UTM

UNIVERSITI TEKNOLOGI MALAYSIA

SECP2613: SYSTEM ANALYSIS AND DESIGN

System Documentation (SD)

Software Design Descriptions (SDD)

Youth Ventures Student Portfolio Management System (StuPort)

Version 2.0

Date: 11 December 2023

Faculty: Faculty of Computing

Prepared by: Explorer

1. Koh Li Hui A22EC0059
2. Koh Su Xuan A22EC0060
3. Lee Yik Hong A21BE0376
4. Vinesh A/L Vijayakumar A22EC0290

Revision Page

a. **Overview**

System Documentation (SD) aims to provide a comprehensive guide for:

I. Understanding, developing and maintaining a system,
II. Facilitating effective communication,
III. Knowledge sharing,
IV. Troubleshooting,
V. Compliance with standards,

ultimately ensuring system stability, reliability and future evolution. It serves as an important resource that fosters collaboration among clients, guiding development and supporting system operations while enabling efficient adaptation and enhancement over time.

Version 2.0 of SD only covers Software Design Descriptions (SDD) of Youth Ventures Student Portfolio Management System (StuPort) which outline the architectural design, component model and data design of the system by providing a comprehensive overview of the system's structure, including its architectural style, rationale behind design choices, detailed descriptions of subsystems, also known as components and the data involved. By organizing this information, the documentation helps stakeholders understand the system's functionality, how its various parts interact and the underlying data schema, facilitating efficient development, maintenance and future enhancements.

b. **Target Audience**

The targeted audience of SD of Youth Ventures Student Portfolio Management System (StuPort) are :

1. Client - Youth Ventures Asia : The SD serves as a reference for our client, Youth Ventures Asia to comprehend StuPort's functionalities, scope, limitations and expected outcomes. It helps in managing expectations and aligning the final product with their requirements.

2. Youth Ventures Asia's Clients - Partners, Organizers, Students, Lecturers etc. : SD provides insights into the functionalities, features and user interactions within StuPort, ensuring that the StuPort system caters to the needs of Youth Ventures Asia's clients by offering a clear understanding of the system's capabilities and benefits.

3. Project Manager : With SD, project manager oversees the development process, understanding the project's scope, managing timelines and ensuring that the project aligns with the established requirements and goals.

4. Designer : Designer relies on SD to understand user requirements, functionalities and constraints by insights into the system's layout, user interface elements and user experience expectations provided to create intuitive and user-friendly interfaces for StuPort.

5. Database Administrator (DBA) : SD outlines the data requirements, storage structures and interactions with the database. It assists DBA in understanding the data models, relationships and constraints necessary for designing, implementing and maintaining the StuPort database efficiently.

6. Quality Assurance (QA) Tester : QA testers utilize SD to create test cases, scenarios and expected outcomes based on specified requirements, helping in validating StuPort's functionalities, ensuring that the software meets quality standards and performs as expected.

7. System Analyst : System Analysts refer to SD to comprehend the system's architecture, functionalities and dependencies. It aids in analyzing system requirements, identifying potential risks and proposing suitable solutions for StuPort.

8. Documentation Specialist : SD serves as a foundational resource in structuring and organizing comprehensive documentation for StuPort. It provides essential details,

terminology and information necessary for creating user manuals, guides and other supplementary documents.

9. Developer : Developers rely on SD for detailed technical specifications, architectural diagrams, coding guidelines and integration requirements. It guides them in implementing StuPort's functionalities while adhering to the defined standards and specifications.

c. **Project Team Members**

Member Name	**Role**	**Task**	**Status**
KOH LI HUI	- Project Manager - Designer	- **P003: Dashboard Subsystem**	Complete
		- **P006: Feedback Subsystem**	
		- Section 3.1 and Section 3.2	
KOH SU XUAN	- Database Administrator (DBA) - Quality Assurance (QA) Tester	- **P002: Profile Subsystem**	Complete
		- **P007: Resume Subsystem**	
		- Section 2.1 and Section 2.2	
LEE YIK HONG	- System Analyst - Documentation Specialist	- **P001: Authentication Subsystem**	Complete
		- **P004: Activity Subsystem**	

		- **P005: Registration Subsystem** - **P009: Personal Activity Subsystem**	
VINESH A/L VIJAYAKUMAR	- Developer	- **P007: Rewards Subsystem** - Section 1.1 and Section 1.2	Complete

d. **Version Control History**

Version	**Primary Author(s)**	**Description of Version**	**Date Completed**
2.0	KOH LI HUI	Completed Chapter 2, Section 2.2.3, Section 2.2.6	29/12/2023
2.0	KOH LI HUI	Completed Chapter 3, Section 3.1, Section 3.2	31/12/2023
2.0	KOH SU XUAN	Completed Chapter 2, Section 2.2.2, Section 2.2.8	15/02/2024
2.0	LEE YIK HONG	Completed Chapter 2, Section 2.2.1, Section 2.2.4	16/02/2024
2.0	LEE YIK HONG	Completed Chapter 2, Section 2.2.5, Section 2.2.9	17/02/2024
2.0	KOH SU XUAN	Completed Chapter 2, Section 2.1, Section 2.2	17/02/2024
2.0	VINESH A/L VIJAYAKUMAR	Completed Chapter 2, Section 2.2.7	17/02/2023
2.0	VINESH A/L VIJAYAKUMAR	Completed Chapter 1, Section 1.1, Section 1.2	17/02/2023

Table of Contents

1. System Architectural Design .

1.1 Architecture Style and Rationale

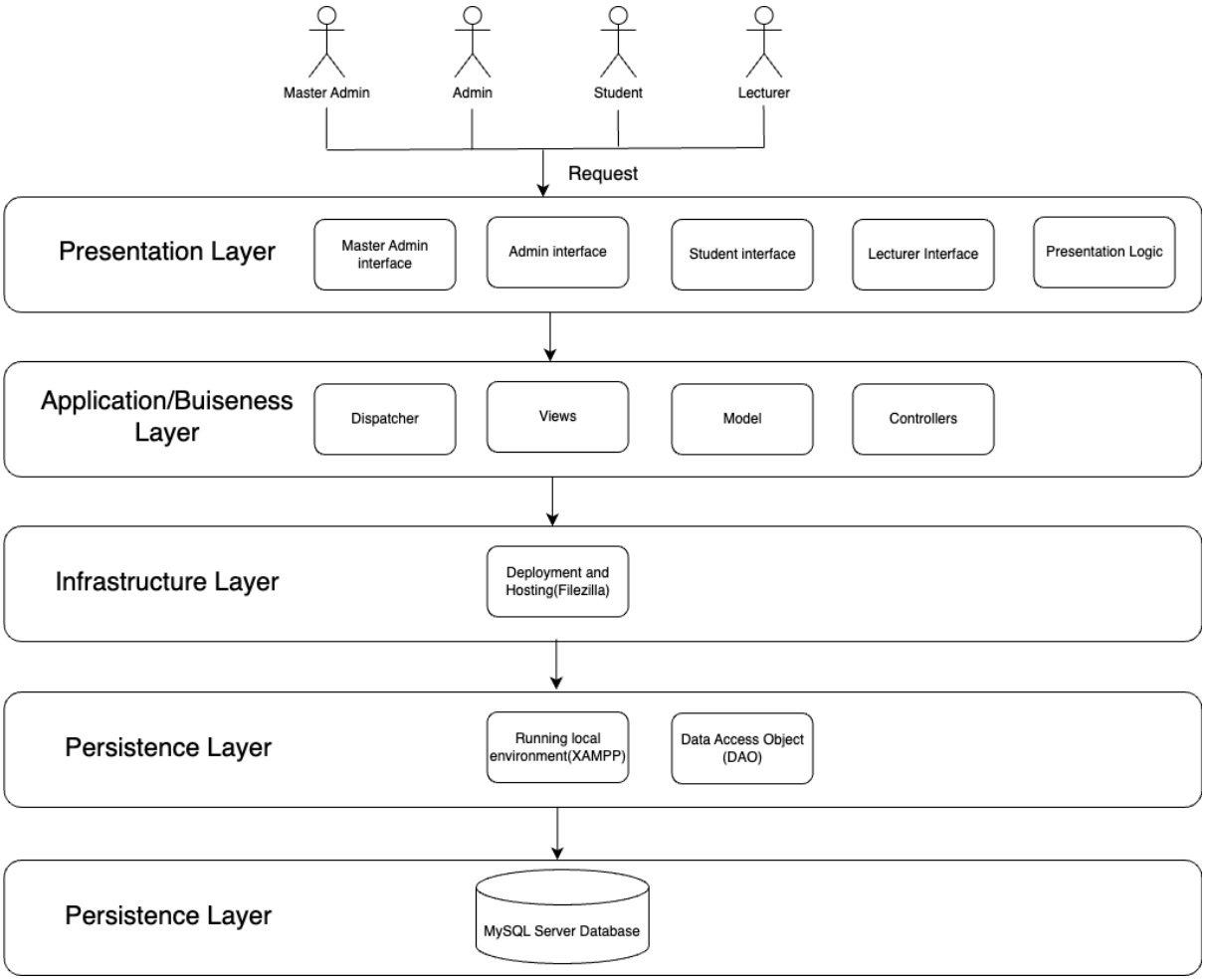

Figure 1.1 : System Architectural Design of StuPort System

The system architecture for Youth Venture involves the interaction of customers, users, system presentation, application, and database, as shown in Figure 1.1. Architecture styles are crucial in software development as they provide a structured approach to designing complex systems. One such style is the Model-View-Controller (MVC) model, which promotes modular and scalable software solutions. This paper focuses on applying MVC to the context of Youth Venture.

MVC divides the application into three components: model, view, and controller, aiming to separate concerns and enhance code organization for improved understanding, development, and maintenance. The model component forms the foundation of the system, encapsulating data and implementing business logic related to managing Youth Venture's operations. It handles operations such as managing customer orders, quotations, and invoices, providing better inventory management alternatives, and ensuring data integrity and consistency. Technologies like relational database management systems (RDBMS) can be used for data storage, and Object-Relational Mapping (ORM) frameworks can simplify interactions between the application and the database.

The view component is crucial in the presentation layer, responsible for presenting data to users and providing a user-friendly interface. It should offer an intuitive and interactive experience to enhance user satisfaction. The controller component acts as an intermediary between the model and the view, handling user input, executing model actions, and updating the view accordingly. This separation enables independent development and testing, contributing to better code maintainability.

Adopting the MVC framework for Youth Venture offers several advantages. It allows for the independent development and testing of each component, improving code maintainability and facilitating future system enhancements. MVC also promotes code organization and reusability, enhancing collaboration among team members and reducing development time. Additionally, MVC aligns with Youth Venture's specific needs, with the model focusing on data operations, the view on user interface design, and the controller on managing user input.

In conclusion, leveraging the MVC architecture in Youth Venture provides benefits such as separation of concerns, code organization, and modularity. These advantages contribute to the system's scalability, maintainability, and extensibility, meeting the

requirements of Youth Venture. Careful consideration of MVC during the architecture design stage establishes a solid foundation for successful system implementation.

1.2 Component Model

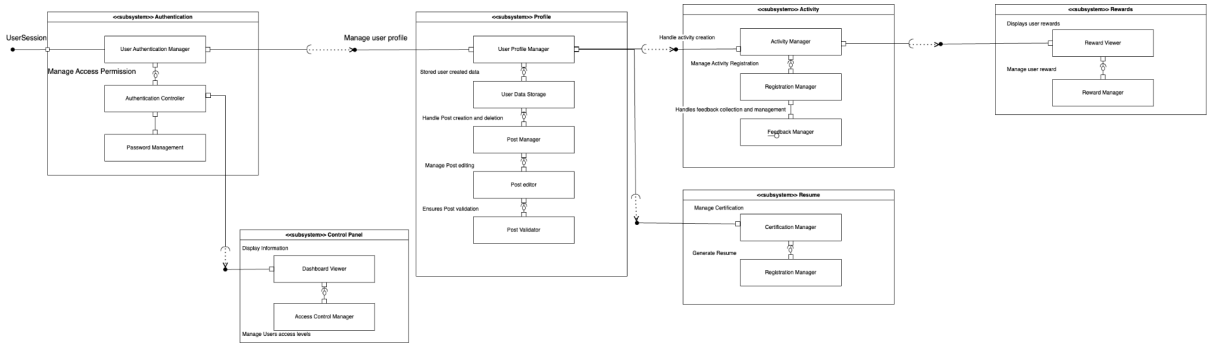

Figure 1.2: Component Diagram of StuPort System

The system architecture diagram provides a comprehensive overview of the different subsystems and their components within a software application. Each subsystem plays a crucial role in the overall functionality of the application, focusing on specific areas such as authentication, user profiles, activities, resumes, rewards, and control panel management.

The Authentication Subsystem is fundamental, managing user sessions and authentication processes. The UserSession component ensures session management, while the User Authentication Manager handles authentication-related operations. The Authentication Controller orchestrates the authentication flow, and Password Management is responsible for handling user password operations securely.

The Profile Subsystem manages user profiles and data. The User Profile Manager oversees user profiles, and User Data Storage stores user-created data. The Post Manager controls post creation and deletion, with the Post Editor** managing post editing. The Post Validator ensures that posts meet specific criteria before publication.

The Activity Subsystem focuses on managing activities within the application. The Activity Manager oversees activity creation, while the Registration Manager manages activity registrations. The Feedback Manager collects and manages feedback from users participating in activities.

The Resume Subsystem manages certifications and resume-related information. The Certification Manager handles certifications, and the Registration Manager likely manages input for generating a user's resume. The Rewards Subsystem handles user rewards. The Reward Viewer displays user rewards, while the Reward Manager manages the distribution and management of rewards.

The Control Panel Subsystem provides users with a dashboard for accessing different parts of the system. The Dashboard Viewer allows users to view a dashboard that aggregates information from various parts of the system. The Access Control Manager manages user access levels to different parts of the system, ensuring appropriate permissions are enforced.

Overall, this architecture diagram provides a structured breakdown of the main functionalities of the application, guiding the development process and ensuring that each component works together seamlessly. The relationships between components, depicted by lines indicating data flow, control flow, or hierarchy, help visualize how different parts of the system interact with each other. This high-level representation is crucial for understanding the system's organization and ensuring that it meets the requirements of its users.

2. Detailed Description of Components

2.1 Complete Package Diagram

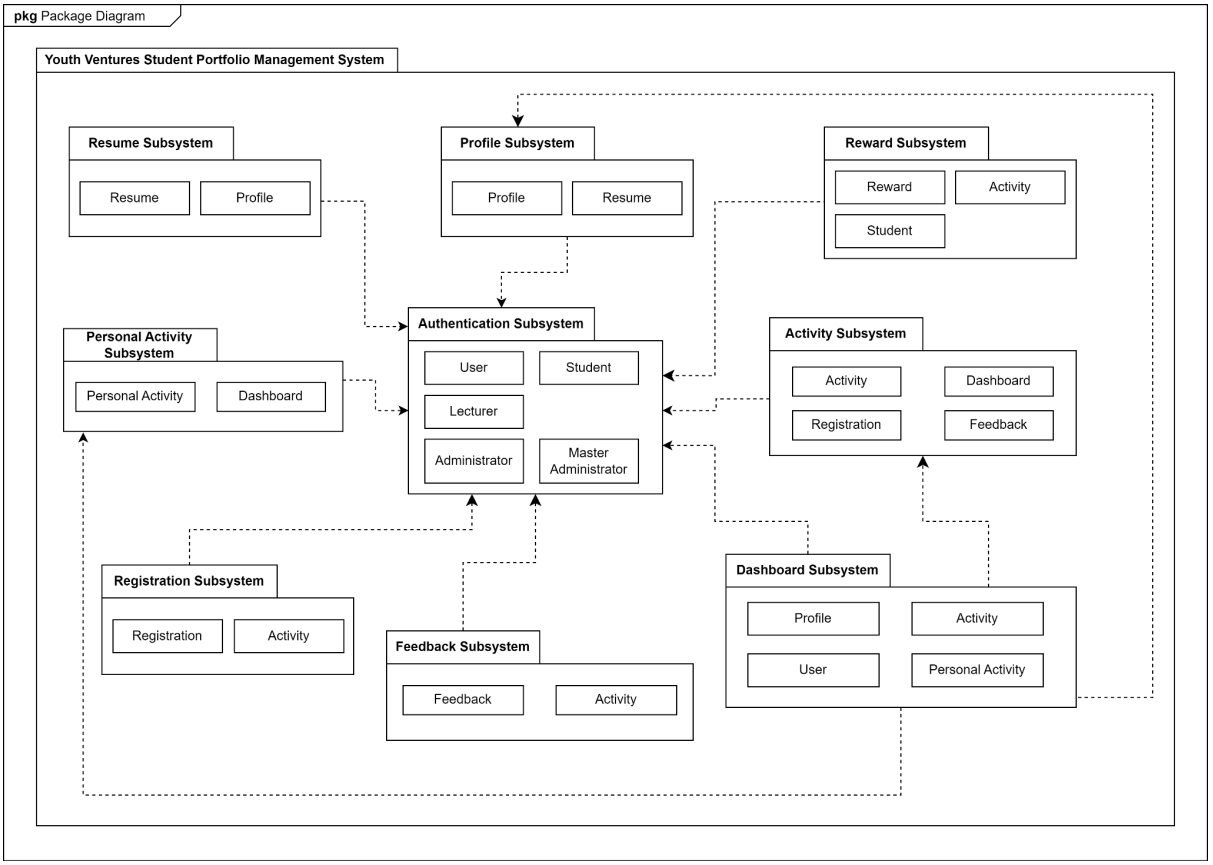

Figure 2.1: Package Diagram for Youth Ventures Student Portfolio Management System (StuPort)

Figure 2.1 shows every component's classes of the Youth Ventures Student Portfolio Management System (StuPort) as described in the previous section to proceed with the planning and progress of the system. Youth Ventures Student Portfolio Management System (StuPort) consists of 9 subsystems. The details of each subsystem is covered in the following section.

2.2 Detailed Description

These tables will be used as reference for the sequence diagrams.

Table 2.2.1: P001 : <Authentication> Subsystem

P001 : <Authentication> Subsystem			
UC001	Log In	SD001	Log In
UC002	Log Out	SD002	Log Out
UC003	Sign Up	SD003	Sign Up

Table 2.2.2: P002 : <Profile> Subsystem

P002 : <Profile> Subsystem			
UC004	Create Profile	SD004	Create Profile
UC005	View Profile	SD005	View Profile
UC006	Edit Profile	SD006	Edit Profile
UC007	Delete Profile	SD007	Delete Profile

Table 2.2.3: P003 : <Dashboard> Subsystem

P003 : <Dashboard> Subsystem			
UC008	View Dashboard	SD008	View Dashboard

Table 2.2.4: P004 : <Activity> Subsystem

P004 : <Activity> Subsystem			
UC009	Publish Event	SD009	Publish Event
UC010	Edit YV Activity Details	SD010	Edit YV Activity Details
UC011	View Activity	SD011	View Activity

Table 2.2.5: P005 : <Registration> Subsystem

P005 : <Registration> Subsystem			
UC012	Register Activity	SD012	Register Activity
UC013	View Register Activity	SD013	View Register Activity

Table 2.2.6: P006 : <Feedback> Subsystem

P006 : <Feedback> Subsystem			
UC014	Create Feedback Form	SD014	Create Feedback Form
UC015	Fill Feedback Form	SD015	Fill Feedback Form
UC016	View Feedback History	SD016	View Feedback History
UC017	View Student Feedback List	SD017	View Student Feedback List

Table 2.2.7: P007 : <Reward> Subsystem

P007 : <Reward> Subsystem			
UC018	View Reward	SD018	View Reward
UC019	Create Reward	SD019	Create Reward
UC020	Give Reward	SD020	Give Reward

Table 2.2.8: P008 : <Resume> Subsystem

P008 : <Resume> Subsystem			
UC021	Download Master Transcript	SD021	Download Master Transcript

Table 2.2.9: P009 : <Personal Activity> Subsystem

P009 : <Personal Activity> Subsystem			
UC022	Add Student Personal Activity	SD022	Add Student Personal Activity
UC023	Edit Student Personal Activity	SD023	Edit Student Personal Activity
UC024	Validate Student Personal Activity	SD024	Validate Student Personal Activity
UC025	View Student Personal Activity	SD025	View Student Personal Activity
UC026	Delete Personal Activity	SD026	Delete Personal Activity
UC027	Assign Personal Activity	SD027	Assign Personal Activity
UC028	View Approved Personal Activity List	SD028	View Approved Personal Activity List

2.2.1 P001: <Authentication> Subsystem

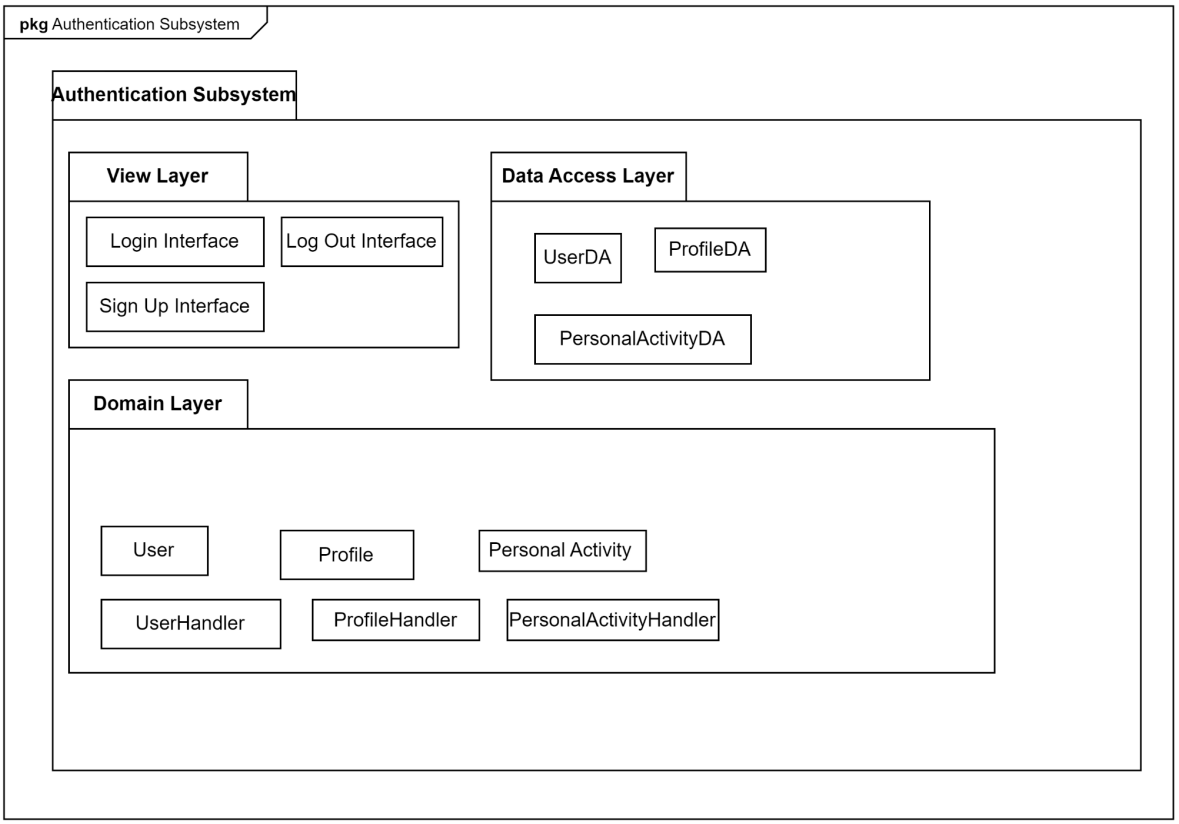

Figure 2.2.1.1: Package Diagram for <Authentication> Subsystem

2.2.1.1 Class Diagram

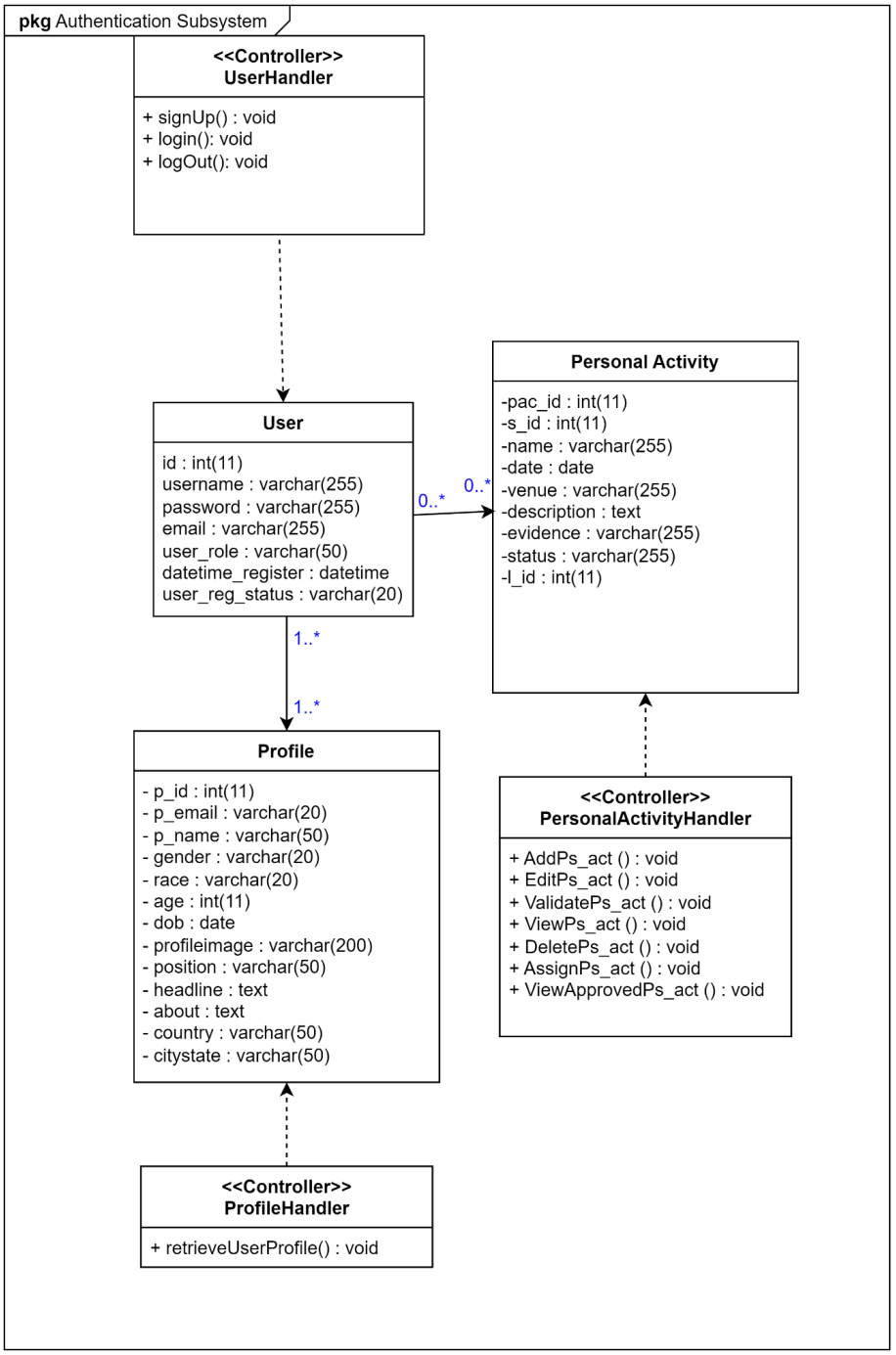

Figure 2.2.1.1.1: Class Diagram for <Authentication> Subsystem

Entity Name	Authentication
Method Name	login

Input	Login details
Output	Login successfully
Algorithm	1. Start 2. The user clicks ‘Login’ 3. The user enters a signed up email address or user ID. 4. The user enters password 5. The user clicks ‘Login’ 6. The user is redirected to home page 7. Invalid email address/userID 7.1 Users enter unsigned email address/user ID 7.2 Users enter the wrong format email address/wrong user ID 7.2 Displays 'Email is not signed-up' message 8. Incorrect password 8.1 User enter invalid password 8.2 System displays ‘Incorrect password’ 8.3 User click ‘Forgot Password’ 8.4 User received verification email in their signed-up email address. 8.5 User fill-in new password 8.6 User fill-in confirm new password 8.7 User reset password

Entity Name	Authentication
Method Name	logOut
Input	None
Output	Log Out successfully
Algorithm	1. Start

	2. The user navigates to the "Log Out" option in the user interface. 3. The user clicks on the "Log Out" option. 4. The system logs the user out. 5. The user is redirected to the login page.

Entity Name	Authentication
Method Name	signUp
Input	Sign Up details
Output	Sign up successfully
Algorithm	1. Start 2. The user clicks 'Register', then will be redirected to the Sign up Page. 3. Users will need to fill their name, role, and valid email address to register a new account. 4. User will enter a new user ID. 5. Users will need to enter the password twice and both passwords entered must be the same. 6. The user clicks 'Sign Up' to submit the registration form. 7. The system verifies the information provided, checking for any errors or missing fields. 8. If all the information is valid, the system creates a new user account. 9. System displays 'Thank you for signing up!" 10. The user is directed to the login page. 11. Invalid email address 11.1 Users enter non-existent email address. 11.2 Error message will be displayed. 12. Email is signed up before

	12.1 Users enter their valid email address but it is registered. 12.2 Display "Email Registered". 12.3 Users will be directed to login page. 13. User ID signed up before 13.1 Users enter their new user ID but it is signed up.. 13.2 Display "User ID Registered". 13.3 Users will be directed to login page. 14. Incorrect password format 14.1 Users enter password which does not meet the specified criteria. 14.2 “Password does not matched the format” displayed. 15. Both passwords do not match 15.1 Users enter the second password which is not same as the first one. 15.2 “Password does not matched” displayed.

2.2.1.2 Sequence Diagram

a) SD001: Sequence Diagram for Log In

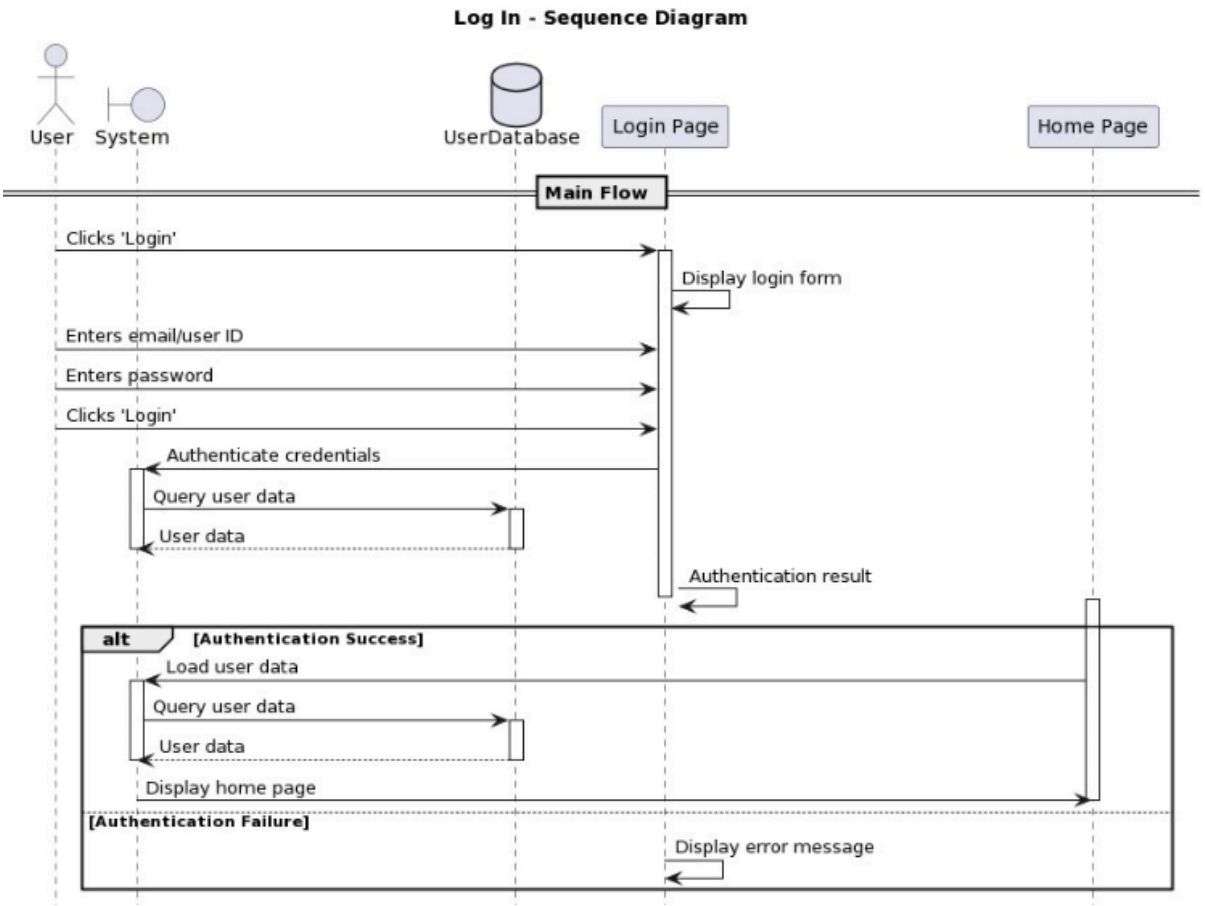

Figure 2.2.1.2.1: Sequence Diagram of Log In

b) SD002: Sequence Diagram for Log Out

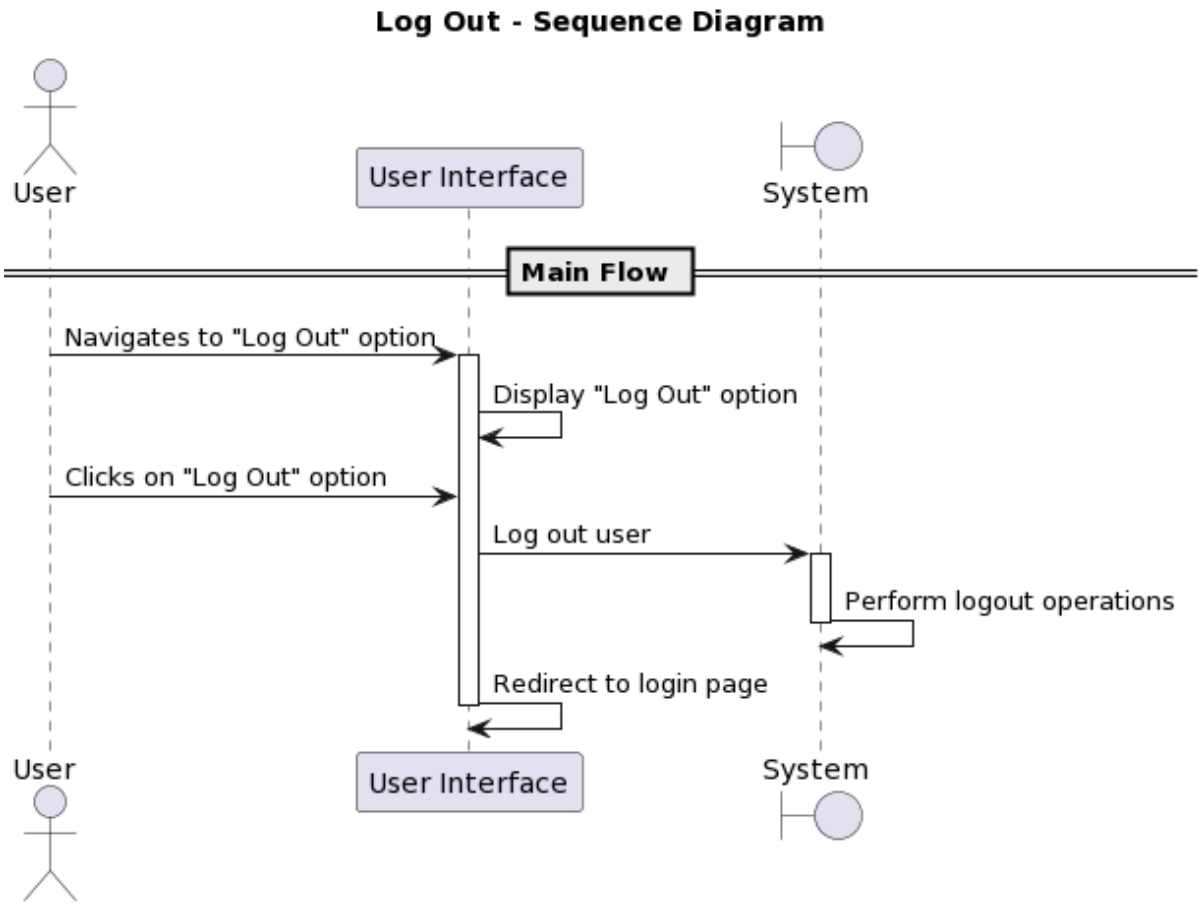

Figure 2.2.1.2.2: Sequence Diagram of Log Out

c) SD003: Sequence Diagram for Sign Up

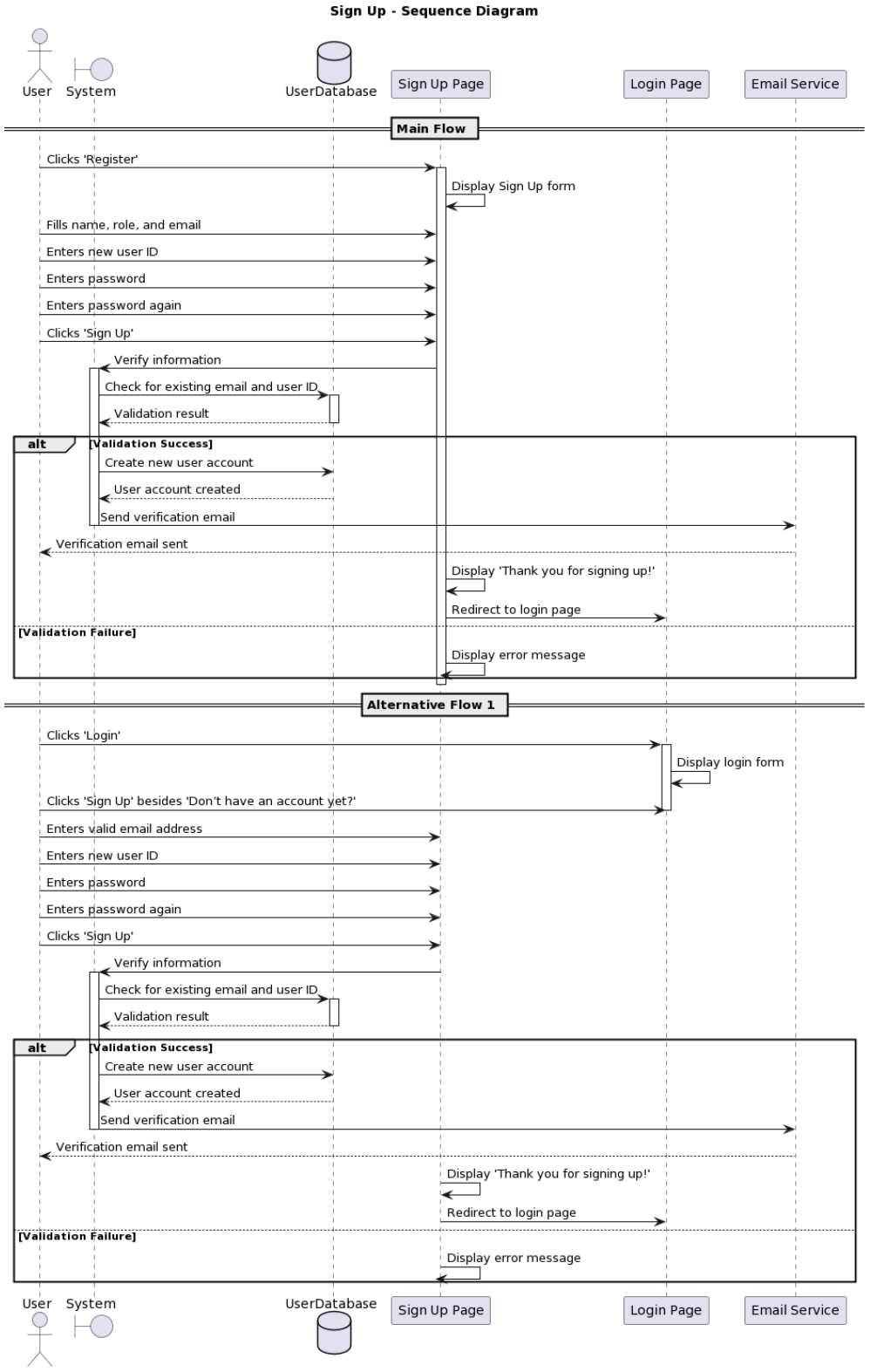

Figure 2.2.1.2.3: Sequence Diagram of Sign Up

2.2.2 P002: <Profile> Subsystem

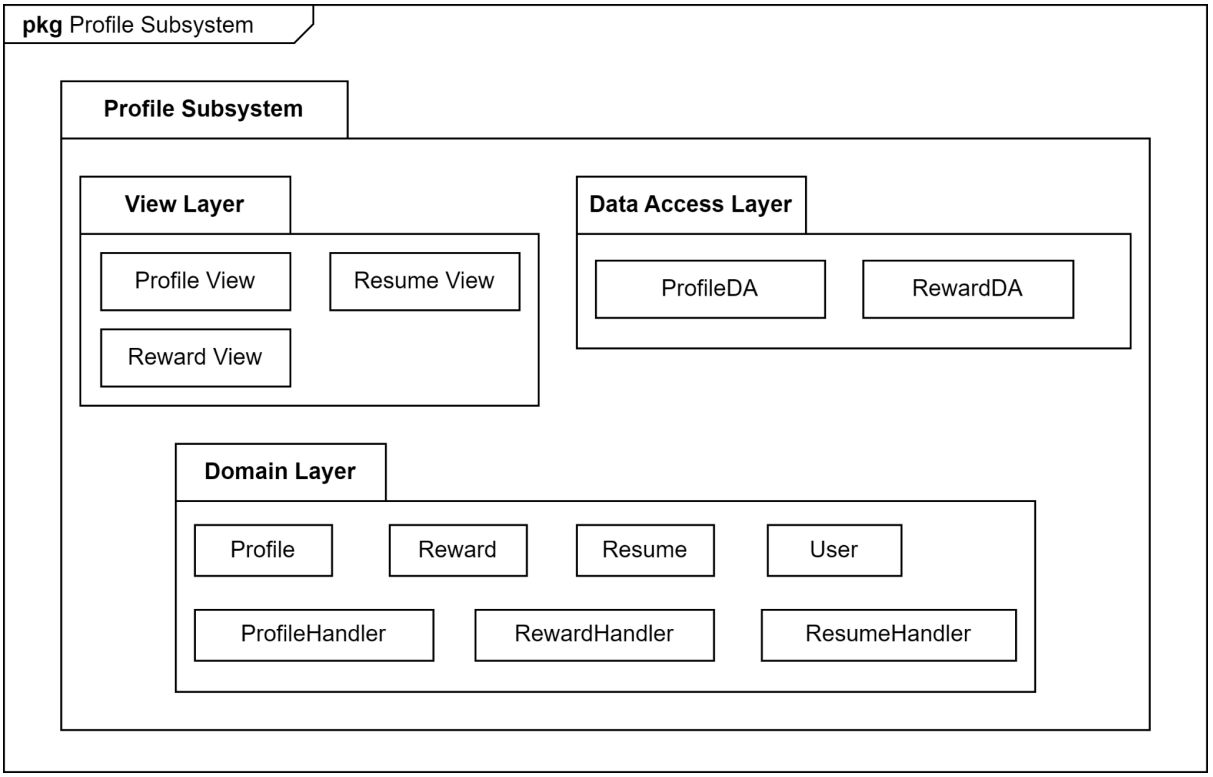

Figure 2.2.2.1: Package Diagram for <Profile> Subsystem

2.2.2.1 Class Diagram

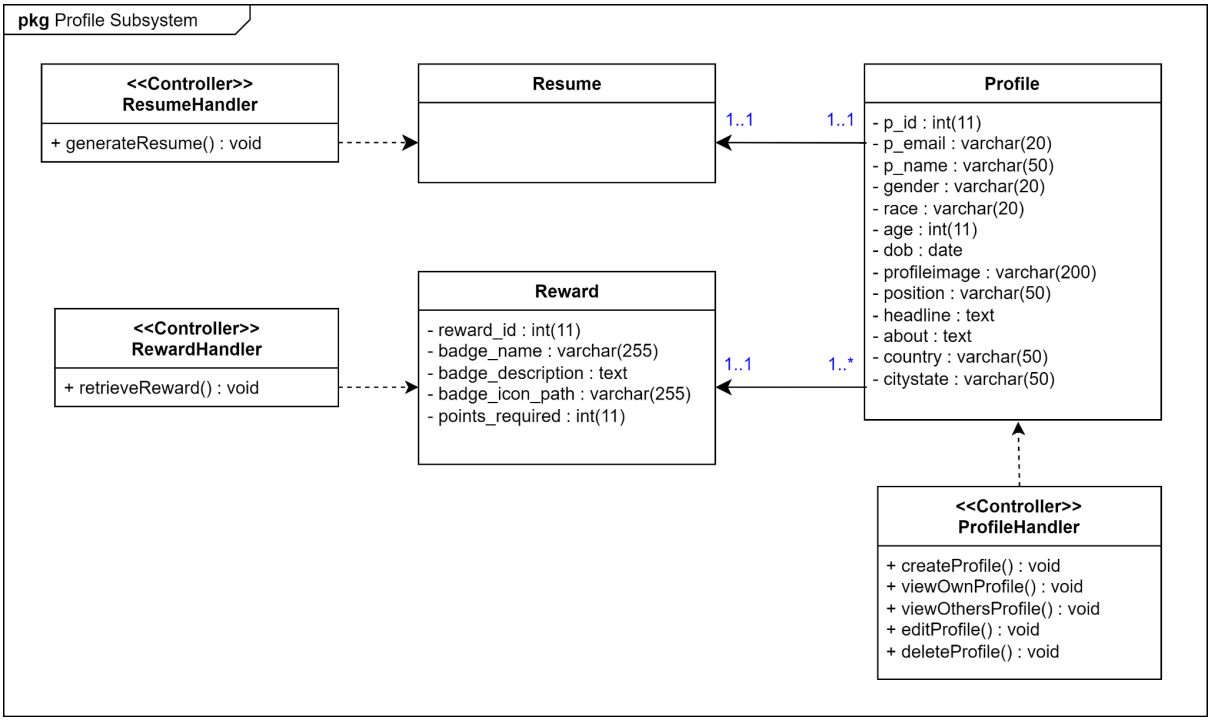

Figure 2.2.2.1.1: Class Diagram of Profile

Entity Name	Profile
Method Name	createProfile
Input	Profile Details
Output	None
Algorithm	1. Start. 2. Fill in profile details needed. 3. Select the “Submit” button. 4. Validate the entered profile details for completeness and correctness. 5. If the profile details is valid: 5.1. Update the profile details of the user. 5.2. Store the updated profile details in ProfileDA. 6. Else if the profile details is invalid:

	6.1. Display error message. 6.2. Handle the correction of profile details appropriately. 6.3. Resubmit the profile details. 7. End.

Entity Name	Profile
Method Name	viewOwnProfile
Input	None
Output	Student's/ Lecturer's Own Profile Information
Algorithm	1. Start. 2. Navigates to the profile section of the system. 3. Displays the profile information of the user. 4. End.

Entity Name	Profile
Method Name	viewOthersProfile
Input	Student's/ Lecturer's Name
Output	Student's/ Lecturer's Profile Information
Algorithm	1. Start. 2. Key in the student's/ lecturer's name in the Search Bar. 3. Searching the entered student's/ lecturer's name. 4. If the student's/ lecturer's name is found: 4.1. Display a list of student and lecturer names that most matches with the searched name. 4.2. User selects the student's/ lecturer's name. 4.3. Retrieve the student's/ lecturer's profile information from ProfileDA. 4.4. Display the student's/ lecturer's profile information.

	5. Else if the student's/ lecturer's name is not found: 5.1. Display error message. 5.2. User re-key in the student's/ lecturer's name in the Search Bar. 6. End.

Entity Name	Profile
Method Name	editProfile
Input	Modified Profile Information
Output	Latest Profile Information
Algorithm	1. Start 2. Retrieve the profile information from ProfileDA. 3. Display the current profile information. 4. User select the "Edit Profile" button. 5. Edit and modify the profile details. 6. Select the "Save" button. 7. Validate the edited profile details. 8. If the profile details is valid: 8.1. Update the edited profile details. 8.2. Store the updated profile details in ProfileDA. 8.3. Display the latest profile information. 9. Else if the profile details is invalid: 9.1. Display error message. 9.2. Handle the correction of profile details appropriately. 9.3. Resubmit the edited profile details. 9.4. Continue with Algorithm Step 7. 10. End

Entity Name	Profile

Method Name	deleteProfile
Input	Student's/ Lecturer's Name
Output	None
Algorithm	1. Start. 2. If the user is Student/ Lecturer 2.1. Select the "Delete Profile" button. 2.2. Confirm the deletion. 2.3. Permanently delete the profile and all related data from the ProfileDA. 2.4. Display success message. 3. If the user is Master Administrator 3.1. Retrieve all users' profile information from ProfileDA. 3.2. Locate the profile for deletion. 3.3. Select the "Delete" button. 3.4. Confirm the deletion. 3.5. Permanently delete the profile and all related data from the ProfileDA. 3.6. Display success message. 4. End.

2.2.2.2 Sequence Diagram

a) SD004: Sequence Diagram for Create Profile

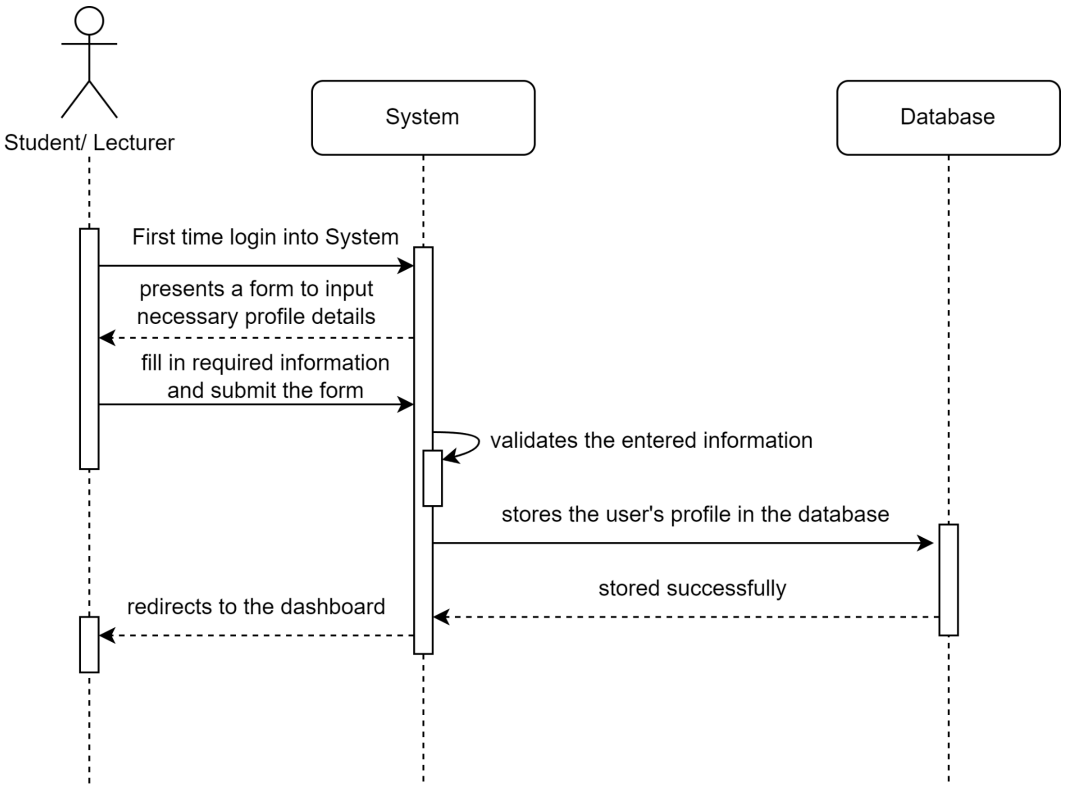

Figure 2.2.2.2.1: Sequence Diagram of Create Profile

b) SD005: Sequence Diagram for View Profile

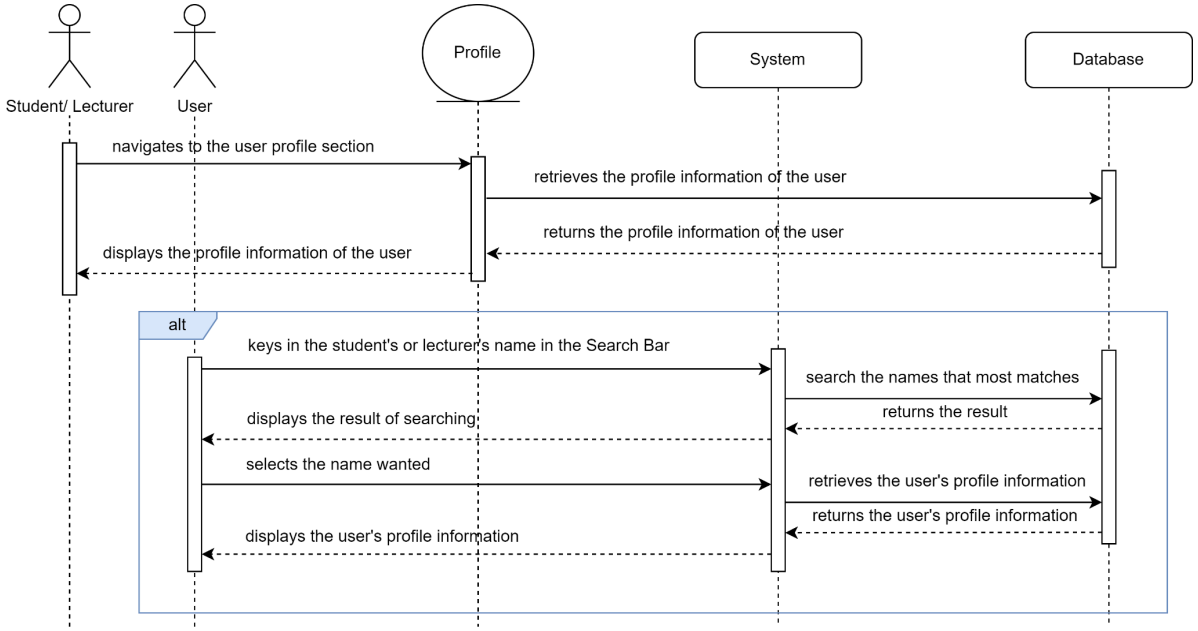

Figure 2.2.2.2.2: Sequence Diagram of View Profile

c) SD006: Sequence Diagram for Edit Profile

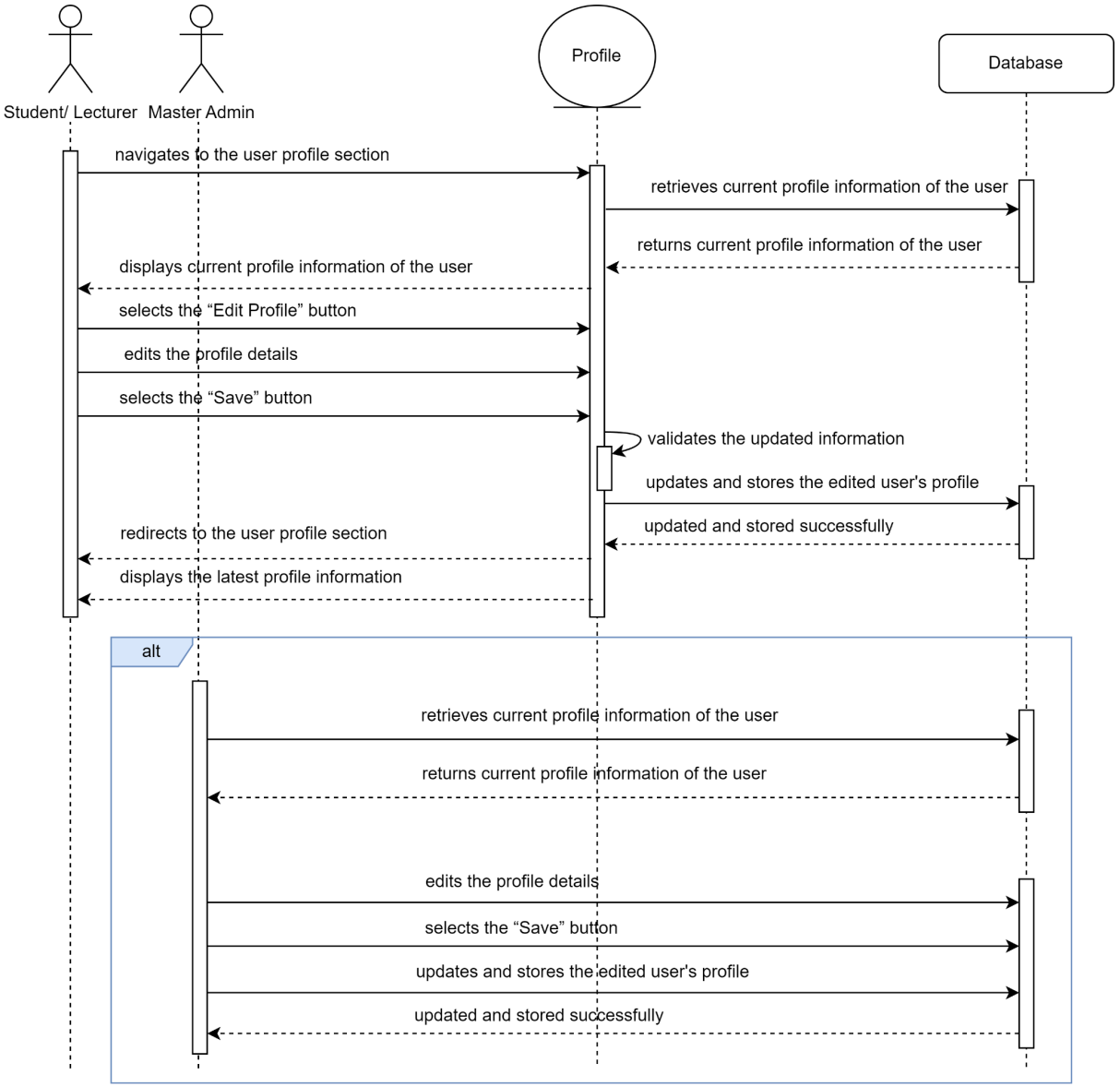

Figure 2.2.2.2.3: Sequence Diagram of Edit Profile

d) SD007: Sequence Diagram for Delete Profile

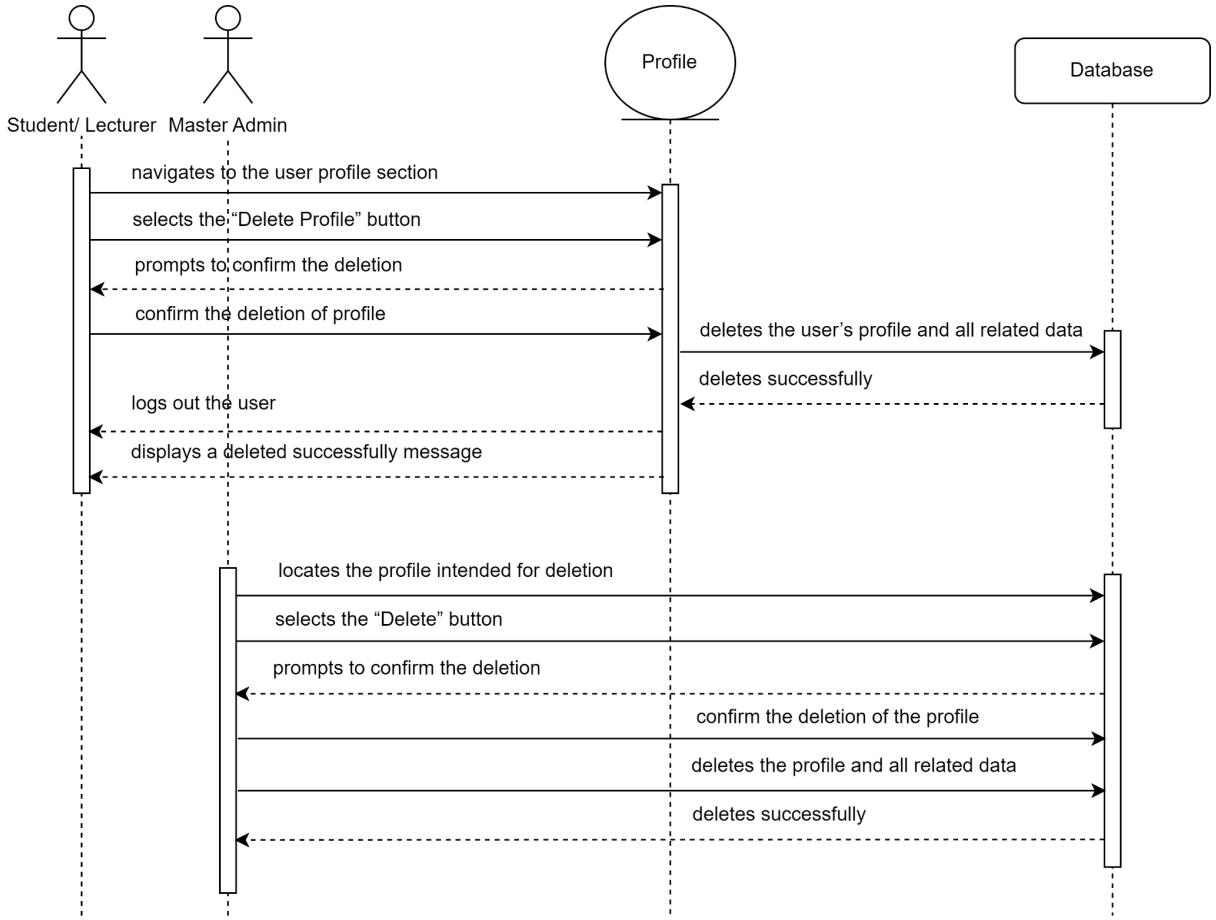

Figure 2.2.2.2.4: Sequence Diagram of Delete Profile

2.2.3 P003: <Dashboard> Subsystem

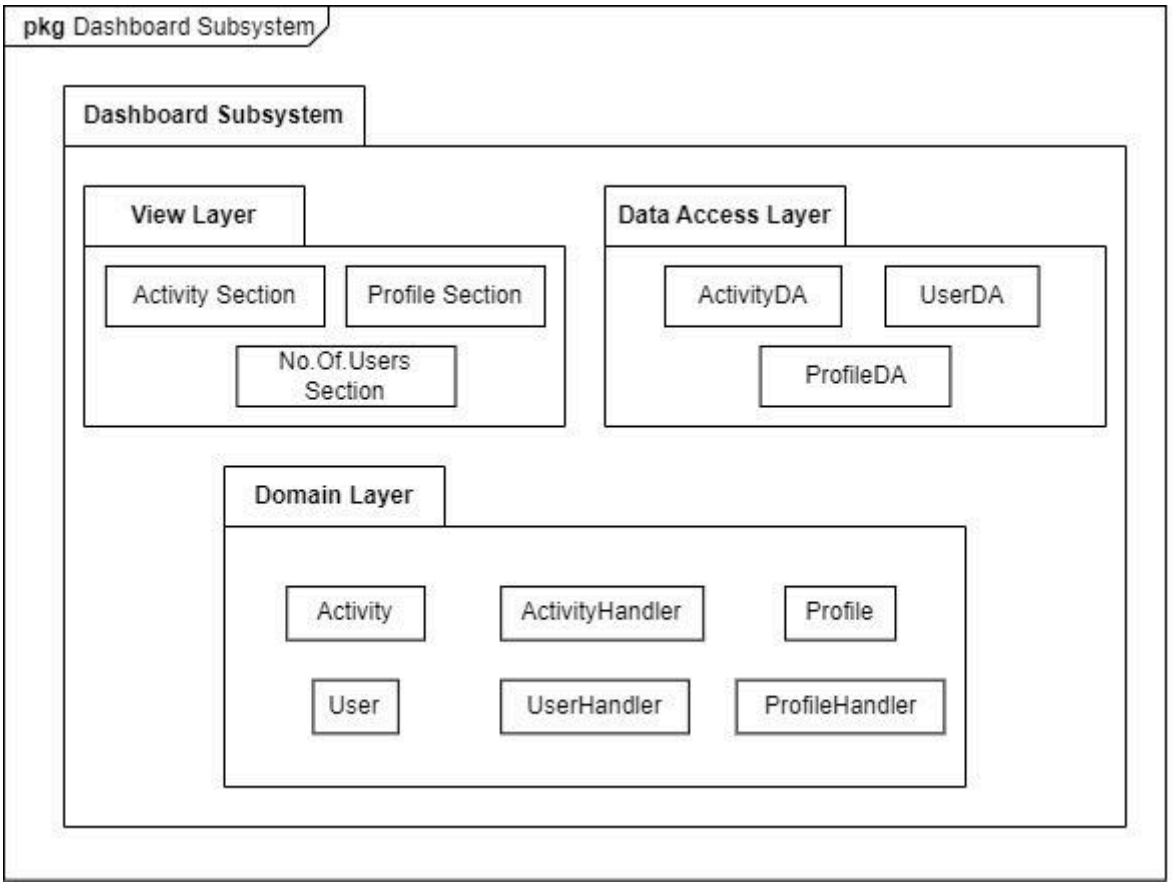

Figure 2.2.3.1: Package Diagram for <Dashboard> Subsystem

2.2.3.1 Class Diagram

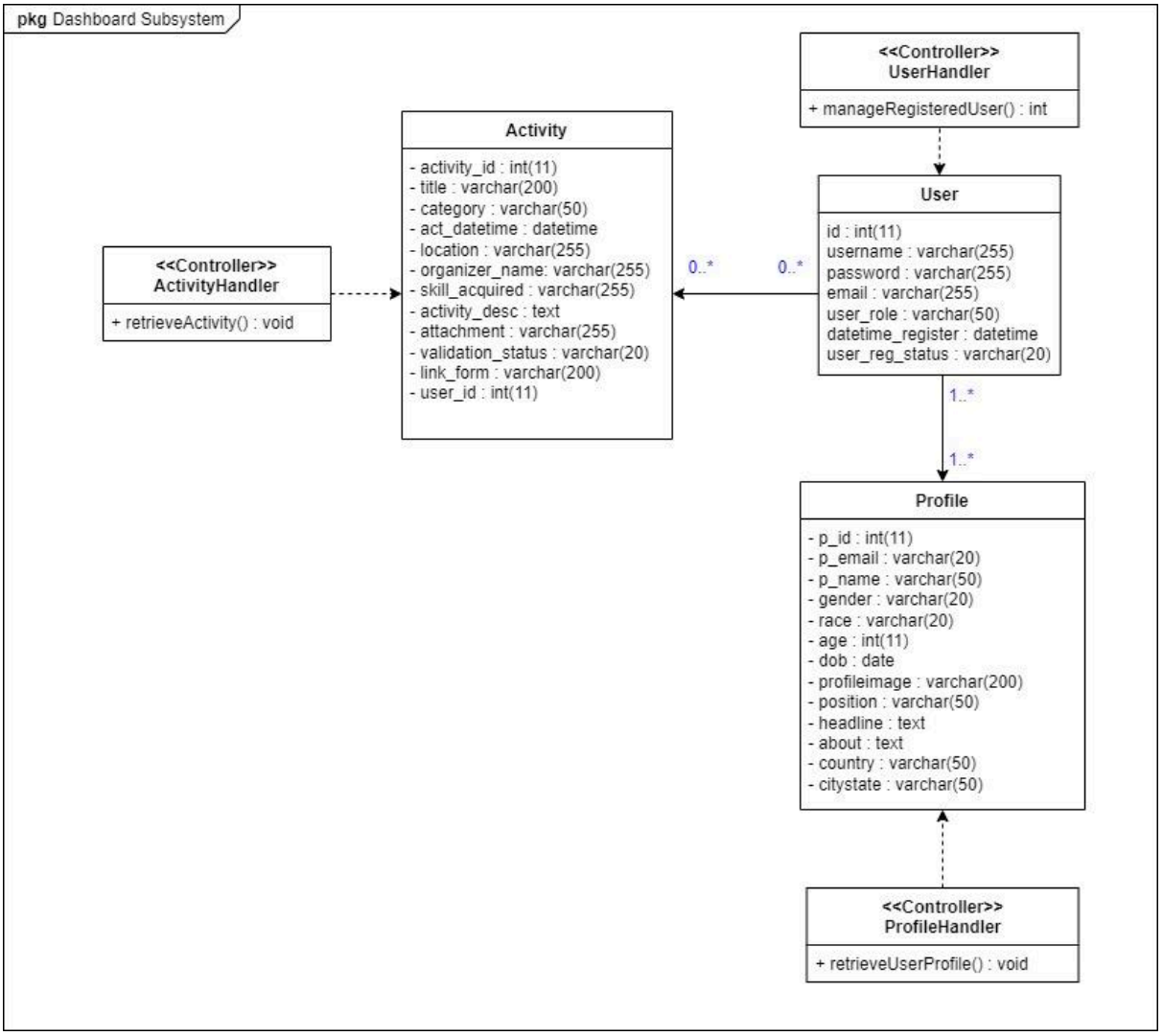

Figure 2.2.3.1.1: Class Diagram of Dashboard

Entity Name	User
Method Name	manageRegisteredUser
Input	None
Output	registered user statistics
Algorithm	1. Start. 2. Retrieve the number of registered users from UserDA.

	3. If the connection to database is successful: 3.1 The registered user statistics with different user roles will be shown on the Master Admin dashboard. 4. Else if the connection to database is unsuccessful: 4.1 Refresh the website again. 5. End.

Entity Name	Profile
Method Name	retrieveUserProfile
Input	None
Output	Student and lecturer profile main information
Algorithm	1. Start. 2. Retrieve the user profile main information from ProfileDA. 3. If the connection to database is successful: 3.1 The profile will be shown on the Student and Lecturer dashboard. 4. Else if the connection to database is unsuccessful: 4.1 Refresh the website again. 5. End.

Entity Name	Activity
Method Name	retrieveActivity
Input	None
Output	List of activities created and its main information
Algorithm	1. Start. 2. Retrieve the activity information from ActivityDA. 3. If the connection to database is successful: 3. The activity will be shown as a list on the activity section of Master Admin and Admin dashboard.

	4. Else if the connection to database is unsuccessful: 4.1 Error is detected. 5.2 Refresh the website again. 5. End.

2.2.3.2 Sequence Diagram

a) SD008: Sequence Diagram for View Dashboard

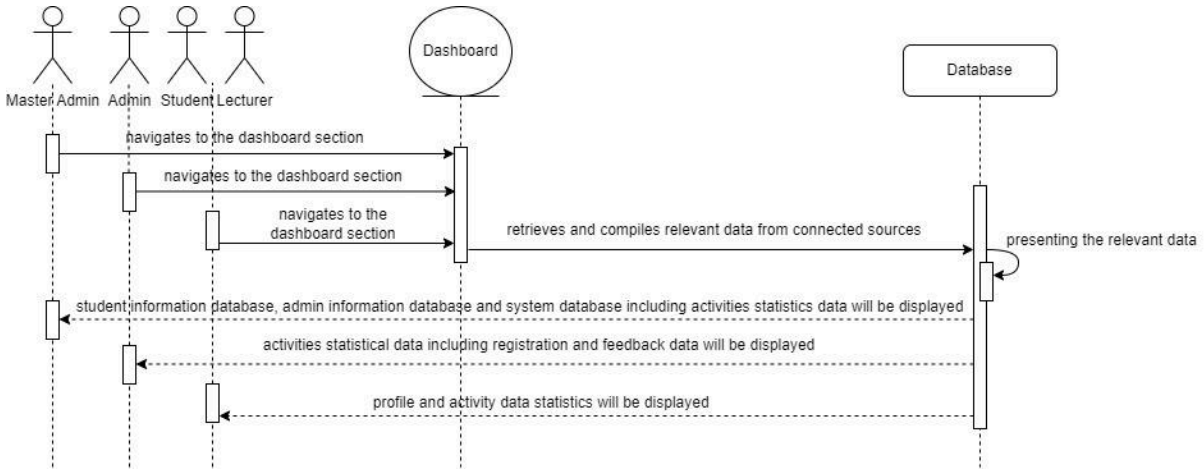

Figure 2.2.3.2.1: Sequence Diagram of View Dashboard

2.2.4 P004: <Activity> Subsystem

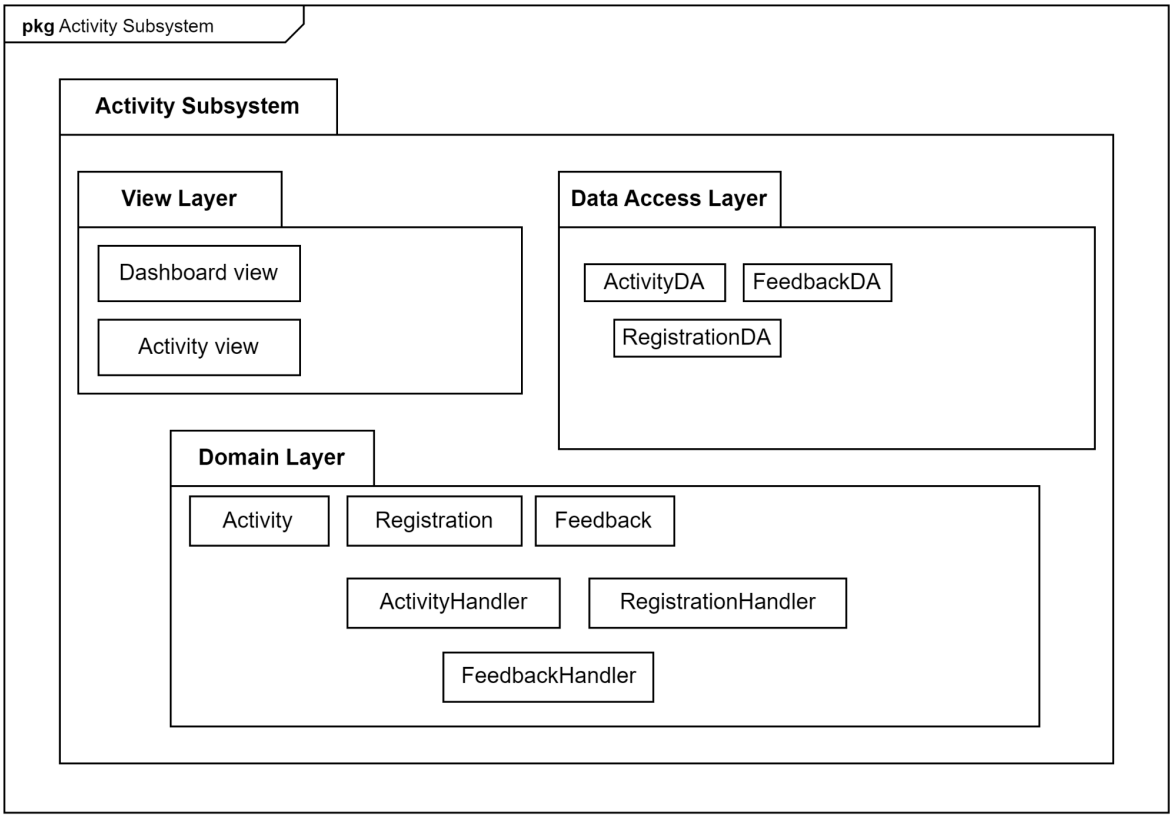

Figure 2.2.4.1: Package Diagram for <Activity> Subsystem

2.2.4.1 Class Diagram

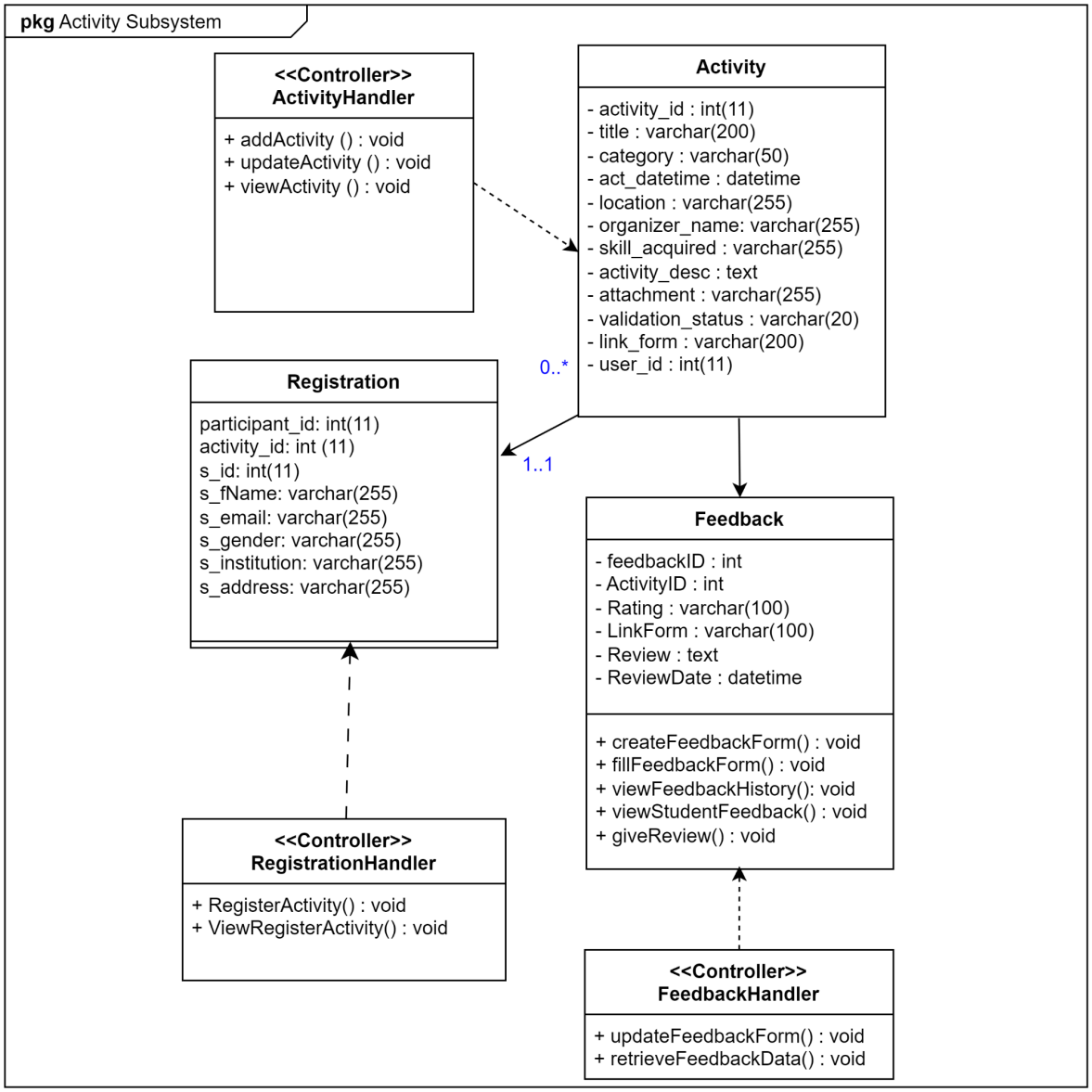

Figure 2.2.4.1.1: Class Diagram for <Activity> Subsystem

Entity Name	activity
Method Name	addActivity
Input	Date, Event Title, Organizers/sponsors, Location, Category.
Output	Success message confirming activity publication
Algorithm	1. Start

	2. Check if Actors is logged into the system and has necessary permissions. 3. Actors selects the activity category from the 6 types of activity. 4. Actors provides information: Date, Event Title, Organizers/sponsors, Location, Category, Registration information (if applicable). 5. Choose type of category. 6. Click 'Submit'. 7. Validate if all required information is provided. 8. If any required information is missing, prompt Actors to provide the necessary details. 9. If Actors answers baseline questions (Alternative Flow 1) or submits final work (Alternative Flow 2), skip to step 9. 10. Activity is published. 11. Display success message confirming activity publication. 12. End

Entity Name	activity
Method Name	updateActivity
Input	Date, Event Title, Organizers/sponsors, Location, Category.
Output	Success message updating activity publication
Algorithm	1. Start 2. Check if Admin or Master Admin is logged into the system. 3. Display a list of activities that are already published. 4. Admin or Master Admin navigates to the activity section on the dashboard. 5. Admin or Master Admin selects and clicks the activity they want to edit.

	6. Display all the information for the relevant activity. 7. Admin or Master Admin clicks the "Edit" button option. 8. Admin or Master Admin scrolls to the parts they want to edit. 9. Admin or Master Admin makes necessary changes to the activity details. 10. Admin or Master Admin clicks the "Save" button. 11. Validate if all required information is provided. 12. Update the activity details in the system. 13. Display success message confirming the update of activity details. 14. End

Entity Name	activity
Method Name	viewActivity
Input	-
Output	Success view interface activity publication
Algorithm	1. Start 2. Check if the Actor is logged into the system. 3. Actor selects the "View Activity" option. 4. Display the details of the desired activity. 5. Actor views the details of the desired activity. 6. Display success message confirming successful viewing of activity details. 7. End

2.2.4.2 Sequence Diagram

a) SD009: Sequence Diagram for Publish Event

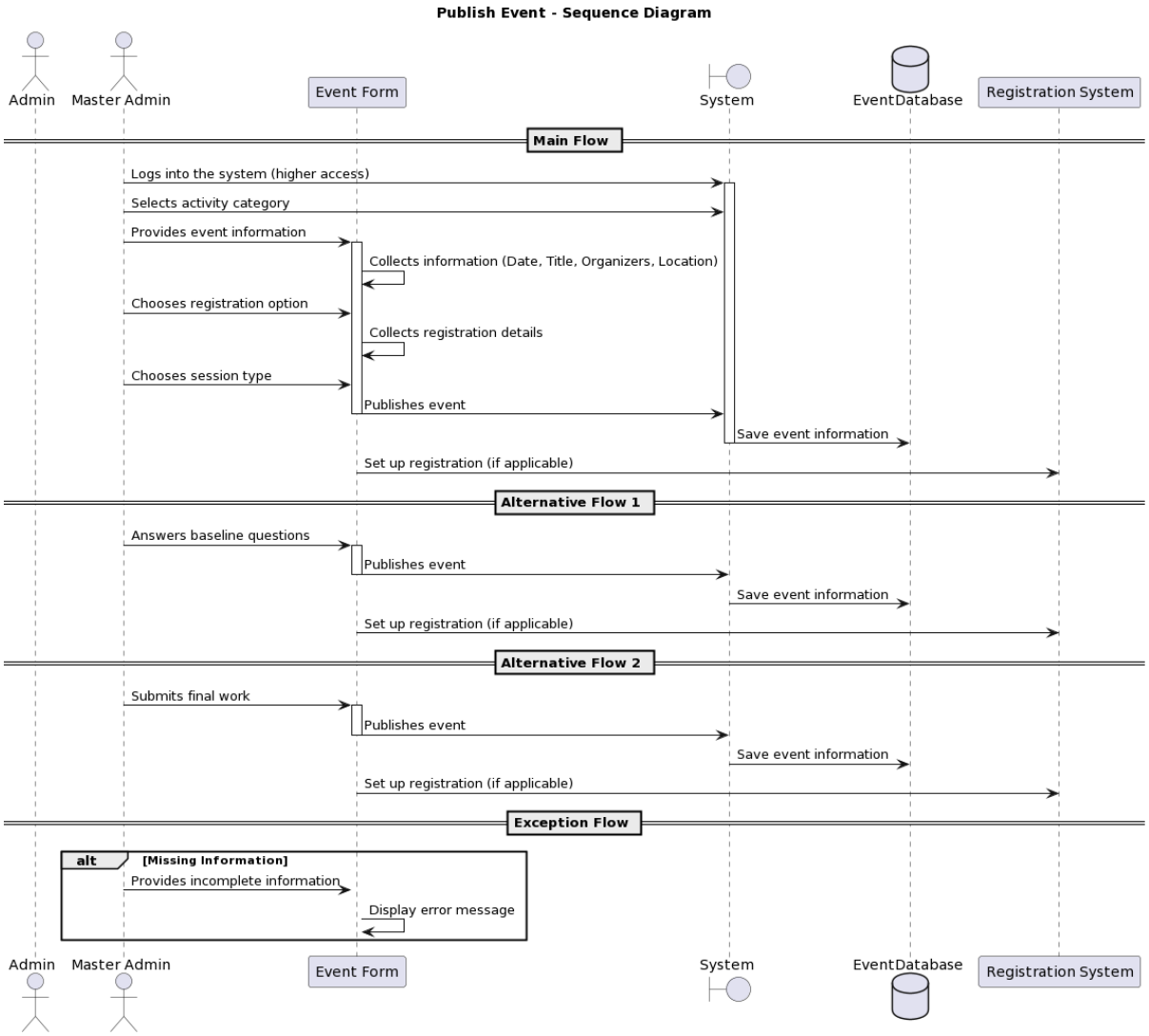

Figure 2.2.4.2.1: Sequence Diagram of Publish Event

b) SD010: Sequence Diagram for Edit YV Activity Details

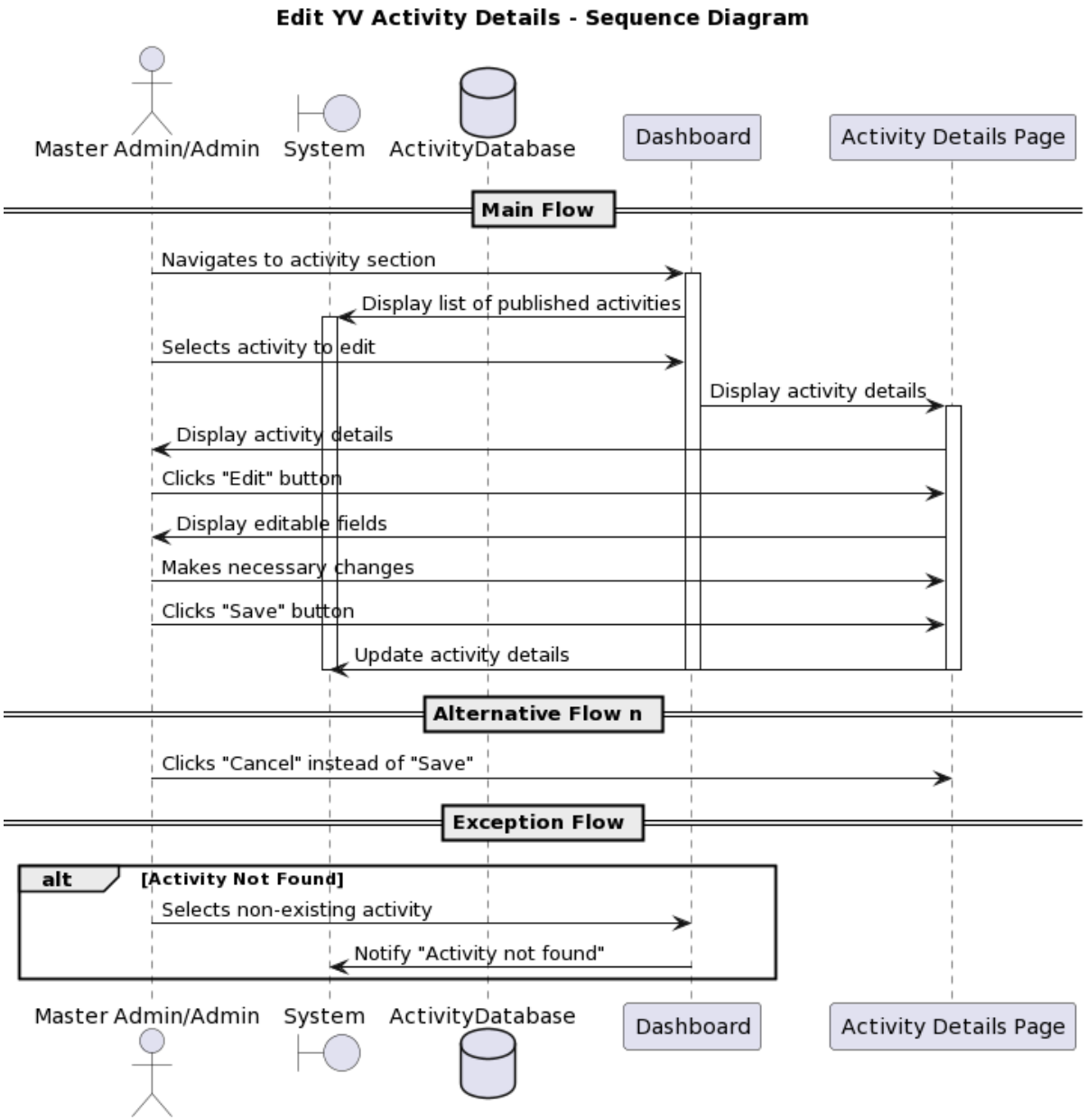

Figure 2.2.4.2.2: Sequence Diagram of Edit YV Activity Details

c) SD011: Sequence Diagram for View Activity

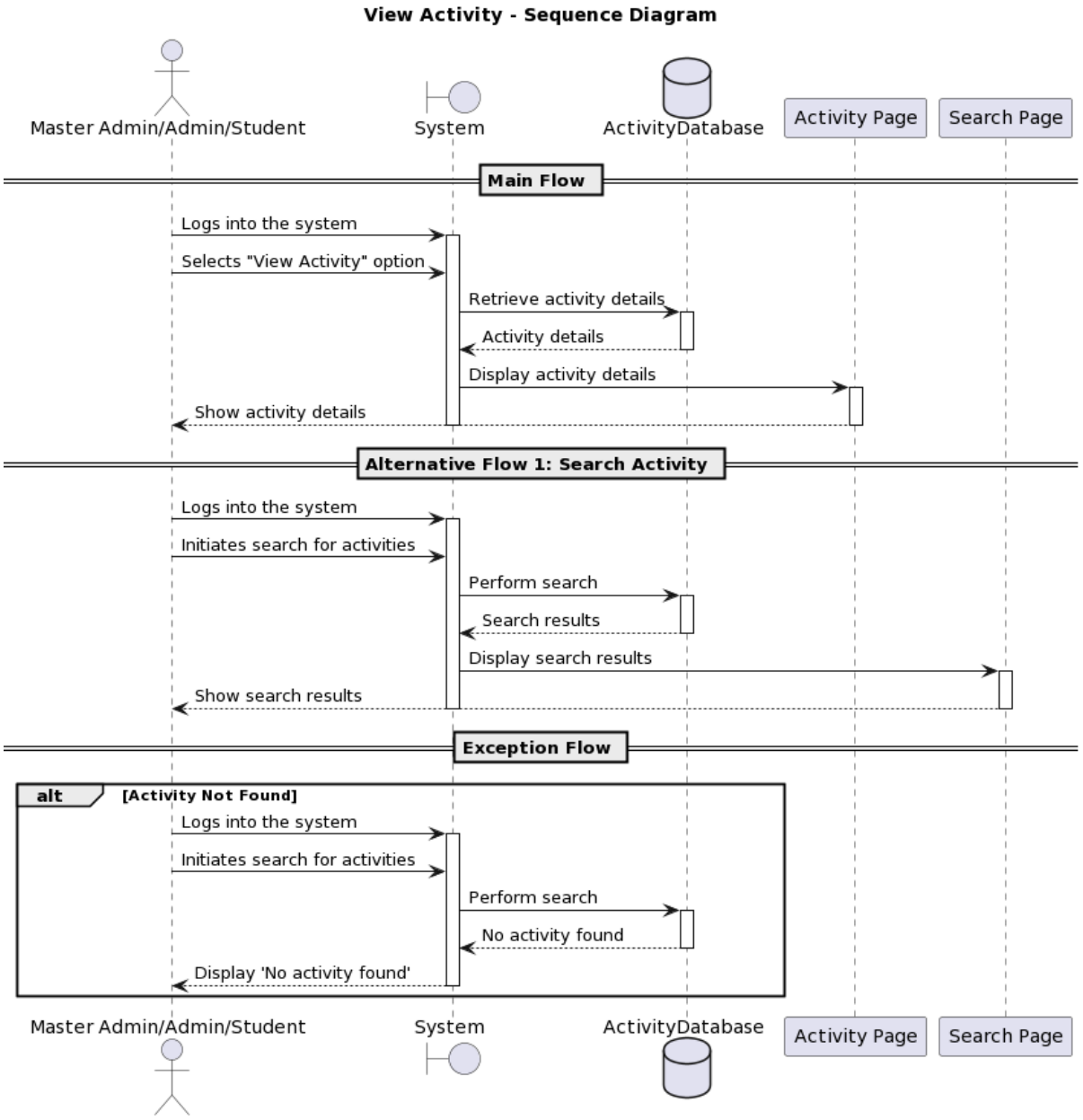

Figure 2.2.4.2.3: Sequence Diagram of View Activity

2.2.5 P005: <Registration> Subsystem

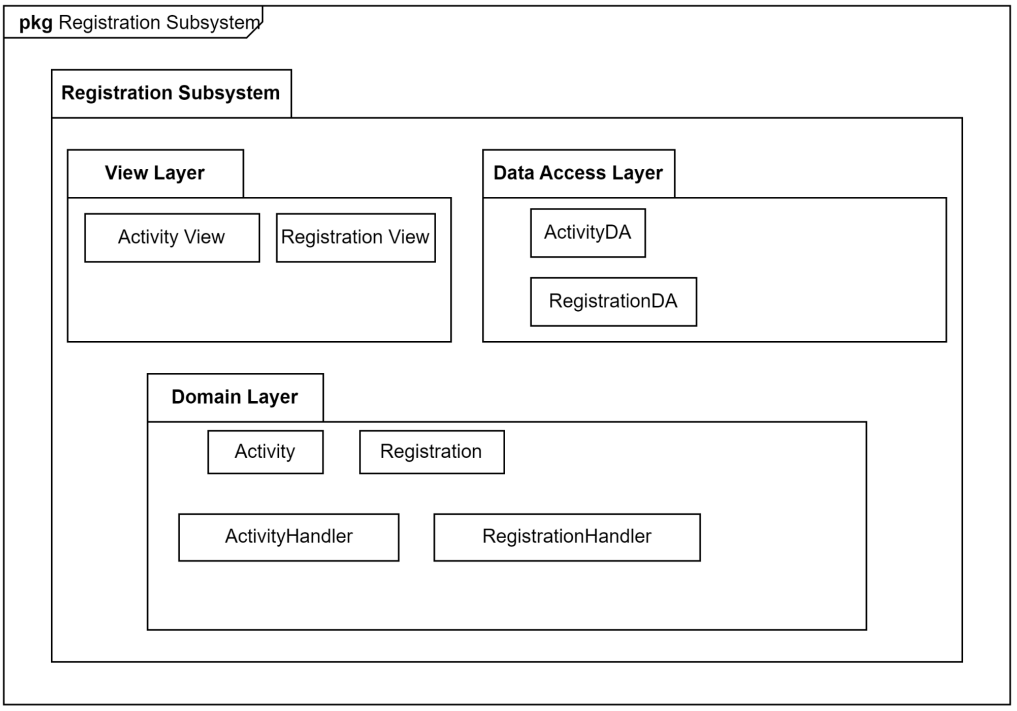

Figure 2.2.5.1: Package Diagram for <Registration> Subsystem

2.2.5.1 Class Diagram

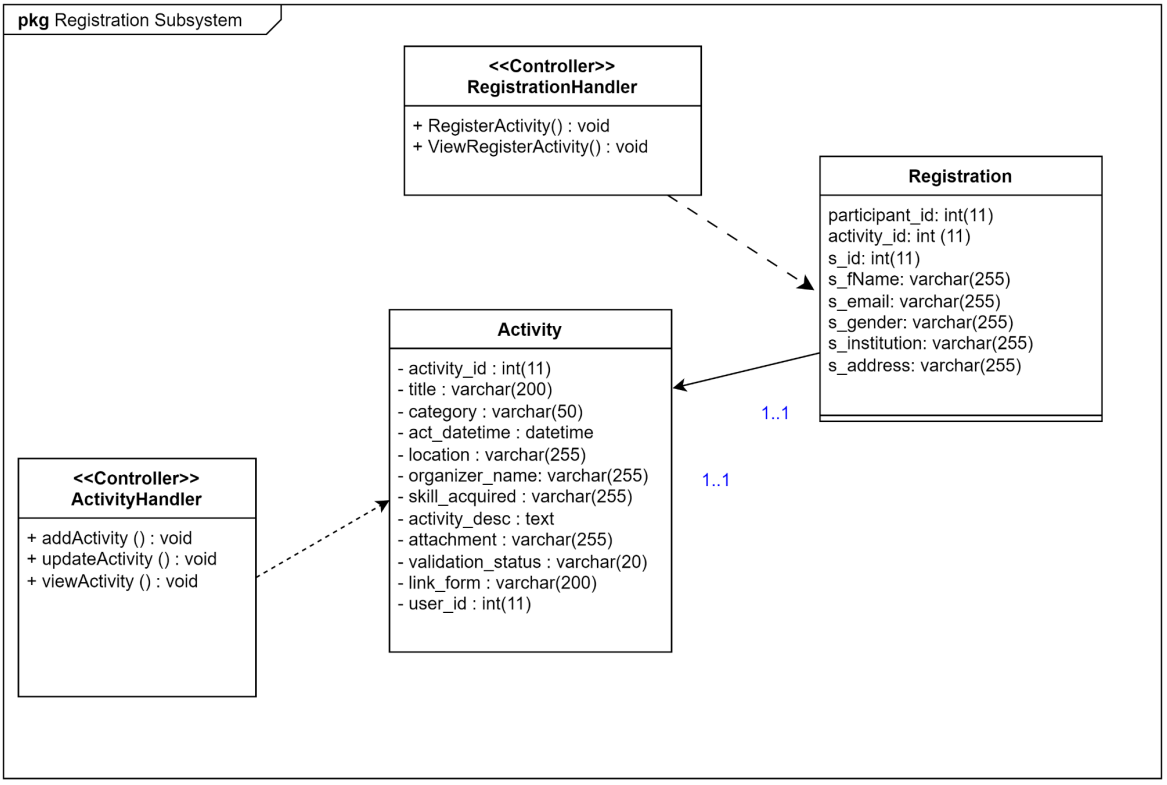

Figure 2.2.5.1.1: Class Diagram of Registration

Entity Name	activity_participant
Method Name	RegisterActivity
Input	-
Output	Button disabled and successfully register activity
Algorithm	1. Start 2. Check if the Admin or Master Admin has created the YV activity. 3. Check if the student is logged into the system. 4. Actor (Student) navigates to register for an activity. 5. Present a list of available activities to the student. 6. Student clicks the 'Register' button for the desired activity.

	7. Disable the 'Register' button and change its appearance to dull yellow to indicate successful registration. 8. Update the system to reflect the student's registration for the activity. 9. Display the registered activity in the Activity Registered List. 10. Postconditions are met: Activity is registered and shown in the Activity Registered List, 'Register' button is disabled and turns dull yellow. 11. End

Entity Name	activity_participant
Method Name	ViewRegisterActivity
Input	-
Output	Successfully view register activity
Algorithm	1. Start 2. Check if the Admin or Master Admin has created the YV activity. 3. Check if the student is logged into the system. 4. Check if the student has registered for the activity. 5. Actor (Student) selects the "Manage Registered Activity" option. 6. Display the details of the registered activity to the student. 7. Postconditions are met: Activity registered is shown in the Activity Registered List, and the 'Register' button is disabled and turns dull yellow. 8. End

2.2.5.2 Sequence Diagram

a) SD012: Sequence Diagram for Register Activity

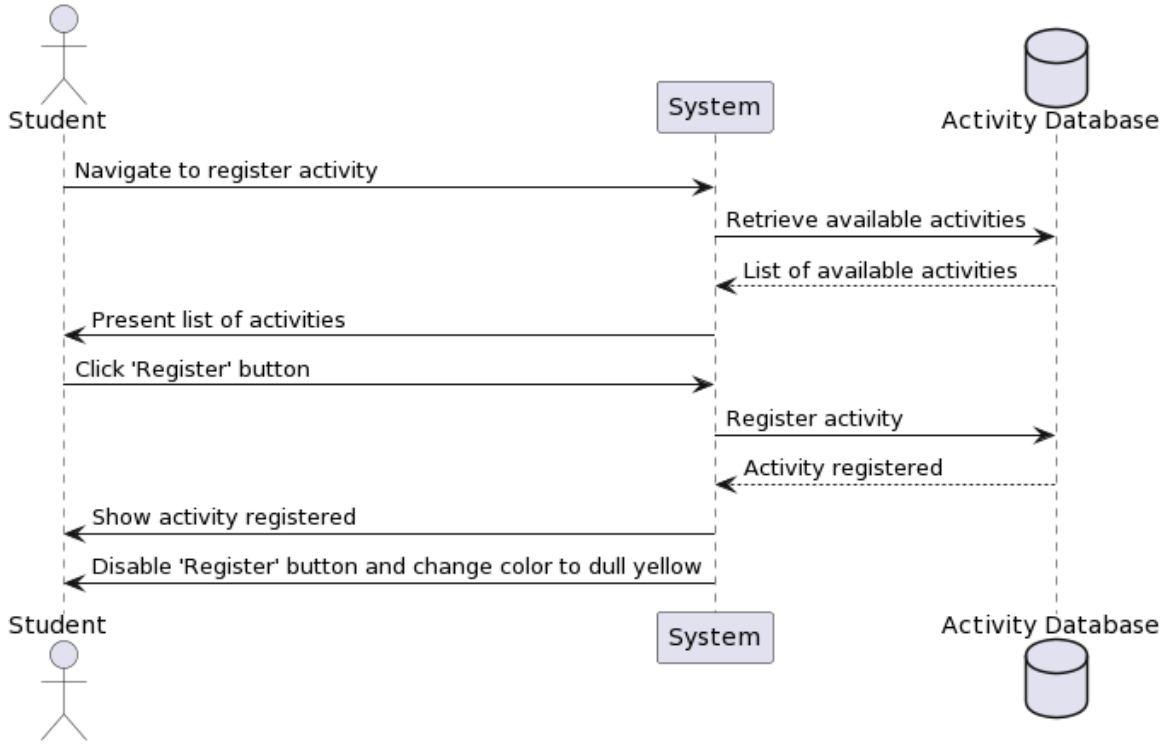

Figure 2.2.5.2.1: Sequence Diagram of Register Activity

b) SD013: Sequence Diagram for View Register Activity

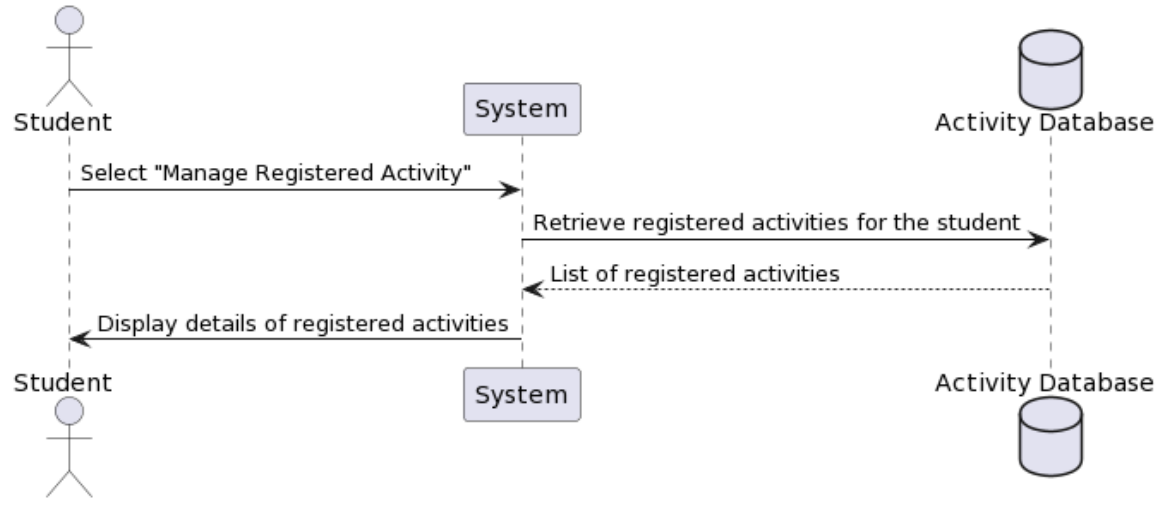

Figure 2.2.5.2.2: Sequence Diagram of View Register Activity

2.2.6 P006: <Feedback> Subsystem

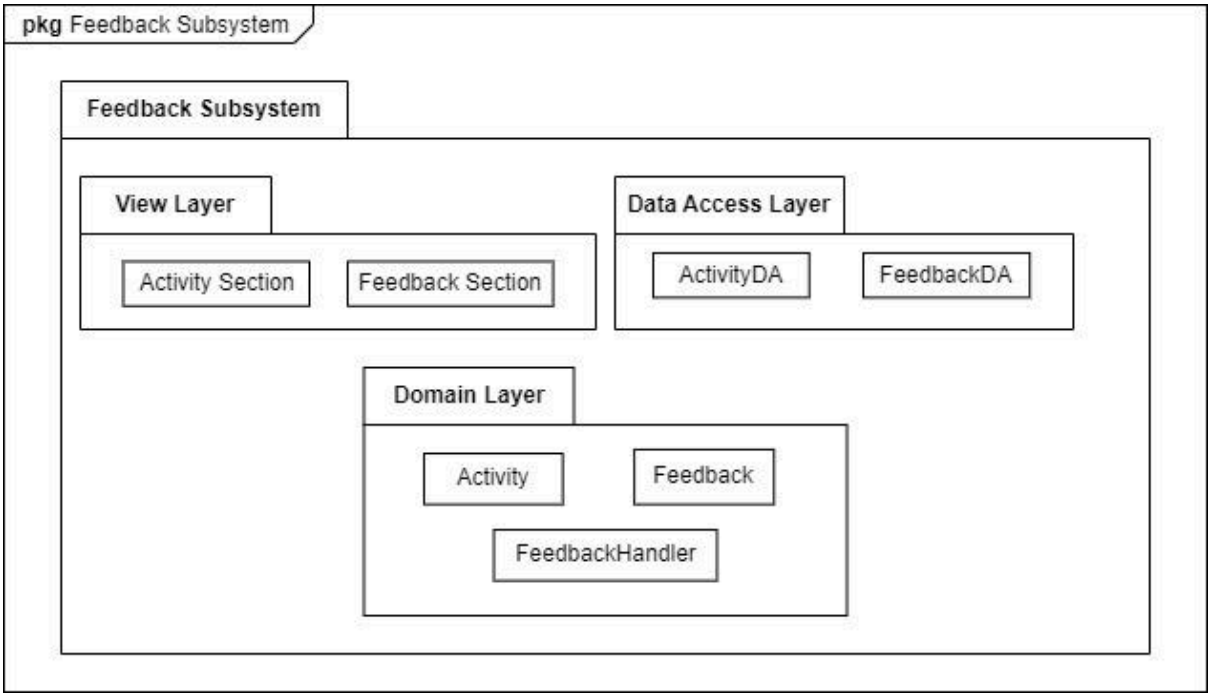

Figure 2.2.6.1: Package Diagram for <Feedback> Subsystem

2.2.6.1 Class Diagram

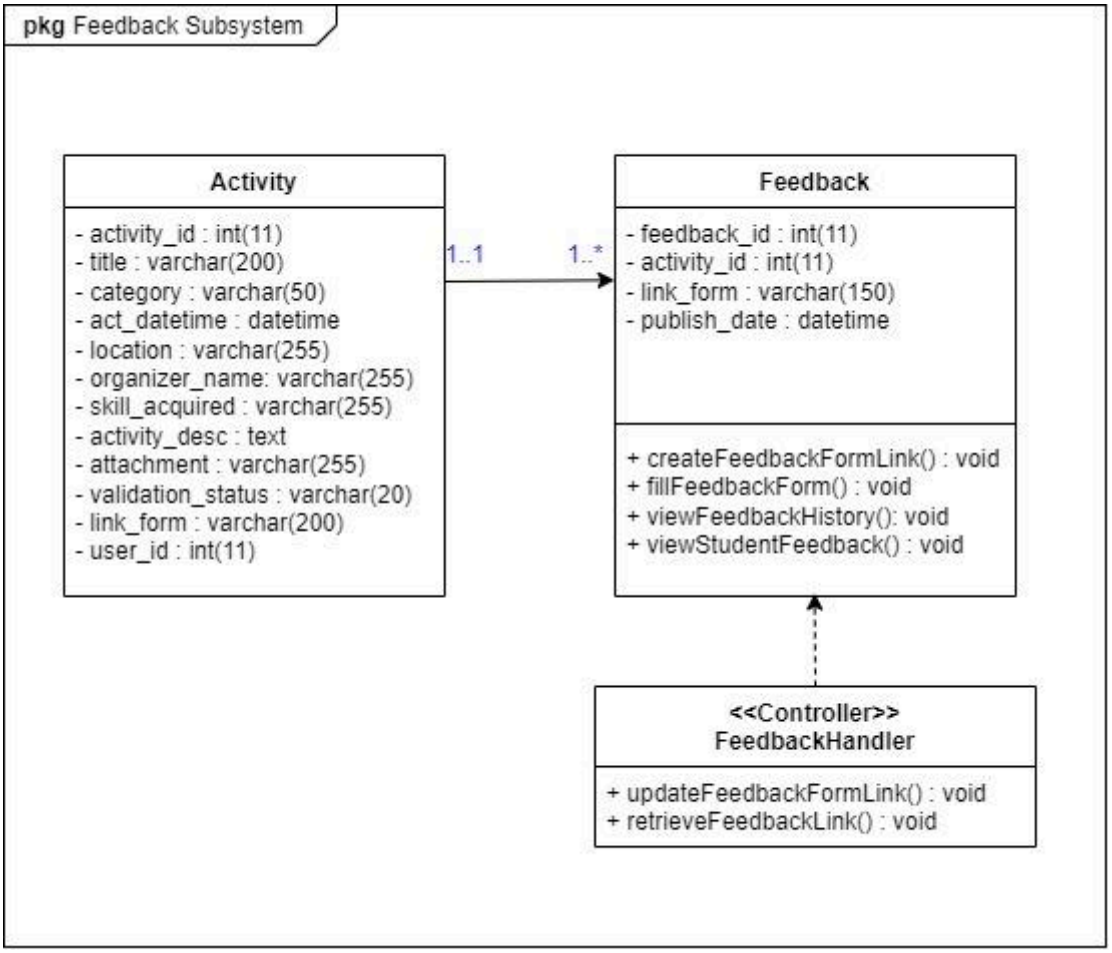

Figure 2.2.6.1.1: Class Diagram of Feedback

Entity Name	Feedback
Method Name	createFeedbackFormLink
Input	Feedback form link
Output	None
Algorithm	1. Start. 2. Input the feedback form link. 3. Validate the feedback form link to ensure it meets any necessary criteria or constraints.

	4. If the feedback form link is valid: 4.1 Update the feedback form link for the specific activity. 4.2 Save the updated feedback form link in FeedbackDA and Activity DA simultaneously. 5. Else if the feedback form link is invalid: 5.1 Error is detected. 5.2 Handle the validation error appropriately. 5.3 Resubmit the link. 6. End.

Entity Name	Feedback
Method Name	fillFeedbackForm
Input	Feedback for an activity
Output	None
Algorithm	1. Start. 2. Retrieve the feedback form link from ActivityDA. 3. Click on a feedback link which is shown on the selected activity section. 4. The system displays the contents of the link which is a feedback form. 5. The student fills in the required information and provides feedback regarding the activity experience. 6. End.

Entity Name	Feedback
Method Name	viewFeedbackHistory
Input	None
Output	Feedback details that has been filled in
Algorithm	1. Start.

	2. Retrieve the feedback form link from ActivityDA. 3. Clicks the feedback link on specific activity for which they want to view feedback. 4. The system retrieves and displays the feedback history for the selected activity. 5. Perform any necessary additional operations related to the feedback information. 6. End.

Entity Name	Feedback
Method Name	viewStudentFeedback
Input	None
Output	Feedback information that students have been filled in
Algorithm	1. Start. 2. Retrieve the feedback form link from ActivityDA. 3. Click on the feedback link for the selected activity. 4. Display the student filled out feedback form information as a list for certain activity to the Master Admin and Admin. 5. End.

2.2.6.2 Sequence Diagram

a) SD018: Sequence Diagram for Create Feedback Form

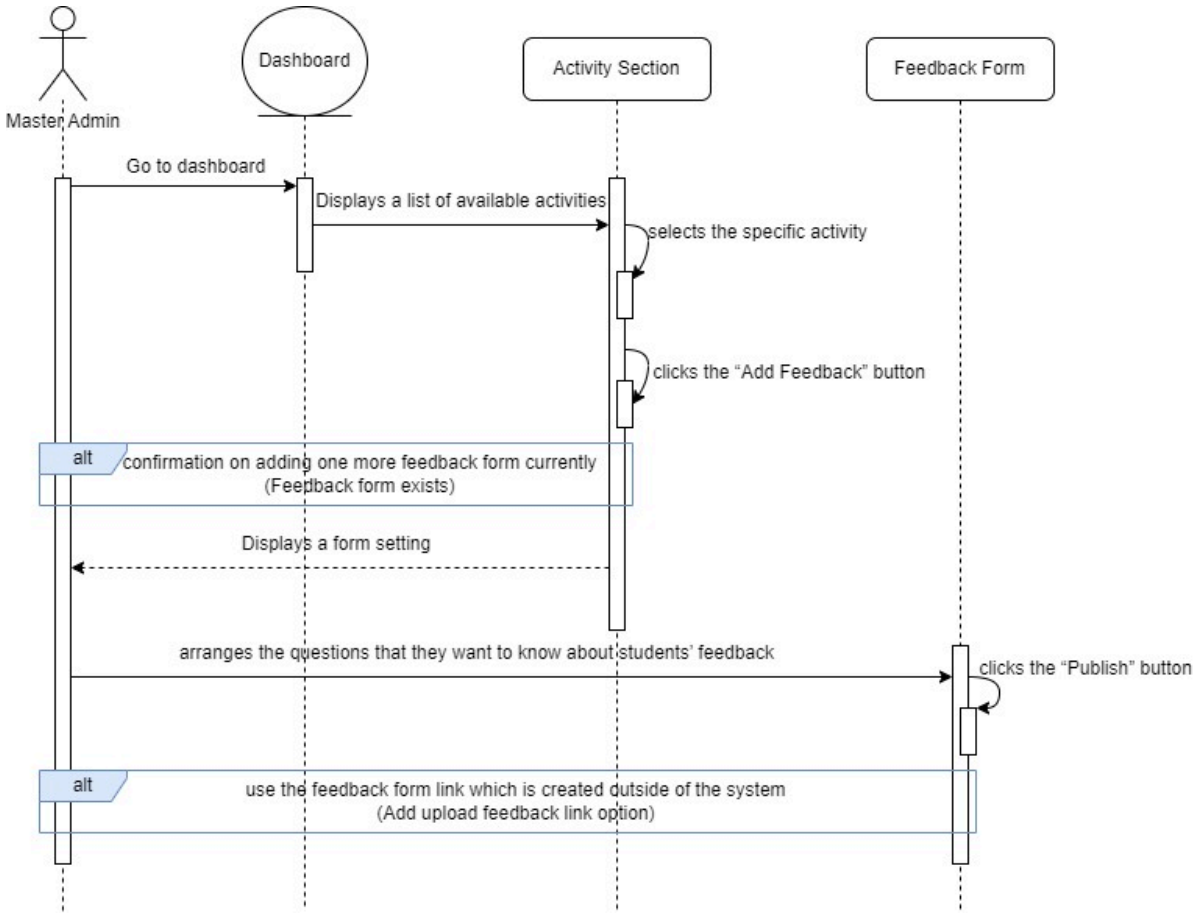

Figure 2.2.6.2.1: Sequence Diagram of Create Feedback Form

b) SD019: Sequence Diagram for Fill Feedback Form

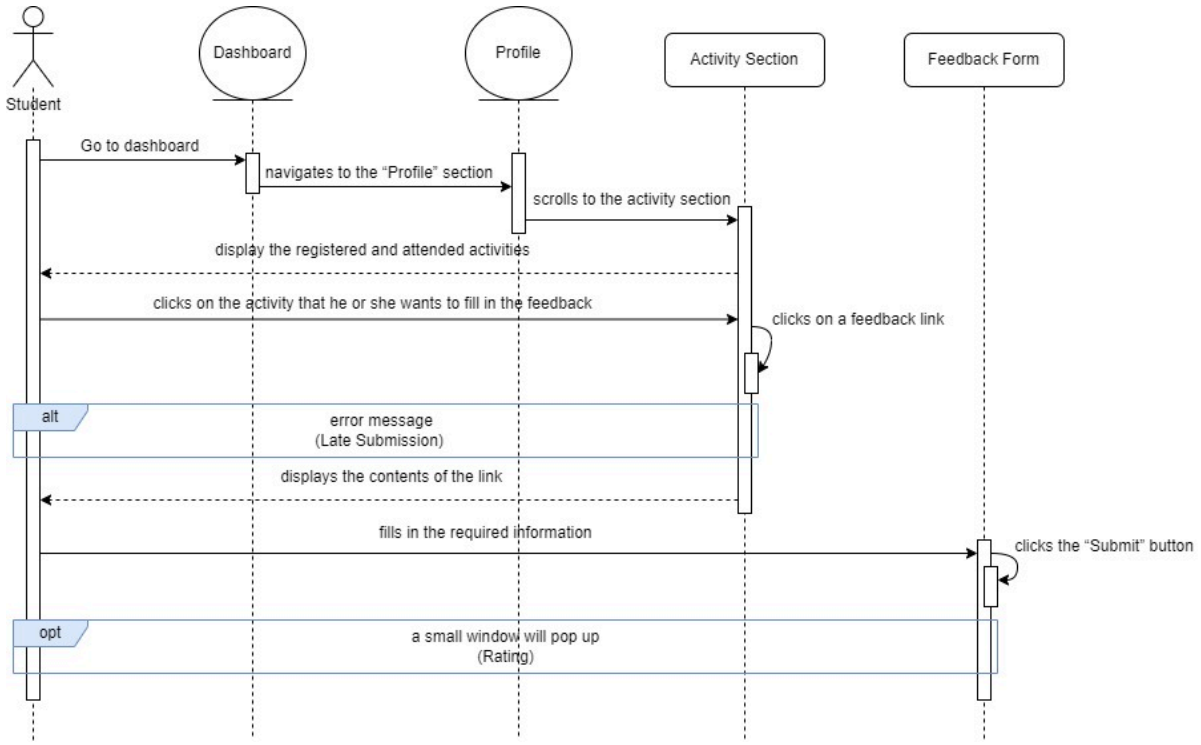

Figure 2.2.6.2.2: Sequence Diagram of Fill Feedback Form

c) SD020: Sequence Diagram for View Feedback History

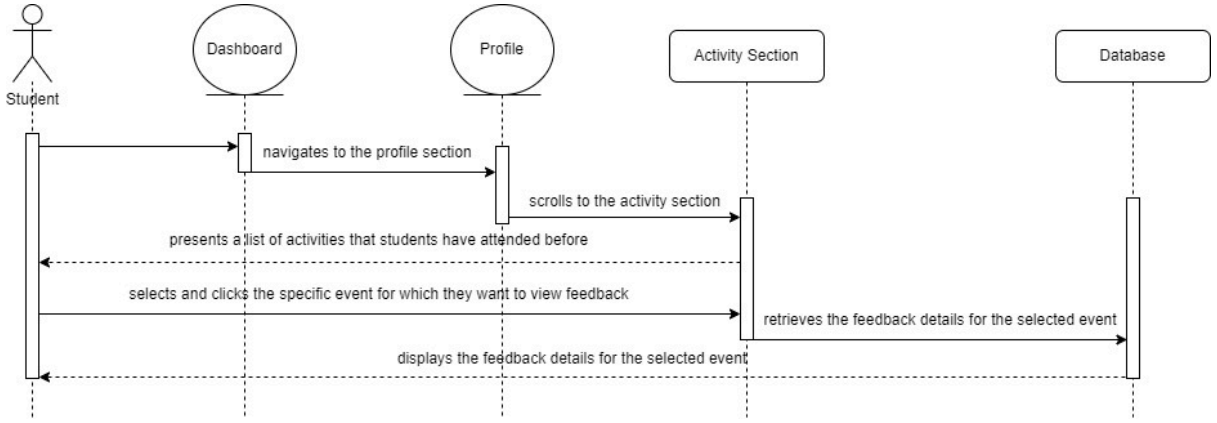

Figure 2.2.6.2.3: Sequence Diagram of View Feedback History

d) SD021: Sequence Diagram for View Student Feedback List

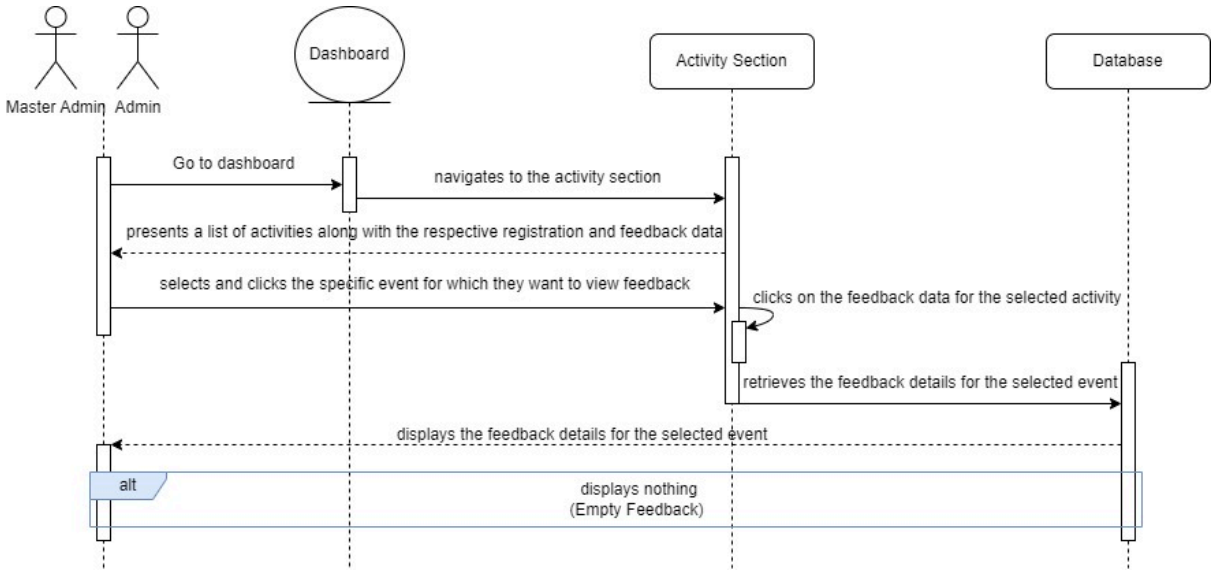

Figure 2.2.6.2.4: Sequence Diagram of View Student Feedback List

e) SD022: Sequence Diagram for Give Review

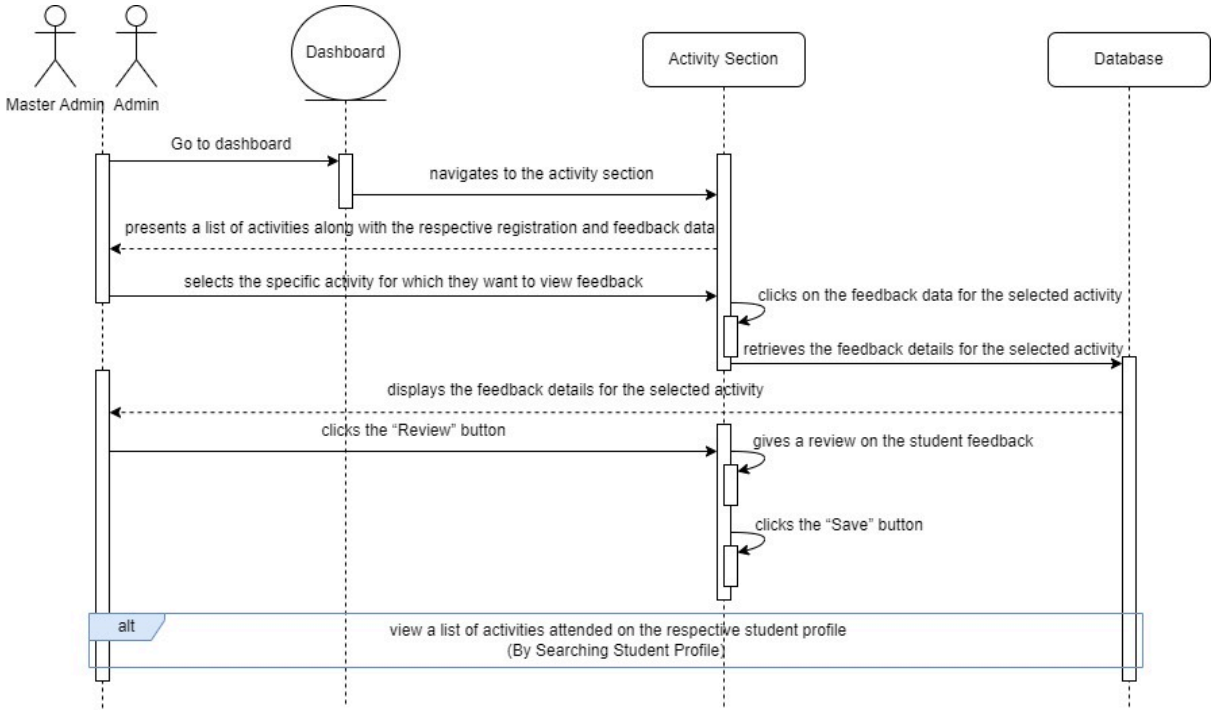

Figure 2.2.6.2.5: Sequence Diagram of Give Review

2.2.7 P007: <Reward> Subsystem

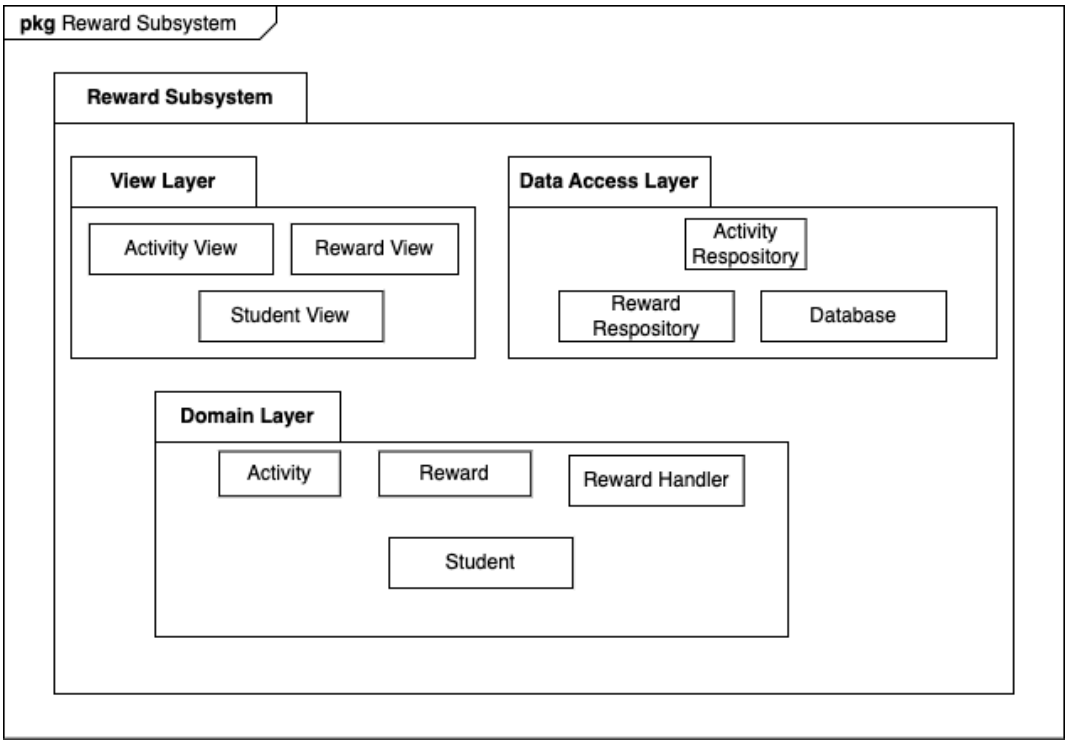

Figure 2.2.7.1: Package Diagram for <Reward> Subsystem

2.2.7.1 Class Diagram

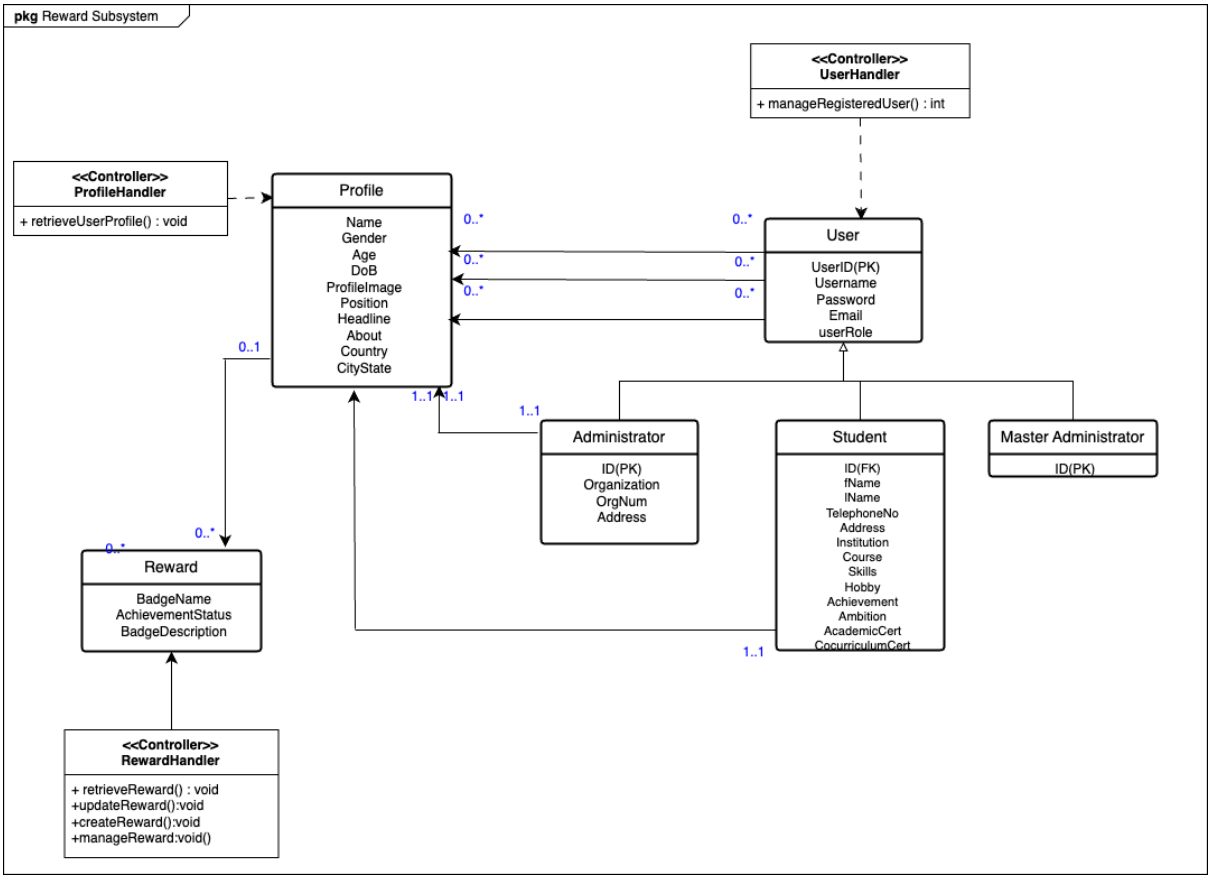

Figure 2.2.7.1.1: Class Diagram of Reward

Entity Name	Reward
Method Name	retrieveReward
Input	None
Output	Reward with all information in the system.
Algorithm	1. Start 2. Check id Student or Master Admin is logged into the system 3. If student is logged in 4. Student can view reward 5. Else if Master Admin is logged into the system 6. Master Admin can create rewards. 7. Master Admin clicks the “create reward” button .

	8. Master Admin enter reward name, description, picture and activities joined. 9. Master Admin can update the reward 10. Master Admin can delete the reward. 11. End

2.2.7.2 Sequence Diagram

a) SD018:*Sequence Diagram of View Reward*

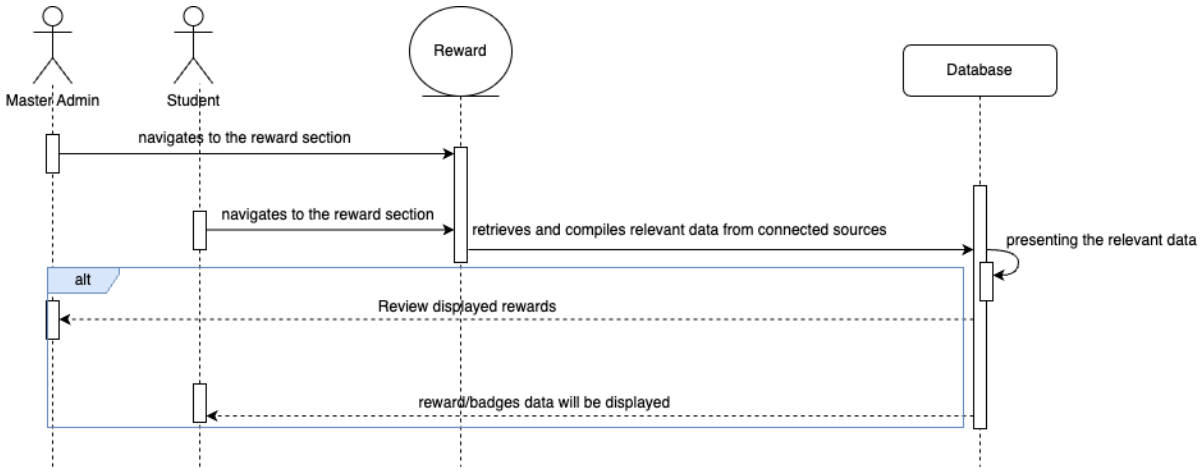

Figure 2.2.7.2.1: Sequence Diagram of View Reward

b) SD019:*Sequence Diagram of Create Reward*

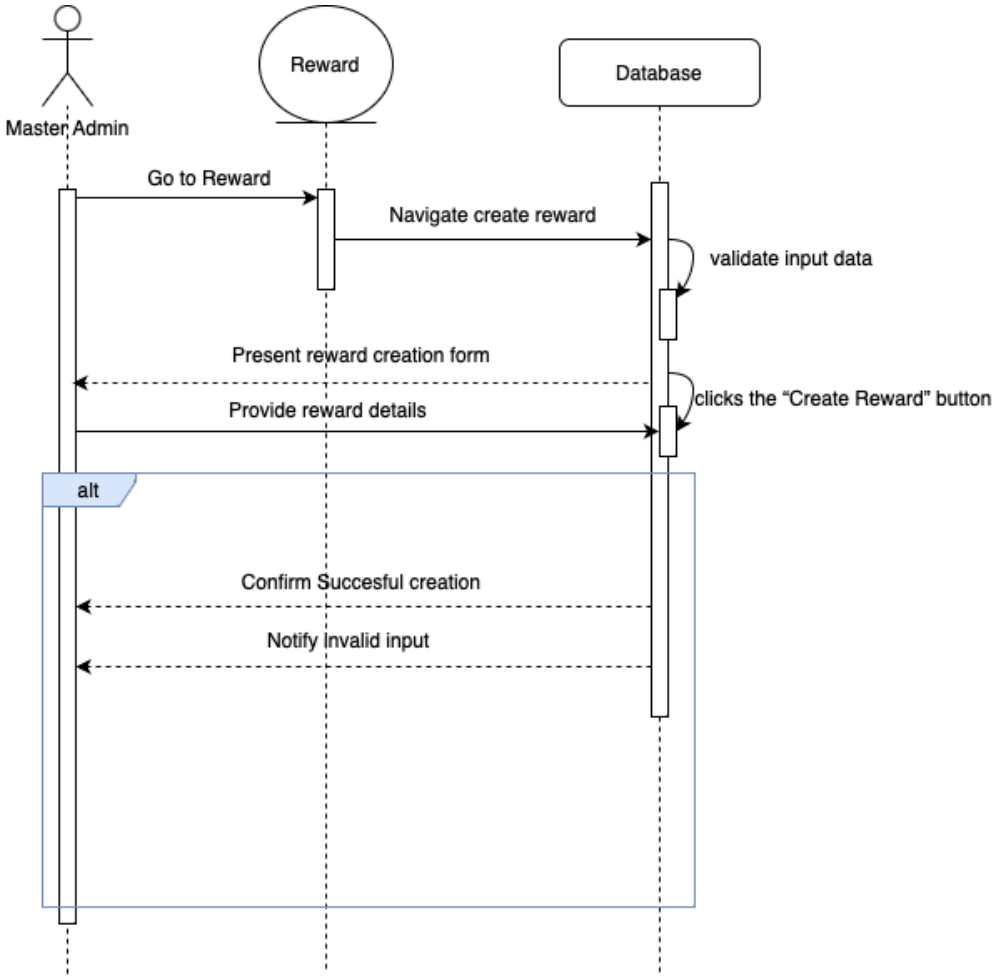

Figure 2.2.7.2.2: Sequence Diagram of Create Reward

c) SD20:*Sequence Diagram of Give Reward*

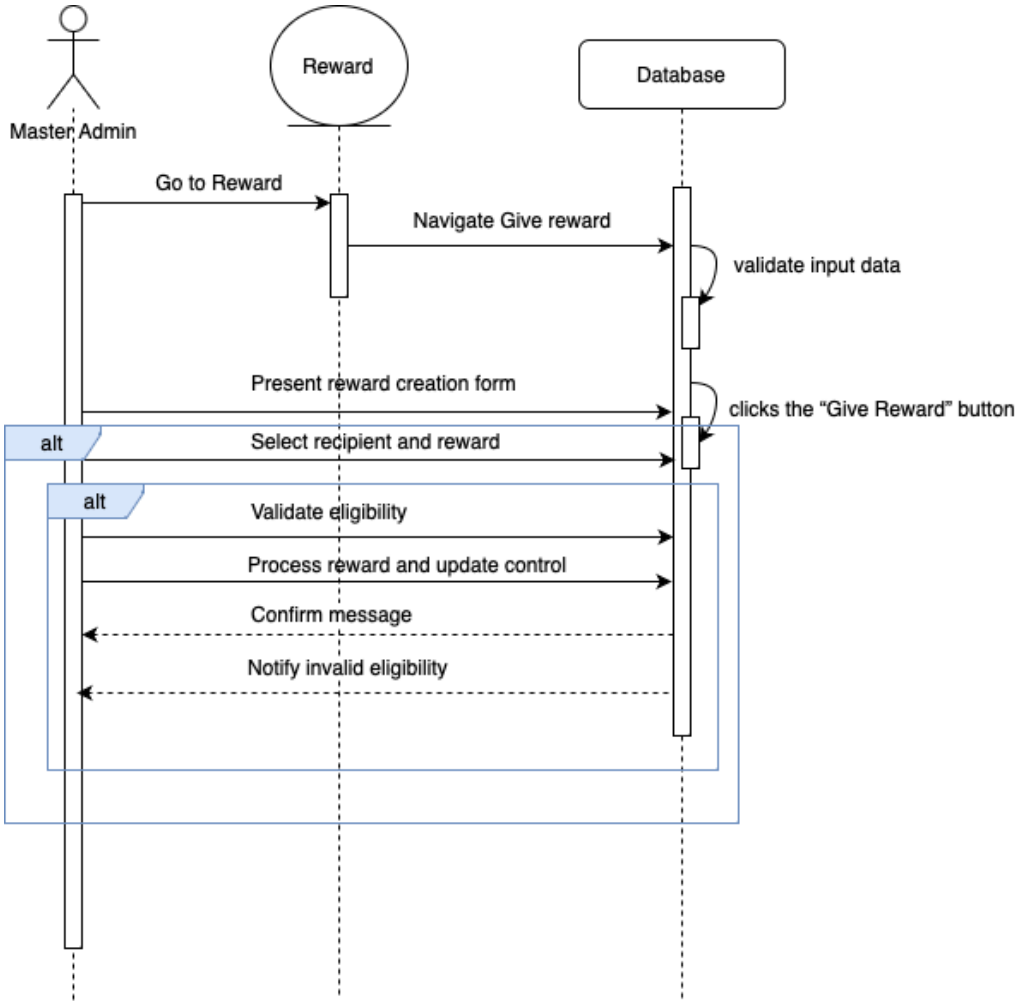

Figure 2.2.7.2.3: Sequence Diagram of Give Reward

2.2.8 P008: <Resume> Subsystem

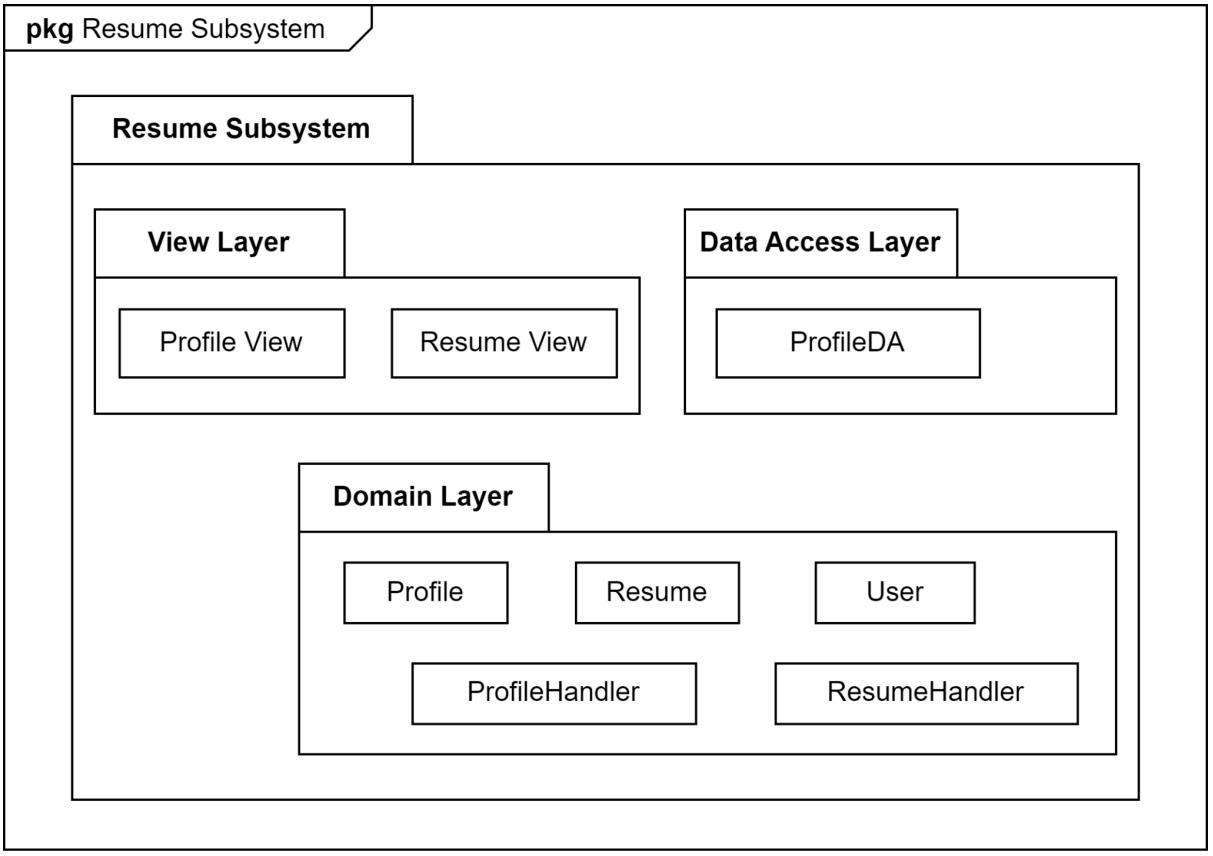

Figure 2.2.8.1: Package Diagram for <Resume> Subsystem

2.2.8.1 Class Diagram

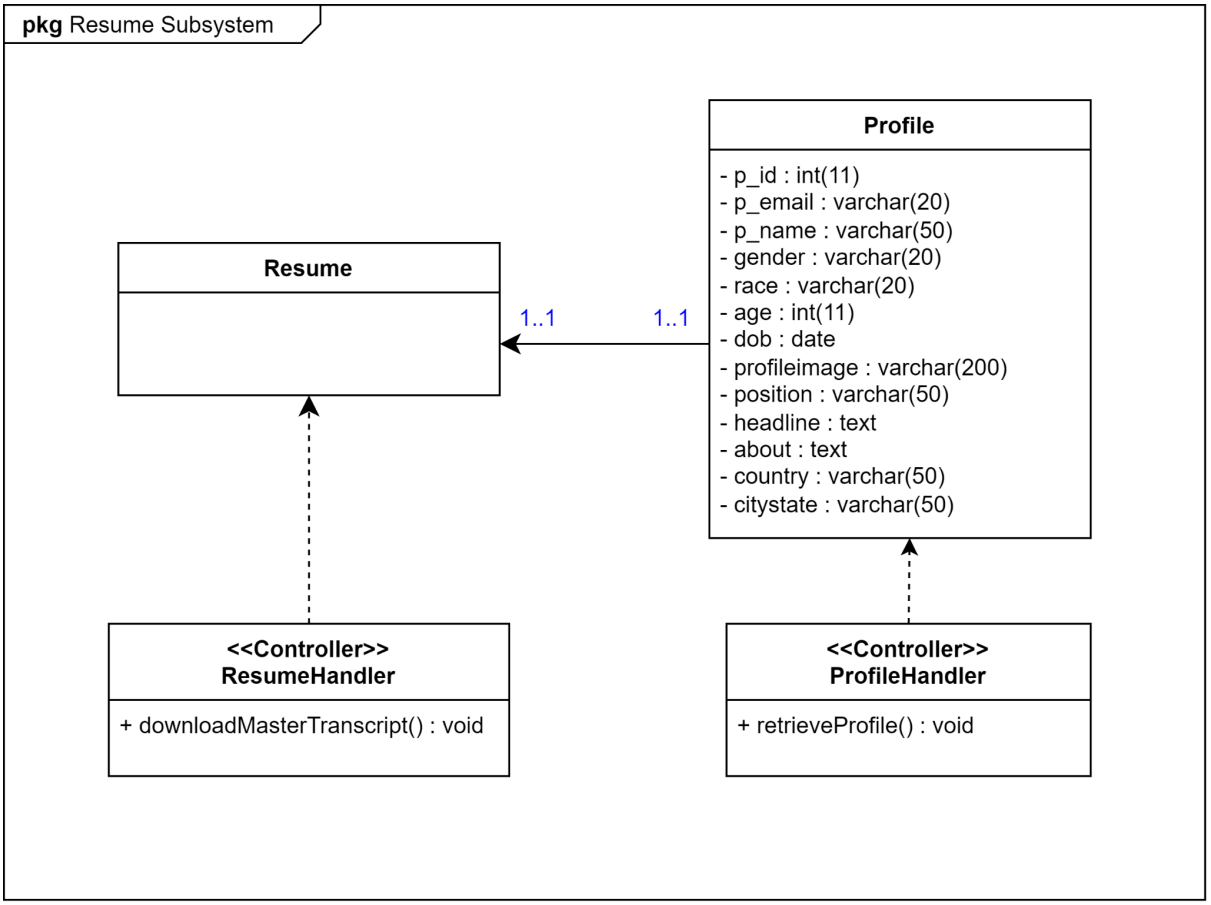

Figure 2.2.8.1.1: Class Diagram of Resume

Entity Name	Resume
Method Name	downloadMasterTranscript
Input	None
Output	Resume with all profile information of the user in the system.
Algorithm	1. Start. 2. If the user is Student 2.1. Select the "Download Resume" option. 2.2. Retrieve all information from ProfileDA. 2.3. Display the preview of the resume to be downloaded. 2.4. Select the “Download” button.

	2.5. Confirm the download. 2.6. Download the master transcript. 2.7. Display success message. 3. End

2.2.8.2 Sequence Diagram

a) SD021: Sequence Diagram for Download Master Transcript

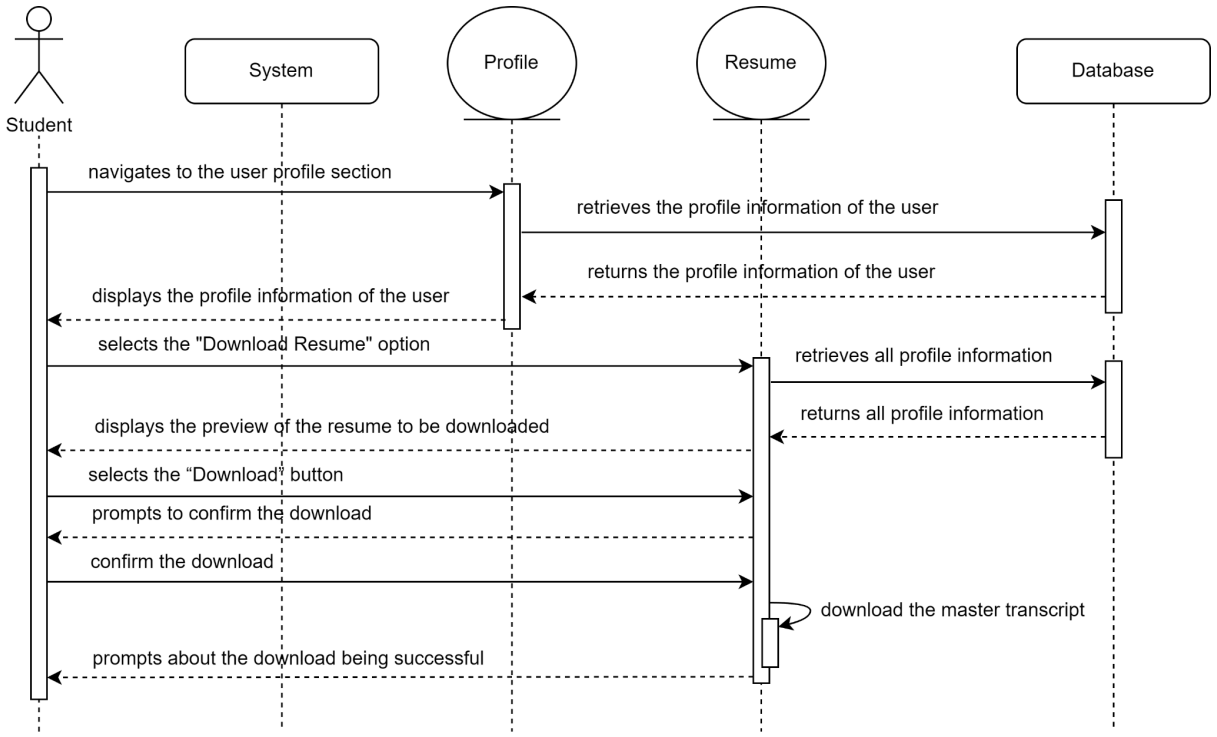

Figure 2.2.8.2.1: Sequence Diagram of Download Master Transcript

2.2.9 P009: <Personal Activity> Subsystem

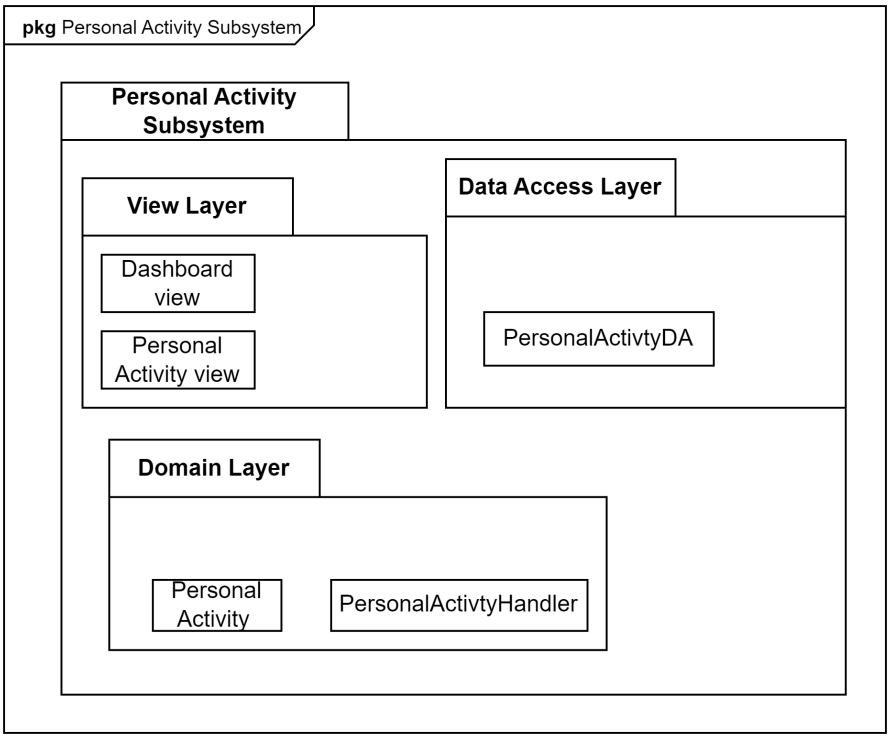

Figure 2.2.9.1: Package Diagram for **<Personal Activity>** *Subsystem*

2.2.9.1 Class Diagram

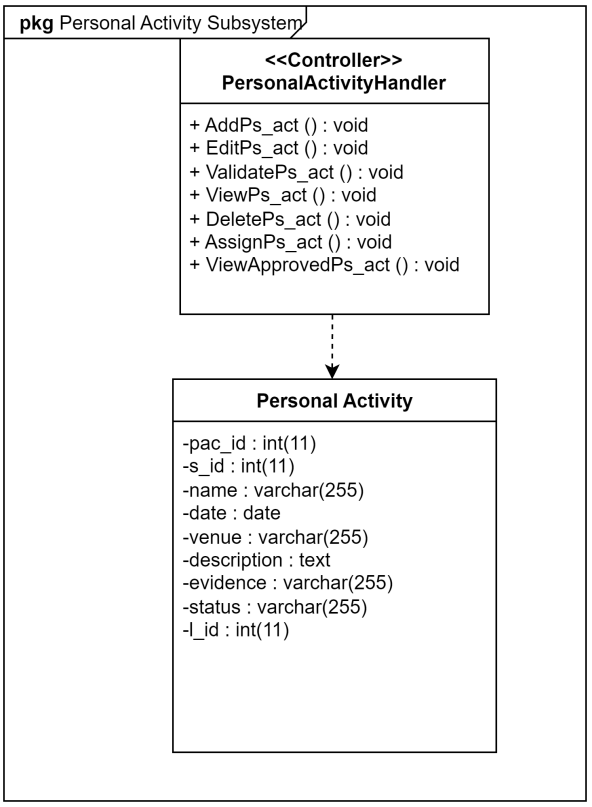

***Figure 2.2.9.1.1: Class Diagram of* Personal Activity**

Entity Name	per_activity
Method Name	AddPs_act
Input	- Event Title, Name of Event Organizers, Date, Duration, Location , Mode, Event description, Evidence
Output	-
Algorithm	1. Start 2. Check if the Student is logged into the system. 3. Student navigates to the profile section on the dashboard. 4. Student scrolls down to the personal activity part. 5. Student selects and clicks the "Create" button option.

	6. Provide input fields for the Student to input details such as Event Title, Name of Event Organizers, Date, Duration, Location, Mode, Event description, and evidence. 7. Student fills in the required information. 8. Student clicks the “Submit” button. 9. Validate if all required information is provided. 10. If any required information is missing, prompt the Student to provide the necessary details. 11. If all information is provided, create the personal activity in the system. 12. Postconditions are met: The personal activity is created and visible in the system. 13. End

Entity Name	per_activity
Method Name	EditPs_act
Input	-
Output	
Algorithm	1. Start 2. Check if the Student is logged into the system. 3. Student navigates to the personal activity section on the dashboard. 4. Display a list of personal activities that are already created by the Student. 5. Student selects and clicks the personal activity that they want to edit. 6. Display all the information for the relevant personal activity. 7. Student clicks the "Update" button option. 8. Student scrolls to the parts they want to edit. 9. Student makes necessary changes to the activity details.

	10. Student clicks the "Submit" button. 11. Validate if all required information is provided. 12. If no personal activity is created, notify the Student. 13. If all information is provided, update the personal activity details in the system. 14. Postconditions are met: The personal activity is updated and visible in the system. 15. End

Entity Name	per_activity
Method Name	ValidatePs_act
Input	-
Output	-
Algorithm	1. Start 2. Check if the Lecturer is logged into the system. 3. Lecturers navigate to the "Manage Personal Activities" page. 4. Display the details of the assigned personal activities listed on the page. 5. For each activity: 5.1 Lecturers assess the evidence provided. 5.2 Lecturers have the option to either "Approve" or "Reject" the activity. 5.2.1 If they choose to "Approve," they click the "Approve" button. 5.2.1.1 The system updates the status of the activity to "Approved." 5.2.2 If they choose to "Reject," they click the "Reject" button.

	5.2.2.1 The system updates the status of the activity to "Rejected." 6. Lecturers continue assessing and approving/rejecting assigned personal activities as needed. 7. If no personal activity is assigned, notify the Lecturer. 8. Postconditions are met: The personal activity is validated and visible in the approved personal activity list. 9. End

Entity Name	per_activity
Method Name	ViewPs_act
Input	-
Output	-
Algorithm	1. Start 2. Check if the Actor (either Student or Lecturer) is logged into the system. 3. Actor selects the "Manage Student Personal Activity" option. 4. Display the details of the Student's personal activities to the Actor. 5. Postconditions are met: The Actor successfully views the personal activity. 6. If no personal activity is found for the Student, notify the Actor. 7. No specific input is required for this operation, as the system retrieves and displays the relevant personal activity data based on the Actor's request. 8. End

Entity Name	per_activity
Method Name	DeletePs_act

Input	-
Output	-
Algorithm	1. Start 2. Check if the Student is logged into the system. 3. Student navigates to the "Manage Personal Activities" page. 4. Student selects the personal activity they wish to delete. 5. Student clicks on the "Delete" button. 6. System displays a confirmation message asking "Are you sure you want to delete this activity?" 7. If the Student confirms deletion: 7.1 System removes the selected personal activity from the list. 7.2 System displays a message confirming the deletion. 8. Postconditions are met: Personal activity is deleted. 9. Alternative flow 1: 9.1 If the Student clicks "Close" or cancels the deletion: 9.2 System closes the confirmation message without deleting the activity. 10. Postconditions: Personal activity is not deleted. 11. If no personal activity is found, notify the Student. 12. No specific input is required for this operation, as the system identifies the selected personal activity for deletion based on the Student's interaction. 13. End

Entity Name	per_activity
Method Name	AssignPs_act
Input	-
Output	-

Algorithm	1. Start 2. Check if the Student is logged into the system. 3. Student navigates to the "Manage Personal Activities" page. 4. Student clicks the "Assign" button in the Action column for the desired personal activity. 5. System displays a list of available lecturers. 6. Student selects the desired lecturer from the list. 7. System updates the 'Lecturer' column in the table with the selected lecturer's name. 8. System sends the details of the personal activity to the assigned lecturer for validation and approval. 9. Postconditions are met: Personal activity is assigned to the selected lecturer. 10. If no personal activity is found, notify the Student. 11. No specific input is required for this operation, as the system identifies the selected personal activity for assignment based on the Student's interaction. 12. End

Entity Name	per_activity
Method Name	ViewApprovedPs_act
Input	-
Output	-
Algorithm	1. Start 2. Check if the Student is logged into the system. 3. Student accesses the dedicated page for viewing approved personal activities. 4. System retrieves and displays a list of approved personal activities.

	5. The list includes details such as Personal Activity Name, Description, Date, Venue, Evidence, and a 'Status' column indicating "Approved." 6. Postconditions are met: Approved personal activities are viewed. 7. If no approved personal activities are found, the system notifies the Student. 8. No specific input is required for this operation, as the system automatically retrieves and displays the approved personal activities based on the Student's access. 9. End

2.2.9.2 Sequence Diagram

a) SD022: Sequence Diagram for Add Student Personal Activity

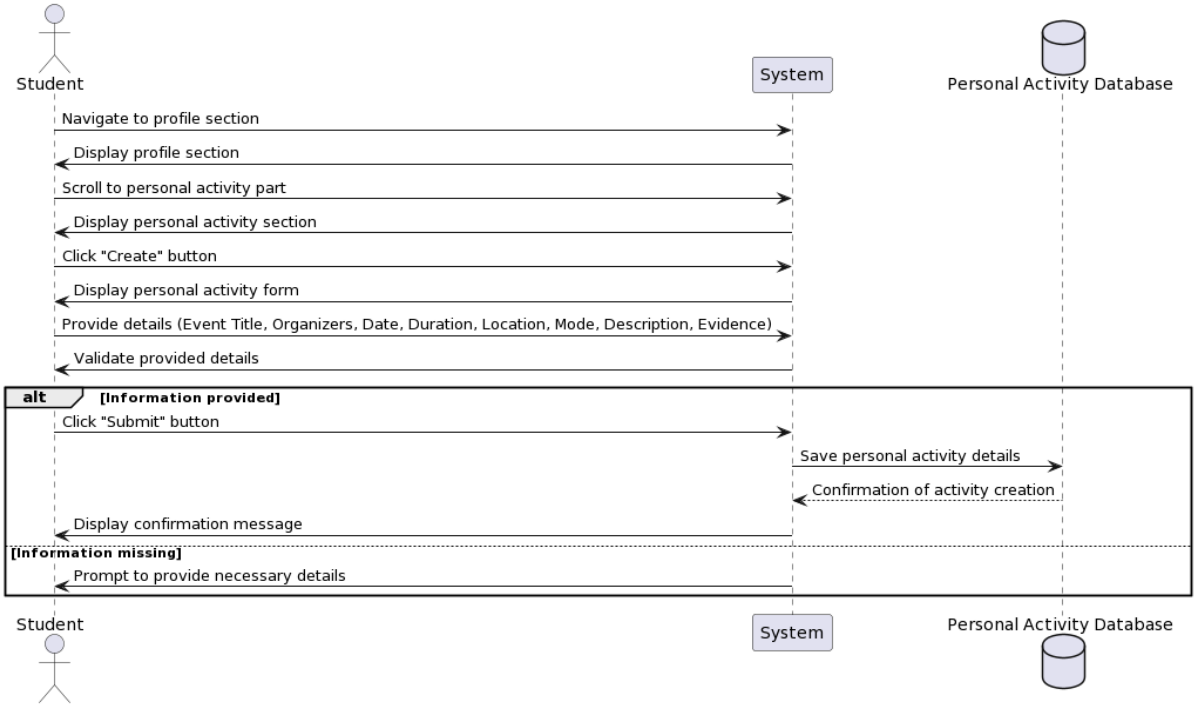

Figure 2.2.9.2.1: Sequence Diagram of Add Student Personal Activity

b) SD023: Sequence Diagram for Edit Student Personal Activity

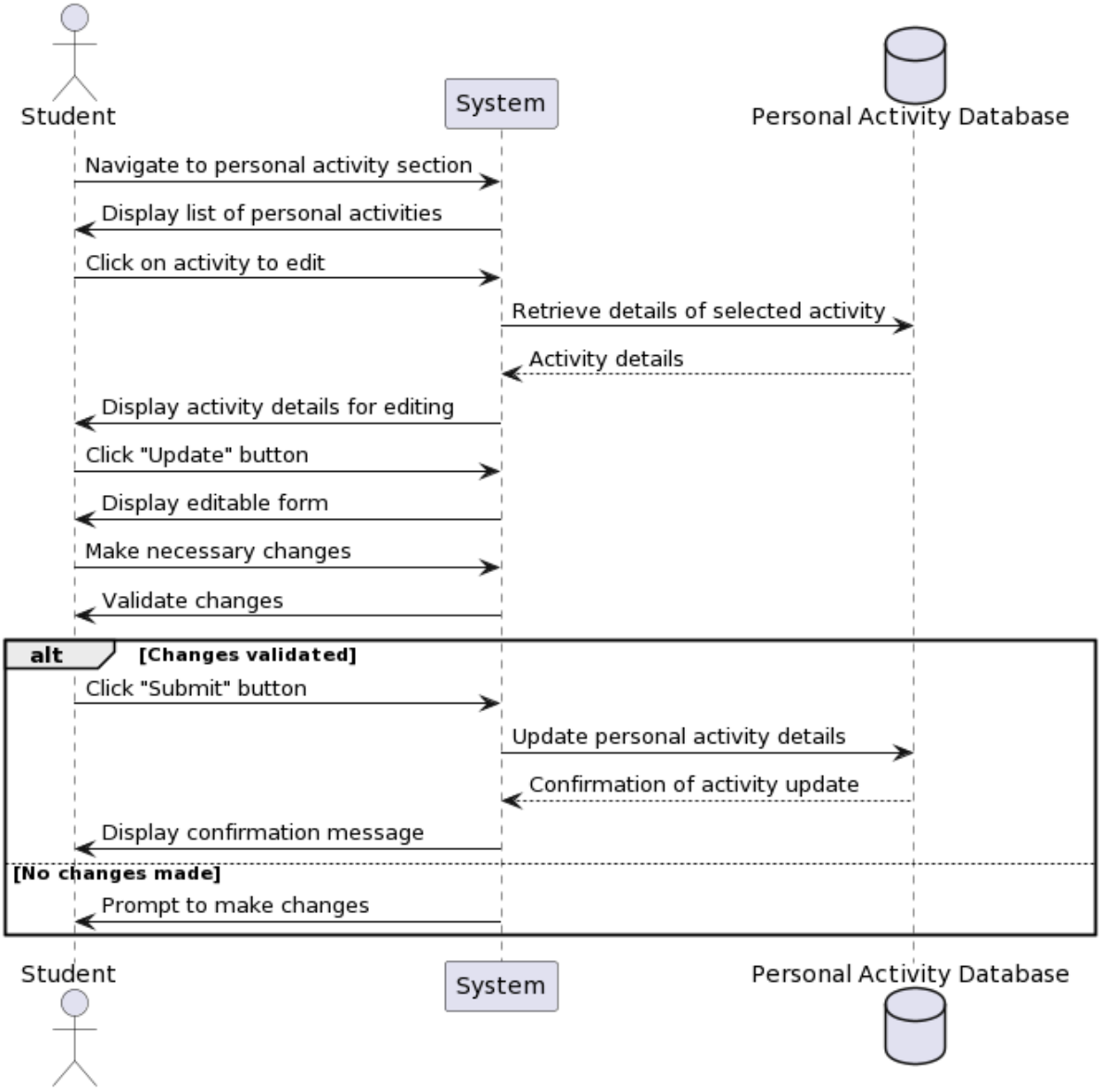

Figure 2.2.9.2.2: Sequence Diagram of Edit Student Personal Activity

c) SD024: Sequence Diagram for Register Activity

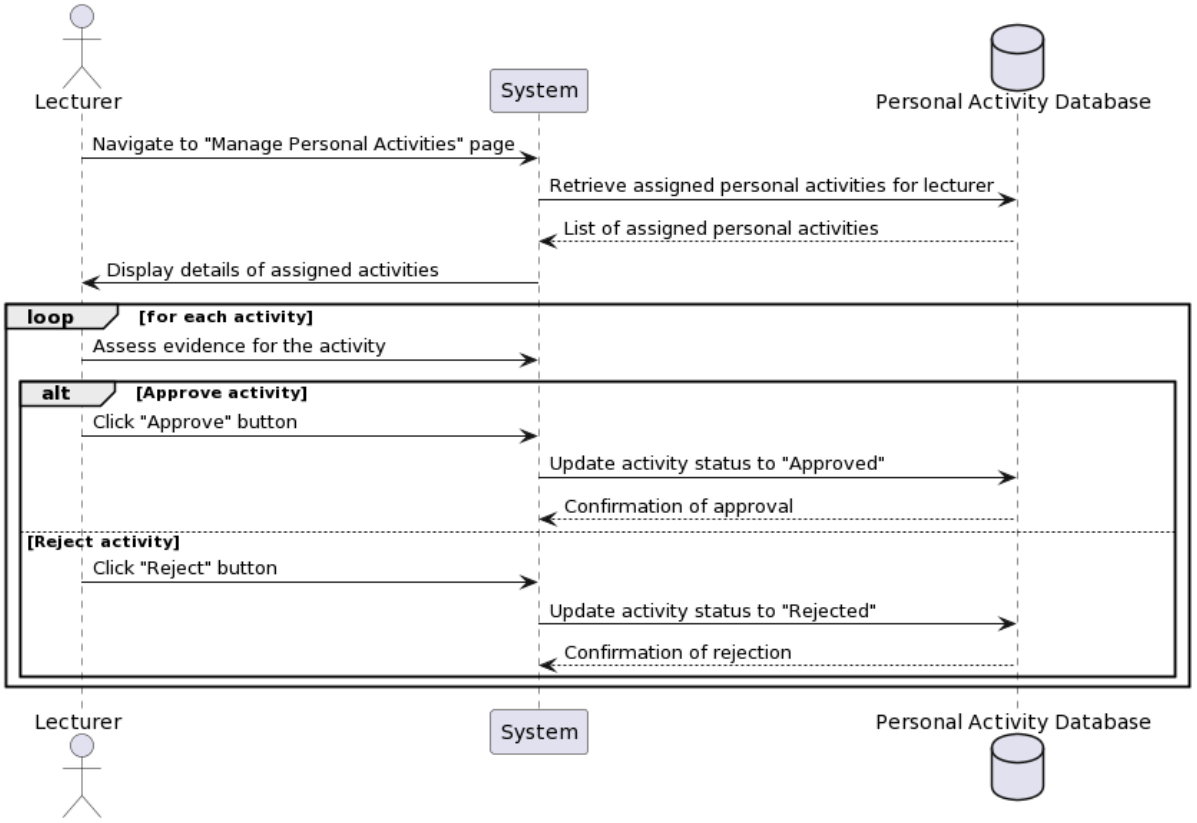

Figure 2.2.9.2.3: Sequence Diagram of Edit Student Personal Activity

d) SD025: Sequence Diagram for View Student Personal Activity

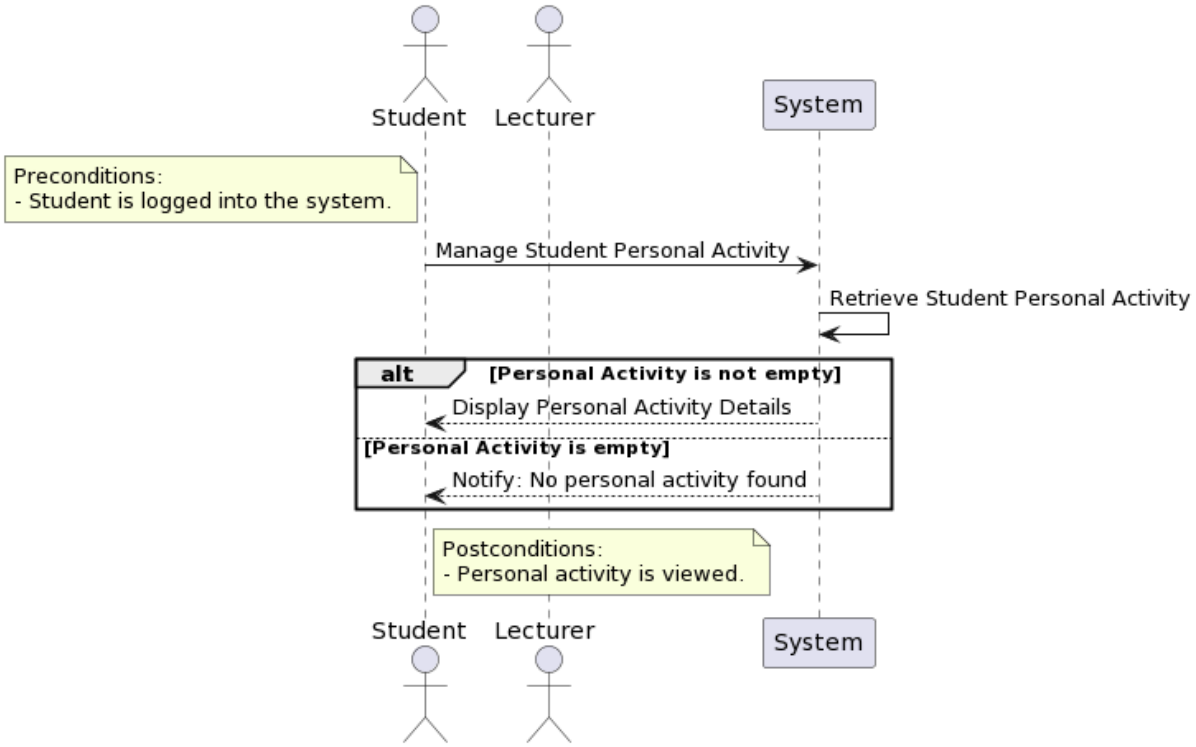

Figure 2.2.9.2.4: Sequence Diagram of View Student Personal Activity

e) SD026: Sequence Diagram for Delete Personal Activity

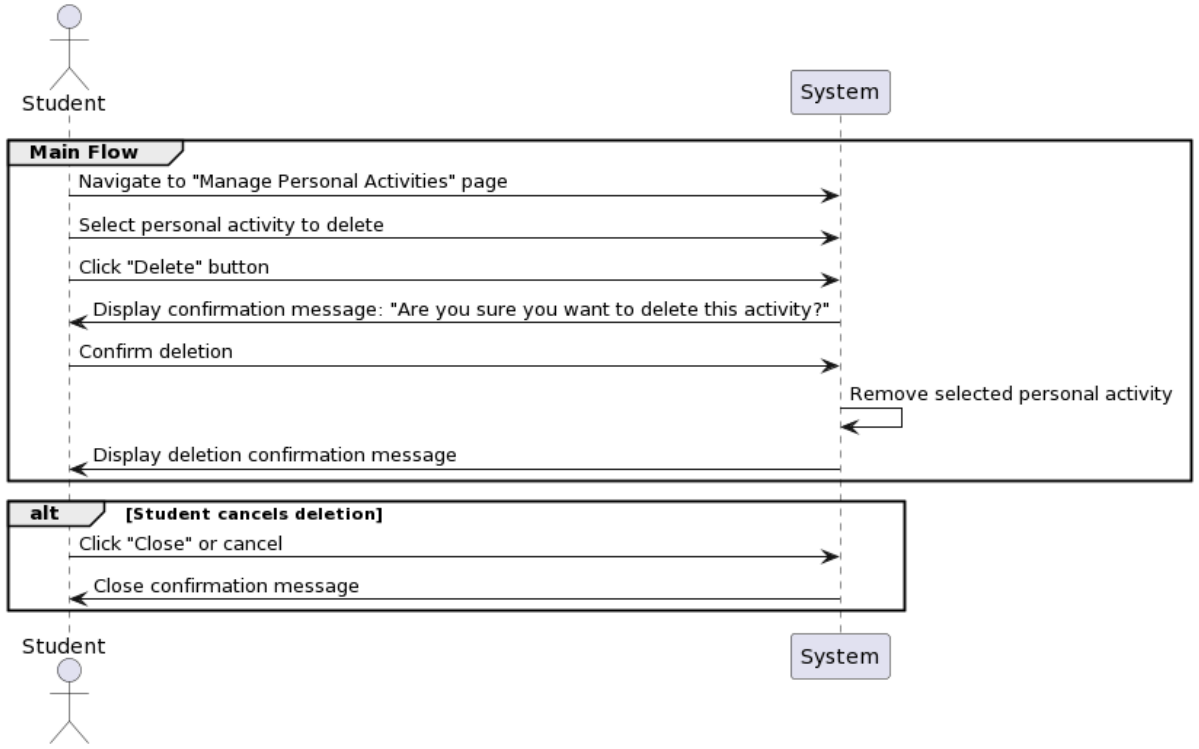

Figure 2.2.9.2.5: Sequence Diagram of Delete Personal Activity

f) SD027: Sequence Diagram for Assign Personal Activity

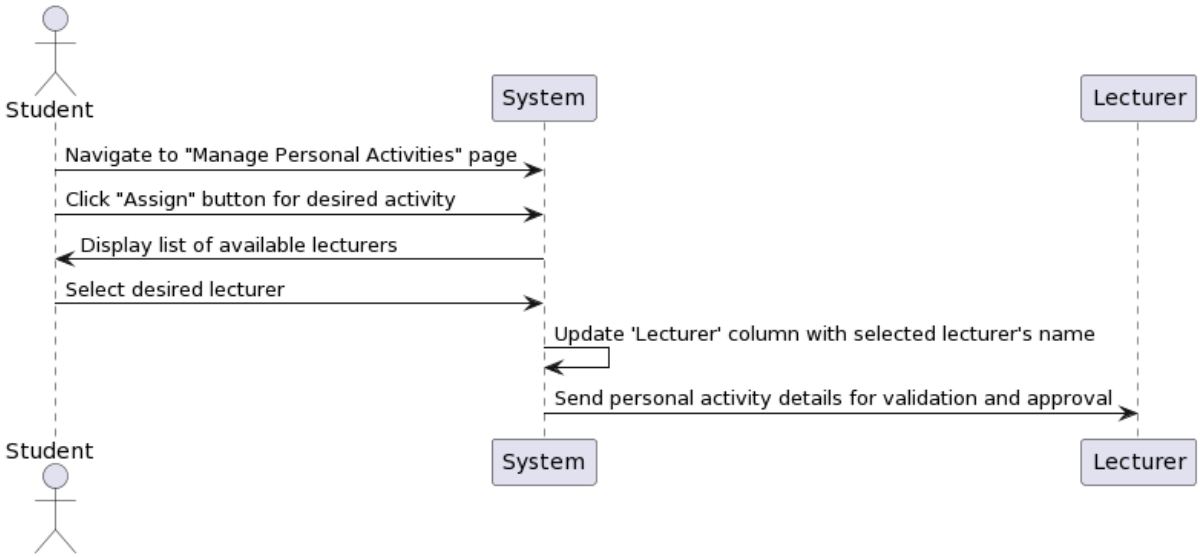

Figure 2.2.9.2.6:Sequence Diagram of Assign Personal Activity

g) SD028: Sequence Diagram for View Approved Personal Activity List

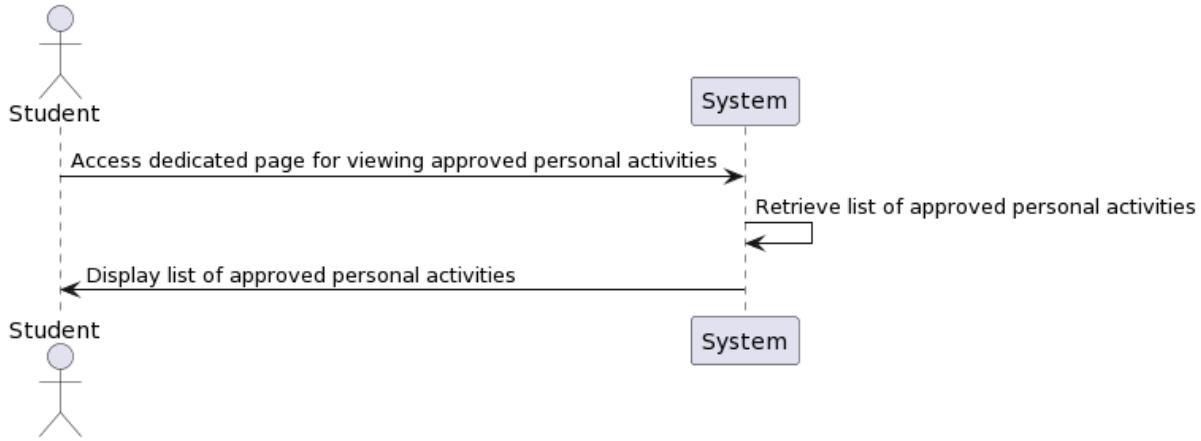

Figure 2.1.28.1: Sequence Diagram of View Approved Personal Activity List

3. Data Design

3.1 Data Description

The major data or systems entities are stored into a relational database named as niagaped_explorer processed and organized into *11* entities as listed in Table 3.1.

Table 3.1: Description of Entities in the Database

No.	Entity Name	Description
1	user	Used to store the information of registered users list on StuPort.
2	student	Used to store the information of registered students list on StuPort.
3	lecturer	Used to store the information of registered lecturers list on StuPort.
4	administrator	Used to store the information of added clients or partners list on StuPort.
5	masteradministrator	Used to store the information of Youth Ventures admins accounts which normally used to handle StuPort.
6	profile	Used to store the information or details of every registered student and lecturer profile.
7	activity	Used to store all activities information which can be created, edited, deleted by Master Admin or Admin and viewed by the students through StuPort.
8	activity_participant	Used to store the activities information which have been registered or participated by the student.
9	feedbacks	Used to store feedback information which need to created by Master Admin or Admin and also filled by each participant through a link for a respective activity.

10	reward	Used to store the rewards details corresponding to every registered student on StuPort.
11	peractivity	Used to store all personal activities information added by the students which can be assigned to lecturer for further validation operation.

3.2 Data Dictionary

3.2.1 Entity: <user>

Attribute Name	Type	Description
id	int(11)	Unique identifier of user
username	varchar(255)	Username of user used for login
password	varchar(255)	Encrypted password of user used for login
email	varchar(255)	Email of user
user_role	varchar(50)	Role of user
datetime_register	datetime	Date Time of registration of user
user_reg_status	varchar(20)	Activation status of user

3.2.2 Entity: <student>

Attribute Name	Type	Description
s_id	int(11)	Unique identifier of student
s_email	varchar(255)	Email of the student
s_fName	varchar(255)	Full name of the student
s_telephone_no	varchar(20)	Telephone number of the student
s_address	varchar(255)	Residential address of the student
s_institution	varchar(255)	Institution of the student
s_course	varchar(255)	Course registered by the student
s_skills	varchar(255)	Skills of the student
s_hobby	varchar(255)	Hobby of the student
s_achievement	varchar(255)	Achievement made by the student
s_ambition	varchar(255)	Ambition of the student
s_academic_cert	varchar(255)	Academic certificate of the student

s_cocurriculum_cert	varchar(255)	Co-curricular certificate of the student
s_gender	varchar(255)	Gender of the student
s_race	int(11)	Race of the student
s_age	varchar(255)	Age of the student

3.2.3 **Entity: <lecturer>**

Attribute Name	**Type**	**Description**
l_id	int(11)	Unique identifier of lecturer
l_email	varchar(255)	Email of the lecturer
l_fName	varchar(255)	Full name of the lecturer
l_telephone_no	varchar(15)	Telephone number of the lecturer
l_address	varchar(255)	Residential address of the lecturer
l_institution	varchar(255)	Institution of the lecturer
l_gender	varchar(10)	Gender of the lecturer
l_race	varchar(255)	Race of the lecturer
l_age	int(11)	Age of the lecturer

3.2.4 **Entity: <administrator>**

Attribute Name	**Type**	**Description**
a_id	int(11)	Unique identifier of administrator

3.2.5 **Entity: <masteradministrator>**

Attribute Name	**Type**	**Description**
ma_id	int(11)	Unique identifier of master administrator

3.2.6 **Entity: <profile>**

Attribute Name	**Type**	**Description**
p_id	int(11)	Unique profile information identifier of user
p_email	varchar(20)	Email of the user as registered
p_name	varchar(50)	Name of the user shown in profile
gender	varchar(20)	Gender of the user shown in profile
race	varchar(20)	Race of the user shown in profile
age	int(11)	Age of the user shown in profile
dob	date	Date of Birth of the user shown in profile
profileimage	varchar(200)	Profile image of the user shown in profile
position	varchar(50)	Position of the user shown in profile
headline	text	Headline of the user shown in profile
about	text	About of the user shown in profile
country	varchar(50)	Country of the user in shown in profile
citystate	varchar(50)	City and state of the user in shown in profile

3.2.7 **Entity: <activity>**

Attribute Name	**Type**	**Description**
activity_id	int(11)	Unique serial number for each activity or event
title	varchar(255)	Title of the activity
category	varchar(50)	Type of the activity
act_datetime	datetime	Date and Time of the activity

location	varchar(255)	Location of the activity
organizer_name	varchar(255)	Name of the organizer of the activity
skill_acquired	varchar(255)	Skills gained from the activity by the student
activity_desc	text	Description of the activity
attachment	varchar(255)	Attachment of the activity
validation_status	varchar(20)	The validation status for the activity attended by student whether valid or invalid
link_form	varchar(200)	feedback form link
user_id	int(11)	user's id

3.2.8 **Entity: <activity_participant>**

Attribute Name	**Type**	**Description**
participant_id	int(11)	Unique serial number for each participant
activity_id	int(11)	Unique serial number for each activity
s_id	int(11)	students id
s_email	varchar(255)	students email
s_fName	varchar(255)	students full name
s_gender	varchar(10)	students gender
s_institution	varchar(10)	students institution of study
s_address	varchar(255)	students address

3.2.9 **Entity: <reward>**

Attribute Name	**Type**	**Description**
reward_id	int(11)	Rewards details correspond to unique serial number
badge_name	varchar(255)	Name of the badges that provided in the system
badge_description	text	Description of the badges

badge_icon_path	varchar(255)	Image path
points_required	int(11)	The number of activities required to acquire different type of badges

3.2.10 **Entity: <feedbacks>**

Attribute Name	**Type**	**Description**
feedback_id	int(11)	Unique serial numbers have been assigned to each feedback link uploaded or published for an activity.
activity_id	int(11)	Feedback details correspond to unique serial number for each activity or event
link_form	varchar(150)	Link of the feedback form given after the activity has been organized
publish_date	datetime	Timestamp of publication of feedback link for a specific activity

3.2.11 **Entity: <per_activity>**

Attribute Name	**Type**	**Description**
pac_id	int(11)	Unique serial numbers personal activity.
s_id	int(11)	Student id
name	varchar(255)	Personal activity name
date	date	Personal activity date
venue	varchar(255)	Personal activity venue
description	text	Personal activity description
evidence	varchar(255)	Personal activity evidence
status	varchar(50)	Personal activity validation status
l_id	int(11)	Lecturer’s id

Appendices

1. Higher Definition System Architectural Design of StuPort System Diagram:

   SDD-Architecture Style and Rationale.jpg

2. Higher Definition Component Diagram of StuPort System Diagram:

   SDD-Component Model.jpg

Log Book #1

FACULTY OF COMPUTING
UTM Johor Bahru

SECP2613 – System Analysis & Design (WBL)

Log Meeting

MEETING AGENDA

Group Name:	Explorer	**Time:**	2:00 p.m. - 4:00 p.m.
Date of Meeting: (DD/MM/YYYY)	14/12/2023	**Location:**	Online Meeting (Google Meet)

1. Meeting Objective
- Discuss and finalize SDD documentation for the Student Portfolio Management System. - Ensure a comprehensive and well-documented plan for system architecture and components.

2. Attendees

Name	**Department/Division**	**E-mail**	**Phone**
KOH LI HUI	School of Computing	kohhui@graduate.utm.my	012-768 2618
KOH SU XUAN	School of Computing	koh.xuan@graduate.utm.my	011-1077 8126
LEE YIK HONG	School of Computing	lee.hong@graduate.utm.my	018-378 8589
VINESH A/L VIJAYAKUMAR	School of Computing	vinesh03@graduate.utm.my	012-346 5289

3. Meeting Agenda

Topic	**Person in Charge (PIC)**	**Action**
- System Architecture Discussion	Koh Li Hui	- Detailed discussion on proposed system architecture. - Consideration of scalability, flexibility, and integration points.
- Functionalities and Modules Breakdown	All members	- The client (Mr. Hanif) provides clear answers to questions regarding their issues and needs.
- Database Design and Data Management	Koh Su Xuan	- All members provide their point of view on the proposed system requirements and its functionalities.
- Task Distribution	Al members	- Assignment of responsibilities for

		documentation task among group members.

4. Meeting Reflection
The meeting was instrumental in aligning team understanding and goals for the StuPort software design documentation (SDD). We clarified system architectures, and assign responsibilities, fostering a collaborative atmosphere. The discussion on functionalities and documentation standards was particularly insightful. With this, we are well-positioned to refine our documentation with clarity and consistency, promoting effective communication throughout the development process.

Prepared by:

……

(Koh Li Hui)

Approved by:

……

(Signature & Stamp)