Club-8
Software Test Report

1. Introduction

1.1 Purpose and Scope

Software testing is a technique for determining whether the actual software product meets the specified requirements and ensuring that it is defect-free. It entails the use of manual or automated methods to evaluate one or more properties of interest by executing software or system components. In contrast to actual requirements, software testing's goal is to find mistakes, gaps, and missing requirements.

This document is the software test report of the social interest club app testing phase of the Club-8 software development project. It contains the results of tests, which were executed during the testing.

1.2 Descriptions and Abbreviations

STR	*Software Test Report*
UC	*Use Case*
TC	*Test Case*

2. Descriptions of Test Activities

2.1 Document Security

This report covers the findings of the "Club-8" web application's testing. It's full of top-secret information. As a result, only authorized individuals should read this report.

2.2 Data Recording, Reduction and Analysis

We used a variety of services and technologies to assist us test our idea. Most of them are manual and other are semi-automatic. For example, during the unit test, when we prepare test data to see if the entered email is valid, we used scripts to generate emails and hand-wrote a few emails to catch errors in unusual circumstances. This aspect of the data generation is semi-automatic because one part of the data was generated automatically while the rest was done manually. The results are then saved for analysis, and the appropriate team is notified. Bug fixing involves analyzing and notifying failure data. Fixing problems is critical to the application's dependability. Checking email is also included in numerous test cases, such as login, signup, alter settings, and so on.

3. Test Preparation

3.1 Hardware Preparation

Ideally, we should keep the integration tests separated from the unit tests and should not run along with the unit tests. We can do this by using a different profile to only run the integration tests. A couple of reasons for doing this could be that the integration tests are time-consuming and might need an actual database to execute.

While testing, developers used their own computers. Other than this we used Docker containers to store data and run the tests.

3.2 Software Preparation

While preparing the data manually, we should select all possible values for an input and back-end variable. We store the data on a MySQL Docker Container. To create testing environment, we used Spring Boot's built-in JUnit4 testing environment.

- There are some predefine constraints for test execution:
  - Actual application is running on Docker engine version v20.10.6 and MySQL version of 5.6. However, tests run on H2 database.
  - JDK version must be 8 or higher (preferably JDK 11 Amazon Coretta).

After we run the Testing Application, Spring automatically tests all the test codes that are provided by the developers and makes back-end & front-end integration tests.

4. Test Results

[Use case tests are performed in accordance with the Test Status Definitions designed in addition to the Software Requirement Definition. Add the Test Status Definition files that you have completed to this report.

Refer to Test Status Definitions (specifying the file name) from following table. For each use case, type the number of activity paths tested and the number of non-working activity paths

If modifications for non-working activity paths have been made ,add the information about the tests (retest) that you run after this fix at the end of the table.]

UC Number	UC Name	TC File	Retest?	Number of Activity Paths	Number of Non-working Activity Paths	Time Spent(min.)
1	Create Club	ClubController	No	4	0	5
2	Create Sub Club	SubClubController	No	9	0	10
3	Edit Club	ClubController	No	2	0	2
4	Edit Sub Club	Subclub Controller	No	3	0	4
5	Create Post	PostController	Yes	35	4	45
6	Create Comment	CommentController	No	9	1	20
7	Like Post	LikeController	No	4	0	5
8	Login	HomeController	No	14	1	30
9	Register	HomeController	No	7	0	10
10	Create Subclub Request	LogController	No	2	0	4
11	Report Member	MemberController	No	3	0	4
12	Send Warning to a Member	MemberController	No	3	0	4

13	Ban Member	MemberController	No	2	0	2
14	Unban Member	MemberController	No	2	0	2
15	View Profile	ProfileController	No	5	0	8
16	View Sub Club	SubclubController	No	7	0	10
17	View Post	PostController	No	2	0	3
18	View Reports	LogController	No	2	0	3
19	View Requests	LogController	No	1	0	1
20	View Warnings	LogController	No	1	0	1
21	View Errors	LogController	No	1	0	1
22	View Infos	LogController	No	1	0	1
23	Join Sub Club	SubclubController	No	2	0	3
TOTAL	**X**	**X**	**X**	**86**	**6**	**177**

4.1 Criteria for Evaluating Results

Most of the use cases are implemented without encountered any issue. Failed tests reported to the developer team to fix. To satisfy the users of Club-8, we need to consider following criteria:

1) All use cases must be reliable.
2) System responses should be fast, back-end operations should be short as possible.
3) After a problem occurs, users should not be affected of that problem. System admins should be warned via Error or Warning Logs that are in their admin panel menu.

5. References

[1]: https://junit.org/junit5/

[2]: https://www.baeldung.com/spring-boot-testing

[3]: https://spring.io/guides/gs/testing-web/