UWO Maps

1.1. Requirements Documentation

1.1.1. Revision history:

Version	Date	Author(s)	Summary of Changes
1.0	Oct 2nd, 2022 4:50 pm	David Alter	Created Main page with title and subtitle, added revision history table, table of contents, and main headings.
1.0.1	Oct 5th, 2022, 1:30 pm	David Alter	Added first draft of Non-Functional Requirements section to the document
1.0.2	Oct 5th, 2022, 4:00 pm	Ryan Cook	Added to the first draft of the Domain Analysis question
1.0.3	Oct 5th, 2022, 4:40 pm	Minsoo Park	Added the first draft of Functional Requirements
1.0.4	Oct 5th, 2022, 5:05 pm	Taoran Gong	Made changes to the draft of functional requirements
1.0.5	Oct 5th, 2022, 4:45 pm	Dayton Crombie, Ryan Cook	Updated all portions of Domain Analysis
1.06	Oct 8th, 2022, 7:30 pm	Minsoo Park	Added the 1-3 Activity Diagrams
1.07	Oct 8th, 2022, 9:00 pm	Dayton Crombie	Added introduction
1.0.8	Oct 9th, 2022, 9:45 am	Minsoo Park	Edited Use cases
1.09	Oct 9th, 2022, 11:55 am	Taoran Gong	Added the 7-9 Activity Diagrams
1.0.10	Oct 9th, 2022, 3:45 pm	Taoran Gong	Made some changes on use cases and activity diagrams
1.0.11	Oct 11th, 2022, 10:00 am	Ryan Cook	Added the summary
1.0.12	Oct 11th, 2022, 11:30 pm	David Alter	Added Activity diagrams 4-6 (first edit)
1.0.13	Oct 11th , 2022, 12:42 pm	David Alter	General Document edits, left comments on certain parts
1.0.14	Oct 11th, 2022, 1:18 pm	Ryan Cook	Edited and Added to the Domain Analysis

1.1.2. Table of contents:

2. Introduction:

2.1.1. 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 provides 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. The application will also include data useful to the user such as class metadata that will give details about certain classrooms.

2.1.2. 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

2.1.3. References:

- CS2212 Group Project Specification, Version 2.0, Fall Session 2022

2.2. Domain Analysis:

2.2.1. What is the domain for this project?

The domain for this project is 'Geographical Information Systems'.

2.2.2. What do we know about the domain?

GIS is a spatial system that creates, manages, analyzes and maps all types of data. In terms of navigating buildings on the Western University campus, GIS' like Google Maps allow users to visualize the location of different buildings on campus in addition to displaying all sorts of different information about these buildings. Such information may include: opening and closing times, how long does it take to get to these buildings, etc... users in this domain are predominately students who are trying to navigate campus buildings.

2.2.3. What are the common issues encountered in this domain?

An issue in the domain is that maps can be outdated (ex. classroom names change, building layouts change) on the GIS which can confuse the user. Also, when searching for specific classrooms using GIS, they will only show you how to get to the entrance of the building and once inside, they do not have information on navigating between rooms. Most GIS' have very little information about the interior of buildings on campus. This can make looking for a classroom or another place inside campus building's difficult to do. Another issue with GIS systems is that they require a large database of maps depending on the number of buildings/area of land they require to map, so having the maps available and storage may be common issues to deal with in the GIS domain.

2.2.4. What are the common solutions to the above issues in this domain?

A solution to dealing with outdated geographical data could be to add a function to allow developers to easily update maps. Since the project is only tracks the western campus, it would be easy to keep up with building changes whenever there is one. A solution to not having information about the interior layout of the buildings would be to have our application map the interiors of the buildings including all the classrooms and points of interest (POIs) in those buildings using a simple Graphical User Interface (GUI). A solution to GIS systems requiring large amounts of map data would be to optimize how the data is stored (images and metadata) by using memory-efficient files.

2.2.5. How can we use this domain understanding to improve or accelerate development of this project?

Through studying the capabilities and limitations of the domain, we can create an app that will solve the problems which our competitors fail to address (mapping the interior of the buildings). This guarantees that our app will be unique and have a purpose and therefore will be valuable to the user. By also knowing the limitations of the domain that we will to operate within, we know what can be accomplish within the given timeframe and will avoid to implement features that are too complex. That will ensure that the development of the project will run smoothly and we will not have to restart when we encounter an impossible task. The domain analysis will also help us set the structure for our app in the planning phase which will accelerate its development in the future. We will be able to successfully create an application that meets the customer requirements.

2.3. **Functional Requirements:**

Functional requirements include required functionality of the application and define a function of a system or its component, usually expressed in the form "system must do (blank)". We have split the following section into three parts as required by the scenario-based model.

- Actors (Users, Developer, Systems)
- Use cases
- Activity Diagrams

2.3.1. Actors

Actor 1

Actor	Users
Description	Users search and explore the interior spaces of the buildings of Western Buildings
Aliases	Student and Staff, Western Community, end-user
Inherits	None
Actor Type	Person (active)
Relationships	Use the application to get the interior space of Western buildings

Actor 2

Actor	Developer
Description	*Developes the system for the user, inputs metadata, cleans the maps*
Aliases	Programmer, Admin
Inherits	Stakeholder/client (Works for Western University)
Actor Type	
Relationships	May also be a user of the application Maintain the application

Actor 3

Actor	Western Building Interior Space Application
Description	This application allows users to explore the floor plans/ interior space/ POI of Western Buildings
Aliases	Program, Project
Inherits	None
Actor Type	External System
Relationships	Created by the developer and serves the user

2.3.2. Use Cases

2.3.2.1. Browsing Maps

Name	Select a building
Primary actor	User
Secondary actors	Application
Goal in context	To select a building on map
Preconditions	The user can access the map
Trigger	When user click a building from a list
Scenario	1. The user browses the map 2. The user clicks a building from a list 3. The user can flip through all the maps for the selected building 4. The application shows a selected building on the map
Alternatives	1. The user can display or hide a layer 2. The user fit or scrolling through the map
Exceptions	1. The user have different accessibility on layer

2.3.2.2. Searching Maps

Name	Search for POI
Primary actor	User
Secondary actors	Application
Goal in context	The searched POI is displayed, and the it is highlighted in some fashion on the map.
Preconditions	The user accesses the map POI includes metadata
Trigger	The user initiates a search for a POI
Scenario	1. The user searches the POI 2. System display POI, short description, and highlight POI on map
Alternatives	1. Using text entry or drop down menu 2. The user fit or scrolling through the map
Exceptions	1. If we use text entry, it should be able to parse the text

2.3.2.3. Built-In Points of Interest (POI)

Name	Built in POI
Primary actor	System developer
Secondary actors	Application
Goal in context	Store the data on POI and set a layer
Preconditions	The system developer access on map / edting tool
Trigger	The system developer creates the new POI for a building

Scenario	1. The developer open new POI 2. Store data on POI 3. Set layer type on POI
Alternatives	1. Use toggle button to provide hide/display function 2. Use menu pop-up to provide hide/display function
Exceptions	1. User might need to restricted for using only one layer at same time

2.3.2.4. Favourites

Use Case	Store POI as favorites
Primary Actor	User
Secondary Actor	application
Goal in Context	Enable users to mark and unmark build-in POI as favorites for quick access
Preconditions	User must have a building to be used as favorite
Trigger	User wants to select one building for quicker access
Scenario	1. User opens the application 2. User searches for a location 3. User can select it with a favorite toggle
Alternatives	Users do not store favorites; instead they manually search for one particular buildings each time
Exceptions	Users are not aware of the favorite function

2.3.2.5. User Created Points of Interest (POI)

Use Case	User created POI
Primary Actor	User
Secondary Actor	application
Goal in Context	Allow users to create their own POI
Preconditions	Users must be able to add layer and to access certain functions
Trigger	Users want to provide their own description to the building
Scenario	1.User opens the program 2.User searches for the building 3.User edits the texts or names of the building and its interior space
Alternatives	One alternative is that user can report whatever they want to change to the developer
Exceptions	1. No information provided to user to edit POI 2. User fails to change POI because the program does not allow public to change it

2.3.2.6. Editing Tool/Mode

Use Case	Edit Tool/Mode
Primary Actor	System developer
Secondary Actor	Application

Goal in Context	To allow developers to edit the metadata for buildin POI
Preconditions	The program must exist and allow developers to make changes
Trigger	The developer needs to edit certain data
Scenario	1.The developers open the program 2. The developers enter the password and log in to their admin account 3. The program shows a special menu like “edit” 4. The developer can edit it freely
Alternatives	
Exceptions	

2.3.2.7. Housekeeping

Use Case	Housekeeping
Primary Actor	Application
Secondary Actor	User
Goal in Context	To allow user exit the program and provide certain help function if needed
Preconditions	Clear program menu for help/exit
Trigger	The user finishes with the program or gets stuck with some program functions
Scenario	1.User opens the program 2. User cannot find a certain tab/function 3. User sees a clear help function on the menu 4. User goes to the help page to solve their problems 5. User is able to exit the program
Alternatives	None
Exceptions	The help function is not designed or is even more confused to use

2.3.2.8. Multi-user System

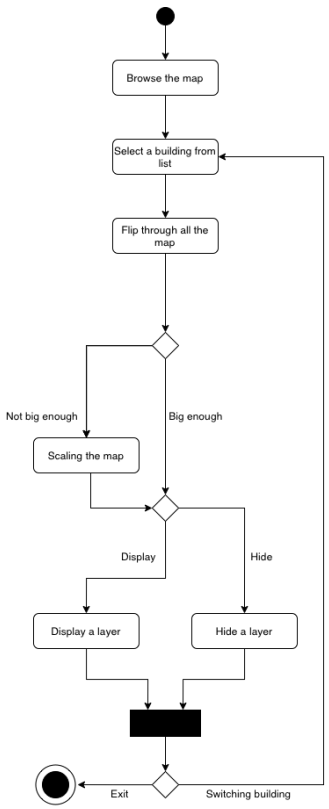
Use Case	Multi-user system
Primary Actor	User
Secondary Actor	Application
Goal in Context	To support multiple users at the same time
Preconditions	Account system must exist
Trigger	Multiple users need the application
Scenario	1.Each of the users opens the application 2. Each of the users enters their account information (username + password) 3. They log in to their own accounts 4. Explore all features of the applications
Alternatives	None
Exceptions	User forgets their login username/password

2.3.2.9. Extra Classroom Metadata

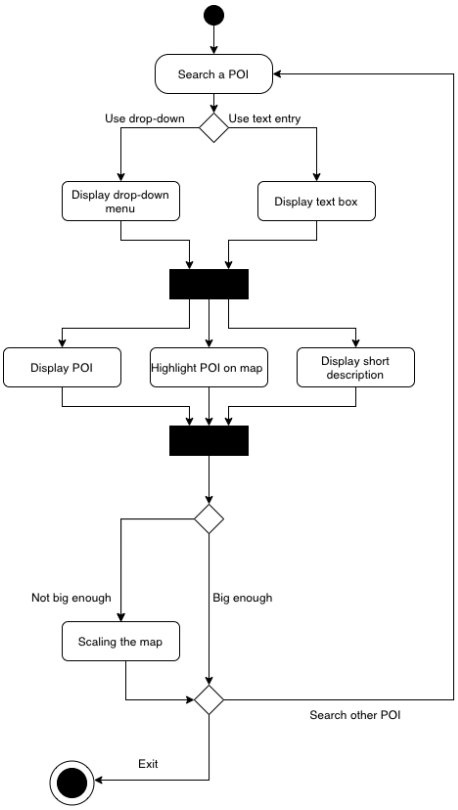
Name	Add extra information
Primary actor	System developer
Secondary actors	*Application*
Goal in context	Each classroom has extra metadata; User get more information on a particular classroom
Preconditions	A classroom POI is already created
Trigger	*The developer initiates to add extra data on classroom POI*
Scenario	1. Developer enters the system 2. Add seating capacity, seating type, projection screen type, information about A/V equipment, pictures of the class room, and in some cases exam seating charts
Alternatives	Not all of the information must be shown at once
Exceptions	

c. Activity diagrams

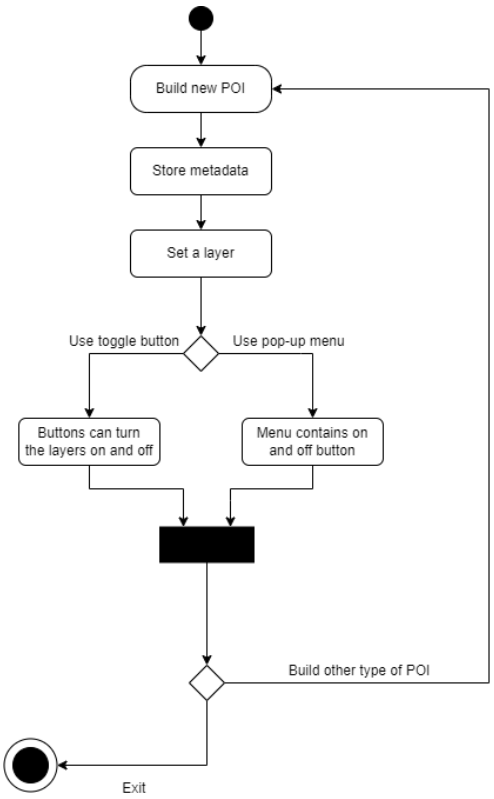
1. Browsing maps



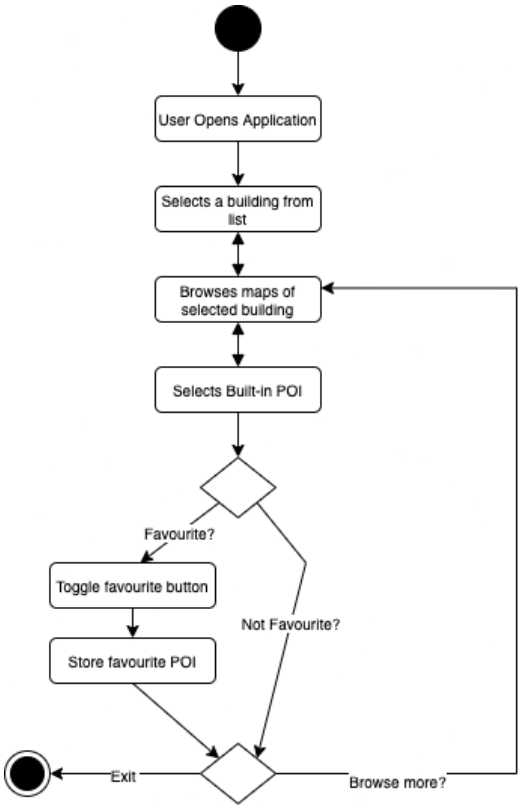
2. Searching maps



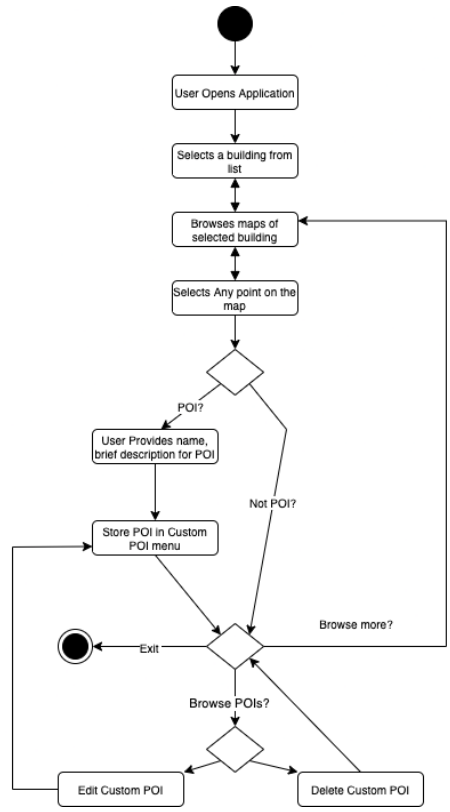
3. Built-In Points of Interest (POI)



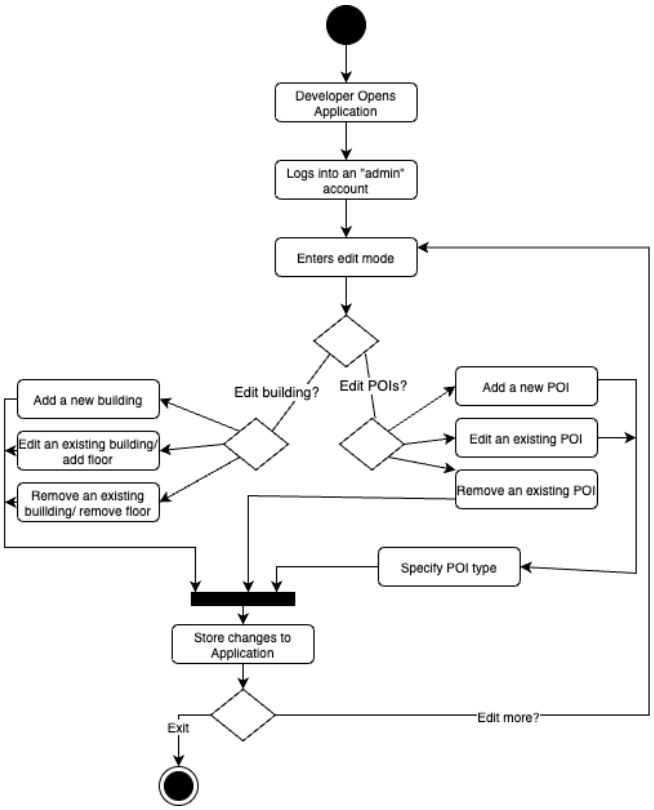
4. Favourites



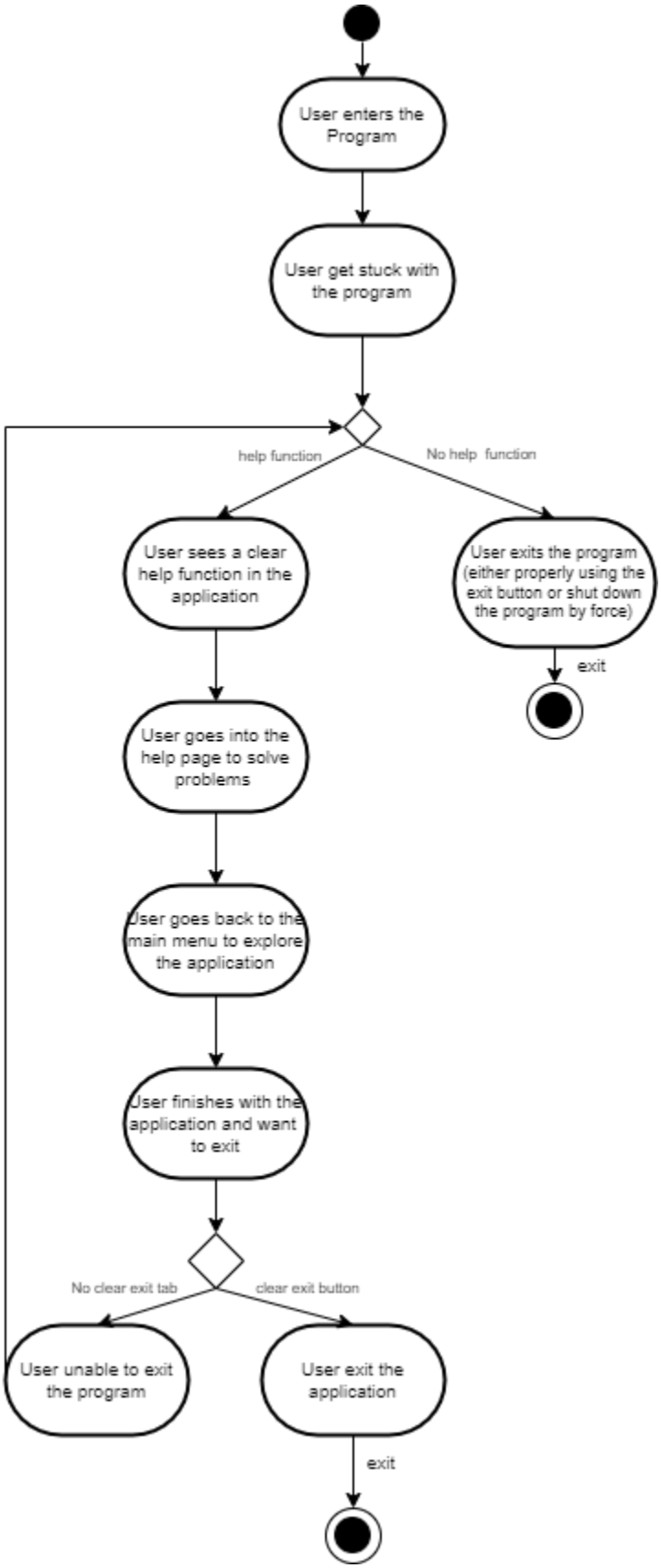
5. User Created Points of Interest (POI)



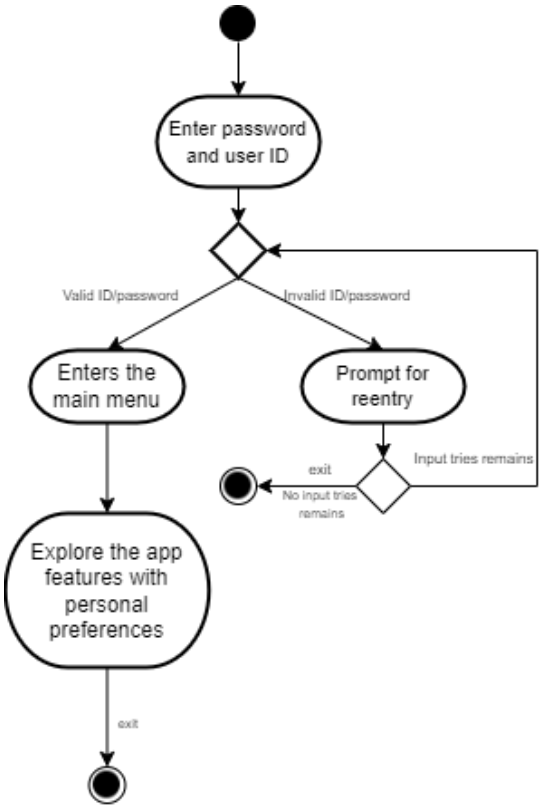
6. Editing Tool/Mode



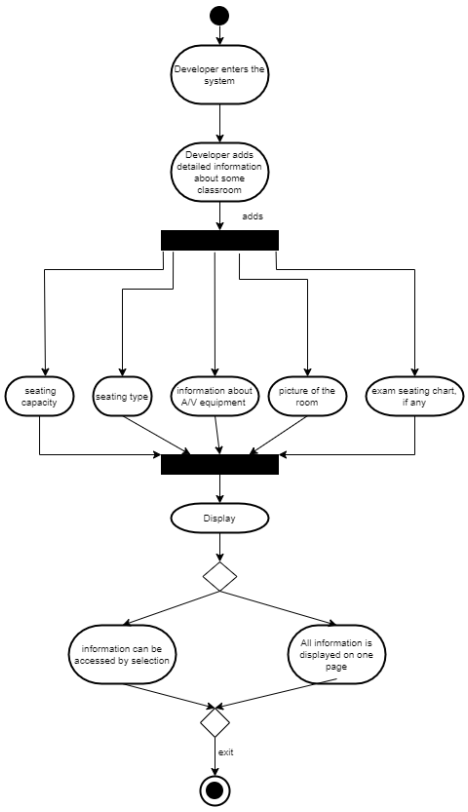
7. Housekeeping



8. Multi-User System



9. Extra Classroom Metadata



2.4. **Non-Functional Requirements:**

In additional to the functional requirements of our application as discussed above, during the duration of this project, we must also additionally

adhere to a few non-functional requirements, as requested by our stakeholders. These requirements differ from the ones above because they can be

thought of an overall property of the system as a whole or of a particular aspect and not a specific function that UWO Maps must perform.

2.4.1. Languages and frameworks:

One of the requirements proposed to us by our stakeholders is that the application must be developed in **Java**, This may be due to the ease in running the

application after installing a JVM on a client's computer. Furthermore, our application will need to include a GUI (Graphical User Interface) for interacting with Western

University's maps, the stakeholders of the project left the choice of GUI framework up to us. After extensive research and thorough comparison between Swing framework

and JavaFX, we have ultimately decided to implement the GUI **using the Spring framework**, mostly for the ease of debugging on NetBeans IDE.

2.4.2. Maps and Metadata:

An important non-functional requirement of our project is that both the maps and metadata are stored locally, not requiring an internet connection to load the maps onto the application.

We needed to consider a few factors in order to adhere to this requirement, such as:

- Will we keep the maps in a PDF format, potentially having to use a third-party PDF library to use them in Java?
- Or, will we change the maps to a different format such as PNG or JPEG?
- for the metadata, how will we store it without an internet connection? JSON, XML, SQLite?
- Will we need to use a third-party library for working with the metadata?

After assigning different team members to individually research and answers these questions, we have decided on the following: To avoid having to work with a third-party library in regards to the

PDFs, we will convert the maps to a **PNG format**, this will not only save us time, but will also allow us to make use of some easy-to-use tools in the Java Standard Library such as *ImageIO* and *BufferedImage.*

When it comes to handling the metadata, We have decided that the best tool to store the metadata is **JSON** because it is the shortest compared to XML and SQLite, and it is the most simple to read.

We will need to use an additional Java JSON library which we have not yet decided upon, but the two main options are JSON-simple and GSON (google-gson).

2.4.3. Coding and Naming conventions:

We recognize that it is crucial in a software project such as this one to write clean code, and in order to due that, we must remain consistent in applying coding conventions across all files in our

application. Therefore, after discussion, all developers agree that we will abide by the following conventions in our files:

- All code in the application must be commented using **Javadoc**.
- Indentation is 4 spaces
- We will use official **Oracle Code Conventions for Java**

2.4.4. Other Non-Functional Requirements:

One of the non-functional requirements discussed earlier in this section was that the application must be built in Java so it can be complied on any machine, thus another requirement given this

is that the application should be executable on a **Windows 10 system** with a **standard Java installation**. The application should also **not need to use any other libraries** other than the Standard Java

Library (with the exclusion of a JSON library, which we will **provide instructions** to use). Furthermore, Our application must not create, modify, or delete files outside of the directory in which the

application is installed, and subdirectories of this directory. Additionally, we understand there must be a visible response to every user action. Erroneous actions or actions that could

not succeed must be met with a useful, professional error message. Finally, we agree to use and abide by sound software engineering principles when designing this application.

2.4.5. Code repository:

As stated in the Project Specification document, All members agree that project code must be checked into the designated repository assigned to us and members must commit and push code frequently.

2.5. **Summary:**

As technology continues to improve, we are able to come with better solutions to solve the problems that we face in our lives. For students at Western, one problem is the difficulty of navigating the complex interiors of many of the buildings at Western. Students searching for class rooms, places to eat, and the washrooms often struggle with looking at the complex PDFs that exist now as they try and search the interior of these buildings which are often unfamiliar to them. Through creation of a GIS app that can browse and search through different buildings with a simple and easy to use UI, students can start navigating these buildings with ease. This app utilizes the current maps at Western to allow users to search for different points of interest in addition to creating their own. Students will even be able to access details about each classroom to prepare them for the day ahead. In essence, this app will help students save the time and anxiety while trying to explore buildings on campus which will help them in their day to day lives.