UTAR COMMUNITY APPLICATION FOR STUDENTS

BY

CHANG PING YEN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2023

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: UTAR Community Application For Students

Academic Session: June 2023

I CHANG PING YEN

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

	Verified by,
yen	[illegible]
(Author's signature)	(Supervisor's signature)
Address:	
NO 41, Indian Settlement	
31700 Malim Nawar	Tan Joi San
Perak	Supervisor's name
Date: 12/9/2023	**Date**: 14 Sep 2023

Universiti Tunku Abdul Rahman			
Form Title : **Sample of Submission Sheet for FYP/Dissertation/Thesis**			
Form Number: **FM-IAD-004**	Rev No.: **0**	Effective Date: **21 JUNE 2011**	Page No.: **1 of 1**

FACULTY/INSTITUTE* OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 12/09/2023

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that **Chang Ping Yen** (ID No: **19ACB02846**) has completed this final year project/ dissertation/ thesis* entitled **"UTAR Community Application for Students"** under the supervision of **Dr Tan Joi San** (Supervisor) from the Department of Computer Science, Faculty/Institute* of Information and Communication Technology.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

yen

____
(*Chang Ping Yen*)

DECLARATION OF ORIGINALITY

I declare that this report entitled "**UTAR Community Application for Students**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	yen
Name	:	CHANG PING YEN
Date	:	12/9/2023

ACKNOWLEDGEMENTS

I want to express my deepest gratitude to my supervisor Dr Tan Joi San as I would not have been able to start my project without her supervision and guidance. Throughout my journey of completing my final year project, she has provided me with valuable and constructive suggestions to improve my project. Besides, I would like to thank my final year project moderator Dr Sarah A'fifah Binti Abdullah Sani, for being involved in evaluating my work. By receiving feedback from my moderator, I can review the strengths and weaknesses of my project and make amendments accordingly.

I would also like to thank my family members, especially my parents for their support. They have sacrificed their money and effort to provide me good education so that I can strive in society, and I am thankful for that. Without their contribution and words of encouragement, I would have never been able to make it this far.

Finally, I would like to thank all my friends and those who have supported me throughout my time studying at UTAR. The mental support I got meant a lot to me, and I am grateful for all the nice things and people I have met.

ABSTRACT

Universities nowadays are taking advantage of modern technology by releasing their own community apps, which aim to improve students' connections and enhance their university experience. First-year students often have a much more difficult time adjusting to a new environment. New to the environment, first-year students are bound to have questions they would like to enquire others for help, but most of the time, they lack connections and do not know whom to seek out. In addition, students will also have to deal with the dilemma of building new friendships as they are either shy or afraid of social rejection. Furthermore, students also lack a proper platform which allows them to share opinions anonymously. Most of the time, students wish to share their thoughts on specific issues with everyone but are afraid of judgment and the risk of cyberbullying. **UTAR Community Application for Students**, an online university community mobile application, is developed to solve the problems mentioned. The app's target users are UTAR (University Tunku Abdul Rahman) students. It aims to provide a platform for students to make friends easily within their surroundings. Besides, the app allows students to engage in anonymous forums, encouraging freedom of speech. Lastly, the app allows first-year students to register as mentees and be assigned to a senior who will support and facilitate first-year students' journey throughout the peer mentoring program. The project uses RAD-based methodology, which includes 4 phases while a laptop and mobile phone will be used for developing and testing the app.

TABLE OF CONTENTS

FIRST COVER **i**

REPORT STATUS DECLARATION FORM **ii**

SUBMISSION OF FINAL YEAR PROJECT **iii**

DECLARATION OF ORIGINALITY **iv**

ACKNOWLEDGEMENTS **v**

ABSTRACT **vi**

TABLE OF CONTENTS **vii**

LIST OF FIGURES **x**

LIST OF TABLES **xv**

LIST OF ABBREVIATIONS **xvi**

Chapter 1 Introduction **1**

1.1 Problem Statement and Motivation 3

1.2 Objectives 5

1.3 Project Scope and Direction 6

1.4 Contributions 8

1.5 Report Organization 8

Chapter 2 Literature Review **9**

2.1 Previous Works on Community App 9

2.1.1 hi-hive Community [2] 9

2.1.2 Raklet [3] 14

2.1.3 Reddit [4] 18

2.1.4 InterNations [6] 22

2.1.5 ZeeMee [8] 26

2.2 Summary 30

Chapter 3 System Design **32**

3.1 Use Case Diagram 32

3.2 Use Case Description 33

3.3 Mobile App Development 55

3.3.1 Android Emulator Setup 55

3.3.2 Connect Flutter Project with Firebase 56
3.3.3 Signup and Login Function Development (Admin / Student) 58
3.3.4 Upload User Profile Picture Function Development (Student) 60
3.3.5 Add Forum Post Function Development (Student) 61
3.3.6 Like Forum Post Function Development (Student) 62
3.3.7 Delete Forum Post Function Development (Admin / Student) 63
3.3.8 View Forum Post Function Development (Admin / Student) 64
3.3.9 Share Location Function Development (Student) 66
3.3.10 View Meetup History Function Development (Admin / Student) 70
3.3.11 View / Add / Edit User (Admin) 72
3.3.12 Peer Mentor Module Function Development (Student) 76
3.3.13 Add / Delete / Edit Options Function Development (Admin) 80
3.3.14 Profanity detection Function Development 82

Chapter 4 Methodology and Tools 83
4.1 Methodologies 83
4.2 System Requirement 85
4.2.1 Hardware 85
4.2.2 Software 86
4.3 Timeline 88
4.4 Summary 90

Chapter 5 Implementation and Testing 91
5.1 Sign Up and Login 91
5.2 Active Current Location 93
5.3 Share Location Request 94
5.4 Add and Delete Forum Post 98
5.5 Comment and Like Forum Post 100
5.6 Profanity Detector 101
5.7 Register Mentor / Mentee 102
5.8 Match Mentor / Mentee 103
5.9 Schedule Meeting 105
5.10 Terminate Mentor Mentee Relationship 107
5.11 Upload Profile Photo 108
5.12 Admin Home Page 109
5.13 Admin Users Page 110

5.14 Summary 116

Chapter 6 Conclusion 117

6.1 Project Review, Discussion and Conclusion 117

6.2 Future Work 119

REFERENCES 120

FINAL YEAR PROJECT WEEKLY REPORT 121

POSTER 126

PLAGIARISM CHECK RESULT 127

CHECKLIST FOR FYP2 THESIS SUBMISSION 131

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1.1.1	Users creating their community	9
Figure 2.1.1.2	Create New Chat Function	10
Figure 2.1.1.3	In app messaging interface	10
Figure 2.1.1.4	Marketplace Product Listings	11
Figure 2.1.1.5	Seller and Product Information	11
Figure 2.1.1.6	Product Listing Creation (Firearms)	12
Figure 2.1.2.1	Welcome announcement	14
Figure 2.1.2.2	Application form for members	14
Figure 2.1.2.3	Membership Plan panel pt1	15
Figure 2.1.2.4	Membership Plan panel pt2	15
Figure 2.1.2.5	Job Board Example	16
Figure 2.1.3.1	Reddit homepage	18
Figure 2.1.3.2	Sample Reddit post	18
Figure 2.1.3.3	Reddit bottom navigation	19
Figure 2.1.3.4	Anonymous browsing option	20
Figure 2.1.4.1	Search function for location-based communities	22
Figure 2.1.4.2	Explore events page	23
Figure 2.1.4.3	Event description page	23
Figure 2.1.4.4	InterNations membership plan	24
Figure 2.1.4.5	Review of InterNations app	25
Figure 2.1.5.1	Homepage of ZeeMee App	26
Figure 2.1.5.2	Audio Room	27
Figure 2.1.5.3	Update Application Status	28
Figure 2.1.5.4	Verify Admittance	28
Figure 2.1.5.5	Post creation (firearms)	29
Figure 3.1	Use case diagram for UTAR Community Application for Students	32
Figure 3.3.1.1	Install Android SDK Command-line Tools	55
Figure 3.3.1.2	Create virtual device	55

Figure 3.3.2.1	Register app	56
Figure 3.3.2.2	Instructions to download json file	56
Figure 3.3.2.3	Contents of json file	57
Figure 3.3.2.4	Flutter project successfully connected to Firebase	57
Figure 3.3.3.1	auth_controller.dart: Sign Up function	58
Figure 3.3.3.2	auth_controller.dart: Login function	59
Figure 3.3.4.1	profile_controller.dart : Upload Profile Picture function	60
Figure 3.3.5.1	forum_controller.dart : Add Forum Post function	61
Figure 3.3.6.1	forum_controller.dart : Like Forum Post function	62
Figure 3.3.7.1	forum_controller.dart : Delete Forum Post function (User)	63
Figure 3.3.7.2	home_admin_controller.dart : Delete Forum Post function (Admin)	63
Figure 3.3.8.1	forum_view.dart : View Forum Post (User) pt1	64
Figure 3.3.8.2	forum_view.dart : View Forum Post (User) pt2	64
Figure 3.3.8.3	forum_view.dart : View Forum Post (User) pt3	65
Figure 3.3.8.4	home_admin_view.dart : View Forum Post (Admin)	65
Figure 3.3.9.1	home_view.dart : Home Page View (Google Maps)	66
Figure 3.3.9.2	home_controller.dart : Get Location Function	66
Figure 3.3.9.3	location_services.dart : Geolocator package	67
Figure 3.3.9.4	home_controller.dart pt1 : Send Request to Meet	68
Figure 3.3.9.5	home_controller.dart pt2 : Send Request to Meet	68
Figure 3.3.9.6	home_view.dart	69
Figure 3.3.9.7	scanQR_view.dart : Scan QR Code Function	69
Figure 3.3.10.1	history_view.dart : Approved Meetup History (User)	70
Figure 3.3.10.2	home_controller.dart	70
Figure 3.3.10.3	meetup_admin_view.dart : Approved Meetup History Panel (Admin)	71
Figure 3.3.11.1	users_admin_view.dart : View User List	72
Figure 3.3.11.2	users_admin_controller.dart	73
Figure 3.3.11.3	users_add_admin_view.dart : Create User	73
Figure 3.3.11.4	users_admin_controller.dart : Create User	74

Figure 3.3.11.5	users_detail_admin_view.dart : Edit User	75
Figure 3.3.11.6	users_admin_controller.dart : Edit User	75
Figure 3.3.12.1	dashboard_mentee_view.dart (Pending Request View)	76
Figure 3.3.12.2	dashboard_mentee_view.dart (Reject Request View)	76
Figure 3.3.12.3	match_view.dart : Match Mentor (Mentee View)	77
Figure 3.3.12.4	meeting_mentee_view.dart : Meetings Panel	78
Figure 3.3.12.5	peer_mentor_controller.dart : Create Meeting	78
Figure 3.3.12.6	match_mentor_view.dart (Mentor View)	79
Figure 3.3.13.1	setting_admin_controller.dart: addYearsOfStudy()	80
Figure 3.3.13.2	setting_admin_controller.dart: editYearsOfStudy()	80
Figure 3.3.13.2	setting_admin_controller.dart: deleteYearsOfStudy()	81
Figure 3.3.14.1	forum_controller.dart: checkProfanity()	82
Figure 4.1	RAD Methodology	83
Figure 4.3	Gantt Chart for project timeline	89
Figure 5.1.1	Login UI	92
Figure 5.1.2	Sign Up UI	92
Figure 5.2.1	Set Radius function	93
Figure 5.2.2	Display Location With Marker	93
Figure 5.3.1	Other User's Location Marker	94
Figure 5.3.2	Request Permission for Location Sharing	94
Figure 5.3.3	Request to Meet	95
Figure 5.3.4	Reject / Approve Request to Meet	95
Figure 5.3.5	Approved Request	96
Figure 5.3.6	Navigate to User Location / Scan QR Code / Yes,No Option	96
Figure 5.3.7	QR Code for Approved Request	97
Figure 5.3.8	Chang attempt to scan QR Code	97
Figure 5.3.9	Meetup Status Success	97
Figure 5.4.1	Add Forum Post	98
Figure 5.4.2	Confession Page View (Different user posts)	98
Figure 5.4.3	My Post Panel	99
Figure 5.4.4	Delete Post	99

Figure 5.5.1	Add Comment	100
Figure 5.5.2	Comment Posted Under Post	100
Figure 5.6.1	Profanity Detection	101
Figure 5.6.2	User under temporary ban	101
Figure 5.7.1	Fill in details for registration	102
Figure 5.7.2	Error register as mentor	102
Figure 5.8.1	Match View for Mentee	103
Figure 5.8.2	Match View for mentor	103
Figure 5.8.3	Dashboard View for Mentee (Before assigned mentor)	104
Figure 5.8.4	Dashboard View for Mentee (After assigned mentor)	104
Figure 5.9.1	'Meet' panel	105
Figure 5.9.2	Fill in details to create new meeting	105
Figure 5.9.3	Pending meeting request	106
Figure 5.9.4	Upcoming meeting request	106
Figure 5.9.5	Meeting not done	106
Figure 5.9.6	Completed meeting	106
Figure 5.10.1	Terminate Relationship	107
Figure 5.10.2	Reason of Termination	107
Figure 5.10.3	Termination Notification	107
Figure 5.11.1	Upload Profile Photo UI	108
Figure 5.11.2	Image Chosen	108
Figure 5.12.1	Admin Home Page (View Forum Post)	109
Figure 5.12.2	View Post Details	109
Figure 5.13.1	List of users	110
Figure 5.13.2	Add new user	110
Figure 5.13.3	List of users (New user added)	111
Figure 5.13.4	Update user details	111
Figure 5.13.5	Meetup History	112
Figure 5.13.6	Meeting History	112
Figure 5.13.7	Add Faculty	112
Figure 5.13.8	Faculty successfully added	112
Figure 5.13.9	Delete Faculty	113
Figure 5.13.10	Faculty successfully deleted	113

Figure 5.13.11	Edit Years of Study	114
Figure 5.13.12	Edit Successful	114
Figure 5.13.13	Mentor Details	115
Figure 5.13.14	Mentee Details	115

LIST OF TABLES

Table Number	Title	Page
Table 2.2	Comparison of existing community apps	30
Table 3.2.1	Sign Up Use Case Description	33
Table 3.2.2	Login Use Case Description	34
Table 3.2.3	View Forum Post Use Case Description	35
Table 3.2.4	Add Forum Post Use Case Description	36
Table 3.2.5	Delete Forum Post Use Case Description	37
Table 3.2.6	Comment Forum Post Use Case Description	38
Table 3.2.7	Like Forum Post Use Case Description	39
Table 3.2.8	View Meetup History Use Case Description	40
Table 3.2.9	Share Location Use Case Description	41
Table 3.2.10	View Meeting History Use Case Description	43
Table 3.2.11	Register Peer Mentor Use Case Description	44
Table 3.2.12	Match Mentor Mentee Use Case Description	45
Table 3.2.13	Add Meeting Use Case Description	46
Table 3.2.14	Delete Meeting History Use Case Description	47
Table 3.2.15	View User Profile Use Case Description	48
Table 3.2.16	Add User Use Case Description	49
Table 3.2.17	Edit User Use Case Description	50
Table 3.2.18	View Mentor Mentee Use Case Description	51
Table 3.2.19	Add Settings Use Case Description	52
Table 3.2.20	Delete Settings Use Case Description	53
Table 3.2.21	Edit Settings Use Case Description	54
Table 4.2.1.1	Specifications of laptop	85
Table 4.2.1.2	Specifications of mobile device	85
Table 4.2.2.1	Software Components for mobile app development	86

LIST OF ABBREVIATIONS

API	Application Programming Interface
COVID-19	Coronavirus Disease
CPU	Central Processing Unit
GPS	Global Positioning System
GPU	Graphics Processing Unit
IDE	Integrated Development Environment
Mods	Moderators
QR Code	Quick Response Code
RAD	Rapid Application Development
RAM	Random Access Memory
SDK	Software Development Kit
UTAR	University Tunku Abdul Rahman

Chapter 1 Introduction

The rapid evolvement of technology throughout the years has driven globalization and digital transformation. Technology has played an essential role in our society as it has been responsible for altering the way people think and act. From communicating only through text messages to phone calls and now video conferencing, technology has become an essential tool to drive human engagement, and it also profoundly changed how humans interact with one another. With online resources being well established and accessible by more people, it has led to the rise of virtual communities. Virtual communities are people with a common interest who use the digital internet to communicate [1]. It is very powerful as virtual communities can bring together like-minded strangers of specific interest from any part of the world without any restrictions. Before virtual communities dominated cyberspace, for people to connect with others of similar interests, they would have to attend various interest clubs and physically engage with the members by participating in activities organized by the club. However, as virtual communities became increasingly popular, people have begun to use them as an easier way to connect with their peers. Virtual communities come in different forms, and it is crucial to identify the type of community to cater to the target community's needs. Networking communities mainly focus on helping like-minded people to create links and help each other achieve their goals through information sharing. Hence it will be the focus of this project.

In this modern era, virtual communities began blooming as a way of communication, leading to the creation of community-based applications. Community-based apps are mobile applications accessible on phones or tablets, allowing a specific community to gather and engage virtually. These like-minded individuals use such communication tools to build their community and even host online discussions with fellow members. They treat virtual communities as open and safe spaces to express their thoughts without fearing being judged by others. An important thing to note is that even though there is no geographical delimitation where virtual communities can take place, to maintain peacefulness within the community, owners of the community will have to set community guidelines and take immediate action against those who violate them. After noticing the impact of community-based apps, many industries, including entertainment, education, retail, and many more, have developed their own community app.

CHAPTER 1 INTRODUCTION

The growth of community apps has vastly affected the education industry. Many educational institutions have increased their investment in developing their community app in hopes of improving the engagement with their students. University community apps primarily exist to better aid students throughout their journey across higher education. A good functioning university community app should provide convenience to users and be easy to use. The app should have a simple yet appealing design, making it easy to navigate the app effortlessly so users would not lose interest quickly. Many university community apps in the market provide services such as facilitating communication between instructors, students, and parents. Instructors can use such apps to upload learning materials and update students' grades, allowing students to access them anytime and anywhere easily. Besides, some apps also offer the function for instructors to communicate with parents so that parents may constantly gain updates about their child's learning progress. Other university community apps include functions such as an online student inquiries helpdesk, online forums, conference/event management and many more, depending on the needs of the university.

For this project, the proposed mobile application: **UTAR Community Application for Students**, is targeted at the education field. However, unlike other apps in the market, which focus on engaging students with instructors or instructors with parents, this app is solely for student engagement. It means only UTAR students will be the primary users of the app. Students can use the app to perform various functions, such as building their network and making new friends with similar interests, creating or joining forums to share knowledge and seeking support or guidance by applying for peer mentoring programs.

In summary, community-based apps are beneficial for students in driving communication and self-development. Therefore, it is suggested that every educational institution, especially higher education, adopt community-based applications as it helps open up more opportunities for students to make friends and allows students to gain new insights during online discussions.

1.1 Problem Statement and Motivation

UTAR Community Application for Students is a community-based mobile application for the use of UTAR students which aims to solve the following issues:

i) Difficult for shy students to make friends

As students further their studies in university, it can be difficult for them to make friends in a new environment as students lack social connections, subsequently making them antisocial. The social anxiety that students experience makes them afraid of engaging with others, fearing the risk of being rejected or judged by others. If students have had bad experiences when making friends, then it is likely that they will not be willing to open up to others anymore to protect themselves against betrayal. Furthermore, some students lack social skills and do not know how to start a conversation with others or engage in small talk. Even if they tried to, they might end up saying the wrong things, making others feel awkward or uncomfortable. Plus, students often have busy schedules, so there are not many natural opportunities to physically meet new people. Students must put in extra time and effort, such as joining various events to get exposed to more opportunities to meet new people. It requires a lot of commitment, and many students are not fully ready to commit.

ii) Lack of platform for students to express their thoughts anonymously

Nowadays, almost everyone is using the internet to share information and connect with others. Most people's lives seem fine and doing well on the internet, but that may not necessarily be the case. People tend to share the positive things they wish to portray to others, but they are hesitant when posting deeper issues due to the fear of being judged by others. Still, there is a need for a platform to allow people to express their thoughts, whether positive or negative. There is no proper platform for UTAR students to share their thoughts and perform discussions anonymously with everyone online. UTAR only provides students with individual Microsoft Teams accounts linked to students' institution email. This collaboration app is only built for work purposes, which helps connect students and instructors without physically meeting each other. Through Microsoft Teams, students can only use it to send chat messages, share documents, attend meetings, or create group channels freely, but their identities will be known to others. Although many platforms exist, such as confession pages, Reddit, 4chan, forums, etc., that allow users to comment anonymously, it is still not specialized enough for a specific community.

iii) Difficult for first-year students to seek the right person's help for guidance and support

Most newly registered students to UTAR came from various backgrounds and often required time to adapt to a new environment. Such students often have questions regarding their faculty, program or even the university's academic culture, but they do not know which authority to approach. The only way to clear their doubts is to seek answers from Head of Department, Faculty General Office, or their academic advisors. Such authorities usually provide answers from the academician's perspective and may not always be suitable or helpful for the students. Although UTAR organizes Peer Mentoring Programmes for new intake students, students often are hesitant to join such events as such events usually associate with a large group of people and come with fixed routines, which may not be suitable for every student. Even if students join such events, they have no guarantee of meeting mentors/seniors of the same faculty as them. In the end, students either seek the wrong person for advice or take too much time to seek the right person for help.

1.2 Objectives

The project aims to develop a user-friendly mobile application for UTAR students. Students can use the app to locate their peers, anonymous content sharing and join mentoring programs. The objectives of this system development are:

1) To allow students to locate peers nearby

Students can allow the app to access their current location. Then, based on their current location, they can view other users nearby and choose a user to request location sharing. The other user can choose to approve or reject the request. For approved requests, the user will receive a notification indicating the request has been approved, and they will be directed to Google Maps for navigation.

2) To provide students with an anonymous forum which promotes freedom of expression

Students can express their opinions anonymously on text-based discussion forums. Although the platform promotes freedom of expression, there are still certain limitations, and students must comply with the community rules before posting. The moderator can view and delete every user's post if it regulates community rules, such as posting offensive materials or attacking or degrading other users. Students can create a new post for discussion, like a particular post, and join/interact with other existing discussions by leaving comments.

3) To allow students to participate in peer mentoring programs

Students can register as mentors or mentees at the beginning of the trimester. Mentees must set up their profiles by choosing the type of support they need and answering the survey. Mentors will also have to select the support they can provide mentees. After the mentor and mentee successfully set up their profiles, the mentee may search for the right mentor based on their preference and establish a mentor-mentee relationship. Once the connection is established, the mentor and mentee can complete a series of activities together and evaluate each other at the end of the peer mentoring program.

1.3 Project Scope and Direction

The **"UTAR Community Application for Students"** mobile application allows UTAR students to make friends easily by connecting them with their peers. Furthermore, the app also serves as a platform for students to self-express and share ideas, given that they abide by the community rules and regulations. Finally, the app also assists first-year students throughout the beginning of their university journey by providing them with the support they need, as the transition to higher education can be a daunting experience for some students. Therefore, the app will include functions such as locating peers nearby, an anonymous confession forum and a peer mentoring matching mechanism.

Students must create a user account using a valid UTAR email and a strong password. If students use other email domains to register the account, the app will prompt an error message stating that the registration is unsuccessful. This ensures that only UTAR students can use the app and not anyone else. After students successfully set up their accounts, they can use the functions provided in the app freely. The first function is the **"Friends Nearby"** feature, where students can make new friends by locating peers nearby and allowing the app to access their current location. Then, based on their current location, students can choose users they are interested in becoming friends with, then the other party will receive a notification, and they may decide whether to accept or reject the location-sharing request.

The following function is **"UTAR Forum"** feature, which allows students to create anonymous posts about discussions on specific topics, knowledge sharing, and seek feedback/opinions or confessions. Students will be issued an anonymous username that keeps their identity hidden upon joining the forum. Students get to view, like and comment on any post anonymously. Although UTAR Forum promotes freedom of speech and content sharing, students still must obey the Community Standards, which include prohibited posting of any content related to nudity/sexually suggestive, hate speech, violence, spam, etc.

The last function of the app is the **"Peer Mentoring"** feature. This feature allows first-year students having trouble adapting to the new university environment to seek help from their seniors. Interested first-year students can sign up as a mentee, and during the signup process, they will be given a few support choices they wish to seek from a mentor. These choices include Academics, Personal or Co-curricular. Then, students must complete a survey about the characteristics they seek in a mentor. Seniors interested in helping their juniors may register themselves as mentors. Mentors will be categorized into the 3 groups mentioned previously, so students can look for the right person to seek support. The system will assign 1 mentor to 1 mentee. Mentors and mentees can either be from the same or different courses. The Peer Mentoring session will last for a minimum of 1 month to a maximum of 1 year and must be updated occasionally to determine whether the mentor-mentee relationship is ongoing. At the end of each session, the mentor and mentee can rate and evaluate one another to reflect on the effectiveness of the whole peer mentoring journey.

In summary, a project is carefully planned to develop a **"UTAR Community Application for Students"** mobile application, which contains functions for students to locate and make new friends, share content anonymously and participate in peer mentoring programs. All these functions serve to enhance students' experience in higher education.

1.4 Contributions

The main contribution of the proposed application is to solve the issues stated in the problem statement. The proposed system acts as a platform for students to locate friends near their compound, making it much easier to connect with their peers. Plus, students can express their thoughts by joining anonymous discussions and gaining more knowledge through content sharing. Besides sharing content, students can also share deep conversations, get advice, and share secrets with strangers without worrying about their identities being exposed. Lastly, students can use the app to register for peer mentoring programs as mentors or mentees to create a more friendly and caring university environment.

1.5 Report Organization

Chapter 1 introduces and explains community and university community apps in general. It also discusses the issues students face when they have just enrolled in university, such as difficulties in making new friends, needing a proper platform to express their opinions, and knowing whom to approach for help. Therefore, the idea of developing a UTAR community app named **"UTAR Community Application for Students"** is proposed to solve such issues. This app aims to enhance students' university life and build stronger connections among students. Students can locate peers nearby, interact with discussions anonymously, and participate in peer mentoring programs. More details regarding the project will be shown in upcoming chapters. In Chapter 2, related systems are reviewed based on their strengths and weaknesses. Then, Chapter 3 showcases the code for each function development. Next, Chapter 4 discusses the system requirements, methodologies, and overall timeline. Moving on, Chapter 5 showcases screenshots of the app. Lastly, Chapter 6 concludes the findings of the project.

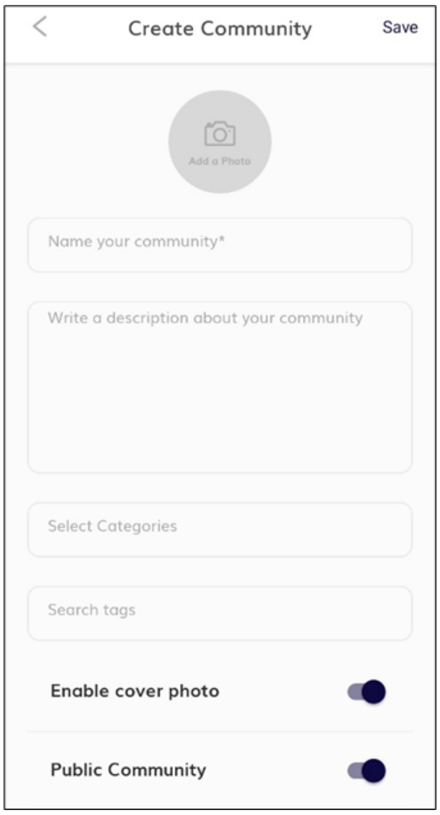
Chapter 2 Literature Review

This chapter showcases the reviews of similar projects based on their strengths and weaknesses. The strengths and weaknesses will be further described and summarized in a table for comparison.

2.1 Previous Works on Community App

2.1.1 hi-hive Community [2]

hi-hive Community App is a university-based community app that allows users to use digital platforms to create and manage their communities. Although UTAR mainly uses hi-hive for student attendance tracking purposes, it can be a great app for hosting various multitier communities if fully utilized. Users can create communities or subcommunities easily by filling in the required attributes such as community name, community description, category, and tags. Plus, users can personalize their community by customizing and applying various settings, as shown in Figure 2.1.1.1.



Enable cover photo

Public Community

Figure 2.1.1.1 Users creating their community

Strengths

i) In-App Chat Function

hi-hive Community offers an in-app chat function for its users. Users can create new chats with registered users as shown in Figure 2.1.1.2 and Figure 2.1.1.3. Users can send text messages, images, audio, videos, and document files to community members. It is a great communication tool enabling users to share private information instantly.

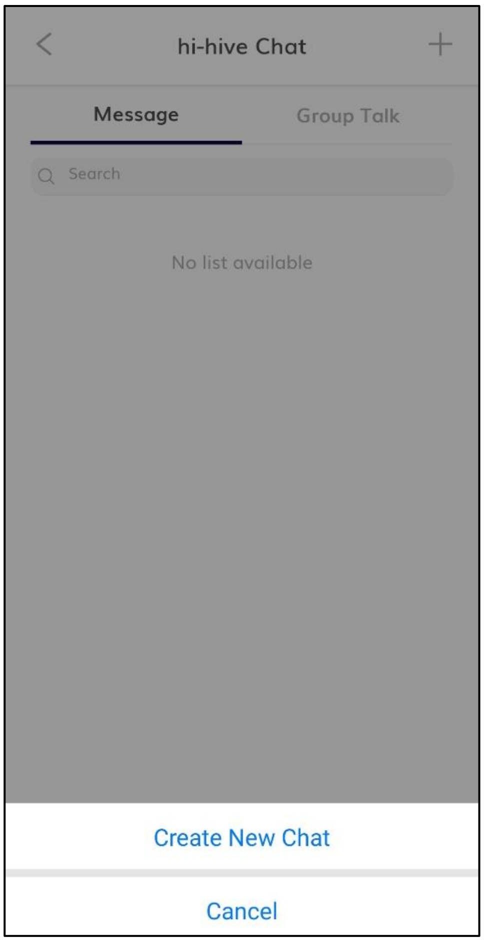


Figure 2.1.1.2 Create New Chat Function

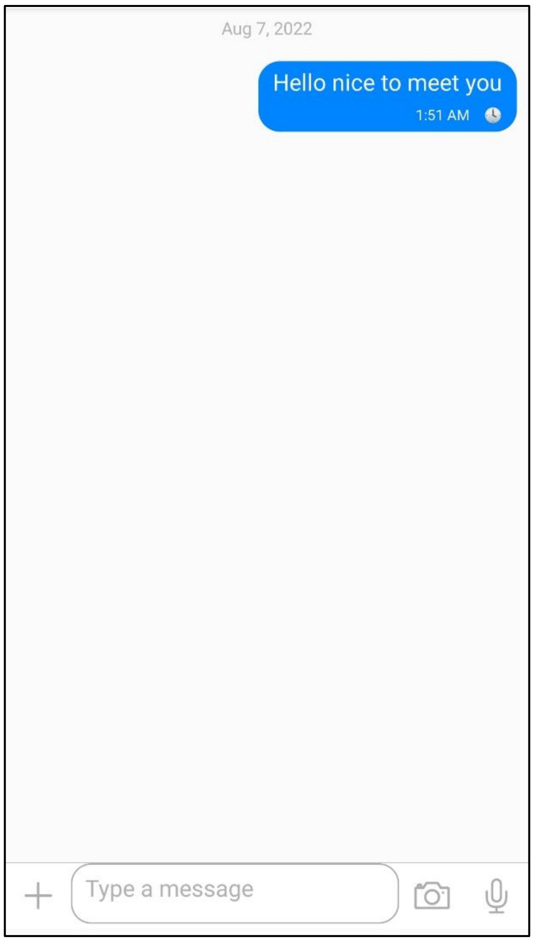


Figure 2.1.1.3 In-app messaging interface

ii) In-App Marketplace

hi-hive Community also offers In-App Marketplace where users can list their new or old items and sell them conveniently on the platform. Based on Figure 2.1.1.4, the marketplace will display a list of products with their respective picture, title, and price. Once users are interested in the product, they may click on the picture, and the app will direct them to the product description page, where they can learn more details about the product and the seller, as shown in Figure 2.1.1.5. When users wish to buy the product from the seller, they may click on the "Contact Seller" button, and the app will prompt 2 options for the user, either Call or Message the seller. The "Contact Seller" button made it very convenient for interested buyers to contact sellers as they can directly contact the sellers.

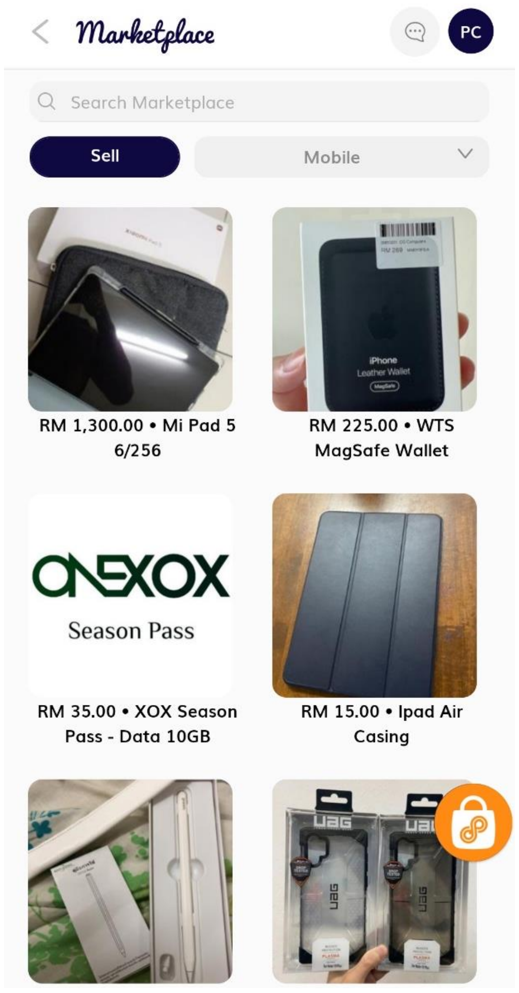
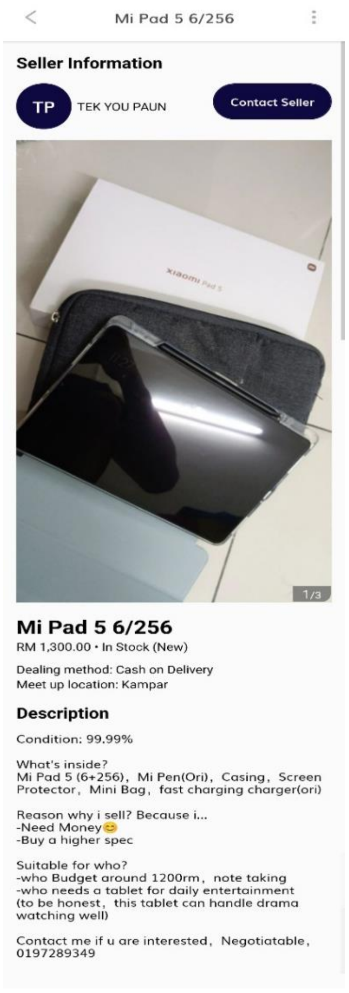

Marketplace
PC
Search Marketplace
Sell
Mobile
RM 1,300.00 • Mi Pad 5 6/256
RM 225.00 • WTS MagSafe Wallet
ONEXOX
Season Pass
RM 35.00 • XOX Season Pass - Data 10GB
RM 15.00 • Ipad Air Casing

Figure 2.1.1.4 Marketplace Product Listings



Mi Pad 5 6/256
RM 1,300.00 • In Stock (New)

Dealing method: Cash on Delivery
Meet up location: Kampar

Description

Condition: 99.99%

What's inside?
Mi Pad 5 (6+256), Mi Pen(Ori), Casing, Screen Protector, Mini Bag, fast charging charger(ori)

Reason why i sell? Because i...
-Need Money😊
-Buy a higher spec

Suitable for who?
-who Budget around 1200rm, note taking
-who needs a tablet for daily entertainment
(to be honest, this tablet can handle drama watching well)

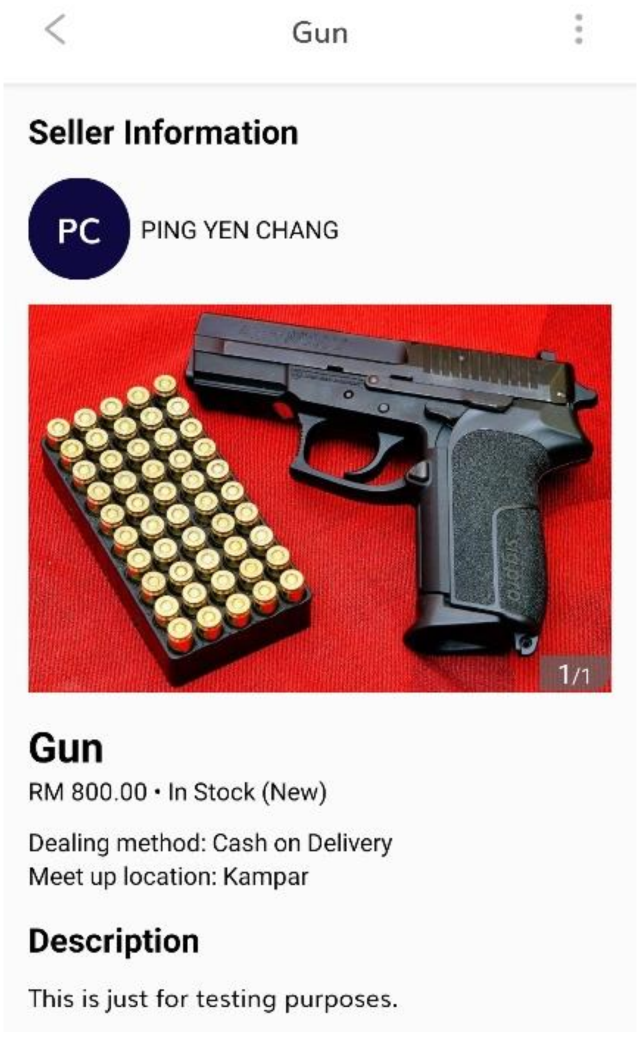
Contact me if u are interested, Negotiatable, 0197289349

Figure 2.1.1.5 Seller and Product Information

Weakness

i) Marketplace product listing is not moderated

There is no limitation on the things people can sell. Anyone can sell anything on the marketplace by creating a new listing. Figure 2.1.1.6 shows a sample product listing created for testing purposes. Even though gun laws in Malaysia are very restrictive, sellers can still create product listings to sell firearms illegally. Without proper moderation, users can take advantage of the Marketplace platform and sell anything they want, whether legal or illegal.



PING YEN CHANG

Gun

RM 800.00 • In Stock (New)

Dealing method: Cash on Delivery
Meet up location: Kampar

Description

This is just for testing purposes.

Figure 2.1.1.6 Product Listing Creation (Firearms)

ii) Inconvenient QR Code scanning

Scanning QR codes is inconvenient for users as it takes a long time for them to scan their QR codes. The responsiveness of the app is not just slow sometimes. It may occur to users that they cannot scan the QR code. For UTAR students case, they are facing issues with scanning their QR code attendance. Students either cannot scan the QR code, or their attendance is not recorded after scanning the QR code.

iii) Lack of search bar

The hi-hive Community does not offer a search bar for its users. It makes it harder for users to look for what they need and may lead to a poor user experience.

2.1.2 Raklet [3]

Raklet is a corporate-based community app that empowers various communities to meet, network and grow. Users can also monetize their communities through memberships and other digital tools.

Strengths

i) Membership application forms

Users interested in joining a community can click on the green **"Become Member"** button as shown in Figure 2.1.2.1. Once they click the button, they will be directed to an application form where users must input their email address, first/last name and agree to the terms and conditions. Once users click apply, the community admin will receive the membership application for review before approving the user as a member. This function helps community leaders increase the application process's efficiency as all details about the user are automatically captured and stored in the database.

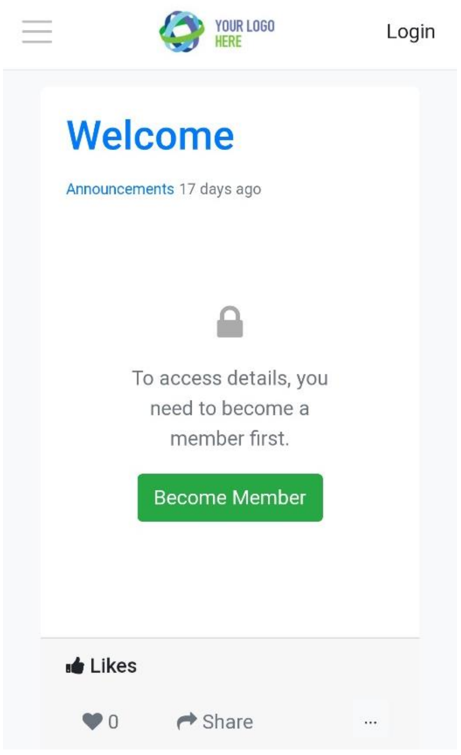
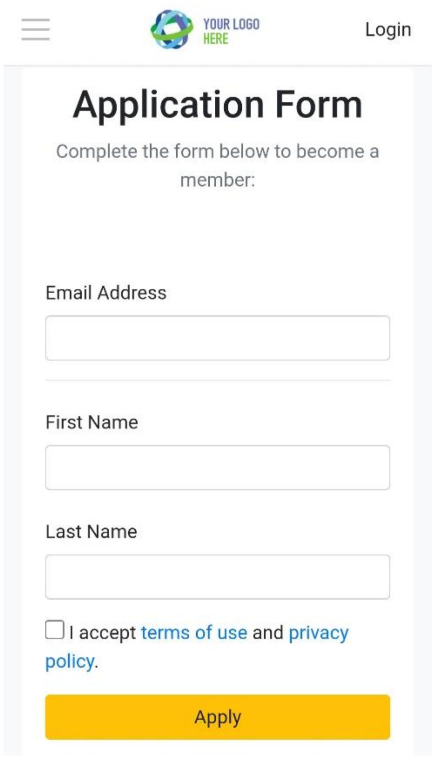

YOUR LOGO HERE
Login
Welcome
Announcements 17 days ago
To access details, you need to become a member first.
Become Member
Likes
0
Share
...

Figure 2.1.2.1 Welcome announcement



Application Form

Complete the form below to become a member:

Email Address

First Name

Last Name

☐ I accept terms of use and privacy policy.

Apply

Figure 2.1.2.2 Application form for members

ii) Membership Plans

Community leaders can create membership plans and collect payments from members. Members can subscribe to membership plans and make payments easily by adding their credit card information. The amount on the credit card will be deducted based on their subscription membership plan. Members who subscribe to the membership plan are eligible to gain extra benefits compared to regular members. When community leaders create a membership plan, they can decide the membership fees and the payment interval, whether it is one-time, yearly or monthly. Besides, community leaders can decide whether to cancel a member's subscription or leave it if members do not pay accordingly. The membership plan also includes automated reminders to remind members about specific events such as payment received, required email activation, membership approval etc.

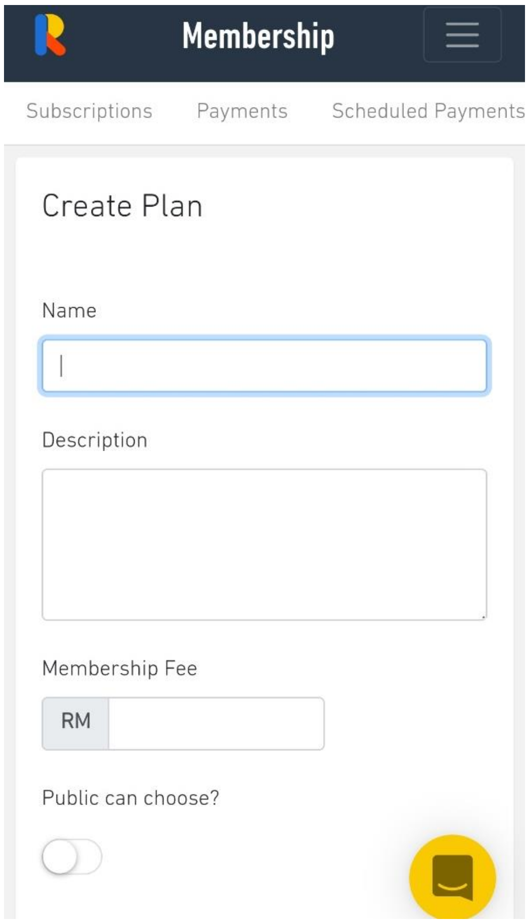


Figure 2.1.2.3 Membership Plan panel pt1

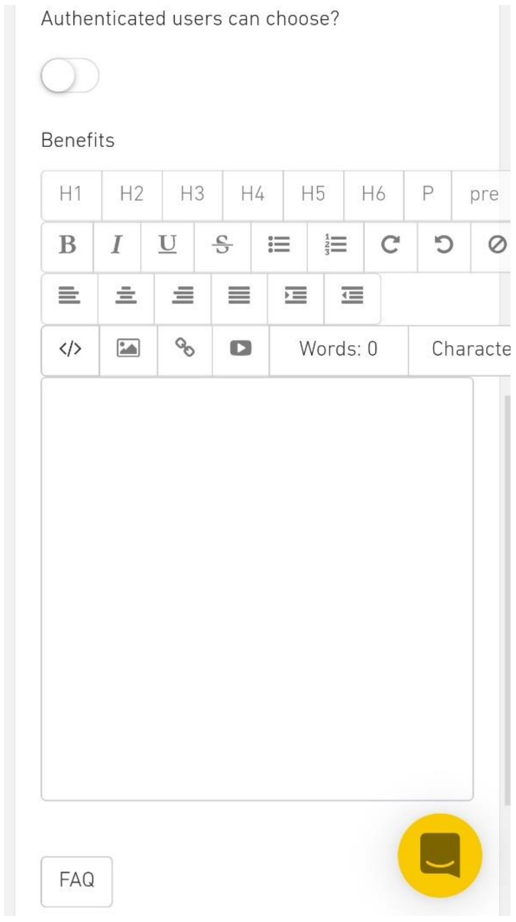
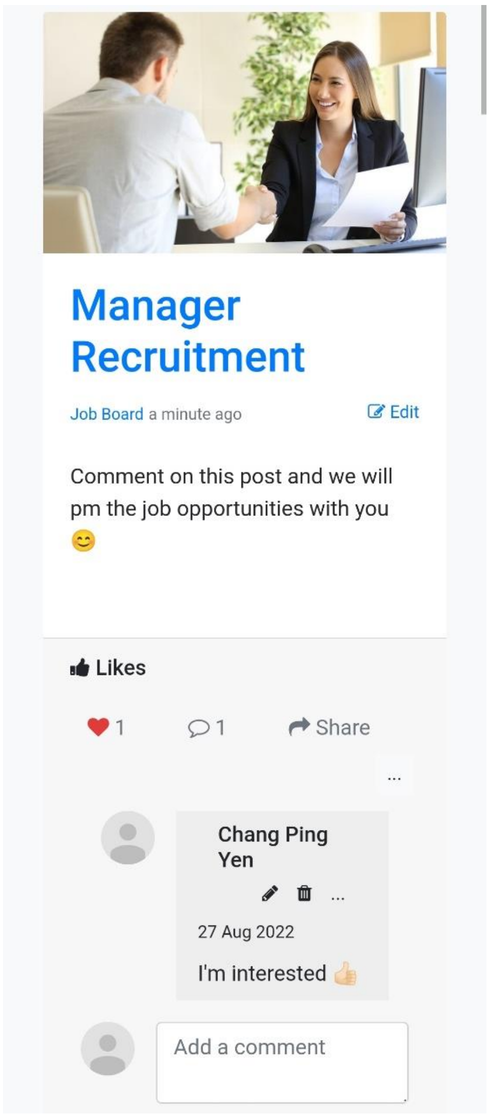


Figure 2.1.2.4 Membership Plan panel pt2

iii) Board Creation

Users interested in seeking jobs may join relevant communities to get exposed to more job options. Community leaders can create various boards for their community members to post. For this app, the more common boards created are Job Boards, where members can browse the job listings and search for job opportunities. Figure 2.1.2.5 shows an example job board posted, and members within the community can freely engage with the post by liking, commenting or sharing.


Manager Recruitment

Job Board a minute ago

Comment on this post and we will pm the job opportunities with you 😊

Chang Ping Yen

27 Aug 2022

I'm interested 👍

Figure 2.1.2.5 Job Board Example

Weakness

i) Requires internet access

For users to use the app, their mobile phones must have an internet or data connection. Without a stable internet connection, they can no longer gain access to the app. It is because the app requires a server to help transmit data.

ii) Confusing menus and forms

Although the app offers a variety of forms for members to fill in, it can be very confusing for members on certain occasions as the forms contain many attributes required to fill in by members.

2.1.3 Reddit [4]

Reddit is a popular website home to many communities ranging from sports, gaming, beauty, etc. Community members can share content by posting images, text, or videos. They can also comment or vote on the post. The content with many comments and high votes will rise to the top for more people to view. Figure 2.1.3.1 and 2.1.3.2 shows a sample interface of Reddit in dark mode.

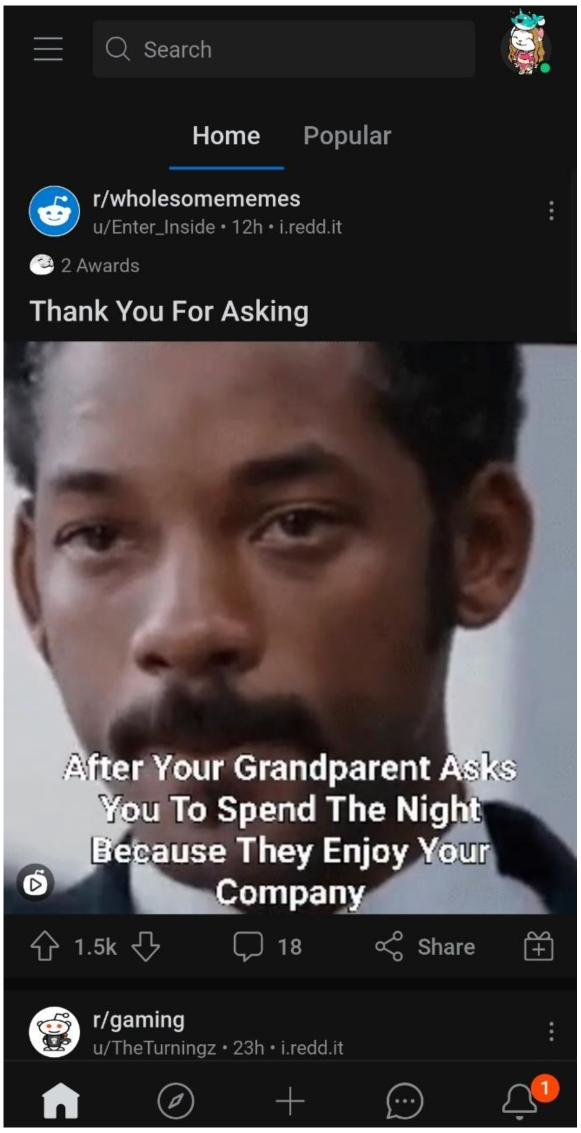
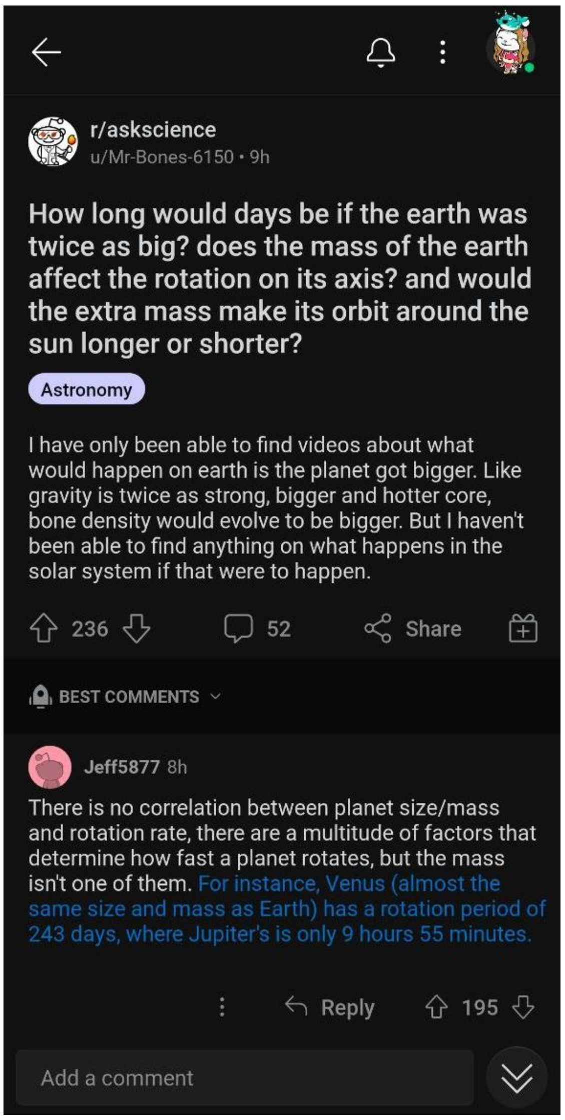

Search
Home
Popular
r/wholesomememes
u/Enter_Inside • 12h • i.redd.it
2 Awards
Thank You For Asking
After Your Grandparent Asks You To Spend The Night Because They Enjoy Your Company
1.5k
18
Share
r/gaming
u/TheTurningz • 23h • i.redd.it
1

Figure 2.1.3.1 Reddit homepage


r/askscience
u/Mr-Bones-6150 • 9h

How long would days be if the earth was twice as big? does the mass of the earth affect the rotation on its axis? and would the extra mass make its orbit around the sun longer or shorter?

Astronomy

I have only been able to find videos about what would happen on earth is the planet got bigger. Like gravity is twice as strong, bigger and hotter core, bone density would evolve to be bigger. But I haven't been able to find anything on what happens in the solar system if that were to happen.

Jeff5877 8h

There is no correlation between planet size/mass and rotation rate, there are a multitude of factors that determine how fast a planet rotates, but the mass isn't one of them. For instance, Venus (almost the same size and mass as Earth) has a rotation period of 243 days, where Jupiter's is only 9 hours 55 minutes.

Figure 2.1.3.2 Sample Reddit post

Strengths

i) Bottom navigation

Figure 2.1.3.3 shows the Reddit app has a bottom navigation bar that allows users to navigate to different destinations and perform various functions. The first icon represents the home icon, where users can easily navigate to the app's homepage to view interesting posts from community members. Then the next icon represents the discovery icon, where users can discover other communities on different topics such as science, memes, celebrity and many more. Then the middle icon is the add new post function, where users can create a new post in an image, video, text, link, poll, or talk. After deciding on the type of content they would like to create, they can post it on their profile or in other communities. The fourth icon represents the chat icon. Users can start a direct chat by copying an invite link and posting it to let other users join freely or directly input the individual's username.

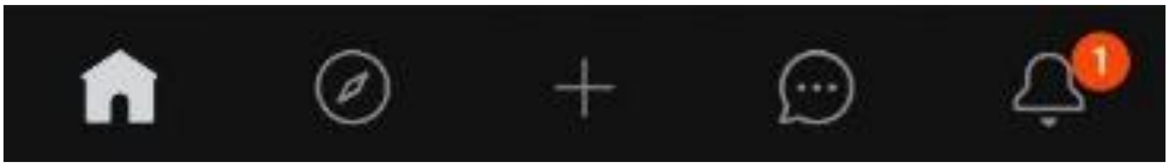


Figure 2.1.3.3 Reddit bottom navigation

ii) Ranking algorithms

Reddit has its unique ranking algorithm for each post and comment. For the Reddit story, it uses the Reddit hot ranking algorithm. This algorithm uses the number of votes and submission time to determine where a post will rank [5]. For a post, the first votes are much more valuable than a later vote, which means the impact of the upvotes will degrade as the story gets older. For the comment ranking, it uses the "Wilson score interval". Comments are ranked by confidence and data sampling. The submission time is irrelevant and would not affect the ranking. For example, a comment with 30 upvotes and 3 downvotes will still rank higher than comment with 1 upvote but 0 downvotes.

iii) Anonymous browsing

Reddit allows its users to perform anonymous browsing as shown in Figure 2.1.3.4. Anonymous browsing means users can browse the app without creating an account. It allows users to experience what the app can provide, and if they find the app interesting or wish to have the app personalize their experience, they may proceed to register an account.

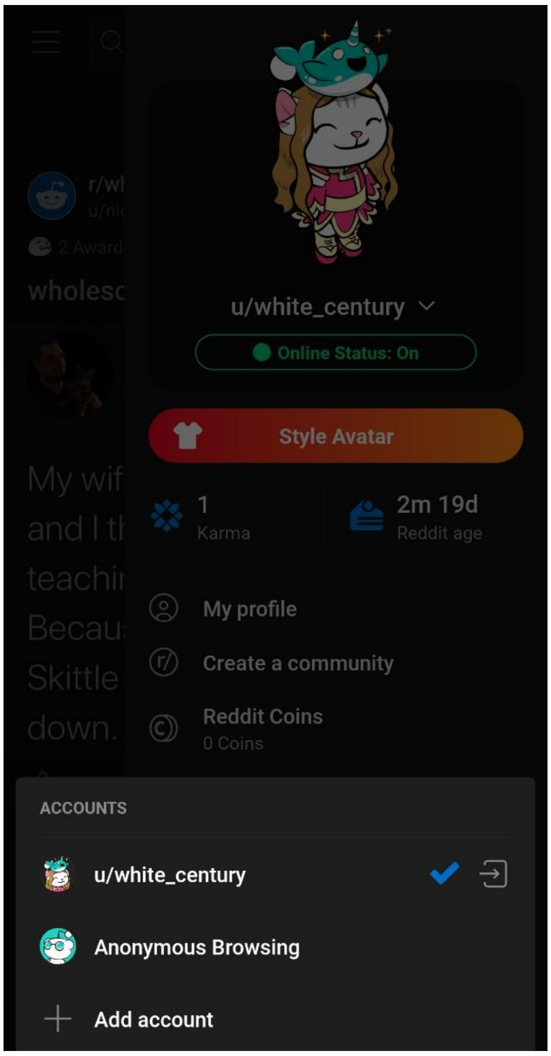


Figure 2.1.3.4 Anonymous browsing option

Weakness

i) Abuse of power among moderators

Moderators, also known as mods on Reddit, are the Reddit users who volunteer to moderate Reddit communities. They have the power to remove any posts/comments and even ban other users who break the community rules. Certain smaller Reddit communities, also known as subreddits, can have stringent posting rules which moderators often set. These posting rules are too strict that new members are often too intimidated to post as they fear their posts violating the rules and being removed.

ii) Source is not reliable

Anyone on Reddit can share any information on the platform without verifying its truthfulness. Therefore, it is easy for uninformed users to spread fake information.

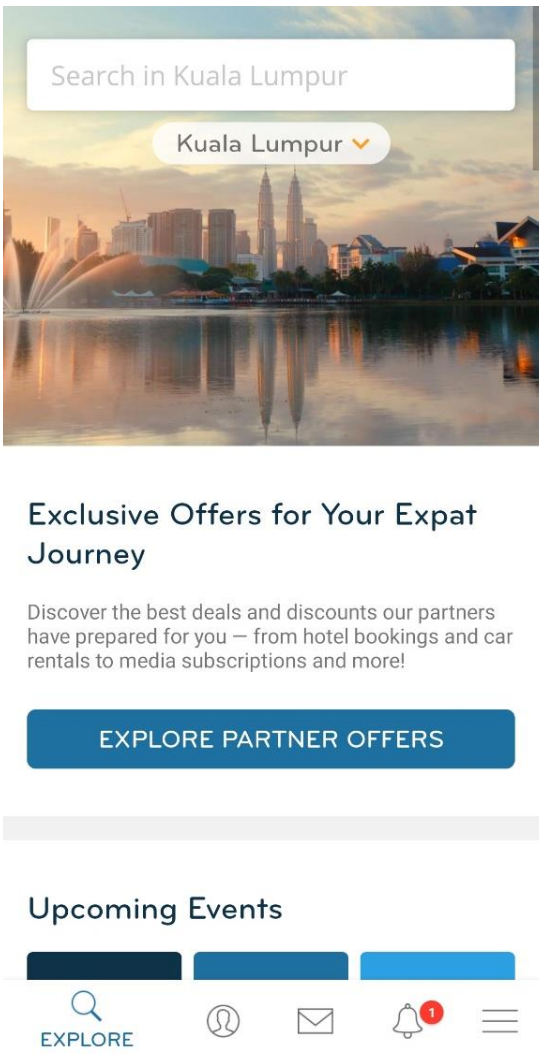
2.1.4 InterNations [6]

InterNations is an app that connects expats with people within their current location. Users can use the app to socialize with others both online and physically.

Strengths

i) Location-based community

The app allows users to search for communities based on location. Users can either allow the app to access their location or key in the desired location to identify whether there exists a community for that location.



Exclusive Offers for Your Expat Journey

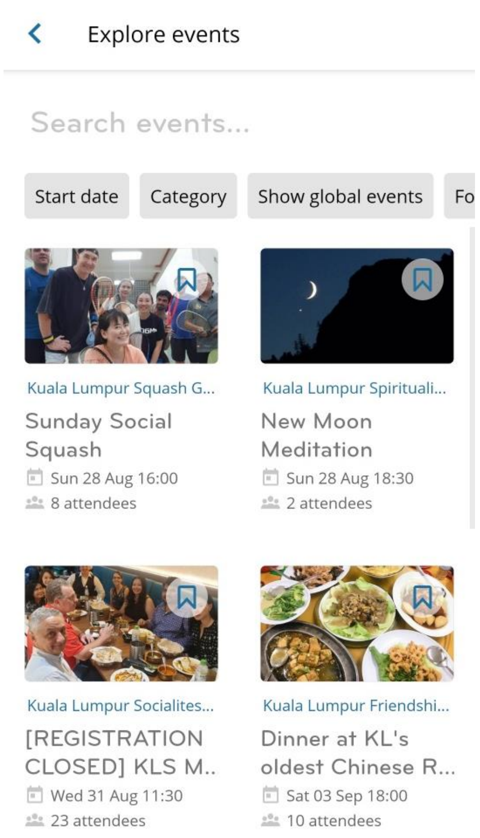
Discover the best deals and discounts our partners have prepared for you — from hotel bookings and car rentals to media subscriptions and more!

Upcoming Events

Figure 2.1.4.1 Search function for location-based communities

ii) Explore events

The app also allows users to explore events hosted near their location. Figure 2.1.4.2 shows a list of events available nearby the user's location. The available event will show basic information, such as the event title, date, and the number of attendees. It allows users to have a brief overview of the event, and if they are interested, they may further click on the event to gain more details. After clicking on the event, users will be directed to the event's description page, as shown in Figure 2.1.4.3, where more detailed information about the event is shown. Users may attend, invite, share or even report the event.



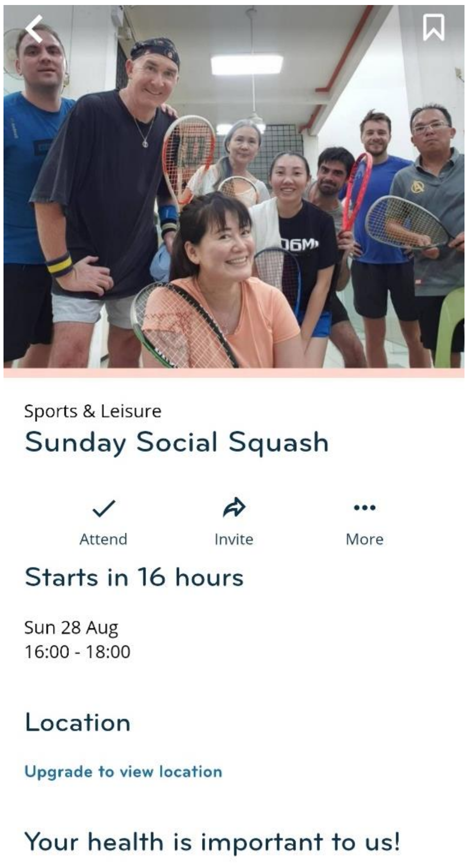
Kuala Lumpur Squash G...
Sunday Social Squash
Sun 28 Aug 16:00
8 attendees

Kuala Lumpur Spirituali...
New Moon Meditation
Sun 28 Aug 18:30
2 attendees

Kuala Lumpur Socialites...
[REGISTRATION CLOSED] KLS M..
Wed 31 Aug 11:30
23 attendees

Kuala Lumpur Friendshi...
Dinner at KL's oldest Chinese R...
Sat 03 Sep 18:00
10 attendees

Figure 2.1.4.2 Explore events page



Sports & Leisure
Sunday Social Squash

Starts in 16 hours

Sun 28 Aug
16:00 - 18:00

Location

Upgrade to view location

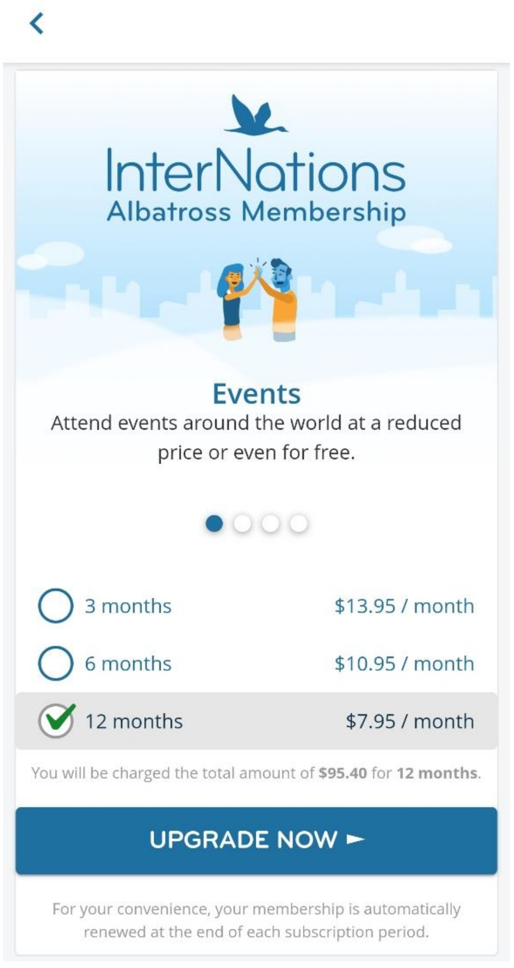
Your health is important to us!

Figure 2.1.4.3 Event description page

Weakness

i) Opt-out billing system

There are 2 types of membership statuses in this app: basic member and albatross. For the albatross membership, the app uses an opt-out billing system which will automatically renew users' subscriptions, and the amount will be credited to users' bills without the user's consent. If users refuse to pay, the company will send spam emails reminding users to pay. It also has unclear guidelines on how to cancel the membership, and users often do not know how to cancel their membership and would end up requiring help from the app's support team.



InterNations
Albatross Membership

Events
Attend events around the world at a reduced price or even for free.

3 months	$13.95 / month
6 months	$10.95 / month
✓ 12 months	$7.95 / month

You will be charged the total amount of **$95.40** for **12 months**.

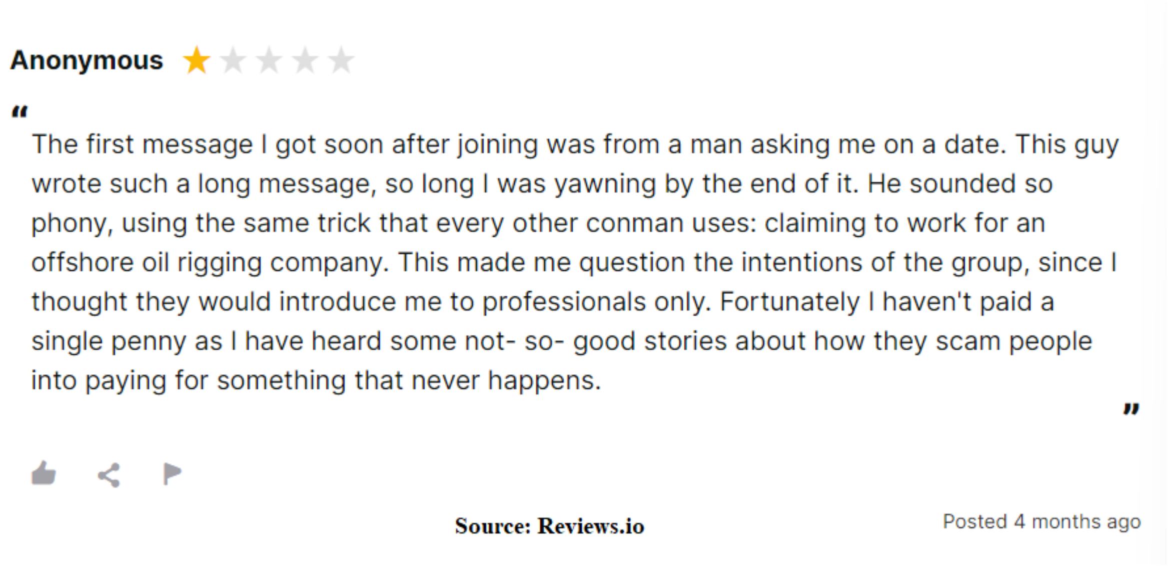
UPGRADE NOW ►

For your convenience, your membership is automatically renewed at the end of each subscription period.

Figure 2.1.4.4 InterNations membership plan

ii) No regulations on Internet Harassment

Based on Figure 2.1.4.5, shows a review of a woman being harassed by men on the app. Although there are community guidelines for the app, where users can block individuals, many still take advantage of the app's good intentions to harass and prey on innocent users.



Anonymous ★☆☆☆☆

"The first message I got soon after joining was from a man asking me on a date. This guy wrote such a long message, so long I was yawning by the end of it. He sounded so phony, using the same trick that every other conman uses: claiming to work for an offshore oil rigging company. This made me question the intentions of the group, since I thought they would introduce me to professionals only. Fortunately I haven't paid a single penny as I have heard some not- so- good stories about how they scam people into paying for something that never happens."

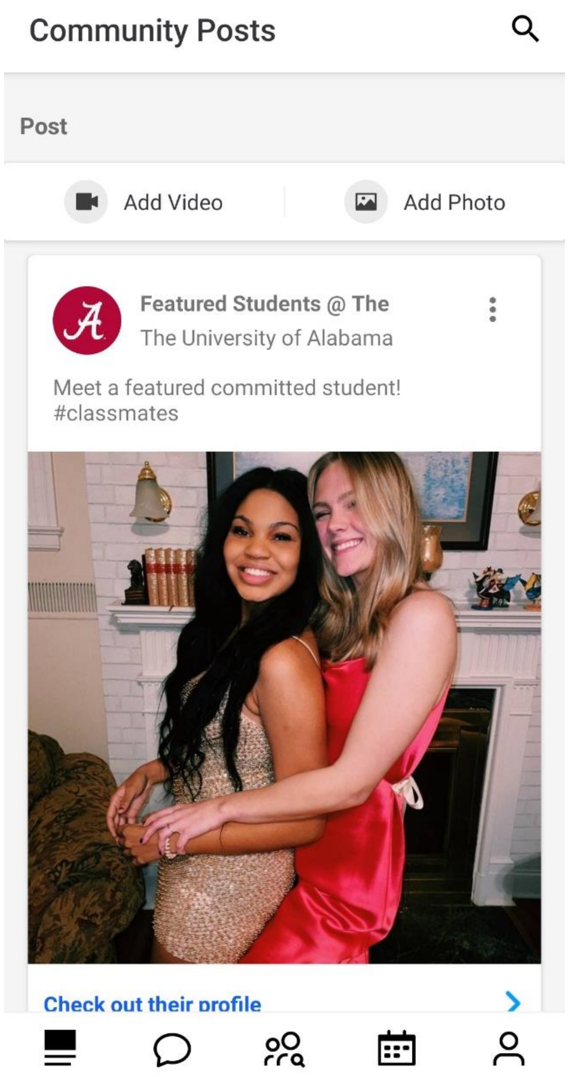
Source: Reviews.io

Posted 4 months ago

Figure 2.1.4.5 Review of InterNations app [7]

2.1.5 ZeeMee [8]

ZeeMee is an app for college students and applicants. Like most community apps, it allows users to post photos and videos with captions. Students can search the community based on the university there are enrolled in or interested in applying. Figure 2.1.5.1 shows a sample community post from students at the University of Alabama.



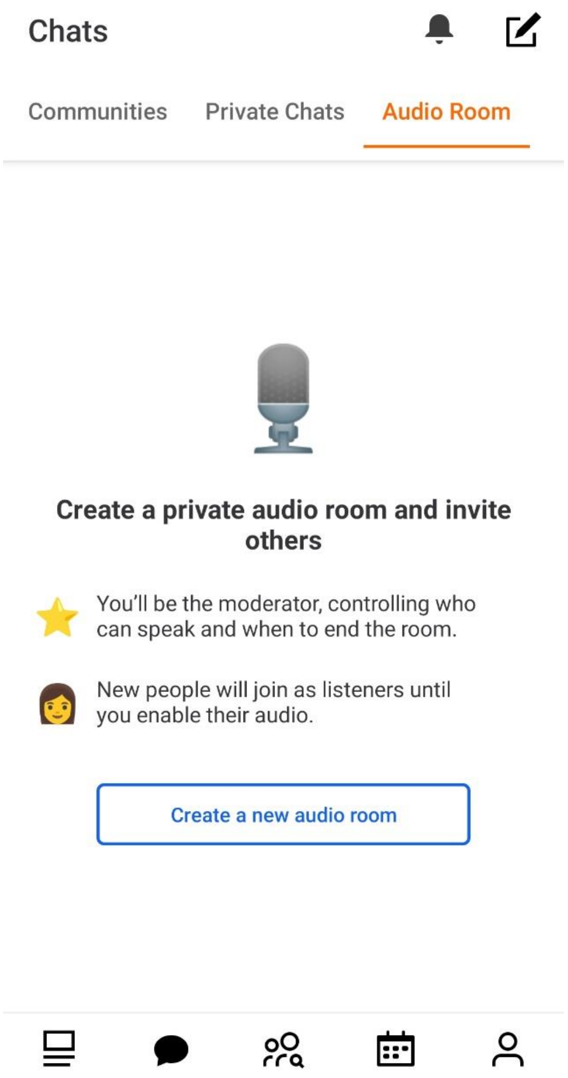
Meet a featured committed student!
#classmates

Figure 2.1.5.1 Homepage of ZeeMee App

Strengths

i) Audio Room

Figure 2.1.5.2 shows the audio room function where users can create an audio room where participants can chat using voice. The creator, also known as the moderator, can control who can speak and when to end the room. Once the room is created, moderators can copy the invite link and send the link to other ZeeMee users. Users who click the link will join as new listeners, and their microphones will be muted until they enable their audio.



Create a private audio room and invite others

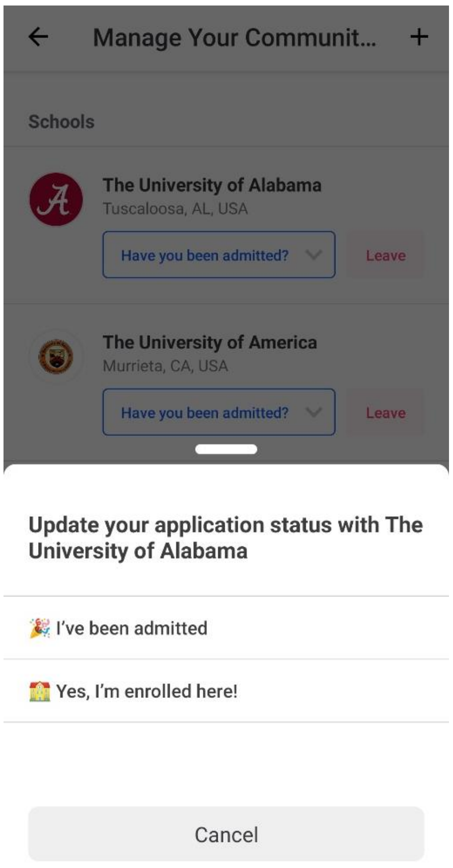
You'll be the moderator, controlling who can speak and when to end the room.

New people will join as listeners until you enable their audio.

Figure 2.1.5.2 Audio Room

ii) Identity Verification

Figure 2.1.5.3 and Figure 2.1.5.4 show the identity verification steps that students must do to verify their identity as one of the university's students before joining the university community. Users can update their application status, either admitted or enrolled. Admitted students must upload a picture of their acceptance letter or student ID. Once students are verified, they may perform more features such as viewing other admitted students, joining admitted students' chats and finding suggested roommates.


Manage Your Communit...

Schools

The University of Alabama
Tuscaloosa, AL, USA
Have you been admitted?
Leave

The University of America
Murrieta, CA, USA
Have you been admitted?
Leave

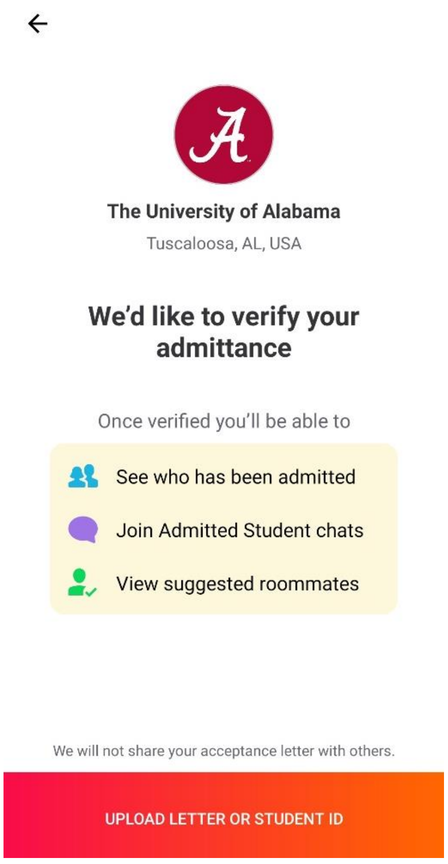
Update your application status with The University of Alabama

🎉 I've been admitted

Yes, I'm enrolled here!

Cancel

Figure 2.1.5.3 Update Application Status


The University of Alabama
Tuscaloosa, AL, USA

We'd like to verify your admittance

Once verified you'll be able to

See who has been admitted

Join Admitted Student chats

View suggested roommates

We will not share your acceptance letter with others.

UPLOAD LETTER OR STUDENT ID

Figure 2.1.5.4 Verify Admittance

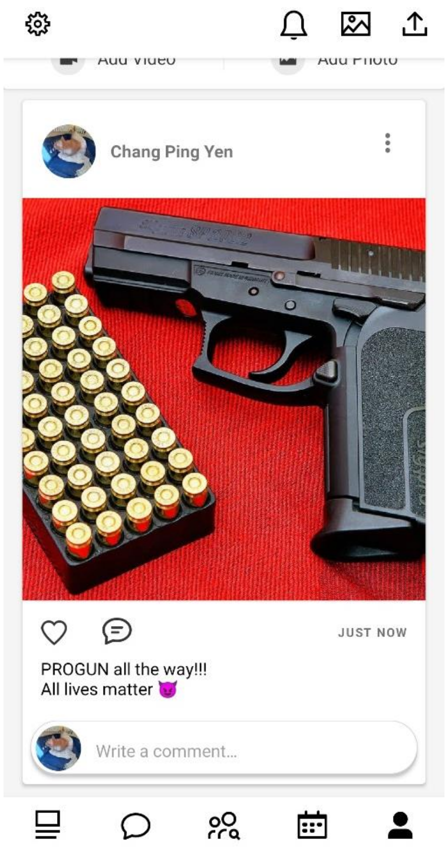
iii) Connect with friends/roommates

The app allows users to connect with friends/roommates from the same college. Furthermore, the search function includes other filters such as class of the year, interest, major and state.

Weakness

i) Lack of moderation on content sharing

Although this app is for college students, the app lacks moderation in content sharing. Users of the app can share and post any content to their heart's desire, even if they post anything inappropriate such as firearms, drugs, sexual etc., as shown in Figure 2.1.5.5.



PROGUN all the way!!!
All lives matter 😈

Figure 2.1.5.5 Post creation (firearms)

ii) Privacy Policy Problem

According to the terms and conditions written on the ZeeMee official website, there is a statement saying if users do not activate any privacy settings, then the user's profile will be publicly available to all users of the app [8]. This default privacy setting is an issue as most students would not look through the terms and conditions before agreeing to them, which means the students are likely unaware that their profiles are publicly visible to everyone.

2.2 Summary

For this chapter, research has been done on 5 similar community apps (**hi-hive Community, Raklet, Reddit, InterNations, and ZeeMee**). Their overall strengths and weakness are summarized in a table as shown in Table 2.2.

Features	**hi-hive Community**	**Raklet**	**Reddit**	**InterNations**	**ZeeMee**
Types of community	University Community	Corporate Community	Social Community	Expat Community	University Community
Login/Signup to use	Yes	Yes	No	Yes	Yes
Search function	No	Yes	Yes	Yes	Yes
Anonymous browsing	No	No	Yes	No	No
Chat function	Yes	Yes	Yes	Yes	Yes
Community creation	Yes, but with admin approval	Yes	Yes	Only can create events	Only can create events
Content Moderation	Weak	Depends on moderator	Depends on moderator	Weak	Weak
View Profile	No	No	Yes	Yes	Yes
Membership Plans	No	- Subscription plan for community leaders to access exclusive dashboard functions - Respective community leaders decide the	- Reddit Premium allows users to access exclusive content from the best commenters on Reddit	- InterNations basic membership provides limited features, which only involve allowing members to	No

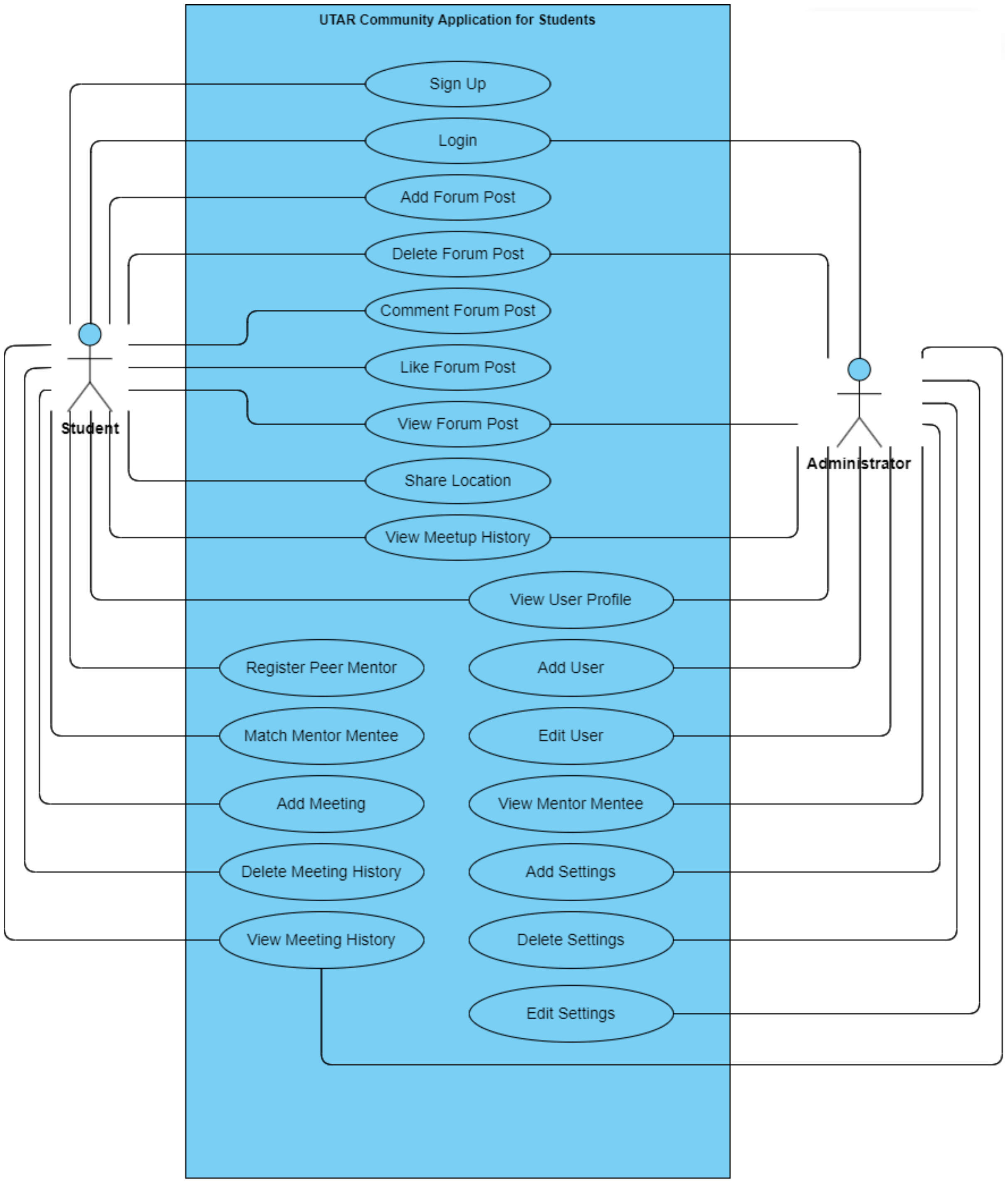
		membership plans of community members		search and view other member's post - Albatross membership provides users more features but comes with unclear billing terms and conditions	
Connectivity Requirements	Requires internet connection	Requires internet connection	Requires internet connection	Requires internet connection	Requires internet connection
Overall design of the user interface and app usability	Simple design but is hard to navigate.	It can be confusing for new users. They may need time to figure out ways to navigate the app.	Simple and easy to navigate, but users may be overwhelmed with too much content.	Simple and easy to navigate but with limited functions for basic members.	Simple in design and easy to use.

Table 2.2 Comparison of existing community apps

Chapter 3 System Design

Chapter 3 explains the overall system design, including a system use case diagram and use case descriptions to show the interactions between the user and the system. Next, the chapter also showcases screenshots of the program development with relevant explanations.

3.1 Use Case Diagram



3.2 Use Case Description

Use Case ID	F001	
Use Case	Sign Up	
Purpose	To allow new students to create an account to access the mobile application.	
Actor	Student	
Trigger	User clicks on the "Sign Up" button.	
Precondition	Account does not exist in the system.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student fills in full name.
	2	Student fills in email.
	3	Student fills in password and confirm password.
	4	Student clicks on the "Sign Up" button.
	5	System validates all fields entered.
	6	System will redirect user to respective panel.
Sub Flow	-	
Alternate Flow – Full Name is empty	5a.1	Student did not input full name.
	5a.2	System validates all fields entered.
	5a.3	System displays error message: "Please Input Full Name"
Alternate Flow – Invalid Email	5b.1	Student inputs invalid email.
	5b.2	System validates all fields entered.
	5b.3	System displays error message: "Email not valid!".
Alternate Flow – Invalid password	5c.1	Student inputs invalid password.
	5c.2	System validates all fields entered.
	5c.3	System displays error message: "Password cannot less than 6 characters!".
Alternate Flow – Email is empty	5d.1	Student / admin did not input email.
	5d.2	System validates all fields entered.
	5d.3	System displays error message: "Please Input Email".
Alternate Flow – Password is empty	5e.1	Student did not input password.
	5e.2	System validates all fields entered.
	5e.3	System displays error message: "Please Input Password ".

Table 3.2.1 Sign Up Use Case Description

Use Case ID	F002	
Use Case	Login	
Purpose	To allow students and administrators gain access to the mobile application.	
Actor	Student, Admin	
Trigger	User clicks on the "Login" button.	
Precondition	Account exists in the system and not logged in.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student / admin fills in email and password.
	2	Student / admin clicks on the "Login" button.
	3	System validates whether the account exists, and the email and password are correct.
	4	System will redirect user to respective panel.
Sub Flow	-	
Alternate Flow – Invalid Email	3a.1	Student / admin inputs invalid email.
	3a.2	System validates user email and password.
	3a.3	System displays error message: "Email not valid!".
Alternate Flow – Invalid Password	3b.1	Student / admin inputs invalid password.
	3b.2	System validates user email and password.
	3b.3	System displays error message: "The password is invalid".
Alternate Flow – Email is empty	3c.1	Student / admin did not input email.
	3c.2	System validates user email and password.
	3c.3	System displays error message: "Please Input Email".
Alternate Flow – Password is empty	3d.1	Student / admin did not input password.
	3d.2	System validates user email and password.
	3d.3	System displays error message: "Please Input Password".

Table 3.2.2 Login Use Case Description

Use Case ID	F003	
Use Case	View Forum Post	
Purpose	To allow students and administrators view forum posts.	
Actor	Student, Admin	
Trigger	User clicks on the "View post" / "Home" option.	
Precondition	Account is logged in and on home page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student / admin chooses view post option in forum page.
	2	System validates student's / admin's option.
	3	Student / admin can view forum posts.
Sub Flow – Choose "View Post" option	2a.1 2a.2	Student clicks on the "View Post" button. Student gets to view all anonymous posts posted by other users.
Sub Flow – Choose "My Post" option	2b.1 2b.2	Student clicks on the "My Post" button. Student gets to view their own posts.
Sub Flow – Choose specific user	2c.1 2c.2	Admin clicks on "Home" option from side navigation menu. Admin gets to view posts made by all users with their identities shown.
Alternate Flow	-	

Table 3.2.3 View Forum Post Use Case Description

Use Case ID	F004	
Use Case	Add Forum Post	
Purpose	To allow students create anonymous forum post.	
Actor	Student	
Trigger	Student clicks on the "+" button.	
Precondition	Account is logged in and on forum page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student clicks on the "+" button in forum page.
	2	Student writes their post in the dialog box.
	3	Student chooses option for add forum post.
	4	System validates student's option.
	5	Post is posted in forum page.
Sub Flow	-	
Alternate Flow – Cancel post	4a.1 4a.2	Student chooses Cancel option. Post is not posted.

Table 3.2.4 Add Forum Post Use Case Description

Use Case ID	F005	
Use Case	Delete Forum Post	
Purpose	To allow students and administrators delete forum posts.	
Actor	Student, Admin	
Trigger	User clicks on the "Delete" button.	
Precondition	Account is logged in and on forum / home page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student / admin clicks on a post in forum / home page.
	2	Student / admin clicks on the "Delete" button.
	3	Student / admin chooses option for delete forum post.
	4	System validates student's / admin's option.
	5	Post is deleted from forum page.
Sub Flow	-	
Alternate Flow – Cancel delete	4a.1	Student / admin chooses Cancel option.
	4a.2	Post is not deleted from forum page.

Table 3.2.5 Delete Forum Post Use Case Description

Use Case ID	F006	
Use Case	Comment Forum Post	
Purpose	To allow students comment on forum posts.	
Actor	Student	
Trigger	Student clicks on the comment button under individual user's post.	
Precondition	Account is logged in and on forum page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student clicks on a user's post.
	2	Student writes their comment in the dialog box.
	3	Student chooses option for comment forum post.
	4	System validates student's option.
	5	Comment is posted.
Sub Flow	-	
Alternate Flow – Cancel comment	4a.1 4a.2	Student chooses Cancel option. Comment is not posted.

Table 3.2.6 Comment Forum Post Use Case Description

Use Case ID	F007	
Use Case	Like Forum Post	
Purpose	To allow students like forum posts.	
Actor	Student	
Trigger	User clicks on the "♥" button under individual user's post.	
Precondition	Account is logged in and on forum page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student clicks on a user's post.
	2	Student clicks on the "♥" button.
	3	System displays like count +1.
	4	Student successfully like the post.
Sub Flow	-	
Alternate Flow – Unlike	2a.1	Student double clicks on the "♥" button.
	2a.2	Student unlike the post.

Table 3.2.7 Like Forum Post Use Case Description

Use Case ID	F008	
Use Case	View Meetup History	
Purpose	To allow students and administrators view meetup history.	
Actor	Student, Admin	
Trigger	User clicks on "View Meetup History"	
Precondition	Account is logged in and on meetup history page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student / admin chooses view meetup history option.
	2	System validates student's / admin's option.
	3	Student / admin can view meetup history.
Sub Flow – Choose "Approved" option	2a.1 2a.2	Student / admin clicks on the "Approved" option. Student / admin gets to view all approved meetup request.
Sub Flow – Choose "Rejected" option	2b.1 2b.2	Student / admin clicks on the "Rejected" option. Student / admin gets to view all rejected meetup requests. It will include the reason of rejection.
Sub Flow – Choose "Pending" option	2c.1 2c.2	Student / admin clicks on the "Pending" option. Student / admin gets to view all pending meetup requests.
Alternate Flow	-	

Table 3.2.8 View Meetup History Use Case Description

Use Case ID	F009	
Use Case	Share Location	
Purpose	To allow students share their location with one another.	
Actor	Student A, Student B	
Trigger	Student clicks on "Active Current Location" button.	
Precondition	Account is logged in and on homepage.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student A and Student B clicks on "Active Current Location" button.
	2	Student A clicks on Student B's location marker, which is available on the map.
	3	System displays dialog box with message: "Ask Student B to meet" and a button to request location permission.
	4	Student A clicks on "Yes request permission" button in dialog box.
	5	Student B receives a request to meet.
	6	Student B chooses option for request to meet.
	7	Student A receives approved request.
	8	Student A clicks on button to navigate to Student B's location.
	9	When Student A meets Student B, Student A will click on "Scan QR Code".
	10	Student B display respective QR code in Meetup History panel for Student A to scan.
	11	Student A scans Student B's QR code.
	12	System display success message to indicate students are meeting the right person.
	13	Student A chooses meetup status option.
	14	Students completed the meetup and data will be recorded into the system.
Sub Flow	-	
	6a.1	Student B rejects the request.

Alternate Flow – Reject request to meet	6a.2	Location is not shared, and meetup is rejected.
Alternate Flow – Scan wrong QR code	11a.1 11a.2	Student A scan the wrong QR code. System displays error message: "Invalid QR code".
Alternate Flow – Choose "No" option	13a.1 13a.2	Student A clicks on "No" option for meetup status. Meetup is not successful, and data will be recorded into the system.

Table 3.2.9 Share Location Use Case Description

Use Case ID	F010	
Use Case	View Meeting History	
Purpose	To allow students and administrators view meeting history.	
Actor	Student, Admin	
Trigger	User clicks on the "Meeting History" option.	
Precondition	Student: Account is logged in and on peer meetings page. Admin: Account is logged in and on meeting history page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student / admin clicks on "Meeting History" option.
	2	Student / admin can view meeting history which includes information such as the name of mentor / mentee, meeting name, date and time, status and type of meeting (Physical meeting / Online meeting).
Sub Flow – Choose "Accepted" / "Done" option	2a.1 2a.2	Student clicks on "Accepted" button. / Admin clicks on "Done" button. Student / admin gets to view all completed meetings.
Sub Flow – Choose "Upcoming" / "Not Done" option	2b.1 2b.2	Student clicks on "Upcoming" button. Admin clicks on "Not Done" button. Student / admin gets to view all accepted but yet to complete meetings.
Sub Flow – Choose "Pending" option	2c.1 2c.2	Student / admin clicks on "Pending" button. Student / admin gets to view all pending to be accepted / rejected meetings.
Sub Flow – Choose "Reject" option	2d.1 2d.2	Student / admin clicks on "Reject" button. Student / admin gets to view all rejected meetings.
Alternate Flow	-	

Table 3.2.10 View Meeting History Use Case Description

Use Case ID	F011	
Use Case	Register Peer Mentor	
Purpose	To allow students register as mentor / mentee.	
Actor	Student	
Trigger	Student clicks on peer mentor page.	
Precondition	Account is logged in and has not registered as mentor / mentee before.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student clicks on peer mentor page.
	2	Student chooses whether to register as mentee or mentor.
	3	System requests student to fill in required details.
	4	Student fills in their details.
	5	Student clicks on submit button.
	6	System validates whether all details are filled in.
	7	System will redirect student to peer mentor home page.
Sub Flow	-	
Alternate Flow – Details are empty	6a.1	Student did not fill in all details required.
	6a.2	System displays error message: "Please enter details required".
Alternate Flow – Student academic years is less than 1 year	6b.1	Student wants to register as mentor, but years of study is less than 1 year.
	6b.2	System displays error message: "Less than 1 year of study cannot be mentor".

Table 3.2.11 Register Peer Mentor Use Case Description

Use Case ID	F0012	
Use Case	Match Mentor Mentee	
Purpose	To allow students choose their mentor based on their preference.	
Actor	Student	
Trigger	Student clicks on the "Match" button.	
Precondition	Account is logged in and on peer mentor page.	
Scenario Name	**Step**	**Action**
Main Flow	1	System displays a list of available mentors for matching.
	2	Mentee clicks the "Match" button for the mentors they want to match with.
	3	Mentor receives a match request from mentee.
	4	Mentor chooses option for match request.
	5	System validates match request.
	6	Mentee successfully matched with mentor.
Sub Flow	-	
Alternate Flow – Cancel match request	2a.1 2a.2	Mentee cancels match request. Mentor does not receive match request.
Alternate Flow – Reject match request	5a.1 5a.2 5a.3 5a.4	Mentor chooses Reject option. System request mentor to provide reason of rejection. Mentor fills in the reason of rejection. Mentee receives a message showing the reason of rejection.

Table 3.2.12 Match Mentor Mentee Use Case Description

Use Case ID	F013	
Use Case	Add Meeting	
Purpose	To allow students schedule meetings with their mentor / mentee.	
Actor	Student	
Trigger	Student clicks on the "+" button.	
Precondition	Account is logged in and on peer meetings page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student clicks on the "+" button in peer meetings page.
	2	System requests student to fill in required details such as meeting name, date, and time.
	3	Student fills in the details.
	4	Student chooses option for add meeting.
	5	System validates whether all details are filled in and student's option.
	6	Meeting with mentor/mentee is scheduled.
Sub Flow	-	
Alternate Flow – Cancel add meeting	5a.1	Student chooses Cancel option.
	5a.2	Meeting is not scheduled.
Alternate Flow – Details are empty	5b.1	Student did not fill in all details required.
	5b.2	System displays error message: "Please enter all required details".

Table 3.2.13 Add Meeting Use Case Description

Use Case ID	F014	
Use Case	Delete Meeting History	
Purpose	To allow students delete meeting history.	
Actor	Student	
Trigger	Student clicks on the "Delete" button.	
Precondition	Account is logged in and on peer meetings page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student chooses a meeting history.
	2	Student clicks on the "Delete" button.
	3	Student chooses option for delete meeting.
	4	System validates student's option.
	5	Meeting history is deleted.
Sub Flow	-	
Alternate Flow – Cancel delete	4a.1	Student chooses Cancel option.
	4a.2	Scheduled meeting is not deleted.

Table 3.2.14 Delete Meeting History Use Case Description

Use Case ID	F015	
Use Case	View User Profile	
Purpose	To allow students and administrators view user profile.	
Actor	Student, Admin	
Trigger	Student, Admin clicks on User Profile.	
Precondition	Account is logged in and on user profile page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Student / admin clicks on User Profile.
	2	Student / admin gets to view all user information such as name and email.
Sub Flow	-	
Alternate Flow	-	

Table 3.2.15 View User Profile Use Case Description

Use Case ID	F016	
Use Case	Add User	
Purpose	To allow administrators add a new user.	
Actor	Admin	
Trigger	Admin clicks on the "+" button.	
Precondition	Account is logged in and on admin panel.	
Scenario Name	**Step**	**Action**
Main Flow	1	Admin clicks on the "+" button in admin panel.
	2	System requests admin to fill in required details such as full name, email, and password.
	3	Admin fills in student's details.
	4	Admin chooses option for add user.
	5	System validates whether all details are filled in and admin's option.
	6	New user is added.
Sub Flow	-	
Alternate Flow – Cancel add user	5a.1 5a.2	Admin chooses Cancel option. New user is not added.
Alternate Flow – Details are empty	5b.1 5b.2	Admin did not fill in all details required. System displays error message: "Please enter all required details".

Table 3.2.16 Add User Use Case Description

Use Case ID	F017	
Use Case	Edit User	
Purpose	To allow administrators edit an existing user.	
Actor	Admin	
Trigger	Admin clicks on the "Update User" button.	
Precondition	Admin is logged in and click on a user to edit.	
Scenario Name	**Step**	**Action**
Main Flow	1	Admin clicks on a user.
	2	Admin edit information of user.
	3	Admin chooses option for edit user.
	4	System validates whether all details are filled in and admin's option.
	5	User information is updated.
Sub Flow	-	
Alternate Flow – Cancel edit user	4a.1 4a.2	Admin chooses Cancel option. User information is not edited.
Alternate Flow – Details are empty	4b.1 4b.2	Admin did not fill in all details required. System displays error message: "Please enter all required details".

Table 3.2.17 Edit User Use Case Description

Use Case ID	F018	
Use Case	View Mentor Mentee	
Purpose	To allow administrators view list of mentor / mentee.	
Actor	Admin	
Trigger	Admin clicks on the "Mentor" / "Mentee" button.	
Precondition	Admin is logged in and on "Peer Mentor" page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Admin clicks on "Peer Mentor" page.
	2	Admin chooses option to view mentor / mentee.
	3	System validates admin's option.
	4	Admin chooses specific user under mentor / mentee.
	5	System validates admin's option.
	6	Admin get to view details about mentor / mentee.
Sub Flow – Choose "Mentor" option	3a.1	Admin chooses Mentor option.
	3a.2	System display list of mentors.
Sub Flow – Choose "Mentee" option	3b.1	Admin chooses Mentee option.
	3b.2	System display list of mentees.
Alternate Flow	-	

Table 3.2.18 View Mentor Mentee Use Case Description

Use Case ID	F019	
Use Case	Add Settings	
Purpose	To allow administrators add options for faculty and years of study.	
Actor	Admin	
Trigger	Admin clicks on the "Settings" option.	
Precondition	Admin is logged in and on settings page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Admin clicks on Faculty / Years of Study.
	2	Admin clicks on "Add Data" button.
	3	Admin fill in required details.
	4	System validates whether details are filled in and admin's option.
	5	New option is added.
Sub Flow	-	
Alternate Flow – Cancel add data	4a.1 4a.2	Admin chooses Cancel option. New option is not added.
Alternate Flow – Details are empty	4b.1 4b.2	Admin did not fill in details required. System displays error message: "Please enter all required details".

Table 3.2.19 Add Settings Use Case Description

Use Case ID	F020	
Use Case	Delete Settings	
Purpose	To allow administrators delete options for faculty and years of study.	
Actor	Admin	
Trigger	Admin clicks on the "Settings" option.	
Precondition	Admin is logged in and on settings page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Admin clicks on Faculty / Years of Study.
	2	Admin clicks on existing options.
	3	Admin clicks on "Delete" button.
	4	Admin chooses option for delete option.
	5	System validates admin's option.
	6	Option is deleted.
Sub Flow	-	
Alternate Flow – Cancel delete	5a.1 5a.2	Admin chooses Cancel option. Option is not deleted.

Table 3.2.20 Delete Settings Use Case Description

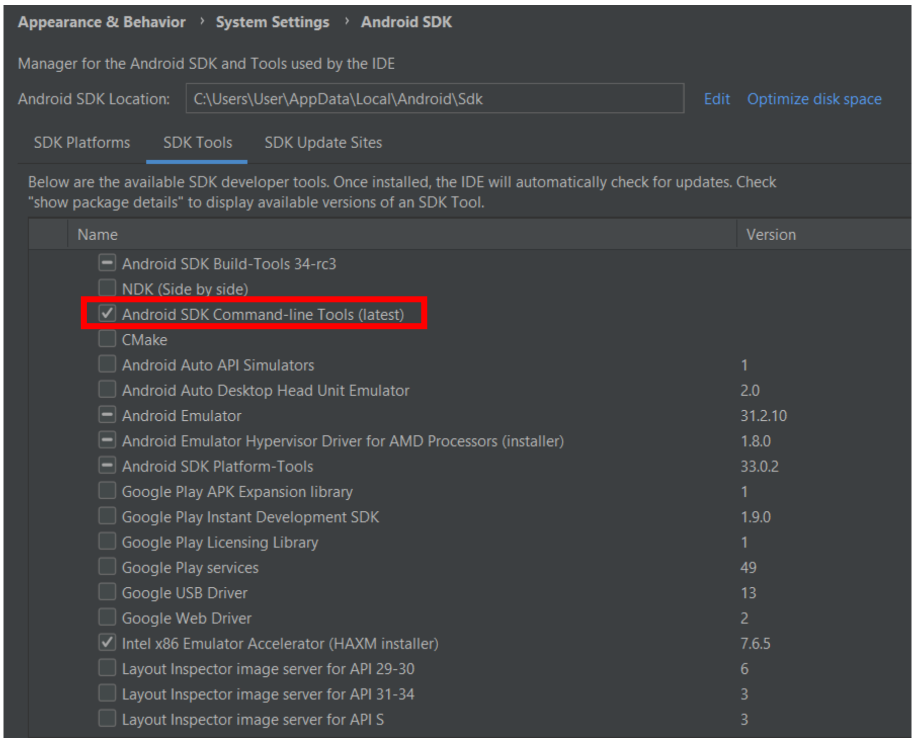
Use Case ID	F021	
Use Case	Edit Settings	
Purpose	To allow administrators edit options for faculty and years of study.	
Actor	Admin	
Trigger	Admin clicks on the "Settings" option.	
Precondition	Admin is logged in and on settings page.	
Scenario Name	**Step**	**Action**
Main Flow	1	Admin clicks on Faculty / Years of Study.
	2	Admin clicks on existing options.
	3	Admin clicks on "Edit" button.
	4	Admin chooses option for edit option.
	5	System validates admin's option.
	6	Option is edited.
Sub Flow	-	
Alternate Flow – Details are empty	5a.1 5a.2	Admin did not fill in details required. System displays error message: "Please enter all required details".
Alternate Flow – Cancel edit	5b.1 5b.2	Admin chooses Cancel option. Option is not edited.

Table 3.2.21 Edit Settings Use Case Description

3.3 Mobile App Development

3.3.1 Android Emulator Setup

Before testing the app on an actual mobile device, the app will be running on an Android Emulator. First, install Android SDK Command-line Tools (latest) package as shown in Figure 3.3.1.1. After the installation, configuration on the Android SDK and other platform tools is done.



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by the IDE

Android SDK Location: C:\Users\User\AppData\Local\Android\Sdk Edit Optimize disk space

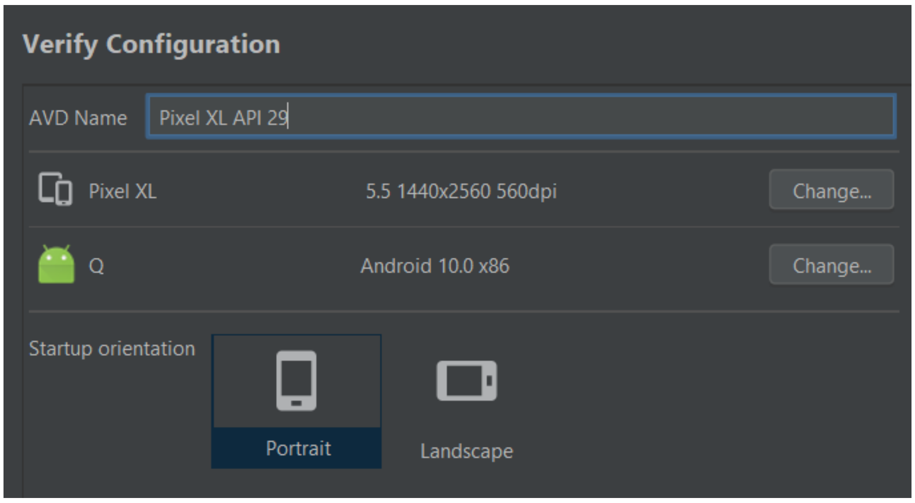
SDK Platforms | SDK Tools | SDK Update Sites

Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version
☐ Android SDK Build-Tools 34-rc3	
☐ NDK (Side by side)	
✓ Android SDK Command-line Tools (latest)	
☐ CMake	
☐ Android Auto API Simulators	1
☐ Android Auto Desktop Head Unit Emulator	2.0
☐ Android Emulator	31.2.10
☐ Android Emulator Hypervisor Driver for AMD Processors (installer)	1.8.0
☐ Android SDK Platform-Tools	33.0.2
☐ Google Play APK Expansion library	1
☐ Google Play Instant Development SDK	1.9.0
☐ Google Play Licensing Library	1
☐ Google Play services	49
☐ Google USB Driver	13
☐ Google Web Driver	2
✓ Intel x86 Emulator Accelerator (HAXM installer)	7.6.5
☐ Layout Inspector image server for API 29-30	6
☐ Layout Inspector image server for API 31-34	3
☐ Layout Inspector image server for API S	3

Figure 3.3.1.1 Install Android SDK Command-line Tools

Later, Android Studio is used to create a virtual device by choosing the device definition and Android API level, as shown in Figure 3.3.1.2.



Verify Configuration

AVD Name: Pixel XL API 29

Pixel XL — 5.5 1440x2560 560dpi — Change...

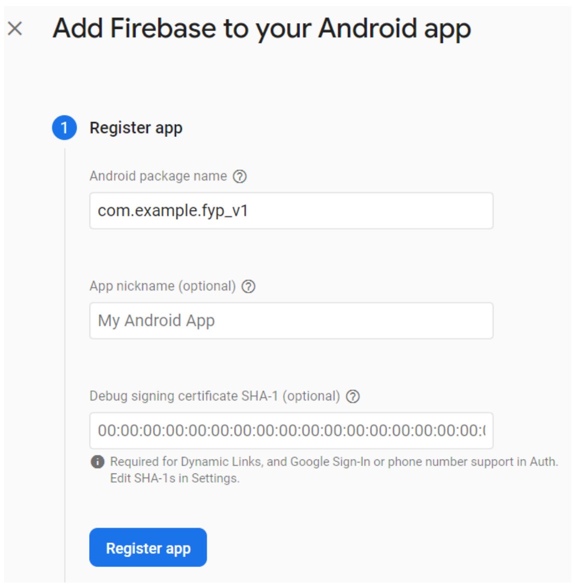
Q — Android 10.0 x86 — Change...

Startup orientation: Portrait | Landscape

Figure 3.3.1.2 Create virtual device

3.3.2 Connect Flutter Project with Firebase

The mobile application uses Firebase as the backend to store all the user data. First, create a new Firebase project on Firebase Console. The Firebase project will be given the name of "signup-login". Once the Firebase project is successfully created, the next step is to add Firebase to the Android App. Based on Figure 3.3.2.1, the Android app is registered by filling in the Android package name, also known as the application ID, which can be found in the Flutter project build.gradle file. For this Flutter project, the application ID is: "com.example.fyp_v1".


Add Firebase to your Android app

1 Register app

Android package name: com.example.fyp_v1

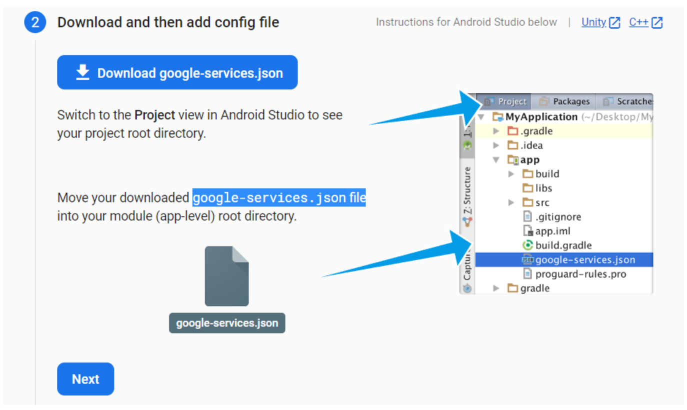
App nickname (optional): My Android App

Debug signing certificate SHA-1 (optional): 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0

Required for Dynamic Links, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings.

Figure 3.3.2.1 Register app

Next, follow the instructions by downloading the google-services.json file and moving it into the project app directory, as shown in Figure 3.3.2.2. The google-services.json file is essential as it contains credentials and configuration settings, which helps auto-initialize the Firebase on app startup. Figure 3.3.2.3 shows the contents inside the file, including project info, client info, api key and services.

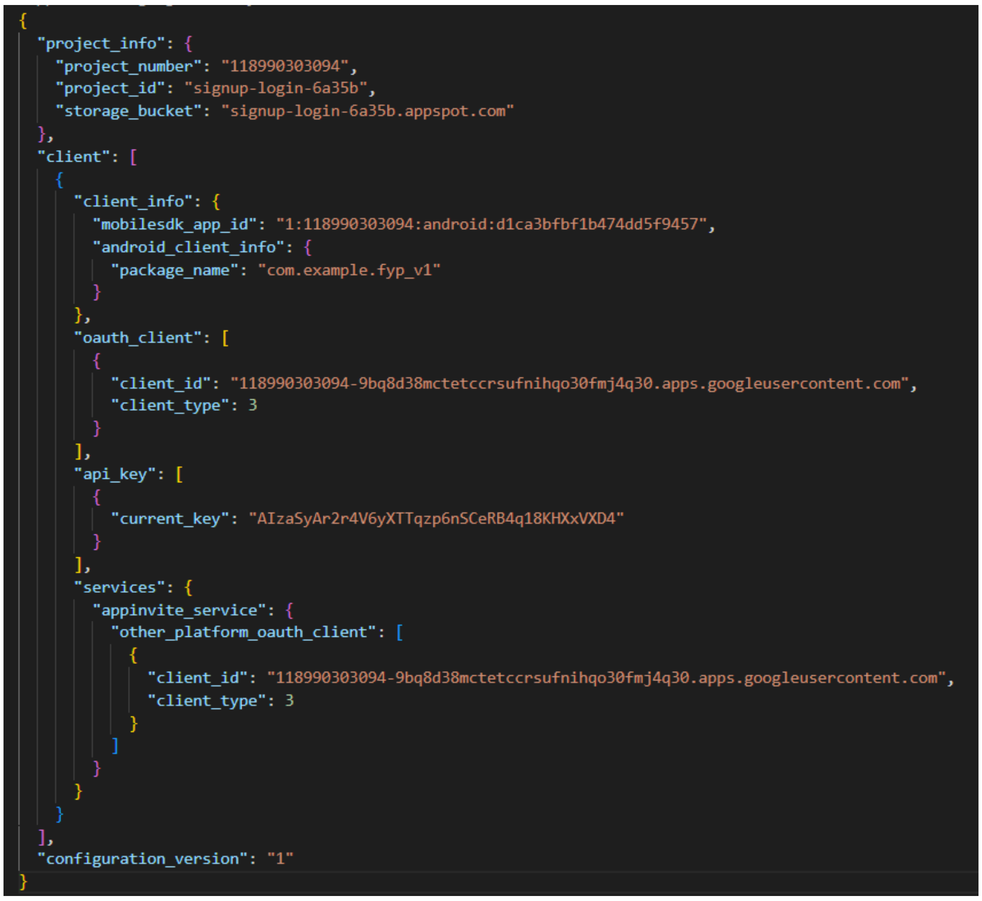

2 Download and then add config file

Instructions for Android Studio below | Unity C++

Switch to the **Project** view in Android Studio to see your project root directory.

Move your downloaded google-services.json file into your module (app-level) root directory.

Figure 3.3.2.2 Instructions to download json file



```json
{
  "project_info": {
    "project_number": "118990303094",
    "project_id": "signup-login-6a35b",
    "storage_bucket": "signup-login-6a35b.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:118990303094:android:d1ca3bfbf1b474dd5f9457",
        "android_client_info": {
          "package_name": "com.example.fyp_v1"
        }
      },
      "oauth_client": [
        {
          "client_id": "118990303094-9bq8d38mctetccrsufnihqo30fmj4q30.apps.googleusercontent.com",
          "client_type": 3
        }
      ],
      "api_key": [
        {
          "current_key": "AIzaSyAr2r4V6yXTTqzp6nSCeRB4q18KHXxVXD4"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": [
            {
              "client_id": "118990303094-9bq8d38mctetccrsufnihqo30fmj4q30.apps.googleusercontent.com",
              "client_type": 3
            }
          ]
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

Figure 3.3.2.3 Contents of json file

Then, add the classpath under the dependencies in the project-level build.gradle file and apply relevant plugins and dependencies in the app-level build.gradle file. Once finished, continue to the console. The connection between the Flutter project and Firebase will be established, as shown in Figure 3.3.2.4.

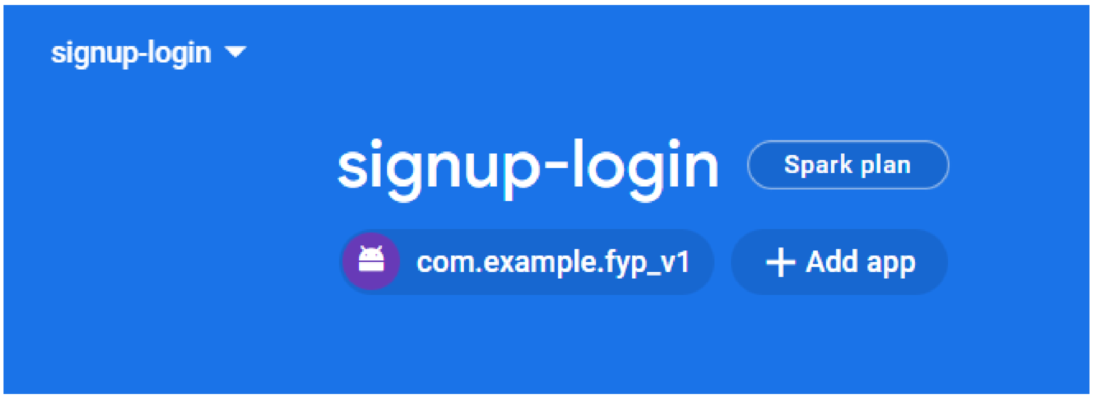
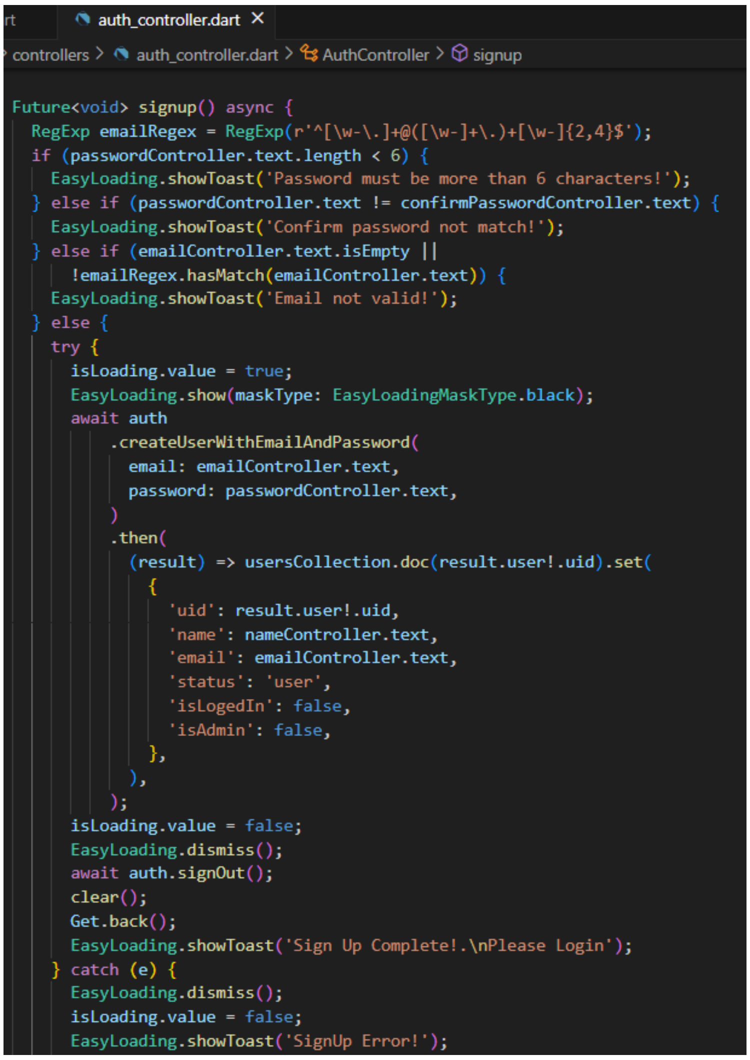


Figure 3.3.2.4 Flutter project successfully connected to Firebase

3.3.3 Signup and Login Function Development (Admin / Student)

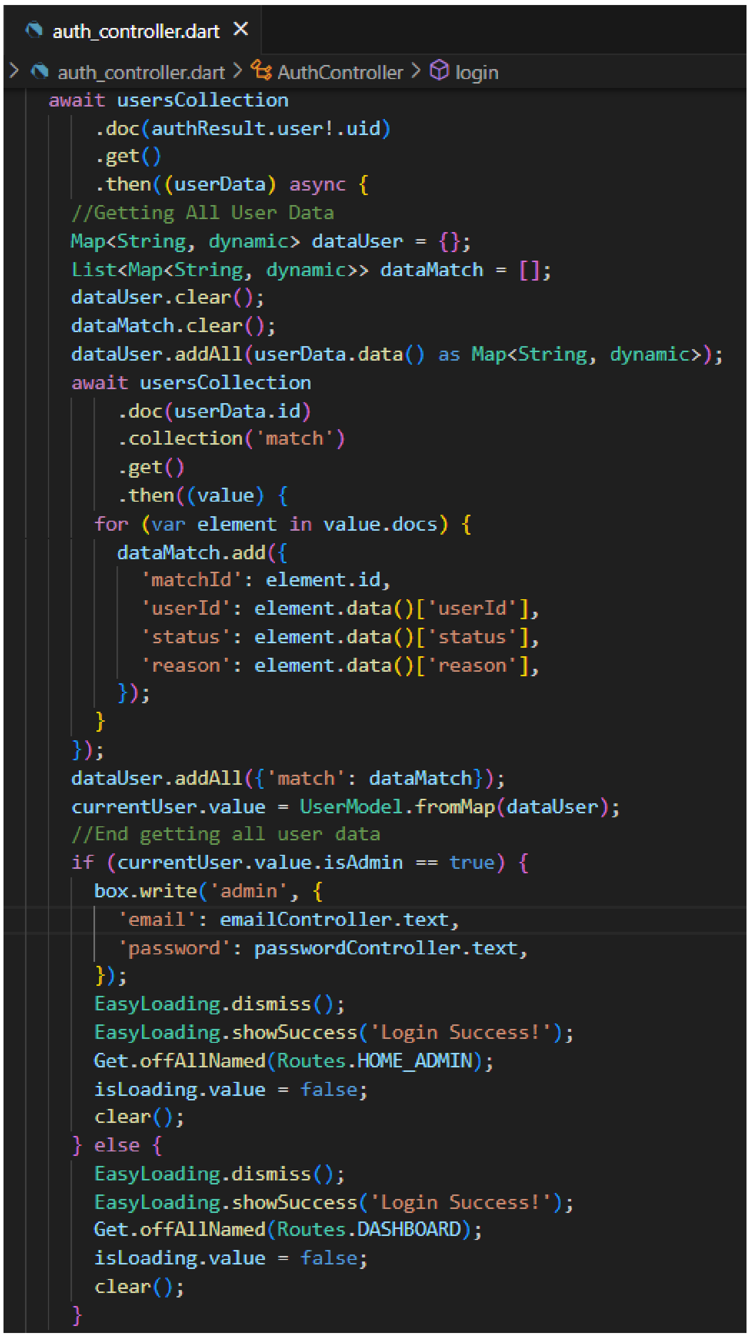
For the signup function, there will be 4 text editing controllers: emailController, passwordController, confirmPasswordController, and nameController to retrieve and manipulate user input. Users must first enter their full name, email, and confirm password. Figure 3.3.3.1 shows the code that handles the validation for signup function. There are a series of conditions in order for the user to successfully sign up. The code will validate whether the email address entered is in the correct format and check whether the length of the password entered is more than 6 characters. The code also checks whether the password and confirm password match and whether the email field is empty. If any of these conditions are not met, an error message will be sent to alert the users. Once all conditions are fulfilled, the app will create a new user account with the details entered by the user and show a success message to indicate to users that the signup was successful. If there is an error in the signup process, the app will show a failure message to indicate that the signup has failed.


auth_controller.dart
controllers > auth_controller.dart > AuthController > signup

```dart
Future<void> signup() async {
  RegExp emailRegex = RegExp(r'^[\w-\.]+@([\w-]+\.)+[\w-]{2,4}$');
  if (passwordController.text.length < 6) {
    EasyLoading.showToast('Password must be more than 6 characters!');
  } else if (passwordController.text != confirmPasswordController.text) {
    EasyLoading.showToast('Confirm password not match!');
  } else if (emailController.text.isEmpty ||
      !emailRegex.hasMatch(emailController.text)) {
    EasyLoading.showToast('Email not valid!');
  } else {
    try {
      isLoading.value = true;
      EasyLoading.show(maskType: EasyLoadingMaskType.black);
      await auth
          .createUserWithEmailAndPassword(
            email: emailController.text,
            password: passwordController.text,
          )
          .then(
            (result) => usersCollection.doc(result.user!.uid).set(
              {
                'uid': result.user!.uid,
                'name': nameController.text,
                'email': emailController.text,
                'status': 'user',
                'isLogedIn': false,
                'isAdmin': false,
              },
            ),
          );
      isLoading.value = false;
      EasyLoading.dismiss();
      await auth.signOut();
      clear();
      Get.back();
      EasyLoading.showToast('Sign Up Complete!.\nPlease Login');
    } catch (e) {
      EasyLoading.dismiss();
      isLoading.value = false;
      EasyLoading.showToast('SignUp Error!');
```

Figure 3.3.3.1 auth_controller.dart: Sign Up function

Moving on to the login function, once user entered their details and successfully authenticated, the app will fetch their data stored in the Firestore as shown in Figure 3.3.3.2. Depending on the user type, the app will route the user to different dashboards. Administrator will be routed to the admin panel, whereas student will be routed to the student dashboard panel.

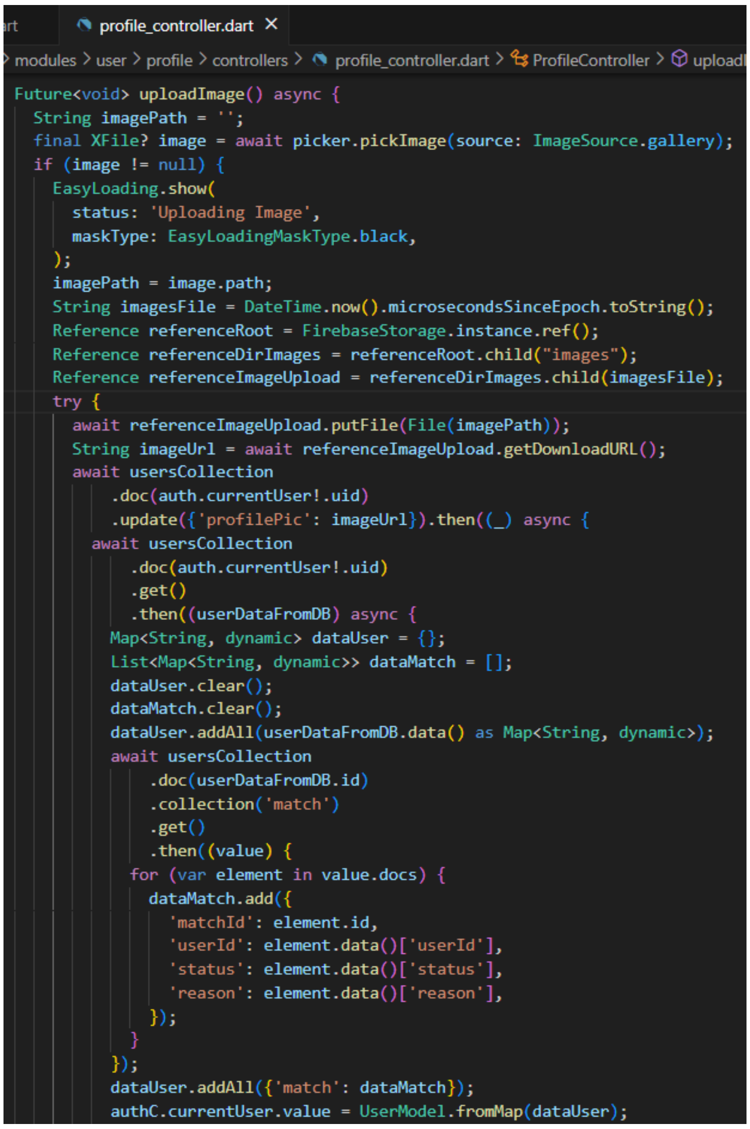


```
await usersCollection
    .doc(authResult.user!.uid)
    .get()
    .then((userData) async {
  //Getting All User Data
  Map<String, dynamic> dataUser = {};
  List<Map<String, dynamic>> dataMatch = [];
  dataUser.clear();
  dataMatch.clear();
  dataUser.addAll(userData.data() as Map<String, dynamic>);
  await usersCollection
      .doc(userData.id)
      .collection('match')
      .get()
      .then((value) {
    for (var element in value.docs) {
      dataMatch.add({
        'matchId': element.id,
        'userId': element.data()['userId'],
        'status': element.data()['status'],
        'reason': element.data()['reason'],
      });
    }
  });
  dataUser.addAll({'match': dataMatch});
  currentUser.value = UserModel.fromMap(dataUser);
  //End getting all user data
  if (currentUser.value.isAdmin == true) {
    box.write('admin', {
      'email': emailController.text,
      'password': passwordController.text,
    });
    EasyLoading.dismiss();
    EasyLoading.showSuccess('Login Success!');
    Get.offAllNamed(Routes.HOME_ADMIN);
    isLoading.value = false;
    clear();
  } else {
    EasyLoading.dismiss();
    EasyLoading.showSuccess('Login Success!');
    Get.offAllNamed(Routes.DASHBOARD);
    isLoading.value = false;
    clear();
  }
```

Figure 3.3.3.2 auth_controller.dart: Login function

3.3.4 Upload User Profile Picture Function Development (Student)

Figure 3.4.4.1 shows the uploadImage() function which enables the user to pick an image from their gallery and stores the chosen image by uploading it to Firebase Storage and gets the download URL.

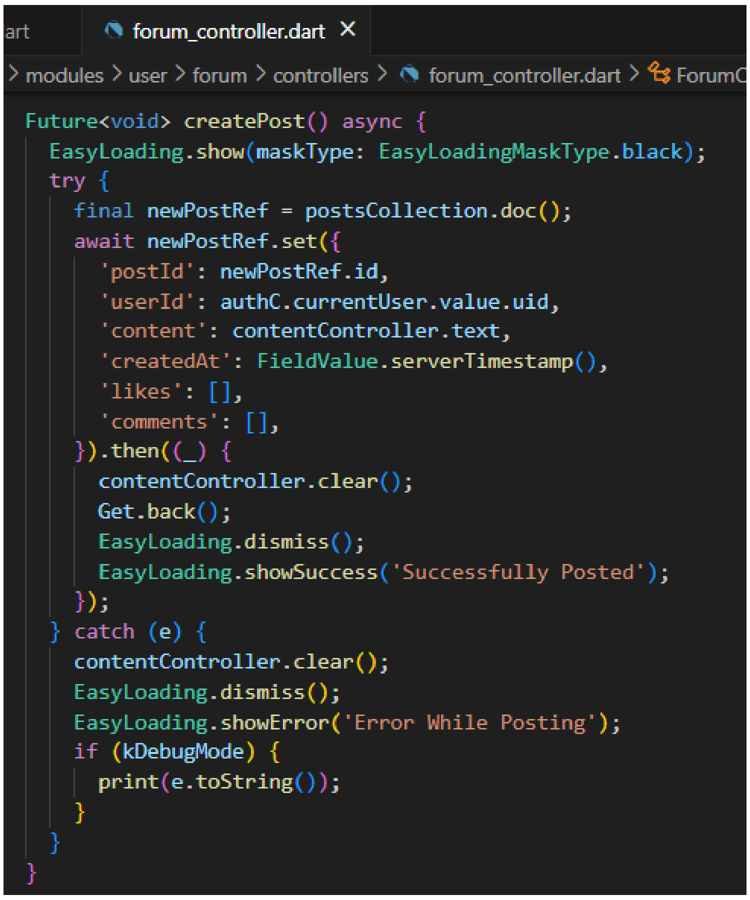

art
profile_controller.dart ×
modules > user > profile > controllers > profile_controller.dart > ProfileController > uploadI

```
Future<void> uploadImage() async {
  String imagePath = '';
  final XFile? image = await picker.pickImage(source: ImageSource.gallery);
  if (image != null) {
    EasyLoading.show(
      status: 'Uploading Image',
      maskType: EasyLoadingMaskType.black,
    );
    imagePath = image.path;
    String imagesFile = DateTime.now().microsecondsSinceEpoch.toString();
    Reference referenceRoot = FirebaseStorage.instance.ref();
    Reference referenceDirImages = referenceRoot.child("images");
    Reference referenceImageUpload = referenceDirImages.child(imagesFile);
    try {
      await referenceImageUpload.putFile(File(imagePath));
      String imageUrl = await referenceImageUpload.getDownloadURL();
      await usersCollection
          .doc(auth.currentUser!.uid)
          .update({'profilePic': imageUrl}).then((_) async {
        await usersCollection
            .doc(auth.currentUser!.uid)
            .get()
            .then((userDataFromDB) async {
          Map<String, dynamic> dataUser = {};
          List<Map<String, dynamic>> dataMatch = [];
          dataUser.clear();
          dataMatch.clear();
          dataUser.addAll(userDataFromDB.data() as Map<String, dynamic>);
          await usersCollection
              .doc(userDataFromDB.id)
              .collection('match')
              .get()
              .then((value) {
            for (var element in value.docs) {
              dataMatch.add({
                'matchId': element.id,
                'userId': element.data()['userId'],
                'status': element.data()['status'],
                'reason': element.data()['reason'],
              });
            }
          });
          dataUser.addAll({'match': dataMatch});
          authC.currentUser.value = UserModel.fromMap(dataUser);
```

Figure 3.3.4.1 profile_controller.dart : Upload Profile Picture function

3.3.5 Add Forum Post Function Development (Student)

Figure 3.3.5.1 shows the createPost() function whereby after the user clicks on the "Submit" button in the dialog box, the user triggers the function to add new forum posts anonymously. The FieldValue.serverTimestamp() is used to reflect the exact time that a post is created.

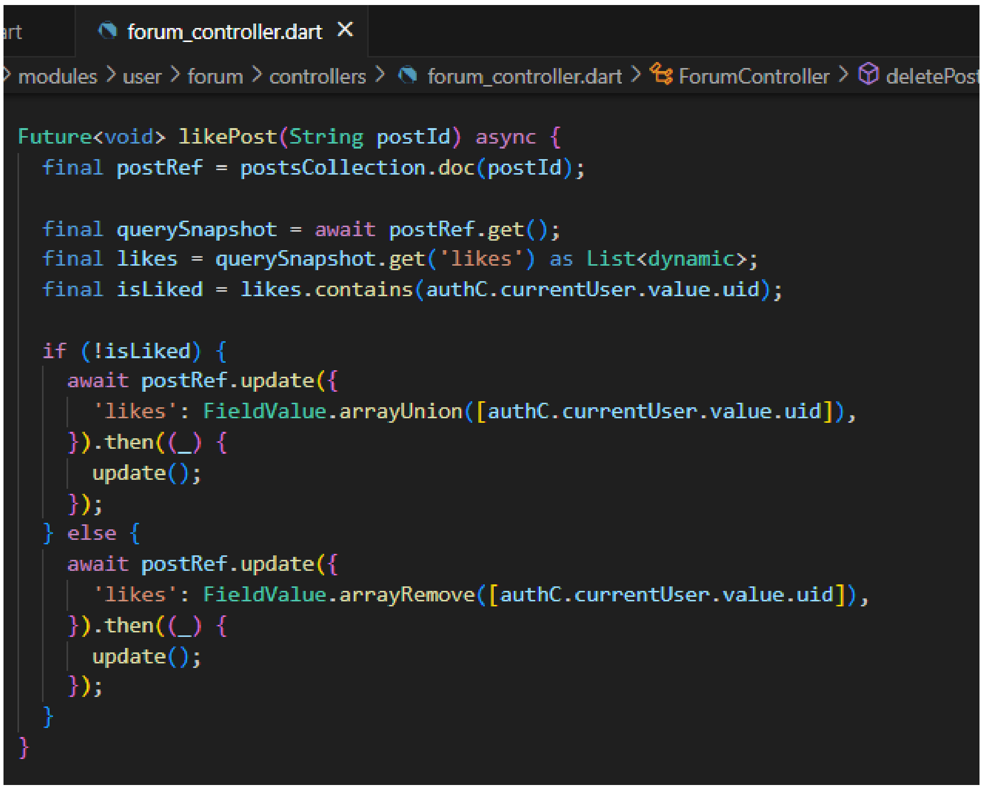

art
forum_controller.dart ×
› modules › user › forum › controllers › forum_controller.dart › ForumC

```
Future<void> createPost() async {
  EasyLoading.show(maskType: EasyLoadingMaskType.black);
  try {
    final newPostRef = postsCollection.doc();
    await newPostRef.set({
      'postId': newPostRef.id,
      'userId': authC.currentUser.value.uid,
      'content': contentController.text,
      'createdAt': FieldValue.serverTimestamp(),
      'likes': [],
      'comments': [],
    }).then((_) {
      contentController.clear();
      Get.back();
      EasyLoading.dismiss();
      EasyLoading.showSuccess('Successfully Posted');
    });
  } catch (e) {
    contentController.clear();
    EasyLoading.dismiss();
    EasyLoading.showError('Error While Posting');
    if (kDebugMode) {
      print(e.toString());
    }
  }
}
```

Figure 3.3.5.1 forum_controller.dart : Add Forum Post function

3.3.6 Like Forum Post Function Development (Student)

Figure 3.3.6.1 shows the likePost() function whereby after the user clicks on the "♥" button of another user's post, the like count will be updated. FieldValue arrayUnion is used to add a new user to the likes array of a forum post and increase the like count +1, whereas FieldValue arrayRemove is used to remove the current user from the array and decrease the like count –1.

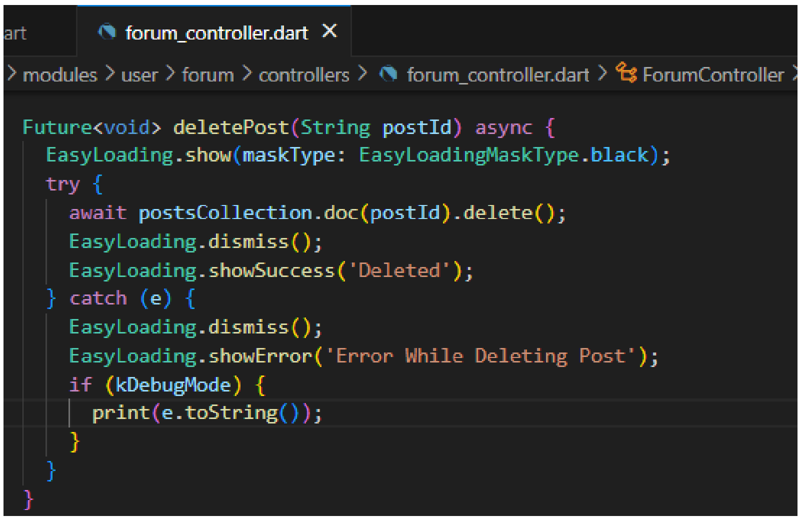

art
forum_controller.dart ×
modules > user > forum > controllers > forum_controller.dart > ForumController > deletePost

```
Future<void> likePost(String postId) async {
  final postRef = postsCollection.doc(postId);

  final querySnapshot = await postRef.get();
  final likes = querySnapshot.get('likes') as List<dynamic>;
  final isLiked = likes.contains(authC.currentUser.value.uid);

  if (!isLiked) {
    await postRef.update({
      'likes': FieldValue.arrayUnion([authC.currentUser.value.uid]),
    }).then((_) {
      update();
    });
  } else {
    await postRef.update({
      'likes': FieldValue.arrayRemove([authC.currentUser.value.uid]),
    }).then((_) {
      update();
    });
  }
}
```

Figure 3.3.6.1 forum_controller.dart : Like Forum Post function

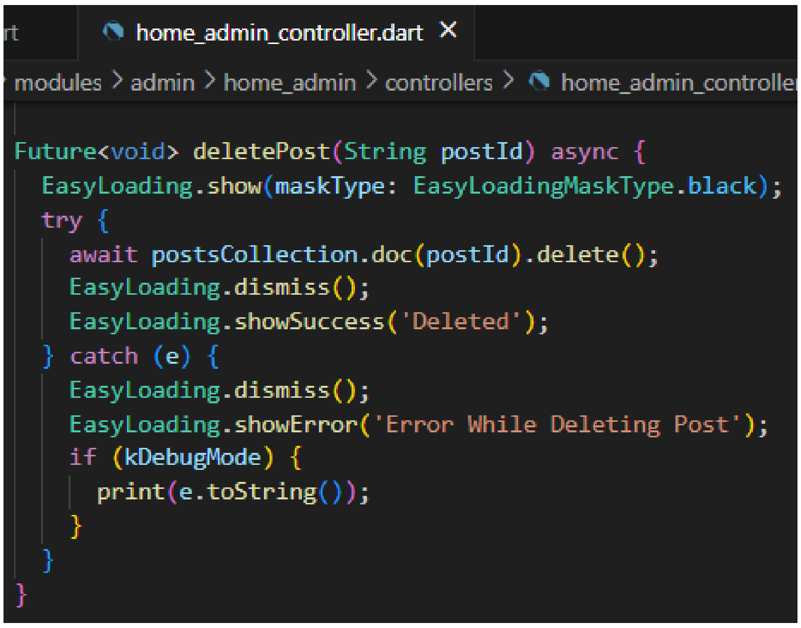
3.3.7 Delete Forum Post Function Development (Admin / Student)

Figure 3.3.7.1 and Figure 3.3.7.2 shows the deletePost() function whereby the user (admin and student) will trigger the function after clicking their post and the "Delete" icon. For user who successfully deleted the post there will be a success message and if there is error in deleting the post, the app will prompt an error message.


art
forum_controller.dart ×
modules > user > forum > controllers > forum_controller.dart > ForumController

```
Future<void> deletePost(String postId) async {
  EasyLoading.show(maskType: EasyLoadingMaskType.black);
  try {
    await postsCollection.doc(postId).delete();
    EasyLoading.dismiss();
    EasyLoading.showSuccess('Deleted');
  } catch (e) {
    EasyLoading.dismiss();
    EasyLoading.showError('Error While Deleting Post');
    if (kDebugMode) {
      print(e.toString());
    }
  }
}
```

Figure 3.3.7.1 forum_controller.dart : Delete Forum Post function (User)

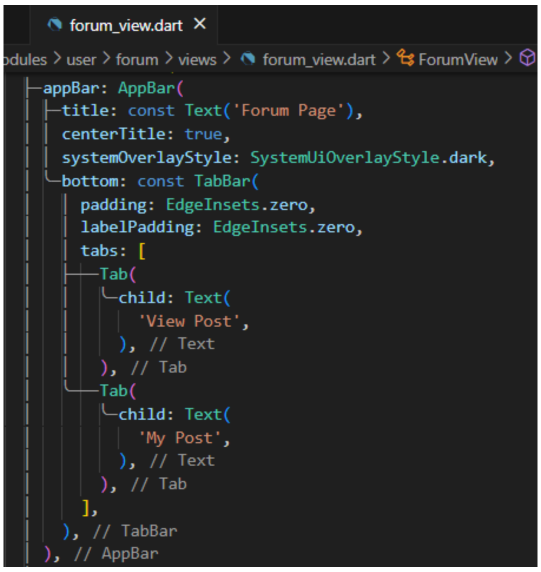

rt
home_admin_controller.dart ×
modules > admin > home_admin > controllers > home_admin_controlle

```
Future<void> deletePost(String postId) async {
  EasyLoading.show(maskType: EasyLoadingMaskType.black);
  try {
    await postsCollection.doc(postId).delete();
    EasyLoading.dismiss();
    EasyLoading.showSuccess('Deleted');
  } catch (e) {
    EasyLoading.dismiss();
    EasyLoading.showError('Error While Deleting Post');
    if (kDebugMode) {
      print(e.toString());
    }
  }
}
```

Figure 3.3.7.2 home_admin_controller.dart : Delete Forum Post function (Admin)

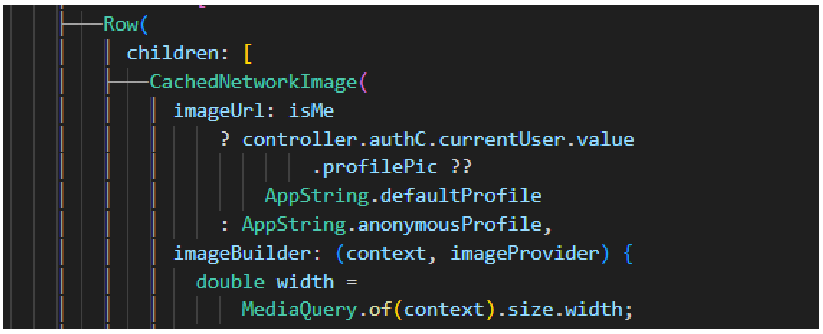
3.3.8 View Forum Post Function Development (Admin / Student)

Figures 3.3.8.1 shows the code for the forum view whereby there are 2 main panels for the forum view, which is **View Post** Panel and **My Post** panel. Under the View Post panel, the app will display posts from other users with an anonymous profile picture (AppString.anonymousProfile) and under the name of Anonymous User as shown in Figure 3.3.8.2 and Figure 3.3.8.3. Besides, they can even view their own post under My Post panel showing their own profile picture (AppString.defaultProfile) and name.



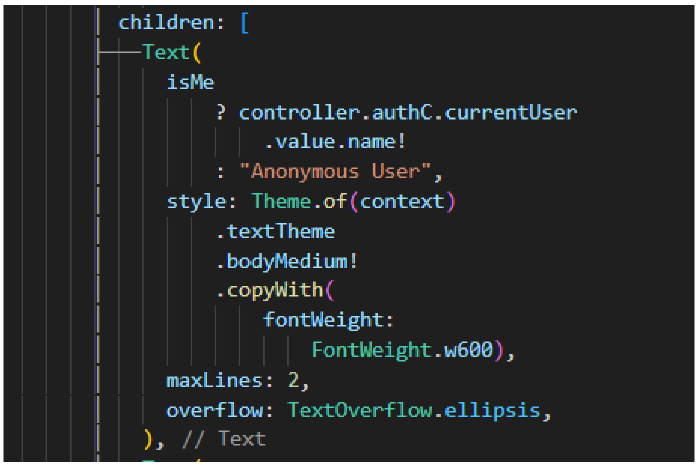
```
forum_view.dart ×
odules > user > forum > views > forum_view.dart > ForumView >
  appBar: AppBar(
    title: const Text('Forum Page'),
    centerTitle: true,
    systemOverlayStyle: SystemUiOverlayStyle.dark,
    bottom: const TabBar(
      padding: EdgeInsets.zero,
      labelPadding: EdgeInsets.zero,
      tabs: [
        Tab(
          child: Text(
            'View Post',
          ), // Text
        ), // Tab
        Tab(
          child: Text(
            'My Post',
          ), // Text
        ), // Tab
      ],
    ), // TabBar
  ), // AppBar
```

Figure 3.3.8.1 forum_view.dart : View Forum Post (User) pt1



```
Row(
  children: [
    CachedNetworkImage(
      imageUrl: isMe
          ? controller.authC.currentUser.value
                  .profilePic ??
              AppString.defaultProfile
          : AppString.anonymousProfile,
      imageBuilder: (context, imageProvider) {
        double width =
            MediaQuery.of(context).size.width;
```

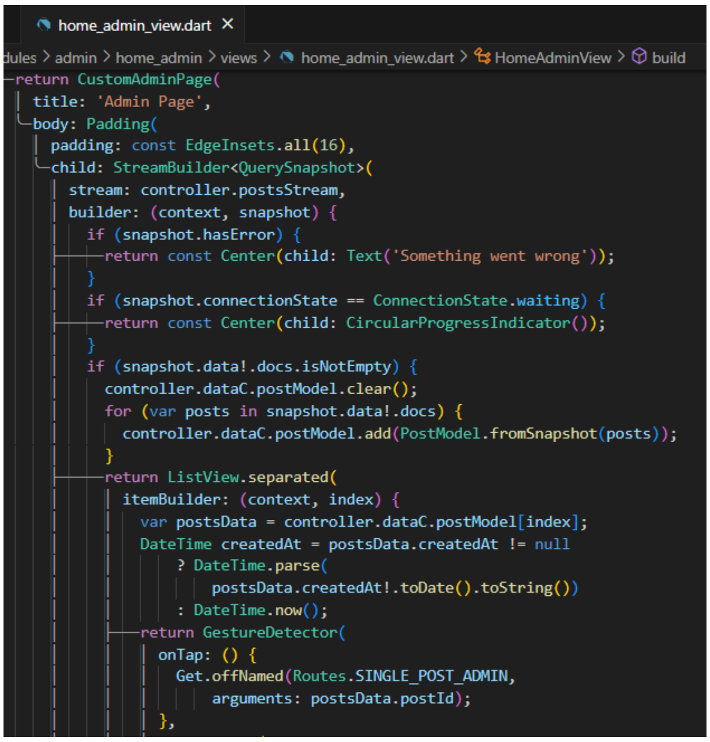
Figure 3.3.8.2 forum_view.dart : View Forum Post (User) pt2



```dart
children: [
  Text(
    isMe
        ? controller.authC.currentUser
            .value.name!
        : "Anonymous User",
    style: Theme.of(context)
        .textTheme
        .bodyMedium!
        .copyWith(
            fontWeight:
                FontWeight.w600),
    maxLines: 2,
    overflow: TextOverflow.ellipsis,
  ), // Text
```

Figure 3.3.8.3 forum_view.dart : View Forum Post (User) pt3

Figure 3.3.8.4 shows the code to display forum post for admin view. If the docs collection is not empty, the admin will view a list of posts on the Admin Page which is also known as the Home Page. If the admin clicks on a particular user, the app will detect the gesture and route admin to SINGLE_POST_ADMIN page where it will display the contents of a particular post.

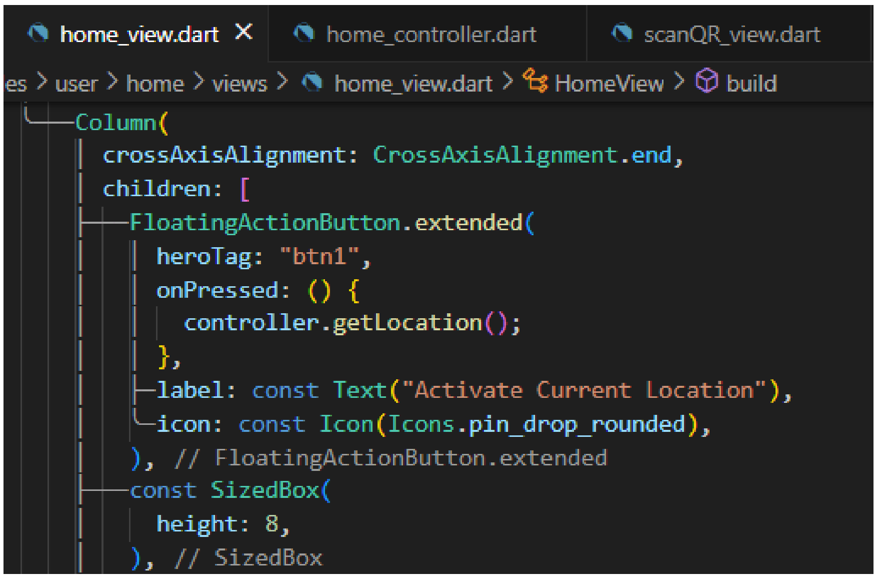


```dart
return CustomAdminPage(
  title: 'Admin Page',
  body: Padding(
    padding: const EdgeInsets.all(16),
    child: StreamBuilder<QuerySnapshot>(
      stream: controller.postsStream,
      builder: (context, snapshot) {
        if (snapshot.hasError) {
          return const Center(child: Text('Something went wrong'));
        }
        if (snapshot.connectionState == ConnectionState.waiting) {
          return const Center(child: CircularProgressIndicator());
        }
        if (snapshot.data!.docs.isNotEmpty) {
          controller.dataC.postModel.clear();
          for (var posts in snapshot.data!.docs) {
            controller.dataC.postModel.add(PostModel.fromSnapshot(posts));
          }
          return ListView.separated(
            itemBuilder: (context, index) {
              var postsData = controller.dataC.postModel[index];
              DateTime createdAt = postsData.createdAt != null
                  ? DateTime.parse(
                      postsData.createdAt!.toDate().toString())
                  : DateTime.now();
              return GestureDetector(
                onTap: () {
                  Get.offNamed(Routes.SINGLE_POST_ADMIN,
                      arguments: postsData.postId);
                },
```

Figure 3.3.8.4 home_admin_view.dart : View Forum Post (Admin)

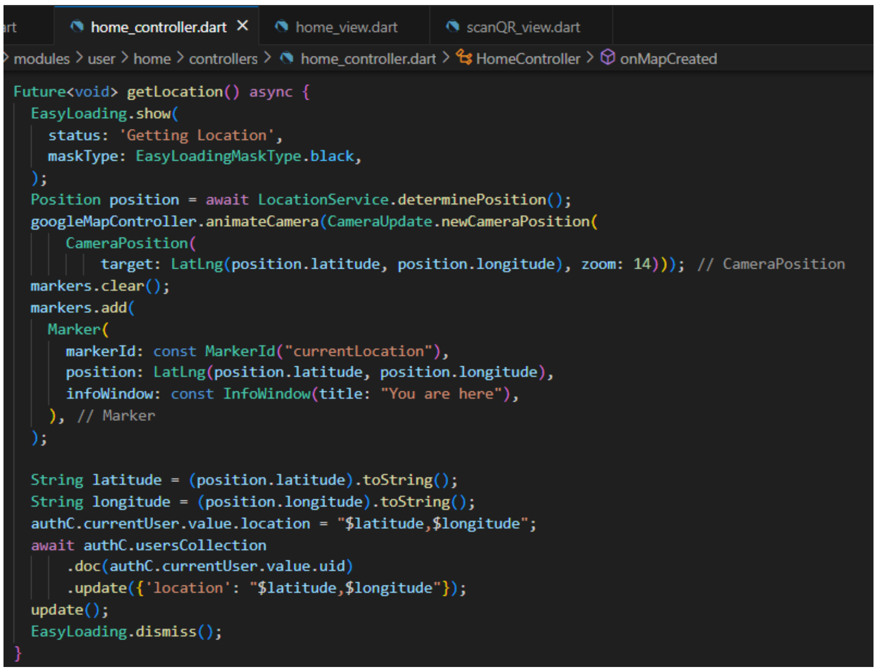
3.3.9 Share Location Function Development (Student)

Figure 3.3.9.1 shows the home page view for students. For students to use the share location function, students must click on the "Activate Current Location" button to trigger the getLocation(). Figure 3.3.9.2 and Figure 3.3.9.3 show that the app will try to retrieve the user's current GPS location using the Geolocator package. The app will first validate whether GPS service is allowed in the current device. If it is not permitted, the app will request permission from the user to enable GPS service. Depending on the user's actions, the app will react accordingly.


home_view.dart
home_controller.dart
scanQR_view.dart
es > user > home > views > home_view.dart > HomeView > build

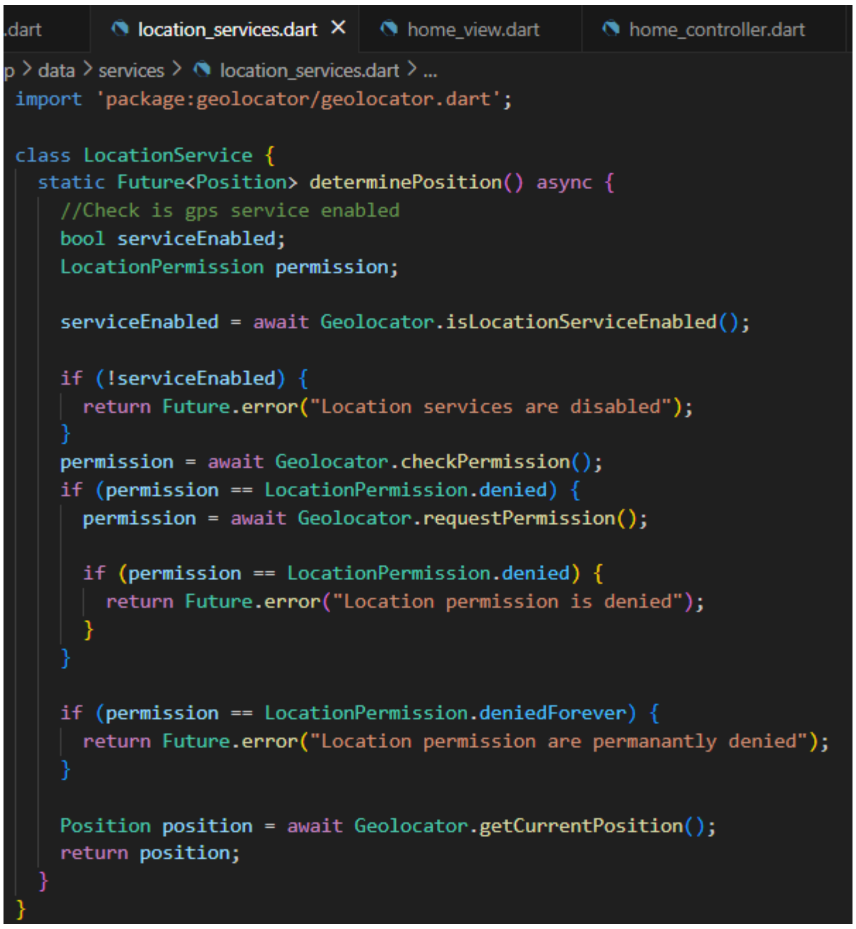
```
Column(
  crossAxisAlignment: CrossAxisAlignment.end,
  children: [
    FloatingActionButton.extended(
      heroTag: "btn1",
      onPressed: () {
        controller.getLocation();
      },
      label: const Text("Activate Current Location"),
      icon: const Icon(Icons.pin_drop_rounded),
    ), // FloatingActionButton.extended
    const SizedBox(
      height: 8,
    ), // SizedBox
```

Figure 3.3.9.1 home_view.dart : Home Page View (Google Maps)


home_controller.dart
home_view.dart
scanQR_view.dart
modules > user > home > controllers > home_controller.dart > HomeController > onMapCreated

```
Future<void> getLocation() async {
  EasyLoading.show(
    status: 'Getting Location',
    maskType: EasyLoadingMaskType.black,
  );
  Position position = await LocationService.determinePosition();
  googleMapController.animateCamera(CameraUpdate.newCameraPosition(
      CameraPosition(
          target: LatLng(position.latitude, position.longitude), zoom: 14))); // CameraPosition
  markers.clear();
  markers.add(
    Marker(
      markerId: const MarkerId("currentLocation"),
      position: LatLng(position.latitude, position.longitude),
      infoWindow: const InfoWindow(title: "You are here"),
    ), // Marker
  );

  String latitude = (position.latitude).toString();
  String longitude = (position.longitude).toString();
  authC.currentUser.value.location = "$latitude,$longitude";
  await authC.usersCollection
      .doc(authC.currentUser.value.uid)
      .update({'location': "$latitude,$longitude"});
  update();
  EasyLoading.dismiss();
}
```

Figure 3.3.9.2 home_controller.dart : Get Location Function

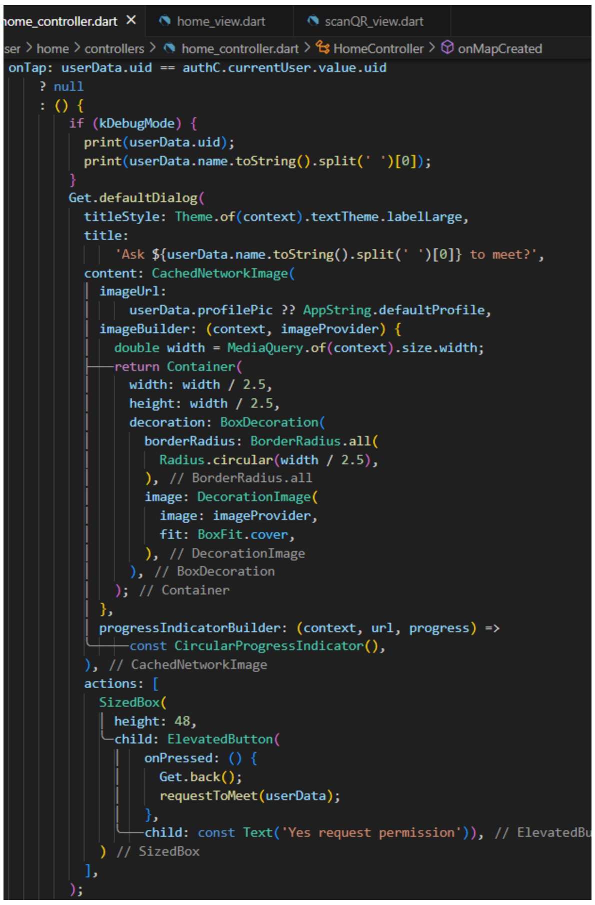

.dart   location_services.dart ×   home_view.dart   home_controller.dart

p > data > services > location_services.dart > ...

```dart
import 'package:geolocator/geolocator.dart';

class LocationService {
  static Future<Position> determinePosition() async {
    //Check is gps service enabled
    bool serviceEnabled;
    LocationPermission permission;

    serviceEnabled = await Geolocator.isLocationServiceEnabled();

    if (!serviceEnabled) {
      return Future.error("Location services are disabled");
    }
    permission = await Geolocator.checkPermission();
    if (permission == LocationPermission.denied) {
      permission = await Geolocator.requestPermission();

      if (permission == LocationPermission.denied) {
        return Future.error("Location permission is denied");
      }
    }

    if (permission == LocationPermission.deniedForever) {
      return Future.error("Location permission are permanantly denied");
    }

    Position position = await Geolocator.getCurrentPosition();
    return position;
  }
}
```

Figure 3.3.9.3 location_services.dart : Geolocator package

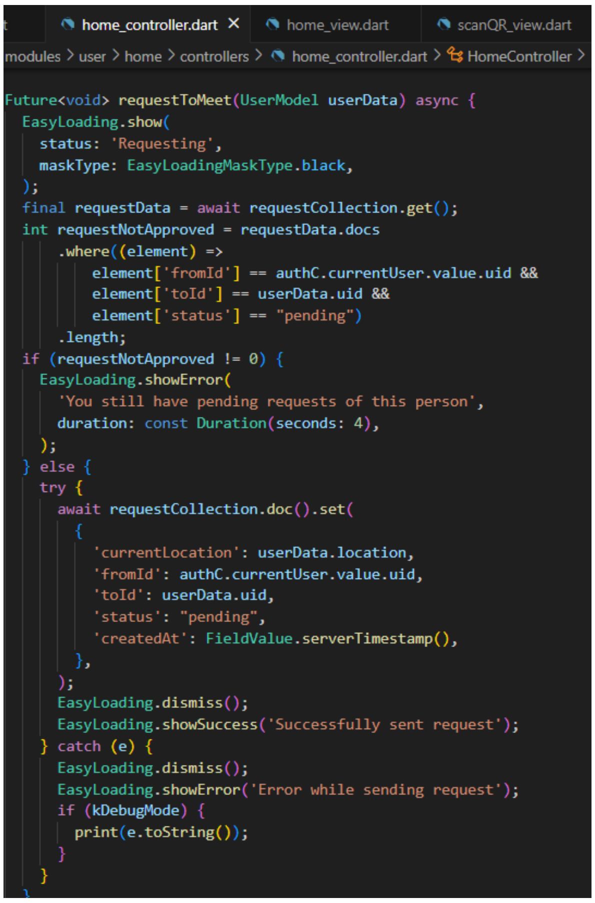
Once the app successfully gets the device's location, the app will display a marker on the map to indicate the current user's location. Around the user, there will be other markers to indicate other user's location and user can request the location of others by clicking on their marker and "Yes request permission" button, which will trigger the requestToMeet() as shown in Figure 3.3.9.4 and Figure 3.3.9.5. The line if (requestNotApproved != 0) indicates that if user A has sent a request to user B but has not been approved, then the user will receive an error message 'You still have pending requests of this person'. It is to prevent users from spamming multiple requests. Only when the other user accepted / rejected the request then only the user can make a new request.



home_controller.dart
home_view.dart
scanQR_view.dart
ser > home > controllers > home_controller.dart > HomeController > onMapCreated

```dart
onTap: userData.uid == authC.currentUser.value.uid
    ? null
    : () {
        if (kDebugMode) {
          print(userData.uid);
          print(userData.name.toString().split(' ')[0]);
        }
        Get.defaultDialog(
          titleStyle: Theme.of(context).textTheme.labelLarge,
          title:
              'Ask ${userData.name.toString().split(' ')[0]} to meet?',
          content: CachedNetworkImage(
            imageUrl:
                userData.profilePic ?? AppString.defaultProfile,
            imageBuilder: (context, imageProvider) {
              double width = MediaQuery.of(context).size.width;
              return Container(
                width: width / 2.5,
                height: width / 2.5,
                decoration: BoxDecoration(
                  borderRadius: BorderRadius.all(
                    Radius.circular(width / 2.5),
                  ), // BorderRadius.all
                  image: DecorationImage(
                    image: imageProvider,
                    fit: BoxFit.cover,
                  ), // DecorationImage
                ), // BoxDecoration
              ); // Container
            },
            progressIndicatorBuilder: (context, url, progress) =>
                const CircularProgressIndicator(),
          ), // CachedNetworkImage
          actions: [
            SizedBox(
              height: 48,
              child: ElevatedButton(
                  onPressed: () {
                    Get.back();
                    requestToMeet(userData);
                  },
                  child: const Text('Yes request permission')), // ElevatedBu
            ) // SizedBox
          ],
        );
```

Figure 3.3.9.4 home_controller.dart pt1 : Send Request to Meet

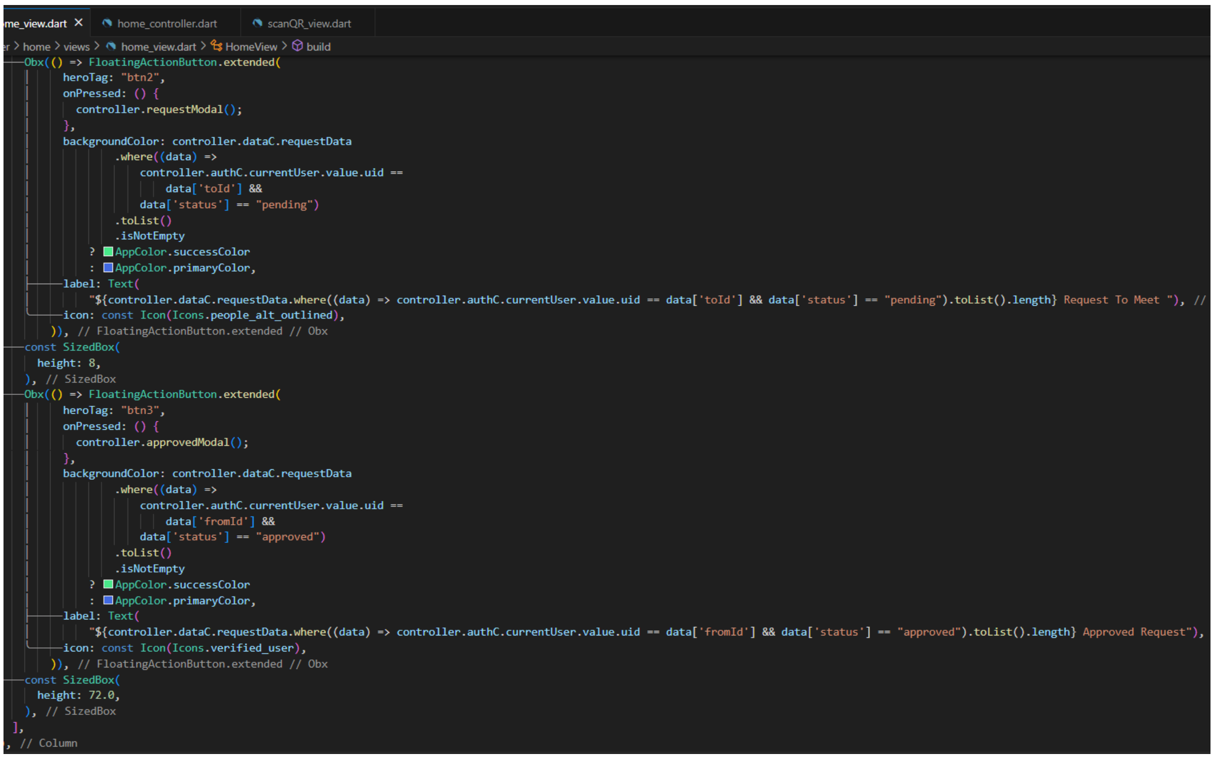


home_controller.dart
home_view.dart
scanQR_view.dart
modules > user > home > controllers > home_controller.dart > HomeController >

```dart
Future<void> requestToMeet(UserModel userData) async {
  EasyLoading.show(
    status: 'Requesting',
    maskType: EasyLoadingMaskType.black,
  );
  final requestData = await requestCollection.get();
  int requestNotApproved = requestData.docs
      .where((element) =>
          element['fromId'] == authC.currentUser.value.uid &&
          element['toId'] == userData.uid &&
          element['status'] == "pending")
      .length;
  if (requestNotApproved != 0) {
    EasyLoading.showError(
      'You still have pending requests of this person',
      duration: const Duration(seconds: 4),
    );
  } else {
    try {
      await requestCollection.doc().set(
        {
          'currentLocation': userData.location,
          'fromId': authC.currentUser.value.uid,
          'toId': userData.uid,
          'status': "pending",
          'createdAt': FieldValue.serverTimestamp(),
        },
      );
      EasyLoading.dismiss();
      EasyLoading.showSuccess('Successfully sent request');
    } catch (e) {
      EasyLoading.dismiss();
      EasyLoading.showError('Error while sending request');
      if (kDebugMode) {
        print(e.toString());
      }
    }
```

Figure 3.3.9.5 home_controller.dart pt2 : Send Request to Meet

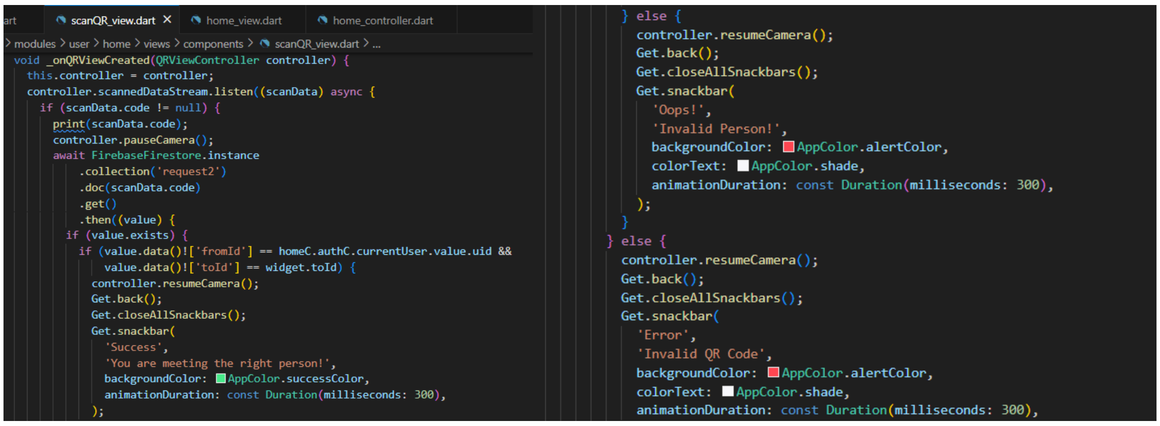
For the user who receives the request, the app will display +1 on the "Request To Meet" button and after they choose to accept the request, the other user's device will display +1 on the "Approved Request" button as shown in Figure 3.3.9.6.



```
me_view.dart ×   home_controller.dart   scanQR_view.dart
r > home > views > home_view.dart > HomeView > build
Obx(() => FloatingActionButton.extended(
      heroTag: "btn2",
      onPressed: () {
        controller.requestModal();
      },
      backgroundColor: controller.dataC.requestData
              .where((data) =>
                  controller.authC.currentUser.value.uid ==
                      data['toId'] &&
                  data['status'] == "pending")
              .toList()
              .isNotEmpty
          ? AppColor.successColor
          : AppColor.primaryColor,
      label: Text(
          "${controller.dataC.requestData.where((data) => controller.authC.currentUser.value.uid == data['toId'] && data['status'] == "pending").toList().length} Request To Meet "), //
      icon: const Icon(Icons.people_alt_outlined),
    )), // FloatingActionButton.extended // Obx
const SizedBox(
  height: 8,
), // SizedBox
Obx(() => FloatingActionButton.extended(
      heroTag: "btn3",
      onPressed: () {
        controller.approvedModal();
      },
      backgroundColor: controller.dataC.requestData
              .where((data) =>
                  controller.authC.currentUser.value.uid ==
                      data['fromId'] &&
                  data['status'] == "approved")
              .toList()
              .isNotEmpty
          ? AppColor.successColor
          : AppColor.primaryColor,
      label: Text(
          "${controller.dataC.requestData.where((data) => controller.authC.currentUser.value.uid == data['fromId'] && data['status'] == "approved").toList().length} Approved Request"),
      icon: const Icon(Icons.verified_user),
    )), // FloatingActionButton.extended // Obx
const SizedBox(
  height: 72.0,
), // SizedBox
 ],
, // Column
```

Figure 3.3.9.6 home_view.dart

To ensure students are meeting up with the right person, the app has a scan QR code function, as shown in Figure 3.3.9.7. If the QR code scanned by the user is valid, a success message will appear, whereas if the QR code scanned is invalid, an error message will appear.

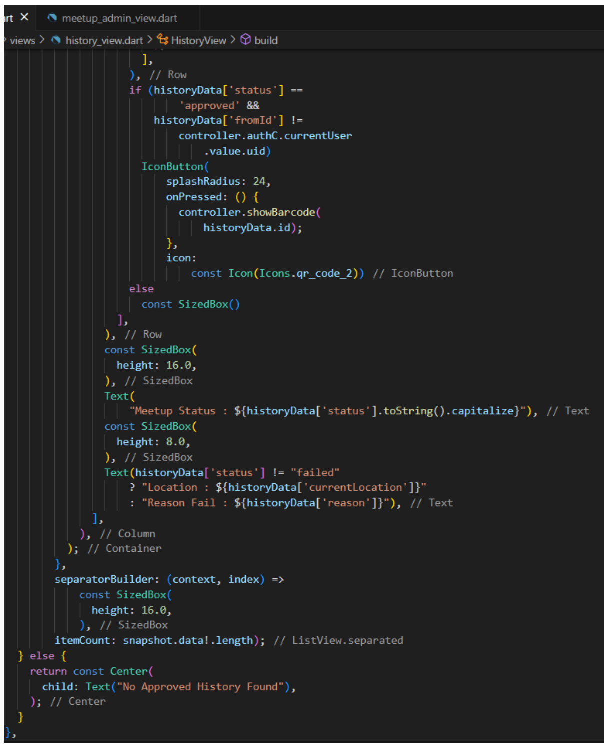


```
art   scanQR_view.dart ×   home_view.dart   home_controller.dart
> modules > user > home > views > components > scanQR_view.dart > ...
void _onQRViewCreated(QRViewController controller) {
  this.controller = controller;
  controller.scannedDataStream.listen((scanData) async {
    if (scanData.code != null) {
      print(scanData.code);
      controller.pauseCamera();
      await FirebaseFirestore.instance
          .collection('request2')
          .doc(scanData.code)
          .get()
          .then((value) {
        if (value.exists) {
          if (value.data()!['fromId'] == homeC.authC.currentUser.value.uid &&
              value.data()!['toId'] == widget.toId) {
            controller.resumeCamera();
            Get.back();
            Get.closeAllSnackbars();
            Get.snackbar(
              'Success',
              'You are meeting the right person!',
              backgroundColor: AppColor.successColor,
              animationDuration: const Duration(milliseconds: 300),
            );
          } else {
            controller.resumeCamera();
            Get.back();
            Get.closeAllSnackbars();
            Get.snackbar(
              'Oops!',
              'Invalid Person!',
              backgroundColor: AppColor.alertColor,
              colorText: AppColor.shade,
              animationDuration: const Duration(milliseconds: 300),
            );
          }
        } else {
          controller.resumeCamera();
          Get.back();
          Get.closeAllSnackbars();
          Get.snackbar(
            'Error',
            'Invalid QR Code',
            backgroundColor: AppColor.alertColor,
            colorText: AppColor.shade,
            animationDuration: const Duration(milliseconds: 300),
```

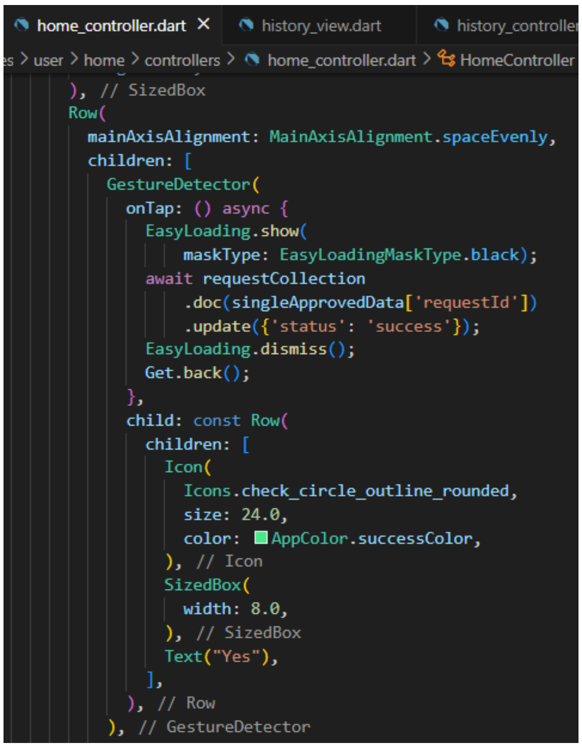
Figure 3.3.9.7 scanQR_view.dart : Scan QR Code Function

3.3.10 View Meetup History Function Development (Admin / Student)

Both admin and student can view the meetup history in their respective panels. The meetup history has 3 pages which are Approved, Pending, Rejected. Figure 3.3.10.1 and Figure 3.3.10.2 shows the approved meetup history view for users where the meetup status will only be 'Success' if the user manually clicks on 'Yes' button in the home page. If user clicks on 'No' then regardless the request is approved, the meetup status will still be 'Failed'. The 'Success' meetup will include the location coordinates of the meetup whereas the 'Failed' meetup will include the reason of meetup failure. If there are no existing meetup history in the Approved page, then the app will display a line of text 'No Approved History Found'.



```
rt ×   meetup_admin_view.dart
> views > history_view.dart > HistoryView > build
                   ],
                 ), // Row
                 if (historyData['status'] ==
                         'approved' &&
                     historyData['fromId'] !=
                         controller.authC.currentUser
                             .value.uid)
                   IconButton(
                       splashRadius: 24,
                       onPressed: () {
                         controller.showBarcode(
                             historyData.id);
                       },
                       icon:
                           const Icon(Icons.qr_code_2)) // IconButton
                 else
                   const SizedBox()
               ],
             ), // Row
             const SizedBox(
               height: 16.0,
             ), // SizedBox
             Text(
                 "Meetup Status : ${historyData['status'].toString().capitalize}"), // Text
             const SizedBox(
               height: 8.0,
             ), // SizedBox
             Text(historyData['status'] != "failed"
                 ? "Location : ${historyData['currentLocation']}"
                 : "Reason Fail : ${historyData['reason']}"), // Text
           ],
         ), // Column
       ); // Container
     },
     separatorBuilder: (context, index) =>
         const SizedBox(
           height: 16.0,
         ), // SizedBox
     itemCount: snapshot.data!.length); // ListView.separated
 } else {
   return const Center(
     child: Text("No Approved History Found"),
   ); // Center
 }
},
```

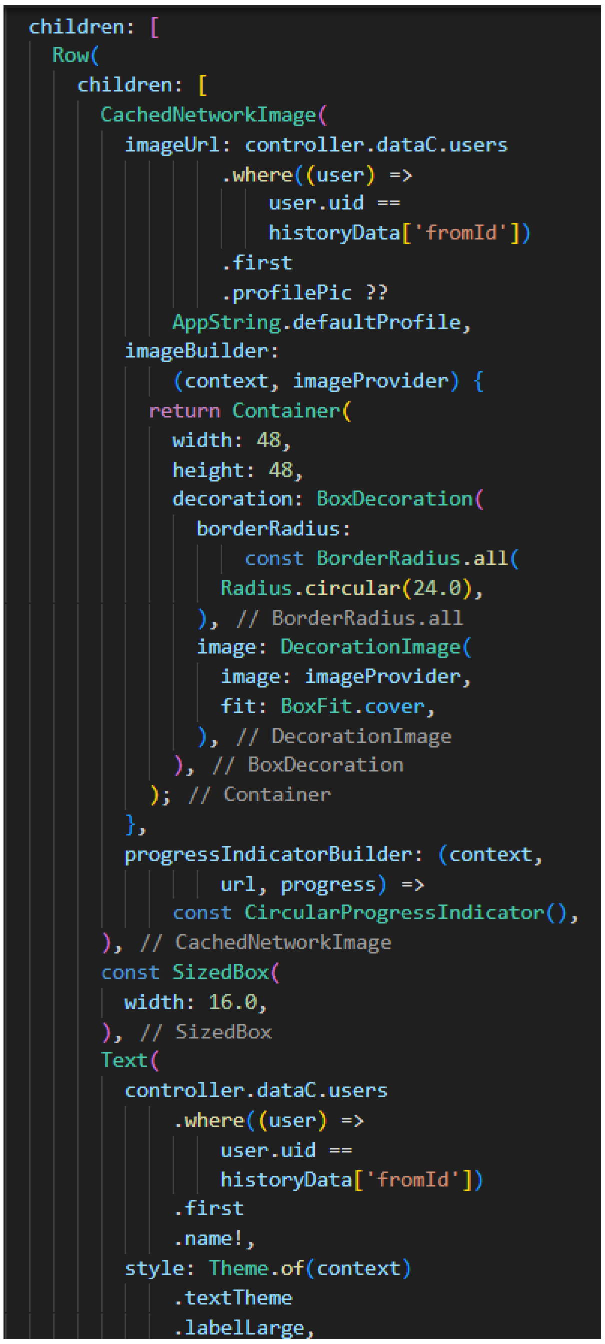


```
home_controller.dart ×   history_view.dart   history_controlle
s > user > home > controllers > home_controller.dart > HomeController
), // SizedBox
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    GestureDetector(
      onTap: () async {
        EasyLoading.show(
            maskType: EasyLoadingMaskType.black);
        await requestCollection
            .doc(singleApprovedData['requestId'])
            .update({'status': 'success'});
        EasyLoading.dismiss();
        Get.back();
      },
      child: const Row(
        children: [
          Icon(
            Icons.check_circle_outline_rounded,
            size: 24.0,
            color: AppColor.successColor,
          ), // Icon
          SizedBox(
            width: 8.0,
          ), // SizedBox
          Text("Yes"),
        ],
      ), // Row
    ), // GestureDetector
```

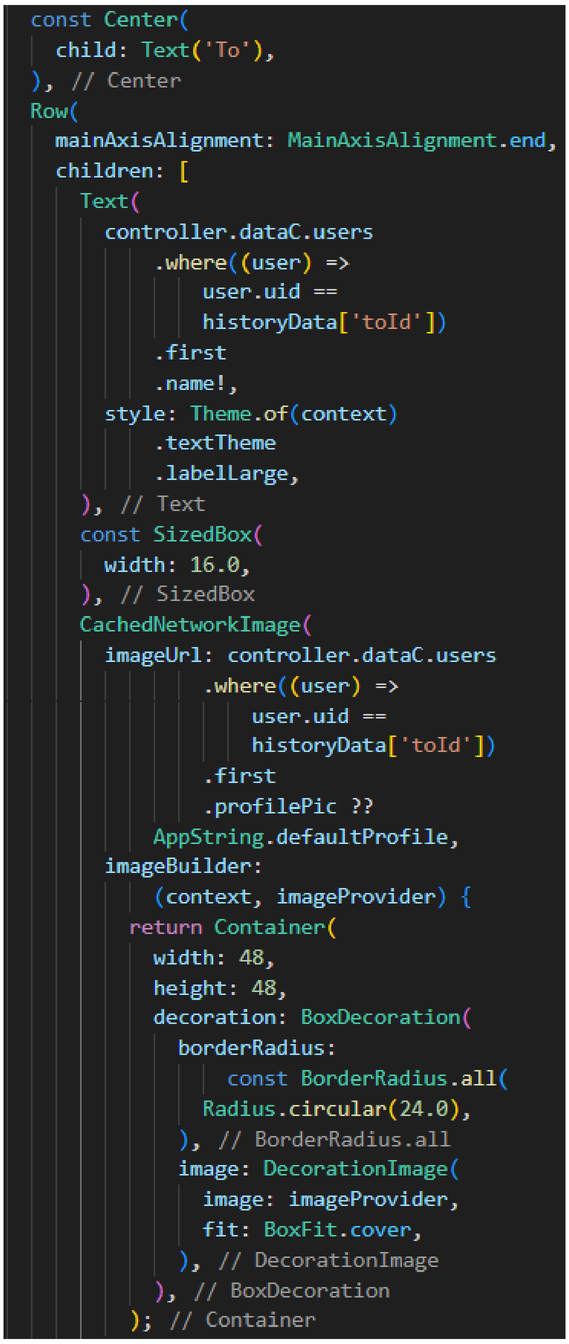
Figure 3.3.10.1 history_view.dart : Approved Meetup History (User)

Figure 3.3.10.2 home_controller.dart

Similarly, the admin can also view the meetup history of users. Figure 3.3.10.3 shows the Approved Meetup History in admin view whereby admin can view the meetup history showing the details of the user who made the request (fromId) to the user who receives the request (toId). The app will display the profile image of both users and their name.



```
children: [
  Row(
    children: [
      CachedNetworkImage(
        imageUrl: controller.dataC.users
                .where((user) =>
                    user.uid ==
                    historyData['fromId'])
                .first
                .profilePic ??
            AppString.defaultProfile,
        imageBuilder:
            (context, imageProvider) {
          return Container(
            width: 48,
            height: 48,
            decoration: BoxDecoration(
              borderRadius:
                  const BorderRadius.all(
                Radius.circular(24.0),
              ), // BorderRadius.all
              image: DecorationImage(
                image: imageProvider,
                fit: BoxFit.cover,
              ), // DecorationImage
            ), // BoxDecoration
          ); // Container
        },
        progressIndicatorBuilder: (context,
                url, progress) =>
            const CircularProgressIndicator(),
      ), // CachedNetworkImage
      const SizedBox(
        width: 16.0,
      ), // SizedBox
      Text(
        controller.dataC.users
            .where((user) =>
                user.uid ==
                historyData['fromId'])
            .first
            .name!,
        style: Theme.of(context)
            .textTheme
            .labelLarge,
```

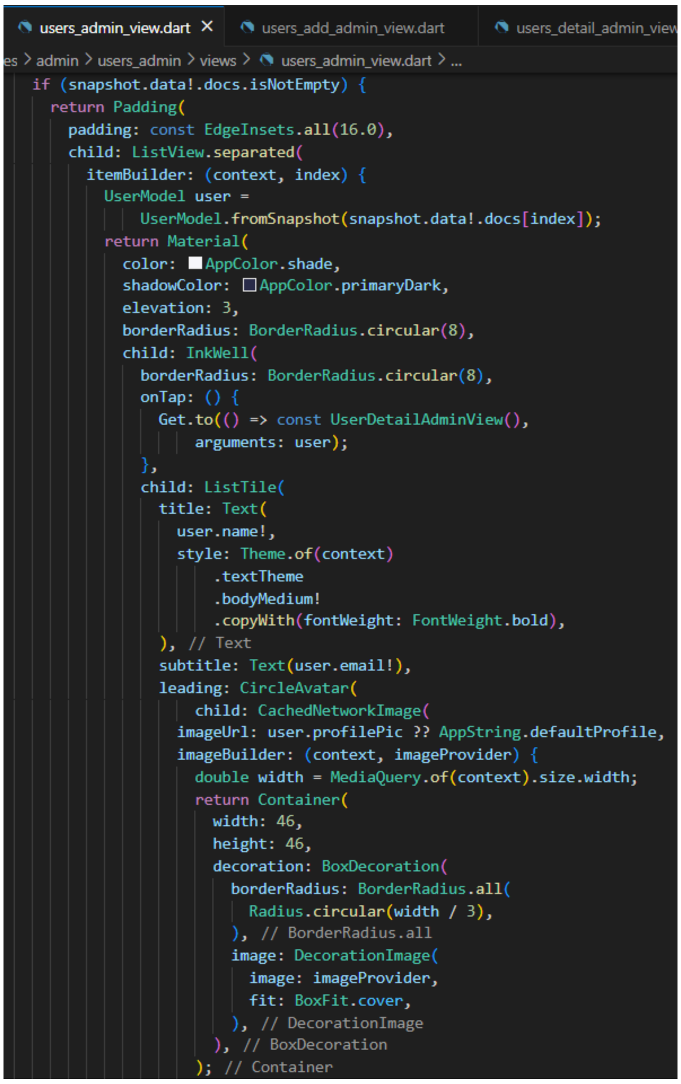


```
const Center(
  child: Text('To'),
), // Center
Row(
  mainAxisAlignment: MainAxisAlignment.end,
  children: [
    Text(
      controller.dataC.users
          .where((user) =>
              user.uid ==
              historyData['toId'])
          .first
          .name!,
      style: Theme.of(context)
          .textTheme
          .labelLarge,
    ), // Text
    const SizedBox(
      width: 16.0,
    ), // SizedBox
    CachedNetworkImage(
      imageUrl: controller.dataC.users
              .where((user) =>
                  user.uid ==
                  historyData['toId'])
              .first
              .profilePic ??
          AppString.defaultProfile,
      imageBuilder:
          (context, imageProvider) {
        return Container(
          width: 48,
          height: 48,
          decoration: BoxDecoration(
            borderRadius:
                const BorderRadius.all(
              Radius.circular(24.0),
            ), // BorderRadius.all
            image: DecorationImage(
              image: imageProvider,
              fit: BoxFit.cover,
            ), // DecorationImage
          ), // BoxDecoration
        ); // Container
```

Figure 3.3.10.3 meetup_admin_view.dart : Approved Meetup History Panel (Admin)

3.3.11 View / Add / Edit User (Admin)

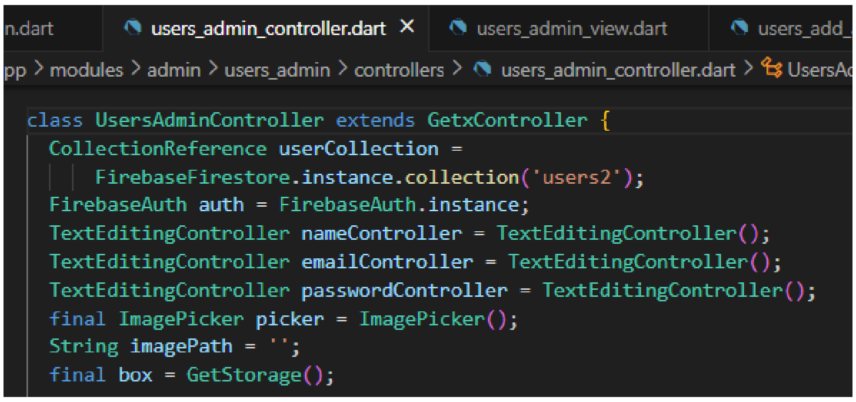
Admin can view the list of existing users. Figure 3.3.11.1 shows that if the Firestore Database is not empty, then the app will read the data available in the Firestore collections named 'users2' then proceed to display the respective username, email and profile picture that is saved in the database during registration process.


users_admin_view.dart
users_add_admin_view.dart
users_detail_admin_view
es > admin > users_admin > views > users_admin_view.dart > ...

```
if (snapshot.data!.docs.isNotEmpty) {
  return Padding(
    padding: const EdgeInsets.all(16.0),
    child: ListView.separated(
      itemBuilder: (context, index) {
        UserModel user =
            UserModel.fromSnapshot(snapshot.data!.docs[index]);
        return Material(
          color: AppColor.shade,
          shadowColor: AppColor.primaryDark,
          elevation: 3,
          borderRadius: BorderRadius.circular(8),
          child: InkWell(
            borderRadius: BorderRadius.circular(8),
            onTap: () {
              Get.to(() => const UserDetailAdminView(),
                  arguments: user);
            },
            child: ListTile(
              title: Text(
                user.name!,
                style: Theme.of(context)
                    .textTheme
                    .bodyMedium!
                    .copyWith(fontWeight: FontWeight.bold),
              ), // Text
              subtitle: Text(user.email!),
              leading: CircleAvatar(
                  child: CachedNetworkImage(
                imageUrl: user.profilePic ?? AppString.defaultProfile,
                imageBuilder: (context, imageProvider) {
                  double width = MediaQuery.of(context).size.width;
                  return Container(
                    width: 46,
                    height: 46,
                    decoration: BoxDecoration(
                      borderRadius: BorderRadius.all(
                        Radius.circular(width / 3),
                      ), // BorderRadius.all
                      image: DecorationImage(
                        image: imageProvider,
                        fit: BoxFit.cover,
                      ), // DecorationImage
                    ), // BoxDecoration
                  ); // Container
```

Figure 3.3.11.1 users_admin_view.dart : View User List

Admin can also create a new user and edit information for an existing user. Figure 3.3.11.2 shows that there are 3 main text editing controller which are nameController, emailController and passwordController which will retrieve the admin's input. In Figure 3.3.11.3, the app will validate whether admin has keyed in all the required details. If admin has not entered all required details, then the app will display error message to alert the admin.

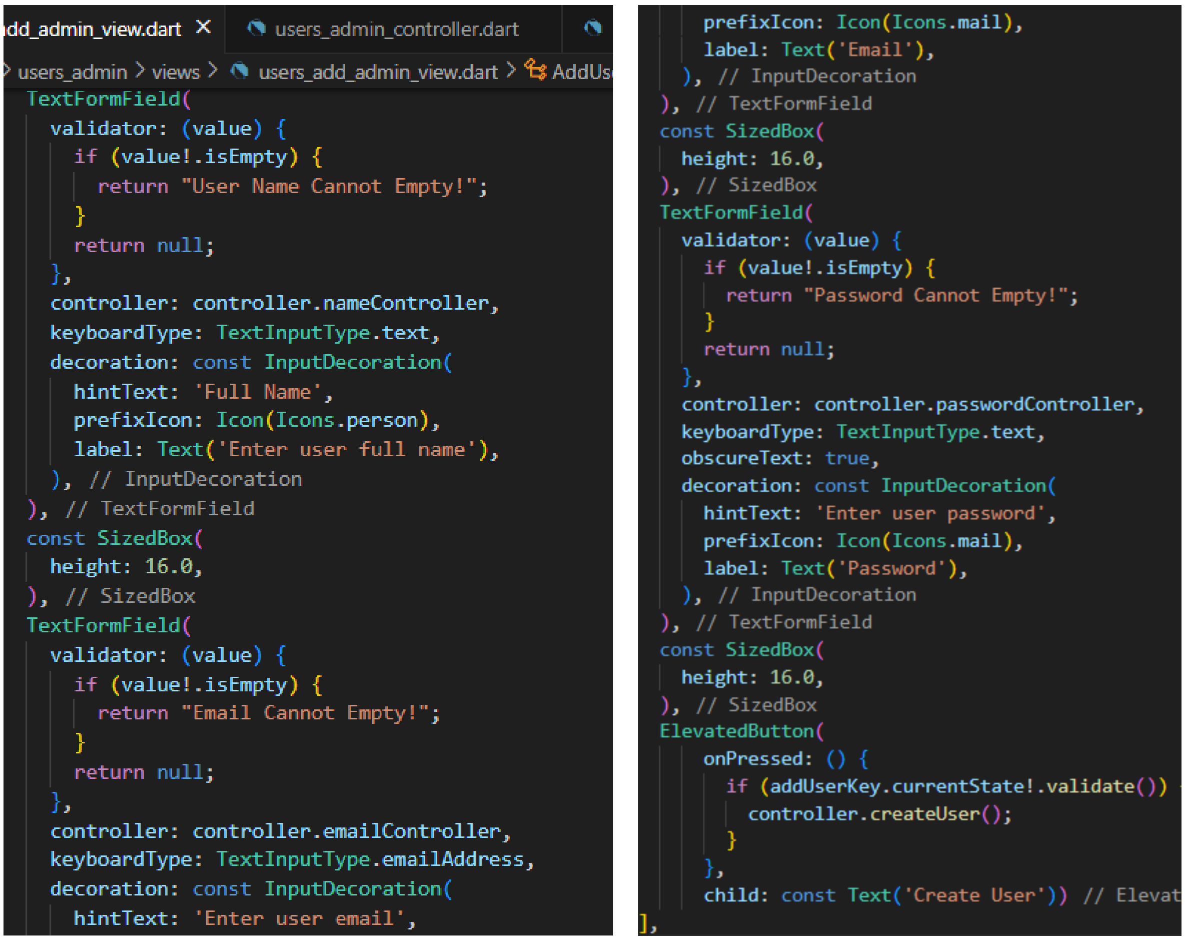


n.dart | users_admin_controller.dart × | users_admin_view.dart | users_add_

pp > modules > admin > users_admin > controllers > users_admin_controller.dart > UsersAc

```
class UsersAdminController extends GetxController {
  CollectionReference userCollection =
      FirebaseFirestore.instance.collection('users2');
  FirebaseAuth auth = FirebaseAuth.instance;
  TextEditingController nameController = TextEditingController();
  TextEditingController emailController = TextEditingController();
  TextEditingController passwordController = TextEditingController();
  final ImagePicker picker = ImagePicker();
  String imagePath = '';
  final box = GetStorage();
```

Figure 3.3.11.2 users_admin_controller.dart



dd_admin_view.dart × | users_admin_controller.dart

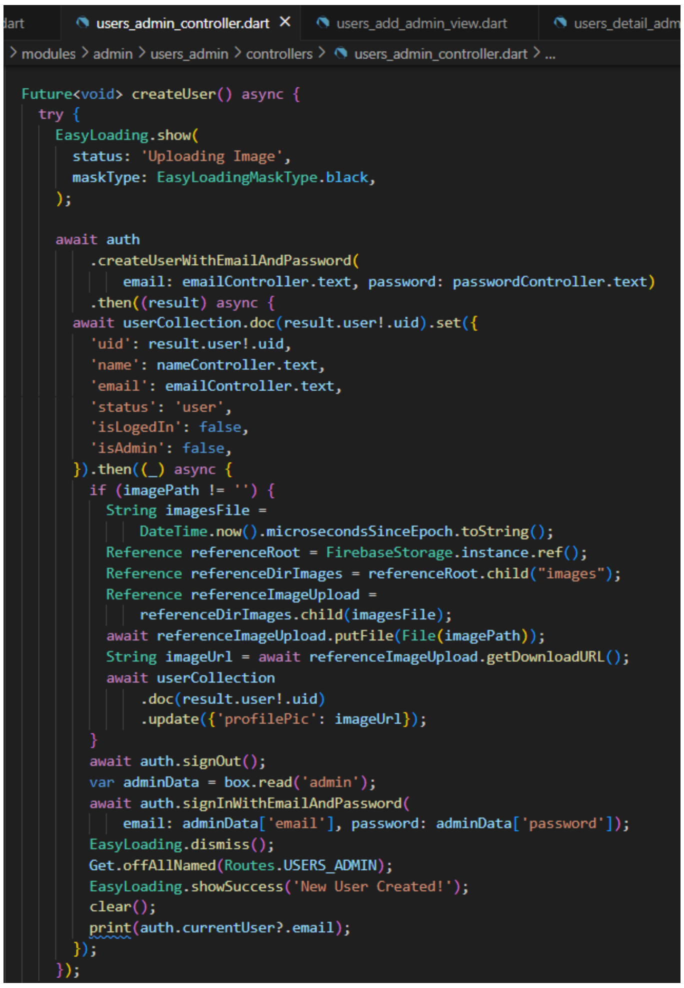
users_admin > views > users_add_admin_view.dart > AddUs

```
TextFormField(
  validator: (value) {
    if (value!.isEmpty) {
      return "User Name Cannot Empty!";
    }
    return null;
  },
  controller: controller.nameController,
  keyboardType: TextInputType.text,
  decoration: const InputDecoration(
    hintText: 'Full Name',
    prefixIcon: Icon(Icons.person),
    label: Text('Enter user full name'),
  ), // InputDecoration
), // TextFormField
const SizedBox(
  height: 16.0,
), // SizedBox
TextFormField(
  validator: (value) {
    if (value!.isEmpty) {
      return "Email Cannot Empty!";
    }
    return null;
  },
  controller: controller.emailController,
  keyboardType: TextInputType.emailAddress,
  decoration: const InputDecoration(
    hintText: 'Enter user email',
```

```
    prefixIcon: Icon(Icons.mail),
    label: Text('Email'),
  ), // InputDecoration
), // TextFormField
const SizedBox(
  height: 16.0,
), // SizedBox
TextFormField(
  validator: (value) {
    if (value!.isEmpty) {
      return "Password Cannot Empty!";
    }
    return null;
  },
  controller: controller.passwordController,
  keyboardType: TextInputType.text,
  obscureText: true,
  decoration: const InputDecoration(
    hintText: 'Enter user password',
    prefixIcon: Icon(Icons.mail),
    label: Text('Password'),
  ), // InputDecoration
), // TextFormField
const SizedBox(
  height: 16.0,
), // SizedBox
ElevatedButton(
  onPressed: () {
    if (addUserKey.currentState!.validate())
      controller.createUser();
    }
  },
  child: const Text('Create User')) // Elevat
],
```

Figure 3.3.11.3 users_add_admin_view.dart : Create User

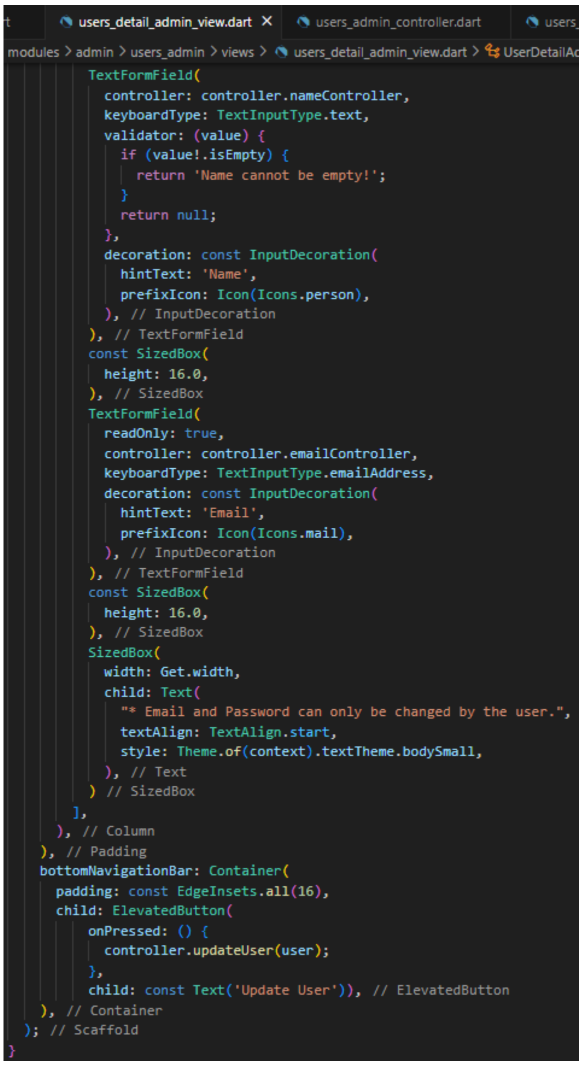
In Figure 3.3.11.4 await auth.createUserWithEmailAndPassword() is used to create a new user account with the given email address and password. Data of the new user will be stored in the Firestore database.


dart
users_admin_controller.dart
users_add_admin_view.dart
users_detail_adm
> modules > admin > users_admin > controllers > users_admin_controller.dart > ...

```dart
Future<void> createUser() async {
  try {
    EasyLoading.show(
      status: 'Uploading Image',
      maskType: EasyLoadingMaskType.black,
    );

    await auth
        .createUserWithEmailAndPassword(
            email: emailController.text, password: passwordController.text)
        .then((result) async {
      await userCollection.doc(result.user!.uid).set({
        'uid': result.user!.uid,
        'name': nameController.text,
        'email': emailController.text,
        'status': 'user',
        'isLogedIn': false,
        'isAdmin': false,
      }).then((_) async {
        if (imagePath != '') {
          String imagesFile =
              DateTime.now().microsecondsSinceEpoch.toString();
          Reference referenceRoot = FirebaseStorage.instance.ref();
          Reference referenceDirImages = referenceRoot.child("images");
          Reference referenceImageUpload =
              referenceDirImages.child(imagesFile);
          await referenceImageUpload.putFile(File(imagePath));
          String imageUrl = await referenceImageUpload.getDownloadURL();
          await userCollection
              .doc(result.user!.uid)
              .update({'profilePic': imageUrl});
        }
        await auth.signOut();
        var adminData = box.read('admin');
        await auth.signInWithEmailAndPassword(
            email: adminData['email'], password: adminData['password']);
        EasyLoading.dismiss();
        Get.offAllNamed(Routes.USERS_ADMIN);
        EasyLoading.showSuccess('New User Created!');
        clear();
        print(auth.currentUser?.email);
      });
    });
```

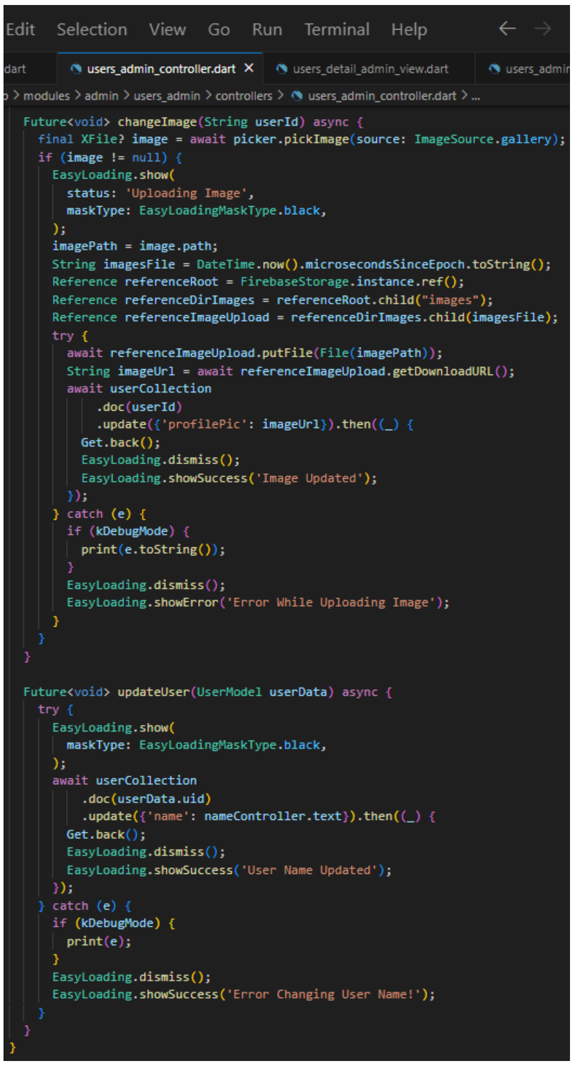
Figure 3.3.11.4 users_admin_controller.dart : Create User

Plus, admin can edit information of existing users as shown in Figure 3.3.11.5 and Figure 3.3.11.6. The nameController allows admin to edit the name of an existing user and if the admin leaves the field empty then the app will prompt an error message to alert the admin. Once the admin is done editing the username then clicking on the 'Update User' button will trigger the updateUser() whereby the app will look for the respective doc and updates data on the document. Admin can also update the profile picture of a user whereby the app allows admin to choose an image from the device gallery and once click on the 'Update User' button, the image will be updated and stored in the Firebase.


users_detail_admin_view.dart
users_admin_controller.dart
modules > admin > users_admin > views > users_detail_admin_view.dart > UserDetailAc

```dart
                    TextFormField(
                      controller: controller.nameController,
                      keyboardType: TextInputType.text,
                      validator: (value) {
                        if (value!.isEmpty) {
                          return 'Name cannot be empty!';
                        }
                        return null;
                      },
                      decoration: const InputDecoration(
                        hintText: 'Name',
                        prefixIcon: Icon(Icons.person),
                      ), // InputDecoration
                    ), // TextFormField
                    const SizedBox(
                      height: 16.0,
                    ), // SizedBox
                    TextFormField(
                      readOnly: true,
                      controller: controller.emailController,
                      keyboardType: TextInputType.emailAddress,
                      decoration: const InputDecoration(
                        hintText: 'Email',
                        prefixIcon: Icon(Icons.mail),
                      ), // InputDecoration
                    ), // TextFormField
                    const SizedBox(
                      height: 16.0,
                    ), // SizedBox
                    SizedBox(
                      width: Get.width,
                      child: Text(
                        "* Email and Password can only be changed by the user.",
                        textAlign: TextAlign.start,
                        style: Theme.of(context).textTheme.bodySmall,
                      ), // Text
                    ) // SizedBox
                  ],
                ), // Column
              ), // Padding
              bottomNavigationBar: Container(
                padding: const EdgeInsets.all(16),
                child: ElevatedButton(
                    onPressed: () {
                      controller.updateUser(user);
                    },
                    child: const Text('Update User')), // ElevatedButton
              ), // Container
            ); // Scaffold
          }
```

Figure 3.3.11.5 users_detail_admin_view.dart : Edit User

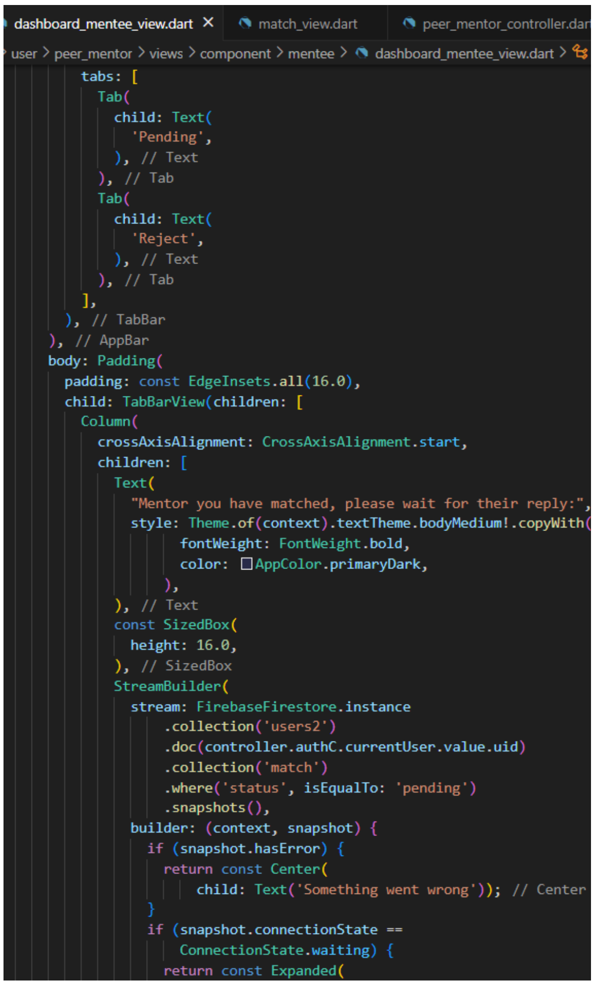

Edit
Selection
View
Go
Run
Terminal
Help
users_admin_controller.dart
users_detail_admin_view.dart
modules > admin > users_admin > controllers > users_admin_controller.dart > ...

```dart
  Future<void> changeImage(String userId) async {
    final XFile? image = await picker.pickImage(source: ImageSource.gallery);
    if (image != null) {
      EasyLoading.show(
        status: 'Uploading Image',
        maskType: EasyLoadingMaskType.black,
      );
      imagePath = image.path;
      String imagesFile = DateTime.now().microsecondsSinceEpoch.toString();
      Reference referenceRoot = FirebaseStorage.instance.ref();
      Reference referenceDirImages = referenceRoot.child("images");
      Reference referenceImageUpload = referenceDirImages.child(imagesFile);
      try {
        await referenceImageUpload.putFile(File(imagePath));
        String imageUrl = await referenceImageUpload.getDownloadURL();
        await userCollection
            .doc(userId)
            .update({'profilePic': imageUrl}).then((_) {
          Get.back();
          EasyLoading.dismiss();
          EasyLoading.showSuccess('Image Updated');
        });
      } catch (e) {
        if (kDebugMode) {
          print(e.toString());
        }
        EasyLoading.dismiss();
        EasyLoading.showError('Error While Uploading Image');
      }
    }
  }

  Future<void> updateUser(UserModel userData) async {
    try {
      EasyLoading.show(
        maskType: EasyLoadingMaskType.black,
      );
      await userCollection
          .doc(userData.uid)
          .update({'name': nameController.text}).then((_) {
        Get.back();
        EasyLoading.dismiss();
        EasyLoading.showSuccess('User Name Updated');
      });
    } catch (e) {
      if (kDebugMode) {
        print(e);
      }
      EasyLoading.dismiss();
      EasyLoading.showSuccess('Error Changing User Name!');
    }
  }
}
```

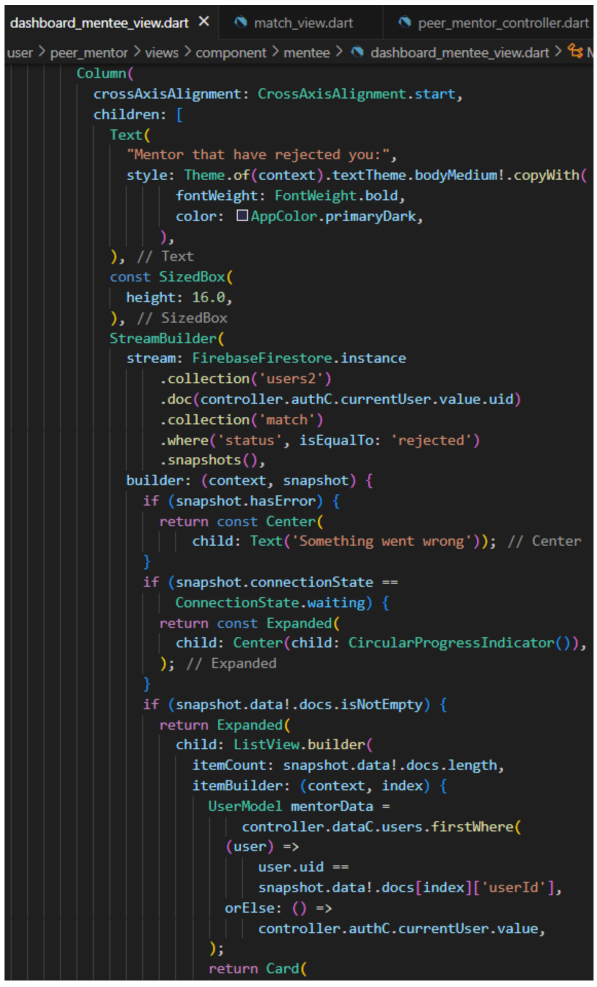
Figure 3.3.11.6 users_admin_controller.dart : Edit User

3.3.12 Peer Mentor Module Function Development (Student)

Students are given the option to register as a mentee or a mentor, however there are requirements to register as a mentor which is the years of study must not be less than 1 year. It is to ensure that the person registered as a mentor is a qualified person. After the student has registered a profile, the app will prompt them to different view (Mentor View / Mentee View). For the Mentee View, there are 3 main panels which are Dashboard, Match Mentor and Meet. For the mentee dashboard, if mentee has not had a mentor yet then their dashboard will only show pending and rejected match requests made by the mentee as shown in Figure 3.3.12.1 and Figure 3.3.12.2. Only after the mentee has a matching mentor then the app will display the matching mentor together with the meeting history.



```
dashboard_mentee_view.dart ×   match_view.dart   peer_mentor_controller.dar
user > peer_mentor > views > component > mentee > dashboard_mentee_view.dart >
          tabs: [
            Tab(
              child: Text(
                'Pending',
              ), // Text
            ), // Tab
            Tab(
              child: Text(
                'Reject',
              ), // Text
            ), // Tab
          ],
        ), // TabBar
      ), // AppBar
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: TabBarView(children: [
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
                "Mentor you have matched, please wait for their reply:",
                style: Theme.of(context).textTheme.bodyMedium!.copyWith(
                      fontWeight: FontWeight.bold,
                      color: AppColor.primaryDark,
                    ),
              ), // Text
              const SizedBox(
                height: 16.0,
              ), // SizedBox
              StreamBuilder(
                stream: FirebaseFirestore.instance
                    .collection('users2')
                    .doc(controller.authC.currentUser.value.uid)
                    .collection('match')
                    .where('status', isEqualTo: 'pending')
                    .snapshots(),
                builder: (context, snapshot) {
                  if (snapshot.hasError) {
                    return const Center(
                        child: Text('Something went wrong')); // Center
                  }
                  if (snapshot.connectionState ==
                      ConnectionState.waiting) {
                    return const Expanded(
```

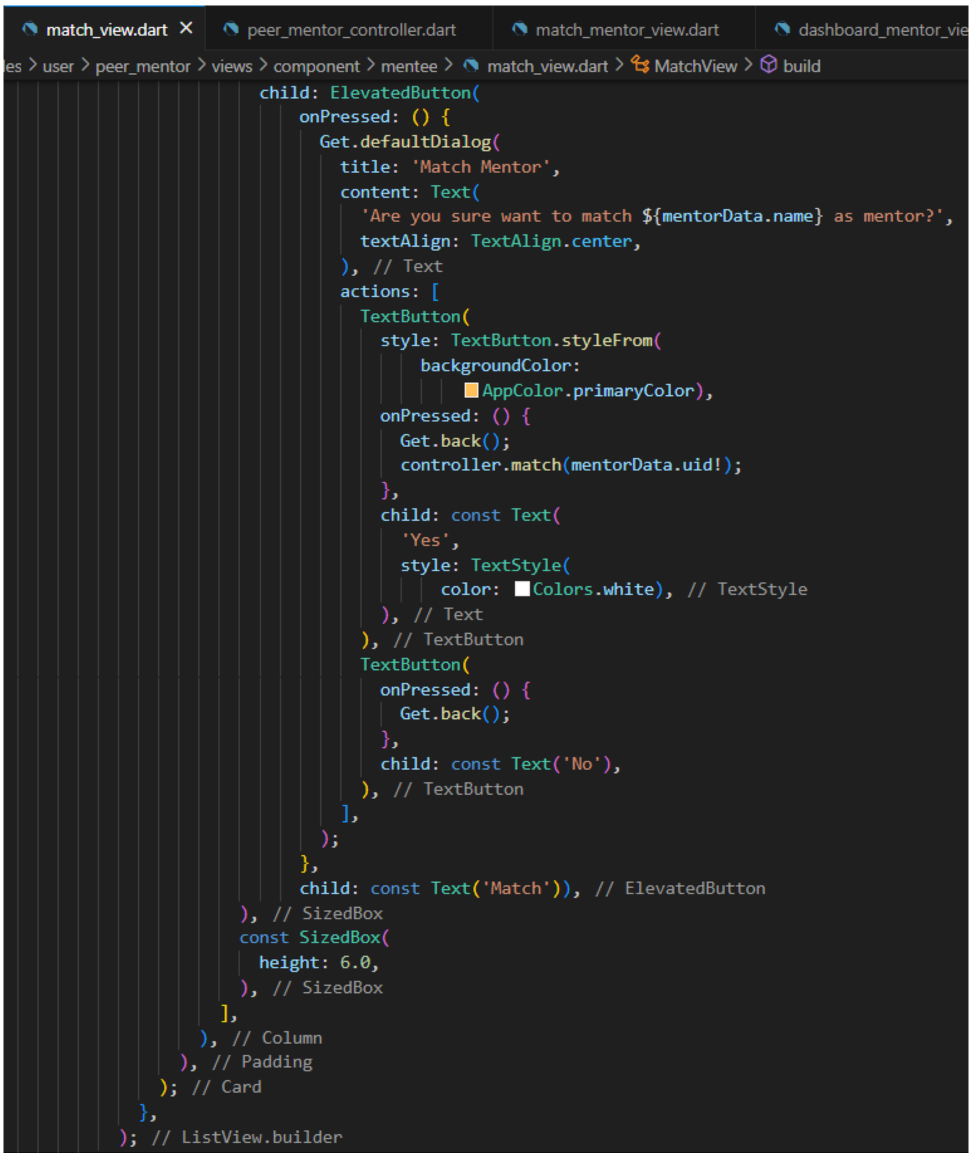


```
dashboard_mentee_view.dart ×   match_view.dart   peer_mentor_controller.dart
user > peer_mentor > views > component > mentee > dashboard_mentee_view.dart >
          Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
                "Mentor that have rejected you:",
                style: Theme.of(context).textTheme.bodyMedium!.copyWith(
                      fontWeight: FontWeight.bold,
                      color: AppColor.primaryDark,
                    ),
              ), // Text
              const SizedBox(
                height: 16.0,
              ), // SizedBox
              StreamBuilder(
                stream: FirebaseFirestore.instance
                    .collection('users2')
                    .doc(controller.authC.currentUser.value.uid)
                    .collection('match')
                    .where('status', isEqualTo: 'rejected')
                    .snapshots(),
                builder: (context, snapshot) {
                  if (snapshot.hasError) {
                    return const Center(
                        child: Text('Something went wrong')); // Center
                  }
                  if (snapshot.connectionState ==
                      ConnectionState.waiting) {
                    return const Expanded(
                      child: Center(child: CircularProgressIndicator()),
                    ); // Expanded
                  }
                  if (snapshot.data!.docs.isNotEmpty) {
                    return Expanded(
                      child: ListView.builder(
                        itemCount: snapshot.data!.docs.length,
                        itemBuilder: (context, index) {
                          UserModel mentorData =
                              controller.dataC.users.firstWhere(
                            (user) =>
                                user.uid ==
                                snapshot.data!.docs[index]['userId'],
                            orElse: () =>
                                controller.authC.currentUser.value,
                          );
                          return Card(
```

Figure 3.3.12.1 dashboard_mentee_view.dart (Pending Request View)

Figure 3.3.12.2 dashboard_mentee_view.dart (Reject Request View)

For the Match Mentor page, the app will display the list of mentors available for matching and after sending a match request to the mentor, the request will turn into a pending request and appear in the Pending Request page in Dashboard. All these data will be stored in the database under the collection 'users2' and for that specific user there will be another collection 'match' which contains information about the request status is equal to pending and the respective user ID that the request is made for as shown in Figure 3.3.12.1 and Figure 3.3.12.3.

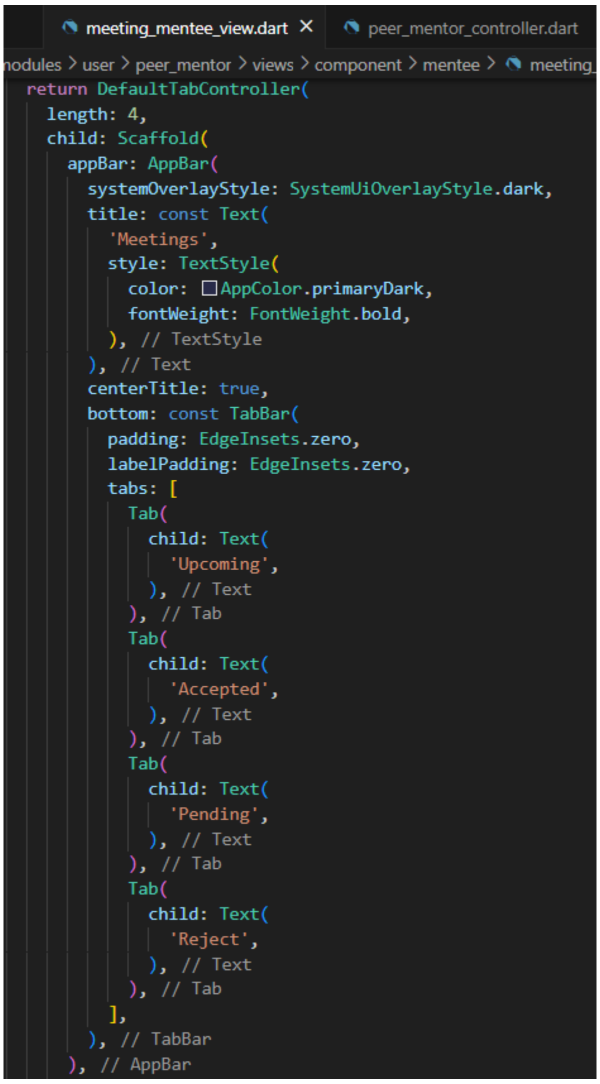

match_view.dart ×
peer_mentor_controller.dart
match_mentor_view.dart
dashboard_mentor_vie
es > user > peer_mentor > views > component > mentee > match_view.dart > MatchView > build

```dart
              child: ElevatedButton(
                  onPressed: () {
                    Get.defaultDialog(
                      title: 'Match Mentor',
                      content: Text(
                        'Are you sure want to match ${mentorData.name} as mentor?',
                        textAlign: TextAlign.center,
                      ), // Text
                      actions: [
                        TextButton(
                          style: TextButton.styleFrom(
                              backgroundColor:
                                  AppColor.primaryColor),
                          onPressed: () {
                            Get.back();
                            controller.match(mentorData.uid!);
                          },
                          child: const Text(
                            'Yes',
                            style: TextStyle(
                                color: Colors.white), // TextStyle
                          ), // Text
                        ), // TextButton
                        TextButton(
                          onPressed: () {
                            Get.back();
                          },
                          child: const Text('No'),
                        ), // TextButton
                      ],
                    );
                  },
                  child: const Text('Match')), // ElevatedButton
            ), // SizedBox
            const SizedBox(
              height: 6.0,
            ), // SizedBox
          ],
        ), // Column
      ), // Padding
    ); // Card
  },
); // ListView.builder
```

Figure 3.3.12.3 match_view.dart : Match Mentor (Mentee View)

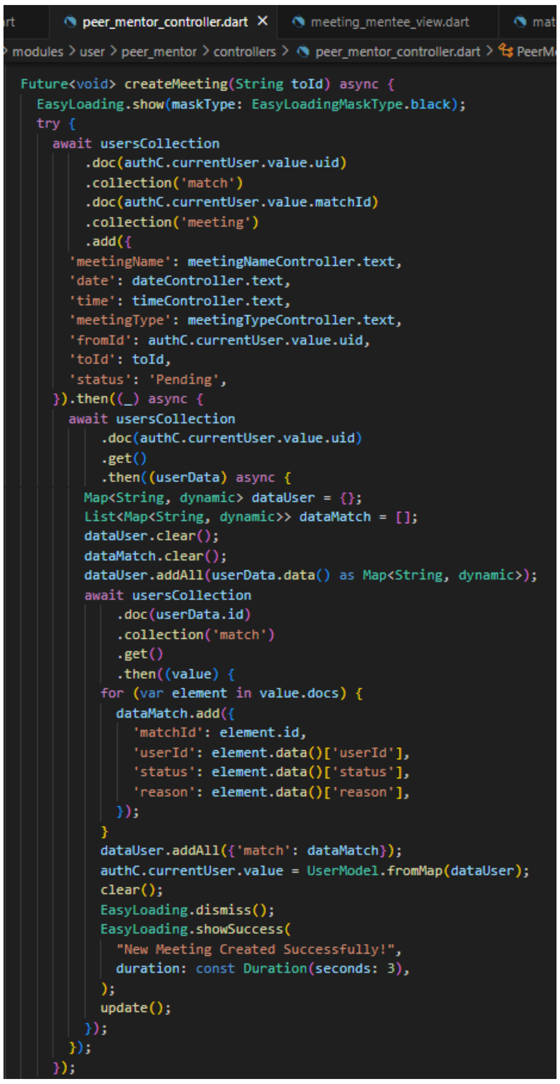
Next, the meetings page under mentee view there will be 4 main panels which are Upcoming, Accepted, Pending and Reject as shown in Figure 3.3.12.4. Under the meetings page, mentee can create a meeting by clicking on a button which will trigger the createMeeting() as shown in Figure 3.3.12.5. There are 4 main Text Editing Controller which are meetingNameController to input the meeting name, dateController to input the meeting date, timeController to input the meeting time and meetingTypeController to input the meeting type (online meeting /

physical meeting). After mentee has successfully created a meeting, the data will be saved under a collection 'meeting' in the Firestore database.


meeting_mentee_view.dart
peer_mentor_controller.dart
nodules > user > peer_mentor > views > component > mentee > meeting_

```dart
return DefaultTabController(
  length: 4,
  child: Scaffold(
    appBar: AppBar(
      systemOverlayStyle: SystemUiOverlayStyle.dark,
      title: const Text(
        'Meetings',
        style: TextStyle(
          color: AppColor.primaryDark,
          fontWeight: FontWeight.bold,
        ), // TextStyle
      ), // Text
      centerTitle: true,
      bottom: const TabBar(
        padding: EdgeInsets.zero,
        labelPadding: EdgeInsets.zero,
        tabs: [
          Tab(
            child: Text(
              'Upcoming',
            ), // Text
          ), // Tab
          Tab(
            child: Text(
              'Accepted',
            ), // Text
          ), // Tab
          Tab(
            child: Text(
              'Pending',
            ), // Text
          ), // Tab
          Tab(
            child: Text(
              'Reject',
            ), // Text
          ), // Tab
        ],
      ), // TabBar
    ), // AppBar
```

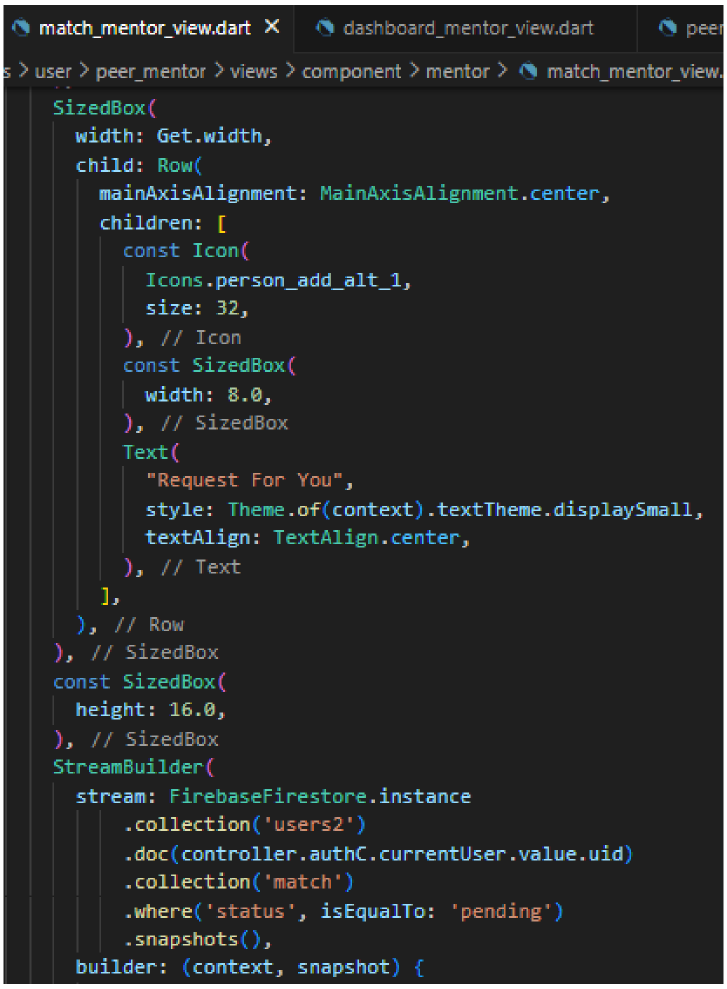
Figure 3.3.12.4 meeting_mentee_view.dart : Meetings Panel


peer_mentor_controller.dart
meeting_mentee_view.dart
modules > user > peer_mentor > controllers > peer_mentor_controller.dart > PeerM

```dart
Future<void> createMeeting(String toId) async {
  EasyLoading.show(maskType: EasyLoadingMaskType.black);
  try {
    await usersCollection
        .doc(authC.currentUser.value.uid)
        .collection('match')
        .doc(authC.currentUser.value.matchId)
        .collection('meeting')
        .add({
      'meetingName': meetingNameController.text,
      'date': dateController.text,
      'time': timeController.text,
      'meetingType': meetingTypeController.text,
      'fromId': authC.currentUser.value.uid,
      'toId': toId,
      'status': 'Pending',
    }).then((_) async {
      await usersCollection
          .doc(authC.currentUser.value.uid)
          .get()
          .then((userData) async {
        Map<String, dynamic> dataUser = {};
        List<Map<String, dynamic>> dataMatch = [];
        dataUser.clear();
        dataMatch.clear();
        dataUser.addAll(userData.data() as Map<String, dynamic>);
        await usersCollection
            .doc(userData.id)
            .collection('match')
            .get()
            .then((value) {
          for (var element in value.docs) {
            dataMatch.add({
              'matchId': element.id,
              'userId': element.data()['userId'],
              'status': element.data()['status'],
              'reason': element.data()['reason'],
            });
          }
          dataUser.addAll({'match': dataMatch});
          authC.currentUser.value = UserModel.fromMap(dataUser);
          clear();
          EasyLoading.dismiss();
          EasyLoading.showSuccess(
            "New Meeting Created Successfully!",
            duration: const Duration(seconds: 3),
          );
          update();
        });
      });
    });
```

Figure 3.3.12.5 peer_mentor_controller.dart : Create Meeting

For the mentor view, there are also 3 main panels which are Dashboard, Match and Meet. The mentor view of the Dashboard and Meetings page shares an almost similar view as the mentee, only the Match page is different. For the Mentor Match page, the app will display the current mentor's profile and all the requests received from mentees as shown in Figure 3.3.12.6. The status for the match request is under pending and is stored inside the Firebase until the mentor decided to accept or reject the request.

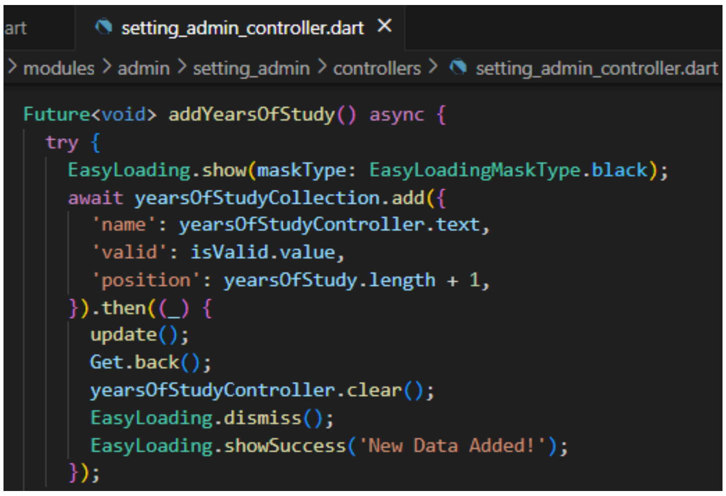

match_mentor_view.dart
dashboard_mentor_view.dart
peer
s > user > peer_mentor > views > component > mentor > match_mentor_view.

```
SizedBox(
  width: Get.width,
  child: Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      const Icon(
        Icons.person_add_alt_1,
        size: 32,
      ), // Icon
      const SizedBox(
        width: 8.0,
      ), // SizedBox
      Text(
        "Request For You",
        style: Theme.of(context).textTheme.displaySmall,
        textAlign: TextAlign.center,
      ), // Text
    ],
  ), // Row
), // SizedBox
const SizedBox(
  height: 16.0,
), // SizedBox
StreamBuilder(
  stream: FirebaseFirestore.instance
      .collection('users2')
      .doc(controller.authC.currentUser.value.uid)
      .collection('match')
      .where('status', isEqualTo: 'pending')
      .snapshots(),
  builder: (context, snapshot) {
```

Figure 3.3.12.6 match_mentor_view.dart (Mentor View)

3.3.13 Add / Delete / Edit Options Function Development (Admin)

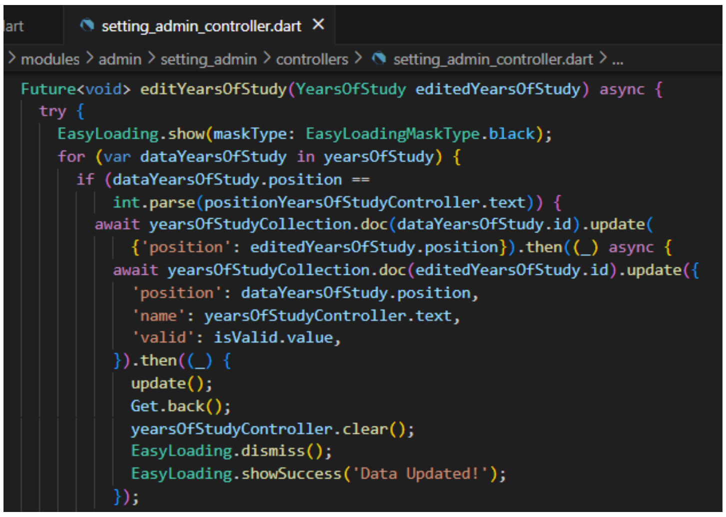
Admin can add, delete and even edit the faculty options and years of study options which is required when user registers as mentor / mentee. When admin clicks on the 'Add Data' button, the app will trigger the addYearsOfStudy() as shown in Figure 3.3.13.1 whereby the app will take the name from yearsOfStudyController as the option name, and validate whether the option is applicable for mentor or not as to register for mentor the requirement is years of study must not be less than 1 year and lastly the position of the option will be located.


art | setting_admin_controller.dart ×
> modules > admin > setting_admin > controllers > setting_admin_controller.dart

```dart
Future<void> addYearsOfStudy() async {
  try {
    EasyLoading.show(maskType: EasyLoadingMaskType.black);
    await yearsOfStudyCollection.add({
      'name': yearsOfStudyController.text,
      'valid': isValid.value,
      'position': yearsOfStudy.length + 1,
    }).then((_) {
      update();
      Get.back();
      yearsOfStudyController.clear();
      EasyLoading.dismiss();
      EasyLoading.showSuccess('New Data Added!');
    });
```

Figure 3.3.13.1 setting_admin_controller.dart: addYearsOfStudy()

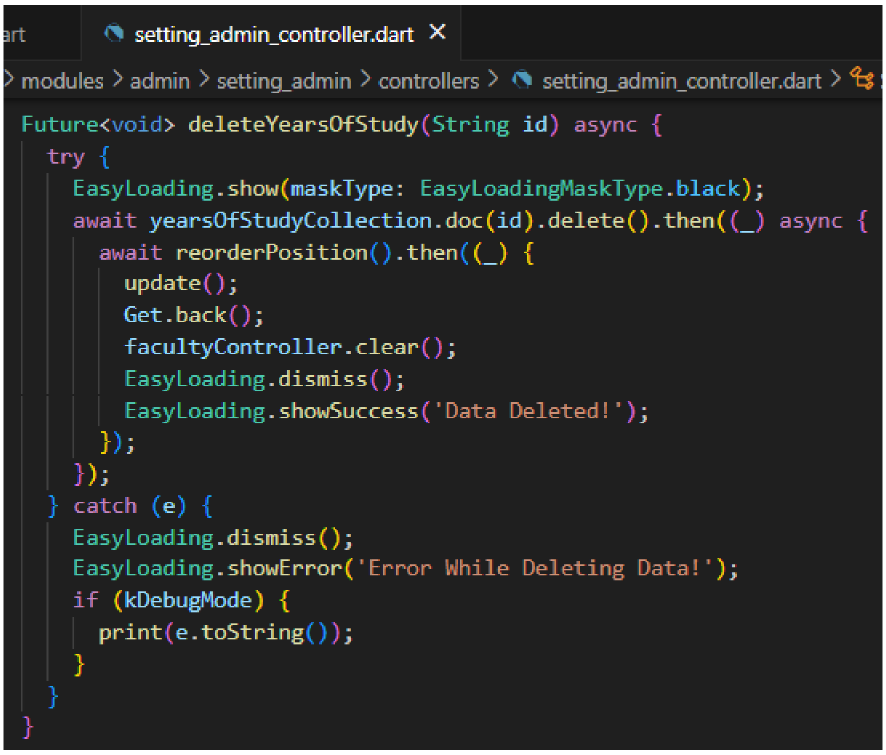
If the admin wants to edit an existing option, admin can click on an option and there will be fields which admin can edit which are the name of the option, position of the option and a checkbox for options which are applicable to mentor. After user done editing the option, user clicks on the 'Edit' button to trigger the editYearsOfStudy() where all the information will be validated and updated to the Firebase as shown in Figure 3.3.13.2.


art | setting_admin_controller.dart ×
> modules > admin > setting_admin > controllers > setting_admin_controller.dart > ...

```dart
Future<void> editYearsOfStudy(YearsOfStudy editedYearsOfStudy) async {
  try {
    EasyLoading.show(maskType: EasyLoadingMaskType.black);
    for (var dataYearsOfStudy in yearsOfStudy) {
      if (dataYearsOfStudy.position ==
          int.parse(positionYearsOfStudyController.text)) {
        await yearsOfStudyCollection.doc(dataYearsOfStudy.id).update(
            {'position': editedYearsOfStudy.position}).then((_) async {
          await yearsOfStudyCollection.doc(editedYearsOfStudy.id).update({
            'position': dataYearsOfStudy.position,
            'name': yearsOfStudyController.text,
            'valid': isValid.value,
          }).then((_) {
            update();
            Get.back();
            yearsOfStudyController.clear();
            EasyLoading.dismiss();
            EasyLoading.showSuccess('Data Updated!');
          });
```

Figure 3.3.13.2 setting_admin_controller.dart: editYearsOfStudy()

Lastly, to delete an option, admin can click on an option and there will be a 'Delete' button which will trigger the deleteYearsOfStudy() where the current option will be deleted from the yearsOfStudyCollection in the Firebase as shown in Figure 3.3.13.3.

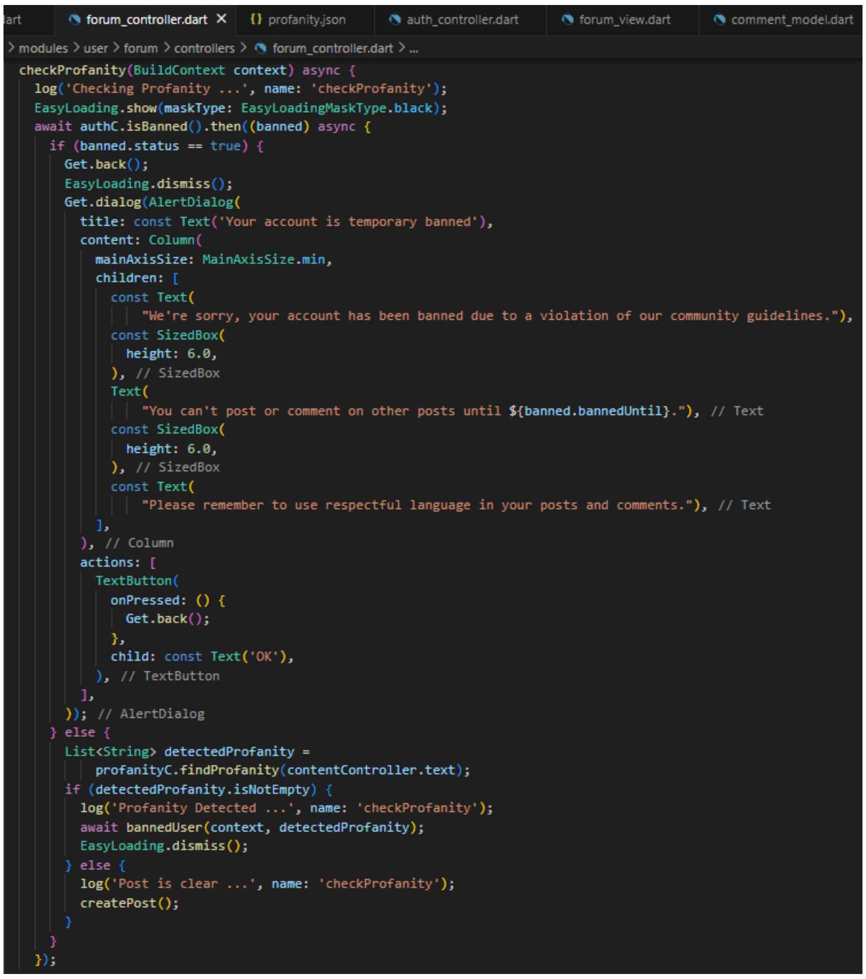

art
setting_admin_controller.dart
modules > admin > setting_admin > controllers > setting_admin_controller.dart >

```
Future<void> deleteYearsOfStudy(String id) async {
  try {
    EasyLoading.show(maskType: EasyLoadingMaskType.black);
    await yearsOfStudyCollection.doc(id).delete().then((_) async {
      await reorderPosition().then((_) {
        update();
        Get.back();
        facultyController.clear();
        EasyLoading.dismiss();
        EasyLoading.showSuccess('Data Deleted!');
      });
    });
  } catch (e) {
    EasyLoading.dismiss();
    EasyLoading.showError('Error While Deleting Data!');
    if (kDebugMode) {
      print(e.toString());
    }
  }
}
```

Figure 3.3.13.3 setting_admin_controller.dart: deleteYearsOfStudy()

3.3.14 Profanity detection Function Development

Students can create posts and comment anonymously, however they will need to obey the community guidelines which is to not include any swear and obscene language in their post. Therefore, a profanity detector is created to detect any profanity language. If found any profanity language used in a post/comment, the user shall be banned. The check profanity function will be triggered each time a student makes a post or a comment. Figure 3.3.14.1 shows the profanity detector checking profanity for a post/comment. The detector will first validate whether the current user is under ban, if the user is under ban, then the user cannot make a post/comment for a certain period of time. If the user is not under ban, then the checkProfanity() will validate the presence of profanity words, if there are any profanity words detected then the user will be placed under ban whereas if there is not profanity detected then the student can post/comment.



```dart
checkProfanity(BuildContext context) async {
  log('Checking Profanity ...', name: 'checkProfanity');
  EasyLoading.show(maskType: EasyLoadingMaskType.black);
  await authC.isBanned().then((banned) async {
    if (banned.status == true) {
      Get.back();
      EasyLoading.dismiss();
      Get.dialog(AlertDialog(
        title: const Text('Your account is temporary banned'),
        content: Column(
          mainAxisSize: MainAxisSize.min,
          children: [
            const Text(
                "We're sorry, your account has been banned due to a violation of our community guidelines."),
            const SizedBox(
              height: 6.0,
            ), // SizedBox
            Text(
                "You can't post or comment on other posts until ${banned.bannedUntil}."), // Text
            const SizedBox(
              height: 6.0,
            ), // SizedBox
            const Text(
                "Please remember to use respectful language in your posts and comments."), // Text
          ],
        ), // Column
        actions: [
          TextButton(
            onPressed: () {
              Get.back();
            },
            child: const Text('OK'),
          ), // TextButton
        ],
      )); // AlertDialog
    } else {
      List<String> detectedProfanity =
          profanityC.findProfanity(contentController.text);
      if (detectedProfanity.isNotEmpty) {
        log('Profanity Detected ...', name: 'checkProfanity');
        await bannedUser(context, detectedProfanity);
        EasyLoading.dismiss();
      } else {
        log('Post is clear ...', name: 'checkProfanity');
        createPost();
      }
    }
  });
```

Figure 3.3.14.1 forum_controller.dart: checkProfanity()

Chapter 4 Methodology and Tools

Chapter 4 explains the methodology and approach used to develop the project. Both hardware and software used for project development are listed. Lastly, a Gantt chart will show the overall timeline for Final Year Project 1 and Final Year Project 2.

4.1 Methodologies

For this project, the methodology chosen is Rapid Application Development (RAD) methodology. The proposed methodology helps quickly develop prototypes to test various functions. Changes can be made to the design by removing or adding new features without starting from scratch and worrying about any effects on the final product. It helps ensure that the end product is more quality focused and can meet users' requirements. Figure 4.1 shows that RAD methodology has 4 phases: requirements planning, user design, construction, and cutover.

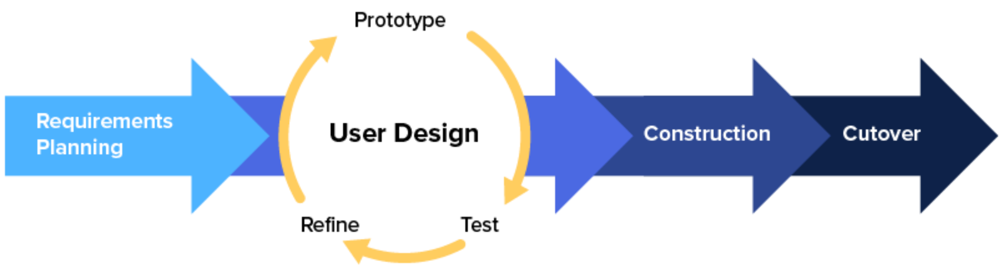


Figure 4.1 RAD Methodology

- **Requirements Planning**

The first phase of RAD is requirements planning. In this phase, an overall idea of how the project will be implemented is formed by outlining the problem's scope and corresponding solutions. The initial idea is to develop a community app for UTAR students to help solve the common issues they face in their higher education journey. Then, literature research on related community apps will be carried out to analyze further the strengths and weaknesses of existing apps which can be implemented in the final product. The existing apps reviewed are hi-hive Community, Raklet, Reddit, InterNations and ZeeMee. After reviewing the literature study, the project scope and objectives can be specified. Moreover, it is also important to identify both hardware and software specifications required for the project to ensure that the app may be developed successfully without worrying about hardware/software compatibility issues.

- **User Design**

Once the project objective and scope have been decided, the project moves on to the second phase, User Design. In this phase, user designs are built into various initial prototypes based on user requirements drafted during the requirements planning phase. In the early stage of prototyping, the prototypes do not necessarily need to satisfy all functions and may only involve the essential functions. The key functions of the **UTAR Community Application for Students** mobile application are location sharing, anonymous forum posting and mentor-mentee registration. Furthermore, this phase also involves constant testing each time a new prototype is developed. Testing the app helps validate whether the app can perform the functions as expected and identify the existence of any bugs. If the app is not performing as expected or there are major bugs, then debug process takes place where issues within the app will be fixed so that the app can function properly without glitches.

- **Construction**

In the construction phase, the prototypes developed will be refined and converted into a working model. This phase considers functions and the app's overall aesthetics and usability. Any changes or new functions can still be implemented to improve the overall aspect of the app.

- **Cutover**

It is the final phase, where the prototype is finalized and launched for use by others. For this phase, the final product will be optimized and maintained by performing full-scale testing to continue looking for bugs. Then the final output of the project will be documented for future reference.

4.2 System Requirement

4.2.1 Hardware

The hardware involved in this project is a laptop and an Android mobile device. Table 4.2.1.1 shows the laptop specifications required for the project. A laptop with Microsoft Windows 11 will be used to install the relevant software tools needed for app development. Then, an Android mobile phone will be used for testing and deploying the app. Table 4.2.1.2 shows the Android mobile device specification used for the project.

Components	Specifications
Model	ASUS TUF Gaming A15 FA506II_FA506II
CPU	AMD Ryzen 7 4800H with Radeon Graphics, 2900 Mhz, 8 Core(s), 16 Logical Processor(s)
Operating System	Microsoft Windows 11 Home Single Language
GPU	NVIDIA GeForce GTX 1650 Ti AMD Radeon (TM) Graphics
RAM	16.0 GB
Storage	Kingston SSD 512GB OM8PCP3512F-AB NVMe

Table 4.2.1.1 Specifications of laptop

Components	Specifications
Model	Oppo R17 Pro
CPU	x 2.2 GHz Kryo 360, 6x 1.7 GHz Kryo 360, Cores: 8
Operating System	ColorOS 5.2 (Android 8.1 Oreo)
GPU	Qualcomm Adreno 616, 500 MHz
RAM	8GB
Storage	128GB

Table 4.2.1.2 Specifications of mobile device

4.2.2 Software

Table 4.2.2.1 shows the software requirements to develop the project. The IDE tool chosen for mobile app development is Visual Studio Code 1.77.3. Many developers commonly use it to create various software applications. Before testing the app on a physical mobile device, Android Studio version Electric Eel | 2022.1.1 Patch 1 is used to create Android emulators to simulate the app in a virtual environment. Next, Firebase: a NoSQL cloud database, is used for the backend development. Finally, the Flutter framework is used for building the Android application, while the primary programming language for front-end development is Dart.

Components	Specifications
Tools	**<u>Visual Studio Code</u>** Visual Studio Code is an open-source code editor which allows developers to edit and debug code in various programming languages, including Dart, C++, Python, PHP and many more. Plus, Visual Studio code has a list of free extensions available for installation, which can further support development workflow [9].
	<u>Android Studio Emulator</u> Android Emulator simulates Android devices on a laptop/computer, allowing developers to test an Android application on various devices with different API levels without needing a physical device. [10]
	<u>Firebase</u> Firebase is a toolset that provides hosted backend services which help developers build and improve their apps. These services include Authentication, Realtime Database, Hosting and many more. Using Firebase as a backend allows connected users easily store and sync their data in real-time. [11]

Languages, Libraries and Framework	**Flutter** Flutter is a powerful and reliable open-source framework for creating mobile applications for both Android and iOS. It includes useful features such as Hot reload, Cross-platform compatibility etc., which helps developers produce high-quality and fast-performing applications. [12]
	Dart Dart is an open-source, client-side programming language used to code Flutter applications. It is used for developing the front-end of cross-platform mobile applications and comes with pre-built libraries that can be imported to optimize the developer's tasks. [13]

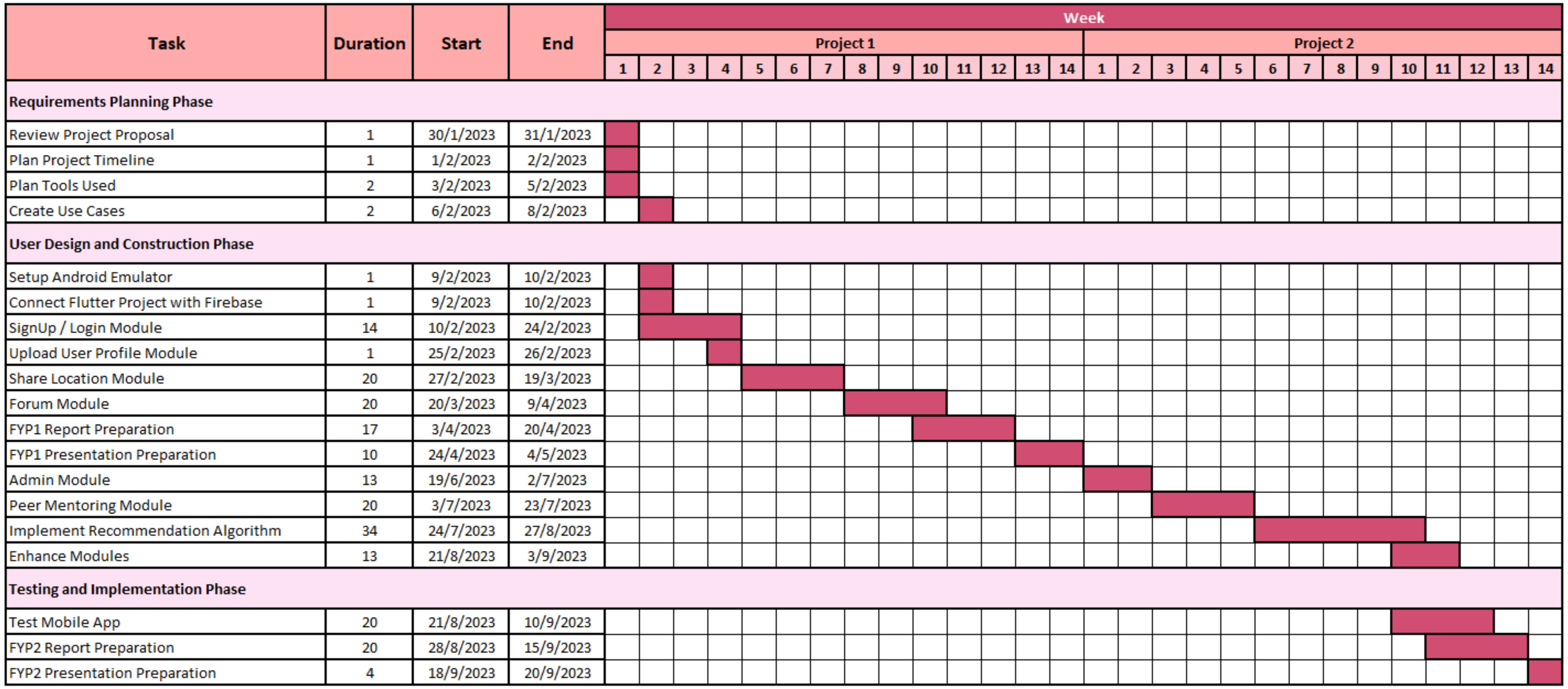
Table 4.2.2.1 Software Components for mobile app development

4.3 Timeline

The project timeline is planned based on the methodology, which consist of 3 main phases: Requirements planning phase, User design and construction phase, Testing and implementation phase. Figure 4.3.1 shows the timeline for Final Year Project 1 (January 2023 trimester) and Final Year Project 2 (June 2023 trimester). The duration for the whole project is around 28 weeks which is about 7 months.

For the first 2 weeks of Final Year Project 1, requirements planning activities are conducted to ensure the project's requirements are clear and the project can complete in time according to the project timeline with proper planning. For the upcoming weeks, certain modules are developed such as SignUp / Login Module, Upload User Profile Module, Share Location Module and Forum Module. Tools used for development and the Firebase database are also configured to move on to project development. At the end of the trimester, Final Year Project 1 report is written and submitted, and a presentation on the project is done.

For Final Year Project 2, the main tasks to complete are the Admin Module, Peer Mentoring Module, and implementing the profanity filter into the application. It is estimated that all modules will be completed by Week 10, and testing and enhancement of the modules can be done. In the meantime, the Final Year Project 2 report is prepared and submitted before the deadline (15/9/23). Lastly, the full project will be presented at the final week (20/9/23).



Task	Duration	Start	End	Week																											
				Project 1														Project 2													
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Requirements Planning Phase																															
Review Project Proposal	1	30/1/2023	31/1/2023	🟥																											
Plan Project Timeline	1	1/2/2023	2/2/2023	🟥																											
Plan Tools Used	2	3/2/2023	5/2/2023	🟥																											
Create Use Cases	2	6/2/2023	8/2/2023		🟥																										
User Design and Construction Phase																															
Setup Android Emulator	1	9/2/2023	10/2/2023		🟥																										
Connect Flutter Project with Firebase	1	9/2/2023	10/2/2023		🟥																										
SignUp / Login Module	14	10/2/2023	24/2/2023		🟥	🟥	🟥																								
Upload User Profile Module	1	25/2/2023	26/2/2023				🟥																								
Share Location Module	20	27/2/2023	19/3/2023					🟥	🟥	🟥																					
Forum Module	20	20/3/2023	9/4/2023								🟥	🟥	🟥																		
FYP1 Report Preparation	17	3/4/2023	20/4/2023										🟥	🟥	🟥																
FYP1 Presentation Preparation	10	24/4/2023	4/5/2023													🟥	🟥														
Admin Module	13	19/6/2023	2/7/2023															🟥	🟥												
Peer Mentoring Module	20	3/7/2023	23/7/2023																	🟥	🟥	🟥									
Implement Recommendation Algorithm	34	24/7/2023	27/8/2023																				🟥	🟥	🟥	🟥	🟥				
Enhance Modules	13	21/8/2023	3/9/2023																								🟥	🟥			
Testing and Implementation Phase																															
Test Mobile App	20	21/8/2023	10/9/2023																								🟥	🟥	🟥		
FYP2 Report Preparation	20	28/8/2023	15/9/2023																									🟥	🟥	🟥	
FYP2 Presentation Preparation	4	18/9/2023	20/9/2023																												🟥

Figure 4.3 Gantt Chart for project timeline

4.4 Summary

In summary, the methodology used for this project is the RAD-based methodology as changes can be made during the development process without having to worry about the effects on the final product. Then, the hardware and software specifications for project development are specified in the chapter. A laptop and mobile phone are required for app development, while the main programming language is Dart. Lastly, a Gantt chart shows the overview of the project timeline for better visualization.

Chapter 5 Implementation and Testing

In Final Year Project 2, all modules are fully developed for student and administrator. These modules also include the 3 main functions which are Friends Nearby, UTAR Forum, and Peer Mentoring. The UTAR Community Application for Students is named Community App and has a logo resembling the letter "U" for UTAR. The app has been tested on emulators and physical devices, meaning it can run on various Android devices. For this app, only students are required to register an account to use the app, whereas administrators can just log in with a default admin account. There are certain pages where both admin and students share the same view, but their access levels would be different. This is because students will mainly use the app functions whereas admin will only use the app to moderate user activity, giving admin the privilege to view and alter user's information.

<u>Sign Up / Login Module</u>

5.1 Sign Up and Login

When a user launches the app on a device, the user will be directed to the login page as shown in Figure 5.1.1. Users can log in to the app by filling in their email and password and then click on the Login button. Once the user clicks the Login button, the system will validate the data filled in by the user. If the user did not fill in their email / password, the system will alert the user by showing a message requesting that the user fill in their email / password. The system will prompt an error message for invalid email and password combinations, and the user must edit their login information. Users must ensure they have a valid user account and login details filled in correctly to log in to the app.

If a user has not registered an account, the user can click on the Signup text on the login page. After clicking on the Signup text, user will be redirected to the signup page as shown in Figure 5.1.2. Users can register an account by filling in their full name, email and password, then click the Sign Up button. Once the user clicks the Sign Up button, the system will validate the data filled in by the user. If user leaves any fields empty, the system will alert the user by showing a message requesting the user fill in their information. For those invalid data entered in the fields, the system will alert the user and the user must edit their signup information. Furthermore, the user must ensure the password entered is valid with a minimum of 6 characters, and both password fields must have matching passwords. Once users have filled in their signup details correctly, they can successfully create an account and will be directed to the login page to log in to the app.

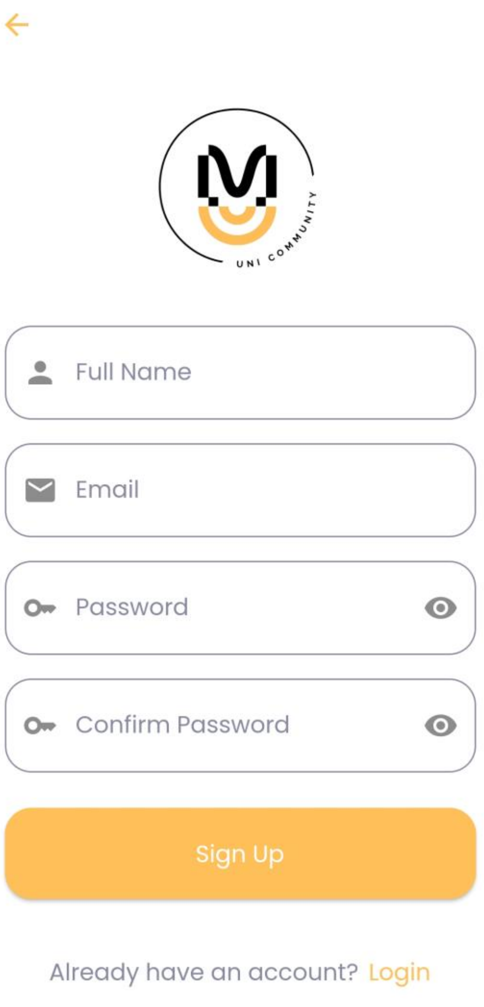


Figure 5.1.1 Login UI

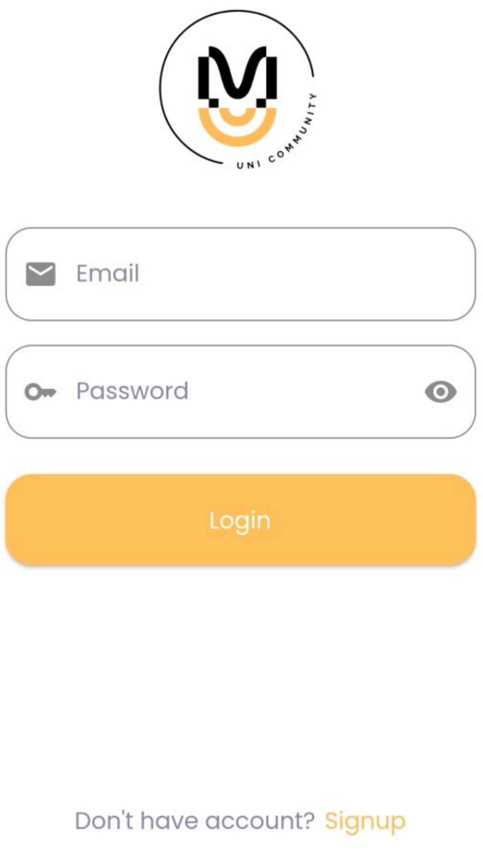


Figure 5.1.2 Sign Up UI

Share Location Module

5.2 Active Current Location

When a student successfully login to the app, they will reach the homepage, which shows a Google Map with buttons on the bottom right corner and a button on the top right corner. The button on the top right corner allows users to filter available users within the set radius, as shown in Figure 5.2.1. Before accessing the current location, the app will show a default location on the map, and the user must click the "Active Current Location" button. Before clicking on the button, the user must allow the app to access the device's location, allowing the app to get the user's current location and display it with a marker on the map, as shown in Figure 5.2.2. As the user clicks on their own location marker, there will be an info window with the message "You are here".

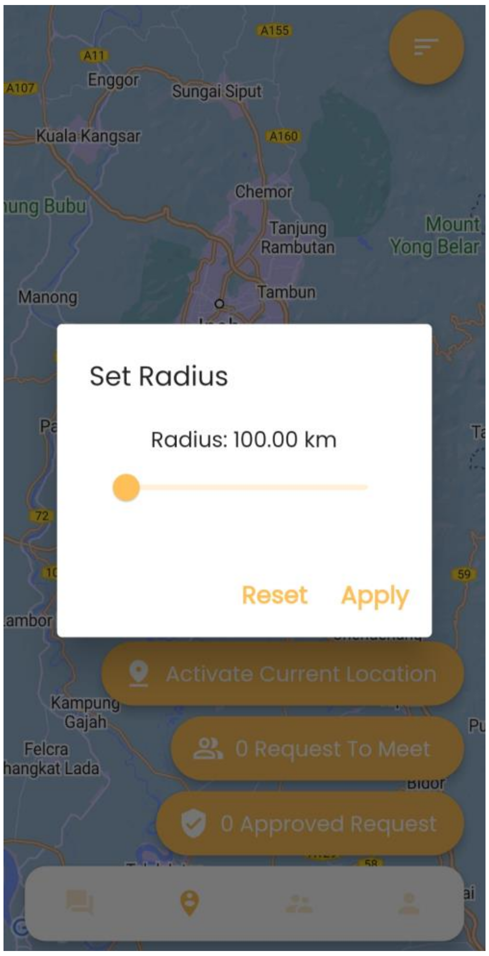


Figure 5.2.1 Set Radius function

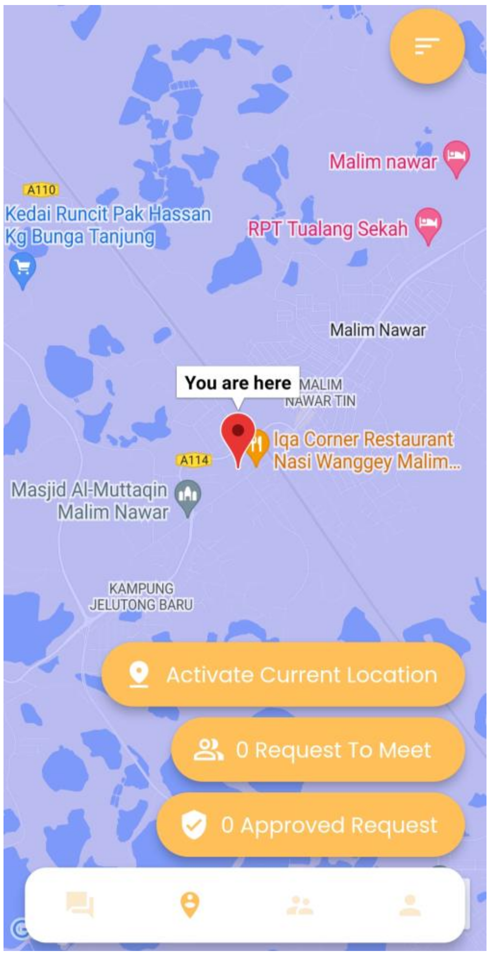


Figure 5.2.2 Display Location With Marker

5.3 Share Location Request

The app enables the user to view other available users nearby. In Figure 5.3.1, the current user's location marker is in red, whereas the other user is in yellow, named Chai Yi Yep. If the current user wishes to meet Chai and send a request for location sharing, they can click on Chai's location marker, and the app will prompt a dialog box with the message: "Ask Chai to meet?" and the current user can click on the button with the text: "Yes request permission". After clicking the button, a notification will inform the current user that the request has been successfully sent to Chai.

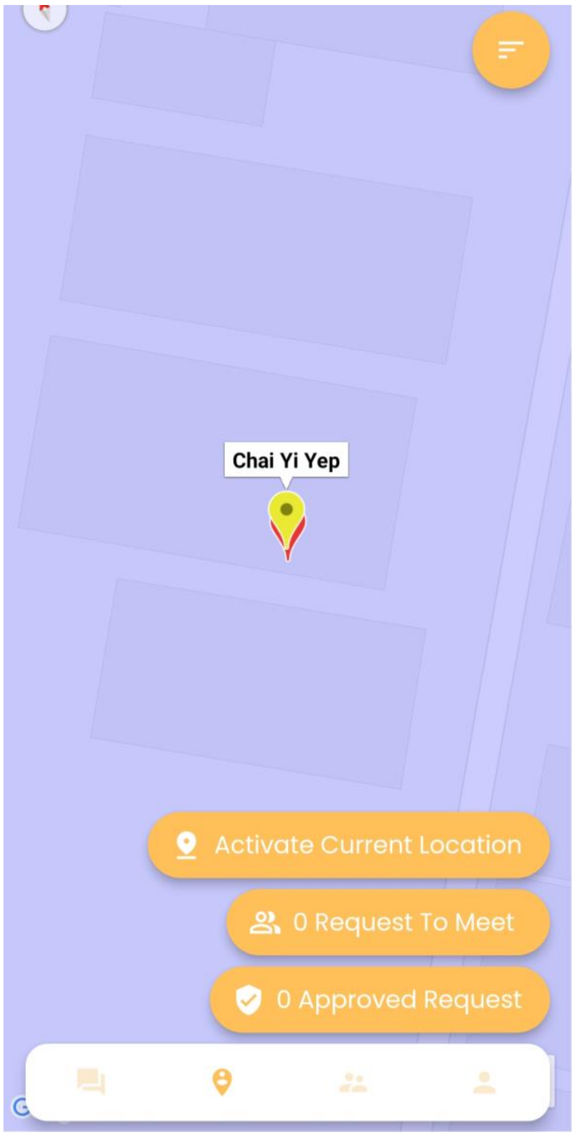


Figure 5.3.1 Other User's Location Marker

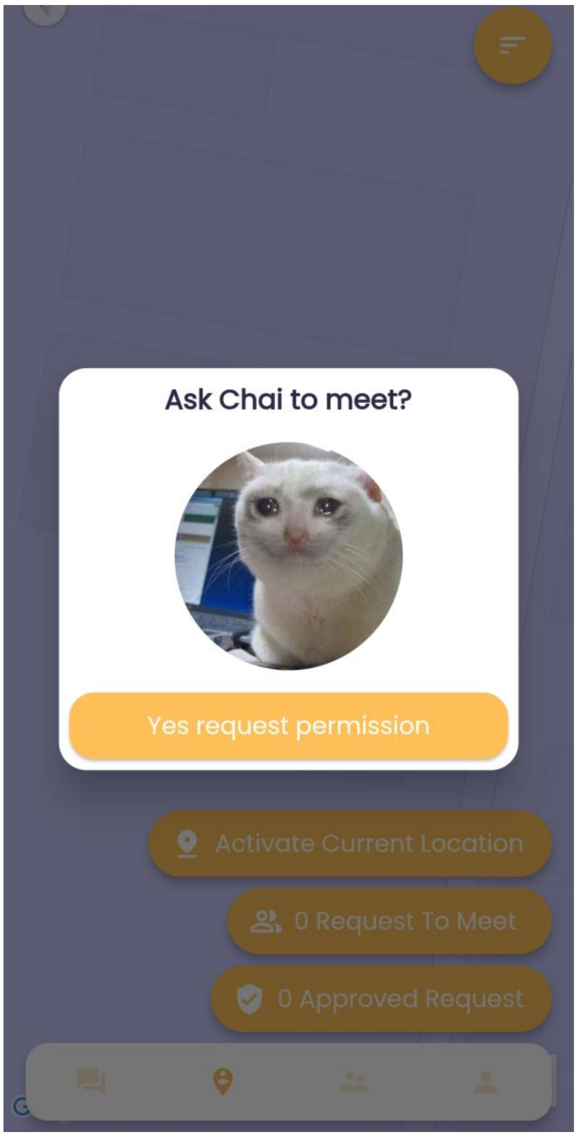
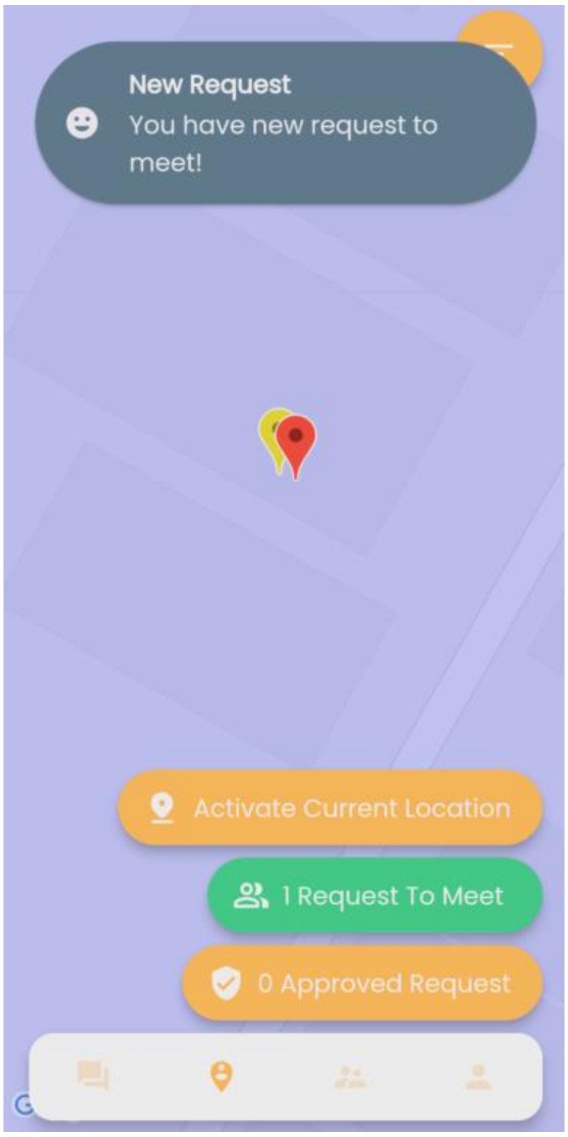


Figure 5.3.2 Request Permission for Location Sharing

Chai will receive a request to meet +1 on their side as shown in Figure 5.3.3, and as Chai clicks on the request to meet button, there will be a request with the message of which user wants to meet in this case, the user named Chang Ping Yen. Chai can approve the request by clicking on the 'Approve' button or reject the request by clicking on the 'Reject' button as shown in Figure 5.3.4.



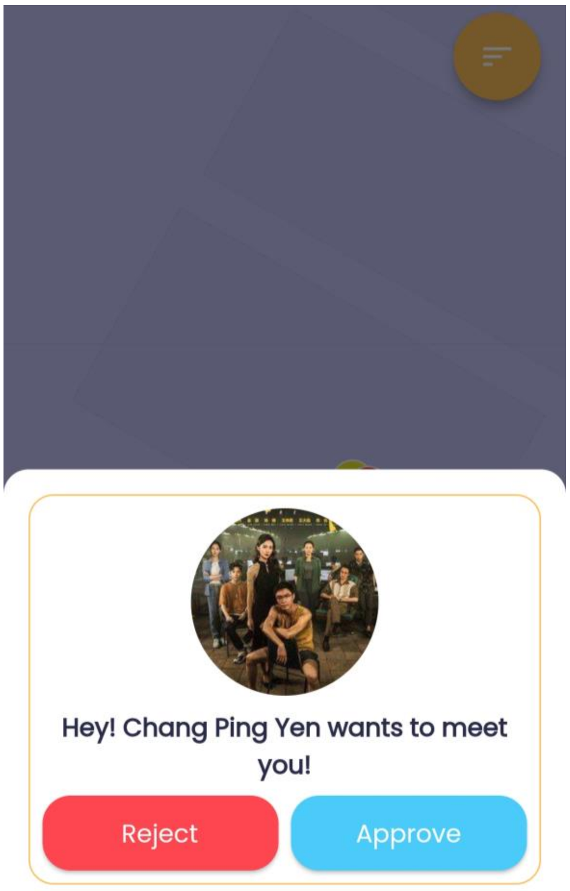


Figure 5.3.3 Request to Meet

Figure 5.3.4 Reject / Approve Request to Meet

Chai can proceed to approve the request and Chang will receive an approved request +1 on their side as shown in Figure 5.3.5. As Chang clicks on the approved request button, a window will pop out with the message indicating Chai has approved the request, and 2 buttons will appear as shown in Figure 5.3.6. The top button (Click Here to Navigate) allows Chang to navigate to Chai's location, and the bottom button (Scan QR Code) allows Chang to scan the

QR code when meeting Chai. Chang can then click the top button and be directed to the Google Maps app to navigate to Chai's location.

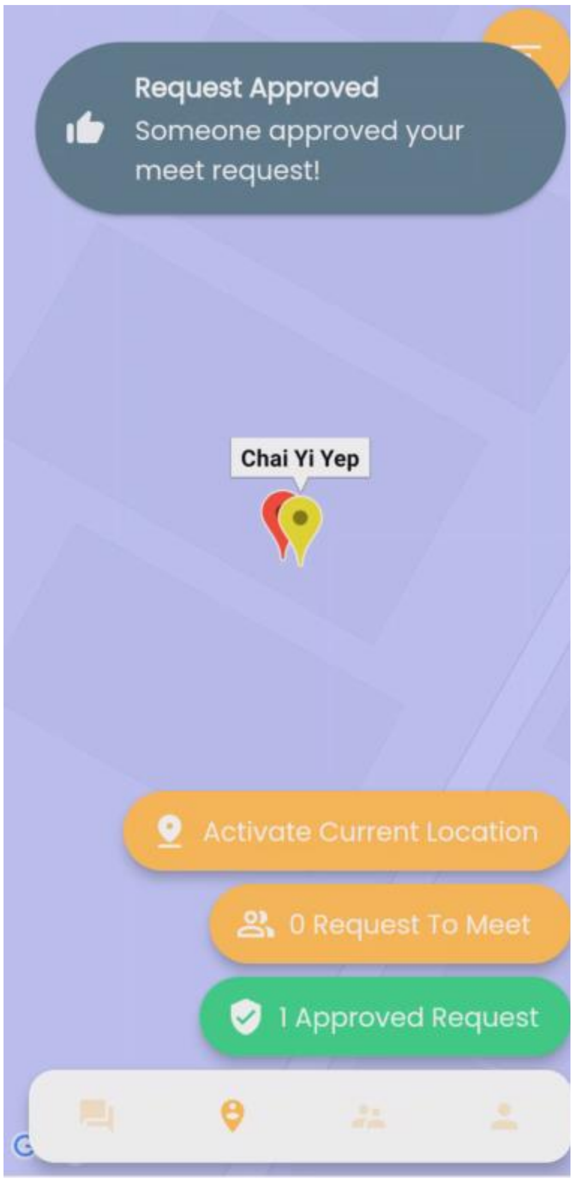


Figure 5.3.5 Approved Request

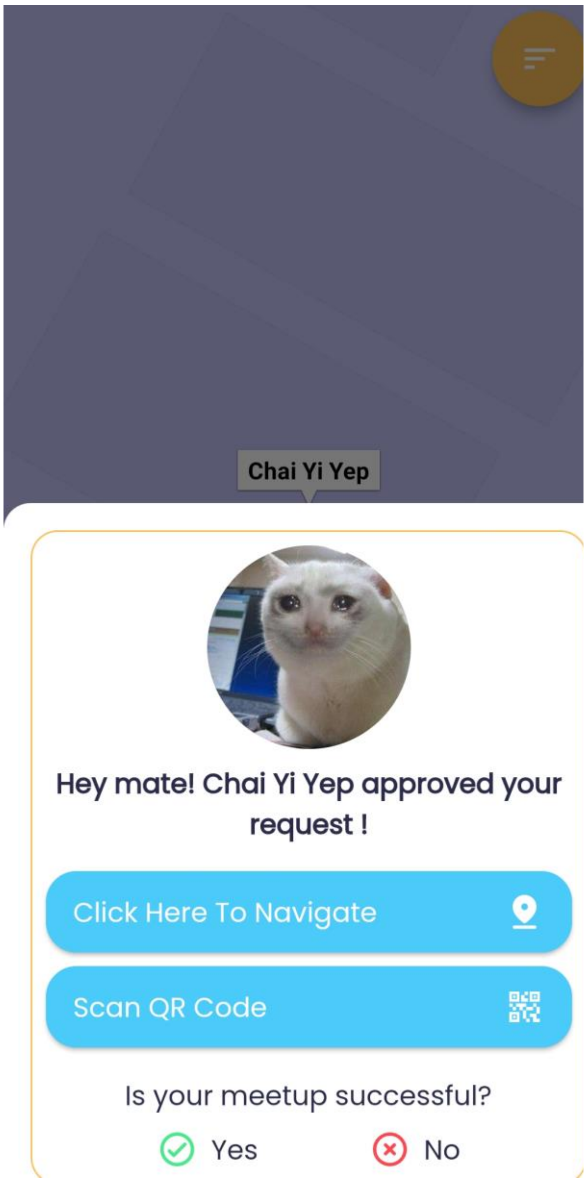
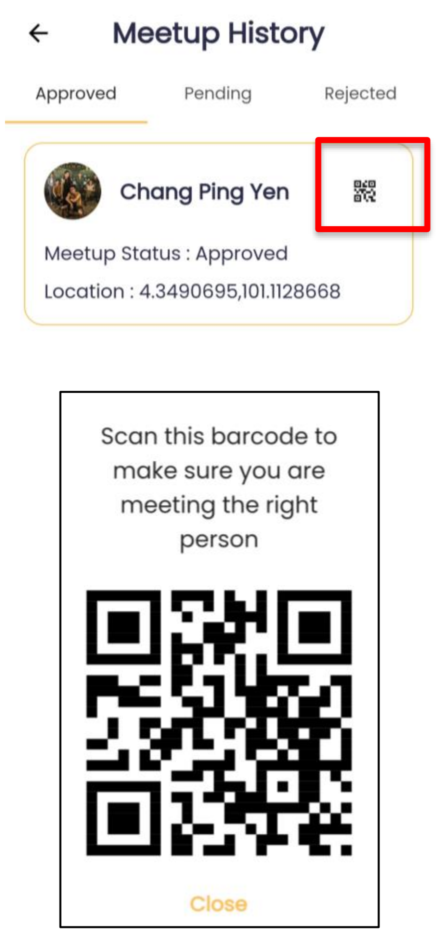

Chai Yi Yep
Hey mate! Chai Yi Yep approved your request !
Click Here To Navigate
Scan QR Code
Is your meetup successful?
Yes
No

Figure 5.3.6 Navigate to User Location / Scan QR Code / Yes,No Option

Once Chang has reached Chai's location, to determine whether Chang is meeting the correct person, Chang can click on the bottom button to scan a QR Code provided by Chai. The QR code is on the meetup history page under Approved requests. At the Approved requests panel, Chai will see the information of the request he approved and a QR code symbol on the right. A QR code will appear after clicking on the QR code symbol, as shown in Figure 5.3.7. Then, Chang can point their phone camera and scan the QR as shown in Figure 5.3.8.


Meetup History

Approved Pending Rejected

Chang Ping Yen

Meetup Status : Approved

Location : 4.3490695,101.1128668

Scan this barcode to make sure you are meeting the right person

Close

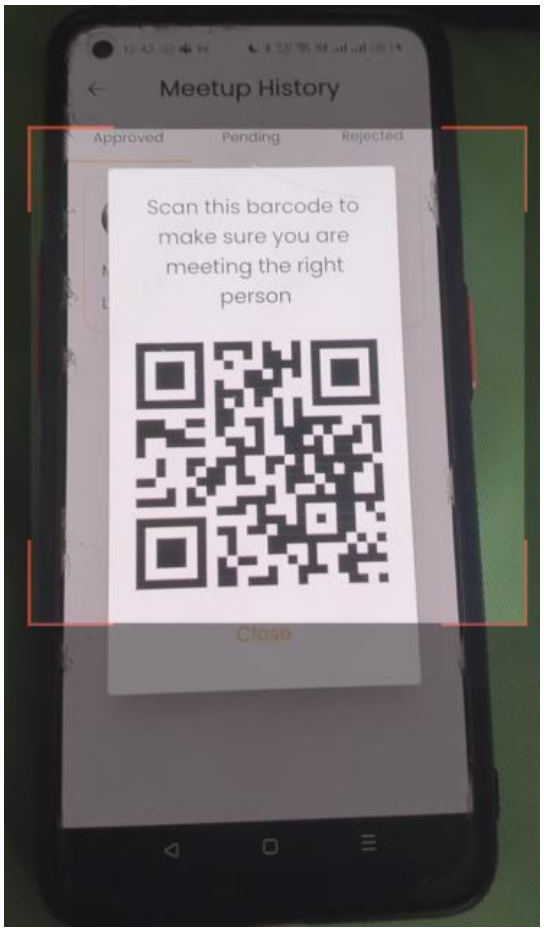
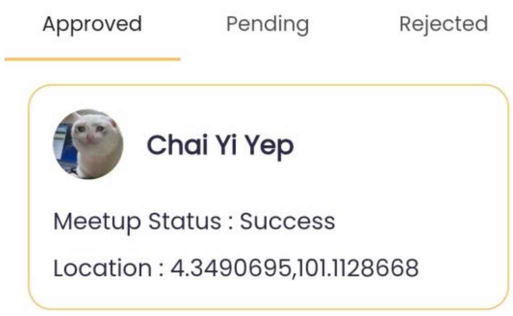

Meetup History
Approved
Pending
Rejected
Scan this barcode to make sure you are meeting the right person
Close

Figure 5.3.7 QR Code for Approved Request *Figure 5.3.8 Chang attempt to scan QR Code*

Lastly, Chang needs to update the app whether the meetup was successful or not by clicking on 'Yes' or 'No' as shown in Figure 5.3.6. This is a crucial step as even though a request is approved, the meetup status will only stay at approved and only after Chang clicks on 'Yes', the meetup status will be updated to 'Success' as shown in Figure 5.3.9.


Approved Pending Rejected

Chai Yi Yep

Meetup Status : Success

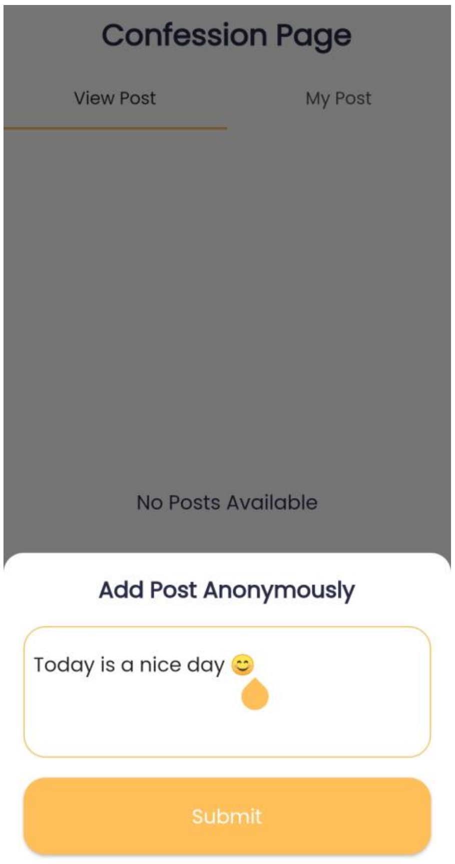
Location : 4.3490695,101.1128668

Figure 5.3.9 Meetup Status Success

<u>Forum Module</u>

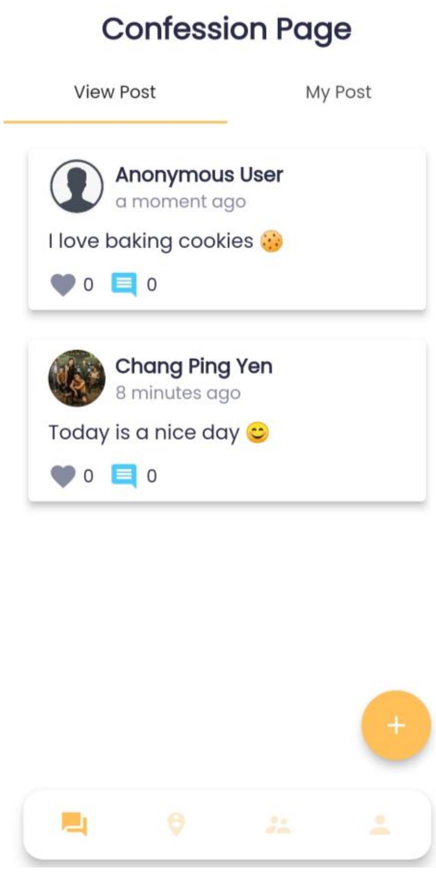
5.4 Add and Delete Forum Post

Users can create anonymous forum posts by clicking on the "+" button in the bottom right corner, and a dialog box will appear and request the user to type in their post as shown in Figure 5.4.1. Then, the user can click Submit, and the post will be posted on the Confession Page, as shown in Figure 5.4.2. Users can view their posts with their profile picture and name in the View Post panel. However, the current user can only view an anonymous profile picture and name for other users' posts. It is to protect the identity of all users as the purpose of the confession page is to allow students to speak up anonymously without concern about privacy issues. Users have 2 options when viewing the forum posts and they can choose to either view all user posts or view their post by clicking on the "My Post" panel, which will show a list of posts made by the current user. For each post created, there would be a timestamp to indicate when the post was made, and the order of the post is based on the latest time posted. It is a way to display new posts at the top, whereas old posts at the bottom.



Today is a nice day 😊

Figure 5.4.1 Add Forum Post



Anonymous User
a moment ago
I love baking cookies 🍪
0 0

Chang Ping Yen
8 minutes ago
Today is a nice day 😊
0 0

Figure 5.4.2 Confession Page View (Different user posts)

Users can also delete their posts under 'My Post' panel, where user can see their own posts and a red delete icon on the post, as shown in Figure 5.4.3. Once users click the delete icon, the post will be deleted, and a message 'deleted' will appear to indicate that post deletion is successful as shown in Figure 5.4.4.

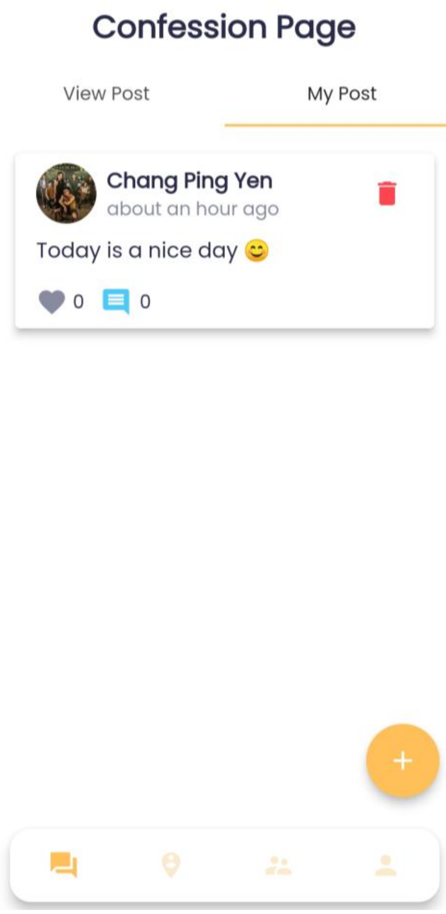


Figure 5.4.3 My Post Panel

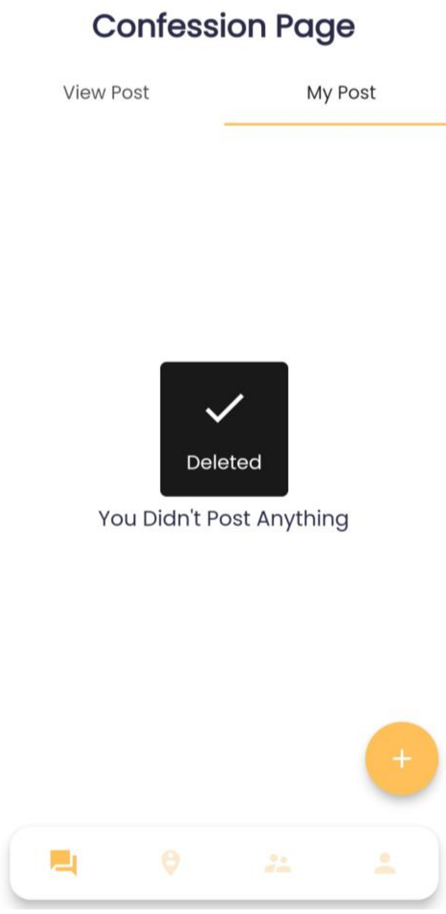


Figure 5.4.4 Delete Post

5.5 Comment and Like Forum Post

Users can comment on other users' posts or their own posts by clicking on a post, then click on the comment icon and there will be a dialog box will appear and request the user to type in their comment as shown in Figure 5.5.1. Then, the user can click Ok, and the comment will be posted under the post, as shown in Figure 5.5.2. Users can also like the post by clicking on the "♥" icon, and the like count will increase by 1.

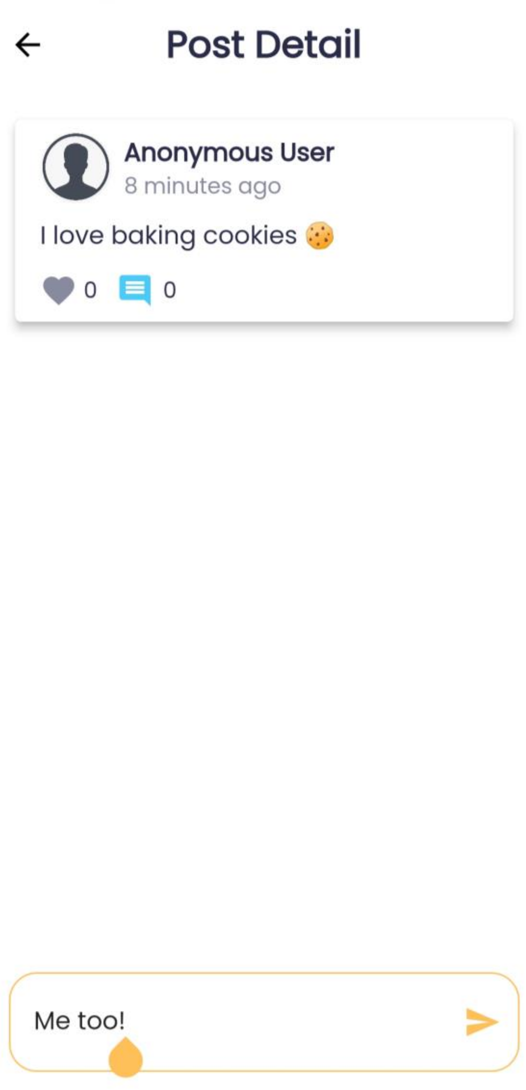


Figure 5.5.1 Add Comment

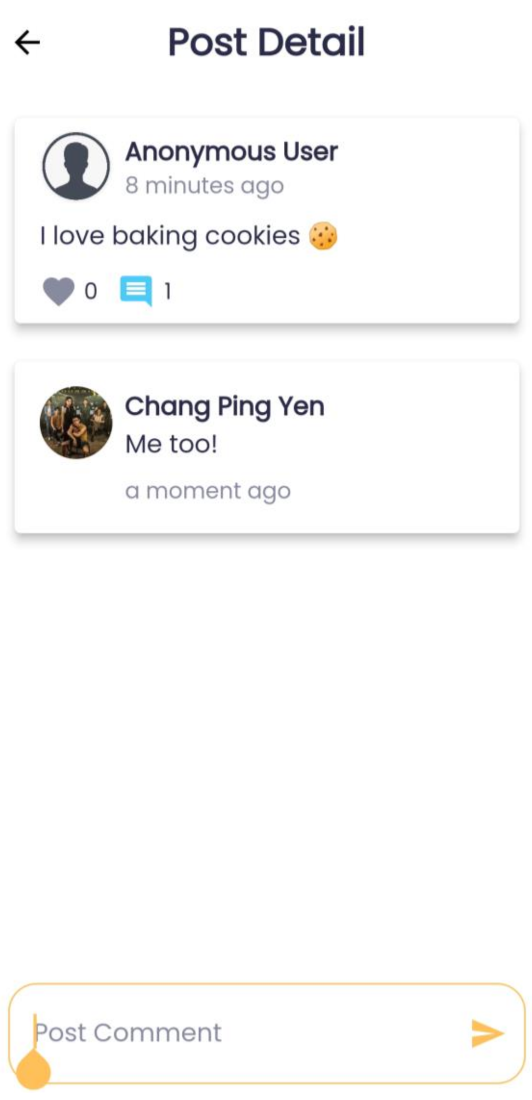
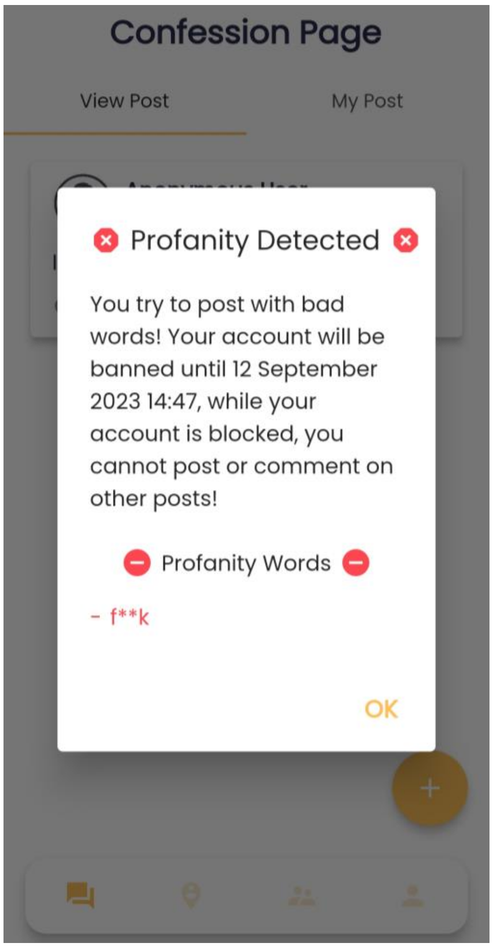


Figure 5.5.2 Comment Posted Under Post

5.6 Profanity Detector

Although users can create posts and comment anonymously, they will need to obey community guidelines, which prohibit the use of profanity. The profanity detector will detect profanity language, and if a user is found to be using profanity language in their post/comment, they will be banned temporarily, as shown in Figure 5.6.1. Each ban lasts for 10 minutes, and the maximum banning time is 5 hours. If users reach 5 hours time ban, their account will be banned permanently, meaning they will be unable to use the account. Figure 5.6.2 shows the user attempting to make another post while still under ban, thus stopping the user from creating a post.

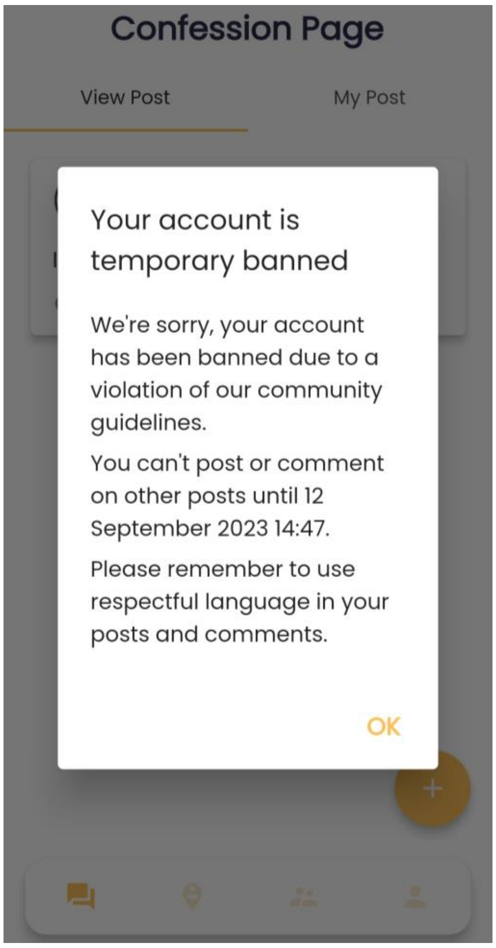


You try to post with bad words! Your account will be banned until 12 September 2023 14:47, while your account is blocked, you cannot post or comment on other posts!

Profanity Words

- f**k

Figure 5.6.1 Profanity Detection



Your account is temporary banned

We're sorry, your account has been banned due to a violation of our community guidelines.

You can't post or comment on other posts until 12 September 2023 14:47.

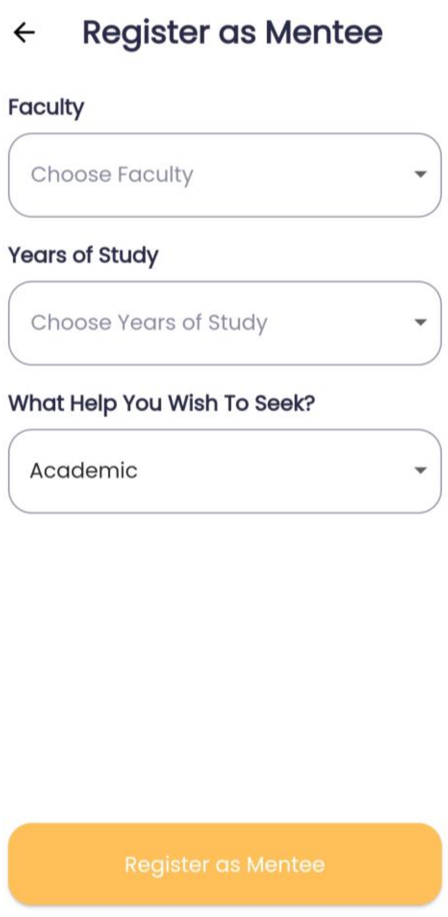
Please remember to use respectful language in your posts and comments.

Figure 5.6.2 User under temporary ban

Peer Mentor Module

5.7 Register Mentor / Mentee

Once the student reaches the Peer Mentor panel, they can choose to register as a mentor or mentee. Regardless of choosing mentor or mentee option, the user must fill in the information as shown in Figure 5.7.1. To register as a mentor, there is an extra requirement, which is that the years of study must be at least 1 year. It is to ensure the mentors are qualified enough to guide the mentees. If a mentor chooses the Less than 1 year option, the app will prompt an error message as shown in Figure 5.7.2. Furthermore, all details must be filled in, and any empty fields will prompt an error message. After filling in the details, the app will lead the mentor/mentee to their respective views.



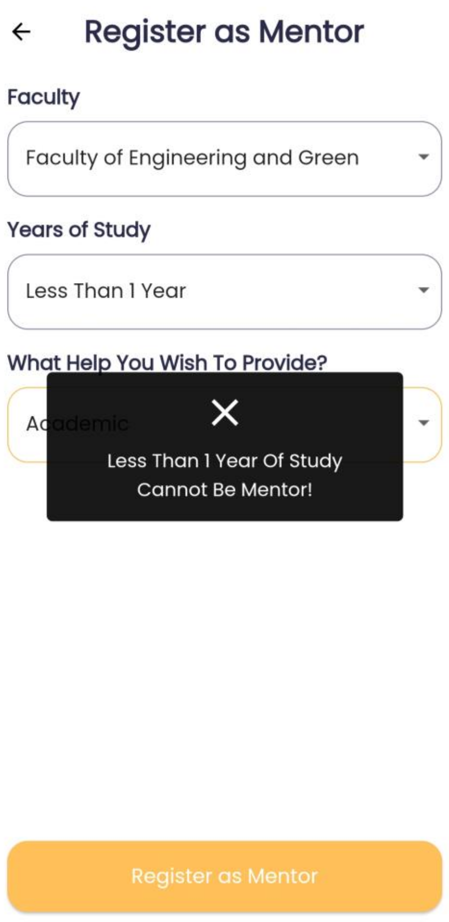
Register as Mentee

Faculty: Choose Faculty

Years of Study: Choose Years of Study

What Help You Wish To Seek?: Academic

Figure 5.7.1 Fill in details for registration



Register as Mentor

Faculty: Faculty of Engineering and Green

Years of Study: Less Than 1 Year

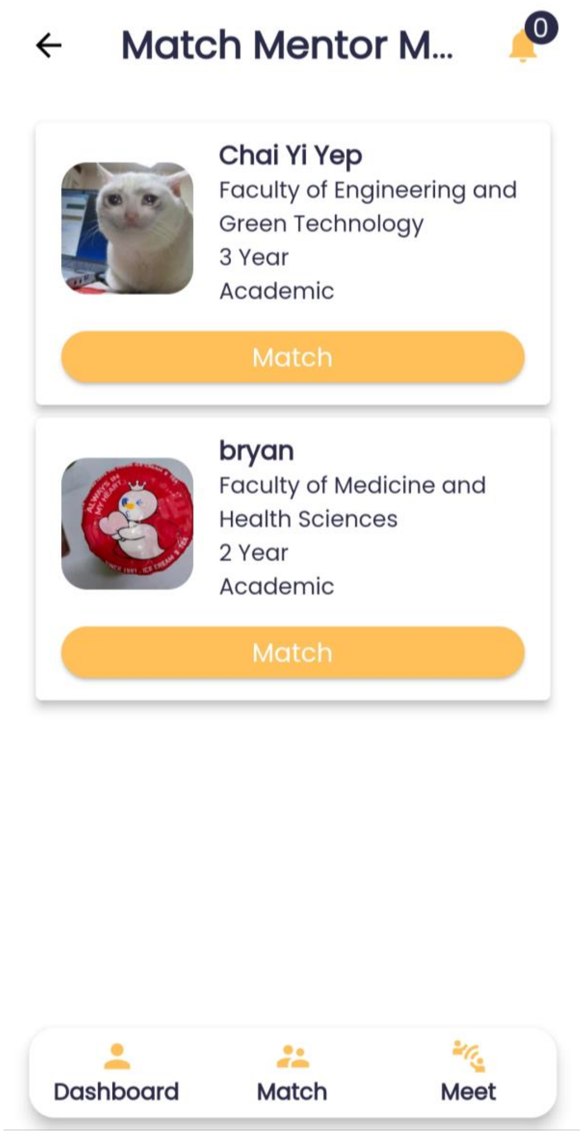
What Help You Wish To Provide?: Academic

Less Than 1 Year Of Study Cannot Be Mentor!

Figure 5.7.2 Error register as mentor

5.8 Match Mentor / Mentee

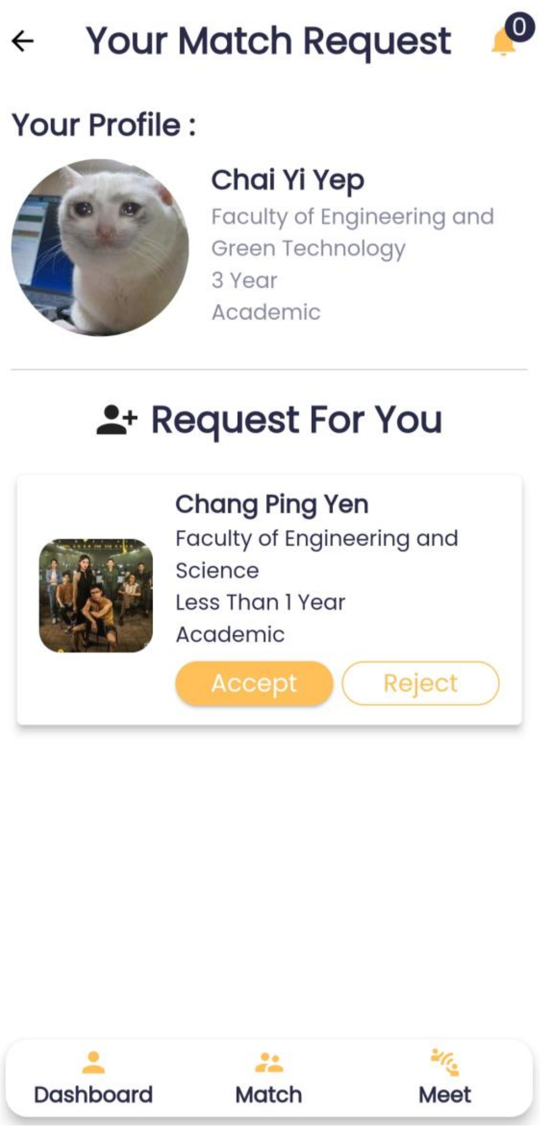
After successfully registering as a mentee, mentee can click on the 'Match' panel where a list of mentors will be available for matching as shown in Figure 5.8.1. Based on the mentee's preference, he/she can choose a mentor to match by clicking on the 'Match' button, which will then trigger the app to send a request to the specific mentor as shown in Figure 5.8.2. Mentor can choose to accept or reject the mentee and if mentor rejects a mentee, he/she needs to write down the reason of rejection.



Chai Yi Yep
Faculty of Engineering and Green Technology
3 Year
Academic

bryan
Faculty of Medicine and Health Sciences
2 Year
Academic

Figure 5.8.1 Match View for Mentee



Your Profile :

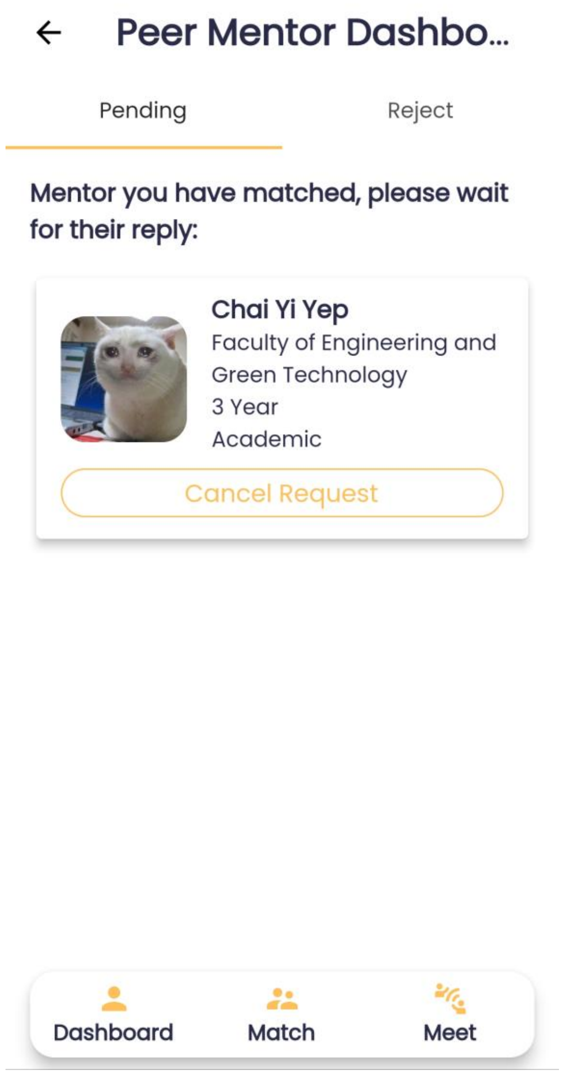
Chai Yi Yep
Faculty of Engineering and Green Technology
3 Year
Academic

Request For You

Chang Ping Yen
Faculty of Engineering and Science
Less Than 1 Year
Academic

Figure 5.8.2 Match View for mentor

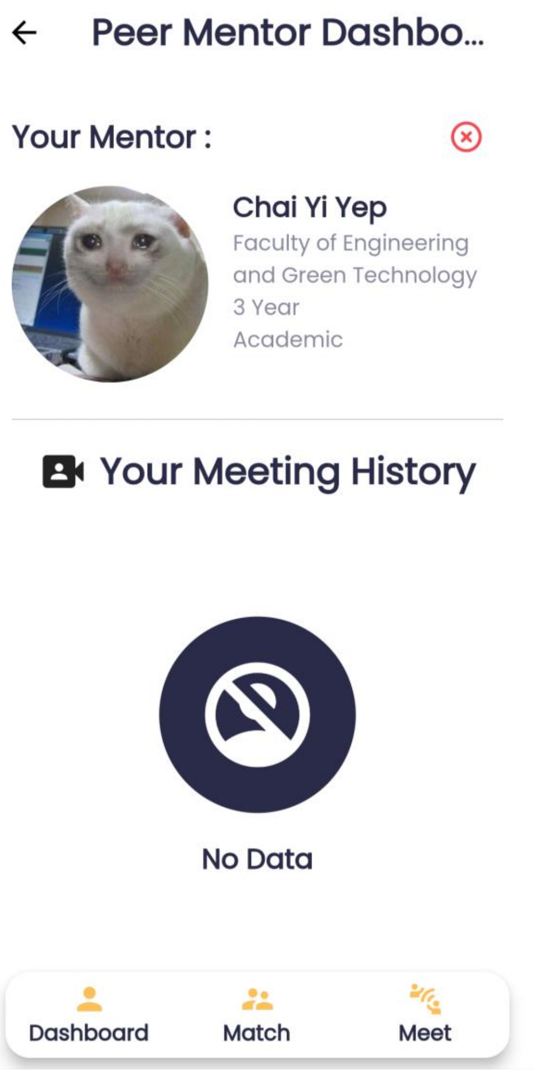
Before a mentor accepts / reject a mentee, the mentee can view the pending requests at the 'Dashboard' panel for mentee view as shown in Figure 5.8.3. Once the mentor accepts the request, the 'Dashboard' will immediately switch the view, displaying only the meeting history between mentor and mentee as shown in Figure 5.8.4.



Mentor you have matched, please wait for their reply:

Chai Yi Yep
Faculty of Engineering and Green Technology
3 Year
Academic

Figure 5.8.3 Dashboard View for Mentee (Before assigned mentor)



Chai Yi Yep
Faculty of Engineering and Green Technology
3 Year
Academic

Figure 5.8.4 Dashboard View for Mentee (After assigned mentor)

5.9 Schedule Meeting

Mentor and mentee can schedule meetings under the 'Meet' panel with a 'Schedule New Meeting' button as shown in Figure 5.9.1. After clicking the button, the app will prompt the mentor/mentee to fill in the required details, such as meeting name, date, time, and type of meeting, to create a new meeting as shown in Figure 5.9.2. If users fail to fill in all required details, the app will prompt an error message, and the user cannot create a new meeting.

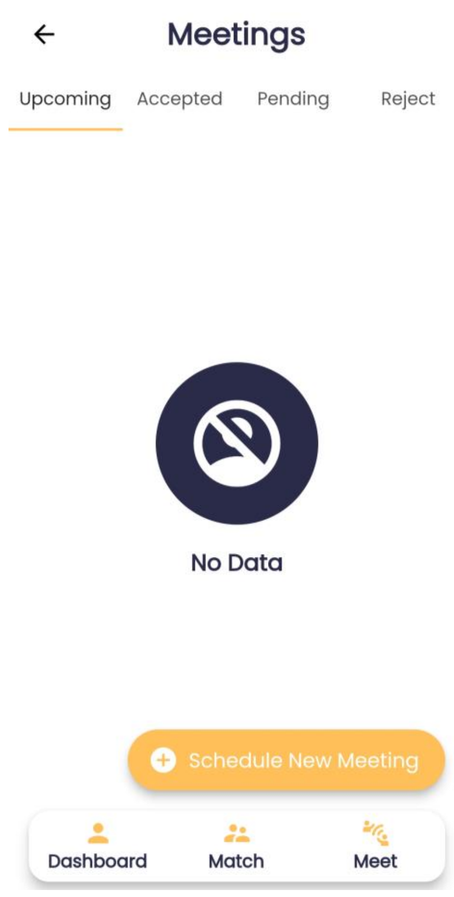
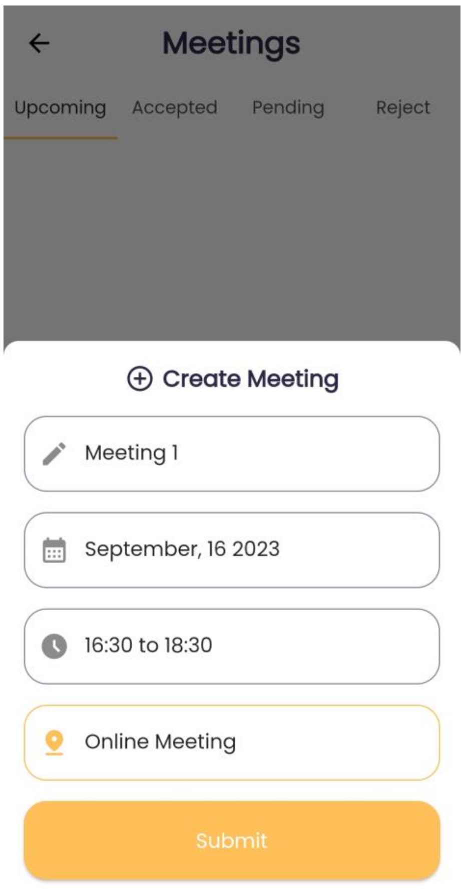


Figure 5.9.1 'Meet' panel



Meeting 1

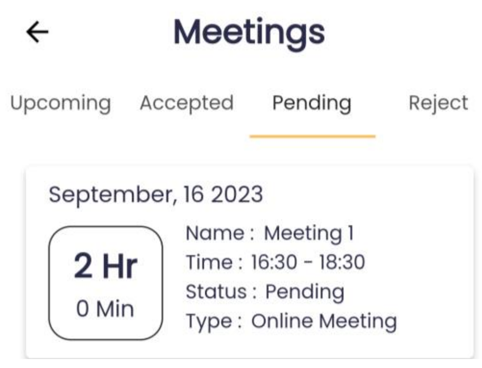
September, 16 2023

16:30 to 18:30

Online Meeting

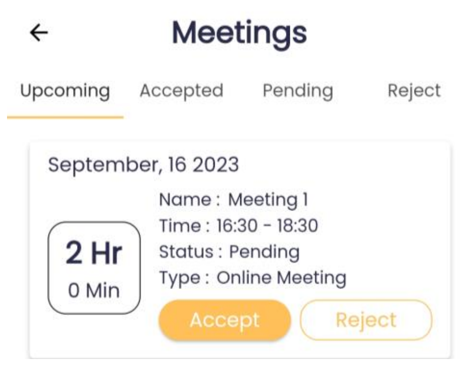
Figure 5.9.2 Fill in details to create new meeting

For the user who created the meeting, the meeting will be under the pending panel waiting for the other user to reply to the request, whereas the user who receives the meeting request will have the meeting under the upcoming panel where he/she gets to decide whether to accept or reject the request as shown in Figure 5.9.3 and Figure 5.9.4. Similarly, if the user chooses to reject the request, he/she will need to write down the reason for rejection.



September, 16 2023
2 Hr
0 Min
Name : Meeting 1
Time : 16:30 - 18:30
Status : Pending
Type : Online Meeting

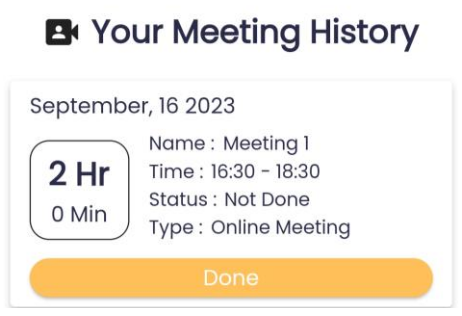
Figure 5.9.3 Pending meeting request



September, 16 2023
2 Hr
0 Min
Name : Meeting 1
Time : 16:30 - 18:30
Status : Pending
Type : Online Meeting

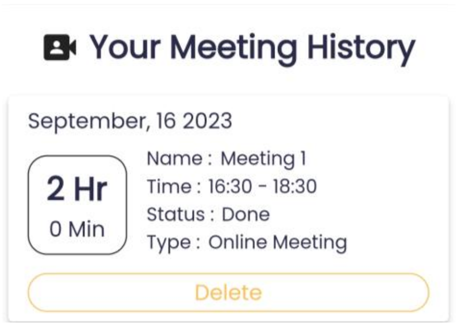
Figure 5.9.4 Upcoming meeting request

After a request is accepted, it will appear in the 'Dashboard' panel as meeting history. Either mentor or mentor can click the 'Done' button to indicate the meeting was conducted. Then, the button will turn into a delete button, which allows users to delete the meeting history if it is no longer needed. The purpose of displaying the meeting history under the dashboard is to serves as a record for users to look back on.



September, 16 2023
2 Hr
0 Min
Name : Meeting 1
Time : 16:30 - 18:30
Status : Not Done
Type : Online Meeting

Figure 5.9.5 Meeting not done

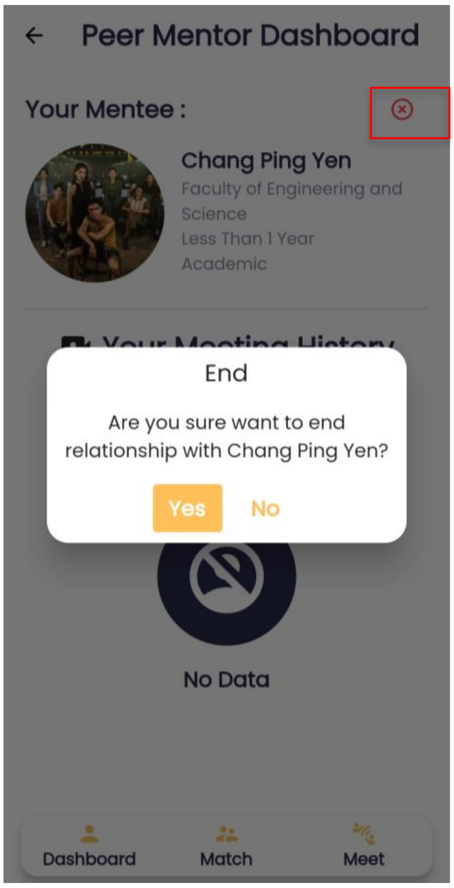


September, 16 2023
2 Hr
0 Min
Name : Meeting 1
Time : 16:30 - 18:30
Status : Done
Type : Online Meeting

Figure 5.9.6 Completed meeting

5.10 Terminate Mentor Mentee Relationship

Mentor and mentee can also terminate their mentor mentee relationship by clicking on the red 'x' on the top right corner and there will be a dialog box prompted out asking the mentor/mentee whether to confirm ending the relationship as shown in Figure 5.10.1. For those who wish to end the relationship, he/she is required to write down the reason of termination so that the other party may know about it as shown in Figure 5.10.2. For users who have been terminated from the relationship, he/she will receive a termination notification which can be found on their 'Match' panel.



Peer Mentor Dashboard

Your Mentee :

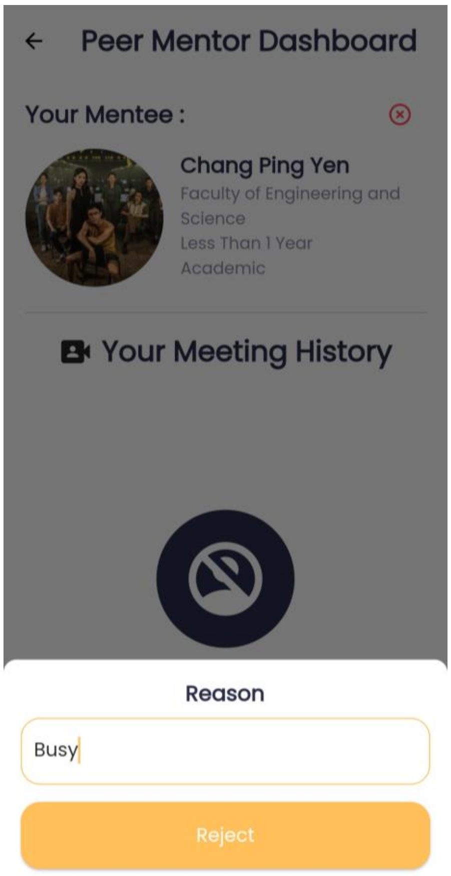
Chang Ping Yen
Faculty of Engineering and Science
Less Than 1 Year
Academic

Your Meeting History

End

Are you sure want to end relationship with Chang Ping Yen?

Figure 5.10.1 Terminate Relationship



Peer Mentor Dashboard

Your Mentee :

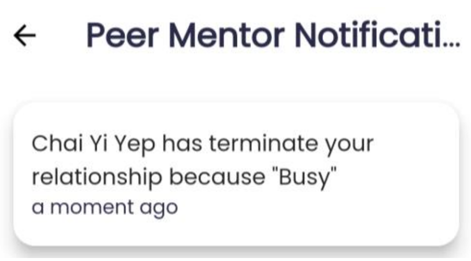
Chang Ping Yen
Faculty of Engineering and Science
Less Than 1 Year
Academic

Your Meeting History

Reason

Busy

Figure 5.10.2 Reason of Termination



Peer Mentor Notificati...

Chai Yi Yep has terminate your relationship because "Busy"
a moment ago

Figure 5.10.3 Termination Notification

<u>**Upload User Profile Module**</u>

5.11 Upload Profile Photo

Users can upload their profile photo by clicking on the pencil icon, and the app will direct them to their gallery to pick a photo. Once the user has chosen a photo, the image will appear on the page, and users will need to click on the upload button to store the image in the database.

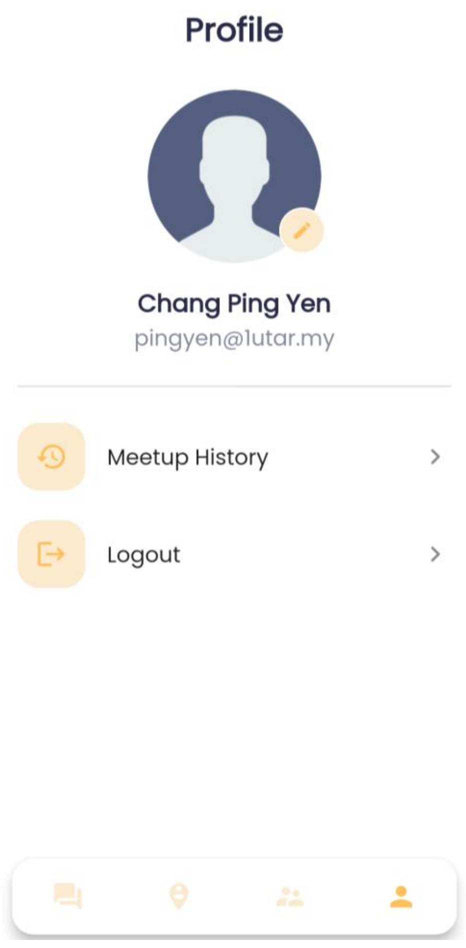


Figure 5.11.1 Upload Profile Photo UI

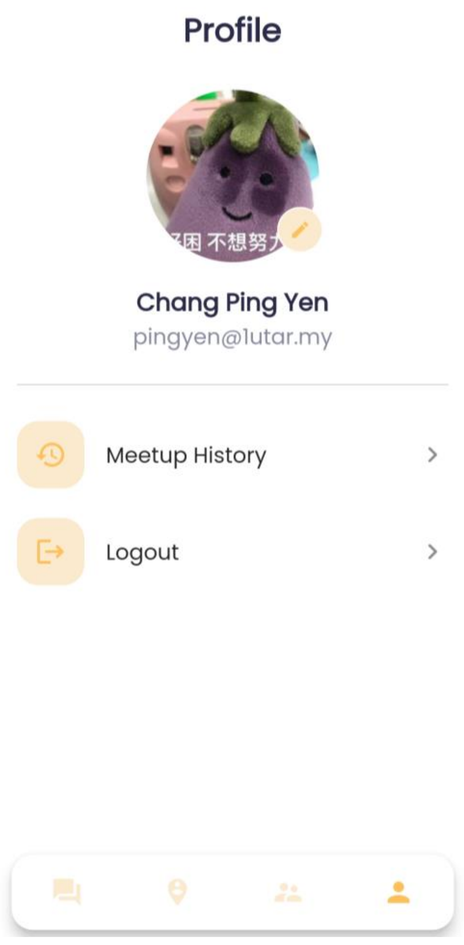


Figure 5.11.2 Image Chosen

<u>Admin Module</u>

5.12 Admin Home Page

Unlike normal users, admin does not require creating an account to log in to the app as the admin has their credentials given by the university authority. Upon logging into the admin account, the admin will first reach the home page, also known as 'Admin Page', where the admin can view all the posts made by students, as shown in Figure 5.12.1. Admin can view the identity of users for all posts and comments for moderation purposes. If an admin wants to view the comments of a particular post, admin can click on the post, and the app will lead the admin to another page with only the details of the certain post as shown in Figure 5.12.2. Besides, the admin can delete any post by simply clicking on the delete icon on the top right corner of each post, and the post will be removed from the database.

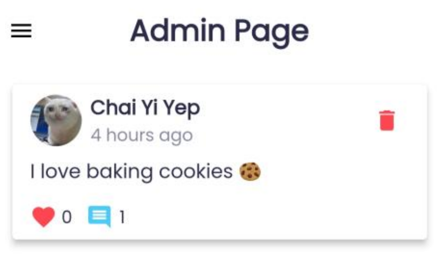


Figure 5.12.1 Admin Home Page (View Forum Post)

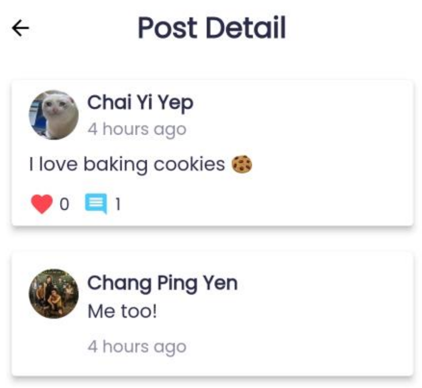
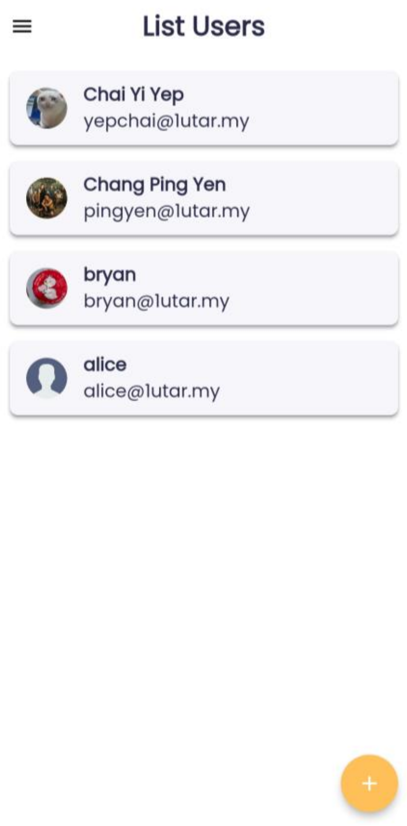


Figure 5.12.2 View Post Details

5.13 Admin Users Page

Moving on, under the admin module there is a 'Users' page where the admin can view a list of users showing their full name and email, as shown in Figure 5.13.1. In addition, the admin can create a new user by clicking on the '+' button at the bottom right corner, and the app will route the admin to a new page where the admin needs to fill in all the relevant details to create a new user as shown in Figure 5.13.2. Admin must fill in all the required information, as failing to do so will make creating a user unsuccessful.



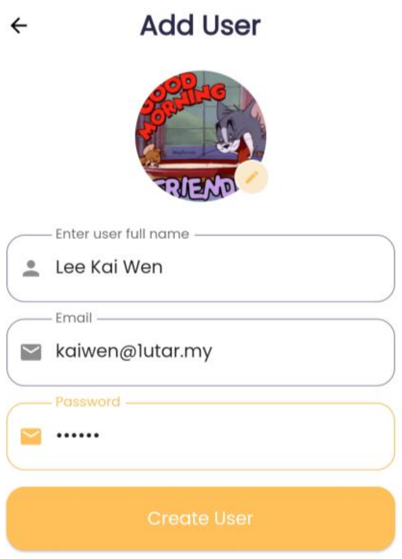
Chai Yi Yep
yepchai@1utar.my

Chang Ping Yen
pingyen@1utar.my

bryan
bryan@1utar.my

alice
alice@1utar.my

Figure 5.13.1 List of users



Enter user full name: Lee Kai Wen

Email: kaiwen@1utar.my

Password: ••••••

Figure 5.13.2 Add new user

When a new user is created, there will be a dialog box to indicate a new user has been successfully created, and the new user will be added to the top of the user list as shown in Figure 5.13.3. The arrangement of the user list is based on when users created the account. The newer accounts will stay at the top, whereas the older accounts will stay at the bottom. Next, the admin can edit user details, such as changing the username and profile picture of the user. After the admin has made the changes, the admin clicks on the 'Update User' to update the user's information in the database.

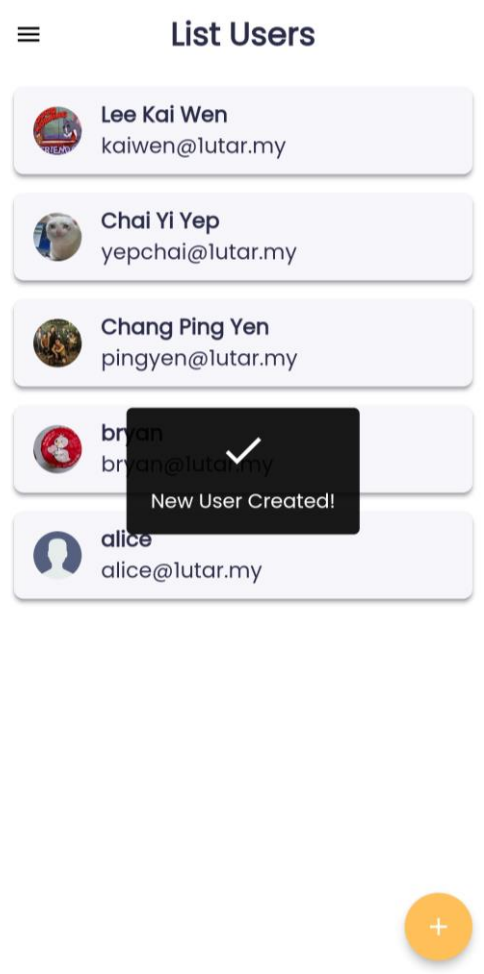


Figure 5.13.3 List of users (New user added)

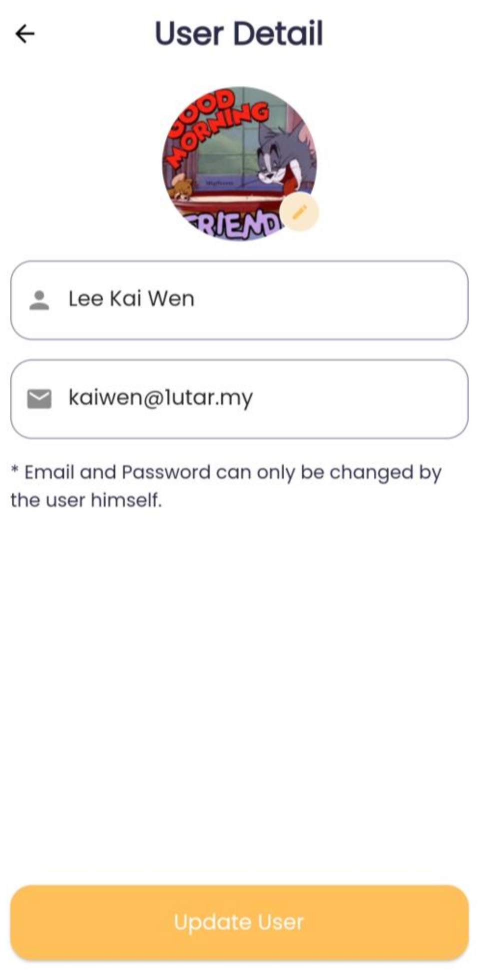
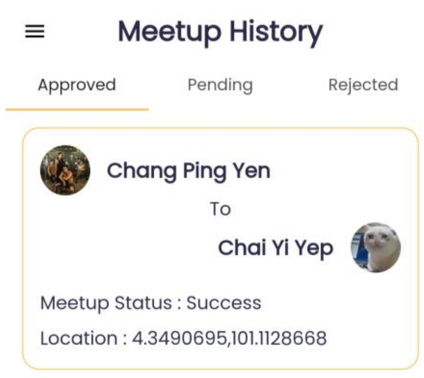

User Detail
Lee Kai Wen
kaiwen@1utar.my
* Email and Password can only be changed by the user himself.
Update User

Figure 5.13.4 Update user details

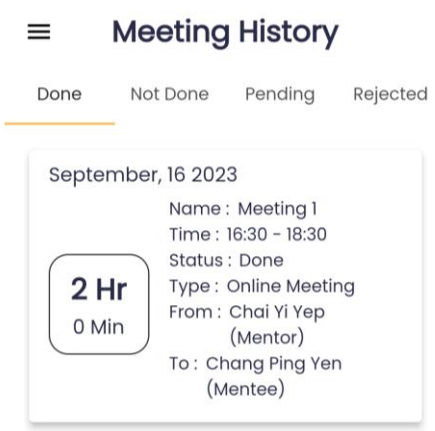
Moreover, the admin can view the meetup history of users in the 'Meetup History' page where there will be 3 panels: Approved, Pending and Rejected meetups. Figure 5.13.5 shows an approved meetup history from the user Chang Ping Yen to another user Chai Yi Yep. The history details include the username, meetup status and location coordinates. Similar to the 'Meetup History' page, there is a 'Meeting History' page where the admin can view all meeting history between mentor and mentees. Under the meeting history, there would be details showing the request made by which user (in this case, user Chai Yi Yep) sent to another user (in this case, Chang Ping Yen).



Meetup Status : Success

Location : 4.3490695,101.1128668

Figure 5.13.5 Meetup History

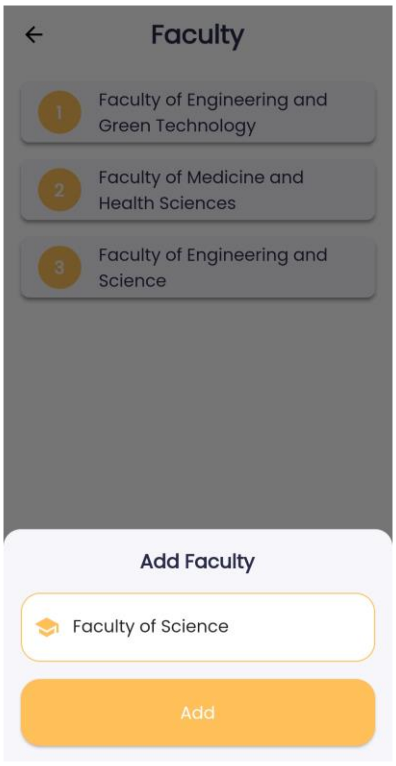


September, 16 2023

Name : Meeting 1
Time : 16:30 - 18:30
Status : Done
Type : Online Meeting
From : Chai Yi Yep (Mentor)
To : Chang Ping Yen (Mentee)

Figure 5.13.6 Meeting History

Another useful function that the admin can use is the configure setting function. The admin can add options for the faculty and years of study which is needed when student register for mentor/mentee. Figure 5.13.7 and Figure 5.13.8 shows the admin adding a faculty option with the name 'Faculty of Science' and after clicking on the 'Add' button, the faculty has been successfully added.


Faculty
1 Faculty of Engineering and Green Technology
2 Faculty of Medicine and Health Sciences
3 Faculty of Engineering and Science
Add Faculty
Faculty of Science
Add

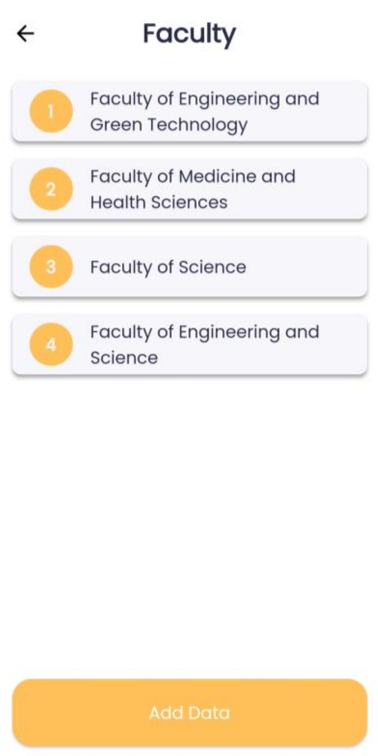

Faculty
1 Faculty of Engineering and Green Technology
2 Faculty of Medicine and Health Sciences
3 Faculty of Science
4 Faculty of Engineering and Science
Add Data

Figure 5.13.7 Add Faculty *Figure 5.13.8 Faculty successfully added*

If the admin wants to edit or delete an option, the admin can click on the particular option, and there would be 2 buttons: the 'Delete' and 'Edit' buttons as shown in Figure 5.13.9. When the admin clicks on the 'Delete' button, the particular faculty option will be deleted, and a message will appear to indicate the faculty option is successfully deleted as shown in Figure 5.13.10. For editing the option, the admin can just edit the field and later click on the 'Edit' button to update the edited information.

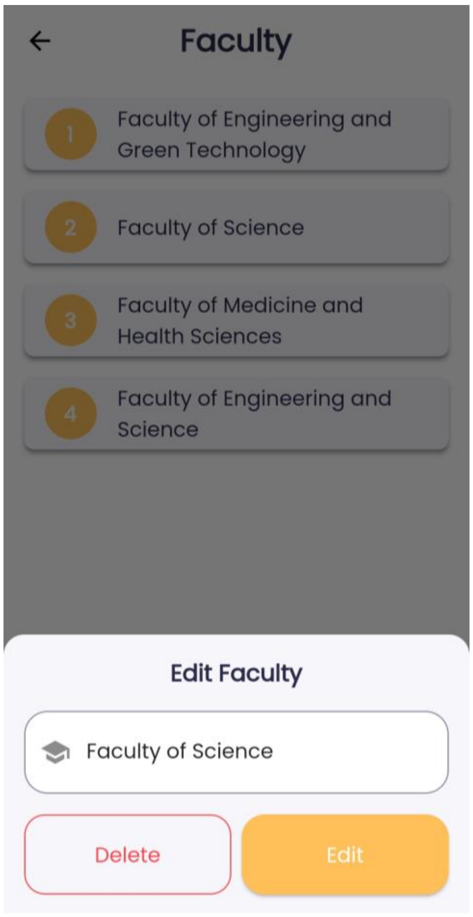

Faculty
1 Faculty of Engineering and Green Technology
2 Faculty of Science
3 Faculty of Medicine and Health Sciences
4 Faculty of Engineering and Science
Edit Faculty
Faculty of Science
Delete
Edit

Figure 5.13.9 Delete Faculty

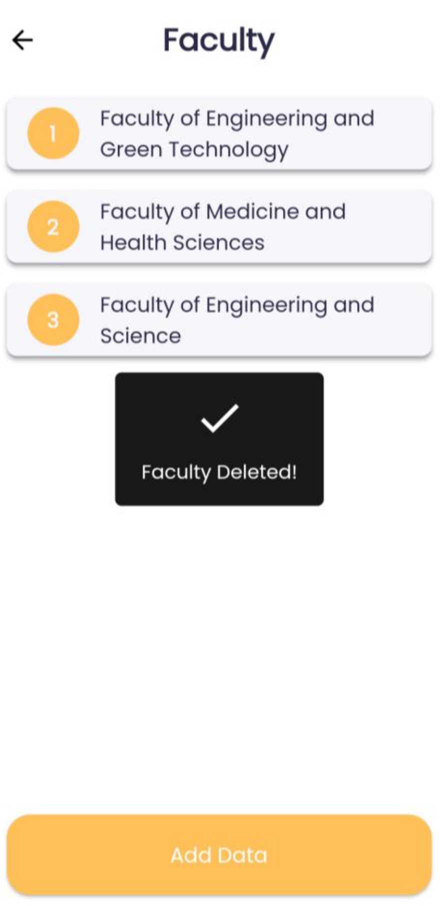

Faculty
1 Faculty of Engineering and Green Technology
2 Faculty of Medicine and Health Sciences
3 Faculty of Engineering and Science
Faculty Deleted!
Add Data

Figure 5.13.10 Faculty successfully deleted

For the years of study option, the add and delete functions are similar to the faculty option. The only difference is the edit function. When the admin wishes to edit the years of study option, the admin not only can edit the option's name but also set the position of the option and decide whether the option is valid for the user to become a mentor, as shown in Figure 513.11. The reason for having extra configurations is because the years of study option is important when registering as a mentor, and those who chose the option 'Less Than 1 Year' shall not be eligible to register as a mentor. After the admin is done editing, Figure 5.3.12 shows the arrangement of 3 Year has changed from the 4th position to the 5th position, indicating the update was successful.

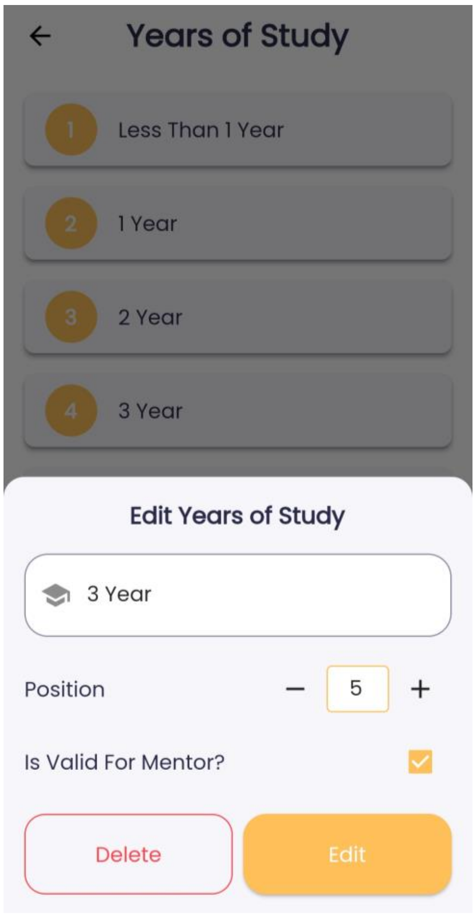


Figure 5.13.11 Edit Years of Study

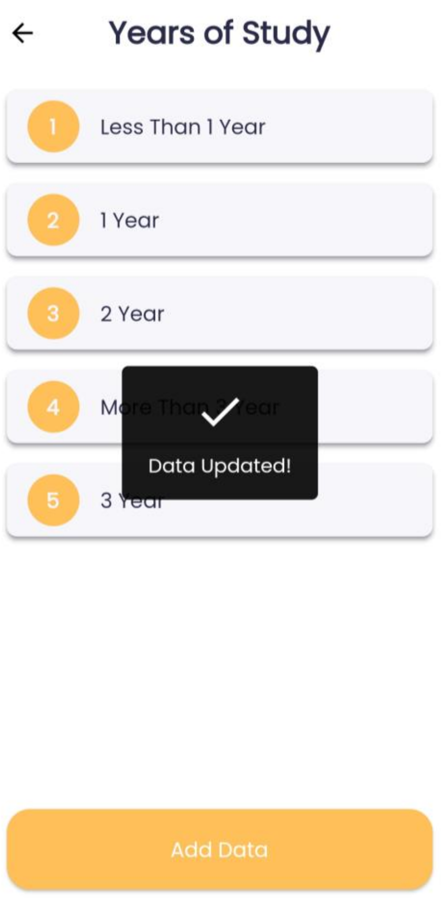
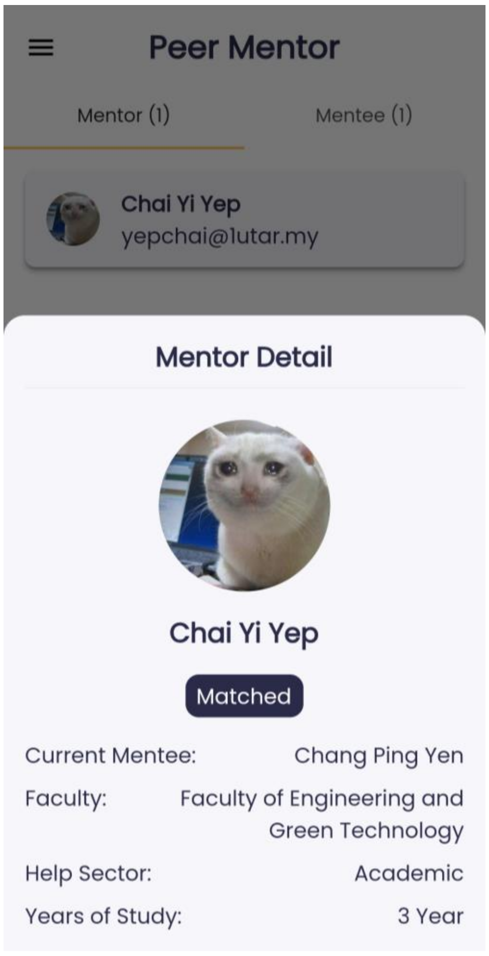


Figure 5.13.12 Edit Successful

Lastly, the admin can view the list of mentors and mentees on the 'Peer Mentor' page. There are 2 main panels which are 'Mentor' panel and 'Mentee' panel. Figure 5.13.13 and Figure 5.13.14 shows the details of mentor and mentee which includes information such as the current mentee/mentor, under which faculty, type of help needed/provided, and years of study.



Chai Yi Yep
yepchai@1utar.my

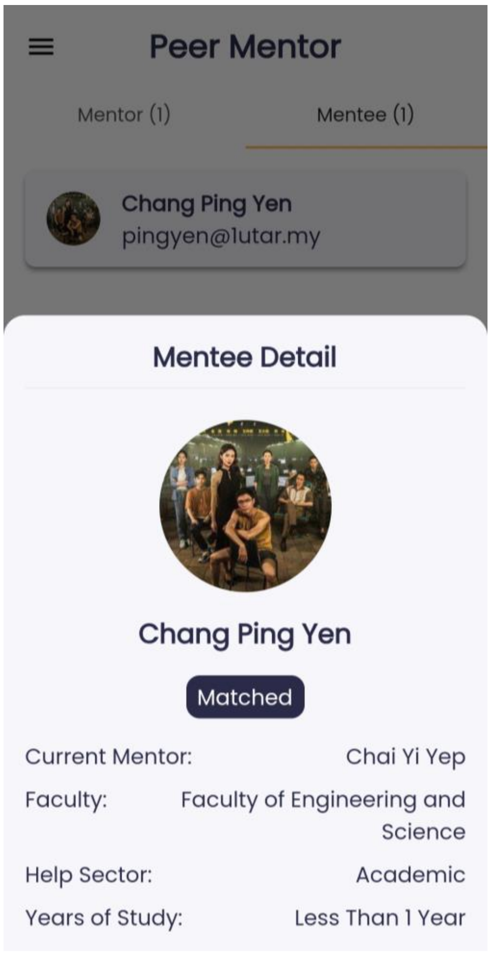
Mentor Detail

Chai Yi Yep

Matched

Current Mentee:	Chang Ping Yen
Faculty:	Faculty of Engineering and Green Technology
Help Sector:	Academic
Years of Study:	3 Year

Figure 5.13.13 Mentor Details



Chang Ping Yen
pingyen@1utar.my

Mentee Detail

Chang Ping Yen

Matched

Current Mentor:	Chai Yi Yep
Faculty:	Faculty of Engineering and Science
Help Sector:	Academic
Years of Study:	Less Than 1 Year

Figure 5.13.14 Mentee Details

5.14 Summary

In summary, the final work consists of modules, including signup/login, share location, forum, and peer mentor modules. These modules aim to tackle all the objectives mentioned, allowing students to locate peers nearby, providing a platform for anonymous information sharing and allowing students to participate in peer mentoring programs. The admin module is also developed to cater to user management activities.

Chapter 6 Conclusion

6.1 Project Review, Discussion and Conclusion

Community applications serve a great purpose by allowing users of similar interests to engage and connect with each other and share information. More and more universities are releasing their own community apps to enhance students' university experience. However, most of the university community apps in the market focus more on engaging students with their instructors and instructors with parents. Hence, the motivation to develop a university community app which focuses on student engagement arises.

The project aims to develop a user-friendly mobile application for UTAR students. Students can use the app to locate their peers near their compound, making it much easier to connect with their peers. Students can also express their thoughts by joining anonymous discussions and gaining more knowledge through content sharing. Plus, students can use the app to register for peer mentoring programs as mentors or mentees to create a more friendly and caring university environment. Even though students are the main users of the app, there will still be an admin to moderate and manage all user activity.

In conclusion, the modules developed have achieved all objectives mentioned, and if further enhanced, the app can be launched for university students to use. Throughout the process of developing the application, there are a number of issues faced. The first challenge is the issue of displaying the user location on the Google Maps. The user can get their current location at the app homepage and display it with a Google Maps marker by clicking the "Active Current Location" button. The issue arises as once users activate their current location, their location will still be active and display on the map even after they logout. It will confuse other users in the app as they will assume that the user is active and send a location-sharing request, but the user is actually inactive. The solution for this is to create a stream of active user data, whereby only those users who are logged into the app will have their location marker displayed. Furthermore, a similar bug appeared whereby the admin's location marker was displayed on Google Maps, and it took some time to find a way to filter the admin from normal users so that the admin's location marker would not appear on Google Maps.

The next challenge faced is integrating the profanity detector and account banning. The user will get banned whenever they try to make a post / comment containing profanity language, and if the user reaches a certain time limit, their account will be permanently banned. The issue faced is that even after the user has exceeded the maximum banned time (5 hours), they can still be logged into the account, though unable to post/comment. The issue lies within the logic of the code, where each time a user logs in, there is no conditional statement to validate whether the user is under banned, allowing the user to log in to the app even if he/she has exceeded the maximum banned time. To solve the issue, the app will first validate whether the user is under permanent ban each time a user logs into the app. If the app detects the user is under permanent ban, then the app will automatically force the user to logout of the app. Besides validating whether the user is under permanent ban, there are extra conditions such as validating whether the user is under temporary ban, user has been banned but the ban is lifted, and the user has not been banned before. By validating such conditions, the app will update the ban status of a user in the database accordingly.

6.2 Future Work

Certain enhancements can be made to this project. Firstly, the app currently only allows users to create text posts, while other forms of media, such as photos, videos, gifs etc., cannot be posted. It is considered a minor imperfection as many modern social media applications allow users to upload and post multiple media files at once. For future work, it is highly recommended to include the function of allowing users to include different media types in their posts to be more interactive. The same also applies to the comment section. However, to include other medias, such as photos and videos into a post, stricter moderation is required. There is a need to implement artificial intelligence to moderate content, as having a content moderator is not enough to moderate all user posts. It is suggested that involving artificial intelligence and human content moderators is 1 of the ways to detect and remove content that may be deemed inappropriate, obscene, violent, etc. Machine learning models are trained to identify inappropriate content in a media and decide whether to remove such content or let the content moderators for further review. Such enhancement would require a lot of work, such as collecting quality data and cleaning the data to train a model with high accuracy. However, such enhancement can be achievable if given enough time and effort.

Another enhancement that can be made is the implementation of notifications for the app. The current application only supports in-app notifications, whereby users can only see the notification when they are using the app. Once users log out of the app, they cannot receive any notification. Notifications are an important element of an app as it helps convey the latest updates happening in the app. For the current app, when a user's post is being liked or commented on, the app will not have any notification. It is another flaw as users will not know when another user has engaged with their post, and they can only know after clicking on the post to see themselves. Similarly, in the app, when a mentee has sent a match request to a mentor, the mentor cannot see the notification unless they are using the app. Therefore, it is necessary for a mobile application to implement push notifications so that the message within the app can be sent to the user's mobile device, which helps alert users about current updates.

REFERENCES

[1] H. Rheingold, "virtual community | Definition, Characteristics, Types, & Facts", Encyclopedia Britannica, 2021. [Online]. Available: https://www.britannica.com/topic/virtual-community. [Accessed: 26- Jul- 2022].

[2] "hi-hive Community", Play.google.com, 2022. [Online]. Available: https://play.google.com/store/apps/details?id=com.slc.hihive.community&hl=en&gl=US. [Accessed: 10- Aug- 2022].

[3] "Raklet - Sell Memberships Online", Raklet - Membership Management Software. [Online]. Available: https://www.raklet.com/. [Accessed: 12- Aug- 2022].

[4] "Homepage - Reddit", Redditinc.com. [Online]. Available: https://www.redditinc.com/. [Accessed: 27- Aug- 2022].

[5] D. Ltd, "How the Reddit Algorithm Works", Datadial.net, 2014. [Online]. Available: https://www.datadial.net/blog/how-the-reddit-algorithm-works/. [Accessed: 27- Aug- 2022].

[6] "Our Mission | InterNations", Internations.org. [Online]. Available: https://www.internations.org/about-internations/?ref=fo_ab. [Accessed: 28- Aug- 2022].

[7] "InterNations GmbH Reviews", reviews.io. [Online]. Available: https://www.reviews.io/company-reviews/store/internations-org/1?filters%5Brating%5D%5B0%5D=1. [Accessed: 28- Aug- 2022].

[8] "ZeeMee", ZeeMee. [Online]. Available: https://www.zeemee.com/. [Accessed: 29- Aug- 2022].

[9] "Documentation for visual studio code," Visual Studio Code, 03-Nov-2021. [Online]. Available: https://code.visualstudio.com/docs. [Accessed: 17-Apr-2023].

[10] "Run apps on the Android Emulator," Android Developers. [Online]. Available: https://developer.android.com/studio/run/emulator. [Accessed: 17-Apr-2023].

[11] "Learn Firebase fundamentals," Firebase Documentation. [Online]. Available: https://firebase.google.com/docs/guides. [Accessed: 17-Apr-2023].

[12] "Flutter documentation," Flutter. [Online]. Available: https://docs.flutter.dev/. [Accessed: 17-Apr-2023].

[13] "Dart overview," Dart. [Online]. Available: https://dart.dev/overview. [Accessed: 17-Apr-2023].

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T4Y3	**Study week no.: Week 2**
Student Name & ID: Chang Ping Yen (19ACB02846)	
Supervisor: Dr Tan Joi San	
Project Title: UTAR Community Application for Students	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Meet supervisor for discussion.

2. WORK TO BE DONE

- Revised on the report written for Final Year Project 1.
- Enhance the whole mobile application.

3. PROBLEMS ENCOUNTERED

- No problems encountered.

4. SELF EVALUATION OF THE PROGRESS

- Progress is still on track.

[illegible]

Supervisor's signature

yen

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T4Y3	**Study week no.: Week 4**
Student Name & ID: Chang Ping Yen (19ACB02846)	
Supervisor: Dr Tan Joi San	
Project Title: UTAR Community Application for Students	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Revised on the report written for Final Year Project 1.
- Revamp the whole mobile application to make it much more structured.
- List down all the remaining modules for the mobile application.

2. WORK TO BE DONE

- Continue revamp the app.
- Implement add on functions for share location function (Scan QR code / notification).
- Change the confession page layout (let students able to view their own profile picture and name instead of anonymous profile/name).
- Improve admin module by adding more panels.

3. PROBLEMS ENCOUNTERED

- No problems encountered.

4. SELF EVALUATION OF THE PROGRESS

- Progress is still on track.

[illegible]

Supervisor's signature

yen

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T4Y3	**Study week no.: Week 6**
Student Name & ID: Chang Ping Yen (19ACB02846)	
Supervisor: Dr Tan Joi San	
Project Title: UTAR Community Application for Students	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Add notification and scan QR code for share location function.
- User can view their own profile in forum post.
- Admin view user list and view forum post.
- Start on Peer Mentor module (Registration).
- Look for algorithms to implement in mobile app.

2. WORK TO BE DONE

- Peer Mentor module (Matching, Create meetings).
- Meetup history view (User and admin).
- Meeting history view (User and admin).
- Identify suitable algorithm to implement.

3. PROBLEMS ENCOUNTERED

- No problems encountered.

4. SELF EVALUATION OF THE PROGRESS

- Progress is still on track.

[illegible]

Supervisor's signature

yen

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: T4Y3	**Study week no.: Week 8**
Student Name & ID: Chang Ping Yen (19ACB02846)	
Supervisor: Dr Tan Joi San	
Project Title: UTAR Community Application for Students	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Peer Mentor module (Matching, Create meetings).
- Meetup history view (User and admin).
- Meeting history view (User and admin).
- Implement profanity detector.
- Meet supervisor for discussion.

2. WORK TO BE DONE

- Delete forum post (User and admin).
- Delete meeting history (User).
- Update user details (User and admin).
- Improve profanity filter to become a ban account function.

3. PROBLEMS ENCOUNTERED

- Profanity detector is not an algorithm, still need to think of another algorithm to implement.

4. SELF EVALUATION OF THE PROGRESS

- Progress is still on track.

[illegible] yen

Supervisor's signature Student's signature

Trimester, Year: T4Y3	**Study week no.: Week 10**
Student Name & ID: Chang Ping Yen (19ACB02846)	
Supervisor: Dr Tan Joi San	
Project Title: UTAR Community Application for Students	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Delete forum post (User and admin).
- Delete meeting history (User).
- Update user details (User and admin).
- Improve profanity filter to become a ban account function.

2. WORK TO BE DONE

- Implement radius filter to display users based on preference.
- Start writing FYP report.
- Meet supervisor for discussion.

3. PROBLEMS ENCOUNTERED

- Having difficulty implementing radius filter.

4. SELF EVALUATION OF THE PROGRESS

- Progress is still on track.

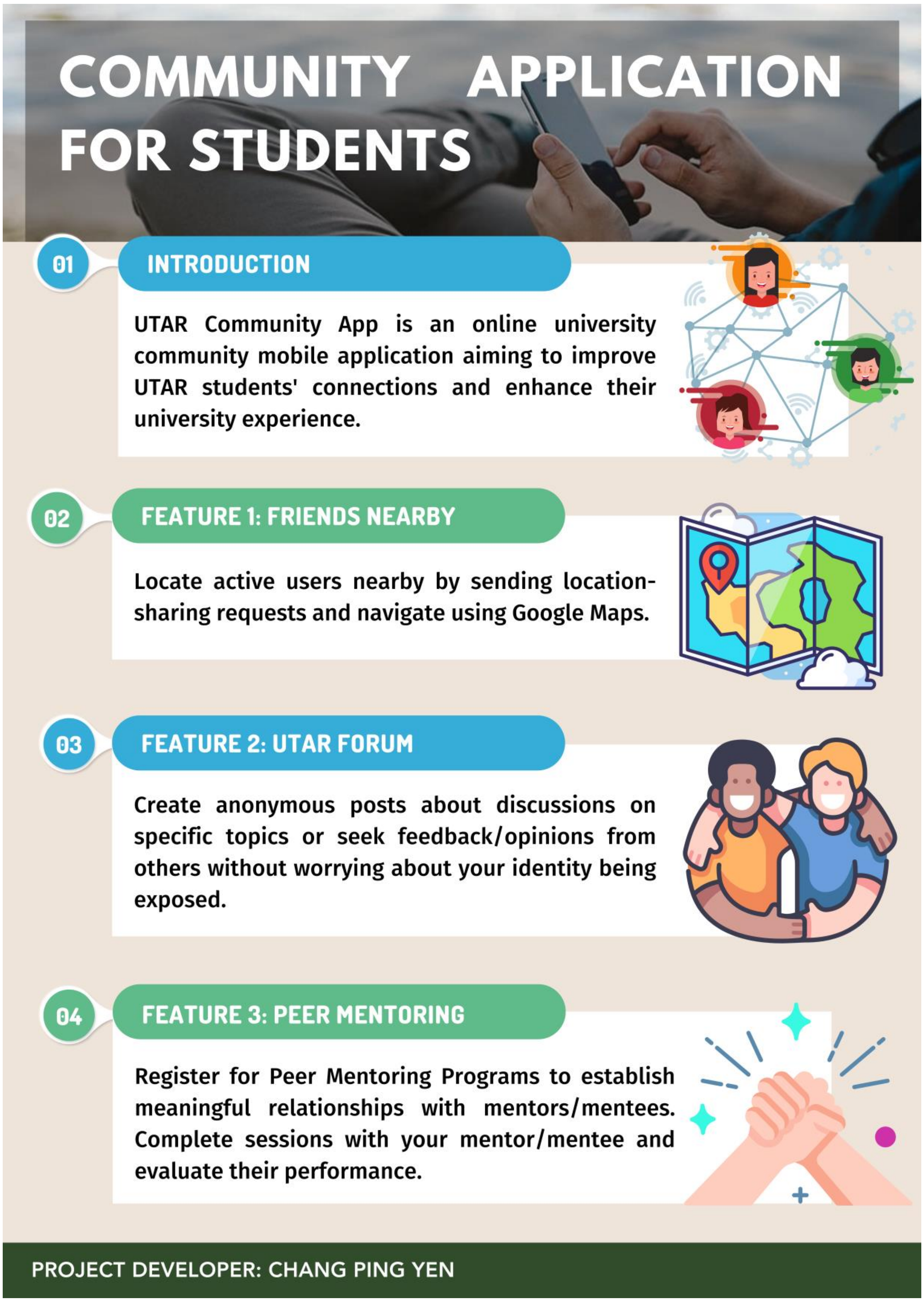
[illegible]

Supervisor's signature

yen

Student's signature

POSTER



COMMUNITY APPLICATION FOR STUDENTS

01 INTRODUCTION

UTAR Community App is an online university community mobile application aiming to improve UTAR students' connections and enhance their university experience.

02 FEATURE 1: FRIENDS NEARBY

Locate active users nearby by sending location-sharing requests and navigate using Google Maps.

03 FEATURE 2: UTAR FORUM

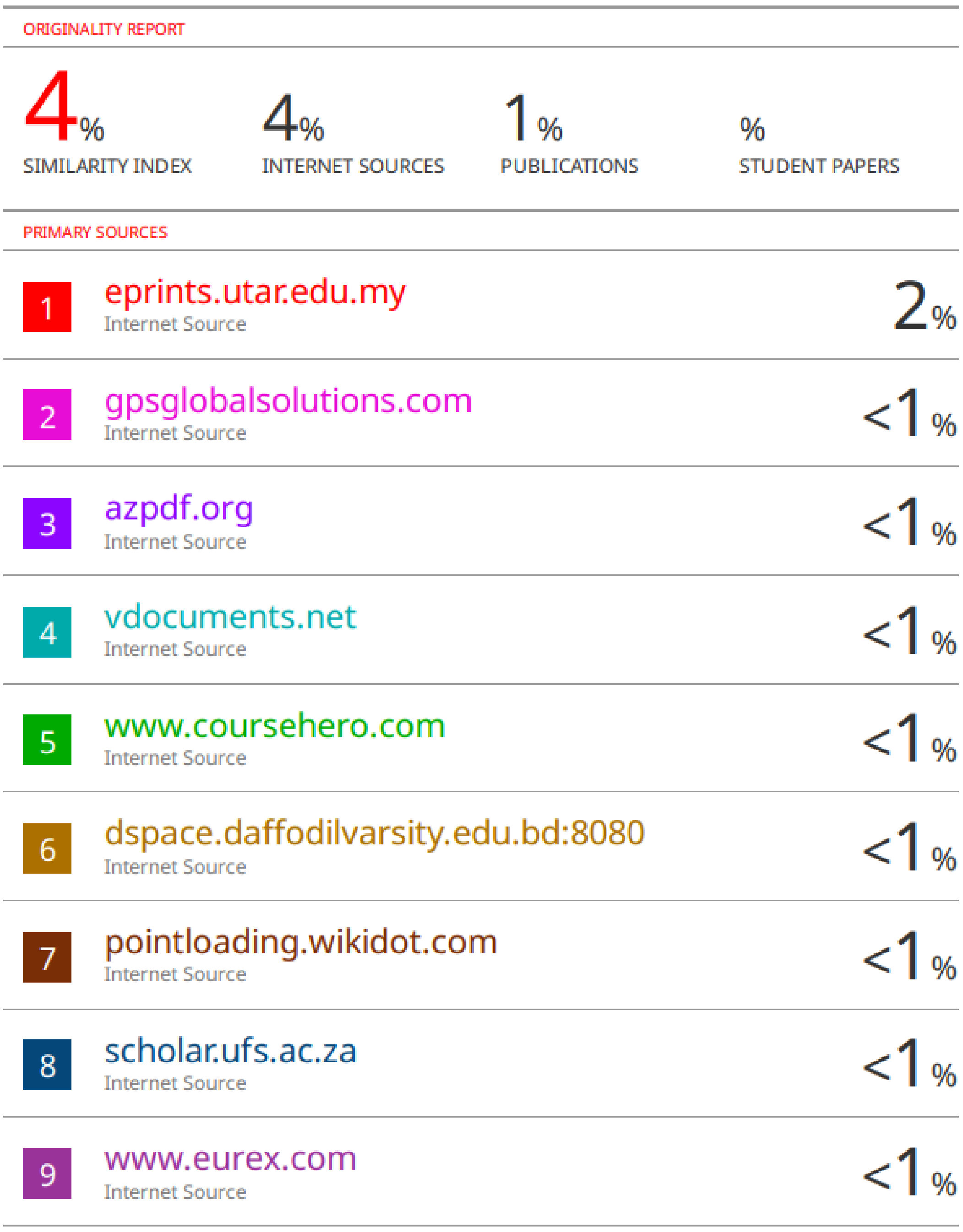
Create anonymous posts about discussions on specific topics or seek feedback/opinions from others without worrying about your identity being exposed.

04 FEATURE 3: PEER MENTORING

Register for Peer Mentoring Programs to establish meaningful relationships with mentors/mentees. Complete sessions with your mentor/mentee and evaluate their performance.

PROJECT DEVELOPER: CHANG PING YEN

PLAGIARISM CHECK RESULT



ORIGINALITY REPORT

4%	4%	1%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	eprints.utar.edu.my Internet Source	2%
2	gpsglobalsolutions.com Internet Source	<1%
3	azpdf.org Internet Source	<1%
4	vdocuments.net Internet Source	<1%
5	www.coursehero.com Internet Source	<1%
6	dspace.daffodilvarsity.edu.bd:8080 Internet Source	<1%
7	pointloading.wikidot.com Internet Source	<1%
8	scholar.ufs.ac.za Internet Source	<1%
9	www.eurex.com Internet Source	<1%

10	resources.fiorano.com Internet Source	<1 %
11	Lucas Jordan, Pieter Greyling. "Practical Android Projects", Springer Science and Business Media LLC, 2011 Publication	<1 %
12	www.udemy.com Internet Source	<1 %
13	developers.redhat.com Internet Source	<1 %
14	www.devicespecifications.com Internet Source	<1 %
15	researchonline.ljmu.ac.uk Internet Source	<1 %
16	www.hp.com Internet Source	<1 %
17	Isabelle Brocas, Juan D. Carrillo, Jorge Tarrasó. "Self-awareness of biases in time perception", Journal of Economic Behavior & Organization, 2018 Publication	<1 %
18	dokument.pub Internet Source	<1 %
19	hdl.handle.net Internet Source	<1 %

20	www.ktm2day.com Internet Source	<1 %
21	"Multi level Authentication for secure Attendance System", International Journal of Recent Technology and Engineering, 2020 Publication	<1 %
22	Muhammad Iqbal Pranatio, Ari Yanuar Ridwan, Mohammad Deni Akbar. "Design of Rice Logistic Geographic Information Systems Using SCOR, FMEA, and AHP Method to Support National Food Security Risk Mitigation System", 2022 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob), 2022 Publication	<1 %
23	dr.lib.iastate.edu Internet Source	<1 %
24	es.slideshare.net Internet Source	<1 %
25	mdpi-res.com Internet Source	<1 %
26	technodocbox.com Internet Source	<1 %
27	unina.stidue.net Internet Source	<1 %
	www.george.gov.za	
28	Internet Source	<1 %
29	www.nature.com Internet Source	<1 %

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	CHANG PING YEN
ID Number(s)	19ACB02846
Programme / Course	Bachelor Of Computer Science
Title of Final Year Project	UTAR Community Application for Students

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 4 % **Similarity by source** Internet Sources: 4 % Publications: 1 % Student Papers: 0 %	No comment
Number of individual sources listed of more than 3% similarity: 0	No comment

Parameters of originality required and limits approved by UTAR are as Follows:

(i) **Overall similarity index is 20% and below, and**
(ii) **Matching of individual sources listed must be less than 3% each, and**
(iii) **Matching texts in continuous block must not exceed 8 words**

Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

[illegible]

Signature of Supervisor	Signature of Co-Supervisor
Name: Tan Joi San	Name:
Date: 14 Sep 2023	Date:

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	19ACB02846
Student Name	Chang Ping Yen
Supervisor Name	Dr Tan Joi San

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
N/A	Front Plastic Cover (for hardcopy)
✓	Title Page
✓	Signed Report Status Declaration Form
✓	Signed FYP Thesis Submission Form
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
N/A	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
N/A	Appendices (if applicable)
✓	Weekly Log
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
✓	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

yen
____
(Signature of Student)
Date: 12/9/2023