

**AN INTERACTIVE SWIFT PROGRAMMING
LANGUAGE E-LEARNING PLATFORM FOR IOS
APPLICATION DEVELOPMENT**

LYE BOON JET

UNIVERSITI TUNKU ABDUL RAHMAN

**AN INTERACTIVE SWIFT PROGRAMMING LANGUAGE E-LEARNING
PLATFORM FOR IOS APPLICATION DEVELOPMENT**

LYE BOON JET

**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**

September 2021

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 :



Name :

LYE BOON JET

ID No. :

17UEB01376


Date :

15/9/2021

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**AN INTERACTIVE SWIFT PROGRAMMING LANGUAGE E-LEARNING PLATFORM FOR IOS APPLICATION DEVELOPMENT**” was prepared by **LYE BOON JET** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : YONG YOKE LENG

Date : 15/9/2021

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.

© 2021, Lye Boon Jet. All right reserved.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my research supervisor, Dr. Yong Yoke Leng for her invaluable advice, guidance and her enormous patience throughout the development of the research. She is always willing to sacrifice her precious time to help me overcome the barriers that I faced in this project. Without her, it will be hard for me to deliver the project on time.

I would also like to thank all volunteers for testing my implemented system. I appreciate their passions and their constructive feedback for this project, so that I am able to improve the work in the future.

Lastly, I would also like to express my gratitude to my loving parents and friends who had helped and given me encouragement to complete this project.

ABSTRACT

Swift programming language is a famous programming language that is adopted by lots of developers to develop iOS, iPadOS, macOS, tvOS and watchOS applications as it is modern, fast and safe. However, Mac and iPad are usually required to learn Swift programming language and there's no existing Swift programming language-centric learning platform for learners. Therefore, this project is to develop an interactive Swift programming language e-learning platform for students. This platform can let students read materials, do exercises and graded quizzes, write codes in the embedded online code editors, view profiles, and chat with other online users. It also allows administrators to modify the course content, chat with students, as well as manage student's accounts and view student performance. Evolutionary prototyping has been adopted as the software development methodology to implement systems in several iterations. Requirements were gathered by looking at literature reviews. 12 students and 1 lecturer from UECS3263 iOS Application Development course have been chosen to test the system which was hosted in the web hosting services. A system usability score of 84.43 had been obtained from the tester's response. In short, all objectives were achieved and the platform was opened to all students who registered UECS3263 iOS Application Development in the May 2021 trimester.

TABLE OF CONTENTS

DECLARATION		i
APPROVAL FOR SUBMISSION		ii
ACKNOWLEDGEMENTS		iv
ABSTRACT		v
TABLE OF CONTENTS		vi
LIST OF TABLES		xi
LIST OF FIGURES		xiii
LIST OF SYMBOLS / ABBREVIATIONS		xx
LIST OF APPENDICES		xxi
 CHAPTER		
1	INTRODUCTION	1
1.1	Introduction	1
1.2	Problem Statement	3
1.2.1	There's only a limited number of e-learning platforms for learning Swift programming.	3
1.2.2	Swift programming language requires specific devices to run natively.	4
1.2.3	Some e-learning platforms for Swift programming are using outdated Swift versions.	6
1.2.4	There are many online Swift compilers purely for the user to compile Swift code, without any tutorial and any peer mentoring.	6
1.3	Project Objectives	6
1.4	Project Solution	7
1.5	Project Approach	7
1.6	Project Scope	8
1.6.1	Target Users	8
1.6.2	Front-end modules covered	8

	1.6.3 Back-end modules covered	10
	1.6.4 Modules that not covered in this project	10
	1.6.5 Assumptions of this project	11
	1.7 Conclusion of the Chapter	11
2	LITERATURE REVIEW	12
	2.1 Introduction	12
	2.2 Swift Programming Language	12
	2.2.1 Introduction of Swift Programming Language	12
	2.2.2 Comparison with Objective-C Programming Language	14
	2.2.3 Swift Programming Language in Education	17
	2.3 E-learning and Online Programming	18
	2.3.1 E-learning in Higher Education	18
	2.3.2 Online Programming	21
	2.3.3 Existing Online Programming Platform	22
	2.4 Methodology Research	25
	2.4.1 Waterfall Development	25
	2.4.2 Evolutionary Prototype Development	26
	2.4.3 Agile Development	28
	2.4.4 Comparison of Software Methodology	28
	2.5 Conclusion of the Chapter	29
3	METHODOLOGY	31
	3.1 Chosen Software Development Methodology	31
	3.1.1 Requirements Gathering and Analysis	31
	3.1.2 System Design and Implementation	32
	3.1.3 Software Testing	33
	3.1.4 Software Deployment	33
	3.2 Web Hosting Services	33
	3.2.1 InfinityFree	34
	3.2.2 Hostinger	34
	3.2.3 Awardspace	34
	3.2.4 Comparison of Web Hosting Service	34
	3.3 Development and Prototyping Tools	36
	3.3.1 Programming Language	36

	3.3.2 Framework	36
	3.3.3 Server and Database System	37
	3.3.4 Integrated Development Environment	37
	3.3.5 Prototyping Tools	37
	3.3.6 Online Code Editor	37
	3.4 Project Plan	38
	3.4.1 Work Breakdown Structure and Gantt Chart	38
	3.5 Conclusion of the Chapter	38
4	PROJECT SPECIFICATION	39
	4.1 Introduction	39
	4.2 Requirements Specifications	39
	4.2.1 Functional Requirements	39
	4.2.2 Non-functional Requirements	40
	4.2.3 Assumptions	40
	4.3 Use Case Diagram	40
	4.4 Use Case Description	41
	4.5 Preliminary User Interface Design	54
	4.6 Conclusion of the Chapter	64
5	SYSTEM DESIGN	65
	5.1 System Architecture Design	65
	5.2 Designed UML Diagrams	66
	5.2.1 Class Diagram	66
	5.2.2 Activity Diagram	66
	5.3 Database Design	72
	5.3.1 Entity Relationship Diagram (ERD)	72
	5.3.2 Data Dictionary	72
	5.4 Implemented User Interface	80
	5.4.1 Introduction Screen	80
	5.4.2 Login Screen	81
	5.4.3 Reset Password Screen	82
	5.4.4 Onboarding Screen	83
	5.4.5 Home Screen and Navigation Bar	83
	5.4.6 Topic Lesson Screen	87
	5.4.7 Exercise Screen	94

	5.4.8 Graded Quiz Screen	97
	5.4.9 Profile Screen	102
	5.4.10 Code Playground Screen	106
	5.4.11 Chat Box Screen	107
	5.4.12 Register Student Screen	108
	5.4.13 Error Screen	108
6	SYSTEM IMPLEMENTATION	111
6.1	System Modules	111
6.2	Student and Administrator Modules	114
6.2.1	Login Module	114
6.2.2	Logout Module	115
6.2.3	Read Topic Lesson Modules	116
6.2.4	Online Code Editor Modules	119
6.2.5	Exercise Module	120
6.2.6	Chat Box Module	122
6.3	Student-only Modules	125
6.3.1	Reset Password Module	125
6.3.2	Graded Quiz Module	128
6.3.3	Student Profile Module	131
6.4	Administrator-only Modules	134
6.4.1	Administrator Profile Module	134
6.4.2	Register Student Module	136
6.4.3	Modify Course Content Module	139
7	SYSTEM TESTING	141
7.1	Testing Types	141
7.2	Unit Testing	141
7.3	Integration Testing	142
7.4	Usability Testing and UAT Testing	143
7.4.1	UAT Testing Result	143
7.4.2	Usability Testing Result	146
8	CONCLUSION	150
8.1	Conclusion	150
8.2	Limitations	151
8.3	Recommendations	151

REFERENCES	153
APPENDICES	157

LIST OF TABLES

Table 2-1: Comparison of Swift Programming Language and Objective-C Programming Language (Karczewski, 2020; Altexsoft, 2018)	14
Table 2-2: Comparison of 5 researched existing online programming platform	23
Table 2-3: Comparison between three development methodologies (Dennis et al, 2015)	29
Table 3-1: Comparison between Three Web Hosting Services	35
Table 4-1: Use Case Description of Read Materials	41
Table 4-2: Use Case Description of Edit Codes in Code Editor	42
Table 4-3: Use Case Description of Do Non-Graded Exercise	43
Table 4-4: Use Case Description of Take Graded Quizzes	44
Table 4-5: Use Case Description of View Profile	45
Table 4-6: Use Case Description of Chat in Chat Box	46
Table 4-7: Use Case Description of Login the System	47
Table 4-8: Use Case Description of Reset Password	48
Table 4-9: Use Case Description of Logout the System	49
Table 4-10: Use Case Description of Manage Student Account	50
Table 4-11: Use Case Description of Modify Lesson Content	52
Table 5-1: Data dictionary for the table “topicitles”	72
Table 5-2: Data dictionary for the table “topicsections”	73
Table 5-3: Data dictionary for the table “exercises”	74
Table 5-4: Data dictionary for the table “quizzes”	75
Table 5-5: Data dictionary for the table “quizhistories”	76
Table 5-6: Data dictionary for the table “students”	78
Table 5-7: Data dictionary for the table “admins”	78

Table 5-8: Data dictionary for the table “chats”	79
Table 6-1: System Module Table	111
Table 7-1: Summarized unit testing results	141
Table 7-2: Summarized integration test results	142
Table 7-3: Summarized student UAT test results	143
Table 7-4: Summarized administrator UAT test results	144
Table 7-5: User Satisfaction Survey template	146
Table 7-6: SUS result	148
Table 7-7: Graded and Rating of SUS score (Usabilitest, n.d.)	148

LIST OF FIGURES

Figure 1-1: List of available programming languages in BitDegree	3
Figure 1-2: List of available programming languages in W3Schools	4
Figure 1-3: Available platform for Swift Playground based on the Apple Official Website (Apple, n.d.)	4
Figure 1-4: Minimum requirements for Xcode based on the Apple Developer Website (Apple, n.d.)	5
Figure 1-5: Market Share of Worldwide Desktop Operating System, as on January 2021 (Statcounter, n.d.)	5
Figure 1-6: Flow of evolutionary prototyping (Collegenote, n.d.)	8
Figure 2-1: First example of Objective-C code and Swift code comparison (Hubbatt, 2017)	16
Figure 2-2: Second example of Objective-C code and Swift code comparison (Fojtik, 2020)	17
Figure 2-3: Flow of waterfall development (Dennis et al, 2015)	26
Figure 2-4: Flow of evolutionary prototyping development (Dennis et al, 2015)	27
Figure 2-5: Flow of agile development (Dennis et al, 2015)	28
Figure 4-1: Welcome Page	54
Figure 4-2: Student Login Page	54
Figure 4-3: Admin Login Page	55
Figure 4-4: Change Password Page for First Time Login	55
Figure 4-5: Reset Password First Page	56
Figure 4-6: Reset Password Second Page	56
Figure 4-7: Home Page	57
Figure 4-8: Mega Menu and Navigation Bar	57
Figure 4-9: Listing of Content (Eg: Lessons)	58
Figure 4-10: Lesson Content Page	58

Figure 4-11: Exercise Page	59
Figure 4-12: Graded Quiz Page	59
Figure 4-13: Quiz Result Page after completing all Questions	60
Figure 4-14: Code Editor Page	60
Figure 4-15: Student Profile Page	61
Figure 4-16: Chat Box Page	61
Figure 4-17: Admin Profile	62
Figure 4-18: Delete Student Account Modal View Page	62
Figure 4-19: Register Student Page	63
Figure 4-20: Lesson Content List with Add, Modify and Delete Buttons	63
Figure 4-21: Course Content Editor	64
Figure 5-1: Designed System Architecture	66
Figure 5-2: Activity Diagram for Chat in Chat Box	67
Figure 5-3: Activity Diagram for Do Non-Graded Exercises	67
Figure 5-4: Activity Diagram for Edit Codes in Code Compiler	68
Figure 5-5: Activity Diagram for Login the System	68
Figure 5-6: Activity Diagram for Logout the System	69
Figure 5-7: Activity Diagram for Manage Student Account	69
Figure 5-8: Activity Diagram for Modify Lesson Content	69
Figure 5-9: Activity Diagram for Read Materials	70
Figure 5-10: Activity Diagram for Reset Password	70
Figure 5-11: Activity Diagram for Take Graded Quizzes	71
Figure 5-12: Activity Diagram for View Profile	71
Figure 5-13: Entity Relationship Diagram	72
Figure 5-14: Implemented Welcome Screen	80

Figure 5-15: Implemented About Screen	81
Figure 5-16: Student Login Screen	81
Figure 5-17: Admin Login Screen	82
Figure 5-18: Reset Password Screen	82
Figure 5-19: Onboarding Screen	83
Figure 5-20: Student Home Screen	84
Figure 5-21: Student Navigation Bar for Lessons	84
Figure 5-22: Student Navigation Bar for Exercises	85
Figure 5-23: Student Navigation Bar for Graded Quizzes	85
Figure 5-24: Student Navigation Bar for Other Functions	86
Figure 5-25: Administrator Home Screen	86
Figure 5-26: Administrator Navigator Bar	87
Figure 5-27: Topic List Screen	88
Figure 5-28: Topic Screen	88
Figure 5-29: Sections in the Topic Screen	89
Figure 5-30: Section with Opened Online Code Editor in the Topic Screen	89
Figure 5-31: Admin Topic List Screen	90
Figure 5-32: Admin Add Topic Modal Screen	90
Figure 5-33: Admin Edit Topic Name Modal Screen	91
Figure 5-34: Admin Delete Topic Name Modal Screen	91
Figure 5-35: Upper Part of Admin Add Topic Section Modal Screen	92
Figure 5-36: Bottom Part of Admin Add Topic Section Modal Screen	92
Figure 5-37: Upper Part of Admin Edit Topic Section Modal Screen	93
Figure 5-38: Bottom Part of Admin Edit Topic Section Modal Screen	93
Figure 5-39: Delete Topic Section Screen	94

Figure 5-40: Exercise List Screen	95
Figure 5-41: Student Exercise Screen	95
Figure 5-42: Admin Exercise Screen	96
Figure 5-43: Admin Add Exercise Question Modal Screen	96
Figure 5-44: Admin Edit Exercise Question Modal Screen	97
Figure 5-45: Admin Delete Exercise Question Modal Screen	97
Figure 5-46: Graded Quiz List Screen	98
Figure 5-47: Graded Quiz Preparing Screen	98
Figure 5-48: First Question in Graded Quiz Screen	99
Figure 5-49: Last Question in Graded Quiz Screen	99
Figure 5-50: Graded Quiz Result Screen	100
Figure 5-51: Admin Graded Quiz Screen	100
Figure 5-52: Admin Add Graded Quiz Question Modal Screen	101
Figure 5-53: Admin Edit Graded Quiz Question Modal Screen	101
Figure 5-54: Admin Delete Graded Quiz Question Modal Screen	102
Figure 5-55: Upper Part of Student Profile Screen	103
Figure 5-56: Bottom Part of Student Profile Screen	103
Figure 5-57: Admin Profile Screen	104
Figure 5-58: Upper Part of Admin Profile Screen with Searched Student	104
Figure 5-59: Bottom Part of Admin Profile Screen with Searched Student	105
Figure 5-60: Admin Delete Student Modal Screen	105
Figure 5-61: Admin Student List Screen	106
Figure 5-62: Code Playground Screen	106
Figure 5-63: Student Chat Box Screen	107
Figure 5-64: Admin Chat Box Screen	107

Figure 5-65: Register Student Screen	108
Figure 5-66: Error 404 Screen	109
Figure 5-67: Error 403 Screen for New Student	109
Figure 5-68: Error 403 Screen for Existing Student	110
Figure 6-1: Code segment of the “LoginController” controller class	115
Figure 6-2: Code segment of the “UserActivity” middleware class	115
Figure 6-3: Code segment of the logout method from the “AuthenticatesUsers” trait	116
Figure 6-4: Code segment of the related topic section data fetching method via Axios API	117
Figure 6-5: Upper part code segment of mapping stored data into HTML components in render method	118
Figure 6-6: Bottom part code segment of mapping stored data into HTML components in render method	119
Figure 6-7: Code segment of handling embedded online code editor opening or closing	120
Figure 6-8: Code segment of checking correct answer from a question	121
Figure 6-9: Code segment of submitting answers from a question	122
Figure 6-10: Code segment of converting MySQL date time format into readable format.	123
Figure 6-11: Code segments of finding online user in “ChatController” controller class	124
Figure 6-12: Code segments of mapping stored chat data into HTML elements	125
Figure 6-13: Code segment of sending chats into database via Axios API	125
Figure 6-14: Code segment of delete chat based on chat ID via Axios API	125
Figure 6-15: Code segments of “ForgotPasswordController” controller class	126
Figure 6-16: Code segments of validating password format and length	127

Figure 6-17: Code segments of updating student's password via Axios API with PUT method	127
Figure 6-18: Code segments of "updatePassword" method in "StudentController" controller class	128
Figure 6-19: Code segment of starting the quiz	129
Figure 6-20: Code segments of submitting the answers from the quiz	129
Figure 6-21: Code segments of storing quiz history once the quiz is completed	130
Figure 6-22: Code segments of calculating quiz result into percentage form	130
Figure 6-23: Code segments of showing each question answer after completing the quiz	131
Figure 6-24: Code segment of loading student credentials based on student ID	132
Figure 6-25: Code segment of mapping related quiz histories in reverse order	132
Figure 6-26: Code segment of finding latest score from the same quizzes and calculating average performance	133
Figure 6-27: Code segment of finding student data based on student ID	134
Figure 6-28: Code segments of delete student modal view	135
Figure 6-29: Code segments of deleting student via Axios API with DELETE method	135
Figure 6-30: Code segments of "deleteStudent" method in "AdminController" controller class	136
Figure 6-31: Code segment of loading existing students via Axios API with GET method and React lifecycle.	136
Figure 6-32: Code segment of checking student Email format	137
Figure 6-33: Code segment of checking duplication of credentials with existing students	137
Figure 6-34: Code segment of adding new student via Axios API with POST method	138
Figure 6-35: Code segment of "registerAStudent" method in "AdminController" controller class	138

- Figure 6-36: Code segment of add exercise question via Axios API with POST method 139
- Figure 6-37: Code segment of edit exercise question via Axios API with PUT method 140
- Figure 6-38: Code segment of delete exercise question via Axios API with DELETE method 140
- Figure 6-39: Code segment of “addExercise” method, “editExercise” method and “deleteExercise” method in “AdminController” class controller 140

LIST OF SYMBOLS / ABBREVIATIONS

MVC	Model-View-Controller
LLVM	Low Level Virtual Machine
ERD	Entity Relationship Diagram
UAT	User Acceptance Testing
WWDC	Worldwide Developer Conferences
UML	Unified Modelling Language
API	Application Programming Interface
SUS	System Usability Scale

LIST OF APPENDICES

APPENDIX A: Work Breakdown Structure	157
APPENDIX B: Project Gantt Chart	160
APPENDIX C: User Interface Flow Chart	163
APPENDIX D: Use Case Diagram	165
APPENDIX E: Designed Class Diagram	166
APPENDIX F: Unit Test Cases	167
APPENDIX G: Integration Test Cases	202
APPENDIX H: Student UAT Test Cases	211
APPENDIX I: Administrator UAT Test Cases	297
APPENDIX J: User Satisfaction Survey Response	308

CHAPTER 1

INTRODUCTION

The purpose of this chapter is to provide a short but crucial briefing for this project. There are six sections in this introduction chapter. Section 1.1 is to brief about the overall background of this project. Section 1.2 is to provide the statement of problems in detail. Section 1.3 is to list out the detailed project objectives. Section 1.4 is to explain the overall project solution to solve the stated problems. Section 1.5 is to brief the approach to conduct this project. Section 1.6 is to list out the project scope with details, including front-end, back-end, users involved, and out-of-coverage scopes.

1.1 Introduction

Ever since the mobile devices' introduction in the 2000s, smartphones and tablets have become the most important electronic devices in our life. We use mobile devices for social media, entertainment, gaming, productivity, etc. Because of that, the applications for mobile devices increase rapidly to fulfil the demand of the users. It has also contributed to the influx of mobile app developers trying to cater to various apps' market requests.

Currently, the two main operating systems dominating the smartphone market are iOS and Android. According to Diffen (n.d.), iOS is an operating system developed by Apple, and it was officially announced on July 29, 2007. iOS is almost closed source with some open-source components, and it is only available within Apple devices, such as iPhone, iPad and iPod touch. Whereas Android is an operating system developed by Google and it was officially announced on September 23, 2008. Android is an open-sourced mobile operating system and it is pre-installed by various smartphone vendors.

Although Android has a larger market share than iOS, the application revenue in the iOS platform is always higher than that of the Android platform. According to Curry (2021), the revenue of the iOS App Store in the first, second, third, and fourth quarter of the year 2020 are \$15.0 billion, \$17.3 billion, \$18.6 billion, and \$21.4

billion respectively. On the other hand, the revenue of Android Google Play Store in the first, second, third, and fourth quarter of the year 2020 are \$8.3 billion, \$9.6 billion, \$10.3 billion and \$10.4 billion respectively. The statistics listed encourage mobile app developers to explore and develop various applications in the iOS platform.

To develop an iOS application, a developer may use Objective-C, Swift, or a combination of both. In recent years, the number of Swift developers has been increasing while Objective-C developers are decreasing. JetBrains (n.d.) had performed an analysis to determine the percentage of Swift and Objective-C developers. In the year 2019, there are 53% pure-Swift developers, 15% pure-Objective-C developers and 31% hybrid developers who utilised both Swift and Objective-C at the same time. But in the year 2020, pure-Swift developers increased to 70%, whereas pure-Objective-C developers had fallen to 17%, and hybrid developers had decreased to 13%. Therefore, this statistic shows that the Swift programming language is slowly dominating for iOS app development. Hence, it is necessary to learn the Swift language for this domain.

An online programming platform is a good choice for students to learn a new programming language or enhance a programming language. However, some students might face some barriers when using an online programming platform to learn the Swift programming language. For example, it is hard to find an online platform that only focuses on the Swift programming language. There is also no multi-user communication system to enable students to interact with one another to exchange ideas and help one another. Apart from that, they are also unable to express their feelings during or after learning the Swift programming language. Zhang et al. (2018) found out that communication and discussion of ideas are crucial to support online learning for students.

Therefore, the project aims to develop an online Swift programming centric platform for students to ensure they can learn Swift programming with just a browser and Internet connection. Quizzes and exercises will be provided to train and evaluate their knowledge after they have completed each topic. Online code editor and compiler will also be provided to let students code their desired Swift code and show

their expected output. Finally, chat rooms will be available for students to share their solutions with other students, and support for students who face difficulties when learning.

1.2 Problem Statement

Before proposing a solution, one or more of the problems currently faced must be made clear. In this project, there are four main problems found via online research. These problems are:

1.2.1 There's only a limited number of e-learning platforms for learning Swift programming.

Most of the available e-learning platforms for programming only focus on the most popular programming languages such as C++, Java, C#, and HTML. It is hard to find an online-programming platform that is Swift-centric. For example, refer to Figure 1-1 and Figure 1-2, W3School, Edx, and BitDegree do not provide Swift Programming Language for students to learn.

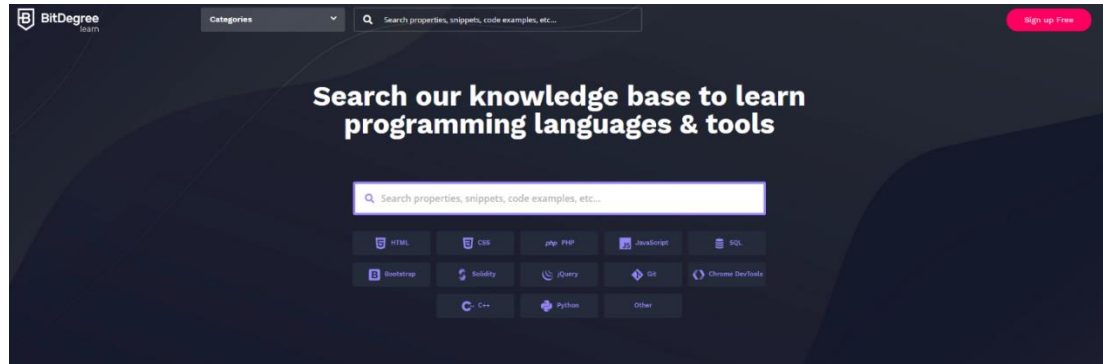


Figure 1-1: List of available programming languages in BitDegree

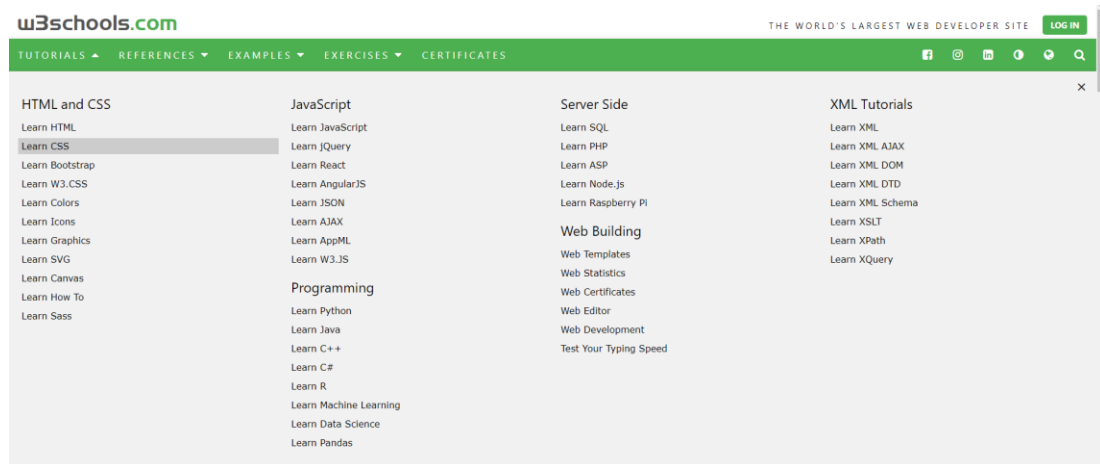


Figure 1-2: List of available programming languages in W3Schools

1.2.2 Swift programming language requires specific devices to run natively.

Swift programming is created and developed by Apple Inc. and it is used in Apple operating systems such as iOS, iPadOS and macOS. Therefore, the most convenient way to learn Swift programming is to use Swift Playground or Xcode. However, according to Figure 1-3 and Figure 1-4, Swift Playground is only available on iPad and Mac. In contrast, Xcode is only available on Mac, and not all students have these devices to learn Swift programming language. According to Figure 1-5, Statcounter (n.d.) shows that as of January 2021, Windows has 76.26% of the market share whereas macOS (formerly named OS X) has only 16.91% of the market share in the desktop operating system market.



Figure 1-3: Available platform for Swift Playground based on the Apple Official Website (Apple, n.d.)

Minimum requirements and supported SDKs

Xcode Version	Minimum OS Required	SDK	Architecture	OS	Simulator	Swift
Xcode 12.3	macOS Catalina 10.15.4 (Intel-based Mac)	iOS 14.3	x86_64	iOS 9.0-14.3	iOS 10.3.1-14.3	Swift 4
		macOS 11.1	armv7	iPadOS 13.0-14.3	tvOS 10.2-14.3	Swift 4.2
Xcode 12.2	macOS Catalina 10.15.4 (Intel-based Mac) macOS Big Sur 11.0 (Apple silicon Mac)	iOS 14.2	x86_64	iOS 9.0-14.2	iOS 10.3.1-14.2	Swift 4
		macOS 11	armv7	iPadOS 13.0-14.2	tvOS 10.2-14.2	Swift 4.2
Xcode 12.1	macOS Catalina 10.15.4 (Intel-based Mac) macOS Big Sur 11.0 (Apple silicon Mac)	tvOS 14.3	armv7s	macOS 10.9-11.0	watchOS 3.2-7.2	Swift 5.3
		watchOS 7.3	arm64	tvOS 9.0-14.3	watchOS 3.2-7.1	Swift 5.3
Xcode 12	macOS Catalina 10.15.4 (Intel-based Mac)	DriverKit 20.0	arm64e	tvOS 9.0-14.2	watchOS 2.0-7.0	Swift 5.3
		macOS 10.15.6	armv7	iOS 9.0-14.1	iOS 10.3.1-14.1	Swift 4
Xcode 11.7	macOS Catalina 10.15.2	tvOS 14	armv7s	iPadOS 13.0-14.1	tvOS 10.2-14.0	Swift 4.2
		watchOS 7.0	arm64	macOS 10.9-11.0	watchOS 2.0-7.0	Swift 5.3
Xcode 11.6	macOS Catalina 10.15.2	DriverKit 20.0	arm64e	tvOS 9.0-14.0	watchOS 2.0-7.0	Swift 5.3
		macOS 10.15.6	armv7	iOS 8.0-13.7	iOS 10.3.1-13.7	Swift 4
Xcode 11.5	macOS Catalina 10.15.2	tvOS 13.4	armv7s	iPadOS 13.0-13.7	tvOS 10.2-13.4	Swift 4.2
		watchOS 6.2	arm64	macOS 10.6-10.15.6	watchOS 3.2-6.2	Swift 5.2
Xcode 11.4	macOS Catalina 10.15.2	DriverKit 19.0	arm64e	tvOS 9.0-13.4	watchOS 2.0-6.2	Swift 5.2
		macOS 10.15.6	armv7	iOS 8.0-13.6	iOS 10.3.1-13.6	Swift 4
Xcode 11.3	macOS Catalina 10.15.2	tvOS 13.4	armv7s	iPadOS 13.0-13.6	tvOS 10.2-13.4	Swift 4.2
		watchOS 6.2	arm64	macOS 10.6-10.5.6	watchOS 3.2-6.2	Swift 5.2
Xcode 11.2	macOS Catalina 10.15.2	DriverKit 19.0	arm64e	tvOS 9.0-13.4	watchOS 2.0-6.2	Swift 5.2
		macOS 10.15.4	armv7	iOS 8.0-13.5	iOS 10.3.1-13.5	Swift 4
Xcode 11.1	macOS Catalina 10.15.2	tvOS 13.4	armv7s	iPadOS 13.0-13.5	tvOS 10.2-13.4	Swift 4.2
		watchOS 6.2	arm64	macOS 10.6-10.15.4	watchOS 3.2-6.2	Swift 5.2
Xcode 11.0	macOS Catalina 10.15.2	DriverKit 19.0	arm64e	tvOS 9.0-13.4	watchOS 2.0-6.2	Swift 5.2
		macOS 10.15.2	armv7	iOS 8.0-13.5	iOS 10.3.1-13.5	Swift 4

Figure 1-4: Minimum requirements for Xcode based on the Apple Developer Website (Apple, n.d.)

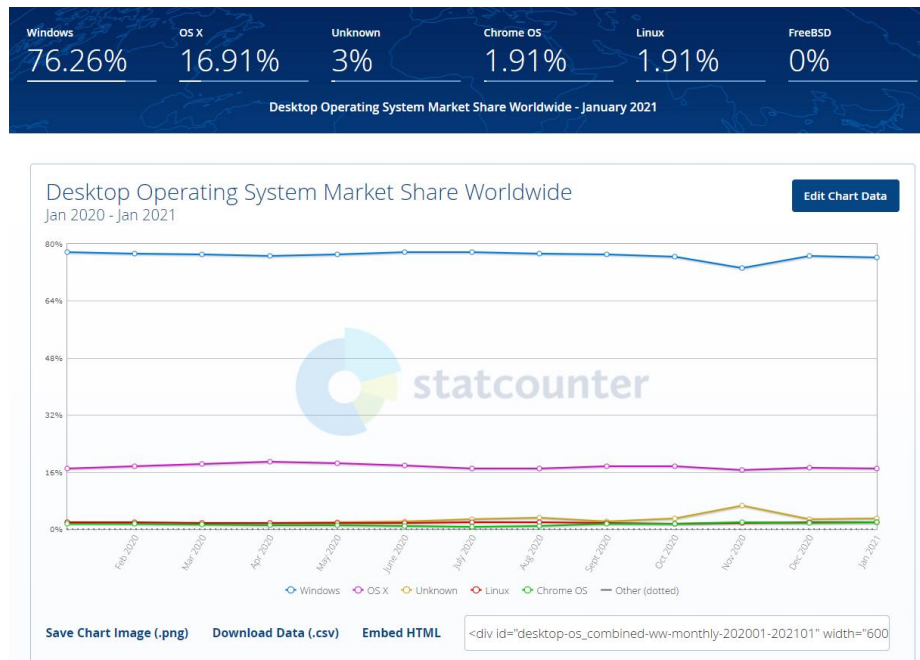


Figure 1-5: Market Share of Worldwide Desktop Operating System, as on January 2021 (Statcounter, n.d.)

1.2.3 Some e-learning platforms for Swift programming are using outdated Swift versions.

As Swift language is continuously updated by having new features added or removing some features, and the overall syntax and structure might be changed during the update, some learned syntax is not available in the latest Swift language version. Although many online Swift compilers use the latest Swift version, some e-learning programming platforms are still using the old version of Swift compiler. As of September 16, 2020, the latest Swift version is 5.3 (Swift, n.d.). However, Tutorialspoint uses Swift 4.0, and Sololearn uses Swift 4.2.

1.2.4 There are many online Swift compilers purely for the user to compile Swift code, without any tutorial and any peer mentoring.

Most of the online Swift compilers, such as IDEOne, TECHIE Delight, OnlineGDB and Paiza.io only provide the code editor and output display. It does not offer any step-by-step tutorials briefing or forums/chatbox systems for students to discuss, ask questions, or share opinions. The students are unable to learn programming languages well with zero concepts of it.

1.3 Project Objectives

The main objective of the project is to develop a web-based platform for UTAR students to learn Swift programming language and enhance their skills for preparation of iOS application development course.

The four sub-objectives are:

1. To analyze the existing online programming platform with the perspective of strengths and weaknesses.
2. To research the characteristics of Swift Programming Language and compare it to its predecessor, Objective-C programming language.
3. To design and implement the web application by adopting evolutionary prototyping method.
4. To test and evaluate the web application functionalities by using unit testing, integration testing, usability testing and user acceptance testing.

1.4 Project Solution

To achieve the project objectives and solve the problems stated in Section 1.2, a web application that supports the Swift programming language has been developed. The model-view-controller had been used as the web application architecture. Svirca (2020) states that model-view-controller can be divided into three vital parts: model, view and controller. The model-view-controller architecture is usually used to develop mobile and web applications nowadays. Model-view-controller provides some advantages for developing web applications, such as speeding up the process of development and creating multiple views components.

In model-view-controller architecture, a model usually connects to a database to handle the data and its logics. A view usually acts as a front-end by generating user interface components for the user to view and interact. A controller acts as an interconnector to connect and communicate both view and the model by receiving requests (Svirca, 2020).

For the web application framework, Laravel has been used to develop the online programming platform for Swift. This is because Laravel uses a model-view-controller as the web application framework. It uses PHP as the programming language for development, so it is easy to use, fast, and elegant. Laravel can also be used to develop both front-end and back-end systems. Besides, WampServer and Awardspace had been used as the local server-side and global server-side respectively for this web application, and MySQL had been used as the database to store data.

1.5 Project Approach

In this project, the methodology that had been applied to develop the system was evolutionary prototyping. According to Figure 1-6, After the initial requirements had been gathered in the planning and analysis phase, a system prototype, which is also called a part of the system, had been designed to validate and provide comments. A loop from the design evolutionary prototyping methodology has been instrumental in collecting feedback from the user until the final prototype is deemed suitable. After that, the implementation phase had been reached to become a whole, workable and fully functioning system.

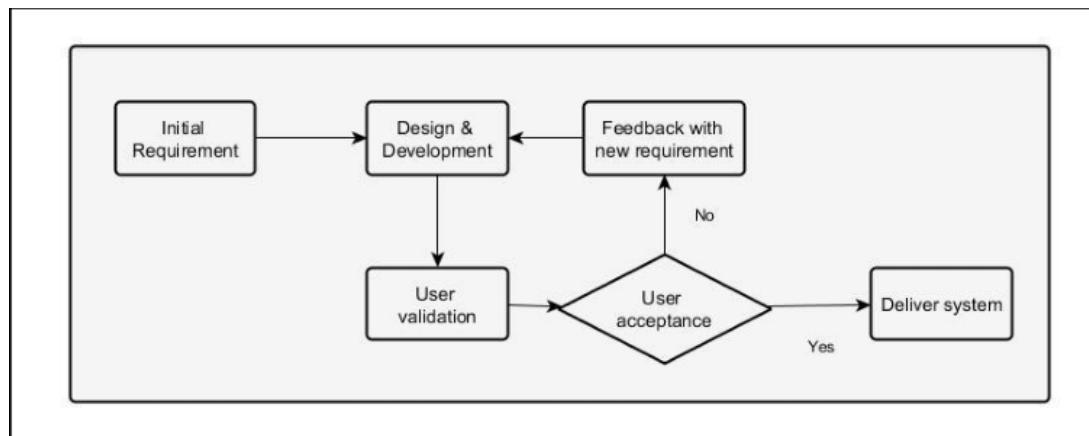


Figure 1-6: Flow of evolutionary prototyping (Collegenote, n.d.)

1.6 Project Scope

1.6.1 Target Users

i. Students

Students may read the materials, such as concepts and briefings of Swift programming language provided by the platform. Students may also take quizzes and exercises after they have completed each topic. For graded quizzes, the students will receive instant feedback once they have completed the quizzes. Students are also able to chat with other students via chat room.

ii. Administrators

The administrators may add new materials, update existing materials, or delete the old materials that are no longer used. Administrators may also manage the student accounts for this platform by registering or deleting the account, as well as using chat room to chat with other students.

1.6.2 Front-end modules covered

1.6.2.1 Login Function

- i. The user can login using Student ID and a randomly generated password (for the first time) and self-reset their password.
- ii. If the user login with their account for the first time, the user will be prompted to reset the password.

- iii. The system would provide a reset password option if the user forgot his/her password.

1.6.2.2 Lesson Lists

- i. The lesson consists of basic data types, conditional statements, loops, functions, classes, structures, etc.
- ii. Users can follow the brief tutorial and concepts of Swift programming language.
- iii. The user can take non-graded exercises as a practice, such as fill in the blanks with code segments.
- iv. Users can edit codes via code editor in each lesson.
- v. Users can take graded quizzes after completing each topic, such as multiple-choice questions.
- vi. Users can re-attempt each graded quiz with unlimited trials.
- vii. Users can view the results once the user completed the graded quizzes.

1.6.2.3 Profile Function

- i. Basic statistics for overall performance.
- ii. Graded quiz attempt history.
- iii. Logout function.

1.6.2.4 Chat Room

- i. The chat is in the form of a group-chat form with multiple users.
- ii. Each user in the chat room can be shown the name and the student ID or admin ID, with a bracket.

1.6.2.5 Administrative Matters (Only available for administrator)

- i. Register a student's account based on the list regarding the students who registered for the iOS Application Development course provided by the Faculty General Officer, with their student ID as username, and automatically generate a random password for the user.
- ii. Delete the student's account if the student has graduated, terminated or withdrawn from the studies.
- iii. Modify the exercises, quizzes and tutorial briefings.

1.6.3 Back-end modules covered

1.6.3.1 Accounts

- i. Account data is stored in the database.
- ii. Attributes for account are student ID, student name, quizzes attempting history, and awarded grade in each quiz.
- iii. Overall performance: Total score obtained from all topics divided by the maximum score from all topics, the only the latest attempt score will be obtained to calculate the overall performance.
- iv. User password has been encrypted for security purposes.

1.6.3.2 Quizzes and Exercises

- i. Validate the answers filled in the blanks with case sensitivity.
- ii. Validate the answer chosen in multiple choice questions.
- iii. Store total marks earned in each graded quiz and its attempt.

1.6.3.3 Code Editor and Compiler

- i. Pre-load defined codes in the pre-existing Swift compiler for each topic.

1.6.3.4 Chat Box

- i. Store chats into the database and allows chat deletion, to reduce the server stress.

1.6.4 Modules that not covered in this project

Due to the limitation of time and technologies, the following items did not covered in the project:

- i. Storyboard features for iOS app UI layout, such as dragging UI components.
- ii. UIKit and SwiftUI tutorials and exercises, as current online Swift compilers are unable to import and compile UIKit and SwiftUI libraries.
- iii. Registration of accounts by users, as this module was handled by the system administrator.
- iv. Mail notification for reset password code and temporary password to students.

1.6.5 Assumptions of this project

- i. Swift materials used are based on Swift 5.3, but it is compatible with Swift 5, Swift 4.2 and Swift 4.
- ii. No time limit for each graded quiz. Therefore, students can ease themselves to take the quiz without any time restriction panics.
- iii. The administrator is the lecturer who handles the iOS Application Development course.

1.7 Conclusion of the Chapter

In conclusion, four problem statements have been defined for this project. One main objective with four sub-objectives is also defined to solve the stated problems. This project has been used as model-view-architecture to develop the web application and evolutionary prototyping has been adopted as the project approach.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter has three main sections which are Section 2.2, Section 2.3 and Section 2.4. Section 2.2 discusses the background of Swift programming language, which has been used as a programming medium for this project, and how the interconnection between Swift programming language with education. Next, Section 2.3 describes e-learning in higher education and the online programming platform, including the reviewing and comparison of existing online programming systems. Last, Section 2.4 briefs about the researched methodologies and comparison of these software development methodologies.

2.2 Swift Programming Language

It is vital to explain why the Swift programming language had been used as a medium of online programming instead of other programming languages. Therefore, this section had discussed the overview briefing of Swift programming language with its characteristics, the comparison of its predecessor programming language, Objective-C. After that, the section also stated about the importance of Swift Programming Language in the education sector, to attract developers and students to learn this programming language.

2.2.1 Introduction of Swift Programming Language

Swift Programming Language is a powerful programming language that was announced and released by Apple in the WWDC 2014, and it is fully tailored to cope with Apple's operating system platforms such as iOS, iPadOS, macOS, watchOS and tvOS (Apple, n.d.; Swift, n.d.). Apple Inc. claims that Swift Programming Language has some several major characteristics that attract developers to code it with fun and interactive, which are:

- i. **Modern.** Swift programming language provides a clean and concise syntax for developers so that they need not to memorize a lot of unwanted things for the particular language, and increase the readability and maintainability of the APIs. (Apple, n.d.; Chernova and Nazarov, 2020)

The most focused point is Swift programming language do not require any semicolons unless there are two expressions in the same line. Besides, Swift programming language also provides support for Unicode, especially emojis. It also supports tuples that store multiple values in a single variable at the same time, and return more than one value in a function. (Apple, n.d.)

- ii. **High Performance.** García et al (2015) and Apple (n.d.) mentioned that Swift programming language uses the same compiler that its predecessor programming language, Objective-C, which is called LLVM, or Low-Level Virtual Machine, in full name. Which means LLVM can convert Swift programming language into native code of Apple Devices, such as iPhone, iPad, iPod touch and Mac and fully optimize them.
- iii. **Safe.** Swift programming language always puts their focus on letting developers type their codes safely by eliminating the whole classes of codes that are not safe, to ensure no serious bugs will occur when the software has been finally released to the customers. To do so, Swift programming language always check and ensure the variables or constant are initialized with values before it was used in other places, provide type inference which auto convert the variable's data type to decrease unnecessary errors, and optionals that enables some necessary variables, constants or returned value to be null value to prevent massive runtime crashes during the execution. (Apple, n.d.)
- iv. **Compatibility and Interoperability.** Although there are lots of Swift-only frameworks, APIs and libraries nowadays, developers still can adopt the Swift programming language into existing Objective-C projects for compatibility, as Swift programming language is allowed to use Objective-C APIs and libraries (García et al, 2015). Furthermore, developers need not to make any changes in their existing code that uses Swift 4 when they compile it with the latest compiler, such as the compiler for Swift 5, as Swift 5 provides fully backward compatibility for Swift 4. (Apple, n.d.)

In the late of 2015, Apple had made Swift programming language into an open-source programming language by releasing the whole Swift programming language source code, vital libraries, bug reporter, debugger and package manager in both GitHub and Swift.org website (Apple, n.d.; Swift, n.d.). Which means everyone can contribute to the source code project not only to enhance and improve, but also express their feelings and experiences into it. As on March 3 2021, there are 856 contributors, 324 branches and 1775 tags in the Swift programming language repository.

2.2.2 Comparison with Objective-C Programming Language

To attract developers to learn and adopt Swift programming language into their iOS application development project, it is vital to compare Swift programming language with its predecessor, Objective-C programming language. Table 2-1 shows the comparison of both programming languages based on some categories, and briefing of each category has been provided.

Table 2-1: Comparison of Swift Programming Language and Objective-C Programming Language (Karczewski, 2020; Altexsoft, 2018)

Category	Swift	Objective-C
Performance	High and fast	Low and slow
Safety	High, as the existing of type-inferences and optionals	Low, as the existing of null pointers
Syntax and Complexity	Clean, short and simple	Messy, long, and high usage of symbols
Community Support	Open-source with huge communities in GitHub	Close-source and maintained over 30 years
Maintainability	Highly maintainable	Low maintainability
Available Platform	Xcode and Swift Playground	Xcode only

2.2.2.1 Performance

Apple claimed that Swift programming language is 2.6 times faster than its predecessor, Objective-C, as Objective-C uses runtime code compilation to execute the applications, which means extra indirection levels will be required to call an

object from other classes (Altexsoft, 2018). Karczewski (2020) also explained that Swift supports dynamic libraries which Objective-C does not support, to improve the performance of launching apps by using automatic reference counting to optimize the memory management.

According to Singh and Kaur (2017), they performed an algorithm of 100 times 100 matrices with 10 loops in different programming languages such as Swift, Objective-C, Java, and Python. They found out that Java and Swift only need 22 seconds and 29 seconds respectively to complete the algorithm, but for Objective-C and Python, they consume 145 seconds and 310 seconds respectively to finish the algorithm.

2.2.2.2 Safety

The Objective-C programming language has some design that may cause serious bugs to occur when executing the application. For example, Objective-C uses pointers to allocate the memory (García et al, 2015). Pointers might expose the value to let some malicious programmer have high accessibility of that particular data from the memory (Altexsoft, 2018).

In contrast, Swift programming language does not face the same issue as Objective-C, as it has removed the pointer to prevent any unprotected data from being exposed to others. Besides, type inferences and optionals were added into the Swift programming language to increase the security.

2.2.2.3 Syntax and complexity

One of many reasons that lots of developers criticize the Objective-C programming language is because of its messy syntax. Objective-C programming language not only requires lots of redundant symbols such as asterisks (*), sign (%) and semicolon (;), but also lots of lines, specialized string tokens and unwanted parentheses (Altexsoft, 2018). On the flip side, Swift programming language eliminates the use of asterisks and semicolon symbols, and makes the overall syntax become more English-like to make the whole code become shorter and readable. Figure 2-1 shows an algorithm in both Objective-C and Swift programming language.

Objective-C

```
const int count = 10;
double price = 23.55;

NSString *firstMessage = @"Swift is awesome. ";
NSString *secondMessage = @"What do you think?";
NSString *message = [NSString stringWithFormat:@"%s%s", firstMessage, secondMessage];

NSLog(@"%@", message);
```

Swift

```
let count = 10
var price = 23.55

let firstMessage = "Swift is awesome. "
let secondMessage = "What do you think?"
var message = firstMessage + secondMessage

print(message)
```

Figure 2-1: First example of Objective-C code and Swift code comparison (Hubbart, 2017)

García et al (2015) implemented the same code in both Swift and Objective-C programming language with one main method and two override methods. They used XMLParser to choose an XML parser example for evaluating the code lines, words, reserved words and switch case numbers from both codes. They found out that in these three methods, although Swift programming language have similar number of code lines needed as Objective-C, but the overall number of characters and words are lesser than that of Objective-C. Besides, based on Figure 2-2, Fojtik (2020) compared two codes based on Objective-C and Swift that runs same algorithm, and justified that Swift programming language only use 76% of length to implement the code, compare to Objective-C.

The sample code in Objective-C:

```
- (IBAction)buttonConvert:(id)sender
{
    double miles = 0;
    double meters = 0
    meters = [inputText.text doubleValue];
    miles = meters / 1.609;
    NSString *textMile = [NSString stringWithFormat:@"Distance: %.31f ml", miles];
    labelOutput.text = textMile;
}
```

The sample code in Swift:

```
@IBAction func buttonConvert(_ sender: UIButton) {
    var miles : Double
    var meters : Double
    meters = Double(inputText.text!)!
    miles = meters / 1609.0
    labelOutput.text = "Distance: \(miles) ml"
}
```

Figure 2-2: Second example of Objective-C code and Swift code comparison (Fojtik, 2020)

2.2.2.4 Maintainability

Swift programming language has high maintainability compare to Objective-C, as Swift only requires one file (.swift) to define a class, instead of two files which are header (.h) and implementation (.m) to declare one class in Objective-C programming language (García et al, 2015). This means that developers only need to focus one file instead of two files at the same time, and reduce the time needed to synchronize components in two files (Altexsoft, 2018).

2.2.3 Swift Programming Language in Education

Swift programming language is not only for development purposes, but also helps to ignite the curiosity and confidence to learn programming and problem-solving skills. Apple plays a vital role in educational sectors to make the whole programming learning environment to become more fun, easy, and interactive. To do so, Apple has provided a Swift Playground app for students to learn Swift programming language, and it is available in iPad and Mac, so students are able to download them via Apple App Store (Apple, n.d.). Fojtik (2020) states that through Swift Playground, students are able to control a game character by implementing the code segments into the blanks, so that the game character can walk, rotate and collect gems properly.

Furthermore, Swift programming language education helps students without any Swift fundamentals be able to develop some simple programs quickly and easily. Through Fojtik's (2020) research, he surveyed 48 first year computer science undergraduate students in the University of Ostrava, and asked them to provide comments on code that using Swift programming language, and found out that 84% of the students are able to do so correctly, compare to Objective-C. He also used questionnaire to conduct another research and 35 students are involved, and 86% of the students stated that Swift language is simple after they have completed the Swift programming course.

2.3 E-learning and Online Programming

As everything is moving towards online, including the education sectors, therefore this section discussed the overview of E-learning in Higher Institution and the importance of the online programming platform in the e-learning domain. Besides, 5 existing online programming platform systems were also evaluated and compared to have further information of how online programming platforms work.

2.3.1 E-learning in Higher Education

2.3.1.1 Definition of E-learning

Although the term "E-learning" looks simple and easy to understand as "digitized teaching and learning", there are still a lot of people, even though higher institutions are always confused about its true meaning. Nguyen et al (2019) categorized this term into four perspectives which are technology-driven, delivery-system-oriented, communication oriented and educational-paradigm-oriented.

In technology driven perspective, Nguyen et al (2019) defined E-learning as "the use of electronic means for various learning purposes", which means teachers and students will use digital devices with internet connection, whether it is either wired or wireless, to access learning resources and meet together in a virtual meeting room that act as a virtual classroom.

In a delivery-system-oriented perspective, E-learning is a type of learning that supplies education via electronic format and centers the accessibility and progress to obtain learning resources, rather than the final result itself (Nguyen et al,

2019). This represents that educators will deliver education and its systems to students through the connection of the internet, regardless of the places and time.

In communication-oriented definitions, according to Nguyen et al (2019), it can be explained as a way that allows teachers and students to use electronic communication devices and systems to interact and exchange knowledge with each other via communication. This means that if the student is in doubt, he/she may use a built-in microphone to ask the teacher regarding his question, and the teacher may answer him/her via microphone too.

In educational-paradigm-oriented perspective, Nguyen et al (2019) states that e-learning helps to improve the existing educational model, by applying new and various multimedia elements and Internet into it, to support the accessing of resources and services, and help students to build confidence and smooth the learning process.

In short, e-learning not only optimizes the assessments and examinations conducting method, but also enhances the resources for e-learning. It also stimulates educators to innovate and improve the teaching and learning methods (Nguyen et al, 2019).

2.3.1.2 E-learning and COVID-19

As COVID-19 pandemic had spread across the world since the Late 2019 to Early 2020, there were lots of regional or country governments that started to announce long periods of lockdown to prevent any further spreading of the virus. Because of that, lots of higher educational institutions had closed their campuses for a period of time (Radha et al, 2020). However, to ensure the planned academic activities will keep going as usual, many higher educational institutions started to use e-learning as a platform for teaching and learning, to replace the conventional classroom which is face-to-face (Ali, 2020).

E-learning is important for higher educational institutions because nowadays students can obtain massive amounts of information from the Internet easily, instead of just a book or printed materials provided by lecturers. In the research conducted

by Ali (2020), he found out that students in this era are inclined to have a strong connection with ICT technologies, as they have been exposed to ICT since they were born. According to Jesse (2015, cited in Ali, 2020), There are 99.8% of students can access to their smartphones to text, visit social media and numerous of mobile application that is able to download from the app store, rather than just talking physically, this shows that they have high acceptance degree towards the online learning. Radha et al (2020) also conducted a research to find out the trend and willingness of higher educational institution students to use e-learning platforms via Google Form questionnaire, and they found out that most of the students are interested in e-learning education and the use of the online platform is rising. In the questionnaire, 82.29% of students are willing to use e-learning platforms and 82.86% of students agree that their self-study skills have been improved while using e-learning platforms. This shows that E-learning has become a trend and unpreventable choices in higher education sectors.

2.3.1.3 E-learning Review

To make the e-learning environment become more flexible, fun and interactive, it is important to review the possible factors that influence e-learning experience and satisfaction of a student, and the ICT adopted for teaching and learning.

Nortvig et al (2018) separated the factors into three categories which are the design of course structure, roles and relations of an educator, and the identity of students and his/her learning community. According to Nortvig et al (2018), if an e-learning course structure is designed well, it will not only provide positive interactions between internet-based and non-internet based academic activities, but also the students, teachers and the course content. In teacher roles and relations, e-learning platform helps educators to build a strong teaching existence and build a positive online learning relationship (Nortvig et al, 2018). Whereas for student identity sectors, a good e-learning platform will provide appropriate spaces of both web-based and non-web-based form for students to participate in learning communities for supporting their learning experience and identity (Nortvig et al, 2018).

For ICT adoption, Jayachithra (2020) mentioned that technology is becoming popular and crucial to transform the learning process by enhancing the educators-learners relationship and academic achievements as well as the access of educational curriculum, which conventional education is unable to do so. These technologies included Wi-Fi, remote routers and collaboration tools. Jayachithra (2020) performed an experiment research by choosing 80 Year-2 undergraduate students, divided into e-learning groups and conventional groups, and conducted both pre-test and post-test for both groups. She found out that e-learning group students obtained higher post-test scores mean value (75.77) compared to conventional group (49.67). This shows that e-learning platforms help students to improve their academic performance and achieve academic milestones easily.

2.3.2 Online Programming

As e-learning platforms rise among the higher educational institutions due to the pandemic, undergraduate students that are studying computer science, information system or software engineering-related majors have to learn different kinds of programming languages and code them to become a workable program in their coursework and practical training. To let students improve their coding and development skills, online programming platforms exist for them, regardless of whether the students are novice, moderate or expert in programming.

According to Gandraß et al (2020), online programming platforms can be divided into two which are introductory and professional. Introductory online programming platforms usually face users who lack or have only basic programming fundamentals; therefore, this platform usually only supports few or even one programming language, providing instant visual feedback to lower the barrier for beginners. In this platform, courses usually consist of only basic algorithms and control flow, such as loops and conditionals. In contrast, programming platforms are well-designed for potential software developers that have high ability to code the program well, and there are lots of companies using this type of online programming platform to identify, choose and hire them. Therefore, it usually does not provide visual feedback to the user. Task range in this platform type is too wide (from easy to

hard). Also, professional platforms usually provide competition to other users to gain motivation (Gandraß et al, 2020).

2.3.3 Existing Online Programming Platform

2.3.3.1 W3School

W3School is the largest web online programming platform in the world that allows developers and learners to learn web application development. It is free and does not need an account to access the whole platform, although signup and login of account is provided. W3School consists of various web application programming languages such as HTML, CSS, JavaScript, PHP, SQL and XML, and basic programming languages such as Python, Java, C++, C# and R. In the course and tutorial, it supplies code examples, exercises, quizzes, and tutorial briefing. Once the user finishes the quizzes, it will show feedback to the user immediately. It also lets users interact with the platform by editing the code and showing the output. Forum is included in the platform for registered users to discuss and share opinions.

2.3.3.2 Edabit

Edabit is a free interactive online programming platform that enables users to try challenges for different programming languages. It provides C#, C++, Java, JavaScript, PHP, Python, Ruby and Swift programming languages. The user can choose any programming language and its difficulty, from “Very Easy” to “Expert”, and select their desired challenges to answer. Each challenge consists of a code editor with hidden and self-defined test codes, the test codes will check the codes that user typed is matched with the expected codes. However, there are some restrictions when using these online programming platforms. An account is required to check answers and find solutions from others, it also provides quizzes but only for Python and JavaScript programming languages.

2.3.3.3 BitDegree Learn

BitDegree Learn is a part of its superset online course platform, BitDegree, which mainly focuses on learning programming languages. It provides lots of famous programming language and web application language courses such as HTML, CSS, PHP, JavaScript, SQL, BootStrap, Solidity, JQuery, Git, C++, and Python. In each

programming language course, there are lots of tutorials regarding features, classes, functions etc. Users can view briefings, code samples in the course and experience coding via a prepared code editor and compiler. However, there are no quizzes and peer pressure on this platform.

2.3.3.4 Solo Learn

Solo Learn is also a free and interactive online programming platform that allows users to learn coding skills. It provides approximately 21 programming courses for users to choose and learn, including Swift 4 programming language. In the course, it has many categories from the basics to the advanced topics such as classes and functions. Users can learn while reading materials provided and try to do coding by themselves. It also provides interactive quizzes and exercises in the form of multiple-choice questions and fill in the blanks for users who already have an account. Solo Learn also includes a code playground for users to type any code they want in the code editor, and they also can ask questions or answer questions in the provided forums.

2.3.3.5 CodeAcademy

CodeAcademy has two types of categories to learn computer science related knowledge which are “Languages” and “Subjects”. In the “Languages” category, CodeAcademy provides many well-known programming languages, including Swift programming language, whereas in the “Subjects” category, it covers web design and development, machine learning, game development, mobile development etc. Each programming language subject has one basic course that allows the user to look up materials, write codes in code editor and show output at the same time. However, to access intermediate or advanced courses, the user has to pay money for subscribing to a monthly or annually professional plan. Besides, CodeAcademy also provides peer pressure which are forums and chat, but it does not have a physical chat box on the platform. Instead, it will direct to a Discord link to invite users to join in, the users have to login or register a Discord account to do so.

2.3.3.6 Comparison of Existing Online Programming Platform

Table 2-2: Comparison of 5 researched existing online programming platform

E-learning platform	W3School	Edabit	BitDegree Learn	Solo Learn	CodeAcademy
Interactive	Yes	Yes	Yes	Yes	Yes
Free	Yes	Yes	Yes	Yes	Yes, with plans that cost money
Signup and login function	Yes	Yes	Yes	Yes	Yes
Account required to access the overall platform?	No	Yes, for checking answer and find solution	No	Yes, for quizzes	Yes
Available programming languages and framework	HTML, CSS, JavaScript, SQL, PHP, Python, Java, C++, C#, Bootstrap	C#, C++, Java, JavaScript, PHP, Python, Ruby, Swift	HTML, CSS, PHP, JavaScript, SQL, Bootstrap, Solidity, JQuery, C++, Python	Python, Java, C++, JavaScript, C#, PHP, SQL, HTML, CSS, C, React + Redux, Angular + NestJS, Ruby, Swift, JQuery	HTML, CSS, Python, JavaScript, Java, SQL, Bash/Shell, Ruby, C++, R, C#, PHP, Go, Swift, Kotlin
Tutorial/Instructions/Courses included	Yes	Yes	Yes	Yes	Yes
Code Editor	Yes	Yes	Yes	Yes	Yes

and Code Compiler					
Quizzes, exercises and instant feedback	Yes	JavaScript and Python only	No	Yes	Yes
Peer pressure function	Yes, forum	No	No	Yes, forum	Yes, forum. For chat, a discord link will be provided

Based on the compared existing online programming platform as shown as Table 2-2, Solo Learn is the most complete online programming platform from other compared online programming platform, as it is interactive and free, provide signup and login function, included Swift programming language, tutorial, code editor function, quizzes and instant feedback. It also supports peer pressure function. Although W3School is also almost complete, it lacks the Swift programming language in the platform. Therefore, some characteristics of the existing online programming platform had been included in this project, especially the peer pressure function.

2.4 Methodology Research

This section discussed three software development methodology, with its pros and cons. Then, discussed software development methodology were compared in a table form. A good software development methodology is vital in the software development lifecycle to reduce the unwanted cost and increase the efficiency to produce a high-quality software product.

2.4.1 Waterfall Development

Pressman (2001, cited in Chauhan et al, 2017) defined that the waterfall model is a traditional and sequential software development model that is proceed in linear form, the phases, which are planning, analysis, design, implementation, testing, and etc. flows straightly from start to end, like a waterfall. In the waterfall development model, each phase has to be finished and approved by the project sponsor before

proceeding into the next phase, so it is not possible to have any overlapping phases and it is suitable for requirement-cleared and small projects (Dennis et al, 2015; Sharma, 2016). Chauhan et al (2017) and Sharma (2016) also mentioned that the waterfall model allows the project manager to control and manage software project lifecycle, as documentation writing, review and user approval will be executed in each phase. However, although going back to the previous phase in the waterfall model is possible, it is very difficult to do so (Dennis et al, 2015). Besides, the waterfall model has been blamed due to the highly wastage of cost and time, and it might be unable to deliver the expected requirements properly (Chauhan et al, 2017).

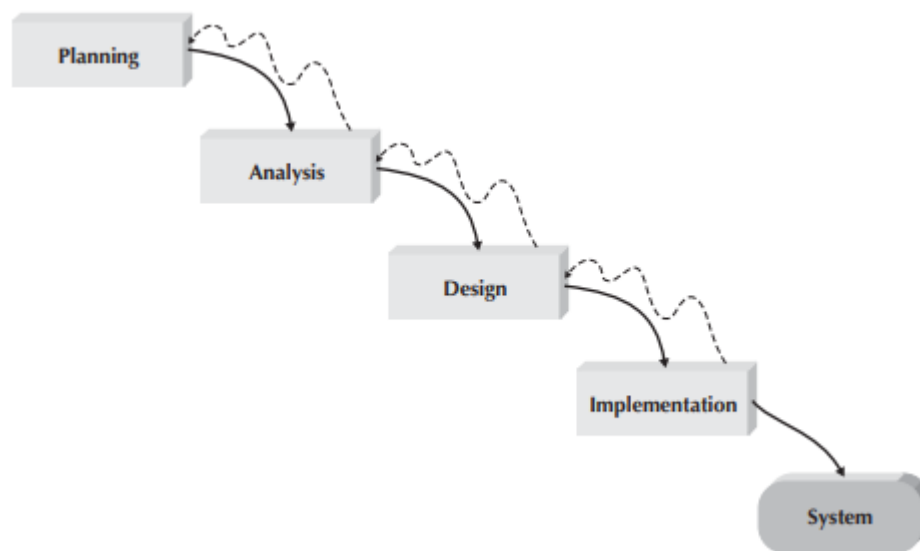


Figure 2-3: Flow of waterfall development (Dennis et al, 2015)

2.4.2 Evolutionary Prototype Development

Prototyping is vital in the software development life cycle, as it creates one or more concepts and ideas to create the whole software, and helps software stakeholders, especially end users to provide comments and feedbacks regarding the user design and flow back to the developers, to make improvement on the system before it become an actual product. (Kurcwald, 2019).

Dennis et al (2015) and Jayasinghe (2020) explained that evolutionary prototyping, or breadboard prototyping is a methodology that builds a small portion of workable prototype and enhances it constantly once feedback is received, to become a fully workable software. After the planning phase is done, the analysis

phase, design phase, and implementation phase will be conducted at the same time and repeatedly as an iteration to produce a system prototype with few features, so that the user is able to test the system prototype and give comments on it to the developers. As the number of cycles increases, more improvement and new functions will be added into the software prototype, until the prototype becomes totally mature and almost satisfied by the user, then it will become a final product.

According to Dennis et al (2015), this software methodology allows developers to produce an interactive system extremely fast for the user, to help them to refine actual requirements as soon as possible. Jayasinghe said that the evolutionary prototyping model helps to gain client satisfaction regarding the system prototype. Developers are also able to stop developing the system if they find out some serious issue after several iterations when producing the system prototype (Khalid, 2018).

Nevertheless, there are some cons by using this methodology. Dennis et al (2015) state that evolutionary prototyping models are not suitable for huge and complex systems, as changes across the iterations may cause the initial concept and design to become poor. Khalid (2018) also mentioned that this methodology may lead to being unable to set a time frame for the software project, as the project itself becomes boundaryless.

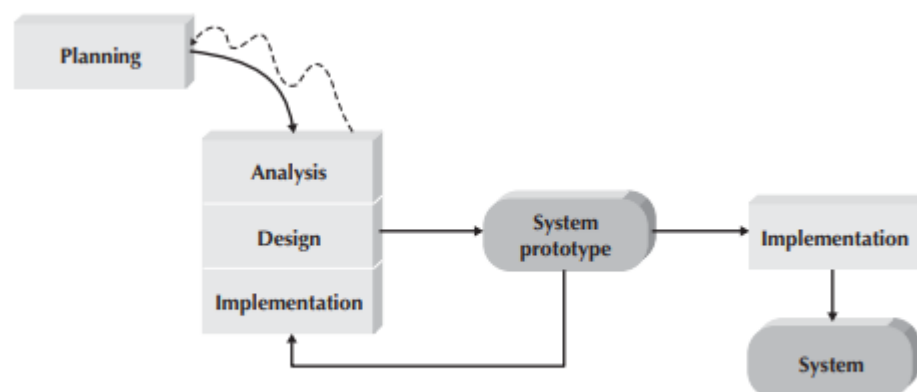


Figure 2-4: Flow of evolutionary prototyping development (Dennis et al, 2015)

2.4.3 Agile Development

Anwer et al (2017) explained the meaning of agile development:

“Provide an iterative and evolutionary development paradigm with more emphasis on changing requirements, customer satisfaction, and team collaboration.”

This means that the developers who use this methodology will develop and deliver the software or systems very quickly, early, frequently and continuously to satisfy the customer, as it is programming centric. Agile development also enables the ability to change requirements in any phase, even though it is in the late phases of software development. This methodology also advocates simplicity, which means avoiding any unnecessary work during the software development lifecycle, and the importance of communication physically in the development team, to make the team members believe in each other (Dennis et al, 2015; Chauhan et al, 2017).

However, Dennis et al (2015) stated that agile methodologies have some major problems, such as require co-location of development team as it is offshored or outsourced, high probability to devolve into prototyping development if the project is not managed properly, and low-quality assurance of the system due to no actual documentation for the software project.

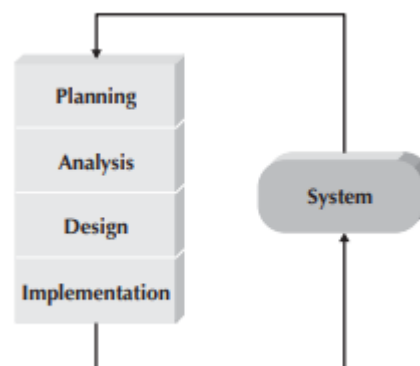


Figure 2-5: Flow of agile development (Dennis et al, 2015)

2.4.4 Comparison of Software Methodology

Table 2-3: Comparison between three development methodologies (Dennis et al, 2015)

Criteria	Waterfall	Evolutionary Prototyping	Agile
Unclear user requirements	Poor	Excellent	Excellent
Unfamiliar technology	Poor	Poor	Poor
Complex systems	Good	Poor	Poor
Reliable systems	Good	Poor	Good
Short time schedule	Poor	Excellent	Excellent
Schedule visibility	Poor	Excellent	Good

Based on the compared development methodologies as listed in Table 2-3, waterfall development methodology is suitable in systems that are complex and moderate-to-high reliability, but not suited in short-period projects or projects with lots of unclear requirements and unfamiliar technology. Evolutionary prototyping development methodology is good in short-period projects and projects with lots of requirements uncertainties due to the prototyping iterations, but it only works in simple systems. Agile development methodology is superb in systems with unclear requirements, high reliability and short time schedule, but it becomes not suitable when the system is complex, or unfamiliar technologies are adopted.

2.5 Conclusion of the Chapter

Due to high performance, more secure, cleaner and readable syntax, Swift programming language has started to become more popular and attractive for developers and students to learn and develop iOS apps. Lots of developers are also starting to make the transition of their iOS and macOS apps from pure Objective-C to hybrid or pure Swift programming language, and lots of universities and colleges start to offer iOS application development courses for students and the public.

Also, as digital products start to “fuse” into our daily lifestyle, it is inevitable to make a transition from conventional methods into online and electronic methods,

including education sectors. E-learning has become a trend from now to conduct teaching and learning in higher education without the restriction of time and venue. Besides, there are lots of students choosing computer-science related as their undergraduate studies. Therefore, online programming platforms have started to be recommended by lots of lecturers for students to practice their coding skills anytime and anywhere, with just an Internet connection.

For researched software development methodologies, evolutionary prototyping has been chosen in this project. This is because this methodology not only helps to save lots of time to complete the system development, as the prototype can be continually to be developed into a full system instead of discard it and redo a new full functional system, but also be able to collect sufficient feedbacks from the user once the user had experienced with the small portion of the system prototype.

CHAPTER 3

METHODOLOGY

The objective of this chapter is to provide an explanation regarding the methodology applied for this project. There are four sections in this methodology chapter. Section 3.1 explained the chosen software development methodology for this project based on the research in the previous chapter. Section 3.2 is to discuss and compare the web hosting service. Section 3.3 is to discuss the tools that were adopted to develop the system, including programming languages, framework, database, server, prototyping tools, and online code editor and compiler. Section 3.4 is to provide the project plan, with attached work breakdown structure and Gantt chart.

3.1 Chosen Software Development Methodology

Based on the research of various software development methodology in the previous chapter, which is Section 2.4, evolutionary prototyping had been chosen to develop the system in this project. In evolutionary prototyping, it can be separated into different phases which are requirements gathering and analysis, system design with many iterations and implementation, testing, and deployment. During the evolutionary prototyping, workable prototypes which are considered as part of the system had been produced and evaluated by users to gain feedback and comments, so that the prototypes can be improved based on the collected feedback.

3.1.1 Requirements Gathering and Analysis

To gather requirements, literature review had been conducted by reviewing journals and conference papers. Therefore, no questionnaire or survey was conducted. After the requirements are collected, the analysis phase has been performed to filter out useful and important requirements, and they have been categorized into both functional and non-functional requirements. Based on the functional and non-functional requirements, use case diagrams and use case descriptions had been designed.

3.1.2 System Design and Implementation

In this phase, the whole system and its related documentation had been designed and implemented by producing workable system prototypes. This phase has three iterations of prototyping which can be categorized into first iteration, second iteration, and final iteration.

3.1.2.1 First Iteration of Prototyping

In the first iteration, the topic and lesson prototype, code editor prototype, graded quiz prototype and exercise prototype had been developed. Each topic prototype contains the briefing and tutorials for the particular topic, hard-coded code examples and the link towards the code editor with predefined codes. For the code editor prototype, Paiza.io had been used as the Swift compiler for users to edit codes and view outputs. Graded quiz prototype consisted of 10 multiple choice questions with 4 options per question, the system recorded the answer selected by the user, and showed to them their result after attempting the graded quiz. Exercise prototype consisted of 7 fill-in-the-blanks non-graded questions for users to practice themselves. Due to the time restrictions, these prototypes had been designed by using basic HTML, CSS and JavaScript programming languages, without using any web application framework.

After the prototypes were done, a mini UAT testing had been conducted for users to experience them to collect any constructive feedback for further enhancement of these prototypes.

Besides, during this iteration, Axure RP had been used to design the overview user interface of the whole system without any interactions, to provide a clear concept of the system flow to the users.

3.1.2.2 Second Iteration of Prototyping

The second iteration had been on designing essential UML diagrams, such as class diagrams, activity diagrams, and entity relationship diagrams. Next, the group chat box system prototype, login prototype, user profile prototype and the user interface for the whole system based on the previous iteration design were fully developed by

using Laravel framework. Authentication and authorization of login were also developed in the login prototype.

Similar to the previous iteration, a mini UAT testing had been performed for these implemented prototypes to gain feedback from the users for further improvements.

3.1.2.3 Final Iteration of Prototyping

The last iteration had been focused on administrative related modules of the system which are to register students accounts, manage students accounts, and make amendment of the topic content by using Laravel framework. After that, all prototypes that developed in these three iterations had been already integrated into a final and full functional system.

3.1.3 Software Testing

In this phase, four types of testing were conducted which are unit testing, integration testing, usability testing and UAT testing. Unit testing was performed to test each function or module, whereas integration testing was used to test the linkage between two or more functions when they are integrated. Later, usability testing was conducted to test the system to ensure it is able to let users use it easily. Lastly, UAT testing was conducted by inviting 10 to 15 students who registered for UECS3263 in the May 2021 trimester to use the final system. During the final UAT testing, feedback was collected to make a finalized enhancement.

3.1.4 Software Deployment

After the testing phase for the final system is done, the final system has been delivered to the user via web hosting services. Users may access the web URL to use the system. Final report and presentation slides were prepared to demonstrate the final system.

3.2 Web Hosting Services

It is meaningless if the developed web application can be used only in localhost instead of applicable in the world wide web. According to Website.com (n.d.), a Web Hosting Service allows individual developers or developers in the same organization

to upload a website with one or more web pages, even though a whole web application onto the web hosting service provider's server, to let users around the world access the website. Mostly, it contains file transfer protocol (FTP) to upload necessary web app files into the server, one or more databases to store data, email account features to send email notification to users, and website manager to manage web domains. Therefore, a list of web hosting services are compared, and the most suitable web hosting service is chosen.

3.2.1 InfinityFree

InfinityFree is a free web hosting service provider sponsored by a web service provider named iFastnet. It provides unlimited storage size and speed to store web application files, and it provides up to 400 MySQL 5.6 databases with the support of PHP 5.4, 5.5, 5.6 and 7.4. However, it does not provide the InnoDB database engine as well as the PHP Mail function, unless the customer spends real money to upgrade into iFastnet premium service.

3.2.2 Hostinger

Hostinger is a paid web hosting service provider and website domain registrar. It provides three monthly paid web hosting plans which are "Single", "Premium", and "Business" that cost \$ 1.39 USD, \$ 2.59 USD, and \$3.99 USD respectively. This monthly paid web hosting plan provides 30 GB, 100 GB, and 200 GB storage respectively. Except "Single", other plans provide unlimited web bandwidth. Nevertheless, these three web hosting plans provide InnoDB database engine and PHP Mail function.

3.2.3 Awardspace

Awardspace is a web hosting service provider that provides both free and paid web hosting plans. Both plans provide a MySQL database with InnoDB database engine and one email account. However, Free plan users are unable to use email service although they have one, as the email service in free plan are only eligible in full domain websites, but free plan users are allowed to register at most 3 subdomains.

3.2.4 Comparison of Web Hosting Service

Table 3-1: Comparison between Three Web Hosting Services

Web Hosting Service	InfinityFree	Hostinger	Awardspace
Free	Yes	No	Yes, with paid plans
Database support	Yes, without InnoDB support	Yes, with InnoDB support	Yes, with InnoDB support
PHPMYAdmin	Yes	Yes	Yes
PHP Mail function	No	Yes	Yes, but unable to use in subdomain for free users
Email account	No	1 for “Single”, 100 for “Premium” and “Business”	1 for free, 1000 for basic paid plan, unlimited for Pro Plus and Max Pack plan
Storage Size	Unlimited	30 GB for “Single”, 100 GB for “Premium”, 200 GB for “Business”	1 GB for free, unlimited for paid
Bandwidth Size	Unlimited	100 GB for “Single”, unlimited for “Premium” and “Business”	5 GB for free, unlimited for paid
Ad-Free	Yes	Yes	Yes

Based on the Table 3-1, Awardspace had been chosen as a web hosting service. Although it only provides 1 GB storage space and 5 GB bandwidth, it provides a free InnoDB database engine. In contrast, InfinityFree provides unlimited storage space and bandwidth but no free InnoDB database engine.

3.3 Development and Prototyping Tools

It is necessary to have well-planned tools to execute the project well. In this section, planned programming languages, frameworks, server, database system, code editor and prototyping software had been listed out to build the system properly.

3.3.1 Programming Language

3.3.1.1 Web-based Programming Language

Since this project was developing a web application, five crucial web-based programming languages have been used, which are HTML, CSS, JavaScript, PHP and SQL.

HTML means “Hypertext Markup Language”, it sets the whole structure of the webpage, similar to a backbone of the web page. CSS means “Cascade Style Sheet” that decorate the webpage and become more vibrant and user-friendly with various colours, font sizes, and font types. JavaScript is a scripting language that is mostly used for client-side and some necessary interaction. In contrast, PHP is a server-side programming language that communicates with the server. Last but not least, SQL means Structured Query Language that is usually used in the database for creating tables, insert data, update data, retrieve data, and retrieve data.

3.3.1.2 Swift Programming Language

Swift programming language has been used to create code examples for briefing, exercises, quizzes, and code samples in the embedded code compiler. The version of the Swift programming language that has been used is Swift 5.3.

3.3.2 Framework

3.3.2.1 Laravel

Laravel is a web application framework that uses model-view-controller architecture. PHP programming language is used to develop Laravel-based web applications.

3.3.2.2 React

React is an open-source web application framework that was created and developed by Facebook Inc. It is usually used to develop the front-end of the web application

with various UI components and interactions. React can be installed into the Laravel framework, and the programming language used for React is JavaScript.

3.3.3 Server and Database System

3.3.3.1 MySQL

MySQL is a relational database management system that is popular for web developers to use. It is able to add, edit, view and delete data from the table or the table itself. Apart from that, it can also define relationships, primary keys and foreign keys. InnoDB was used as the database engine to store foreign key and constraint relationships.

3.3.3.2 WampServer

WampServer is a software that allows the local computer to become a localhost server for web applications. It provides MySQL and MariaDB databases, as well as the support of PHP environments. It only supports Microsoft Windows operating systems.

3.3.4 Integrated Development Environment

3.3.4.1 Visual Studio Code

Visual Studio Code is a free, open-source and lightweight source-code editor that is developed by Microsoft. It is the default source code editor for Laravel framework and React framework.

3.3.5 Prototyping Tools

3.3.5.1 Axure RP 9

Axure RP is a powerful prototyping tool to make either web or mobile prototypes. It can be done in either a low fidelity prototype with only the UI design, or a high-fidelity prototype with interactions in it. Some prototypes were produced by using Axure RP due to the time limitations, especially the initial UI design.

3.3.6 Online Code Editor

3.3.6.1 Paiza.io

Paiza.io is an online code compiler and code editor to let users do coding, run programs and watch output. It supports various types of programming languages

including Swift. It also supports web embedding to let users embed the projects into the web application.

3.4 Project Plan

A project should have a project plan to ensure each task is assigned well with the stipulated time, so that there is no wastage of time and resources during the execution of the project.

3.4.1 Work Breakdown Structure and Gantt Chart

The work breakdown structure and Gantt chart had been attached as in Appendix A and Appendix B respectively.

3.5 Conclusion of the Chapter

In short, every phase including iterations from the chosen software development methodology for this project had been defined. Three web hosting services were compared and the most suitable web hosting service was chosen to deploy the system. Besides, developing and prototyping tools which had been adopted for this project were listed out. Project plan has been designed and planned to ensure that the work is always on track. Last but not least, preliminary user interfaces were sketched and designed for further system implementation purposes.

CHAPTER 4

PROJECT SPECIFICATION

The objective of this chapter is to provide details of specifications in this project. There are four sections in this project specification chapter. Section 4.1 is to explain the short description of requirements gathering. Section 4.2 is to list out all the requirements specifications, including functional requirements, non-functional requirements, and assumptions. Section 4.3 is to state out the use case diagram. Section 4.4 is to list out all use case descriptions based on the use case diagram. Section 4.5 is to provide preliminary user interface design for the following iterations.

4.1 Introduction

As stated in Chapter 3 Section 3.1.1, the requirements were gathered by conducting research on literature review and studies. Based on the research paper from Gandraß et al (2020), an online programming platform should consist of questions and problem solving, visual feedback, on-boarding function, guidelines, and peer-pressure to gain motivation for students. For lecturer or admin, an online programming platform should provide a function for modification of course content and student progress tracing function for them.

4.2 Requirements Specifications

4.2.1 Functional Requirements

1. The system should allow the user to login the system.
2. The system should allow the user to reset password.
3. The system should allow the user to logout.
4. The system should allow the user to read and follow Swift programming materials.
5. The system should allow the user to edit codes and produce output in the embedded code compiler.
6. The system should provide non-graded exercises for users as a practice.
7. The system should provide graded quizzes after the user completes each topic.

8. The system should allow the user to retake the graded quizzes with unlimited trials.
9. The system should allow the user to view the results once the user has completed the graded quizzes.
10. The system should allow the user to view his/her basic statistics for overall performance.
11. The system should allow the user to view his/her graded quiz attempt history.
12. The system should provide a chat box for users to communicate with other users.
13. The system should allow the administrator to register the user account.
14. The system should allow the administrator to delete the user account.
15. The system should allow the administrator to monitor the student's progress.
16. The system should allow the administrator to modify the exercises, quizzes and tutorial briefings.

4.2.2 Non-functional Requirements

1. The system should provide an interface to the user to reset password when the user login his/her account for the first time.
2. The system should obtain the latest score of the user when the user attempts the same quizzes more than once.
3. The system should allows the administrator to manually delete any chat data.
4. The system should hash the password to prevent anyone except the user itself from retrieving the password.
5. The system should display the online users in the chat box list.

4.2.3 Assumptions

1. No time limit for each graded quiz. Therefore, students can ease themselves to take the quiz without any time restriction panics.
2. The administrator is the lecturer who handles the iOS Application Development course.

4.3 Use Case Diagram

The use case diagram had been attached as in Appendix D.

4.4 Use Case Description

Table 4-1: Use Case Description of Read Materials

Use Case Name: READ MATERIALS	ID: 1	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: <ol style="list-style-type: none">1. Student – He/She will read the provided materials and briefings of Swift Programming Language in the platform.		
Brief Description: READ MATERIALS use case describe how the student uses the platform to read Swift Programming Language tutorial briefings and materials.		
Trigger: Students want to read Swift Programming Language tutorial briefings and materials before taking exercises, quizzes or code changing.		
Relationships: Association: Student Include: N/A Extend: N/A Generalization: N/A		
Normal Flow of Events: <ol style="list-style-type: none">1. The student wants to read Swift Programming Language tutorial briefings and materials before taking exercises, quizzes or code changing.2. The students will choose any one of the topics from the topic list.3. The students will read all the materials from that particular topic.		

Table 4-2: Use Case Description of Edit Codes in Code Editor

Use Case Name: EDIT CODES IN CODE EDITOR	ID: 2	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
<p>Stakeholders and Interests:</p> <ol style="list-style-type: none"> 1. He/She will edit Swift codes in the provided Swift code editor and compiler in the platform. 		
<p>Brief Description: EDIT CODES IN CODE COMPILER use case describes how the student edits Swift codes in the provided Swift code editor and compiler in the platform.</p>		
<p>Trigger: Students want to edit Swift code during the reading materials of that topic, or edit their desired Swift code in the provided playground.</p>		
<p>Relationships:</p> <p>Association: Student</p> <p>Include: N/A</p> <p>Extend: Edit in The Topic, Edit in The Code Playground</p> <p>Generalization: N/A</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The student wants to edit Swift code during the reading materials of that topic, or edit their desired Swift code in the provided playground. 2. The student will go to the online code editor from the topic. Continue to E2: Edit in The Topic. 3. The student will go to the code playground. Continue to E3: Edit in The Code Playground. 		
<p>Alternate/Exceptional Flows:</p> <p>E2: Edit in the topic</p> <ol style="list-style-type: none"> 1. The student will modify codes that are pre-defined by the platform for that particular topic. 2. The student will run the codes and view the output and the status (pass or fail). <p>E3: Edit in the code playground</p> <ol style="list-style-type: none"> 1. The student will type their desired Swift codes. 		

2. The student will run the codes and view the output.

Table 4-3: Use Case Description of Do Non-Graded Exercise

Use Case Name: DO NON-GRADED EXERCISES	ID: 3	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: 1. Student – He/She will do non-graded exercises as a trial.		
Brief Description: DO NON-GRADED EXERCISES use case describes how the student does non-graded exercises provided by the platform as a practice.		
Trigger: The student wants to do non-graded exercises as a practice or enhance his/her own understanding of Swift Programming Language.		
Relationships: Association: N/A Include: N/A Extend: N/A Generalization: N/A		
Normal Flow of Events: 1. The student wants to do non-graded exercise as a practice or enhance his/her own understanding of Swift Programming Language. 2. The student does non-graded exercise by filling in the blanks. 3. The student checks the solution with the answer.		

Table 4-4: Use Case Description of Take Graded Quizzes

Use Case Name: TAKE GRADED QUIZZES	ID: 4	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
<p>Stakeholders and Interests:</p> <p>1. Student – He/She will take graded quizzes after completing a topic.</p>		
<p>Brief Description: TAKE GRADED QUIZZES use case describes how the student takes graded quizzes after he/she completes a topic.</p>		
<p>Trigger: Student wants to take graded quizzes after he/she completes a topic.</p>		
<p>Relationships:</p> <p>Association: Student</p> <p>Include: N/A</p> <p>Extend: Retake Graded Quizzes</p> <p>Generalization: N/A</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The student wants to take graded quizzes after he/she completes a topic. 2. The student answers 10 multiple choice questions with 4 options per question. 3. The system stores the student's selected answer for each question. 4. After the student answers all the questions, the system displays the number of correct answers and its percentage. 5. The student may re-attempt the same graded quiz. Continue to E5: Retake Graded Quizzes 		
<p>Alternate/Exceptional Flows:</p> <p>E5: Retake Graded Quizzes</p> <ol style="list-style-type: none"> 1. If the student wants to try the same graded quiz again, he/she may perform it with the normal flow of 1 to 4. 		

Table 4-5: Use Case Description of View Profile

Use Case Name: VIEW PROFILE	ID: 5	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
<p>Stakeholders and Interests:</p> <ol style="list-style-type: none"> 1. Student – He/She will view his/her own account profile. 		
<p>Brief Description: VIEW PROFILE use case describes how the student views his/her own account profile.</p>		
<p>Trigger: Student wants to view his/her own account profile.</p>		
<p>Relationships:</p> <p>Association: Student</p> <p>Include: View Basic Statistics of Performance</p> <p>Extend: View Quiz Attempt History</p> <p>Generalization: N/A</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The student goes to his/her user profile. 2. The student can view his/her username and student ID. 3. The student can view his/her performance statistics. Continue to S3: View Basic Statistics of Performance. 4. The student can view his/her graded quiz attempt history. Continue to E4: View Quiz Attempt History. 		
<p>Sub Flows:</p> <p>S3: View Basic Statistics of Performance.</p> <p>3a. The student can view his/her percentage of total correct answers over the total answers of all graded quizzes.</p> <p>3b. The student can view his/her percentage of total correct answers over the total answers of each graded quiz.</p>		
<p>Alternate/Exceptional Flows:</p> <p>E4: View Quiz Attempt History.</p> <ol style="list-style-type: none"> 1. If the students attempted more than once for a graded quiz, he/she can view 		

each attempt history for that particular quiz.

Table 4-6: Use Case Description of Chat in Chat Box

Use Case Name: CHAT IN CHAT BOX	ID: 6	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
Stakeholders and Interests: <ol style="list-style-type: none"> 1. Student – He/She will chat with other students in the chat group. 		
Brief Description: CHAT IN CHAT BOX use case describes how a student chats with other students in the chat group, for discussion and expression of feelings.		
Trigger: The student wants to chat with other students in the chat box.		
Relationships: Association: Students Include: N/A Extend: N/A Generalization: N/A		
Normal Flow of Events: <ol style="list-style-type: none"> 1. The student wants to chat with other users in the chat box. 2. The student types his text in the chat box. 3. The student sends the chat in the chat box. 4. The student views other user’s conversation in the same chat box, and the online users list. 		

Table 4-7: Use Case Description of Login the System

Use Case Name: LOGIN THE SYSTEM	ID: 7	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
<p>Stakeholders and Interests:</p> <ol style="list-style-type: none"> 1. Student – He/she will login to the system to utilize the functions provided by the platform. 		
<p>Brief Description: LOGIN THE SYSTEM use case will describe how the student login to the system to utilize the functions provided by the platform.</p>		
<p>Trigger: The student wants to login to the system to utilize the functions provided by the platform.</p>		
<p>Relationships:</p> <p>Association: Student</p> <p>Include: N/A</p> <p>Extend: Reset Password</p> <p>Generalization: N/A</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The student wants to login to the system to utilize the functions provided by the platform. 2. The student input his/her student ID as the username. 3. The student input his/her password (own reset password or random generated password provided by the administrator). 4. The student login into the system. 5. The student reset his/her password. Continue to E5: Reset Password 		
<p>Alternate/Exceptional Flows:</p> <p>E5: Reset Password</p> <ol style="list-style-type: none"> 1. If the student forgot his/her password, or the student is having his/her first-time login, the student has to input his/her new password and confirmation password. 2. The student accepts his/her reset password. 		

Table 4-8: Use Case Description of Reset Password

Use Case Name: RESET PASSWORD	ID: 7.1	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
<p>Stakeholders and Interests:</p> <ol style="list-style-type: none"> 1. Student – He/she will reset his/her password when he/she forgot his/her password, or the student is having his/her first-time login 		
<p>Brief Description: RESET PASSWORD use case will describe how the student resets his/her password when he/she forgot his/her password, or the student is having his/her first-time login.</p>		
<p>Trigger: The student wants to reset his/her password when he/she forgot his/her password, or the student is having his/her first-time login.</p>		
<p>Relationships:</p> <p>Association: Student</p> <p>Include: N/A</p> <p>Extend: N/A</p> <p>Generalization: N/A</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The student wants to reset his/her password when he/she forgot his/her password, or the student is having his/her first-time login. 2. The student input his/her new password and confirmed password. 3. The student accepts his/her reset password. 		

Table 4-9: Use Case Description of Logout the System

Use Case Name: LOGOUT THE SYSTEM	ID: 8	Importance Level: High
Primary Actor: Student	Use Case Type: Detail, Essential	
<p>Stakeholders and Interests:</p> <ol style="list-style-type: none"> 1. Student – He/she will logout the system after they use the platform. 		
<p>Brief Description: LOGOUT THE SYSTEM use case describes how the student logs out of the system after they use the platform.</p>		
<p>Trigger: The student wants to logout the system after they use the platform.</p>		
<p>Relationships:</p> <p>Association: Student</p> <p>Include: N/A</p> <p>Extend: N/A</p> <p>Generalization: N/A</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The student wants to logout the system after they use the platform. 2. The student logs out of the system. 		

Table 4-10: Use Case Description of Manage Student Account

Use Case Name: MANAGE STUDENT ACCOUNT	ID: 9	Importance Level: High
Primary Actor: Administrator	Use Case Type: Detail, Essential	
Stakeholders and Interests: 1. Administrator – He/She will manage all the existing student accounts.		
Brief Description: MANAGE STUDENT ACCOUNT use case describes how the administrator manages all the existing student accounts.		
Trigger: Administrator wants to manage all the existing student accounts.		
Relationships: Association: Administrator Include: Monitor Student Progress Extend: Delete Student Account, Register Student Account Generalization: N/A		
Normal Flow of Events: 1. Administrator wants to manage all the existing student accounts. 2. Administrator login with his/her administrator account. 3. Administrators may monitor the student performance. Continue to S3: Monitor Student Progress. 4. Administrators may delete an existing student account. Continue to E4: Delete Student Account. 5. Administrators may register a new student account. Continue to E5: Register Student Account.		
Sub Flow: S3: Monitor Student Progress. 3a. Administrator will search the student account by inputting the student ID. 3b. Administrator will view the student's total correct answer over the total answer of all graded quizzes. 3c. Administrator will view the student's total correct answer over the total answer of each graded quiz. 3d. Administrator will view the student's attempt history of the same graded quiz,		

if the student had tried more than once for the same graded quiz.

Alternate/Exceptional Flows:

E4: Delete Student Account

1. If the student has graduated or withdrawn from the studies, the administrator will delete the student's existing account.

E5: Register Student Account

1. If the student has registered the UECS3263 iOS Application Development subject, the administrator will get a student list from the Faculty General Officer of LKC FES.
2. The administrator will key in the student ID and the student's name manually.
3. The system will auto generate a random password for the student account.
4. The system will store the student account into the database.

Table 4-11: Use Case Description of Modify Lesson Content

Use Case Name: MODIFY LESSON CONTENT	ID: 10	Importance Level: High
Primary Actor: Administrator	Use Case Type: Detail, Essential	
<p>Stakeholders and Interests:</p> <ol style="list-style-type: none"> 1. Administrator – He/She will modify the lesson content to the latest version based on the Swift Documentation. 		
<p>Brief Description: MODIFY LESSON CONTENT use case describes how the administrator modifies the lesson content to make sure the content fits the latest version of Swift Documentation if necessary.</p>		
<p>Trigger: Administrator wants to modify the lesson content.</p>		
<p>Relationships:</p> <p>Association: Administrator</p> <p>Include: Add/Edit/Delete Exercises, Add/Edit/Delete Quizzes, Add/Edit/Delete Materials</p> <p>Extend: N/A</p> <p>Generalization: N/A</p>		
<p>Normal Flow of Events:</p> <ol style="list-style-type: none"> 1. The administrator wants to modify the lesson content. 2. The administrator can go to any of the topic materials for modification. Continue to S2: Add/Edit/Delete Materials. 3. The administrator can go to any of the exercises for modification. Continue to S3: Add/Edit/Delete Exercises. 4. The administrator can go to any of the graded quizzes for modification. Continue to S4: Add/Edit/Delete Quizzes. 		
<p>Sub Flow:</p> <p>S2: Add/Edit/Delete Materials</p> <p>2a. The administrator can add new topic materials, or;</p> <p>2b. The administrator can edit existing topic materials, or;</p>		

2c. The administrator can delete existing topic materials.

S3: Add/Edit/Delete Exercises

3a. The administrator can add new exercises, or;

3b. The administrator can edit existing exercises, or;

3c. The administrator can delete existing exercises.

S4: Add/Edit/Delete Quizzes

4a. The administrator can add new quizzes, or;

4b. The administrator can edit existing quizzes, or;

4c. The administrator can delete existing quizzes.

4.5 Preliminary User Interface Design

During the first iteration of prototyping, all essential preliminary user interface designs had been done by using Axure RP 9 Prototyping software, as shown in Figure 4-1 to Figure 4-21. Flow of the initial user interface design had been attached as in Appendix C.

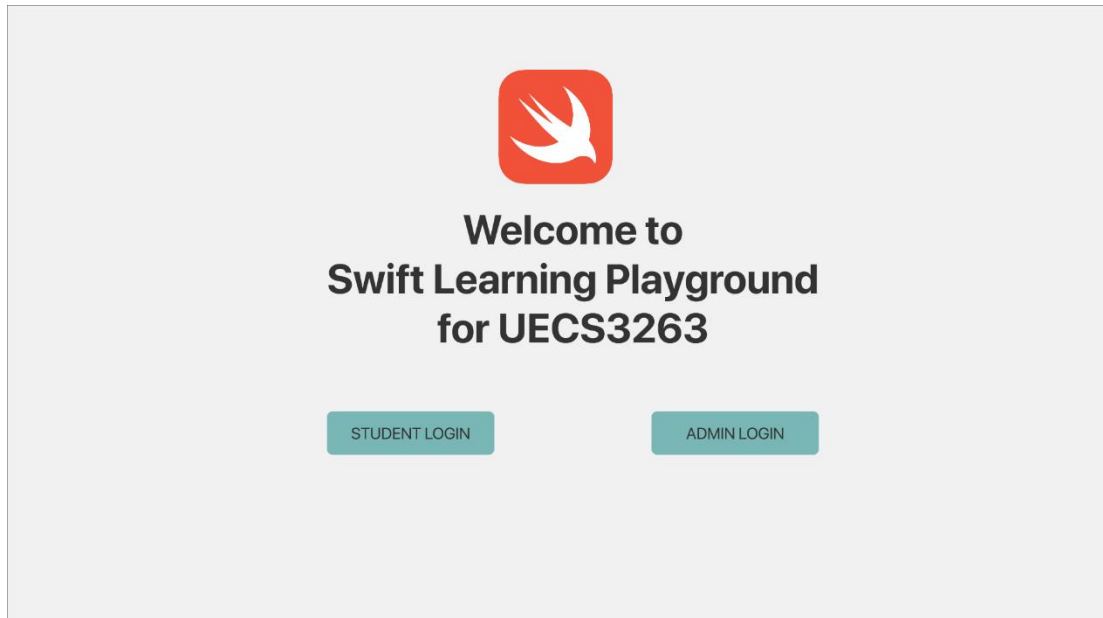


Figure 4-1: Welcome Page

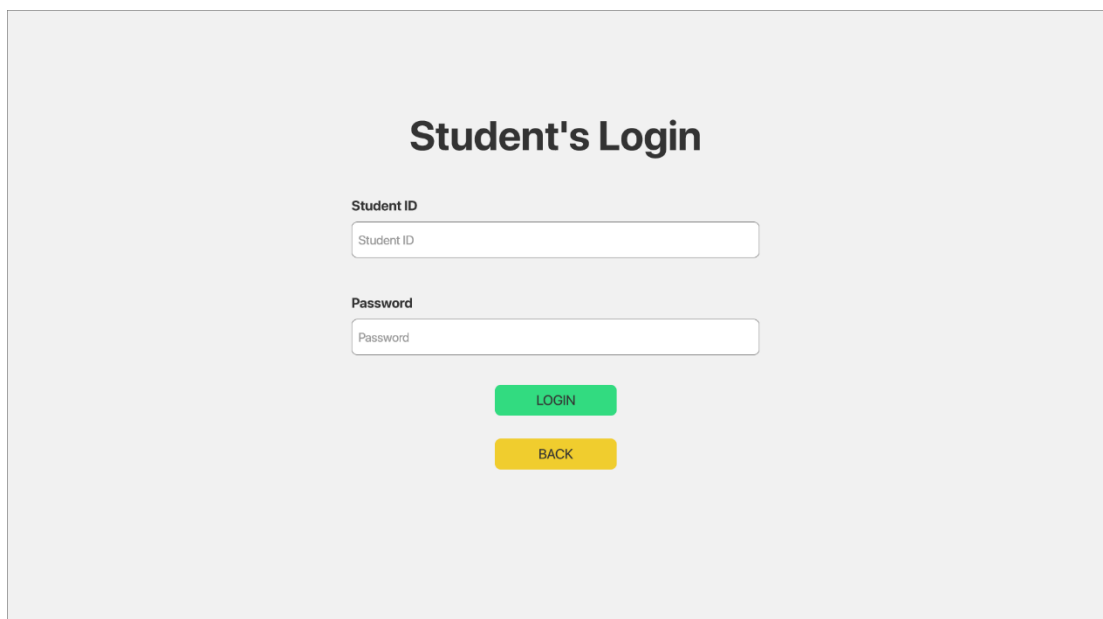
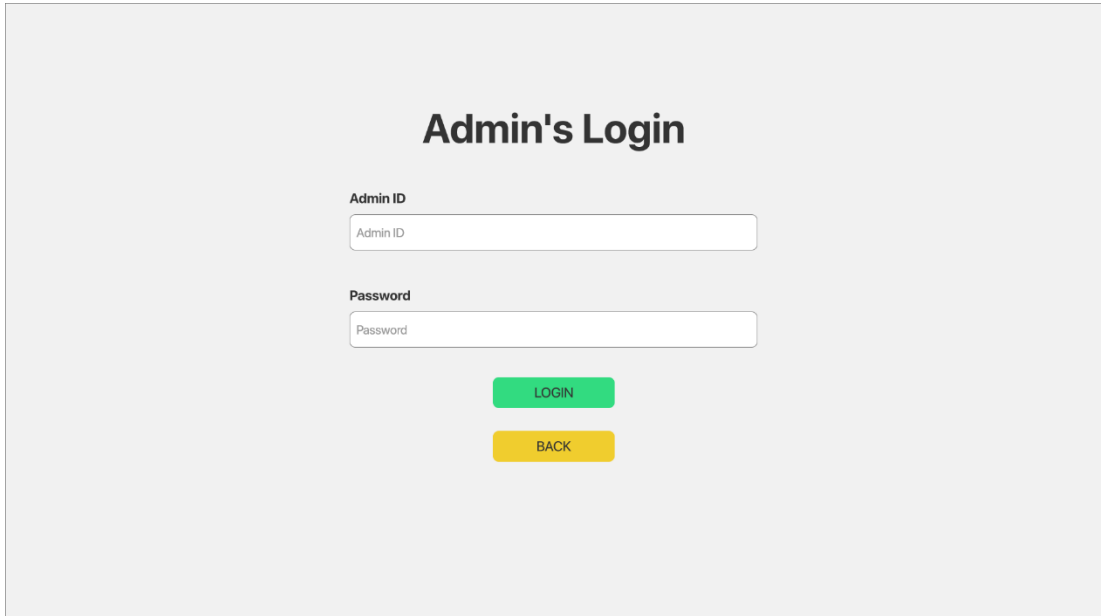


Figure 4-2: Student Login Page



Admin's Login

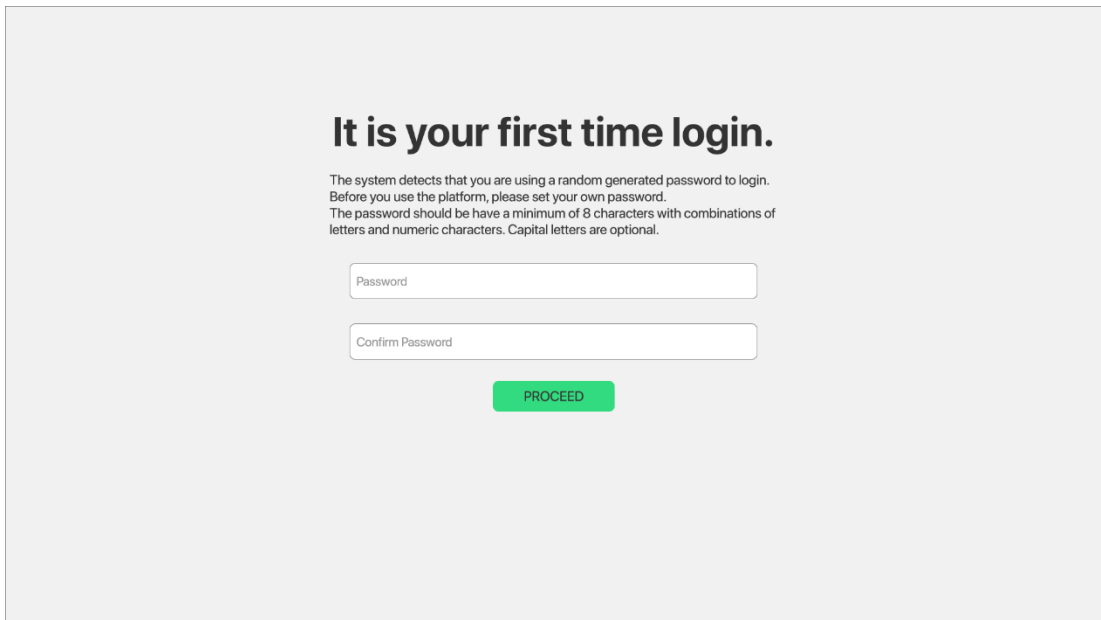
Admin ID

Password

LOGIN

BACK

Figure 4-3: Admin Login Page



It is your first time login.

The system detects that you are using a random generated password to login. Before you use the platform, please set your own password. The password should be have a minimum of 8 characters with combinations of letters and numeric characters. Capital letters are optional.

PROCEED

Figure 4-4: Change Password Page for First Time Login

Reset Password Step 1

Enter your UTAR student email. Then, key in the six digit passcode.

Figure 4-5: Reset Password First Page

Reset Password Step 2

Now, please set your new password.
The password should be have a minimum of 8 characters with combinations of letters and numeric characters. Capital letters are optional.

Figure 4-6: Reset Password Second Page

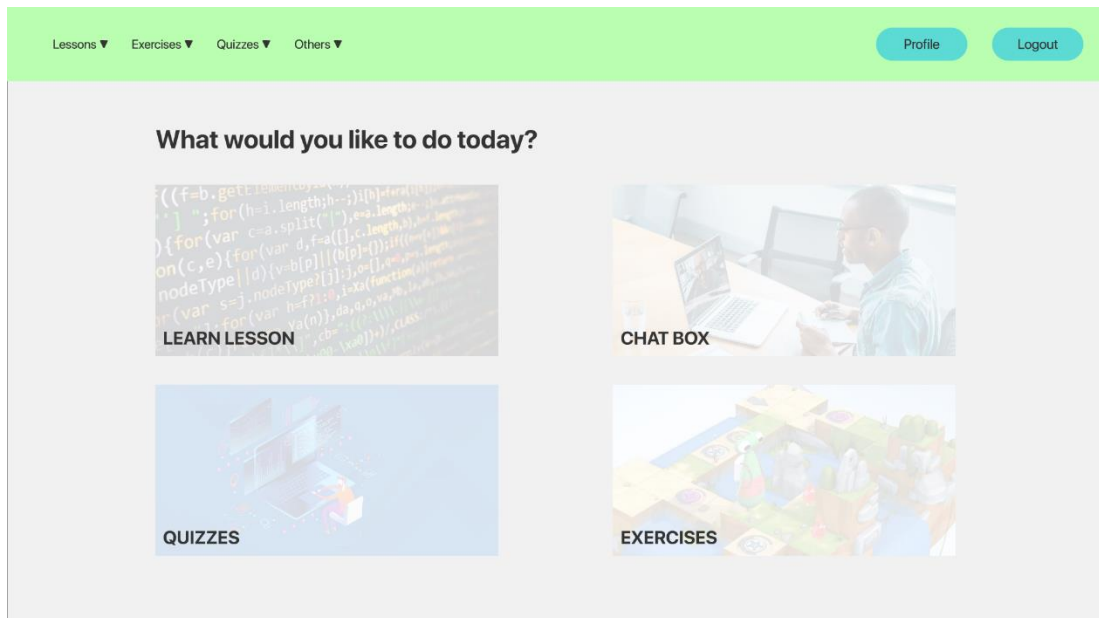


Figure 4-7: Home Page

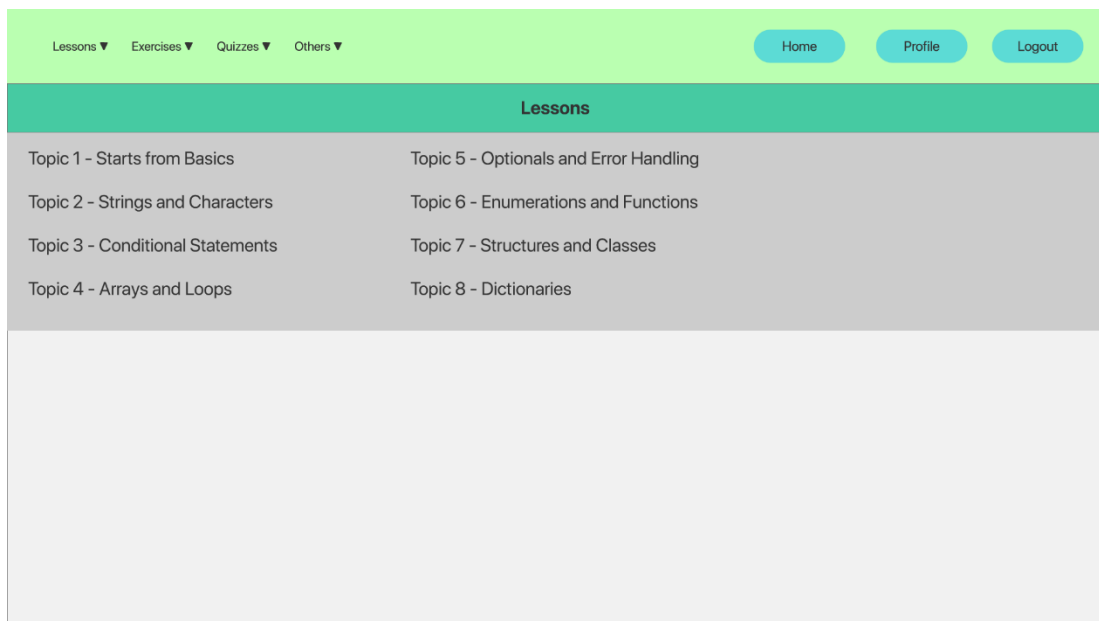


Figure 4-8: Mega Menu and Navigation Bar

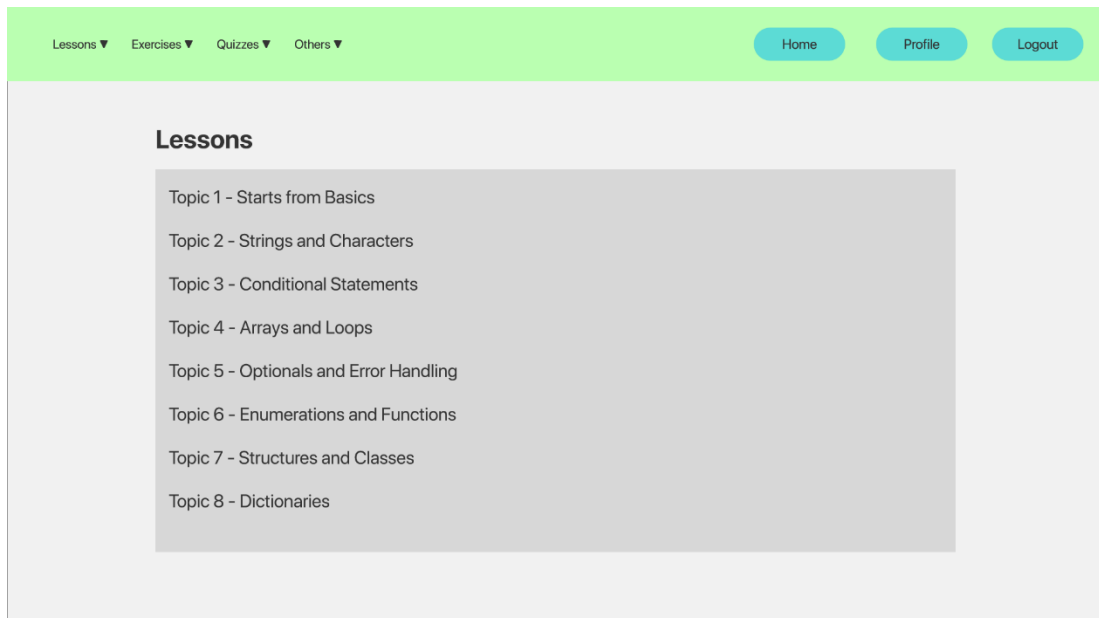


Figure 4-9: Listing of Content (Eg: Lessons)

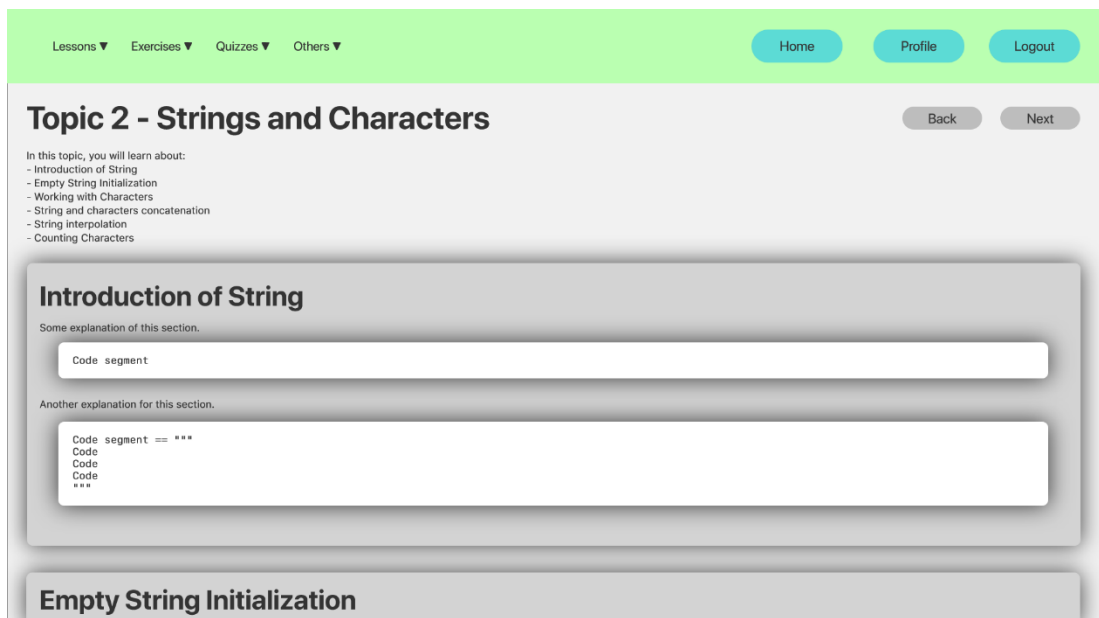


Figure 4-10: Lesson Content Page

Lessons ▾ Exercises ▾ Quizzes ▾ Others ▾ [Home](#) [Profile](#) [Logout](#)

Exercise 1

This exercise consists of 10 questions. Please read the instructions carefully and fill in the blanks with the required words.

Question 1

Determine the constant and variable based on the code segment below.

```
 numberOfDays = 7  
 myAge = 22  
myAge = 23
```

[Submit](#) [Check Answer](#)

Question 2

Determine the constant and variable based on the code segment below.

```
 numberOfDays = 7  
 myAge = 22  
myAge = 23
```

Figure 4-11: Exercise Page

Lessons ▾ Exercises ▾ Quizzes ▾ Others ▾ [Home](#) [Profile](#) [Logout](#)

Quiz 1

This quiz consists of 10 multiple choice questions. You will get one point for each correct answer, but don't worry, graded quizzes have no time limit. After you had completed all the questions, you will see your total score for this result, you may also check it in the profile.

Question 1

Based on the code below, why is this code have a compile error?

```
let someVariable = 5;  
someVariable = 6;
```

Option 1 Option 2
 Option 3 Option 4

[Next](#)

Figure 4-12: Graded Quiz Page

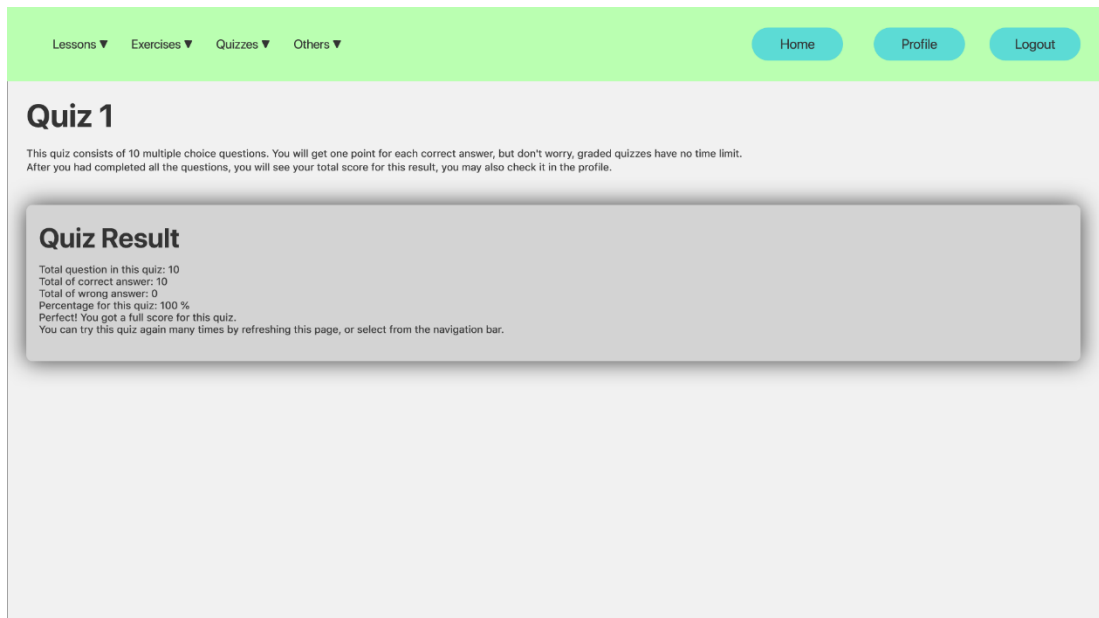


Figure 4-13: Quiz Result Page after completing all Questions

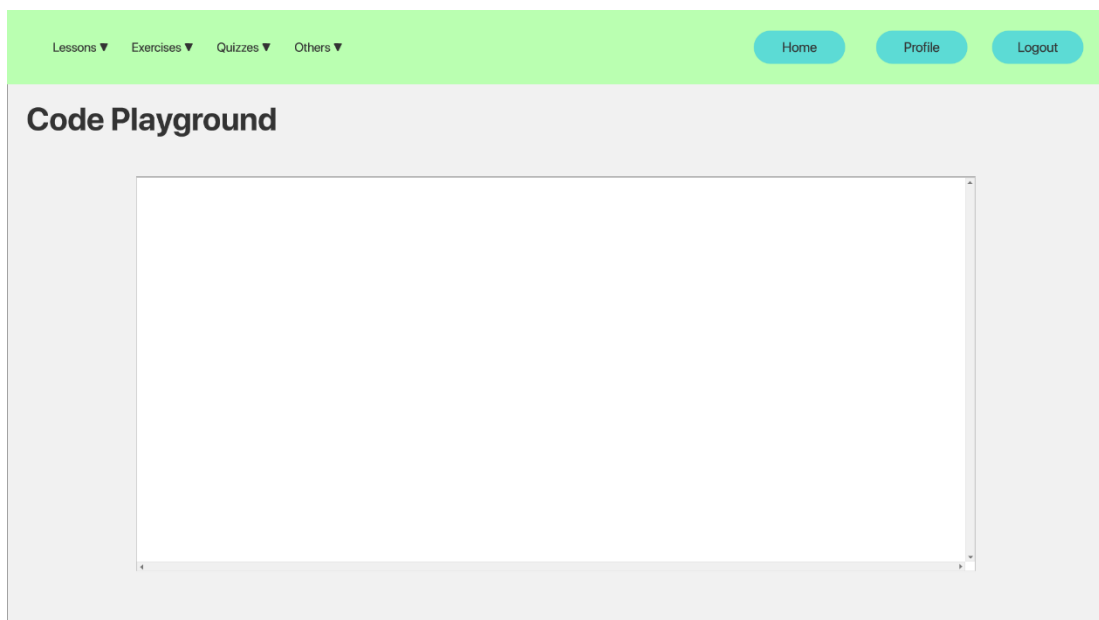


Figure 4-14: Code Editor Page

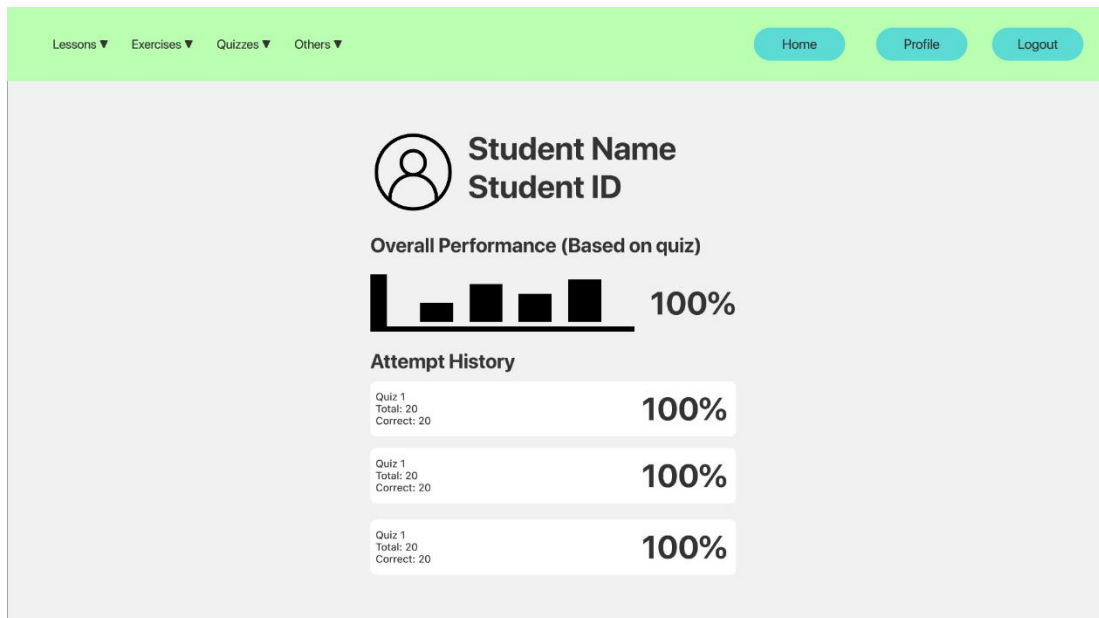


Figure 4-15: Student Profile Page

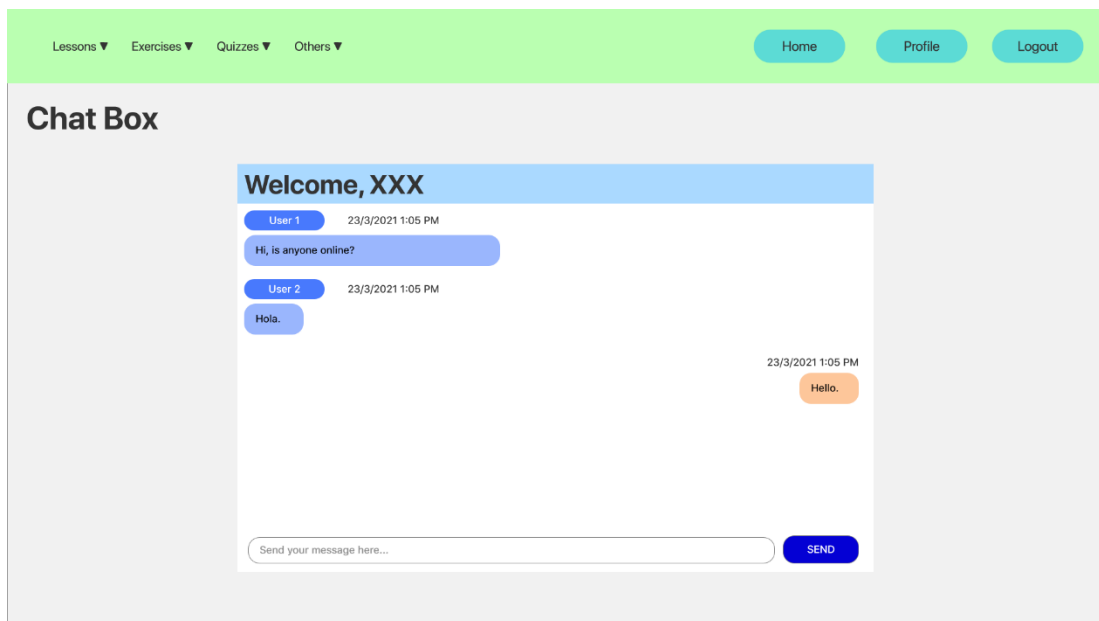


Figure 4-16: Chat Box Page

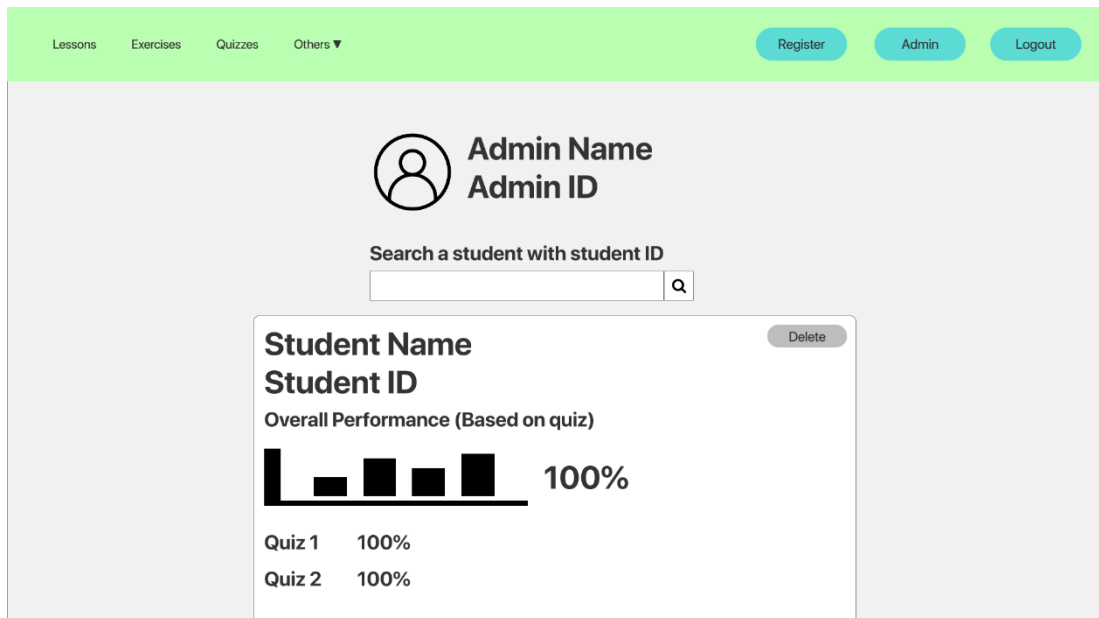


Figure 4-17: Admin Profile

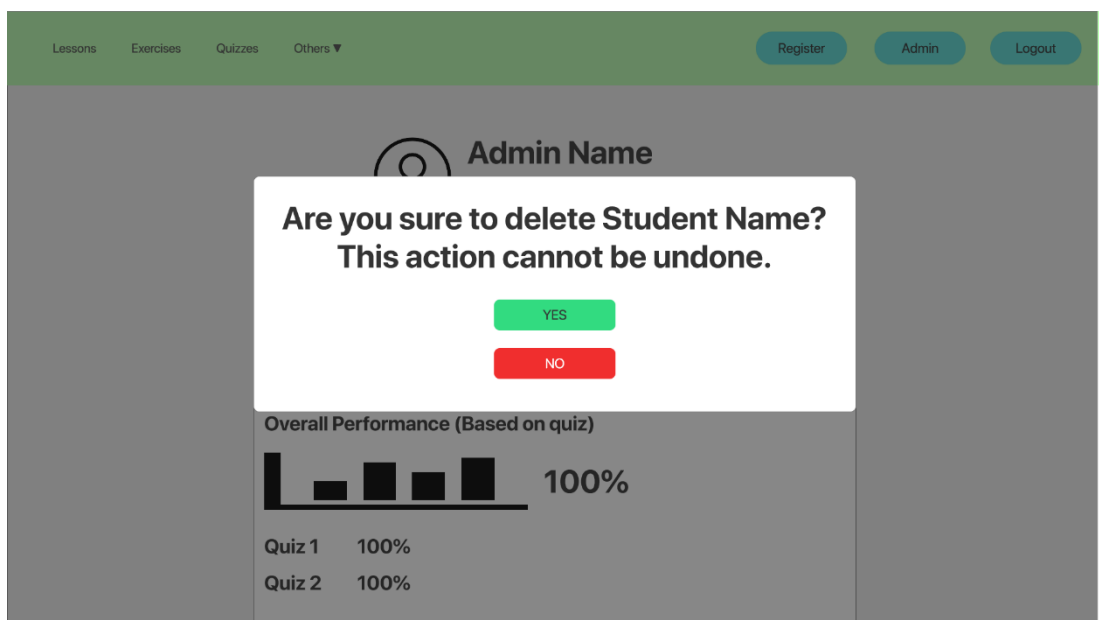


Figure 4-18: Delete Student Account Modal View Page

Lessons Exercises Quizzes Others ▼ Register Admin Logout

Register a Student

Student's ID (Without "UEB")
Student's ID

Student's Name
Student's Name

Student's UTAR Email
Student's UTAR Email

A random password will be generated automatically. Students will be required to update their password when they first time login.

REGISTER

Figure 4-19: Register Student Page

Lessons Exercises Quizzes Others ▼ Register Admin Logout

Lessons

Add

Topic 1 - Starts from Basics	Modify	Delete
Topic 2 - Strings and Characters	Modify	Delete
Topic 3 - Conditional Statements	Modify	Delete
Topic 4 - Arrays and Loops	Modify	Delete
Topic 5 - Optionals and Error Handling	Modify	Delete
Topic 6 - Enumerations and Functions	Modify	Delete
Topic 7 - Structures and Classes	Modify	Delete
Topic 8 - Dictionaries	Modify	Delete

Figure 4-20: Lesson Content List with Add, Modify and Delete Buttons

The image shows a web interface for editing course content. At the top, there is a green navigation bar with links for 'Lessons', 'Exercises', 'Quizzes', and 'Others'. To the right of this bar are three buttons: 'Register', 'Admin', and 'Logout'. Below the navigation bar is a grey header for the 'Modify Lesson' page, which includes a 'Submit' button on the right. The main content area is a white box containing a 'List of Sections' field, a 'Section 1' form with a 'Section Name' input and 'Section Text' area, and a 'Code Example (Optional)' form with a 'Section Text' area.

Figure 4-21: Course Content Editor

4.6 Conclusion of the Chapter

In a nutshell, the method to gather requirements has been defined which is conducting literature review and finding requirements from research papers. 16 functional requirements, 5 non-functional requirements and 2 assumptions are defined. A use case diagram has been designed and 11 use case descriptions from the use case diagram are stated properly. Last but not least, preliminary user interfaces were sketched and designed for further system implementation purposes.

CHAPTER 5

SYSTEM DESIGN

The purpose of this chapter is to describe the overall web application system design. Section 5.1 is to describe the system architecture design of the web application. Section 5.2 is to provide the designed UML diagrams. Section 5.3 is to explain the database design and its database dictionary. Section 5.4 is to provide actual developed user interfaces.

5.1 System Architecture Design

Two types of software architecture design existed in the system, which are Model-View-Controller (MVC) Architecture and Client-Server Architecture.

As stated in Section 1.4, the system was fully developed with Laravel MVC architecture, in which the system can be split into three parts which are Model, View, and Controller. The model connects to the MySQL database that contains numerous tables and their columns, including primary key and foreign key. The view lets users view the data from the database and provide input that needs to be sent into the database. The controller is a medium to fetch data from modal to the view or handles requests from view to the modal. However, the front-end framework in the view is unable to get data from the database and send requests to the controller to handle them. Therefore, application programming interfaces (APIs) are used by calling them to connect the front-end framework with the controller.

Besides, since the system had been deployed into the web hosting services, therefore Client-Server Architecture had also been used in the system. Sarangam (2020) stated that Client-Server Architecture is an architecture design that allows users from client-side to access the web resources from the server through the Internet. On the client-side, users use a browser from a device to access the Internet by entering a URL in the address bar. If the URL is existing, the server which connects to the Internet domain will fetch out all necessary resources to the client-

side or receive necessary input from the client-side. Figure 5-1 shows the designed system architecture in the system.

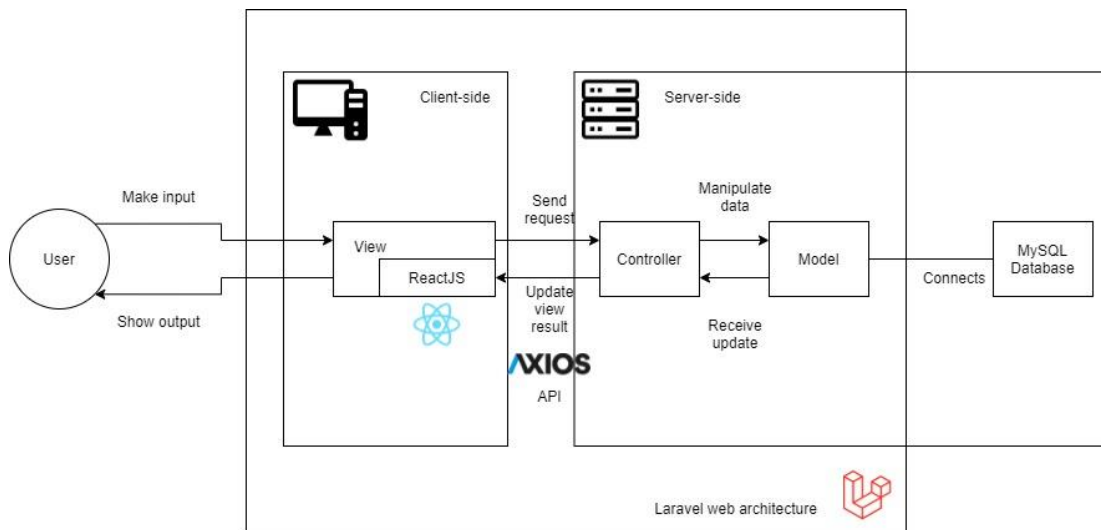


Figure 5-1: Designed System Architecture

5.2 Designed UML Diagrams

During the second iteration, all essential UML diagrams were designed. Class diagram and activity diagram had been provided.

5.2.1 Class Diagram

Class diagram was provided to show the relationship between model classes and the actions between controller class and model class. Designed class diagram had been attached as in Appendix E.

5.2.2 Activity Diagram

Figure 5-2 to Figure 5-12 shows the Activity Diagram based on each use case.

Activity Diagram for Chat In Chat Box

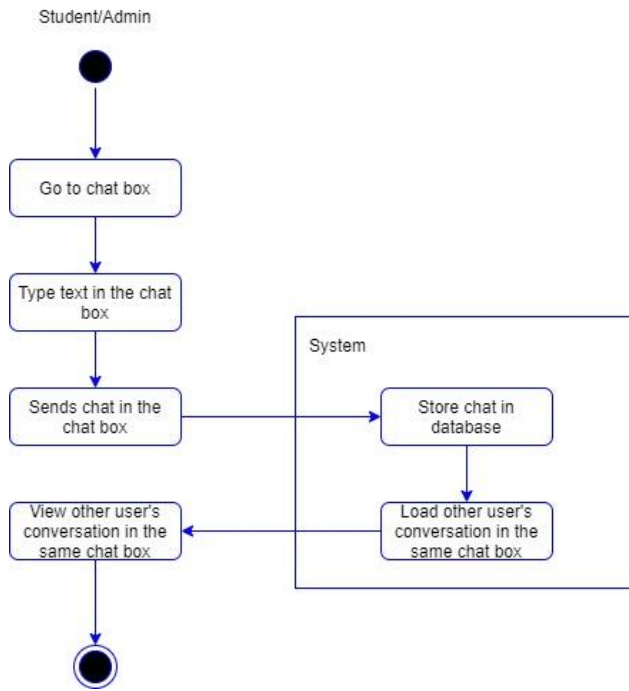


Figure 5-2: Activity Diagram for Chat in Chat Box

Activity Diagram for Do Non-Graded Exercises

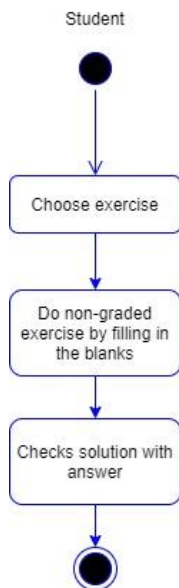


Figure 5-3: Activity Diagram for Do Non-Graded Exercises

Activity Diagram for Edit Codes In Code Compiler

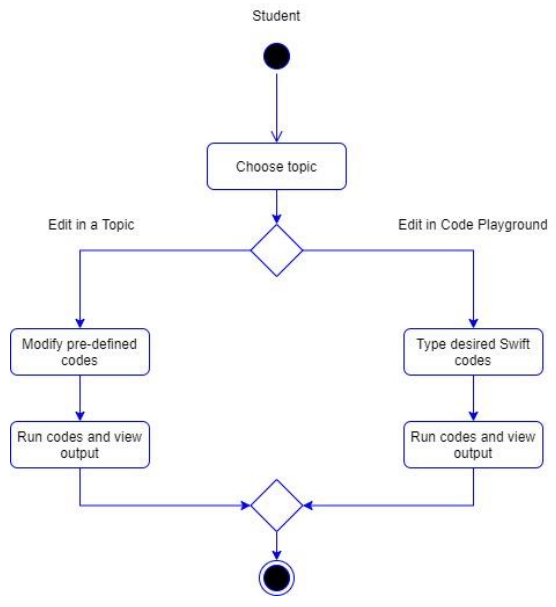


Figure 5-4: Activity Diagram for Edit Codes in Code Compiler

Activity Diagram for Login The System

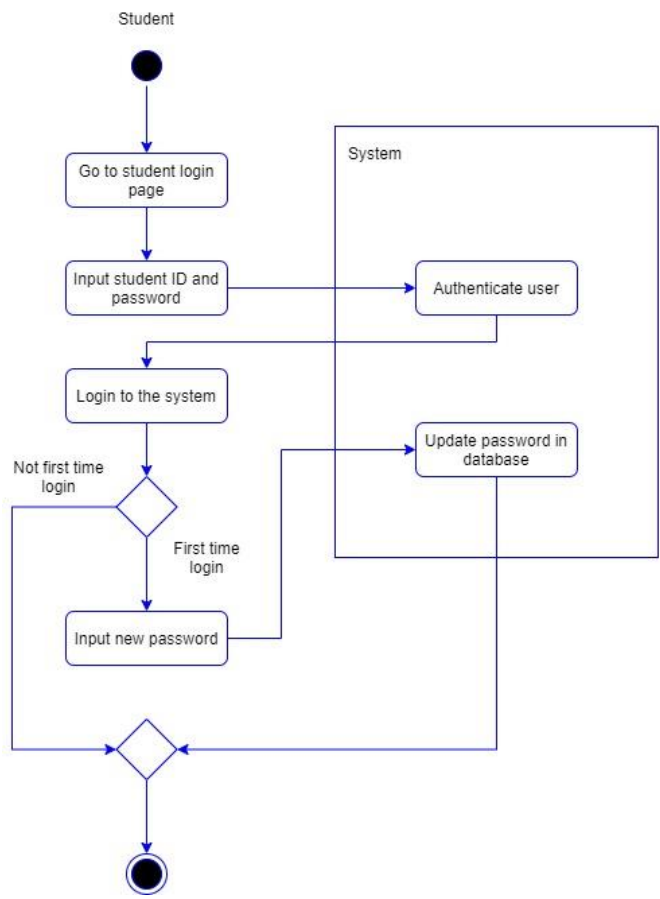


Figure 5-5: Activity Diagram for Login the System



Figure 5-6: Activity Diagram for Logout the System

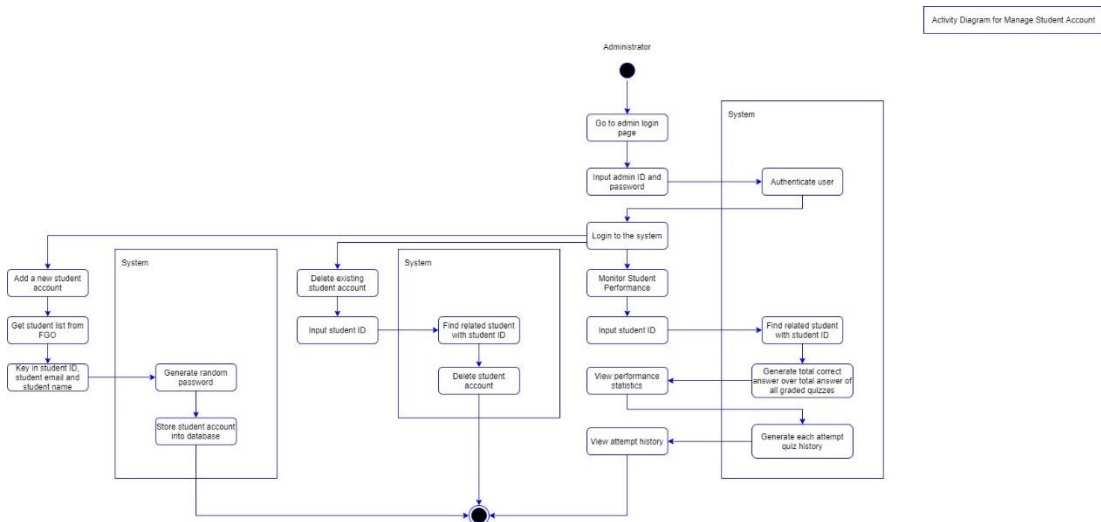


Figure 5-7: Activity Diagram for Manage Student Account

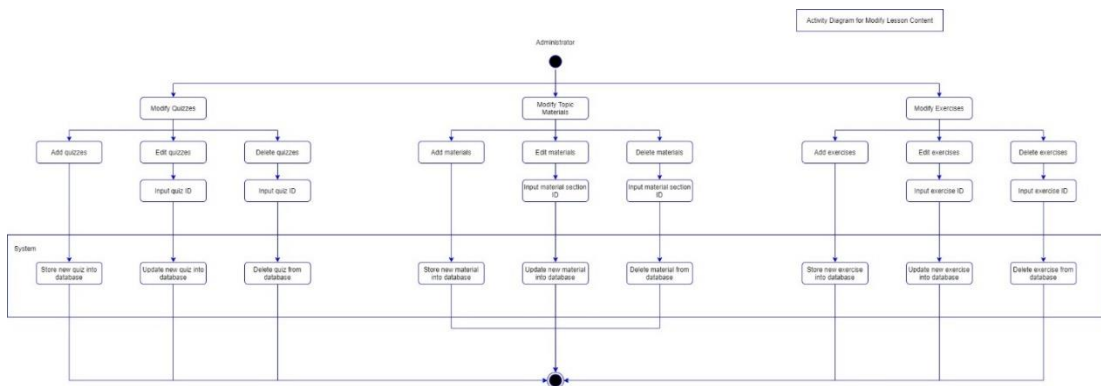


Figure 5-8: Activity Diagram for Modify Lesson Content

Activity Diagram for Read Materials

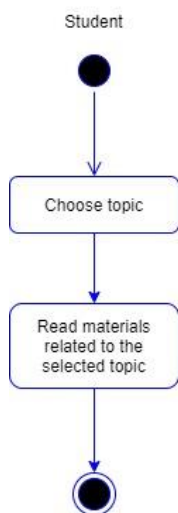


Figure 5-9: Activity Diagram for Read Materials

Activity Diagram for Reset Password

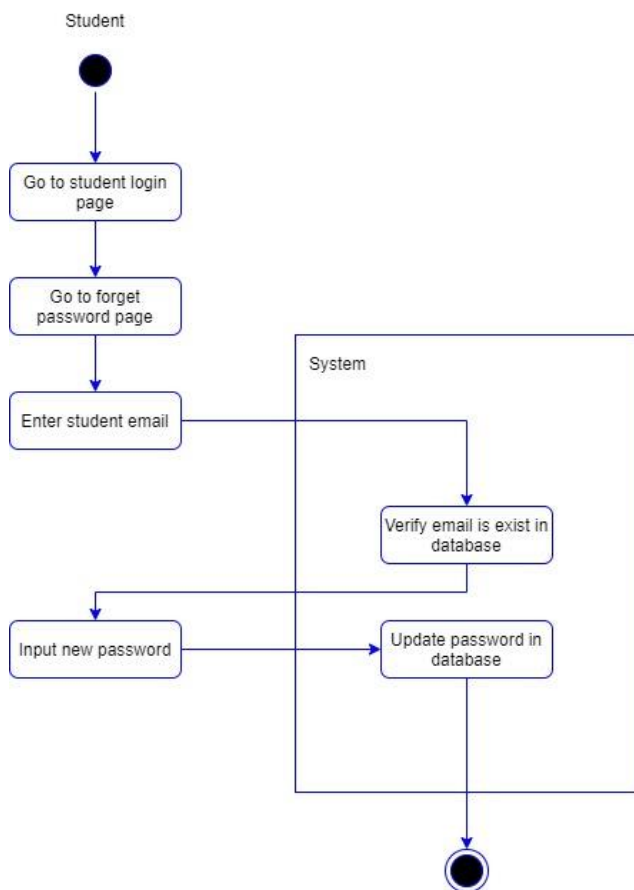


Figure 5-10: Activity Diagram for Reset Password

Activity Diagram for Take Graded Quizzes

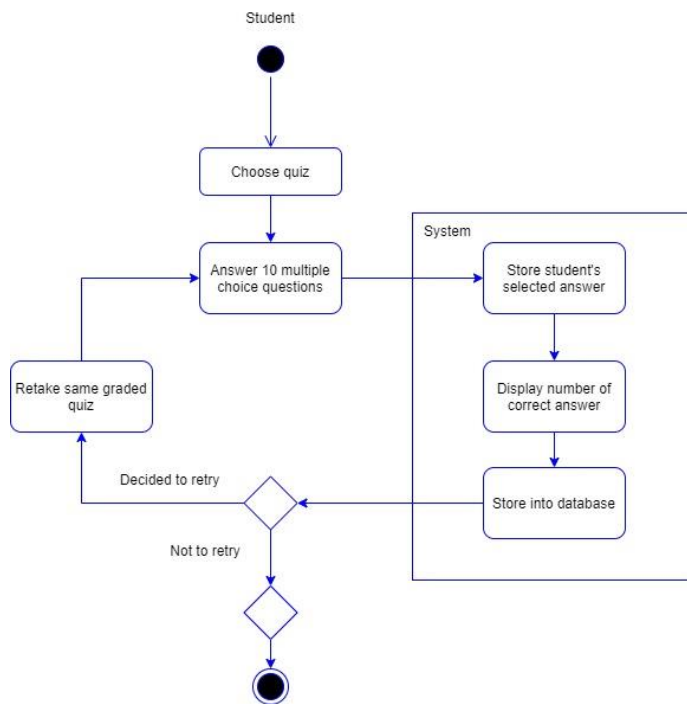


Figure 5-11: Activity Diagram for Take Graded Quizzes

Activity Diagram for View Profile

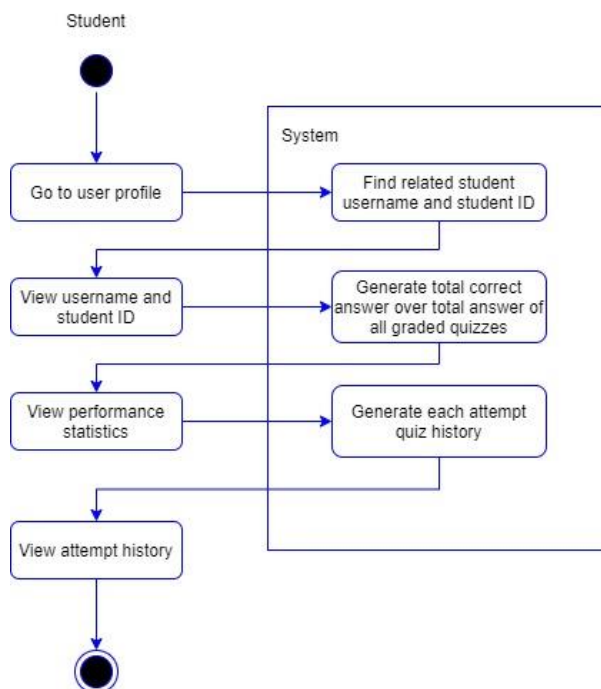


Figure 5-12: Activity Diagram for View Profile

5.3 Database Design

5.3.1 Entity Relationship Diagram (ERD)

Eight tables were designed in the database. Each table had at least an one-to-many relationship with other table. Table “topicitles” represents the name of the topic which affects the exercises, quizzes and lesson sections related to the same topic. Table “topicsections” represents the section with briefing and code examples related to the same topic. Table “exercises” represents the information of exercise questions related to the same topic. Table “quizzes” represents the information of graded quiz questions related to the same topic. Table “quizhistories” represents the history of the quiz related to the respective topic and the student, once the student completed the quiz. Table “students” represents the information of the students. Table “admin” represents the information of the administrator. Table “chats” represents the information of the chat data related to the respective student or administrator. Figure 5-13 shows the diagram of the designed entity relationship diagram.

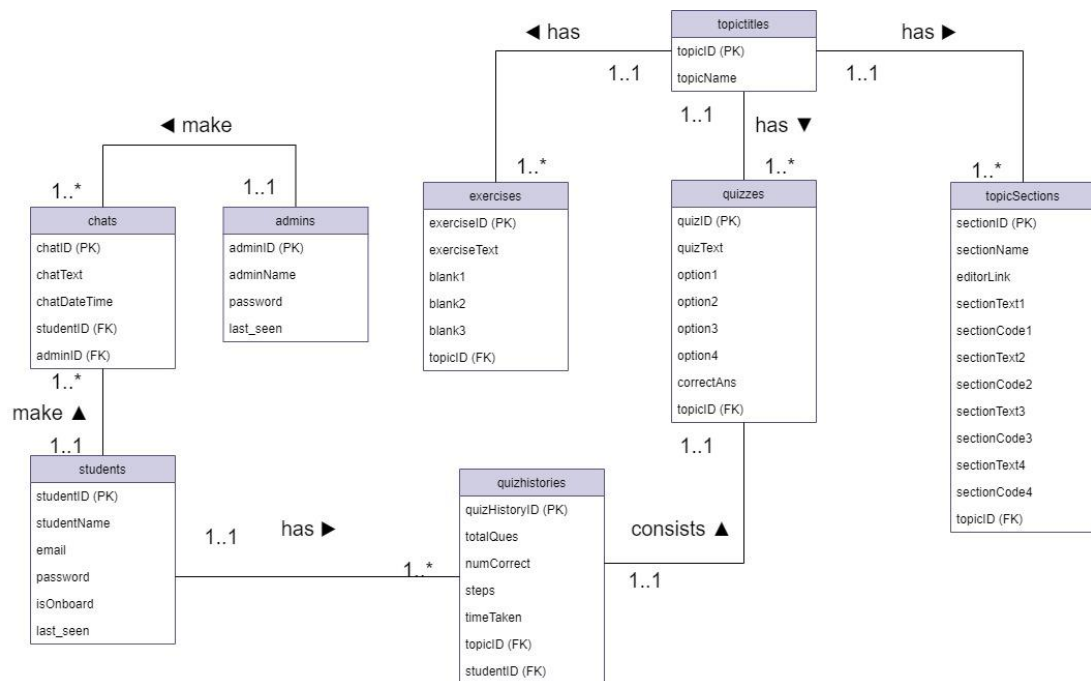


Figure 5-13: Entity Relationship Diagram

5.3.2 Data Dictionary

Table 5-1: Data dictionary for the table “topicitles”

Column Name	Description	Data Type	Size	Primary Key?	Foreign Key?	FK reference table	Nullable
topicID	Unique identification for all topics	Integer	11	Yes	-	-	No
topicName	Topic's title name	varchar	255	-	-	-	No

Table 5-2: Data dictionary for the table “topicsections”

Column Name	Description	Data Type	Size	Primary Key?	Foreign Key?	FK reference table	Nullable
sectionID	Unique identification for all sections	Integer	11	Yes	-	-	No
sectionName	Section name	varchar	255	-	-	-	No
editorLink	Link for online code editor	varchar	255	-	-	-	Yes
sectionText1	Text for the first section paragraph	text	65535	-	-	-	No
sectionCode1	Code text for the first section paragraph	text	65535	-	-	-	Yes
sectionText2	Text for the second section	text	65535	-	-	-	Yes

	paragraph						
sectionCode 2	Code text for the second section paragraph	text	6553 5	-	-	-	Yes
sectionText 3	Text for the third section paragraph	text	6553 5	-	-	-	Yes
sectionCode 3	Code text for the third section paragraph	text	6553 5	-	-	-	Yes
sectionText 4	Text for the fourth section paragraph	text	6553 5	-	-	-	Yes
sectionCode 4	Code text for the fourth section paragraph	text	6553 5	-	-	-	Yes
topicID	Unique identification for all topics	Integer	11	-	Yes	topics	No

Table 5-3: Data dictionary for the table “exercises”

Column Name	Description	Data Type	Size	Primary Key?	Foreign Key?	FK reference table	Nullable
exerciseID	Unique	Integer	11	Yes	-	-	No

	identification for all exercises	r					
exerciseText	Text description for an exercise	text	65535	-	-	-	No
blank1	Pre-defined correct answer for the first blank	varchar	255	-	-	-	No
blank2	Pre-defined correct answer for the second blank	varchar	255	-	-	-	Yes
blank3	Pre-defined correct answer for the third blank	varchar	255	-	-	-	Yes
topicID	Unique identification for all topics	Integer	11	-	Yes	topic titles	No

Table 5-4: Data dictionary for the table “quizzes”

Column Name	Description	Data Type	Size	Primary Key?	Foreign Key?	FK reference table	Nullable
quizID	Unique identification	Integer	11	Yes	-	-	No

	n for all quizzes						
quizText	Text description for a quiz	text	65535	-	-	-	No
option1	First option	varchar	255	-	-	-	No
option2	Second option	varchar	255	-	-	-	No
option3	Third option	varchar	255	-	-	-	Yes
option4	Fourth option	varchar	255	-	-	-	Yes
correctAnswers	Option with correct answer	enum	'a', 'b', 'c', 'd'	-	-	-	No
topicID	Unique identification for all topics	Integer	11	-	Yes	topicitles	No

Table 5-5: Data dictionary for the table “quizhistories”

Column Name	Description	Data Type	Size	Primary Key?	Foreign Key?	FK reference table	Nullable
quizHistoryID	Unique identification for all quiz histories	Integer	11	Yes	-	-	No
totalQues	Total number of	Integer	11	-	-	-	No

	questions exist in the quiz						
numCorrect	Total number of correct answers performed in the quiz	Integer	11	-	-	-	No
steps	Total number of steps go to previous question and next question	Integer	11	-	-	-	No
timeTaken	Time taken to complete all question within the same quiz	Integer	11	-	-	-	No
topicID	Unique identification for all quizzes related to the respective topic	Integer	11	-	Yes	quizzes	No
studentID	Unique identification for all students	Integer	11	-	Yes	students	No

Table 5-6: Data dictionary for the table “students”

Column Name	Description	Data Type	Size	Primary Key?	Foreign Key?	FK reference table	Nullable
studentID	Unique identification for all students	Integer	11	Yes	-	-	No
studentName	Student name	varchar	255	-	-	-	No
email	Student email	varchar	255	-	-	-	No
tempPassword	Student temporary password	varchar	255	-	-	-	Yes
password	Student password with encryption	varchar	255	-	-	-	No
isOnboard	Check whether is first time login to the system	bool	1	-	-	-	No
last_seen	Last login time	datetime	8 bytes	-	-	-	Yes

Table 5-7: Data dictionary for the table “admins”

Column Name	Description	Data Type	Size	Primary Key?	Foreign Key?	FK reference table	Nullable
-------------	-------------	-----------	------	--------------	--------------	--------------------	----------

adminID	Unique identification for all administrators	Integer	11	Yes	-	-	No
adminName	Administrator name	varchar	255	-	-	-	No
password	Administrator password with encryption	varchar	255	-	-	-	No
last_seen	Last login time	datetime	8 bytes	-	-	-	Yes

Table 5-8: Data dictionary for the table “chats”

Column Name	Description	Data Type	Size	Primary Key?	Foreign Key?	FK reference table	Nullable
chatID	Unique identification for all chats	integer	11	Yes	-	-	No
chatText	Text description of a chat	text	65536	-	-	-	No
chatDateTime	Date and time when the chat sends	datetime	8 bytes	-	-	-	No
studentID	Unique identification for all	Integer	11	-	Yes	students	Yes

	students						
adminID	Unique identification for all admins	Integer	11	-	Yes	admins	Yes

5.4 Implemented User Interface

All necessary user interfaces were successfully implemented based on the preliminary designed user interface as stated in Chapter 4 Section 4.5. During the prototype development iteration, some user interfaces were deleted or modified, and some new user interfaces were added. Figure 5-14 to Figure 5-68 shows all of the implemented user interface.

5.4.1 Introduction Screen

Introduction screen is the first screen when the user is starting to use the system. It contains a welcome page with “Student Login” button, “Admin Login” button, and “About” button. The “Student Login” button and “Admin Login” button will redirect the user to their respective login page, whereas the “About” button will redirect the user to the about system page to provide briefing for this platform.

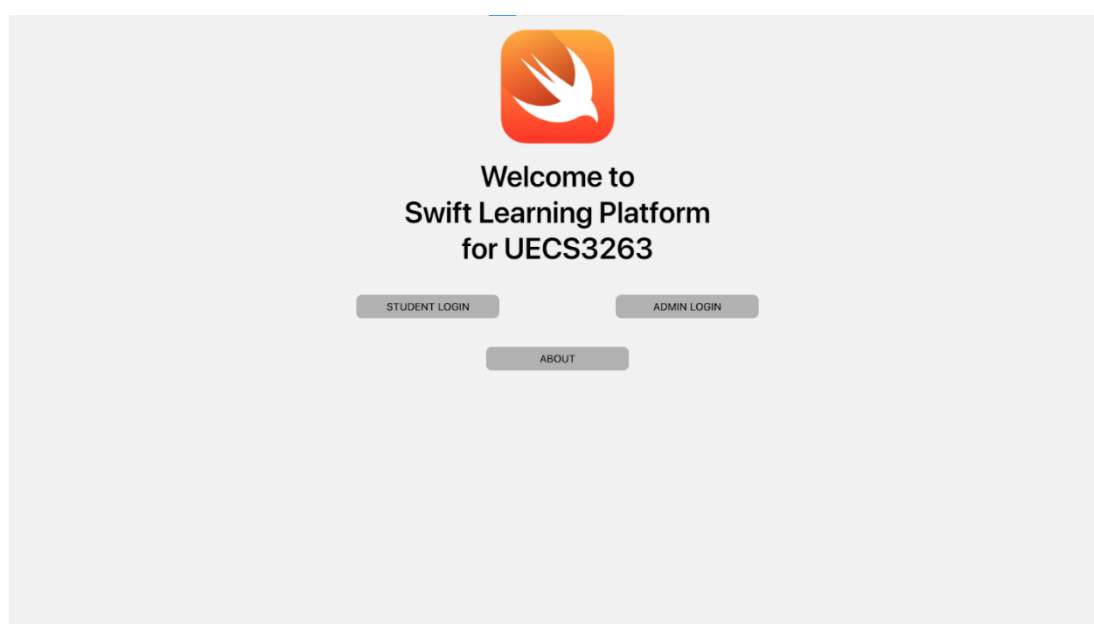


Figure 5-14: Implemented Welcome Screen

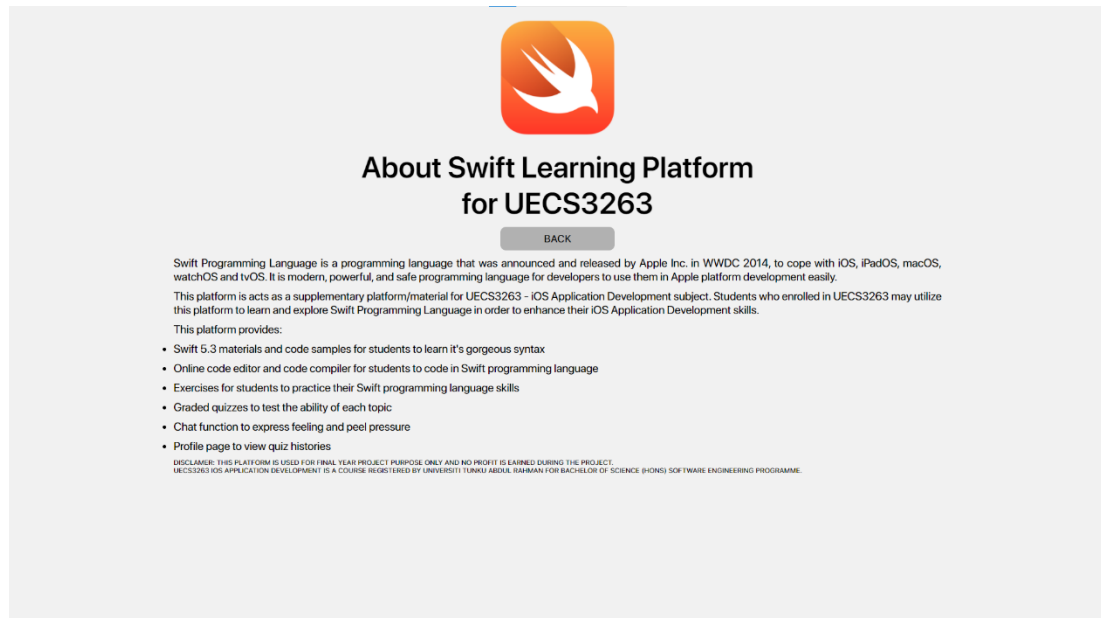


Figure 5-15: Implemented About Screen

5.4.2 Login Screen

There are two types of login screen which are student login screen and administrator login screen, as shown as Figure 5-16 and Figure 5-17. Each of the login screens require the user to key in his/her respective user ID and password. For the student login screen, a reset password button is also provided.

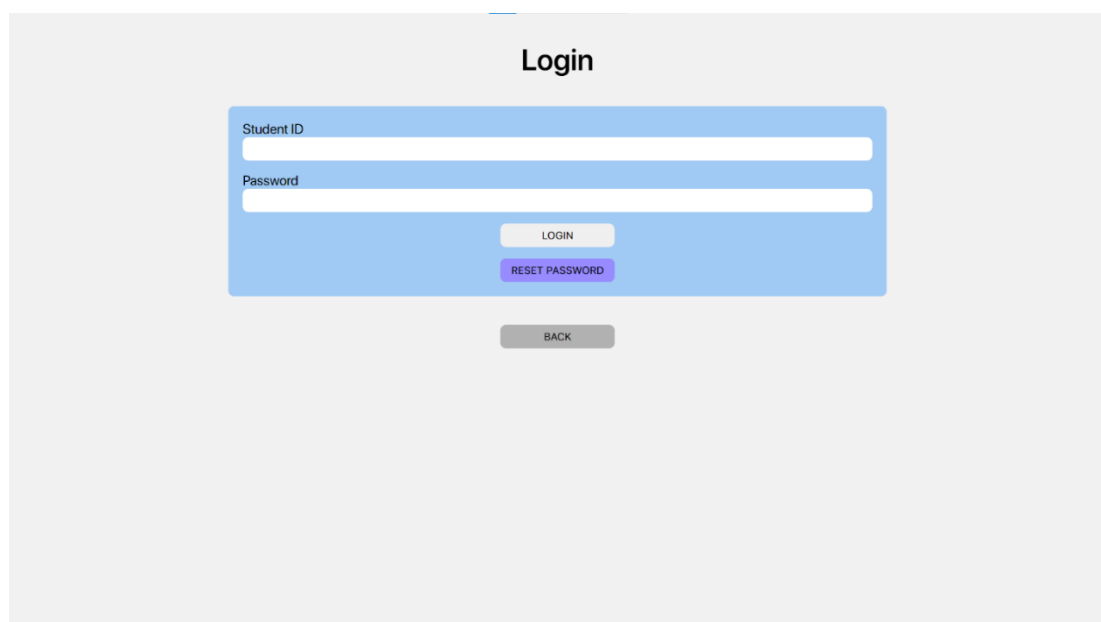
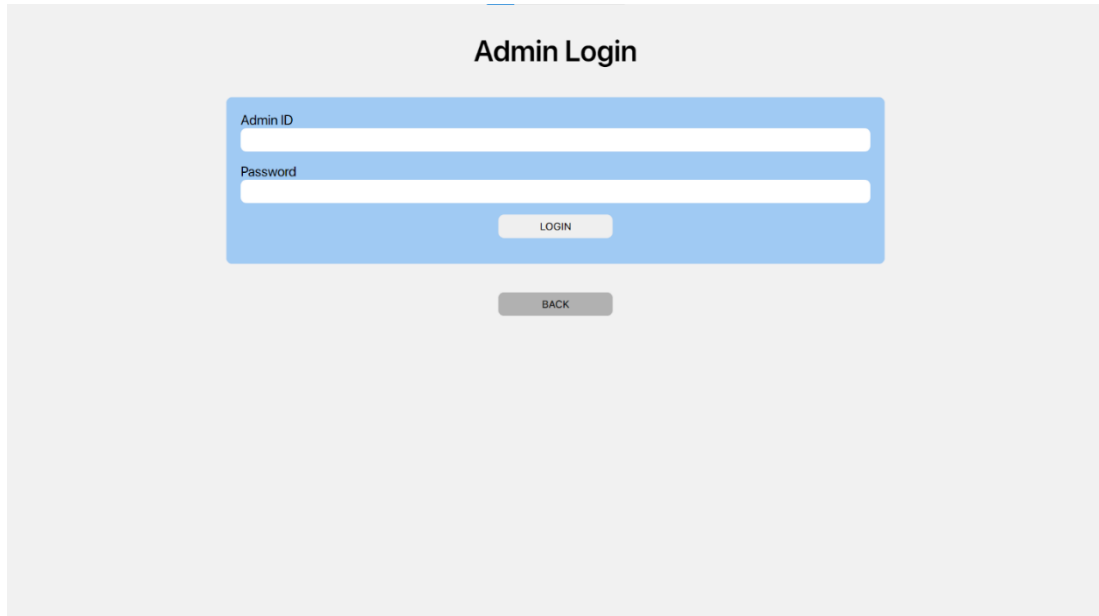


Figure 5-16: Student Login Screen

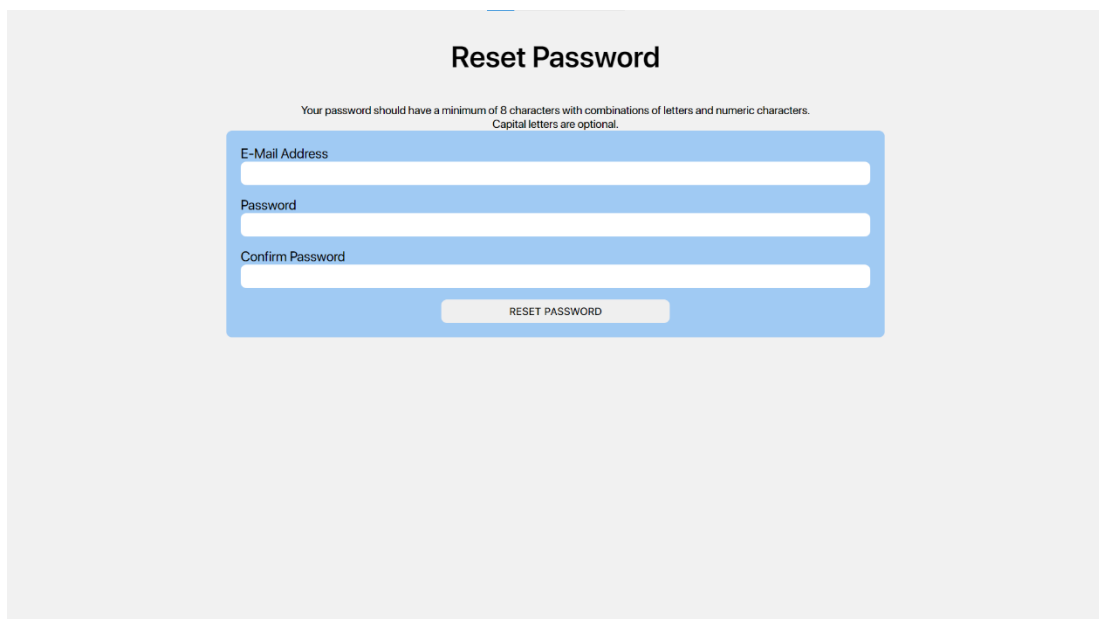


The image shows a web form titled "Admin Login". It features a blue header with the title. Below the header is a blue-bordered form box containing two input fields: "Admin ID" and "Password". Below these fields are two buttons: a light blue "LOGIN" button and a grey "BACK" button.

Figure 5-17: Admin Login Screen

5.4.3 Reset Password Screen

Reset password screen is only applicable for student accounts which have completed the onboarding procedure. Students may key in their student Email, new password and confirmation password to reset their password. Figure 5-18 shows the reset password screen.

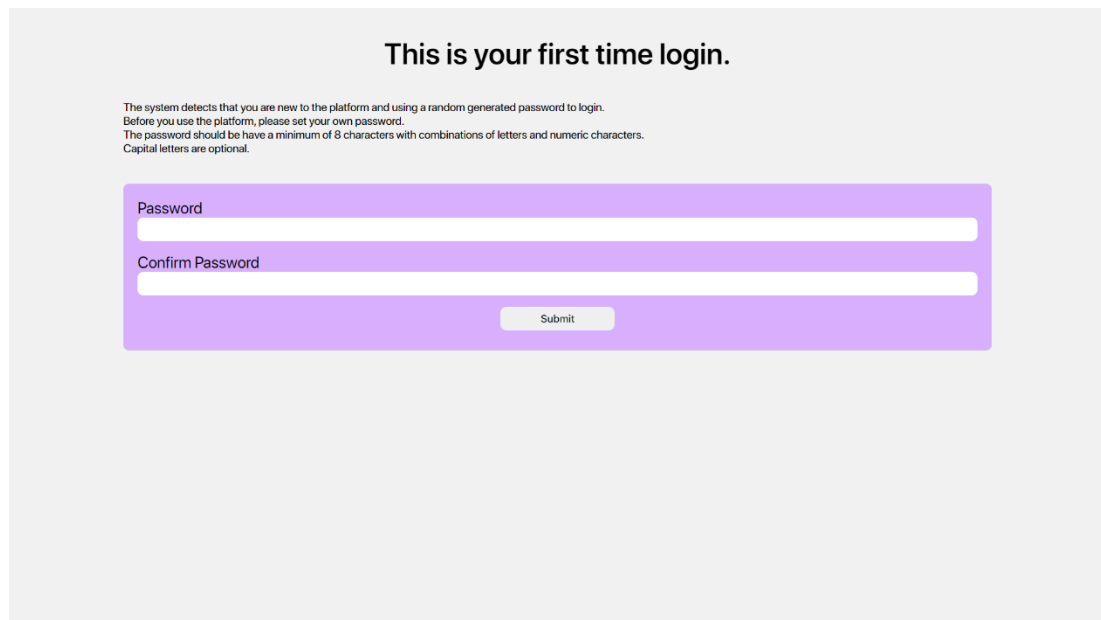


The image shows a web form titled "Reset Password". It features a blue header with the title. Below the header is a blue-bordered form box. At the top of the form box, there is a note: "Your password should have a minimum of 8 characters with combinations of letters and numeric characters. Capital letters are optional." Below this note are three input fields: "E-Mail Address", "Password", and "Confirm Password". Below these fields is a light blue "RESET PASSWORD" button.

Figure 5-18: Reset Password Screen

5.4.4 Onboarding Screen

When a registered student login his/her account for the first time, they will be redirected to the onboarding page which requires the student to reset their password. After the student reset his/her account password, he/she will be redirected to the home screen.



The screenshot shows a light gray background with the following content:

- This is your first time login.**
- The system detects that you are new to the platform and using a random generated password to login.
Before you use the platform, please set your own password.
The password should have a minimum of 8 characters with combinations of letters and numeric characters.
Capital letters are optional.
- A purple-bordered form containing:
 - A text input field labeled "Password".
 - A text input field labeled "Confirm Password".
 - A "Submit" button centered below the input fields.

Figure 5-19: Onboarding Screen

5.4.5 Home Screen and Navigation Bar

There are two types of home screen which are student home screen and administrator home screen. Users who logged into the system will be redirected to their respective home screen. Each home screen contains the greeting text with the username, four big buttons, and a navigation bar. Students may proceed to learn a lesson, take an exercise or graded quiz, view their own profile, chat in a chat box, and code freely in the code playground. Administrators may proceed to register a new student, modify lesson contents, exercise questions or graded quiz questions, check a student's performance, and chat in chat box.

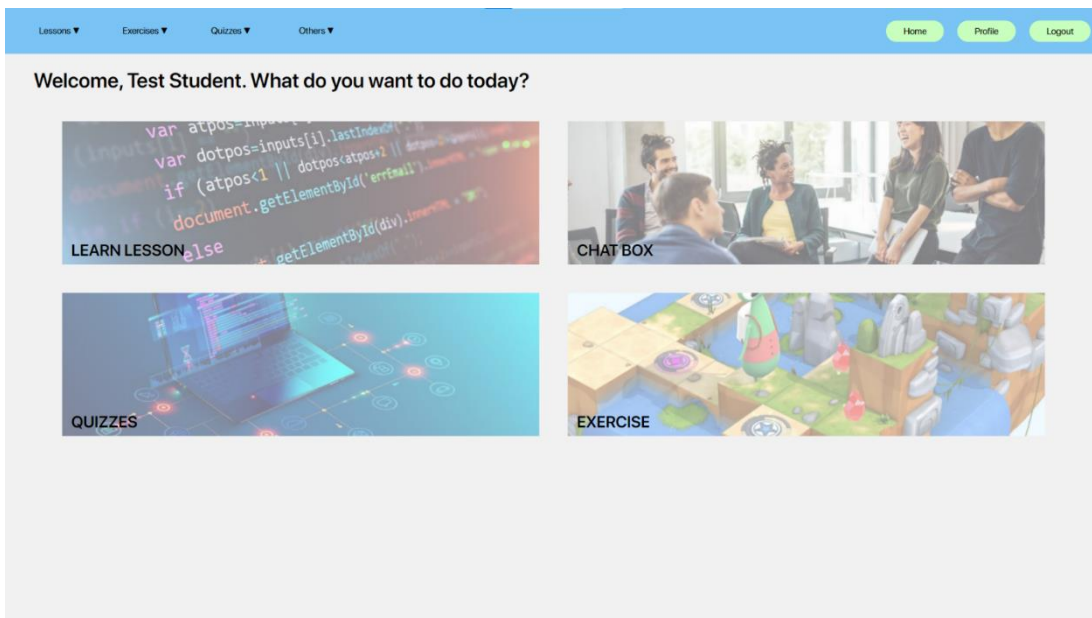


Figure 5-20: Student Home Screen

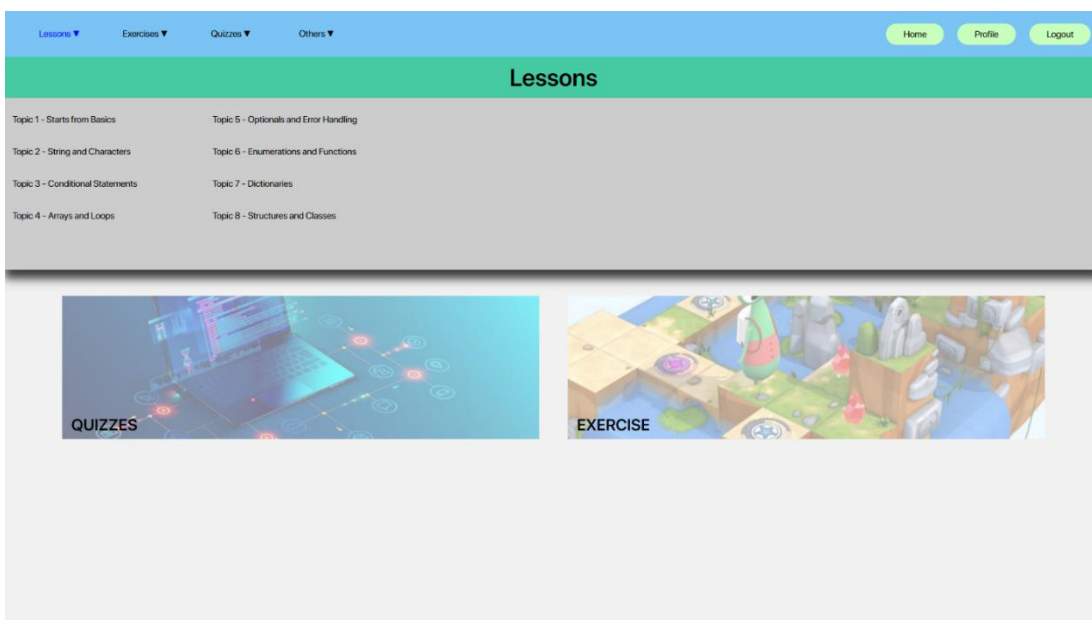


Figure 5-21: Student Navigation Bar for Lessons

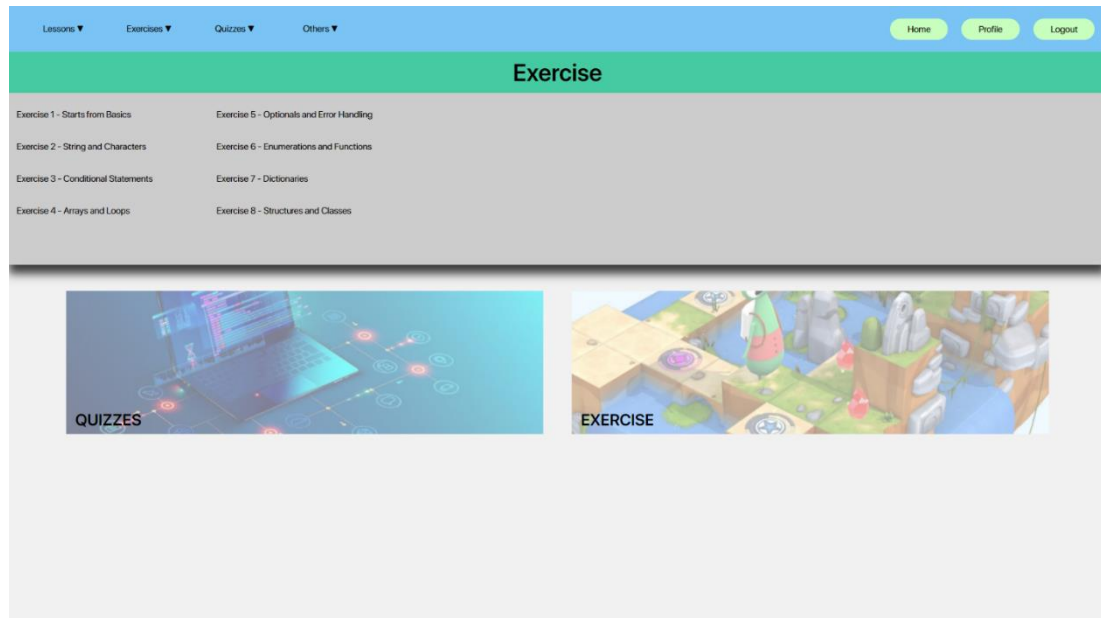


Figure 5-22: Student Navigation Bar for Exercises

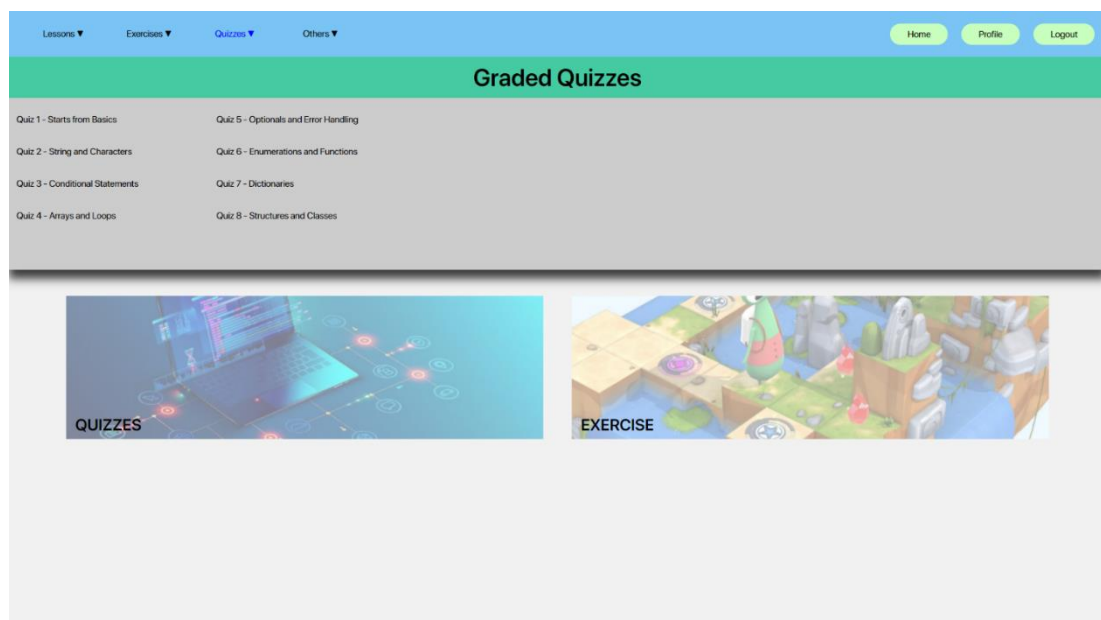


Figure 5-23: Student Navigation Bar for Graded Quizzes

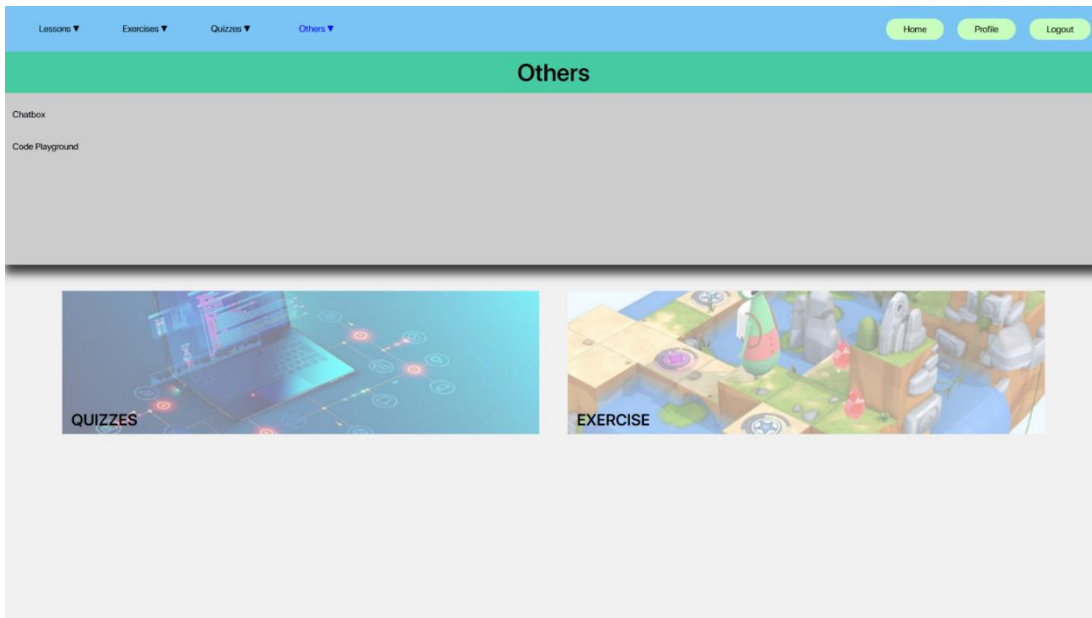


Figure 5-24: Student Navigation Bar for Other Functions

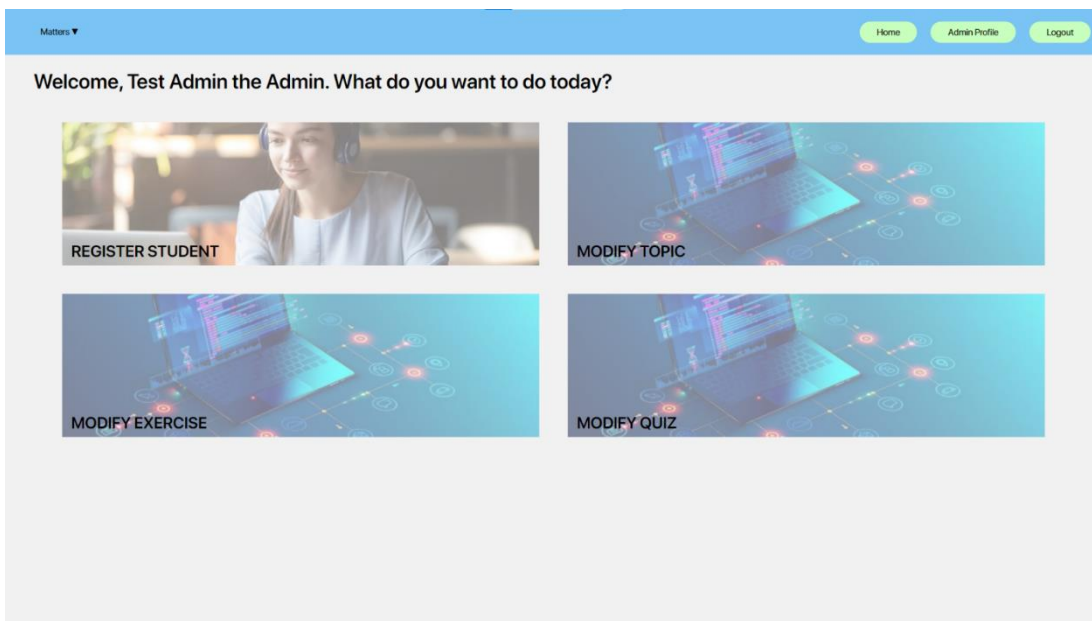


Figure 5-25: Administrator Home Screen

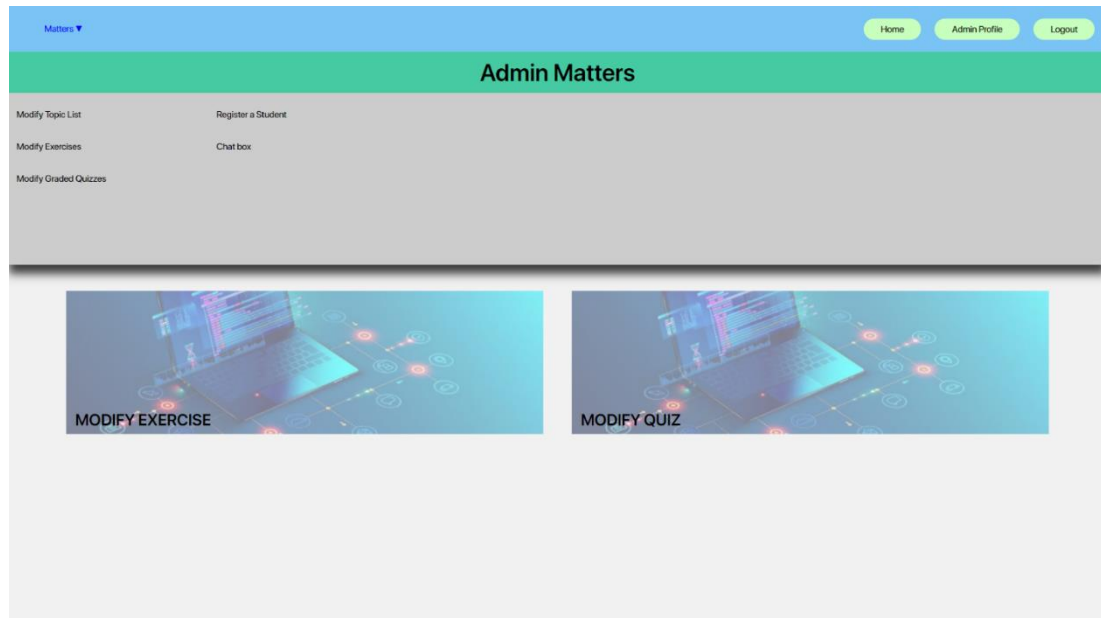


Figure 5-26: Administrator Navigator Bar

5.4.6 Topic Lesson Screen

The topic lesson screen provides a list of topics for students to choose to learn the Swift programming language. Each topic contains a list of sections in the beginning of the screen, buttons for users to go to the next or previous topic, and a series of topic sections related to the same topic. Each topic section may contain explanations, code examples, and a button that opens Paiza.io online code editor to code.

For the administrator, in addition to the prior screen, he/she may also add, edit, or delete a topic. The topic will affect all related topic sections, exercise questions and graded quiz sections. The administrator may also add, edit, or delete a topic section within the same topic. Each modification provides a modal screen to prevent any maloperation by the administrator.

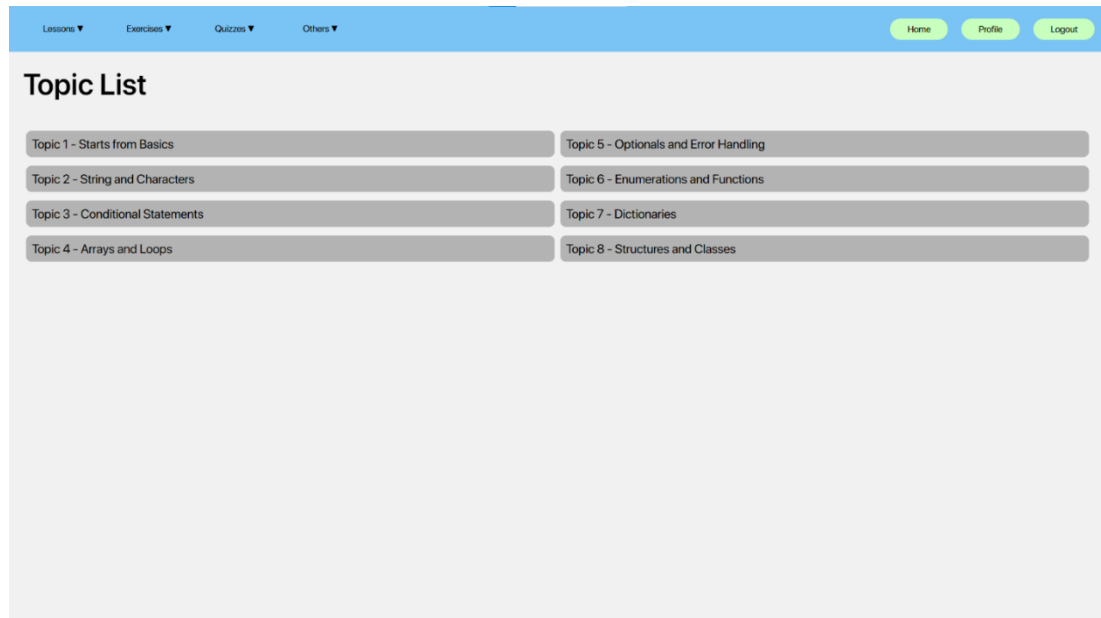


Figure 5-27: Topic List Screen

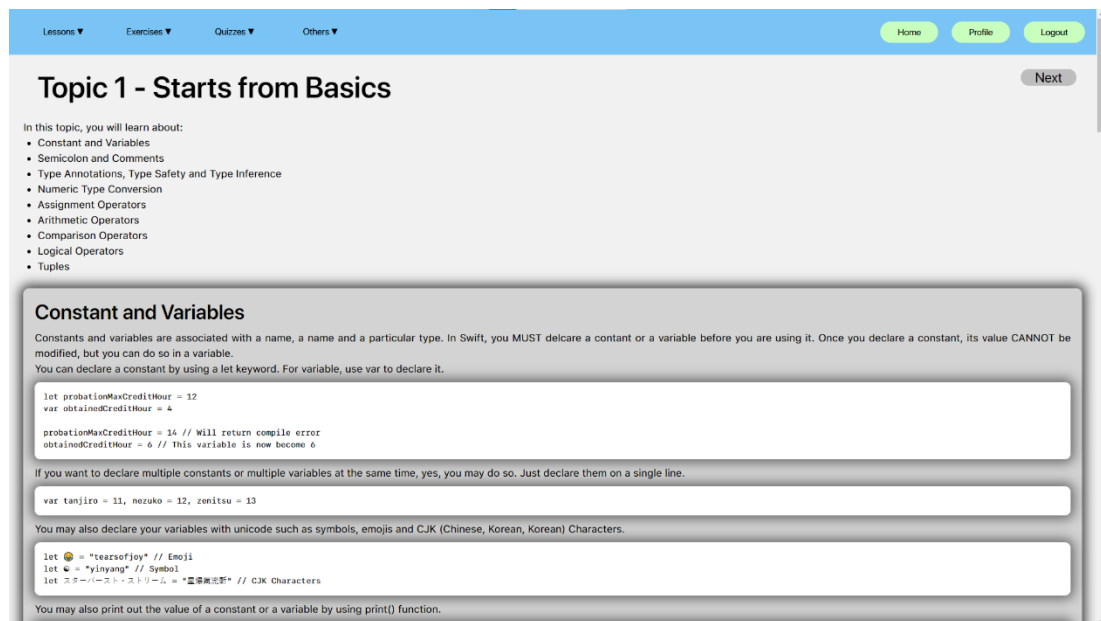


Figure 5-28: Topic Screen

However, you are encouraged to put a type annotation for each constant and variable, so that you won't be confused with the assigned data type of the particular constant or variable. To do so, just put a colon (:) after the constant or variable name, followed by a space and the data type name. You may also assign a value to a type-annotated variable or constant.

```
var myUniversity: String = "UTAR"
```

Swift is a type-safe language that encourages you to be clear about the value types your code can work with, so that you won't accidentally assign a value with wrong data type. You can use `type(of:)` to check the annotated type of the constant or variable.

```
print(type(of:myUniversity)) // return type String
```

Numeric Type Conversion

You can convert an integer to a floating-point number and vice versa. However, you MUST make the numeric constant or variable explicit! Explicit means you had type-annotated a constant or a variable. Please take note that if you convert a floating-point number into an integer, it will be truncated (totally obsolete all the decimal numbers).

```
let six = 6 //this is an integer
let fourPointEightSevenSixThree = 4.8763 //this is a floating-point number
let newNumber = Double(six) + fourPointEightSevenSixThree
// newNumber is inferred to be Double type, and its new value is 10.8763

let newerNumber = Int(newNumber)
// newerNumber is inferred to be Int type, and its new value is 10.
```

You may round off a floating-point value into an integer by using a `round()` function.

```
newerNumber = round(newNumber)
// newerNumber is rounded into 11.
```

[Click to open the code editor for this section!](#)

Assignment Operators

Assignment operators are used to initialize or modify a variable with a value, or initialize a constant with a value. The symbol of assignment operators is `=`.

```
let abc = 10 // constant abc is now equal to 10
var def = abc // variable def is now equal to 10
```

Figure 5-29: Sections in the Topic Screen

`// newerNumber is inferred to be Int type, and its new value is 10.`

You may round off a floating-point value into an integer by using a `round()` function.

```
newerNumber = round(newNumber)
// newerNumber is rounded into 11.
```

[Click to close the code editor for this section!](#)

paiza.io

```
main.swift
1 //You should strictly follow the instructions when converting an floating-point number into an integer.
2
3 //Declare a variable named "new" with a value of 4.869
4
5 //Declare a constant named "new0" to convert new into an integer
6
7 //Declare a constant named "new1" to round-off new's value
8
9 //print both "new0" and "new1" to check that their value is totally different
```

Run (Ctrl-Enter)

Output Input (0.00 sec)

[PaizaCloud](#)

Assignment Operators

Figure 5-30: Section with Opened Online Code Editor in the Topic Screen

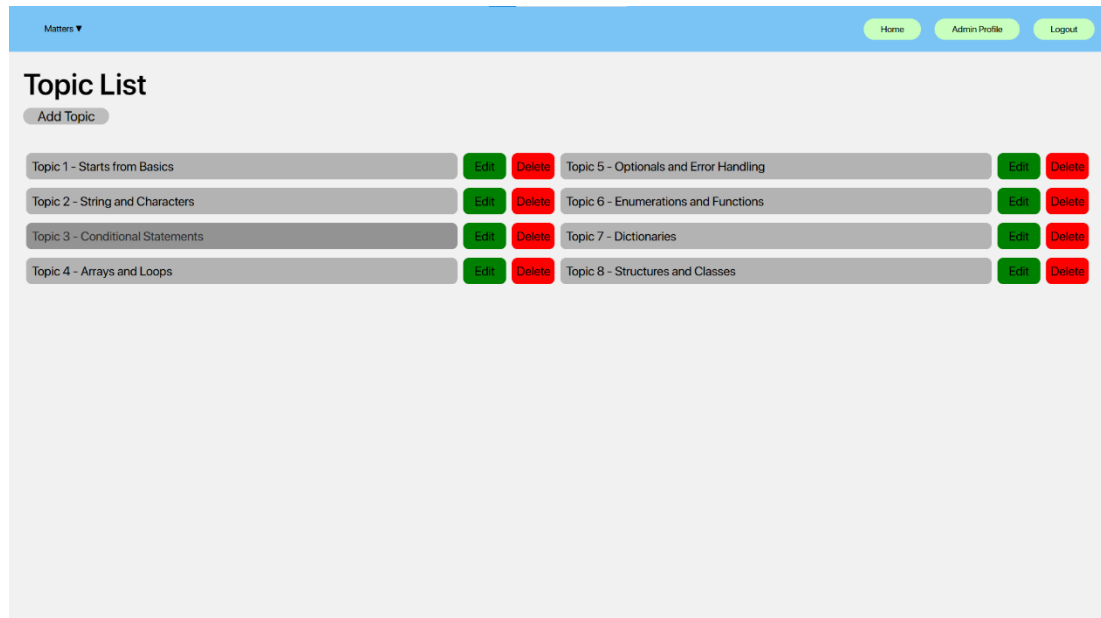


Figure 5-31: Admin Topic List Screen

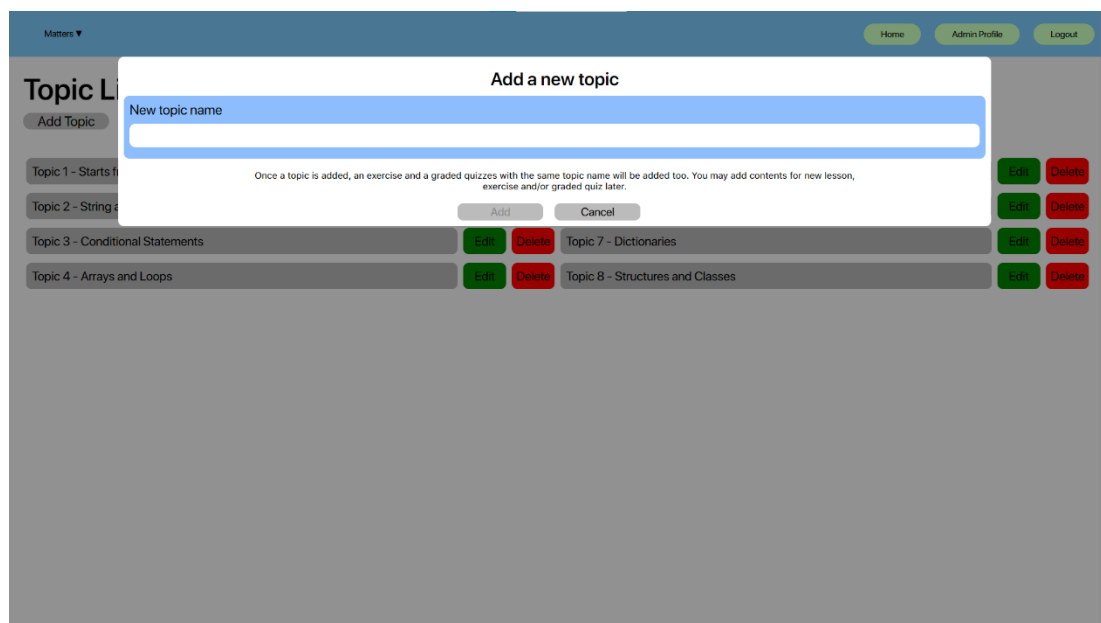


Figure 5-32: Admin Add Topic Modal Screen

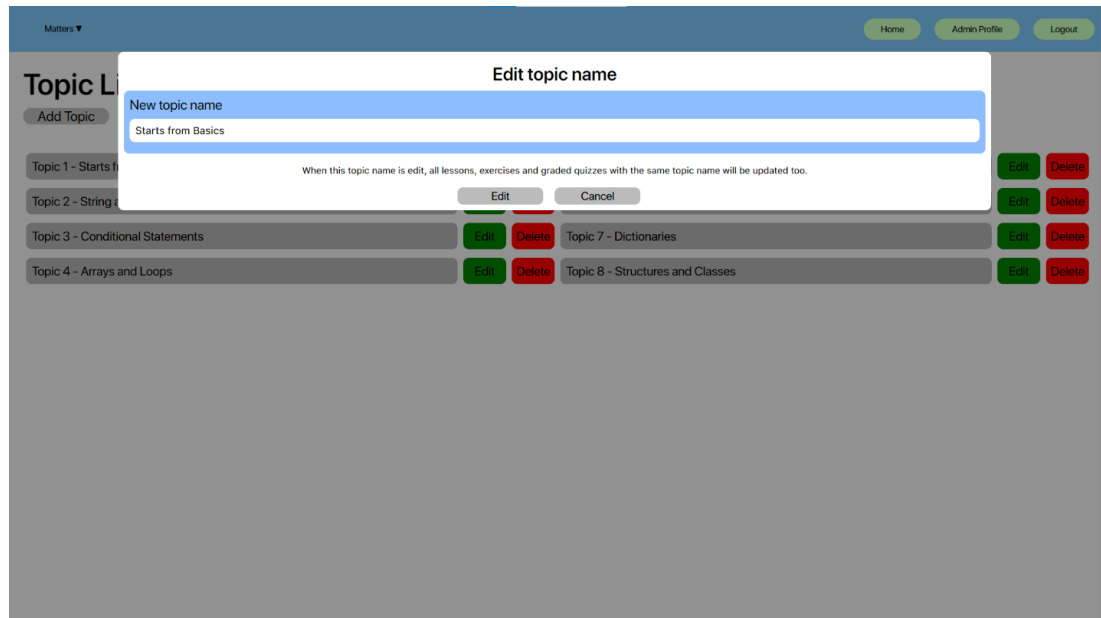


Figure 5-33: Admin Edit Topic Name Modal Screen

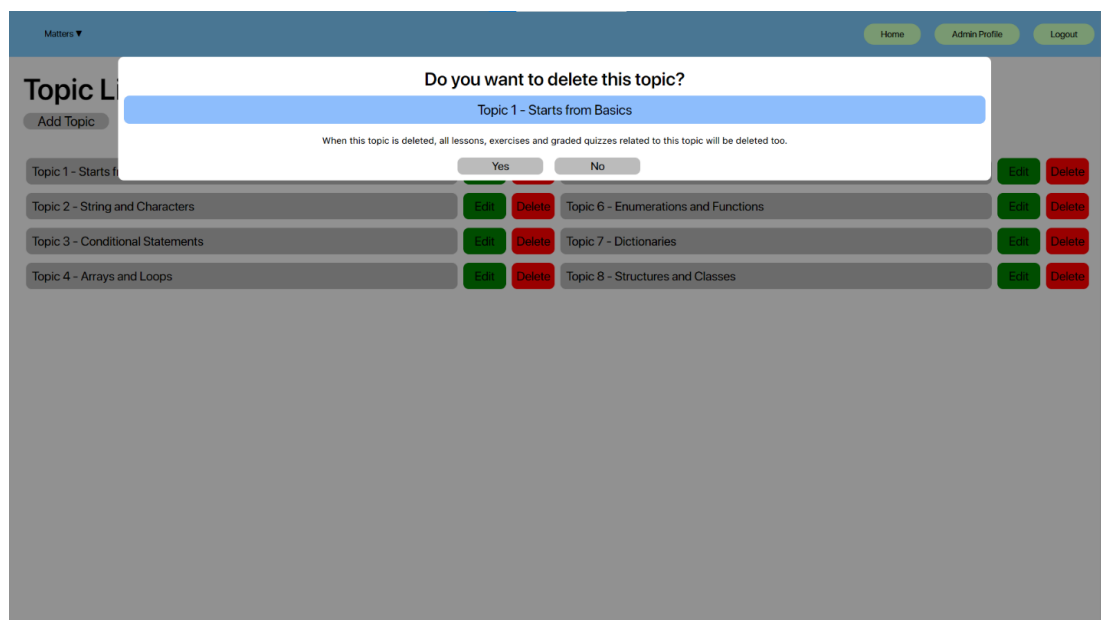


Figure 5-34: Admin Delete Topic Name Modal Screen

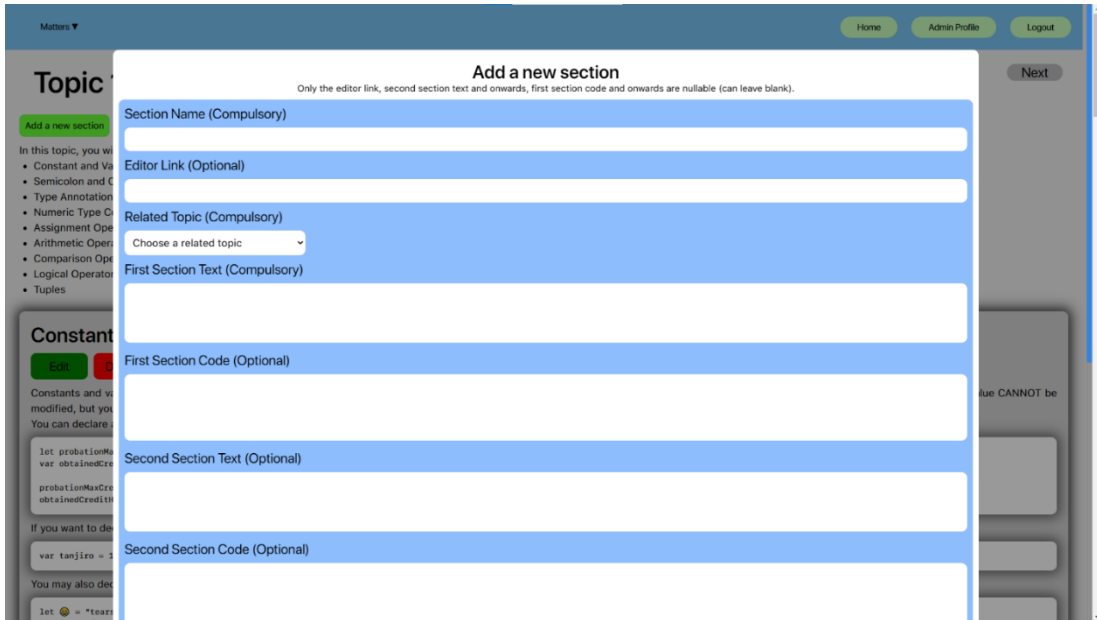


Figure 5-35: Upper Part of Admin Add Topic Section Modal Screen



Figure 5-36: Bottom Part of Admin Add Topic Section Modal Screen

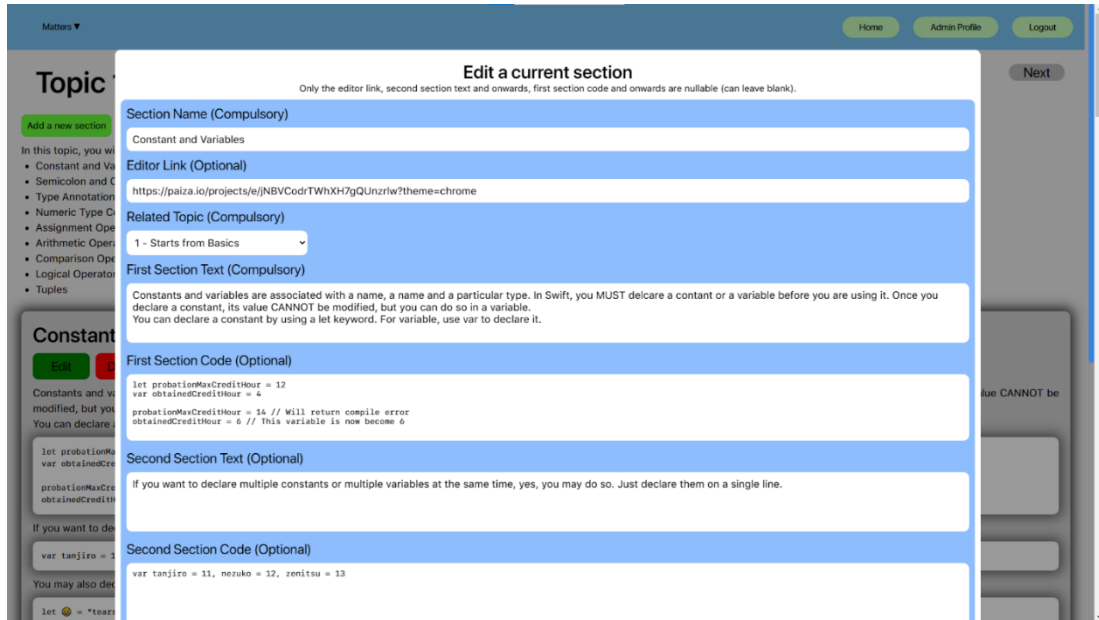


Figure 5-37: Upper Part of Admin Edit Topic Section Modal Screen

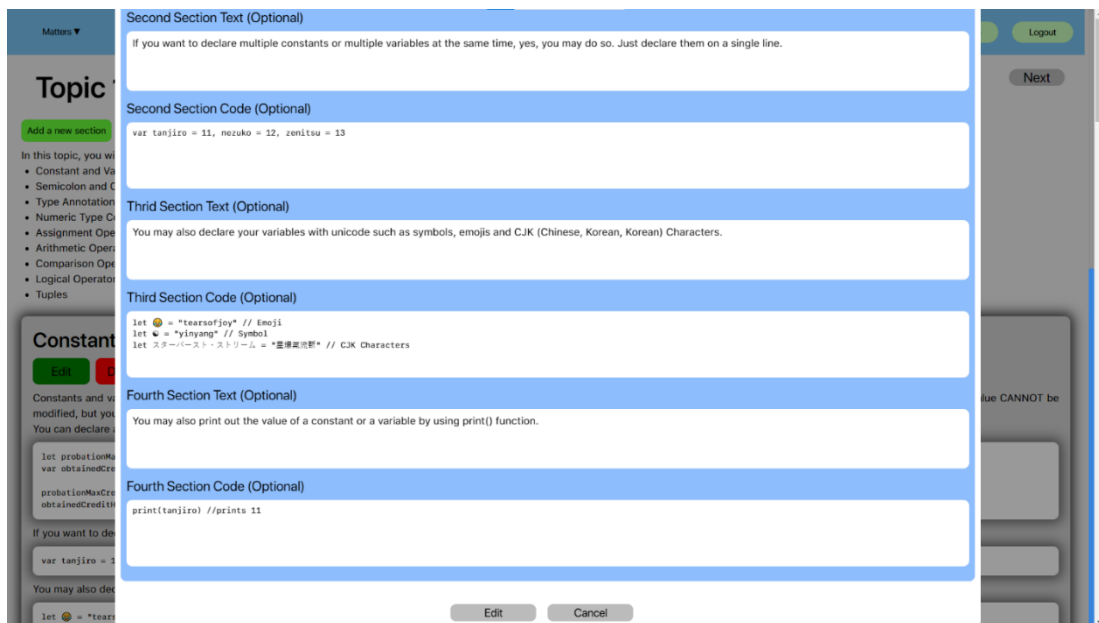


Figure 5-38: Bottom Part of Admin Edit Topic Section Modal Screen

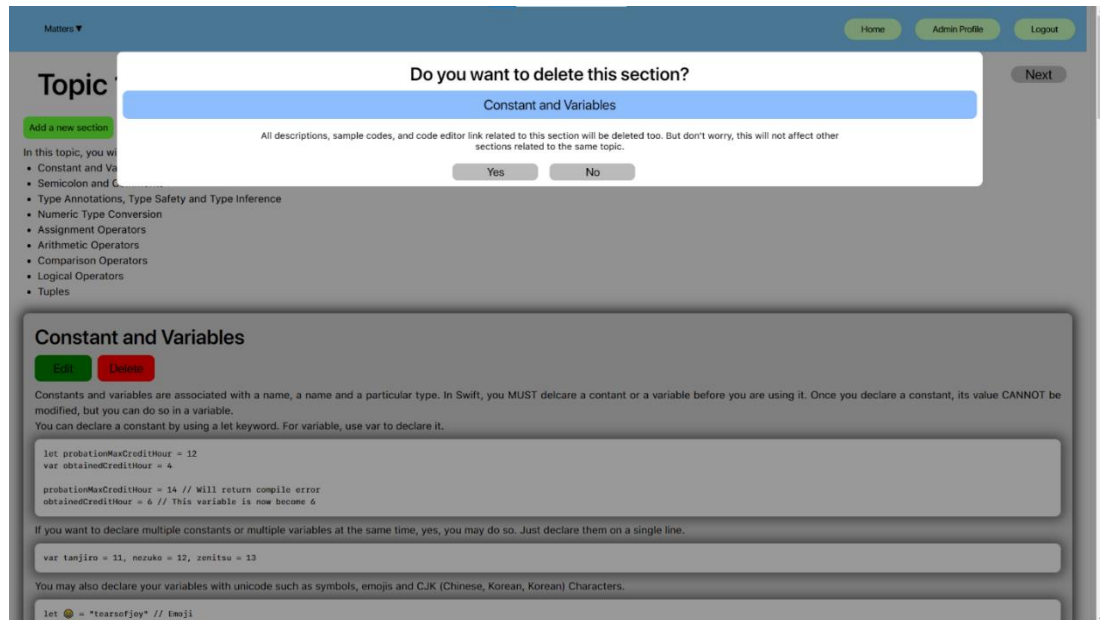


Figure 5-39: Delete Topic Section Screen

5.4.7 Exercise Screen

The exercise screen provides a list of exercises for students to practice their Swift programming language skills. Each exercise contains a minimum of 7 questions. Each question provides a check answer button, submit button, a question test, and at least one blank to fill in the answer.

For the administrator, in addition to the prior screen, he/she may also add, edit, or delete an exercise question. Each modification provides a modal screen to prevent any maloperation by the administrator.

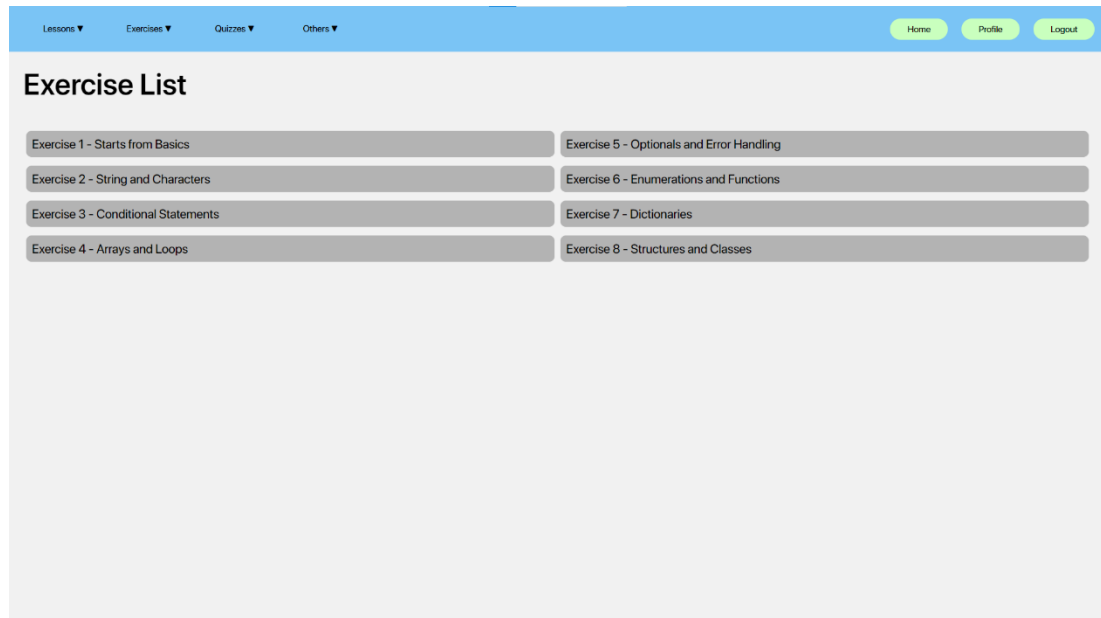


Figure 5-40: Exercise List Screen

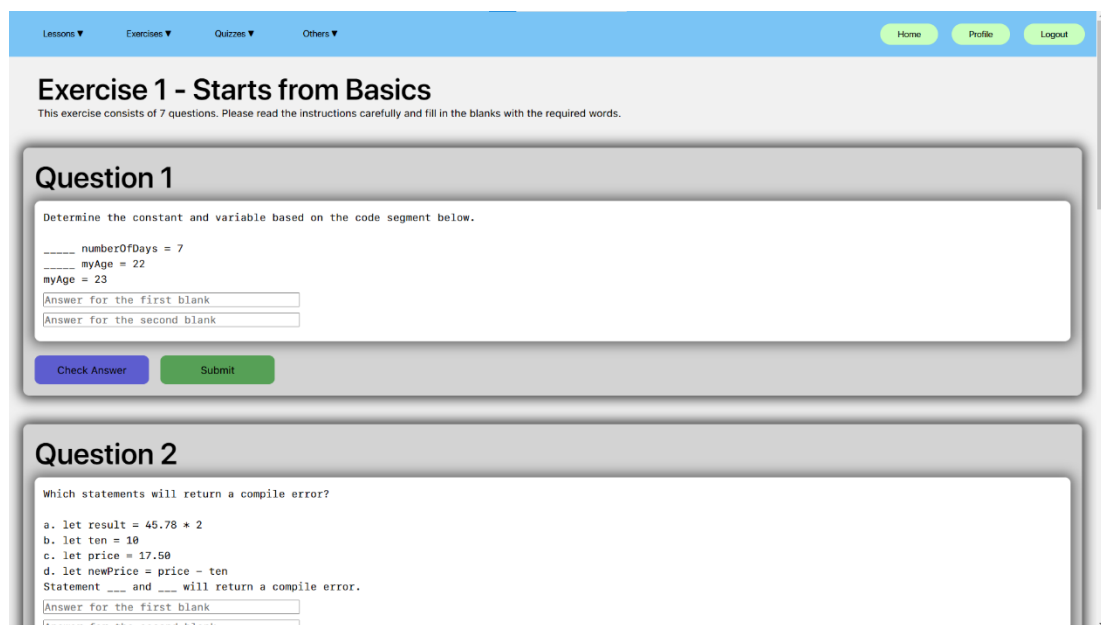


Figure 5-41: Student Exercise Screen

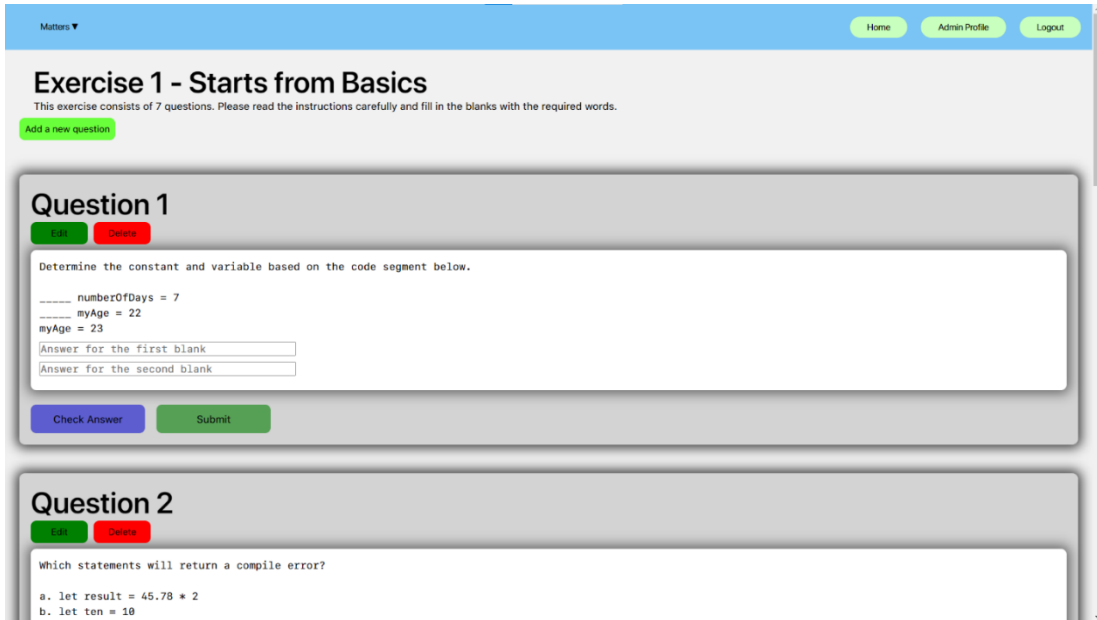


Figure 5-42: Admin Exercise Screen

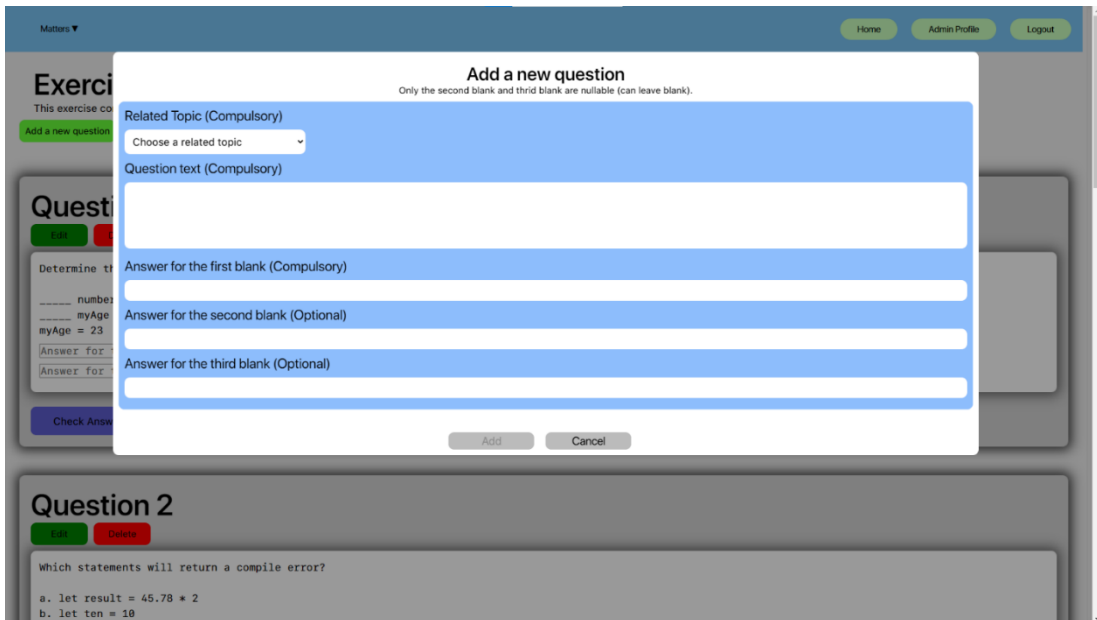


Figure 5-43: Admin Add Exercise Question Modal Screen

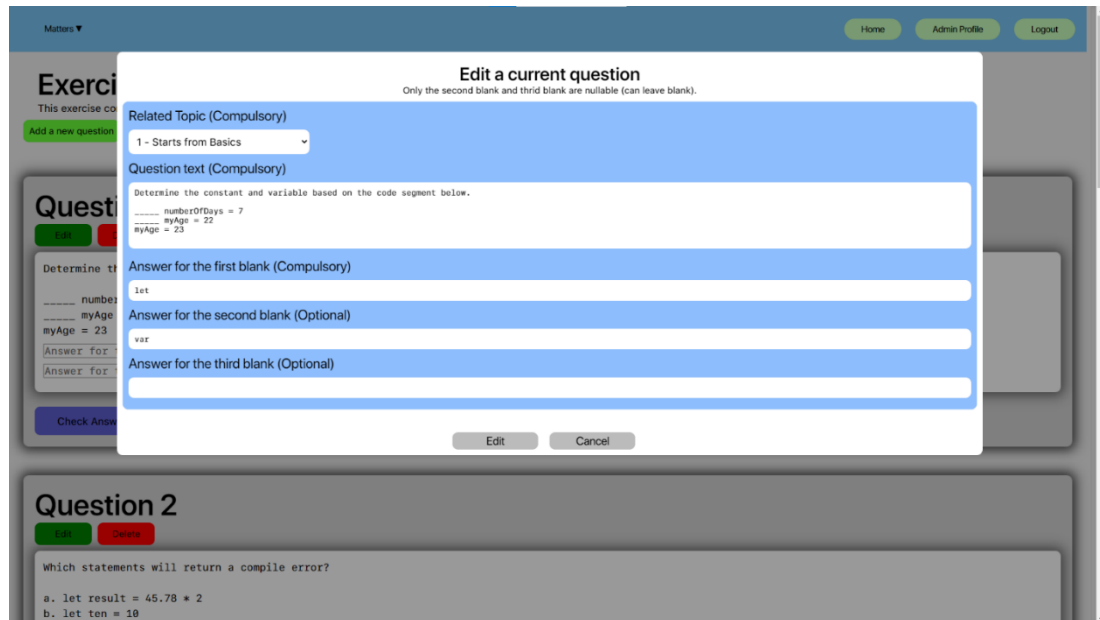


Figure 5-44: Admin Edit Exercise Question Modal Screen

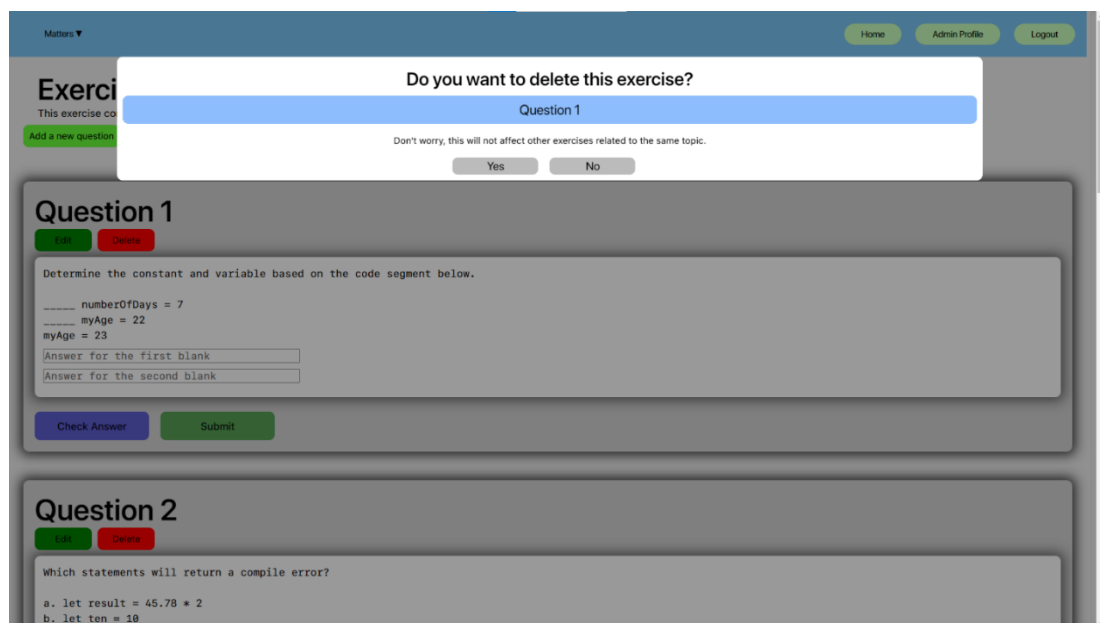


Figure 5-45: Admin Delete Exercise Question Modal Screen

5.4.8 Graded Quiz Screen

The graded quiz screen provides a list of graded quizzes for students to test their Swift coding performance after they completed a topic. Each graded quiz contains a minimum of 10 questions. Students must click the “Start” button to start the graded quizzes. Each question consists of a question text, two to four options, and back/next button. At the last question. A submit button is provided. Once the quiz is submitted,

the student is able to view the graded quiz result, and the correct answer in each question.

For administrator, in addition to prior screen, he/she may also add, edit, or delete a graded question. Each modification provides a modal screen to prevent any maloperation by the administrator.

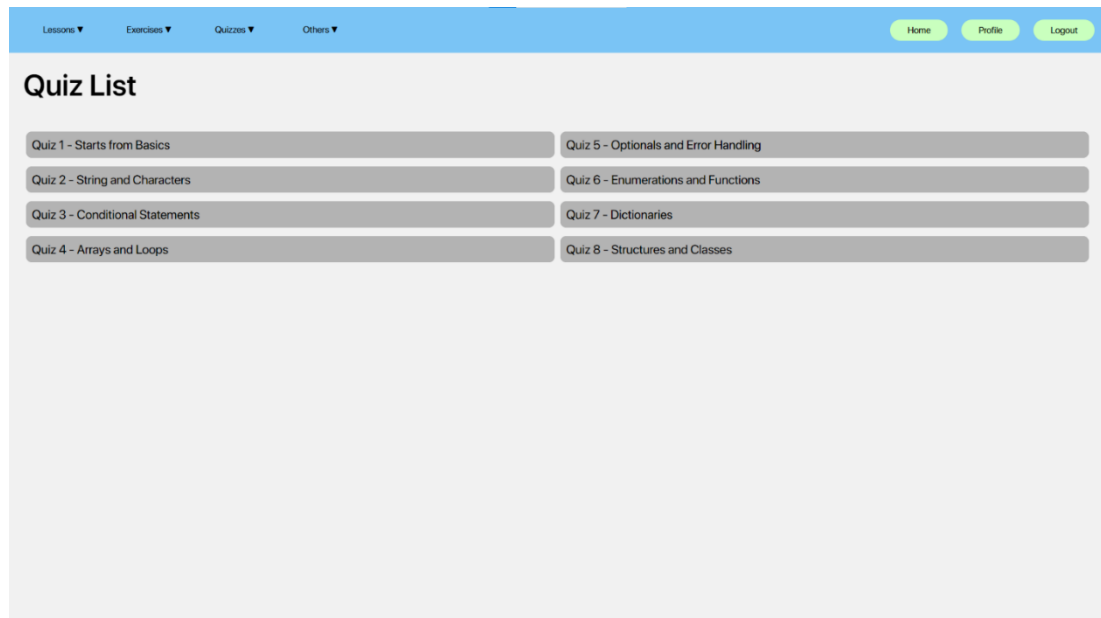


Figure 5-46: Graded Quiz List Screen

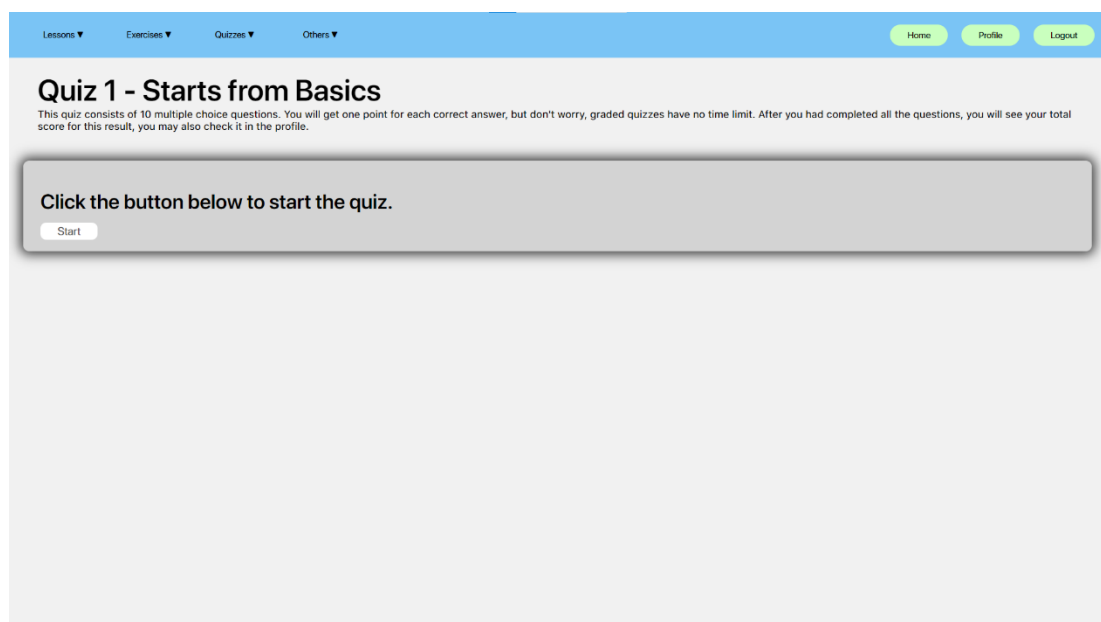


Figure 5-47: Graded Quiz Preparing Screen

The screenshot shows a web interface for a quiz. At the top, there is a navigation bar with links for Lessons, Exercises, Quizzes, and Others. On the right side of the navigation bar, there are buttons for Home, Profile, and Logout. The main heading is "Quiz 1 - Starts from Basics". Below the heading, there is a sub-heading "Question 1". The question text asks: "Based on the code below, why is this code have a compile error?". The code provided is:

```
let someVariable = 5;
someVariable = 6;
```

 There are four radio button options: "The value should be number with decimals", "The constant does not infer with a type", "Constant value cannot be further modified", and "The whole statement does not have semicolon". A "Next" button is located at the bottom of the question area.

Figure 5-48: First Question in Graded Quiz Screen

The screenshot shows the same web interface as Figure 5-48, but for the final question. The heading is "Quiz 1 - Starts from Basics". The sub-heading is "Question 10". The question text asks: "Below is the code sample for a tuple and an array. What is not the difference between tuples and array?". The code provided is:

```
let myself1 = (1, 2, 3)
let myself2 = [1, 2, 3]
```

 There are four radio button options: "Tuples can store different types of data but array cannot", "Tuples and array are list of data", "Tuples use round brackets and array use square brackets", and "Tuple values are immutable but for array, it is mutable". There are two buttons at the bottom: "Prev" and "Submit".

Figure 5-49: Last Question in Graded Quiz Screen

Lessons ▾ Exercises ▾ Quizzes ▾ Others ▾ [Home](#) [Profile](#) [Logout](#)

Quiz 1 - Starts from Basics

This quiz consists of 10 multiple choice questions. You will get one point for each correct answer, but don't worry, graded quizzes have no time limit. After you had completed all the questions, you will see your total score for this result, you may also check it in the profile.

Quiz Result

You have completed this quiz!
Your score: 100% (10/10)

You can try this quiz again anytime by refreshing this page, or select from the navigation bar, or click the retry button.

[Retry](#)

Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10

Question Text

Based on the code below, why is this code have a compile error?

```
let someVariable = 5;
someVariable = 6;
```

Your Answers (Correct answer will be highlighted as green color, else it will be highlighted as red color)

The value should be number with decimals

The constant does not infer with a type

Constant value cannot be further modified

The whole statement does not have semicolon

Figure 5-50: Graded Quiz Result Screen

Matters ▾ [Home](#) [Admin Profile](#) [Logout](#)

Quiz 1 - Starts from Basics

When editing the quizzes, please make sure each question consists of at least 2 options, especially the first two options should not be blank.

[Add a new question](#)

Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10

Question Text [Code](#) [Delete](#)

Based on the code below, why is this code have a compile error?

```
let someVariable = 5;
someVariable = 6;
```

Available Options (Correct answer will be highlighted as green color)

The value should be number with decimals

The constant does not infer with a type

Constant value cannot be further modified

The whole statement does not have semicolon

Figure 5-51: Admin Graded Quiz Screen

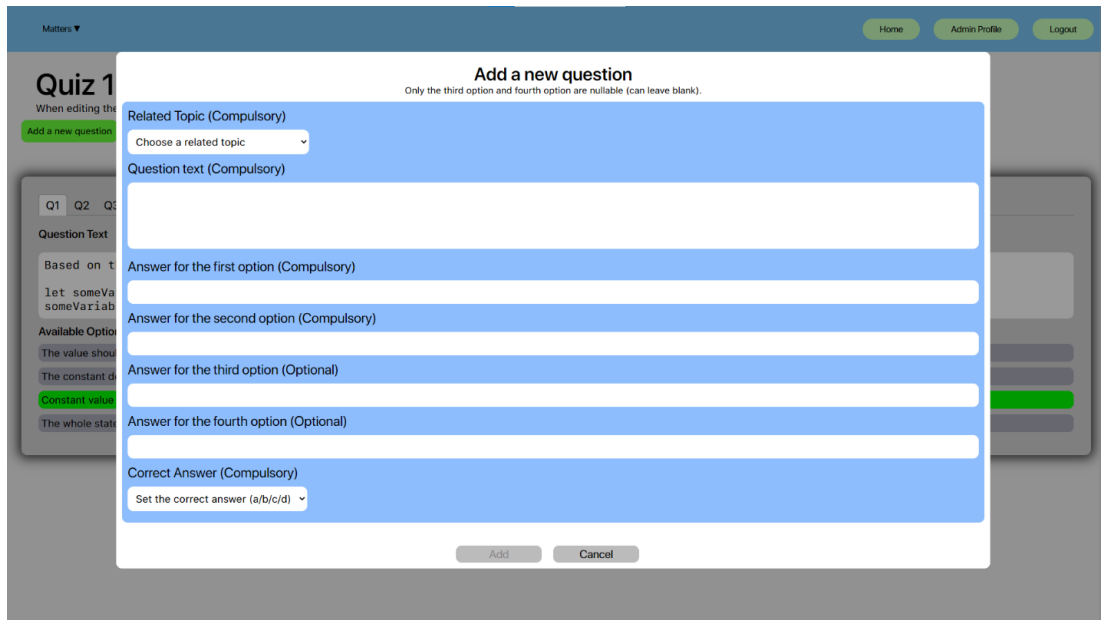


Figure 5-52: Admin Add Graded Quiz Question Modal Screen

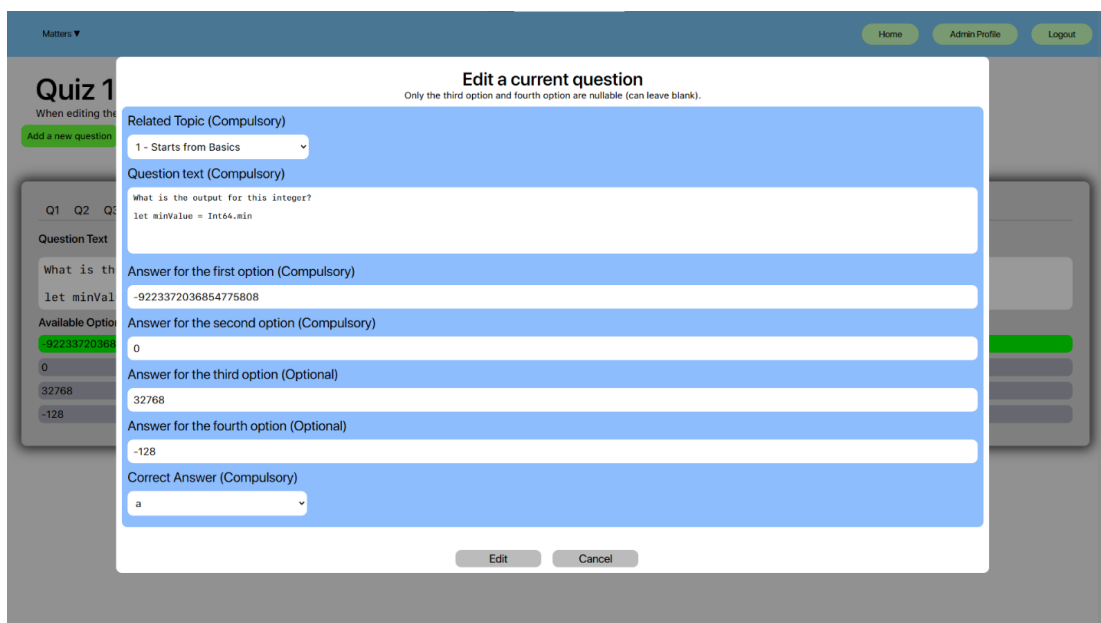


Figure 5-53: Admin Edit Graded Quiz Question Modal Screen

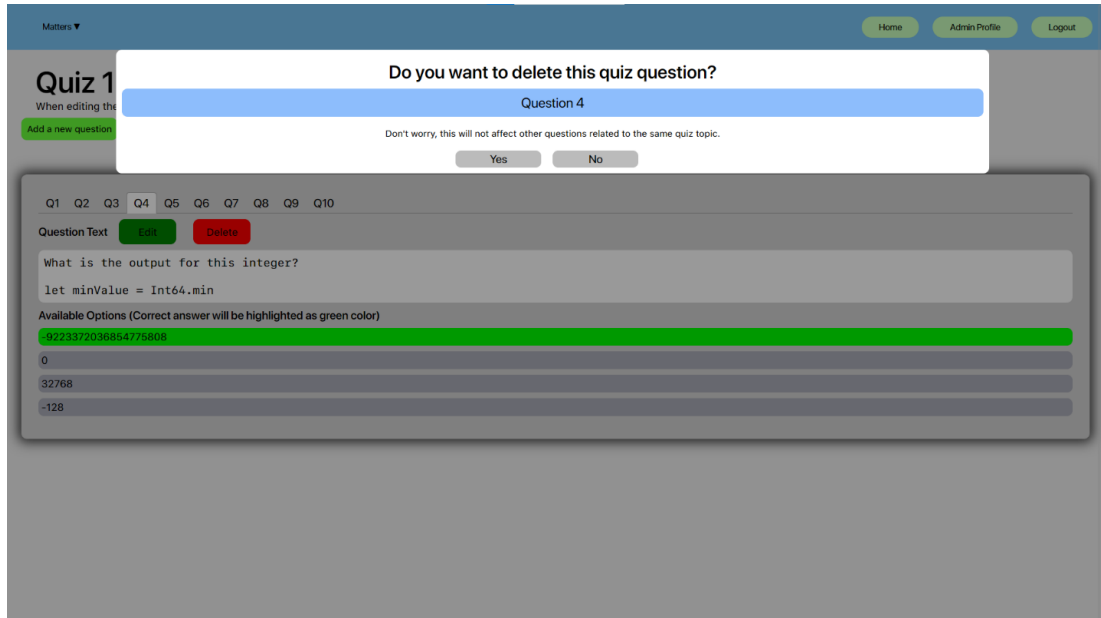


Figure 5-54: Admin Delete Graded Quiz Question Modal Screen

5.4.9 Profile Screen

There are two types of profile screen which are student profile screen and administrator profile screen. In the student profile screen, the student can view his/her student's name, student ID and student Email. The student may also view his/her average performance and quiz histories. Whereas in the administrator profile screen, the student can view his/her administrator's name and administrator ID. The administrator may also view all student basic information such as ID, name, Email and temporary password. The administrator profile screen also provides a search function to search any registered student, in order to view the student's average performance and quiz histories.

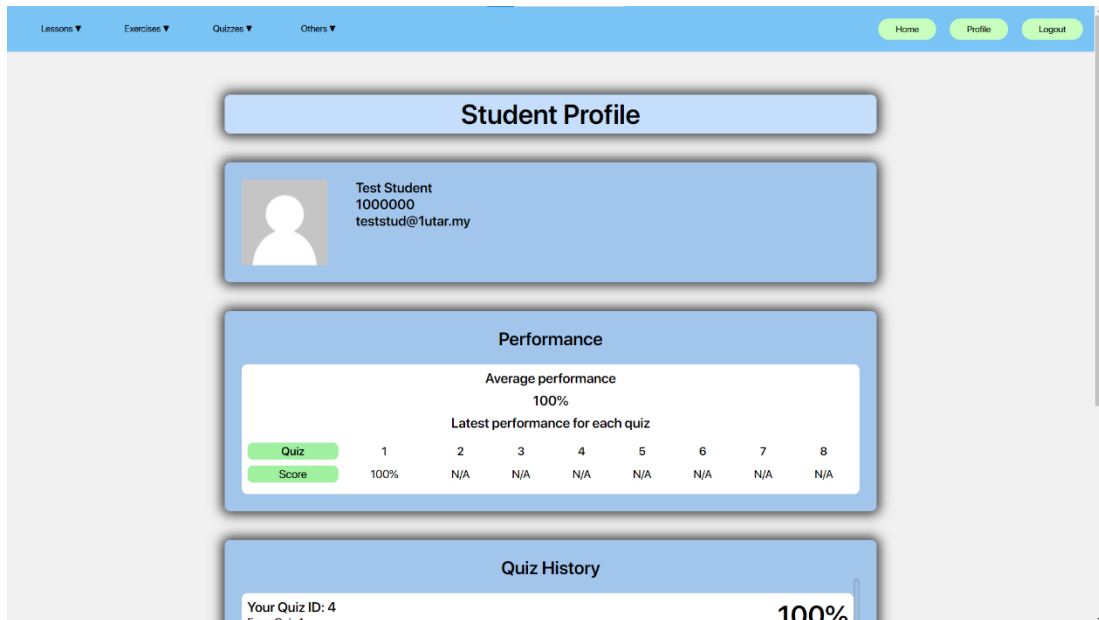


Figure 5-55: Upper Part of Student Profile Screen

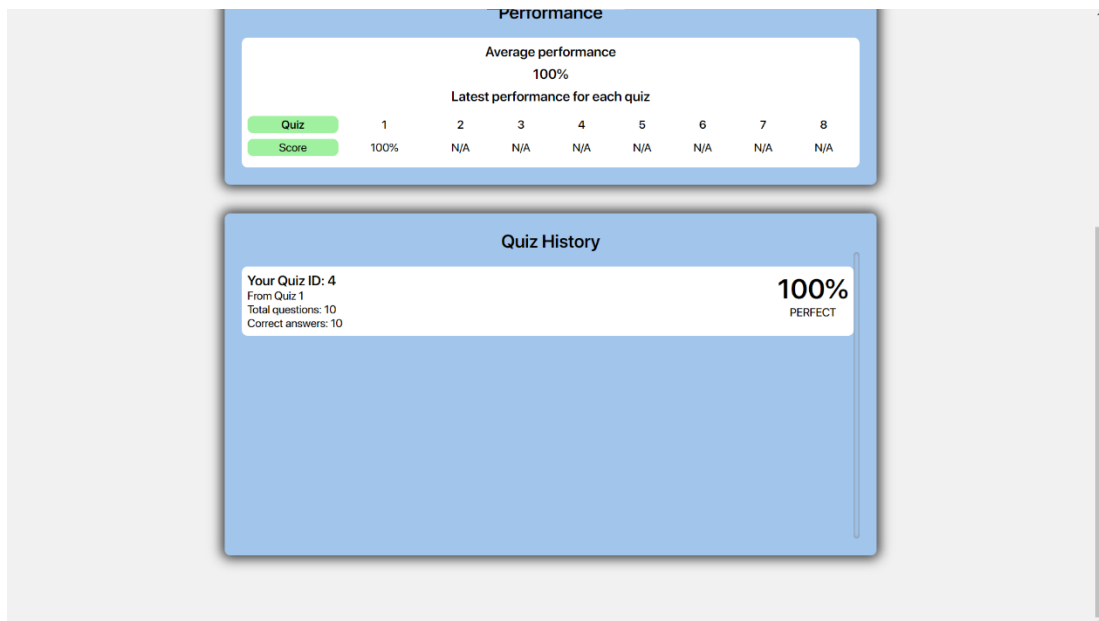


Figure 5-56: Bottom Part of Student Profile Screen

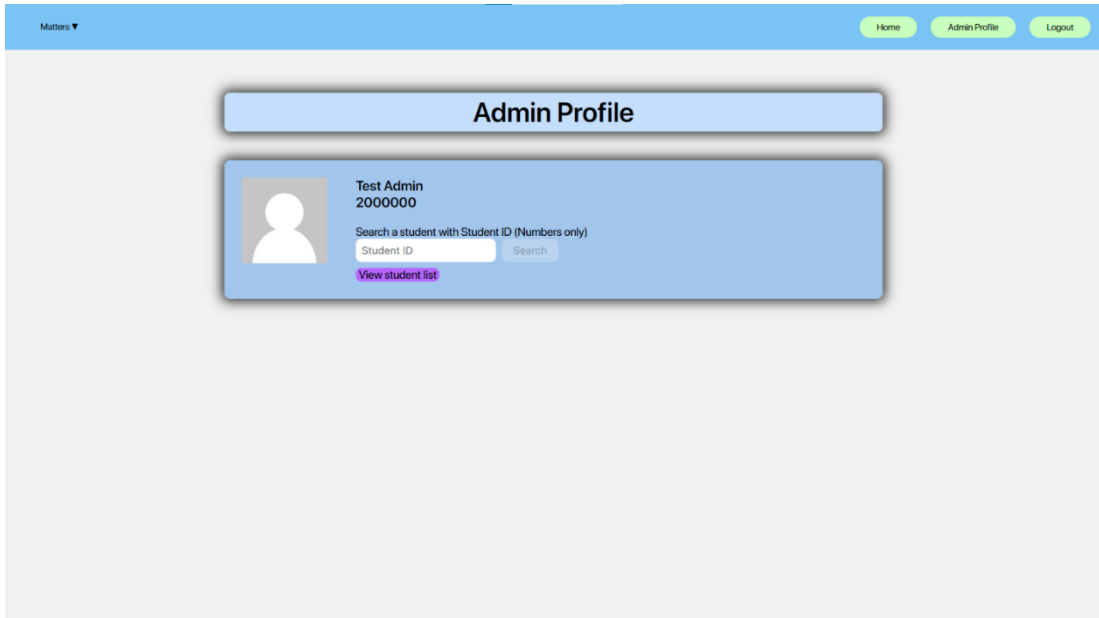


Figure 5-57: Admin Profile Screen

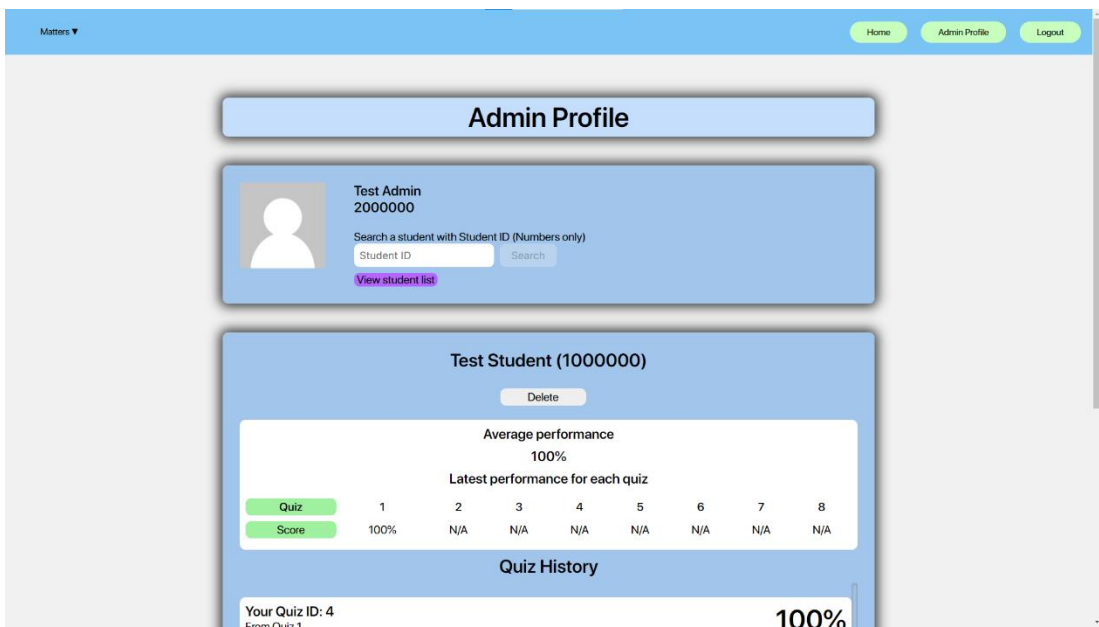


Figure 5-58: Upper Part of Admin Profile Screen with Searched Student

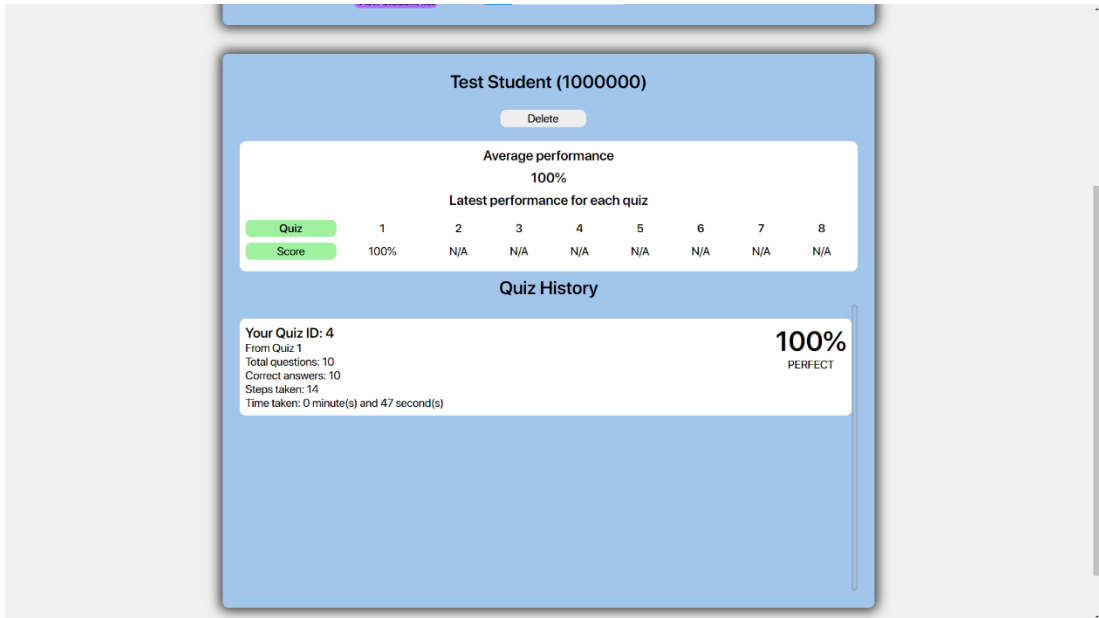


Figure 5-59: Bottom Part of Admin Profile Screen with Searched Student

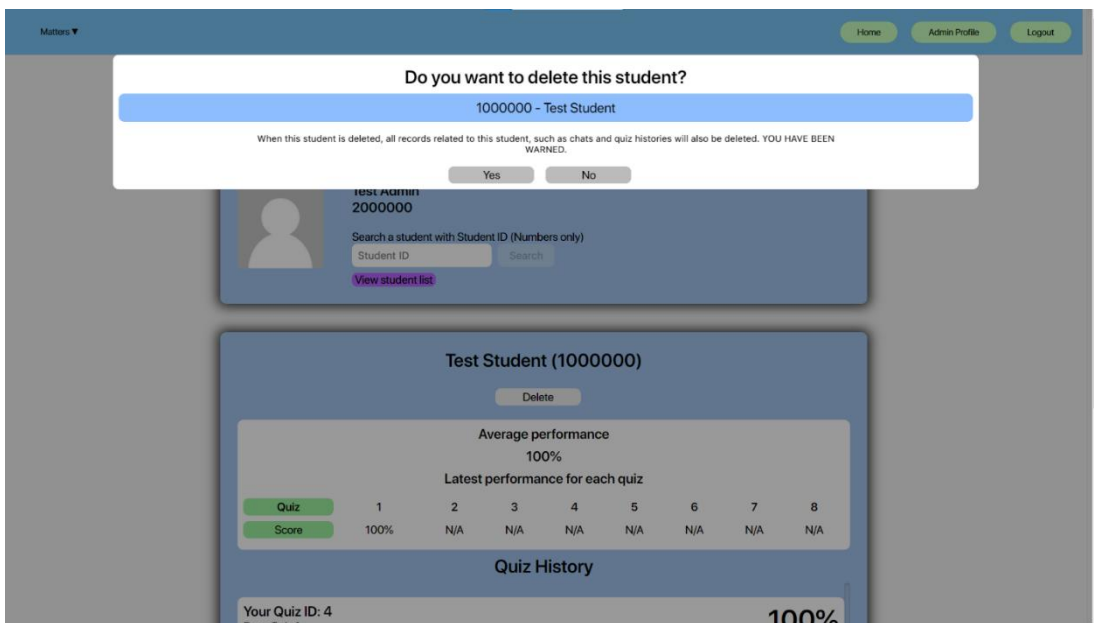
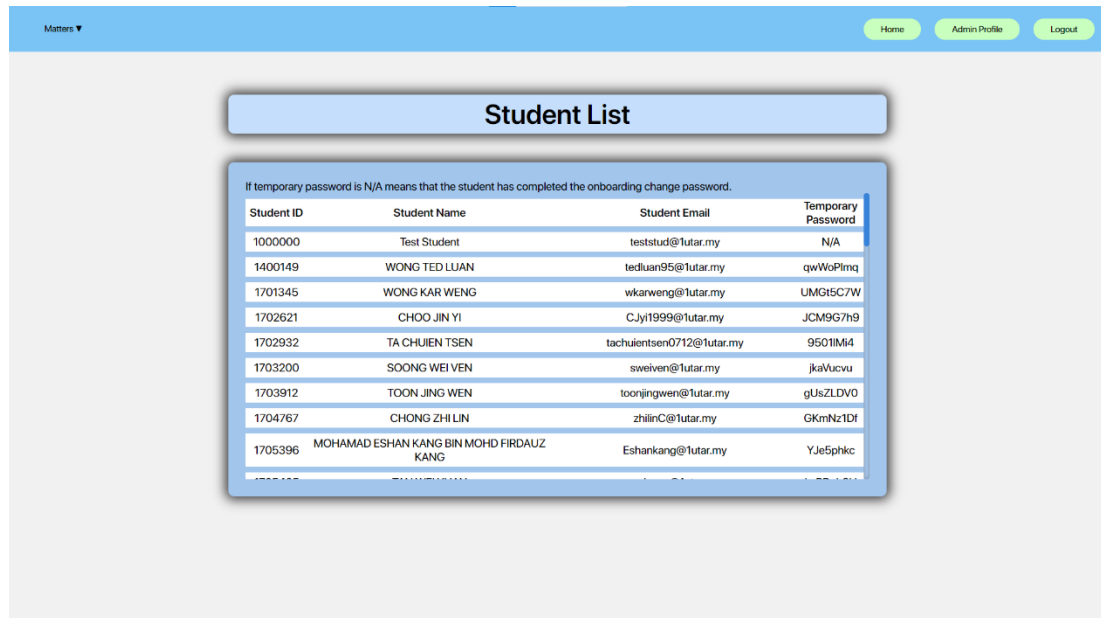


Figure 5-60: Admin Delete Student Modal Screen



Student ID	Student Name	Student Email	Temporary Password
1000000	Test Student	teststud@tutar.my	N/A
1400149	WONG TED LUAN	tedluan95@tutar.my	qwWoPlmq
1701345	WONG KAR WENG	wkarweng@tutar.my	UMGt5C7W
1702621	CHOO JIN YI	C.Jyi1999@tutar.my	JCM9G7n9
1702932	TA CHUIEN TSEN	tachuentsen0712@tutar.my	9501IMi4
1703200	SOONG WEI VEN	sweiven@tutar.my	jkaVucvu
1703912	TOON JING WEN	toonjingwen@tutar.my	gUsZLDV0
1704767	CHONG ZHILIN	zhilinC@tutar.my	GKmNz1DI
1705396	MOHAMAD ESHAN KANG BIN MOHD FIRDAUZ KANG	Eshankang@tutar.my	YJe5phkc

Figure 5-61: Admin Student List Screen

5.4.10 Code Playground Screen

Unlike the online code editor in the topic section, the online code editor in the code playground screen can let users to type and run their own Swift code as they want.

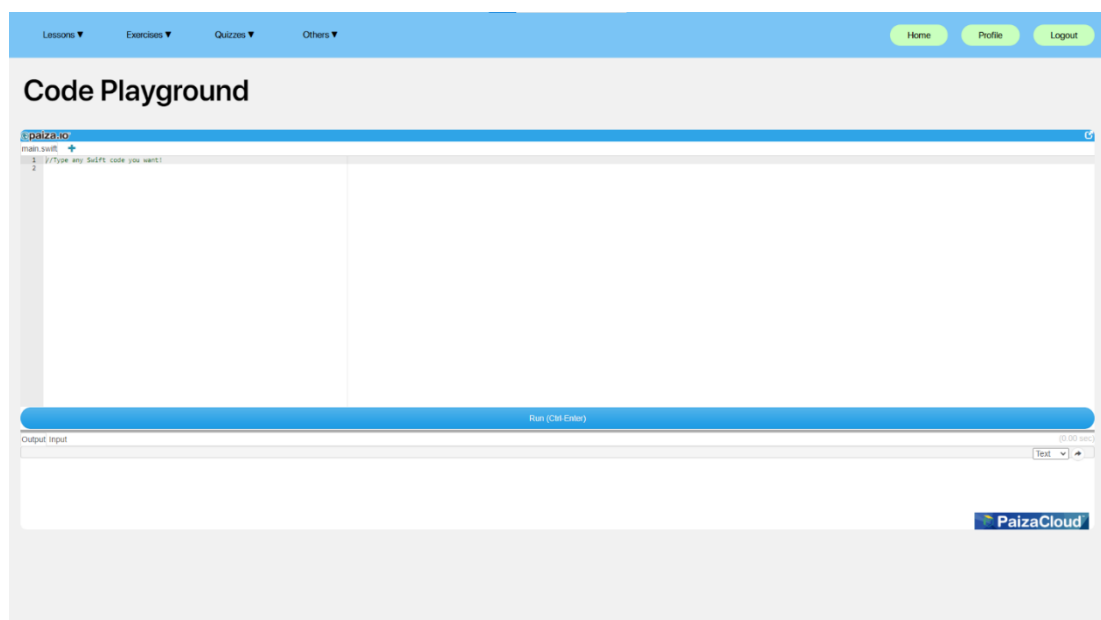


Figure 5-62: Code Playground Screen

5.4.11 Chat Box Screen

The chat box screen provides a group chat function for students and administrators to communicate with each other by sending chat texts. Students and administrators are also able to view a list of online users. For administrators, he/she may also delete any chat text sent by any users.

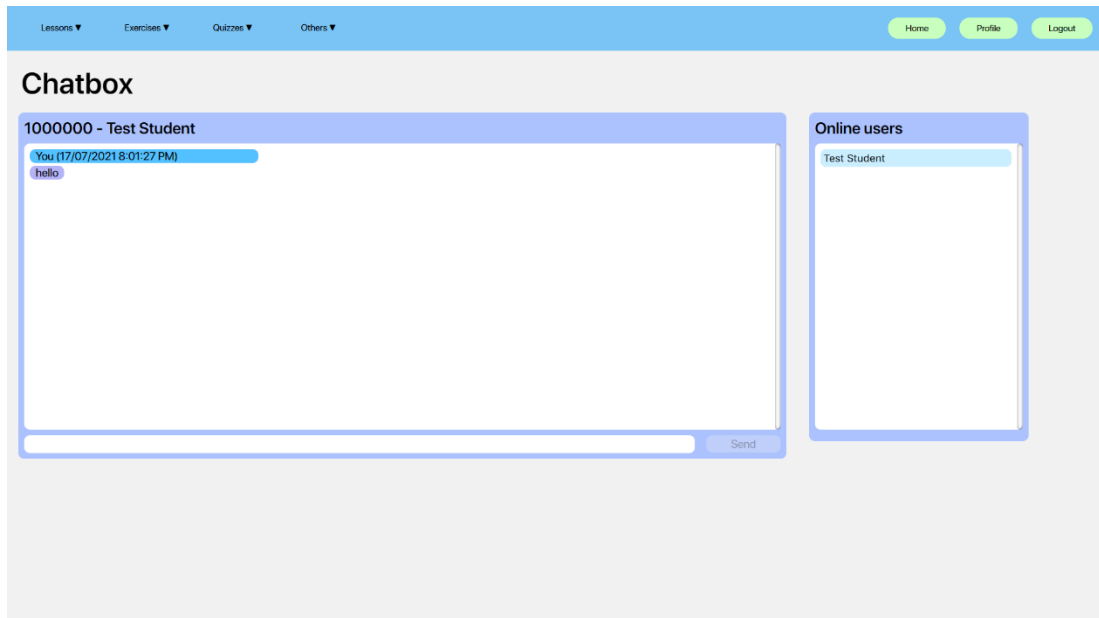


Figure 5-63: Student Chat Box Screen

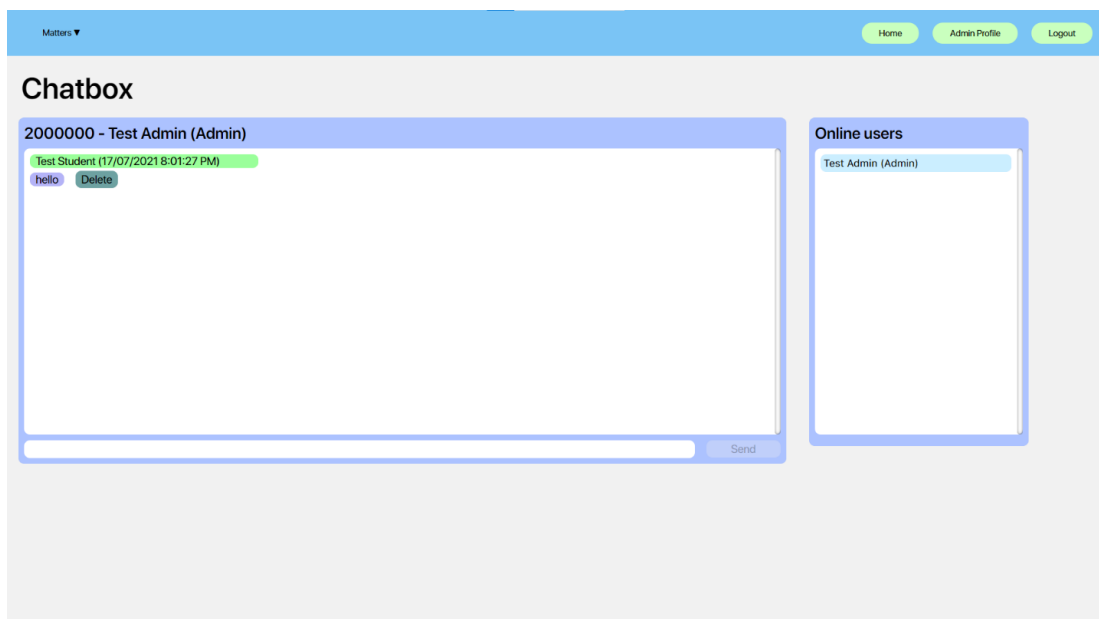


Figure 5-64: Admin Chat Box Screen

5.4.12 Register Student Screen

The register student screen is only available for administrators. The administrator may register a new student by key in the student ID, student name and student UTAR Email.

Matters ▾ Home Admin Profile Logout

Register Student

Student ID (Without "UEB")
Student ID

Student Name
Student Name

Student UTAR Email
Student UTAR Email

A random password will be generated automatically and send to their respective UTAR email. Students will be required to update their password when they login to the system for the first time.

REGISTER

Figure 5-65: Register Student Screen

5.4.13 Error Screen

If the user is trying to access a route that doesn't exist, the user will be redirected to error 404 screen. Besides, if the student who is yet to complete the onboard sessions is trying to access other student routes such as home screen, the student will be redirected to error 403 screen and vice versa.

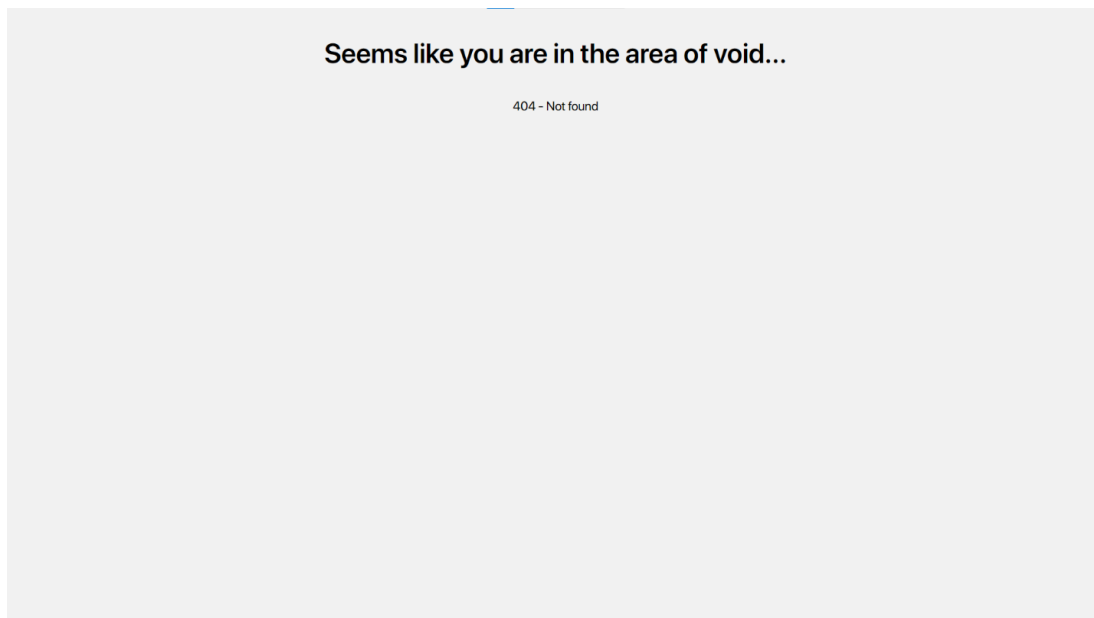


Figure 5-66: Error 404 Screen

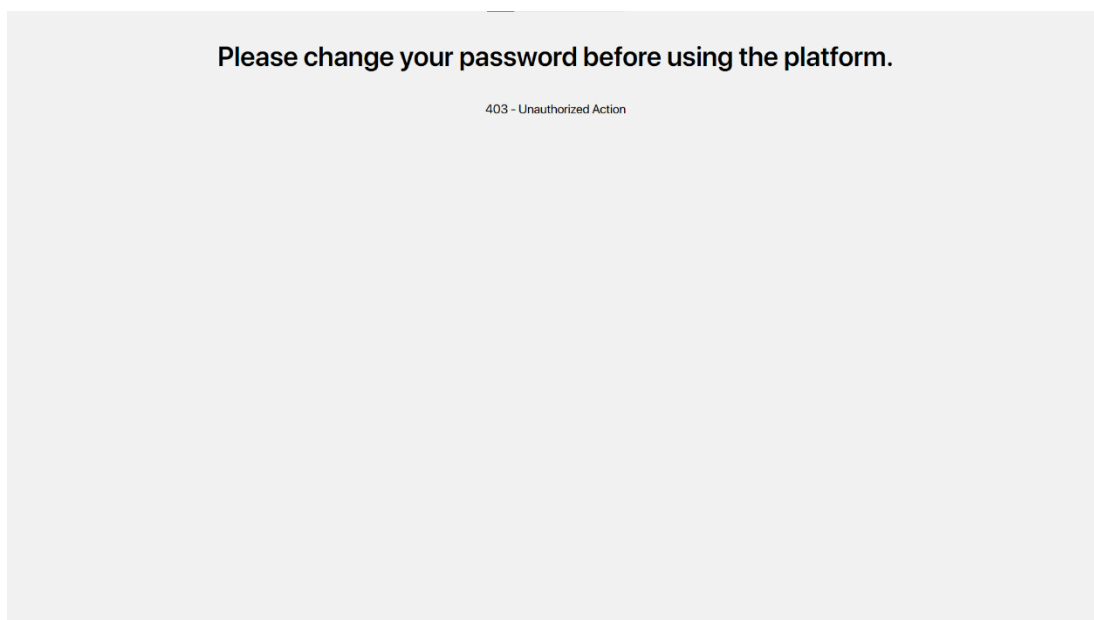


Figure 5-67: Error 403 Screen for New Student

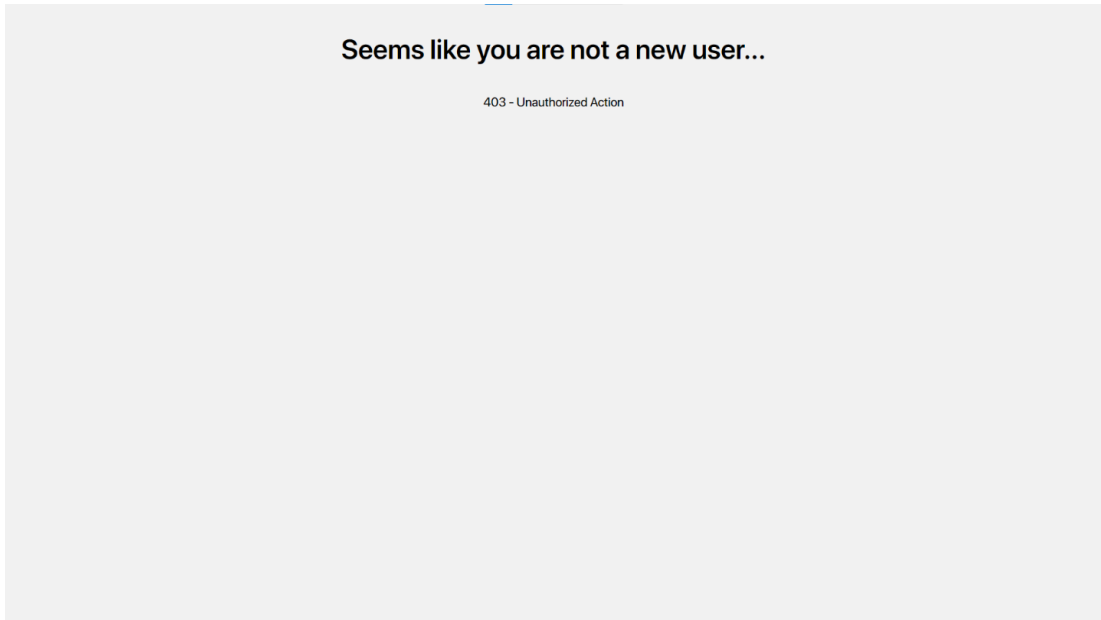


Figure 5-68: Error 403 Screen for Existing Student

CHAPTER 6

SYSTEM IMPLEMENTATION

The purpose of this chapter is to explain the code implementation for each module within the system. Section 6.1 is to provide description for the modules with respective functional and/or non-functional requirements and its associated users. Section 6.2 is to provide illustration for the implemented student and admin modules with associated code segments. Section 6.3 is to provide illustration for the implemented student-specific modules with associated code segments. Section 6.4 is to provide illustration for the implemented administrator-specific modules with associated code segments.

6.1 System Modules

The implemented system can be divided into many system modules based on user type and respective requirements, as shown as Table 6-1.

Table 6-1: System Module Table

Modules	Associated requirement(s)	Associated user(s)
Login	The system should allow the user to login the system.	Student and administrator
Reset Password	<p>The system should allow the user to reset password.</p> <p>The system should provide an interface to the user to reset password when the user login his/her account for the first time.</p> <p>The system should hash the password to prevent</p>	Student

	anyone except the user itself from retrieving the password.	
Logout	The system should allow the user to logout.	Student and administrator
Read Topic Lessons	The system should allow the user to read and follow Swift programming materials.	Student and administrator
Online Code Editor	The system should allow the user to edit codes and produce output in the embedded code compiler.	Student and administrator
Exercise	The system should provide non-graded exercises for users as a practice.	Student and administrator
Graded Quiz	<p>The system should provide graded quizzes after the user completes each topic.</p> <p>The system should allow the user to retake the graded quizzes with unlimited trials.</p> <p>The system should allow the user to view the results once the user has completed the graded quizzes.</p>	Student
Student Profile	The system should allow the user to view his/her basic statistics for overall	Student

	<p>performance.</p> <p>The system should allow the user to view his/her graded quiz attempt history.</p> <p>The system should obtain the latest score of the user when the user attempts the same quizzes more than once.</p>	
Administrator Profile	<p>The system should allow the administrator to monitor the student's progress.</p> <p>The system should allow the administrator to delete the user account.</p>	Administrator
Chat Box	<p>The system should provide a chat box for users to communicate with other users.</p> <p>The system should display the online users in the chat box.</p> <p>The system should only save the last 3 days of the conversation in the chat box.</p>	Student and administrator

Register Student	The system should allow the administrator to register the user account.	Administrator
Modify Course Content	The system should allow the administrator to modify the exercises, quizzes and tutorial briefings.	Administrator

6.2 Student and Administrator Modules

6.2.1 Login Module

Two types of logins in the login module were implemented for student and administrator respectively. Student login function was handled by Laravel’s default PHP trait which is “AuthenticatesUsers” and used by a controller class named “LoginController” whereas the admin login function was self-defined in the same controller class. In this section, only the “LoginController” controller class will be explained.

In the “LoginController” class, there are two self-defined methods for administrators which are “showAdminLoginForm”, “adminLogin”, and one default method for all users which are “redirectTo”. The “showAdminLoginForm” returns the login blade view for administrators, similar to “showLoginForm” for students in the “AuthenticatesUsers” trait. Then, “adminLogin” method will validate the input from administrator login blade view, generate a session for administrator and redirect the administrator to the intended route, or throw an exception with a message in the same login view if the credential or validation is invalid.

Once the user has logged into the system, “UserActivity” middleware class will insert the cache of last login time into administrator or student database table, and update the cache every second, until the user has logged out from the system. Figure 6-1 and Figure 6-2 shows the code segments of “LoginController” controller class and “UserActivity” middleware class respectively.

```

protected function redirectTo(){
    if (auth()->user()->isOnboard == false){
        return '/onboard';
    }
    return '/home';
}

/**
 * Create a new controller instance.
 *
 * @return void
 */
public function __construct()
{
    $this->middleware('guest')->except('logout');
    $this->middleware('guest:admin')->except('logout');
}

public function showAdminLoginForm()
{
    return view('auth.login', ['url' => 'admin']);
}

public function adminLogin(Request $request)
{
    $this->validate($request, [
        'adminID' => 'required|numeric',
        'password' => [
            'required',
            'string',
            'min:8',
            'regex:[0-9]/',
            'regex:[a-zA-Z]/',
        ],
    ], 1);

    if (Auth::guard('admin')->attempt(['adminID' => $request->adminID, 'password' => $request->password])) {
        return redirect()->intended('/admin/home');
    }

    throw ValidationException::withMessages([
        'adminID' => [trans('auth.failed')],
    ]);

    return back()->withInput($request->only('adminID'));
}

```

Figure 6-1: Code segment of the “LoginController” controller class

```

public function handle($request, Closure $next)
{
    if(Auth::check()){
        if (Auth::guard('web')->check()) {
            Cache::put('student-is-online-' . Auth::user()->studentID, true);
            // last seen
            Student::where('studentID', Auth::user()->studentID)->update(['last_seen' => (new \DateTime())->format("Y-m-d H:i:s")]);
        } else{
            Cache::put('admin-is-online-' . Auth::guard('admin')->user()->adminID, true);
            // last seen
            Admin::where('adminID', Auth::user()->adminID)->update(['last_seen' => (new \DateTime())->format("Y-m-d H:i:s")]);
        }
    }

    return $next($request);
}

```

Figure 6-2: Code segment of the “UserActivity” middleware class

6.2.2 Logout Module

The logout method from “AuthenticatesUsers” trait was slightly modified during the implementation. When the user sends a logout request, the method will pull out the cache of the last login time of the user based on their user type. Then, the user session will become invalidated status and the token is regenerated. After that, the

user will be redirected to the default system view which is the welcome view. Figure 6-3 shows the code segment of the logout method.

```
public function logout(Request $request)
{
    if(Auth::guard('admin')->check()){
        Cache::pull('admin-is-online-' . Auth::guard('admin')->user()->adminID,
            true);
    }

    if(Auth::guard('web')->check()){
        Cache::pull('student-is-online-' . Auth::user()->studentID, true);
    }

    $this->guard()->logout();

    $request->session()->invalidate();

    $request->session()->regenerateToken();

    if ($response = $this->loggedOut($request)) {
        return $response;
    }

    return $request->wantsJson()
        ? new JsonResponse([], 204)
        : redirect('/');
}
```

Figure 6-3: Code segment of the logout method from the “AuthenticatesUsers” trait

6.2.3 Read Topic Lesson Modules

In the “Topic” components from ReactJS framework, Axios API with GET method was used to fetch related topic sections data from database to the view. The fetched data are stored into many array-type component state objects in the “loadSectionList” method with React lifecycle, and mapped out all the stored data as HTML elements in the render method. Each fetch topic section data includes section name, up to four explanations, up to four code samples, and an optional online code editor link. Figure 6-4 to Figure 6-5 shows the code segments for lesson modules.

```
loadSectionList(){
  var thePath = window.location.pathname;
  var theID = thePath.split('/').pop();
  axios.get(window.location.origin + '/api/relatedTopic/' + theID).then(
    (response) => {
      let {isEditorOpen, editorList} = this.state;

      for (var i = 0; i < response.data.length; i++){
        isEditorOpen[i] = false;
        editorList[i] = "";
      }
      this.setState({
        relatedSectionList:response.data,
        isEditorOpen,
        editorList,
      });
    })
}
```

Figure 6-4: Code segment of the related topic section data fetching method via Axios API

```

contentPara = this.state.relatedSectionList.map((relatedSection,
sectionIndex) => {

    let editorButton;
    let editorContent = this.state.editorList[sectionIndex];

    let firstCodeEx;
    let secondCodeEx;
    let thridCodeEx;
    let fourthCodeEx;

    if(relatedSection.sectionCode1 != null){
        firstCodeEx = <div className="lessonContentCode"
id="codeSample">
            <p>{relatedSection.sectionCode1}</p>
        </div>
    }
    if(relatedSection.sectionCode2 != null){
        secondCodeEx = <div className="lessonContentCode"
id="codeSample">
            <p>{relatedSection.sectionCode2}</p>
        </div>
    }
    if(relatedSection.sectionCode3 != null){
        thridCodeEx = <div className="lessonContentCode"
id="codeSample">
            <p>{relatedSection.sectionCode3}</p>
        </div>
    }
    if(relatedSection.sectionCode4 != null){
        fourthCodeEx = <div className="lessonContentCode"
id="codeSample">
            <p>{relatedSection.sectionCode4}</p>
        </div>
    }
}

```

Figure 6-5: Upper part code segment of mapping stored data into HTML components in render method

```

if(relatedSection.editorLink != null){

    if(this.state.isEditorOpen[sectionIndex] != true){
        editorButton = <button className="tryEditor" onClick={()=>
            this.codeEditor(sectionIndex, relatedSection.editorLink)}
            >Click to open the code editor for this section!</button>
    } else {
        editorButton = <button className="tryEditor" onClick={()=>
            this.codeEditor(sectionIndex, relatedSection.editorLink)}
            >Click to close the code editor for this section!</button>
    }
}

return(
    <div key={relatedSection.sectionID}
        className="lessonContentPara" id={relatedSection.sectionName}>
        <h3>{relatedSection.sectionName}</h3>
        {this.state.isAdmin == true ?
        <div className="adminButtons">
            <EditSectionModal relatedSection={relatedSection}
                updateSectionList={this.updateSectionList} topicList=
                {this.state.topicList}/>
            <DeleteSectionModal relatedSection={relatedSection}
                updateSectionList={this.updateSectionList}/>
        </div> : ""}

        <p>{relatedSection.sectionText1}</p>
        {firstCodeEx}
        <p>{relatedSection.sectionText2}</p>
        {secondCodeEx}
        <p>{relatedSection.sectionText3}</p>
        {thridCodeEx}
        <p>{relatedSection.sectionText4}</p>
        {fourthCodeEx}
        {editorButton}
        {editorContent}
    </div>
)

```

Figure 6-6: Bottom part code segment of mapping stored data into HTML components in render method

6.2.4 Online Code Editor Modules

Some of the topic sections contain a link to open embedded Paiza.io online code editor. In default, the code editor open status for each topic section with code editor link is set into false. If the open code editor button is pressed, the code editor open

status will be set into true, and an iframe element with the editor link will be created and loaded. If the same open code editor button is pressed again, the code editor's open status will become false, as well as the iframe element will be cleared. Figure 6-7 shows the code segment for the online code editor module.

```
codeEditor(sectionIndex, editorLink){
  let {isEditorOpen, editorList} = this.state;

  if(isEditorOpen[sectionIndex] == false){
    isEditorOpen[sectionIndex] = true;
    editorList[sectionIndex] =
    <div className="editorWrap">
      <div className="blockExternal"></div>
      <iframe src={`${editorLink}`} scrolling="no" seamless="seamless"
      frameBorder="0"></iframe>
    </div>;

    this.setState({
      isEditorOpen,
      editorList,
    });
  } else {
    isEditorOpen[sectionIndex] = false;
    editorList[sectionIndex] = "";

    this.setState({
      isEditorOpen,
      editorList,
    });
  }
}
```

Figure 6-7: Code segment of handling embedded online code editor opening or closing

6.2.5 Exercise Module

Similar to the Read Topic Lesson Module, model relationship, data fetching via API and array mapping are implemented to show all exercises content with the same topic in the view, such as question text and at least one answer template. Nevertheless, the “loadExerciseList” method also sets lots of boolean-value arrays to store each question button's disable status, and a string-value array to store correct answers for the user to check.

When the user clicks the check answer button, the submit button and input will be disabled. and the input value will be assigned to the correct answer based on

the selected question. When the user clicks the check answer button from the same question again, the system will enable the input field and submit button, and clear the input from the correct answer to blank.

For answer submission, a method is called to check if each answer is correct or wrong. If any answer from the question is wrong, an error message will be generated to the user. If all answers are correct, the check answer button, submit button, and all input will be hidden and disabled, and a message will be generated to the user. Figure 6-8 and Figure 6-9 show code segments for exercise module.

```

checkAns(questionIndex, ans1, ans2, ans3){
  let tempIsDisArr = this.state.isDisableList.slice();
  let tempDisArr = this.state.disableList.slice();
  let tempCheArr = this.state.checkAnsValue.slice();
  let tempSubBtnArr = this.state.submitButtonHide.slice();
  let tempOutMsgArr = this.state.outputMsg.slice();

  if(tempIsDisArr[questionIndex] != true){
    tempCheArr[questionIndex] = [ans1, ans2, ans3];
    tempDisArr[questionIndex] = [true, true, true];
    tempIsDisArr[questionIndex] = true;
    tempSubBtnArr[questionIndex] = true;
    tempOutMsgArr[questionIndex] = "You are checking the correct answer,
    submit button will be disabled. Click again the check answer button to
    answer the question.";
    this.setState({
      disableList:tempDisArr,
      isDisableList:tempIsDisArr,
      checkAnsValue:tempCheArr,
      submitButtonHide:tempSubBtnArr,
      outputMsg:tempOutMsgArr,
    })
  } else {
    tempCheArr[questionIndex] = ["", "", ""];
    tempDisArr[questionIndex] = [false, false, false];
    tempIsDisArr[questionIndex] = false;
    tempSubBtnArr[questionIndex] = false;
    tempOutMsgArr[questionIndex] = "";
    this.setState({
      disableList:tempDisArr,
      isDisableList:tempIsDisArr,
      checkAnsValue:tempCheArr,
      submitButtonHide:tempSubBtnArr,
      outputMsg:tempOutMsgArr,
    })
  }
}
}

```

Figure 6-8: Code segment of checking correct answer from a question


```

submitAns(questionIndex, ans1, ans2, ans3){
  let inputAns = this.state.checkAnsValue[questionIndex];
  let {submitButtonHide, checkButtonHide, disableList, outputMsg} = this.
  state;

  let isCorrect = [];

  if(ans1 != null){
    if(inputAns[0] == ans1){
      isCorrect[0] = true;
    } else {
      isCorrect[0] = false;
    }
  }
  if(ans2 != null){
    if(inputAns[1] == ans2){
      isCorrect[1] = true;
    } else {
      isCorrect[1] = false;
    }
  }
  if(ans3 != null){
    if(inputAns[2] == ans3){
      isCorrect[2] = true;
    } else {
      isCorrect[2] = false;
    }
  }

  if(isCorrect[0] != false && isCorrect[1] != false && isCorrect[2] != false){
    disableList[questionIndex] = [true, true, true];
    submitButtonHide[questionIndex] = true;
    checkButtonHide[questionIndex] = true;
    outputMsg[questionIndex] = "All of your answers are correct!"
    this.setState({submitButtonHide, checkButtonHide, disableList,
    outputMsg});
  } else {
    outputMsg[questionIndex] = "Wrong answer for some blank(s), please try
    again!"

    this.setState({outputMsg});
  }
}

```

Figure 6-9: Code segment of submitting answers from a question

6.2.6 Chat Box Module

Axios API with GET method fetched all chat data and stored it into an array in the React frontend. The stored array will be mapped to render each object into HTML elements in the render method. During the rendering, a method called “sliceDate” is called to convert the date and time format from MySQL format into readable 12-hour format. Besides, the chat header color will be different based on the user type.

If the user sends a chat to the system, the “handleSubmit” will be called to pack the text, the converted date and time, and user ID together, and send the objects into the database via Axios API with POST method.

If the administrator clicks the delete chat button to delete any chat, a method will be called to send a chat ID to the controller via Axios API with DELETE method.

To show the user who is online, the “onlineStatus” method from “ChatController” will be triggered to search all the user’s cache by using a for loop. If the method detects the user has an online status cache, the user’s name will be added into an array and returned to the front end. Figure 6-10 to Figure 6-14 shows the code segments for the chat box module.

```
sliceDate(dateTime){
  let year = dateTime.slice(0,4);
  let month = dateTime.slice(5,7);
  let day = dateTime.slice(8,10);
  let hour = parseInt(dateTime.slice(11,13));
  let minute = dateTime.slice(14,16);
  let second = dateTime.slice(17,19);
  let meridiem = "";

  if (hour >= 12 && hour <=23){
    if(hour > 12){
      hour = hour - 12;
    }
    meridiem = "PM"
  } else {
    if (hour == 0){
      hour = hour + 12;
    }
    meridiem = "AM"
  }

  hour = hour.toString();
  var newDateFormat = day + "/" + month + "/" + year + " " + hour + ":" +
  minute + ":" + second + " " + meridiem;
  return newDateFormat;
}
```

Figure 6-10: Code segment of converting MySQL date time format into readable format.

```

function onlineStatus()
{
    $students = Student::all();
    $admins = Admin::all();
    $onlineList = [];
    foreach ($students as $student) {
        if (Cache::has('student-is-online-' . $student->studentID)){
            array_push($onlineList, $student->studentName);
        }
    }
    foreach ($admins as $admin) {
        if (Cache::has('admin-is-online-' . $admin->adminID)){
            array_push($onlineList, $admin->adminName . ' (Admin)');
        }
    }
    return $onlineList;
}

```

Figure 6-11: Code segments of finding online user in “ChatController” controller class

```

let chats = this.state.chatList.map((chat) => {

    let newDateFormat = this.sliceDate(chat.chatDateTime);
    //Rearrange date time format for display purpose

    if(chat.studentID != null){
        return(
            <div key={chat.chatID} className="chatBubble" id={chat.
                studentID == this.state.authenticatedStudentID ? "ownChat":
                "otherChat"}>
                <div className="chatUser">
                    {chat.studentID == this.state.authenticatedStudentID ?
                    <p>You ({newDateFormat})</p> :
                    <p>{chat.student.studentName} ({newDateFormat})</p>}
                </div>
                <div className="chatContent">
                    <p>{chat.chatText}</p>
                </div>
            </div>
        );
    }

    if (chat.adminID != null){
        return(
            <div key={chat.chatID} className="chatBubble" id="adminChat">
                <div className="chatUser">
                    <p>{chat.admin.adminName} ({newDateFormat})</p>
                </div>
                <div className="chatContent">
                    <p>{chat.chatText}</p>
                </div>
            </div>
        );
    }
});

```

Figure 6-12: Code segments of mapping stored chat data into HTML elements

```

handleSubmit(e){
  e.preventDefault();
  let chatText = this.state.holdingChatText;
  let chatDateTime = this.convertDate();
  let studentID = this.state.authenticatedStudentID;
  axios.post(window.location.origin + '/api/chat', {
    chatText, chatDateTime, studentID,
  }).then((response) => {
    this.setState({
      holdingChatText: "",
    })
  });
}

```

Figure 6-13: Code segment of sending chats into database via Axios API

```

deleteChat(chatID){
  axios.delete(window.location.origin + '/api/chat/' + chatID);
}

```

Figure 6-14: Code segment of delete chat based on chat ID via Axios API

6.3 Student-only Modules

6.3.1 Reset Password Module

Two types of reset password functions were implemented which are normal reset password and onboarding reset password. The normal reset password function was handled by a controller class named “ForgotPasswordController”. Once the student clicks the forgot password button, the “showResetPasswordForm” method will return the reset password view to the student. Once the student fills in all necessary information in the view and clicks the reset button, this data will be passed into the “submitResetPasswordForm” method. This method will validate these data. Next, the student will be searched based on email to check the student’s onboarding status. If the student is a new student, the system will block the student from resetting the password, otherwise, the student will be redirected to the login page and the password will be updated.

The onboarding reset password function forces every student who is using the system for the first time to reset their password. Once the new student key in their new password and click the submit button, the system will check the password length and regular expression. If all of the validations are correct, the student password will be updated and hashed by using Axios API with PUT method. Figure 6-15 shows the code segment of normal reset password function whereas Figure 6-16 to Figure 6-18 shows the code segment of onboarding reset password.

```

public function showResetPasswordForm() {
    return view('auth.passwords.forgetPasswordLink');
}

public function submitResetPasswordForm(Request $request)
{
    $request->validate([
        'email' => 'required|email|exists:students',
        'password' => [
            'required',
            'string',
            'min:8',
            'confirmed',
            'regex:/[0-9]/',
            'regex:[a-zA-Z]/',
        ],
        'password_confirmation' => 'required'
    ], [
        'email.exists' => 'This email is not exist from the system.',
        'password.regex' => 'Invalid format. The password should have a minimum of 8
        characters with combinations of letters and numeric characters.'
    ]);

    $student = Student::where('email', $request->email)->first();

    if($student->isOnboard == false){
        return back()->with('message', 'New students are not allowed to reset their
        password before proceeding onboarding.');
```

```

    } else {
        $student->update(['password' => Hash::make($request->password)]);
        return redirect('/login')->with('message', 'Your password has been reset
        successfully!');
```

```

    }
}

```

Figure 6-15: Code segments of “ForgotPasswordController” controller class

```

handleSubmit(e){
  e.preventDefault();
  let {password, confirmPassword} = this.state;
  if (password.length > 0){
    const regex1 = /^(?:[0-9]+[a-z]|[a-z]+[0-9])[a-z0-9]*$/i
    if(password.match(regex1)){
      if (password == confirmPassword){
        if (password.length >= 8){
          this.updatePassword();
        } else {
          this.setState({
            message:"Your password should at least 8 alphanumeric
            characters!",
            confirmPassword: "",
          });
        }
      } else if (password != confirmPassword) {
        this.setState({
          message:"Your password and confirm password are not match!",
          confirmPassword: "",
        });
      }
    } else {
      this.setState({
        message:"Your password should contains at least one letter and
        one numeric characters!",
      });
    }
  } else {
    this.setState({
      message:"Your password and confirm password should not be empty!",
    });
  }
}

```

Figure 6-16: Code segments of validating password format and length

```

updatePassword(){
  let {password} = this.state;
  axios.put(window.location.origin + '/api/student/updatepassword/' + this.
  state.id,{
    password,
  }).then((response) => {
    this.setState({
      message:"",
      isSuccessful: true,
    });
  });
}

```

Figure 6-17: Code segments of updating student's password via Axios API with PUT method

```
function updatePassword(Request $request, $studentID){
    $student = Student::findOrFail($studentID);

    $student->tempPassword = null;
    $student->password = Hash::make($request->password);
    $student->isOnboard = true;
    $student->save();
    return $student;
}
```

Figure 6-18: Code segments of “updatePassword” method in “StudentController” controller class

6.3.2 Graded Quiz Module

All quiz questions from the selected topic were fetched and stored into an array by using Axios API with GET method. However, all questions will be listed out one-by-one based on the next or previous button, instead of listing out at once with array mapping.

Once the student clicks the start button to do the quiz question, the timer function will be started to calculate the time taken until the student clicks the submit button. When the submit button is triggered, a method will be called to check the total number of answers with the total number of questions. If there is any question with an empty answer, the system will pop out an alert to the student until the student completes all questions. After that, the method will calculate the number of correct answers from the student’s answer, and a quiz record will be stored into the database with total number of questions, number of correct answers made by student, time taken, steps count, related topic and related student.

The system also calculates the marks that the student obtained from the quiz, and lists out all answers made by students. If the student answers the question correctly, the answer will be shown as green. Otherwise, the wrong answer made by the student will be shown as red and the correct answer will be shown as green. Figure 6-19 to Figure 6-23 shows the code segments for the graded quiz module.

```

startButtonClick(){
  this.setState({
    isReady: true,
    timeTaken: this.state.timeTaken,
    timerStart: Date.now() - this.state.timeTaken,
  });
  this.timer = setInterval(() => {
    this.setState({
      timeTaken: Date.now() - this.state.timerStart
    });
  }, 10);
}

```

Figure 6-19: Code segment of starting the quiz

```

submitButtonClick(){
  let {storedAns, correctAnsList, numCorrect, steps} = this.state;
  let tempConvAns = [];

  if (storedAns.includes(undefined) || storedAns.length == 0 || storedAns.length != correctAnsList.length){
    alert("You have some incomplete questions.");
  } else {
    clearInterval(this.timer);
    steps++;
    for (var i = 0; i < storedAns.length; i++){
      switch (storedAns[i]){
        case '0':
          tempConvAns[i] = 'a';
          break;
        case '1':
          tempConvAns[i] = 'b';
          break;
        case '2':
          tempConvAns[i] = 'c';
          break;
        case '3':
          tempConvAns[i] = 'd';
          break;
      }
    }

    for (var i = 0; i < tempConvAns.length; i++){
      if(tempConvAns[i] == correctAnsList[i]){
        numCorrect++;
      }
    }

    this.setState({
      numCorrect,
      isSubmit: true,
      steps,
    });
  }
}

```

Figure 6-20: Code segments of submitting the answers from the quiz


```
storeQuizHistory(){
  let totalQues = this.state.relatedQuizList.length;
  let {numCorrect, steps, timeTaken} = this.state;
  let topicID = this.state.topicTitle.topicID;
  let studentID = this.state.authenticatedStudentID;
  axios.post(window.location.origin + '/api/quizhistory', {
    totalQues, numCorrect, steps, timeTaken, topicID, studentID,
  });
}
```

Figure 6-21: Code segments of storing quiz history once the quiz is completed

```
let resultCalculation = (correct, total) => {
  let percentage = Math.round((correct / total) * 100);
  return percentage;
}
```

Figure 6-22: Code segments of calculating quiz result into percentage form

```

let avaOptionList = [];
for (var i = 0; i < 4; i++){
  avaOptionList[i] = eval("question.option" + (i+1));
}

let avaOption = avaOptionList.map((option, oIndex) => {
  if(option != null){
    if(tempConvAns[qIndex] == oIndex){
      return(
        <li key={option} className="quizOptionsList"
          id="correct">{option}</li>
      );
    }
    if (storedAns[qIndex] != tempConvAns[qIndex] &&
      storedAns[qIndex] == oIndex){
      return(
        <li key={option} className="quizOptionsList"
          id="wrong">{option}</li>
      );
    }
    return(
      <li key={option} className="quizOptionsList">
        {option}</li>
    );
  }
});

return(
  <TabPanel>
    <p id="legend">Question Text</p>
    <p id="codesegment">
      {question.quizText}
    </p>
    <p id="legend">Your Answers (Correct answer will be
      highlighted as green color, else it will be highlighted
      as red color)</p>
    <ul className="quizOptions">
      {avaOption}
    </ul>
  </TabPanel>
);

```

Figure 6-23: Code segments of showing each question answer after completing the quiz

6.3.3 Student Profile Module

When the student accesses the profile function, the student ID, student name and student email will be loaded to the student via Axios API with GET method. The quiz histories related to the student will also be loaded and stored into an array. Array reverse map is implemented to make the latest quiz record at the top position. For the student's performance, if the student has more than one quiz record from the same quiz, the system will only fetch the score from the latest attempt. An average performance from all attempted quizzes will be calculated and shown to the student. Figure 6-24 to Figure 6-26 shows the code segments of the student profile module.

```

loadAuthenticatedUser(){
  const authUserID = $('#studentpage').attr("authUserID");
  axios.get(window.location.origin + '/api/student/' + authUserID).then(
    (response) => {
      this.setState({
        authenticatedUser: response.data,
      });
    });
}

```

Figure 6-24: Code segment of loading student credentials based on student ID

```

let histories = this.state.quizHistoryList.reverse().map(
  (relatedHistory) => {

    let quizpercentage = Math.round((relatedHistory.numCorrect /
    relatedHistory.totalQues) * 100);
    let perfectText;
    if(quizpercentage == 100){
      perfectText =
        <p>PERFECT</p>
    }
    return(
      <div className="historylist" key={relatedHistory.quizHistoryID}>
        <div className="historydet">
          <h2>Your Quiz ID: {relatedHistory.quizHistoryID}</h2>
          <p>From Quiz {relatedHistory.topicID}</p>
          <p>Total questions: {relatedHistory.totalQues}</p>
          <p>Correct answers: {relatedHistory.numCorrect}</p>
        </div>
        <div className="historyperc">
          <h2>{quizpercentage}%</h2>
          {perfectText}
        </div>
      </div>
    );
  });

```

Figure 6-25: Code segment of mapping related quiz histories in reverse order

```

if(this.state.topicLoaded && this.state.historyLoaded){
  let {quizHistoryList, topics} = this.state;
  let quizLatest = [];
  let temp = [];
  let copyList = quizHistoryList.slice();

  for (var i = 0; i < topics.length; i++){
    temp[i] = copyList.filter(list => list.topicID == topics[i].topicID)
    ; //based on each topic, starting from topic 1 (quiz 1), filter the
    records into sub array
  }

  for (var j = 0; j < temp.length; j++){
    temp[j].sort((a,b) => parseFloat(a.quizHistoryID) - parseFloat(b.
    quizHistoryID)); // sort the filtered array for each topic based on
    the latest record instead of the highest mark.
    for (var k = 0; k < temp[j].length; k++){
      quizLatest[j] = Math.round((temp[j][k].numCorrect / temp[j][k].
      totalQues) * 100); //calculate percentage based on the latest
      record
    }
  }

  let totalScore = 0;
  let totalCount = 0;
  let averageScore;
  for (var a = 0; a < quizLatest.length; a++){
    if(quizLatest[a] != null){
      totalScore = totalScore + quizLatest[a];
      totalCount = totalCount + 1;
    }
  }

  if (totalCount != 0){
    averageScore = Math.round(totalScore / totalCount);
  } else {
    averageScore = 0;
  }

  let eachQuizScore = this.state.topics.map((topic, index) => {
    let tempData = "N/A";
    if (quizLatest[index] != null){
      tempData = quizLatest[index] + "%";
    }
    return(
      <td key={topic.topicID}>{tempData}</td>
    );
  });
}

```

Figure 6-26: Code segment of finding latest score from the same quizzes and calculating average performance

6.4 Administrator-only Modules

6.4.1 Administrator Profile Module

The administrator profile is almost the same as the student profile, but the administrator provides a search function to search a student's credential, quiz histories and overall performance. Axios API with GET method will be used to find a student from the database based on the input student ID. If the student ID doesn't exist from the database, the API will catch an error and generate a pop-up alert to the administrator. Otherwise, all of the related student data will be fetched and stored into an array.

On the other hand, when the administrator wants to delete a searched student account by clicking the delete button, a modal view will be prompted to the administrator to confirm the action, in order to prevent any misoperation. If the administrator clicks the "Yes" button to confirm, the system will delete the student account, including all quiz histories and chats related to this student via Axios API with DELETE method. The modal view will automatically disappear. Figure 6-27 to Figure 6-30 shows the code segment of the administrator profile module.

```

findStudent(studentID){
  axios.get(window.location.origin + '/api/student/' + studentID).catch(
  (error) => {
    if(error.response.status == "404"){
      alert("The student ID you key in is not exist.");
    }
    this.setState({
      loadedStudent: {
        studentID: "",
        studentName: "",
        email: "",
      },
      loading: false,
      quizHistoryList: [],
      historyLoaded: false,
    });
  });
  }).then((response) => {
    this.setState({
      loadedStudent: response.data,
    });
  });
  axios.get(window.location.origin + '/api/quizhistory/' + studentID).then(
  (response) => {
    this.setState({
      quizHistoryList: response.data,
      historyLoaded: true,
      loading: false,
      searchID: "",
    });
  });
});
}

```

Figure 6-27: Code segment of finding student data based on student ID

```

render(){
  let {deleteStudentModal} = this.state;
  let {loadedStudent} = this.props;
  return(
    <div>
      <button id="delete" onClick={this.toggleDeleteStudentModal}>Delete</button>
      <Modal isOpen={deleteStudentModal} toggle={this.props.toggleDeleteStudentModal}>
        <ModalHeader toggle={this.props.toggleDeleteStudentModal}>Do you want to delete this student?</ModalHeader>

        <ModalBody>
          <div className="deleteContent">
            <p>{loadedStudent.studentID} - {loadedStudent.studentName}</p>
          </div>
        </ModalBody>

        <ModalFooter>
          <p>When this student is deleted, all records related to this student, such as chats and quiz histories will also be deleted. YOU HAVE BEEN WARNED.</p>
          <button id="confirmDelete" onClick={this.deleteStudent}>
            Yes
          </button>
          <button id="cancel" onClick={this.toggleDeleteStudentModal}>
            No
          </button>
        </ModalFooter>
      </Modal>
    </div>
  );
}

```

Figure 6-28: Code segments of delete student modal view

```

deleteStudent(studentID){
  axios.delete(window.location.origin + '/api/student/' + studentID).then(
    (response) => {
      alert("Student has been deleted successfully.")
      this.props.closeStudentDetails();
    }
  );
}

```

Figure 6-29: Code segments of deleting student via Axios API with DELETE method

```

function deleteStudent($studentID){
  $student = Student::findOrFail($studentID);
  $student->delete();
  return 204;
}

```

Figure 6-30: Code segments of “deleteStudent” method in “AdminController” controller class

6.4.2 Register Student Module

In the initial state, all existing students from the database will be loaded via Axios API with GET method for further verification. When the administrator clicks the register button after filling in the new student credentials such as new student ID, new student name and new student email, two methods will be called to check the email format and the duplication of existing student accounts. When the input email follows the UTAR student email format (@lutar.my) and there’s no duplication of any credentials from existing students, a new student data will be added into the database with randomly generated and hashed password via Axios API with POST method. Otherwise error messages will be shown to the administrator. Figure 6-31 to Figure 6-35 shows the code segment of the register student module.

```
loadExistingStudent(){
  axios.get(window.location.origin + '/api/students').then((response) => {
    this.setState({
      existingStudentList: response.data,
    })
  });
}

componentDidMount(){
  this.loadExistingStudent();
}
```

Figure 6-31: Code segment of loading existing students via Axios API with GET method and React lifecycle.

```

checkEmailFormat(email){
  let emailRegex = /^((([^<>()\\[\]\\.,:;@"]+)(\\.[^<>()\\[\]\\.,:;@"]+)*)(("[.
+"))@((\\[[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\.[0-9]{1,3}\\))|(([a-zA-Z\\-0-9]+\\.)
)+[a-zA-Z]{2,}))$/;

  let emailCorrect = true;
  if(emailRegex.test(email)){
    if(email.indexOf("@utar.my", email.length - "@utar.my".length) !== -1){
      this.setState({
        emailErrorMsg: "",
      });
    } else {
      emailCorrect = false;
      this.setState({
        emailErrorMsg: "This email is not a UTAR student email!",
      });
    }
  } else {
    emailCorrect = false;
    this.setState({
      emailErrorMsg: "This email format is invalid!",
    });
  }

  return emailCorrect;
}

```

Figure 6-32: Code segment of checking student Email format

```

checkExistingAccount(studentID, studentName, email){
  let {existingStudentList} = this.state;
  let notExist = [true, true, true]; //first is for ID, second is for Name,
  third is for utar email
  for(var i = 0; i < existingStudentList.length; i++){
    if(studentID == existingStudentList[i].studentID){
      notExist[0] = false;
    }
    if(studentName == existingStudentList[i].studentName){
      notExist[1] = false;
    }
    if(email == existingStudentList[i].email){
      notExist[2] = false;
    }
  }

  if(notExist.includes(false)){
    this.setState({
      existingErrorMsg: "The student ID/name/UTAR email is already exist!",
    });
  } else {
    this.setState({
      existingErrorMsg: "",
    });
  }

  return notExist;
}

```

Figure 6-33: Code segment of checking duplication of credentials with existing students


```

handleSubmit(e){
  e.preventDefault();

  let {studentID, studentName, email, processingMsg} = this.state;
  let isUTAREmail = this.checkEmailFormat(email);
  let allNotExist = this.checkExistingAccount(studentID, studentName, email);

  if (isUTAREmail == true && allNotExist.every(Boolean)){
    processingMsg = "Registering...Please wait...",
    this.setState({
      processingMsg,
    });
    axios.post(window.location.origin + '/api/addStudent', {
      studentID, studentName, email,
    }).then((response) => {
      alert("You have successfully register a student.");
      this.loadExistingStudent();
      this.setState({
        processingMsg: "",
        emailErrorMsg: "",
        existingErrorMsg: "",
        studentID: "",
        studentName: "",
        email: "",
      });
    });
  }
}
}

```

Figure 6-34: Code segment of adding new student via Axios API with POST method

```

function registerAStudent(Request $request){

  $randomPassword = Str::random(8);
  $student = new Student;
  $student->studentID = $request->studentID;
  $student->studentName = $request->studentName;
  $student->email = $request->email;
  $student->tempPassword = $randomPassword;
  $student->password = Hash::make($randomPassword);
  $student->isOnboard = false;

  $student->save();

  return $student;
}

```

Figure 6-35: Code segment of “registerAStudent” method in “AdminController”
controller class

6.4.3 Modify Course Content Module

Administrators may modify a topic name, topic section, exercise question or graded quiz question by performing adding, editing, or deleting methods. As the implementation of modifying course content is similar for topic section and quiz, therefore only implementation of modifying exercise questions will be explained as an example.

When the add question button is clicked, a modal view will appear to let the administrator fill in the question text, correct answers, and related topics. The add button will be disabled until all compulsory inputs are filled in. Once the button is clicked, the Axios API with POST method will store all requested input into the exercise database. The edit question is similar to the add question function, but the selected question's current text, related topic, and correct answers will be loaded into the input field, and the method of Axios API will be the PUT method. For deleting an exercise question, a delete confirmation modal view will be shown to prevent misoperation. Once the delete action has been confirmed, the Axios API with DELETE method will delete the exercise question based on the given exercise ID. Figure 6-36 to Figure 6-40 shows the code segment of the modifying course content module.

```
addExercise(){
  axios.post(window.location.origin + '/api/addExercise', this.state.
  newExerciseData).then((response) => {
    this.props.updateExerciseList();
    this.setState({
      addExerciseModal: !this.state.addExerciseModal,
      newExerciseData:{
        exerciseText:"",
        blank1:"",
        blank2:"",
        blank3:"",
        topicID:"default",
      }
    });
  });
}
```

Figure 6-36: Code segment of add exercise question via Axios API with POST method

```

editExercise(exerciseID){
  axios.put(window.location.origin + '/api/exercise/' + exerciseID, this.state.
  editExerciseData).then((response) => {
    this.props.updateExerciseList();
    this.setState({
      editExerciseModal:!this.state.editExerciseModal,
      editExerciseData:{
        exerciseText:"",
        blank1:"",
        blank2:"",
        blank3:"",
        topicID:"default",
      }
    });
  });
}
}

```

Figure 6-37: Code segment of edit exercise question via Axios API with PUT method

```

deleteExercise(exerciseID){
  axios.delete(window.location.origin + '/api/exercise/' + exerciseID).then(
  (response) => {
    this.props.updateExerciseList();
  });
}

```

Figure 6-38: Code segment of delete exercise question via Axios API with DELETE method

```

function addExercise(Request $request){
  return Exercise::create($request->all());
}

function editExercise(Request $request, $exerciseID){
  $exercise = Exercise::findOrFail($exerciseID);
  $exercise->update($request->all());
  return $exercise;
}

function deleteExercise($exerciseID){
  $exercise = Exercise::findOrFail($exerciseID);
  $exercise->delete();
  return 204;
}

```

Figure 6-39: Code segment of “addExercise” method, “editExercise” method and “deleteExercise” method in “AdminController” class controller

CHAPTER 7

SYSTEM TESTING

The purpose of this chapter is to provide explanation of the conducted system testing in this project. Section 7.1 is to give briefing on the types of conducted testing. Section 7.2 is to describe the conducted unit testing. Section 7.3 is to illustrate the conducted integration testing. Section 7.4 is to describe the conducted usability testing and UAT testing.

7.1 Testing Types

Four types of testing were conducted after implementing the project, which are unit testing, integration testing, usability testing, and user acceptance testing (UAT). In unit testing, each unit of the system component has been tested to guarantee that every system will perform well. In integration testing, communication between the system and the database were tested to make sure the database data will be modified when the system sends a request. In usability testing and UAT testing, end-users were tested to use the system to ensure that the system is easy to use, and the whole system works well when the system was hosted in the web hosting service.

7.2 Unit Testing

In the unit testing, 13 unit test modules with a total of 95 unit test cases were conducted. 94 out of 95 test cases are passed during the unit testing. Table 7-1 shows the summary of the unit testing results. All unit test cases had been attached as Appendix F.

Table 7-1: Summarized unit testing results

Unit Test Module ID	Unit Test Module Name	Number of unit test cases in the module	Number of passed unit test cases in the module
UT-TC-001	Introduction	4	4
UT-TC-002	Login System	12	12
UT-TC-003	Student	8	8

	Onboarding		
UT-TC-004	Reset Password	7	7
UT-TC-005	Lesson	5	5
UT-TC-006	Exercise	6	6
UT-TC-007	Graded Quizzes	11	11
UT-TC-008	Online Code Editor	4	4
UT-TC-009	Student Profile	9	9
UT-TC-010	Online Chat Box	6	5
UT-TC-011	Register Student	5	5
UT-TC-012	Student Profile	5	5
UT-TC-013	Modifying Course Content	13	13
Total		95	94

The failed unit test case is the 72th unit test case from the online chat box test module, which is testing whether the system will automatically delete chat data that has existed more than 3 days. However, during the testing, this function can only be triggered by using the command line interface.

7.3 Integration Testing

In the integration testing, 6 integration test modules with a total of 19 integration test cases were executed. Axios API with POST method, PUT method and DELETE method between the system with the database are tested. All test cases are passed during the integration testing. Table 7-2 shows the summary of the integration test results. All integration test cases had been attached as in Appendix G.

Table 7-2: Summarized integration test results

Integration Test Module ID	Integration Test Module Name	Number of integration test cases in the module	Number of passed integration test cases in the module
IT-TC-001	Password Reset	2	2

	and Onboarding System		
IT-TC-002	Graded Quizzes	1	1
IT-TC-003	Chat Box	2	2
IT-TC-004	Register Student	1	1
IT-TC-005	Delete Student	1	1
IT-TC-006	Edit Course Content	12	12
Total		19	19

7.4 Usability Testing and UAT Testing

The system had been uploaded into the web hosting server after conducting unit testing and integration testing in the local server. 13 testers which consists of 12 students and 1 lecturer from UECS3263 iOS Application Development were recruited to perform usability testing and UAT testing at the same time. During the testing, one-to-one meetings had been conducted in Microsoft Teams to provide briefings, test documents, accounts and a website link to the testers. Each tester followed the test scenarios and expected results stated in each test case, and filled in the status as pass or fail after executing each test case. Once the tester completed the testing, the tester submitted the completed test cases to the meeting chat, and a Google Form link had been given to fill in the User Satisfaction Survey.

7.4.1 UAT Testing Result

12 students had conducted 21 student-related test cases and 1 lecturer had conducted 36 administrator-related test cases. For student-related testing, 11 students passed all test cases but 1 student failed 2 out of 21 test cases. For administrator-related testing, all test cases are passed. Table 7-3 and Table 7-4 shows the summarized test results conducted by students and administrator respectively. UAT test results conducted by students had been attached as in Appendix H, whereas test results conducted by the administrator had been attached as in Appendix I.

Table 7-3: Summarized student UAT test results

Test Case ID	Test Case Type	Number of	Number of passed
--------------	----------------	-----------	------------------

		executions	test cases performed
TC-LI-001	Login	12	12
TC-RP-001	Reset Password	12	12
TC-LI-002	Login	12	12
TC-OB-001	Onboarding	12	12
TC-OB-002	Onboarding	12	12
TC-TL-001	Learn Lesson	12	12
TC-TL-002	Learn Lesson	12	12
TC-CP-001	Code Playground	12	11
TC-CP-002	Code Playground	12	11
TC-EE-001	Exercise	12	12
TC-EE-002	Exercise	12	12
TC-EE-003	Exercise	12	12
TC-EE-004	Exercise	12	12
TC-GQ-001	Graded Quizzes	12	12
TC-GQ-002	Graded Quizzes	12	12
TC-GQ-003	Graded Quizzes	12	12
TC-GQ-004	Graded Quizzes	12	12
TC-PF-001	Student Profile	12	12
TC-CB-001	Chat Box	12	12
TC-CB-002	Chat Box	12	12
TC-CB-003	Chat Box	12	12
TC-LO-001	Logout	12	12

Table 7-4: Summarized administrator UAT test results

Test Case ID	Test Case Type	Number of executions	Number of passed test cases performed
TC-LIA-001	Login	1	1
TC-MTA-001	Add Topic	1	1
TC-MTA-002	Edit Topic Name	1	1

TC-MTA-003	Delete Topic	1	1
TC-TLA-001	Modify Lesson	1	1
TC-TLA-002	View Lesson	1	1
TC-TLA-003	Add Lesson Section	1	1
TC-TLA-004	Edit Lesson Section	1	1
TC-TLA-005	Delete Lesson Section	1	1
TC-CPA-001	Code Playground	1	1
TC-EEA-001	Exercises	1	1
TC-EEA-002	Exercises	1	1
TC-EEA-003	Exercises	1	1
TC-EEA-004	Exercises	1	1
TC-EEA-005	Add Exercise Question	1	1
TC-EEA-006	Edit Exercise Question	1	1
TC-EEA-007	Delete Exercise Question	1	1
TC-GQA-001	Graded Quizzes	1	1
TC-GQA-002	Graded Quizzes	1	1
TC-GQA-005	Add Quiz Question	1	1
TC-GQA-006	Edit Quiz Question	1	1
TC-GQA-007	Delete Quiz Question	1	1
TC-RSA-001	Register Student	1	1
TC-RSA-002	Register Student	1	1
TC-RSA-003	Register Student	1	1
TC-RSA-004	Register Student	1	1
TC-PFA-001	Admin Profile	1	1
TC-PFA-002	Student List	1	1

TC-PFA-003	Search Student	1	1
TC-PFA-004	Delete Student	1	1
TC-PFA-005	Search Student	1	1
TC-CBA-001	Chat Box	1	1
TC-CBA-002	Chat Box	1	1
TC-CBA-003	Chat Box	1	1
TC-CBA-004	Chat Box	1	1
TC-LOA-001	Logout	1	1

The failed test cases that a student tester performed were code editor testing. When the tester opened code editor link from topic section or code playground page, the code editor is unable to load the codes, and an error “`{{‘run’|i18n}}` (Ctrl-Enter)” is shown. It has been confirmed that this issue may occur when the tester is using an incompatible browser, instead of the web hosting service or the web application itself.

7.4.2 Usability Testing Result

All questions in the User Satisfaction Survey were adopted from System Usability Scale (SUS), Brooke, J. (1986). There are two sections in this survey which are Section A and Section B. Section A consists of 10 rating questions, from the lowest rating (Strongly Disagree, 1 mark) to the high rating (Strongly Agree, 5 marks). The odd questions are positive questions whereas the even questions are negative questions, whereas Section B consists of 4 non-rating questions for respondents to write short answers. Table 7-5 shows the template of the User Satisfaction Survey. Each tester’s response had been attached as in Appendix J.

Table 7-5: User Satisfaction Survey template

No	Question	Strongly Disagree (1)	2	3	4	Strongly Agree (5)
1	I think that I would like to use this website/system for learning Swift Programming					

	Language and related matters.					
2	I found the website/system unnecessarily complex.					
3	I thought the website/system was easy to use.					
4	I think that I would need the support of a technical person to be able to use this website/system.					
5	I found this website/system was easily moved through without a lot of backtracking or data re-entry.					
6	I thought there was too much inconsistency in this website/system.					
7	I would imagine that most people would learn to use this website/system very quickly.					
8	I found the system/website very awkward to use.					
9	I felt very confident using the system/website.					
10	I needed to learn a lot of things before I could get going with this website/system.					

What did you like best about the site?
What did you like least about the site?
If you were to describe this site to a colleague in a sentence or two, what would you

say?
Do you have any other final comments or questions?

Table 7-6: SUS result

Question	Total score obtained from 13 testers	Maximum score can be obtained from 13 testers	Average score
1	45	52	3.46
2	39	52	3.00
3	46	52	3.54
4	43	52	3.31
5	44	52	3.38
6	43	52	3.31
7	46	52	3.54
8	45	52	3.46
9	43	52	3.31
10	45	52	3.46
Total Score	439		33.77
SUS Score			84.43

According to Brooke, J. (1986, cited in Smyk, 2020), the SUS score from each respondent can be calculated by adopting the formula below:

1. Sum up all the odd question scores and minus 1.
2. 5 minus the total score obtained from even questions.
3. Add both calculated scores together and multiply by 2.5, the range of the SUS score should be between 0 to 100.

Table 7-7: Graded and Rating of SUS score (Usabilitest, n.d.)

SUS Score	Grade	Rating
80.4 to 100	A	Excellent

69 to 80.3	B	Good
68	C	Average
51 to 67	D	Poor
0 to 51	F	Awful

Based on the calculated SUS score from Table 7-6 and SUS score grade as shown as Table 7-7, the obtained SUS score from respondents is 84.43 with Grade A and Excellent Rating. This shows that the implemented system is user-friendly, consistent, and easy to use.

CHAPTER 8

CONCLUSION

The purpose of this chapter is to draw an ending for this project. Section 8.1 is to conclude performed works in this project. Section 8.2 is to list out limitations occurred in the project. Section 8.3 is to list out recommendations to overcome the listed limitations in the future.

8.1 Conclusion

All objectives stated in Chapter 1 Section 1.3 were successfully achieved in this project, which include:

1. To analyze the existing online programming platform with the perspective of strengths and weaknesses.
2. To research the characteristics of Swift Programming Language and compare it to its predecessor, Objective-C programming language.
3. To design and implement the web application by adopting evolutionary prototyping methodology.
4. To test and evaluate the web application functionalities by using unit testing, integration testing, usability testing and user acceptance testing.

For the first achieved objective, 5 existing online programming platform were chosen to compare the features, pros and cons. For the second achieved objective, research papers and literature reviews were conducted to find out the Swift Programming Language and Objective-C programming language. For the third achieved objective, requirements are defined, UML diagrams and user interface are designed properly, and all stated requirements are implemented into the system. For the fourth achieved objectives, All test cases are passed in the integration testing and almost all test cases are passed in the unit testing and user acceptance testing.

It's confirmed that the implemented Swift Programming Language E-Learning Platform for iOS Application Development is the best platform for students who are taking UECS3263 iOS Application Development, as this platform can be

used with a browser and no Mac or iPad is required to learn Swift Programming Language. Besides, the implemented system had become available to all iOS Application Development students until the end of the May 2021 trimester.

8.2 Limitations

Some limitations were faced during the implementation, testing and deployment of the system for this project, which include:

1. Unable to implement mail notification of temporary password and six digit code verification for students, as the adopted web hosting service prohibits the use of mail function in subdomain websites.
2. A non-functional requirement has to be changed due to the lack of crontab in the server, therefore deletion of chat with more than 3 days can be only done manually.
3. Unable to upload profile image function, as file type input is unable to send to controller via Axios API.
4. Unable to generate PDF file for student's performance.
5. Unable to make the chat data reload instantly.

8.3 Recommendations

Some recommendations and enhancement from Section 8.2 were listed for future work:

1. Find a mail service provider that is fully compatible with the chosen web hosting service provider, without purchasing any domain from the web hosting service.
2. Add cron jobs into the server to perform the chat deletion automatically.
3. Encrypt the image file into Base64 data type before sending it to the controller via API.
4. Design a PDF file template to store data into it.
5. Find a free web hosting service with a high-performance server that supports InnoDB engine, and a huge number of concurrent connections for data reload every second.

6. Collaborate with data scientists to perform data predictions and machine learning by using quiz time taken and number of steps going the question backward and forward.
7. Improve the user interface to fit with mobile web app layout.

REFERENCES

Ali, W., 2020. *Online and remote learning in higher education institutes: a necessity in light of COVID-19 pandemic*. *Higher Education Studies*, 10(3), pp.16 – 25 [online]. Available at <<https://www.researchgate.net/publication/341460604>> [Accessed: 7th March 2021]

Altexsoft, 2018. *Swift vs Objective-C: out with the old, in with the new* [online]. Available at <<https://www.altexsoft.com/blog/engineering/swift-vs-objective-c-out-with-the-old-in-with-the-new/>> [Accessed: 2nd March 2021]

Apple, n.d., *Xcode – Support – Apple Developer*. [online] Available at: <<https://developer.apple.com/support/xcode/>> [Accessed: 10th February 2021]

Apple, n.d., *Swift*. [online] Available at: <<https://developer.apple.com/swift>> [Accessed: 2nd March 2021]

Apple, n.d., *Swift Playgrounds*. [online] Available at: <<https://www.apple.com/swift/playgrounds/>> [Accessed: 10th February 2021]

Anwer, F., Aftab, S., Shah, S. S. M., Waheed, U., 2017. *Comparative analysis of two popular agile process models: extreme programming and scrum*. *International Journal of Computer Science and Telecommunications*, 8(2), pp.1 – 7 [online]. Available at: <<https://www.researchgate.net/publication/316845761>> [Accessed: 27th February 2021]

Chauhan, D. B., Rana, A., Sharma, N. K., 2017. *Impact of development methodology on cost & risk for development projects*. *2017 6th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, pp.267 – 272 [online]. Available at: <<https://ieeexplore.ieee.org/document/8342436>> [Accessed: 27th February 2021]

Chernova E.A., Nazarov A.D., 2020. *Digital competencies of the future programmer: Swift*. *2nd International Scientific and Practical Conference on Digital Economy (ISCDE 2020)*, pp 559 – 562 [online]. Available at: <<https://www.atlantispress.com/proceedings/iscde-20/125947820>> [Accessed: 2nd March 2021]

Curry, D., 2021. *App revenue data (2020)*. [online] Available at: <<https://www.businessofapps.com/data/app-revenues/>> [Accessed: 10th February 2021]

Dennis, A., Wixom, B. H., Tegarden, D., 2015. *System analysis & design: An object-oriented approach with UML. 5th edition*. John Wiley and Sons [online]. Available at: <<http://search.ebscohost.com.libezp2.utar.edu.my/>> [Accessed: 27th February 2021]

Differ, n.d., *Android vs iOS: difference and comparison*. [online] Available at: <https://www.differ.com/difference/Android_vs_iOS> [Accessed: 10th February 2021]

Fojtik, R., 2020. *Swift a new programming language for development and education*. *Digital Science 2019*, pp.284 – 295 [online]. Available at: <https://www.researchgate.net/publication/338081271_Swift_a_New_Programming_Language_for_Development_and_Education> [Accessed: 2nd March 2021]

García, C. G., Espada, J. P., G-Bustelo, B. C. P., Lovelle, J. M. C., 2015. *Swift vs. Objective-C: a new programming language*. *International Journal of Artificial Intelligence and Interactive Multimedia*, 3(3), pp.74 – 81. Available at <<https://documat.unirioja.es/servlet/articulo?codigo=5574309>> [Accessed: 2nd March 2021]

Gandraß, N., Hinrichs, T., Schmolitzky, A., 2020. *Towards an online programming platform complementing software engineering education*. *SEUH 2020*, pp.27 – 35 [online]. Available at <<https://www.semanticscholar.org/paper/Towards-an-Online-Programming-Platform-Software-Gandra%C3%9F-Hinrichs/9c0bdfaf3cdc5c45b888999eb21ddf1239e62dce>> [Accessed: 6th March 2021]

Hubbarrt, M., 2017. *Beginning iOS 11 Programming with Swift* [electronic print]. Available at <<https://mhreviews.wordpress.com/2017/11/21/beginning-ios-11-programming-with-swift/>> [Accessed: 2nd March 2021]

Jayasinghe, P., 2020. *Throwaway prototyping vs evolutionary prototyping*. *Medium* [online]. Available at: <<https://medium.com/@pavithrajayasinghe9529/throwaway-prototyping-vs-evolutionary-prototyping-8302be3baf33>> [Accessed: 27th February 2021]

JetBrains, n.d., *Swift & Objective-C 2019 – The state of developer ecosystem in 2019 infographic*. [online] Available at: <<https://www.jetbrains.com/lp/devecosystem-2019/swift-objc/>> [Accessed: 10th February 2021]

JetBrains, n.d., *Swift & Objective-C – The state of developer ecosystem in 2020 infographic*. [online] Available at: <<https://www.jetbrains.com/lp/devecosystem-2020/swift-objc/>> [Accessed: 10th February 2021]

StarCounter, n.d., *Desktop operating system market share worldwide* [online]. Available at: <<https://gs.statcounter.com/os-market-share/desktop/worldwide>> [Accessed: 10th February 2021]

Karczewski, D., 2020. *Swift vs Objective-C: which should you pick for your next iOS mobile app?* *Ideamotive* [online]. Available at <<https://www.ideamotive.co/blog/swift-vs-objective-c-which-should-you-pick-for-your-next-ios-mobile-app>> [Accessed: 2nd March 2021]

Khalid, H., 2018. *Difference between evolutionary prototyping and throw-away prototyping*. *Prototype Info* [online]. Available at: <<https://prototypeinfo.com/evolutionary-prototyping-and-throw-away-prototyping/>> [Accessed: 27th February 2021]

Kurcwald, K., 2019. *App prototyping – why it is essential and how to do it?* *Monterail* [online]. Available at: <<https://www.monterail.com/blog/prototyping-in-software-development#:~:text=A%20prototype%20is%20an%20essential,on%20how%20to%20improve%20it.>> [Accessed: 27th February 2021]

Nguyen, Q. L. H. T. T., Nguyen, P. T., Huynh, V. D. B., 2019. *Roles of e-learning in higher education*. *Journal of Critical Reviews*, 6(4), pp.7 – 13 [online]. Available at: <<http://www.jcreview.com/fulltext/197-1576580263.pdf?1577175597>> [Accessed: 5th March 2021]

Nortvig, A., Petersen, A. K., Balle, S. H., 2018. *A literature review of the factors influencing e-learning and blended in relation to learning outcome, student satisfaction and engagement*. *Electronic Journal of e-Learning*, 16(1), pp.46 – 55 [online]. Available at <<https://eric.ed.gov/?id=EJ1175336>> [Accessed: 6th March 2021]

Radha, R., Mahalakshmi, K., Kumar, V. S., Saravanakumar, AR., 2020. *E-learning during lockdown of covid-19 pandemic: a global perspective*. *International Journal of Control and Automation*, 13(4), pp.1088 – 1099 [online]. Available at: <<https://www.researchgate.net/publication/342378341>> [Accessed: 6th March 2021]

Sarangam, A., 2020. *What Is Client Server Architecture? An Overview* [online]. Available at: <<https://www.jigsawacademy.com/blogs/cyber-security/what-is-client-server-architecture/>> [Accessed: 14th July 2021]

Sharma, L., 2016. *Waterfall model*. *ToolSQA* [online]. Available at <<https://www.toolsqa.com/software-testing/waterfall-model/>> [Accessed: 27th February 2021]

Singh, B., Kaur, R., 2017. *Raising performance of iPhone using Swift language over other programming languages*. *International Journal of Advance Research, Ideas and Innovations in Technology*, 3(6), pp.991 – 994 [online]. Available at <https://www.academia.edu/download/55483864/Raising_Performance_of_iPhone_using_Swift_Language_over_Other_Programming_Languages.pdf> [Accessed: 2nd March 2021]

Smyk, A., 2020. *The System Usability Scale & How It's Used in UX*. *Adobe Xd Ideas* [online]. Available at <<https://xd.adobe.com/ideas/process/user-testing/sus-system-usability-scale-ux/>> [Accessed: 27th July 2021]

Svirca, Z., 2020. *Everything you need to know about MVC architecture. Towards Data Science*. [online] Available at: <<https://towardsdatascience.com/everything-you-need-to-know-about-mvc-architecture-3c827930b4c1>> [Accessed: 17th February 2021]

Collegenote, n.d., *Software Engineering 2071* [online]. Available at <<https://collegenote.pythonanywhere.com/pastpapers/4451/question/#gsc.tab=0>> [Accessed: 10th February 2021]

Swift, n.d., *About Swift* [online]. Available at <<https://swift.org/about/>> [Accessed: 2nd March 2021]

Swift, n.d., *Document revision history* [online]. Available at: <<https://docs.swift.org/swift-book/RevisionHistory/RevisionHistory.html>> [Accessed: 17th February 2021]

Usabilitest, n.d., *System Usability Scale (SUS) Plus* [online]. Available at : <<https://www.usabilitest.com/system-usability-scale>> [Accessed: 27th July 2021]

Website.com, n.d., *What is Web Hosting?* [online]. Available at <<https://www.website.com/beginnerguides/webhosting/6/1/what-is-web-hosting?.ws&source=SC>> [Accessed: 13th July 2021]

Zhang, P., Song, Y., Kang, B., Chen, W., 2018. *Online programming platform based on crowdsourcing. The 13th International Conference on Computer Science & Education (ICCSE 2018)*, pp. 302-307 [online]. Available at <<https://ieeexplore.ieee.org/document/8468735>> [Accessed: 17th February 2021]

APPENDICES

APPENDIX A: Work Breakdown Structure

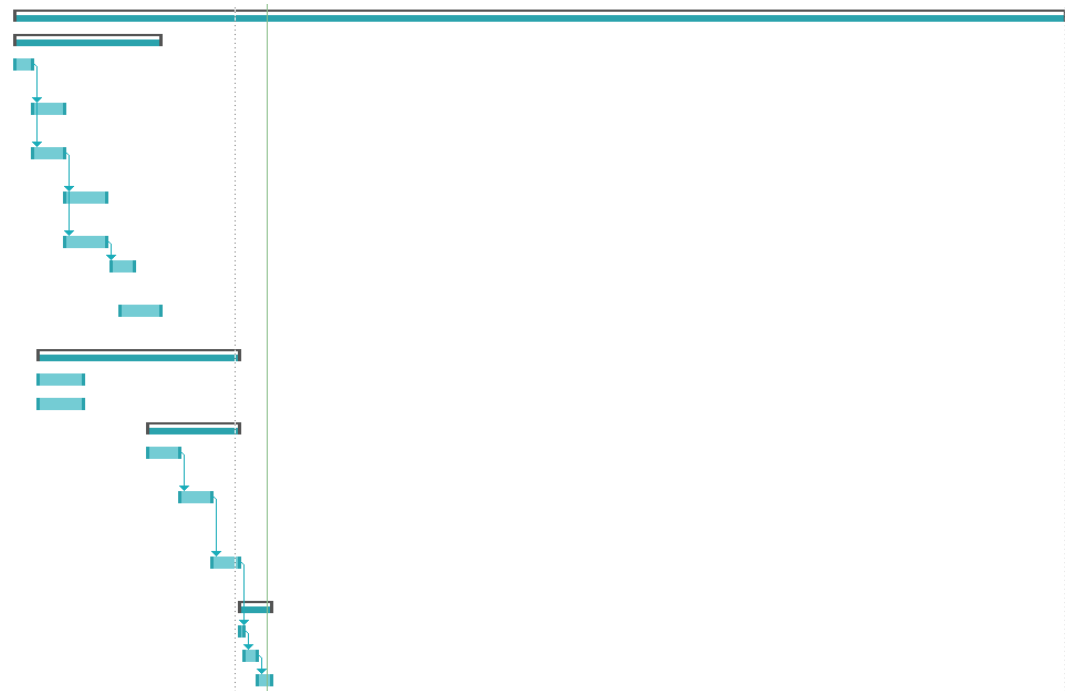
Task
1.0 Project Initiation
1.1 Define project background
1.2 define problem statement
1.3 Define project objective
1.4 Define project solution
1.5 Define project approach
1.6 Define project scope
1.7 Prepare preliminary report
2.0 Literature Review
2.1 Lookup relevant literatures
2.2 Prepare literature matrix
2.3 Prepare Chapter 2 writing
2.3.1 Review on Swift programming language
2.3.2 Review on software development methodologies
2.3.3 Review on e-learning and online programming platform
3.0 Methodology
3.1 Define project plan
3.2 Determine final software development methodology
3.3 Determine adopted developing tools
4.0 Project Specification
4.1 Define requirements specification
4.1.1 Define functional requirements
4.1.2 Define non-functional requirements
4.2 Design use case diagram
4.3 Define use case description
5.0 Prototype Design and Final Implementation
5.1 First iteration

5.1.1 Develop initial system prototype
5.1.1.1 Develop code editor and topic content
5.1.1.2 Develop quizzes and exercises
5.1.2 Design basic user interface
5.1.3 UAT testing by users
5.1.4 Collect feedback from users
5.1.5 Improve prototype
5.2 Second iteration
5.2.1 Design UML Diagrams
5.2.1.1 Design Class Diagram
5.2.1.2 Design Sequence Diagram
5.2.1.3 Design Activity Diagram
5.2.1.4 Design ERD Diagram
5.2.2 Design second system prototype
5.2.2.1 Develop chat box system
5.2.2.2 Develop login function
5.2.2.3 Develop whole user interface
5.2.3 UAT testing by users
5.2.4 Collect Feedback from users
5.2.5 Improve prototype
5.3 Final iteration
5.3.1 Develop final system prototype
5.3.1.1 Develop administrative system
5.3.1.2 Integrate all prototype into final system
6.0 Testing
6.1 Unit Testing
6.2 Integration Testing
6.3 UAT and Usability Testing for the final system
6.3.1 Collect feedback from the final system
6.4 Improve the final system
7.0 Project Closure
7.1 Deploy the final system

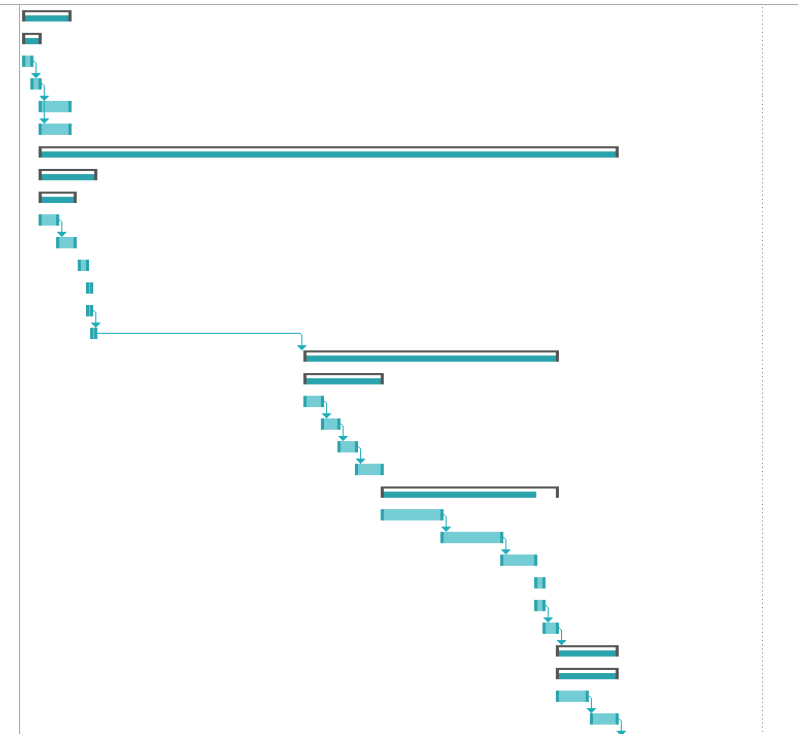
7.2 Finalize the final report and presentation slides

APPENDIX B: Project Gantt Chart

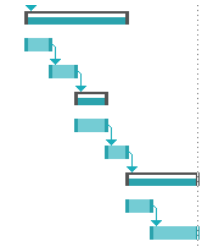
Web-based Swift online programming platform	165 days?	Mon 18/1/21	Fri 3/9/21
1.0 Project Initiation	24 days?	Mon 18/1/21	Thu 18/2/21
Define project background	4 days	Mon 18/1/21	Thu 21/1/21
Define problem statement	5 days	Fri 22/1/21	Thu 28/1/21
Define project objective	5 days	Fri 22/1/21	Thu 28/1/21
Define project solution	7 days	Fri 29/1/21	Sat 6/2/21
Define project approach	7 days	Fri 29/1/21	Sat 6/2/21
Define project scope	5 days	Mon 8/2/21	Fri 12/2/21
Prepare preliminary report	7 days	Wed 10/2/21	Thu 18/2/21
2.0 Literature Review	32 days	Sat 23/1/21	Sun 7/3/21
Lookup relevant literatures	7 days	Sat 23/1/21	Mon 1/2/21
Prepare literature matrix	7 days	Sat 23/1/21	Mon 1/2/21
Prepare Chapter 2 writing	15 days	Tue 16/2/21	Sun 7/3/21
Review on Swift programming language	5 days	Tue 16/2/21	Mon 22/2/21
Review on software development methodologies	5 days	Tue 23/2/21	Mon 1/3/21
Review on e-learning and online programming platform	5 days	Tue 2/3/21	Sun 7/3/21
3.0 Methodology	6 days	Mon 8/3/21	Sun 14/3/21
Define project plan	1 day	Mon 8/3/21	Mon 8/3/21
Determine final software development methodology	3 days	Tue 9/3/21	Thu 11/3/21
Determine adopted developing tools	2 days	Fri 12/3/21	Sun 14/3/21



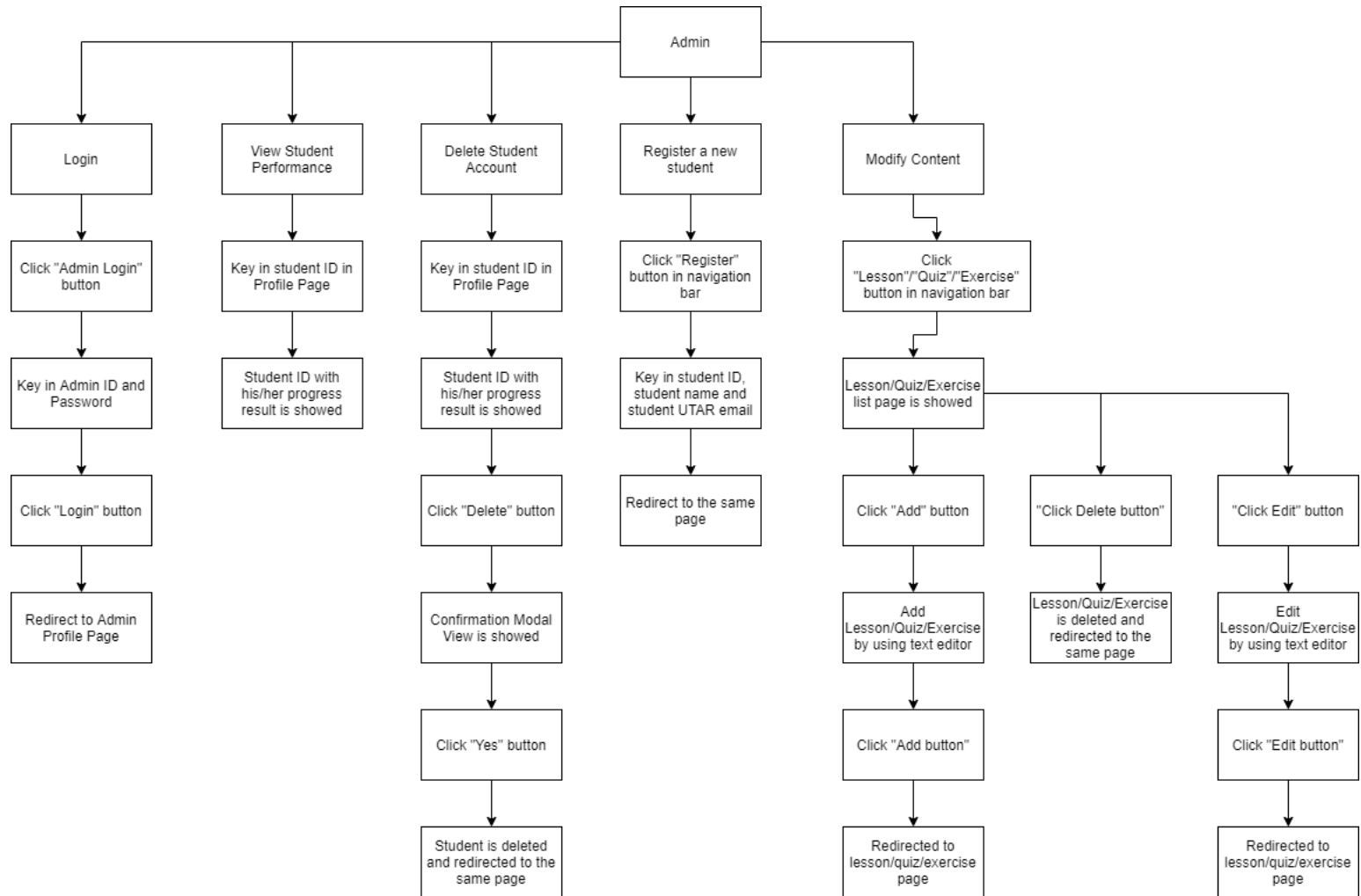
4.0 Project Specification	9 days	Mon 15/3/21	Thu 25/3/21
Define requirements specification	4 days	Mon 15/3/21	Thu 18/3/21
Define functional requirements	2 days	Mon 15/3/21	Tue 16/3/21
Define non-functional requirements	2 days	Wed 17/3/21	Thu 18/3/21
Design use case diagram	5 days	Fri 19/3/21	Thu 25/3/21
Define use case description	5 days	Fri 19/3/21	Thu 25/3/21
4.0 Prototype Design and Final Implementation	97 days?	Fri 19/3/21	Sat 31/7/21
First Iteration	9 days	Fri 19/3/21	Wed 31/3/21
Develop initial system prototype	6 days	Fri 19/3/21	Fri 26/3/21
Develop code editor and topic content	2 days	Fri 19/3/21	Mon 22/3/21
Develop quizzes and exercises	4 days	Tue 23/3/21	Fri 26/3/21
Design basic user interface	2 days	Sun 28/3/21	Mon 29/3/21
UAT testing by users	1 day	Tue 30/3/21	Tue 30/3/21
Collect Feedback from users	1 day	Tue 30/3/21	Tue 30/3/21
Improve prototype	1 day	Wed 31/3/21	Wed 31/3/21
Second Iteration	43 days	Thu 20/5/21	Sat 17/7/21
Design UML Diagrams	13 days	Thu 20/5/21	Sun 6/6/21
Design Class Diagram	3 days	Thu 20/5/21	Sun 23/5/21
Design Sequence Diagram	4 days	Mon 24/5/21	Thu 27/5/21
Design Activity Diagram	2 days	Fri 28/5/21	Mon 31/5/21
Design ERD Diagram	5 days	Tue 1/6/21	Sun 6/6/21
Design second system prototype	31 days	Mon 7/6/21	Sat 17/7/21
Develop chatbox system	11 days	Mon 7/6/21	Sun 20/6/21
Develop login function	11 days	Mon 21/6/21	Sun 4/7/21
Develop whole user interface	6 days	Mon 5/7/21	Mon 12/7/21
UAT Testing by users	2 days	Tue 13/7/21	Wed 14/7/21
Collect feedback from users	2 days	Tue 13/7/21	Wed 14/7/21
Improve prototype	3 days	Thu 15/7/21	Sat 17/7/21
Final iteration	12 days?	Sun 18/7/21	Sat 31/7/21
Develop final system prototype	12 days?	Sun 18/7/21	Sat 31/7/21
Develop administrative system	7 days	Sun 18/7/21	Sat 24/7/21
Integrate all prototype into final system	6 days	Mon 26/7/21	Sat 31/7/21

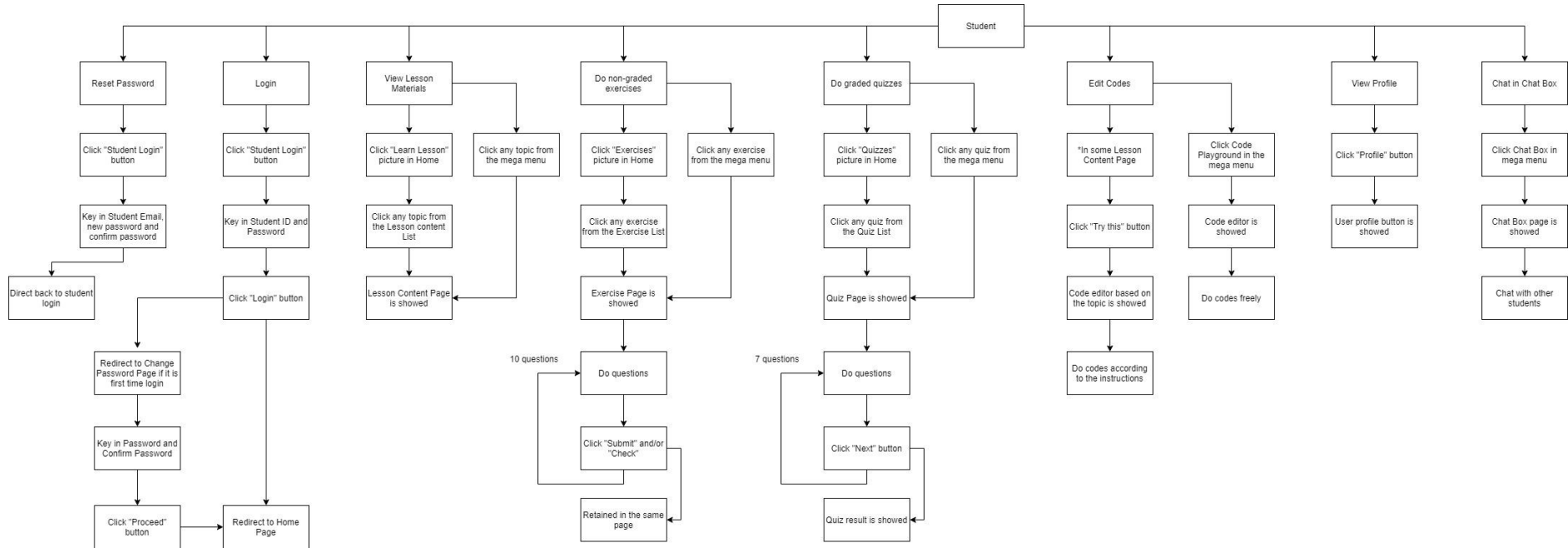


4 6.0 Testing	16 days?	Sun 1/8/21	Fri 20/8/21	51		
Unit Testing	5 days	Sun 1/8/21	Thu 5/8/21			
Integration Testing	3 days	Fri 6/8/21	Tue 10/8/21	53		
4 UAT and Usability Testing for the final system	4 days?	Wed 11/8/21	Mon 16/8/21	54		
Collect feedback from the final system	4 days	Wed 11/8/21	Mon 16/8/21			
Improve the final system	4 days	Tue 17/8/21	Fri 20/8/21	56		
4 7.0 Project closure	11 days	Sat 21/8/21	Fri 3/9/21	57		
Deploy the final system	4 days	Sat 21/8/21	Wed 25/8/21			
Finalize the final report and presentation slides	7 days	Thu 26/8/21	Fri 3/9/21	59		

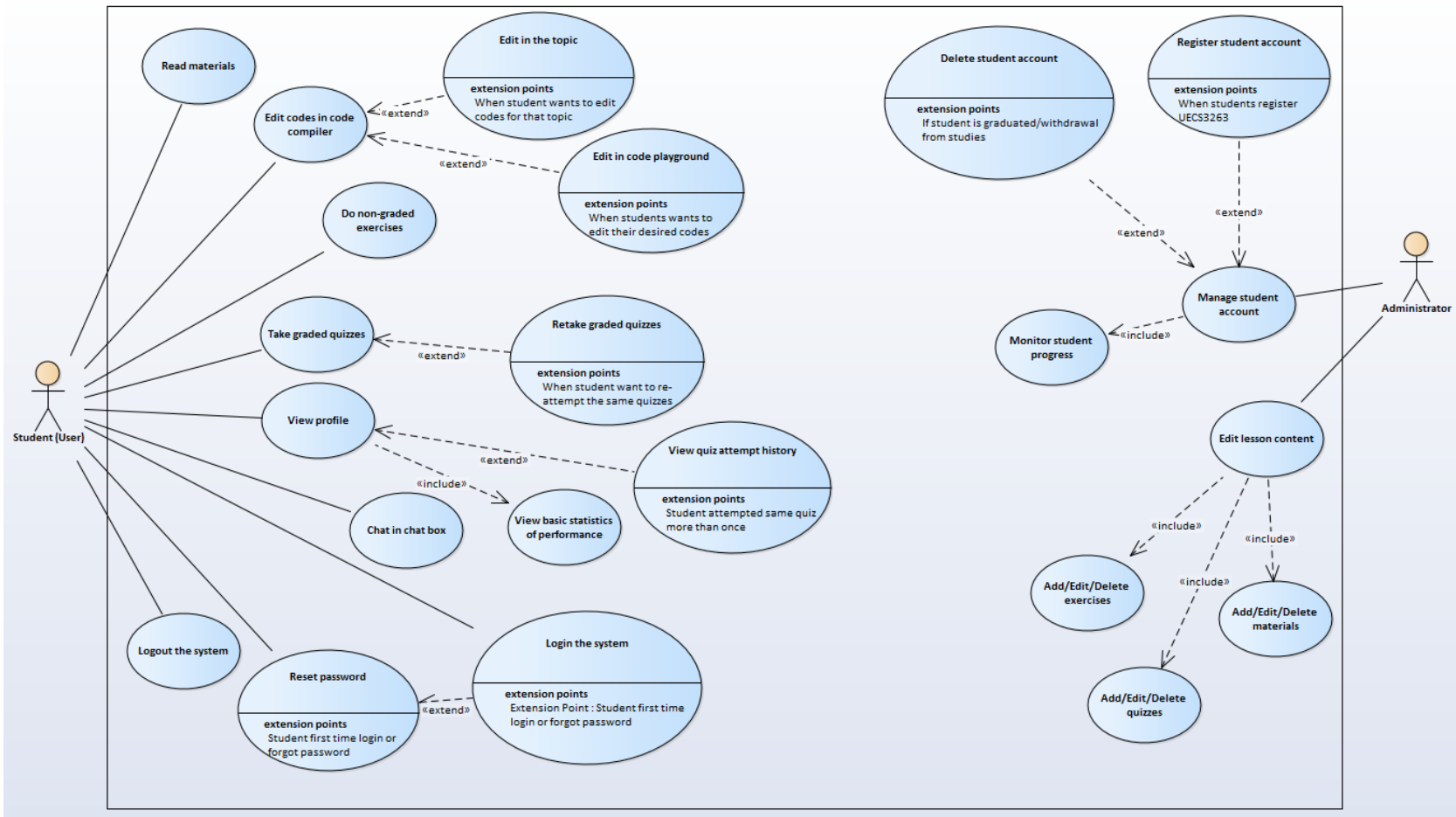


APPENDIX C: User Interface Flow Chart

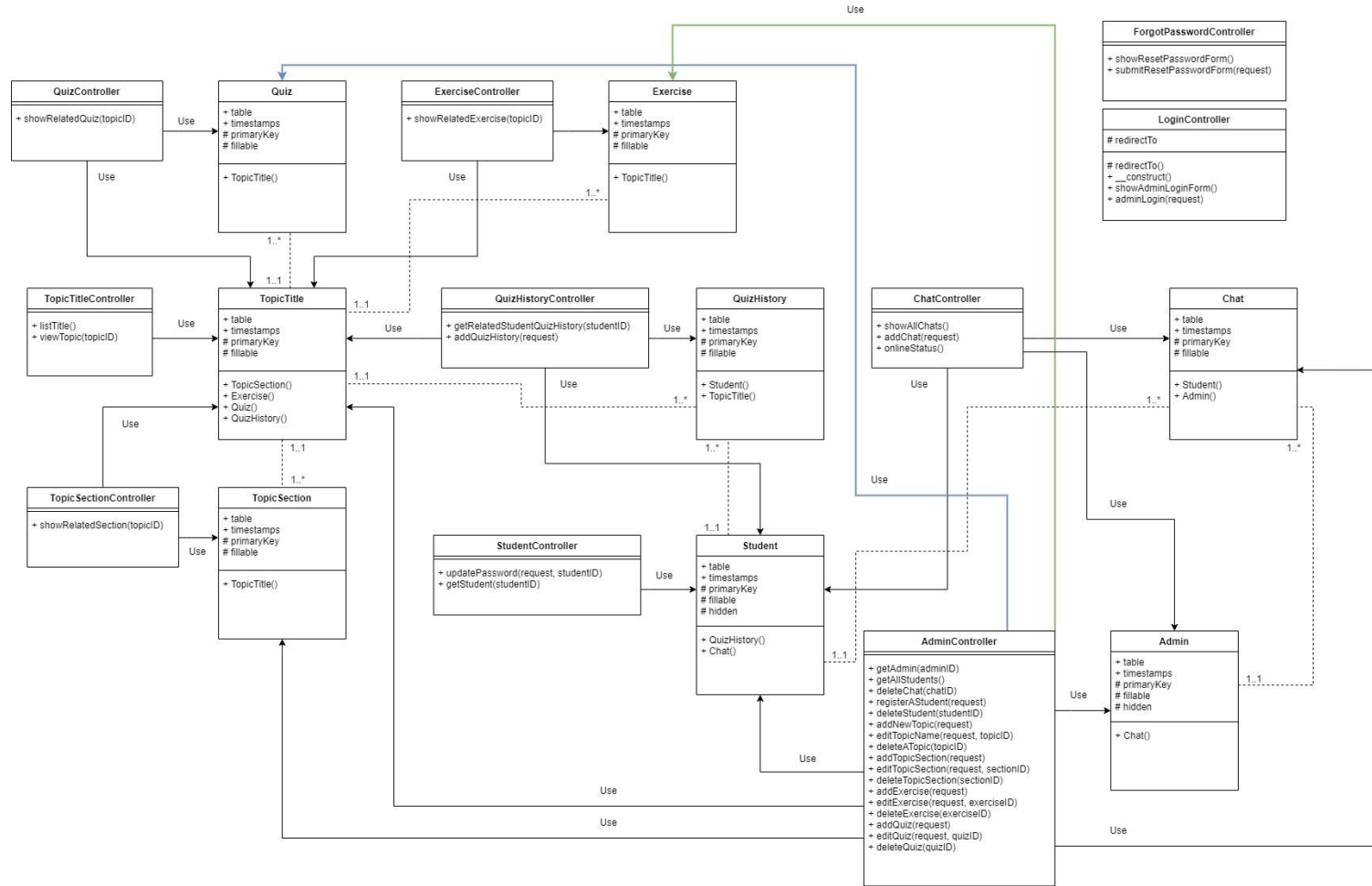




APPENDIX D: Use Case Diagram



APPENDIX E: Designed Class Diagram



APPENDIX F: Unit Test Cases

Module: Introduction		Test Module ID: UT-TC-001				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the introduction of the website is shown.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
1	Load welcome screen	1. Go to 127.0.0.1:8000	Command “php artisan serve” is executed, and Google Chrome browser is used.	N/A	Welcome screen is shown	Pass
2	Load student login screen	1. Click “Student Login” button	N/A	N/A	Login screen for student is shown	Pass
3	Load admin login screen	1. Click “Admin Login” button	N/A	N/A	Login screen for admin is shown	Pass
4	Load about screen	1. Click “About” button	N/A	N/A	About screen for admin is shown	Pass

Module: Login System		Test Module ID: UT-TC-002				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the login system for both student and admin are worked properly.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
5	Student login into the system for the first time	1. Key in Student ID and password 2. Click "Login" button	The registered student account should be a new user account.	Student ID and temporary password	Student will login into the system and redirect to the onboard screen.	Pass
6	Student login into the system	1. Key in Student ID and password 2. Click "Login" button	The registered student account should be a current account.	Student ID and self-reset password	Student will login into the system and redirect to the student home screen.	Pass
7	Student try to access	1. type "http://127.0.0.1:8000/login" in the address bar	The student must login into the system first	N/A	Student is redirected to the home page	Pass

	login page after login					
8	Student try to access admin-related page after login	1. type "http://127.0.0.1:8000/admin/home" in the address bar	The student must login into the system first	N/A	Student is redirected to the home page	Pass
9	Student logout from the system	1. Click "Logout" button on the navigation bar	N/A	N/A	Student will logout from the system and redirect to the welcome home screen.	Pass
10	Student login into the system with invalid credentials	1. Key in Student ID and invalid password 2. Click "Login button"	N/A	Student ID with invalid password	An error message is shown.	Pass

11	User try to access pages that requires login	1. type “http://127.0.0.1:8000/home” in the address bar	The user should not login into the system	N/A	User redirected to the student login page.	Pass
12	User try to access admin-related page before login	1. type “http://127.0.0.1:8000/admin/home” in the address bar	The user should not login into the system	N/A	User redirected to the admin login page.	Pass
13	Admin login into the system	1. Key in Admin ID and password 2. Click “Login” button	N/A	Admin ID with password	Admin will login into the system and redirect to the admin home screen.	Pass
14	Admin logout from the system	1. Click “Logout” button on the navigation bar	N/A	N/A	Admin will logout from the system and redirect to the welcome home screen.	

15	Admin try to access login page after login	1. type "http://127.0.0.1:8000/login" in the address bar	The admin must login into the system first	N/A	Student is redirected to the admin home page	Pass
16	Admin try to access student-related page after login	1. type "http://127.0.0.1:8000/home" in the address bar	The admin must login into the system first	N/A	Admin is redirected to the admin home page	Pass

Module: Student Onboarding		Test Module ID: UT-TC-003				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the onboarding system for the first time student is worked properly.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
17	Student reset password in the onboarding page	1. Key in password and confirm password.	The registered student account should be a new user account and login into the system.	New password	Student is redirected to the home screen.	Pass
18	Student reset password in the onboarding	1. Key in password and confirm password with 7 characters.	The registered student account should be a new user account and login into the system.	123 as password	An error message is shown.	Pass

	page with less than 8 characters						
19	Student reset password in the onboarding page with 8 numeric characters only	1. Key in password and confirm password with 8 numeric characters only.	The registered student account should be a new user account and login into the system.	11111111 password	as	An error message is shown.	Pass
20	Student reset password in the onboarding page with 8 letters only	1. Key in password and confirm password with 8 numeric characters letters only.	The registered student account should be a new user account and login into the system.	aaaaaaaa password	as	An error message is shown.	Pass

21	Student reset password in the onboarding page with different confirmation password	1. Key in password and confirm password that is not match with the password	The registered student account should be a new user account and login into the system.	12345abc as confirmation password. Password should not be 12345abc	An error message is shown.	Pass
22	Student leave empty for both password and confirm password	1. Leave blank for both password input and confirmation password input.	The registered student account should be a new user account and login into the system.	N/A	An error message is shown.	Pass
23	Student before onboard try to access	1. type “http://127.0.0.1:8000/home” in the address bar	The registered student account should be a new user account and login	N/A	403 error screen is shown.	Pass

	other web pages		into the system.			
24	Student after onboard try to access onboarding page	1. type "http://127.0.0.1:8000/onboard" in the address bar	The registered student account should be a current account and login into the system.	N/A	403 error screen is shown.	Pass

Module: Reset		Test Module ID: UT-TC-004				
Password						
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the reset password function is worked properly						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
25	Student reset password before login	1. Key in email, password and confirm password	N/A	Existing student email, new password and new confirmation password	Student is redirected to login screen with reset successful message	Pass
26	Block new student to reset password before onboarding	1. Key in email, password and confirm password	The registered student account should be a new user account.	New student email, new password and new confirmation password	Student is redirect back to the same page and block message is shown	Pass

27	Student key in email that is unregistered	1. Key in unregistered email, password and confirm password	N/A	Unregistered student email, new password and confirmation password	Student is redirect back to the same page and error message is shown	Pass
28	Student key in password that is different with confirmation password	1. Key in registered email, password and confirm password that is different with password	Password and confirm password should be different	Existing student email, new password, and different confirmation password	Student is redirect back to the same page and error message is shown	Pass
29	Student key in password with less than 8 characters	1. Key in password and confirm password with 7 characters.	N/A	123 as password	Student is redirect back to the same page and error message is shown	Pass
30	Student key in password with 8	1. Key in password and confirm password with 8 numeric characters only.	N/A	11111111 as password	Student is redirect back to the same page and error message is shown	Pass

	numeric characters only						
31	Student key in password with 8 letters only	1. Key in password and confirm password with 8 numeric characters letters only.	N/A	aaaaaaaa password	as	Student is redirect back to the same page and error message is shown	Pass

Module: Lesson		Test Module ID: UT-TC-005				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the lesson content is able to show students how to learn lessons.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
32	Student enter Topic List	1. On home page, click “Learn Lesson” button	N/A	N/A	Topic List with a list of topics are shown.	Pass
33	Student choose a topic	1. On topic list page, click “Topic 1” button	N/A	N/A	Topic 1 with section contents are shown.	Pass
34	Student click section anchor link	1. In the selected topic page, click any anchor link from the section list	N/A	N/A	Webpage will jump into the selected section.	Pass
35	Student click next button for next topic	1. In the selected topic page, click “Next button”	N/A	N/A	The system will jump to the next topic.	Pass
36	Student click	1. In the selected topic page,	N/A	N/A	The system will jump to	Pass

	back button for previous topic	click "Back button"			the previous topic.	
--	--------------------------------------	---------------------	--	--	---------------------	--

Module: Exercise		Test Module ID: UT-TC-006				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the exercise content is able to let students to practice their coding skills.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
37	Student enter Exercise List	1. On home page, click “Exercise” button	N/A	N/A	Exercise List with a list of exercises are shown	Pass
38	Student choose an exercise	1. On exercise list page, click “Exercise 1” button	N/A	N/A	Exercise 1 with related questions are shown	Pass
39	Student check the answer	1. On any question, click “Check Answer” button	N/A	N/A	Blank is disabled and show the correct answer. The submit button is hidden.	Pass
40	Student stop checking the answer	1. On the same question, click again the “Check Answer” button	N/A	N/A	Blank is enabled and the correct answer is wiped. The submit button is	Pass

					shown.	
41	Student submit the correct answer	1. On any question, key in the correct answer and click "Submit" button	N/A	Correct answer for the question	Blank is disabled. The submit button and check answer button are hidden. Correct message is shown.	Pass
42	Student submit the wrong answer	1. On any question, key in the wrong or blank answer and click "Submit" button	N/A	Wrong or blank answer for the question	Wrong answer message is shown.	Pass

Module: Graded Quizzes		Test Module ID: UT-TC-007				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the quiz content is able to show, and able to show results once the student has completed the quiz.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
43	Student enter Quiz List	1. On home page, click “Quizzes” button	N/A	N/A	Quiz List with a list of quizzes are shown	Pass
44	Student choose a quiz	1. On quiz list page, click “Quiz 1” button	N/A	N/A	Quiz 1 with start screen is shown	Pass
45	Student start a quiz	1. On the selected quiz page, click “Start”	N/A	N/A	Question 1 is shown	Pass
46	Student go to next question	1. On the question 1, click “Next” button	N/A	N/A	Question 1 is jump to Question 2	Pass
47	Student go to previous question	1. On the question 2, click “Back” button	N/A	N/A	Question 2 is jump back to Question 1	Pass

48	Student do not fully answer all questions	1. On any 5 questions, choose an answer. 2. On any remaining 5 questions, do not choose an answer 3. On the last question, click “Submit” button	N/A	N/A	An alert box will be shown.	Pass
49	Student leave empty for all questions.	1. Do not choose an answer for each question. 2. On the last question, click “Submit” button	N/A	N/A	An alert box will be shown.	Pass
50	Student submit all answered questions	1. Choose an answer for each question.	N/A	N/A	Result screen is shown.	Pass
51	Result percentage calculation	1. View number of correct answers over number of total questions.	N/A	N/A	Result percentage is calculated correctly.	Pass
52	Student is able to view	1. Answer all questions and clicked “Submit” button	N/A	N/A	Correct answers and wrong answers for the	Pass

	their answer.				quiz are shown.	
53	Student is able to retake the same quiz.	1. Click "Retry" button	N/A	N/A	The quiz screen will redirect back to the start quiz screen.	Pass

Module: Online code Editor		Test Module ID: UT-TC-008				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the online code editor is working well.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
54	Load online code editor in lesson section	<ol style="list-style-type: none"> 1. Click the “Learn Lesson” button in home. 2. Click the “Topic 1” button in the topic list. 3. Proceed to any section with an online code editor. 3. Click “Click to open the code editor for this section” button. 	N/A	N/A	Online code editor frame is loaded. Comments for each step are loaded.	Pass
55	Turn off online code editor in	<ol style="list-style-type: none"> 1. Click “Click to close the code editor for this section” button. 	The code editor frame should be opened before	N/A	Online code editor frame is off.	Pass

	lesson section					
56	Type codes and run codes in the online code editor.	1. Type any Swift codes in the online code editor. 2. Click “Run” button	N/A	N/A	Output is shown	Pass
57	Load online code editor in code playground page	1. Click “Code Playground” in Others > Code Playground in navigation bar	N/A	N/A	Online code editor frame is loaded.	Pass

Module: Student Profile		Test Module ID: UT-TC-009				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the student profile is functioning well.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
58	Student enter student profile	1. Click “Profile” on the navigation bar	N/A	N/A	Student file page is loaded. Student name, ID and email are displayed correctly.	Pass
59	Student check overall performance without any quiz history	1. View “Performance” section	The student haven’t taken any quiz before	N/A	Average performance is 0%, all score in each quiz are stated as N/A	Pass
60	Student	1. View “Quiz History” section	The student haven’t	N/A	Message “You do not	Pass

	check quiz history list without any quiz history		taken any quiz before		have any quiz history right now” is shown.	
61	Student check overall performance with one quiz history	1. View “Performance” section	The student has taken a quiz before	N/A	Average performance is calculated correctly, the score in taken quiz is shown, other untaken quiz remains N/A	Pass
62	Student check quiz history list with a quiz history	1. View “Quiz History” section	The student has taken a quiz before	N/A	Quiz history with quiz ID, from which quiz, total questions, correct answers, and percentage are shown	Pass
63	Student check overall performance with two quiz histories	1. View “Performance” section	The student has taken two same quizzes before	N/A	Average performance is calculated correctly, the score in taken quiz is shown as the latest one, other untaken quiz	Pass

	from same quiz				remains N/A	
64	Student check quiz history list with two quiz histories from the same quiz	1. View “Quiz History” section	The student has taken two same quizzes before	N/A	Quiz history with quiz ID, from the related quiz, total questions, correct answers, and percentage are shown. The latest quiz from the same quiz should be at the top.	Pass
65	Student check overall performance with two quiz histories from different quiz	1. View “Performance” section	The student has taken at least two different quizzes before	N/A	Average performance is calculated correctly, the score in taken quiz is shown as the latest one, other untaken quiz remains N/A	Pass
66	Student check quiz history list	1. View “Quiz History” section	The student has taken at least two different quizzes	N/A	Quiz history with quiz ID, from the related quiz, total questions, correct	Pass

	with two quiz histories from different quiz		before		answers, and percentage are shown. The latest quiz should be at the top.	
--	--	--	--------	--	--	--

Module: Online Chat Box		Test Module ID: UT-TC-010				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the online chat box system is working well.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
67	User enter chat box	1. Click Others > "Chat box" button in the navigation bar	N/A	N/A	Chat box page is loaded. Name and ID is shown, online user list is shown	Pass
68	User check online list	1. View "Online users" at the right side	Two users are online at the same time, regardless of student or admin	N/A	At least two online users are existing.	Pass
69	User send a chat	1. Type any text in the text area field. 2. Click "Send" button	N/A	Any text	Chat is sent successfully. Sent chat is shown.	Pass
70	User view	1. View the chat content box	N/A	N/A	Other user's name, time	Pass

	chat from other users				sent and chat text is shown.	
71	Admin delete chat text	1. Admin click “delete” button	The user must be admin	Any existing chat	The selected chat is deleted.	Pass
72	Automatically delete chat that is more than 3 days	1. View the chat content box	N/A	Chat which is more than 3 days	Chat that is more than 3 days should not be appeared in chat box	Fail

Module: Register Student		Test Module ID: UT-TC-011				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the register student function handles by admin is working well.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
73	Admin enter register student page	1. Click "Register Student" button in admin home screen	N/A	N/A	Register student page is loaded	Pass
74	Admin register a student	1. Key in Student ID, student name, and student email 2. Click "Register" button	N/A	Any student that has not been registered	Successful alert box is shown. Input field will be cleared.	Pass
75	Admin register a duplicate student	1. Key in existing Student ID, student name, and student email 2. Click "Register" button	N/A	Any student that has been registered	Error message is shown.	Pass
76	Admin	1. Key in Student ID, student	The student should	Invalid email:	Error message is shown.	Pass

	register a student with invalid email format	name, and student email with invalid format 2. Click “Register” button	not be registered in the system	aaa		
77	Admin register a student with invalid UTAR student email format	1. Key in Student ID, student name, and student email with invalid UTAR student email format 2. Click “Register” button	The student should not be registered in the system	Invalid email: aaa@bbb.com	Error message is shown.	Pass

Module: Student Profile		Test Module ID: UT-TC-012				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the admin profile is functioning well.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
78	Admin enter admin profile	1. Click "Admin Profile" on the navigation bar	N/A	N/A	Admin file page is loaded. Admin name and ID are displayed correctly. Search bar is shown.	Pass
79	Admin view student list	1. Click "View student list" button	N/A	N/A	Student list page is loaded. List of student ID, name, email and temporary password (if got) are shown.	Pass
80	Admin	1. Key in registered student ID	N/A	N/A	Student details (name,	Pass

	search a registered student	in search bar. 2. Click "Search" button			ID, performance and history) are shown.	
81	Admin search an unregistered student	1. Key in unregistered student ID in search bar. 2. Click "Search" button	N/A	N/A	Error alert box is shown.	Pass
82	Delete registered student	1. On the searched student, click "Delete" button 2. Click "Yes" button	N/A	N/A	Successful alert box is shown.	Pass

Module: Modifying Course Content		Test Module ID: UT-TC-013				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the course content modifications are worked properly.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
83	Admin add a topic	1. On topic list page, click “Add topic” button 2. Key in a new topic name 3. Click “Add” button	N/A	A new topic name	A new topic, exercise and quiz are created at the same time.	Pass
84	Admin edit a topic name	1. On topic list page, click “Edit” button beside the selected topic 2. Key in a new topic name 3. Click “Edit” button	N/A	A new topic name	The name of the selected topic, exercise and quiz are changed at the same time.	Pass
85	Admin delete a topic	1. On topic list page, click “Delete” button beside the selected topic	N/A	Any selected topic	The selected topic, exercise and quiz are deleted at the same time.	Pass

		2. Click “Yes” button				
86	Admin add a new topic section	1. Select a topic from the topic list 2. Click “Add a new section” button 3. Fill in necessary information. 4. Click “Add” button	N/A	A new topic section	A new topic section is created	Pass
87	Admin edit a topic section	1. Click “Edit” button from any topic section from the same topic 2. Edit any information 3. Click “Edit” button	N/A	Any selected topic section	The selected topic section is edited.	Pass
88	Admin delete a topic section	1. Click “Delete” button from any topic section from the same 2. Click “Delete” button	N/A	Any selected topic section	The selected topic section is deleted.	Pass
89	Admin add a new exercise question	1. Select an exercise from the exercise list 2. Click “Add a new question” button 3. Fill in necessary information.	N/A	A new exercise question	A new exercise question is created	Pass

		4. Click “Add” button				
90	Admin edit an exercise question	1. Click “Edit” button from any question from the same exercise 2. Edit any information 3. Click “Edit” button	N/A	Any selected exercise question	The selected exercise question is edited.	Pass
91	Admin delete an exercise question	1. Click “Delete” button from any question from the same exercise 2. Click “Delete” button	N/A	Any selected exercise question	The selected exercise question is deleted.	Pass
92	Admin add a new quiz question	1. Select a quiz from the quiz list 2. Click “Add a new question” button 3. Fill in necessary information. 4. Click “Add” button	N/A	A new quiz question	A new quiz question is created	Pass
93	Admin edit a quiz question	1. Click “Edit” button from any question from the same quiz 2. Edit any information 3. Click “Edit” button	N/A	Any selected quiz question	The selected quiz question is edited.	Pass
94	Admin delete	1. Click “Delete” button from	N/A	Any selected quiz	The selected quiz	Pass

	a quiz question	any question from the same quiz 2. Click "Delete" button		question	question is deleted.	
95	Admin add tab indent in code text section	1. Click "Add" or "Delete" from any topic section/exercise question/quiz question 2. In the section code or question text, press Tab key	N/A	Tab key	An indent is added	Pass

APPENDIX G: Integration Test Cases

Module: Password reset and onboarding system		Test Module ID: IT-TC-001				
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the password, temporary password and onboarding status data is modified in the database through a web application.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
1	Student change password in password reset page	1. Key in email, password and confirm password in password reset page 2. Click “reset” button	N/A	Existing student email, new password and new confirmation password	Related student’s password is changed in database	Pass
2	Student change password in onboarding	1. Key in Password and confirm password in onboarding page 2. Click “reset” button	The registered student account should be a new user account and login into the system.	New password	Related student’s password is changed, temporary password is set to null, and onboarding status is	Pass

	page				changed to true in database	
--	------	--	--	--	-----------------------------	--

Module: Graded quizzes			Test Module ID: IT-TC-002			
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the quiz history is added in the database through a web application.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
3	Add a quiz history	<ol style="list-style-type: none"> 1. Select a quiz from quiz list 2. Start the quiz 3. Finish all questions 4. Click "Submit button" 	Student should login into the system	Any quiz	A new quiz history record is added in the quiz history table	Pass

Module: Chat box		Test Module ID: IT-TC-003				
Created by: Lye Boon Jet		Executed by: Lye Boon Jet				
Description						
To test the chat data is modified in the database through a web application.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
4	Add a chat text	1. Type any text in the text area field. 2. Click "Send" button	User should login into the system	Any text	A new chat data is added in the chat table	Pass
5	Admin delete a chat text	1. Admin click "delete" button	The user must be admin	Any existing chat	Based on the chat ID, the chat is deleted in the chat table.	Pass

Module: Register student			Test Module ID: IT-TC-004			
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test a new student is added to the database through a web application.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
6	Add a new student	1. Key in Student ID, student name, and student email 2. Click "Register" button	N/A	Any student that has not been registered	A new student is added in the student table. The added student's temporary password is created, and the onboarding status is set to false	Pass

Module: Delete student		Test Module ID: IT-TC-005				
Created by: Lye Boon Jet		Executed by: Lye Boon Jet				
Description						
To test an existing student is deleted from the database by the administrator through a web application.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
7	Delete registered student	1. On the searched student, click "Delete" button 2. Click "Yes" button	N/A	N/A	Deleted student is not exist anymore in the student table	Pass

Module: Edit course content			Test Module ID: IT-TC-006			
Created by: Lye Boon Jet			Executed by: Lye Boon Jet			
Description						
To test the topic lesson, exercise and quiz are modified in the database by the administrator through a web application.						
Test case no.	Test scenario	Steps	Prerequisite	Test data	Expected result	Status (Pass/Fail)
8	Admin add a topic	<ol style="list-style-type: none"> 1. On topic list page, click “Add topic” button 2. Key in a new topic name 3. Click “Add” button 	N/A	A new topic name	A new topic is added in the topic title table	Pass
9	Admin edit a topic name	<ol style="list-style-type: none"> 1. On topic list page, click “Edit” button beside the selected topic 2. Key in a new topic name 3. Click “Edit” button 	N/A	A new topic name	The name of the selected topic is updated in the topic title table	Pass
10	Admin delete a topic	<ol style="list-style-type: none"> 1. On topic list page, click “Delete” button beside the selected topic 	N/A	Any selected topic	The selected topic is deleted in the topic title table. Any related topic	Pass

		2. Click “Yes” button			sections, exercise questions and quiz questions are also deleted in their respective tables.	
11	Admin add a new topic section	1. Select a topic from the topic list 2. Click “Add a new section” button 3. Fill in necessary information. 4. Click “Add” button	N/A	A new topic section	A new section is created in the topic section table.	Pass
12	Admin edit a topic section	1. Click “Edit” button from any topic section from the same topic 2. Edit any information 3. Click “Edit” button	N/A	Any selected topic section	The selected topic section data is updated in the topic section table.	Pass
13	Admin delete a topic	1. Click “Delete” button from any topic section from the same	N/A	Any selected topic section	The selected topic section data is deleted in the topic section table.	Pass

	section	2. Click “Delete” button				
14	Admin add a new exercise question	1. Select an exercise from the exercise list 2. Click “Add a new question” button 3. Fill in necessary information. 4. Click “Add” button	N/A	A new exercise question	A new exercise question is created in the exercise table.	Pass
15	Admin edit an exercise question	1. Click “Edit” button from any question from the same exercise 2. Edit any information 3. Click “Edit” button	N/A	Any selected exercise question	The selected exercise question data is edited in the exercise table.	Pass
16	Admin delete an exercise question	1. Click “Delete” button from any question from the same exercise 2. Click “Delete” button	N/A	Any selected exercise question	The selected exercise question data is deleted in the exercise table.	Pass
17	Admin add a new	1. Select a quiz from the quiz list	N/A	A new quiz question	A new quiz question is created in the quiz table	Pass

	quiz question	<ol style="list-style-type: none"> 2. Click “Add a new question” button 3. Fill in necessary information. 4. Click “Add” button 				
18	Admin edit a quiz question	<ol style="list-style-type: none"> 1. Click “Edit” button from any question from the same quiz 2. Edit any information 3. Click “Edit” button 	N/A	Any selected quiz question	The selected quiz question data is edited in the quiz table.	Pass
19	Admin delete a quiz question	<ol style="list-style-type: none"> 1. Click “Delete” button from any question from the same quiz 2. Click “Delete” button 	N/A	Any selected quiz question	The selected quiz question data is deleted in the quiz table.	Pass

APPENDIX H: Student UAT Test Cases

UAT Testing				
UAT Test Case	Tester Name: Lee Yan		Tester Type: Student	Testing Date: 23/7/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 1. Student clicks the student login button. 2. Student fills in the credentials and click login. ER: 1. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 1. Student clicks the student login button. 2. Student clicks the reset password button. 3. Student fills in the student email, new password and confirm password. 4. Student clicks the reset password button. ER: 1. Student will be redirect to student login page with the message: "Your password has been reset	Prerequisites: N/A Test Data: Student own student email and new password	Pass

		successfully!”		
TC-LI-002	Login	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks the student login button. 2. Student fills in the new credentials and click login. <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 1. Student logs to the system for the first time. 2. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 3. Student submits the new password. <p>ER:</p> <ol style="list-style-type: none"> 1. The student will be redirected into the home page. 	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks the back button once they complete the onboarding change password. <p>ER:</p> <ol style="list-style-type: none"> 1. Error 403 will be displayed. 	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p>	Pass

			Test Data: N/A	
TC-TL-001	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks the “Learn Lesson” button. 2. Student clicks any topic from the topic list. <p>ER:</p> <ol style="list-style-type: none"> 1. Student is able to view all lesson sections related to the selected topic. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 1. Student click the “Next” button once. 2. Student click the “Back” button once. <p>ER:</p> <ol style="list-style-type: none"> 1. When the “Next” button is clicked, it will be able to redirect to the next topic. 2. When the “Back” button is clicked, it will be able to redirect back to the previous topic. <p>*Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks the “Click to open the code editor for this section!” 2. Student types codes based on the instructions. 3. Student runs the code. <p>ER:</p>	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		<ol style="list-style-type: none"> 1. Paiza.io online Swift compiler will be appeared. 2. Code is able to be typed. 3. Run button is worked and output is shown. 		
TC-CP-002	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 1. Student go to navigation bar > others > code playground. 2. Student types Swift code whatever they like. 3. Student runs the code. <p>ER:</p> <ol style="list-style-type: none"> 1. Paiza.io online Swift compiler will be appeared. 2. Code is able to be typed. 3. Run button is worked and output is shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks the “Exercise” button. 2. Student clicks any exercise from the exercise list. <p>ER:</p> <ol style="list-style-type: none"> 1. Student is able to view all questions related to the selected exercise. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks check answer button in a question. 2. Student clicks check answer button again. <p>ER:</p> <ol style="list-style-type: none"> 1. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer. 	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		<ol style="list-style-type: none"> When check answer button is clicked for the second time, the submit button will be appeared, and blank for the question will be enabled and clear the correct answer. 		
TC-EE-003	Exercises	<p>TS:</p> <ol style="list-style-type: none"> Student fills in the blank with correct answer. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <ol style="list-style-type: none"> Student fills in the blank with wrong answer. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> A message “Wrong answer for some blank(s), please try again!” will be shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> Student clicks the “Graded Quizzes” button. Student clicks any quiz from the quiz list. <p>ER:</p> <ol style="list-style-type: none"> Student is able to view “Click the button below to start the quiz” . 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass

TC-GQ-002	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> 1. Students clicks the “Start” button. 2. Student do all the questions. 3. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> 1. Student is able to view the quiz result. 2. Student is able to check the correct answer in each question. 	<p>Prerequisites: Student must complete TC-GQ-001 first.</p> <p>Test Data: N/A</p>	Pass
TC-GQ-003	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks the “Start” button. 2. Student skip some questions. 3. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> 1. Alert “You have some incomplete questions.” is shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks “Retry” button. 2. Student clicks the “Start” button. 3. Student do all the questions. 4. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> 1. Student is able to retry the same quiz. 	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks “Profile” on the navigation bar. <p>ER:</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p>	Pass

		<ol style="list-style-type: none"> 1. Student can see his/her name, student ID and student email. 2. Student can view the quiz history record. 3. Student can view the performance. The performance should be the latest in each quiz. 	Test Data: N/A	
TC-CB-001	Chat Box	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks “Chat Box” button. <p>ER:</p> <ol style="list-style-type: none"> 1. Student can view the chat box interface, with his/her student ID and name. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CB-002	Chat Box	<p>TS:</p> <ol style="list-style-type: none"> 1. Student view the online user list. <p>ER:</p> <ol style="list-style-type: none"> 1. Student is able to see list of online users, including himself/herself. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CB-003	Chat Box	<p>TS:</p> <ol style="list-style-type: none"> 1. Student types some text and clicks “send” button. <p>ER:</p> <ol style="list-style-type: none"> 1. Chat text is appeared, as well as the name and datetime. 	<p>Prerequisites: N/A</p> <p>Test Data: Any text</p>	Pass
TC-LO-001	Logout	<p>TS:</p> <ol style="list-style-type: none"> 1. Student clicks the logout button. <p>ER:</p> <ol style="list-style-type: none"> 1. Student will be redirected to welcome page. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass

		2. Student is unable to access any web page that requires login.		
--	--	--	--	--

UAT Testing				
UAT Test Case	Tester Name: Justin Chia Yu Chern		Tester Type: Student	Testing Date: 23/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 3. Student clicks the student login button. 4. Student fills in the credentials and click login. ER: 2. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 5. Student clicks the student login button. 6. Student clicks the reset password button. 7. Student fills in the student email, new password and confirm password. 8. Student clicks the reset password button. ER: 2. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <ol style="list-style-type: none"> 3. Student clicks the student login button. 4. Student fills in the new credentials and click login. <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 4. Student logs in to the system for the first time. 5. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 6. Student submits the new password. <p>ER:</p> <ol style="list-style-type: none"> 2. The student will be redirected into the home page. 	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 2. Student clicks the back button once they complete the onboarding change password. <p>ER:</p> <ol style="list-style-type: none"> 2. Error 403 will be displayed. 	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 3. Student clicks the “Learn Lesson” button. 4. Student clicks any topic from the topic list. <p>ER:</p> <ol style="list-style-type: none"> 2. Student is able to view all lesson sections related to the selected topic. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 3. Student click the “Next” button once. 4. Student click the “Back” button once. <p>ER:</p> <ol style="list-style-type: none"> 3. When the “Next” button is clicked, it will be able to redirect to the next topic. 4. When the “Back” button is clicked, it will be able to redirect back to the previous topic. <p>*Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 4. Student clicks the “Click to open the code editor for this section!” 5. Student types codes based on the instructions. 6. Student runs the code. <p>ER:</p> <ol style="list-style-type: none"> 4. Paiza.io online Swift compiler will be appeared. 5. Code is able to be typed. 	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		6. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 4. Student go to navigation bar > others > code playground. 5. Student types Swift code whatever they like. 6. Student runs the code. <p>ER:</p> <ol style="list-style-type: none"> 4. Paiza.io online Swift compiler will be appeared. 5. Code is able to be typed. 6. Run button is worked and output is shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 3. Student clicks the “Exercise” button. 4. Student clicks any exercise from the exercise list. <p>ER:</p> <ol style="list-style-type: none"> 2. Student is able to view all questions related to the selected exercise. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 3. Student clicks check answer button in a question. 4. Student clicks check answer button again. <p>ER:</p> <ol style="list-style-type: none"> 3. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer. 4. When check answer button is clicked for the second time, the submit button will be appeared, 	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <ol style="list-style-type: none"> Student fills in the blank with correct answer. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <ol style="list-style-type: none"> Student fills in the blank with wrong answer. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> A message “Wrong answer for some blank(s), please try again!” will be shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> Student clicks the “Graded Quizzes” button. Student clicks any quiz from the quiz list. <p>ER:</p> <ol style="list-style-type: none"> Student is able to view “Click the button below to start the quiz” . 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> Students clicks the “Start” button. 	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<ol style="list-style-type: none"> 5. Student do all the questions. 6. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> 3. Student is able to view the quiz result. 4. Student is able to check the correct answer in each question. 	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> 4. Student clicks the “Start” button. 5. Student skip some questions. 6. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> 2. Alert “You have some incomplete questions.” is shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> 5. Student clicks “Retry” button. 6. Student clicks the “Start” button. 7. Student do all the questions. 8. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> 2. Student is able to retry the same quiz. 	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <ol style="list-style-type: none"> 2. Student clicks “Profile” on the navigation bar. <p>ER:</p> <ol style="list-style-type: none"> 4. Student can see his/her name, student ID and student email. 5. Student can view the quiz history record. 	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		6. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 2. Student clicks “Chat Box” button. ER: 2. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 2. Student view the online user list. ER: 2. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 2. Student types some text and clicks “send” button. ER: 2. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 2. Student clicks the logout button. ER: 3. Student will be redirected to welcome page. 4. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

UAT Testing				
UAT Test Case	Tester Name: Liaw Yoong Tung		Tester Type: Student	Testing Date: 23/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 5. Student clicks the student login button. 6. Student fills in the credentials and click login. ER: 3. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 9. Student clicks the student login button. 10. Student clicks the reset password button. 11. Student fills in the student email, new password and confirm password. 12. Student clicks the reset password button. ER: 3. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass
TC-LI-	Login	TS:	Prerequisites:	Pass

002		<p>5. Student clicks the student login button. 6. Student fills in the new credentials and click login.</p> <p>ER: The student will be login and redirected into the home page.</p>	<p>Student should complete reset password before login.</p> <p>Test Data: Student own ID without UEB and self-reset password</p>	
TC-OB-001	Onboarding	<p>TS: 7. Student logs in to the system for the first time. 8. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 9. Student submits the new password.</p> <p>ER: 3. The student will be redirected into the home page.</p>	<p>Prerequisites: Student should be first time login.</p> <p>Test Data: Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS: 3. Student clicks the back button once they complete the onboarding change password.</p> <p>ER: 3. Error 403 will be displayed.</p>	<p>Prerequisites: Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 5. Student clicks the “Learn Lesson” button. 6. Student clicks any topic from the topic list. <p>ER:</p> <ol style="list-style-type: none"> 3. Student is able to view all lesson sections related to the selected topic. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 5. Student click the “Next” button once. 6. Student click the “Back” button once. <p>ER:</p> <ol style="list-style-type: none"> 5. When the “Next” button is clicked, it will be able to redirect to the next topic. 6. When the “Back” button is clicked, it will be able to redirect back to the previous topic. <p>*Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 7. Student clicks the “Click to open the code editor for this section!” 8. Student types codes based on the instructions. 9. Student runs the code. <p>ER:</p> <ol style="list-style-type: none"> 7. Paiza.io online Swift compiler will be appeared. 8. Code is able to be typed. 9. Run button is worked and output is shown. 	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	<p>Fail</p> <p>[“{{{run} 18n}} (Ctrl-Enter” occurs when loading playground]</p>

TC-CP-002	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 7. Student go to navigation bar > others > code playground. 8. Student types Swift code whatever they like. 9. Student runs the code. <p>ER:</p> <ol style="list-style-type: none"> 7. Paiza.io online Swift compiler will be appeared. 8. Code is able to be typed. 9. Run button is worked and output is shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	<p>Fail</p> <p>[“{{‘run’ i18n}} (Ctrl-Enter” occurs when loading playground]</p>
TC-EE-001	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 5. Student clicks the “Exercise” button. 6. Student clicks any exercise from the exercise list. <p>ER:</p> <ol style="list-style-type: none"> 3. Student is able to view all questions related to the selected exercise. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	<p>Pass</p>
TC-EE-002	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 5. Student clicks check answer button in a question. 6. Student clicks check answer button again. <p>ER:</p> <ol style="list-style-type: none"> 5. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer. 6. When check answer button is clicked for the second time, the submit button will be appeared, 	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	<p>Pass</p>

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <ol style="list-style-type: none"> Student fills in the blank with correct answer. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <ol style="list-style-type: none"> Student fills in the blank with wrong answer. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> A message “Wrong answer for some blank(s), please try again!” will be shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> Student clicks the “Graded Quizzes” button. Student clicks any quiz from the quiz list. <p>ER:</p> <ol style="list-style-type: none"> Student is able to view “Click the button below to start the quiz” . 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> Students clicks the “Start” button. 	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		8. Student do all the questions. 9. Student clicks submit button. ER: 5. Student is able to view the quiz result. 6. Student is able to check the correct answer in each question.	Test Data: N/A	
TC-GQ-003	Graded Quizzes	TS: 7. Student clicks the “Start” button. 8. Student skip some questions. 9. Student clicks submit button. ER: 3. Alert “You have some incomplete questions.” is shown.	Prerequisites: N/A Test Data: N/A	Pass
TC-GQ-004	Graded Quizzes	TS: 9. Student clicks “Retry” button. 10. Student clicks the “Start” button. 11. Student do all the questions. 12. Student clicks submit button. ER: 3. Student is able to retry the same quiz.	Prerequisites: Student must reach the quiz result view. Test Data: N/A	Pass
TC-PF-001	Student Profile	TS: 3. Student clicks “Profile” on the navigation bar. ER: 7. Student can see his/her name, student ID and student email. 8. Student can view the quiz history record.	Prerequisites: Student must complete TC-GQ-002. Test Data: N/A	Pass

		9. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 3. Student clicks “Chat Box” button. ER: 3. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 3. Student view the online user list. ER: 3. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 3. Student types some text and clicks “send” button. ER: 3. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 3. Student clicks the logout button. ER: 5. Student will be redirected to welcome page. 6. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

UAT Testing				
UAT Test Case	Tester Name: Low Chen Wan		Tester Type: Student	Testing Date: 23/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 7. Student clicks the student login button. 8. Student fills in the credentials and click login. ER: 4. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 13. Student clicks the student login button. 14. Student clicks the reset password button. 15. Student fills in the student email, new password and confirm password. 16. Student clicks the reset password button. ER: 4. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <ol style="list-style-type: none"> 7. Student clicks the student login button. 8. Student fills in the new credentials and click login. <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 10. Student logs in to the system for the first time. 11. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 12. Student submits the new password. <p>ER:</p> <ol style="list-style-type: none"> 4. The student will be redirected into the home page. 	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 4. Student clicks the back button once they complete the onboarding change password. <p>ER:</p> <ol style="list-style-type: none"> 4. Error 403 will be displayed. 	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 7. Student clicks the “Learn Lesson” button. 8. Student clicks any topic from the topic list. <p>ER:</p> <ol style="list-style-type: none"> 4. Student is able to view all lesson sections related to the selected topic. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 7. Student click the “Next” button once. 8. Student click the “Back” button once. <p>ER:</p> <ol style="list-style-type: none"> 7. When the “Next” button is clicked, it will be able to redirect to the next topic. 8. When the “Back” button is clicked, it will be able to redirect back to the previous topic. <p>*Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 10. Student clicks the “Click to open the code editor for this section!” 11. Student types codes based on the instructions. 12. Student runs the code. <p>ER:</p> <ol style="list-style-type: none"> 10. Paiza.io online Swift compiler will be appeared. 11. Code is able to be typed. 	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		12. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 10. Student go to navigation bar > others > code playground. 11. Student types Swift code whatever they like. 12. Student runs the code. <p>ER:</p> <ol style="list-style-type: none"> 10. Paiza.io online Swift compiler will be appeared. 11. Code is able to be typed. 12. Run button is worked and output is shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 7. Student clicks the “Exercise” button. 8. Student clicks any exercise from the exercise list. <p>ER:</p> <ol style="list-style-type: none"> 4. Student is able to view all questions related to the selected exercise. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 7. Student clicks check answer button in a question. 8. Student clicks check answer button again. <p>ER:</p> <ol style="list-style-type: none"> 7. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer. 8. When check answer button is clicked for the second time, the submit button will be appeared, 	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 7. Student fills in the blank with correct answer. 8. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> 4. Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <ol style="list-style-type: none"> 7. Student fills in the blank with wrong answer. 8. Student clicks submit button. <p>ER:</p> <ol style="list-style-type: none"> 4. A message “Wrong answer for some blank(s), please try again!” will be shown. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> 7. Student clicks the “Graded Quizzes” button. 8. Student clicks any quiz from the quiz list. <p>ER:</p> <ol style="list-style-type: none"> 4. Student is able to view “Click the button below to start the quiz” . 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> 10. Students clicks the “Start” button. 	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<p>11. Student do all the questions. 12. Student clicks submit button.</p> <p>ER:</p> <p>7. Student is able to view the quiz result. 8. Student is able to check the correct answer in each question.</p>	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <p>10. Student clicks the “Start” button. 11. Student skip some questions. 12. Student clicks submit button.</p> <p>ER:</p> <p>4. Alert “You have some incomplete questions.” is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <p>13. Student clicks “Retry” button. 14. Student clicks the “Start” button. 15. Student do all the questions. 16. Student clicks submit button.</p> <p>ER:</p> <p>4. Student is able to retry the same quiz.</p>	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <p>4. Student clicks “Profile” on the navigation bar.</p> <p>ER:</p> <p>10. Student can see his/her name, student ID and student email. 11. Student can view the quiz history record.</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		12. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 4. Student clicks “Chat Box” button. ER: 4. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 4. Student view the online user list. ER: 4. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 4. Student types some text and clicks “send” button. ER: 4. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 4. Student clicks the logout button. ER: 7. Student will be redirected to welcome page. 8. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

UAT Testing				
UAT Test Case	Tester Name: Siew Liang Han		Tester Type: Student	Testing Date: 23/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 9. Student clicks the student login button. 10. Student fills in the credentials and click login. ER: 5. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 17. Student clicks the student login button. 18. Student clicks the reset password button. 19. Student fills in the student email, new password and confirm password. 20. Student clicks the reset password button. ER: 5. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <ol style="list-style-type: none"> 9. Student clicks the student login button. 10. Student fills in the new credentials and click login. <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 13. Student logs in to the system for the first time. 14. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 15. Student submits the new password. <p>ER:</p> <ol style="list-style-type: none"> 5. The student will be redirected into the home page. 	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 5. Student clicks the back button once they complete the onboarding change password. <p>ER:</p> <ol style="list-style-type: none"> 5. Error 403 will be displayed. 	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 9. Student clicks the “Learn Lesson” button. 10. Student clicks any topic from the topic list. <p>ER:</p> <ol style="list-style-type: none"> 5. Student is able to view all lesson sections related to the selected topic. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 9. Student click the “Next” button once. 10. Student click the “Back” button once. <p>ER:</p> <ol style="list-style-type: none"> 9. When the “Next” button is clicked, it will be able to redirect to the next topic. 10. When the “Back” button is clicked, it will be able to redirect back to the previous topic. <p>*Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 13. Student clicks the “Click to open the code editor for this section!” 14. Student types codes based on the instructions. 15. Student runs the code. <p>ER:</p> <ol style="list-style-type: none"> 13. Paiza.io online Swift compiler will be appeared. 14. Code is able to be typed. 	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		15. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <p>13. Student go to navigation bar > others > code playground.</p> <p>14. Student types Swift code whatever they like.</p> <p>15. Student runs the code.</p> <p>ER:</p> <p>13. Paiza.io online Swift compiler will be appeared.</p> <p>14. Code is able to be typed.</p> <p>15. Run button is worked and output is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <p>9. Student clicks the “Exercise” button.</p> <p>10. Student clicks any exercise from the exercise list.</p> <p>ER:</p> <p>5. Student is able to view all questions related to the selected exercise.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <p>9. Student clicks check answer button in a question.</p> <p>10. Student clicks check answer button again.</p> <p>ER:</p> <p>9. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer.</p> <p>10. When check answer button is clicked for the second time, the submit button will be appeared,</p>	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <p>9. Student fills in the blank with correct answer. 10. Student clicks submit button.</p> <p>ER:</p> <p>5. Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <p>9. Student fills in the blank with wrong answer. 10. Student clicks submit button.</p> <p>ER:</p> <p>5. A message “Wrong answer for some blank(s), please try again!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <p>9. Student clicks the “Graded Quizzes” button. 10. Student clicks any quiz from the quiz list.</p> <p>ER:</p> <p>5. Student is able to view “Click the button below to start the quiz” .</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <p>13. Students clicks the “Start” button.</p>	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<p>14. Student do all the questions. 15. Student clicks submit button.</p> <p>ER:</p> <p>9. Student is able to view the quiz result. 10. Student is able to check the correct answer in each question.</p>	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <p>13. Student clicks the “Start” button. 14. Student skip some questions. 15. Student clicks submit button.</p> <p>ER:</p> <p>5. Alert “You have some incomplete questions.” is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <p>17. Student clicks “Retry” button. 18. Student clicks the “Start” button. 19. Student do all the questions. 20. Student clicks submit button.</p> <p>ER:</p> <p>5. Student is able to retry the same quiz.</p>	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <p>5. Student clicks “Profile” on the navigation bar.</p> <p>ER:</p> <p>13. Student can see his/her name, student ID and student email. 14. Student can view the quiz history record.</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		15. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 5. Student clicks “Chat Box” button. ER: 5. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 5. Student view the online user list. ER: 5. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 5. Student types some text and clicks “send” button. ER: 5. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 5. Student clicks the logout button. ER: 9. Student will be redirected to welcome page. 10. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass [Remarks: Still able to go back to Chat page by clicking

				back button on the browser]
--	--	--	--	--------------------------------

UAT Testing				
UAT Test Case	Tester Name: Soh Yan Wei		Tester Type: Student	Testing Date: 24/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 11. Student clicks the student login button. 12. Student fills in the credentials and click login. ER: 6. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 21. Student clicks the student login button. 22. Student clicks the reset password button. 23. Student fills in the student email, new password and confirm password. 24. Student clicks the reset password button. ER: 6. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <ol style="list-style-type: none"> 11. Student clicks the student login button. 12. Student fills in the new credentials and click login. <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 16. Student logs in to the system for the first time. 17. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 18. Student submits the new password. <p>ER:</p> <ol style="list-style-type: none"> 6. The student will be redirected into the home page. 	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 6. Student clicks the back button once they complete the onboarding change password. <p>ER:</p> <ol style="list-style-type: none"> 6. Error 403 will be displayed. 	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <p>11. Student clicks the “Learn Lesson” button. 12. Student clicks any topic from the topic list.</p> <p>ER:</p> <p>6. Student is able to view all lesson sections related to the selected topic.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <p>11. Student click the “Next” button once. 12. Student click the “Back” button once.</p> <p>ER:</p> <p>11. When the “Next” button is clicked, it will be able to redirect to the next topic. 12. When the “Back” button is clicked, it will be able to redirect back to the previous topic. *Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <p>16. Student clicks the “Click to open the code editor for this section!” 17. Student types codes based on the instructions. 18. Student runs the code.</p> <p>ER:</p> <p>16. Paiza.io online Swift compiler will be appeared. 17. Code is able to be typed.</p>	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		18. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <p>16. Student go to navigation bar > others > code playground.</p> <p>17. Student types Swift code whatever they like.</p> <p>18. Student runs the code.</p> <p>ER:</p> <p>16. Paiza.io online Swift compiler will be appeared.</p> <p>17. Code is able to be typed.</p> <p>18. Run button is worked and output is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <p>11. Student clicks the “Exercise” button.</p> <p>12. Student clicks any exercise from the exercise list.</p> <p>ER:</p> <p>6. Student is able to view all questions related to the selected exercise.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <p>11. Student clicks check answer button in a question.</p> <p>12. Student clicks check answer button again.</p> <p>ER:</p> <p>11. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer.</p> <p>12. When check answer button is clicked for the second time, the submit button will be appeared,</p>	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <p>11. Student fills in the blank with correct answer. 12. Student clicks submit button.</p> <p>ER:</p> <p>6. Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <p>11. Student fills in the blank with wrong answer. 12. Student clicks submit button.</p> <p>ER:</p> <p>6. A message “Wrong answer for some blank(s), please try again!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <p>11. Student clicks the “Graded Quizzes” button. 12. Student clicks any quiz from the quiz list.</p> <p>ER:</p> <p>6. Student is able to view “Click the button below to start the quiz” .</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <p>16. Students clicks the “Start” button.</p>	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<p>17. Student do all the questions. 18. Student clicks submit button.</p> <p>ER:</p> <p>11. Student is able to view the quiz result. 12. Student is able to check the correct answer in each question.</p>	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <p>16. Student clicks the “Start” button. 17. Student skip some questions. 18. Student clicks submit button.</p> <p>ER:</p> <p>6. Alert “You have some incomplete questions.” is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <p>21. Student clicks “Retry” button. 22. Student clicks the “Start” button. 23. Student do all the questions. 24. Student clicks submit button.</p> <p>ER:</p> <p>6. Student is able to retry the same quiz.</p>	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <p>6. Student clicks “Profile” on the navigation bar.</p> <p>ER:</p> <p>16. Student can see his/her name, student ID and student email. 17. Student can view the quiz history record.</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		18. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 6. Student clicks “Chat Box” button. ER: 6. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 6. Student view the online user list. ER: 6. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 6. Student types some text and clicks “send” button. ER: 6. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 6. Student clicks the logout button. ER: 11. Student will be redirected to welcome page. 12. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

UAT Testing				
UAT Test Case	Tester Name: Thian Qi Wee		Tester Type: Student	Testing Date: 24/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 13. Student clicks the student login button. 14. Student fills in the credentials and click login. ER: 7. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 25. Student clicks the student login button. 26. Student clicks the reset password button. 27. Student fills in the student email, new password and confirm password. 28. Student clicks the reset password button. ER: 7. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <ul style="list-style-type: none"> 13. Student clicks the student login button. 14. Student fills in the new credentials and click login. <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <ul style="list-style-type: none"> 19. Student logs in to the system for the first time. 20. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 21. Student submits the new password. <p>ER:</p> <ul style="list-style-type: none"> 7. The student will be redirected into the home page. 	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <ul style="list-style-type: none"> 7. Student clicks the back button once they complete the onboarding change password. <p>ER:</p> <ul style="list-style-type: none"> 7. Error 403 will be displayed. 	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <p>13. Student clicks the “Learn Lesson” button. 14. Student clicks any topic from the topic list.</p> <p>ER:</p> <p>7. Student is able to view all lesson sections related to the selected topic.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <p>13. Student click the “Next” button once. 14. Student click the “Back” button once.</p> <p>ER:</p> <p>13. When the “Next” button is clicked, it will be able to redirect to the next topic. 14. When the “Back” button is clicked, it will be able to redirect back to the previous topic. *Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <p>19. Student clicks the “Click to open the code editor for this section!” 20. Student types codes based on the instructions. 21. Student runs the code.</p> <p>ER:</p> <p>19. Paiza.io online Swift compiler will be appeared. 20. Code is able to be typed.</p>	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		21. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <p>19. Student go to navigation bar > others > code playground.</p> <p>20. Student types Swift code whatever they like.</p> <p>21. Student runs the code.</p> <p>ER:</p> <p>19. Paiza.io online Swift compiler will be appeared.</p> <p>20. Code is able to be typed.</p> <p>21. Run button is worked and output is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <p>13. Student clicks the “Exercise” button.</p> <p>14. Student clicks any exercise from the exercise list.</p> <p>ER:</p> <p>7. Student is able to view all questions related to the selected exercise.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <p>13. Student clicks check answer button in a question.</p> <p>14. Student clicks check answer button again.</p> <p>ER:</p> <p>13. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer.</p> <p>14. When check answer button is clicked for the second time, the submit button will be appeared,</p>	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <p>13. Student fills in the blank with correct answer. 14. Student clicks submit button.</p> <p>ER:</p> <p>7. Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <p>13. Student fills in the blank with wrong answer. 14. Student clicks submit button.</p> <p>ER:</p> <p>7. A message “Wrong answer for some blank(s), please try again!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <p>13. Student clicks the “Graded Quizzes” button. 14. Student clicks any quiz from the quiz list.</p> <p>ER:</p> <p>7. Student is able to view “Click the button below to start the quiz” .</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <p>19. Students clicks the “Start” button.</p>	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<p>20. Student do all the questions. 21. Student clicks submit button.</p> <p>ER:</p> <p>13. Student is able to view the quiz result. 14. Student is able to check the correct answer in each question.</p>	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <p>19. Student clicks the “Start” button. 20. Student skip some questions. 21. Student clicks submit button.</p> <p>ER:</p> <p>7. Alert “You have some incomplete questions.” is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <p>25. Student clicks “Retry” button. 26. Student clicks the “Start” button. 27. Student do all the questions. 28. Student clicks submit button.</p> <p>ER:</p> <p>7. Student is able to retry the same quiz.</p>	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <p>7. Student clicks “Profile” on the navigation bar.</p> <p>ER:</p> <p>19. Student can see his/her name, student ID and student email. 20. Student can view the quiz history record.</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		21. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 7. Student clicks “Chat Box” button. ER: 7. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 7. Student view the online user list. ER: 7. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 7. Student types some text and clicks “send” button. ER: 7. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 7. Student clicks the logout button. ER: 13. Student will be redirected to welcome page. 14. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

UAT Testing				
UAT Test Case	Tester Name: Yeoh Chee Wei		Tester Type: Student	Testing Date: 23/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 15. Student clicks the student login button. 16. Student fills in the credentials and click login. ER: 8. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 29. Student clicks the student login button. 30. Student clicks the reset password button. 31. Student fills in the student email, new password and confirm password. 32. Student clicks the reset password button. ER: 8. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <p>15. Student clicks the student login button. 16. Student fills in the new credentials and click login.</p> <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <p>22. Student logs in to the system for the first time. 23. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 24. Student submits the new password.</p> <p>ER:</p> <p>8. The student will be redirected into the home page.</p>	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <p>8. Student clicks the back button once they complete the onboarding change password.</p> <p>ER:</p> <p>8. Error 403 will be displayed.</p>	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <p>15. Student clicks the “Learn Lesson” button. 16. Student clicks any topic from the topic list.</p> <p>ER:</p> <p>8. Student is able to view all lesson sections related to the selected topic.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <p>15. Student click the “Next” button once. 16. Student click the “Back” button once.</p> <p>ER:</p> <p>15. When the “Next” button is clicked, it will be able to redirect to the next topic. 16. When the “Back” button is clicked, it will be able to redirect back to the previous topic. *Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <p>22. Student clicks the “Click to open the code editor for this section!” 23. Student types codes based on the instructions. 24. Student runs the code.</p> <p>ER:</p> <p>22. Paiza.io online Swift compiler will be appeared. 23. Code is able to be typed.</p>	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		24. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <p>22. Student go to navigation bar > others > code playground.</p> <p>23. Student types Swift code whatever they like.</p> <p>24. Student runs the code.</p> <p>ER:</p> <p>22. Paiza.io online Swift compiler will be appeared.</p> <p>23. Code is able to be typed.</p> <p>24. Run button is worked and output is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <p>15. Student clicks the “Exercise” button.</p> <p>16. Student clicks any exercise from the exercise list.</p> <p>ER:</p> <p>8. Student is able to view all questions related to the selected exercise.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <p>15. Student clicks check answer button in a question.</p> <p>16. Student clicks check answer button again.</p> <p>ER:</p> <p>15. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer.</p> <p>16. When check answer button is clicked for the second time, the submit button will be appeared,</p>	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <p>15. Student fills in the blank with correct answer. 16. Student clicks submit button.</p> <p>ER:</p> <p>8. Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <p>15. Student fills in the blank with wrong answer. 16. Student clicks submit button.</p> <p>ER:</p> <p>8. A message “Wrong answer for some blank(s), please try again!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <p>15. Student clicks the “Graded Quizzes” button. 16. Student clicks any quiz from the quiz list.</p> <p>ER:</p> <p>8. Student is able to view “Click the button below to start the quiz” .</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <p>22. Students clicks the “Start” button.</p>	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<p>23. Student do all the questions. 24. Student clicks submit button.</p> <p>ER:</p> <p>15. Student is able to view the quiz result. 16. Student is able to check the correct answer in each question.</p>	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <p>22. Student clicks the “Start” button. 23. Student skip some questions. 24. Student clicks submit button.</p> <p>ER:</p> <p>8. Alert “You have some incomplete questions.” is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <p>29. Student clicks “Retry” button. 30. Student clicks the “Start” button. 31. Student do all the questions. 32. Student clicks submit button.</p> <p>ER:</p> <p>8. Student is able to retry the same quiz.</p>	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <p>8. Student clicks “Profile” on the navigation bar.</p> <p>ER:</p> <p>22. Student can see his/her name, student ID and student email. 23. Student can view the quiz history record.</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		24. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 8. Student clicks “Chat Box” button. ER: 8. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 8. Student view the online user list. ER: 8. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 8. Student types some text and clicks “send” button. ER: 8. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 8. Student clicks the logout button. ER: 15. Student will be redirected to welcome page. 16. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

UAT Testing				
UAT Test Case	Tester Name: Yong Yung Shen		Tester Type: Student	Testing Date: 23/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 17. Student clicks the student login button. 18. Student fills in the credentials and click login. ER: 9. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 33. Student clicks the student login button. 34. Student clicks the reset password button. 35. Student fills in the student email, new password and confirm password. 36. Student clicks the reset password button. ER: 9. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <ul style="list-style-type: none"> 17. Student clicks the student login button. 18. Student fills in the new credentials and click login. <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <ul style="list-style-type: none"> 25. Student logs in to the system for the first time. 26. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 27. Student submits the new password. <p>ER:</p> <ul style="list-style-type: none"> 9. The student will be redirected into the home page. 	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <ul style="list-style-type: none"> 9. Student clicks the back button once they complete the onboarding change password. <p>ER:</p> <ul style="list-style-type: none"> 9. Error 403 will be displayed. 	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <p>17. Student clicks the “Learn Lesson” button. 18. Student clicks any topic from the topic list.</p> <p>ER:</p> <p>9. Student is able to view all lesson sections related to the selected topic.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <p>17. Student click the “Next” button once. 18. Student click the “Back” button once.</p> <p>ER:</p> <p>17. When the “Next” button is clicked, it will be able to redirect to the next topic. 18. When the “Back” button is clicked, it will be able to redirect back to the previous topic. *Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <p>25. Student clicks the “Click to open the code editor for this section!” 26. Student types codes based on the instructions. 27. Student runs the code.</p> <p>ER:</p> <p>25. Paiza.io online Swift compiler will be appeared. 26. Code is able to be typed.</p>	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		27. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <p>25. Student go to navigation bar > others > code playground.</p> <p>26. Student types Swift code whatever they like.</p> <p>27. Student runs the code.</p> <p>ER:</p> <p>25. Paiza.io online Swift compiler will be appeared.</p> <p>26. Code is able to be typed.</p> <p>27. Run button is worked and output is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <p>17. Student clicks the “Exercise” button.</p> <p>18. Student clicks any exercise from the exercise list.</p> <p>ER:</p> <p>9. Student is able to view all questions related to the selected exercise.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <p>17. Student clicks check answer button in a question.</p> <p>18. Student clicks check answer button again.</p> <p>ER:</p> <p>17. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer.</p> <p>18. When check answer button is clicked for the second time, the submit button will be appeared,</p>	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <p>17. Student fills in the blank with correct answer. 18. Student clicks submit button.</p> <p>ER:</p> <p>9. Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <p>17. Student fills in the blank with wrong answer. 18. Student clicks submit button.</p> <p>ER:</p> <p>9. A message “Wrong answer for some blank(s), please try again!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <p>17. Student clicks the “Graded Quizzes” button. 18. Student clicks any quiz from the quiz list.</p> <p>ER:</p> <p>9. Student is able to view “Click the button below to start the quiz” .</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <p>25. Students clicks the “Start” button.</p>	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<p>26. Student do all the questions. 27. Student clicks submit button.</p> <p>ER:</p> <p>17. Student is able to view the quiz result. 18. Student is able to check the correct answer in each question.</p>	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <p>25. Student clicks the “Start” button. 26. Student skip some questions. 27. Student clicks submit button.</p> <p>ER:</p> <p>9. Alert “You have some incomplete questions.” is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <p>33. Student clicks “Retry” button. 34. Student clicks the “Start” button. 35. Student do all the questions. 36. Student clicks submit button.</p> <p>ER:</p> <p>9. Student is able to retry the same quiz.</p>	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <p>9. Student clicks “Profile” on the navigation bar.</p> <p>ER:</p> <p>25. Student can see his/her name, student ID and student email. 26. Student can view the quiz history record.</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		27. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 9. Student clicks “Chat Box” button. ER: 9. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 9. Student view the online user list. ER: 9. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 9. Student types some text and clicks “send” button. ER: 9. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 9. Student clicks the logout button. ER: 17. Student will be redirected to welcome page. 18. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

UAT Testing				
UAT Test Case	Tester Name: Benjamin Leong E-Jenn		Tester Type: Student	Testing Date: 25/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 19. Student clicks the student login button. 20. Student fills in the credentials and click login. ER: 10. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 37. Student clicks the student login button. 38. Student clicks the reset password button. 39. Student fills in the student email, new password and confirm password. 40. Student clicks the reset password button. ER: 10. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <p>19. Student clicks the student login button. 20. Student fills in the new credentials and click login.</p> <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <p>28. Student logs in to the system for the first time. 29. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 30. Student submits the new password.</p> <p>ER:</p> <p>10. The student will be redirected into the home page.</p>	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <p>10. Student clicks the back button once they complete the onboarding change password.</p> <p>ER:</p> <p>10. Error 403 will be displayed.</p>	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <p>19. Student clicks the “Learn Lesson” button. 20. Student clicks any topic from the topic list.</p> <p>ER:</p> <p>10. Student is able to view all lesson sections related to the selected topic.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <p>19. Student click the “Next” button once. 20. Student click the “Back” button once.</p> <p>ER:</p> <p>19. When the “Next” button is clicked, it will be able to redirect to the next topic. 20. When the “Back” button is clicked, it will be able to redirect back to the previous topic. *Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <p>28. Student clicks the “Click to open the code editor for this section!” 29. Student types codes based on the instructions. 30. Student runs the code.</p> <p>ER:</p> <p>28. Paiza.io online Swift compiler will be appeared. 29. Code is able to be typed.</p>	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		30. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <p>28. Student go to navigation bar > others > code playground.</p> <p>29. Student types Swift code whatever they like.</p> <p>30. Student runs the code.</p> <p>ER:</p> <p>28. Paiza.io online Swift compiler will be appeared.</p> <p>29. Code is able to be typed.</p> <p>30. Run button is worked and output is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <p>19. Student clicks the “Exercise” button.</p> <p>20. Student clicks any exercise from the exercise list.</p> <p>ER:</p> <p>10. Student is able to view all questions related to the selected exercise.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <p>19. Student clicks check answer button in a question.</p> <p>20. Student clicks check answer button again.</p> <p>ER:</p> <p>19. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer.</p> <p>20. When check answer button is clicked for the second time, the submit button will be appeared,</p>	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <p>19. Student fills in the blank with correct answer. 20. Student clicks submit button.</p> <p>ER:</p> <p>10. Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <p>19. Student fills in the blank with wrong answer. 20. Student clicks submit button.</p> <p>ER:</p> <p>10. A message “Wrong answer for some blank(s), please try again!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <p>19. Student clicks the “Graded Quizzes” button. 20. Student clicks any quiz from the quiz list.</p> <p>ER:</p> <p>10. Student is able to view “Click the button below to start the quiz” .</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <p>28. Students clicks the “Start” button.</p>	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<p>29. Student do all the questions. 30. Student clicks submit button.</p> <p>ER:</p> <p>19. Student is able to view the quiz result. 20. Student is able to check the correct answer in each question.</p>	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <p>28. Student clicks the “Start” button. 29. Student skip some questions. 30. Student clicks submit button.</p> <p>ER:</p> <p>10. Alert “You have some incomplete questions.” is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <p>37. Student clicks “Retry” button. 38. Student clicks the “Start” button. 39. Student do all the questions. 40. Student clicks submit button.</p> <p>ER:</p> <p>10. Student is able to retry the same quiz.</p>	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <p>10. Student clicks “Profile” on the navigation bar.</p> <p>ER:</p> <p>28. Student can see his/her name, student ID and student email. 29. Student can view the quiz history record.</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		30. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 10. Student clicks “Chat Box” button. ER: 10. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 10. Student view the online user list. ER: 10. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 10. Student types some text and clicks “send” button. ER: 10. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 10. Student clicks the logout button. ER: 19. Student will be redirected to welcome page. 20. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

UAT Testing				
UAT Test Case	Tester Name: Chua Qing Wen		Tester Type: Student	Testing Date: 23/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 21. Student clicks the student login button. 22. Student fills in the credentials and click login. ER: 11. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 41. Student clicks the student login button. 42. Student clicks the reset password button. 43. Student fills in the student email, new password and confirm password. 44. Student clicks the reset password button. ER: 11. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <ol style="list-style-type: none"> 21. Student clicks the student login button. 22. Student fills in the new credentials and click login. <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 31. Student logs in to the system for the first time. 32. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 33. Student submits the new password. <p>ER:</p> <ol style="list-style-type: none"> 11. The student will be redirected into the home page. 	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <ol style="list-style-type: none"> 11. Student clicks the back button once they complete the onboarding change password. <p>ER:</p> <ol style="list-style-type: none"> 11. Error 403 will be displayed. 	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 21. Student clicks the “Learn Lesson” button. 22. Student clicks any topic from the topic list. <p>ER:</p> <ol style="list-style-type: none"> 11. Student is able to view all lesson sections related to the selected topic. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <ol style="list-style-type: none"> 21. Student click the “Next” button once. 22. Student click the “Back” button once. <p>ER:</p> <ol style="list-style-type: none"> 21. When the “Next” button is clicked, it will be able to redirect to the next topic. 22. When the “Back” button is clicked, it will be able to redirect back to the previous topic. <p>*Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 31. Student clicks the “Click to open the code editor for this section!” 32. Student types codes based on the instructions. 33. Student runs the code. <p>ER:</p> <ol style="list-style-type: none"> 31. Paiza.io online Swift compiler will be appeared. 32. Code is able to be typed. 	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		33. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <p>31. Student go to navigation bar > others > code playground. 32. Student types Swift code whatever they like. 33. Student runs the code.</p> <p>ER:</p> <p>31. Paiza.io online Swift compiler will be appeared. 32. Code is able to be typed. 33. Run button is worked and output is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <p>21. Student clicks the “Exercise” button. 22. Student clicks any exercise from the exercise list.</p> <p>ER:</p> <p>11. Student is able to view all questions related to the selected exercise.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <p>21. Student clicks check answer button in a question. 22. Student clicks check answer button again.</p> <p>ER:</p> <p>21. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer. 22. When check answer button is clicked for the second time, the submit button will be appeared,</p>	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <p>21. Student fills in the blank with correct answer. 22. Student clicks submit button.</p> <p>ER:</p> <p>11. Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <p>21. Student fills in the blank with wrong answer. 22. Student clicks submit button.</p> <p>ER:</p> <p>11. A message “Wrong answer for some blank(s), please try again!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <p>21. Student clicks the “Graded Quizzes” button. 22. Student clicks any quiz from the quiz list.</p> <p>ER:</p> <p>11. Student is able to view “Click the button below to start the quiz” .</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <p>31. Students clicks the “Start” button.</p>	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<p>32. Student do all the questions. 33. Student clicks submit button.</p> <p>ER:</p> <p>21. Student is able to view the quiz result. 22. Student is able to check the correct answer in each question.</p>	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <p>31. Student clicks the “Start” button. 32. Student skip some questions. 33. Student clicks submit button.</p> <p>ER:</p> <p>11. Alert “You have some incomplete questions.” is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <p>41. Student clicks “Retry” button. 42. Student clicks the “Start” button. 43. Student do all the questions. 44. Student clicks submit button.</p> <p>ER:</p> <p>11. Student is able to retry the same quiz.</p>	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <p>11. Student clicks “Profile” on the navigation bar.</p> <p>ER:</p> <p>31. Student can see his/her name, student ID and student email. 32. Student can view the quiz history record.</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		33. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 11. Student clicks “Chat Box” button. ER: 11. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 11. Student view the online user list. ER: 11. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 11. Student types some text and clicks “send” button. ER: 11. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 11. Student clicks the logout button. ER: 21. Student will be redirected to welcome page. 22. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

UAT Testing				
UAT Test Case	Tester Name: Wong Yuk Han		Tester Type: Student	Testing Date: 23/07/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LI-001	Login	TS: 23. Student clicks the student login button. 24. Student fills in the credentials and click login. ER: 12. The student will be login and redirected into the home page.	Prerequisites: Student should not be first time login. Test Data: Student own ID without UEB and self-reset password.	Pass
TC-RP-001	Reset Password	TS: 45. Student clicks the student login button. 46. Student clicks the reset password button. 47. Student fills in the student email, new password and confirm password. 48. Student clicks the reset password button. ER: 12. Student will be redirect to student login page with the message: "Your password has been reset successfully!"	Prerequisites: N/A Test Data: Student own student email and new password	Pass

TC-LI-002	Login	<p>TS:</p> <p>23. Student clicks the student login button. 24. Student fills in the new credentials and click login.</p> <p>ER:</p> <p>The student will be login and redirected into the home page.</p>	<p>Prerequisites:</p> <p>Student should complete reset password before login.</p> <p>Test Data:</p> <p>Student own ID without UEB and self-reset password</p>	Pass
TC-OB-001	Onboarding	<p>TS:</p> <p>34. Student logs in to the system for the first time. 35. Student sets a new password with minimum 8 characters with combination of letters and numeric characters. 36. Student submits the new password.</p> <p>ER:</p> <p>12. The student will be redirected into the home page.</p>	<p>Prerequisites: Student should be first time login.</p> <p>Test Data:</p> <p>Student own ID without UEB and temporary password.</p>	Pass
TC-OB-002	Onboarding	<p>TS:</p> <p>12. Student clicks the back button once they complete the onboarding change password.</p> <p>ER:</p> <p>12. Error 403 will be displayed.</p>	<p>Prerequisites:</p> <p>Student should complete onboarding procedure.</p> <p>Test Data: N/A</p>	Pass

TC-TL-001	Learn Lesson	<p>TS:</p> <p>23. Student clicks the “Learn Lesson” button. 24. Student clicks any topic from the topic list.</p> <p>ER:</p> <p>12. Student is able to view all lesson sections related to the selected topic.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TL-002	Learn Lesson	<p>TS:</p> <p>23. Student click the “Next” button once. 24. Student click the “Back” button once.</p> <p>ER:</p> <p>23. When the “Next” button is clicked, it will be able to redirect to the next topic. 24. When the “Back” button is clicked, it will be able to redirect back to the previous topic. *Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CP-001	Code Playground	<p>TS:</p> <p>34. Student clicks the “Click to open the code editor for this section!” 35. Student types codes based on the instructions. 36. Student runs the code.</p> <p>ER:</p> <p>34. Paiza.io online Swift compiler will be appeared. 35. Code is able to be typed.</p>	<p>Prerequisites: Student is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		36. Run button is worked and output is shown.		
TC-CP-002	Code Playground	<p>TS:</p> <p>34. Student go to navigation bar > others > code playground.</p> <p>35. Student types Swift code whatever they like.</p> <p>36. Student runs the code.</p> <p>ER:</p> <p>34. Paiza.io online Swift compiler will be appeared.</p> <p>35. Code is able to be typed.</p> <p>36. Run button is worked and output is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-001	Exercises	<p>TS:</p> <p>23. Student clicks the “Exercise” button.</p> <p>24. Student clicks any exercise from the exercise list.</p> <p>ER:</p> <p>12. Student is able to view all questions related to the selected exercise.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-002	Exercises	<p>TS:</p> <p>23. Student clicks check answer button in a question.</p> <p>24. Student clicks check answer button again.</p> <p>ER:</p> <p>23. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer.</p> <p>24. When check answer button is clicked for the second time, the submit button will be appeared,</p>	<p>Prerequisites: Student must complete TC-EE-001 first.</p> <p>Test Data: N/A</p>	Pass

		and blank for the question will be enabled and clear the correct answer.		
TC-EE-003	Exercises	<p>TS:</p> <p>23. Student fills in the blank with correct answer. 24. Student clicks submit button.</p> <p>ER:</p> <p>12. Submit button and check answer button will be hidden, and blank for the question will be disabled and show the correct answer. A message “All of your answers are correct!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EE-004	Exercises	<p>TS:</p> <p>23. Student fills in the blank with wrong answer. 24. Student clicks submit button.</p> <p>ER:</p> <p>12. A message “Wrong answer for some blank(s), please try again!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-001	Graded Quizzes	<p>TS:</p> <p>23. Student clicks the “Graded Quizzes” button. 24. Student clicks any quiz from the quiz list.</p> <p>ER:</p> <p>12. Student is able to view “Click the button below to start the quiz” .</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-002	Graded Quizzes	<p>TS:</p> <p>34. Students clicks the “Start” button.</p>	<p>Prerequisites: Student must complete TC-GQ-001 first.</p>	Pass

		<p>35. Student do all the questions. 36. Student clicks submit button.</p> <p>ER:</p> <p>23. Student is able to view the quiz result. 24. Student is able to check the correct answer in each question.</p>	Test Data: N/A	
TC-GQ-003	Graded Quizzes	<p>TS:</p> <p>34. Student clicks the “Start” button. 35. Student skip some questions. 36. Student clicks submit button.</p> <p>ER:</p> <p>12. Alert “You have some incomplete questions.” is shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQ-004	Graded Quizzes	<p>TS:</p> <p>45. Student clicks “Retry” button. 46. Student clicks the “Start” button. 47. Student do all the questions. 48. Student clicks submit button.</p> <p>ER:</p> <p>12. Student is able to retry the same quiz.</p>	<p>Prerequisites: Student must reach the quiz result view.</p> <p>Test Data: N/A</p>	Pass
TC-PF-001	Student Profile	<p>TS:</p> <p>12. Student clicks “Profile” on the navigation bar.</p> <p>ER:</p> <p>34. Student can see his/her name, student ID and student email. 35. Student can view the quiz history record.</p>	<p>Prerequisites: Student must complete TC-GQ-002.</p> <p>Test Data: N/A</p>	Pass

		36. Student can view the performance. The performance should be the latest in each quiz.		
TC-CB-001	Chat Box	TS: 12. Student clicks “Chat Box” button. ER: 12. Student can view the chat box interface, with his/her student ID and name.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-002	Chat Box	TS: 12. Student view the online user list. ER: 12. Student is able to see list of online users, including himself/herself.	Prerequisites: N/A Test Data: N/A	Pass
TC-CB-003	Chat Box	TS: 12. Student types some text and clicks “send” button. ER: 12. Chat text is appeared, as well as the name and datetime.	Prerequisites: N/A Test Data: Any text	Pass
TC-LO-001	Logout	TS: 12. Student clicks the logout button. ER: 23. Student will be redirected to welcome page. 24. Student is unable to access any web page that requires login.	Prerequisites: N/A Test Data: N/A	Pass

APPENDIX I: Administrator UAT Test Cases

UAT Testing				
UAT Test Case	Tester Name: Yong Yoke Leng		Tester Type: Admin	Testing Date: 24/7/2021
Test Case ID	Test Case Type	Test Scenario (TS) and Expected Result (ER)	Prerequisites & Test Data	Status: Pass/Fail and Remarks
TC-LIA-001	Login	TS: 25. Admin clicks the admin login button. 26. Admin fills in the credentials and click login. ER: 13. The admin will be login and redirected into the admin home page.	Prerequisites: N/A Test Data: Admin own ID and own password.	Pass
TC-MTA-001	Add Topic	TS: 1. Admin clicks the “Modify Topic” button. 2. Admin clicks the add topic button. 3. Admin fills in a new topic name. 4. Admin clicks add button. ER: 1. A new topic is created.	Prerequisites: N/A Test Data: Any desired topic name.	Pass
TC-MTA-002	Edit Topic Name	TS: 1. Admin clicks the “Edit” button. 2. Admin change the topic name.	Prerequisites: N/A	Pass

		<p>3. Admin clicks edit button.</p> <p>ER:</p> <p>1. The topic name is changed.</p>	Test Data: Any desired topic name.	
TC-MTA-003	Delete Topic	<p>TS:</p> <p>1. Admin clicks the “Delete” button. 2. Admin clicks the “Yes” button.</p> <p>ER:</p> <p>1. The topic is deleted.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TLA-001	Modify Lesson	<p>TS:</p> <p>25. Admin clicks the “Learn Lesson” button. 26. Admin clicks any topic from the topic list.</p> <p>ER:</p> <p>13. Admin is able to view all lesson sections related to the selected topic.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TLA-002	View Lesson	<p>TS:</p> <p>25. Admin clicks the “Next” button once. 26. Admin clicks the “Back” button once.</p> <p>ER:</p> <p>25. When the “Next” button is clicked, it will be able to redirect to the next topic. 26. When the “Back” button is clicked, it will be able to redirect back to the previous topic. *Do not click them continuously.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-TLA-	Add Lesson	<p>TS:</p> <p>1. Admin clicks the “Add a new section” button.</p>	Prerequisites: N/A	Pass

003	Section	<ol style="list-style-type: none"> 2. Admin fills in the descriptions. 3. Admin clicks “Add” button. <p>ER:</p> <ol style="list-style-type: none"> 1. An indent is added when tab is pressed in code section text field. 2. A new section is added. 	Test Data: Any information, but the topic should be same as the current topic.	
TC-TLA-004	Edit Lesson Section	<p>TS:</p> <ol style="list-style-type: none"> 1. Admin clicks the “Edit” button on any section. 2. Admin changes the descriptions. 3. Admin clicks “Edit” button. <p>ER:</p> <ol style="list-style-type: none"> 1. An indent is added when tab is pressed in code section text field. 2. The selected section is edited. 	<p>Prerequisites: N/A</p> <p>Test Data: Any information, but the topic should be same as the current topic.</p>	Pass
TC-TLA-005	Delete Lesson Section	<p>TS:</p> <ol style="list-style-type: none"> 1. Admin clicks the “Delete” button on any section. 2. Admin clicks “Yes” button. <p>ER:</p> <ol style="list-style-type: none"> 1. The selected section is deleted. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CPA-001	Code Playground	<p>TS:</p> <ol style="list-style-type: none"> 37. Admin clicks the “Click to open the code editor for this section!” 38. Admin types codes based on the instructions. 39. Admin runs the code. <p>ER:</p> <ol style="list-style-type: none"> 37. Paiza.io online Swift compiler will be appeared. 	<p>Prerequisites: Admin is in any lesson topic.</p> <p>Test Data: N/A</p>	Pass

		<p>38. Code is able to be typed.</p> <p>39. Run button is worked and output is shown.</p>		
TC-EEA-001	Exercises	<p>TS:</p> <p>25. Admin clicks the “Exercise” button.</p> <p>26. Admin clicks any exercise from the exercise list.</p> <p>ER:</p> <p>13. Admin is able to view all questions related to the selected exercise, and admin-related buttons.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EEA-002	Exercises	<p>TS:</p> <p>25. Admin clicks check answer button in a question.</p> <p>26. Admin clicks check answer button again.</p> <p>ER:</p> <p>25. When check answer button is clicked for the first time, the submit button will be hidden, and blank for the question will be disabled and show the correct answer.</p> <p>26. When check answer button is clicked for the second time, the submit button will be appeared, and blank for the question will be enabled and clear the correct answer.</p>	<p>Prerequisites: Admin must complete TC-EEA-001 first.</p> <p>Test Data: N/A</p>	Pass
TC-EEA-003	Exercises	<p>TS:</p> <p>25. Admin fills in the blank with correct answer.</p> <p>26. Admin clicks submit button.</p> <p>ER:</p> <p>13. Submit button and check answer button will be hidden, and blank for the question will be</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass

		disabled and show the correct answer. A message “All of your answers are correct!” will be shown.		
TC-EEA-004	Exercises	<p>TS:</p> <p>25. Admin fills in the blank with wrong answer. 26. Admin clicks submit button.</p> <p>ER:</p> <p>13. A message “Wrong answer for some blank(s), please try again!” will be shown.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-EEA-005	Add Exercise Question	<p>TS:</p> <p>1. Admin clicks the “Add a new question” button. 2. Admin fills in the descriptions. 3. Admin clicks “Add” button.</p> <p>ER:</p> <p>1. An indent is added when tab is pressed in question text field. 2. A new question is added.</p>	<p>Prerequisites: N/A</p> <p>Test Data: Any information, but the topic should be same as the current topic.</p>	Pass
TC-EEA-006	Edit Exercise Question	<p>TS:</p> <p>1. Admin clicks the “Edit” button on any question. 2. Admin changes the descriptions. 3. Admin clicks “Edit” button.</p> <p>ER:</p> <p>1. An indent is added when tab is pressed in question text field. 2. The selected exercise is edited.</p>	<p>Prerequisites: N/A</p> <p>Test Data: Any information, but the topic should be same as the current topic.</p>	Pass
TC-EEA-	Delete Exercise	TS:	Prerequisites: N/A	Pass

007	Question	<ol style="list-style-type: none"> Admin clicks the “Delete” button on any question. Admin clicks “Yes” button. <p>ER:</p> <ol style="list-style-type: none"> The selected question is deleted. 	Test Data: N/A	
TC-GQA-001	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> Admin clicks the “Graded Quizzes” button. Admin clicks any quiz from the quiz list. <p>ER:</p> <ol style="list-style-type: none"> Admin is able to view the quiz edit page. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQA-002	Graded Quizzes	<p>TS:</p> <ol style="list-style-type: none"> Admin clicks on any question. <p>ER:</p> <ol style="list-style-type: none"> Question Text will be shown. Available options will be shown. Correct answer will be highlighted as green color. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-GQA-005	Add Quiz Question	<p>TS:</p> <ol style="list-style-type: none"> Admin clicks the “Add a new question” button. Admin fills in the descriptions. Admin clicks “Add” button. <p>ER:</p> <ol style="list-style-type: none"> An indent is added when tab is pressed in question text field. A new question is added. 	<p>Prerequisites: N/A</p> <p>Test Data: Any information, but the topic should be same as the current topic.</p>	Pass

TC-GQA-006	Edit Quiz Question	<p>TS:</p> <ol style="list-style-type: none"> Admin clicks the “Edit” button on any question. Admin changes the descriptions. Admin clicks “Edit” button. <p>ER:</p> <ol style="list-style-type: none"> An indent is added when tab is pressed in question text field. The selected question is edited. 	<p>Prerequisites: N/A</p> <p>Test Data: Any information, but the topic should be same as the current topic.</p>	Pass
TC-GQA-007	Delete Quiz Question	<p>TS:</p> <ol style="list-style-type: none"> Admin clicks the “Delete” button on any question. Admin clicks “Yes” button. <p>ER:</p> <ol style="list-style-type: none"> The selected question is deleted. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-RSA-001	Register Student	<p>TS:</p> <ol style="list-style-type: none"> Admin clicks “Register Student” button. Admin fills in student ID, student name and student UTAR email. Admin clicks “register” button. <p>ER:</p> <ol style="list-style-type: none"> Student is successfully registered with alert message. 	<p>Prerequisites: N/A</p> <p>Test Data: Any student hasn’t been registered, please refer to the student list.</p>	Pass
TC-RSA-002	Register Student	<p>TS:</p> <ol style="list-style-type: none"> Admin fills in the same student credential. Admin clicks “register” button. 	<p>Prerequisites: Admin must finish TC-RSA-001 first.</p>	Pass

		<p>ER:</p> <ol style="list-style-type: none"> 1. A message “The student ID/name/UTAR email is already exist!” is shown. 	<p>Test Data: Same test data as in TC-RSA-001.</p>	
TC-RSA-003	Register Student	<p>TS:</p> <ol style="list-style-type: none"> 1. Admin use email that is not following the email format (xxx@yyy.zz) 2. Admin clicks “register” button. <p>ER:</p> <ol style="list-style-type: none"> 1. A message “This email format is invalid!” is shown. 	<p>Prerequisites: N/A</p> <p>Test Data: Any student hasn’t been registered, but the email test data use “aaa”</p>	Pass
TC-RSA-004	Register Student	<p>TS:</p> <ol style="list-style-type: none"> 1. Admin use email that is not following the UTAR student email format (xxx@lutar.my) 2. Admin clicks “register” button. <p>ER:</p> <ol style="list-style-type: none"> 1. A message “This email is not a UTAR student email” is shown. 	<p>Prerequisites: N/A</p> <p>Test Data: Any student hasn’t been registered, but the email test data use “aaa@gmail.com”</p>	Pass
TC-PFA-001	Admin Profile	<p>TS:</p> <ol style="list-style-type: none"> 13. Admin clicks “Admin Profile” on the navigation bar. <p>ER:</p> <ol style="list-style-type: none"> 37. Admin can see his/her name, student ID and student email. 38. Admin can see view student list button. 39. Admin can see a search bar. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass

TC-PFA-002	Student List	<p>TS:</p> <ol style="list-style-type: none"> Admin clicks “View student list” button in the admin profile page. <p>ER:</p> <ol style="list-style-type: none"> Admin can view all students with the credentials: Student ID, Student Name, Student Email, Temporary Password (If no then it will show N/A). 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-PFA-003	Search Student	<p>TS:</p> <ol style="list-style-type: none"> Admin key in existing student ID and click search button in admin profile page. <p>ER:</p> <ol style="list-style-type: none"> Student Name, ID, average performance and quiz history will be appeared. 	<p>Prerequisites: Student must be registered</p> <p>Test Data: Registered student ID, refer to the registered student list.</p>	Pass
TC-PFA-004	Delete Student	<p>TS:</p> <ol style="list-style-type: none"> Admin click “Delete” button. Admin click “Yes button”. <p>ER:</p> <ol style="list-style-type: none"> The student is deleted. 	<p>Prerequisites: Student must be registered, and the admin must complete TC-PFA-003 first.</p> <p>Test Data: Registered student ID, refer to the registered student list.</p>	Pass
TC-PFA-	Search Student	<p>TS:</p> <ol style="list-style-type: none"> Admin key in non-existing student ID and click 	<p>Prerequisites: Student must not be</p>	Pass

005		<p>search button in admin profile page.</p> <p>ER:</p> <ol style="list-style-type: none"> 1. An alert “The student ID you key in is not exist.” is appeared. 	<p>registered</p> <p>Test Data: 1000000</p>	
TC-CBA-001	Chat Box	<p>TS:</p> <ol style="list-style-type: none"> 13. Admin clicks “Chat Box” button. <p>ER:</p> <ol style="list-style-type: none"> 13. Admin can view the chat box interface, with his/her admin ID and name, with bracket (Admin). 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CBA-002	Chat Box	<p>TS:</p> <ol style="list-style-type: none"> 13. Admin views the online user list. <p>ER:</p> <ol style="list-style-type: none"> 13. Admin is able to see list of online users, including himself/herself. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
TC-CBA-003	Chat Box	<p>TS:</p> <ol style="list-style-type: none"> 13. Admin types some text and clicks “send” button. <p>ER:</p> <ol style="list-style-type: none"> 13. Chat text is appeared, as well as the name and datetime. 	<p>Prerequisites: N/A</p> <p>Test Data: Any text</p>	Pass
TC-CBA-004	Chat Box	<p>TS:</p> <ol style="list-style-type: none"> 1. Admin clicks “delete” button on any chat. <p>ER:</p> <ol style="list-style-type: none"> 1. The selected chat is deleted. 	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass

TC-LOA-001	Logout	<p>TS: 13. Admin clicks the logout button.</p> <p>ER: 25. Admin will be redirected to welcome page. 26. Admin is unable to access any web page that requires login.</p>	<p>Prerequisites: N/A</p> <p>Test Data: N/A</p>	Pass
------------	--------	---	---	------

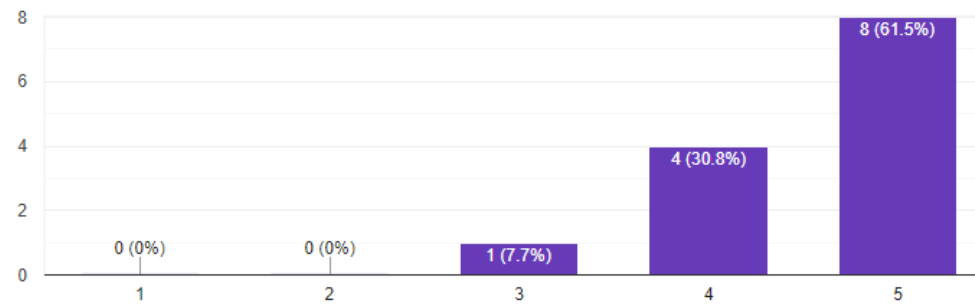
APPENDIX J: User Satisfaction Survey Response



I thought the website/system was easy to use.

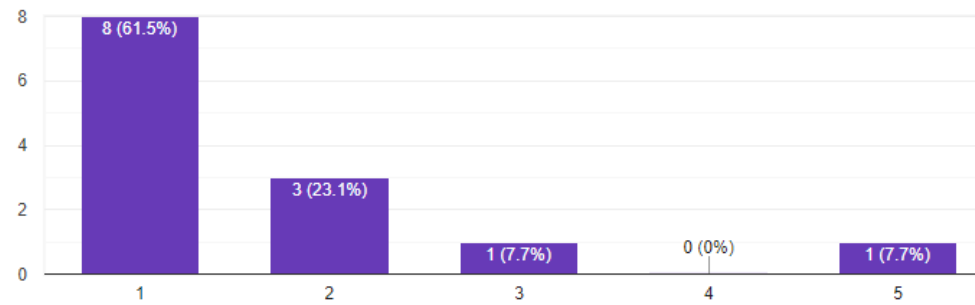


13 responses



I think that I would need the support of a technical person to be able to use this website/system.

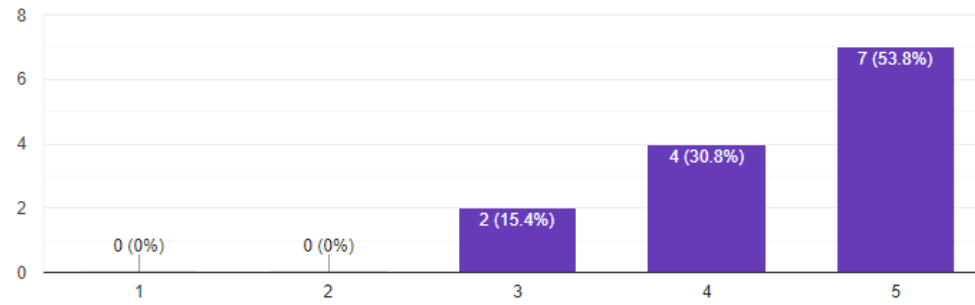
13 responses



I found this website/system was easily moved through without a lot of backtracking or data re-entry.

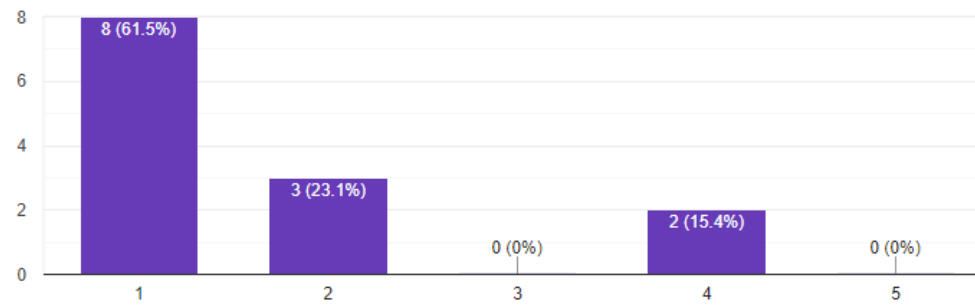


13 responses



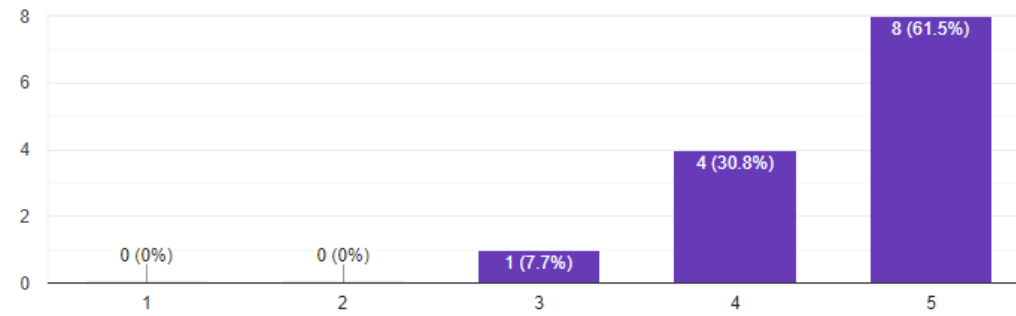
I thought there was too much inconsistency in this website/system.

13 responses



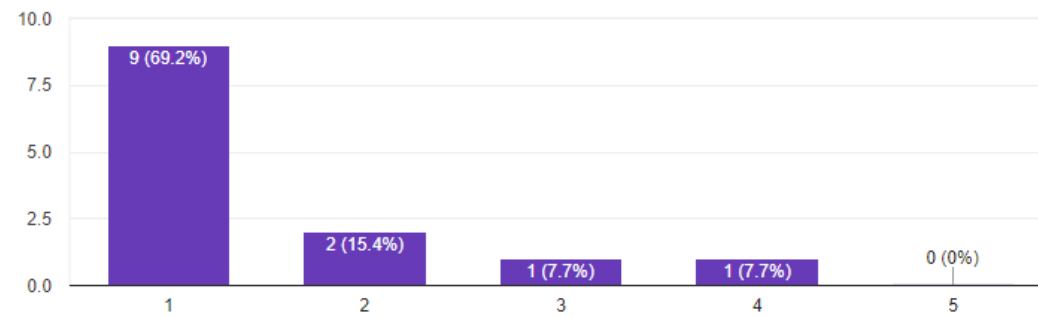
I would imagine that most people would learn to use this website/system very quickly.

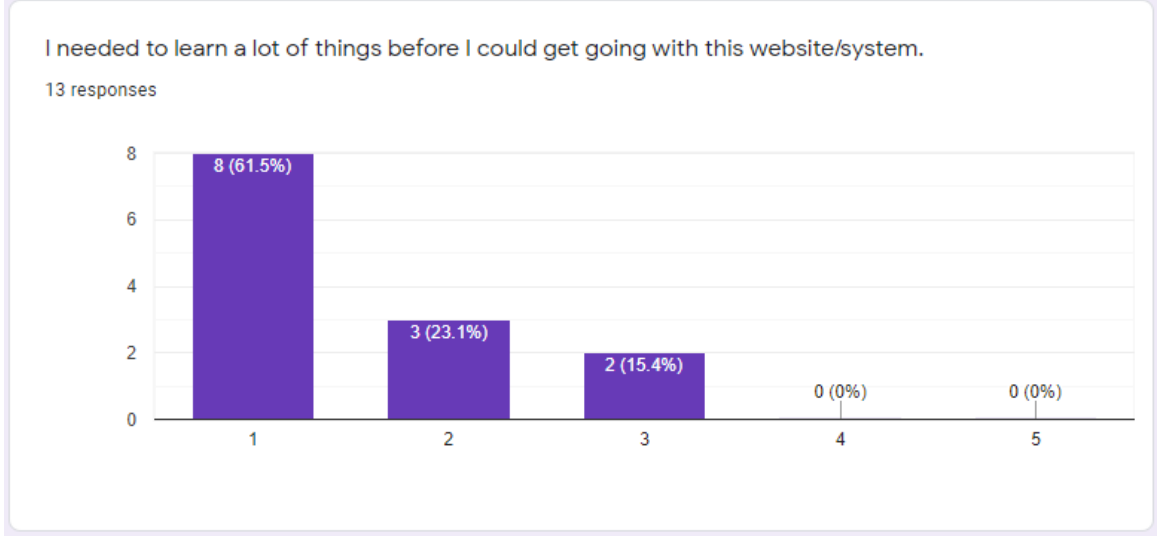
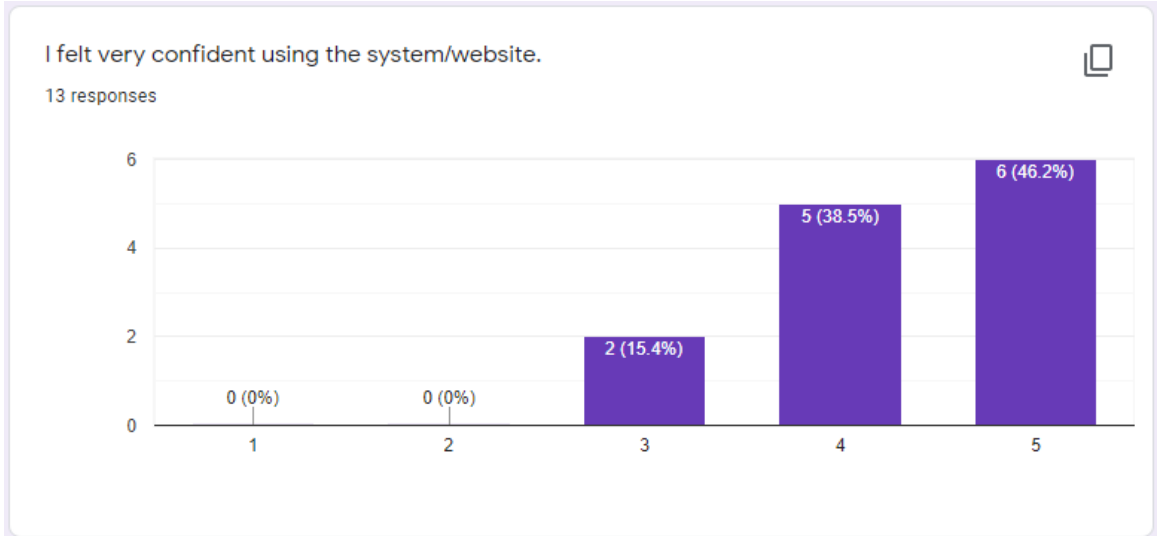
13 responses



I found the system/website very awkward to use.

13 responses





Section 2

What did you like best about the site?

13 responses

I like the interface of the web side and is easy to be use.

Simple interface and easy to use, just like most social platform but this one is for education.

Easy navigation options, consistent styling, sufficient information provided and most importantly the chatbox function is quite interesting.

the built-in compiler

Simple UI design and user friendly

Clean and easy to use interface with all the required featured implemented

being able to run code

Simplicity

The chatbox system

It is very informative and it can help me to learn a lot about Swift

the user interface is attractive

The topics (Lesson)

Chat feature

What did you like least about the site?

13 responses

the login interface can be more attractive

Design is not the best, but still above average

-

the colour scheme about the website

N/A

Loading time for the Exercise

chat box is hidden inside 'Others'

UI

The button for answering the quiz

Nothing honestly

na

The Playground

Not being able to view total number of questions available in Quiz

If you were to describe this site to a colleague in a sentence or two, what would you say?

13 responses

Best Swift learning site ever

A learning platform that is suitable for learning basic Swift language.

"If you wish to learn Swift programming language and track your performance, this website would be a good choice."

good place for doing revision and learn swift.

This is a good platform to learn swift, easy to use and learn swift language in a simple way.

Good e-learning platforms to get started with for online learning

It is a website to start learning Swift language and you do not need coding application such as XCode to start.

Able to learn Swift programming without the need to purchase a Mac

Its a website where you can learn swift programming easily

It is really an interesting site for beginners of swift

this website is easy to use and learn

This is a good website where you can learn the basics of Swift language where there will be lesson to learn and quiz to test your knowledge

The chat feature would be an added bonus to collaborative learning

Do you have any other final comments or questions?

13 responses

Very good website

No questions.

No. Overall it is an interactive website that has a good color matching and styling and providing a lot of lessons, exercises and quizzes that related to the Swift programming.

overall is good. Try to improve the design to make it responsive to browser size.

Good work.

None

Maybe put the chatbox on the top bar with an icon to make it more recognizable.

Good job with your FYP !

More lessons to be added

No

great user experience

This website is good place to learn about basics of Swift Programming Language. However it still needs some touch up in opening the Playground

Wouldn't entirely say this is a fault in design, but the way how "performance" (under profile page) is represented just confused me a little.

