Test Document

for

SciQuizzy

Table of Contents

1. Objectives

1.1. Purpose

This document describes the plan for testing SciQuizzy, a web application. This Test Plan document supports the following objectives:

- Identify existing project information and the software that should be tested.
- List the recommended test requirements (high level).
- Recommend and describe the testing strategies to be employed.
- Identify the required resources and provide an estimate of the test efforts.
- List the deliverable elements of the test activities.

1.2. Scope

This Test Plan describes the integration and system tests that will be conducted on SciQuizzy following the completion of the web application by integrating the subsystems and components identified in the Software Requirements Specification.

The interfaces between the following subsystems will be tested:

- Login system
- Main quiz
- Leaderboard

2. Requirements for test

The listing below identifies those items (use cases, functional requirements, non-functional requirements) that have been identified as targets for testing. This list represents what will be tested.

2.1 Files System Integrity Testing

- Verify access to user files during login in and sign up
- Verify access to leaderboard files during display of leaderboard

2.2. Function Testing

- Verify the functionalities used in login system ie. Functions to check for user's existence, adding new user during sign up and function to verify password during login in
- Verify the functionalities used in the main quiz ie. Functions involved in getting questions from API, displaying questions, keeping track of timer, handling the submit, switching over to the next question and display final score
- Verify functionalities used in fetching, displaying and updating leaderboard

2.3. User Interface Testing

- Verify ease of navigating through the pages
- Verify button clicks

- Verify mobile responsiveness

2.4. Performance Testing

- Measure time to fetch user details and leaderboard from the files
- Measure time to fetch questions from the API
- Measure time taken for transition between pages

3. Test Strategy

The Test Strategy presents the recommended approach to the testing of the software applications. The previous section on Test Requirements described what will be tested; this describes how it will be tested. The main considerations for the test strategy are the techniques to be used and the criterion for knowing when the testing is completed.

3.1. File System Integrity Testing

Test Objective	Ensure file access methods and processes function properly and without data corruption
Technique	• Invoke each file access method and process, seeding each with valid and invalid data • Inspect the files to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved
Completion Criteria	All file access methods and processes happen without any data corruption

3.2. Function Testing

The goals of these tests are to verify proper data acceptance, processing and retrieval

Test Objective	Ensure proper data entry, processing, and retrieval
Technique	Execute each use case, use case flow, or function, using valid and invalid data and verify whether expected results are produced for valid data and appropriate error messages and displayed for invalid data.
Completion Criteria	All planned tests have been executed and identified defects have been addressed

3.3. User Interface Testing

User Interface testing verifies a user's interaction with the software. The goal of UI Testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the applications.

Test Objective	Ensure that window to window and field to field navigation through the application

	properly reflects requirements, including the use of access methods (tab keys, mouse movements, accelerator keys). Ensure that window objects and characteristics, such as menus, size, position, state, and focus conform to standards.
Technique	Create and modify tests for each window to verify proper navigation and object states for each application window and objects
Completion Criteria	Each window successfully verified to remain consistent

3.4. Performance Testing

Performance testing measures response times, transaction rates, and other time sensitive requirements. The goal of Performance testing is to verify and validate the performance requirements have been achieved

Test Objective	Validate System Response time for transactions such as fetching user and leaderboard details from files, fetching questions from API, etc
Technique	Used timer for various test cases
Completion Criteria	Successful completion of the test scripts without any failures and within the expected time allocation

3.5. Tools used

Postman, Browser console

4. Resources

Human Resources	Responsibilities
Ankita Priya	File system integrity testing
Nanthana	Function testing
Sayanti Maiti	User interface testing
Suhani Bhardwaj	Performance testing

5. Schedule of Testing Activities

Milestone Task	Start Date	End Date
Test Planning	1/4/23	2/4/23
Test Design	2/4/23	4/4/23
Test Development	7/4/23	10/4/23
Test Execution	14/4/23	18/4/23

Test Evaluation	19/4/23	22/4/23

6. Test Cases and Test Data

Test Case ID	Test Case Description	Test Case Data	Expected Results
TC01	Check user login with valid data	Email id: test2@0.com Password: 123	The message “Successfully logged in” must be displayed and user must be redirected to the main quiz.
TC02	Check user login with invalid data (incorrect password)	Email id: test2@0.com Password: 12	The message “Incorrect password” must be displayed and user must stay in the login page
TC03	Check user login with invalid data (incorrect email)	Email id: 2@0.com Password: 123	The message “User doesn’t exist” must be displayed and user must stay in the login page
TC04	Check user sign up with valid data	Email id: test3@1.com Password: 1234 Username: test3	The message “User registered” must be displayed and user must stay in the login page
TC05	Check user sign up with invalid data (with email that is already used)	Email id: test2@0.com Password: 1234 Username: test3	The message “User already exists” must be displayed and user must stay in the login page
TC06	Check user sign up with invalid data (without an email id)	Email id: Password: 1234 Username: test3	The message “Fill in all the fields” must be displayed and user must stay in the login page
TC07	Choose a correct answer in the quiz	Question no. – 3 Correct option – 2	The message “Your answer is correct” must be displayed
TC08	Choose an incorrect answer in the quiz	Question no. – 4 Incorrect option – 1	The message “Your answer is incorrect” must be displayed along with the correct answer
TC09	Head to leaderboard with updates	Click “leaderboard” with the user setting a new record	The user must be redirected to the leaderboard and the users position in the leaderboard must be updated
TC10	Head to leaderboard without updates	Click “leaderboard” with the user being unable to break the old record	The user must be redirected to the leaderboard and the users

			position in the leaderboard must stay the same
TC11	Click retry button	Click “retry”	A new quiz must begin
TC12	Click logout button	Click “logout”	The user must be redirected to the login page

7. Requirement Tracability Matrix

Test case ID	Requirements	User login	User sign up	Choosing an option	Go to leaderboard	Click retry button	Click retry button
TC01		✓					
TC02		✓					
TC03		✓					
TC04			✓				
TC05			✓				
TC06			✓				
TC07				✓			
TC08				✓			
TC09					✓		
TC10					✓		
TC11						✓	
TC12							✓
No. of test cases passed		3	3	2	2	1	1

8. White Box Testing

White box testing is a testing technique that examines the program structure and derives test data from the program logic or code. Unit testing was performed first where each block of code was rigorously verified and analyzed using a wide range of test cases. The techniques involved are

8.1. Statement coverage

This technique is aimed at exercising all programming statements with minimal tests

Total number of statements = 1996

Number of statements exercised = 706

Statement testing = (706/1996) * 100 = 35%

8.2. Branch coverage

Total number of decision outcomes = 13

Number of decision outcomes tested = 12

Branch testing = (12/13) * 100 = 92%

8.3. Path Coverage

Total number of paths in the program = 25948

Number of paths exercised = 8472

Path coverage = (8472/25948) * 100 = 32%

9. Black Box Testing

The test cases chosen above are in accordance with the requirements of black box testing. The categories of black box testing that were considered and verified are as follows:

9.1. Functional testing

Test cases were designed for the functionalities that happen in the web app. The functionalities that were verified are login, signup, attempting the quiz, navigating to the leaderboard, retrying and logging out.

9.2. Non-Functional testing

Test cases were also framed for testing all the branches in the functionalities. They elucidated below:
Login: The input given by user can be valid or invalid

Signup: The input given by user can be valid or invalid or may have already been used by another user

Main quiz: The option chosen by the user can be correct or wrong

Leaderboard: The user may or may not break his/her previous records

10. Cover Sheet for Program Unit Notebook

Section	Contents	Due date	Completed date	Review date
1	Requirements	4/3/23	7/3/23	8/3/23
2	Architectural design	22/3/23	23/3/23	23/3/23
3	Detailed design	27/3/23	27/3/23	28/3/23
4	Test plan	1/4/23	2/4/23	3/4/23

5	Source code	14/4/23	15/7/23	15/7/23
6	Test results	16/4/23	18/7/23	19/7/23
7	Change requests	23/4/23	25/4/23	25/4/23
8	Notes	27/4/23	29/4/23	29/4/23

Release Approval: _____ Date: _____