

Asset Management System

Team 2








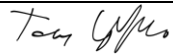
By:

Abed, Callum, Dev, Jay, Mustafa, Parthiv, Satwik and Tom

GitLab Link:

<https://gitlab.cim.rhul.ac.uk/TeamProject02/TeamProject02>

Statement of Relative Contribution:

	Design, Planning, and Coordination %	Coding and Testing %	Other %	Signature
Satwik	12.125	12.1875	13.53125	
Callum	12.125	12.5	12.6875	
Parthiv	11.5625	12.1875	13.53125	
Jay	13.375	12.8125	12.125	
Dev	11.5625	12.1875	12	
Abed	11.875	12.5	12	
Mustafa	13.375	12.8125	12.125	
Tom	14	12.8125	12	

Each member submitted an anonymous forum to submit percentages for each member and category. The displayed percentages were a result of calculating the averages of all percentages each member gave to the group.

Technical Documentation:

In this following section of the report, the planning and design, and technological choices of the project will be discussed in terms of why the decisions made were taken and the impact we predict for it to have in real-time use.

➤ Technology choices:

In the creation of this project, there were 4 major sections in which technology choices could be made and were made. These sections consisted of: Database, Backend framework(s), Backend, and Frontend.

For the database host, we chose to use PgAdmin. Our reasoning for this was that with PgAdmin we were able to see our database in real-time, and it would generate ER diagrams using the tables we had created using SQL. The generating of ER diagrams saved us considerable time in our planning and designing process which was then spent on development.

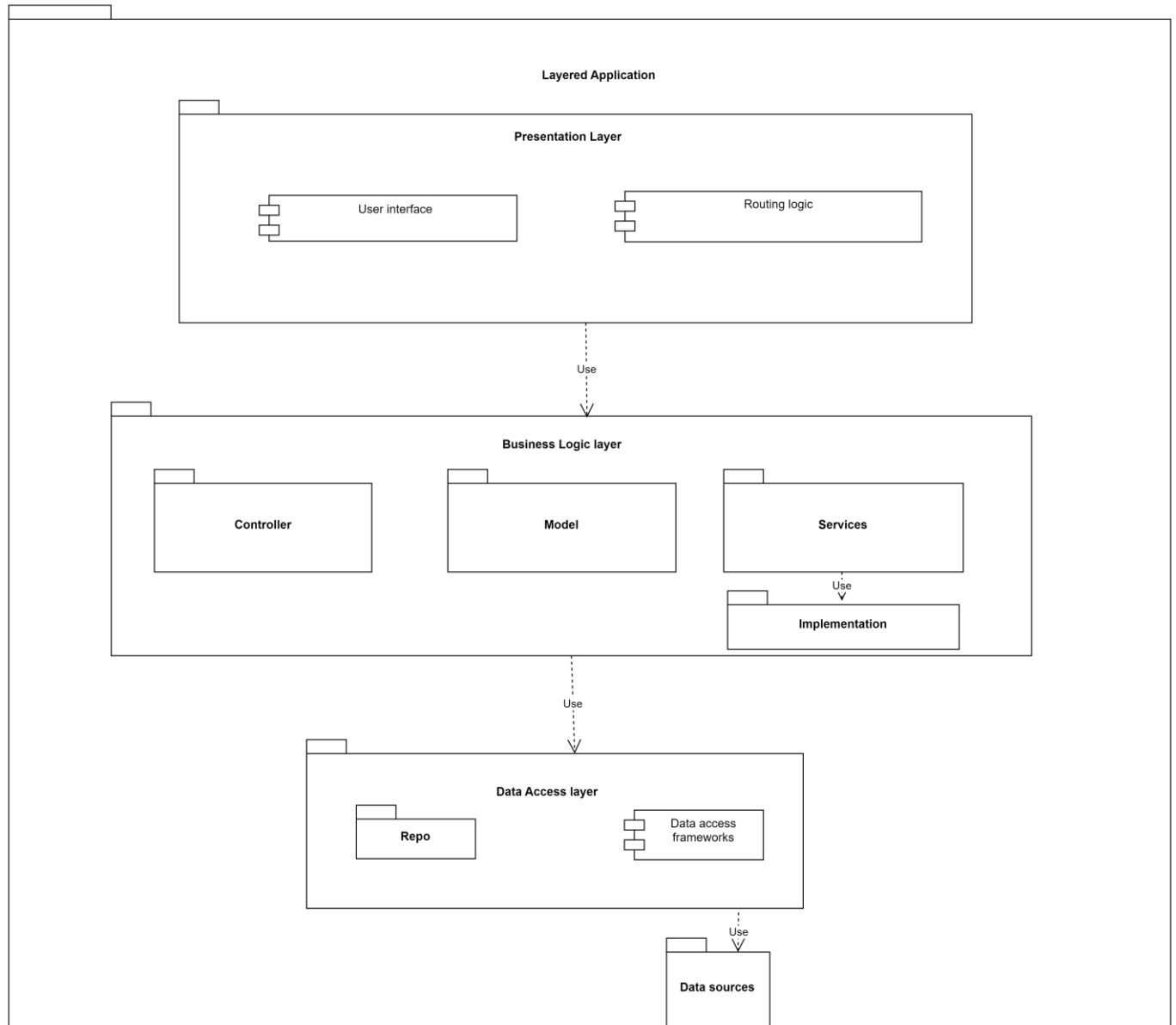
To help us communicate between backend and database, we chose the Springboot API framework. Springboot was chosen as it was industry standard, and it would handle all complications behind the scenes, which allowed us to just jump into the deep end with the development of the website (frontend).

For the coding language of the backend, we chose Java; we chose java as it was the coding language that All members of the group were confident coding in, furthermore it also worked quite well with Springboot and JDBC. Java also provided us with more flexibility in terms of making our own objects to use in the ways we intended to.

In terms of frontend, we chose to use JavaScript with the React library. This allowed us to use a variety of skills from HTML, JavaScript and CSS which made the creation of our website easier. With JavaScript we could incorporate various formats to achieve the ideal look for our website.

In addition to these technologies, we also used POSTMAN to do our entity testing. POSTMAN was used to send entity information to the PostgreSQL database, to test whether data was properly being stored or not, and to see if the database logic was designed correctly.

➤ Architecture:



Description:

Presentation layer:

User Interface: Classes to provide a graphical user interface and display data on the web page. HTML, CSS and JS files

Routing logic: Classes to handle routing from the front-end layer to the backend layer.
JavaScript files.

Business Logic Layer:

Controller:

```
/**
```

This package contains classes for handling HTTP requests related to various functionalities within the project.

These classes utilize Spring annotations and interact with services to perform actions on the system's data. */

Model:

```
/**
```

This package defines various entities used within the application to represent data stored in the database. Each class is annotated with JPA annotations to define the persistence behaviour.

```
*/
```

Services:

```
/**
```

This package defines the services available to interact with data.

Each service method defines functionalities related to CRUD operations, searching, sorting and data retrieval.

```
*/
```

Implementation:

```
/**
```

This package contains implementation classes for the services defined in the cs2815.project.service package. These classes typically interact with the repository layer and perform business logic related to the data entities.

```
*/
```

Data access layer:

Repo:

/**

This package defines repositories for interacting with the database to manage different entities within the application. Each repository extends JpaRepository, providing basic CRUD operations and the ability to define custom queries using JPQL.

/*

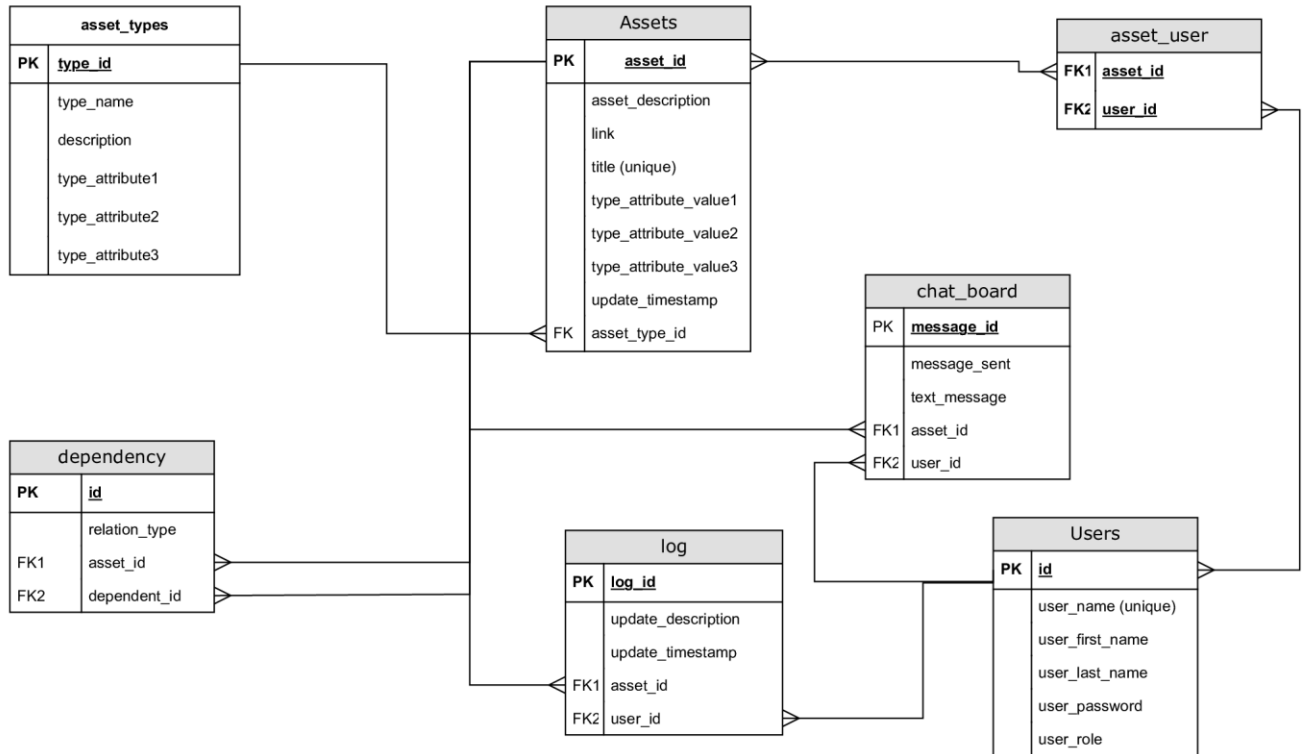
Data access frameworks:

This includes frameworks such as JPA and Spring Data that allow us to access data sources

Data sources:

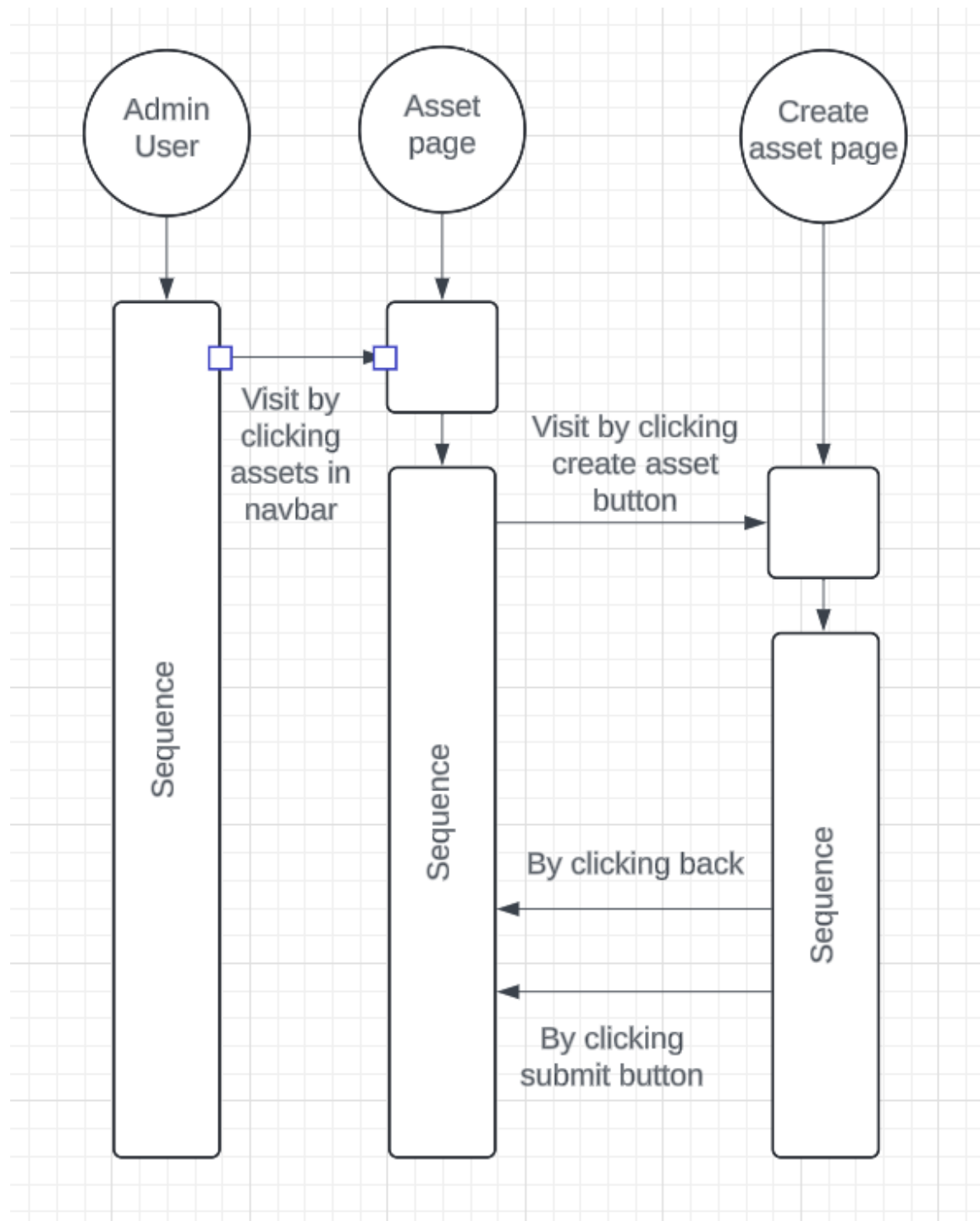
These are the tables that we are accessing throughout the system.

➤ Database Design:



➤ UML Sequence Diagram:

This UML sequence diagram is of an Admin user creating a new asset:



➤ User Stories:

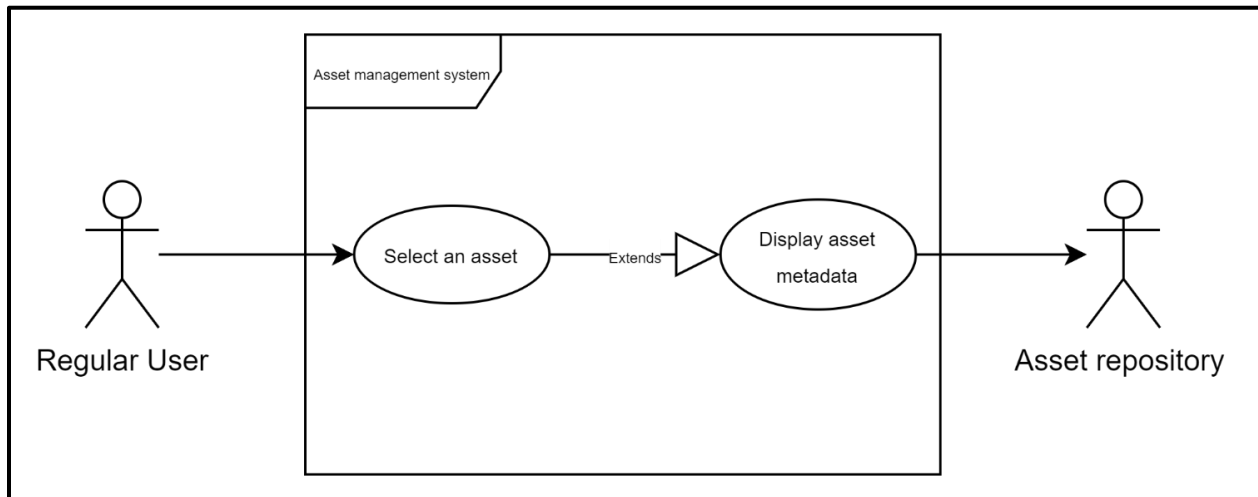
Title: View Asset Metadata

Preconditions: User is logged in and has access to an asset

Flow of events:

1. User selects an asset
2. System displays the asset metadata
3. User can scroll and view all metadata fields

Postconditions: User can see and understand the asset's information.



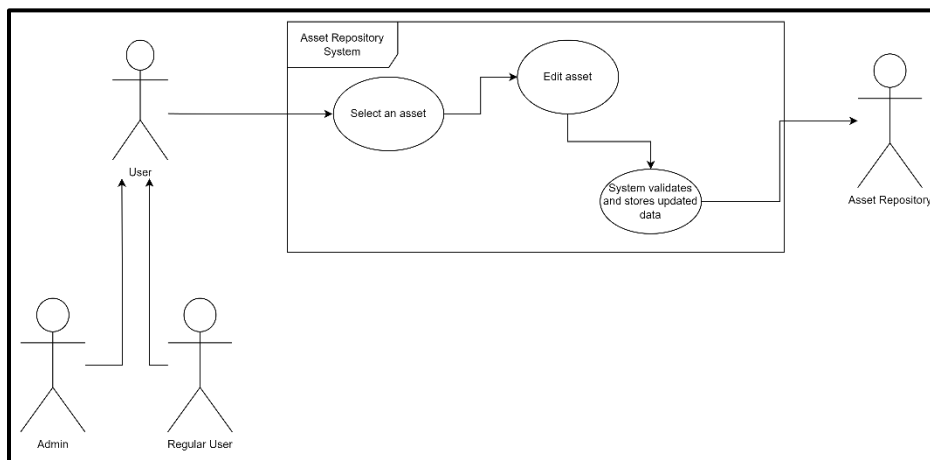
Title: Edit metadata about an owned asset

Preconditions: Any type of user is logged in

Flow of events:

1. User selects an asset and accesses its metadata editing function
2. User chooses the specific metadata data fields
3. User edits the metadata of the selected fields
4. User submits changes

Postconditions: The asset's metadata is updated to reflect user modifications



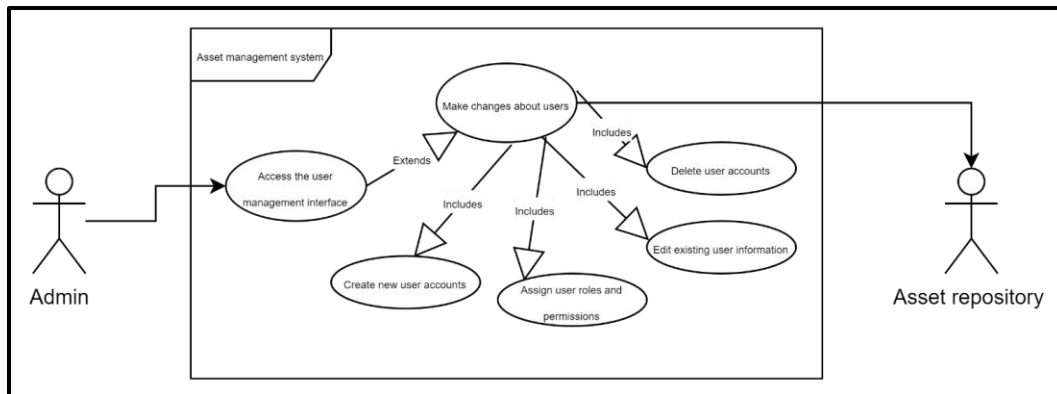
Title: Managing user permissions (inherits all functionality of a Regular User)

Preconditions: Admin user is logged in

Flow of events:

1. Admin User can access user management interface
2. Admin User can:
 - a. Create new user accounts
 - b. Assign user roles and permissions
 - c. Edit existing user information
 - d. Delete user accounts

Postconditions: Admin users can define user access levels and control the system's overall user base.



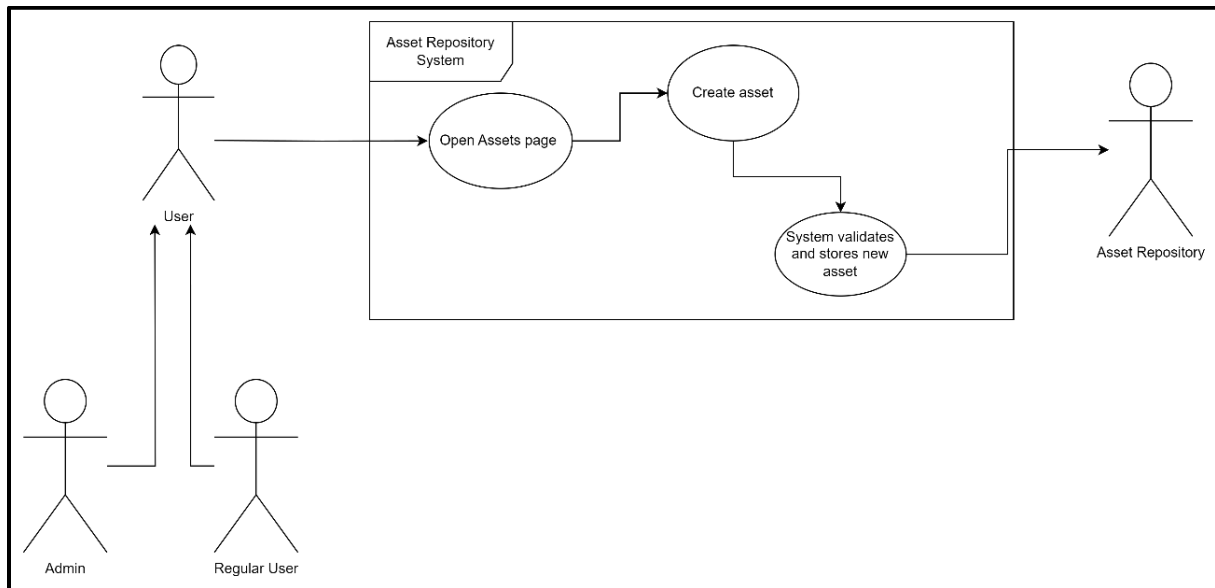
Title: Create asset

Preconditions: Anyone is logged in

Flow of events:

1. User selects an option to create an asset
2. User creates an asset using the form
3. Asset is uploaded to system

Postconditions: Asset is created and added to the database and displayed on the website



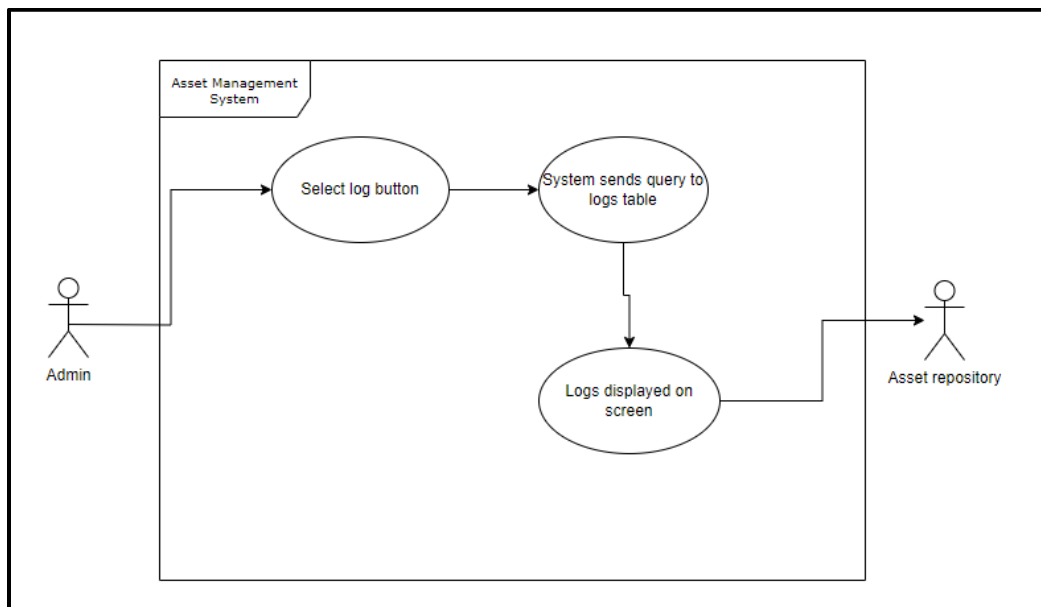
Title: View logs

Preconditions: Admin is logged in

Flow of events:

1. Admin selects option to view logs
2. System queries database table which stores logs
3. System sorts data and displays it on the page

Postconditions: All logs are displayed and can be viewed



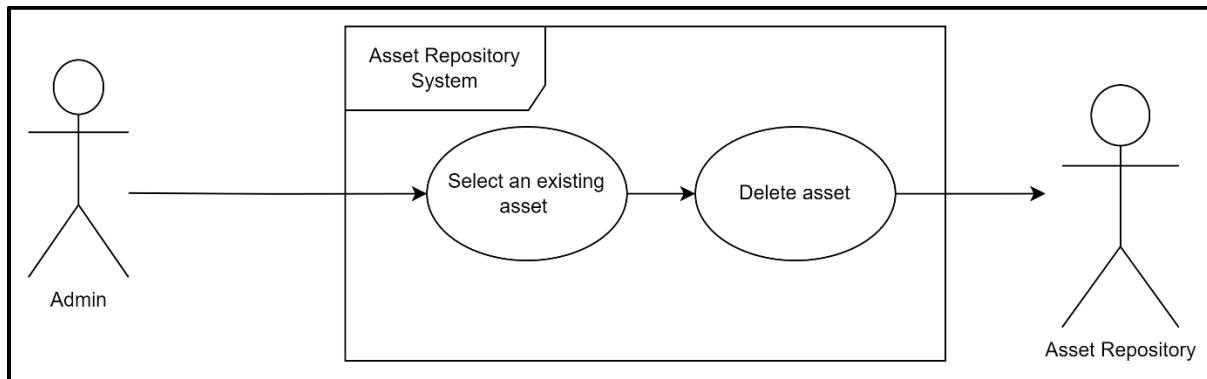
Title: Delete Assets

Preconditions: Admin is logged on

Flow of events:

1. Admin enters assets page
2. Deletes desired asset
3. System saves the change to the database

Postconditions: Asset is deleted from database and no longer displayed on the page



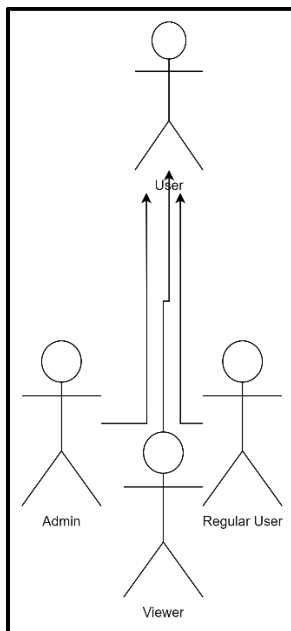
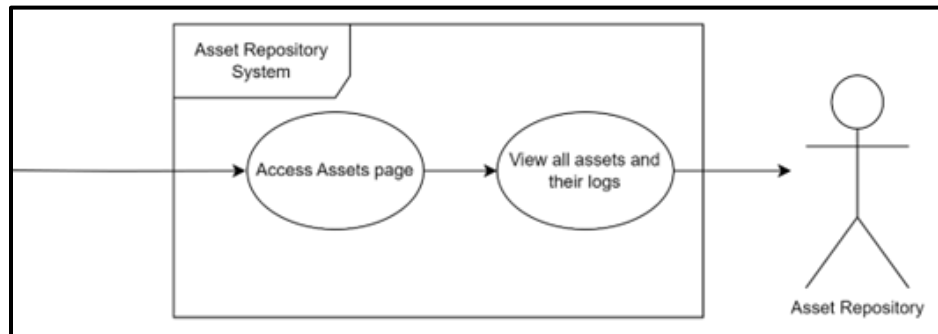
Title: Viewing assets

Preconditions: Anyone is logged on

Flow of events:

1. User enters assets page
2. User can view all the assets

Postconditions: All assets are displayed on the page



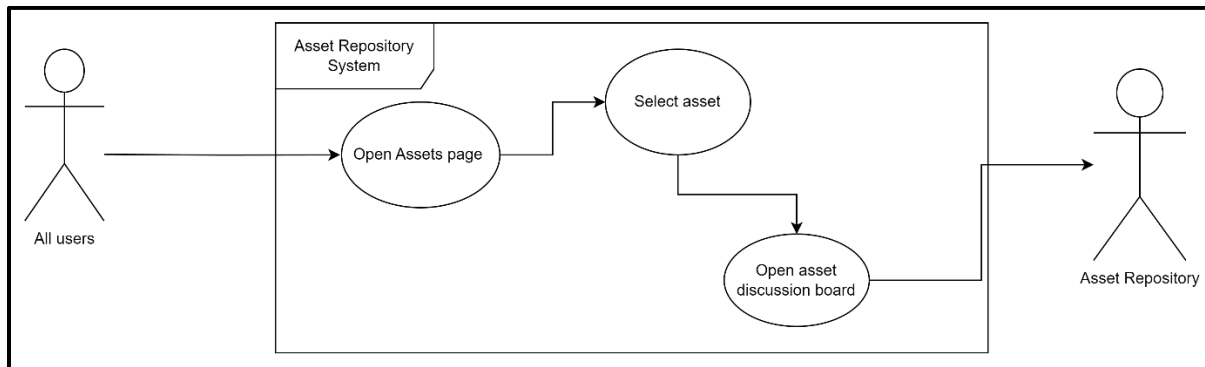
Title: Use discussion board

Preconditions: Anyone is logged in

Flow of events:

1. User logs in
2. Navigates to asset page and selects desired asset
3. User selects discussion board option
4. User can read or send messages on that asset's discussion board

Postconditions: Discussion board pop up will open and can be used



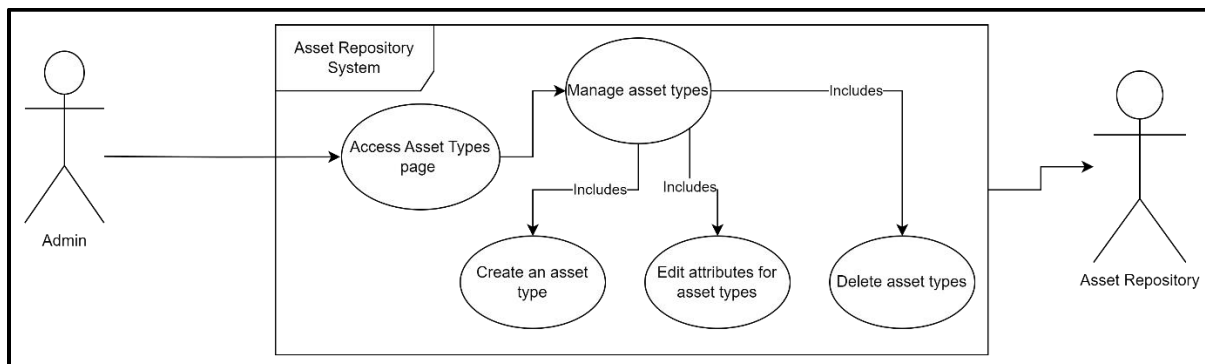
Title: Managing Asset Types

Preconditions: Must be logged in as an administrator

Flow of events:

1. Admin logs in
2. Navigates to asset type page
3. On this page, the administrator can create, delete and edit asset types

Postconditions: The desired asset type will be created, modified or deleted and can be used when creating assets



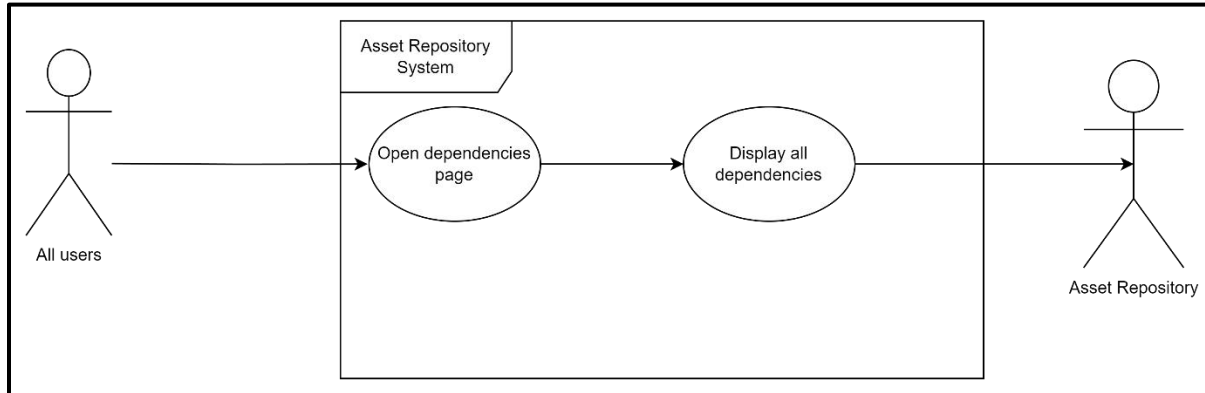
Title: View all dependencies

Preconditions: Anyone must be logged in

Flow of events:

1. User should log in
2. Navigate to the dependencies page

Postconditions: All dependencies are displayed on the screen



➤ Video Demonstration:

<https://www.youtube.com/watch?v=VjkrRPFVkdU>

