Cobb Connect Testing & Acceptance

Project Testing and Acceptance Plan

Granger Cobb Institute for Senior Living

Cobb Connect

Anthony Salvione,Gabriel Righi, Joshua Frey,

10/26/2023

TABLE OF CONTENTS

I. Introduction

I.1. Project Overview

Provide a brief summary of your software that is being tested. Outline all major functionality for which testing is crucial.

Since we are continuing a project instead of creating one from the ground up, we will not be testing components of software that were given to us. This is because the previous group should have already tested these components, so it would be unnecessary for us to retest them. To that end we will only be testing things that we have created, and parts of the code that we have changed. The main objectives we have for our project is to port it to an endpoint that is accessible and increase user functionality based on feedback.

Since we are only having to test what we add, we will essentially just be testing every feature. This is different than a traditional testing scheme where you would have to test large components of your website. Since we are only making small changes, we will only need to test our small changes, as the overall website has already been tested.

The major functionality that will need to be tested is as follows:

The user page is a new page we added that aggerages the post and information about a user. This page should only be accessible to the user for which it is designated and should properly gather all data about the user that is present on the website. We will need to test to make sure malicious users cannot access other users' websites, and to make sure that the data is gathered properly.

An admin account should be able to access all parts of the website, and should be able to query user data. We will need to ensure that an admin account is able to delete user posts and accounts, and also is able to properly gather information on all users on the site.

Geolocating is a large part of the website draw. The map that shows where users are is one of the key criteria for success. We will need to verify that with our new API user location is properly ascertained.

The most important part of our project that does not relate to the code itself is the project's portability. We will need to ensure that the docker container created for the project is able to spin up the website with no outside intervention.

I.2. Test Objectives and Schedule

Test Objectives:
Registration:

The previous functionality of a user registering to the website was as follows. The user makes an account with an email and a password. Then the user will be able to post and access web pages anonymously. We want to change that so a user must give account details before they can access any pages. This would take the form of the user being greeted with a message “you are not logged in” whenever they try to access a page without having registered first with full name info and such. We will test this using selenium web driver.

Location Data:

Currently the user enters location data at the profile page, and they are returned a latitude and longitude for their current location. We changed this system as the previous one was not working properly. We will need to test the functionality of this system with various locations to ensure the new API works properly.

Profile Page:

We are in the process of implementing a profile page. The profile page contains information about the user's profile and posts they have made. We will need to test to make sure that the users information is properly gathered and that all the posts they have made are properly displayed. We will test this using selenium.

Admin Page:

We added an admin page where an admin user is able to access information about the website. The admin has access to a special page that can only be viewed if they are the admin. We will need to verify that this page is inaccessible to other users, and that an admin is able to access it. We will also need to verify the information on the admin page using selenium.

Profile Page:

We will add post sorting, and the ability to like and add comments to post. We will need to test all the changes we make to the profile page, and these are the 3 that we currently have slated to create. We will test all of these using selenium webdriver, which will allow us to test the website programmatically.

Required Resources:

The only thing we need to test our website is the website itself. We can add dummy data to the database and unit test functions. We will need to use the package selenium webdriver to test much of the website's functionality. Selenium webdriver is a package that allows someone to programmatically interact with a website and can allow a tester to ensure repeatable tests.

Schedule:

Since we are only testing features that we implement, it would make sense to test things as we go, so we don’t have to do a lot of testing at the end. This means that our schedule for testing a component should line up with our schedule for deploying a component. That means that if we introduce a new feature that say sorts posts, we will also introduce testing with that feature to verify that it works. We should deliver our testing code in dart files that test functions of our program. We will also include python files that use selenium to test functionality.

I.3. Scope

This document outlines the comprehensive testing and acceptance plan for our project, focusing on the software components developed by our team. As previously mentioned, we will not retest components provided by the previous group, as their testing should have covered those areas. Our scope primarily includes the testing of new features and modifications, such as user registration, location data handling, profile pages, admin functionality, and user interactions. This plan aims to ensure the proper functionality, security, and usability of these specific elements, while also emphasizing the critical aspect of portability by confirming that the project can be seamlessly deployed using a Docker container.

II. Testing Strategy

II.1. We will be using continuous integration. As we create new functionality we will add it to a queue of things that need to be tested. Since we are not engaging in any overall system design, it will be fairly easy to test each new component we add.
II.2. Once we have added a new feature, we will identify the tests that verify the features functionality.
II.3. We will then determine how we expect the feature to work, and document that in a form.
II.4. Then we will verify the results of the test.
II.5. If the test result is different than our expected result, we will update our feature until the expected test case is met.
II.6. Once our feature has met the required test cases, we will document any changes we made to the feature
II.7. Then we will put the test up for a merge to main
II.8. Some group mate will review the test and the new feature and approve a merge to main

III. Test Plans

III.1. Unit Testing

Google Maps API Integration:

Objective of test: To verify that integration with Google Maps API for all location-related features are satisfied and correct.

User Authentication and Management:

Objective of test: To ensure that all user authentication functions operate as they should and that data is being correctly stored in Firebase.

Error Handling and Validation:

Objective of test: To ensure that this website handles errors correctly and validates correctly.

Edge Cases:

Objective of test: To address edge cases and scenarios that are exceptional

User Preferences:

Objective of test: To ensure that preference settings and all publicly displayed information is correctly saved and applied.

III.2. Integration Testing

Google Maps API Integration:

Component 1: Geocoding addresses

Test geocoding functionality to ensure that valid addresses are accurately converted to coordinates.

Verify that the function correctly handles invalid addresses and API failures.

Component 2: Map Display

Confirm that maps are displayed with the correct location markers and applicable features

Check that when the user interacts with the map, all zooming and navigation function as expected.

User Authentication and Management:

Component 1: User Registration

Test that user registration correctly saves user information into the database

Check to see that all passwords are secure when hashed

Confirm that error handling during registration (like creating an account with the same email) works as expected.

Component 2: User Login

Confirm that the user login authenticates all valid users and rejects all invalid users

Test login tokens to see if they are secure

Component 3: User Profile Updating

Test to see if users can correctly and accurately update their information

Check to see if information changes are correctly replaced in the Firebase database

Error Handling and Validation:

Component 1: Input Validation

Test all input validations for user registration and updating

Check to ensure that invalid inputs are always rejected and the appropriate error message is displayed.

Component 2: Error Handling

Check to ensure that all error handling implementations are in place for unexpected bugs and issues.

Edge Cases:

Component 1: Boundary Conditions

Test the extreme scenarios (maximum and minimum length inputs, etc).

Check to ensure that the website can handle these scenarios

Component 2: Concurrency

Test simultaneous user actions to ensure data integrity

User Preferences:

Component 1: Preferences

Test that users can customize their desired preferences, such as notifications or post filtering

Ensure that preferences are correctly saved and applied

III.3. System Testing

III.3.1. Functional testing:

Our team's plan for functional testing will revolve around testing the functional requirements stated in the Requirements and Specifications document. Not all the functionalities are listed in the Requirements and Specifications document because our team is currently hosting a beta test for beta testers to receive feedback on what new functionalities to add, get rid of, or change. Each new functional feature our team implements will be tested by us and all the use cases stated in the use case diagram in the Requirements and Specifications document will be tested to verify if they perform how they are expected to. Each team member has documented in the Github issues section their acceptance criteria for each functional feature that is being implemented. If all the acceptance criteria have been met then the feature is functioning correctly and if it doesn't pass the acceptance criteria then the developer working on that feature will solve the error.

III.3.2. Performance testing:

Our team's plan for performance testing will revolve around testing the non-functional requirements of the Requirements and Specification document. The non-functional requirements are based on performance, scalability, robustness, and responsiveness. It encompasses load testing to evaluate the site's response under typical user loads and stress testing to identify its limits. Testing the robustness will ensure that the website remains stable under various usage scenarios. Scalability testing will allow our team to see the website's ability to accommodate user growth, with automatic storage scaling. Response time and resource utilization testing focus on responsiveness and resource efficiency. A successful performance testing process will ensure the website meets its performance objectives, providing a smooth and reliable user experience, while also validating its adherence to the scalability and robustness requirements. Any non-functional requirements not met will be looked at by our team and will likely require our team to change our code and algorithms to be more efficient.

III.3.3. User Acceptance Testing:

Client prefers project requirements are acceptance criteria. We validate our project requirements using test plans and test cases documented in previous sections.

IV. Environment Requirements

The unit testing can be performed using dart’s testing package that is already provided when creating a flutter project. The integration testing and functional testing can be performed using an open-source framework and tool for automating web browsers with Selenium Webdriver and outside blackbox testing. There will be no specific hardware requirements to test our project.

V. Glossary

Selenium WebDriver: A tool for automating web browsers, commonly used for web application testing.

Firebase: A platform for developing web and mobile applications, often used for real-time database storage.

Beta Testing: Testing performed by a select group of users before a full release to gather feedback.

Load Testing: Testing the system's response under typical user loads.

Stress Testing: Testing the system's behavior under extreme conditions beyond its specifications.

Scalability Testing: Evaluating the system's ability to accommodate user growth.

Robustness Testing: Ensuring the system remains stable under various usage scenarios.

Responsiveness Testing: Testing the system's response time and resource utilization.

Acceptance Test: A test performed by end-users to compare the system to initial requirements.

Beta Testers: Users who participate in beta testing to provide feedback on the software.