Group 5 Software Test Plan for CityMeet Site

By Elliot Gong, Alan Mai, Alejandro Vargas, Vivian Casas, Darshan Patel, Isidoro Flores, Deion Stapleton, Siying Chen

1. Introduction

1.1 Purpose

This document outlines the testing guidelines and methodologies to ensure proper functionality, performance, and integration of CityMeet, a web-based platform for community-based media and support.

1.2 Scope

Testing will be conducted on completed features of CityMeet, which include user accounts, content posting, geospatial mapping, and seamless integration with public services/resources provided by the Los Angeles city government.

1.3 References

Google Maps API Documentation
Supabase Documentation
Vercel Documentation
Mapbox API Documentation
Instagram API Documentation

2. Test Strategy

2.1 Objective

Identify and correct any errors in the CityMeet website to promote a smooth and error-free user experience.

2.2 Approach

Both manual and automated tests will be utilized to validate individual features, grouped components, the system as a whole, integration, and the website's performance under heavy conditions.

2.3 Tools

We will use GitHub to log tests, record results, and manage tasks.

2.4 Environments

Different areas for development, testing, staging, and production will be created. This way, each phase can be tested separately and thoroughly.

2.5 Entry and Exit Criteria

Entry:

- Test environment set up with all necessary configuration
- Test data is available

Exit:

- All test cases executed
- All critical defects are closed
- Test summary report reviewed and approved

3. Scope of Testing

3.1 Functional Testing

3.1.1 Unit testing:

Developers will test individual methods to ensure that the expected input results in the intended output. Individual methods will be tested with incorrect inputs as well to ensure proper error handling.

3.1.2 Component Testing

Developers will test groups of units in unison to ensure proper method interaction and dependencies. As well as proper interface interactions.

3.1.3 System Testing:

All components are tested together to ensure the system functions as a whole with proper input handling.

3.1.4 Regression Testing:

Developers will test whether new code components interact with the system as intended or if they change how older methods function.

3.2 Non-Functional Testing

3.2.1 Performance and Load

We will stress test the performance of our web application by uploading a high number of photos to the user account. The purpose of this test is to make sure performance remains the same.

3.2.2 Alpha testing

We will use the application in our day-to-day life to ensure it is working fine with no bugs or glitches in our application.

3.2.3 Beta testing

We will release our web application to a handful of people to simulate real-world experience, and this will help ensure our web application functions as intended with little to no bugs.

4. Consideration of Infrastructure

4.1 Server Configuration

1. Our database is stored on Supabase. We are confident in their ability to protect our data.

4.2 Database

1. Supabase is where we will host our database(s).
    1.1. We have chosen to use
    1.2. Our database will be on the Supabase cloud because it can handle scaling, security, and backups. They have a good authentication feature too.
        1.2.1. If we wish to scale upward, Supabase is capable of doing that

5. Risks or Mitigation Plan

5.1 Risks

- Incomplete features
- Compatibility issues between different devices or web browsers
- Unsatisfactory user experience
- System crashes/ compromised system performance
- Security vulnerabilities such as data loss, hacking, and malicious code injection

5.2 Mitigation

- Cross Browser Testing + Cross Device Testing
- User testing to identify potential issues with design and interface
- Performance testing to improve speed/responsiveness/stability under strenuous circumstances
- Access Control Testing to prevent unauthorized users from having unauthorized access to the database.

6. Resourcing

6.1 Team Composition

- 1 test manager
- 2 Database Test Engineers
- 2 Test Engineers for functional testing
- 2 Front-end Test engineers

7. Milestones and Deliverables

7.1 Milestones

Sprint 02.1.0 - 02.1.3

- Setting up a test environment for profile creations.
- Writing and defining test cases.

Sprint 02.1.4- 02.1.6

- Setting up testing for post-creation.
- Implementing group page creation.
- Security and performance testing
- User testing
- Refactor if needed

Sprint 02.2.0 - 02.2.6

- User testing
- Refactoring if needed
- Additional security and performance testing
- Final report

7.2 Deliverables

- Source code
- Deployment instructions
- Documentation and Release Notes
- Release notes