# MALAYSIA-JAPAN INTERNATIONAL INSTITUTE OF TECHNOLOGY (MJIIT)

SEMESTER 1-2024/25

## SOFTWARE ENGINEERING

COURSE CODE: SECJ 2203; SECTION – 16

**LECTURER: DR ROZANA ISMAIL**

## A3: Software Design Document (SDD)

## Group - 6

| GROUP MEMBERS | MATRIC NUMBER |
|---|---|
| TAHIYA NUZHATH KHAN (LEADER) | A23MJ4025 |
| PASHMIA BINTE WALID (TESTER) | A23MJ4023 |
| SAUMIK HASAN ARPON (DESIGNER) | A23MJ3009 |
| ZABIR MUHAMMAD SALEH (DEVELOPER) | A22MJ9161 |

# Contents

# 3.0 System Architectural Design

## 3.1 Architecture Style and Rationale

The Layered Architecture is ideal for the MyGreen UTM App, as it effectively meets the requirements of modularity, scalability, and maintainability. By dividing the system into distinct layers, this architecture ensures clear separation of responsibilities, simplifies development, and supports cross-platform deployment. It enhances flexibility by allowing independent evolution of each layer while maintaining robustness and reliability for future enhancements.

- **Presentation Layer**: Focuses on user interaction, providing an intuitive interface for managing sustainability initiatives and rewards.
- **Business Logic Layer**: Handles the core functionality, including resource tracking, point allocation, and report generation.
- **Data Access Layer**: Manages secure and efficient communication with the database for storing and retrieving data.

**1. Presentation Layer**

**Responsibilities:**

- Manages all user interactions with the system.
- Displays processed data in a user-friendly format, such as leaderboards and dashboards.
- Collects user inputs like QR scans or activity submissions and forwards them to the Business Logic Layer.
- Ensures compatibility across platforms, including web and mobile devices.

**Implementation in MyGreen UTM:**

- Interfaces for login, activity tracking, and green reward redemption.
- Leaderboards to display rankings and user achievements.
- Dashboards for analytics and reports on sustainability metrics.
- Developed using technologies like React, Flutter, or HTML/CSS for cross-platform compatibility.

**2. Business Logic Layer**

**Responsibilities:**

- Processes and enforces the system's rules and core functionality.
- Acts as a mediator between the Presentation Layer and the Data Access Layer.

- Handles authentication, activity validation, and sustainability calculations.

**Implementation in MyGreen UTM:**

- **Activity Tracking**: Validates QR scans and logs user participation in initiatives.
- **Gamification Engine**: Allocates points and awards badges based on predefined rules.
- **Sustainability Analysis**: Processes data for resource usage, aggregating energy, water, and waste metrics.
- **Notification System**: Sends reminders and updates about initiatives, leaderboards, and rewards.
- Developed using programming languages like Python, Java, or Node.js.

**3. Data Access Layer**

**Responsibilities:**

- Ensures secure and efficient data storage, retrieval, and updates.
- Manages database interactions and ensures consistency and integrity.
- Facilitates communication with external systems for sustainability integrations.

**Implementation in MyGreen UTM:**

- **User Database**: Stores profiles, participation logs, and reward histories.
- **Resource Data Repository**: Maintains data on energy, water, and waste usage.
- **Greenleaf API Integration**: Synchronizes sustainability achievements with external platforms.
- Built using technologies like MySQL, MongoDB, or PostgreSQL.

## Subsystems and Their Collaborations

- ❖ **User Management Subsystem**

**Responsibilities**: Manages user authentication and profile information.
**Collaborations**:

- Provides user information to the Sustainability Initiative Subsystem for assigning tasks.
- Enables the Green Rewards Subsystem to track user participation and allocate rewards.

- ❖ **Sustainability Initiative Subsystem**

**Responsibilities**: Handles sustainability initiatives and tracks user participation.
**Collaborations**:

- Receives user assignments from the User Management Subsystem.
- Sends participation data to the Green Rewards Subsystem for point allocation.

- Shares insights with the Reporting and Analytics Subsystem to monitor initiative progress.

❖ **Resource Monitoring Subsystem**

**Responsibilities**: Tracks energy, water, and waste usage metrics.
**Collaborations**:

- Supplies resource data to the Reporting and Analytics Subsystem for generating insights.
- Contributes monitored data to the Business Logic Layer for sustainability analysis.

❖ **Reporting and Analytics Subsystem**

**Responsibilities**: Generates detailed reports and dashboards with actionable insights.
**Collaborations**:

- Aggregates data from the Resource Monitoring Subsystem for energy, water, and waste reports.
- Supplies analytics to the Sustainability Initiative Subsystem to evaluate initiative effectiveness.

❖ **Green Rewards Subsystem**

**Responsibilities**: Allocates points based on participation and facilitates reward redemption.
**Collaborations**:

- Receives participation data from the Sustainability Initiative Subsystem.
- Updates reward information to the User Management Subsystem for display on user profiles.

## 3.2 Component Model

The component diagram for MyGreen UTM visually represents the system's modular design, highlighting the five main subsystems: User Management, Sustainability Initiative, Resource Monitoring, Reporting and Analytics, and Green Rewards. Each subsystem contains components that encapsulate specific functionalities, such as user authentication, managing sustainability initiatives, monitoring resources like energy, water, and waste, generating reports, and rewarding green behaviors. The diagram demonstrates the flow of data and control between components, such as resource monitoring feeding data into the report generation process and tracking participation linking to the rewards system. This design ensures scalability, maintainability, and seamless integration between the components to achieve UTM’s sustainability goals. It provides a clear view of how each component interacts to support the overall system functionality.

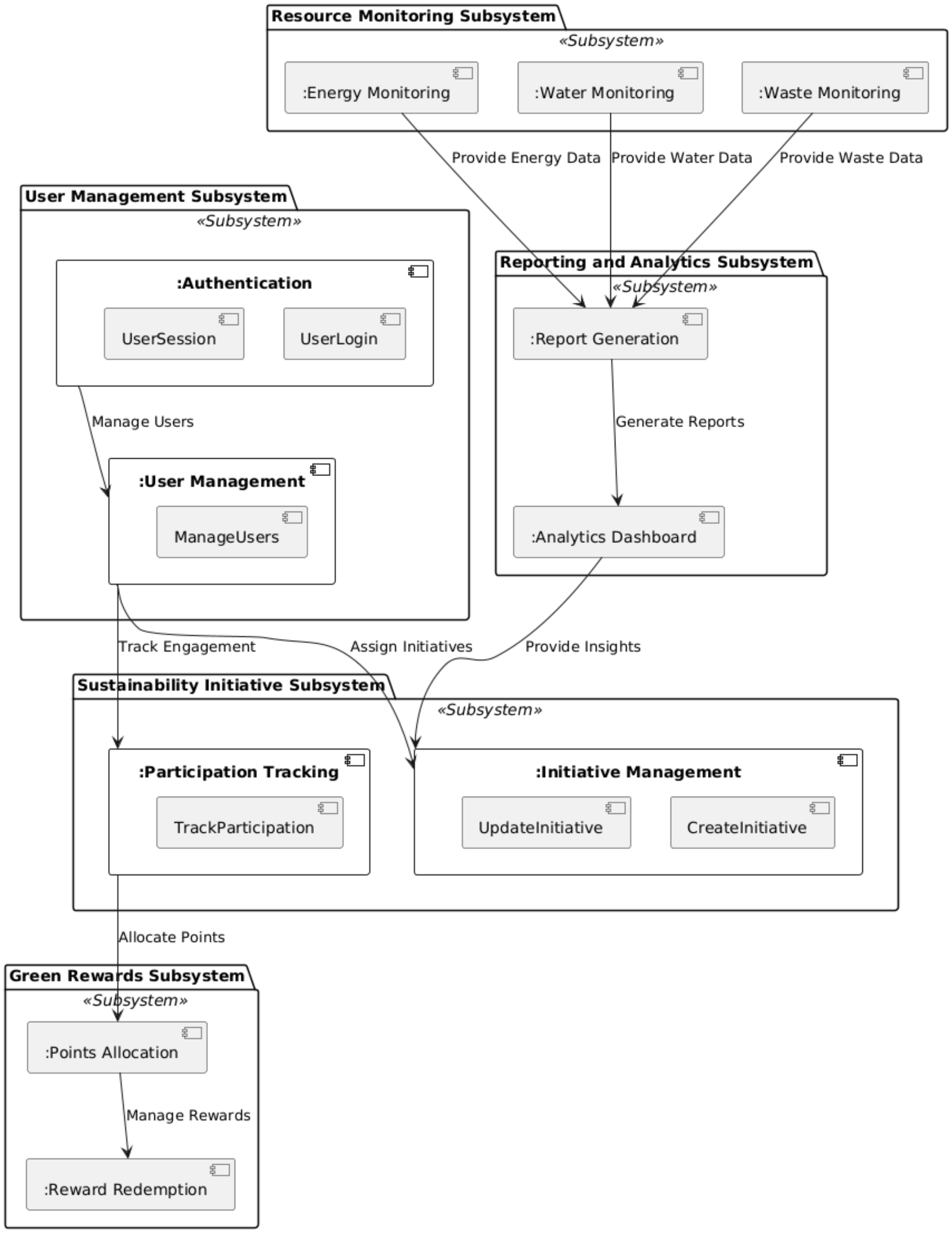
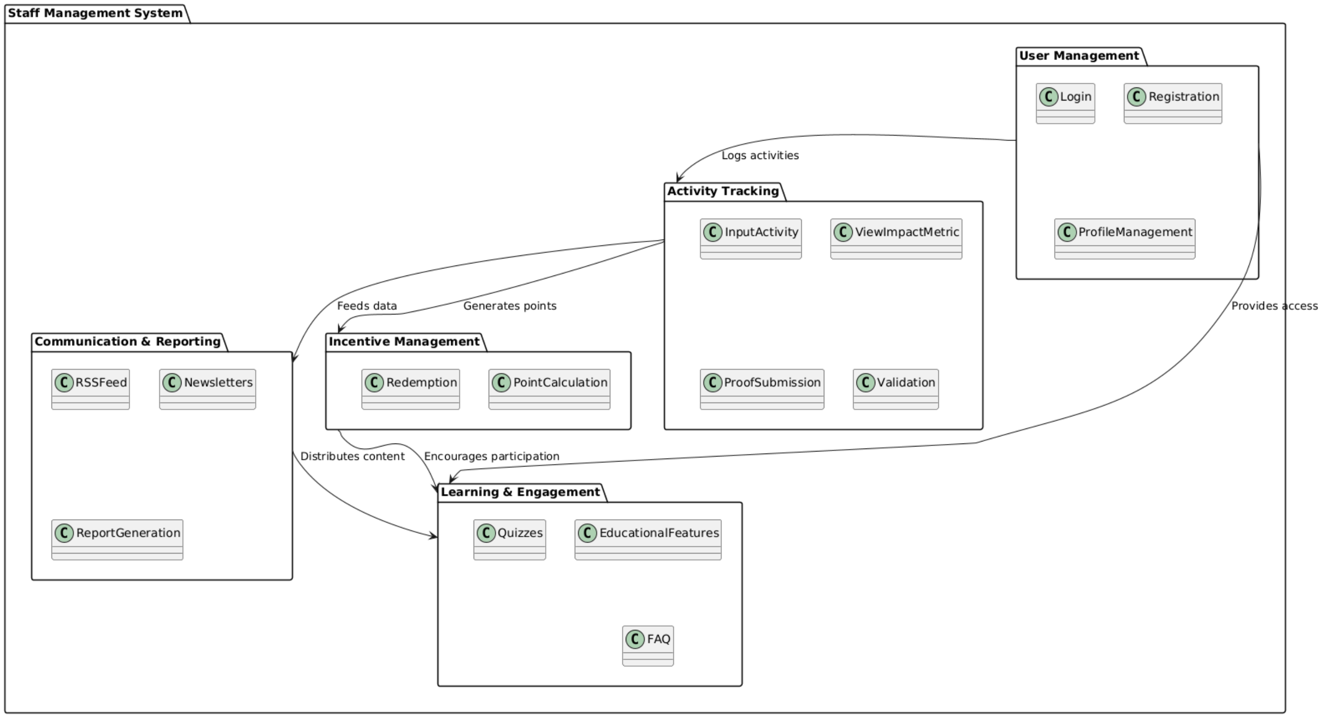


Figure 1: Component Diagram

# 4.0 Detailed Description of Components

## 4.1 Complete Package Diagram

The diagram represents a Staff Management System, comprising several interconnected packages: User Management for registration, login, and profile management; Activity Tracking for logging and validating user activities; Incentive Management for calculating and redeeming rewards based on activity; Learning & Engagement for educational features, quizzes, and FAQs to encourage growth; and Communication & Reporting for disseminating progress via RSS feeds, newsletters, and reports. These components work together, with User Management enabling access to learning, Activity Tracking feeding data to Incentive Management and Communication & Reporting, and Incentive Management motivating participation in learning, fostering a holistic approach to employee engagement, development, and performance tracking.

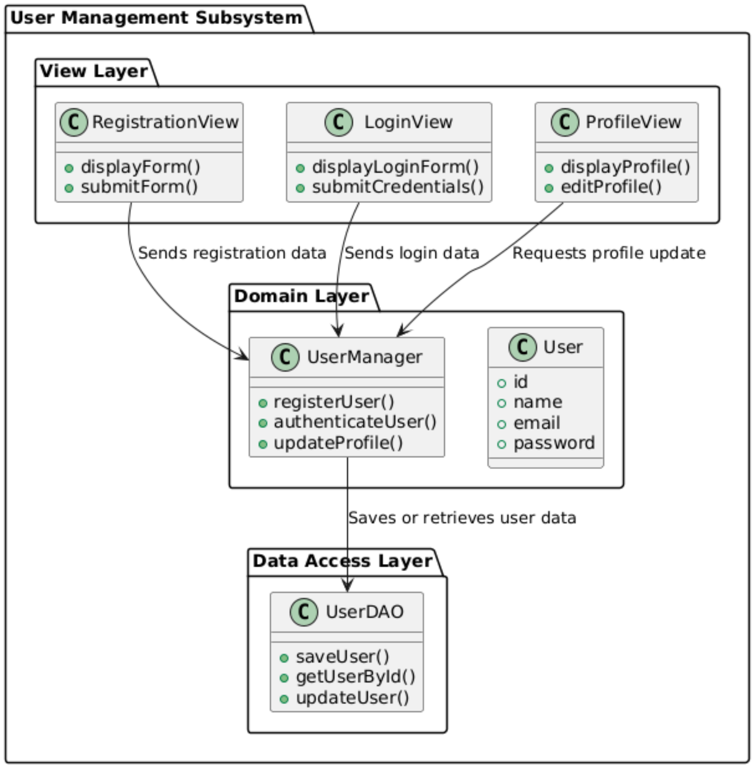


## 4.2 Detailed Description

The Staff Management System is divided into several subsystems, each with distinct classes, use cases, and sequence diagrams.

1. **User Management**: Includes classes for **Registration**, **Login**, and **Profile Management**. Use cases include user registration, login, and profile updates, with sequence diagrams illustrating interactions between users, the system, and the database for each action.
2. **Activity Tracking**: Comprising **Input Activity**, **View Impact Metric**, **Proof Submission**, and **Validation** classes. Use cases include logging activities, viewing metrics, submitting proof, and validating proof, shown in sequence diagrams for each flow.
3. **Incentive Management**: Contains **Point Calculation** and **Redemption** classes. Use cases cover point calculation based on activities and redemption for rewards, illustrated in sequence diagrams with user interactions.
4. **Learning & Engagement**: Features **Educational Features**, **Quizzes**, and **FAQ**. Use cases include accessing learning content, taking quizzes, and viewing FAQs, with sequence diagrams detailing user interactions.
5. **Communication & Reporting**: Includes **RSS Feed**, **Newsletters**, and **Report Generation**. Use cases involve viewing RSS feeds, receiving newsletters, and generating reports, captured in sequence diagrams.

# 4.2.1 P001: User Management Subsystem



## 4.2.1.1 Class Diagram

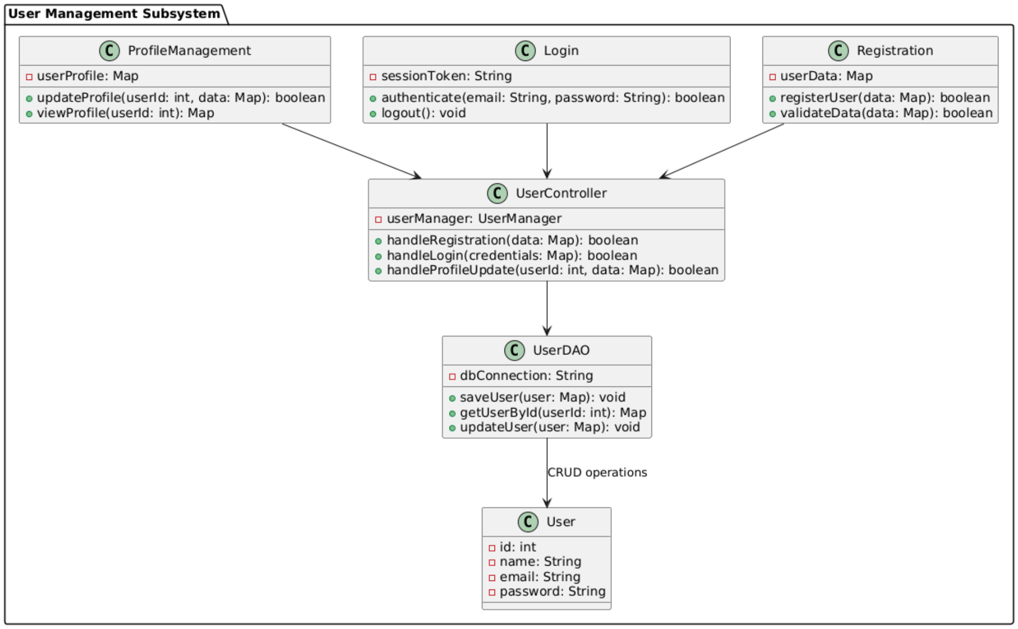

User Management Subsystem
ProfileManagement
userProfile: Map
updateProfile(userId: int, data: Map): boolean
viewProfile(userId: int): Map
Login
sessionToken: String
authenticate(email: String, password: String): boolean
logout(): void
Registration
userData: Map
registerUser(data: Map): boolean
validateData(data: Map): boolean
UserController
userManager: UserManager
handleRegistration(data: Map): boolean
handleLogin(credentials: Map): boolean
handleProfileUpdate(userId: int, data: Map): boolean
UserDAO
dbConnection: String
saveUser(user: Map): void
getUserById(userId: int): Map
updateUser(user: Map): void
CRUD operations
User
id: int
name: String
email: String
password: String

**Table of Methods for Each Entity**

| Entity | Method Name | Input | Output |
|---|---|---|---|
| **RegistrationView** | displayForm | None | Displays registration form |
| | submitForm | User data (name, email, password) | Registers the user |
| **LoginView** | displayLoginForm | None | Displays login form |
| | submitCredentials | User credentials (email, password) | Authenticates user |
| **ProfileView** | displayProfile | User ID | Displays profile data |
| | editProfile | Updated user data (name, email, etc.) | Updates user profile |
| **User** | N/A | N/A | Represents a user entity |
| **UserManager** | registerUser | User data (name, email, password) | User object |
| | authenticateUser | User credentials (email, password) | Boolean (True/False) |
| | updateProfile | User ID, updated user data | Updated User object |
| **UserDAO** | saveUser | User object | None |
| | getUserById | User ID | User object |
| | updateUser | User object | None |

**RegistrationView**

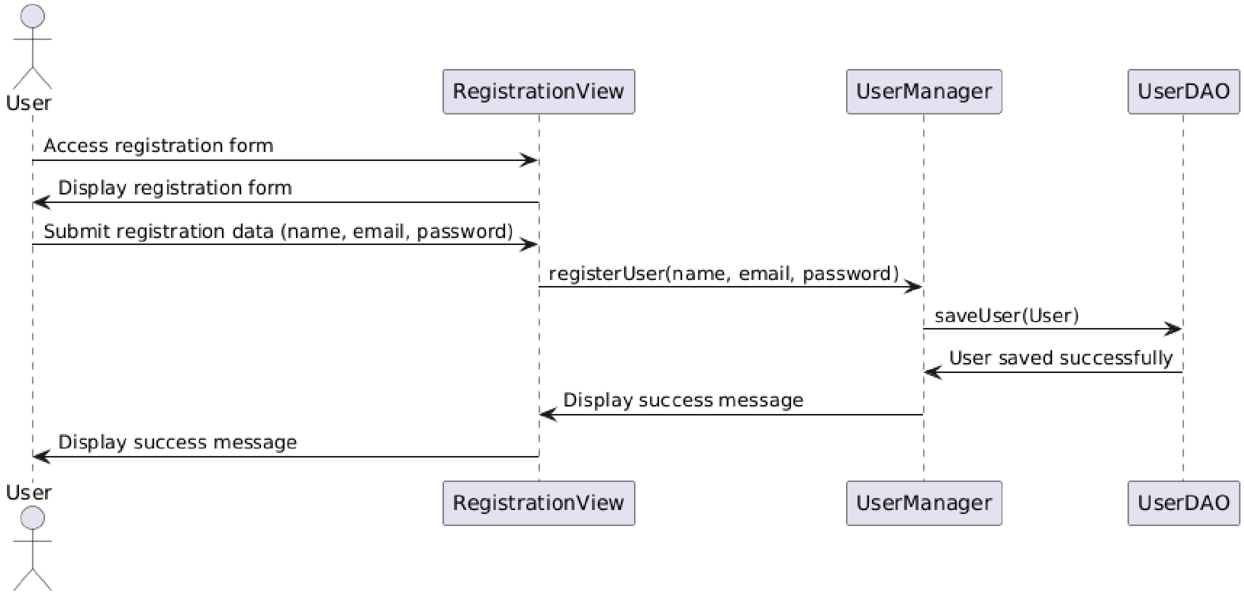
| Entity | RegistrationView |
|---|---|
| **Method Name** | displayForm |
| **Input** | None |
| **Output** | Displays registration form |
| **Algorithm:** | Step 1: Start<br>Step 2: Display registration form<br>Step 3: End |

**ProfileView**

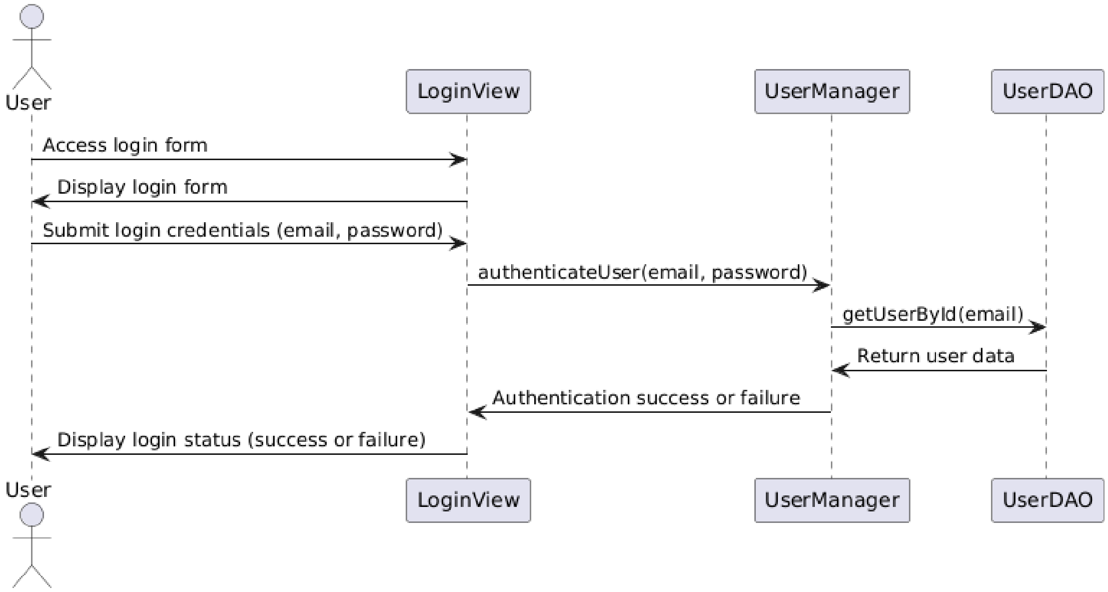
| **Entity** | **ProfileView** |
|---|---|
| **Method Name** | displayProfile |
| **Input** | User ID |
| **Output** | Displays profile data |
| **Algorithm:** | Step 1: Start<br>Step 2: Input User ID<br>Step 3: Call UserManager.updateProfile() to get profile data<br>Step 4: Display profile data<br>Step 5: End |

## 4.2.1.2 Sequence Diagram

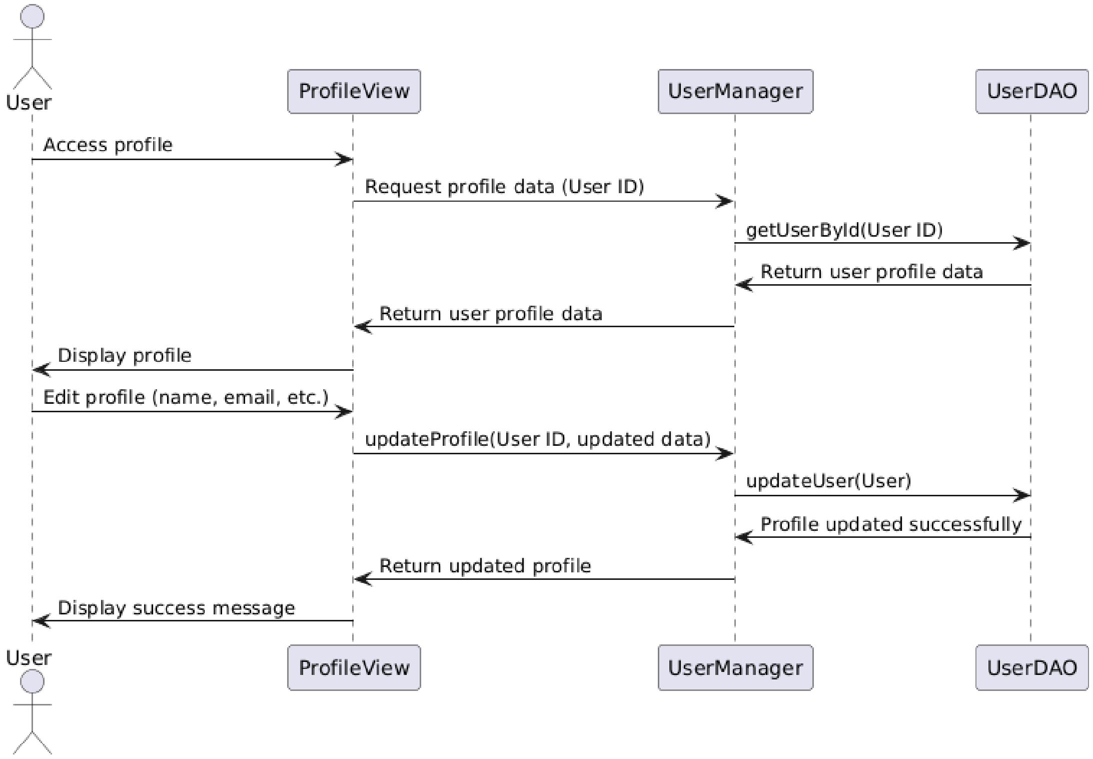
### a) SD001 Sequence Diagram for User Registration



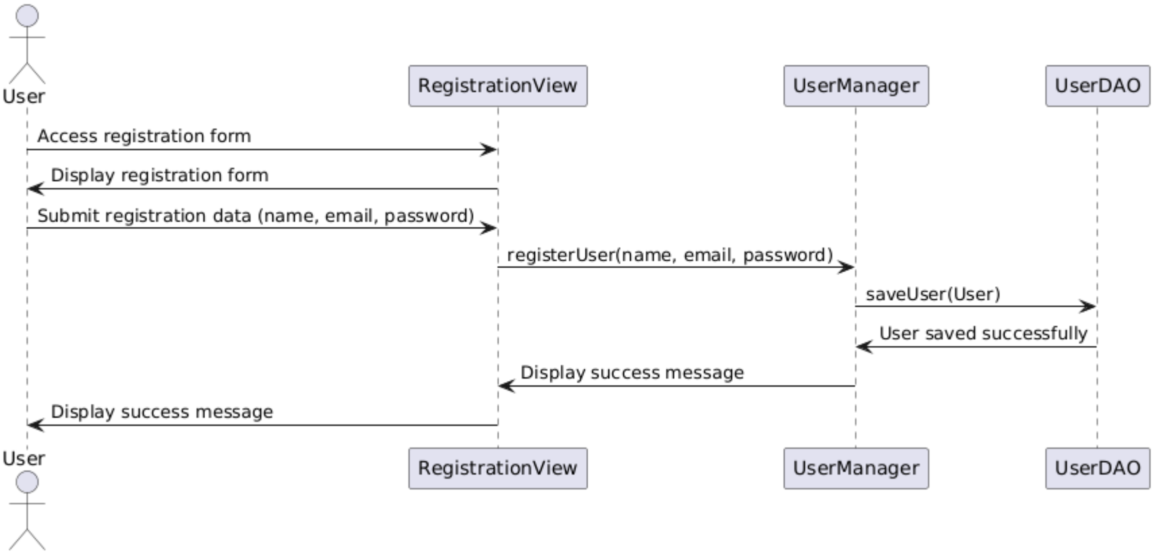
## b) SD002 Sequence Diagram for User Login


User
LoginView
UserManager
UserDAO
Access login form
Display login form
Submit login credentials (email, password)
authenticateUser(email, password)
getUserById(email)
Return user data
Authentication success or failure
Display login status (success or failure)
User
LoginView
UserManager
UserDAO

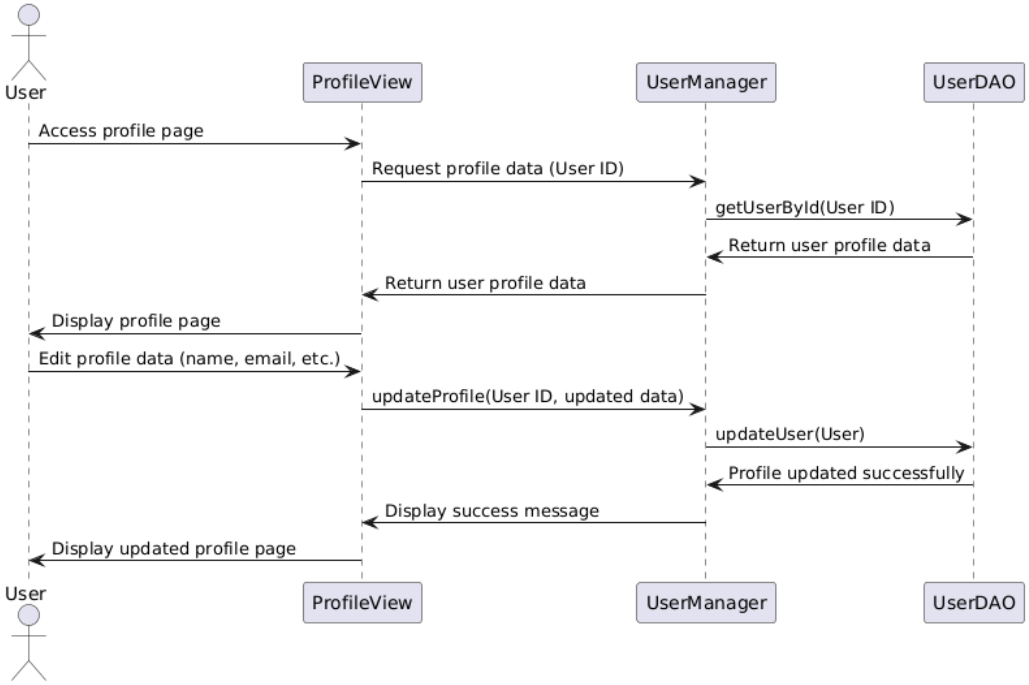
## c) SD003 Sequence Diagram for Profile Display and Edit



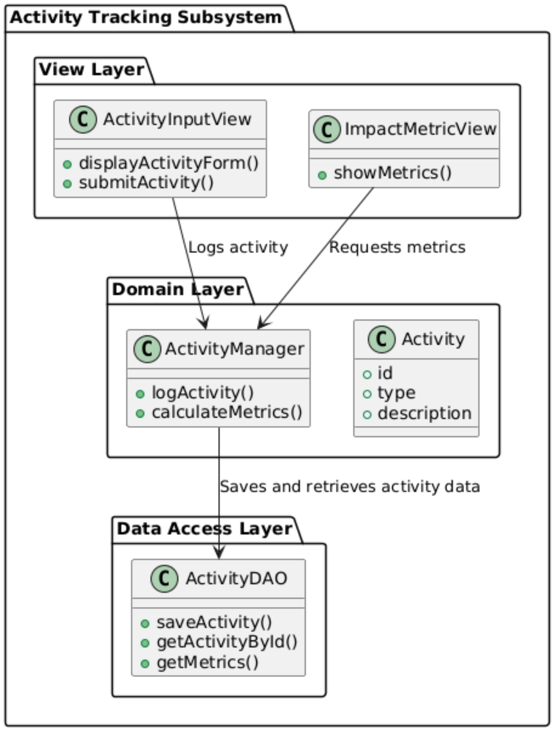
## d) SD004 Sequence Diagram for User Registration - Flow



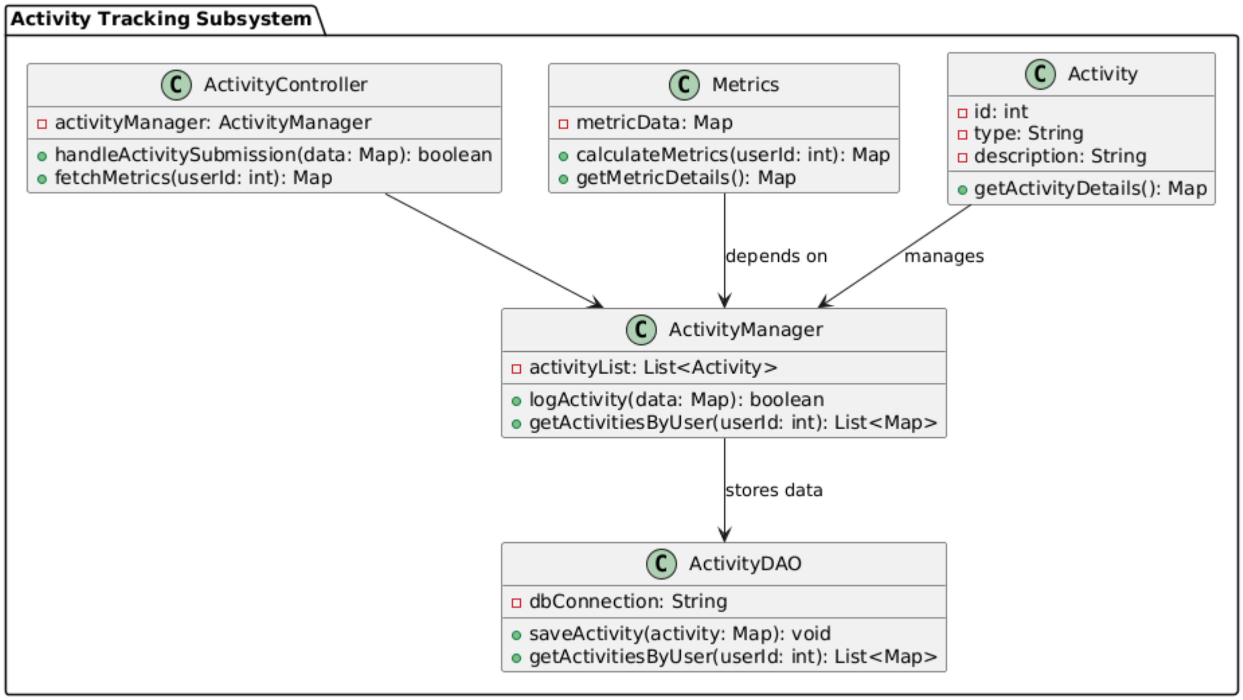
## e) SD003 Sequence Diagram for Profile Edit – Flow



## 4.2.2 P002: Activity Tracking Subsystem



### 4.2.2.1 Class Diagram


Activity Tracking Subsystem
ActivityController
activityManager: ActivityManager
handleActivitySubmission(data: Map): boolean
fetchMetrics(userId: int): Map
Metrics
metricData: Map
calculateMetrics(userId: int): Map
getMetricDetails(): Map
Activity
id: int
type: String
description: String
getActivityDetails(): Map
depends on
manages
ActivityManager
activityList: List<Activity>
logActivity(data: Map): boolean
getActivitiesByUser(userId: int): List<Map>
stores data
ActivityDAO
dbConnection: String
saveActivity(activity: Map): void
getActivitiesByUser(userId: int): List<Map>

## Table of Methods for Each Entity

| **Entity** | **Method Name** | **Input** | **Output** |
|---|---|---|---|
| **ActivityInputView** | displayActivityForm | None | Displays activity input form |
|  | submitActivity | Activity data (type, description) | Logs the activity |
| **ImpactMetricView** | showMetrics | None | Displays impact metrics |
| **Activity** | N/A | N/A | Represents an activity entity |
| **ActivityManager** | logActivity | Activity data (type, description) | Activity object |
|  | calculateMetrics | Activity data | Calculated metrics |
| **ActivityDAO** | saveActivity | Activity object | None |
|  | getActivityById | Activity ID | Activity object |
|  | getMetrics | None | Impact metrics |

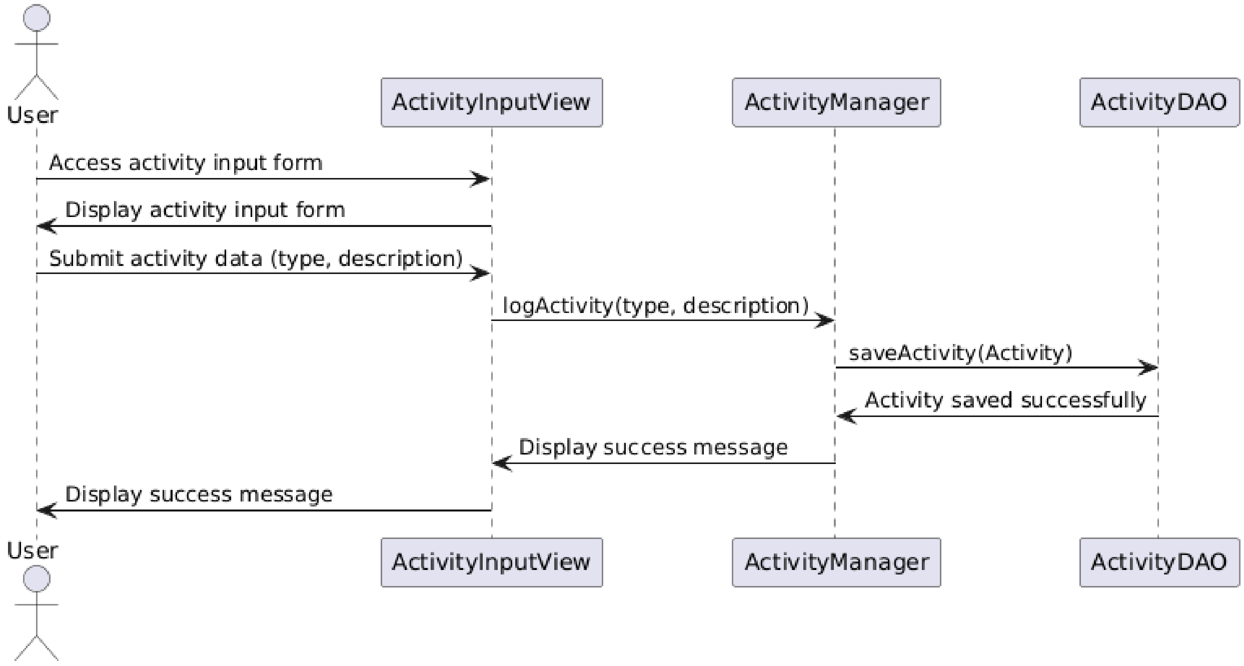
### ActivityInputView

| | |
|---|---|
| **Entity** | **ActivityInputView** |
| **Method Name** | displayActivityForm |
| **Input** | None |
| **Output** | Displays activity input form |
| **Algorithm** | Step 1: Start<br>Step 2: Display activity input form<br>Step 3: End |

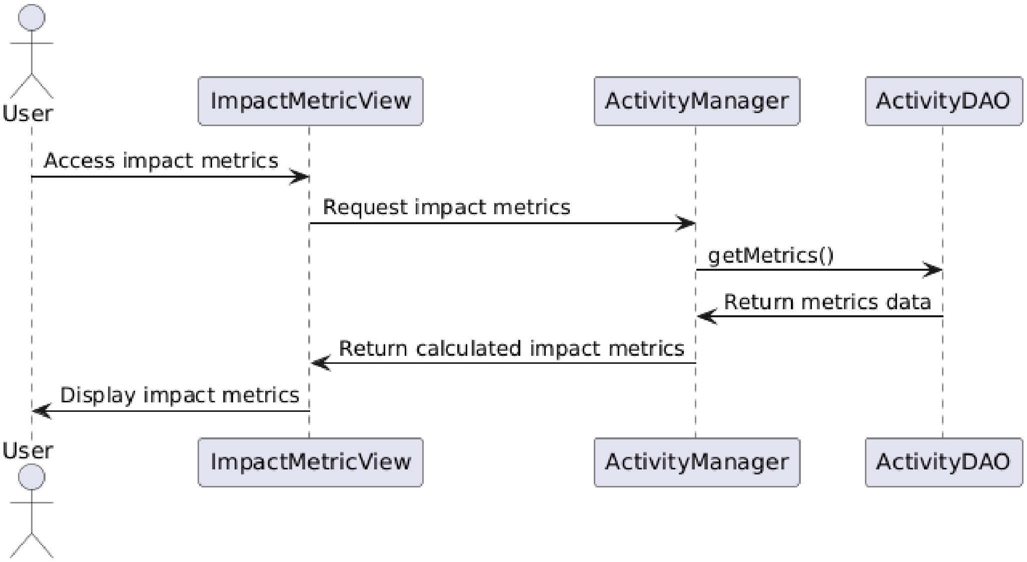
| **Entity** | **ActivityManager** |
|---|---|
| **Method Name** | logActivity |
| **Input** | Activity data (type, description) |
| **Output** | Activity object |
| **Algorithm** | Step 1: Start<br>Step 2: Input activity data (type, description)<br>Step 3: Validate activity data<br>Step 4: Call ActivityDAO.saveActivity() to save activity<br>Step 5: Return Activity object<br>Step 6: End |

## 4.2.2.2 Sequence Diagram

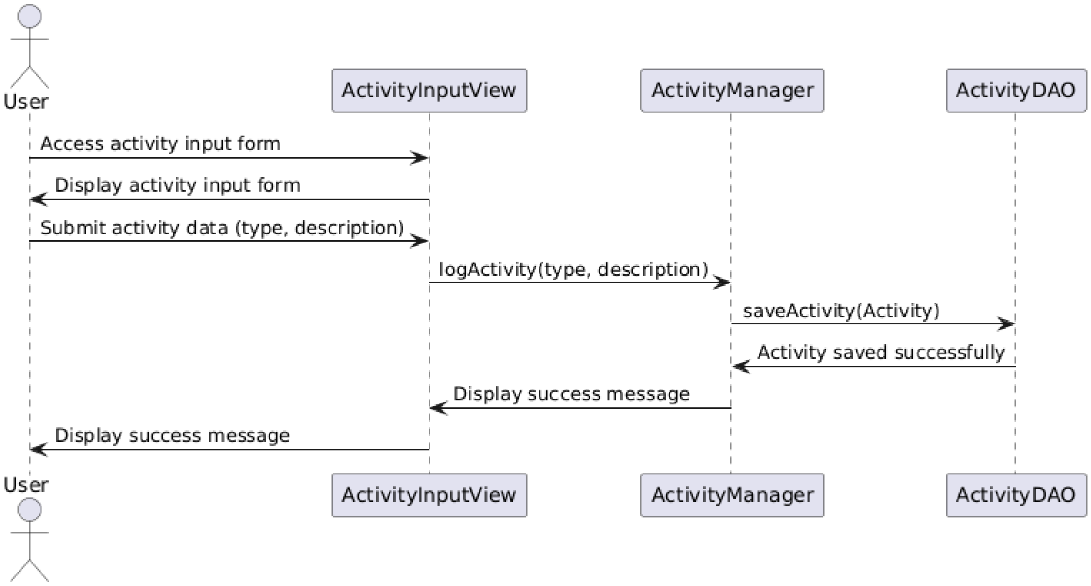
### a) Sequence diagram for Log Activity



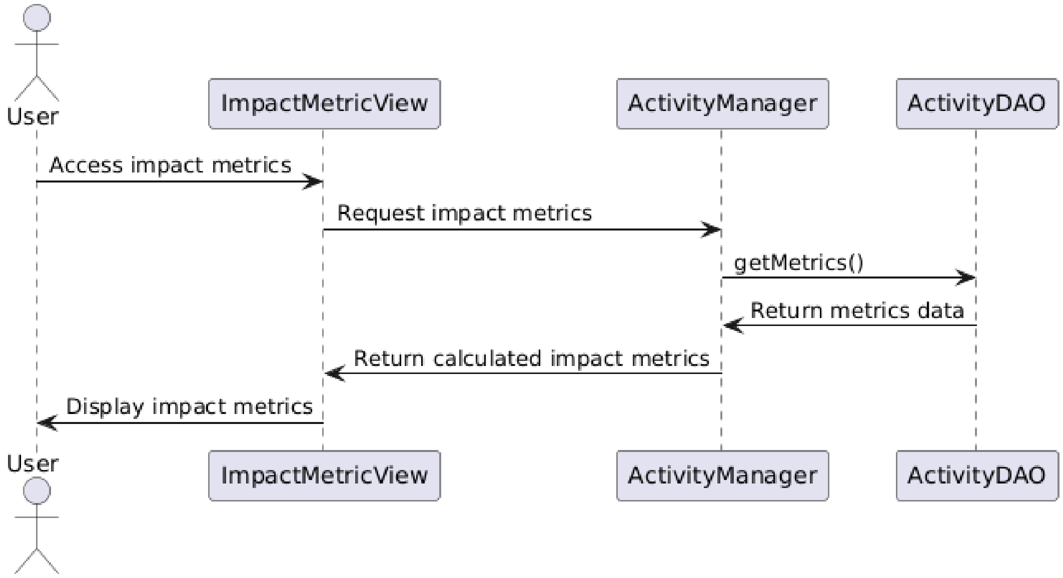
## b) Sequence diagram for Show Impact Metrics



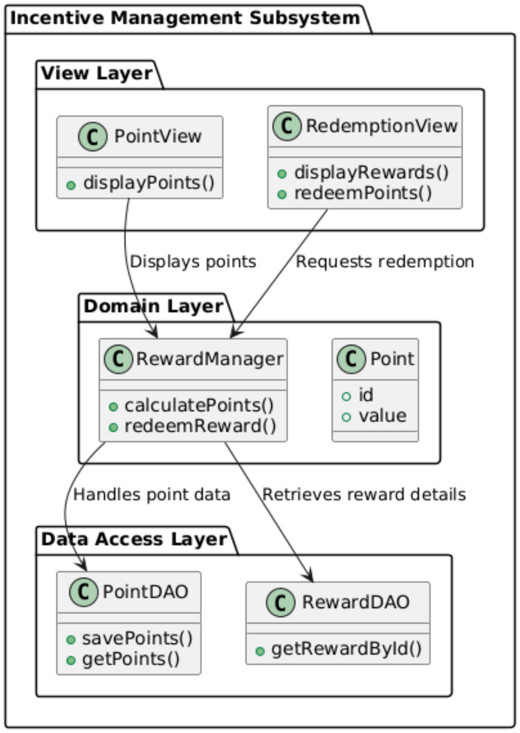
## c) Sequence diagram for Log Activity – Flow



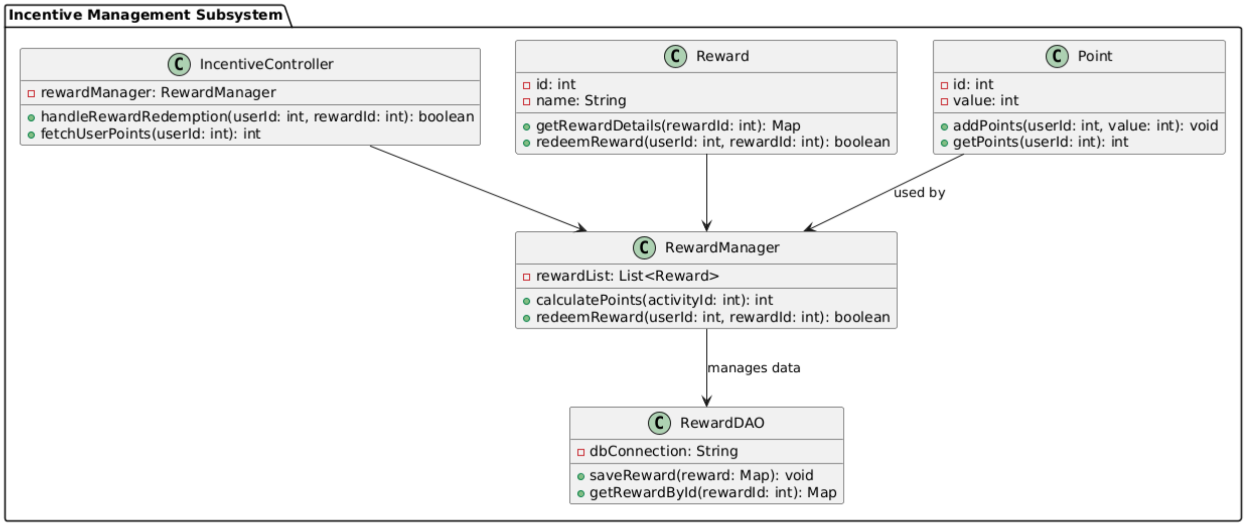
### d) Sequence diagram for Calculate Metrics – Flow



## 4.2.3 P003: Incentive Management Subsystem



## 4.2.3.1 Class Diagram


Incentive Management Subsystem
IncentiveController
rewardManager: RewardManager
handleRewardRedemption(userId: int, rewardId: int): boolean
fetchUserPoints(userId: int): int
Reward
id: int
name: String
getRewardDetails(rewardId: int): Map
redeemReward(userId: int, rewardId: int): boolean
Point
id: int
value: int
addPoints(userId: int, value: int): void
getPoints(userId: int): int
used by
RewardManager
rewardList: List&lt;Reward&gt;
calculatePoints(activityId: int): int
redeemReward(userId: int, rewardId: int): boolean
manages data
RewardDAO
dbConnection: String
saveReward(reward: Map): void
getRewardById(rewardId: int): Map

## Table of Methods for Each Entity

| **Entity** | **Method Name** | **Input** | **Output** |
|---|---|---|---|
| **PointView** | displayPoints | None | Displays points |
| **RedemptionView** | displayRewards | None | Displays reward options |
| | redeemPoints | Points to redeem | Redeemed reward |
| **Point** | N/A | N/A | Represents a point entity |
| **RewardManager** | calculatePoints | Activity or transaction data | Total points |
| | redeemReward | Points, Reward ID | Redeemed reward |
| **PointDAO** | savePoints | Point data (value) | None |
| | getPoints | User ID or Point ID | Point data |
| **RewardDAO** | getRewardById | Reward ID | Reward details |

**RewardManager**

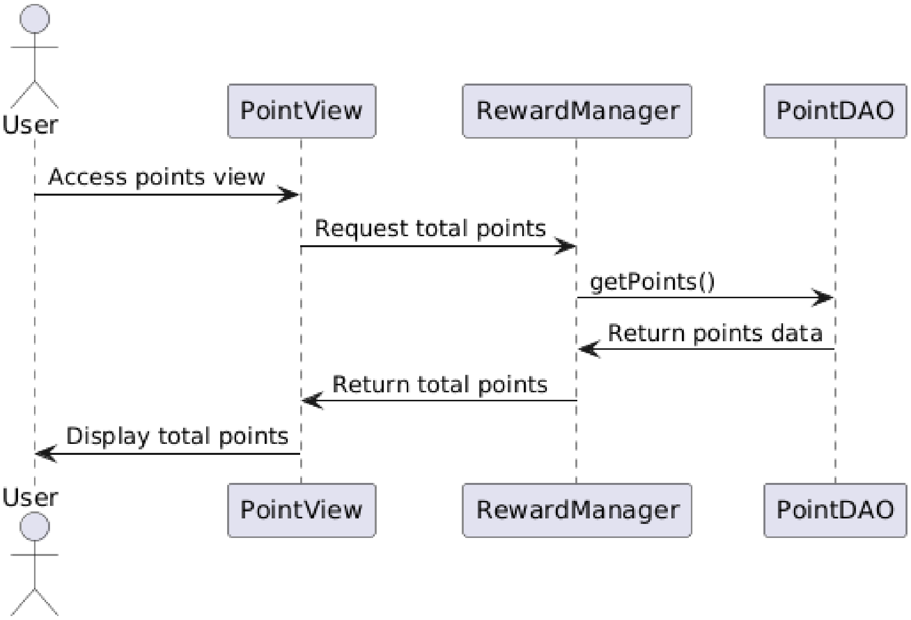
| **Entity** | **RewardManager** |
|---|---|
| **Method Name** | calculatePoints |
| **Input** | Activity or transaction data |
| **Output** | Total points |
| **Algorithm** | Step 1: Start<br>Step 2: Input activity or transaction data<br>Step 3: Calculate total points based on input data<br>Step 4: Return total points<br>Step 5: End |

**RewardDAO**

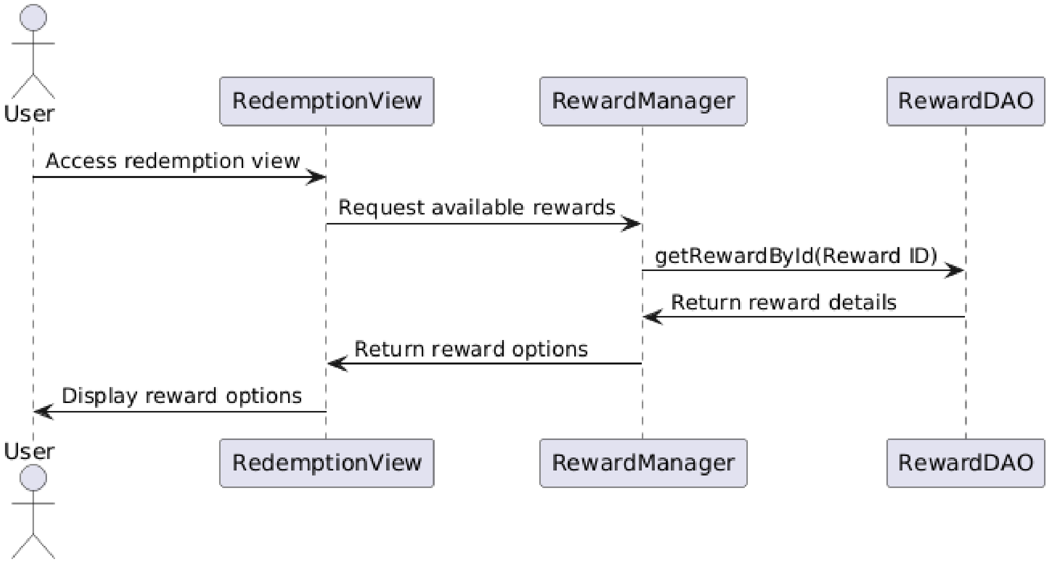
| **Entity** | **RewardDAO** |
|---|---|
| **Method Name** | getRewardById |
| **Input** | Reward ID |
| **Output** | Reward details |
| **Algorithm** | Step 1: Start<br>Step 2: Input Reward ID<br>Step 3: Retrieve reward details from the database<br>Step 4: Return reward details<br>Step 5: End |

## 4.2.3.2 Sequence Diagram

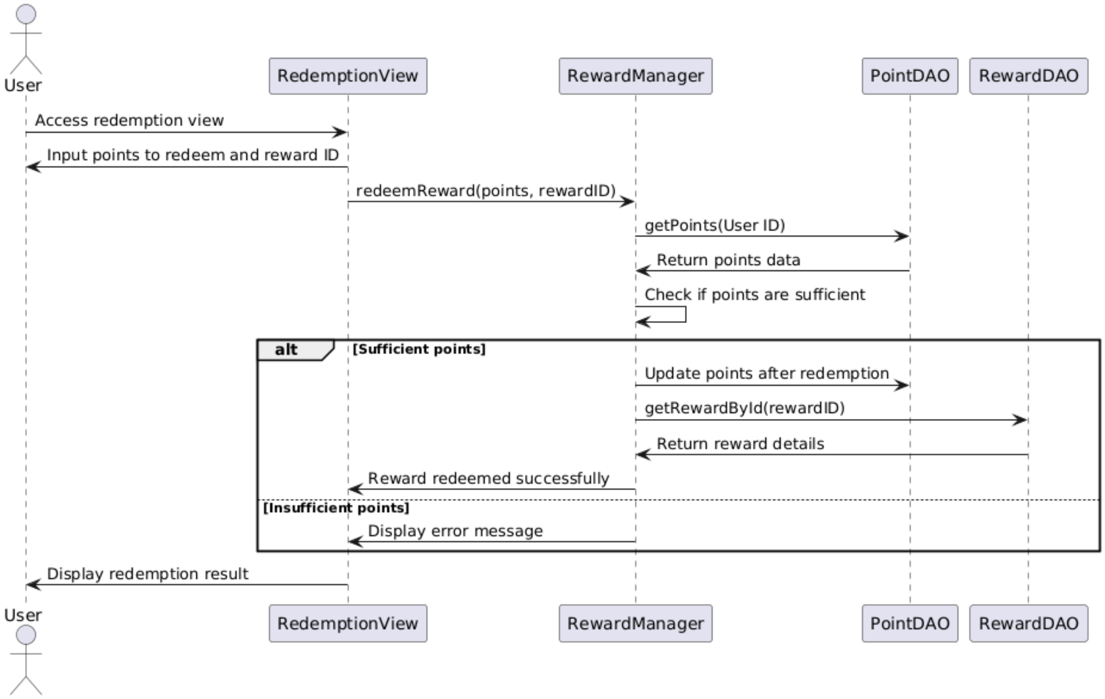
### a) Sequence diagram for Display Points



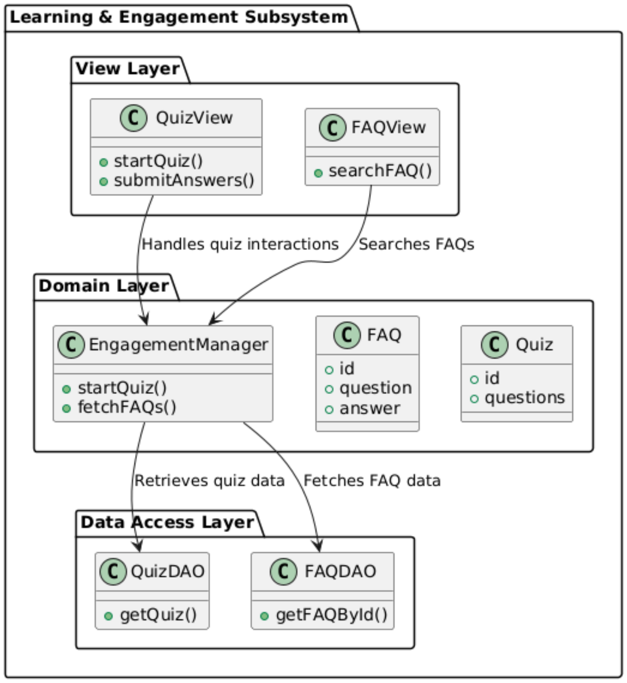
## b) Sequence diagram for Display Rewards



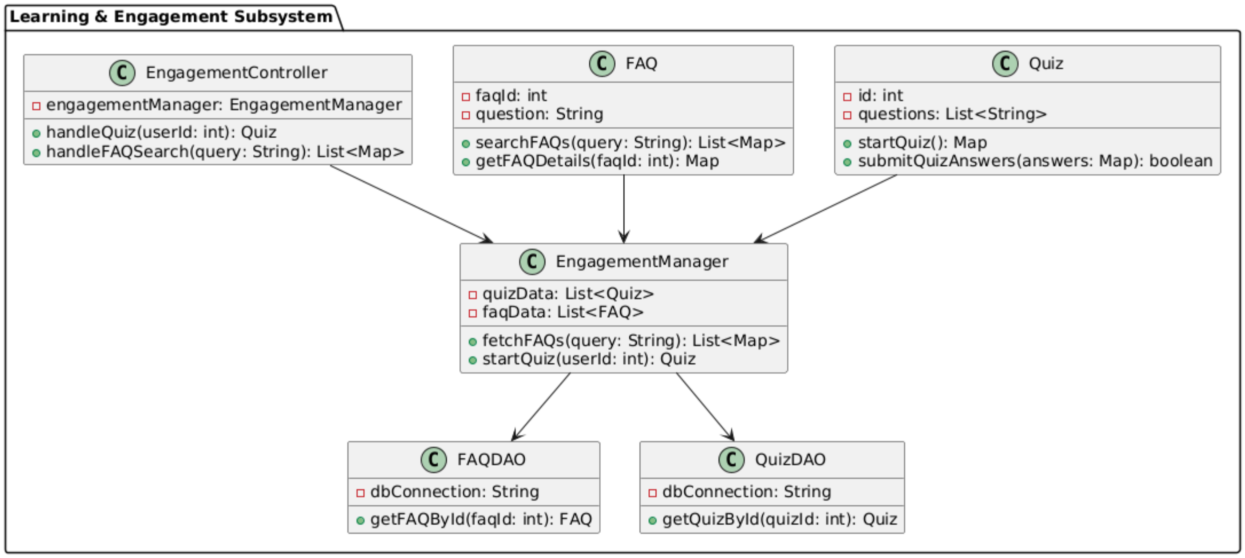
## c) Sequence diagram for Redeem Points



# 4.2.4 P004: Learning & Engagement Subsystem



## 4.2.4.1 Class Diagram


Learning & Engagement Subsystem
EngagementController
engagementManager: EngagementManager
handleQuiz(userId: int): Quiz
handleFAQSearch(query: String): List<Map>
FAQ
faqId: int
question: String
searchFAQs(query: String): List<Map>
getFAQDetails(faqId: int): Map
Quiz
id: int
questions: List<String>
startQuiz(): Map
submitQuizAnswers(answers: Map): boolean
EngagementManager
quizData: List<Quiz>
faqData: List<FAQ>
fetchFAQs(query: String): List<Map>
startQuiz(userId: int): Quiz
FAQDAO
dbConnection: String
getFAQById(faqId: int): FAQ
QuizDAO
dbConnection: String
getQuizById(quizId: int): Quiz

## Table of Methods for Each Entity

| Entity | Method Name | Input | Output |
|---|---|---|---|
| **QuizView** | startQuiz | None | Starts the quiz |
| | submitAnswers | User's answers to quiz questions | Submits answers |
| **FAQView** | searchFAQ | Search query (e.g., keyword) | Display search results |
| **Quiz** | N/A | N/A | Represents a quiz |
| **FAQ** | N/A | N/A | Represents an FAQ entry |
| **EngagementManager** | startQuiz | None | Starts the quiz |
| | fetchFAQs | Search query | Returns FAQs |
| **QuizDAO** | getQuiz | Quiz ID | Quiz data |
| **FAQDAO** | getFAQById | FAQ ID | FAQ data |

**FAQView**

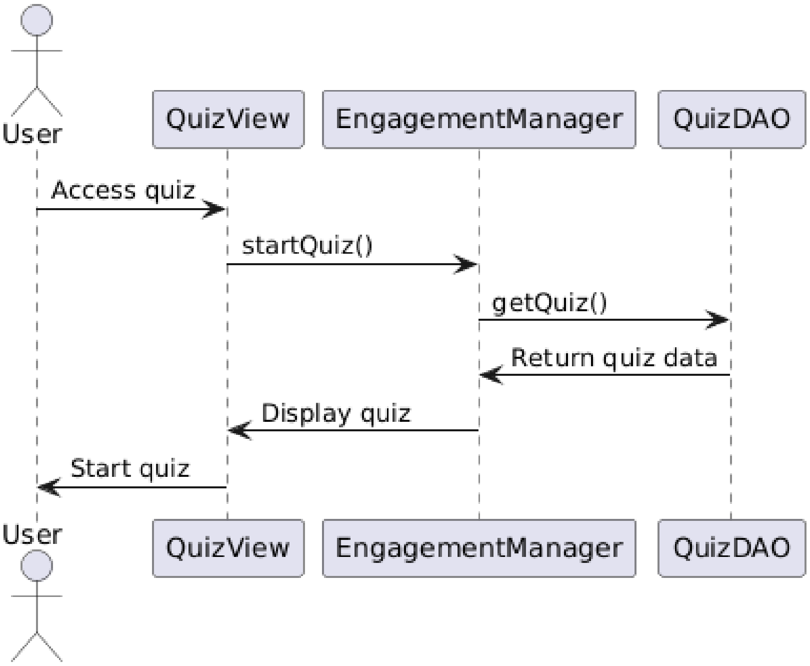
| **Entity** | **FAQView** |
|---|---|
| **Method Name** | searchFAQ |
| **Input** | Search query (e.g., keyword) |
| **Output** | Display search results |
| **Algorithm** | Step 1: Start<br>Step 2: Input search query<br>Step 3: Call EngagementManager.fetchFAQs() to search FAQs<br>Step 4: Display search results<br>Step 5: End |

**FAQDAO**

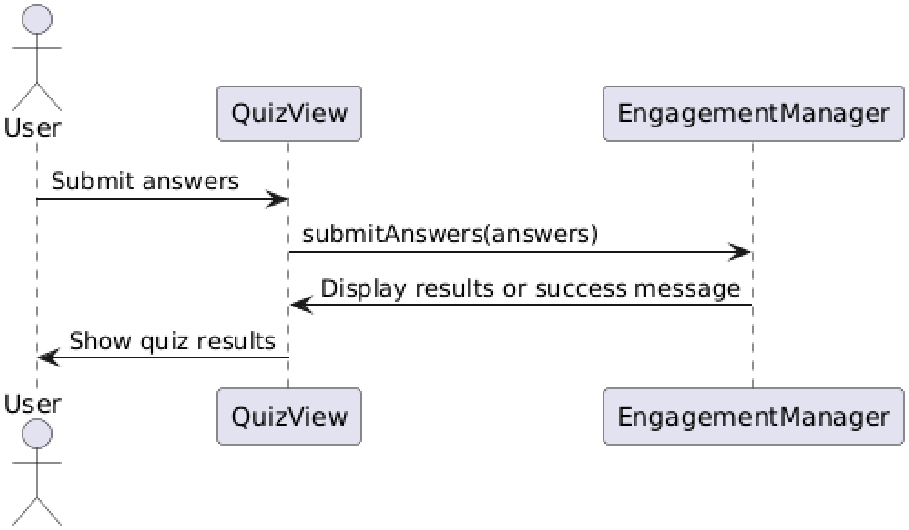
| **Entity** | FAQDAO |
|---|---|
| **Method Name** | getFAQById |
| **Input** | FAQ ID |
| **Output** | FAQ data |
| **Algorithm** | Step 1: Start<br>Step 2: Input FAQ ID<br>Step 3: Retrieve FAQ data from the database<br>Step 4: Return FAQ data<br>Step 5: End |

## 4.2.4.2 Sequence Diagram

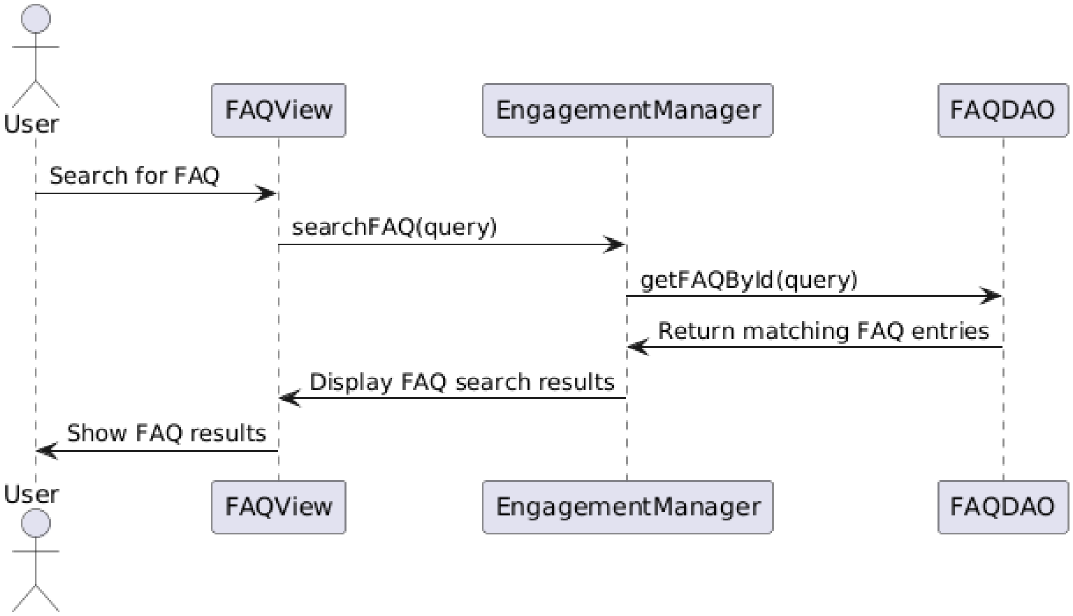
### a) Sequence diagram for Start Quiz



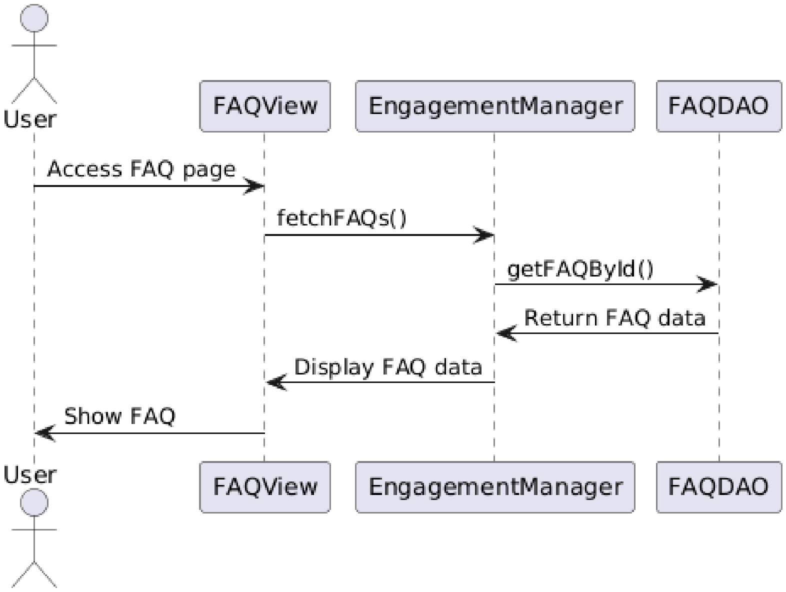
### b) Sequence diagram for Submit Answers



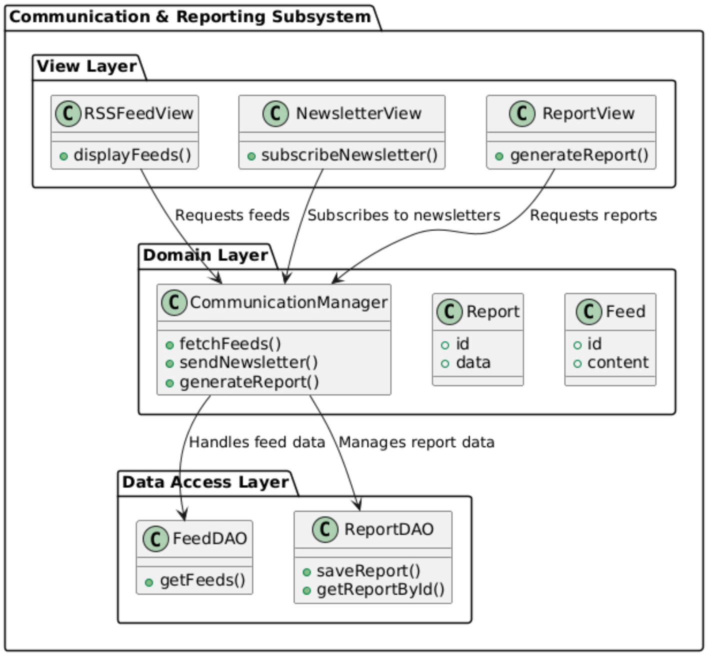
### c) Sequence diagram for Search FAQ



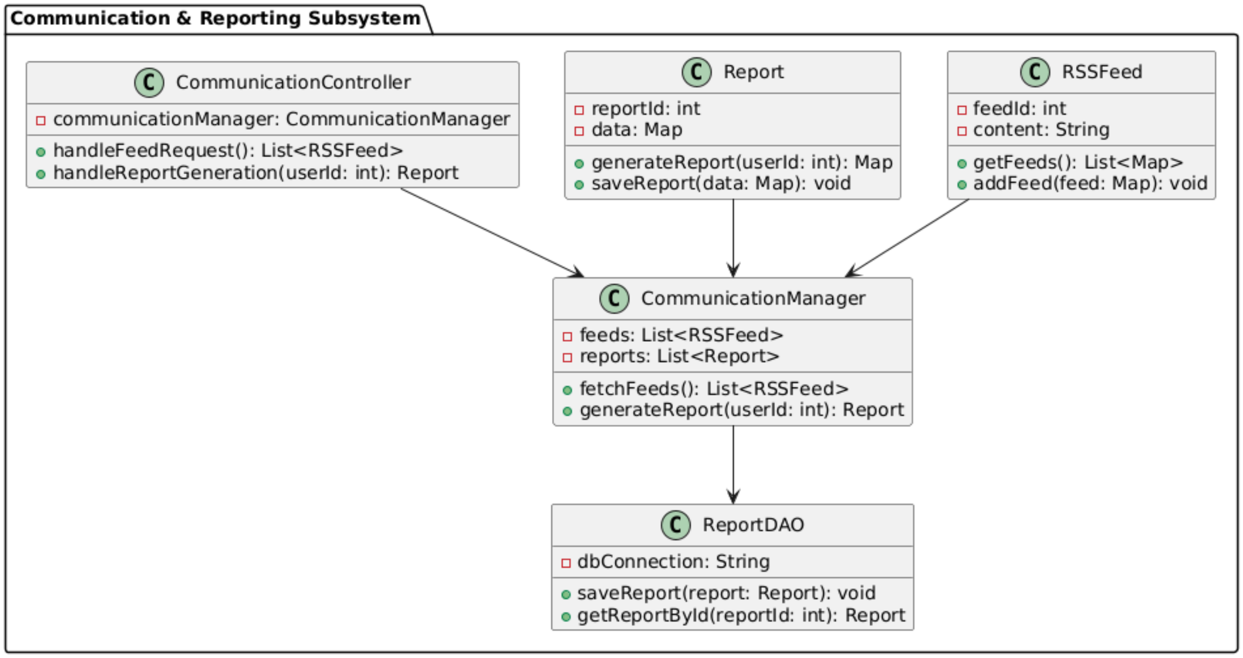
## d) Sequence diagram for View FAQ



### 4.2.5 P005: Communication & Reporting Subsystem



#### 4.2.5.1 Class Diagram


Communication & Reporting Subsystem
CommunicationController
communicationManager: CommunicationManager
handleFeedRequest(): List<RSSFeed>
handleReportGeneration(userId: int): Report
Report
reportId: int
data: Map
generateReport(userId: int): Map
saveReport(data: Map): void
RSSFeed
feedId: int
content: String
getFeeds(): List<Map>
addFeed(feed: Map): void
CommunicationManager
feeds: List<RSSFeed>
reports: List<Report>
fetchFeeds(): List<RSSFeed>
generateReport(userId: int): Report
ReportDAO
dbConnection: String
saveReport(report: Report): void
getReportById(reportId: int): Report

## Table of Methods for Each Entity

| Entity | Method Name | Input | Output |
|---|---|---|---|
| **RSSFeedView** | displayFeeds | None | Displays RSS feed |
| **NewsletterView** | subscribeNewsletter | User email | Subscribes to newsletter |
| **ReportView** | generateReport | Report parameters (e.g., report type) | Generates report |
| **Feed** | N/A | N/A | Represents a feed |
| **Report** | N/A | N/A | Represents a report |
| **CommunicationManager** | fetchFeeds | None | Fetches RSS feeds |
| | sendNewsletter | User email | Sends newsletter |
| | generateReport | Report parameters (e.g., report type) | Generates report |
| **FeedDAO** | getFeeds | None | Returns RSS feed data |
| **ReportDAO** | saveReport | Report data | Saves report data |
| | getReportById | Report ID | Returns report data |

**NewsletterView**

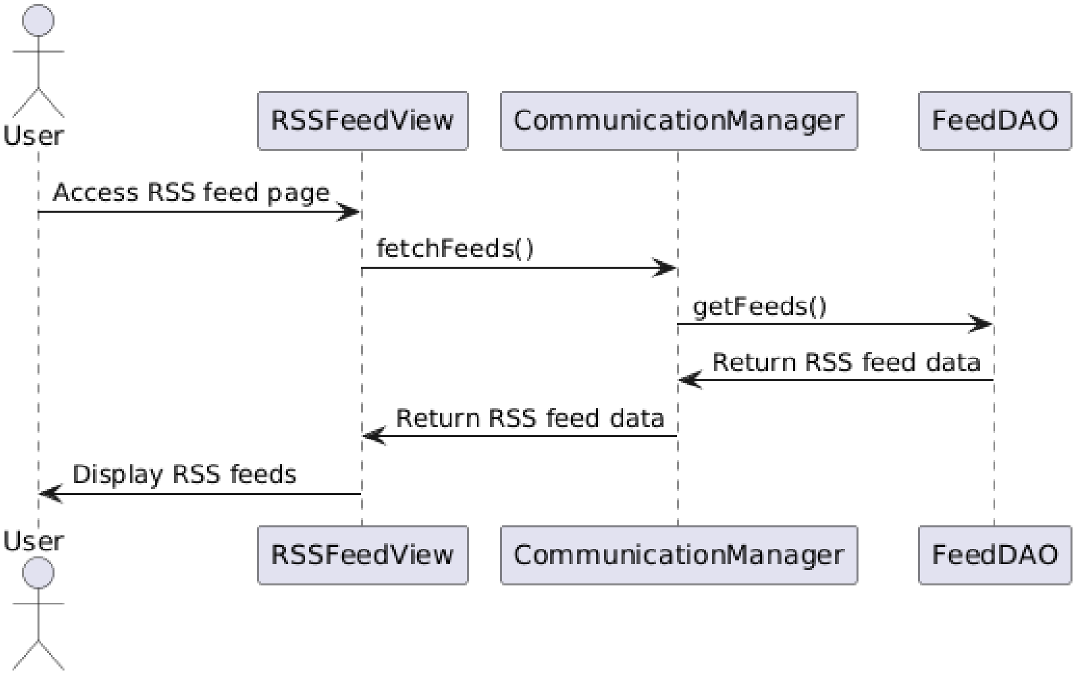
| | |
|---|---|
| **Entity** | NewsletterView |
| **Method Name** | subscribeNewsletter |
| **Input** | User email |
| **Output** | Subscribes the user to the newsletter |
| **Algorithm** | Step 1: Start<br>Step 2: Input user's email<br>Step 3: Call CommunicationManager.sendNewsletter(email)<br>Step 4: Display subscription confirmation<br>Step 5: End |

**ReportView**

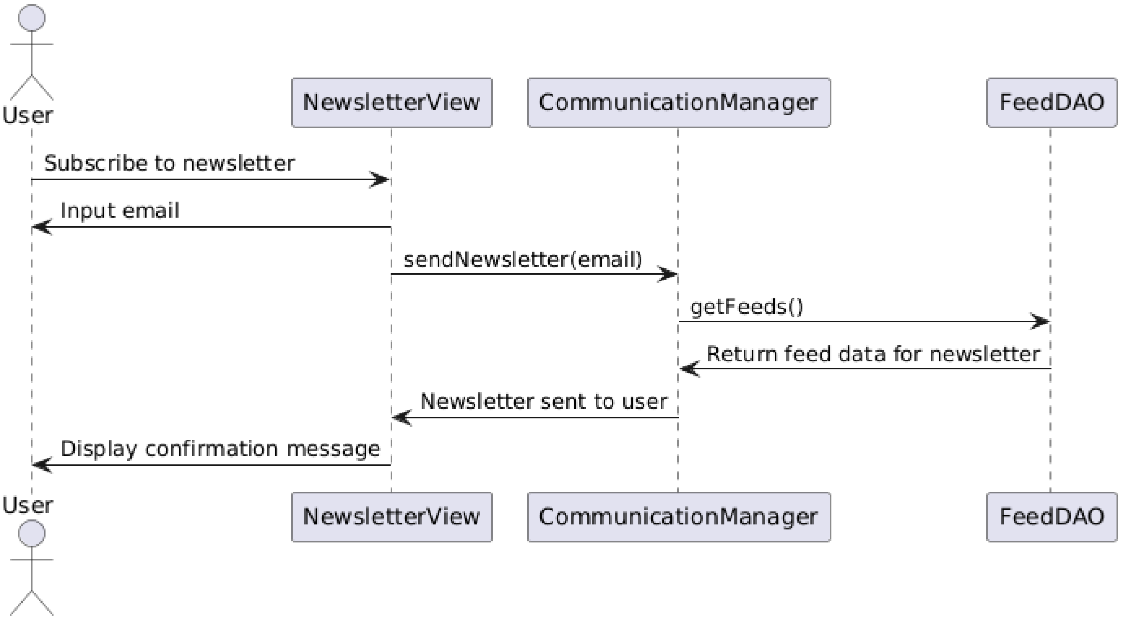
| **Entity** | ReportView |
|---|---|
| **Method Name** | generateReport |
| **Input** | Report parameters (e.g., report type) |
| **Output** | Generates report |
| **Algorithm** | Step 1: Start<br>Step 2: Input report parameters (type, filters, etc.)<br>Step 3: Call CommunicationManager.generateReport(parameters)<br>Step 4: Display generated report<br>Step 5: End |

## 4.2.5.2 Sequence Diagram

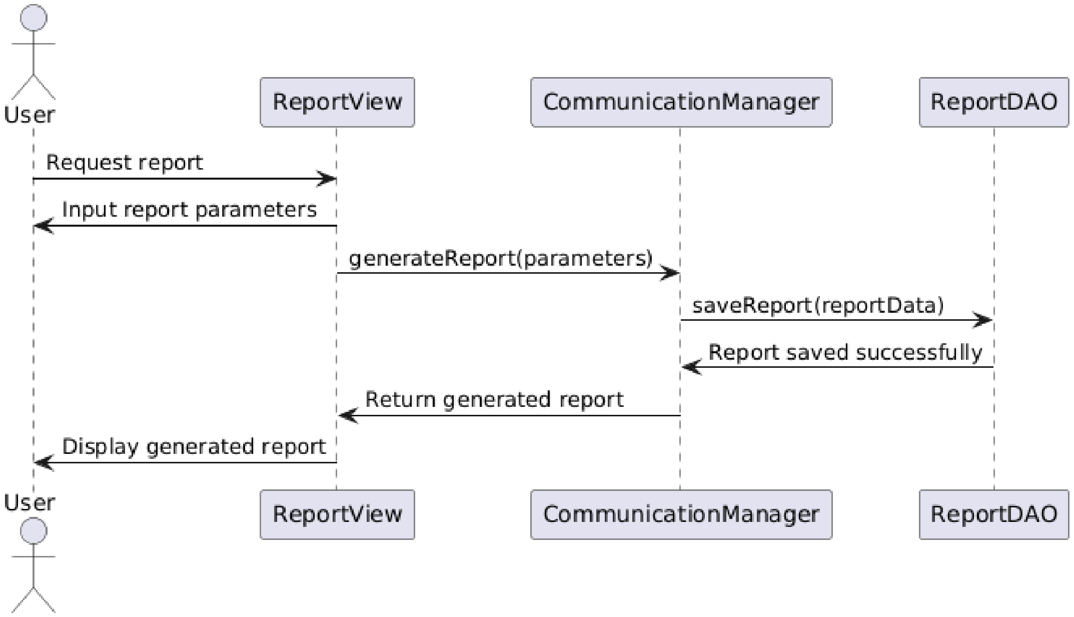
### a) Sequence diagram for Display RSS Feeds



## b) Sequence diagram for Subscribe to Newsletter



## c) Sequence diagram for Generate Report



# 5.0 DATA DESIGN

## 5.1 Data Description

The database for the MyGreen UTM application is essential for managing sustainability operations within the organization. It handles core functionalities for staff, focusing on key data entities like users, sustainability programs, and participation tracking. The relational database processes and organizes data to track carbon footprints, manage programs, and generate reports. By analyzing user activities and participation, it provides insights to monitor sustainability progress, assess program effectiveness, and guide data-driven decisions. It also enables seamless integration with other systems, ensuring efficient data flow and accurate reporting. This comprehensive approach allows for timely adjustments to sustainability strategies based on real-time data. The database's structure supports scalability, making it adaptable to future growth and evolving sustainability goals. It ensures that all stakeholders have access to up-to-date information, fostering collaboration and transparency across the organization. The system is designed for ease of use, allowing both technical and non-technical staff to interact with the data efficiently. Additionally, its robust security features safeguard sensitive information, ensuring compliance with data protection regulations. Ultimately, the database empowers the organization to achieve its environmental goals while promoting continuous improvement in sustainability practices.

**Table 5.1: Description of Entities in the Database**

| No. | Entity Name | Description |
|---|---|---|
| 1 | User | Stores staff information, including login credentials and profiles |
| 2 | Carbon Footprint | Logs user activities and their environmental impact |
| 3 | Green Initiative | Details of proposed and ongoing eco – friendly initiatives |
| 4 | Participation | Tracks staff involvement in green initiatives and activities |
| 5 | Reports | Holds data for generating performance and progress analytics |

## 5.2 Data Dictionary

The data dictionary specifies the structure and attributes of each entity, ensuring seamless integration and interaction between system components. Each entity is represented with attributes that define its role in the database.

### 5.2.1 Entity 1: User

| Attribute Name | Type | Description |
|---|---|---|
| UserID | Integer | Unique identifier for each user |
| Name | String | Full name of the user |
| Email | String | User’s email address for login |
| Password | String | Encrypted password for secure authentication |
| Role | String | User’s role within the system (e.g., staff) |

### 5.2.2 Entity 2: Carbon Footprint

| Attribute Name | Type | Description |
|---|---|---|
| EntryID | Integer | Unique identifier for each carbon footprint entry |
| UserID | Integer | Links the entry to a specific user |
| Date | Date | Date of the activity logged |
| Activity | String | Description of the activity (e.g., commute) |
| EmissionValue | Float | Estimated carbon emissions for the activity |

### 5.2.3 Entity 3: Green Initiative

| Attribute name | Type | Description |
|---|---|---|
| InitiativeID | Integer | Unique identifier for each initiative |
| Title | String | Name of the green initiative |
| Description | Text | Details about the initiative’s goals and tasks |
| StartDate | Date | Date when the initiative begins |
| EndDate | Date | Planned end date of the initiative |

### 5.2.4 Entity 4: Participation

| Attribute Name | Type | Description |
|---|---|---|
| ParticipationID | Integer | Unique identifier for each participation record |
| UserID | Integer | Links the participation on to a specific user |
| InitiativeID | Integer | Links the participation on to a specific initiative |
| Date | Date | Date of staff involvement |

### 5.2.5 Entity 5: Reports

| Attribute Name | Type | Description |
|---|---|---|
| ReportID | Integer | Unique identifier for each report |
| Title | String | Title of the generated report |
| GeneratedDate | Date | Date when the report was created |
| Summary | Text | Overview of findings in the report |