4 Testing

Testing is an essential part of any software development project, and it is particularly crucial for a web application like ours that aims to help users find the optimal living location by creating heatmaps based on their daily trips. And to ensure that our application functions correctly and meets user requirements, we need a comprehensive testing strategy that covers all aspects of our system's design. One of the unique challenges to testing our system is the reliance on third-party APIs, particularly Google APIs, to generate the heatmaps. Since these APIs are external to our system, they can be subject to unexpected changes, which can impact our application's functionality. So, we need to ensure that our testing strategy includes detailed testing of the integration with these APIs. Another challenge in testing our system is the diverse range of user inputs that our application needs to handle, including daily trip frequency and timing. This makes it essential to perform extensive testing of the input validation and processing functions to ensure that the application can handle a wide variety of inputs and produce accurate heatmaps. We are going to address these challenges and ensure the overall quality of our application by implementing a testing strategy that includes a range of testing techniques, including unit testing, integration testing, interface testing, and user acceptance testing.

4.1 Unit Testing

1. User Account Creation and Management: We will create test cases for user account creation, login, logout, and password reset functionalities. Which can be done using tools like JUnit.

2. Input Validation Functions: We will test the functions responsible for validating user inputs, including input data types, format, and range. Which can be done using tools like JUnit or NUnit.

3. Heatmap Generation: We will test it by creating test cases for generating heatmaps based on user inputs and verifying that the heatmaps are generated correctly. Which can be done using testing frameworks like JUnit, NUnit, or TestNG. Outputed heat maps can also be verified by comparing the weights to hand generated versions. Hand generated versions will be calculated using google maps to find the commute time from the start to each location and taking the average. This should match the heat map weight.

4. API Integration: we will test the functions responsible for integrating with Google APIs to generate heatmaps based on user inputs. Which can be done using testing frameworks that support API testing, like Postman.

5. User Authentication and Authorization: we will test the functions responsible for user authentication and authorization. Which can be done using frameworks like Spring Security.

When testing each unit, we will consider testing both the expected behavior and the boundary conditions, which will include testing both valid and invalid inputs and edge cases that test the limits of our application. And to perform these tests, we are going to use various testing frameworks and tools, such as JUnit, NUnit, or TestNG. These frameworks can be integrated with our build and deployment process to automate testing.

4.2 Interface Testing

1. User Interface: This interface allows users to create accounts, provide input for daily trips frequency and timing, view the generated heatmaps, and manage their accounts. We need to test the user interface of our web application to ensure that it works as expected. We will use tools like Selenium, Cypress, or TestCafe. We wil also deploy a beta version of this application for review by realtors. The review process has been approved by our client.

2. Google API Interface: This interface enables our web application to integrate with Google APIs to generate heatmaps based on user inputs. We need to test the integration between our web application and Google APIs. We will use tools like Postman or SoapUI to create API tests and ensure that the API calls return the expected results.

3. Finally, we need to test the interaction between the user interface and Google API interfaces to ensure that our web application works as expected. We can do this by monitoring data results from the integrated API and comparing them to the expected results.

4.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

The first critical path in our design is the input and output streams to and from the used API's. This is because we do not have full control over how the data will be manipulated and what the results will be. We will test this by first using a bare bones framework to run several API requests and monitor output. Requests will be generated based on normal and edge case inputs (i.e. a single location, or a very large amount of locations spread between a great distance). We will use tools such as JUnit and NUnit to test for exceptions by comparing the structure of the output to the expected structure. After ensuring the edge case and normal tests match our expected structure for data returns, we will them generate heat maps from these and compare to hand created ones described above.

The second critical path is the user interface and input stream on the client end.

4.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

System testing will be done using the BrowserStack suite of tools. These tools will alow us to view the website across 300 different platforms including mobile based. BrowserStack will also allow us to test the responsiveness of the website to ensure we maintain usability across screen and device types. Overall functionality will be tested by generating heatmaps on each of the 300 browsers and comparing them either manually or through image recognition to determine if any meaningful differences arise. The BrowserStack application ill allow us to test our design on each platform concurrently by simulating each of the requests from a different browser type.

A beta version of this application will be reviewed by currently practicing realtors to ensure functionality meets the demands of our users.

4.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure they do not break? Is it driven by requirements? Tools?

We have created our site as modularly as possible and used general CSS styles for buttons to ensure future additions will adhere to our current design. The critical features implemented in this program are the integration of the APIs. The modularity allows us to make modifications to the functions without disrupting the current progress. Front-end visual or structural changes will need to be tested as stated above on 4.4 for browser support and responsiveness. We will reuse the BrowserStack suite of tools to ensure any additions to the visual or structural design in the front end does not hamper the end result.

Tests created using the BrowserStack testing tools will be automated and used to determine whether new functionality breaks old functionality.

4.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

Our client has been involved within our acceptance testing through periodic meetings. We have met once a week to update progress made. While we display these updates we are receiving feedback on design choices. This will help make sure that non-functional requirements are met. As development progresses these meetings will continue. As functional features are added, we will test then receive feedback from the client.

Additionally, we have received information that there is interest in our product by members of the housing and real estate industry. We are planning to hold a showcase once the product is further developed to receive feedback.

4.7 Security Testing (if applicable)

Our project will be dealing with user information including their frequented locations as well as listed housing information, because of this we need to assure that our user's sessions and data are properly encrypted, and that our site is safe from injections and other common vulnerabilities. We will test for a variety of potential web-based security issues using various penetration testing tools including Burp Suite, SQLMap, Nikto, and more. We plan to use the OWASP Top Ten list of web-application vulnerabilities for a baseline of vulnerabilities to test for using these tools. This list highlights broken access control, cryptographic failures, site injections, and request forgeries as some of the most common vulnerabilities on the web.

4.8 Results

What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.