CS3337 Software Test Plan Group 6

1. Introduction
   1.1 Purpose:
   - This document details the software testing strategies employed for EurekaEats, a web-based platform that assists users in discovering local restaurant recommendations. The goal is to ensure the robustness of functionality, seamless integration, and optimal platform performance.

   1.2 Scope
   - The testing encompasses all integral functionalities of EurekaEats, including database reliability through MongoDB, integration with external APIs such as Google Maps and Google Places, and CRUD operations with the database.

   1.3 References
   [1] EurekaEats Requirements Document
   [2] Google Maps Platform API Documentation
   - Google Places API Documentation (included in Maps Platform)

   [3] Yelp Fusion API Documentation
2. Test Strategy
   2.1 Objective
   - To detect and address flaws or inconsistencies within EurekaEats, aiming to achieve a user-friendly and seamless experience without errors.

   2.2 Approach
   - We will employ manual and automated testing approaches to verify individual components, assess integrations, evaluate the end-to-end system, and gauge the website's performance under various load conditions.

   2.3 Tools
   - Server-Side Python Functions Testing: PyTest
   - FrontEnd JavaScript Testing: Jest
   - Automated Tests: Selenium
   - Quality Code Testing and Analysis: Sonarqube

   2.4 Environment
   - Development, Testing, staging, and production environments will be set up to ensure isolated and focused testing phases.

   2.5 Entry and Exit Criteria
   Entry:
   - Test environment set up with all necessary configurations
   - Test data is available

   Exit:
   - Test cases are executed
   - Critical defects are closed
   - Test summary reviewed and approved
3. Scope of Testing
   3.1 Functional Testing
   3.1.1 Unit Testing

- During each sprint, developers continually conduct unit tests on individual application components in isolation, allowing us to confirm proper functionality.

3.1.2 Integrated Unit Testing

- Towards the middle to end of each planning increment, developers collaborate to validate how integrated components interact and interface with one another, ensuring their proper coordination. This process occurs as each component from a story is integrated with others, following individual unit tests.

3.1.3 System Integration Testing

- Upon nearing the end of a sprint, the development team verifies that the entire system, composed of multiple components, functions correctly and aligns with the specified requirements.

3.1.4 User Acceptance Testing (UAT)

- Before a release, the Product Owner selects a group of local users who want to eat good to perform User Acceptance Testing (UAT). This allows developers to establish that our web application meets the standard of the average user.

3.1.5 Regression Testing

- Testers ensure that recent changes, such as feature additions or modifications, have not adversely impacted existing features. The regression test suite is executed as part of an automated Continuous Integration/Continuous Delivery (CI/CD) pipeline.

3.2 Non-Functional Testing

3.2.1 Security Testing

- Using SonarQube, our developers can identify and fix security issues on our web application.

3.2.2 Usability Testing

- The UI design will be compared to the UX standards of our group and confirm that the user will have a seamless experience

3.2.3 Performance and Load Test

- During each major release of a version of the web application, developers will stress and performance test the application. This will include verifying that the system responds to a user's action within 2 seconds and can handle many users on the app.

4. Consideration of Infrastructure

4.1 Server Configuration

- Tests will be run on a test web-hosting environment, specifications of the web-hosting environment are unknown for now.

4.2 Database

- We will conduct tests on a dedicated test database using MongoDB, which has regular backup procedures and can perform rollbacks when necessary.

5. Risks or Mitigation Plan

5.1 Risks
- Integration discrepancies
- Data loss and security vulnerabilities
- Delays
- Third-party dependencies
- Scalability

5.2 Mitigation Plan
- Data validation checks
- Regular database backups
- Third-party dependency update + depreciation monitoring
- Proper upkeep of web application documentation

6. Resourcing

6.1 Team Composition
- 1 Test Manager
- 3 Test Engineers
- 1 Database Test Engineers

7. Milestones and Deliverables

7.1 Milestone

PI 23.1(current):
- Test Environment Setup
- Unit/Integrated Unit testing
- System Integration Testing completion
- Coding standard testing and usability testing
- Testing for developed features
- Recurring security/performance baseline testing

PI 23.2 (forecasted):
- User acceptance testing and fixing
- Performance and load testing and fixing
- Regression/smoke testing
- Recurring security/performance testing
- Final report

7.2 Deliverables
- Bug reports
- Test completion report
- MongoDB database integrity/consistency report