could get the contact number and document id of the tutor while not the registered tutor seeker.


5:13 PM 81%

Tutor Profile

Register your demo class by filling in the details

Teaching Syllabus *

SPM (Form 4 - 5)

Teaching Subject *

Mathematics

Teaching Approach *

Successful

Demo class is registered and wait for approving.

OK

5:14 PM 81%

Tutor Profile

Register your formal class by filling in the details

Teaching Syllabus *

SPM (Form 4 - 5)

Teaching Subject *

Mathematics

Teaching Approach *

Successful

Formal class is registered and tutor will approach you within 48 hours.

OK

Submit

5:14 PM 81%

Tutor Profile

Register your formal class by filling in the details

Teaching Syllabus *

SPM (Form 4 - 5)

Teaching Subject *

Mathematics

Teaching Approach *

Unsucessful

You have raised same request before, kindly approach the tutor if he/she do not approach you within 48 hours.

OK

Submit

5:14 PM 82%

Tutor Profile

Register your demo class by filling in the details

Teaching Syllabus *

SPM (Form 4 - 5)

Teaching Subject *

Mathematics

Teaching Approach *

Unsucessful

You have raised this request before, kindly wait for approving.

OK

5:14 PM 82%

Tutor Profile

Register your demo class by filling in the details

Teaching Syllabus *

SPM (Form 4 - 5)

Teaching Subject *

Mathematics

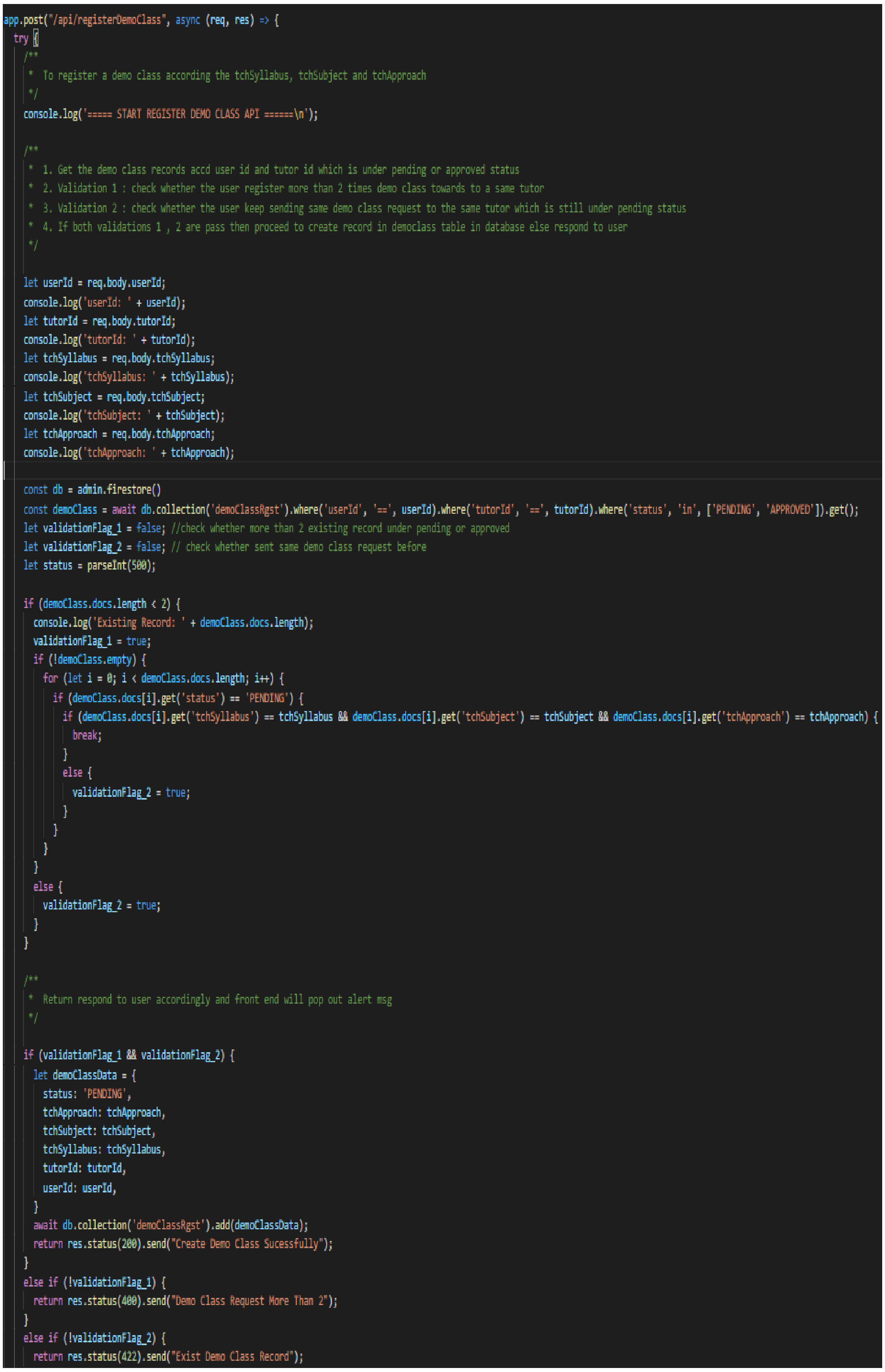
Teaching Approach *

Unsucessful

Your demo class requests are more than two, kindly choose another tutor.

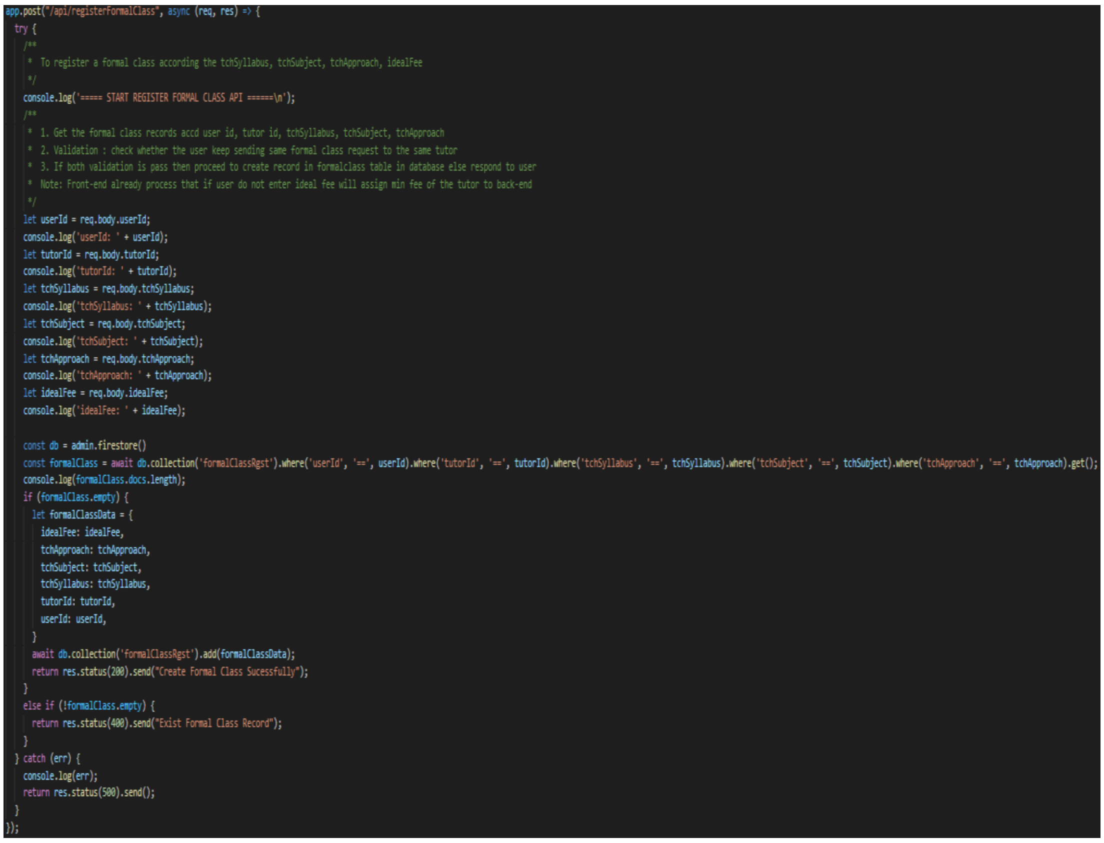
OK

Figure 6.42: Search (Demo and Formal Class Request Status)



```
app.post("/api/registerDemoClass", async (req, res) => {
  try {
    /**
     *  To register a demo class according the tchSyllabus, tchSubject and tchApproach
     */
    console.log('===== START REGISTER DEMO CLASS API ======\n');

    /**
     *  1. Get the demo class records accd user id and tutor id which is under pending or approved status
     *  2. Validation 1 : check whether the user register more than 2 times demo class towards to a same tutor
     *  3. Validation 2 : check whether the user keep sending same demo class request to the same tutor which is still under pending status
     *  4. If both validations 1 , 2 are pass then proceed to create record in democlass table in database else respond to user
     */

    let userId = req.body.userId;
    console.log('userId: ' + userId);
    let tutorId = req.body.tutorId;
    console.log('tutorId: ' + tutorId);
    let tchSyllabus = req.body.tchSyllabus;
    console.log('tchSyllabus: ' + tchSyllabus);
    let tchSubject = req.body.tchSubject;
    console.log('tchSubject: ' + tchSubject);
    let tchApproach = req.body.tchApproach;
    console.log('tchApproach: ' + tchApproach);

    const db = admin.firestore()
    const demoClass = await db.collection('demoClassRgst').where('userId', '==', userId).where('tutorId', '==', tutorId).where('status', 'in', ['PENDING', 'APPROVED']).get();
    let validationFlag_1 = false; //check whether more than 2 existing record under pending or approved
    let validationFlag_2 = false; // check whether sent same demo class request before
    let status = parseInt(500);

    if (demoClass.docs.length < 2) {
      console.log('Existing Record: ' + demoClass.docs.length);
      validationFlag_1 = true;
      if (!demoClass.empty) {
        for (let i = 0; i < demoClass.docs.length; i++) {
          if (demoClass.docs[i].get('status') == 'PENDING') {
            if (demoClass.docs[i].get('tchSyllabus') == tchSyllabus && demoClass.docs[i].get('tchSubject') == tchSubject && demoClass.docs[i].get('tchApproach') == tchApproach) {
              break;
            }
            else {
              validationFlag_2 = true;
            }
          }
        }
      }
      else {
        validationFlag_2 = true;
      }
    }

    /**
     *  Return respond to user accordingly and front end will pop out alert msg
     */

    if (validationFlag_1 && validationFlag_2) {
      let demoClassData = {
        status: 'PENDING',
        tchApproach: tchApproach,
        tchSubject: tchSubject,
        tchSyllabus: tchSyllabus,
        tutorId: tutorId,
        userId: userId,
      }
      await db.collection('demoClassRgst').add(demoClassData);
      return res.status(200).send("Create Demo Class Sucessfully");
    }
    else if (!validationFlag_1) {
      return res.status(400).send("Demo Class Request More Than 2");
    }
    else if (!validationFlag_2) {
      return res.status(422).send("Exist Demo Class Record");
```

Figure 6.43: Section Code (Server API) – Register Demo Class Part 1



```javascript
app.post("/api/registerFormalClass", async (req, res) => {
  try {
    /**
     *  To register a formal class according the tchSyllabus, tchSubject, tchApproach, idealFee
     */
    console.log('===== START REGISTER FORMAL CLASS API ======\n');
    /**
     *  1. Get the formal class records accd user id, tutor id, tchSyllabus, tchSubject, tchApproach
     *  2. Validation : check whether the user keep sending same formal class request to the same tutor
     *  3. If both validation is pass then proceed to create record in formalclass table in database else respond to user
     *  Note: Front-end already process that if user do not enter ideal fee will assign min fee of the tutor to back-end
     */
    let userId = req.body.userId;
    console.log('userId: ' + userId);
    let tutorId = req.body.tutorId;
    console.log('tutorId: ' + tutorId);
    let tchSyllabus = req.body.tchSyllabus;
    console.log('tchSyllabus: ' + tchSyllabus);
    let tchSubject = req.body.tchSubject;
    console.log('tchSubject: ' + tchSubject);
    let tchApproach = req.body.tchApproach;
    console.log('tchApproach: ' + tchApproach);
    let idealFee = req.body.idealFee;
    console.log('idealFee: ' + idealFee);

    const db = admin.firestore()
    const formalClass = await db.collection('formalClassRgst').where('userId', '==', userId).where('tutorId', '==', tutorId).where('tchSyllabus', '==', tchSyllabus).where('tchSubject', '==', tchSubject).where('tchApproach', '==', tchApproach).get();
    console.log(formalClass.docs.length);
    if (formalClass.empty) {
      let formalClassData = {
        idealFee: idealFee,
        tchApproach: tchApproach,
        tchSubject: tchSubject,
        tchSyllabus: tchSyllabus,
        tutorId: tutorId,
        userId: userId,
      }
      await db.collection('formalClassRgst').add(formalClassData);
      return res.status(200).send("Create Formal Class Sucessfully");
    }
    else if (!formalClass.empty) {
      return res.status(400).send("Exist Formal Class Record");
    }
  } catch (err) {
    console.log(err);
    return res.status(500).send();
  }
});
```

Figure 6.44: Section Code (Server API) – Register Formal Class

For demo class and formal class registration, the "/api/registerDemoClass" API or "/api/registerFormalClass" API is called when the “Submit” button is pressed in their respective screen. For register demo class API, the demo class documents whose status is pending or approved in ‘demoClassRgst’ collection is retrieved and conducted 2 validations checking before creating the new demo class document. Validation 1 is used to validate the same registered tutor seeker is only allowed to register 2 demo classes to the same registered tutor. Validation 2 is used to validate the registered seeker is not allowed to send the same class request to the same tutor. For formal class registration, only validation 2 in demo class registration is applied while the flow of API is similar. Last, the create action status will be returned to the front-end and an alert message is displayed to the registered tutor seeker in figure 6.39.

6.3.3.5 Profile Activities

The profile stack is the main in handling the profile editing activity which includes the upload photo activity that is explained above. Besides, the sign-out and delete account activity also is handled in the profile stack.

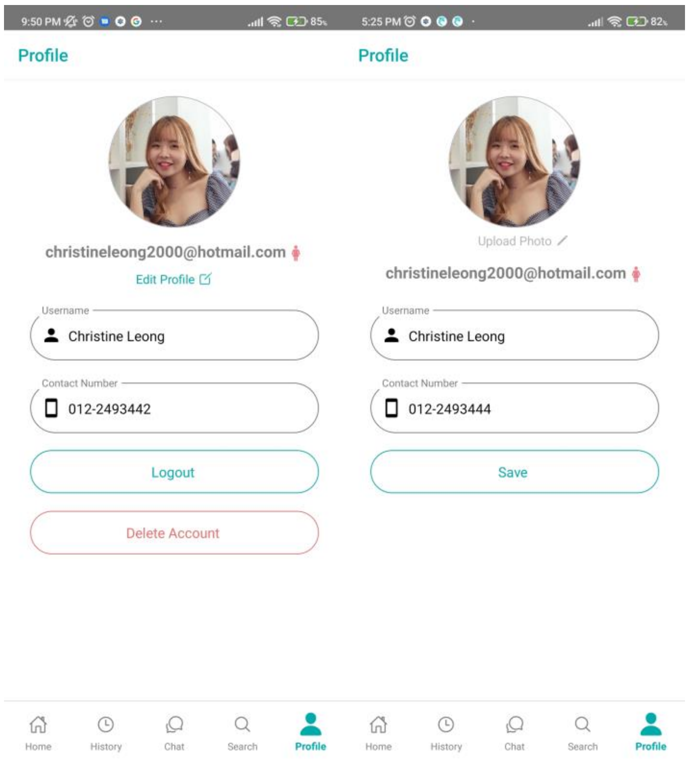
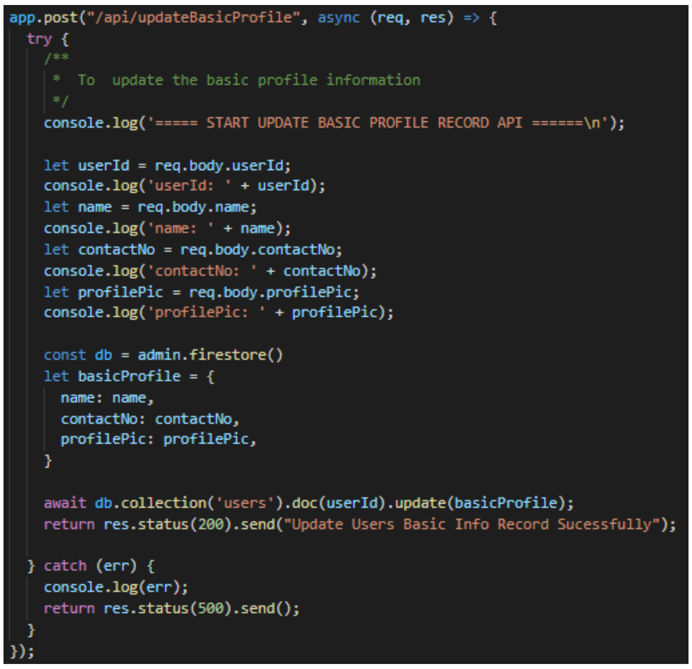
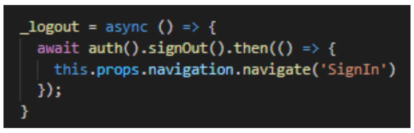


Figure 6.45: Profile (Display Profile, Edit Profile, Sign Out Account)



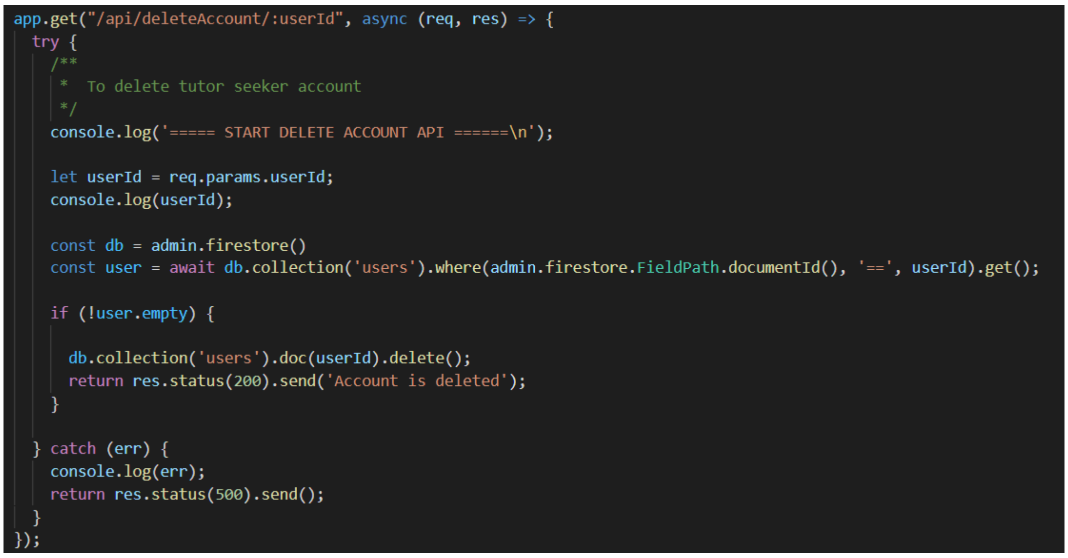
```
app.post("/api/updateBasicProfile", async (req, res) => {
  try {
    /**
     *  To  update the basic profile information
     */
    console.log('===== START UPDATE BASIC PROFILE RECORD API ======\n');

    let userId = req.body.userId;
    console.log('userId: ' + userId);
    let name = req.body.name;
    console.log('name: ' + name);
    let contactNo = req.body.contactNo;
    console.log('contactNo: ' + contactNo);
    let profilePic = req.body.profilePic;
    console.log('profilePic: ' + profilePic);

    const db = admin.firestore()
    let basicProfile = {
      name: name,
      contactNo: contactNo,
      profilePic: profilePic,
    }

    await db.collection('users').doc(userId).update(basicProfile);
    return res.status(200).send("Update Users Basic Info Record Sucessfully");

  } catch (err) {
    console.log(err);
    return res.status(500).send();
  }
});
```

Figure 6.46: Section Code (Server API) – Update Basic Profile



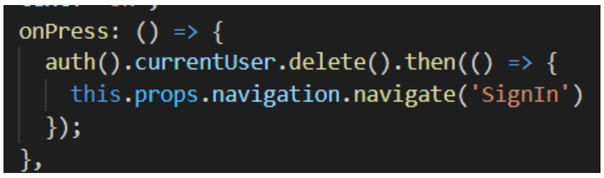
```
_logout = async () => {
  await auth().signOut().then(() => {
    this.props.navigation.navigate('SignIn')
  });
}
```

Figure 6.47: Section Code (Front-end) – Sign Out Account



```
app.get("/api/deleteAccount/:userId", async (req, res) => {
  try {
    /**
     *  To delete tutor seeker account
     */
    console.log('===== START DELETE ACCOUNT API ======\n');

    let userId = req.params.userId;
    console.log(userId);

    const db = admin.firestore()
    const user = await db.collection('users').where(admin.firestore.FieldPath.documentId(), '==', userId).get();

    if (!user.empty) {

      db.collection('users').doc(userId).delete();
      return res.status(200).send('Account is deleted');
    }

  } catch (err) {
    console.log(err);
    return res.status(500).send();
  }
});
```

Figure 6.48: Section Code (Server API) – Delete Account



```
onPress: () => {
  auth().currentUser.delete().then(() => {
    this.props.navigation.navigate('SignIn')
  });
},
```

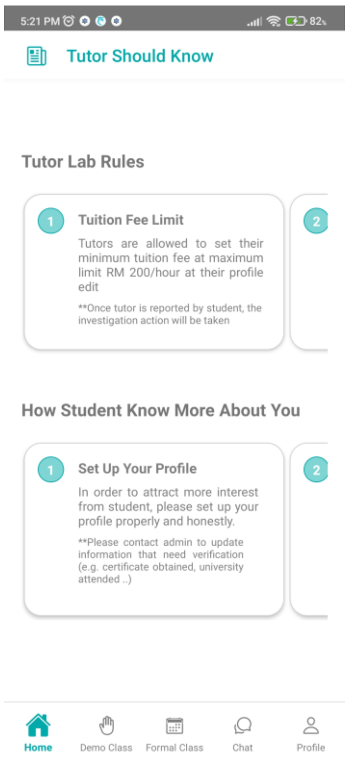
Figure 6.49: Section Code (Front-end) – Delete Account

The registered tutor seeker’s profile information is retrieved when signing in and be saved into AsyncStorage. Therefore, AsyncStorage is applied for displaying the profile while the text input is changed to editable when the registered tutor seeker presses the ‘Edit Profile’ text. The "/api/updateBasicProfile" API is called to update the ‘users’ document when the registered tutor seeker press the ‘Save’ button after modifying the name or mobile number. Besides, the application will ‘memorize’ the sign-in status of the user and redirect the user to their home screen when opening the application. The user will be redirected to the sign-in screen and sign out the account (will not sign in automatically when next open the application) after the ‘Loguot’ button is pressed. The "/api/deleteAccount" API is called to delete the ‘users’ document while the account also is removed from the authentication user list in Firebase authentication when the “delete account” button is pressed and confirm to delete the account.

6.3.4 Registered Tutor Side

6.3.4.1 Home Activities

The home stack is responsible for displaying some information without calling API while the side activities such as AsyncStorage, Firebase Cloud Messaging (FCM) token, and push notification display settings also be handled in the registered tutor side' home stack as similar in registered tutor seeker side. Implementations are explained in the home activities of the registered tutor seeker side above.


5:21 PM 82%

Tutor Should Know

Tutor Lab Rules

1 Tuition Fee Limit

Tutors are allowed to set their minimum tuition fee at maximum limit RM 200/hour at their profile edit

**Once tutor is reported by student, the investigation action will be taken

2

How Student Know More About You

1 Set Up Your Profile

In order to attract more interest from student, please set up your profile properly and honestly.

**Please contact admin to update information that need verification (e.g. certificate obtained, university attended ..)

2

Figure 6.50: Home (Registered Tutor)

6.3.4.2 Demo Class Activities

The demo class stack handles the displaying information of the demo class from the registered tutor seeker and updating the demo class's status activity while it also involves the chat activities which is explained in the section above.

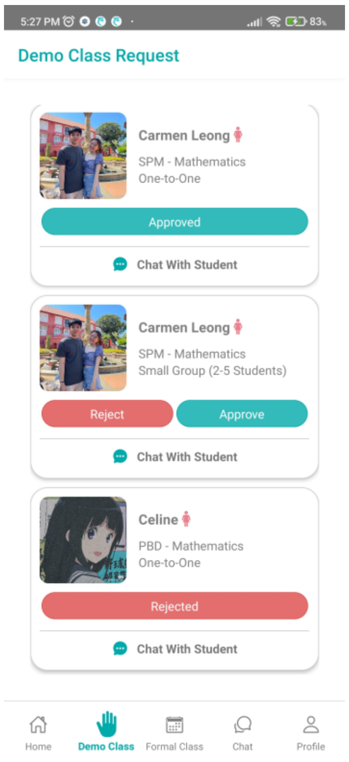
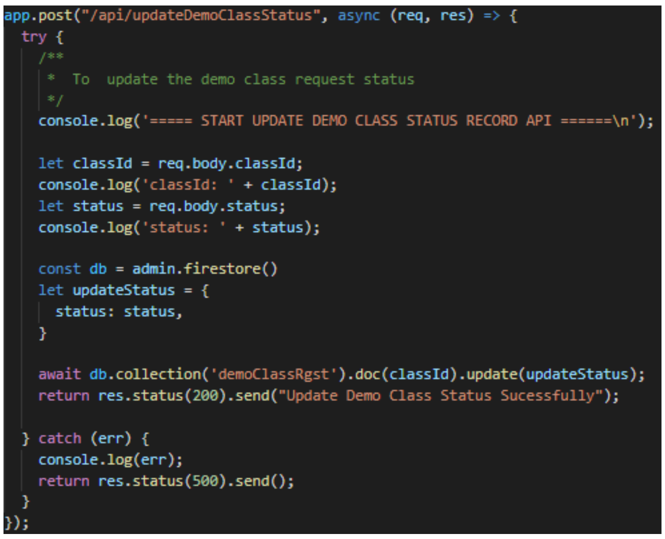
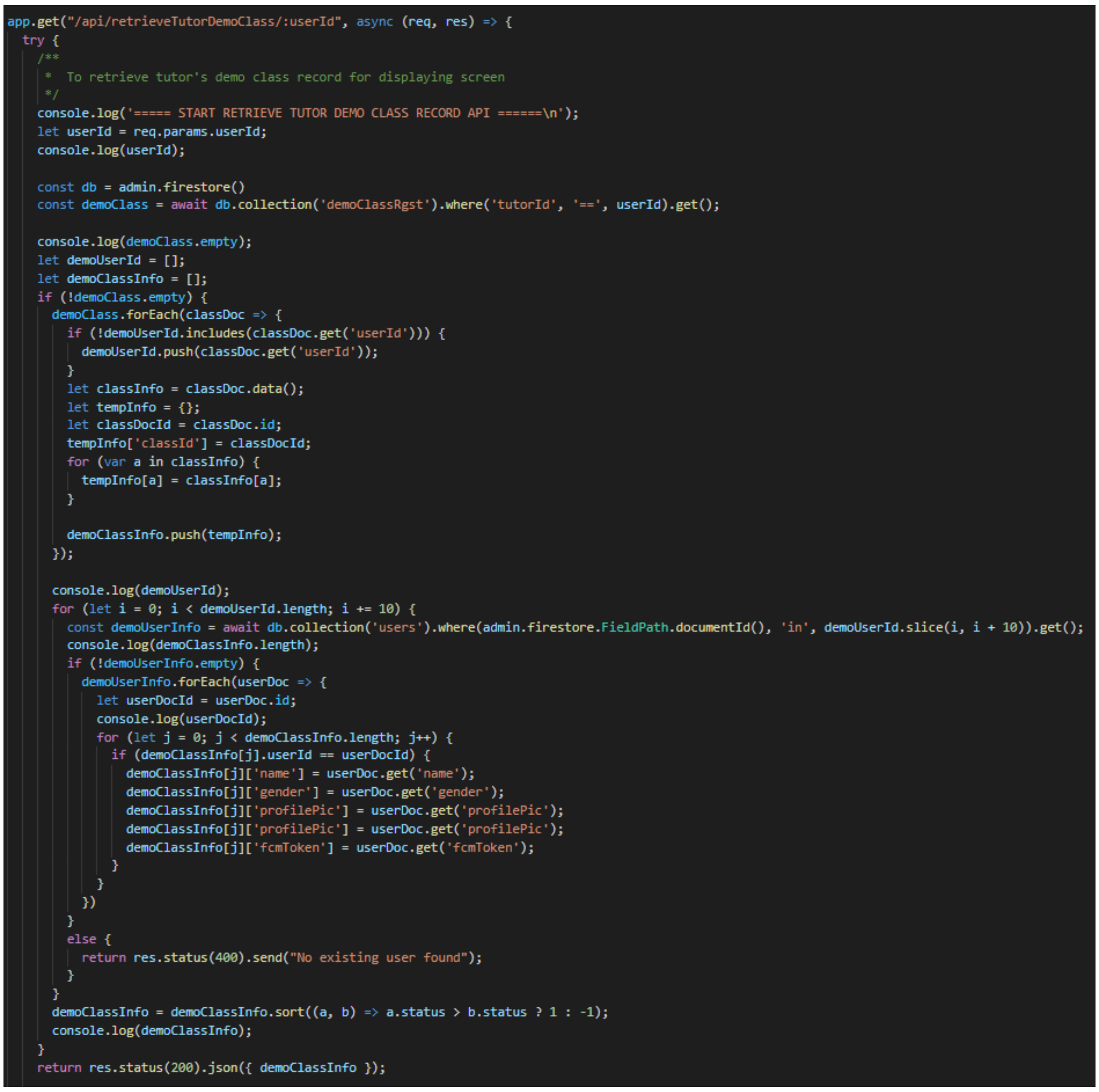


Figure 6.51: Demo Class (Display and Manage Demo Class Requests)



```
app.post("/api/updateDemoClassStatus", async (req, res) => {
  try {
    /**
     *  To  update the demo class request status
     */
    console.log('===== START UPDATE DEMO CLASS STATUS RECORD API ======\n');

    let classId = req.body.classId;
    console.log('classId:  ' + classId);
    let status = req.body.status;
    console.log('status:  ' + status);

    const db = admin.firestore()
    let updateStatus = {
      status: status,
    }

    await db.collection('demoClassRqst').doc(classId).update(updateStatus);
    return res.status(200).send("Update Demo Class Status Sucessfully");

  } catch (err) {
    console.log(err);
    return res.status(500).send();
  }
});
```

Figure 6.52: Section Code (Server API) – Update Demo Class Status



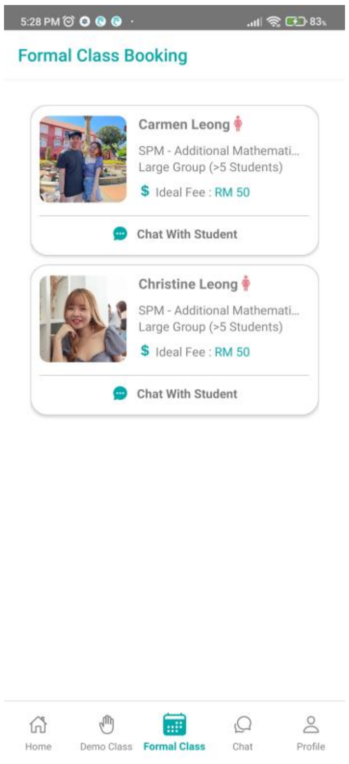
```
app.get("/api/retrieveTutorDemoClass/:userId", async (req, res) => {
  try {
    /**
     *  To retrieve tutor's demo class record for displaying screen
     */
    console.log('===== START RETRIEVE TUTOR DEMO CLASS RECORD API ======\n');
    let userId = req.params.userId;
    console.log(userId);

    const db = admin.firestore()
    const demoClass = await db.collection('demoClassRgst').where('tutorId', '==', userId).get();

    console.log(demoClass.empty);
    let demoUserId = [];
    let demoClassInfo = [];
    if (!demoClass.empty) {
      demoClass.forEach(classDoc => {
        if (!demoUserId.includes(classDoc.get('userId'))) {
          demoUserId.push(classDoc.get('userId'));
        }
        let classInfo = classDoc.data();
        let tempInfo = {};
        let classDocId = classDoc.id;
        tempInfo['classId'] = classDocId;
        for (var a in classInfo) {
          tempInfo[a] = classInfo[a];
        }

        demoClassInfo.push(tempInfo);
      });

      console.log(demoUserId);
      for (let i = 0; i < demoUserId.length; i += 10) {
        const demoUserInfo = await db.collection('users').where(admin.firestore.FieldPath.documentId(), 'in', demoUserId.slice(i, i + 10)).get();
        console.log(demoClassInfo.length);
        if (!demoUserInfo.empty) {
          demoUserInfo.forEach(userDoc => {
            let userDocId = userDoc.id;
            console.log(userDocId);
            for (let j = 0; j < demoClassInfo.length; j++) {
              if (demoClassInfo[j].userId == userDocId) {
                demoClassInfo[j]['name'] = userDoc.get('name');
                demoClassInfo[j]['gender'] = userDoc.get('gender');
                demoClassInfo[j]['profilePic'] = userDoc.get('profilePic');
                demoClassInfo[j]['profilePic'] = userDoc.get('profilePic');
                demoClassInfo[j]['fcmToken'] = userDoc.get('fcmToken');
              }
            }
          })
        }
        else {
          return res.status(400).send("No existing user found");
        }
      }
      demoClassInfo = demoClassInfo.sort((a, b) => a.status > b.status ? 1 : -1);
      console.log(demoClassInfo);
    }
    return res.status(200).json({ demoClassInfo });
```

Figure 6.53: Section Code (Server API) – Retrieve Demo Class

The AsyncStorage is applied in getting the userId as the parameter to call the "/api/retrieveTutorDemoClass/:userId" API. The API returns the list of information that includes the demo class’s information and the registered tutor seeker’s information. The "/api/updateDemoClassStatus" API is called when the registered tutor press the approve or reject button to update the status value of the ‘demoClassRgst’ document.

6.3.4.3 Formal Class Activities

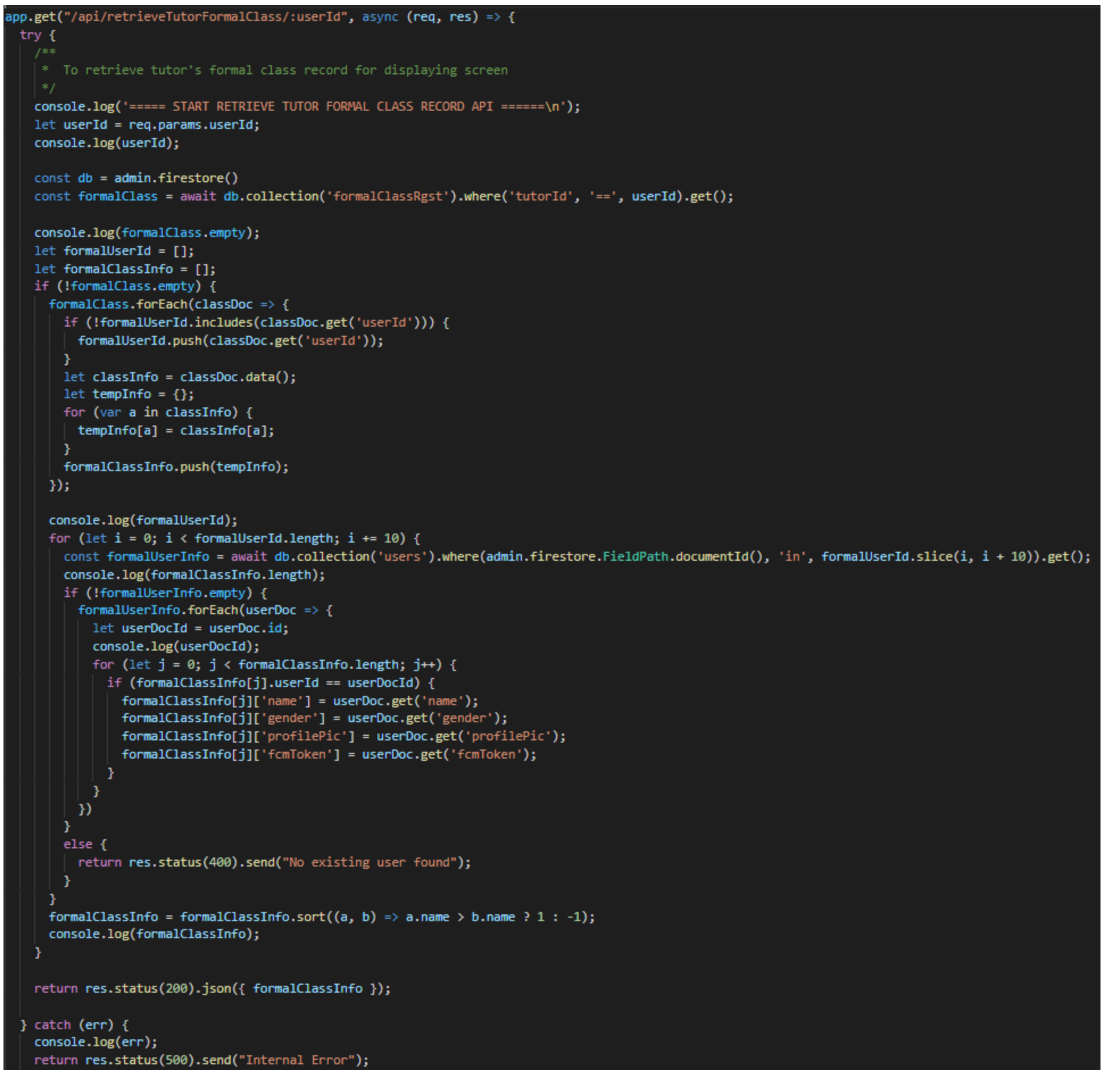
The formal class stack handles the displaying information of the formal class from the registered tutor seeker while it also involves the chat activities which is explained in the section above.


Formal Class Booking

Carmen Leong
SPM - Additional Mathemati...
Large Group (>5 Students)
$ Ideal Fee : RM 50
Chat With Student

Christine Leong
SPM - Additional Mathemati...
Large Group (>5 Students)
$ Ideal Fee : RM 50
Chat With Student

Figure 6.54: Formal Class (Display Formal Class Requests)



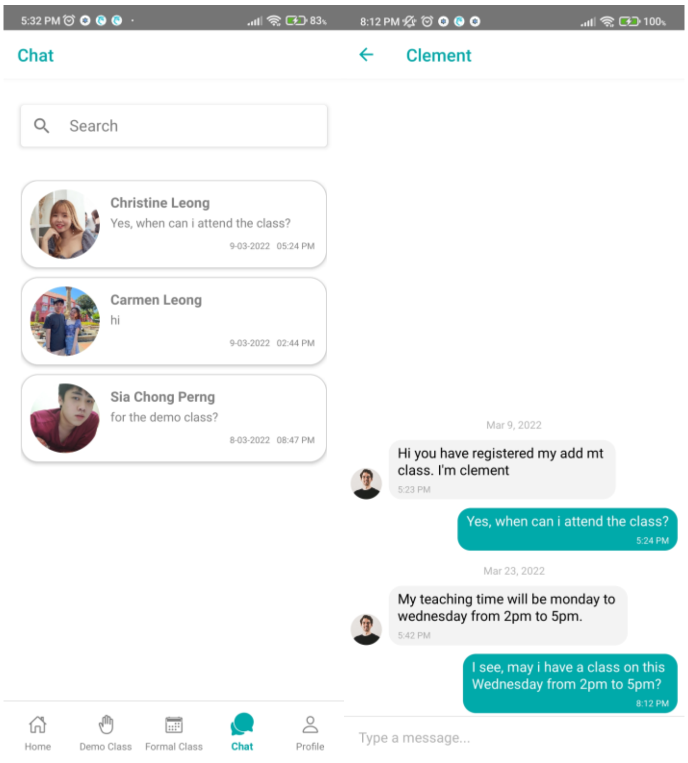
```
app.get("/api/retrieveTutorFormalClass/:userId", async (req, res) => {
  try {
    /**
     *  To retrieve tutor's formal class record for displaying screen
     */
    console.log('===== START RETRIEVE TUTOR FORMAL CLASS RECORD API ======\n');
    let userId = req.params.userId;
    console.log(userId);

    const db = admin.firestore()
    const formalClass = await db.collection('formalClassRgst').where('tutorId', '==', userId).get();

    console.log(formalClass.empty);
    let formalUserId = [];
    let formalClassInfo = [];
    if (!formalClass.empty) {
      formalClass.forEach(classDoc => {
        if (!formalUserId.includes(classDoc.get('userId'))) {
          formalUserId.push(classDoc.get('userId'));
        }
        let classInfo = classDoc.data();
        let tempInfo = {};
        for (var a in classInfo) {
          tempInfo[a] = classInfo[a];
        }
        formalClassInfo.push(tempInfo);
      });

      console.log(formalUserId);
      for (let i = 0; i < formalUserId.length; i += 10) {
        const formalUserInfo = await db.collection('users').where(admin.firestore.FieldPath.documentId(), 'in', formalUserId.slice(i, i + 10)).get();
        console.log(formalClassInfo.length);
        if (!formalUserInfo.empty) {
          formalUserInfo.forEach(userDoc => {
            let userDocId = userDoc.id;
            console.log(userDocId);
            for (let j = 0; j < formalClassInfo.length; j++) {
              if (formalClassInfo[j].userId == userDocId) {
                formalClassInfo[j]['name'] = userDoc.get('name');
                formalClassInfo[j]['gender'] = userDoc.get('gender');
                formalClassInfo[j]['profilePic'] = userDoc.get('profilePic');
                formalClassInfo[j]['fcmToken'] = userDoc.get('fcmToken');
              }
            }
          })
        }
        else {
          return res.status(400).send("No existing user found");
        }
      }
      formalClassInfo = formalClassInfo.sort((a, b) => a.name > b.name ? 1 : -1);
      console.log(formalClassInfo);
    }

    return res.status(200).json({ formalClassInfo });

  } catch (err) {
    console.log(err);
    return res.status(500).send("Internal Error");
```

Figure 6.55: Section Code (Server API) – Retrieve Formal Class

The AsyncStorage is applied in getting the userId as the parameter to call the "/api/retrieveTutorFormalClass/:userId" API. The API returns the list of information that includes the formal class's information and the registered tutor seeker's information.

6.3.4.4 Chat Activities

The chat activities on a registered tutor side are the same as the chat activities on the registered tutor seeker side while the only difference will be the activities access screen. The chat activities can be accessed by using the bottom navigation tab and the "Chat With Student" button on the demo class screen and the formal class screen on the registered tutor side.



Hi you have registered my add mt class. I'm clement

Yes, when can i attend the class?

My teaching time will be monday to wednesday from 2pm to 5pm.

I see, may i have a class on this Wednesday from 2pm to 5pm?

Figure 6.56: Chat (Registered Tutor)

6.3.4.5 Profile Activities

The profile stack is the main in handling the profile editing activity which includes the upload photo activity that is explained on the registered tutor seeker side. Besides, the sign-out account activity also is handled in the profile stack.

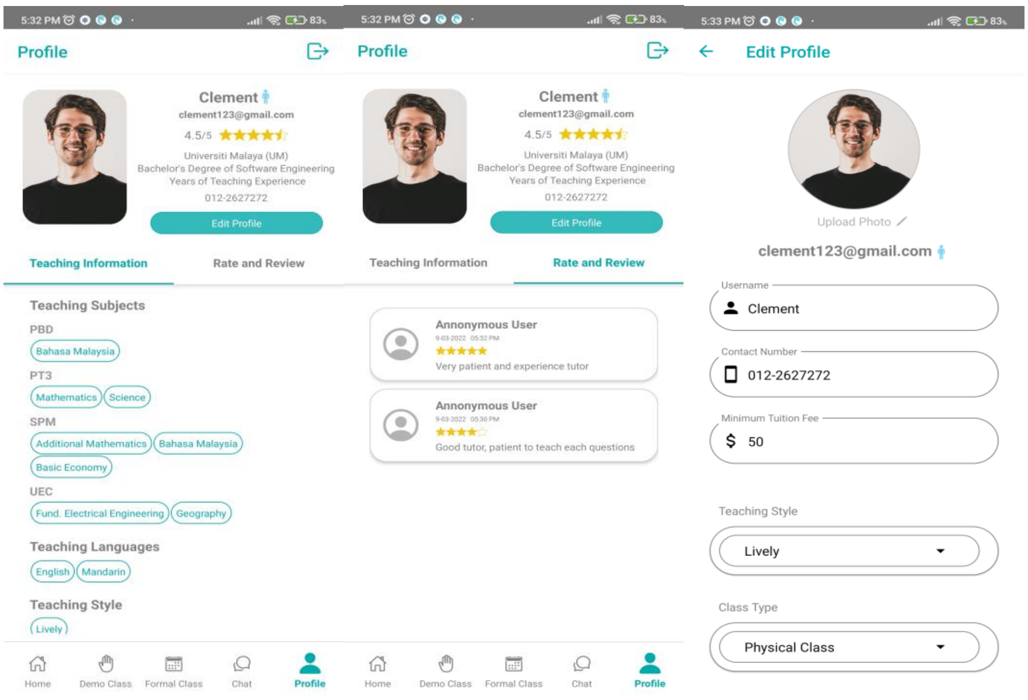
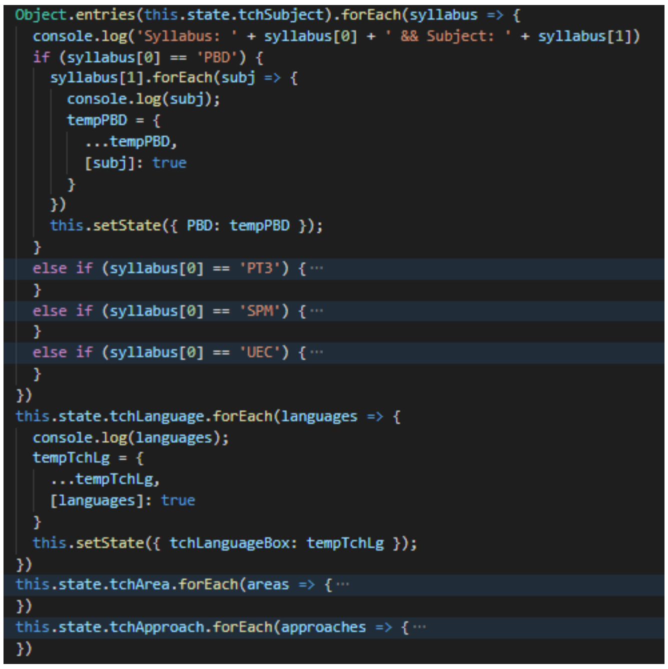


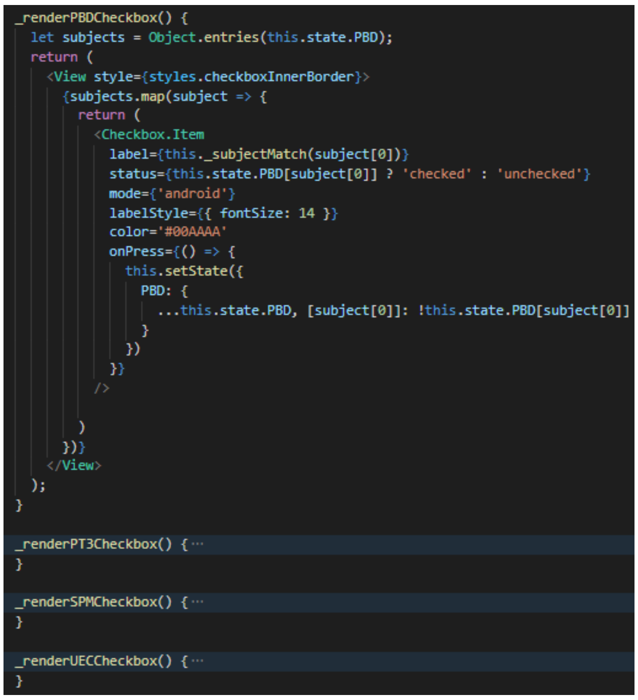
Figure 6.57: Registered Tutor Profile (Profile Display and Edit)

The display of the registered tutor’s profile is similar to the view tutor profile activity on the registered tutor seeker side. The "/api/retrieveProfileInfo/:userId/:role" API and "/api/retrieveRateReview/:tutorId" API are called when entering the profile screen to retrieve the tutor’s information and tutor’s rate and review respectively while the implementation is explained on the registered tutor seeker side.



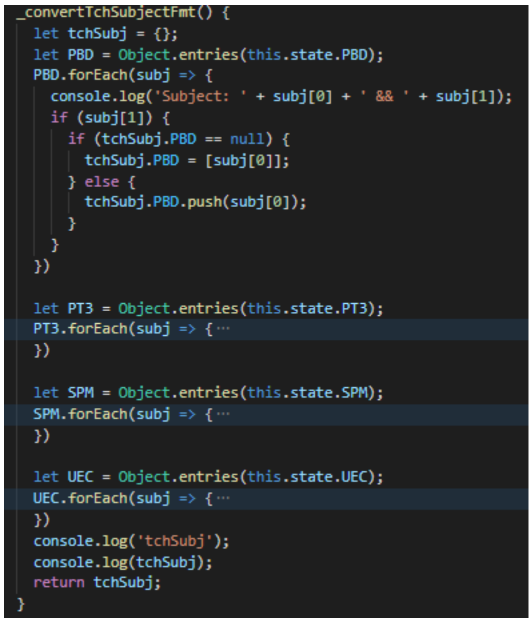
```
Object.entries(this.state.tchSubject).forEach(syllabus => {
  console.log('Syllabus: ' + syllabus[0] + ' && Subject: ' + syllabus[1])
  if (syllabus[0] == 'PBD') {
    syllabus[1].forEach(subj => {
      console.log(subj);
      tempPBD = {
        ...tempPBD,
        [subj]: true
      }
    })
    this.setState({ PBD: tempPBD });
  }
  else if (syllabus[0] == 'PT3') {⋯
  }
  else if (syllabus[0] == 'SPM') {⋯
  }
  else if (syllabus[0] == 'UEC') {⋯
  }
})
this.state.tchLanguage.forEach(languages => {
  console.log(languages);
  tempTchLg = {
    ...tempTchLg,
    [languages]: true
  }
  this.setState({ tchLanguageBox: tempTchLg });
})
this.state.tchArea.forEach(areas => {⋯
})
this.state.tchApproach.forEach(approaches => {⋯
})
```

Figure 6.58: Section Code (Front-end) – Edit Tutor Profile Part 1



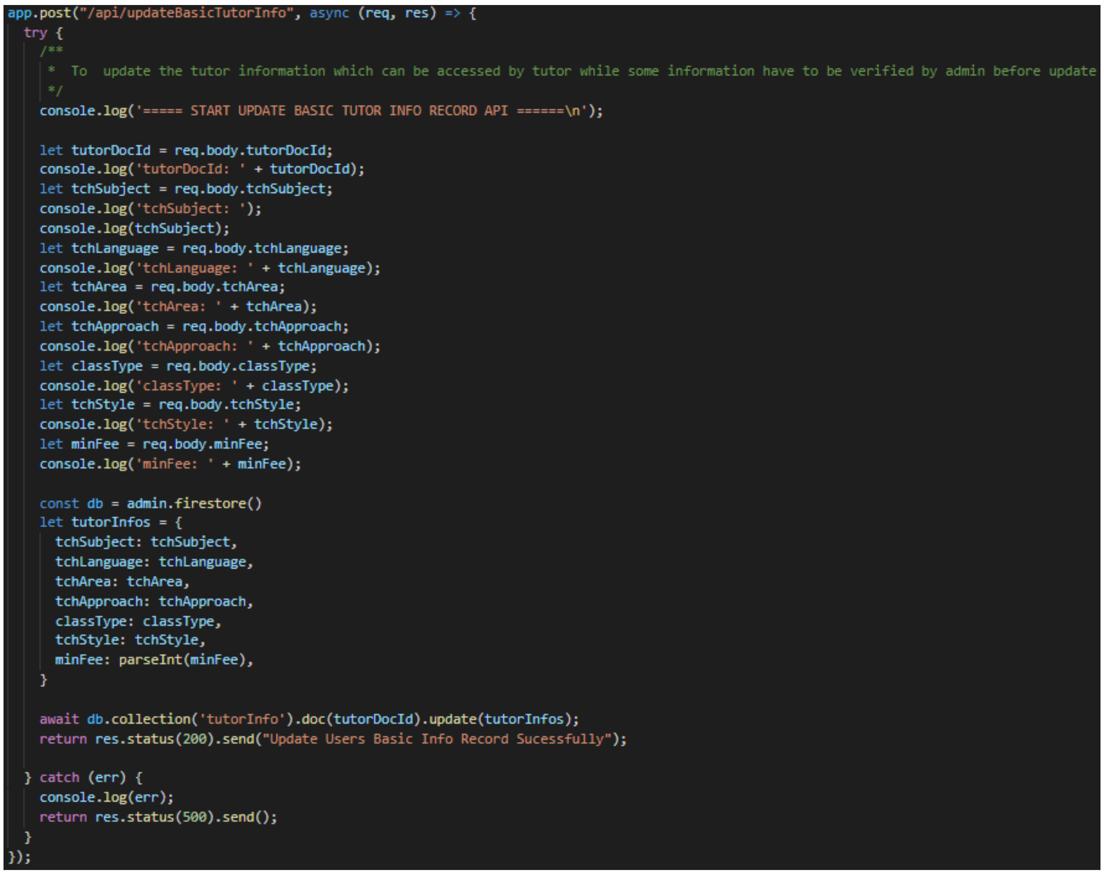
```
_renderPBDCheckbox() {
  let subjects = Object.entries(this.state.PBD);
  return (
    <View style={styles.checkboxInnerBorder}>
      {subjects.map(subject => {
        return (
          <Checkbox.Item
            label={this._subjectMatch(subject[0])}
            status={this.state.PBD[subject[0]] ? 'checked' : 'unchecked'}
            mode={'android'}
            labelStyle={{ fontSize: 14 }}
            color='#00AAAA'
            onPress={() => {
              this.setState({
                PBD: {
                  ...this.state.PBD, [subject[0]]: !this.state.PBD[subject[0]]
                }
              })
            }}
          />

        )
      })}
    </View>
  );
}

_renderPT3Checkbox() {⋯
}

_renderSPMCheckbox() {⋯
}

_renderUECCheckbox() {⋯
}
```

Figure 6.59: Section Code (Front-end) – Edit Tutor Profile Part 2



```
_convertTchSubjectFmt() {
  let tchSubj = {};
  let PBD = Object.entries(this.state.PBD);
  PBD.forEach(subj => {
    console.log('Subject: ' + subj[0] + ' && ' + subj[1]);
    if (subj[1]) {
      if (tchSubj.PBD == null) {
        tchSubj.PBD = [subj[0]];
      } else {
        tchSubj.PBD.push(subj[0]);
      }
    }
  })

  let PT3 = Object.entries(this.state.PT3);
  PT3.forEach(subj => {···
  })

  let SPM = Object.entries(this.state.SPM);
  SPM.forEach(subj => {···
  })

  let UEC = Object.entries(this.state.UEC);
  UEC.forEach(subj => {···
  })
  console.log('tchSubj');
  console.log(tchSubj);
  return tchSubj;
}
```

Figure 6.60: Section Code (Front-end) – Edit Tutor Profile Part 3



```
app.post("/api/updateBasicTutorInfo", async (req, res) => {
  try {
    /**
     *  To  update the tutor information which can be accessed by tutor while some information have to be verified by admin before update
     */
    console.log('===== START UPDATE BASIC TUTOR INFO RECORD API ======\n');

    let tutorDocId = req.body.tutorDocId;
    console.log('tutorDocId: ' + tutorDocId);
    let tchSubject = req.body.tchSubject;
    console.log('tchSubject: ');
    console.log(tchSubject);
    let tchLanguage = req.body.tchLanguage;
    console.log('tchLanguage: ' + tchLanguage);
    let tchArea = req.body.tchArea;
    console.log('tchArea: ' + tchArea);
    let tchApproach = req.body.tchApproach;
    console.log('tchApproach: ' + tchApproach);
    let classType = req.body.classType;
    console.log('classType: ' + classType);
    let tchStyle = req.body.tchStyle;
    console.log('tchStyle: ' + tchStyle);
    let minFee = req.body.minFee;
    console.log('minFee: ' + minFee);

    const db = admin.firestore()
    let tutorInfos = {
      tchSubject: tchSubject,
      tchLanguage: tchLanguage,
      tchArea: tchArea,
      tchApproach: tchApproach,
      classType: classType,
      tchStyle: tchStyle,
      minFee: parseInt(minFee),
    }

    await db.collection('tutorInfo').doc(tutorDocId).update(tutorInfos);
    return res.status(200).send("Update Users Basic Info Record Sucessfully");

  } catch (err) {
    console.log(err);
    return res.status(500).send();
  }
});
```

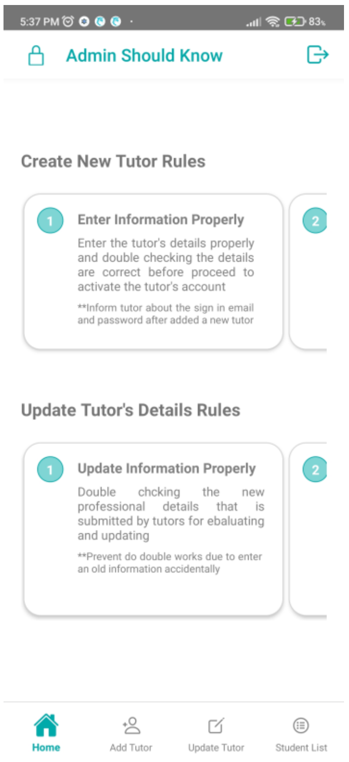
Figure 6.61: Section Code (Server API) – Update Tutor Basic Information

The registered tutor is directed to the edit screen when pressing the “Edit Profile” button. The componentDidMount in the edit profile screen will first check the values in the variable and assign true to the variable boolean if the value matched with the key of boolean as the figure 6.56 in different multiple values variable (array). Then, the checkbox component is rendered according to the boolean in the variable as the ticked box or unticked box (figure 6.57). The format of the values in the checkbox is converted into the original format of the variable when the “Save” button is pressed and ready to call the "/api/updateBasicProfile" and "/api/updateBasicTutorInfo" APIs to update the ‘users’ document and ‘tutorInfo’ document in Firestore. In addition, the upload photo activity is involved in this activity which is explained in the above section.

6.3.5 Admin Side

6.3.5.1 Home Activities

The home activities on the admin side only applied the AsyncStorage in storing the admin’s information while also the sign-out activity is accessible on the home screen. The activities are explained in the above section.



Create New Tutor Rules

1 Enter Information Properly

Enter the tutor's details properly and double checking the details are correct before proceed to activate the tutor's account

**Inform tutor about the sign in email and password after added a new tutor

2

Update Tutor's Details Rules

1 Update Information Properly

Double chcking the new professional details that is submitted by tutors for ebaluating and updating

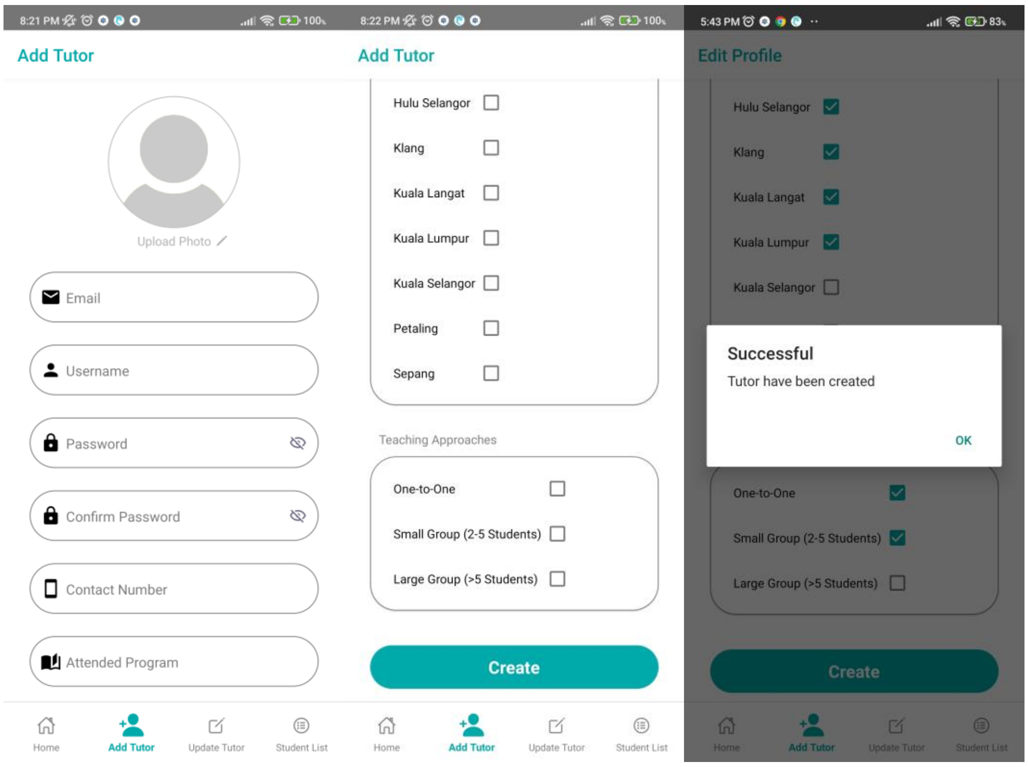
**Prevent do double works due to enter an old information accidentally

2

Figure 6.62: Home (Admin)

6.3.5.2 Add Tutor Activities

The add tutor activities could be considered as the combination of upload photo activity, modified edit profile activity (front-end), and sign-up activity. The front-end of the edit profile activity checks the existing values of a variable and renders the checkbox with tick and untick while the checkbox that is rendered in add tutor activities is blank and set all as unticked as default.



Hulu Selangor ☐
Klang ☐
Kuala Langat ☐
Kuala Lumpur ☐
Kuala Selangor ☐
Petaling ☐
Sepang ☐

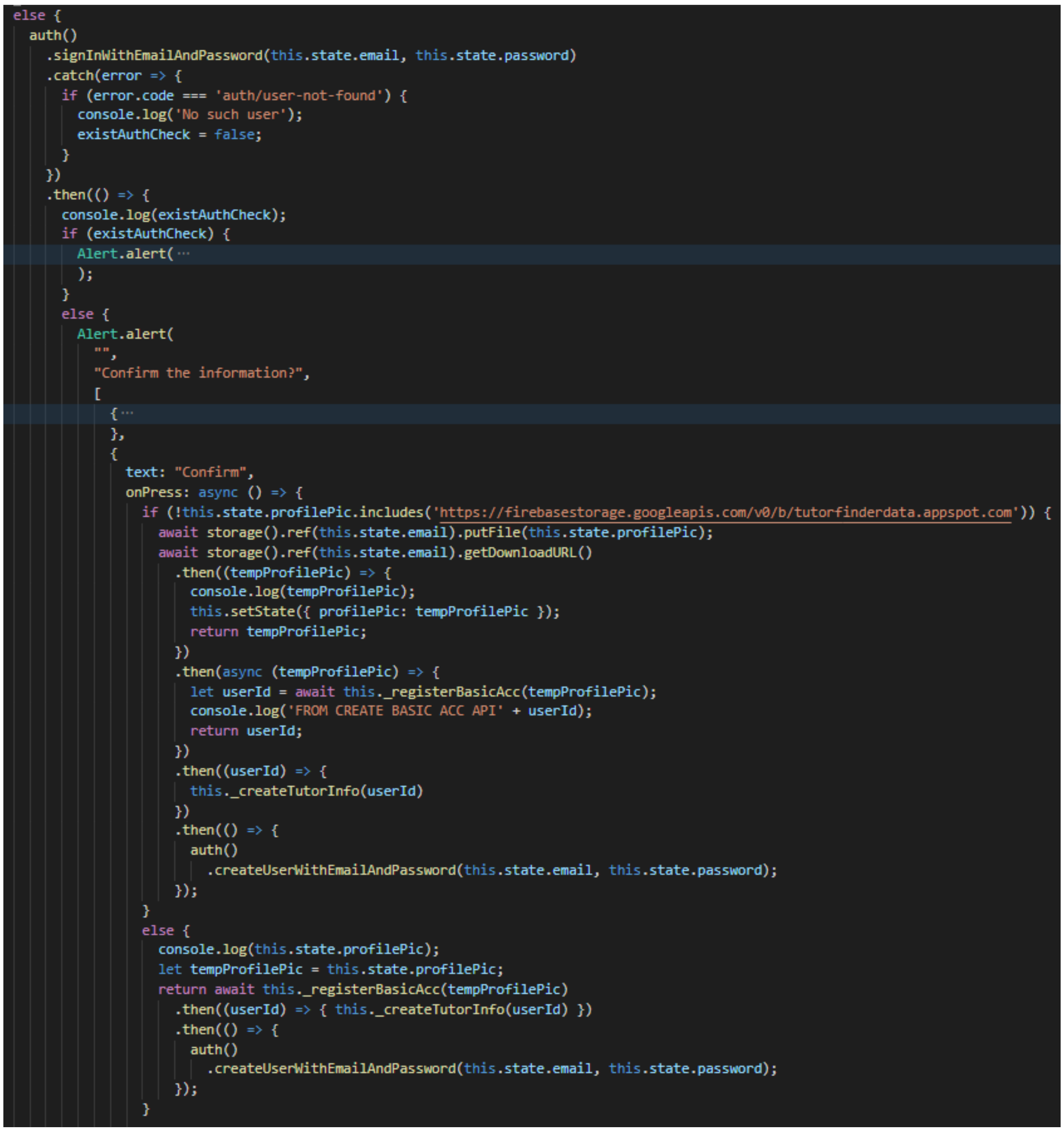
Teaching Approaches
One-to-One ☐
Small Group (2-5 Students) ☐
Large Group (>5 Students) ☐

Hulu Selangor ✓
Klang ✓
Kuala Langat ✓
Kuala Lumpur ✓
Kuala Selangor ☐

Successful
Tutor have been created

One-to-One ✓
Small Group (2-5 Students) ✓
Large Group (>5 Students) ☐

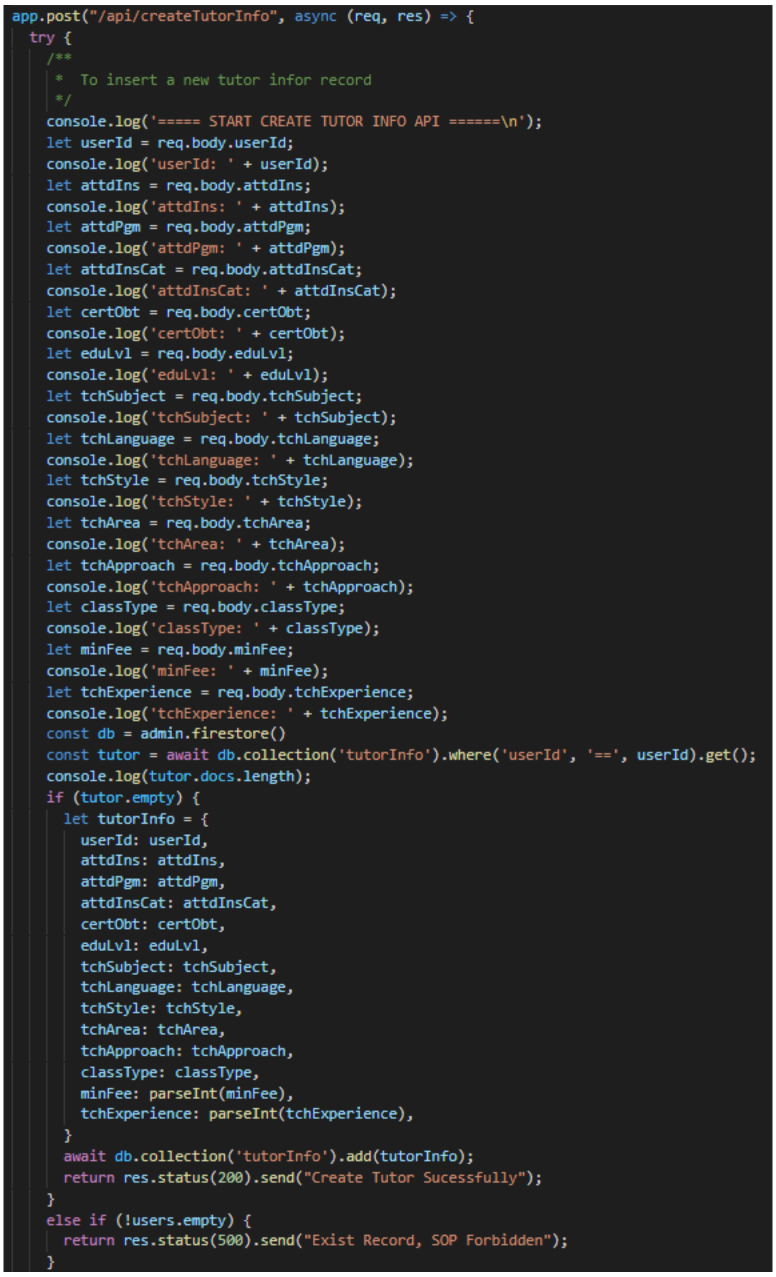
Figure 6.63: Add Tutor



```
else {
  auth()
    .signInWithEmailAndPassword(this.state.email, this.state.password)
    .catch(error => {
      if (error.code === 'auth/user-not-found') {
        console.log('No such user');
        existAuthCheck = false;
      }
    })
    .then(() => {
      console.log(existAuthCheck);
      if (existAuthCheck) {
        Alert.alert(···
        );
      }
      else {
        Alert.alert(
          "",
          "Confirm the information?",
          [
            {···
            },
            {
              text: "Confirm",
              onPress: async () => {
                if (!this.state.profilePic.includes('https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com')) {
                  await storage().ref(this.state.email).putFile(this.state.profilePic);
                  await storage().ref(this.state.email).getDownloadURL()
                    .then((tempProfilePic) => {
                      console.log(tempProfilePic);
                      this.setState({ profilePic: tempProfilePic });
                      return tempProfilePic;
                    })
                    .then(async (tempProfilePic) => {
                      let userId = await this._registerBasicAcc(tempProfilePic);
                      console.log('FROM CREATE BASIC ACC API' + userId);
                      return userId;
                    })
                    .then((userId) => {
                      this._createTutorInfo(userId)
                    })
                    .then(() => {
                      auth()
                        .createUserWithEmailAndPassword(this.state.email, this.state.password);
                    });
                }
                else {
                  console.log(this.state.profilePic);
                  let tempProfilePic = this.state.profilePic;
                  return await this._registerBasicAcc(tempProfilePic)
                    .then((userId) => { this._createTutorInfo(userId) })
                    .then(() => {
                      auth()
                        .createUserWithEmailAndPassword(this.state.email, this.state.password);
                    });
                }
```

Figure 6.64: Section Code (Front-end) – Create Tutor Information

The flow in adding a new tutor is similar to adding a new tutor seeker (sign up). The sign-in function in Firebase Authentication is applied to check the existence of the account, then proceed to upload the profile picture to Firebase Storage followed by the APIs calls which are the "/api/createTutorInfo" and "/api/registerBasicAcc" API. If all status is returned from the server, then create the authenticated account by using the Firebase Authentication.

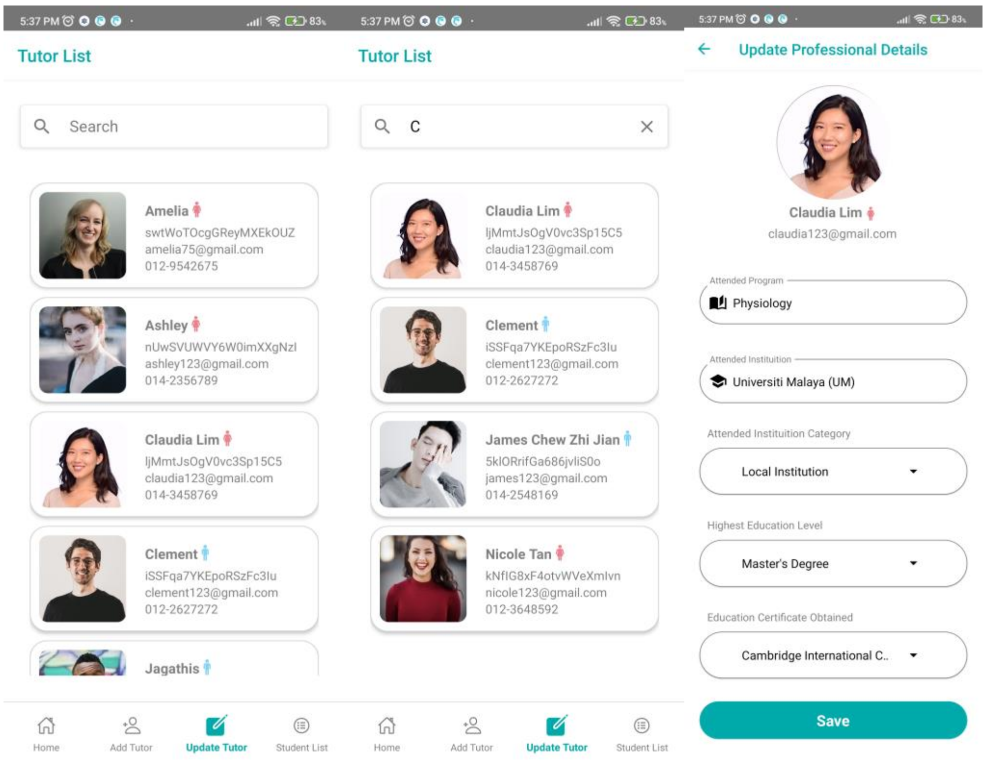


```
app.post("/api/createTutorInfo", async (req, res) => {
  try {
    /**
     *  To insert a new tutor infor record
     */
    console.log('===== START CREATE TUTOR INFO API ======\n');
    let userId = req.body.userId;
    console.log('userId: ' + userId);
    let attdIns = req.body.attdIns;
    console.log('attdIns: ' + attdIns);
    let attdPgm = req.body.attdPgm;
    console.log('attdPgm: ' + attdPgm);
    let attdInsCat = req.body.attdInsCat;
    console.log('attdInsCat: ' + attdInsCat);
    let certObt = req.body.certObt;
    console.log('certObt: ' + certObt);
    let eduLvl = req.body.eduLvl;
    console.log('eduLvl: ' + eduLvl);
    let tchSubject = req.body.tchSubject;
    console.log('tchSubject: ' + tchSubject);
    let tchLanguage = req.body.tchLanguage;
    console.log('tchLanguage: ' + tchLanguage);
    let tchStyle = req.body.tchStyle;
    console.log('tchStyle: ' + tchStyle);
    let tchArea = req.body.tchArea;
    console.log('tchArea: ' + tchArea);
    let tchApproach = req.body.tchApproach;
    console.log('tchApproach: ' + tchApproach);
    let classType = req.body.classType;
    console.log('classType: ' + classType);
    let minFee = req.body.minFee;
    console.log('minFee: ' + minFee);
    let tchExperience = req.body.tchExperience;
    console.log('tchExperience: ' + tchExperience);
    const db = admin.firestore()
    const tutor = await db.collection('tutorInfo').where('userId', '==', userId).get();
    console.log(tutor.docs.length);
    if (tutor.empty) {
      let tutorInfo = {
        userId: userId,
        attdIns: attdIns,
        attdPgm: attdPgm,
        attdInsCat: attdInsCat,
        certObt: certObt,
        eduLvl: eduLvl,
        tchSubject: tchSubject,
        tchLanguage: tchLanguage,
        tchStyle: tchStyle,
        tchArea: tchArea,
        tchApproach: tchApproach,
        classType: classType,
        minFee: parseInt(minFee),
        tchExperience: parseInt(tchExperience),
      }
      await db.collection('tutorInfo').add(tutorInfo);
      return res.status(200).send("Create Tutor Sucessfully");
    }
    else if (!users.empty) {
      return res.status(500).send("Exist Record, SOP Forbidden");
    }
```

Figure 6.65: Section Code (Server API) – Create Tutor Information

6.3.5.3 Update Tutor Activities

The update tutor stack is used to update the professional information of the tutors after the professional information that is submitted by the registered tutor is verified by the admin.



Tutor List

Search

Amelia
swtWoTOcgGReyMXEkOUZ
amelia75@gmail.com
012-9542675

Ashley
nUwSVUWVY6W0imXXgNzl
ashley123@gmail.com
014-2356789

Claudia Lim
ljMmtJsOgV0vc3Sp15C5
claudia123@gmail.com
014-3458769

Clement
iSSFqa7YKEpoRSzFc3lu
clement123@gmail.com
012-2627272

Jagathis

Tutor List

C

Claudia Lim
ljMmtJsOgV0vc3Sp15C5
claudia123@gmail.com
014-3458769

Clement
iSSFqa7YKEpoRSzFc3lu
clement123@gmail.com
012-2627272

James Chew Zhi Jian
5klORrifGa686jvliS0o
james123@gmail.com
014-2548169

Nicole Tan
kNfIG8xF4otvWVeXmlvn
nicole123@gmail.com
012-3648592

Update Professional Details

Claudia Lim
claudia123@gmail.com

Attended Program: Physiology

Attended Institution: Universiti Malaya (UM)

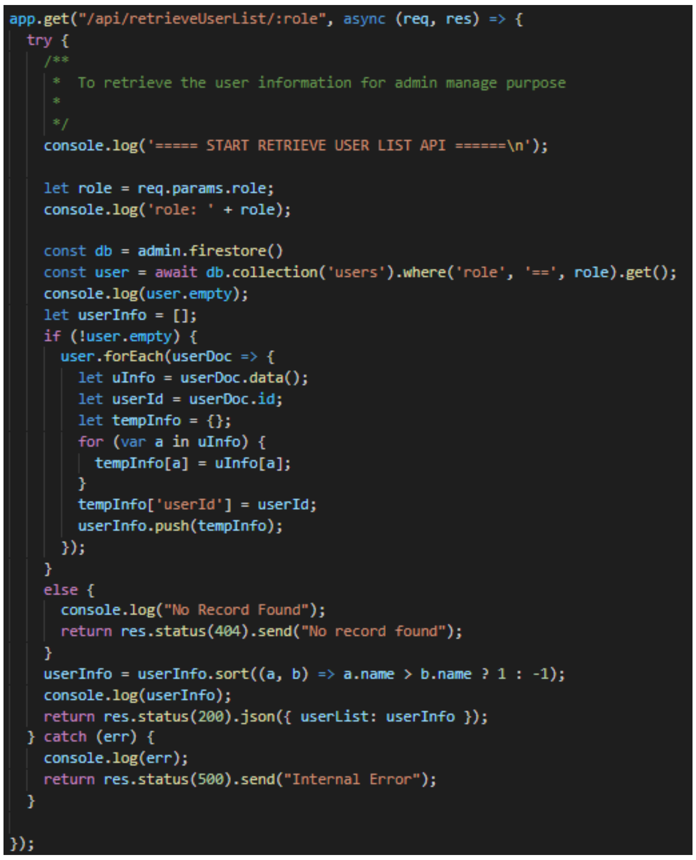
Attended Institution Category: Local Institution

Highest Education Level: Master's Degree

Education Certificate Obtained: Cambridge International C..

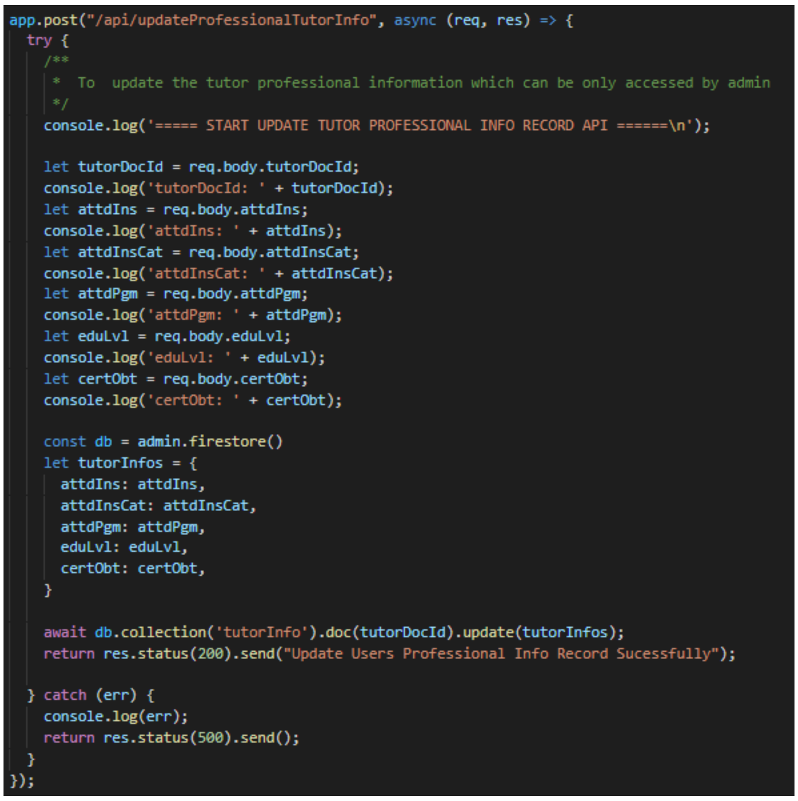
Figure 6.66: Update Tutor

The "/api/retrieveChatUserList/:userId" API is called when didFocus on the update tutor tab to retrieve the user list with basic information according to the role which is student or tutor for displaying purpose. The admin press the selected tutor display column to navigate to tutor edit screen. The information is get from the previous screen by using getParam function while the "/api/updateProfessionalTutorInfo" API is triggered when the “Save” button is pressed. The API updates the matched ‘tutorInfo’ document and the status will be returned to the front-end.



```
app.get("/api/retrieveUserList/:role", async (req, res) => {
  try {
    /**
     *  To retrieve the user information for admin manage purpose
     *
     */
    console.log('===== START RETRIEVE USER LIST API ======\n');

    let role = req.params.role;
    console.log('role: ' + role);

    const db = admin.firestore()
    const user = await db.collection('users').where('role', '==', role).get();
    console.log(user.empty);
    let userInfo = [];
    if (!user.empty) {
      user.forEach(userDoc => {
        let uInfo = userDoc.data();
        let userId = userDoc.id;
        let tempInfo = {};
        for (var a in uInfo) {
          tempInfo[a] = uInfo[a];
        }
        tempInfo['userId'] = userId;
        userInfo.push(tempInfo);
      });
    }
    else {
      console.log("No Record Found");
      return res.status(404).send("No record found");
    }
    userInfo = userInfo.sort((a, b) => a.name > b.name ? 1 : -1);
    console.log(userInfo);
    return res.status(200).json({ userList: userInfo });
  } catch (err) {
    console.log(err);
    return res.status(500).send("Internal Error");
  }

});
```

Figure 6.67: Section Code (Server API) – Retrieve User List

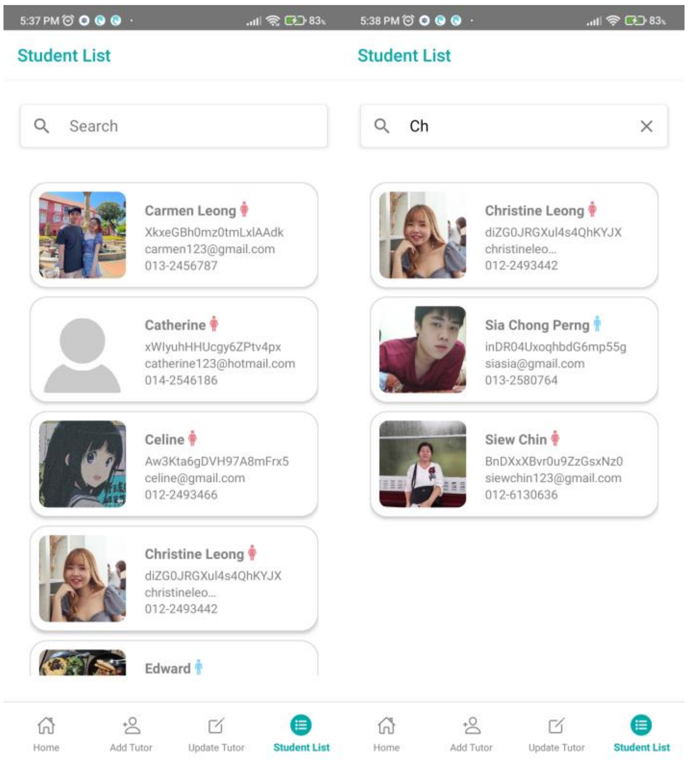


```
app.post("/api/updateProfessionalTutorInfo", async (req, res) => {
  try {
    /**
     *  To  update the tutor professional information which can be only accessed by admin
     */
    console.log('===== START UPDATE TUTOR PROFESSIONAL INFO RECORD API ======\n');

    let tutorDocId = req.body.tutorDocId;
    console.log('tutorDocId: ' + tutorDocId);
    let attdIns = req.body.attdIns;
    console.log('attdIns: ' + attdIns);
    let attdInsCat = req.body.attdInsCat;
    console.log('attdInsCat: ' + attdInsCat);
    let attdPgm = req.body.attdPgm;
    console.log('attdPgm: ' + attdPgm);
    let eduLvl = req.body.eduLvl;
    console.log('eduLvl: ' + eduLvl);
    let certObt = req.body.certObt;
    console.log('certObt: ' + certObt);

    const db = admin.firestore()
    let tutorInfos = {
      attdIns: attdIns,
      attdInsCat: attdInsCat,
      attdPgm: attdPgm,
      eduLvl: eduLvl,
      certObt: certObt,
    }

    await db.collection('tutorInfo').doc(tutorDocId).update(tutorInfos);
    return res.status(200).send("Update Users Professional Info Record Sucessfully");

  } catch (err) {
    console.log(err);
    return res.status(500).send();
  }
});
```

Figure 6.68: Section Code (Server API) – Update Professional Tutor Info

6.3.5.4 User List Activities

The "/api/retrieveChatUserList/:userId" API is called when didFocus on the student list tab to retrieve the user list with basic information according to the role which is student or tutor for displaying purpose. The API is explained in the above section.



Carmen Leong
XkxeGBh0mz0tmLxlAAdk
carmen123@gmail.com
013-2456787

Catherine
xWIyuhHHUcgy6ZPtv4px
catherine123@hotmail.com
014-2546186

Celine
Aw3Kta6gDVH97A8mFrx5
celine@gmail.com
012-2493466

Christine Leong
diZG0JRGXul4s4QhKYJX
christineleo...
012-2493442

Edward

Christine Leong
diZG0JRGXul4s4QhKYJX
christineleo...
012-2493442

Sia Chong Perng
inDR04UxoqhbdG6mp55g
siasia@gmail.com
013-2580764

Siew Chin
BnDXxXBvr0u9ZzGsxNz0
siewchin123@gmail.com
012-6130636

Figure 6.69: User List

CHAPTER 7

SYSTEM TESTING

7.1 Introduction

The unit testing, API testing, usability testing, and User Acceptance Testing (UAT) are conducted to test the fulfilment of functional and non-functional requirements of this application. Automation testing is not applied in this application due to this application is considered a small project that involves only one contributor without a continuous integration process that needs automation testing.

7.2 Unit Testing

The unit testing is completed during the development process manually to ensure the functionality of each activity in meeting the requirement specifications. The test case with results is shown below.

Table 7.1: Unit Testing – Log In Account

Test Case ID	UNT-001	**Test Case Title**	Log In Account	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Log in with a valid email and password	1. Enter registered email and password. 2. Press the “Sign In” button.	1. Log in successfully and navigate to the home screen.	1. Log in successfully and navigate to the home screen.	Pass
Log in with an invalid email or password	1. Enter unregistered email or wrong password. 2. Press the “Sign In” button.	1. System display alert message “Incorrect email or password, please try again”.	1. System display alert message “Incorrect email or password, please try again”.	Pass
Log in with empty input	1. Press the “Sign In” button	1. System display alert message “Please in both email and password to login”.	1. System display alert message “Please in both email and password to log in”.	Pass
Log in with an invalid format email	1. Enter invalid email format and password. 2. Press the “Sign In” button.	1. System display alert message “Invalid email.”.	1. System display alert message “Invalid email.”.	Pass

Table 7.2: Unit Testing – Sign Up Account

Test Case ID	UNT-002	**Test Case Title**	Sign Up Account		
Test Case Description	**Execution Steps**		**Expected Result**	**Actual Result**	**Status**
Sign up with a valid input	1. Upload profile picture. 2. Enter valid format of username, password, confirm password, email, contact number, and gender. 3. Press the "Sign Up" button.		1. System display alert message "sign up successfully" and navigate to the login screen.	1. System display alert message "sign up successfully" and navigate to the login screen.	Pass
Sign up with invalid input	1. Upload profile picture. 2. Enter invalid format of username, password, confirm password, email, contact number, or gender. 3. Press the "Sign Up" button.		1. System display alert message " Please ensure the required fields in a valid format".	1. System display alert message " Please ensure the required fields in a valid format".	Pass
Sign up with empty input	1. Upload profile picture. 2. Missing of username, password, confirm password, email or contact number. 3. Press the "Sign Up" button.		1. System display alert message "Please fill in all required fields".	1. System display alert message "Please fill in all required fields".	Pass

Table 7.3: Unit Testing – Search Tutor

Test Case ID	UNT-003	**Test Case Title**	Search Tutor	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Search tutor with mandatory field selected	1. Select the mandatory search fields and other optional search fields. 2. Press the "Search" button. 3. Press result 1/2/3 randomly.	1. The system navigates to the search result screen and displays the result accordingly	1. The system navigates to the search result screen and displays the result accordingly	Pass
Search tutor with mandatory field unselected	1. Select the other optional search fields, the mandatory search field is empty. 2. Press the "Search" button.	1. The system displays the alert message " Please fill in required fields as minimum search criteria"	1. The system displays the alert message " Please fill in required fields as minimum search criteria"	Pass

Table 7.4: Unit Testing – Send Message

Test Case ID	UNT-004	**Test Case Title**	Send Message	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Send chat message	1. View profile and press on "Chat with tutor/student". 2. Enter message and press send icon	1. The system sends chat and a push notification is displayed.	1. The system sends chat and a push notification is displayed.	Pass

Table 7.5: Unit Testing – Send Demo Class Request

Test Case ID	UNT-005	**Test Case Title**	Send Demo Class Request		
Test Case Description	**Execution Steps**		**Expected Result**	**Actual Result**	**Status**
Send a new request with mandatory fields selected	1. Press the "Free Demo Class" button. 2. Select mandatory fields. 3. Press "Submit" button.		1. The system displays alert message "Demo class is registered and wait for approving"	1. The system displays alert message "Demo class is registered and wait for approving"	Pass
Send the same request with mandatory fields selected	1. Press the "Free Demo Class" button. 2. Select mandatory fields that are selected before. 3. Press "Submit" button.		1. The system displays alert message "You have raised this request before, kindly wait for approving".	1. The system displays alert message "You have raised this request before, kindly wait for approving".	Pass
Send a request to the same tutor for the third time.	1. Press the "Free Demo Class" button. 2. Select mandatory fields. 3. Press "Submit" button. 4. Repeat for 3 times.		1. The system displays alert message "Your demo class requests are more than two, kindly choose another tutor".	1. The system displays alert message "Your demo class requests are more than two, kindly choose another tutor".	Pass

Table 7.6: Unit Testing – Book Formal Class

Test Case ID	UNT-006	**Test Case Title**	Book Formal Class	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Send a new request with mandatory fields selected	1. Press the "Book Formal Class" button. 2. Select mandatory fields and enter ideal fee. 3. Press "Submit" button.	1. The system displays alert message "Formal class is registered and tutor will approach you within 48 hours"	1. The system displays alert message "Formal class is registered and tutor will approach you within 48 hours"	Pass
Send the same request with mandatory fields selected	1. Press the "Book Formal Class" button. 2. Select same mandatory fields and enter ideal fee. 3. Press "Submit" button.	1. The system displays alert message " You have raised same request before, kindly approach the tutor if he/she do not approach you within 48 hours".	1. The system displays alert message " You have raised same request before, kindly approach the tutor if he/she do not approach you within 48 hours".	Pass

Table 7.7: Unit Testing – Rate and Review

Test Case ID	UNT-007	Test Case Title	Rate and Review	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Give rate and review to tutor	1. View history of requested class and press "Rate and Review" button. 2. Enter rate and review. 3. Press "Submit"	1. The system displays alert message "Thanks for giving rate and review. Your rate and review will be displayed at the tutor's" and back to the previous screen	1. The system displays alert message "Thanks for giving rate and review. Your rate and review will be displayed at the tutor's profile" and back to the previous screen	Pass

Table 7.8: Unit Testing – Delete Account

Test Case ID	UNT-008	Test Case Title	Delete Account	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Delete existing account	1. Press "Delete Account" button and make confirmation.	1. The system displays alert message "Account have been deleted" and navigate to sign in screen.	1. The system displays alert message "Account have been deleted" and navigate to sign in screen.	Pass

Table 7.9: Unit Testing – Edit Profile

Test Case ID	UNT-009	**Test Case Title**	Edit Profile	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Update profile details with all valid fields entered	1. Upload new profile picture and modify profile details.	1. The system displays alert message "Profile have been updated".	1. The system displays alert message "Profile have been updated".	Pass

Table 7.10: Unit Testing – Manage Demo Class

Test Case ID	UNT-010	**Test Case Title**	Manage Demo Class	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Update request status	1. View the requested list. 2. Press accept or reject button.	1. The system updates the request status and display the latest status of request.	1. The system updates the request status and display the latest status of request.	Pass

Table 7.11: Unit Testing – View Formal Class

Test Case ID	UNT-011	**Test Case Title**	View Formal Class	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
View formal class booking	1. Press "formal class" navigation button	1. The system display booked formal class list.	1. The system display booked formal class list.	Pass

Table 7.12: Unit Testing – Add Tutor

Test Case ID	UNT-012	**Test Case Title**	Add Tutor	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Add new tutor with valid details data	1. Enter all fields with valid format of data. 2. Press “Create” button. 3. Press “Yes” to confirm entered information	1. The system displays alert message “Tutor have been created” and navigate to update tutor screen.	1. The system displays alert message “Tutor have been created” and navigate to update tutor screen.	Pass
Add new tutor with invalid details data.	1. Enter invalid fields with invalid format of data. 2. Press “Create” button.	1. The system displays alert message “Missing fields or invalid format fields exist”.	1. The system displays alert message “Missing fields or invalid format fields exist”.	Pass
Add new tutor with missing fields.	1. Enter details with some missing fields 2. Press “Create” button.	1. The system displays alert message “Missing fields or invalid format fields exist”.	1. The system displays alert message “Missing fields or invalid format fields exist”.	Pass
Add existing tutor	1. Enter details with existing email. 2. Press “Create” button.	1. The system displays alert message “This email have been registered, please use another email”.	1. The system displays alert message “ This email have been registered, please use another email”.	Pass

Table 7.13: Unit Testing – Update Tutor’s Professional Detail

Test Case ID	UNT-013	**Test Case Title**	Update Tutor’s Professional Detail	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
Update tutor’s professional detail with all fields filled in	1. Select tutor and view existing data. 2. Enter and select the verified professional data. 3. Press “Save” button.	1. The system displays alert message “ Profile have been updated” and navigate to previous screen.	1. The system displays alert message “ Profile have been updated” and navigate to previous screen.	Pass
Update tutor’s professional detail with some missing fields.	1. Select tutor and view existing data. 2. Delete some existing data. 3. Press “Save” button.	2. The system displays alert message “All fields should have value and minimum one selected item”.	2. The system displays alert message “All fields should have value and minimum one selected item”.	Pass

Table 7.14: Unit Testing – View Tutor Seeker List

Test Case ID	UNT-014	**Test Case Title**	View Tutor Seeker List	
Test Case Description	**Execution Steps**	**Expected Result**	**Actual Result**	**Status**
View tutor seeker list	1. Enter name to search related tutor seeker.	1. The system displays related tutor seeker list.	1. The system displays related tutor seeker list.	Pass

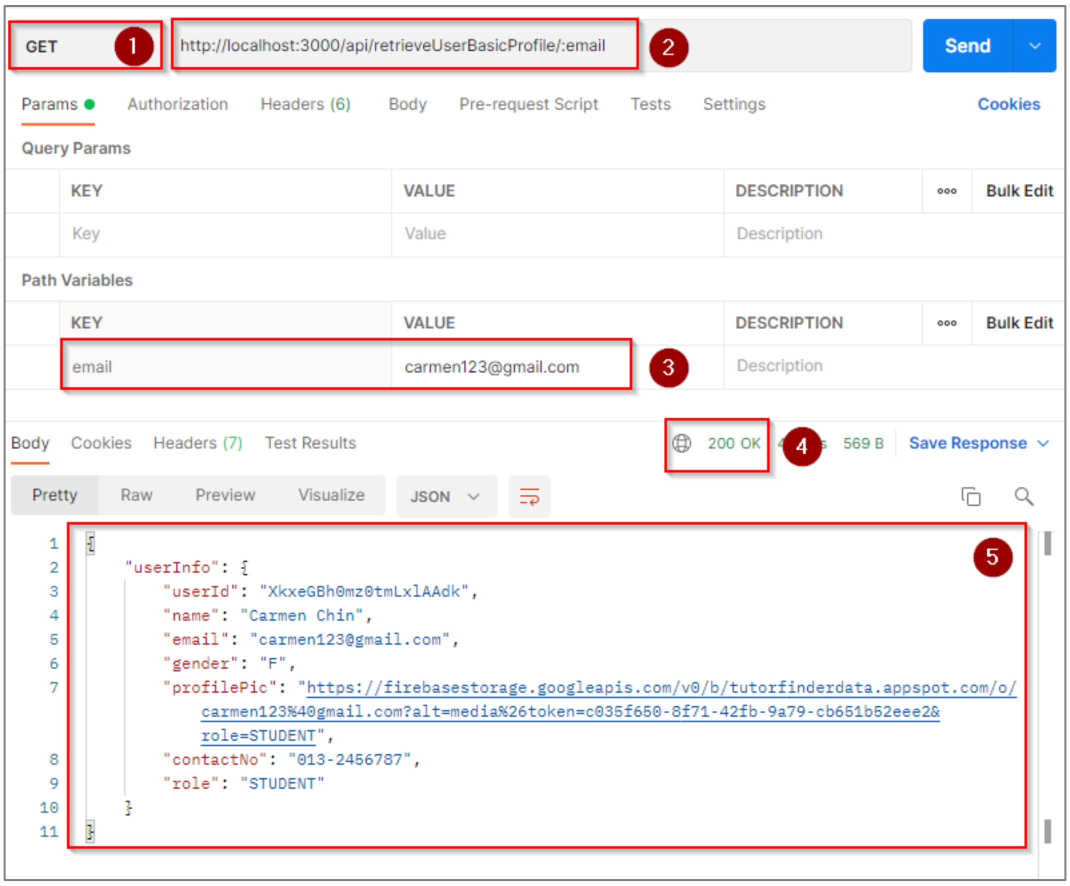
7.3 Application Programming Interface (API) Testing

The majority of the function in this application involved the API in the server (Express.js). Therefore, API testing is conducted to verify the functionality of each API in the server is complete and accurate by using Postman.

Figure 7.15 is labelled with numbers to show the API testing information located within the screenshot. The testing information in the figure within this section is displayed in the same way.

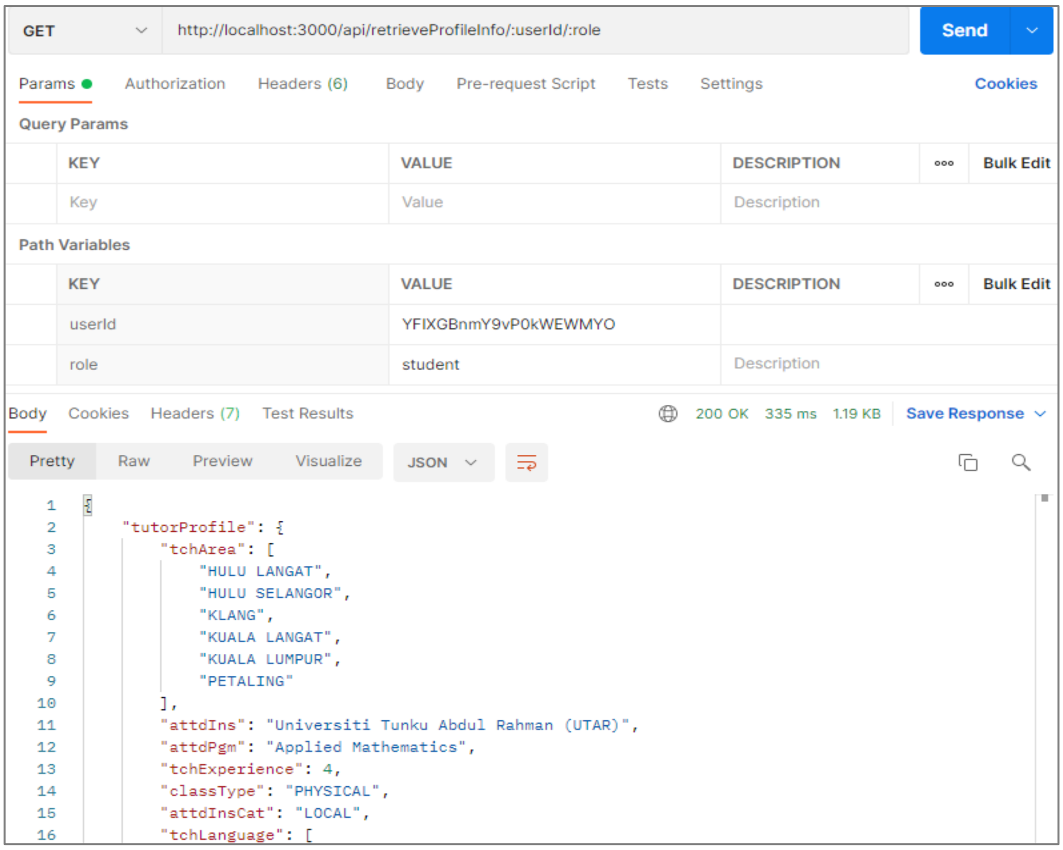
Table 7.15: Testing Information Numbering

Number	Testing Information
1	API method (GET/POST)
2	API route
3	Parameters (GET method)/ Raw body (POST method)
4	Response status (200 – success, 4XX – client error)
5	Response in JSON format



```
{
    "userInfo": {
        "userId": "XkxeGBh0mz0tmLxlAAdk",
        "name": "Carmen Chin",
        "email": "carmen123@gmail.com",
        "gender": "F",
        "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/o/
            carmen123%40gmail.com?alt=media%26token=c035f650-8f71-42fb-9a79-cb651b52eee2&
            role=STUDENT",
        "contactNo": "013-2456787",
        "role": "STUDENT"
    }
}
```

Figure 7.1: API Testing – Retrieve User Basic Profile



GET http://localhost:3000/api/retrieveProfileInfo/:userId/:role

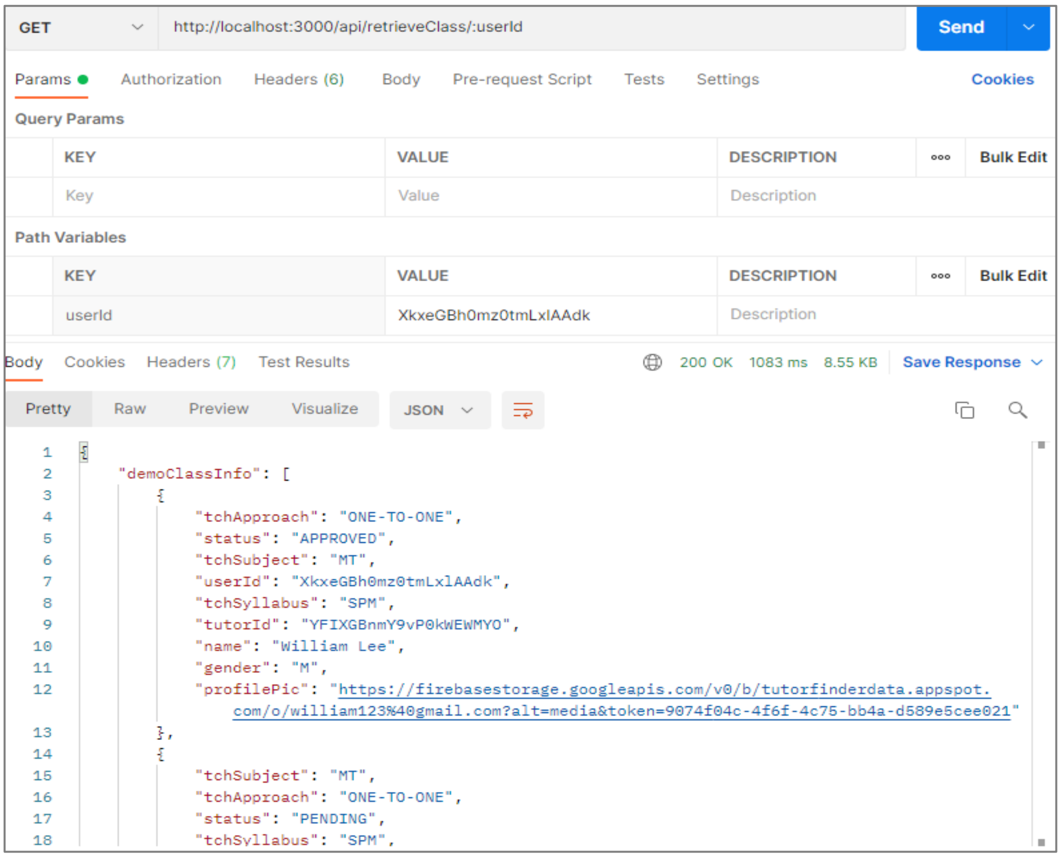
Path Variables

KEY	VALUE	DESCRIPTION
userId	YFIXGBnmY9vP0kWEWMYO	
role	student	Description

200 OK 335 ms 1.19 KB

```
{
    "tutorProfile": {
        "tchArea": [
            "HULU LANGAT",
            "HULU SELANGOR",
            "KLANG",
            "KUALA LANGAT",
            "KUALA LUMPUR",
            "PETALING"
        ],
        "attdIns": "Universiti Tunku Abdul Rahman (UTAR)",
        "attdPgm": "Applied Mathematics",
        "tchExperience": 4,
        "classType": "PHYSICAL",
        "attdInsCat": "LOCAL",
        "tchLanguage": [
```

Figure 7.2: API Testing – Retrieve Profile Info



GET http://localhost:3000/api/retrieveClass/:userId

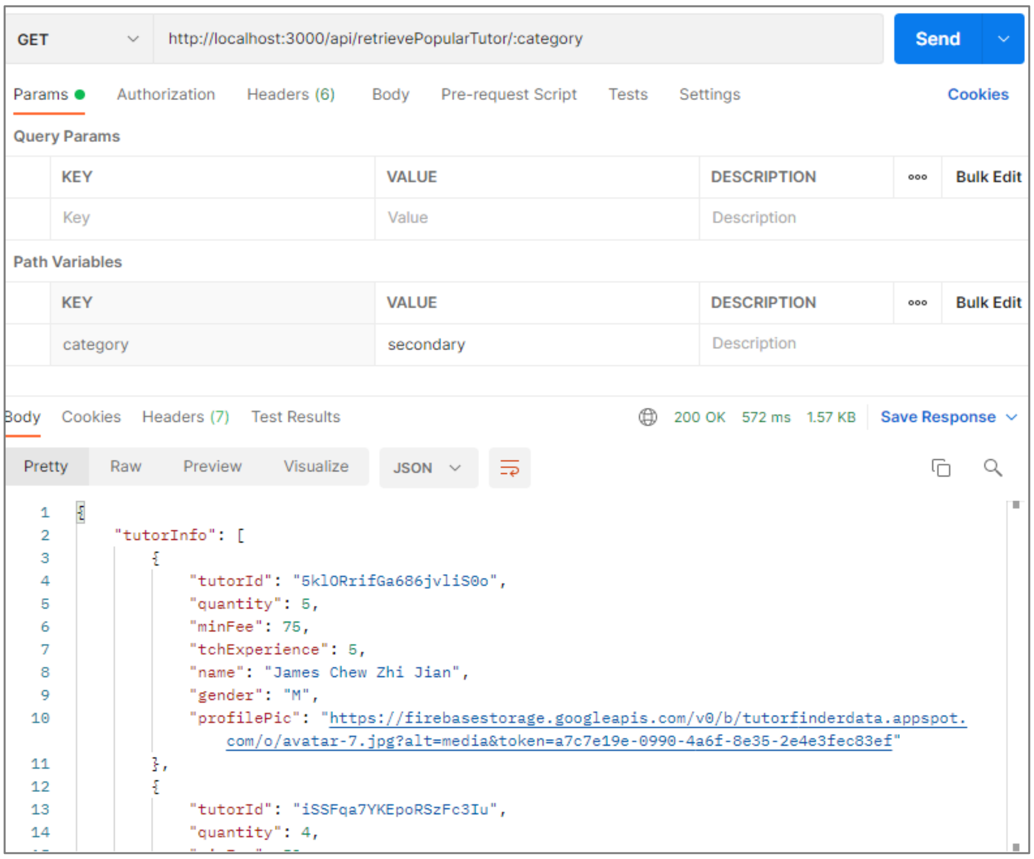
Path Variables

KEY	VALUE	DESCRIPTION
userId	XkxeGBh0mz0tmLxlAAdk	Description

200 OK 1083 ms 8.55 KB

```
{
    "demoClassInfo": [
        {
            "tchApproach": "ONE-TO-ONE",
            "status": "APPROVED",
            "tchSubject": "MT",
            "userId": "XkxeGBh0mz0tmLxlAAdk",
            "tchSyllabus": "SPM",
            "tutorId": "YFIXGBnmY9vP0kWEWMYO",
            "name": "William Lee",
            "gender": "M",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.
                com/o/william123%40gmail.com?alt=media&token=9074f04c-4f6f-4c75-bb4a-d589e5cee021"
        },
        {
            "tchSubject": "MT",
            "tchApproach": "ONE-TO-ONE",
            "status": "PENDING",
            "tchSyllabus": "SPM",
```

Figure 7.3: API Testing – Retrieve Class



GET http://localhost:3000/api/retrievePopularTutor/:category

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

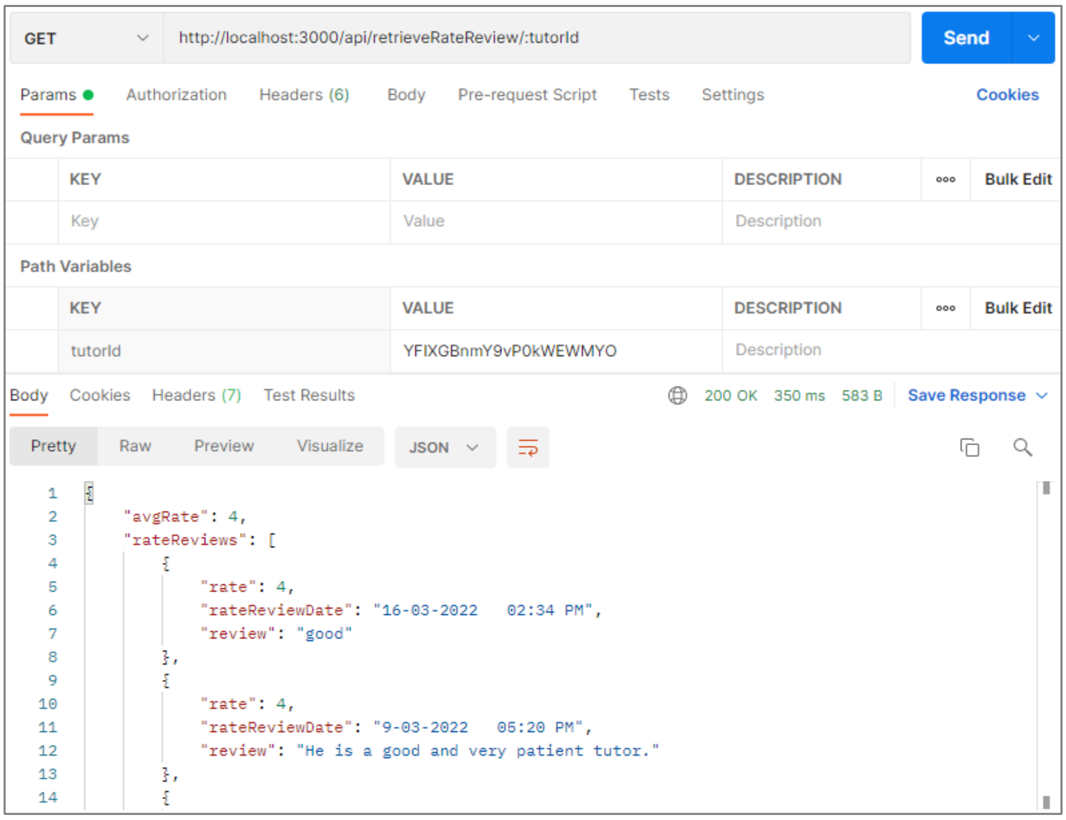
KEY	VALUE	DESCRIPTION
category	secondary	Description

Body Cookies Headers (7) Test Results — 200 OK 572 ms 1.57 KB Save Response

Pretty Raw Preview Visualize JSON

```
{
    "tutorInfo": [
        {
            "tutorId": "5kl0RrifGa686jvliS0o",
            "quantity": 5,
            "minFee": 75,
            "tchExperience": 5,
            "name": "James Chew Zhi Jian",
            "gender": "M",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.
                com/o/avatar-7.jpg?alt=media&token=a7c7e19e-0990-4a6f-8e35-2e4e3fec83ef"
        },
        {
            "tutorId": "iSSFqa7YKEpoRSzFc3Iu",
            "quantity": 4,
```

Figure 7.4: API Testing – Retrieve Popular Tutor



GET http://localhost:3000/api/retrieveRateReview/:tutorId

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

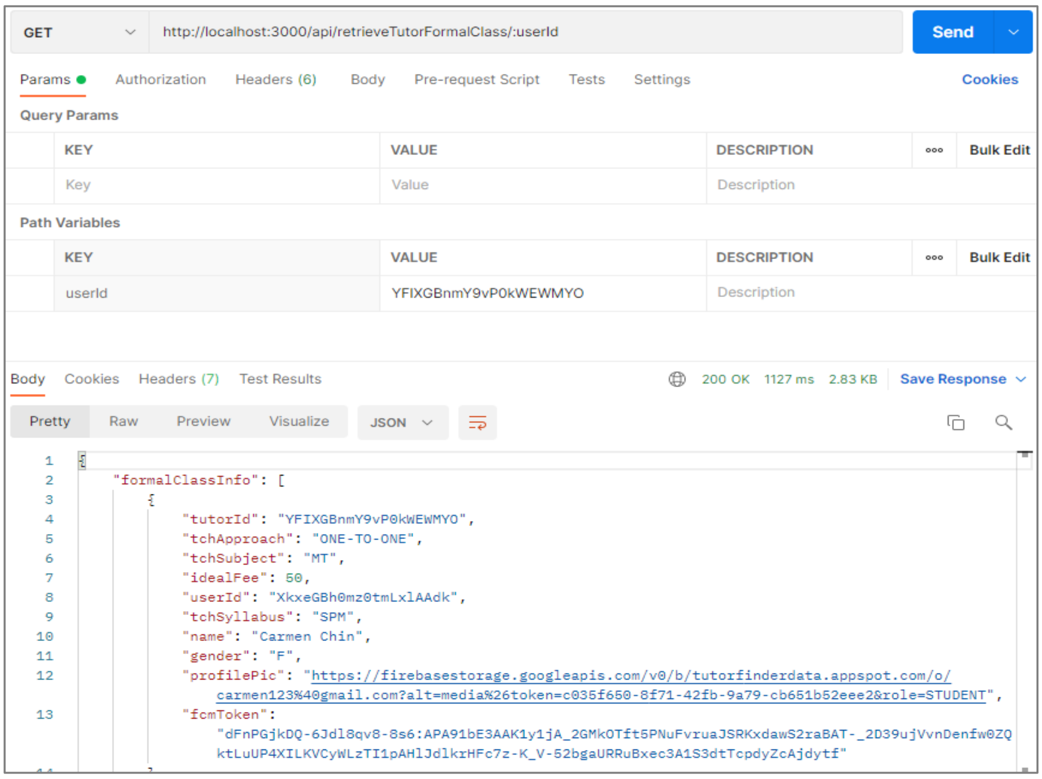
KEY	VALUE	DESCRIPTION
tutorId	YFIXGBnmY9vP0kWEWMYO	Description

Body Cookies Headers (7) Test Results — 200 OK 350 ms 583 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "avgRate": 4,
    "rateReviews": [
        {
            "rate": 4,
            "rateReviewDate": "16-03-2022   02:34 PM",
            "review": "good"
        },
        {
            "rate": 4,
            "rateReviewDate": "9-03-2022   05:20 PM",
            "review": "He is a good and very patient tutor."
        },
        {
```

Figure 7.5: API Testing – Retrieve Rate Review



GET http://localhost:3000/api/retrieveTutorFormalClass/:userId

Params ● | Authorization | Headers (6) | Body | Pre-request Script | Tests | Settings | Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

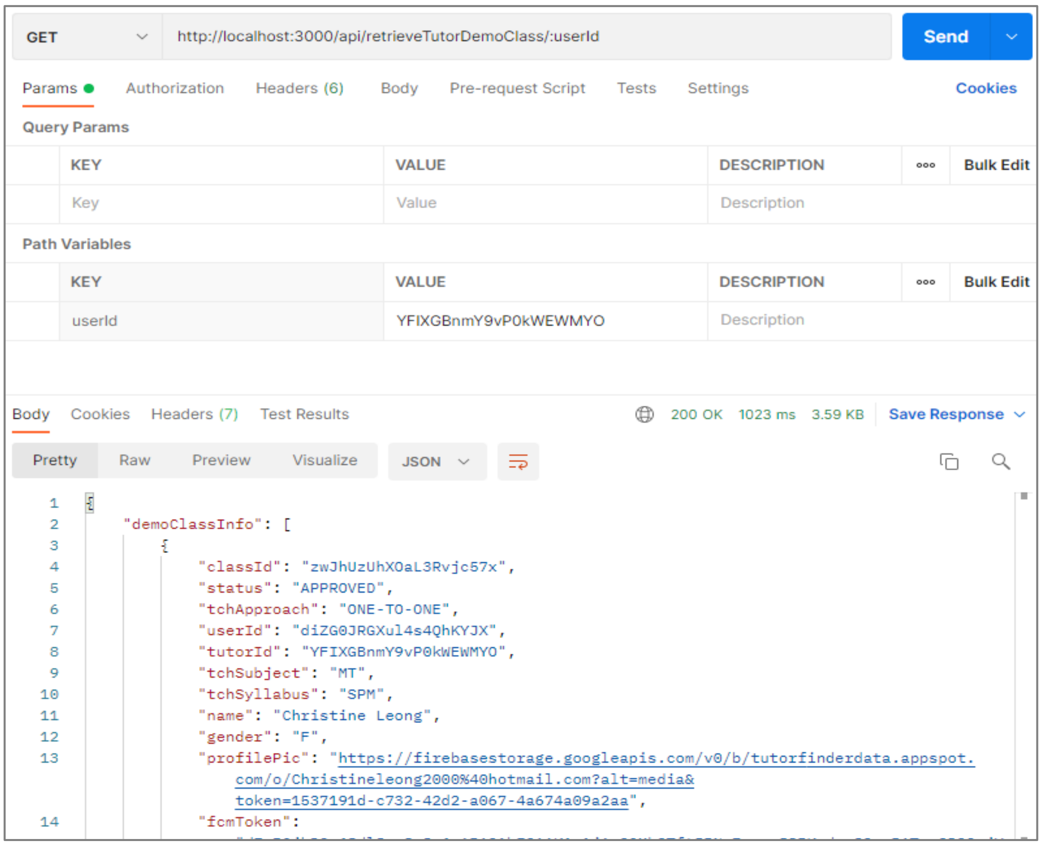
KEY	VALUE	DESCRIPTION
userId	YFIXGBnmY9vP0kWEWMYO	Description

Body | Cookies | Headers (7) | Test Results — 200 OK 1127 ms 2.83 KB — Save Response

Pretty | Raw | Preview | Visualize | JSON

```json
{
    "formalClassInfo": [
        {
            "tutorId": "YFIXGBnmY9vP0kWEWMYO",
            "tchApproach": "ONE-TO-ONE",
            "tchSubject": "MT",
            "idealFee": 50,
            "userId": "XkxeGBh0mz0tmLxlAAdk",
            "tchSyllabus": "SPM",
            "name": "Carmen Chin",
            "gender": "F",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/o/
                carmen123%40gmail.com?alt=media%26token=c035f650-8f71-42fb-9a79-cb651b52eee2&role=STUDENT",
            "fcmToken":
                "dFnPGjkDQ-6Jdl8qv8-8s6:APA91bE3AAK1y1jA_2GMkOTft5PNuFvruaJSRKxdawS2raBAT-_2D39ujVvnDenfw0ZQ
                ktLuUP4XILKVCyWLzTI1pAHlJdlkrHFc7z-K_V-52bgaURRuBxec3A1S3dtTcpdyZcAjdytf"
```

Figure 7.6: API Testing – Retrieve Tutor Formal Class



GET http://localhost:3000/api/retrieveTutorDemoClass/:userId

Params ● | Authorization | Headers (6) | Body | Pre-request Script | Tests | Settings | Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path Variables

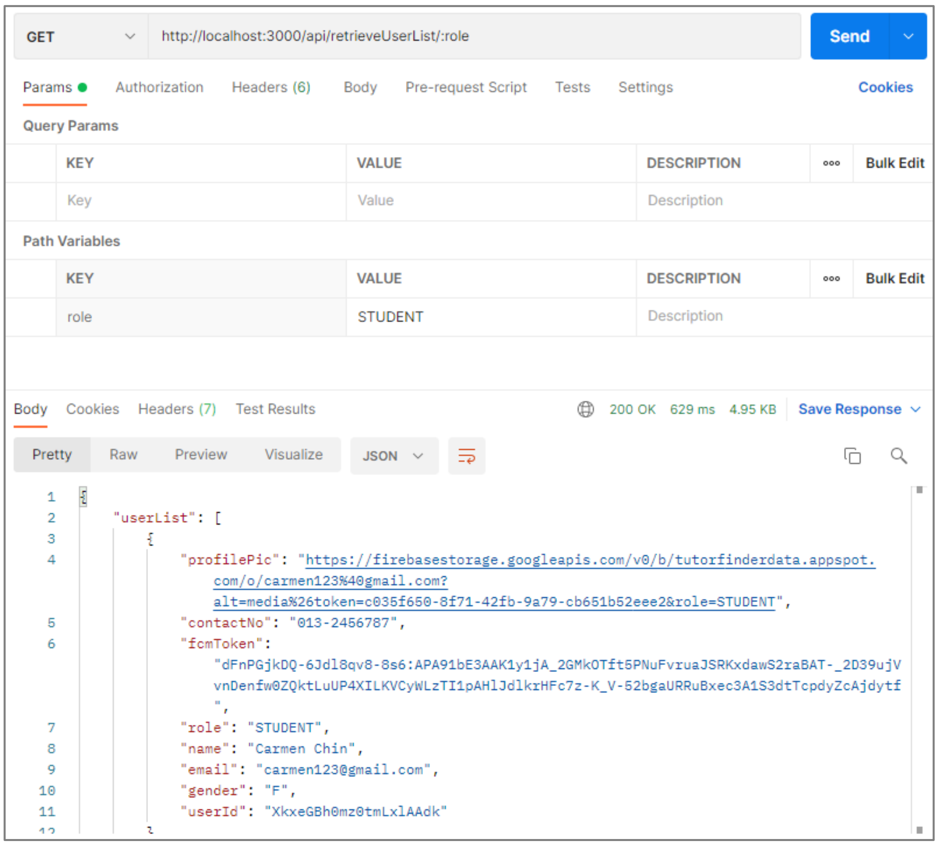
KEY	VALUE	DESCRIPTION
userId	YFIXGBnmY9vP0kWEWMYO	Description

Body | Cookies | Headers (7) | Test Results — 200 OK 1023 ms 3.59 KB — Save Response

Pretty | Raw | Preview | Visualize | JSON

```json
{
    "demoClassInfo": [
        {
            "classId": "zwJhUzUhXOaL3Rvjc57x",
            "status": "APPROVED",
            "tchApproach": "ONE-TO-ONE",
            "userId": "diZG0JRGXul4s4QhKYJX",
            "tutorId": "YFIXGBnmY9vP0kWEWMYO",
            "tchSubject": "MT",
            "tchSyllabus": "SPM",
            "name": "Christine Leong",
            "gender": "F",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.
                com/o/Christineleong2000%40hotmail.com?alt=media&
                token=1537191d-c732-42d2-a067-4a674a09a2aa",
            "fcmToken":
```

Figure 7.7: API Testing – Retrieve Tutor Demo Class


GET http://localhost:3000/api/retrieveUserList/:role

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

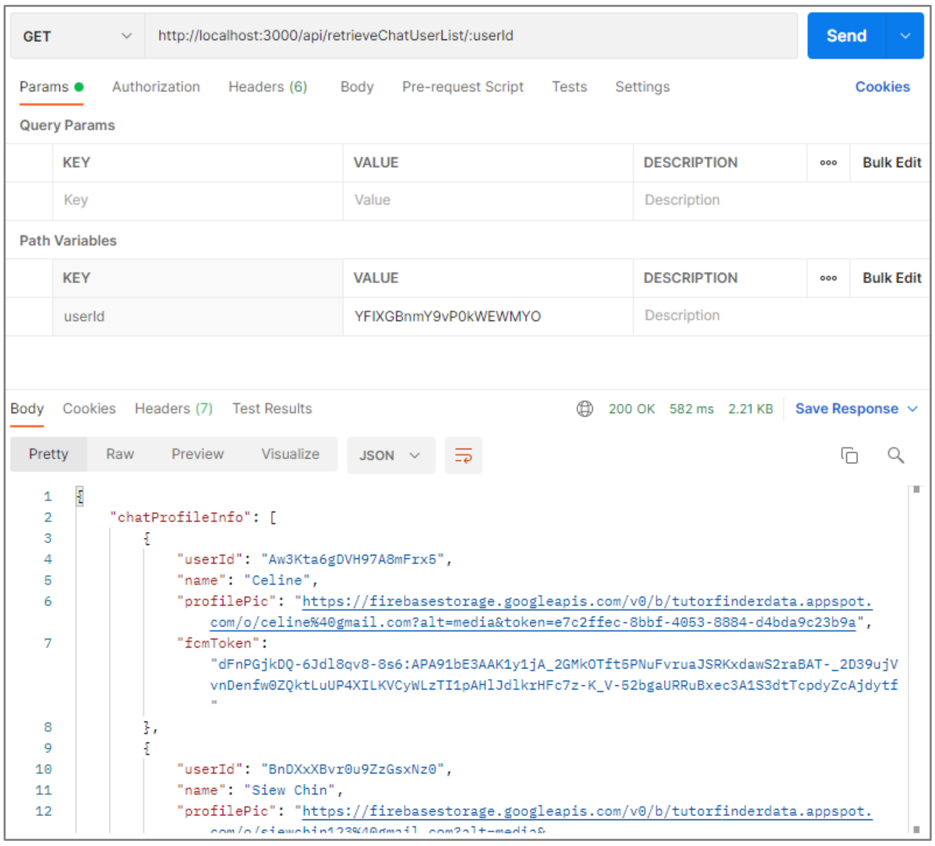
Path Variables

KEY	VALUE	DESCRIPTION
role	STUDENT	Description

200 OK 629 ms 4.95 KB

```
{
    "userList": [
        {
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.
                com/o/carmen123%40gmail.com?
                alt=media%26token=c035f650-8f71-42fb-9a79-cb651b52eee2&role=STUDENT",
            "contactNo": "013-2456787",
            "fcmToken":
                "dFnPGjkDQ-6Jdl8qv8-8s6:APA91bE3AAK1y1jA_2GMkOTft5PNuFvruaJSRKxdawS2raBAT-_2D39ujV
                vnDenfw0ZQktLuUP4XILKVCyWLzTI1pAHlJdlkrHFc7z-K_V-52bgaURRuBxec3A1S3dtTcpdyZcAjdytf
                ",
            "role": "STUDENT",
            "name": "Carmen Chin",
            "email": "carmen123@gmail.com",
            "gender": "F",
            "userId": "XkxeGBh0mz0tmLxlAAdk"
```

Figure 7.8: API Testing – Retrieve User List


GET http://localhost:3000/api/retrieveChatUserList/:userId

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

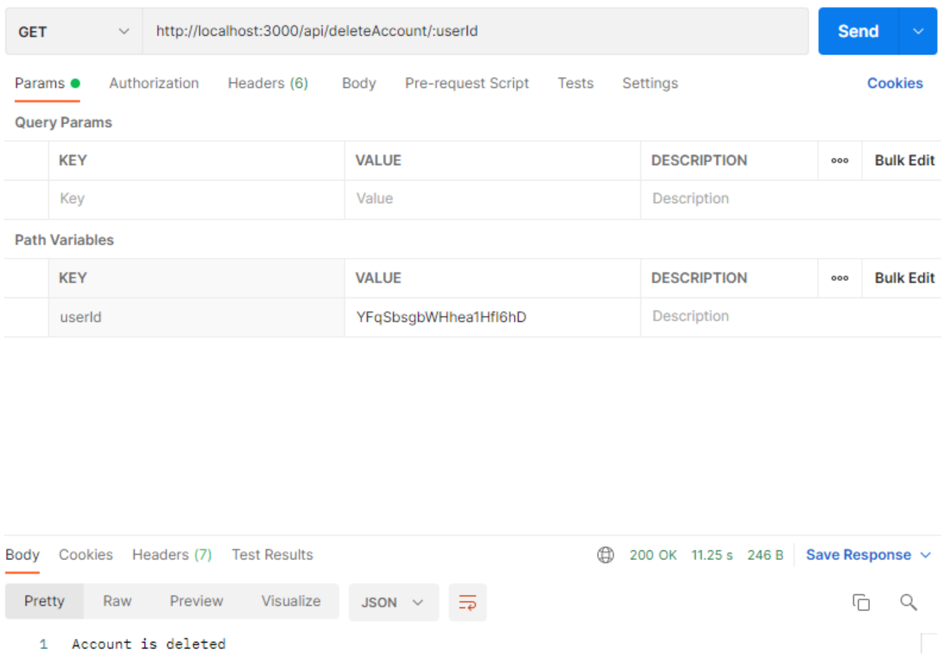
Path Variables

KEY	VALUE	DESCRIPTION
userId	YFIXGBnmY9vP0kWEWMYO	Description

200 OK 582 ms 2.21 KB

```
{
    "chatProfileInfo": [
        {
            "userId": "Aw3Kta6gDVH97A8mFrx5",
            "name": "Celine",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.
                com/o/celine%40gmail.com?alt=media&token=e7c2ffec-8bbf-4053-8884-d4bda9c23b9a",
            "fcmToken":
                "dFnPGjkDQ-6Jdl8qv8-8s6:APA91bE3AAK1y1jA_2GMkOTft5PNuFvruaJSRKxdawS2raBAT-_2D39ujV
                vnDenfw0ZQktLuUP4XILKVCyWLzTI1pAHlJdlkrHFc7z-K_V-52bgaURRuBxec3A1S3dtTcpdyZcAjdytf
                "
        },
        {
            "userId": "BnDXxXBvr0u9ZzGsxNz0",
            "name": "Siew Chin",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.
```

Figure 7.9: API Testing – Retrieve Chat User List



GET http://localhost:3000/api/deleteAccount/:userId

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

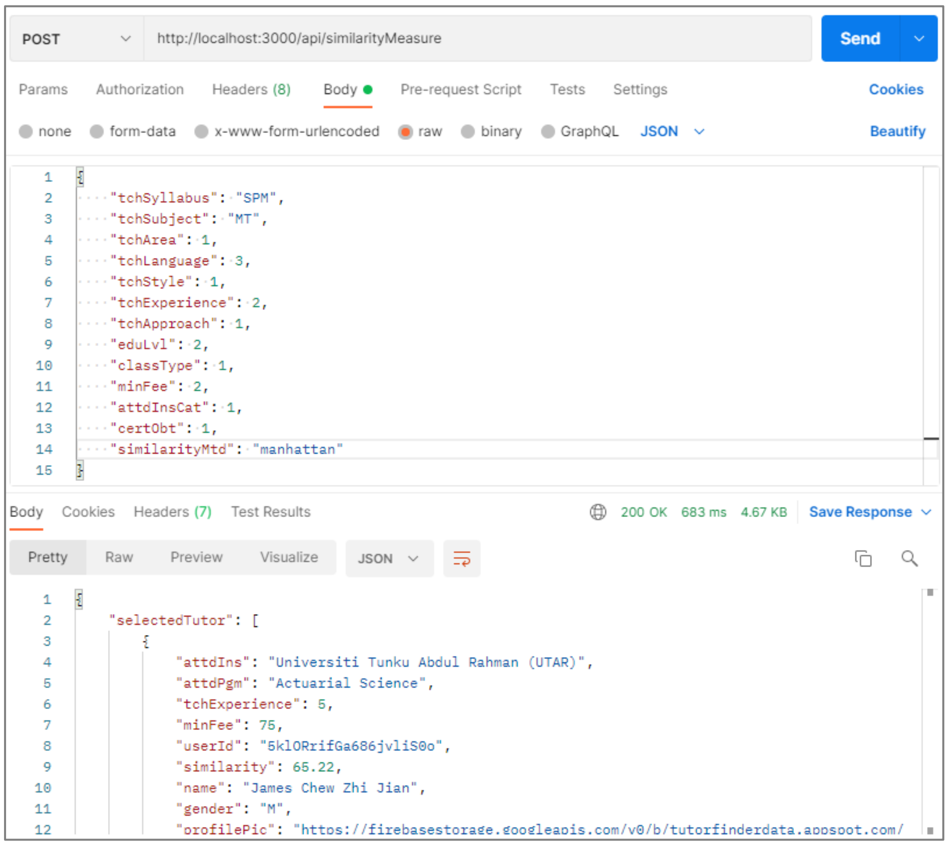
Path Variables

KEY	VALUE	DESCRIPTION
userId	YFqSbsgbWHhea1Hfl6hD	Description

200 OK 11.25 s 246 B

```
Account is deleted
```

Figure 7.10: API Testing – Delete Account



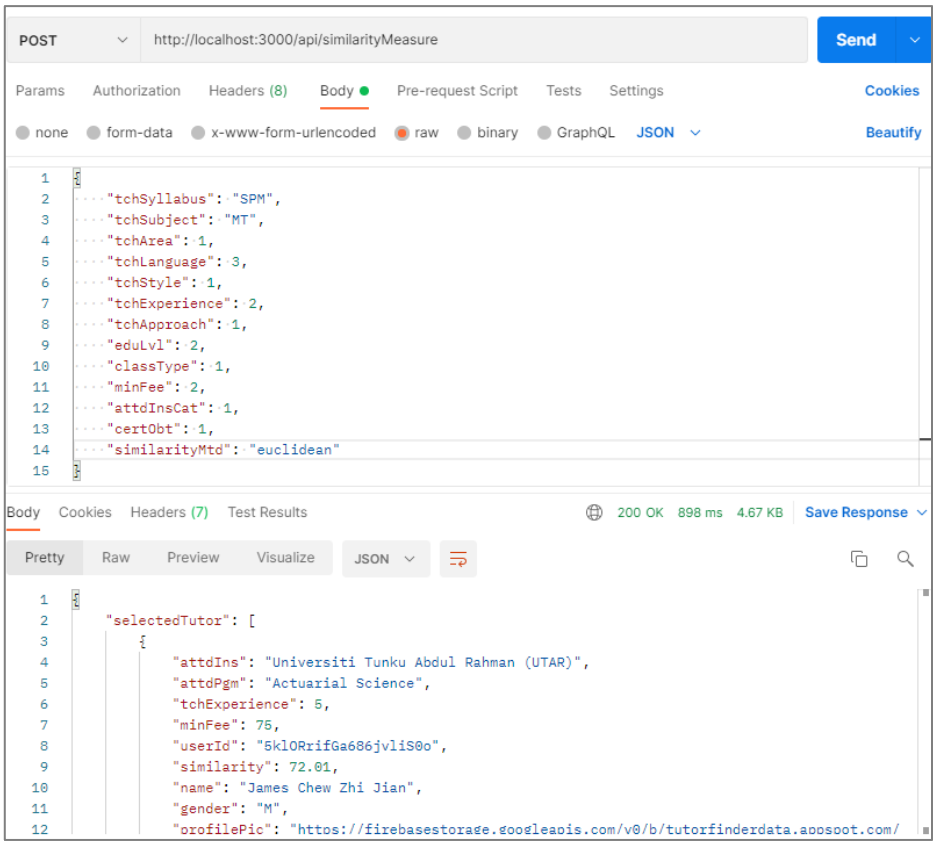
POST http://localhost:3000/api/similarityMeasure

```
{
    "tchSyllabus": "SPM",
    "tchSubject": "MT",
    "tchArea": 1,
    "tchLanguage": 3,
    "tchStyle": 1,
    "tchExperience": 2,
    "tchApproach": 1,
    "eduLvl": 2,
    "classType": 1,
    "minFee": 2,
    "attdInsCat": 1,
    "certObt": 1,
    "similarityMtd": "manhattan"
}
```

200 OK 683 ms 4.67 KB

```
{
    "selectedTutor": [
        {
            "attdIns": "Universiti Tunku Abdul Rahman (UTAR)",
            "attdPgm": "Actuarial Science",
            "tchExperience": 5,
            "minFee": 75,
            "userId": "5klORrifGa686jvliS0o",
            "similarity": 65.22,
            "name": "James Chew Zhi Jian",
            "gender": "M",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/
```

Figure 7.11: API Testing – Similarity Measure (Manhattan)



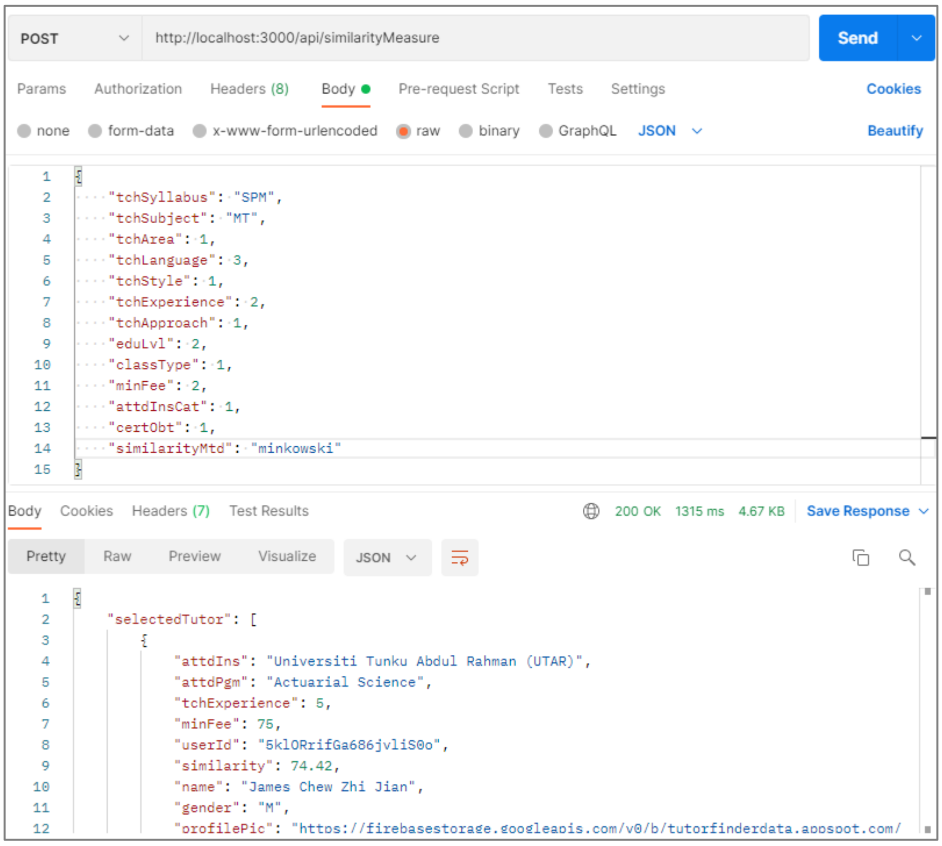
```
http://localhost:3000/api/similarityMeasure
```

```
{
    "tchSyllabus": "SPM",
    "tchSubject": "MT",
    "tchArea": 1,
    "tchLanguage": 3,
    "tchStyle": 1,
    "tchExperience": 2,
    "tchApproach": 1,
    "eduLvl": 2,
    "classType": 1,
    "minFee": 2,
    "attdInsCat": 1,
    "certObt": 1,
    "similarityMtd": "euclidean"
}
```

200 OK 898 ms 4.67 KB

```
{
    "selectedTutor": [
        {
            "attdIns": "Universiti Tunku Abdul Rahman (UTAR)",
            "attdPgm": "Actuarial Science",
            "tchExperience": 5,
            "minFee": 75,
            "userId": "5klORrifGa686jvliS0o",
            "similarity": 72.01,
            "name": "James Chew Zhi Jian",
            "gender": "M",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/
```

Figure 7.12: API Testing – Similarity Measure (Euclidean)



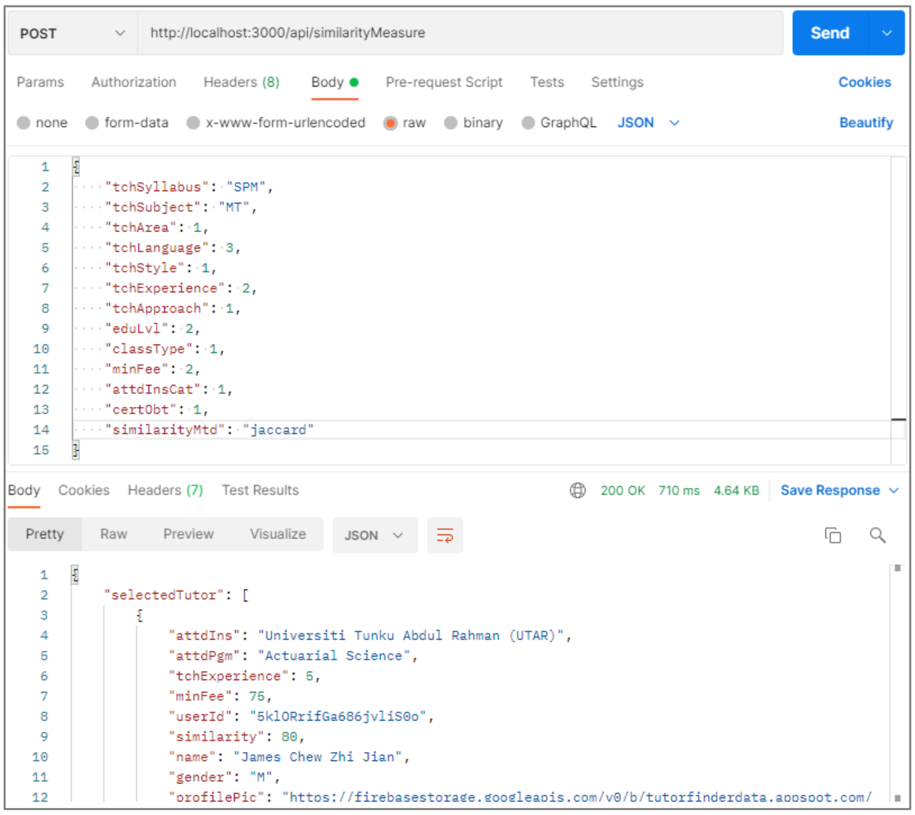
```
http://localhost:3000/api/similarityMeasure
```

```
{
    "tchSyllabus": "SPM",
    "tchSubject": "MT",
    "tchArea": 1,
    "tchLanguage": 3,
    "tchStyle": 1,
    "tchExperience": 2,
    "tchApproach": 1,
    "eduLvl": 2,
    "classType": 1,
    "minFee": 2,
    "attdInsCat": 1,
    "certObt": 1,
    "similarityMtd": "minkowski"
}
```

200 OK 1315 ms 4.67 KB

```
{
    "selectedTutor": [
        {
            "attdIns": "Universiti Tunku Abdul Rahman (UTAR)",
            "attdPgm": "Actuarial Science",
            "tchExperience": 5,
            "minFee": 75,
            "userId": "5klORrifGa686jvliS0o",
            "similarity": 74.42,
            "name": "James Chew Zhi Jian",
            "gender": "M",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/
```

Figure 7.13: API Testing – Similarity Measure (Minkowski)



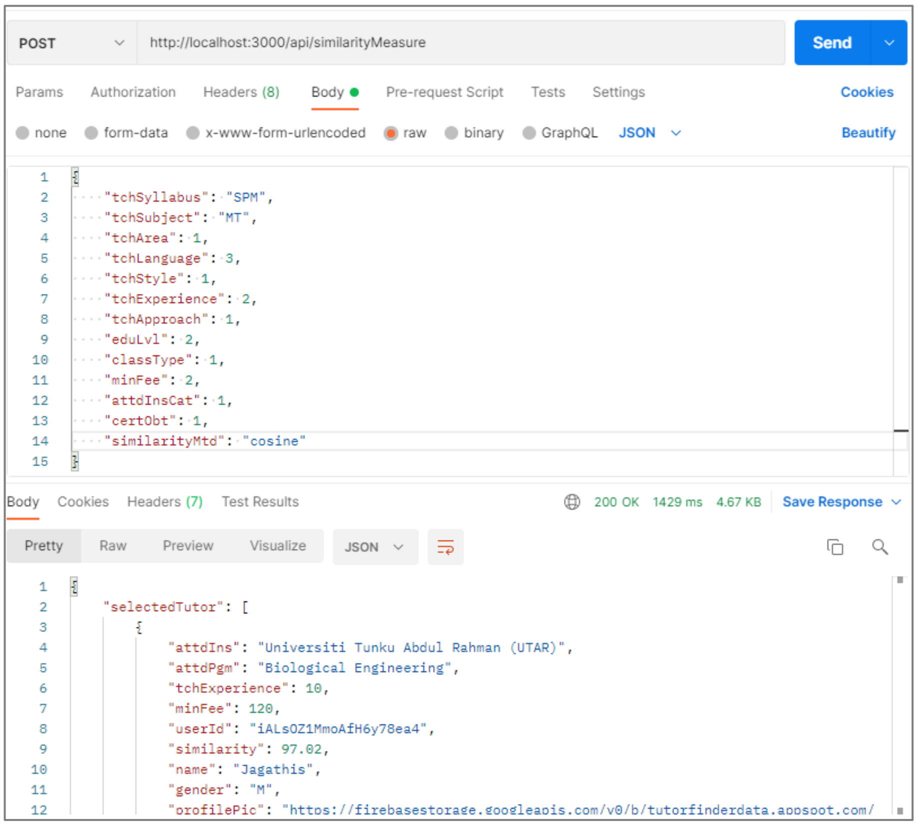
http://localhost:3000/api/similarityMeasure

```
{
    "tchSyllabus": "SPM",
    "tchSubject": "MT",
    "tchArea": 1,
    "tchLanguage": 3,
    "tchStyle": 1,
    "tchExperience": 2,
    "tchApproach": 1,
    "eduLvl": 2,
    "classType": 1,
    "minFee": 2,
    "attdInsCat": 1,
    "certObt": 1,
    "similarityMtd": "jaccard"
}
```

200 OK 710 ms 4.64 KB

```
{
    "selectedTutor": [
        {
            "attdIns": "Universiti Tunku Abdul Rahman (UTAR)",
            "attdPgm": "Actuarial Science",
            "tchExperience": 5,
            "minFee": 75,
            "userId": "5klORrifGa686jvliS0o",
            "similarity": 80,
            "name": "James Chew Zhi Jian",
            "gender": "M",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/
```

Figure 7.14: API Testing – Similarity Measure (Jaccard)



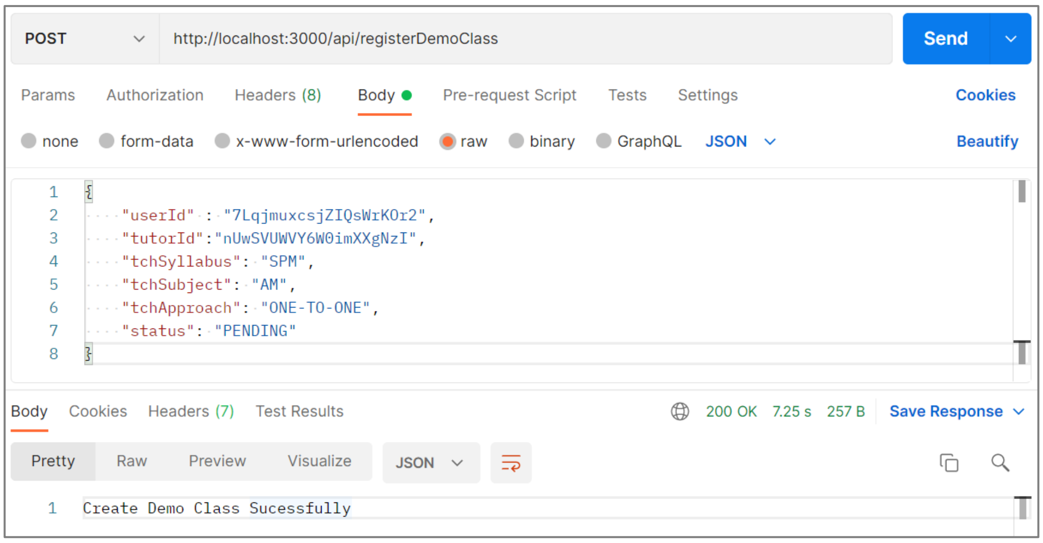
http://localhost:3000/api/similarityMeasure

```
{
    "tchSyllabus": "SPM",
    "tchSubject": "MT",
    "tchArea": 1,
    "tchLanguage": 3,
    "tchStyle": 1,
    "tchExperience": 2,
    "tchApproach": 1,
    "eduLvl": 2,
    "classType": 1,
    "minFee": 2,
    "attdInsCat": 1,
    "certObt": 1,
    "similarityMtd": "cosine"
}
```

200 OK 1429 ms 4.67 KB

```
{
    "selectedTutor": [
        {
            "attdIns": "Universiti Tunku Abdul Rahman (UTAR)",
            "attdPgm": "Biological Engineering",
            "tchExperience": 10,
            "minFee": 120,
            "userId": "iALsOZ1MmoAfH6y78ea4",
            "similarity": 97.02,
            "name": "Jagathis",
            "gender": "M",
            "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/
```

Figure 7.15: API Testing – Similarity Measure (Cosine)

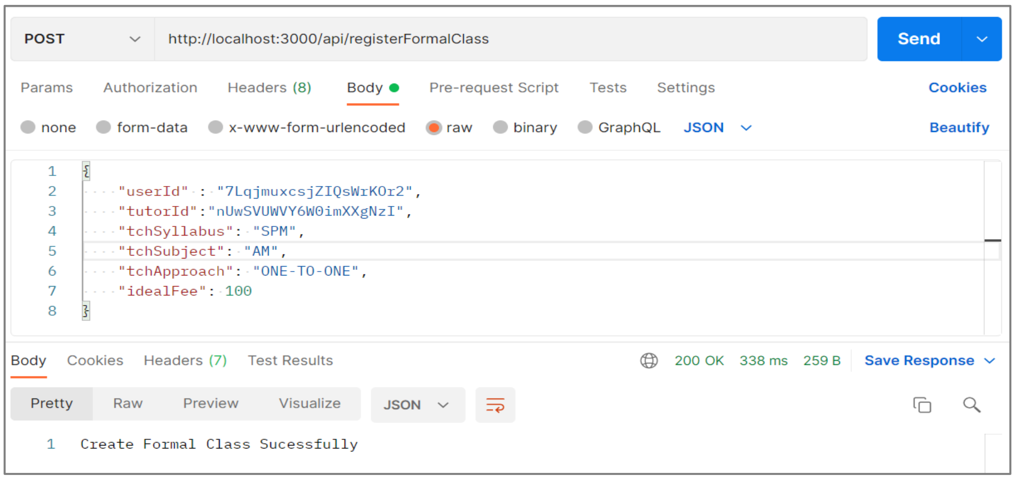


```
http://localhost:3000/api/registerDemoClass
```

```
{
    "userId" : "7LqjmuxcsjZIQsWrKOr2",
    "tutorId":"nUwSVUWVY6W0imXXgNzI",
    "tchSyllabus": "SPM",
    "tchSubject": "AM",
    "tchApproach": "ONE-TO-ONE",
    "status": "PENDING"
}
```

```
Create Demo Class Sucessfully
```

Figure 7.16: API Testing – Register Demo Class

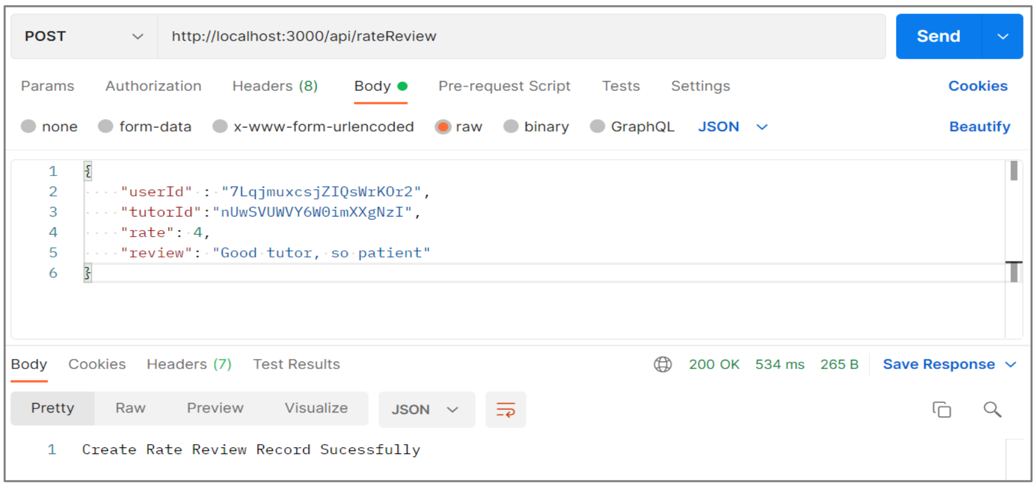


```
http://localhost:3000/api/registerFormalClass
```

```
{
    "userId" : "7LqjmuxcsjZIQsWrKOr2",
    "tutorId":"nUwSVUWVY6W0imXXgNzI",
    "tchSyllabus": "SPM",
    "tchSubject": "AM",
    "tchApproach": "ONE-TO-ONE",
    "idealFee": 100
}
```

```
Create Formal Class Sucessfully
```

Figure 7.17: API Testing – Register Formal Class

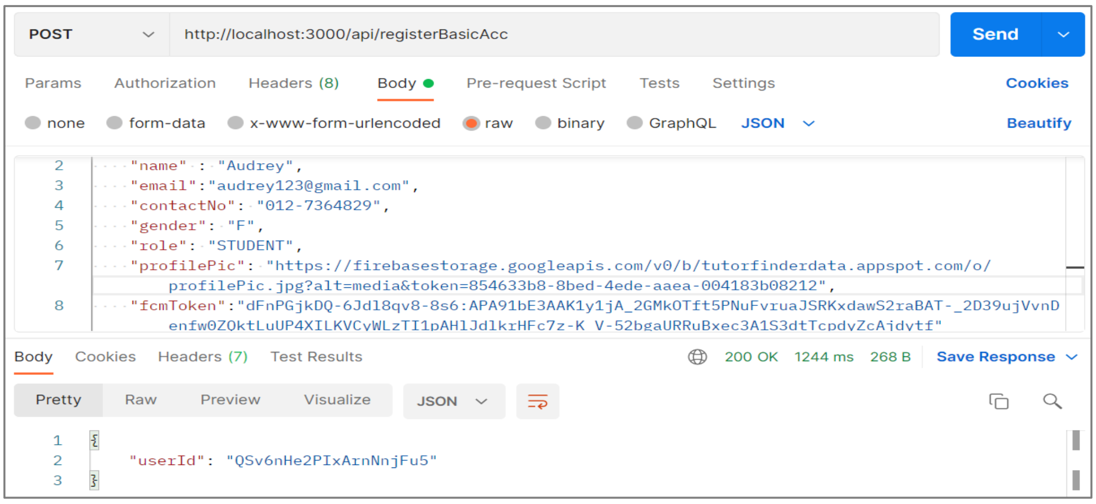


```
http://localhost:3000/api/rateReview
```

```
{
    "userId" : "7LqjmuxcsjZIQsWrKOr2",
    "tutorId":"nUwSVUWVY6W0imXXgNzI",
    "rate": 4,
    "review": "Good tutor, so patient"
}
```

```
Create Rate Review Record Sucessfully
```

Figure 7.18: API Testing – Rate Review



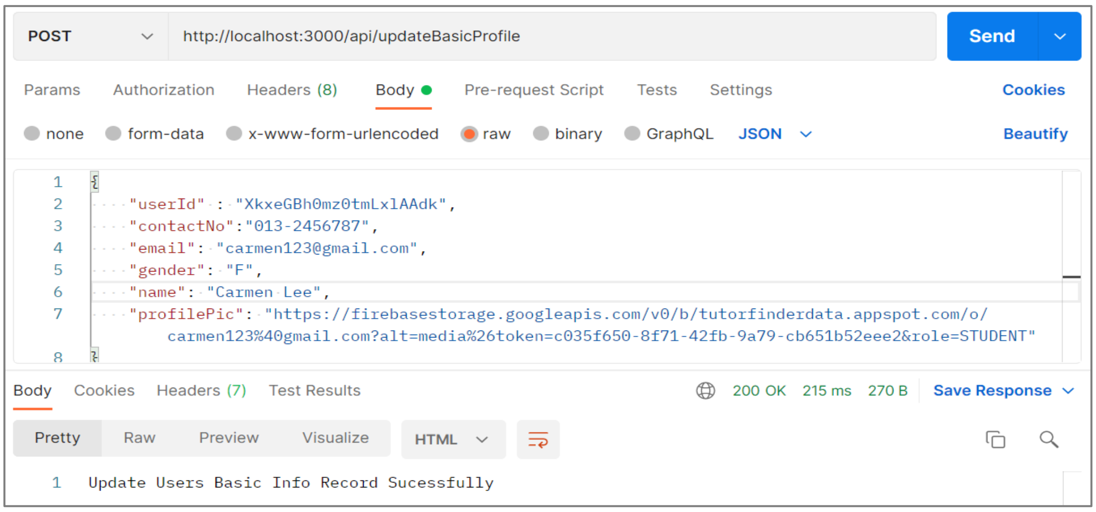
http://localhost:3000/api/registerBasicAcc

```
    "name" : "Audrey",
    "email":"audrey123@gmail.com",
    "contactNo": "012-7364829",
    "gender": "F",
    "role": "STUDENT",
    "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/o/
        profilePic.jpg?alt=media&token=854633b8-8bed-4ede-aaea-004183b08212",
    "fcmToken":"dFnPGjkDQ-6Jdl8qv8-8s6:APA91bE3AAK1y1jA_2GMkOTft5PNuFvruaJSRKxdawS2raBAT-_2D39ujVvnD
        enfw0ZQktLuUP4XILKVCyWLzTI1pAHlJdlkrHFc7z-K_V-52bgaURRuBxec3A1S3dtTcpdyZcAjdytf"
```

200 OK 1244 ms 268 B

```
{
    "userId": "QSv6nHe2PIxArnNnjFu5"
}
```

Figure 7.19: API Testing – Register Basic Account



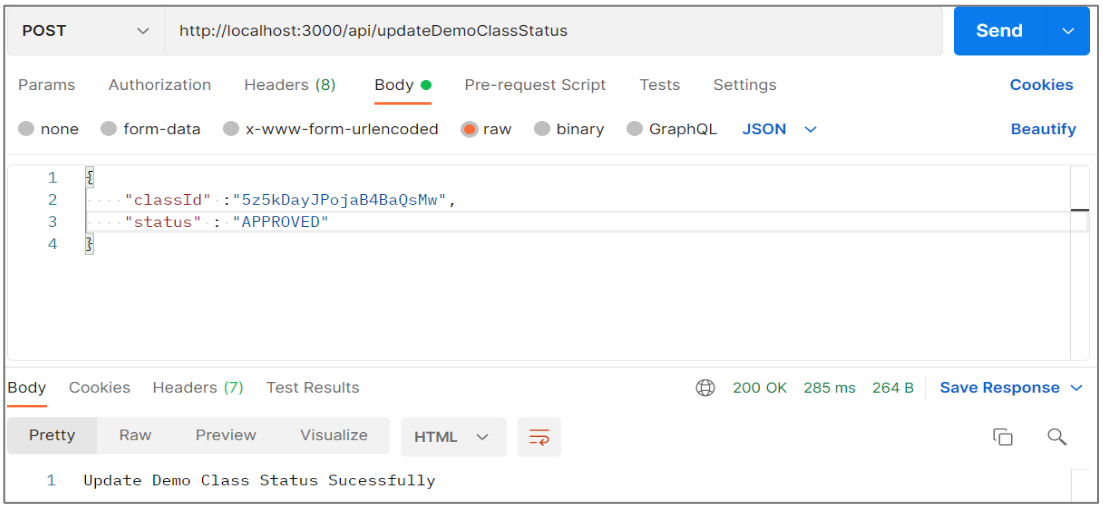
http://localhost:3000/api/updateBasicProfile

```
{
    "userId" : "XkxeGBh0mz0tmLxlAAdk",
    "contactNo":"013-2456787",
    "email": "carmen123@gmail.com",
    "gender": "F",
    "name": "Carmen Lee",
    "profilePic": "https://firebasestorage.googleapis.com/v0/b/tutorfinderdata.appspot.com/o/
        carmen123%40gmail.com?alt=media%26token=c035f650-8f71-42fb-9a79-cb651b52eee2&role=STUDENT"
}
```

200 OK 215 ms 270 B

```
Update Users Basic Info Record Sucessfully
```

Figure 7.20: API Testing – Update Basic Profile



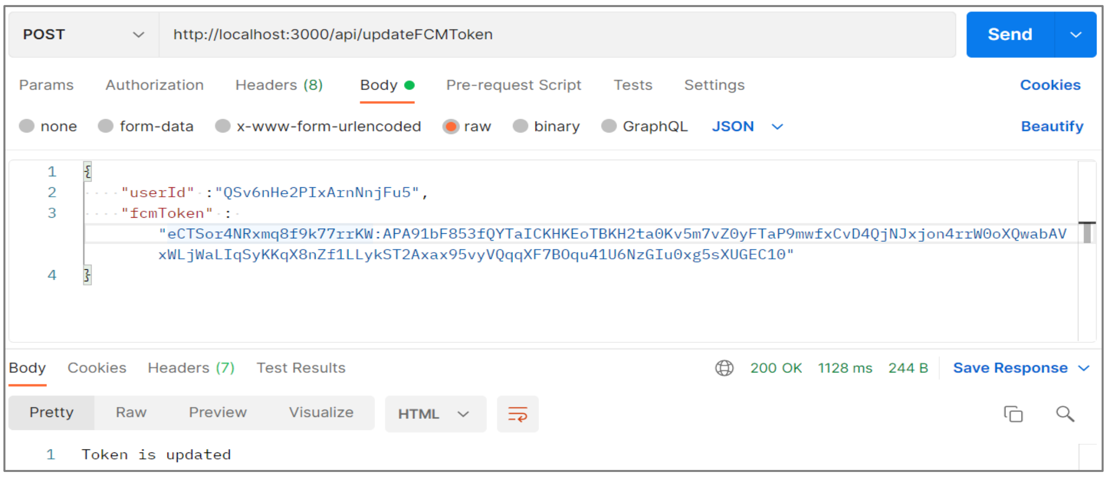
http://localhost:3000/api/updateDemoClassStatus

```
{
    "classId" :"5z5kDayJPojaB4BaQsMw",
    "status" : "APPROVED"
}
```

200 OK 285 ms 264 B

```
Update Demo Class Status Sucessfully
```

Figure 7.21: API Testing – Update Demo Class Status



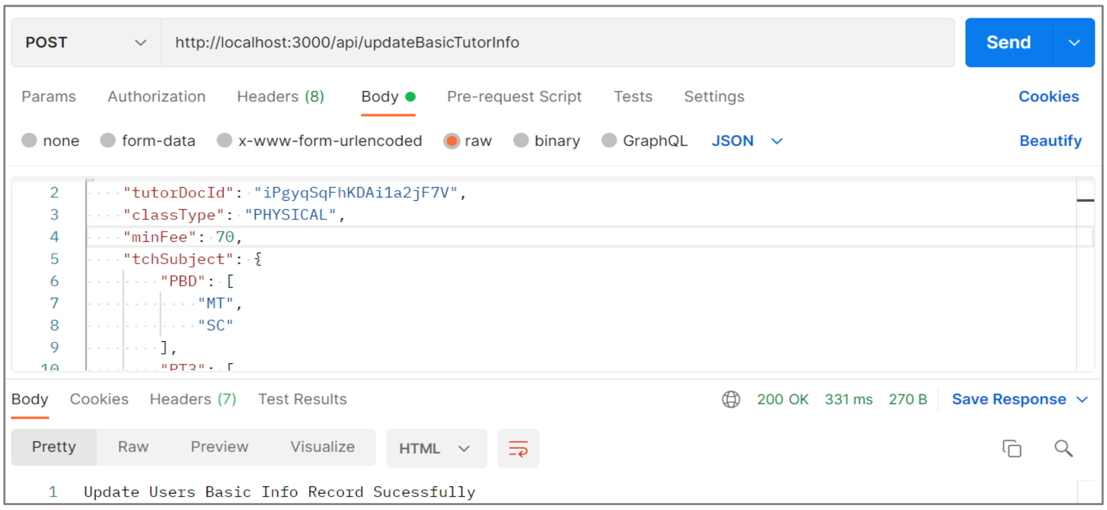
http://localhost:3000/api/updateFCMToken

```
{
    "userId" :"QSv6nHe2PIxArnNnjFu5",
    "fcmToken" : 
        "eCTSor4NRxmq8f9k77rrKW:APA91bF853fQYTaICKHKEoTBKH2ta0Kv5m7vZ0yFTaP9mwfxCvD4QjNJxjon4rrW0oXQwabAV
        xWLjWaLIqSyKKqX8nZf1LLykST2Axax95vyVQqqXF7BOqu41U6NzGIu0xg5sXUGEC10"
}
```

200 OK 1128 ms 244 B

```
Token is updated
```

Figure 7.22: API Testing – Update FCM Token



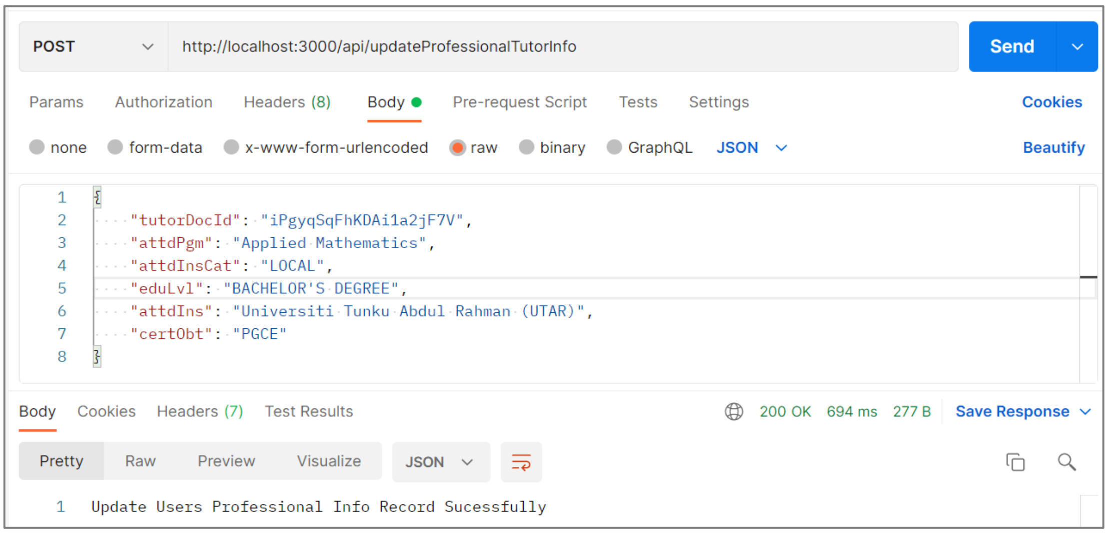
http://localhost:3000/api/updateBasicTutorInfo

```
    "tutorDocId": "iPgyqSqFhKDAi1a2jF7V",
    "classType": "PHYSICAL",
    "minFee": 70,
    "tchSubject": {
        "PBD": [
            "MT",
            "SC"
        ],
        "PT3": [
```

200 OK 331 ms 270 B

```
Update Users Basic Info Record Sucessfully
```

Figure 7.23: API Testing – Update Basic Tutor Information



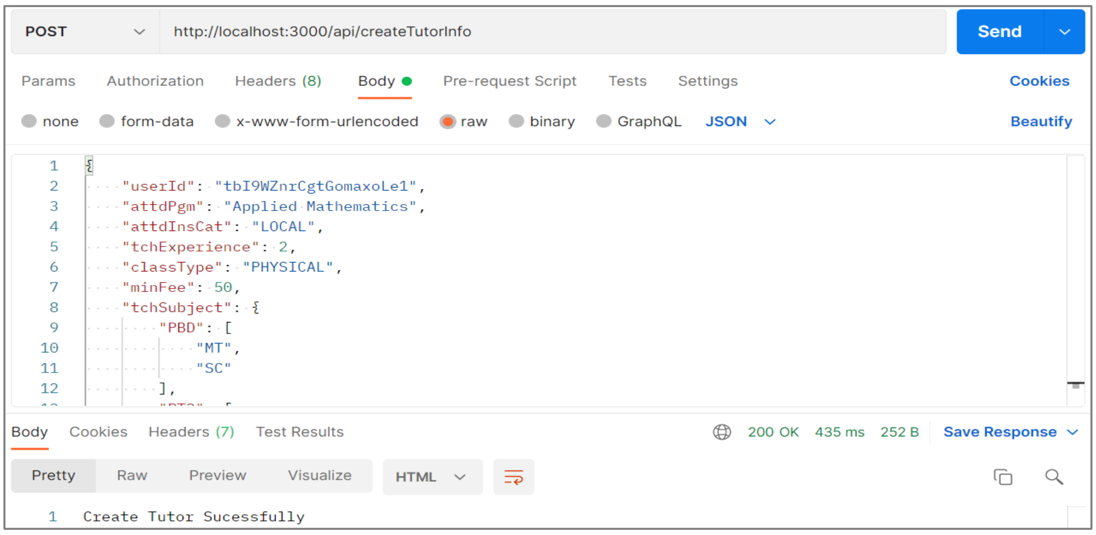
http://localhost:3000/api/updateProfessionalTutorInfo

```
{
    "tutorDocId": "iPgyqSqFhKDAi1a2jF7V",
    "attdPgm": "Applied Mathematics",
    "attdInsCat": "LOCAL",
    "eduLvl": "BACHELOR'S DEGREE",
    "attdIns": "Universiti Tunku Abdul Rahman (UTAR)",
    "certObt": "PGCE"
}
```

200 OK 694 ms 277 B

```
Update Users Professional Info Record Sucessfully
```

Figure 7.24: API Testing – Update Professional Tutor Information



```
{
    "userId": "tbI9WZnrCgtGomaxoLe1",
    "attdPgm": "Applied Mathematics",
    "attdInsCat": "LOCAL",
    "tchExperience": 2,
    "classType": "PHYSICAL",
    "minFee": 50,
    "tchSubject": {
        "PBD": [
            "MT",
            "SC"
        ],
```

```
Create Tutor Sucessfully
```

Figure 7.25: API Testing – Create Tutor Information

7.4 User Acceptance Testing (UAT)

There are 9 users who are invited to conduct UAT while there are 3 users for each role activity testing which are tutor seeker, registered tutor, and admin. UAT test cases template is shown below while the results are displayed in the appendix section. The status of each action could be “pass”, “pass with help” or “fail”, pass with help indicate that the user completes the task with the hints given by the test moderator. The UAT is conducted virtually by using AnyDesk.

7.4.1 Tutor Seeker Side

Table 7.16: UAT – Create and Login Account

Test Case ID	UAT-TS-001	
Test Module	Create and Login Account	
Test Description	**Status**	**Feedbacks**
Able to upload a profile picture		
Able to insert personal information		
Able to submit the details to register an account		
Able to log in to the registered account		

Table 7.17: UAT – Search and View Ideal Tutor

Test Case ID	UAT-TS-002	
Test Module	Search and View Ideal Tutors	
Test Description	**Status**	**Feedbacks**
Able to set learning preference		
Able to search for a tutor		
Able to select different search results (result 1, result 2, result 3, result 4, result 5) which applied different similarity measure methods		
Able to view different search results		
Able to view tutor's teaching and basic profile details		
Able to read tutor's rates and reviews		

Table 7.18: UAT – Chat With Tutor

Test Case ID	UAT-TS-003	
Test Module	Chat With Tutor	
Test Description	**Status**	**Feedbacks**
Able to initiate a chat with a tutor		
Able to view chat list		
Able to view chat history with tutors		
Able to view push notifications from tutor's chat message		
Able to reply to chat messages from tutor		

Table 7.19: UAT – Register and View Classes

Test Case ID	UAT-TS-004	
Test Module	Register and View Classes	
Test Description	**Status**	**Feedbacks**
Able to raise demo class		
Able to view the status of demo class request		
Able to book a formal class		
Able to view the class details of formal class		

Table 7.20: UAT – View and Edit Profile (Tutor Seeker)

Test Case ID	UAT-TS-005	
Test Module	View and Edit Profile	
Test Description	**Status**	**Feedbacks**
Able to view personal profile details		
Able to upload a new profile picture		
Able to modify personal details		
Able to save a new profile		

7.4.2 Registered Tutor Side

Table 7.21: UAT – Log In Account (Registered Tutor)

Test Case ID	UAT-T-001	
Test Module	Log In Account	
Test Description	**Status**	**Feedbacks**
Able to log in to the existing account from the admin		

Table 7.22: UAT – View and Edit Profile (Registered Tutor)

Test Case ID	UAT-T-002	
Test Module	View and Edit Profile	
Test Description	**Status**	**Feedbacks**
Able to view personal profile details		
Able to view rates and reviews from tutor seeker		
Able to upload a new profile picture		
Able to modify personal information		
Able to modify teaching information		
Able to edit profile successfully		

Table 7.23: UAT – Manage Class

Test Case ID	UAT-T-003	
Test Module	Manage Class	
Test Description	**Status**	**Feedbacks**
Able to view demo class requests list		

Able to view formal class bookings list		
Able to accept or reject demo class request		

Table 7.24: UAT – Chat With Tutor Seeker

Test Case ID	UAT-T-004	
Test Module	Chat With Tutor Seeker	
Test Description	**Status**	**Feedbacks**
Able to initiate a chat with tutor seeker		
Able to view chat list		
Able to view chat history with tutor seeker		
Able to view push notifications from tutor seeker's chat message		
Able to reply chat messages from tutor seeker		

7.4.3 Admin Side

Table 7.25: UAT – Log In Account (Admin)

Test Case ID	UAT-A-001	
Test Module	Log In Account	
Test Description	**Status**	**Feedbacks**
Able to log in to an existing account		

Table 7.26: UAT – Manager Tutor

Test Case ID	UAT-A-002	
Test Module	Manage Tutor	
Test Description	**Status**	**Feedbacks**
Able to upload a profile picture of tutor		
Able to insert personal and teaching details of the tutor		
Able to create an account for a tutor by submitting the details		
Able to search for a tutor		
Able to view the tutor list and the tutor profile		
Able to modify tutor's professional details		
Able to edit tutor profile by submitting the new professional details		

Table 7.27: UAT – View Tutor Seeker

Test Case ID	UAT-A-003	
Test Module	View Tutor Seeker	
Test Description	**Status**	**Feedbacks**
Able to view tutor seeker list		
Able to search tutor seekers by name		
Able to view tutor's seeker details		

7.5 Usability Testing

Usability testing is conducted with the same end-user in user acceptance testing which are 3 tutor seekers, 3 registered tutors, and 3 admin. Besides testing the usability of the function in this application usability testing played the main role in identifying the best similarity measure method which could meet the expectation of the tutor seeker. The usability testing is conducted virtually by using AnyDesk.

7.5.1 Usability Testing Scenario

7.5.1.1 Tutor Seeker Side

Table 7.28: Usability Testing Scenario – Tutor Seeker

ID	Test Scenario	Scenario Description
1	Register account	- Scenario: - You are a tutor seeker who wishes to find your ideal tutor by using this application. You want to register an account for this application when first entering this application. Task: - Sign up for an account
2	Login account	Scenario: - You wish to log in to the account that you have registered in scenario 1. Task: - Log in to your registered account.
3	Search ideal tutor and book a formal class	Scenario: - You wish to search for your ideal tutor based on your learning preferences. Once you found the tutor matched your ideal tutor criteria you book a formal class with the tutor. Task: - Search for the ideal tutor according to your

		preference. - Browse the 5 results generated from the application. - View the profile of the tutor that you are interested in after getting the search result. - Register a formal class with the tutor that matched your ideal tutor criteria.
4	Send message	Scenario: - You wish to ask about the available teaching time of the tutor that you have booked a formal class with him/her. Task: - Send a message to the tutor that you have booked a formal class with him. Chat Message: Hi, I have booked a formal class with you, may I know your teaching time?
5	Raise demo class requests and check request history.	Scenario: - You browse the popular tutor on the home screen and wish to try the demo class from one of the tutors on the list that you are interested in. Task: - Browse popular tutors. - Raise a demo class on one of the tutors that you are interested in on the popular list. - View the status of the demo class request and formal class details that you have raised just now.
6	Reply message	Scenario: - You received a push notification that stated a message is replied from the tutor that you raised a formal class in scenario 3. Task: - View the chat list and find the message box. - Reply to a message from the tutor. Chat Message: I see, I will let you know ASAP.
7	Edit profile	Scenario: - You decide to modify your personal information and let the tutor get your latest profile. Task: - Edit your profile information.

7.5.1.2 Registered Tutor Side

Table 7.29: Usability Testing Scenario – Registered Tutor

ID	Test Scenario	Scenario Description
1	Login account	Scenario: - You are a registered tutor of the application who wants to log in to your registered account to perform activities. Task: - Log in to your registered account. Email : clement123@gmail.com Password: Clement123#
2	Mange demo class	Scenario: - You have received some demo class requests from registered tutor seekers and wish to approve or reject the requests after viewing the request list. Task: - Approve or reject the pending request from "Carmen".
3	Arrange formal class	Scenario: - You have received some formal class bookings from registered tutor seekers and wish to discuss with them the class details such as time and price via chat after viewing the booking list. Task: - View formal class booking list. - Discuss class details with "Carmen" Chat Message: Hi you have registered my class, my teaching time will be every weekday from 8am to 5pm. Kindly let me know your preferred time.
4	Reply message	Scenario: - You received a push notification that stated you have a message from "Christine Leong", and you wish to reply to her message. Task: - Reply message from "Christine Leong". Chat Message: For your case, it should be RM70 per hour.

5	Edit profile	Scenario: - You have read the rate and review from the student who attended your class and feels that you are not suitable to teach BM. You decide to remove BM from your teaching subject and modify your personal information as well. Task: - View the rate and review from the student. - Edit your profile by removing BM from the teaching syllabus and any one of your personal information.

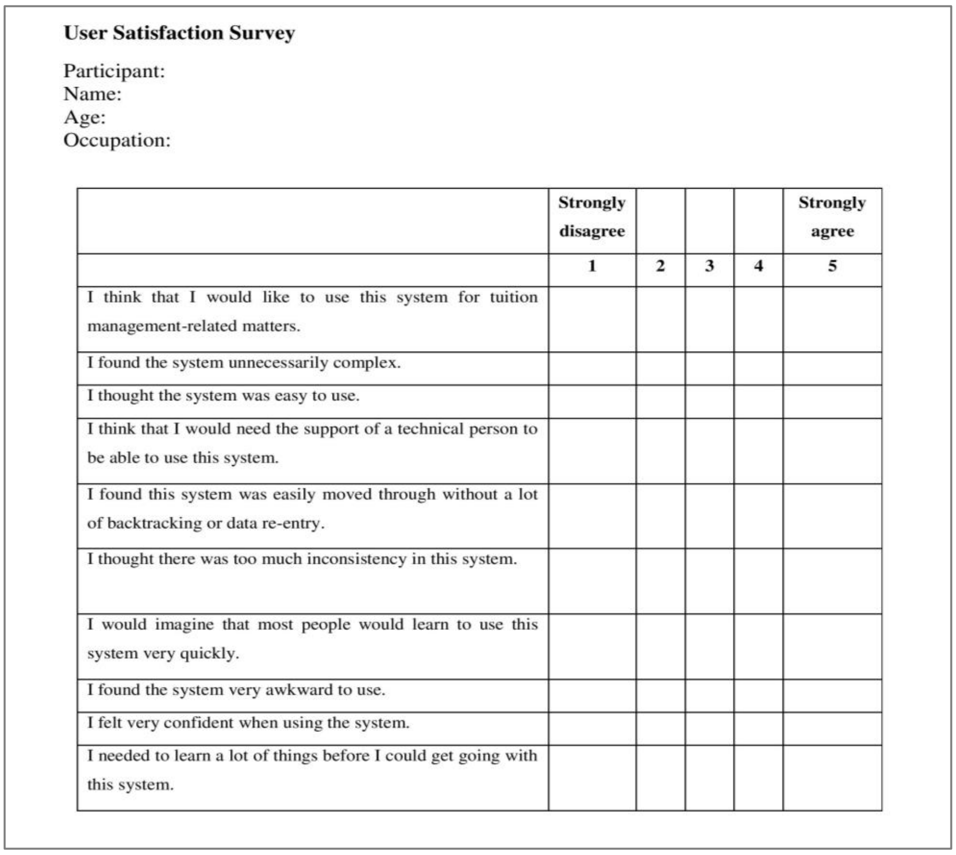
7.5.1.3 Admin Side

Table 7.30: Usability Testing Scenario – Admin

ID	Test Scenario	Scenario Description
1	Login account	Scenario: - You are an admin of the application who wants to log in to your registered account to perform activities. Task: - Log in to your registered account. Email : chongperngsia@hotmail.com Password: Chongperng123#
2	Add tutor	Scenario: - A new tutor who joined this platform, and you are responsible to register an account for him/her. Task: - Register a new tutor with verified information. Attended Institution : Universiti Malaya (UM) Attended Instituition Category : Local Highest Education Level : Bachelor’s Degree
3	Update tutor	Scenario: - The tutor that you registered in scenario 2 has submitted his/her updated professional information. You have verified the information and proceed to update his/her professional information in your profile. Task: - Update the listed information to the profile that

		you just created in scenario 2. Attended Institution : Stanford University Attended Instituition Category : Oversea Highest Education Level : Master's Degree
4	View registered tutor seeker	Scenario: - You have received a complaint from "Carmen" in reporting a registered tutor violating the platform's rules by charging a non-reasonable tuition fee. You have to write a report to management with the complainer's profile information of her such as name, contact number, and user id. Task: - View the profile information of "Carmen".

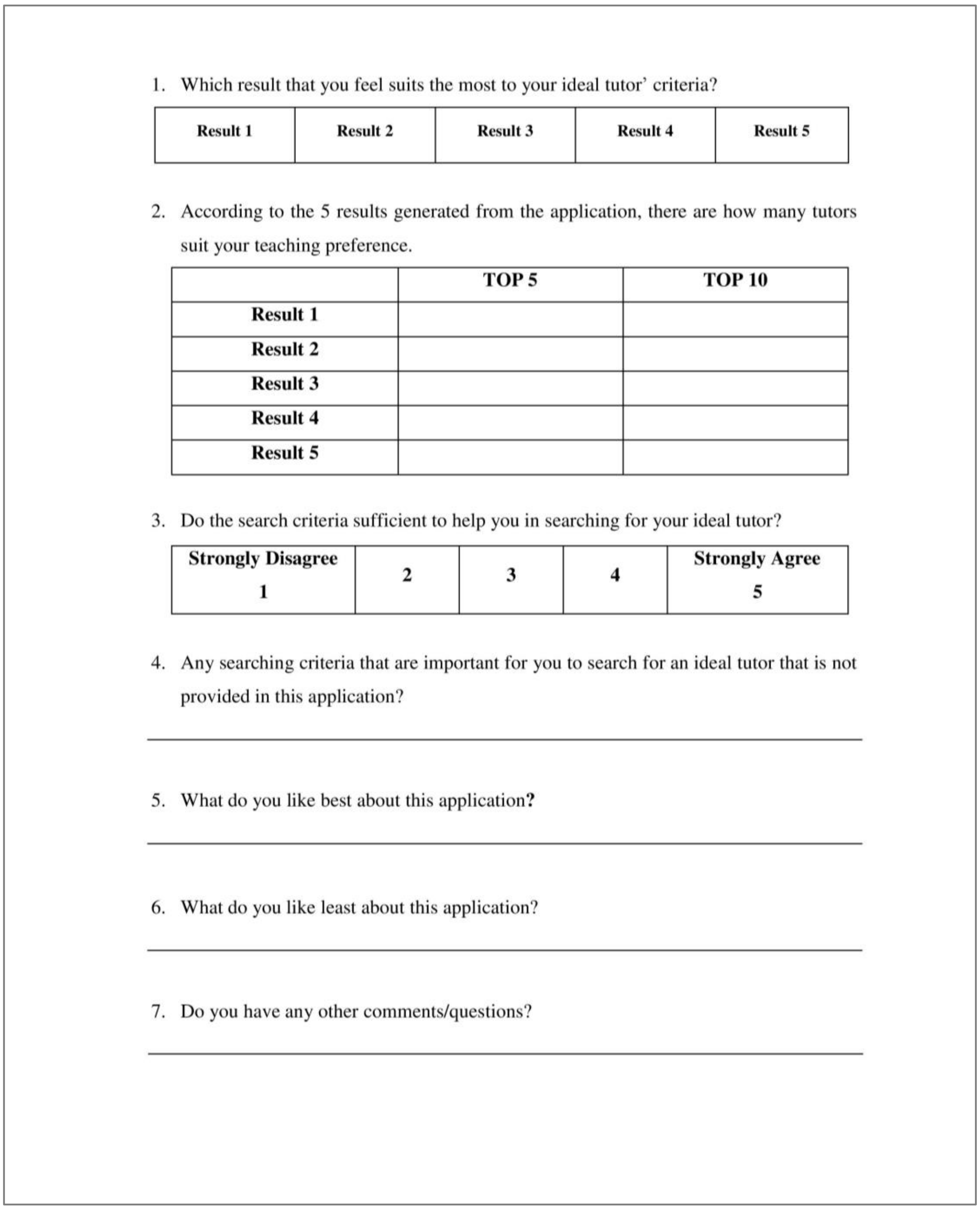
7.5.2 User Satisfaction Survey Form


User Satisfaction Survey

Participant:
Name:
Age:
Occupation:

	Strongly disagree				Strongly agree
	1	2	3	4	5
I think that I would like to use this system for tuition management-related matters.					
I found the system unnecessarily complex.					
I thought the system was easy to use.					
I think that I would need the support of a technical person to be able to use this system.					
I found this system was easily moved through without a lot of backtracking or data re-entry.					
I thought there was too much inconsistency in this system.					
I would imagine that most people would learn to use this system very quickly.					
I found the system very awkward to use.					
I felt very confident when using the system.					
I needed to learn a lot of things before I could get going with this system.					

Figure 7.26: User Satisfaction Survey Form – Tutor Seeker (Part 1)



1. Which result that you feel suits the most to your ideal tutor' criteria?

Result 1	**Result 2**	**Result 3**	**Result 4**	**Result 5**

2. According to the 5 results generated from the application, there are how many tutors suit your teaching preference.

	TOP 5	**TOP 10**
Result 1		
Result 2		
Result 3		
Result 4		
Result 5		

3. Do the search criteria sufficient to help you in searching for your ideal tutor?

Strongly Disagree 1	**2**	**3**	**4**	**Strongly Agree 5**

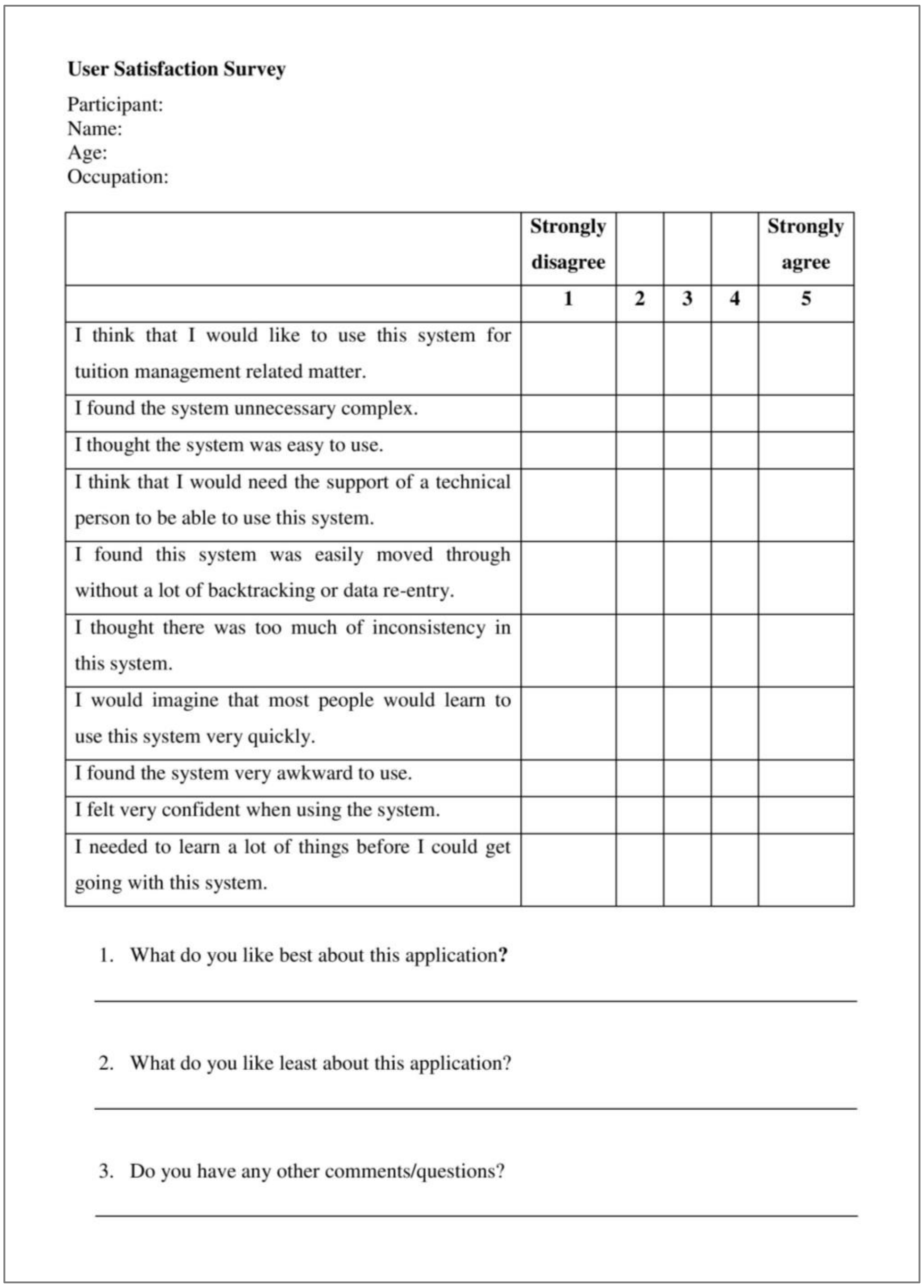
4. Any searching criteria that are important for you to search for an ideal tutor that is not provided in this application?

5. What do you like best about this application**?**

6. What do you like least about this application?

7. Do you have any other comments/questions?

Figure 7.27: User Satisfaction Survey Form – Tutor Seeker (Part 2)



User Satisfaction Survey

Participant:
Name:
Age:
Occupation:

	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management related matter.					
I found the system unnecessary complex.					
I thought the system was easy to use.					
I think that I would need the support of a technical person to be able to use this system.					
I found this system was easily moved through without a lot of backtracking or data re-entry.					
I thought there was too much of inconsistency in this system.					
I would imagine that most people would learn to use this system very quickly.					
I found the system very awkward to use.					
I felt very confident when using the system.					
I needed to learn a lot of things before I could get going with this system.					

1. What do you like best about this application**?**
2. What do you like least about this application?
3. Do you have any other comments/questions?

Figure 7.28: User Satisfaction Survey Form – Admin and Registered Tutor

7.5.3 Usability Test Result Analysis

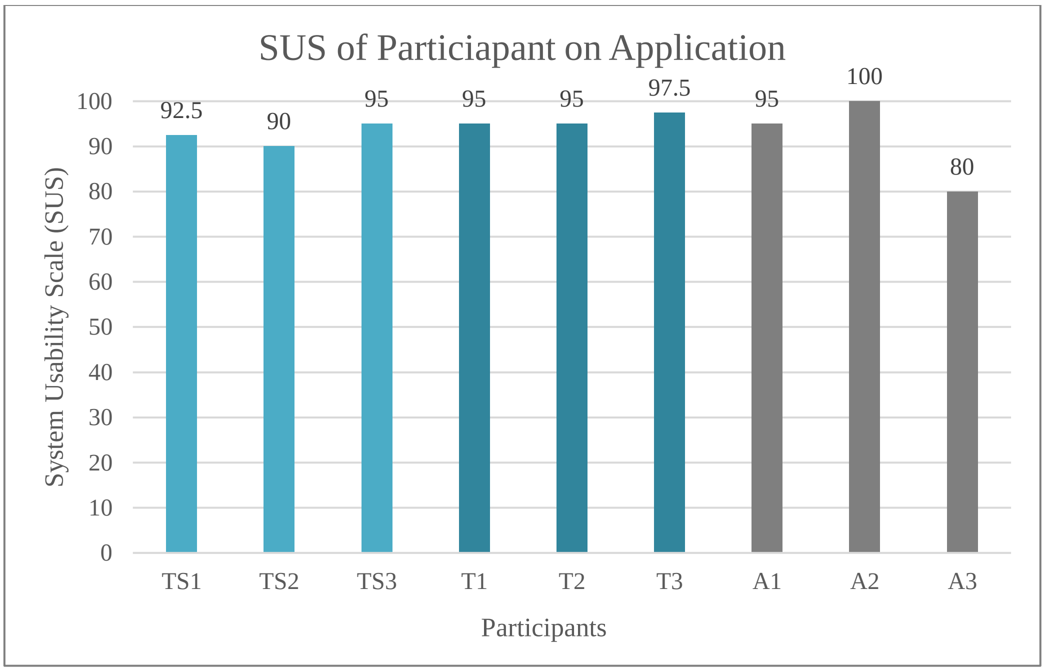


Figure 7.29: SUS of Participant on Application

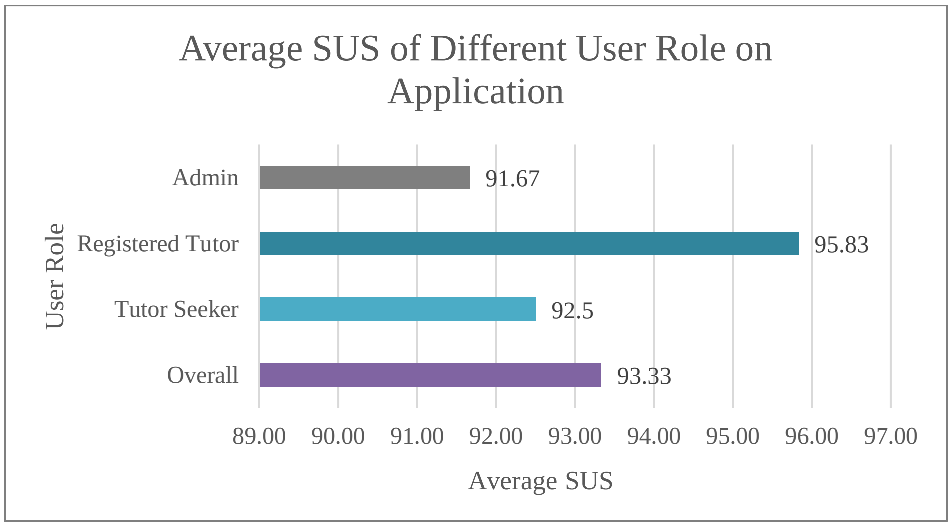


Figure 7.30: SUS of Participant on Application

The System Usability Scale is the metric to indicate the usability of an application toward the users while the passing SUS shall equal to and above 68. The figure above shows the SUS of 9 participants on this intelligent mobile private tutor finders application and an average SUS of different user roles on the application. Each SUS achieved a minimum of 68 while the average SUS from users achieved 93.33 which indicates that this application possesses features of high usability and user-friendliness. The SUS of the admin role which is 91.67 is lower when compared with the registered tutor and tutor seeker which are 95.83 and 92.5 respectively. Therefore, the feedbacks from the admin have to be prioritized in order to improve the usability within the admin role while the other users' feedback should not be neglected. The results of the usability test from participants are attached as an appendix to this report.

Table 7.31: Matching Rate of Similarity Measures' Search Results

Participant	**Top 5 Result**					**Top 10 Result**				
	R1	R2	R3	R4	R5	R1	R2	R3	R4	R5
TS1	2	2	2	2	**4**	4	4	4	4	**8**
TS2	3	3	2	2	**4**	4	4	4	5	**7**
TS3	3	3	2	3	**4**	4	3	3	4	**7**

R1: Result 1 (Manhattan Distance) **R3:** Result 3 (Minkowski Distance)
R2: Result 2 (Euclidean Distance) **R4:** Result 4 (Jaccard Similarity Coefficient)
R5: Result 5 (Cosine Similarity)

The 3 participants in conducting the usability testing of the tutor seeker role fulfilled the targeted user criteria which are a parent, primary school student, and, secondary school student. These 3 types of targeted users chose result 5 as the result that suits their expectations the most which there are more tutors on the returned result list that suit their ideal tutor's criteria. Result 5 is generated by applying the Cosine similarity measures which calculate the angle differences between two objects' coordinates. Cosine similarity measure has an obvious outstanding matched result than the other 4 similarity measures while the other 4 similarity measures perform similarly without a huge difference that is shown as the matched result above. Jaccard performs better when compared with Manhattan, Euclidean, and Minkowski in the top 10 results. As the literature review stated, Manhattan performed better than Euclidean when handling high dimension data (more features) comparison. Therefore, the priority (high to low) in applying the similarity measures methods in the searching function in this application should be Cosine Similarity, Jaccard Similarity Coefficient, Manhattan Distance, Euclidean Distance, and, Minkowski Distance.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

8.1 Conclusions

In conclusion, the objectives and requirements that are set at the initial stage of this project are achieved. The intelligent mobile private tutor finders application is developed while it enables the tutor seeker to search their ideal tutor according to their learning preferences. The searching function applied the similarity measures method while eliminating the SQL exact matching method to provide a more quantitative metric to tutor seekers in choosing their ideal tutor. The SUS of this application achieved an average of 93.33.

According to the literature review that is conducted, the similarity measures method is selected as the solution to improve the accuracy of search results which includes Manhattan Distance, Euclidean Distance, Minkowski Distance, and Jaccard Coefficient Similarity, and Cosine Similarity. Then, the application is planned to be developed by using a prototyping model. At the same time, an online survey is conducted to collect the tutor selection criteria of users from different age-group. After the analysis is implemented on the collected data, there are 11 criteria is applied as the search criteria in this application to easier tutor seeker in finding the real match ideal tutor which includes teaching syllabus, teaching language, teaching style, teaching experience, teaching area, teaching approach, education level, class type, tuition fee, attended institution and certificate obtained. The application is designed as a three-tier architecture to separate the presentation tier, application tier, and data tier. After the application is developed, unit testing, API testing, UAT, and usability testing are conducted to test the functionality and non-functionality performance of the application.

According to the feedback from testing, Cosine Similarity, Jaccard Similarity Coefficient, and Manhattan are selected as the final similarity measure method in the search function of this application. Tutor seekers could use this application to search for tutors, chat with a tutor, and raise for a demo or formal class request. Tutors could treat this application as a platform to expose themselves and manage their classes.

8.2 Limitations and Future Enhancement

There are some limitations is found after receiving feedback from end-user that could be fixed and enhanced in the future for this project.

i. Only focus on tutor searching

There is a suggestion from a tutor seeker some enhancement features such as online teaching, and monitoring students system could be integrated into this application as an online teaching platform that helps in finding tutors and conducting classes, and monitoring assignments from tutors for both parent and tutor side.

ii. Lack of payment handling

This application only provides a platform to contact the tutor while payment handling is not included. This could give a chance for the malicious user in getting the bank account of the tutor to perform a malicious action. Therefore, payment handling should be provided to guarantee the benefits of the tutor and tutor seeker. The tutor seekers could make payments by using the application without asking for the account number of the tutor.

iii. Lack of data visualization display

The admin side only provides a basic CRUD action to manage the users while lacks data visualization to track the data of the application. For example, the class registration rate could be collected with the tutor and tutor seeker data and displayed in a graph after being analyzed by using an algorithm. This analysis graph could help the admin track the popularity of tutors, usage of the application, and so on to implement strategies to target more users to use the application.

In a nutshell, the enhancement can be implemented and integrated in the future. After these enhancements are conducted, this application can be considered a one-stop online tutoring system that provides the services of finding tutors, purchasing classes, conducting classes, and monitoring classes. Besides, the admin side could have a more systematic analysis and controls on the users in order to attract more users and make it a profit platform by charging a commission on each transaction that is conducted via this platform.

REFERENCES

Accurate Reviews, 2020. *The best free Agile Software*. [online] Available at: <https://www.accuratereviews.com/free-agile-software/> [Accessed 9 Jul. 2021].

Aggarwal, C.C., Hinneburg, A. and Keim, D.A., 2001. On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1973(June), pp.420–434.

Alshamrani, A. and Bahattab, A., 2015. A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. *IJCSI International Journal of Computer Science Issues*, [online] 12(1), pp.106–111. Available at: <https://www.academia.edu/10793943/A_Comparison_Between_Three_SDLC_Models_Waterfall_Model_Spiral_Model_and_Incremental_Iterative_Model> [Accessed 8 Jul. 2021].

Awad, M.A., 2005. *A Comparison between Agile and Traditional Software Development Methodologies*. [online] *Global Journal of Computer Science and Technology*. Crawley. Available at: <https://www.academia.edu/38895564/A_Comparison_between_Agile_and_Traditional_Software_Development_Methodologies> [Accessed 7 Jul. 2021].

Babich, N., 2019. *Top 7 Usability Testing Methods*. [online] Available at: <https://xd.adobe.com/ideas/process/user-testing/top-7-usability-testing-methods/> [Accessed 10 Jul. 2021].

Balaji, S. and Murugaiyan, D.M.S., 2012. Waterfall vs v-model vs agile : A comparative study on SDLC. *WATEERFALL Vs V-MODEL Vs AGILE : A COMPARATIVE STUDY ON SDLC*, [online] 2(1), pp.26–30. Available at: <https://mediaweb.saintleo.edu/Courses/COM430/M2Readings/WATEERFALLVs V-MODEL Vs AGILE A COMPARATIVE STUDY ON SDLC.pdf> [Accessed 8 Jul. 2021].

Berger, H., Beynon-Davies, P. and Cleary, P., 2004. The Utility of a Rapid Application Development (RAD) Approach for a Large Complex Information Systems Development. In: *ECIS 2004 Proceedings*. [online] Wales: European Conference on Information Systems (ECIS).p.7. Available at: <http://aisel.aisnet.org/ecis2004/7/> [Accessed 9 Jul. 2021].

Bora, M.D.J. and Gupta, D.A.K., 2014. Effect of Different Distance Measures on the Performance of K-Means Algorithm: An Experimental Study in Matlab. [online] 5(2), pp.2501–2506. Available at: <http://arxiv.org/abs/1405.7471> [Accessed 3 Jul. 2021].

Bray, M., 1999. *The shadow education system: private tutoring and its implications for planners*. 2nd ed. [online] *Economics of Education Review*. French: UNESCO International Institute for Educational Planning (IIEP). Available at: <https://www.researchgate.net/publication/44839572_The_Shadow_education_system_private_tutoring_and_its_implications_for_planners> [Accessed 18 Jun. 2021].

Brooke, J., 1995. SUS: A 'Quick and Dirty' Usability Scale. *Usability Evaluation In Industry*, [online] (July), pp.207–212. Available at: <https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale> [Accessed 13 Jul. 2021].

Brownlee, J., 2020. *No Title4 Distance Measures for Machine Learning*. [online] Available at: <https://machinelearningmastery.com/distance-measures-for-machine-learning/> [Accessed 3 Jul. 2021].

Carew, J., 2020. *How to choose between a rules-based vs. machine learning system*. [online] Available at: <https://www.techtarget.com/searchenterpriseai/feature/How-to-choose-between-a-rules-based-vs-machine-learning-system> [Accessed 8 May 2022].

Chandra, V., 2015. Comparison between Various Software Development Methodologies. *International Journal of Computer Applications*, [online] 131(9), pp.7–10. Available at: <https://www.ijcaonline.org/research/volume131/number9/chandra-2015-ijca-907294.pdf> [Accessed 9 Jul. 2021].

DeepAI, 2016. *What is the Jaccard Index?* [online] Available at: <https://deepai.org/machine-learning-glossary-and-terms/jaccard-index> [Accessed 4 Jul. 2021].

Department of Statistics Malaysia, 2021. *Principal Statistics of Tuition Centres by State*. [online] Available at: <https://www.data.gov.my/data/ms_MY/dataset/principal-statistics-of-tuition-centres-by-state> [Accessed 18 Jun. 2021].

Despa, M.L., 2014. Comparative Study on Agile software development methodologies. *Database Systems Journal*, [online] 5(3), pp.37–56. Available at: <https://dbjournal.ro/archive/17/17_4.pdf> [Accessed 1 Jul. 2021].

Dima, A.M. and Maassen, M.A., 2018. From waterfall to agile software: Development models in the IT sector, 2006 to 2018. impacts on company management. *Journal of International Studies*, [online] 11(2), pp.315–326. Available at: <https://jois.eu/files/21_557_Dima.pdf> [Accessed 9 Jul. 2021].

Dubey, M.A., Jain, M.A. and Mantri, M.A., 2015. Comparative Study: Waterfall V/S Agile Model. *International Journal of Engineering Sciences & Research Technology*, [online] 4(3), pp.70–75. Available at:

<https://www.academia.edu/11483288/COMPARATIVE_STUDY_WATERFALL_V_S_AGILE_MODEL> [Accessed 9 Jul. 2021].

Elastic NV, 2021. *SQL Limitations*. [online] Available at: <https://www.elastic.co/guide/en/elasticsearch/reference/current/sql-limitations.html> [Accessed 2 Jul. 2021].

Esplin, C., 2016. *What is Firebase?* [online] Available at: <https://howtofirebase.com/what-is-firebase-fcb8614ba442> [Accessed 1 Jul. 2021].

Fatimah, D.D.S., Supriatna, A.D. and Kurniawati, R., 2018. Design of personnel information systems using rapid application development method. In: *MATEC Web of Conferences*. [online] Garut: MATEC Web of Conferences. Available at: <https://www.matec-conferences.org/articles/matecconf/abs/2018/56/matecconf_aasec2018_03016/matecconf_aasec2018_03016.html> [Accessed 9 Jul. 2021].

Firebase, 2021a. *Firebase Realtime Database*. [online] Available at: <https://firebase.google.com/docs/database>.

Firebase, 2021b. *Privacy and Security in Firebase*. [online] Available at: <https://firebase.google.com/support/privacy> [Accessed 1 Jul. 2021].

Gallagher, A., Dunleavy, J. and Reeves, P., 2019. *The Agile Method: Everything you need to know*. [online] Available at: <https://developer.ibm.com/devpractices/devops/articles/agile-method-everything-you-need-to-know/> [Accessed 9 Jul. 2021].

Gatsou, C., Politis, A. and Zevgolis, D., 2013. Exploring inexperienced user performance of a mobile tablet application through usability testing. In: *2013 Federated Conference on Computer Science and Information Systems, FedCSIS 2013*. [online] Krakow: IEEE.pp.557–564. Available at: <https://search-ebscohost-com.libezp2.utar.edu.my/login.aspx?direct=true&db=edseee&AN=edseee.6644056&site=eds-live&scope=site> [Accessed 11 Jul. 2021].

Grootendorst, M., 2021. *9 Distance Measures in Data Science*. [online] Available at: <https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa> [Accessed 4 Jul. 2021].

Han, J., Kamber, M. and Pei, J., 2012. Getting to Know Your Data. In: D. Cerra, ed. *Data mining: Data mining concepts and techniques,* 3rd ed. [online] Waltham: Morgan Kaufmann.pp.39–82. Available at: <https://doi.org/10.1016/B978-0-12-381479-1.00002-2> [Accessed 4 Jul. 2021].

Harrison, O., 2018. *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. [online] Available at: <https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761> [Accessed 4 Jul.

2021].

Helland, P., 2016. The singular success of SQL. *Communications of the ACM*, 59(8), pp.38–41.

hotjar, 2021. *The different types of usability testing methods for your projects*. [online] Available at: <https://www.hotjar.com/usability-testing/methods/> [Accessed 10 Jul. 2021].

IBM Cloud Education, 2020. *Three-Tier Architecture*. [online] Available at: <https://www.ibm.com/cloud/learn/three-tier-architecture> [Accessed 22 Mar. 2022].

Islam, M.R., Islam, M.R. and Mazumder, T.A., 2010. Mobile application and its global impact. *International Journal of Engineering & ...*, [online] 10(06), pp.104–111. Available at: <http://ijens.org/107506-0909 IJET-IJENS.pdf> [Accessed 17 Jun. 2021].

Javatpoint, 2021. *K-Nearest Neighbor(KNN) Algorithm for Machine Learning*. [online] Available at: <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning> [Accessed 4 Jul. 2021].

Kenayathulla, H.B. and Ubbudari, M., 2017. Private Tutoring in Malaysia: the Nexus Between Policy, People and Place. *Malaysian Online Journal of Educational Management*, [online] 5(2), pp.42–59. Available at: <http://mojem.um.edu.my/filebank/published_article/11335/(page 42-59) PRIVATE TUTORING .pdf> [Accessed 18 Jun. 2021].

Lu, W., Zhao, X., Sun, C., Chen, R. and Duan, G., 2020. A Novel Weighted Integration Dynamic Time Regularization and Euclidean Distance Optimization Algorithm for Power Data Mining. In: *2020 IEEE 6th International Conference on Computer and Communications, ICCC 2020*. [online] Cheng Du: IEEE.pp.2033–2037. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/9345154/authors#authors> [Accessed 3 Jul. 2021].

M.Barnum, C., 2021. Establishing the essentials. In: A. Akeh and C. Hockaday, eds. *Usability Testing Essentials: Ready, Set ...Test!* [online] Cambridge: Elsevier Inc.pp.9–31. Available at: <https://books.google.com.my/books?hl=zh-CN&lr=&id=L6_SDwAAQBAJ&oi=fnd&pg=PP1&dq=usability++testing+&ots=k AcKU6C6zd&sig=iNYoVZQYE1RslbY2KfP6hO66DwA#v=onepage&q=usability testing&f=false> [Accessed 10 Jul. 2021].

Moran, K., 2019. *Usability Testing 101*. [online] Available at: <https://www.nngroup.com/articles/usability-testing-101/> [Accessed 10 Jul. 2021].

Myalapalli, V.K. and Teja, B.L.R., 2015. High Performance PL / SQL Programming. [online] Pune: IEEE. Available at: <https://ieeexplore-ieee-org.libezp2.utar.edu.my/document/7087001/authors#authors> [Accessed 2 Jul. 2021].

Netguru, 2021. *What Is React Native? Complex Guide for 2021*. [online] Available at: <https://www.netguru.com/glossary/react-native#benefits-of-react-native> [Accessed 16 Jul. 2021].

Nishanthi, R., 2020. Understanding of the Importance of Mother Tongue Learning. *ISSN: 2456-6470*, [online] 5(1), pp.77–80. Available at: <https://www.ijtsrd.com/humanities-and-the-arts/sociology/35846/understanding-of-the-importance-of-mother-tongue-learning/rajathurai-nishanthi> [Accessed 18 Jun. 2021].

Niwattanakul, S., Singthongchai, J., Naenudorn, E. and Wanapu, S., 2013. Using of jaccard coefficient for keywords similarity. In: *Lecture Notes in Engineering and Computer Science*. [online] Hong Kong: International Association of Engineers (IAENG).pp.380–384. Available at: <http://www.iaeng.org/publication/IMECS2013/IMECS2013_pp380-384.pdf> [Accessed 4 Jul. 2021].

OECD, 2020. Strengthening online learning when schools are closed. *Tackling Coronavirus (COVID-19): Contributing to a global effort*, [online] pp.1–14. Available at: <https://www.oecd.org/coronavirus/policy-responses/strengthening-online-learning-when-schools-are-closed-the-role-of-families-and-teachers-in-supporting-students-during-the-covid-19-crisis-c4ecba6c/#boxsection-d1e29> [Accessed 8 May 2022].

playbook ux, 2021. *10 Popular Usability Testing Methods*. [online] Available at: <https://www.playbookux.com/10-popular-usability-testing-methods/> [Accessed 10 Jul. 2021].

Polamuri, S., 2015. *FIVE MOST POPULAR SIMILARITY MEASURES IMPLEMENTATION IN PYTHON*. [online] Available at: <https://dataaspirant.com/five-most-popular-similarity-measures-implementation-in-python/> [Accessed 3 Jul. 2015].

Ragunath, P., Velmourougan, S., Davachelvan, P., Kayalvizhi, S. and Ravimohan, R., 2010. Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle (SDLC). *International Journal of Computer Science and Network Security*, [online] 10(1), pp.112–119. Available at: <http://paper.ijcsns.org/07_book/201001/20100115.pdf> [Accessed 8 Jul. 2021].

Rahim, S., Chowdhury, A.E., Nandi, D. and Rahman, M., 2018. ScrumFall: A Hybrid Software Process Model. *International Journal of Information Technology and Computer Science*, [online] 10(12), pp.41–48. Available at: <http://www.mecs-press.org/ijitcs/ijitcs-v10-n12/v10n12-6.html> [Accessed 9 Jul. 2021].

Sauro, J., 2011. *Measuring Usability with the System Usability Scale (SUS)*. [online] Available at: <https://measuringu.com/sus/> [Accessed 13 Jul. 2021].

scikit-learn, 2020. *sklearn.neighbors.DistanceMetric*. [online] Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.DistanceMetric.html#sklearn.neighbors.DistanceMetric> [Accessed 4 Jul. 2021].

Sharma, N., 2019. *Importance of Distance Metrics in Machine Learning Modelling*. [online] Available at: <https://towardsdatascience.com/importance-of-distance-metrics-in-machine-learning-modelling-e51395ffe60d> [Accessed 3 Jul. 2021].

Shirkhorshidi, A.S., Aghabozorgi, S. and Ying Wah, T., 2015. A Comparison study on similarity and dissimilarity measures in clustering continuous data. *PLoS ONE*, [online] 10(12), pp.1–20. Available at: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0144059#sec001> [Accessed 3 Jul. 2021].

Simelane, L. and Zuva, T., 2019. Decision Support Framework for the Adoption of Software Development Methodologies. In: *Proceedings - 2019 International Multidisciplinary Information Technology and Engineering Conference, IMITEC 2019*. [online] Vanderbijlpark: IEEE. Available at: <https://search-ebscohost-com.libezp2.utar.edu.my/login.aspx?direct=true&db=edseee&AN=edseee.9015859&site=eds-live&scope=site> [Accessed 7 Jul. 2021].

Snyder, H., 2019. Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, [online] 104(August), pp.333–339. Available at: <https://doi.org/10.1016/j.jbusres.2019.07.039> [Accessed 21 Jul. 2021].

Susanto, A. and Meiryani, 2019. System Development Method with The Prototype Method. *International Journal of Scientific and Technology Research*, [online] 8(7), pp.141–144. Available at: <http://www.ijstr.org/final-print/july2019/System-Development-Method-With-The-Prototype-Method.pdf> [Accessed 9 Jul. 2021].

The Government of Malaysia, 2019. *Mendapat Pendidikan Formal*. [online] Available at: <https://www.malaysia.gov.my/portal/content/29545> [Accessed 18 Jun. 2021].

tutorialspoint, 2021. *SDLC - RAD Model*. [online] Available at: <https://www.tutorialspoint.com/sdlc/sdlc_rad_model.htm> [Accessed 9 Jul. 2021].

Uniqtech, 2020. *No TitleUnderstand Jaccard Index, Jaccard Similarity in Minutes*. [online] Available at: <https://medium.com/data-science-bootcamp/understand-jaccard-index-jaccard-similarity-in-minutes-25a703fbf9d7> [Accessed 4 Jul. 2021].

usability.gov, 2021a. *System Usability Scale (SUS)*. [online] Available at: <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html> [Accessed 13 Jul. 2021].

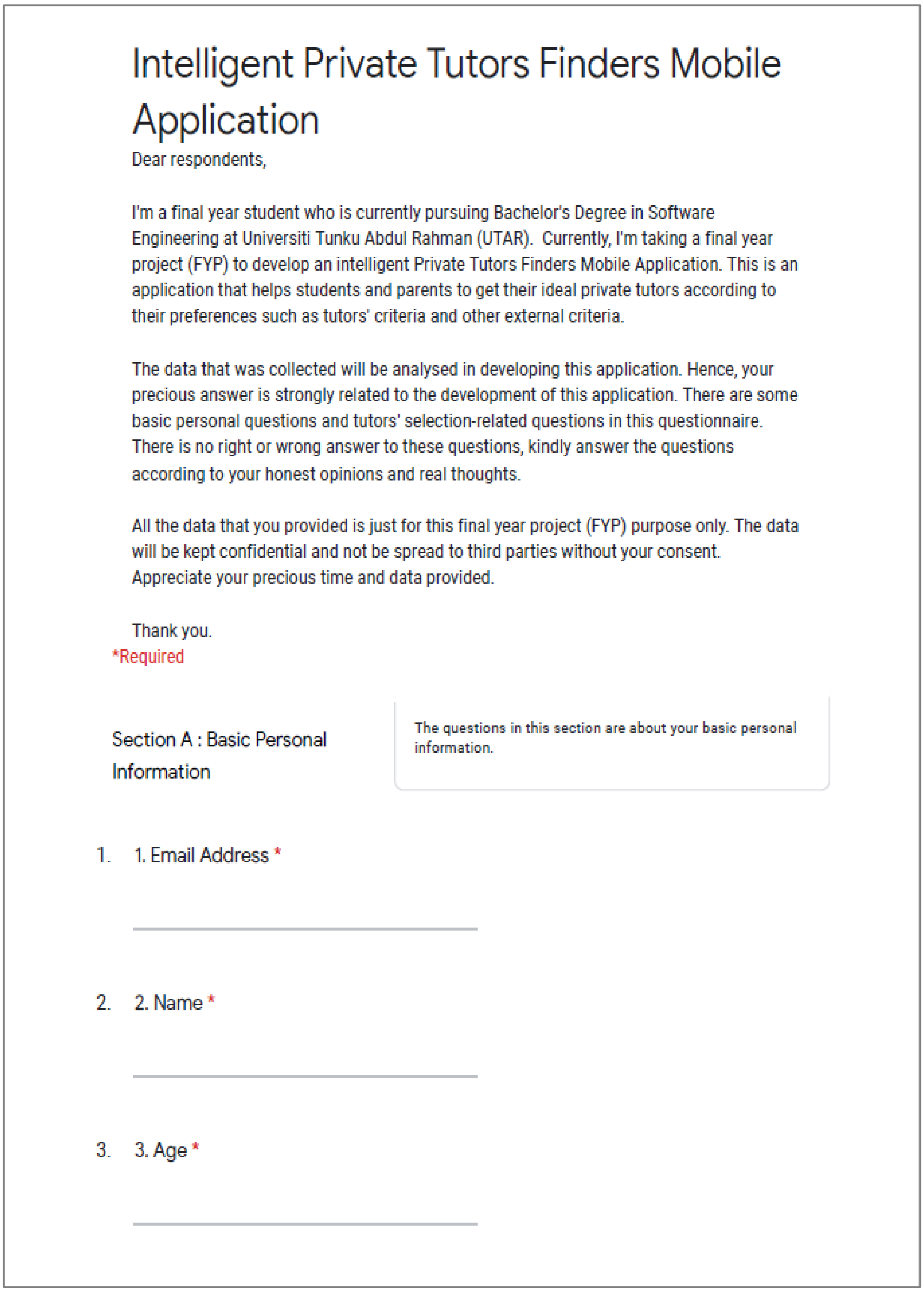
usability.gov, 2021b. *Usability Testing*. [online] Available at: <https://www.usability.gov/how-to-and-tools/methods/usability-testing.html> [Accessed 10 Jul. 2021].

W3Schools, 2021a. *Introduction to SQL*. [online] Available at: <https://www.w3schools.com/sql/sql_intro.asp> [Accessed 2 Jul. 2021].

W3Schools, 2021b. *SQL GROUP BY Statement*. [online] Available at: <https://www.w3schools.com/sql/sql_groupby.asp> [Accessed 2 Jul. 2021].

APPENDICES

APPENDIX A: Questionnaire



Intelligent Private Tutors Finders Mobile Application

Dear respondents,

I'm a final year student who is currently pursuing Bachelor's Degree in Software Engineering at Universiti Tunku Abdul Rahman (UTAR). Currently, I'm taking a final year project (FYP) to develop an intelligent Private Tutors Finders Mobile Application. This is an application that helps students and parents to get their ideal private tutors according to their preferences such as tutors' criteria and other external criteria.

The data that was collected will be analysed in developing this application. Hence, your precious answer is strongly related to the development of this application. There are some basic personal questions and tutors' selection-related questions in this questionnaire. There is no right or wrong answer to these questions, kindly answer the questions according to your honest opinions and real thoughts.

All the data that you provided is just for this final year project (FYP) purpose only. The data will be kept confidential and not be spread to third parties without your consent. Appreciate your precious time and data provided.

Thank you.

*Required

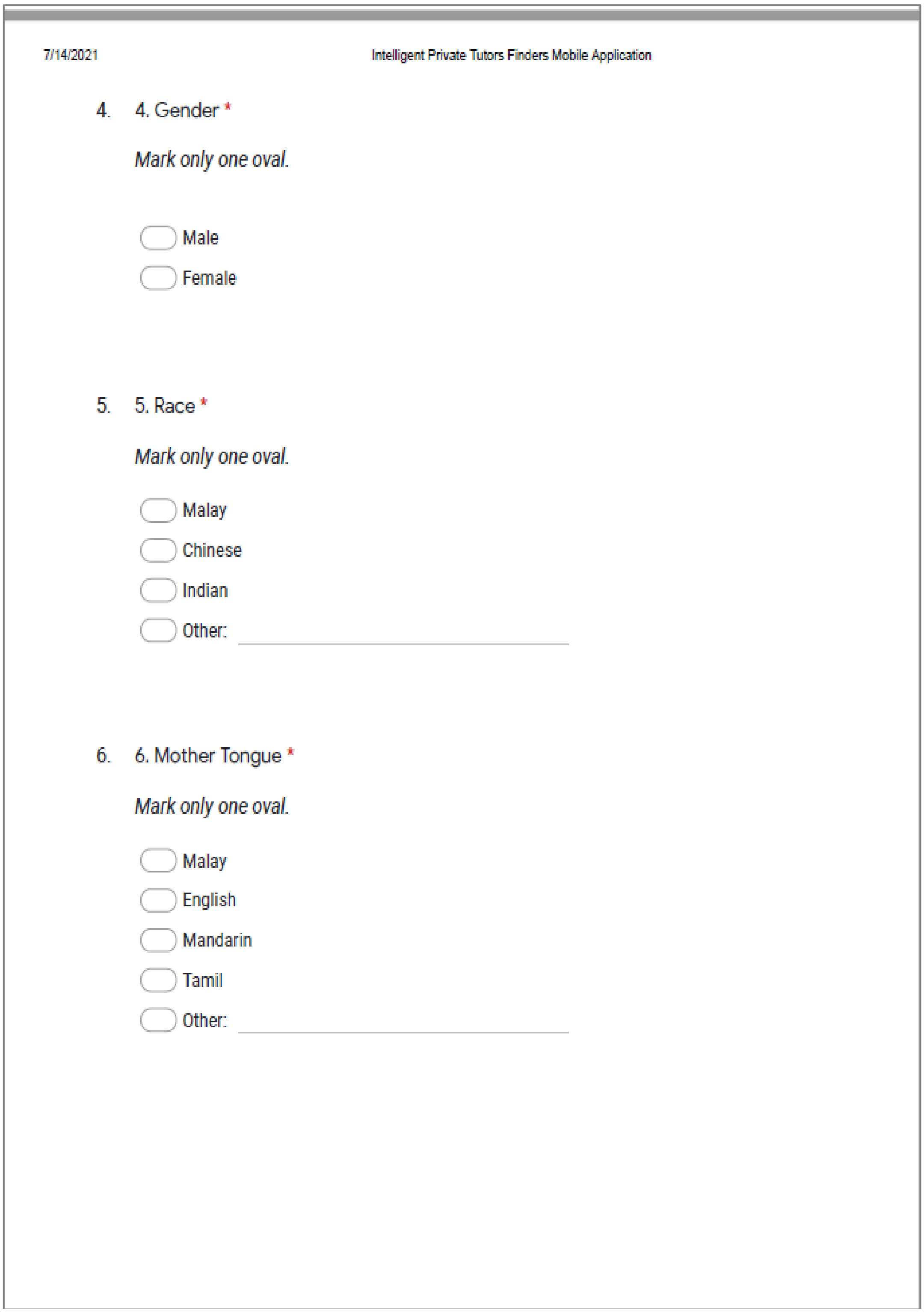
Section A : Basic Personal Information

The questions in this section are about your basic personal information.

1. 1. Email Address *

2. 2. Name *

3. 3. Age *



4. 4. Gender *

Mark only one oval.

◯ Male
◯ Female

5. 5. Race *

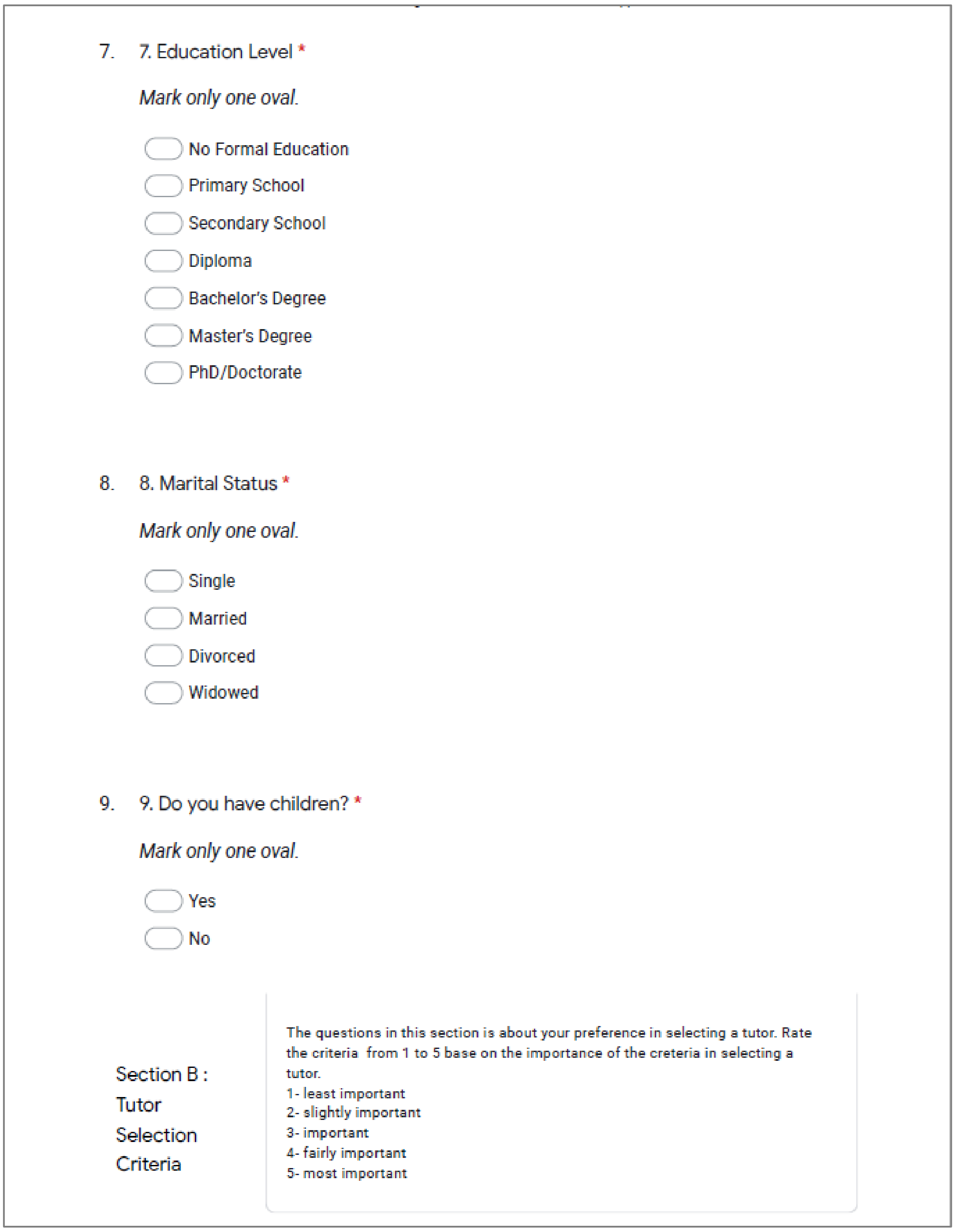
Mark only one oval.

◯ Malay
◯ Chinese
◯ Indian
◯ Other: ____________________

6. 6. Mother Tongue *

Mark only one oval.

◯ Malay
◯ English
◯ Mandarin
◯ Tamil
◯ Other: ____________________



7. 7. Education Level *

Mark only one oval.

- No Formal Education
- Primary School
- Secondary School
- Diploma
- Bachelor's Degree
- Master's Degree
- PhD/Doctorate

8. 8. Marital Status *

Mark only one oval.

- Single
- Married
- Divorced
- Widowed

9. 9. Do you have children? *

Mark only one oval.

- Yes
- No

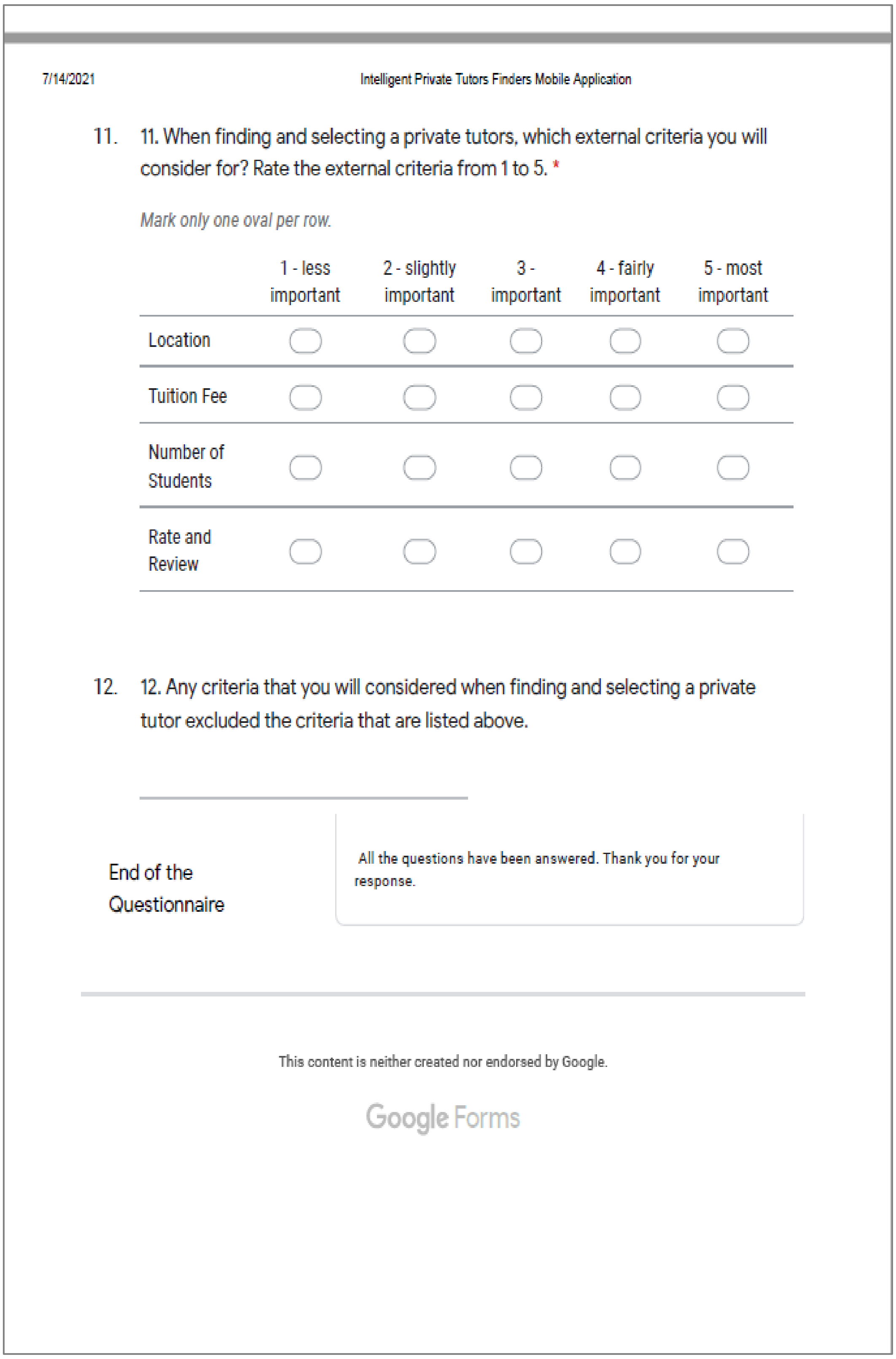
Section B : Tutor Selection Criteria

The questions in this section is about your preference in selecting a tutor. Rate the criteria from 1 to 5 base on the importance of the creteria in selecting a tutor.
1- least important
2- slightly important
3- important
4- fairly important
5- most important

10. 10. When finding and selecting a private tutors, which criteria of tutors you will consider for? Rate the criteria of tutors from 1 to 5. *

Mark only one oval per row.

	1 - less important	2 - slightly important	3 - important	4 - fairly important	5 - most important
Gender	◯	◯	◯	◯	◯
Race	◯	◯	◯	◯	◯
Religion	◯	◯	◯	◯	◯
Education Level	◯	◯	◯	◯	◯
Attended School/College/University (E.g. Graduated from University Malaya)	◯	◯	◯	◯	◯
Certificate Obtained	◯	◯	◯	◯	◯
Teaching Experiences	◯	◯	◯	◯	◯
Teaching Languages	◯	◯	◯	◯	◯
Teaching Styles (E.g. Humour/Relax/Strict/Active)	◯	◯	◯	◯	◯
Teaching Approach (E.g. One to One / Small Group / Large Group)	◯	◯	◯	◯	◯
Employment Status (E.g. Full Time Tutor/ Part Time Tutor)	◯	◯	◯	◯	◯
Tuition Type (E.g. Home Tuition/ Online Tuition)	◯	◯	◯	◯	◯



11. 11. When finding and selecting a private tutors, which external criteria you will consider for? Rate the external criteria from 1 to 5. *

Mark only one oval per row.

	1 - less important	2 - slightly important	3 - important	4 - fairly important	5 - most important
Location	◯	◯	◯	◯	◯
Tuition Fee	◯	◯	◯	◯	◯
Number of Students	◯	◯	◯	◯	◯
Rate and Review	◯	◯	◯	◯	◯

12. 12. Any criteria that you will considered when finding and selecting a private tutor excluded the criteria that are listed above.

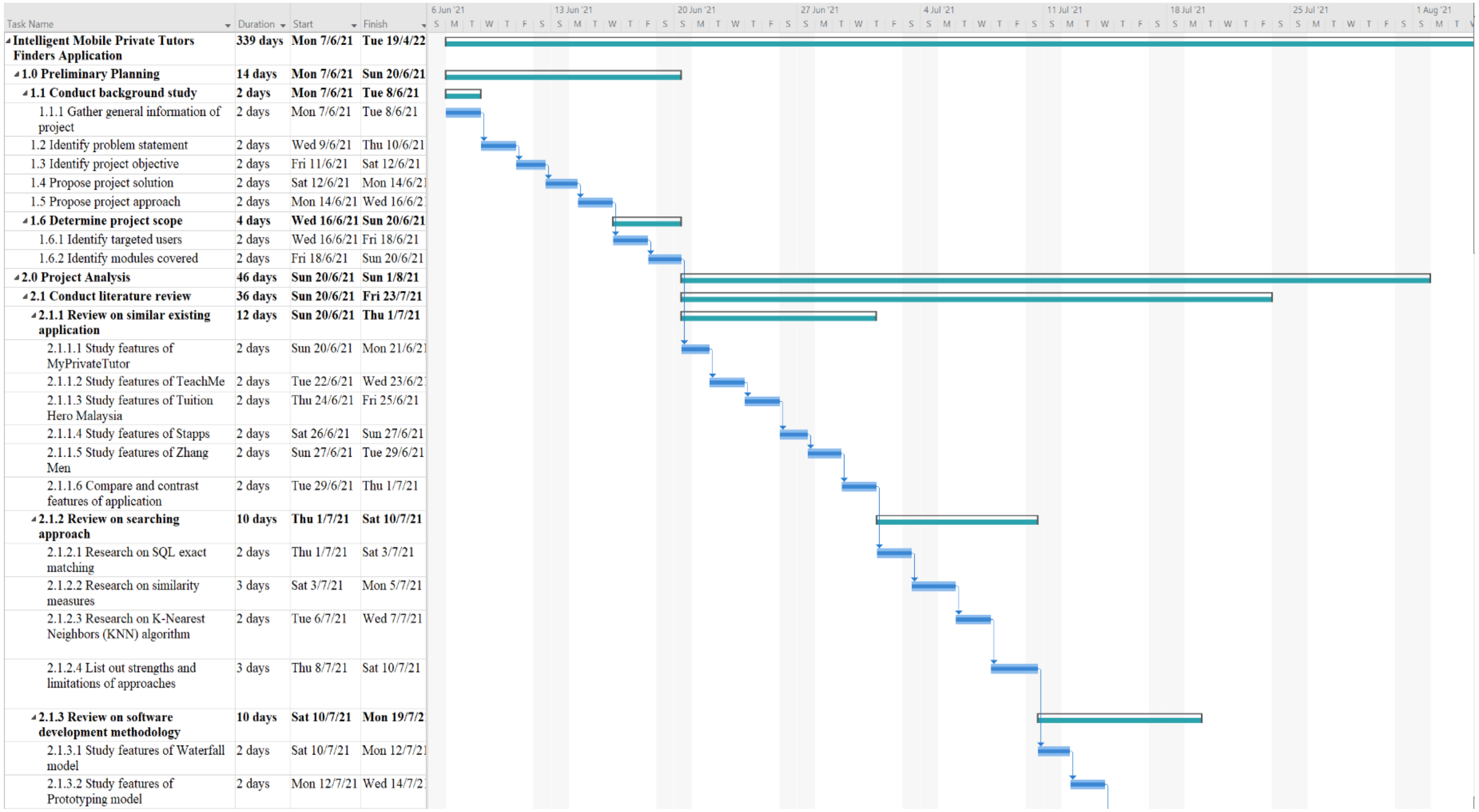
End of the Questionnaire

All the questions have been answered. Thank you for your response.

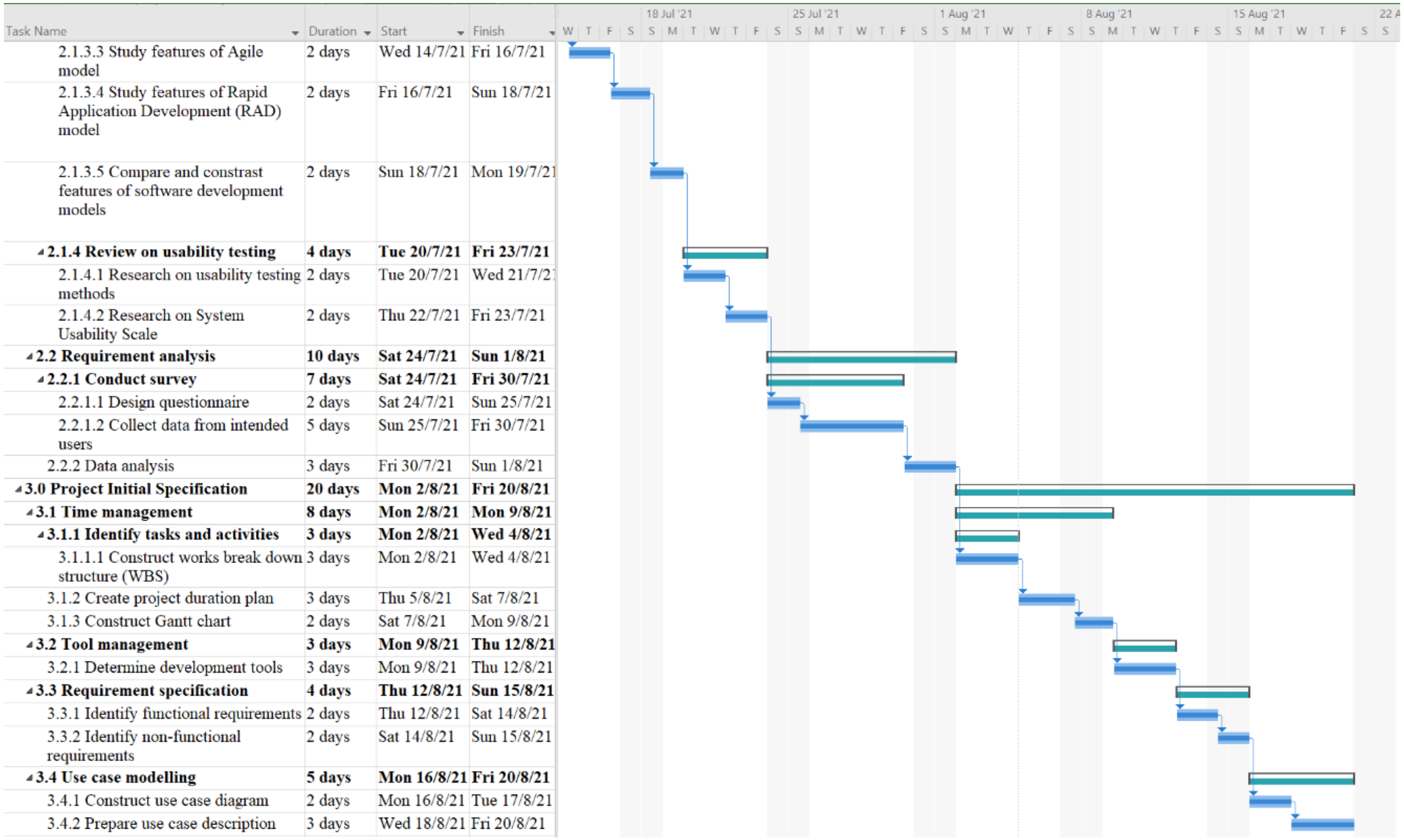
This content is neither created nor endorsed by Google.

Google Forms

APPENDIX B: Gantt Chart

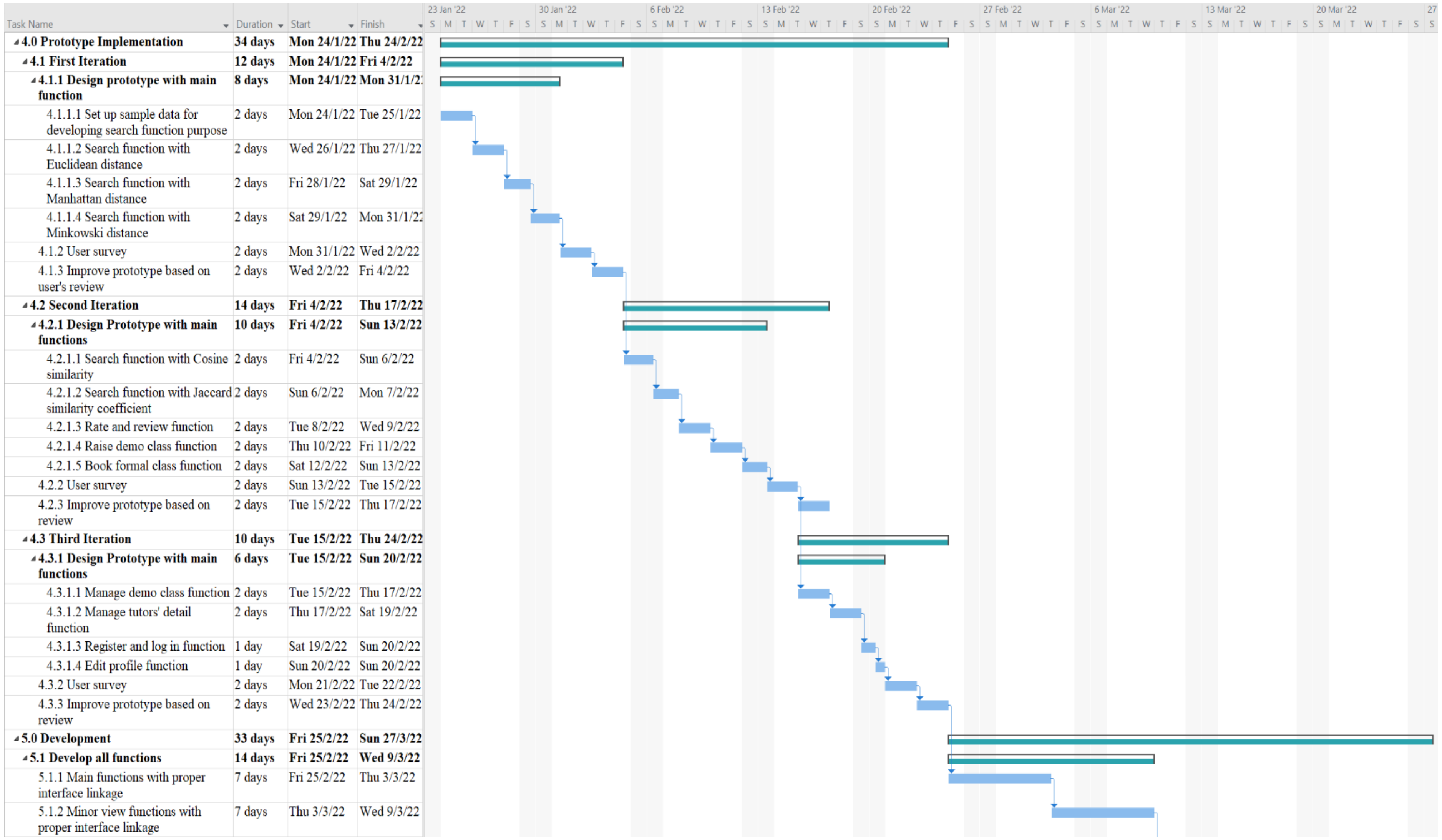


Task Name	Duration	Start	Finish
Intelligent Mobile Private Tutors Finders Application	**339 days**	**Mon 7/6/21**	**Tue 19/4/22**
1.0 Preliminary Planning	**14 days**	**Mon 7/6/21**	**Sun 20/6/21**
1.1 Conduct background study	**2 days**	**Mon 7/6/21**	**Tue 8/6/21**
1.1.1 Gather general information of project	2 days	Mon 7/6/21	Tue 8/6/21
1.2 Identify problem statement	2 days	Wed 9/6/21	Thu 10/6/21
1.3 Identify project objective	2 days	Fri 11/6/21	Sat 12/6/21
1.4 Propose project solution	2 days	Sat 12/6/21	Mon 14/6/21
1.5 Propose project approach	2 days	Mon 14/6/21	Wed 16/6/2
1.6 Determine project scope	**4 days**	**Wed 16/6/21**	**Sun 20/6/21**
1.6.1 Identify targeted users	2 days	Wed 16/6/21	Fri 18/6/21
1.6.2 Identify modules covered	2 days	Fri 18/6/21	Sun 20/6/21
2.0 Project Analysis	**46 days**	**Sun 20/6/21**	**Sun 1/8/21**
2.1 Conduct literature review	**36 days**	**Sun 20/6/21**	**Fri 23/7/21**
2.1.1 Review on similar existing application	**12 days**	**Sun 20/6/21**	**Thu 1/7/21**
2.1.1.1 Study features of MyPrivateTutor	2 days	Sun 20/6/21	Mon 21/6/21
2.1.1.2 Study features of TeachMe	2 days	Tue 22/6/21	Wed 23/6/2
2.1.1.3 Study features of Tuition Hero Malaysia	2 days	Thu 24/6/21	Fri 25/6/21
2.1.1.4 Study features of Stapps	2 days	Sat 26/6/21	Sun 27/6/21
2.1.1.5 Study features of Zhang Men	2 days	Sun 27/6/21	Tue 29/6/21
2.1.1.6 Compare and contrast features of application	2 days	Tue 29/6/21	Thu 1/7/21
2.1.2 Review on searching approach	**10 days**	**Thu 1/7/21**	**Sat 10/7/21**
2.1.2.1 Research on SQL exact matching	2 days	Thu 1/7/21	Sat 3/7/21
2.1.2.2 Research on similarity measures	3 days	Sat 3/7/21	Mon 5/7/21
2.1.2.3 Research on K-Nearest Neighbors (KNN) algorithm	2 days	Tue 6/7/21	Wed 7/7/21
2.1.2.4 List out strengths and limitations of approaches	3 days	Thu 8/7/21	Sat 10/7/21
2.1.3 Review on software development methodology	**10 days**	**Sat 10/7/21**	**Mon 19/7/2**
2.1.3.1 Study features of Waterfall model	2 days	Sat 10/7/21	Mon 12/7/21
2.1.3.2 Study features of Prototyping model	2 days	Mon 12/7/21	Wed 14/7/2

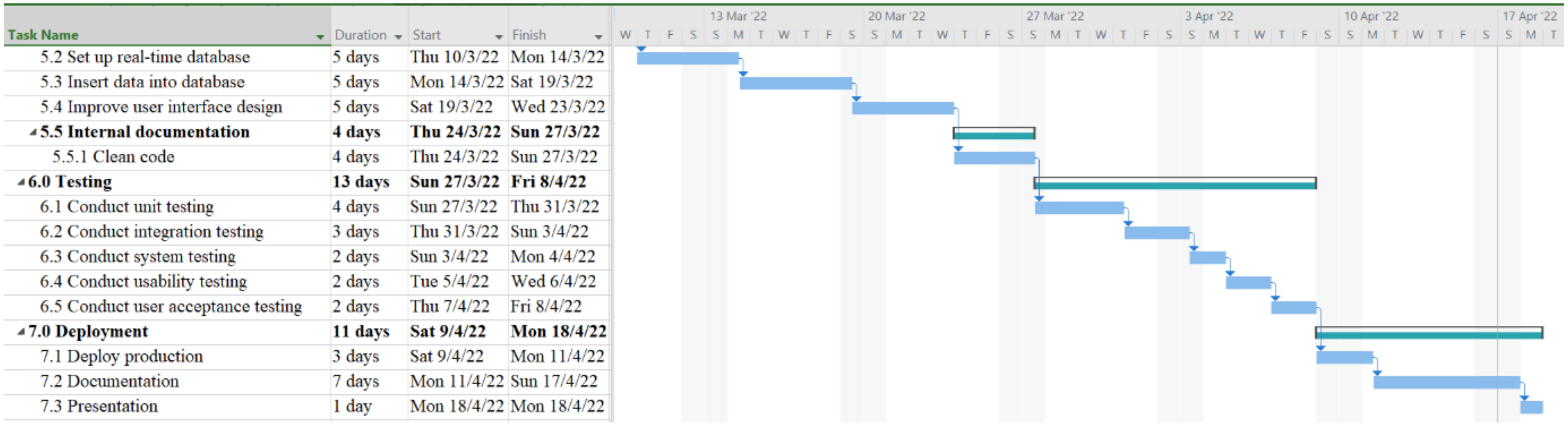


Task Name	Duration	Start	Finish
2.1.3.3 Study features of Agile model	2 days	Wed 14/7/21	Fri 16/7/21
2.1.3.4 Study features of Rapid Application Development (RAD) model	2 days	Fri 16/7/21	Sun 18/7/21
2.1.3.5 Compare and constrast features of software development models	2 days	Sun 18/7/21	Mon 19/7/21
2.1.4 Review on usability testing	**4 days**	**Tue 20/7/21**	**Fri 23/7/21**
2.1.4.1 Research on usability testing methods	2 days	Tue 20/7/21	Wed 21/7/2
2.1.4.2 Research on System Usability Scale	2 days	Thu 22/7/21	Fri 23/7/21
2.2 Requirement analysis	**10 days**	**Sat 24/7/21**	**Sun 1/8/21**
2.2.1 Conduct survey	**7 days**	**Sat 24/7/21**	**Fri 30/7/21**
2.2.1.1 Design questionnaire	2 days	Sat 24/7/21	Sun 25/7/21
2.2.1.2 Collect data from intended users	5 days	Sun 25/7/21	Fri 30/7/21
2.2.2 Data analysis	3 days	Fri 30/7/21	Sun 1/8/21
3.0 Project Initial Specification	**20 days**	**Mon 2/8/21**	**Fri 20/8/21**
3.1 Time management	**8 days**	**Mon 2/8/21**	**Mon 9/8/21**
3.1.1 Identify tasks and activities	**3 days**	**Mon 2/8/21**	**Wed 4/8/21**
3.1.1.1 Construct works break down structure (WBS)	3 days	Mon 2/8/21	Wed 4/8/21
3.1.2 Create project duration plan	3 days	Thu 5/8/21	Sat 7/8/21
3.1.3 Construct Gantt chart	2 days	Sat 7/8/21	Mon 9/8/21
3.2 Tool management	**3 days**	**Mon 9/8/21**	**Thu 12/8/21**
3.2.1 Determine development tools	3 days	Mon 9/8/21	Thu 12/8/21
3.3 Requirement specification	**4 days**	**Thu 12/8/21**	**Sun 15/8/21**
3.3.1 Identify functional requirements	2 days	Thu 12/8/21	Sat 14/8/21
3.3.2 Identify non-functional requirements	2 days	Sat 14/8/21	Sun 15/8/21
3.4 Use case modelling	**5 days**	**Mon 16/8/21**	**Fri 20/8/21**
3.4.1 Construct use case diagram	2 days	Mon 16/8/21	Tue 17/8/21
3.4.2 Prepare use case description	3 days	Wed 18/8/21	Fri 20/8/21

18 Jul '21
25 Jul '21
1 Aug '21
8 Aug '21
15 Aug '21
W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S



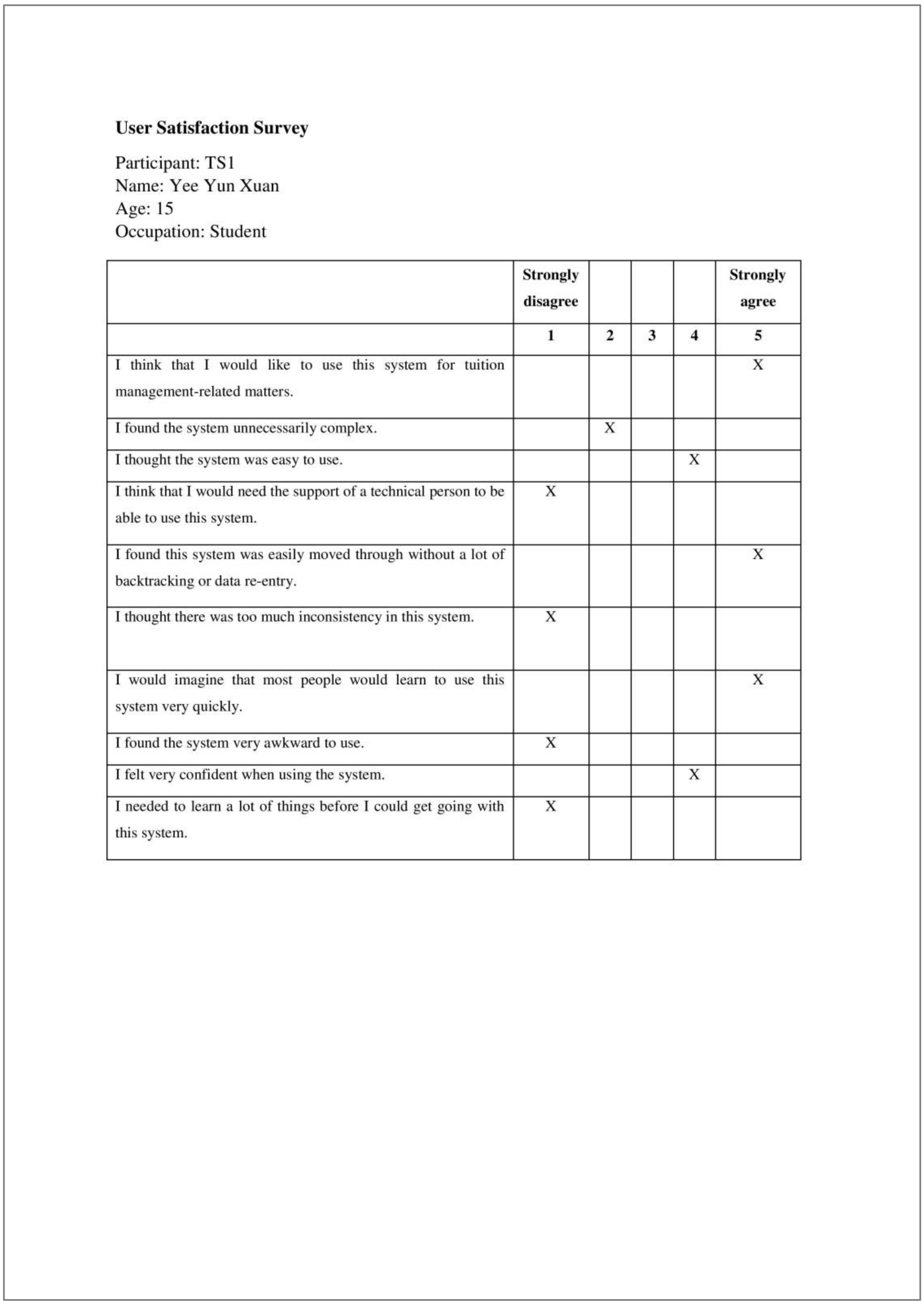
Task Name	Duration	Start	Finish
4.0 Prototype Implementation	**34 days**	**Mon 24/1/22**	**Thu 24/2/22**
4.1 First Iteration	**12 days**	**Mon 24/1/22**	**Fri 4/2/22**
4.1.1 Design prototype with main function	**8 days**	**Mon 24/1/22**	**Mon 31/1/2**
4.1.1.1 Set up sample data for developing search function purpose	2 days	Mon 24/1/22	Tue 25/1/22
4.1.1.2 Search function with Euclidean distance	2 days	Wed 26/1/22	Thu 27/1/22
4.1.1.3 Search function with Manhattan distance	2 days	Fri 28/1/22	Sat 29/1/22
4.1.1.4 Search function with Minkowski distance	2 days	Sat 29/1/22	Mon 31/1/2
4.1.2 User survey	2 days	Mon 31/1/22	Wed 2/2/22
4.1.3 Improve prototype based on user's review	2 days	Wed 2/2/22	Fri 4/2/22
4.2 Second Iteration	**14 days**	**Fri 4/2/22**	**Thu 17/2/22**
4.2.1 Design Prototype with main functions	**10 days**	**Fri 4/2/22**	**Sun 13/2/22**
4.2.1.1 Search function with Cosine similarity	2 days	Fri 4/2/22	Sun 6/2/22
4.2.1.2 Search function with Jaccard similarity coefficient	2 days	Sun 6/2/22	Mon 7/2/22
4.2.1.3 Rate and review function	2 days	Tue 8/2/22	Wed 9/2/22
4.2.1.4 Raise demo class function	2 days	Thu 10/2/22	Fri 11/2/22
4.2.1.5 Book formal class function	2 days	Sat 12/2/22	Sun 13/2/22
4.2.2 User survey	2 days	Sun 13/2/22	Tue 15/2/22
4.2.3 Improve prototype based on review	2 days	Tue 15/2/22	Thu 17/2/22
4.3 Third Iteration	**10 days**	**Tue 15/2/22**	**Thu 24/2/22**
4.3.1 Design Prototype with main functions	**6 days**	**Tue 15/2/22**	**Sun 20/2/22**
4.3.1.1 Manage demo class function	2 days	Tue 15/2/22	Thu 17/2/22
4.3.1.2 Manage tutors' detail function	2 days	Thu 17/2/22	Sat 19/2/22
4.3.1.3 Register and log in function	1 day	Sat 19/2/22	Sun 20/2/22
4.3.1.4 Edit profile function	1 day	Sun 20/2/22	Sun 20/2/22
4.3.2 User survey	2 days	Mon 21/2/22	Tue 22/2/22
4.3.3 Improve prototype based on review	2 days	Wed 23/2/22	Thu 24/2/22
5.0 Development	**33 days**	**Fri 25/2/22**	**Sun 27/3/22**
5.1 Develop all functions	**14 days**	**Fri 25/2/22**	**Wed 9/3/22**
5.1.1 Main functions with proper interface linkage	7 days	Fri 25/2/22	Thu 3/3/22
5.1.2 Minor view functions with proper interface linkage	7 days	Thu 3/3/22	Wed 9/3/22



Task Name	Duration	Start	Finish
5.2 Set up real-time database	5 days	Thu 10/3/22	Mon 14/3/22
5.3 Insert data into database	5 days	Mon 14/3/22	Sat 19/3/22
5.4 Improve user interface design	5 days	Sat 19/3/22	Wed 23/3/22
5.5 Internal documentation	**4 days**	**Thu 24/3/22**	**Sun 27/3/22**
5.5.1 Clean code	4 days	Thu 24/3/22	Sun 27/3/22
6.0 Testing	**13 days**	**Sun 27/3/22**	**Fri 8/4/22**
6.1 Conduct unit testing	4 days	Sun 27/3/22	Thu 31/3/22
6.2 Conduct integration testing	3 days	Thu 31/3/22	Sun 3/4/22
6.3 Conduct system testing	2 days	Sun 3/4/22	Mon 4/4/22
6.4 Conduct usability testing	2 days	Tue 5/4/22	Wed 6/4/22
6.5 Conduct user acceptance testing	2 days	Thu 7/4/22	Fri 8/4/22
7.0 Deployment	**11 days**	**Sat 9/4/22**	**Mon 18/4/22**
7.1 Deploy production	3 days	Sat 9/4/22	Mon 11/4/22
7.2 Documentation	7 days	Mon 11/4/22	Sun 17/4/22
7.3 Presentation	1 day	Mon 18/4/22	Mon 18/4/22

13 Mar '22
20 Mar '22
27 Mar '22
3 Apr '22
10 Apr '22
17 Apr '22
W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T

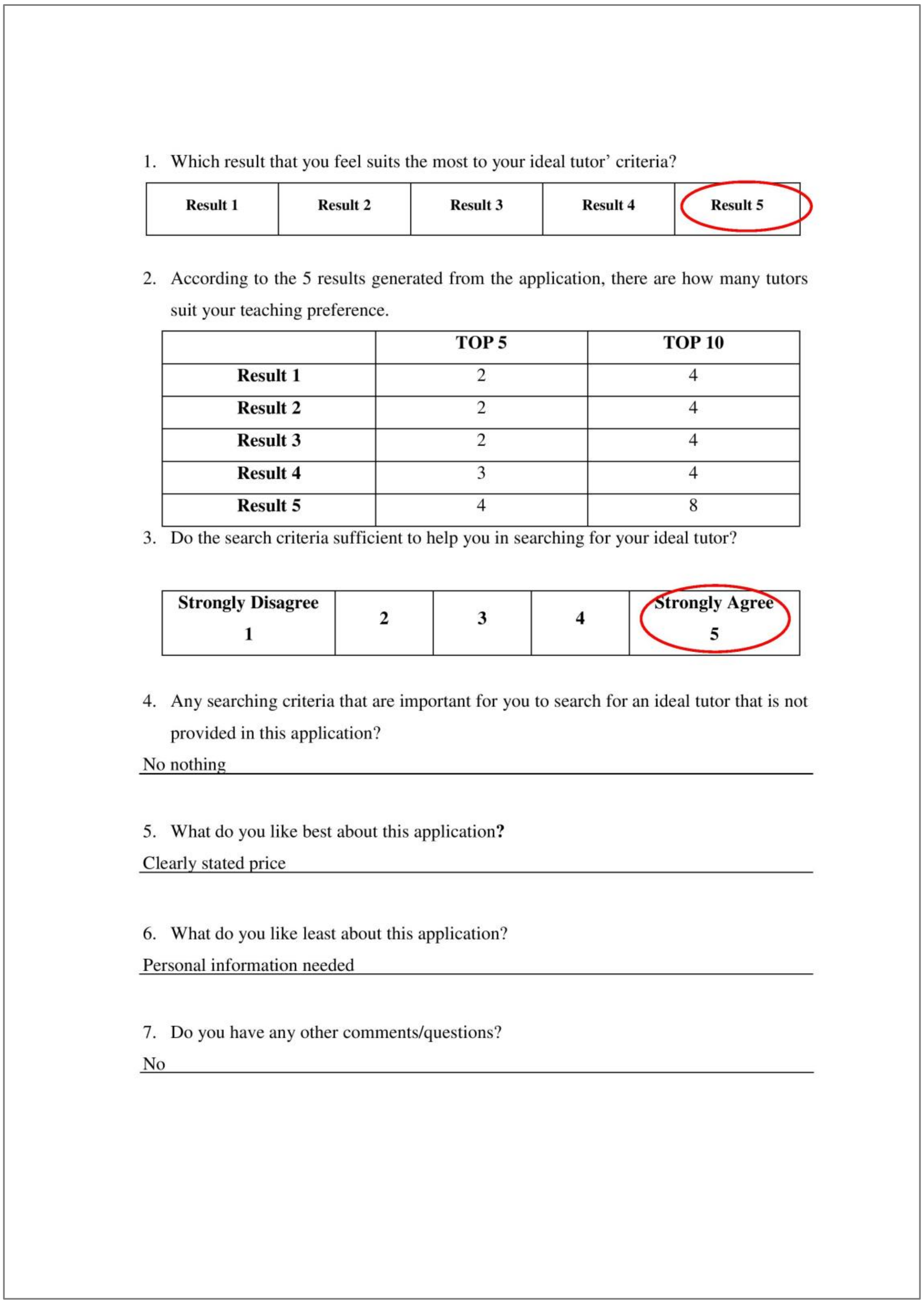
APPENDIX C: Usability Test Result



User Satisfaction Survey

Participant: TS1
Name: Yee Yun Xuan
Age: 15
Occupation: Student

	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management-related matters.					X
I found the system unnecessarily complex.		X			
I thought the system was easy to use.				X	
I think that I would need the support of a technical person to be able to use this system.	X				
I found this system was easily moved through without a lot of backtracking or data re-entry.					X
I thought there was too much inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly.					X
I found the system very awkward to use.	X				
I felt very confident when using the system.				X	
I needed to learn a lot of things before I could get going with this system.	X				



1. Which result that you feel suits the most to your ideal tutor' criteria?

Result 1	**Result 2**	**Result 3**	**Result 4**	[✓ **Result 5**]

2. According to the 5 results generated from the application, there are how many tutors suit your teaching preference.

	TOP 5	**TOP 10**
Result 1	2	4
Result 2	2	4
Result 3	2	4
Result 4	3	4
Result 5	4	8

3. Do the search criteria sufficient to help you in searching for your ideal tutor?

Strongly Disagree 1	**2**	**3**	**4**	[✓ **Strongly Agree 5**]

4. Any searching criteria that are important for you to search for an ideal tutor that is not provided in this application?

No nothing

5. What do you like best about this application**?**

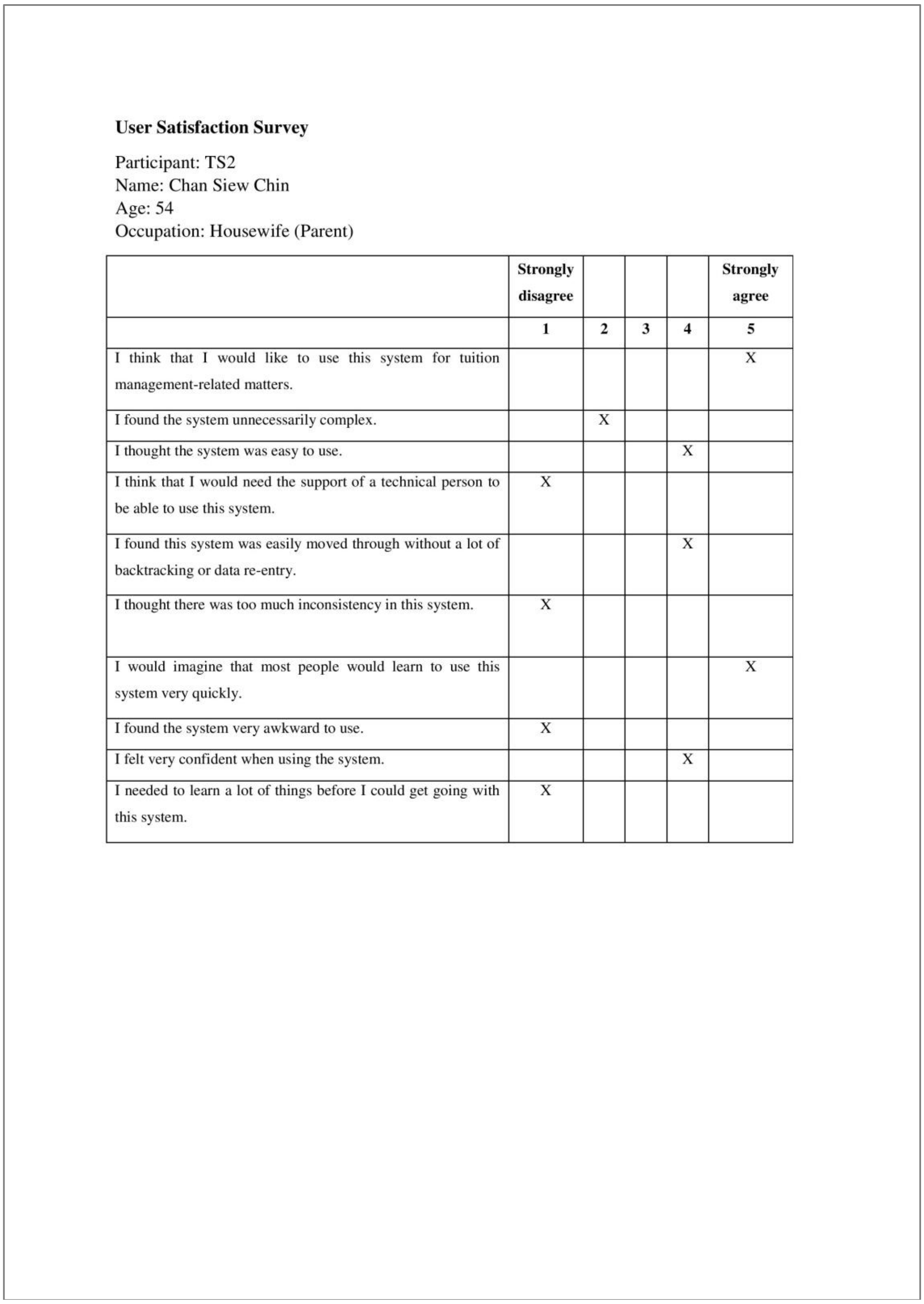
Clearly stated price

6. What do you like least about this application?

Personal information needed

7. Do you have any other comments/questions?

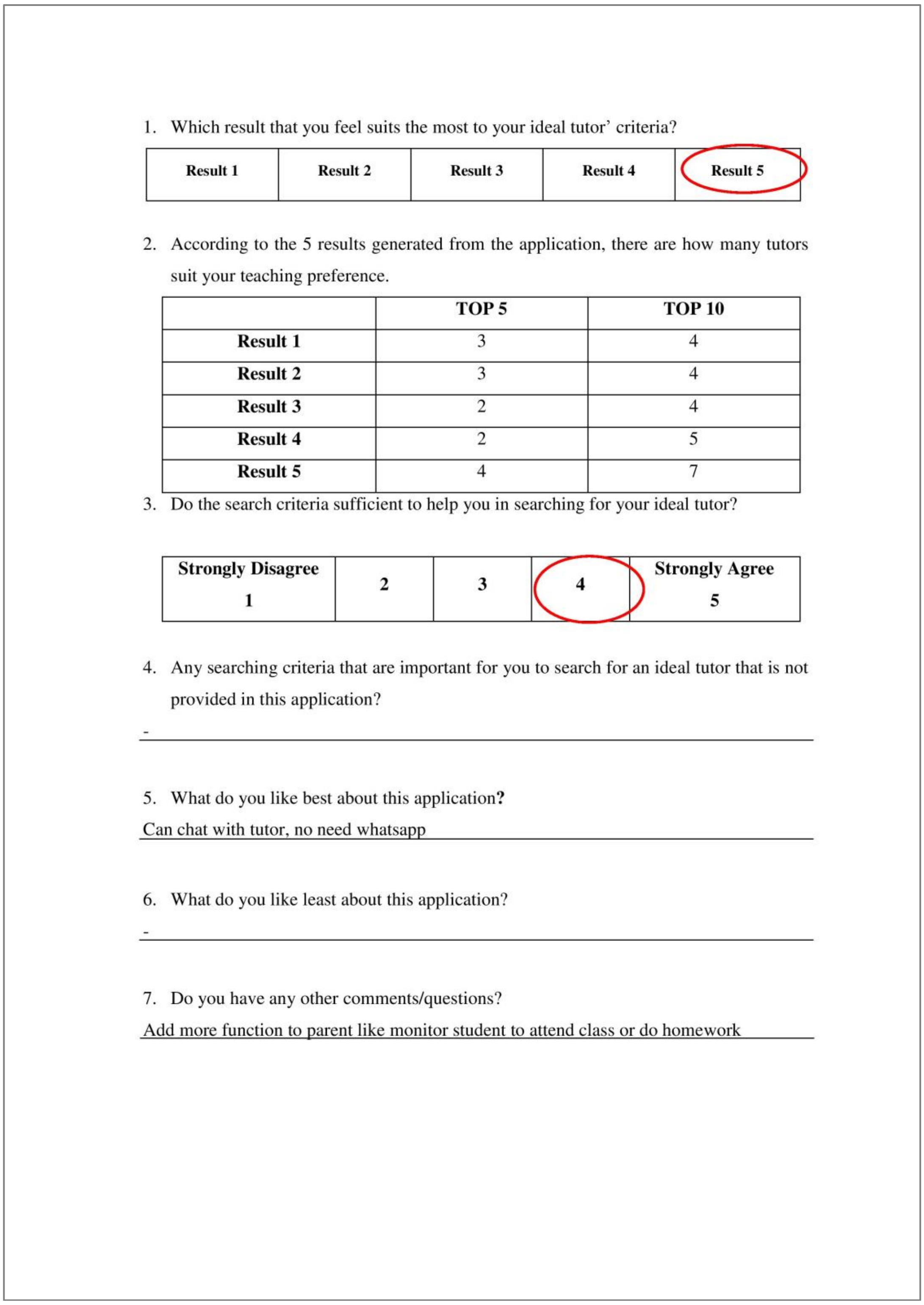
No



User Satisfaction Survey

Participant: TS2
Name: Chan Siew Chin
Age: 54
Occupation: Housewife (Parent)

	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management-related matters.					X
I found the system unnecessarily complex.		X			
I thought the system was easy to use.				X	
I think that I would need the support of a technical person to be able to use this system.	X				
I found this system was easily moved through without a lot of backtracking or data re-entry.				X	
I thought there was too much inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly.					X
I found the system very awkward to use.	X				
I felt very confident when using the system.				X	
I needed to learn a lot of things before I could get going with this system.	X				



1. Which result that you feel suits the most to your ideal tutor' criteria?

Result 1	**Result 2**	**Result 3**	**Result 4**	[✓ **Result 5**]

2. According to the 5 results generated from the application, there are how many tutors suit your teaching preference.

	TOP 5	TOP 10
Result 1	3	4
Result 2	3	4
Result 3	2	4
Result 4	2	5
Result 5	4	7

3. Do the search criteria sufficient to help you in searching for your ideal tutor?

Strongly Disagree 1	**2**	**3**	[✓ **4**]	**Strongly Agree 5**

4. Any searching criteria that are important for you to search for an ideal tutor that is not provided in this application?

-

5. What do you like best about this application**?**

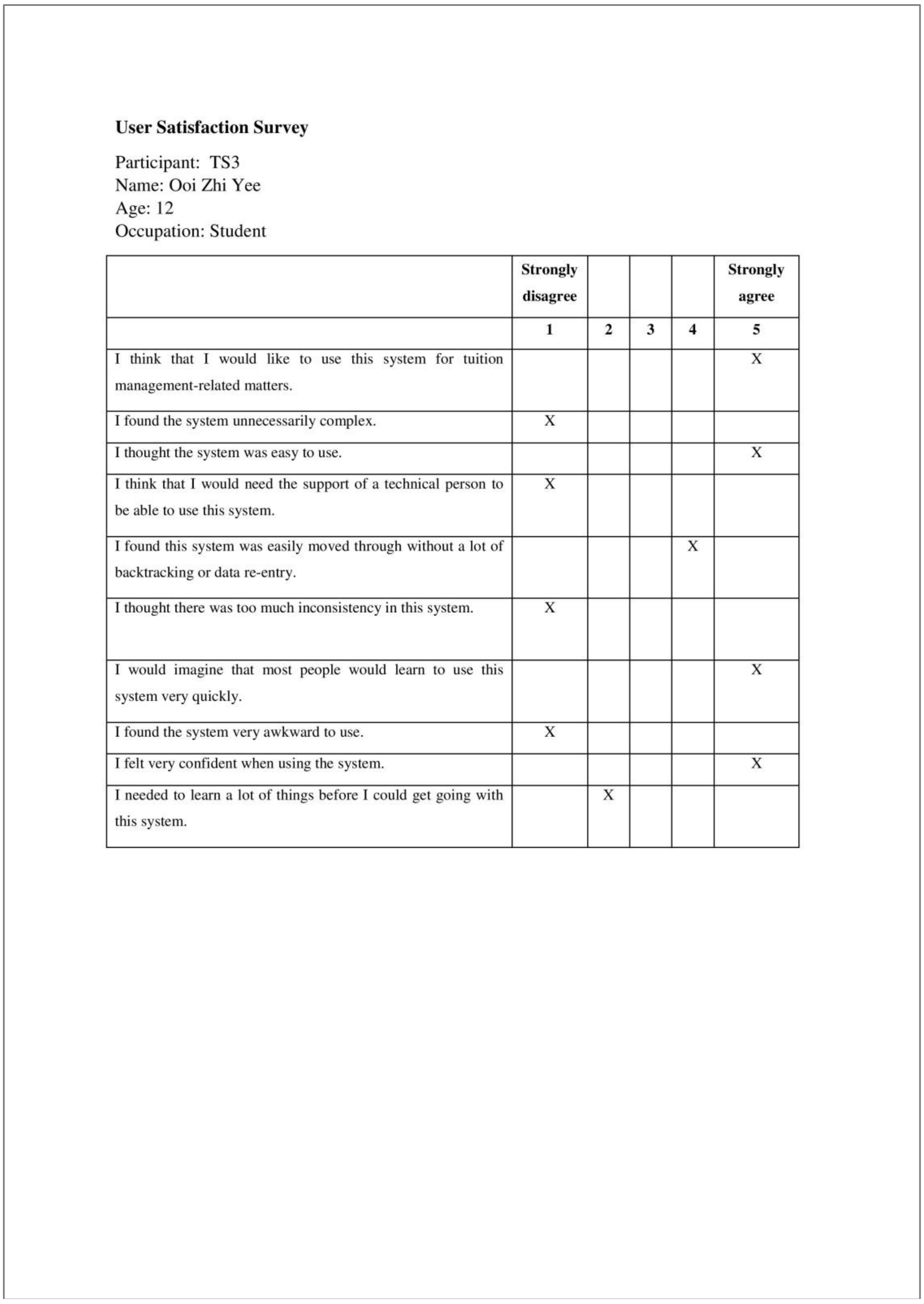
Can chat with tutor, no need whatsapp

6. What do you like least about this application?

-

7. Do you have any other comments/questions?

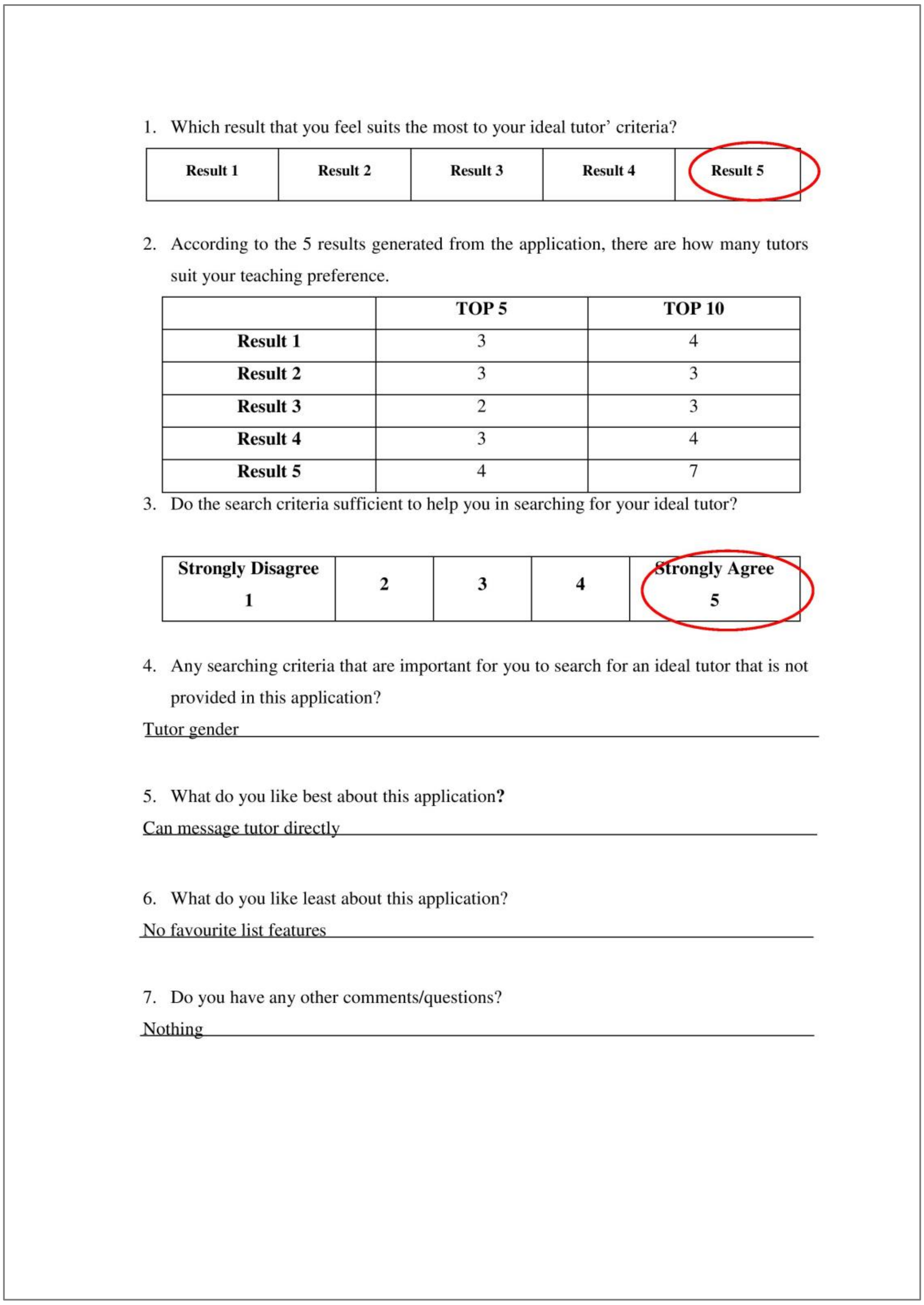
Add more function to parent like monitor student to attend class or do homework



User Satisfaction Survey

Participant: TS3
Name: Ooi Zhi Yee
Age: 12
Occupation: Student

	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management-related matters.					X
I found the system unnecessarily complex.	X				
I thought the system was easy to use.					X
I think that I would need the support of a technical person to be able to use this system.	X				
I found this system was easily moved through without a lot of backtracking or data re-entry.				X	
I thought there was too much inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly.					X
I found the system very awkward to use.	X				
I felt very confident when using the system.					X
I needed to learn a lot of things before I could get going with this system.		X			



1. Which result that you feel suits the most to your ideal tutor' criteria?

Result 1	**Result 2**	**Result 3**	**Result 4**	[✓ **Result 5**]

2. According to the 5 results generated from the application, there are how many tutors suit your teaching preference.

	TOP 5	**TOP 10**
Result 1	3	4
Result 2	3	3
Result 3	2	3
Result 4	3	4
Result 5	4	7

3. Do the search criteria sufficient to help you in searching for your ideal tutor?

Strongly Disagree 1	**2**	**3**	**4**	[✓ **Strongly Agree 5**]

4. Any searching criteria that are important for you to search for an ideal tutor that is not provided in this application?

Tutor gender

5. What do you like best about this application**?**

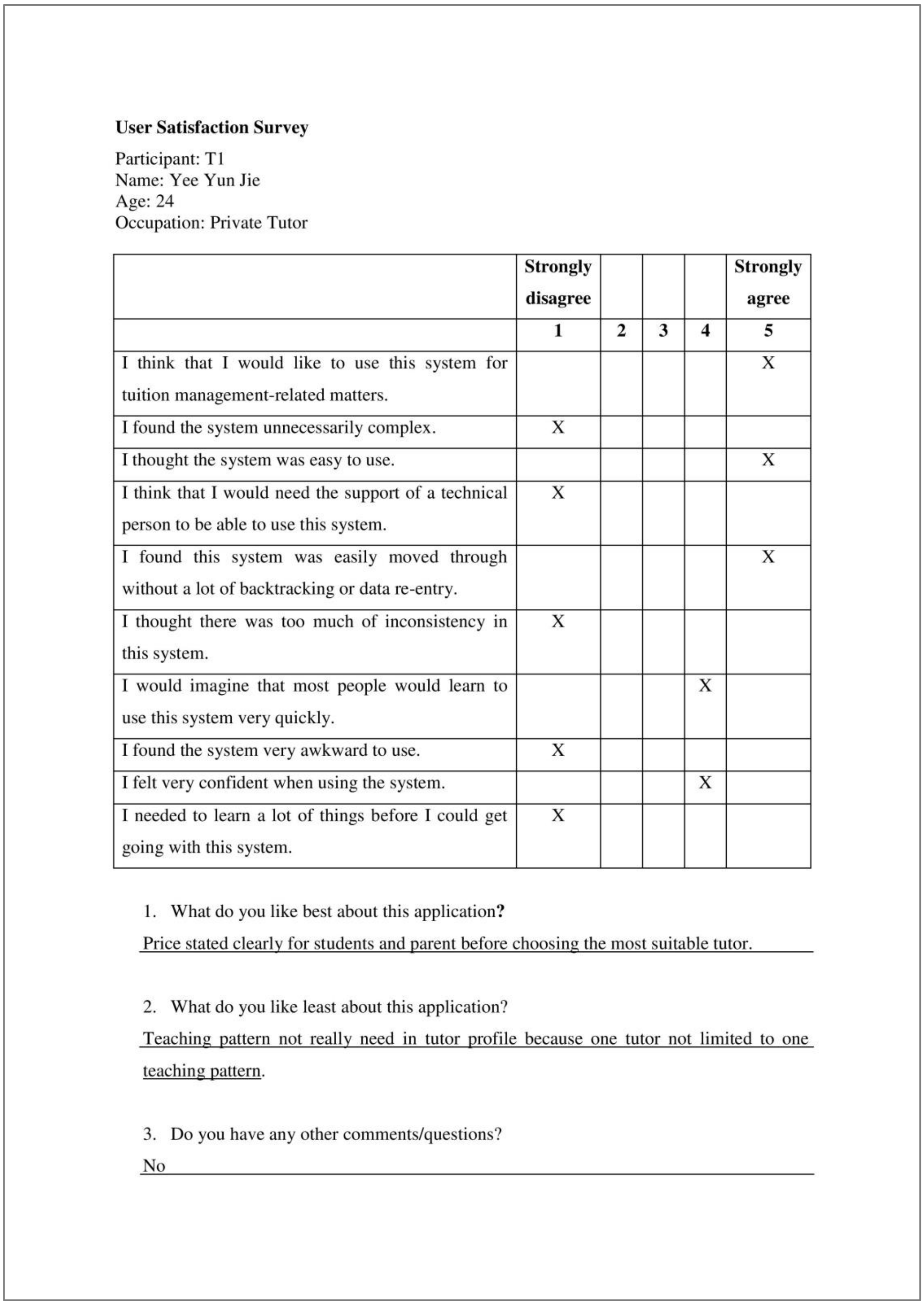
Can message tutor directly

6. What do you like least about this application?

No favourite list features

7. Do you have any other comments/questions?

Nothing



User Satisfaction Survey

Participant: T1
Name: Yee Yun Jie
Age: 24
Occupation: Private Tutor

	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management-related matters.					X
I found the system unnecessarily complex.	X				
I thought the system was easy to use.					X
I think that I would need the support of a technical person to be able to use this system.	X				
I found this system was easily moved through without a lot of backtracking or data re-entry.					X
I thought there was too much of inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly.				X	
I found the system very awkward to use.	X				
I felt very confident when using the system.				X	
I needed to learn a lot of things before I could get going with this system.	X				

1. What do you like best about this application**?**

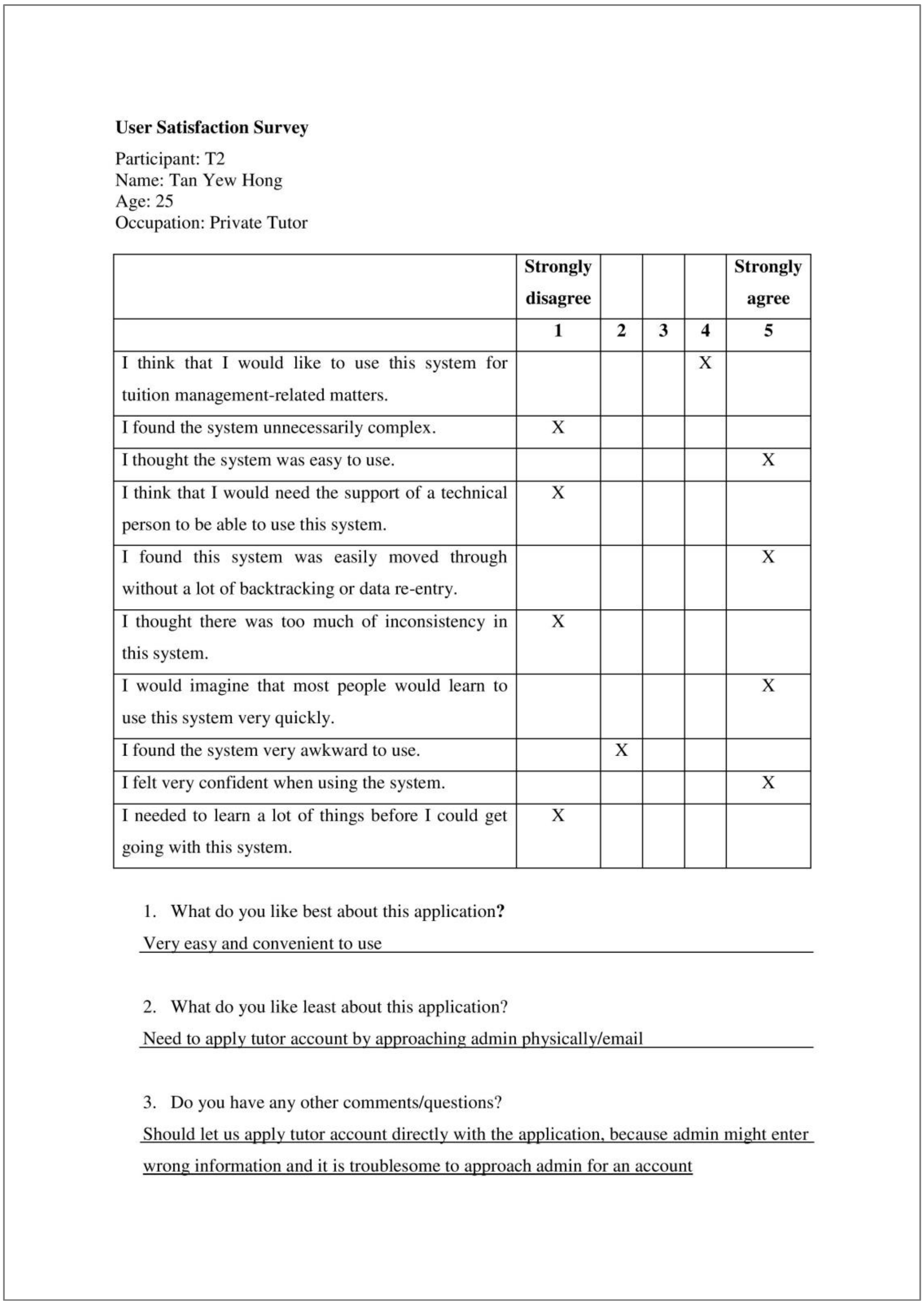
Price stated clearly for students and parent before choosing the most suitable tutor.

2. What do you like least about this application?

Teaching pattern not really need in tutor profile because one tutor not limited to one teaching pattern.

3. Do you have any other comments/questions?

No



User Satisfaction Survey

Participant: T2
Name: Tan Yew Hong
Age: 25
Occupation: Private Tutor

	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management-related matters.				X	
I found the system unnecessarily complex.	X				
I thought the system was easy to use.					X
I think that I would need the support of a technical person to be able to use this system.	X				
I found this system was easily moved through without a lot of backtracking or data re-entry.					X
I thought there was too much of inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly.					X
I found the system very awkward to use.		X			
I felt very confident when using the system.					X
I needed to learn a lot of things before I could get going with this system.	X				

1. What do you like best about this application**?**

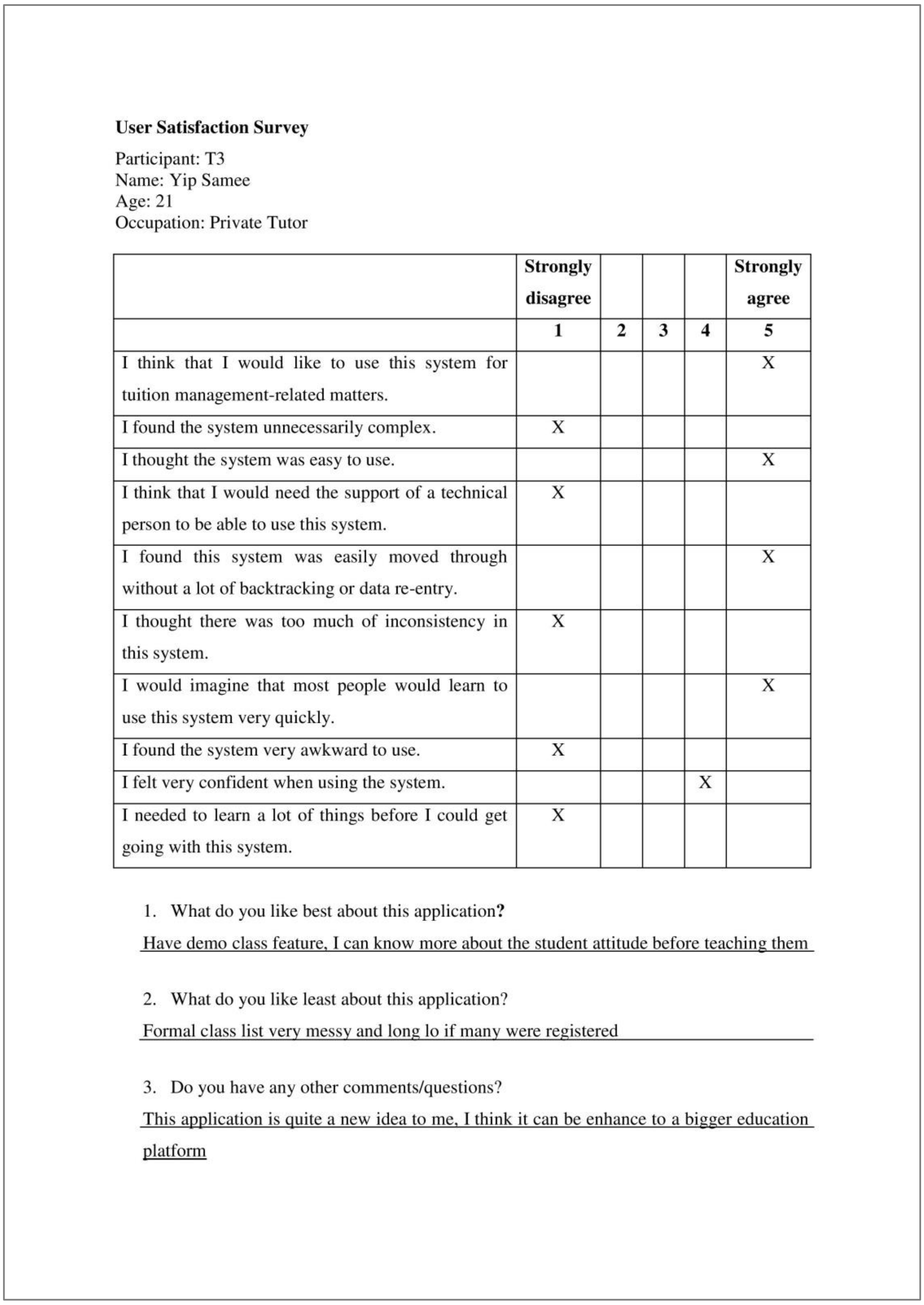
Very easy and convenient to use

2. What do you like least about this application?

Need to apply tutor account by approaching admin physically/email

3. Do you have any other comments/questions?

Should let us apply tutor account directly with the application, because admin might enter wrong information and it is troublesome to approach admin for an account



User Satisfaction Survey

Participant: T3
Name: Yip Samee
Age: 21
Occupation: Private Tutor

	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management-related matters.					X
I found the system unnecessarily complex.	X				
I thought the system was easy to use.					X
I think that I would need the support of a technical person to be able to use this system.	X				
I found this system was easily moved through without a lot of backtracking or data re-entry.					X
I thought there was too much of inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly.					X
I found the system very awkward to use.	X				
I felt very confident when using the system.				X	
I needed to learn a lot of things before I could get going with this system.	X				

1. What do you like best about this application**?**

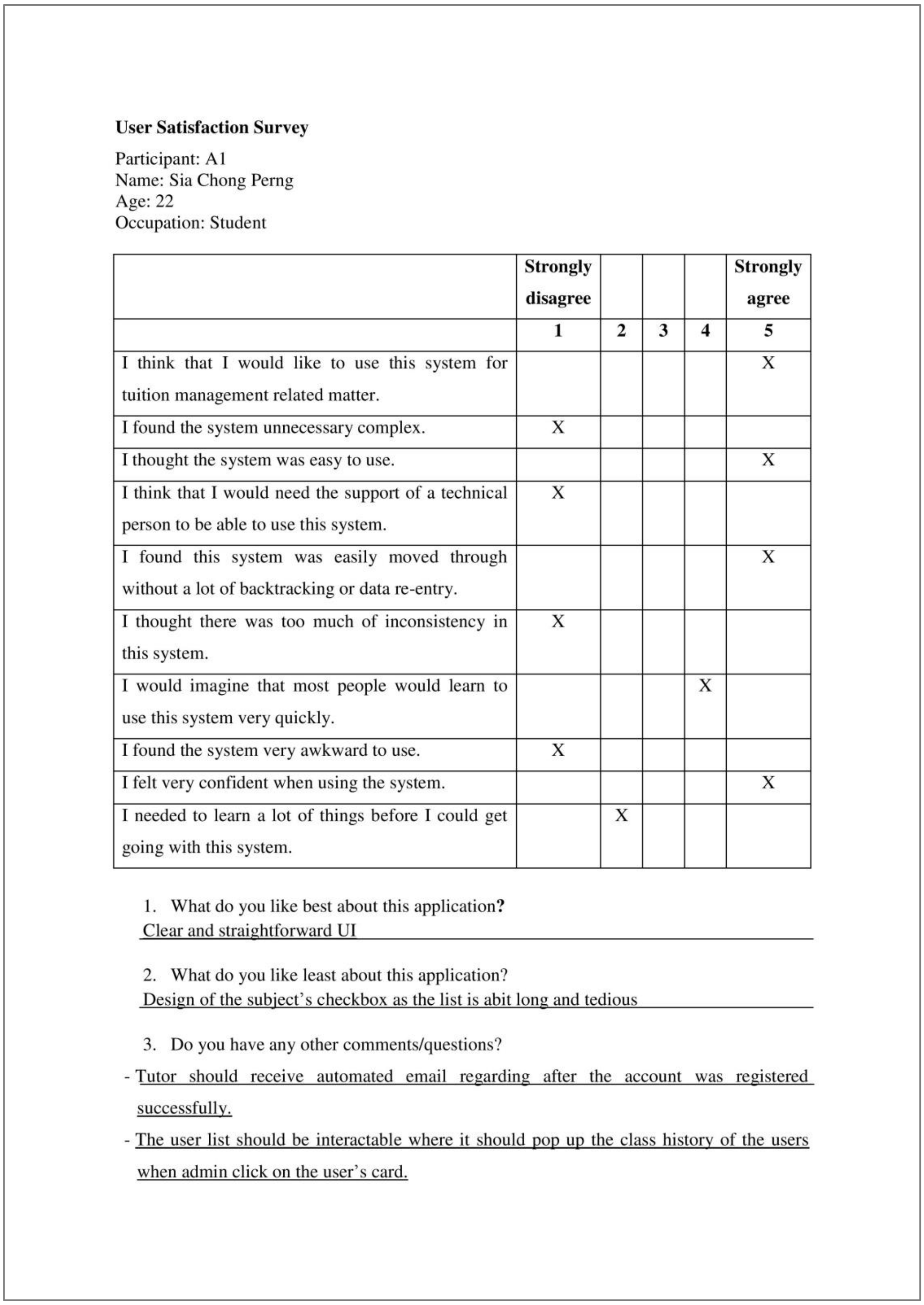
Have demo class feature, I can know more about the student attitude before teaching them

2. What do you like least about this application?

Formal class list very messy and long lo if many were registered

3. Do you have any other comments/questions?

This application is quite a new idea to me, I think it can be enhance to a bigger education platform



User Satisfaction Survey

Participant: A1
Name: Sia Chong Perng
Age: 22
Occupation: Student

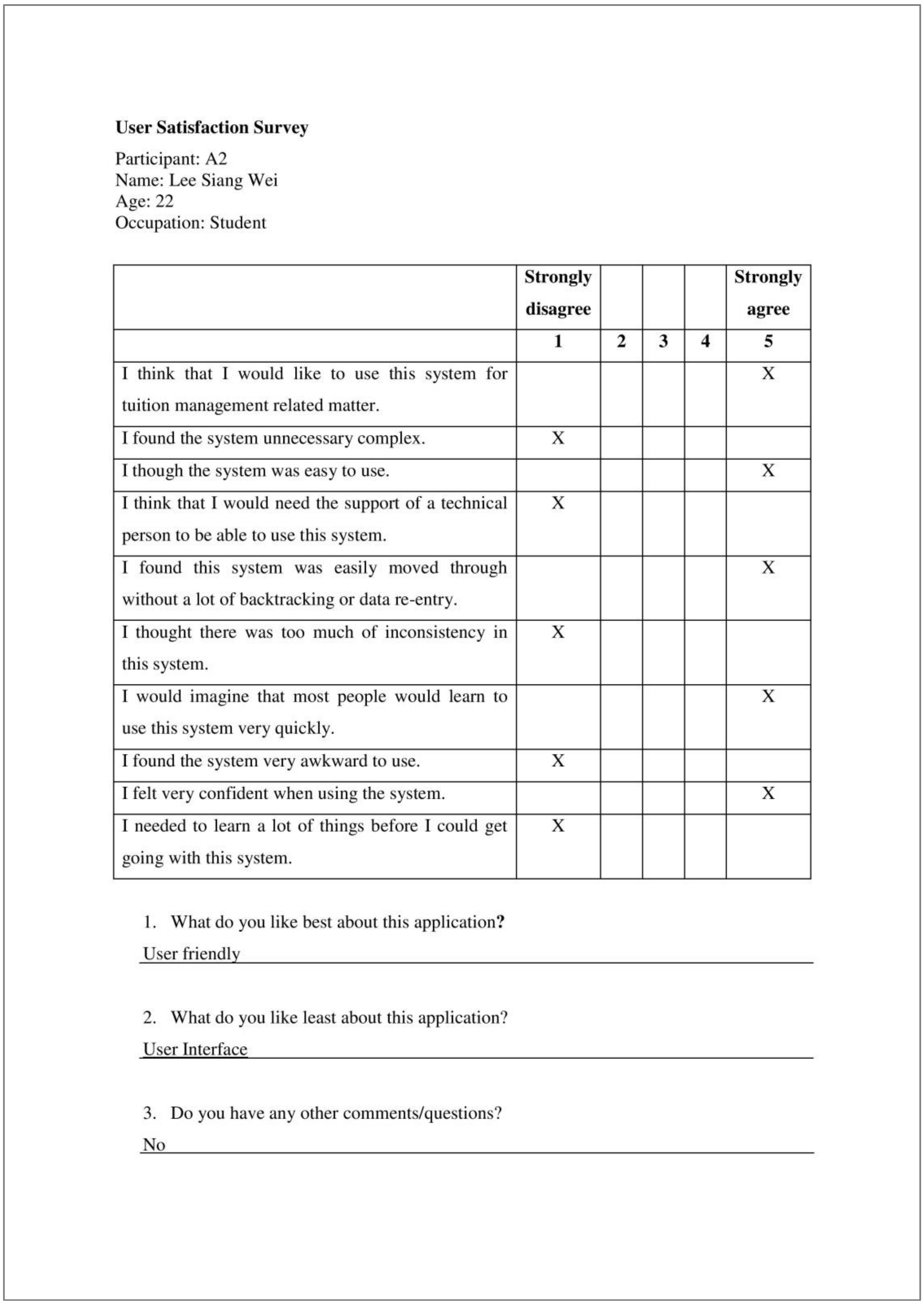
	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management related matter.					X
I found the system unnecessary complex.	X				
I thought the system was easy to use.					X
I think that I would need the support of a technical person to be able to use this system.	X				
I found this system was easily moved through without a lot of backtracking or data re-entry.					X
I thought there was too much of inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly.				X	
I found the system very awkward to use.	X				
I felt very confident when using the system.					X
I needed to learn a lot of things before I could get going with this system.		X			

1. What do you like best about this application**?**
Clear and straightforward UI

2. What do you like least about this application?
Design of the subject's checkbox as the list is abit long and tedious

3. Do you have any other comments/questions?

- Tutor should receive automated email regarding after the account was registered successfully.
- The user list should be interactable where it should pop up the class history of the users when admin click on the user's card.



User Satisfaction Survey

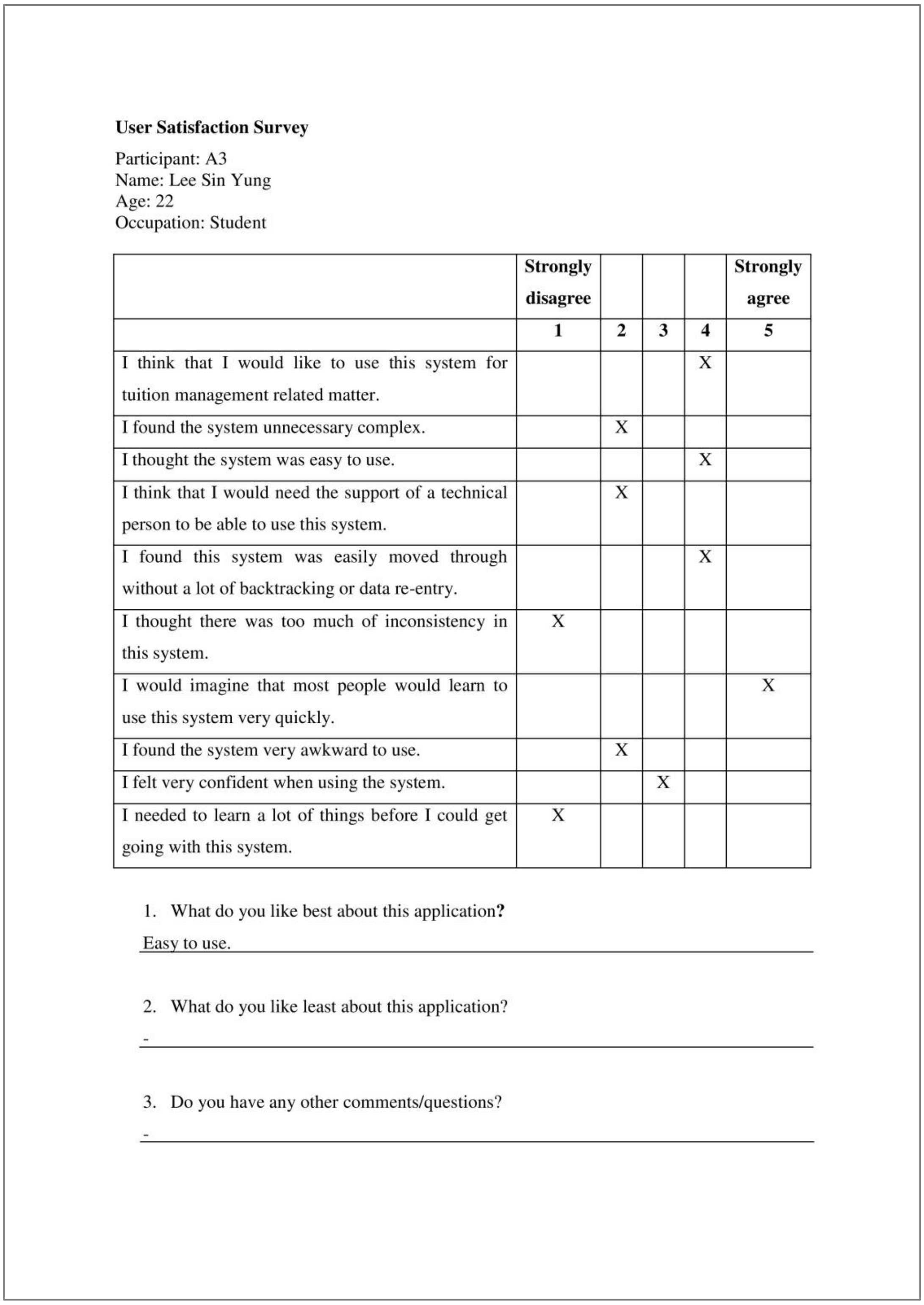
Participant: A2
Name: Lee Siang Wei
Age: 22
Occupation: Student

	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management related matter.					X
I found the system unnecessary complex.	X				
I though the system was easy to use.					X
I think that I would need the support of a technical person to be able to use this system.	X				
I found this system was easily moved through without a lot of backtracking or data re-entry.					X
I thought there was too much of inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly.					X
I found the system very awkward to use.	X				
I felt very confident when using the system.					X
I needed to learn a lot of things before I could get going with this system.	X				

1. What do you like best about this application**?**
User friendly

2. What do you like least about this application?
User Interface

3. Do you have any other comments/questions?
No



User Satisfaction Survey

Participant: A3
Name: Lee Sin Yung
Age: 22
Occupation: Student

	Strongly disagree				**Strongly agree**
	1	**2**	**3**	**4**	**5**
I think that I would like to use this system for tuition management related matter.				X	
I found the system unnecessary complex.		X			
I thought the system was easy to use.				X	
I think that I would need the support of a technical person to be able to use this system.		X			
I found this system was easily moved through without a lot of backtracking or data re-entry.				X	
I thought there was too much of inconsistency in this system.	X				
I would imagine that most people would learn to use this system very quickly.					X
I found the system very awkward to use.		X			
I felt very confident when using the system.			X		
I needed to learn a lot of things before I could get going with this system.	X				

1. What do you like best about this application**?**

Easy to use.

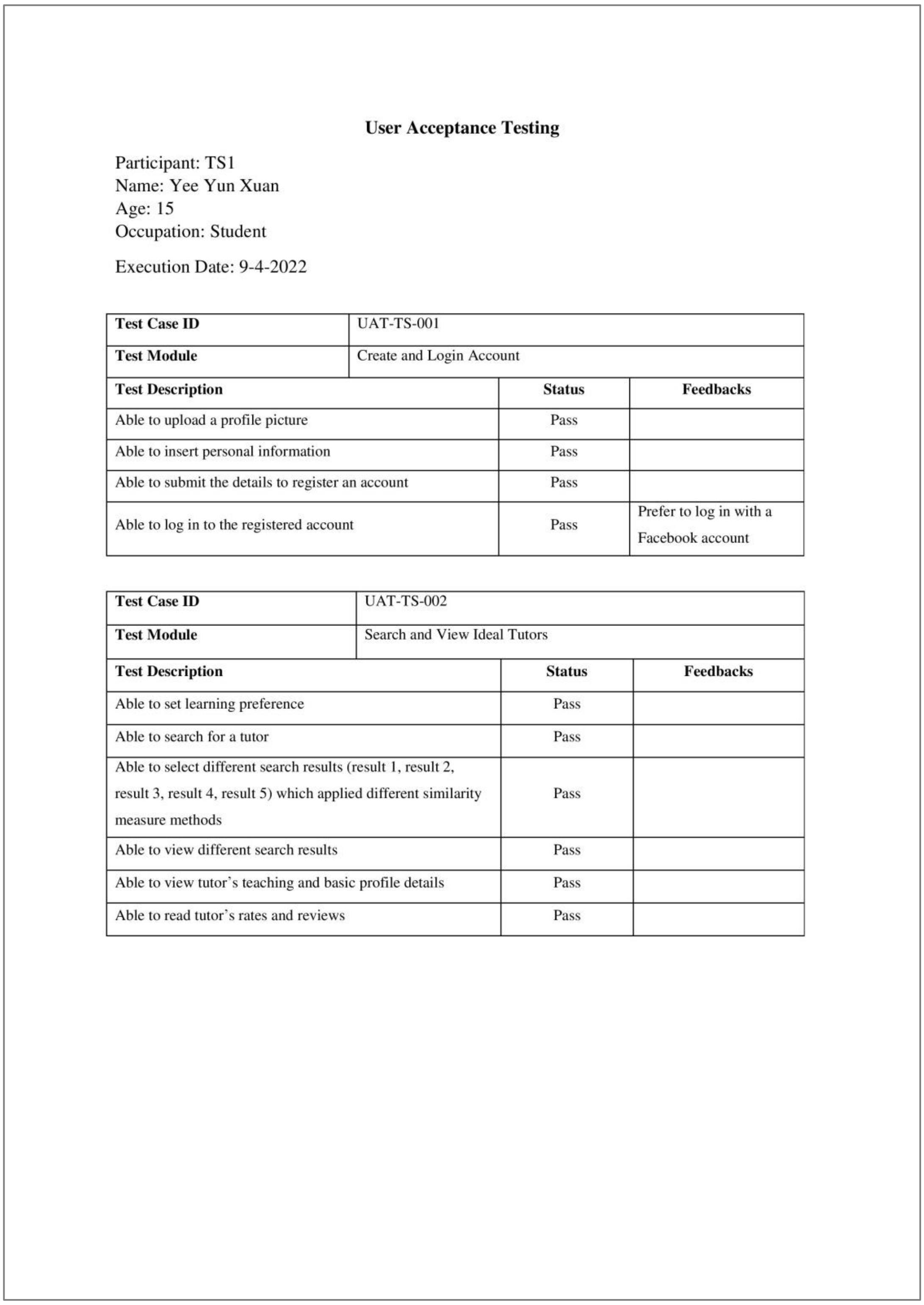
2. What do you like least about this application?

-

3. Do you have any other comments/questions?

-

APPENDIX D: User Acceptance Testing Results



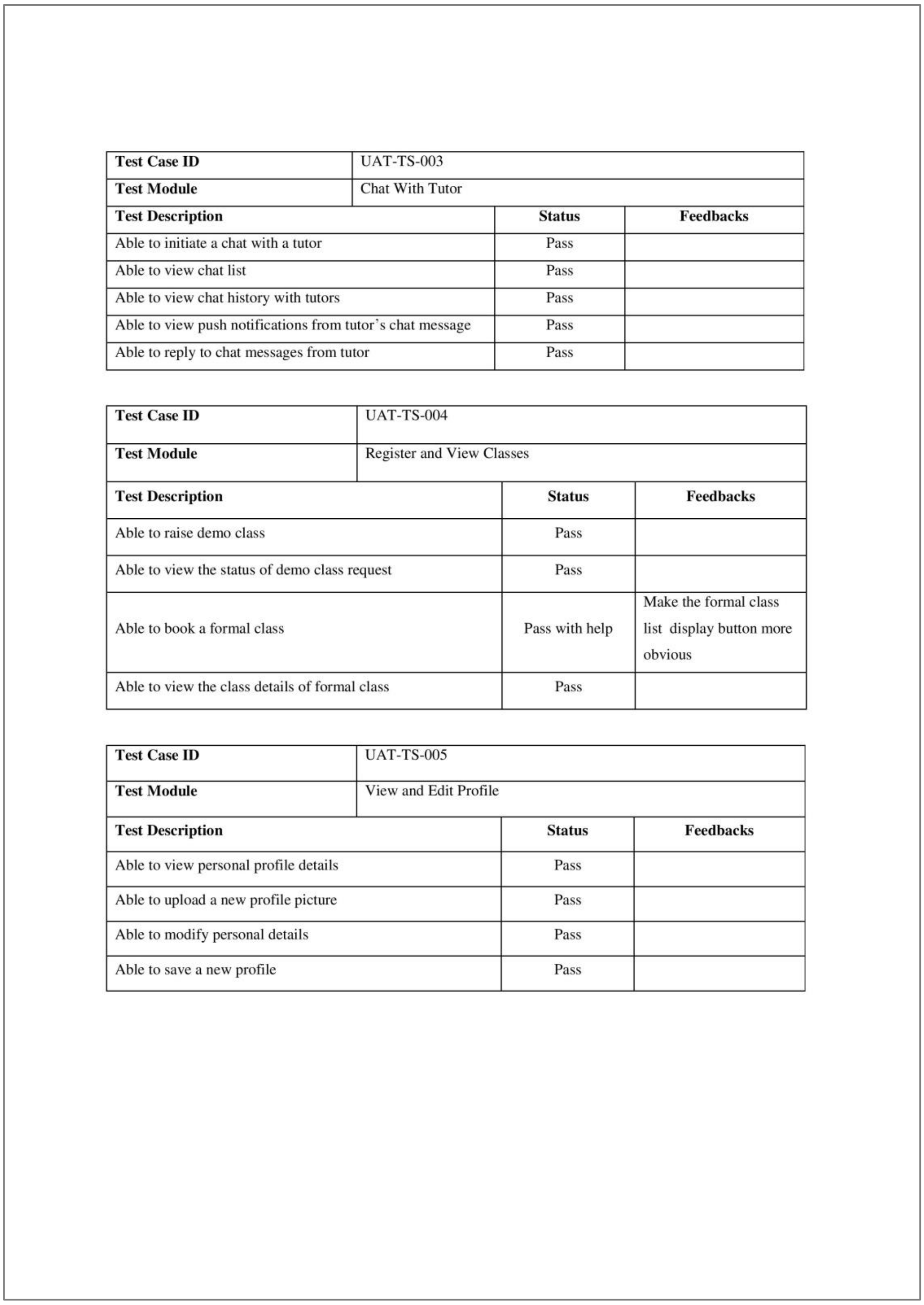
User Acceptance Testing

Participant: TS1
Name: Yee Yun Xuan
Age: 15
Occupation: Student

Execution Date: 9-4-2022

Test Case ID	UAT-TS-001	
Test Module	Create and Login Account	
Test Description	**Status**	**Feedbacks**
Able to upload a profile picture	Pass	
Able to insert personal information	Pass	
Able to submit the details to register an account	Pass	
Able to log in to the registered account	Pass	Prefer to log in with a Facebook account

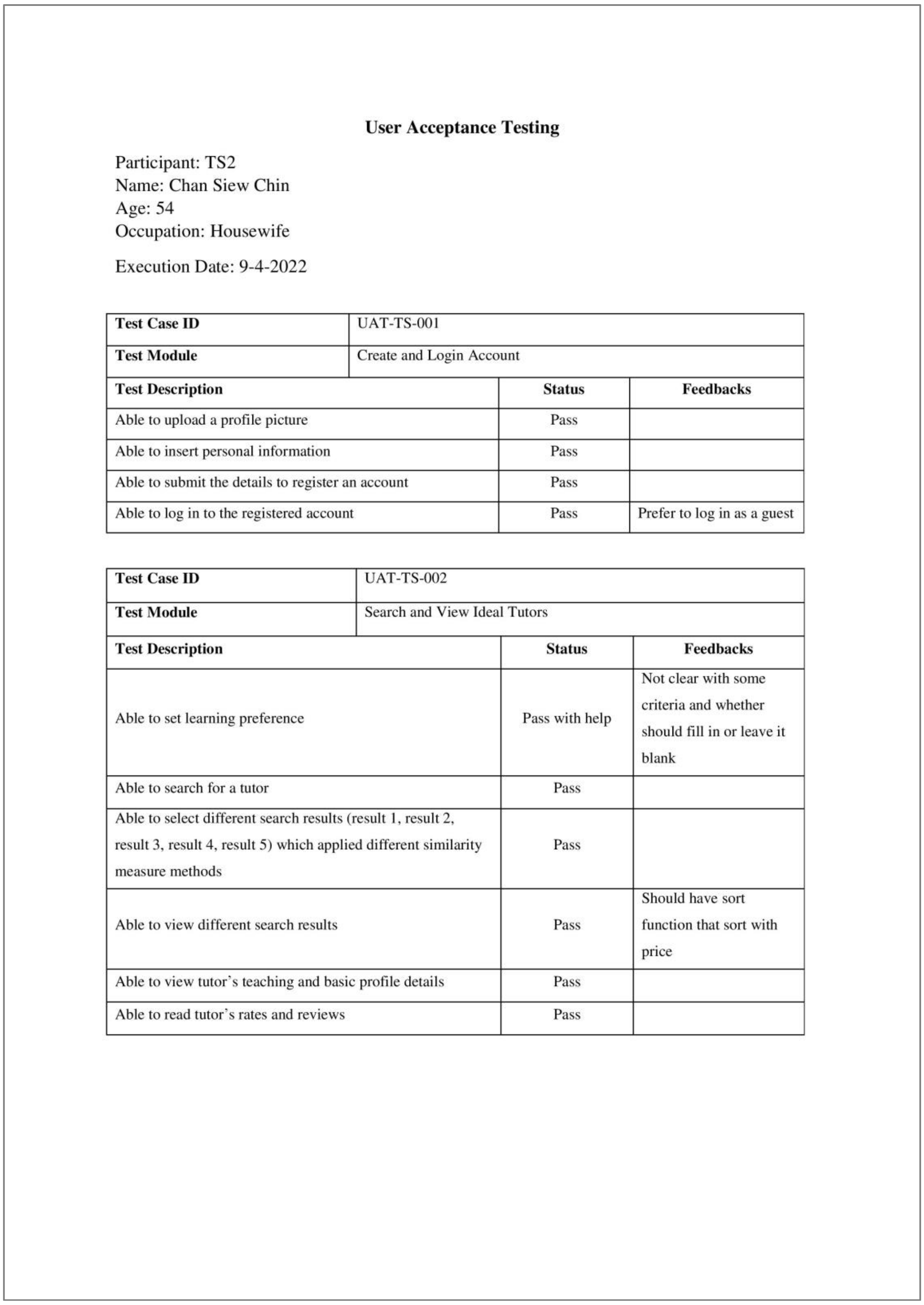
Test Case ID	UAT-TS-002	
Test Module	Search and View Ideal Tutors	
Test Description	**Status**	**Feedbacks**
Able to set learning preference	Pass	
Able to search for a tutor	Pass	
Able to select different search results (result 1, result 2, result 3, result 4, result 5) which applied different similarity measure methods	Pass	
Able to view different search results	Pass	
Able to view tutor's teaching and basic profile details	Pass	
Able to read tutor's rates and reviews	Pass	



Test Case ID	UAT-TS-003	
Test Module	Chat With Tutor	
Test Description	**Status**	**Feedbacks**
Able to initiate a chat with a tutor	Pass	
Able to view chat list	Pass	
Able to view chat history with tutors	Pass	
Able to view push notifications from tutor's chat message	Pass	
Able to reply to chat messages from tutor	Pass	

Test Case ID	UAT-TS-004	
Test Module	Register and View Classes	
Test Description	**Status**	**Feedbacks**
Able to raise demo class	Pass	
Able to view the status of demo class request	Pass	
Able to book a formal class	Pass with help	Make the formal class list display button more obvious
Able to view the class details of formal class	Pass	

Test Case ID	UAT-TS-005	
Test Module	View and Edit Profile	
Test Description	**Status**	**Feedbacks**
Able to view personal profile details	Pass	
Able to upload a new profile picture	Pass	
Able to modify personal details	Pass	
Able to save a new profile	Pass	



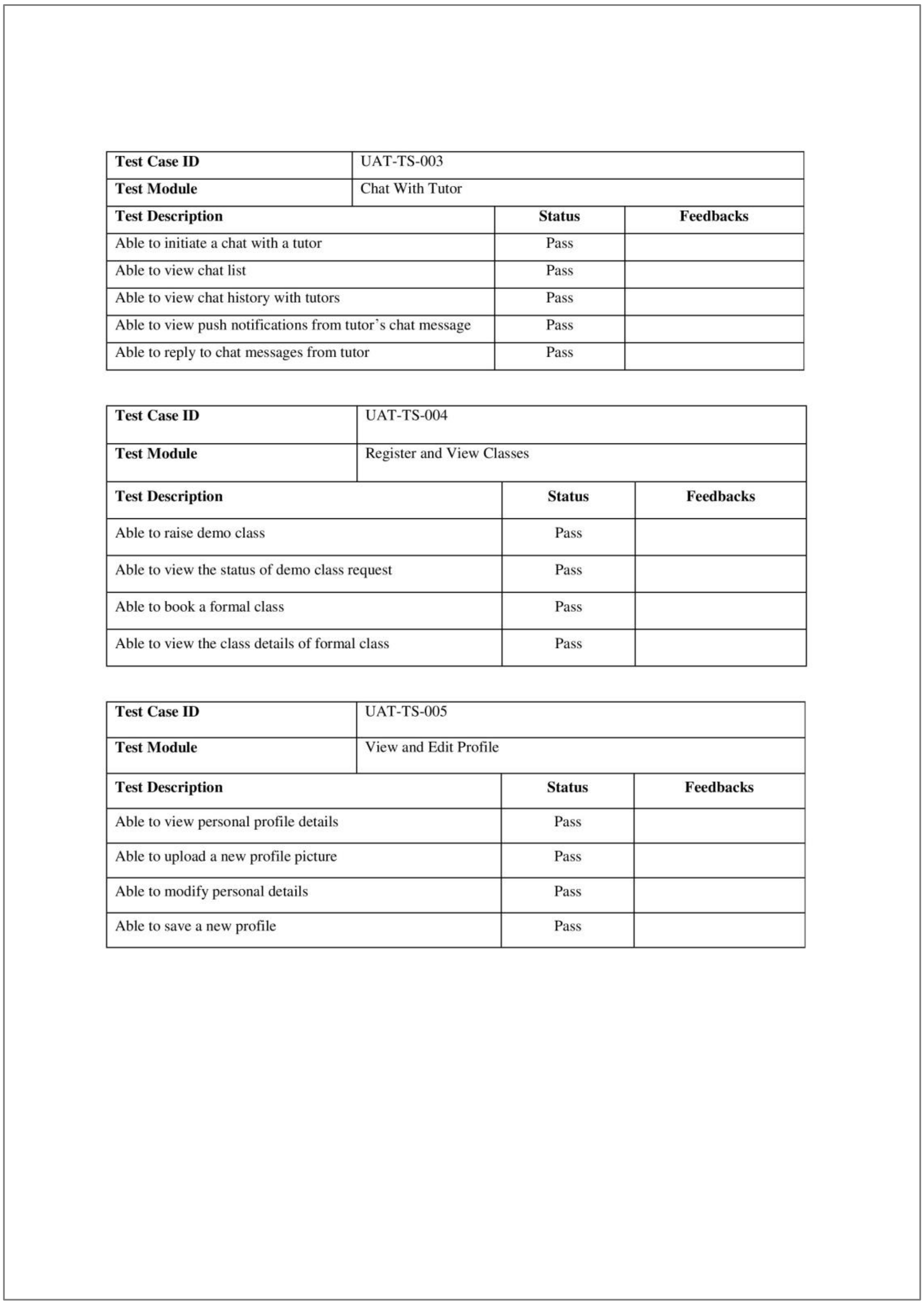
User Acceptance Testing

Participant: TS2
Name: Chan Siew Chin
Age: 54
Occupation: Housewife

Execution Date: 9-4-2022

Test Case ID	UAT-TS-001	
Test Module	Create and Login Account	
Test Description	**Status**	**Feedbacks**
Able to upload a profile picture	Pass	
Able to insert personal information	Pass	
Able to submit the details to register an account	Pass	
Able to log in to the registered account	Pass	Prefer to log in as a guest

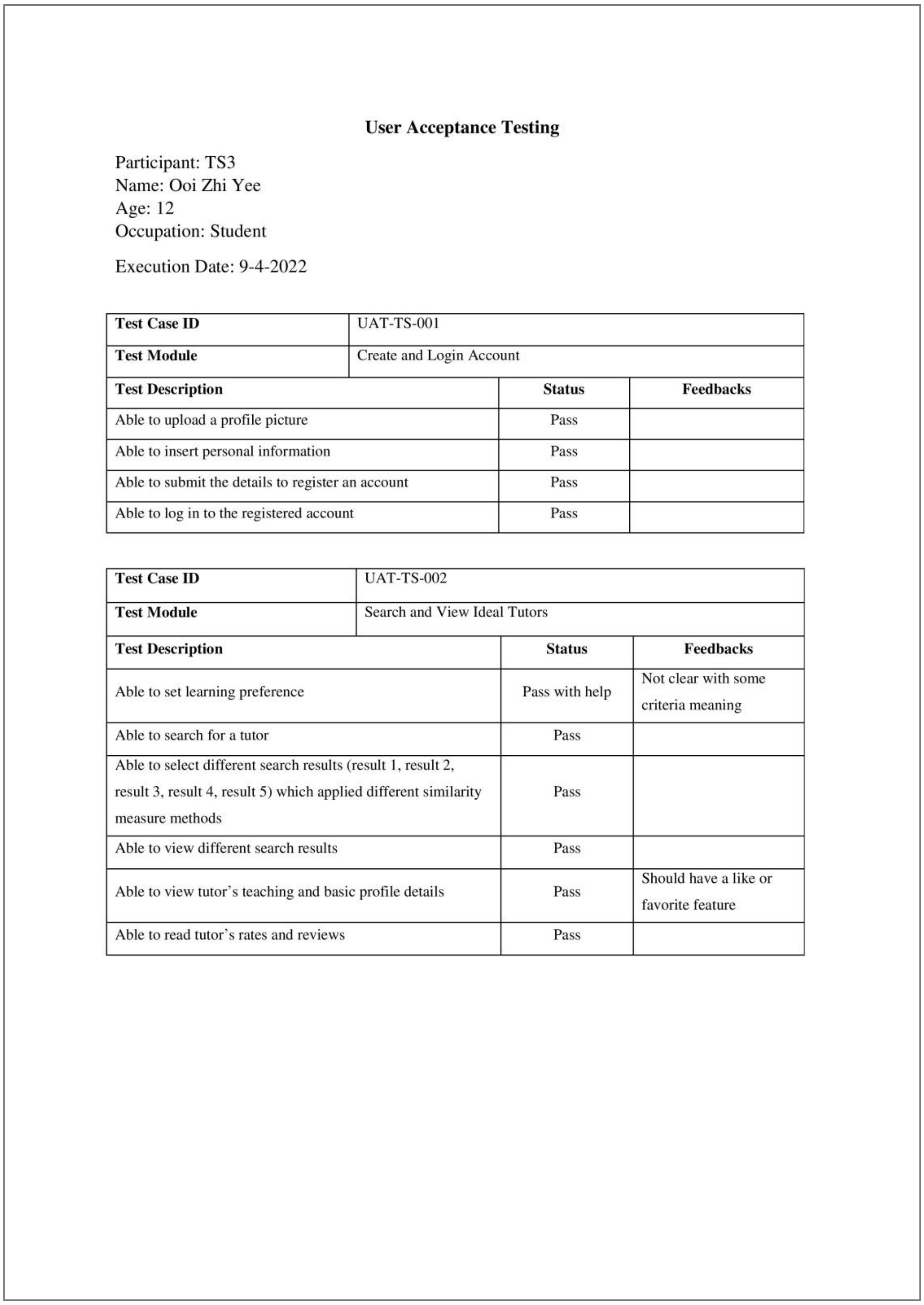
Test Case ID	UAT-TS-002	
Test Module	Search and View Ideal Tutors	
Test Description	**Status**	**Feedbacks**
Able to set learning preference	Pass with help	Not clear with some criteria and whether should fill in or leave it blank
Able to search for a tutor	Pass	
Able to select different search results (result 1, result 2, result 3, result 4, result 5) which applied different similarity measure methods	Pass	
Able to view different search results	Pass	Should have sort function that sort with price
Able to view tutor's teaching and basic profile details	Pass	
Able to read tutor's rates and reviews	Pass	



Test Case ID	UAT-TS-003	
Test Module	Chat With Tutor	
Test Description	**Status**	**Feedbacks**
Able to initiate a chat with a tutor	Pass	
Able to view chat list	Pass	
Able to view chat history with tutors	Pass	
Able to view push notifications from tutor's chat message	Pass	
Able to reply to chat messages from tutor	Pass	

Test Case ID	UAT-TS-004	
Test Module	Register and View Classes	
Test Description	**Status**	**Feedbacks**
Able to raise demo class	Pass	
Able to view the status of demo class request	Pass	
Able to book a formal class	Pass	
Able to view the class details of formal class	Pass	

Test Case ID	UAT-TS-005	
Test Module	View and Edit Profile	
Test Description	**Status**	**Feedbacks**
Able to view personal profile details	Pass	
Able to upload a new profile picture	Pass	
Able to modify personal details	Pass	
Able to save a new profile	Pass	



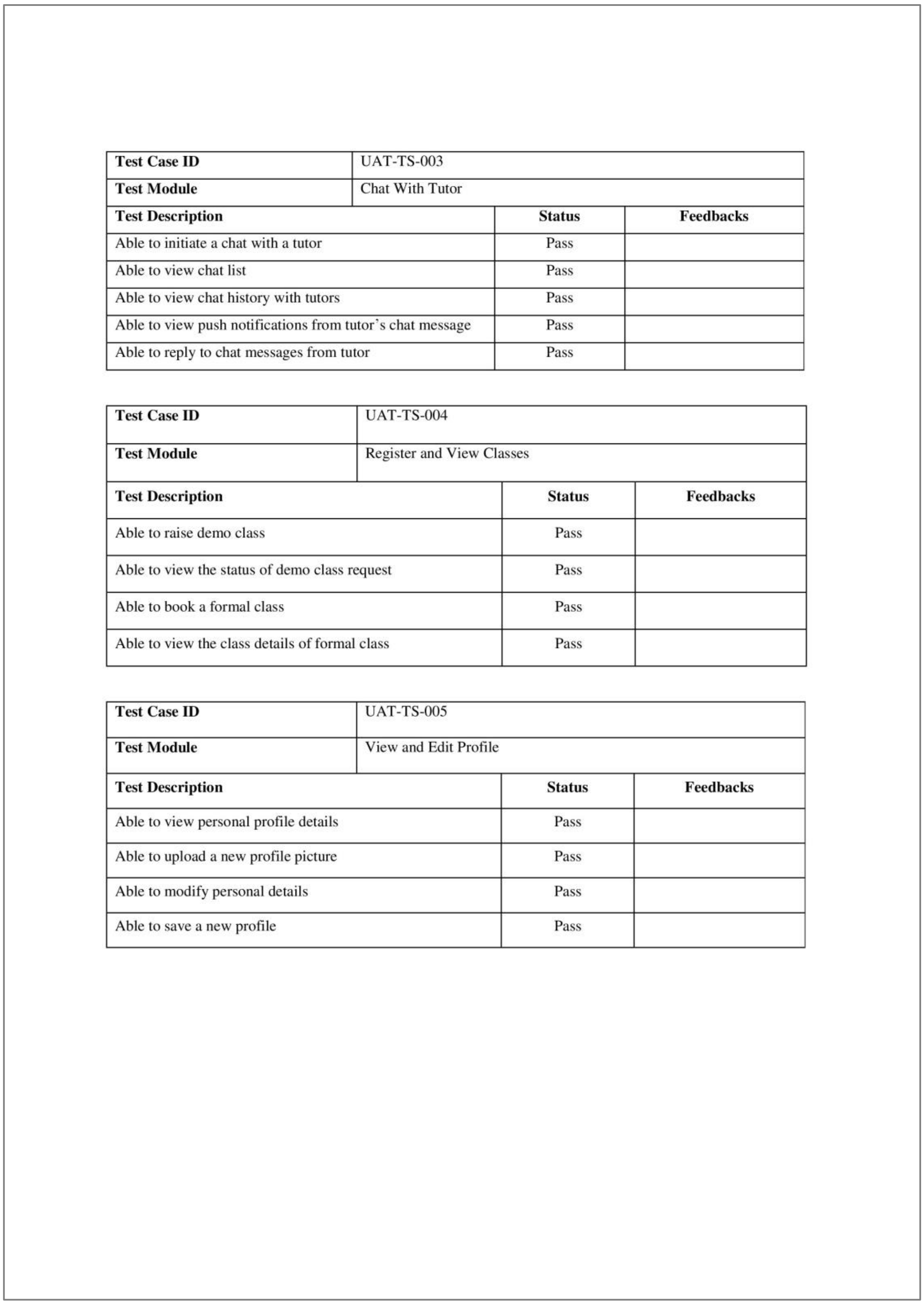
User Acceptance Testing

Participant: TS3
Name: Ooi Zhi Yee
Age: 12
Occupation: Student

Execution Date: 9-4-2022

Test Case ID	UAT-TS-001	
Test Module	Create and Login Account	
Test Description	**Status**	**Feedbacks**
Able to upload a profile picture	Pass	
Able to insert personal information	Pass	
Able to submit the details to register an account	Pass	
Able to log in to the registered account	Pass	

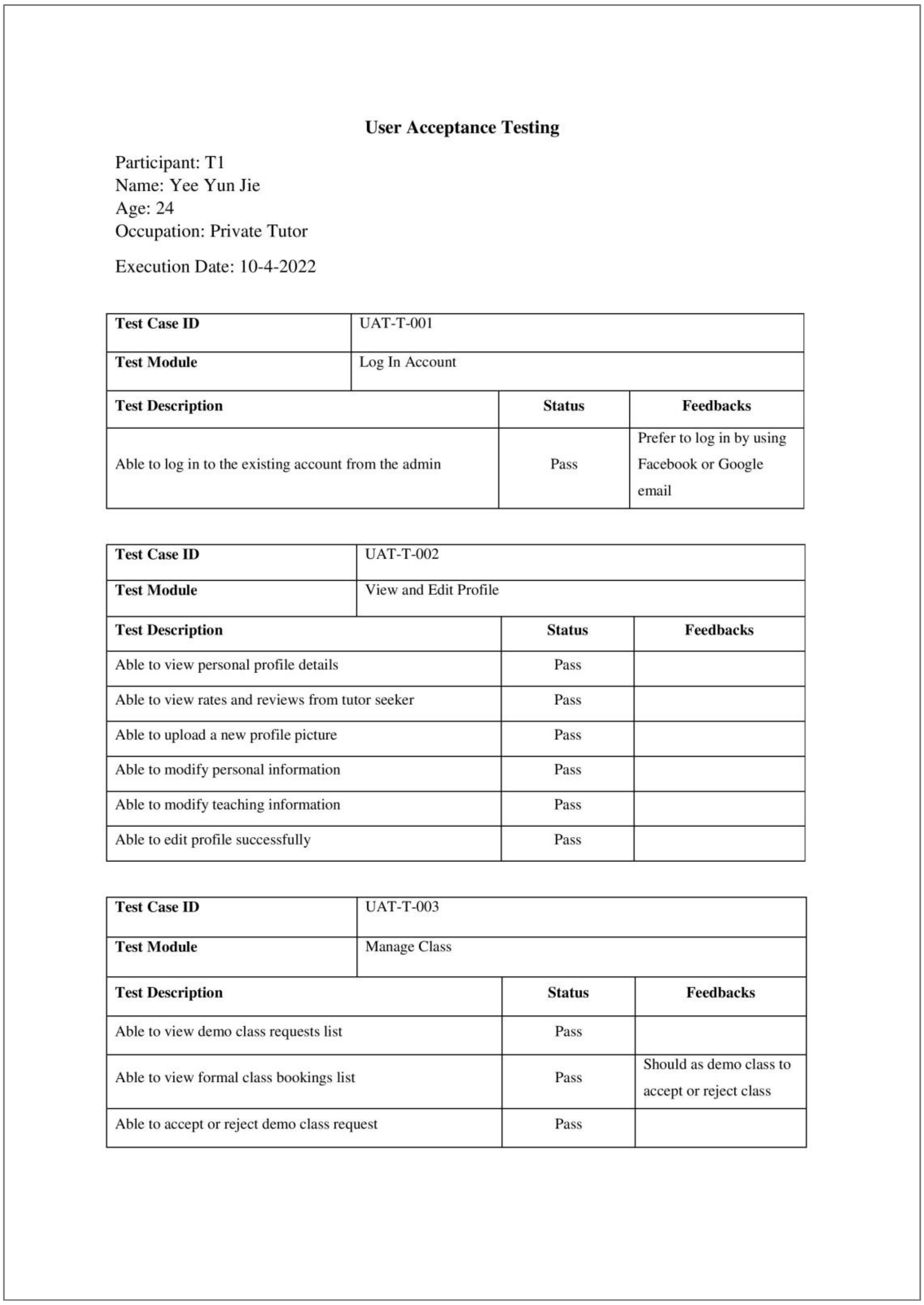
Test Case ID	UAT-TS-002	
Test Module	Search and View Ideal Tutors	
Test Description	**Status**	**Feedbacks**
Able to set learning preference	Pass with help	Not clear with some criteria meaning
Able to search for a tutor	Pass	
Able to select different search results (result 1, result 2, result 3, result 4, result 5) which applied different similarity measure methods	Pass	
Able to view different search results	Pass	
Able to view tutor's teaching and basic profile details	Pass	Should have a like or favorite feature
Able to read tutor's rates and reviews	Pass	



Test Case ID	UAT-TS-003	
Test Module	Chat With Tutor	
Test Description	**Status**	**Feedbacks**
Able to initiate a chat with a tutor	Pass	
Able to view chat list	Pass	
Able to view chat history with tutors	Pass	
Able to view push notifications from tutor's chat message	Pass	
Able to reply to chat messages from tutor	Pass	

Test Case ID	UAT-TS-004	
Test Module	Register and View Classes	
Test Description	**Status**	**Feedbacks**
Able to raise demo class	Pass	
Able to view the status of demo class request	Pass	
Able to book a formal class	Pass	
Able to view the class details of formal class	Pass	

Test Case ID	UAT-TS-005	
Test Module	View and Edit Profile	
Test Description	**Status**	**Feedbacks**
Able to view personal profile details	Pass	
Able to upload a new profile picture	Pass	
Able to modify personal details	Pass	
Able to save a new profile	Pass	



User Acceptance Testing

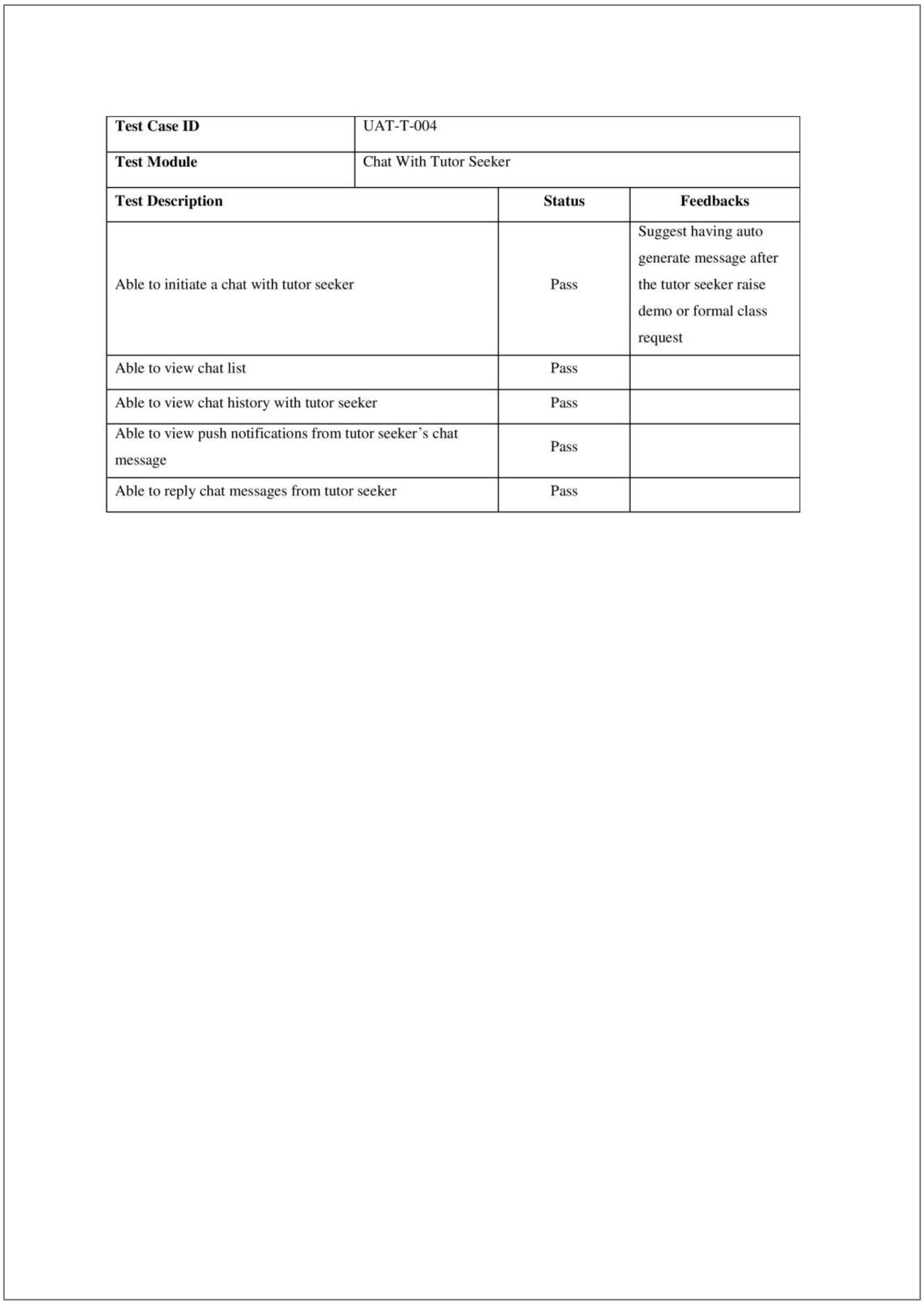
Participant: T1
Name: Yee Yun Jie
Age: 24
Occupation: Private Tutor

Execution Date: 10-4-2022

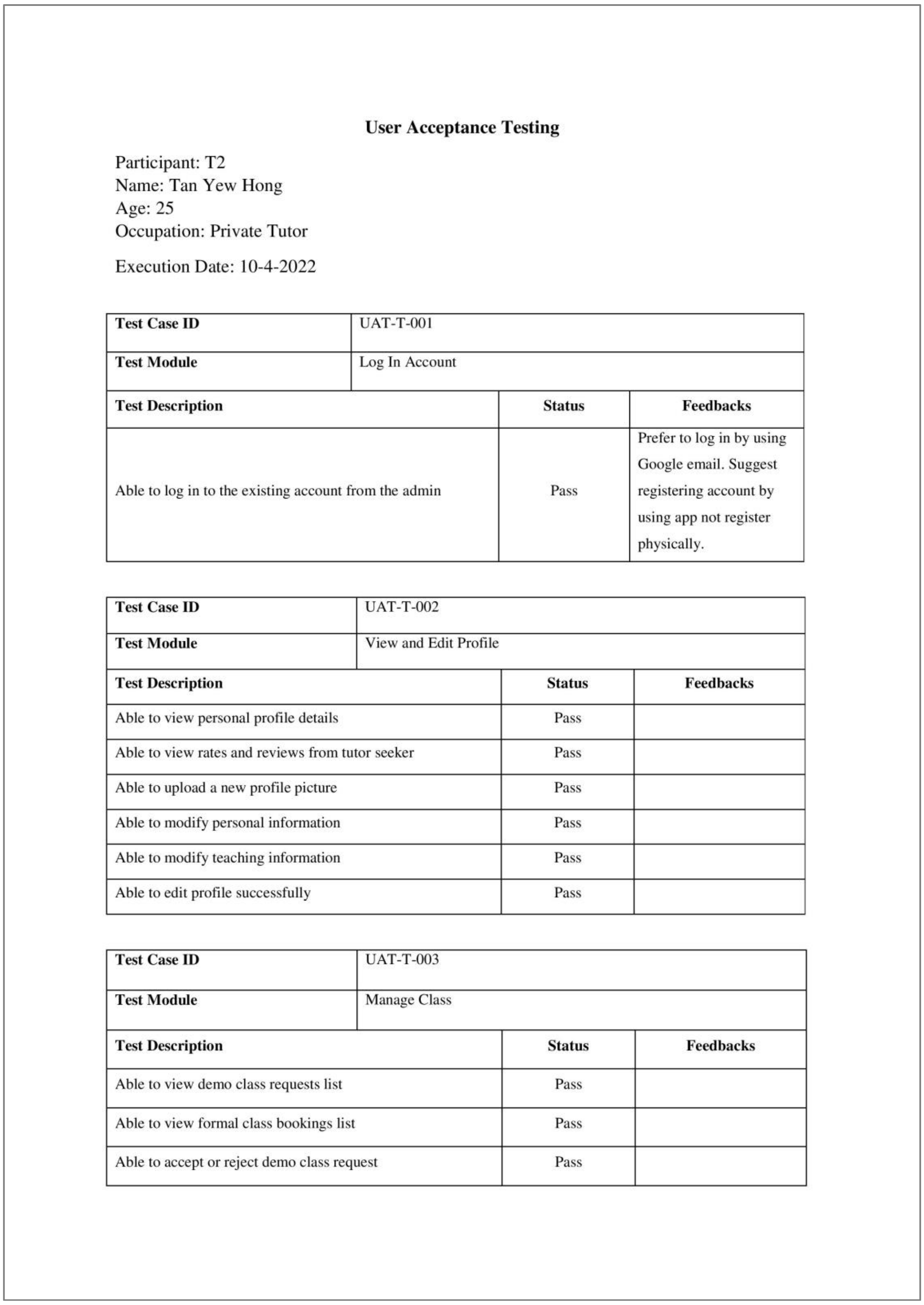
Test Case ID	UAT-T-001	
Test Module	Log In Account	
Test Description	**Status**	**Feedbacks**
Able to log in to the existing account from the admin	Pass	Prefer to log in by using Facebook or Google email

Test Case ID	UAT-T-002	
Test Module	View and Edit Profile	
Test Description	**Status**	**Feedbacks**
Able to view personal profile details	Pass	
Able to view rates and reviews from tutor seeker	Pass	
Able to upload a new profile picture	Pass	
Able to modify personal information	Pass	
Able to modify teaching information	Pass	
Able to edit profile successfully	Pass	

Test Case ID	UAT-T-003	
Test Module	Manage Class	
Test Description	**Status**	**Feedbacks**
Able to view demo class requests list	Pass	
Able to view formal class bookings list	Pass	Should as demo class to accept or reject class
Able to accept or reject demo class request	Pass	



Test Case ID	UAT-T-004	
Test Module	Chat With Tutor Seeker	
Test Description	**Status**	**Feedbacks**
Able to initiate a chat with tutor seeker	Pass	Suggest having auto generate message after the tutor seeker raise demo or formal class request
Able to view chat list	Pass	
Able to view chat history with tutor seeker	Pass	
Able to view push notifications from tutor seeker's chat message	Pass	
Able to reply chat messages from tutor seeker	Pass	



User Acceptance Testing

Participant: T2
Name: Tan Yew Hong
Age: 25
Occupation: Private Tutor

Execution Date: 10-4-2022

Test Case ID	UAT-T-001
Test Module	Log In Account

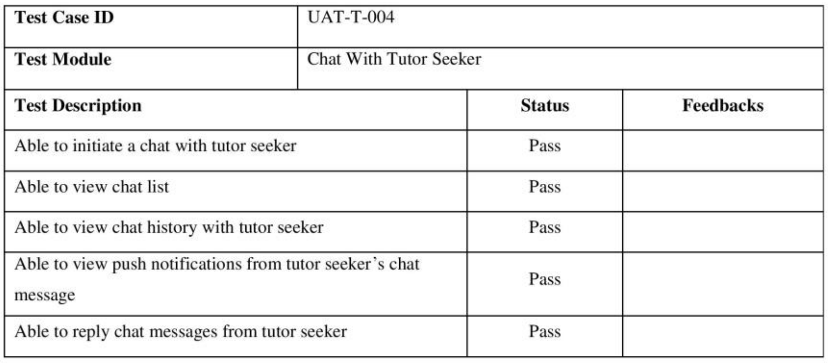
Test Description	**Status**	**Feedbacks**
Able to log in to the existing account from the admin	Pass	Prefer to log in by using Google email. Suggest registering account by using app not register physically.

Test Case ID	UAT-T-002
Test Module	View and Edit Profile

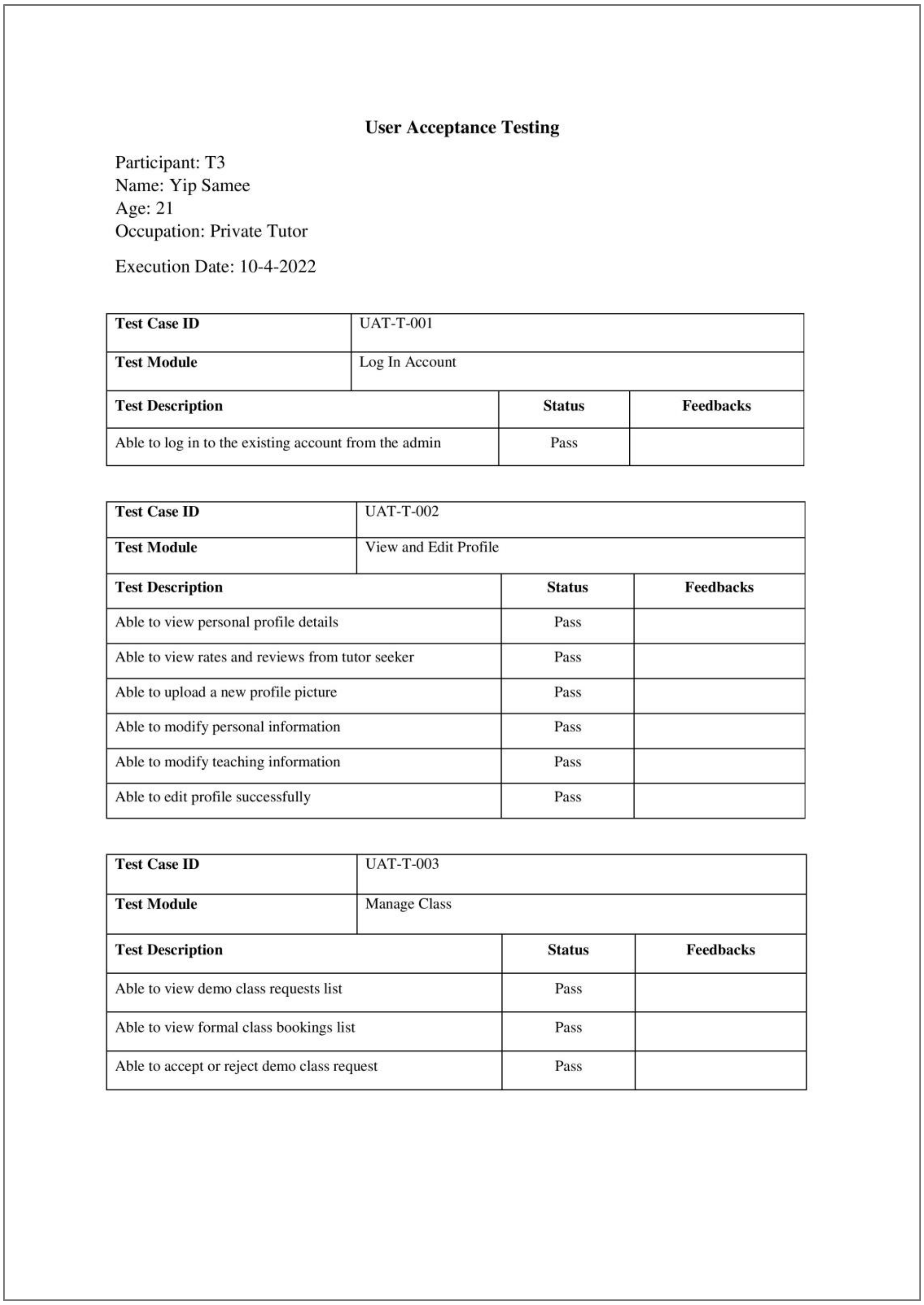
Test Description	**Status**	**Feedbacks**
Able to view personal profile details	Pass	
Able to view rates and reviews from tutor seeker	Pass	
Able to upload a new profile picture	Pass	
Able to modify personal information	Pass	
Able to modify teaching information	Pass	
Able to edit profile successfully	Pass	

Test Case ID	UAT-T-003
Test Module	Manage Class

Test Description	**Status**	**Feedbacks**
Able to view demo class requests list	Pass	
Able to view formal class bookings list	Pass	
Able to accept or reject demo class request	Pass	



Test Case ID	UAT-T-004	
Test Module	Chat With Tutor Seeker	
Test Description	**Status**	**Feedbacks**
Able to initiate a chat with tutor seeker	Pass	
Able to view chat list	Pass	
Able to view chat history with tutor seeker	Pass	
Able to view push notifications from tutor seeker's chat message	Pass	
Able to reply chat messages from tutor seeker	Pass	



User Acceptance Testing

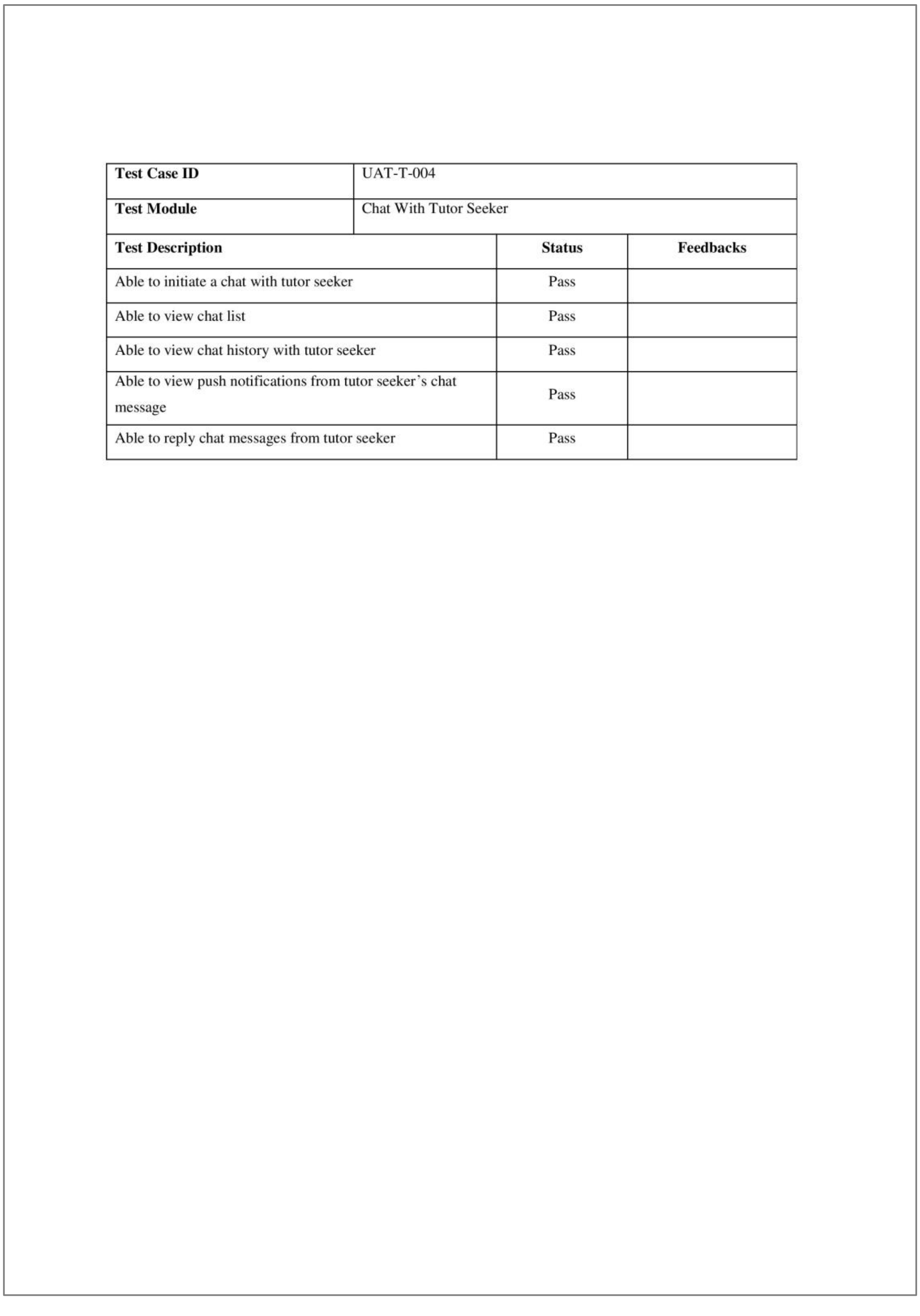
Participant: T3
Name: Yip Samee
Age: 21
Occupation: Private Tutor

Execution Date: 10-4-2022

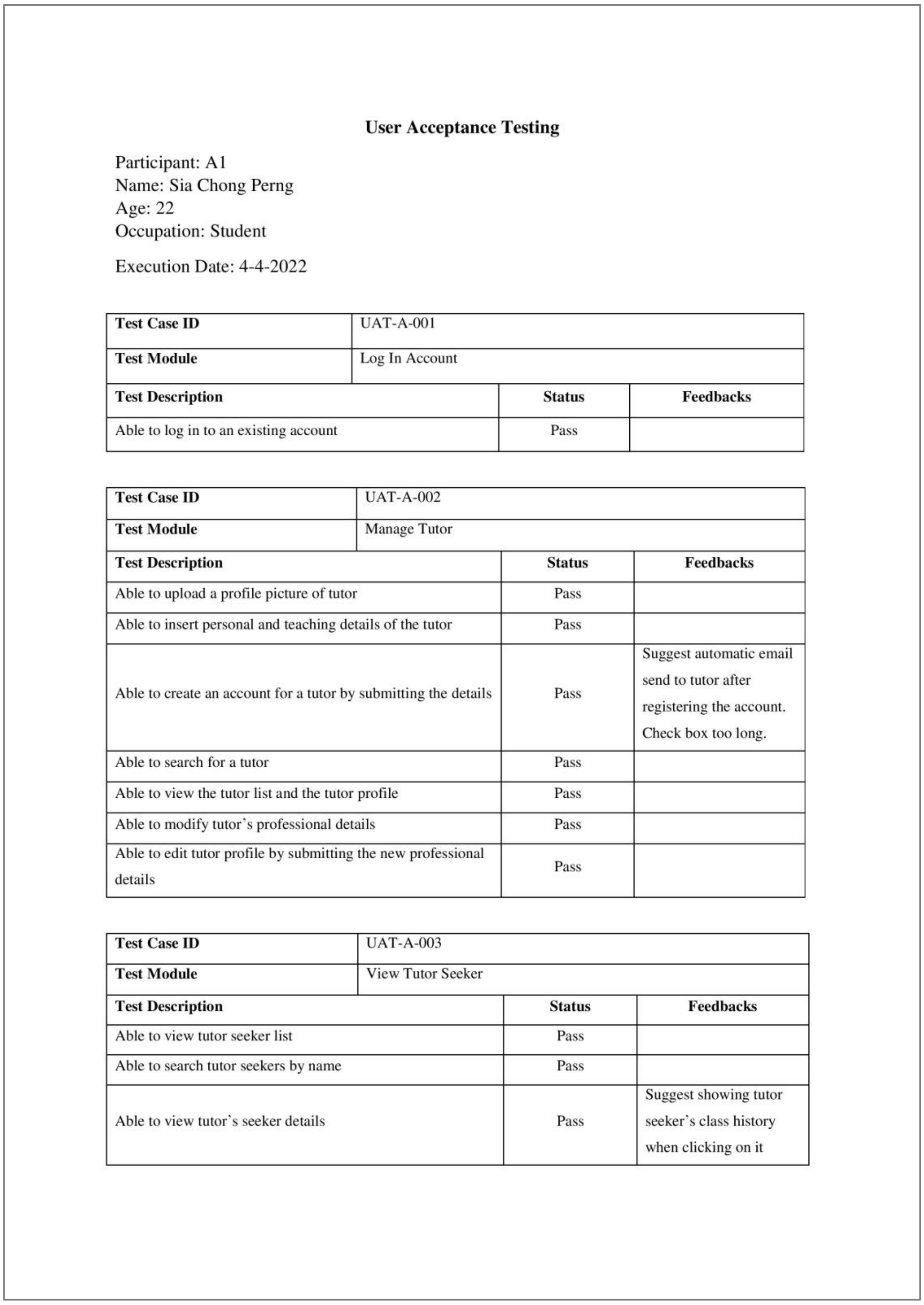
Test Case ID	UAT-T-001	
Test Module	Log In Account	
Test Description	**Status**	**Feedbacks**
Able to log in to the existing account from the admin	Pass	

Test Case ID	UAT-T-002	
Test Module	View and Edit Profile	
Test Description	**Status**	**Feedbacks**
Able to view personal profile details	Pass	
Able to view rates and reviews from tutor seeker	Pass	
Able to upload a new profile picture	Pass	
Able to modify personal information	Pass	
Able to modify teaching information	Pass	
Able to edit profile successfully	Pass	

Test Case ID	UAT-T-003	
Test Module	Manage Class	
Test Description	**Status**	**Feedbacks**
Able to view demo class requests list	Pass	
Able to view formal class bookings list	Pass	
Able to accept or reject demo class request	Pass	



Test Case ID	UAT-T-004	
Test Module	Chat With Tutor Seeker	
Test Description	**Status**	**Feedbacks**
Able to initiate a chat with tutor seeker	Pass	
Able to view chat list	Pass	
Able to view chat history with tutor seeker	Pass	
Able to view push notifications from tutor seeker’s chat message	Pass	
Able to reply chat messages from tutor seeker	Pass	



User Acceptance Testing

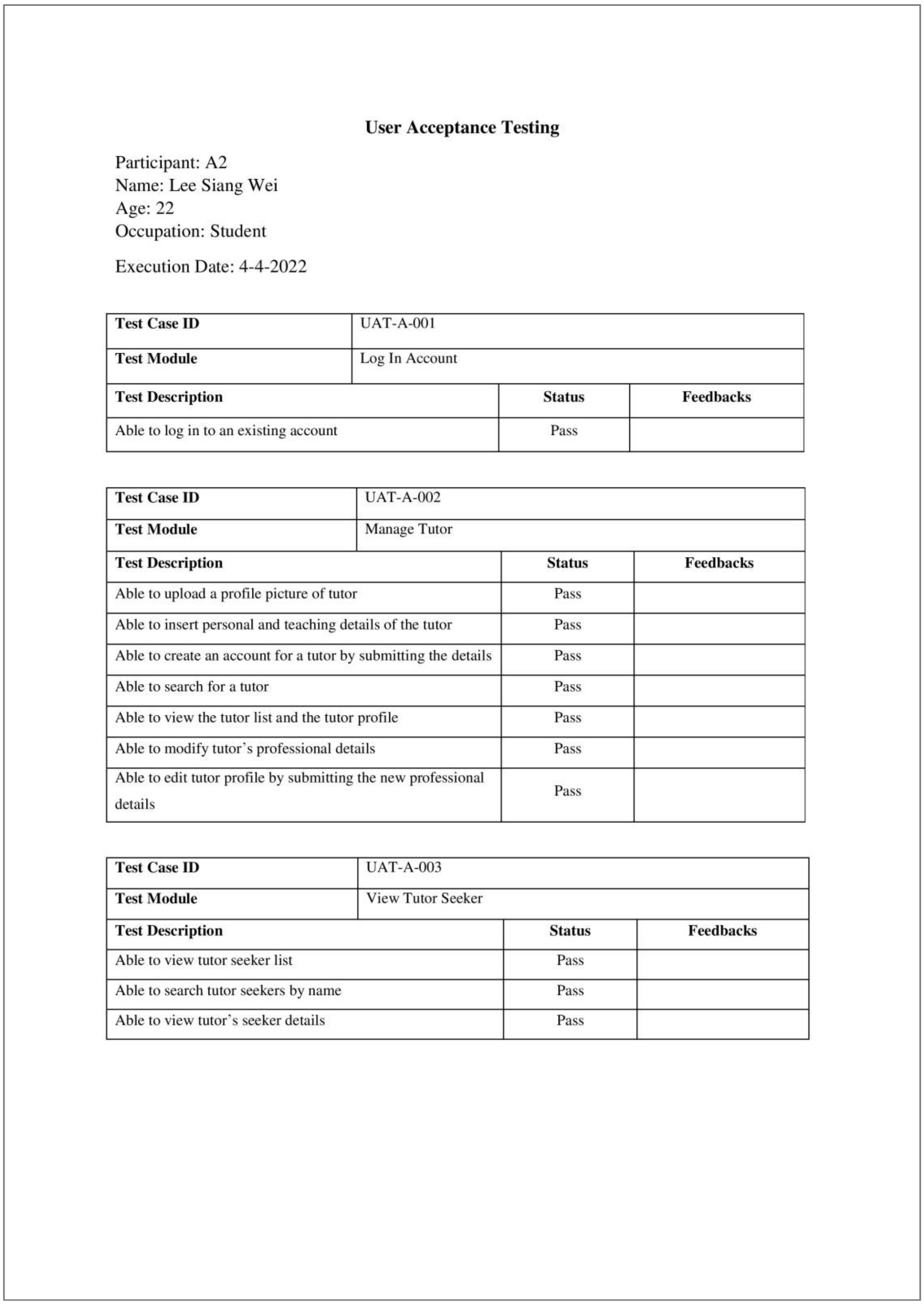
Participant: A1
Name: Sia Chong Perng
Age: 22
Occupation: Student

Execution Date: 4-4-2022

Test Case ID	UAT-A-001	
Test Module	Log In Account	
Test Description	**Status**	**Feedbacks**
Able to log in to an existing account	Pass	

Test Case ID	UAT-A-002	
Test Module	Manage Tutor	
Test Description	**Status**	**Feedbacks**
Able to upload a profile picture of tutor	Pass	
Able to insert personal and teaching details of the tutor	Pass	
Able to create an account for a tutor by submitting the details	Pass	Suggest automatic email send to tutor after registering the account. Check box too long.
Able to search for a tutor	Pass	
Able to view the tutor list and the tutor profile	Pass	
Able to modify tutor's professional details	Pass	
Able to edit tutor profile by submitting the new professional details	Pass	

Test Case ID	UAT-A-003	
Test Module	View Tutor Seeker	
Test Description	**Status**	**Feedbacks**
Able to view tutor seeker list	Pass	
Able to search tutor seekers by name	Pass	
Able to view tutor's seeker details	Pass	Suggest showing tutor seeker's class history when clicking on it



User Acceptance Testing

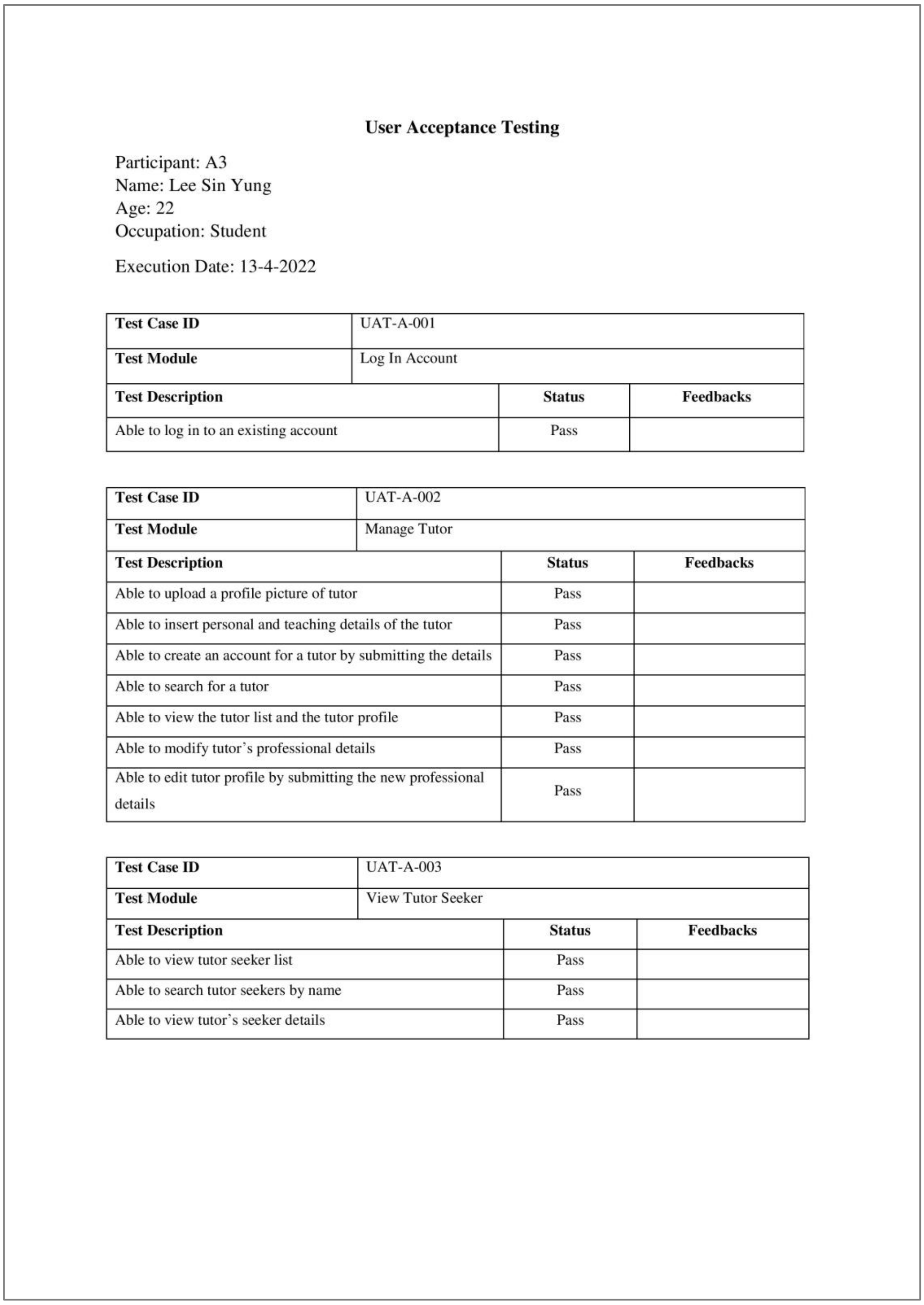
Participant: A2
Name: Lee Siang Wei
Age: 22
Occupation: Student

Execution Date: 4-4-2022

Test Case ID	UAT-A-001	
Test Module	Log In Account	
Test Description	**Status**	**Feedbacks**
Able to log in to an existing account	Pass	

Test Case ID	UAT-A-002	
Test Module	Manage Tutor	
Test Description	**Status**	**Feedbacks**
Able to upload a profile picture of tutor	Pass	
Able to insert personal and teaching details of the tutor	Pass	
Able to create an account for a tutor by submitting the details	Pass	
Able to search for a tutor	Pass	
Able to view the tutor list and the tutor profile	Pass	
Able to modify tutor's professional details	Pass	
Able to edit tutor profile by submitting the new professional details	Pass	

Test Case ID	UAT-A-003	
Test Module	View Tutor Seeker	
Test Description	**Status**	**Feedbacks**
Able to view tutor seeker list	Pass	
Able to search tutor seekers by name	Pass	
Able to view tutor's seeker details	Pass	



User Acceptance Testing

Participant: A3
Name: Lee Sin Yung
Age: 22
Occupation: Student

Execution Date: 13-4-2022

Test Case ID	UAT-A-001		
Test Module	Log In Account		
Test Description		**Status**	**Feedbacks**
Able to log in to an existing account		Pass	

Test Case ID	UAT-A-002		
Test Module	Manage Tutor		
Test Description		**Status**	**Feedbacks**
Able to upload a profile picture of tutor		Pass	
Able to insert personal and teaching details of the tutor		Pass	
Able to create an account for a tutor by submitting the details		Pass	
Able to search for a tutor		Pass	
Able to view the tutor list and the tutor profile		Pass	
Able to modify tutor's professional details		Pass	
Able to edit tutor profile by submitting the new professional details		Pass	

Test Case ID	UAT-A-003		
Test Module	View Tutor Seeker		
Test Description		**Status**	**Feedbacks**
Able to view tutor seeker list		Pass	
Able to search tutor seekers by name		Pass	
Able to view tutor's seeker details		Pass	