SECP3204: Software Engineering WBL

System Design Descriptions (SDD)

UTM Student Feedback System

Version 1.0

24/05/2023

School of Computing, Faculty of Engineering

Prepared by: Limapuluh

Table of Contents

3. System Architectural Design

3.1 Architecture Style and Rationale

Architecture styles are important in software development because they provide a structured approach to designing and organizing complex systems. Developers can create scalable and maintainable modular software solutions by using an appropriate architectural style, such as the Model-View-Controller (MVC) model. Although there are various architectural styles, this paper focuses on the use of MVC in a student feedback system. The Model-View-Controller (MVC) architecture divides an application into three main components: model, view and controller. This model is intended to separate concerns and promote code organization, making the system easier to understand, develop, and maintain. This model represents the application's data and business logic. It contains the essential functions, such as data access, operation, and verification. Views are in charge of presenting data to users in a user-friendly manner. It is primarily concerned with the visualization and user interface modules. The controller handles user input as an intermediary between the model and the view, updating the model and the view accordingly.

The model component is the foundation of the MVC architecture. It encapsulates the data and implements the student feedback system's business logic. The model component interfaces data access and operation, ensuring data integrity and consistency. It applies business rules and validates user input. Various technologies or frameworks can be used to realise the model component. For example, you can store and manage student feedback data using a relational database management system (RDBMS). Object Relational Mapping (ORM) frameworks can provide an abstraction by simplifying the interaction between the application and the database.

The presentation layer of the student feedback system is the focus of the view component. It is in charge of displaying data to users and presenting the user interface. To improve the user experience, the view component should provide an intuitive interactive interface. We can simplify the implementation of the view component by using web-based technologies such as HTML, CSS, and JavaScript. Furthermore, the front-end frame is similar to React, Angular, or Vue. Js can make development easier by providing reusable UI components.

The Controller component serves as a bridge between the model and the view. It takes user input, executes appropriate model actions, and updates the view accordingly. The Controller component ensures that the model and view are kept separate, allowing for independent development and testing. You can use programming languages like Java, Python, or JavaScript to implement the controller component. Spring MVC, Django, and Express. Js frameworks include built-in support for implementing controllers and managing request handling.

The decision to use the MVC framework in the student feedback system was made for several reasons. To begin with, MVC implementation's separation of concerns allows each component to be developed and tested independently. This modularity improves code maintainability and is advantageous for future system enhancements or updates. Second, MVC encourages code organization and reusability. Following the MVC pattern allows developers to organize their code base into different components, making it easier to navigate and understand. This encourages team members to work together and shortens development time. Furthermore, MVC is in line with the needs and goals of the student feedback system. The model component handles data access, operation, and verification, as well as ensuring the accuracy and dependability of student feedback. The view component is responsible for creating an appealing, user-friendly interface, whereas the controller component is in charge of managing user input and updating the system accordingly.

In a nutshell, using the MVC architecture for the student feedback system has numerous advantages. MVC's separation of concerns, code organization, and modularity contribute to the system's scalability, maintainability, and extensibility. Developers can meet the project's requirements by implementing model, view, and controller components to create a robust, user-friendly student feedback system. The architecture design stage is critical in the software development process, laying the groundwork for a successful implementation. Developers can ensure a well-structured and effective system by carefully considering the use of the MVC framework.

3.2 **Component Model**

The component model section of the architecture design document is critical in describing the structure and interactions of system components. The component model provides a clear representation of the system architecture and aids in understanding how different parts of the system work together by summarizing components and their relationships.

The following component diagram illustrates the structure and relationships of the system components:

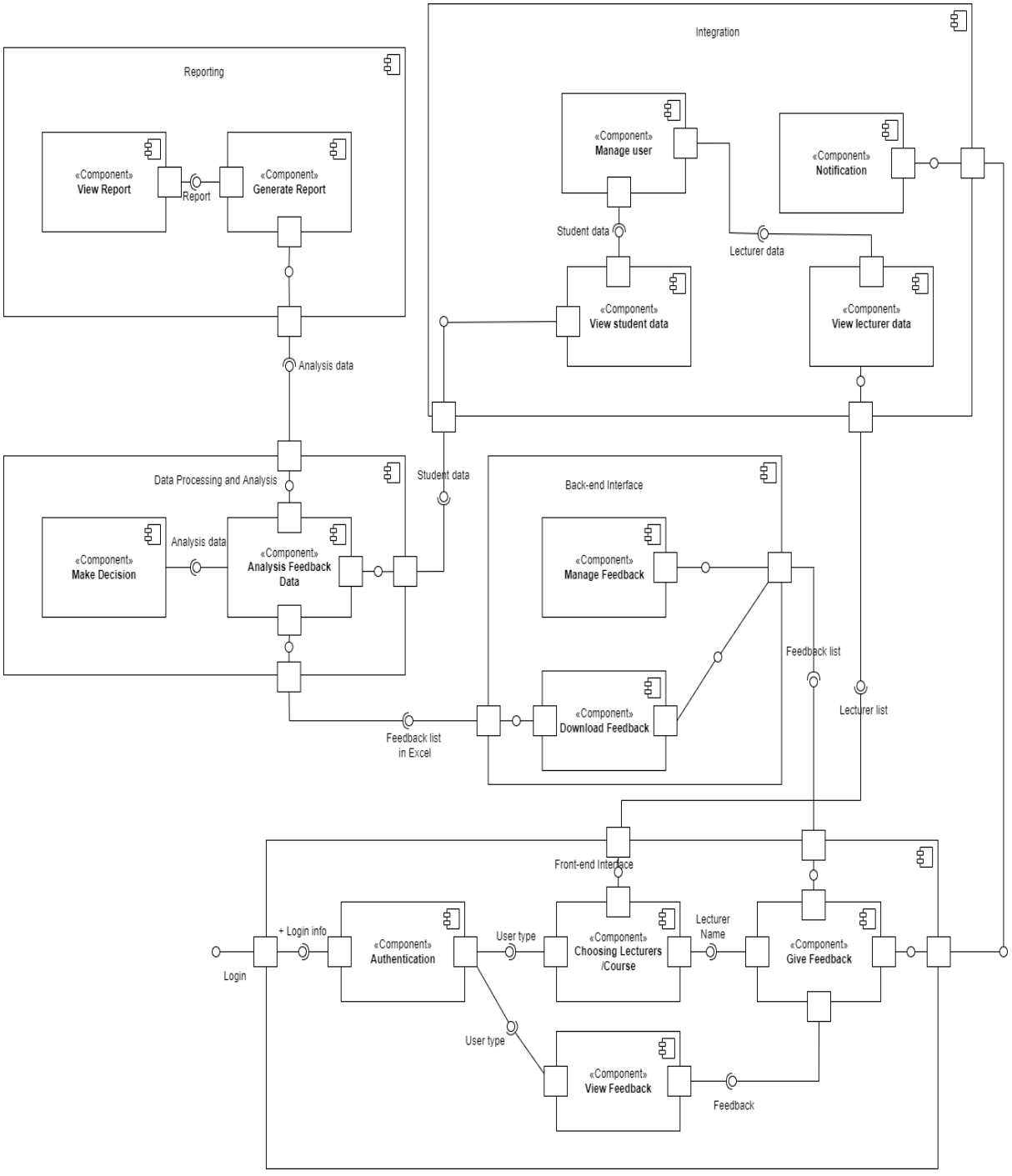


Figure 3.1: Component Diagram of <UTM Student Feedback System>

The component diagram is composed of five subsystems, which contain components providing corresponding functions. Each component either interacts with each other or is an intra-subsystem via interface. The five subsystems are "Front-end Interface", "Back-end Interface", "Data Processing and Analysis Module", "Reporting Module" and "Integration Module".

The "Front-end Interface" subsystem consists of an "Authentication" component, a "Choosing Lecturers/Courses" component, a "Give Feedback" component and a "View Feedback" component. The "Authentication" component enables the system to verify each user and determine "who they are" and "did they enter the valid username and password". It provides user authentication service, which consists of the Login and User Session provider interface. From the view level, every time a user logs in, the login function of the Authentication component will be called. While the "Choosing Lecturers/Courses" component is responsible for providing a user-friendly interface for students to select their preferred lecturers or courses. The Give Feedback component allows students to submit feedback on their courses and lecturers. The "View Feedback" component allows the instructor to view student feedback. It presents the feedback data in a meaningful way, facilitating their analysis and decision-making.

A "Manage Feedback" component and a "Download Feedback" component are part of the "Back-end Interface" subsystem. The MVC architecture's Controller component is in charge of dealing with business logic related to management feedback. It accepts feedback, validates the data, and communicates with the Model component to store and retrieve feedback information. A Controller component responsible for downloading feedback data in Excel format. It communicates with the Model component to retrieve the required data and create a downloadable file.

"Analyzing Feedback Data" and "Making Decision" are two components of the "Data Processing and Analysis Module" subsystem. The MVC architecture's model component is in charge of processing and analyzing Feedback Data. It employs a variety of data analysis techniques and algorithms to glean insights, identify patterns, and extract meaningful information from feedback data. The Controller component uses the Model's analyzed feedback data to assist lecturers in making important decisions. According to the analysis results, it offers suggestions, opinions, and actionable information.

The "Reporting Module" subsystem is made up of two components: "Generate Report" and "View Report." The controller component is in charge of generating a detailed report based on the summarised feedback data. It retrieves the required data from the model, formats it, and generates a report for display or export. Administrators can view the generated reports using the View component. It presents the reports in an easy-to-understand format, giving administrators a thorough understanding of overall feedback trends and enabling them to make informed decisions.

Lastly, the "Integration Module" subsystem contains "Notification", "Manage User", "View Student Data" and "View Lecturer Data" components. The controller component is responsible for handling the system notification. It interacts with Models and Views, and sends notifications to lecturers and administrators when a student. Submit a feedback form to a specific lecturer. Besides, while the lecturer replies to the feedback received from the student, notification is also sent to the student. When any lecturer generates his feedback report, the administrator might also be notified. The "Manage User" component allows administrators to manage user accounts, including students, lecturers, and other administrative personnel. It interacts with the Model to perform user management operations such as creating, modifying and deleting user accounts. The "View Student Data" component enables lecturers to view student data related to their courses. It retrieves and delivers relevant student information from the Model to the View component, allowing lecturers to gain insights into student performance and feedback. The "View Lecturer Data" is responsible for delivering lecturer-related data to the View component for students. It retrieves and displays the detailed information of the lecturers' teaching, so that students can make informed decisions based on the qualifications of the lecturer and previous feedback.

4. **Detailed Description of Components**

4.1 **Complete Package Diagram**

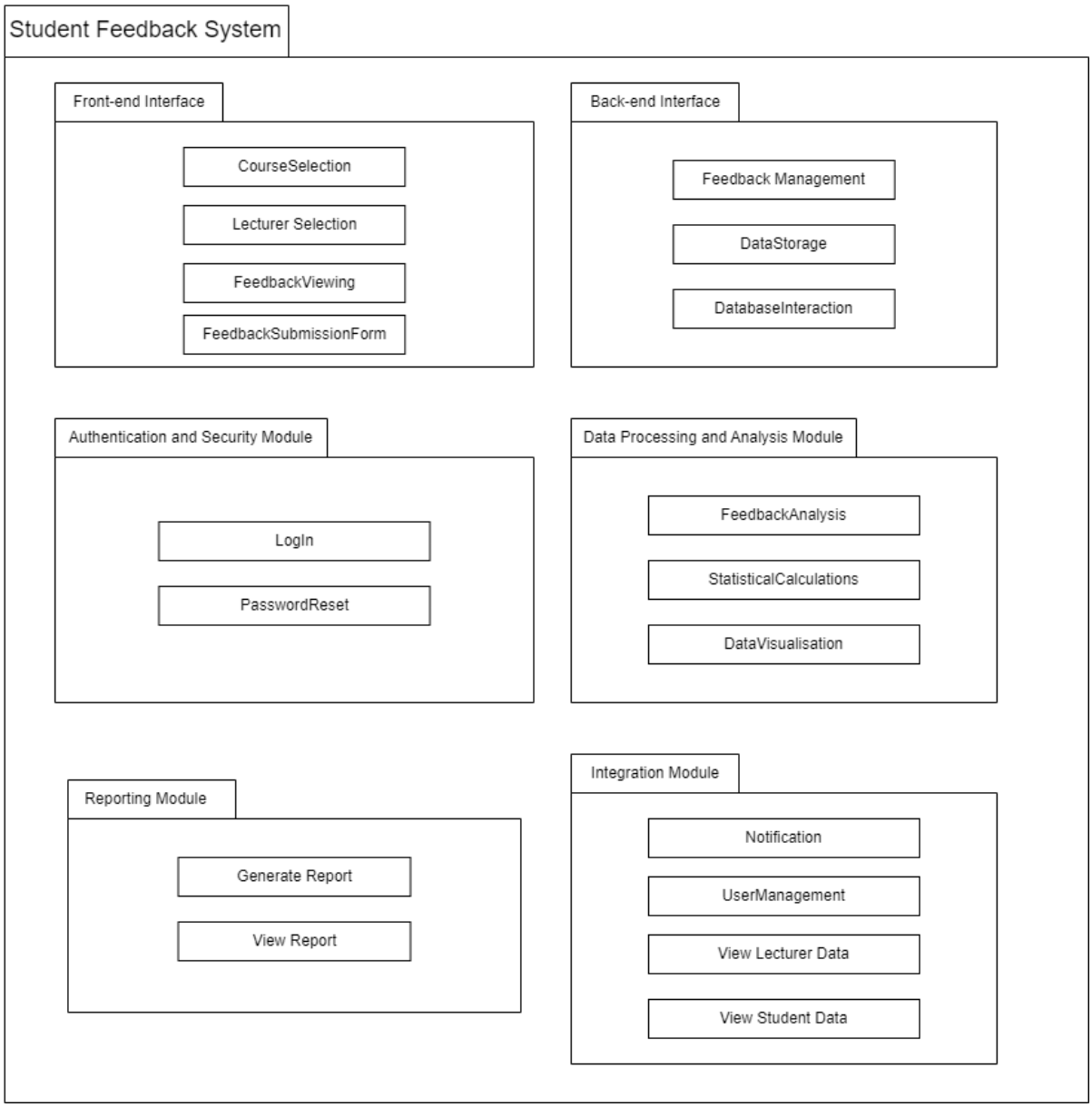


Figure 4.1 Package Diagram for <UTM Student Feedback System>

Figure above shows the 6 subsystems in UTM Student Feedback System in Package Diagram form, where each subsystem is described with their functionalities. Components in each subsystem will later be identified in a more detailed manner.

4.2 **Detailed Description**

4.2.1 **P001: <Authentication and Security Module> Subsystem**

Figure 4.2.1.1: Package Diagram for <Authentication and Security Module> Subsystem

4.2.1.1 **Class Diagram**

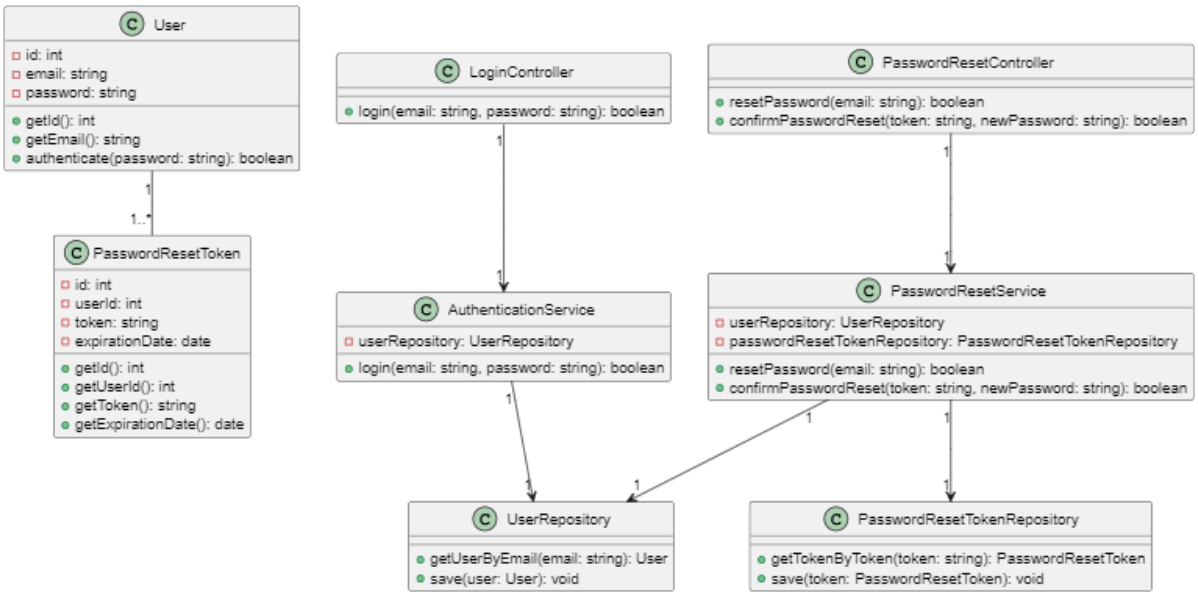


Figure 4.2.1.2: Class Diagram for <Authentication and Security Module> Subsystem

Entity Name	Method Name	Input	Output	Algorithm
User	getId()	-	int	Return the ID of the user
User	getEmail()	-	string	Return the email of the user

User	authenticate()	password: string	boolean	Compare input password with user's password and return true if they match
PasswordResetToken	getId()	-	int	Return the ID of the password reset token
PasswordResetToken	getUserId()	-	int	Return the user ID associated with the token
PasswordResetToken	getToken()	-	string	Return the token string
PasswordResetToken	getExpirationDate()	-	date	Return the expiration date of the token
LoginController	login()	email: string, password: string	boolean	Call the AuthenticationService to verify the login credentials and return true if successful
PasswordResetController	resetPassword()	email: string	boolean	Call the PasswordResetService to initiate the password reset process and return true if successful
PasswordResetController	confirmPasswordReset()	token: string, newPassword: string	boolean	Call the PasswordResetService to confirm the password reset with the provided token and new password, and return true if successful

AuthenticationService	login()	email: string, password: string	boolean	Retrieve the user from the UserRepository based on the email, then call the authenticate() method of the user to verify the password
PasswordResetService	resetPassword()	email: string	boolean	Retrieve the user from the UserRepository based on the email, generate a unique password reset token, save it in the PasswordResetTokenRepository, and send an email to the user with the token
PasswordResetService	confirmPasswordReset()	token: string, newPassword: string	boolean	Retrieve the token from the PasswordResetTokenRepository based on the token string, validate the token's expiration date, retrieve the user associated with the token from the UserRepository, update the user's password, and save the changes
UserRepository	getUserByEmail()	email: string	User	Retrieve the user from the database based on the email
UserRepository	save()	user: User	-	Save the user in the database
PasswordResetTokenRepository	getTokenByToken()	token: string	PasswordResetToken	Retrieve the password reset token from the database based on the token string

PasswordResetToken Repository	save()	token: PasswordReseTToken	-	Save the password reset token in the database

4.2.1.2 **Sequence Diagrams**

a) SD001: Log In (UTM Email)

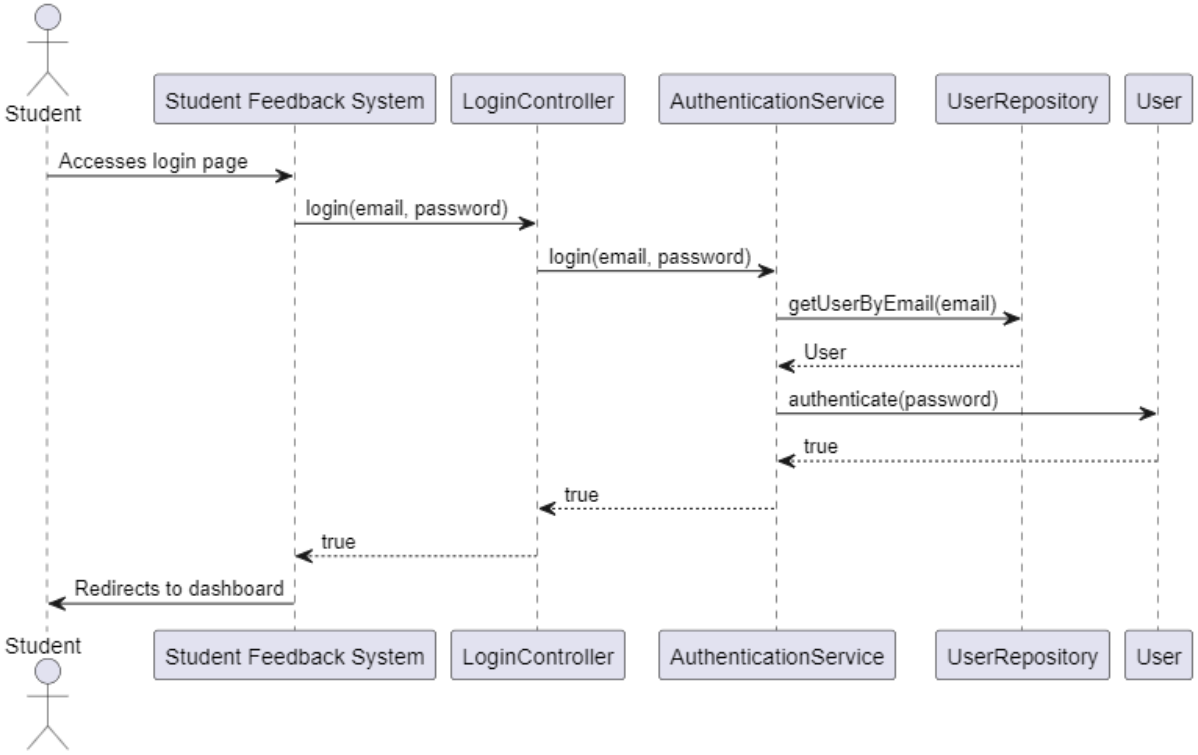


Figure 4.2.1.3: Sequence Diagram for <Log In (UTM Email)>

b) SD002: Sequence diagram for Reset Password

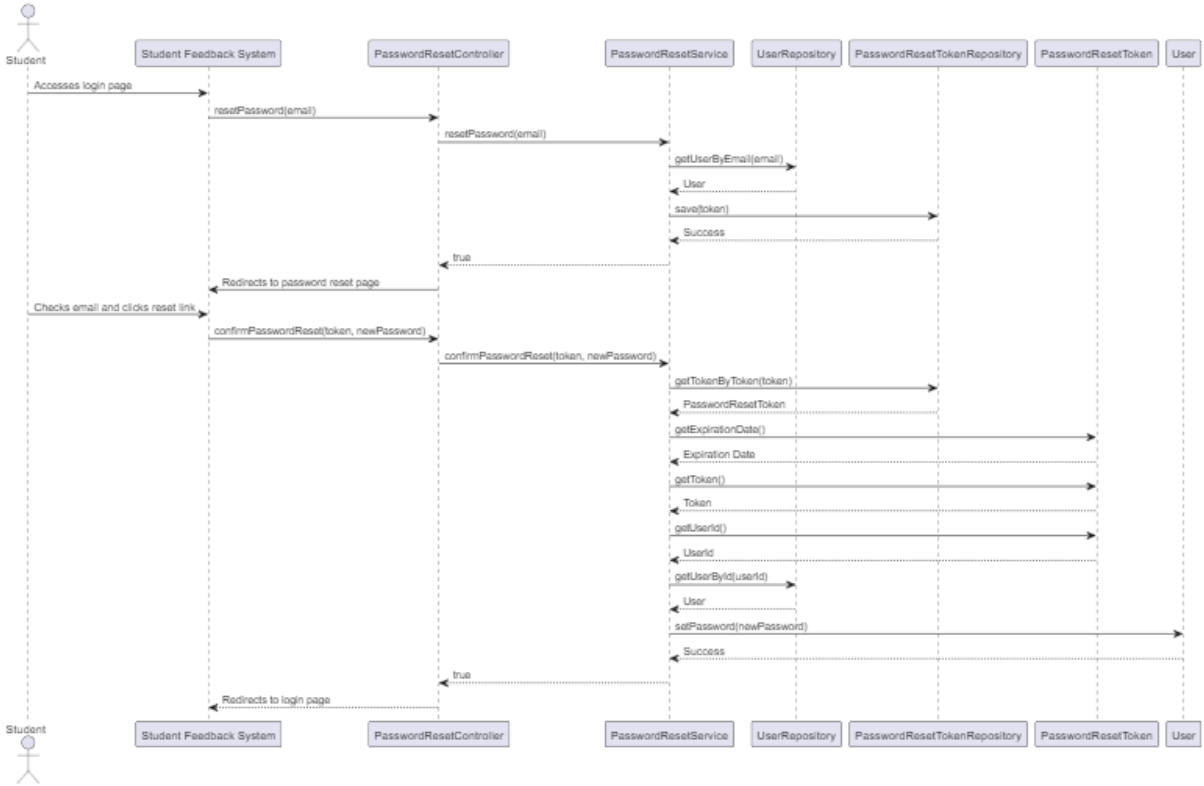


Figure 4.2.1.4: Sequence Diagram for <Reset Password>

4.2.2 **P002: <Front-end Interface> Subsystem**

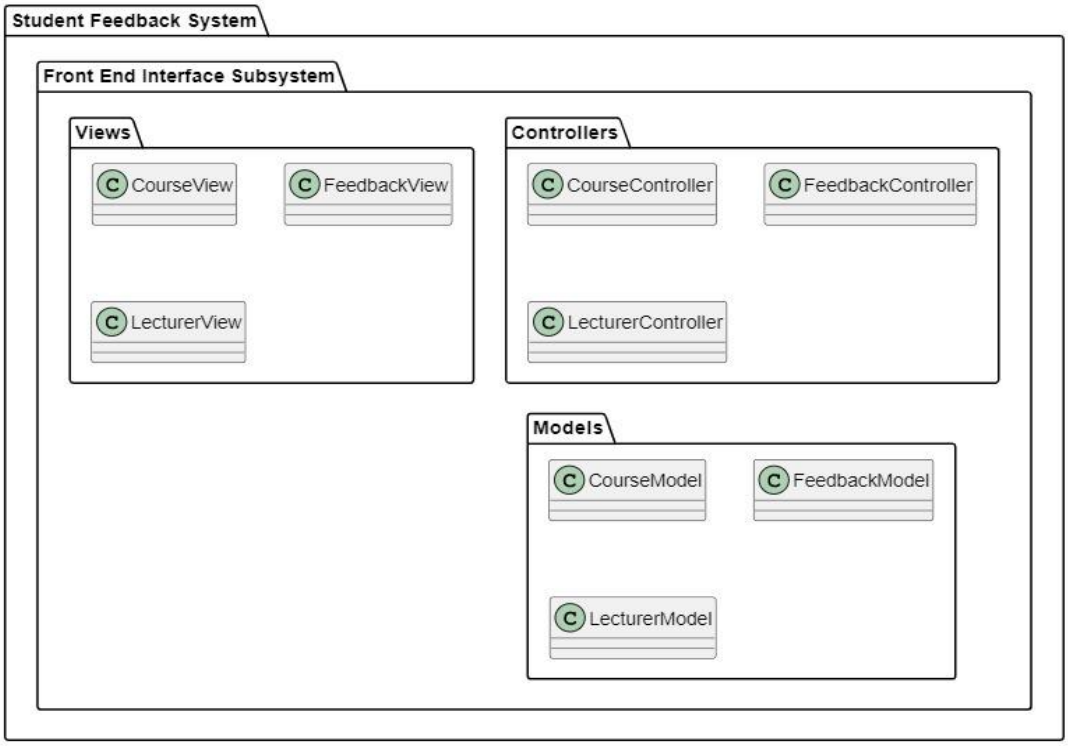


Figure 4.2.2: Package Diagram for <Front-end Interface> Subsystem

4.2.2.1 **Class Diagram**

Figure 4.2.2.1: Class Diagram for <Front-end Interface> Subsystem

Entity Name	Course
Method Name	getCourseCode() : string
Input	None
Output	Course code as string
Algorithm	Return the code of the course

Entity Name	Course
Method Name	getCourseName() : string
Input	None
Output	Course name as string
Algorithm	Return the name of the course

Entity Name	Feedback
Method Name	getStudentName() : string
Input	None
Output	Student name as string
Algorithm	Return the course associated with the feedback.

Entity Name	Feedback
Method Name	getCourse() : Course
Input	None
Output	Course object
Algorithm	Return the name of the course
Entity Name	Feedback
Method Name	getLecturer() : Lecturer
Input	None
Output	Lecturer object
Algorithm	Return the lecturer associated with the feedback.

Entity Name	Feedback
Method Name	getResponses() : List
Input	None
Output	List objects
Algorithm	Return the list of responses given by the student for the feedback.

Entity Name	Feedback
Method Name	getComments() : string

Input	None
Output	Comments as a string
Algorithm	Return the comments provided by the student for the feedback.

Entity Name	Lecturer
Method Name	getLecturerCode() : string
Input	None
Output	Lecturer code as string
Algorithm	Return the lecturer code of the lecturer.

Entity Name	Lecturer
Method Name	getLecturerName() : string
Input	None
Output	Lecturer name as string
Algorithm	Return the name of the lecturer

Entity Name	Question
Method Name	getQuestionText() : string
Input	None
Output	Question text as a string
Algorithm	Return the text of the question

Entity Name	Question
Method Name	getResponseOptions() : List
Input	None
Output	List objects
Algorithm	Return the list of response options for the question.

Entity Name	Response
Method Name	getQuestion() : Question
Input	None
Output	Question object

Algorithm	Return the question associated with the response.

Entity Name	Response
Method Name	getAnswer() : string
Input	None
Output	Answer as a string
Algorithm	Return the answer provided for the question.

4.2.2.2 **Sequence Diagram**

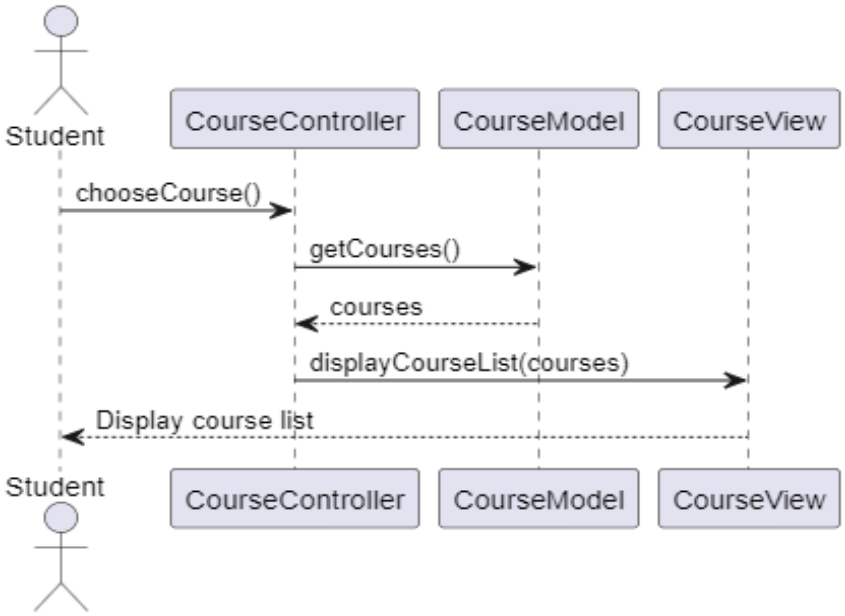


Figure 4.2.2.2.1: Sequence Diagram for <Choosing Lecturer / Course>

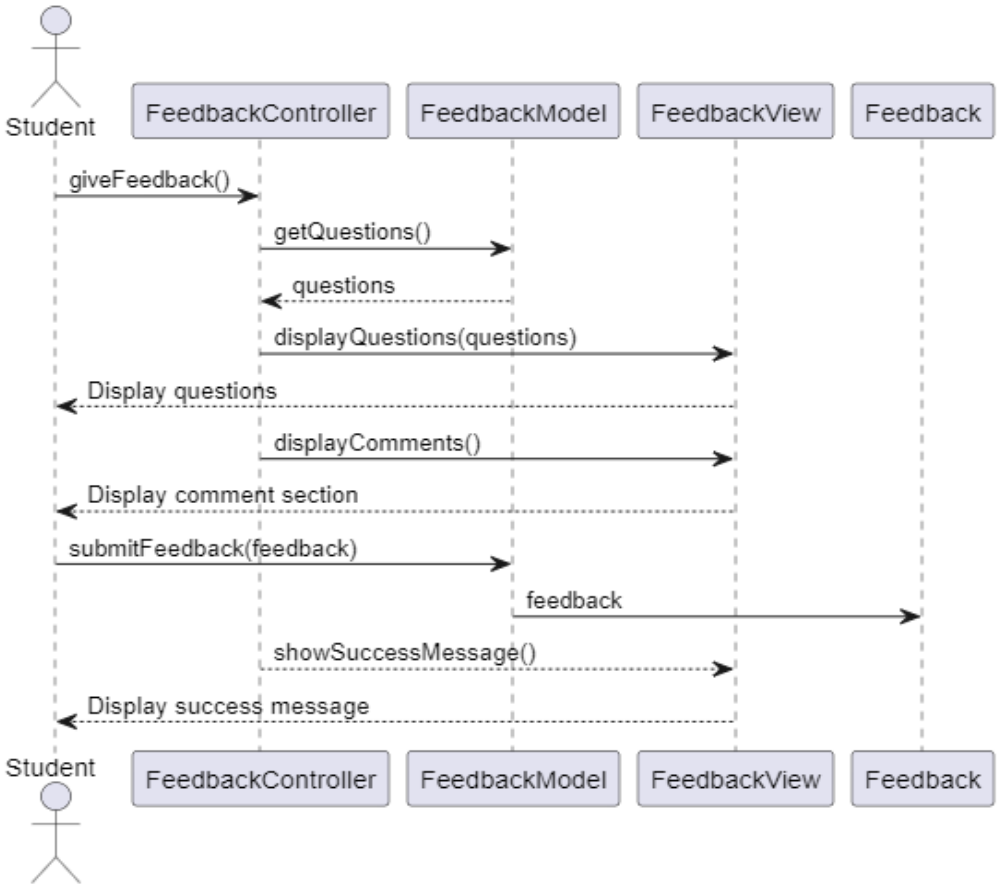


Figure 4.2.2.2.2: Sequence Diagram for <Give Feedback>

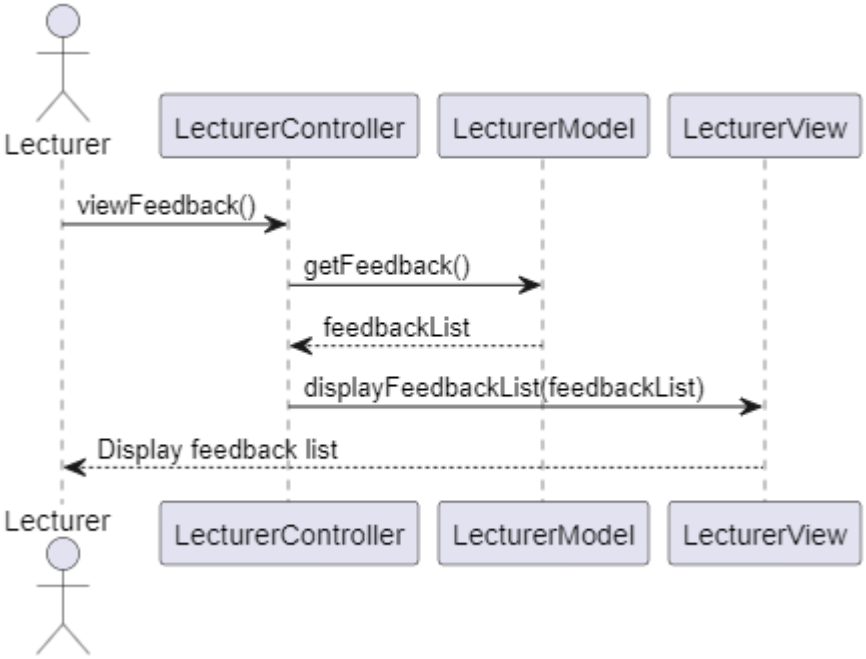


Figure 4.2.2.2.3: Sequence Diagram for <View Feedback>

4.2.3 **P003: <Back-end Interface> Subsystem**

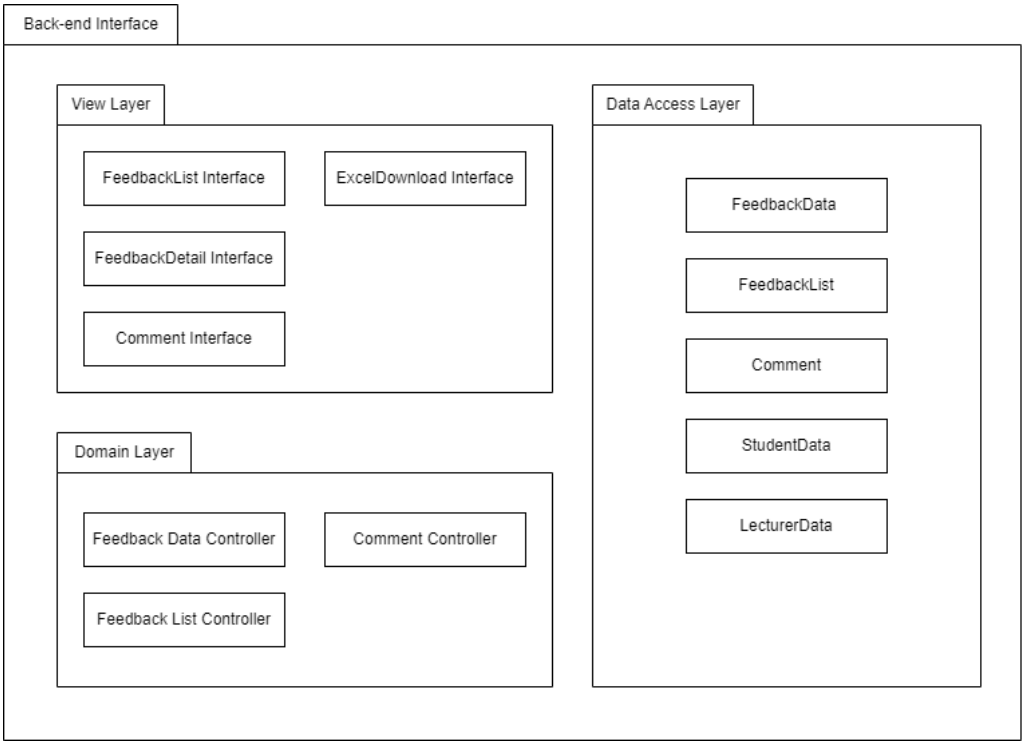


Figure 4.2.3 Package Diagram for <Back-end Interface> Subsystem

4.2.3.1 **Class Diagram**

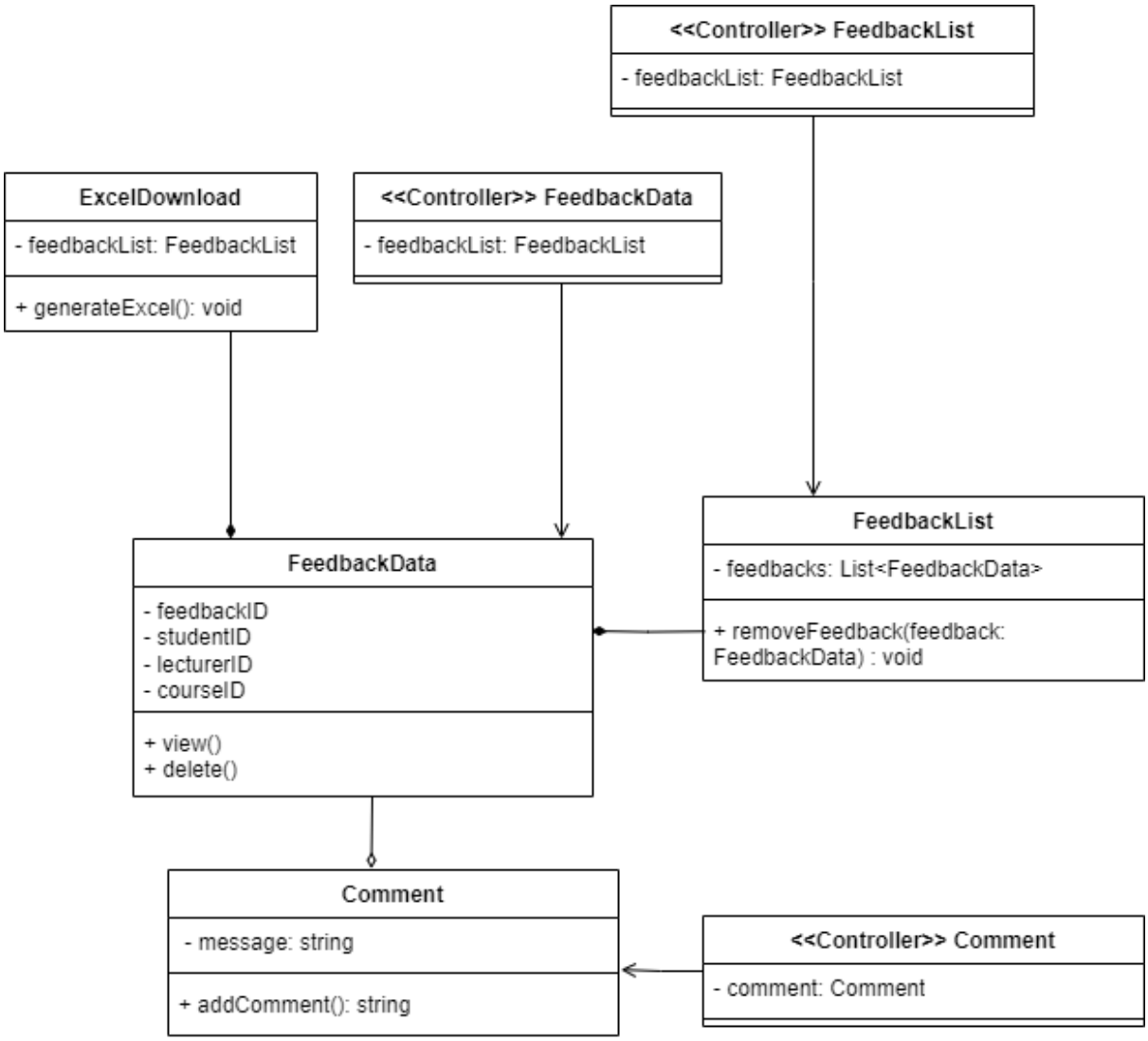


Figure 4.2.3.1 Class Diagram for <Back-end Interface> Subsystem

Entity Name	Comment
Method Name	addComment
Input	message, feedbackID
Output	-
Algorithm	1. Start 2. Search the feedback according to feedbackID 3. Prompt user to enter the comment for the feedback

	4. User click “Confirm” to create Comment object 5. Messages becomes linked to the feedbackID 6. End

Entity Name	FeedbackList
Method Name	removeFeedback
Input	feedbackID
Output	-
Algorithm	1. Start 2. Search for corresponding FeedbackData object based on feedbackID 3. Prompt user to confirm deletion of the object 4. Delete the object from the FeedbackList 5. End

Entity Name	ExcelDownload
Method Name	generateExcel
Input	feedbackID
Output	Excel file
Algorithm	1. Start 2. Search the FeedbackData in database based on feedbackID 3. User presses the “Download Excel” button 4. Excel file is downloaded to user’s device 5. End

4.2.3.2 **Sequence Diagram**

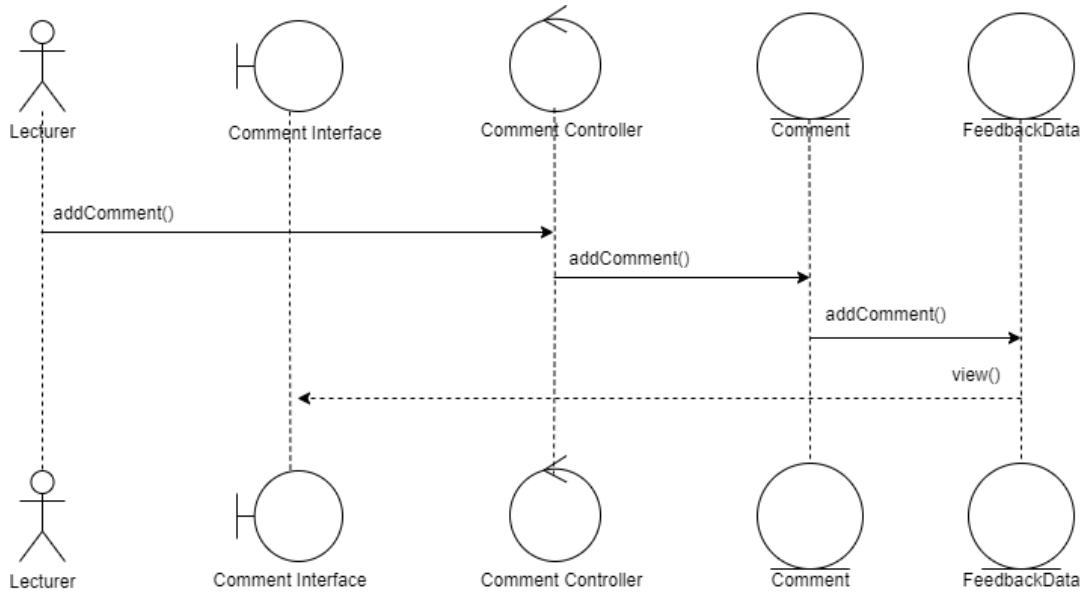


Figure 4.2.3.2.1 Sequence Diagram for <Add Comment>

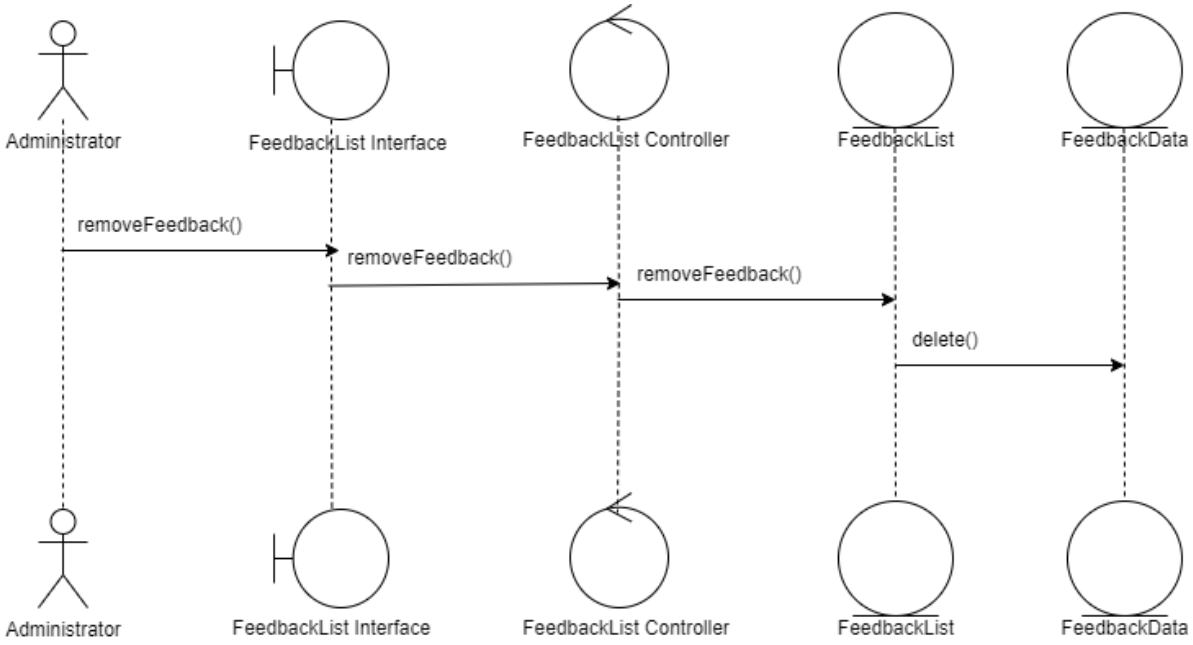


Figure 4.2.3.2.2 Sequence Diagram for <Remove Feedback>

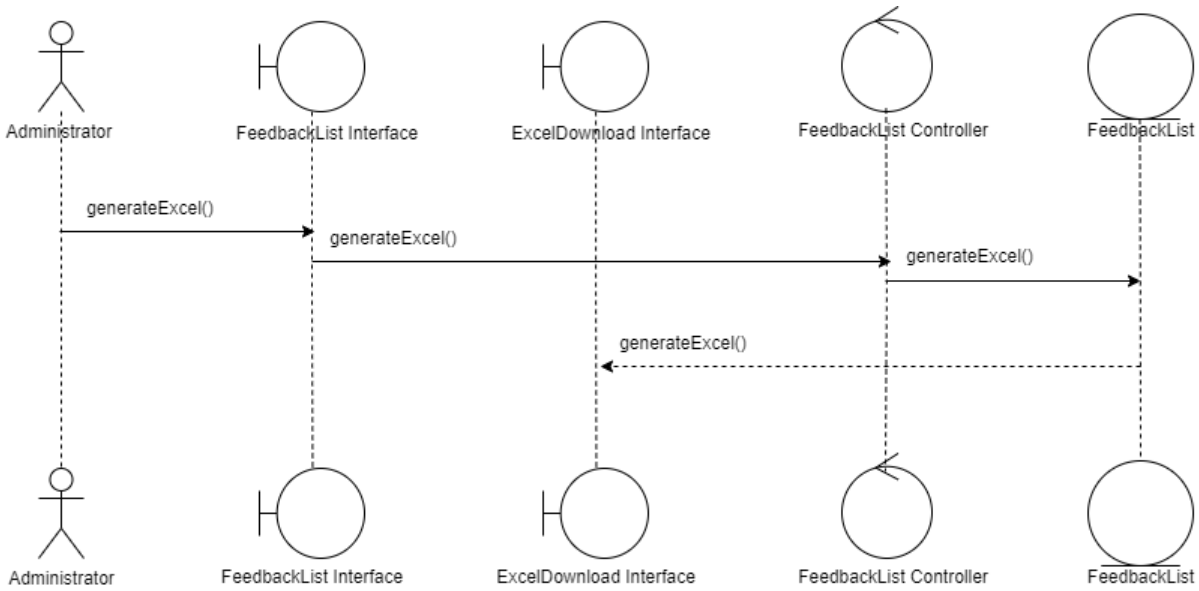


Figure 4.2.3.2.3 Sequence Diagram for <Download Excel>

4.2.4 **P004: <Data Processing and Analysis Module> Subsystem**

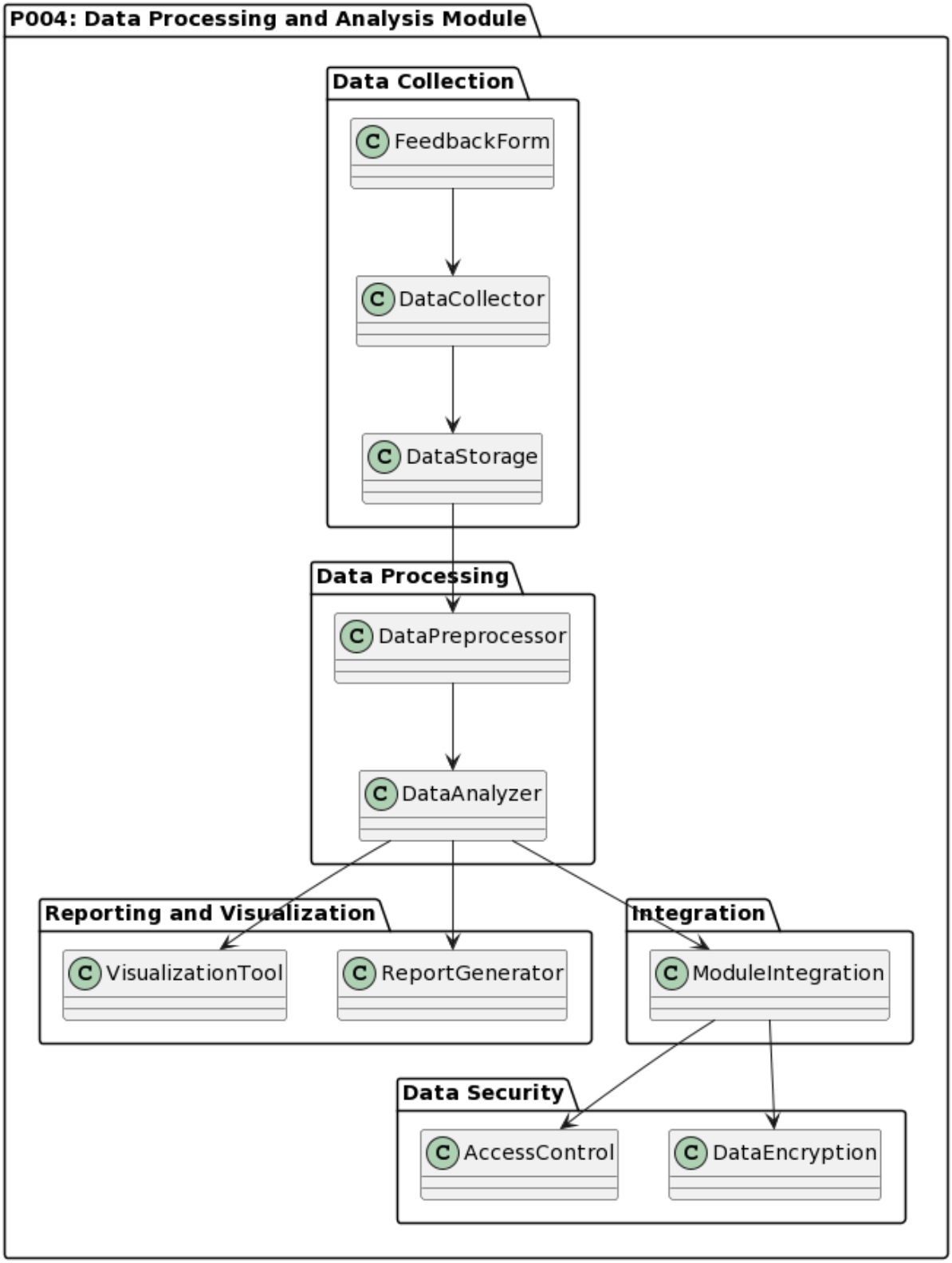


Figure 4.2.4 Package Diagram P004: <Data Processing and Analysis Module> Subsystem

4.2.4.1 **Class Diagram**

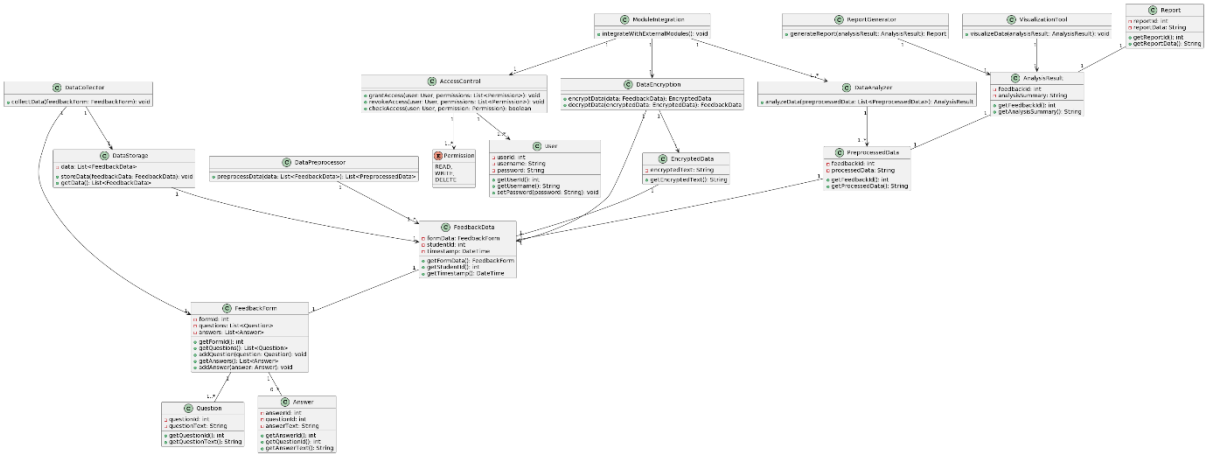


Figure 4.2.4.1 Class Diagram P004: <Data Processing and Analysis Module> Subsystem

Entity Name	FeedbackForm
Method Name	addQuestion
Input	question (Question)
Output	void
Algorithm	1. Add the question to the list of questions in the FeedbackForm entity.

Entity Name	DataCollector
Method Name	collectData
Input	feedbackForm (FeedbackForm)
Output	void
Algorithm	1. Collect the data from the feedbackForm entity. 2. Store the collected data in the DataStorage entity.

Entity Name	DataStorage
Method Name	storeData
Input	feedbackData (FeedbackData)
Output	void
Algorithm	1. Store the feedbackData in the DataStorage entity.

Entity Name	DataPreprocessor
Method Name	preprocessData
Input	data (List<FeedbackData>)
Output	List<PreprocessedData>
Algorithm	1. Iterate over each feedbackData in the data input. 2.Preprocess the feedbackData and generate the corresponding preprocessedData. 3. Add the preprocessedData to the list of preprocessed data. Return the list of preprocessed data.

Entity Name	DataAnalyzer
Method Name	analyzeData
Input	preprocessedData (List<PreprocessedData>)
Output	AnalysisResult
Algorithm	1. Analyze the preprocessedData and generate an analysis summary. 2. Create a new AnalysisResult entity with the analysis summary. 3. Return the AnalysisResult entity.

Entity Name	ReportGenerator
Method Name	generateReport
Input	analysisResult (AnalysisResult)
Output	Report
Algorithm	1. Generate a report based on the analysisResult. 2. Create a new Report entity with the report data. 3. Return the Report entity.

4.2.4.2 **Sequence Diagram**

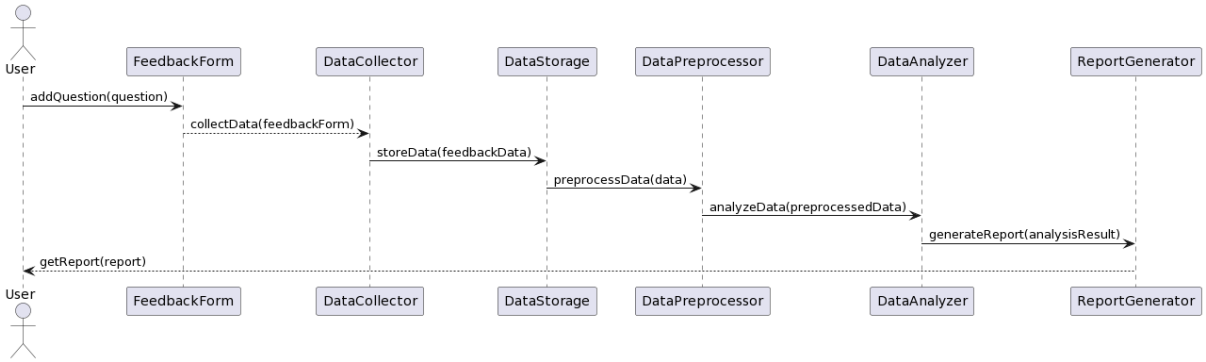


Figure 4.2.4.2 Sequence Diagram P004: <Data Processing and Analysis Module> Subsystem

4.2.5 **P005: <Reporting Module> Subsystem**

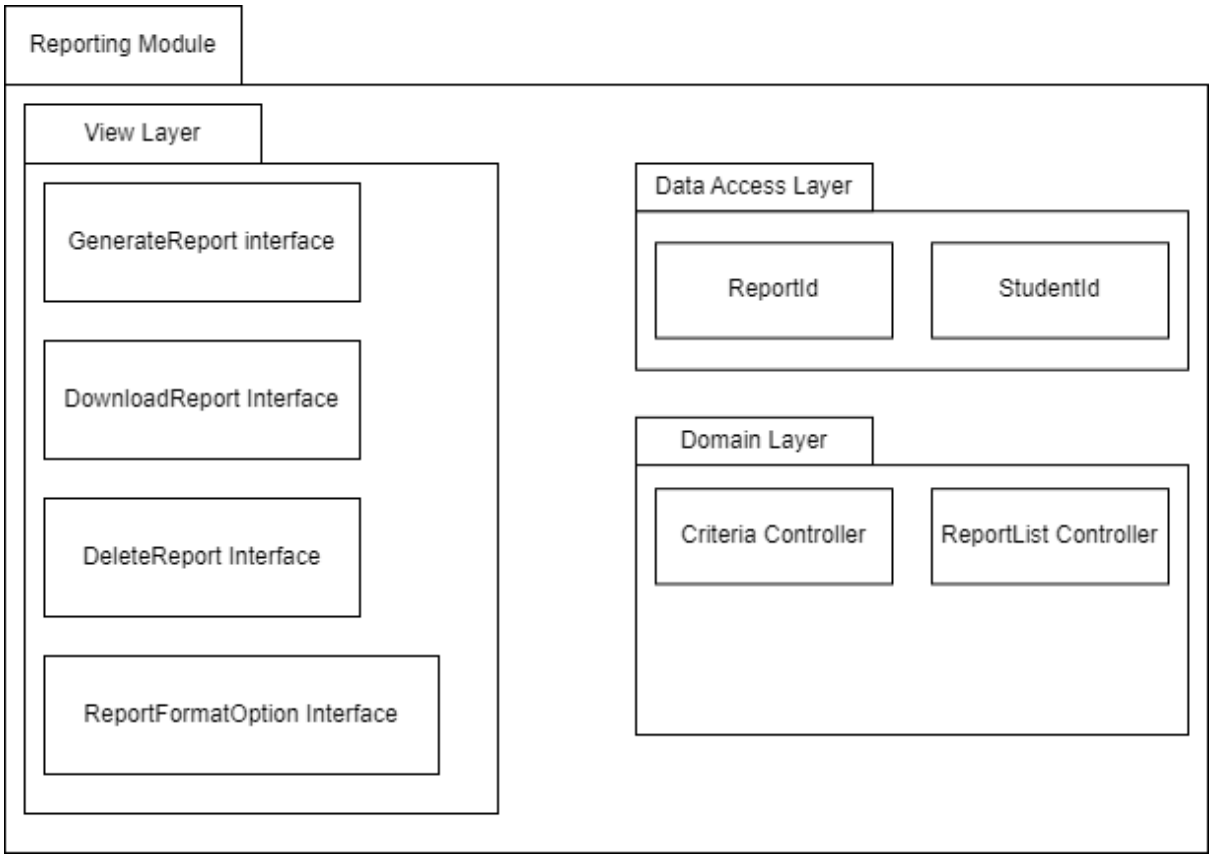


Figure 4.2.5: Package Diagram for <Reporting Module> Subsystem

4.2.5.1 **Class Diagram**

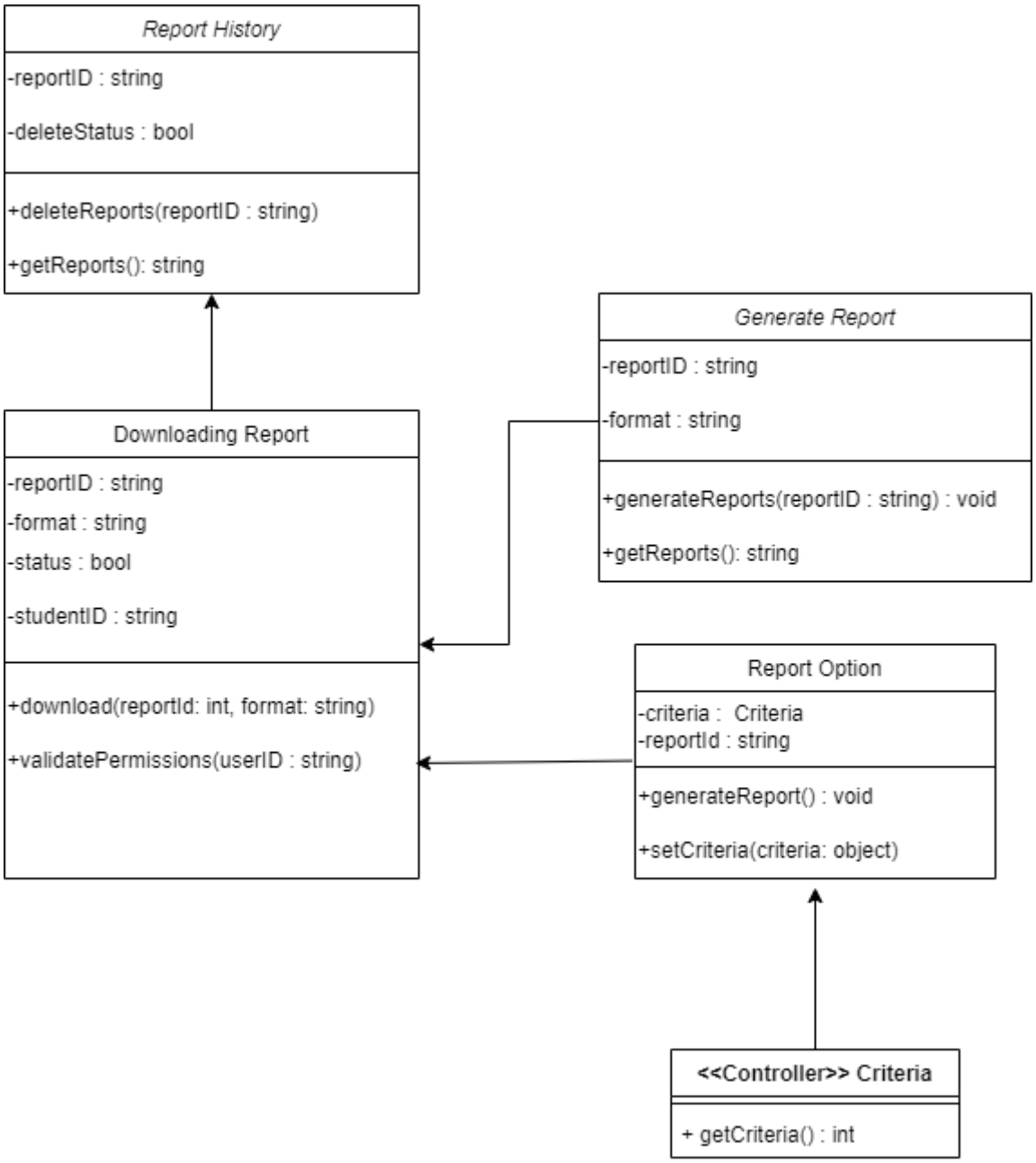


Figure 4.2.5.1: Class Diagram for <Reporting Module> Subsystem

Entity Name	Downloading Report
Method Name	download
Input	reportId,format
Output	report

Algorithm	1. Start 2. Read reportId 3. Read format 4. Get report 5. End

Entity Name	Downloading Report
Method Name	validatePermissions
Input	userID
Output	status
Algorithm	1. Start 2. Read userID 3. If userID is true 3.1. Get status equal to true 4. If userID is false 4.1. End 5. End

Entity Name	Report History
Method Name	deleteReports
Input	reportId
Output	deleteStatus
Algorithm	1. Start 2. Read reportId 3. Get deleteStatus 4. End

Entity Name	Report History
Method Name	getReports
Input	reportId
Output	report
Algorithm	1. Start 2. Read reprotId 3. Get report 4. End

Entity Name	Report Option
Method Name	serCriteria
Input	criteria
Output	-
Algorithm	1. Start 2. Read criteria 3. End

Entity Name	Report Option
Method Name	generateReport
Input	criteria
Output	report,reportId
Algorithm	1. Start 2. Read criteria 3. Get report 4. Get reportId 5. End

Entity Name	Generate Report
Method Name	generateReports
Input	format/criteria
Output	reportId
Algorithm	1. Start 2. Read format or criteria 3. Generate Report 4. Get reportId 5. End

Entity Name	Generate Report
Method Name	getReports
Input	reportId
Output	report
Algorithm	1. Start 2. Read reportId 3. Get report 4. End

4.2.5.2 **Sequence Diagram**

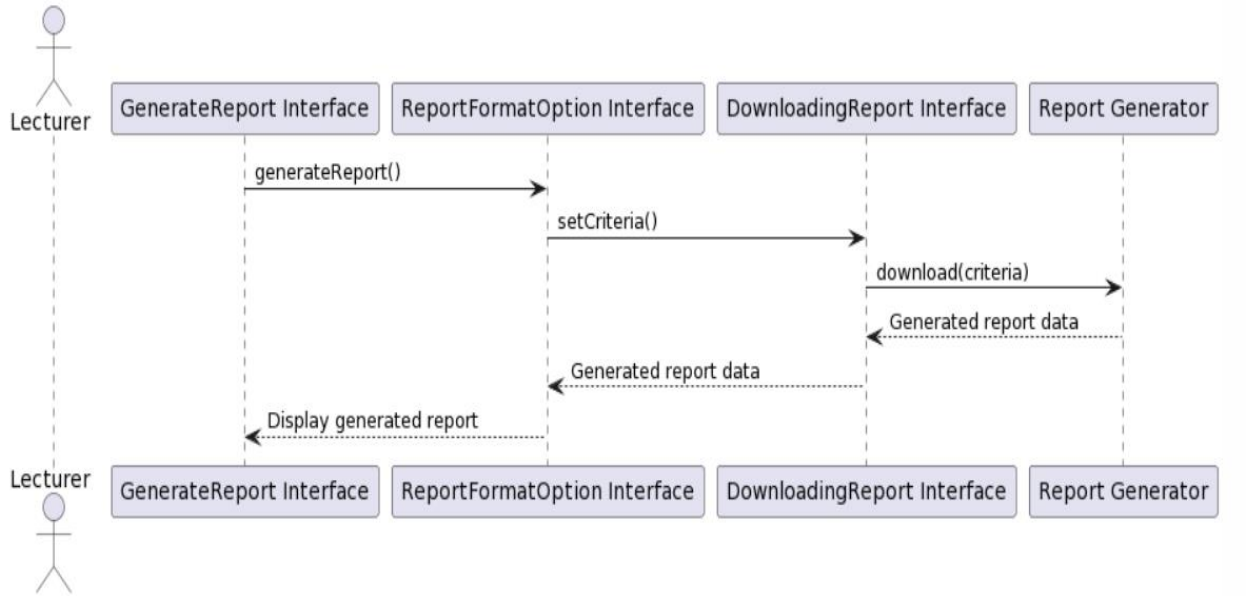


Figure 4.2.5.2.1: Sequence Diagram for <Generate Report>

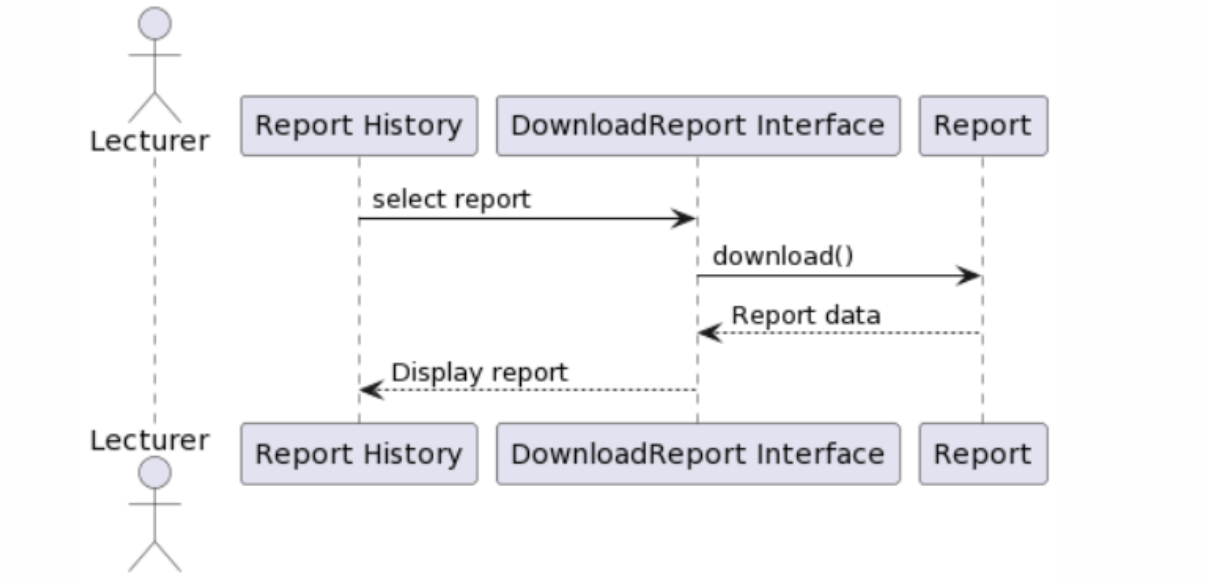


Figure 4.2.5.2.2: Sequence Diagram for <View Report>

4.2.6 **P006: <Integration Module> Subsystem**

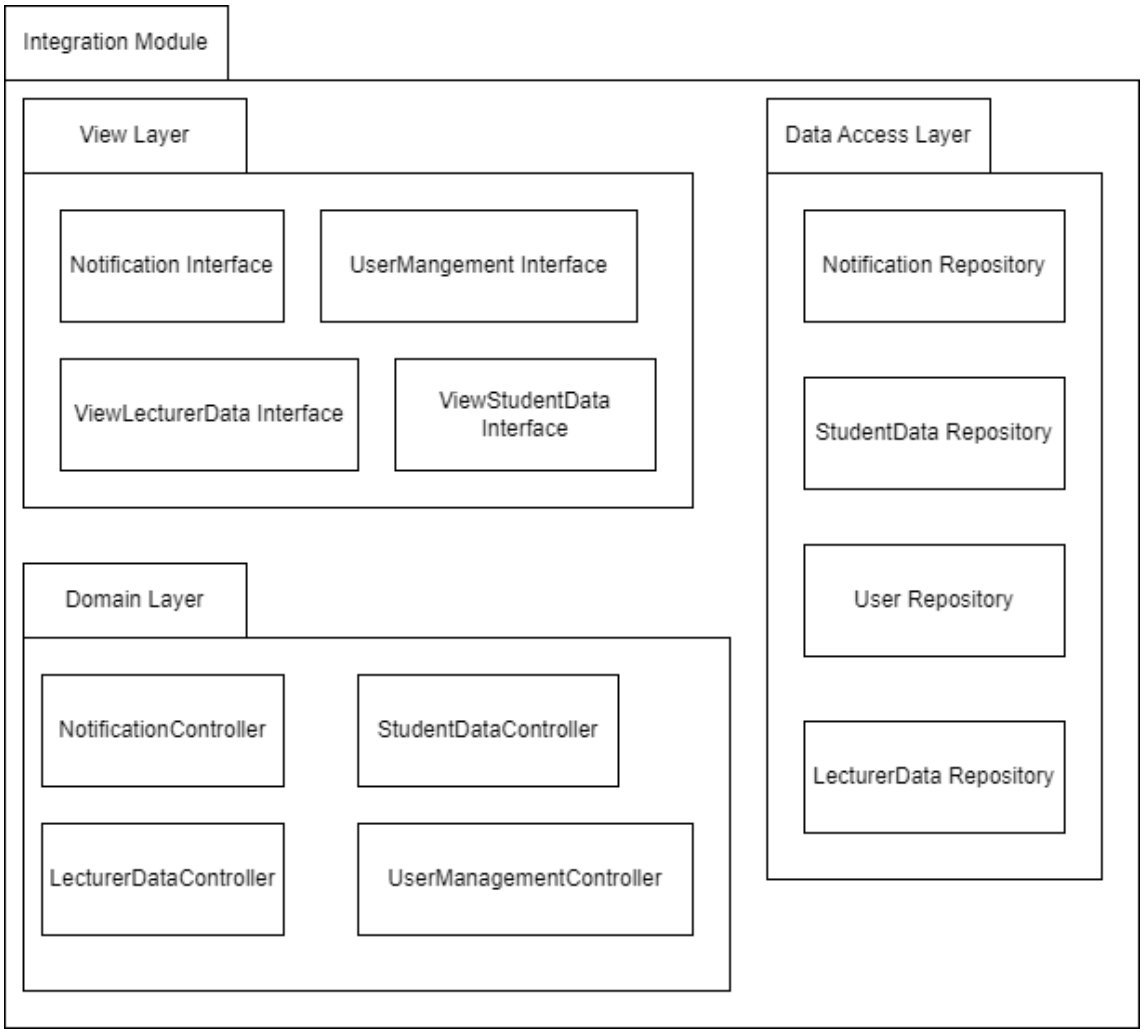


Figure 4.2.6: Package Diagram for <Integration Module> Subsystem

4.2.6.1 **Class Diagram**

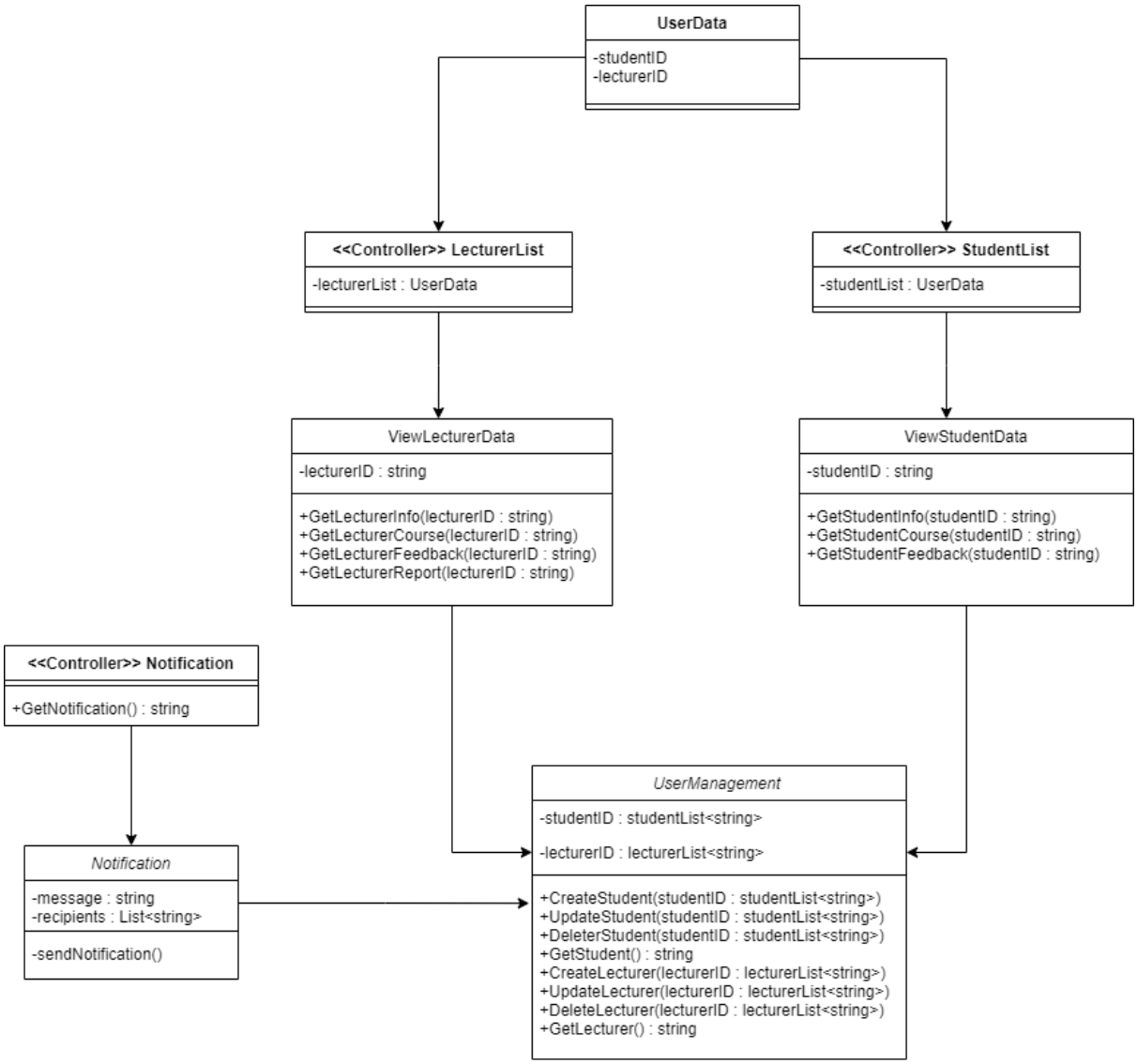


Figure 4.2.6.1: Class Diagram for <Integration Module> Subsystem

Entity Name	Notification
Method Name	sendNotification
Input	recipients
Output	message
Algorithm	1. Start 2. Get recipients 3. Read message 4. Send message 5. End

Entity Name	ViewStudentData
Method Name	GetStudentInfo
Input	studentID
Output	student’s info
Algorithm	1. Start 2. Read studentID 3. Get student’s info 4. End

Entity Name	ViewStudentData
Method Name	GetStudentCourse
Input	studentID
Output	student’s course
Algorithm	1. Start 2. Read studentID 3. Get student’s course 4. End

Entity Name	ViewStudentData
Method Name	GetStudentFeedback
Input	studentID
Output	student’s feedbacks
Algorithm	1. Start 2. Read studentID 3. Get student’s feedbacks 4. End

Entity Name	ViewLecturerData
Method Name	GetLecturerInfo
Input	lecturerID
Output	lecturer’s info

Algorithm	1. Start 2. Read lecturerID 3. Get lecturer's info 4. End

Entity Name	ViewLecturerData
Method Name	GetLecturerCourse
Input	lecturerID
Output	lecturer's course
Algorithm	1. Start 2. Read lecturerID 3. Get lecturer's course 4. End

Entity Name	ViewLecturerData
Method Name	GetLecturerFeedback
Input	lecturerID
Output	lecturer's feedback
Algorithm	1. Start 2. Read lecturerID 3. Get lecturer's feedbacks 4. End

Entity Name	ViewLecturerData
Method Name	GetLecturerReport
Input	lecturerID
Output	lecturer's report
Algorithm	1. Start 2. Read lecturerID 3. Get lecturer's report 4. End

Entity Name	UserManagement
Method Name	CreateStudent
Input	student’s info
Output	studentID
Algorithm	1. Start 2. Enter student’s info 3. Create new account 4. Create new studentID 5. End

Entity Name	UserManagement
Method Name	UpdateStudent
Input	studentID
Output	-
Algorithm	1. Start 2. Read studentID 3. Update student’s info 4. Update new studentID 5. End

Entity Name	UserManagement
Method Name	DeleteStudent
Input	studentID
Output	-
Algorithm	1. Start 2. Read studentID 3. Delete student account 4. Update new student list 5. End

Entity Name	UserManagement
Method Name	CreateLecturer
Input	lecturer’s info
Output	lecturerID
Algorithm	1. Start 2. Read lecturerID 3. Create new lecturer account 4. Create new lecturerID 5. End

Entity Name	UserManagement
Method Name	UpdateLecturer
Input	lecturerID
Output	-
Algorithm	1. Start 2. Read studentID 3. Update student’s info 4. Update new studentID 5. End

Entity Name	UserManagement
Method Name	DeleteLecturer
Input	lecturerID
Output	-
Algorithm	1. Start 2. Read studentID 3. Delete student account 4. Update new student list 5. End

4.2.6.2 **Sequence Diagram**

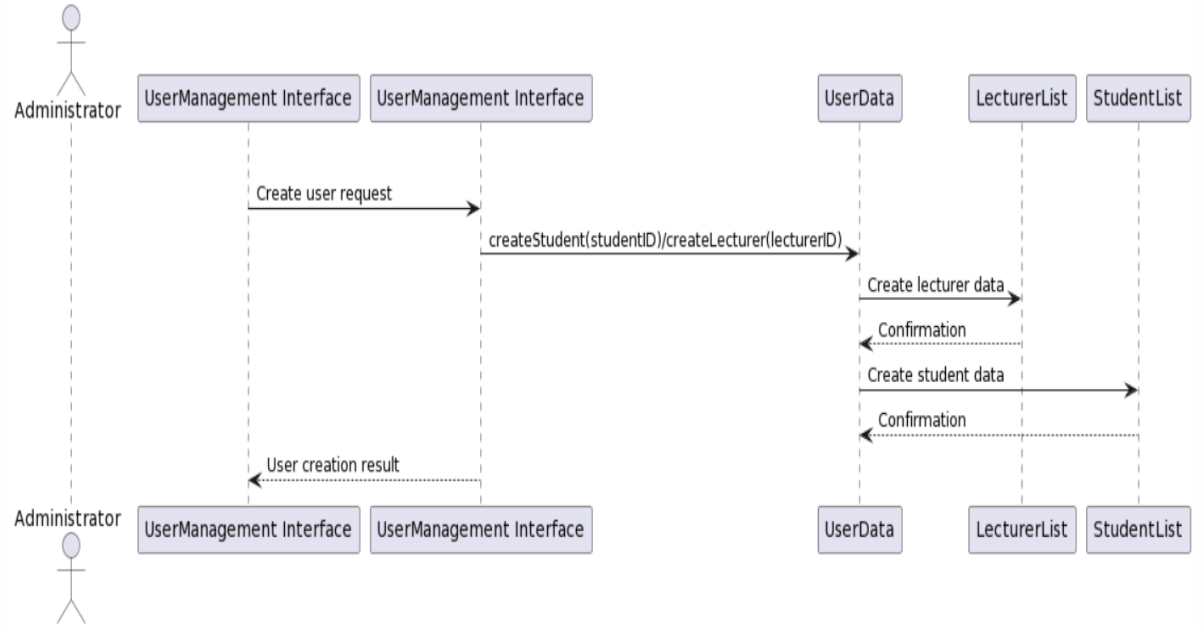


Figure 4.2.6.2.1: Sequence Diagram for <UserManagement>

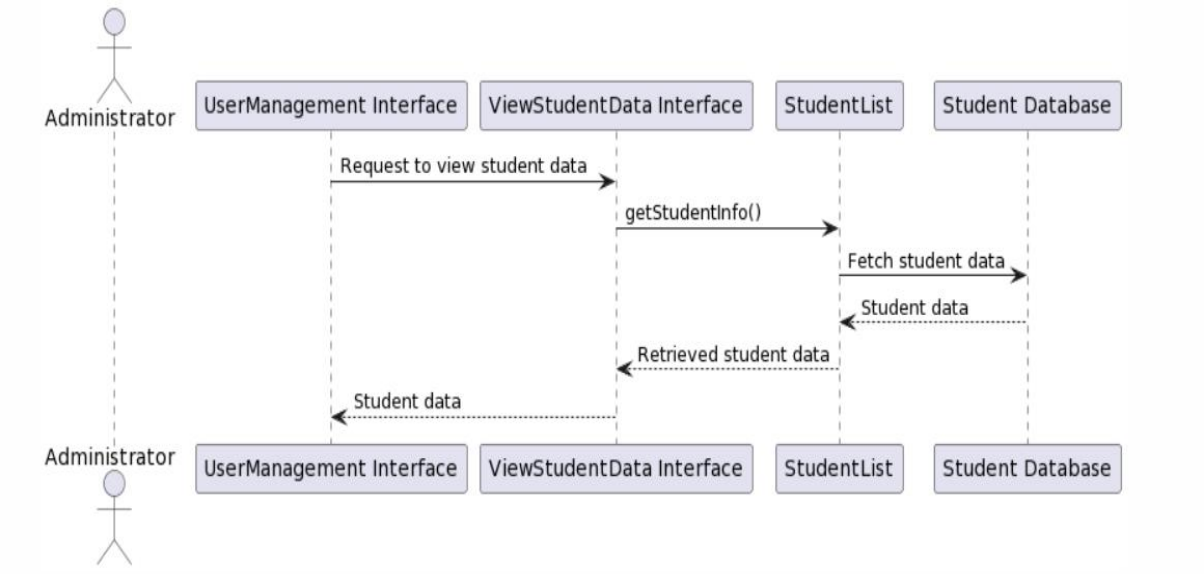


Figure 4.2.6.2.2: Sequence Diagram for <ViewStudentData>

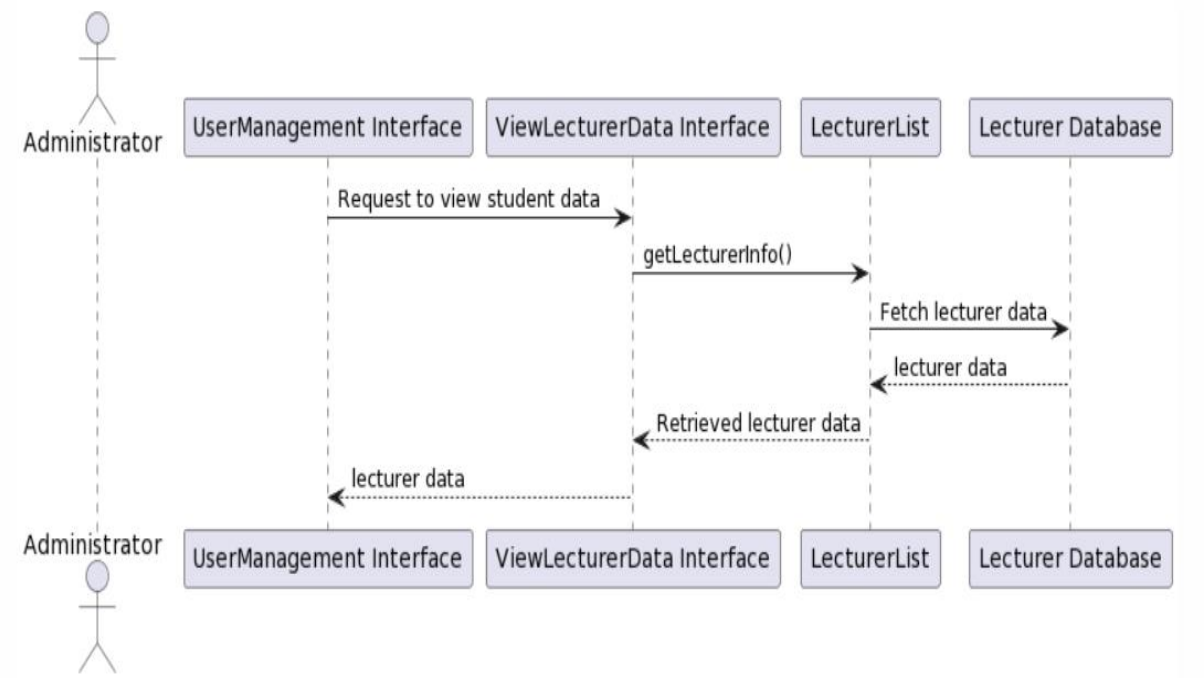
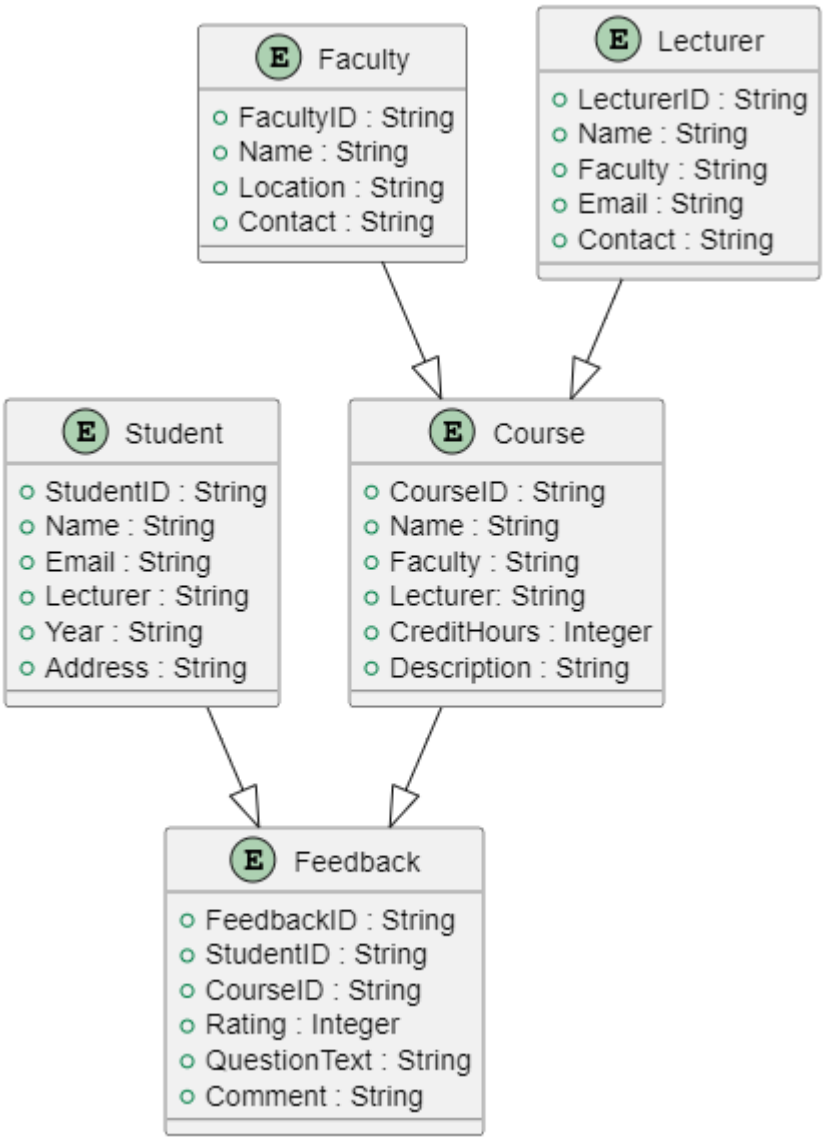


Figure 4.2.6.2.3: Sequence Diagram for <ViewLecturerData>

5. Data Design

5.1 **Data Description**



The major data or systems entities are stored into a relational database named as db_feedback processed and organized into *5* entities as listed in Table 5.1.

Table 5.1: Description of Entities in the Database

No.	Entity Name	Description
1	Student	Stores information about the students enrolled in the University Technology Malaysia (UTM) and their feedback.

2	Course	Contains details about the courses offered at UTM.
3	Faculty	Stores information about the faculties or departments within UTM.
4	Feedback	Stores the feedback submitted by students for various courses.
5	Lecturer	Contains details about the lecturers who teach the courses at UTM.

5.2 **Data Dictionary**

Entity: <Student>

Attribute Name	**Type**	**Description**
StudentID	String	Unique identifier for each student.
Name	String	Name of the student.
Email	String	UTM email address of the student.
Faculty	String	Name of the faculty to which the student belongs.
Year	String	Year of admission for the student.
Address	String	Residential address of the student.

Entity: <Course>

Attribute Name	**Type**	**Description**
CourseID	String	Unique identifier for each course.
Name	String	Name of the course.
Faculty	String	Name of the faculty offering the course.
Lecturer	String	Name of the lecturer teaching the course.
CreditHours	Integer	Number of credit hours assigned to the course.
Description	String	Brief description of the course.

Entity: <Faculty>

Attribute Name	Type	Description
FacultyID	String	Unique identifier for each faculty.
Name	String	Name of the faculty.
Location	String	Location or address of the faculty.
Contact	String	Contact number of the faculty.

Entity: <Feedback>

Attribute Name	Type	Description
FeedbackID	String	Unique identifier for each feedback.
StudentID	String	Identifier of the student who submitted the feedback.
CourseID	String	Identifier of the course for which the feedback is submitted.
Rating	Integer	Rating given by the student for the course. (Scale 1-5)
QuestionText	String	Question texts given to the students that are doing the feedback.
Comment	String	Comments or additional feedback provided by the student.

Entity: <Lecturer>

Attribute Name	Type	Description
LecturerID	String	Unique identifier for each lecturer.
Name	String	Name of the lecturer.
Faculty	String	Name of the faculty to which the lecturer belongs.
Email	Integer	Email address of the lecturer
Contact	String	Contact number of the lecturer

6. User Interface Design

6.1 **Overview of User Interface**

The interface includes a general section of the user interface includes essential components such as the login page, menu, reset password feature, enter code functionality, and new password setting. These elements ensure that users can securely access the system, navigate through different sections, and manage their account information effectively.

The student section focuses on providing a seamless user experience for students. The user page serves as a central hub, offering a holistic view of relevant information and options available to students. Within this section, students can access their user profile to view. The lecturer list feature enables students to choose the respective lecturer they wish to provide feedback on, while the submit new feedback functionality allows them to fill out and submit feedback forms. The system guides students through the process, providing a clear interface for submitting feedback and including options for additional comments. The feedback history component allows students to review their previously submitted feedback, facilitating transparency and accountability. Moreover, the notification list keeps students informed of important updates and messages related to their feedback.

For lecturers, the user interface design focuses on efficiency and convenience. The lecturer page serves as a dedicated area for lecturers, providing access to relevant information and functionalities. Lecturers can view their profile information, ensuring accurate and up-to-date records. The view feedback feature allows lecturers to access and review the feedback submitted by students, facilitating effective evaluation and analysis. Additionally, the lecturer reply functionality enables lecturers to respond to student feedback, fostering communication and addressing concerns. The generate report feature empowers lecturers to generate comprehensive reports based on the feedback received. Lecturers can then download these reports for further analysis and decision-making. The history report component provides a record of generated reports, ensuring easy access to past data. Similar to the student section, the notification list keeps lecturers informed about relevant notifications and updates.

Lastly, the admin section focuses on administrative tasks and user management. The admin page provides access to administrative functions and options. Admins can manage user accounts, including students and lecturers, through features such as managing user pages and performing account deletions. The view and edit user data components enable admins to access and modify user information as needed, ensuring accurate records and efficient administration. The notification list keeps admins informed of system updates and important messages.

6.2 Screen Images

6.2.1 General

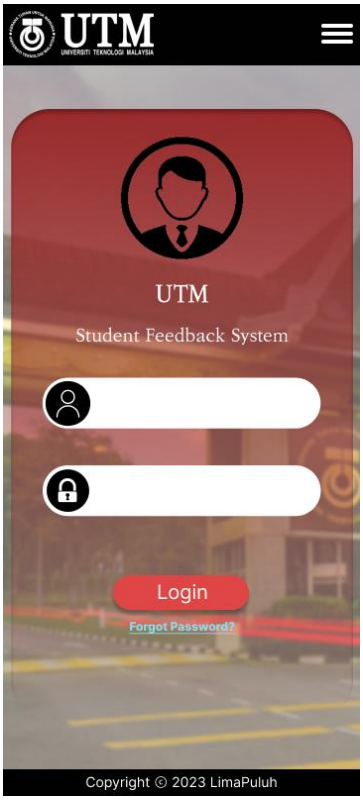


Figure 6.2.1.1 Login Page

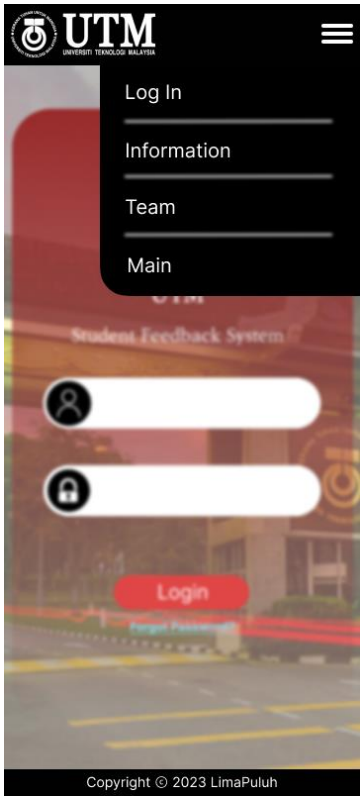


Figure 6.2.1.2 Menu Page

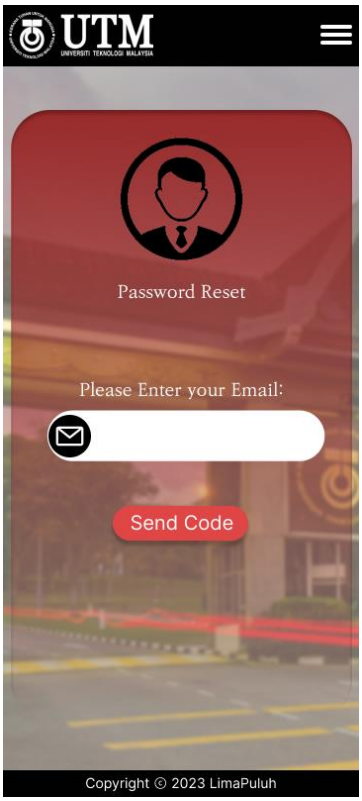


Figure 6.2.1.3 Reset Password Page

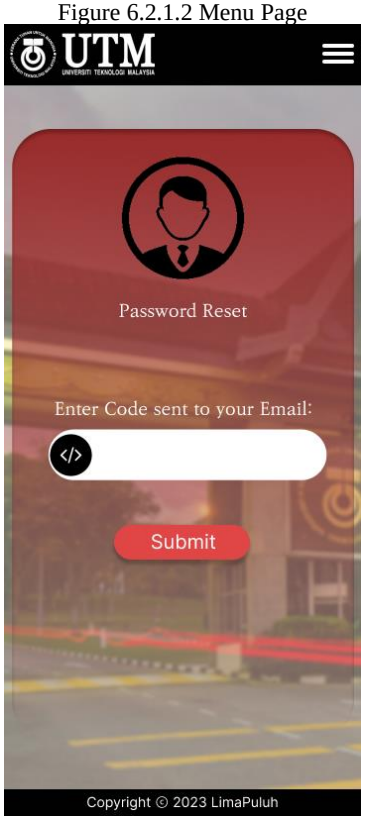


Figure 6.2.1.4 Enter Code Page

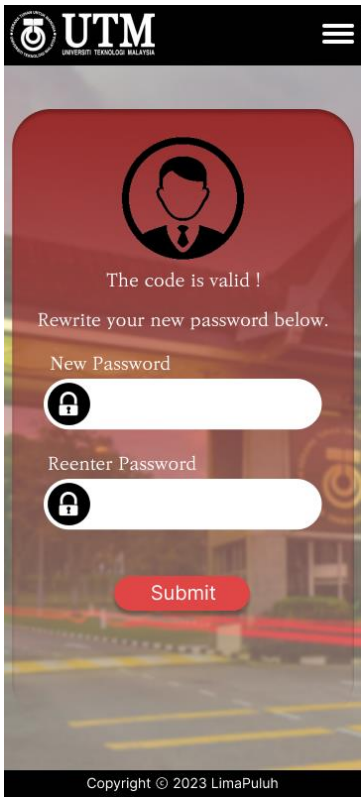


Figure 6.2.1.5 New Password Page

6.2.2 Student

Figure 6.2.2.1 User Page

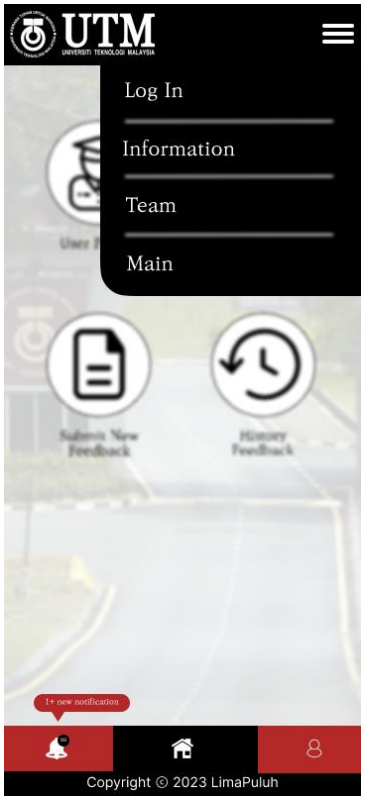


Figure 6.2.2.2 User Page (Menu)

Figure 6.2.2.3 User Profile Page

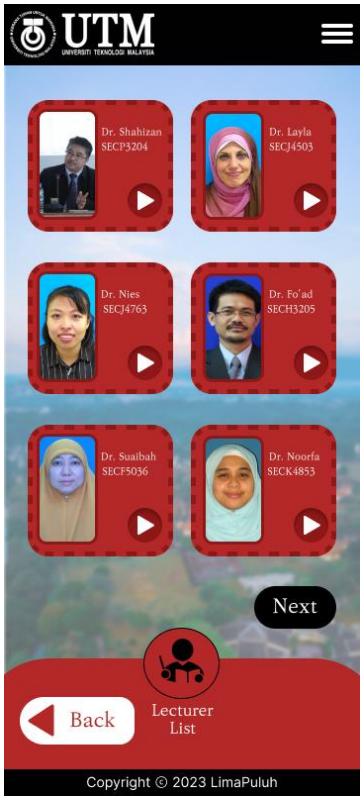


Figure 6.2.2.4 Lecturer List Page

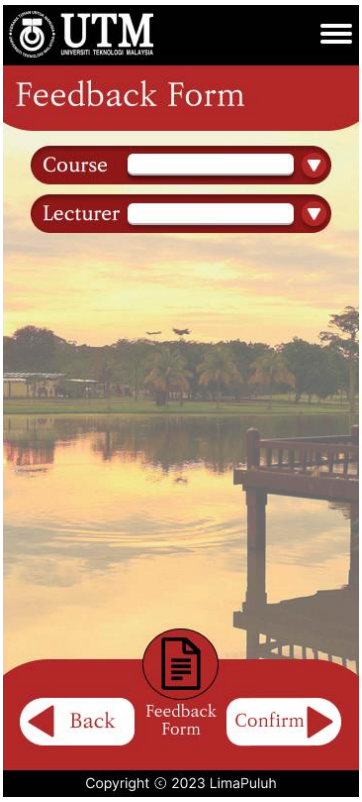


Figure 6.2.2.5 Submit New Feedback Page

Figure 6.2.2.6 Submit New Feedback Page (Selected)

Figure 6.2.2.7 Submit New Feedback Page (Completed)

Figure 6.2.2.8 Feedback Comment Page

Figure 6.2.2.9 Feedback Comment Page (View)

Figure 6.2.2.10 Feedback History Page

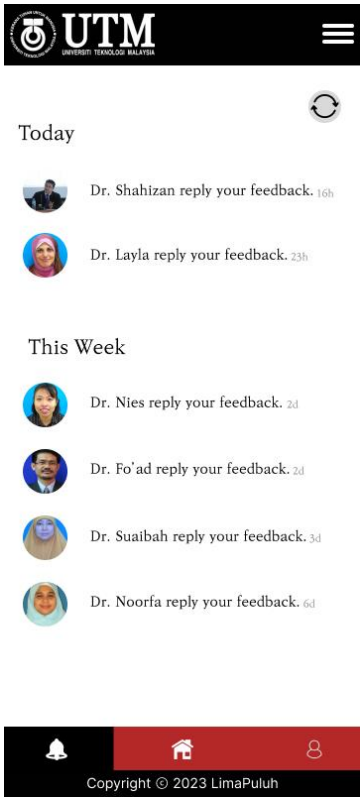


Figure 6.2.2.11 Notification List (student) Page

6.2.3 Lecturer

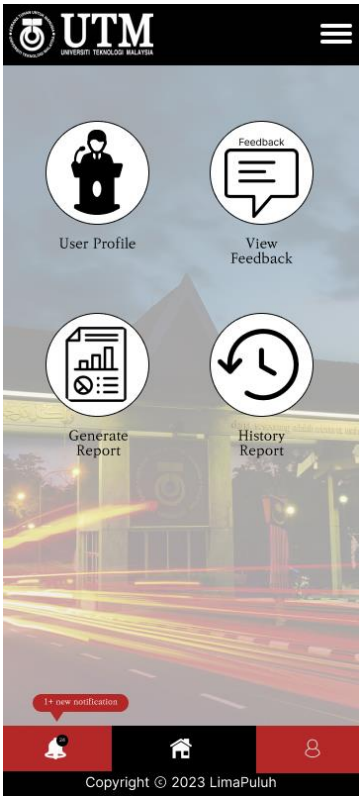


Figure 6.2.3.1 Lecturer Page

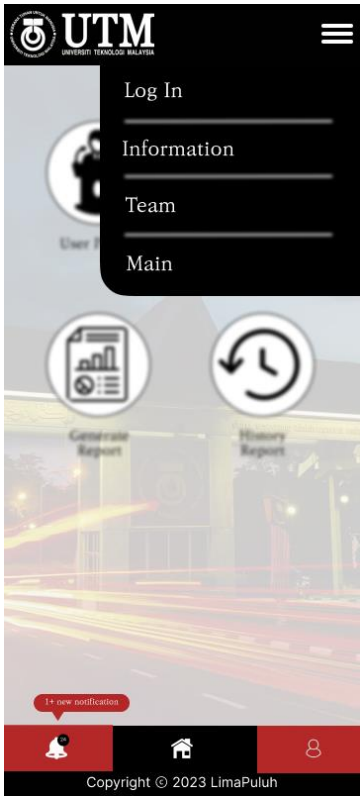


Figure 6.2.3.2 Lecturer Page (Menu)

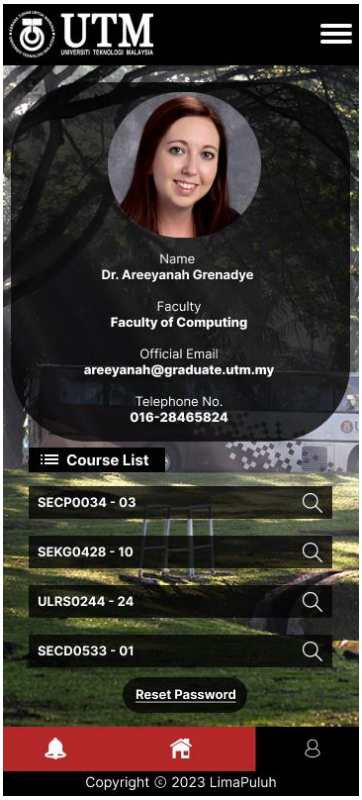


Figure 6.2.3.3 Lecturer Profile Page

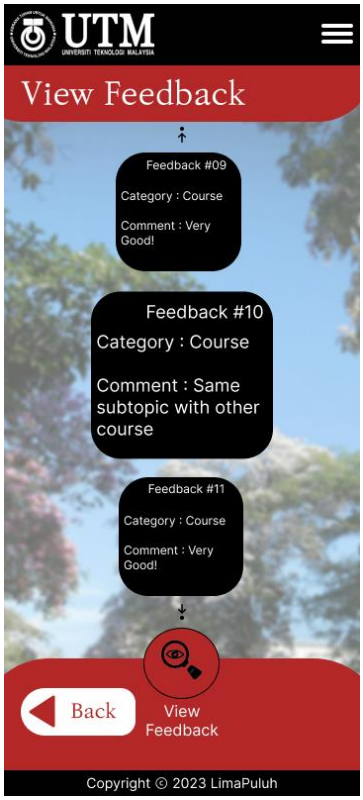


Figure 6.2.3.4 View Feedback Page

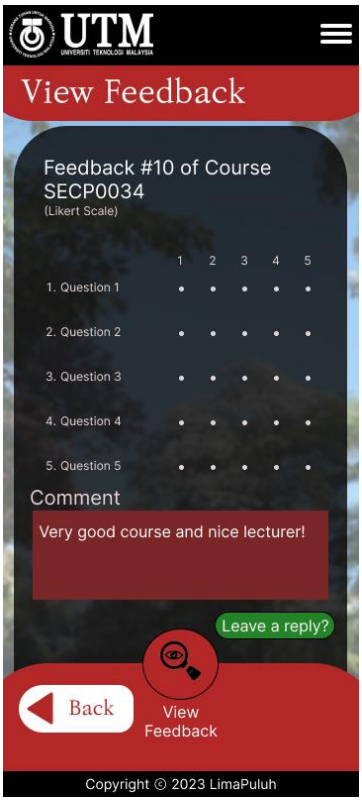


Figure 6.2.3.5 View Feedback Page (Individual)

Figure 6.2.3.6 Lecturer Reply Page

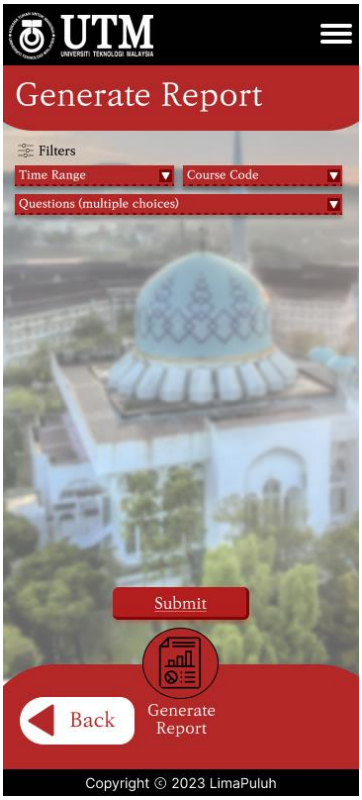


Figure 6.2.3.7 Generate Report Page

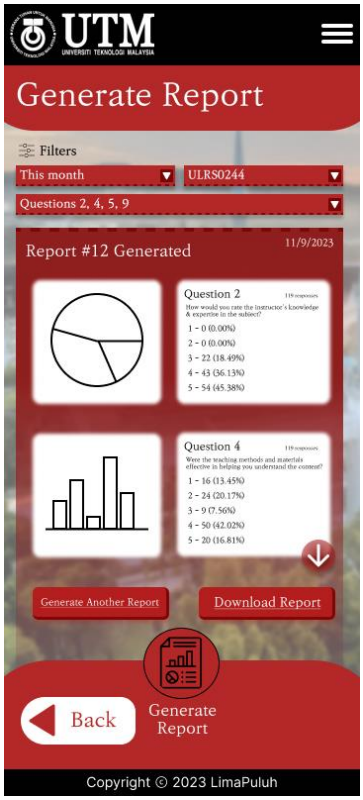


Figure 6.2.3.8 Generate Report Page (Complete)

Figure 6.2.3.9 Download Report Page

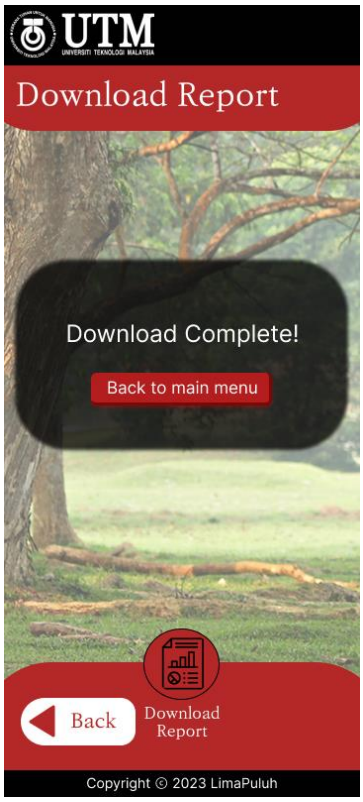


Figure 6.2.3.10 Download Report Page (Complete)

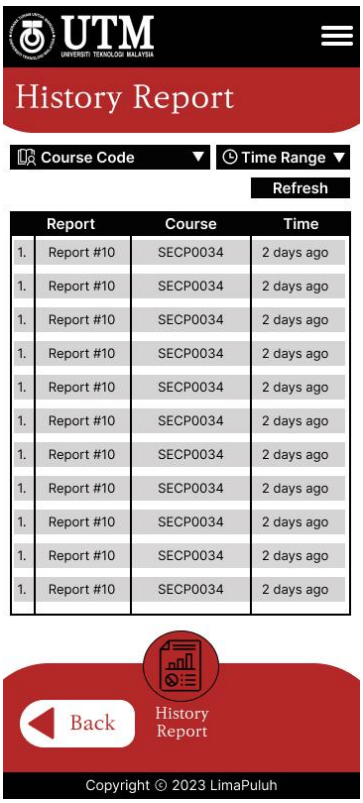


Figure 6.2.3.11 History Report Page

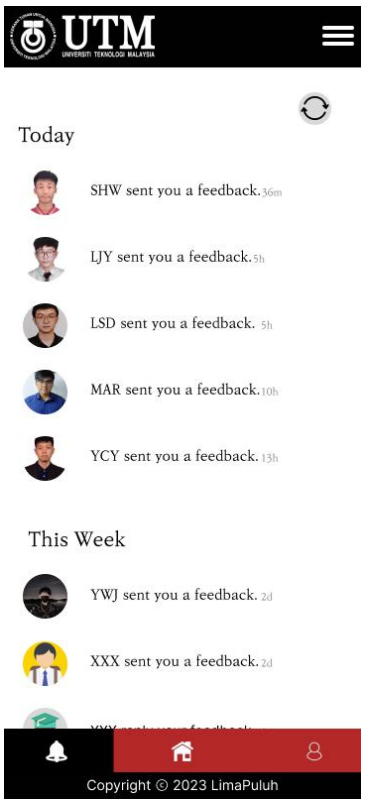


Figure 6.2.3.12 Notification List Page (Lecturer)

6.2.4 Admin

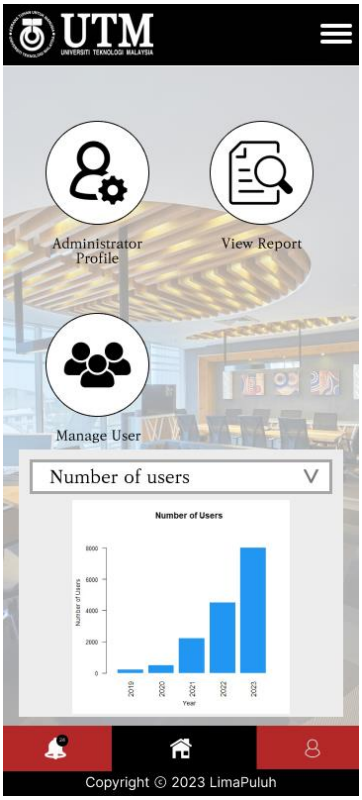


Figure 6.2.4.1 Admin Page

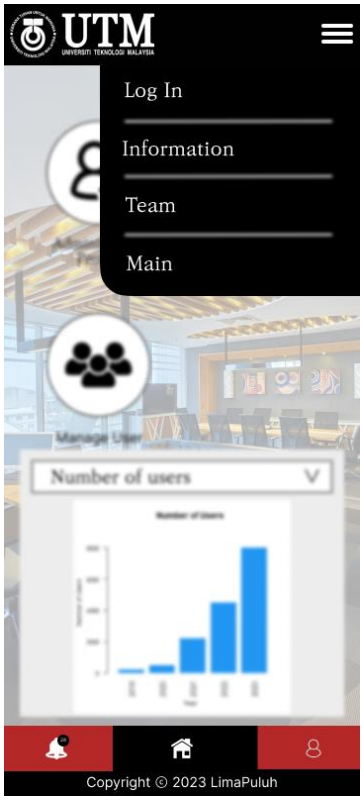


Figure 6.2.4.2 Admin Page (Menu)

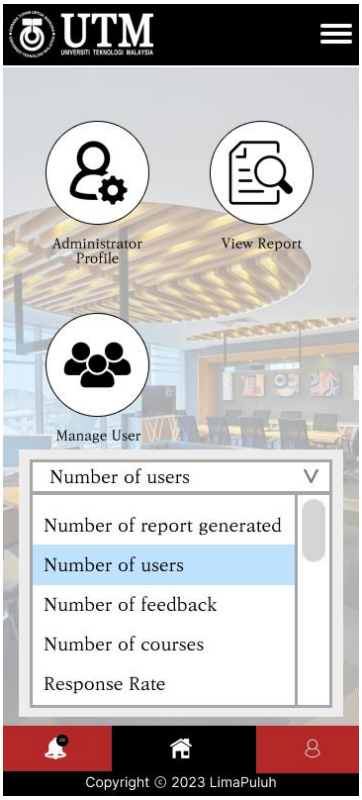


Figure 6.2.4.3 Admin Page (Data)

Figure 6.2.4.4 Admin Profile Page

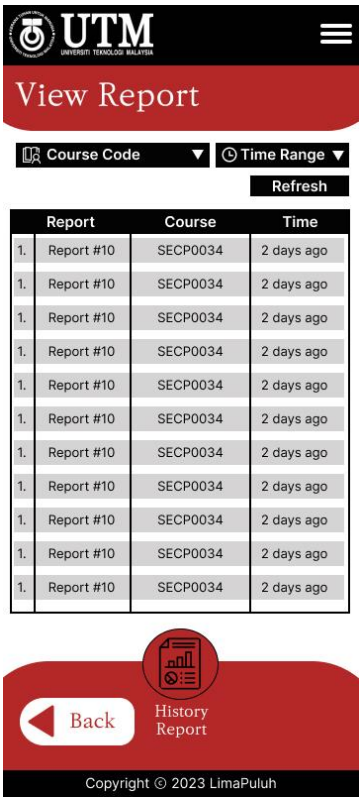


Figure 6.2.4.5 View Report Page

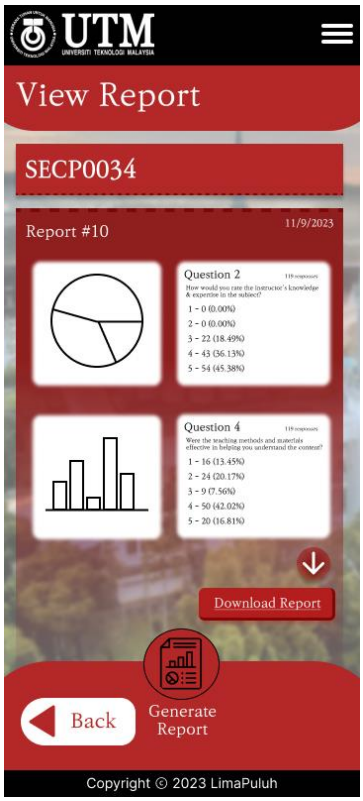


Figure 6.2.4.6 View Report Page (Detail)

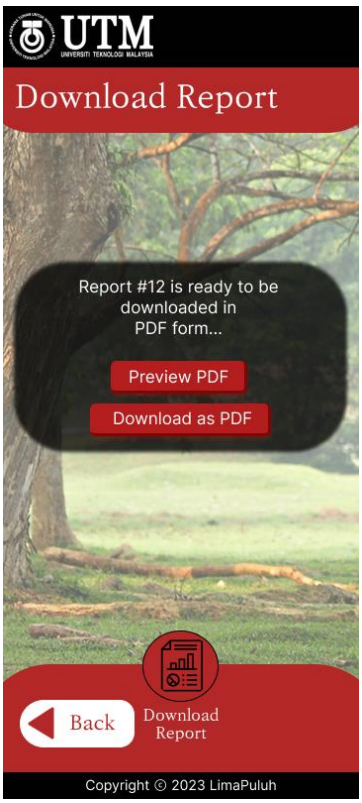


Figure 6.2.4.7 Download Report Page

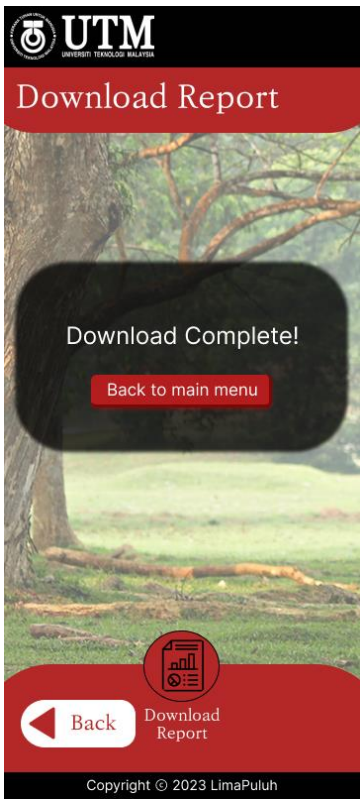


Figure 6.2.4.8 Download Report Page (Complete)

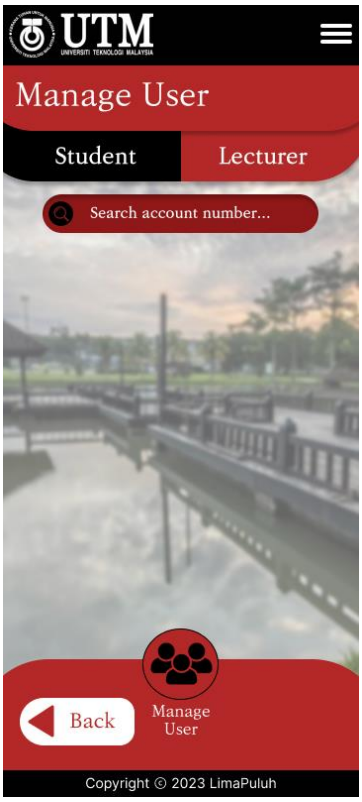


Figure 6.2.4.9 Manage User (Student) Page 1

Figure 6.2.4.10 Manage User (Student) Page 2

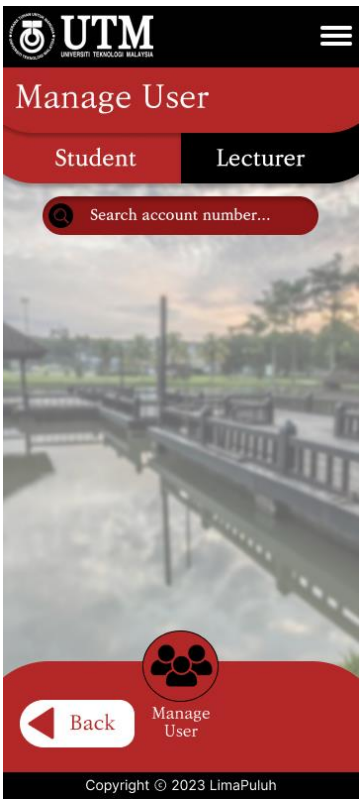


Figure 6.2.4.11 Manage User (Lecturer) Page 1

Figure 6.2.4.12 Manage User (Lecturer) Page 2

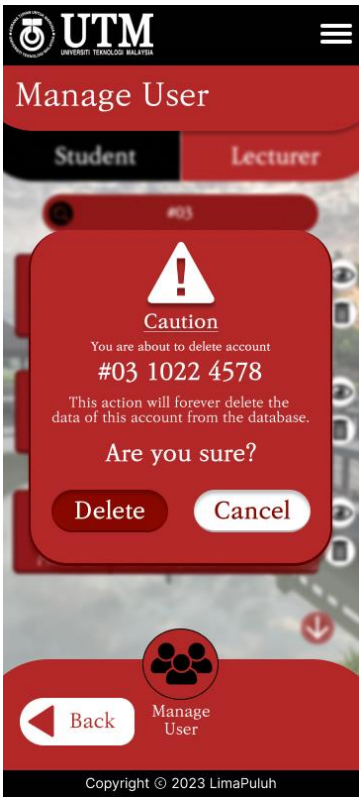


Figure 6.2.4.13 Delete account (Student) Page

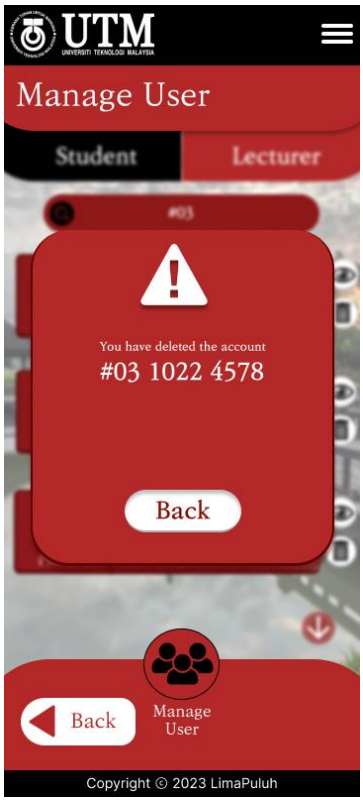


Figure 6.2.4.14 Delete account complete (Student) Page

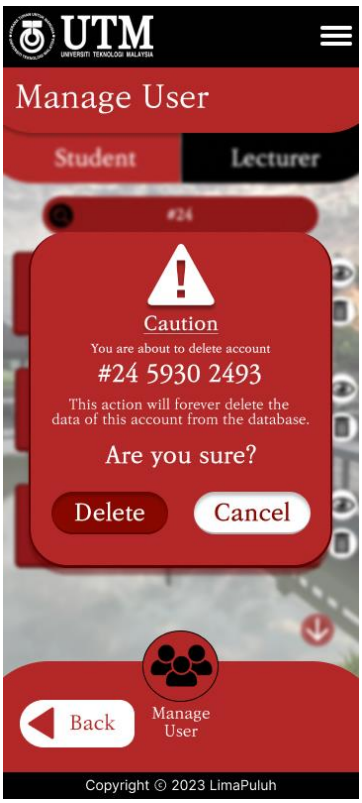


Figure 6.2.4.15 Delete account (Lecturer) Page

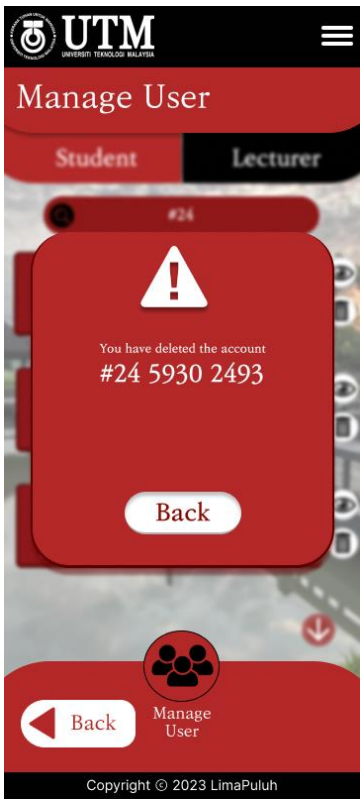


Figure 6.2.4.16 Delete account complete (Lecturer) Page

Figure 6.2.4. 17 View User Data (Student) Page

Figure 6.2.4.18 Edit User Data (Student) Page

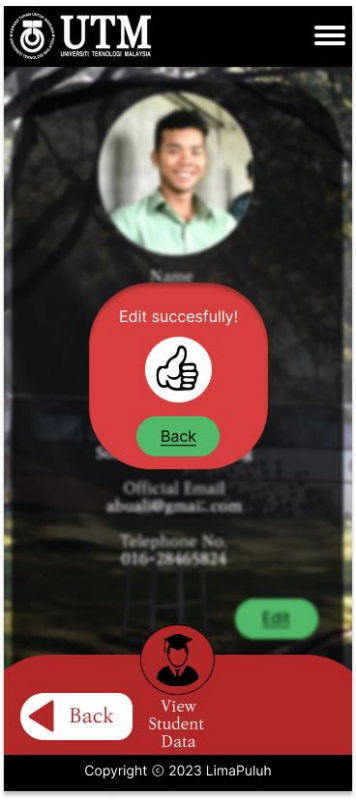


Figure 6.2.4.19 Edit User Data Complete (Student) Page

Figure 6.2.4.20 View User Data (Lecturer) Page

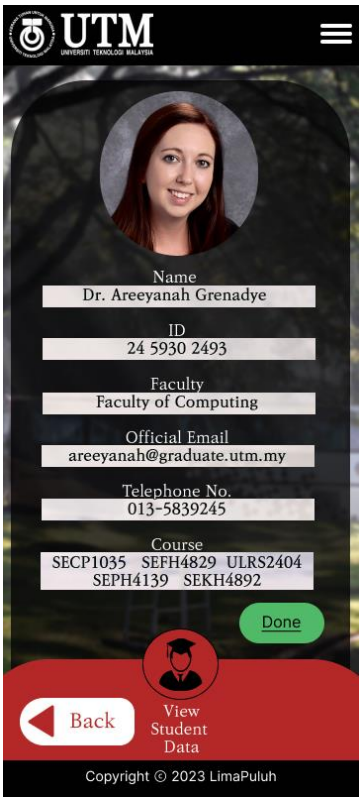


Figure 6.2.4.21 Edit User Data (Lecturer) Page

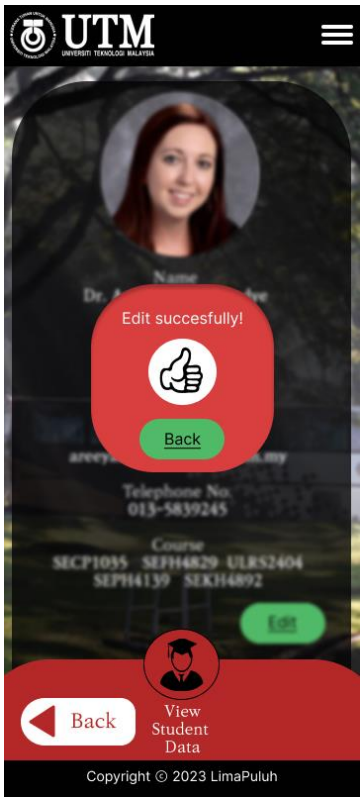


Figure 6.2.4.22 Edit User Data Complete (Lecturer) Page

Figure 6.2.4.23 Notification List (Student) Page

Figure 6.2.4.24 Notification List (Lecturer) Page