# TEST REPORT

**Testing Social Network**
**TEAM 1**
**Ensure high quality of the platform**

**Dipl. Eng. Teodora Hristova**
**Dipl. Eng. Ivailo Tabakov**
**Dipl. Eng. Ivan Petkov**

| Document Location | | | Category | Audience |
|---|---|---|---|---|
| Team1 OneDrive | | | Final Project | **Telerik Academy Alpha Trainers** |
| **Date 19.10** | **Author TH, IT & IP** | **Version I.1** | | |

# I. Introduction

This document explains the various activities performed as part of the Testing of the 'Weare Social network' application. This report is intended to provide a clear and structured overview of the testing process and its results, with annotation on any significant findings.

It presents detailed account of the testing encompassing different aspects of the tested application.

It outlines the strategies, methodologies and used tools for testing. Along with that it outlines the types of tests that were executed.

# II. Application overview

The WEare Social Network application enables you to:

- Connect with people.
- Create, comment and like posts.
- Get a feed of the newest/most relevant posts of your connections.

# III. Scope

## 1. In Scope

Functional Testing for the following modules:

- Registration
- Sign in & sign out
- Profile information
- Profile search
- Connect to / disconnect form others
- Post actions
- Comments actions
- Admin actions

API Testing for the following operations:

- Users
- Users - CRUD Operations
- Add/approve friend
- Get news feed
- Posts - CRUD operations
- Comment posts
- Like/unlike posts/comments

## 2. Out of Scope

- Profile search by profession
- Profile information – skills to offer
- Filter posts by skill category
- Edit own posts as a user
- Delete own posts as a user
- Edit own comments as a user
- Delete own comments as a user
- Friends list
- Users list

### 3. Items not tested

## IV. Metrics

Please, refer to this link for further details WEare Web Application Testing Dashboard in Jira.

All bugs reported can be found here: Bug tracking board

## V. Results of Performed Testing Types

### 1. Exploratory testing

The testing process began with a freestyle exploratory phase that aimed to provide a solid foundation for the subsequent testing activities. During this initial stage, our team to familiarizes themselves with the application and ensures its readiness for in-depth evaluation.

First and foremost, a preliminary smoke test was conducted to verify the application's basic functionality and confirm that it was operational. This step ensured that the app was up and running, setting the stage for further testing.

Following the smoke test, our team proceeded to explore the main features of the application. This exploratory phase was instrumental in acquainting testers with the intricacies of the system. It allowed them to interact with the app and gain a comprehensive understanding of how it worked.

By engaging in this exploration, our testing team began to identify potential areas of interest for future functional testing. This phase laid the groundwork for a structured and strategic testing plan, enabling us to prioritize and focus on the most critical aspects of the application during subsequent testing phases.

In summary, the freestyle exploratory testing served as a crucial initial step in our testing process. It ensured that the application was running and provided valuable insights that informed our testing strategy moving forward.

### 2. Functional manual testing

The project's initial requirements, as outlined in the project scope and specifications, provided a clear set of features and functionalities that the application was expected to deliver. Functional testing was carried out to ensure that these requirements were met.

All functionalities within the scope of testing has been covered.

**Number of test cases executed: 94**

- PASS: 70
- FAIL: 24

All bugs found during this stage of testing are accessible here.

Number of issues found: 25

By priority:

- High: 12
- Medium: 8

- Low: 5

Most affected feature: User registration

## 3. Missing Features from exploratory and functional manual testing

**Profile search by email**

Expected Behaviour: Users should be able to find others by email

Actual Behaviour: Currently, there is no functionality for users to search for profiles using email addresses.

Impact: The absence of the profile search by email feature may hinder users from efficiently connecting with others or verifying the identity of potential contacts, potentially limiting the platform's usability and user engagement. This feature's absence could also reduce the platform's effectiveness in fostering user connections and networking.

**Reset password**

Expected Behaviour: Users should be able to change their passwords

Actual Behaviour: Users do not have the ability to reset or change their passwords. In case of forgotten or compromised passwords, there is no self-service password reset functionality available.

Impact: Without the ability to reset passwords, users may face difficulties in regaining access to their accounts in case of forgotten or compromised passwords, potentially leading to frustration and a higher volume of support requests.

**Email verification (confirm email)**

Expected Behaviour: Users expect to receive a confirmation email after signing up, which they must click to verify their email address and activate their account.

Actual Behaviour: Users do not receive a confirmation email, and their email addresses remain unverified.

Impact: The absence of email verification could lead to potential issues with account security. Unverified email addresses may result in a higher risk of fraudulent accounts and reduced user trust. Additionally, it may lead to users not being able to receive important notifications and updates.

**Identity verification:**

Expected Behaviour: Users should have the option to complete an identity verification process to confirm their identity, which may include submitting official documents or other identifying information.

Actual Behaviour: There is no identity verification process in place, and users can access the platform without confirming their identity.

Impact: The lack of identity verification can increase the risk of impersonation, fraudulent accounts, and misuse of the platform for malicious purposes. It may also affect the platform's ability to provide a safe and trustworthy environment for its users.

**Delete user by admin:**

Expected Behaviour: Administrators should have the capability to delete user accounts when necessary, ensuring the ability to manage and maintain the user base.
Actual Behaviour: Administrators do not have the functionality to delete user accounts.
Impact: The absence of the ability to delete user accounts by administrators may lead to challenges in managing and maintaining the platform's user base, potentially allowing for the persistence of problematic or malicious accounts. This can affect the overall security and integrity of the platform.

## 4. REST API Testing

REST API Testing is open-source web automation testing technique that is used for testing RESTful APIs for web applications. The purpose of REST API testing is to record the response of REST API by sending various HTTP/S requests to check if REST API is working fine or not. REST API testing is used for Integration testing and Functional testing
“WEare Social Network application” REST API testing is developed with REST Assured library and Postman testing tool for executing HTTP requests and test API responses. Methods used are GET, POST, PUT, and DELETE. It includes:

- Check the JSON Schema validation, the Field Type, and the Mandatory Fields
- Validate the Response headers and Positive Test cases response
- Identify the handling of API error codes
- Verify the HTTP response and its code status
- Validate Response payload to determine their format and readability
- Verify APIs with input parameters.
- Validate end-to-end CRUD (create, read, update, and delete) flow for application API

Swagger is used for the REST API documentation that is provided. The document explains the input parameters, output responses and endpoint URLs, as well as the models of the API.
Automated tests check response code, response message and response body in API Testing test cases. They make sure that the test harness verifies the functionality as well as expose the failures. Exploring the implemented REST API was made to ensure that the API produce useful results from successive calls. The conclusion is that the received responses do not contain clear, useful and comprehensive JSON text. For some calls there is not JSON text in the response at all(Edit Post, Edit Comment, …). All bugs found during this stage can be found here.
The approach then was to figure out way to use the implemented REST API features and responses to ensure that the required functionalities are working in a way that was meant to.
This approach includes the use of different API calls and using the provided data from responses to receive from the API confirmations that it produces useful results from successive calls, the requests perform the required operations and data is stored as it should.

Number of test cases executed:44

PASS: 40

FAIL: 4

Number of issues found: 20

### Missing Features

- DELETE User
- GET News Feed

### Extra features

- Skills – CRUD operations

## 5. Automation Testing

User Interface Automation Tests

For the UI automation tests, we used Java along with Selenium WebDriver, These tests have been thoughtfully organized into four distinct groups, each tailored to specific functionality:

- **SeleniumUserTest:** This group focuses on testing user-related functionalities.
- **SeleniumConnectionTest:** These tests are designed to evaluate the connection-related features.
- **SeleniumPostTest:** Here, we scrutinize the behavior of post-related functionalities.
- **SeleniumCommentTest:** This group examines the application's response to comment-related actions.

### Page Object Model (POM):

To enhance maintainability and code organization, we implemented the Page Object Model (POM). This design pattern places the necessary methods for each test in dedicated reference pages, simplifying test case development and maintenance.

### Comprehensive Test Architecture:

Because of the scope of the project we decided to merge the RESTAssured API tests which allowed us to perform API requests and establish essential preconditions and post-execution actions. This synergy of Selenium and RESTAssured enhances our ability to maintain a clean test environment by removing unnecessary data after test execution.

This testing framework not only ensures the reliability and consistency of the UI with efficient handling of backend operations.

Number of test cases executed: 14

PASS: 14

FAIL:0

Number of issues found:

## Lessons learned

## Conclusions

Based on the test results, at this stage we would recommend postponing the application in order to fix the registered bugs and add the missing elements from the specification.