Testing Documentation

WesternU GIS

This is a testing documentation for our map software for Western University.

Version	Date	Author (s)	Summary of Changes
0.1	April 6th, 2023	Shikhar	Made the document
0.2	April 6th,, 2023	Shikhar	Made the Test Plan
1	April 6th, 2023	Shikhar	Added and made Introduction section
1.1	April 6th, 2023	Shikhar	Added summary and reformatted document to add subpages
2.0	April 6th, 2023	Shikhar	Finalized and organized all information for tests in a table and gave them proper formatting following testing documentation specifications.
2.1	April 6th, 2023	Shikhar	Finished Testing Documentation, Ready for Review and Final Submission
2.2	April 6th, 2023	Ribal	Reviewed Testing Documentation: ▪ Added, changed, and/or removed certain sections of text to improve clarity ▪ Reviewed up to but not including text highlighted in red (the red text needs to be significantly changed or removed)

Table of Contents

Testing Documentation Introduction

Welcome to the Geographical Information System testing documentation for the University of Western Ontario Campus project! This project aims to create
a comprehensive GIS application for the university campus, including various maps and visualizations that campus administrators can use for students,
faculty, and visitors. The goal of this project is to provide a better understanding of the campus and its surroundings, identify areas that may need improvement, and guide future development and planning efforts.

Overview

The software being tested is a mapping application that allows users to view and interact with building maps. The application allows users to browse through maps and floors, search for points of interest, and display information about selected points of interest. The primary objective of this software is to provide users with an easy-to-use and effective tool for navigating buildings and finding points of interest within them.

Objectives

The general objectives of the project are to ensure that our GIS application is:

- functional
- user-friendly,
- and meets the requirements of the system.

The specific testing objectives include

- testing the accuracy
- and completeness of the maps, ensuring that the search function works correctly,
- and verifying that users can easily navigate the application.

References

The following documents provide context and assist in the understanding of this testing documentation:

- Servos, D. (2023). UWO. CS2212 Group Project Specification. https://owl.uwo.ca/access/content/group/f2f3488e-ca75-4b34-978c-0737c707927e/Team%20Project/Project%20Specification/CS2212%20Group%20Project%20Specification%20Winter%202023.pdf Accessed Feb 28, 2023.
  Accessibility Floor Plans. (2014). Faculties Management, UWO. Accessed Feb 28, 2023. https://wufloorplans.uwo.ca/
  Site Plan of Pedestrian Enclosed Walkways. (2009). Faculties Engineering, UWO. Accessed Feb https://floorplans.uwo.ca/maps/PedestrianTunnels.pdf
  28, 2023.

- Design documentation

- Requirements documentation

Test Plan:

Here's our plan for unit testing, integration testing, validation testing, and system testing for the classes in the project:

1) Unit Testing

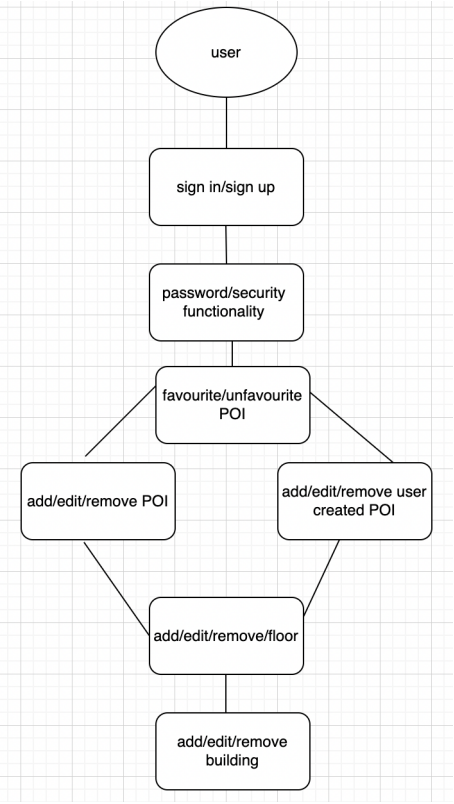
For each class, write unit tests to verify the correct functionality of individual methods and behavior of objects.

- Test that the Building class displays the correct building or location map.
- Test that the PointOfInterest class stores and retrieves information accurately.
- Test that the Layer class groups points of interest together based on category.
- Test that the User class can create their ID and set passwords.
- Test that the User class can identify develepor account correctly.
- Test that the POI class stores information about user-created points of interest, and allows for editing and removal.
- Testing the getPOI method is working as expected across all classes.
- Test that the EditingTools class allows developers to edit metadata for built-in points of interest, and that the metadata is updated correctly.
- Testing that the UserHelp button provides a comprehensive help section and an accurate About screen.

2) Integration Testing

Individual software modules (collections of classes, components, etc.) are merged and tested as a unit during integration testing. The objective is to group the unit-tested parts together and construct a tested program structure that adheres to the architectural design. Consider the class diagrams created in the requirements specification and group specific methods and classes together as a starting point based on this objective.

To ensure clusters are grouped in an efficient, logical, and complete manner, we started off with a simple flow chart representing functional dependencies, meaning which
clusters make sense to be tested together, and in what order. Consider the following sketch:



Name	**Floor object functionality and integration with building objects, and JSON files**
Test Case Description	This test case's goal is to confirm that adding, editing, and removing floors from building objects can be done correctly and that the correct JSON building object is changed in the Building JSON file.

Test Steps:	Login into a account with a developer access rights By clicking the add building button, you can add a building object to a JSON file. You can then add a floor by providing a map file and a floor number. (these are the minimum attributes to create a floor) Verify in the building JSON file that the floor has been successfully added to the appropriate building. Select the previously established building and floor on the developer screen by returning there. Change the floor number and floor image by selecting edit floor. To validate the changes, choose update. Verify in the building JSON file that the floor has been successfully modified under the appropriate building. returning to the developer panel, choose on the the right structure and the newly revised floor. Decide to eliminate the floor. Simply click "Yes, I want to delete." (Floor should be removed from the JSON file) Verify in the JSON file that the floor has been removed from the correct building. Close 'Exit' the programme by clicking the button.
Prerequisites	The user must have an account with software developer status, and signed into this account.
Expected Results	The JSON file's floor is correctly produced and shown under the appropriate building. Editing of the floor is done correctly, and changes appear under the right building within the JSON file In the JSON file, the floor is correctly erased and eliminated from the appropriate building.
Test Category	Integration test
Requirement	Software developers need to be able to easily add, edit and remove floors from building
Automation	Manual

Date Run	April 6th 2023
Pass/Fail	Pass
Test Results	The results were as expected
Remarks	N/A

Name	**Adding, editing and Removing pre-defined POIs. POI integration with floor objects and JSON files**
Test Case Description	This test case's goal is to confirm that floor POIs connected to specific floors can be added and withdrawn. The relevant floor, under the proper building, should be updated in the Buildings JSON file when adding, removing, and editing POIs.
Test Steps:	Run the Java class named Developer_Screen by selecting it. Utilize the add building button to create an object for a building (buildings should be added to JSON files), and then supply a map file and a floor number to construct a floor in the (these are the the necessary requirements to build a floor) Choose any place on the map to create a new POI. A menu will appear on the screen, asking for the name of the POI, the room number, the description, and the layer. Once the necessary data has been entered, select "Create a POI" to finalise the POI. Verify that the POI has been added to the JSON file under the appropriate building and floor. Return to the developer panel and choose the appropriate building. and the most recent POI Change the name, room number, description, and layer by selecting modify POI. After entering this data, choose update to verify the changes. Verify that the POI has been correctly changed in the JSON file under the appropriate building and floor information. Return to the developer panel and choose the appropriate option. a structure and the recently revised POI Choose "Delete POI" from the menu. Simply click "Yes, I want to delete." (POI should be removed from the JSON file) Verify in the JSON file that the POI has been deleted from the appropriate building. 'Exit' the programme by clicking the button.
Prerequisites	The user must have an account with software developer status, and signed into this account.
Expected Results	When a POI is added, the POI with the correct information appears in the building JSON file under the appropriate building and floor. After editing the POI, In the building JSON file, the modifications are listed under the appropriate building and floor. The POI is removed from the appropriate location in the building JSON file when it is deleted in the GUI.
Test Category	Integration test
Requirement	Software developers must be able to create built-in POIs of a specific layer for floors in the buildings of the application

Automation	Manual
Date Run	April 6th, 2023
Pass/Fail	Pass
Test Results	The results were as expected
	N/A

Name	**Adding/removing/editing user-created POIs. POI integration with floor objects, and JSON files**

Test Case Description	This test case's goal is to confirm that user-created POIs may be added, deleted, and modified. When user-generated POIs are added, changed, or eliminated, Accordingly, the relevant user in the user JSON file needs to be modified.
Test Steps:	Choose any place on the map to create a new POI. The following information must be entered into a menu that will appear on the screen: Description, room number, and POI's name once the necessary data has been entered By choosing "Create a POI," you can confirm the POI. Verify that the POI has been added to the JSON file under the appropriate building and floor. return to the user interface, choose the relevant building, and the recently developed POI Make changes to the name, room number, and description by selecting edit your POI. Select update after entering this data to verify the changes. Verify that the POI has been correctly changed in the JSON file under the appropriate building and floor information. Return to the user interface and choose the appropriate a structure and the recently revised POI Choose "Delete your POI" from the menu. Simply click "Yes, I want to delete." (POI should be removed from the JSON file) Verify in the JSON file that the POI has been deleted from the appropriate building. Close the programme choosing the "exit" button
Prerequisites	A building and floor object must exist and the user must have a user account they are signed into.
Expected Results	When a POI is created, the appropriate user POI list in the user JSON file contains it. When the POI is changed, the corresponding user POI list in the user JSON file displays the modifications. The user's POI list in the user JSON file is updated when a POI is deleted.
Test Category	Integration test
Requirement	Users should be able to store at least 5 created POIs. These POIs should be able to get deleted and edited
Automation	Manual
Date Run	April 6th, 2023
Pass/Fail	Pass
Test Results	The results were as expected
Remarks	N/A

Name	**Favourite/unfavouriting POIs. POI integration with floor objects, and JSON files**
Test Case Description	This test case's goal is to confirm that user- and pre-defined POIs can both be favorited and unfavorited. When a user adds a POI to their favourites list, the POI should appear in the JSON file under the relevant user's favourites list.

Test Steps:	From the drop-down menu, pick a pre-existing POI. Choose your preferred button. (star button) To see if the POI is currently a favorite, check the favourites drop-down menu. Make that the favorited POI is saved under the correct user by checking the JSON file. Return to the user interface and choose the newly-favorited POI from the drop-down menu. To unfavorite the POI, select the fave button (star button). To see if the POI is still available, look in the favourite drop-down menu. Verify the JSON file to ensure that the favorited POI has been deleted. Choose any place on the map to create a new POI. The following information must be entered into a menu that will appear on the screen: Description, room number, and POI's name once the necessary data has been entered By choosing "Create a POI," you can confirm the POI. From the drop-down menu, pick the newly created POI. Choose your preferred button. (star button) To see if the POI is currently a favorite, check the favourites drop-down menu. Make that the favorited POI is saved under the correct user by checking the JSON file. Return to the user interface and choose the newly-favorited POI from the drop-down menu. To unfavorite the POI, select the fave button (star button). To see if the POI is still available, look in the favourite drop-down menu. Verify the JSON file to ensure that the favorited POI has been deleted. Choose the "exit" button to exit the programme.
Prerequisites	A building and floor object must exist, and the user must have a user account they are signed into.
Expected Results	When a user favourites a POI, the favorited POI appears in the favourites drop-down menu and under the correct user's favourites list in the user JSON file. The user JSON file and the favourite drop-down menu should both be updated when a favourite is de-favorited.
Test Category	Integration test
Requirement	Users should be able to favourite and unfavourite POIs for easy and quick access
Automation	Manual
Date Run	April 6th, 2023
Pass/Fail	Pass
Test Results	The results were as expected
Remarks	N/A

Name	**Signing up and signing into a regular user account. User class integration with cryptography class and JSON files.**
Test Case Description	This test case's goal is to confirm that a user may register for an account and sign in. Additionally, this test case aims to confirm that users' private information, such as their favorites, and Between sessions, POIs are preserved. The correct data need to appear in the user JSON file after a user establishes an account. whenever user information If favourites or user-created POIs are modified, the JSON file should reflect the changes under the appropriate user. User passwords ought to be kept as encrypted bytes. arrays in the user's JSON files, as well as the private and public keys, ought to be kept in the appropriate files with the user's username.

Test Steps:	With a unique username and password, create a new user account. Select "sign up" and then "go to login screen" to confirm. Use the freshly created credentials to log in. Verify the JSON file to make sure the user credentials, including the encrypted byte array containing the user's password, are stored inside. Return to the user panel and add a fresh POI. Verify the newly created POI is stored under the correct user by looking at the JSON file. (no other user accounts should have this POI) Favorite this recently added POI. Verify the JSON file to make sure the newly formed POI is now also favorited. (no other user accounts should have this in their favourites) then quit the programme Start the application once more using the same login information. Verify that the fave and the POI are still listed in the appropriate drop-down menus. Verify the POI and the favourite are still stored in the JSON file under the right user by checking it. Choose the "exit" button to exit the programme.
Prerequisites	A building, floor, and built-in POI need to exist.

Expected Results	The user is able to register for an account and sign in, and a new account appears with the right data in the user JSON file. The JSON file correctly stores the POI that was produced. The user can sign in to their account once more using their previous credentials. Even when the application is ended, the POI is added to the user's favourites list and saved in the appropriate place in the JSON user file. Even after the application has been dismissed, the POI is still kept in the POI list and favourite list, both in the GUI and JSON file, under the appropriate user.
Test Category	Integration test
Requirement	Multi-user functionality, users need to be able to and log into personal accounts where information about their user created POIs and favourites is stored in between sessions
Automation	Manual
Date Run	April 6th, 2023
Pass /Fail	Pass
Test Results	The results were as expected
Remarks	N/A

Name	**Signing up and signing into a developer account. User integration with cryptography class and JSON files**
Test Case Description	This test case's goal is to confirm that accounts can be established with unique software developer access, which makes it possible to use particular features like adding and Buildings are taken down. All other accounts in the system should be updated whenever a developer makes a change. When programmers add, modify, or change The relevant information should be updated in the building JSON file if you add, change, or remove pre-defined POIs, remove buildings and floors, or do any of these things.

Test Steps:	With a unique username and password, create a new user account. Select "sign up" and then "go to login screen" to confirm. Use the freshly created credentials to log in. Verify the building's one-building status and the number of floors it possesses. Close your user account. With an unused username, a password, and the developer access code, create a new developer account. Select "sign up" and then "go to login screen" to confirm. Use the freshly created credentials to log in. Verify that the account has been saved in the JSON file with the appropriate permissions by checking the JSON file. Return to the developer screen and use the add building button to construct an object for a building (building should be added to JSON). Return to the developer screen and delete a floor from a building that already exists, not the one that was recently constructed. Verify the JSON file to make sure the new building a floor was taken out of the other building and that one has been added Sign out of the developer account Access a user account Look for the new structure. Verify in the current structure that the floor has been taken out. Close your user account. Authenticate with the developer account. Make a fresh POI (see above for in-detail description on how to create POI) Verify the JSON file to ensure a new global poi has been generated. Sign out now of developer account. Authenticate your user account. Verify the newly formed POI is valid. Choose the "exit" button to exit the programme.
Prerequisites	A regular account to exist used to sign into during testing to validate a building has been created on this remote account
Expected Results	When a new account is made, the necessary data is added to the user JSON files. (including correct Boolean for developer access or not) When a building is added, removed, or modified, the building JSON files are appropriately updated. The appropriate adjustments are applied to the JSON building file when the developer adds a new building, removes a floor from an existing building, and builds a new POI. image taken from the default user account.
Test Category	Integration test

Requirement	The ability to add, amend, and remove buildings and floors requires special access accounts that software engineers can sign into. It's important to save these adjustments between sessions. the application's other accounts, as well as being accessible to them.
Automation	Manual
Date Run	April 6th, 2023
Pass /Fail	Pass
Test Results	The results were as expected
Remarks	N/A

3) Validation Testing

Validation testing is done to make sure the right product was produced. Validation tests are used to confirm that the usability criteria and other nonfunctional requirements are met in addition to the functional requirements. Most of the functional requirements had already been thoroughly tested during integration testing. A multi-user system with sign-in and sign-out capabilities, for instance, confirms the functional requirement of having proper integration of the user JSON files and other classes when signing in to and out of accounts, and that the appropriate information is updated and saved in the correct JSON files. Since integration tests have been conducted, validation tests can be carried out after a black-box testing procedure, concentrating on the "big picture" of runnimg the application through its GUI and making sure it works as planned, as seen in the test scenarios below. Additionally, the group compiled the initial project specifications into a requirements document and went over it collectively during our last group meeting to make sure validation testing is sufficient and comprehensive.

Name	**Signing up for an account + multi-user system**
Test Case Description	creating an account and logging in. Users should have the option to create any type of account they like. After the accounts are created, the user logs in. Users ought to be allowed to construct and bookmark POIs. User-created POIs ought to be allowed to be edited and deleted by the user. It should be possible for other users to see changes made by software developers. If a frequent user modifies something, just that person's account should display the update.
Test Steps:	With a unique username and password, create a new user account. Select "sign up" and then "go to login screen" to confirm. Use these credentials to log in. In any of the buildings, create a new POI, and then like it. Choose the "exit" button to exit the programme. Create a new developer account by choosing "sign up" and then "go to login screen," then confirm by entering an unused username, a password, and the correct developer key. Use these credentials to log in. Verify that the recently added POI from the user account is not included among the global POIs in the global POIs menu. Establish a worldwide POI Choose the "exit" button to exit the programme. Re-enter your user name in the programme. Verify that the newly established global POI and the user-made POI are both visible. Change the POI the POI to your favourites Choose the "exit" button to exit the programme. open or create a different user account Verify that there are no favourites from other accounts and that there are only global POIs, including the one created during this test. Choose the "exit" button to exit the programme.
Prerequisites	The application is downloaded on the operating system, and the existing metadata file has been run to ensure there are buildings and POIs to be worked with by the developer.
Expected Results	Both recently established accounts have been configured with the appropriate permission levels. The user can add/edit POIs, add and remove favorites, and more. The developer has the ability to alter building attributes and create/update/delete POIs that are visible to all users.
Test Category	Validation testing
Requirement	The application is required to support multiple users were each user account has their own POIs that are not shared with any other users. Furthermore, developers are able to add/edit/delete POIs and these changes should be visible to all users.
Automation	Manual
Date Run	April 6th, 2023

Pass /Fail	Pass
Test Results	The results were as expected

Remarks	N/A

Name	**Software developers can create, edit and remove buildings and the floors/POIs in the building**
Test Case Description	Users can only sign up for a software developer if they have the proper credentials. Users that input the incorrect secret key should not be allowed to create a software developer account. Try with the wrong private key, and just like go in and try a few features, or you can say something like.
Test Steps:	Try setting up a fresh developer account. Enter the wrong developer key along with the (new) login and password. There will be a mistake. Try to create the developer account once more, but this time, enter the right key. Use the newly established developer account to log in. Add a structure Add a floor and a worldwide POI to the just built building. Edit the just made global POI. Choose the "exit" button to exit the programme.
Prerequisites	The application is downloaded on the operating system, and the existing metadata file has been run to ensure there are buildings and POIs to be worked with by the newly created developer account.
Expected Results	The developer account can only be created using the valid developer key The developer can edit/create/delete building, floors and POIs
Test Category	Validation testing
Requirement	The application must support a developer account that has admin permissions. With these permissions the developer can edit/create /delete building, floors and POIs.
Automation	Manual
Date Run	April 6th, 2023
Pass /Fail	Pass
Test Results	The results were as expected
Remarks	This requirement has been validated multiple times in other parts of the testing, this test case servers as a more specific test performed for the related classes

Name	**Users browsing and searching the map**
Test Case Description	User can browse and search the maps in the application (favourite and create/edit POI, easily switching between the maps and buildings, selecting a POI to show that it can be used for quick and easy access)

Test Steps:	With a unique username and password, create a new user account. Select "sign up" and then "go to login screen" to confirm. Use the freshly created credentials to log in. Look through the structures and pick one. Now, using your mouse, click anywhere on the map to add a POI. (see above for complete steps on how to make a POI as a user) Go to a random floor in a different building by choosing it from the menu. Click the freshly added POI for the first building in the POI drop-down menu. Edit the POI in the first building where it has been created now. Following POI favourite it's modification Choose a different building from the drop-down option once more. Visit the favourites drop-down menu from the new building and choose the just added favourite. Then, remove the favourite and make sure it is no longer in the favourites drop-down menu. Choose the "exit" button to exit the programme.
Prerequisites	Two buildings with floors and POIs must be created prior to running this test
Expected Results	The desired permissions are successfully added to a brand-new user account. The user has the option to create and change POIs. Using the drop-down menu, the user can switch between structures. The user can choose their POI and get sent there right away.
Test Category	Validation testing
Requirement	The interface needs to be intuitive and pleasant to use. User's need to easily be able to browse and switch between maps. Selecting POIs and favourites should be easily and up the correct map with the correct layer
Automation	Manual
Date Run	April 6th, 2023
Pass/Fail	Pass
Test Results	The results were as expected
Remarks	N/A

Name	**Housekeeping**
Test Case Description	The user experience should be smooth and intuitive. The application should be easy to exit, there should be a help menu if the user gets stuck, and there should be helpful error messages when an account cannot be created, or the sign-in information is not correct. Developers are forced to confirm before a big change is saved, such as when deleting a building.
Test Steps:	Create a new developer account by choosing "sign up" and then "go to login screen," then confirm using an unused username, a password, and the developer key. When attempting to sign in, the erroneous password is entered. Sign in using the correct information after dismissing the error notice stating that your username or password is incorrect. after logging in, choose the "help" option, go through it, and then click "x" Create a new building by clicking the "add building",then Select "Delete Building Button" at this point. Select "YES, I'm sure" to confirm this procedure. Choose the "exit" button to exit the programme.
Prerequisites	The application must be downloaded and the metadata file needs to have been run.
Expected Results	The ability to create a new account with the appropriate permissions is available to the developer. A notification advising the developer of the mistake is displayed if they provide the incorrect credentials. The developer can add or remove buildings once they have signed in. A confirmation popup is sent to the developer when they attempt to delete a building to make sure they wish to do so.
Test Category	Validation testing
Requirement	The user experience should be smooth and intuitive. There needs to be a help menu for when user's get stuck, and the system needs to be able to be easily closed, safely

Automation	Manual

Date Run	April 6th, 2023
Pass /Fail	Pass
Test Results	The results were as expected
Remarks	This feature has been validated multiple times while testing other components

4) System Testing

When the software is combined with additional system components like hardware, people, information, and processes, this is known as system testing. As we progressed up to higher-order clusters, we eventually performed system-wide integration testing because the integration tests were carried out using a bottom-up methodology. This means that testing doesn't have to be lengthy because our software simply needs to run on a local computer system.

Hardware, individuals, and processes should all be included in system testing. The system should be simple and affordable to install, run on the proper hardware system—in this case, a Windows computer. Additionally, a system's defence mechanisms must function as planned. System testing has a higher likelihood of being successfully integrated at this stage of the testing strategy because it has been implemented during software design and earlier testing components.

Here are few cases that we would like to go over:

Name	**Deployment testing**
Test Case Description	Opening up the zip folder with all project components and following the instructions in the readme file to install and run the application. This will ensure all vital parts of the project are included in the zip folder for the application to properly run. Once the application is installed, created an account and run through the application to ensure no major bugs propagated during the installation process.
Test Steps:	Extract the file from the zip folder using the desktop. Launch your preferred IDE and start a new project. Import the file from the zip that was extracted once this project has been created. The json-20220924_1 file should be added to your build and run paths. Activate the metadata file. Create a new developer account by choosing "sign up" and then "go to login screen," then confirm using an unused username, a password, and the developer key. Create a new building by clicking the "add building" button. The building will now be deleted after you click the "delete building button." Add a worldwide POI Choose the "exit" button to exit the programme. With a unique username and password, create a new user account. Select "sign up" and then "go to login screen" to confirm. Use the freshly created credentials to log in. Look through the structures and pick one. Choose a few of the POIs for this building, then make one. Then, choose another building from the drop-down option. Create a new POI in the new structure. Choose the POI you generated in the first building from the POI menu. Change this POI Choose the "exit" button to exit the programme.

Prerequisites	Windows operating system, an IDE that runs java and the ZIP file with the project in it.
Expected Results	With the JRE, the programme can run on any machine. You can include the JSON file in the run path. The area of the developer and user accounts was made with the appropriate permissions. The user can add/remove favourites and create/edit POIs. Buildings, floors, and POIs can all be created, edited, or removed by the developer.
Test Category	System Testing

Requirement	The application needs to be contained in a folder with all required libraries and and a read-me file with instructions. The application should be easy and attainable to properly install
Automation	Manually
Date Run	April 7th, 2023
Pass /Fail	Pass
Test Results	The results were as expected
Remarks	N/A

Name	**Security Testing**
Test Case Description	Testing that the security features in place work as intended. When user's create passwords, the passwords are stored as an encrypted byte array in the user JSON file. Security testing can be performed to confirm user's passwords are not being stored as the string they inputed, that user's can only sign into accounts for which they know the valid user name and password for, and software developer accounts can only be created with the appropriate private.
Test Steps:	With a unique username and password, create a new user account. Select "sign up" and then "go to login screen" to confirm. You're trying to log in, but your password is wrong. Get rid of the error message Try signing in right away with the right password. Verify the JSON file to make sure the password was encrypted and not just stored as the password itself. Choose the "exit" button to exit the programme. Create a new developer account with the incorrect developer key, an unused username, and a password. The account won't be established, and an error message is displayed. Get rid of the error message Create a new developer account once more with the proper developer key, a password, and an unused username. Verify by choosing "sign up" and then "go to login screen." You're trying to log in, but your password is wrong. Get rid of the error message Try signing in right away with the right password. Verify the JSON file to make sure the password has been encrypted and not saved as a string of the actual password and to make sure this account has developer permissions. Choose the "exit" button to exit the programme.
Prerequisites	The JSON files that will store the user data have been created
Expected Results	The JSON file contains the user's or developer's information when they create an account. Account passwords are kept by the user, however they are encrypted and do not display the actual password. If a user or developer types in the wrong username or password, an error message advising them of the wrong credentials is displayed.
Test Category	System Testing
Requirement	Extra feature: secure user accounts. Developer access should not be attainable for regular users
Automation	Manually

Date Run	April 7th, 2023

Pass /Fail	Pass
Test Results	The results were as expected
Remarks	The security functionally feature has been tested multiple times during other tests as well. This test provides an example.

Name	**Stress Testing**
Test Case Description	Create 5 user created POIs, and favourite 10 POIs under one account. Log out, and sign back in and make sure their information has been properly saved.
Test Step:	With a unique username and password, create a new user account. Select "sign up" and then "go to login screen" to confirm. Utilize the newly established account to log into the programme. Make 5 POIs. Favorite the five global POIs and the freshly added POIs. Choose the "exit" button to exit the programme. Re-enter the account and verify that your saved POIs and preferences are still there. Choose the "exit" button to exit the programme.
Prerequisites	There has to be a building with floors and built-in POIs (at least 5 built in, so the user can favourite 10 total POIs)
Expected Results	All POIs and favourites are properly created and stored.
Test Category	System Testing
Requirement	Information about user-created and favourites in one user account should not be viewable from other user accounts. The application must be able to support at least 5 user-created POIs, and 10 favourites.
Automation	Manually
Date Run	April 7th, 2023
Pass /Fail	Pass
Test Results	The results were as expected
Remarks	A user "technically" can create more than 10 User-Created POIs and favour more that 10 build-in POIs; however, it will sacrifice the functionality of displaying the POIs for a layer, as the POI label array has a size of 10.

Summary

The testing documentation for the GIS project

- outlines the objectives of the project,
- provides a description of the Test
- and describes the responsibilities of each class and their collaborators.

It serves as a reference for testing the software and ensuring it meets the requirements and objectives of the project.