

Faculty of Design, Informatics and Business

CMP307

Coursework 1 – part 2

**Date: Thursday, 31 October 2024**

**Victoria Jane Beek**

**2406584@abertay.ac.uk**

Contents

[Class Diagram 3](#_Toc181309478)

[Use Case Diagram and Description 5](#_Toc181309479)

# Class Diagram

A screenshot of a computer

Description automatically generated

Class: `AssetTrackingApp` - This is the main class of the application. It handles everything about managing users and assets, plus it shows the graphical interface.

Attributes:

* `users: HashMap<String, User>`: This stores user accounts, where the username is the key, and the user details are stored in a `User` object.
* `tableModel: DefaultTableModel`: This would manage data for a table, but it's not used right now.
* Input Fields: There are text fields for username, password, and asset details like asset tag, system name, etc.
* `departmentComboBox: JComboBox<String>`: A dropdown menu for selecting a department.

Methods:

* `AssetTrackingApp()`: Initializes the app and sets up the window and database.
* `createNewDatabase()`: Makes a new SQLite database and creates a table for assets.
* `initializeUsers()`: Adds some default users for logging in.
* `createLoginPanel()`: Creates the login screen.
* `createMainPanel()`: Sets up the main asset management screen.
* `createAddAssetPanel()`: Makes the panel for adding new asset information.
* `authenticate()`: Checks if the username and password are correct.
* `addAsset()`: Adds a new asset to the database.
* `clearAssetFields()`: Resets the input fields after adding an asset.
* `logout()`: Goes back to the login screen.
* `switchToPanel(panelName: String)`: Changes which panel is visible.
* `styleButton(button: JButton)`: Styles buttons for a consistent look.
* `main(args: String[])`: The starting point of the application.

Class: `User` - this class represents a user account in the app.

Attributes:

* `username: String`: The user’s login name.
* `password: String`: The user’s password.
* `firstName: String`: The user’s first name.
* `lastName: String`: The user’s last name.
* `email: String`: The user’s email address.

Methods:

* `User(...)`: The constructor to create a new user with details.
* Getter Methods: These methods let other parts of the program get the user's details like their name and email.

Relationships

- `AssetTrackingApp` and `User`: The `AssetTrackingApp` class uses many `User` objects, as it can handle multiple users at the same time.

In summary, the class diagram for `AssetTrackingApp` shows how the program is structured. The main class manages users and assets, while the `User` class keeps track of individual user information. This design helps with user login and asset management in a clear and organized way.

# Use Case Diagram and Description

A diagram of a system

Description automatically generated

In developing the Asset Tracking System, I created a use case diagram that illustrates the interactions between users and the system's functionalities, particularly focusing on the process of adding an asset. The diagram visually represents the relationships between different actors and the key actions they can perform within the system. At the centre of the diagram is the system itself, labelled "Asset Tracking System," which encompasses all functionalities available to users.

The primary actor in this system is the User. This individual represents anyone who interacts with the Asset Tracking System, including employees and administrators responsible for managing assets. Within this context, I identified three critical use cases: Authenticate User, Add Asset, and Logout User.

1. The first use case, Authenticate User, describes the process by which the user enters their username and password to gain access to the system. For this action to occur, the user must possess valid credentials. Once authenticated, the user is granted access to the system's main functionalities, allowing them to perform various asset management tasks.
2. The second use case, Add Asset, details the steps a user takes to input asset information into the system. This includes filling out a form with essential details such as the asset tag, system name, model, manufacturer, and other relevant data. Before this action can take place, the user must be authenticated to ensure that only authorized personnel can add assets. Upon successful submission, the asset is recorded in the database, and the user receives a confirmation message indicating that the operation was successful.
3. Lastly, the Logout User use case allows the user to exit the system securely. For a user to log out, they must first be logged in. After choosing to log out, the system returns the user to the login screen, ensuring that the session is ended appropriately.

Overall, the use case diagram effectively communicates how users will interact with the Asset Tracking System. It highlights essential functionalities such as authentication, adding assets, and logging out, providing a clear understanding of user roles and system capabilities.