**LOGBLOCK SOCIAL MEDIA SERVICE**

Iteration Test Plan

Version 1.1

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 16/12/2024 | 1.0 | Documentation setup and initial version | Ngũ Kiệt Hùng |
| 26/12/2024 | 1.1 | Documentation finalization | Trần Thanh Long |

Table of Contents

1. Introduction 4

1.1 Purpose 4

2. Target Test Items 4

3. Environmental Needs 4

3.1 Hardware Requirements 4

3.2 Software in the Test Environment 4

3.3 Productivity and Support Tools 4

4. Responsibilities, Staffing, and Training Needs 4

4.1 People and Roles 4

**<Iteration/ Master> Test Plan**

# Introduction

## Purpose

This Test Plan Document describes the strategies, procedures, and resources required to guarantee the functionality and non-functionality of the e-commerce website selling shoes through the means of testing. This document acts as a testing process road map, making sure that every main component of the website is checked over and verified in-depth before being submitted for validation.

## Scope

This test plan covers mostly Business logic related services, as well as the integration components to bridge User Interface and Business logic. The testing procedure will be conducted internally and not publicly released as a beta testing procedure. The testing phase will produce a minimum viable product to allow basic functionality verification and validation.

## Test Approach

The testing process will involve both manual and automated testing methods. Functional tests will mostly be the main approach in ensuring correctness of data state. Data consistency will also be considered on client side Single Page Application while ensuring non-functional requirements such as Security and Performance.

## Assumptions and Constraints

Assumptions**:**

* The development team will provide a stable build for testing.
* All necessary testing environments, tools, and resources will be available.
* The project requirements will remain consistent during the testing period.

Constraints:

* The testing process may be limited by time constraints, requiring prioritization of critical test cases of Business Logic handling and data access authorization.
* Due to the incompleteness of the current system state, no extensive cross-site session consistency test will be performed.

# Target Test Items

* **Functional Requirements:**
  + User authentication
  + Front-end components
  + Back-end components
  + Controllers components
  + Feeds generation
* **Non-functional Requirements:**
  + Compatibility testing
  + Security testing
* Areas:
  + Profile Service Testing:
    - **Testing Account Creation and Profile Updates**: The tests check the ability to create accounts and update user profile information such as display name, bio description, profile image, and privilege level.
    - **Verifying Data Accuracy in the Database**: After profile information is updated, tests confirm that the data in the database matches the entered values.
    - **Testing User Lookup by Email**: The tests also check the ability to retrieve user profiles from the database using the email, ensuring that the correct data is returned when searched.
  + Connection Service Testing:
    - **Testing Connection Creation**: The tests confirm the ability to establish a connection between two users through the connection DTOs. After a connection is created, the tests ensure that the connection is saved in the database.
    - **Testing Connection Deletion**: The tests validate the functionality of deleting a connection, confirming that the connection is removed from the database after the deletion request.
  + Report Service Testing:
    - **Testing Report Creation and Storage**: The tests check the ability to create a report from a user and ensure that the report is stored in the database.
    - **Testing Report Deletion**: The tests confirm the ability to delete a report and check that the report is deleted from the database.
  + Post Service Testing:
    - **Testing Post Creation and Updates**: The tests verify the ability to create new posts for users and check that the post data is accurately stored in the database.
    - **Testing Post Retrieval and Updates**: The tests also check the ability to retrieve posts from the database and ensure that the post content, author, and other attributes are correct.
  + ExpertSuggestedSolution Service Testing:
    - **Testing ESS Creation**: The testCreateESS method tests whether a suggested solution can be created successfully. It mocks the addESS method in the repository and verifies that the correct ESS ID is returned by the service.
    - **Testing ESS Retrieval**: The testGetESS method tests the retrieval of an expert suggested solution by ID. It mocks the findById method in the repository and checks if the correct ESS is returned.
    - **Testing ESS Update**: The testUpdateESS method verifies that an expert suggested solution can be updated. It mocks
  + Comment Service Testing:
    - **Comment Creation**: Verifies that a comment is successfully created and stored in the database, ensuring the returned comment ID is correct.
    - **Comment Update**: Tests the ability to update an existing comment, ensuring the updated comment is correctly stored and retrieved.
    - **Comment Retrieval**: Validates the correct retrieval of a comment by its ID, confirming that the fetched data matches the original.
    - **Comment Deletion**: Ensures that a comment can be successfully deleted, verifying the deletion result.
    - **Updating Non-Existing Comment**: Checks the behavior when trying to update a non-existing comment, ensuring it returns an error or an invalid result.
    - **Deleting Non-Existing Comment**: Verifies the system's response when attempting to delete a non-existing comment, ensuring it returns a failure.
  + Newsfeed Service Testing:
    - **Valid Connections**: Ensures the feed is correctly generated when there are valid connections, verifying that posts from connected users are included in the result.
    - **No Connections**: Tests the behavior when the user has no connections, ensuring the feed is empty.
    - **Empty Posts**: Verifies that the feed remains empty if connected users have no posts.
    - **Null Connections**: Handles the edge case where the connection data is null, ensuring the feed generation gracefully returns an empty list.
  + Exploration Feed Service Testing:
    - **Trending Posts Available**: Validates the feed generation with trending posts, ensuring the feed contains correct posts with proper captions.
    - **No Trending Posts**: Ensures the feed returns an empty list when there are no trending posts in the repository.
  + AdminService Testing Areas:
    - **Retrieve All Reports**
      * **Valid Reports**: Ensures that all reports are retrieved successfully from the repository and mapped correctly to the response.
      * **No Reports**: Verifies that the service returns an empty list when there are no reports in the repository.
    - **Remove Post**
      * **Successful Removal**: Ensures that a post is successfully removed when it exists in the repository.
      * **Post Not Found**: Tests the behavior when attempting to remove a post that does not exist, ensuring proper handling of the scenario.
    - **Close Account**
      * **Successful Account Closure**: Verifies that the account is successfully deleted when it exists in the system.
      * **Account Not Found**: Ensures that the service handles the case where the account to be deleted does not exist gracefully.
* Techniques:
  + **Unit Testing with Mockito**: **Using Mockito to Mock Dependencies**: In many tests, Mockito is used to mock repositories or dependent classes such as ProfileRepository, ConnectionRepository, ReportRepository. This helps isolate the testing logic from external dependencies like the database.
  + **Integration Testing**: **Combining Unit and Integration Testing**: Integration testing techniques are used when it is necessary to ensure that different system components (e.g., ProfileService, ConnectionService) work correctly when integrated with each other and with the actual database.
  + **Transactional Testing**: **Using @Transactional to Ensure Database Integrity**: You have used @Transactional in the tests, meaning each test runs within a transaction and will be rolled back after the test is finished. This helps protect the database from unwanted changes during testing.
  + **Assertions**: **Using Assertion Methods to Verify Results**: You use assertion methods like assertEquals(), assertNotNull(), assertTrue(), assertFalse() to check the accuracy of results in each test case. These assertions ensure that the values returned by the system are what you expect.
  + **Data-Driven Testing**: **Using Test Data from Lists**: In tests like testScenarioInsertionStandard(), you create a list of test cases with different input sets and perform testing on each data set. This helps test with multiple data sets without writing repetitive code.
  + **Test Profiles**: **Using @ActiveProfiles("test") to Ensure Tests Run in a Separate Test Configuration Environment**: This helps separate the test environment from production, preventing actual data from being changed during testing.

# Environmental Needs

## Hardware Requirements

* **RAM:** 8GB or higher.
* **CPU:** Intel Core i5 or higher.
* **GPU:** Not required.
* **Screen Resolution:** 1920x1080 or higher.

## Software in the Test Environment

The following base software elements are required in the test environment for this *Test Plan*.

| **Software Name** | **Purpose** | **Version** | **Type and Other Notes** |
| --- | --- | --- | --- |
| Firefox | Browser Compatibility Testing | 133.0.3 | Internet Browser |
| Google Chrome | Browser Compatibility Testing | 131.0.6778.204 | Internet Browser |
| Windows 11, 64 bit | OS Compatibility Testing | 24H2 | Operating System |
| Arch Linux x86-64 | OS Compatibility Testing | 6.12.4 | Operating System |
| Debian Linux x86-64 | OS Compatibility Testing | Bookworm 12 | Operating System |
| Postman | API Testing | 11.23.3 | Testing Software |
| Docker | Create environment to run the software | 27.3.1 | Software Environment |
| PostgreSQL | Database server for the website | 16.3 | Database Server |

## Productivity and Support Tools

The following tools will be employed to support the test process for this *Test Plan*.

| **Tool Category or Type** | **Tool Brand Name** | **Vendor or In-house** | **Version** |
| --- | --- | --- | --- |
| Defect Tracking | MS Excel 2010 | MS | 2010 |
| Test Cases Description | MS Excel 2010 | MS | 2010 |
| JUnit | xUnit | xUnit | 5.10.0 |

# Responsibilities, Staffing, and Training Needs

## People and Roles

This table shows the staffing assumptions for the test effort.

| **Human Resources** | | |
| --- | --- | --- |
| **Role** | **Minimum Resources Recommended**  **(number of full-time roles allocated)** | **Specific Responsibilities or Comments** |
| Test Manager | Ngũ Kiệt Hùng | Provides management oversight.  Responsibilities include:   * planning and logistics * agree mission * identify motivators * acquire appropriate resources * present management reporting * advocate the interests of test * evaluate effectiveness of test effort |
| Tester | Ngũ Kiệt Hùng  Trần Thanh Long  Trần Nguyễn Nhật Cường Nguyễn Thế Thanh Long | Implements and executes the tests.  Responsibilities include:   * implement tests and test suites * execute test suites * log results * analyze and recover from test failures * document incidents |
| Test System Administrator | Ngũ Kiệt Hùng | Ensures testing environment and assets are managed and maintained.  Responsibilities include:   * administer test management system * install and support access to, and recovery of, test environment configurations and test labs |