The Hashemite University, Zarqa, Jordan
Faculty of Prince Al-Hussein Bin Abdallah II For Information Technology
Computer Science and Applications Department

HUBOT

A project submitted
in partial fulfillment of the requirements for the
B.Sc. Degree in Computer Science and Applications

By

Mohammad Haitham Fathi Salah (2035083)
Mohammad Abdulnnasir Abdallah Khattab (2011447)
Tariq Mohammad Adel Alkurdi (1437177)

Supervised by

Dr. Ashraf Hamdan Rashid Aljammal

Committee Member Names

First Committee Member's First Name Middle Initial Last Name

Second Committee Member's First Name Middle Initial Last Name

Month Year

5/2023

CERTIFICATE

It is hereby certified that the project titled <***HUBOT***>, submitted by undersigned, in partial fulfillment of the award of the degree of “Bachelor in Computer Science and Applications” embodies original work done by them under my supervision.

All the analysis, design and system development have been accomplished by the undersigned. Moreover, this project has not been submitted to any other college or university.

Mohammad Haitham Fathi Salah (2035083)

Mohammad Abdulnnasir Abdallah Khattab (2011447)

Tariq Mohammad Adel Alkurdi (1437177)

ABSTRACT

The HUBOT mobile app is a comprehensive solution designed to enhance the campus experience for students by providing various features and functionalities. This project aimed to address the challenges faced by students in managing their academic schedules, accessing campus resources, and obtaining relevant information. The investigation involved the development of the app using Flutter for the frontend and Java for the backend, with Firebase as the real-time database and REST APIs for communication.

HUBOT will have key features such as a GPA calculator, campus map with indoor tracking capabilities, FAQ section, comprehensive chatbot, smart recommendation system, and a student day manager. These features were selected to be in the first version of HUBOT during to they are essential features. these features designed to streamline academic tasks, improve navigation within the campus, facilitate communication with a responsive chatbot, provide personalized recommendations, and offer a convenient platform for managing daily schedules.

HUBOT future additions and enhancement are countless, and we are not trying just to solve student problems and issues, we are trying to build a wholl environment that will add many new things to our campus and open up to new and other doors.

Its benefit will not stop in student it will raise the university services and in future HUBOT will contain many users such as employees, instructors or just and foreign visitor.

In conclusion we are not just trying to build a bot, we are establishing for a new era of university life, HUBOT future addition are countless…

ACKNOWLEDGEMENTS

This page is optional. It is where you may put your personal word of thanks to anyone who has helped you in your work.

TABLE OF CONTENTS

ABBREVIATIONS

→ I.e. is the abbreviation for id Est and means “in other words.”

LIST OF FIGURES

Figure 23: Caption for Figure 23 via *References-> part of GBA calculator code*

LIST OF TABLES

CHAPTER 1:INTRODUCTION

1.1 OVERVIEW

THIS CHAPTER INTRODUCES THE HUBOT PROJECT, WHICH IS A CHATBOT APPLICATION AIMED AT PROVIDING ASSISTANCE TO STUDENTS IN VARIOUS ASPECTS OF THEIR UNICERSITY DAYS. IT PROVIDES A GENERAL OVERVIEW OF THE PROJECT AND ITS MAIN FEATURES.

1.2 PROJECT MOTIVATION

THE MOTIVATION BEHIND DEVELOPING THE HUBOT PROJECT WAS TO PROVIDE STUDENTS WITH A CONVENIENT AND EASY-TO-USE TOOL TO HELP THEM IN EFFICENT WAY, GUIED THEM AND MANAGING THEIR ACADEMIC TASKS. THE PROJECT AIMS TO ADDRESS THE CHALLENGES THAT STUDENTS FACE IN MANAGING THEIR COURSEWORK AND ACADEMIC SCHEDULES. BY PROVIDING A USER-FRIENDLY CHATBOT INTERFACE.

THE PROJECT IS IMPORTANT BECAUSE IT HAS THE POTENTIAL TO ENHANCE STUDENTS' ACADEMIC EXPERIENCES BY PROVIDING THEM WITH AN ACCESSIBLE AND PERSONALIZED TOOL FOR MANAGING THEIR ACADEMIC WORK AND PROVIDE PERSONALIZED GUIDNESS OR HELP IN VARRIOUS ASPECTS. ADDITIONALLY, THE PROJECT AIMS TO INCREASE STUDENT ENGAGEMENT AND RETENTION BY OFFERING A PLATFORM THAT ALLOWS FOR MORE EFFICIENT AND EFFECTIVE ACADEMIC SUPPORT.

THE NEW IDEA PROPOSED BY THIS PROJECT IS THE DEVELOPMENT OF AN INTELLIGENT CHATBOT THAT CAN PROVIDE PERSONALIZED ACADEMIC ASSISTANCE TO STUDENTS.

1.3 PROBLEM STATEMENT

THE HUBOT PROJECT ADDRESSES THE ISSUE OF STUDENTS STRUGGLING ON FINDING THE INFORMATION THEY NEED AND TO MANAGE THEIR ACADEMIC WORKLOAD AND SCHEDULES AND MANY MORE. MANY STUDENTS FACE CHALLENGES IN KEEPING UP WITH COURSEWORK, ASSIGNMENTS, AND DEADLINES, LEADING TO POOR ACADEMIC PERFORMANCE AND INCREASED STRESS LEVELS. THE PROJECT AIMS TO PROVIDE A SOLUTION TO THESE PROBLEMS BY OFFERING A CHATBOT THAT CAN PROVIDE RELEVANT AND PERSONALIZED SUPPORT, HELP,INFORMATION AND GOOD ACADEMIC EXPERIANCE.

1.4 PROJECT AIM AND OBJECTIVES

THE OVERALL AIM OF THE HUBOT PROJECT IS TO DEVELOP AN INTELLIGENT CHATBOT THAT CAN ASSIST STUDENTS IN MANAGING THEIR ACADEMIC WORKLOAD, SCHEDULES AND EVERY THING RELATED TO THEIR UNIVERSITY LIFE. THE SPECIFIC OBJECTIVES OF THE PROJECT INCLUDE:

- DEVELOPING A CHATBOT THAT CAN UNDERSTAND AND RESPOND TO STUDENT QUERIES AND REQUESTS.
- PROVIDING PERSONALIZED ACADEMIC ASSISTANCE TO STUDENTS, INCLUDING REMINDERS FOR ASSIGNMENTS AND DEADLINES, STUDY TIPS, AND RECOMMENDATIONS.
- OFFERING A USER-FRIENDLY AND ACCESSIBLE INTERFACE FOR STUDENTS TO MANAGE THEIR ACADEMIC TASKS.
- IMPROVING STUDENT ENGAGEMENT AND RETENTION BY OFFERING A TOOL THAT ENHANCES THEIR ACADEMIC EXPERIENCE.

- CREATE A MAP FOR OUR UNIVERSITY CAMPUS THAT CAN NAVIGATE EVEN INSIDE BUILDINGS.

- ANSWER STUDENT FREQUENT QUIESTION.

1.5 PROJECT LIMITATIONS

THE HUBOT PROJECT HAS SEVERAL LIMITATIONS AND PARAMETERS THAT NEED TO BE CONSIDERED. FIRSTLY, THE CHATBOT IS DESIGNED TO PROVIDE ASSISTANCE ONLY TO STUDENTS AND MAY NOT BE SUITABLE FOR OTHER USER GROUPS *(TILL THIS POINT)*. ADDITIONALLY, THE PROJECT IS LIMITED TO PROVIDING UNIVERSITY SUPPORT AND DOES NOT ADDRESS OTHER AREAS OF STUDENT LIFE OUT SIDE THE HASHEMITE UNIVERSITY CAMPUS.

1.6 PROJECT EXPECTED OUTPUT

THE DESIRED OUTPUT OF THE HUBOT PROJECT IS AN INTELLIGENT CHATBOT THAT CAN PROVIDE PERSONALIZED ACADEMIC SUPPORT TO STUDENTS. THE CHATBOT SHOULD BE USER-FRIENDLY, ACCESSIBLE, AND CAPABLE OF UNDERSTANDING AND RESPONDING TO STUDENT QUERIES AND REQUESTS.

1.7 PROJECT SCHEDULE

THE PROJECT SCHEDULE INCLUDES SEVERAL MILESTONES, ACTIVITIES, AND DELIVERABLES, WITH INTENDED START AND FINISH DATES. THE PROJECT IS EXPECTED TO BE COMPLETED WITHIN SIX MONTHS, WITH THE FOLLOWING MAJOR MILESTONES:

- RESEARCH AND PLANNING PHASE (MONTH 1)

- DEVELOPMENT AND TESTING PHASE (MONTHS 2-4)

- INTEGRATION AND DEPLOYMENT PHASE (MONTHS 5-6)

1.8 REPORT ORGANIZATION

THE REMAINDER OF THIS REPORT IS ORGANIZED AS FOLLOWS. CHAPTER 2 PROVIDES AN OVERVIEW OF THE RELATED WORK IN CHATBOT DEVELOPMENT AND ACADEMIC SUPPORT TOOLS. CHAPTER 3 DESCRIBES THE METHODOLOGY USED IN DEVELOPING THE HUBOT PROJECT. CHAPTER 4 PRESENTS THE DESIGN AND ARCHITECTURE COMPONENTS OF THE CHATBOT. CHAPTER 5 DISCUSSES THE IMPLEMENTATION AND TESTING OF THE CHATBOT. CHAPTER 6 PRESENTS THE FUTURE WORK AND CONCLUDES THE REPORT.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This literature review will examine the available literature in the field of chatbots and their applications, with a focus on the development of Hubot. The review aims to provide a comprehensive understanding of the current state of the art in this field and how Hubot fits into it.

But to be considered that the idea of making AI bots for universities doesn't exist in Jordan and not widely common in the world, and the idea of making such intelligent bot to be the student friend and assistant it's nearly doesn't exist.

2.2 Existing Systems

We have done a lot of researches trying to find a tool that is nearly similar to our tool but we didn't find, in Jordan none of the current universities have any Chabot app, in the world it's not widely exist in universities some of the big universities have done chatbots as web pages and a very shy number of universities have chatbots as mobile application and they are only accessed from their university country.

Another issue that all the chatbots are connected with their university database and systems so they provide a range of services we are not aiming to provide.

Even if maybe there is some common features but we are varying in the way and the content of the features we have, as an example let's take campus map feature as an example, many have this feature but nearly none of them do track inside buildings and so on to the other features we have.

- **Tools discussed:**

1- CSE Peer Helper

The CSE Peer Helper chatbot developed by the University of Minnesota is built using a number of programming languages and technologies. The primary programming language used to develop the chatbot is Java, along with several related libraries and frameworks, including the Microsoft Bot Framework and Apache OpenNLP for natural language processing

2- CRIMSON CONCRIEGE

The Crimson Concierge chatbot, developed by the University of Alabama, was built using the Amazon Lex service and implemented in the Node.js programminglanguage. Amazon Lex is a service provided by Amazon Web Services (AWS) that enables developers to build conversational interfaces for applications using natural language understanding and automatic speech recognition.

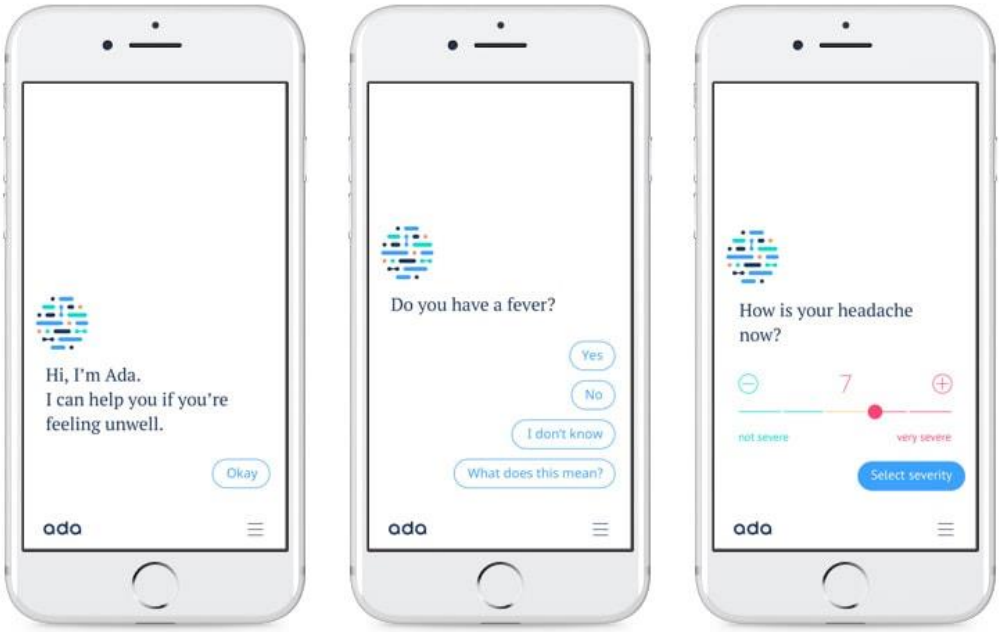


Figure 1: Caption for Figure 1 via *References-> example of chatbot UI*

3- POUNCE

Georgia State University's chatbot, Pounce, was built using the Microsoft BotFramework and implemented in the C# programming language. The Bot Framework provides a platform for building, testing, and deploying chatbots,and C# is a commonly used language within the framework. Additionally, Pounce utilizes natural language processing (NLP) to understand user input.

4- ASK BLUE

Ask Blue, the chatbot developed by the University of Michigan, was built using the Oracle Digital Assistant platform, which allows developers to buildconversational interfaces using natural language processing and machine learning technologies. The programming languages used to create the chatbot are not publicly available, as the Oracle Digital Assistant platform abstracts the underlying implementation details.

Tool\ Features	recommendation	Technical Support	Assignment Help	House and Dining	Student day manager	Personalized Assistance	Financial Aid	map
CSE Helper	T	T	T	X	X	X	X	X
Crimson	X	X	X	T	X	T	X	X
Pounce	T	X	X	X	X	X	T	T
Ask Blue	X	X	X	T	X	T	X	X
HUBOT	T	X	X	X	T	T	X	T

Table 1: Caption for Table 1 via *References-> comparision table*

- Green T denoted to existing feature.
- Red X denoted to excluded feature.

2.3 Overall Problems of Existing Systems

Despite the popularity of chatbots, there are still several challenges that exist in their development and deployment. One of the main challenges is natural language processing, which is critical for understanding user input and generating appropriate responses.

While existing systems have made significant contributions to the field, they are not without their limitations and challenges. Through our literature review, we have identified several common issues encountered in these systems. These include limited functionality, lack of personalized recommendations and complex user interfaces.

2.4 Overall Solution Approach

Based on our analysis of the existing systems and the identified problems, we have developed an overall solution approach for HUBOT. Our approach involves the utilization of Flutter as the frontend framework and Java as the backend programming language. To overcome the limitations of dependency on university servers and databases, we have opted to integrate Firebase as a real-time database and REST API provider.

By leveraging Firebase, we can ensure the availability of real-time data updates, seamless integration with the app's frontend, and the ability to store user-specific information securely. This approach enables us to create a mobile app that functions independently of university infrastructure while still providing valuable assistance to students.

CHAPTER 3: REQUIREMENT ANALYSIS

3.1 Stakeholders

In the case of HUBOT, we have identified the following primary stakeholders:

Students: The primary users of the HUBOT system who will benefit from its features and functionalities in accessing information, receiving assistance, and navigating the campus effectively.

Administration: Administrative personnel who oversee the overall management.

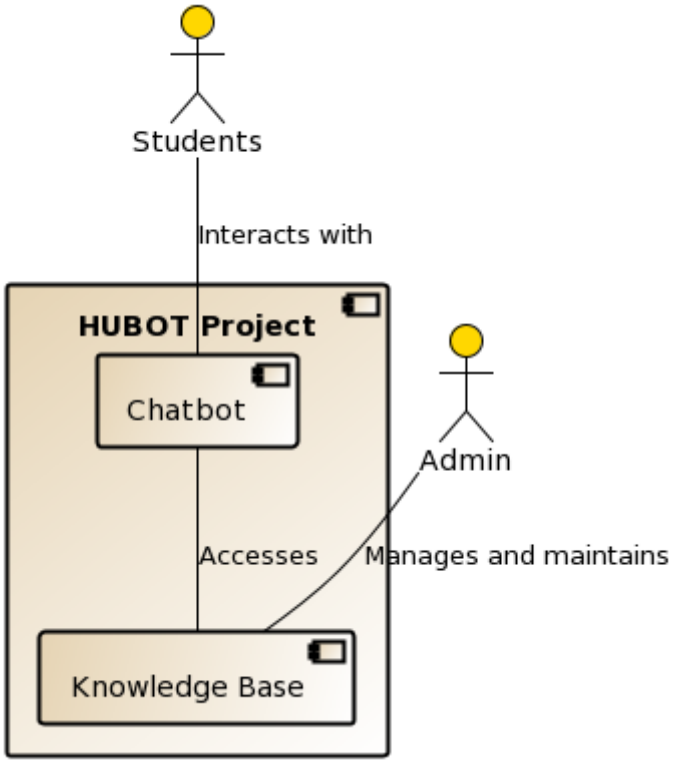


Figure 2: Caption for Figure 2 via *References-> stakeHolders*

The diagram depicts the relationships within the HUBOT project, showcasing how different individuals and components interact. Here's a brief explanation:

Stakeholders:

Students: They interact with the system by engaging with the Chatbot component, seeking assistance and accessing information.

Admin: As the system administrator, I am responsible for managing and maintaining the Knowledge Base component.

Components:

HUBOT Project: This encompasses the overall project, including the Chatbot and Knowledge Base components.

Chatbot: It serves as the interactive interface through which students communicate with the system, asking questions and receiving responses.

Knowledge Base: It acts as a repository of knowledge, which the Chatbot accesses to provide students with answers and assistance.

The diagram visually represents the interaction between students and the Chatbot component, which, in turn, relies on the Knowledge Base for relevant information.

As the admin, I oversee the management and maintenance of the Knowledge Base to ensure its accuracy and relevance.

Overall, the diagram effectively represents the relationships between stakeholders and components within the HUBOT project, emphasizing the flow of information and interactions between them.

3.2 Use Case Diagram

3.2.1 User

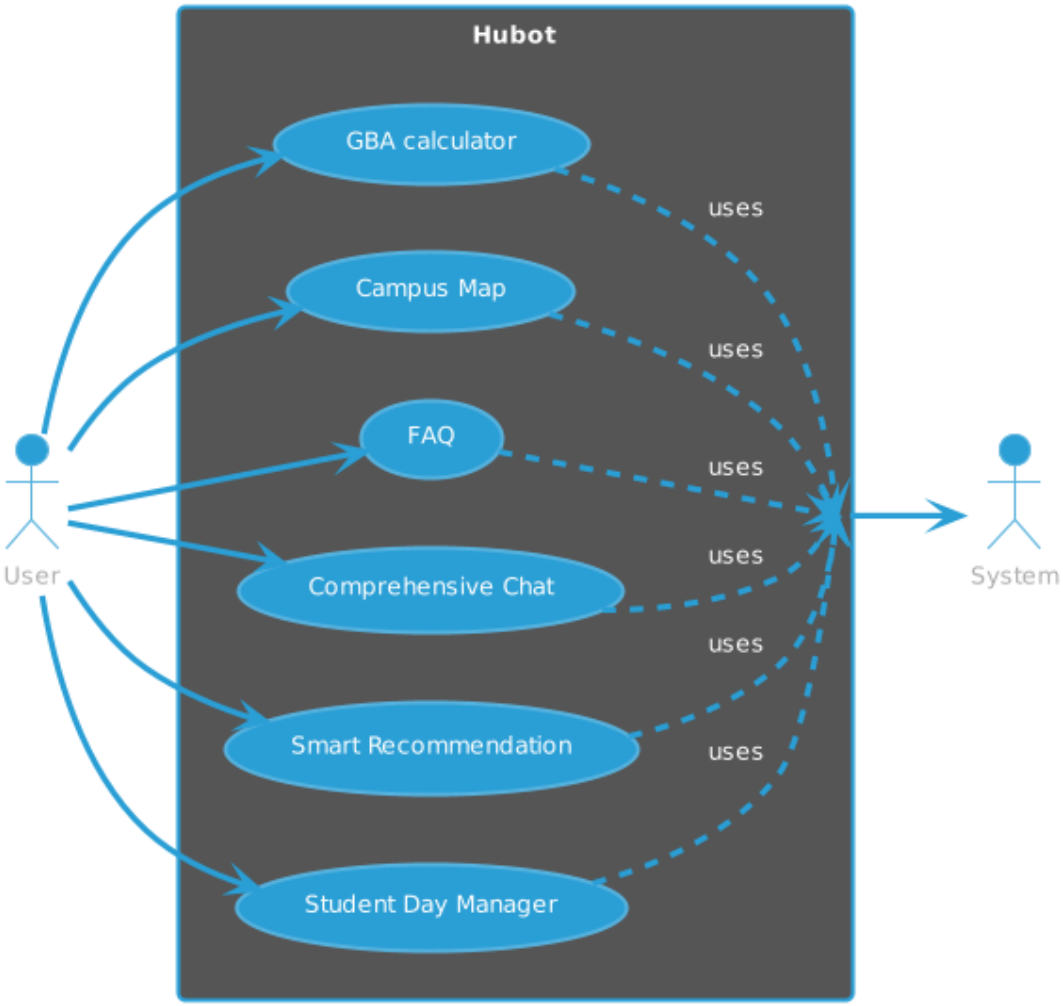


Figure 3: Caption for Figure 3 via *References->user use case diagram*

The use case diagram depicts the different functionalities of the HUBOT application and how they interact with the **users** and the system. The actors in the diagram are the user and the system, while the HUBOT application is represented as a rectangle containing six use cases.

The first use case is the GPA calculator, which allows users to calculate their Grade Point Average (GPA) easily. The second use case is the Campus Map, which provides users with a comprehensive map of the university campus, including information about buildings, facilities, and services. The third use case is the FAQ, which provides users with answers to frequently asked questions about the university and its services.

The fourth use case is the Comprehensive Chat, which allows users to ask the **HUBOT** and get answers to their questions in a comprehensive manner. The fifth use case is the Smart Recommendation, which provides personalized recommendations to users based on their preferences, interests, and "past interactions" "in future" with the application. The sixth use case is the Student Day Manager, which helps students manage their daily activities and schedules more efficiently.

The use case diagram also shows that the user interacts directly with each of the six use cases, while the Hubot application is responsible for managing and coordinating the interactions between the user and the system. The diagram demonstrates that the Hubot application is an essential tool for students, providing them with a range of features and functionalities to support their academic and personal lives.

3.2.2 Admin

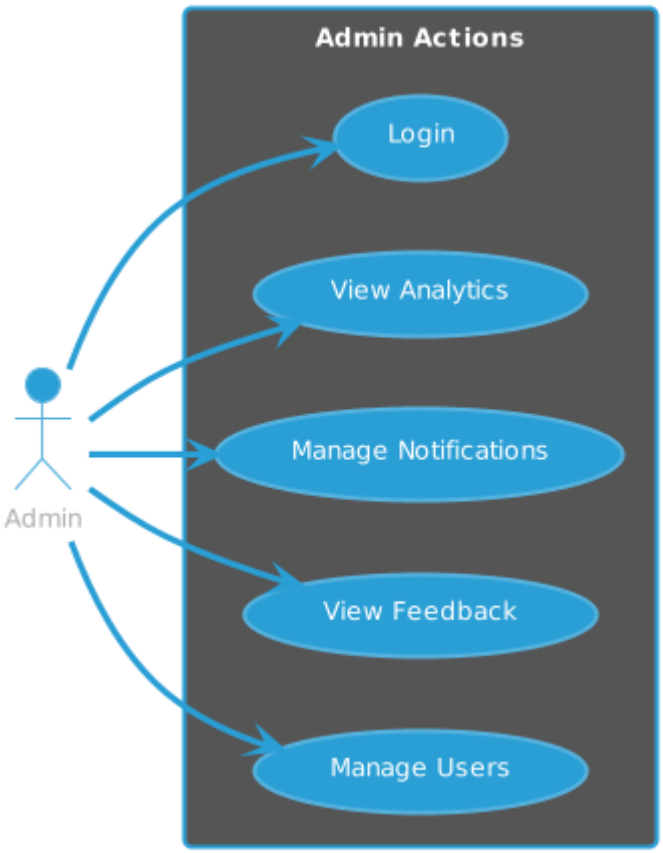


Figure 4: Caption for Figure 4 via *References-> admin use case diagram*

The above picture represents a Use Case Diagram for the **Admin** Actions in a Hubot mobile application. The diagram shows an actor named "Admin" who has access to five different actions that they can perform within the application. The actions are represented as use cases within a rectangle titled "Admin Actions".

The first use case is "Login", which implies that the Admin needs to authenticate themselves in order to access the application. The next use case is "View Analytics", which would allow the Admin to see data and statistics about the application usage and other important metrics.

The third use case is "Manage Notifications", which suggests that the Admin has the ability to create, edit or delete notifications that are displayed within the application. This use case could also include the ability to send push notifications to the user's device.

The fourth use case is "View Feedback", which would allow the Admin to view feedback from users of the application. This use case could include the ability to filter feedback by category or time period.

The final use case is "Manage Users", which would allow the Admin to manage user accounts within the application. This use case could include the ability to create, edit, or delete user accounts, as well as reset user passwords or view user activity logs.

Overall, this Use Case Diagram provides a clear and concise representation of the key actions that an Admin can perform within the Hubot mobile application. It shows the main features and functionalities of the Admin side of the application and how it can be used to manage and monitor the application's users and data.

3.3 non-Functional User Requirements

When designing a chatbot like Hubot, it's important to consider not just the functional requirements (i.e. what the chatbot should be able to do), but also the non-functional requirements that will ensure the chatbot is safe, secure, usable, and easy to maintain over time. Here are the non-functional user requirements for the Hubot project:

3.3.1 Execution Qualities

- Safety: Hubot should not be able to cause harm to users or their data. It should be designed with appropriate safeguards and access controls to prevent any malicious actions.
- Security: Hubot should be designed with strong security measures to protect users' privacy and data. It should use encryption and secure communication protocols to protect sensitive information.
- Usability: Hubot should be easy to use and understand for users of all technical levels. It should have a clear and simple interface, with intuitive commands and responses.

3.3.2 Evolution Qualities

- Testability: Hubot should be designed with testability in mind, so that it can be easily tested and debugged during development and maintenance. This includes having clear error messages and logging features.
- Maintainability: Hubot should be designed with maintainability in mind, so that it can be easily updated and modified as needed over time. This includes using modular, reusable code and following best practices for code organization and documentation.
- Extensibility: Hubot should be designed with extensibility in mind, so that it can be easily extended with new functionality as needed. This includes having a clear plugin architecture and API for developers to build upon.
- Scalability: Hubot should be designed with scalability in mind, so that it can handle a growing number of users and requests over time.

By considering these non-functional user requirements, we can ensure that Hubot is not just a functional chatbot, but a reliable and sustainable tool for users to interact with over time.

CHAPTER 4:ARCHITECTURE AND DESIGN

4.1 Software (System) Architecture

4.1.1 Logical view

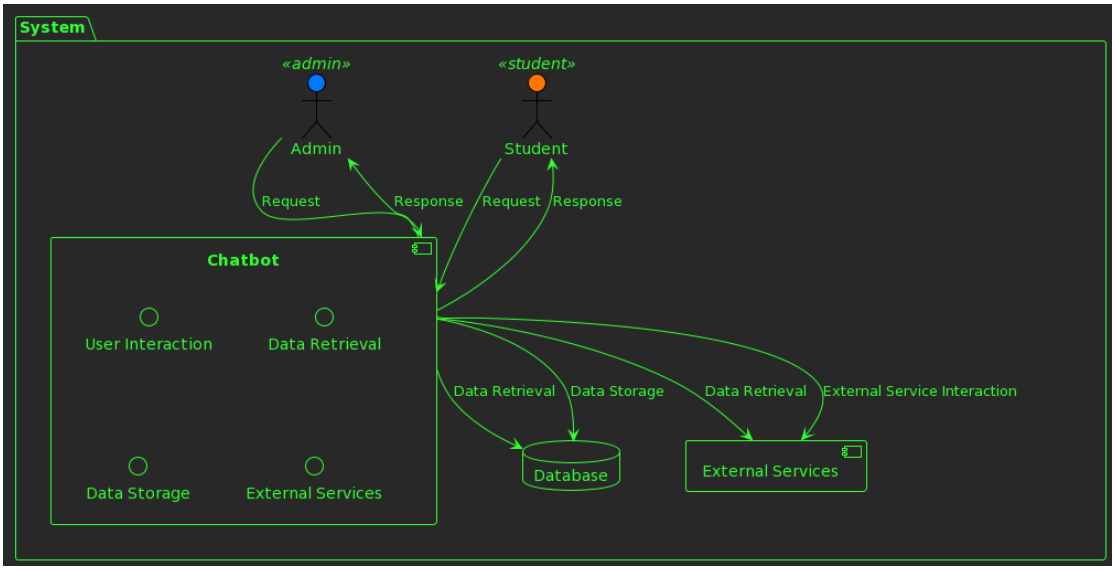


Figure 5: Caption for Figure 5 via *References-> logical view diagram*

4.1.2 Details of each component in a separate section.

4.1.2.1 Chatbot:

The Chatbot component serves as the core component of the system. It includes the following interfaces:

User Interaction: This interface enables the Chatbot to interact with users, specifically students. It handles user inputs, processes requests, and provides responses.

Data Retrieval: This interface allows the Chatbot to retrieve relevant data from the database. It can fetch information such as student records, course details, assignments, and deadlines.

Data Storage: This interface enables the Chatbot to store data into the database. It can save user preferences, progress, or any other relevant information for future reference.

External Services: This interface facilitates communication with external services or APIs. It allows the Chatbot to retrieve additional data or perform actions beyond the system's scope.

4.1.2.2 Database:

The Database component is responsible for storing and managing data related to the system. It stores information such as student records, course details, assignments, and other relevant data. The HUBOT component interacts with the Database component through the Data Retrieval and Data Storage interfaces to fetch and store data as needed.

4.1.2.3 External Services:

The External Services component represents external services or APIs that the system may utilize. It can include services for accessing additional educational resources, retrieving real-time information, or integrating with other systems. The Chatbot component interacts with the External Services component through the Data Retrieval and External Service Interaction interfaces to retrieve relevant data or perform specific actions.

Each component plays a crucial role in the system's overall functionality, enabling the Chatbot to provide user interaction, retrieve and store data, and interact with external services when necessary.

4.2 Software design

4.2.1 User UML sequence/communication diagram

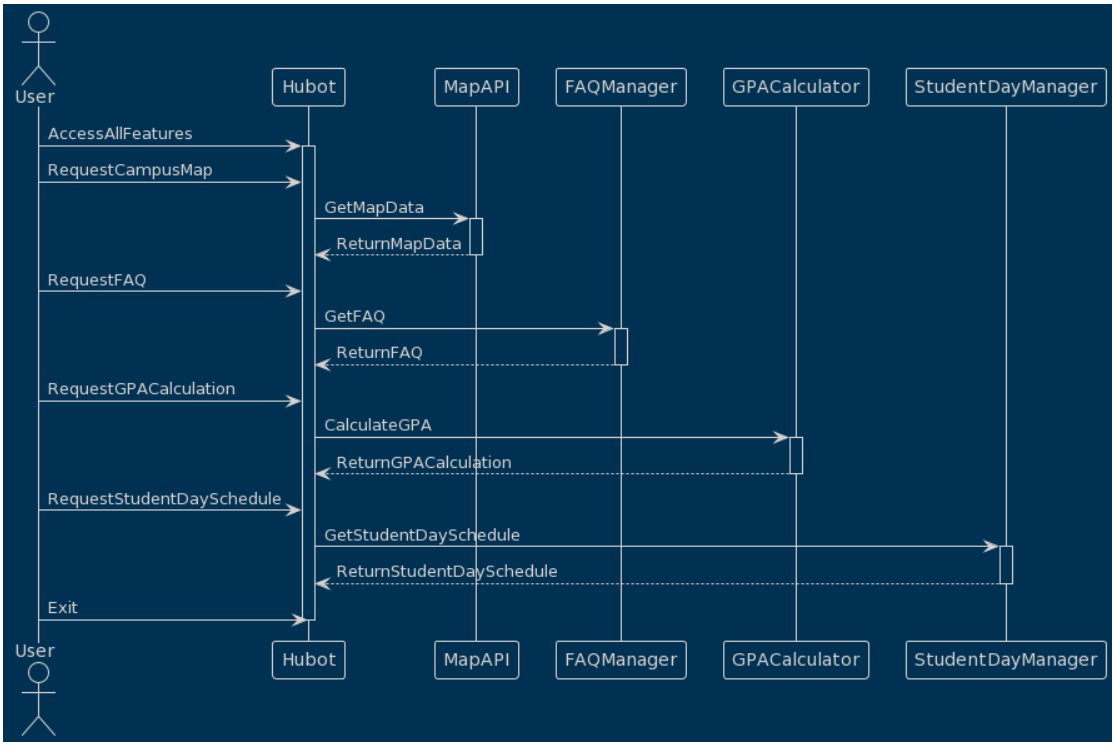


Figure 6: Caption for Figure 6 via *References-> User uml sequence diagram*

1. **Access All Features (sync):** The user synchronously accesses all the available features of the Hubot system, as it requires immediate interaction and response from Hubot without any delay.
2. **Request Campus Map (sync):** The user synchronously requests the campus map, and Hubot interacts with the Map API synchronously. This ensures that the user receives the map data in a timely manner, as it's a direct and immediate response to the user's request.
3. **Request GPA Calculator (sync):** The user synchronously requests to calculate their GPA, and Hubot processes the calculation synchronously. This allows for immediate feedback and results to be presented to the user without any delay.

4. **Request Comprehensive Chat (async):** The user asynchronously initiates a comprehensive chat with Hubot, and Hubot interacts with the Chatbot API asynchronously. This is because the Chatbot API may take some time to process the user's messages and generate appropriate responses. Asynchronous communication ensures that the user can continue the conversation while waiting for responses from the Chatbot API.

5. **Exit (sync):** The user synchronously exits the Hubot system, ensuring that the exit operation is completed before terminating the interaction.

You can notice that the comprehensive chat is indicated as asynchronous, meaning that there might be a delay in responses as Hubot communicates with the Chatbot API in the background.

4.2.2 Admin UML sequence/communication diagram

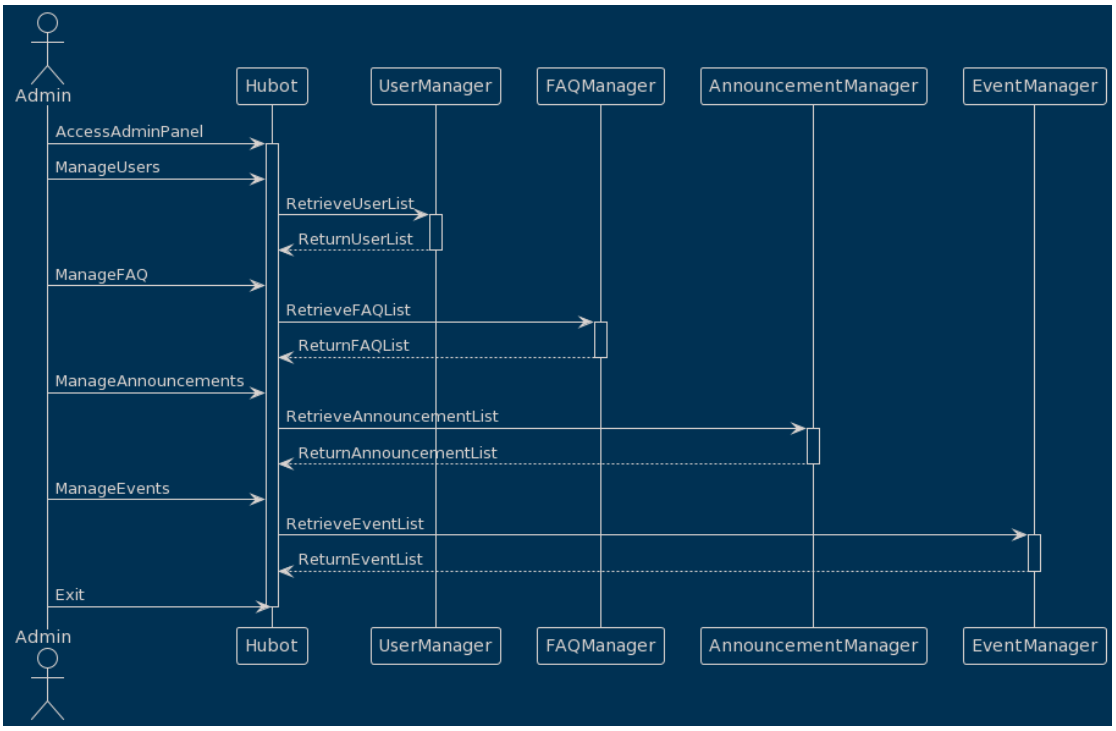


Figure 7: Caption for Figure 7 via *References-> Admin uml sequence diagram*

1. **Access Admin Panel (sync):** The admin synchronously accesses the admin panel to perform administrative tasks. Immediate access and interaction are required as the admin needs to have real-time control and visibility over the system's management.
2. **Manage Users (sync):** The admin synchronously manages users and interacts with the User Manager to retrieve and manipulate user data. Synchronous communication is employed to ensure immediate control and manipulation of user information, allowing the admin to perform user-related tasks in real-time.
3. **Manage FAQ (sync):** The admin synchronously manages the frequently asked questions (FAQ) feature, including adding, editing, and deleting FAQs. Synchronous communication is used to ensure that changes to the FAQ are immediately reflected and available for users. This allows for real-time management of the FAQ, ensuring that users always have up-to-date information.

4. **Manage Announcements (sync):** The admin synchronously manages announcements and interacts with the Announcement Manager to retrieve the list of announcements. Synchronous communication enables immediate access to the announcement list, ensuring that the admin can promptly view, edit, or delete announcements as needed.
5. **Manage Events (sync):** The admin synchronously manages events and interacts with the Event Manager to retrieve the list of events. Synchronous communication allows for real-time access to the event list, enabling the admin to perform event-related tasks without delay.
6. **Exit (sync):** The admin synchronously exits the Hubot system, ensuring that any ongoing operations are completed before terminating the interaction.

The reason use of synchronous communication in these scenarios ensures immediate control, visibility, and real-time updates for the admin, enabling efficient management of various system aspects.

4.2.3 Class diagram

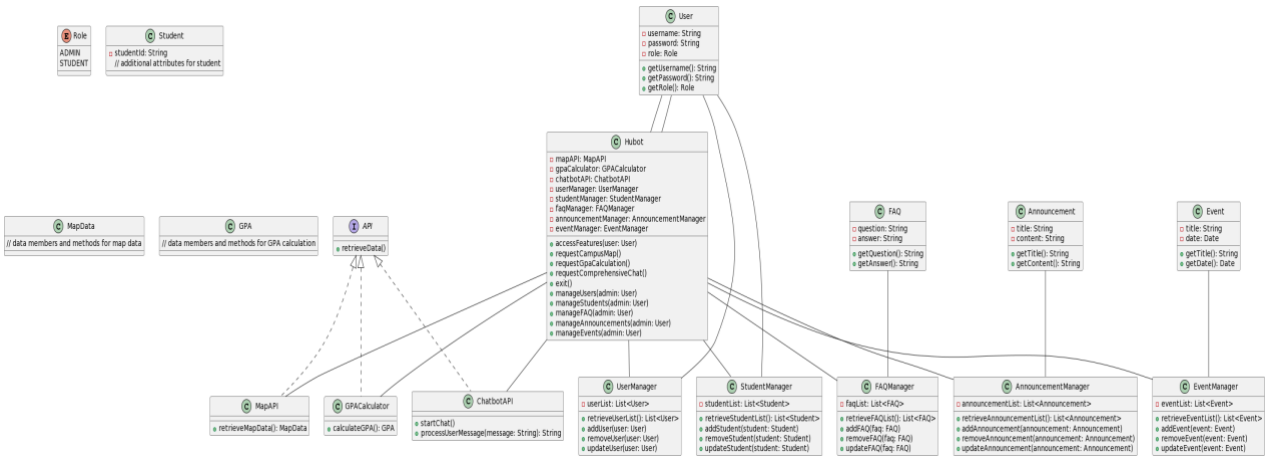


Figure 8: Caption for Figure 8 via *References-> class diagram*

4.2.4 ER-Diagram

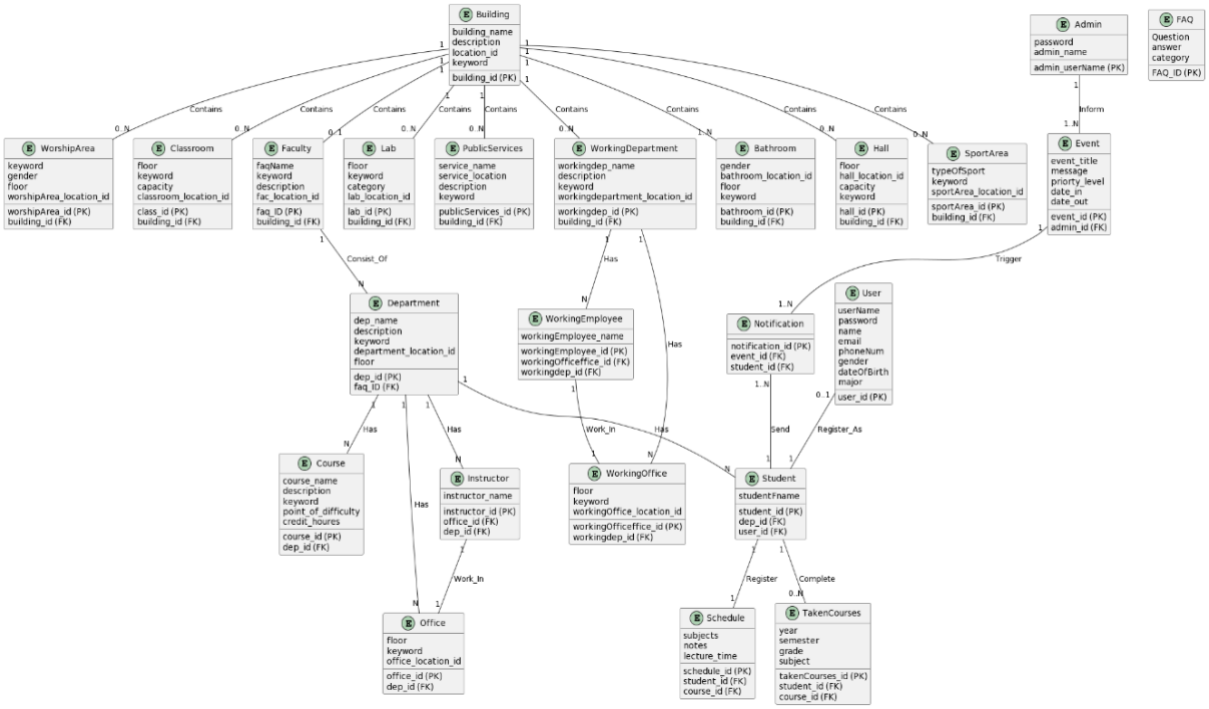


Figure 9: Caption for Figure 9 via *References-> ER-diagram*

4.3 User interface design (prototype)

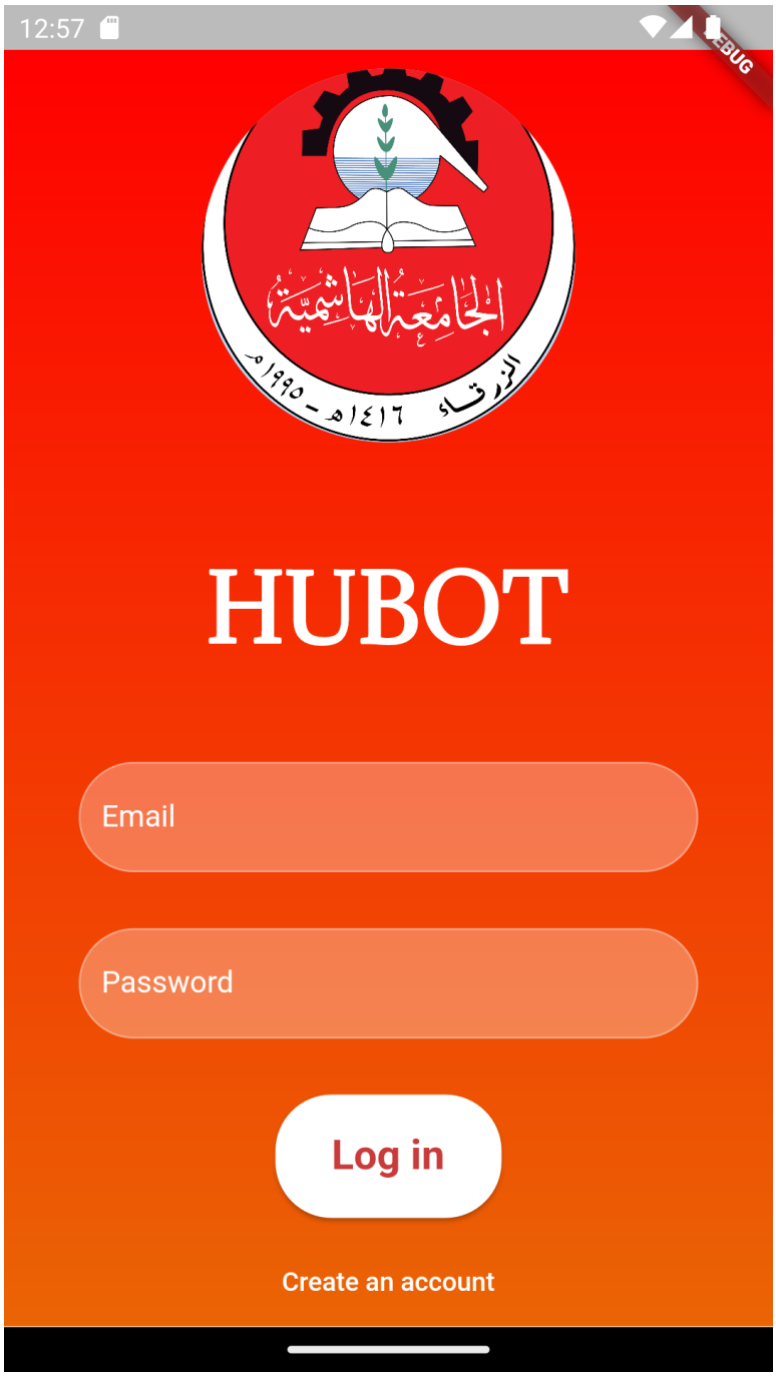


Figure 10: Caption for Figure 10 via *References->log in page*

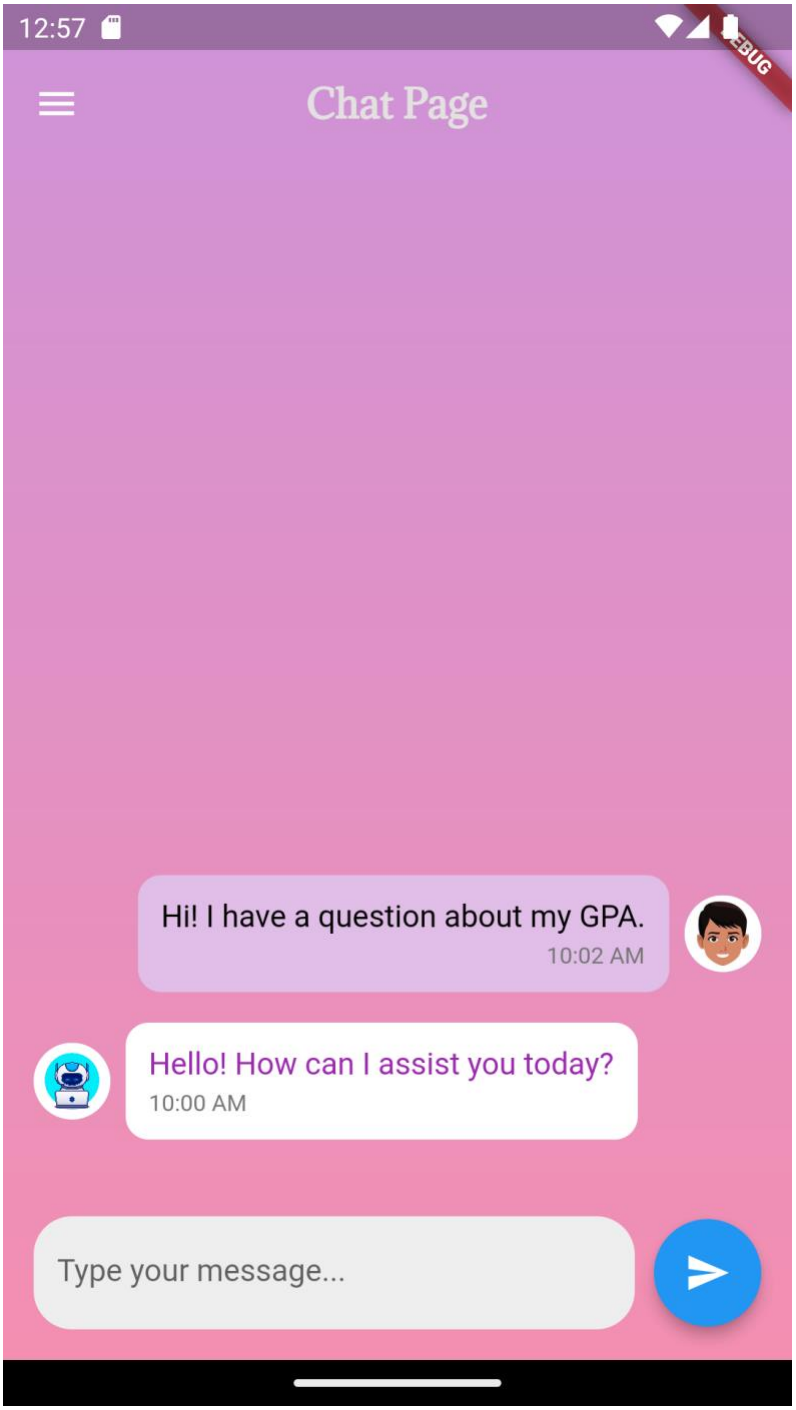


Figure 11: Caption for Figure 11 via *References-> chatbot page*

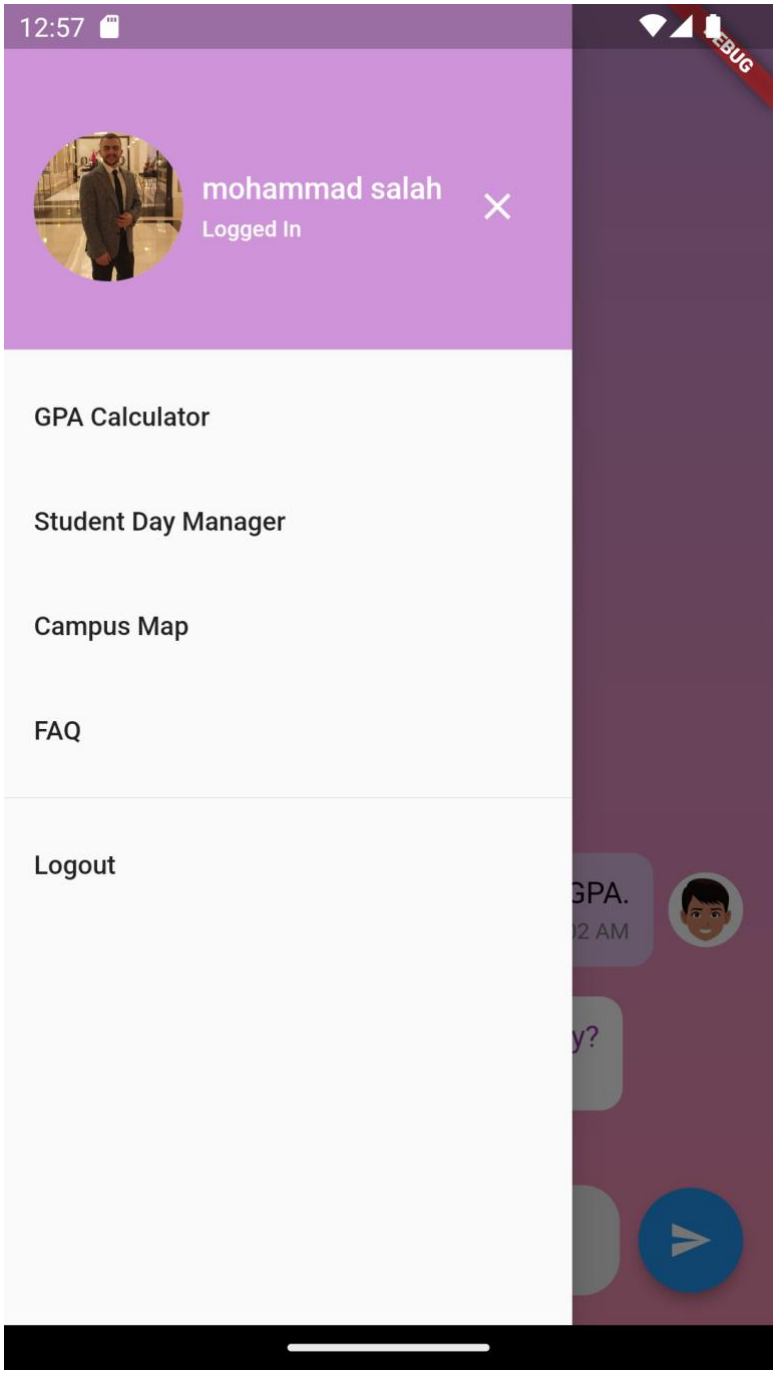


Figure 12: Caption for Figure 12 via *References-> features slide menu*

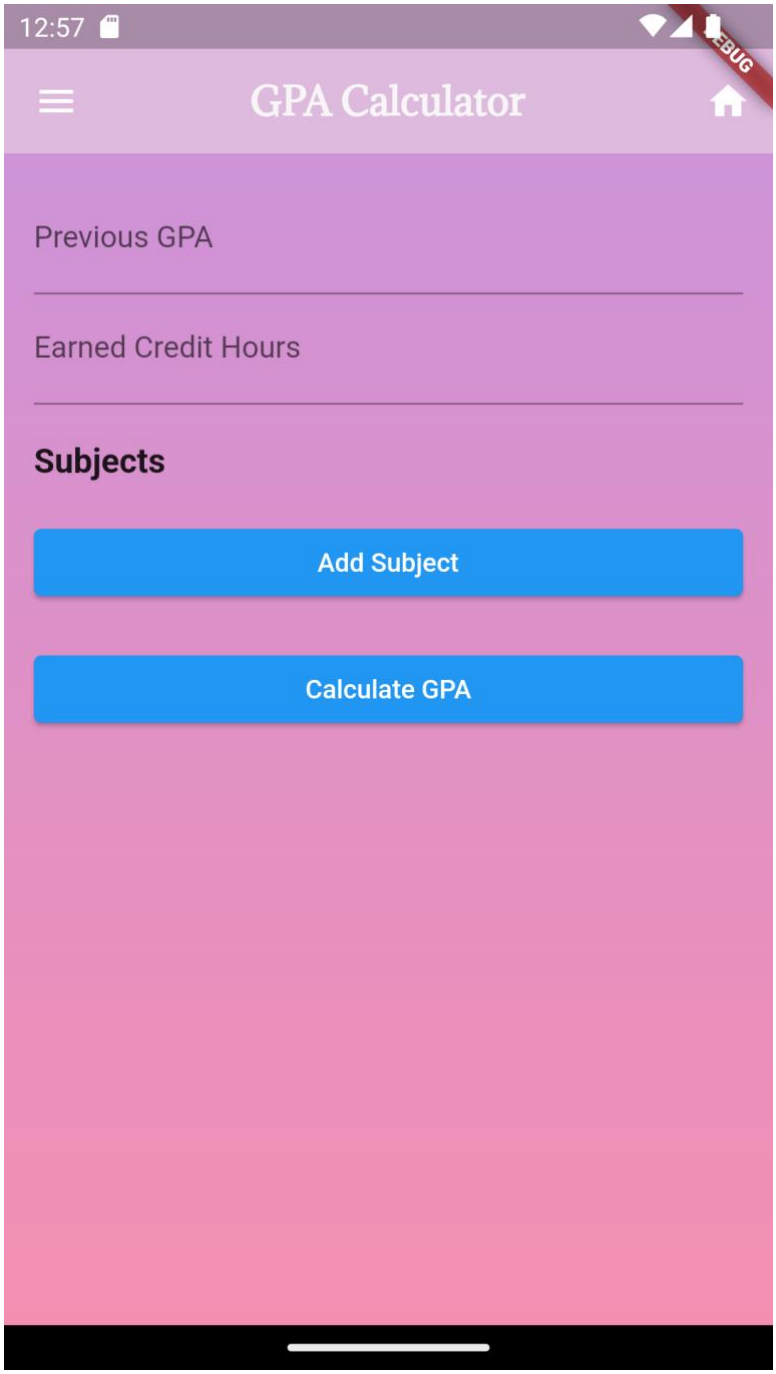


Figure 13: Caption for Figure 13 via *References->GPA calculator diagram*

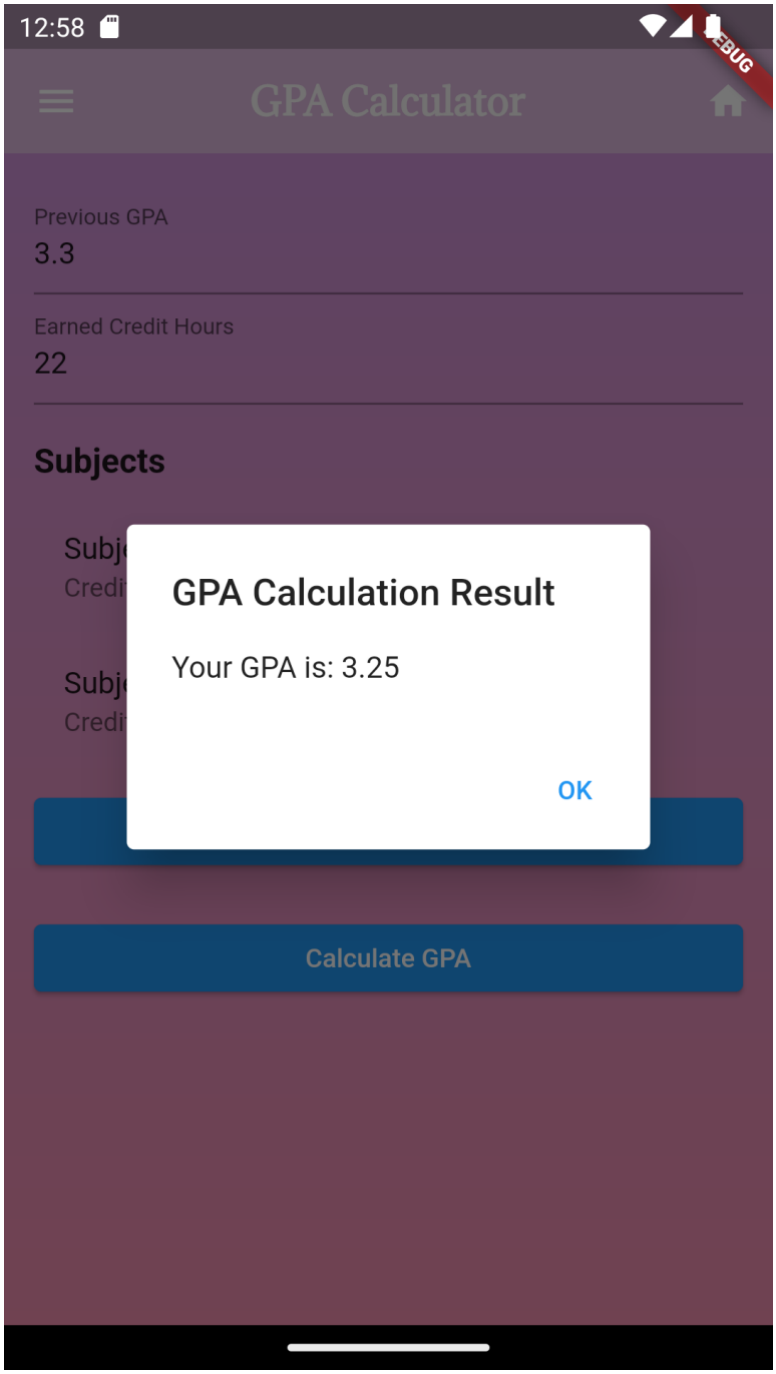


Figure 14: Caption for Figure 14 via *References-> GPA calculator example*

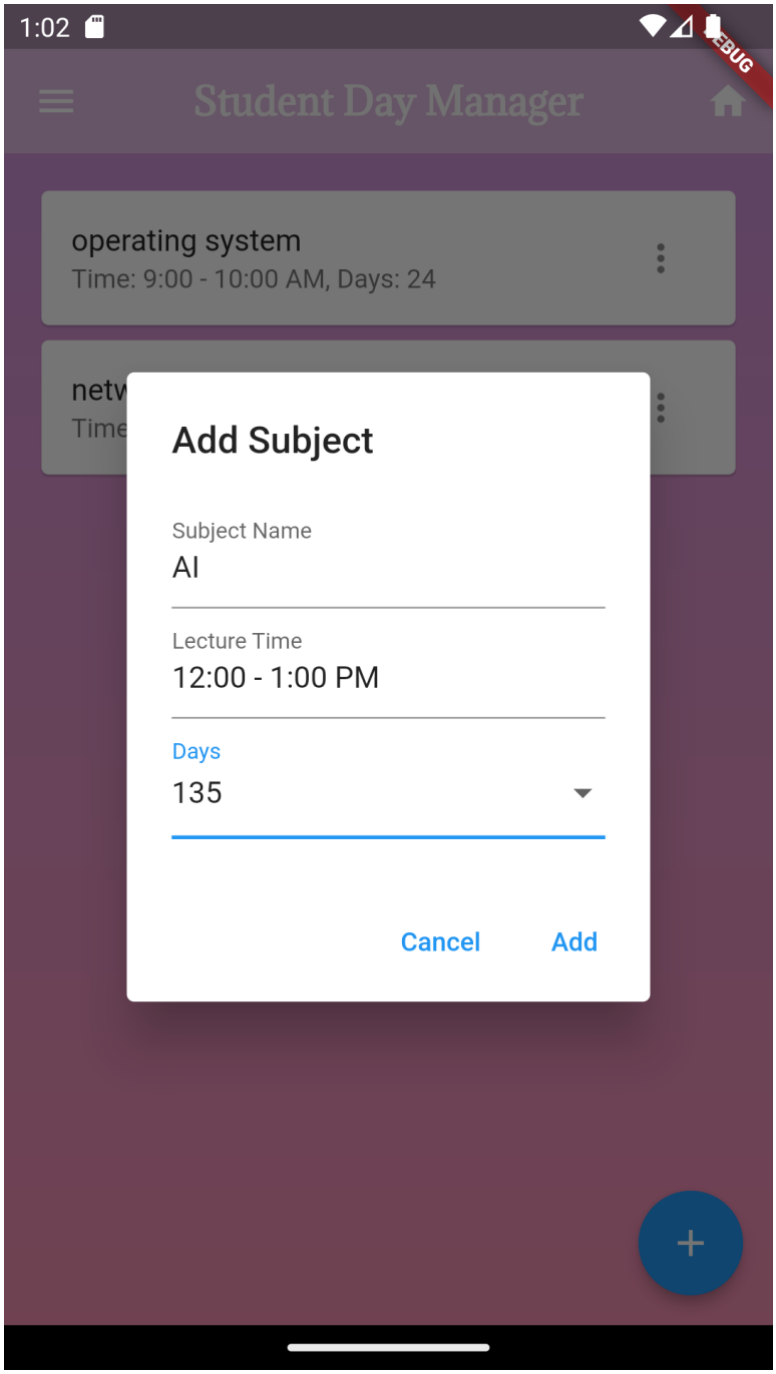


Figure 15: Caption for Figure 15 via *References-> student day manager page*

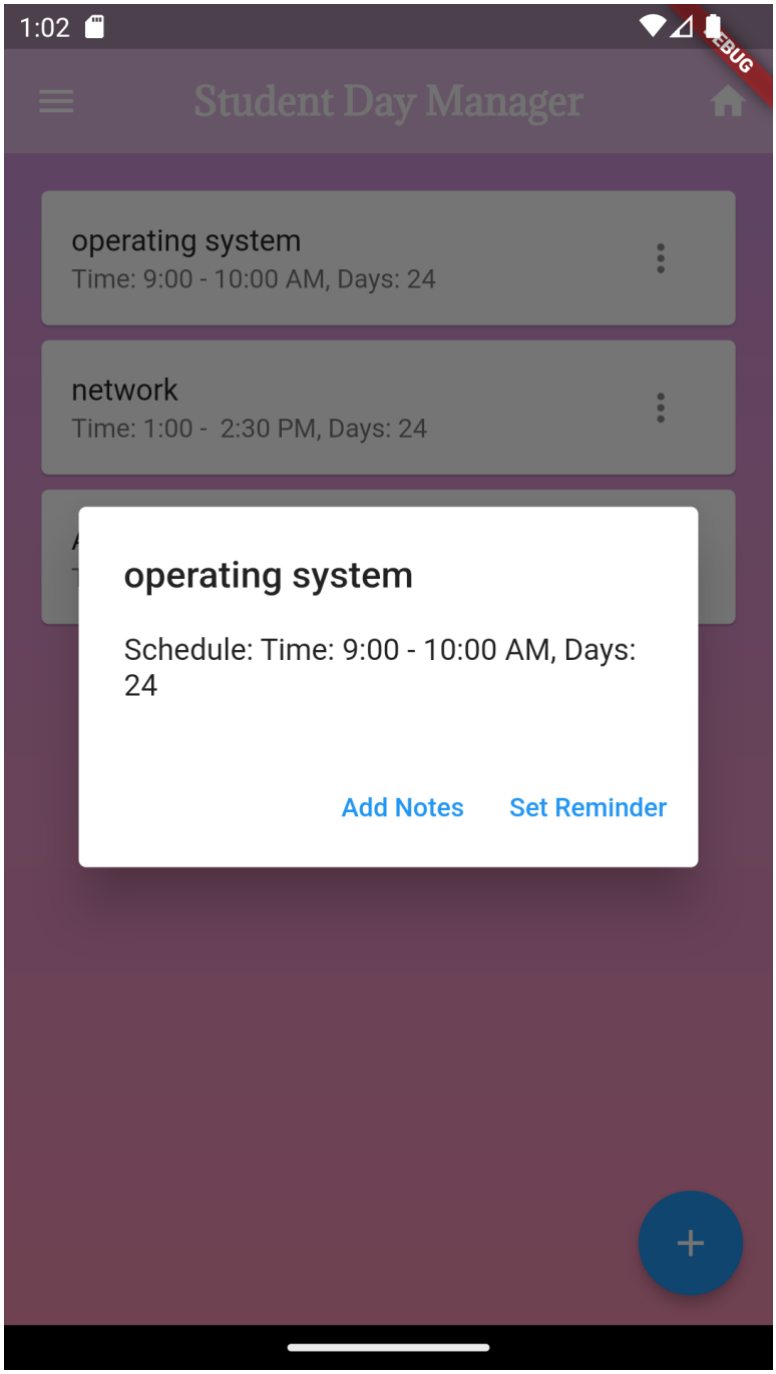


Figure 16: Caption for Figure 16 via *References-> student day manager options*

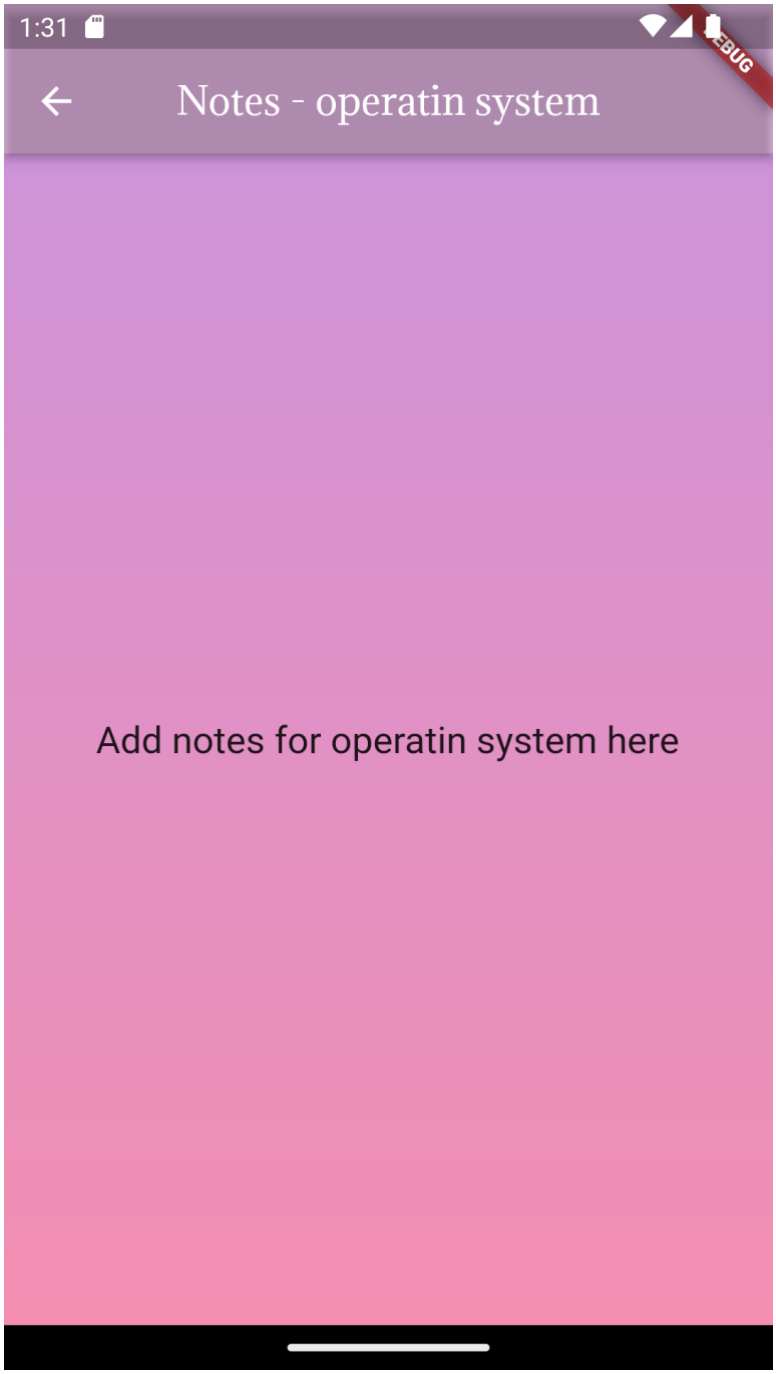


Figure 17: Caption for Figure 17 via *References-> student day manager subject note interface*

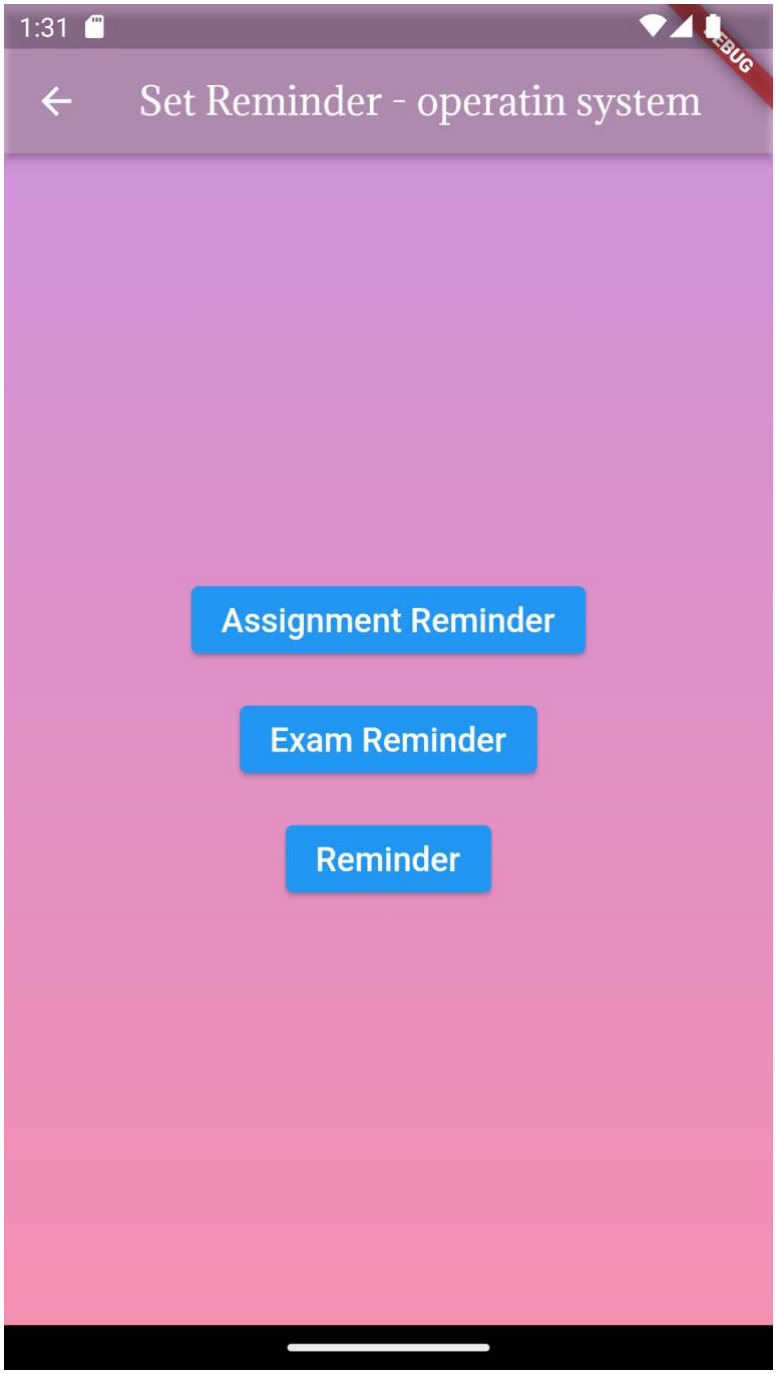


Figure 18: Caption for Figure 18 via *References-> student day manager setting reminders interface*

CHAPTER 5: IMPLEMENTATION PLAN

5.1 Description of Implementation

In this section of the Project Implementation Plan, we will provide a detailed description of the implementation process for the HUBOT solution. It outlines how the information system will be deployed, installed, and transitioned into an operational system. This includes an overview of the major tasks and components involved in the implementation, as well as the necessary resources to support the implementation effort.

The implementation of HUBOT will involve several key steps and components. These include:

Deployment and Installation: The HUBOT system will be deployed and installed on the necessary hardware infrastructure. This includes setting up servers, network configurations, and software installations. The installation process will ensure that all components of the system are properly installed and configured to function seamlessly.

Transition to Operational System: Once the installation is complete, the HUBOT system will undergo a transition phase to become fully operational. This includes conducting testing and quality assurance procedures to ensure the system's functionality, reliability, and performance meet the required standards.

Resource Requirements: The implementation effort will require various resources to support its success. These resources include hardware, such as servers and networking equipment, software applications, databases, etc.

5.2 Programming language and technology

The implementation of the HUBOT system will utilize the following programming languages and technologies:

Flutter: The HUBOT mobile application will be developed using the Flutter framework. Flutter provides a cross-platform development environment that allows for the creation of native-like mobile apps for both Android and iOS platforms.

Java: The server-side components and backend functionalities of the HUBOT system will be implemented using Java. Java is a widely used programming language known for its reliability and scalability.

Visual Studio Code: also commonly referred to as VS Code, IDE is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git [10].

Eclipse IDE: Eclipse is a popular IDE that offers a comprehensive set of tools and features for software development, including support for Java development. It provides a user-friendly interface, code editing capabilities, debugging tools, and project management features.

Firebase: Firebase will be used as the backend infrastructure and database for the HUBOT system. Firebase is a comprehensive platform provided by Google that offers various services, including real-time database, authentication, cloud storage, and hosting. It provides a scalable and reliable backend solution, allowing for seamless data storage, retrieval, and synchronization across different devices.

Rest API: In the context of the HUBOT system, the RESTful API will provide a set of endpoints that the mobile app can interact with to retrieve and manipulate data. These endpoints will follow the principles of REST, such as using HTTP methods (GET, POST, PUT, DELETE) to perform specific actions on resources and returning responses in a standardized format, typically JSON (JavaScript Object Notation).

Spring Boot framework: The implementation of the HUBOT system will leverage the Spring Boot framework to develop the server-side components and RESTful API. Spring Boot is a powerful Java-based framework that simplifies the development of robust and scalable applications.

Using Spring Boot with the Eclipse IDE, we can take advantage of its extensive tooling support and seamless integration. Eclipse provides features like code auto-completion, debugging, and project management, enhancing productivity during the development process.

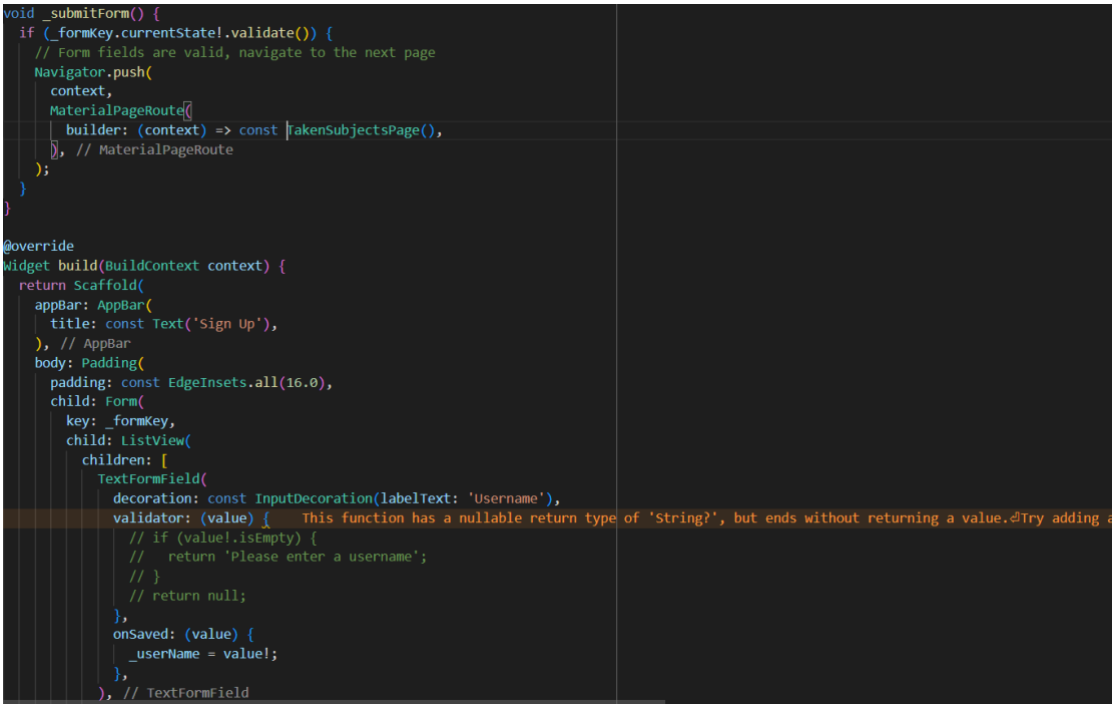
Additionally, Spring Boot offers integration with various databases, including Firebase, allowing us to store and retrieve data efficiently. We can leverage the Firebase SDK and APIs to interact with the Firebase Real-time Database or Firestore for data persistence.

By utilizing Spring Boot, we can build a scalable, efficient, and maintainable server-side application for the HUBOT system. It provides a robust foundation for implementing the RESTful API, handling requests, managing data, and ensuring the security and integrity of the system.

5.3 part of implementation if possible

```
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 40),
  child: TextFormField(
    decoration: InputDecoration(
      labelText: 'Email',
      labelStyle: const TextStyle(color: Colors.white),
      filled: true,
      fillColor: Colors.white.withOpacity(0.3),
      enabledBorder: OutlineInputBorder(
        borderSide:
            BorderSide(color: Colors.white.withOpacity(0.3)),
        borderRadius: BorderRadius.circular(30),
      ), // OutlineInputBorder
      focusedBorder: OutlineInputBorder(
        borderSide:
            BorderSide(color: Colors.white.withOpacity(0.7)),
        borderRadius: BorderRadius.circular(30),
      ), // OutlineInputBorder
    ), // InputDecoration
  ), // TextFormField
), // Padding
const SizedBox(height: 30),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 40),
  child: TextFormField(
    obscureText: true,
    decoration: InputDecoration(
      labelText: 'Password',
      labelStyle: const TextStyle(color: Colors.white),
      filled: true,
      fillColor: Colors.white.withOpacity(0.3),
      enabledBorder: OutlineInputBorder(
        borderSide:
            BorderSide(color: Colors.white.withOpacity(0.3)),
        borderRadius: BorderRadius.circular(30),
      ), // OutlineInputBorder
      focusedBorder: OutlineInputBorder(
```

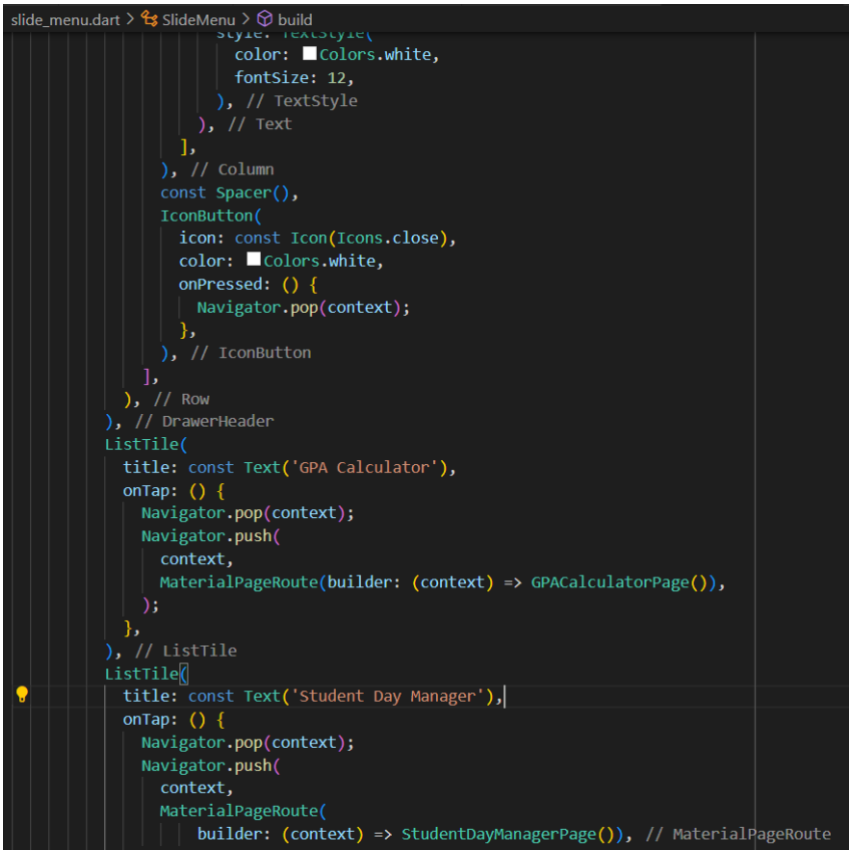
Figure 19: Caption for Figure 19 via *References-> part of login page*



```
void _submitForm() {
  if (_formKey.currentState!.validate()) {
    // Form fields are valid, navigate to the next page
    Navigator.push(
      context,
      MaterialPageRoute(
        builder: (context) => const TakenSubjectsPage(),
      ), // MaterialPageRoute
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Sign Up'),
    ), // AppBar
    body: Padding(
      padding: const EdgeInsets.all(16.0),
      child: Form(
        key: _formKey,
        child: ListView(
          children: [
            TextFormField(
              decoration: const InputDecoration(labelText: 'Username'),
              validator: (value) {    This function has a nullable return type of 'String?', but ends without returning a value.Try adding a
                // if (value!.isEmpty) {
                //   return 'Please enter a username';
                // }
                // return null;
              },
              onSaved: (value) {
                _userName = value!;
              },
            ), // TextFormField
```

Figure 20: Caption for Figure 20 via *References-> part of sigu up page code*



```
slide_menu.dart > SlideMenu > build
                    color: Colors.white,
                    fontSize: 12,
                  ), // TextStyle
                ), // Text
              ],
            ), // Column
            const Spacer(),
            IconButton(
              icon: const Icon(Icons.close),
              color: Colors.white,
              onPressed: () {
                Navigator.pop(context);
              },
            ), // IconButton
          ],
        ), // Row
      ), // DrawerHeader
      ListTile(
        title: const Text('GPA Calculator'),
        onTap: () {
          Navigator.pop(context);
          Navigator.push(
            context,
            MaterialPageRoute(builder: (context) => GPACalculatorPage()),
          );
        },
      ), // ListTile
      ListTile(
        title: const Text('Student Day Manager'),
        onTap: () {
          Navigator.pop(context);
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => StudentDayManagerPage()), // MaterialPageRoute
```

Figure 21: Caption for Figure 21 via *References-> part of slide menu code*

```
padding: EdgeInsets.all(16.0),    Use 'const' with the constructor to improve perfo
child: Column(
  children: [
    if (_subjectDataList.isEmpty)
    const  Text(
        'No subjects added',
        style: TextStyle(fontSize: 16),
      ), // Text
    if (_subjectDataList.isNotEmpty)
      Expanded(
        child: ListView.builder(
          itemCount: _subjectDataList.length,
          itemBuilder: (context, index) {
            SubjectData subjectData = _subjectDataList[index];
            return Card(
              child: InkWell(
                onTap: () {
                  _showSubjectDialog(subjectData);
                },
                child: ListTile(
                  title: Text(subjectData.subjectName),
                  subtitle: Text(subjectData.schedule),
                  trailing: PopupMenuButton<String>(
                    onSelected: (value) {
                      if (value == 'remove') {
                        _removeSubject(subjectData);
                      } else if (value == 'edit') {
                        _editSubject(subjectData);
                      }
                    },
                    itemBuilder: (BuildContext context) {
                      return {'remove', 'edit'}.map((String choice) {
                        return PopupMenuItem<String>(
                          value: choice,
                          child: Text(choice),
                        ); // PopupMenuItem
                      }).toList();
```

Figure 22: Caption for Figure 22 via *References-> part of student day manager code*

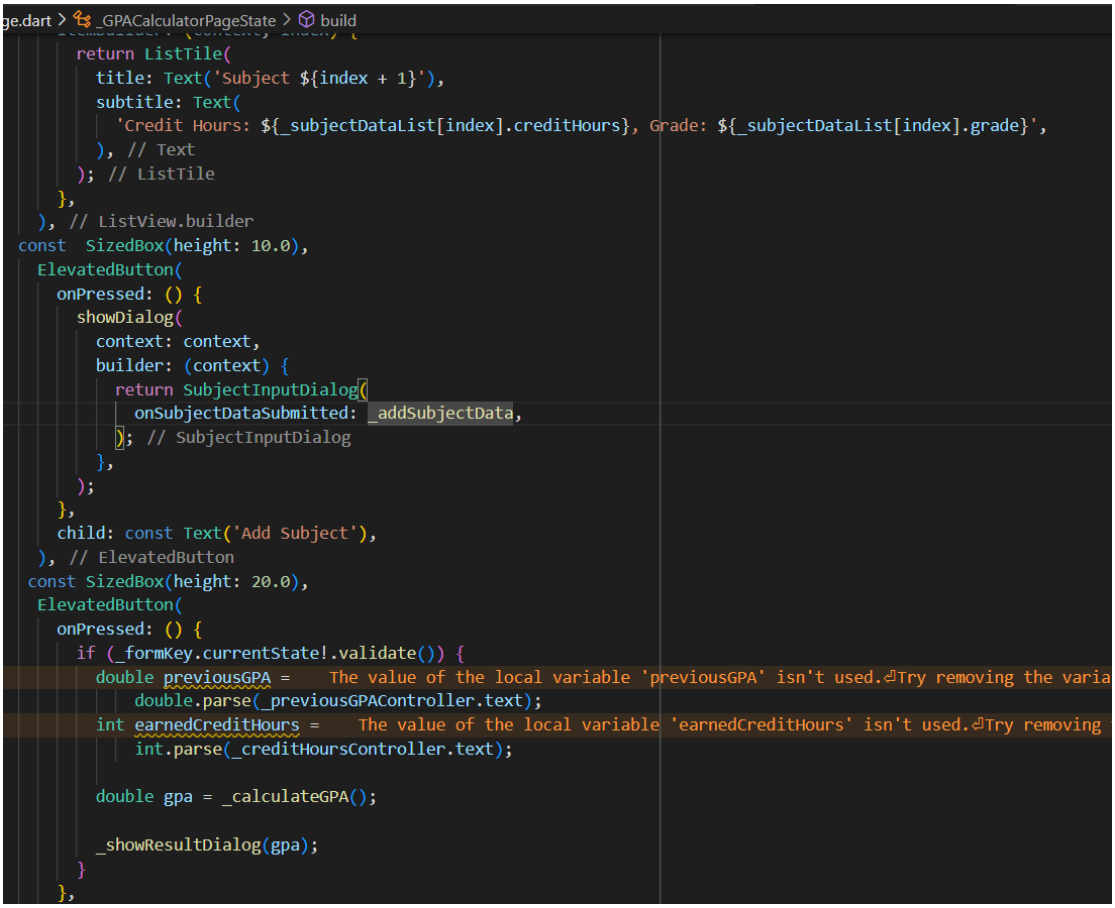


Figure 23: Caption for Figure 23 via *References-> part of GBA calculator code*

CHAPTER 6: TESTING PLAN

Describe the scope, approach, resources and schedule of intended test activities. It identifies amongst others test items, the features to be tested, the testing tasks, test coverage, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice.

6.1 Black-box

In order to ensure the quality and reliability of the HUBOT system, various black-box testing techniques will be employed. These techniques focus on testing the system's functionality without considering its internal structure or implementation details. The following black-box techniques will be utilized:

Equivalence Partitioning: Test cases will be designed to cover different equivalence classes of input data. For example, if the system accepts numerical input, test cases will be created to represent valid and invalid ranges of numbers.

Boundary Value Analysis: Test cases will focus on the boundary values of input ranges to ensure that the system handles them correctly. For instance, if a field accepts values

from 1 to 100, test cases will include values like 1, 100, and values just below and above these boundaries.

Decision Table Testing: This technique will be used to test the system's behavior based on different combinations of input conditions. Decision tables will be created to represent different scenarios and test cases will be designed accordingly.

6.2 White-box

White-box testing techniques will be employed to assess the internal structure and implementation of the HUBOT system. As the code for the project is available, the following white-box techniques will be used:

Statement Coverage: Test cases will be designed to ensure that every line of code is executed at least once. This technique helps identify any dead or unreachable code segments.

Branch Coverage: Test cases will be created to ensure that all possible branches and decision points within the code are exercised. This helps uncover potential logical errors or missing conditions.

Path Coverage: Test cases will be designed to cover all possible paths through the code. This technique ensures that every possible combination of conditions and loops is tested.

6.3Testing automation

1. To improve the efficiency and effectiveness of testing, automation tools will be utilized. The following automation tools will be used for controlling test execution and comparing actual outcomes with predicted outcomes:

J Unit: JUnit is a widely used testing framework for Java applications. It provides a set of annotations and assertions that facilitate the creation and execution of automated tests. JUnit will be used for unit testing individual components of the HUBOT system.

2. Deciding which test cases to automate will be based on factors such as the frequency of execution, complexity, and criticality of the test case. Test cases that are repetitive, time-consuming, or require extensive manual effort will be prioritized for automation. Conclusion and Results

The conclusion is a required part that closes the document with a brief summary of the study including the problems found and the proposed solution. Most importantly, it should recommend to the readers the benefits of pursuing the project based on the researcher's analysis.

CHAPTER 7:CONCLUSION AND RESULTS

7.1 **Summary of Accomplished Project:**

- more than 65% of prototype interfaces are ready.
- we have determined the main functions we want to provide in HUBOT.
- implemented the dart code for log-in page.

7.2 Future Work: there is still room for further development and enhancement. Here are some recommendations for future work:

Enhancing Chatbot Intelligence: Continuously improve the Chabot's natural language processing capabilities to understand and respond to a wider range of user queries accurately.

Expanding Smart Recommendation System: Extend the smart recommendation feature to suggest not only subjects and class schedules but also extracurricular activities, study resources, and campus events based on the user's preferences and academic profile.

Integration with External Systems: Explore the possibility of integrating the app with other university systems and services, such as the library system, student information system, or event management platform, to provide a more comprehensive and seamless user experience.

Collaborative Features: Implement features that enable users to collaborate on group projects, share study materials, and facilitate communication among students within the app.

Personalization and Customization: Allow users to personalize their app experience by customizing the app's theme, layout, and notification preferences according to their preferences.

Overall, the HUBOT mobile app presents a promising solution to improve the efficiency and convenience of campus life. Its current features provide a solid foundation, and future enhancements can further elevate the app's functionality and user experience. By pursuing this project, users can benefit from a comprehensive and intuitive tool that simplifies various aspects of their academic journey while fostering a connected and engaged campus community.