1. Main Page

WEGIS (Western Expedient Geographical Information System)

This Document outlines the Requirements Documentation for WEGIS in CS2212B

Version	Date	Author(s)	Summary of Changes
	29 Jan 2023	Daniel, Foster, Matt	Created and organized confluence pages
	30 Jan 2023	Daniel	Updated Use Cases 6-10, made actor diagrams
	30 Jan 2023	Matthew	Updated use cases 1-5
	30 Jan 2023	Marc, Foster	Created Activity Diagrams 1-8
	31 Jan 2023	Jane	Created Use cases 11-15
	5 Feb 2023	Jane	Created non-functional requirements page
	5 Feb 2023	Daniel	Finished use cases 6-10
	6 Feb 2023	Matthew	Finished introduction
	6 Feb 2023	Marc, Foster	Made changes to the Activity Diagram 4,5 and 8
	6 Feb 2023	Marc	Updated the summary page.
	7 Feb 2023	Foster	Created Activity Diagrams 9-14 Created Team Contract Draft
	7 Feb 2023	Daniel	Finished Domain Analysis
	7 Feb 2023	Matthew	Added table of contents on main page
	7 Feb 2023	Marc, Foster	Added Use Case Diagram

2. Introduction

Overview

University buildings can often be confusing to navigate. Navigating outdoors has become much easier through the use of smart phones, GPS, and mapping services, which allow people to locate buildings with ease. However, many of these services do not do a good job of showing maps of interior spaces, especially with buildings that have multiple floors. Western provides the public with floor plans of all the buildings on campus, but they are only PDFs with no metadata, meaning they are not easy to navigate.

The main purpose of this program is to use the maps provided by Western to allow users to explore the floor layout as well as search for specific points on the map. This includes searching for specific rooms in buildings, locating points of interest in a building, browsing through the available maps, and allowing the user to create their own points of interest. The program will also have an editing tool to allow other developers to create and edit map metadata.

Objectives

The objectives of this project will be to:

- Apply principles of software engineering towards a real-world problem
- Work with, interpreting and following a detailed specification provided
- Create models of requirements and design from a specification
- Implement all the requirements and designs in Java, while dealing with decisions that were made earlier in the design process
- Create a user-friendly user-interface
- Write robust and efficient code
- Write well-documented code in Java using best practices
- Reflect on good and bad design decisions that have been made throughout the project

References

Project specification document

3. Domain Analysis

Software Domain: Desktop Application

Subdomain: Geographic Information System (building map viewer)

What do we know about this domain:

1. GIS provides a graphical display of some geographical area, in this case, the inside of a building
2. Map viewer applications are used to provide individuals with directional information
3. Building map viewers have the same effect, but specifically for the interior of buildings
4. The building map should include information on rooms
5. The building map should include maps for each floor

Common issues encountered in this domain:

1. Displaying a multi-floor building cannot be done with just one map
2. Having many POIs could crowd the map and negatively affect readability
3. The map could be too large to display properly

Common solutions to the above issues:

1. The user should be able to select which floor to view
2. The user should be able to toggle on/off displaying certain POIs or POI types
3. Scrolling can be introduced in order to see the whole map

4.1 Actors

1. User

Description	An individual who is currently using the program. The user is able to view maps and POIs and create their own custom POIs.
Aliases	Viewer
Inherits	None
Actor Type	Person
Active/Passive	Active

2. Admin

Description	A specialized user who is able to edit the maps and the built-in POIs.
Aliases	Creator, Editor
Inherits	User
Actor Type	Person
Active/Passive	Active

3. Weather

Description	Weather API that allows the program to depict the current weather on Campus
Aliases	None
Inherits	None
Actor Type	External System
Active/Passive	Passive

4.2 First Five Functional Requirements

Browsing maps

A function to easily browse through all the maps for all the supporting buildings.

Name	Browsing Maps
Primary actor	User
Secondary actors	Admin
Goal in context	Allow the user to browse through all the available maps.
Preconditions	The user has launched the program.
Trigger	User selects a building from a list, and is then able to flip through all the options.
Scenario	1. User opens program 2. User selects a building from a list
Alternatives	None
Exceptions	None
Priority	Highest, must be implemented

Scrolling maps

When viewing a map, the user can scroll to view other areas of the map if the map is too large to fit the screen.

Name	Scrolling Maps
Primary actor	User
Secondary actors	Admin
Goal in context	Allow the user to scroll through the selected map.
Preconditions	The user has launched the program and selected a map.
Trigger	User selects a building from a list, and is then able to scroll through the map if it is too large.
Scenario	1. User opens program 2. User selects a building from a list
Alternatives	None
Exceptions	The map is not larger than the screen, meaning no scrolling is required.
Priority	Highest, must be implemented

Displaying Layers

Each map has the ability to display/hide layers. Layers can be things like classrooms, washrooms, or user-defined points of interest.

Name	Displaying Layers
Primary actor	User
Secondary actors	Admin
Goal in context	Allows the user to toggle layers (display/hide points on interest on the map).
Preconditions	The user has launched the program and selected a map.
Trigger	User selects a building from a list, and then is able to toggle different layers on the map.
Scenario	1. User opens program 2. User selects a building from a list

Alternatives	None
Exceptions	There are no layers
Priority	Highest, must be implemented

Searching Maps

User can search for points of interest in buildings, typing in keywords such as the name, room number, description, etc. Once user instantiates a search, the map is shown and if the map is large, zooms to the correct spot.

Name	Searching Maps
Primary actor	User
Secondary actors	Admin
Goal in context	Allow the user to search points of interests in buildings.
Preconditions	The user has launched the program.
Trigger	User selects the search bar
Scenario	1. User opens program 2. User selects the search bar
Alternatives	None
Exceptions	None
Priority	Highest, must be implemented

POI Discovery

User can see a list of all existing POIs.

Name	POI Discovery
Primary actor	User
Secondary actors	Admin
Goal in context	Allow the user to browse through all the available maps.
Preconditions	The user has launched the program.
Trigger	User selects a floor.
Scenario	1. User opens program 2. User selects a floor. 3. User can see available POI categories on each floor from the list and see all POIs under a particular category. 4. User can select a POI from that list. 5. User can see POI on the map and view its description. 6. User can either exit the program or view the available POI list again.
Alternatives	None
Exceptions	None
Priority	Highest, must be implemented

4.3 Second 5 Functional Requirements

1. Built-In POI

   A number of points of interest are displayed, represented by some images layered over the opened map. POIs should be displayed or hidden at the user's request. Built-in POIs can only be created, edited, or deleted by an admin.

Name	Built-In POI
Primary actor	User
Secondary actors	Admin
Goal in context	POIs are marked and easily accessible.
Preconditions	The user has launched the program and selected a map
Trigger	The user chooses to display some POI.
Scenario	1. User opens the program 2. User selects a map to view 3. User selects POIs to display 4. POIs appear on the map
Alternatives	User Created POI
Exceptions	The building or floor has no POIs
Priority	High

2. Clicking on POI

   The user must be able to click on a POI and the system must highlight said POI. Its name, room number, and a short description should be displayed, as well as a button to favorite the POI and a button to close the popup.

Name	Clicking on POI
Primary actor	User
Secondary actors	N/A
Goal in context	Access and read information associated with the selected POI
Preconditions	The user has launched the program and selected a map
Trigger	The user clicks on a POI icon
Scenario	1. The user clicks on a POI icon 2. Information about the selected room is displayed & the POI is highlighted 3. User clicks the "exit" button 4. Information popup closes, POI is unhighlighted
Alternatives	N/A
Exceptions	N/A
Priority	Medium

3. Favourites

   The user has the option to mark POIs as favourites. These favourites can be accessed through a menu. When selected from the menu, the location of the POI is displayed.

Name	Favourites
Primary actor	User
Secondary actors	N/A
Goal in context	Bookmark a POI in order to visit it faster at a later time

Preconditions	The user has launched the program, opened a map, and selected the POI that they would like to bookmark
Trigger	Clicking the bookmark button found on the POI information popup
Scenario	1. User opens the program 2. User opens a map 3. User clicks on a POI 4. POI information popup appears 5. User clicks the "favourite" button 6. POI is added to favourites list 7. User closes the POI information popup 8. User opens favourites list 9. Clicking on a favourite shows its location on the map
Alternatives	N/A
Exceptions	POI has already been selected as a favourite
Priority	Medium

4. User Created POI

   Custom POIs can be inserted by the individual user. These POIs will only be visible to that user. They are displayed on one layer that can be toggled on and off.

Name	**User Created POI**
Primary actor	User
Secondary actors	
Goal in context	The user is able to create custom POIs that can be displayed on the map
Preconditions	The user has opened the program and selected a map
Trigger	The user chooses to insert a POI
Scenario	1. User opens the program 2. User selects a map 3. User selects the option to insert a POI 4. User enters the metadata (image, title, room number, description) 5. User selects the location of the POI 6. POI image is inserted at the area 7. POI data is inserted in the POI list 8. User is returned to the map display page
Alternatives	Built-in POIs
Exceptions	Built-in POI already exists for that room
Priority	High, will take some work

5. Persistent Data
   Data entered by the user, including their username, password, and any user-created POIs are saved so that they can be accessed every time the program is used.

Name	**Persistent Data**
Primary actor	Admin
Secondary actors	N/A
Goal in context	User's data is stored permanently

Preconditions	The user has entered some data (ex: User created POI, username/password)
Trigger	The user enters new data
Scenario	1. User opens the program 2. User enters some data (username, password, user-created POI) 3. Data is saved into a file where it can be accessed later
Alternatives	N/A
Exceptions	N/A
Priority	Medium

4.4 Last Five Functional Requirements

Exit/Close and Navigation

An easy and clean exit/close from the current state of application. If there's any unsaved work, it should warn the user to save work before exit/close.

Name	Exit/Close and Navigation
Primary actor	User
Secondary actors	Administrator
Goal in context	Allow user to exit/close the application.
Preconditions	The program has to be launched.
Trigger	User clicks on the exit button on the screen.
Scenario	1. User opens the program 2. User clicks on the exit button
Alternatives	Unsaved work exist 1. User clicks on the exit button. 2. Program warns to save work before exit 3. User chooses to exit with or without saving
Exceptions	None
Priority	High

User Help

Provides instructions on how to use a program when a user is stuck in a program.

Name	User Help
Primary actor	User
Secondary actors	
Goal in context	To help user if they get stuck on the program.
Preconditions	The program has to be launched.
Trigger	User clicks on the help button on the screen.
Scenario	1. User opens the program 2. User clicks on the help button
Alternatives	None
Exceptions	None
Priority	Medium

Editing Tool/Mode

Provides a special mode for the developers to easily access and edit the metadata for the built-in POI.

Name	Editing Tool/Mode
Primary actor	Administrator
Secondary actors	
Goal in context	Edit the metadata for the built-in POI.
Preconditions	The program has to be launched.
Trigger	Access as a developer's special mode.

Scenario	1. Administrator logs in with a special account. (or 1. Separate tool that edits the same metadata file or 1. Run the application with a special command line option)
Alternatives	None
Exceptions	None
Priority	High

Secure the User Data

Secure the user privacy from others when storing user data.

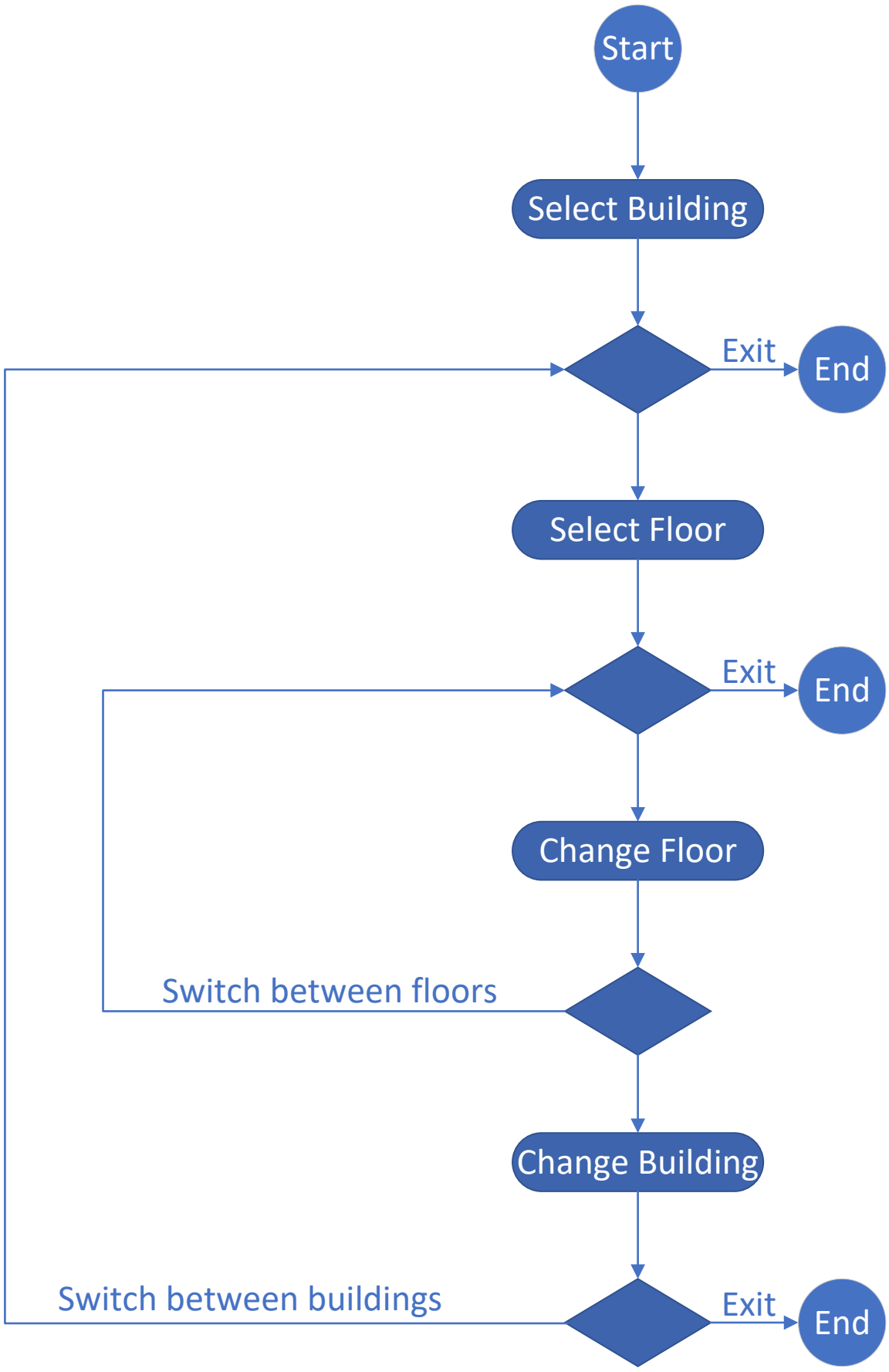
Name	**Secure the User Data**
Primary actor	User
Secondary actors	
Goal in context	Secure privacy of user data.
Preconditions	Contains data that has to be protected.
Trigger	Tries to access the user data.
Scenario	1. User opens the program 2. User enters key or password to decrypt their data
Alternatives	None
Exceptions	If user enters incorrect key or password 1. Ask the user to enter again
Priority	High

Current Weather

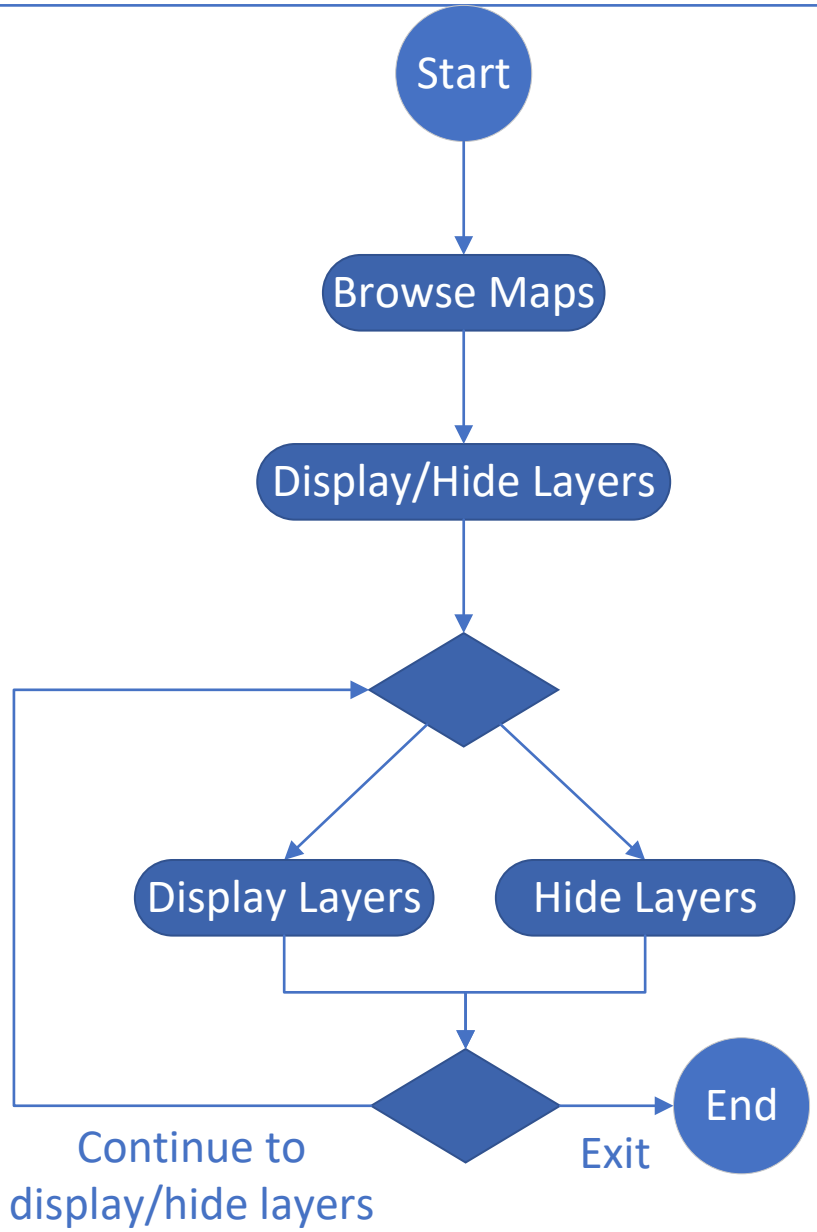
Display the current weather on campus on the screen.

Name	**Current Weather**
Primary actor	User
Secondary actors	Weather API
Goal in context	Display the current weather on campus on the screen.
Preconditions	The program has to be launched.
Trigger	Checks the current weather on the screen.
Scenario	1. User opens the program 2. Checks on the display on the screen
Alternatives	None
Exceptions	None
Priority	Lowest

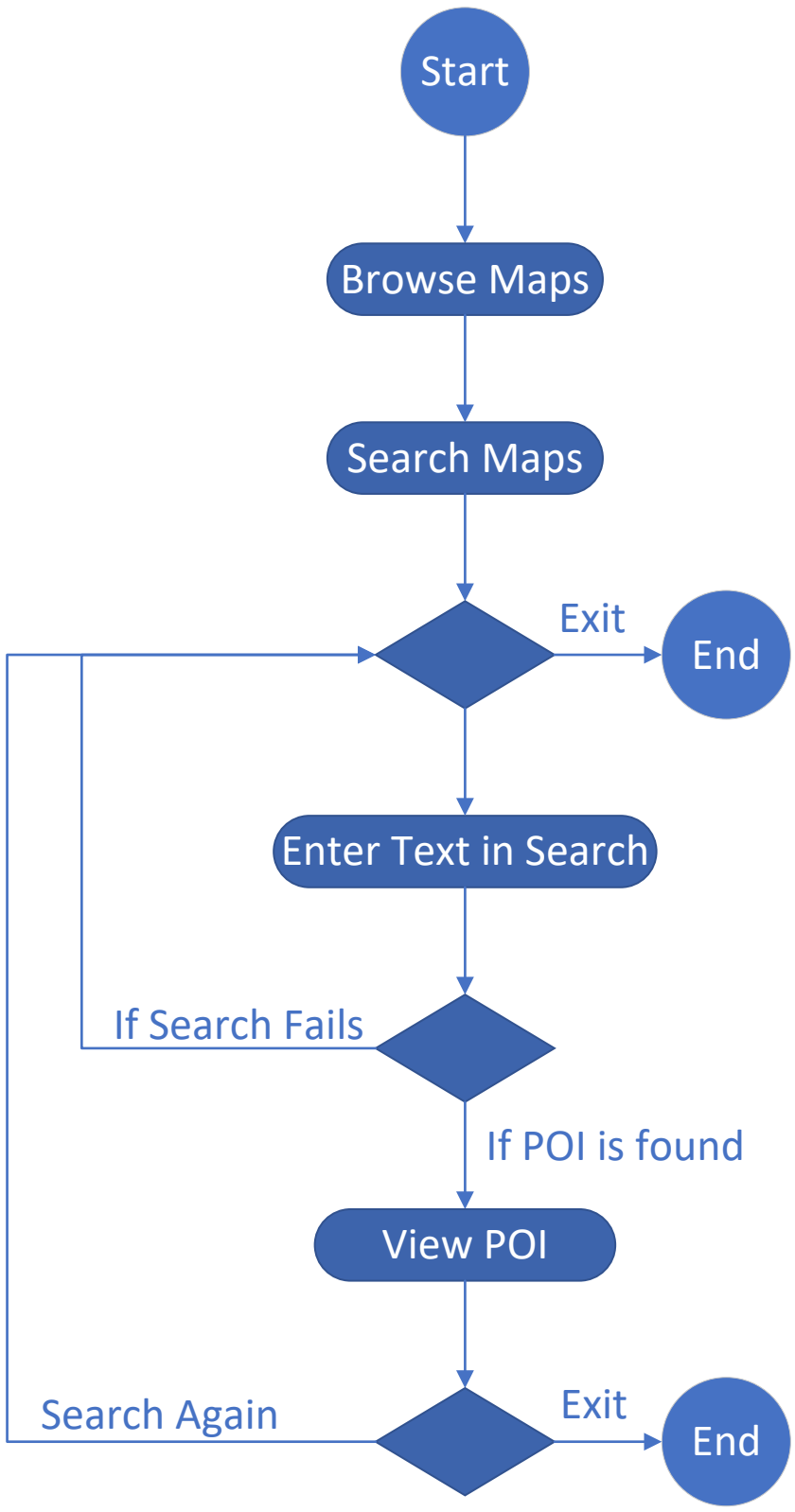
1. Browse Maps



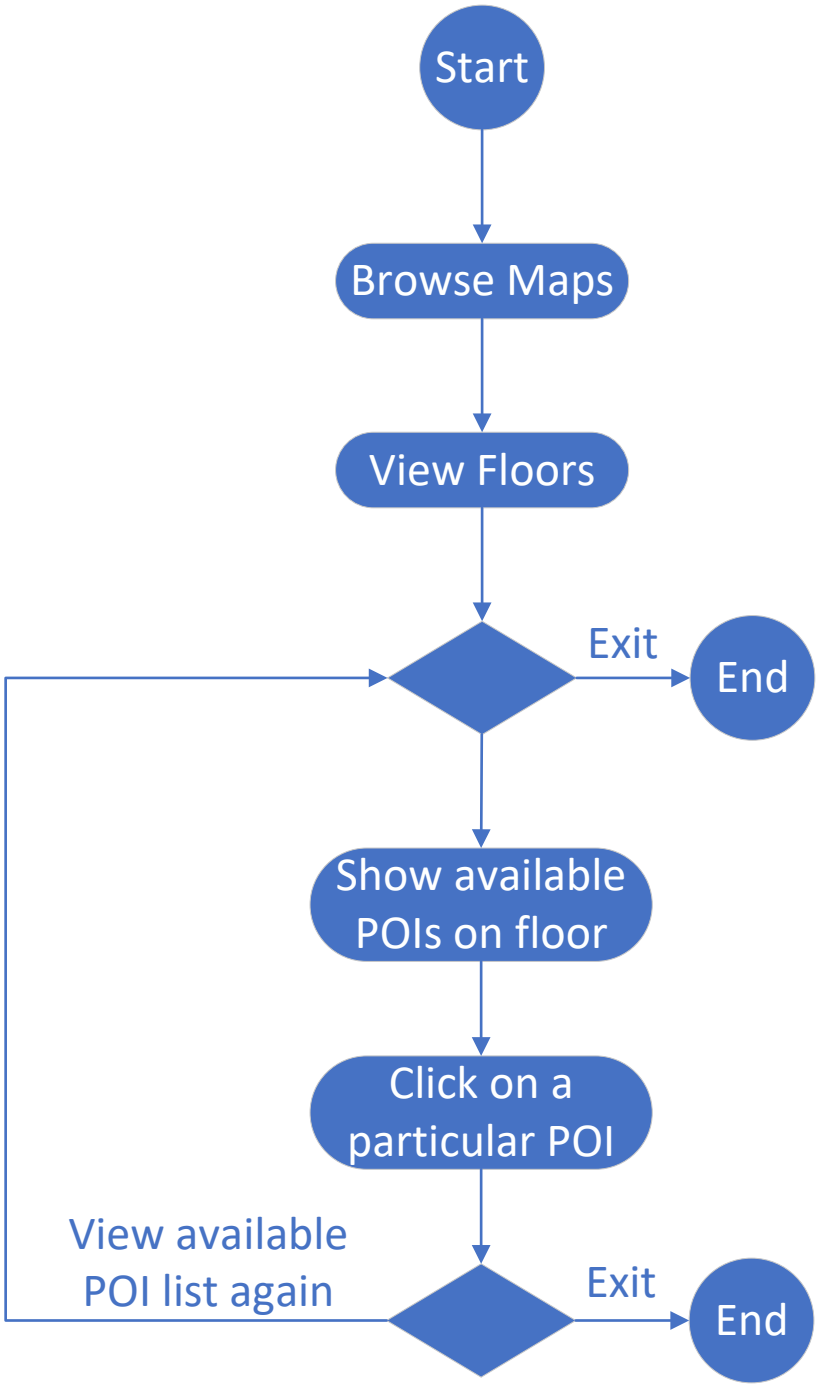
3. Layers


Start
Browse Maps
Display/Hide Layers
Display Layers
Hide Layers
End
Continue to display/hide layers
Exit

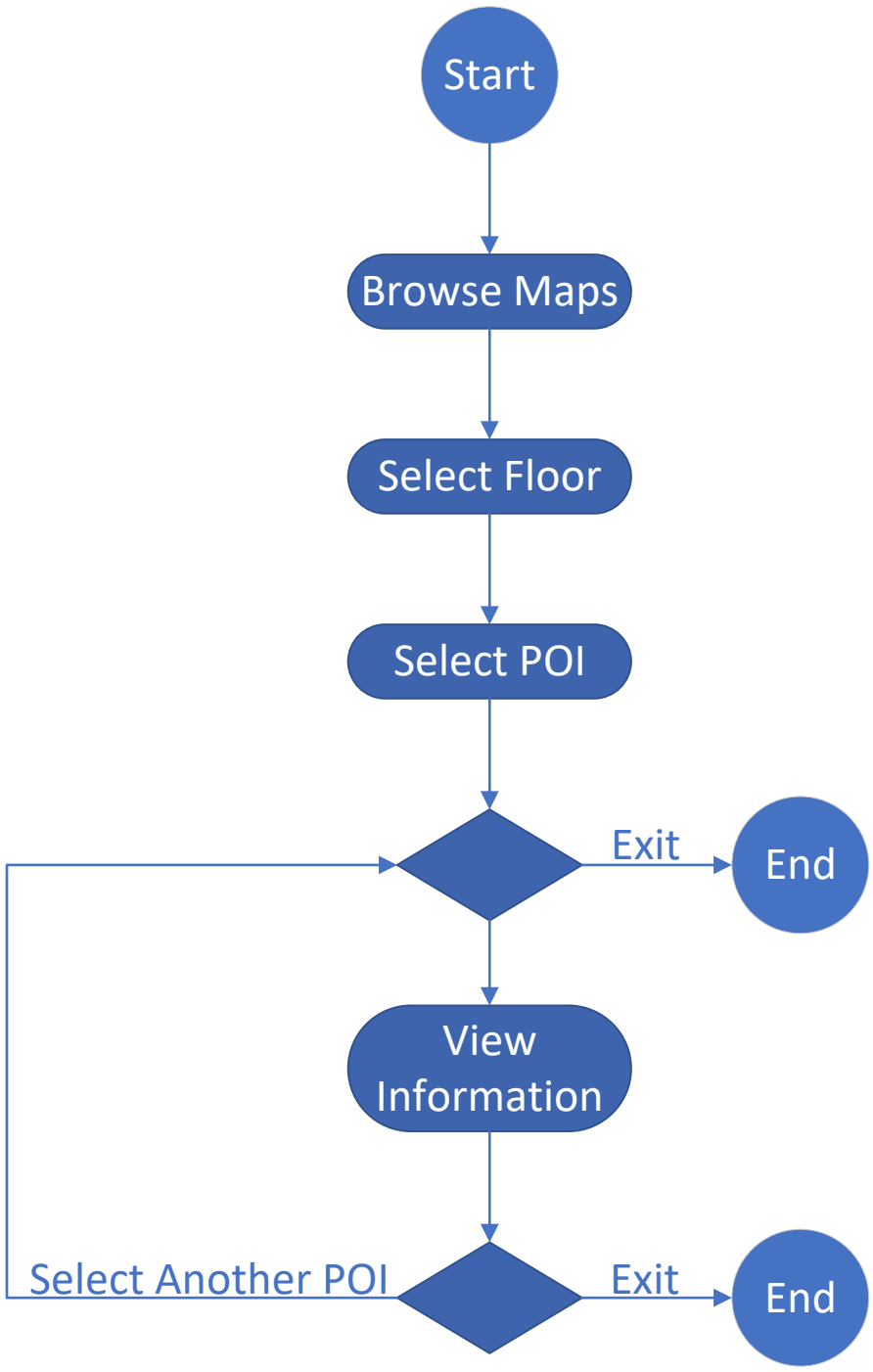
4. Search Maps



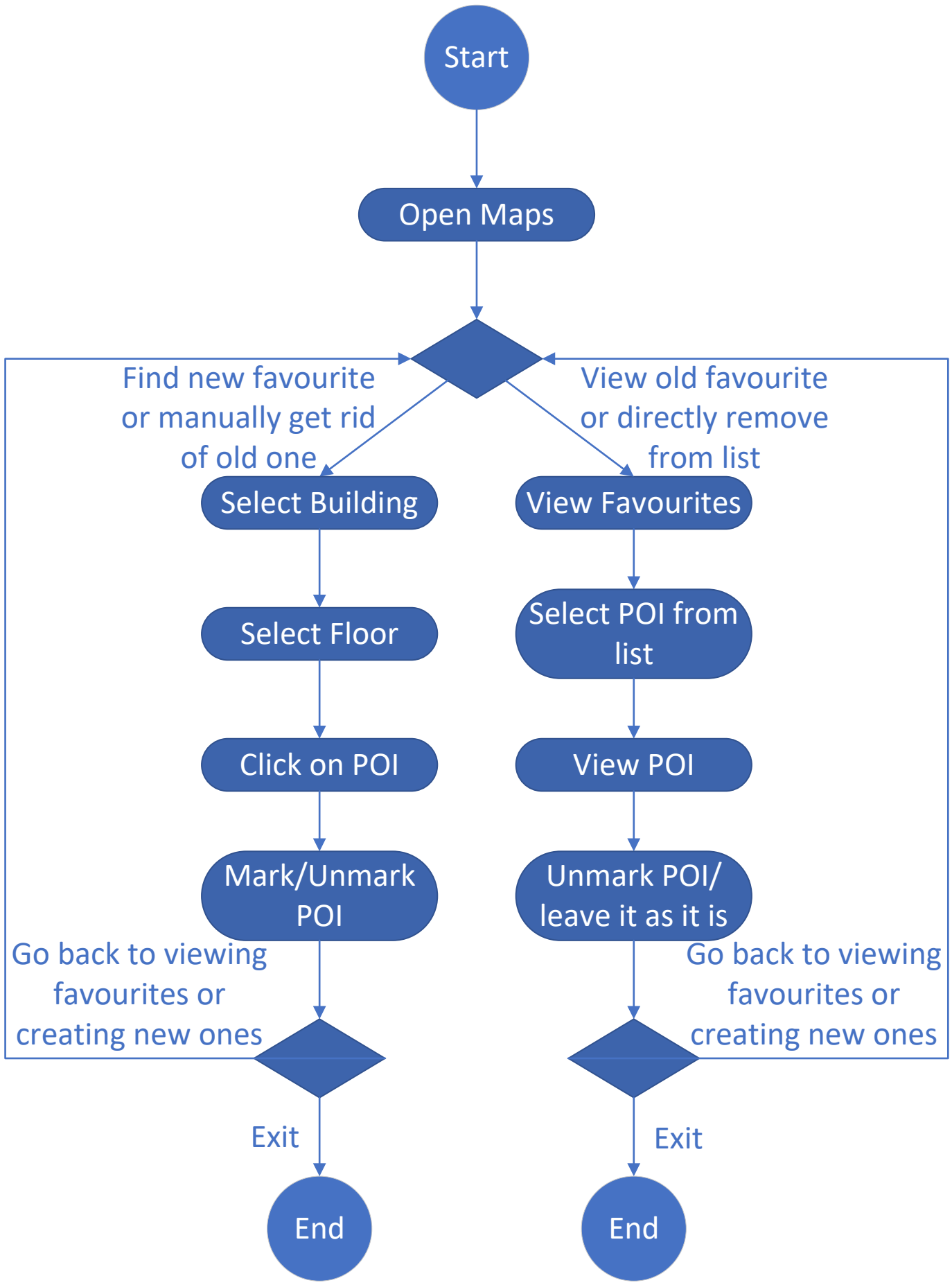
5. POI Discovery



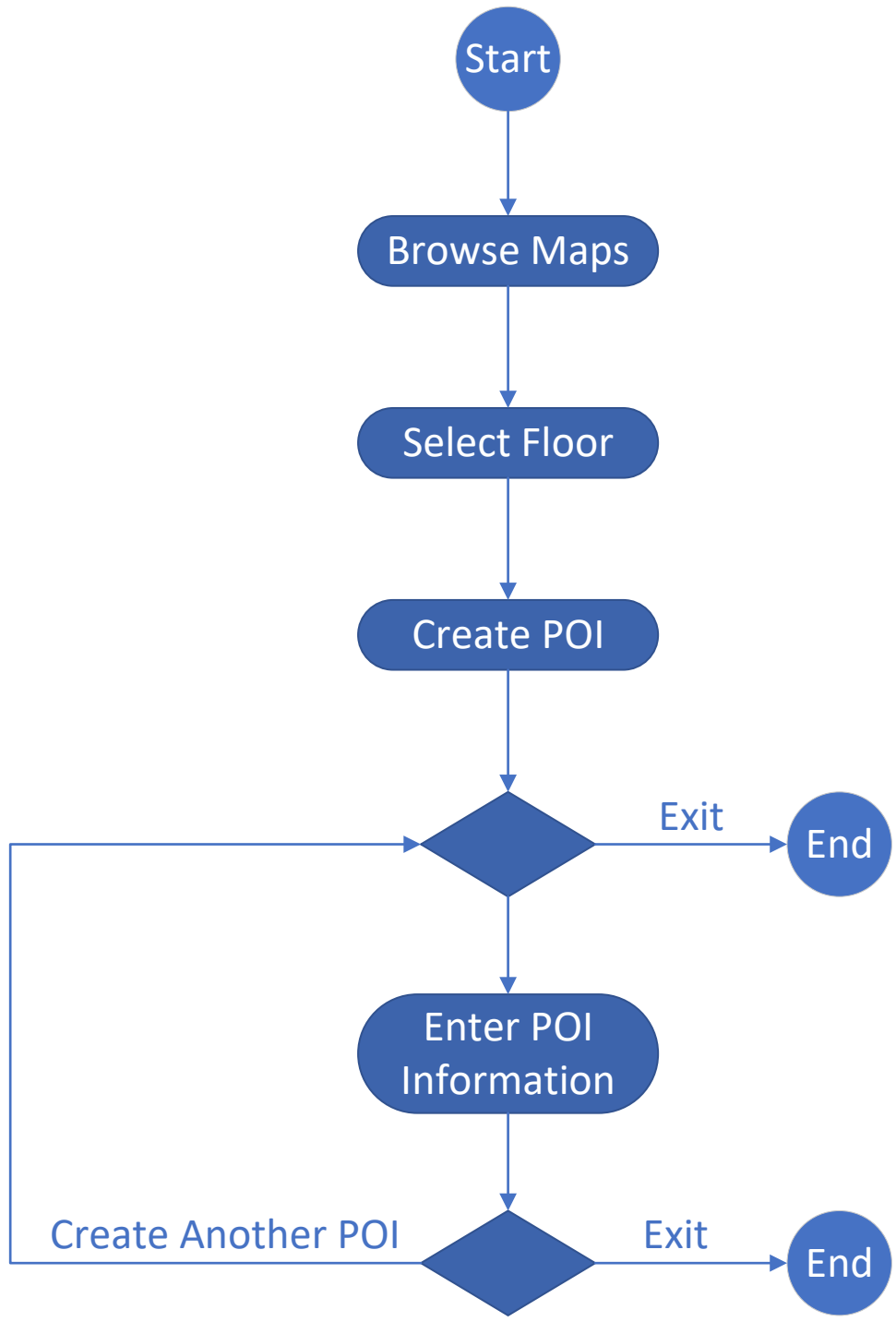
7. Clicking on POIs



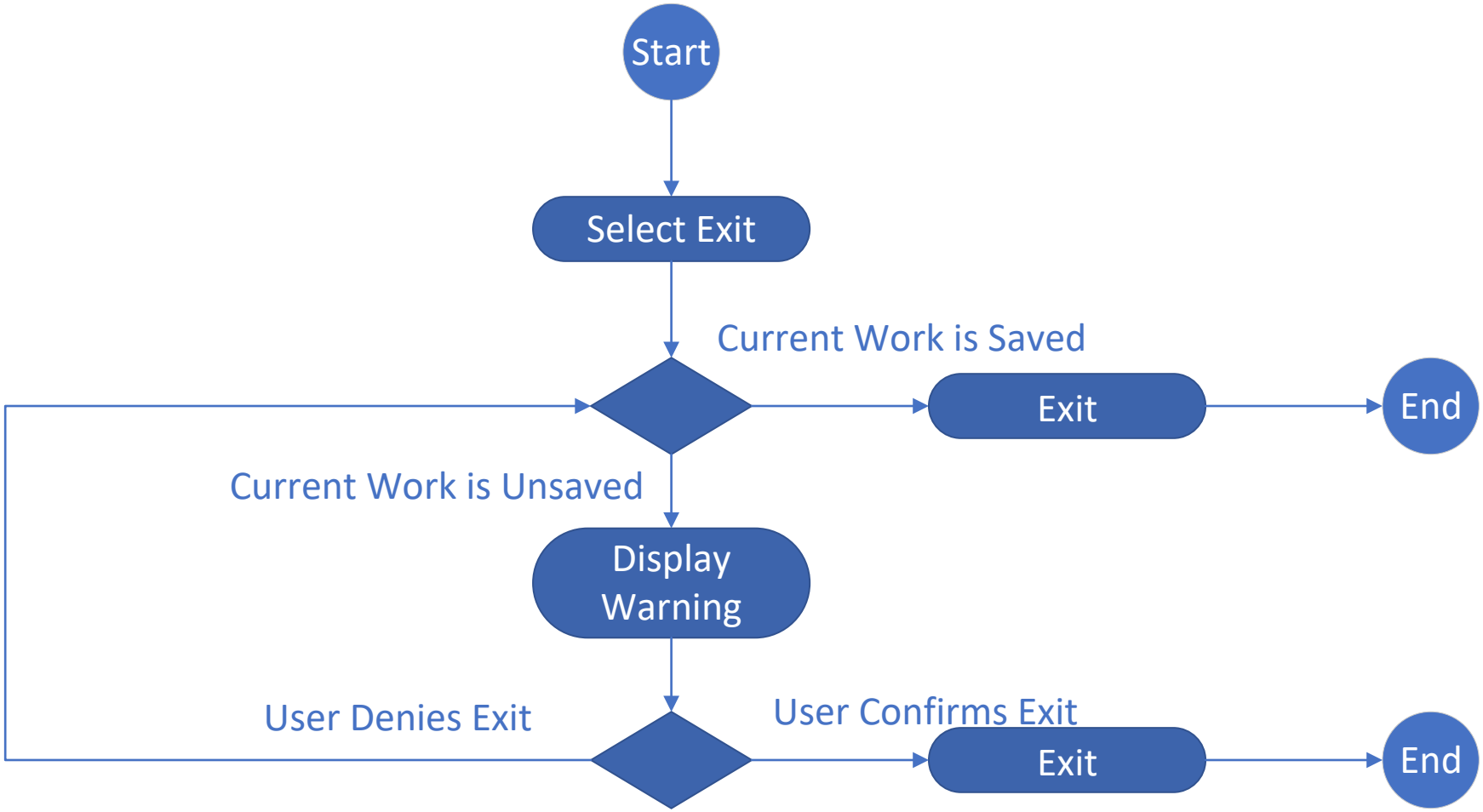
8. Favourites



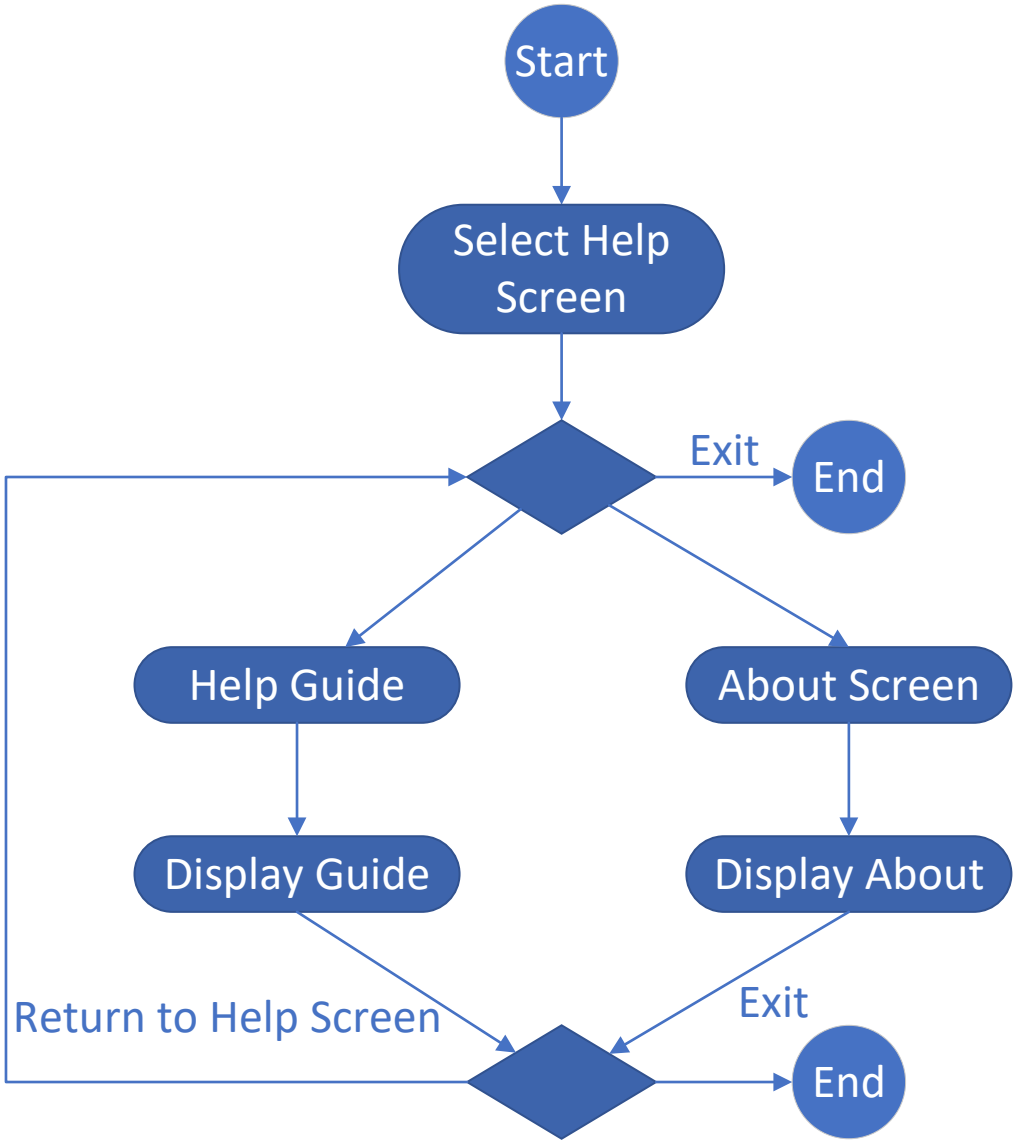
9. User Created POI



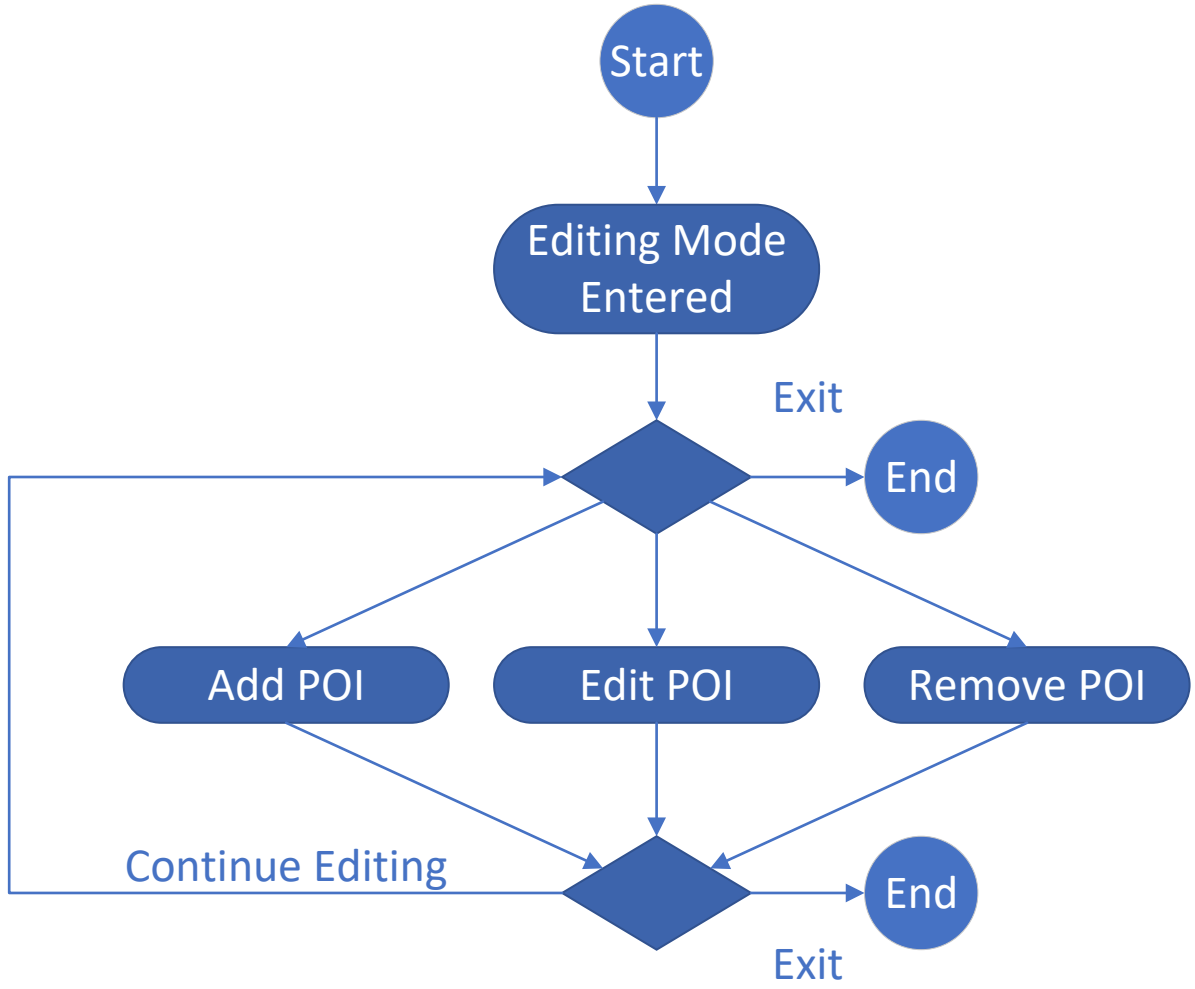
11. Exit/Close and Navigation



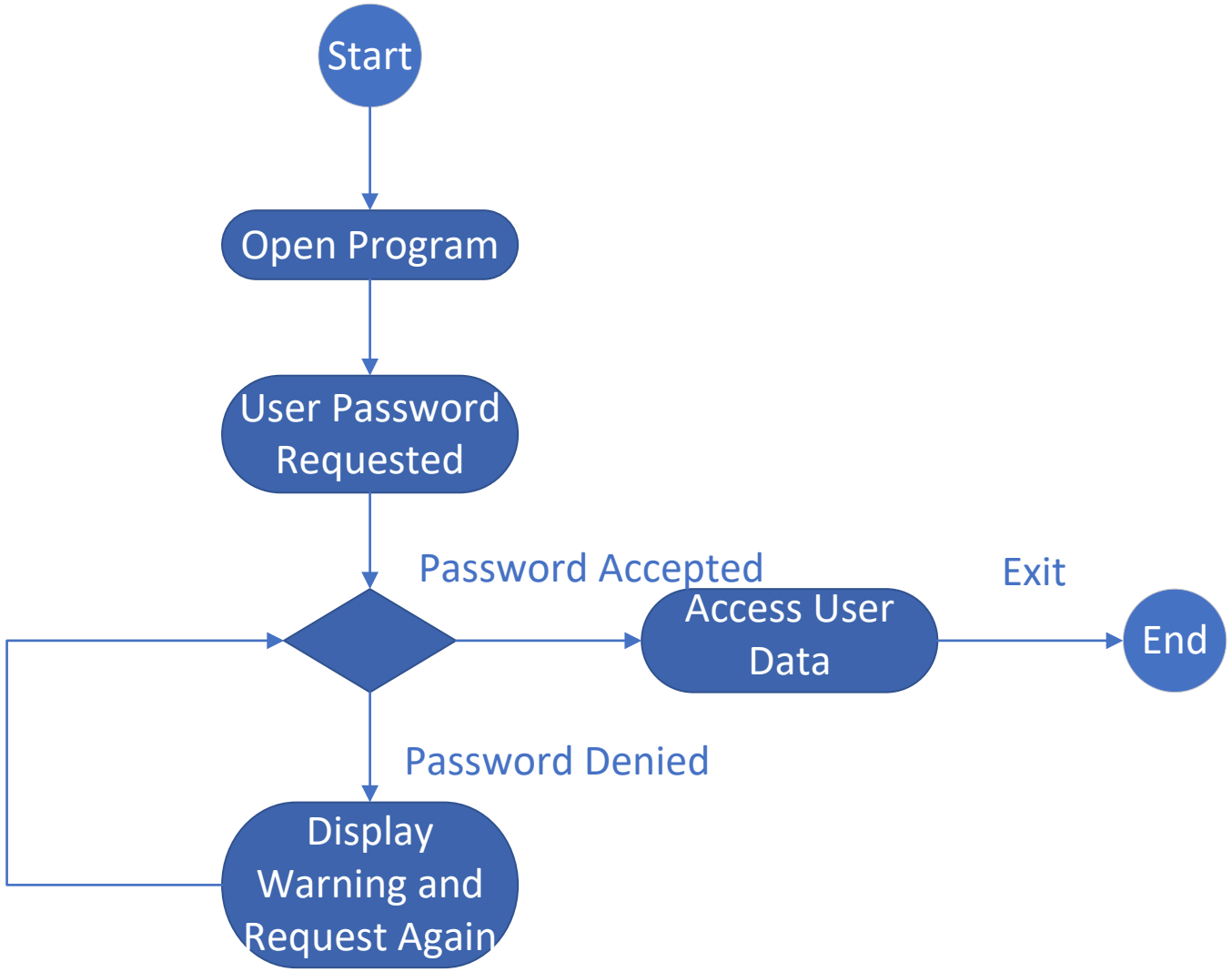
12. User Help


Start
Select Help Screen
Exit
End
Help Guide
About Screen
Display Guide
Display About
Return to Help Screen
Exit
End

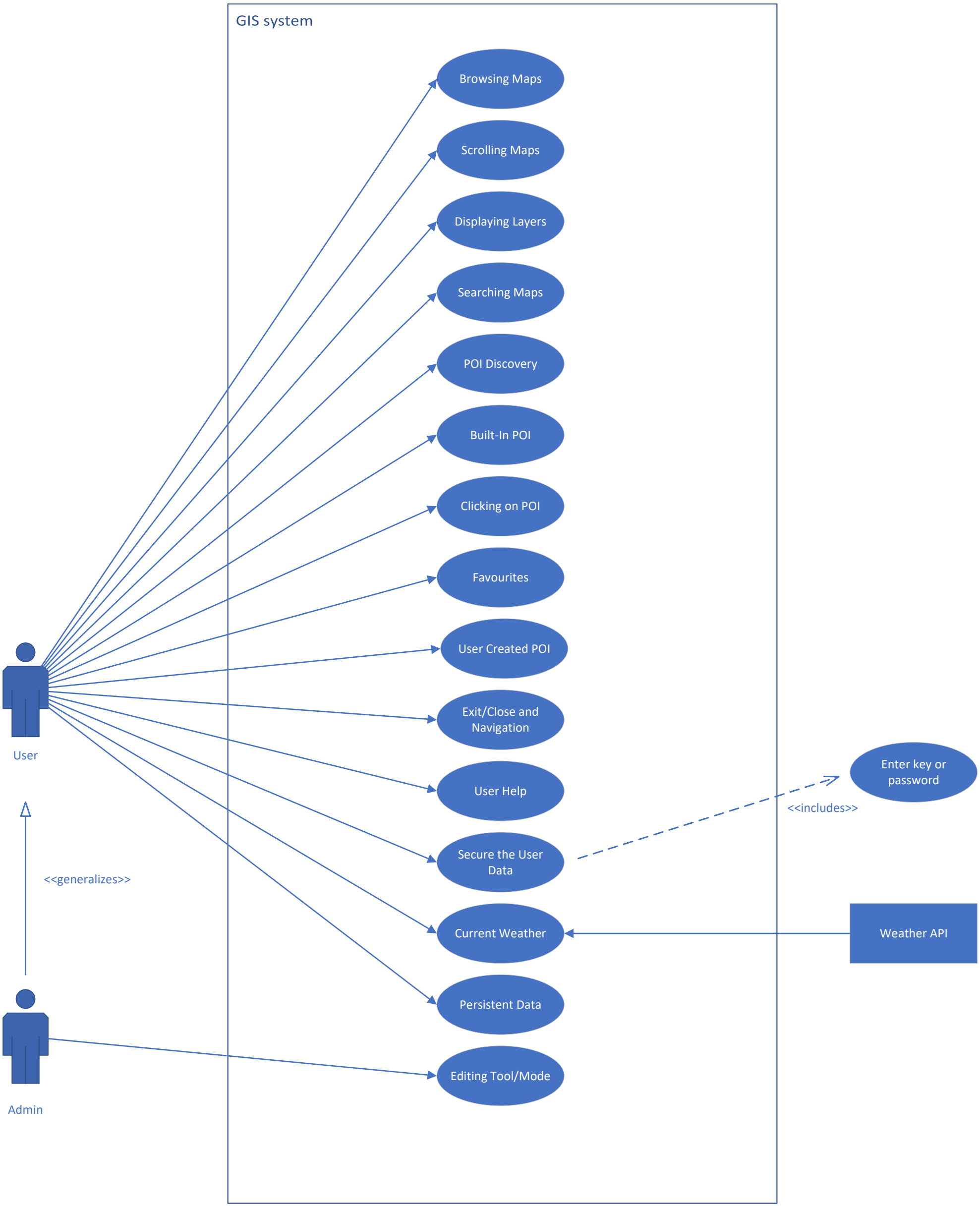
13. Editing Tool/Mode



14. Secure the User Data



Use Case Diagram



5. Non-Functional Requirements

1. Application must be developed with Java 19
2. Application must be developed in a standardized program of Java graphical user interface. Either of Swing or JavaFX can be selected. Lessons on how to use these tools will not be provided in this course and should self-study by doing research as a group.
3. Application must store map metadata locally and no internet connection is required. This can be done using any of the methods selected in the group, such as JSON, XML, CSV, TSV, etc. One suggested option is JSON.
4. Application must be connected to weather service API provided by WeatherAPI or OpenWeather or another provider.
5. Maps are recommended to be stored as images. If stored as PDF, third party PDF library should be used.
6. Application must use pre-existing encryption and security libraries to secure user privacy. Should not attempt to create own encryption algorithm.
7. Any other libraries are not required, but can be used if wished. It must be easy to use and should provide instructions to TA.
8. Application must be developed using NetBeans IDE and packaged as a Maven Project. The included pom.xml file must contain any required dependencies or plugins correctly.
9. All files and codes must be stored in the Bitbucket Git repository, which will be provided and noticed for each group once created. Group should keep track of work, instead of committing files at the end of project.
10. All designs and diagrams must be stored on Confluence.
11. All tasks and updates on project must be tracked on Jira.
12. Majority of code must go through JUnit 5 tests sufficient number of times.
13. Any coding conventions should be determined by teams and applied consistently to the code.
14. Application must be executable on a Windows 10 system with Java 19.
15. Application must not have any permission to access files outside of directory where its located.
16. Visual response must be provided for every user action performed.
17. File size of the application should be limited to at most 1 gigabyte.
18. Application should run efficiently without using any unnecessary computing resources.
19. Application's GUI must be designed in consideration of the user interface and user experience design.
20. Applications must have a login account system and must be included when designing the user interface.
21. Always have the Principles of Software Engineering in mind when designing the application.

6. Summary

In this document, the Main Page is used to identify what this document is and to provide a revision history for said document. Presentation of the client's (the university) issue/request, outlining the objectives of the project, and any references that have been used to provide context to this document can be found on the Introduction Page. Providing a description of the software domain and some of the common issues encountered in said domain and their respective solutions is the purpose of the Domain Analysis page. Outlined on the Functional Requirements page are the actors involved in the scenario, the various use cases (along with their respective activity diagrams) to showcase certain features that the program should possess, and the use case diagram to display the interaction between the program and the actors (primary and external). Lastly, the Non-Functional Requirements page is used to recognize how the programmer and the software should go about doing things, the various types of software that can be used in the development of the program, and all the coding, testing, and storage practices that should be strictly followed.

Terminology	Definition
Use Case	A use case represents a set of actions performed by a system for a specific goal. It also shows how the users interact with the system.
Software domain	A domain is the targeted subject area of a computer program. The software domain would be the application that was created for the domain.
Actors	Actors are a role that the user (or external system) plays in the use case