

Group/Project

1 Domain Analysis (25 pts)

a Domain Model	8.00	8.00
i. Concept Definitions	2.00	2.00
ii. Association Definitions	2.00	2.00
iii. Attribute Definitions	2.00	2.00
iv. Traceability Matrix	2.00	2.00
b System Operation Contracts	3.00	3.00
c Data Model and Persistent Data Storage	3.00	3.00
d Mathematical Model / Algorithms	3.00	0.00

2 Interaction Diagrams (30 pts)

a Diagrams	20.00	20.00
b Description of Diagram	5.00	0.00
c Alternate Solution Description #	5.00	0.00

3 Class Diagram & Interface Specification (15 pts)

a Class Diagram	5.00	5.00
b Data Types and Operation Signatures	5.00	5.00
c Traceability Matrix	5.00	5.00

4 Algorithm and Data Structures(12 pts)

a Algorithms	4.00	4.00
b Data Structures	4.00	4.00
c Concurrency	4.00	4.00

5 User Interface Design and Implementation(4 pts)

a Description (+)	2.00	2.00
b Ease of Use (Usability)	2.00	2.00

6 Test Case Design(15 pts)

a List and Describe Test Case	5.00	4.00
b Discuss Test Coverage / Integration Test	5.00	3.00
c Plan for testing Algo, Non-Fnx Req. and UI Req.	5.00	3.00

7 Project Management and Plan of Work(19 pts)

a Merging the Contributions	10.00	10.00
b Project Coordination	5.00	5.00
c Plan of Work	2.00	2.00
d Breakdown of Responsibilities	2.00	2.00

PENALTY FOR LATE SUBMISSION**PENALTY FOR NO REFERENCE****(-5)**

Total Value/ Points 120.00 102.00

REPORT#2: VALUE/ GRADE 10.00 8.50

85%



Software Engineering

CMPS4131

Tamika Chen, Raynisha Cornelio, Javier Castellanos, Abner Mencia, Alex Peraza

Report 2



Apr 3, 2024

Table of Contents

Table of Contents	2
Contribution Breakdown	3
Responsibility Matrix	4
Responsibility Allocation Chart	5
Analysis and Domain Modeling	6
Concept Definitions	6
Association Definitions	9
Attribute Definitions	11
Traceability Matrix	13
Domain Models	14
System Operation and Contracts	17
Data Model and Persistent Data Storage	20
Interaction Diagrams	21
Algorithms & Data Structures	24
Class Diagram	24
Data Types and Operation Signature	25
Traceability Matrix	26
UI Design & Implementation	29
Design of Test	32
Overview of Design Tests	32
Project Management & Plan of Work	38
Issues Encountered	38
Breakdown of Responsibilities	38
References	39



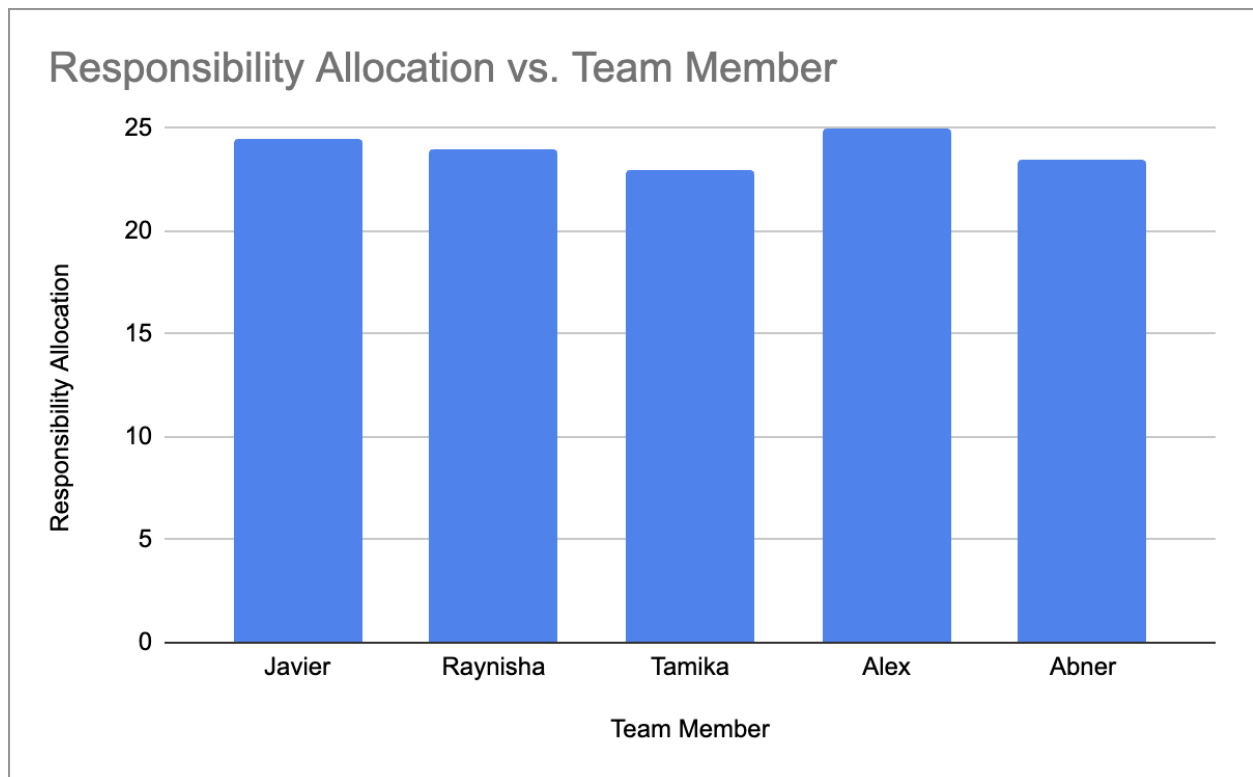
Contribution Breakdown

Signature Block			
Statement	I did my share of the work, and I have a general understanding of the contents of the assignment.		
Team Member	Contribution	Signature	Date
Tamika Chen	Assisted with Analysis and Domain Modeling, Interaction Diagrams, Design of Tests and Plan of Work		03/04/24
Raynisha Cornelio	Assisted with Analysis and Domain Modeling and Plan of Work		03/04/24
Javier Castellanos	Assisted with Analysis and Domain Modeling and Interaction Diagrams	Javier Castellanos	03/04/24
Abner Mencia	Assisted with Class Diagrams and Interface Specifications, Algorithms and Data Structures and Design of Tests	Abner Mencia	03/04/24
Alex Peraza	Assisted with Interaction Diagrams, Algorithms and Data Structures and UI Design and Implimentation	alexPeraza	03/04/24

Responsibility Matrix

Responsibility Level	Team Member Names				
	Tamika Chen	Raynisha Cornelio	Javier Castellanos	Alex Peraza	Abner Mencia
Sec. 1: Analysis and Domain Modeling (25 pts)	4%	58%	38%		
Sec. 2: Interaction Diagrams (30 pts)	20%		50%	30%	
Sec. 3: Class Diagrams and Interface Specification ((15 pts)					100%
Sec. 4: Algorithms and Data Structures (12 pts)				75%	25%
Sec. 5: User Interface Design and Implementation (4 pts)				100%	
Sec. 6: Design of Tests (15 pts)	43%				57%
Sec. 7: Project Management and Plan of Work (19 pts)	50%	50%			

Responsibility Allocation Chart



Analysis and Domain Modeling

Concept Definitions

UC-1: Register(AddUser)

Responsibility Description	Concept Name
Coordinates the actions associated with the user registration	Controller
Render the registration page for users	Page Maker
HTML form specifying the data required for user registration	Interface Page
Verifies and processes the user's registration data	Upload Request
Ensures all the necessary fields are filled out correctly	Upload checker
Manages the database interaction for storing user registration data	Database Connection

Table 1: Concepts for Register

UC-2: Login

Responsibility Description	Concept Name
Coordinates the actions of all concepts associated with the use case and delegates the work to other concepts.	Controller
Validates user credentials during the login process	Authentication
Renders the page that is specified after a successful login	Page Maker
Manages user sessions and authentication tokens	Sessions management
Establishes a connection with the database to verify user credentials	Database Connection

HTML form specifying the data required for user logging in	Interface Page
--	----------------

Table 2: Concepts for Login

UC-3 View Profile:

Responsibility Description	Concept Name
Coordinates the actions of all concepts associated with viewing user profiles	Controller
Renders the profile page for user	Page Maker
HTML document displaying user profile information	Interface Page
Retrieves and displays user-specific profile information	Profile Data
Manages the database interaction for retrieving profile data	Database Connection

Table 3: Concepts for Viewing Profile

UC-8 File Report:

Responsibility Description	Concept Name
Coordinates the actions and handles user interactions	Controller
Renders the report form interface	Report Form Renderer
Provides fields for users to input report details	Report Form
Validates the submitted report for completeness and accuracy	Report Validator
HTML document displaying user profile information	Interface Page

Processes and submits the reports to the appropriate authorities	Report Submission
Renders the profile page for user	Page Maker

Table 4: Concepts for filing a report

UC14- Log Report:

Responsibility Description	Concept Name
Coordinates the logging of reports and delegates tasks as necessary	Controller
Renders the form for security personnel to input report details	Report Form
Prepares and executes database queries to store the logged reports	Database connection

Table 5: Concepts for logging a report

UC17- Post to Notice Board:

Responsibility Description	Concept Name
Coordinates the posting of notices on the notice board and delegates tasks as necessary	Controller
Renders the form for authorized users to input notice details	Notice Board Form
Prepares and executes database queries to store the posted notices	Database connection

Table 6: Concepts for posting on the notice board

Association Definitions

UC-1: Register(AddUser)

Concept Pair	Association Description	Association Name
Controller ↔ Page Maker	The controller passes the request to Page Maker to render the registration page.	render registration page
Controller ↔ Upload Request	Controller forwards user registration data to Upload Request for verification and processing.	process registration data
Upload Checker ↔ Interface Page	Upload Checker validates the user input specified in the Interface Page.	validate user input
Upload Request ↔ Database Connection	Upload Request interacts with Database Connection to store the user's registration information.	store registration data

Table 7: Association for Register

UC-2: Login

Concept Pair	Association Description	Association Name
User ↔ Authentication System	User provides login credentials to the Authentication System.	provide credentials
Authentication System ↔ User	Authentication System verifies user credentials and grants access.	verify credentials

Table 8: Association for Login

UC-3: View Profile

Concept Pair	Association Description	Association Name
User ↔ Profile Database	User requests profile information from the Profile Database.	Request information

Profile Database ↔ User	Profile Database retrieves and provides profile information to the user.	Provide information
-------------------------	--	---------------------

Table 9: Association for View Profile

UC-8: File a Report

Concept Pair	Association Description	Association Name
User ↔ Report Form	User submits a report using the Report Form	Submit Report
Report Form ↔ Security	Report Form forwards the report to the Security team	Forward report

Table 10: Association for File a Report

UC-14: Log Report

Concept Pair	Association Description	Association Name
User ↔ Report Logging System	User submits a report using the Report Logging System.	submit report
Report Logging System ↔ User	Report Logging System logs the report submitted by the user.	log report

Table 11: Association for Log Report

UC-17: Post to Notice Board

Concept Pair	Association Description	Association Name
Admin ↔ Notice Board	Admin posts a notice to the Notice Board.	post notice
Notice Board ↔ User	Notice Board displays the posted notice to the user.	display notice

Attribute Definitions

UC-1: Register(AddUser)

Concept	Attributes	Attribute Description
Upload Request	User Registration Data	User's registration data including first name, last name, email, address, phone number, and password.
Upload Checker	User Input	User input data to be validated.
Database Connection	User Registration Data	User registration data to be stored in the database.

Table 13: Attribute for Register

UC-2: Login

Concept	Attributes	Attribute Description
User	Login credentials	User's login credentials (username/password).

Table 14: Attribute for Login

UC-3: View Profile

Concept	Attributes	Attribute Description
Profile Database	Profile Information	User's profile information (e.g., name, contact details).

Table 15: Attribute for View Profile

UC-8: File a Report

Concept	Attributes	Attribute Description
Report Form	Report Data	Data submitted in the report.

UC-14: Log Report

Concept	Attributes	Attribute Description
Report Logging System	Report Data	Data submitted in the report.

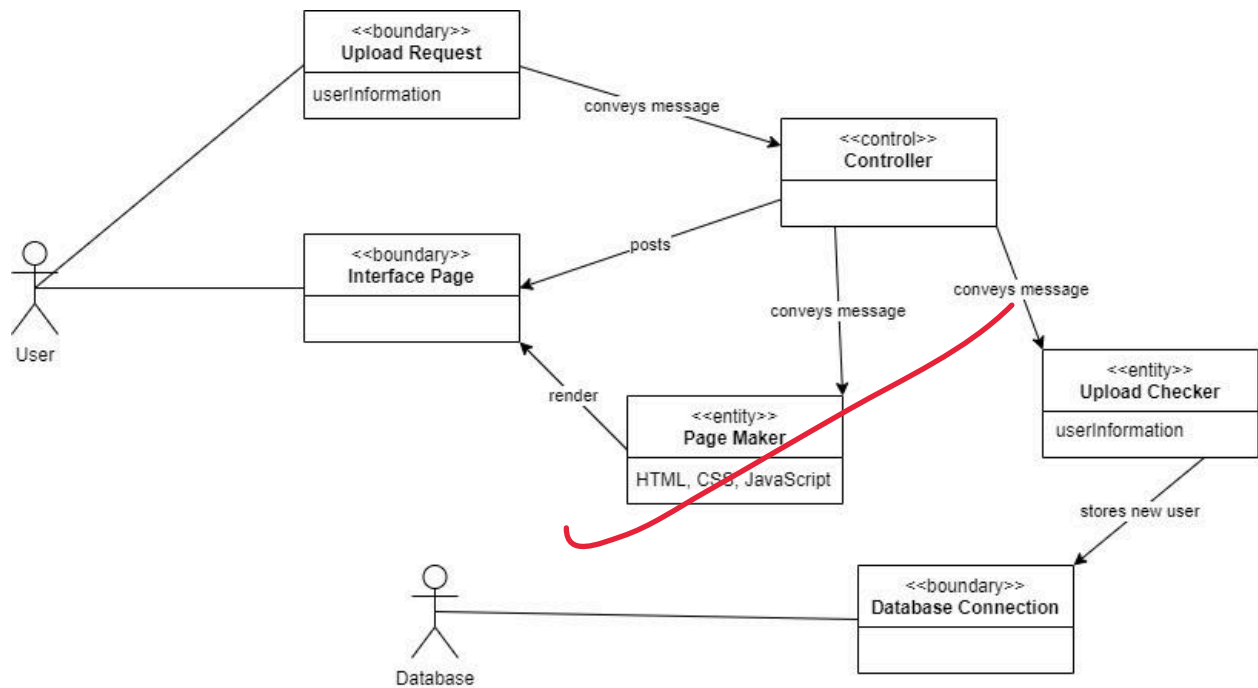
Table 16: Attribute for Log Report

Traceability Matrix

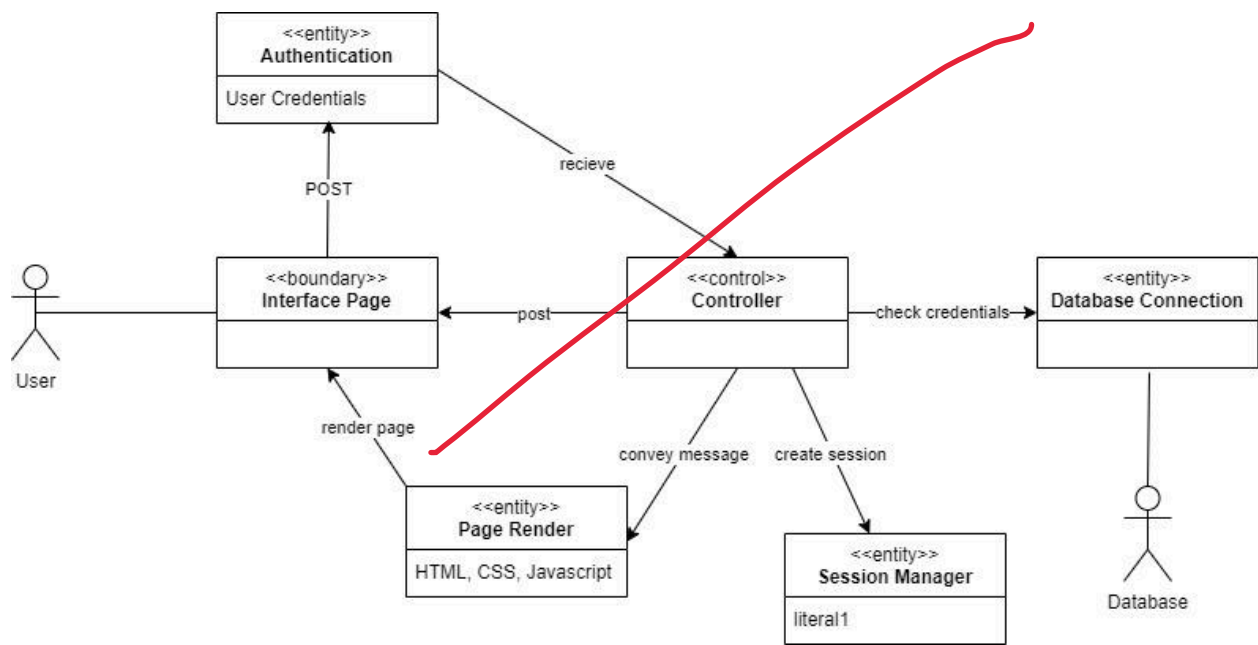
		Domain Concepts													
Use Cases	PW	Controller	<u>Page Maker</u>	<u>Interface Page</u>	<u>Upload Request</u>	Upload Checker	Database Connection	Authentication	Sessions Management	Profile Data	Report Form Renderer	Report Form	Report Validator	Report Submission	Notice Board Form
UC-1		X	X	X	X	X	X								
UC-2		X	X	X			X	X	X						
UC-3		X	X	X			X			X					
UC-4		X	X	X			X	X		X					
UC-5		X	X	X			X								X
UC-6		X	X	X			X								X
UC-7		X	X	X	X	X	X								
UC-8		X	X	X							X	X	X	X	
UC-9		X	X	X			X								
UC-10		X	X	X			X								
UC-11		X	X	X			X								
UC-12		X	X	X			X								
UC-13		X	X	X			X								
UC-14		X	X	X			X				X	X	X	X	
UC-15		X	X	X			X								
UC-16		X	X	X			X								
UC-17		X	X	X			X								X
UC-18		X	X	X			X								
UC-19		X	X	X			X				X	X	X	X	

Domain Models

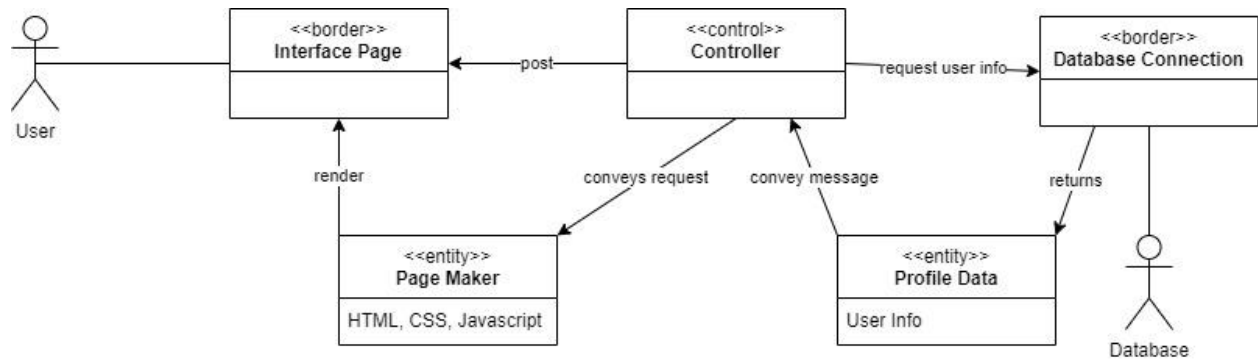
UC-1 Register



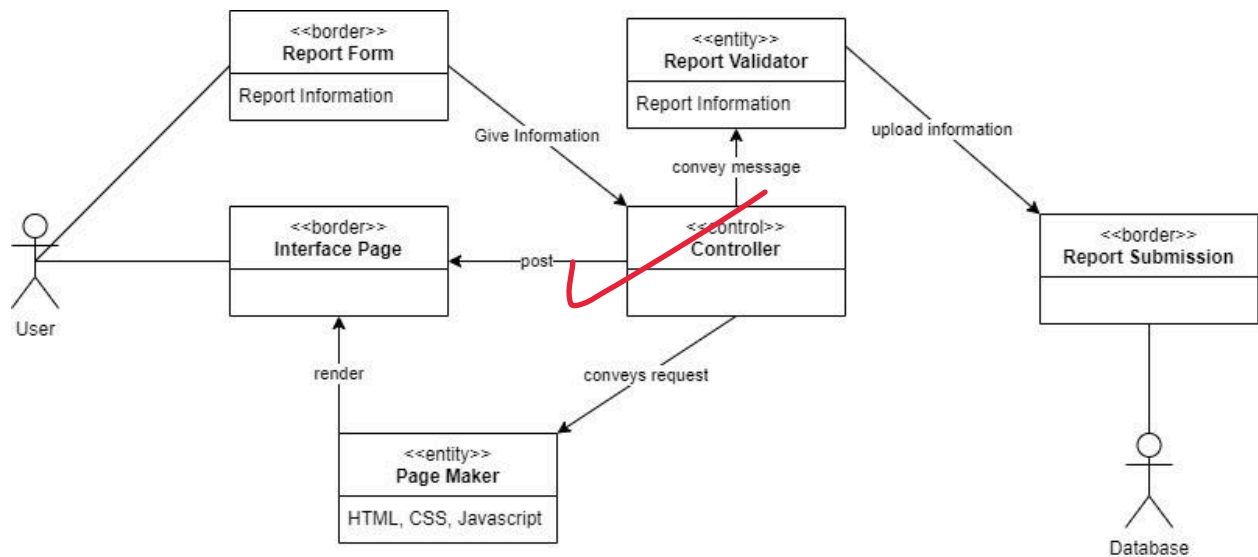
UC-2-Login



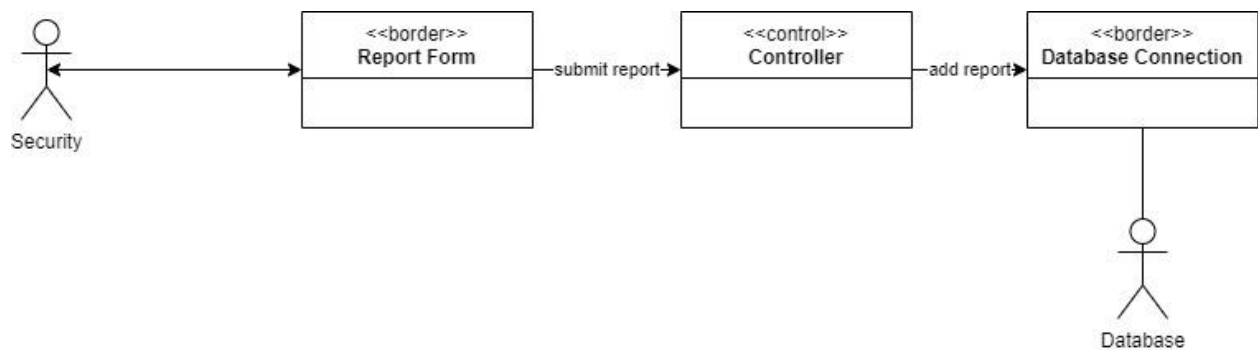
UC-3-View Profile



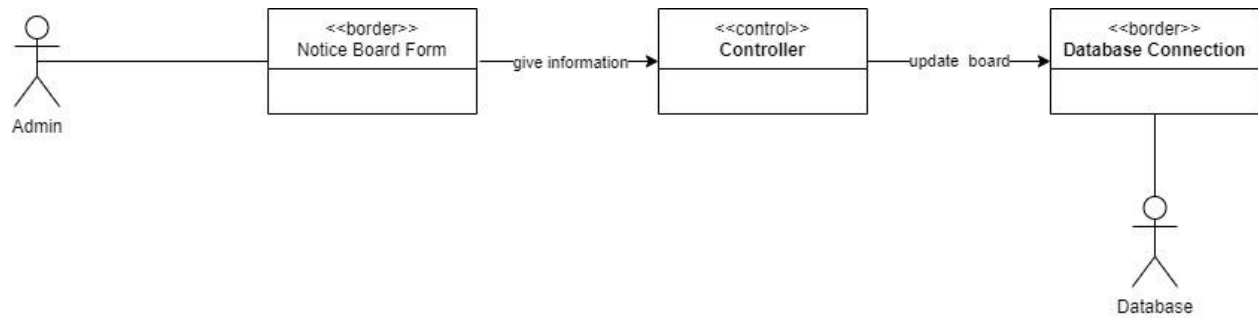
UC-8 File a Report



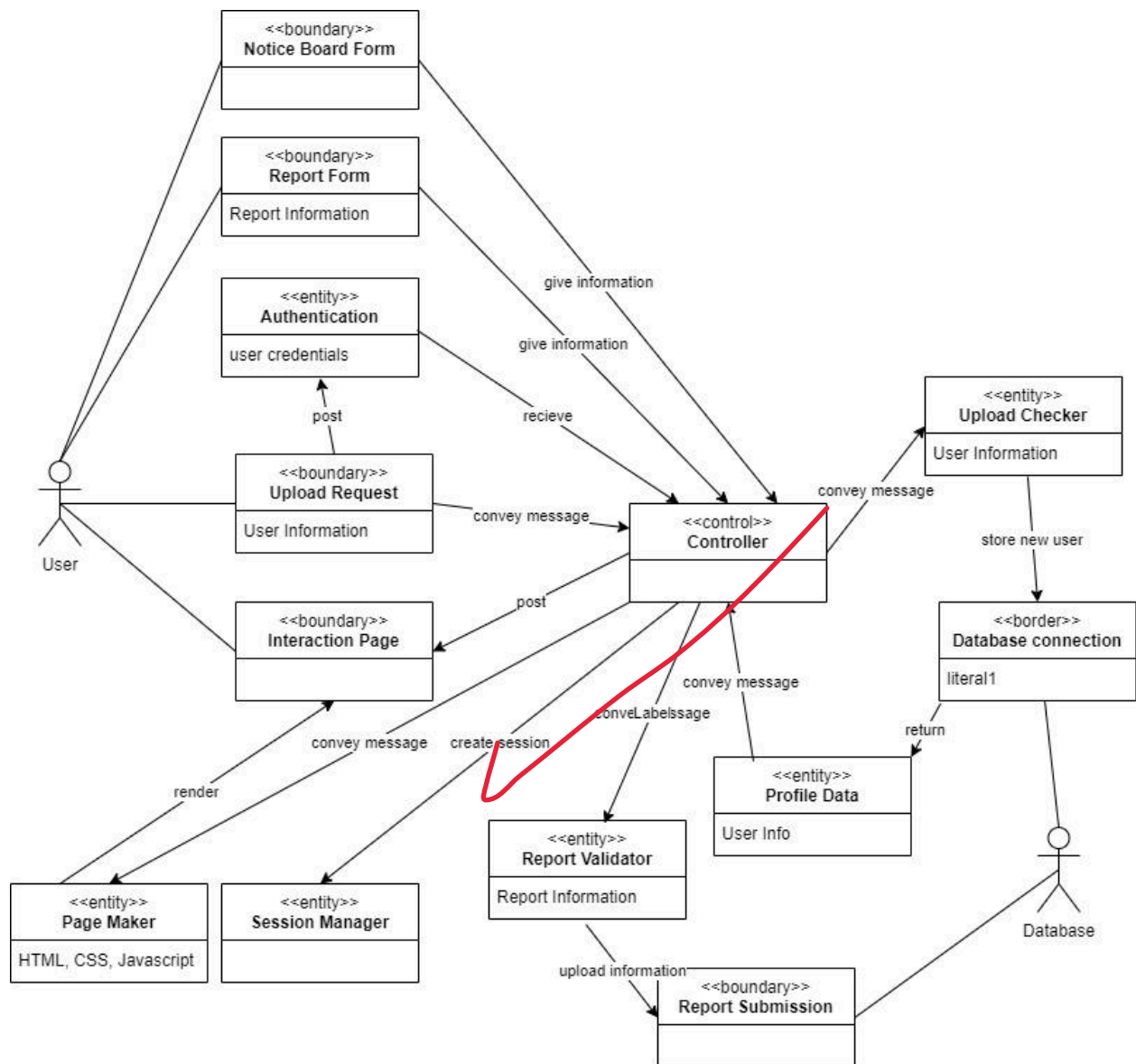
UC-14 Log Report



UC-17 Post to Notice Board



Domain Model of Entire System



System Operation and Contracts

Table 17: System Operation Contract for Register

Contract Name	createUser(username, fname, mname, lname, dob, gender, imagedata)
Cross Reference	Use Case: Register
Responsibility	Add new users information to the system
Type	System
Pre-condition	<ul style="list-style-type: none"> Email must not be taken Internet connection must be available Email is a valid UB email System is up and running
Post-condition	<ul style="list-style-type: none"> A new user has been added to the database Email is no longer available for register Email is now ready to log in

Table 18: System Operation Contract for Login

Contract Name	login(username, password)
Cross Reference	Use Case: Login
Responsibility	Give registered users access to the database
Type	System
Pre-condition	<ul style="list-style-type: none"> Email is already registered in the database Internet connection is available System is up and running Credentials are valid and correct
Post-condition	<ul style="list-style-type: none"> User has access to system features A session has been created by the session manager

Table 19: System Operation Contract for View Profile

Contract Name	profile()
Cross Reference	Use Case: View Profile
Responsibility	Show the user information about themselves for the app

Type	System
Pre-condition	<ul style="list-style-type: none"> • User is logged in • Internet connection is available • System is up and running
Post-condition	<ul style="list-style-type: none"> • User can see or edit information about themselves for the app

Table 20: System Operation Contract for File a Report

Contract Name	createReport(type_of_incident, location, description, is_anonymous, device_location)
Cross Reference	Use Case: File a Report
Responsibility	Create a report from the provided fields
Type	System
Pre-condition	<ul style="list-style-type: none"> • User is logged in • Internet connection is available • System is up and running • All fields have valid information
Post-condition	<ul style="list-style-type: none"> • Adds a new report to the database

Table 21: System Operation Contract for Log Report

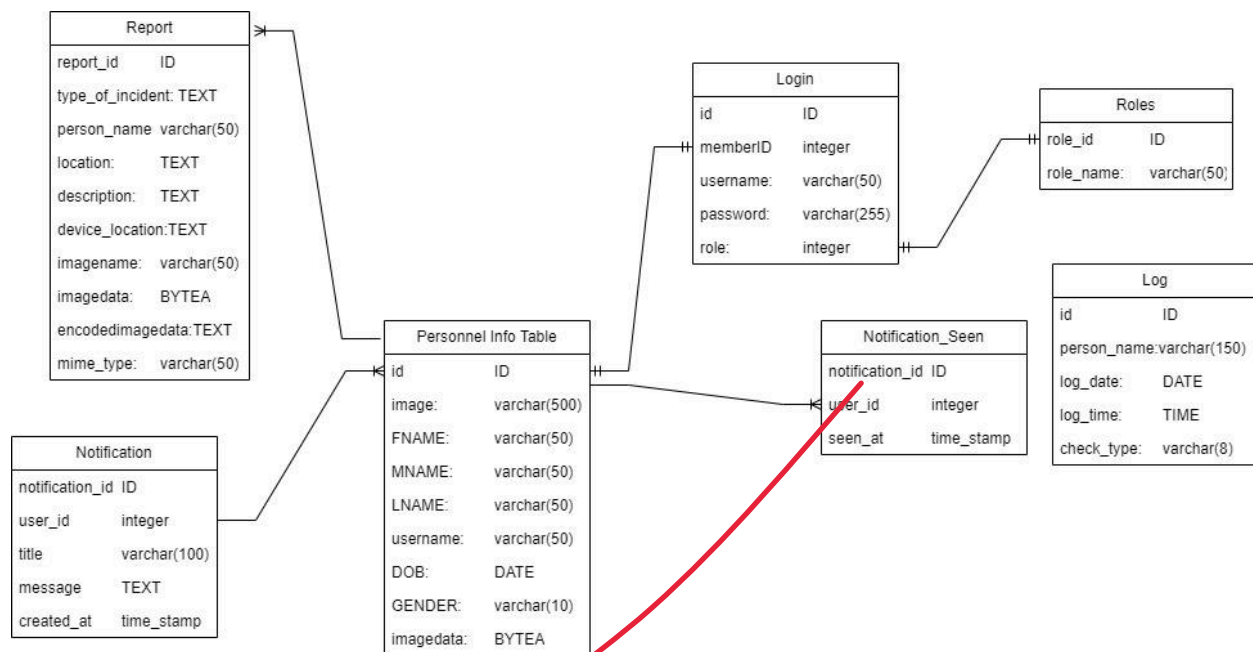
Contract Name	createLog(date, time, checkType)
Cross Reference	Use Case: Log Report
Responsibility	Allow security personnel to create their daily reports from the provided fields
Type	System
Pre-condition	<ul style="list-style-type: none"> • User is logged in • Internet connection is available • System is up and running • User is at the end of their shift • User has clocked in for work
Post-condition	<ul style="list-style-type: none"> • Report from the security has been added to the database

Table 22: System Operation Contract for Post to Notice Board

Contract Name	createNotice(title, message)
----------------------	------------------------------

Cross Reference	Use Case: Post to Notice Board
Responsibility	Allows admins to post any notices or news to the notice board so users can view
Type	System
Pre-condition	<ul style="list-style-type: none">• User is logged in• Internet connection is available• System is up and running• User is an admin
Post-condition	<ul style="list-style-type: none">• A new notice has been added to the database• The notice is displayed on users notice board

Data Model and Persistent Data Storage



This system was designed to hold as much data as it could. The purpose of data storing is for archiving purposes. For security reasons, it is important to archive all reports or as much data as possible to aid security personnel in investigations if needed. Also another reason storing data is to secure user information since the system will have a register and login in features. Data storing is again proven useful to manage the data and make it easier to retrieve and display information where needed.

The “Roles” table will have three(3) values in it. These values include admin, student, and guard. The type of access to features will depend on the type of role the user is given.

The “Notification” table is used to manage the notifications that will be sent out by the system. The “user_id” in the table will be used to keep track of who all have received the notification or who should receive the notification.

The “Notification_seen” table is used to manage the notifications that have already been seen by the users. The “user_id” in the table is to keep track of who all have already seen this notification.

Interaction Diagrams

Figure 1: Sequence Diagram for Register(UC-1)

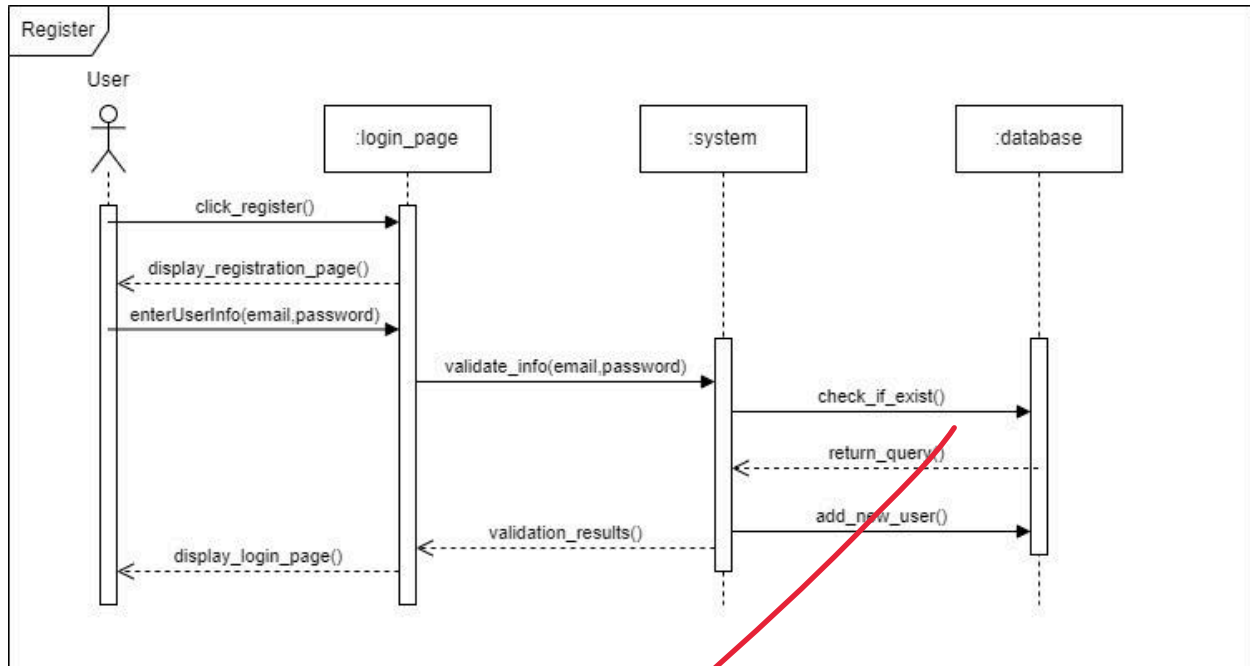


Figure 2: Sequence Diagram for Login(UC-2)

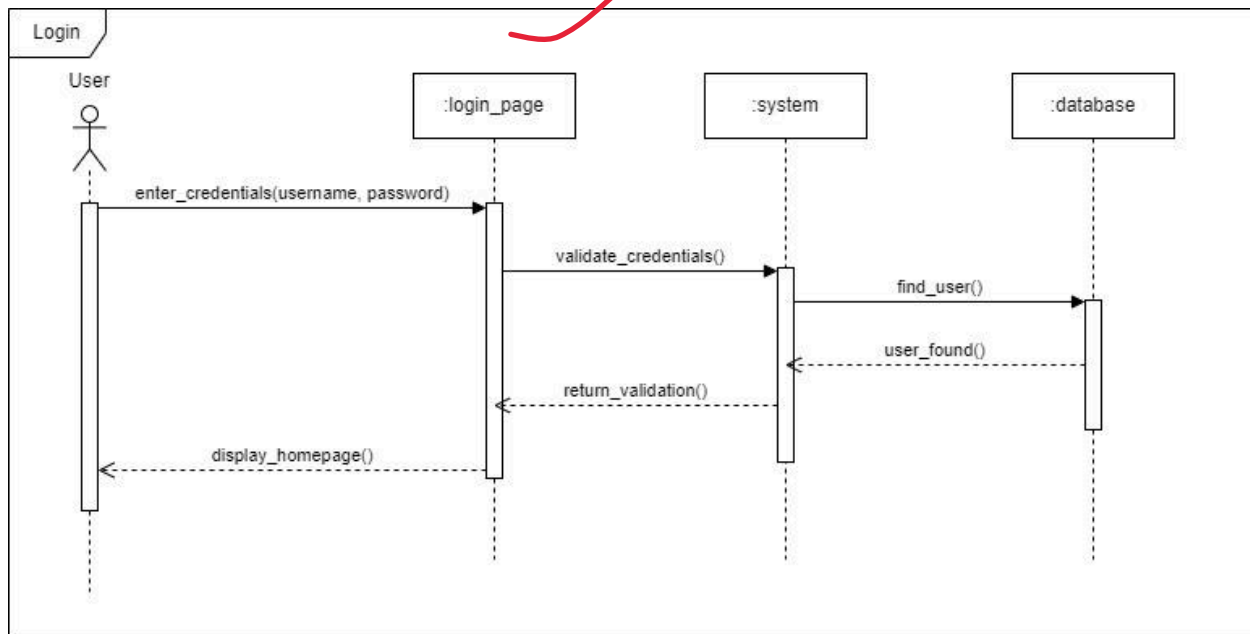


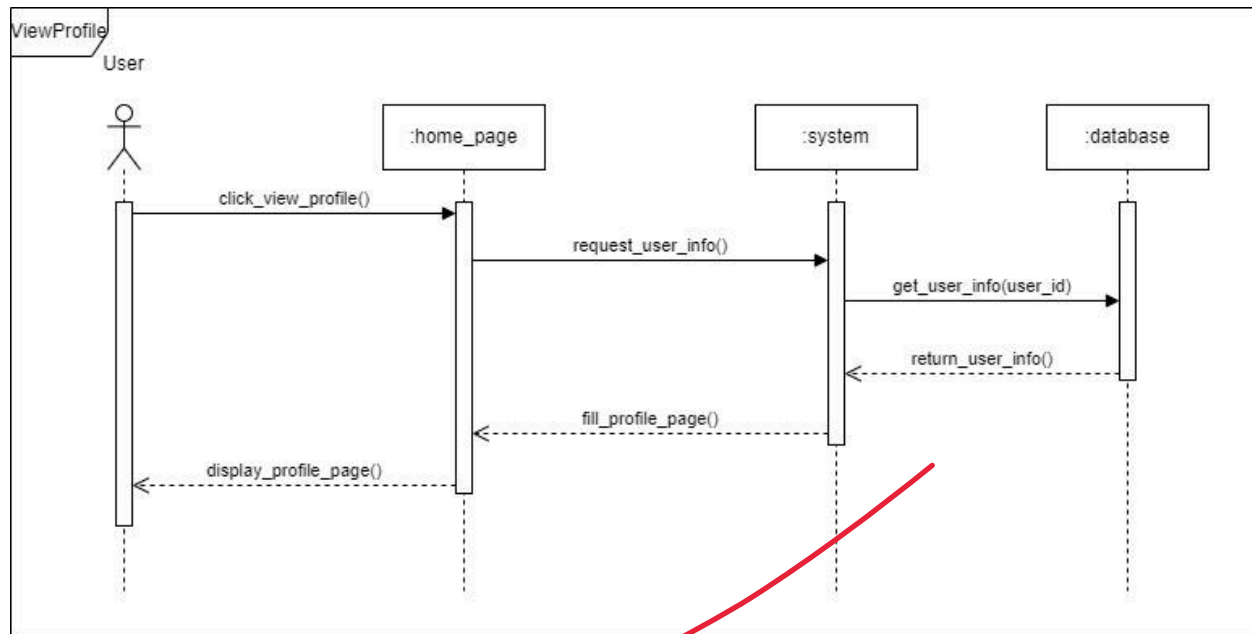
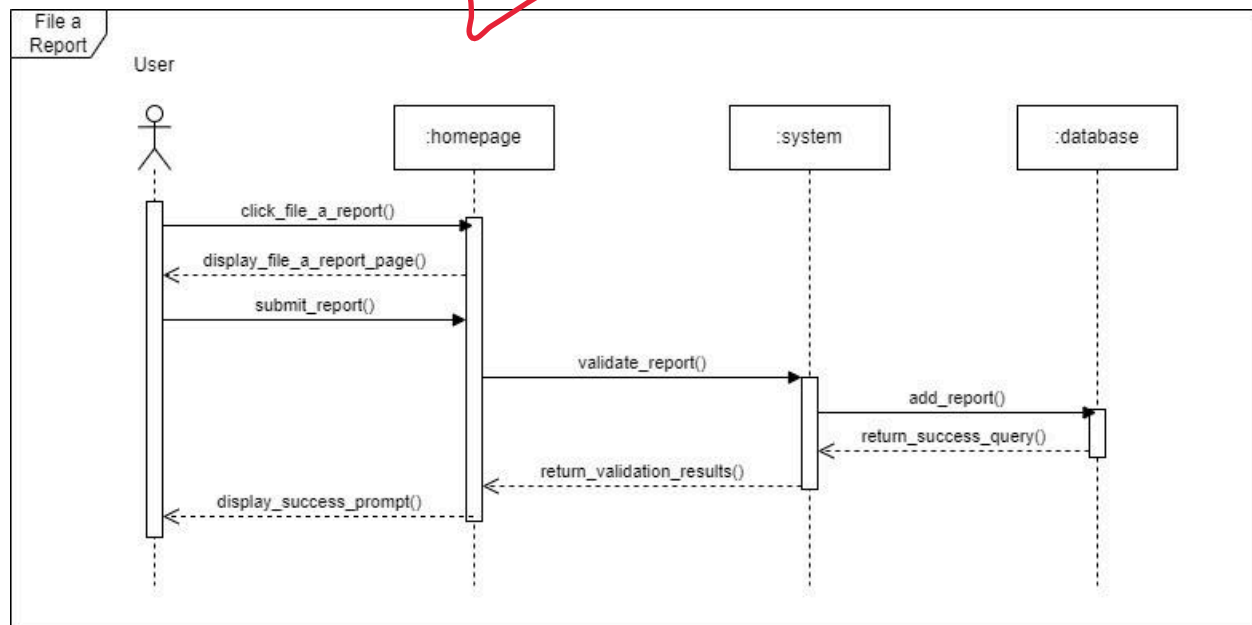
Figure 3: Sequence Diagram for View Profile(UC-3)*Figure 4: Sequence Diagram for File a Report(UC-8)*

Figure 5: Sequence Diagram for Log Report(UC-14)

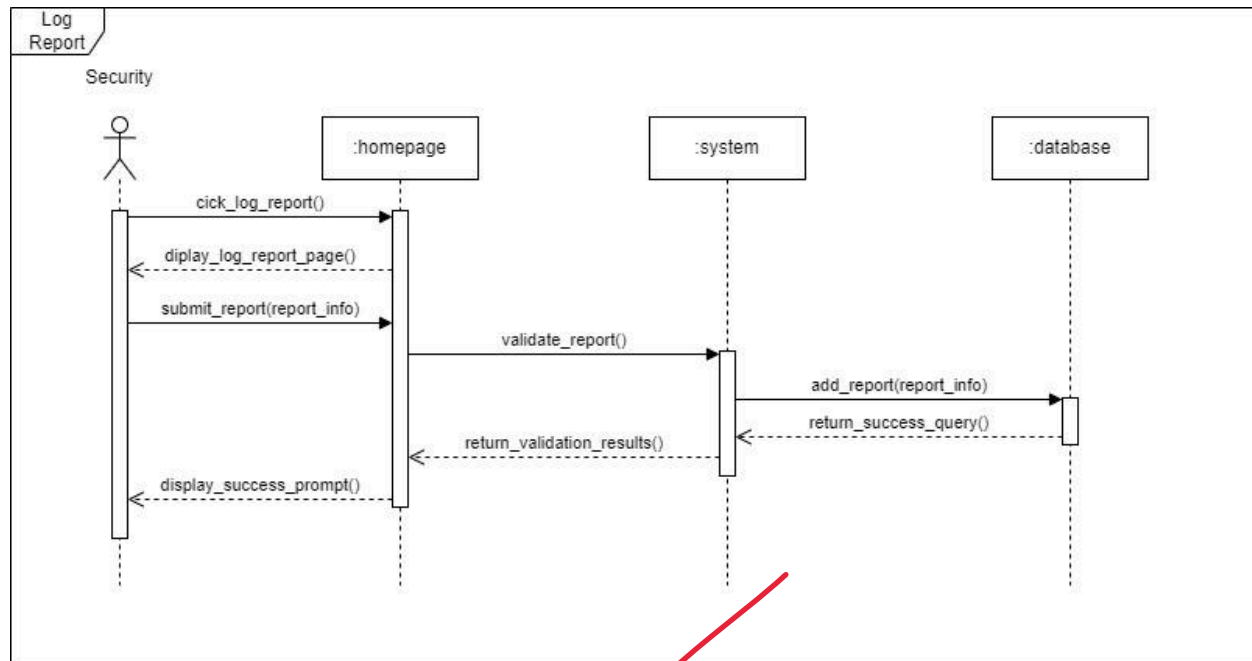
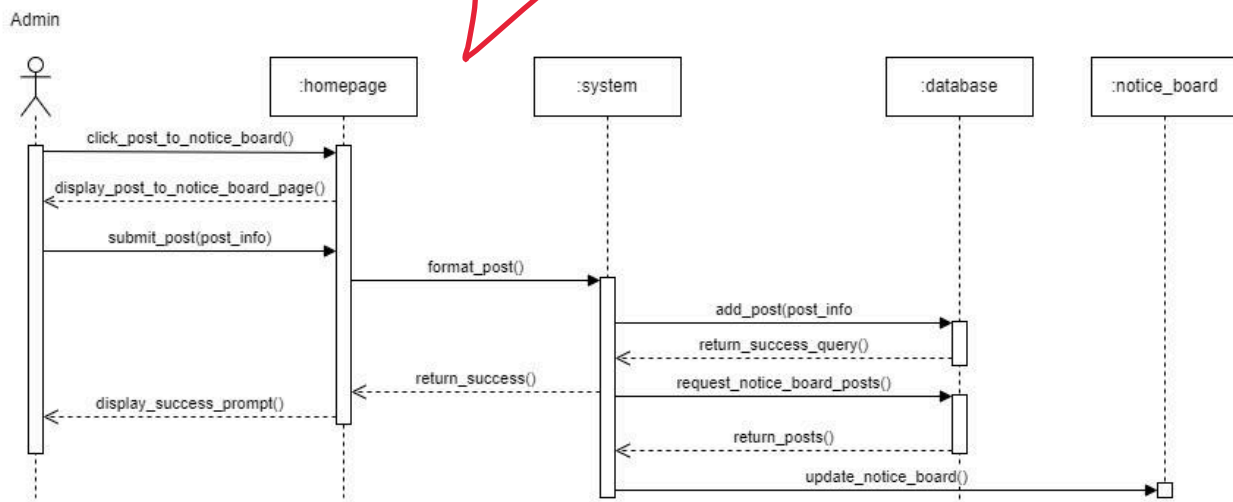


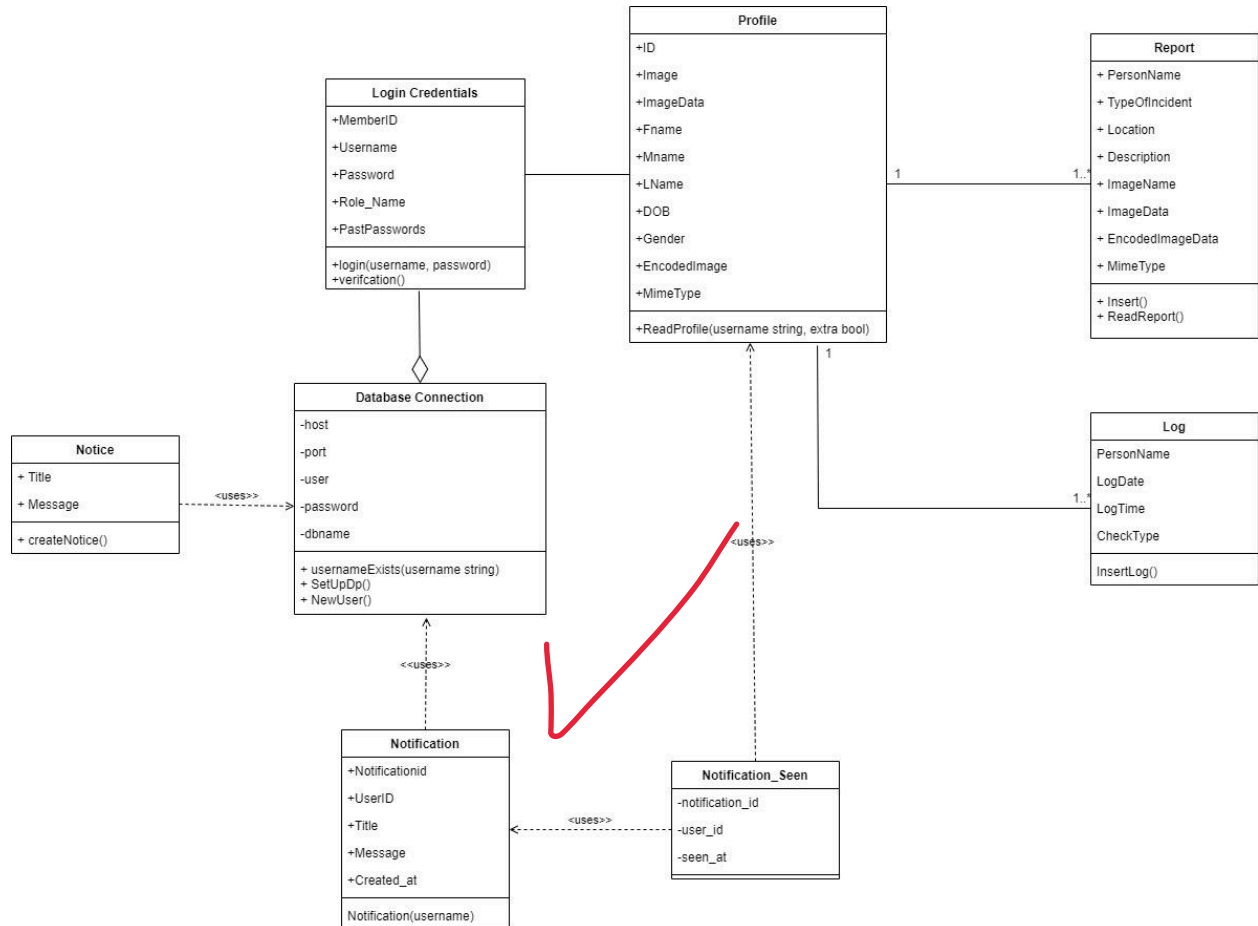
Figure 6: Sequence Diagram for Post to Notice Board(UC-17)



description and alternate

Algorithms & Data Structures

Class Diagram



Data Types and Operation Signature

Login Credentials	
MemberID	int
Username	string
Password	string
Role_Name	int
PastPasswords	string
login(username, password) void verification() void	

Report	
PersonName	string
TypeOfIncident	string
Location	string
Description	string
ImageName	string
ImageData	[]byte
EncodedImageData	string
MimeType	string
Insert()	int, error
ReadReport()	void

Profile	
ID	int
Image	string
ImageData	[]byte
Fname	string
Mname	string
LName	string
DOB	time.Time
Gender	string
EncodedImage	string
MimeType	string
+ReadProfile(username string, extra bool) []struct, error	

1

Log	
PersonName	string
LogDate	time.Time
LogTime	time.Time
CheckType	string
InsertLog()	int, error

Notification	
Notificationid	int
UserID	int
Title	string
Message	string
Created_at	time.Time
Notification(username) string, error	

Notification_Seen	
notification_id	int
user_id	int
seen_at	time.Time

Notice	
Title	string
Message	string
InsertNotice()	int, error

1..*

Database Connection	
host	string
port	int
user	string
password	string
dbname	string
+ usernameExists(username string) bool, error + SetUpDp() void + NewUser() int, error	

Traceability Matrix

	Login Credentials	Report	Profile	Log	Notification	Notification_Seen	Notice	Database Controller
Controller								
Page Maker								
Interface Page								
Upload Request								
Upload Checker								
Database Connection								
Authentication								
Sessions Management								
Profile Data								
Report Form Renderer								
Report Form								
Report Validator								
Report Submission								
Notice Board Form								

Login Credentials:

- Deals with login data from the login page.
- Successful login creates session
- Requires profile Data and database connection.
- Required page to be rendered

Reports:

- Evolved from Report Validator as it deals with validating and formatting reports
- Requires page to be rendered and a successful database connection
- Concerned with primary actor student

Profile:

- Evolved from Profile Data as it deals with displaying data for user
- Concerned with all actors
- Requires database connection

Log:

- Concerned with primary actor Security
- Deals with daily report submitted by actor
- Unlike the report from the student actors, does not use upload checker

Notifications:

- Deals with managing and formatting notifications created by system admins

Notification_Seen:

- Used to manage users who have seen the notification

Notice:

- A very simple class used to manage the title and message from notices created by the system admin to output to the notice board

Database Controller:

- Main database controller, used to serve as a controller for all queries. Evolved from Database Connection

The UB Campus Security application will utilize:

1. **Hash Tables:** In the UB campus safety application, the concept of hash tables can be applied through database indexing. For instance, indexing on critical columns like username in the LOGIN table facilitates swift retrieval of user login details. This indexing mechanism enhances the efficiency of data retrieval processes within the application. Similar indexing techniques can be employed across other tables, such as notification and Report, optimizing data retrieval performance. The utilization of indexing serves a comparable purpose by enabling rapid data lookup based on specific keys, thereby enhancing the overall efficiency and responsiveness of the application.
2. **Arrays:** In the JavaScript files for the UB campus safety application, arrays play a crucial role in managing various elements and event listeners within the user interface. Specifically, arrays are utilized to store references to DOM elements representing list items (li_items) and facilitate the handling of mouse hover and mouse leave events for

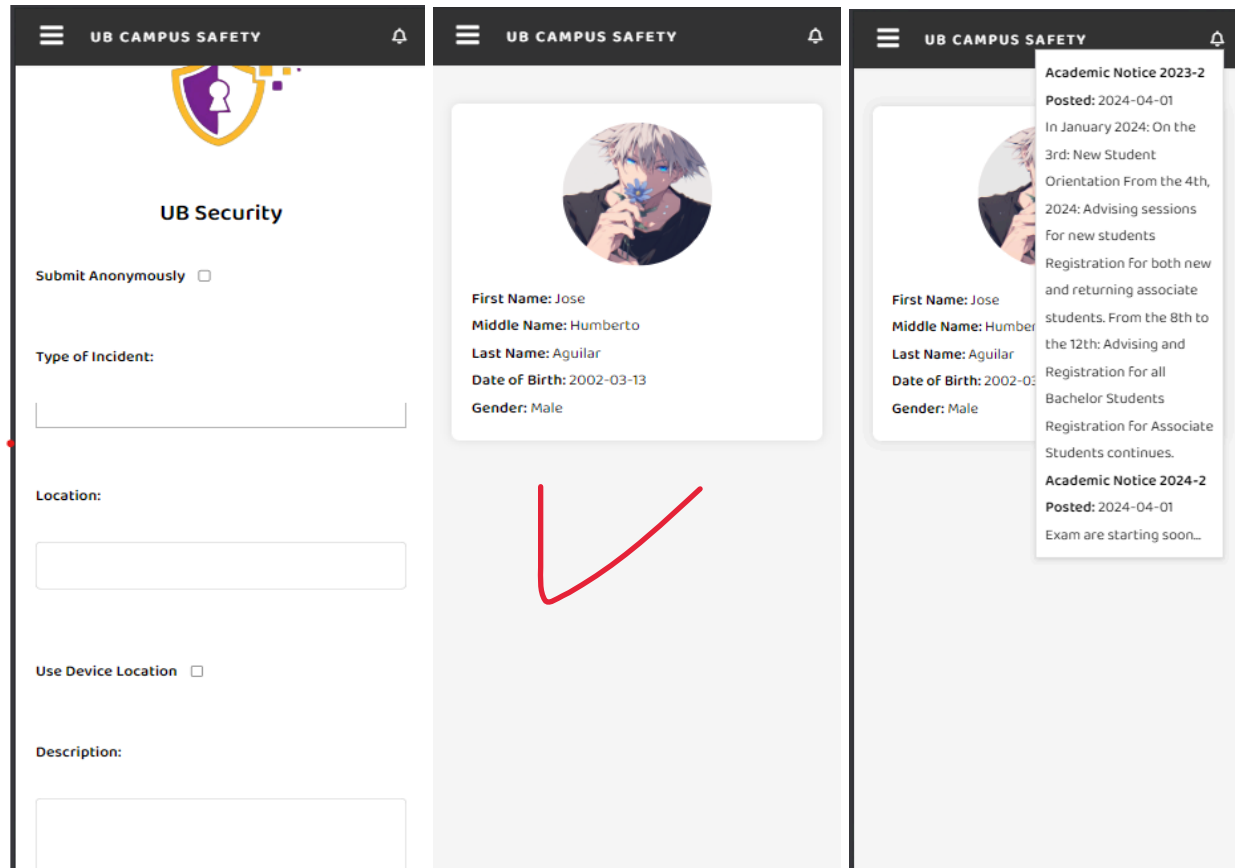
each list item. Additionally, arrays are employed to store references to essential HTML elements such as the hamburger menu and the wrapper element. By utilizing arrays, the application efficiently manages and manipulates multiple elements of the same type, enhancing the overall user experience and interactivity of the interface.

3. Criteria used to choose the Data Structure:

- Efficiency and speed of data retrieval: Arrays are chosen for fast and easy traversal.
- Security: Hash tables are used to store sensitive information securely.
- Scalability: Arrays and hash tables can be easily scaled up to handle a large amount of data and users.
- Flexibility: JS is used for data exchange due to its flexibility and lightweight nature.

UI Design & Implementation

Student



These diagrams demonstrate the Home look of the Student's interface then when the student clicks on the hamburger menu icon it will show the option to select either profile or to add a report and also if the user clicks on the notification bell, it will display the notice that the administrator has added.

Security Guard

The interface consists of four panels, each with a hamburger menu icon and the text 'UB CAMPUS SAFETY'.

Top Left Panel: Features a shield icon with a person silhouette. Below it is the text 'UB Security'. There is a checkbox labeled 'Submit Anonymously' and a text input field for 'Type of Incident:'. Below that is a text input field for 'Location:'. There is a checkbox labeled 'Use Device Location' and a text input field for 'Description:'.

Top Middle Panel: Titled 'Reports', it displays a list of incident reports. The first report shows 'Person Name: Anonymous', 'Type of Incident: Vandalism', 'Location: 17.5045661, -88.1962133', and 'Description: not sure'. It includes a small image of a building. The second report shows 'Person Name: Jose Aguilar', 'Type of Incident: Vandalism', 'Location: 17.5045661, -88.1962133', and a detailed description of graffiti. It includes a small image of a building. The third report shows 'Person Name: Anonymous' and 'Type of Incident: Assault'.

Top Right Panel: Titled 'Check in / Check out Form', it has a 'Select Date:' dropdown menu showing '04/02/2024'. Below it is a 'Select Time:' dropdown menu showing '03:15 PM'. There is a 'Select Check Type:' dropdown menu showing 'Check In'. A blue 'Submit' button is at the bottom.

Bottom Left Panel: Displays a user profile card with a circular avatar of a police officer. Below the avatar, the text reads: 'First Name: John', 'Middle Name: Aguilar', 'Last Name: Sanchez', 'Date of Birth: 1987-06-16', and 'Gender: Male'.

The following diagrams show the Security guard panel. When the user logs in, they will be greeted with the check-in / check-out form so that they can submit. When the user has done that, they can access the other panels by clicking on the hamburger menu and selecting which panel they want to visit.

Administrator

Add Notice

Title:

Message:

Submit

Add User

TYPE: Admin

username:

First Name:

Middle Name:

Last Name:

User Profile

First Name: Alex
Middle Name: Humberto
Last Name: Peraza
Date of Birth: 2003-03-12
Gender: Male

Check in / check out Log's

Name: John Sanchez
Date: 2024-03-31
Time: 6:08 PM
Status: checkin

Name: John Sanchez
Date: 2024-03-31
Time: 6:11 PM
Status: checkout

Reports

Person Name: Anonymous
Type of Incident: Vandalism
Location: 17.5045661, -88.1962133
Description: not sure
Image:

Person Name: Jose Aguilar
Type of Incident: Vandalism
Location: 17.5045661, -88.1962133
Description: The graffiti, a kaleidoscope of colors and shapes, seemed to scream for attention, demanding recognition in a world often blinded by routine. From the bold, sweeping curves of vibrant red to the intricate, labyrinthine patterns of deep indigo, each mark carried with it a message—a cry for expression, a rebellion against the mundane.
Image: No image available

Person Name: Anonymous
Type of Incident: Assault

The following diagrams show the admin panel. When the user logs in, they will be greeted with the notice form, if the administrator has something important to say or remind the user about any events the administrator can submit a notice form.. If the user chooses not to submit a notice, they can access the other panels by clicking on the hamburger menu and selecting which panel they want to visit from what they wish to do, like registering a new user, viewing a report or log and checking their profile.

Design of Test

Overview of Design Tests

The functionalities used in the test cases were essential elements that were directly related to the overall functioning of the system and our priority needs. It's crucial to remember that it is not possible to automate regression testing for all aspects of a project. It is very difficult, if not impossible, to build an automated script due to its intricacy and the vast number of alternative circumstances. Also, creating such a script would probably take more time and effort than it would save. For now, humans are still better suited for this activity because they are mentally capable to evaluate, react to, and calculate elements that robots are not yet able to handle. We are planning to evaluate our six primary use cases through testing.

UC-#1: Register(AddUser)

Test Case	TC-1
Use Case being tested	Register
Criteria for Success/Fail	Tests that a user can be successfully added to the Database and create default Credentials
Input Data:	Alphanumeric
Test Procedure:	
Step 1: Create a mock Database connection	Use Mock library to create a fake database connection which simulates the table we are going to query.
Step 2: Use data models with mock Database	Create an instance of the our data models with mock database connection and define sample input data
Step 3: create mock of helper function (usernameExist)	Create a mock which will return false(username does not exist) so that there are no repetition on usernames
Step 4: Define Expected results	Create definitions for Expected queries to our mock database and definitions of their return values to both tables, user and login credentials tables.
Step 5: Call Function	We call the NewUser Function With Sample data which was previously defined and print errors in any Otherwise move to

	next step
Step 6: Check Return Value	Check if Function Returned previously defined Expected ID (1), if not, fail test
Step 7: Check for other database Expectation	Verify that Database did not had more Queries/Inserts than previously defined

UC-2: Login

Test Case	TC-2
Use Case being tested	Login
Criteria for Success/Fail	Test that a registered user can login with their correct credentials
Input Data:	Alphanumeric
Test Procedure:	
Step 1: Create a mock Database connection	Use Mock library to create a fake database connection which simulates the table we are going to query.
Step 2: Create mock http & call Function	Create Mock request to simulate method POST And a response Recorder and pass to verification function
Step 3: redirect	If step 2 Successful, pass test

UC-3: View Profile

Test Case	TC-3
Use Case being tested	ViewProfile
Criteria for Success/Fail	Test that a user can view their profile details
Input Data:	Alphanumeric
Test Procedure:	
Step 1: Create a mock Database connection	Use Mock library to create a fake database connection which simulates the table we are going to query.
Step 2: Use data models with mock Database	Create an instance of the our data models with mock database connection and define sample input data
Step 3: Define Expected Queries	Create definitions for Expected queries to our mock database
Step 4: Call Function	We call the ReadProfile Function With Sample data which was previously defined and print errors in any Otherwise move to next step
Step 5: Define Expected Return Value	Define Expected Return Value on our models
Step 6: Compare Results	Compare to see if Expected results are the same as the actual returned results
Step 7: Check for other database Expectation	Verify that Database did not had more Queries/Inserts than previously defined

UC-8 File a Report

Test Case	TC-4
Use Case being tested	FileAReport
Criteria for Success/Fail	Test if a user can generate and submit a report and if it was posted to the database
Input Data:	Alphanumeric
Test Procedure:	
Step 1: Create a mock Database connection	Use Mock library to create a fake database connection which simulates the table we are going to query.
Step 2: Use data models with mock Database	Create an instance of the our data models with mock database connection and define sample input data
Step 3: Define Expected results	Create definitions for Expected queries to our mock database
Step 4: Call Function	We call the Insert Function With Sample data which was previously defined and print errors in any Otherwise move to next step
Step 5: Check Return Value	Check if Function Returned previously defined Expected ID (1), if not, fail test
Step 6: Check for other database Expectation	Verify that Database did not had more Queries/Inserts than previously defined

UC-14 Log Report

Test Case	TC-5
Use Case being tested	LogReport
Criteria for Success/Fail	Test that security guards can log information and if it was posted to the database
Input Data:	Alphanumeric
Test Procedure:	
Step 1: Create a mock Database connection	Use Mock library to create a fake database connection which simulates the table we are going to query.
Step 2: Use data models with mock Database	Create an instance of the our data models with mock database connection and define sample input data
Step 3: Define sample input data	Create definitions for Input data to our mock database and Define Expected Query
Step 4: Call Function	We call the InsertLog Function With Sample data which was previously defined and print errors in any Otherwise move to next step
Step 5: Check Return Value	Check if Function Returned previously defined Expected ID , if not, fail test
Step 6: Check for other database Expectation	Verify that Database did not had more Queries/Inserts than previously defined

UC-17 Post to Notice Board

Test Case	TC-6
Use Case being tested	PostToNoticeBoard
Criteria for Success/Fail	Test that admins can post information to notice boards
Input Data:	Alphanumeric
Test Procedure:	
Step 1: Create a mock Database connection	Use Mock library to create a fake database connection which simulates the table we are going to query.
Step 2: Use data models with mock Database	Create an instance of the our data models with mock database connection and define sample input data
Step 3: Define sample Data & Query	Define sample data And Query that is to be inserted to mock database
Step 4: Call Function	Call Function With sample Data
Step 5: Verify Return Value	Verify if returned ID matched Expected ID which was previously defined
Step 6: Check for other database Expectation	Verify that Database did not had more Queries/Inserts than previously defined

Project Management & Plan of Work

Issues Encountered

Compiling this report faced challenges due to scheduling conflicts among team members who juggle work and daily responsibilities, resulting in difficulties in coordinating meetings and setting deadlines, as well as making key decisions regarding programming languages and task assignments. Additionally, poor time management in certain phases of the report process was influenced by external factors. One significant challenge that nearly caused us to start over was the unexpected corruption of the database. It occurred seemingly out of nowhere, resulting in the loss of all updated tables and data. This setback necessitated extensive efforts to troubleshoot the issue and ultimately required us to rebuild the database from scratch. Despite the frustration and delay it caused, we remained committed to ensuring the integrity and functionality of our system. Through perseverance and teamwork, we successfully overcame this obstacle and ultimately strengthened our understanding of database management and system resilience. Other than that, everything was smooth sailing and we got everything done in time.

TASK TITLE	Assigned To	START DATE	DUE DATE	DURATION	PHASE ONE												PHASE TWO														
					WEEK 1					WEEK 2					WEEK 3					WEEK 4				WEEK 5				WEEK 6			
					M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T	W	R	F	M	T
Report 2: System Design		2/28/24	4/3/24	35																											
Analysis and Domain Modeling		3/5/24	3/10/24																												
Conceptual Model	Tamika, Raynisha	3/5/24	3/8/24	3																											
System Operation Contracts	Javier, Alex	3/5/24	3/10/24	5																											
Data Model and Persistent Data Storage	Tamika	3/5/24	3/10/24	5																											
Interaction Diagrams		3/5/24	3/10/24																												
Interaction Diagrams	Javier, Tamika	3/5/24	3/10/24	5																											
Class Diagrams	Javier, Abner, Raynisha, Tamika, Alex	3/5/24	3/10/24	5																											
Data Types and Operation Signatures	Alex, Raynisha	3/5/24	3/10/24	5																											
Traceability Matrix	Tamika, Javier	3/10/24	3/10/24	1																											
Algorithms and Data Structures		3/10/24	4/3/24																												
Data Structures	Raynisha	3/10/24	3/10/24	1																											
User Interface Design and Implementation	Tamika, Raynisha, Abner, Alex, Javier	3/10/24	3/16/24	6																											
Design of Tests	Abner, Tamika	3/15/24	3/30/24	15																											
Project Management and Plan of Work	Tamika, Raynisha, Abner	2/28/24	4/3/24	34																											

References

Booch, G., & Jain, S. (2024, January 3). *Domain Modeling - Software Engineering*. GeeksforGeeks. Retrieved March 20, 2024, from <https://www.geeksforgeeks.org/software-engineering-domain-modeling/>

Tuck, K. (2020, February 21). *Regression testing can never be fully automated*. Kevin Tuck. Retrieved March 31, 2024, from <https://kevintuck.co.uk/regression-testing-can-never-be-fully-automated/>

