SECP2613: SYSTEM ANALYSIS AND DESIGN WBL

Software Requirements Specifications (SRS)

Youth Ventures Student Portfolio Management System (StuPort)

Version 1.0

Date: 15 November 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 1.0 of SD only covers System Requirements Specification (SRS) of Youth Ventures Student Portfolio Management System (StuPort) which provides a detailed understanding of what the system will accomplish, how it will behave and what constraints of the system by clarifying the documentation including but not limited to the purpose, scope, system features, design constraints and software system attributes.

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	- **Module 003: Dashboard Module**	Complete
		- **Module 006: Feedback Module**	
		- Section 2.2	
		- Section 2.3	
		- Section 2.4	
KOH SU XUAN	- Database Administrator (DBA) - Quality Assurance (QA) Tester	- **Module 002: Profile Module**	Complete
		- **Module 007: Resume Module**	
		- Revision Page	
		- Table of Contents	
		- Section 1.1	
		- Section 1.2	
LEE YIK HONG	- System Analyst - Documentation Specialist	- **Module 001: Authentication Module**	Complete
		- **Module 004: Activity Module**	
		- **Module 005: Registration Module**	

		- **Module 009: Personal Activity Module**	
		- Section 2.1	
VINESH A/L VIJAYAKUMAR	- Developer	- **Module 007: Rewards Module**	Complete
		- Section 1.3	
		- Section 1.4	
		- Section 1.5	

d. **Version Control History**

Version	**Primary Author(s)**	**Description of Version**	**Date Completed**
1.0	KOH LI HUI	Completed Section 2.2, Section 2.3, Section 2.4	27/11/2023
1.0	KOH LI HUI	Completed Module 003, Module 006	30/11/2023
1.0	KOH SU XUAN	Completed Module 002, Module 007	02/12/2023
1.0	LEE YIK HONG	Completed Module 001, Module 004, Module 005, Module 009	02/12/2023
1.0	VINESH A/L VIJAYAKUMAR	Completed Module 007	03/12/2023
1.0	LEE YIK HONG	Completed Section 2.1	10/12/2023
1.0	VINESH A/L VIJAYAKUMAR	Completed Section 1.3, Section 1.4, Section 1.5	11/12/2023
1.0	KOH SU XUAN	Completed Revision Page, Table of Contents, Section 1.1, Section 1.2	12/12/2023

Table of Contents

1	**Introduction**			**7**
	1.1	**Purpose**		**7**
	1.2	**Scope**		**7**
	1.3	**Definitions, Acronyms and Abbreviations**		**8**
	1.4	**References**		**9**
	1.5	**Overview**		**9**
2	**Specific Requirements**			**13**
	2.1	**System Features**		**13**
		2.1.1	UC001: Use Case <Log In>	25
		2.1.2	UC002: Use Case <Log Out>	30
		2.1.3	UC003: Use Case <Sign Up>	32
		2.1.4	UC004: Use Case <Create Profile>	38
		2.1.5	UC005: Use Case <View Profile>	42
		2.1.6	UC006: Use Case <Edit Profile>	45
		2.1.7	UC007: Use Case <Delete Profile>	49
		2.1.8	UC008: Use Case <View Dashboard>	52
		2.1.9	UC009: Use Case <Publish Event>	56
		2.1.10	UC0010: Use Case <Edit YV Activity Details>	59
		2.1.11	UC011: Use Case <View Activity>	63
		2.1.12	UC012: Use Case <Register Activity>	66
		2.1.13	UC013: Use Case <View Register Activity>	68
		2.1.14	UC014: Use Case <Create Feedback Form>	71
		2.1.15	UC015: Use Case <Fill Feedback Form>	74
		2.1.16	UC016: Use Case <View Feedback History>	77

		2.1.17	UC017: Use Case <View Student Feedback List>	79
		2.1.18	UC018: Use Case <View Reward>	83
		2.1.18	UC019: Use Case <Create Reward>	86
		2.1.20	UC020: Use Case <Give Reward>	90
		2.1.21	UC021: Use Case <Download Master Transcript>	94
		2.1.22	UC022: Use Case <Add Student Personal Activity>	98
		2.1.23	UC023: Use Case <Edit Student Personal Activity>	101
		2.1.24	UC024: Use Case <Validate Student Personal Activity>	104
		2.1.25	UC025: Use Case <View Student Personal Activity>	107
		2.1.26	UC026: Use Case <Delete Personal Activity>	109
		2.1.27	UC027: Use Case <Assign Personal Activity>	112
		2.1.28	UC028: Use Case <View Approved Personal Activity List>	115
	2.2	**Performance and Other Requirements**		**117**
	2.3	**Design Constraints**		**118**
	2.4	**Software System Attributes**		**119**
Appendices				**121**

1. Introduction

1.1. Purpose

This SD describes the functionality, features and interactions within the system. It aims at providing a comprehensive understanding of the system's modules, actors and their respective roles in various use cases. This is achieved through developing the use case description, sequence diagram and activity diagram for each use case to showcase how the actors will interact with the system in each situation and each feature of the system.

The intended audience for this SD includes but is not limited to Client (Youth Ventures Asia), the end users of the system (Youth Ventures Asia's Clients - Partners, Organizers, Students, Lecturers etc.), Project Manager, Designer, Database Administrator (DBA), Quality Assurance (QA) Tester, System Analyst, Documentation Specialist and Developer. Each audience will utilize SD as mentioned above in Revision Page b. Target audience. In summary, SD will provide clarity and guidance in the system's development, maintenance, evaluation and usage.

1.2. Scope

The software product to be produced is a web-based management system designed for Youth Ventures that able to efficiently collect and manage students information and empower master administrators and admin to create more effective programs and activities by the analytic features on the level of satisfaction towards the programs and activities filled by the students, named Youth Ventures Student Portfolio Management System, also known as ***StuPort***.

StuPort will be equipped with several main features aligned with the user requirements, which are:

I. User Authentication and Profile Management
II. Dashboard Presentation
III. Activity and Personal Activity Management
IV. Feedback Handling and Reward Allocation

V. Resume Management,

each of the features will be involved by the users of the StuPort who are the master administrator, administrator and students. In contrast, StuPort will not involve any financial transactions primarily, as well as it will not replace or act as a primary educational platform for academic courses.

StuPort is developed and aimed to be utilized by students, admin from partners of Youth Ventures and master admin from Youth Ventures by enabling the following benefits, objectives and goals:

I. Efficient Student Management - Streamlining the process of collecting students' information who registered in Youth Ventures and ease of sharing the students' information with Youth Ventures's clients when needed.
II. Enhanced Data Management - Centralised the data collected leading to an easy data retrieval and management.
III. Improved Collaboration - Facilitating communication and collaboration between Youth Ventures and their clients, in terms of the management of activities, registration, feedback and reward within the system.

In conclusion, the system, StuPort, will be developed aligned with higher-level specifications as expected by the client, Youth Ventures Asia, and serves as a foundational reference for the System Requirements Specification (SRS) to ensure consistency across all detailed system specifications and client's expectations.

1.3. Definitions, Acronyms and Abbreviation

Term	Definition
Youth Ventures Student Portfolio Management System	The software products being developed for the YouthVentures admin to view, manage, and organize registered students activity in the whole country.
SD	System Documentation
SRS	System Requirements Specification -

	A document that outlines the requirements and objectives of the software product being developed.
UC	Use Case
QA	Quality Assurance
YV	Youth Ventures
StuPort	Student Portfolio Management System

1.4. References

1. Department of English, Linguistics, and Writing Studies at SLCC. (2020, September 10). *Software Requirements Specification (SRS)*. Pressbooks. https://slcc.pressbooks.pub/technicalwritingatslcc/chapter/software-requirements-specification-srs/
2. Sinanaj, E. (2022, March 24). Writing a software requirements specification document. *Medium*. https://enisinanaj.medium.com/writing-a-software-requirements-specification-document-97d622805aef#:~:text=References%20is%20about%20listing%20any,a%20vision%20and%20scope%20document.
3. Grafiati. (2022, February 1). *Academic literature on the topic "Software requirements specification (SRS)."* Grafiati. https://www.grafiati.com/en/literature-selections/software-requirements-specification-srs/

1.5. Overview

The System Documentation (SD) serves as a comprehensive guide for the Youth Ventures Student Portfolio Management System (StuPort). It encompasses various aspects crucial for understanding, developing, and maintaining the system, promoting effective communication,

knowledge sharing, troubleshooting, compliance with standards and ensuring long-term system stability and evolution.

Contents of the SD:

1. Understanding the System:

Detailed insights into system functionalities, scope, and limitations. Clarification of documentation, including purpose, features, design constraints and software system attributes.

2. Development and Maintenance:

Guidance for developers through technical specifications, coding guidelines, and integration requirements. Use of architectural diagrams and design constraints to support system development.

3. Effective Communication:

Facilitation of clear communication among project team members, clients, and end-users. Structured information for documentation specialists to create user manuals and guides.

4. Knowledge Sharing:

Centralized resource for project team members to understand system architecture, functionalities, and dependencies. Reference point for sharing essential details with clients and end-users.

5. Troubleshooting:

In-depth information for quality assurance testers to create test cases and scenarios. Identification of potential risks for system analysts and proposed solutions.

6. Compliance with Standards:

Adherence to established standards through detailed technical specifications and coding guidelines. Reference for designers to ensure user interfaces align with specified standards.

7.System Stability and Evolution:

Overview of version control history to track developments and completed tasks. Enables efficient adaptation and enhancement over time.

Organization of the SD:

Target Audience:

Clearly defines the stakeholders who will benefit from the SD and how they will utilize it.

Project Team Members:

Lists team members, their roles, assigned tasks, and current status of completion.

Version Control History:

Tracks the evolution of the SD, specifying versions, primary authors, descriptions, and completion dates.

Table of Contents:

Provides a structured outline for easy navigation within the document.

Introduction:

Describes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the SD.

Specific Requirements:

Outlines system features, performance requirements, design constraints, and software system attributes.

Conclusion:

Summarizes the overarching goals of the SD and its significance in the development and maintenance of StuPort.

The SD is meticulously organized to facilitate efficient information retrieval and support collaboration among team members, clients and end-users throughout the lifecycle of StuPort.

2. Specific Requirement

2. System Features

The Youth Ventures Student Portfolio Management System, referred to as StuPort, is a web-based management solution specifically crafted for Youth Ventures Asia. It aims to streamline the efficient collection and management of student information, empowering master administrators, administrators, lecturers and students alike. StuPort, a comprehensive platform, focuses on key features such as User Authentication and Profile Management, Dashboard Presentation, Activity Management, Feedback Handling, Reward Allocation, Resume Management, and the newly added Personal Activity. This latest feature empowers users to define and track individual tasks, goals, and activities within the platform, enriching the user experience with a more personalized and goal-oriented approach. Robust User Authentication and Profile Management ensure secure and personalized access, while the Dashboard Presentation feature provides users with a clear overview of their activities. Activity Management is streamlined to facilitate efficient organization and onboarding processes. Feedback Handling and Reward Allocation foster dynamic interaction, creating a well-rounded platform that meets the diverse needs of users. While StuPort won't facilitate financial transactions or replace academic platforms, it seeks to enhance student management, centralize data for easy retrieval, and foster improved collaboration between Youth Ventures and its clients. By addressing the needs of the intended users, including master administrators, administrators, lecturers and students, StuPort strives to meet the objectives of efficient student management, enhanced data management and improved collaboration.

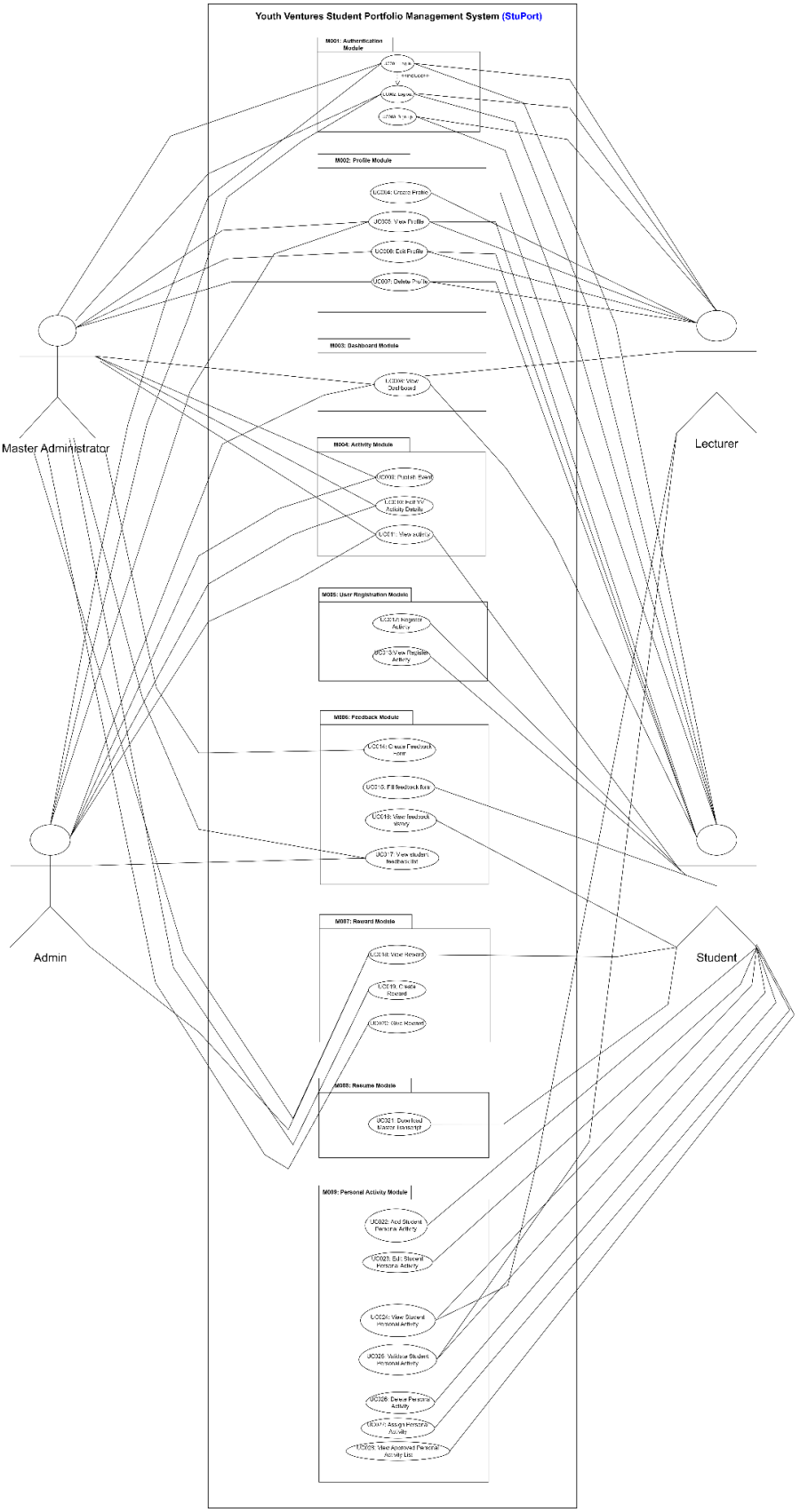


Figure 2.1: Use Case Diagram for Youth Ventures Student Portfolio Management System (StuPort)

Prepared by Group Explorer

Table 2.1: Description of Module and Functions Youth Ventures Student Portfolio Management System (StuPort)

Module	**Function**	**Description**
Authentication Module	UC001: Log In	This use case involves the process of a user logging into the system. The user logs in by entering their credentials and is redirected to the home page upon successful authentication.
	UC002: Log Out	This use case is triggered when the user navigates to the "Log Out" option in the user interface. After clicking on the "Log Out" option, the system logs the user out, redirecting them to the login page.
	UC003: Sign Up	This use case involves the user clicking 'Register,' providing name, role, valid email, a new user ID, and matching passwords. Upon clicking 'Sign Up,' the system verifies and creates a new account, displaying a confirmation message. The user is then directed to the login page.
Profile Module	UC004: Create Profile	This use case begins with the system presenting a form for inputting necessary profile details. The user, whether a lecturer or student, fills in the required information, with lecturers providing specific details relevant to their role such as organizational information and contact details, while students enter personal details like educational background, course and hobbies. After the user submits the filled form, the system validates the entered information for

		completeness and correctness. If any input is invalid, an error message is displayed and the User is required to correct the invalid profile information. Valid information is stored in the database and the user is redirected to the edit profile section.
	UC005: View Profile	This use case involves the User (Student/ Lecturer) navigating to the profile section. The system displays the user's profile information, allowing the User to view the presented details. The Master Administrator has to key in the student's/ lecturer's name in the Search Bar to search their profile. The system then provides a list of student and lecturer names that best match the search keyword. Once the Master Administrator selects a specific student or lecturer from the list, the system displays their respective profile information. The Master Administrator can then view the displayed profile information.
	UC006: Edit Profile	This use case begins with the User (Student/ Lecturer) navigating to the profile section. The system displays the current profile information and the User selects "Edit Profile". After modifying the details, the User clicks "Save". The system validates the changes; if invalid, an error message is displayed and the User is required to correct the invalid profile information. Valid updates are stored in the database and the User is redirected to the profile

		section to view the latest information. To edit the profile of a student or lecturer, the Master Administrator navigates to the database profile table of the user and edits the profile details. After the Master Administrator selects the “Save” button, the system updates and stores the edited user’s profile in the database and displays a successful message.
	UC007: Delete Profile	For Students and Lecturers, the process begins with navigating to the profile section, selecting "Delete Profile" and confirming the deletion. Canceling the process redirects the user to the profile section. After confirmation for deletion, the system then permanently deletes the user’s profile and related data, logs them out and displays a deleted successful message. For Master Administrators, they locate and select the "Delete" button for the profile of a student or lecturer intended for deletion. The confirmation prompt follows and cancels will return the master administrators to the database profile table interface. Upon confirmation, the system permanently deletes the student's or lecturer’s profile and related data, displaying a deleted successful message.
Dashboard Module	UC008: View Dashboard	This use case involves the Master Admin, Admin, or Student navigating to the dashboard section. The system retrieves data from connected sources and compiles relevant

		information. The generated dashboard is then displayed, presenting specific data based on the user's role. For the Student, profile and reward/badges statistics are shown. The Master Admin views student information, admin details, and overall system data. The Admin observes activities statistics, including registration and feedback data.
Activity Module	UC009: Publish Event	This use case involves a system where Master Admin and Admin users can add new activities. Users fill out a form with details such as Activity Title, Description, Category, Date and Time, Location, Organizer Name, Skill Acquired, and Attachment. Submitted information is stored in a database table called 'activity'. Users can manage these activities on a dedicated page, including updating and deleting entries. The system integrates a feedback module for user interaction and allows for easy navigation between adding, updating, and deleting activities.
	UC010: Edit YV Activity Details	The "Edit YV Activity" feature allows Master Admin and Admin users to seamlessly modify activity details from the "Manage Activities" page. By clicking the "Update" button next to the desired activity, users are directed to a form identical to the one used for adding activities. After submitting the edits, the changes are updated in the 'activity' database table. Users are then redirected back to the "Manage Activities" page, where the modified details are reflected in

		the respective activity row.
	UC011:View Activity	The "View YV Activity" feature allows Master Admins, Admins, and Students to access a table displaying YV activities on the "Manage Activities" page. The table includes columns for Activity Title, Description, Category, Date and Time, Location, Organizer Name, Skill Acquired, and Attachment. Master Admins and Admins have an additional "Action" column for managing activities, while Students can only view the information without editing or deleting. The page serves as a comprehensive overview of all activities created by Master Admins and Admins, facilitating easy navigation and understanding of each activity's details. Additionally, students can register for activities, as the page is integrated with a registration module.
Registration Module	UC012: Register Activity	The "Register YV Activity" feature enables students to register for activities created by Master Admins and Admins. On the "View Activities" page, students can click the "Register" button next to each activity, initiating the registration process. After successful registration, the button changes color to dull yellow, indicating registration completion and preventing duplicate registrations.
	UC013: View Register Activity	The "View Registered YV Activity" module offers students a dedicated page to see all activities they've registered for. Similar to the "View Activities" page, it displays Activity Title,

		Description, Category, Date and Time, Location, Organizer Name, Skill Acquired, and Attachment. The key difference is in the "Action" column, where the status of each activity is labeled as 'Registered'. This page allows students to conveniently track their participation history and stay engaged with their chosen activities.
Feedback Module	UC014: Create Feedback From	In this use case, the Master Admin, viewing a list of activities, selects a specific one and clicks "Add Feedback." The system presents a form for arranging feedback questions, and upon completion, the Master Admin clicks "Publish."
	UC015: Fill Feedback Form	In this use case, the student navigates to the "Activity" section, scrolls to the activity area, and views registered and attended activities. Selecting a specific activity, the student clicks on its feedback link, leading to a displayed feedback form. The student completes the form, providing feedback on the activity experience, and concludes the process by submitting the feedback form.
	UC016: View Feedback History	In this use case, the student navigates to the "Activity" section, and views a list of attended activities. Selecting a specific event, the system retrieves and displays feedback details for the chosen activity, presenting them in a list format.
	UC017: View Student Feedback List	In this use case, the Master Admin or Admin navigates to the activity section on the dashboard, where the system presents a list of

		activities with registration and feedback data. The user selects a specific activity and clicks on its feedback data. The system then retrieves and displays detailed feedback for the selected activity through a link, including a list of students and their respective feedback replies.
Reward Module	UC018: View Reward	In this use case, the Admin, Master Admin, or Student navigates to the "View Reward" section, refines rewards using specified criteria, and reviews the displayed reward information.
	UC019: Create Reward	In this use case, the Master Admin navigates to the "Reward" section, inputs details for a new reward in a presented form, and, upon successful validation, the system creates the reward, confirming to the Master Admin

	UC020: Give Reward	Flow of events: The Master Admin navigates to the "Give Reward" section in the system. The system presents a form or interface for the Master Admin to select a user or entity and choose a reward to give. The Master Admin selects the recipient and the specific reward to be given. The system validates the eligibility of the selected recipient for the chosen reward. If eligibility is confirmed, the system processes the reward and updates the recipient's record. The Master Admin receives confirmation of the successful reward giving.
Resume Module	UC021: Download Master Transcript	This use case enables Students to download their Master Transcript by navigating to the profile section, selecting "Download Resume" and confirming the download. The system ensures a streamlined process with confirmation prompts and the option to cancel the download action.
Personal Activity Module	UC022: Add Student Personal Activity	The "Add Personal Activity" use case allows students to add their personal activities to the system. By clicking "Create," users fill out a form with details such as Personal Activity Name, Description, Date, Venue, and Evidence upload. Submitted information is displayed on the "Manage Personal Activities" page in a table format. Users can update or delete activities by clicking the respective buttons in the Action column, which navigates them to update forms or

		confirmation messages accordingly.
	UC023: Edit Student Personal Activity	Students can edit their personal activities by clicking the "Update" button on the "Manage Personal Activities" page. This action takes them to a form similar to the one used for adding personal activities. After submitting changes, the modifications are reflected on the page.
	UC024: Validate Student Personal Activity	Lecturers validate and approve assigned personal activities on the "Manage Personal Activities" page. They can view details of the assigned activities and choose to either "Approve" or "Reject" based on the evidence provided. Clicking "Approve" updates the status to "Approved," while clicking "Reject" rejects the activity.
	UC025: View Student Personal Activity	Both students and lecturers can access personal activities on the "Manage Personal Activities" page. The table presents columns for Personal Activity Name, Description, Date, Venue, Evidence, and Action. Students can add, update, delete, and assign personal activities to lecturers. Lecturers have options to approve or reject assigned activities and can also view their details.
	UC026: Delete Personal Activity	Students can delete personal activities by clicking the "Delete" button on the "Manage Personal Activities" page. A confirmation message appears before deletion to ensure

		intentionality. Confirming the deletion removes the activity, while clicking "Close" preserves the existing activities.
	UC027: Assign Personal Activity	Students can assign their personal activities to a specific lecturer by clicking the "Assign" button in the Action column on the "Manage Personal Activities" page. Upon clicking "Assign," a list of available lecturers is displayed, and students can select the desired lecturer. The 'Lecturer' column in the table is updated with the selected lecturer's name. The assigned lecturer receives the details of the personal activity for validation and approval. The 'Status' column initially shows "Waiting" until the lecturer approves or rejects the assignment.
	UC028: View Approved Personal Activity List	Students can access a dedicated page to view their approved personal activities. The list includes details such as Personal Activity Name, Description, Date, Venue, Evidence, and a 'Status' column indicating "Approved." This page offers students a clear overview of their validated and approved personal activities.

Module 001: Authentication module

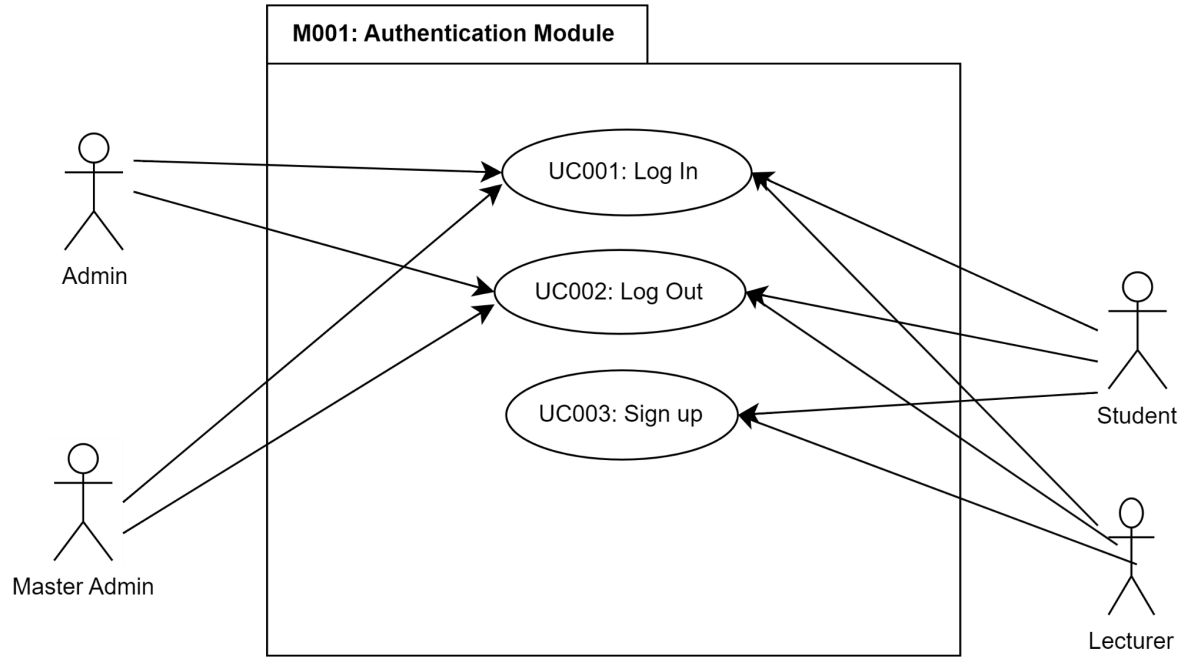


Figure 2.1 (a): Use Case Diagram for Authentication module

The use case includes in this create event module is stated as below:

I. UC001: Log In
II. UC002: Log Out
III. UC003: Sign Up

2.1.1. UC001: Use Case <Log In>

Table 2.1.1: Use Case Description for Log In

Use case: Log In
ID: UC001
Actors: 1. Lecturer 2. Student
Preconditions: ● The user has a signed up account and user ID.

● The user has active internet connection to the system.
Flow of events: 1. The user clicks ‘Login’ 2. The user enters a signed up email address or user ID. 3. The user enters password 4. The user clicks ‘Login’ 5. The user is redirected to home page
Postconditions: ● The user is successfully authenticated and logged in.
Alternative flow 1: 1. The user clicks ‘Sign Up’ 2. The user enters signed up email address 3. System displays ‘Email is already signed up’ 4. The user clicks ‘Login’ besides ‘Already signed up?’ 5. The user enters signed up email address or user ID 6. The user enters password 7. The user clicks ‘Login’ 8. The user is redirected to home page
Postconditions: ● The user is successfully authenticated and logged in.
Exception flow: 1. Invalid email address/userID 1.1 Users enter unsigned email address/user ID 1.2 Users enter the wrong format email address/wrong user ID 1.2 Displays 'Email is not signed-up' message 2. Incorrect password 3.1 User enter invalid password 3.2 System displays ‘Incorrect password’ 3.3 User click ‘Forgot Password’

	3.4 User received verification email in their signed-up email address. 3.5 User fill-in new password 3.6 User fill-in confirm new password 3.7 User reset password

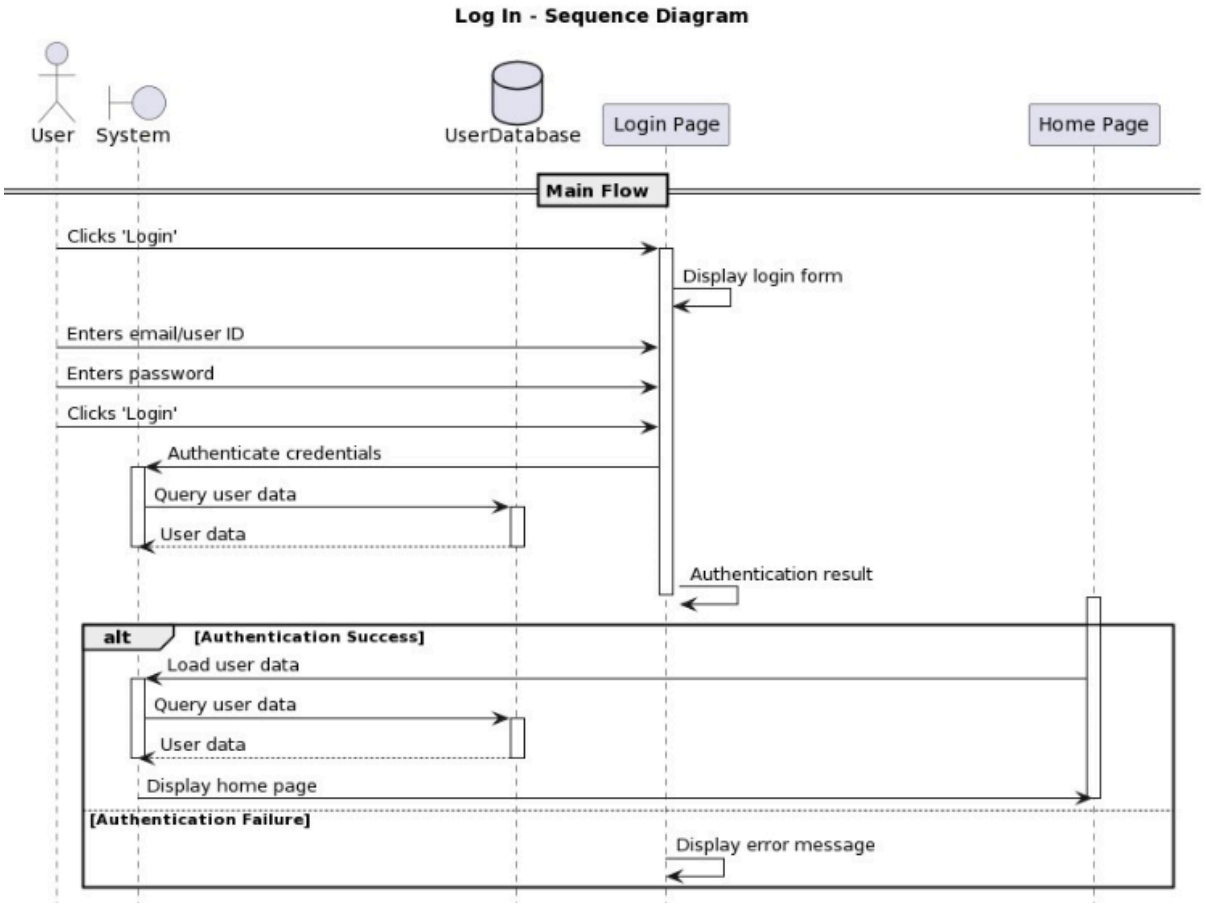


Figure 2.1.1.1.1: Sequence Diagram of Log In

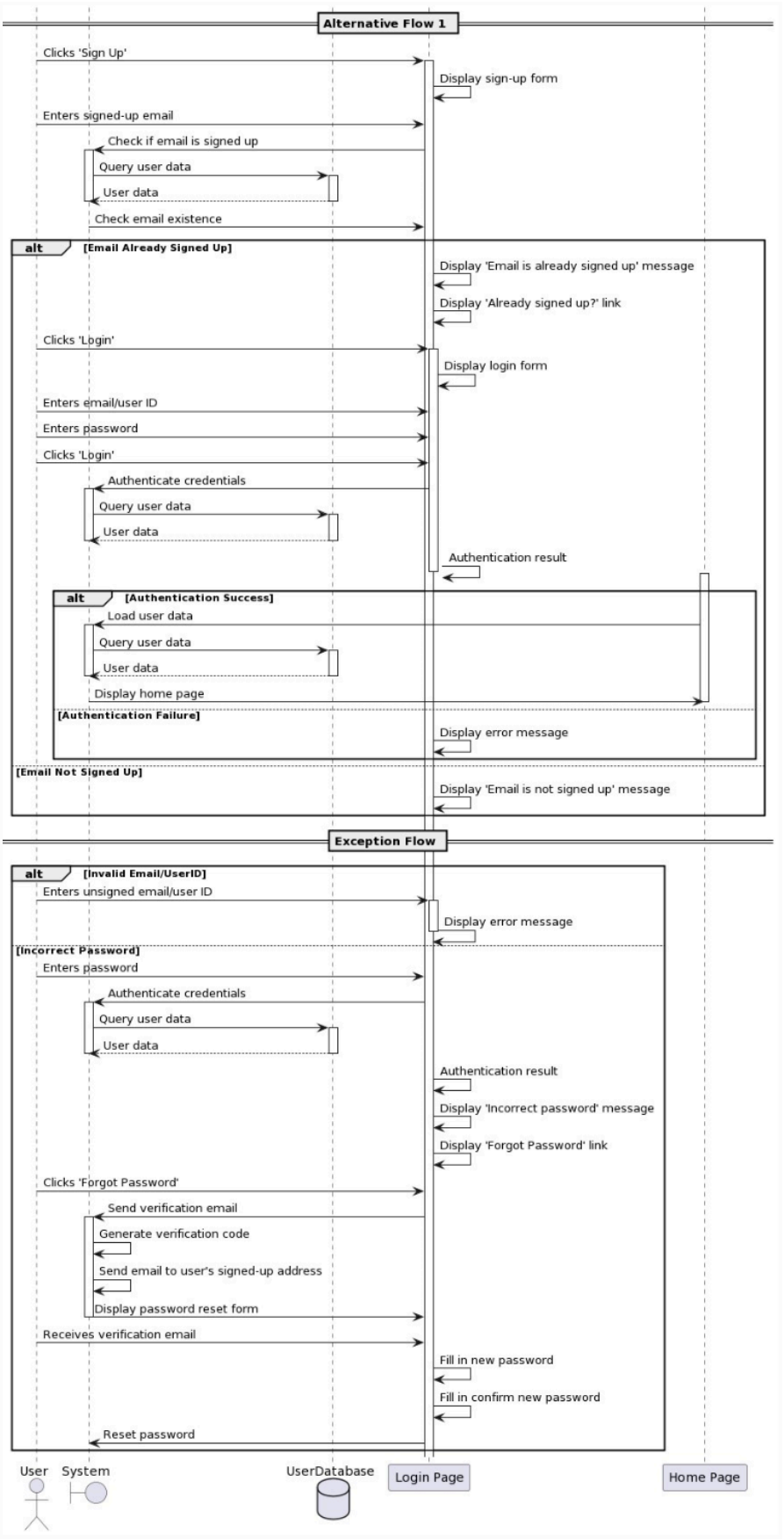


Figure 2.1.1.1.2: Sequence Diagram of Log In

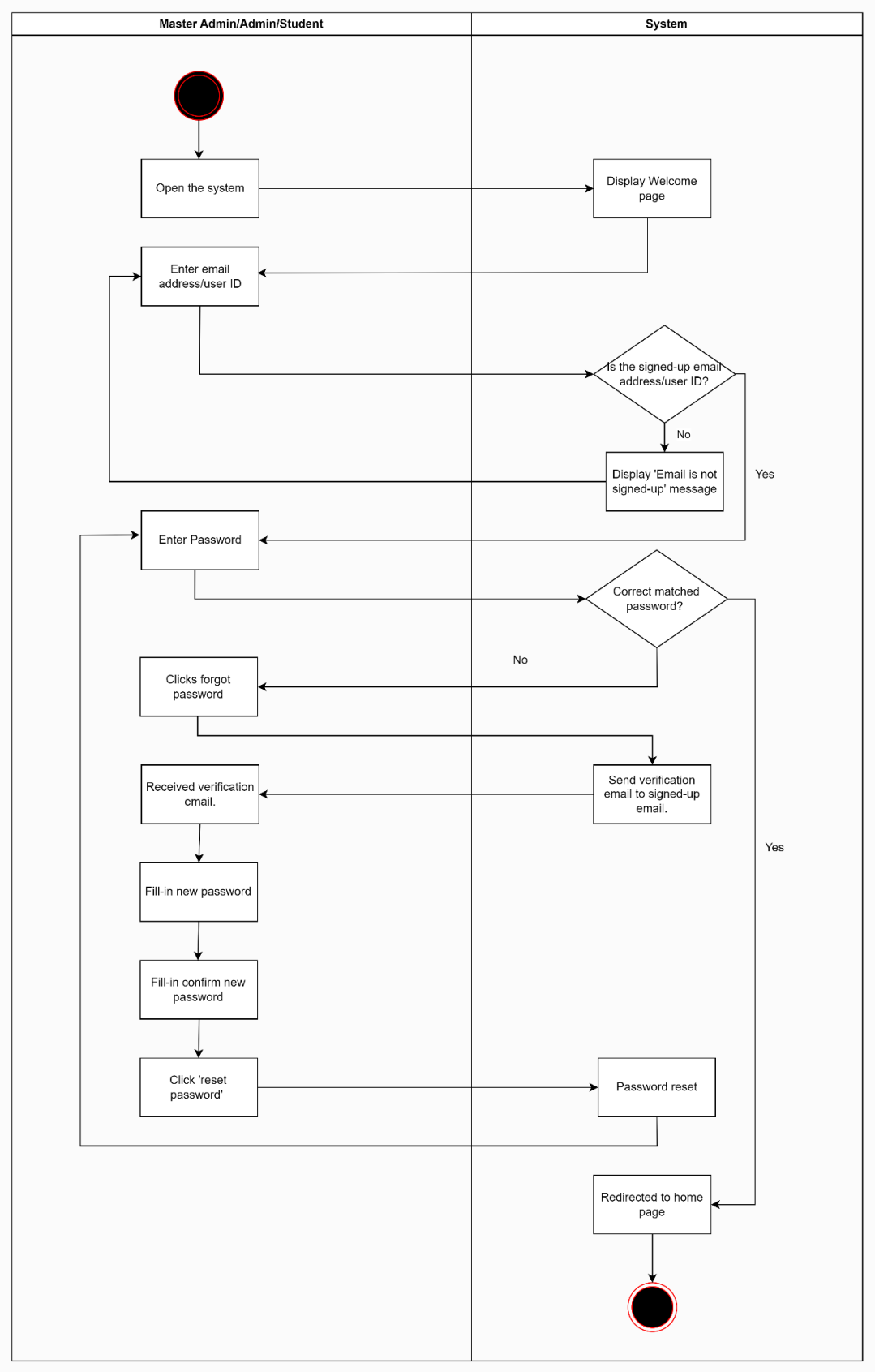


Figure 2.1.1.2: Activity Diagram of Log In

2.1.2. UC002: Use Case <Log Out>

Table 2.1.2: Use Case Description for Log Out

Use case: Log Out
ID: UC002
Actors: 1. Admin 2. Master Admin 3. Student 4. Lecturer
Preconditions: ● The user is currently logged into the system.
Flow of events: 1. The user navigates to the "Log Out" option in the user interface. 2. The user clicks on the "Log Out" option. 3. The system logs the user out. 4. The user is redirected to the login page.
Postconditions: 1. The user is successfully logged out of the system.

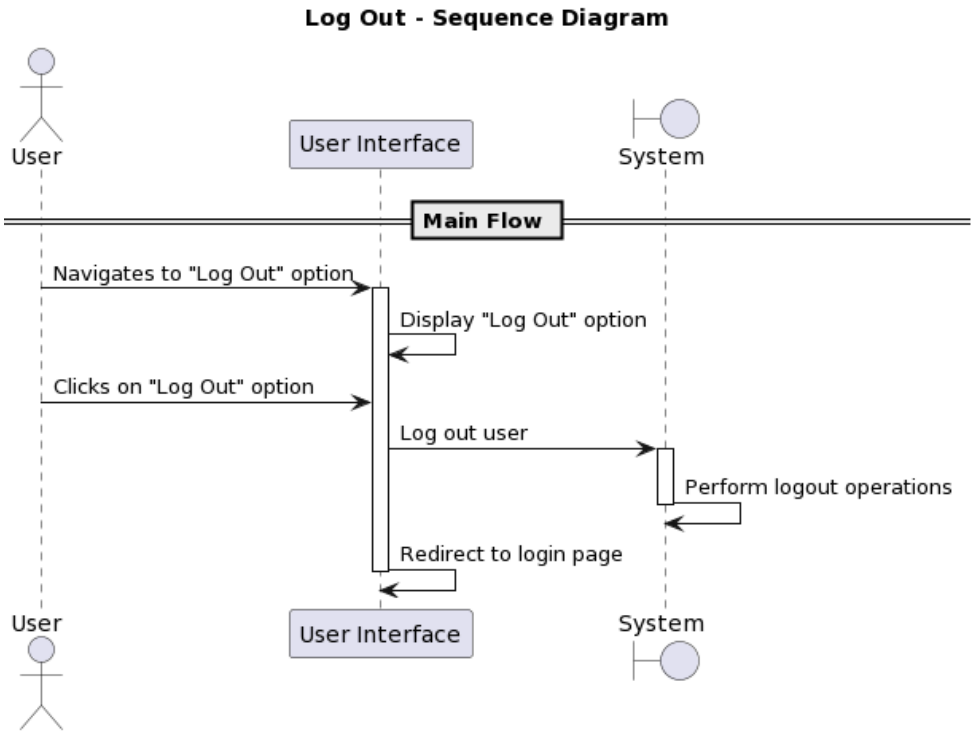


Figure 2.1.2.1: Sequence Diagram of Log Out

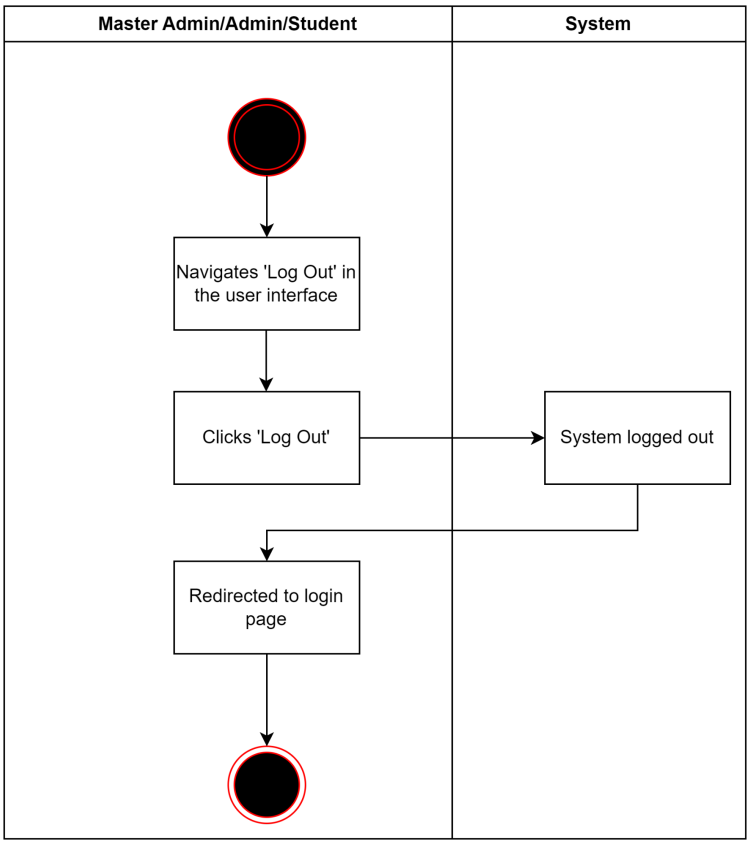


Figure 2.1.2.2: activity Diagram of Log Out

2.1.3. UC003: Use Case <Sign Up>

Table 2.1.3: Use Case Description for Sign Up

Use case: Sign Up
ID: UC003
Actors: 1. Lecturer 2. Student
Preconditions: ● Users must have active internet connection to the system. ● Users must have their own valid email address for sign up.
Flow of events: 1. The user clicks 'Register', then will be redirected to the Sign up Page. 2. Users will need to fill their name, role, and valid email address to register a new account. 3. User will enter a new user ID. 4. Users will need to enter the password twice and both passwords entered must be the same. 5. The user clicks 'Sign Up' to submit the registration form. 6. The system verifies the information provided, checking for any errors or missing fields. 7. If all the information is valid, the system creates a new user account. 8. System displays 'Thank you for signing up!" 9. The user is directed to the login page.
Postconditions: ● The user successfully signed up and his information is registered in the system. ● The user receives a verification email for their successful sign-ups.

- Verification is required before the user logs in.

Alternative flow *1*:

1. The user clicks ‘Login’
2. The user clicks ‘Sign Up’ besides ‘Don’t have an account yet?’
3. The user enters a valid email address.
4. The user enters new user ID.
5. The user enters password.
6. The user enters password for second time.
7. The user clicks ‘Sign Up’.
8. System displays ‘Thank you for signing up!”
9. The user is directed to the login page..

Postconditions:

- The user successfully signed up and his information is registered in the system.
- The user receives a verification email about their successful sign-ups.
- Verification is required before the user logs in.

Exception flow (if any):

1. Invalid email address
   1.1 Users enter non-existent email address.
   1.2 Error message will be displayed.
2. Email is signed up before
   2.1 Users enter their valid email address but it is registered.
   2.2 Display "Email Registered".
   2.3 Users will be directed to login page.
3. User ID signed up before
   3.1 Users enter their new user ID but it is signed up..
   3.2 Display "User ID Registered".
   3.3 Users will be directed to login page.

4. Incorrect password format

    4.1 Users enter password which does not meet the specified criteria.

    4.2 “Password does not matched the format” displayed.

5. Both passwords do not match

    5.1 Users enter the second password which is not same as the first one.

    5.2 “Password does not matched” displayed.

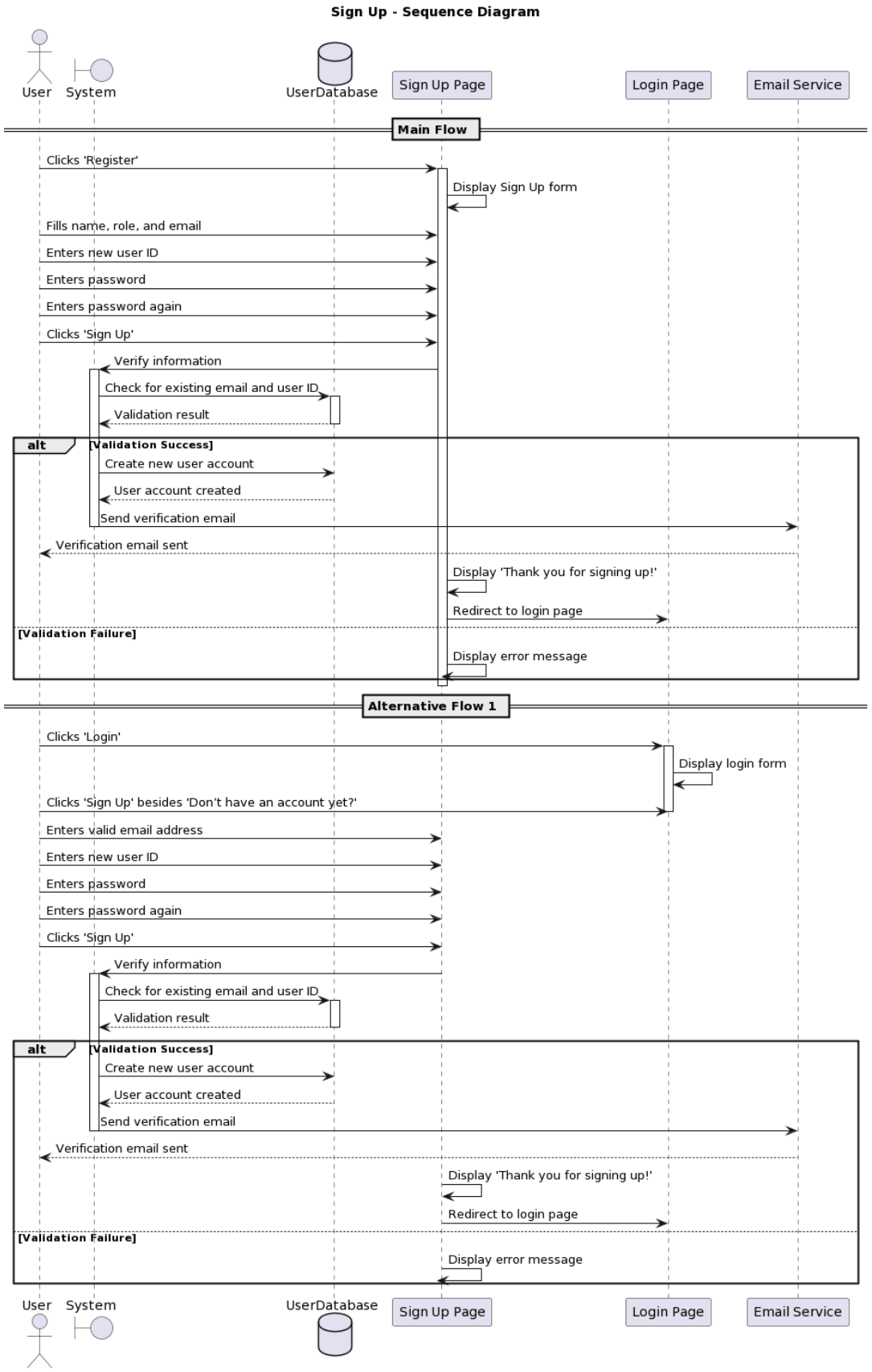
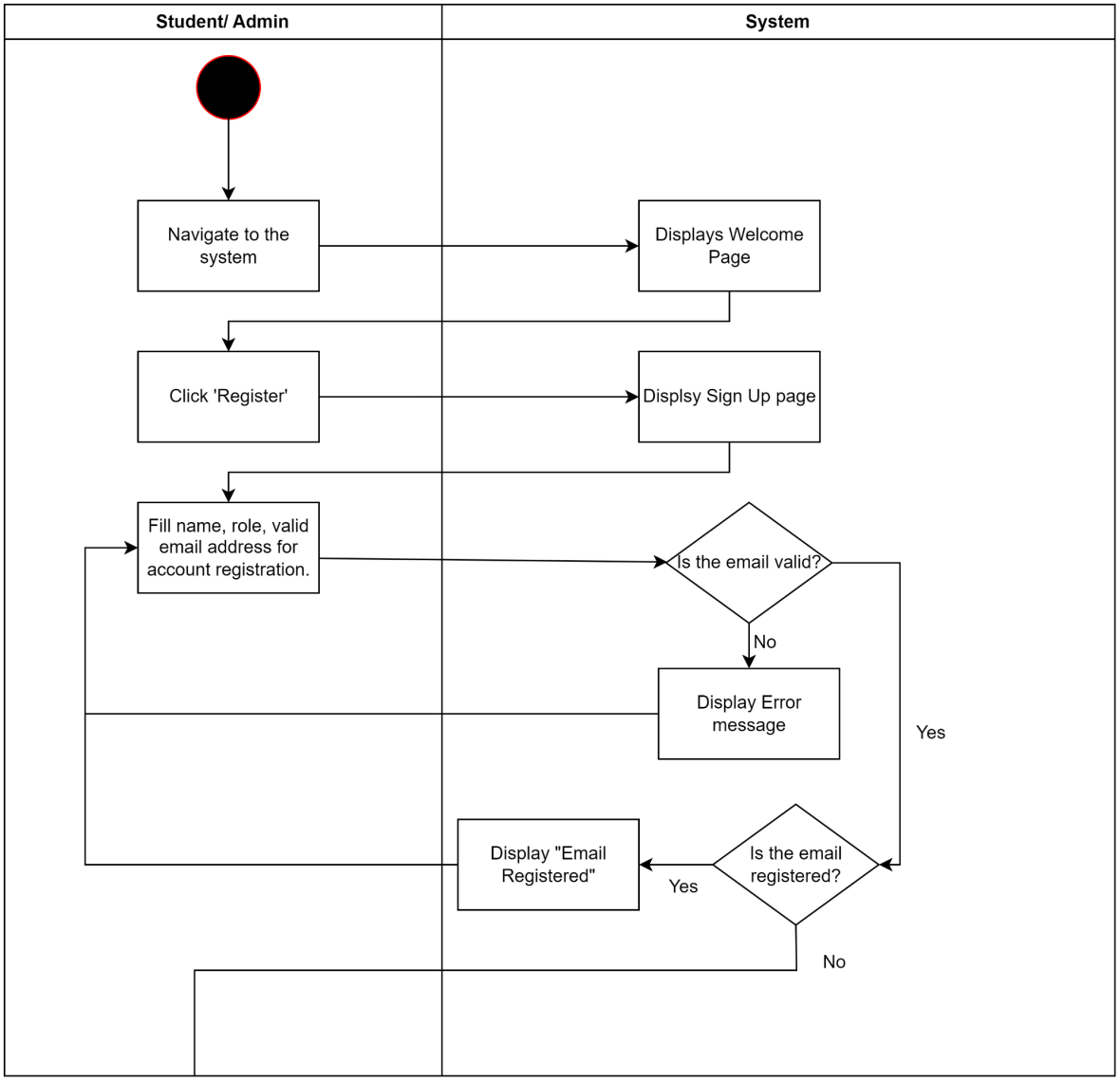


Figure 2.1.3.1: Sequence Diagram of Sign Up


Student/ Admin
System
Navigate to the system
Displays Welcome Page
Click 'Register'
Displsy Sign Up page
Fill name, role, valid email address for account registration.
Is the email valid?
No
Yes
Display Error message
Display "Email Registered"
Is the email registered?
Yes
No

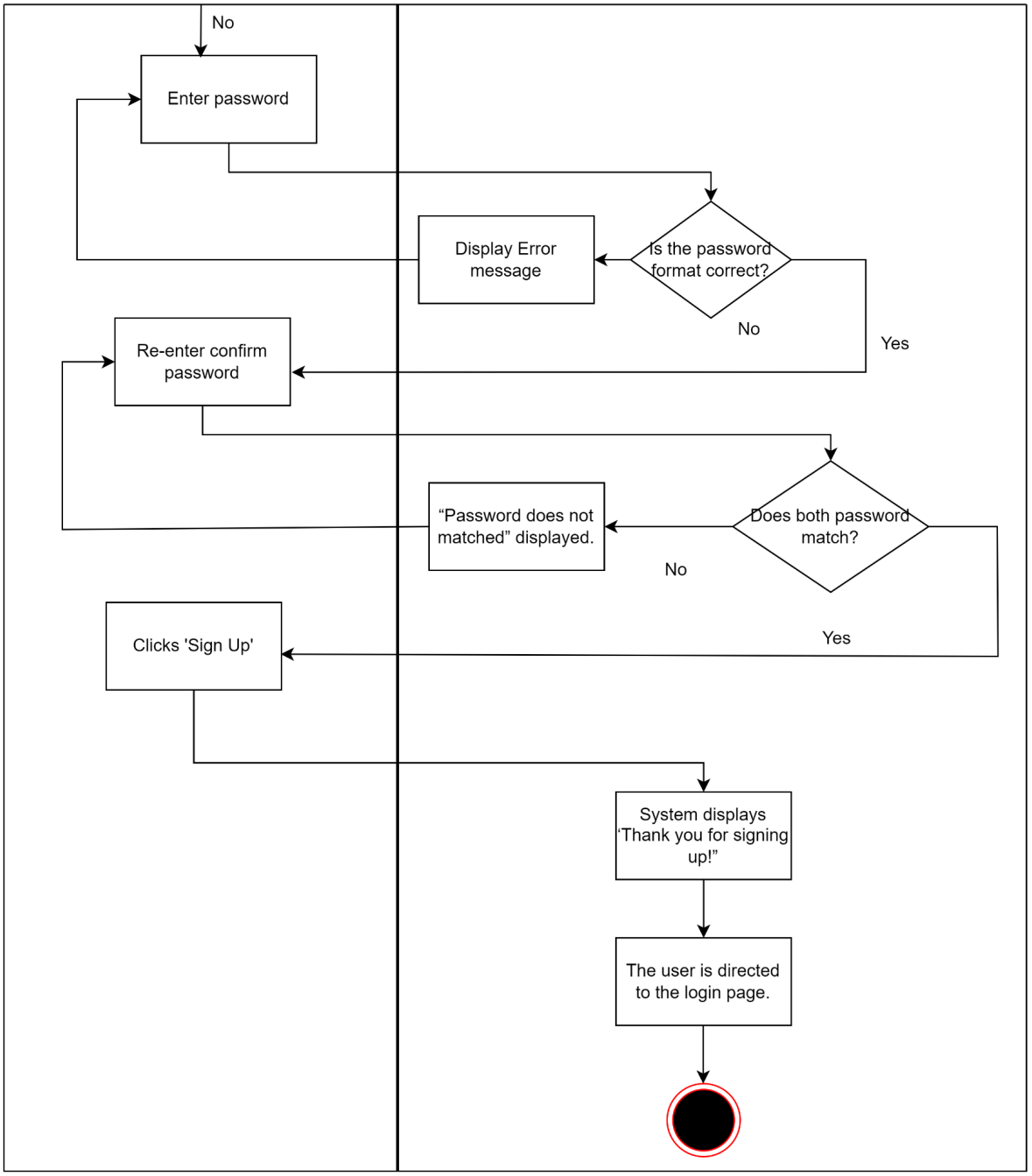


Figure 2.1.3.2: Activity Diagram of Sign Up

Module 002: Profile module

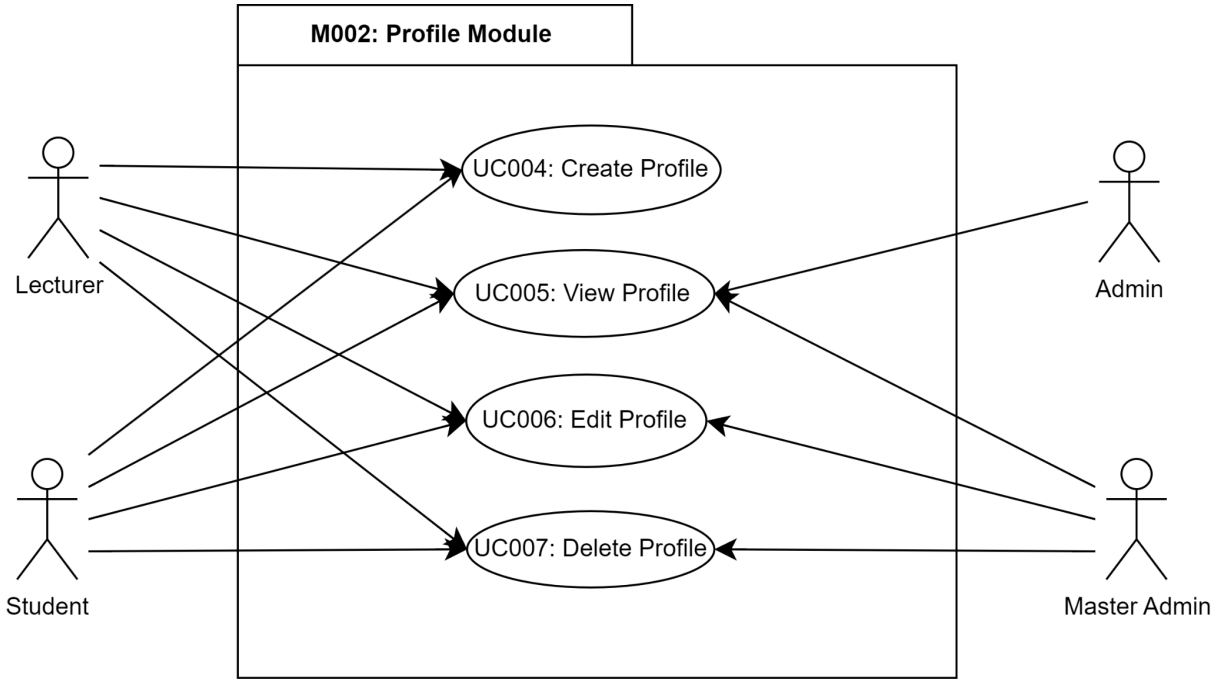


Figure 2.1 (b): Use Case Diagram for Profile module

The use case includes in this create event module is stated as below:

I. UC004: Create Profile
II. UC005: View Profile
III. UC006: Edit Profile
IV. UC007: Delete Profile

2.1.4. UC004: Use Case <Create Profile>

Table 2.1.4: Use Case Description for Create Profile

Use case: Create Profile
ID: UC004
Actors: 1. Student 2. Lecturer
Preconditions:

- The User is logged into the system.
- The User has an existing account in the system.
- The User is new to the system.

Flow of events:

1. The system presents a form to input necessary profile details such as personal information, contact information, educational background and other required data.
2. The User fills in the required information.
3. If the user is an Lecturer:
    a. The Lecturer fills in the specific details relevant to their role such as organizational information and contact details.
4. If the user is a Student:
    a. The Student fills in personal details such as educational background, course and hobby.
5. The User submits the filled form for profile creation.
6. The system validates the entered information for completeness and correctness.
    a. If the input is invalid, Exception Flow 1 is followed.
7. The system stores the user's profile in the database.
8. The User is redirected to the edit profile section.

Postconditions:

- The user's profile details are successfully stored in the system's database.

Exception flow:

1. Invalid Input
    a. System displays an error message to prompt the user to fill in the missing or corrected information.
    b. System continues with Flow of Events 5.

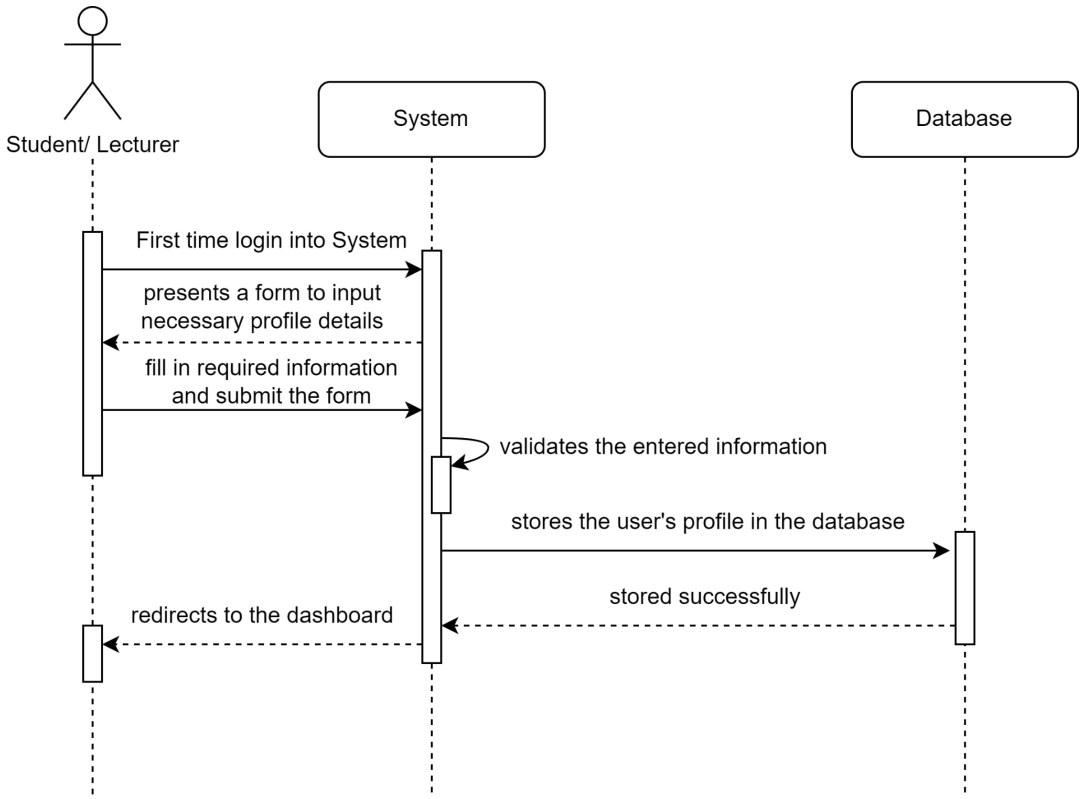


Figure 2.1.4.1: Sequence Diagram of Create Profile

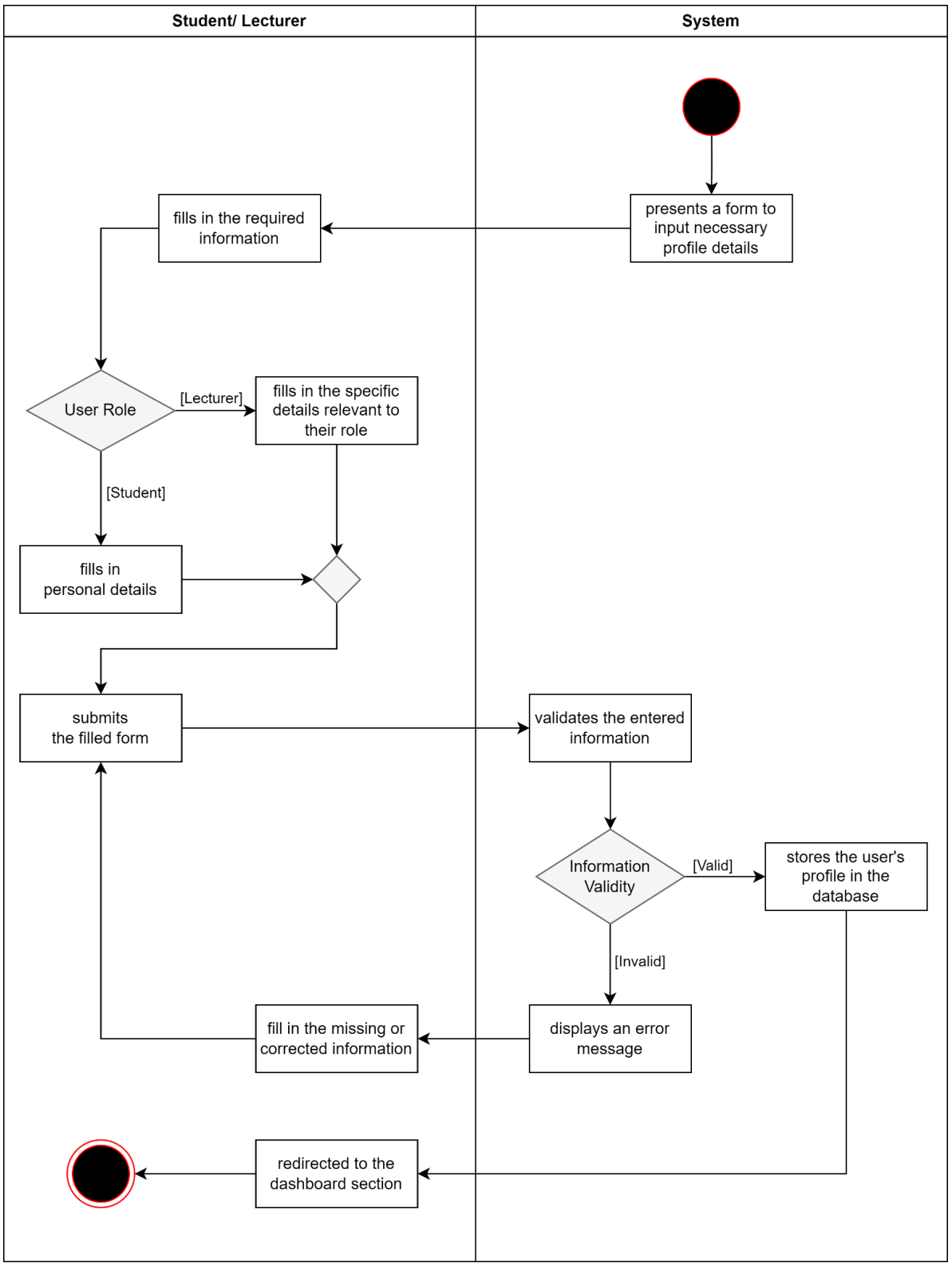


Figure 2.1.4.2: Activity Diagram of Create Profile

2.1.5. UC005: Use Case <View Profile>

Table 2.1.5: Use Case Description for View Profile

Use case: View Profile
ID: UC005
Actors: 1. Student 2. Lecturer 3. Master Administrator
Preconditions: ● The User is logged into the system. ● The User has an existing account in the system.
Flow of events: 1. The User (Student/ Lecturer) navigates to the profile section of the system. 2. The system displays the profile information of the user. 3. The User (Student/ Lecturer) views the displayed profile information.
Postconditions: ● The User has successfully viewed the profile information.
Alternative flow 1: View student's/ lecturer's profile. 1. The User keys in the student's/ lecturer's name in the Search Bar to search the student's/ lecturer's profile. a. If the requested profile does not exist, Exception Flow 1 is followed. 2. The system displays a list of student and lecturer names that most matches with the searching keywords. 3. The User selects the student's/ lecturer's name which they want to view on their profile. 4. The system displays the student's/ lecturer's profile information based on the user's request.

5. The User views the displayed profile information.
Postconditions: ● The User has successfully viewed the requested profile information displayed by the system.
Exception flow: 1. Requested profile does not exist a. System notifies the user that the profile information data could not be found. b. System continues with Alternative Flow 1 Step 1.

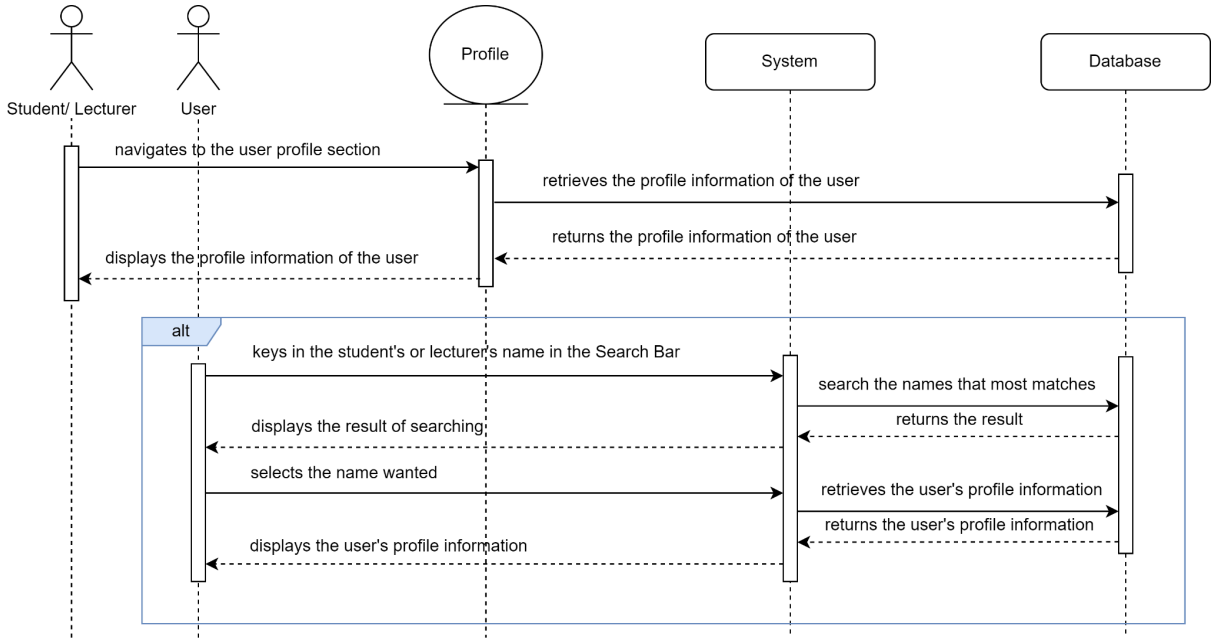


Figure 2.1.5.1: Sequence Diagram of View Profile

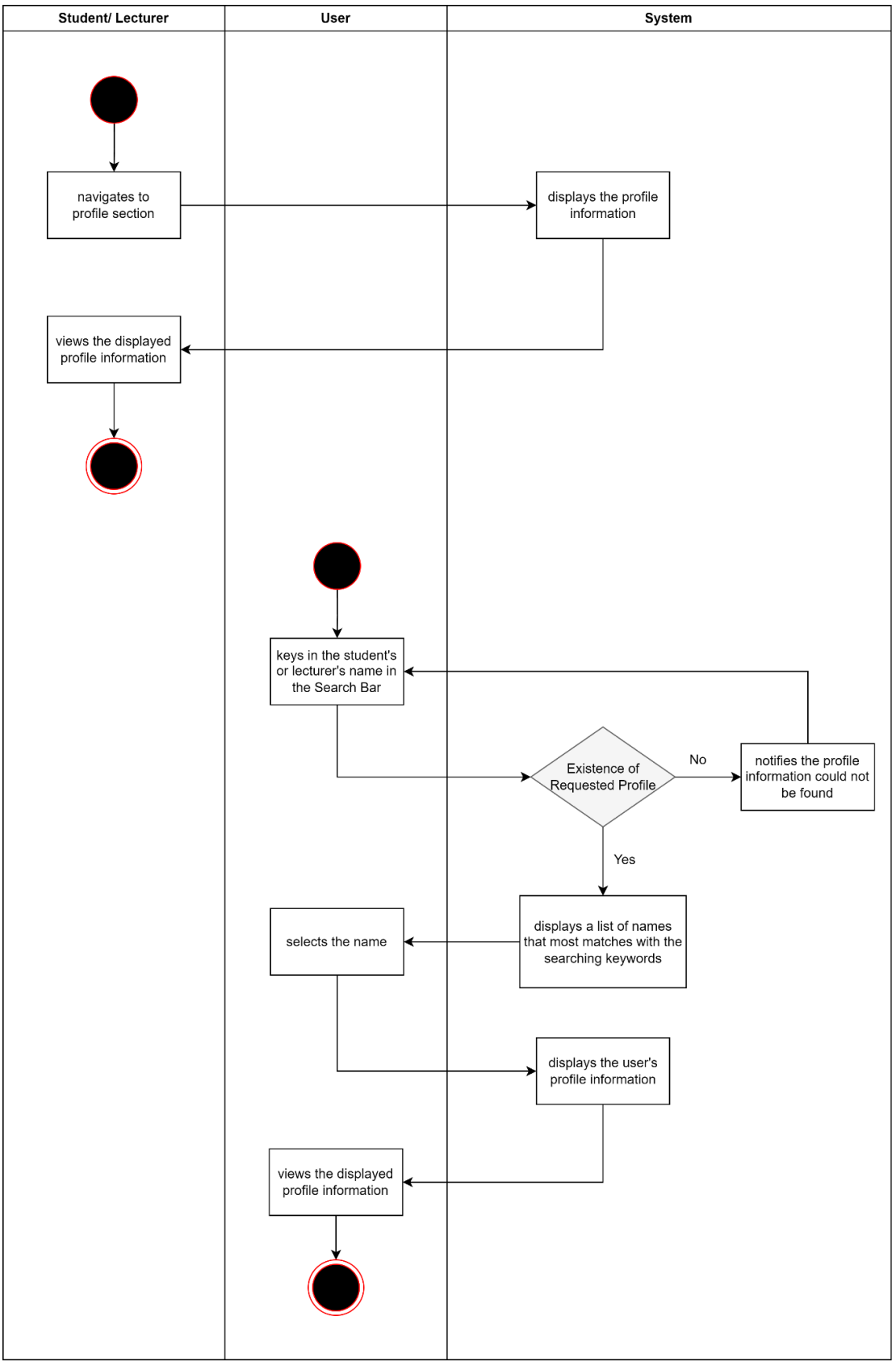


Figure 2.1.5.2: Activity Diagram of View Profile

2.1.6. UC006: Use Case <Edit Profile>

Table 2.1.6: Use Case Description for Edit Profile

Use case: Edit Profile
ID: UC006
Actors: 1. Student 2. Lecturer 3. Master Administrator
Preconditions: ● The User is logged into the system. ● The User has an existing account in the system.
Flow of events: 1. The User (Student/ Lecturer) navigates to the profile section of the system. 2. The system displays the current profile information of the user. 3. The User selects the "Edit Profile" button on the Profile interface. 4. The User edits and modifies the profile details as desired. 5. The User selects the "Save" button to save the changes and edits made to the profile information. 6. The system validates the updated information. a. If the input is invalid, Exception Flow 1 is followed. 7. The system updates and stores the edited user's profile in the database. 8. The User is redirected to the profile section. 9. The system displays the latest profile information of the user.
Postconditions: ● The User's profile is updated with the edited information. ● The system stores the updated profile data.

Alternative flow 1: The user is Master Administrator

1. The User (Master Administrator) navigates to the database profile table of a user (Student/ Lecturer).
2. The system displays the current profile information of the user (Student/ Lecturer).
3. The User selects the “Edit” button in the database.
4. The User edits and modifies the profile details as desired.
5. The User selects the “Save” button to save the changes and edits made to the profile information.
6. The system validates the updated information.
    a. If the input is invalid, Exception Flow 2 is followed.
7. The system updates and stores the edited user's profile in the database.
8. The system displays the latest profile information of the user (Student/ Lecturer).

Postconditions:

- The User's (Student/ Lecturer) profile is updated with the edited information.
- The system stores the updated profile data.

Exception flow 1:

1. Invalid Input
    a. System displays an error message to prompt the user to correct the invalid profile information.
    b. System continues with Flow of Events 5.

Exception flow 2:

1. Invalid Input
    c. System displays an error message to prompt the user to correct the invalid profile information.
    d. System continues with Alternative Flow 1 Step 5.

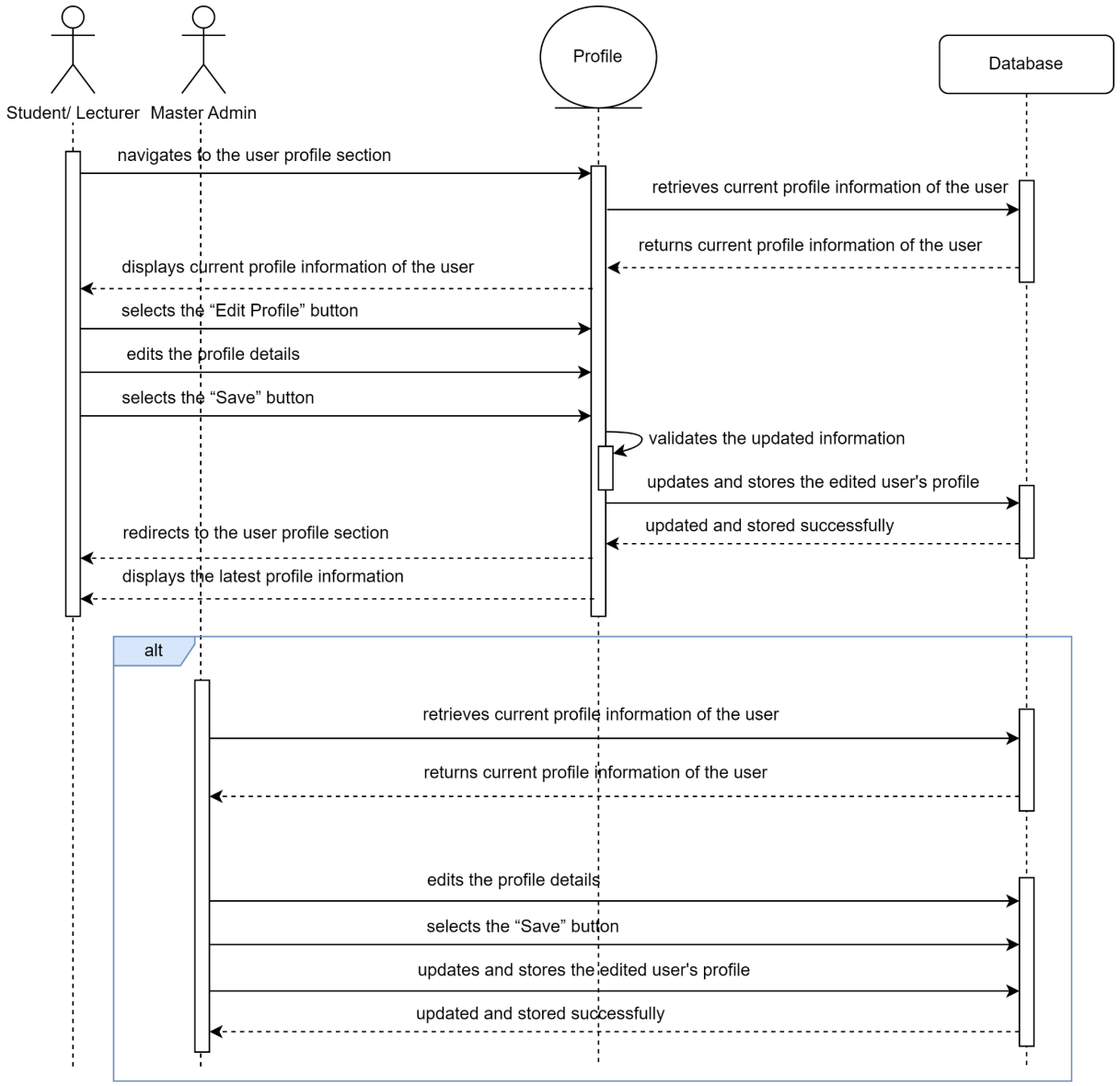


Figure 2.1.6.1: Sequence Diagram of Edit Profile

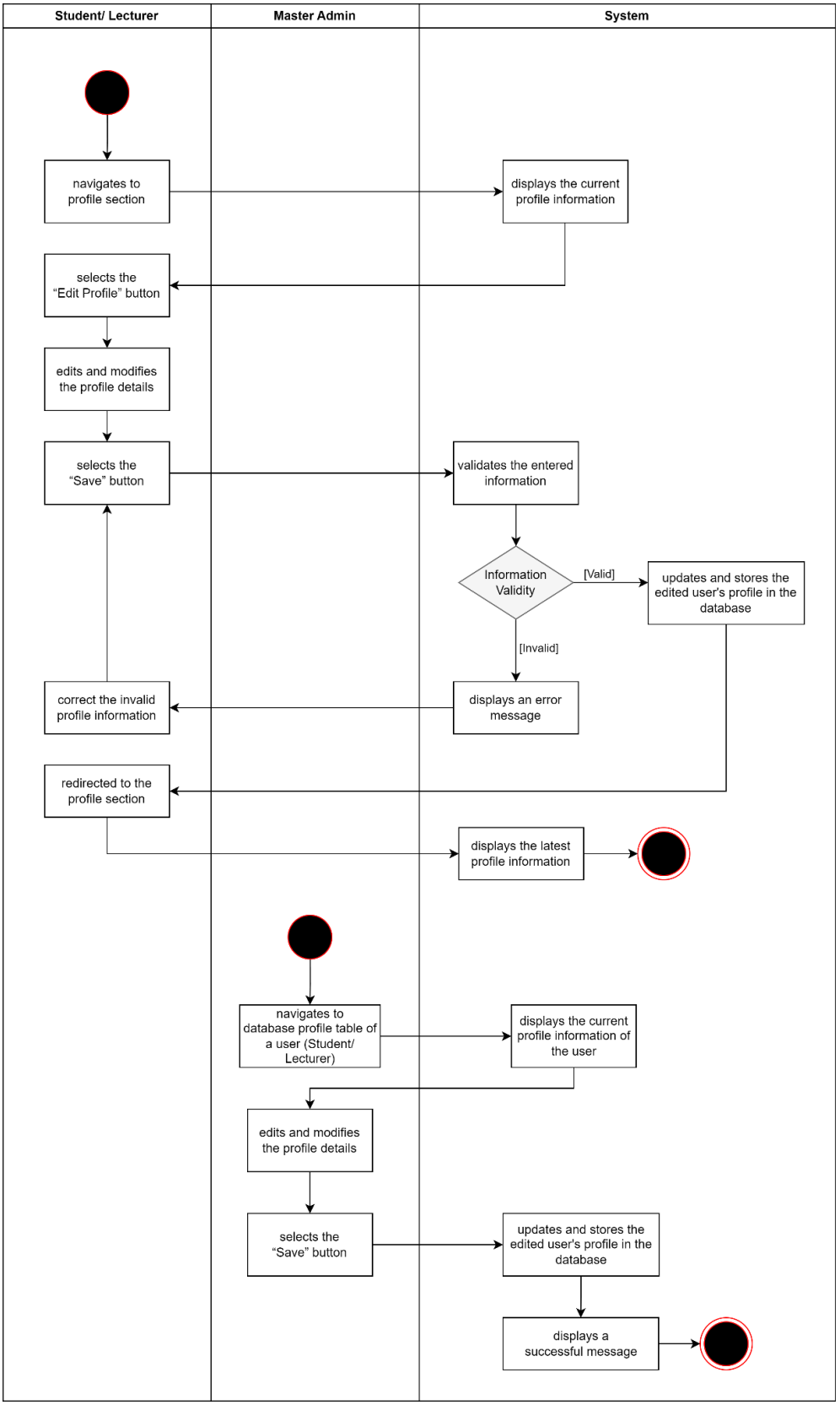


Figure 2.1.6.2: Activity Diagram of Edit Profile

2.1.7. UC007: Use Case <Delete Profile>

Table 2.1.7: Use Case Description for Delete Profile

Use case: <Delete Profile>
ID: UC007
Actors: 1. Student 2. Lecturer 3. Master Administrator
Preconditions: ● The User is logged into the system. ● The User has an existing account in the system.
Flow of events: 1. If the user is Student/ Lecturer a. The User navigates to the profile section. b. The User selects the “Delete Profile” button on the Profile interface. c. The system prompts the user to confirm the deletion. i. If the user cancels the deletion process, the system backs to the user profile section interface. d. The system permanently deletes the user’s profile and all related data from the database of the system. e. The system logs out the user and displays a message indicating the profile was deleted successfully. 2. If the user is Master Administrator a. The User locates the profile of a student or lecturer intended for deletion in the database. b. The User selects the “Delete” button to delete the student's or lecturer’s profile. c. The system prompts the user to confirm the deletion.

i. If the user cancels the deletion process, the system backs to the database profile table interface.

d. The system permanently deletes the student's or lecturer's profile and all related data from the database of the system.

e. The system displays a message indicating the profile was deleted successfully.

Postconditions:

- The chosen-for-deletion profile and all related data is permanently deleted from the database of the system.

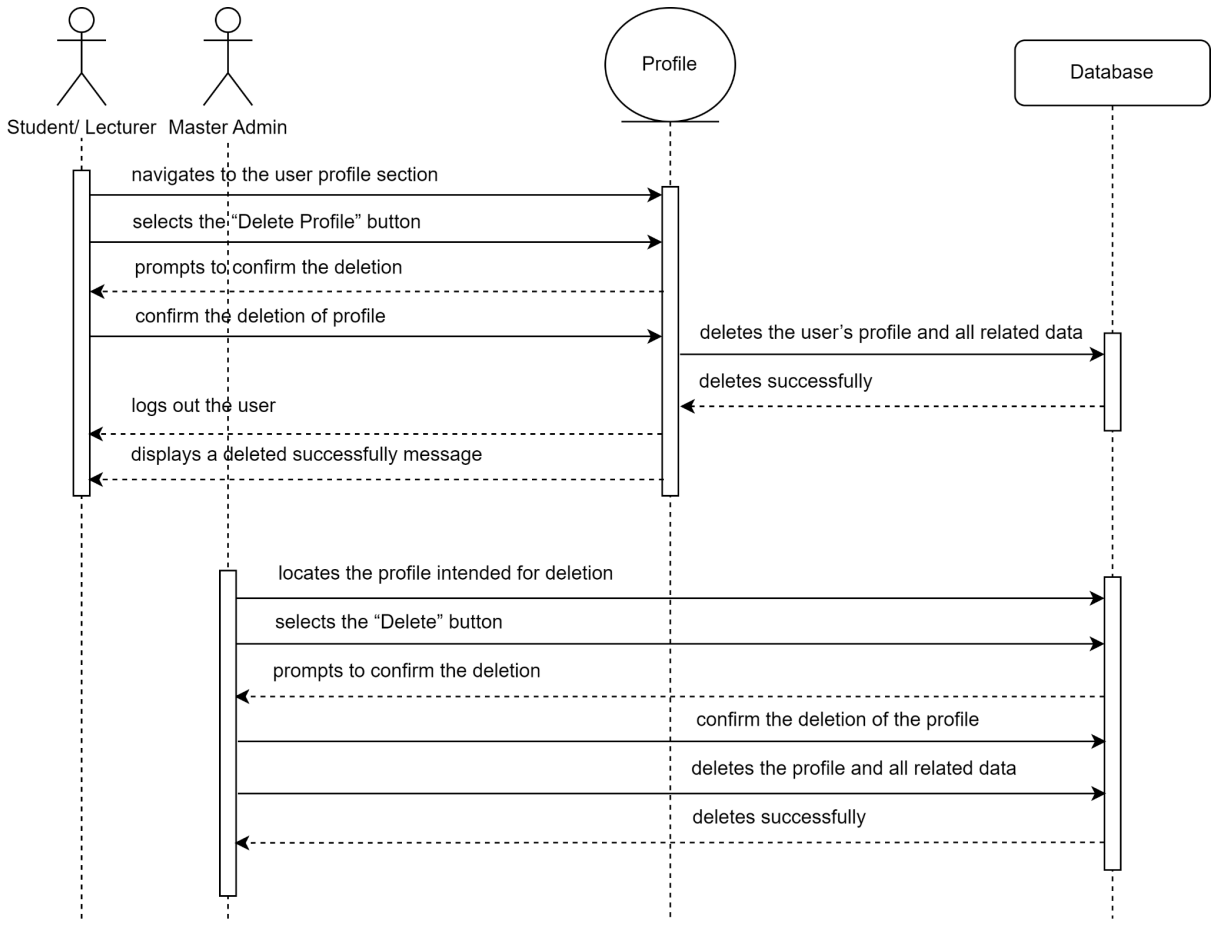


Figure 2.1.7.1: Sequence Diagram of Delete Profile

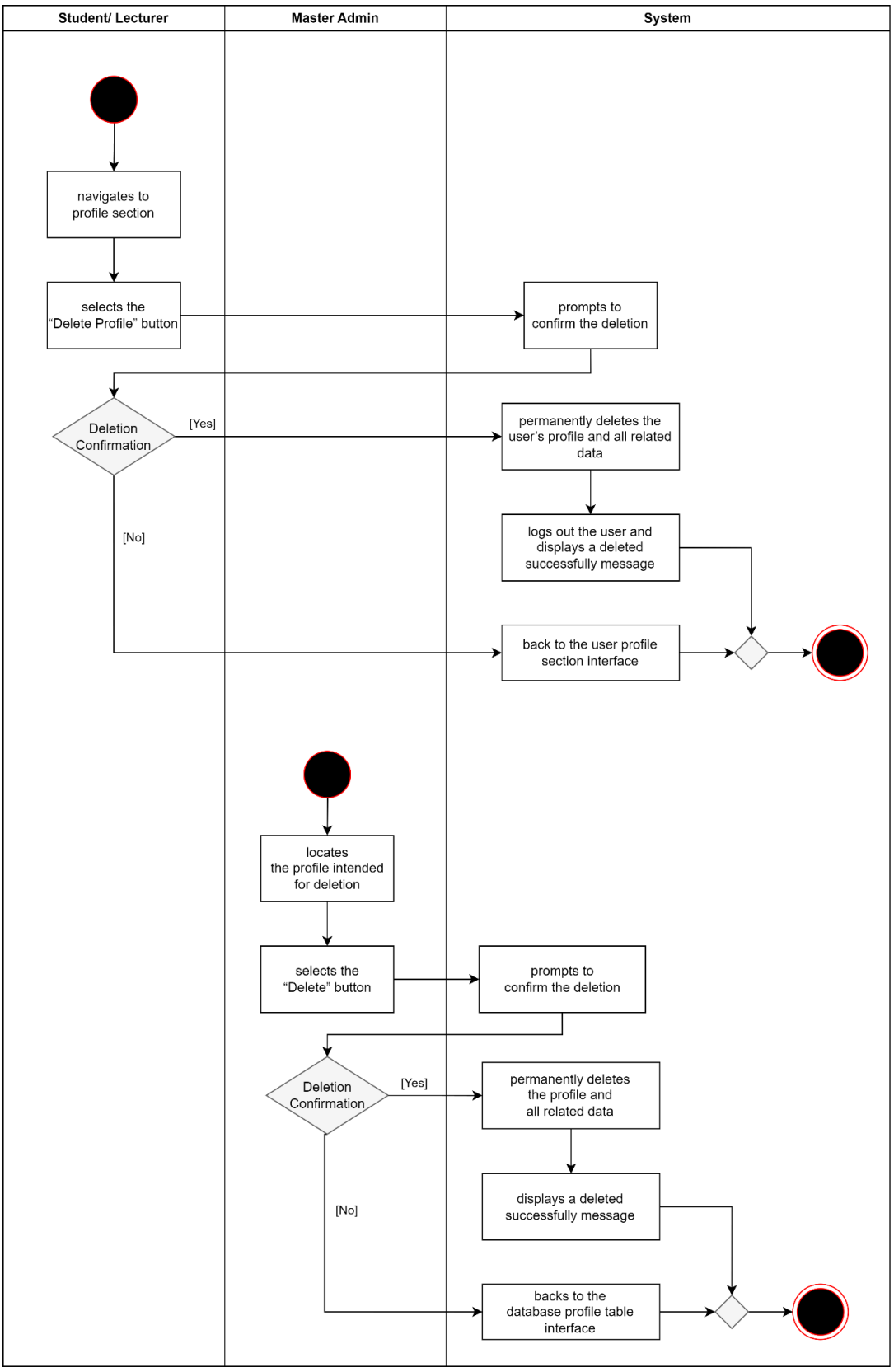


Figure 2.1.7.2: Activity Diagram of Delete Profile

Module 003: Dashboard module

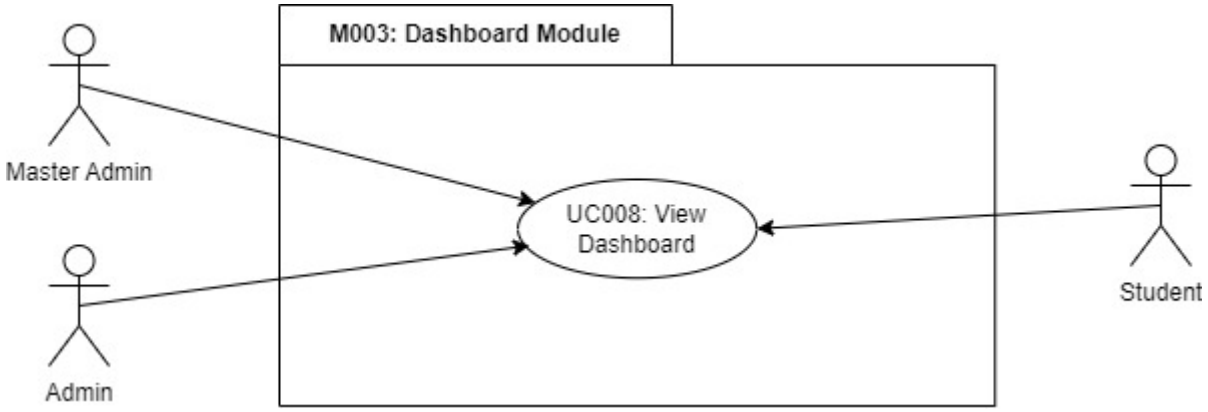


Figure 2.1 (c): Use Case Diagram for Dashboard module

The use case includes in this create event module is stated as below:

I. UC008: View Dashboard

2.1.8. UC008: Use Case <View Dashboard>

Table 2.1.8: Use Case Description for View Dashboard

Use case: View Dashboard
ID: UC008
Actors: 1. Master Admin 2. Admin 3. Student 4. Lecturer
Preconditions: ● The Master Admin, Admin, Student or Lecturer is logged into the "StuPort" system.

- The system must be operational and have the necessary data sources connected to generate the dashboard.

Flow of events:

1. The Master Admin, Admin, Student or Lecturer navigates to the dashboard section of the application.
2. The system retrieves and compiles relevant data from connected sources.
3. The dashboard is generated and displayed, presenting the relevant data.

    3.1 On the student dashboard, profile and activity data statistics will be displayed.

    3.2 On the Master Admin dashboard, student information database, admin information database and system database including activities statistics data will be displayed.

    3.3 On the Admin dashboard, activities statistical data including registration and feedback data will be displayed.

Postconditions:

- The Master Admin, Admin, Student or Lecturer can successfully view and interact with the dashboard.

Exception flow 1: Connectivity Issue on Data.

1. In Step 2 and 3, if there are connectivity issues or data source problems, the system may display an error message or a partial dashboard with available data.
2. The Master Admin, Admin, Student or Lecturer is informed of the problem and asked to refresh the website again.

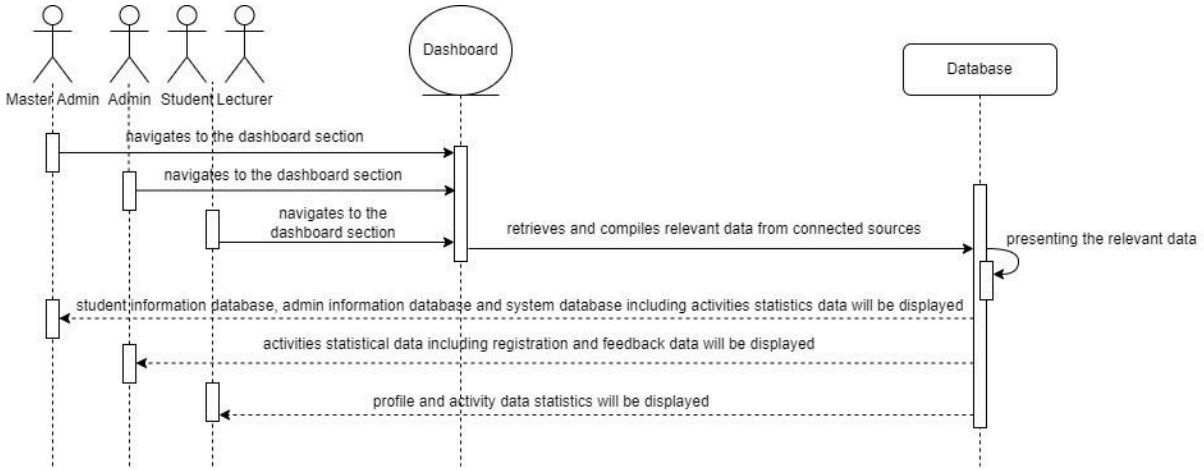


Figure 2.1.8.1: Sequence Diagram of View Dashboard

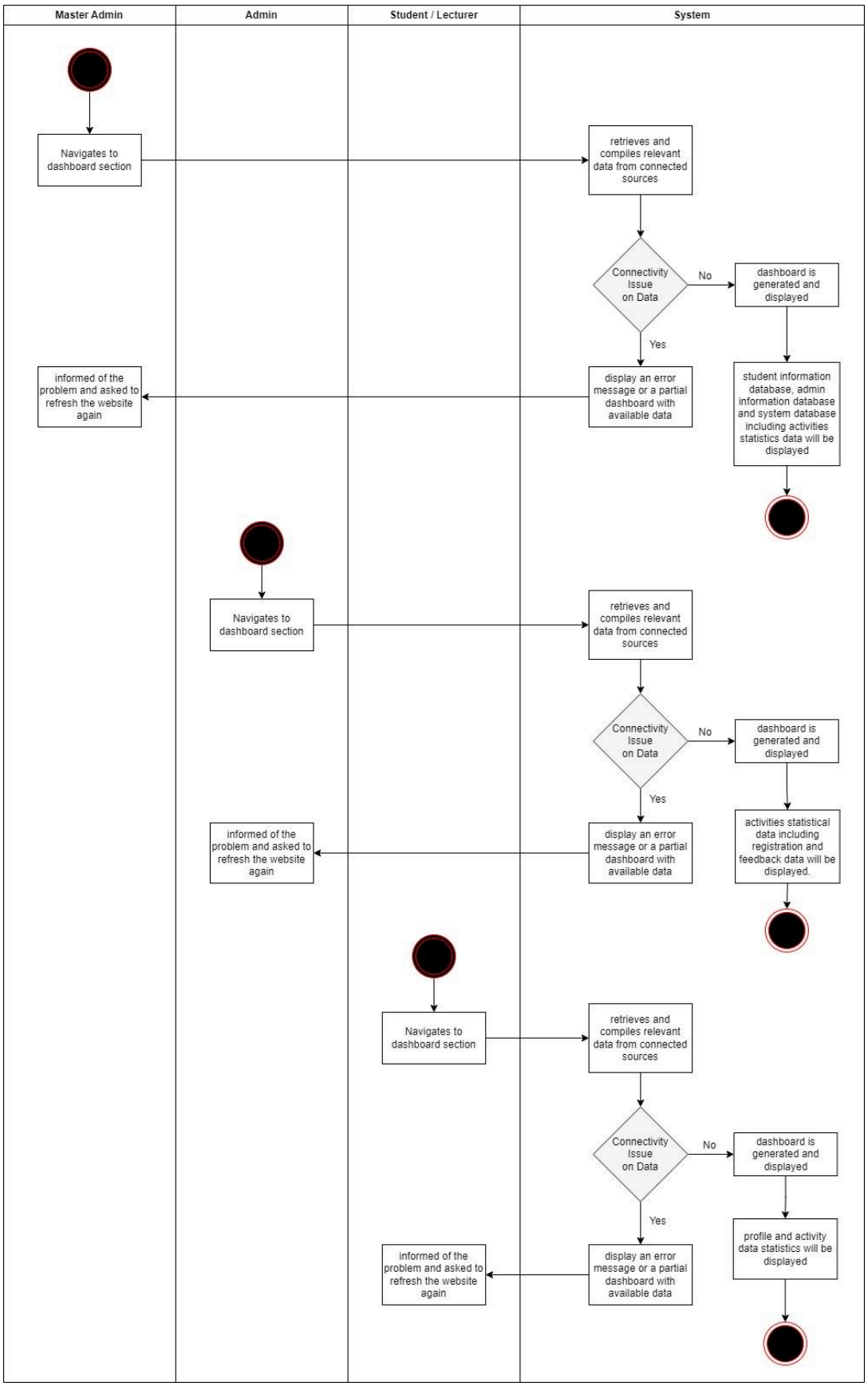


Figure 2.1.8.2: Activity Diagram of View Dashboard

Module 004: Activity module

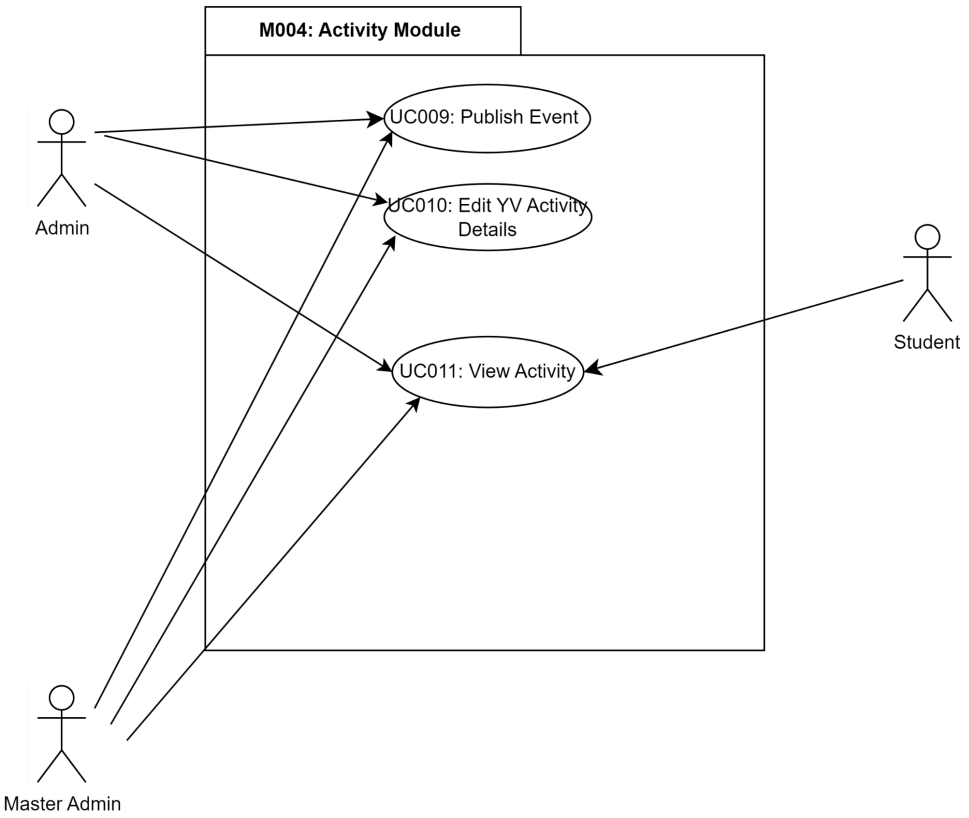


Figure 2.1 (d): Use Case Diagram for Activity module

The use case includes in this create event module is stated as below:

I. UC009: Publish Event
II. UC010: Edit YV Activity Details
III. UC011:View Activity

2.1.9. UC009: Use Case <Publish Event>

Table 2.1.9: Use Case Description for Publish Event

Use case: Publish Event
ID: UC009
Actors: 1. Admin

2. Master Admin
Preconditions: ● Actors are logged into the system. ● Actors have the necessary permissions to publish events.
Flow of events: 1. Actors select the activity category from the 6 types of activity. 2. Actors provides information (Date, Event Title, Organizers/sponsors, Location. 3. Choose a type of category. 4. Click 'Submit'. 5. Activity is published.
Postconditions: ● The event is published and visible in the system. ● Registration information, if applicable, is set up according to the Actor's choice.
Alternative flow 1: 1. Actors answer baseline questions. (Workshop is one session) 2. Activity is published.
Postconditions: ● The activity is successfully published.
Alternative flow 2: 1. Actors submit final work. (Workshop is project-based) 2. Activity is published.
Postconditions: ● The activity is successfully published.
Exception flow: ● If required information is missing, the system prompts the Admin to provide the necessary details.

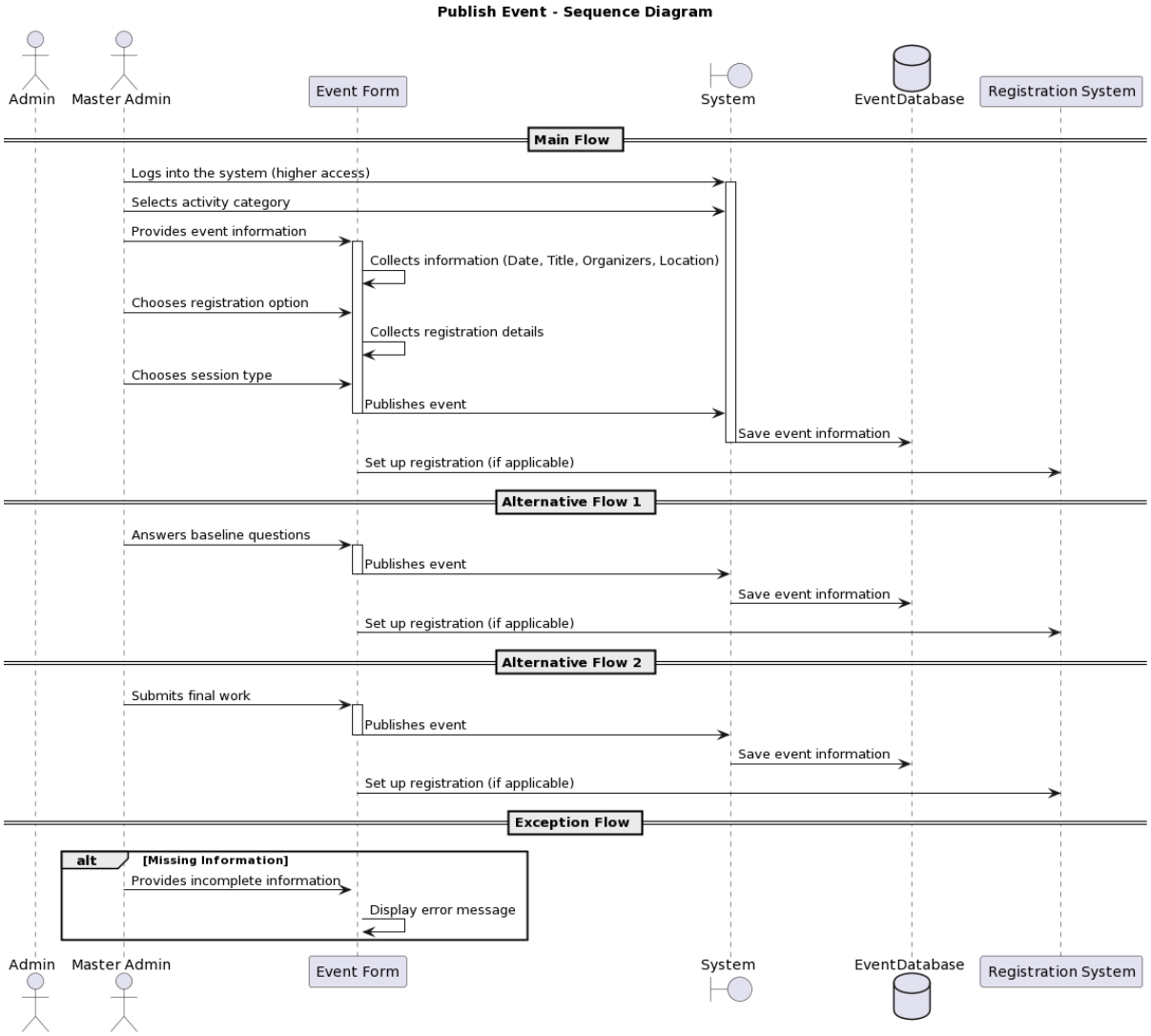


Figure 2.1.9.1: Sequence Diagram of Publish Event

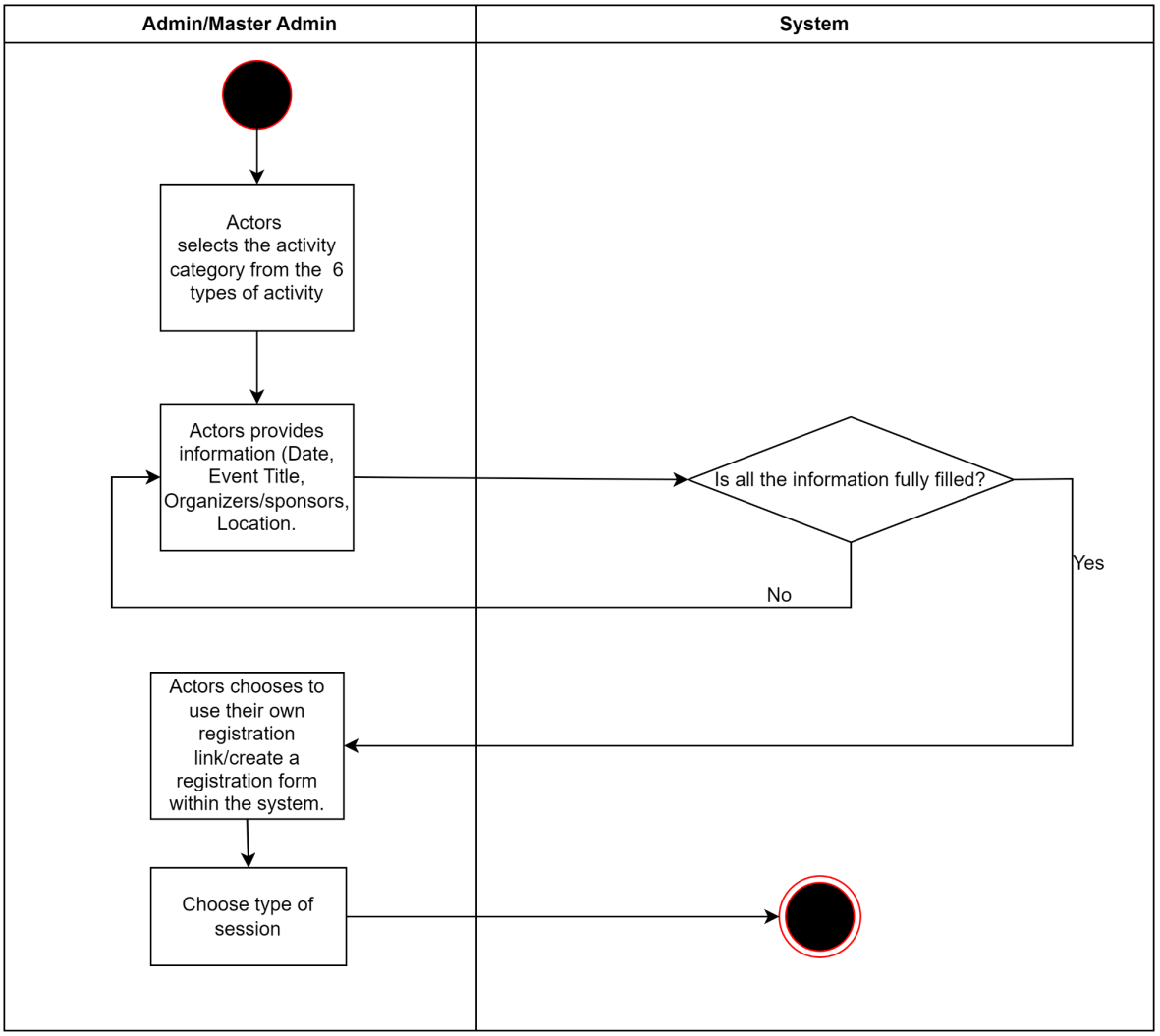


Figure 2.1.9.2:Activity Diagram of Publish Event

2.1.10. UC010: Use Case <Edit YV Activity Details>

Table 2.1.10: Use Case Description for Edit YV Activity Details

Use case: Edit YV Activity Details
ID: UC010
Actors: 1. Master Admin 2. Admin
Preconditions:

● Admin or Master Admin is logged into the system. ● The system has displayed a list of activities that had been published.
Flow of events: 1. Admin or Master Admin navigates to the activity section on the dashboard. 2. The system will display a list of activities that are already published. 3. Admin or Master Admin selects and clicks the activity that he or she wants to edit. 4. The system will display all the information for the relevant activity. 5. Admin or Master Admin clicks the "Edit" button option. 6. Admin or Master Admin scrolls to the parts that he or she wants to edit. 7. Admin or Master Admin makes necessary changes to the activity details. 8. Admin or Master Admin clicks the "Save" button.
Postconditions: ● The activity details are updated in the system.
Exception flow: ● If the activity is not found, the system notifies the actors.

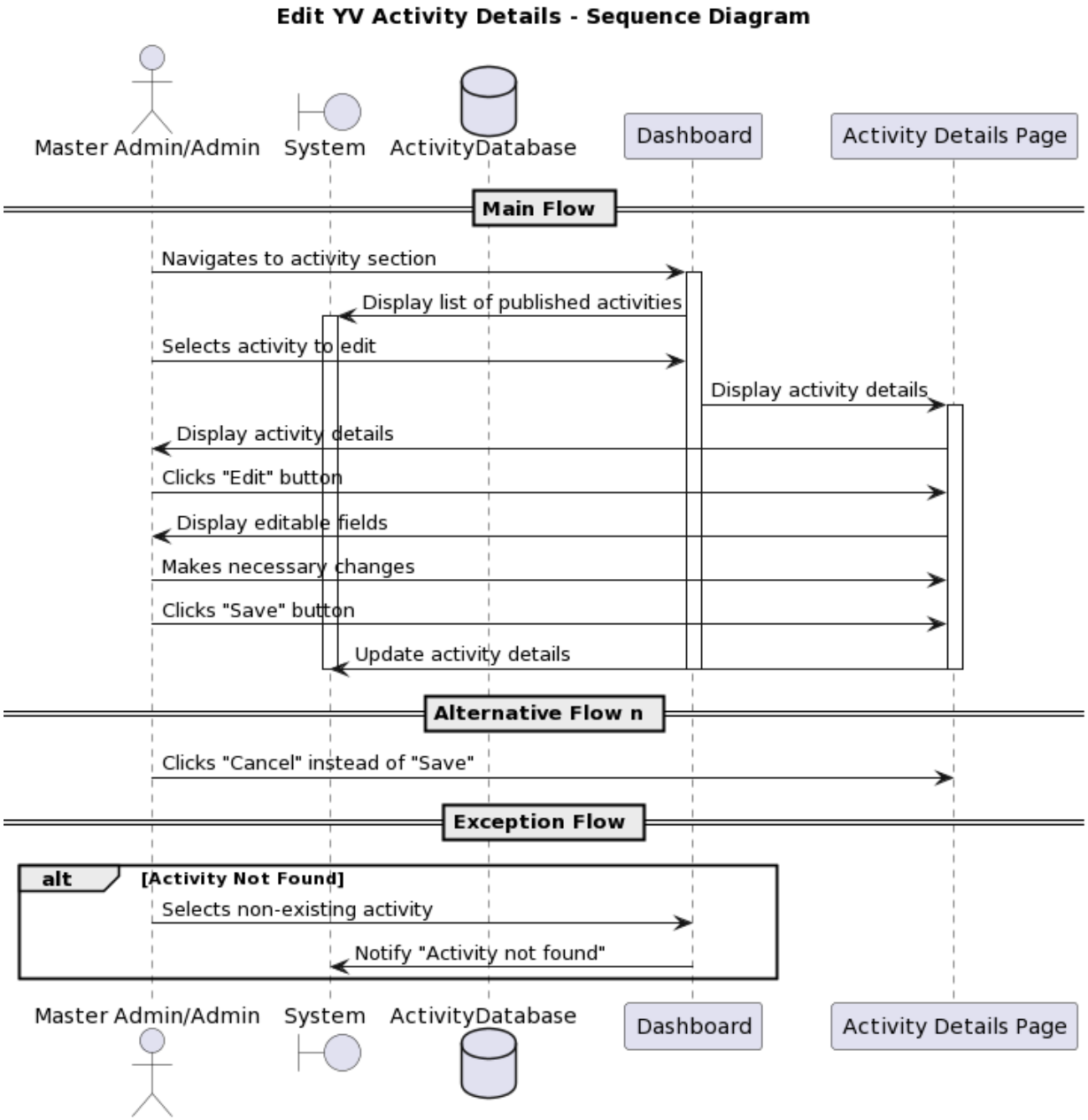


Figure 2.1.10.1: Sequence Diagram of Edit YV Activity Details

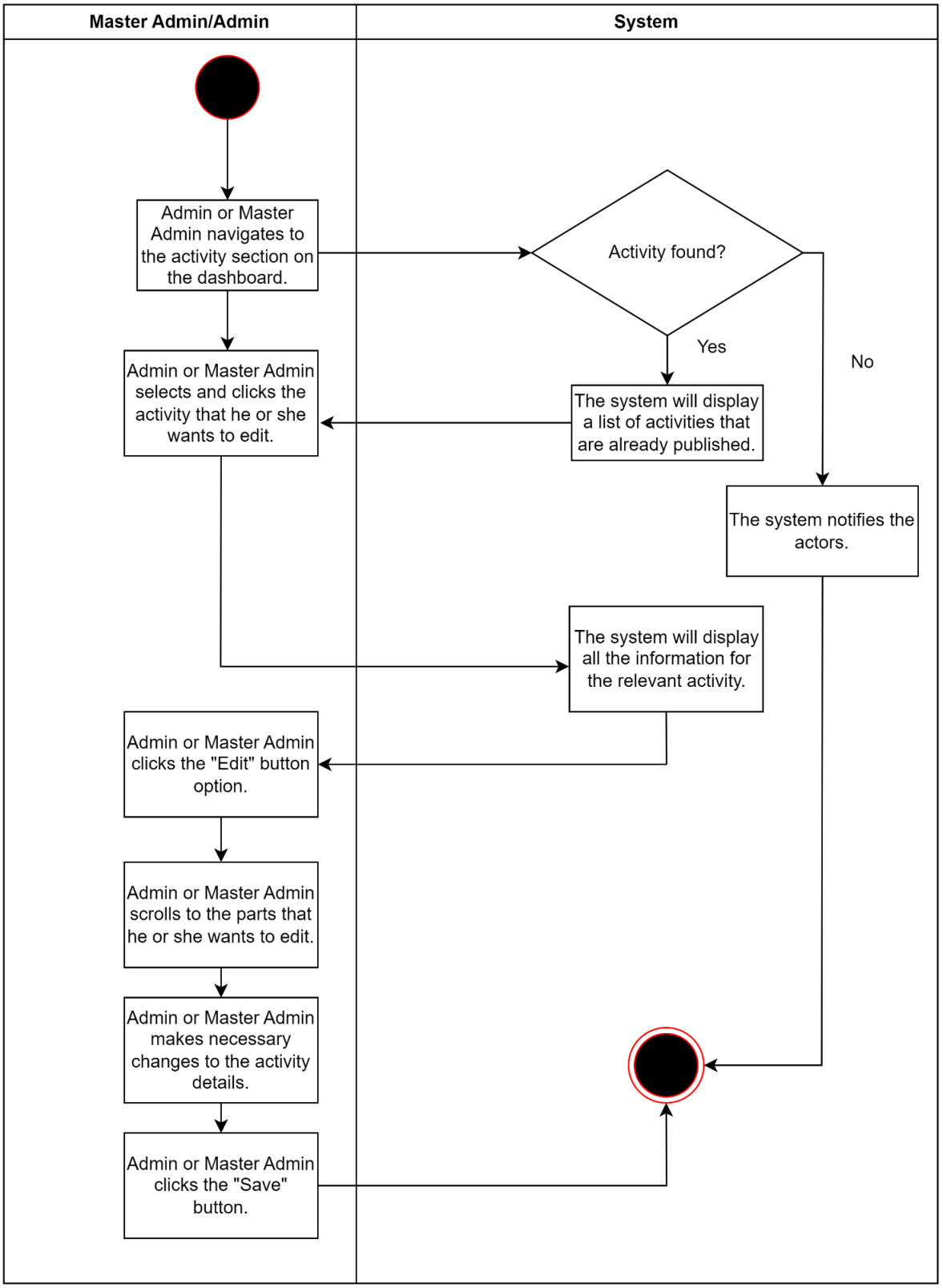


Figure 2.1.10.2: Activity Diagram of Edit YV Activity Details

2.1.11. UC011: Use Case <View Activity>

Table 2.1.11: Use Case Description for View Activity

Use case: View Activity
ID: UC011
Actors: 1. Master Administrator 2. Administrator 3. Student
Preconditions: ● Actor is logged into the system.
Flow of events: 1. Actor selects the "View Activity" option. 2. Actor views the details of the desired activity.
Postconditions: ● The actor has successfully viewed the activity details.
Exception flow: ● If the activity is not found, the system displays the actor ‘No activity found’.

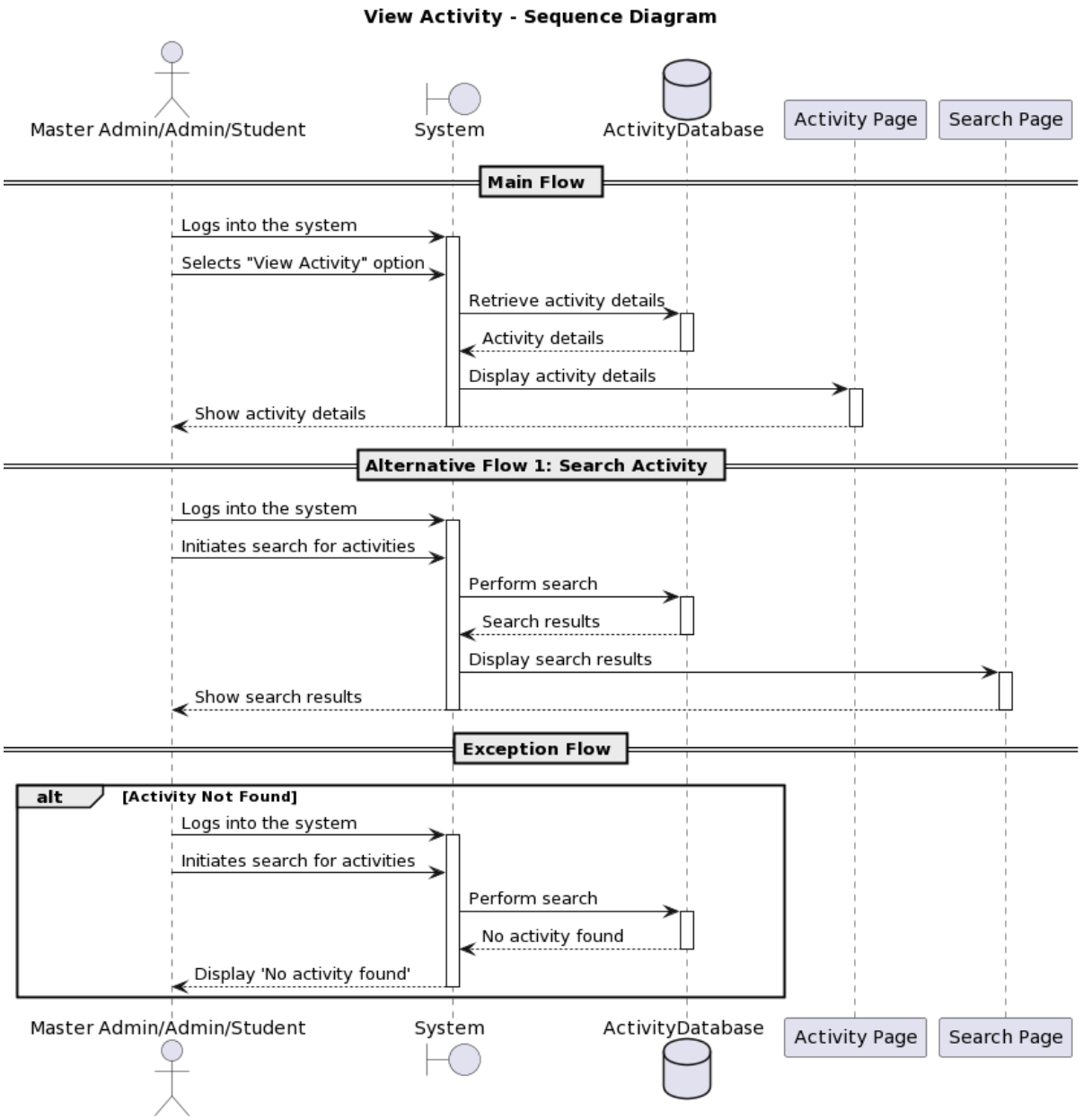


Figure 2.1.11.1: Sequence Diagram of View Activity

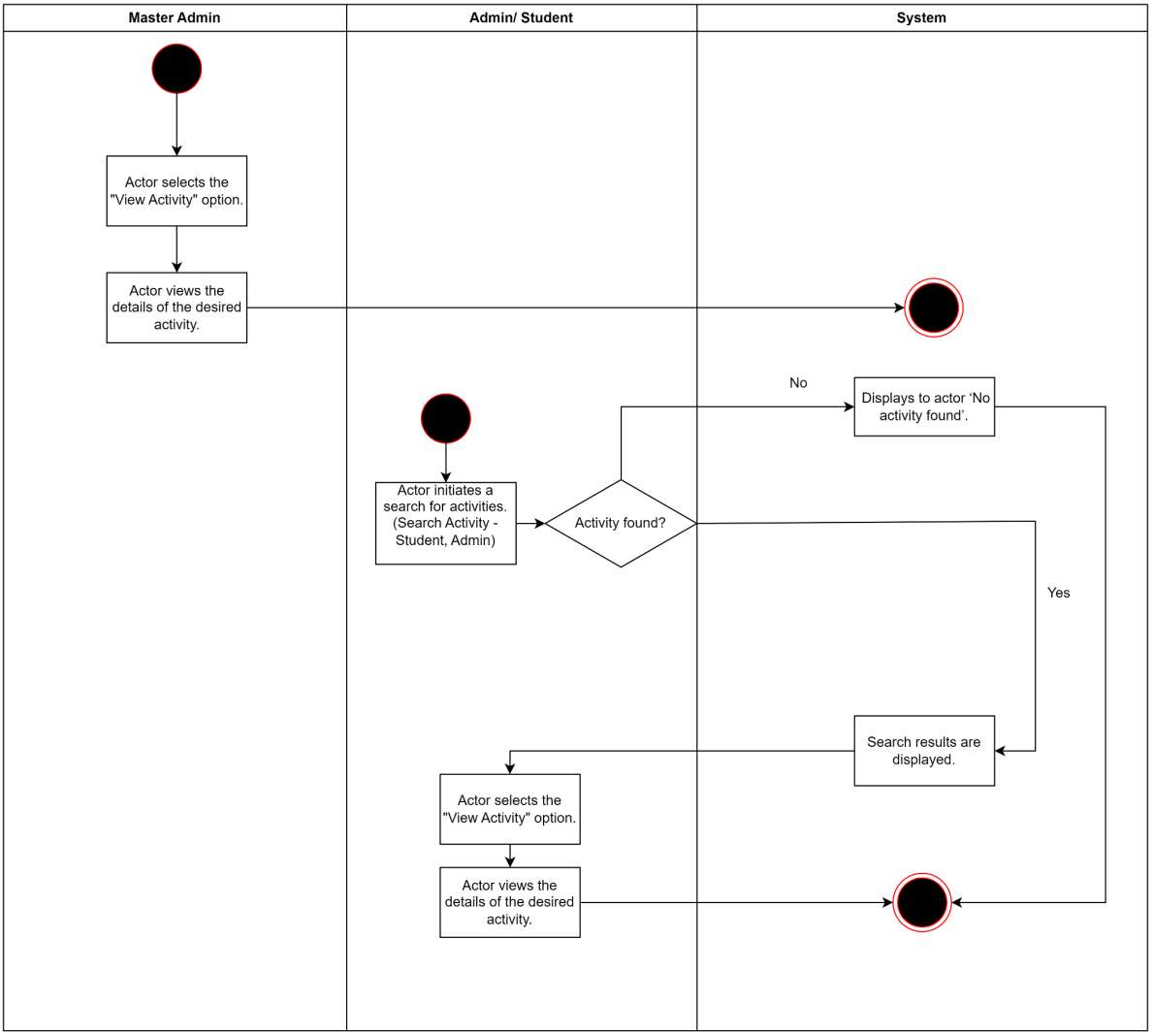


Figure 2.1.11.2: Activity Diagram of View Activity

Module 005: Registration module

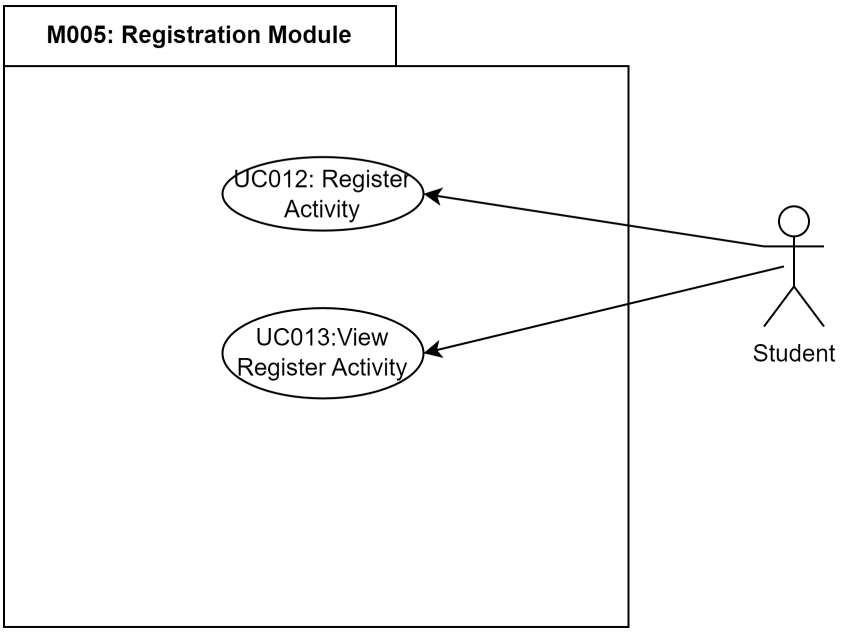


Figure 2.1 (e): Use Case Diagram for Registration module

The use case includes in this create event module is stated as below:

I. UC012: Register Activity
II. UC013: View Registered YV Activity

2.1.12. UC012: Use Case <Register Activity >

Table 2.1.12: Use Case Description for Register Activity

Use case: Register Activity
ID: UC012
Actors: 1. Student
Preconditions: ● The Admin or Master Admin created YV activity.

• The student logged in the system
Flow of events: 1. Actor navigates to register activity. 2. Present list of available activities. 3. Student clicks the 'Register' button. 4. The system made the 'Register' button disabled and turned dull yellow.
Postconditions: • Activity registered show in Activity Registered List • 'Register' button disabled and turns dull yellow.

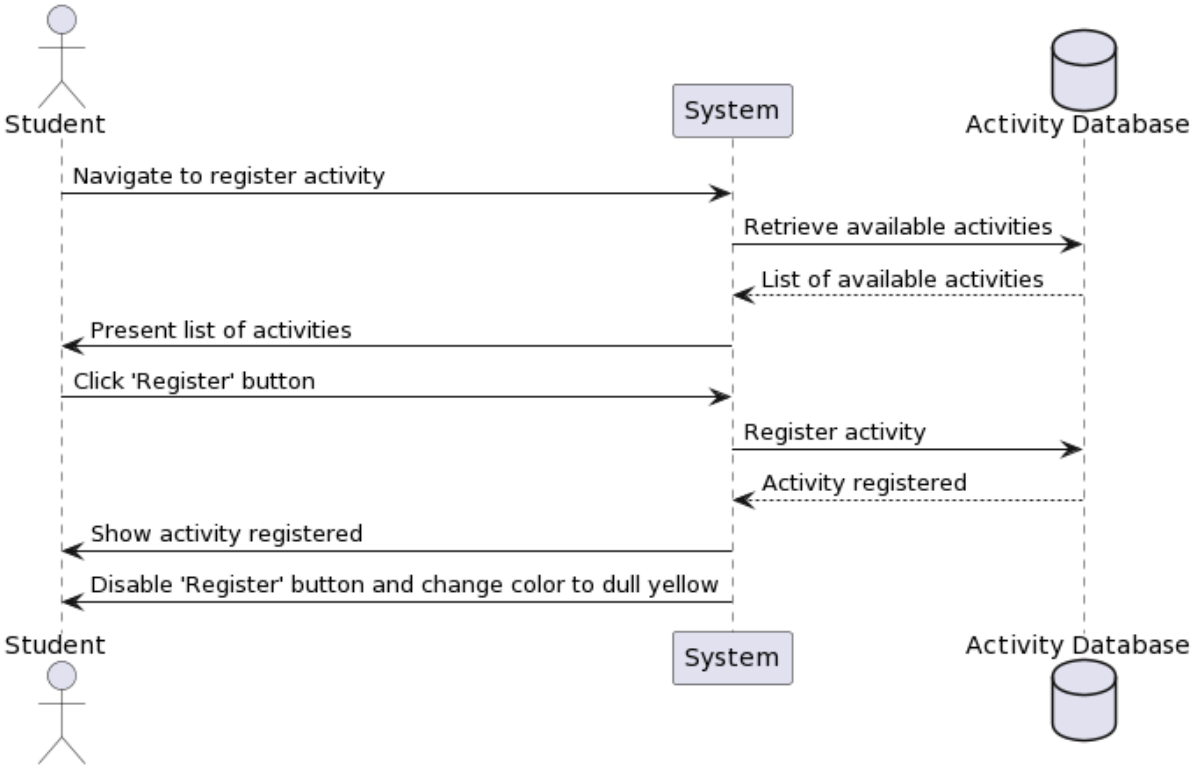


Figure 2.1.12.1: Sequence Diagram of Register Activity

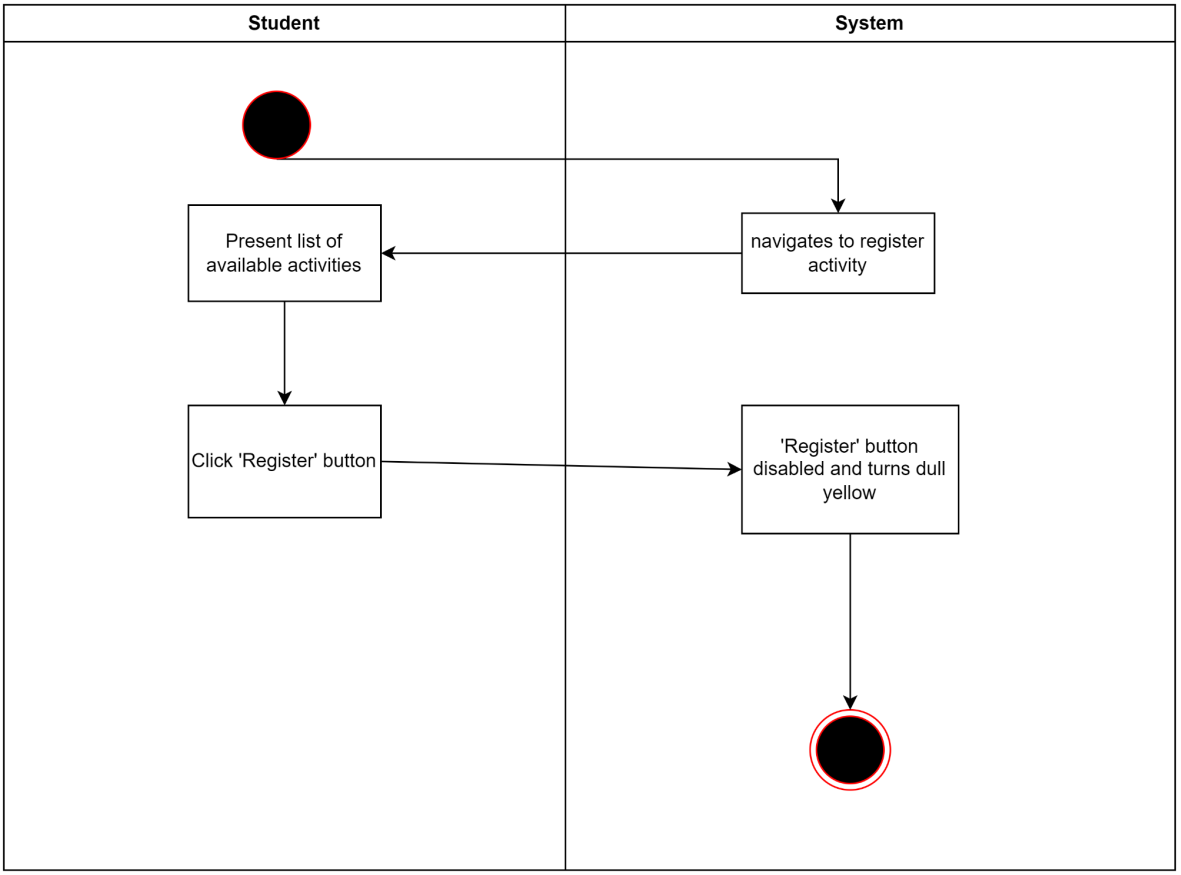


Figure 2.1.12.2: Activity Diagram of Register Activity

2.1.13. UC013: Use Case <View Register Activity >

Table 2.1.13: Use Case Description for View Register Activity

Use case: View Register Activity
ID: UC013
Actors: 1. Student
Preconditions: • The Admin or Master Admin created YV activity. • The student logged in the system . • The student registered the activity.

Flow of events:
1. Actor selects the "Manage Registered Activity" option. 2. Actor views the details of the registered activity.
Postconditions:
● Activity registered show in Activity Registered List ● 'Register' button disabled and turns dull yellow.

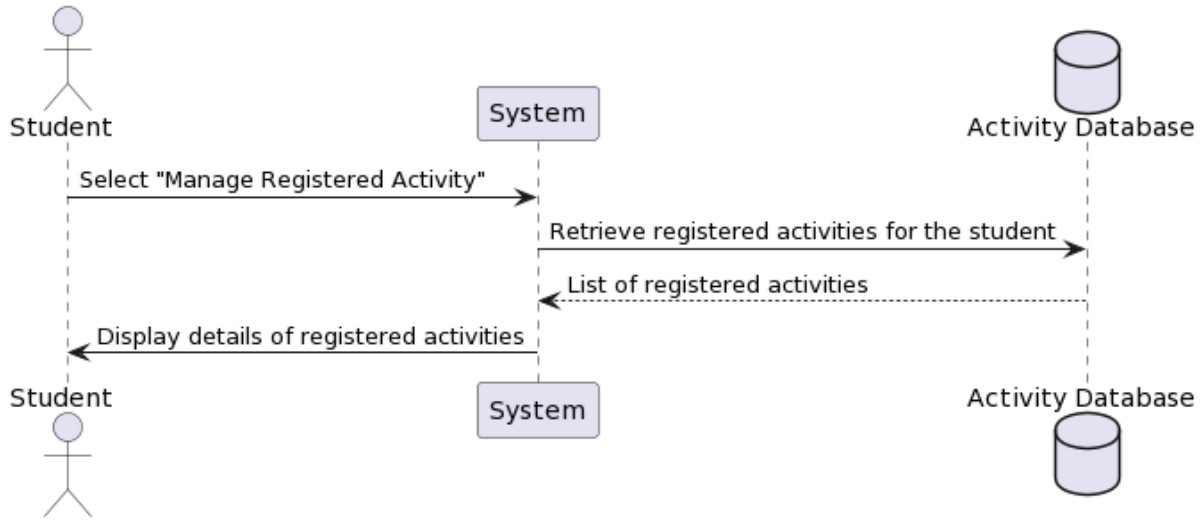


Figure 2.1.13.1: Sequence Diagram of View Register Activity

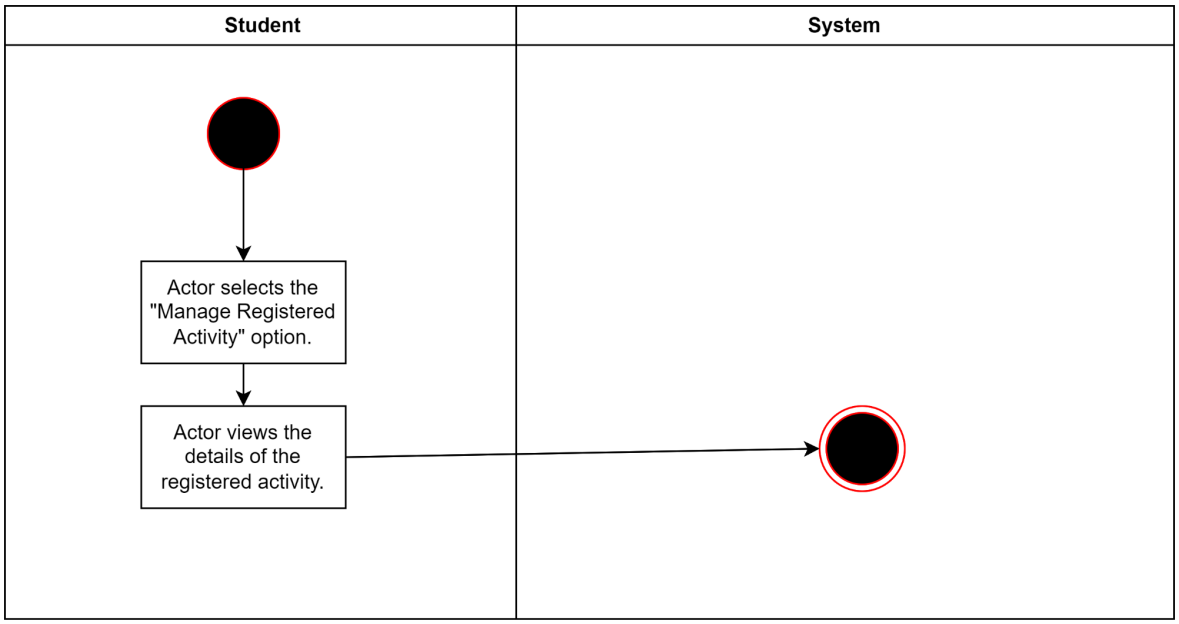


Figure 2.1.13.2: Activity Diagram of View Register Activity

Module 006: Feedback module

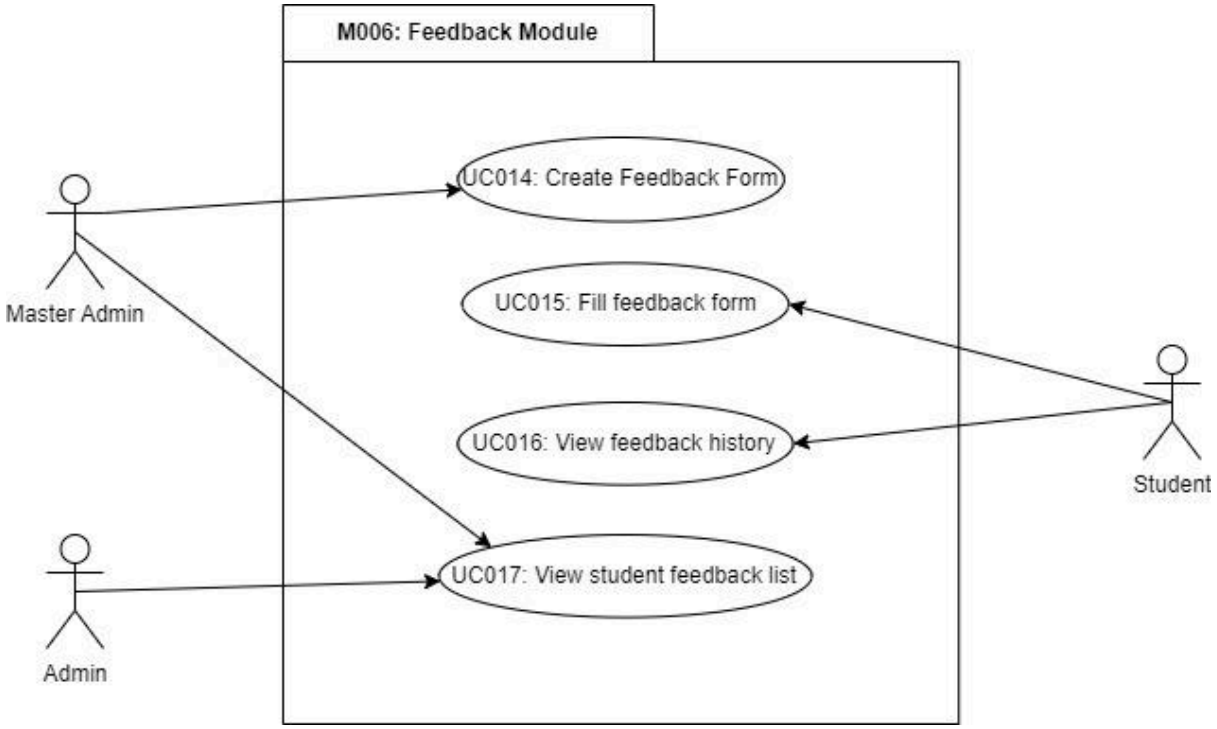


Figure 2.1 (f): Use Case Diagram for Feedback module

The use case includes in this create event module is stated as below:

I. UC014: Create Feedback From
II. UC015: Fill Feedback Form
III. UC016: View Feedback History
IV. UC017: View Student Feedback List

2.1.14. UC014: Use Case <Create Feedback Form>

Table 2.1.14: Use Case Description for Create Feedback Form

Use case: Create Feedback Form
ID: UC014
Actors: 1. Master Admin

2. Admin
Preconditions: ● Master Admin or Admin logged in to the “StuPort” system.
Flow of events: 1. The system displays a list of available activities. 2. The Master Admin or Admin selects the specific activity for which feedback links need to be created. 3. The Master Admin or Admin clicks the “Add Feedback” button. 4. The system displays a form setting. 5. The Master Admin or Admin upload and paste the feedback link which consists of the questions they want to know about students’ feedback on certain activity. 6. The Master Admin clicks the “Submit” button.
Postconditions: ● The feedback form link for the selected activity is successfully uploaded and stored in the system. ● The students can view and access the uploaded links of their registered activity.
Alternative flow 1: Feedback form exists. 1. In Step 2, if the feedback form already exists on the activity page, the Master Admin or Admin cannot click the “Add Feedback” button anymore, the system will display the link they have been added for certain activity.
Postconditions: ● The Master Admin or Admin is prompted to upload feedback form link for another activity.

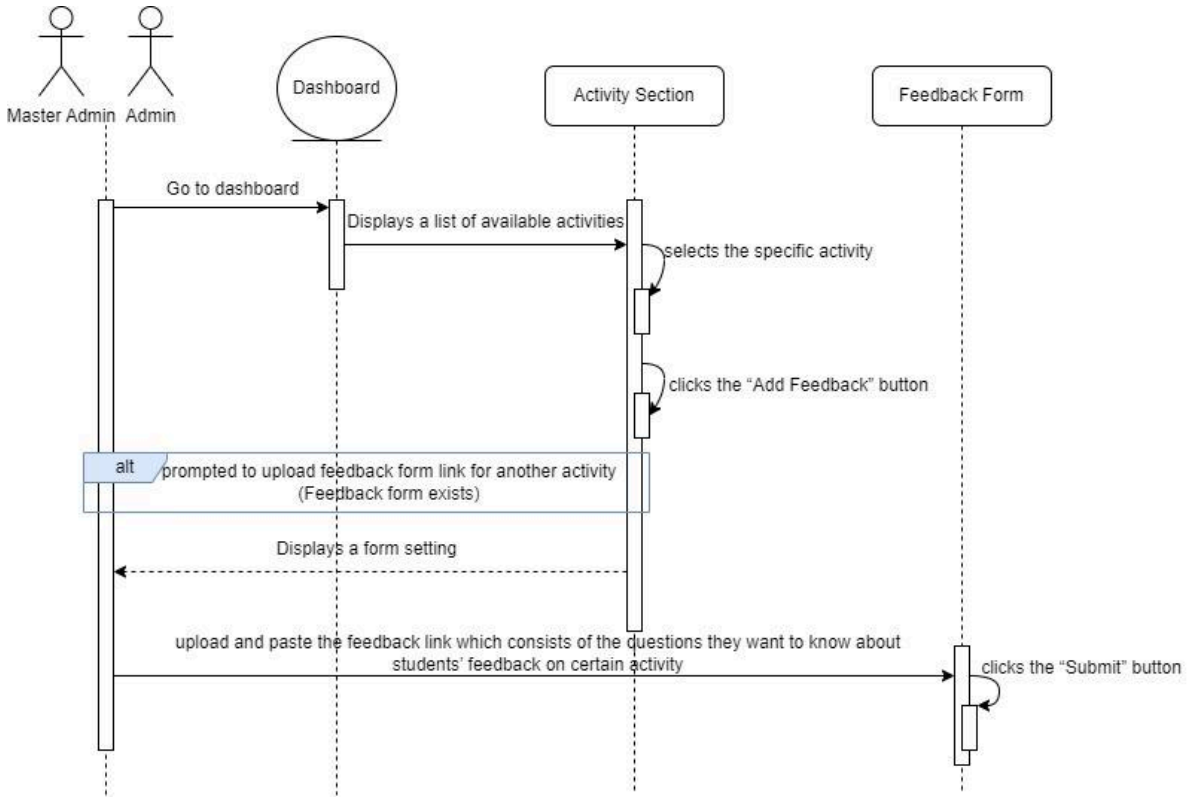


Figure 2.1.14.1: Sequence Diagram of Create Feedback Form

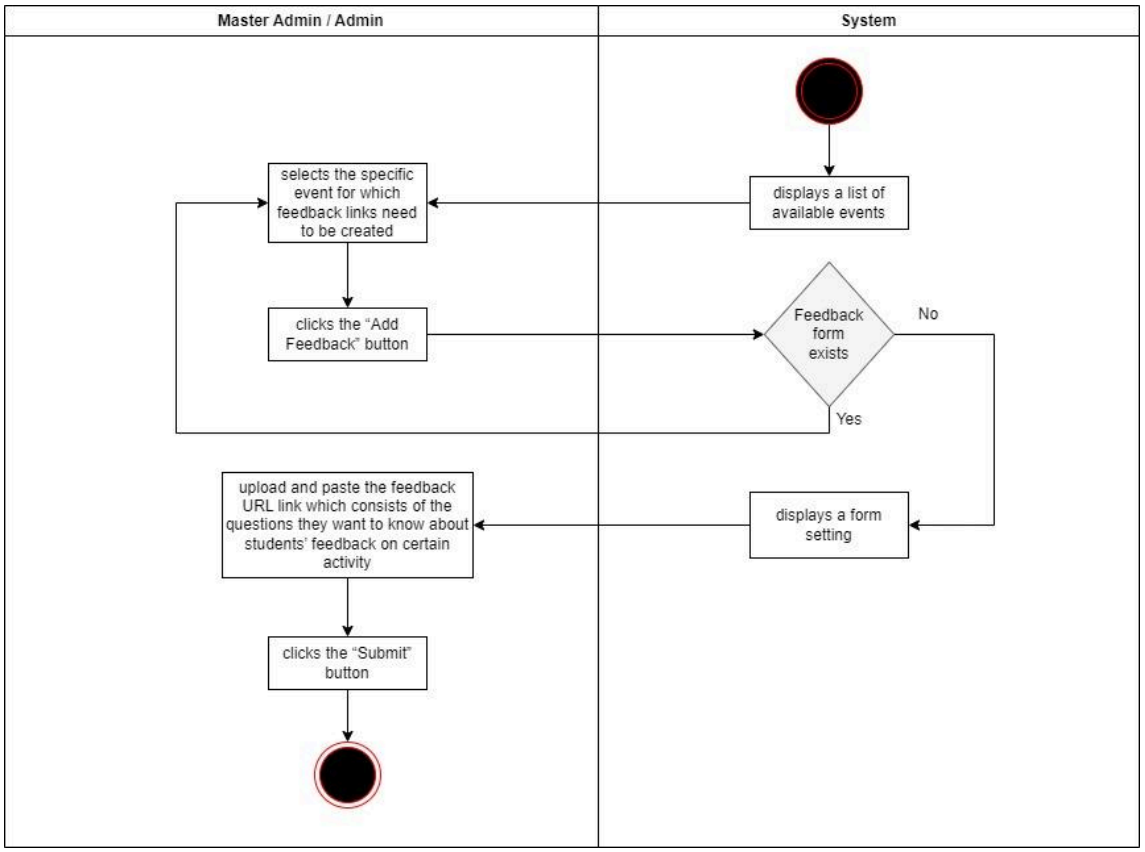


Figure 2.1.14.2: Activity Diagram of Create Feedback Form

2.1.15. UC015: Use Case <Fill Feedback Form>

Table 2.1.15: Use Case Description for Fill Feedback Form

Use case: Fill Feedback form
ID: UC015
Actors: 1. Student
Preconditions: ● The student logged in to the "StuPort" system. ● The student is registered for the activity. ● The activity has taken place, and the feedback link is open.
Flow of events: 1. The student navigates to the "Activity" section on the dashboard. 2. The system will display a list of registered and attended activities. 3. The student selects on the activity that he or she wants to fill in the feedback. 4. The student clicks on a feedback link which is shown on the selected activity section. 5. The system displays the contents of the link which is a feedback form. 6. The student fills in the required information and provides feedback regarding the activity experience.
Postconditions: ● The student's feedback for the activity is successfully recorded in the system for further operation.

Alternative flow 1: Late Submission. 1. In step 5, if the student attempts to click the feedback link after the submission deadline has passed, the system will display an error message. 2. The student will be prompted to go back to the activity section on the dashboard interface.
Postconditions: ● The feedback forms are not open to students to fill in anymore due to being late.
Exception flow 1: Feedback Form Not Found 1. In step 4, if the student attempts to fill in the feedback link before the activity has been held or before the feedback form link has been uploaded, he or she cannot see any feedback form along with the selected event details.
Exception flow 2: Required field(s) is/are empty. 1. In step 6, if the student does not fill in the required field in feedback form, an error message will be displayed by the system. 2. Submission cannot proceed.

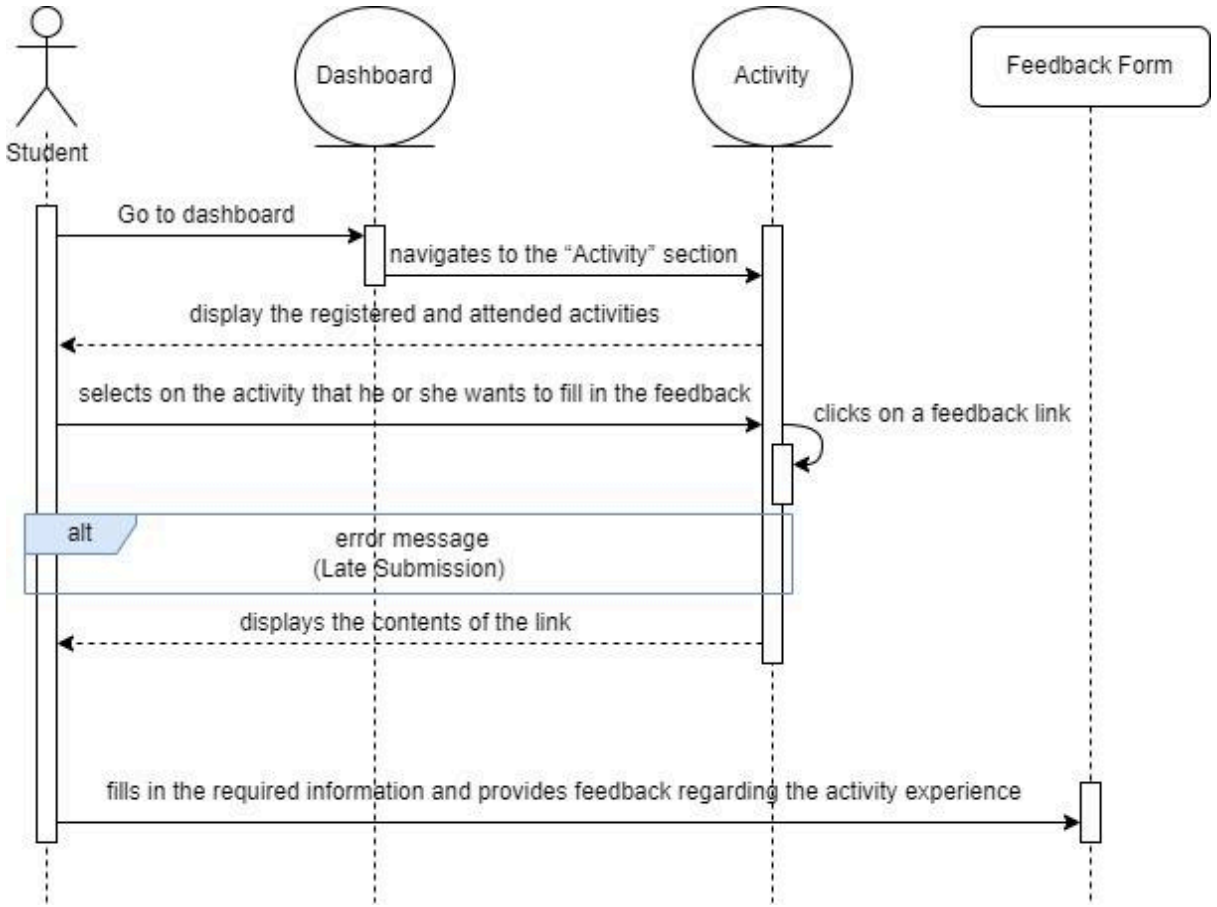


Figure 2.1.15.1: Sequence Diagram of Fill Feedback Form

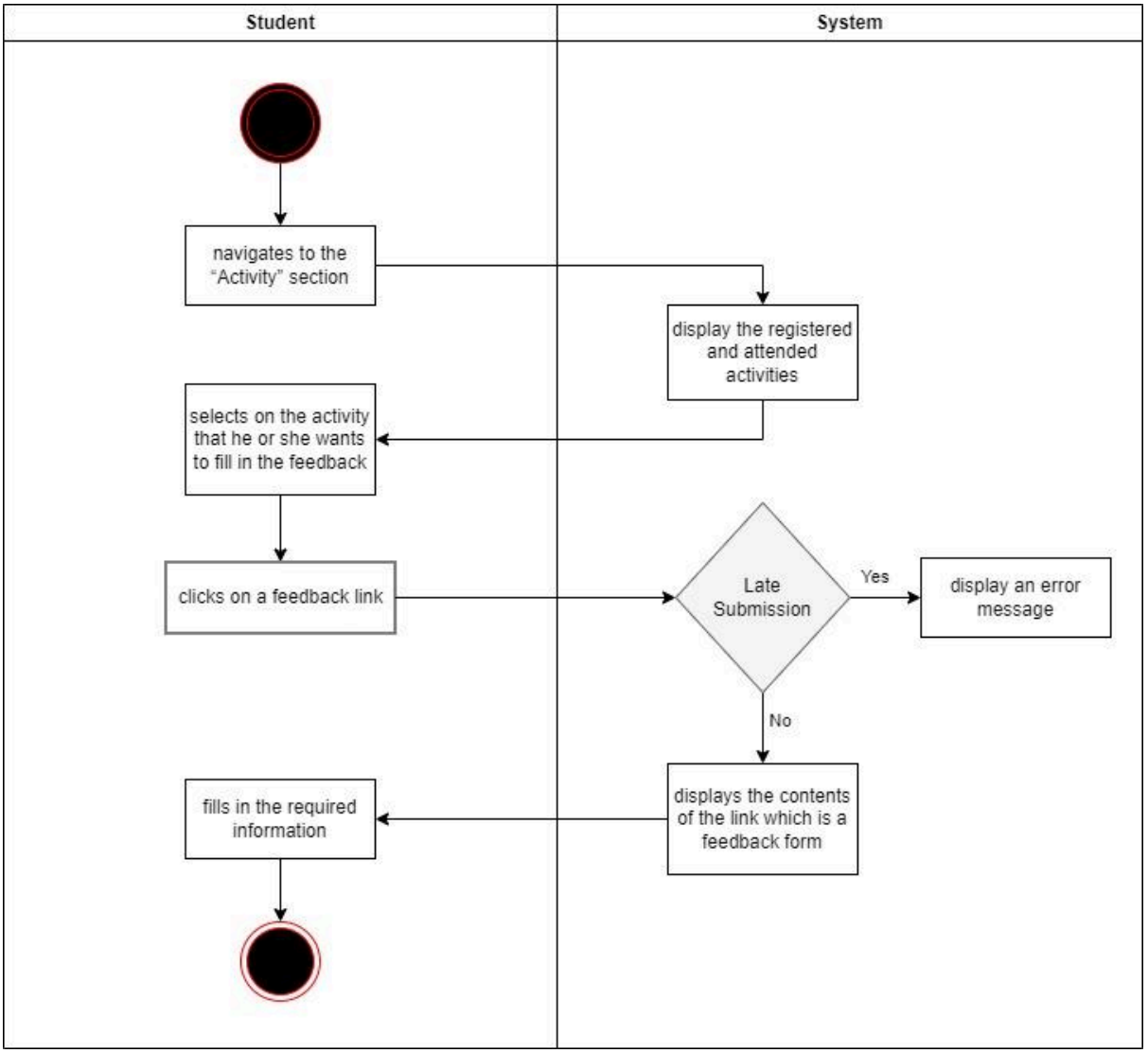


Figure 2.1.15.2: Activity Diagram of Fill Feedback Form

2.1.16. UC016: Use Case <View Feedback History>

Table 2.1.16: Use Case Description for View Feedback History

Use case: View Feedback History
ID: UC016
Actors: 1. Student
Preconditions:

• The feedback form is filled in by the student and uploaded successfully. • The student logged in to the “StuPort” system.
Flow of events: 1. The student navigates to the activity section on the dashboard. 2. The system presents a list of activities that students have registered and attended before. 3. The student selects and 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.
Postconditions: • The student has successfully viewed the feedback for the selected activity.
Exception flow (if any): No Feedback History Can Be Seen. 1. In step 4, if no student has filled in the feedback form for the selected event, the system will not display any feedback history which indicates that there is no feedback filled in yet. 2. The student will be prompted to continue fill in the feedback form of selected activity.

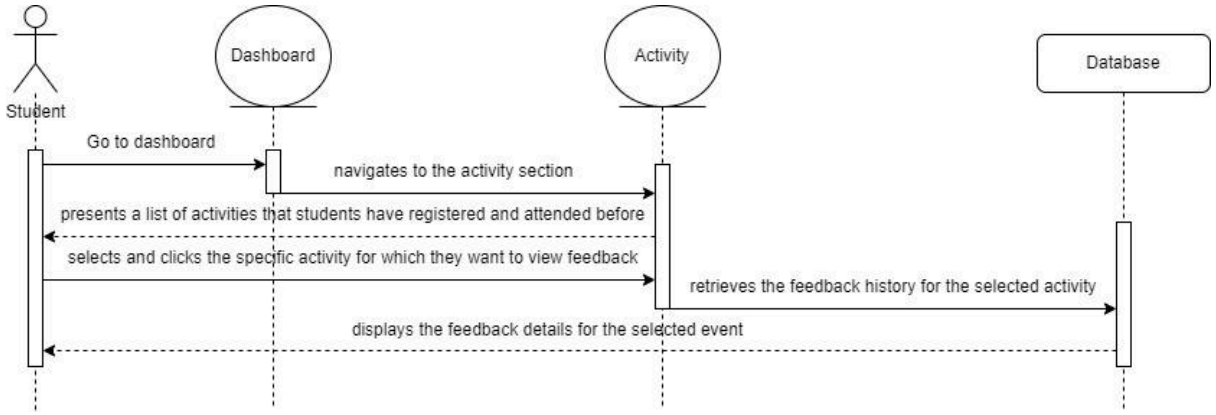


Figure 2.1.16.1: Sequence Diagram of View Feedback History

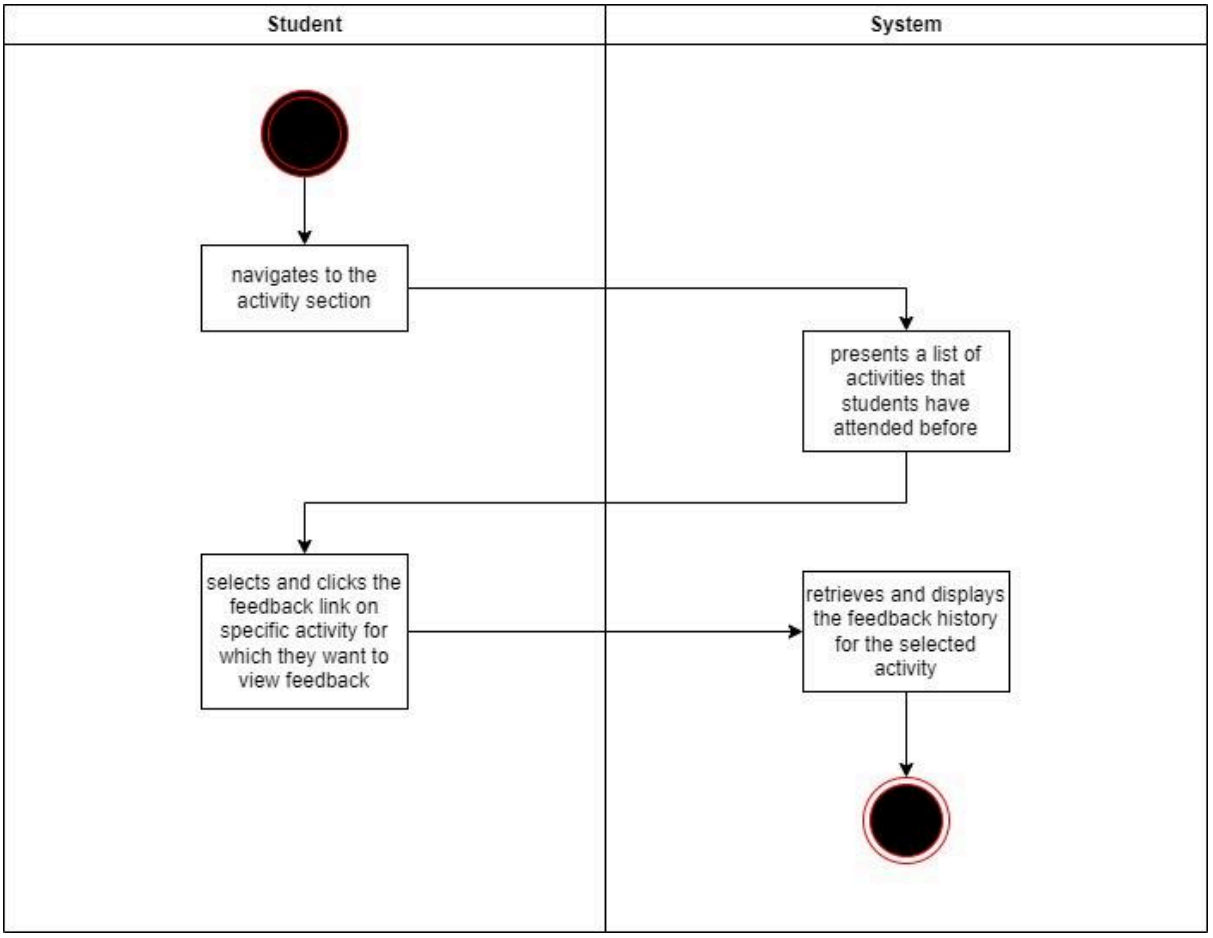


Figure 2.1.16.2: Activity Diagram of View Feedback History

2.1.17. UC017: Use Case <View Student Feedback List>

Table 2.1.17: Use Case Description for View Student Feedback List

Use case: View Student Feedback List
ID: UC017
Actors: 1. Master Admin 2. Admin
Preconditions: ● The feedback form has been uploaded and opened to the public.

• The Master Admin or Admin is successfully logged into the "StuPort" system.
Flow of events: 1. The Master Admin or Admin navigates to the activity section on the dashboard. 2. The system presents a list of activities along with the respective registration and feedback data. 3. The Master Admin or Admin selects the specific activity for which they want to view feedback. 4. The Master Admin or Admin clicks on the feedback link for the selected activity. 5. The system retrieves and displays the feedback details for the selected activity, including the list of students with their respective feedback replies in a list.
Postconditions: • The Master Admin or Admin has successfully viewed the feedback for the selected event. • The Master Admin or Admin can record and pay attention to the student feedback data for further improvement in upcoming activities.
Alternative flow 1: Empty Feedback. 1. In step 5, if no student has filled in the attendance form for the selected event, the system displays nothing which indicates that there is no feedback received yet. 2. The Master Admin or Admin can choose to either continue viewing the feedback of another activity or proceed to other tasks.
Postconditions: • The system does not display any student feedback details for the selected event.

• The Master Admin or Admin is aware that no feedback has been filled by students for the selected event.
Exception flow 1: No Feedback Can Be Found. 1. In step 3, if the Master Admin or Admin selects the activity that the feedback that does not exist or hasn't been published, the system will display nothing. 2. The Master Admin or Admin is prompted to select another activity from the list of activities to view feedback data.

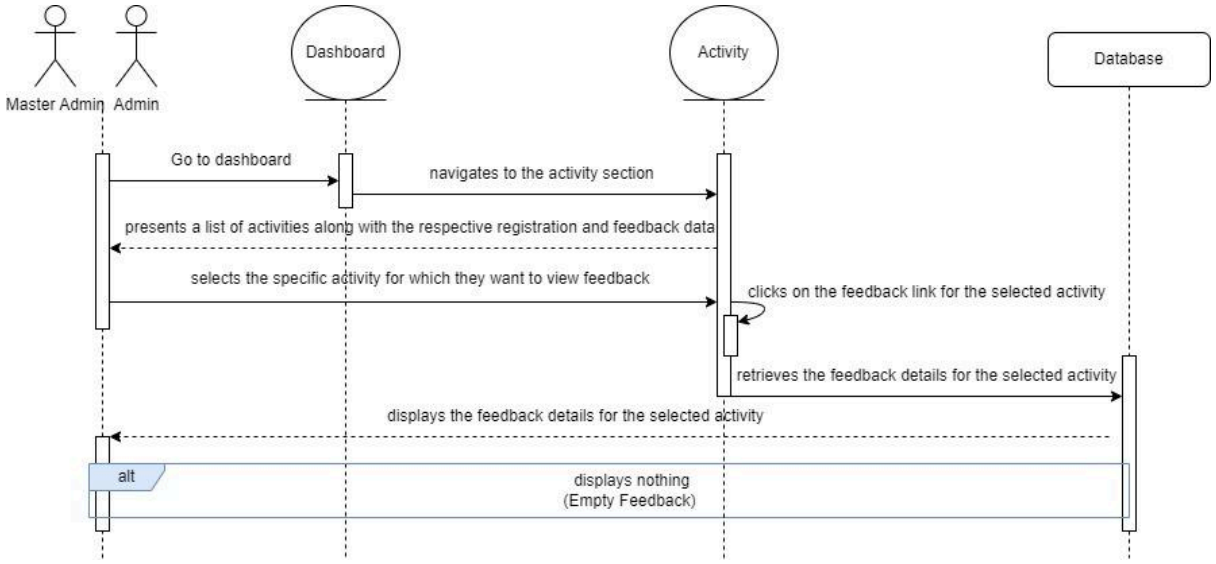


Figure 2.1.17.1: Sequence Diagram of View Student Feedback List

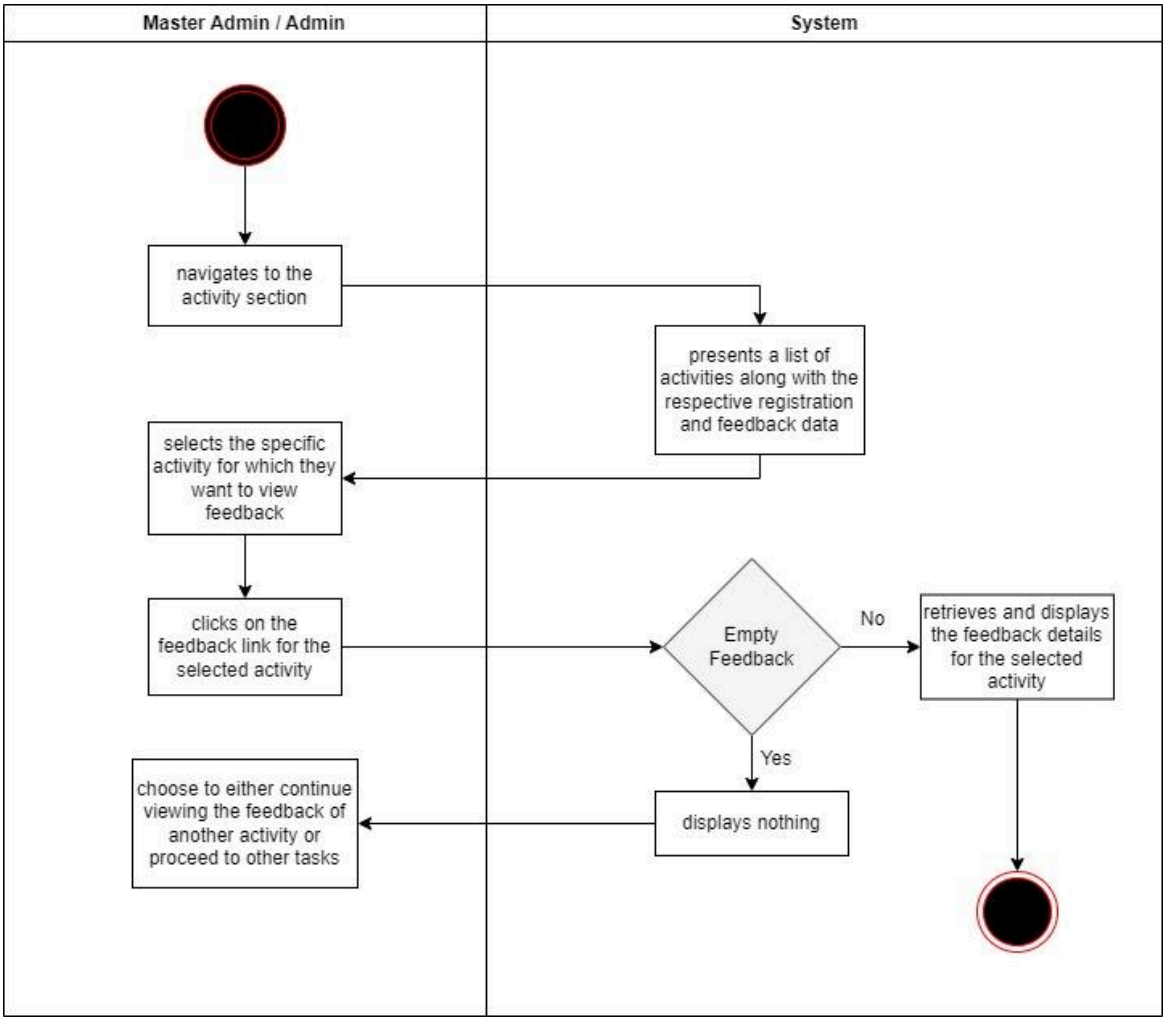


Figure 2.1.17.2: Activity Diagram of View Student Feedback List

Module 007: Reward module

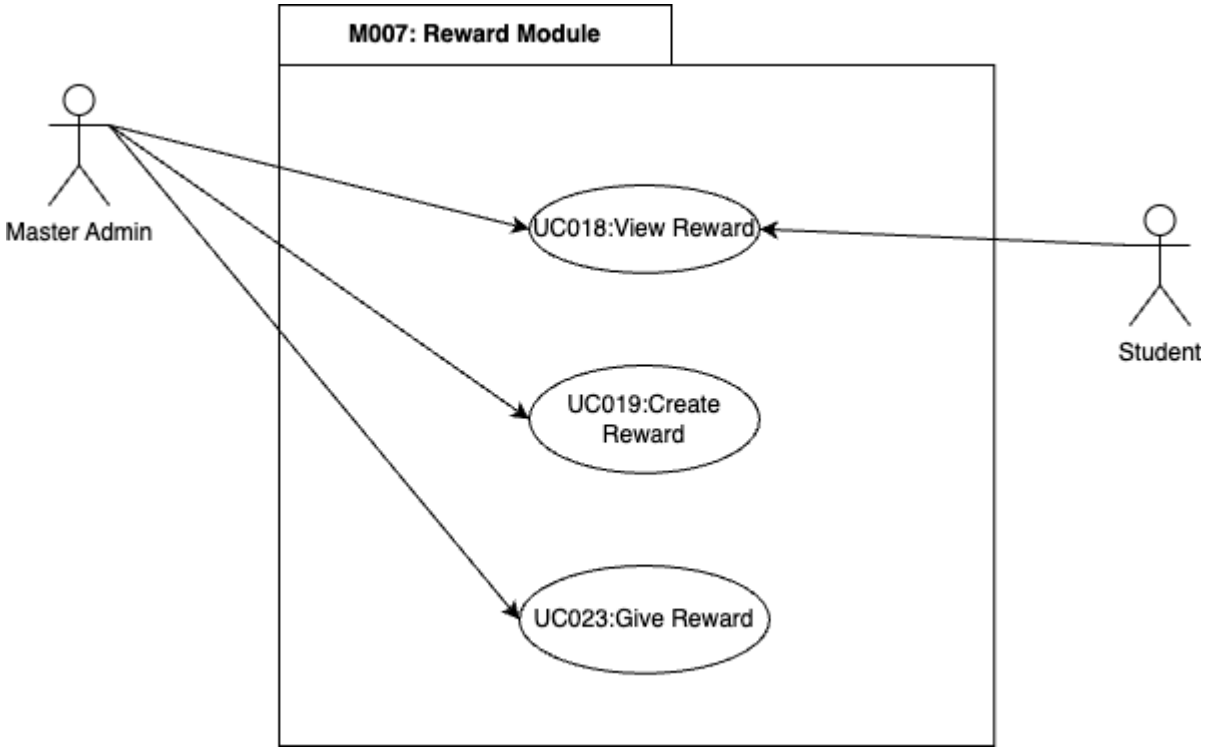


Figure 2.1 (g): Use Case Diagram for Reward module

The use case includes in this create event module is stated as below:

I. UC018: View Reward
II. UC019: Create Reward
III. UC020: Give Reward

2.1.18. UC018: Use Case <View Reward>

Table 2.1.18: Use Case Description for View Reward

Use case: View Reward
ID: UC018
Actors: 1. Master Admin 2. Student

Preconditions:

- The actor (Admin, Master Admin, or Student) is authenticated.
- There are rewards available in the system.

Flow of events:

1. The Admin, Master Admin, or Student navigates to the "View Reward" section in the system.
2. The system presents a user interface displaying a list or summary of available rewards.
3. The Admin, Master Admin, or Student selects specific criteria or filters to refine the displayed rewards.
4. The system retrieves and displays the relevant rewards based on the specified criteria.
5. The Admin, Master Admin, or Student reviews the displayed rewards.

Postconditions:

- The system maintains the current state of the reward display.
- The Admin, Master Admin, or Student has reviewed the rewards.

Alternative flow 1: Invalid Input.

1. If the input provided by the Admin, Master Admin, or Student is invalid or incomplete, the system notifies the user.
2. The user corrects the input.
3. The flow continues from step 4 of the main flow.

Postconditions:

- The system provides feedback on the nature of the invalid input.
- The user corrects the input, and the main flow continues.

Alternative flow 2: No Rewards Found. 1. If there are no rewards matching the specified criteria, the system notifies the Admin, Master Admin, or Student. 2. The user adjusts the criteria or filters. 3. The flow continues from step 4 of the main flow.
Postconditions: ● The system provides feedback indicating no matching rewards. ● The Admin, Master Admin, or Student adjusts the criteria, and the main flow continues.

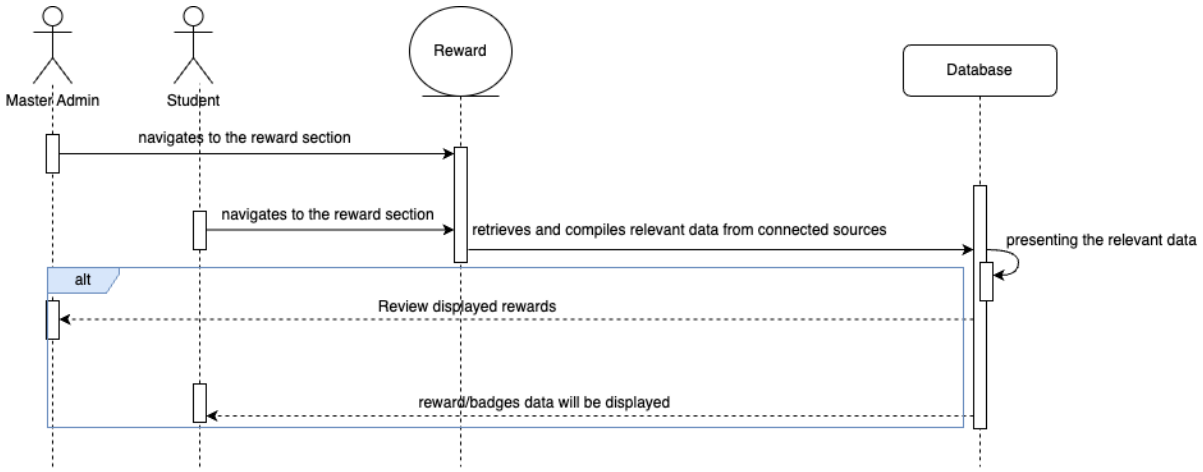


Figure 2.1.18.1: Sequence Diagram of View Reward

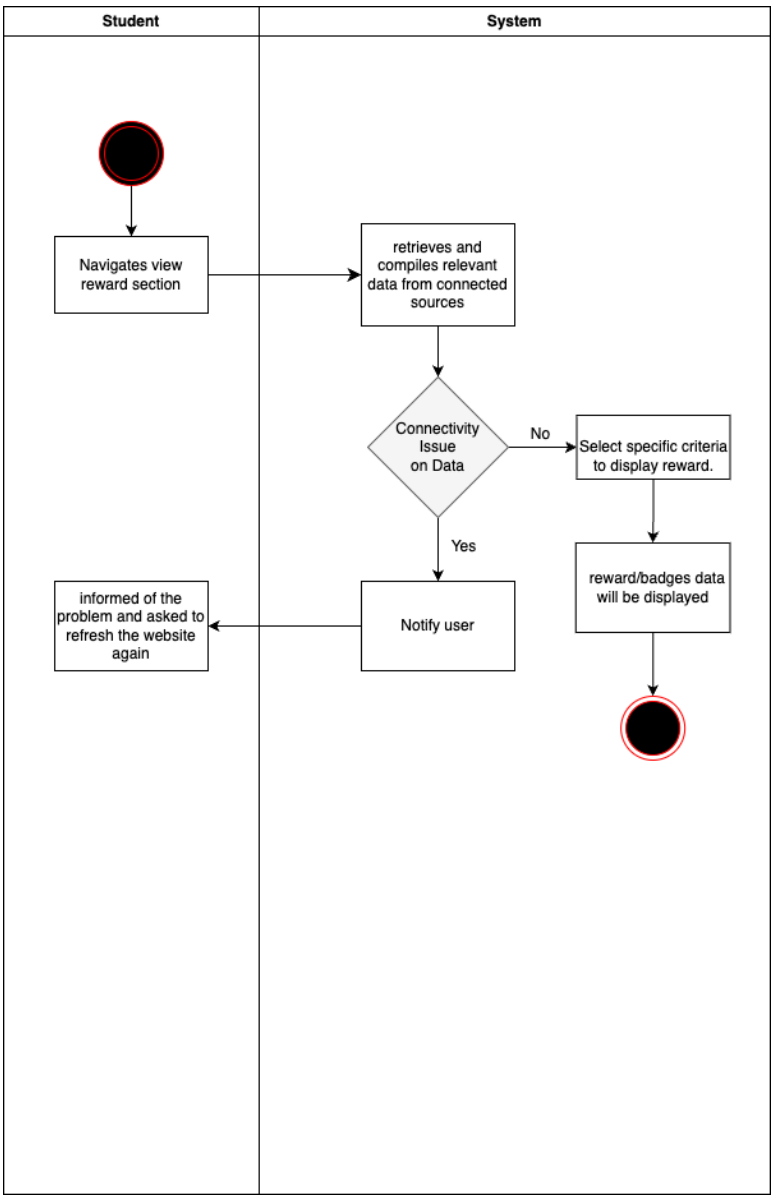


Figure 2.1.18.2: Activity Diagram of View Reward

2.1.19. UC019: Use Case <Create Reward>

Table 2.1.19: Use Case Description for Create Reward

Use case: Create Reward
ID: UC019
Actors: 1. Master Admin

Preconditions:

- Master Admin is authenticated and has appropriate privileges.
- The system is in a state ready to create a new reward.

Flow of events:

1. The Master Admin navigates to the "Reward" section in the system.
2. The system presents a form or interface for the Master Admin to input the details of the new reward.
3. The Master Admin provides information such as the reward name, description, and any associated criteria.
4. The system validates the input data.
5. If validation is successful, the system creates a new reward in the system.
6. The Master Admin receives confirmation of the successful creation of the reward.

Postconditions:

- A new reward is created and stored in the system.
- The Master Admin has access to the newly created reward for management.

Alternative flow 1: Invalid Input.

1. If the input provided by the Master Admin is invalid or incomplete, the system notifies the Master Admin.
2. The Master Admin corrects the input.
3. The flow continues from step 4 of the main flow.

Postconditions:

- The system provides feedback on the nature of the invalid input.
- The Master Admin corrects the input, and the main flow continues.

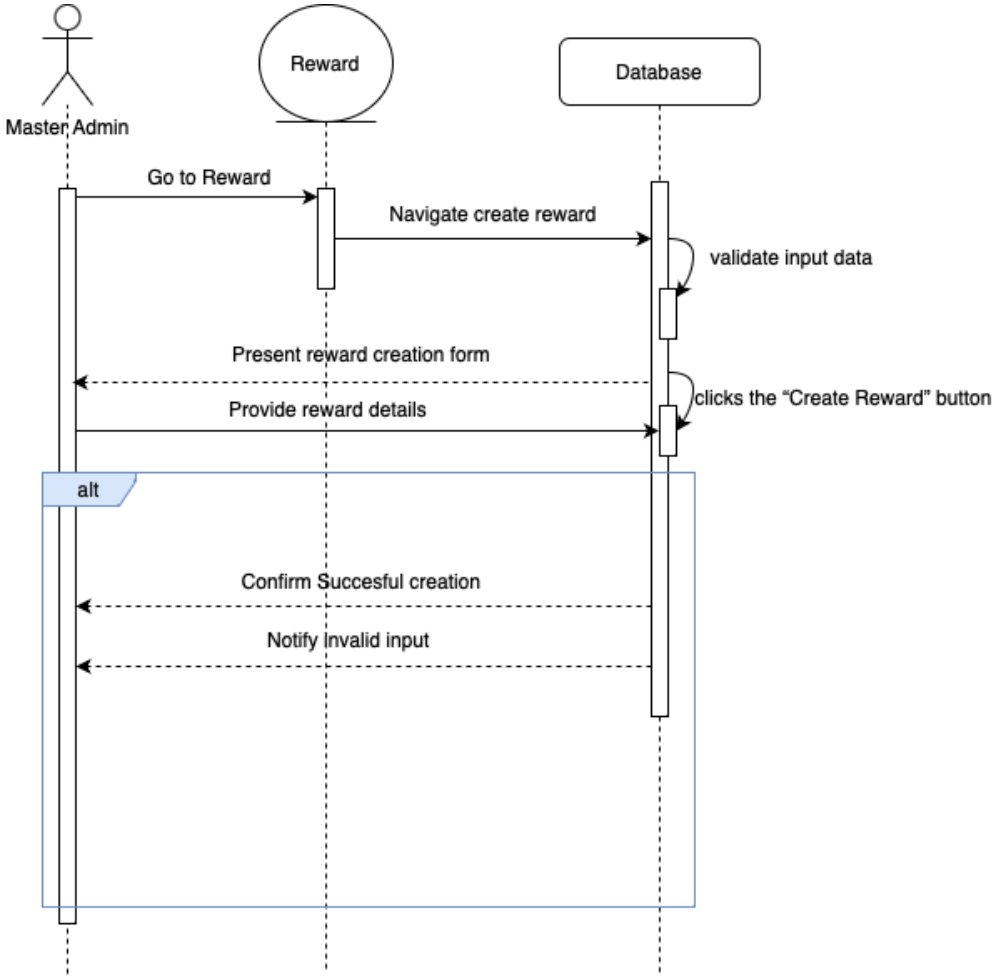


Figure 2.1.19.1: Sequence Diagram of Create Reward

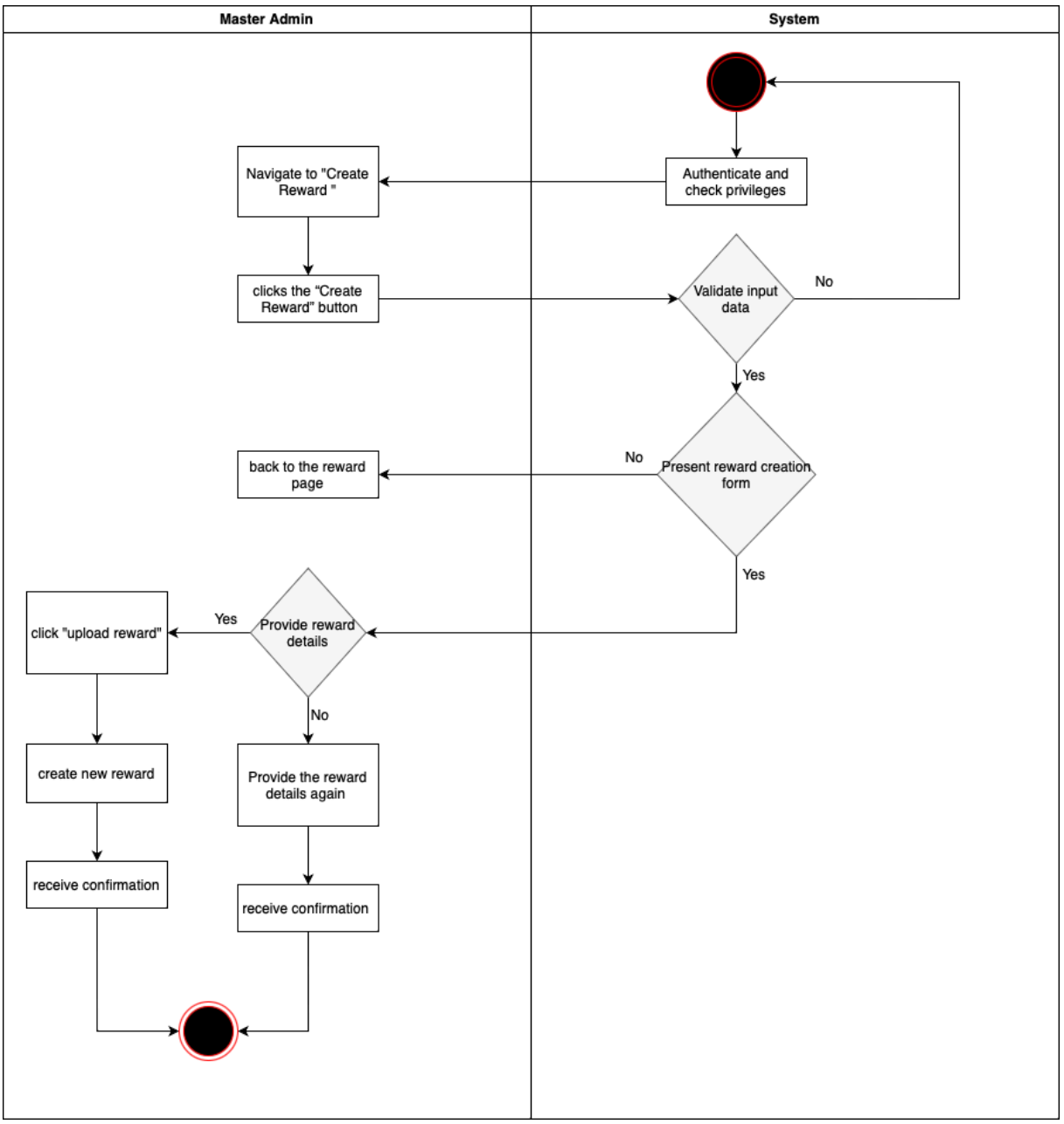


Figure 2.1.19.1: Activity Diagram of Create Reward

2.1.20. UC020: Use Case <Give Reward>

Table 2.1.20: Use Case Description for Give Reward

Use case: Give Reward
ID: UC020
Actors: 1. Master Admin
Preconditions: ● The Master Admin is authenticated and has appropriate privileges. ● There is a user or entity eligible to receive a reward.
Flow of events: 1. The Master Admin navigates to the "Give Reward" section in the system. 2. The system presents a form or interface for the Master Admin to select a user or entity and choose a reward to give. 3. The Master Admin selects the recipient and the specific reward to be given. 4. The system validates the eligibility of the selected recipient for the chosen reward. 5. If eligibility is confirmed, the system processes the reward and updates the recipient's record. 6. The Master Admin receives confirmation of the successful reward giving.
Postconditions: ● The selected user or entity receives the specified reward. ● The system logs the reward transaction.

● The Master Admin has access to the record of the reward given.
Alternative flow 1: Invalid Eligibility. 1. If the selected recipient is not eligible for the chosen reward, the system notifies the Master Admin. 2. The Master Admin may choose an alternative recipient or select a different reward. 3. The flow continues from step 2 of the main flow.
Postconditions: ● The system provides feedback on the invalid eligibility. ● The Master Admin may choose an alternative recipient or reward, and the main flow continues

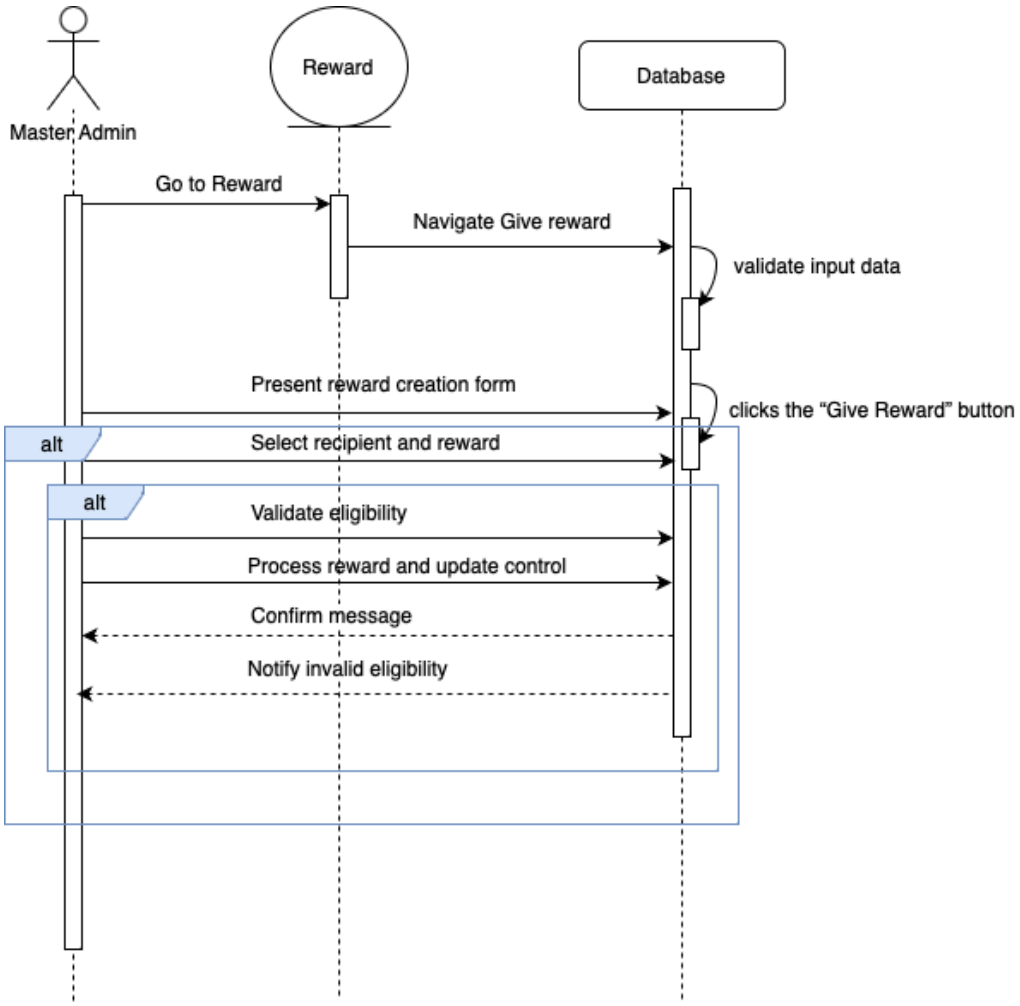


Figure 2.1.20.1: Sequence Diagram of Give Reward

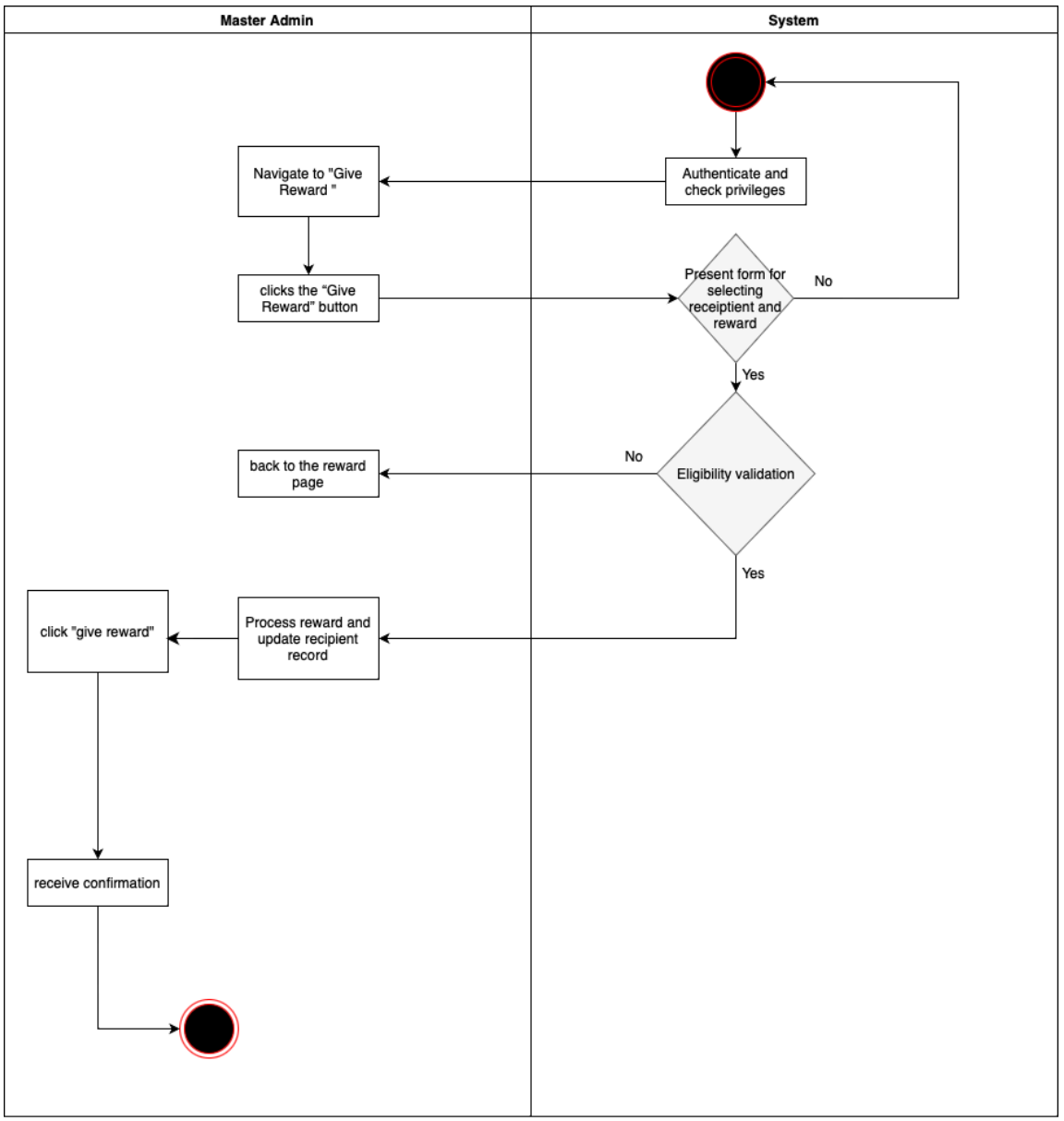


Figure 2.1.20.2: Activity Diagram of Give Reward

Module 008: Resume module

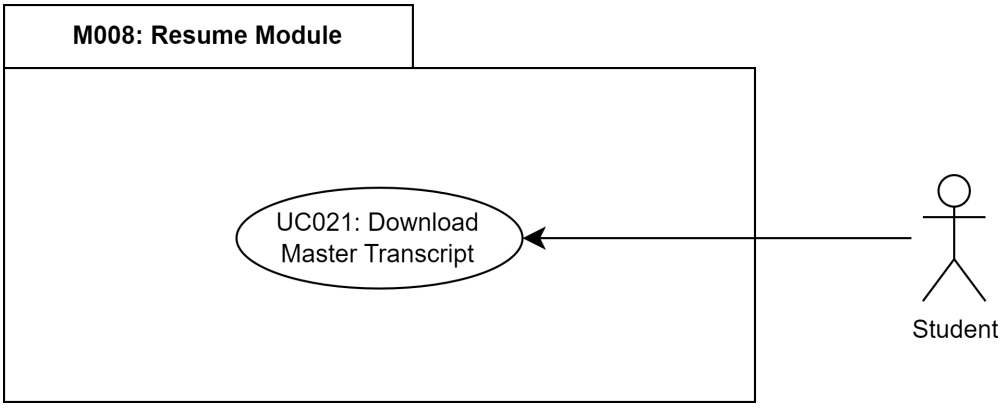


Figure 2.1 (h): Use Case Diagram for Resume module

The use case includes in this create event module is stated as below:

I. UC021: Download Master Transcript

2.1.21. UC021: Use Case <Download Master Transcript>

Table 2.1.21: Use Case Description for Download Master Transcript

Use case: Download Master Transcript
ID: UC021
Actors: 1. Student 2. Lecturer 3. Master Administrator
Preconditions: 1. The User is logged into the system. 2. The User has an existing account in the system.
Flow of events: 1. If the user is Student a. User navigates to the profile section. b. System displays the profile information of the user along with the options available within the interface.

c. Student selects the "Download Resume" option from the interface.
d. System retrieves all profile information of the user from the database.
e. System displays the preview of the resume of the user to be downloaded.
f. User selects the “Download” button to download the master transcript.
g. System prompts the user to confirm the download.
    i. If the user cancels the download action, the system backs to the profile section interface.
h. System prompts the user about the download being successful.

Postconditions:

- The requested master transcript is successfully downloaded to the device.

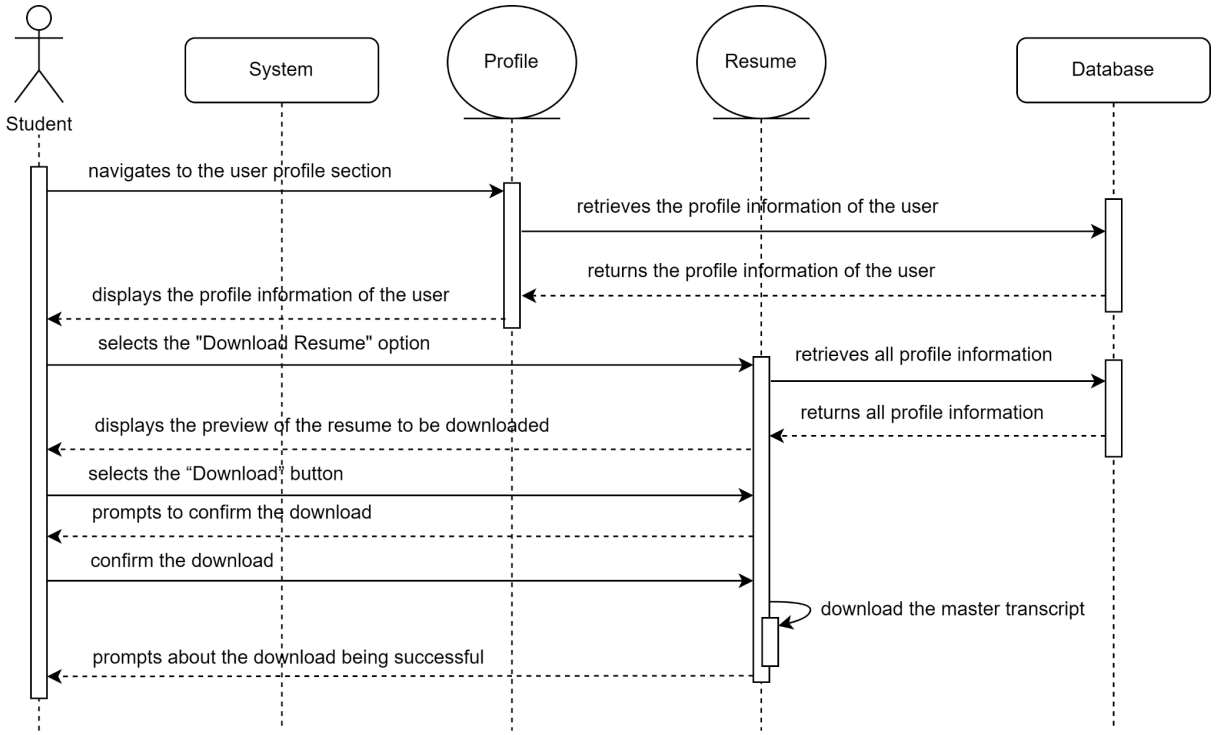


Figure 2.1.21.1: Sequence Diagram of Download Master Transcript

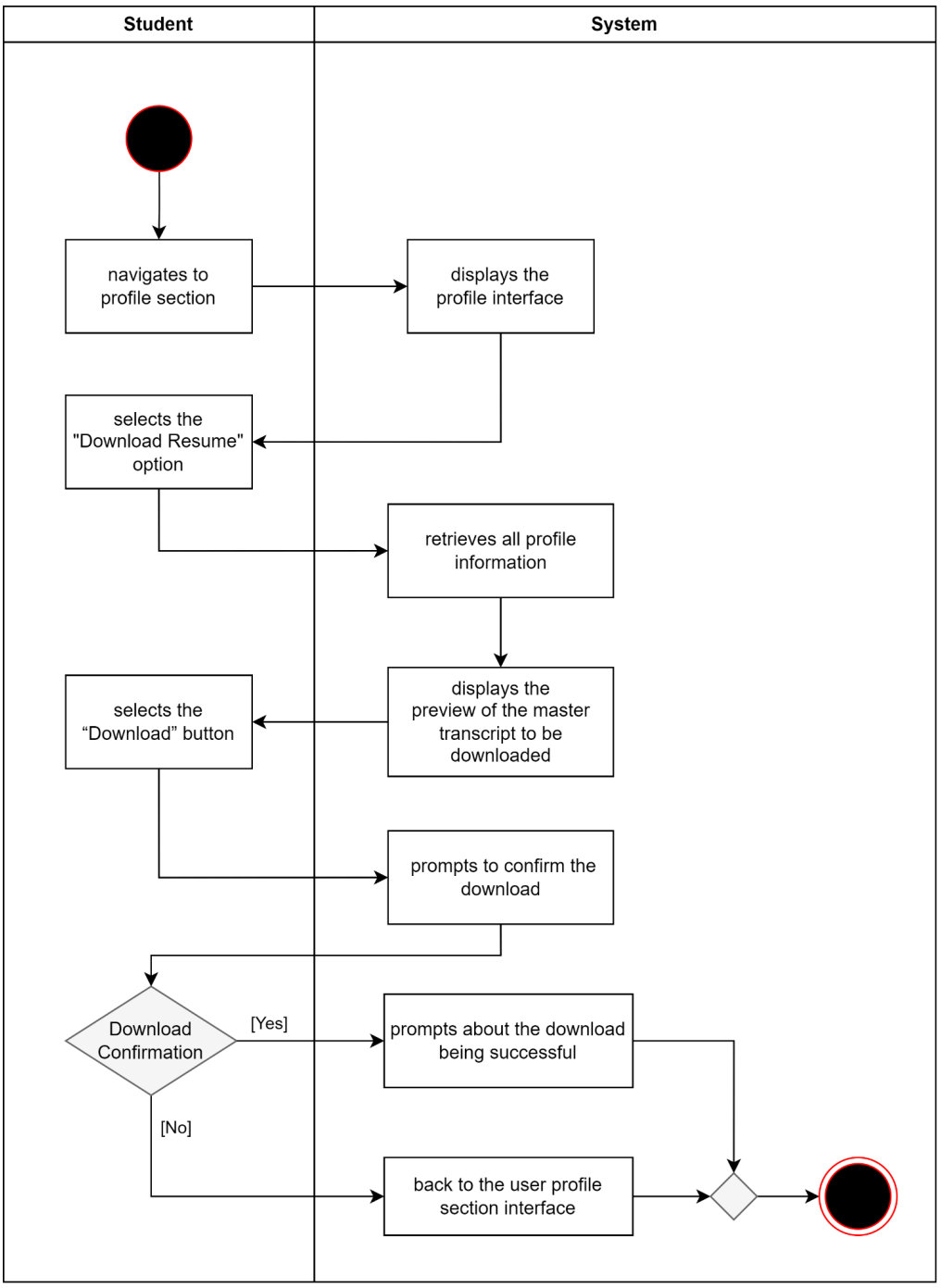


Figure 2.1.21.2: Activity Diagram of Download Master Transcript

Module 009: Personal Activity module

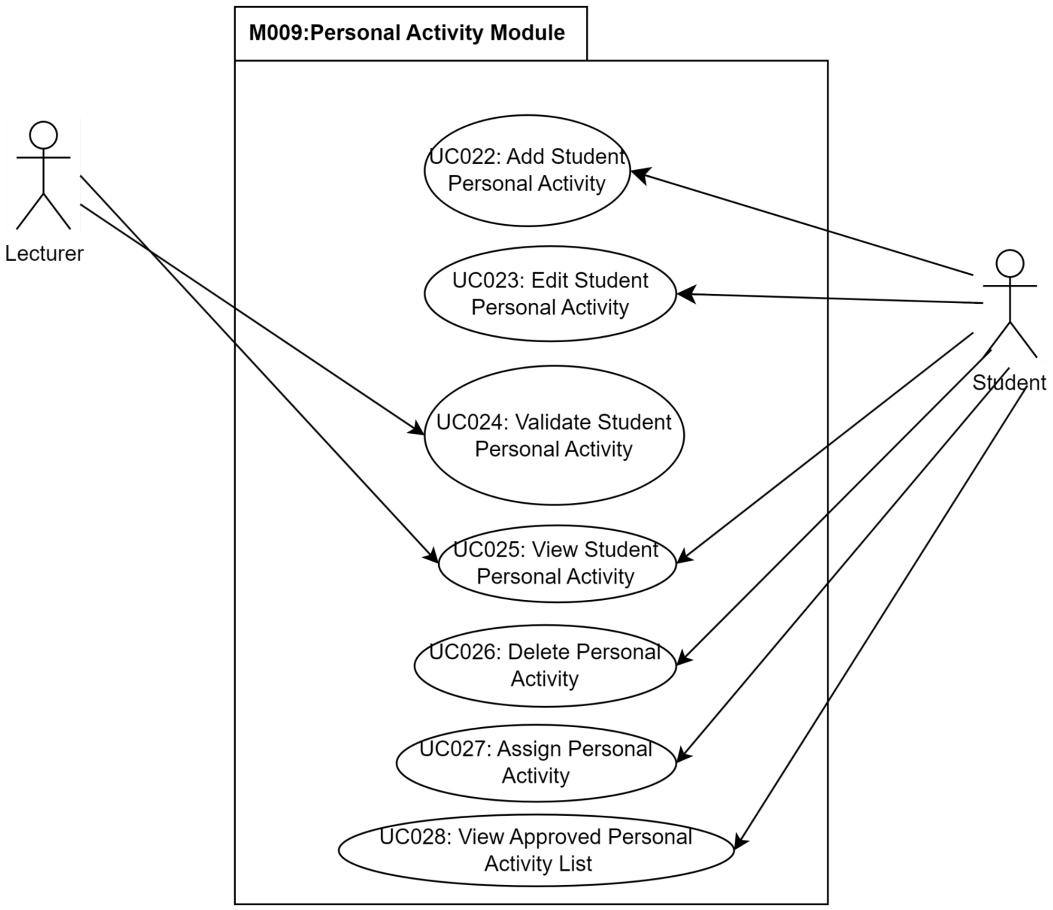


Figure 2.1 (d): Use Case Diagram for Personal Activity module

The use case includes in this create event module is stated as below:

I. UC022: Add Student Personal Activity
II. UC023: Edit Student Personal Activity
III. UC024: Validate Student Personal Activity
IV. UC025: View Student Personal Activity
V. UC026: Delete Personal Activity
VI. UC027: Assign Personal Activity
VII. UC028: View Approved Personal Activity List

2.1.22. UC022: <Add Student Personal Activity>

Table 2.1.22: Use Case Description for Add Student Personal Activity

Use case: Add Student Personal Activity
ID: UC022
Actors: 1. Student
Preconditions: ● Actor is logged into the system..
Flow of events: 1. Student navigates to the profile section on the dashboard. 2. Student scrolls down to the personal activity part. 3. Student selects and clicks the "Create" button option. 4. The Student provides details such as Event Title, Name of Event Organizers, Date, Duration, Location, Mode, Event description and evidence 5. Student clicks “Submit” button.
Postconditions: ● The personal activity is created and visible in the system.
Exception flow: ● If required information is missing, the system prompts the actor to provide the necessary details.

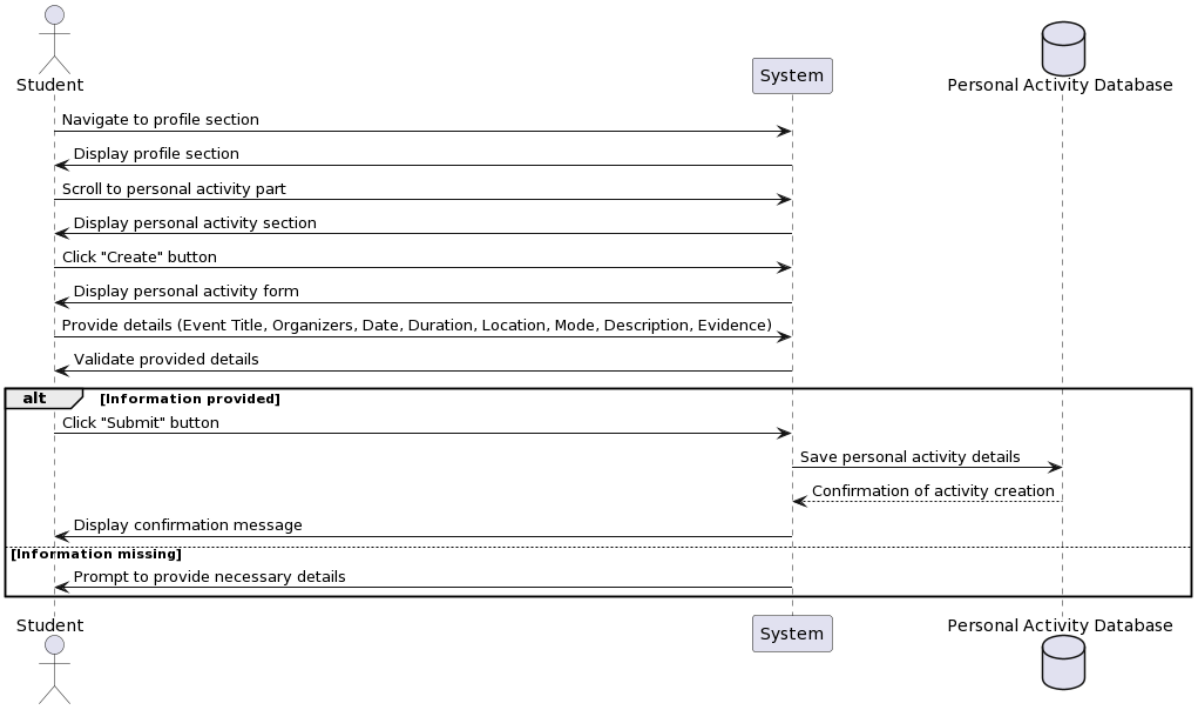


Figure 2.1.22.1: Sequence Diagram of Add Student Personal Activity

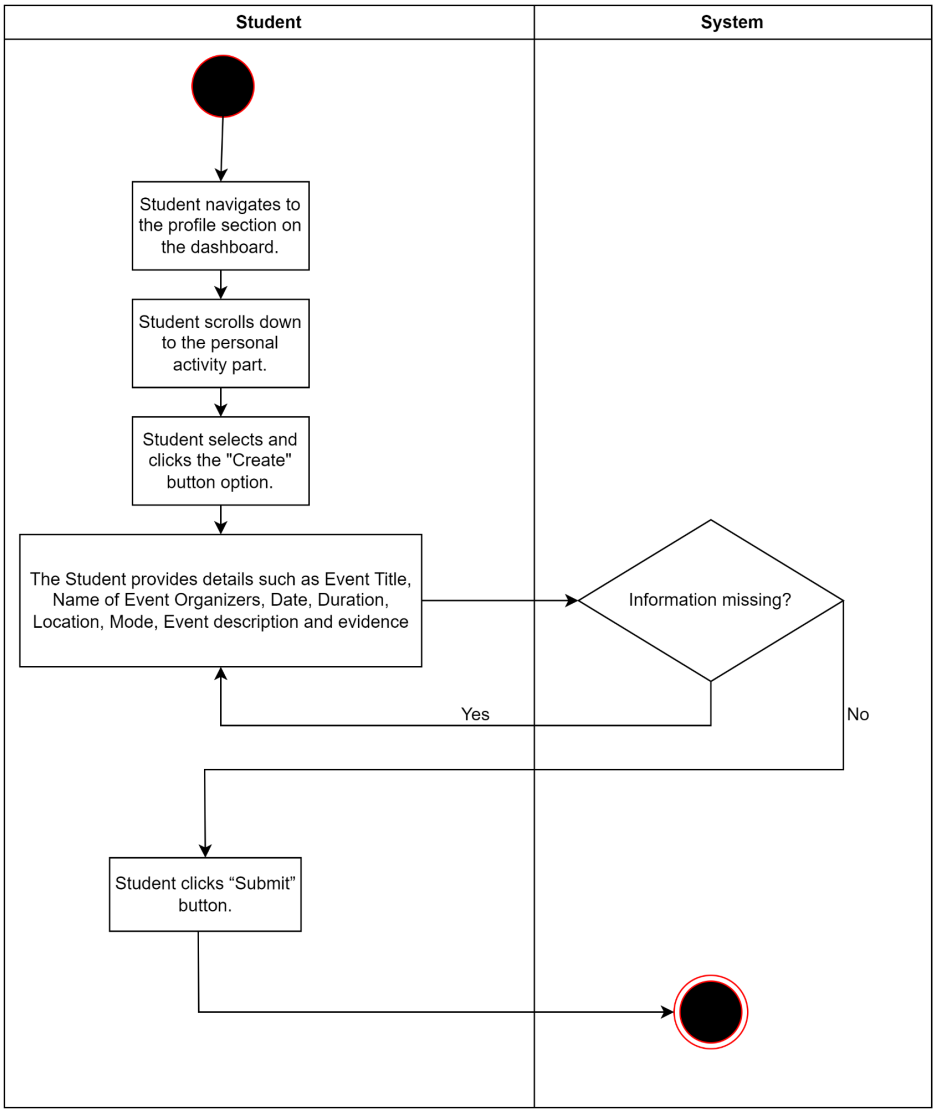


Figure 2.1.22.2:Activity Diagram of Add Student Personal Activity

2.1.23. UC023: <Edit Student Personal Activity>

Table 2.1.23: Use Case Description for Edit Student Personal Activity

Use case: Edit Student Personal Activity
ID: UC023
Actors: 1. Student
Preconditions: ● Actor is logged into the system..
Flow of events: 1. User to the personal activity section on the dashboard. 2. The system will display a list of personal activities that are already created. 3. Personal selects and clicks the activity that he or she wants to edit. 4. The system will display all the information for the relevant activity. 5. Student clicks the "Update" button option. 6. Student scrolls to the parts that he or she wants to edit. 7. Student makes necessary changes to the activity details. 8. Student clicks the "Submit" button.
Postconditions: ● The personal activity is updated and visible in the system.
Exception flow: ● If no personal activity is created The system notifies the actors.

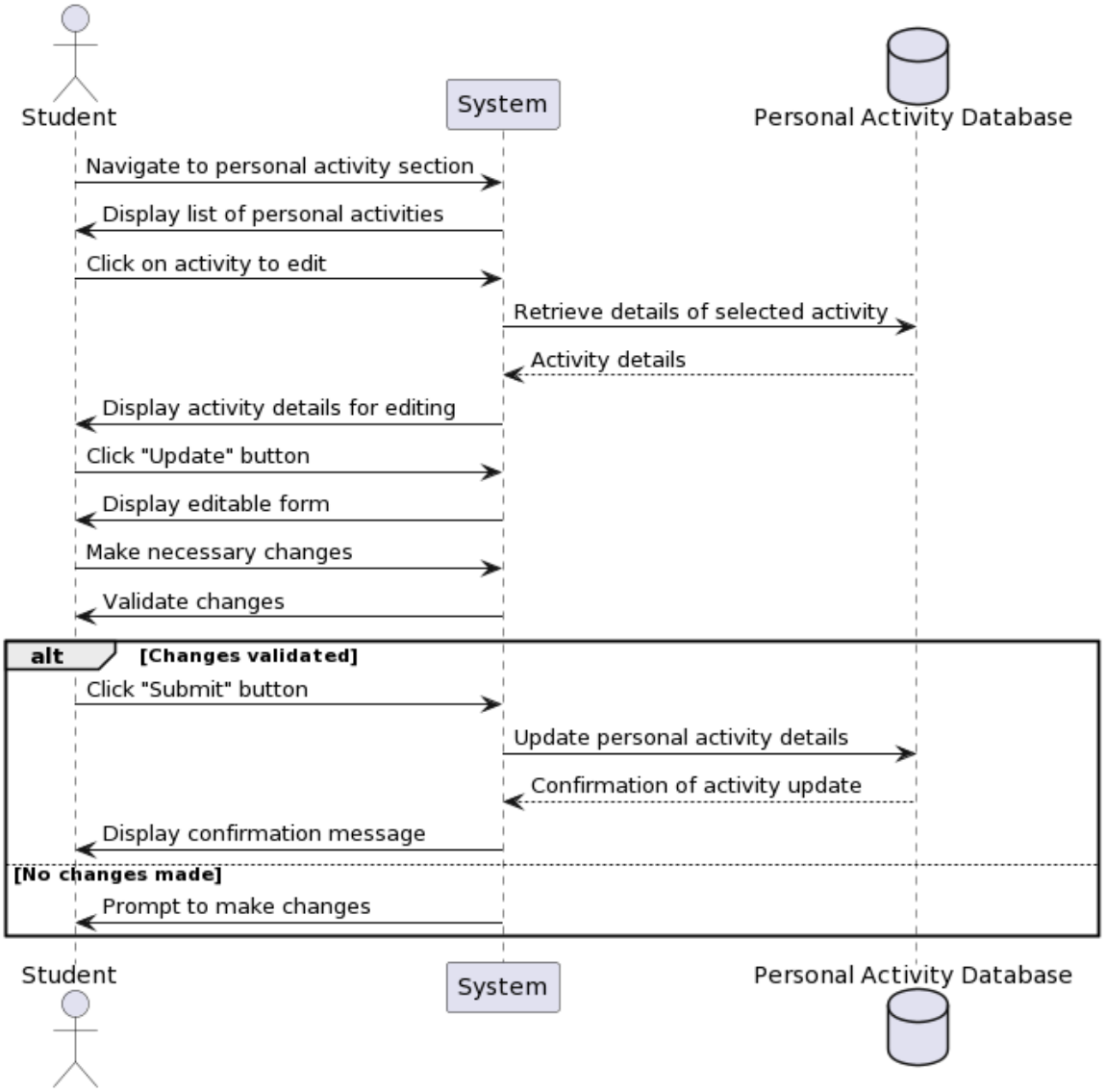


Figure 2.1.23.1: Sequence Diagram of Edit Student Personal Activity

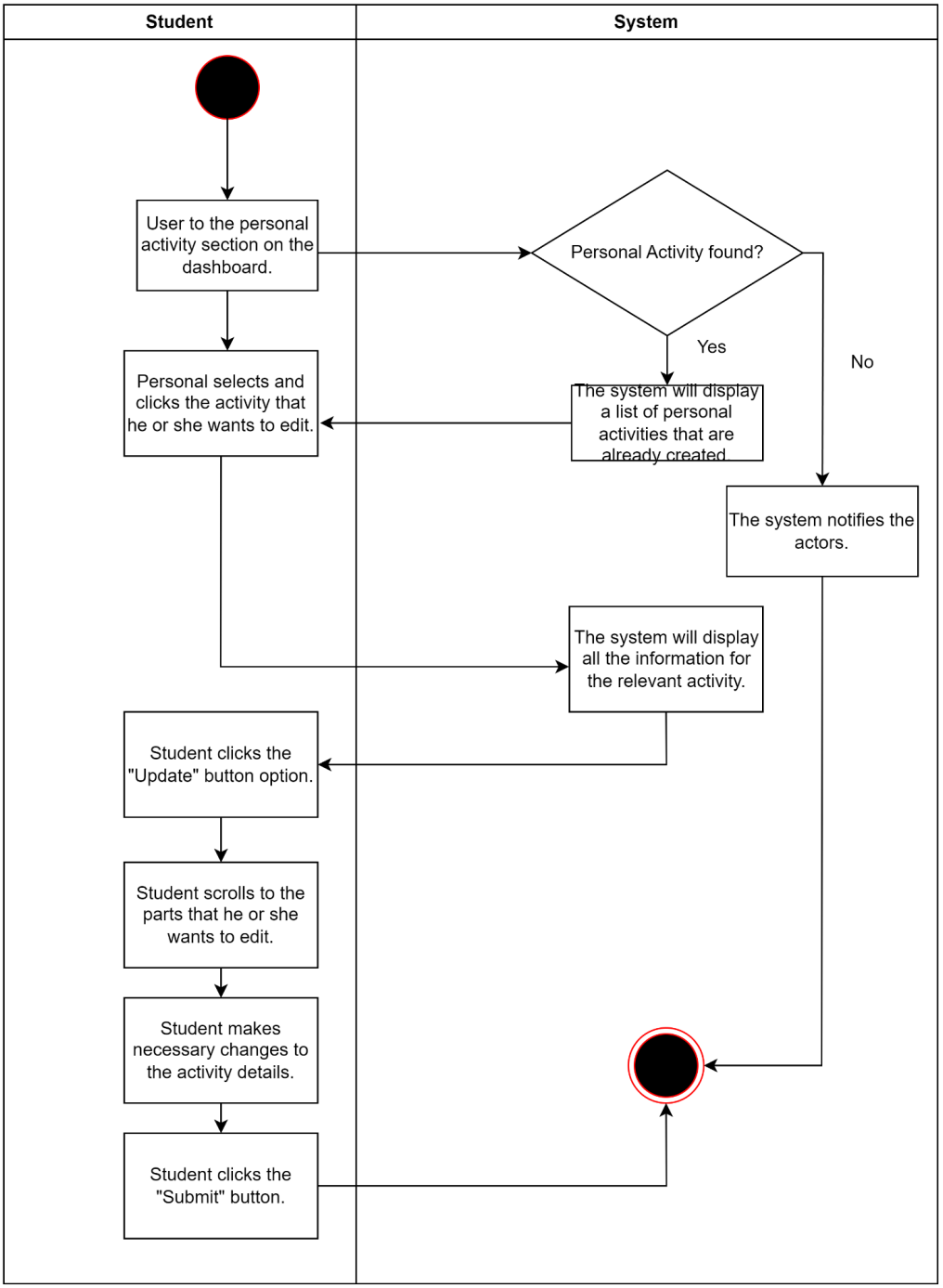


Figure 2.1.23.2:Activity Diagram of Edit Student Personal Activity

2.1.24. UC024: <Validate Student Personal Activity>

Table 2.1.24: Use Case Description for Validate Student Personal Activity

Use case: Validate Student Personal Activity
ID: UC024
Actors: 1. Lecturer
Preconditions: ● Actor is logged into the system..
Flow of events: 1. Lecturers navigate to the "Manage Personal Activities" page. 2. They view the details of the assigned personal activities listed on the page. 3. For each activity, lecturers assess the evidence provided. 4. Lecturers have the option to either "Approve" or "Reject" the activity. 5. If they choose to "Approve," they click the "Approve" button. 6. The system updates the status of the activity to "Approved." 7. If they choose to "Reject," they click the "Reject" button. 8. The system updates the status of the activity to "Rejected." 9. Lecturers continue assessing and approving/rejecting assigned personal activities as needed.
Postconditions: ● The personal activity is validated and visible in the approved personal activity.
Exception flow: ● If no personal activity is assigned the system notifies the actors.

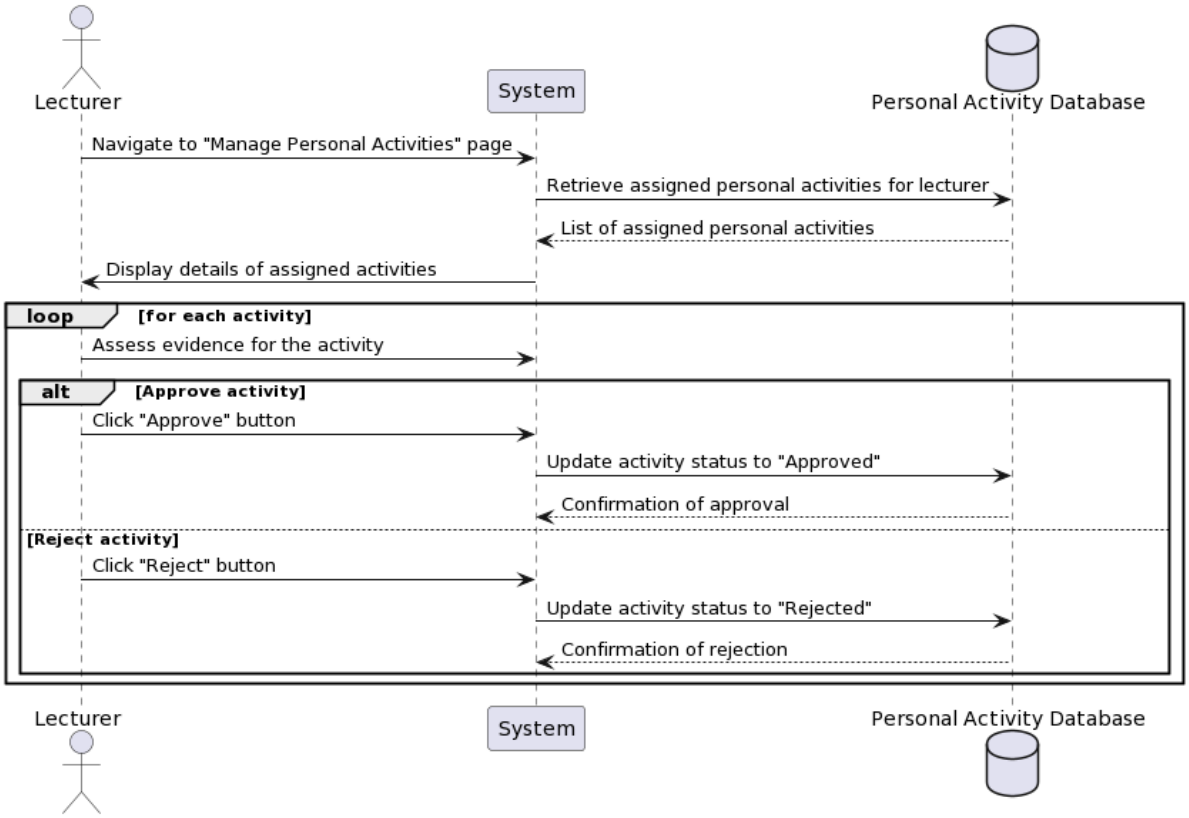


Figure 2.1.24.1: Sequence Diagram of Validate Student Personal Activity

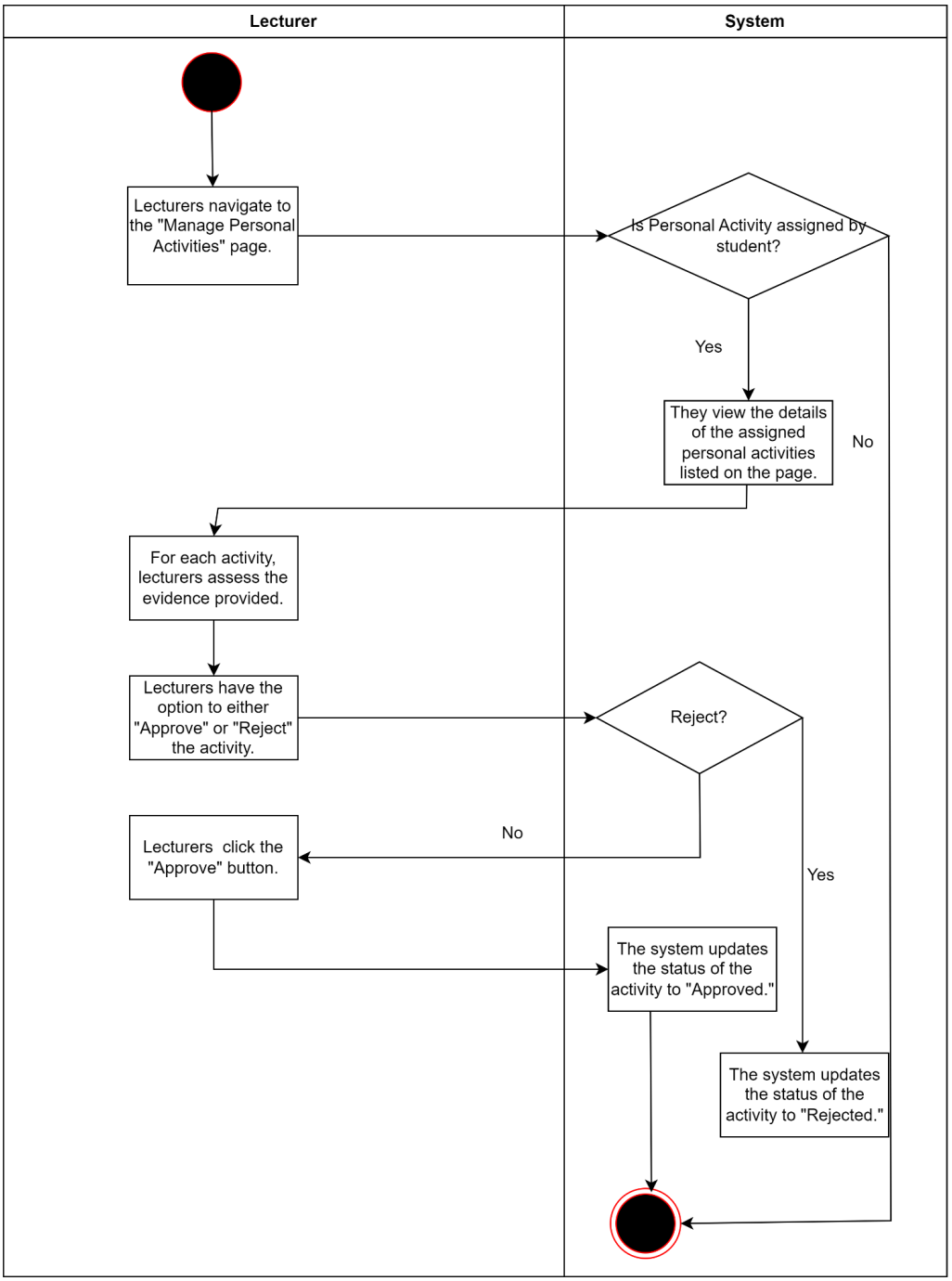


Figure 2.1.24.2:Activity Diagram of Validate Student Personal Activity

2.1.25. UC025: Use Case <View Student Personal Activity>

Table 2.1.25: Use Case Description for View Student Personal Activity

Use case: View Student Personal Activity
ID: UC025
Actors: 1. Student 2. Lecturer
Preconditions: ● Actors are logged into the system..
Flow of events: 1. Actor selects the "Manage Student Personal Activity" option. 2. Actor views the details of the student personal activities.
Postconditions: ● The personal activity is viewed.
Exception flow: ● If no personal activity is empty the system notifies the actors.

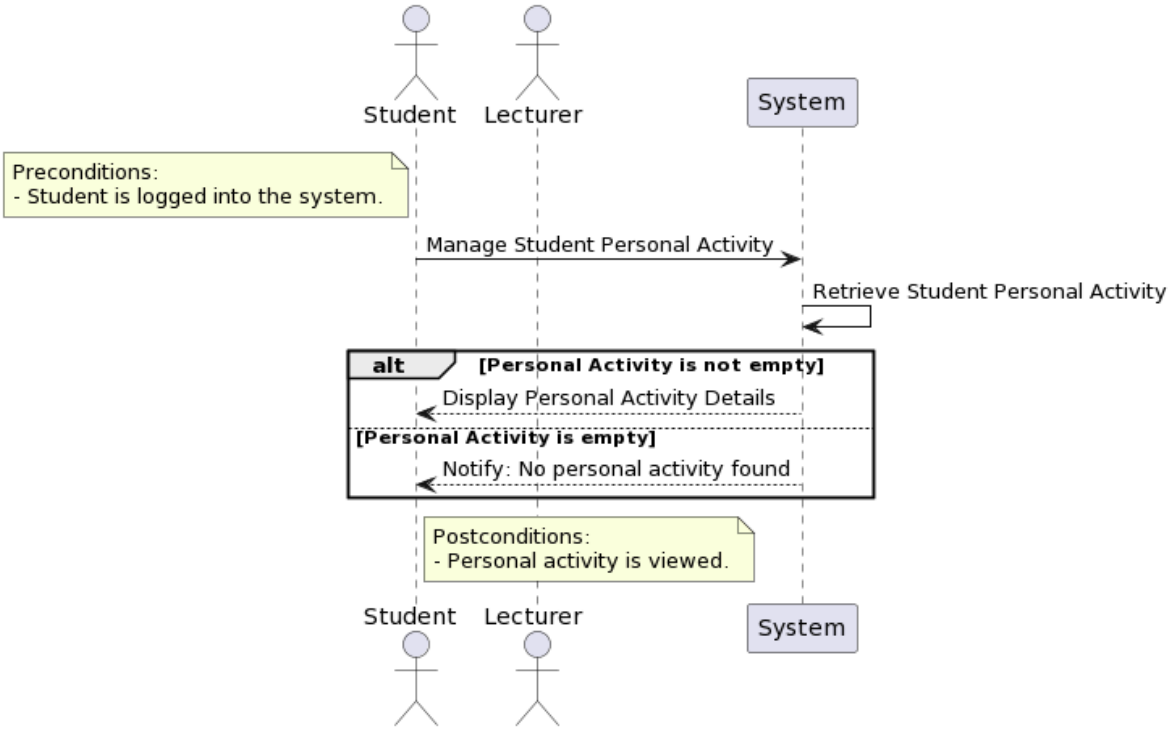


Figure 2.1.25.1: Sequence Diagram of View Student Personal Activity

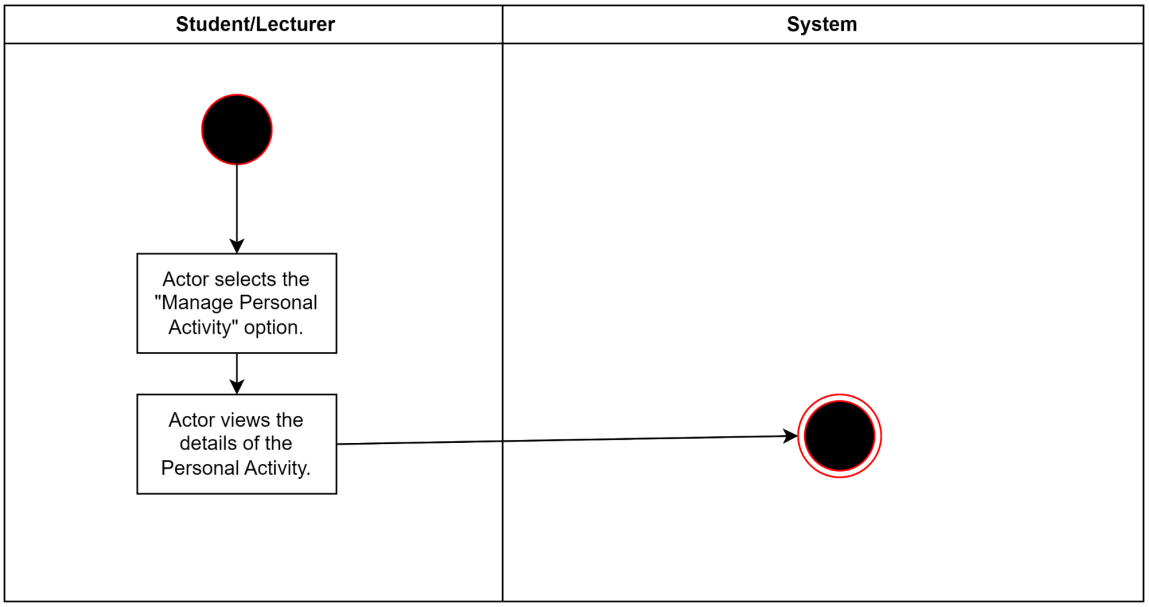


Figure 2.1.25.2:Activity Diagram of View Student Personal Activity

2.1.26. UC026: Use Case <Delete Personal Activity>

Table 2.1.26: Use Case Description for Delete Personal Activity

Use case: Delete Personal Activity
ID: UC026
Actors: 1. Student
Preconditions: ● Actor is logged into the system..
Flow of events: 1. Student navigates to the "Manage Personal Activities" page. 2. Student selects the personal activity they wish to delete. 3. Student clicks on the "Delete" button. 4. System displays a confirmation message asking "Are you sure you want to delete this activity?" 5. If the Student confirms deletion: 5.1 System removes the selected personal activity from the list. 6. System displays a message confirming the deletion.
Postconditions: ● Personal activity deleted
Alternative flow 1: 1. If the Student clicks "Close" or cancels the deletion: 2. System closes the confirmation message without deleting the activity.
Postconditions: 1. Personal activity not deleted
Exception flow: ● If no personal activity is empty the system notifies the actors.

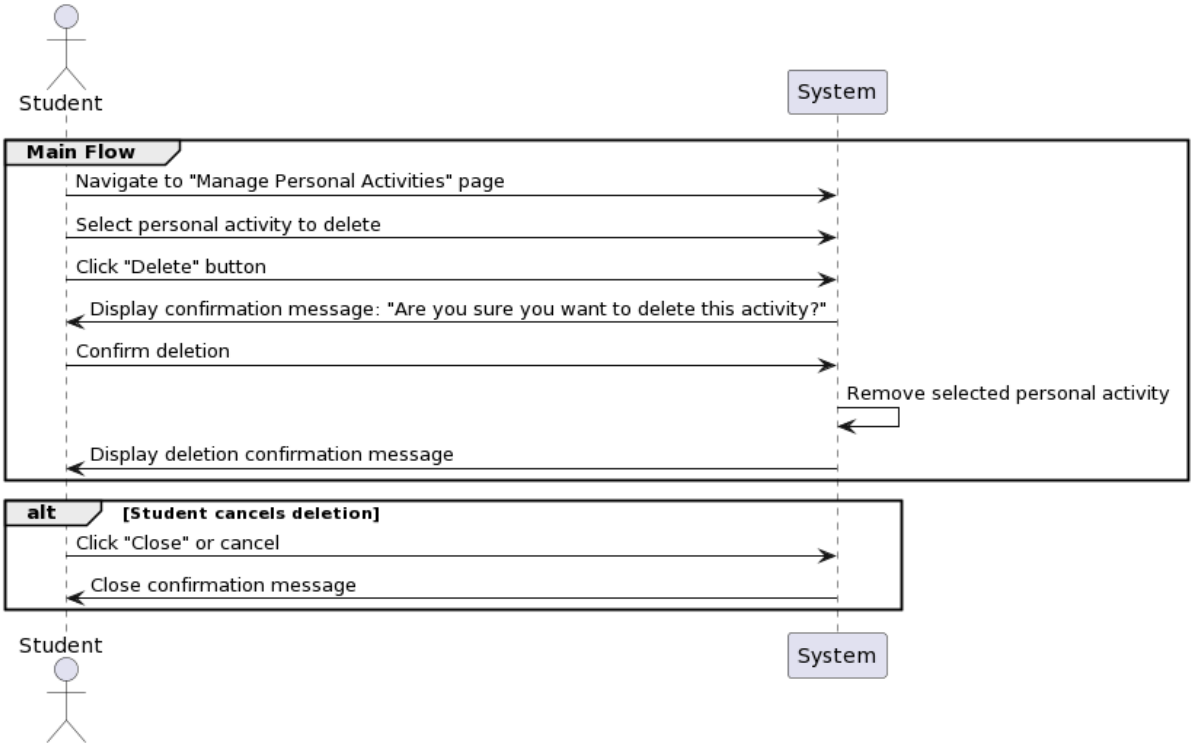


Figure 2.1.26.1: Sequence Diagram of Delete Personal Activity

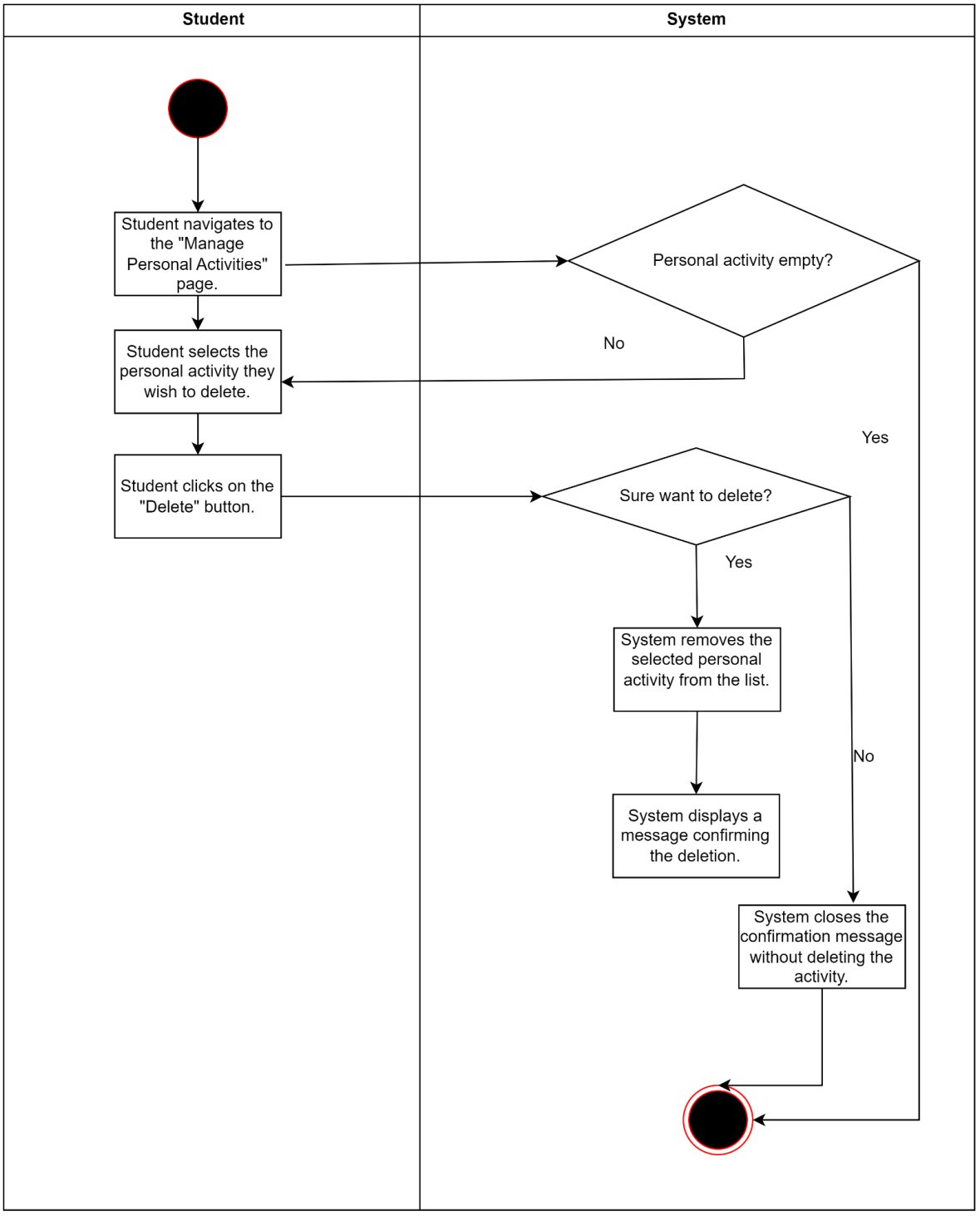


Figure 2.1.26.2:Activity Diagram of Delete Personal Activity

2.1.27. UC027: Use Case <Assign Personal Activity>

Table 2.1.27: Use Case Description for Assign Personal Activity

Use case: Assign Personal Activity
ID: UC027
Actors: 1. Student
Preconditions: ● Actor is logged into the system..
Flow of events: 1. Student navigates to the "Manage Personal Activities" page. 2. Student clicks the "Assign" button in the Action column for the desired personal activity. 3. System displays a list of available lecturers. 4. Student selects the desired lecturer from the list. 5. System updates the 'Lecturer' column in the table with the selected lecturer's name. 6. System sends the details of the personal activity to the assigned lecturer for validation and approval.
Postconditions: ● Personal activity assigned to the selected lecturer
Exception flow: ● If no personal activity is empty the system notifies the actors.

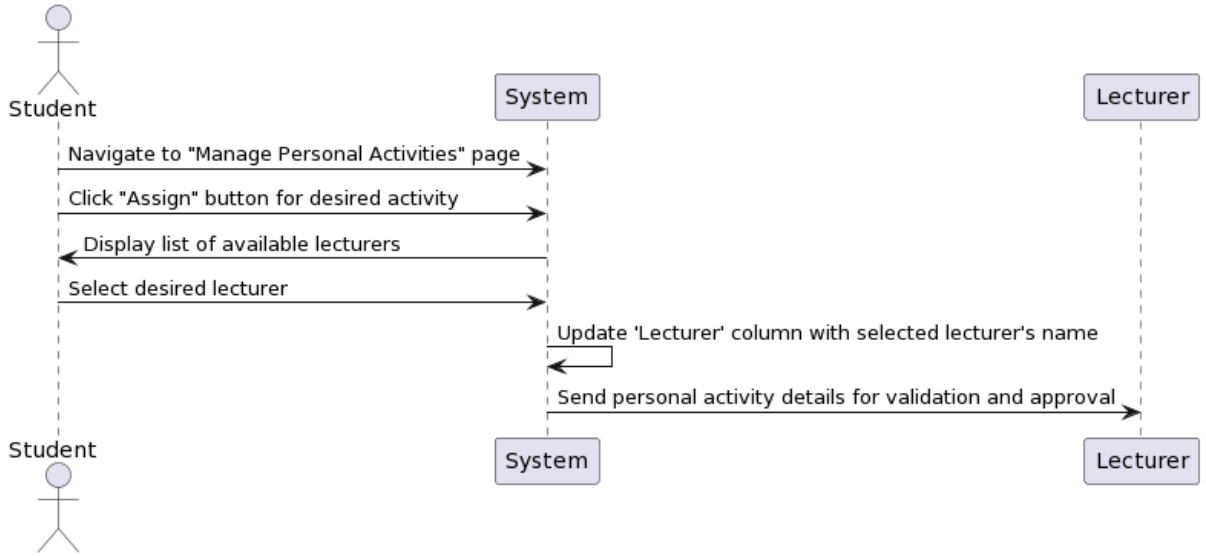


Figure 2.1.27.1: Sequence Diagram of Assign Personal Activity

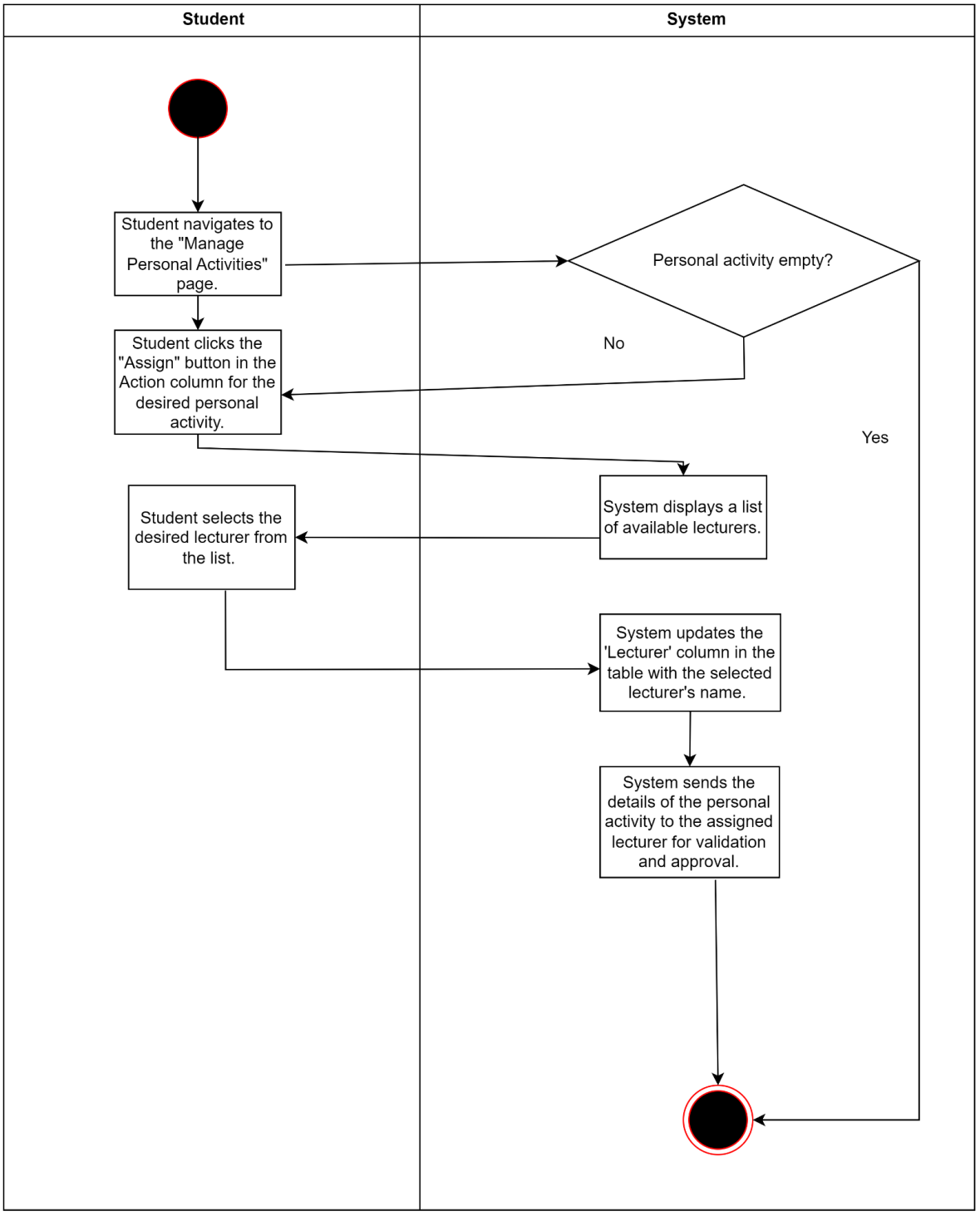


Figure 2.1.27.2:Activity Diagram of Assign Personal Activity

2.1.28. UC028: Use Case <View Approved Personal Activity List>

Table 2.1.28: Use Case Description for View Approved Personal Activity List

Use case: View Approved Personal Activity List
ID: UC028
Actors: 1. Student
Preconditions: ● Actor is logged into the system..
Flow of events: 1. Student accesses the dedicated page for viewing approved personal activities. 2. System displays a list of approved personal activities, including details such as Personal Activity Name, Description, Date, Venue, Evidence, and a 'Status' column indicating "Approved."
Postconditions: ● Approved personal activities are viewed.
Exception flow: ● If no approved personal activities are found, the system notifies the actor.

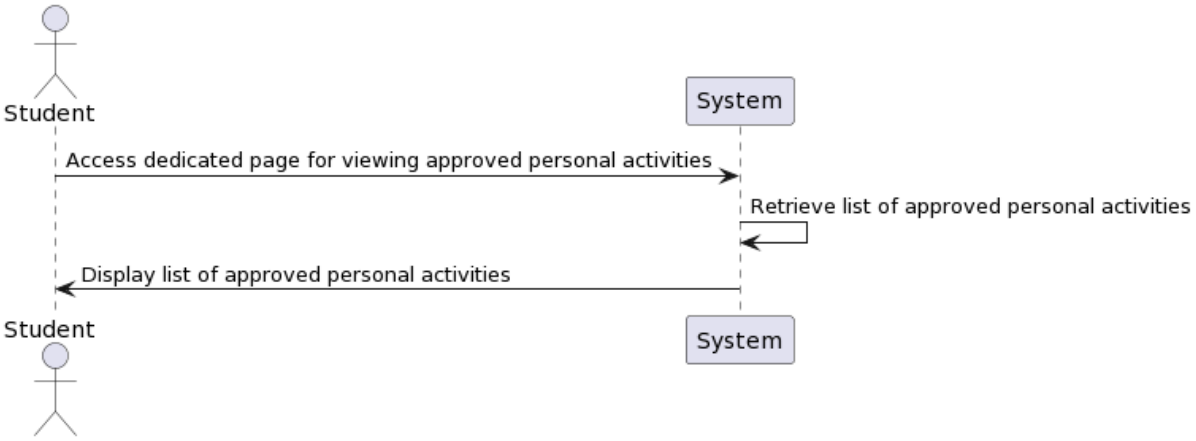


Figure 2.1.28.1: Sequence Diagram of Assign Personal Activity

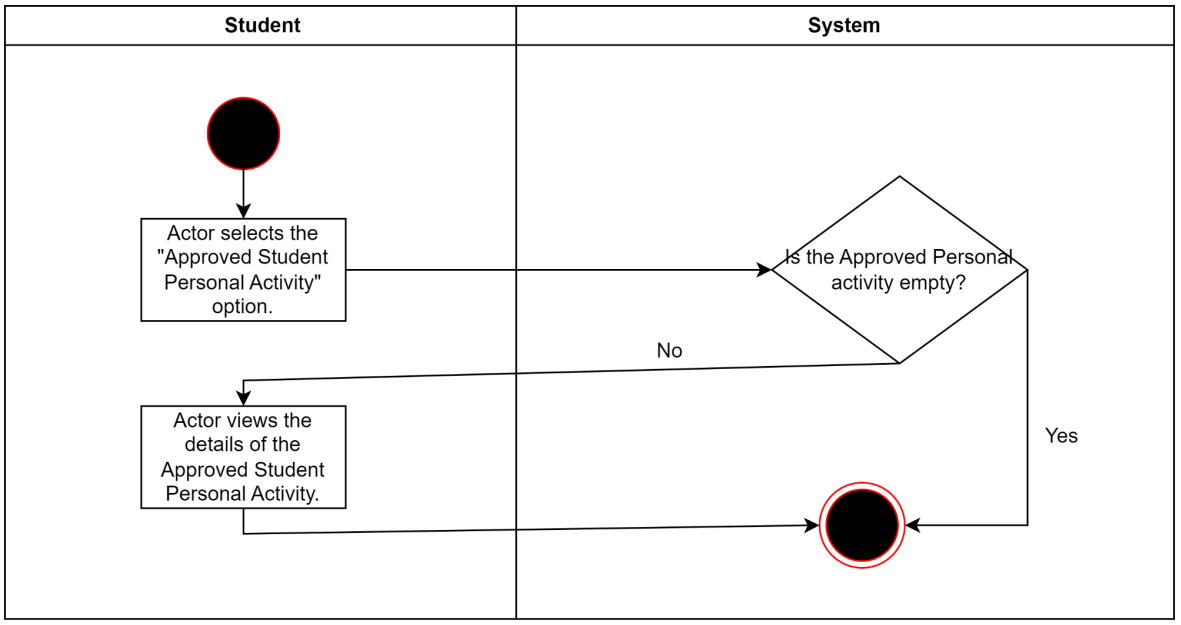


Figure 2.1.28.2:Activity Diagram of Assign Personal Activity

2.2. Performance and Other Requirements

1. **POR1 Response Time**
    - The system should provide an ability of the system to load fast and respond immediately for common user operations with an user request or multiple users requests at a time.
2. **POR2 Data Processing Speed**
    - Data processing operations, such as importing or exporting large sets of student data including activity and feedback data, should be optimized to complete within acceptable timeframes.
3. **POR3 Capacity**
    - The system must be capable of supporting the maximum number of users or the amount of data that the system can handle while maintaining its optimal performance and scalability.
4. **POR4 Safety**
    - The system should be able to create a secure and safe environment for the users by preventing or mitigating potential hazards, virus infection or risks.
5. **POR5 Scalability**
    - The system architecture should be designed to scale horizontally to accommodate a growing number of users and portfolios without a proportional decrease in performance.
6. **POR6 Availability**
    - The system is accessible 24/7 and operational for users whenever they need to interact with it.
7. **POR7 Maintainability**
    - The system should be designed with clear and comprehensive documentation, enabling developers to understand and maintain the codebase effectively. Regular updates and bug fixes should be manageable without significant disruptions.

2.3. Design Constraints

1. **DC1 Technology Stack Constraints**
    - The system must be developed using the organization's approved technology stack, which includes specific programming languages, frameworks, and databases
2. **DC2 Security Constraints**
    - The system must be designed by implementing specific security measures, adhering to data protection guidelines, and ensuring the confidentiality and integrity of user data. Only the authentication users have access to log in to this system.
3. **DC3 User Access Controls Constraints**
    - Design constraints may specify specific user roles and permissions, outlining who can access, modify, or delete student portfolio data.
4. **DC4 Scalability Constraints**
    - The design should account for scalability requirements specified by the organization, allowing the system to scale horizontally or vertically to accommodate increased data and user loads.
5. **DC5 Usability Guidelines Constraints**
    - The system's user interface design should align with the organization's usability guidelines and standards to provide a consistent and user-friendly experience.
6. **DC6 Budgetary Constraints**
    - The design and development of the system must adhere to budgetary constraints specified by the organization, including cost limitations for hardware, software, and ongoing maintenance.
7. **DC7 Cultural and Language Considerations**
    - If the organization operates in a multicultural environment, the system design should consider cultural and language preferences, including support for multiple languages.

2.4. Software System Attributes

1. **SSA1 Usability**
    - The system should be user-friendly, and easy to navigate, with features such as tooltips, contextual help, and error messages to assist users.
    - The system should be able to notify the admin and master admin when any registration, or feedback is received and the student when any review or edit is made.
    - Based on users' request, the interface needs to be attractive and easy to use, with clear menus, navigation paths, and aesthetically pleasing design elements.
2. **SSA2 Responsiveness**
    - The system should provide a responsive user experience, ensuring that actions and commands are executed promptly.
    - Based on users' request, the system responds quickly to user inputs, providing real-time feedback for operations such as data entry and updates.
3. **SSA3 Security**
    - The system must implement robust security measures to protect sensitive student data.
    - Based on users' request, the system ensures data confidentiality, integrity, and availability, with features such as role-based access control and encryption.
4. **SSA4 Collaboration**
    - The system should support collaboration among users, allowing for shared access and editing of portfolio information.
    - Based on users' request, features that facilitate collaborative work, such as comments, discussion threads, and version history.
5. **SSA5 Scalability**
    - The system should be scalable to accommodate the growth of user data and increasing numbers of portfolios.
    - Based on users' request, a system can scale seamlessly as the user base and data volume expand.
6. **SSA6 Reliability**
    - The system must be reliable, minimizing downtime and ensuring that users can access their portfolios without interruption.

- Users prioritize a reliable system that is available whenever they need to use it.

7. SSA7 Portability

- The system should be platform-agnostic, supporting access through commonly used web browsers and ensuring a consistent user experience across different devices such as desktops, tablets, and mobile phones.

Appendices

1. Higher Definition Overview UseCase Diagram:

   SRS-Updated Overview UseCase.jpg