UTM
UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING
UTM Johor Bahru

SECJ 2203: Software Engineering

Semester 01, 2024/2025

System Documentation (SD)

Dengue Prevention and Education System

Version 2.0

12/1/2025

Prepared by:

1. **AHMAD MUZHAFFAR PRIHANTONY - A23CS4035**
2. **JASON JOEL JOHNNY - A23CS0091**
3. **LEE JIAN AI - A23CS0234**
4. **NGEOW ZHI YU - A23CS0255**

Presentation link:

https://drive.google.com/drive/folders/1ztEQTQNzB861G8_aYFQb_wXqGX8RPF7f?usp=sharing

Revision Page

a. **Overview**

The current version of system documentation involves system architectural design, detailed description of components which involves each description of subsystem, data design, user interface design, requirements matrix, test cases, data and expected results as well as appendices.

Target Audience

Stakeholder

b. **Project Team Members**

Member Name	Role	Task	Status
AHMAD MUZHAFFAR PRIHANTONY	Recorder	• 8.7 TC007 • 8.8 TC008	Complete
JASON JOEL JOHNNY	Researcher	• 8.5 TC005 • 8.6 TC006	Complete
LEE JIAN AI	Accuracy Checker	• 8.3 TC003 • 8.4 TC004	Complete
NGEOW ZHI YU	Moderator	• 8.1 TC001 • 8.2 TC002	Complete

c. **Version Control History**

Version	Primary Author(s)	Description of Version	Date Completed
2.0	Ngeow Zhi Yu	Completed the SDD- Software Design Description and STD- Software Test Description	11/1/2024

Table of Contents

1. Introduction

1.1 Purpose

The purpose of the SD was to define all the requirements that were needed in the Dengue Prevention and Education system. Software requirements specifications, or SRSs, are documents that list all of the features and capabilities that software must have. They also specify any prerequisites that must be met by the system. We strive to produce and develop a new Augmented Reality (AR) Dengue Awareness and Education Application that will be improved in order to add more enhanced features. In this document, all the user expectations towards the software will be specified. We will use a use case description along with a sequence diagram for each use case in the SRS to get a more detailed understanding of each process in the system. All the documents related to the Dengue Prevention and Education system will be recorded in order to view all the issues and improvements that can be made towards each process clearly. The requirements have been gathered from 7 November 2024 from the stakeholders, and all of them are recorded in the SD.

1.2 Scope

The software product is **Dengue Defender**, a Dengue Prevention and Education System that aims to raise public awareness and promote preventive actions against dengue outbreaks. It provides real-time dengue case statistics, high-risk zone alerts, and educational resources to help individuals and communities stay informed and prepared. Key features include augmented reality (AR) scanning for interactive engagement, gamification elements like quizzes to enhance learning, and offline accessibility to ensure information is available even in areas with limited connectivity. The system's primary objectives are to reduce dengue cases, support proactive measures, and foster community engagement through accessible and innovative tools. By combining technology, education, and user-friendly features, Dengue Defender seeks to improve public health and empower users to combat dengue effectively.

1.3 Definitions, Acronyms and Abbreviation

SD – System Documentation

SRS – Software Requirements Specification

SDLC - Software Development Life Cycle

AR - Augmented Reality

1.4 References

1. Amanda Athuraliya. (2022, December 12). Sequence Diagram Tutorial-Complete Guide with Examples. https://creately.com/guides/sequence-diagram-tutorial/
2. Joan Ang. (2023, Jun 08). Activity Diagram: Examples, How to Draw, Benefits. https://venngage.com/blog/activity-diagram/
3. UXPin. (2024, March 13). 7 Constraints in Design and How to Overcome Them https://www.uxpin.com/studio/blog/constraints-in-design/
4. Ramesh, R., & Reddy, C. S. (2021). Metrics for software requirements specification quality quantification. *Computers & Electrical Engineering*, *96*, 107445. https://doi.org/10.1016/j.compeleceng.2021.107445

1.5 Overview

The current version of the Software Design (SD) document provides a detailed explanation of the software's requirements, covering external interfaces and system features. For external interfaces, it describes the user interface, hardware interface, software interface, and communication interface, ensuring all interaction points are well-defined. System features are explained through multiple diagrams, including use case, activity, domain model, and state machine diagrams, each accompanied by detailed descriptions for clarity. The document also addresses performance and other key system requirements. Future iterations will expand to include a detailed architectural design, comprehensive descriptions of components, refined data and user interface designs, and well-defined test cases. The SD document evolves in alignment with the Software Development Life Cycle (SDLC) phases, ensuring all stages from requirements analysis to testing are documented with clarity and precision.

2. Specific Requirements

2.1 User Roles

User roles in software development represent groups of users type and their interactions with our system, Dengue Prevention and Education System. We have three user roles in this system which are user, admin and health officer.

2.1.1 User Role 1: < User >

User Need

A user needs a way to manage their own account to log in to the system so that they can check dengue risk and report breeding sites as well as complete the quiz.

User Stories

US001: As a user, I want to be able to log into the system so that I can use the Dengue Prevention and Education System.

US002: As a user, I want to check Dengue Risk so that I can avoid traveling to high-risk areas.

US003: As a user, I want to report the breeding site so that the system can immediately update the area's risk level.

US004: As a user, I want to complete a quiz so that I can test my knowledge about the dengue topic.

2.1.2 User Role 2: < Admin >

User Need

An admin needs a way to manage users and ensure system data is backed up securely to maintain the system's integrity and usability.

User Stories

US005: As an admin, I want to manage user accounts so that I can maintain system organization and operational functionality.

US006: As an admin, I want to back up data so that I can ensure information security and availability in case of technical issues.

2.1.3 User Role 3 : < Health Officer >

User Need

A health officer needs to generate reports on dengue cases and send any alerts needed to inform users about the dengue high-risk areas.

User Stories

US007: As a health officer, I want to analyze and monitor patterns of a dengue case by creating reports on it.

US008: As a health officer, I want to notify the users to inform them about the areas that have a high risk for dengue spreading.

2.2 System Features

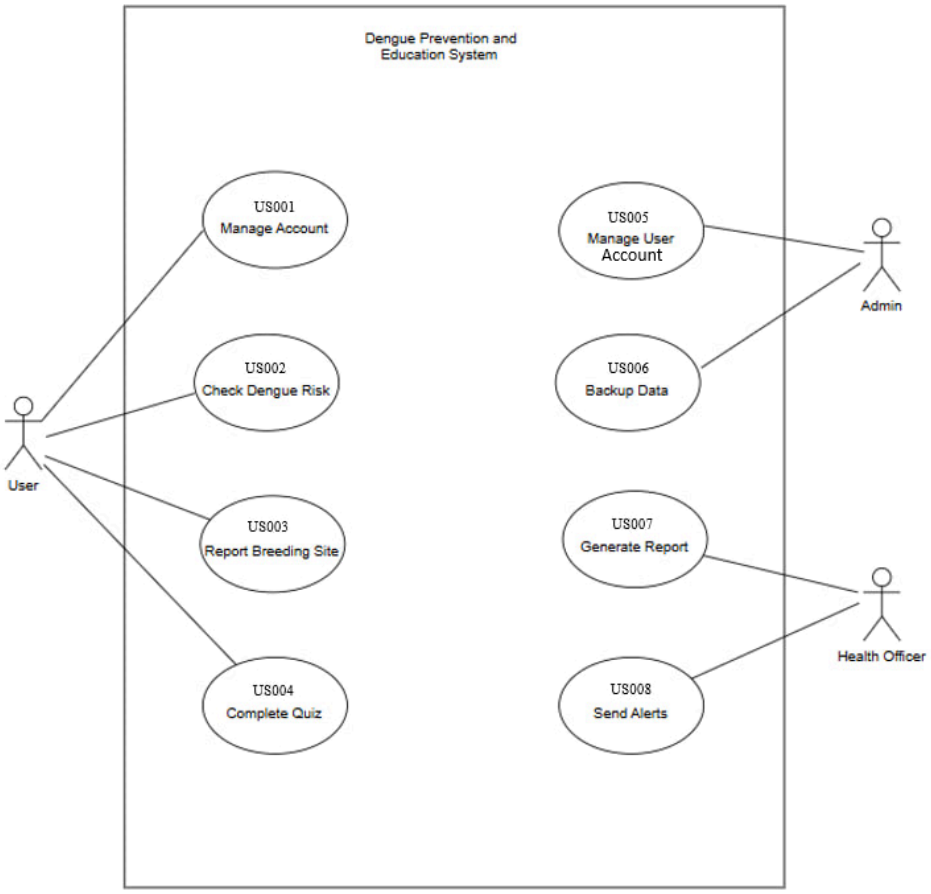


Figure 2.2.1: Use Case Diagram for <Dengue Prevention and Education System>

Table 2.0: Description of Module and Functions for <Dengue Prevention and Education System>

Use case	Function	Description
UC001	Manage Account	This use case allows users to manage their account settings, such as changing their password or profile information.
UC002	Check Dengue Risk	This use case allows users to check the dengue risk level in their area.
UC003	Report Breeding Site	This use case allows users to check the dengue risk level in their area.
UC004	Complete Quiz	This use case allows users to take a quiz to test their knowledge about dengue prevention.
UC005	Manage User Account	This use case allows administrators to manage user accounts, such as adding, deleting, or modifying user information.

UC006	Backup Data	This use case allows administrators to back up system data to ensure data integrity and security.
UC007	Generate Report	This use case allows administrators to generate reports on dengue cases, breeding sites, and user activity.
UC008	Send Alerts	This use case allows the system to send alerts to users about dengue outbreaks or other relevant information.

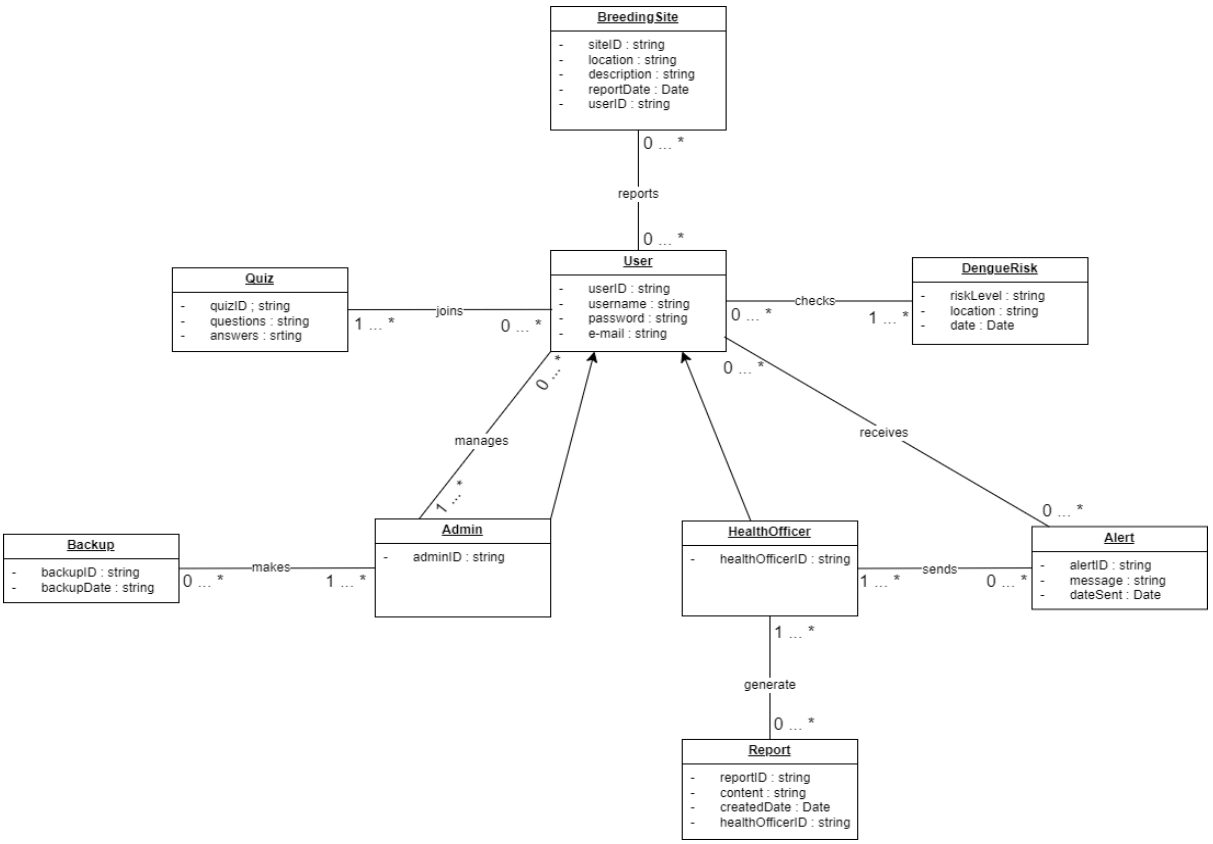


Figure 2.2.2: Domain Model for <Dengue Prevention and Education System>

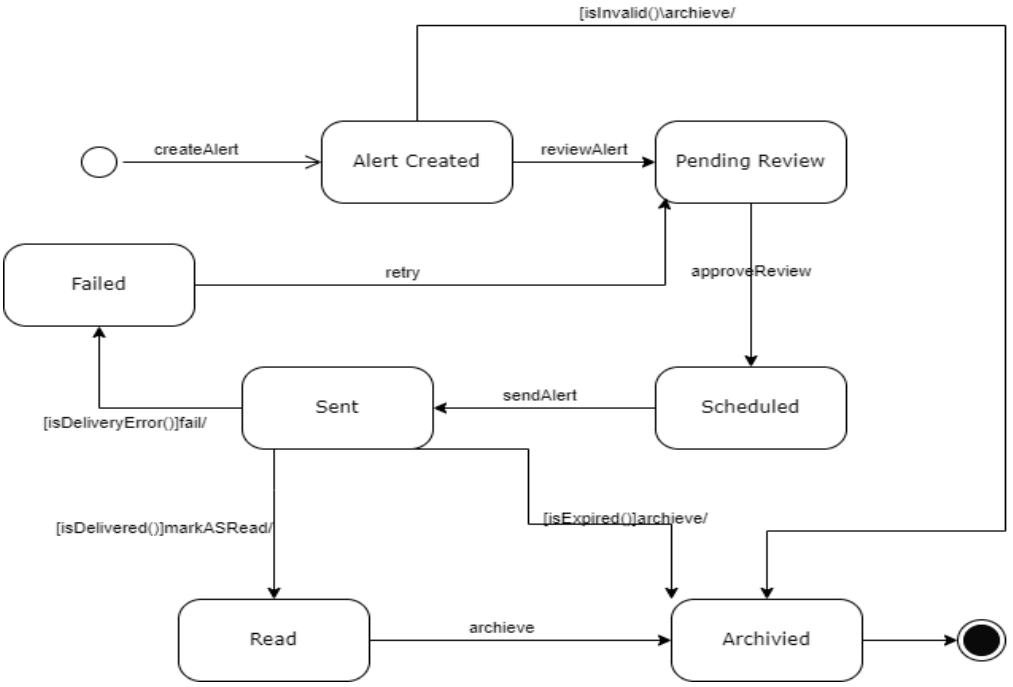


Figure 2.2.3: State Diagram for <Alert>

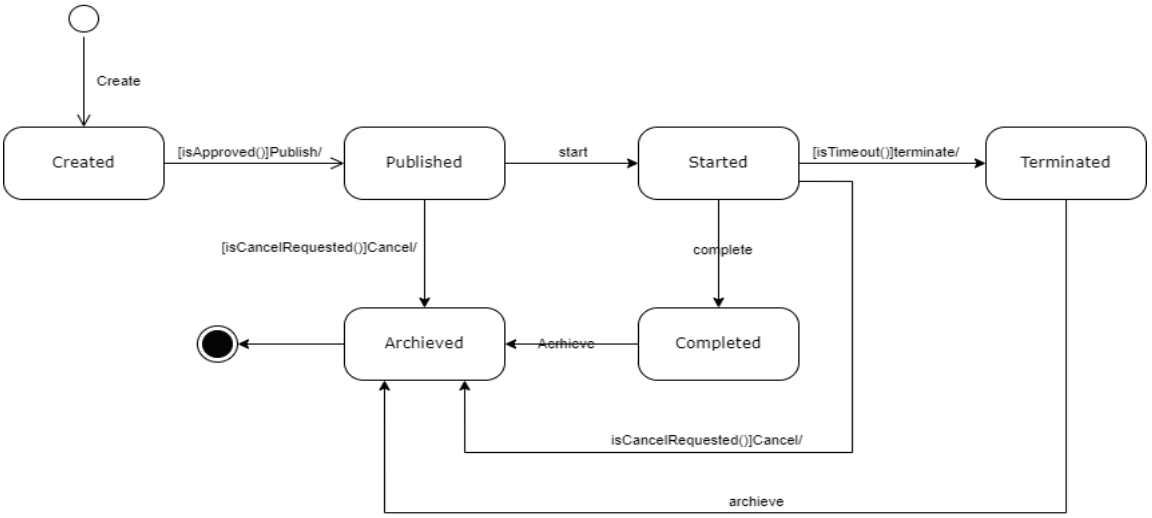


Figure 2.2.4: State Diagram for <Quiz>

2.3 Launch Phase

Sprint	Feature	ID	User Story	Status	Assignee
Sprint 1	Manage Account	1	As a user, I want to be able to log into the system So that I can use the Dengue Prevention and Education System.	Done	NGEOW ZHI YU
	Check Dengue Risk	2	As a user, I want to check Dengue Risk so that I can avoid traveling to high-risk areas.	Done	NGEOW ZHI YU
	Report Breeding Site	3	As a user, I want to report the breeding site so that the system can immediately update the area's risk level	Done	LEE JIAN AI
Sprint 2	Complete Quiz	4	As a user, I want to complete a quiz so that I can test my knowledge about the dengue topic.	Done	LEE JIAN AI
	Manage User Account	5	As an admin, I want to manage user accounts so that I can maintain system organization and operational functionality.	Done	JASON JOEL JOHNNY
	Data Backup	6	As an admin, I want to back up data so that I can ensure information security and availability in case of technical issues.	Done	JASON JOEL JOHNNY
Sprint 3	Generate Report	7	As a health officer, I want to produce reports on dengue cases to analyze and monitor patterns.	Done	AHMAD MUZHAFFAR PRIHANTONY
	Send Alerts	8	As a health officer, I want to notify users to inform them of areas with a high risk of dengue.	Done	AHMAD MUZHAFFAR PRIHANTONY

2.4 User Story Details

2.4.1 US001: User Story <Manage Account>

Table 2.1: User Story Description for <Manage Account>

User Story ID	US001
User Story Name	Manage Account
User Story Description	As a user, I want to be able to log into the system So that I can use the Dengue Prevention and Education System
Acceptance Criteria(s)	Precondition: User has registered an account and logged into the system.
Normal Flow(s)- NF	1. User access to the Dengue Prevention and Education System login page 2. User enters their username and password a. If user does not fill in all the fields, **AF1** is executed b. If user enters incorrect username or password, **AF2** is executed 3. Use case ends when user successfully logs in to their account
Alternative Flow(s) - AF	**AF1. Not fill in all fields:** 1. If user does not fill in all the fields at login page, system displays user should fill in all the fields to login 2. Return to **NF2** **AF2. Enters incorrect username or password:** 1. If the user enters an incorrect username or password, the system displays wrong username or password and prompt user to try again. 2. Return to **NF2**

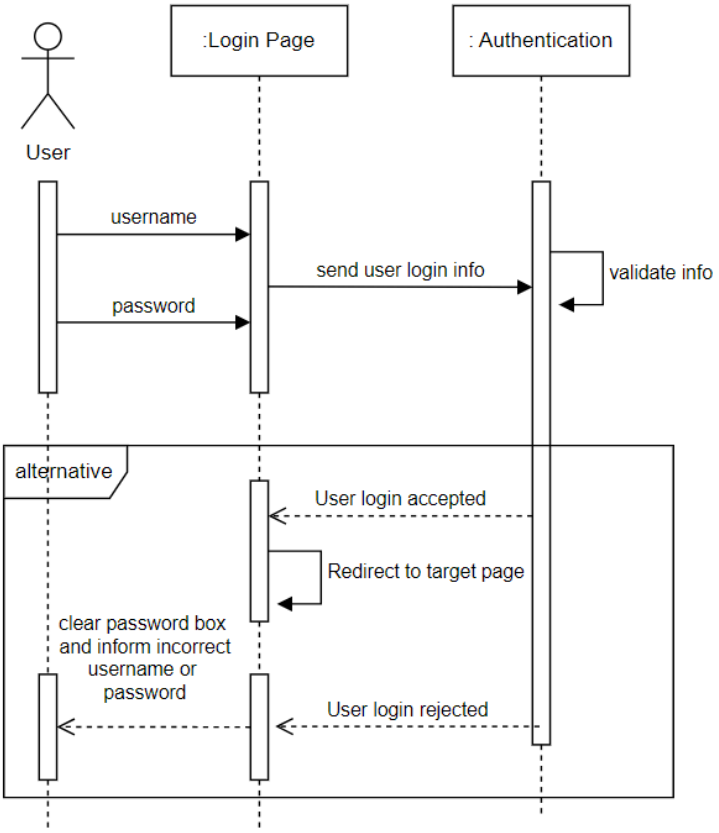


Figure 2.1.1: Sequence Diagram for <Manage Account>

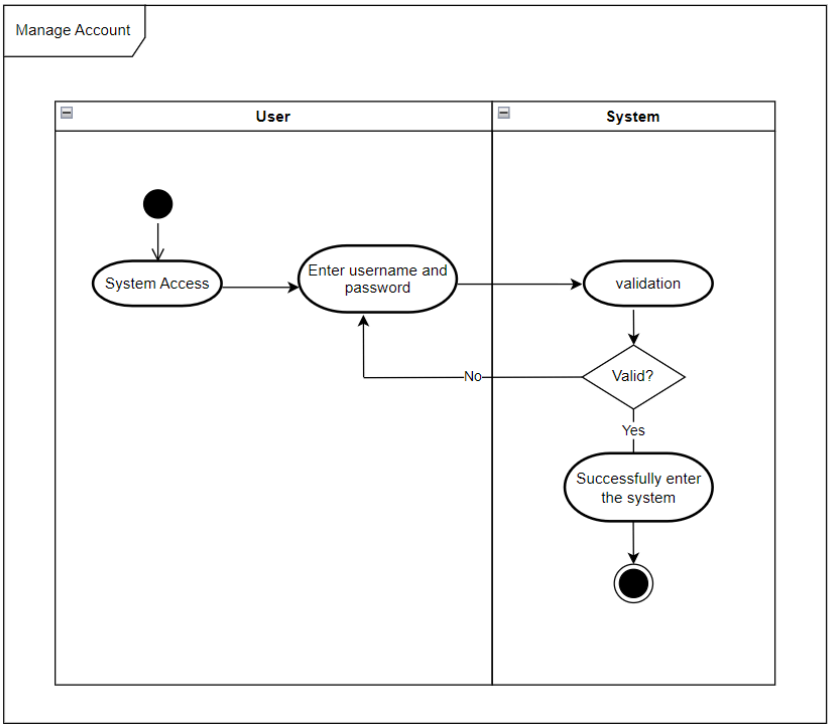


Figure 2.1.2: Activity Diagram for <Manage Account>

2.4.2 **US002: User Story <Check Dengue Risk>**

Table 2.2: User Story Description for <Check Dengue Risk>

User Story ID	US002
User Story Name	Check Dengue Risk
User Story Description	As a user, I want to check Dengue Risk so that I can avoid traveling to the high-risk area
Acceptance Criteria(s)	Precondition: The user has successfully logged into the system Postcondition: The system displays the Dengue Risk Level for each state Other condition: Internet connectivity is required for real-time data update
Normal Flow(s)-NF	1. Users review the Dengue Risk Check Page. 2. The system displays all the state's Dengue Risk level 3. Users enter their state and detail their location a. If the user does not fill in the fields and press the search button, **AF1** is executed b. If the user enters an invalid state or undefined location, **AF2** is executed 4. The system displays the Dengue Risk Level (High, Medium, Low) for the provided location.
Alternative Flow(s) - AF	**AF1. Not fill in the fields and press search button:** 1. If the user does not fill in the fields and press the search button, the system displays user should fill in the fields before pressing the search button. 2. Return to **NF3** **AF2. Enters incorrect username or password:** 1. If the user enters an invalid state or undefined location, the system displays invalid state or undefined location and prompts the user to re-enter the state or location. 2. Return to **NF3**

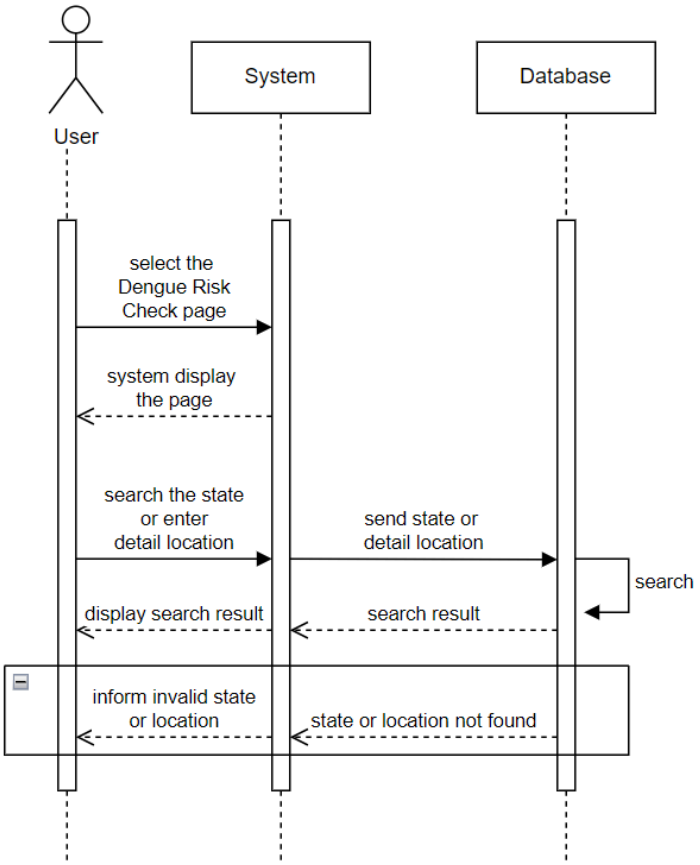


Figure 2.2.1: Sequence Diagram for <Check Dengue Risk>

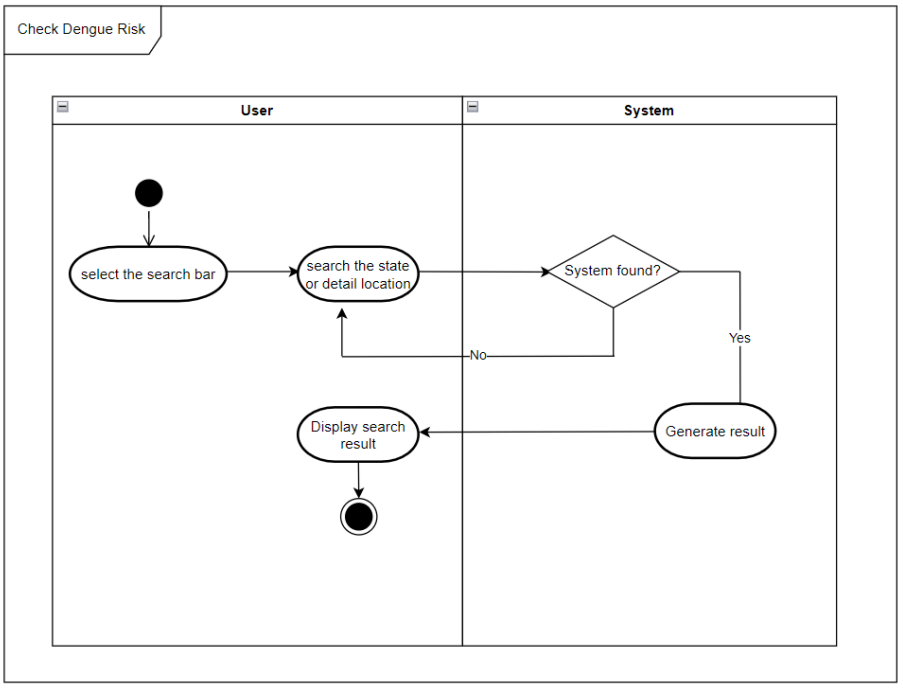


Figure 2.2.2: Activity Diagram for <Check Dengue Risk>

2.4.3 **US003: User Story <Report Breeding Site>**

Table 2.3 User Story Description for <Report Breeding Site>

User Story ID	US003
User Story Name	Report Breeding Site
User Story Description	As a user, I want to report the breeding site so that the system can immediately update the area's risk level.
Acceptance Criteria(s)	**Pre-condition:** The user has successfully logged into the system. **Post-condition:** The system records the reported case and updates the area's risk level **Other conditions:** Internet connectivity is required to ensure the case is reported
Normal Flow(s)- NF	1. The user navigates to the Report Problems button in the Dengue Risk checking page. 2. The user uploads a picture of the breeding site and writes a suitable description, along with the specific area where the problem was found. 2.1. If the user's report filing fails due to a bad internet connection or system failure, **EF1** will be executed. 2.2. If the user does not key in the required information (evidence photo, description, and location), **AF1** will be executed. 3. The system will receive the report, and proceed to update the area's risk level synchronously.
Alternative Flow(s) - AF	**AF1: Required information not keyed in for filing a report:** 1. If the user does not fill in the details for the breeding site report, the system will return to the top of the report page to prompt the user to fill in all the fields to report. 2. Return to **NF2.**
Exception Flow(s) - EF	**EF1: Report submission fails** 1. If the user fails to file the report due to a bad internet connection or system failure, an error message will show up to ask the user: 1.1. Check if their internet connection is stable. (Internet failure) 1.2. Try again later. (System failure) 2. Return to **NF2**.

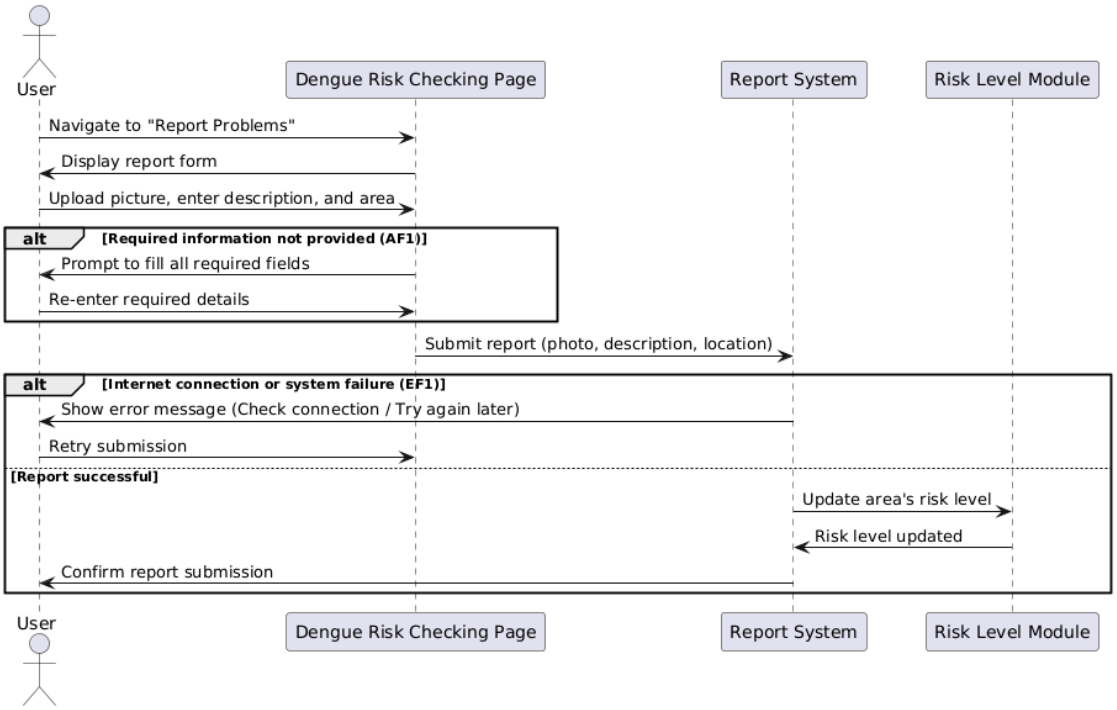


Figure 2.3.1: Sequence Diagram for <Report Breeding Site>

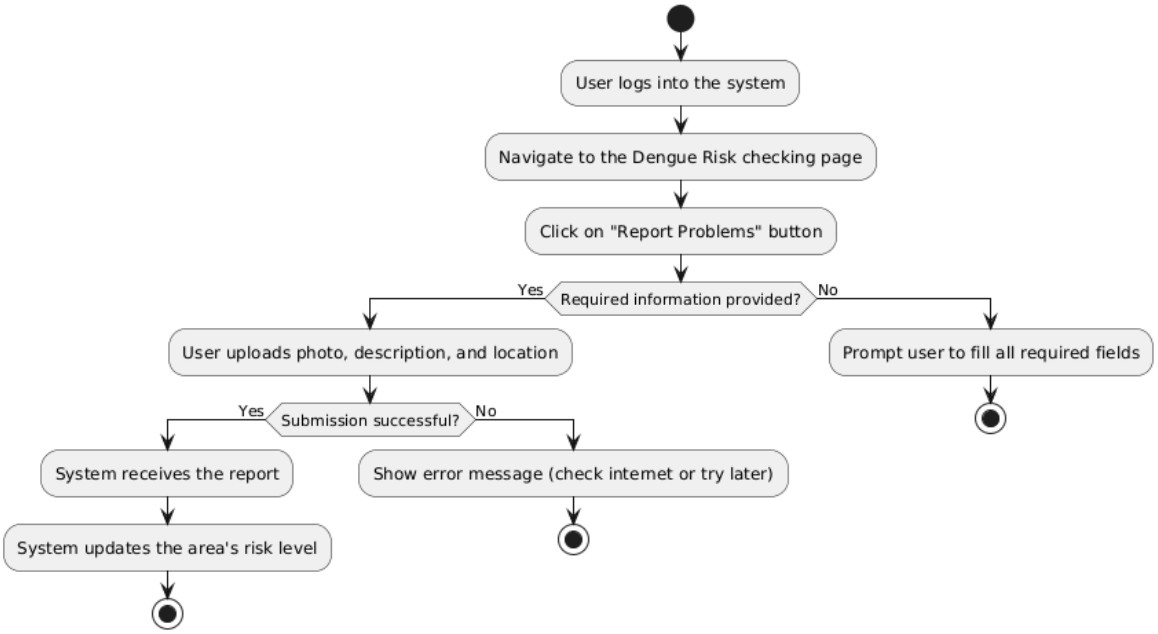


Figure 2.3.2: Activity Diagram for <Report Breeding Site>

2.4.4 **US004 User Story <Complete Quiz>**

Table 2.4 User Story Description for <Complete Quiz>

User Story ID	US004
User Story Name	Complete Quiz
User Story Description	As a user, I want to complete a quiz so that I can test my knowledge about the dengue topic
Acceptance Criteria(s)	**Pre-condition:** The user has successfully logged into the system. **Post-condition:** The system records the result of the quiz that the user has answered
Normal Flow(s)- NF	1. The user navigates to the Quizzes page. 2. The user selects one of the quiz banks related to the dengue topic. 3. The user starts answering the quiz. 3.1. If the user stops answering in the middle of the quiz, **AF1** will be triggered. 4. After the quiz is finished, the result will be shown and recorded.
Alternative Flow(s) - AF	**AF1: The quiz stopped to be answered in the middle of it:** 3. If the user does not finish answering the quiz, but rather they have stopped their progress in the middle of it, the system will show a message asking if the user really wants to quit as the progress will not be saved. 4. Return to **NF3.**

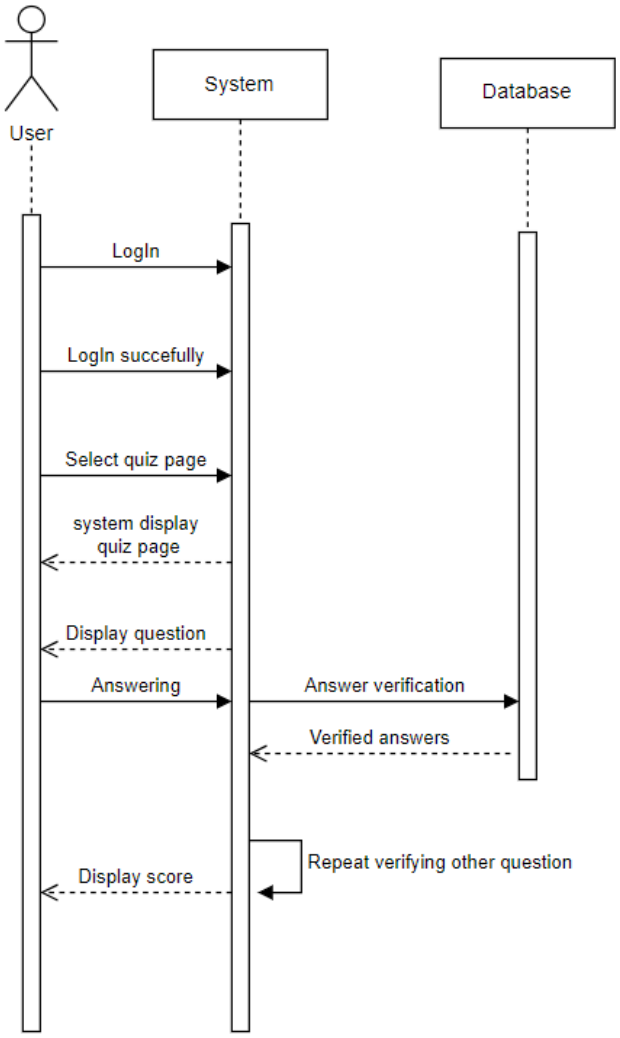


Figure 2.4.1: Sequence Diagram for <Complete quiz>

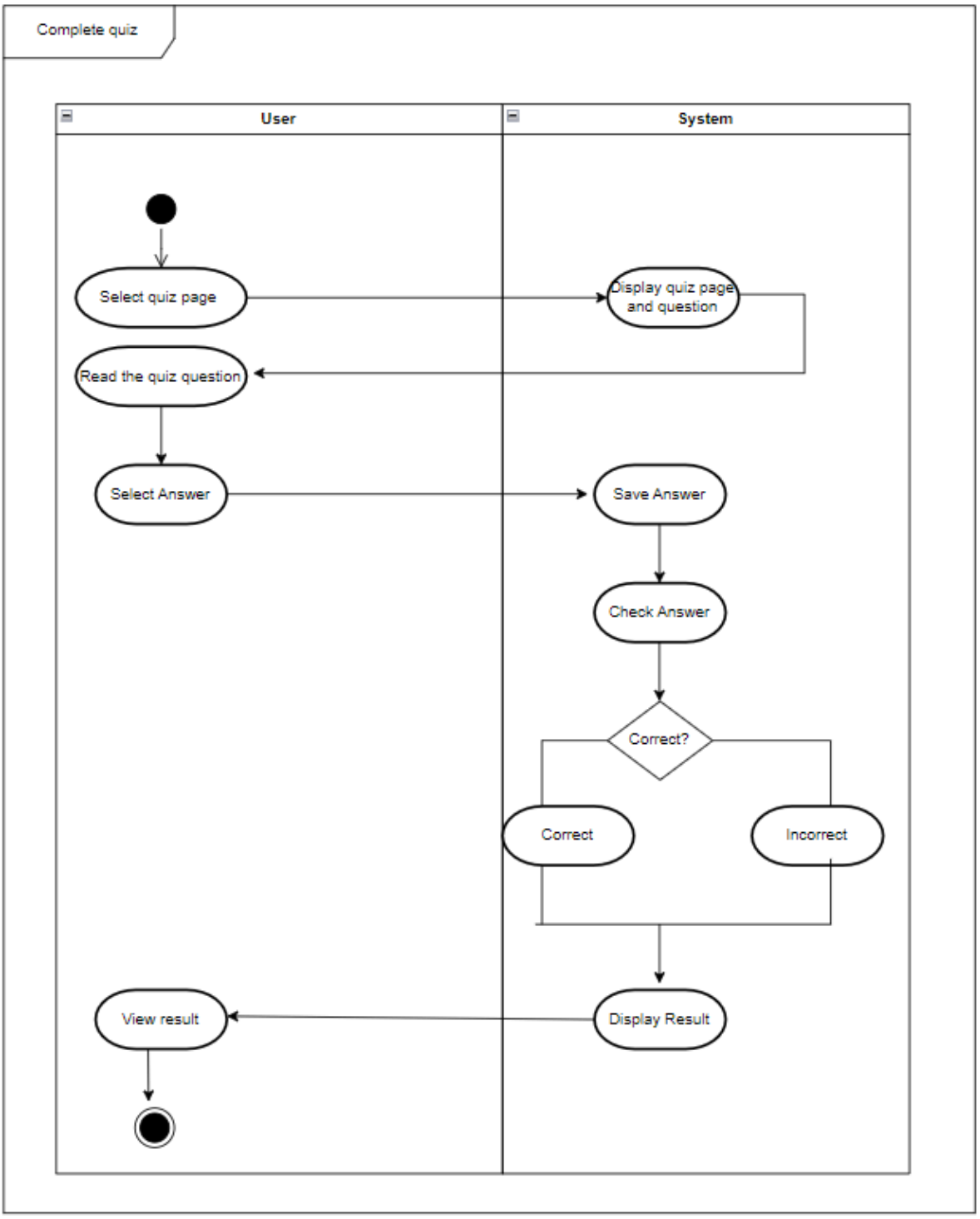


Figure 2.4.2: Activity Diagram for <Complete quiz>

2.4.5 US005 User Story <Manage User Account>

Table 2.5: User Story Description for <Manage User Account>

User Story ID	US005
User Story Name	Manage User Accounts
User Story Description	As an admin, I want to manage user accounts so that I can maintain system organization and operational functionality.
Acceptance Criteria(s)	**Pre-condition**: The admin is logged into the system. **Post-condition**: User accounts are updated (created, modified, or deleted) successfully. **Other Conditions**: Changes to user accounts are logged for audit purposes.
Normal Flow(s)- NF	1. The admin logs into the system. 2. The admin selects the "Manage User Accounts" option. 3. The admin performs one of the following actions: 3.1. Add a new user account. 3.2. Edit an existing user account. 3.3. Delete an existing user account. 3.3.1. If the admin cancels the operation, **AF1** is executed. 4. The system updates the user account database and displays a confirmation message.
Alternative Flow(s) - AF	AF1. **Admin cancels operation** 1. If the admin cancels the operation at any point, no changes are made to user accounts. 2. Return to **NF2**.

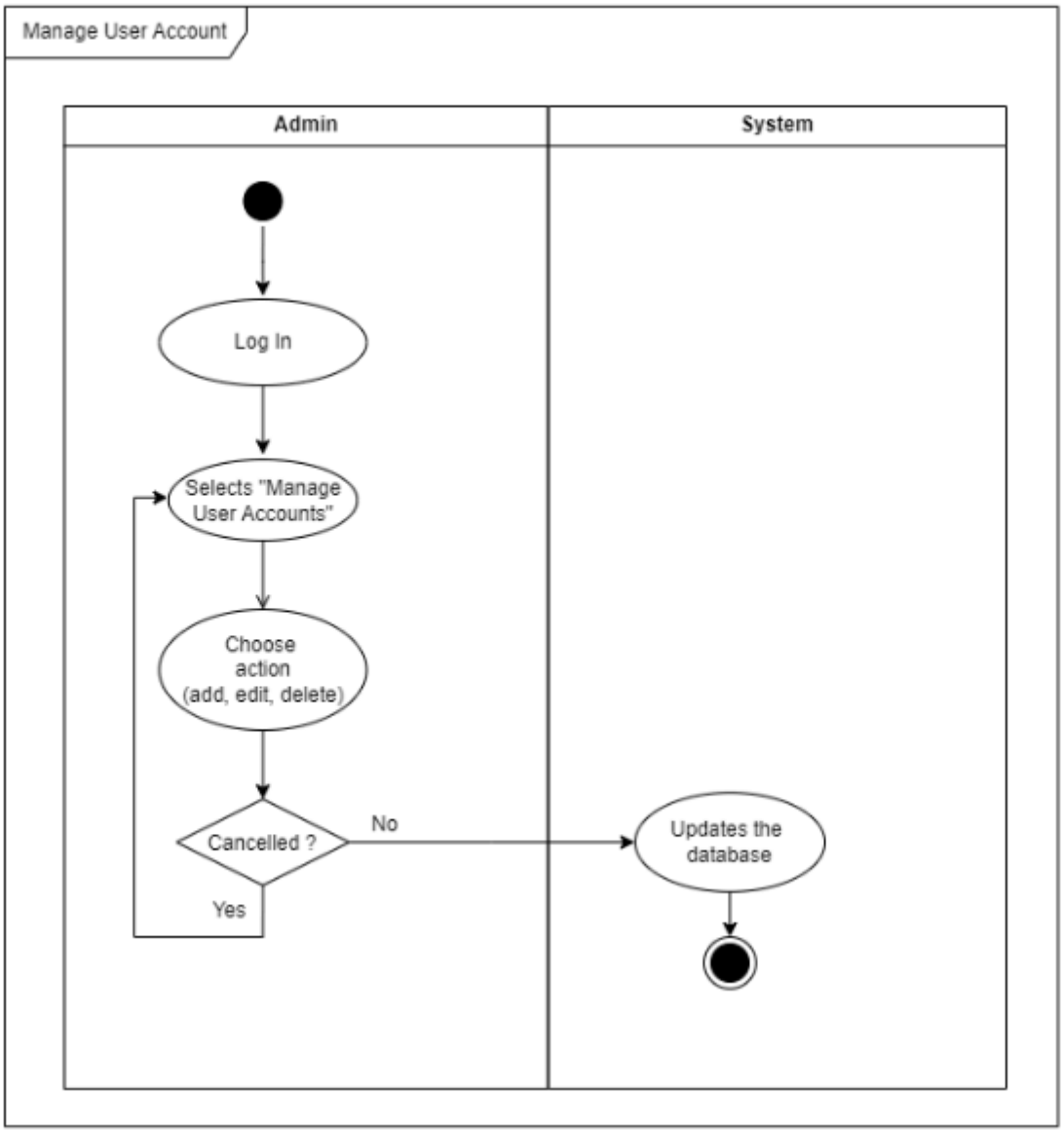


Figure 2.5.2: Activity Diagram for <Manage User Account>

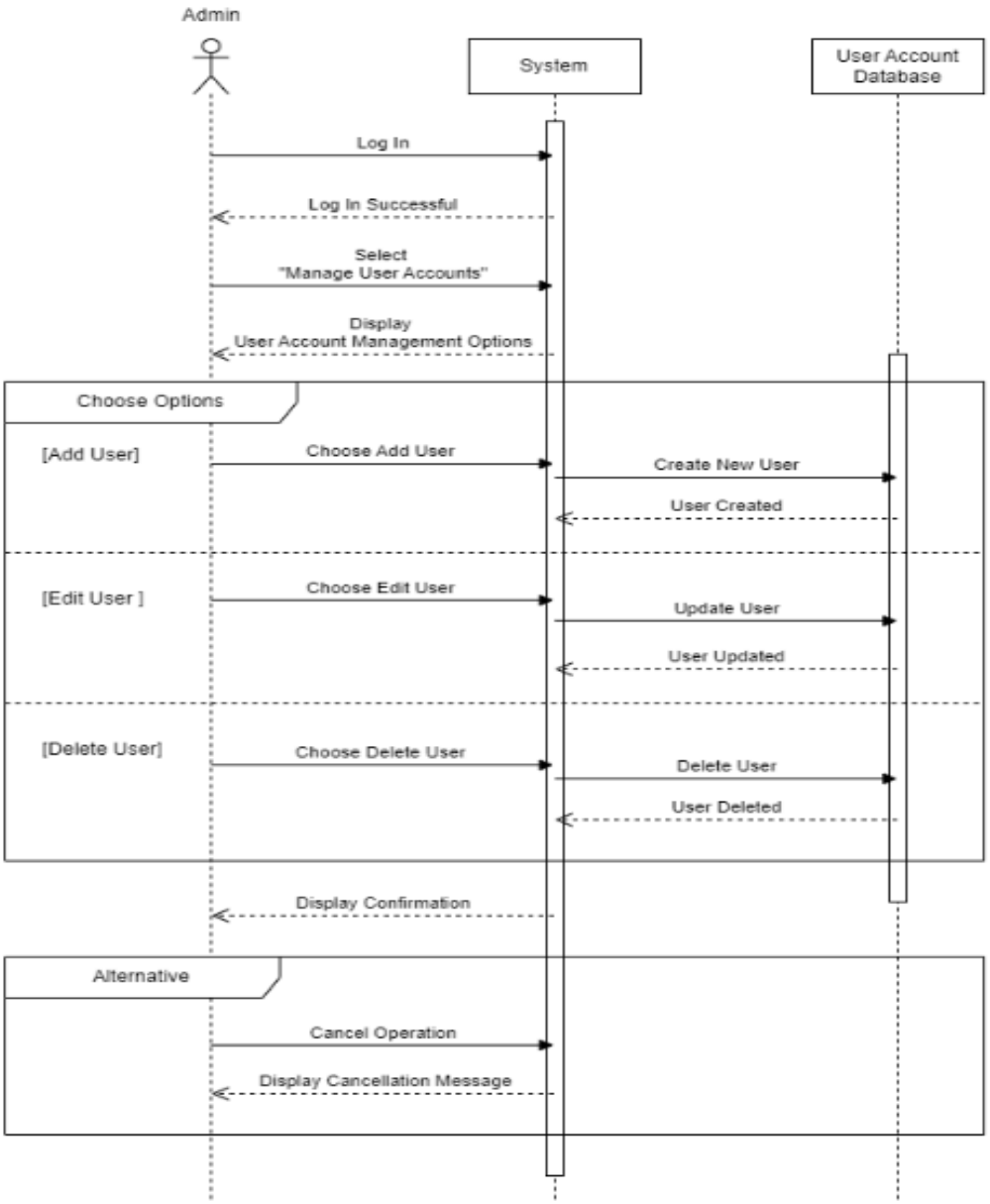


Figure 2.5.1: Sequence Diagram for <Manage User Account>

2.4.6 US006 User Story <Data Backup>

Table 2.6: User Story Description for <Data Backup>

User Story ID	US006
User Story Name	Data Backup
User Story Description	As an admin, I want to back up data so that I can ensure information security and availability in case of technical issues.
Acceptance Criteria(s)	**Pre-condition**: The admin is logged into the system. **Post-condition**: The system successfully creates a secure backup of data. **Other Conditions**: Backup logs are maintained with timestamps and admin details.
Normal Flow(s)-NF	1. The admin logs into the system. 2. The admin navigates to the "Data Backup" section. 3. The admin initiates the backup process by selecting the "Backup Now" option. a. If the admin schedules the backup process for a later time instead of performing it immediately, **AF1** is executed. 4. The system validates the admin's request and starts the backup. a. If the backup process fails due to insufficient storage or connectivity issues, **EF1** is executed. 5. The system completes the backup process and displays a confirmation message.
Alternative Flow(s) - AF	**AF1. Admin Schedules Backup Process For Later** 1. The admin schedules the backup process for a later time instead of performing it immediately. 2. The system logs the scheduling action and displays a confirmation message. 3. Return to **NF2**.
Exception Flow(s) - EF	**EF1. The Backup Process Fails** 1. System displays an error message and logs the failure for review. 2. Return to **NF4**.

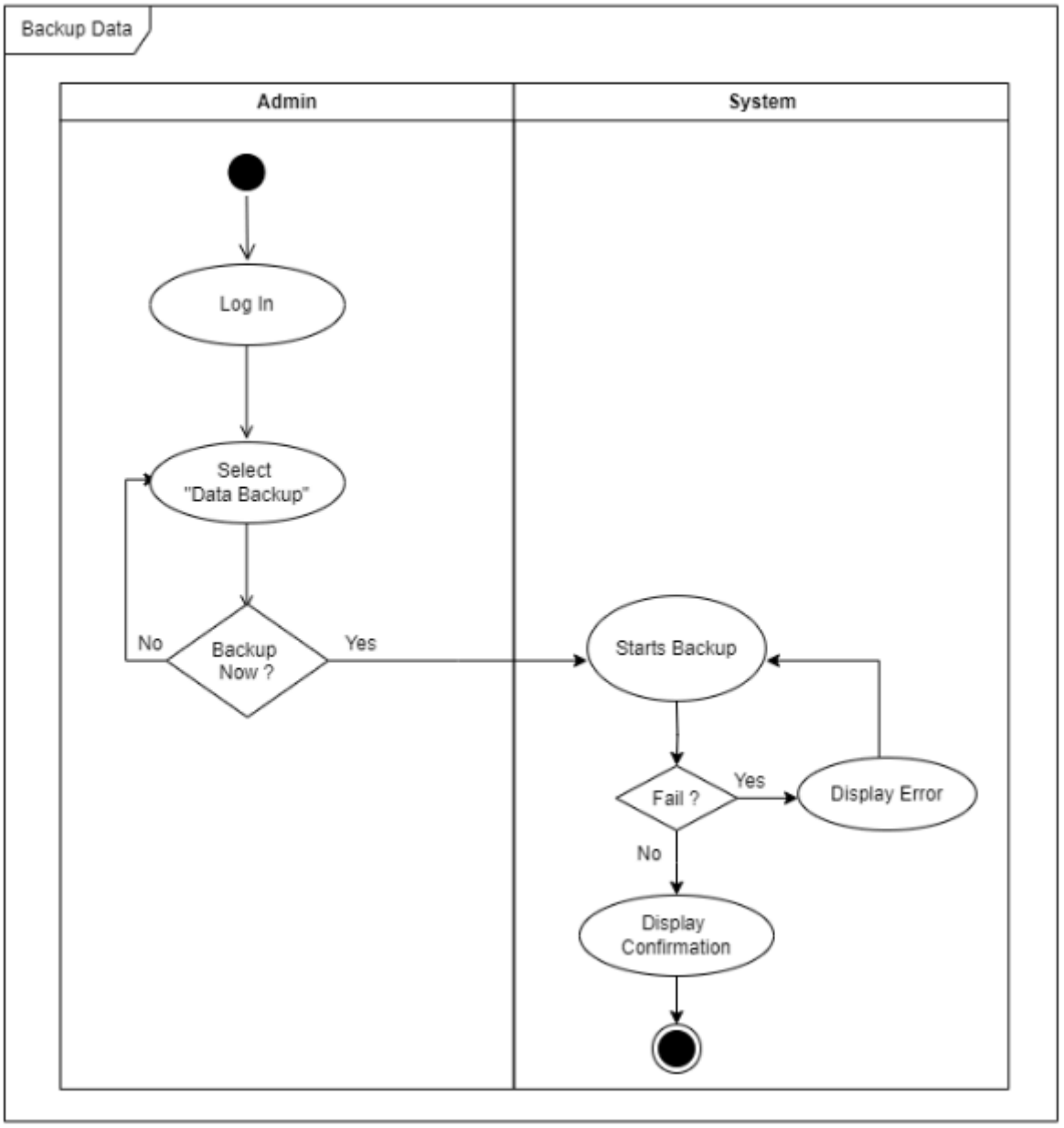


Figure 2.6.2: Activity Diagram for <Backup Data>

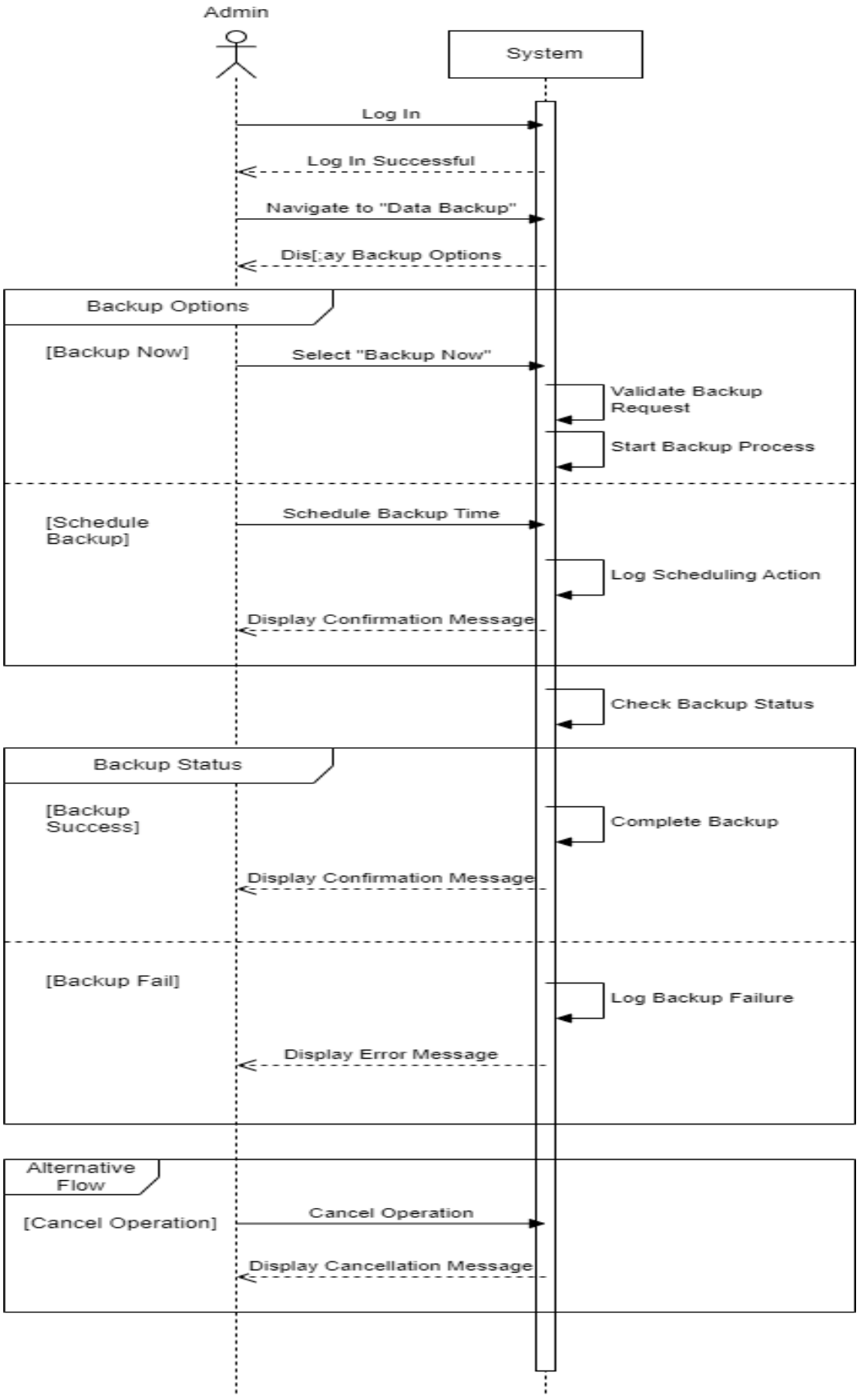


Figure 2.6.1: Sequence Diagram for <Backup Data>

2.4.7 US007 User Story <Generate Report>

Table 2.7: User Story Description for <Generate Report>

User Story ID	US007
User Story Name	Generate Report
User Story Description	As a health officer, I want to produce reports on dengue cases to analyze and monitor patterns.
Acceptance Criteria(s)	**Pre-condition**: - The health officer is logged into the system. - Dengue case data is available in the system. **Post-condition**: Generate a report and save it in the system for future access.
Normal Flow(s)- NF	1. The health officer navigates to the "Generate Report" section. 2. The system prompts the health officer to select report parameters (e.g., time range, location). 3. The health officer specifies the parameters and confirms. 4. The system retrieves dengue case data based on the selected parameters. 5. The system processes the data and generates a report in the desired format (e.g., PDF, Excel). 6. The system notifies the health officer that the report has been generated and provides a download link.
Alternative Flow(s) - AF	**AF1. If there is insufficient data for the selected parameters** 1. The system notifies the health officer about the issue. 2. The health officer can adjust the parameters or cancel the process.

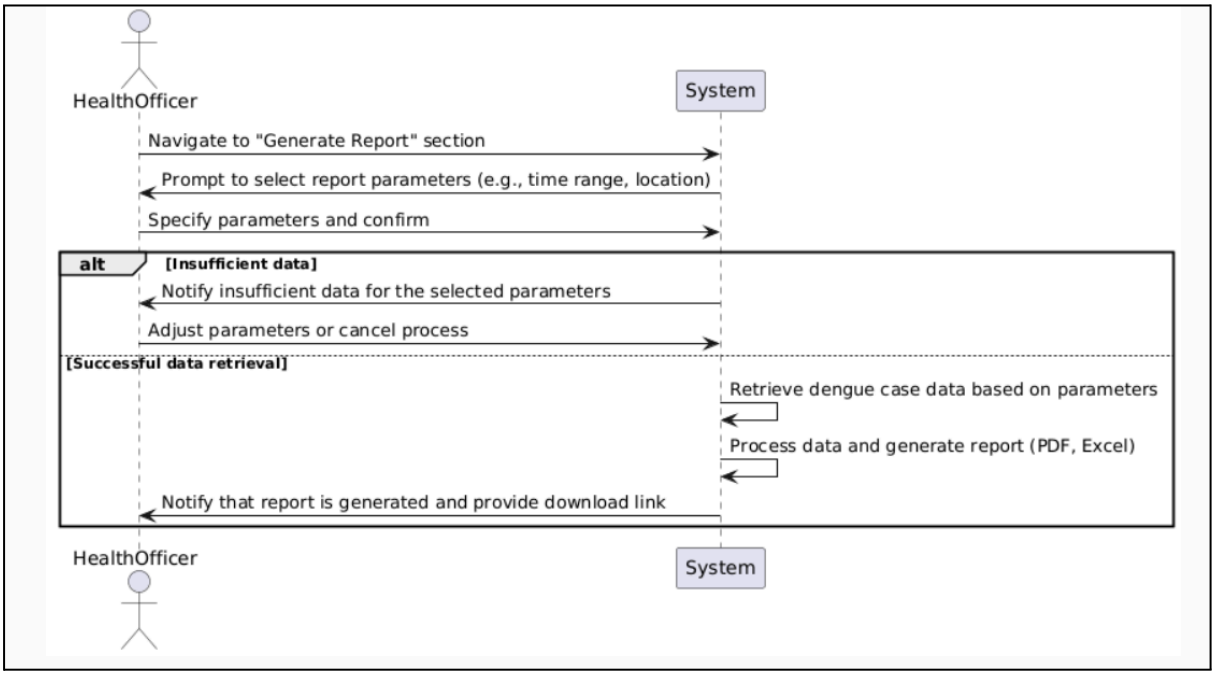


Figure 2.7.1: Sequence Diagram for <Generate Report>

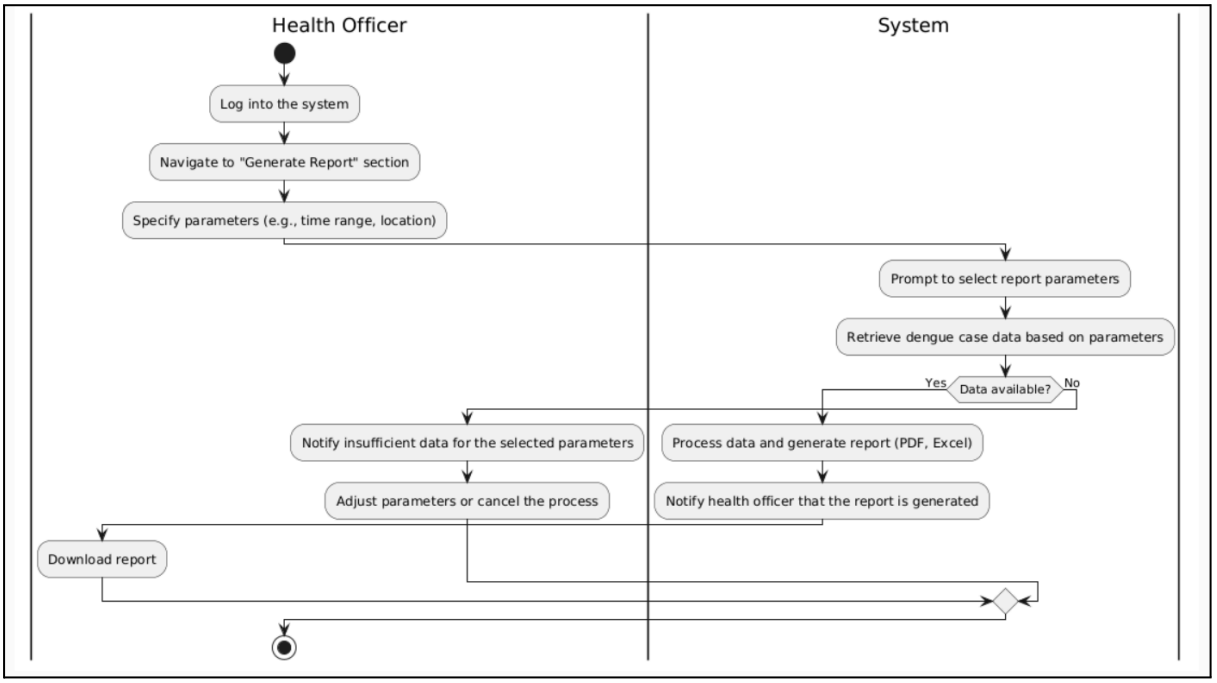


Figure 2.7.2: Activity Diagram for <Generate Report>

2.4.8 US008 User Story <Send Alerts>

Table 2.8: User Story Description for <Send Alerts>

User Story ID	US008
User Story Name	Send Alerts
User Story Description	As a health officer, I want to notify users to inform them of areas with a high risk of dengue.
Acceptance Criteria(s)	**Pre-condition**: - The health officer is logged into the system. - High-risk area data is available in the system. **Post-condition**: Alerts are sent to all relevant users.
Normal Flow(s)-NF	1. The health officer navigates to the "Send Alerts" section. 2. The system prompts the health officer to select the high-risk areas to include in the alert. 3. The health officer selects the areas and confirms the alert content. 4. The system retrieves the list of users in the selected areas. 5. The system sends the alert message to the identified users (via email, SMS, or app notifications). 6. The system notifies the health officer that the alerts have been sent successfully.
Alternative Flow(s) - AF	**AF1. If no users are found in the selected high-risk areas** 1. The system notifies the health officer about the issue. 2. The health officer can adjust the selected areas or cancel the process.

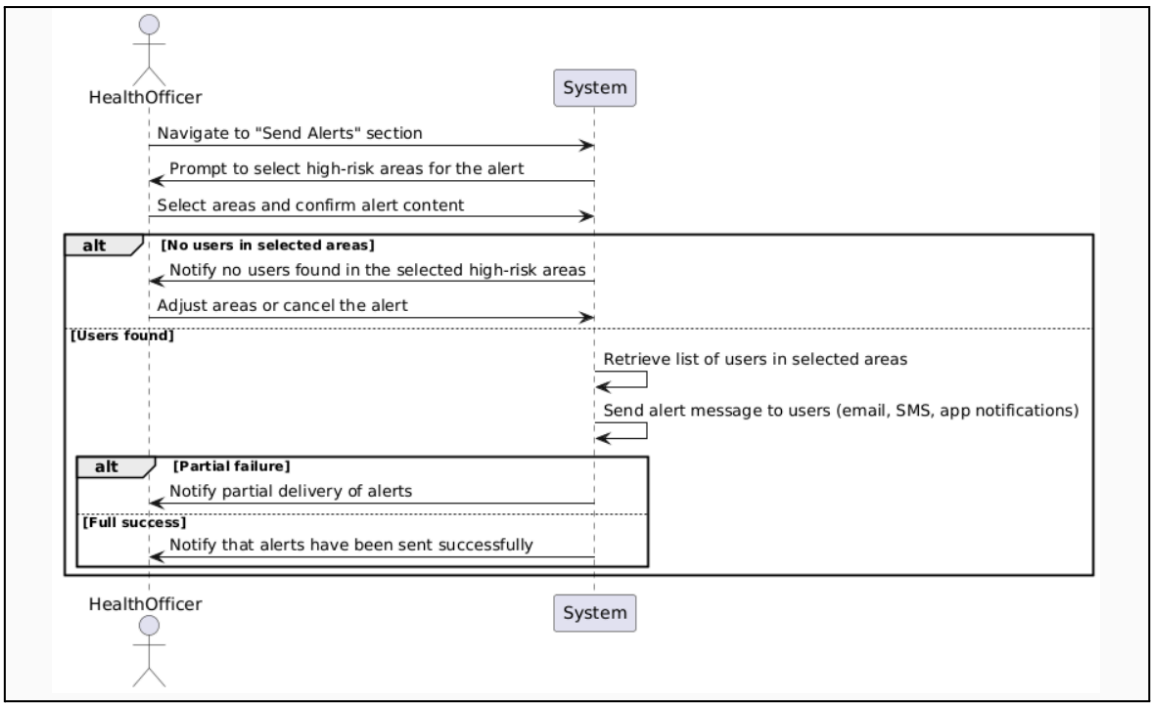


Figure 2.8.1: Sequence Diagram for <Send Alerts>

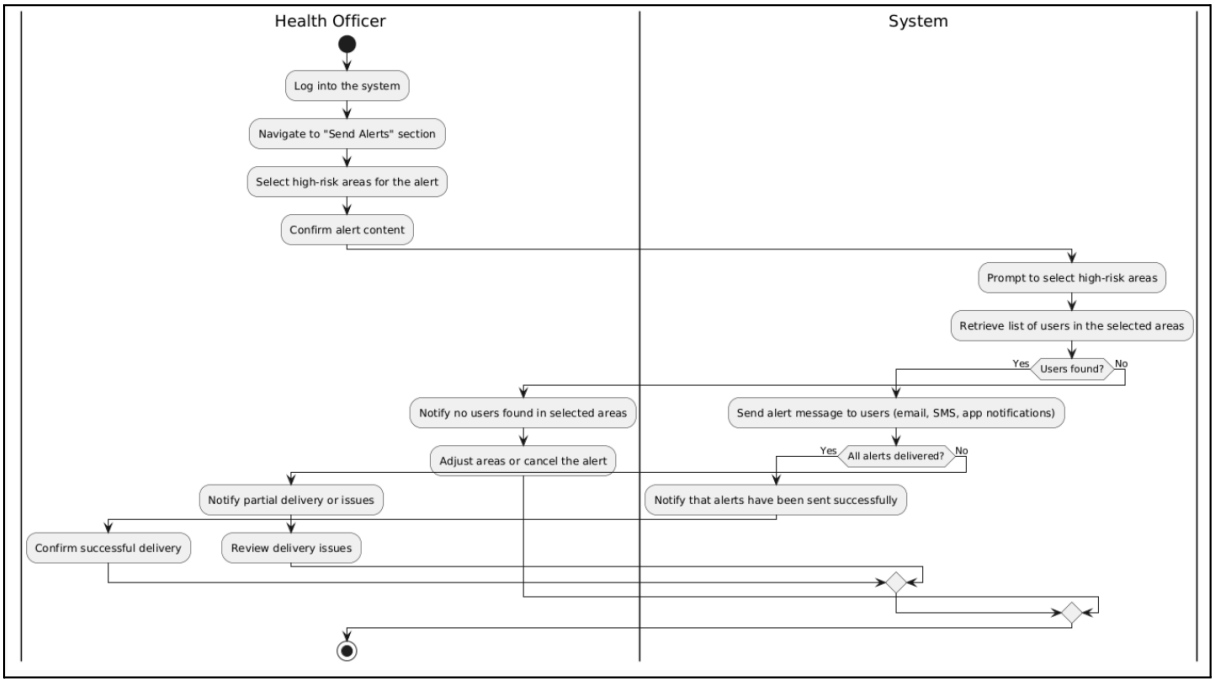


Figure 2.8.2: Activity Diagram for <Send Alerts>

2.5 Performance and Other Requirements

- Response Time: The average response time per interface must be less than 2 seconds.
- Capacity: The system must be able to handle at least 500 concurrent users.
- Security: The system have authentication and data encryption to prevent unauthorized access
- Reliability: The system operates 24/7 and almost no downtime to serve users' needs.
  : The system should updated immediately when changes made in the database
- Usability: The system should ease to use by user to interact with system
- Efficiency: The system shall be able to verify user login information within 3 seconds
- Maintainability: The system must be able to maintain all the data despite there is a software update
- Availability: The system must connected to Internet
- Portability: The system shall be able to run on desktop, laptop, or mobile devices

2.6 Design Constraints

- Technical constraints
  - Device compatibility: The application needs to work on various mobile devices, including low-spec smartphones and tablets commonly used in both urban and rural settings.
  - Hardware requirements: AR features may require advanced sensors like gyroscopes, which may not be available on all devices.
  - Offline functionality: Limited internet access in rural areas requires offline storage of essential data, which may restrict the availability of real-time features.
  - Battery consumption: AR and real-time functionalities can rapidly drain device batteries, which can limit usage.
  - Data integration: Real-time data integration (such as dengue outbreak data and weather conditions) relies on having access to most recent health and meteorological databases.
  - Real-time processing: The app needs to manage geolocation information, provide real-time updates, and render AR efficiently to prevent lags or crashes.

- Cost constraints
  - Development budget: Developing advanced AR features, gamification elements, and predictive analytics may require significant resources, including skilled developers and specialized software tools.
  - Data storage & hosting: Hosting geolocation and real-time data may require scalable servers, increasing operational costs.
  - Maintenance cost: Continuous investment will be necessary for regular updates to AR models, gamification content, and real-time data integration.

- Legal & Ethical constraints
  - Data privacy: Location, health-related data, and user information must comply with any legal regulations and data privacy laws.
  - Content accuracy: Learning resources, predictive analytics, and tutorials need to be precise and approved by relevant health organizations to prevent misinformation.

- Environmental constraints
  - AR usage environment: Real-time Augmented Reality features typically require proper lighting and space for optimal functionality, which may limit usability in low-light areas.

3 System Architectural Design

3.1 Architecture Style and Rationale

The architecture style that we are going to implement is client and server architecture. This architecture, as the name implies, consists of two parties: a server and a client. There can be different servers providing various functionalities.

In our proposed application, the Dengue Prevention and Education System with AR capabilities, this architecture is highly suitable as it supports services like educational content, interactive AR experiences, quizzes, and a heatmap feature, all while allowing multiple users to access these services simultaneously.

For instance, there can be a database server that stores resources such as AR assets, videos, images, and other learning materials about dengue prevention and symptoms. Additionally, there can be an application server that handles interactive functionalities, including AR-based simulations, quizzes, access to the educational content, and a heatmap feature.

The heatmap functionality plays a crucial role in visualizing dengue-prone areas using real-time or historical data. It allows users to identify high-risk zones based on reported cases, enabling targeted preventive actions. This feature can retrieve and update data from the server dynamically, ensuring users have access to the latest information.

This setup allows users to engage with AR features, heatmaps, and educational content directly through the network without requiring downloads. However, users also have the option to pre-download specific materials for offline use.

Another advantage of this architecture is centralized data management and protection, as all critical data resides on the server. This centralization facilitates secure data handling and ensures the reliability of educational resources. Moreover, the independence of client nodes within the network simplifies system upgrades, minimizing disruptions to users and enabling seamless integration of new features and improvements.

3.2 Component Model

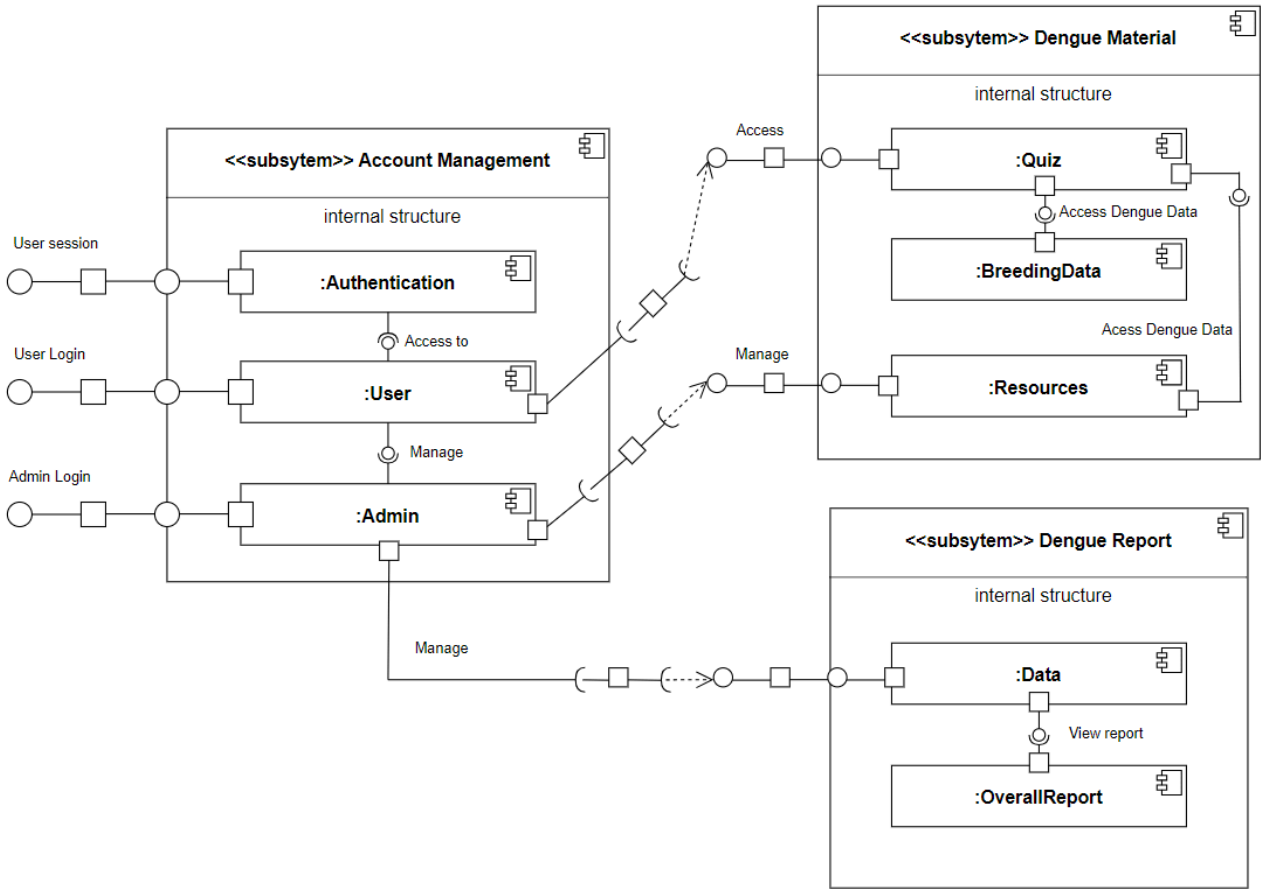


Figure 3.1: Component Diagram of <Dengue Prevention and Education System>

4 Detailed Description of Components

4.1 Complete Package Diagram

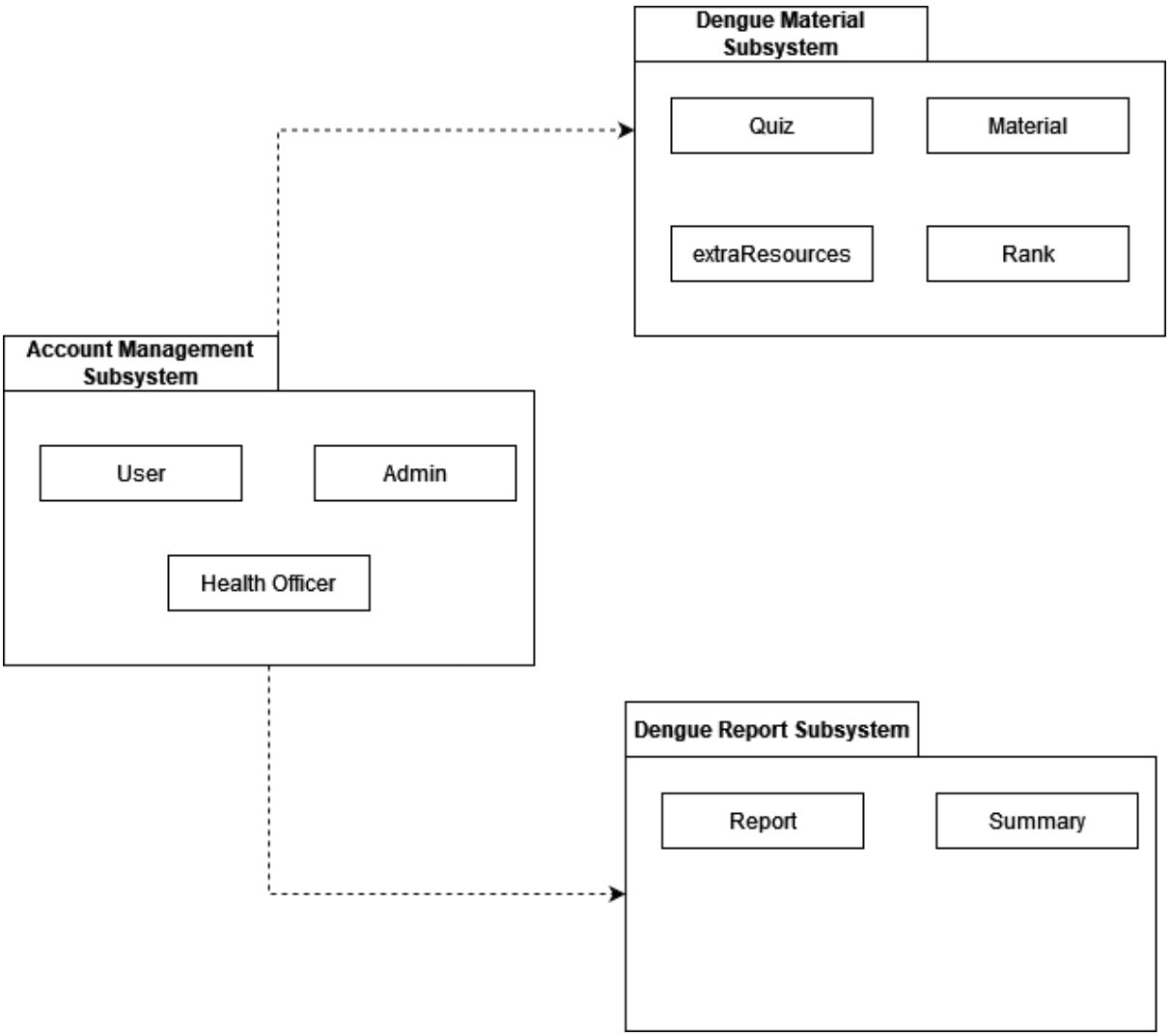


Figure 4.1: Package Diagram for <Dengue Prevention and Education System>

4.2 Detailed Description

4.2.1 P001: <Account Management> Subsystem

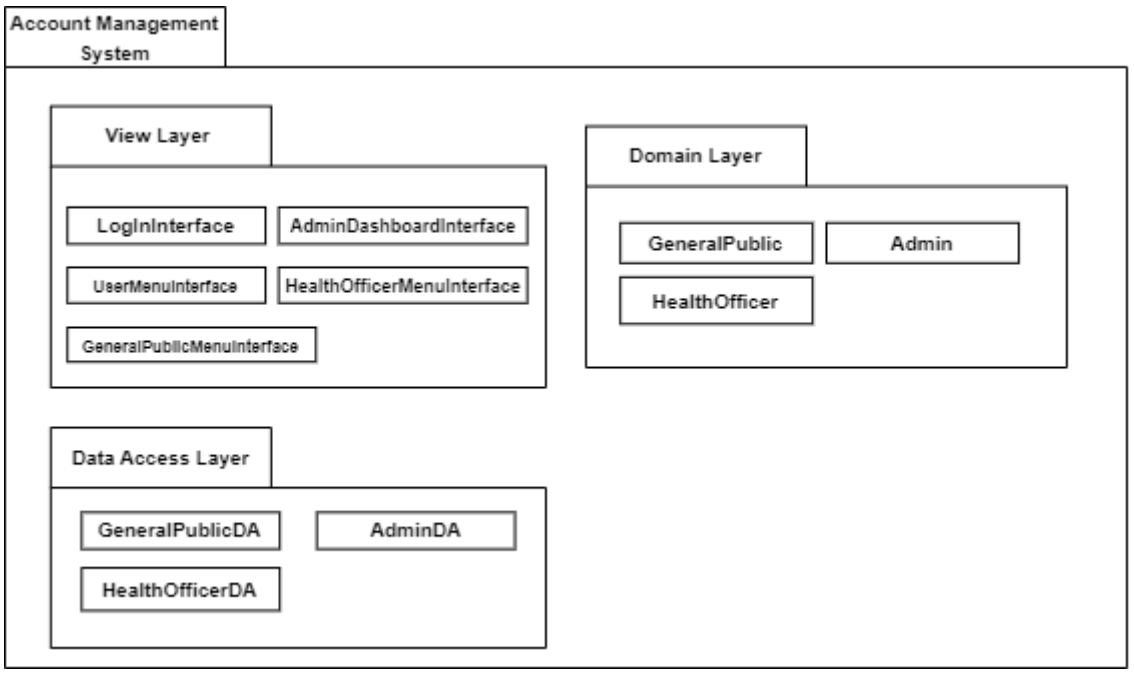


Figure 4.2.1: Package Diagram for <Account Management> Subsystem

4.2.1.1 Class Diagram

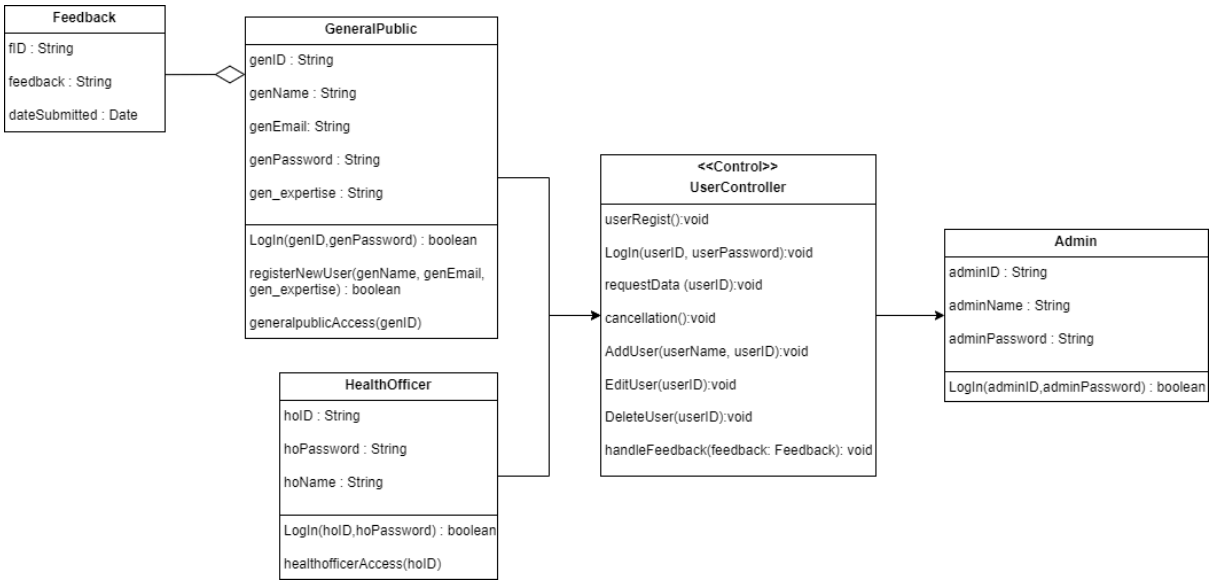


Figure 4.2.1.1: Class Diagram for <Account Management> Subsystem

Entity Name	GeneralPublic
Method Name	registerNewUser
Input	gpName, gpEmail, gp_expertise
Output	gp_ID
Algorithm	1. Start 2. Check if a user with the given email already exists in the database. 3. If the email already exists, return False. 4. Generate a unique gpID for the new user. 5. Create a new GeneralPublic object with the input details: a. Set gpID to the generated ID. b. Set gpName to the input gpName. c. Set gpPassword to the input gpPassword. d. Set gp_Expertise to the input gpExpertise. e. Set email to the input email. 6. Save the new GeneralPublic object to the database. 7. Return True if the registration is successful. 8. End

Entity Name	GeneralPublic
Method Name	Login
Input	gpID, gpPassword
Output	None
Algorithm	1. Start 2. Retrieve the GeneralPublic record from the database using gpID. 3. If no record is found, return False. 4. Compare the retrieved password with gpPassword. 5. If the passwords match, return True. 6. Else, return False. 7. End

Entity Name	GeneralPublic
Method Name	generalpublicAccess
Input	gpID
Output	None
Algorithm	1. Start 2. Retrieve the GeneralPublic record using gpID. 3. If the record exists, return the associated access level. 4. If no record exists, return "No Access". 5. End

Entity Name	HealthOfficer
Method Name	Login
Input	hoID, hoPassword
Output	None
Algorithm	1. Start 2. Retrieve the HealthOfficer record using hoID. 3. If no record is found, return False. 4. Compare the retrieved password with hoPassword. 5. If they match, return True. 6. Else, return False. 7. End

Entity Name	UserController
Method Name	deleteUser
Input	userID
Output	-
Algorithm	1. Start 2. Check if the userID exists in the database. 3. If it exists, delete the user record. 4. Return True if deletion is successful, False otherwise. 5. End

Entity Name	UserController
Method Name	userRegister
Input	userID
Output	-
Algorithm	1. Start 2. Check if the user ID already exists in the database. 3. If it exists, return False. 4. Save the new user record in the database. 5. Return True. 6. End

Entity Name	UserController
Method Name	AddUser
Input	userName, userID
Output	-
Algorithm	1. Start 2. Check if a user with the given userID exists. 3. If it exists, return False. 4. Create a new user object with userName and userID. 5. Save the user object to the database. 6. Return True. 7. End

4.2.1.2 Sequence Diagram

a) SD001: Sequence diagram for User Login

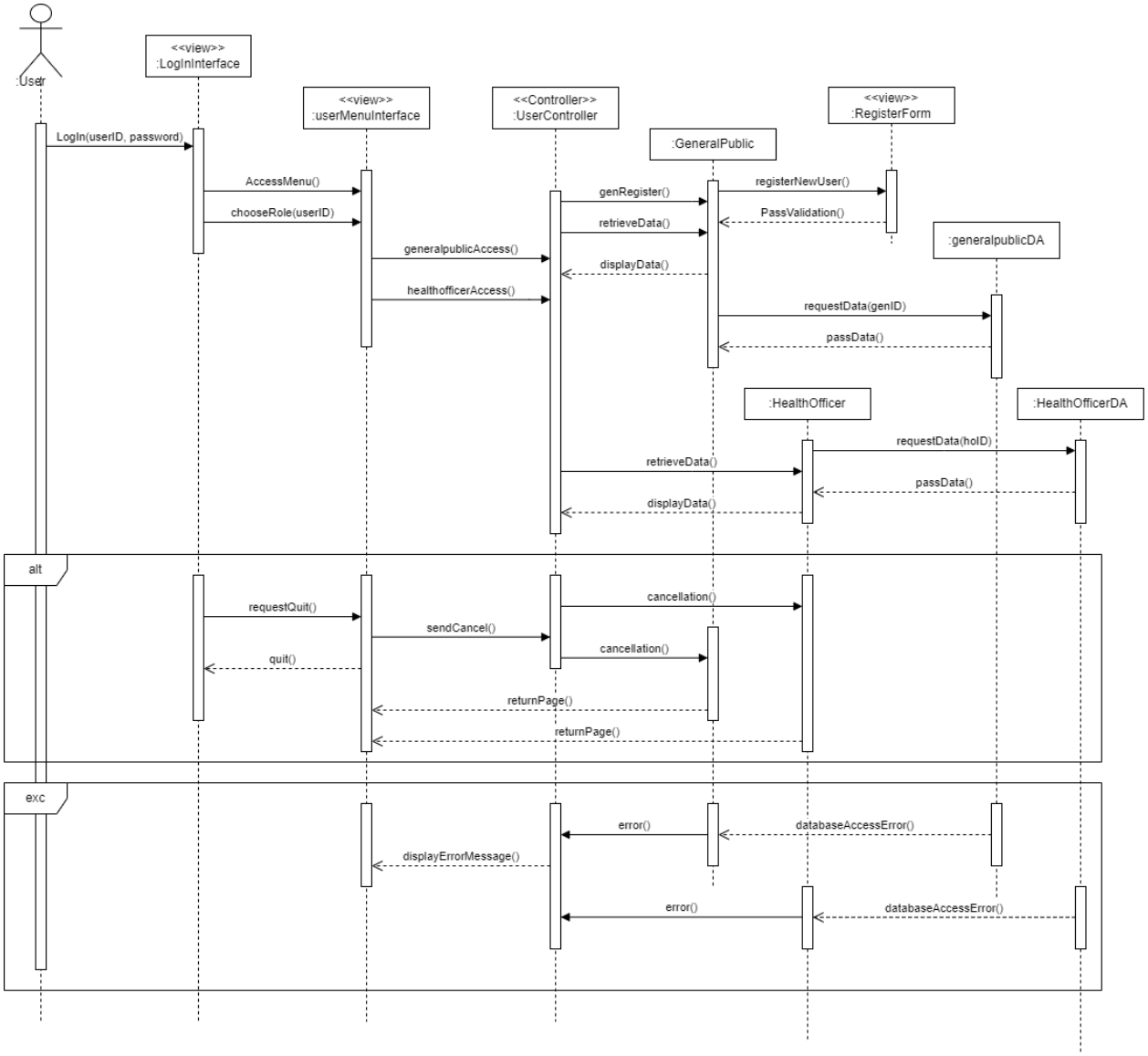


Figure 4.4: Sequence Diagram for < User Login >

b) SD002: Sequence diagram for Admin Manage General Public Data

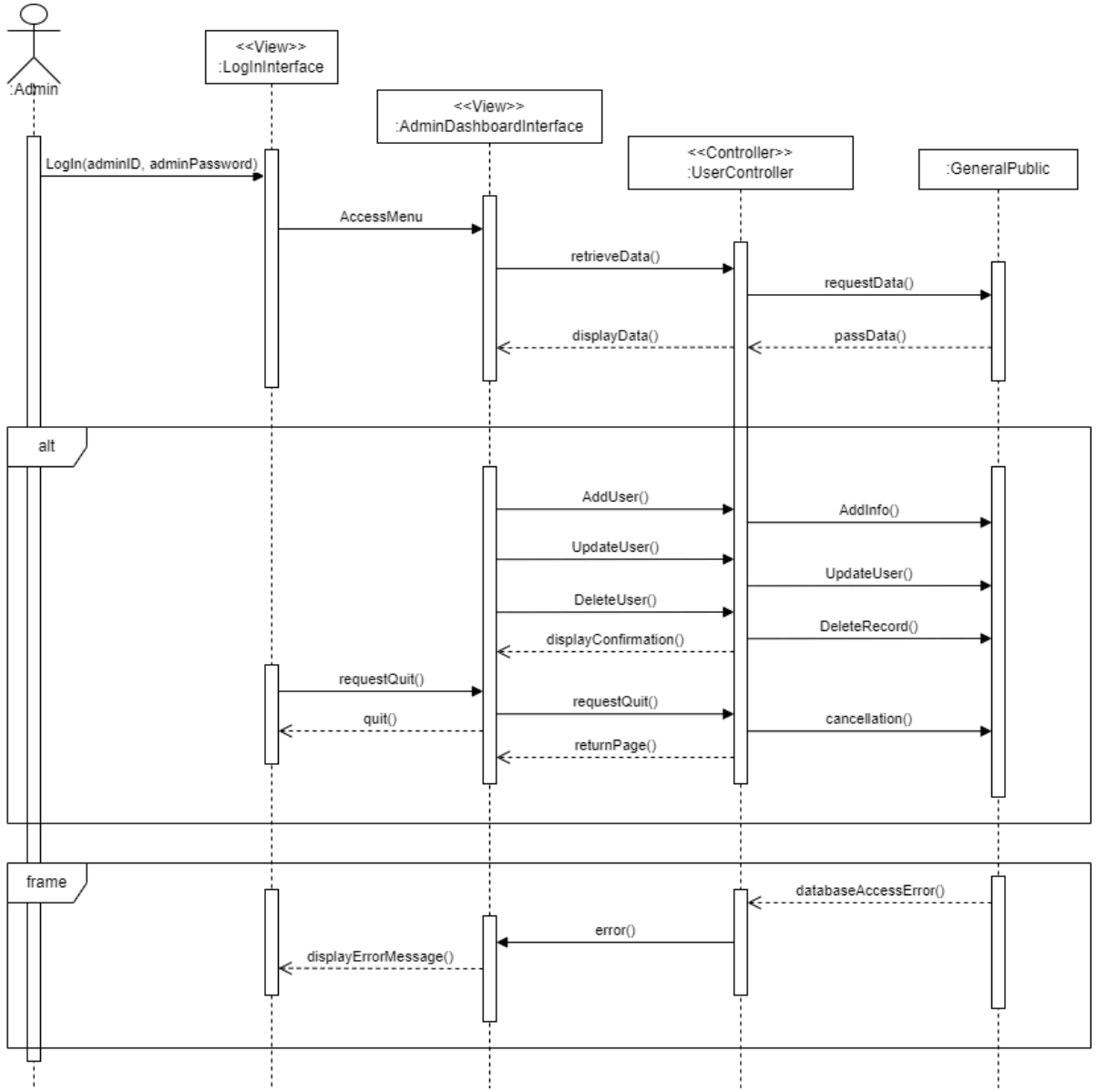


Figure 4.5: Sequence Diagram for <Admin Manage General Public Data>

4.2.2 P002: <Dengue Material> Subsystem

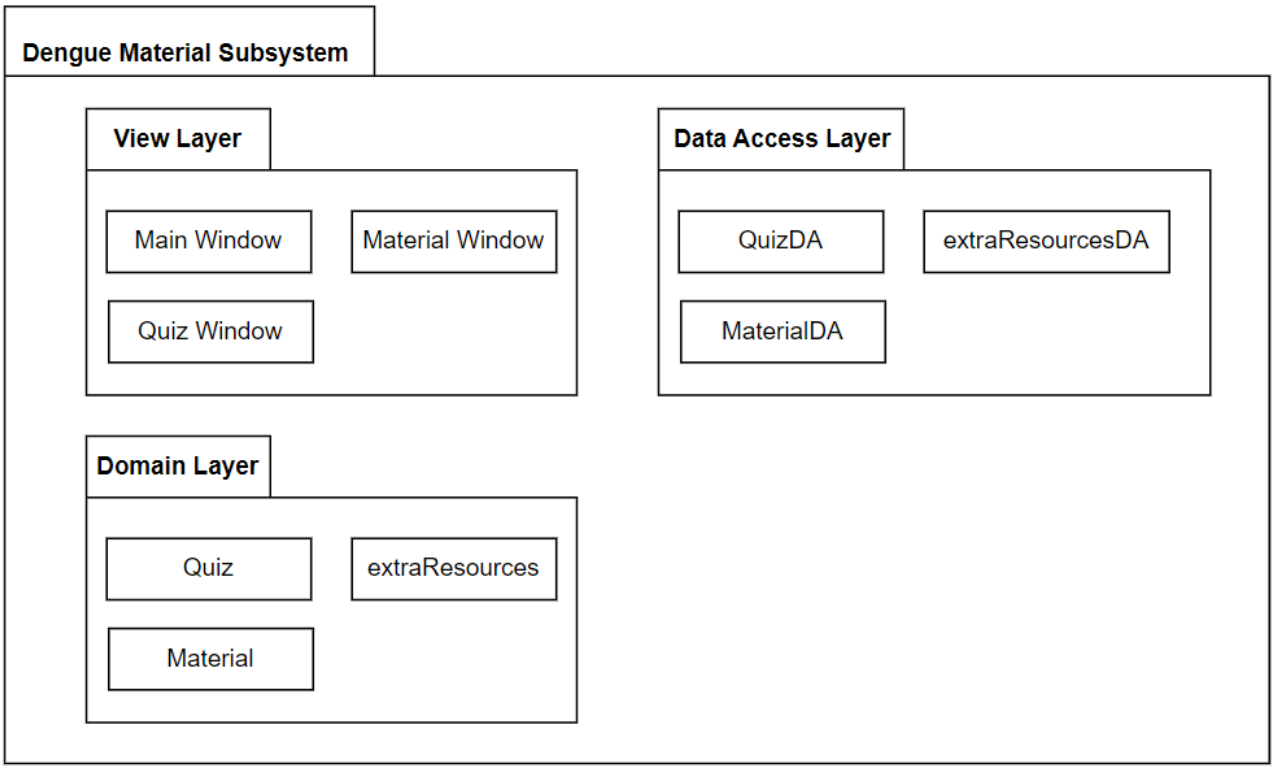


Figure 4.2.2: Package Diagram for <Dengue Material> Subsystem

4.2.2.1 Class Diagram

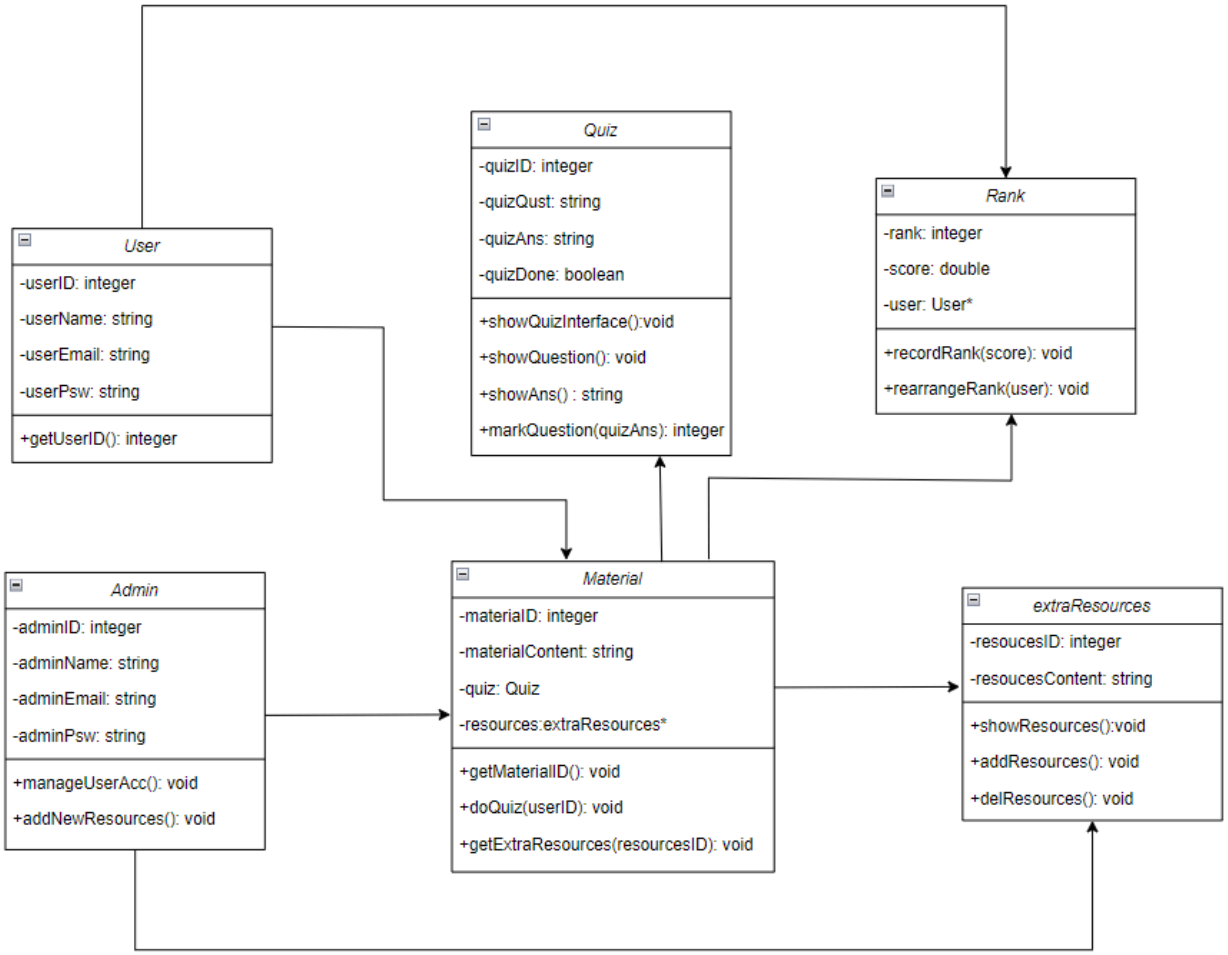


Figure 4.2.2.1: Class Diagram for <Dengue Material> Subsystem

Entity Name	Material
Method Name	getMaterialID():void
Input	None
Output	Return materialID
Algorithm	1. Start 2. Return materialID 3. End

Entity Name	Material
Method Name	doQuiz(userID):void
Input	userID: integer
Output	Display the quiz page for user
Algorithm	1. Start 2. Read userID 3. call quiz.showQuizInterface() 4. End

Entity Name	Material
Method Name	getExtraResources(resourcesID):void
Input	resourcesID(): integer
Output	Display extraResources
Algorithm	1. Start 2. Print resourcesContent 3. End

Entity Name	User
Method Name	getUserID(): integer
Input	None
Output	Return UserID
Algorithm	1. Start 2. Return UserID 3. End

Entity Name	Admin
Method Name	manageUserAcc():void
Input	None
Output	None
Algorithm	1. Start 2. Get User information 3. Update user account 4. End

Entity Name	Admin
Method Name	addNewResources():void
Input	None
Output	None
Algorithm	1. Start 2. Get resourcesID and resourcesContent 3. Update newResources 4. End

Entity Name	Quiz
Method Name	showQuizInterface(): void
Input	None
Output	Display quiz interface
Algorithm	1. Start 2. Show quiz interface 3. End

Entity Name	Quiz
Method Name	showQuestion(): void
Input	None
Output	Display quiz questions
Algorithm	1. Start 2. Show quiz questions 3. End

Entity Name	Quiz
Method Name	showAns(): string
Input	None
Output	Return quiz answers
Algorithm	1. Start 2. return quizAns 3. End

Entity Name	Quiz
Method Name	markQuestion(quizAns): integer
Input	quizAns: string
Output	Display score
Algorithm	1. Start 2. Mark for each question 3. Display overall score 4. End

Entity Name	Rank
Method Name	recordRank(score): void
Input	score: double
Output	None
Algorithm	1. Start 2. Read score for each questions 3. Record overall score 4. End

Entity Name	Rank
Method Name	rearrangeRank(user): void
Input	user:User*
Output	display new rank
Algorithm	1. Start 2. Read each rank for new user 3. Display new rank 4. End

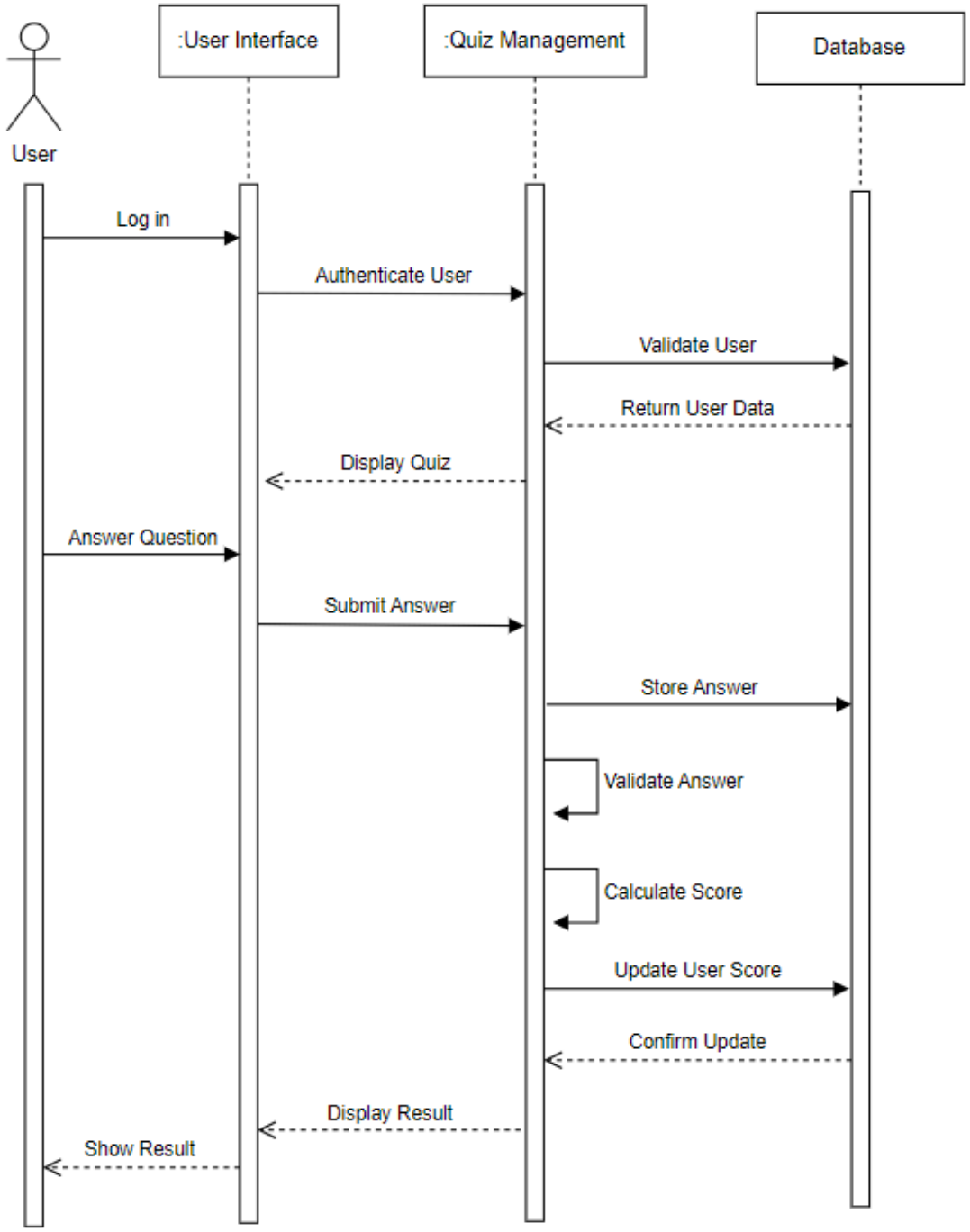
Entity Name	extraResources
Method Name	showResources():void
Input	None
Output	Display resources
Algorithm	1. Start 2. Read resources added 3. Display resourcesContent 4. End

Entity Name	extraResources
Method Name	addResources():void
Input	None
Output	None
Algorithm	1. Start 2. Read resourcesContent 3. Update resourcesContent 4. End

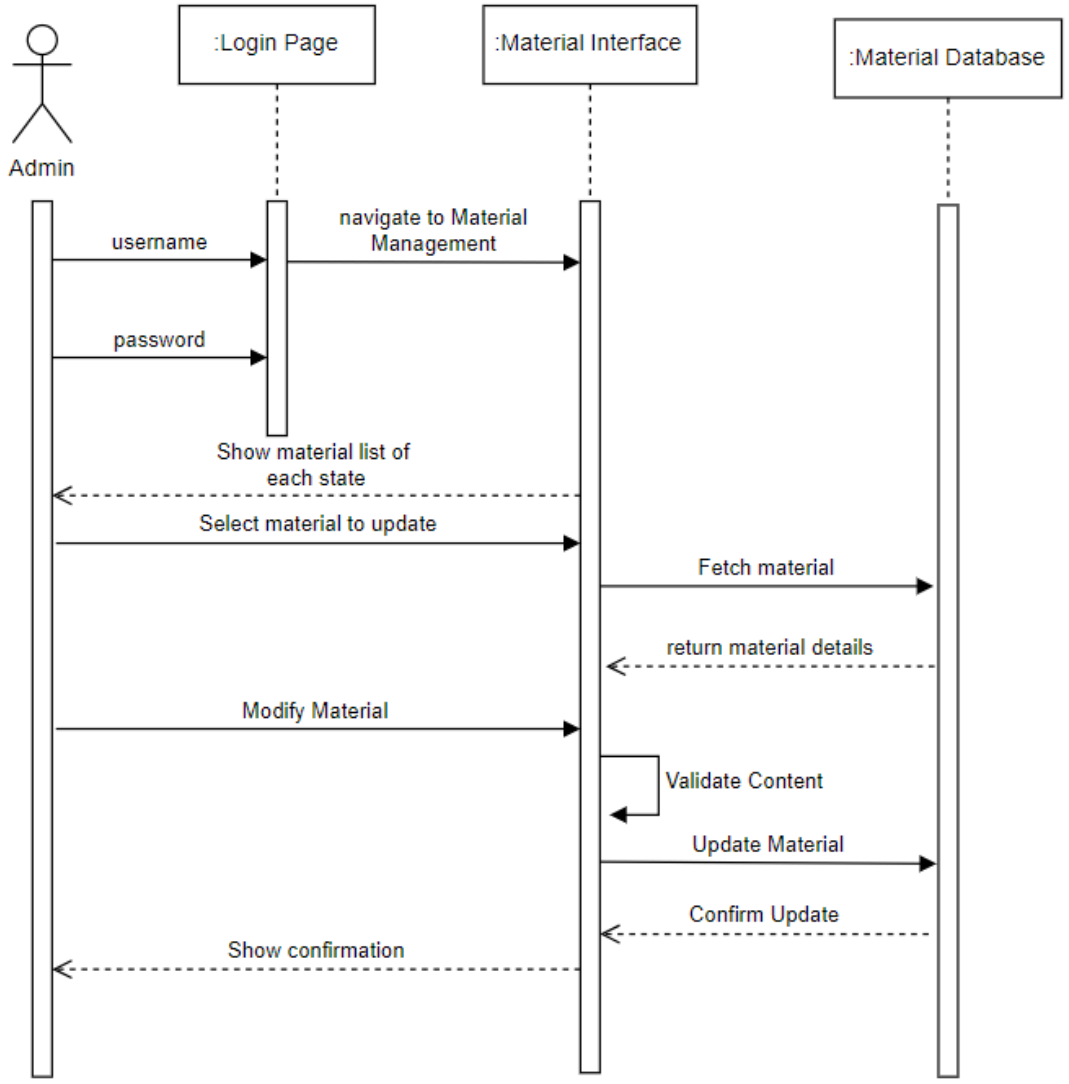
Entity Name	extraResources
Method Name	delResources():void
Input	None
Output	None
Algorithm	1. Start 2. Get resourcesID 3. Delete resourcesContent 4. End

4.2.2.2 Sequence Diagram

a) SD003: Sequence diagram for Quiz



b) SD004: Sequence diagram for Update Material



4.2.3 P003: <Dengue Report> Subsystem

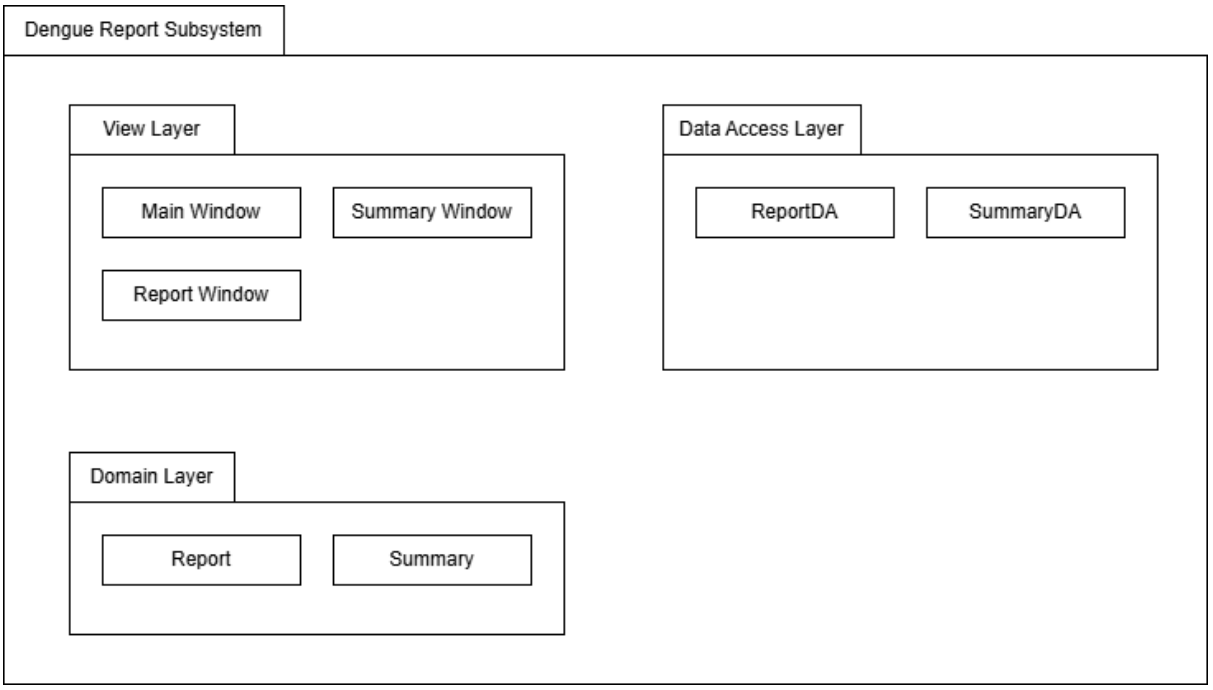


Figure 4.2.3: Package Diagram for <Dengue Report> Subsystem

4.2.3.1 Class Diagram

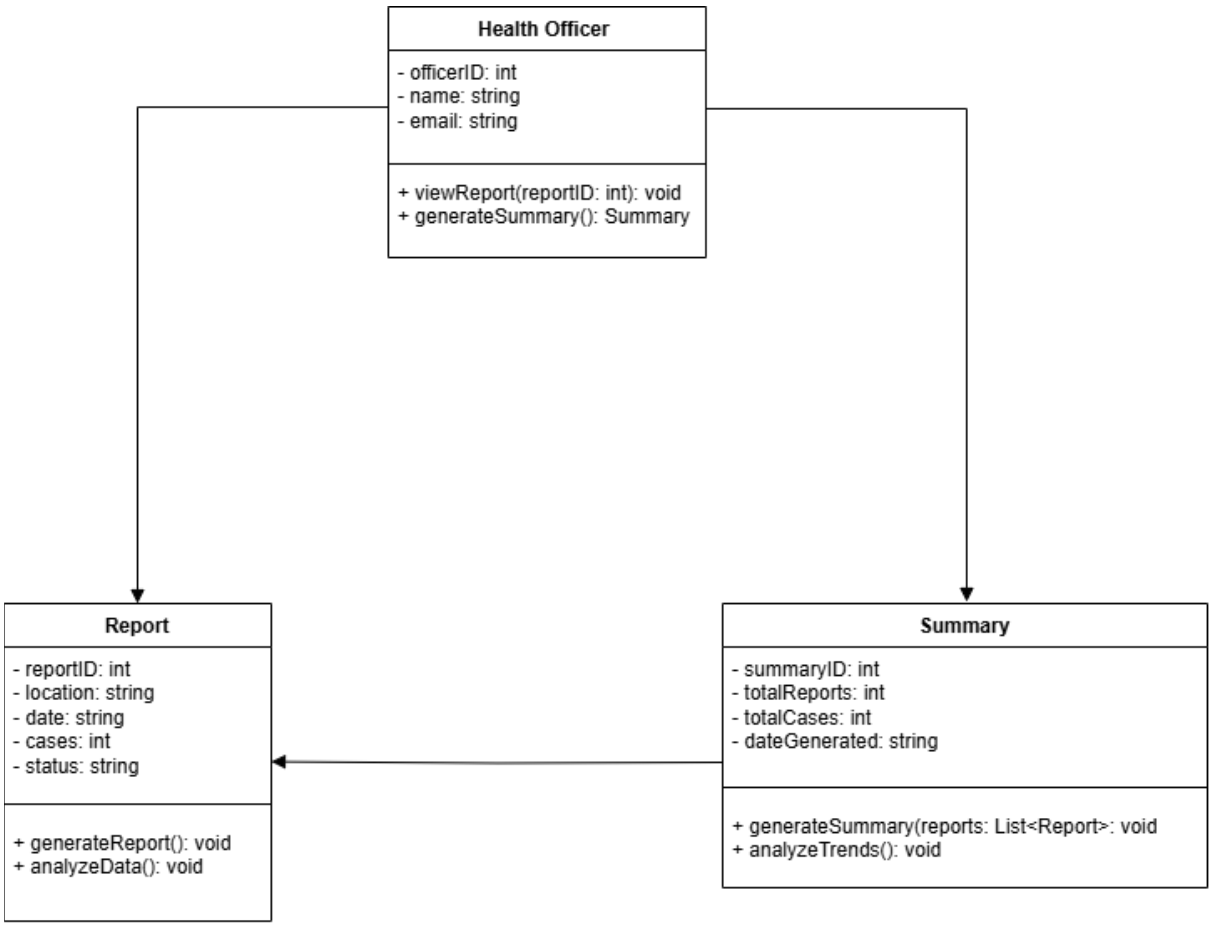


Figure 4.2.3.1: Class Diagram for <Dengue Report> Subsystem

Entity Name	Health Officer
Method Name	viewReport()
Input	reportID: int
Output	void
Algorithm	1. Start 2. Retrieve Report object using reportID 3. Display the report details 4. End

Entity Name	Health Officer
Method Name	generateSummary()
Input	None
Output	Summary
Algorithm	1. Start 2. Collect all Report data 3. Calculate total reports and cases 4. Return Summary object 5. End

Entity Name	Report
Method Name	generateReport()
Input	None
Output	void
Algorithm	1. Start 2. Collect data (location, data, cases, status, etc.) 3. Save data as a new Report 4. End

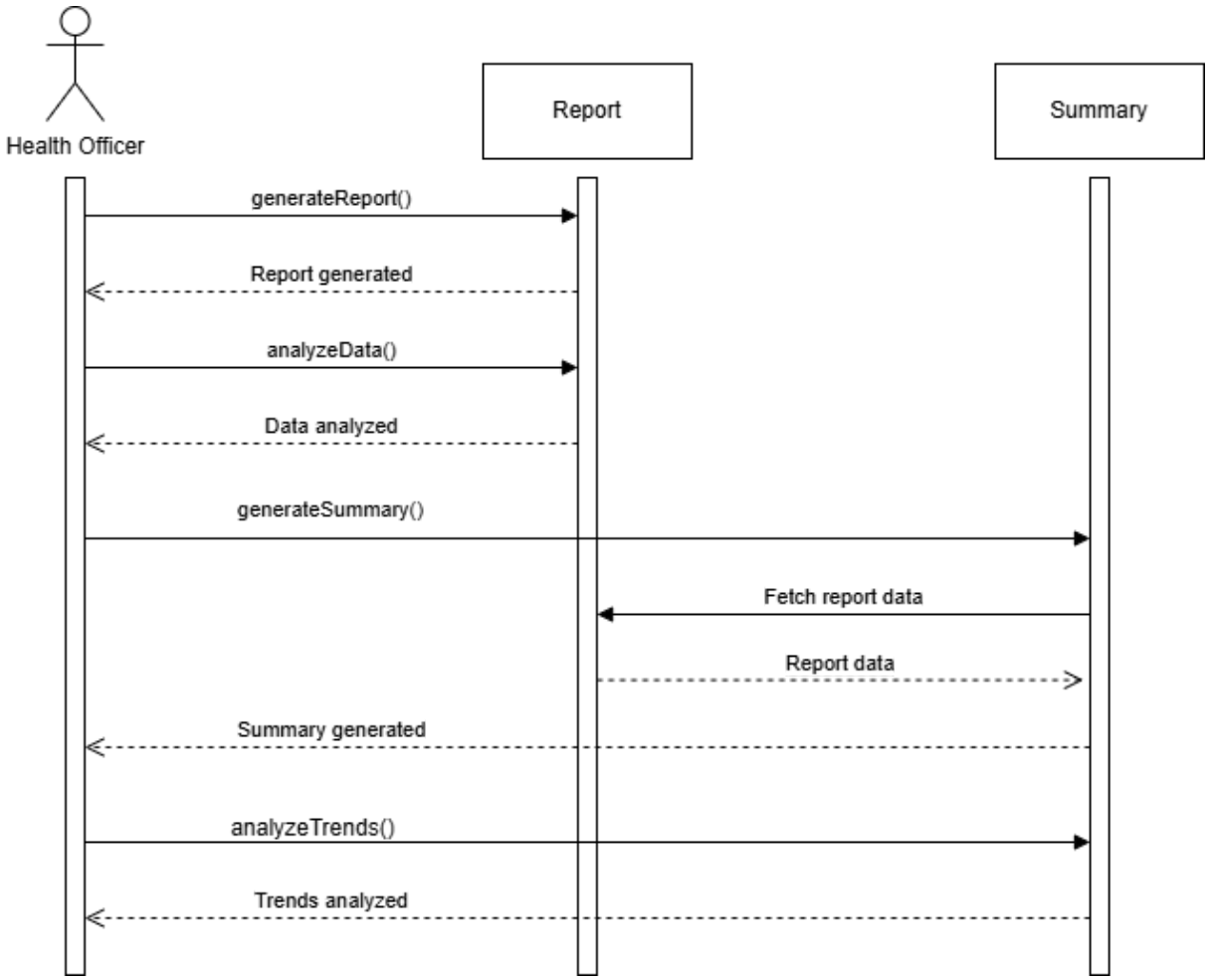
Entity Name	Report
Method Name	analyzeData()
Input	None
Output	void
Algorithm	1. Start 2. Process the report data 3. Identify trends or patterns 4. Display results 5. End

Entity Name	Summary
Method Name	generateSummary()
Input	reports: List<Report>
Output	void
Algorithm	1. Start 2. Aggregate data from reports list 3. Compute total reports and cases 4. Update object attributes 5. End

Entity Name	Summary
Method Name	analyzeTrends()
Input	None
Output	void
Algorithm	1. Start 2. Analyze historical report data 3. Identify and highlight trends 4. Display findings 5. End

4.2.3.2 Sequence Diagram

SD005: Sequence Diagram for Generate report and overall summary



Data Design

5.1 Data Description

The major data or systems entities are stored in a relational database named **Dengue Prevention and Education Database System**, processed and organized into ***9*** entities as listed in Table 5.1.

Table 5.1: Description of Entities in the Database

No.	Entity Name	Description
1.	User	This entity describes all the data regarding the users who have registered in our system. The data includes the user ID, user name, their email, their password, and their expertise. The primary key for this entity is userID.
2.	Admin	This entity records the data regarding the admins who is responsible for managing our system. The data stored in this entity includes the admin ID, admin name, and admin password. The primary key for this entity is adminID.
3.	Health Officer	This entity registers the data regarding the health officers that is tasked to handle the reports and notifications within the system. The data stored in this entity includes health officer ID, health officer name, and the health officer password. The primary key for this entity is hoID.
4.	Material	This entity describes all the data regarding the material that were provided in our system. The data stored in this entity includes the material ID, material content, the quiz, and the extra resources. The primary key is materialID.
5.	Quiz	This entity describes all the data regarding the provided quizzes in our system. The data stored in this entity includes the quiz ID, the quiz questions, the quiz answers, and the quiz's state. The primary key is the quizID.
6.	extraResources	This entity records the data regarding any extra resources that is provided in our system. This entity includes the resources ID, and the resources content. The primary key will be the resourcesID.
7.	Rank	The entity records the data regarding the rank of the user for the quiz that they have answered in the system. This entity includes the rank, the user's score, and the user. The primary key is rank.
8.	Report	This entity records all the data regarding the report the user has filed with our system. This entity includes report ID, the location, the date, the cases, and the current status. The primary key is reportID.

9.	Summary	This entity describes all the data regarding the summary of all the reports that were recorded in our system. This entity includes the summary ID, the total of the reports, the total cases, and data when the report was generated. The primary key is summaryID.

5.2 Data Dictionary

5.2.1 Entity: <Admin>

Attribute Name	Type	Description
adminID {PK}	string	Each unique identity by numbers represents an admin.
adminName	string	The name of the admin when they were registered in the system.
adminPassword	string	A secret code used by the admin for logging into the system

5.2.2 Entity: <extraResources>

Attribute Name	Type	Description
resourcesID {PK}	integer	A unique identity by numbers that represents every extra resource.
resourcesContent	string	Show the content or information of the extra resource.

5.2.3 Entity: <HealthOfficer>

Attribute Name	Type	Description
hoID {PK}	string	Each unique identity by numbers represents a health officer.
hoPassword	string	A secret code used by the health officer to log into the system.
hoName	string	The health officer's name that was registered in the system.

5.2.4 Entity: <Material>

Attribute Name	Type	Description
materialID {PK}	integer	Shows the unique identity by numbers for each material.

materialContent	string	Shows the content or information of the material.

5.2.5 Entity: <Quiz>

Attribute Name	Type	Description
quizID {PK}	integer	Shows the unique identity for each quiz, represented by numbers.
quizQust	string	Displays the question required to be answered for the quiz.
quizAns	string	Displays the answers to the questions when the quiz is done.
quizDone	boolean	Shows the status of the quiz, whether it is done or still in progress.

5.2.6 Entity: <Rank>

Attribute Name	Type	Description
rank {PK}	integer	Shows the ranking of the users for the quiz they has taken.
score	double	Displays the score of the user after they finish the quiz.

5.2.7 Entity: <Report>

Attribute Name	Type	Description
reportID {PK}	integer	A unique identity by numbers that represents the report filed.
location	string	The accurate location of the reported case is by a string of addresses.
date	string	The date when the report was filed, is presented by DD/MONTH/YY.
cases	integer	The number of cases that had happened during the report.
status	string	The current status of the report will show whether it is still happening or resolved.

5.2.8 Entity: <Summary>

Attribute Name	Type	Description
summaryID {PK}	integer	A unique identity by numbers for the summary made from the chosen category.
totalReports	integer	The total number of accumulated reports in the system.
totalCases	integer	The total number of cases recorded in the system.
dateGenerated	string	The date when the summary is generated, is presented by DD/MONTH/YY.

5.2.9 Entity: <User>

Attribute Name	Type	Description
userID {PK}	string	Each user has a unique identity which is represented by a combination of numbers and alphabets.
userName	string	The name of the user when they register in the system
userEmail	string	The email used by the user
userPassword	string	A secret code used by the user for logging into the system
userExpertise	string	A suitable expertise that represents the user best.

User Interface Design

6.1 Overview of User Interface

The interface will include the login page, the main page, the material page, and much more. Firstly, the user is required to register an account beforehand in order to login to the system. After the registration, the user will enter their username and password to login the system. The user will land on the main page, where they can navigate through the application and use the functions. For the user, they can check the materials on dengue, do some quizzes, and check or file reports. The system will guide the user through the pages, and a few actions will trigger certain feedback, for example when a quiz is completed. Thus, the user will be able to use the system and try out all the awaited functions.

6.2 Screen Images

In the report, only the crucial interfaces will be shown, since all of the interfaces will take up the unnecessary space. The Google Drive link given will include all of the designed interfaces: https://drive.google.com/drive/folders/1XW5tWEnzB7wi1OZgr6BxSjDgkr3rBv72?usp=sharing

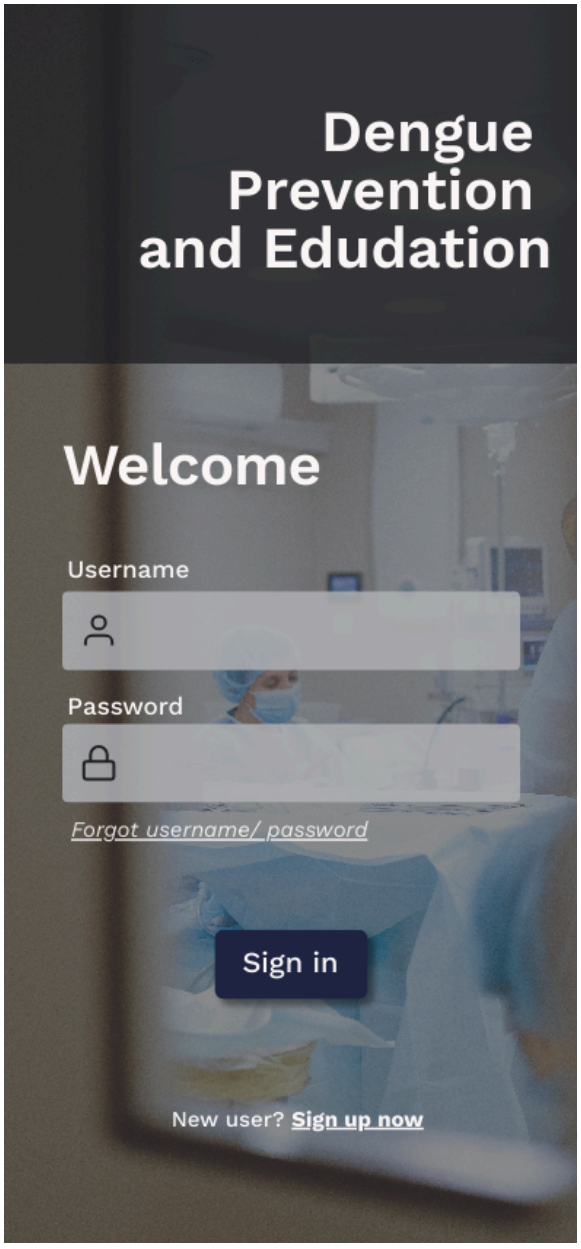


Figure 6.1: Interface for <Login>

Figure 6.2: Interface for <Create Account>

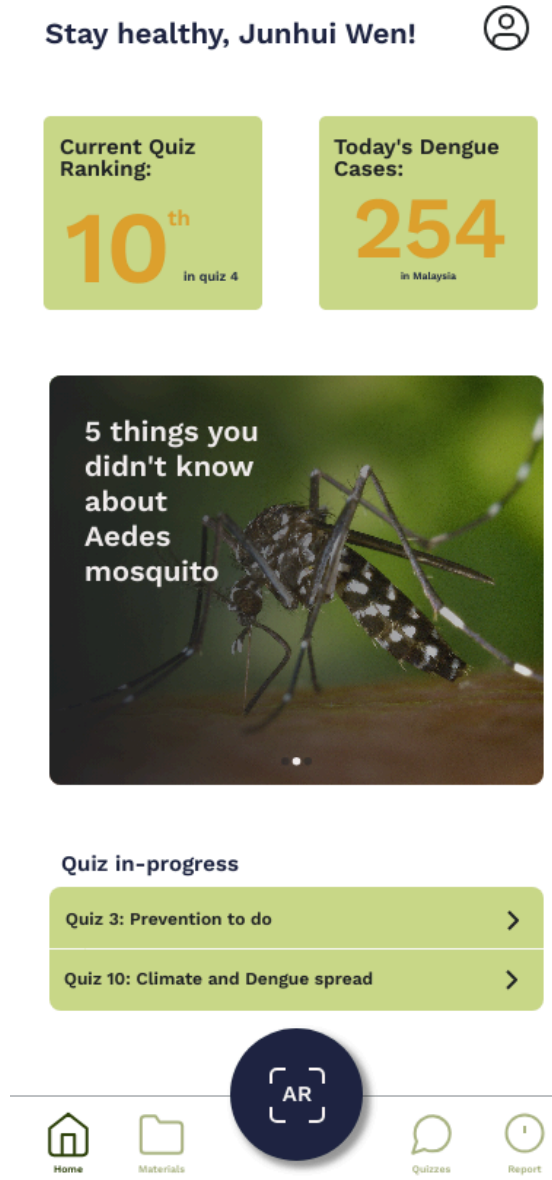


Figure 6.3: Interface for <Main Page>

Figure 6.4: Interface for <Materials>

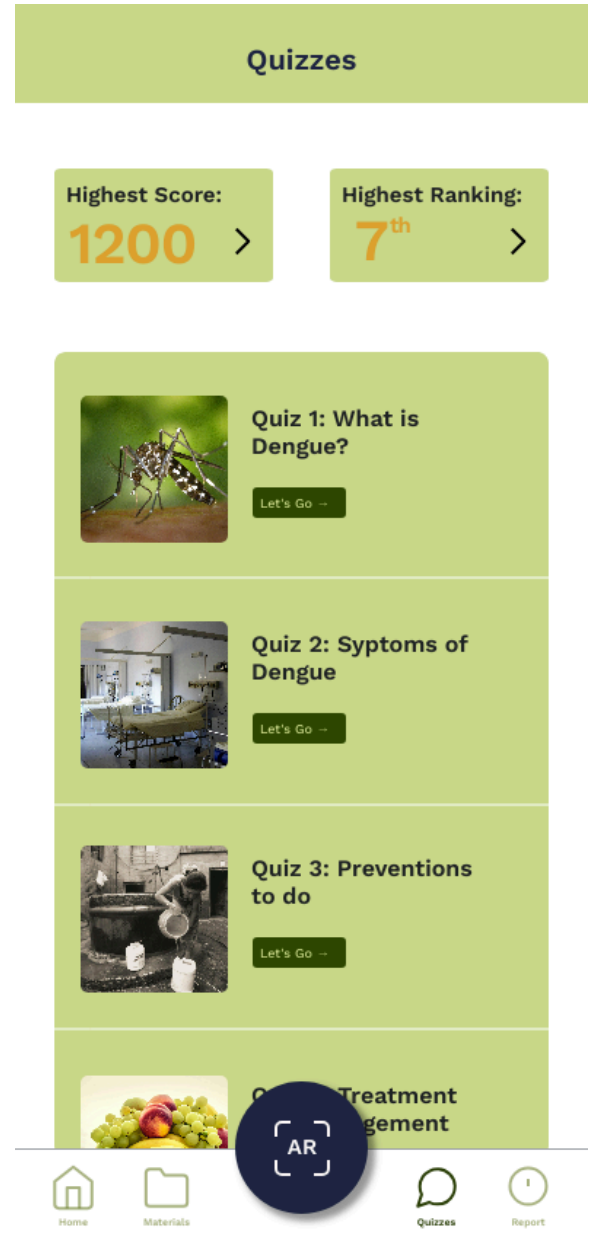


Figure 6.5: Interface for <Quizzes>

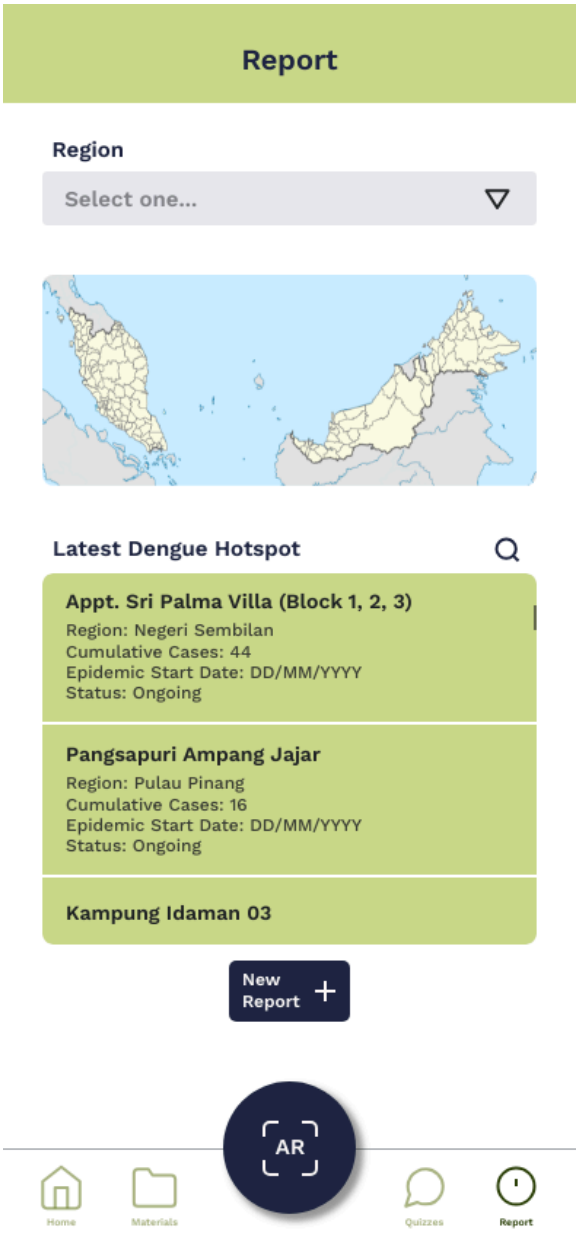


Figure 6.6: Interface for <Report>

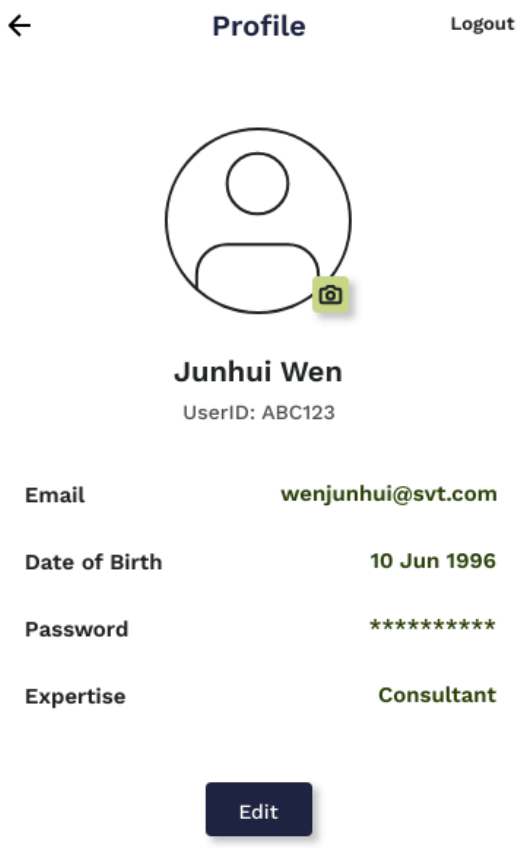


Figure 6.7: Interface for <Profile>

7 Requirements Matrix

	User	Admin	Health Officer	Material	Quiz	extraResources	Rank	Report	Summary
P001, UC001, SD001	X								
P001, UC002, SD002	X							X	
P002, UC004, SD003	X				X		X		
P002, UC005, SD004		X		X					
P003, UC007, SD005			X					X	X

Table 7.1: Description of Entities in the Database

8 Test Cases, Data and Expected Results

8.1 Test TC001 for Module <Account Management Subsystem>: <Manage Account (UC001)>

This test contains the following test cases:

(a) TC001_01: Login account with valid inputs

(b) TC001_01: Reset Password

8.1.1 TC001_01: Login account with valid inputs

Test Case ID:	TC001_01	Test Case Description:	Test the login functionality
Created by:	Ngeow Zhi Yu	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	Access to Internet		1.	username= zhiyu123
2.	Registered User Account		2.	password= 123xyz

Test Conditions	
1. Verify user are able to login to Dengue Prevention and Education System	

Step#	Step Details	Expected Result
1.	Navigate to Login page of Dengue Prevention and Education System	Show the login page
2.	Input username and password	Correctly enter username and password
3.	Click Sign in	User successfully enter the system and show the main menu

8.1.2 TC001_02: Reset Password

Test Case ID:	TC001_02	Test Case Description:	Reset account password
Created by:	Ngeow Zhi Yu	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	Access to Internet		1.	username= zhiyu123
2.	Registered User Account		2.	password= 123xyz
			3.	email= ngeowzhiyu@graduate.utm.my

Test Conditions	
1. Verify username, initial password, email are correct before resetting password	

Step#	Step Details	Expected Result
1.	Navigate to Profile page	Show the user profile
2.	Click on forget password	Show interface to reset password
3.	Enter email, username and initial password for authentication and click enter	All entered data matches the stored data.
4.	Enter new password	The new password meets the prerequisites and successfully change the password

8.2 Test TC002 for Module <Dengue Material Subsystem>: <Check Dengue Risk (UC002)>

This test contains the following test cases:

(a) TC002_01: Check dengue risk with specific location
(b) TC002_02: Check dengue risk with invalid location

8.2.1 TC002_01: Check dengue risk with specific location

Test Case ID:	TC002_01	Test Case Description:	Check dengue risk level with specific location
Created by:	Ngeow Zhi Yu	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	Access to Internet		1.	state= Johor
2.	Log in to the system		2.	town= Skudai

Test Conditions	
1. Verify on checking dengue risk level for a specific location	

Step#	Step Details	Expected Result
1.	Navigate to Dengue Risk Results Page	Show the Dengue Risk Results Page
2.	Search for the state and that town	Correctly enter state and town
3.	Click search	Shown the dengue risk level for that specific location

8.2.2 TC002_02: Check dengue risk with invalid location

Test Case ID:	TC002_01	Test Case Description:	Check dengue risk level with invalid location
Created by:	Ngeow Zhi Yu	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	Access to Internet		1.	state= Penang
2.	Log in to the system		2.	town= Skudai

Test Conditions	
1. Unable to verify on checking dengue risk level for a invalid location	

Step#	Step Details	Expected Result
1.	Navigate to Dengue Risk Results Page	Show the Dengue Risk Results Page
2.	Search for the state and that town	Enter state and town incorrectly
3.	Click search	Show invalid location: state and town does not match

8.3 Test TC003 for Module <Dengue Report Subsystem> : <Report Breeding Site (UC003)>

This test contains the following test cases:

(a) TC003_01: Report Breeding Site with all required information
(b) TC003_02: Report Breeding Site with missing information

8.3.1 TC003_01: Report Breeding Site with all required information

Test Case ID:	TC003_01	Test Case Description:	Report Breeding Site with all required information
Created by:	Lee Jian Ai	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	Access to Internet		1.	location = Near Jalan AAA, Taman BBB, Skudai, Johor
2.	Log in to the system		2.	description = Accumulated water in an abandoned tire
			3.	photo = YES

Test Conditions	

1. Able to report breeding site with given information

Step#	Step Details	Expected Result
1.	Navigate to Dengue Risk page, click Report Problems button	Show the Report Breeding Site page
2.	Fill in all the required information (description, photo, and location)	Correctly enter the description and location, attach the evidence photo
3.	Click submit	Show that the report is submitted successfully

8.3.2 TC003_02: Report Breeding Site with missing information

Test Case ID:	TC003_02	Test Case Description:	Report Breeding Site with missing information
Created by:	Lee Jian Ai	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	Access to Internet		1.	location = Near Jalan AAA, Taman BBB, Skudai, Johor
2.	Log in to the system		2.	description =
			3.	photo = NO

Test Conditions	
1. Unable to report breeding site with missing information	

Step#	Step Details	Expected Result
1.	Navigate to Dengue Risk page, click Report Problems button	Show the Report Breeding Site page
2.	Fill in all the required information (description, photo, and location)	Correctly enter the location, didn't fill in the description and didn't attach an evidence photo
3.	Click submit	Show "Error: Missing Information", report cannot be submitted

8.4 Test TC004 for Module <Dengue Material Subsystem>: <Complete Quiz (UC004)>

This test contains the following test cases:

(a) TC004_01: Answer the quiz until the end
(b) TC004_02: Answer the quiz and quit halfway

8.4.1 TC004_01: Answer the quiz until the end

Test Case ID:	TC004_01	Test Case Description:	Answer the quiz until the end	
Created by:	Lee Jian Ai	Version:	1.0	
No.	Prerequisites		No.	Test Data
1.	Access to Internet		1.	quizDone = YES
2.	Log in to the system			
Test Conditions				
1. Answer the quiz and complete it				
Step#	Step Details	Expected Result		
1.	Navigate to Quizzes page, select Quiz 1	Show the Quiz 1 page		
2.	Answer all the questions, one by one	Answer all questions, none of them skipped		
3.	Click done	Show that the quiz is done, show all correct answers		

8.4.2 TC004_02: Answer the quiz and quit halfway

Test Case ID:	TC004_02	Test Case Description:	Answer the quiz and quit halfway
Created by:	Lee Jian Ai	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	Access to Internet		1.	quizDone = NO
2.	Log in to the system			

Test Conditions	
1. Answer the quiz but do not complete it, exit halfway	

Step#	Step Details	Expected Result
1.	Navigate to Quizzes page, select Quiz 1	Show the Quiz 1 page
2.	Answer all the questions, one by one	Answer the questions and stopped
3.	Click back	Show that “The quiz is incomplete, do you want to save your progress?”, and proceed to quit the quiz

8.5 Test TC005 for Module <Account Management Subsystem>: <Manage User Account (UC005)>

This test contains the following test cases:

(a) TC005_01: Test <Edit General Public Data (SD001)>
(b) TC005_02: Test <Delete General Public Data (SD001)>

8.5.1 TC005_01: Test <Edit General Public Data (SD001)>

Test Case ID:	TC005_01	Test Case Description:	Edit General Public Data
Created by:	Jason Joel	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	Admin must have access to the Account Management Subsystem.		1.	admin_id = admin_user
2.	User account records must exist in the system.		2.	admin_password = admin_password123
3.			3.	gpID = user123
4.			4.	old_gpEmail = old_email@example.com
5.			5.	new_gpEmail = new_email@example.com

Test Conditions	

1. Admin privileges are active.
2. Account Management Subsystem is operational.

Step#	Step Details	Expected Result
1.	Log in to the Account Management Subsystem	Admin dashboard loads successfully.
2.	Select a user account to edit.	User account details are displayed.
3.	Modify the required fields with valid inputs	Fields update with the new data.
4.	Save changes.	Confirmation message appears.

8.5.2 TC005_02: Test <Delete General Public Data (SD001)>

Test Case ID:	TC005_02	Test Case Description:	Delete General Public Data
Created by:	Jason Joel	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	Admin must have access to the Account Management Subsystem.		1.	admin_id = admin_user
2.	User account records must exist in the system.		2.	admin_password = admin_password123
3.			3.	gpID = user456

Test Conditions	

2. Admin privileges are active.
3. Account Management Subsystem is operational.

Step#	Step Details	Expected Result
1.	Log in to the Account Management Subsystem.	Admin dashboard loads successfully.
2.	Select a user account to delete.	User account details are displayed.
3.	Confirm the deletion action.	Confirmation prompt appears.
4.	Confirm the deletion prompt.	User account is removed successfully.

8.6 Test TC006 for Module <Dengue Report Subsystem>: <Backup Data (UC006)>

This test contains the following test cases:

(a) TC006_01: Test <Create Data Backup (SD002)>

8.6.2 TC006_01: Test <Create Data Backup (SD000)>

Test Case ID:	TC006_01	Test Case Description:	Create Data Backup
Created by:	Jason Joel	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	The Dengue Report Subsystem must be accessible.		1.	admin_id = admin_user
2.	Admin privileges must be active.		2.	admin_password = admin_password123
3.	Data to be backed up must exist in the database.		3.	path = /backup/dengue_database

Test Conditions	

1. Backup functionality is operational.
2. Adequate storage space is available in the backup destination.

Step#	Step Details	Expected Result
1.	Log in to the Dengue Report Subsystem.	Admin dashboard loads successfully.
2.	Navigate to the Backup Data feature.	Backup Data interface is displayed.
3.	Specify the database backup destination path.	Path is accepted without errors.
4.	Initiate the database backup process.	Backup process starts successfully.
5.	Verify the database backup is completed successfully.	Confirmation message appears, and the database is saved at the specified path

8.7 Test TC007 for Module <Dengue Report Subsystem>: <Generate Report (UC007)>

This test contains the following test cases:

(a) TC007_01: Generate report with valid parameters
(b) TC007_02: Generate report with invalid parameters

8.7.1 TC007_01: Generate report with valid parameters

Test Case ID:	TC007_01	Test Case Description:	Generate report if there are valid parameters
Created by:	A Muzhaffar P.	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	The health officer is logged into the system.		1.	Time range: January 2025
2.	Dengue case data is available in the database.		2.	Location: Johor

Test Conditions	
1. Verify that the health officer can generate a report when valid parameters are provided.	

Step#	Step Details	Expected Result
1.	Navigate to the "Generate Report" section.	The "Generate Report" interface is displayed.
2.	Select time range and location parameters.	Parameters are accepted without errors.
3.	Confirm report generation.	The system generates the report and provides a download link.
4.	Download the generated report.	The report is successfully downloaded in the selected format.

8.7.2 TC007_02: Generate report with invalid parameters

Test Case ID:	TC007_02	Test Case Description:	Generate report if there are invalid parameters
Created by:	A Muzhaffar P.	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	The health officer is logged into the system.		1.	Time range: March 2027 (no data available)
2.	Dengue case data is available in the database.		2.	Location: Invalid location name

Test Conditions	

1. Verify that the system handles invalid parameters correctly.

Step#	Step Details	Expected Result
1.	Navigate to the "Generate Report" section.	The "Generate Report" interface is displayed.
2.	Select an invalid time range or location.	The system displays an error message indicating insufficient data.
3.	Retry with different parameters or cancel.	The system allows the health officer to modify parameters or cancel.

8.8 TC008 for Module <Dengue Material Subsystem>: <Send Alerts (UC008)>

This test contains the following test cases:

(a) TC008_01: Send alerts to users in high-risk areas
(b) TC008_02: Handle no users in selected high-risk areas

8.8.1 TC008_01: Send alerts to users in high-risk areas

Test Case ID:	TC008_01	Test Case Description:	Send alerts if there are users in high-risk areas
Created by:	A Muzhaffar P.	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	The health officer is logged into the system.		1.	High-risk areas: Kuala Lumpur, Selangor
2.	High-risk area data is available in the database.		2.	Alert message: "High dengue risk!"

Test Conditions	

1. Verify that the system sends alerts successfully to users in high-risk areas.

Step#	Step Details	Expected Result
1.	Navigate to the "Send Alerts" section.	The "Send Alerts" interface is displayed.
2.	Select high-risk areas and compose alert messages.	The system accepts the selected areas and alerts messages without error.
3.	Confirm and send alerts.	Alerts are sent to all users in the selected areas.
4.	Verify alert delivery.	Users receive alerts via email, SMS, or app notification.

8.8.2 TC008_02: Handle no users in selected high-risk areas

Test Case ID:	TC008_02	Test Case Description:	Handle alerts if there are no users in high-risk areas
Created by:	A Muzhaffar P.	Version:	1.0

No.	Prerequisites		No.	Test Data
1.	The health officer is logged into the system.		1.	High-risk area: Area with no users
2.	High-risk area data is available in the database.		2.	Alert message: "Stay safe, dengue cases rising!"

Test Conditions	

1. Verify that the system handles cases where no users are found in the selected high-risk areas.

Step#	Step Details	Expected Result
1.	Navigate to the "Send Alerts" section.	The "Send Alerts" interface is displayed.
2.	Select a high-risk area with no registered users.	The system notifies the health officer about the absence of users.
3.	Retry with different areas or cancel.	The system allows modification or cancellation of the alert process.

Appendix A: Traceability Matrix

Test Case ID	Use Case ID	Package ID
TC001 for <Account Management> Subsystem ● TC001_01 ● TC001_02	UC001	P001
TC002 for <Dengue Material> Subsystem ● TC002_01 ● TC002_02	UC002	P002
TC003 for <Dengue Report> Subsystem ● TC003_01 ● TC003_02	UC003	P003
TC004 for <Dengue Material> Subsystem ● TC004_01 ● TC004_02	UC004	P002
TC005 for <Account Management> Subsystem ● TC005_01 ● TC005_02	UC005	P001
TC006 for <Dengue Report> Subsystem ● TC006_01	UC006	P003
TC007 for <Dengue Report> Subsystem ● TC007_01 ● TC007_02	UC007	P003
TC008 for <Dengue Material> Subsystem ● TC008_01 ● TC008_02	UC008	P002