**HelloWorldByMe**

Software Testing Plan



April 4th, 2025

**Project Sponsor:**

Kevin Daily

**Faculty Mentor(s):**

Brian Donnelly, Savannah Chappus

**Team Members:**

Elizabeth Knight

Joey Banaszak

Jessica Maldonado Olivas

Samantha Madderom

Version 1

**Table of Contents**

[**Introduction 3**](#_ef1cjv3tbehs)

[**Unit Testing 3**](#_yo98dj3bt13g)

[**Integration Testing 3**](#_dnncz9rfxrud)

[**Usability/End-User Testing 3**](#_hhziuv5i7eij)

[**Conclusion 3**](#_6l81i7mjx25o)

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# **Introduction**

The number of individuals experiencing homelessness on a single night was at an all-time high in January 2024, totaling 771,480. Of these, nearly 20% displayed patterns of chronic homelessness. Unlike individuals who experience temporary homelessness due to financial hardship or other unforeseen circumstances, those suffering from chronic homelessness live on the streets for at least a full year, and often longer. This type of homelessness is not just a matter of financial instability but is typically related to more complex and enduring issues. People often experience chronic homelessness because of severe mental illness, substance use disorders, physical disabilities, or other medical conditions. Once they are on the streets, it can be extremely difficult and complicated for them to get back into stable housing.

Our client, Kevin Daily, lives in Tuscon, Arizona, and is an active member of his community. He, along with local restaurant owner Josh Jacobson, has put in volunteer hours at many shelters and organizations focused on helping people on the streets. They have witnessed firsthand the struggles of the homeless, as well as the lack of communication between these organizations. As a result, they have put together a project to try and alleviate this issue, starting in Pima County, for a few reasons. Firstly, there has been a massive increase in homelessness in Pima County in particular, rising 300% between 2019 and 2020. It is also the origin of the fentanyl crisis in Arizona, with 510 people overdosing in 2023.

When it comes to helping the homeless, many organizations offer services to these people. However, they are often working individually toward the same goal. There should be more collaboration among these entities to provide better and more efficient care and services to these people. Our solution is to create a web application where service provider entities can share information and resources to better assist individuals who are on the streets, particularly those who are chronically homeless. This includes an internal messaging platform for cross-organization communication and location services for accurate and up-to-date location tracking. We hope our solution will enable greater communication and collaboration between service providers so that more people can get the help they need.

“Software testing is the process of evaluating and verifying that a software product or application does what its supposed to do.” Good software testing will look out for potential bugs in the code, checking for edge cases where the logic may fail. Software testing can be a tool to use to ensure the product you’ve created meets the requirements of the project. There are several different types of testing with different objectives. The ones we will be focusing on for this project are:

* *Unit Testing*: Validating that the individual software components work as expected. A unit is the smallest testable component of an application.
* *Integration Testing*: Validating that the software components function together as expected.
* *Usability/End-User Testing*: Validating how easily a customer can use the product to complete a task.

These are the most important types of testing for us to do because of the nature of our project. Our solution contains several different elements that we developed independently to begin with. As a result, we will need to use unit testing to catch bugs in the different components early on so the issues aren’t exaggerated by integration. We need integration testing to ensure that all of our components are connected together properly and can work in conjunction without breaking. Finally, we must ensure that our solution is not only functional but also easy to use. Our goal was to design our system in such a way that minimal training would be required to use it. We will use usability testing to ensure that we accomplished that.

# 

# 

# **Unit Testing**

As one of the most common ways to ensure constant rapid problem solving, unit testing consists of testing out the smallest functional units of code. The main purpose of unit testing involves verifying that individual components of a software are functioning as intended and discovering hurtful bugs early. If one of the components fails the automated unit tests in any way, developers can isolate and troubleshoot the root cause of the failures. Utilizing a unit testing process highlights “modular thinking paradigms and improves test coverage and quality”, allowing developers to spend less time debugging and more time coding more features. (“What is Unit Testing?”) This well-known practice could lead to scalable, maintainable code, as any problem that arises can be easily found and resolved directly. By incorporating unit testing into our testing and validation stage, we will be able to increase code stability and ensure that future work doesn’t unintentionally damage the already existing functionalities.

Our unit testing frameworks/libraries consist of:

* **Jest** 
  + A JavaScript testing framework that utilizes **code coverage** as the main metric. The percentage of the statements, decision branches, functions, and lines are covered. Jest is also known to be fast, efficient, and has extensive documentation.
* **React Testing Library**
  + A React-based testing library that takes on an end-user approach to unit testing. Instead of the typical number-heavy metric, React Testing Library focuses on the **overall software’s behavior**, such as UI component functionality, expected user behavior, accessibility of elements, and state and prop handling.
* **Mocha** 
  + A JavaScript testing framework that doesn’t provide measurement of any kind, but structure to the testing process. However, it is known to have compatibility with libraries that measure code coverage, performance, and mocking.

***Navigation***

The navigation component is essential for the clients themselves, ensuring that they can get the help they need on time. Utilizing the Jest testing framework, the purpose of the navigation system will be to ensure that it functions correctly and meets specific requirements, including adding, retrieving, selecting, and deleting clients as information is updated. With those functionalities in mind, the main functions that will be tested include:

**Navigator Input Form**

* saveClient(): function that will save each input field and store in temp object
* searchClient(): function that looks through all existing users for a match
* addNote(): function that allows for additional info about client through a textarea
* clearForm(): function that resets the form

**Mapping Interface**

* initializeMap(): function that initializes map elements
* searchMapClient(): function that searches and retrieves existing user mapping data
* loadClient(person): function that retrieves user’s input field data and their markers
* saveMarker(name, age, gender, situation, lat, lng): function that saves markers that are already there and markers that are being made
* placeMarker(event): function that allows for markers to be put on map

**Service Matching**

* matching(client): function that defines the matching logic based on user and shelters’ availability
* display(matches): function that displays results of service matching

**Equivalence Partitions**

| Input | Valid Partition | Invalid Partition |
| --- | --- | --- |
| Client Names | Text input | Empty string, too long, and just whitespace |
| Age | Integer between 1-120 | 0, 120+ |
| Gender | “M”, “F”, “N/A” | Letters that don’t match |
| Situation Description | Nonempty text input | Empty string, too long, just whitespace |
| Coordinates | Real latitude and longitude | NULL, or invalid coords |

**Boundary Testing**

* Client name
  + Shortest valid: “A”
  + Longest invalid: “A” \* 256
* Age
  + Minimum: 1
  + Maximum: 120
* Gender
  + Valid genders: “F”, “M”, “N/A”
  + Could be more flexible in the future, but for right now, these are available
* Situation
  + Valid short input: “homeless”
  + Invalid input: “”
* Coordinates
  + Min and max latitudes and longitudes possible

**Sample Test Cases**

***Test Case 1: Saving a marker that was recently placed***

const fakeClient = {“Bob”: Age: 23, Gender: “M”, Situation: “Can’t work due to manic episodes”, Coordinates: [23.2333, 89.9998]};

saveMarker(“Bob”, 23, “M”, “Can’t work due to manic episodes”, [23.2333, 89.9998]);

Results = assert(clients, true);

***Expected:*** Pass

***Test Case 2: Retrieving a previous client***

const person = “Bob”;

loadClient(person);

Results = assertEquals(person.hasOwnProperty(“Bob”), true);

***Expected:*** Pass

**Handling Invalid Inputs**

* The system will be able to identify invalid inputs
* The invalid inputs would be rejected and an error message would be displayed
* Prevent invalid inputs from being processed.

***Messaging***

The messaging system is an integral part of the project, as it allows for some cross-organization communication. Thus we want to ensure that the sending and receiving of messages is working as expected.

User Interface

The user interface is one of the most fundamental aspects of the website as it allows the user to navigate across web pages in a seamless manner. The unit testing needed to be done will ensure that the specific functionalities are doing what they are supposed to and returning the correct components so the user gets where they need to go. The main goal with this is to test the logic behind the interface rather than the rendering itself. The three main components of the test will include

1. Input Validation
   * Testing whether the correct validation techniques are being used
   * Incorrect information is being known with error messages
   * Format checks and character limits.
2. Button Click Logic
   * The button should trigger the correct
3. State Changes
   * The internal aspects of the forms react when the user types something in (i.e username and messages

# **Integration Testing**

Integration testing is a type of software testing that involves highlighting how each component of a system interacts with each other and as a group. The main purpose of integration testing is to always be considering the aftermath of how internal interactions shape the execution of specific tasks. Additionally, ensuring that the expected behavior of a software happens is essential when guaranteeing the system is working properly and that the results are the desired results. It helps uncover bugs that were not addressed during unit testing, especially with complex systems. With integration testing ensuring the project’s overall reliability, it minimizes the error risks when testing how components work together. With the goal of integration testing in mind, there are five modules/interfaces to be tested:

* Login/Logout Auditing System: ensures authentication of users and roles
* Messaging System: allows different organizations to communicate with each other
* Database: In charge of overall CRUD operations when it comes to the general user, client, etc.
* Overall User Interface: engages with user and allows for ease of interaction
* Navigation: allows clients to be kept informed and in contact while still being processed for shelter availability

**Overall Integration Plan**

*Navigation*:

1. Verify Map Rendering
   1. Demonstrate that the map exists and is correctly displayed when navigation is clicked on.
   2. Simulate the act of zooming in and out and panning to confirm correct functionality.
2. Test Marker Placement
   1. Mock a new user creation and show that data is being retrieved correctly.
   2. Add markers to map at randomized locations in the city of Tucson.
   3. Verify that the marker appears at the correct location on the map.
3. Database Integration
   1. Simulate a marker being added to the database.
   2. Verify that the client’s input field data is properly stored in the database.
   3. Mock a user lookup and check that the existing data is retrieved and displayed correctly.

**Harnesses, Mocks, and/or Stubs**

* Test Harness
  + **Jest** will be used to automate front-end interactions and ensure that the navigation system works as expected.
* Mock Database
  + Mocked database calls would simulate database operations and storage without setting up a whole separate database for testing.
* Mock API Interactions (Node.js)
  + **Mocha** and the pairing library, **Supertest,** would mock the backend (Node.js) API responses used for CRUD operations

**How the correctness of data exchanges and function calls will be verified**

1. End-to-end flow
   1. Check to make sure that the expected behavior is happening and consult the database and UI, making sure that the behavior is being reflected.
2. Error handling
   1. Ensures that any issues that occur will be handled gracefully through guided troubleshooting for users.

# 

# **Usability/End-User Testing**

Usability testing is crucial for understanding how effectively a user can interact with a given software as a whole. This last step of the testing process involves the observation of users as they attempt to complete tasks on it. As the users try to complete the tasks, they are usually observed by researchers, with any challenges being noted. Testing the usability of a software can determine hidden pain points a user may face and provide opportunities to improve them. Testing sessions can either be moderated or unmoderated and in-person or remote. Additionally, conducting user testing in various environments, especially with our project, introduces altered criteria for different categories of users:

* **End users (general user)**
  + *Technical Expertise*
    - *Basic to Intermediate*
    - The end users will be interacting with the user interface to perform certain tasks concerning their designated role in their organization
      * Ex: uploading a document or video, messaging another user, joining an organization, etc.
    - The tasks given to them will be designed to get a better idea on how to improve the user experience and test our front end functionalities.
  + *Expectations*
    - These users will expect a simple, intuitive interface with clear instructions to complete the tasks. They are more likely familiar with common web applications, but not with complex technical systems.
* **Admin**
  + *Technical Expertise*
    - *Intermediate to Advanced*
    - The administrators will be responsible for managing roles, individual users, and permissions.
    - Each task that will be performed by them will ensure that our role-based access is up to par and that the admins are capable of navigating and executing their responsibilities.
  + *Expectations*
    - The admins will expect a easily navigable role management interface, allowing them to manage role assignments, and monitor organization activity.
* **Navigators/Shelter Personnel**
  + *Technical Expertise*
    - *Intermediate*
    - The navigators and shelter personnel will be able to use the system to collect client information, determine service availability, and coordinate placements that are accepted.
  + *Expectations*
    - Navigators and shelter personnel will expect a fast, intuitive interface for entering and retrieving information, and accurate mapping and search/filter abilities.

**Scope and Intensity of User Testing**

With careful consideration and planning, the scope and intensity of the user testing will demonstrate the entirety of our project’s usability coverage. Therefore, the following scope includes:

* Functionality testing
  + The functionality testing will focus on the core workflows of our web applications and confirm that the expected behavior occurs and is easy to navigate.
  + Core workflows include a user logging in and out, assigning roles, uploading multimedia, messaging other users.
* Navigation testing
  + Our navigation testing will ensure that the majority of users can find and use the features with minimal guidance.
  + Features include the dashboard, mapping interface, messaging interface, navigator input forms, and role/organization operations
* Error handling
  + Tests how well our project responds to user errors and will provide methods of helpful troubleshooting or recovery opportunities
  + Scenarios concerning the error handling will include invalid user inputs, missing information, and incorrect actions.
* Cross-platform usability testing
  + Demonstrates and ensures how smoothly the web application runs on various browsers and devices.
  + The Selenium Testing Framework will be the forefront of our cross-platform usability testing

**Testing Techniques**

*User Studies*

* The expected users will be given a list of tasks to complete by using the web application. Each of the tasks will represent one of the common workflows future users will perform.

*Survey/Questionnaire*

* At the end of each user study, the expected users will receive a survey or questionnaire that will allow them to reflect on their experience.
* The survey or questionnaire will ask for quantitative feedback like scaling and qualitative feedback such as open-ended questions

*Automated testing*

* Selenium testing library
  + In case in person usability testing is not possible or if the existing results need more concrete backing, Selenium will provided automated testing that will ensure that expected functionality is operational.

**Test Sessions**

* ***Number of test sessions***
  + 5-10 total
* ***What types of users will participate***
  + Participants will be chosen based on their roles and familiarity with similar systems. They most likely have careers/jobs in the target fields that we are developing for.
* ***How data will be recorded***
  + Observations of participants
  + Surveys/Questionnaires

**Usability Testing Timeline**

**(Would need to work with client on timeline)**

**How Results will be Analyzed**

* Quantitative Analysis
  + Error rates
  + Scale ratings
  + Task completion percentages
* Qualitative Analysis
  + Observational notes
  + User feedback

# 

# **Conclusion**

* Summarize overall testing plan
* Reinforce how this plan ensures a functional, reliable, and user-friendly product
* End with strong statement on how this plan will deliver high quality software