Design Documentation

UWO MAPS

Group 12 - Design Documentation

Version	Date	Author(s)	Summary of Changes
0.0	12 Oct 2022	Will	Created document. Filled with requirements from OWL.
1.0	19 Oct 2022	Will	Added images for UI mockup
2.0	20 Oct 2022	Hanyun Guo	Finish introduction part
3.0	21 Oct 2022	Hanyun Guo, Xiaohui Sun	Summary part draft version
4.0	21 Oct 2022	Hanyun Guo	Finish file format part
5.0	25 Oct 2022	Will, Yangxiuye Gu	Changed startup/login page
6.0	26 Oct 2022	Yangxiuye Gu , Xiaohui Sun ,@Taysean	Class diagram
7.0	28 Oct 2022	Will	Development environment and UI Mockup
8.0	28 Oct 2022	Yangxiuye Gu, Taysean Wilson, Hanyun Guo	UML Class Diagram - Diagram
9.0	28 Oct 2022	Taysean Wilson, Yangxiuye Gu	ULM Class Diagram - Description
10.0	28 Oct 2022	Will	Addition to summary. Removal of project requirements.

- 1.1. Introduction
- 1.2. Class Diagrams
  - 1.2.1. UML Class Diagram
- 1.3. User Interface Mockup
- 1.4. File Formats
- 1.5. Development Environment
- 1.6. Summary

1.1. Introduction

Overview

As buildings are often spread over a large sprawling campus, navigating a university can be a daunting task. Even though smartphones, GPS, and mapping services like Google Maps are helpful for moving around outdoors and locating buildings, these services are inadequate for navigating interior spaces. As buildings are typically composed of multiple floors with different layouts that are hard to represent in a single, flat, 2D map.

The University of Western Ontario has made the floor plans of all of its buildings available to the public to assist people with navigating interior spaces. While the primary use case is to identify accessible features of the campus and its buildings to those in need, the maps available through this service can be useful to anyone wanting to locate a particular room in a particular building. Unfortunately, the maps are provided as PDF files with no metadata which makes them readily searchable or easy to use. Furthermore, while a layer of accessibility is baked into each map, there are opportunities for other points of interest and other useful data to be layered onto the maps that are not explored.

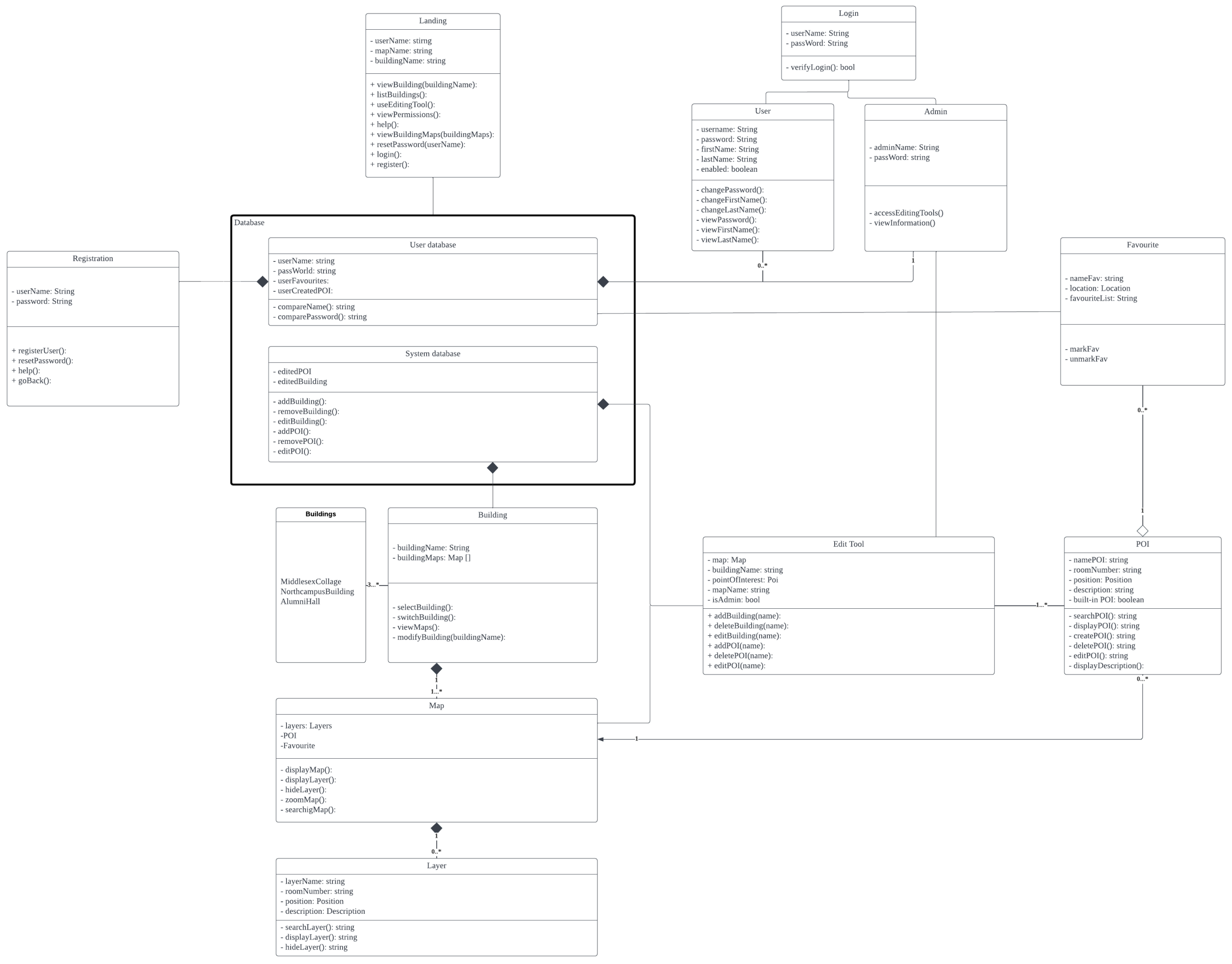
The main purpose of this project is to create an application that leverages the maps made available by Western to allow users to search and explore its interior spaces. The application will enable users to browse three campus buildings supported by the application, search for a points of interest in the building, create and save points of interest, mark or unmark their favorite points of interest, and view the current weather on campus. The application also has an accompanying editing tool to facilitate the creation and editing of map metadata by developers for the application.

Objective

- applying the principles of software engineering to a real-world problem
- working with, interpreting, and following a detailed specification
- creating models of requirements and design from such a specification
- implementing the design in Java and having to deal with decisions made earlier in the design process
- creating graphical, user-facing content and applications
- writing robust and efficient code
- write good, clean, well-documented Java code that adheres to best practices
- reflecting on good/bad design decisions made throughout the project

1.2. Class Diagrams

1.2.1. UML Class Diagram



Class: Building	
The Building class represents all three buildings with names and maps. Administrators could add, edit and change the buildings in the system. Users could switch the buildings and browsing the map	
Responsibility:	**Collaborator:**
Assigned building name	
Presented building map	
Switched Building	User
Add another building	Administrator
Remove another building	Administrator

Class: Map	
The class map provide details information on the map of different buildings.	
Responsibility:	**Collaborator:**

Presenting POI,layers, favorites and classroom	
Searching POI	Users
Show/Hide layers	Users
Browsing the building map	Users
Scaling the map	Users

Class: Layer	
The class Layer represents washrooms and a layer for each type of point of interest (e.g. classrooms, restaurants, labs, user defined points of interest, etc.). When a user would like to display the layer, there will be a pin on the map for highlighting.	
Responsibility:	**Collaborator:**
layer for accessability	
layer for POI	
displayed or hided layer	user
Edited layer	Administrator

Class: POI	
The class POI allow users to mark and unmark	
Responsibility:	Collaborator:
came with built-in POI	
User Created POI	User
Added description of POI	User
display POI on the map	User
Select POI from POI list	User
Edited buit-in POI	Administrator

Class: Favourite	
The class favourite allows users to mark and unmark built-in POI as favourites for quick access. POI marked as favourites can be accessible through some sort of menu or list; when selected, the corresponding location is shown on the appropriate map, just as if the user had searched for the location manually. This would allow users to store the locations of all of their classrooms, for example, for easy access.	
Responsibility:	**Collaborator:**
Marked POI as favourite	User
Unmarked POI as favourite	User
Select favourite from favourite list	User

Class: Login	
The class Login will permit general users or administrators to login to an existing account.	
Responsibility:	**Collaborator:**
Login to a user or administration account	User/Admin

Class: Landing
The Landing class has direct access to both the System Database and the User Database. The Landing class will permit User Login and Registration and determine the permissions the logged-in user has and how they interact with the application. At this page, users will be able to view the buildings, select a building, or view the maps that a building contains before accessing the building.

Responsibility:	**Collaborator:**
Respond to valid or invalid username and passwords	
Permit user/admin sign ups	User/Admin
Permit user/admin logins	User/Admin

Class: User	
The User class stores information for a singular general user with restricted permissions in the application. A user will have a first name, last name, username and password. They will be able to access Buildings, Points of Interest (POI) and Maps; but users will not be able to modify build-in data by the Administrator.	
Responsibility:	**Collaborator:**
Marked POI as favourite	User
Unmarked POI as favourite	User
Create user-defined POI	User
Modify user-defined POI	User

Class: Admin	
The Admin class stores information for a singular Administration user with access to the System's Editing Tools. An administrator will have a admin name and password.	
Responsibility:	**Collaborator:**
Access Editing Tools	Administrator
Add data to the System Database	Administrator
Remove data from the System Database	Administrator
Modify data in the System Database	Administrator
Add users to the User Database	Administrator
Remove users from the User Database	Administrator
Modify user information inside the User Database	Administrator

Class: Registration	
The class Registration allows users to create an account with a unique account name.	
Responsibility:	**Collaborator:**
Registered a user account	User/Admin

Class: User Database	
The database that stores, maintains, and modifies data (information) on the System's Users and Administrators. This information is related to user permissions, as well as storage of account passwords and usernames. Permissions are primary related to the Editing Tool which permits full access to the development of the System Database.	
Responsibility:	**Collaborator:**
Store user's information created POI and Favourite list.	User

Class: System Database	
The database that stores, maintains, and modifies data (information) that pertains to the System's Maps. The Maps will be stored inside of databases called Buildings. Each map will store layers that define its respective areas. Maps will also consist of Points of Interest (POIs) that can be user-defined or build-in by the Administrator. User permissions will determine whether the User or Admin can access the Editing Tool which enables them to modify the data that is stored inside the System Database.	
Responsibility:	**Collaborator:**

	Administrator

Class: Editing Tools	
The Editing Tools class will be accessible only by the System's Administrator. The Editing Tools will provide functions to add, delete, and modify information in the System Database. This class will permit the Administrator to make changes to the System's Buildings, Maps, Layers, Points of Interest (POI), and Favourite POIs.	
Responsibility:	**Collaborator:**
Modifying System Database	Administrator

1.3. User Interface Mockup

The user page will be displayed after someone logs into their account. The task bar displays ways to reach certain options such as changing their password, enabling editor mode, and accessing the help menu. This is the place where most would look to find this sort of information.

The primary goal of the user page is to allow access to information with as few steps as possible. The left hand side of the screen displays all the sorting options for the user where they can select the building, floor, POI type, and finally the POI. Creating one centralized location for all search options reduces the user's memory load so they can simply see what they need. The available POIs will update as the user selects different sorting requirements.

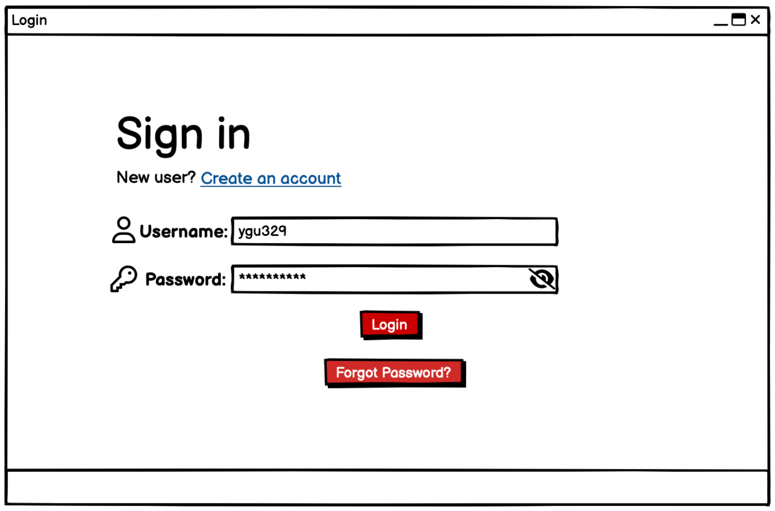
When a user selects a POI the map will recenter itself to that location and will display relevant information in the top left of the map. The goal is to clearly show the user the location on the map as well as any other relevant information.

Users can enter editor mode through clicking the on screen button or through the taskbar menu. The graphic in the bottom right corner is the make is as easy as possible for the user to access and to be aware of what mode they are in. When the user is in editor mode they can click on the map and it will open a menu for a new POI where they are able to enter the required information. Users in editor mode can also delete any user-created POI by selecting that POI through the selection menus then clicking “remove”. They must be in editor mode to do so.

Admins and developers are presented with a very similar but different UI. Admins will have access to edit all aspects of existing POI. In addition they will be able to add new buildings and floors through the edit taskbar menu. Admins will create a new building then will create floors and upload the map files for each. The admins can then, much like users in edit mode, click on the map to create a new POI.

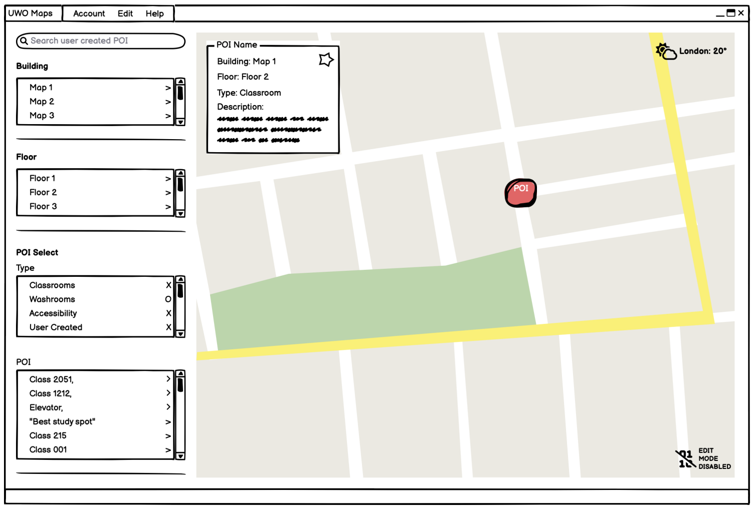
CS2212 Mockup.bmpr

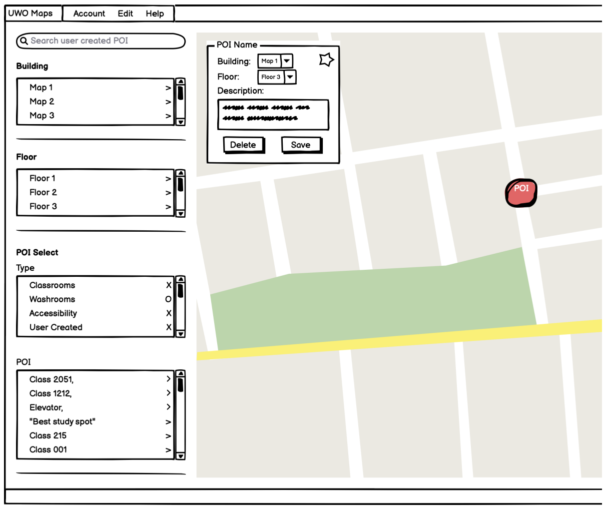
Startup Login



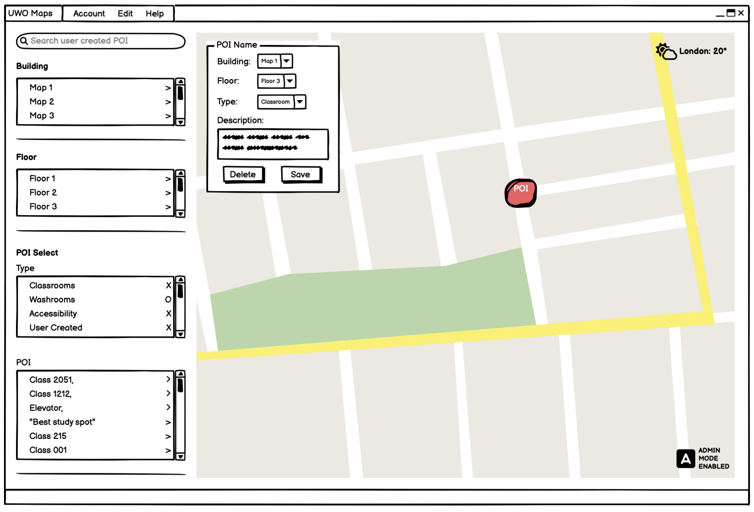
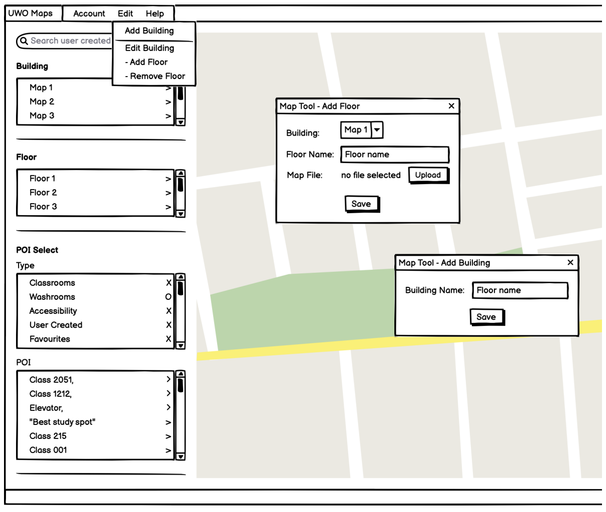
USER

USER (Default Mode)	**USER (Editor Mode)**





ADMIN

ADMIN (POI Edit)	ADMIN (MAP Edit)
	

1.4. File Formats

The Points of Interest metadata will be stored as JSON files. All the Points of Interest in one building are stored in the same JSON file. Therefore, there are three JSON files in total. The metadata for each Point of Interest will include:

1. A Short description of the Point of Interest
2. Name or room number label that is searchable
3. Position on the map (X, Y coordinates)
4. Point of Interest type (e.g. classroom, washroom, etc.)
5. The name of the building (i.e. the Point of Interest is located in which building)
6. The floor number

The map files will be converted to image files in our application. All the accessibility information will be kept in order to provide users with the maximum map information that contains in the application. The application will contain information of the three buildings:

1. Middlesex College
2. North Campus Building
3. Alumni Hall

1.5. Development Environment

Development will be done using Apache NetBeans IDE 15 and JDK 19. Swift and the built-in UI creation tool in NetBeans will be used to create the UI. OpenWeatherMap API and free student account will be used for the current weather extra feature; the internal library java.net.HttpURLConnection will be used to handle this. Photoshop will be used to convert the map PDFs into standardized images. An external library Gson will be used to convert JSON files to java objects and back. Jira will be used for project management. Bitbucket will be used as the code repository.

1.6. Summary

In the Class diagram, we defined sensible classes with attributes, methods, and descriptions.

1. Building: The Building class represents all three buildings with names and maps. Administrators could add, edit and change the buildings in the system. Users could switch buildings and browse the map
2. Map: The class map provide details information on the map of different buildings.
3. Layer: The class Layer represents washrooms and a layer for each type of point of interest. When a user would like to display the layer, there will be a pin on the map for highlighting.
4. POI: The class POI represents users to designate a point on the map and provide a name and description at a minimum. Our application comes with various types of points of interest (POI) built-in that can be separately layered over the map when browsing or searching. The user should be able to choose which POI is being shown by displaying and hiding layers.
5. Favourite: The class Favourite allows users to mark and unmark built-in POI as favourites for quick access.

The user interface is designed to make the experience of finding and creating POI as simple as possible. The section lists the reasons certain design decisions were made and what benefit they hope to provide the user. The UI mockup images give a broad scope of what the application is expected to provide to the user.

The developmental framework remains under review, as the code base progresses, additions will be made. So far the IDE, JDK, libraries and API that are to be use are listed and seek to guide development but are not strict requirements.

In the file format section, we specify that we will use JSON files and create one file per building. There are six pieces of metadata information stored in each Point of Interest, including the type of Point of Interest, the Short description of the Point of Interest, etc.