SECJ2203: Software Engineering

System Documentation (SD)
(STD)
Final Year Project (FYPi) Management System
<GradX>

Version 3.0

July 2023

Faculty of Computing

Prepared by: Assassins

Revision Page

a. **Overview**

The "GradX" software system is designed to serve as an online platform for managing and recording student Final Year Project (FYPi) projects in the Data Engineering program at the Faculty of Computing, Universiti Teknologi Malaysia. The purpose of this system is to streamline the workflow of FYPi projects, simplify the documentation process, and establish a standardized evaluation system using rubric formats. This system aims to provide students with the ability to track their progress, submit project proposals, design documentation, and final reports. It also enables industry and university coaches to assess and provide valuable feedback on students' work. The external interfaces, including user interfaces, hardware interfaces, software interfaces, and communication interfaces, will be detailed in the system documentation's particular needs section. It will also explain system characteristics using use case diagrams, domain models, and state machine diagrams. This part will also include project group members' discussions about performance needs, design constraints, software system properties, and other project requirements.

b. **Target Audience**

Target audience for this system are final year Data Engineering students, University Coaches, Industry Coaches and Coordinator.

c. Project Team Members

List the team members in a table by stating their roles and the status for each assigned task e.g. by sections for this SD version (complete, partially complete, incomplete). If the assigned tasks are not done and have been assigned to other team members, state accordingly.

Member Name	Role	Task	Status
Malleylene Peneh (A21EC0052)	Team Member	revision page, references, use case: design project proposal, assign UC and examiner, calculate student mark, software system attributes	complete
Farah Nabilah Binti Najmudin (A21EC0023)	Team Member	introduction, user characteristic,domain model,use case: login, checking project progress, provide feedback	complete
Puteri Nur Eleeya Syafika Binti Mohd Zabidi (A21EC0124)	Team Member	system features, use case diagram, use case: evaluate student work,prepare necessary form	complete
Siti Nurkamilah Binti Saiful Bahari (A21EC0131)	Leader	system features, use case diagram, use case: submit project proposal, submit project solution, provide rubrics, design constraints, references	complete

d. Version Control History

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Malleylene Peneh	Section 1 - 2 of SRS	22/5/2023
2.0	Puteri Nur Eleeya Syafika Binti Mohd Zabidi	Section 3 - 5 of SDD	16/6/2023
3.0	Siti Nurkamilah Binti Saiful Bahari	Section 6 -7 of STD	2/7/2023

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this System Documentation (SD) is to offer a detailed overview of the system development process. It encompasses the System Requirements Specification (SRS), System Design Document (SDD), and System Testing Document (STD). The SD acts as a reference document for individuals involved in the system development process, such as coordinators, students, university coaches, and industry coaches. It aims to provide a clear understanding of the system requirements and design processes, ensuring all users are aligned and working towards the same objectives. Through structured and comprehensive documentation, the SD facilitates the success of the system development process by minimizing errors, enhancing efficiency, and improving overall outcomes for the 4th year final project.

1.2 Scope

One software that can assist in a fourth-year final project is "GradX." GradX is a comprehensive project management software that provides tools and features to help you plan, organize, and execute your final project effectively. It allows you to create project timelines, set milestones, assign tasks to team members, track progress, and manage resources. Additionally, it offers collaboration features to facilitate communication and coordination among project team members. With GradX, you can streamline your project management processes and enhance the overall efficiency of your final year project.

The scope of the software product includes the following:

- Create submission for project proposal and design solution.
- To track the student progress and feedback from the university coach and industry coach
- To organize the milestones and student progress breakdown into tasks and subtasks.
- Allow the coordinator to release the official form for marking and feedback.
- Allow to monitor the progress of students and the status of each task and ensure that everything is on track.

1.3 Definitions, Acronyms and Abbreviation

Definitions of all terms, acronyms and abbreviations used are to be defined here.

Term	Definition
GradX	The software product being developed to manage the student's 4th year final project and determining the great flow of project progress during internship.
SRS	System Requirements Specification - a document that outlines the requirements and objectives of the software product being developed.
SDD	System Design Document - a document that describes the system architecture, components, and interfaces in detail.
UC	University Coach - a faculty member who will ensure that the student is meeting the necessary academic requirements and will also oversee the student's progress in completing the project proposal.
IC	Industrial Coach - a professional from the relevant field of study in their organization and will guide the student through the project.

1.4 References

1. Bowers, M. (2020). Design Constraints Are Not Restraints – They Stoke Creativity. *Toptal Design Blog*. https://www.toptal.com/designers/ui/design-constraints
2. Editor. (2019, December 9). Technical Documentation in Software Development: Types, Best Practices, and Tools. *AltexSoft*. https://www.altexsoft.com/blog/business/technical-documentation-in-software-development-types-best-practices-and-tools/
3. Lutkevich, B. (2022). software documentation. *Software Quality*. https://www.techtarget.com/searchsoftwarequality/definition/documentation
4. Admin. (2023, February 22). *The Basics of Software Quality Attributes*. Codoid. https://codoid.com/software-testing/the-basics-of-software-quality-attributes/

1.5 Overview

The System Documentation (SD) provides a comprehensive description of the GradX software product being developed. This comprises three main sections: System Requirements Specification (SRS) and System Design Document (SDD)

2. Specific Requirements

2.1 User characteristics

In this section, we will provide a brief introduction to the different user characteristics mentioned in the requirements. Each user plays a specific role within the GradX system and has distinct needs and responsibilities. These user characteristics illustrate the GradX system's many roles and responsibilities. Each user interacts with the system in a unique way, with varied levels of technical competence and various work needs.

2.1.1 Final Year Student

- The Final Year student will use the system to submit their project proposals online.
- They should have basic computer skills and familiarity with web-based applications
- They may have varying levels of technical expertise.
- They will require a medium to submit their projects online.

2.1.2 University Coach

- The University Coach will use the system to check the progress of students.
- They will use the system to evaluate students.
- They will generate reports based on student progress.
- They may require technical training on how to use the system.

2.1.3 Industry Coach

- The Industry Coach will use the system to check the progress of students.
- They will use the system to evaluate students.
- They will generate reports based on student progress.
- They may require technical training on how to use the system.

2.1.4 Examiner

- The examiner will use the system to evaluate final year projects.
- They will provide feedback

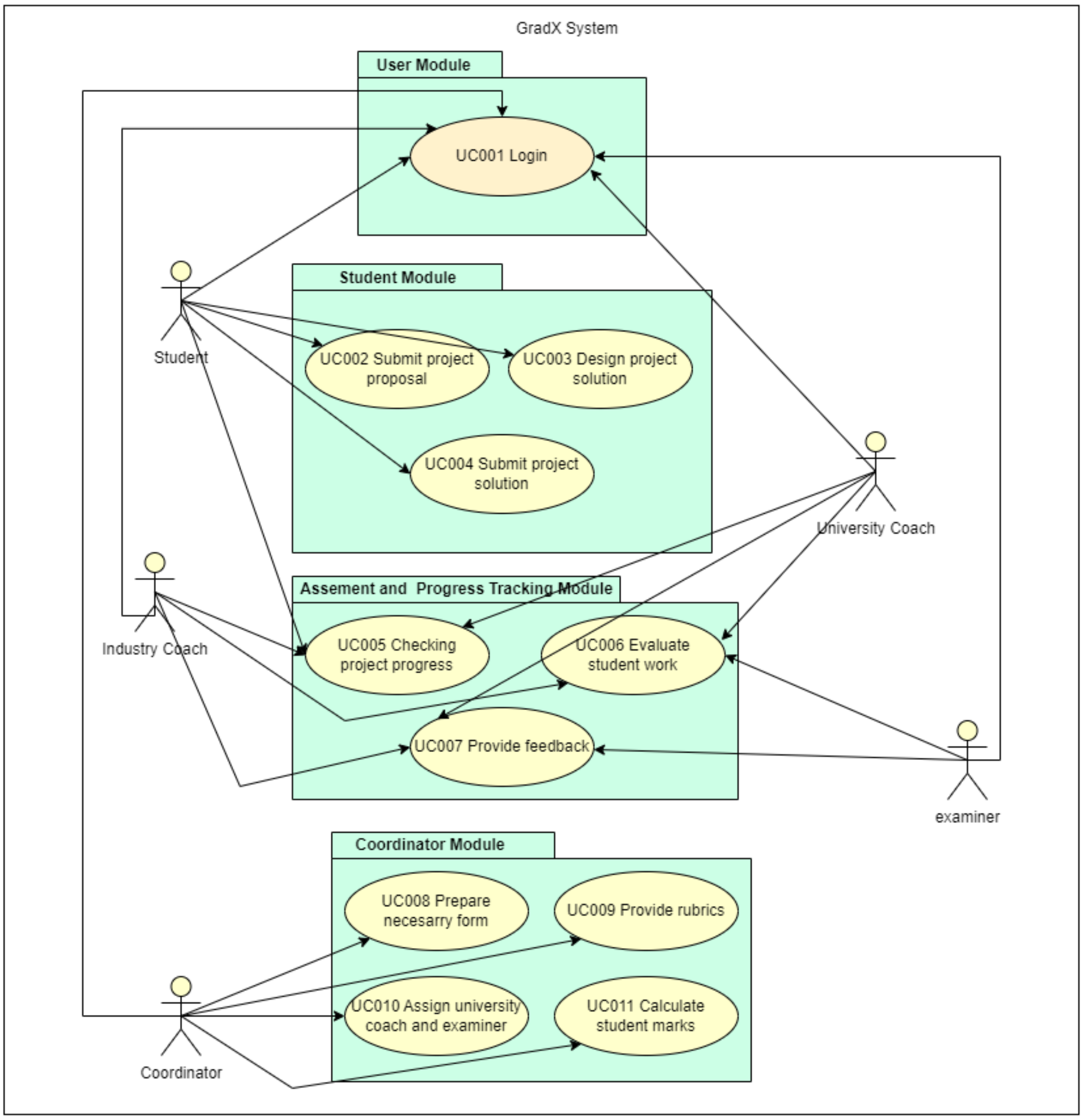
2.1.5 Coordinator

- The coordinator will prepare rubrics for evaluation.
- They will prepare evaluation forms for coaches and examiners to use.

2.2 System Features

The GradX is a software system that operates on desktop computers including windows and iOS operating systems. The system provides a means for UTM students in their 4th year to manage their Final Project. The system also implemented a real-time cloud-database for better user experiences.

The system features are illustrated in Figure 2.2.1 below. The detailed description of each module and functions is tabulated in Table 2.2.



GradX System
User Module
UC001 Login
Student Module
UC002 Submit project proposal
UC003 Design project solution
UC004 Submit project solution
Assement and Progress Tracking Module
UC005 Checking project progress
UC006 Evaluate student work
UC007 Provide feedback
Coordinator Module
UC008 Prepare necesarry form
UC009 Provide rubrics
UC010 Assign university coach and examiner
UC011 Calculate student marks
Student
Industry Coach
University Coach
examiner
Coordinator

Figure 2.2.1 Use case diagram for <GradX>

Table 2.2: Description of Module and Functions for <GradX>

Module	**Function**	**Description**
User Module	UC001 - Login	This use case allows students to login as a user for the system
Student Module	UC0002 - Project Proposal Submission	This use case allows student to submit project proposal through the system
	UC0003 -Design Project Solution	This use case includes the students designing the project solution.
	UC004 - Project Solution Submission	This use case allows student to submit project solution through the system
Assessment and Progress Tracking Module	UC005 - Checking Project Progress	This use case allows UC, IC and Coordinator to track the progress of students' work progress.
	UC006 - Evaluate Student Work	This use case allows UC, IC and examiners to evaluate the students' work.
	UC007 - Provide Feedback	This use case is to allow the examiners to review and give feedback to student work
Coordinator Module	UC008 - Prepare Necessary Form	This use case for coordinator to provide the necessary form for the UC, IC and examiners.
	UC009 - Provide Rubrics	This use case describes the activity of providing project rubrics to students and examiners involved.
	UC010 - Assign UC and Examiner	This use case allows the coordinator to assign UC and examiners for the students.
	UC0011 - Calculate Student Mark	This use case involves the calculation of student marks by the coordinator.

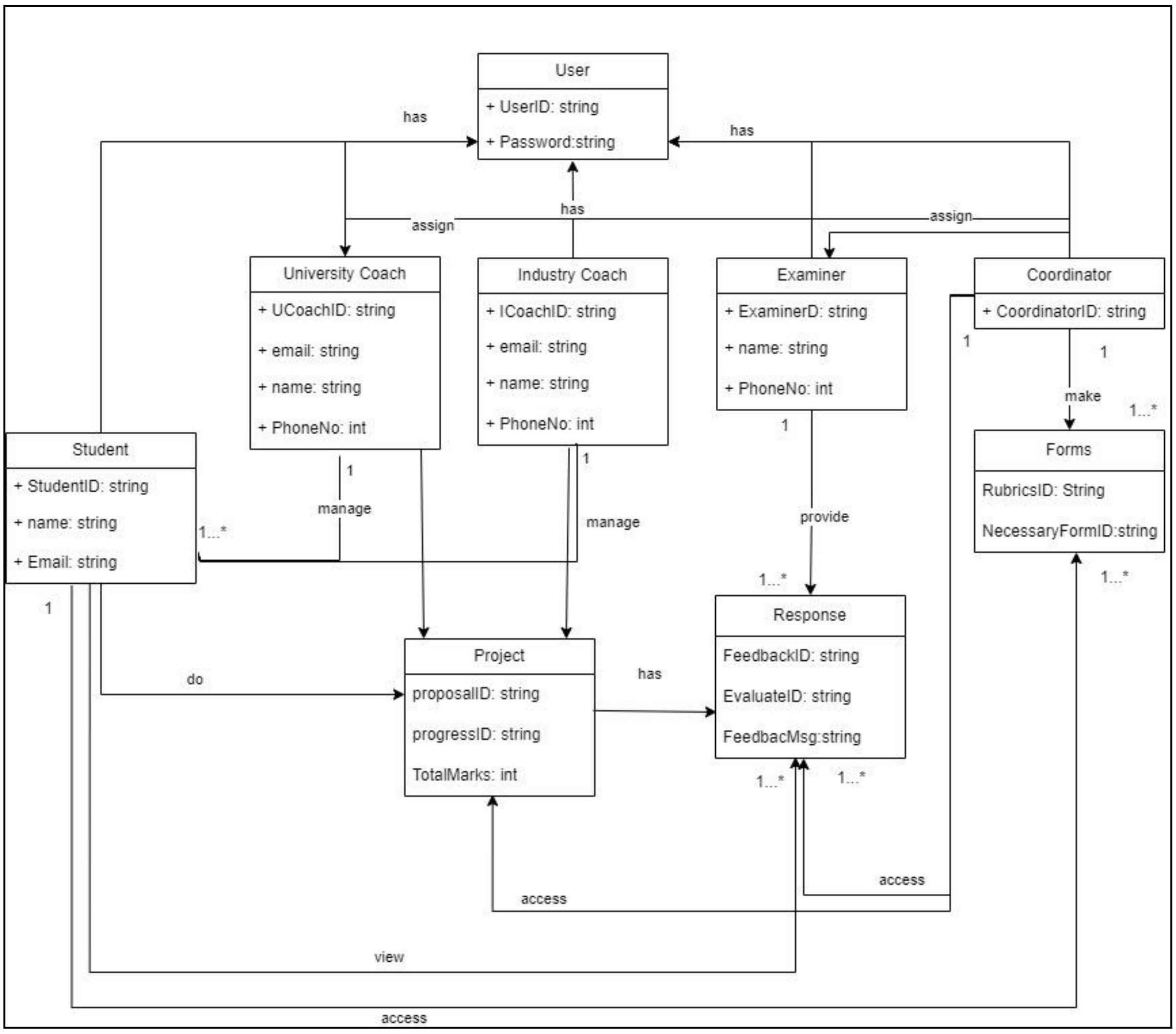


Figure 2.2.2: Domain Model for <GradX>

2.2.1 UC001: Use Case <Login>

Table 2.2.1: Use Case Description for <Login>

Use case: <Login>
ID: UC001
Actors: Student, UC,IC, Examiner,Coordinator
Preconditions: 1. Has an internet connection to access the login page. 2. Got their own login name with their password.
Flow of events: 1. Users will enter their login name and password. 2. The system will check and validate the user entered name and password. 3. System will direct the user to the homepage. 4. Login use case ends.
Postconditions: 1. Users can proceed to do their tasks in the system once directed to the homepage.
Exception flow (if any): 1. Invalid name or password. 1.1. System will display an error message.

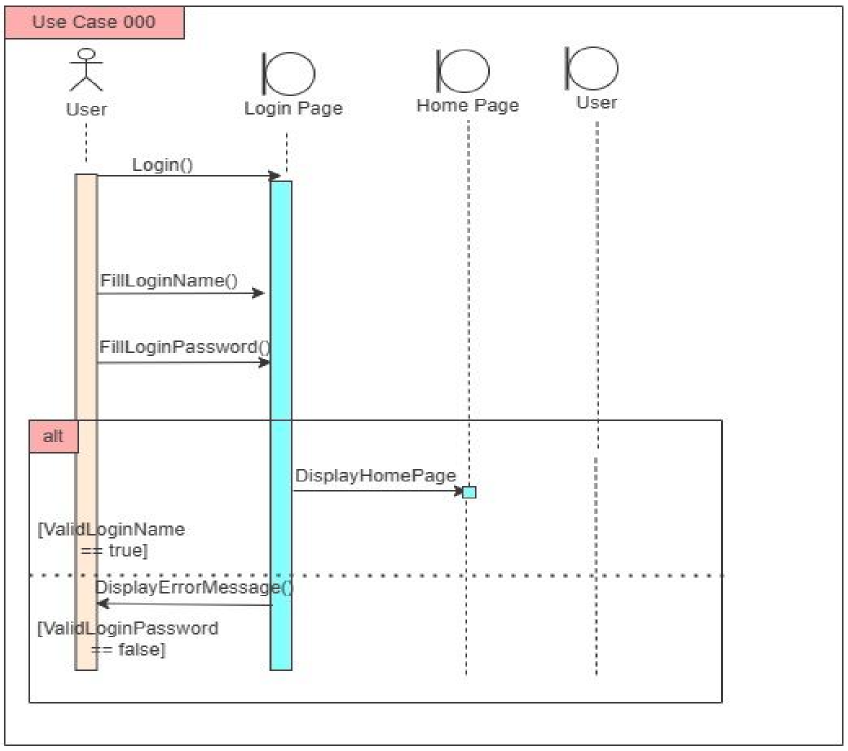


Figure 2.2.1.1 Sequence Diagram for <Login>

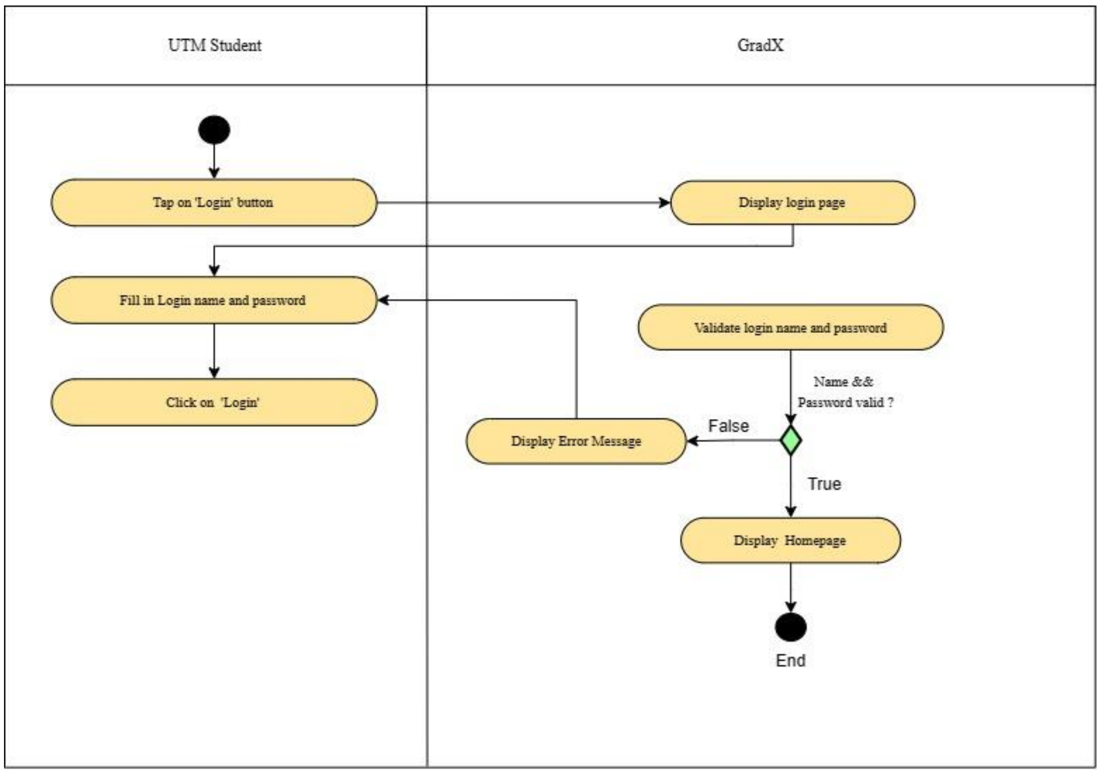


Figure 2.2.1.2: Activity Diagram for <Login>

2.2.2 UC002: Use Case <Project Proposal Submission>

Table 2.2.2: Use Case Description for <Project Proposal Submission>

Use case: <Project Proposal Submission>
ID: UC002
Actors: Student
Preconditions: A valid student is logged on to the system.
Flow of events: 1. Students click on the project submission menu. 2. Students upload the project proposal. 3. Students click “Submit Button”. 4. Message “Successfully submitted” will appear along with the Student details, supervisor name, submission date and time.
Postconditions: 1. Project proposal successfully submitted. 2. Students continue with design solutions. 3. The UC and IC has received and acknowledged the submitted project solutions. Evaluation and feedback on the project solutions will be provided, allowing for further actions or decisions in the project.
Alternative flow *n*: 1. Resubmission: Students are allowed to resubmit their project solution based on the feedback received, they follow the same submission.

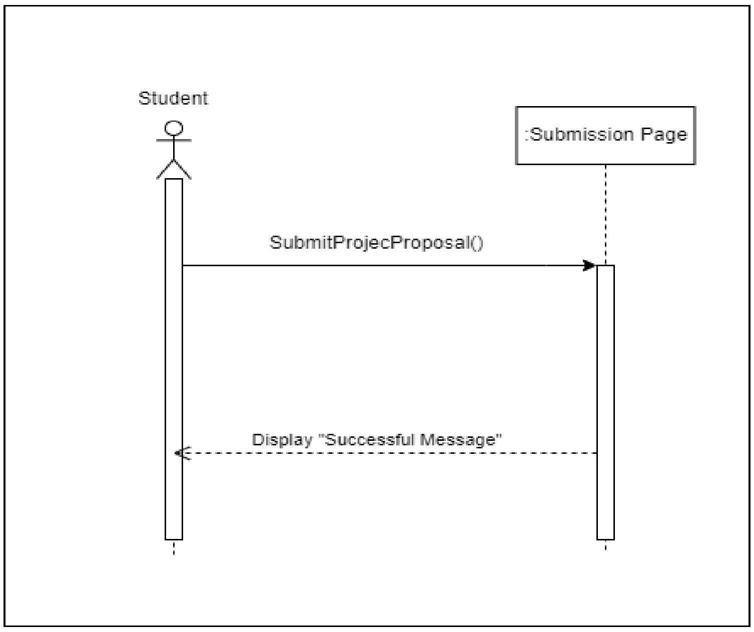


Figure 2.2.2.1: Sequence Diagram for <Submit Project Proposal>

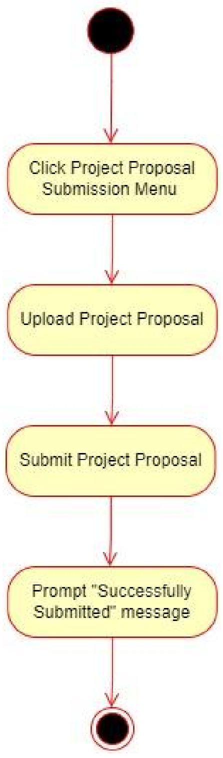

Click Project Proposal Submission Menu
Upload Project Proposal
Submit Project Proposal
Prompt "Successfully Submitted" message

Figure 2.2.2.2: Activity Diagram for <Submit Project Proposal>

2.2.3 UC*003*: Use Case <Design Project Solution>

Table 2.2.3: Use Case Description for <Design Project Solution>

Use case: <Design Project Solution>
ID: UC003
Actors: Student, UC
Preconditions: 1. Student has logged into the system. 2. The project proposal has been approved.
Flow of events: 1. Students select the "Design Project Solution" option from the system menu. 2. Students fill in the details and specifications about their project solution design. 3. Students save the project solution form using the "Save" button. 4. Students can continue working on the saved project solution form. 5. Once the project solution is finalized the student can proceed to the "Submit Project Solution" use case to formally submit the solution.
Postconditions: 1. The project solution is designed and approved. 2. The student has the choice to go on working on the project solution form that has been saved. 3. The completed project solution may be formally submitted and evaluated using the "Submit Project Solution" use case.
Alternative flow *n*: 1. If the UC or IC finds problems or concerns with the suggested project solution, they provide the student comments for changes.
Exception flow (if any): 1. Reports: If there are technical problems or system faults when storing or viewing the project solution form, the student can contact the system administrator for help.

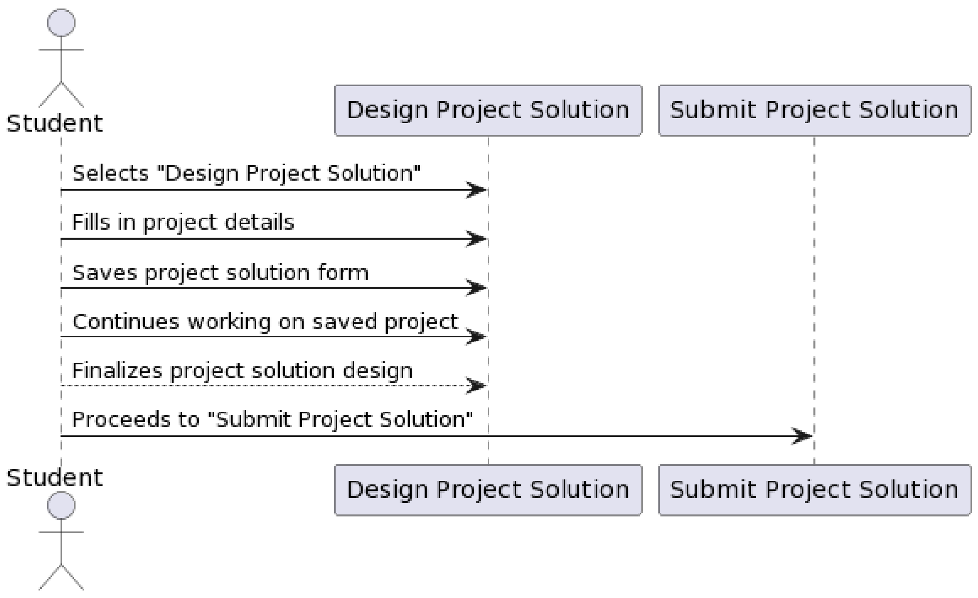

Student
Design Project Solution
Submit Project Solution
Selects "Design Project Solution"
Fills in project details
Saves project solution form
Continues working on saved project
Finalizes project solution design
Proceeds to "Submit Project Solution"
Student
Design Project Solution
Submit Project Solution

Figure 2.2.3.1: Sequence Diagram for <Design Project Solution>

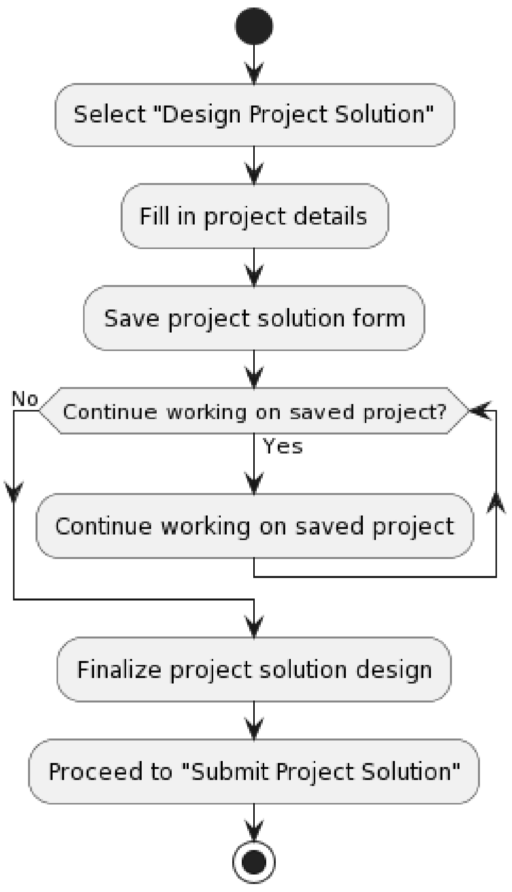

Select "Design Project Solution"
Fill in project details
Save project solution form
No
Continue working on saved project?
Yes
Continue working on saved project
Finalize project solution design
Proceed to "Submit Project Solution"

Figure 2.2.3.2: Activity Diagram for <Design Project Solution>

2.2.4 UC004: Use Case <Project Solution Submission>

Table 2.2.4: Use Case Description for <Submit Project Solution>

Use case: <Project Solution Submission>
ID: UC004
Actors: Student
Preconditions: 1. A valid student is logged on to the system. 2. Students have submitted project proposals.
Flow of events: 1. Students click on the project submission menu. 2. Student upload project solution. 3. Students click on the "Submit" button. 4. Message "Successfully Submitted" will appear along with the Student details, supervisor name, submission date and time.
Postconditions: 1. Project solution successfully submitted. 2. The UC and IC has received and acknowledged the submitted project solutions. Evaluation and feedback on the project solutions will be provided, allowing for further actions or decisions in the project.

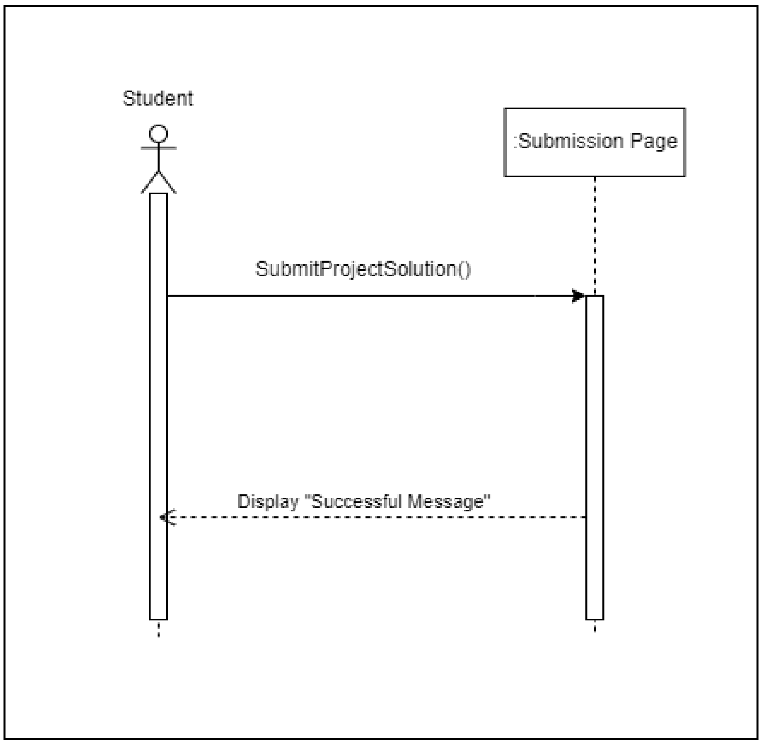


Figure 2.2.4.1: Sequence Diagram for <Submit Project Solution>

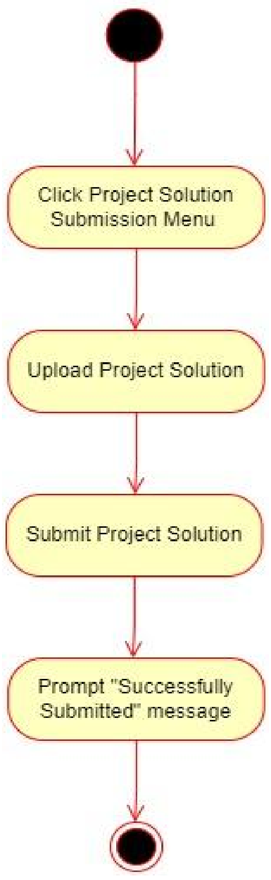

Click Project Solution Submission Menu
Upload Project Solution
Submit Project Solution
Prompt "Successfully Submitted" message

Figure 2.2.4.2: Activity Diagram for <Submit Project Solution>

2.2.5 UC005: Use Case <Checking Project Progress>

Table 2.2.5: Use Case Description for <Checking Project Progress>

Use case: <Checking Project Progress>
ID: UC005
Actors: UC, IC
Preconditions: 1. Student must have submitted their progress in the system to do checking
Flow of events: 1. UC and IC selected and clicked the submitted subtask of the project 2. The system will display the task chosen. 3. UC and IC do the checking progress.
Postconditions: 1. UC and IC had done checking the progress of project from student

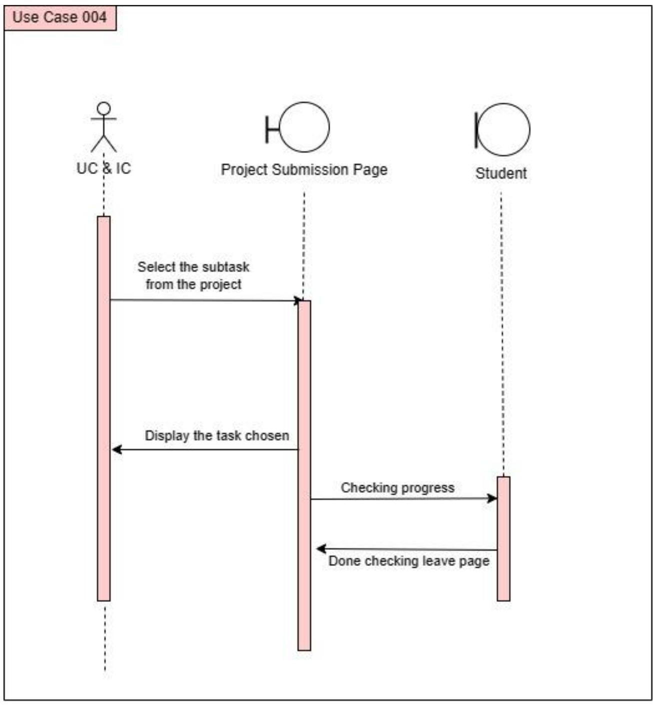


Figure 2.2.5.1 Sequence Diagram for <Checking Project Progress>

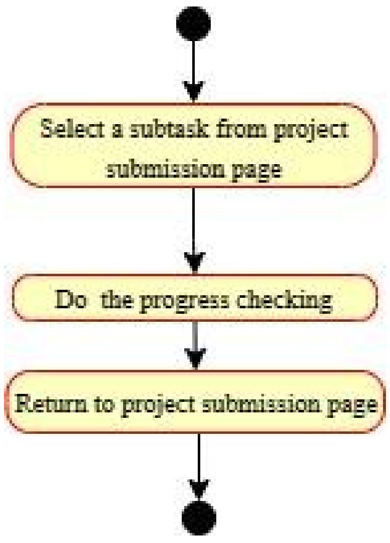

Select a subtask from project submission page
Do the progress checking
Return to project submission page

Figure 2.2.5.2: Activity Diagram for <Checking Project Progress>

2.2.6 UC006: Use Case <Evaluate Student Work>

Table 2.2.6: Use Case Description for <Evaluate Student Work>

Use case: <Evaluate Student Work>
ID: UC0006
Actors: UC, IC, Examiners
Preconditions: 1. Students have submitted their work through the system. 2. Students submitted their work using the right type of file. 3. Users can access and download the rubric and evaluation form that has been provided by the coordinator.
Flow of events: 1. The user receives the student's work that has been submitted through the system. 2. The user needs to download the student submission. 3. The user downloads all the rubrics and evaluation forms. 4. Key in the marks in the evaluation forms based on the rubrics. 5. The user uploads the evaluation forms. 6. Message “Successfully Uploaded” will appear. 7. The system will notify the student.
Postconditions: 1. Evaluation successfully uploaded. 2. Students can access the evaluation that has been uploaded.
Alternative flow *n*: 1. Reuploaded: The user can reupload their evaluation form if any error occurs.

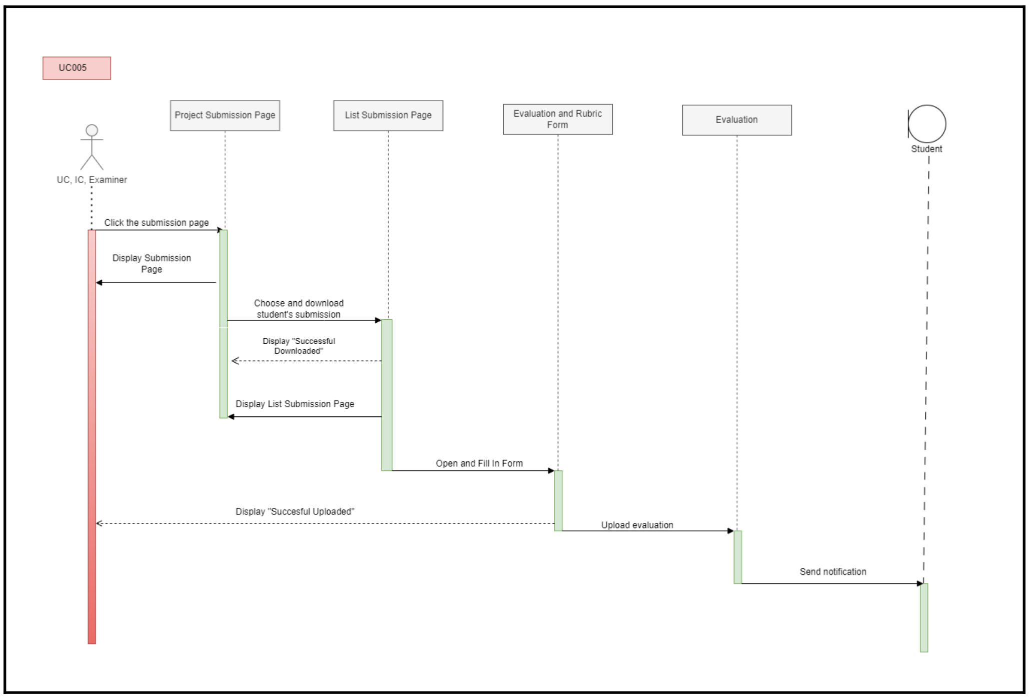

UC005
Project Submission Page
List Submission Page
Evaluation and Rubric Form
Evaluation
Student
UC, IC, Examiner
Click the submission page
Display Submission Page
Choose and download student's submission
Display "Successful Downloaded"
Display List Submission Page
Open and Fill In Form
Display "Succesful Uploaded"
Upload evaluation
Send notification

Figure 2.2.6.1: Sequence Diagram for <Evaluate Student Work>

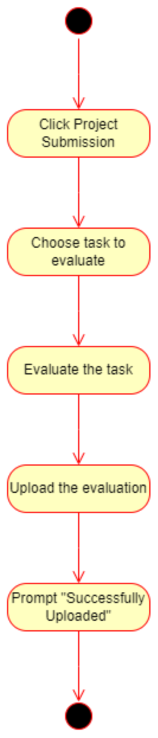


Figure 2.2.6.2: Activity Diagram for <Evaluate Student Work>

2.2.7 UC007: Use Case <Provide Feedback>

Table 2.2.7: Use Case Description for <Provide Feedback>

Use case: <Provide Feedback>
Description: This use case is for
ID: UC007
Actors: UC,IC
Preconditions: 1. User must download the feedback form provided by coordinator
Flow of events: 1. Fill in the feedback form that has been downloaded 2. User must upload the feedback form in the system after complete it
Postconditions: 1. User has provided feedback or guidance in form. 2. Students will be notified and view the feedback given.

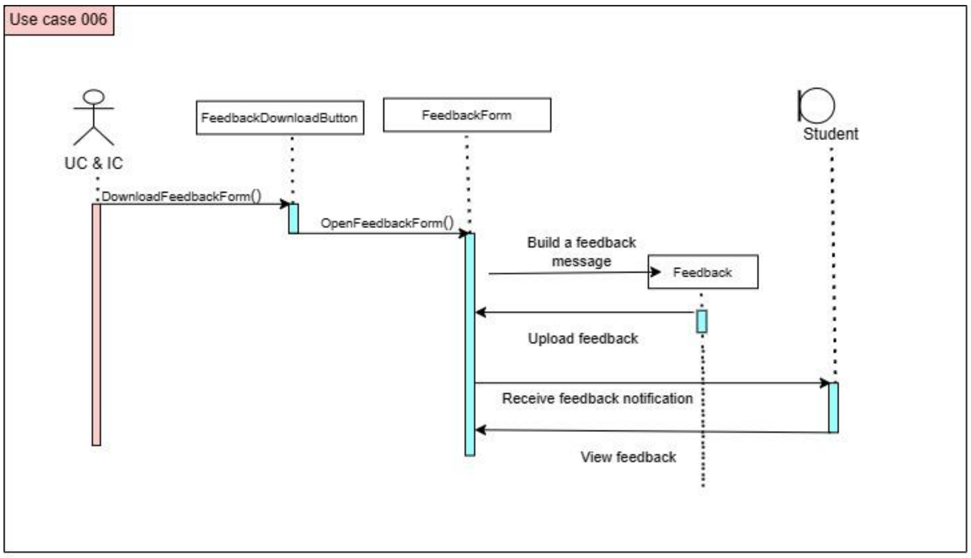


Figure 2.2.7.1 Sequence Diagram for <Provide Feedback>

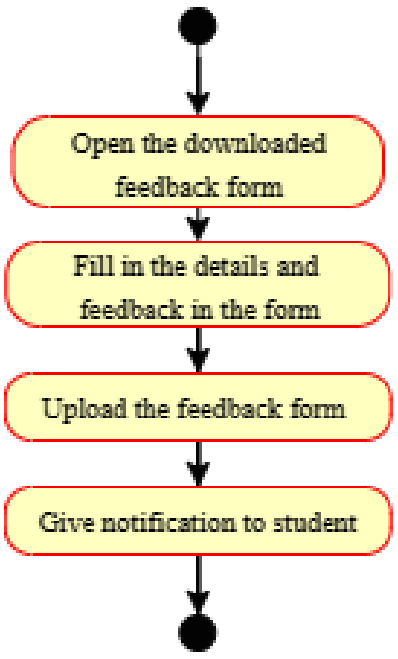

Open the downloaded feedback form
Fill in the details and feedback in the form
Upload the feedback form
Give notification to student

Figure 2.2.7.2: Activity Diagram for <Provide feedback>

2.2.8 UC008: Use Case <Provide Necessary Form>

Table 2.2.8: Use Case Description for <Provide Necessary Form>

Use case: <Provide Necessary Form>
ID: UC008
Actors: Coordinator
Preconditions: 1. Coordinator successfully logged on to the systems
Flow of events: 1. Coordinator uploads all the necessary form into the system 2. Coordinator select to share with UC, IC and examiners. 3. UC, IC and examiners will get notified of the shared rubrics.
Postconditions: 1. UC, IC and examiners can see all the necessary forms through the system. 2. UC, IC and examiners are able to download all the necessary forms.
Alternative flow *n*: 1. Forms modification: If any form is updated and uploaded, UC, IC and the examiner will receive an alert notification.

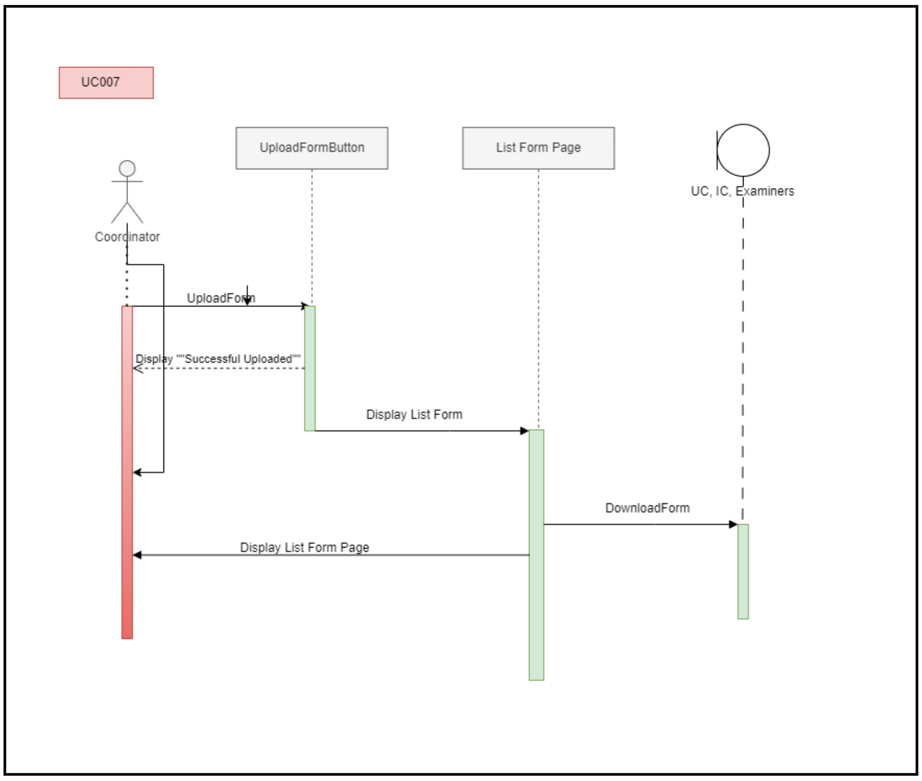


Figure 2.2.8.1: Sequence Diagram for <Provide Necessary Form>

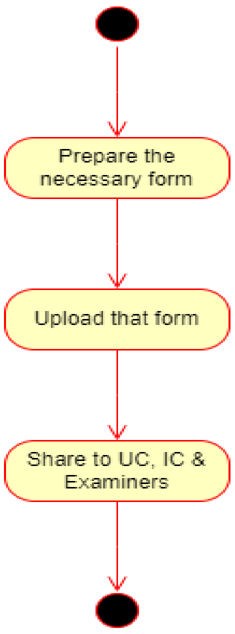


Figure 2.2.8.2: Activity Diagram for <Provide Necessary Form>

2.2.9 UC009: Use Case <Provide Rubrics>

Table 2.2.9: Use Case Description for <Provide Rubrics>

Use case: <Provide Rubrics>
ID: UC009
Actors: Coordinator, Students, Examiner
Preconditions: Coordinator successfully logged on to the systems
Flow of events: 1. Coordinator uploads the rubric into the system 2. Coordinator select to share with students and examiner 3. Student and examiner will get notified of the shared rubrics
Alternative flow *n*: 1. Rubrics modification: If the rubrics are updated and uploaded, both the student and the examiner will receive an alert notification.
Postconditions: 1. Evaluation can be done based on the rubrics 2. Student and Examiner can see the rubrics through the system

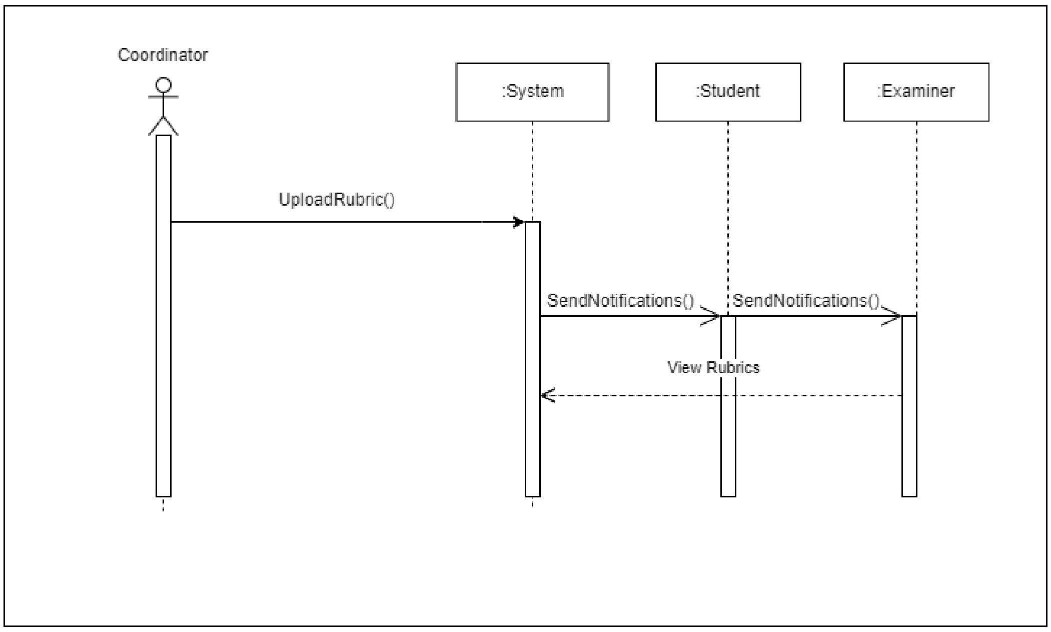


Figure 2.2.9.1: Sequence Diagram for <Provide Rubrics>

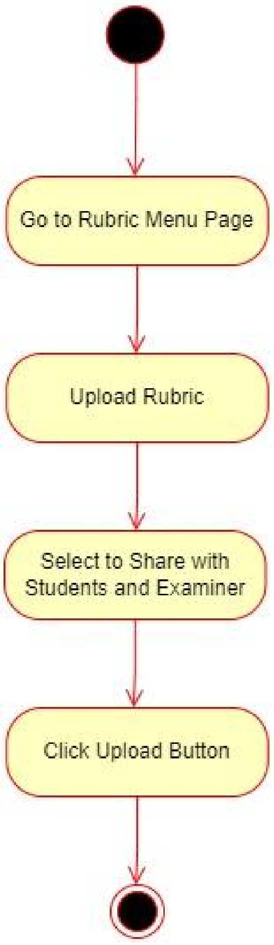


Figure 2.2.9.2: Activity Diagram for <Provide Rubrics>

2.2.10 UC010: Use Case <Assign UC and Examiner>

Table 2.2.10: Use Case Description for <Assign UC and Examiner>

Use case: <Assign UC and Examiner>
ID: UC010
Actors: Coordinator
Preconditions: 1. Coordinator has logged into the system. 2. Students have submitted their project proposal. 3. The project proposal has been reviewed and approved.
Flow of events: 1. Coordinator selects the "Assign University Coach and Examiner" option from the system menu. 2. The system displays a list of available UC and examiners. 3. Coordinator selects a university coach from the list. 4. Coordinator selects an examiner from the list. 5. The system assigns the selected UC and examiner to the respective student's project.
Postconditions: 1. UC and examiner are assigned to the student's project. 2. Assigned UC and examiners receive notifications and access to the project details. 3. Evaluation and assessment of the project can proceed under the assigned UC and examiner.
Alternative flow *n*: 1. If the originally assigned UC or examiner becomes unavailable or unable to fulfill their responsibilities, the coordinator can reassign a new UC or examiner to the student's project.
Exception flow (if any): 1. Report: If there are any technical difficulties or system errors while assigning the UCand examiner, the coordinator can report the issue to the system administrator for resolution. 2. Consultation: If there are disagreements or conflicts regarding the assignment of the UC or examiner, the coordinator can initiate discussions and consultations with relevant stakeholders to resolve the issue.

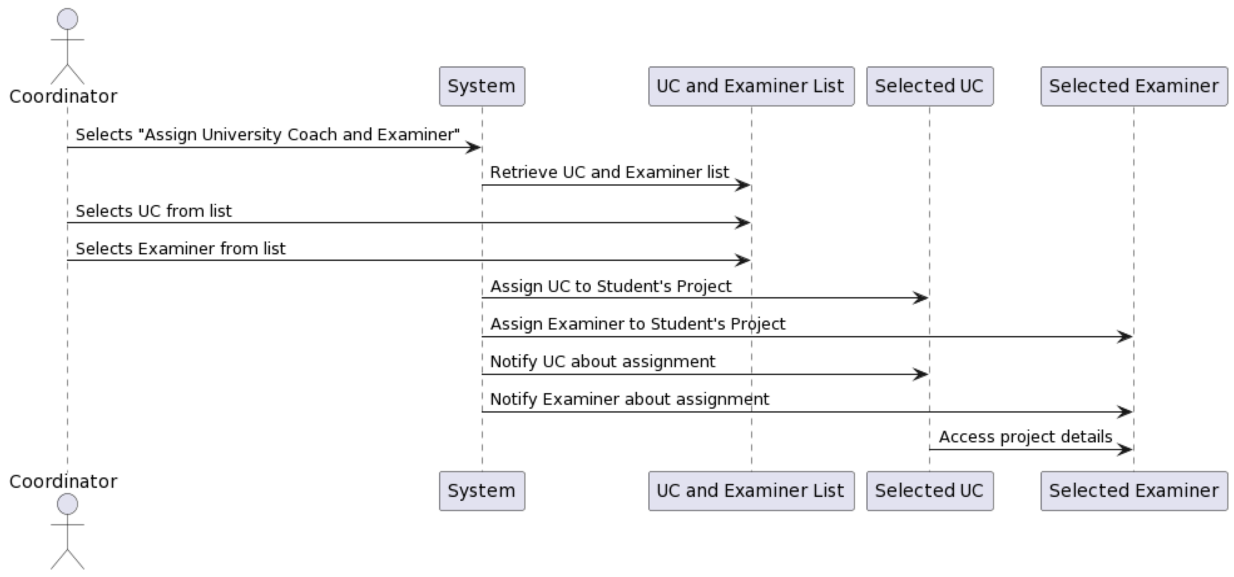

Coordinator
System
UC and Examiner List
Selected UC
Selected Examiner
Selects "Assign University Coach and Examiner"
Retrieve UC and Examiner list
Selects UC from list
Selects Examiner from list
Assign UC to Student's Project
Assign Examiner to Student's Project
Notify UC about assignment
Notify Examiner about assignment
Access project details

Figure 2.2.10.1: Sequence Diagram for <Assign UC and Examiner>

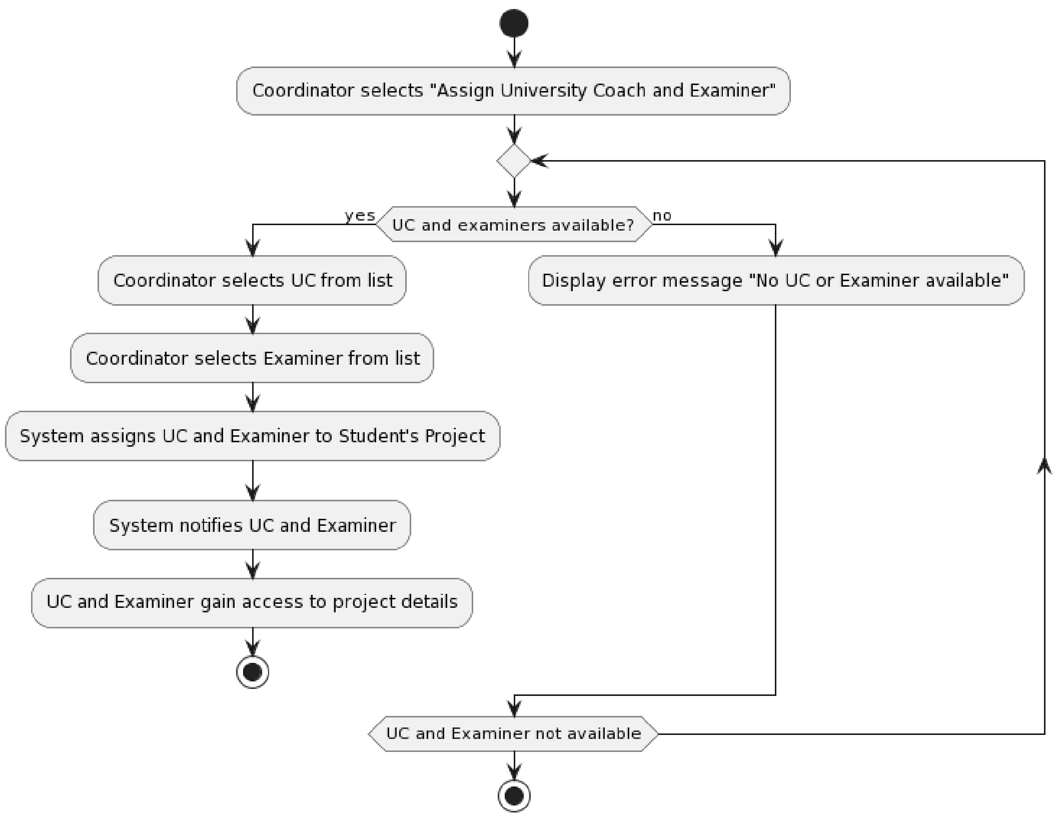

Coordinator selects "Assign University Coach and Examiner"
yes
UC and examiners available?
no
Coordinator selects UC from list
Display error message "No UC or Examiner available"
Coordinator selects Examiner from list
System assigns UC and Examiner to Student's Project
System notifies UC and Examiner
UC and Examiner gain access to project details
UC and Examiner not available

Figure 2.2.10.2: Activity Diagram for <Assign UC and Examiner>

2.2.11 UC011: Use Case <Calculate Student Marks>

Table 2.2.11: Use Case Description for <Calculate Student Marks>

Use case: <Calculate Student Marks>
ID: UC011
Actors: Examiner, Coordinator
Preconditions: 1. The assigned examiner has completed the evaluation of the student's project. 2. The project marks calculation process is initiated.
Flow of events: 1. Examiner selects the "Calculate Student Mark" option from the system menu. 2. The system displays a list of students who have submitted their project solutions. 3. Examiner selects the student for whom the mark needs to be calculated. 4. The system retrieves the evaluation results and relevant assessment criteria for the selected student's project. 5. Examiner reviews the evaluation results, assesses each criterion, and assigns marks accordingly. 6. Examiner enters the calculated marks for each criterion into the system. 7. The system calculates the total mark based on the allocated marks for each criterion. 8. Examiner reviews and verifies the calculated total mark. 9. Examiner submits the calculated marks for the selected student. 10. The system updates the student's record with the calculated marks.
Postconditions: 1. The student's project marks are calculated and recorded in the system. 2. The student can view their final marks for the project. 3. The recorded marks are used for final grading and assessment purposes.
Alternative flow *n*: 1. Revision: If there are any errors or inconsistencies in the submitted marks, the system prompts the examiner to review and correct the marks before resubmitting. 2. Re-evaluate: If the coordinator notices any problems or inconsistencies in the computed marks, they can ask the examiner to re-evaluate or alter the marks.

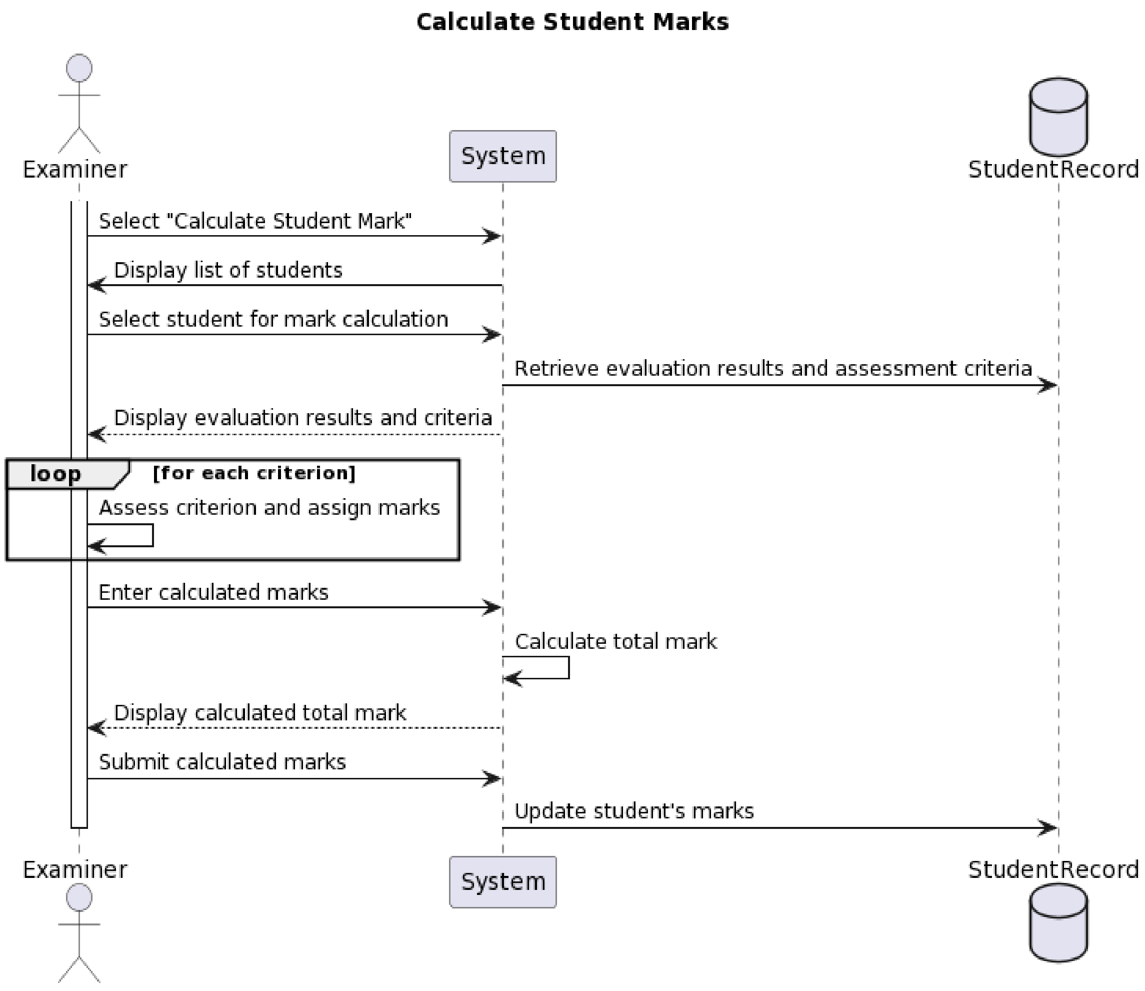

Calculate Student Marks
Examiner
System
StudentRecord
Select "Calculate Student Mark"
Display list of students
Select student for mark calculation
Retrieve evaluation results and assessment criteria
Display evaluation results and criteria
loop
[for each criterion]
Assess criterion and assign marks
Enter calculated marks
Calculate total mark
Display calculated total mark
Submit calculated marks
Update student's marks
Examiner
System
StudentRecord

Figure 2.2.11.1: Sequence Diagram for <Calculate Student Marks>

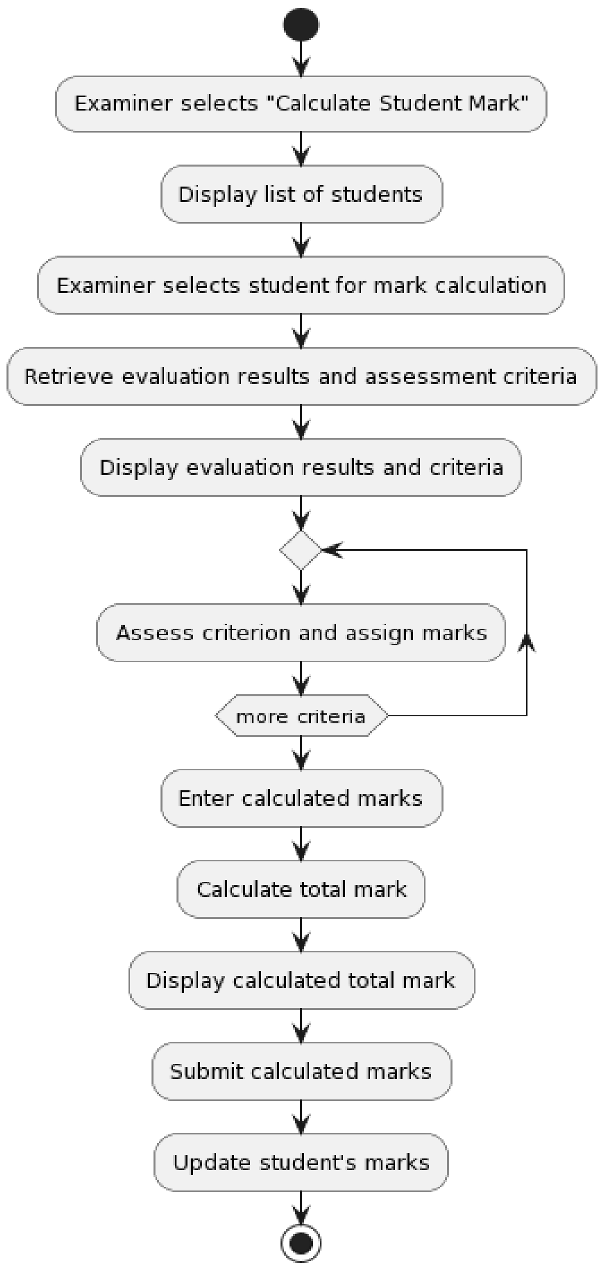

Examiner selects "Calculate Student Mark"
Display list of students
Examiner selects student for mark calculation
Retrieve evaluation results and assessment criteria
Display evaluation results and criteria
Assess criterion and assign marks
more criteria
Enter calculated marks
Calculate total mark
Display calculated total mark
Submit calculated marks
Update student's marks

Figure 2.2.11.2: Activity Diagram for <Calculate Student Mark>

2.3 Software System Attributes, Performance and Other Requirements

Software System Attributes:

Usability: The system should be user-friendly and intuitive, with a clear and easy-to-navigate interface.

Maintainability: The system should be designed with modular and easily maintainable code, with clear documentation to aid in future maintenance and updates.

Compatibility: The system should be compatible with various web browsers and operating systems commonly used by students and faculty.

Performance:

Response Time: The system should be able to respond to user requests within a reasonable time frame, ideally less than 3 seconds.

Throughput: The system should be able to handle a large number of concurrent users and requests during peak usage times, without compromising performance.

Capacity: The system should be able to handle large amounts of data and user activity, without experiencing slow-downs or crashes.

Other Requirements:

Security: The system should incorporate appropriate security measures to protect user data and prevent unauthorized access, such as encryption, secure authentication, and regular backups.

Legal and Regulatory: The system should comply with relevant laws, regulations, and standards, such as data privacy laws and accessibility guidelines.

Environmental: The system should be designed with energy efficiency in mind, with features such as automatic power-saving modes and optimized server configurations to reduce energy consumption.

2.4 Design Constraints

Usability constraints: In order for the system to be easily accessible and understood by users, it needs to follow a basic knowledge of other websites.

System constraints: Additions to the system must require minimal or no modifications to the existing system.

Security constraints: An integrated secure authentication system shall be provided to prevent unauthorized access. Access to the system is restricted to coordinators, university coaches, industry coaches, examiners, and fourth-year Data Engineering students.

Compatibility constraints: The system should be designed so that it can be accessed both via desktop and mobile devices.

Performance constraints: With a response time of less than 3 seconds, the system must be able to handle 500 simultaneous users.

3. System Architectural Design

3.1 Architecture Pattern and Rationale

For this project, we have decided to use the Model View Controller (MVC) architectural pattern. MVC Distinguishes presentation and interaction from system data. The system is divided into three logical components that communicate with one another.The Model component maintains the system data and the actions that are performed on that data. The View component specifies and maintains how data is displayed to the user. The Controller component organizes user interaction (key presses, mouse clicks, and so forth) and delivers it to the View and the Model.

MVC is normally used when there are multiple ways to view and interact with data. Also used when the future requirements for interaction and presentation of data are unknown.

MVC encourages a modular, structured approach to programme development, which improves maintainability. If updates or new features are to be added to the programme, they can be done without affecting the others. This decreases the possibility of unwanted side effects and makes the codebase easier to understand and alter.

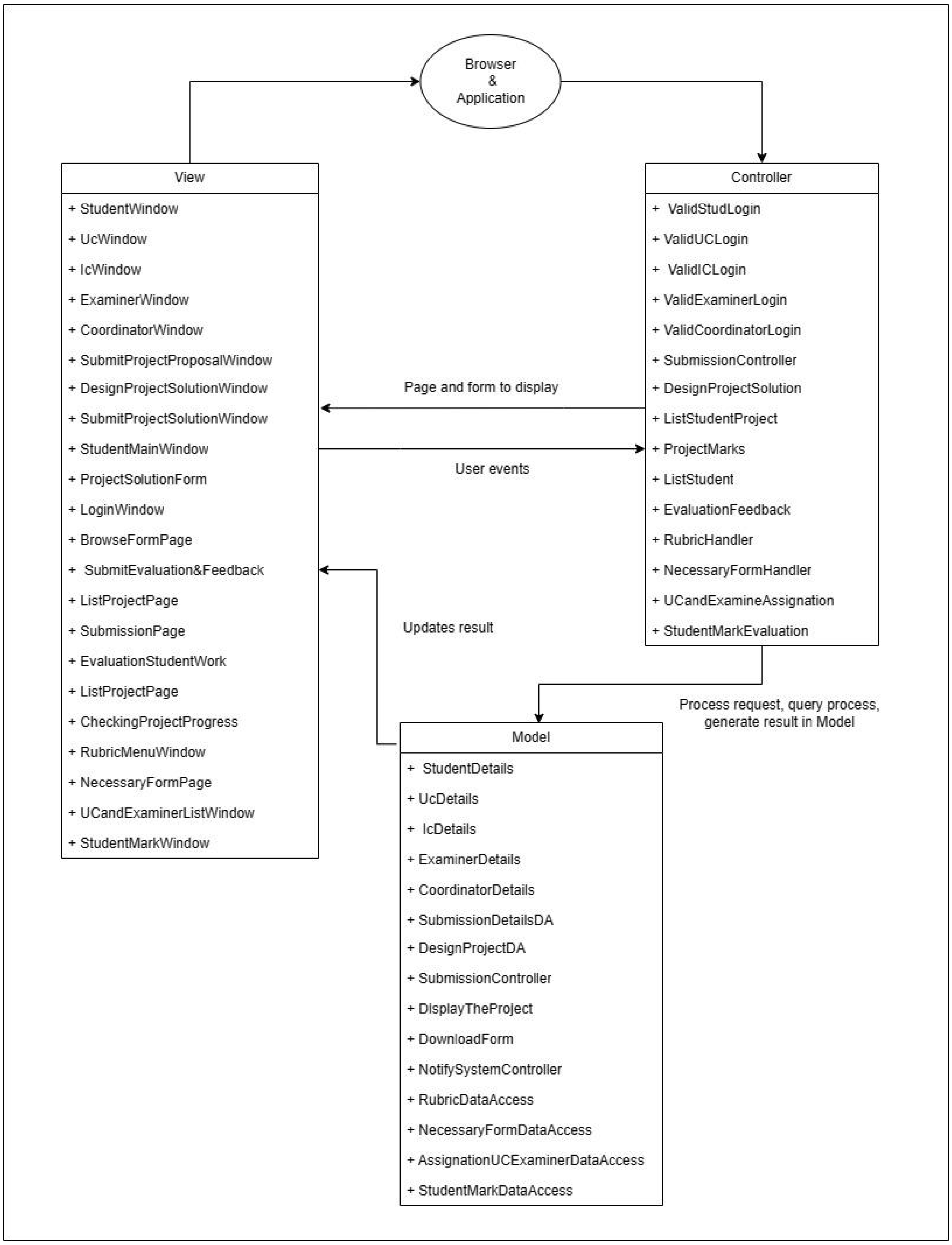


Figure 3.1: Architecture Diagram for <GradX>

3.2 Component Model

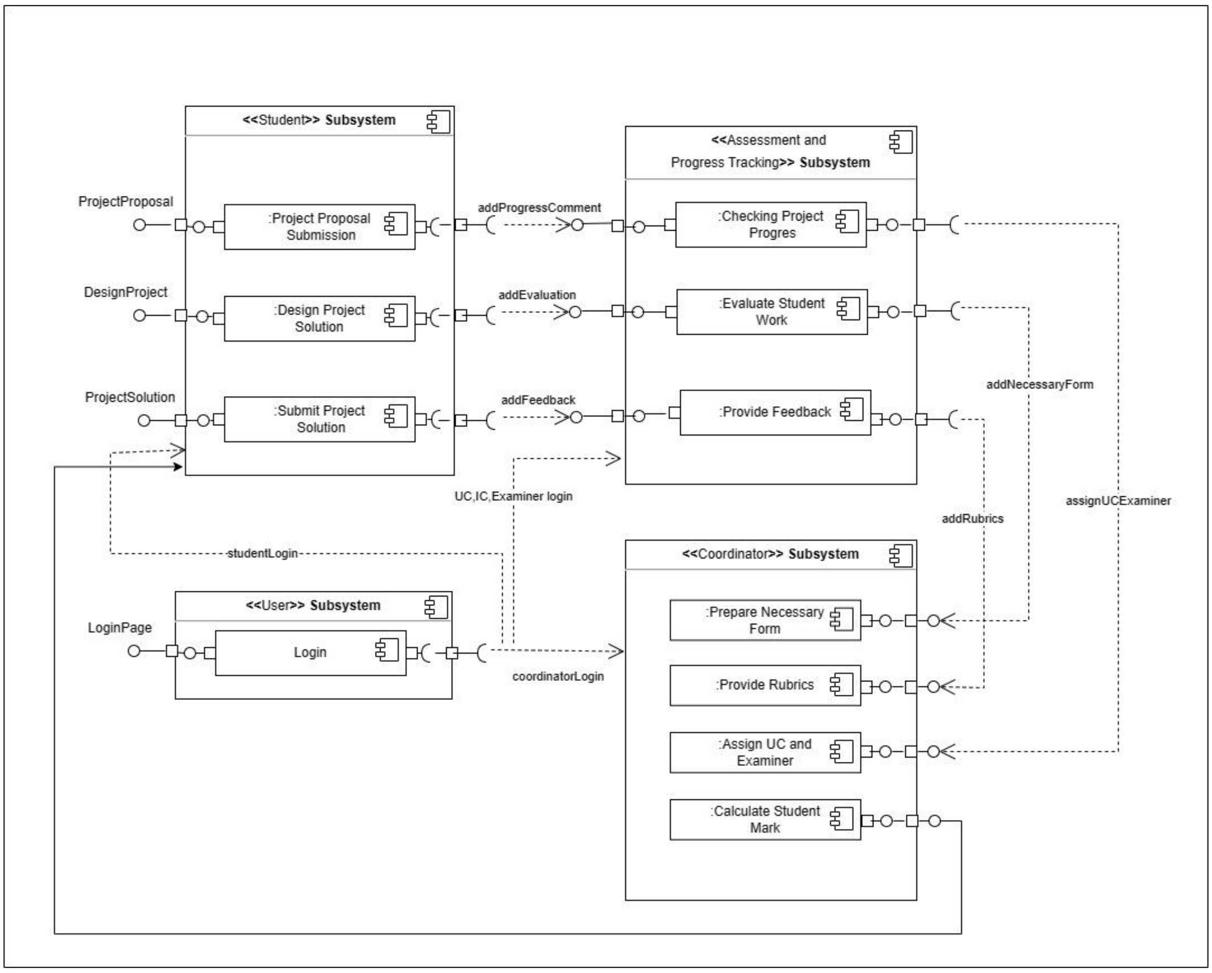


Figure 3.2: Component Diagram of <GradX>

4. Detailed Description of Components

4.1. Complete Package Diagram

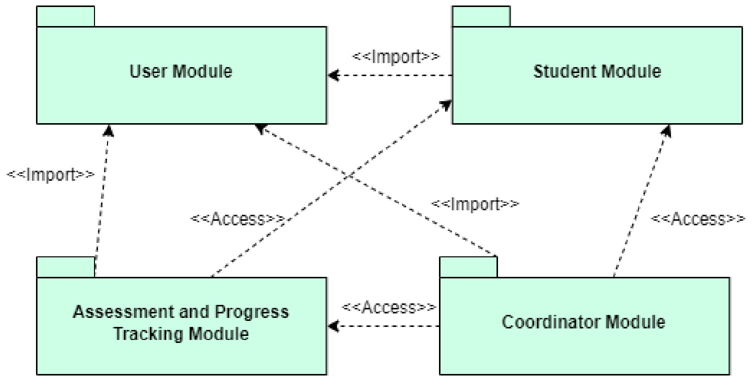


Figure 4.1: Package Diagram for <GradX System>

The complete package diagram comprises four distinct modules: the User module, the Student module, the Assessment and Progress Tracking module, and the Coordinator module. The User module facilitates the login process for all users. The Student module oversees all activities pertaining to students within the system. The Coordinator module is responsible for managing system activities specifically associated with the coordinator role. Lastly, the Assessment and Progress Tracking modules handle the activities of other system users. Before performing their respective functions, the Student module, Coordinator module, and Assessment and Progress Tracking module import data from the User module.

The Coordinator module relies on data from both the Student module and the Assessment and Progress Tracking modules to ensure seamless functionality. Similarly, the Assessment and Progress Tracking modules access data from the Student module to effectively carry out their designated tasks.

4.2. Detailed Description

4.2.1. P001: <User> Subsystem

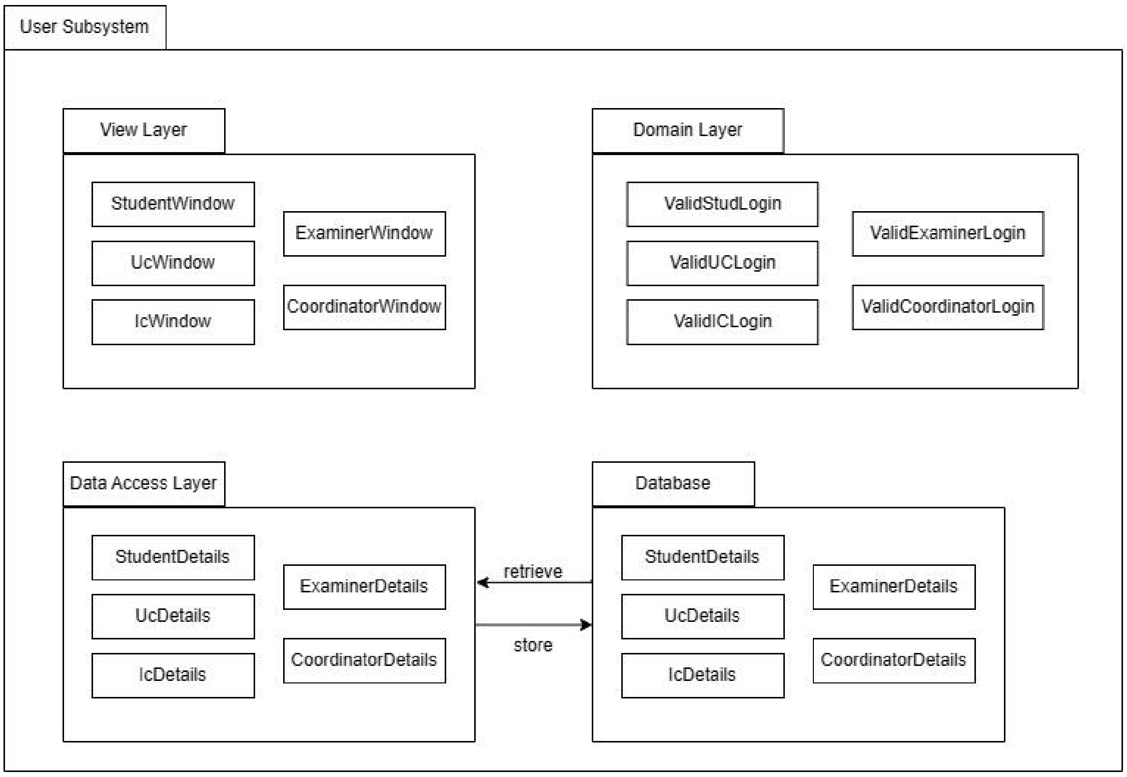


Figure 4.2: Package Diagram for <User> Subsystem

4.2.1.1. Class Diagram

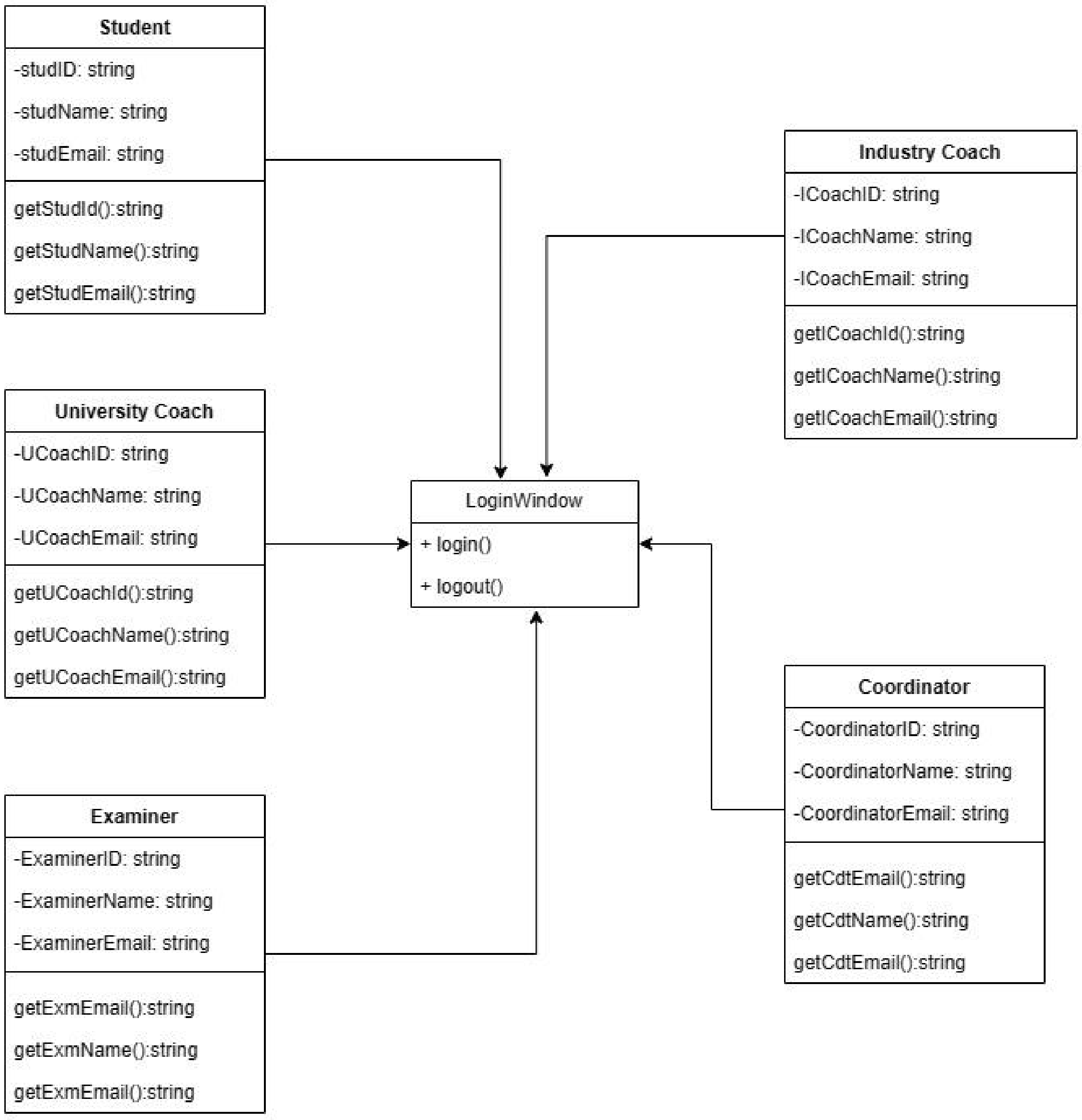
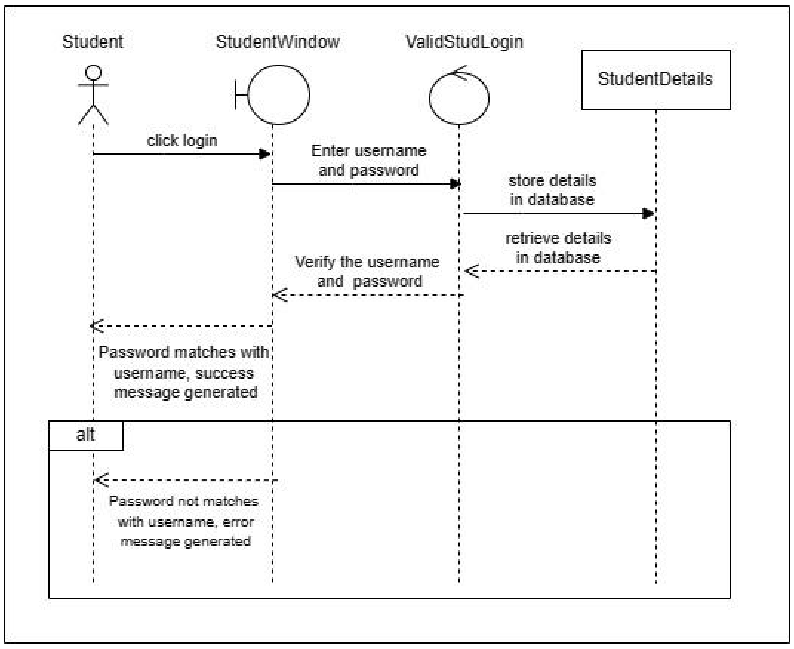


Figure 4.3: Class Diagram for <Login> Subsystem

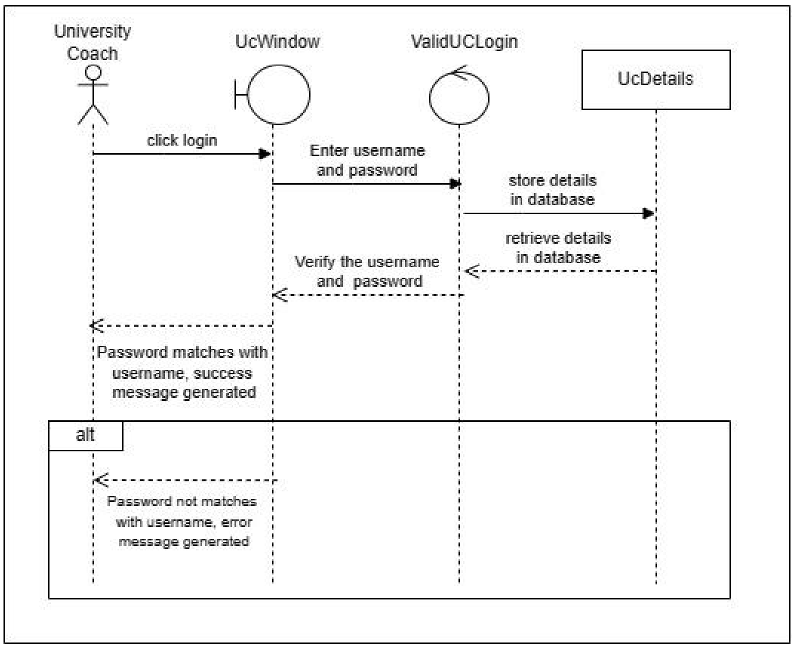
Entity Name	User
Method Name	Login()
Input	1. username (string) 2. Password (string)
Output	1. Success message (string) 2. Error message (string)
Algorithm	1. Start 2. Receive the username and password from the user 3. Verify the username and password are correct and not empty 4. Retrieve and store within the database. 5. If the password matches, a success message will be generated otherwise the incorrect message will be generated. 6. If no matching username is found, an error message is generated to indicate invalid username. 7. Return the success or error message based on the login result. 8. End

4.2.1.2. Sequence Diagram

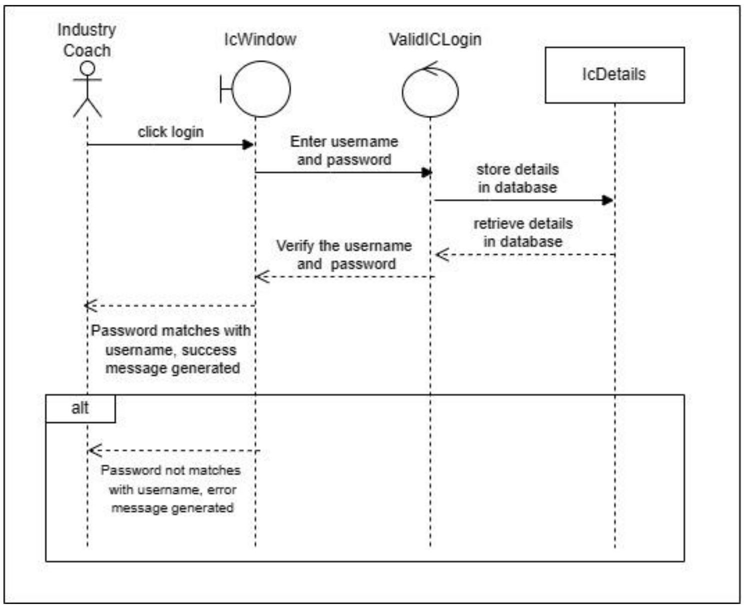
a) SD001: Sequence diagram for Student Login


Student
StudentWindow
ValidStudLogin
StudentDetails
click login
Enter username and password
store details in database
retrieve details in database
Verify the username and password
Password matches with username, success message generated
alt
Password not matches with username, error message generated

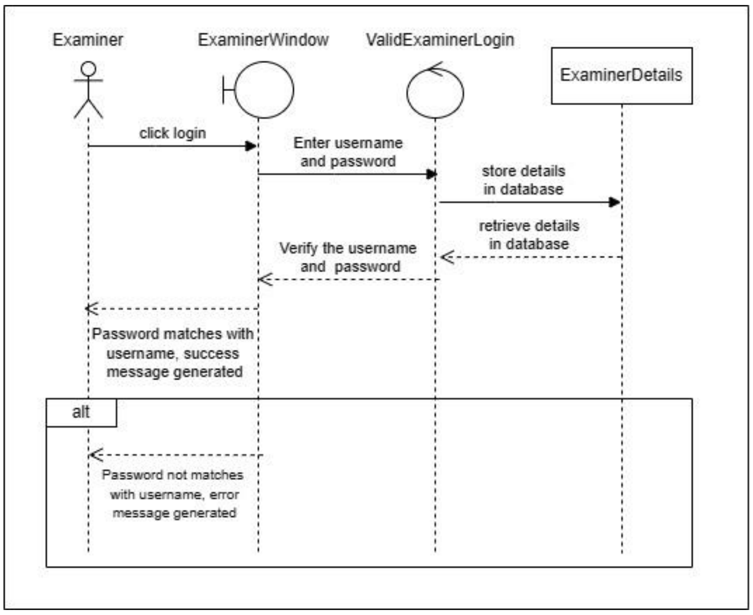
b) SD002: Sequence diagram for University Coach Login


University Coach
UcWindow
ValidUCLogin
UcDetails
click login
Enter username and password
store details in database
retrieve details in database
Verify the username and password
Password matches with username, success message generated
alt
Password not matches with username, error message generated

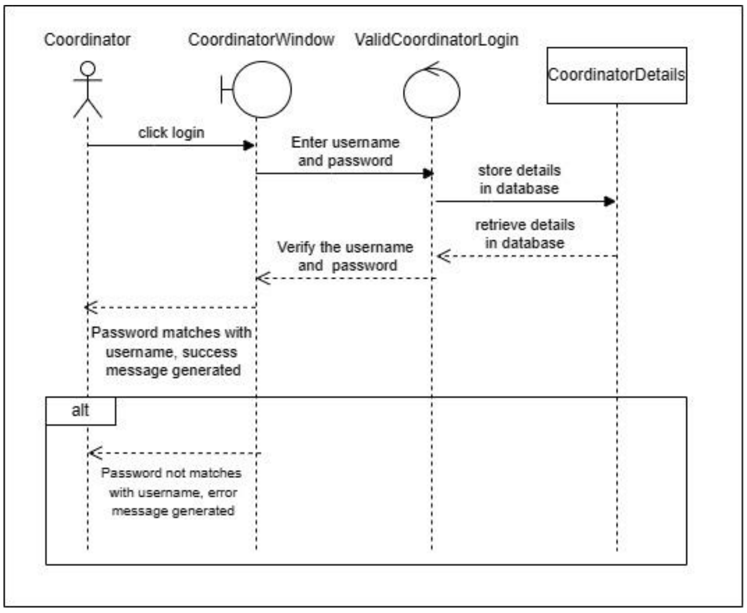
c) SD003: Sequence diagram for Industry Coach Login


Industry Coach
IcWindow
ValidICLogin
IcDetails
click login
Enter username and password
store details in database
retrieve details in database
Verify the username and password
Password matches with username, success message generated
alt
Password not matches with username, error message generated

d) SD004: Sequence diagram for Examiner Login


Examiner
ExaminerWindow
ValidExaminerLogin
ExaminerDetails
click login
Enter username and password
store details in database
retrieve details in database
Verify the username and password
Password matches with username, success message generated
alt
Password not matches with username, error message generated

e) SD005: Sequence diagram for Coordinator Login


Coordinator
CoordinatorWindow
ValidCoordinatorLogin
CoordinatorDetails
click login
Enter username and password
store details in database
retrieve details in database
Verify the username and password
Password matches with username, success message generated
alt
Password not matches with username, error message generated

4.2.2. P002: <Student> Subsystem

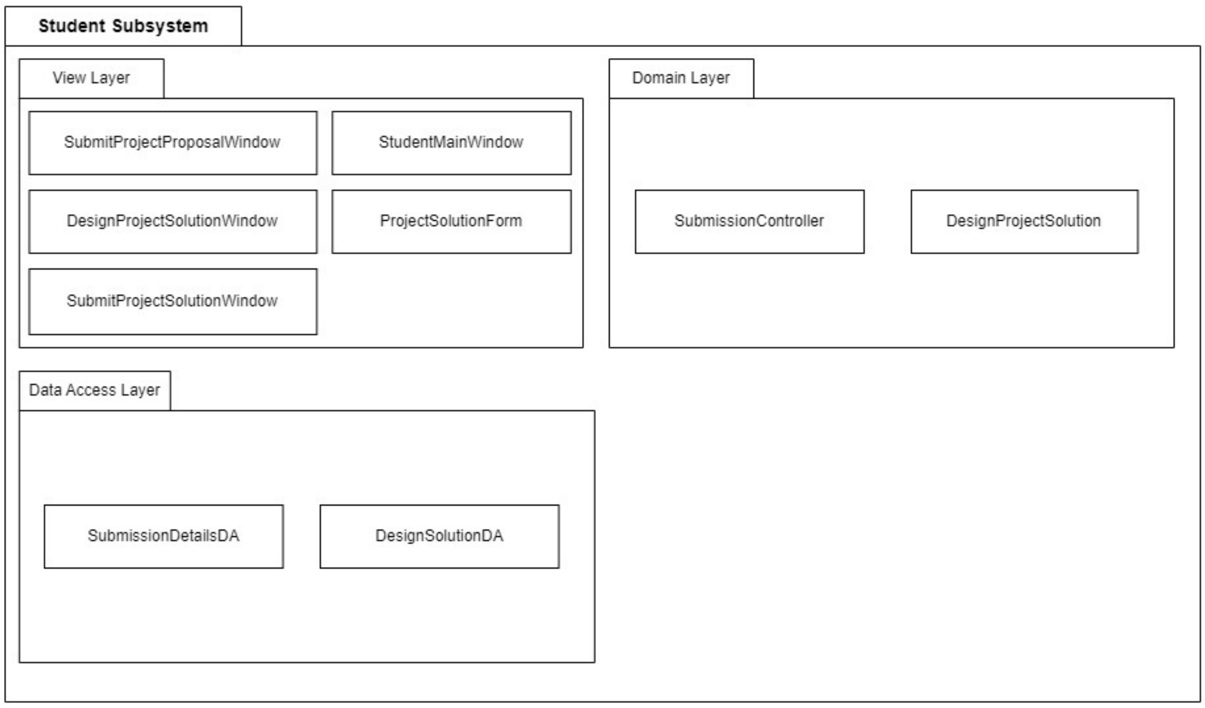
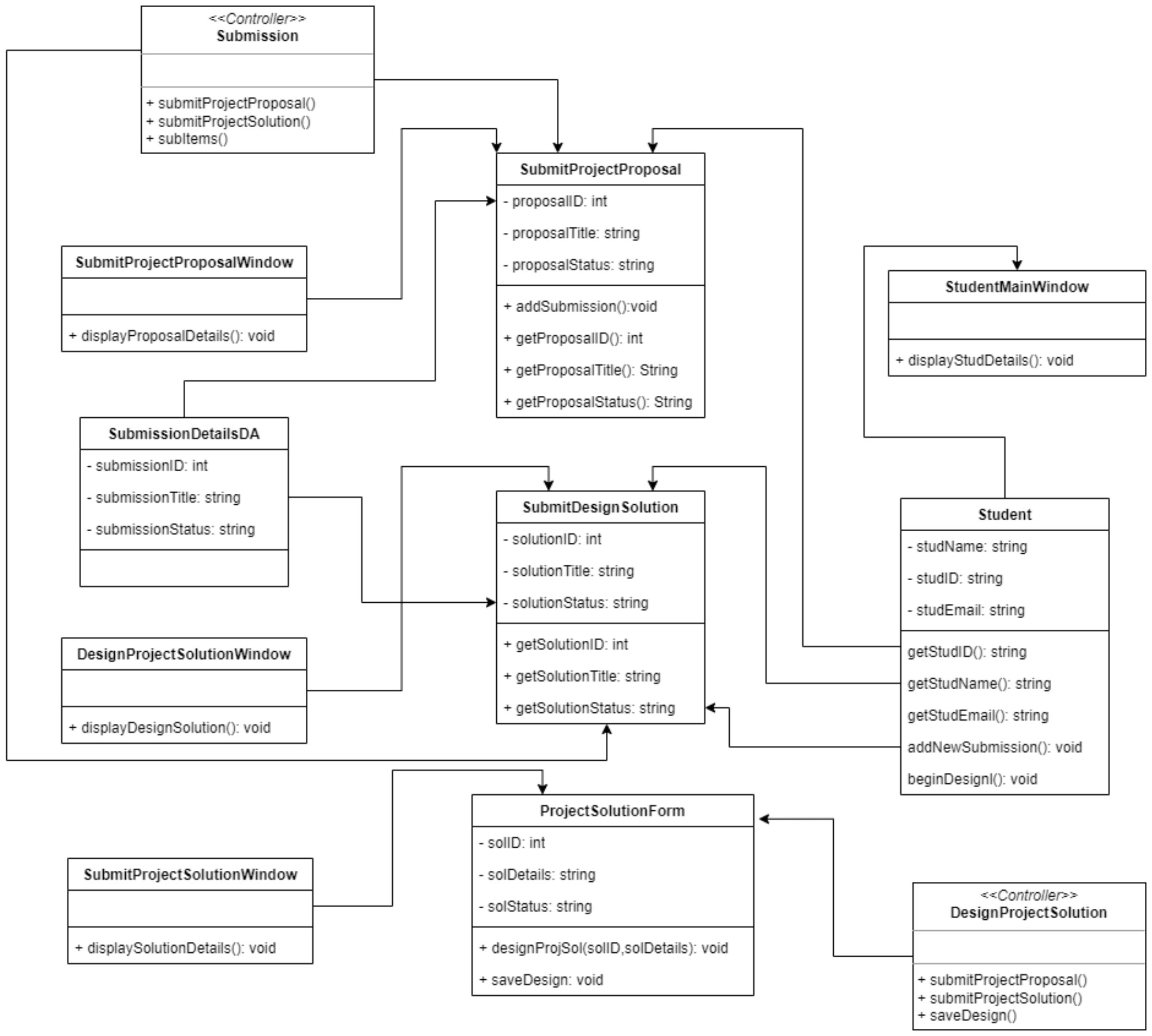


Figure 4.3: Package Diagram for <Student> Subsystem

4.2.2.1. Class Diagram



Entity Name	SubmitProjectProposal
Method Name	addSubmission()
Input	● Project proposal title ● Project proposal file
Output	● Project proposal ID ● Successful message ● Project proposal status
Algorithm	1. Start 2. Navigate to the submit project proposal window 3. Key in project proposal title and details (optional) 4. Upload proposal file 5. Select proposal file from local or cloud storage 6. Submit the proposal by clicking on the “submit” button

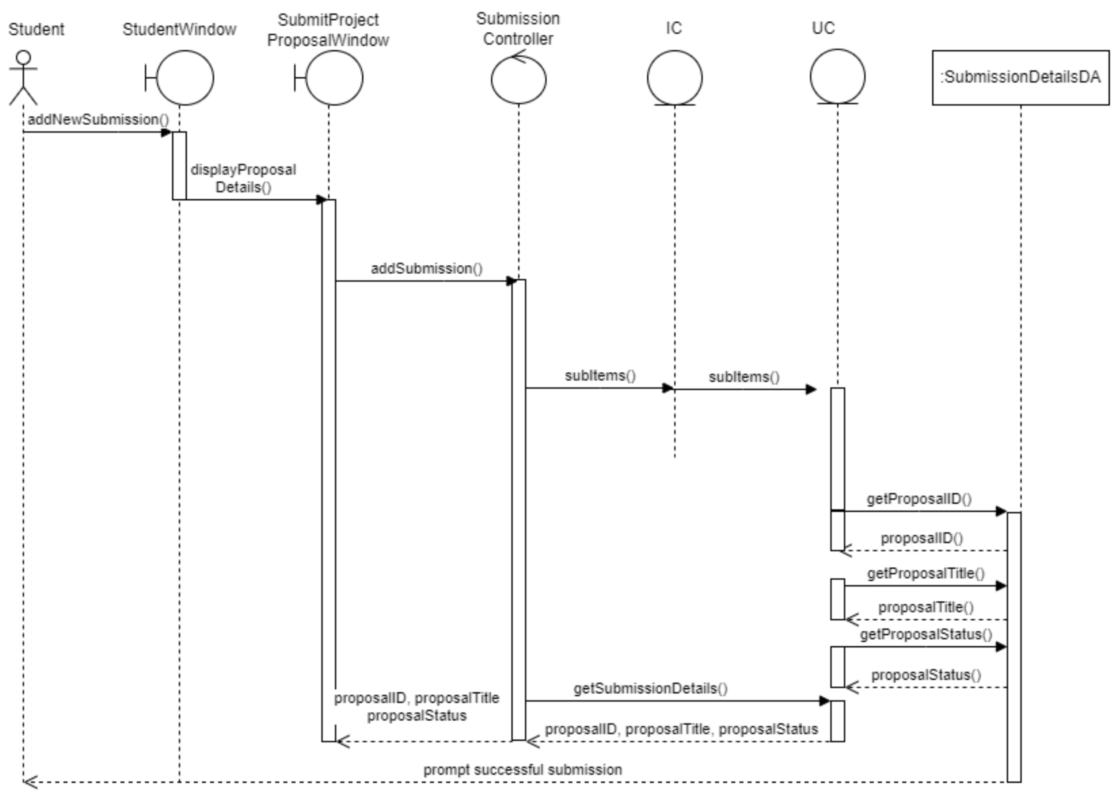
	7. Prompt "Successful Submission" message, proposal id and proposal status upon successful file submission. 8. Stop

Entity Name	ProjectSolutionForm
Method Name	designProjSol()
Input	● Design project information
Output	● User interface of the system for accessing the design software. ● Design solution ID ● Exported or downloaded design files in the desired format. ● Confirmation messages to save design
Algorithm	1. Start 2. Navigate to the submit design project solution window 3. Create new design project 4. Key in design project information 5. Choose and use the design tools to complete the design 6. Save design the progress and confirm the action by clicking the confirm button 7. Export and download the file in desired format 8. If want to modify existing design, click on the saved designs 9. Stop

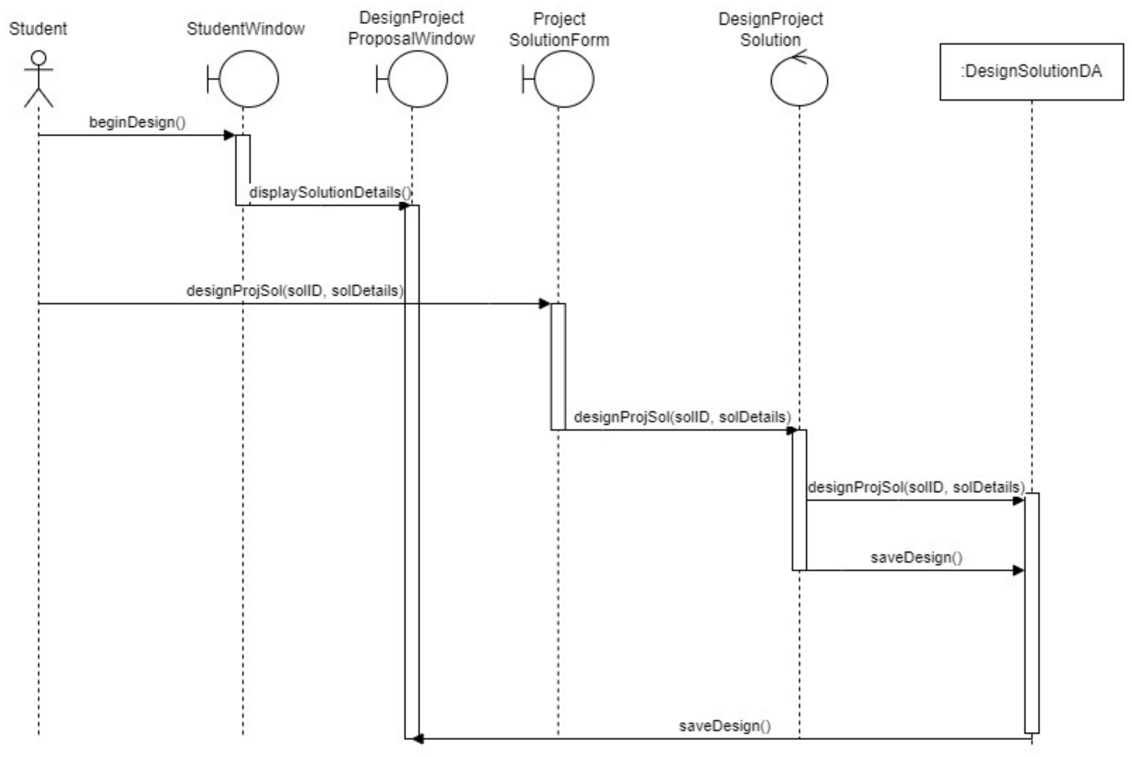
Entity Name	SubmitProjectSolution
Method Name	addSubmission()
Input	● Project solution title ● Project solution file
Output	● Project solution ID ● Successful message ● Project solution status
Algorithm	1. Start 2. Navigate to the submit project solution window 3. Key in project solution title and details (optional) 4. Upload project solution file 5. Select project solution file from local or cloud storage 6. Submit the proposal by clicking on the "submit" button 7. Prompt "Successful Submission" message, solution id and project solution status upon successful file submission. 8. Stop

4.2.2.2. Sequence Diagram

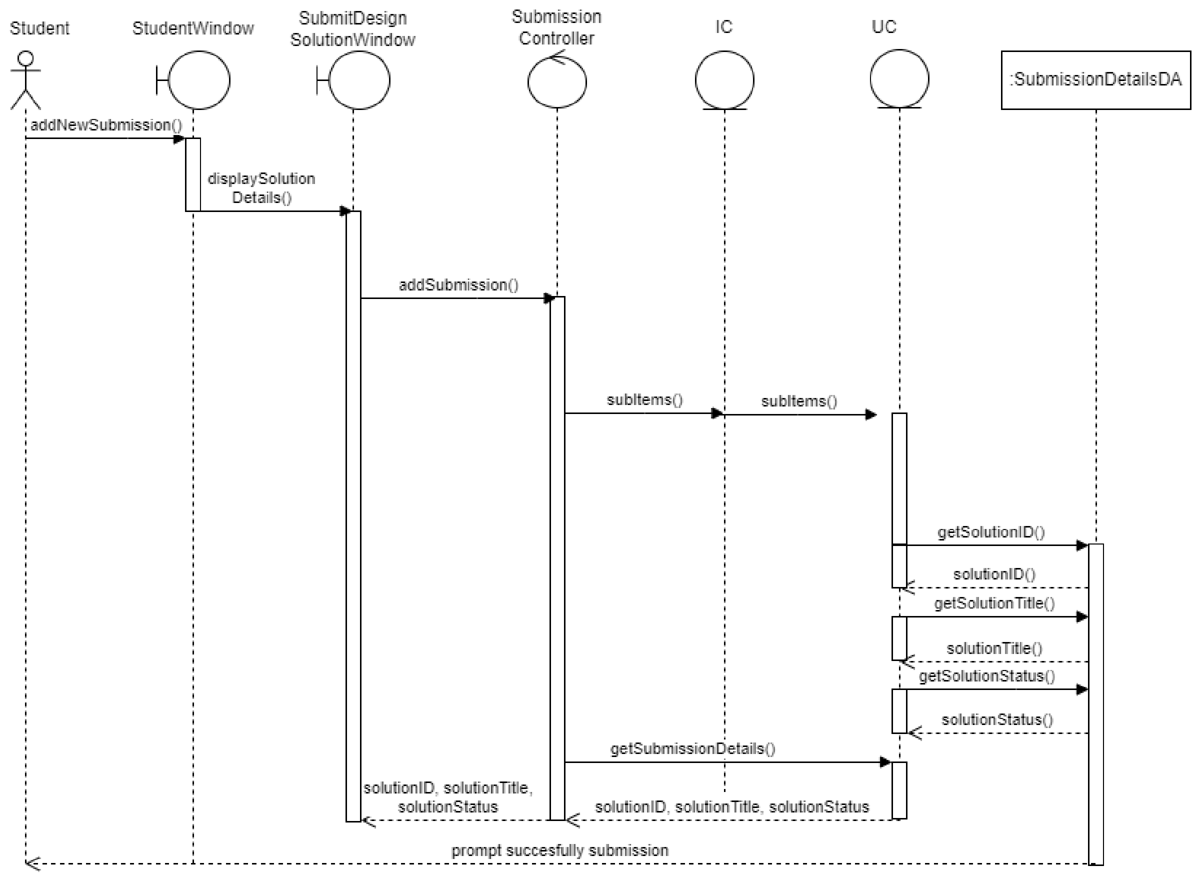
a) SD006: Sequence diagram for Submit Project Proposal



b) SD007: Sequence Diagram for Design project solution



c) SD008: Sequence diagram for Submit Project Solution



4.3. P003: <Assessment and Progress Tracking> Subsystem

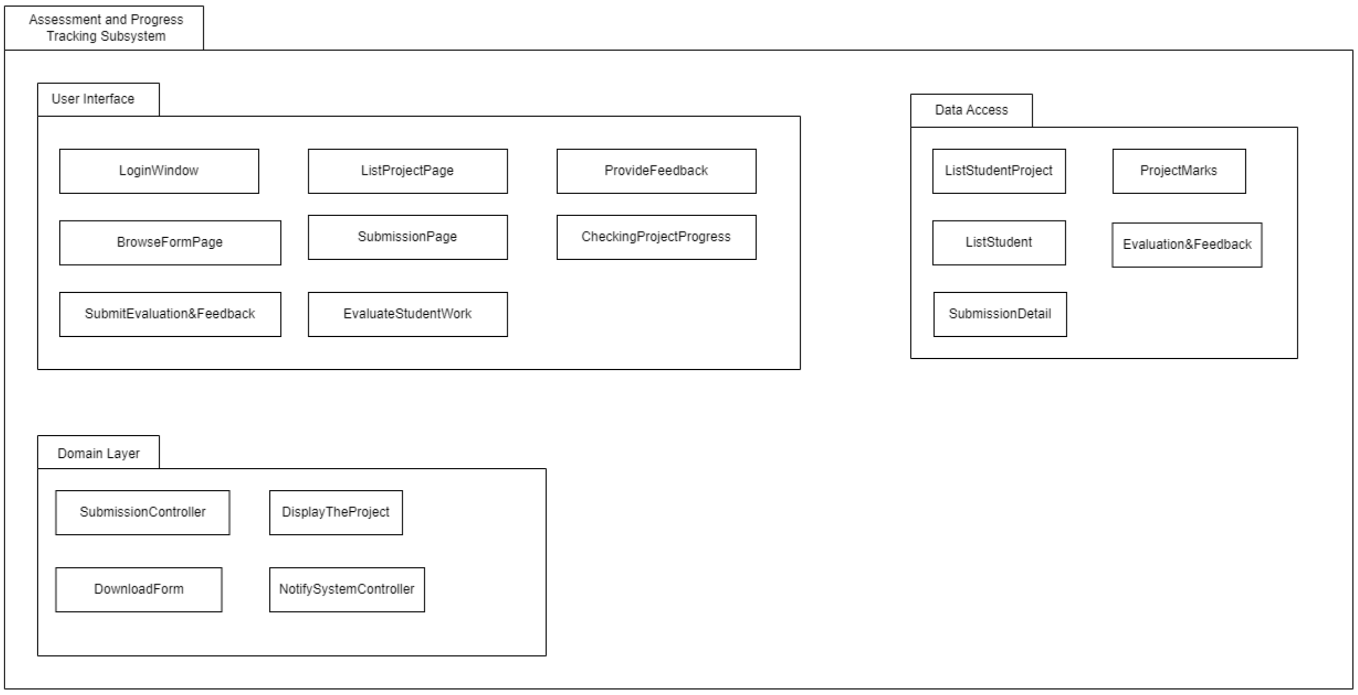
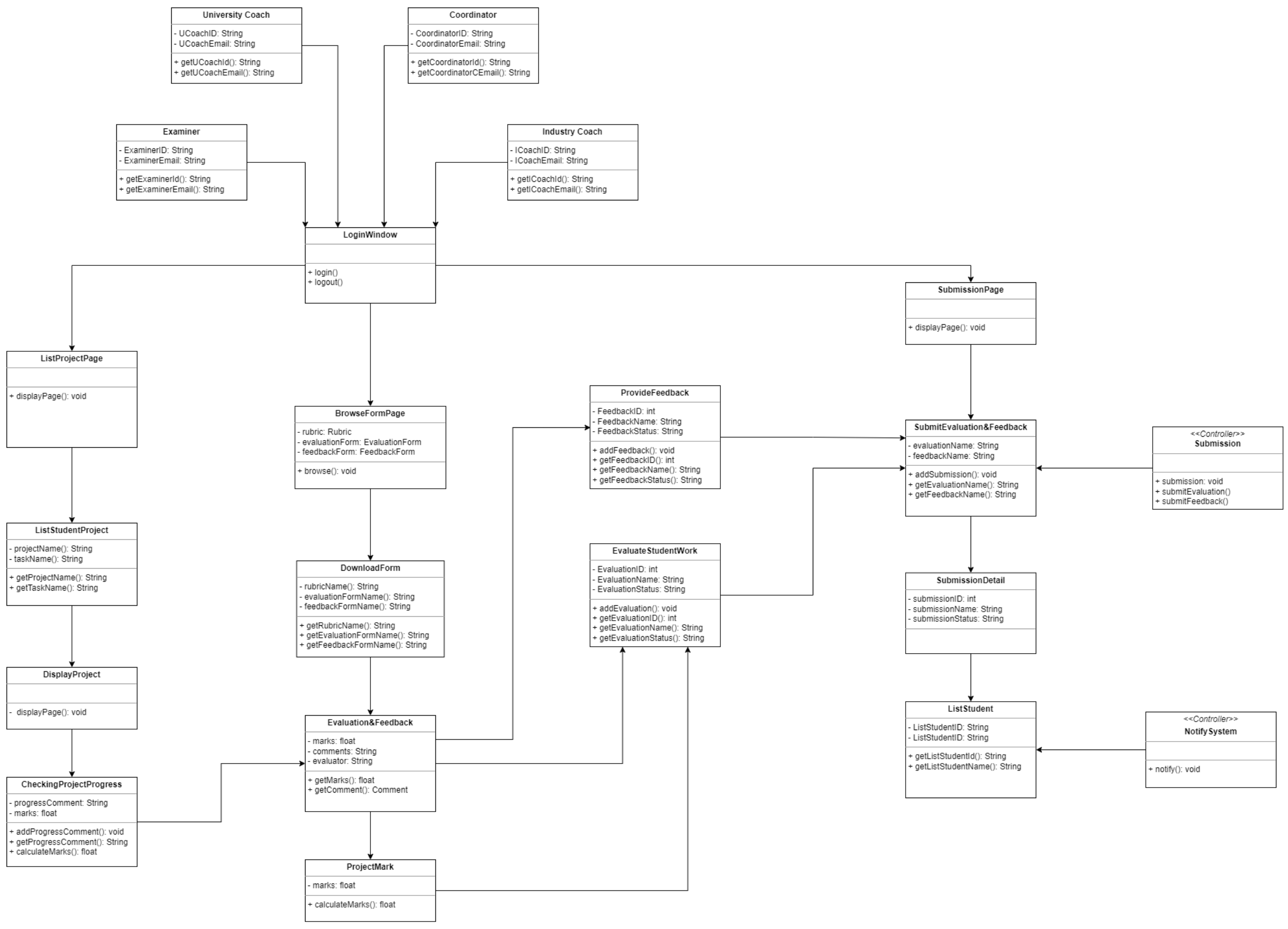


Figure : Package Diagram for <Assessment and Progress Tracking> Subsystem

4.2.3.1. Class Diagram



Entity Name	EvaluateStudentWork

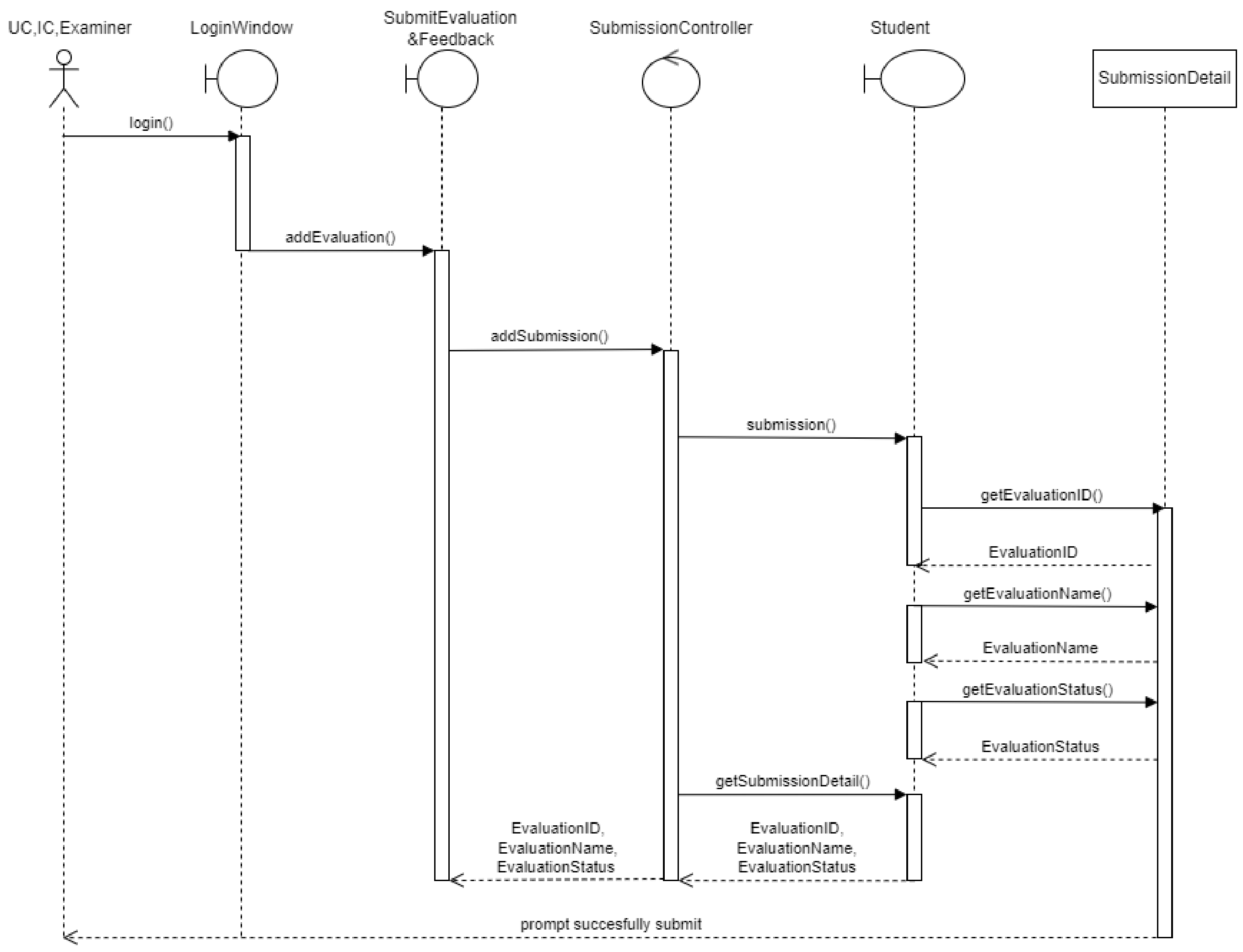
Method Name	addEvaluation()
Input	● Marks ● Evaluator
Output	● Marks ● EvaluationID ● EvaluationName ● EvaluationStatus
Algorithm	1. Start 2. Navigate to browse form page 3. Choose evaluate form 4. Key in evaluation based on student project and rubric 5. Submit the evaluation by clicking on the “submit” button 6. Prompt “Successful Submit” message, evaluation id and evaluation status upon successful file submission. 7. The system will notify the student once evaluation successfully uploads 8. Stop

Entity Name	ProvideFeedback
Method Name	addFeedback()
Input	● Comments
Output	● Comments ● FeedbackID ● FeedbackName ● FeedbackStatus
Algorithm	1. Start 2. Navigate to browse form page 3. Choose feedback form 4. Key in feedback based on student project 5. Submit the feedback by clicking on the “submit” button 6. Prompt “Successful Submit” message, feedback id and feedback status upon successful file submission. 7. The system will notify the student once feedback successfully uploads. 8. Stop

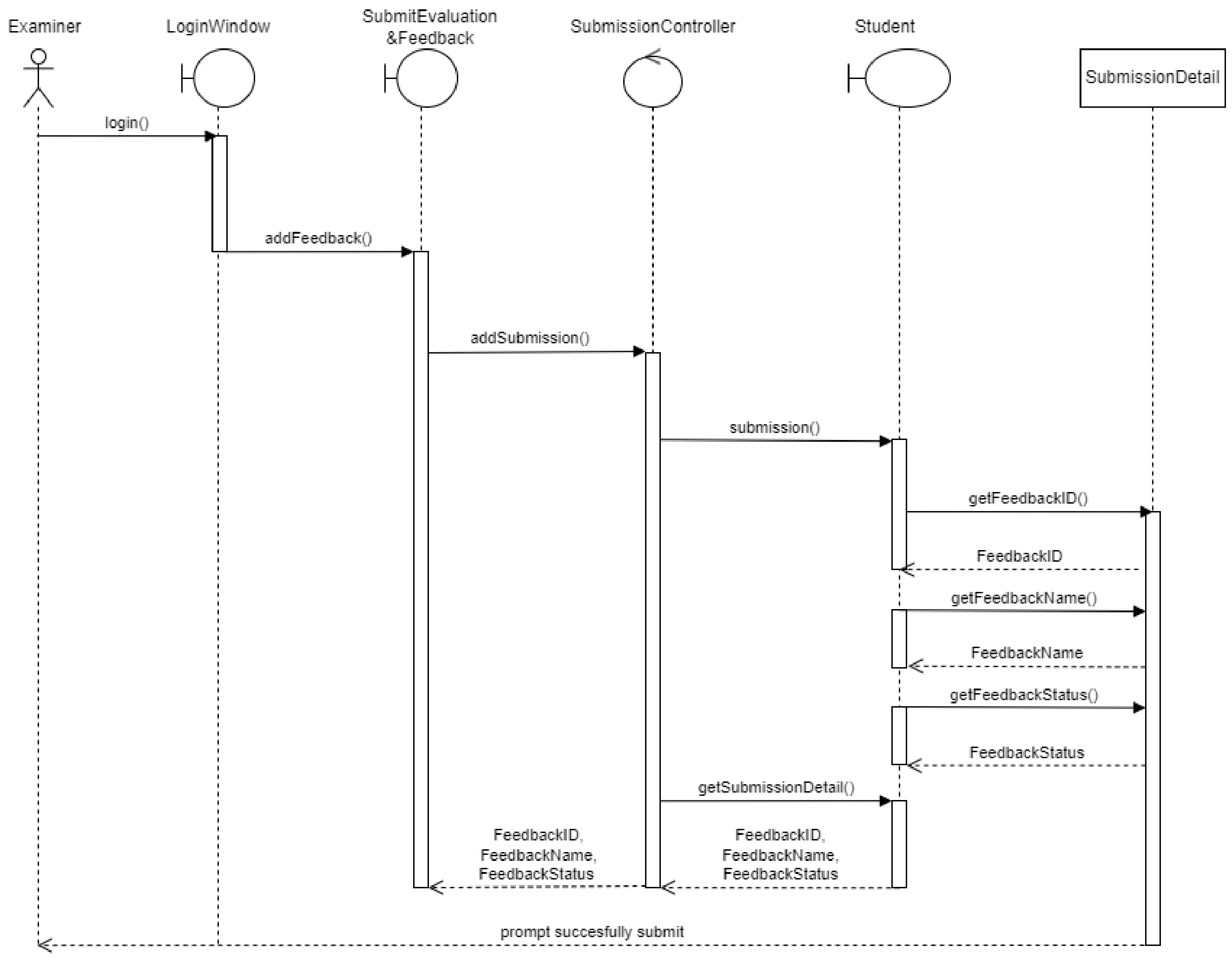
Entity Name	CheckingProjectProgress

Method Name	addProgressComment()
Input	● progressComment ● Evaluator
Output	● Comments ● Marks ● Evaluation ● Feedback
Algorithm	1. Start 2. Navigate to list project page 3. Choose the project form the list of student project 4. The system will display the project 5. The user can start checking the progress of the project 6. Stop

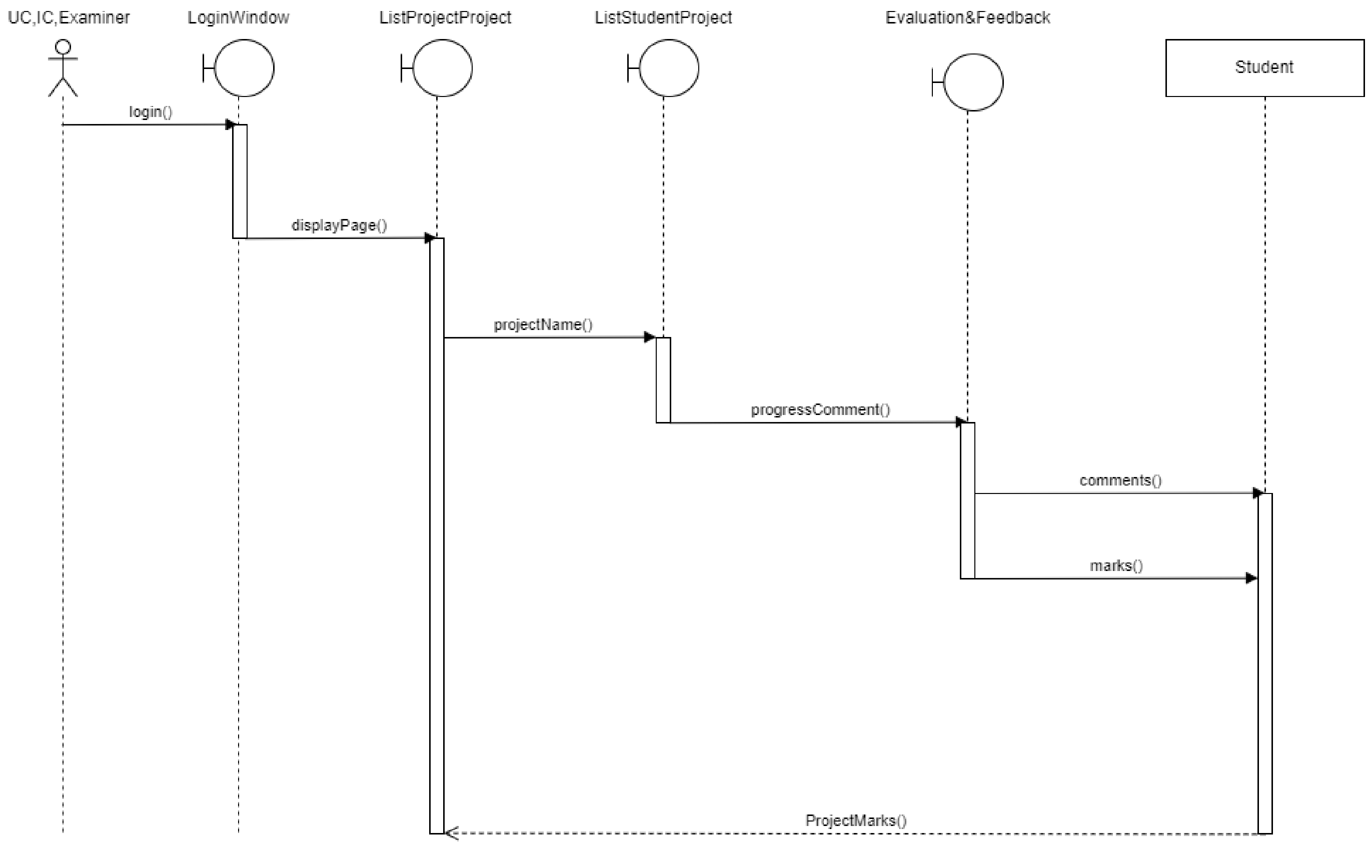
4.2.3.2. Sequence Diagram



SD009: Sequence diagram for Evaluate Student Work



SD010: Sequence diagram for Provide Feedback



SD011: Sequence diagram for Checking Project Progress

4.4. P004: <Coordinator> Subsystem

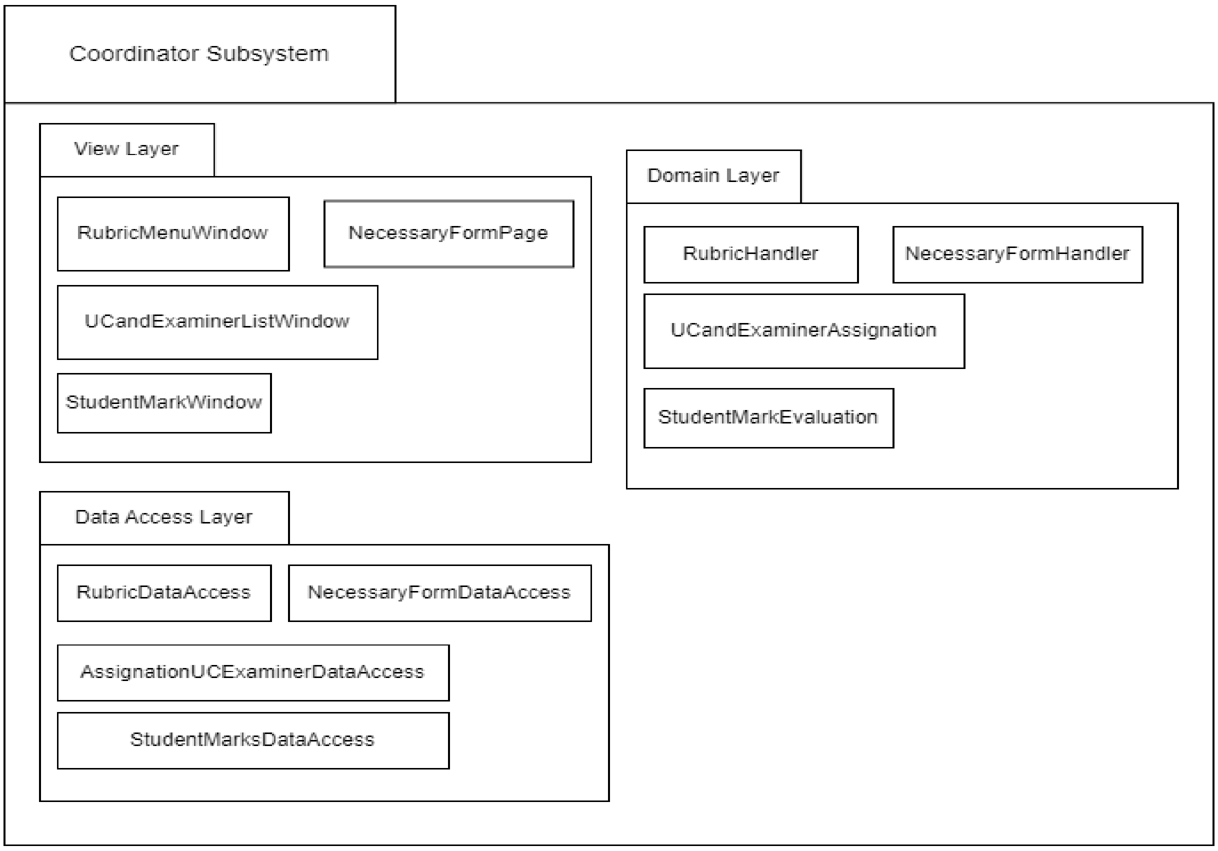
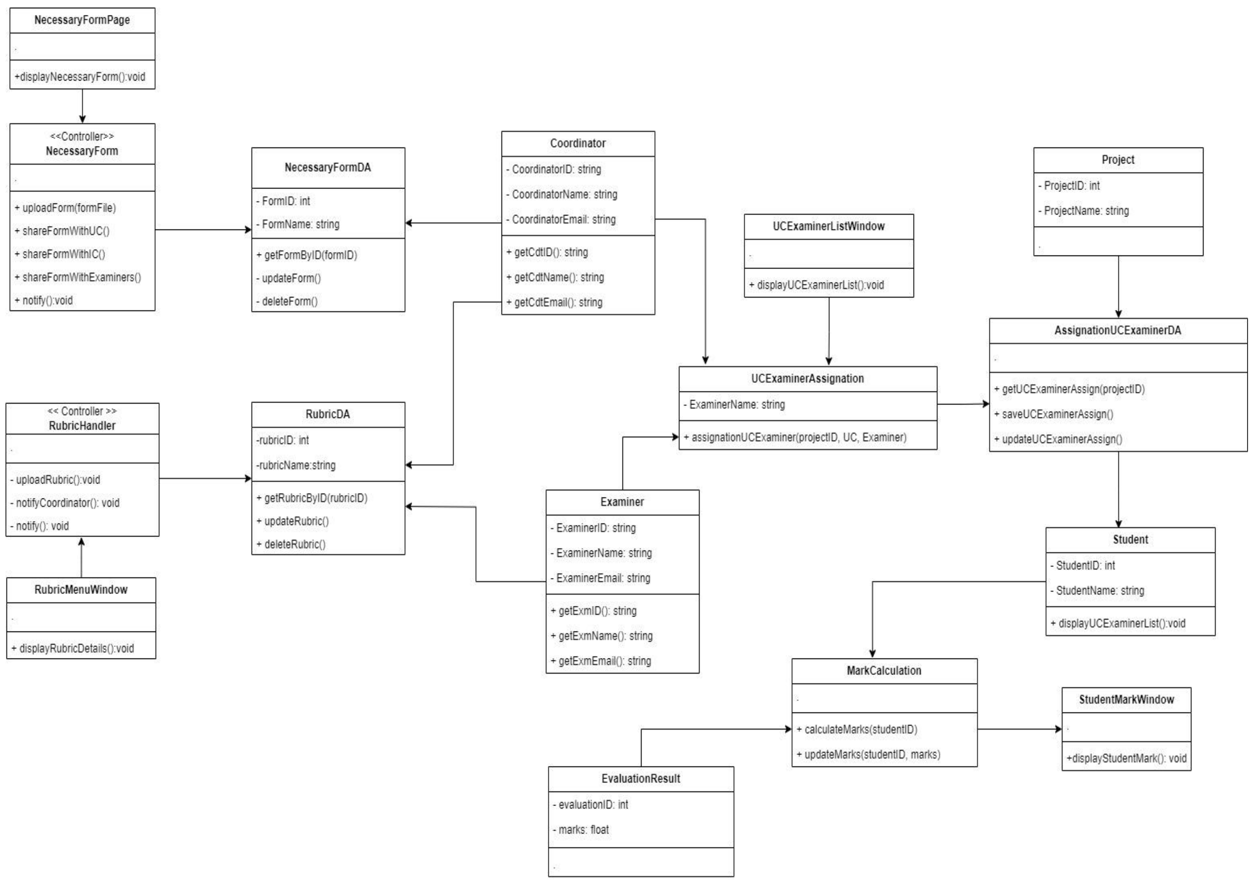


Figure 4.4: Package Diagram for <Coordinator> Subsystem

4.2.4.1. Class Diagram



Entity Name	ProvideNecessaryForm
Method Name	addNecessaryForm()
Input	● Form title ● Form file
Output	● Form ID ● Form status
Algorithm	1. Start 2. Navigate to the browse Necessary Form page. 3. Choose the necessary form to be provided. 4. Collect the form title and form file from the coordinator. 5. Store the form file in the system's database. 6. Associate the form title and file with a unique form identifier. 7. Save the form details in the system.

	8. Notify UC, IC, examiners about the availability of the new form. 9. Prompt a "Successful Submit" message, form ID, and form status upon successful file submission. 10. The system will notify the students and examiners once the form has been successfully uploaded. 11. Stop

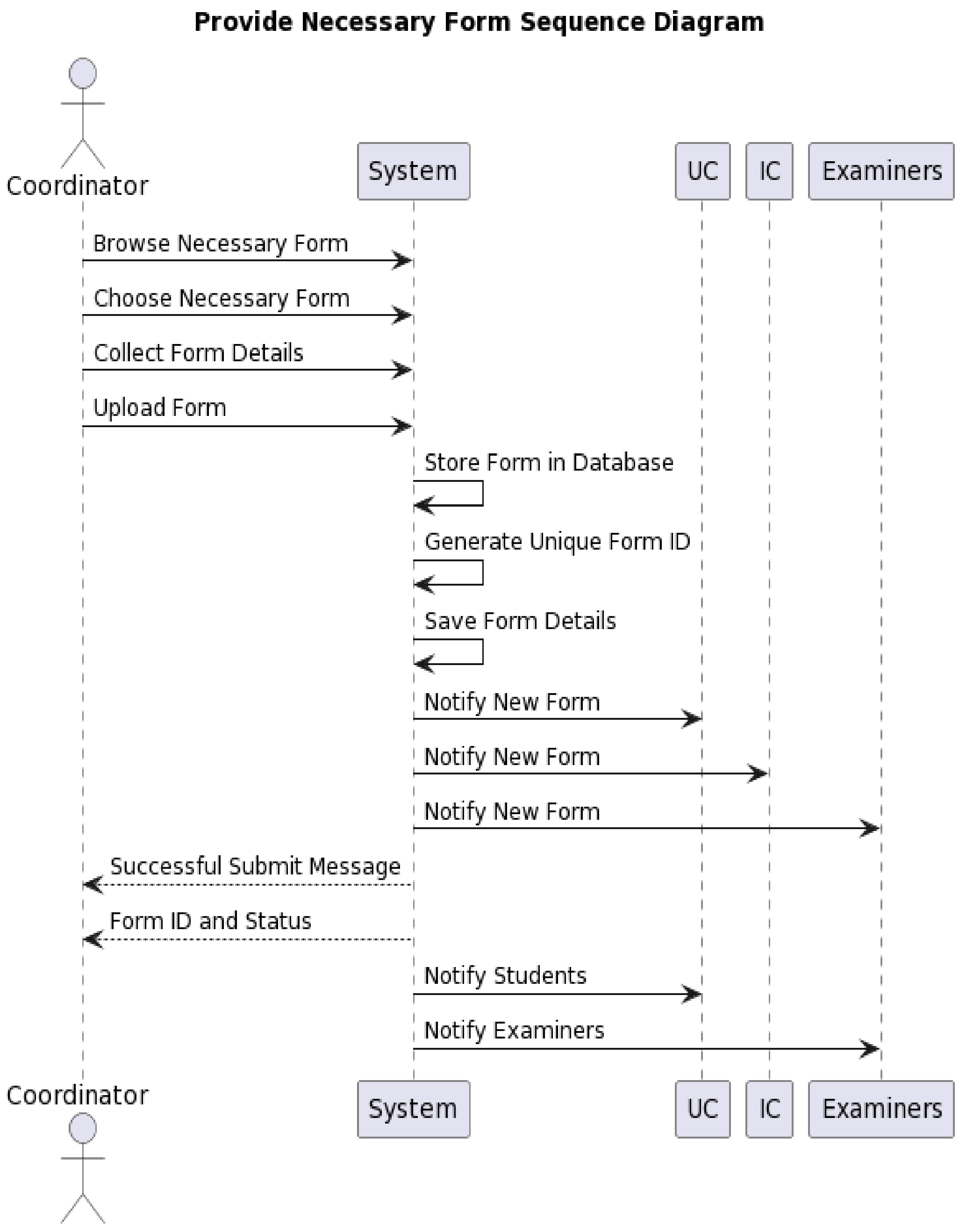
Entity Name	ProvideRubrics
Method Name	addRubrics()
Input	● Rubric title ● Rubric file
Output	● Rubric ID
Algorithm	1. Start 2. Navigate to the browse rubrics page. 3. Choose the rubric to be provided. 4. Submit the rubric by clicking on the "Submit" button. 5. Prompt a "Successful Submit" message, rubric ID, and rubric status upon successful submission. 6. Save the rubric details in the system. 7. Notify students, examiners about the new rubric availability. 8. The system will notify the students and examiners once the rubric has been successfully uploaded. 9. Stop

Entity Name	AssignUCExaminer
Method Name	assignUCExaminer()
Input	● ProjectID ● List of available UCs and examiners ● Selected UC ● Selected examiner
Output	● Updated project record ● Notification to the assigned UC and examiner

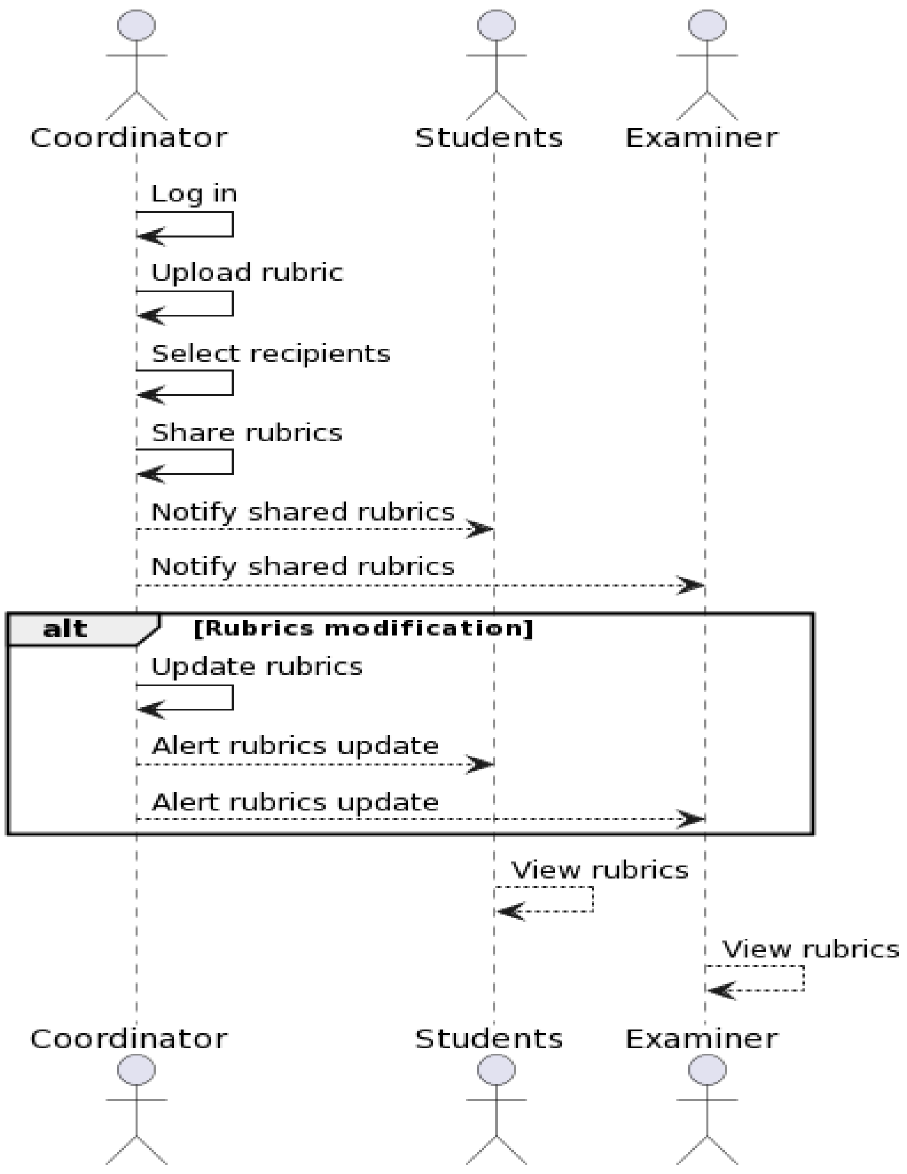
Algorithm	1. Start 2. Navigate to the Assignation UC Examiner page. 3. Display the student's project details and the list of available UCs and examiners. 4. Coordinator select a UC and an examiner for the student's project. 5. Update the project record with the assigned UC and examiner. 6. Save the updated project details in the system. 7. Notify the assigned UC and examiner about the project assignment. 8. Prompt a confirmation message to the coordinator upon successful assignation. 9. Stop

Entity Name	CalculateStudentMarks
Method Name	calculateMarks()
Input	● Student ID ● Project evaluation results
Output	● Calculated total marks ● Updated marks for the student's project ● Notification to the student regarding final marks ● Confirmation message to the coordinator
Algorithm	1. Start 2. Navigate to the mark calculation page. 3. Obtain the student's project evaluation results based on the Student ID. 4. Calculate the total marks based on the assigned scores for each criterion. 5. Store the calculated marks in the system's database. 6. Save the updated marks for the student's project. 7. Notify the student about their final marks for the project. 8. Prompt a confirmation message to the coordinator upon successful mark calculation. 9. Stop

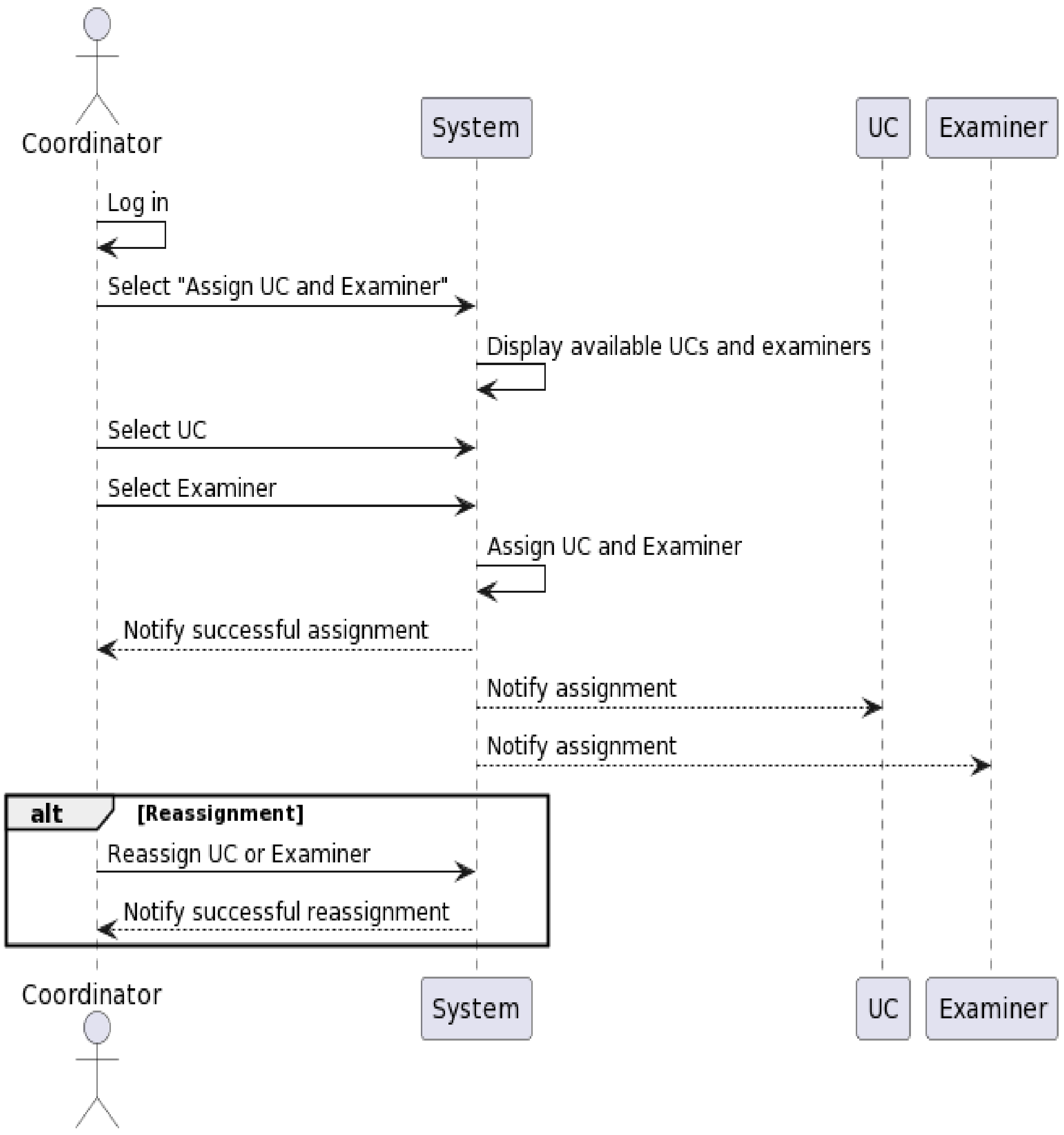
4.2.4.2. Sequence Diagram



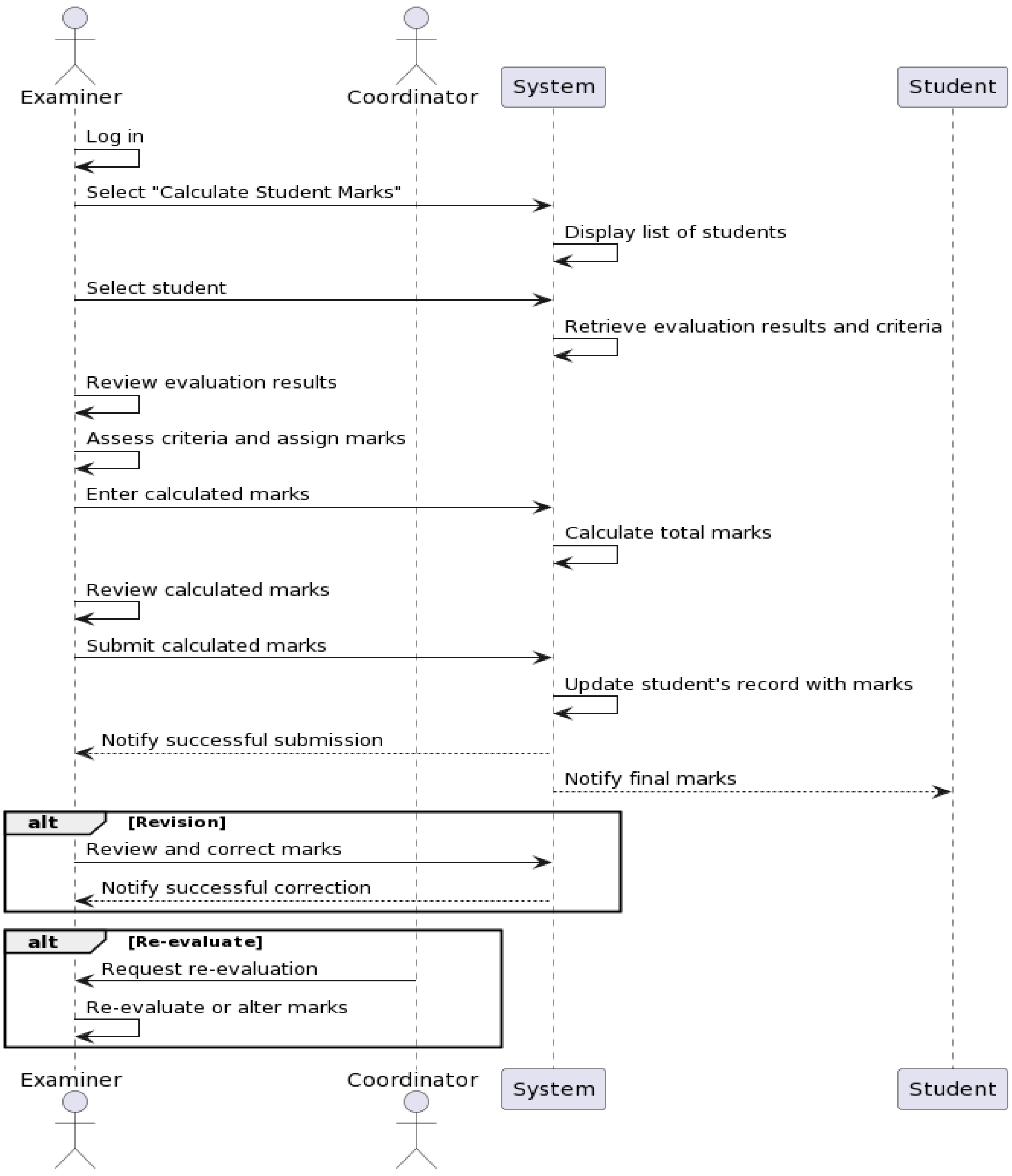
SD012: Sequence diagram for Provide Necessary Form



SD013: Sequence diagram for Provide Rubrics



SD014: Sequence diagram for Assign UC Examiner



SD015: Sequence diagram for Calculate Student Marks

5. Data Design

5.1. Data Description

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

Table 5.1: Description of Entities in the Database

No.	Entity Name	Description
1	Student	Represents the final year students who will be using the system to submit their project proposals online.
2	University Coach	Represents the university coaches who will use the system to check the progress of students, evaluate them, and generate reports based on student progress.
3	Industry Coach	Represents the industry coaches who will use the system to check the progress of students, evaluate them, and generate reports based on student progress.
4	Examiner	Represents the examiners who will use the system to evaluate the final year projects and provide feedback.
5	Coordinator	Represents the coordinator who will prepare rubrics for evaluation and evaluation forms for coaches and examiners to use.

5.2. Data Dictionary

5.2.1. Entity: <User>

Attribute Name	**Type**	**Description**
StudID	VARCHAR2	This attribute serves as a unique identifier for the Final Year Student entity. It is used to uniquely identify each student in the database. A primary key for students.
UCoachID	VARCHAR2	This attribute serves as a unique identifier for the University Coach entity. It is used to uniquely identify each university coach in the database.
ICoachID	VARCHAR2	This attribute serves as a unique identifier for the Industry Coach entity. It is used to uniquely identify each industry coach in the database.
ExaminerID	VARCHAR2	This attribute serves as a unique identifier for the Examiner entity. It is used to uniquely identify each examiner in the database.
CoordinatorID	VARCHAR2	This attribute serves as a unique identifier for the Coordinator entity. It is used to uniquely identify the coordinator in the database.
StudName	VARCHAR2	This attribute stores the name of the Final Year Student.
UCoachName	VARCHAR2	This attribute stores the name of the University Coach.
ICoachName	VARCHAR2	This attribute stores the name of the Industry Coach.
ExaminerName	VARCHAR2	This attribute stores the name of the Examiner.
CoordinatorName	VARCHAR2	This attribute stores the name of the Coordinator.
StudEmail	VARCHAR2	This attribute stores the email address of the Final Year Student.
UCoachEmail	VARCHAR2	This attribute stores the email address of the University Coach.
ICoachEmail	VARCHAR2	This attribute stores the email address of the Industry Coach.
ExaminerEmail	VARCHAR2	This attribute stores the email address of the Examiner.
CoordinatorEmail	VARCHAR2	This attribute stores the email address of the Coordinator.

5.2.2. Entity: <SubmitProjectProposal>

Attribute Name	Type	Description
proposalID	int	Uniquely identifies a member of Submit Project Proposal
proposalTitle	string	Title of the member of submit project proposal
proposalStatus	string	Status of the submitted project proposal

5.2.3. Entity: <DesignProjectSolution>

Attribute Name	Type	Description
designSolID	int	Uniquely identifies a member of Design Project Solution
designInfo	string	Information related to design solution

5.2.4. Entity: <SubmitDesignSolution>

Attribute Name	Type	Description
solutionID	int	Uniquely identifies a member of Submit Project Proposal
solutionTitle	string	Title of the member of submit project solution
solutionStatus	string	Status of the submitted project solution

5.2.5. Entity: <EvaluateStudentWork>

Attribute Name	Type	Description
evaluationID	int	Uniquely identifies a member of Submit Evaluation
evaluationName	string	Name of the member of evaluation
evaluationStatus	string	Status of the submitted evaluation

5.2.6. Entity: <ProvideFeedback>

Attribute Name	Type	Description
feedbackID	int	Uniquely identifies a member of Submit Feedback
feedbackName	string	Name of the member of feedback
feedbackStatus	string	Status of the submitted

5.2.7. Entity: <CheckingProjectProgress>

Attribute Name	Type	Description
progressComment	String	Update of the project progress
marksID	float	Uniquely identifies a member of Student Marks

5.2.8. Entity: <ProvideNecessaryForm>

Attribute Name	Type	Description
formID	int	Uniquely identifies a member of Necessary Form
formTitle	String	Title of the member of necessary form
formStatus	String	Status of the submitted necessary form

5.2.9. Entity: <ProvideRubrics>

Attribute Name	Type	Description
rubricID	int	Uniquely identifies a member of Rubrics
rubricFile	String	Status of the submitted rubric form

5.2.10. Entity: <AssignUCExaminer>

Attribute Name	Type	Description
UCExaminerAssign	String	Assignation of the member of UC and Examiner Assignation
updateUCExaminerAssign	String	Update of the assignation of the member of UC and Examiner Assignation

5.2.11. Entity: <CalculateStudentMarks>

Attribute Name	Type	Description
marksID	int	Uniquely identifies a member of Student Marks
evaluationID	int	Uniquely identifies a member of Student Evaluation

6. Requirements Traceability Matrix

In software development, a requirement traceability matrix is for the stakeholders to easily trace and track the progress of requirements through the identification of associated use cases, sequence diagrams, and test cases, ensuring that all requirements are properly implemented and tested in the software development process.

The table below shows how each package item (subsystem) relates to the use cases within that package. The use cases are further connected to their corresponding sequence diagrams, which depict the interaction between system components. Additionally, the test case IDs are provided, indicating which test cases are associated with each use case.

Table 6.1: Example of RTM for <GradX>

Package Item	Use Case ID	Use Case Description	Sequence Diagram ID	Sequence Diagram Description	Test Case ID
Package 1: User Subsystem	UC001	Login	SD-001 until SD-005	Login	TC-001
Package 2: Student Subsystem	UC0002	Project Proposal Submission	SD-006	Submit proposal	TC-002
	UC0003	Design Project Solution	SD-007	Design a project solution	TC-003
	UC004	Project Solution Submission	SD-008	Submit project solution	TC-004
Package 3: Assessment and Progress Tracking Subsystem	UC005	Checking Project Progress	SD-009	Checking project progress	TC-005
	UC006	Evaluate Student Work	SD-010	Evaluate student work	TC-006
	UC007	Provide Feedback	SD-011	Provide feedback	TC-007
	UC008	Prepare Necessary Form	SD-012	Provide necessary form	TC-008

Package 4: Coordinator Subsystem	UC009	Provide Rubrics	SD-013	Provide rubrics	TC-009
	UC010	Assign UC and Examiner	SD-014	Assign UC and Examiner	TC-010
	UC0011	Calculate Student Mark	SD-015	Calculate student marks	TC-011

7. Test Cases

Based on this section, it appears to be a test execution report for testing the login functionality of the GradX site. Here's an explanation of the different sections in the report:

Tester's Name: Specifies the names of the testers involved in executing the test cases. In this case, the testers are Kamilah, Puteri, and Malley.

Date Tested: Indicates the date when the testing was performed. In this case, it is mentioned as 28-June-23 and forwarding date.

Test Case (Fail/Pass/Not): This column indicates the outcome of each test case execution. The possible values are "Fail" if the test case fails, "Pass" if it passes, "Not executed" if the test case was not executed, and "Suspended" if the test case execution was halted or suspended.

S#: This column represents the serial number or index of the test scenario or test case.

Prerequisites: This column lists the prerequisites or conditions that need to be fulfilled before executing the corresponding test case. In this case, it mentions prerequisites related to the availability of an internet connection and having valid login credentials for different roles (User, UCoach, ICoach, Examiner, and Coordinator).

Test Data: This column provides the test data or inputs used for the corresponding test case execution. It includes sample data such as StudID, stud2001 for test case #1, UCoachID, unicoach_31 for test case #2, ICoachID, comcoach_7 for test case #3, ExaminerID, exm99 for test case #4, and CoordinatorID, coord23 for test case #5.

Test Scenario: This section describes the specific test scenario being executed. In this case, the scenario is to verify whether a user can successfully log in by entering a valid username and password.

Step #: This column denotes the step number of the test case execution.

Step Details: This column provides the details of each step to be performed during the test case execution. It includes actions such as navigating to the GradX site, entering the username and password, and clicking the submit button.

Expected Results: This column describes the expected results or outcomes of each step. For example, the expected result of step #3 is that the user should be successfully logged in.

Actual Results: This column captures the actual results observed during the test case execution.

Pass/Fail/Not executed/Suspended: This column indicates the final outcome of each test step. It mentions whether the step passed, failed, was not executed, or was suspended.

The report provides an overview of the test execution progress and outcomes for the specified test cases. It helps track the success or failure of each test step and allows stakeholders to assess the quality and stability of the login functionality in the GradX site.

1.1 **TC001: Test <User> Subsystem: <Login (UC001)>**

This test contains the following test cases:

(a) TC001_01: Test <Scenario of sequence diagram1 until diagram 5 (SD001 -SD005)>

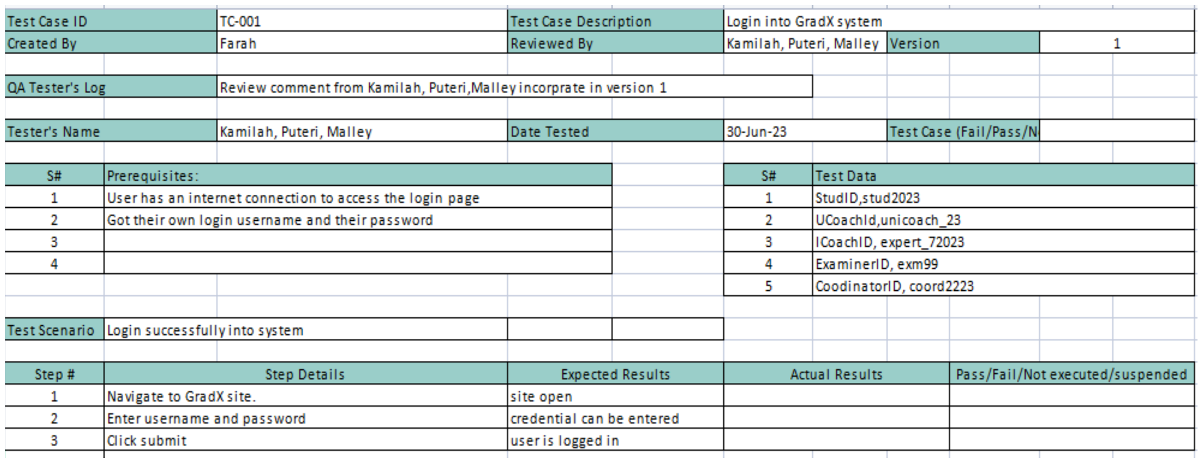
1.1.1 **TC001_01: Test <state scenario of sequence diagram1 (SD001)>**

This test contains the following scenarios:

(a) TC001_01: Test <normal scenario of sequence diagram1 until diagram5 (SD001-SD005)>

(b) TC001_02: Test <alternate scenario of sequence diagram1 until diagram5 (SD001-SD005)>

Table 7.1: TC001_01_01 - <Normal Scenario of sequence diagram1 (SD001 until diagram5 (SD005))>



Test Case ID	TC-001	Test Case Description	Login into GradX system		
Created By	Farah	Reviewed By	Kamilah, Puteri, Malley	Version	1
QA Tester's Log	Review comment from Kamilah, Puteri,Malley incorprate in version 1				
Tester's Name	Kamilah, Puteri, Malley	Date Tested	30-Jun-23	Test Case (Fail/Pass/N	

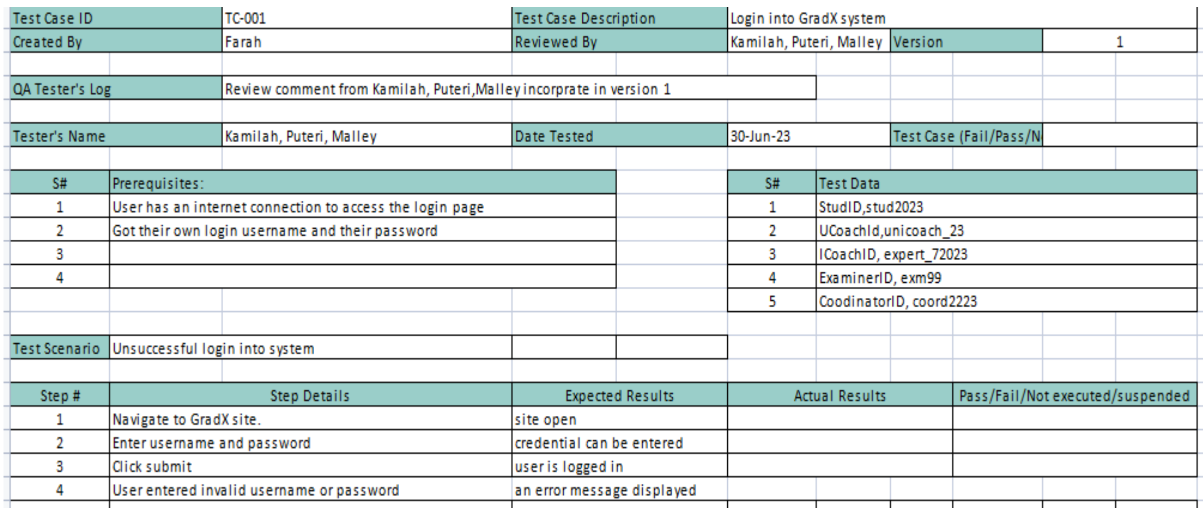
S#	Prerequisites:
1	User has an internet connection to access the login page
2	Got their own login username and their password
3	
4	

S#	Test Data
1	StudID,stud2023
2	UCoachId,unicoach_23
3	ICoachID, expert_72023
4	ExaminerID, exm99
5	CoodinatorID, coord2223

Test Scenario	Login successfully into system

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	Navigate to GradX site.	site open		
2	Enter username and password	credential can be entered		
3	Click submit	user is logged in		

Table 7.2: TC001_01_02 - <Alternate Scenario of sequence diagram1 (SD001 until diagram5 (SD005))>



Test Case ID	TC-001	Test Case Description	Login into GradX system		
Created By	Farah	Reviewed By	Kamilah, Puteri, Malley	Version	1
QA Tester's Log	Review comment from Kamilah, Puteri,Malley incorprate in version 1				
Tester's Name	Kamilah, Puteri, Malley	Date Tested	30-Jun-23	Test Case (Fail/Pass/N	

S#	Prerequisites:
1	User has an internet connection to access the login page
2	Got their own login username and their password
3	
4	

S#	Test Data
1	StudID,stud2023
2	UCoachId,unicoach_23
3	ICoachID, expert_72023
4	ExaminerID, exm99
5	CoodinatorID, coord2223

Test Scenario	Unsuccessful login into system

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	Navigate to GradX site.	site open		
2	Enter username and password	credential can be entered		
3	Click submit	user is logged in		
4	User entered invalid username or password	an error message displayed		

1.2 **TC002: Test <Student> Subsystem: <Project Proposal Submission (UC002)>**

This test contains the following test cases:

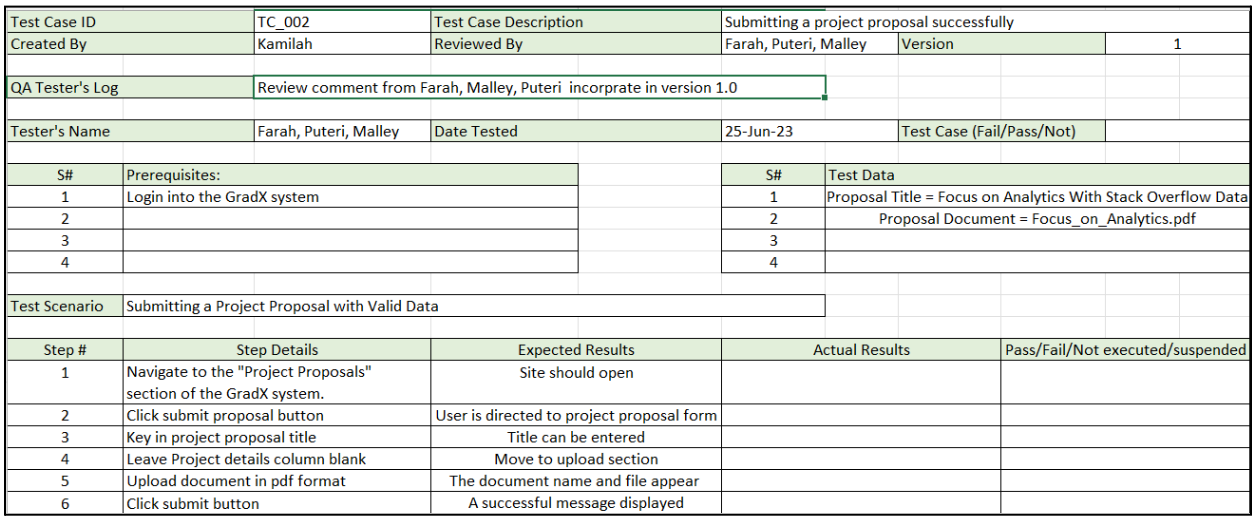
(a) TC002_01: Test <Scenario of sequence diagram1 (SD002)>

1.2.1 **TC002_01: Test <Submit Project Proposal (SD002)>**

This test contains the following scenarios:

(a) TC002_01_01: Test <normal scenario of sequence diagram1 (SD002)>

Table 7.3: TC002_01_01 - <Normal Scenario of sequence diagram1 (SD002)>



Test Case ID	TC_002	Test Case Description	Submitting a project proposal successfully		
Created By	Kamilah	Reviewed By	Farah, Puteri, Malley	Version	1
QA Tester's Log	Review comment from Farah, Malley, Puteri incorprate in version 1.0				
Tester's Name	Farah, Puteri, Malley	Date Tested	25-Jun-23	Test Case (Fail/Pass/Not)	
S#	Prerequisites:		S#	Test Data	
1	Login into the GradX system		1	Proposal Title = Focus on Analytics With Stack Overflow Data	
2			2	Proposal Document = Focus_on_Analytics.pdf	
3			3		
4			4		
Test Scenario	Submitting a Project Proposal with Valid Data				
Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended	
1	Navigate to the "Project Proposals" section of the GradX system.	Site should open			
2	Click submit proposal button	User is directed to project proposal form			
3	Key in project proposal title	Title can be entered			
4	Leave Project details column blank	Move to upload section			
5	Upload document in pdf format	The document name and file appear			
6	Click submit button	A successful message displayed			

1.3 **TC003: Test <Student> Subsystem: <Design Project Solution (UC003)>**

This test contains the following test cases:

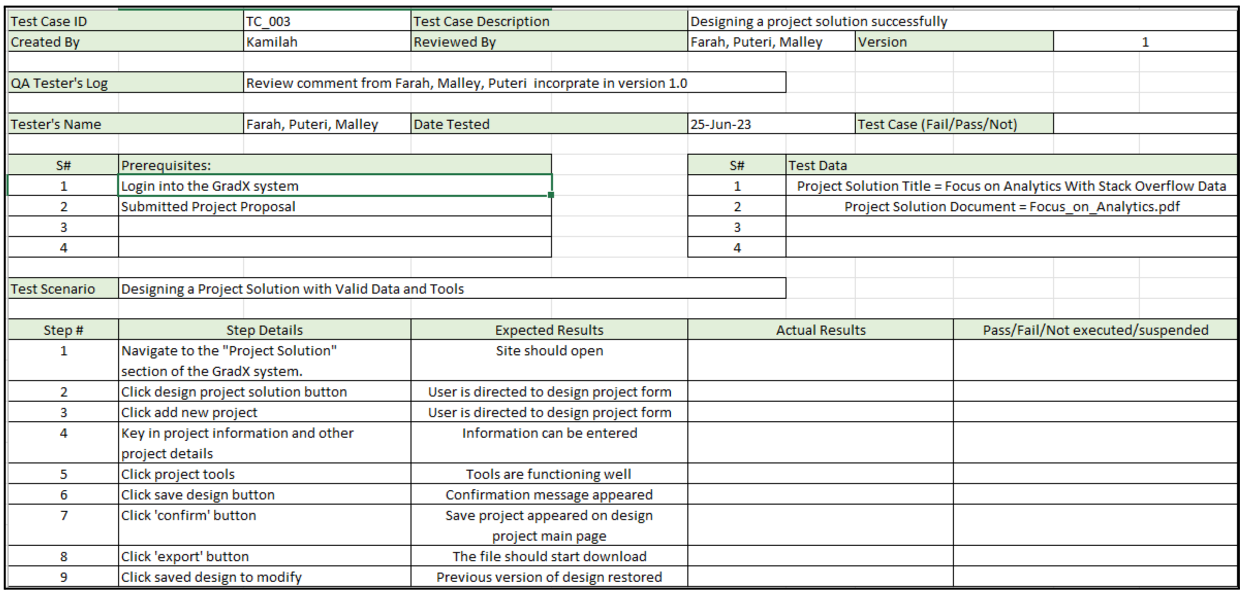
(a) TC003_01: Test <Scenario of sequence diagram1 (SD003)>

1.3.1 **TC003_01: Test <Design a Project Solution (SD003)>**

This test contains the following scenarios:

(c) TC003_01_01: Test <normal scenario of sequence diagram1 (SD003)>

Table 7.4: TC003_01_01 - <Normal Scenario of sequence diagram3 (SD003)>



Test Case ID	TC_003	Test Case Description	Designing a project solution successfully		
Created By	Kamilah	Reviewed By	Farah, Puteri, Malley	Version	1
QA Tester's Log	Review comment from Farah, Malley, Puteri incorprate in version 1.0				
Tester's Name	Farah, Puteri, Malley	Date Tested	25-Jun-23	Test Case (Fail/Pass/Not)	

S#	Prerequisites:
1	Login into the GradX system
2	Submitted Project Proposal
3	
4	

S#	Test Data
1	Project Solution Title = Focus on Analytics With Stack Overflow Data
2	Project Solution Document = Focus_on_Analytics.pdf
3	
4	

Test Scenario	Designing a Project Solution with Valid Data and Tools

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	Navigate to the "Project Solution" section of the GradX system.	Site should open		
2	Click design project solution button	User is directed to design project form		
3	Click add new project	User is directed to design project form		
4	Key in project information and other project details	Information can be entered		
5	Click project tools	Tools are functioning well		
6	Click save design button	Confirmation message appeared		
7	Click 'confirm' button	Save project appeared on design project main page		
8	Click 'export' button	The file should start download		
9	Click saved design to modify	Previous version of design restored		

1.4 **TC004: Test <Student> Subsystem: <Project Solution Submission (UC004)>**

This test contains the following test cases:

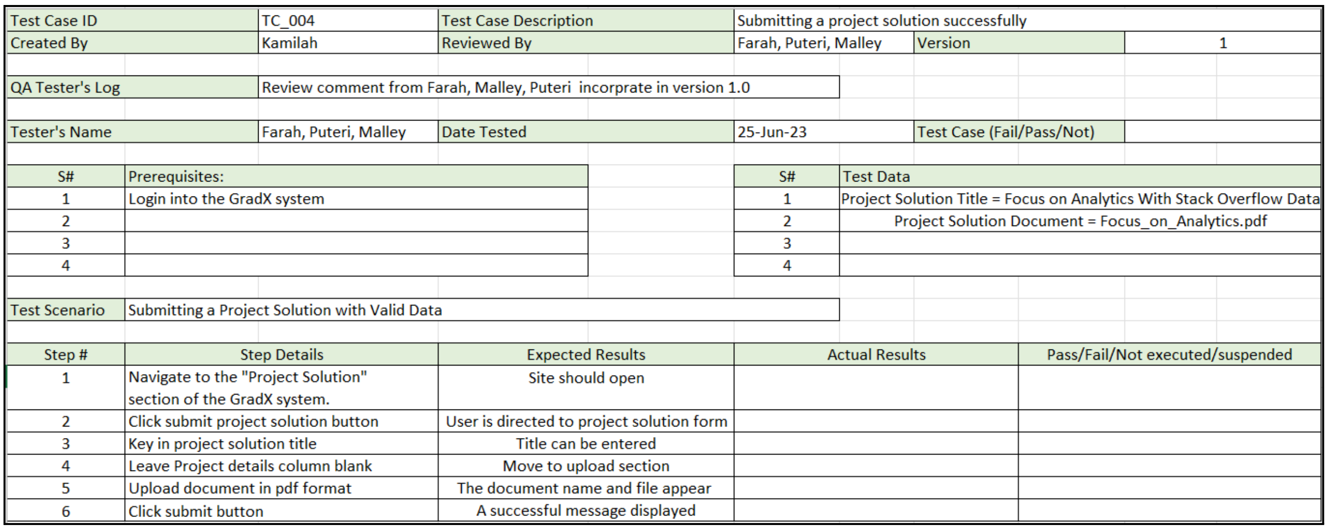
(b) TC004_01: Test <Scenario of sequence diagram1 (SD004)>

1.4.1 **TC004_01: Test < Submit Project Solution (SD001)>**

This test contains the following scenarios:

(d) TC004_01_01: Test <normal scenario of sequence diagram1 (SD004)>

Table 7.5: TC004_01_01 - <Normal Scenario of sequence diagram1 (SD004)>



Test Case ID	TC_004	Test Case Description	Submitting a project solution successfully		
Created By	Kamilah	Reviewed By	Farah, Puteri, Malley	Version	1
QA Tester's Log	Review comment from Farah, Malley, Puteri incorprate in version 1.0				
Tester's Name	Farah, Puteri, Malley	Date Tested	25-Jun-23	Test Case (Fail/Pass/Not)	

S#	Prerequisites:
1	Login into the GradX system
2	
3	
4	

S#	Test Data
1	Project Solution Title = Focus on Analytics With Stack Overflow Data
2	Project Solution Document = Focus_on_Analytics.pdf
3	
4	

Test Scenario	Submitting a Project Solution with Valid Data

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	Navigate to the "Project Solution" section of the GradX system.	Site should open		
2	Click submit project solution button	User is directed to project solution form		
3	Key in project solution title	Title can be entered		
4	Leave Project details column blank	Move to upload section		
5	Upload document in pdf format	The document name and file appear		
6	Click submit button	A successful message displayed		

1.5 **TC005: Test <Assessment and Progress Tracking> Subsystem: <Checking Project Progress (UC005)>**

This test contains the following test cases:

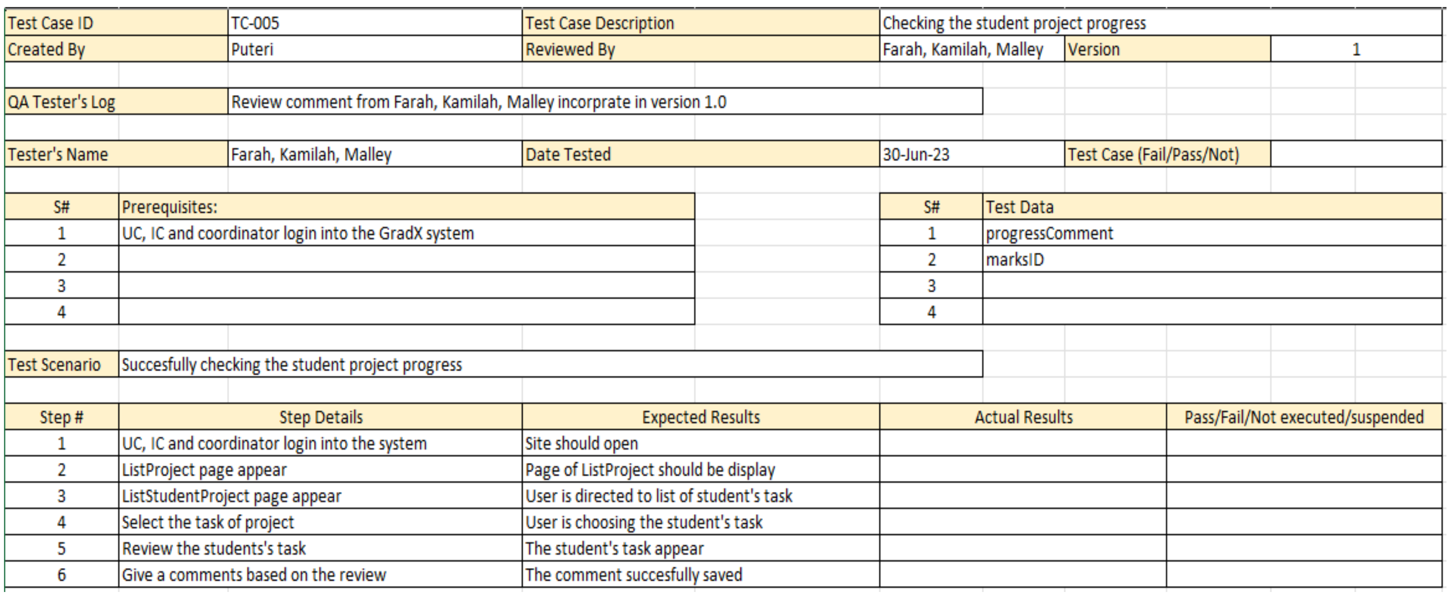
(c) TC005_01: Test <Scenario of sequence diagram9 (SD009)>

1.5.1 **TC001_01: Test <state scenario of sequence diagram9 (SD009)>**

This test contains the following scenarios:

(e) TC005_01_01: Test <normal scenario of sequence diagram9 (SD009)>

Table 7.6: TC005_01_01 - <Normal Scenario of sequence diagram9 (SD009)>



Test Case ID	TC-005	Test Case Description	Checking the student project progress		
Created By	Puteri	Reviewed By	Farah, Kamilah, Malley	Version	1

QA Tester's Log	Review comment from Farah, Kamilah, Malley incorprate in version 1.0

Tester's Name	Farah, Kamilah, Malley	Date Tested	30-Jun-23	Test Case (Fail/Pass/Not)	

S#	Prerequisites:
1	UC, IC and coordinator login into the GradX system
2	
3	
4	

S#	Test Data
1	progressComment
2	marksID
3	
4	

Test Scenario	Succesfully checking the student project progress

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	UC, IC and coordinator login into the system	Site should open		
2	ListProject page appear	Page of ListProject should be display		
3	ListStudentProject page appear	User is directed to list of student's task		
4	Select the task of project	User is choosing the student's task		
5	Review the students's task	The student's task appear		
6	Give a comments based on the review	The comment succesfully saved		

1.6 **TC006: Test <Assessment and Progress Tracking> Subsystem: <Evaluate Student Work (UC006)>**

This test contains the following test cases:

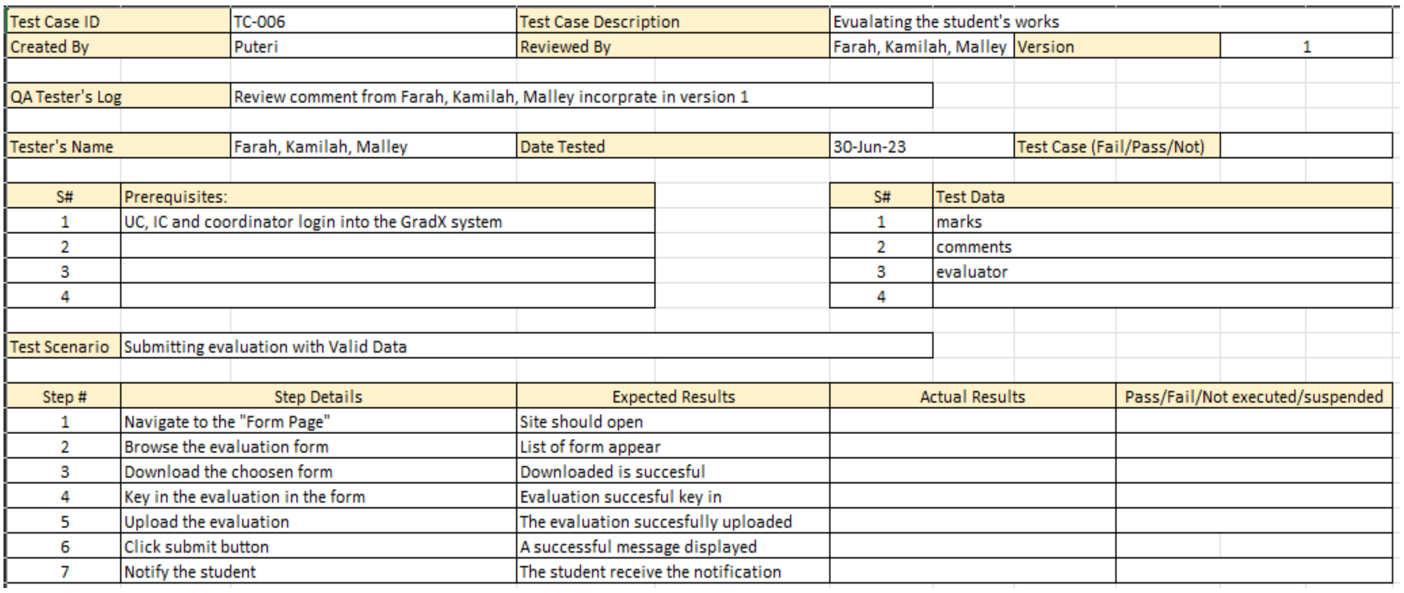
(d) TC006_01: Test <Scenario of sequence diagram10 (SD010)>

1.6.1 **TC006_01: Test <state scenario of sequence diagram1 (SD001)>**

This test contains the following scenarios:

(f) TC006_01_01: Test <normal scenario of sequence diagram10 (SD010)>

Table 7.7: TC006_01_01 - <Normal Scenario of sequence diagram10 (SD010)>



Test Case ID	TC-006	Test Case Description	Evualating the student's works		
Created By	Puteri	Reviewed By	Farah, Kamilah, Malley	Version	1
QA Tester's Log	Review comment from Farah, Kamilah, Malley incorprate in version 1				
Tester's Name	Farah, Kamilah, Malley	Date Tested	30-Jun-23	Test Case (Fail/Pass/Not)	

S#	Prerequisites:
1	UC, IC and coordinator login into the GradX system
2	
3	
4	

S#	Test Data
1	marks
2	comments
3	evaluator
4	

Test Scenario	Submitting evaluation with Valid Data

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	Navigate to the "Form Page"	Site should open		
2	Browse the evaluation form	List of form appear		
3	Download the choosen form	Downloaded is succesful		
4	Key in the evaluation in the form	Evaluation succesful key in		
5	Upload the evaluation	The evaluation succesfully uploaded		
6	Click submit button	A successful message displayed		
7	Notify the student	The student receive the notification		

1.7 TC007: Test <Assessment and Progress Tracking> Subsystem: <Provide Feedback (UC007)>

This test contains the following test cases:

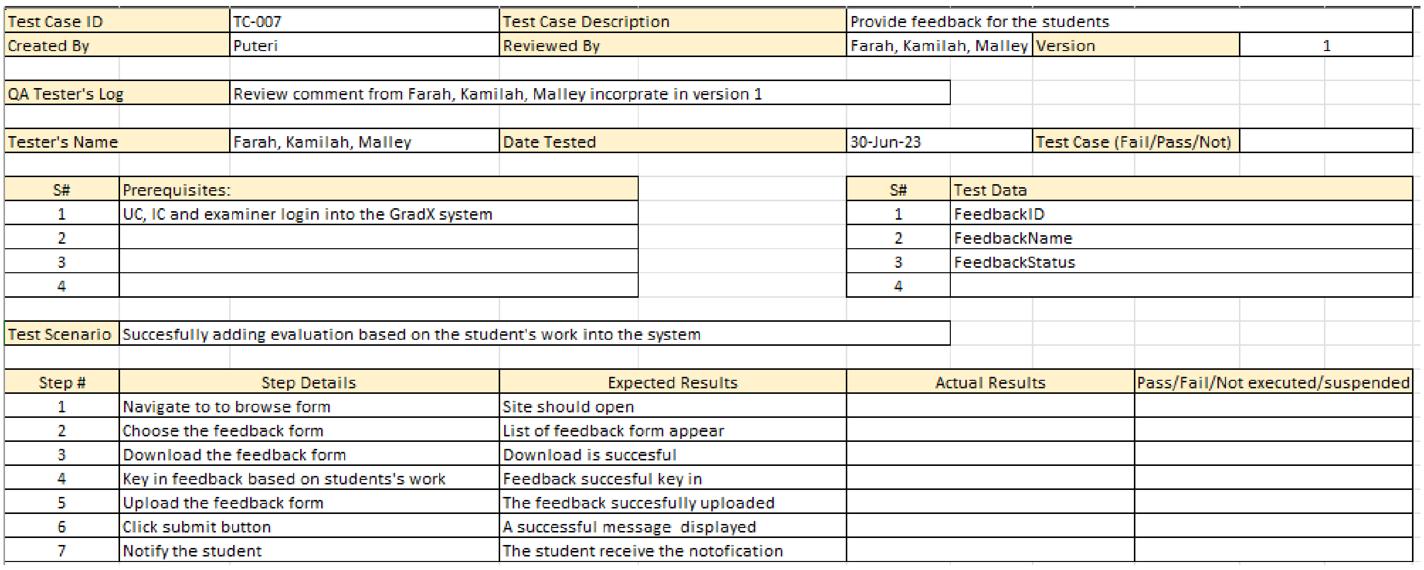
(e) TC007_01: Test <Scenario of sequence diagram11 (SD011)>

1.7.1 TC007_01: Test <state scenario of sequence diagram1 (SD011)>

This test contains the following scenarios:

(g) TC007_01_01: Test <normal scenario of sequence diagram11 (SD011)>

Table 7.8: TC007_01_01 - <Normal Scenario of sequence diagram11 (SD011)>



Test Case ID	TC-007	Test Case Description	Provide feedback for the students		
Created By	Puteri	Reviewed By	Farah, Kamilah, Malley	Version	1
QA Tester's Log	Review comment from Farah, Kamilah, Malley incorprate in version 1				
Tester's Name	Farah, Kamilah, Malley	Date Tested	30-Jun-23	Test Case (Fail/Pass/Not)	

S#	Prerequisites:	S#	Test Data
1	UC, IC and examiner login into the GradX system	1	FeedbackID
2		2	FeedbackName
3		3	FeedbackStatus
4		4	

Test Scenario	Succesfully adding evaluation based on the student's work into the system

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	Navigate to to browse form	Site should open		
2	Choose the feedback form	List of feedback form appear		
3	Download the feedback form	Download is succesful		
4	Key in feedback based on students's work	Feedback succesful key in		
5	Upload the feedback form	The feedback succesfully uploaded		
6	Click submit button	A successful message displayed		
7	Notify the student	The student receive the notofication		

1.8 **TC009: Test <Coordinator> Subsystem: <Provide Rubrics (UC009)>**

This test contains the following test cases:

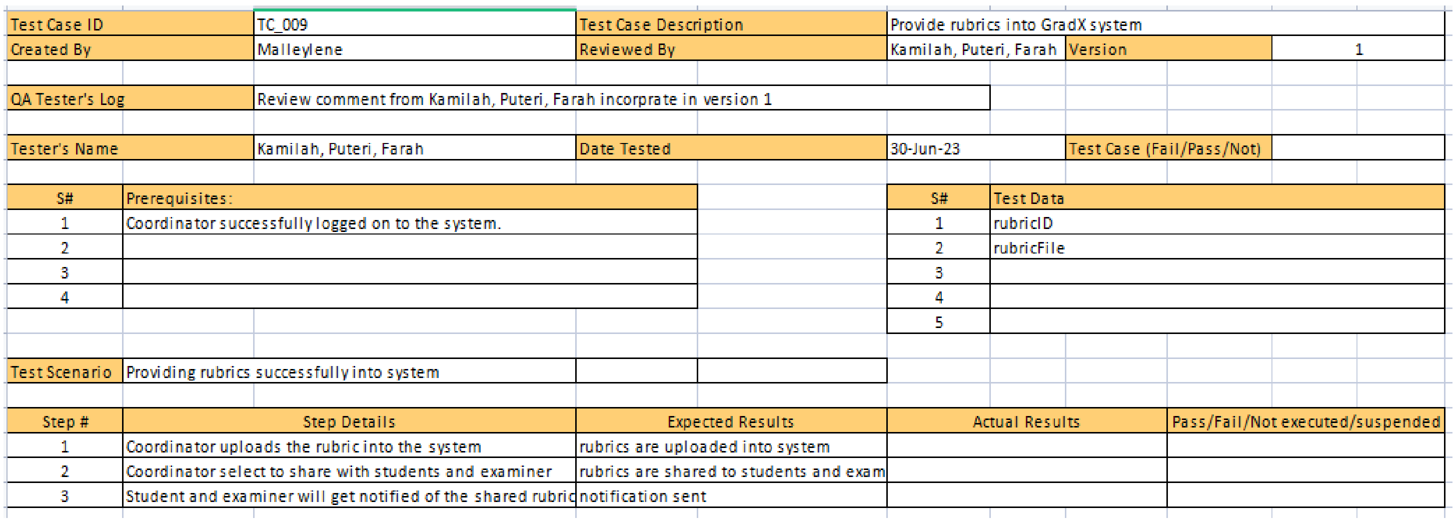
(f) TC009_01: Test <Scenario of sequence diagram9 (SD013)>

1.8.1 **TC009_01: Test <state scenario of sequence diagram9 (SD013)>**

This test contains the following scenarios:

(h) TC009_01_01: Test <normal scenario of sequence diagram9 (SD013)>

Table 7.9: TC009_01_01 - <Normal Scenario of sequence diagram9 (SD013)>



Test Case ID	TC_009	Test Case Description	Provide rubrics into GradX system		
Created By	Malleylene	Reviewed By	Kamilah, Puteri, Farah	Version	1
QA Tester's Log	Review comment from Kamilah, Puteri, Farah incorprate in version 1				
Tester's Name	Kamilah, Puteri, Farah	Date Tested	30-Jun-23	Test Case (Fail/Pass/Not)	

S#	Prerequisites:
1	Coordinator successfully logged on to the system.
2	
3	
4	

S#	Test Data
1	rubricID
2	rubricFile
3	
4	
5	

Test Scenario	Providing rubrics successfully into system

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	Coordinator uploads the rubric into the system	rubrics are uploaded into system		
2	Coordinator select to share with students and examiner	rubrics are shared to students and exam		
3	Student and examiner will get notified of the shared rubric	notification sent		

1.9 **TC010: Test <Coordinator> Subsystem: <Assign UC Examiner (UC010)>**

This test contains the following test cases:

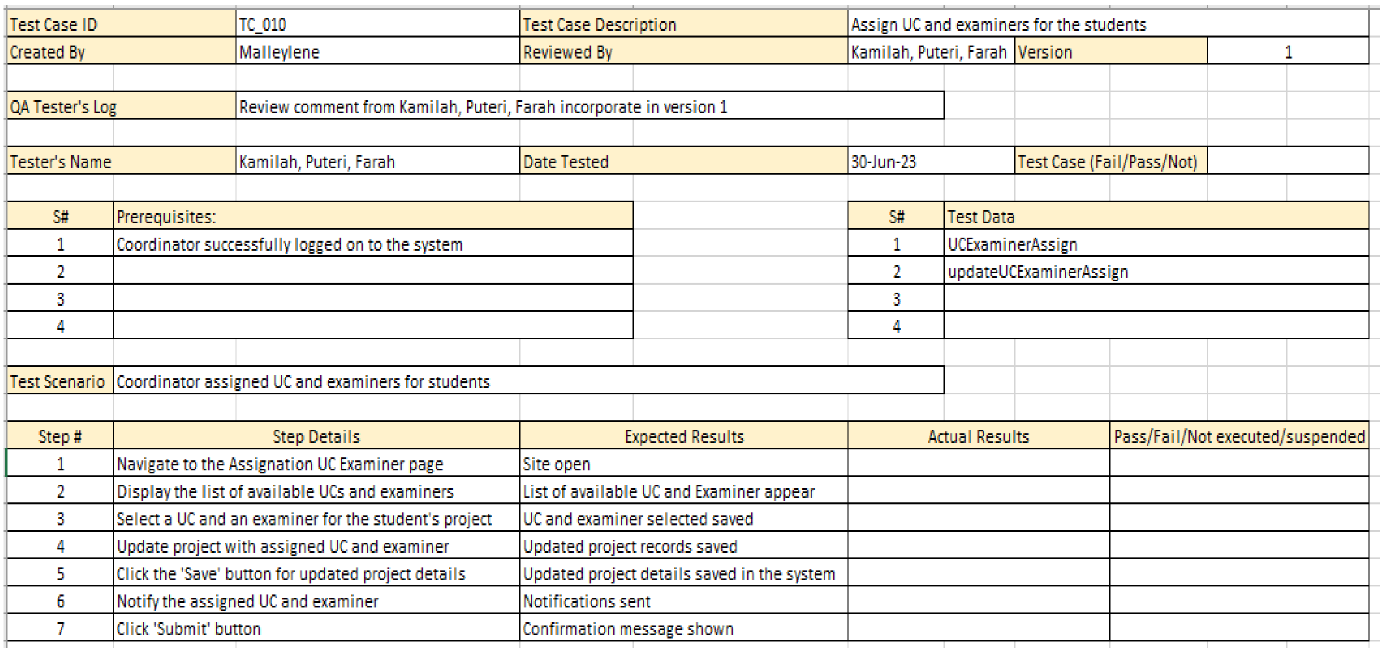
(g) TC010_01: Test <Scenario of sequence diagram14 (SD014)>

1.9.1 **TC010_01: Test <state scenario of sequence diagram14 (SD014)>**

This test contains the following scenarios:

(i) TC0010_01_01: Test <normal scenario of sequence diagram14 (SD014)>

Table 7.10: TC010_01_01 - <Normal Scenario of sequence diagram14 (SD014)>



Test Case ID	TC_010	Test Case Description	Assign UC and examiners for the students		
Created By	Malleylene	Reviewed By	Kamilah, Puteri, Farah	Version	1
QA Tester's Log	Review comment from Kamilah, Puteri, Farah incorporate in version 1				
Tester's Name	Kamilah, Puteri, Farah	Date Tested	30-Jun-23	Test Case (Fail/Pass/Not)	

S#	Prerequisites:	S#	Test Data
1	Coordinator successfully logged on to the system	1	UCExaminerAssign
2		2	updateUCExaminerAssign
3		3	
4		4	

Test Scenario	Coordinator assigned UC and examiners for students

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	Navigate to the Assignation UC Examiner page	Site open		
2	Display the list of available UCs and examiners	List of available UC and Examiner appear		
3	Select a UC and an examiner for the student's project	UC and examiner selected saved		
4	Update project with assigned UC and examiner	Updated project records saved		
5	Click the 'Save' button for updated project details	Updated project details saved in the system		
6	Notify the assigned UC and examiner	Notifications sent		
7	Click 'Submit' button	Confirmation message shown		

1.10 TC011: Test <Coordinator> Subsystem: <Calculate Student Marks (UC011)>

This test contains the following test cases:

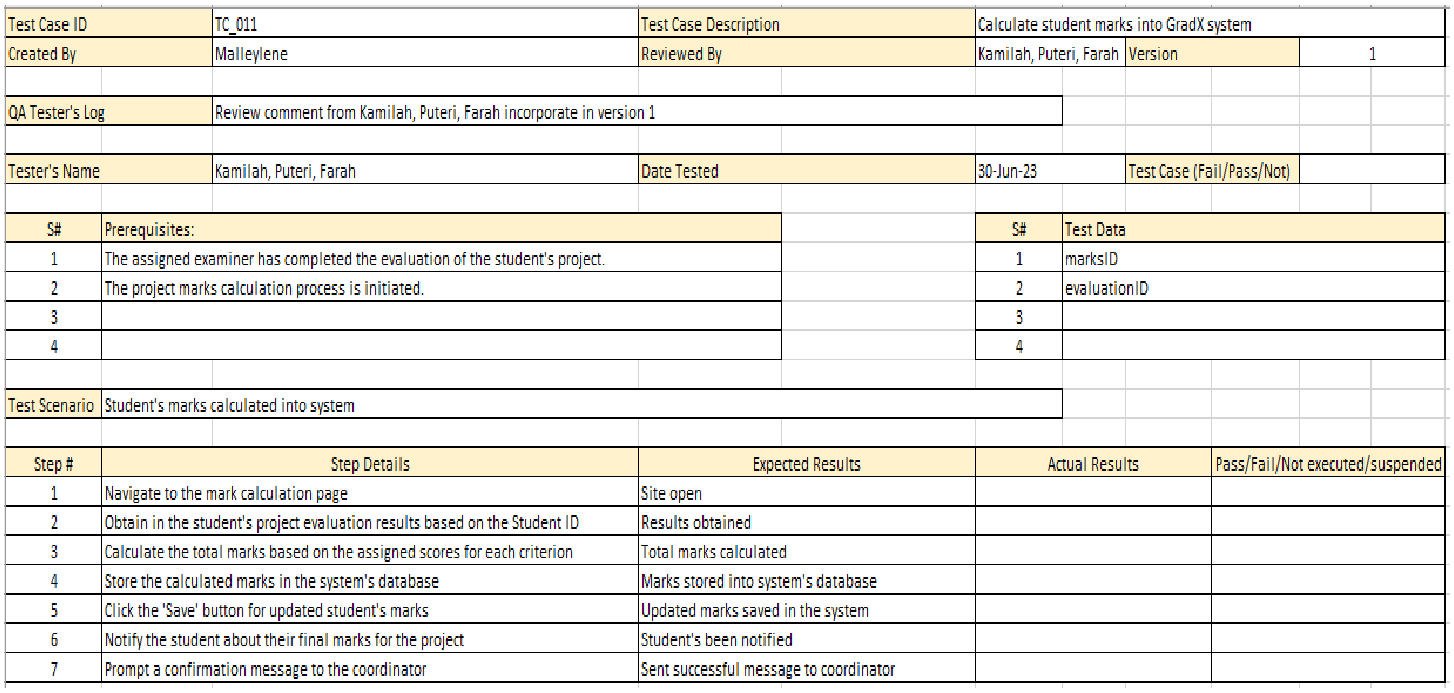
(h) TC011_01: Test <Scenario of sequence diagram15 (SD015)>

1.10.1 TC011_01: Test <state scenario of sequence diagram15 (SD015)>

This test contains the following scenarios:

(j) TC011_01_01: Test <normal scenario of sequence diagram15 (SD015)>

Table 7.11: TC011_01_01 - <Normal Scenario of sequence diagram15 (SD015)>



Test Case ID	TC_011	Test Case Description	Calculate student marks into GradX system		
Created By	Malleylene	Reviewed By	Kamilah, Puteri, Farah	Version	1

QA Tester's Log	Review comment from Kamilah, Puteri, Farah incorporate in version 1

Tester's Name	Kamilah, Puteri, Farah	Date Tested	30-Jun-23	Test Case (Fail/Pass/Not)	

S#	Prerequisites:
1	The assigned examiner has completed the evaluation of the student's project.
2	The project marks calculation process is initiated.
3	
4	

S#	Test Data
1	marksID
2	evaluationID
3	
4	

Test Scenario	Student's marks calculated into system

Step #	Step Details	Expected Results	Actual Results	Pass/Fail/Not executed/suspended
1	Navigate to the mark calculation page	Site open		
2	Obtain in the student's project evaluation results based on the Student ID	Results obtained		
3	Calculate the total marks based on the assigned scores for each criterion	Total marks calculated		
4	Store the calculated marks in the system's database	Marks stored into system's database		
5	Click the 'Save' button for updated student's marks	Updated marks saved in the system		
6	Notify the student about their final marks for the project	Student's been notified		
7	Prompt a confirmation message to the coordinator	Sent successful message to coordinator		