Test Plan

Project Name: Testing WEare Social Network
Created By: Hristo Hristov
Edited By: Mina Mladenova and Vladislav Adamov
Date: 27.10.2024

Table Of Contents

1. Introduction

This test plan outlines the QA strategy for testing the WEare Social Network Web Application. The application allows users to connect with people, create, comment on, and like posts, and get a feed of the latest or most relevant posts from their connections.

The purpose of this document is to define the strategy, approach, roles and responsibilities of the parties involved in the test process of the project. It describes what will be tested, by whom and their responsibilities.

This document covers the functional, performance, usability, and security aspects of the application.

2. Scope of Testing

In-Scope:

- Functional testing of user registration, login, and profile management.
- Testing of post creation, comment, and like functionalities.
- Verification of the personalized and public feed.
- Testing of connection requests and approvals.
- Admin functionalities, including profile and content moderation.
- REST API testing for CRUD operations on users and posts.

Out-of-Scope:

- Testing integrations with third-party identity verification services (unless specified).
- Testing beyond relational database requirements, like alternative database support.

3. Objectives

- Validate that the system meets all functional requirements and offers reliable user experience.
- Identify and resolve bugs before release to ensure high-quality delivery.
- Ensure that security requirements, especially around user authentication and data privacy, are met.
- Verify REST API operations and ensure they are documented and follow REST design best practices.

---

4. Test Base

Testing will be executed based on written specifications. These are stored and shared in electronic format according to the listed below.

- Jira stories
- WEare app API documentation
- Social Network Test Requirements
- Social Network Project Specification

---

5. Test Strategy

During each sprint it will be decided which test cases will be automated having in mind possibilities and efficiency. Test automation will be done on test cases that successfully passed the manual tests and have no blocking or critical issues open.

5.1. Functional Testing

- Test **Registration and Login**: Validate username, password, and email constraints, email verification, and login/logout functionalities.
- Test **Profile Management**: Verify that users can update profile info, manage privacy settings, and view other profiles based on privacy controls.
- Test **Posting and Interaction**: Test the ability to create, edit, delete, and set visibility for posts; validate commenting and liking functionalities.
- Test **Accessibility and Search for Anonymous Users:** Verify landing page view, search users and that only public information is visible
- Test **Connection Requests**: Verify sending, approving, and removing connections.
- Test **Admin Controls**: Ensure admin users can edit/delete profiles, posts, and comments.

5.2. API Testing

- CRUD operations on user profiles, posts, and comments.
- Validate API endpoints for managing connections, creating and commenting on posts, and retrieving personalized/public feeds.

- Test API responses, status codes, and error handling.

5.3. Security Testing

- Test secure handling of passwords.
- Verify that access controls are correctly enforced across public and private content.
- Check that private data cannot be accessed without authentication.

5.4. Compatibility Testing

- Test the application across popular web browsers (e.g., Chrome, Firefox, Safari) and devices (desktop, mobile).

---

6. Test Environment

- **Hardware**: Laptop
- **OS**: Windows 10, MacOS
- **Frontend**: Web browser (Chrome version 131.0.6778.70, Firefox version 132.0.2, Safari version 18.1.1)
- **Backend**: REST API, relational database
- **Admin Console**: Accessible only with admin privileges
- **Database**: Scripted test database with sample data
- **Local Deployment Environment:** Docker 4.35.1

---

7. Test tools

Following tools and utilities will be used during testing:

- **Jira** – Project and defect tracking tool
- **X-ray for Jira** – Test case management
- **Google Chrome, Mozilla Firefox, Apple Safari** emulators to test the responsive designs
- **Selenium WebDriver** – development of automated test scripts
- **Postman**
- **REST Assured**
- **Docker Desktop**

---

8. Test Data Requirements

- Sample user accounts (regular and admin).
- Test data for posts, comments, and likes.
- Sample connection requests and approvals.
- API-specific data for CRUD and feed generation tests.

9. Entry Criteria

- The WEare software is deployed and ready for testing.
- Test environment is set up and validated.
- Test data is prepared and accessible.

10. Exit Criteria

- All high priority test cases are executed with a pass status.
- All security vulnerabilities identified have been addressed.
- No blocking problems persist

11. Schedule

The testing should be performed in the period 28th of October - 25th of November 2024. It will be divided on several Sprints as follows:

- The first week (28 Oct.- 4 Nov. 2024) - planning, setting up the process and the tools, creating a Test Plan, developing a High-level test cases and starting the manual testing.
- The second week (4 Nov. - 11 Nov. 2024) - performing manual testing, bug reporting in Jira.
- The third week (11-18 Nov. 2024) - starting the automation testing.
- The fourth week (18-25 Nov. 2024) - finishing the automation testing and reporting of the test results, last-minute fixes and preparation of the presentation itself.
  During this time the team will gather on regular meetings and discuss the progress on the project.

12. Test Execution

12.1. Test Case Development

- Detailed test cases for functional requirements, security checks, and API responses.
- Identify high-priority test cases covering user-critical features like login, profile management, and posting.

12.2. Bug Tracking and Reporting

- Use a project management tool (e.g., JIRA, GitLab Issues) to log, track, and manage bugs.
- Regular reporting on test progress and defects found.

- Defect Life Cycle:

- **New**: When any new defect is identified by the tester, it falls in the 'New' state. It is the first state of the Bug Life Cycle. The tester provides a proper Defect document to the Development team so that the development team can refer to Defect Document and can fix the bug accordingly.
- **Assigned**: Defects that are in the status of 'New' will be approved and that newly identified defect is assigned to the development team for working on the defect and to resolve that. When the defect is assigned to the developer team the status of the bug changes to the 'Assigned' state.
- **Open**: In this 'Open' state the defect is being addressed by the developer team and the developer team works on the defect for fixing the bug. Based on some specific reason if the developer team feels that the defect is not appropriate then it is transferred to either the 'Rejected' or 'Deferred' state.
- **Fixed**: After necessary changes of codes or after fixing identified bug developer team marks the state as 'Fixed'.
- **Pending Request**: During the fixing of the defect is completed, the developer team passes the new code to the testing team for retesting. And the code/application is pending for retesting on the Tester side so the status is assigned as 'Pending Retest'.
- **Retest**: At this stage, the tester starts work of retesting the defect to check whether the defect is fixed by the developer or not, and the status is marked as 'Retesting'.
- **Reopen**: After 'Retesting' if the tester team found that the bug continues like previously even after the developer team has fixed the bug, then the status of the bug is again changed to 'Reopened'. Once again bug goes to the 'Open' state and goes through the life cycle again. This means it goes for Re-fixing by the developer team.
- **Verified**: The tester re-tests the bug after it got fixed by the developer team and if the tester does not find any kind of defect/bug then the bug is fixed and the status assigned is 'Verified'.
- **Closed**: It is the final state of the Defect Cycle, after fixing the defect by the developer team when testing found that the bug has been resolved and it does not persist then they mark the defect as a 'Closed' state. Describe the life cycle from detection to closure.

- Severity Levels:
    - Blocking: This severity level implies that the process has been completely shut off and no further action can be taken.
    - High: This is a significant flaw that causes the system to fail. However, certain parts of the system remain functional.
    - Medium: This flaw results in unfavorable behavior but the system remains functioning.
    - Low: This type of flaw won't cause any major breakdown in the system.
- Priority Levels:
    - Low: The defect is irritant but a repair can be done once the more serious defects can be fixed.
    - Medium: The defect should be resolved during the normal course of the development but it can wait until a new version is created.
    - High: The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed.

12.3. Test Automation

- Automate test cases for REST API endpoints using tools like Postman or Rest Assured.

- Automate repetitive UI and functional tests using Selenium WebDriver.

---

13. Test Metrics and Reporting

- **Test Case Execution Status**: Number of test cases passed, failed, and blocked.
- **Defect Density**: Number of defects per module or feature.
- **Test Coverage**: Percentage of application code and functionality covered by test cases.
- **Performance Benchmarks**: Metrics for response times and load-handling capacity.

---

14. Risks and Mitigations

Risk	Mitigation
Scope Creep	Define and freeze feature requirements before beginning test case development.
Insufficient Test Data	Generate representative test data for all scenarios before test execution.
Delays in Test Automation	Prioritize manual testing for core features while automation is in progress.
Security Vulnerabilities	Conduct regular security reviews and tests to identify vulnerabilities early.

---

15. Test Deliverables

- Test plan
- Test report
- Automated tests code/repository link
- Script for execution of automated tests
- A document with the required prerequisites to run the tests
- A document with a concise description of how to run the tests, filter them, etc.
- High-level test cases

---

16. Our Team

- Vladislav Adamov, vladislav.adamov.a61@learn.telerikacademy.com – leading the Selenium automation, engaged in manual testing, creating reports.
- Mina Mladenova, mina.mladenova.a61@learn.telerikacademy.com – keeps the Test Plan updated, task planning, organizing the workflow in Jira, creating test and bug templates, manual testing and logging bugs, API testing (Postman), compatibility testing, engaged in Selenium automation.

- Hristo Hristov, hristo.hristov.a61@learn.telerikacademy.com – created the Test Plan, engaged in task planning, setup of the projects in Jira (integration with Xray) and GitHub, engaged in organizing the Jira workflow, engaged in manual testing, creating Selenium tests.