ADVISEE-ADVISOR ONLINE COMMUNICATION PLATFORM

BY

CHIAM JIA YING

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)

JANUARY 2021

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: ADVISEE-ADVISOR ONLINE COMMUNICATION PLATFORM

Academic Session: JANUARY 2021

I CHIAM JIA YING
(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,

[illegible] [illegible]

(Author's signature) (Supervisor's signature)

Address:
20, LORONG HARMONI 1,
TAMAN BUKIT MAS,
14000 BUKIT MERTAJAM

Tan Joi San
Supervisor's name

Date: 15th APRIL 2021

Date: 15th April 2021

ADVISEE-ADVISOR ONLINE COMMUNICATION PLATFORM

By

Chiam Jia Ying

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)

JANUARY 2021

DECLARATION OF ORIGINALITY

I declare that this report entitled "**ADVISEE-ADVISOR ONLINE COMMUNICATION PLATFORM**" 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	:	[illegible]
Name	:	Chiam Jia Ying
Date	:	15th April 2021

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and sincere appreciation to my supervisor, Dr. Tan Joi San whom given me this opportunity to engage in this project and develop a system to ease the communication between advisee and advisor. She had guided me largely and gave me invaluable suggestions and encouragements throughout this project development process. This project would not be possible without the help from Dr. Tan. Besides, I would also like to thanks my moderator Dr. Jasmina Khaw for her comments of this project.

I also wish to express my sincere thanks to my friends whom help me in testing the chatbot and gave their feedback for future improvements. Additionally, I owe my gratitude to my family members. They had given me unwavering love and support for all these years throughout the course.

ABSTRACT

University Tunku Abdul Rahman (UTAR) provides a portal for advisor to add consultation notes during consultation meeting with the advisee. However, it is not efficient as advisees need to find their advisor privately to book a time with the advisor for consultation. The number of consultation meetings also added bulk to the advisor especially when advisees are asking repeating questions. Hence, this project titled “Advisee-advisor Online Communication Platform” is developed and serves to improve the existing academic advising system of UTAR by increasing the effectiveness of advisee-advisor communication and provide an alternative way to ease the communication between advisee and advisor by implementing various functionalities, including chatbot replies and analysis, calendar events, booking of meetings, and CGPA calculator and target CGPA calculator. This project is done based on Rapid Application Development (RAD) methodology, and it includes 4 phases, which are the requirements planning phase, user design, construction and feedback phase, and lastly the finalise product and implementation phase. For this project’s website development, it is built using HTML, CSS and JavaScript for front-end, and Node.js for server-side. Python is used to develop the chatbot, and for database, MySQL is used.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BoW	Bag-of-words
CGPA	Cumulative Grade Point Average
CSS	Cascading Style Sheets
DBMS	Database Management System
FAS	Faculty of Arts and Social Science
FBF	Faculty of Business and Finance
FICT	Faculty of Information and Communication Technology
FSc	Faculty of Science
GPA	Grade Point Average
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IAAS	Intelligent Advance Automated System
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
JWS	Java Web Server
MySQL	My Structured Query Languages
n.d.	No date
NLU	Natural Language Understanding
pp.	Pages
PWS	PHP Web Server
SDK	Software Development Kit
SDLC	Software Development Life Cycle
SPAS	Student Planning and Advisory System
UTAR	Universiti Tunku Abdul Rahman
UWI	University of the West Indies

CHAPTER 1 INTRODUCTION

1.1 Problem Statement

Academic advising is crucial for a student's success in university. However, traditional approaches of communication between advisee and advisor consists of some problems which will cause ineffective academic support and guidance to the advisee. The problems include: ineffective time booking with academic advisors, inconvenience in face-to-face consultations, repetition of same questions from different advisees and no data analysis on advisees' enquiries.

Academic advisors are not always in their office as they are also lecturers and tutors and therefore, they need to go to classes and teach students, monitor examinations and much more. Thus, advisees need to ask their academic advisors about their available time slot and book their time beforehand through email in order to meet them in their office. However, due to the tons of workloads of advisor, this will cause them often be busy and not checking on emails frequently. In addition, their email will also receive many other emails per day, such as upcoming activities in university, emails from other students of their classes, some promotion and spam emails. These problems will lead to advisor missing out the message of advisee's enquiry and therefore he could not help the advisee on time due to ineffective booking of time for consultation meetings.

Besides, advisee is required to go to his or her advisor's office in order to meet the advisor as traditional academic advising is often conducted face-to-face. However, some advisees are shy to speak out their problems during face-to-face consultation with their advisor. Hence, it then leads to ineffective communication between advisee-advisor. In addition, there are also some situations where advisor is out of station or on leave. Moreover, on 11 March 2020, Covid-19 outbreak has been proclaimed by World Health Organization as a pandemic (World Health Organization 2020) and it is still ongoing. It affected the whole world, universities' campuses are closed, and all teaching and learning are conducted online. Therefore, it is impossible for advisee to meet their advisors in campus face-to-face and thus the advisee-advisor meeting is affected.

In many universities, the number of students is increasing from year to year while the number of advisors remains the same and thus the ratio of advisees assigned to each advisor increases. Therefore, this leads to the advisor needs to handle more inquiries from the advisee. However, some general questions asked by the advisee are also frequently being asked by other advisees. As the consultation with advisee is always face-to-face and one-to-one, the advisor will then need to continuously repeating the same answer. It is very time consuming and mentally demanding as it requires patience for the advisor to answer the same general questions over and over again. Other than that, there are no system for advisors to collect the frequently asked questions by the advisee and analyzing of data for administration purpose.

1.2 Background and Motivation

Academic advising is defined as situations where the institution assigns a representative to help students by giving guidance and insight in academic, social or personal affairs in terms of notifying, recommending, counselling, disciplining, guiding, mentoring or teaching (Kuhn 2008). Academic advising is available in universities or college where each student is assigned to an academic advisor which has knowledge about their course once they start pursuing their studies at the institution.

Academic advisor plays an important role in guiding students during their university life, especially for those who just entered the university as they require guidance to familiarize themselves with the university's system. Students will often face problems in academic like planning course structure, examination and coursework issues, internship enquiries, final year project matter and much more. With the guidance of experienced academic advisors, they can provide informative suggestions to guide and lead the student to find his own pathway on universities' curriculum, social, future careers and even bring life aspirations. Based on 2019 National Student Satisfaction and Priorities Report, 86% - 92% of students from different categories of institutions rated academic advisor which has knowledge about the requirements in the major of them as one of the important key factors of the academic experience (Ruffalo Noel Levitz 2019). Moreover, students will be dissatisfied if they receive insufficient information or improper advice by the advisor as it might lead to dropout or delay of graduation due to wrongly rearrangement of course structure.

The main motivation of developing this project is due to the lack of functions provided for advisee and advisor in University Tunku Abdul Rahman (UTAR) portal. Advisor can only interact with advisee by adding consultation notes during the consultation session. However, there are no functions to communicate between advisee and advisor in the portal, such as booking of meetings and question enquiring. Hence, these provides motivation for the implementation of this project.

1.3 Project Objectives

In this project, the main objective is to develop a website to increase effectiveness in advisee-advisor communication and alleviate the workloads of advisors, and also to obtain the data of frequently asked questions by advisees.

i. To ease the booking of meeting time slot for consultation

This website allows advisee to view the timetable of their advisor by accessing the advisor's profile. The timetable shows the available timeslots slots of the advisor. The advisee can then book the time slot of their advisor when they see the timetable of the advisor is empty for the slot. Academic advisors no longer need to find meeting requests of their advisees one by one in email, and can now manage all the bookings through the website itself. Advisor can have a clear view of their own timetable whether there is a scheduled booking for meetings and with which advisee. In addition, advisor can also book a meeting with advisee with the same way in this website which is convenient especially when there is some important notice need to be given to the advisee, or the advisee have forgotten to file a booking for an academic advising session for the semester.

ii. To provide another alternative for advisees to ask about inquiries

Advisee can use the chatbot to help them in their inquiries. As some advisees are timid to voice out their problems, it is helpful for them as chatbots are not humans, and they can be more comfortable when chatting with the chatbots. The chatbot is also working 24/7 and it is convenient for advisee because they can ask inquiries and receive the solution whenever they want. It is also time efficient for advisees as they would not need to book time with their advisor and meet them face-to-face just to ask a few questions. If the chatbot could not answer the questions asked by the advisee, the chatbot will then recommend the advisee to chat with the advisor instead.

iii. To improve the efficiency of advisors in answering duplicated questions and data analysis in frequently asked questions of advisees

The chatbot will assists advisors by helping them to answer some general questions of the advisee which does not require decision making. By using the chatbot, the advisor can relieve some of their workload as they do not need to answer the general questions which are repeatedly asked by their advisees. Furthermore, the chatbot can help in analysing the data of most frequently asked questions and conclude in a chart in order to let advisors have a better overview of the questions asked. The data can also be used by the administration of the institute and they can post an announcement to inform advisee.

1.4 Project Scope

The scope of the project is to enhance the communication between advisee and advisor by providing them a user-friendly website as a platform to communicate. The features include login, advisee homepage, advisor homepage, admin homepage, GPA and CGPA calculator, chat with chatbot, adding, editing and delete events in calendar, and the account of advisor and advisee can only be added by the administrator.

i. Login module

The page needs to be login with either admin account, advisor account, or advisee account in order to proceed to the website, as the platform will only be available for the institutions' student and personnel. The admin, advisor and advisee have different privileges. Thus, after they login, different functionalities will be provided to them.

ii. Profile module

Users can view their own profile in the website. They can edit some of their personal information in their profile, and also view the profile of their advisees or advisor such as personal information, contact, and photo. Hence, it is convenient to find the contacts and information, and it is also helpful for advisee and advisor which has not met in person before meeting can get to know each other. In addition, advisor is able to view the credits hour obtained and the CGPA of the advisee.

iii. Event calendar module

Users will have a calendar in their profile where they can arrange their own schedule in the event calendar. They can add events, edit events and delete events in the calendar. Besides, advisees and advisor can view the calendar of the opposite side to know their available timeslot. Advisee and advisor can request booking of consultation meetings of the opposite side through their profile directly. When there is a request of booking received, it will not be shown in the calendar first, but in the manage booking section, and advisor or advisee will then manage the bookings and choose whether to accept or to reject the booking. If accepted, the meeting time can be seen by both the advisor and advisee in their own calendar.

iv. Meeting notes module

During consultation, there is a note function for advisor to write down the comments during the session and the comments are saved in the system after the session. Advisee and advisor can scroll back and view the comments from previous advising sessions, and anything to follow up since last consultation meeting. Hence, this module is useful to document the consultation session and provide future reference for both advisee and advisor.

v. Chatbot module

A chatbot is implemented in this project to help advisor to save some workload by helping to answer non-decision-making questions by the advisee. The chatbot is available for the advisee and can be accessed anytime. If there are questions that the chatbot does not understand, it will suggest the user to ask his advisor. The chatbot will also help to analyse the questions asked by the advisee, and most commonly asked questions is listed in the dashboard of administrator, and also enquiry analysis page of administrator and advisor.

vi. Cumulative Grade Point Average (CGPA) and target CGPA calculator module

There will also be a CGPA and target CGPA calculator for the advisor and advisee. Advisor can use the CGPA calculator to forecast the advisee's result and help to give advice on how to improve the results of the upcoming semester, and use the target CGPA calculator to know the average GPA needed to be obtained from the remaining credits hours in order to achieve the target CGPA. The advisee can also check on the CGPA and target CGPA calculator and set their own planning to obtain the result.

vii. Administration module

Administrator is able to add in new accounts of advisee and advisor. Administrator can then assign new advisee to the advisor and view the full list of advisor-advisee. Administrator can also add announcements which will be shown in the dashboard of the system.

1.5 Proposed Approach / Study

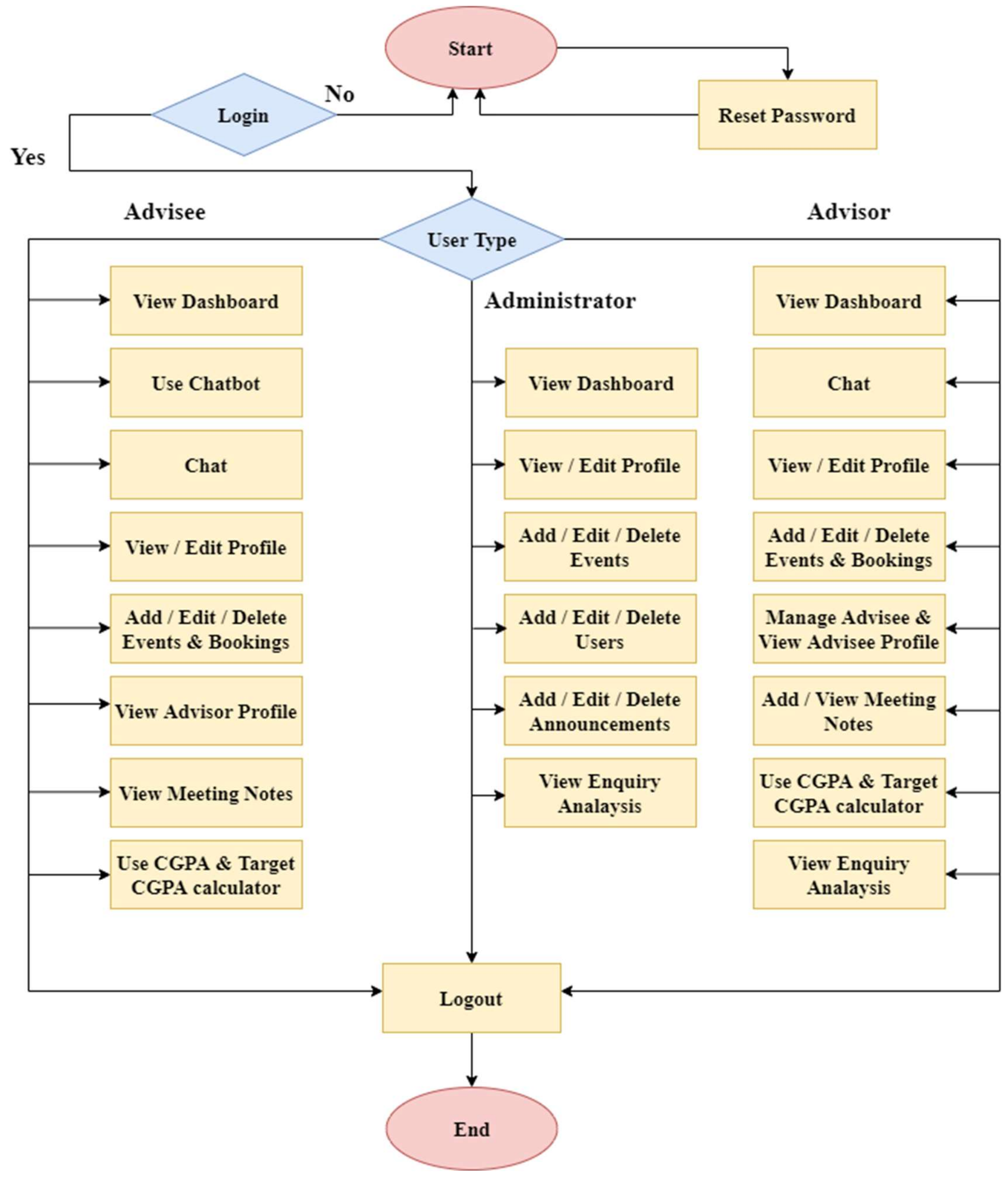


Figure 1.5.1 Flowchart of Advisee-advisor Online Communication Platform

A flowchart of the project is shown in Figure 1.5.1. The flowchart starts with 2 options, which is login, and reset password. Every user must login first in order to perform any actions. If the user has forgotten his password, user must reset the password of the account. If login unsuccessful, user will be prompt to login again. If successfully logged in, the user type of the account will be identified. Advisee, advisor and administrator can each perform different actions in the website. Advisee is able to view the dashboard once login, use the chatbot for enquiries and chat with advisor. In the profile page, users can also view and edit own profile, add, edit or delete events in the event calendar. In addition, advisee can also view the advisor profile page, add, edit or delete bookings in the calendar of advisor and view meeting notes. Furthermore, CGPA calculator and Target CGPA calculator can also be used by advisee. Advisor has almost same functions as advisees, however chatbot is not supported for advisor, and advisor have additional function to view all of his advisees and manage all of the bookings. On the other hand, administrator is able to view dashboard, view profile, edit profile, and add, edit and delete events in the event calendar too. In addition, administrator also has the right to add, edit and delete announcement and users.

1.6 Highlight of what have been achieved

Advisee-advisor Online Communication Platform is developed to provide an online platform for advisee-advisor to improve the communication quality and information transfer. One of the important features of this project is the event calendar. Every user will be having their own calendar in their profile, where they can update their own academic schedule or planning for reminders. In addition, advisee can view the calendar of their advisor through the platform and directly schedule a time to meet via the platform, and also vice versa. It is convenient as the available time of the opposite side is shown clearly at their calendar, and the process of schedule coordination is not required anymore. The personal details of user can also be viewed in the profile page. Besides, there is a meeting notes function where advisor can add notes and comments during the meeting. The meeting notes can then be saved as a reference for the advisee.

Furthermore, there is a chatbot developed in this website which can help the advisor to answer the general questions asked by advisees. The chatbot will then help to analyze the data of all the questions asked and will be shown in administrator’s dashboard, and also the enquiry analysis page. The data can then be used by administrator to improve the chatbot and the information posted in UTAR official website. Advisor can also view the most frequently asked question, and help to clarify the information to his advisees. The website also consists of a CGPA calculator and target CGPA calculator which can be used by both academic advisors and advisees for forecasting and planning purposes. Administrator, which is the highest level of hierarchy in this project, is able to manage all users, and also manage the announcements posted in the website. Administrator is able to perform create, read, update and delete users and announcements.

1.7 Report Organization

This report is separated into 6 chapters, and the information of each chapters are as following:

Chapter 1 is defining the problem statement and motivation to work on this project, and background of advisee-advisor is investigated. The project objectives are set according to the problem statement, and the modules in project scope are planned to achieve the objectives. All the achievements in this project are also listed here.

Chapter 2 is covering the literature review of similar projects and existing websites. The strength and weaknesses of each existing website are also listed and compared.

Chapter 3 consists of the use-cases diagram of the whole system. The use-case description of each use-case is also written. In addition, the code of website development and chatbot development is also explained.

Chapter 4 states the methodology, the tools and requirements used to build this project. The timeline of this project is also stated in this chapter.

In Chapter 5, the overview of the implementation of project is listed. The testing of chatbot and its feedback is also explained.

Lastly in Chapter 6, project review is made. The novelties and contribution made in this project is also highlighted. Improvements that can be made are listed in the future works in this chapter.

1.8 Summary

In this chapter, the definition and importance of academic advising is introduced. However, the traditional advisee-advisor communication in university such as UTAR is ineffective in booking time for consultation, inconvenience of face-to-face consultations, and increase of workload for advisee due to duplicated enquiries by advisee, and there is no data analysis of advisee's enquiry which can be used for administration purpose. Therefore, it is important to establish a platform to enhance the communication between advisee and advisor in order to improve the efficiency in supporting the advisee in academic matters. The project aims to refine the communication between advisee and advisor in terms of time, efficiency and convenience. 7 modules are developed in this project to achieve the objective in this project, which are login module, profile module, event calendar module, meeting notes module, chatbot module, CGPA and target CGPA calculator and administration module. The flowchart and what have been achieved in this project is also stated. In report organization, the overview of each of the chapters done in this project is listed.

CHAPTER 2 LITERATURE REVIEW

2.1 Overview

Similar projects and existing websites for advisee-advisor are reviewed in this chapter. Similar projects are reviewed to have an overview of the objectives of developing the project, the process of developing the project and to find out what are the achievements and limitations of the project, and the suggestions for future work. On the other hand, existing websites are also reviewed and the strength and weaknesses of each of them are also summarised for comparison.

2.2 Similar Projects

2.2.1 Student Planning and Advisory System (SPAS) (Kerk 2011)

SPAS was developed by a student in Universiti Tunku Abdul Rahman (UTAR) in 2011 with the objectives to bring UTAR an environment for e-learning, to increase the effectiveness of academic advising, platform for discussing career enquires, and systematical system for increasing work efficiency. It is planned to solve the problems faced by students (advisee), lecturers (advisor) and administrator of the university.

The system was developed using the waterfall model as its methodology, which included planning, analysis, design and implementation. During the planning phase, the author had set the title of the project, listed the background of the project issue, found the problem statement and also the objectives and scope, then set the timeline of the project using Gantt chart.

Next, the author had reviewed different techniques and tools for website development in the analysis phase and decided to use the authentication method for authenticating the login of users and giving permissions, a Database Management System (DBMS) for storing the data, and the programming language used to develop the system is PHP and used the Apache server which worked well together with PHP and yet is open-source.

The system, database, user interface and function integrations were then designed. There were three categories of functions designed, which are for students, lecturers and administrator respectively. The flow chart and functions of the system are listed out in Figure 2.2.1.1.

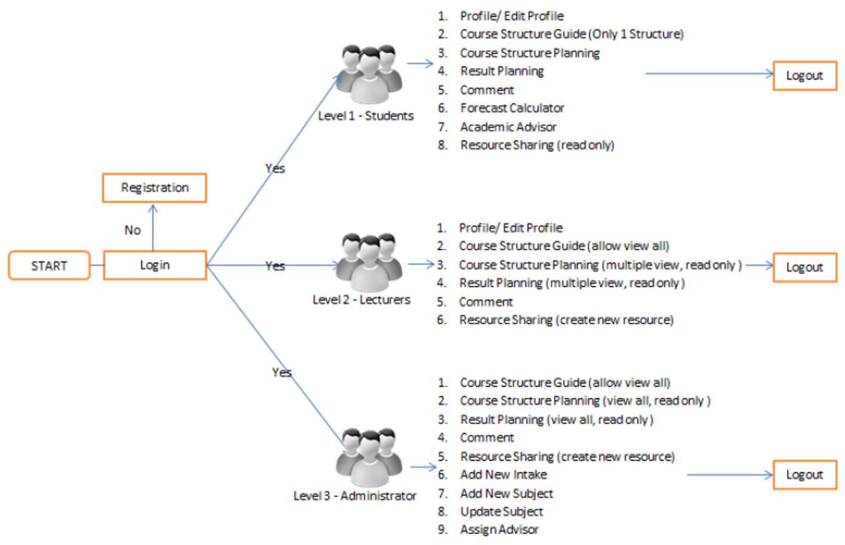

START
Login
No
Registration
Yes
Yes
Yes
Level 1 - Students
1. Profile/ Edit Profile
2. Course Structure Guide (Only 1 Structure)
3. Course Structure Planning
4. Result Planning
5. Comment
6. Forecast Calculator
7. Academic Advisor
8. Resource Sharing (read only)
Logout
Level 2 - Lecturers
1. Profile/ Edit Profile
2. Course Structure Guide (allow view all)
3. Course Structure Planning (multiple view, read only)
4. Result Planning (multiple view, read only)
5. Comment
6. Resource Sharing (create new resource)
Logout
Level 3 - Administrator
1. Course Structure Guide (allow view all)
2. Course Structure Planning (view all, read only)
3. Result Planning (view all, read only)
4. Comment
5. Resource Sharing (create new resource)
6. Add New Intake
7. Add New Subject
8. Update Subject
9. Assign Advisor
Logout

Figure 2.2.1.1 Flow chart of the Student Planning and Advisory System (Kerk 2011)

Students and lecturers were able to view and edit their profile information, and the students could also view the information of their advisor through SPAS. There was a course structure guide function where students can only see their own recommended course structure and plan their own course structure, lecturers were able to view the course structure guides by their advisees, and administrator was able to view all the course structure available in the system. The planning of course structures consist of duplication checking, pre-requisite checking of the subject, and information checking, including the subject code and credit hours. Students could estimate their Grade Point Average (GPA) of the semester by using the result planning function, and estimate their CGPA gained by using the forecast CGPA calculator.

Communication between advisee and advisor has become easier as there will be comment section in the planning section. In addition, advisors and administrators could use the system for sharing comments, files, and links of website through the resource sharing page to share to the students. For administrating purpose, the administrator had the privilege to assign advisor to new students, or change the advisor of students. The system could also be used by the administrator to add new intake course structure guide, add new subjects for the course or edit information of existing course. Lastly, the system is implemented by coding and testing of the system. Different cases and the expected output were planned and listed out for testing out the functions in the system.

All in all, the system has contributed an easy way for students of UTAR to plan their own course structure, and also the forecast of results which brings motivation for students to achieve their goals. The convenience of academic advising and documentation were also improved in the system and the paper work for documentations and repeated work were also reduced. However, there were some limitations in the system and the author recommended to improve in future work. The consultation time of the academic advisor was not provided in the system and thus students could not book the time for consultations easily, and a calendar was recommended to be implement in the system. The communication method was also not convenient as the comment section was only available in one page of the system. A live chat function could be developed in the system so that the students could communicate with their advisor through the system and improve the relationship of advisee-advisor. Besides, subject adding function which subject could only be added one at a time, and thus input of information through CSV files were suggested. Lastly, the results of the students were not able to be integrated to the system was also one of the limitations met in the system.

2.2.2 AdviseMe: An Intelligent Web-Based Application for Academic Advising (Henderson & Goodridge 2015)

AdviseMe is a web-based application which was done for Faculty of Science and Technology in University of the West Indies (UWI) as there were some issues faced in the traditional academic advising method. As the traditional advising process was all paper based, thus it increased the workload of the academic advisor as they also have their primary work as lecturers. The quality of advising was also affected due to inadequate time and inefficiency of advising as everything need to be done manually. Paper based documents during academic advising also led to improper documentation of information. The AdviseMe system was then delivered to solve the issues and improve academic advising quality by migrating from paper-based documentation method of the traditional system to a web-based system. AdviseMe used an Intelligent Advance Automated System (IAAS) which decreased the need for student providing information to generate good results and also to reduce the workload of advisors.

The system used PHP Web Server (PWS) to control the data flow of services within the system. For email communication, PWS would send request to the server of the

Email to send email to users. PWS could also be used to retrieve from or store data to MySQL database by using the user interface of the system for system management. Moreover, PWS would communicate with RESTful Java Web Server (JWS) in providing the intelligent reasoning engine for course advising service. The JWS will be retrieving data from MySQL to generate the RDF-based student profiles, and also create user defined and system defined rules which will be used to apply on the RDF files. Along with an ontology named 'AdviseMeOnt' which used to create interference with the information of student's transcript, an advisory information result would be produced. After generating the results, the results would be transformed to JSON format so that it was in a readable form and the results of course advising would be sent back to PWS through HTTP response for output.

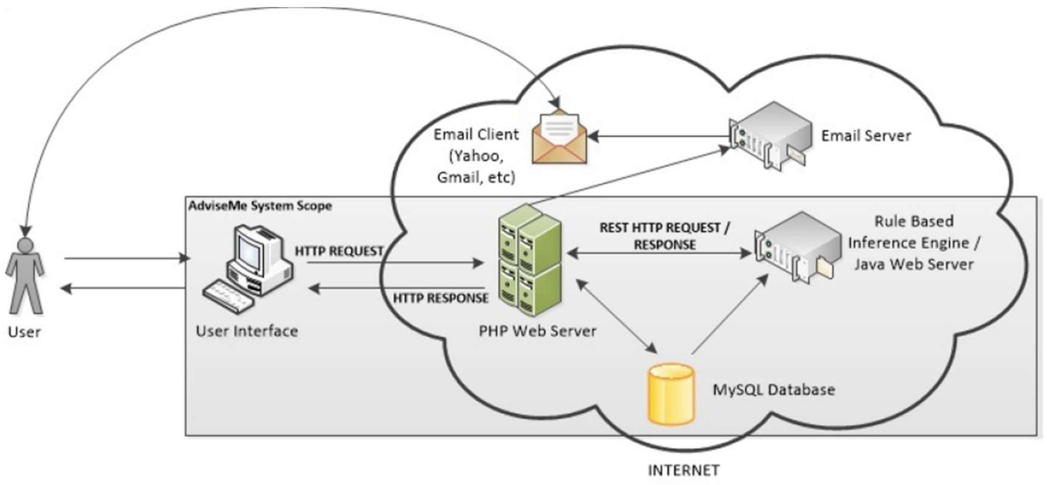


Figure 2.2.2.1 The flow of AdviseMe system (Henderson & Goodridge 2015)

The methodology consists of three phases, the pre-implementation phase, implementation phase and post implementation phase. During the pre-implementation phase, the authors had collected opinion from the side of advisors and also students about the traditional advisory method and a new system for advisory. The Deputy Dean and also Head of Advisory Unit of the Faculty of Science and Technology in UWI were the representative of advisors; and on the other hand, the opinion of students which ranged from first year undergraduates to postgraduate students were collected using questionnaire where the students were anonymous while filling in the questionnaire.

For the implementation phase, the RESTful JWS was implemented to the AdviseMe system to generate academic advises. The 'AdviseMeOnt' Ontology, the user and system defined rules, student profiles could all be created and updated, and also the results of advisory information could be produced. These services would then be transformed to JSON object if the request of service is successful. The PWS would also be implemented for the functionality of the system. Functions for three categories of users including students, advisors and administrators were developed. Students were able to view the comments of advisors from all previous advising meetings, view auto-generated course advising by the system, contact human advisors, view their own graduation status and progress of course, as well as view their information of oral examination. The program structure and common student issues, and also downloading of summary of mentioned information about the student could be accessed by both advisors and students. An advisor could also place comments for students through the system. For administrators, they were able to manage the rules of the system, the content and information in the system, and also the entities in the JWS. After all functionalities are developed, the post implementation phase was carried out. Volunteering students entered their academic transcript into the system to test the functionality and their comments about the user interface and the functionalities were recorded.

The system consisted of some limitations, which the system was only designed for degree programs which has a simple courses path. Besides, the students which contact the human advisors for enquiry through email might not have fast responses from their advisors and would cause unsatisfactory of students. The administrators could only input the student transcript into the system manually was also one of the shortcomings of the system. There were also some enhancements for the system proposed by the author which can be done in future work. The system could be connected to the student information system to process the student transcript, live chats or video conferences functions in the system for enhancing the communication experience of advisee-advisor, a blog for listing out common issues asked by students, and also a program planning function to collect data from student advising and to improve in the allocation of teaching resources in the future.

2.3 Existing Websites

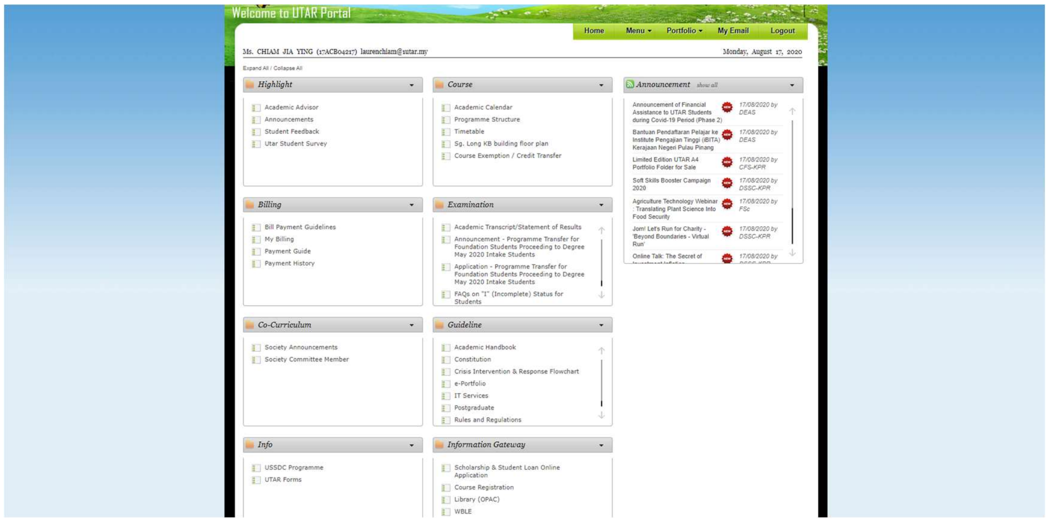
2.3.1 Universiti Tunku Abdul Rahman (UTAR) Portal

Universiti Tunku Abdul Rahman (UTAR) is well known by Malaysians as it is one of the top not-for-profit private university in the country. It was first established in 13 August 2002, and now it has two campuses, which are Kampar, Perak campus and Bandar Sungai Long, Selangor campus. The vision of UTAR is to become a global university which excel in education and bringing impact to transform the society. UTAR provides over 110 of academic programs and the enrolment has reached more than 23,000 students. UTAR provides a portal website for students to view all university related information.

First and foremost, only authorised accounts can log in to the portal of UTAR. After student's login, there are lists of function menus can be seen. Students can see their own profile after accessing the portfolio. The personal information and photo of the student will be shown. Students can also change their password of their account through the link in the profile page. There is also a tab for student to be redirected to their own student email. In the highlights, students can also access to their academic advisor page, where they can view all the information about their academic advisor, and the previous consultation notes written by the advisor can be accessed by clicking on the serial number of the consultation.

In the course structure menu, students can view their course related information, including the academic calendar of the university for the year, the suggested program structure of the student's course, the timetable of the semester and also the course that is exempted by the university. They can also see examination information, such as the academic transcript which lists out all previous examination results, the upcoming final examination timetable, final examination result of most recent final exam, and announcements about the examination in the examination menu.

The student can view their co-curriculum and soft skill programs information through the portal. Application forms related to activities in campus and academic purposes are provided in the portal too. There are also links provided to redirect the students to other UTAR's websites. Lastly, there is an announcement column where all announcements made by UTAR will be shown there.



Ms. CHIAM JIA YING (17ACB04217) laurenchiam@1utar.my

Monday, August 17, 2020

Expand All / Collapse All

Highlight
- Academic Advisor
- Announcements
- Student Feedback
- Utar Student Survey

Course
- Academic Calendar
- Programme Structure
- Timetable
- Sg. Long KB building floor plan
- Course Exemption / Credit Transfer

Announcement show all

Announcement of Financial Assistance to UTAR Students during Covid-19 Period (Phase 2)	17/08/2020 by DEAS
Bantuan Pendaftaran Pelajar ke Institute Pengajian Tinggi (iBITA) Kerajaan Negeri Pulau Pinang	17/08/2020 by DEAS
Limited Edition UTAR A4 Portfolio Folder for Sale	17/08/2020 by CFS-KPR
Soft Skills Booster Campaign 2020	17/08/2020 by DSSC-KPR
Agriculture Technology Webinar : Translating Plant Science Into Food Security	17/08/2020 by FSc
Jom! Let's Run for Charity - 'Beyond Boundaries - Virtual Run'	17/08/2020 by DSSC-KPR
Online Talk: The Secret of	17/08/2020 by

Billing
- Bill Payment Guidelines
- My Billing
- Payment Guide
- Payment History

Examination
- Academic Transcript/Statement of Results
- Announcement - Programme Transfer for Foundation Students Proceeding to Degree May 2020 Intake Students
- Application - Programme Transfer for Foundation Students Proceeding to Degree May 2020 Intake Students
- FAQs on "I" (Incomplete) Status for Students

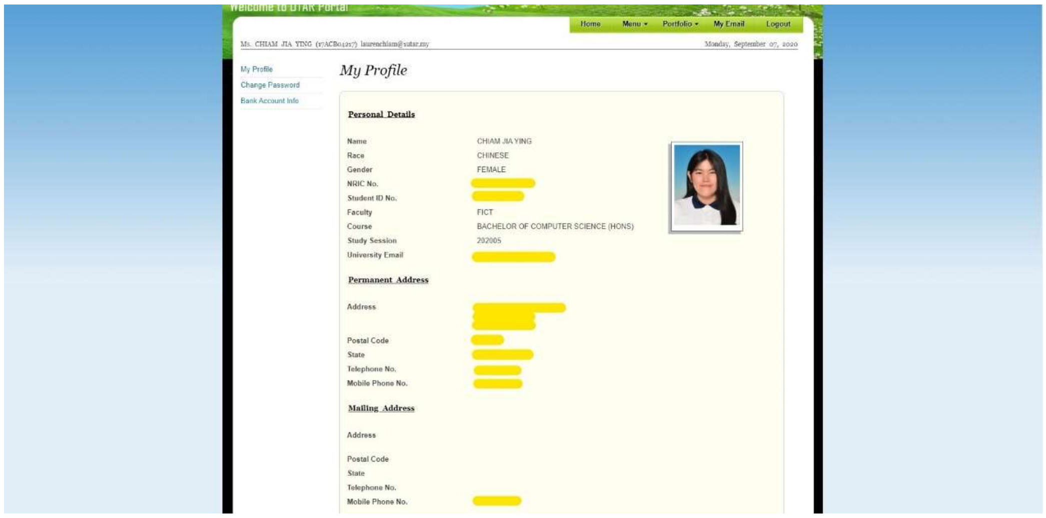
Co-Curriculum
- Society Announcements
- Society Committee Member

Guideline
- Academic Handbook
- Constitution
- Crisis Intervention & Response Flowchart
- e-Portfolio
- IT Services
- Postgraduate
- Rules and Regulations

Info
- USSDC Programme
- UTAR Forms

Information Gateway
- Scholarship & Student Loan Online Application
- Course Registration
- Library (OPAC)
- WBLE

Figure 2.3.1.1 The main page of student portal after login



Ms. CHIAM JIA YING (17ACB04217) laurenchiam@1utar.my

Monday, September 07, 2020

My Profile

Personal Details

Name	CHIAM JIA YING
Race	CHINESE
Gender	FEMALE
NRIC No.	[redacted]
Student ID No.	[redacted]
Faculty	FICT
Course	BACHELOR OF COMPUTER SCIENCE (HONS)
Study Session	202005
University Email	[redacted]

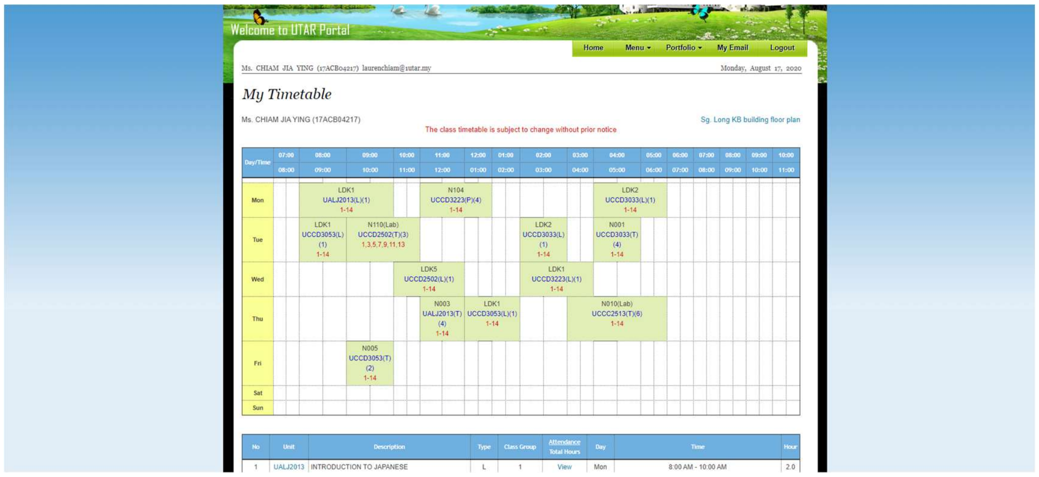
Permanent Address

Address	[redacted]
Postal Code	[redacted]
State	[redacted]
Telephone No.	[redacted]
Mobile Phone No.	[redacted]

Mailing Address

Address	
Postal Code	
State	
Telephone No.	
Mobile Phone No.	[redacted]

Figure 2.3.1.2 Student profile page



Ms. CHIAM JIA YING (17ACB04217) laurenchiam@1utar.my

Monday, August 17, 2020

My Timetable

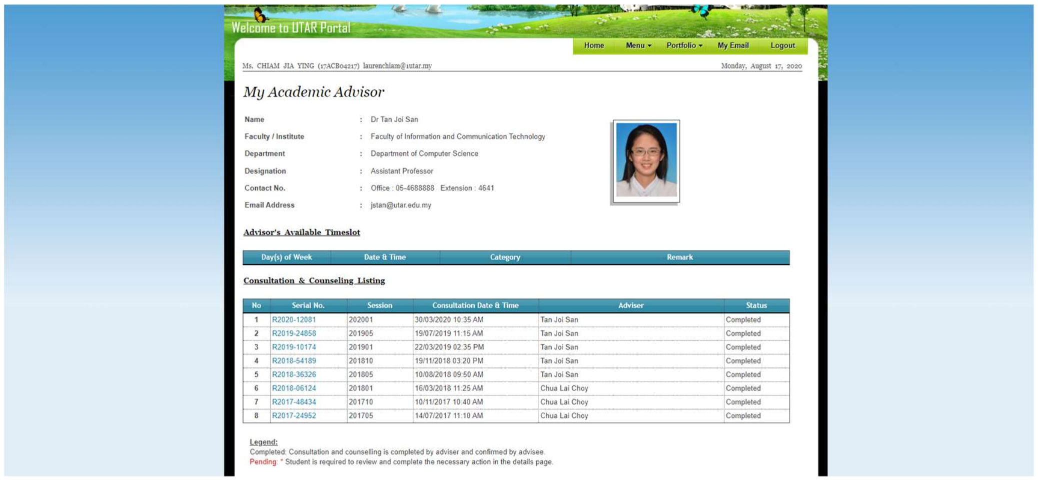
Ms. CHIAM JIA YING (17ACB04217)

The class timetable is subject to change without prior notice

Day/Time	07:00 08:00	08:00 09:00	09:00 10:00	10:00 11:00	11:00 12:00	12:00 01:00	01:00 02:00	02:00 03:00	03:00 04:00	04:00 05:00	05:00 06:00	06:00 07:00	07:00 08:00	08:00 09:00	09:00 10:00	10:00 11:00
Mon		LDK1 UALJ2013(L)(1) 1-14			N104 UCCD3223(P)(4) 1-14					LDK2 UCCD3033(L)(1) 1-14						
Tue		LDK1 UCCD3053(L) (1) 1-14	N110(Lab) UCCD2502(T)(3) 1,3,5,7,9,11,13					LDK2 UCCD3033(L) (1) 1-14		N001 UCCD3033(T) (4) 1-14						
Wed				LDK5 UCCD2502(L)(1) 1-14				LDK1 UCCD3223(L)(1) 1-14								
Thu					N003 UALJ2013(T) (4) 1-14	LDK1 UCCD3053(L)(1) 1-14			N010(Lab) UCCC2513(T)(6) 1-14							
Fri			N005 UCCD3053(T) (2) 1-14													
Sat																
Sun																

No	Unit	Description	Type	Class Group	Attendance Total Hours	Day	Time	Hour
1	UALJ2013	INTRODUCTION TO JAPANESE	L	1	View	Mon	8:00 AM - 10:00 AM	2.0

Figure 2.3.1.3 Timetable of student for current semester



Ms. CHIAM JIA YING (17ACB04217) laurenchiam@1utar.my Monday, August 17, 2020

My Academic Advisor

Name	:	Dr Tan Joi San
Faculty / Institute	:	Faculty of Information and Communication Technology
Department	:	Department of Computer Science
Designation	:	Assistant Professor
Contact No.	:	Office : 05-4688888 Extension : 4641
Email Address	:	jstan@utar.edu.my

Advisor's Available Timeslot

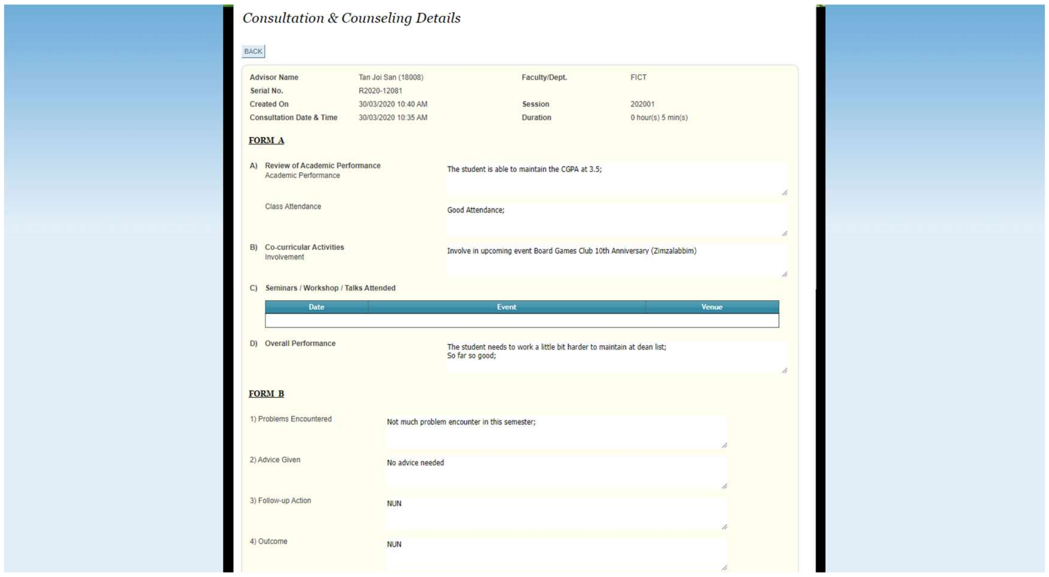
Day(s) of Week	Date & Time	Category	Remark

Consultation & Counseling Listing

No	Serial No.	Session	Consultation Date & Time	Adviser	Status
1	R2020-12081	202001	30/03/2020 10:35 AM	Tan Joi San	Completed
2	R2019-24858	201905	19/07/2019 11:15 AM	Tan Joi San	Completed
3	R2019-10174	201901	22/03/2019 02:35 PM	Tan Joi San	Completed
4	R2018-54189	201810	19/11/2018 03:20 PM	Tan Joi San	Completed
5	R2018-36326	201805	10/08/2018 09:50 AM	Tan Joi San	Completed
6	R2018-06124	201801	16/03/2018 11:25 AM	Chua Lai Choy	Completed
7	R2017-48434	201710	10/11/2017 10:40 AM	Chua Lai Choy	Completed
8	R2017-24952	201705	14/07/2017 11:10 AM	Chua Lai Choy	Completed

Legend:
Completed: Consultation and counselling is completed by adviser and confirmed by advisee.
Pending: * Student is required to review and complete the necessary action in the details page.

Figure 2.3.1.4 Academic advisor information page



Consultation & Counseling Details

Advisor Name	Tan Joi San (18008)	Faculty/Dept.	FICT
Serial No.	R2020-12081		
Created On	30/03/2020 10:40 AM	Session	202001
Consultation Date & Time	30/03/2020 10:35 AM	Duration	0 hour(s) 5 min(s)

FORM A

A) Review of Academic Performance
Academic Performance: The student is able to maintain the CGPA at 3.5;
Class Attendance: Good Attendance;

B) Co-curricular Activities
Involvement: Involve in upcoming event Board Games Club 10th Anniversary (Zimzalabbim)

C) Seminars / Workshop / Talks Attended

Date	Event	Venue

D) Overall Performance: The student needs to work a little bit harder to maintain at dean list;
So far so good;

FORM B

1) Problems Encountered: Not much problem encounter in this semester;
2) Advice Given: No advice needed
3) Follow-up Action: NUN
4) Outcome: NUN

Figure 2.3.1.5 The notes and comments of academic advisor after consultation

Strength

- Advisee can access all academic related information in the website.
- Advisee can view their suggested course structure through the website.
- The advisor profile can be viewed through the website.
- Advisor can add their available timeslot in the platform for advisee to view.
- The administrator can make top announcement through pop up on the bottom of the main page.

Weakness

- The icon of each menu and submenu is the same and is confusing to the users as it does not bring out the meaning of the menu and submenu.
- The website design is not responsive for smaller screens such as mobiles.
- There is no communication tool between advisee and advisor within the website and they can only connect through email.
- Advisee need to find their advisor personally to book their time for consulting.

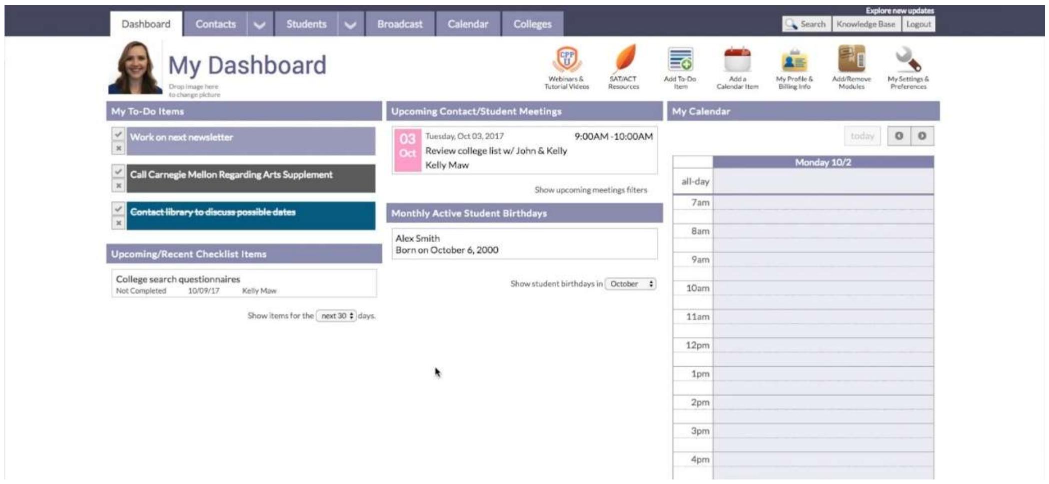
2.3.2 CollegePlannerPro

CollegePlannerPro is a college consulting advisory website, which was co-founded by Travis and Brooke Daly. It was created due to Brooke's college planning company needs to have a solution to manage the information and notes of the advisory process. It was established in 2010 and it is now evolved and provides services for thousands of independent college planners.

The advisor and advisee can login with different login page to access the website. For advisor, after login, the advisor will see a dashboard where to-do lists, upcoming checklists, upcoming student meetings, and student birthdays. There is also a calendar where advisor can add in their own schedule in it. Besides, advisor can view the contacts of the advisee's parents in the contact tab.

For the advisor's student tab, the advisor can set meetings with advisee and also set a reminder for the meeting. Besides, a checklist can also be set by the advisor to remind himself of the things that need to complete with the advisee. The advisee will not see the checklist set by the advisor. In addition, if there are tasks that the advisor wants to assign to the advisee, the advisor can set it through the to-do list of the advisee. For communication, there is also a conversation column where the advisor can chat with the advisee there. The advisor can write down comments for the advisee through the student notes & email section, and the advisor can choose to keep it in the platform, or send the notes to the advisee email additionally. File transfer function is also provided by the platform where the advisor can send upload the files to the platform, and the access can be shared to multiple advisees or set to be accessed by only one advisee. The test scores of the advisee can also be displayed in the student profile.

On the other hand, advisee can login with the account which is given by the advisor. After login, they can view the to-do lists set by the advisor, the scheduled meetings with the advisor, the chat conversation with the advisor, and also own's personal information. The advisee can also schedule a meeting with the advisor in the platform. For test scores, advisee need to key in his test scores to the system manually. There is also a files tab where the files shared by the advisor will be seen there. Lastly, there is a calendar tab, which the to-do list set by the advisor and the meeting time with advisor will be shown here, and advisee can add on new events to the calendar manually.



My Dashboard

Drop image here to change picture

My To-Do Items

Work on next newsletter

Call Carnegie Mellon Regarding Arts Supplement

~~Contact library to discuss possible dates~~

Upcoming/Recent Checklist Items

College search questionnaires
Not Completed 10/09/17 Kelly Maw

Show items for the next 30 days.

Upcoming Contact/Student Meetings

03 Oct Tuesday, Oct 03, 2017 9:00AM - 10:00AM
Review college list w/ John & Kelly
Kelly Maw

Show upcoming meetings filters

Monthly Active Student Birthdays

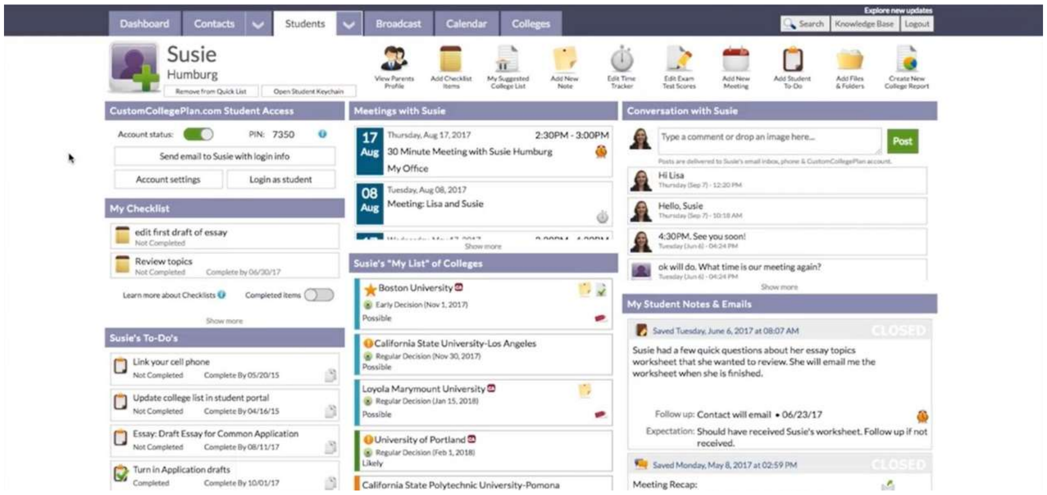
Alex Smith
Born on October 6, 2000

Show student birthdays in October

My Calendar

Monday 10/2

Figure 2.3.2.1 Advisor's dashboard (CollegePlannerPro n.d.)



Susie
Humburg

CustomCollegePlan.com Student Access

Account status: PIN: 7350

Send email to Susie with login info

Account settings | Login as student

My Checklist

edit first draft of essay
Not Completed

Review topics
Not Completed Complete by 06/30/17

Learn more about Checklists Completed items

Susie's To-Do's

Link your cell phone
Not Completed Complete By 05/20/15

Update college list in student portal
Not Completed Complete By 04/16/15

Essay: Draft Essay for Common Application
Not Completed Complete By 08/11/17

Turn in Application drafts
Completed Complete By 10/01/17

Meetings with Susie

17 Aug Thursday, Aug 17, 2017 2:30PM - 3:00PM
30 Minute Meeting with Susie Humburg
My Office

08 Aug Tuesday, Aug 08, 2017
Meeting: Lisa and Susie

Susie's "My List" of Colleges

Boston University
Early Decision (Nov 1, 2017)
Possible

California State University-Los Angeles
Regular Decision (Nov 30, 2017)
Possible

Loyola Marymount University
Regular Decision (Jan 15, 2018)
Possible

University of Portland
Regular Decision (Feb 1, 2018)
Likely

California State Polytechnic University-Pomona

Conversation with Susie

Type a comment or drop an image here...

Posts are delivered to Susie's email inbox, phone & CustomCollegePlan account.

Hi Lisa
Thursday (Sep 7) - 12:20 PM

Hello, Susie
Thursday (Sep 7) - 10:18 AM

4:30PM. See you soon!
Tuesday (Jun 6) - 04:24 PM

ok will do. What time is our meeting again?
Tuesday (Jun 6) - 04:24 PM

My Student Notes & Emails

Saved Tuesday, June 6, 2017 at 08:07 AM CLOSED

Susie had a few quick questions about her essay topics worksheet that she wanted to review. She will email me the worksheet when she is finished.

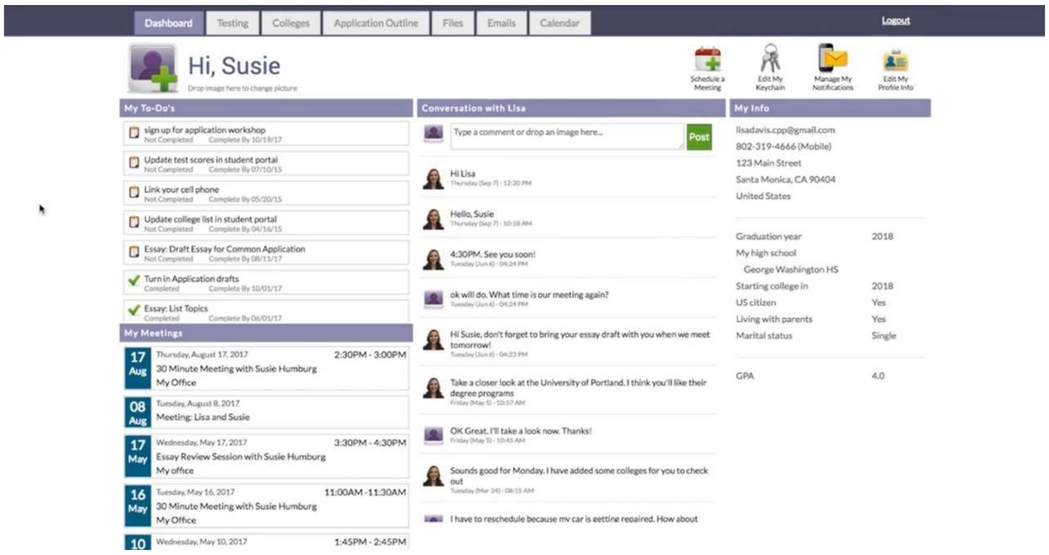
Follow up: Contact will email • 06/23/17

Expectation: Should have received Susie's worksheet. Follow up if not received.

Saved Monday, May 8, 2017 at 02:59 PM CLOSED

Meeting Recap:

Figure 2.3.2.2 Advisee profile view from advisor's account (CollegePlannerPro n.d.)



Hi, Susie

Drop image here to change picture

My To-Do's

- sign up for application workshop — Not Completed — Complete By 10/19/17
- Update test scores in student portal — Not Completed — Complete By 07/10/15
- Link your cell phone — Not Completed — Complete By 05/20/15
- Update college list in student portal — Not Completed — Complete By 04/16/15
- Essay: Draft Essay for Common Application — Not Completed — Complete By 08/11/17
- ✓ Turn in Application drafts — Completed — Complete By 10/01/17
- ✓ Essay: List Topics — Completed — Complete By 06/01/17

My Meetings

- 17 Aug — Thursday, August 17, 2017 — 2:30PM - 3:00PM — 30 Minute Meeting with Susie Humburg — My Office
- 08 Aug — Tuesday, August 8, 2017 — Meeting: Lisa and Susie
- 17 May — Wednesday, May 17, 2017 — 3:30PM - 4:30PM — Essay Review Session with Susie Humburg — My office
- 16 May — Tuesday, May 16, 2017 — 11:00AM -11:30AM — 30 Minute Meeting with Susie Humburg — My Office
- 10 — Wednesday, May 10, 2017 — 1:45PM - 2:45PM

Conversation with Lisa

Type a comment or drop an image here...

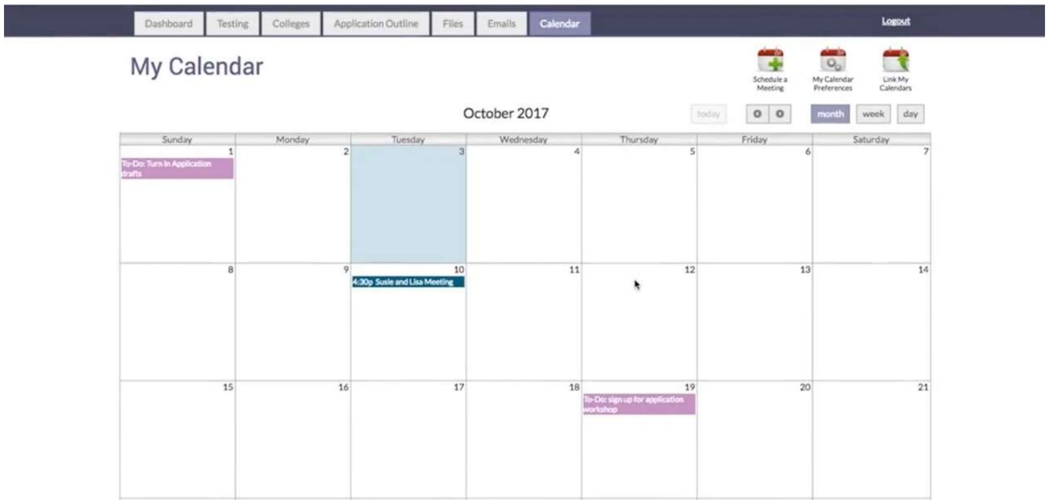
- Hi Lisa — Thursday (Sep 7) - 12:20 PM
- Hello, Susie — Thursday (Sep 7) - 10:18 AM
- 4:30PM. See you soon! — Tuesday (Jun 6) - 04:24 PM
- ok will do. What time is our meeting again? — Tuesday (Jun 6) - 04:24 PM
- Hi Susie, don't forget to bring your essay draft with you when we meet tomorrow! — Tuesday (Jun 6) - 04:23 PM
- Take a closer look at the University of Portland. I think you'll like their degree programs — Friday (May 5) - 10:57 AM
- OK Great. I'll take a look now. Thanks! — Friday (May 5) - 10:45 AM
- Sounds good for Monday. I have added some colleges for you to check out — Tuesday (Mar 24) - 08:15 AM
- I have to reschedule because my car is getting repaired. How about

My Info

lisadavis.cpp@gmail.com
802-319-4666 (Mobile)
123 Main Street
Santa Monica, CA 90404
United States

Graduation year	2018
My high school	
George Washington HS	
Starting college in	2018
US citizen	Yes
Living with parents	Yes
Marital status	Single
GPA	4.0

Figure 2.3.2.3 Advisee's dashboard (CollegePlannerPro n.d.)



Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1 To-Do: Turn in Application drafts	2	3	4	5	6	7
8	9	10 4:30p Susie and Lisa Meeting	11	12	13	14
15	16	17	18	19 To-Do: sign up for application workshop	20	21

Figure 2.3.2.4 Calendar view of advisee's event (CollegePlannerPro n.d.)

Strength

- Contains chat conversation function for advisee-advisor communication in the platform itself.
- Calendar view for both advisee and advisor to have a clear overview of the schedule.
- The advisee and advisor can set notification to either send to their email or send through text messages as reminders.
- The advisee can book meetings with his advisor through the website in the calendar.

Weakness

- Advisee cannot view the profile and information of the advisor.
- There is no administrator for the website.
- User interface is packed as there are three columns of contents in one page.

2.3.3 FlightPath

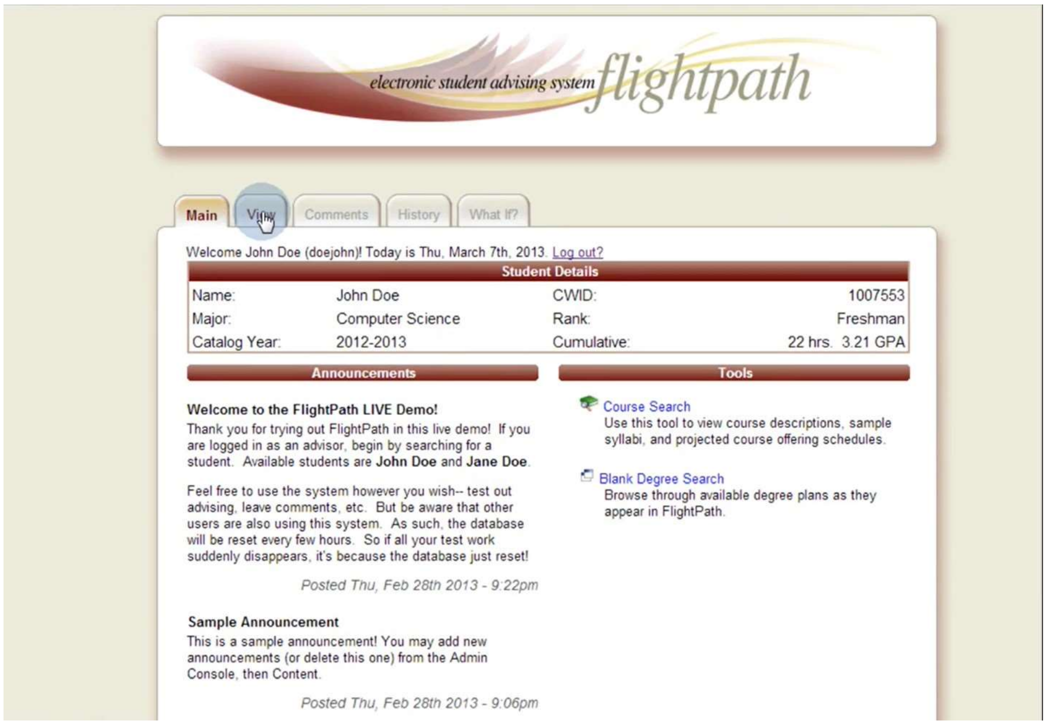
FlightPath is an electronic student advising system which was established in 2006 by University of Louisiana, and was released as open source in 2013. It was developed by Richard Peacock, and he is currently still maintaining the website getflightpath.com (FlightPath n.d.). The latest release of the website is the version 5.2.1, which was released on March 2020. It can work on various browsers, such as Google Chrome, Mozilla Firefox, Internet Explorer and Safari. The project was designed to standardize the academic advising process and to document the previous advisory history for future referencing by the advisee and advisor.

The platform is provided for administrator, advisor and advisee, and all of them are provided with different access and features. Administrator is eligible to assign permissions to each role and edit basic system requirements. The administrator is also in charge of update the degree requirements input. They can use the banner function to make important announcement which will be shown at most top of every page of the website, and also announcement function where both advisee and advisor can see on their main page.

On the other hand, advisor can search their advisee by searching their names or student ID, view their degree progression and requirements, and can provide suggestions to the advisee on course structure arrangements, or assign the substitution of subjects. Comments can be given to the advisee, which the advisor can choose to only let other advisors view it by using the faculty comment, or set the permission to also let the advisee to view. The advisor can also browse the advising session history of the advisee.

For advisee, they can see their academic information, including their name, course, student ID, GPA, credit hours obtained and grades. Besides, they can use the platform to view their degree progression and the course structure. There will be pie charts showing the percentage of completion of different category of courses, and the course structure will be listed out below the pie charts. The advising history and the advisor's comments can also be seen by the advisee through the history page.

For both the advisee and advisor, there is a function named "What If?", allowing them to stimulate a scenario where advisee changes course and credit transfer to a different degree, then estimate the completion of progress in the changed degree. It is useful especially for advisor if they realize that the advisee is not coping well with the current degree and want to find a more suitable program for their advisee.



Welcome John Doe (doejohn)! Today is Thu, March 7th, 2013. Log out?

Student Details

Name:	John Doe	CWID:	1007553
Major:	Computer Science	Rank:	Freshman
Catalog Year:	2012-2013	Cumulative:	22 hrs. 3.21 GPA

Announcements

Welcome to the FlightPath LIVE Demo!
Thank you for trying out FlightPath in this live demo! If you are logged in as an advisor, begin by searching for a student. Available students are **John Doe** and **Jane Doe**.

Feel free to use the system however you wish-- test out advising, leave comments, etc. But be aware that other users are also using this system. As such, the database will be reset every few hours. So if all your test work suddenly disappears, it's because the database just reset!

Posted Thu, Feb 28th 2013 - 9:22pm

Sample Announcement
This is a sample announcement! You may add new announcements (or delete this one) from the Admin Console, then Content.

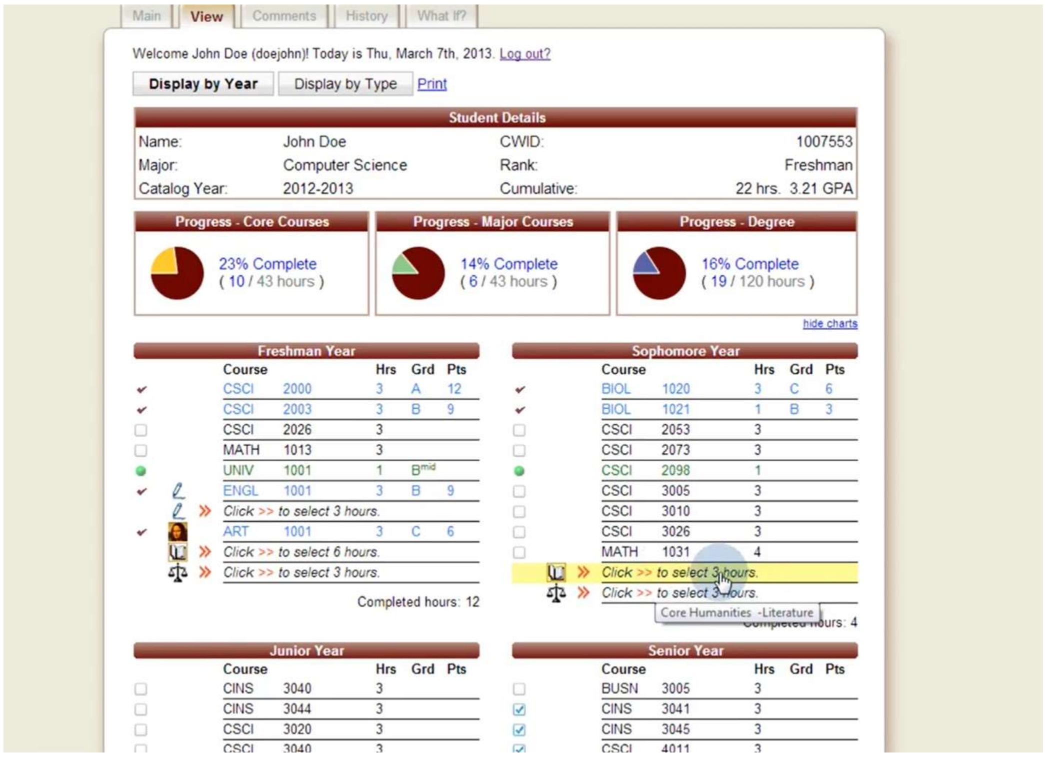
Posted Thu, Feb 28th 2013 - 9:06pm

Tools

Course Search
Use this tool to view course descriptions, sample syllabi, and projected course offering schedules.

Blank Degree Search
Browse through available degree plans as they appear in FlightPath.

Figure 2.3.3.1 Advisee's main page (FlightPath 2013)



Welcome John Doe (doejohn)! Today is Thu, March 7th, 2013. Log out?

Student Details

Name:	John Doe	CWID:	1007553
Major:	Computer Science	Rank:	Freshman
Catalog Year:	2012-2013	Cumulative:	22 hrs. 3.21 GPA

Progress - Core Courses	Progress - Major Courses	Progress - Degree
23% Complete (10 / 43 hours)	14% Complete (6 / 43 hours)	16% Complete (19 / 120 hours)

Freshman Year

	Course		Hrs	Grd	Pts
✓	CSCI	2000	3	A	12
✓	CSCI	2003	3	B	9
☐	CSCI	2026	3		
☐	MATH	1013	3		
●	UNIV	1001	1	B^{mid}	
✓	ENGL	1001	3	B	9
	Click >> to select 3 hours.				
✓	ART	1001	3	C	6
	Click >> to select 6 hours.				
	Click >> to select 3 hours.				

Completed hours: 12

Sophomore Year

	Course		Hrs	Grd	Pts
✓	BIOL	1020	3	C	6
✓	BIOL	1021	1	B	3
☐	CSCI	2053	3		
☐	CSCI	2073	3		
●	CSCI	2098	1		
☐	CSCI	3005	3		
☐	CSCI	3010	3		
☐	CSCI	3026	3		
☐	MATH	1031	4		
	Click >> to select 3 hours.				
	Click >> to select 3 hours.				

Core Humanities -Literature

Completed hours: 4

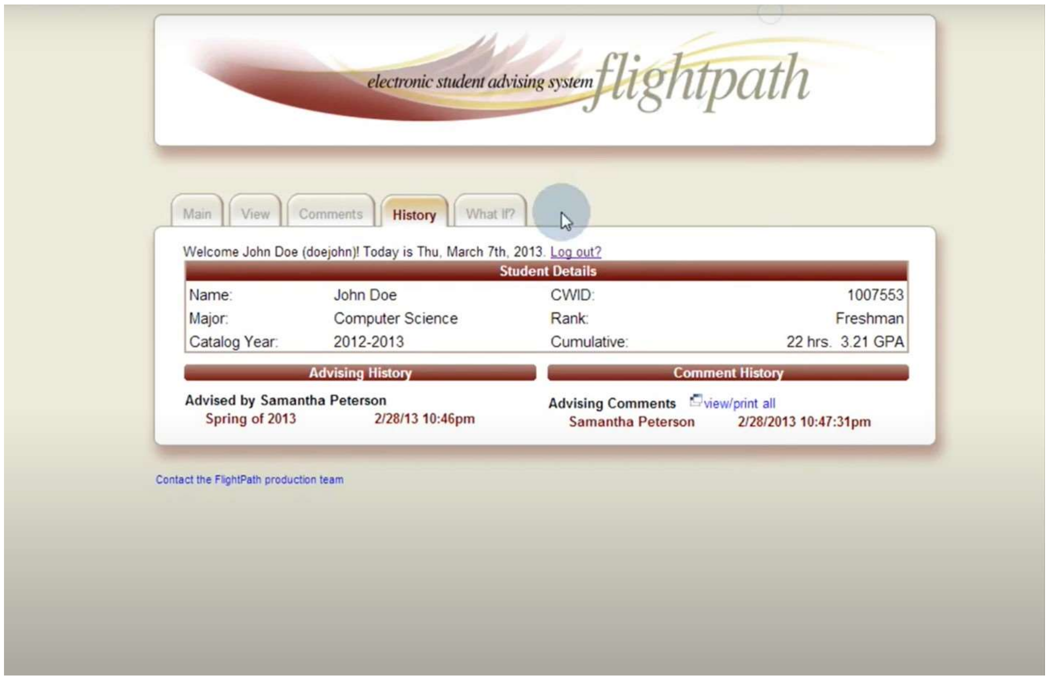
Junior Year

	Course		Hrs	Grd	Pts
☐	CINS	3040	3		
☐	CINS	3044	3		
☐	CSCI	3020	3		
☐	CSCI	3040	3		

Senior Year

	Course		Hrs	Grd	Pts
☐	BUSN	3005	3		
✓	CINS	3041	3		
✓	CINS	3045	3		
✓	CSCI	4011	3		

Figure 2.3.3.2 Academic information of advisee and progress of completion of degree (FlightPath 2013)



Welcome John Doe (doejohn)! Today is Thu, March 7th, 2013. Log out?

Student Details

Name:	John Doe	CWID:	1007553
Major:	Computer Science	Rank:	Freshman
Catalog Year:	2012-2013	Cumulative:	22 hrs. 3.21 GPA

Advising History

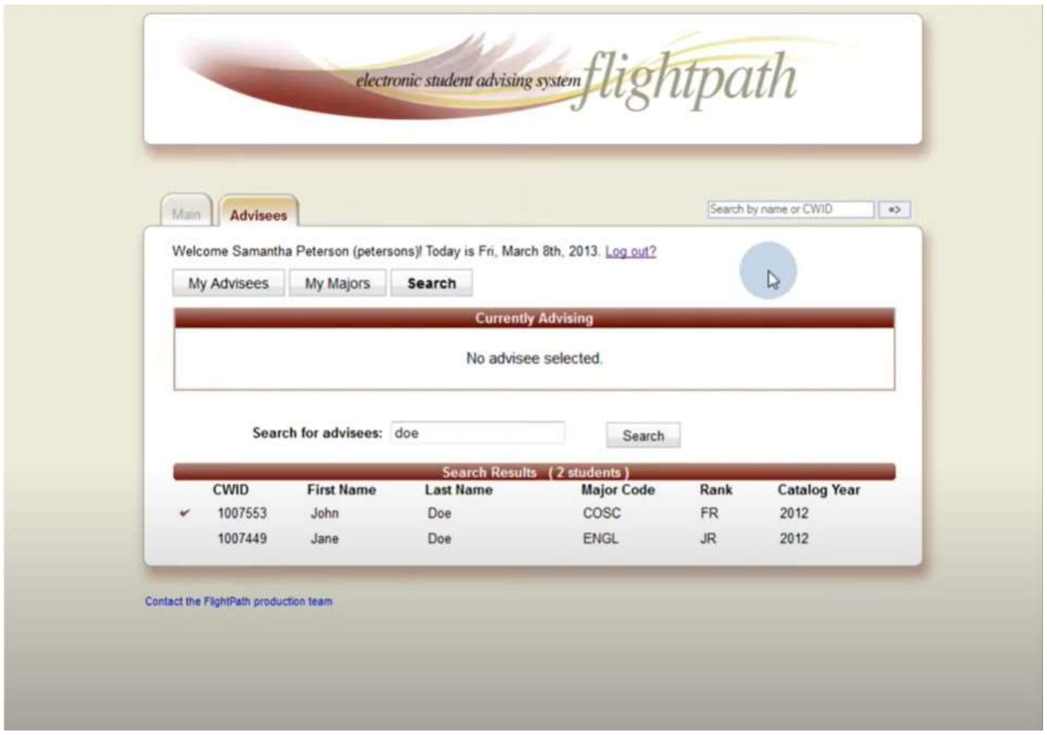
Advised by Samantha Peterson
Spring of 2013 2/28/13 10:46pm

Comment History

Advising Comments view/print all
Samantha Peterson 2/28/2013 10:47:31pm

Contact the FlightPath production team

Figure 2.3.3.3 Advising history and comment history by the advisor (FlightPath 2013)



Welcome Samantha Peterson (petersons)! Today is Fri, March 8th, 2013. Log out?

Currently Advising

No advisee selected.

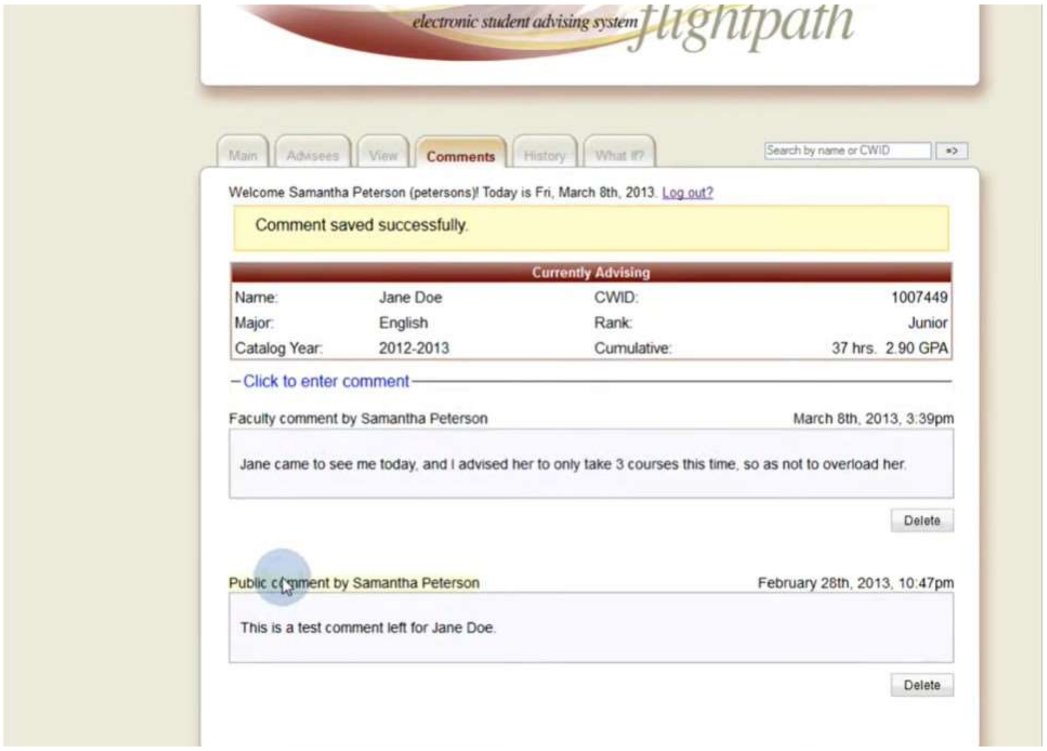
Search for advisees: doe Search

Search Results (2 students)

	CWID	First Name	Last Name	Major Code	Rank	Catalog Year
✓	1007553	John	Doe	COSC	FR	2012
	1007449	Jane	Doe	ENGL	JR	2012

Contact the FlightPath production team

Figure 2.3.3.4 Search advisee using name of advisee (FlightPath 2013)



Welcome Samantha Peterson (petersons)! Today is Fri, March 8th, 2013. Log out?

Comment saved successfully.

Currently Advising

Name:	Jane Doe	CWID:	1007449
Major:	English	Rank:	Junior
Catalog Year:	2012-2013	Cumulative:	37 hrs. 2.90 GPA

Click to enter comment

Faculty comment by Samantha Peterson March 8th, 2013, 3:39pm

Jane came to see me today, and I advised her to only take 3 courses this time, so as not to overload her.

Public comment by Samantha Peterson February 28th, 2013, 10:47pm

This is a test comment left for Jane Doe.

Figure 2.3.3.5 Comments by advisors for faculty access only and public comment which can be accessed by both advisee and advisor (FlightPath 2013)

Strength

- The website is responsive and responds when the size of display changes.
- There is a banner function for administrator to write top announcement and notice the users at the top of page of the website.
- The course structure of the advisee is shown in the website.

Weakness

- There is no communication method in the website.
- The information of the advisor such as email and phone number are not provided in the website, thus the advisee could not know the contact of the advisor and could only find the advisor through the office.
- The timeslot available of the advisor is not provided in the platform, thus advisee need to go to the advisor's office to know whether the advisor is free, and is very inconvenient for the advisee.

2.4 Summary

In this chapter, two similar projects named SPAS and AdviseMe has been reviewed. For SPAS, the author has used a waterfall methodology for the development of the system, and has contributions on bringing students ease in planning by using the planning tool, improving education quality and improve the efficiency of document management in university. There are also some limitations as the time constraint of the project is short. The students' results could not be export from the department of examination, the consultation time of the advisor is not provided, communication method in the platform is limited, and the subjects can only be added one by one manually. On the other hand, AdviseMe has used a methodology consisting of pre-implementation phase, implementation phase and post implementation phase. Functionalities for students' academic information and issues, automated course-advising, comments from advisors, communication with human academic advisors, system and information management which are driven on PWS were developed. The limitations of the system such as the limited support of degree courses for automated academic advising, slow response of advisors through email, and student transcript which needs to be key in manually were listed. For both projects, recommendations were given by the authors to enhance the system in future works.

Three existing websites, including UTAR Portal, CollegePlannerPro and FlightPath are reviewed, and the strengths and weaknesses are listed out. Their strengths and weaknesses are then summarised and concluded in Table 2.4.1.

Table 2.4.1 Comparison of strengths and weaknesses of existing websites

	UTAR Portal	**CollegePlannerPro**	**FlightPath**
Design of User Interface	Simple design, however the icons used for menus and submenus are not meaningful.	Attractive design with purple theme and uses icons for different functions, but packed due to three columns of activities in a page.	Simple design with red theme.
Responsive web design	Not responsive in mobile view.	A mobile version is provided for mobile users.	Responsive in mobile view.
Page roles	Administrator, advisor, advisee	Advisor, advisee	Administrator, advisor, advisee
Advisor profile and information	Advisor's profile and information is shown to the advisees.	Not provided for advisee.	Not provided for advisee.
Advisee-advisor communication method	Not provided in the platform.	Advisee and advisor can have a conversation using the chat function in the platform.	Not provided in the platform.
Available timeslot of the advisor	Advisor can add their available timeslot for the advisee to view.	Advisor can add their available timeslot for the advisee to view.	Not provided for advisee to view.
Booking of meeting time	Not provided in the platform.	Can book directly through the platform.	Not provided in the platform.
Course structure	Course structure can be viewed by advisee and advisor.	Not provided in the platform.	Course structure can be viewed by advisee, can be view

			and edited by advisor.
Event calendar	Not provided in the platform.	There is event calendar for advisee and advisor where they can add and edit their events in the calendar.	Not provided in the platform.
Top announcements	There will be pop up at the bottom of the page.	No top announcement can be made.	Administrator can set a banner on top of the page
Notification	Not provided.	Advisee and advisor can set notifications which notify through email or text messages.	Not provided.

CHAPTER 3 SYSTEM DESIGN

3.1 Overview

The use case diagram and descriptions are done in this chapter to show the overview of the system, and the development process is briefly explained through the code of the project.

3.2 Use-Cases Diagram

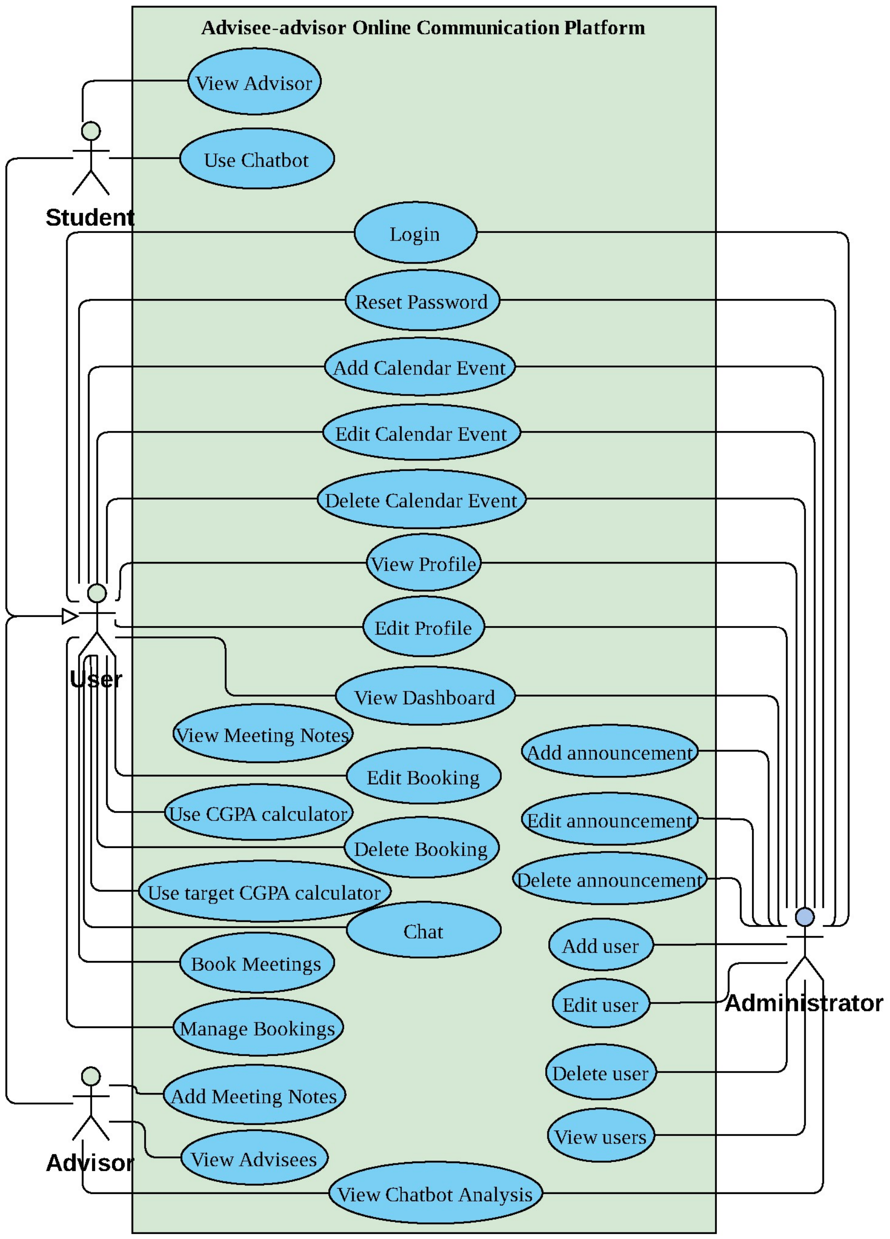


Figure 2.3.3 Use-case Diagram for Advisee-Advisor Online Communication Platform

3.3 Use-case Description

Table 3.3.1 Login Use Case Description

Use Case ID	00001
Use Case Name	Login
Brief Description	Student, advisor and administrator is allowed to access to the platform.
Actor	Advisee, Advisor, Administrator
Trigger	Go to login page of the platform.
Precondition	Account must exist in the system and has not logged in.
Normal flow of events	1. Advisee / Advisor / Administrator fill in email and password. 2. The system validates whether the account exists, and the email and password are correct. 3. The system provides a session cookie for the advisee / advisor / administrator. 4. The system will redirect them to their dashboard.
Sub flows	3a. Advisee / advisor / administrator does not need to login again within 5 days. 3b. If the session cookies expired, user need to re-login to the platform again.
Alternate flows	2a. If the account does not exist, or the combination of email and password is false, the system will show the error message to the advisee / advisor / administrator, and reinsertion is needed until valid email and password is inserted.

Table 3.3.2 Reset Password Use Case Description

Use Case ID	00002
Use Case Name	Reset Password
Brief Description	Student, advisor and administrator is allowed to reset password if they forget their password.
Actor	Advisee, Advisor, Administrator
Trigger	Clicked on “Reset” button in the login page.
Precondition	Account must exist in the system and advisee / advisor / administrator forgotten their password.
Normal flow of events	1. Advisee / Advisor / Administrator click on Reset button to forget password section in login page.

	2. Advisee / Advisor / Administrator key in their email address and click reset. 3. The system will then send reset password link to their email. 4. Advisee / Advisor / Administrator click on the link and will be redirected to the Reset Password page. 5. Advisee / Advisor / Administrator key in their new password and confirm new password to reset their password.
Sub flows	-
Alternate flows	-

Table 3.3.3 View Dashboard Use Case Description

Use Case ID	00003
Use Case Name	View Dashboard
Brief Description	Dashboard will be showing the summary information of the system, for the user, such as announcements, calendar in list view, booking status and etc.
Actor	Advisee, Advisor, Administrator
Trigger	The dashboard page is selected in the navigation bar.
Precondition	Account must be logged in
Normal flow of events	1. Advisee / Advisor / Administrator enter the platform using default URL. 2. Advisee / Advisor / Administrator will be redirected to the dashboard page if account still have active login. 3. Advisee / Advisor / Administrator is now able to view the details in dashboard
Sub flows	-
Alternate flows	1a. Advisee / Advisor / Administrator enter the platform using dashboard URL. 1b. Advisee / Advisor / Administrator click on the “Dashboard” link in the navigation bar.

Table 3.3.4 View Profile Use Case Description

Use Case ID	00004
Use Case Name	View Profile

Brief Description	Profile will be showing the personal information and calendar of the advisee / advisor / administrator.
Actor	Advisee, Advisor, Administrator
Trigger	Clicked on “Profile” link from the side navigation bar or from the dropdown menu on the top navigation bar.
Precondition	Account must be logged in.
Normal flow of events	1. Advisee / Advisor / Administrator click on the “Profile” link in the navigation bar or dropdown menu. 2. Advisee / Advisor / Administrator can then see their personal details, such as name, faculty, phone number, email address and profile photo. Advisee can see their own course, CGPA, credit hours, and current semester, whereas advisor and administrator can view their department. 3. Advisee / Advisor / Administrator can also view their own calendar and plan their schedules.
Sub flows	-
Alternate flows	1a. Advisee / Advisor / Administrator enter the platform using profile URL.

Table 3.3.5 Edit Profile Use Case Description

Use Case ID	00005
Use Case Name	Edit Profile
Brief Description	Advisee / Advisor / administrator can edit limited information in their profile, such as their phone number and address.
Actor	Advisee, Advisor, Administrator
Trigger	Clicked on edit profile button in the profile page.
Precondition	Account must be logged in and is in profile page.
Normal flow of events	1. Advisee / Advisor / Administrator click on the edit profile button. 2. There will be pop up modal and advisee / advisor / administrator can key in their information. 3. Advisee / Advisor / Administrator click on the save button. 4. New information is saved and updated after the page auto refreshed.
Sub flows	-
Alternate flows	-

Table 3.3.6 Add Calendar Event Use Case Description

Use Case ID	00006
Use Case Name	Add Calendar Event
Brief Description	Advisee / Advisor / Administrator can add their own events in their calendar.
Actor	Advisee, Advisor, Administrator
Trigger	Clicked on "Add Event" link in dropdown menu of the calendar in profile page.
Precondition	Account must be logged in and is in profile page.
Normal flow of events	1. Advisee / Advisor / Administrator click on the "+" button to toggle dropdown menu of calendar. 2. Advisee / Advisor / Administrator click on the "Add Event" link in the dropdown menu. 3. There will be a form in a popup modal for advisee / advisor / administrator to fill in the event details, including the title, start date and time, end date and time and description. 4. Advisee / Advisor / Administrator click on the save button. 5. New event is saved and advisee / advisor / administrator can view the new event in their calendar after the auto refresh of the page.
Sub flows	-
Alternate flows	3a. If the required column in the form is not completed, the system will display message at the column to inform the advisee / advisor / administrator. 3b. If the end date of the event is earlier than the start date, the system will show an alert message to the advisee / advisor / administrator and let them edit the details. 3c. If the new event has overlapped an existing event or a scheduled booking, the system will show an alert message to the advisee / advisor / administrator and let them edit the details.

Table 3.3.7 Edit Calendar Event Use Case Description

Use Case ID	00007
Use Case Name	Edit Calendar Event
Brief Description	Advisee / Advisor / Administrator can edit their own events in their calendar.
Actor	Advisee, Advisor, Administrator

Trigger	Clicked on the edit button in event detail popup modal.
Precondition	Account must be logged in and is in profile page.
Normal flow of events	1. Advisee / Advisor / Administrator click on the particular event in the calendar. 2. There will be a popup modal for advisee / advisor / administrator to view the event details, including the title, start time, end time and description. 3. Advisee / Advisor / Administrator click on the edit button. 4. There will be a form in a popup modal for advisee / advisor / administrator to edit the event details, including the title, start date and time, end date and time and description. 5. Advisee / Advisor / Administrator click on the save button. 6. Event is edited, and advisee / advisor / administrator can view the edited event in their calendar after the auto refresh of the page.
Sub flows	-
Alternate flows	4a. If the required column in the form is not completed, the system will display message at the column to inform the advisee / advisor / administrator. 4b. If the end date of the event is earlier than the start date, the system will show an alert message to the advisee / advisor / administrator and let them edit the details. 4c. If the edited event has overlapped an existing event or a scheduled booking, the system will show an alert message to the advisee / advisor / administrator and let them edit the details.

Table 3.3.8 Delete Calendar Event Use Case Description

Use Case ID	00008
Use Case Name	Delete Calendar Event
Brief Description	Advisee / Advisor / Administrator can delete their own events in their calendar.
Actor	Advisee, Advisor, Administrator
Trigger	Clicked on the delete button in event detail popup modal.
Precondition	Account must be logged in and is in profile page.
Normal flow of events	1. Advisee / Advisor / Administrator click on the particular event in the calendar.

	2. There will be a popup modal for advisee / advisor / administrator to view the event details, including the title, start time, end time and description. 3. Advisee / Advisor / Administrator click on the delete button. 4. There will be a popup modal for advisee / advisor / administrator to confirm that they want to delete the event. 5. Advisee / Advisor / Administrator click on the confirm delete button. 6. Event is deleted, and advisee / advisor / administrator can view the event is deleted from their calendar after the auto refresh of the page.
Sub flows	-
Alternate flows	-

Table 3.3.9 View Advisor Use Case Description

Use Case ID	00009
Use Case Name	View Advisor
Brief Description	Advisee can view their advisor details and calendar in the advisor profile page, and also booking details.
Actor	Advisee
Trigger	Clicked on the “Advisor” link in the side navigation bar.
Precondition	Account must be logged in and must be an advisee’s account.
Normal flow of events	1. Advisee click on the “Advisor” link in the side navigation bar. 2. Advisee can see the personal details of his assigned advisor, including name, faculty, department, phone number, email address and profile photo. 3. Advisee can view the working hours calendar of their advisor. 4. Advisee can also book meetings in the advisor page. 5. Previous meeting notes can also be seen in the advisor page.
Sub flows	-
Alternate flows	1a. Advisee enter the platform using advisor URL.

Table 3.3.10 View Advisees Use Case Description

Use Case ID	00010
Use Case Name	View Advisees

Brief Description	Advisor can view their advisee details and calendar in the advisee profile page, and also booking details.
Actor	Advisor
Trigger	Clicked on the “Advisees” link in the side navigation bar.
Precondition	Account must be logged in and must be an advisor’s account.
Normal flow of events	1. Advisor click on the “Advisees” link in the navigation bar. 2. Advisor will be redirected to “Manage Advisees” Page. 3. Advisor can view a table which includes the details of the all the advisees under him. 4. Advisor can click on the particular advisee’s ID and will be redirected to the profile of the advisee. 5. Advisee can see the personal details of his assigned advisor, including name, faculty, course, phone number, email address, CGPA, current semester, credit hours and profile photo. 6. Advisor can also view bookings, manage bookings and perform booking of meeting in the advisee page. 7. Previous meeting notes can be seen in the advisee page and advisor can add a new meeting notes in the advisee page.
Sub flows	-
Alternate flows	1a. Advisor enter the platform using advisees/:student_id URL.

Table 3.3.11 View Users Use Case Description

Use Case ID	00011
Use Case Name	View Users (Advisors and Advisees)
Brief Description	Administrator can view details of all advisees and advisors.
Actor	Administrator
Trigger	Clicked on the “Manage Users” link in the side navigation bar.
Precondition	Account must be logged in and must be an administrator’s account.
Normal flow of events	1. Administrator click on the “Manage Users” link in the navigation bar and will be redirected to “Manage Users” Page. 2. Administrator can view one table of advisors and another table of advisees. 3. Administrator can perform add user, edit user and delete user in the “Manage Users” page. 4. Administrator can also click on the user’s ID and will be redirected to the profile of the user.

	5. Administrator can view the personal details of the users.
Sub flows	-
Alternate flows	1a. Advisee enter the platform using user URL.

Table 3.3.12 Add Users Use Case Description

Use Case ID	00012
Use Case Name	Add users (Advisors and Advisees)
Brief Description	Administrator can add new advisor or advisee in the system.
Actor	Administrator
Trigger	Clicked on the “Add Advisor” or “Add Advisees” button in the “Manage Users” page.
Precondition	An administrator’s account must be logged in and in the “Manage Users” page.
Normal flow of events	1. Administrator click on the “Add Advisor” or “Add Advisees” button in the “Manage Users” page. 2. Administrator will be redirected to “Add New Advisor” or “Add New Advisee” page. 3. Administrator key in the details of the user in the form of the page. 4. Administrator click on save button. 5. Administrator will be redirected to the “Manage Users” page and the user will be added.
Sub flows	-
Alternate flows	-

Table 3.3.13 Edit Users Use Case Description

Use Case ID	00013
Use Case Name	Edit users (Advisors and Advisees)
Brief Description	Administrator can edit details of all advisees and advisors.
Actor	Administrator
Trigger	Clicked on the edit button in the table row of the user.
Precondition	An administrator’s account must be logged in and in the “Manage Users” page.
Normal flow of events	1. Administrator click on the edit button in the same row of the user in the table.

	2. Administrator will be redirected to "Edit Advisor" or "Edit Advisee" page. 3. Administrator key in the details of the user in the form of the page. 4. Administrator click on save button. 5. Administrator will be redirected to the "Manage Users" page and the edited information is updated.
Sub flows	-
Alternate flows	-

Table 3.3.14 Delete Users Use Case Description

Use Case ID	00014
Use Case Name	Delete users (Advisors and Advisees)
Brief Description	Administrator can delete advisor or advisee.
Actor	Administrator
Trigger	Clicked on the delete button in the table row of the user.
Precondition	An administrator's account must be logged in and in the "Manage Users" page.
Normal flow of events	1. Administrator click on the delete button in the same row of the user in the table. 2. There will be a popup modal to let administrator double confirm to delete the user. 3. Administrator click on confirm delete button. 4. Administrator will be redirected to the "Manage Users" page and the user table is updated.
Sub flows	-
Alternate flows	-

Table 3.3.15 Book Meetings Use Case Description

Use Case ID	00015
Use Case Name	Book Meetings
Brief Description	Advisee and advisor can book meetings with the opposite side.
Actor	Advisee, Advisor
Trigger	Click on the "Book Meeting" link in the dropdown of "My Bookings with Advisor"/ "Scheduled Bookings with Advisee" table.

Precondition	Advisee’s account must be logged in and in the “Advisor” page / Advisor’s account must be logged in and in the “Advisee” page.
Normal flow of events	1. Advisee / Advisor click on the “Book Meeting” link in the dropdown of “My Bookings with Advisor” table. 2. There will be a popup modal to let advisee / advisor fill in the details of the booking including the purpose of meeting, title, start date and time, end date and time, and description. 3. Advisee / Advisor click on submit button. 4. The booking will be added.
Sub flows	-
Alternate flows	2a. There will be an alert if users fill in bookings before the current time. 2b. If the booking end date is earlier than the start date, the system will show alert to the advisee and advisee will need to edit the details. 2c. If the booking is not in office hours (Monday to Friday, 8.00am – 6.00pm), the system will show alert to inform the advisee / advisor to edit the details. 2d. If the advisee / advisor already has a personal event overlapping the booking time, there will be alert to inform the advisee / advisor to edit the details. 2e. If the advisor already has scheduled meetings overlapping the booking time, there will be alert to inform the advisee / advisor to edit the details.

Table 3.3.16 Edit Bookings Use Case Description

Use Case ID	00016
Use Case Name	Edit Bookings
Brief Description	Advisee / Advisor can edit the book meetings
Actor	Advisee / Advisor
Trigger	Click on the edit button on the same row of the event in the “My Bookings with Advisor” / “Scheduled Bookings with Advisee” table.
Precondition	Advisee’s account must be logged in and in the “Advisor” page / Advisor’s account must be logged in and in the “Advisee” page.
Normal flow of events	1. Advisee / Advisor click on the edit button on the same row of the event in the “My Bookings with Advisor” / “Scheduled Bookings with Advisee” table.

	2. There will be a popup modal to let advisee / advisor edit the details of the booking including the purpose of meeting, title, start date and time, end date and time, and description. 3. Advisee / Advisor click on save button. 4. The booking will be edited and the event will be updated to pending status.
Sub flows	-
Alternate flows	1a. If the current time has passed the start time of the booking, the booking can no longer be updated. 2a. There will be an alert if users fill in bookings before the current time. 2b. If the booking end date is earlier than the start date, the system will show alert to the advisee and advisee will need to edit the details. 2c. If the booking is not in office hours (Monday to Friday, 8.00am – 6.00pm), the system will show alert to inform the advisee to edit the details. 2d. If the advisee / advisor already has a personal event overlapping the booking time, there will be alert to inform the advisee / advisor to edit the details. 2e. If the advisor already has scheduled meetings overlapping the booking time, there will be alert to inform the advisee / advisor to edit the details.

Table 3.3.17 Delete Booking Use Case Description

Use Case ID	00017
Use Case Name	Delete Booking
Brief Description	Advisee / Advisor can delete the booking.
Actor	Advisee / Advisor
Trigger	Click on the delete button on the same row of the event in the “My Bookings with Advisor” / “Scheduled Bookings with Advisee” table.
Precondition	Advisee’s account must be logged in and in the “Advisor” page.
Normal flow of events	1. Advisee / Advisor click on the delete button on the same row of the event in the “My Bookings with Advisor” / Scheduled Bookings with Advisee” table. 2. There will be a popup modal to let advisee / advisor to confirm to delete the booking.

	3. Advisee / Advisor click on confirm delete button. 4. The booking will be deleted.
Sub flows	-
Alternate flows	-

Table 3.3.18 Manage Bookings Use Case Description

Use Case ID	00018
Use Case Name	Manage Bookings
Brief Description	Advisee / Advisor can accept or reject the bookings.
Actor	Advisee / Advisor
Trigger	Click on the tick button or "x" button on the same row of the event in "My Bookings with Advisor" / Scheduled Bookings with Advisee" table.
Precondition	Advisee's account must be logged in and in the "Advisor" page / Advisor's account must be logged in and in the "Advisee" page.
Normal flow of events	1. "My Bookings with Advisor" / Scheduled Bookings with Advisee" table will show all bookings between the advisor and advisee and a tick button and "x" button if the booking status is pending. 2. If the event is pending, advisee / advisor can click on the tick button to accept the booking, or press the "x" button, to reject the booking. 3. The booking status will be updated from pending to accepted if the advisee / advisor accepts the event, and from pending to rejected if the advisee / advisor rejects the event, and the last updated time will also be updated.
Sub flows	2a. When the advisee / advisor clicks the "x" button, there will be a modal popup to let advisee / advisor fill in the reason of rejecting the event, and need to be submitted to reject the booking.
Alternate flows	1a. Advisor can choose to click on the "Bookings" link in the side navigation bar to redirect to "Bookings" page, and manage meetings booked by all advisees. 1b. If the event status is accepted, the table will only show a "x" button. 1c. If the event status is rejected, the table will only show a tick button.

	2a. If the advisee / advisor clicks on the tick button of the booking, and the booking has overlapped an existing event of the advisor, there will be an alert to let the advisor to reject the booking. 2b. If the advisee / advisor clicks on the tick button of the booking, and the booking has overlapped a pending booking or an accepted booking, there will be a confirmation alert to let advisor to confirm accept this booking and reject the overlapped booking.

Table 3.3.19 Add Meeting Notes Use Case Description

Use Case ID	00019
Use Case Name	Add Meeting Notes
Brief Description	Advisor can add meeting notes for the accepted bookings.
Actor	Advisor
Trigger	Click on the “Add Meeting Notes” link in the dropdown of “Meeting Notes” table in profile of the advisee.
Precondition	Advisor’s account must be logged in and is inside the profile of the advisee.
Normal flow of events	1. Advisor clicks on the “Add Meeting Notes” link in the dropdown of “Meeting Notes” table in profile of the advisee. 2. There will be a popup modal to let the advisor choose which accepted events to add meeting notes for. 3. Advisor click ok to confirm, and will be redirected to the “Add Meeting Notes” page. 4. Advisor can key in the details of the meeting into the form and click submit. 5. The status of the booking will be updated from accepted to completed after the advisor submitted the meeting notes.
Sub flows	-
Alternate flows	3a. If the booking type is academic advising, there will be a template form for the advisor to fill in. 3b. If the booking type is general consultation, there will be a textbox for advisor to fill in with any format they want.

Table 3.3.20 View Meeting Notes Use Case Description

Use Case ID	00020
Use Case Name	View Meeting Notes
Brief Description	Advisee / Advisor can add meeting notes for the accepted bookings.
Actor	Advisee, Advisor
Trigger	Access to the profile page of the advisee.
Precondition	Advisee's / Advisor's account must be logged in and is inside the profile of the advisee.
Normal flow of events	1. There will be a table called "Meeting Notes" in the profile of the advisee. 2. Advisee / Advisor can view the meeting note summary in the table. 3. Advisee / Advisor can click the ID of the meeting note to and redirected to the "Meeting Note Details" to view the details.
Sub flows	-
Alternate flows	-

Table 3.3.21 Use CGPA Calculator Use Case Description

Use Case ID	00021
Use Case Name	Use CGPA Calculator
Brief Description	Advisee / Advisor can use the CGPA Calculator to predict the CGPA obtained by the advisee.
Actor	Advisee, Advisor
Trigger	Click on the "CGPA calculator" link in the navigation bar.
Precondition	Advisee's / Advisor's account must be logged in.
Normal flow of events	1. Advisee / Advisor click on the "CGPA calculator" link in the navigation bar and will be redirected to the CGPA calculator page. 2. Advisee / Advisor key in the prior CGPA and credits earned for the previous semesters, and the credit hour and predicted grade for the current semester. 3. Advisee / Advisor click on the calculate button. 4. The calculator will then show the GPA of current semester and the CGPA achieved.
Sub flows	2a. Advisee / Advisor can choose the course code and the credit hour column will automatically be filled in.

	2b. Advisee / Advisor need to click the "add row" button to add a new row for inserting another course, credit hour and grade.
Alternate flows	3a. Advisee / Advisor can click on the reset button to reset the page.

Table 3.3.22 Use Target CGPA Calculator Use Case Description

Use Case ID	00022
Use Case Name	Use Target CGPA Calculator
Brief Description	Advisee / Advisor can use the target CGPA Calculator to estimate the required GPA of the remaining credit hours to obtain the target CGPA of the advisee.
Actor	Advisee, Advisor
Trigger	Click on the "Target CGPA calculator" link in the navigation bar.
Precondition	Advisee's / Advisor's account must be logged in.
Normal flow of events	1. Advisee / Advisor click on the "Target CGPA calculator" link in the navigation bar and will be redirected to the Target CGPA calculator page. 2. Advisee / Advisor key in the prior CGPA and credits earned for the previous semesters, and the target CGPA and credit hours remaining of the upcoming semester(s). 3. Advisee / Advisor click on the calculate button. 4. The calculator will then show the GPA needed to obtain for all remaining credits.
Sub flows	-
Alternate flows	3a. Advisee / Advisor can click on the reset button to reset the page.

Table 3.3.23 Use Chatbot Use Case Description

Use Case ID	00023
Use Case Name	Use Chatbot
Brief Description	Advisee can use the chatbot to enquire their questions.
Actor	Advisee
Trigger	Click on the floating chat button on the bottom right of the platform.
Precondition	Advisee's account must be logged in.
Normal flow of events	1. Advisee clicks on the floating chat button on the bottom right of the platform. 2. Advisee can start to chat with the chatbot and ask questions.

	3. The chatbot will send the solution for the advisee.
Sub flows	-
Alternate flows	3a. The chatbot will suggest advisee to contact advisor if the chatbot does not know how to solve the question.

Table 3.3.24 View Chatbot Analysis Use Case Description

Use Case ID	00024
Use Case Name	View Chatbot Analysis
Brief Description	Advisor / Administrator can view the analysis of the enquiries asked by the advisees.
Actor	Advisor / Administrator
Trigger	View in the dashboard page.
Precondition	Advisor's / Administrator's account must be logged in and be in the dashboard page.
Normal flow of events	1. Advisor / Administrator enter the platform using default URL and redirected to dashboard. 2. There will be a table which shows the statistics of the analysis of the questions asked by the advisees.
Sub flows	-
Alternate flows	1a. Advisor / Administrator enter the platform using dashboard URL. 1b. Advisor / Administrator click on the "Dashboard" link in the navigation bar. 1c. Advisor / Administrator click on the Enquiry Analysis tab.

Table 3.3.25 Add Announcement Use Case Description

Use Case ID	00025
Use Case Name	Add announcements
Brief Description	Administrator can add announcements that will be shown in the dashboard of all users.
Actor	Administrator
Trigger	Click the "Add Announcement" button in dropdown of "Announcements" table.
Precondition	Administrator's account must be logged in and be in the "Announcements" page.

Normal flow of events	1. Administrator click on "Add Announcement" button in dropdown of "Announcements" table. 2. Administrator will be redirected to "Add Announcement" page. 3. Administrator key in all details of the announcement in the form. 4. Administrator click on the submit button. 5. The new announcement is saved and reflected in the dashboard of the users.
Sub flows	-
Alternate flows	-

Table 3.3.26 Edit announcements Use Case Description

Use Case ID	00026
Use Case Name	Edit announcements
Brief Description	Administrator can edit existing announcements that will be shown in the dashboard of all users.
Actor	Administrator
Trigger	Click the edit button in same row of the announcement in the table.
Precondition	Administrator's account must be logged in and be in the "Announcements" page.
Normal flow of events	1. Administrator click on edit button in same row of the announcement in the table. 2. Administrator will be redirected to "Edit Announcement" page. 3. Administrator edit the details of the announcement in the form. 4. Administrator click on the save button. 5. The edited announcement is saved and reflected in the dashboard of the users.
Sub flows	-
Alternate flows	-

Table 3.3.27 Delete Announcements Use Case Description

Use Case ID	00027
Use Case Name	Delete announcements
Brief Description	Administrator can delete existing announcements that will be shown in the dashboard of all users.
Actor	Administrator

Trigger	Click the delete button in same row of the announcement in the table.
Precondition	Administrator's account must be logged in and be in the "Announcements" page.
Normal flow of events	1. Administrator click on delete button in same row of the announcement in the table. 2. There will be a popup modal to let administrator confirm to delete the announcement. 3. Administrator click on the confirm delete button. 4. The announcement is deleted.
Sub flows	-
Alternate flows	-

Table 3.3.28 Chat Use Case Description

Use Case ID	00028
Use Case Name	Chat
Brief Description	Advisee and advisor can use the chat to send message to the opposite side.
Actor	Advisee, Advisor
Trigger	Click on the message button at the top navigation bar
Precondition	Advisee or advisor's account must be logged in.
Normal flow of events	1. Advisee/ Advisor clicks on the message button at the top navigation bar, and will be redirected to the message page 2. Advisee / Advisor can start to chat with the opposite side. 3. After pressing enter, the message is then sent out to the opposite side.
Sub flows	-
Alternate flows	2a. Advisor can choose which advisee to chat with.

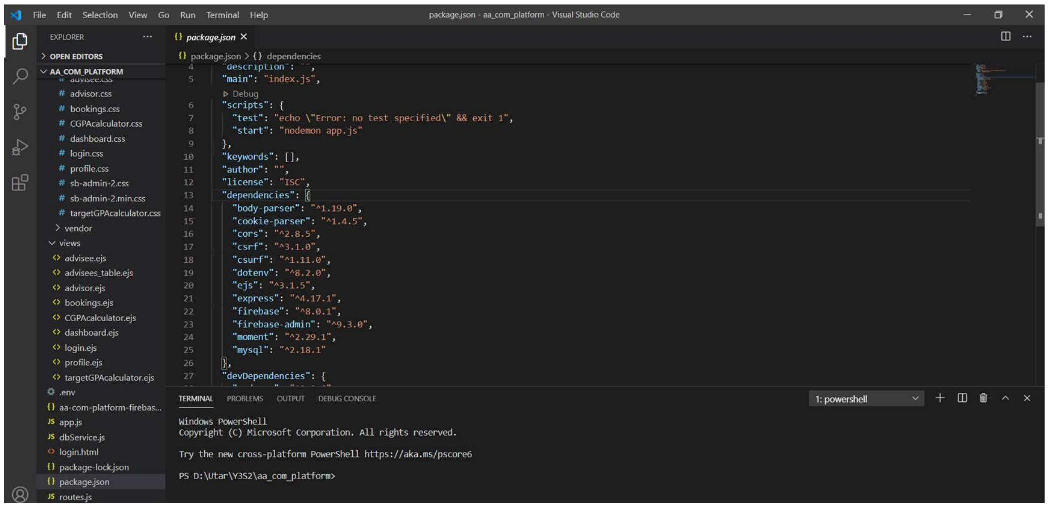
3.4 Website Development

3.4.1 Server-side development

The website is developed for both client side and server side. For client side, HTML, CSS and JavaScript is used, and Node.js serves as the server-side framework to connect back end with the front end of the website, such as passing the database from MySQL and also redirecting the website to different pages. As in Figure 3.3.3.1, all the dependencies downloaded using Node.js for this project is listed here. Dotenv parses the .env file which contains variables into the environment of the application, and in this project, the database information is stored in the .env file. EJS is a template engine used in this project and it provides user to create HTML using plain JavaScript, and it can be used to pass data from the database from the back end to HTML easily during the render of the page. The HTML is inserted in .ejs file and are stored in views folder.

The main code controlling the backend is in the app.js file, which initialise the Firebase admin SDK, set EJS as the view engine, set body parser in order to handle HTTP POST request, public directory to get the CSS and JavaScript file from front end, get the routes from the routes.js file, and also attaching the CSRF token for user when they access the website. App.js also serves to listen the connection of the host and port.

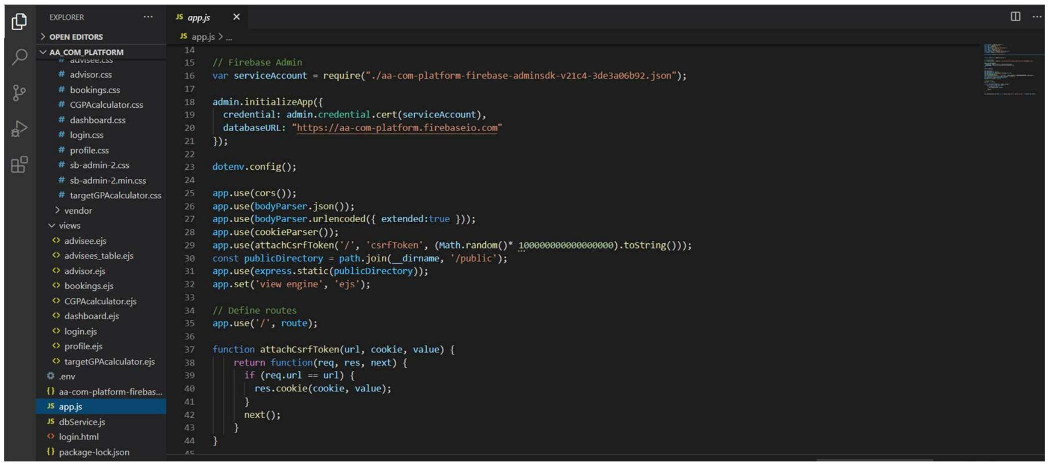
All the routes of the project are defined in the routes.js file. It is used to get GET, POST, PATCH and DELETE request from the front end, and perform actions such as interacting with the database and passing the data obtained back to the front end in JSON or redirecting the user to a path by rendering the page after checking the user's session cookie. DbService.js provides database service by connecting to MySQL aa_com_platform database, and users can perform query, insert, update or delete actions.



```
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "cookie-parser": "^1.4.5",
    "cors": "^2.8.5",
    "csrf": "^3.1.0",
    "csurf": "^1.11.0",
    "dotenv": "^8.2.0",
    "ejs": "^3.1.5",
    "express": "^4.17.1",
    "firebase": "^8.0.1",
    "firebase-admin": "^9.3.0",
    "moment": "^2.29.1",
    "mysql": "^2.18.1"
  },
  "devDependencies": {
```

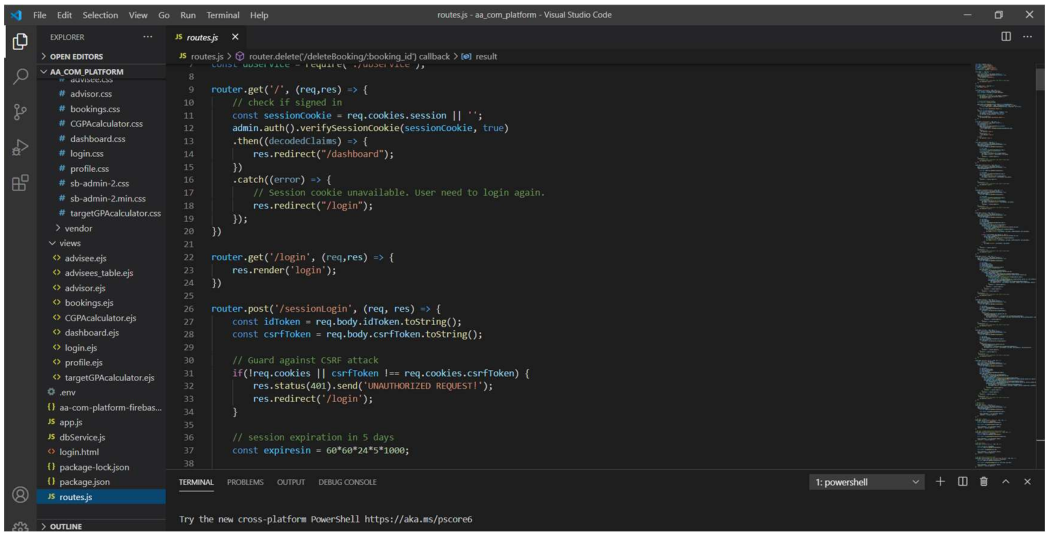
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Utar\Y3S2\aa_com_platform>
```

Figure 3.4.1.1 Package.json



```
// Firebase Admin
var serviceAccount = require("./aa-com-platform-firebase-adminsdk-v21c4-3de3a06b92.json");

admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL: "https://aa-com-platform.firebaseio.com"
});

dotenv.config();

app.use(cors());
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended:true }));
app.use(cookieParser());
app.use(attachCsrfToken('/', 'csrfToken', (Math.random()* 100000000000000000).toString()));
const publicDirectory = path.join(__dirname, '/public');
app.use(express.static(publicDirectory));
app.set('view engine', 'ejs');

// Define routes
app.use('/', route);

function attachCsrfToken(url, cookie, value) {
    return function(req, res, next) {
      if (req.url == url) {
        res.cookie(cookie, value);
      }
      next();
    }
}
```

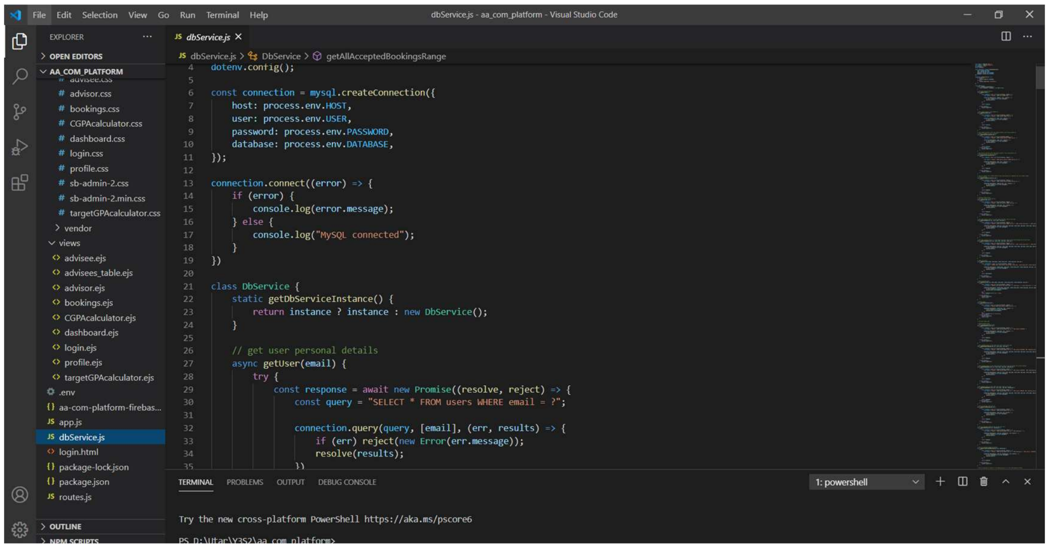
Figure 3.4.1.2 App.js



```
router.get('/', (req,res) => {
    // check if signed in
    const sessionCookie = req.cookies.session || '';
    admin.auth().verifySessionCookie(sessionCookie, true)
    .then((decodedClaims) => {
        res.redirect("/dashboard");
    })
    .catch((error) => {
        // Session cookie unavailable. User need to login again.
        res.redirect("/login");
    });
})

router.get('/login', (req,res) => {
    res.render('login');
})

router.post('/sessionLogin', (req, res) => {
    const idToken = req.body.idToken.toString();
    const csrfToken = req.body.csrfToken.toString();

    // Guard against CSRF attack
    if(!req.cookies || csrfToken !== req.cookies.csrfToken) {
        res.status(401).send('UNAUTHORIZED REQUEST!');
        res.redirect('/login');
    }

    // session expiration in 5 days
    const expiresin = 60*60*24*5*1000;
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6
```

Figure 3.4.1.3 Routes.js



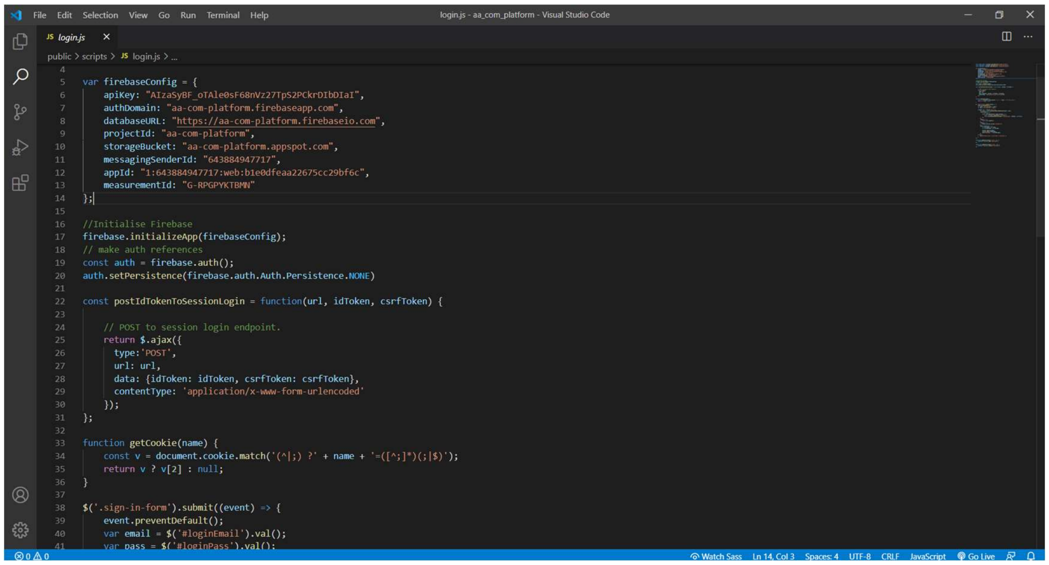
```
dotenv.config();

const connection = mysql.createConnection({
    host: process.env.HOST,
    user: process.env.USER,
    password: process.env.PASSWORD,
    database: process.env.DATABASE,
});

connection.connect((error) => {
    if (error) {
        console.log(error.message);
    } else {
        console.log("MySQL connected");
    }
})

class DbService {
    static getDbServiceInstance() {
        return instance ? instance : new DbService();
    }

    // get user personal details
    async getUser(email) {
        try {
            const response = await new Promise((resolve, reject) => {
                const query = "SELECT * FROM users WHERE email = ?";

                connection.query(query, [email], (err, results) => {
                    if (err) reject(new Error(err.message));
                    resolve(results);
                })
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Utar\Y3S2\aa_com_platform>
```

Figure 3.4.1.4 dbService.js

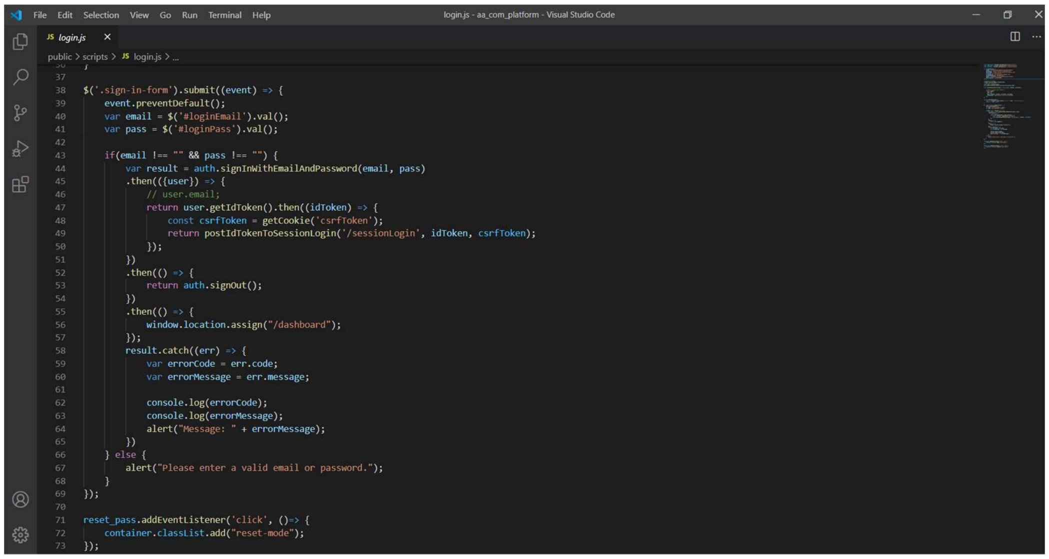
3.4.2 Login Function Development

Firebase is first initialized in order to connect Firebase authentication to the project for the login function. When users key in their email and password in the login page, the user will be signed in using the sign in with email and password function of Firebase authentication. Then, the function will return an ID token. The ID token and the CSRF token provided by the system will be POST to the backend through /sessionLogin URL to verify the ID token and provide a session cookie for the users which will be expiring in 5 days, and the user will remain login within 5 days after login.



```javascript

var firebaseConfig = {
    apiKey: "AIzaSyBF_oTAle0sF68nVz27TpS2PCkrDIbDIaI",
    authDomain: "aa-com-platform.firebaseapp.com",
    databaseURL: "https://aa-com-platform.firebaseio.com",
    projectId: "aa-com-platform",
    storageBucket: "aa-com-platform.appspot.com",
    messagingSenderId: "643884947717",
    appId: "1:643884947717:web:b1e0dfeaa22675cc29bf6c",
    measurementId: "G-RPGPYKTBMN"
};

//Initialise Firebase
firebase.initializeApp(firebaseConfig);
// make auth references
const auth = firebase.auth();
auth.setPersistence(firebase.auth.Auth.Persistence.NONE)

const postIdTokenToSessionLogin = function(url, idToken, csrfToken) {

    // POST to session login endpoint.
    return $.ajax({
      type:'POST',
      url: url,
      data: {idToken: idToken, csrfToken: csrfToken},
      contentType: 'application/x-www-form-urlencoded'
    });
};

function getCookie(name) {
    const v = document.cookie.match('(^|;) ?' + name + '=([^;]*)(;|$)');
    return v ? v[2] : null;
}

$('.sign-in-form').submit((event) => {
    event.preventDefault();
    var email = $('#loginEmail').val();
    var pass = $('#loginPass').val();
```

Figure 3.4.2.1 Login.js

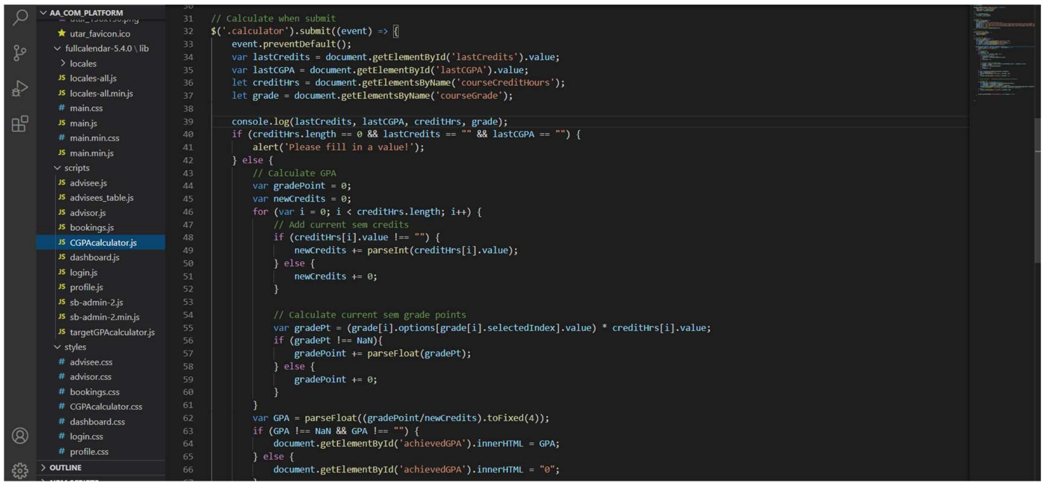


```javascript

$('.sign-in-form').submit((event) => {
    event.preventDefault();
    var email = $('#loginEmail').val();
    var pass = $('#loginPass').val();

    if(email !== "" && pass !== "") {
        var result = auth.signInWithEmailAndPassword(email, pass)
        .then(({user}) => {
            // user.email;
            return user.getIdToken().then((idToken) => {
                const csrfToken = getCookie('csrfToken');
                return postIdTokenToSessionLogin('/sessionLogin', idToken, csrfToken);
            });
        })
        .then(() => {
            return auth.signOut();
        })
        .then(() => {
            window.location.assign("/dashboard");
        });
        result.catch((err) => {
            var errorCode = err.code;
            var errorMessage = err.message;

            console.log(errorCode);
            console.log(errorMessage);
            alert("Message: " + errorMessage);
        })
    } else {
        alert("Please enter a valid email or password.");
    }
});

reset_pass.addEventListener('click', ()=> {
    container.classList.add("reset-mode");
});
```

Figure 3.4.2.2 Login.js Function after Submitting Sign in Form

3.4.3 CGPA Calculator Development

The CGPA calculator will identify all the values key in by the users in the calculator when the user clicks on the submit button. The user will be alerted to fill in a value if the form submitted is empty, else the calculation of the CGPA will proceed. The grade point of each course of the semester is calculated by multiplying the estimated grades of each course with its credit hours. The credit hours of all the inserted courses will be added up to calculate the GPA of the semester, by dividing total credits from the total grade points. The CGPA is also calculated by adding up the previous grade points with the current grade points, and dividing the total credits from previous semesters and current semester. The value of estimated GPA and CGPA is then shown to the user.

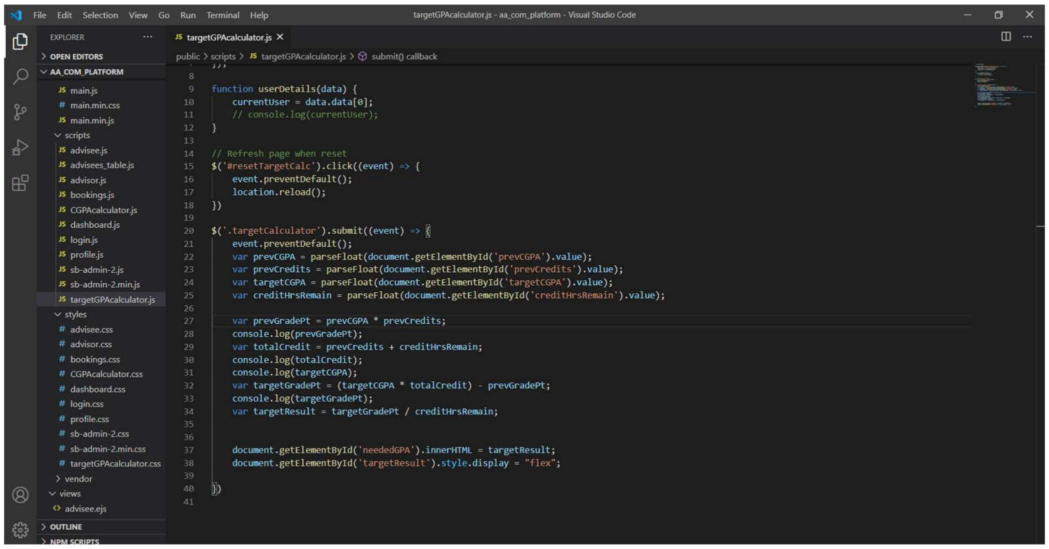


```
// Calculate when submit
$('.calculator').submit((event) => {
    event.preventDefault();
    var lastCredits = document.getElementById('lastCredits').value;
    var lastCGPA = document.getElementById('lastCGPA').value;
    let creditHrs = document.getElementsByName('courseCreditHours');
    let grade = document.getElementsByName('courseGrade');

    console.log(lastCredits, lastCGPA, creditHrs, grade);
    if (creditHrs.length == 0 && lastCredits == "" && lastCGPA == "") {
        alert('Please fill in a value!');
    } else {
        // Calculate GPA
        var gradePoint = 0;
        var newCredits = 0;
        for (var i = 0; i < creditHrs.length; i++) {
            // Add current sem credits
            if (creditHrs[i].value !== "") {
                newCredits += parseInt(creditHrs[i].value);
            } else {
                newCredits += 0;
            }

            // Calculate current sem grade points
            var gradePt = (grade[i].options[grade[i].selectedIndex].value) * creditHrs[i].value;
            if (gradePt !== NaN){
                gradePoint += parseFloat(gradePt);
            } else {
                gradePoint += 0;
            }
        }
        var GPA = parseFloat((gradePoint/newCredits).toFixed(4));
        if (GPA !== NaN && GPA !== "") {
            document.getElementById('achievedGPA').innerHTML = GPA;
        } else {
            document.getElementById('achievedGPA').innerHTML = "0";
```

Figure 3.4.3.1 CGPAcalcultator.js

3.4.4 Target CGPA Calculator Development

When users fill in all the details in the form of target CGPA calculator, the previous grade point is calculated using previous CGPA, multiply by the previous credit hours obtained. The total credit hour is then calculated and used to obtain the target grade points needed to achieve the target CGPA. The target result is then being calculated by dividing the target grade points with the credit hours remaining.

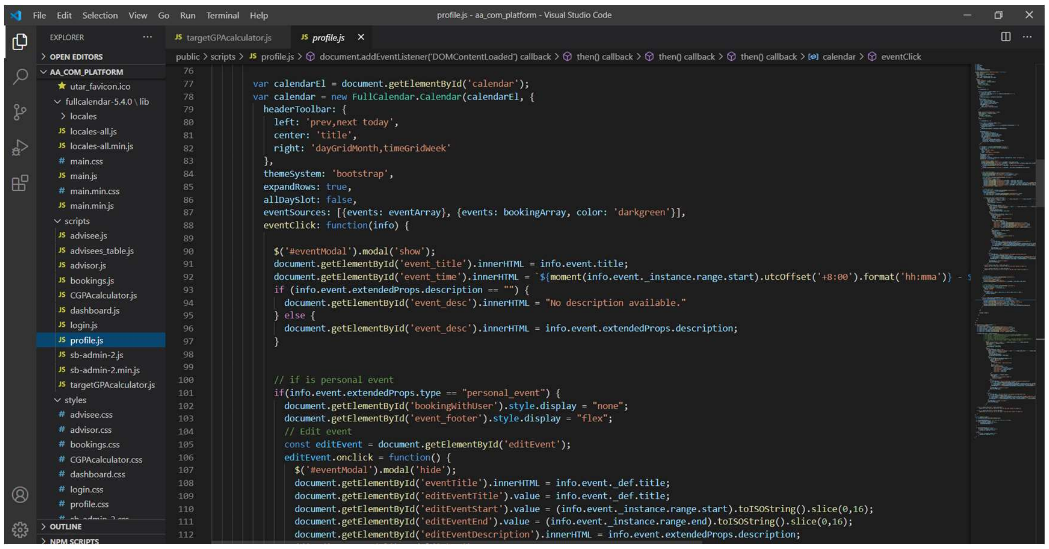


```
function userDetails(data) {
    currentUser = data.data[0];
    // console.log(currentUser);
}

// Refresh page when reset
$('#resetTargetCalc').click((event) => {
    event.preventDefault();
    location.reload();
})

$('.targetCalculator').submit((event) => {
    event.preventDefault();
    var prevCGPA = parseFloat(document.getElementById('prevCGPA').value);
    var prevCredits = parseFloat(document.getElementById('prevCredits').value);
    var targetCGPA = parseFloat(document.getElementById('targetCGPA').value);
    var creditHrsRemain = parseFloat(document.getElementById('creditHrsRemain').value);

    var prevGradePt = prevCGPA * prevCredits;
    console.log(prevGradePt);
    var totalCredit = prevCredits + creditHrsRemain;
    console.log(totalCredit);
    console.log(targetCGPA);
    var targetGradePt = (targetCGPA * totalCredit) - prevGradePt;
    console.log(targetGradePt);
    var targetResult = targetGradePt / creditHrsRemain;


    document.getElementById('neededGPA').innerHTML = targetResult;
    document.getElementById('targetResult').style.display = "flex";

})
```

Figure 3.4.4.1 targetGPAcalculator.js

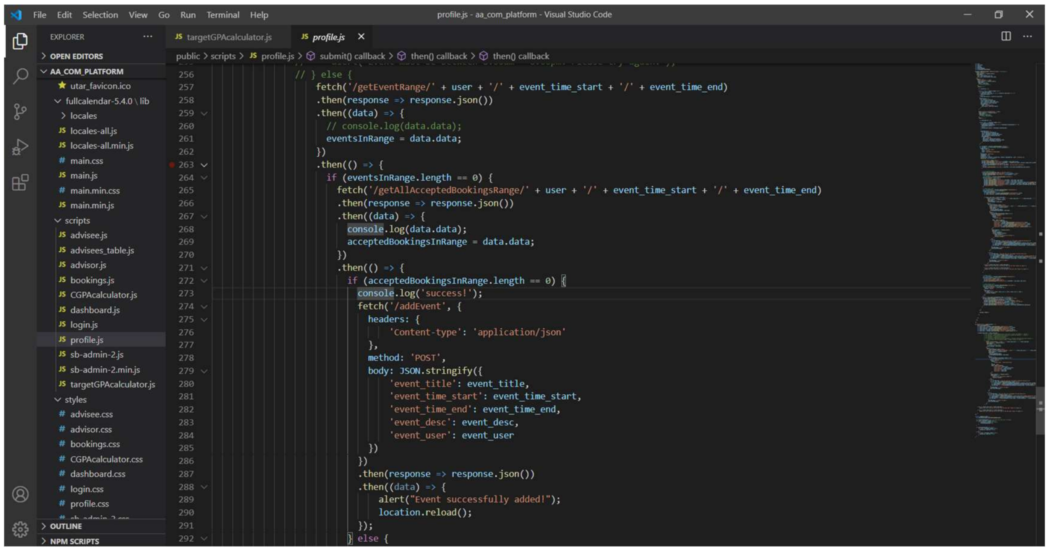
3.4.5 Event Calendar Development

The event calendar shown in the profile page, advisor and advisee page is done using the FullCalendar API. The events and bookings are first obtained from the database using the user ID, and then passed as the event sources for the calendar. The events in the default colour, blue, is personal event of the user, and for the green colour events in the calendar is consultation or academic advising bookings. By clicking on the events, the details of the event will be shown in a pop-up modal, and user can edit or delete the event if it is his or her own event. If the user performs editing, the start time and end time will first be verified that it does not overlap the personal event and booking of the users. After verifying, the details of the event are passed to the backend using HTTP PATCH and MySQL update statement is done by using the id of the event. If user choose to delete the event, the event id is passed through HTTP DELETE and MySQL will perform delete statement using the event id. During adding of an event, the process of verifying the time is same as the event edit, and after verifying, the system will be using HTTP POST to pass all data to the backend and perform MySQL insert statement.



```
var calendarEl = document.getElementById('calendar');
var calendar = new FullCalendar.Calendar(calendarEl, {
  headerToolbar: {
    left: 'prev,next today',
    center: 'title',
    right: 'dayGridMonth,timeGridWeek'
  },
  themeSystem: 'bootstrap',
  expandRows: true,
  allDaySlot: false,
  eventSources: [{events: eventArray}, {events: bookingArray, color: 'darkgreen'}],
  eventClick: function(info) {

    $('#eventModal').modal('show');
    document.getElementById('event_title').innerHTML = info.event.title;
    document.getElementById('event_time').innerHTML = `${moment(info.event._instance.range.start).utcOffset('+8:00').format('hh:mma')} -
    if (info.event.extendedProps.description == "") {
      document.getElementById('event_desc').innerHTML = "No description available."
    } else {
      document.getElementById('event_desc').innerHTML = info.event.extendedProps.description;
    }


    // if is personal event
    if(info.event.extendedProps.type == "personal_event") {
      document.getElementById('bookingWithUser').style.display = "none";
      document.getElementById('event_footer').style.display = "flex";
      // Edit event
      const editEvent = document.getElementById('editEvent');
      editEvent.onclick = function() {
        $('#eventModal').modal('hide');
        document.getElementById('eventTitle').innerHTML = info.event._def.title;
        document.getElementById('editEventTitle').value = info.event._def.title;
        document.getElementById('editEventStart').value = (info.event._instance.range.start).toISOString().slice(0,16);
        document.getElementById('editEventEnd').value = (info.event._instance.range.end).toISOString().slice(0,16);
        document.getElementById('editEventDescription').innerHTML = info.event.extendedProps.description;
```

Figure 3.4.5.1 Profile.js FullCalendar API



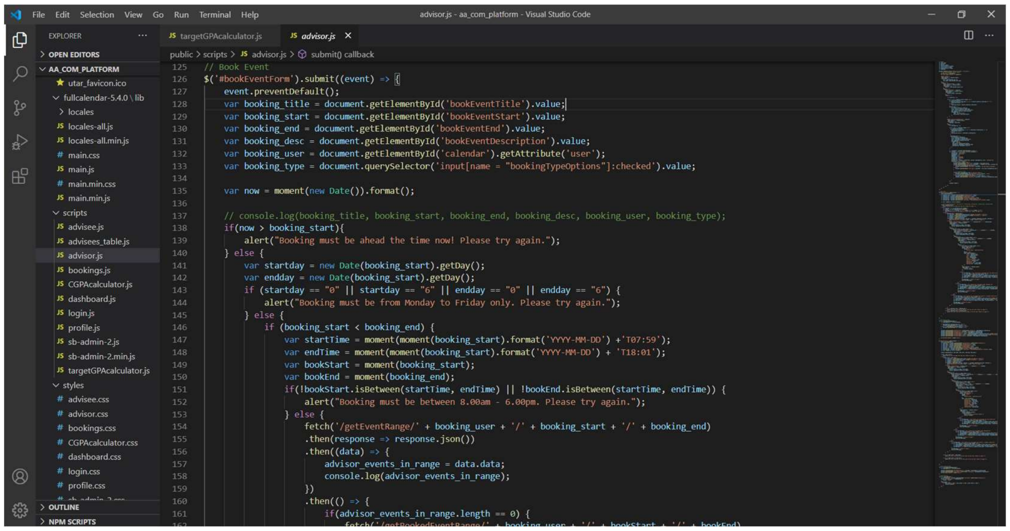
```
// } else {
  fetch('/getEventRange/' + user + '/' + event_time_start + '/' + event_time_end)
  .then(response => response.json())
  .then((data) => {
    // console.log(data.data);
    eventsInRange = data.data;
  })
  .then(() => {
    if (eventsInRange.length == 0) {
      fetch('/getAllAcceptedBookingsRange/' + user + '/' + event_time_start + '/' + event_time_end)
      .then(response => response.json())
      .then((data) => {
        console.log(data.data);
        acceptedBookingsInRange = data.data;
      })
      .then(() => {
        if (acceptedBookingsInRange.length == 0) {
          console.log('success!');
          fetch('/addEvent', {
            headers: {
                'Content-type': 'application/json'
            },
            method: 'POST',
            body: JSON.stringify({
                'event_title': event_title,
                'event_time_start': event_time_start,
                'event_time_end': event_time_end,
                'event_desc': event_desc,
                'event_user': event_user
            })
          })
          .then(response => response.json())
          .then((data) => {
            alert("Event successfully added!");
            location.reload();
          });
        } else {
```

Figure 3.4.5.2 Profile.js Add Event

3.4.6 Meeting Booking Development

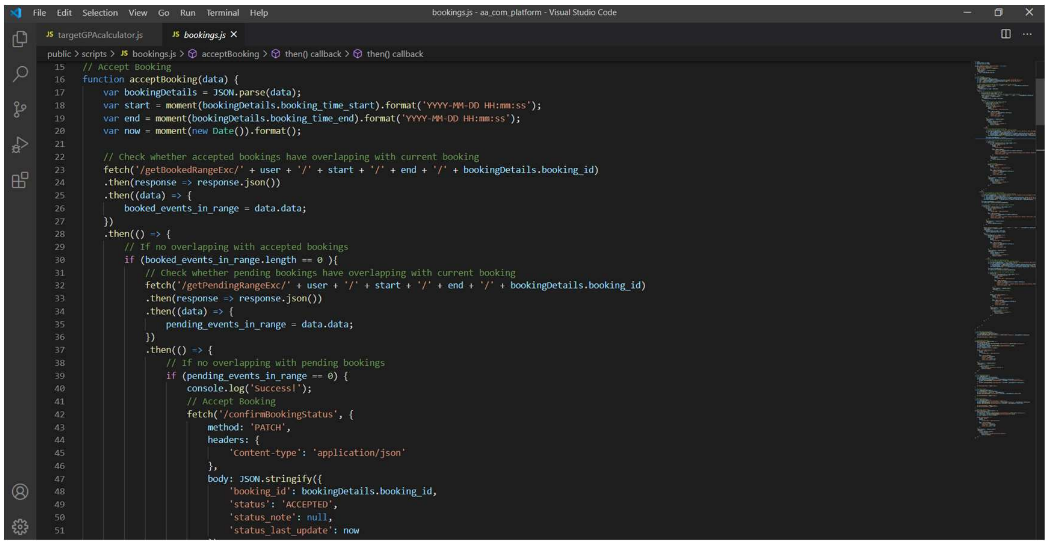
Advisee is able to book meetings with his advisor through the profile of the advisor, and also vice versa. After submitting the booking form, the system will first verify the time booked is ahead the current time as user (advisor or advisee) cannot book time which is over. Next, the time of the booking must be in working hours, which is Monday to Friday 8am to 6pm. The system will then check from the database whether there are any personal events, pending and accepted bookings of the user are overlapping the current booking. If not, the information of the bookings will be passed to the server-side through HTTP POST method, and the database will perform insert statement to the booking table, and the new booking will be reflected in the page. User can also perform edit and delete of the booking if the start time of booking has not reached. Edit of the event will also first perform overlap verification, and then pass all data to the back end through HTTP PATCH and MySQL will perform update to the booking using the booking id. Deleting of the event is done by passing the booking id to back end through HTTP DELETE, and MySQL will delete the booking using the booking id.

Successfully booked events will be having a pending status, and the user (advisee or advisor) booked will need to either accept or reject the booking. When the event is accepted, the system will check on the events and bookings of the accepting user. If there is no overlap, it will then check on all pending and accepted bookings. If there is overlapping of time of the bookings, the system will prompt the user to confirm that he or she wants to accept the current booking and reject the other booking that is overlapping the current booking. If the user confirms or there is no overlapping of time of bookings, the booking will update the status to accepted by using HTTP PATCH and passing the status to the backend and interact with MySQL and perform update statement to the booking with the booking id. If the user chooses to reject the booking, the reason of rejecting the booking is required to be filled and the status will be updated to rejected. On the other hand, if there is already a pending event booked by the user with his advisee or advisor, the event will prompt user to confirm that the booking needs to be rejected as they have already done a booking with his advisor or advisee.


advisor.js - aa_com_platform - Visual Studio Code
targetGPAcalculator.js
advisor.js
public > scripts > advisor.js > submit() callback

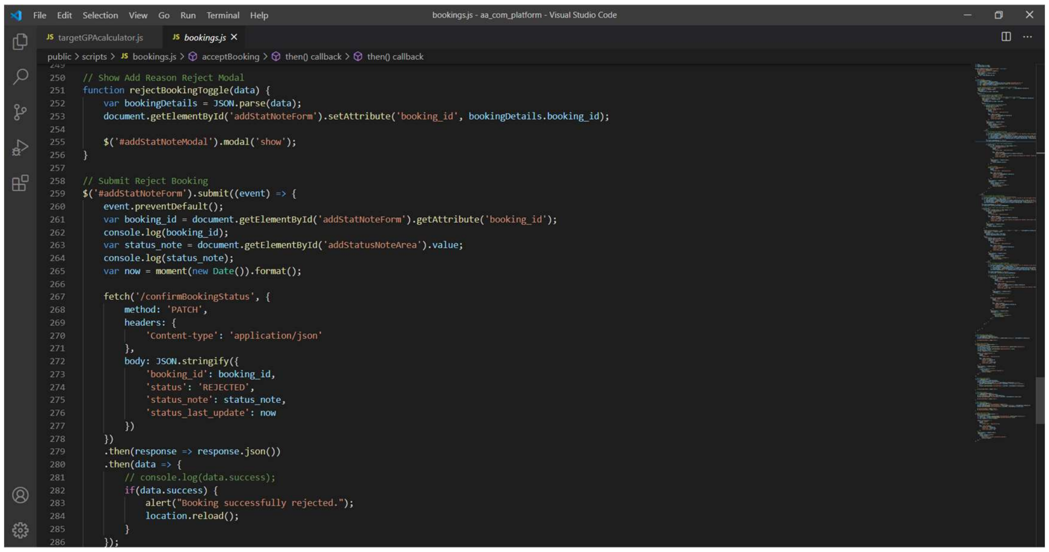
```
// Book Event
$('#bookEventForm').submit((event) => {
    event.preventDefault();
    var booking_title = document.getElementById('bookEventTitle').value;
    var booking_start = document.getElementById('bookEventStart').value;
    var booking_end = document.getElementById('bookEventEnd').value;
    var booking_desc = document.getElementById('bookEventDescription').value;
    var booking_user = document.getElementById('calendar').getAttribute('user');
    var booking_type = document.querySelector('input[name = "bookingTypeOptions"]:checked').value;

    var now = moment(new Date()).format();

    // console.log(booking_title, booking_start, booking_end, booking_desc, booking_user, booking_type);
    if(now > booking_start){
        alert("Booking must be ahead the time now! Please try again.");
    } else {
        var startday = new Date(booking_start).getDay();
        var endday = new Date(booking_start).getDay();
        if (startday == "0" || startday == "6" || endday == "0" || endday == "6") {
            alert("Booking must be from Monday to Friday only. Please try again.");
        } else {
            if (booking_start < booking_end) {
                var startTime = moment(moment(booking_start).format('YYYY-MM-DD') +'T07:59');
                var endTime = moment(moment(booking_start).format('YYYY-MM-DD') + 'T18:01');
                var bookStart = moment(booking_start);
                var bookEnd = moment(booking_end);
                if(!bookStart.isBetween(startTime, endTime) || !bookEnd.isBetween(startTime, endTime)) {
                    alert("Booking must be between 8.00am - 6.00pm. Please try again.");
                } else {
                    fetch('/getEventRange/' + booking_user + '/' + booking_start + '/' + booking_end)
                    .then(response => response.json())
                    .then((data) => {
                        advisor_events_in_range = data.data;
                        console.log(advisor_events_in_range);
                    })
                    .then(() => {
                        if(advisor_events_in_range.length == 0) {
                            fetch('/getBookedEventRange/' + booking_user + '/' + bookStart + '/' + bookEnd)
```

Figure 3.4.6.1 Advisor.js Function After Submitting Booking Form



```
// Accept Booking
function acceptBooking(data) {
    var bookingDetails = JSON.parse(data);
    var start = moment(bookingDetails.booking_time_start).format('YYYY-MM-DD HH:mm:ss');
    var end = moment(bookingDetails.booking_time_end).format('YYYY-MM-DD HH:mm:ss');
    var now = moment(new Date()).format();

    // Check whether accepted bookings have overlapping with current booking
    fetch('/getBookedRangeExc/' + user + '/' + start + '/' + end + '/' + bookingDetails.booking_id)
    .then(response => response.json())
    .then((data) => {
        booked_events_in_range = data.data;
    })
    .then(() => {
        // If no overlapping with accepted bookings
        if (booked_events_in_range.length == 0 ){
            // Check whether pending bookings have overlapping with current booking
            fetch('/getPendingRangeExc/' + user + '/' + start + '/' + end + '/' + bookingDetails.booking_id)
            .then(response => response.json())
            .then((data) => {
                pending_events_in_range = data.data;
            })
            .then(() => {
                // If no overlapping with pending bookings
                if (pending_events_in_range == 0) {
                    console.log('Success!');
                    // Accept Booking
                    fetch('/confirmBookingStatus', {
                        method: 'PATCH',
                        headers: {
                            'Content-type': 'application/json'
                        },
                        body: JSON.stringify({
                            'booking_id': bookingDetails.booking_id,
                            'status': 'ACCEPTED',
                            'status_note': null,
                            'status_last_update': now
```

Figure 3.4.6.2 Bookings.js Accept Booking function



```
// Show Add Reason Reject Modal
function rejectBookingToggle(data) {
    var bookingDetails = JSON.parse(data);
    document.getElementById('addStatNoteForm').setAttribute('booking_id', bookingDetails.booking_id);

    $('#addStatNoteModal').modal('show');
}

// Submit Reject Booking
$('#addStatNoteForm').submit((event) => {
    event.preventDefault();
    var booking_id = document.getElementById('addStatNoteForm').getAttribute('booking_id');
    console.log(booking_id);
    var status_note = document.getElementById('addStatusNoteArea').value;
    console.log(status_note);
    var now = moment(new Date()).format();

    fetch('/confirmBookingStatus', {
        method: 'PATCH',
        headers: {
            'Content-type': 'application/json'
        },
        body: JSON.stringify({
            'booking_id': booking_id,
            'status': 'REJECTED',
            'status_note': status_note,
            'status_last_update': now
        })
    })
    .then(response => response.json())
    .then(data => {
        // console.log(data.success);
        if(data.success) {
            alert("Booking successfully rejected.");
            location.reload();
        }
    });
```

Figure 3.4.6.3 Booking.js Reject Booking Function

3.4.7 Announcement Development

In add announcement page and edit announcement page, there is a column for rich text description, and another one for uploading photo or file. For the rich text editor, summernote, which is a WYSIWYG bootstrap editor is chosen, and it need to be initialised when the website is loaded by using $(‘#summernote’).summernote(). The height of the editor and the functions inside the toolbar can also be edited during the initialisation. Besides, for the photo and file uploading, Firebase Storage is used, thus in the start of the code, Firebase is initialised and reference to firebase storage is made.

When administrator submit the add announcement function, the title, details and file value will be grabbed using their HTML id. The system will first check that the value of title is not empty in HTML, and value in summernote is not empty in Javascript, and then proceed to add a new announcement in the announcement table in MySQL. Next, the system checks whether there is file or image uploaded. If there is no attachment uploaded, the process of creating announcement is completed. On the other hand, if there is attachment uploaded, it will first store to Firebase storage by referring to the storage child and then put the file uploaded to the storage. After the attachment is successfully uploaded to Firebase storage, the download URL of the attachment will be returned, and HTTP POST will be done to connect with MySQL through Node.js and insert a new value, including its file name, file type, file URL and announcement ID into the announcement_media table. The file type is important as at the announcement page, if the attachment is an image, it will be shown directly in the announcement, and if it is a document, it will be shown as a hyperlink in the announcement.

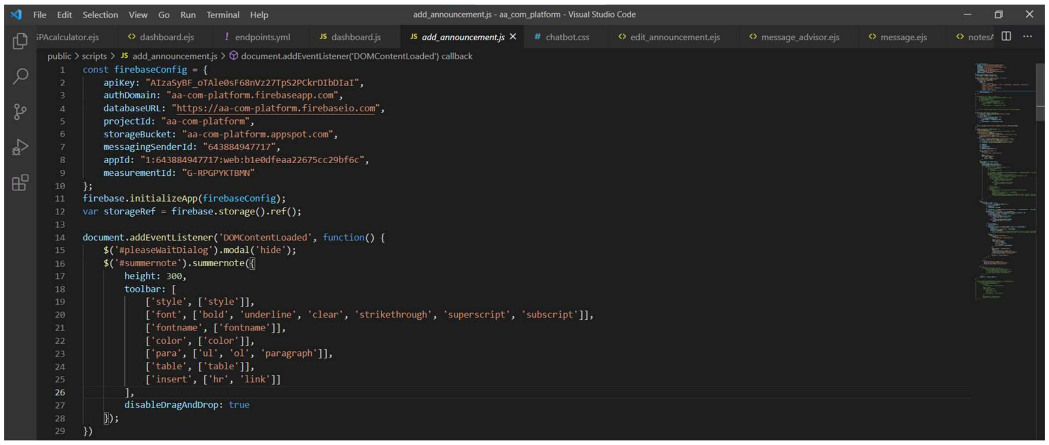

```
const firebaseConfig = {
    apiKey: "AIzaSyBF_oTAle0sF68nVz27TpS2PCkrDIbDIaI",
    authDomain: "aa-com-platform.firebaseapp.com",
    databaseURL: "https://aa-com-platform.firebaseio.com",
    projectId: "aa-com-platform",
    storageBucket: "aa-com-platform.appspot.com",
    messagingSenderId: "643884947717",
    appId: "1:643884947717:web:b1e0dfeaa22675cc29bf6c",
    measurementId: "G-RPGPYKTBMN"
};
firebase.initializeApp(firebaseConfig);
var storageRef = firebase.storage().ref();

document.addEventListener('DOMContentLoaded', function() {
    $('#pleaseWaitDialog').modal('hide');
    $('#summernote').summernote({
        height: 300,
        toolbar: [
            ['style', ['style']],
            ['font', ['bold', 'underline', 'clear', 'strikethrough', 'superscript', 'subscript']],
            ['fontname', ['fontname']],
            ['color', ['color']],
            ['para', ['ul', 'ol', 'paragraph']],
            ['table', ['table']],
            ['insert', ['hr', 'link']]
        ],
        disableDragAndDrop: true
    });
})
```

Figure 3.4.7.1 Add_announcement.js Initialization of Firebase Storage

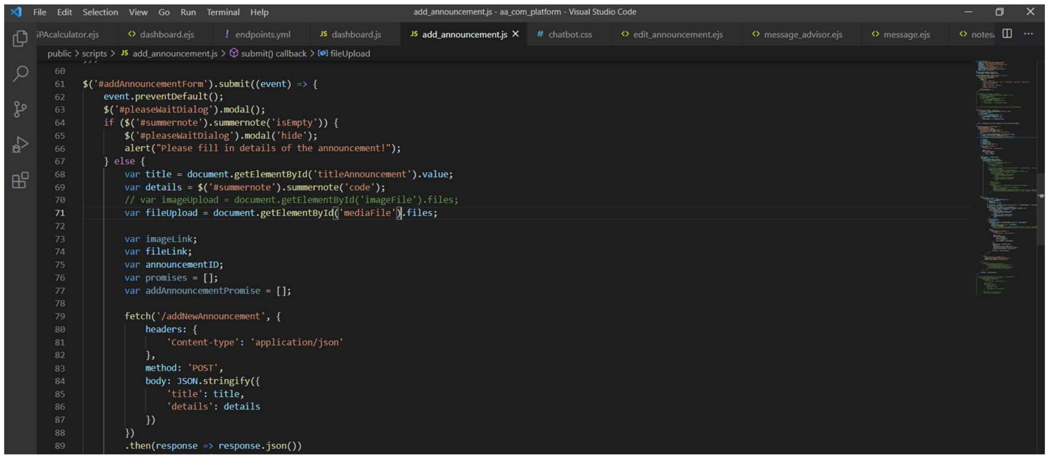
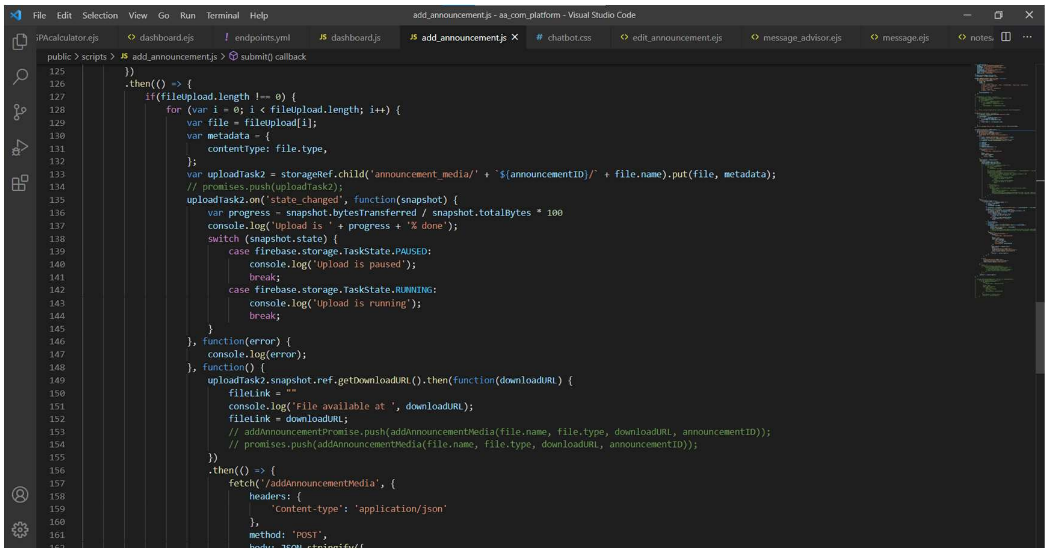
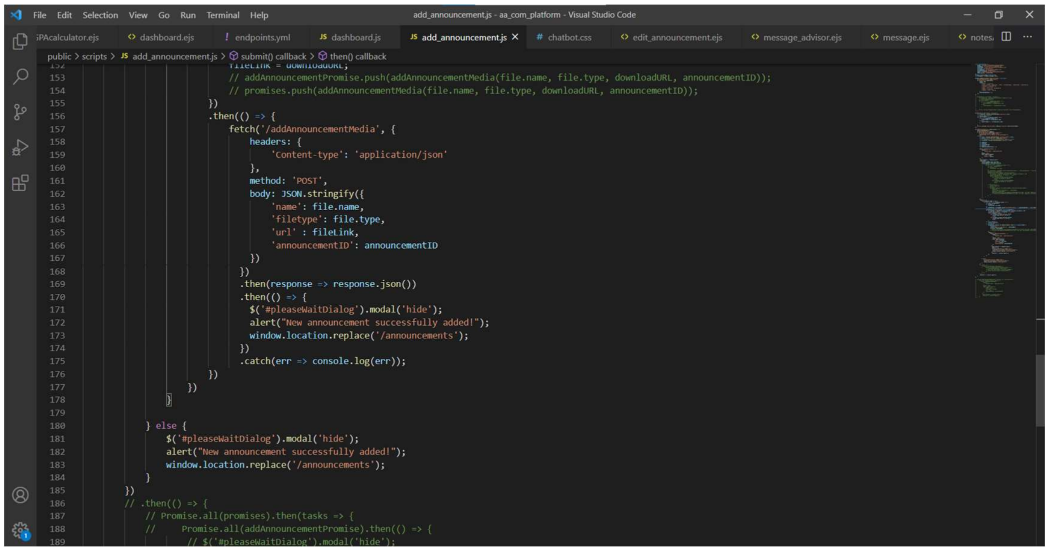

```
$('#addAnnouncementForm').submit((event) => {
    event.preventDefault();
    $('#pleaseWaitDialog').modal();
    if ($('#summernote').summernote('isEmpty')) {
        $('#pleaseWaitDialog').modal('hide');
        alert("Please fill in details of the announcement!");
    } else {
        var title = document.getElementById('titleAnnouncement').value;
        var details = $('#summernote').summernote('code');
        // var imageUpload = document.getElementById('imageFile').files;
        var fileUpload = document.getElementById('mediaFile').files;

        var imageLink;
        var fileLink;
        var announcementID;
        var promises = [];
        var addAnnouncementPromise = [];

        fetch('/addNewAnnouncement', {
            headers: {
                'Content-type': 'application/json'
            },
            method: 'POST',
            body: JSON.stringify({
                'title': title,
                'details': details
            })
        })
        .then(response => response.json())
```

Figure 3.4.7.2 Add_announcement.js Submit Add Announcement Form


File Edit Selection View Go Run Terminal Help
add_announcement.js - aa_com_platform - Visual Studio Code
iPAcalculator.ejs
dashboard.ejs
endpoints.yml
dashboard.js
add_announcement.js
chatbot.css
edit_announcement.ejs
message_advisor.ejs
message.ejs
notes
public > scripts > add_announcement.js > submit() callback

```
})
.then(() => {
    if(fileUpload.length !== 0) {
        for (var i = 0; i < fileUpload.length; i++) {
            var file = fileUpload[i];
            var metadata = {
                contentType: file.type,
            };
            var uploadTask2 = storageRef.child('announcement_media/' + `${announcementID}/` + file.name).put(file, metadata);
            // promises.push(uploadTask2);
            uploadTask2.on('state_changed', function(snapshot) {
                var progress = snapshot.bytesTransferred / snapshot.totalBytes * 100
                console.log('Upload is ' + progress + '% done');
                switch (snapshot.state) {
                    case firebase.storage.TaskState.PAUSED:
                        console.log('Upload is paused');
                        break;
                    case firebase.storage.TaskState.RUNNING:
                        console.log('Upload is running');
                        break;
                }
            }, function(error) {
                console.log(error);
            }, function() {
                uploadTask2.snapshot.ref.getDownloadURL().then(function(downloadURL) {
                    fileLink = ""
                    console.log('File available at ', downloadURL);
                    fileLink = downloadURL;
                    // addAnnouncementPromise.push(addAnnouncementMedia(file.name, file.type, downloadURL, announcementID));
                    // promises.push(addAnnouncementMedia(file.name, file.type, downloadURL, announcementID));
                })
                .then(() => {
                    fetch('/addAnnouncementMedia', {
                        headers: {
                            'Content-type': 'application/json'
                        },
                        method: 'POST',
                        body: JSON.stringify({
```

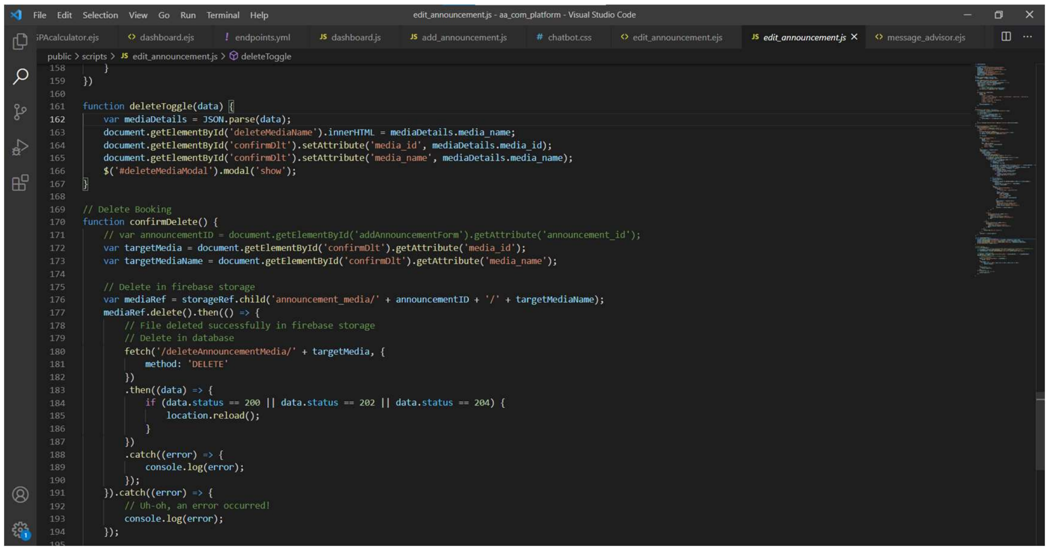
Figure 3.4.7.3 Add_announcement.js Store Attachment into Firebase Storage


File Edit Selection View Go Run Terminal Help
add_announcement.js - aa_com_platform - Visual Studio Code
iPAcalculator.ejs
dashboard.ejs
endpoints.yml
dashboard.js
add_announcement.js
chatbot.css
edit_announcement.ejs
message_advisor.ejs
message.ejs
notes
public > scripts > add_announcement.js > submit() callback > then() callback

```
                    fileLink = downloadURL;
                    // addAnnouncementPromise.push(addAnnouncementMedia(file.name, file.type, downloadURL, announcementID));
                    // promises.push(addAnnouncementMedia(file.name, file.type, downloadURL, announcementID));
                })
                .then(() => {
                    fetch('/addAnnouncementMedia', {
                        headers: {
                            'Content-type': 'application/json'
                        },
                        method: 'POST',
                        body: JSON.stringify({
                            'name': file.name,
                            'filetype': file.type,
                            'url' : fileLink,
                            'announcementID': announcementID
                        })
                      })
                      .then(response => response.json())
                      .then(() => {
                        $('#pleaseWaitDialog').modal('hide');
                        alert("New announcement successfully added!");
                        window.location.replace('/announcements');
                      })
                      .catch(err => console.log(err));
                })
            })
        }

    } else {
        $('#pleaseWaitDialog').modal('hide');
        alert("New announcement successfully added!");
        window.location.replace('/announcements');
    }
})
// .then(() => {
    // Promise.all(promises).then(tasks => {
    //     Promise.all(addAnnouncementPromise).then(() => {
            // $('#pleaseWaitDialog').modal('hide');
```

Figure 3.4.7.4 Add_announcement.js Store Attachment URL into MySQL Database

For the editing of the announcement, administrator can also edit all information of the announcement. Administrator can also delete the announcement attachment in the edit announcement page. Administrator is prompt to confirm delete, and after confirm, the attachment will first be deleted from the Firebase storage, and then HTTP DELETE to connect to MySQL through Node.js and delete the row. If the delete is success, an OK status will be replied and the current page will be reloaded, and administrator can upload a new attachment again. After submitting the edited information, the system will first check whether the title and details are filled, then if filled, it will update the MySQL announcement information by using HTTP PATCH. Then, it will check whether there is new attachment uploaded, or is the previous attachment. If it is an old attachment, the process is then completed; if not, it will check whether there is any new file uploaded. If yes, it will upload the new file, and after successfully uploaded, the download URL will be returned, and the announcement_media table will be updated.

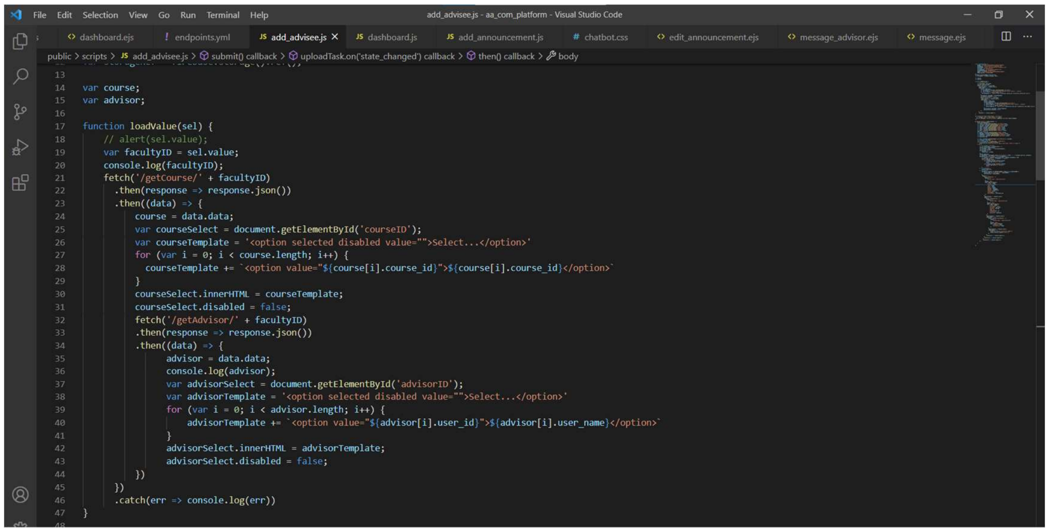

File Edit Selection View Go Run Terminal Help
edit_announcement.js - aa_com_platform - Visual Studio Code
iPAcalculator.ejs
dashboard.ejs
endpoints.yml
dashboard.js
add_announcement.js
chatbot.css
edit_announcement.ejs
edit_announcement.js
message_advisor.ejs
public > scripts > edit_announcement.js > deleteToggle

```
    }
})

function deleteToggle(data) {
    var mediaDetails = JSON.parse(data);
    document.getElementById('deleteMediaName').innerHTML = mediaDetails.media_name;
    document.getElementById('confirmDlt').setAttribute('media_id', mediaDetails.media_id);
    document.getElementById('confirmDlt').setAttribute('media_name', mediaDetails.media_name);
    $('#deleteMediaModal').modal('show');
}

// Delete Booking
function confirmDelete() {
    // var announcementID = document.getElementById('addAnnouncementForm').getAttribute('announcement_id');
    var targetMedia = document.getElementById('confirmDlt').getAttribute('media_id');
    var targetMediaName = document.getElementById('confirmDlt').getAttribute('media_name');

    // Delete in firebase storage
    var mediaRef = storageRef.child('announcement_media/' + announcementID + '/' + targetMediaName);
    mediaRef.delete().then(() => {
        // File deleted successfully in firebase storage
        // Delete in database
        fetch('/deleteAnnouncementMedia/' + targetMedia, {
            method: 'DELETE'
        })
        .then((data) => {
            if (data.status == 200 || data.status == 202 || data.status == 204) {
                location.reload();
            }
        })
        .catch((error) => {
            console.log(error);
        });
    }).catch((error) => {
        // Uh-oh, an error occurred!
        console.log(error);
    });
```

Figure 3.4.7.5 Edit_announcement.js Delete Previous Attachment Function

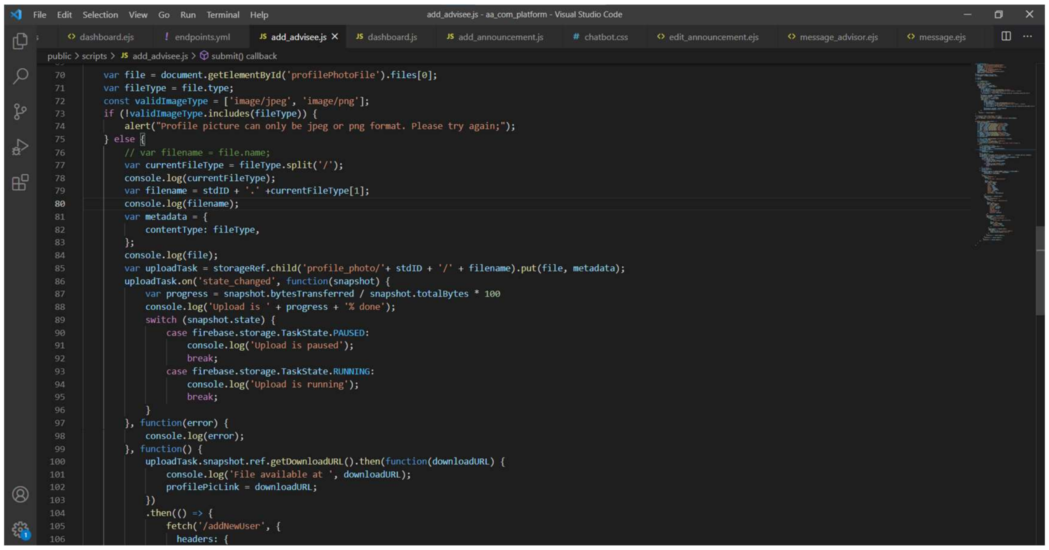
3.4.8 Manage User Development

To create a new user, there will be a form for administrator to key in all the information of the new user, and the form of creating an advisee and advisor is different, however the overall concept of developing it is the same. For creating an advisee, the faculty of advisee will first need to be inserted through the dropdown textbox. Then, there is a loadValue function done, and it will be executed whenever the selection of faculty is changed. The loadValue function will identify the faculty chose by administrator, and then it will HTTP FETCH all the courses and advisors in the faculty. If the administrator does not choose the faculty of students, the course and advisor column will be disabled. This is to ensure that administrator will not accidentally select a course or advisor which is not in the same faculty with the student. After all fields are filled in and submitted, all the values filled will be retrieved, and the profile picture uploaded will be checked whether it is a png or jpeg file. If the format of profile picture is not valid, an alert will be given. If valid, the profile photo will be uploaded to Firebase Storage according to their user ID. After uploading the photo, the download URL will be returned, and a HTTP POST is performed to insert new user's basic information into the Users table in MySQL, and another POST to insert user's information according to their particular user type to either advisee table or advisor table. After that, the email and ID of the new user will be used to create a new user in Firebase Authentication by using Firebase Admin.



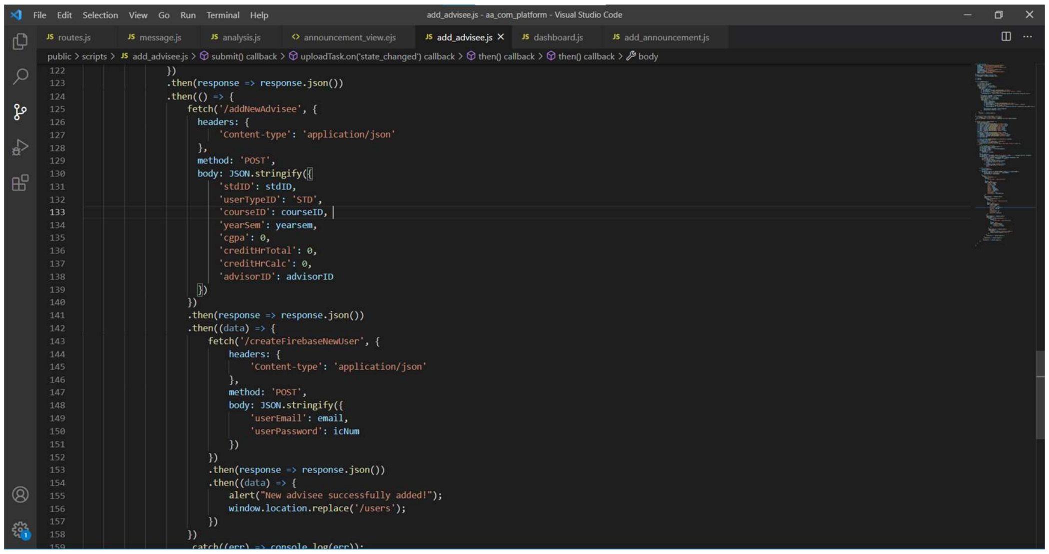
```
var course;
var advisor;

function loadValue(sel) {
    // alert(sel.value);
    var facultyID = sel.value;
    console.log(facultyID);
    fetch('/getCourse/' + facultyID)
      .then(response => response.json())
      .then((data) => {
        course = data.data;
        var courseSelect = document.getElementById('courseID');
        var courseTemplate = '<option selected disabled value="">Select...</option>'
        for (var i = 0; i < course.length; i++) {
          courseTemplate += `<option value="${course[i].course_id}">${course[i].course_id}</option>`
        }
        courseSelect.innerHTML = courseTemplate;
        courseSelect.disabled = false;
        fetch('/getAdvisor/' + facultyID)
        .then(response => response.json())
        .then((data) => {
            advisor = data.data;
            console.log(advisor);
            var advisorSelect = document.getElementById('advisorID');
            var advisorTemplate = '<option selected disabled value="">Select...</option>'
            for (var i = 0; i < advisor.length; i++) {
                advisorTemplate += `<option value="${advisor[i].user_id}">${advisor[i].user_name}</option>`
            }
            advisorSelect.innerHTML = advisorTemplate;
            advisorSelect.disabled = false;
        })
      })
      .catch(err => console.log(err))
}
```

Figure 3.4.8.1 Add_advisee.js Load Course and Advisor Value Function



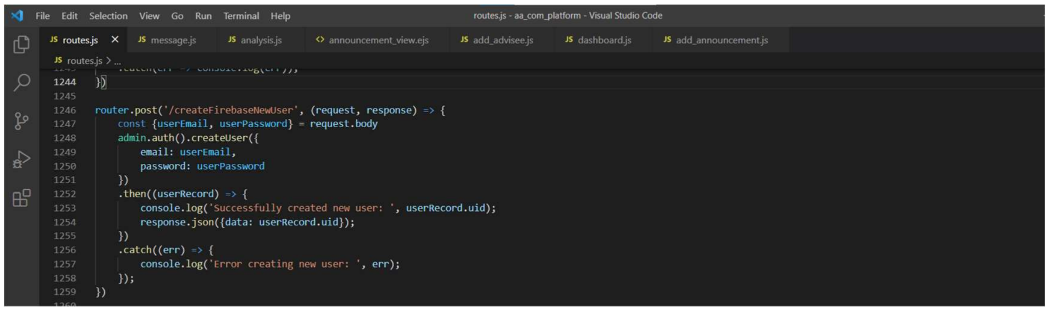
```
    var file = document.getElementById('profilePhotoFile').files[0];
    var fileType = file.type;
    const validImageType = ['image/jpeg', 'image/png'];
    if (!validImageType.includes(fileType)) {
        alert("Profile picture can only be jpeg or png format. Please try again;");
    } else {
        // var filename = file.name;
        var currentFileType = fileType.split('/');
        console.log(currentFileType);
        var filename = stdID + '.' +currentFileType[1];
        console.log(filename);
        var metadata = {
            contentType: fileType,
        };
        console.log(file);
        var uploadTask = storageRef.child('profile_photo/'+ stdID + '/' + filename).put(file, metadata);
        uploadTask.on('state_changed', function(snapshot) {
            var progress = snapshot.bytesTransferred / snapshot.totalBytes * 100
            console.log('Upload is ' + progress + '% done');
            switch (snapshot.state) {
                case firebase.storage.TaskState.PAUSED:
                    console.log('Upload is paused');
                    break;
                case firebase.storage.TaskState.RUNNING:
                    console.log('Upload is running');
                    break;
            }
        }, function(error) {
            console.log(error);
        }, function() {
            uploadTask.snapshot.ref.getDownloadURL().then(function(downloadURL) {
                console.log('File available at ', downloadURL);
                profilePicLink = downloadURL;
            })
            .then(() => {
                fetch('/addNewUser', {
                  headers: {
```

Figure 3.4.8.2 Add_advisee.js Add Advisee Function Part 1



```
                    })
                    .then(response => response.json())
                    .then(() => {
                        fetch('/addNewAdvisee', {
                            headers: {
                                'Content-type': 'application/json'
                            },
                            method: 'POST',
                            body: JSON.stringify({
                                'stdID': stdID,
                                'userTypeID': 'STD',
                                'courseID': courseID,
                                'yearSem': yearsem,
                                'cgpa': 0,
                                'creditHrTotal': 0,
                                'creditHrCalc': 0,
                                'advisorID': advisorID
                            })
                        })
                        .then(response => response.json())
                        .then((data) => {
                            fetch('/createFirebaseNewUser', {
                                headers: {
                                    'Content-type': 'application/json'
                                },
                                method: 'POST',
                                body: JSON.stringify({
                                    'userEmail': email,
                                    'userPassword': icNum
                                })
                            })
                            .then(response => response.json())
                            .then((data) => {
                                alert("New advisee successfully added!");
                                window.location.replace('/users');
                            })
                        })
                        .catch((err) => console.log(err));
```

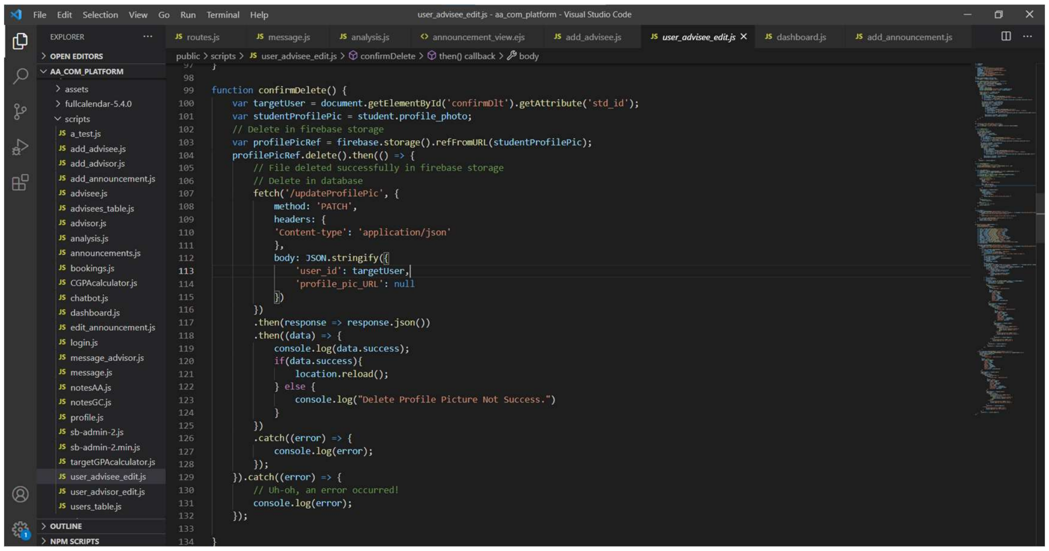
Figure 3.4.8.3 Add_advisee.js Add Advisee Function Part 2



```
})

router.post('/createFirebaseNewUser', (request, response) => {
    const {userEmail, userPassword} = request.body
    admin.auth().createUser({
        email: userEmail,
        password: userPassword
    })
    .then((userRecord) => {
        console.log('Successfully created new user: ', userRecord.uid);
        response.json({data: userRecord.uid});
    })
    .catch((err) => {
        console.log('Error creating new user: ', err);
    });
})
```

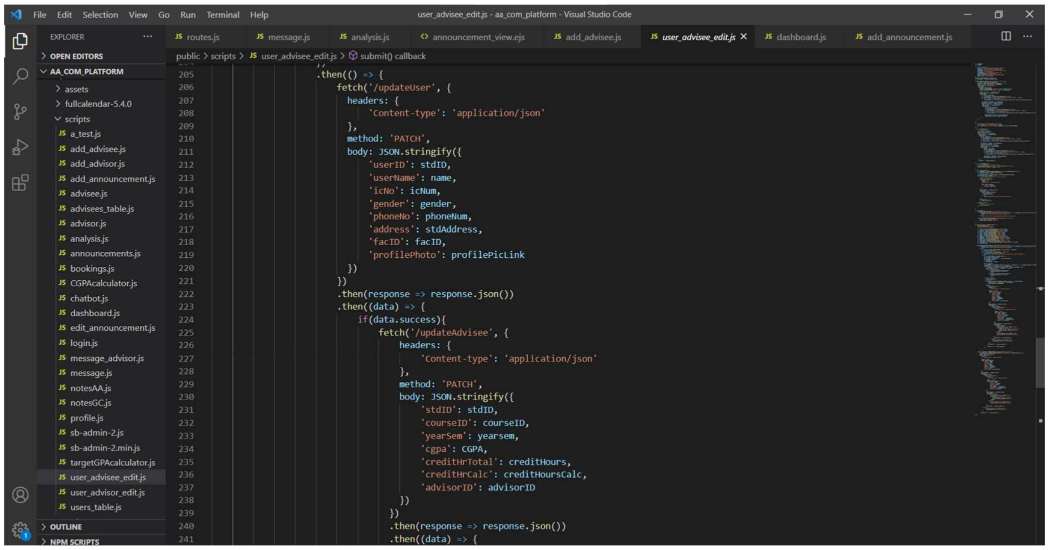
Figure 3.4.8.4 Node.js Create Firebase Auth User Function

Besides, administrator can also edit users. The process of editing users is almost similar as creating users, but the existing information is already filled in when entering the page, and the user ID and email could not be edited again. In order to edit the profile photo of the user, administrator need to delete the existing profile photo first. When administrator click on the delete button, the profile picture will first be deleted in Firebase Storage, and after successfully deleted from Firebase, the user table will update its profile picture URL to null. To save the edited profile successfully, all information must be filled. Thus, after user submit the user edit form, it will check that all the information and profile photo is uploaded. If the profile photo is not modified, a HTTP POST will directly be sent to update the user table and advisee or advisor table. If the profile photo is modified, it will first check that there is new profile photo attached, and upload to Firebase storage if there is any new profile photo attached. After that, the download URL is obtained after finish uploading, and all the new information filled will be updated to database.



```
}

function confirmDelete() {
    var targetUser = document.getElementById('confirmDlt').getAttribute('std_id');
    var studentProfilePic = student.profile_photo;
    // Delete in firebase storage
    var profilePicRef = firebase.storage().refFromURL(studentProfilePic);
    profilePicRef.delete().then(() => {
        // File deleted successfully in firebase storage
        // Delete in database
        fetch('/updateProfilePic', {
            method: 'PATCH',
            headers: {
            'Content-type': 'application/json'
            },
            body: JSON.stringify({
                'user_id': targetUser,
                'profile_pic_URL': null
            })
        })
        .then(response => response.json())
        .then((data) => {
            console.log(data.success);
            if(data.success){
                location.reload();
            } else {
                console.log("Delete Profile Picture Not Success.")
            }
        })
        .catch((error) => {
            console.log(error);
        });
    }).catch((error) => {
        // Uh-oh, an error occurred!
        console.log(error);
    });

}
```

Figure 3.4.8.5 User_advisee_edit.js Delete Profile Photo Function



```
                    .then(() => {
                        fetch('/updateUser', {
                          headers: {
                              'Content-type': 'application/json'
                          },
                          method: 'PATCH',
                          body: JSON.stringify({
                              'userID': stdID,
                              'userName': name,
                              'icNo': icNum,
                              'gender': gender,
                              'phoneNo': phoneNum,
                              'address': stdAddress,
                              'facID': facID,
                              'profilePhoto': profilePicLink
                          })
                        })
                        .then(response => response.json())
                        .then((data) => {
                            if(data.success){
                                fetch('/updateAdvisee', {
                                    headers: {
                                        'Content-type': 'application/json'
                                    },
                                    method: 'PATCH',
                                    body: JSON.stringify({
                                        'stdID': stdID,
                                        'courseID': courseID,
                                        'yearSem': yearsem,
                                        'cgpa': CGPA,
                                        'creditHrTotal': creditHours,
                                        'creditHrCalc': creditHoursCalc,
                                        'advisorID': advisorID
                                    })
                                })
                                .then(response => response.json())
                                .then((data) => {
```

Figure 3.4.8.6 User_advisee_edit.js Update User Function

3.4.9 Analysis Development

When advisees chat with the chatbot, all of the chat history will be stored into the database, and analysis is being carried out. The page will first render a table named Enquiry Fallback. It will state all the questions asked by advisees which the chatbot do not know how to answer. This function enables administrator to know the questions that the chatbot does not support or unable to identify due to lack of training and thus the administrator can improve the chatbot from time to time. At the same time, a pie chart which is separated into different labels according to their categories is also generated. The chart is done using Chart.js API, where the criteria of the chart is defined when generating the chart. The information asked by advisees in the current month is retrieved from database through HTTP GET, and then the count of each of the categories are used as the data of the chart. When one of the categories in the pie chart is clicked, the value of the category will be detected, and according to the category, the information of the particular category will be fetched from the database using HTTP POST. After that, a bar chart which states the detailed category of the questions and its count is generated by using Chart.js API. The colour of the bar chart is also set according to the colour of the category selected in the pie chart.

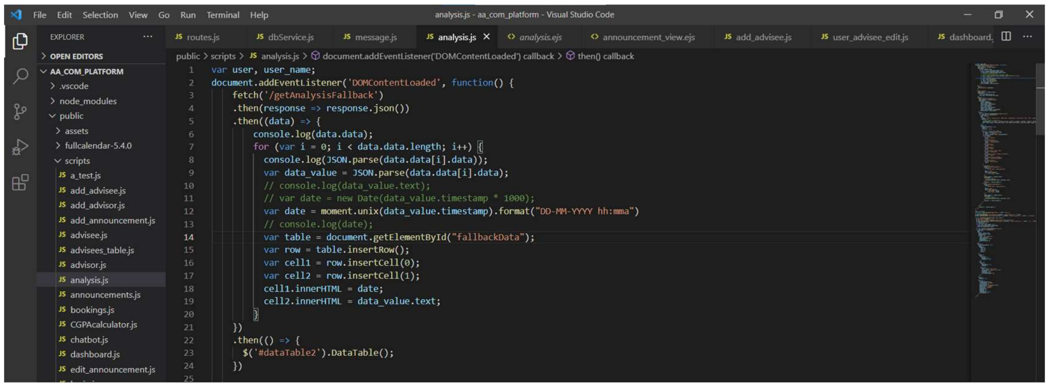

```
var user, user_name;
document.addEventListener('DOMContentLoaded', function() {
    fetch('/getAnalysisFallback')
    .then(response => response.json())
    .then((data) => {
        console.log(data.data);
        for (var i = 0; i < data.data.length; i++) {
          console.log(JSON.parse(data.data[i].data));
          var data_value = JSON.parse(data.data[i].data);
          // console.log(data_value.text);
          // var date = new Date(data_value.timestamp * 1000);
          var date = moment.unix(data_value.timestamp).format("DD-MM-YYYY hh:mma")
          // console.log(date);
          var table = document.getElementById("fallbackData");
          var row = table.insertRow();
          var cell1 = row.insertCell(0);
          var cell2 = row.insertCell(1);
          cell1.innerHTML = date;
          cell2.innerHTML = data_value.text;
        }
    })
    .then(() => {
      $('#dataTable2').DataTable();
    })
```

Figure 3.4.9.1 Analysis.js Generate Enquiry Fallback Table Function

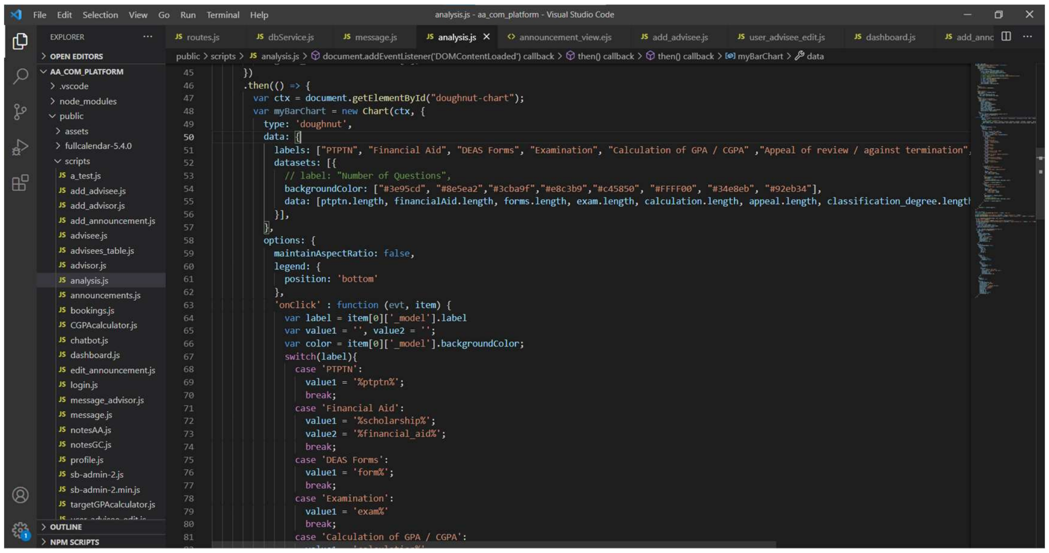
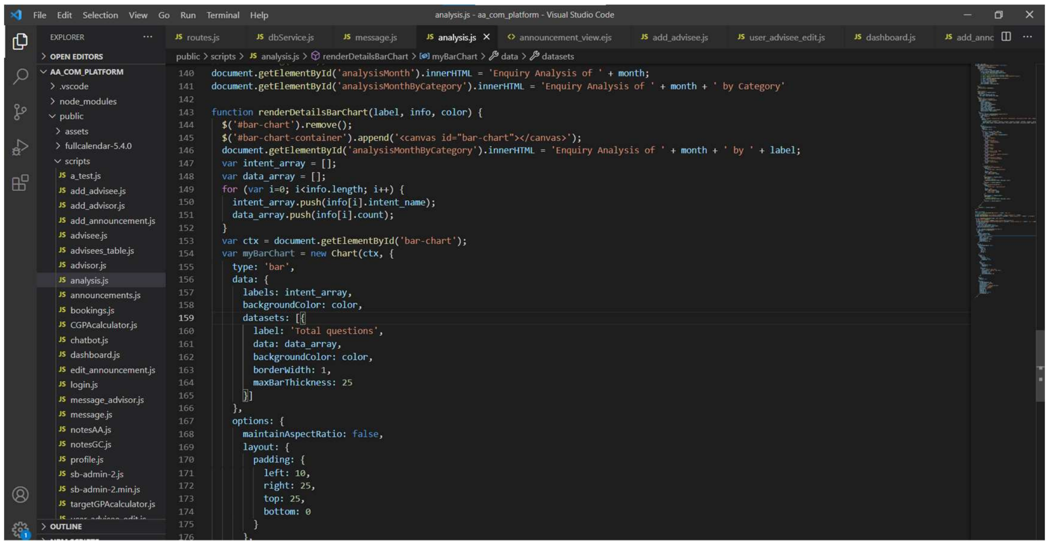

```
      })
      .then(() => {
        var ctx = document.getElementById("doughnut-chart");
        var myBarChart = new Chart(ctx, {
          type: 'doughnut',
          data: {
            labels: ["PTPTN", "Financial Aid", "DEAS Forms", "Examination", "Calculation of GPA / CGPA" ,"Appeal of review / against termination",
            datasets: [{
              // label: "Number of Questions",
              backgroundColor: ["#3e95cd", "#8e5ea2","#3cba9f","#e8c3b9","#c45850", "#FFFF00", "#34e8eb", "#92eb34"],
              data: [ptptn.length, financialAid.length, forms.length, exam.length, calculation.length, appeal.length, classification_degree.lengtl
            }],
          },
          options: {
            maintainAspectRatio: false,
            legend: {
              position: 'bottom'
            },
            'onClick' : function (evt, item) {
              var label = item[0]['_model'].label
              var value1 = '', value2 = '';
              var color = item[0]['_model'].backgroundColor;
              switch(label){
                case 'PTPTN':
                  value1 = '%ptptn%';
                  break;
                case 'Financial Aid':
                  value1 = '%scholarship%';
                  value2 = '%financial_aid%';
                  break;
                case 'DEAS Forms':
                  value1 = 'form%';
                  break;
                case 'Examination':
                  value1 = 'exam%'
                  break;
                case 'Calculation of GPA / CGPA':
```

Figure 3.4.9.2 Analysis.js Generate Pie Chart According to the Category of Questions Asked in the Current Month



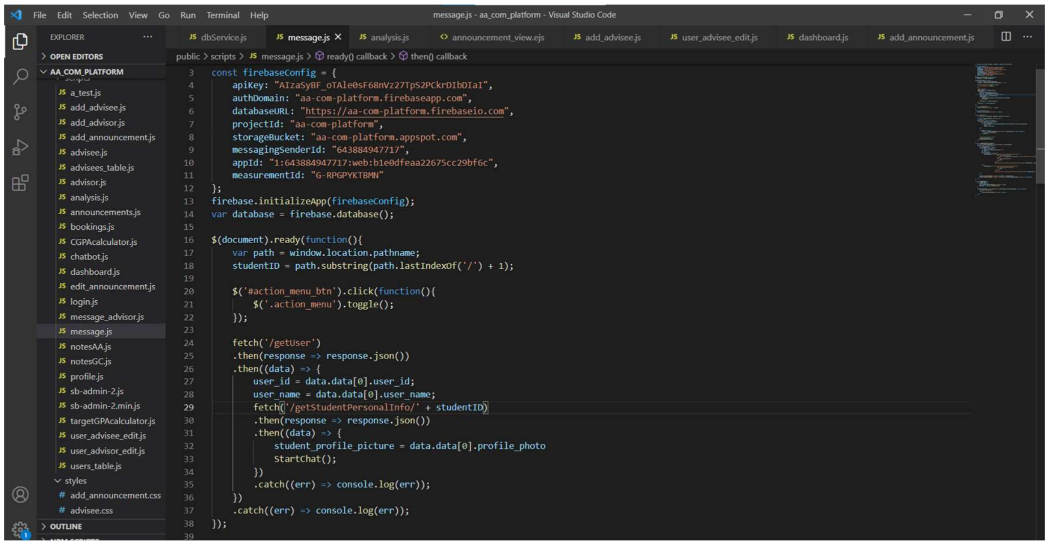
```
document.getElementById('analysisMonth').innerHTML = 'Enquiry Analysis of ' + month;
document.getElementById('analysisMonthByCategory').innerHTML = 'Enquiry Analysis of ' + month + ' by Category'

function renderDetailsBarChart(label, info, color) {
  $('#bar-chart').remove();
  $('#bar-chart-container').append('<canvas id="bar-chart"></canvas>');
  document.getElementById('analysisMonthByCategory').innerHTML = 'Enquiry Analysis of ' + month + ' by ' + label;
  var intent_array = [];
  var data_array = [];
  for (var i=0; i<info.length; i++) {
    intent_array.push(info[i].intent_name);
    data_array.push(info[i].count);
  }
  var ctx = document.getElementById('bar-chart');
  var myBarChart = new Chart(ctx, {
    type: 'bar',
    data: {
      labels: intent_array,
      backgroundColor: color,
      datasets: [{
        label: 'Total questions',
        data: data_array,
        backgroundColor: color,
        borderWidth: 1,
        maxBarThickness: 25
      }]
    },
    options: {
      maintainAspectRatio: false,
      layout: {
        padding: {
          left: 10,
          right: 25,
          top: 25,
          bottom: 0
        }
      },
```

Figure 3.4.9.3 Analysis.js Generate Bar Chart According to the Category Selected from the Pie Chart

3.4.10 Message Development

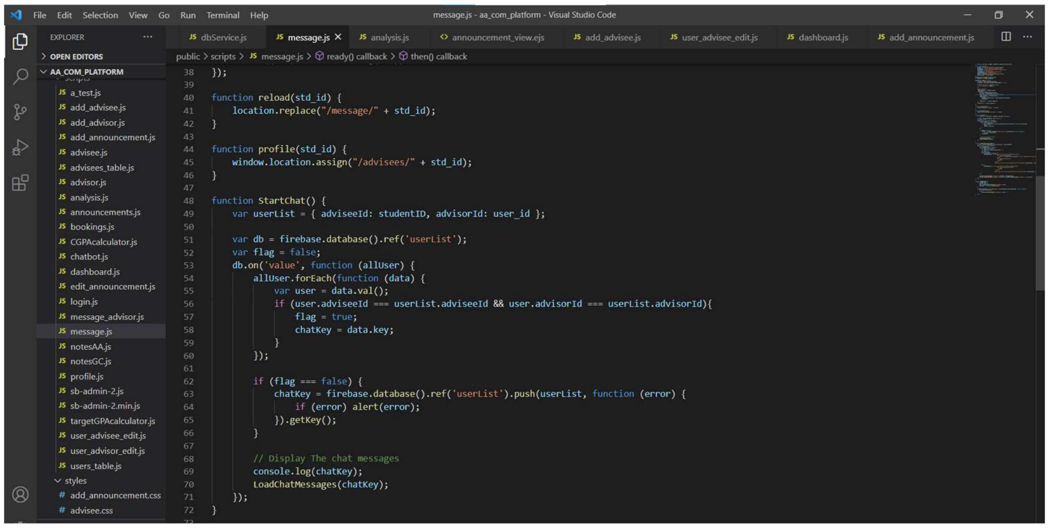
A message function is also developed in this project. Firstly, Firebase Realtime Database is first initialised, as all our chat will be stored there. When the document is ready, the data of the user selected will be retrieved, and the StartChat Function will be executed. The StartChat function is to ensure that there must a key generated for each of the connection between advisee and advisor. The key will be stored in the Firebase Database userList path. The system will first retrieve all of the value in userList path, and compare the current advisee id and advisor id with the value retrieved. If exists, the flag will be true and the key will be retrieved. If there is no existing key containing value of current advisor and advisee id, the current advisor and advisee id will be inserted into the userList path, and a key will be generated. The key obtained is then passed to the LoadChatMessages Function.

There is also a change user function named reload(), and it will be triggered when advisor changes the advisee selected. The page will be redirected to the message page according to the id of advisee selected. There is also a profile function, where user can view the profile of the opposite side by clicking on the View Profile button in the dropdown.



```
const firebaseConfig = {
    apiKey: "AIzaSyBF_oTAle0sF68nVz27TpS2PCkrDIbDIaI",
    authDomain: "aa-com-platform.firebaseapp.com",
    databaseURL: "https://aa-com-platform.firebaseio.com",
    projectId: "aa-com-platform",
    storageBucket: "aa-com-platform.appspot.com",
    messagingSenderId: "643884947717",
    appId: "1:643884947717:web:b1e0dfeaa22675cc29bf6c",
    measurementId: "G-RPGPYKTBMN"
};
firebase.initializeApp(firebaseConfig);
var database = firebase.database();

$(document).ready(function(){
    var path = window.location.pathname;
    studentID = path.substring(path.lastIndexOf('/') + 1);

    $('#action_menu_btn').click(function(){
        $('.action_menu').toggle();
    });

    fetch('/getUser')
    .then(response => response.json())
    .then((data) => {
        user_id = data.data[0].user_id;
        user_name = data.data[0].user_name;
        fetch('/getStudentPersonalInfo/' + studentID)
        .then(response => response.json())
        .then((data) => {
            student_profile_picture = data.data[0].profile_photo
            StartChat();
        })
        .catch((err) => console.log(err));
    })
    .catch((err) => console.log(err));
});
```

Figure 3.4.10.1 Message.js Initialize Firebase Database

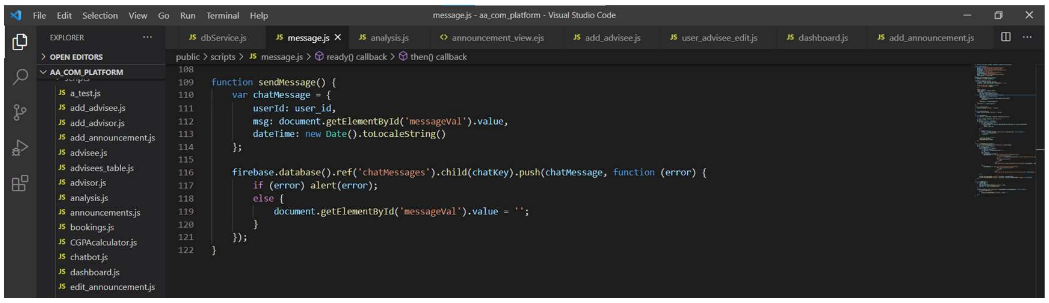


```
});

function reload(std_id) {
    location.replace("/message/" + std_id);
}

function profile(std_id) {
    window.location.assign("/advisees/" + std_id);
}

function StartChat() {
    var userList = { adviseeId: studentID, advisorId: user_id };

    var db = firebase.database().ref('userList');
    var flag = false;
    db.on('value', function (allUser) {
        allUser.forEach(function (data) {
            var user = data.val();
            if (user.adviseeId === userList.adviseeId && user.advisorId === userList.advisorId){
                flag = true;
                chatKey = data.key;
            }
        });

        if (flag === false) {
            chatKey = firebase.database().ref('userList').push(userList, function (error) {
                if (error) alert(error);
            }).getKey();
        }

        // Display The chat messages
        console.log(chatKey);
        LoadChatMessages(chatKey);
    });
}
```

Figure 3.4.10.2 Message.js Change User Function, View Profile Function and StartChart Function

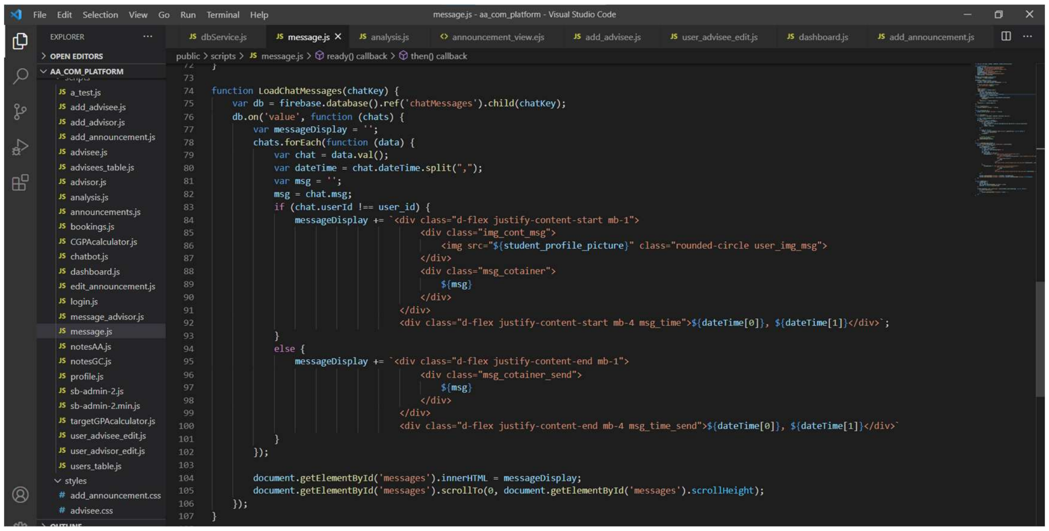
The sendMessage function is executed when the user submits a new message. The value of the message entered, together with the current user id, and the current date time is declared as a JSON value, and is stored in the chatMessages path followed by the key obtained from StartChart function as the child path.

The LoadChatMessages function is to load all previous stored chats between the two users. All of the chat messages are stored in the chatMessages path, and the child path is the key generated in StartChart function. Every previous value stored in it will be retrieved by loop. If the user id in the value obtained is different from the current user id, the message will be shown at the left side of the chat box with the profile picture of the opposite user. On the other hand, if the user id in the value obtained is same as the current user id, the message bubble will be shown at the right side of the chat box. Both of the messages will also show the message sent time at the bottom of the message bubble.



```javascript
function sendMessage() {
    var chatMessage = {
        userId: user_id,
        msg: document.getElementById('messageVal').value,
        dateTime: new Date().toLocaleString()
    };

    firebase.database().ref('chatMessages').child(chatKey).push(chatMessage, function (error) {
        if (error) alert(error);
        else {
            document.getElementById('messageVal').value = '';
        }
    });
}
```

Figure 3.4.10.3 Message.js SendMessage Function



```javascript
}

function LoadChatMessages(chatKey) {
    var db = firebase.database().ref('chatMessages').child(chatKey);
    db.on('value', function (chats) {
        var messageDisplay = '';
        chats.forEach(function (data) {
            var chat = data.val();
            var dateTime = chat.dateTime.split(",");
            var msg = '';
            msg = chat.msg;
            if (chat.userId !== user_id) {
                messageDisplay += `<div class="d-flex justify-content-start mb-1">
                                <div class="img_cont_msg">
                                    <img src="${student_profile_picture}" class="rounded-circle user_img_msg">
                                </div>
                                <div class="msg_cotainer">
                                    ${msg}
                                </div>
                            </div>
                            <div class="d-flex justify-content-start mb-4 msg_time">${dateTime[0]}, ${dateTime[1]}</div>`;
            }
            else {
                messageDisplay += `<div class="d-flex justify-content-end mb-1">
                                <div class="msg_cotainer_send">
                                    ${msg}
                                </div>
                            </div>
                            <div class="d-flex justify-content-end mb-4 msg_time_send">${dateTime[0]}, ${dateTime[1]}</div>`
            }
        });

        document.getElementById('messages').innerHTML = messageDisplay;
        document.getElementById('messages').scrollTo(0, document.getElementById('messages').scrollHeight);
    });
}
```

Figure 3.4.10.4 Message.js LoadChatMessage Function

3.5 Chatbot Development

3.5.1 Overview of Chatbot Development

The chatbot is built by using Rasa, which is a leading conversational AI platform, and personalize conversations can be built by using Rasa. It is an open-sourced framework, or user can also use Rasa Enterprise for more functionalities. Rasa Open Source can be used to build virtual assistant to automate human-computer interactions in websites or social media, such as Facebook Messenger, WhatsApp and much more, and it can also be connected to databases. It supports Natural Language Understanding (NLU), where it will interpret users' message into structure data, and according to the user message, it will decide what the virtual assistant does next by using dialogue management. Rasa Open Source uses YAML for managing the all data, including the intent, entities, stories and rules, and also the domain, which contains the responses. On the other hand, the Action Server is built using Python.

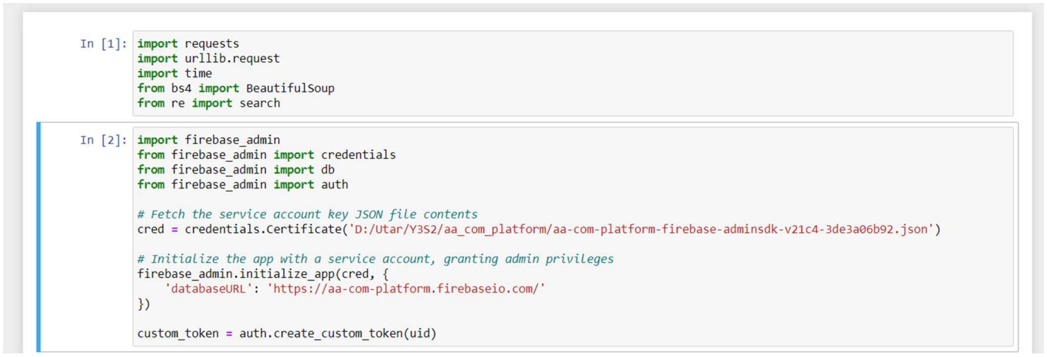
The chatbot need to be trained after filling in all the NLU data and stories. The training data is tokenized when trained by using WhitespaceTokenizer, where every word separated with whitespace will be given a token. Then, the data is assigned a vector value using the Bag-of-words (BoW) model, and data that has high similarities will have a shorter distance between them. The similarity is then measured using the cosine distance of the two vectors. After the training is done, a model file will be created with the trained timestamp as the default file name. The chatbot can now be used either in command line or Rasa X, or hosted in a server. When user input a message, the embeddings will be fit into a non-linear classification, and the message with high similarity with the trained data will have a high confidence score and be predicted.

3.5.2 Web Scraping

All the data used for chatbot replies are retrieved from the UTAR official website. The language used for web scraping is Python, and the main library used is Beautiful Soup. It is a library to pull data from HTML or XML files, and reduce the workload of retrieving data from websites one by one. Firstly, all the libraries used, including BeautifulSoup is initialized. Next, Firebase is initialised and the Firebase Database is connected through Firebase Admin.

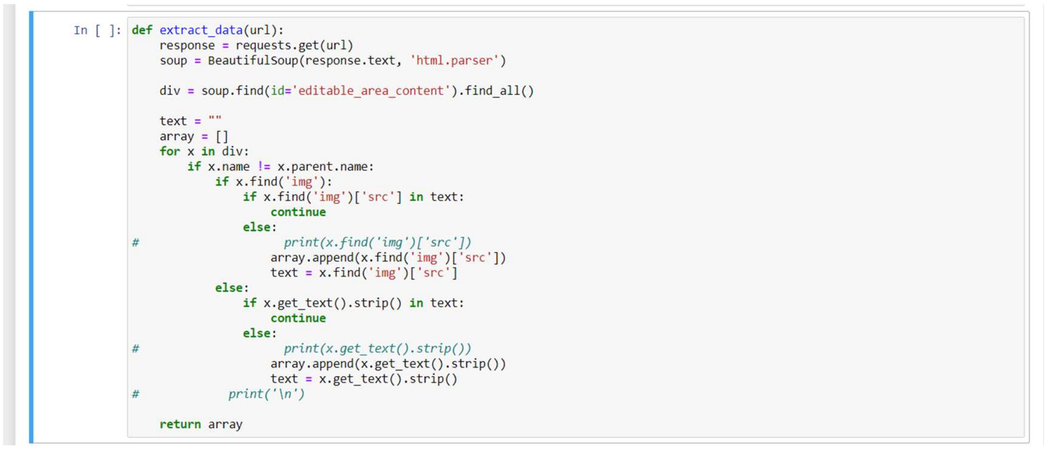
As in Figure 3.5.2.2, a function called extract_data() is defined. First, request.get is used to perform get from the URL, then BeautifulSoup is used to get the html value from the URL. Next, soup.find is used to find the html with id = “editable_area_content” and list all the html values nested in it with bs4.element.ResultSet class. Then, a loop is used to go through all elements in the class. If the element is an image, the image src URL will be stored in an array. If the element is text, then it will get the value of the text and store into the same array. There is also a variable declared as text, which the last value get will be stored in it, and when a new loop is executed, it will check whether the current data obtained is same as the previous data. This is to ensure that there is no duplicated data stored. After the loop is completed, the array will be returned, as shown in Figure 3.5.2.3.

In addition, as most of the page have different information, thus the data in the array returned from the function is restructured and rearranged, and stored into Firebase Database. For some URL that do not need to restructure, another function is made to extract those data from the URL and directly store the value into Firebase Database in the same function. The html value is first being retrieved using soup.find_all, and then the text in the html along with its HTML href URL link is stored into the array. The array is then being looped and stored into an object. The object is then being stored into Firebase Database by using ref.set().



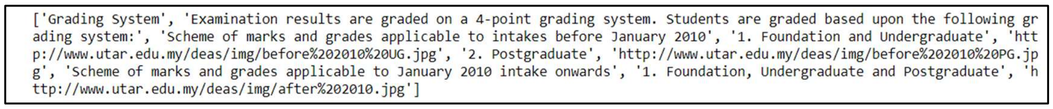
```
In [1]: import requests
        import urllib.request
        import time
        from bs4 import BeautifulSoup
        from re import search

In [2]: import firebase_admin
        from firebase_admin import credentials
        from firebase_admin import db
        from firebase_admin import auth

        # Fetch the service account key JSON file contents
        cred = credentials.Certificate('D:/Utar/Y3S2/aa_com_platform/aa-com-platform-firebase-adminsdk-v21c4-3de3a06b92.json')

        # Initialize the app with a service account, granting admin privileges
        firebase_admin.initialize_app(cred, {
            'databaseURL': 'https://aa-com-platform.firebaseio.com/'
        })

        custom_token = auth.create_custom_token(uid)
```

Figure 3.5.2.1 Initialize Library Used and Initialize Firebase Database Using Firebase Admin



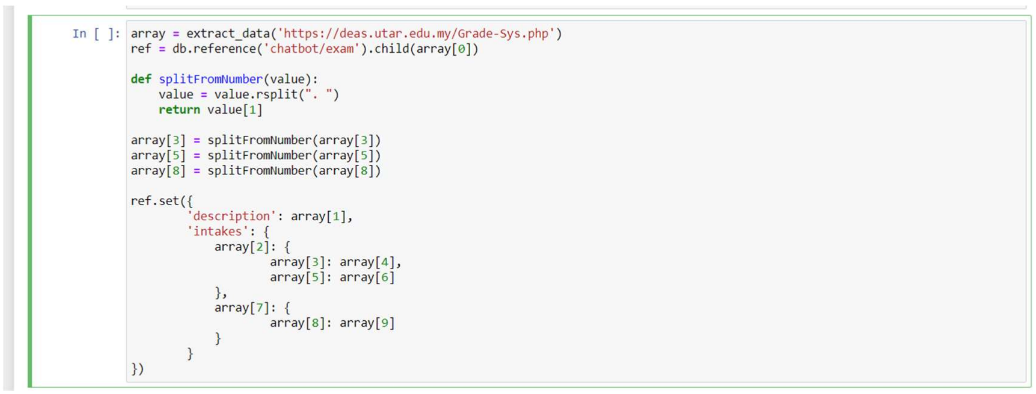
```
In [ ]: def extract_data(url):
            response = requests.get(url)
            soup = BeautifulSoup(response.text, 'html.parser')

            div = soup.find(id='editable_area_content').find_all()

            text = ""
            array = []
            for x in div:
                if x.name != x.parent.name:
                    if x.find('img'):
                        if x.find('img')['src'] in text:
                            continue
                        else:
#                             print(x.find('img')['src'])
                            array.append(x.find('img')['src'])
                            text = x.find('img')['src']
                    else:
                        if x.get_text().strip() in text:
                            continue
                        else:
#                             print(x.get_text().strip())
                            array.append(x.get_text().strip())
                            text = x.get_text().strip()
#                 print('\n')

            return array
```

Figure 3.5.2.2 Function for Scraping Data from Website URL



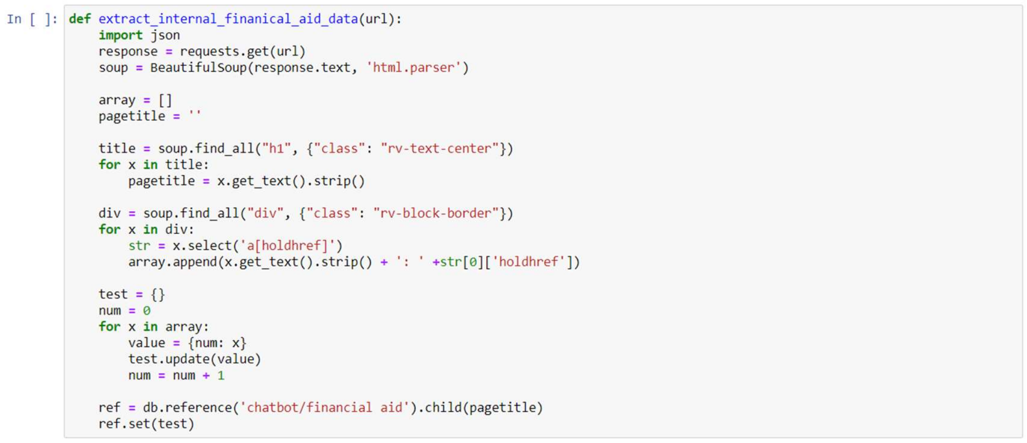
```
['Grading System', 'Examination results are graded on a 4-point grading system. Students are graded based upon the following gr
ading system:', 'Scheme of marks and grades applicable to intakes before January 2010', '1. Foundation and Undergraduate', 'htt
p://www.utar.edu.my/deas/img/before%202010%20UG.jpg', '2. Postgraduate', 'http://www.utar.edu.my/deas/img/before%202010%20PG.jp
g', 'Scheme of marks and grades applicable to January 2010 intake onwards', '1. Foundation, Undergraduate and Postgraduate', 'h
ttp://www.utar.edu.my/deas/img/after%202010.jpg']
```

Figure 3.5.2.3 Array Containing Data Retrieved from Web Scraping



```
In [ ]: array = extract_data('https://deas.utar.edu.my/Grade-Sys.php')
        ref = db.reference('chatbot/exam').child(array[0])

        def splitFromNumber(value):
            value = value.rsplit(". ")
            return value[1]

        array[3] = splitFromNumber(array[3])
        array[5] = splitFromNumber(array[5])
        array[8] = splitFromNumber(array[8])

        ref.set({
                'description': array[1],
                'intakes': {
                    array[2]: {
                            array[3]: array[4],
                            array[5]: array[6]
                    },
                    array[7]: {
                            array[8]: array[9]
                    }
                }
        })
```

Figure 3.5.2.4 Store Value into Firebase Realtime Database

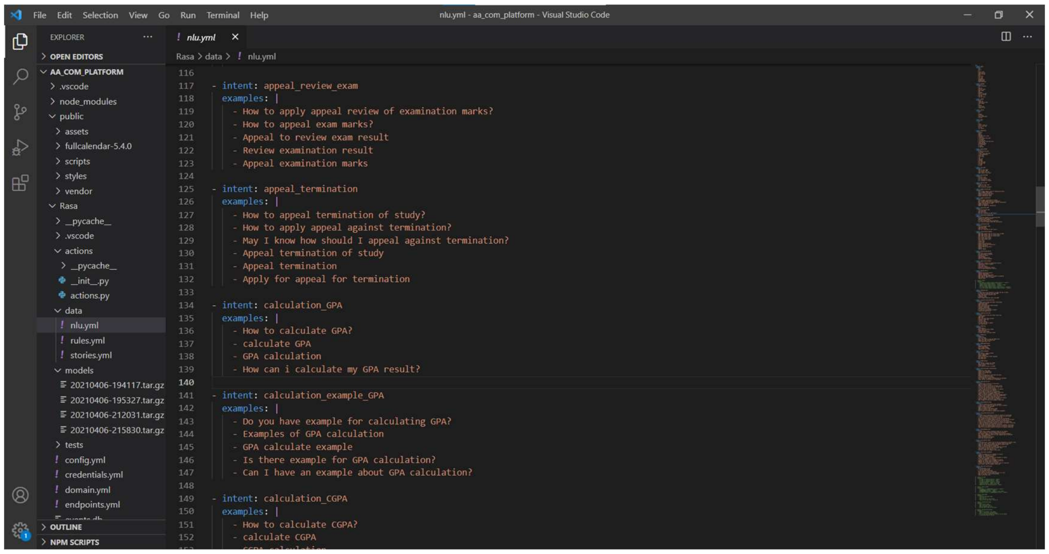


```
In [ ]: def extract_internal_finanical_aid_data(url):
            import json
            response = requests.get(url)
            soup = BeautifulSoup(response.text, 'html.parser')

            array = []
            pagetitle = ''

            title = soup.find_all("h1", {"class": "rv-text-center"})
            for x in title:
                pagetitle = x.get_text().strip()

            div = soup.find_all("div", {"class": "rv-block-border"})
            for x in div:
                str = x.select('a[holdhref]')
                array.append(x.get_text().strip() + ': ' +str[0]['holdhref'])

            test = {}
            num = 0
            for x in array:
                value = {num: x}
                test.update(value)
                num = num + 1

            ref = db.reference('chatbot/financial aid').child(pagetitle)
            ref.set(test)
```

Figure 3.5.2.5 Function for Scraping Data from Website URL and Store into Firebase Realtime Database

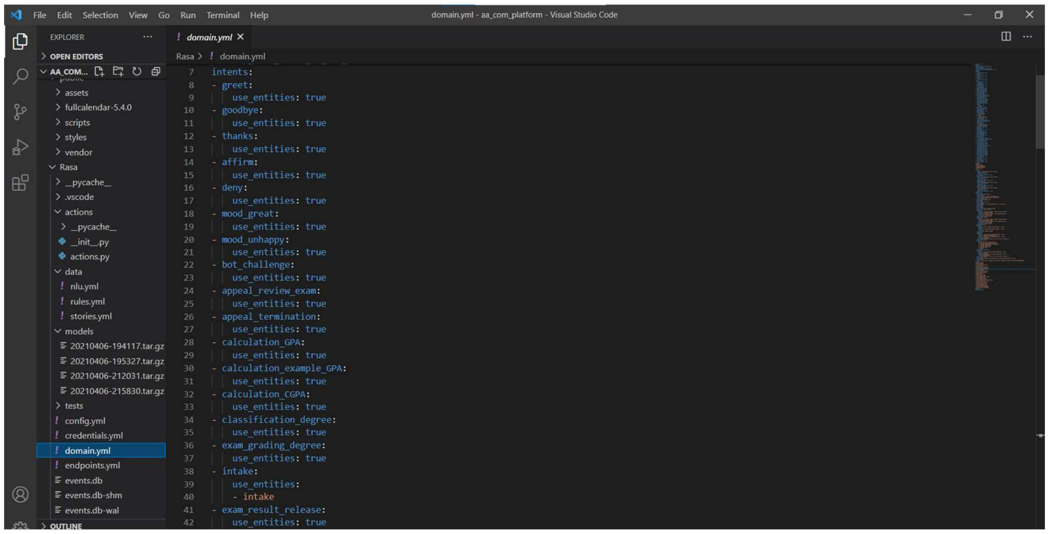
3.5.3 Chatbot Intent, Entities and Slots

Intent is the target of the user input in the chatbot, whereas entities are useful information from the user's input. Both of them are included in NLU and is used in data training. Different utterances of training data are categorised according to different intents. There will be at least 2 examples for each intent in nlu.yml. The intent must then be declared in domain.yml. Besides, some intent might consist of variables, which is called entities in Rasa. The entities need to be declared in nlu.yml, domain.yml under intent and also entities. As in Figure 3.5.3.1, all the intents and their examples are listed out. If the chatbot wants to store the reply of user for further usage, slot is used. In this project, slot is used to store the intake and education level of the user, as different intakes and education level will affect the reply of the chatbot, and the value is stored to be compared in actions.py for retrieving different data from the Firebase Database.



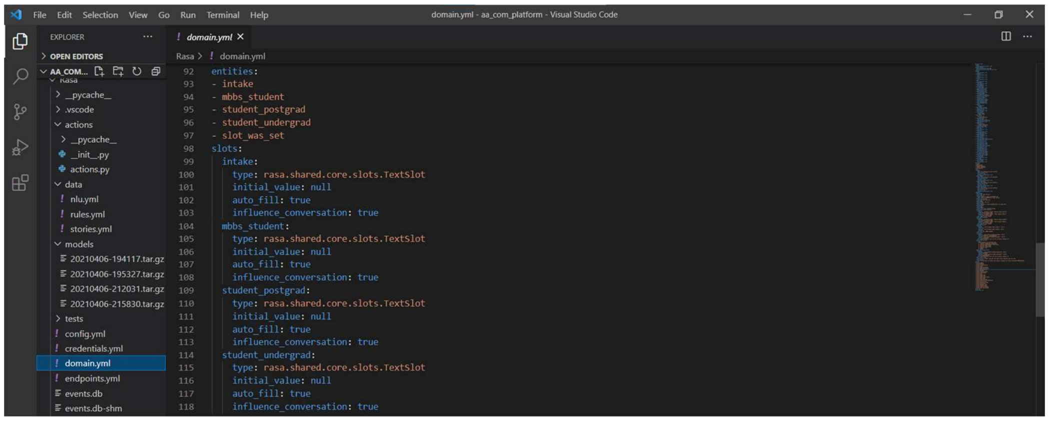
```
- intent: appeal_review_exam
  examples: |
    - How to apply appeal review of examination marks?
    - How to appeal exam marks?
    - Appeal to review exam result
    - Review examination result
    - Appeal examination marks

- intent: appeal_termination
  examples: |
    - How to appeal termination of study?
    - How to apply appeal against termination?
    - May I know how should I appeal against termination?
    - Appeal termination of study
    - Appeal termination
    - Apply for appeal for termination

- intent: calculation_GPA
  examples: |
    - How to calculate GPA?
    - calculate GPA
    - GPA calculation
    - How can i calculate my GPA result?

- intent: calculation_example_GPA
  examples: |
    - Do you have example for calculating GPA?
    - Examples of GPA calculation
    - GPA calculate example
    - Is there example for GPA calculation?
    - Can I have an example about GPA calculation?

- intent: calculation_CGPA
  examples: |
    - How to calculate CGPA?
    - calculate CGPA
```

Figure 3.5.3.1 nlu.yml Intent



```
intents:
- greet:
    use_entities: true
- goodbye:
    use_entities: true
- thanks:
    use_entities: true
- affirm:
    use_entities: true
- deny:
    use_entities: true
- mood_great:
    use_entities: true
- mood_unhappy:
    use_entities: true
- bot_challenge:
    use_entities: true
- appeal_review_exam:
    use_entities: true
- appeal_termination:
    use_entities: true
- calculation_GPA:
    use_entities: true
- calculation_example_GPA:
    use_entities: true
- calculation_CGPA:
    use_entities: true
- classification_degree:
    use_entities: true
- exam_grading_degree:
    use_entities: true
- intake:
    use_entities:
    - intake
- exam_result_release:
    use_entities: true
```

Figure 3.5.3.2 Domain.yml Intent



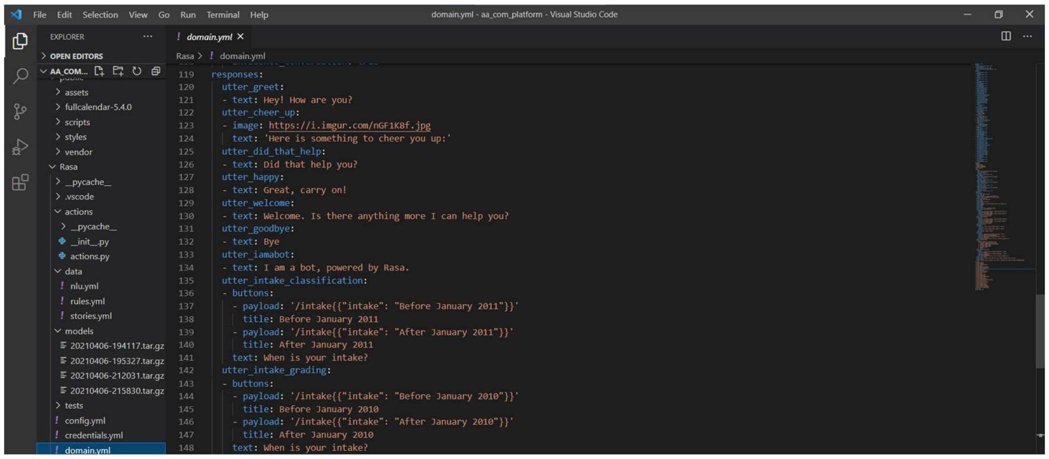
```
entities:
- intake
- mbbs_student
- student_postgrad
- student_undergrad
- slot_was_set
slots:
  intake:
    type: rasa.shared.core.slots.TextSlot
    initial_value: null
    auto_fill: true
    influence_conversation: true
  mbbs_student:
    type: rasa.shared.core.slots.TextSlot
    initial_value: null
    auto_fill: true
    influence_conversation: true
  student_postgrad:
    type: rasa.shared.core.slots.TextSlot
    initial_value: null
    auto_fill: true
    influence_conversation: true
  student_undergrad:
    type: rasa.shared.core.slots.TextSlot
    initial_value: null
    auto_fill: true
    influence_conversation: true
```

Figure 3.5.3.3 Domain.yml Entities and Slots

3.5.4 Chatbot Responses and Custom Actions

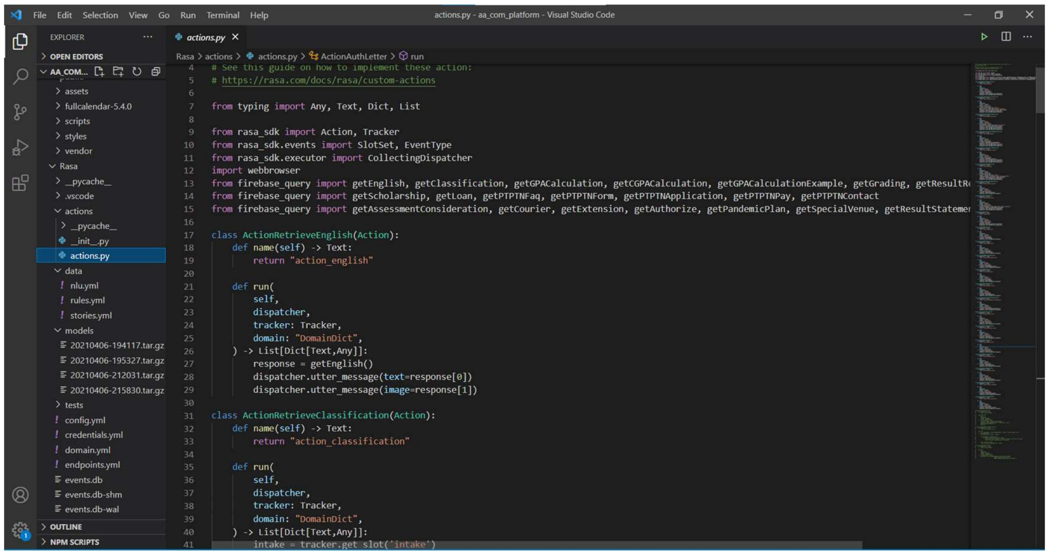
Responses is what the chatbot replies to the user. Each response will start with the name utter_. The responses can either be a text value, image, buttons or even custom output payloads. In this project, text is used for some basic answers and buttons when there are options for users to choose. For buttons, it consists of payload which includes the entity name and the value of entity, and also a title which will be shown in the button for users to choose. When the user clicks on one of the buttons, the entity of the selected button will then be received by the chatbot.

Besides, as the chatbot should be receiving all the data from Firebase Database but not typed in manually, thus custom actions are used. In this project, custom actions are mainly used to call the functions in firebase_query.py. In firebase_query.py, Firebase Admin is first initialised and connected to Firebase Realtime Database. Different functions are defined, and in the function, the data is queried from the path of the Firebase Realtime Database, and then returned as text and image. After the results from firebase.py function is returned to actions.py, by using dispatcher.utter_message and declaring the type of message, the message will be output by the chatbot. In order to use the custom actions, the name of custom actions must be written in the domain.yml.



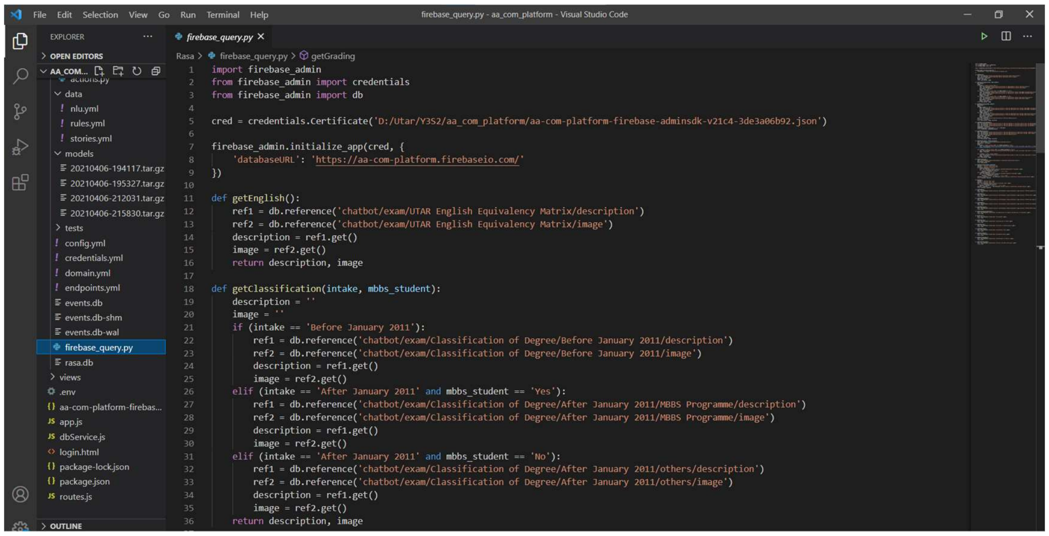
```
responses:
  utter_greet:
  - text: Hey! How are you?
  utter_cheer_up:
  - image: https://i.imgur.com/nGF1K8f.jpg
    text: 'Here is something to cheer you up:'
  utter_did_that_help:
  - text: Did that help you?
  utter_happy:
  - text: Great, carry on!
  utter_welcome:
  - text: Welcome. Is there anything more I can help you?
  utter_goodbye:
  - text: Bye
  utter_iamabot:
  - text: I am a bot, powered by Rasa.
  utter_intake_classification:
  - buttons:
    - payload: '/intake{{"intake": "Before January 2011"}}'
      title: Before January 2011
    - payload: '/intake{{"intake": "After January 2011"}}'
      title: After January 2011
    text: When is your intake?
  utter_intake_grading:
  - buttons:
    - payload: '/intake{{"intake": "Before January 2010"}}'
      title: Before January 2010
    - payload: '/intake{{"intake": "After January 2010"}}'
      title: After January 2010
    text: When is your intake?
```

Figure 3.5.4.1 Domain.yml Responses



```python
# See this guide on how to implement these action:
# https://rasa.com/docs/rasa/custom-actions

from typing import Any, Text, Dict, List

from rasa_sdk import Action, Tracker
from rasa_sdk.events import SlotSet, EventType
from rasa_sdk.executor import CollectingDispatcher
import webbrowser
from firebase_query import getEnglish, getClassification, getGPACalculation, getCGPACalculation, getGPACalculationExample, getGrading, getResultRe
from firebase_query import getScholarship, getLoan, getPTPTNFaq, getPTPTNForm, getPTPTNApplication, getPTPTNPay, getPTPTNContact
from firebase_query import getAssessmentConsideration, getCourier, getExtension, getAuthorize, getPandemicPlan, getSpecialVenue, getResultStatemen

class ActionRetrieveEnglish(Action):
    def name(self) -> Text:
        return "action_english"

    def run(
        self,
        dispatcher,
        tracker: Tracker,
        domain: "DomainDict",
    ) -> List[Dict[Text,Any]]:
        response = getEnglish()
        dispatcher.utter_message(text=response[0])
        dispatcher.utter_message(image=response[1])

class ActionRetrieveClassification(Action):
    def name(self) -> Text:
        return "action_classification"

    def run(
        self,
        dispatcher,
        tracker: Tracker,
        domain: "DomainDict",
    ) -> List[Dict[Text,Any]]:
        intake = tracker.get_slot('intake')
```

Figure 3.5.4.2 Actions.py Custom Actions

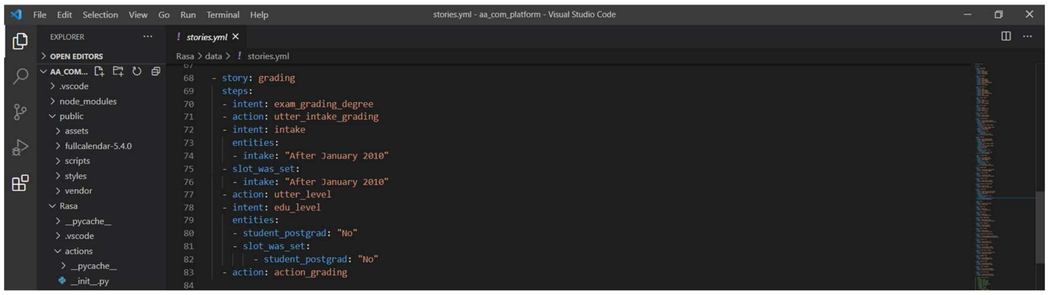


```python
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db

cred = credentials.Certificate('D:/Utar/Y3S2/aa_com_platform/aa-com-platform-firebase-adminsdk-v21c4-3de3a06b92.json')

firebase_admin.initialize_app(cred, {
    'databaseURL': 'https://aa-com-platform.firebaseio.com/'
})

def getEnglish():
    ref1 = db.reference('chatbot/exam/UTAR English Equivalency Matrix/description')
    ref2 = db.reference('chatbot/exam/UTAR English Equivalency Matrix/image')
    description = ref1.get()
    image = ref2.get()
    return description, image

def getClassification(intake, mbbs_student):
    description = ''
    image = ''
    if (intake == 'Before January 2011'):
        ref1 = db.reference('chatbot/exam/Classification of Degree/Before January 2011/description')
        ref2 = db.reference('chatbot/exam/Classification of Degree/Before January 2011/image')
        description = ref1.get()
        image = ref2.get()
    elif (intake == 'After January 2011' and mbbs_student == 'Yes'):
        ref1 = db.reference('chatbot/exam/Classification of Degree/After January 2011/MBBS Programme/description')
        ref2 = db.reference('chatbot/exam/Classification of Degree/After January 2011/MBBS Programme/image')
        description = ref1.get()
        image = ref2.get()
    elif (intake == 'After January 2011' and mbbs_student == 'No'):
        ref1 = db.reference('chatbot/exam/Classification of Degree/After January 2011/others/description')
        ref2 = db.reference('chatbot/exam/Classification of Degree/After January 2011/others/image')
        description = ref1.get()
        image = ref2.get()
    return description, image
```

Figure 3.5.4.3 Firebase.py Firebase Realtime Database Initialization and Functions to Get Value from Database

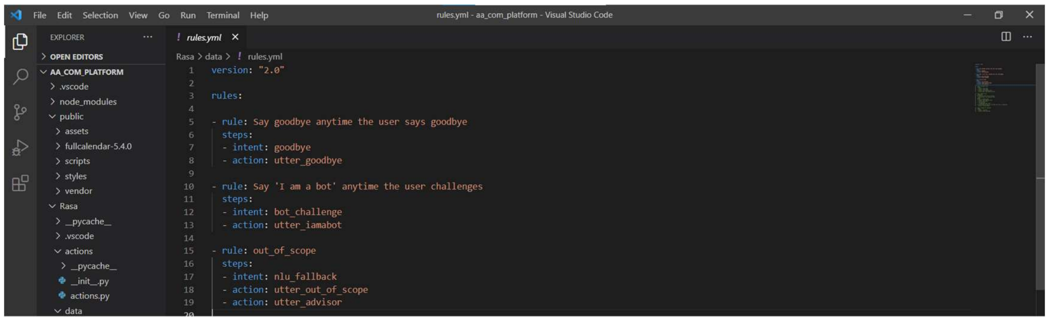
3.5.5 Chatbot Stories and Rules

Both stories and rules are the flow of conversation between user and the chatbot, and are used to train the chatbot for its dialogue management model. Stories is used to train the pattern of conversation in general, whereas rules are conversation path that must be followed strictly and are used to train the RulePolicy of the chatbot. In stories.yml, different stories are declared. Each story includes the name of the story, and also the steps in the story. The steps used in this project are intent, which is the message from user, and action, which is the reply from chatbot. For rules, an out_of_scope rule is declared. This is used for fallback whenever the user asked something that the chatbot have low confidence. Instead of letting the chatbot to reply with a wrong answer, it is a better practice to let chatbot reply with a fallback message instead. In this project, whenever there are any questions with low confidence, the chatbot will let the user to rephrase his answer, and suggest the user whether he wants to ask his advisee instead.



```
- story: grading
  steps:
  - intent: exam_grading_degree
  - action: utter_intake_grading
  - intent: intake
    entities:
    - intake: "After January 2010"
  - slot_was_set:
    - intake: "After January 2010"
  - action: utter_level
  - intent: edu_level
    entities:
    - student_postgrad: "No"
    - slot_was_set:
        - student_postgrad: "No"
  - action: action_grading
```

Figure 3.5.5.1 Stories.yml Story

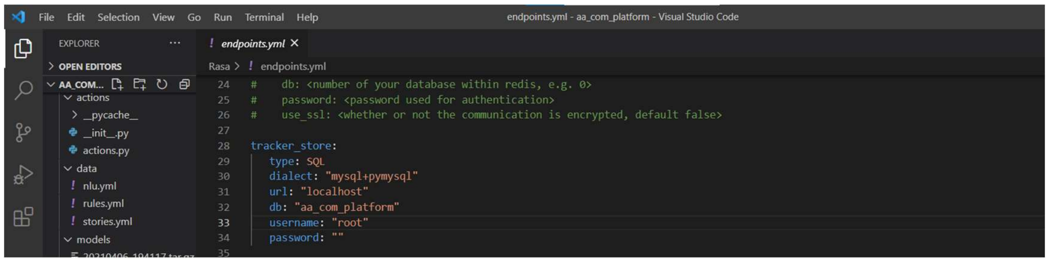


```
version: "2.0"

rules:

- rule: Say goodbye anytime the user says goodbye
  steps:
  - intent: goodbye
  - action: utter_goodbye

- rule: Say 'I am a bot' anytime the user challenges
  steps:
  - intent: bot_challenge
  - action: utter_iamabot

- rule: out_of_scope
  steps:
  - intent: nlu_fallback
  - action: utter_out_of_scope
  - action: utter_advisor
```

Figure 3.5.5.2 Rules.yml Rules

3.5.6 Connecting Chatbot to MySQL Database

In default, the conversation between the chatbot will be stored in the memory, thus if the Rasa server is restarted, all the data will be gone. However, in this project, the whole conversation with the chatbot must be stored in the database for analysis of the data, and thus the chatbot must be connected to the database. In order to connect to MySQL, pymysql is needed to be installed beforehand. After installing, the tracker_store is declared in endpoints.yml. The type of the database is SQL, and dialect is mysql+pymysql. Then, the URL of database, database name, user name and password are set.

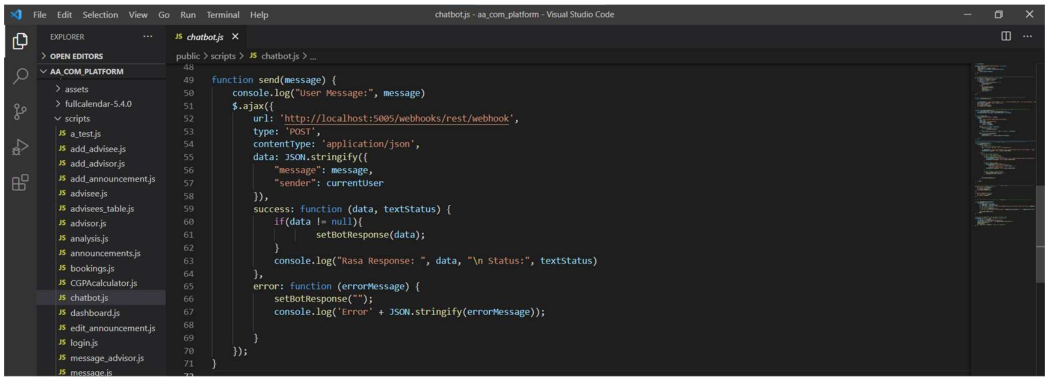


```
#    db: <number of your database within redis, e.g. 0>
#    password: <password used for authentication>
#    use_ssl: <whether or not the communication is encrypted, default false>

tracker_store:
    type: SQL
    dialect: "mysql+pymysql"
    url: "localhost"
    db: "aa_com_platform"
    username: "root"
    password: ""
```

Figure 3.5.6.1 Endpoints.yml Tracker Store

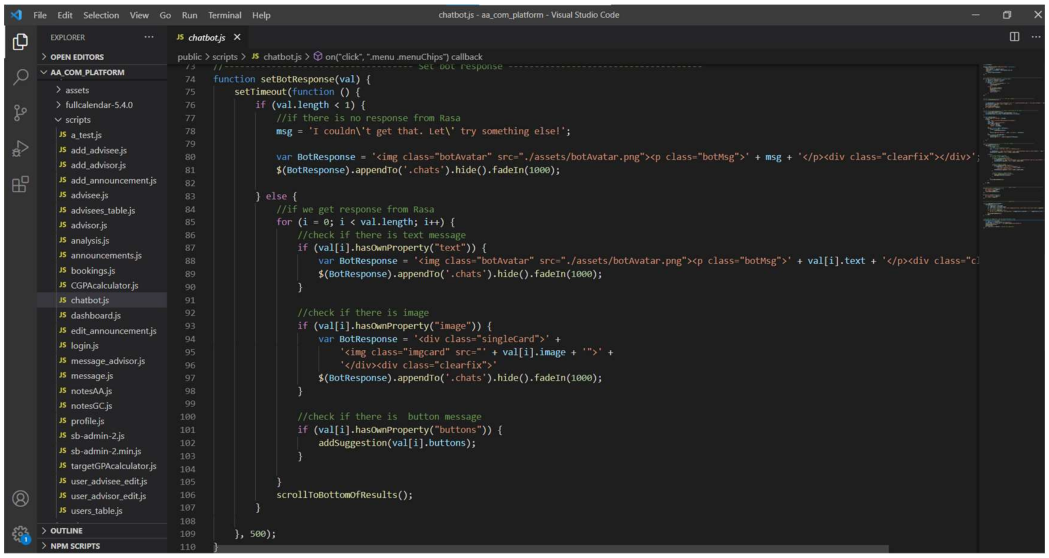
3.5.7 Connecting Chatbot to Website

In chatbot.js send function, a HTTP POST is used to send the message of user to the chatbot. The URL to connect with the chatbot is /webhooks/rest/webhook, and the data send includes the message sent by user, and current user id as the sender. If the message is POST to chatbot successfully, chatbot will return its reply. Then, it will call the function setBotResponse with the reply as its parameter. In setBotResponse, it will identify the type of response and show the response to the user. If it is a text, the reply of chatbot will be shown as a plain text message; if it is an image, the reply will be shown as an image. For buttons, it will show all the buttons and its value.



```
function send(message) {
    console.log("User Message:", message)
    $.ajax({
        url: 'http://localhost:5005/webhooks/rest/webhook',
        type: 'POST',
        contentType: 'application/json',
        data: JSON.stringify({
            "message": message,
            "sender": currentUser
        }),
        success: function (data, textStatus) {
            if(data != null){
                    setBotResponse(data);
            }
            console.log("Rasa Response: ", data, "\n Status:", textStatus)
        },
        error: function (errorMessage) {
            setBotResponse("");
            console.log('Error' + JSON.stringify(errorMessage));

        }
    });
}
```

Figure 3.5.7.1 Chatbot.js Send Message to Chatbot Function



```
//---------------------------------- Set bot response ----------------------------------
function setBotResponse(val) {
    setTimeout(function () {
        if (val.length < 1) {
            //if there is no response from Rasa
            msg = 'I couldn\'t get that. Let\' try something else!';

            var BotResponse = '<img class="botAvatar" src="./assets/botAvatar.png"><p class="botMsg">' + msg + '</p><div class="clearfix"></div>';
            $(BotResponse).appendTo('.chats').hide().fadeIn(1000);

        } else {
            //if we get response from Rasa
            for (i = 0; i < val.length; i++) {
                //check if there is text message
                if (val[i].hasOwnProperty("text")) {
                    var BotResponse = '<img class="botAvatar" src="./assets/botAvatar.png"><p class="botMsg">' + val[i].text + '</p><div class="cl
                    $(BotResponse).appendTo('.chats').hide().fadeIn(1000);
                }

                //check if there is image
                if (val[i].hasOwnProperty("image")) {
                    var BotResponse = '<div class="singleCard">' +
                        '<img class="imgcard" src="' + val[i].image + '">' +
                        '</div><div class="clearfix">'
                    $(BotResponse).appendTo('.chats').hide().fadeIn(1000);
                }

                //check if there is  button message
                if (val[i].hasOwnProperty("buttons")) {
                    addSuggestion(val[i].buttons);
                }

            }
            scrollToBottomOfResults();
        }

    }, 500);
}
```

Figure 3.5.7.2 Chatbot.js Set Bot Response Function

3.6 Summary

The use case diagram and description for advisee, advisor and administrator are included to give a clear view of all the function that advisee, advisor and administrator can perform in this system. The development process of the system is separated to website development and chatbot development, and the functions of the code are explained step by step.

CHAPTER 4 Methodology and Tools

4.1 Overview

In this chapter, the methodology of developing the project is explained. The hardware and software tools used during development of the project are listed out. The timeline of the project is also planned in order to deliver the project on time.

4.2 Methodologies

The methodology used in this project is the Rapid Application Development (RAD) methodology. RAD methodology was adjusted from Software Development Life Cycle (SDLC) phases to develop the program faster to the users. By using this methodology, the project can be iterated and update anytime during the development process as it focuses on rapid prototyping and the feedback.

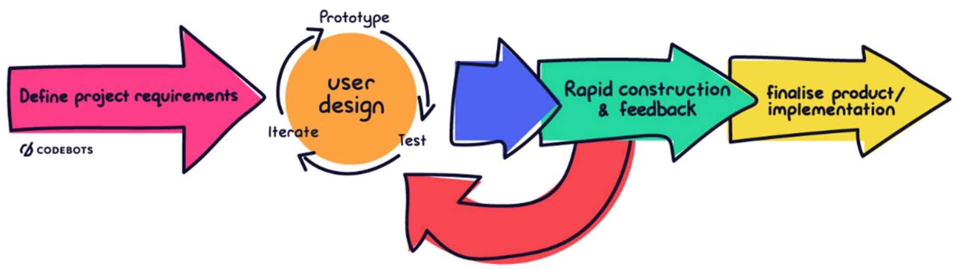

Define project requirements
CODEBOTS
Prototype
user design
Iterate
Test
Rapid construction & feedback
finalise product/ implementation

Figure 4.2.1 RAD Methodology (O'Carroll 2020)

4.2.1 Requirements Planning Phase

The requirements planning phase is the first phase of the methodology. An idea of developing an online communication platform for advisee and advisor is planned and proposed, and the problem statements are written. The objectives of the project are also being listed out in the planning and analysis phase. Literatures and existing systems are reviewed to analyse possible requirements needed by users in the website. The strength and weaknesses of the existing systems are reviewed to develop a well-build website, and the requirements of the project are set. The hardware and software tools used, and the timeline of the project are also being planned during this phase.

4.2.2 User Design

For the user design phase, the prototype of the modules in the project are created, including the login module, profile module, event calendar module, meeting notes module, chatbot module, CGPA and target CGPA calculator module, and administration module. The website is using MySQL, HTML, JavaScript, jQuery and JSON as the languages to develop the functionalities, and HTML and CSS to design the user interface. Besides, web scraping is using Python, and the chatbot is using NLU and Python. The program will then be tested and iterate the prototype if there are some improvements or new requirements needed, and thus the prototype just consists the functions and basic UI design. This phase will be repeated continuously until the project owner agrees to finalize the functions in the prototype.

4.2.3 Construction and Feedback Phase

The third phase of the methodology is the construction phase, which the coding, testing and integration are involved to convert the prototypes into working model. The design of the user interface is then being enhanced using HTML, CSS, and also Bootstrap. Feedback of the user will then be collected, and the user can still propose if the project needs alterations or new functionalities until the project meets the user's expectations.

4.2.4 Finalise Product and Implementation Phase

Four different testing plans including functionality, usability, interface and compatibility tests are implemented. The testing is carried out for all modules in the website. This phase is important to evaluate the functionality of the website has been well built and confirm that the user will not encounter errors when using the website. After that, the documentation of the project is done after all phases are completed. The system design and operation, database design, functionalities and system testing results are recorded. The project is now completed and ready for launch.

4.3 Tools to Use

i. Hardware Requirements

Table 4.3.1 Hardware component required for website development

Component	**Requirements**
Windows	Windows 10 Home Single Language
Processor	Intel® Core™ i7-8550U CPU @ 1.80GHz
Graphic Card	NVIDIA GeForce MX130
RAM	12.00 GB (11.90 GB usable)
Input	Keyboard and mouse
System Type	64-bit operating system, x64-based processor

ii. Software Requirements

Table 4.3.2 Software component required for website development

Component	**Requirements**
Tools	**Visual Studio Code** Visual Studio Code is a code editor which is developed by Microsoft and is provided in cross-platforms, including Windows, Linux and Mac OS X. It has support for JavaScript, TypeScript and Node.js, and extensions can also be added for different languages.
	Firebase Firebase is a platform which is backed by Google. It provides user to build apps fast as it contains many functionalities such as authentication, cloud storage, crash reporting and much more.
	MySQL MySQL is a relational DBMS which uses SQL language and is an open-source software which was first released in 1995.
	Node.js Node.js is an open-source server-side platform built on Chrome's JavaScript engine. It uses asynchronous programming where it is

	non-blocking during handling file request and thus has high efficiency in memory.
	<u>phpMyAdmin</u> phpMyAdmin is an open-source tool written in php to provide an interface for users to manage MySQL in web. There are lots of operations of MySQL can be done in phpMyAdmin.
Languages, Libraries and Frameworks	**<u>Python</u>** Python is an open-source high-level programming language which is high-level built in data structures, dynamic typing and dynamic binding. It has clean syntax and as it emphasises on the readability, thus is very easy to learn and pick up the language.
	<u>Hypertext Markup Language (HTML)</u> HTML is to design the structure of websites and ensure the format of texts and images of the website. It uses either .htm or .html as its file extension.
	<u>Cascading Style Sheets (CSS)</u> CSS is a style sheet language that is used to design the appearance of the websites, such as the layout, fonts and colours. It uses the .css extension for the file name, and it can be shared between multiple pages.
	<u>JavaScript</u> JavaScript is a scripting language which brings functionalities to the website and also enhance the appearance of the website.
	<u>jQuery</u> jQuery is a free library written in JavaScript which is developed by John Resig in 2006, with the purpose of making the client-side HTML scripting easier.
	<u>Bootstrap</u> Bootstrap is a framework of CSS which provides ease on developing mobile-first and responsive websites. It is open source and it supports different browsers.
	<u>Express</u> Express is an open-source framework in Node.js for web application which is simple to be used and yet provides lots of feature for developing websites, web apps and API.

4.4 Timeline

4.4.1 Overview

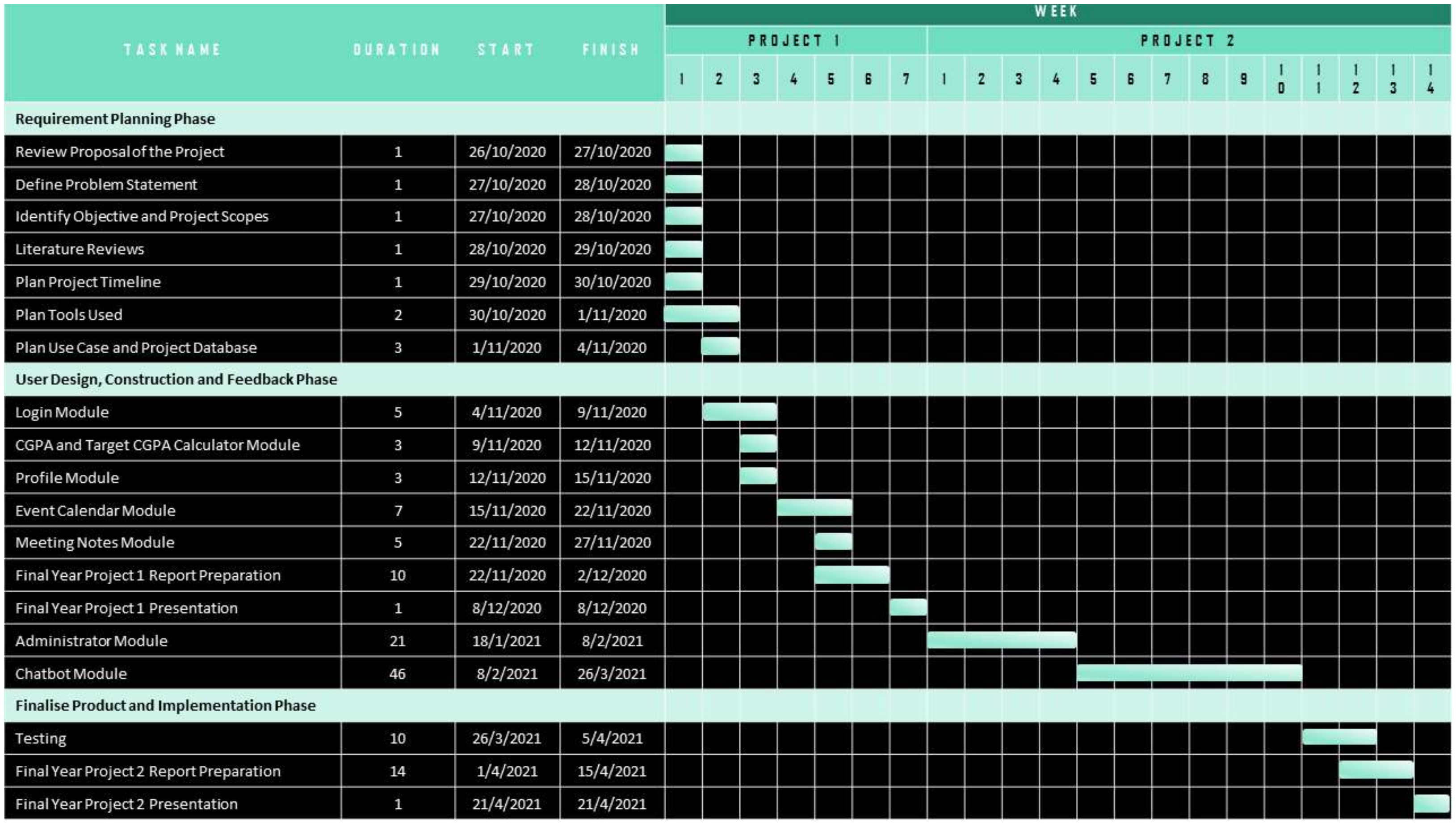
The timeline of the project is planned according to the methodology of the project. Firstly, during the requirement planning phase, the proposal written for the project is reviewed. The problem statement and objective of the project is identified, and the project scope is set in order to achieve the objectives. Few existing websites and similar projects are reviewed and the strength and weaknesses are listed out after reviewing. The project timeline is then planned in order to let the project complete on time. The tools used to develop the website is then chosen after researching. The use case of the project is written and the database tables are created.

Next, during the user design, construction and feedback phase, the login module is done first. This is one of the most important modules as all advisee, advisor and administrator must be logged in in order to interact with the website. Few days are used to research for the ways to implement Firebase Admin SDK for authentication. After that, the CGPA and target CGPA Calculator Module, Profile Module, Event Calendar Module and Meeting Notes Module are developed. The Final Year Project 1 Report is also prepared and presented in this phase.

In Final Year Project 2, the Administrator Module was first developed. The Administrator Module includes Manage Users and Manage Announcements Function. Next, Chatbot Module was developed. At first, the data was scraped from UTAR official website, then stored into database. After that, Rasa chatbot was developed, and lastly connected the chatbot to the website.

During the finalize product and implementation phase, testing of all modules will be done in order to ensure that the project is working fine and to prevent system errors. The chatbot was also being tested by friends and feedback were given. Lastly, the Final Year Project 2 Report was prepared and will be presented to the supervisor and moderator.

4.4.2 Gantt Chart



TASK NAME	DURATION	START	FINISH	WEEK																				
				PROJECT 1							PROJECT 2													
				1	2	3	4	5	6	7	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Requirement Planning Phase																								
Review Proposal of the Project	1	26/10/2020	27/10/2020	🟩																				
Define Problem Statement	1	27/10/2020	28/10/2020	🟩																				
Identify Objective and Project Scopes	1	27/10/2020	28/10/2020	🟩																				
Literature Reviews	1	28/10/2020	29/10/2020	🟩																				
Plan Project Timeline	1	29/10/2020	30/10/2020	🟩																				
Plan Tools Used	2	30/10/2020	1/11/2020	🟩	🟩																			
Plan Use Case and Project Database	3	1/11/2020	4/11/2020		🟩																			
User Design, Construction and Feedback Phase																								
Login Module	5	4/11/2020	9/11/2020		🟩	🟩																		
CGPA and Target CGPA Calculator Module	3	9/11/2020	12/11/2020			🟩																		
Profile Module	3	12/11/2020	15/11/2020			🟩																		
Event Calendar Module	7	15/11/2020	22/11/2020				🟩	🟩																
Meeting Notes Module	5	22/11/2020	27/11/2020					🟩																
Final Year Project 1 Report Preparation	10	22/11/2020	2/12/2020					🟩	🟩															
Final Year Project 1 Presentation	1	8/12/2020	8/12/2020							🟩														
Administrator Module	21	18/1/2021	8/2/2021								🟩	🟩	🟩	🟩										
Chatbot Module	46	8/2/2021	26/3/2021												🟩	🟩	🟩	🟩	🟩	🟩				
Finalise Product and Implementation Phase																								
Testing	10	26/3/2021	5/4/2021																		🟩	🟩		
Final Year Project 2 Report Preparation	14	1/4/2021	15/4/2021																			🟩	🟩	
Final Year Project 2 Presentation	1	21/4/2021	21/4/2021																					🟩

Figure 4.4.2.1 Gantt Chart of the Project Timeline

4.5 Summary

All in all, the methodology used in this project is stated, which is the RAD methodology. RAD methodology helps in reducing risks in the project as testing are done in almost every phase and the errors can be identified at earlier stage and can be solved earlier. It is also convenient as changes and functionalities add on can be easily done during the process of development. The hardware and software tools used to develop the front end and back end of the system are also defined in this chapter. The timeline of the project has been planned and shown in a Gantt chart for better time management to develop the project and ensure that there are no delays on the progress of each task.

CHAPTER 5 IMPLEMENTATION AND TESTING

5.1 Overview

All 7 modules of the project are developed, including the login module, profile module, event calendar module, meeting notes module, chatbot module, CGPA and target CGPA calculator module and administration module. The website is named AAOCP and it has a blue theme, and it is fully mobile responsive. The main pages except the login page consists of a top navigation bar, which users can view their name, profile photo, and a message button to redirect to chat page. There is also a side navigation bar where users can select which page they would like to interact with. There are some pages can be accessed by all advisee, advisor and administrator, such as the dashboard and profile; some can only be accessed by a particular type of user, such as manage bookings can only be accessed by the advisor, whereas manage users and manage announcements can only be accessed by the administrator. Testing is also made for the chatbot, and feedback is being collected and discussed.

5.2 Login Page

When a user access to the default URL of the website, the user will be redirected to the login page. Users are required to sign in by inserting their email and password. If there is no existing user or the combination of email and password is wrong, the system will alert the user and the user will need to edit their sign in information. If the email and password is valid, the user will be given a session cookie which will be expiring in 5 days starting from the login of the user. The system will check whether there is an active session cookie, and the user will be redirected to the dashboard of the website. If the user forgets the password, he or she can click on the reset button and the system will show a column for the user to fill in their email.

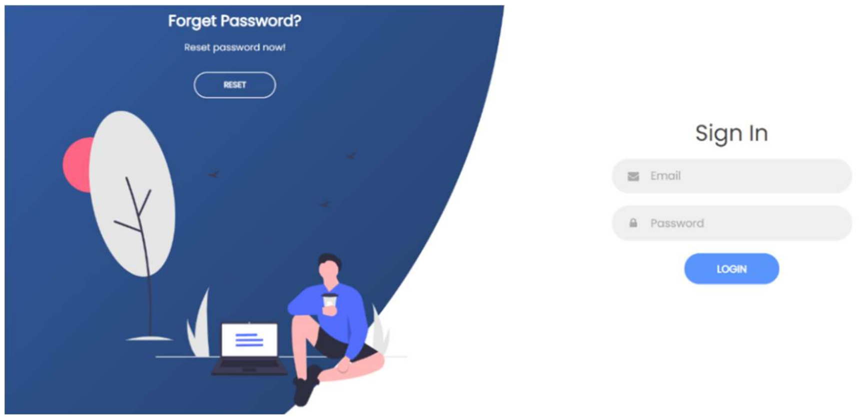
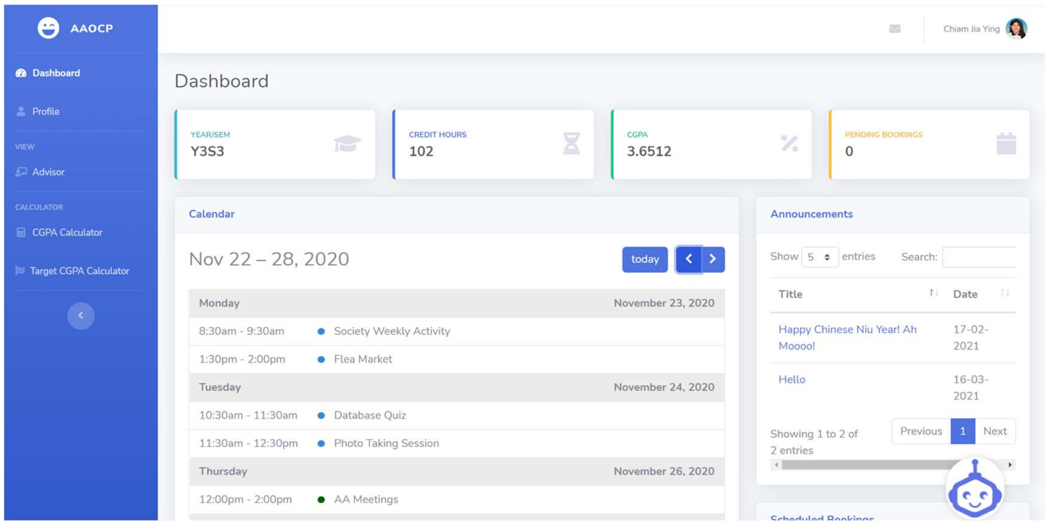


Figure 4.4.2.1 Login Page

5.3 Dashboard

The dashboard of the user consists of some summarized information in the website. For advisee and advisor, the information shown in dashboard includes personal details, website announcements and bookings. The schedule of users can also be viewed in the dashboard. Chatbot is also provided for advisees to ask for enquiries, and administrator can also view the chatbot analysis in this page.



Monday	November 23, 2020
8:30am - 9:30am	Society Weekly Activity
1:30pm - 2:00pm	Flea Market
Tuesday	November 24, 2020
10:30am - 11:30am	Database Quiz
11:30am - 12:30pm	Photo Taking Session
Thursday	November 26, 2020
12:00pm - 2:00pm	AA Meetings

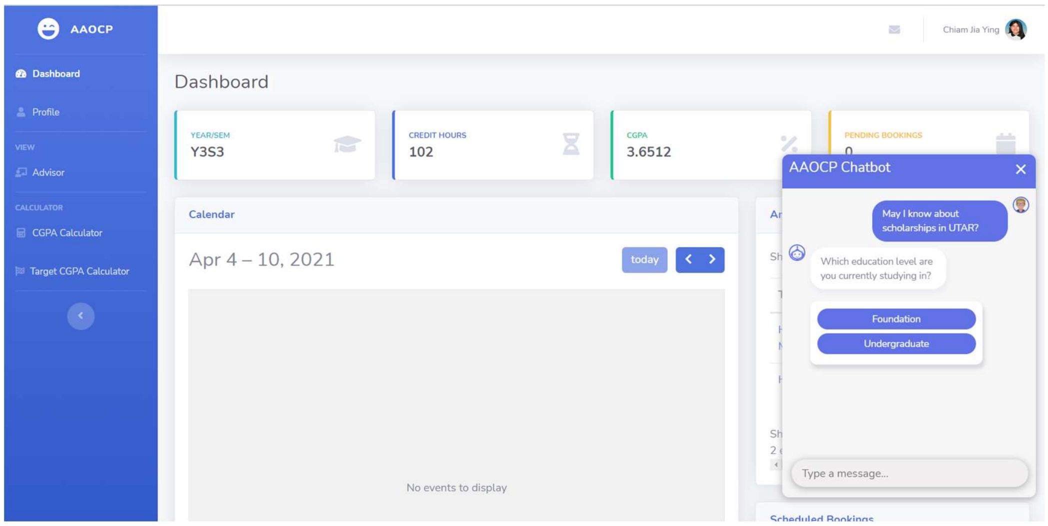
Title	Date
Happy Chinese Niu Year! Ah Moooo!	17-02-2021
Hello	16-03-2021

Showing 1 to 2 of 2 entries

Figure 5.3.1 Dashboard of Advisee

5.4 Chatbot

Chatbot is provided to advisees to ask about their enquiries 24/7. Advisees can enter their enquiry first, and the chatbot will reply them accordingly. As in Figure 5.4.1, if the answer of the chatbot is different according to the condition, the chatbot will pop up different buttons to let advisees choose the correct condition. If advisee asked about something that is out of scope, the chatbot will ask advisee to rephrase the sentence and also suggest him to message their advisor instead. The questions asked by the advisees will be stored in database and the analysis of the enquiries will be shown to advisor and administrator in the Enquiry Analysis page. Administrator can also view the analysis in his dashboard.



May I know about scholarships in UTAR?

Which education level are you currently studying in?

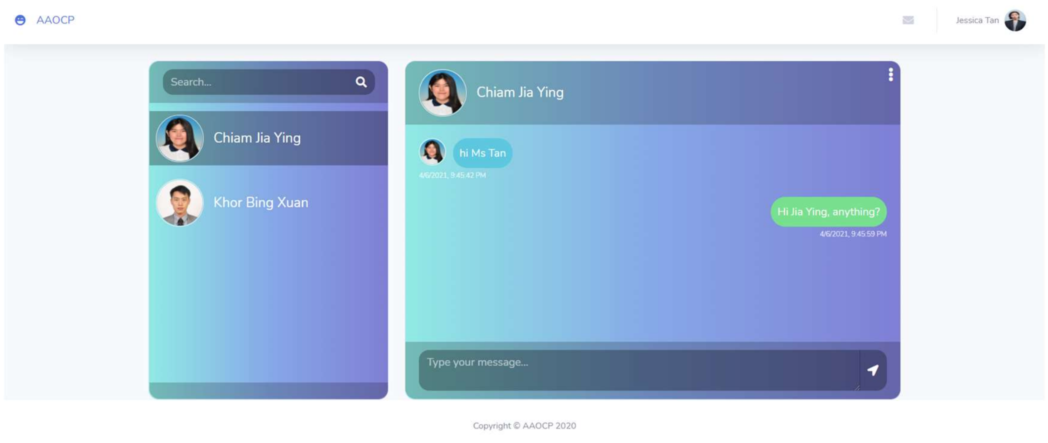
Foundation

Undergraduate

Figure 5.4.1 Chatbot

5.5 Chat

In the top navigation bar of every page, there will be a message icon. Advisee and advisor can click on the message button to proceed to the chat page. Advisor can view all of the advisees he managed in the side bar, and select who to chat with. For advisee, he can only chat with his advisor. The message sent by the opposite site will be in blue colour, and the message sent by the user will be in green. The sent time of the message will also be shown under every message. To send message to the opposite side, user just need to type the message inside the text box and click the send button, or click enter. Users can also view the profile of the opposite site by pressing the option icon button at the top right of the chat box. There will be a dropdown and user can click on View Profile to be redirected to the profile.



hi Ms Tan

4/6/2021, 9:45:42 PM

Hi Jia Ying, anything?

4/6/2021, 9:45:59 PM

Figure 5.5.1 Chat Messages

5.6 Profile

In the profile page, advisee, advisor and administrator can view their name, faculty, course or department, phone number and email. Advisees can also view their CGPA, current semester, and credit hours earned in the page. There is an edit button at the top right of the profile page where they can edit their profile, however the editing of profile is limited as most of the information are not meant to be changed. Besides, there is a personal timetable where users can choose to view the timetable in month view or weekly view. The timetable is to ease users to plan their schedule systematically and to let them have a clear overview of their personal events. The timetable will show previously added events, and for advisor and advisee, the calendar will also show accepted bookings for meetings by clicking the "+" sign at the top right of the timetable, and next clicking on the "Add Event" link. They are required to fill in the event title, start date, end date, and description of the event. The user can also edit or delete the events by clicking on the event in the calendar. There will be a popup modal to let users choose either to edit or delete the event if there is wrong information inserted.

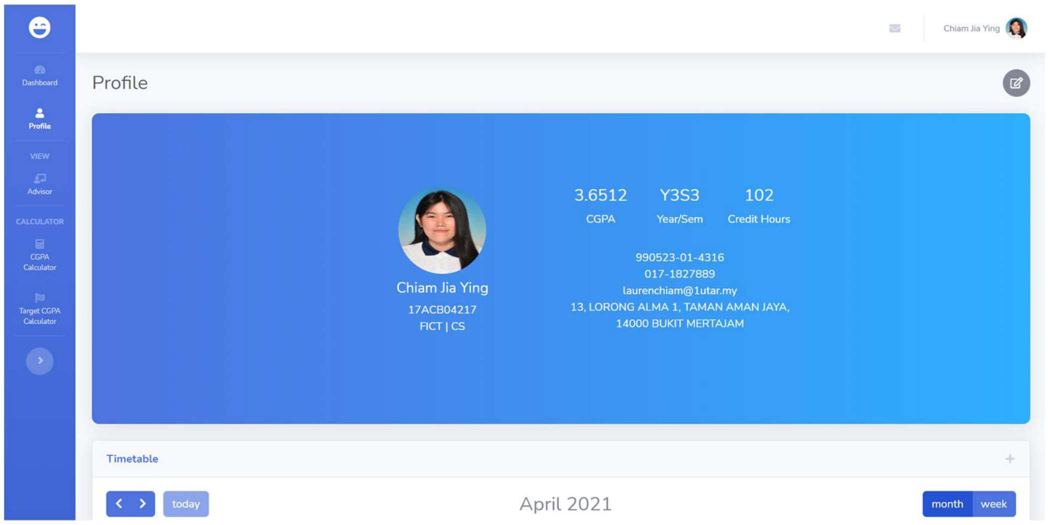


Figure 5.6.1 Profile Page

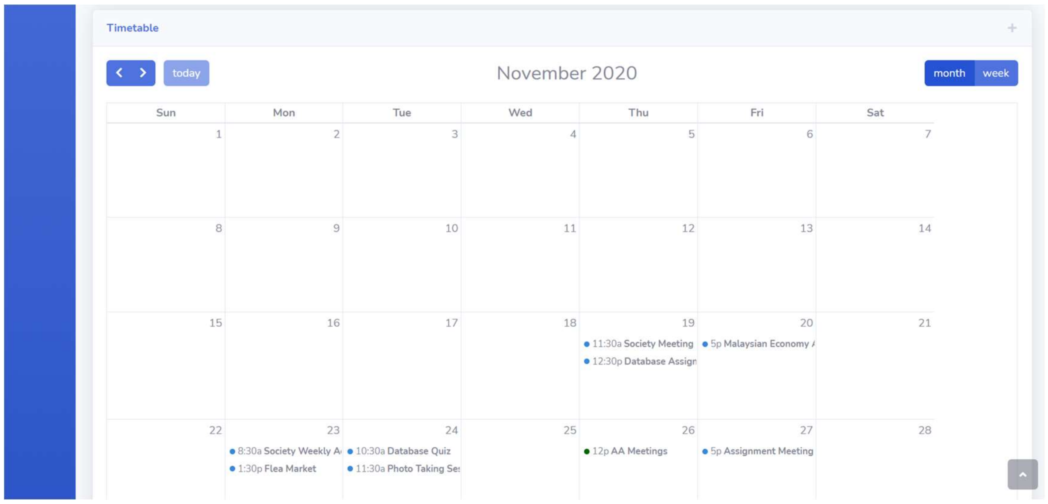
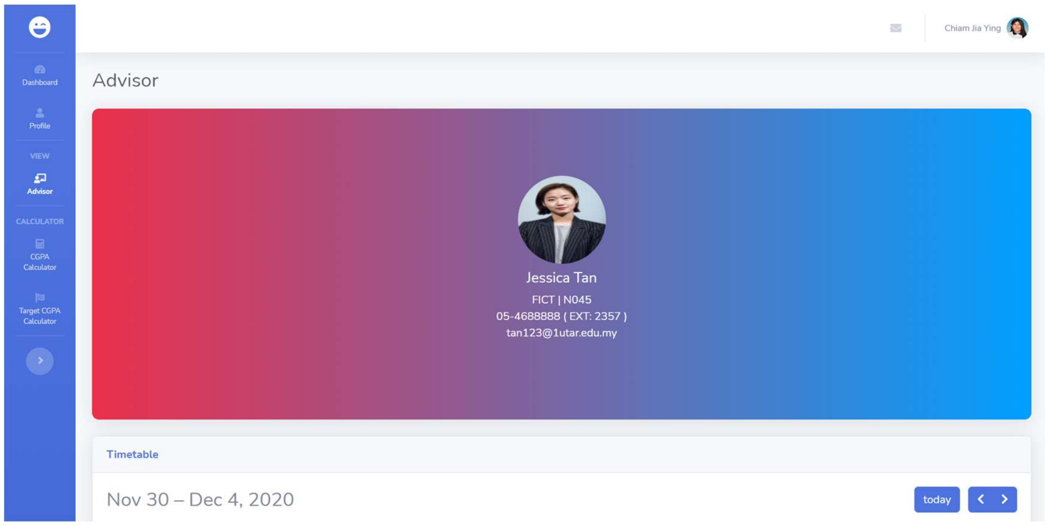


Figure 5.6.2 Timetable in Profile Page

5.7 Advisor Page

Advisee is able to view their advisor's personal information in the advisor page. Besides, the working hours timetable of the advisor can also be viewed by the advisee. The event in blue is the personal event of the advisor, and event in green is accepted booking from advisees. The timetable gives all the advisees a clear view about the availability of advisor and they can proceed to book a meeting. There is a section called "My Bookings with Advisor" where all the bookings done will be shown in a table with the status of the booking and the actions can be done with the booking. If the meeting is done by the advisee, there will be an edit and delete button where advisee can make amendments before reaching the booking start time. If the meeting is booked by the advisor, there will be a tick button and a "x" button, where advisee can choose to accept or reject the booking. ACCEPTED bookings will be shown in the event calendar of both advisor and advisee, while for rejected bookings, the reason rejected can be seen by clicking on the REJECTED word. Advisee can add new bookings by clicking the "+" sign and then the "Add Booking Button". A pop up will be shown to let user fill in the details of the booking, and after saving, the booking will be in a pending status.



Jessica Tan
FICT | N045
05-4688888 (EXT: 2357)
tan123@1utar.edu.my

Nov 30 – Dec 4, 2020

Figure 5.7.1 Advisor Page

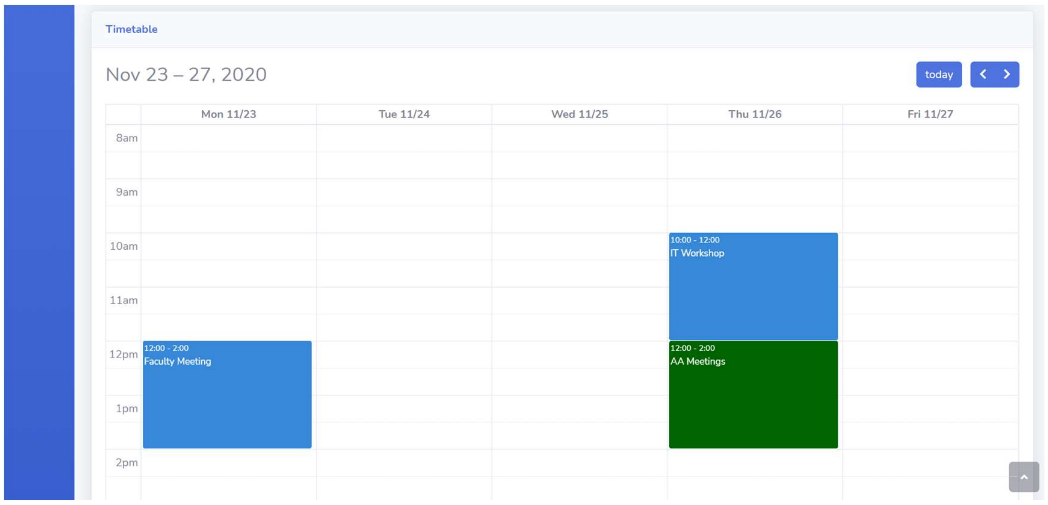
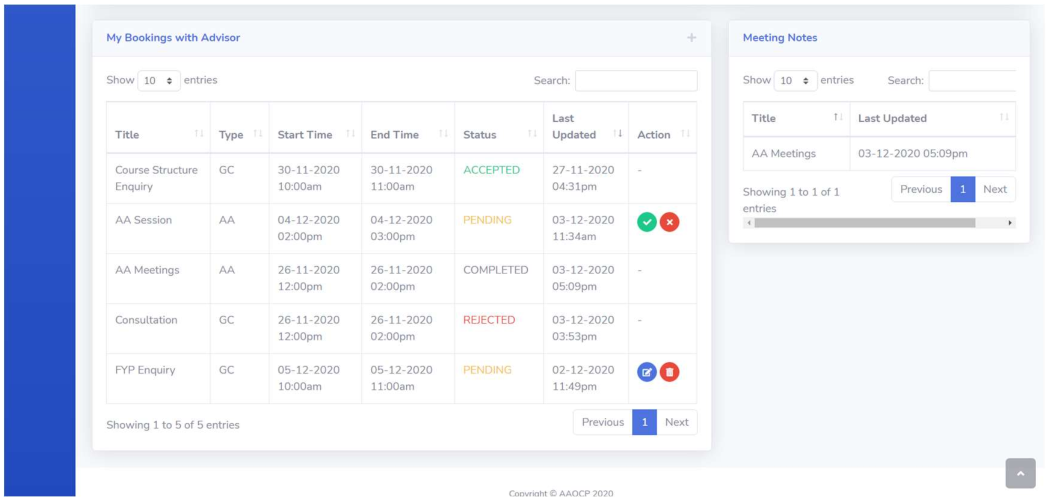


Figure 5.7.2 Timetable of Advisor


My Bookings with Advisor

Show 10 entries Search:

Title	Type	Start Time	End Time	Status	Last Updated	Action
Course Structure Enquiry	GC	30-11-2020 10:00am	30-11-2020 11:00am	ACCEPTED	27-11-2020 04:31pm	-
AA Session	AA	04-12-2020 02:00pm	04-12-2020 03:00pm	PENDING	03-12-2020 11:34am	✓ ✗
AA Meetings	AA	26-11-2020 12:00pm	26-11-2020 02:00pm	COMPLETED	03-12-2020 05:09pm	-
Consultation	GC	26-11-2020 12:00pm	26-11-2020 02:00pm	REJECTED	03-12-2020 03:53pm	-
FYP Enquiry	GC	05-12-2020 10:00am	05-12-2020 11:00am	PENDING	02-12-2020 11:49pm	✎ 🗑

Showing 1 to 5 of 5 entries Previous 1 Next

Meeting Notes

Show 10 entries Search:

Title	Last Updated
AA Meetings	03-12-2020 05:09pm

Showing 1 to 1 of 1 entries Previous 1 Next

Copyright © AAOCP 2020

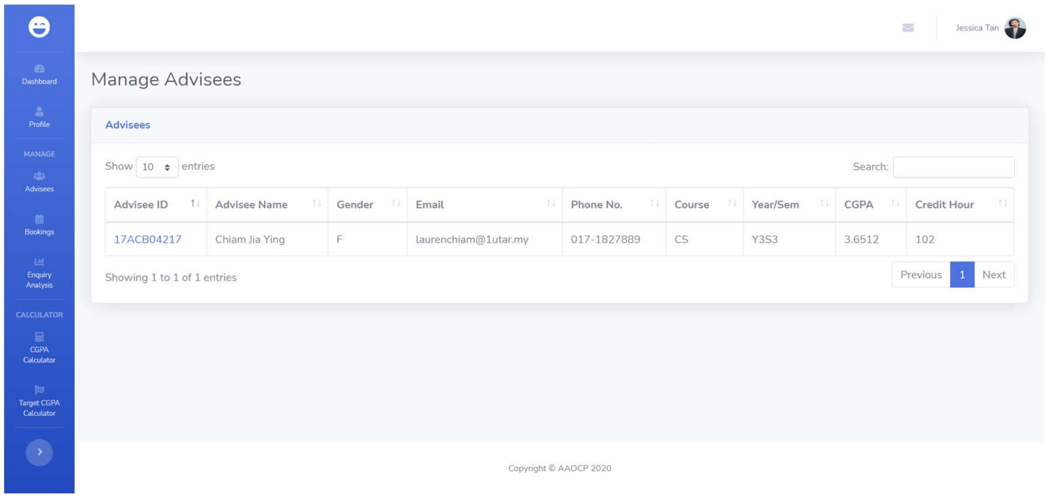
Figure 5.7.3 Bookings with Advisor and View Meeting Notes

5.8 Manage Advisees

As advisors would need to know all the information of the advisees to have a better understanding, advisor is able to view all the advisees' profile which is under his supervision. After clicking on the advisee link on the side nav bar, the advisor will see a table of summarized information of all the advisees. If the advisor wants to view the information of an advisee in details, the advisor can click on the advisee ID in the table, and the page will be redirected to the advisee's profile. The detailed information of the advisee and timetable of the advisee will be shown.

To view all bookings with the particular advisee, advisor can view the table in "Scheduled Bookings with Advisee" section. The table consists of booking in all statuses, including the accepted, rejected, pending and completed meetings, no matter the event is booked by the advisor or advisee. Advisor can then perform accepting or rejecting a booking in the table if the booking is done by the advisee, or edit and delete the booking if the booking is done by advisor himself. Besides, advisor can also add new bookings with the advisee by clicking the "+" button in the "Scheduled Bookings with Advisee" section and then the "Book Meeting" link. Next, advisor will be prompted to fill in all details of the booking, and a pending booking will be added.

In order to add a new meeting note, advisor can click on the "+" button in the "Meeting Notes" section and then the "Add Meeting Notes" link. The advisor will be prompted to choose one of the accepted meetings and the page will then be redirected to "Add Meeting Notes" page, and it will have two different layouts for academic advising meetings and general consultation meetings. After saving the meeting note, the meeting will be completed and the status will be updated.



Manage Advisees

Advisees

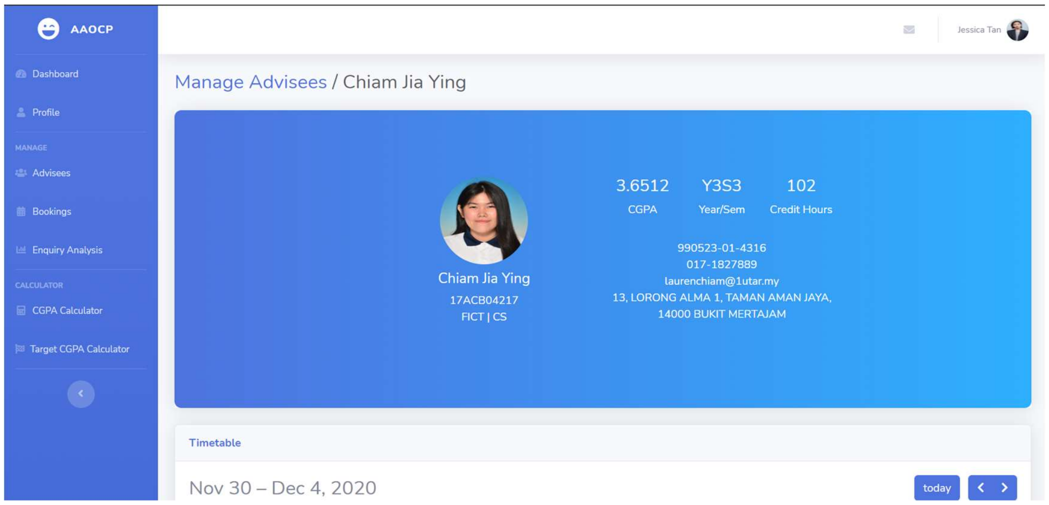
Show 10 entries Search:

Advisee ID	Advisee Name	Gender	Email	Phone No.	Course	Year/Sem	CGPA	Credit Hour
17ACB04217	Chiam Jia Ying	F	laurenchiam@1utar.my	017-1827889	CS	Y3S3	3.6512	102

Showing 1 to 1 of 1 entries Previous 1 Next

Copyright © AAOCP 2020

Figure 5.8.1 Manage Advisees Page



Manage Advisees / Chiam Jia Ying

Chiam Jia Ying
17ACB04217
FICT | CS

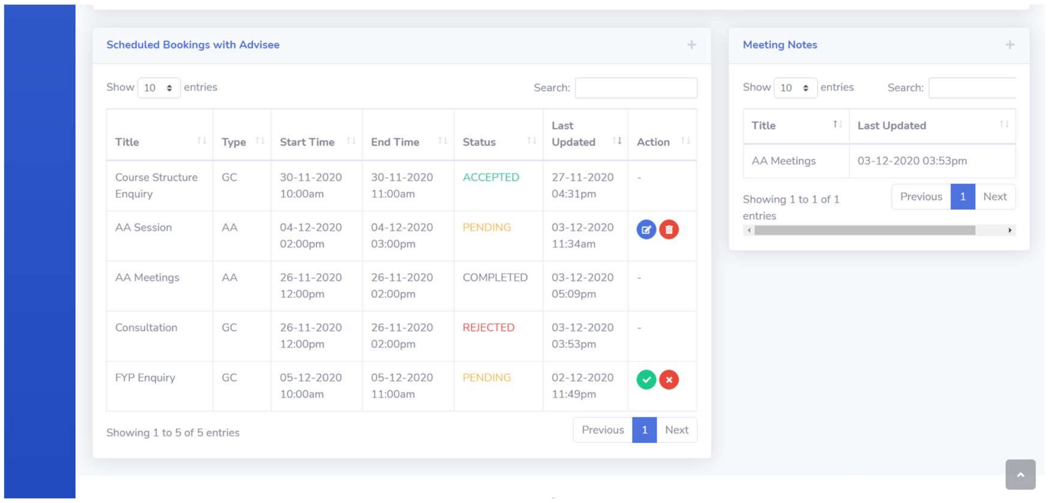
3.6512 CGPA
Y3S3 Year/Sem
102 Credit Hours

990523-01-4316
017-1827889
laurenchiam@1utar.my
13, LORONG ALMA 1, TAMAN AMAN JAYA,
14000 BUKIT MERTAJAM

Timetable

Nov 30 – Dec 4, 2020

Figure 5.8.2 Advisee Profile and Timetable



Scheduled Bookings with Advisee

Show 10 entries Search:

Title	Type	Start Time	End Time	Status	Last Updated	Action
Course Structure Enquiry	GC	30-11-2020 10:00am	30-11-2020 11:00am	ACCEPTED	27-11-2020 04:31pm	-
AA Session	AA	04-12-2020 02:00pm	04-12-2020 03:00pm	PENDING	03-12-2020 11:34am	
AA Meetings	AA	26-11-2020 12:00pm	26-11-2020 02:00pm	COMPLETED	03-12-2020 05:09pm	-
Consultation	GC	26-11-2020 12:00pm	26-11-2020 02:00pm	REJECTED	03-12-2020 03:53pm	-
FYP Enquiry	GC	05-12-2020 10:00am	05-12-2020 11:00am	PENDING	02-12-2020 11:49pm	✓ ✗

Showing 1 to 5 of 5 entries Previous 1 Next

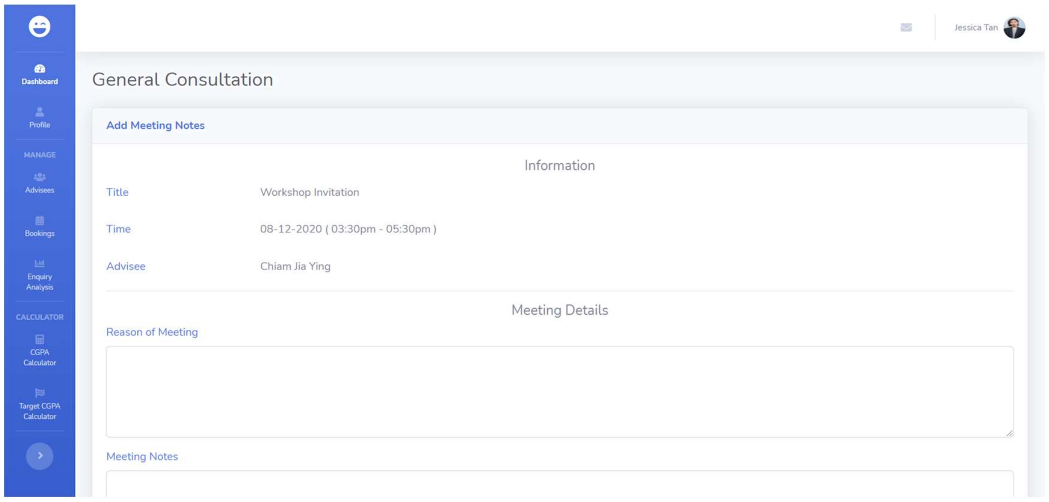
Meeting Notes

Show 10 entries Search:

Title	Last Updated
AA Meetings	03-12-2020 03:53pm

Showing 1 to 1 of 1 entries Previous 1 Next

Figure 5.8.3 Scheduled Bookings with Advisee and Meeting Notes



General Consultation

Add Meeting Notes

Information

Title: Workshop Invitation

Time: 08-12-2020 (03:30pm - 05:30pm)

Advisee: Chiam Jia Ying

Meeting Details

Reason of Meeting

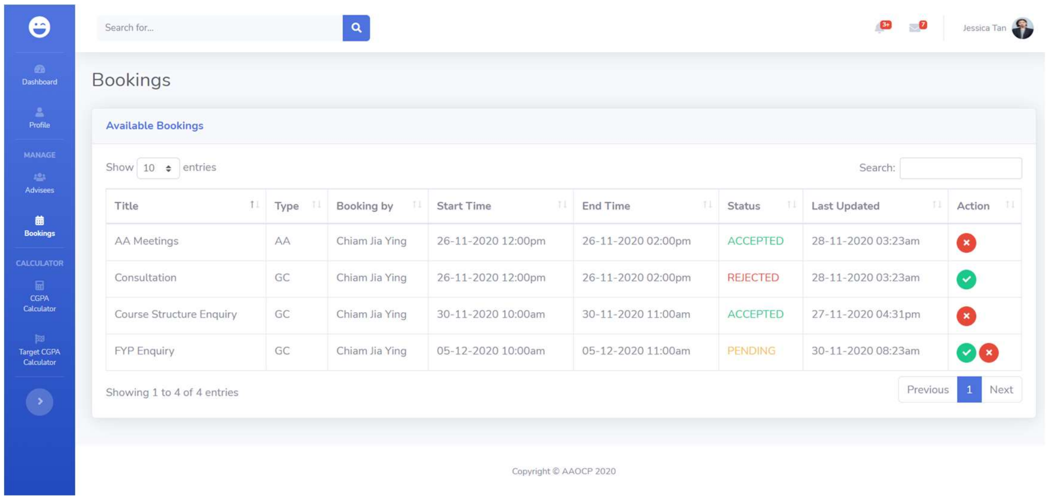
Meeting Notes

Figure 5.8.4 Add Meeting Notes

5.9 Manage Bookings

As advisors are often busy and could not check the bookings by advisee one by one in their profile, thus there is a manage booking page where advisor can manage all the available bookings by the advisees under him. The bookings are shown in a table, with the title, type, booking by, start time, end time, status and last updated time. The booking description can also be viewed in a pop up when clicked on the booking title.

A new booking will have a pending status, and the advisor can either choose to accept or reject the booking. If accepted, the last updated time will be updated, and the status of the event will be changed to accepted. When the advisor accepts a booking and there are some other pending or accepted bookings are overlapping the current booking, the system will alert the advisor the list of overlapped bookings, and if the advisor confirms to proceed, the overlapped bookings will be rejected automatically, with the reason of overlapping other bookings. This will ease the advisor as the advisor does not need to filter and reject the overlapping bookings one by one. If advisor chose to reject, he would need to fill in the reason in the pop up of why he rejected the booking, such as he will not be free during that booking period, and the advisee who booked the meeting can have a better understanding on why the meeting is rejected. Events which are accepted will only show the reject button in the action column, and for events which are rejected, only accepted button will be shown.



Bookings

Available Bookings

Show 10 entries

Search:

Title	Type	Booking by	Start Time	End Time	Status	Last Updated	Action
AA Meetings	AA	Chiam Jia Ying	26-11-2020 12:00pm	26-11-2020 02:00pm	ACCEPTED	28-11-2020 03:23am	✗
Consultation	GC	Chiam Jia Ying	26-11-2020 12:00pm	26-11-2020 02:00pm	REJECTED	28-11-2020 03:23am	✓
Course Structure Enquiry	GC	Chiam Jia Ying	30-11-2020 10:00am	30-11-2020 11:00am	ACCEPTED	27-11-2020 04:31pm	✗
FYP Enquiry	GC	Chiam Jia Ying	05-12-2020 10:00am	05-12-2020 11:00am	PENDING	30-11-2020 08:23am	✓ ✗

Showing 1 to 4 of 4 entries

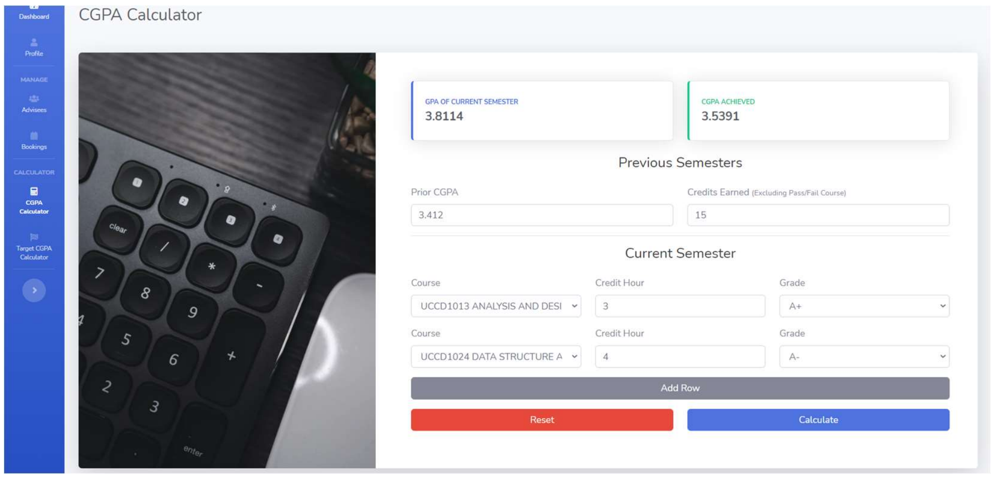
Previous 1 Next

Copyright © AAOCP 2020

Figure 5.9.1 Manage Bookings

5.10 CGPA Calculator

The CGPA calculator is provided for both advisor and advisees to calculate estimated GPA and CGPA for the semester. Advisor or advisee can first enter the CGPA obtained previously, and the credit hours earned excluding pass or fail subjects. Next, advisor or advisee can fill in the credit hour of a course directly without choosing the course code, or choose a course for the current semester first, and the credit hour section will automatically be updated to the credit hour of the course. If there is more than one course taken, advisor or advisee can click on the add row button and there will be a new row to input the information. The grade section will be the predicted grade of the advisee for the subject. Then, the calculate button is clicked to obtain the estimated GPA of current semester and the CGPA achieved. The reset button can be clicked if the user wants to do a new calculation, and the page will be refreshed.



CGPA Calculator

GPA OF CURRENT SEMESTER: 3.8114

CGPA ACHIEVED: 3.5391

Previous Semesters

Prior CGPA: 3.412

Credits Earned (Excluding Pass/Fail Course): 15

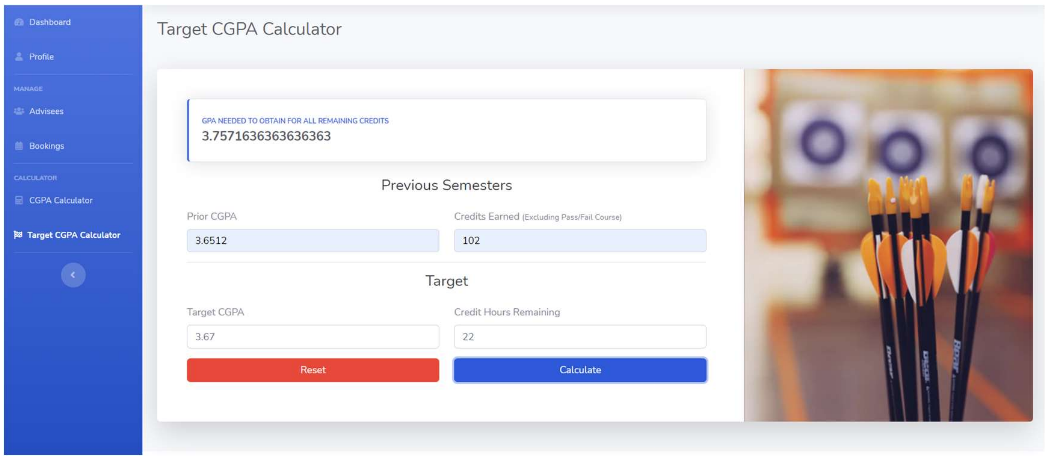
Current Semester

Course	Credit Hour	Grade
UCCD1013 ANALYSIS AND DESI	3	A+
UCCD1024 DATA STRUCTURE A	4	A-

Figure 5.10.1 CGPA Calculator Page

5.11 Target CGPA Calculator

The target CGPA calculator is to let users know the estimated average GPA needed for the remaining credit hours in order to gain the CGPA targeted by the advisee. This target CGPA calculator can be used by both advisor and advisee. There are 4 columns in the form including prior CGPA, Credits Earned, Target CGPA and Credit Hours Remaining, and they are all required to be filled in. There will be an alert if the advisee or advisor does not fill in the column. After clicking on the calculate button, the GPA needed to be obtained to achieve the target CGPA will be shown on the top of the calculator.



Target CGPA Calculator

GPA NEEDED TO OBTAIN FOR ALL REMAINING CREDITS
3.7571636363636363

Previous Semesters

Prior CGPA: 3.6512
Credits Earned (Excluding Pass/Fail Course): 102

Target

Target CGPA: 3.67
Credit Hours Remaining: 22

Figure 5.11.1 Target CGPA Calculator Page

5.12 Enquiry Analysis

All the enquiries asked by advisees in the chatbot are stored in the database, and advisors and administrators can track the categories of the enquiries asked. The pie chart shows the categories of enquiries asked by advisee in current month, and also all categories provided by the chatbot. When user hover the category in the pie chart, it will show the total number of questions asked by advisees for the particular category. Besides, user can also click on the pie chart category, and the detailed information of the category selected will be shown in the bar chart. For example, in Figure 5.12.1, the bar chart shows the enquiry analysis of Financial Aid Category for April, and it consists of 1 question about loan, and another 1 question about scholarship.

Besides, there is also an enquiry fallback table. When user asked a question where the chatbot does not know how to answer, the question will be shown in this table. With this function, if the chatbot does not understand the question due to different sentence structure, the administrator will acknowledge the limit of the chatbot, and refined version of the chatbot can be made according to the questions from the fallback table. Administrator can also know about which questions are currently not supported by the chatbot but is frequently asked by the advisee, and thus the administrator can add on more categories for the chatbot.

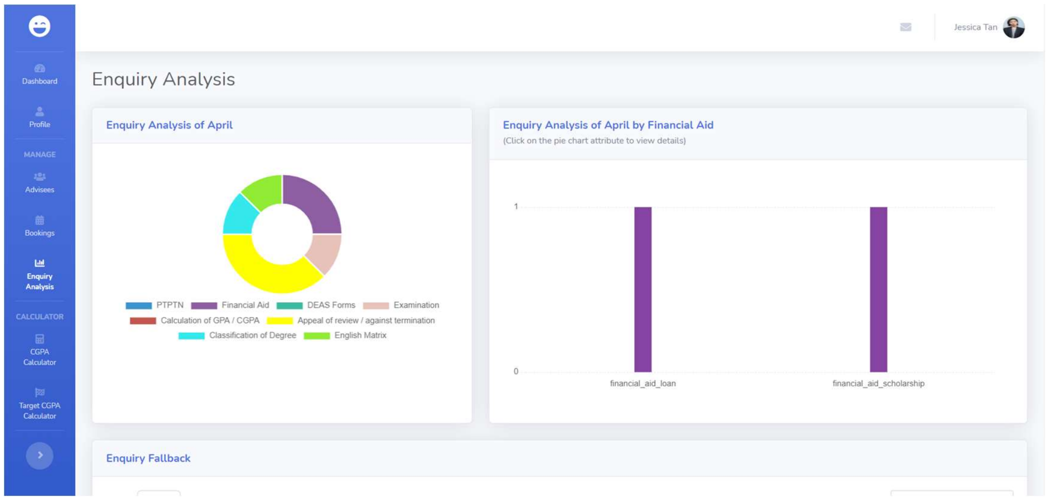
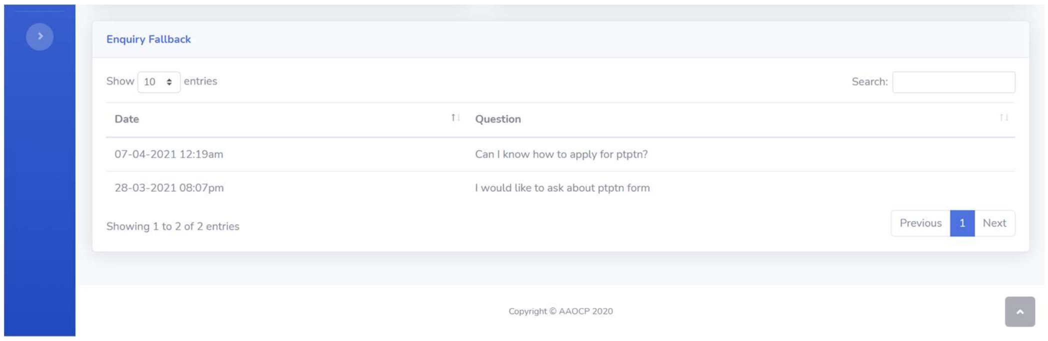


Figure 5.12.1 Enquiry Analysis According to Month and Categories



Enquiry Fallback

Show 10 entries Search:

Date	Question
07-04-2021 12:19am	Can I know how to apply for ptptn?
28-03-2021 08:07pm	I would like to ask about ptptn form

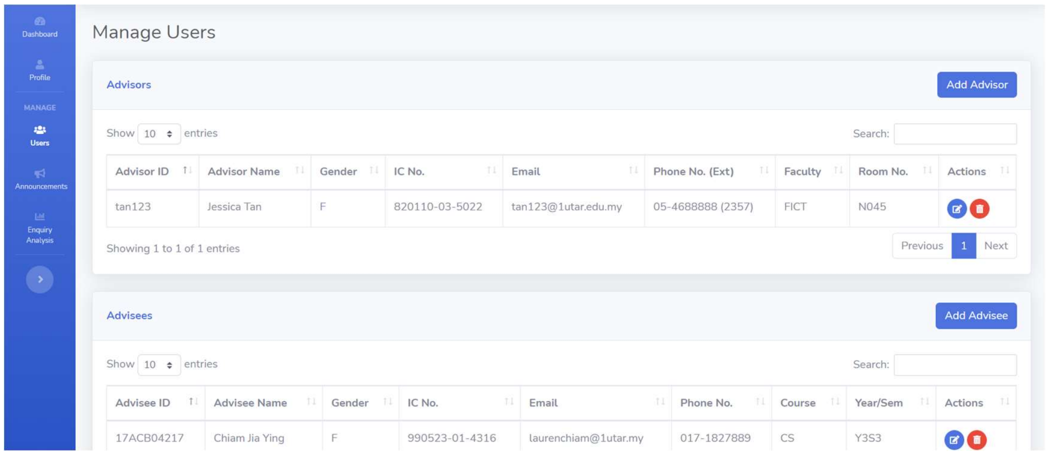
Showing 1 to 2 of 2 entries Previous 1 Next

Copyright © AAOCP 2020

Figure 5.12.2 Enquiry Fallback Table

5.13 Manage Users (Advisee/Advisor)

Administrator can view all the users in the manage users page. The users are separated into two tables, one for advisor, and another table for advisee. There is a add advisor or add advisee button at the top right corner of each table, and after clicking on it, administrator will be redirect to add advisor or advisee page according to the button he or she pressed. Administrator is required to fill in all details of the user and upload the user's profile picture to create a new user. Besides, the brief details of the users can be shown in the table, and there is and edit and delete button for each of the row. Administrator can edit the particular advisee or advisor by clicking the edit button, and the page will be redirected to edit advisee or advisor page. All of the information of the user can be edited except the ID and email of the user. The delete button in the table can also be used to delete the particular user, however once the user is deleted, it can't be recovered again. Thus, when the administrator clicks on the delete button, there will be a pop up to let administrator to confirm the deletion of user.



Manage Users

Advisors

Add Advisor

Show 10 entries Search:

Advisor ID	Advisor Name	Gender	IC No.	Email	Phone No. (Ext)	Faculty	Room No.	Actions
tan123	Jessica Tan	F	820110-03-5022	tan123@1utar.edu.my	05-4688888 (2357)	FICT	N045	

Showing 1 to 1 of 1 entries

Previous 1 Next

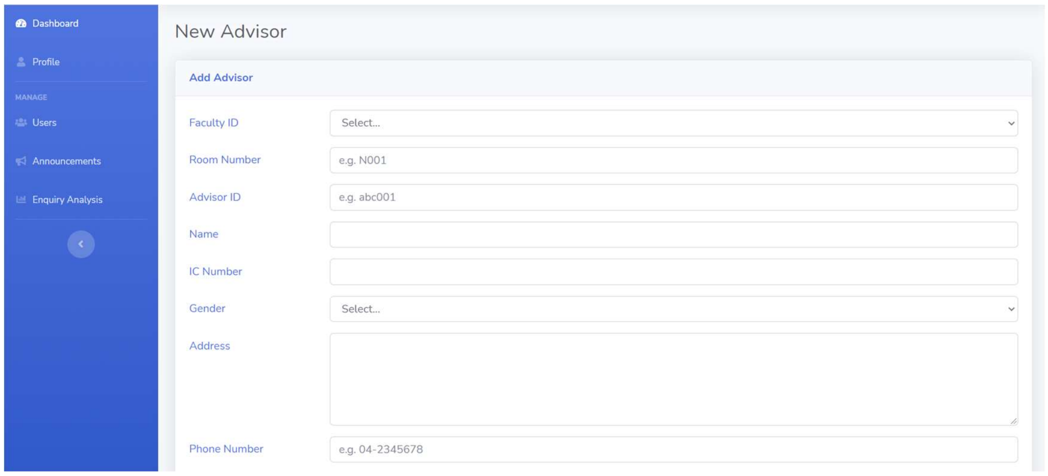
Advisees

Add Advisee

Show 10 entries Search:

Advisee ID	Advisee Name	Gender	IC No.	Email	Phone No.	Course	Year/Sem	Actions
17ACB04217	Chiam Jia Ying	F	990523-01-4316	laurenchiam@1utar.my	017-1827889	CS	Y3S3	

Figure 5.13.1 Manage Users Page

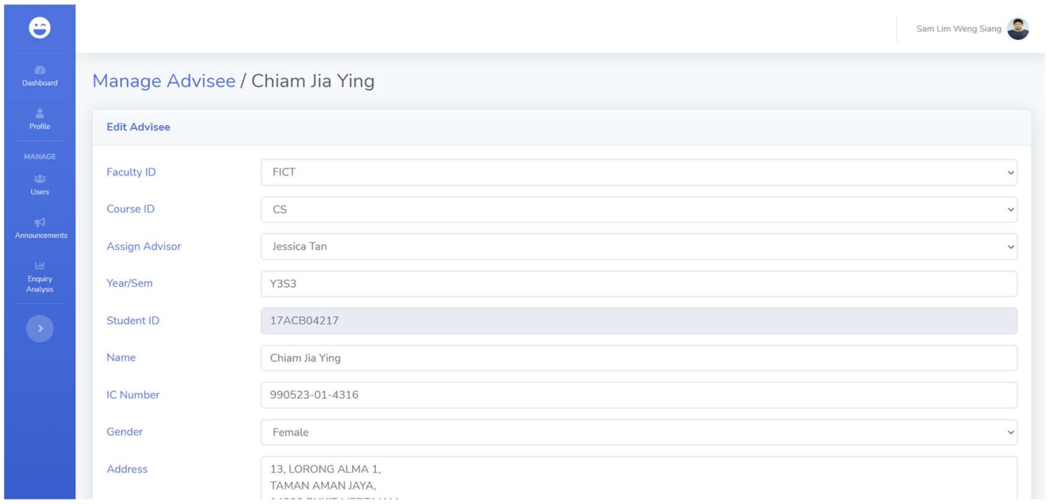


New Advisor

Add Advisor

Faculty ID	Select...
Room Number	e.g. N001
Advisor ID	e.g. abc001
Name	
IC Number	
Gender	Select...
Address	
Phone Number	e.g. 04-2345678

Figure 5.13.2 Add New User Page



Manage Advisee / Chiam Jia Ying

Edit Advisee

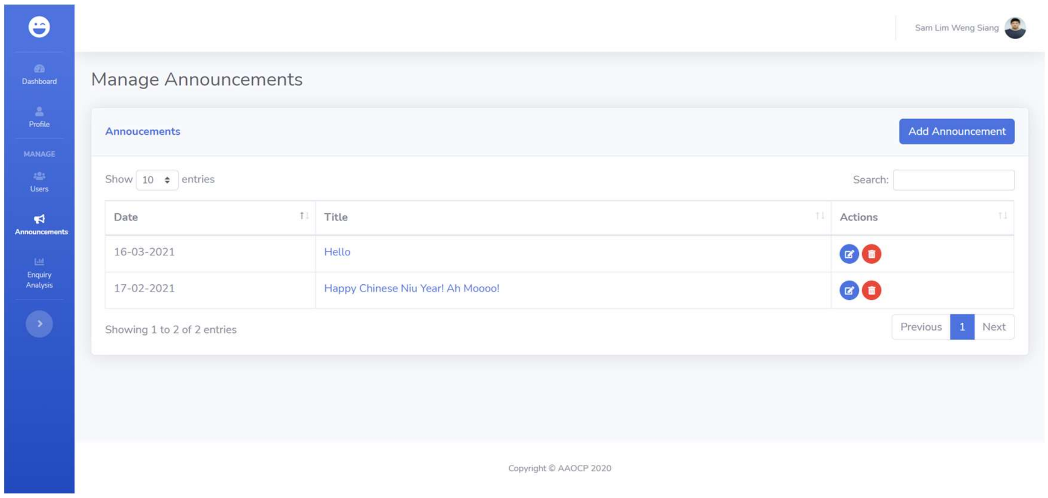
Faculty ID	FICT
Course ID	CS
Assign Advisor	Jessica Tan
Year/Sem	Y3S3
Student ID	17ACB04217
Name	Chiam Jia Ying
IC Number	990523-01-4316
Gender	Female
Address	13, LORONG ALMA 1, TAMAN AMAN JAYA,

Figure 5.13.3 Edit User Page

5.14 Manage Announcements

Administrator can also manage announcements posted in the website via the manage announcement page. Administrator can view all the previous announcements in the table according to the date. Each of the announcement can also be viewed when clicked on the announcement title.

Administrator can also add new announcements by clicking on the add announcement button. The page will be redirected to the add announcement page. The title of the announcement and the announcement details must be filled in, and file or photo can be uploaded if needed. In the details' column, administrator can customize the format of the announcement according to needs, such as font, font colour, font style, text alignment, add tables and much more. If an announcement is only using plain text, the important notice in the announcement might be missed out. Thus, this function is useful, as administrator can highlight the important sentence or keywords in the announcement, the viewers will then notice it clearly. Besides adding announcement, administrator can also edit and delete announcement by clicking on the edit and delete button in each announcement row.



Manage Announcements

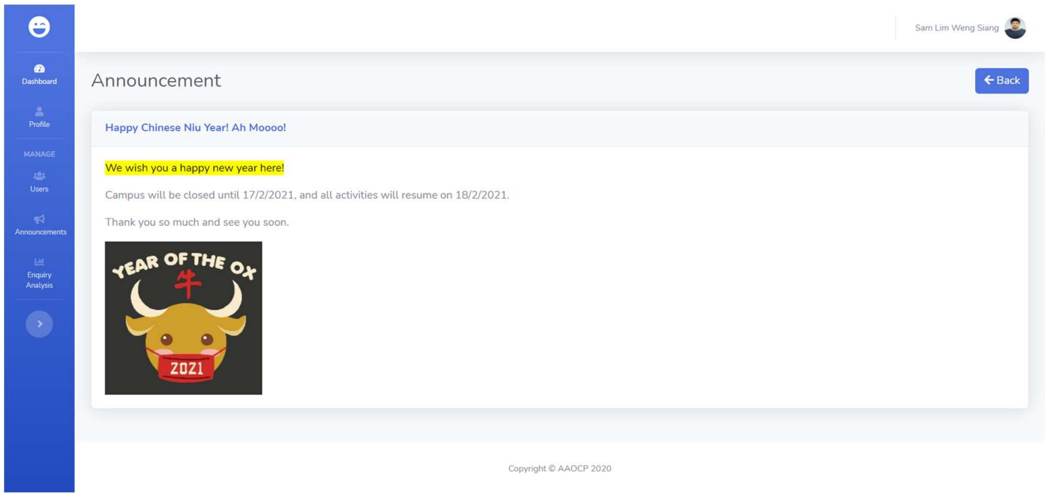
Annoucements

Date	Title	Actions
16-03-2021	Hello	
17-02-2021	Happy Chinese Niu Year! Ah Moooo!	

Showing 1 to 2 of 2 entries

Copyright © AAOCP 2020

Figure 5.14.1 Manage Announcements Page



Announcement

Happy Chinese Niu Year! Ah Moooo!

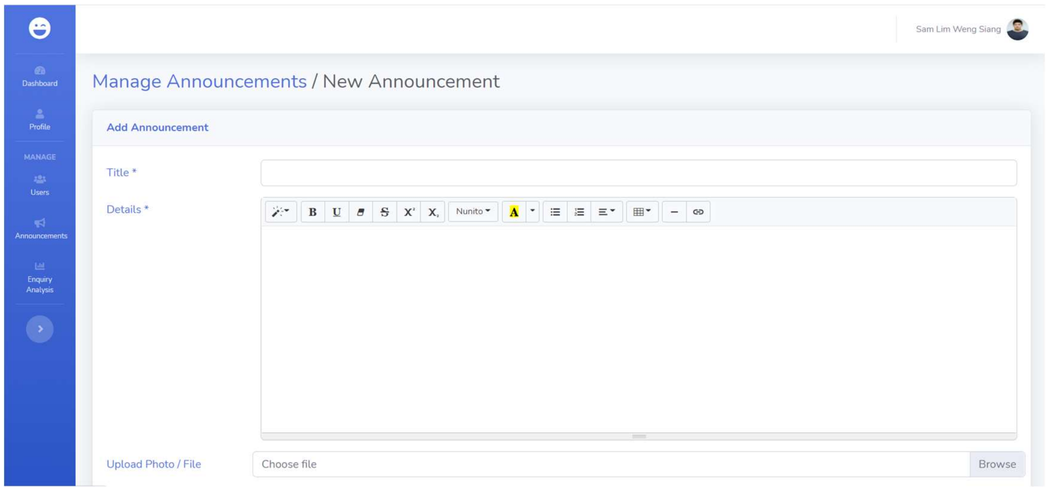
We wish you a happy new year here!

Campus will be closed until 17/2/2021, and all activities will resume on 18/2/2021.

Thank you so much and see you soon.

Copyright © AAOCP 2020

Figure 5.14.2 Announcement Page



Manage Announcements / New Announcement

Add Announcement

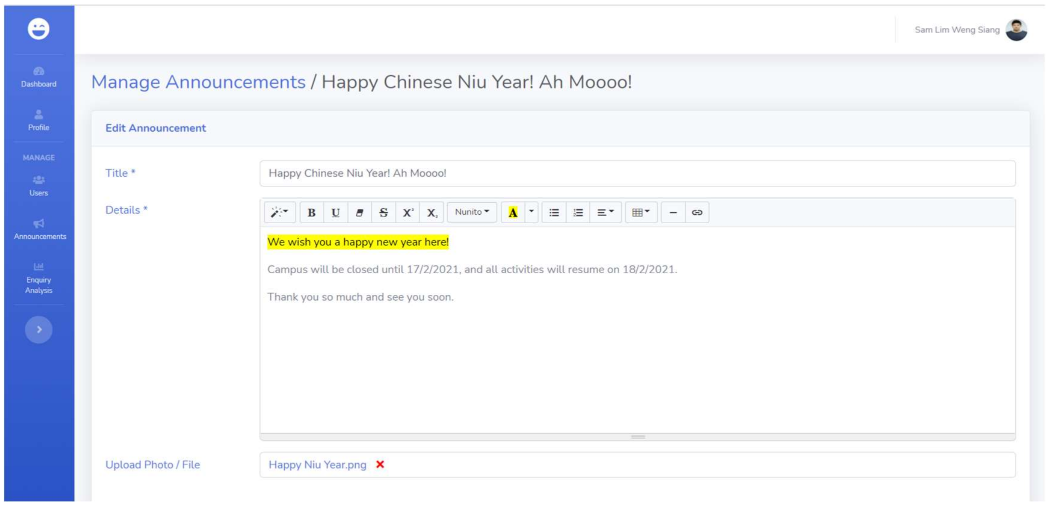
Title *

Details *

Upload Photo / File

Choose file

Figure 5.14.3 Add New Announcement Page



Manage Announcements / Happy Chinese Niu Year! Ah Moooo!

Edit Announcement

Title * Happy Chinese Niu Year! Ah Moooo!

Details * We wish you a happy new year here!

Campus will be closed until 17/2/2021, and all activities will resume on 18/2/2021.

Thank you so much and see you soon.

Upload Photo / File Happy Niu Year.png

Figure 5.14.4 Edit Announcement Page

5.15 User Testing and Feedback

After the project is done, the chatbot is being tested by 10 UTAR students from different faculties. They have tried to use the chatbot and perform testing for a period of time, and then after testing, a feedback form is given to them and the form is filled according to their experience on the chatbot. There are 6 questions in total in the feedback form, which includes the faculty of the user, the user experience of different aspects of the chatbot, and suggestions to improve the chatbot.

Firstly, users need to fill in their faculty in UTAR. Among 10 users, 5 users are from Faculty of Information and Communication Technology (FICT), and another 5 from other faculties, which includes 3 from Faculty of Business and Finance (FBF), 1 from Faculty of Science (FSc), and also 1 from Faculty of Arts and Social Science (FAS). Users from different faculties are needed so that different point of view can be gathered for the feedback of the chatbot. For example, FICT students might focus more on the technical perspective, and other faculty might focus more on the user experience of the chatbot.

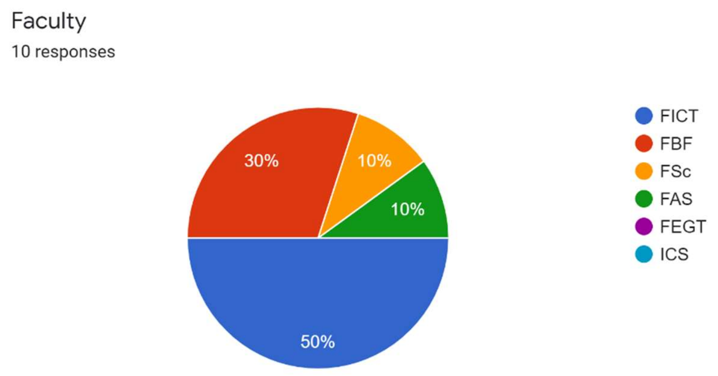


Figure 5.15.1 Faculty of the Users

The second question is asking the users about the understanding of the statements given by the chatbot. This question is asked because if the reply of the chatbot is not well structured, or containing grammar or spelling error, the users will find a hard time to understand what is being replied by the chatbot. The rating in the form is scaled according to 1 (Do Not Understand) to 5 (Clearly Understand). As shown in Figure 5.15.2, half of the users has rated 4 for the statement given by the chatbot, and another half clearly understand what is being replied by the chatbot. Overall, all users understand the statement given by the chatbot and thus there is no big issue about the construction of the sentences and grammar.

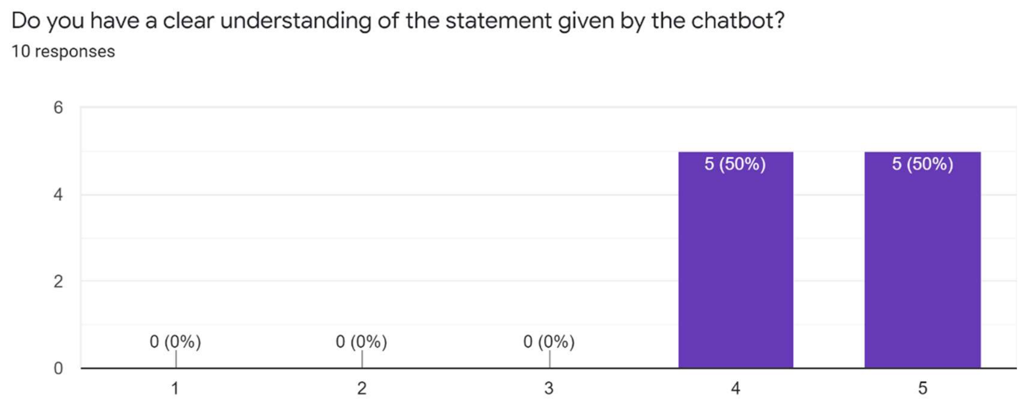


Figure 5.15.2 Rating of the Understanding of Statement Given by the Chatbot

Next, the accuracy of the answer provided by the chatbot is also being asked. This question is asked to know whether the answer provided by the chatbot is what the users need to solve their enquiry. If the accuracy of the answer is low, users will then feel frustrated as the chatbot could not help to solve their enquiry, and they might feel a waste of time. The rating is scaled from 1 (Not Satisfied) to 5 (Very Satisfied). 50% of the users (5 users) have rated 4 for the accuracy, and 3 of the users are very satisfied of the accuracy of the answer. Moreover, 2 users have rated 3 for the accuracy, as they had experienced few inaccurate answers. In short, most users feel that the accuracy of the answer provided can still be improved.

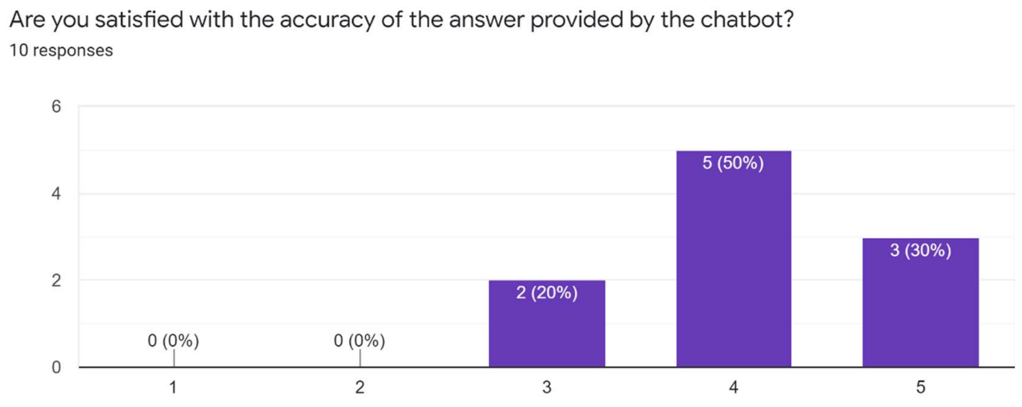


Figure 5.15.3 Rating of the Accuracy of the Answer Provided by the Chatbot

For the fourth question, the ratings about the reply speed of the chatbot is gathered. Reply speed of a chatbot is another crucial factor of a successfully built chatbot. The user experience will be largely reduced if users need to wait for a long time to get a reply after they asked a question. The rating is ascended from 1 (Slow) to 5 (Fast). 6 users had rated 4 for the reply speed, which they feel that the chatbot has a fast reply speed for most of the time, but some questions might need more time, due to more information need to be retrieved from the database. 4 users voted the highest rating for the speed of the chatbot. Overall, all users are satisfied with the speed of the chatbot.

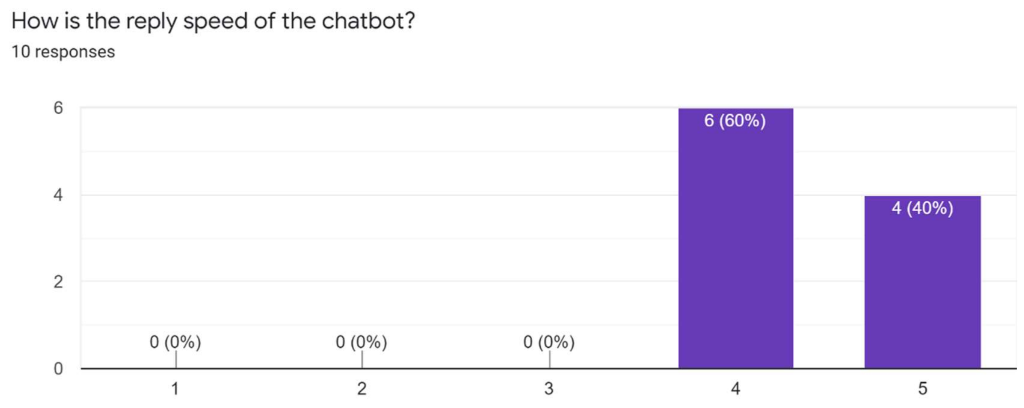


Figure 5.15.4 Ratings of the Reply Speed of the Chatbot

Subsequently, users have rated for their overall experience of using the chatbot. This question is needed as it is the measure of a successfully built chatbot. The users' satisfactory the top priority of this chatbot, as the aim of the chatbot is to serve the users. The rating is scaled from 1 (Not Satisfied) to 5 (Very Satisfied). Most users have rated 4 for their experience after testing the chatbot. Next, 2 users have rated moderate rating (3), as they think that some aspects of the chatbot can still be improved. Only one user is very satisfied about the overall experience of using the chatbot. All in all, the result shows that most users think that there is still room of improvements for the overall experience of the chatbot in future works.

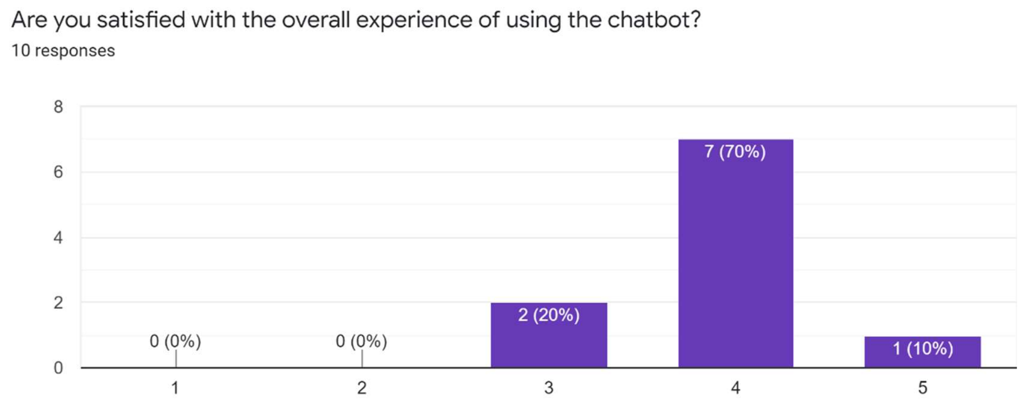
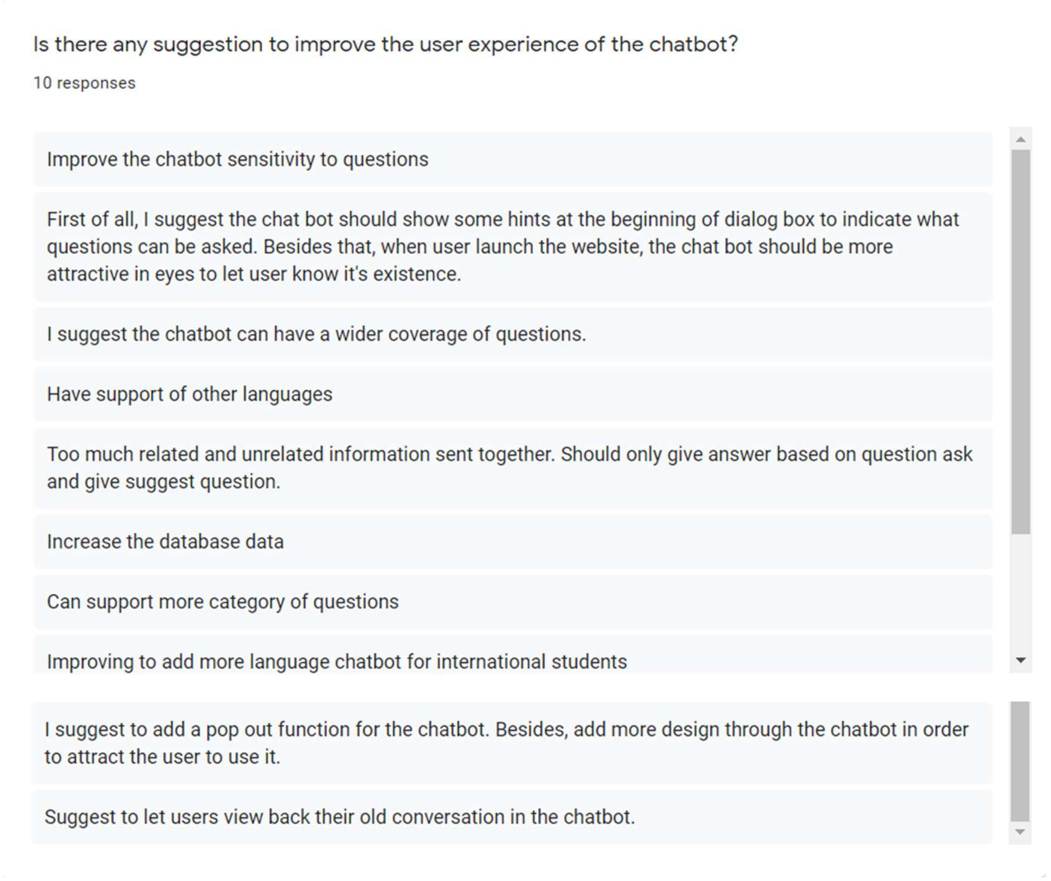


Figure 5.15.5 Rating of the Overall Experience of Using the Chatbot

The last question of this feedback form is an open-ended question. Users is asked to give their suggestions to improve the chatbot in different perspectives. The suggestions can be separated into few categories. Firstly, most users suggest that the chatbot should support more variety of questions. The current chatbot only supports enquiries about the exam and scholarship matters in UTAR. Thus, users suggest that coverage for academic calendar, faculty information, soft skill programs can also be added to the chatbot.

Next, it is the sensitivity of chatbot to questions. Some sentences asked by the user might not be understood by the chatbot and thus user need to rephrase again. For some questions, the chatbot might give more than what the user needs in order to solve their enquiry, and due the answer given is lengthy, user need to look through the long reply to get what they want. There are also users suggest that the chatbot can also support different languages such as Chinese, Malay, or even Japanese. By adding this support, users will feel more comfortable to ask their enquiries as they can use their native language for questioning.

Lastly, there are also suggestions on the functionality of the chatbot. As the chatbot has no hint when user first enter the dashboard, thus a user has suggested to add a notification icon on top right corner of the floating button, and another user suggested to let the chatbot pop up when the user enter the dashboard. Some users also feel unclear about what questions does the chatbot supports as there is no guide or suggestions in the chatbot at first. Thus, a user has recommended to show some hints at the beginning of the dialog to let the user acknowledge what questions is covered by the chatbot. Besides, user cannot view the previous chat with the chatbot after they refresh the page, thus a user has suggested that the previous chat can be stored in the chatbot so that user can have a reference in the future. The user also suggested to improve the design of the chatbot dialog to be more attractive.


Is there any suggestion to improve the user experience of the chatbot?

10 responses

Improve the chatbot sensitivity to questions

First of all, I suggest the chat bot should show some hints at the beginning of dialog box to indicate what questions can be asked. Besides that, when user launch the website, the chat bot should be more attractive in eyes to let user know it's existence.

I suggest the chatbot can have a wider coverage of questions.

Have support of other languages

Too much related and unrelated information sent together. Should only give answer based on question ask and give suggest question.

Increase the database data

Can support more category of questions

Improving to add more language chatbot for international students

I suggest to add a pop out function for the chatbot. Besides, add more design through the chatbot in order to attract the user to use it.

Suggest to let users view back their old conversation in the chatbot.

Figure 5.15.6 Suggestion of Users to Improve the User Experience of the Chatbot

5.16 Summary

All in all, after all of the modules of the project is successfully built, including the login module, profile module, event calendar module, meeting notes module, chatbot module, CGPA and target CGPA calculator module and administration module. Each page of the website is shown in this chapter, and the functionality which can be carried out in the page is explained. Besides, the chatbot built for this project is being tested by 10 UTAR students and their feedback is gathered to ensure that the chatbot has reached the expectation of users. The overall feedback of the users is positive, and many valuable suggestions are given by the users and can be used to improve the chatbot in future works.

CHAPTER 6 CONCLUSION

6.1 Project Review, Discussion and Conclusion

Academic advising is crucial especially for students in universities as it gives important guidance for students in terms of academic, co-curriculum or even in career and life. However, the traditional way of academic advising is not effective as the communication between advisees and advisors will be facing some difficulties, such as ineffective time booking, inconvenience in face-to-face consultation, and duplicated enquiries from advisees. There is also no analysis on the enquiries of the advisees. In addition, the lack of functionalities provided for academic advising in UTAR portal also motivates the development of this project.

This project aims to increase the effectiveness in advisee-advisor communication, lighten the workload of advisor, and obtain the data of frequently asked questions by advisee. Therefore, in this project, various modules are planned in order to achieve the objectives. RAD methodology is used in development of this project. According to the first objective which is to ease the booking of advisor's time slot for consultation, the login module, profile module, event calendar module, meeting notes module and CGPA and target CGPA module is done. Advisee and advisor can now book consultation meetings effectively by using this platform. Both advisee and advisor can add events to their personal calendar to organize their schedule. Besides, both advisee and advisor can view the profile of the opposite side and their available timeslot, and book a time for meeting. The system will also automatically inform advisee and advisor if the booking overlaps an existing event in their own schedule. Besides, according to the second and third objective, chatbot module and administration module is also developed. A chatbot is developed to let advisees ask their enquiries, and all the questions asked are stored in database, and analysis of the questions are shown in the enquiry analysis page of advisor and administrator, and also the dashboard of administrator. The administrator is also eligible to manage users and announcements in the platform.

The problem encountered during this project is during the development of the login module. Firebase Admin SDK is used for authentication as it provides the functionalities needed by the administrator without the need of the credentials of the user. However, the documentation for the login session cookies is limited. Thus, quite some time is spent on finding a workable way to implement the login session cookie in the program developed, but the problem is now solved. Besides, there are also some problems faced during developing the chatbot module. Before starting to develop the chatbot, web scraping is needed to obtain the data from UTAR official website and use it as the reply of the chatbot. As each of the page in UTAR official website are having some difference in their HTML format, and the text and image in the official website must be scrapped concurrently, thus the time used to identify the code to scrap the data from the website is quite long. After scrapping the data, rearranging of the data is needed as the data scrapped might not be arranged well and the format of the data scrapped might contain some symbol which is not suitable to be stored into database directly, and thus the filtering of data also takes quite some time to be done.

6.2 Novelties and Contributions

By applying this project for advisee-advisor communication, it is definitely time saving for both the advisee and advisor. The efficiency of having the information of advisor's free timeslot and timeslot booking for consultation will increase. This is beneficial for advisee and advisor. Advisor no longer needs to scroll through the email one by one to notice the request of advisees, which they might sometimes missed out due to large numbers of emails received every day. On the other hand, bookings for meeting can be made by advisor and advisee directly in the website after viewing the timetable for available timeslot of the opposite side. Thus, the process of booking a consultation meeting is simplified and more time efficient.

The chatbot in this project can be used to help the advisors in replying non decision-making answers and reduce the advisors' job task as they do not need to answer same questions all over again. Advisees can also receive feedback from the chatbot anytime as the chatbot neither need to rest nor having a leave day, and thus satisfactory of advisees increase as their inquiry can be solved efficiently. The chatbot is also very useful to the institution itself as it can be used to analyse the data of questions posed by advisees. The institution can then acknowledge which statements they did not inform the students well and thus they can improve in the statements provided to the students.

The CGPA and target CGPA calculator can help the advisor to predict on the grades the advisee needs to get for the semester in order to achieve the academic goals of the advisee, or giving recommendations to the advisee on building a study plan to improve their results, or even rearranging the program structure to suit the advisee if the advisee is unable to cope with the original program structure arranged by the institution. The advisee can also use the CGPA and target CGPA calculator to motivate themselves to put more effort in accomplishing their study goals.

6.3 Future Work

There are some enhancements can be made in this project. Firstly, the current event calendar can only add events that is occurring once. The platform is currently not supporting users to add repeated events automatically, and users need to add the events one by one. Thus, in future work, this recurrence event function can be added so that users can add events such as weekly classes, or weekly activities of society in the system by just adding it once.

Furthermore, this website can support notification function in the future, which the website currently lacks of. Notification is quite important for most users in order to be notified instantly when there are new updates, such as new announcement posted, new bookings or when the event planned is near to the time planned. Notification can also be added to the chat function when there is new message from advisee or advisor. By having this function, users no longer need to be worrying of missing out important updates and notices. The notification can be made using Firebase Cloud Messaging, and the notification can be in the form of an email, or a web push notification if the user grants the permission to let the website send push notifications to his device.

Moreover, the chatbot in the system can also be improved. The chatbot can be trained to support different languages such as Chinese, Malay and Japanese. Thus, users can use their native language to ask their enquiry and receive the answer in language that they are comfortable with. By supporting multi-languages, it can prevent users having uncertainty about the answer and reduce mistake of interpreting the answer given by the chatbot.

BIBLIOGRAPHY

CollegePlannerPro n.d., 'Watch a Demo of CollegePlannerPro's Software' (video file). Available from: <https://go.collegeplannerpro.com/demo-video>. [21 August 2020].

FlightPath 2013, 'FlightPath Demonstration – Advisor's View' (video file). Available from: < https://www.youtube.com/watch?v=V3aiSeni0lc>. [21 August 2020].

FlightPath 2013, 'FlightPath Demonstration – Student's View' (video file). Available from: < https://www.youtube.com/watch?v=6v0jGpj0hds>. [21 August 2020].

Henderson LK & Goodridge W 2015, 'AdviseMe: An Intelligent Web-Based Application for Academic Advising', *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 6, no. 8, pp. 233-243. Available from: The Science and Information Organization [29 August 2020].

Kerk, RX 2011, *Student Planning and Advisory System (SPAS).* Final Year Project, Universiti Tunku Abdul Rahman. Available from: Universiti Tunku Abdul Rahman Library website <https://elibrary.utar.edu.my/>. [18 August 2020].

Kuhn, TL 2008, 'Historical foundations of academic advising', in *Academic advising: a comprehensive handbook*, Jossey-Bass, San Francisco, pp. 3-16.

Lardinois, F 2015, *Microsoft Launches Visual Studio Code, A Free Cross-Platform Code Editor for OS X, Linux And Windows.* Available from: < https://techcrunch.com/2015/04/29/microsoft-shocks-the-world-with-visual-studio-code-a-free-code-editor-for-os-x-linux-and-windows/ >. [21 August 2020].

O'Carroll, B 2020. Available from: <https://codebots.com/site/img/DT-276_%20(1).png>. [21 August 2020].

O’Carroll, B 2020, *What is Rapid Application Development (RAD).* Available from: <https://codebots.com/app-development/what-is-rapid-application-development-rad>. [21 August 2020].

Rasa n.d., *Components.* Available from: <https://rasa.com/docs/rasa/components>. [11 April 2021].

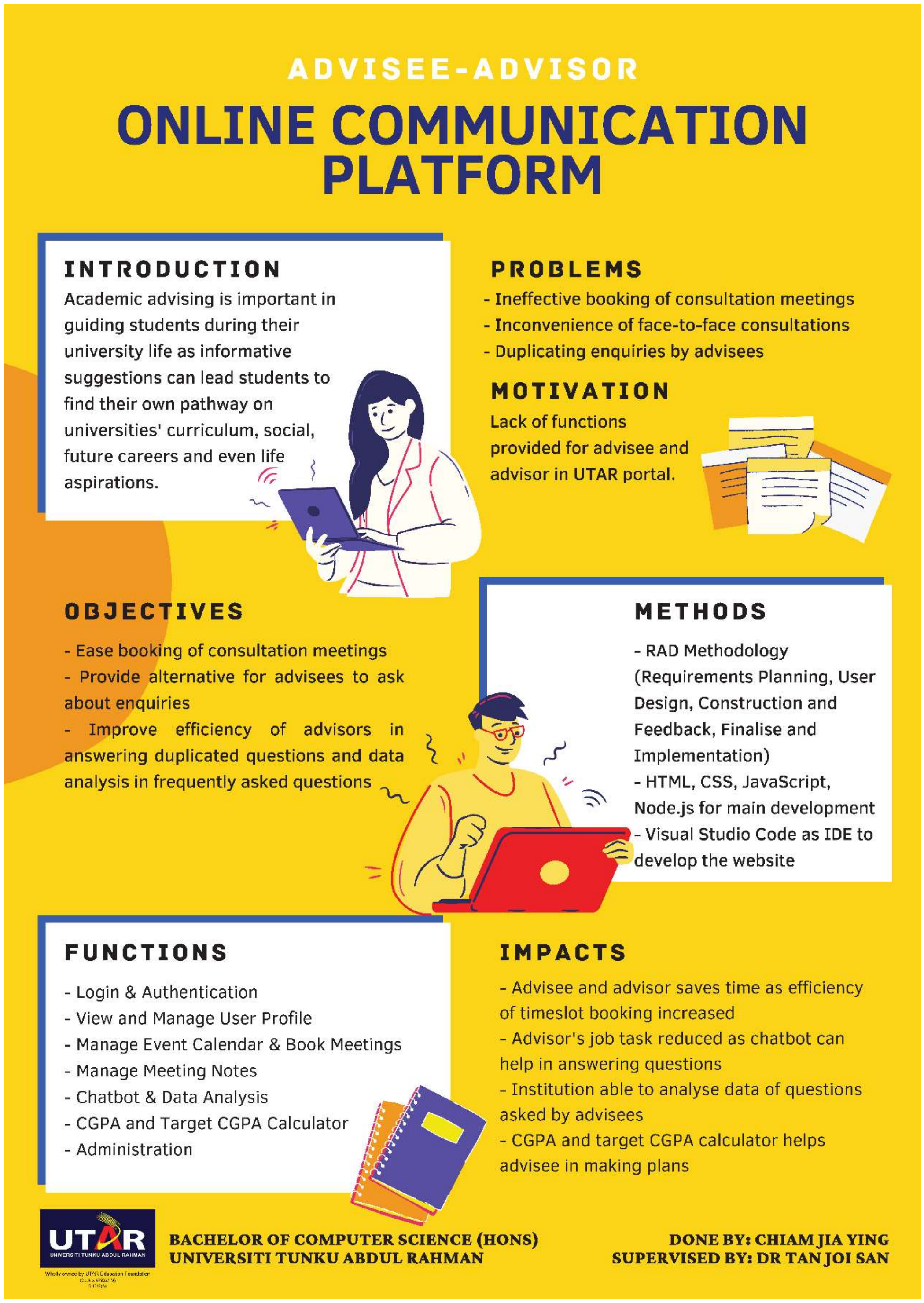
Ruffalo Noel Levitz 2019, *2019 National Student Satisfaction Report*. Available from: <RuffaloNL.com/Satisfaction>. [21 August 2020].

Singh, A 2019, *What is Rapid Application Development (RAD).* Available from: < https://blog.capterra.com/what-is-rapid-application-development/>. [21 August 2020].

Universiti Tunku Abdul Rahman 2020, *Introduction.* Available from: <https://utar.edu.my/Introduction.php>. [21 August 2020].

World Health Organization 2020, *WHO Director-General's opening remarks at the media briefing on COVID-19—11 March 2020*. Available from: <https://www.who.int/dg/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020>. [21 August 2020].

POSTER



ADVISEE-ADVISOR

ONLINE COMMUNICATION PLATFORM

INTRODUCTION

Academic advising is important in guiding students during their university life as informative suggestions can lead students to find their own pathway on universities' curriculum, social, future careers and even life aspirations.

PROBLEMS

- Ineffective booking of consultation meetings
- Inconvenience of face-to-face consultations
- Duplicating enquiries by advisees

MOTIVATION

Lack of functions provided for advisee and advisor in UTAR portal.

OBJECTIVES

- Ease booking of consultation meetings
- Provide alternative for advisees to ask about enquiries
- Improve efficiency of advisors in answering duplicated questions and data analysis in frequently asked questions

METHODS

- RAD Methodology (Requirements Planning, User Design, Construction and Feedback, Finalise and Implementation)
- HTML, CSS, JavaScript, Node.js for main development
- Visual Studio Code as IDE to develop the website

FUNCTIONS

- Login & Authentication
- View and Manage User Profile
- Manage Event Calendar & Book Meetings
- Manage Meeting Notes
- Chatbot & Data Analysis
- CGPA and Target CGPA Calculator
- Administration

IMPACTS

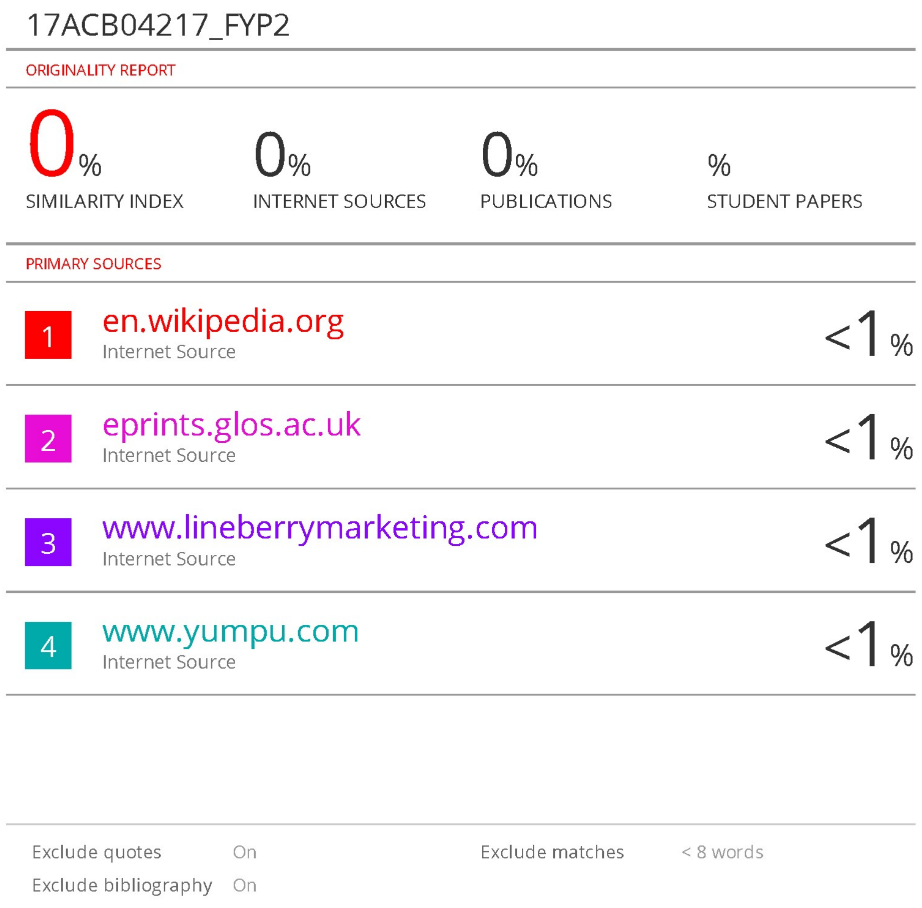
- Advisee and advisor saves time as efficiency of timeslot booking increased
- Advisor's job task reduced as chatbot can help in answering questions
- Institution able to analyse data of questions asked by advisees
- CGPA and target CGPA calculator helps advisee in making plans

UTAR UNIVERSITI TUNKU ABDUL RAHMAN

BACHELOR OF COMPUTER SCIENCE (HONS)
UNIVERSITI TUNKU ABDUL RAHMAN

DONE BY: CHIAM JIA YING
SUPERVISED BY: DR TAN JOI SAN

PLAGIARISM CHECK RESULT



17ACB04217_FYP2

ORIGINALITY REPORT

0%	0%	0%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	en.wikipedia.org Internet Source	<1 %
2	eprints.glos.ac.uk Internet Source	<1 %
3	www.lineberrymarketing.com Internet Source	<1 %
4	www.yumpu.com Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 8 words
Exclude bibliography	On		

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 & COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	CHIAM JIA YING
ID Number(s)	17ACB04217
Programme / Course	BACHELOR OF COMPUTER SCIENCE (HONOURS)
Title of Final Year Project	ADVISEE-ADVISOR ONLINE COMMUNICATION PLATFORM

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceed the limits approved by UTAR)**
Overall similarity index: _0_% **Similarity by source** Internet Sources: _0_ % Publications: _0_% 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
Name: Tan Joi San

Date: 15th April 2021

Signature of Co-Supervisor
Name: __________

Date: __________

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	17ACB04217
Student Name	CHIAM JIA YING
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.
✓	Front Cover
✓	Signed Report Status Declaration Form
✓	Title Page
✓	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)
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*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. [illegible] ______________________ (Signature of Student) Date: 15th April 2021	Supervisor verification. Report with incorrect format can get 5 marks (1 grade) reduction. [illegible] ______________________ (Signature of Supervisor) Date: 15th April 2021