



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

SECJ2203: Software Engineering

System Documentation (SD)

2u2i Final Year Project with Industry (FYP-I)
Management System

Version 2.0

16 June 2022

School of Computing, Faculty of Engineering

Prepared by: Inferno

Name of Members:

1. Amir Iskandar Bin Norkhairulazaddin
2. Iman Ehsan Bin Hassan
3. Muhammad Aiman Bin Abdul Razak
4. Shahril Bin Saiful Bahri

Revision Page

a. Overview

In this system documentation version 1.0, it consists of the documentation components that are adapted from IEEE Recommended Practice for Software Requirements Specification, Software Design Description and Software Test Documentation that has been simplified and customized. In this version, it will cover all of the sections from the introduction until the specific requirements.

b. Target Audience

- Stakeholder
- Development Team
- Users

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
Shahril Bin Saiful Bahri (A20EC0144)	Team Leader	3.2 Component Model 4.2.3 Detail Description of <Notification> subsystem 6.2 Overall Interface Design	Completed
Amir Iskandar Bin Norkhairulazaddin (A20EC0011)	Team Member	4.1 Complete Package Diagram 4.2.4 Detail Description of <Assessment> subsystem 5.1 Data Description	Completed
Iman Ehsan Bin Hassan (A20EC0048)	Team Member	3.1 Architectural Style and Rationale 4.2.1 Detail Description of <Login> subsystem	Completed

		6.1 Overview of Interface	
Muhammad Aiman Bin Abdul Razak (A20EC0082)	Team Member	4.2.2 Detail Description of <Communication> subsystem 5.2 Data Dictionary	Completed

d. **Version Control History**

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Shahril Bin Saiful Bahri (Team Leader)	Completed Chapter 1 and 2, Section 1.1 to 1.5 and 2.1 to 2.5	05/06/2022
2.0	Amir Iskandar Bin Norkhairulazaddin (Team Leader)	Completed Chapter 3 until 6, Section 3.1 to 6.1	16/06/2022

Note:

This System Documentation (SD) template is adapted from IEEE Recommended Practice for Software Requirements Specification (SRS) (IEEE Std. 830-1998), Software Design Descriptions (SDD) (IEEE Std. 1016-1998 1), and Software Test Documentation (IEEE Std. 829-2008) that are simplified and customized to meet the need of SECJ2203 course at School of Computing, UTM. Examples of models are from Arlow and Neustadt (2002) and other sources stated accordingly.

Table of Contents

1	Introduction		3 - 5
	1.1	Purpose	3
	1.2	Scope	3, 4
	1.3	Definitions, Acronyms and Abbreviations	4
	1.4	References	4
	1.5	Overview	4, 5
2	Specific Requirements		6 - 32
	2.1	External Interface Requirements	6 - 8
		2.1.1 User Interfaces	6, 7
		2.1.2 Hardware Interfaces	7
		2.1.3 Software Interfaces	7, 8
		2.1.4 Communication Interfaces	8
	2.2	System Features	8 - 31
		2.2.1 UC001: Use Case <Login>	12, 13
		2.2.2 UC002: Use Case <Verification>	14, 15
		2.2.3 UC003: Use Case <View Message>	15, 16
		2.2.4 UC004: Use Case <Add Message>	16, 17
		2.2.5 UC005: Use Case <Delete Message>	17, 18
		2.2.6 UC006: Use Case <Edit Message>	18, 19
		2.2.7 UC007: Use Case <Notification Data>	19, 20
		2.2.8 UC008: Use Case <View Notification>	20, 21
		2.2.9 UC009: Use Case <Add Notification>	21, 22
		2.2.10 UC010: Use Case <Update Notification>	22, 23
		2.2.11 UC011: Use Case <Delete Notification>	23, 24
		2.2.12 UC012: Use Case <Return Message>	24, 25

		2.2.13	UC013: Use Case <Upload Question>	25, 26, 27
		2.2.14	UC014: Use Case <Take Exam>	27, 28, 29
		2.2.15	UC015: Use Case <View Grades>	29, 30, 31
	2.3		Performance and Other Requirements	31
	2.4		Design Constraints	31, 32
	2.5		Software System Attributes	32
3	System Architectural Design			33-34
	3.1	Architectural Style and Rationale		33
	3.2	Component Model		34
4	Detailed Description of Components			35-61
	4.1	Complete Package Diagram		35
	4.2	Detailed Description		35-61
		4.2.1	P001: <Login> Subsystem	35-40
		4.2.2	P002: <Communication> Subsystem	40-47
		4.2.3	P003: <Notification> Subsystem	47-53
		4.2.4	P004: <Assessment> Subsystem	53-61
5	Data Design			62-66
	5.1	Data Description		62-63
	5.2	Data Dictionary		63-66
6	Interface Design			67-78
	6.1	Overview of Interface		67-78

1. Introduction

1.1 Purpose

This Software Documentation describes the proposed 2u2i Final Year Project with Industry (FYP-I) Management System in which several sections will be clarified. The sections that will be described for this software documentation are the specific requirements, detailed description of components and the requirements matrix. This document contains the system design that explains the architecture of the system. This also includes the complete description of the components used in the system. The data design is also included in this documentation that contains the data description and data dictionary. This documentation was intended for the stakeholder, development team and the system's user. This is to provide them with necessary information about the system that will help them to understand more about the function available in the system. The stakeholder can make a decision, approval or changes to the system based on the documentation. Other than that, the development team can use the documentation to show and monitor their progress in the development process. The audience of the documentation also can provide necessary feedback during the development based on the documentation.

1.2 Scope

The software product is the internship system which mainly focuses on final year students that undergo internships. It is a system where all the students will interact to find their right place in the internship. The system add, delete and edit the messages to ease the process of the communication. This helps the students to ask their lecturer if they have any inquiries. Other than that, there are notification systems where the student and lecturer can view, add, update and delete notifications. This helps students to stay up to date with any new information that is related to internship news.

The scope of this system includes

- User access by authentication
- View message
- Add message
- Delete message
- Edit message
- View notification
- Delete notification
- Upload questions
- View grades
- Take exam

This internship system is an online application that can be accessed by the website which provides both lecturer and student access and find the information more easily. The goal of the system is to create a platform for the student to find their company to do the internships. Meanwhile, the objective is to make lecturer and supervisor jobs easier and save their time of helping their students to find the place for an internship.

1.3 Definitions, Acronyms and Abbreviation

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

Terms, Acronyms/Abbreviation	Meaning
SD	System Documentation
Stakeholder	Individual or group that has an interest in any decision or activity of an organization
Inquiries	An act of asking information
Mnemonic	A system such as pattern of letters, ideas, or associations which assists in remembering something.
C++	An object-oriented computer language.
IDE	Integrated development environment is software for building applications that combines common developer tools into a single graphical user interface (GUI).

1.4 References

1. Sommerville, I. 2016. "Software Engineering", 10th Edition, Pearson.

1.5 Overview .

The 2u2i Final Year Project with Industry (FYP-I) Management System consists of introduction, specific requirements, system architectural design, detailed description of components, data design, interface design, requirements matrix, test cases, as well as appendices. In the introductory phase, we describe the purpose and scope of the project. We have also provided definitions and abbreviations, as well as some references. In specific

requirements, we have covered several external interface requirements which includes user interfaces, hardware interfaces, software interfaces, and communication interfaces. We also explained about several system features which include user access, communication, notification data and assessment. The performance of the project as well as the constraints were also discussed. Other than that, software system attributes were also discussed. In system architectural design, we have explained about architectural style and rationale as well as component model. For detailed description of components, the complete package diagram as well as detailed description of its subsystems were discussed. Data design section includes the data description and data dictionary. For interface design, we have displayed an overview of the interface as a whole. We have also discussed the requirements matrix. For the last part, we discuss the test cases in the project.

2. Specific Requirements

2.1 External Interface Requirements

2.1.1 User Interfaces

User access

- Users are required to enter their username and password.
- Student will enter the main page of student's profile while supervisor will enter the main page of supervisor profile
- Access will be given to the right username and password through verification
- The username and password will be save for future login
- There will be authentication for the user to make sure that the right user enter the system

Notification

- Users can access the notification by going to the interface and choosing notification
- Students are only able to view the notifications that are available in the notification database.
- Coordinator able to modify the notifications database by adding more notifications to the database.
- Coordinator able to modify the notifications database by updating the searched notifications in the database.
- Coordinator able to modify the notification database by deleting the searched notifications in the database.
- The system will display "no data found" if there is no data found in the database.

Communication

- Students, supervisors, coordinators, coaches, and examiners are able to have full accessibility to view, add, delete, and edit a message.
- Users are able to access messages in the system by going to the interface and choosing notification
- Users are able to view existing messages by them or by other people in the system.
- Users are able add a new message in the system.
- Users are able to delete an existing message in the system.
- Users are able to edit an existing message created by them in the system.

- The system will keep an organized list of messages added by users of the system.

Assessment

- Examiners can upload the questions by adding it into the database.
- Students can take the exam online by attempting it on the system.
- Students can view their grade after they have finished their exam.
- Examiners can view their student's grade after their student;s has finished the exam.

2.1.2 Hardware Interfaces

The 2u2i Final Year Project with Industry (FYP-I) Management System is a web based system. The system will be connected to the servers provided by Amazon Web Services and database from MySQL. The web server will have 8 GB of RAM to support the system load while using it. The database will have 1TB capacity to store all of the data. The system can be accessed from multiple devices such as smartphone, laptop or desktop. It also can be accessed by computers that use different operating systems such as Windows, MacOS or Linux. The clients are required to use modern web browsers such as Google Chrome, Microsoft Edge or Mozilla Firefox. The clients also need to have an Internet connection with minimum bandwidth of 10MB/s to access the system. The data from the server will be transferred and received by the clients using basic networking protocols. They also need to enable cookies to ensure they get the best experience using the system.

2.1.3 Software Interfaces

1. Google Chrome (Web Browser)

Name: Google Chrome

Mnemonic: Chrome, Google

Specification number: 103.0.000.00

Version number:

103.0.5060.33 (Windows & macOS & Linux)

103.0.5060.33(Android)

103.0.5060.34(iOS)

Sources: www.google.com/chrome/

Discussion: We use this web browser to operate our systems because it's easier and fast for our users.

2. MySQL (Database)

Name: MySQL

Mnemonic: CREATETABLE, SELECT, INSERT INTO

Specification number: 8.0.00

Version number: 8.0.28

Sources: <https://www.mysql.com/>

Discussion: The database that we used to create, store, search, and also delete all the data in the system.

3. VSC (Programming Software)

Name: Visual Studio Code

Mnemonic: C++, Java, Python

Specification number: 1.67.0

Version number: 1.67.2

Sources: <https://code.visualstudio.com/>

Discussion: The best IDE for us to operate our programming software using many languages such as MySQL and also C++.

2.1.4 Communication Interfaces

This should list the different communication interfaces, such as local network protocols and so on. For receiving student and college management information and encrypting all important information from and into our system for data transfer security, our system will employ Hypertext Transfer Protocol Secure (https). HTTP is a protocol for sending and receiving data.

TCP/IP (Transmission Control Protocol/Internet Protocol) is also used by the system to link network devices via the internet. It may also be used to send data across the Internet. For data transfer, TCP is used. TCP is compatible with IP and enhances it. It's used to send and receive data through a network. It breaks down any communication into a sequence of packets that are transferred from source to destination, where they are reassembled.

2.2 System Features

The system features included in the use case diagram. The use case diagrams will be used to explain the features in the system. The domain model will be used to represent the relationship between the classes in the system. There are also use case descriptions

and sequence diagrams to explain all of the processes involved in the 2u2i Final Year Project with Industry (FYP-I) Management System. The use case description will explain all of the condition and flow of the events of a use case while the sequence diagram will explain the interaction of the classes while the use case is executed.

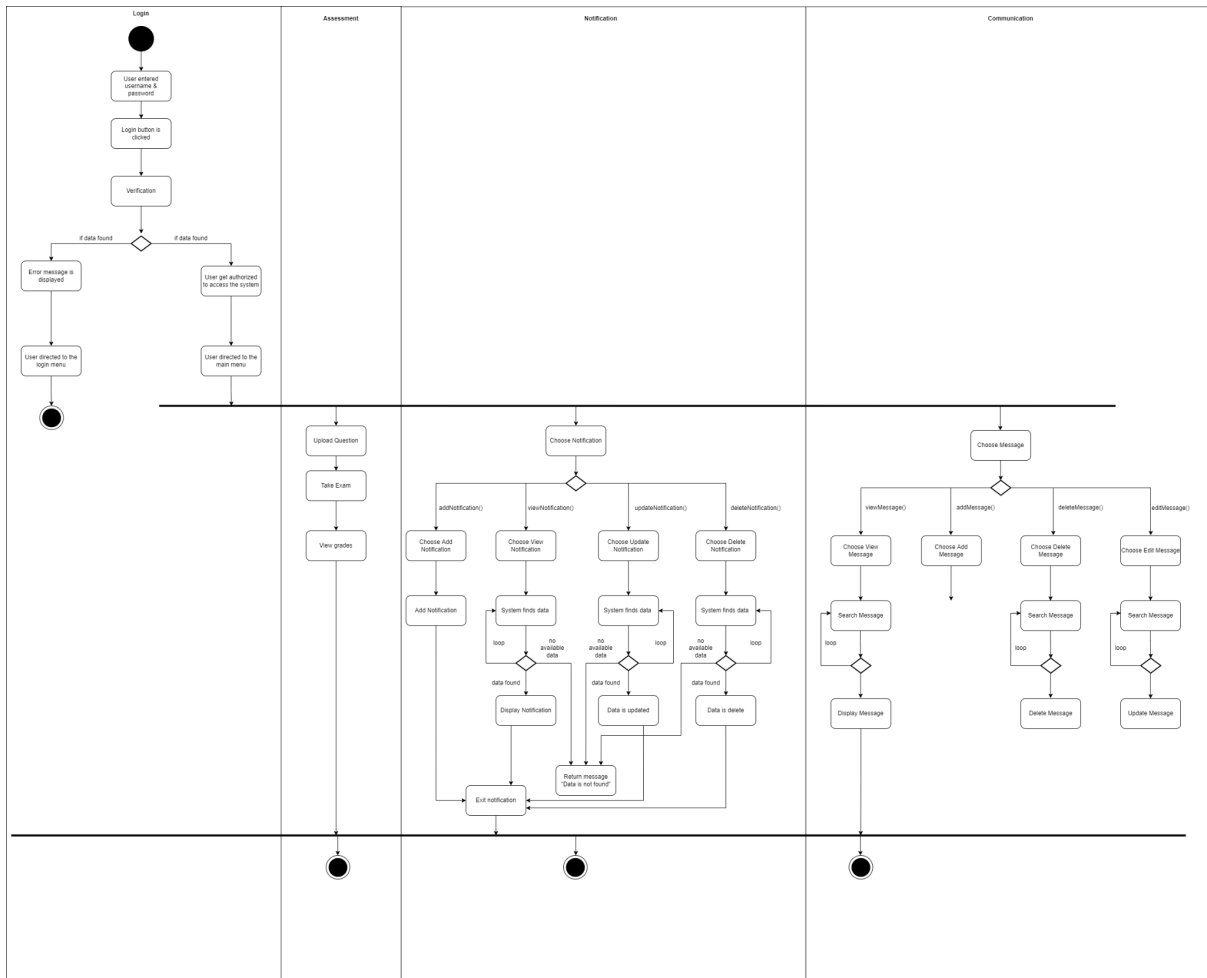


Figure 2.2: Activity Diagram for <2u2i Final Year Project with Industry (FYP-I) Management System>

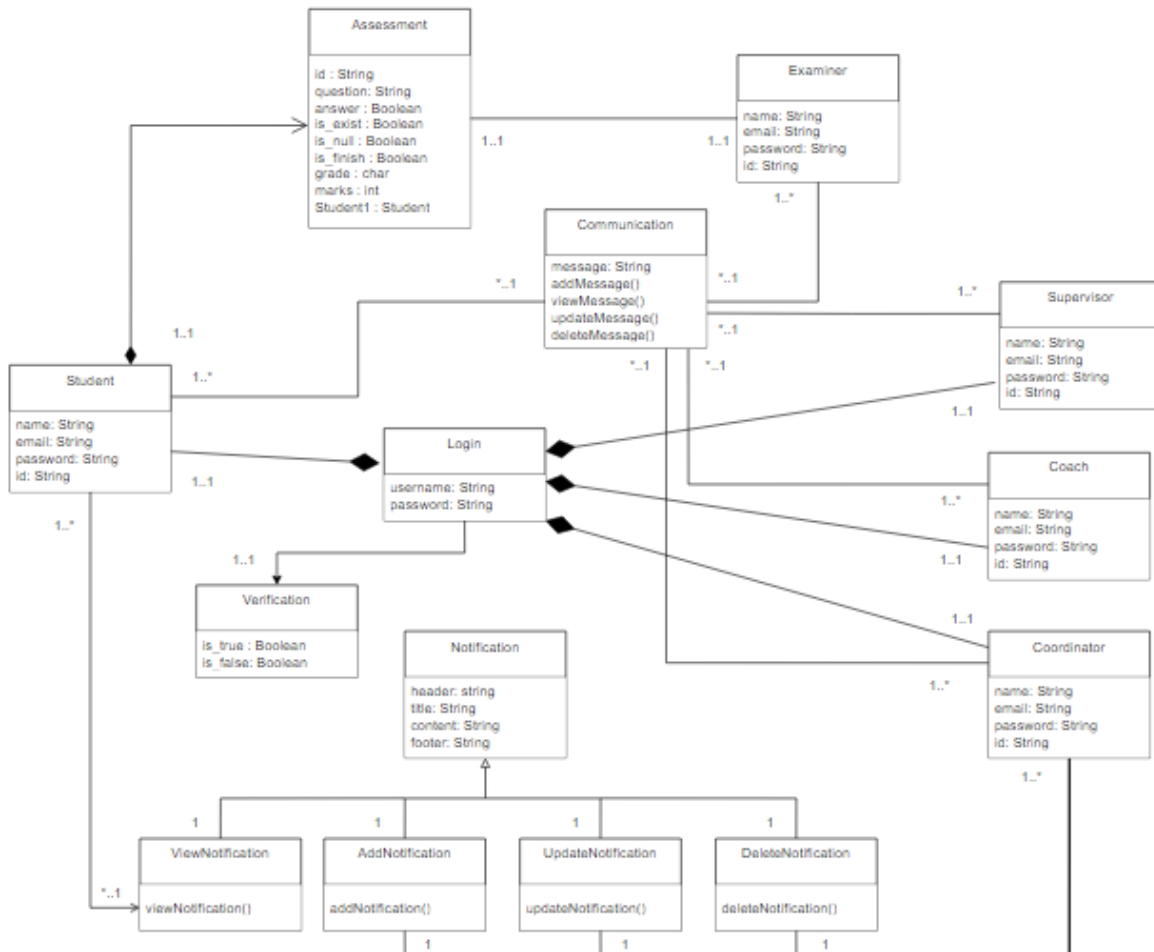


Figure 2.3: Domain Model for <2u2i Final Year Project with Industry (FYP-I) Management System>

2.2.1 UC001: Use Case <Login>

Use Cases: Login
ID: UC1
Actors: A1 Student A2 Supervisor A3 Coordinator A4 Coach A5 Examiner
Includes: UC2 Verification
Extensions Point:
Preconditions:
Flow of Events: 1. Users enter their username and password to the system. 2. Users click on the login button to proceed accessing the system. 3. Include Verification
Postconditions:

Figure 2.4: Use Case Description for <Login>

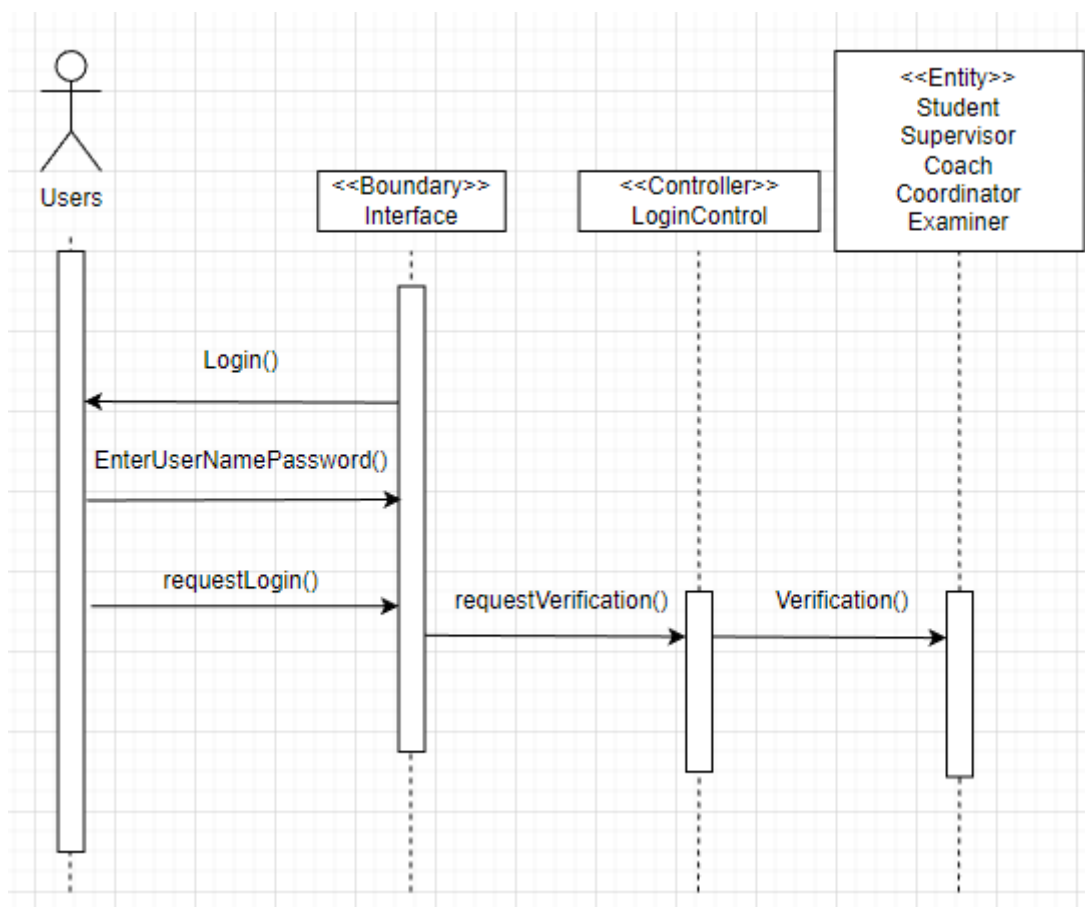


Figure 2.5: Sequence Diagram for <Login>

2.2.2 UC002: Use Case<Verification>

Use Cases: Verification
ID: UC2
Actors:
Includes:
Extensions Point:
Preconditions: 1. The username and password have been entered by the user.
Flow of Events: 1. The username and password will be verified with the data in the database. 2. If the username and password matched with the database 2.1 The user will be authorized and can access the system. 2.2 The user will be directed to the main menu. 3. Else 3.1 The system will display error message and redirected to login page. 3.2 The user needs to enter the correct username and password.
Postconditions: 1. Users are sent to the main page and can access the system.

Figure 2.6: Use Case Description for <Verification>

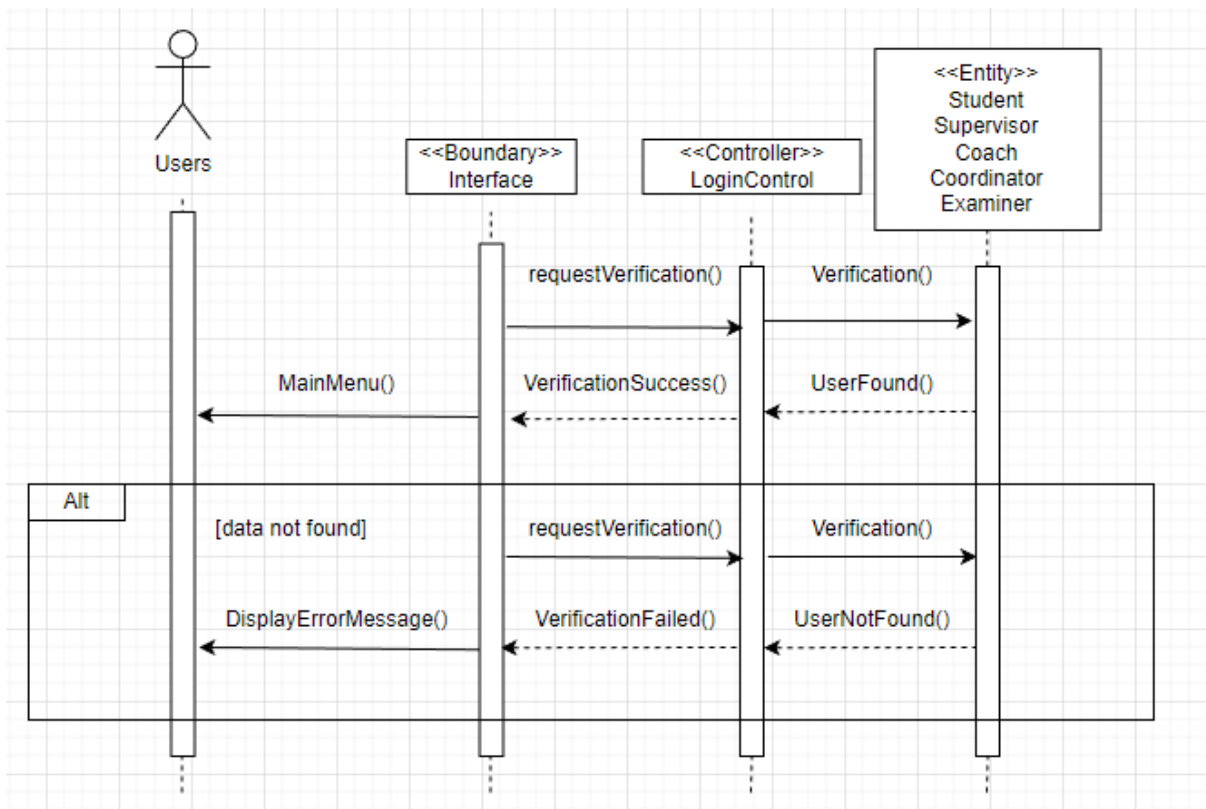


Figure 2.7: Sequence Diagram for <Verification>

2.2.3 UC003: Use Case<View Message>

Use Case: View Message
ID: uc3
Actor: A1 Student A2 Supervisor A3 Coordinator A4 Coach A5 Examiner
Includes:
Extension points:
Preconditions: 1. The actor has logged in to the system. 2. The actor has added message.
Flow of Events: 1. The actor logs into the system. 2. The system gives choices to the actor for messaging. 3. The actor chooses to view a message. 4. The system shows the actor the existing message. 5. The actor logs out of the system.
Postconditions: 1. The message has been viewed by the actor.

Figure 2.8: Use Case Description for <View Message>

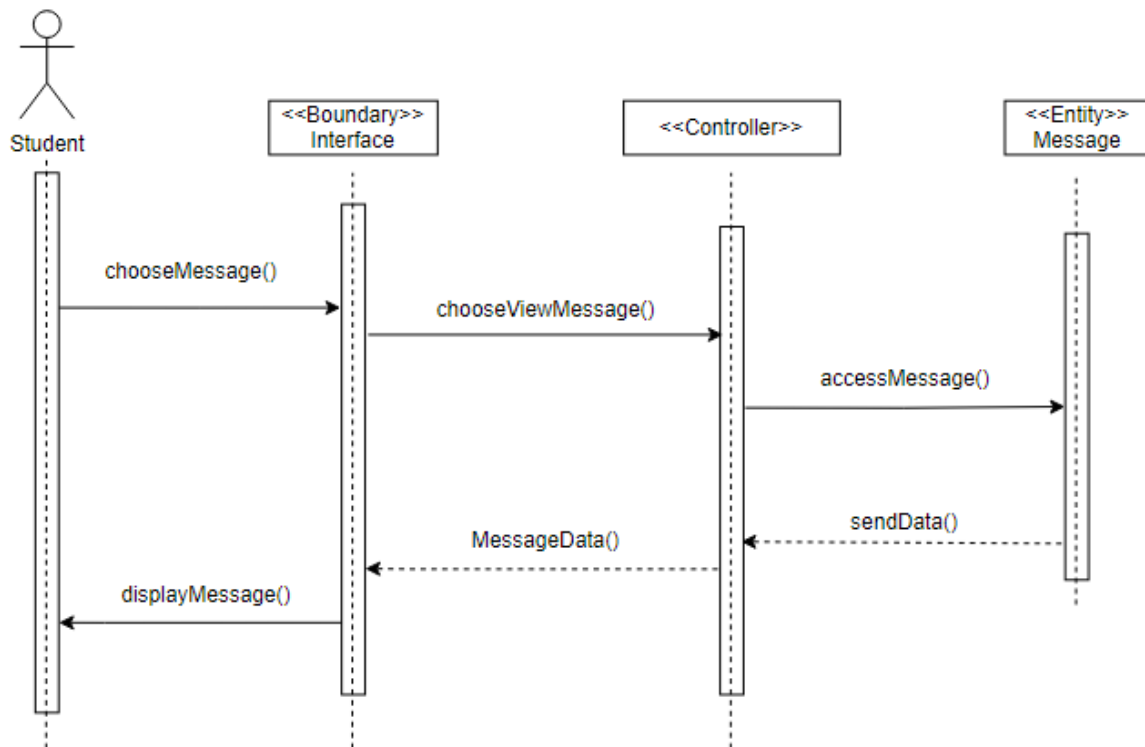


Figure 2.9: Sequence Diagram for <View Message>

2.2.4 UC004: Use Case<Add Message>

Use Case: Add Message
ID: UC4
Actor: A1 Student A2 Supervisor A3 Coordinator A4 Coach A5 Examiner
Includes:
Extension points:
Preconditions: 1. The actor has logged in to the system.
Flow of Events: 1. The actor logs into the system. 2. The system gives choices to the actor for messaging. 3. The actor chooses to add a message. 4. The system displays a text box to add a message. 5. The actor fills in the text box for the message. 6. The system updates. 7. The actor logs out of the system.
Postconditions: 1. The message has been added by the actor.

Figure 2.10: Use Case Description for <Add Message>

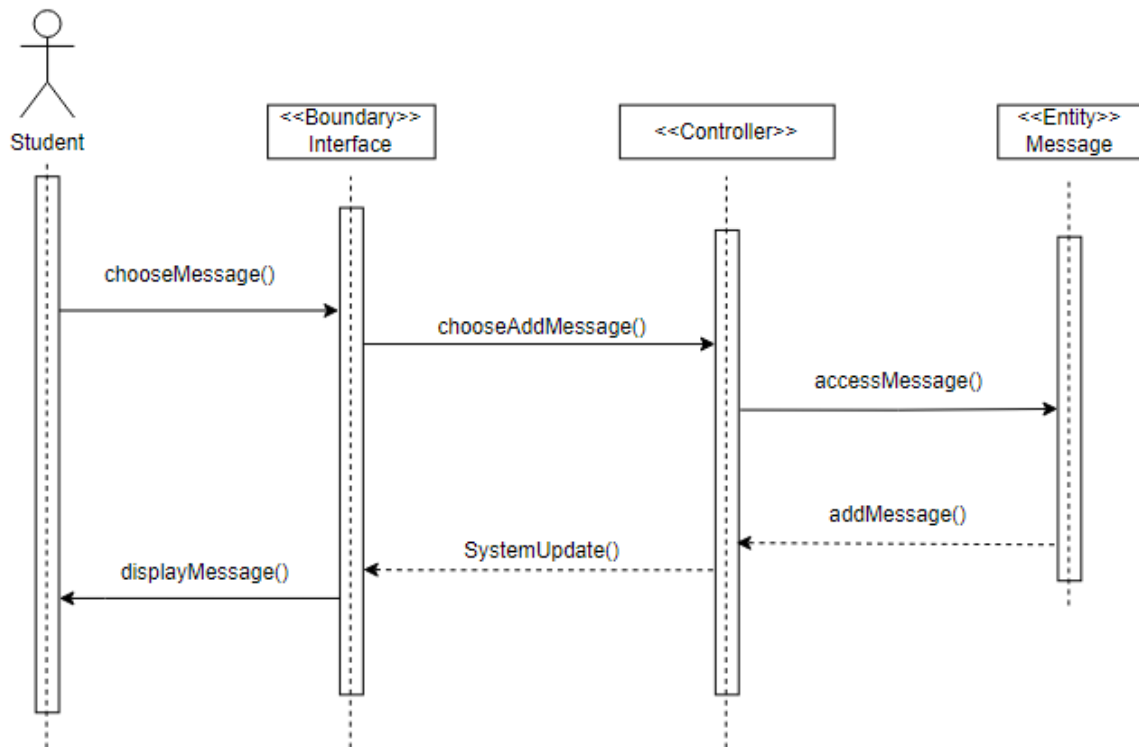


Figure 2.11: Sequence Diagram for <Add Message>

2.2.5 UC005: Use Case<Delete Message>

Use Case: Delete Message
ID: UC5
Actor: A1 Student A2 Supervisor A3 Coordinator A4 Coach A5 Examiner
Includes:
Extension points:
Preconditions: 1. The actor has logged in to the system. 2. The actor has added message.
Flow of Events: 1. The actor logs into the system. 2. The system gives choices to the actor for messaging. 3. The actor chooses to delete a message. 4. The system shows list of messages to delete. 5. The system asks for confirmation on deleting the message. 6. The system updates. 7. The actor logs out of the system.
Postconditions: 1. The message has been deleted by the actor.

Figure 2.12: Use Case Description for <Delete Message>

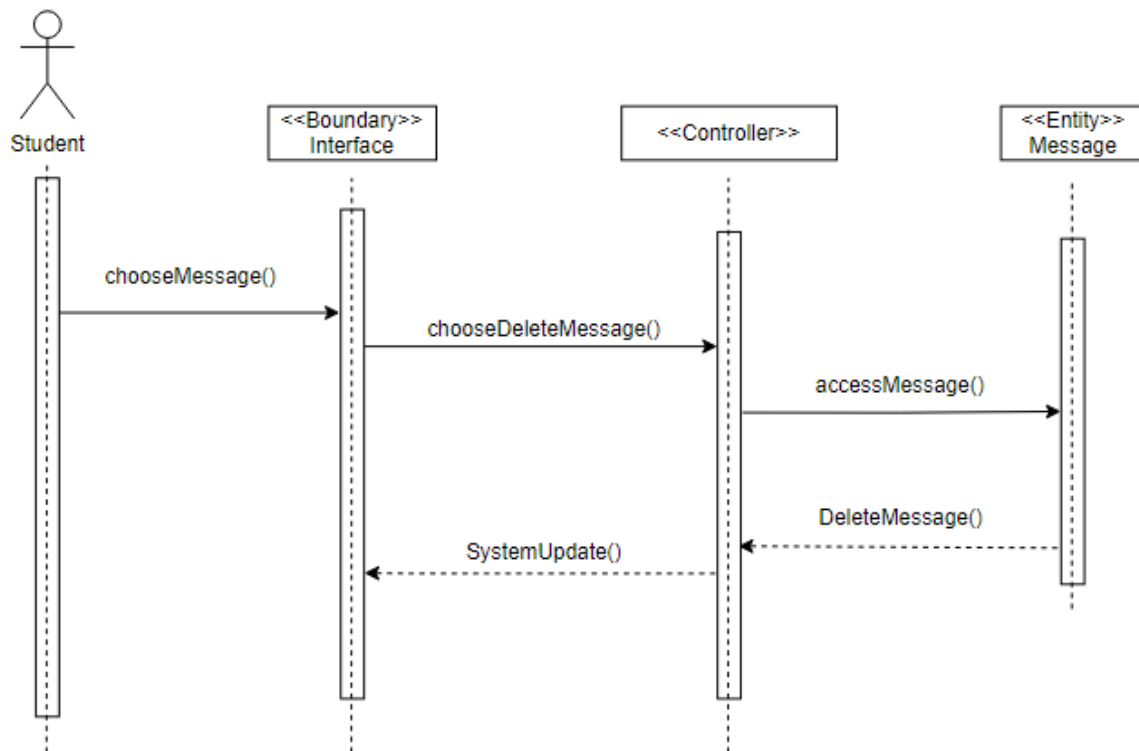


Figure 2.13: Sequence Diagram for <Delete Message>

2.2.6 UC006: Use Case<Edit Message>

Use Case: Edit Message
ID: UC6
Actor: A1 Student A2 Supervisor A3 Coordinator A4 Coach A5 Examiner
Includes:
Extension points:
Preconditions: 1. The actor has logged in to the system. 2. The actor has added message.
Flow of Events: 1. The actor logs into the system. 2. The system gives choices to the actor for messaging. 3. The actor chooses to edit a message. 4. The system displays a text box to edit the message. 5. The actor fills in the text box for the message. 6. The system updates. 7. The actor logs out of the system.
Postconditions: 1. The message has been edited by the actor.

Figure 2.14: Use Case Description for <Edit Message>

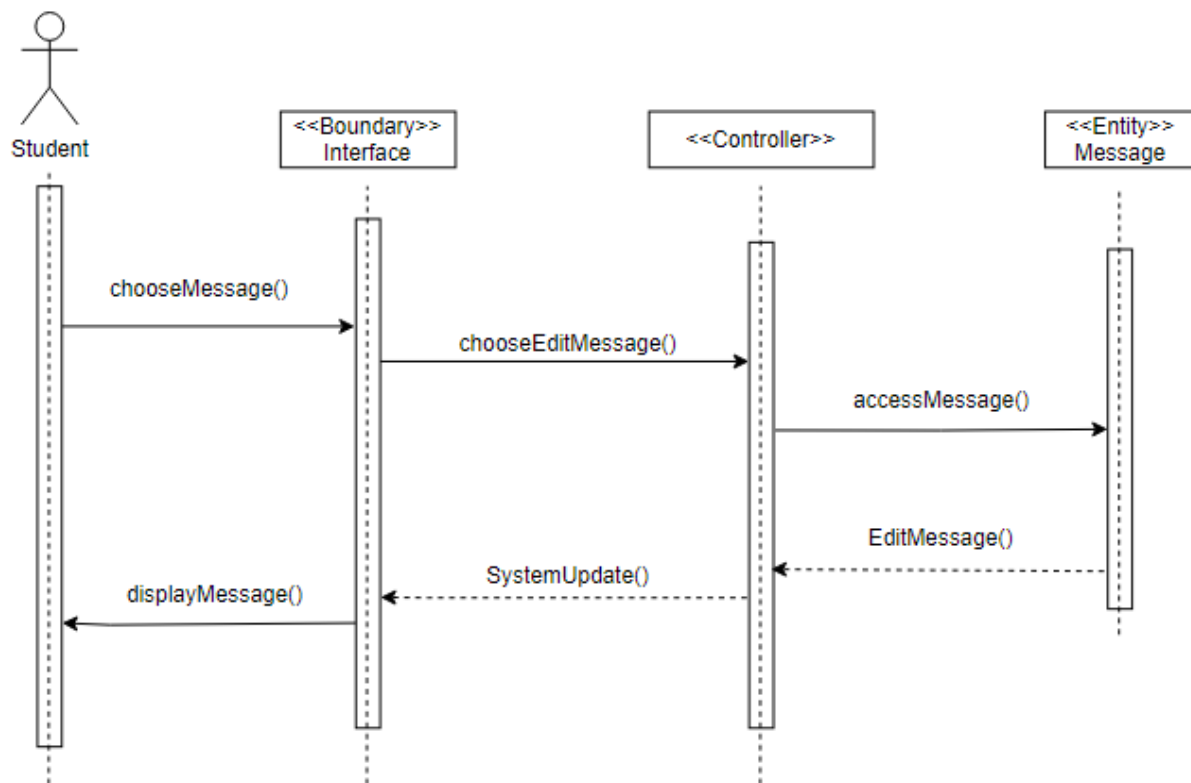


Figure 2.15: Sequence Diagram for <Edit Message>

2.2.7 UC007: Use Case <Notification Data>

Use cases: Notification Data
ID: UC7
Extends: <returnMessage>
Preconditions:
Flow of Events: - If there are data in notification - Return data to users. - System returns back to previous use case. - Else - There are no data available in the system <returnMessage>
Postconditions: - User able to get the data from Notification Data - User will get an error message if there is no data.

Figure 2.16: Use Case Description for <Notification Data>

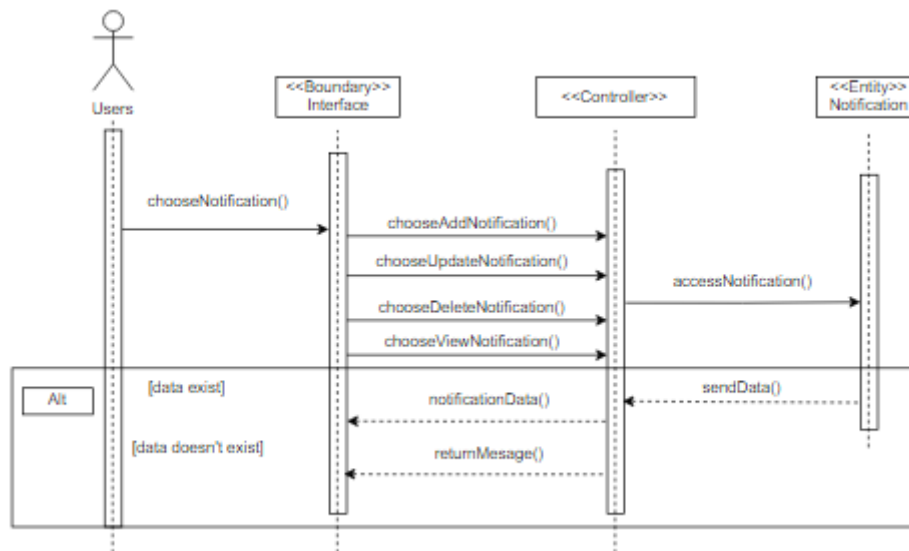


Figure 2.17: Sequence Diagram for <Notification Data>

2.2.8 UC008: Use Case <View Notification>

Use cases: View Notification	
ID:	UC8
Actors:	A1 Student
Includes:	UC7 Notification Data
Preconditions:	1. The student has logged in into the system. 2. Student has entered the interface of the system.
Flow of Events:	1. Student selects "View Notification" 2. include (Notification Data) 3. System will send a request to access the notification database 4 For each data found 4.1. System will print out notification data 5. There is no more data 5.1 System will exit the use case
Postconditions:	1. System able to display the notification to Students

Figure 2.18: Use Case Description for <View Notification>

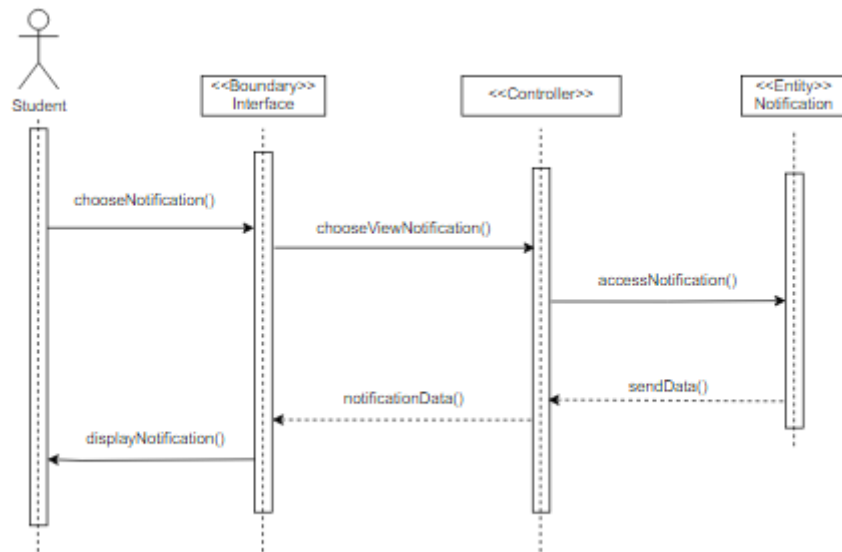


Figure 2.19: Sequence Diagram for <View Notification>

2.2.9 UC009: Use Case <Add Notification>

Use cases: Add Notification
ID: UC9
Actors: A3 Coordinator
Includes: UC7 Notification Data
Preconditions: 1. Coordinator is logged in into the system.
Flow of Events: 1. Coordinator selects "Add Notification" 2. include (Notification Data) 3. System will send a request to access the notification database 4. Coordinator add notification to the database.
Postconditions: 1. Coordinator able to add notification to the system for Student to view.

Figure 2.20: Use Case Description for <Add Notification>

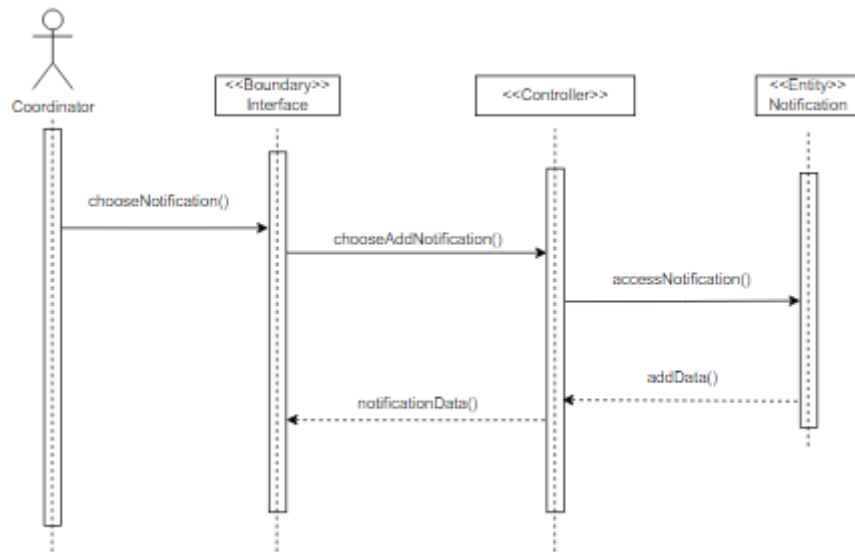


Figure 2.21: Sequence Diagram for <Add Notification>

2.2.10 UC010: Use Case <Update Notification>

Use cases: Update Notification
ID: UC10
Actors: A3 Coordinator
Includes: UC7 Notification Data
Extends: <returnMessage>
Preconditions: 1. Coordinator is logged in into the system.
Flow of Events: - Coordinator selects "Update Notification" - include (Notification Data) - System will send a request to access the notification database - While searching for notification - If notification data is found - Coordinator is able to update the notification - Else <returnMessage>
Postconditions: Coordinator able to update the notification data

Figure 2.22: Use Case Description for <Update Notification>

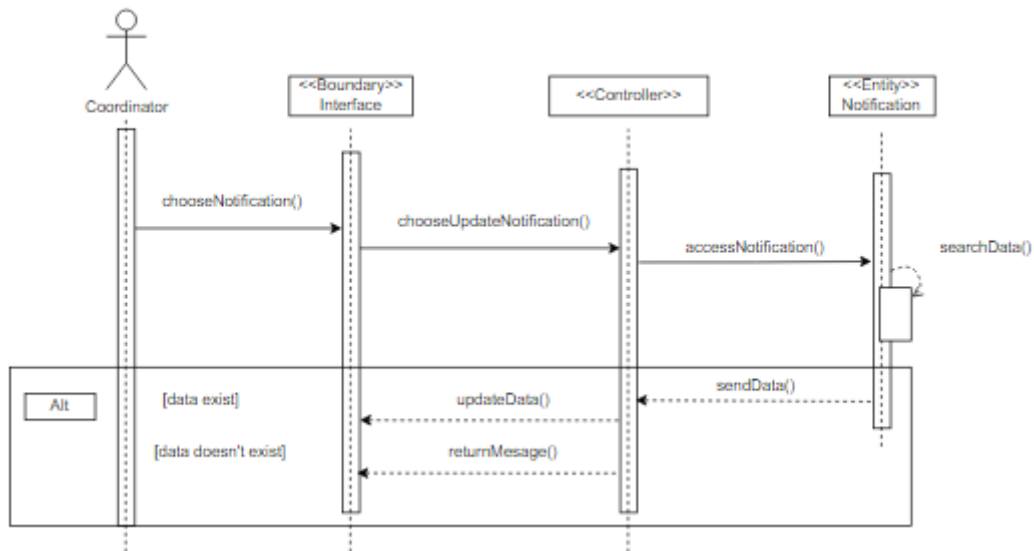


Figure 2.23: Sequence Diagram for <Update Notification>

2.2.11 UC011: Use Case <Delete Notification>

Use cases: Delete Notification
ID: UC11
Actors: A3 Coordinator
Includes: UC7 Notification Data
Extends: <returnMessage>
Preconditions: 1. Coordinator is logged in into the system.
Flow of Events: 1. Coordinator selects "Delete Notification" 2. include (Notification Data) 3. System will send a request to access the notification database 4. While searching for notification 4.1. If notification data is found 4.1.1. Coordinator is able to delete the notification 4.2 Else <returnMessage>
Postconditions: 1. Coordinator able to delete the notification data

Figure 2.24: Use Case Description for <Delete Notification>

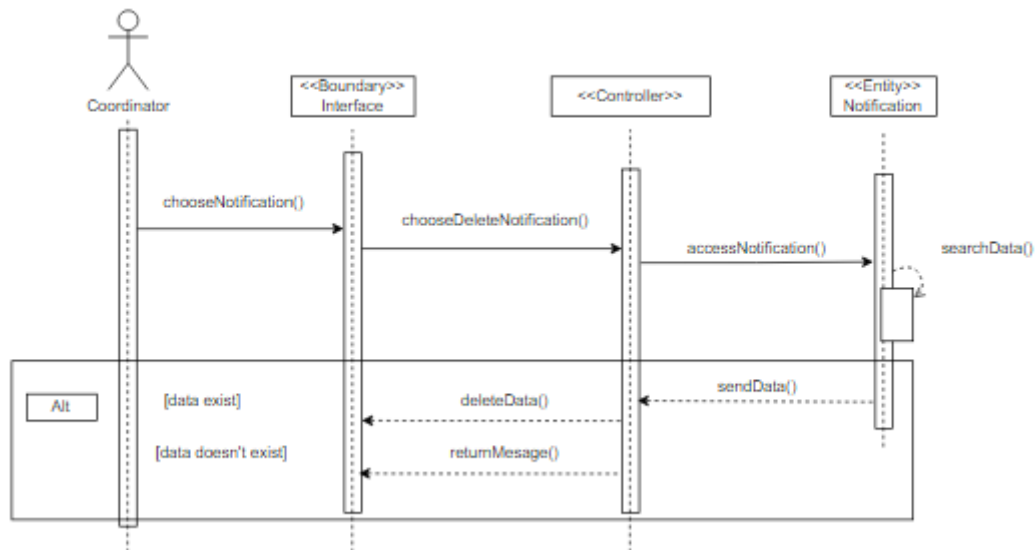


Figure 2.25: Sequence Diagram for <Delete Notification>

2.2.12 UC012: Use Case <Return Message>

Extension use cases: Return Message
ID: UC12
Extends: UC7 Notification Data at <returnMessage> UC10 Update Notification at <returnMessage> UC11 Delete Notification at <returnMessage>
Flow of Events: 1. System will print out a message "No notification data available"
Postconditions: 1. Users can add types of news that they want in their notification 2. Users can view the news that they had selected for their notification

Figure 2.26: Use Case Description for <Return Message>

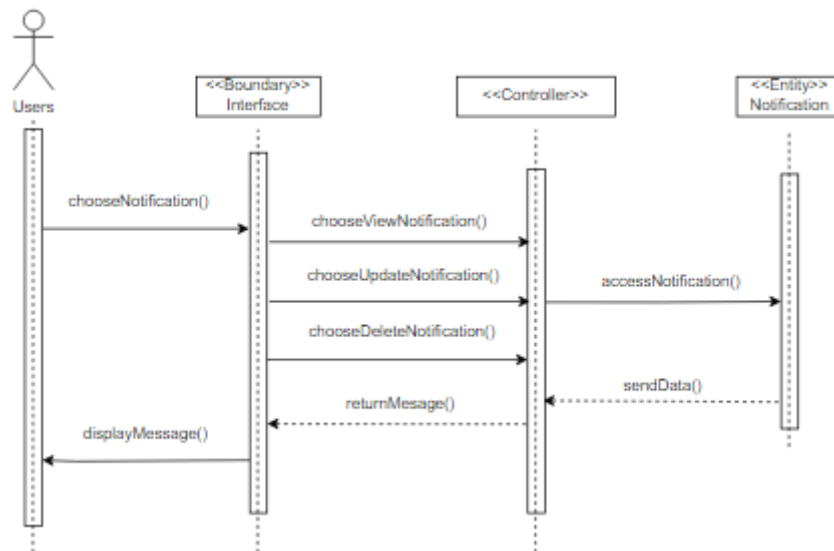


Figure 2.27: Sequence Diagram for <Return Message>

2.2.13 UC013: Use Case<Upload Question>

Use case: Upload Question
ID: UC13
Actors: A5 Examiner
Includes:
Extension Points:
Preconditions: 1. The examiner has logged in to the system.
Flow of Events: - The examiner clicks on the button that they want to upload the question. - If the question not uploaded yet - The file that contains the questions can be uploaded by clicking the upload button. - The answer for each of the question need to be set by clicking the right answer for the question. - The system stores the questions and answers - The exam is ready to be taken by the students - Else - The message about the question has been uploaded will be displayed. - The examiner will be redirected to main page.
Postconditions: 1. The question has been uploaded to the system.

Figure 2.28: Use Case Description for <Upload Question>

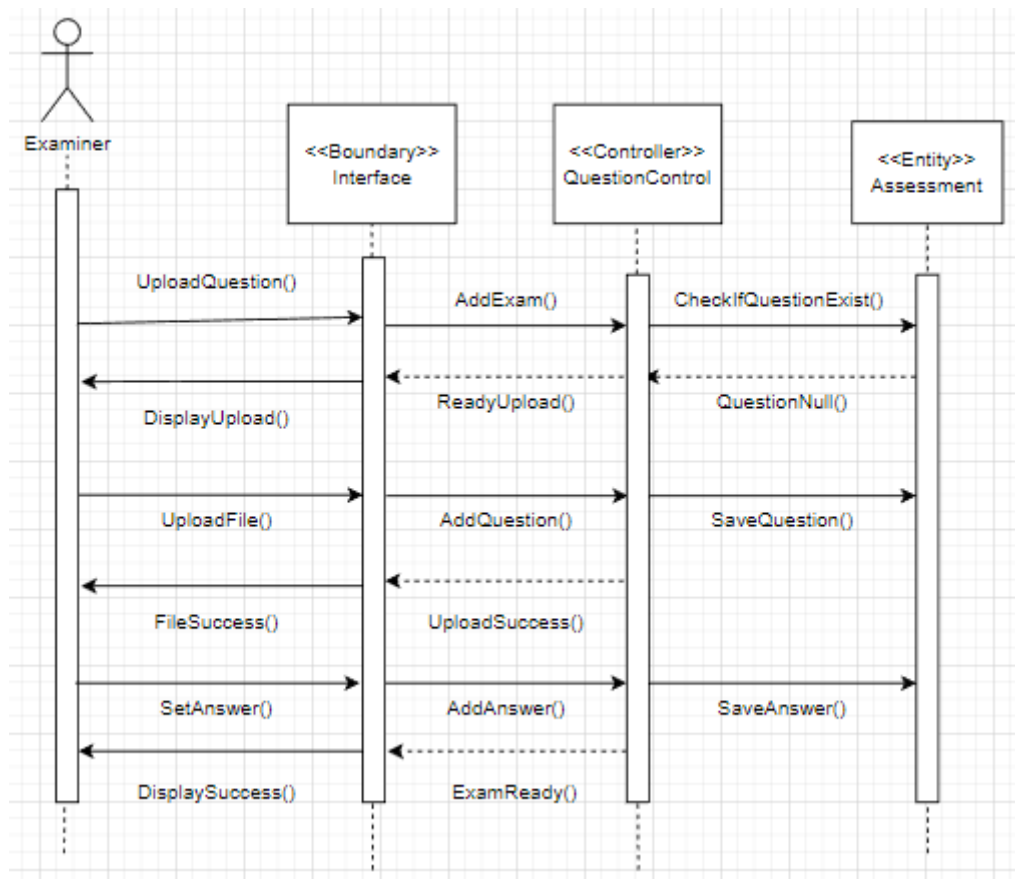


Figure 2.29: Sequence Diagram for <Upload Question> if Question Not Available

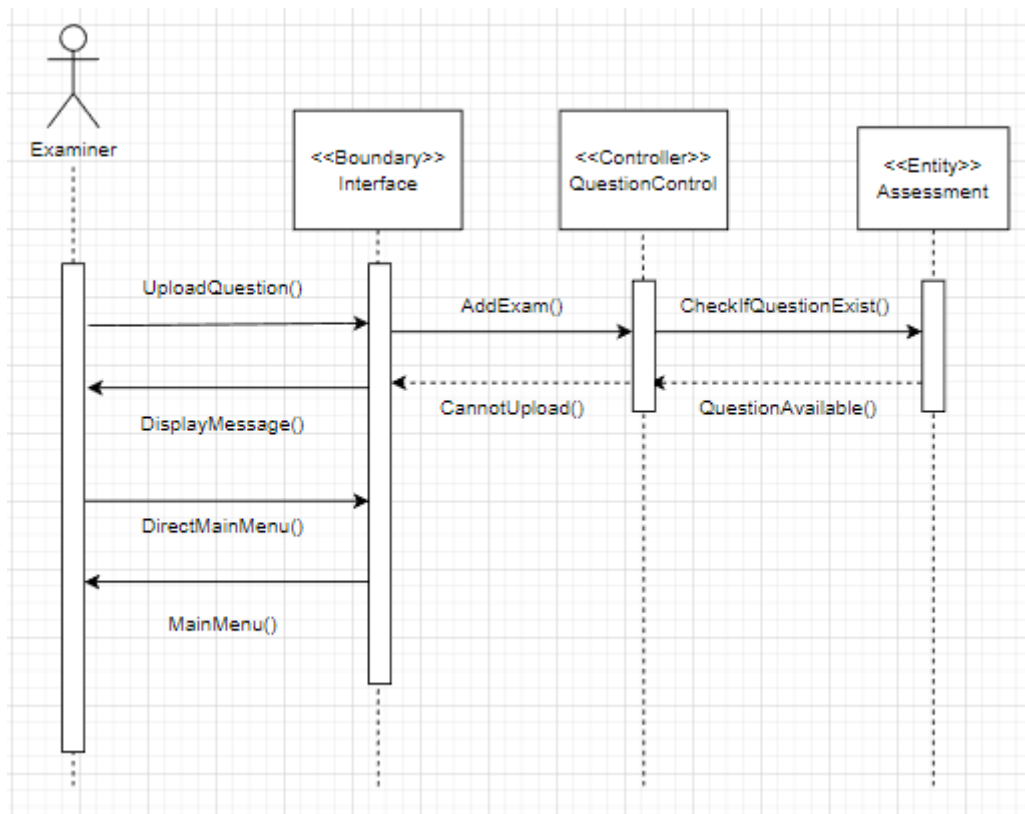


Figure 2.30: Sequence Diagram for <Upload Question> if Question Available

2.2.14 UC014: Use Case<Take Exam>

Use case: Take Exam
ID: UC14
Actors: A1 Student
Includes:
Extension Points: UC014 View Grades
Preconditions: 1. The student has logged in to the system. 2. The question has been uploaded on to the system.
Flow of Events: 1. The students click on the exam that they need to attend. 2. The student attempts the exam by clicking the start button. 3. All the questions are displayed and need to be answered by the student. 4. After all the question has been answered, the student can finish the student by clicking the finish button. 5. If the student has finished the exam 5.1 Extend View Grades
Postconditions: The student has finished the exam.

Figure 2.31: Use Case Description for <Take Exam>

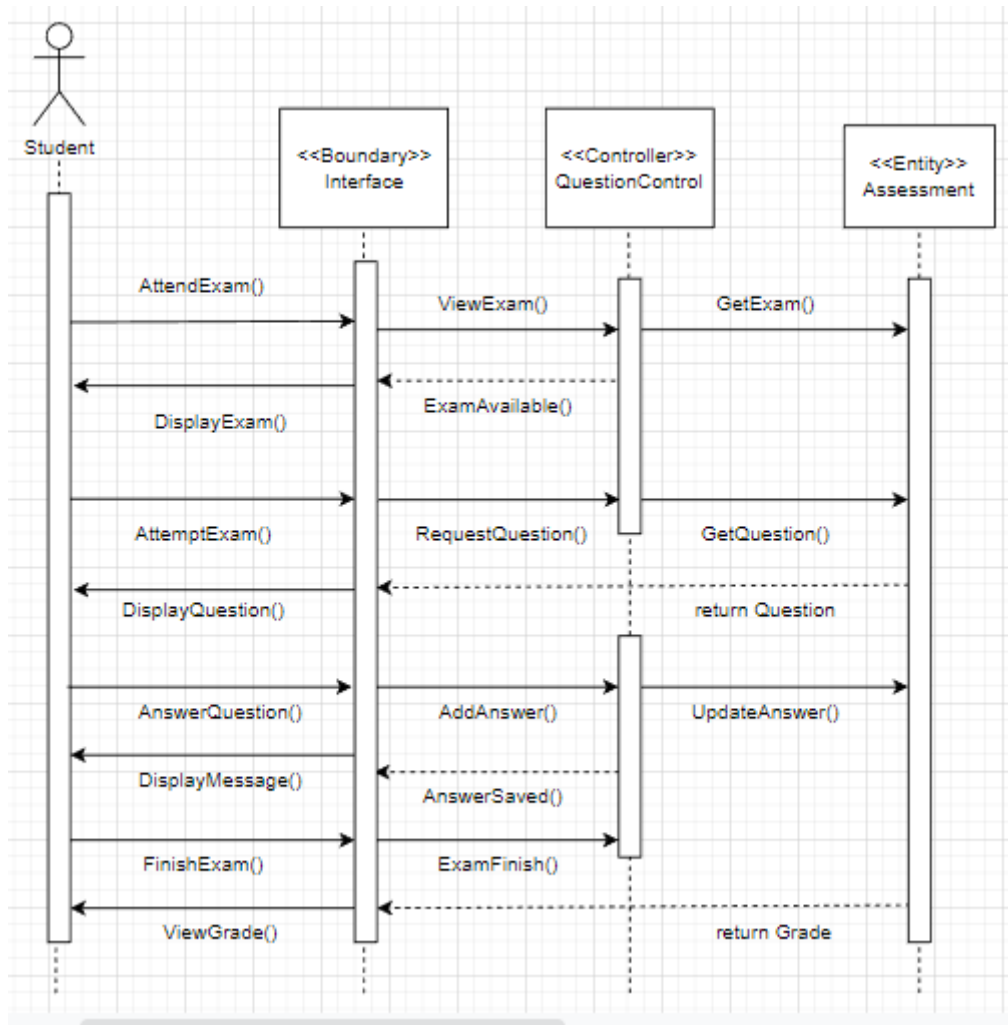


Figure 2.32: Sequence Diagram for <Take Exam>

2.2.15 UC015: Use Case<View Grades>

Use case: View Grades
ID: UC15
Actors: A1 Student A5 Examiner
Includes:
Extension Points:
Preconditions: 1. The student has logged in to the system. 2. The examiner has logged in to the system. 3. The student has finished their exams
Flow of Events: 1. The system will check the student's answer based on the answer that has been set by examiners. 2. The grades obtained after the checking has finished. 3. If the student wants to view grades 1.1 The grades displayed immediately after the exams. 4. If the examiner wants to view grades 2.1 A list of students that has finished the exam will be displayed. 2.2 The examiner can click on the student's name and their grades will be displayed.
Postconditions:

Figure 2.33: Use Case Description for <View Grades>

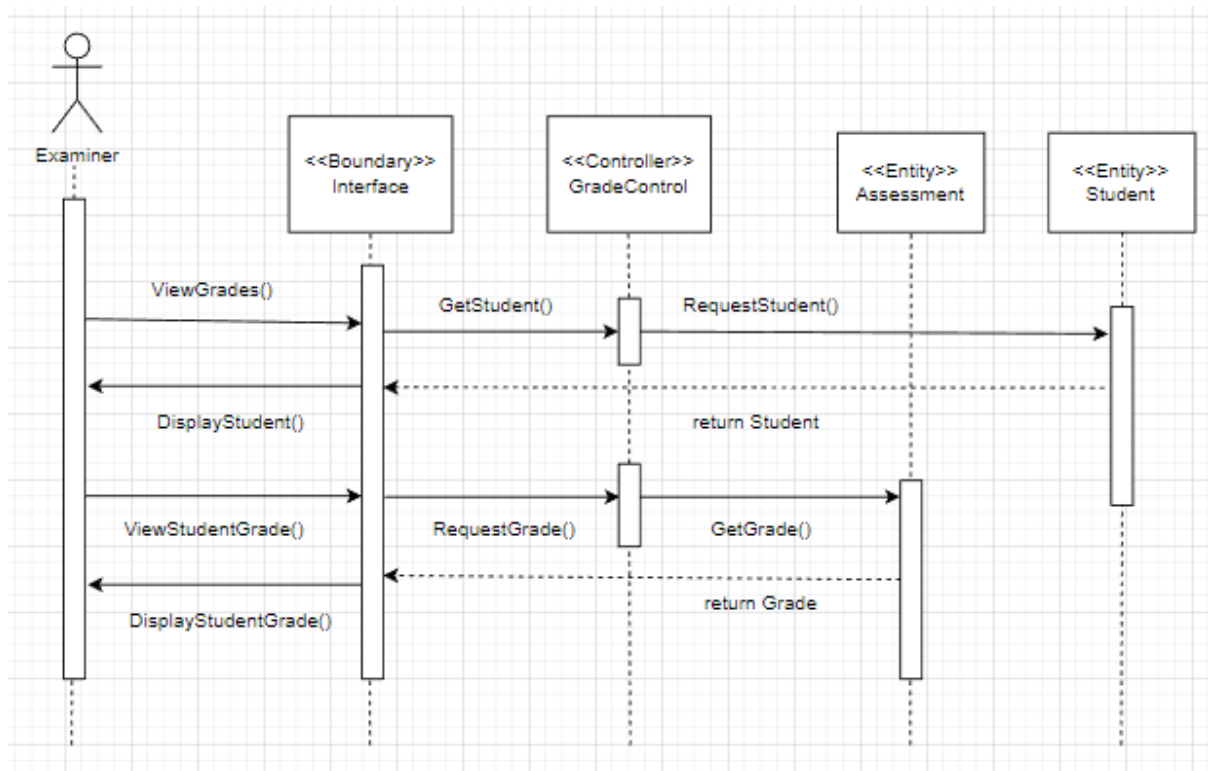


Figure 2.34: Sequence Diagram for <View Grades>

2.3 Performance and Other Requirements

- The **RESPONSE TIME** of the system should be only 1.0 second after the user clicks on the system.
- The **WORKLOAD** of the system should be able to handle more than 1000 students (including staff) and at least 200 users at one time during peak hours.
- The **THROUGHPUT** of the system should be around 10 operations per second. This means that the system can handle updating, adding, deleting and viewing content all at the same time.
- The **CAPACITY** of the system should be around 10 contents in a page with scrolling involved.

2.4 Design Constraints

- The system will be represented following the standards that are used on websites nowadays so the users can easily understand the system.
- The contents of the system shall follow the regulations by the Malaysia Government and Universiti Teknologi Malaysia (UTM).

- The language that will be used in the system is only limited to English and Bahasa Melayu.
- The system shall store the file related to the Final Year Project of 2u2i students that are currently doing their industrial training including their assessment.
- The file stored can only be accessed by those who have authorities.
- The website is constrained by the capacity of access that it can have at one time. Heavy load can cause the transfer of data to be slower and may cause server down.

2.5 Software System Attributes

- The system is available on any web browser
- Unique username and password registered for every user. They can use the username and password to log in to the system.
- Users can view, add, delete, edit messages to any users that are registered to the system.
- Students are able to view notifications that have been sent by the coordinator.
- The students are able to view their grades after they finished the exam.

3.0 System Architectural Design

3.1 Architecture Style and Rationale

The architectural style that we use is layered architecture style. The layered architecture style is organized into horizontal layers where each layer will perform a specific role in the system. There are 4 layers of layered architecture: presentation layer, data service layer, business logic layer and data access layer. It is composed of many layers that function together as a single unit of software. The layered architecture style usually works where the data stores are controlled by layer. The client needs to request the systems to perform certain actions. Basically the data is held on the central database where all subsystems can be accessed and this makes the activities to be more manageable as it is split into smaller tasks. Here are a lot of reasons why we choose this kind of architecture. One of the reasons is this architecture helps us to organize the data of the students and lecturers in a single database and are able to be accessed by the university. With this architecture, we are able to see any development changes in each of the subsystems and are able to make the changes. Other than that, layered architecture is simple and it is easy to implement and learn. Next, it has the consistency in the overall code and the layered projects. Also, it is browsable where all the objects are kept together.

3.2 Component Model

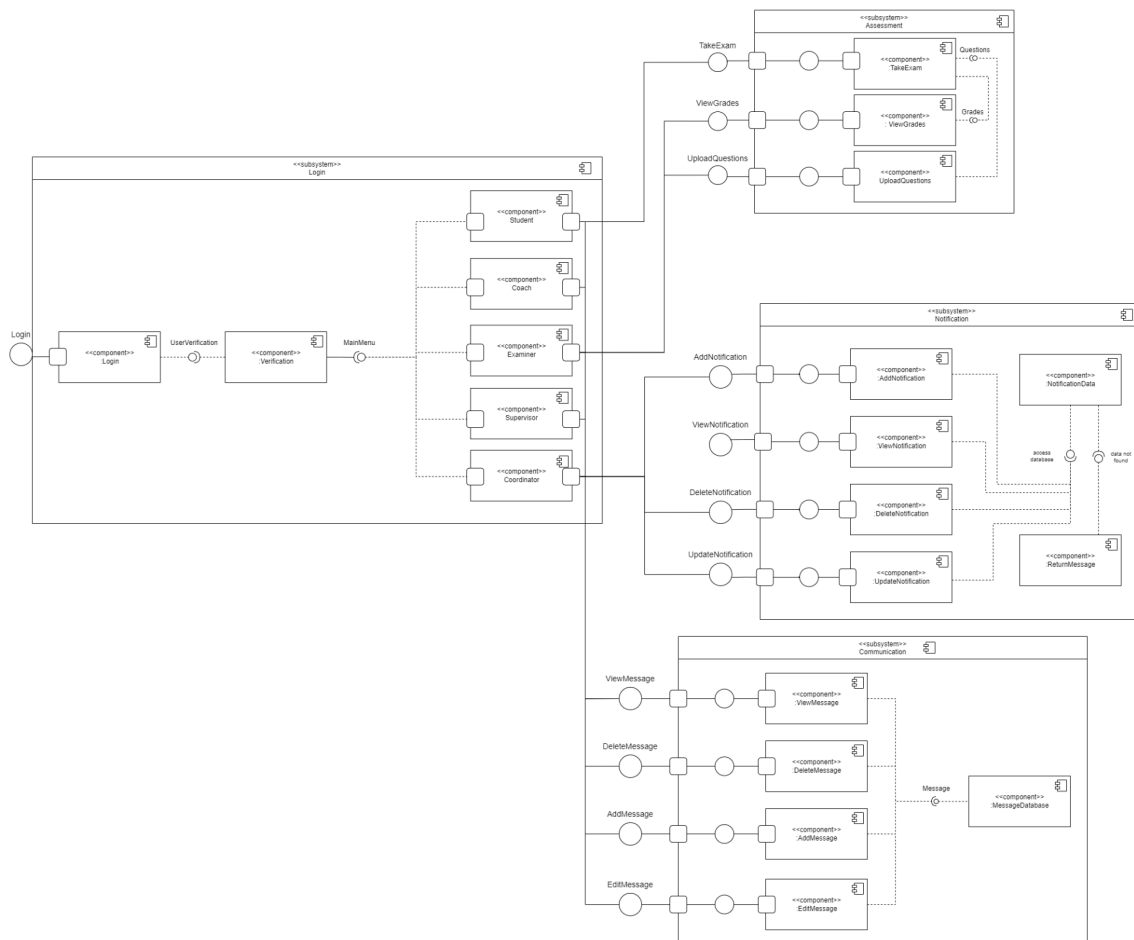


Figure 3.1: Component Diagram of <Inferno 2u2i Final Year Project with Industry (FYP-I) Management System >

In the system, there are a total of 4 subsystems which are the login, assessment, communication and notification subsystem. Each of these subsystems will be connected as shown above. The login subsystem is connected to the other 3 subsystems. This is because the user needs to log into the system first before they can access the system. In the Login subsystem, the verification needs the login information from the login component. Then, the verification component will let the user access into the system according to their role. In the Assessment subsystem, there dependencies exist between the components. The take exam component needs to require the question from the upload question component. Then, the view grades component needs to require grades from the take exam component. In the Notification subsystem, all of the components are connected to the notification data component. In the Communication subsystem, all of the components are connected to the message database.

4.0 Detailed Description of Components

4.1 Complete Package Diagram.

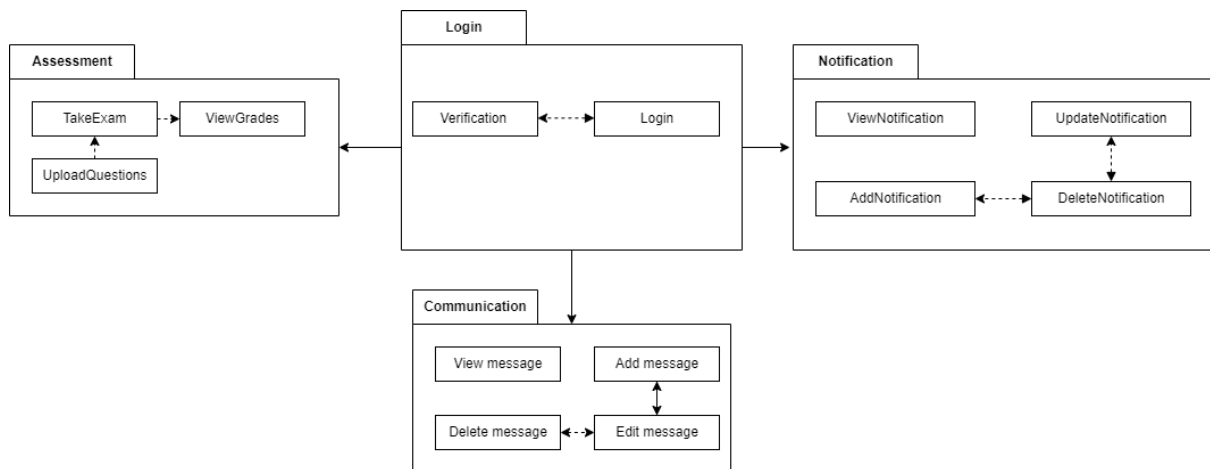


Figure 4.1: Package Diagram for <Inferno 2u2i Final Year Project with Industry (FYP-I) Management System>

4.2 Detailed Description

4.2.1 P001: <Login> Subsystem

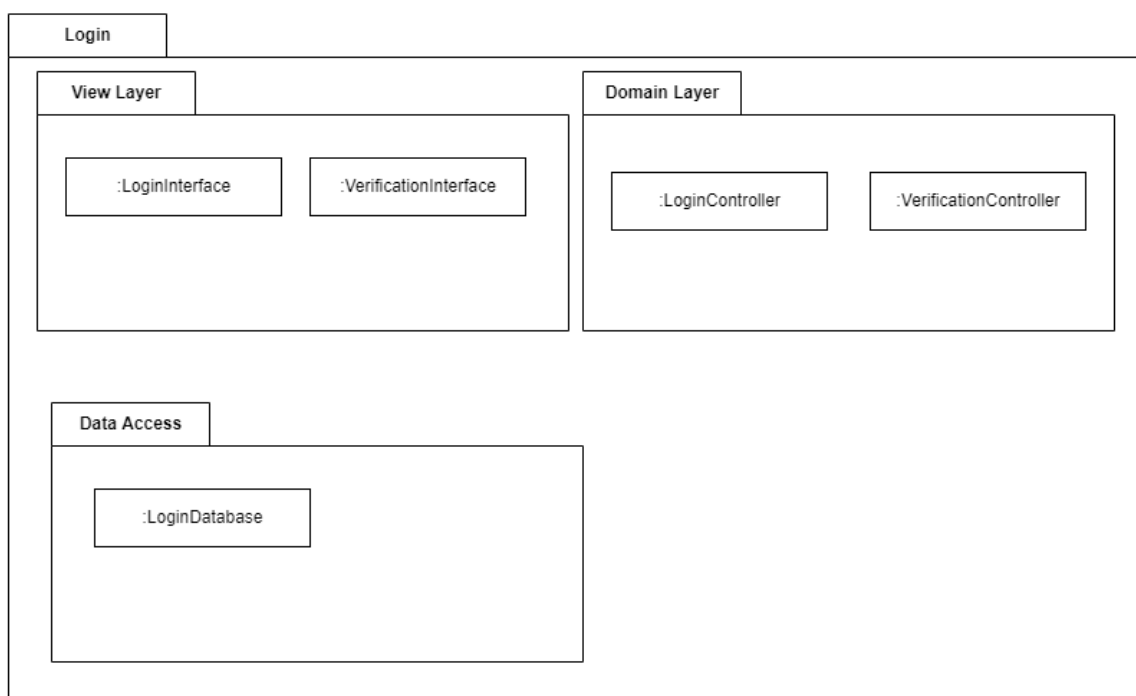


Figure 4.2: Package Diagram for <Login> Subsystem

4.2.1.1 Class Diagram

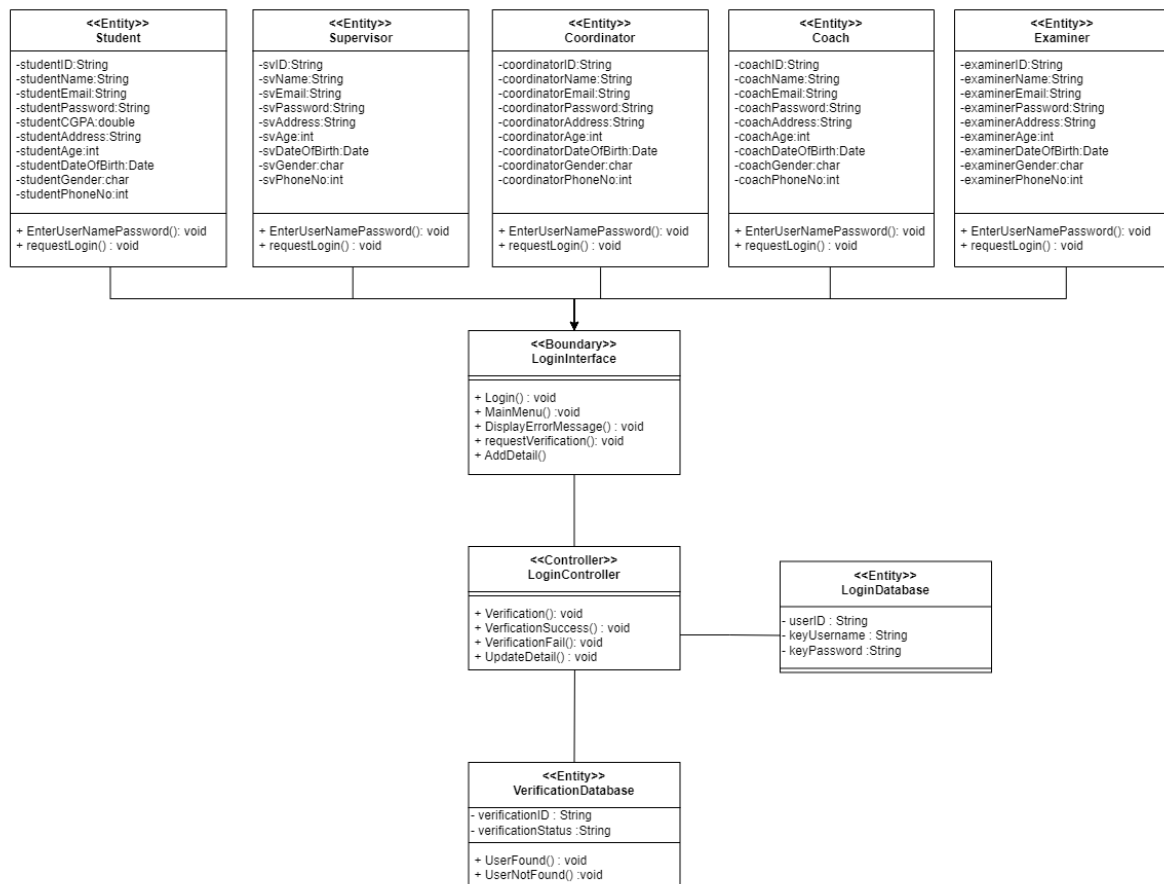


Figure 4.3: Class Diagram for <Login> Subsystem

Entity Name	Student
Method Name	EnterUserNamePassword()
Input	keyUsername, keyPassword
Output	
Algorithm	1. Start 2. User entered their username and password by keyboard 3. End

Entity Name	Student
Method Name	requestLogin()
Input	GUI Input

Output	
Algorithm	1. Start 2. User click on the login button 3. End

Entity Name	Supervisor
Method Name	EnterUserNamePassword()
Input	keyUsername, keyPassword
Output	
Algorithm	1. Start 2. User entered their username and password by keyboard 3. End

Entity Name	Supervisor
Method Name	requestLogin()
Input	GUI Input
Output	
Algorithm	1. Start 2. User click on the login button 3. End

Entity Name	Coordinator
Method Name	EnterUserNamePassword()
Input	keyUsername, keyPassword
Output	
Algorithm	1. Start 2. User entered their username and password by keyboard 3. End

Entity Name	Coordinator
Method Name	requestLogin()
Input	GUI Input
Output	
Algorithm	1. Start 2. User click on the login button 3. End

Entity Name	Coach
Method Name	EnterUserNamePassword()
Input	keyUsername, keyPassword
Output	
Algorithm	1. Start 2. User entered their username and password by keyboard 3. End

Entity Name	Coach
Method Name	requestLogin()
Input	GUI Input
Output	
Algorithm	1. Start 2. User click on the login button 3. End

Entity Name	Examiner
Method Name	EnterUserNamePassword()
Input	keyUsername, keyPassword
Output	
Algorithm	1. Start 2. User entered their username and password by keyboard 3. End

Entity Name	Examiner
Method Name	requestLogin()
Input	GUI Input
Output	
Algorithm	1. Start 2. User click on the login button 3. End

Entity Name	VerificationDatabase
Method Name	UserFound()
Input	keyUsername, keyPassword
Output	verificationStatus
Algorithm	1. Start 2. If keyUsername && keyPassword not null

	2.1 If keyUsername && keyPassword match 2.2 verificationStatus = "success" 3. End
--	---

Entity Name	VerificationDatabase
Method Name	UserNotFound()
Input	keyUsername,keyPassword
Output	verificationStatus
Algorithm	1. Start 2. If keyUsername && keyPassword null 2.1 If keyUsername && keyPassword not matched 2.2 verificationStatus = "fail" 3. End

4.2.1.2 Sequence Diagram

a) SD001: Sequence diagram for Login

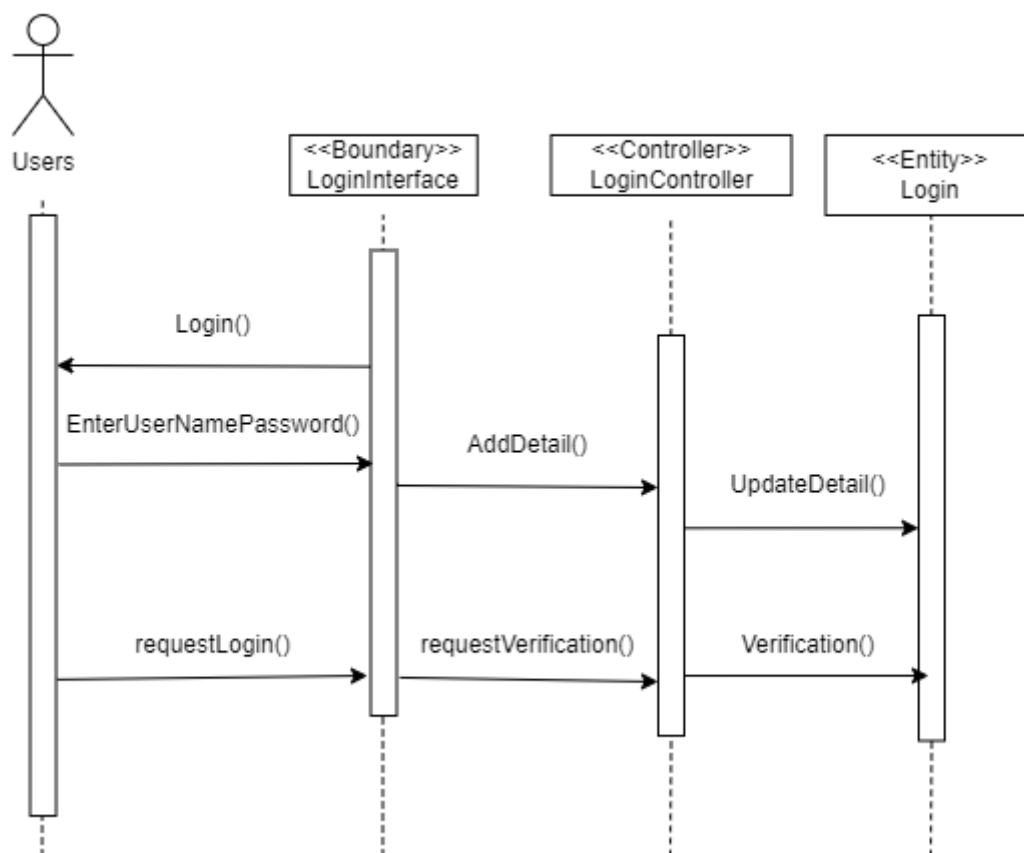


Figure 4.4: Sequence Diagram for <Login>

b) SD002: Sequence diagram for Verification

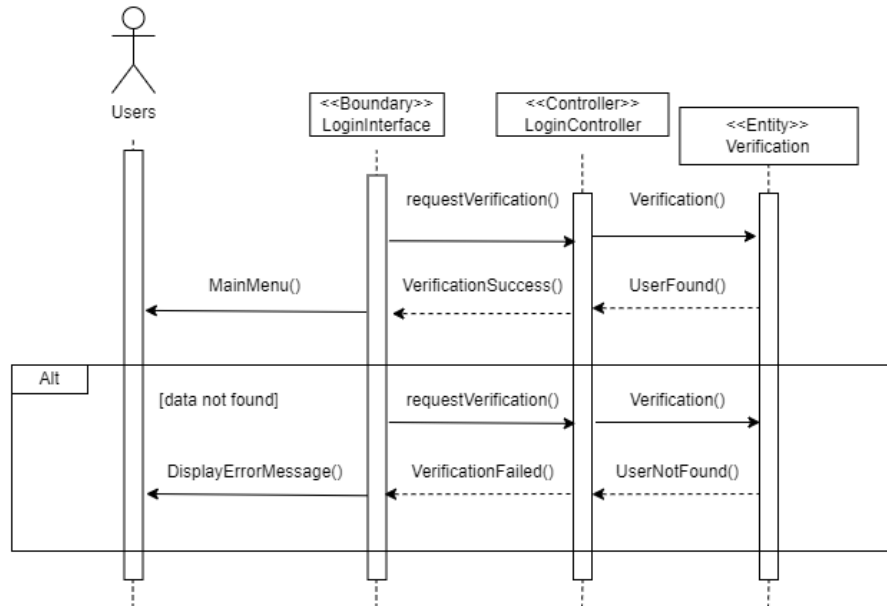


Figure 4.5: Sequence Diagram for <Verification>

4.2.2 P002: <Communication> Subsystem

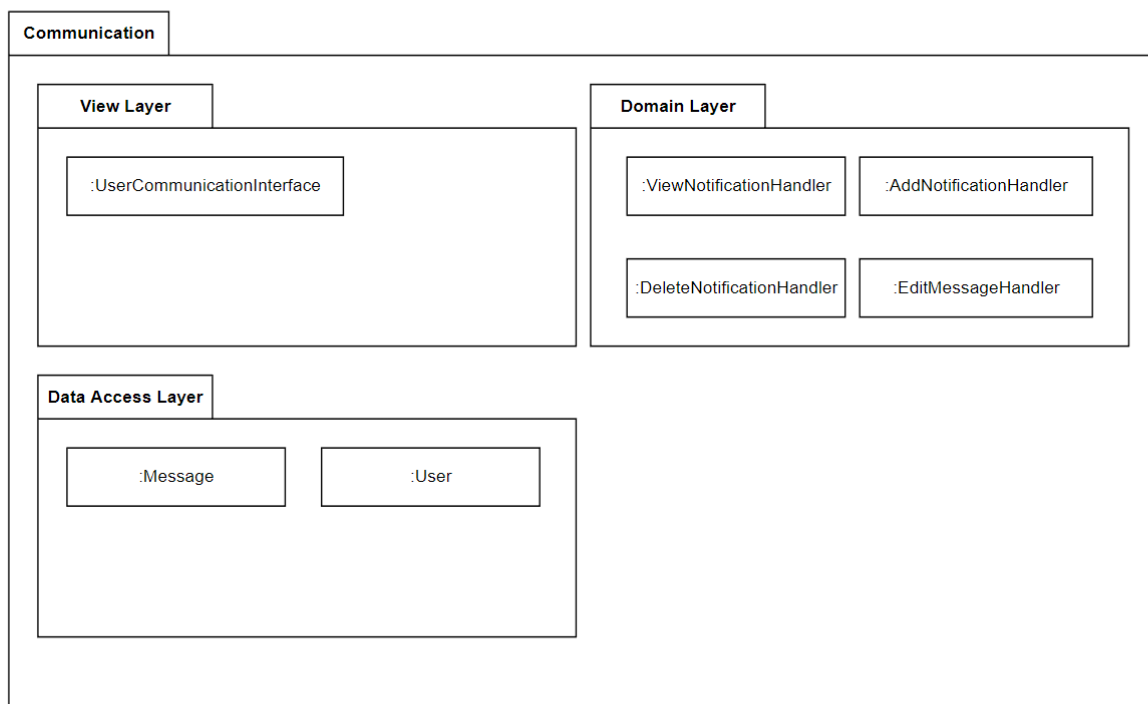


Figure 4.6: Package Diagram for <Communication> Subsystem

4.2.2.1 Class Diagram

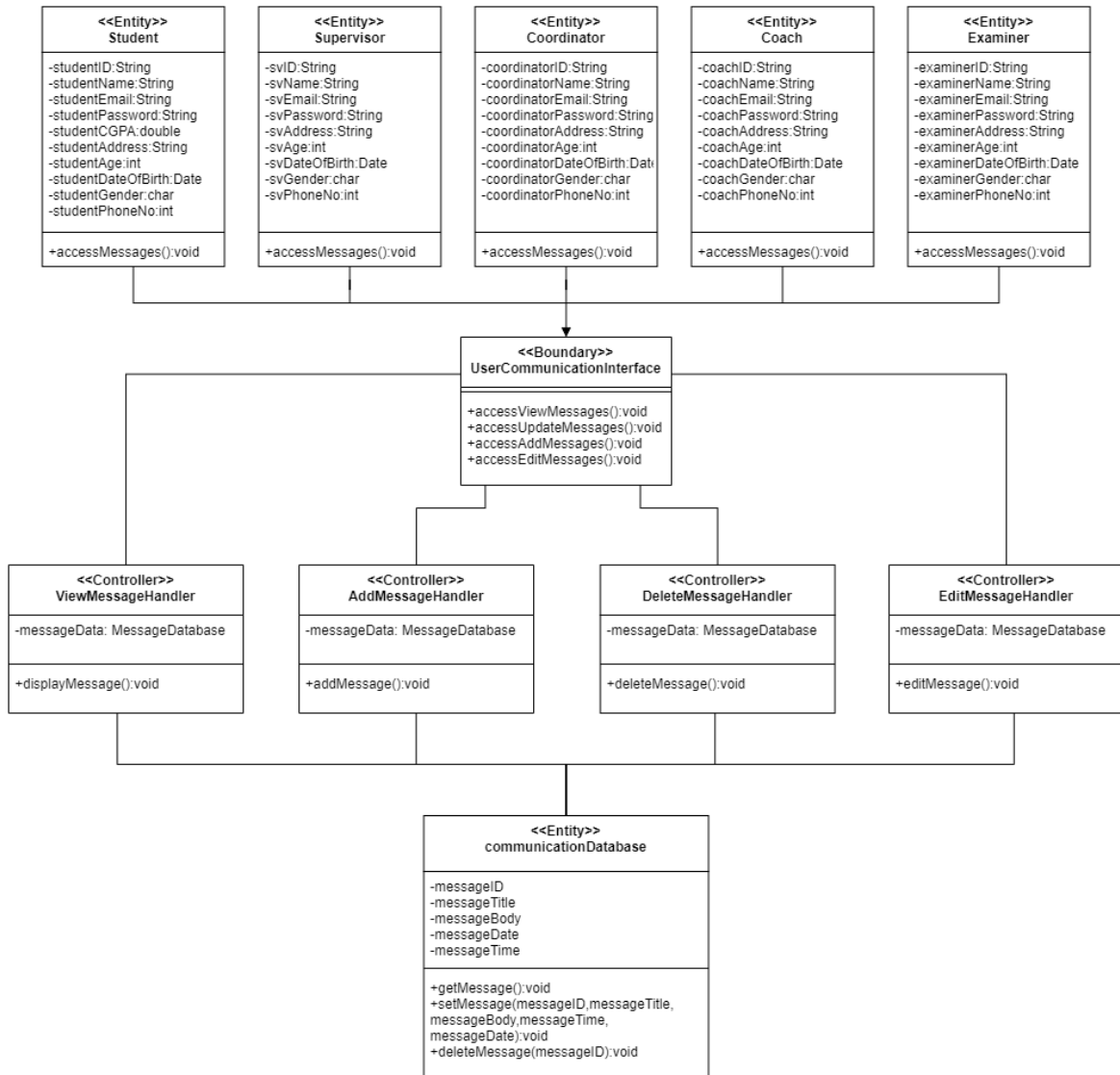


Figure 4.7: Class Diagram for <Communication> Subsystem

Entity Name	Student
Method Name	accessMessages
Input	studentID, studentPassword
Output	messageData
Algorithm	1. Start 2. Student logs in to system with ID and password 3. Student chooses to view, add, delete, or edit message 4. If student chooses to view message, communication database will get and display messages 5. Else if student chooses to add message, student can add message which will be set by communication database

	6. Else if student chooses to delete message, student can delete message which will be removed by communication database 7. Else if student chooses to edit message, student can edit message by resetting the message data in communication database 8. End
--	--

Entity Name	Supervisor
Method Name	accessMessages
Input	svID, svPassword
Output	messageData
Algorithm	1. Start 2. Supervisor logs in to system with ID and password 3. Supervisor chooses to view, add, delete, or edit message 4. If supervisor chooses to view message, communication database will get and display messages 5. Else if supervisor chooses to add message, supervisor can add message which will be set by communication database 6. Else if supervisor chooses to delete message, supervisor can delete message which will be removed by communication database 7. Else if supervisor chooses to edit message, supervisor can edit message by resetting the message data in communication database 8. End

Entity Name	Coordinator
Method Name	accessMessages
Input	coordinatorID, coordinatorPassword
Output	messageData
Algorithm	1. Start 2. Coordinator logs in to system with ID and password 3. Coordinator chooses to view, add, delete, or edit message 4. If coordinator chooses to view message, communication database will get and display messages 5. Else if coordinator chooses to add message, coordinator can add message which will be set by communication database 6. Else if coordinator chooses to delete message, coordinator can delete message which will be removed by communication database 7. Else if coordinator chooses to edit message, coordinator can edit message by resetting the message data in communication database 8. End

Entity Name	Coach
Method Name	accessMessages

Input	coachID, coachPassword
Output	messageData
Algorithm	1. Start 2. Coach logs in to system with ID and password 3. Coach chooses to view, add, delete, or edit message 4. If coach chooses to view message, communication database will get and display messages 5. Else if coach chooses to add message, coach can add message which will be set by communication database 6. Else if coach chooses to delete message, coach can delete message which will be removed by communication database 7. Else if coach chooses to edit message, coach can edit message by resetting the message data in communication database 8. End

Entity Name	Examiner
Method Name	accessMessages
Input	examinerID, examinerPassword
Output	messageData
Algorithm	1. Start 2. Examiner logs in to system with ID and password 3. Examiner chooses to view, add, delete, or edit message 4. If examiner chooses to view message, communication database will get and display messages 5. Else if examiner chooses to add message, examiner can add message which will be set by communication database 6. Else if examiner chooses to delete message, examiner can delete message which will be removed by communication database 7. Else if examiner chooses to edit message, examiner can edit messages by resetting the message data in communication database 8. End

Entity Name	UserCommunicationInterface
Method Name	accessViewMessages, accessAddMessages, accessDeleteMessages, accessEditMessages
Input	-
Output	-
Algorithm	1. Start 2. Interface receives choice by user 3. If user chooses view message, go to viewMessageHandler 4. Else if user chooses add message, go to addMessageHandler 5. Else if user chooses delete message, go to deleteMessageHandler 6. Else if user chooses edit message, go to editMessageHandler 7. End

Entity Name	iewMessageHandler
Method Name	displayMessage
Input	-
Output	-
Algorithm	1. Start 2. viewMessageHandler will display message from communication database 3. End

Entity Name	addMessageHandler
Method Name	addMessage
Input	-
Output	-
Algorithm	1. Start 2. addMessageHandler will add new message data to communication database 3. End

Entity Name	deleteMessageHandler
Method Name	deleteMessage
Input	-
Output	-
Algorithm	1. Start 2. deleteMessageHandler will delete existing message data from communication database 3. End

Entity Name	editMessageHandler
Method Name	editMessage
Input	-
Output	-
Algorithm	1. Start 2. editMessageHandler will edit existing message data from communication database 3. End

Entity Name	MessageDatabase
Method Name	getMessage, setMessage
Input	messageData
Output	messageData
Algorithm	1. Start 2. Database will receive command from user

	3. If user wants to view message, communication database will get and display message data 4. Else if user wants to add message, communication database will set new message into message data in communication database 5. Else if user wants to delete message, communication database will delete message by messageID 6. Else if user wants to edit message, communication database will set new message to the existing message
--	---

4.2.2.2 Sequence Diagram

a) SD003: Sequence diagram for View Message

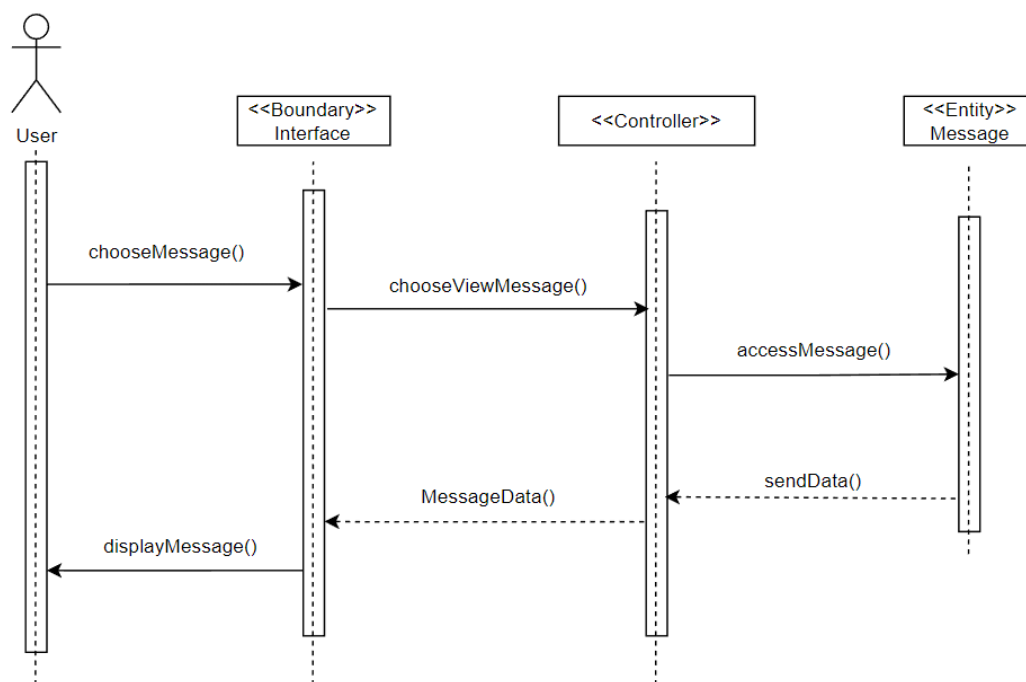


Figure 4.8: Sequence Diagram for <View Message>

b) SD004: Sequence diagram for Add Message

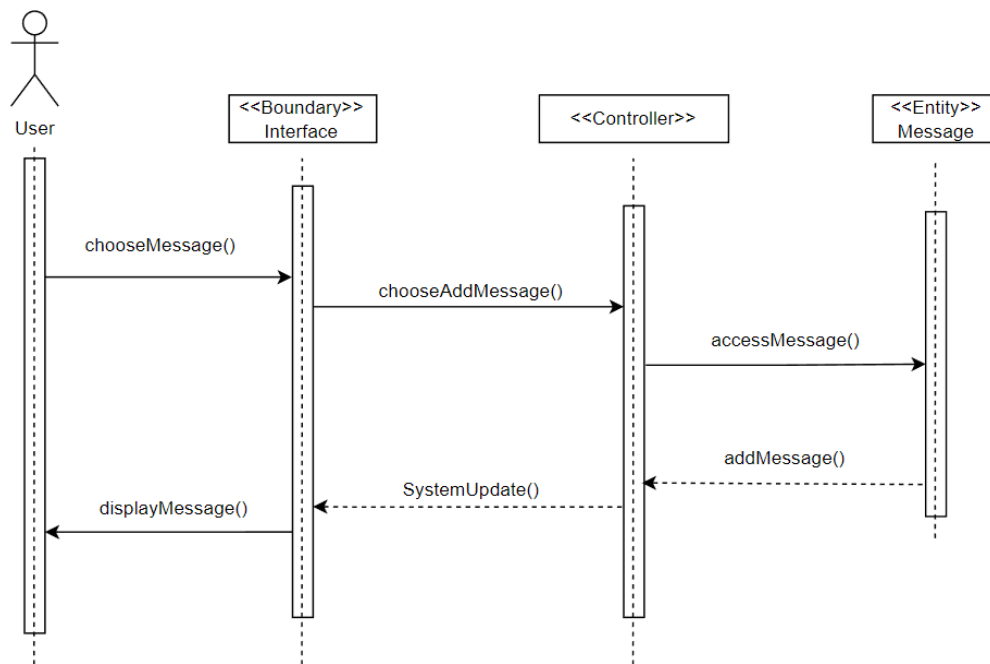


Figure 4.9: Sequence Diagram for <AddMessage>

c) SD005: Sequence diagram for Delete Message

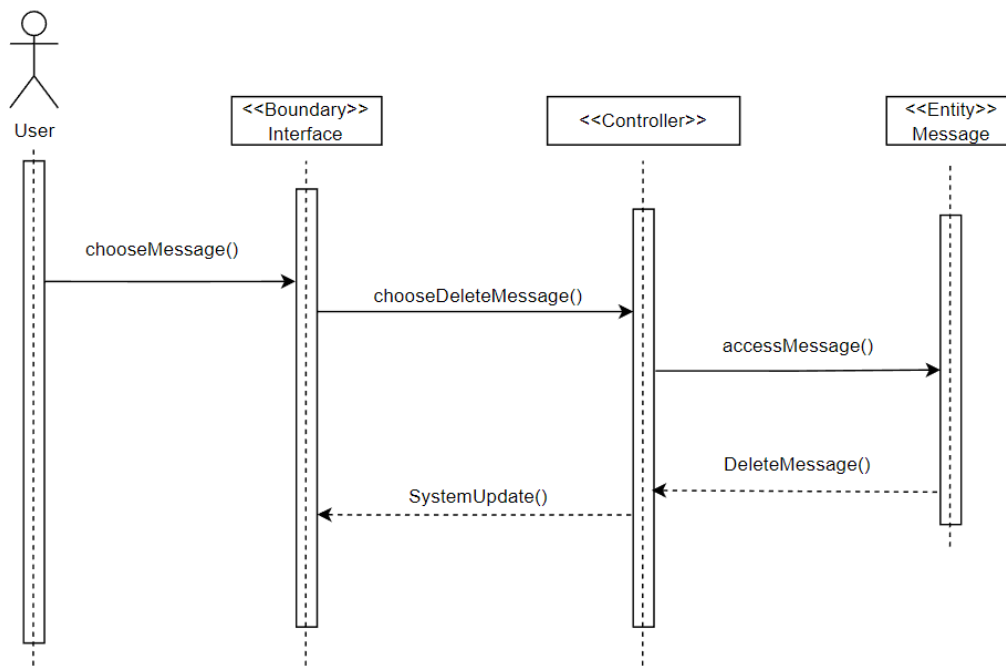


Figure 4.10: Sequence Diagram for <Delete Message>

d) SD006: Sequence diagram for Edit Message

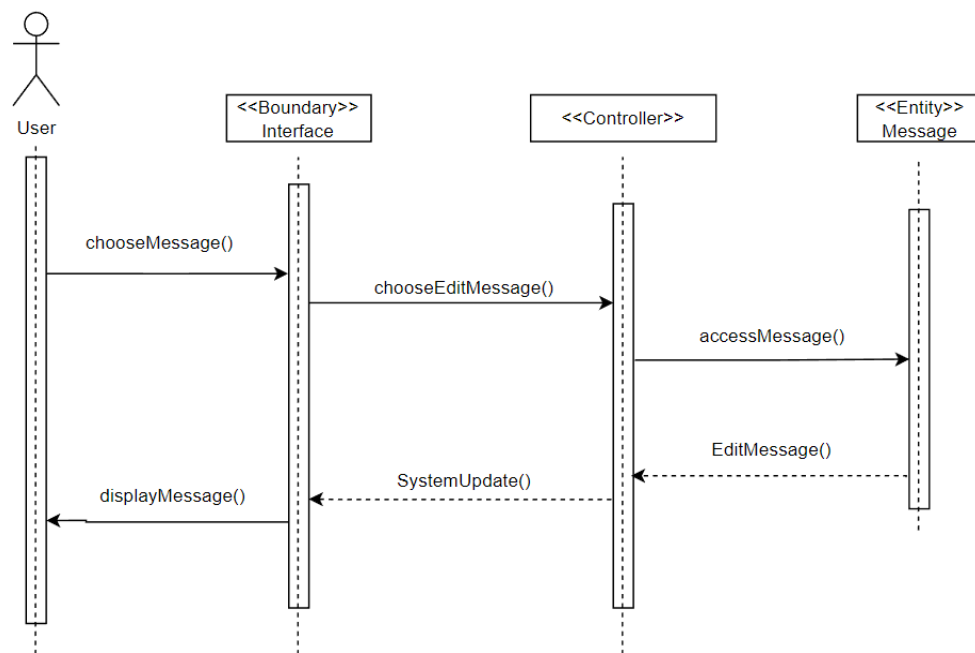


Figure 4.11: Sequence Diagram for <Edit Message>

4.2.3 P003: <Notification> Subsystem

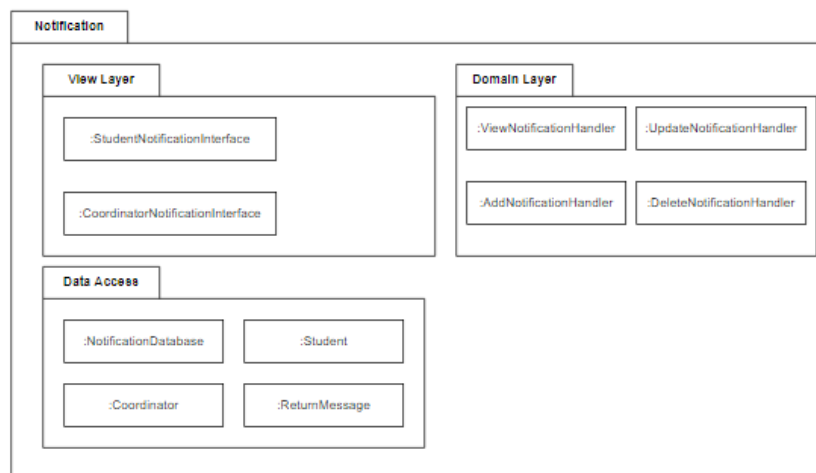


Figure 4.12: Package Diagram for <Notification> Subsystem

4.2.3.1 Class Diagram

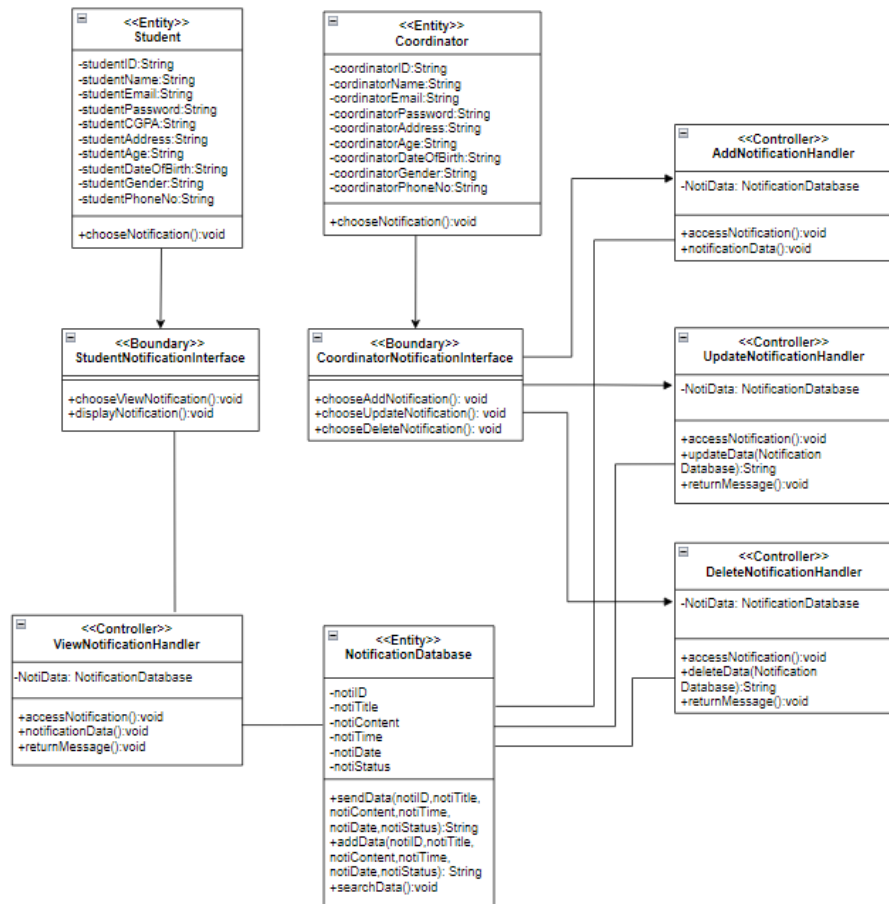


Figure 4.13: Class Diagram for <Notification> Subsystem

Entity Name	Student
Method Name	chooseNotification()
Input	GUI Input
Output	-
Algorithm	1. Start 2. Student choose notification 3. End

Entity Name	Coordinator
Method Name	chooseNotification()
Input	GUI Input
Output	-
Algorithm	1. Start 2. Coordinator choose notification 3. End

Entity Name	NotificationDatabase
--------------------	----------------------

Method Name	sendData() addData() searchData()
Input	notiID, notiTitle, notiContent, notiTime, notiDate, notiStatus
Output	-
Algorithm	1. Start 2. Controller request access to database 3. If the controller wants to add data, it will add it to the database 4. If controller wants data it will search the data using searchData 5. Existed data will be sent back to controller 6. End

Boundary Name	StudentNotificationInterface
Method Name	chooseViewNotification() displayNotification()
Input	-
Output	displayNotification()
Algorithm	1. Start 2. Choose view notification 3. Data from ViewNotificationHandler is shown using displayNotification() 4. End

Boundary Name	CoordinatorNotificationInterface
Method Name	chooseAddNotification() chooseUpdateNotification() chooseDeleteNotification()
Input	-
Output	-
Algorithm	1. Start 2. If choose add notification, go to AddNotificationHandler 3. If choose update notification, go to UpdateNotificationHandler 4. If choose delete notification, go to DeleteNotificationHandler 5. End

Controller Name	ViewNotificationHandler
Method Name	accessNotification() notificationData() returnMessage()
Input	-
Output	-
Algorithm	1. Start 2. Request access data at NotificationDatabase 3. If data exist data will be sent to StudentNotificationInterface

	4. Else it will returnMessage showing error 5. End
--	---

Controller Name	AddNotificationHandler
Method Name	accessNotification() notificationData()
Input	-
Output	-
Algorithm	1. Start 2. Request access data at NotificationDatabase 3. End

Controller Name	UpdateNotificationHandler
Method Name	accessNotification() updateNotificationData() returnMessage()
Input	-
Output	-
Algorithm	1. Start 2. Request access data at NotificationDatabase 3. If data exist coordinator can update data 4. Else it will returnMessage showing error 5. End

Controller Name	DeleteNotificationHandler
Method Name	accessNotification() deletenotificationData() returnMessage()
Input	-
Output	-
Algorithm	1. Start 2. Request access data at NotificationDatabase 3. If data exist coordinator can update data 4. Else it will returnMessage showing error 5. End

4.2.3.2 Sequence Diagram

a)SD007 : Sequence diagram for NotificationDatabase

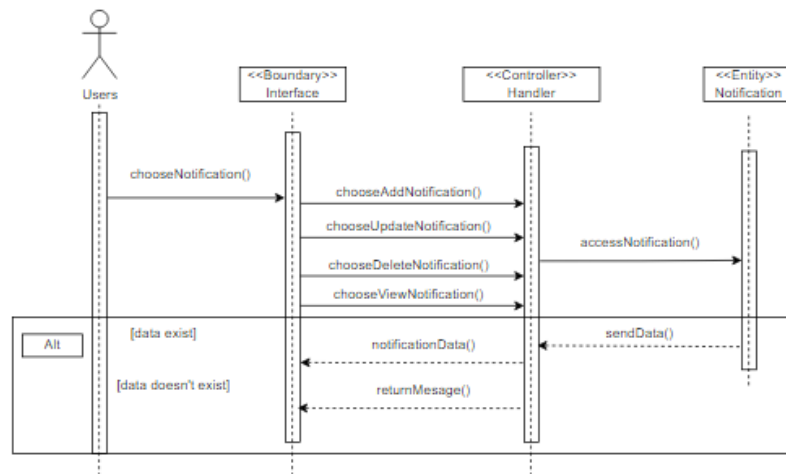


Figure 4.14: Sequence Diagram for <NotificationDatabase>

b)SD008 : Sequence diagram for ViewNotification

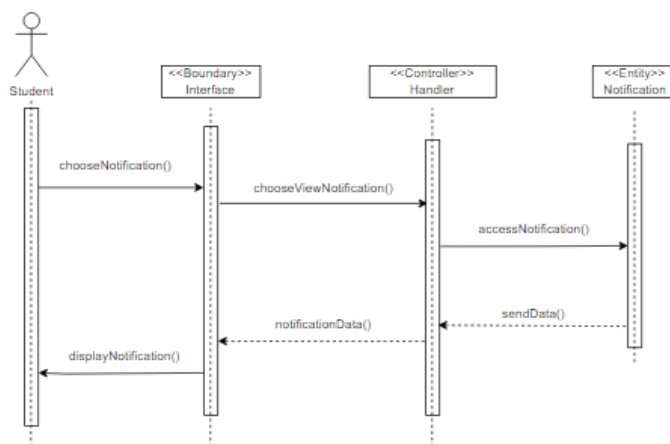


Figure 4.15: Sequence Diagram for <ViewNotification>

c)SD009 : Sequence diagram for AddNotification

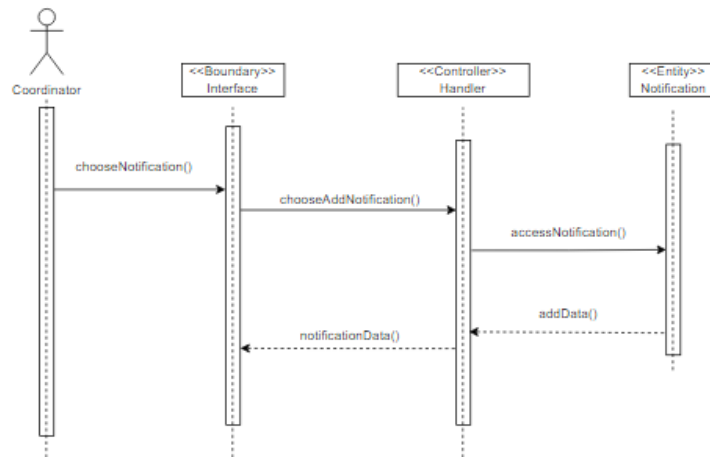


Figure 4.16: Sequence Diagram for <AddNotification>

d)SD010 : Sequence diagram for UpdateNotification

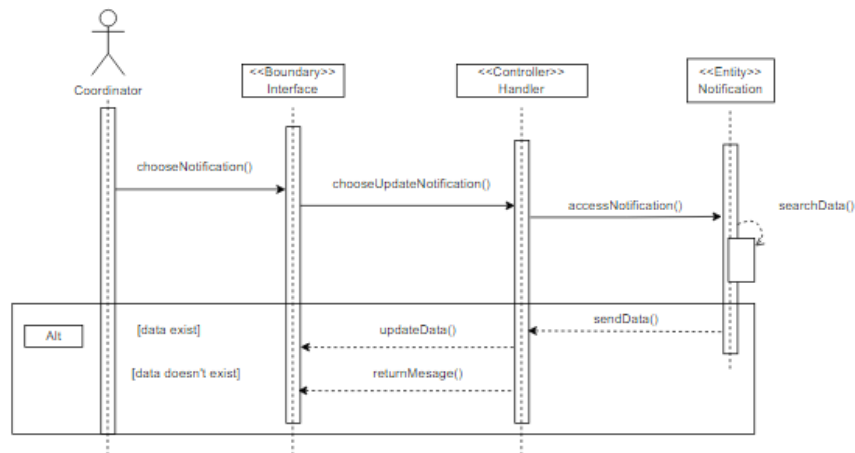


Figure 4.17: Sequence Diagram for <UpdateNotification>

e)SD011 : Sequence diagram for DeleteNotification

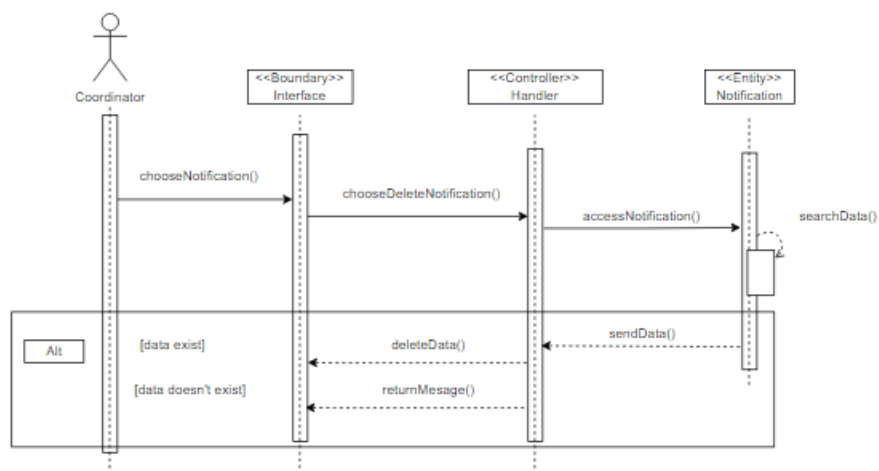


Figure 4.18: Sequence Diagram for <DeleteNotification>

f)SD012 : Sequence diagram for ReturnMessage

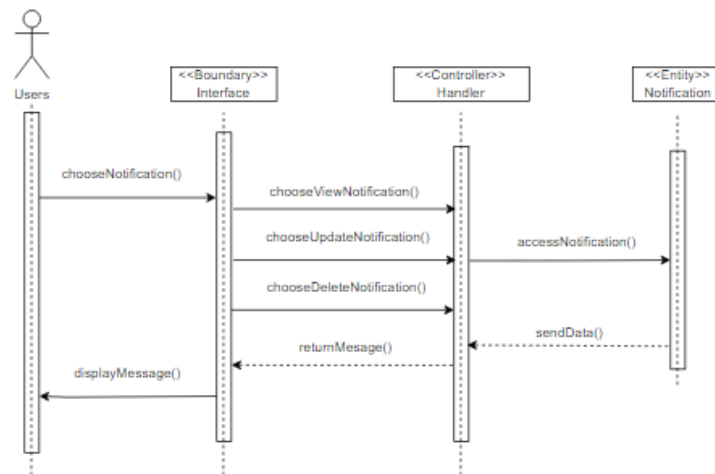


Figure 4.19: Sequence Diagram for <ReturnMessage>

4.2.4 P004: <Assessment> Subsystem

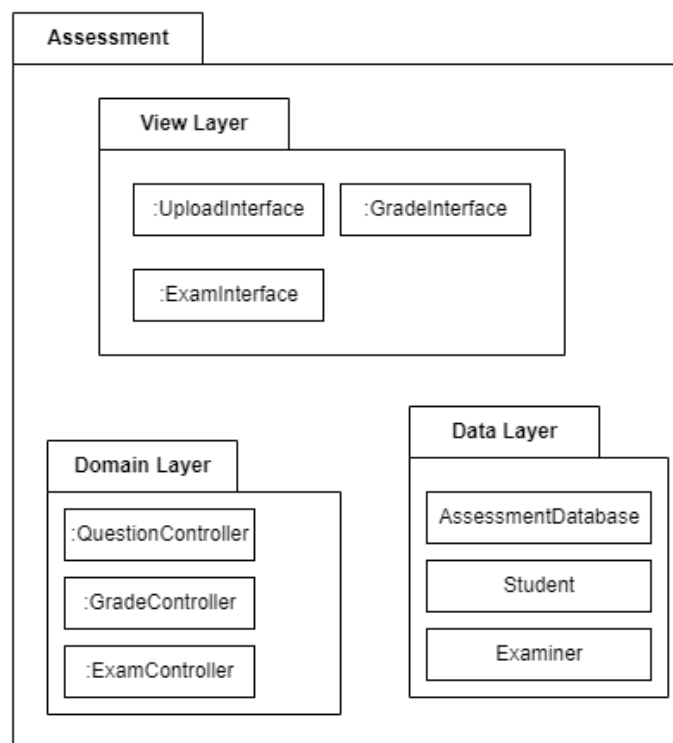


Figure 4.20: Package Diagram for <Assessment> Subsystem

4.2.4.1 Class Diagram

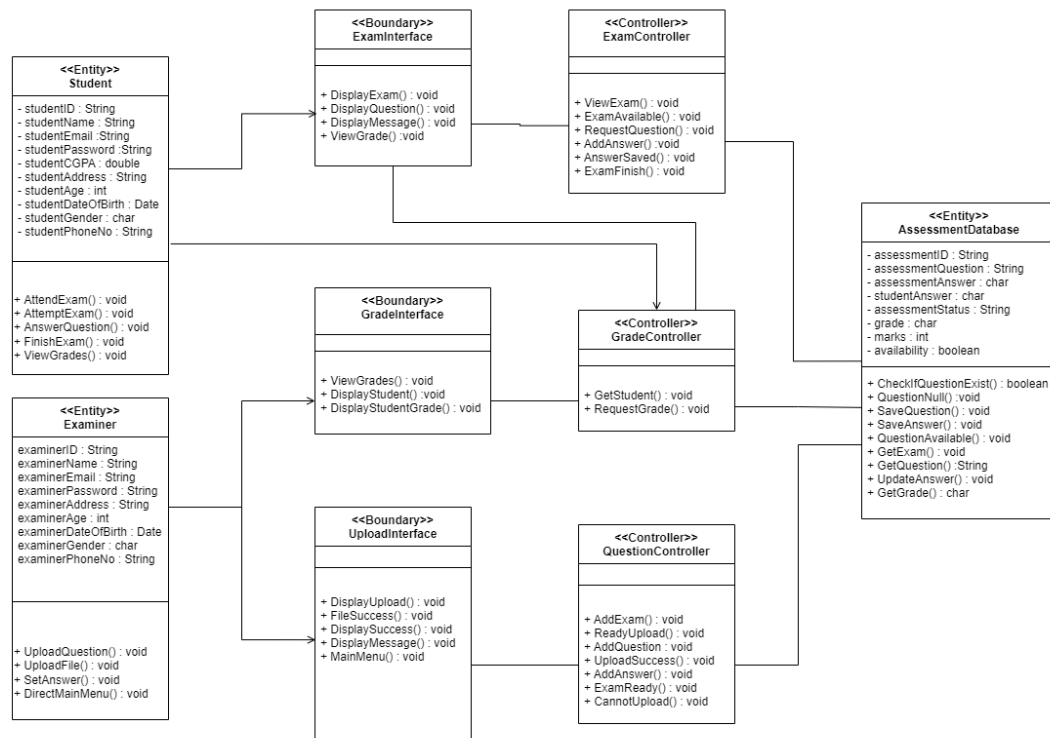


Figure 4.21: Class Diagram for <Assessment> Subsystem

Entity Name	Student
Method Name	AttendExam()
Input	GUI Input
Output	
Algorithm	1. Start 2. User clicks on the attend exam button 3. Exam information is displayed 4. End

Entity Name	Student
Method Name	AttemptExam()
Input	GUI Input
Output	
Algorithm	1. Start

	2. User clicks on the attempt exam button 3. Exam is started and questions is displayed 4. End
--	--

Entity Name	Student
Method Name	AnswerQuestion()
Input	studentAnswer
Output	
Algorithm	1. Start 2. User enters their answer for the question 3. studentAnswer is updated 4. End

Entity Name	Student
Method Name	FinishExam()
Input	GUI Input
Output	
Algorithm	1. Start 2. User clicks on the finish button 3. Message indicate exam has finished is displayed 4. End

Entity Name	Student
Method Name	ViewGrades()
Input	
Output	grade
Algorithm	1. Start 2. Grade is obtained from the database 3. Grade is displayed 4. End

Entity Name	Examiner
Method Name	UploadQuestion()
Input	GUI Input
Output	
Algorithm	1. Start 2. User clicks on the upload question button 3. Upload menu is displayed 4. End

Entity Name	Examiner
Method Name	UploadFile()

Input	assessmentQuestion
Output	
Algorithm	1. Start 2. If assessmentQuestion not available 2.1 User enters the question for the exam 2.2 The assessment question is updated 2.3 Success message is displayed 3. Else 3.1 User cannot upload the question 3.2 Message about the question already available is displayed 3.3 User directed to the main menu. 4. End

Entity Name	Examiner
Method Name	DirectMainMenu()
Input	GUI Input
Output	
Algorithm	1. Start 2. User clicks on the back to main menu button 3. Main menu is displayed 4. End

Entity Name	AssessmentDatabase
Method Name	CheckIfQuestionExist()
Input	
Output	availability
Algorithm	1. Start 2. If assessmentQuestion is not null 2.1 return true 3. End

Entity Name	AssessmentDatabase
Method Name	QuestionNull()
Input	
Output	availability
Algorithm	1. Start 2. if assessmentQuestion is null 2.1 return false 3. End

Entity Name	AssessmentDatabase
Method Name	SaveQuestion()

Input	assessmentQuestion
Output	
Algorithm	1. Start 2. Assessment question entered by the user is updated in the database 3. End

Entity Name	AssessmentDatabase
Method Name	SaveAnswer()
Input	assessmentAnswer
Output	
Algorithm	1. Start 2. Assessment answer entered by the user is updated in the database 3. End

Entity Name	AssessmentDatabase
Method Name	QuestionAvailable()
Input	
Output	
Algorithm	1. Start 2. If assessmentQuestion not null 2.1 availability = true 3. End

Entity Name	AssessmentDatabase
Method Name	GetExam()
Input	
Output	assessmentID
Algorithm	1. Start 2. assessmentID is obtained from the database 3. assessmentID is sent to the controller 4. End

Entity Name	AssessmentDatabase
Method Name	GetQuestion()
Input	
Output	assessmentQuestion
Algorithm	1. Start 2. assessmentQuestion is obtained from the database 3. assessmentQuestion is sent to the controller 4. End

Entity Name	AssessmentDatabase
Method Name	UpdateAnswer()
Input	studentAnswer
Output	
Algorithm	1. Start 2. studentAnswer is updated in the database 3. End

Entity Name	AssessmentDatabase
Method Name	GetGrade()
Input	
Output	grade
Algorithm	1. Start 2. Grade is calculated based on the amount of correct answer by the student 3. If marks<=40 3.1 grade='F' 4. else if marks<=50 4.1 grade='D' 5. else if marks<=60 5.1 grade='C' 6. else if marks<=80 6.1 grade='B' 7. else if marks<=100 7.1 grade='A' 8. return grade 9. End

4.2.4.2 Sequence Diagram

a)SD013 : Sequence diagram for Upload Questions

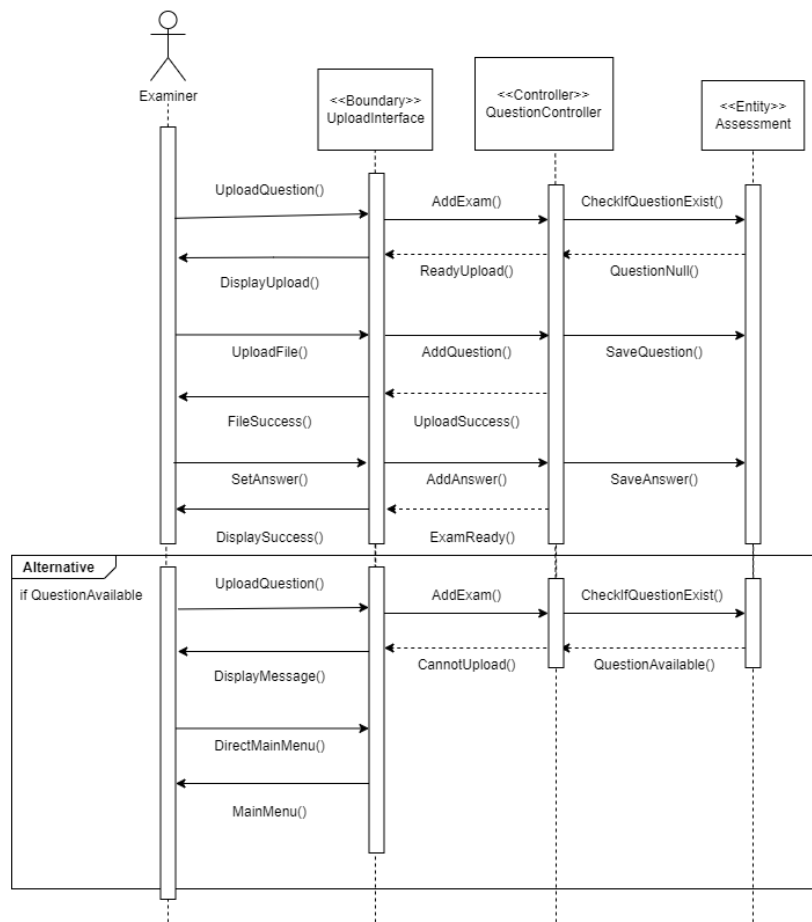


Figure 4.22: Sequence Diagram for <Upload Question>

b)SD014 : Sequence diagram for Take Exam

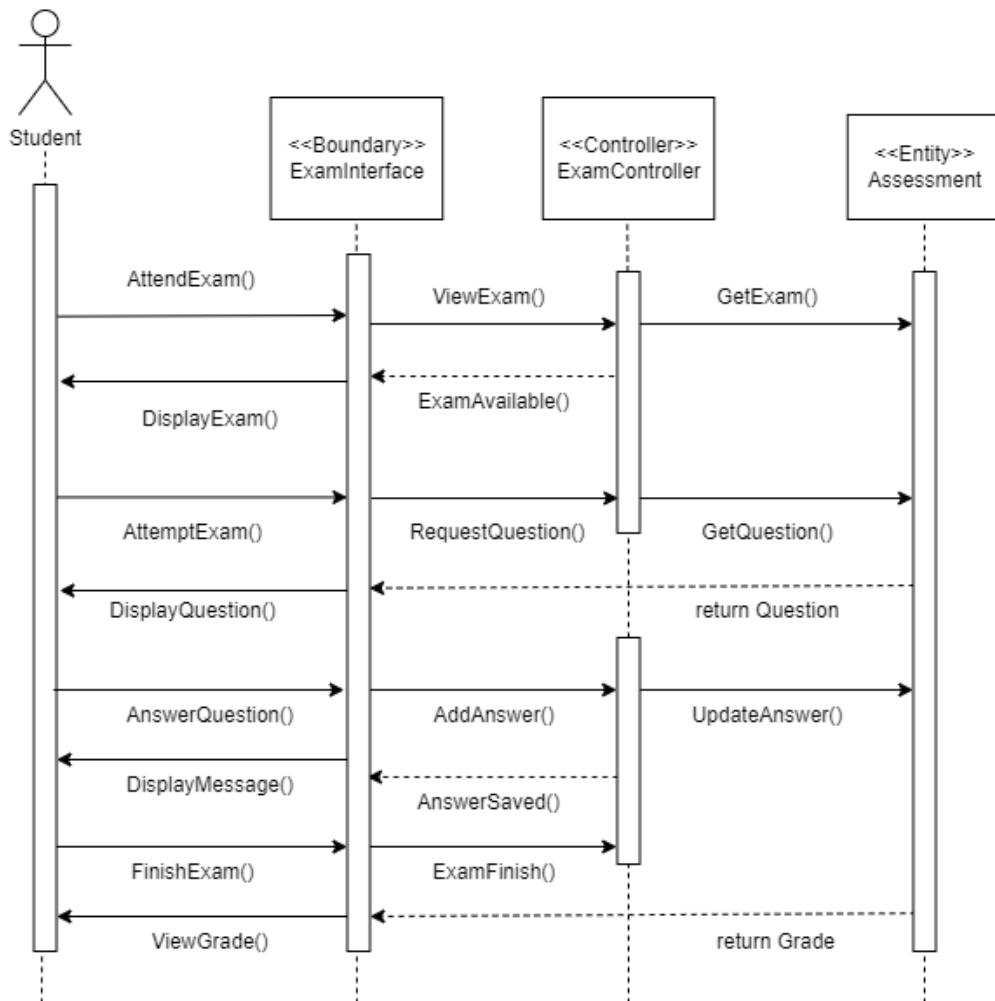


Figure 4.23: Sequence Diagram for <Take Exam>

c)SD015 : Sequence diagram for View Grade

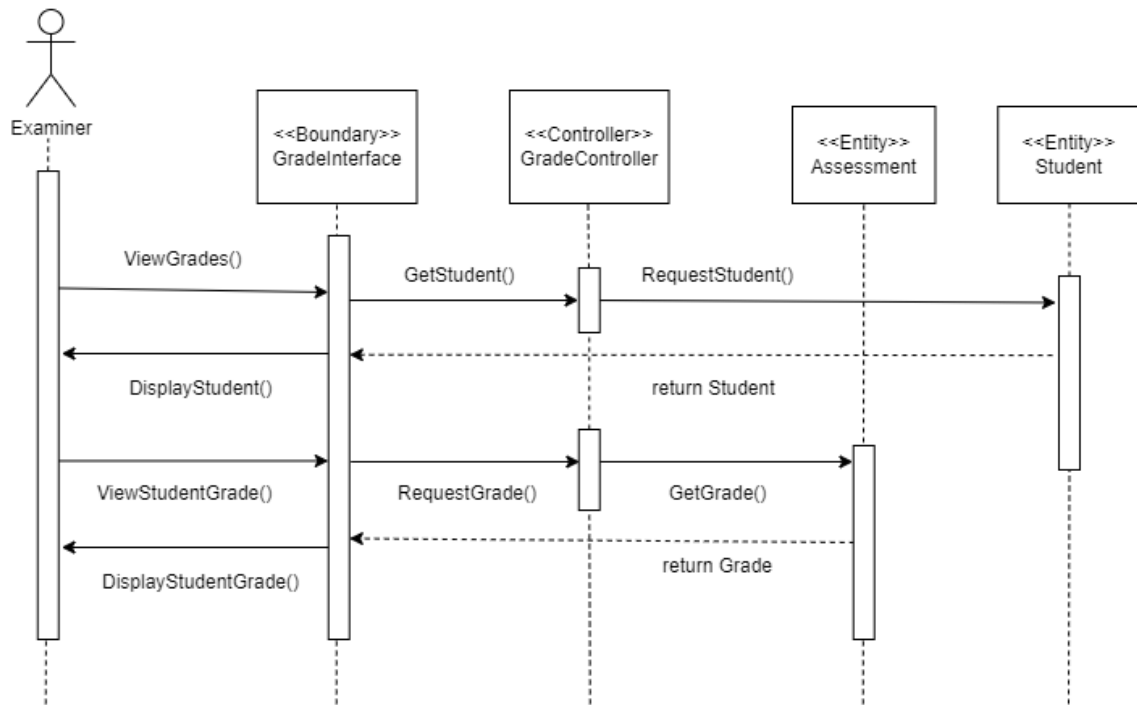


Figure 4.24: Sequence Diagram for <Vire Grades>

5.0 Data Design

5.1 Data Description

The major data or systems entities are stored into a relational database named Inferno 2u2i Final Year Project with Industry (FYP-I) Management System, processed and organized into 10 entities as listed in Table 5.1.

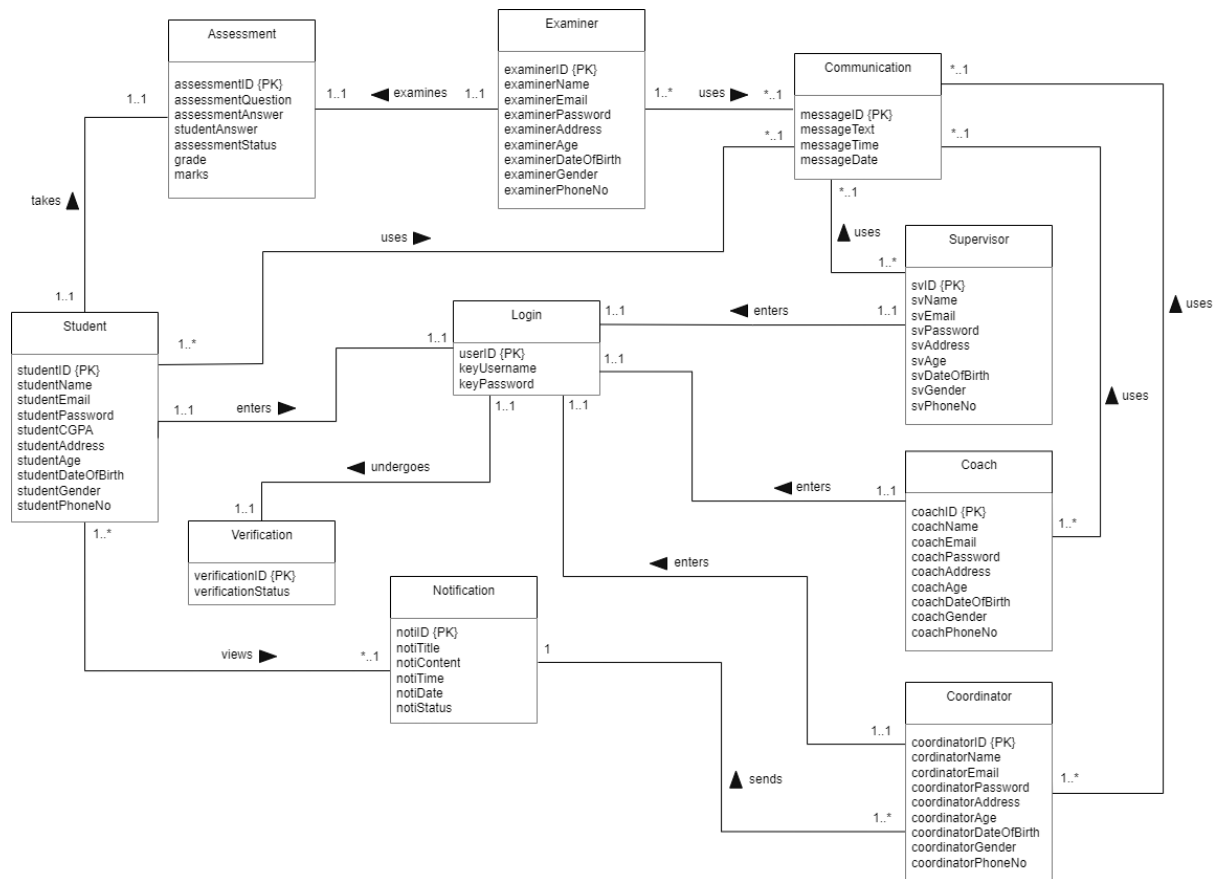


Figure 5.1: Entity Relationship Diagram for <Inferno 2u2i Final Year Project with Industry (FYP-I) Management System>

Table 5.1: Description of Entities in the Database

No.	Entity Name	Description
1.	Student	Store the information of the student
2.	Coach	Store the information of the coach
3.	Supervisor	Store the information of the supervisor
4.	Examiner	Store the information of the examiner
5.	Coordinator	Store the information of the coordinator
6.	Assessment	Store the information related to assessment
7.	Communication	Store the messages between users
8.	Notification	Store the information about notifications
9.	Login	Store the information related to login
10	Verification	Store the information related to verification

5.2 Data Dictionary

5.2.1 Entity: <Student>

Attribute Name	Type	Description
studentID	VARCHAR2(10)	Student ID, Primary Key
studentName	VARCHAR2(50)	Student full name
studentEmail	VARCHAR2(20)	Student email
studentPassword	VARCHAR2(20)	Student password
studentCGPA	NUMERIC(2,2)	Student CGPA
studentAddress	VARCHAR(100)	Student current address
studentAge	INT	Student age
studentDateOfBirth	DATE	Student date of birth
studentGender	CHAR(1)	Student gender
studentPhoneNo	VARCHAR(15)	Student phone number

5.2.2 Entity: <Coach>

Attribute Name	Type	Description
coachID	VARCHAR2(10)	Coach ID, Primary Key
coachName	VARCHAR2(50)	Coach full name
coachEmail	VARCHAR2(20)	Coach email
coachPassword	VARCHAR2(20)	Coach password
coachAddress	VARCHAR2(100)	Coach current address
coachAge	INT	Coach age

coachDateOfBirth	DATE	Coach date of birth
coachGender	CHAR(1)	Coach gender
coachPhoneNo	VARCHAR2(15)	Coach phone number

5.2.3 Entity: <Supervisor>

Attribute Name	Type	Description
svID	VARCHAR2(10)	Supervisor ID, Primary Key
svName	VARCHAR2(50)	Supervisor full name
svEmail	VARCHAR2(20)	Supervisor email
svPassword	VARCHAR2(20)	Supervisor password
svAddress	VARCHAR2(100)	Supervisor current address
svAge	INT	Supervisor age
svDateOfBirth	DATE	Supervisor date of birth
svGender	CHAR(1)	Supervisor gender
svPhoneNo	VARCHAR2(15)	Supervisor phone number

5.2.4 Entity: <Examiner>

Attribute Name	Type	Description
examinerID	VARCHAR2(10)	Examiner ID, Primary Key
examinerName	VARCHAR2(50)	Examiner full name
examinerEmail	VARCHAR2(20)	Examiner email
examinerPassword	VARCHAR2(20)	Examiner password
examinerAddress	VARCHAR2(100)	Examiner current address
examinerAge	INT	Examiner age
examinerDateOfBirth	DATE	Examiner date of birth
examinerGender	CHAR(1)	Examiner gender
examinerPhoneNo	VARCHAR2(15)	Examiner phone number

5.2.5 Entity: <Coordinator>

Attribute Name	Type	Description
coordinatorID	VARCHAR2(10)	Coordinator ID, Primary Key
coordinatorName	VARCHAR2(50)	Coordinator full name
coordinatorEmail	VARCHAR2(20)	Coordinator email
coordinatorPassword	VARCHAR2(20)	Coordinator password
coordinatorAddress	VARCHAR2(100)	Coordinator current address
coordinatorAge	INT	Coordinator age
coordinatorDateOfBirth	DATE	Coordinator date of birth

coordinatorGender	CHAR(1)	Coordinator gender
coordinatorPhoneNo	VARCHAR2(15)	Coordinator phone number

5.2.6 Entity: <Assessment>

Attribute Name	Type	Description
assessmentID	VARCHAR2(10)	Assessment ID, Primary Key
assessmentQuestion	VARCHAR2(200)	Question in the assessment
assessmentAnswer	CHAR(1)	Answer for the question
studentAnswer	CHAR(1)	Answer for the question by the student
assessmentStatus	VARCHAR(20)	Status of the assessment (exist/null/finish)
grade	CHAR(1)	Grade of the assessment obtained by student
marks	INT	Mark of the assessment obtained by student

5.2.7 Entity: <Communication>

Attribute Name	Type	Description
messageID	VARCHAR2(10)	Message ID, Primary Key
messageText	VARCHAR(200)	Text for the message
messageTime	TIME	Time of the message created
messageDate	DATE	Date of the message created

5.2.8 Entity: <Notification>

Attribute Name	Type	Description
notiID	VARCHAR2(10)	Notification ID, Primary Key
notiTitle	VARCHAR(50)	Title of the notification
notiContent	VARCHAR(100)	Content of the notification
notiTime	TIME	Time of the notification created
notiDate	DATE	Date of the notification created
notiStatus	VARCHAR(20)	Status of the notification created

5.2.9 Entity: <Login>

Attribute Name	Type	Description
userID	VARCHAR2(10)	User ID, Primary Key
keyUsername	VARCHAR2(20)	Username key in by user

keyPassword	VARCHAR2(20)	Password key in by user
-------------	--------------	-------------------------

5.2.10 Entity: <Verification>

Attribute Name	Type	Description
verificationID	VARCHAR2(10)	Verification ID, Primary Key
verificationStatus	VARCHAR2(20)	Status for the verification (true/false)

6.0 Interface Design

6.1 Overview of Interface

For login , where all the process started , the user whether student or lecturer will need to login by inserting their username and password. After that, the user is needed to click the button so that the process can be carried on. The system will be required to be verified by finding the data that synchronizes with the existing data. There will be two conditions where if the data have been found, the user will get the authorization to access the system and will be directed to the main menu which contains a lot of option to choose from such as notification , messaging and much more

If the data has not been found, there will be an error message displayed and it will be redirected to the login menu.

For communication, this is the platform where the users are able to communicate with each other through messaging. There will be four functions that can be used for communication such as View Message, Add Message ,Delete Message and Edit Message.The history of messaging and text will be saved into the database.

For View Message, the user is required to search for the message that the user wanted. If the message is nowhere to be found, the user is required to find the message again. Then, the message will be displayed on the screen.

For Add Message, the user is able to add any message to the other users whether they have any questions to ask to their lecturer regarding the status of their internships. This can be done by sending the message including photo, file and much more.

For Delete Message, the users are able to delete the message that they have sent whether it is a photo or file. Sometimes the users have mistakenly sent the messages to the wrong person. so the user needs to search for the message that they want to delete and just delete it. They can delete as much they want . There are no restrictions on deleting the message.

For Edit Message, the users have the rights to edit the messages that they have sent to other users. This can be done by searching the messages that they want to edit. Here the users are able to edit the message and there will be a notification pop out saying that the messages have been updated . This is to prevent any miscommunication between users such as student and lecturer or student and the company that they intern with.

For notification, this is where the user will get their notifications such as email. For Coordinator, the features are Add Notification, Update Notification, and Delete Notification,

while Students features are only View Notification. All data that are related to notification are saved in Notification Data.

For add notification, the coordinator must select Add Notification and it will send the request to the database to access, and the coordinator is able to add notification to the database.

For update notification, the coordinator must select Update Notification and it will send a request to the database to access, and the system will search for the available data. If the data exists, then the coordinator is able to update the notification, else it will return back an error message.

For delete notification, the coordinator must select Delete Notification and it will send a request to the database to access, and the system will search for the available data. If the data exists, then the coordinator is able to delete the notification, else it will return back an error message.

For view notification, the student must select View Notification and it will send a request to the database to access, and the system will search for available data. If the data exists, then the system will print out all the notification data until there is no more available data.

For assessment, this is where the students are able to take exams which are uploaded by the examiners. Features for assessment are Upload Question, Take Exam, and View Grades.

For the Upload Question, the examiner must select the button that they want to upload the question. If the question is not yet uploaded, the file containing the questions can be uploaded by clicking the upload button, then the answer for each question needs to be set by clicking the right answer for the questions. System then stores the questions and answers in the database. Else, if the questions is already uploaded, it will print out a message telling that the questions had been uploaded before, and then the examiner will be redirected to the main page.

For the Take Exam, the exam questions must already be uploaded to the database, and the student needs to click on the exam that they need to attend. Students then attempt the exam by clicking the start button, all questions that are displayed must be answered by the student. After finishing all questions, students are able to click the finish button to complete the Take Exam.

For the View Grades, the exam questions must already be finished by the students. The system will check the student's answer based on the answer sheet that has been uploaded to the database. The grades are obtained after checking is finished. If the student

wants to view grades, the grades will be displayed immediately after the exam. If the examiner wants to view a student's grade, a list of students that finished the exam will be displayed, and the examiner is able to click any student's name to view their grades.

6.2 Overall Interface Design

-Login Interface

The image displays two wireframe screenshots of a login interface, stacked vertically.

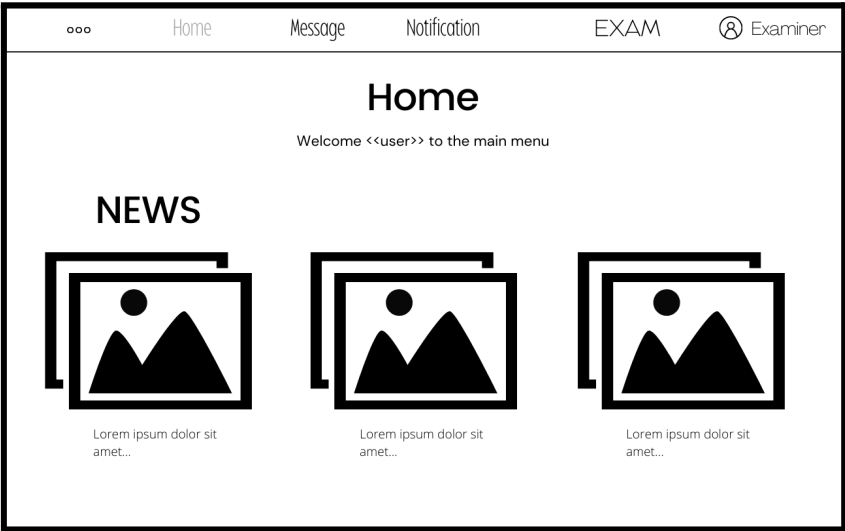
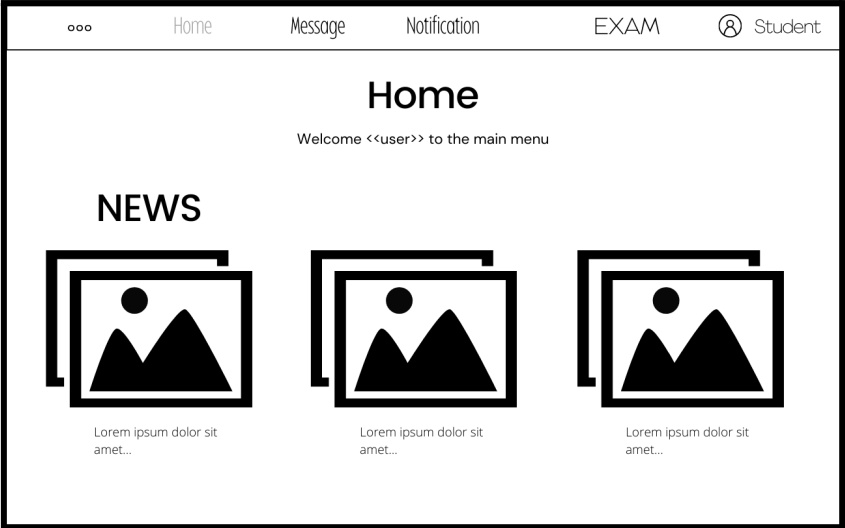
The top screenshot shows the 'Login' screen. It features a header with three dots (ooo) on the left. The main content area is centered and contains the following elements:

- The word 'Login' in a large, bold font.
- The text 'Sign in to continue' in a smaller font below 'Login'.
- A label 'USERNAME' above a text input field containing the text 'example@email.com'.
- A label 'PASSWORD' above a text input field containing six asterisks (*****).
- A black button with the text 'login' in white, positioned below the password field.

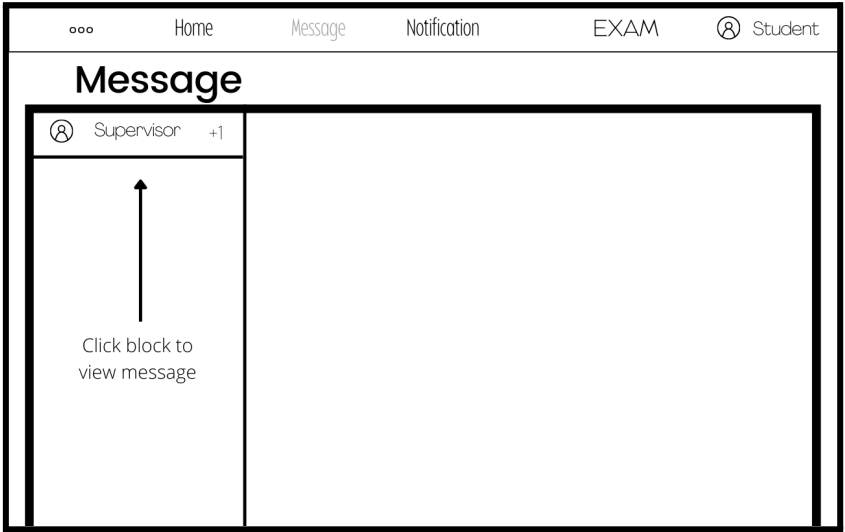
The bottom screenshot shows the 'Login Error' screen. It features a header with three dots (ooo) on the left. The main content area is centered and contains the following elements:

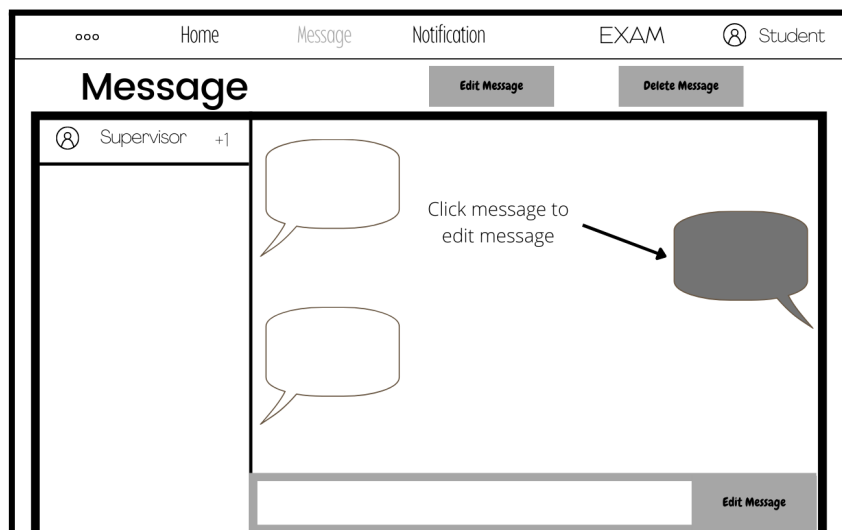
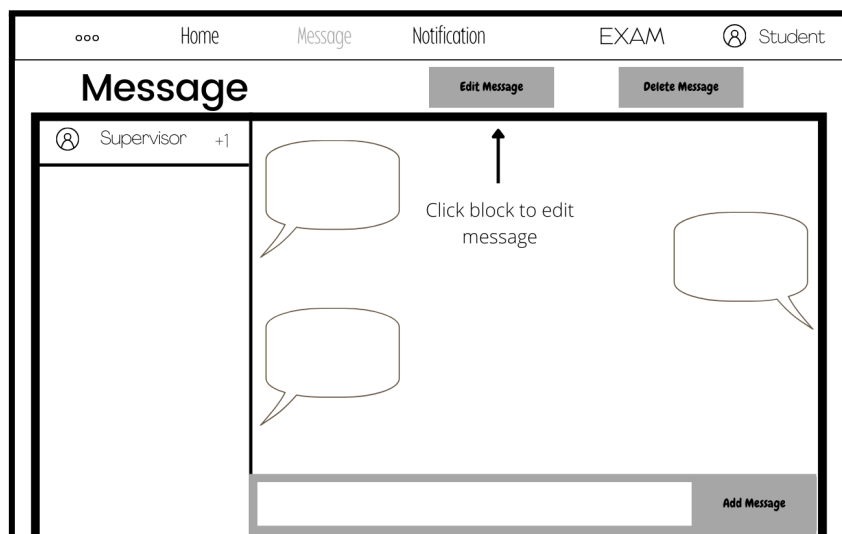
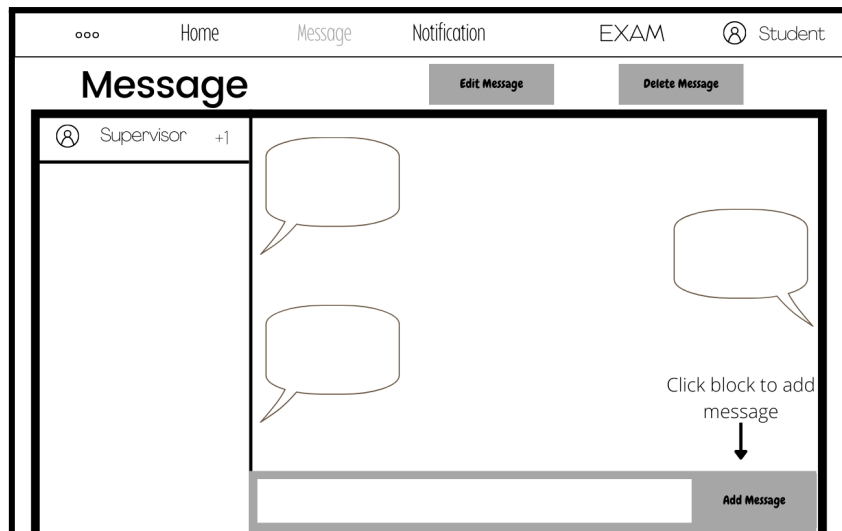
- A red circle with a white 'X' icon at the top.
- The text 'Login Error' in a large, bold font.
- The text 'Please enter the right username/password' in a smaller font below 'Login Error'.
- A label 'USERNAME' above a text input field containing the text 'example@email.com'.
- A label 'PASSWORD' above a text input field containing six asterisks (*****).
- A black button with the text 'try again' in white, positioned below the password field.

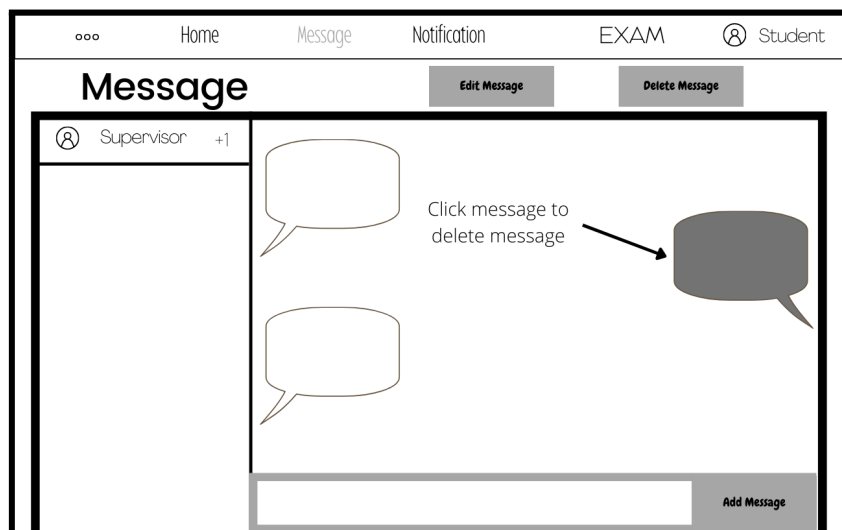
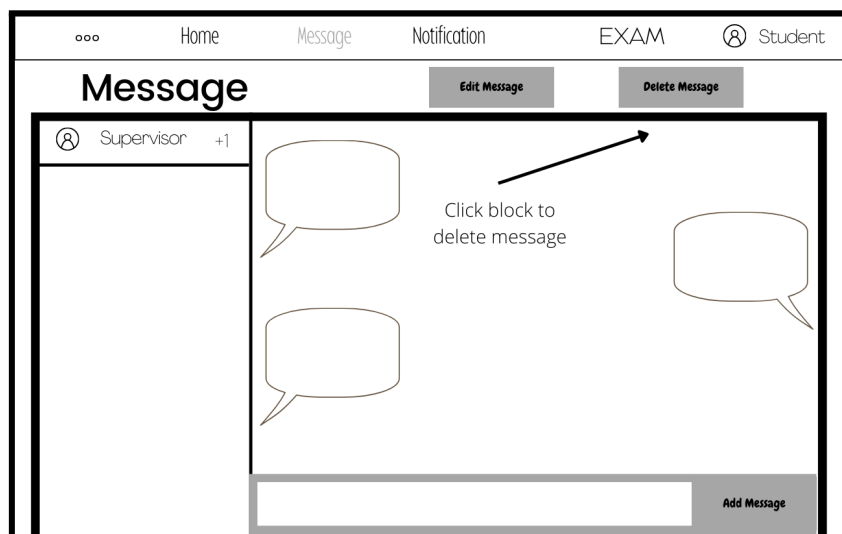
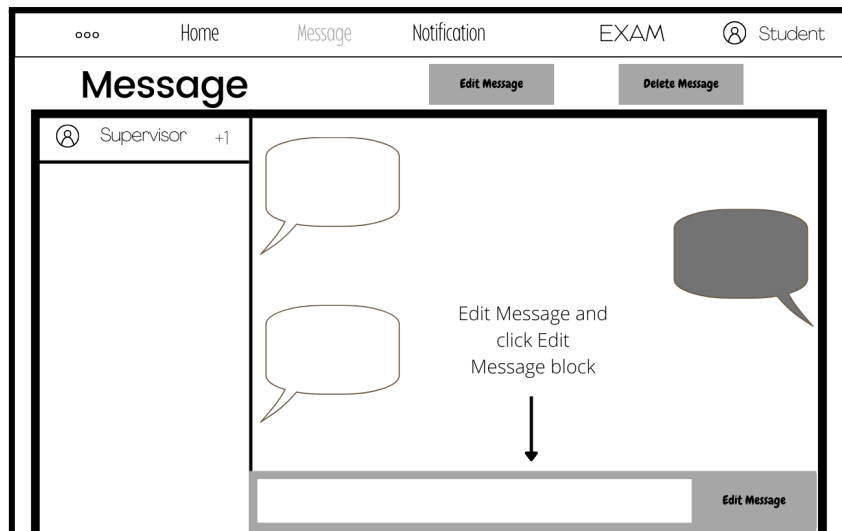
-Home Interface



-Message Interface







-Notification Interface

...	Home	Message	Notification	EXAM	Ⓜ Student
Notification					
✉ View Notification	To : Student From : Coordinator Title: Extension of exam period		DATE: 15/6/2022		
	To : Student From : Coordinator Title: Exam period		DATE: 11/6/2022		

...	Home	Message	Notification	EXAM	Ⓜ Coordinator
Notification					
✉ Add Notification	← Click block to add notification				
✉ Update Notification					
✉ Delete Notification					

...	Home	Message	Notification	EXAM	Ⓜ Coordinator
Notification					
✉	ID: 2234	DATE: 15/6/2022 TIME: 2:43 PM		Add Notification	
To : <<user>> From : Coordinator Title: "BIG TITLE"					
Content Enter text here					

ooo

Home

Message

Notification

EXAM

 Coordinator

Notification

 ID: 2234

DATE: 15/6/2022 TIME: 2:43 PM

Update Notification

To : <<user>>

From : Coordinator

Title: "BIG TITLE"

Content

Enter text here

The screenshot shows the 'Notification' page of the EXAM application. The top navigation bar includes 'Home', 'Message', 'Notification' (active), and 'EXAM'. A user profile icon for 'Coordinator' is on the right. The main heading is 'Notification'. On the left, there is a sidebar with three buttons: 'Add Notification', 'Update Notification', and 'Delete Notification', each with an envelope icon. The 'Delete Notification' button is highlighted. The main content area is empty, with a text prompt 'Click block to delete notification' and an arrow pointing to the 'Delete Notification' button in the sidebar.

ooo	Home	Message	Notification	EXAM	 Coordinator
Notification					
To : Student		From : Coordinator		DATE : 15/6/2022	Delete Notification
Title: Extension of exam period					
To : Student		From : Coordinator		DATE : 11/6/2022	Delete Notification
Title: Exam period					

-Exam Interface

	Home	Message	Notification	EXAM	Examiner
Exam					
Upload Question					
Test 2 (COA SECD8263)		Status: Students not completed	DATE : 13/6/2022		
Start : 8.00 pm					

☰

Home

Message

Notification

EXAM

 Examiner

Exam

 ID: 4123

Upload

Title: "exam title"

Assessment Question

Upload

Assessment Answer

Upload

Date : "enter date here"

Starts : "start time"Ends: "end time"

ooo

Home

Message

Notification

EXAM

 Examiner

Exam

Upload Question

Test 2 (COA SECD8263)

Status: Completed

DATE: 13/6/2022

Start : 8.00 pm

Scores

ooo	Home	Message	Notification	EXAM	 Examiner
Exam					
Student's Result					
Shahril Bin Saiful Bahri					Score
Naruto Anak Lelaki Minato					Score
Ichigo Kurosaki					Score
Man Kedah					Score

Home

Message

Notification

EXAM

Student

Exam

ID: 3242

Title: Test 2 (COA SECD8263)

Name : "student's name"

SCORES:

Marks : ??/??

Grade: ??

EXAM

Home

Message

Notification

EXAM

Student

Exam

Test 2 (COA SECD8263)

Status: Notyet answered

DATE: 13/6/2022

Start : 8.00 pm

Start

Home

Message

Notification

EXAM

Student

Exam

ID: 3242

Title: Test 2 (COA SECD8263)

Start : 8.00 pm

End : 12.00 pm

Start Exam

77

