

Faculty of Design, Informatics and Business

CMP307

Coursework 1 – part 2

**Date: Tuesday, 05 November 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: A map of usernames to User objects for managing registered users.
* ASSET\_TYPES and DEPARTMENTS: Arrays storing predefined options for asset types (e.g., "Laptop") and departments (e.g., "Finance").

### UI Components:

* loginPanel: JPanel — Panel containing login components (username, password, login button).
* mainPanel: JPanel — Panel that displays main content after login, such as welcome messages and options to manage assets.
* cardPanel: JPanel — Acts as a container for switching between different UI panels, such as login and asset management.
* Additional asset-related UI fields: assetTagField, systemNameField, modelField, manufacturerField, typeField, ipAddressField, purchaseDateField, notesField, typeComboBox, departmentComboBox, etc. — Input fields for each asset attribute and dropdowns for selecting options like department or asset type.
* DEPARTMENTS: String[] — An array listing departments (e.g., Finance, Sales). Used to populate the department dropdown in asset or user management interfaces.
* DATABASE\_URL: String — Stores the path to the SQLite database used for storing asset data.

### Methods:

* `AssetTrackingApp()`: Initializes the app and sets up the window and database, use list and login panel
* `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 and password: The user’s login credentials.
* firstName, lastName, email, and department: Basic information about the user.

### 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 four critical use cases: Authenticate User, Add Asset, Logout User and Display Welcome Panel.

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. The third use case, Logout User, 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.
4. Lastly, the fourth and final use case is to welcome the user; once the user successfully logs in, they are directed to a welcome panel personalized with their name. This panel serves as a central hub, offering the user several navigation options to manage assets and view related information. From here, the user can choose actions such as viewing the asset list, adding a new asset, editing existing assets, or deleting an asset. Selecting any of these options takes the user to the corresponding panel for that action, allowing them to seamlessly move between different tasks within the system.

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.