Design Documentation

1.1. 1.1. Design Documentation

Authors: David Alter, Ryan Cook, Dayton Crombie, Taoran Gong, Minsoo Park

1.1.1. 1.1.1. Revision history:

Version	Date	Author(s)	Summary of Changes
1.0	Mon, Oct 24th, 10:00 am	David Alter	Created original document, created headings for individual sections of the document as well as table of contents
1.0.1	Mon Oct 24th, 10:45 am	David Alter	Inserted first draft of development environment section (yet to be reviewed by group)
1.0.2	Tues Oct 25th, 7:30 pm	Dayton Crombie	Added first draft of user-interface mockup
1.0.3	Wed Oct 26th, 4:30 pm	Minsoo Park	Added draft of class diagram
1.0.4	Wed Oct 26th, 4:30 pm	Taoran Gong	Made changes to the class diagram
1.0.5	Thursday	Ryan Cook	Inserted first draft for the file format
1.06	Thur, Oct 27th, 2:30 pm	David Alter	Overlooked file formats, made some changes when started to assist Taoran and Minsoo on Class Diagrams.
1.07	Oct 27, 4:55pm	Taoran Gong	Uploaded first draft of Class Diagrams
1.0.8	Thur, Oct 27th, 6:00 pm	Dayton Crombie	Uploaded interactive Balsamiq mock up file
1.09	Thur, Oct 27th, 6:15 pm	Dayton Crombie	Added introduction
1.0.10	Thur, Oct 27th, 6:23 pm	David Alter	Added some inheritance to the class diagram
1.0.11	Fri, Oct 28th, 9:00 am	Minsoo Park	Added CRC Cards
1.0.12	Fri Oct 28th 9: 40 am	Taoran Gong	Added the final version of class diagram
1.0.13	Fri Oct 28th 12:25 pm	David Alter	Looked over final changes for the document
1.0.14	Fri Oct 28th 1:45 pm	Ryan Cook	Wrote the Summary

1.1.2. Table of contents

1.2. **Introduction:**

1.2.1. Overview:

Navigating a university campus can be a difficult task, with buildings being spread over a large campus and often seeming like mazes once inside. Finding locations outside has been made much easier through smartphones, GPS, and mapping services like Google Maps which can at least help locate buildings without much difficulty. Such products, however, often do not do an adequate job of interior spaces where the map data is incomplete or difficult to work with, as buildings are typically composed of multiple floors with different layouts that are hard to represent in a single, flat, 2D map.

At Western, to assist people with navigating interior spaces, the university has made the floor plans of all of its buildings available to the public. While the primary use 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 simply provided as PDF files with no metadata which makes them readily searchable or easy to use. It is also difficult to determine routes using these maps due to the nature of buildings having multiple floors. Furthermore, while a layer of accessibility is incorporated into each map, there are opportunities for other points of interest and useful data to be layered onto the maps that are not.

The main purpose of the development of this product is to create an application that utilizes the maps made available by Western to allow users to search and explore its interior spaces. At a high level, our application will allow users to search for rooms in buildings, locate points of interest in the buildings, browse through maps provided and create and save their own personal points of interest. Our application must also have an accompanying editing tool to allow for the creation and editing of map metadata by developers for the application. An extra feature we decided to add is that the application will also include classroom metadata that will provide details about certain classrooms.

1.2.2. Objectives:

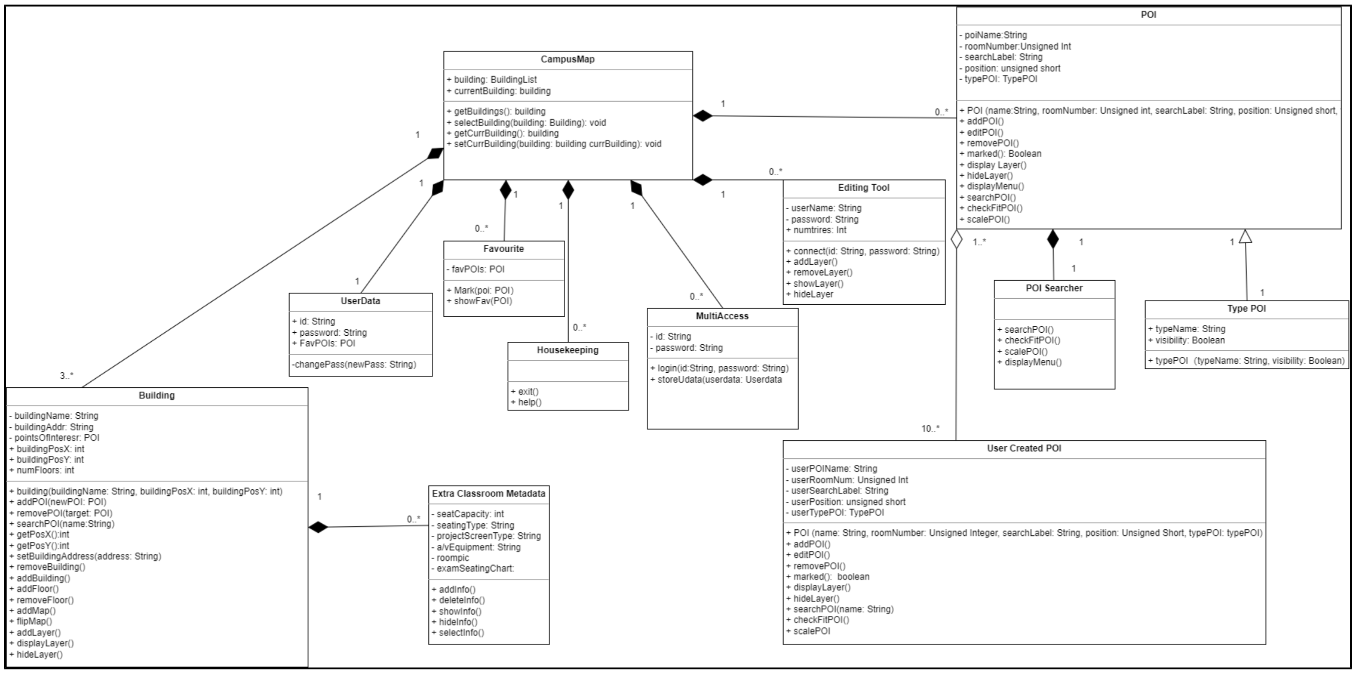
- Create an application that makes it easier for users to navigate campus buildings utilizing the PDF maps Western provides
- Create a graphical user interface that is easy for the customer to use
- Ensure an application that operates smoothly and quickly for the user by writing robust and efficient code
- Fulfill all the project specifications set by the stakeholders (professors)
- Write good, clean, well-documented Java code that adheres to best practices

1.2.3. References:

- CS2212 Group Project Specification, Version 2.0, Fall Session 2022
- Requirements Documentation, CS2212 Group 9, Fall Session 2022

1.3. **Class Diagrams:**

1.3.1. UML Class Diagram



1.3.2. CRC Cards

Class: CampusMap

Basis of map which displaying each building.	
Responsibility:	**Collaborator**:
Associated with each building.	Building
Stores building data list	Building

Class: Building	
Stores metadata of each building. This also including layer system.	
Responsibility:	**Collaborator:**
Sets metadata (floors, positions) on building	
Edits (add, remove) building	
Stores POI data	POI

Class: Extra Classroom Metadata	
Puts extra metadata on classroom	
Responsibility:	**Collaborator:**
Edits information	
Stores data (seat capacity, seat type, room pic, etc..)	POI

Class: POI	
Displays users and developers interesting area with some of information.	
Responsibility:	**Collaborator:**
Stores data on POI	Type POI
Edits (add, remove) POI	
Searches listed POI	POI Searcher

Class: POI Searcher	
POI searcher which includes list of POI	
Responsibility:	**Collaborator:**
Check the fits of displayed POI	POI, CampusMap

Class: Type POI	
Stores each type of POI.	
Responsibility:	**Collaborator:**
Stores type name and visibility	POI

Class: User Created POI	
Specific class of POI which created by user.	
Responsibility:	**Collaborator:**

Set type POI as User Type POI	Type POI
Stores data on POI	
Edits (add, remove) POI	

Class: Favourite	
Marks POI as favourite and displays on the map	
Responsibility:	**Collaborator:**
Marks the POI	POI
Shows on the map	CampusMap

Class: HouseKeeping	
Helps user to quit application and contacts to system developer.	
Responsibility:	**Collaborator:**
Quits the application	
Connects users with staffs	

Class: Multi access	
Supports multi users to connect application on same time.	
Responsibility:	**Collaborator:**
Provides login systems	
Stores user data	UserData

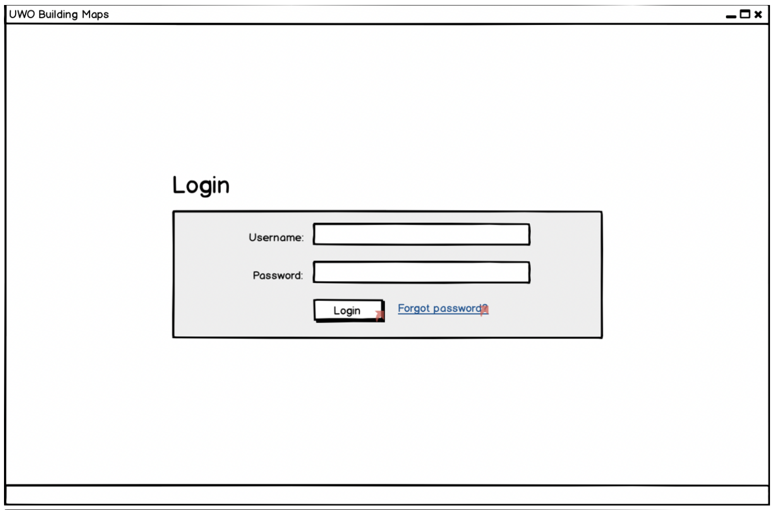
Class: UserData	
Stores User data and manages ID and passowrd	
Responsibility:	**Collaborator:**
Store all users' information	
Change password if user request	Multi access

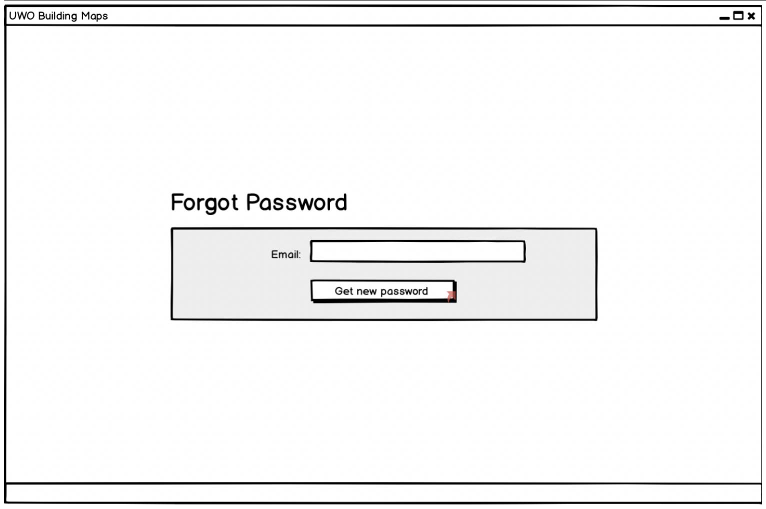
Class: Editing Tools	
Provides efficient tools to design application better.	
Responsibility:	**Collaborator:**
Requires special authorization for using tools.	

1.4. **User Interface Mockup:**

Login Page

Upon opening the application the user will be prompted to login into their account. This is required as each user can have different favourited points of interest. If the user has forgotten their password they can request to change it and be brought to a password reset window.

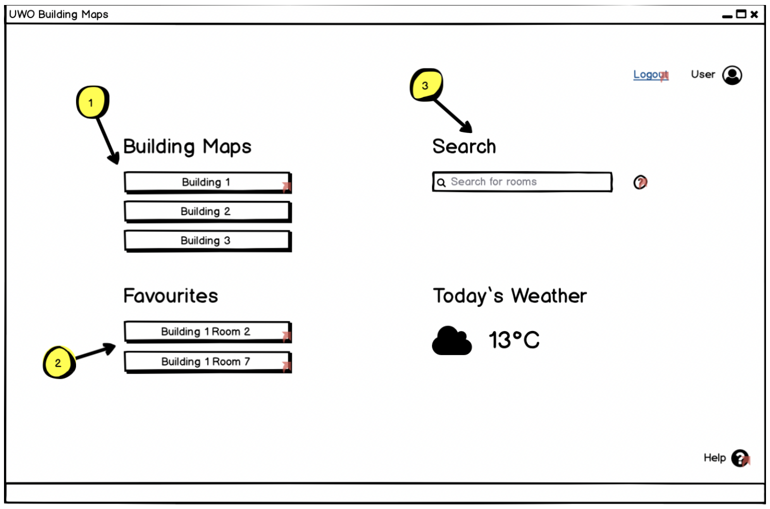




Home Page

After successfully logging in the user will be brought to the home page of the application. From here they can:

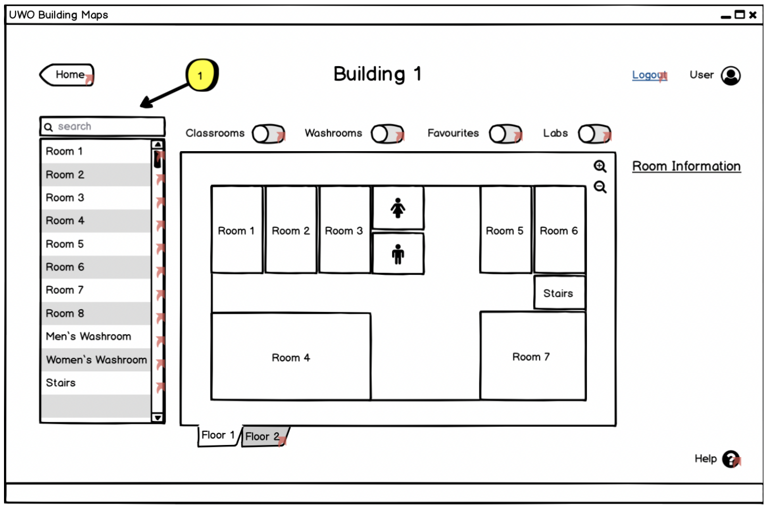
1. Choose a building they would like to view the map for and be brought to that building's specified map
2. Choose to view a room from their list of favourites
3. Search for points of interest by room number or name

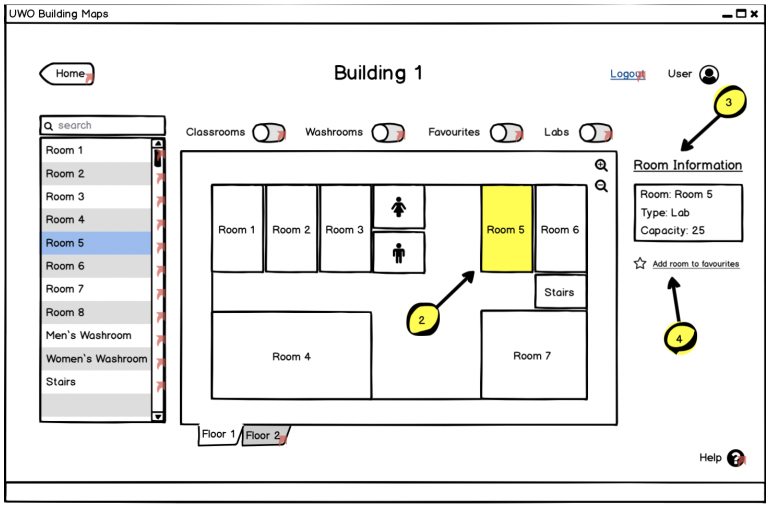


Building Page

Users can select a classroom from the list of all rooms in the building

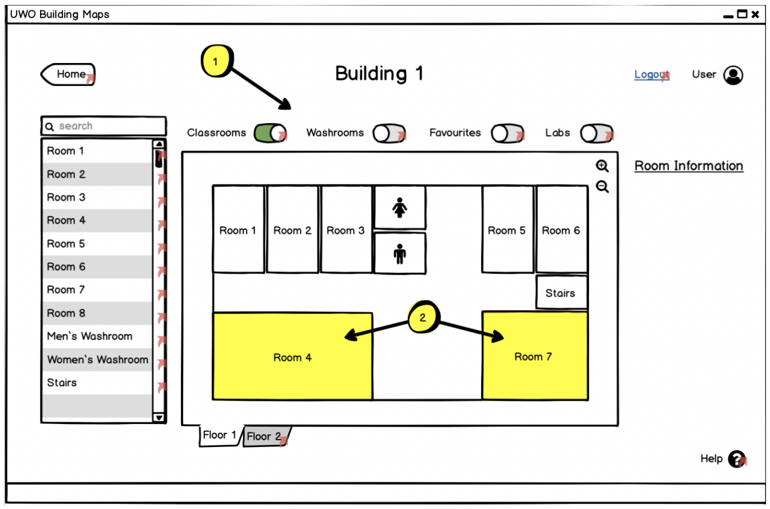
1. List of all rooms in the building, rooms can be selected by clicking on the name
2. Once a room is selected it will be highlighted on the map
3. The rooms metadata will be displayed
4. The room can be added to favourites if not already in the list of favourites





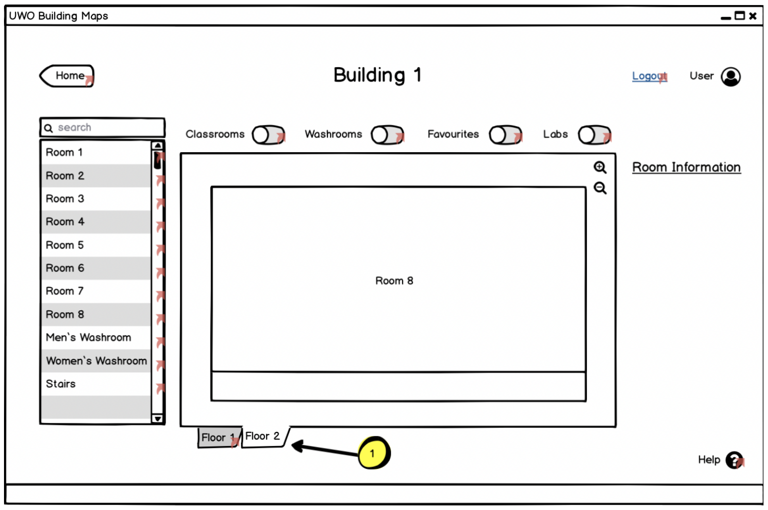
Users can toggle to highlight only certain types of rooms

1. Toggles to filter room types
2. Rooms of the specified type will be highlighted



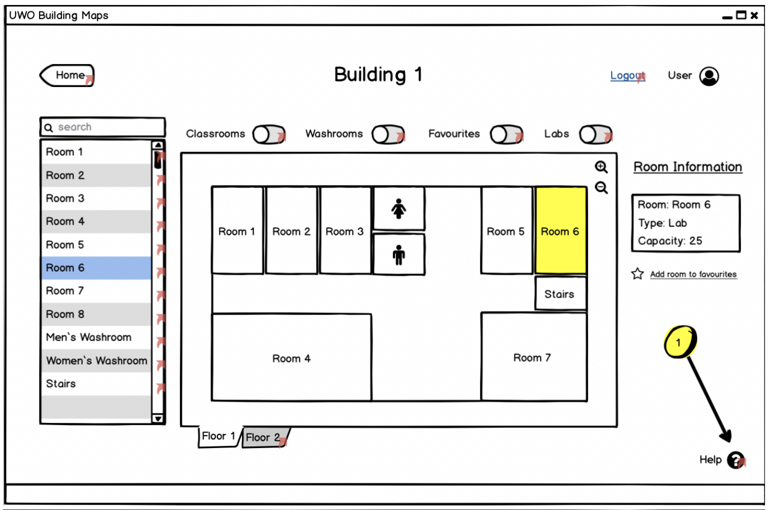
In the case that a building has multiple floors, the user will be able to toggle between viewing each floor

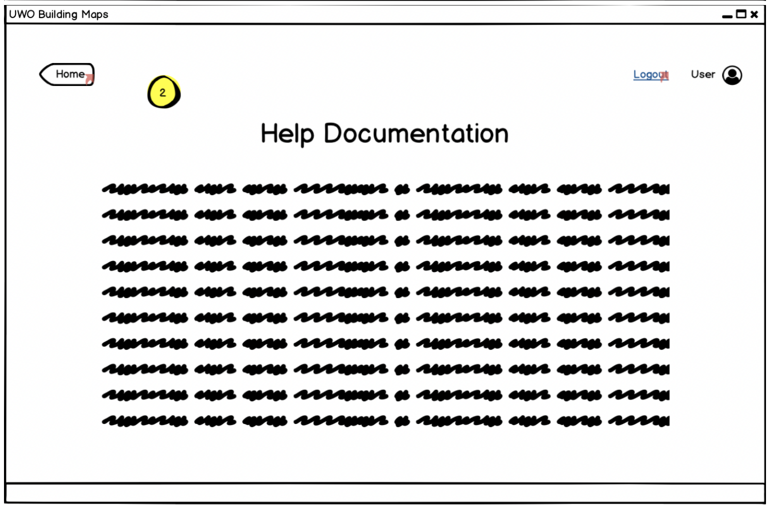
1. Buttons to toggle between floors



If a user needs help on how to use the application they can access the help documentation from any page

1. Button to access help documentation
2. Help documentation





Balsamiq Interactive Mockup File

WireframesMockup.zip

1.5. File Formats

1.5.1. File Format Choice

For this project, I think we are going to JSON as the file format to store point of interest. Since we are storing a small amount of data that we are inputting ourselves, a CSV or TSV file is simply not required. JSON files are much easier to import into Java which is the programming language that we are using for this project. JSON is also simpler and easier to use than XML and since we are using JSON only for storing, updating, and adding data, it is sufficient for our project. JSON is also faster than XML because it is specifically used for data exchange whereas XML is a language with a lot more features which is not required for this project.

1.5.2. Using JSON

When using JSON, we are going to use an external library called JSON.simple. JSON.simple is a Java toolkit which will make it easier to pass JSON data to Java. Since JSON stores data in object notation, all our data will be stored within objects as well. Here's a list of some the POIs we will store:

For points of interest (POIs), we want to store:

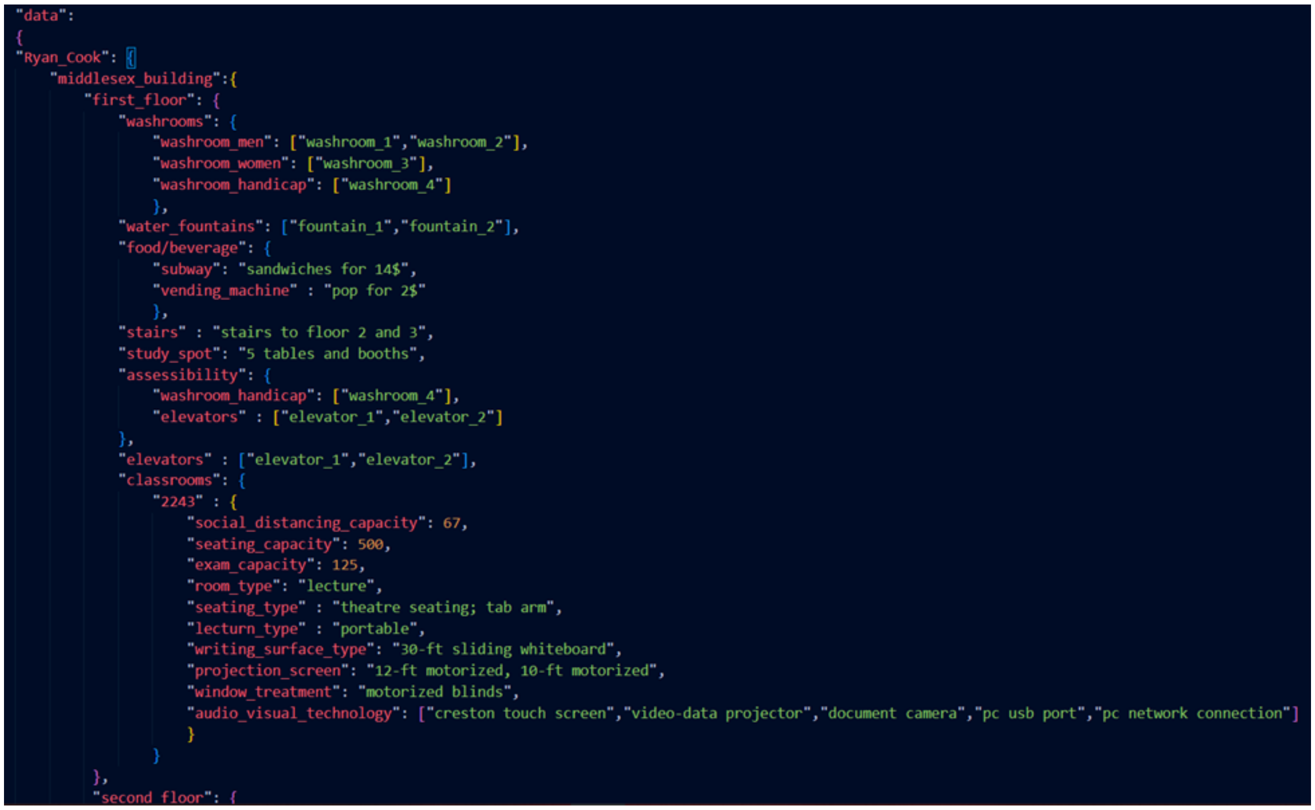
- Washrooms
- Water fountains
- Food/Beverage locations
- Classroom Metadata
- Stairs
- Accessibility
- Elevators
- Study Spots
- Other miscellaneous items

Within classroom metadata, we want to store things like:

- Social Distancing Capacity
- Seating Capacity
- Exam Capacity
- Room Type
- Seating Type
- Type of lectern
- Type of writing Surface
- Type of projection screens
- Type of window treatment
- Audio-visual technology

For example, let's say we are storing data for the user "Ryan Cook" in the JSON code below:

Let's say there is a subway on the first floor of Middlesex, we will store it within JSON like:

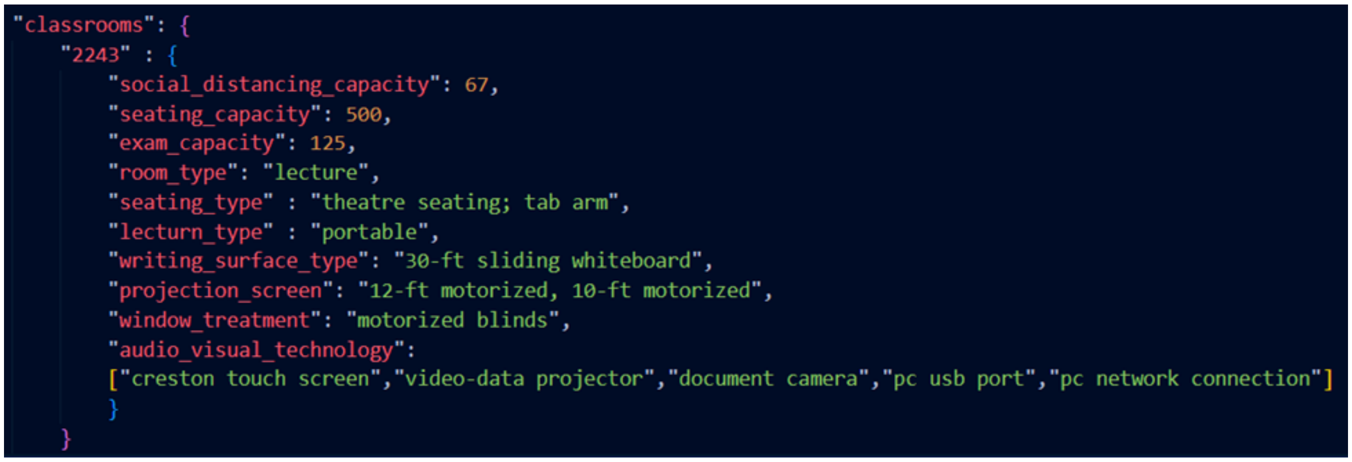


```
"data":
{
"Ryan_Cook": {
    "middlesex_building":{
        "first_floor": {
            "washrooms": {
                "washroom_men": ["washroom_1","washroom_2"],
                "washroom_women": ["washroom_3"],
                "washroom_handicap": ["washroom_4"]
                },
            "water_fountains": ["fountain_1","fountain_2"],
            "food/beverage": {
                "subway": "sandwiches for 14$",
                "vending_machine" : "pop for 2$"
                },
            "stairs" : "stairs to floor 2 and 3",
            "study_spot": "5 tables and booths",
            "assessibility": {
                "washroom_handicap": ["washroom_4"],
                "elevators" : ["elevator_1","elevator_2"]
            },
            "elevators" : ["elevator_1","elevator_2"],
            "classrooms": {
                "2243" : {
                    "social_distancing_capacity": 67,
                    "seating_capacity": 500,
                    "exam_capacity": 125,
                    "room_type": "lecture",
                    "seating_type" : "theatre seating; tab arm",
                    "lecturn_type" : "portable",
                    "writing_surface_type": "30-ft sliding whiteboard",
                    "projection_screen": "12-ft motorized, 10-ft motorized",
                    "window_treatment": "motorized blinds",
                    "audio_visual_technology": ["creston touch screen","video-data projector","document camera","pc usb port","pc network connection"]
                    }
                }
        },
        "second floor": {
```

- Ryan Cook
  - Middlesex building

            - First Floor
                - Food/Beverage
                    - Subway

Or if we are looking at the meta data for a specific for the arbitrary classroom called “classroom 2243”:



```
"classrooms": {
    "2243" : {
        "social_distancing_capacity": 67,
        "seating_capacity": 500,
        "exam_capacity": 125,
        "room_type": "lecture",
        "seating_type" : "theatre seating; tab arm",
        "lecturn_type" : "portable",
        "writing_surface_type": "30-ft sliding whiteboard",
        "projection_screen": "12-ft motorized, 10-ft motorized",
        "window_treatment": "motorized blinds",
        "audio_visual_technology":
        ["creston touch screen","video-data projector","document camera","pc usb port","pc network connection"]
        }
    }
```

- Ryan Cook
    - Middlesex building
        - First Floor
            - Classrooms
                - 2243
                    - Property 1
                    - Property 2 and so on…

1.5.3. Handling Maps

When handling maps, we want to store the maps as images saved as PNG files because we will use transparency to have POIs layered on top of the maps. These images will be stored and used in Java Swing.

1.5.4. Accessibility Information

We will also store accessibility information like the location of elevators, handicap washrooms and other POIs. There will be a specific object (“accessibility”) which will show all the accessibility information on a specific floor.

1.6. **Development Environment:**

In order to complete this project in the most efficient way possible given the timeframe, technically, it is crucial that all members of the development team are on the same page when it comes to the coding environment, tools, and libraries we will be using to turn our ideation into a computer program. Throughout the development cycle, we must also keep in mind that this Java desktop application must run on a Windows 10 system and Javadoc must be used to document our code.

1.6.1. Choice of Integrated Development Environment (IDE)

When it comes to development in Java, some of the most important things to consider in a capable IDE are extensive debugging abilities, speed when developing large applications, and access to plugins and extensions. For these reasons, we could not go with the world's second most popular IDE, VS Code, because of its limited debugging capabilities. So, we needed to find a different IDE which ticks all of the above requirements, after some discussion between Eclipse and IntelliJ, we have ultimately chosen to use **Eclipse for scripting, and NetBeansIDE for GUI testing**. This decision came mainly by the development team's positive past experience with Eclipse, as well as its wide array of plugins.

1.6.2. Tools we will use

We will be using the following software tools to aid us in the development of this project:

- Eclipse IDE for development
- Swing Framework for GUIs
- IDE compilers and debuggers
- Balsamiq software for GUI mockups
- Unit testing to test our code

1.6.3. Additional Required External libraries

Other than the swing framework, we also also need an additional external library to be able to use our classroom metadata (.json files) in our application. For this, we will use **JSON.simple.**

1.7. **Summary:**

In this current phase of the project, the team worked on documenting key components to the overall design of the application. Such design is necessary in order to have a smooth implementation phase and to avoid unexpected delays due to poor planning. The team created UML class diagrams to capture the key requirements of the project. The team also created CRC cards to provide a textual description of the diagram, outlining the main design decisions and choices behind it. In this design phase, an initial wireframe markup was also created to see an initial visual representation of all the features being implemented in this application. There was emphasis on a simple and easy to use UI that still included all the key components required to make the application effective. Finally we decided on using JSON for storing the POIs. We also are converting the maps to PNGs to layer them in addition to using Eclipse and NetBeans for scripting and GUI testing. After designing the overall functionality, UI, and specifications of the app, the team looks to implement our design and start building the application