Testing Documentation

1. Testing Documentation

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

1.1.1. Revision History

Version	Date	Author	Summary of Changes
1.0	Wed Nov 30, 12:50pm	Taoran Gong	Created original document, created headings for individual sections of the document as well as table of contents
1.0.1	Wed Nov 30, 1:05pm	Taoran Gong, Ryan Cook	Added the first draft of introduction
1.0.2	Wed Nov 30, 4:45pm	Taoran Gong	Added the first draft of test cases
1.0.3	Thu Dec 1, 5:30pm	Taoran Gong	Edited the Unit Testing plan.
1.0.4	Thu Dec 1, 6:20 pm	Ryan Cook	Created the first draft of the validation test
1.0.5	Thu Dec 1, 9:15 pm	Taoran Gong	Created the first draft of the integration testing plan
1.0.6	Fri Dec 2, 1:50 pm	Ryan Cook	Finished up the validation tests
1.0.7	Fri Deb 2, 2:10 pm	Ryan Cook	Final edits on the testing document

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.

We need to constantly test and optimize to ensure that our uwoMaps can work properly and also successfully output the correct results. In view of our pursuit of software operation, we will use both previous knowledge and materials from this course in this Testing Document to design test plans and test cases and record the results. In this Testing Documentation, we will apply unit testing, integration testing, validation testing and system testing. In addition, we will also use a mixture of black box testing and white box testing to support the above tests.

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
- Having sufficient and complete test plans to evaluate our work
- Using test cases to access the quality of the software

2.1.3. References:

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

3. Test Plans

3.1.1. Unit Testing

Test Case Name:	Unit Testing (General)
Test Case Description	The unit testing will be used to determine whether the individual units of source code are good to use. JUnit will be used for unit testing.
Test Steps	1. Having all classes/methods ready for unit testing 2. Writing JUnit test files for each core class/method 3. Run JUnit test and record the results 4. Make changes to our code if necessary
Pre-Requisites	The pre-requisites for this unit testing is that we must have all classes and JUnit files ready.
Expected Result	Each class and method work properly and produce the correct result
Test Category	Unit Testing
Requirement	JUnit is used to make sure each method and class work.
Automation	automated using JUnit
Date Run	Nov 30 to Dec 1, 2022
Pass/Fail	Pass
Test Results	At the beginning there were concerns about certain methods, but after communication within the group, we were able to figure it out. The test results are pleasing: it shows that all methods/classes are producing the correct results. At the unit testing level, our software runs well and generates the correct result.
Remarks	• white box testing, only on core classes(confirmed by professor) • automated tests are written in JUnits • After unit testing, we are confident to move forward to integration testing.

3.1.2. Integration Testing

Test Case Name:	Integration Testing
Test Case Description	Integration testing combines software modules and test them as a group.
Test Steps	

Pre-Requisites	Parts of the software program should be ready for testing
Expected Result	Each part of the software should be running on their own
Test Category	login page, the uwomap page, building page, floor page, POIs
Requirement	
Automation	manually
Date Run	December 2nd 2022
Pass/Fail	Pass
Test Results	Login Page: user enters username and password to access their account. Forget password method is also working Map page: The list of available buildings are shown including an edit button for users with access to the admin account Building: The building is shown on the page, able to zoom in, zoom out, information is stored Floor: POIs are shown with different layers, information of floor is stored in variables like numofFloors,etc. POI: POI can be shown/hidden, user is allowed to save their own selected POI Main Page: options allow for the user to search for a POI, go to the map page, go to a list of their favourite POIs and seeing today's weather
Remarks	In Integration Testing, top-down testing and white box testing method is used. Can move forward to Validation Testing

3.1.3. Validation Testing

For validation testing, multiple use cases from the requirement documentation is going to be tested to see if the software outputs to expected result. For any cases involving storing or searching for a POI, washroom 228 on the 2nd floor of the health science building is going used to validate the software.

These use cases are going to be:

- Selecting a building
- Searching for a POI
- Selecting a layer
- Storing a POI as favourite
- Creating a POI personal to the user
- Editing Tool/Mode
- Multi-user system
- Extra Classroom metadata

Test Case Name:	**Selecting a building**
Test Case Description	The user selects a building from the campus map page and is brought the that building's page
Test Steps	• In the map page, the user clicks on the button "health science building" and is brought to the health science building page
Pre-Requisites	The user must be on the map page
Expected Result	The user is brought to the building page of the building they selected. This page has a picture of the building on the campus map and buttons for each floor in that building
Test Category	Building
Requirement	Selecting a building from the use cases
Automation	manually
Date Run	December 2nd, 2022
Pass/Fail	Pass
Test Results	The user is brought to the brought to the health science building page
Remarks	

Test Case Name:	Selecting a layer
Test Case Description	The user select a layer (accessibility or washroom) that they can view
Test Steps	• The user navigates to floor 2 in the health science building • The user can then select the washroom layer to see where the washrooms are on this floor
Pre-Requisites	The user must be on the floor page to be able to set a layer
Expected Result	The map that is displayed shows where the washrooms are on the floor but does not show any accessibility information
Test Category	Floor/POI
Requirement	Built in POIs from use cases
Automation	manually
Date Run	December 2nd, 2022
Pass/Fail	Pass
Test Results	Once the washroom layer is selected, all the washroom on floor 2 of the health science building are highlighted
Remarks	

Test Case Name:	Searching for a POI
Test Case Description	The user types the name of a POI in the search bar in hopes of seeing its location
Test Steps	• On the main page, the user types in the POI they want to search for in the search bar which in this case will be washroom 228 (male washroom on floor 2 of the health science building) • The POIs matching the search come up • The user can then click on the POI they would like to see • The POI is displayed for the user on the map
Pre-Requisites	The user must be on the main page The user is searching for a POI that exist
Expected Result	The POI that the user is searching for comes up with a little description that the user can click on. Clicking on the POI highlights it on the map
Test Category	POI
Requirement	Searching a POI from the use cases
Automation	manually
Date Run	December 2nd, 2022
Pass/Fail	Pass
Test Results	When the user searches for washroom 228, a drop down menu comes down with this washroom as an option. When the user clicks on this POI, the user is brought to the health science building floor 2 map where washroom 228 is highlighted
Remarks	

Test Case Name:	Storing a POI as favourite
Test Case Description	The user stores as a POI as their favourite
Test Steps	• The user navigates to or searches up a POI of their choice, in this case the user searches for washroom 228. • The user then clicks on this POI and is brought to the health science building floor 2 page when the user can see this POI • The user then can click onto the favourite toggle to add this POI as one of their favourites • From the main page the user can navigate to see their favorite POIs from clicking on "Go to my favourites" • The POI they selected is then listed on that page

Pre-Requisites	The user must on the POI page
Expected Result	The POI that the user selected is stored as a favorite in the favorite page
Test Category	POI
Requirement	Storing a POI from the use cases
Automation	manually
Date Run	December 2nd, 2022
Pass/Fail	Pass
Test Results	Washroom 228 is listed as a favourite POI in this favourite POI page
Remarks	

Test Case Name:	**Creating a POI personal to the user**
Test Case Description	The user creates their own POI that they can store and use within the application
Test Steps	• The user can click anywhere on the floor map where they want to place their POI, in this case the user will navigate to the health science building floor 2 page and click anywhere to create a POI of their choice, assume that the user click on room 222 • A pin will drop where they selected and then there will be a prompt asking the user for input • Assume that the user inputs room 222 as a class called class 222 and input a description of their choice, let's say "this classroom has a capacity of 20 and is always available at 2pm"
Pre-Requisites	The user must be on a floor in one of the available buildings
Expected Result	The POI is created and stored in the app as a user created POI
Test Category	POI
Requirement	Creating a POI personal to the user as a use cases
Automation	manually
Date Run	December 2nd, 2022
Pass/Fail	Pass
Test Results	On the health science building floor 2 page, there is a POI called class 222 with the description "this classroom has a capacity of 20 and is always available at 2pm" that highlights room 222 when clicked on
Remarks	

Test Case Name:	**Editing Tool/Mode**
Test Case Description	Users with access to the admin account can remove built-in or user created POIs
Test Steps	• The user opens the program • The user enters a password and username and can log in into their admin account from the login page • On the main page, there is a button called "edit" that regular users cannot see • The user will then be able to remove POIs of their choice • The user searches for a male washroom and selects the male washroom on HSB floor 2 (washroom 228) • The user can select an remove button to remove the POI • The user is prompted again to make sure they want to remove that POI • Once the user logs out of the program, any user that refreshes the program will not the see the male washroom on HSB-2 (washroom 228)
Pre-Requisites	The user must be in the admin account to be able to access the edit button
Expected Result	Any user that refreshes the program can not see the male washroom on HSB floor 2 (washroom 228)

Test Category	Editing
Requirement	Editing tool/mode from the use cases
Automation	manually
Date Run	December 2nd, 2022
Pass/Fail	Pass
Test Results	After the user removed washroom 228 and the program was refreshed, a user logged in and searched for washroom 228. Washroom 228 did not come up
Remarks	

Test Case Name:	**Multi-user System**
Test Case Description	The program can allow multiple users to use the program simultaneously
Test Steps	• Each of the users opens the application • Each of the users enters their account information (username + password) • They log in to their own accounts • The users can then explore all features of the applications • User 1 saves the male washroom on HSB floor 2 as a favourite
Pre-Requisites	Each user must have their own account
Expected Result	User 1 will have the male washroom on HSB floor 2 (washroom 228) saved as a favourite but this won't be the same case for User 2
Test Category	Multi-user System
Requirement	Multi-user System from the use cases
Automation	manually
Date Run	December 2nd, 2022
Pass/Fail	Pass
Test Results	Once two users logged in and user 1 went to their favourites page, they could see washroom 228 as a favourite. On the other hand, once user 2 navigated to the favourites page, user 2 did not see washroom 228 listed as a user
Remarks	

Test Case Name:	**Extra Classroom Metadata**
Test Case Description	The user can select a lecture room POI and receive various information on that classroom
Test Steps	• The user navigates to the health science building and then to floor 2 • The user selects lecture room HSB-236 from the available POIs on this floor • The user then sees that the room is highlighted and can see that the seating capacity in this room is 150
Pre-Requisites	The user must either search up or navigate to the POI
Expected Result	The user can see this POI category type lecture and see that this room has a seating capacity of 150
Test Category	POI
Requirement	Extra classroom metadata from use cases
Automation	manually

Date Run	December 2nd, 2022
Pass/Fail	Pass
Test Results	Once the user navigated to health science building floor 2 and selected HSB-236, HSB-236 is highlighted and the user can see a description that says that the room capacity is 150 among other information
Remarks	

3.1.4. System Testing

Test Case Name:	**System Testing**
Test Case Description	▪ In system testing the complete software will be tested as a whole ▪ Black box testing method will be used
Test Steps	1. Starting from login page, log in with username/password, try the reset password 2. Go into the main page, browse around the map, zoom in/ zoom out 3. Select one of the three buildings, showing information about interior space of that building 4. Inside building, show POIs such as accessibility and washrooms, etc., also pictures and data will be presented 5. Save and unsave user's favourite POI 6. Help function & Exit the program
Pre-Requisites	• Passed unit testing, integration testing, and validation testing
Expected Result	The whole software works together to produce the uwoMaps. Although units of methods/class, integration, and validation have already been tested, system testing is still required to see if the whole software is able to run.
Test Category	Function Testing: are all our functions working?(such as login and save favourite) Performance Testing/ Non-function Testing: run-performance (is it slow/fast with the context of an integrated system, is it a secured software) UI Testing: the visibility of the application: will look at the readability and the view of the software Usability Testing: Is it easy for users to explore the software ?
Requirement	The whole software program must be ready to use
Automation	manually
Date Run	December 2nd, 2022
Pass /Fail	Pass
Test Results	Function Testing: All the functions work well and produce the correct result. Performance Testing: It is fast to run our software with the context of an integrated system. With multi-user access functionality, we are confident to say that we have a secured system. Users can enjoy our program in a safe and fast environment. UI Testing: Although web design and user experience can be very subjective, we are proud to say that our web pages are very simple and easy to use, and the typography is clear and easy to follow. We used the UX design learned in CS2212, making the interface consistent and placing the user in control for our software. Usability Testing: The program is easy to use and self explanatory. Also the exit button is shown on the page as well.
Remarks	The system testing is based on the date run

4. Summary

After completing the requirement documentation and design documentation, our group has been developing the uwoMaps software. We discussed in a meeting what each java class needs to do and how we want to achieve it, and then divided the work for everyone. When each of us pushed the code we wrote to bitbucket, we started testing our program.

This testing process mainly consists of four tests: Unit Testing, Integration Testing, Validation Testing, and System Testing. We first used unit testing to ensure that each class and each method can produce correct results. Although we only submitted the JUnits file on core classes, which is confirmed by the instructor, we have also verified the rest of the classes by ourselves to ensure that we can pass the unit testing on all classes and methods before entering the integration testing. It is also worth mentioning that the white box testing method is used in unit testing. The second test is integration testing. This is mainly to test whether the program can run successfully after integrating several classes. For example, our login is composed of different classes and methods, so the login page can be implemented through integration testing. We are also able to tell whether our login can correctly identify users. Of course, in integration testing, we only focus on the operation of units composed of several classes and methods, rather than combined units running together as a whole. There are several different ways to implement integration testing, our group used the top down testing method. After unit testing and integration testing, we had validation testing. In this step, we will verify whether these use cases can be implemented and produce correct results. For example, when the user is on the map page, he/ she can click on a button that has the name of the building they would like to go to, then the user is brought to the building page of the building they selected. We considered different use cases and tested the program, and the results were all satisfactory, so we also entered the last test: system testing. In system testing, this software will be tested as a whole completed program. Since we all use the software as a backend user, this test is black box testing. In system testing, there are many considerations: we not only evaluate functional and non-functional requirements, but also give a lot of thoughts about UI and usability implementations. We want to make sure that while the software is running and producing the correct results, the user will also have a good experience using it. Finally, the system testing is also completed.

After passing unit testing, integration testing, validation testing, and system testing, uwoMaps is officially completed. These tests are not only a great help to the software, but also make each of us grow into a more mature software engineer, who can evaluate the whole process of a software development from a more comprehensive perspective.