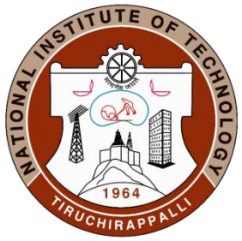


CSPE41

Software Engineering

SciQuizzy

Acknowledgement

We would like to take this opportunity to express our gratitude to all the people who have contributed to the successful completion of this project of a quiz website. Firstly, we would like to extend our heartfelt thanks to our project mentor Dr. C. Ostwald, professor of Software Engineering, NITT for providing us with valuable guidance, support, and feedback throughout the project. Your expertise, insights, and encouragement have been instrumental in shaping this project and making it a success. We would also like to thank our team members for their dedication, hard work, and collaboration in bringing this project to fruition. Your contributions, ideas, and efforts have been invaluable in ensuring the quality and functionality of the website. We would like to express our appreciation to our family and friends for their unwavering support, encouragement, and motivation. Your love and support have been a constant source of inspiration and strength for us. We would also like to thank the contributors of the open-source tools and frameworks used in this project. Your work and dedication have made it possible to build this website quickly and efficiently. Lastly, we would like to express our gratitude to our institution for providing us with the opportunity and resources to undertake this project. We are grateful for the learning and growth opportunities that this project has provided us. Thank you all for your support and contributions to this website project. Your efforts have helped us learn, grow, and achieve our goals.

Team name: Star Techies

Team members:

1. Ankita Priya - 106121014
2. S. Nanthana - 106121081
3. Sayanti Maiti - 106121117
4. Suhani Bhardwaj – 106121135

Table of Contents

Abstract

This project aims to develop a website that enables its users to conduct and take quizzes on a variety of topics. The website will provide an intuitive and user-friendly interface that allows users to easily sign up by creating an account and then login to conduct and take quizzes. The website will be designed to support multiple choice questions. Users will be able to track their quiz performance and view detailed analytics that provide insights into their strengths and weaknesses. At the end of the quiz, the user can see his total score and also his position among others who have attempted the quiz on the leader board. The website will be optimized for mobile devices to ensure users can access quizzes anytime, anywhere. Overall, the website will provide an engaging and interactive platform for users to improve their knowledge and skills by taking quizzes on various topics.

Software Requirements Specification

For

SciQuizzy

SoftwareEngineering

April 30, 2023

1.Introduction:

1.1 Purpose:

This document presents a detailed description of the Quiz Website. It consists of a complete analysis of the system describing its different features of it and its uses. It also defines the abilities and guidelines of the system. The system allows the users to check their performance in a particular topic by answering a set of questions in the given time limit. It also helps in increasing one's scope of knowledge in a particular topic.

1.2 Document Conventions:

The document follows the IEEE format standard (IEEE Std. 830-1998).

1.3 Intended Audience:

The intended audience of this document is Dr. C. Ostwald, professor of Software Engineering, students of NIT Trichy and anyone who has an interest in playing quizzes to test and enhance their knowledge.

1.4 Project Scope:

This quiz website is beneficial to both students and teachers. Students can use this to test their knowledge on a particular topic. It also helps in enhancing their knowledge. Through the quizzes and additional features of time limit per question and score, a teacher can check the progress of his/her class students.

1.5 References:

Pressman, Roger S. Software Engineering: A Practitioner's Approach. McGraw-Hill, 2005.

2. Overall Description

2.1 Product Perspective:

This project represents a simplified version of a quiz website. At a higher level, this project will help the students to challenge themselves and their friends with their knowledge and skills by competing with each other. It will also help the teachers to grasp the student environment and fill in the gaps within the manner of teaching and knowledge. The main goal of this is to help make learning and teaching a fun process for both students and teachers.

2.2 Product Features:

2.2.1 Login/Sign in

Description: This function allows a registered user to log in his account using his mail id and password. If a user is not registered, the website allows the user to sign up first.

Rationale: This provides a sense of security to the user that their information is secure.

2.2.2 Sign Up

Description: This function allows new users to register themselves on the website. This function asks for a username, mail id and password to make a new account. Only unique email ids are allowed. In case the entered email id already exists, the user cannot proceed further unless he enters a new email id. Once the user has registered, he can now use the login function to access the quiz.

Rationale: A user who wishes to give the quiz, must be logged in. However, without signing up, a user cannot access the quiz. This section offers all users a chance to become a member and then access the quiz.

2.2.3 Main Quiz

Description: The main quiz feature consists of a fixed number (10) of interesting sets of multiple-choice questions from various Science topics. On choosing one of the options, it will be shown instantly whether the answer given is correct or wrong.

Rationale: These quizzes are the best way to test one's knowledge and understanding of science by figuring out what your strengths and weaknesses are by answering a varied set of interesting questions. It also helps the users to learn new facts and enhance their knowledge whenever they stumble upon a new question.

2.2.4 Timer

Description: The quiz timer is an important aspect of the quizzing system. In this feature, the user is given only a fixed amount of time to answer a particular question. After the time limit per question is over, the question is immediately changed and the user cannot revert back to the question to change his answer. The time limit used in this project is 7s per question. The bar above the question indicates the time left for a particular question.

Rationale: A quiz timer ensures that the users do not spend too much time on a particular question and answer spontaneously. A fixed time limit helps the user to stay alert and not get distracted. It ensures that the quiz goes on at a steady pace and also increases the level of excitement among the users.

2.2.5 Leaderboard:

Description: Once the quiz is completed, the users can see the leaderboard. The leaderboard consists of usernames and scores of all the participants of the quiz.

Rationale: A leaderboard helps the user to see where he stands among the other participants. This motivates the user to work more on their weaknesses and improve his performance in the next quiz. With the leaderboard, a healthy competitive spirit is built around which makes the quiz more fun and exciting.

2.3 User Classes and Characteristics:

The main classes in the system are

(i) The user (ii) A question (iii) Leaderboard

The user answers a set of questions. For each question, the user either gets a score for every correct answer or no marks for every wrong answer. Every user's final score is recorded in the leaderboard and displayed.

Brief descriptions of these classes are:

- User:
  - Has properties like username, email id and password
  - Associated with the total score after every test and position on the leaderboard
- Question:
  - Has properties like question id, correct answer, incorrect answer, difficulty level
  - Associated with a score per question and time limit per question.
- Leaderboard:
  - Has properties like username and total score

One user is associated with many questions, which means that there exists a one-to-many relationship between the two.

One user is associated with one entry in the leaderboard. This indicates a one-to-one relationship.

3. System Features

This section provides detailed requirements for the website design, including functional requirements.

3.1 General requirements

3.1.1 Login

Description and priority:	This function allows a registered user to log in to his account using his mail id and password. If the user is not registered, the website should allow the user to sign up first. The system will check both the mail id and password when the user attempts to log in. The mail id and password are unique for every user. So if another user is trying to create another account with the same mail id, then the system does not allow it. Both the mail id and password should match with the registered account for the user to be able to log in into the quiz.
Inputs:	Mail id and password
Source:	All inputs are provided by the user.
Outputs:	Indication that the user successfully logged in!
Destination:	The outputs are displayed on the screen as well as stored in the system.
Requires:	The user provides login information, i.e. mail id and password
Pre-conditions:	The user should have signed up already.
Post-conditions:	The user is logged in to system.
Side-effects:	None

3.1.2 Sign up

Description and priority:	This function allows unregistered users to enroll and to create a new account with the website. In order to create a new account, the user has to provide the required information such as username, mail id and password.

	The system checks if the mail id is unique or not, otherwise, it prompts the user to enter a new mail id. The system also checks if all the details have been entered, only then the user is allowed to proceed further.
Inputs:	Username, mail id and password
Source:	All inputs are provided by the user.
Outputs:	Once the account has been registered, the user is directed back to the log in page, so that the user can now log in using the credentials used for sign up.
Pre-conditions:	The user must not have existing account with the website.
Post-conditions:	Once the account is created, the user is directed back to the first page so that he can log in now.
Side-effects:	None

3.1.3 Main Quiz

Description and priority:	Once the user has logged in, he can start the quiz. The quiz consists of 10 multiple-choice questions each with a time limit of 7s. After 7s, the user is displayed the next question on the screen. When the user clicks on an option, it is immediately shown if the answer is correct or wrong. Once the question is answered, the user can move forward using the next option. For every correct answer, the score is updated and recorded.
Inputs:	Users can choose one of the options as their answer to the question.
Source:	The options are already displayed on the screen.

Output:	If answered within the time limit, then a pop-up shows if the answer is correct or wrong. If the time exceeds, then the next question is displayed.
Destination:	The questions are displayed on the screen and the bar above the question indicates the time left for each question. The score is updated after every correct answer and displayed at the end of the quiz.
Pre-conditions:	The user must be logged in.
Post-conditions:	At the end of the quiz, the total score is displayed.

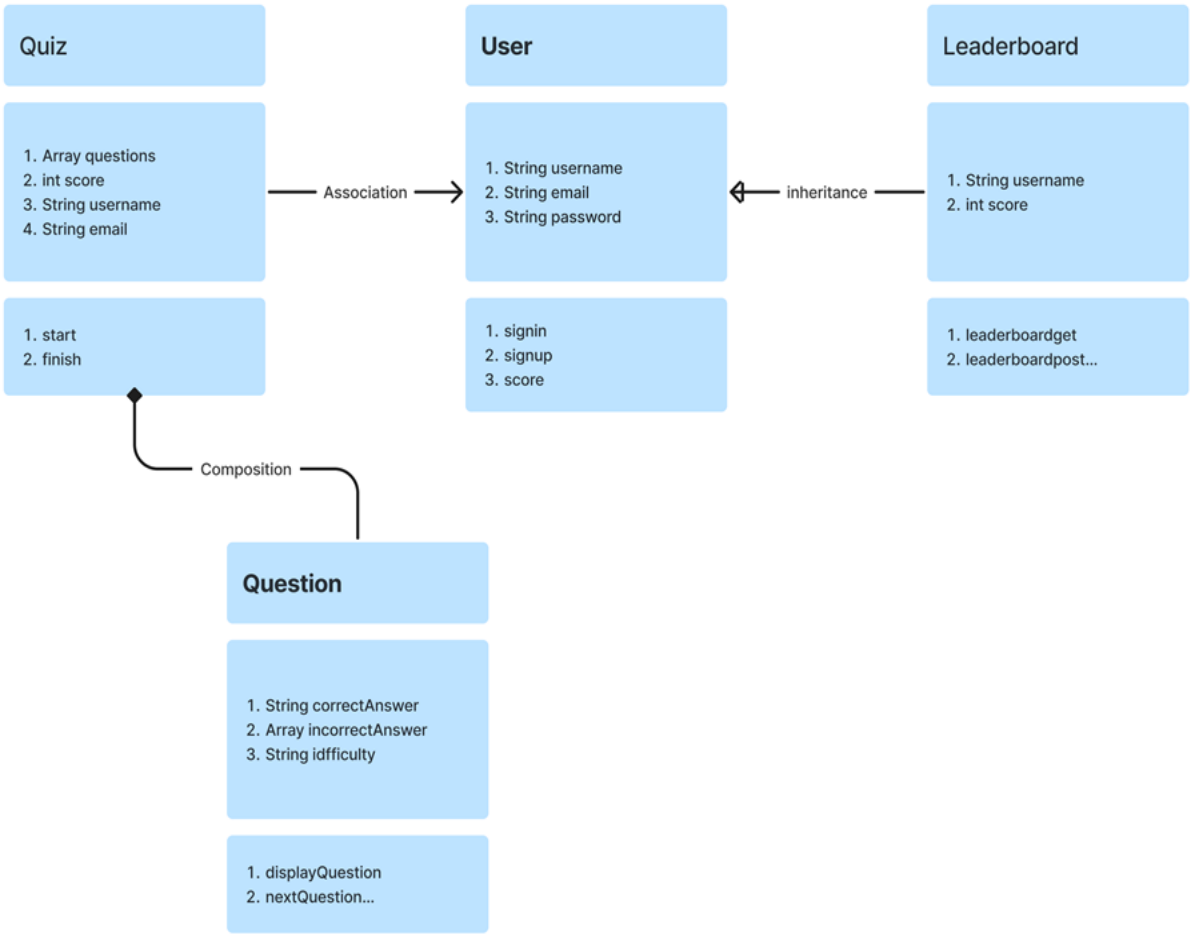
3.1.4 Leaderboard

Description and priority:	At the end of the quiz, the user can see his total score and also his position among others who have attempted the quiz. The leaderboard displays the username and scores of all the participants in the sorted order- that is the user with the highest score is displayed on the top.
Inputs:	No such input. Just at the end of the quiz, click on the leaderboard to see your position.
Source:	The score obtained from the answers given by the users
Output:	Consists of a table displaying the usernames and scores of each user in order.
Pre-conditions:	The user must complete the quiz.
Post-conditions:	The user can just check through the leaderboard for his score and rank among others.
Side-effects:	None

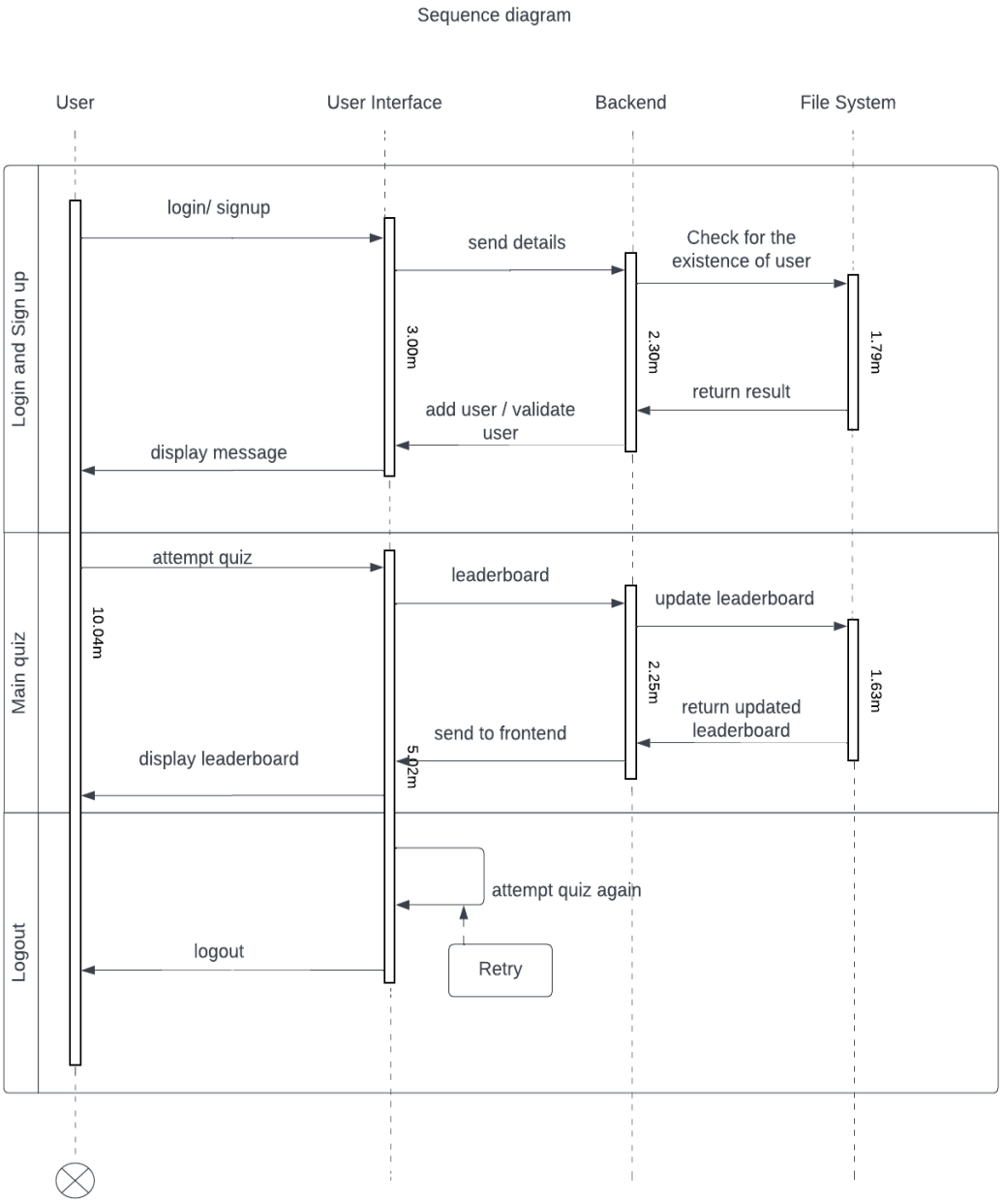
3.2 Diagrams

3.2.1 UML DIAGRAMS:

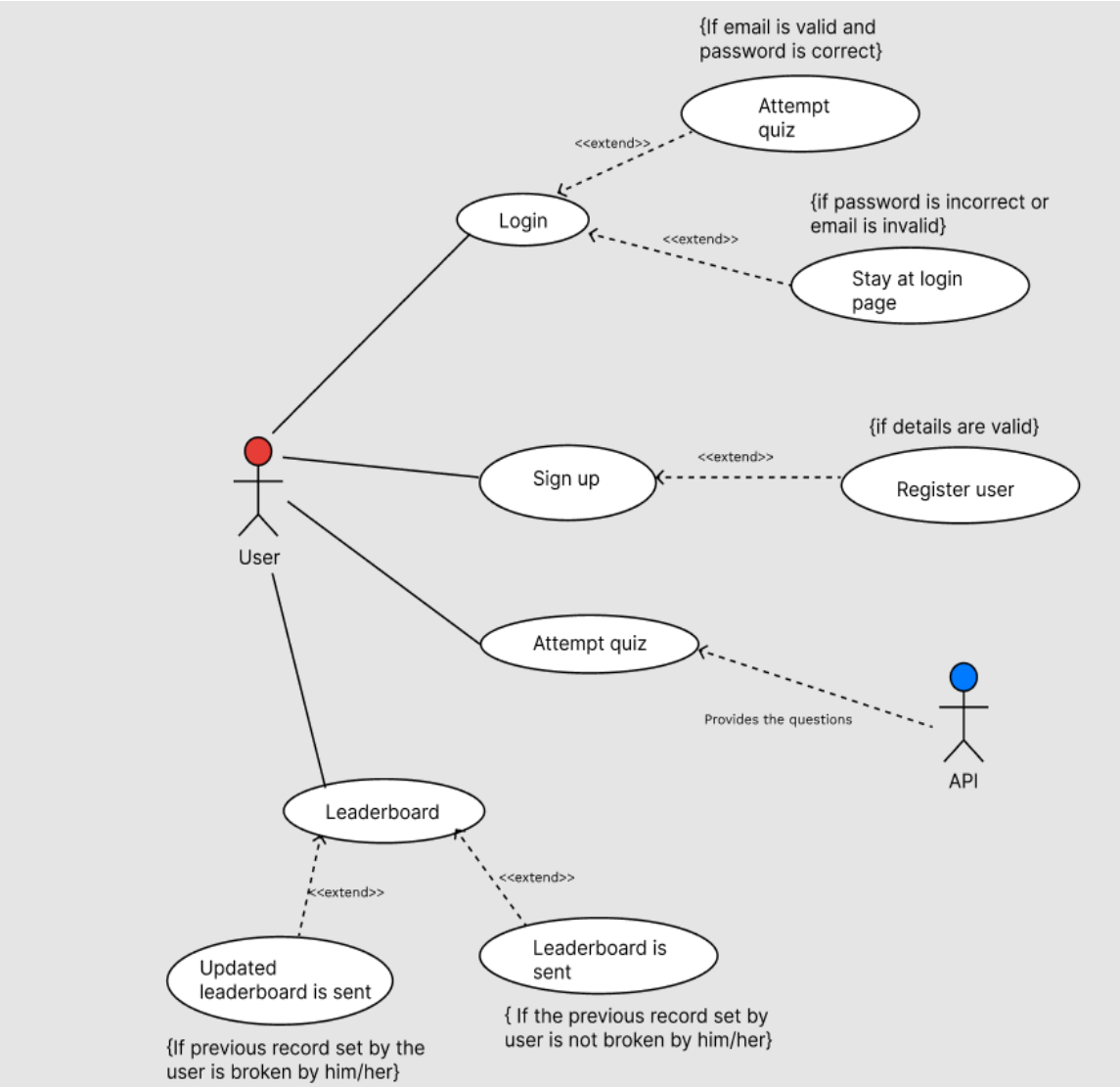
(i) Class diagram:



(ii) Sequence diagram:

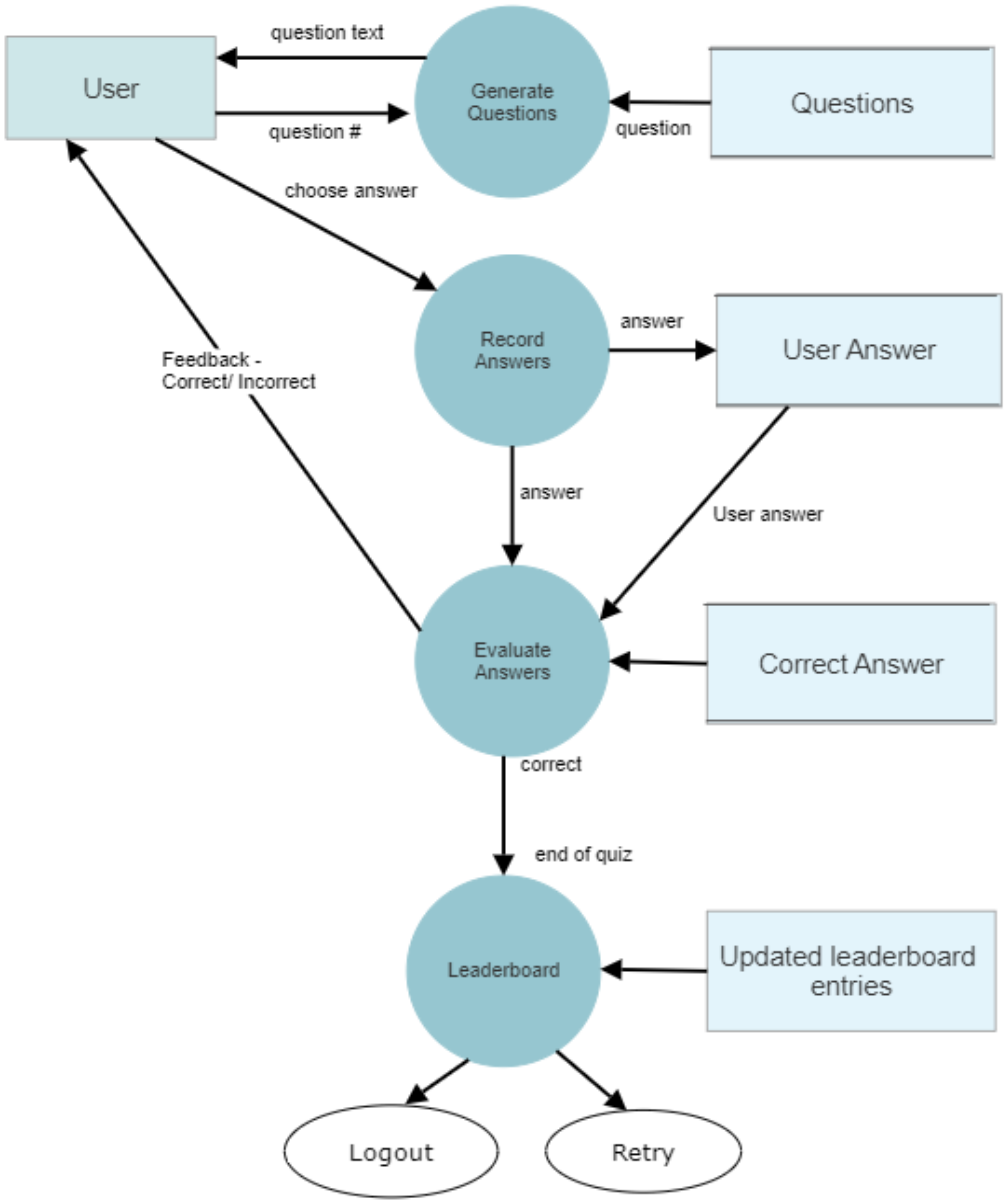


(iii) Use-case diagram:

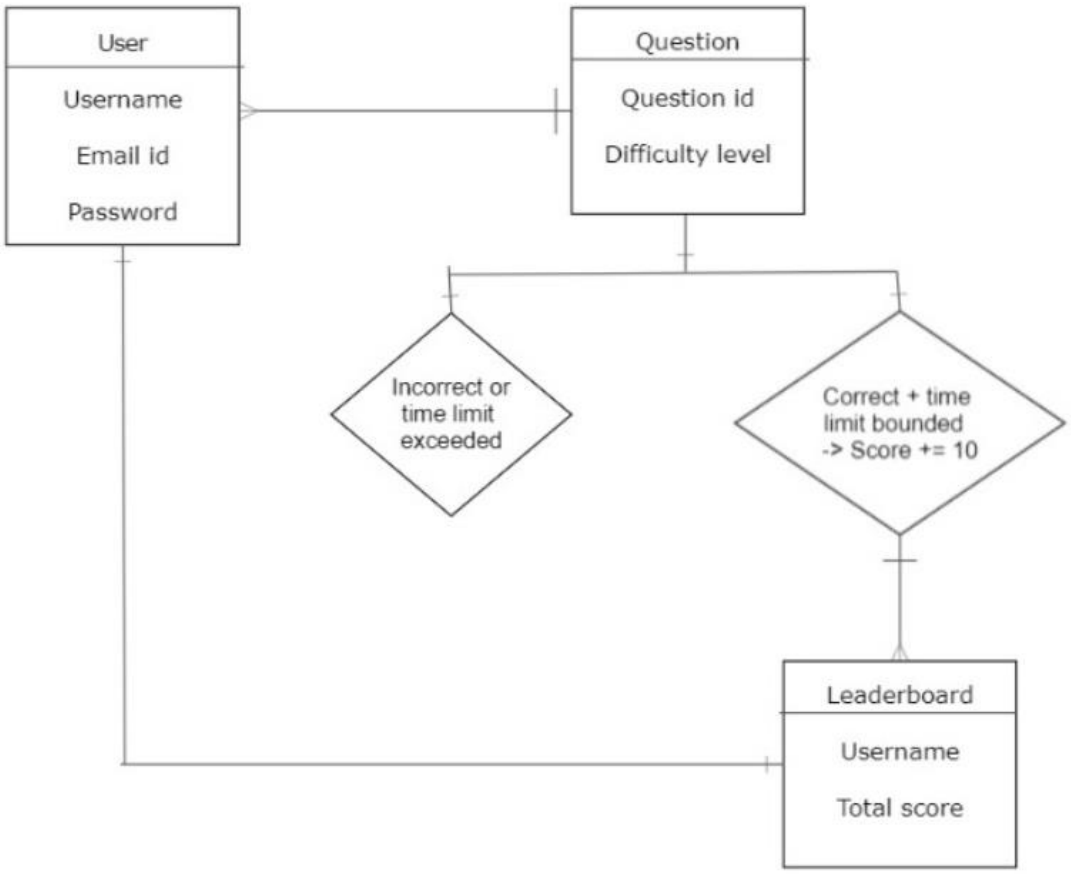


(iv) Activity diagram:

3.2.2 Data Flow diagram



3.2.3 Entity Relationship diagram



4. External Interface Requirements

4.1 User interfaces:

The system has been designed such that it follows all the 3 golden rules of Theo Mandel.

- Rule 1: Place users in control
  - For example: Login system
  - Forgiving, helpful features:
    -Appropriate messages are displayed when input fields are left empty during sign in/sign up.
    - Pop-up messages are displayed if a unique email id is not entered or an incorrect password is entered.
  - Navigable:
    -Buttons to the leaderboard, login page, main quiz

- Rule 2: Reduce the user's memory load
  - Intuitive: Buttons with understandable functionality

- Easy navigation: Uses real-world metaphor. Example: Progress bar

- Rule 3: Make the interface consistent
  - Continuity: Sustain the context of user's tasks
    Log in page-> main quiz-> leaderboard
  - Providing aesthetic appeal and integrity-subtle decorations, gradient background, large buttons for ease of clicking during the quiz.

5. Other Non-Functional Requirements

5.1 Safety requirements:

- The users are asked to enter a password. A strong password must have a minimum of eight characters containing digits and letters both.
- Whenever a new user signs up, only unique mail ids are allowed. If an existing mail id is used, then the user is not allowed to proceed further.

5.2 Software quality metrics:

5.2.1 Usability:

The quizzing website can be used on both Windows and UNIX (Linux) servers.

5.2.2 Security:

Each user has a unique mail id and password. The information related to the user is safe in the system.

5.2.3 Performance:

Performance of the software is measured by determining whether the software is fulfilling the user requirements or not. The website allows the user to test their knowledge through quizzes. Through it, users are able to enhance their knowledge and also work upon their weaknesses.

6. Coupling and Cohesion:

6.1 Cohesion:

- There are separate instances for sign in, sign up features.
- These instances are completely independent of each other and won't affect each other.
- They only depend on the functions in their own instances
- Functional cohesion has been used.

6.2 Coupling:

- database.txt file is accessed by both sign in, sign up features.
- Both the features can modify it.
- Common coupling has been used.

7. Web-App design strategies:

7.1 Design:

- **Intuitive:**
  - Messages appear for every action.
  - There are clearly defined buttons such as submit, retry, leaderboard, logout buttons.

- **Forgiving:**
  - Incorrect actions can be redeemed. For example, if the user forgot to fill in the username before clicking on submit button, then a message will be displayed asking the user to fill in all the required fields.

7.2 Compatibility:

- Can be run on all browsers -Firefox, Microsoft Edge, Google and all Operating systems- Windows and Linux.

7.3 Navigability:

- The navigation on the page is facilitated by well-defined buttons.

7.4 Visual Appeal:

- The page looks visually appealing and pleasing to the eyes because of color coordination, rounded corners, varied font types and font colors.

Code details

Programming practises:
Do's:

1. Goto statements aren't used anywhere in the code
2. Indentation, blank lines, line spacing are properly used to make the program files look more presentable. This can be seen in the following example

```
3.  void check(){
4.
5.    std::string email, pass, line,word;
6.    std::cin>>email;
7.    std::cin>>pass;
8.    std::ifstream f;
9.    std::vector<std::string> s;
10.   s.resize(3);
11.
12.   f.open("database.txt", std::ios::in);
13.   if(f.is_open()){
14.     bool flag = false;
15.     while(std::getline(f,line)){
16.       std::stringstream ss(line);
17.       int i = 0;
18.       while(ss>>word){
19.         s[i] = word;
20.         i++;
21.       }
22.       if(s[1] == email){
23.         flag = true;
24.         if(s[2] == pass){
25.           std::cout<<"200";
26.           f.close();
27.           return;
28.         }
29.         else{
30.           std::cout<<"300";
31.           f.close();
32.           return;
33.         }
34.
35.       }
36.     }
37.     std::cout<<"500";
38.     f.close();
39.     return;
40.   }
41.   else{
```

```
    std::cout<<"404";
    return;
  }

}
```

3. User defined datatypes are used to model entities in the problem domain. This is seen in the following example

```
std::string line,word,score,net,email;

std::ifstream fin;

std::ofstream fout,fnew;

std::vector<std::string> s,res;
```

Don’ts:

1. Nesting is not too deep, it was taken care that the level of nesting did not cross 2. This can be seen in the following example

```
void getleaderboard(){
  std::string line,word;
  std::ifstream f;
  std::string result = "",s = "";

  f.open("leaderboard.txt",std::ios::in);
  if(f.is_open()){
    while(std::getline(f,line)){
      std::stringstream ss(line);
      ss>>word;
      while(ss>>word){
        s = s+word+"***";

      }
      result = result+s+"&&&";
      s.clear();
    }
    f.close();
    std::cout<<result;
    return;
  }
  else{
    std::cout<<"404";
    return;
  }

}
```

2. Limited number of identifiers are used. This can be seen in the following example

```
std::string line,word;

std::ifstream f;

std::vector<std::string> s;
```

Albrecht's Function Point Model

- External inputs: 3
  Consists of username, email id, password
  Simple: 3
- External outputs: 11
  Consists of the score obtained by the user and result for each question (10)
  Average: 10, Simple: 1
- External queries: 4
  Consists of queries for login, signup, getting leaderboard entries, updating leaderboard entries
  Simple: 3, Average: 1
- Internal log files: 2
  Consists of two files, one for storing user details and the other for storing leaderboard entries
  Complex: 2
- External log files: 1
  There is one log file that stores metadata about all the transactions
  Simple: 1

The table containing the weightage is shown below

Item	Simple	Average	Complex
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EFL	5	7	10

I. Unadjusted Function Point Count (UFC) = (3*3) + (1*4) + (10*5) + (3*3) + (1*4) + (2*15) + (1*5) = 111

II. F3, F5, F9, F11, F12, F13 - 0
F1, F2, F6, F7, F8, F14 - 3
F4, F10 – 5

III. TCF = 0.65 + 0.01 * (0 + 18 + 10)
= 0.93

IV. FP = UFC * TCF = 111 * 0.93 = 103.23
Therefore the number of function points developed = 103

Test Document

for

SciQuizzy

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: _____

User Manual

SciQuizzy

1. Introduction

SciQuizzy is a web app that facilitates users to attend instant time-bound quizzes based on scientific concepts, theories and scientific phenomena. It generates intriguing quizzes as many times as the user wants with different non-repetitive questions that can test the users' insight into scientific facts.

1.1. Who can benefit from SciQuizzy:

- Students attending schools and colleges
- Quiz masters who wish to collect questions for conducting Science quizzes
- Science enthusiasts
- Teachers and Professors

1.2. Key features:

- User account creation and login system
- User friendly and intuitive functionalities
- Instant quizzes with a time limit for attending each question
- Evaluation of answers immediately after attempting every question
- Display of correct answers in case the answer chosen by user is wrong
- Scoring system with leaderboard

2. Technical Specifications

Programming languages used: C++, JavaScript, HTML, CSS

Tech stacks and libraries used:

Programming language	Library/ Tech stack	Functionality
C++	String	To handle strings which are sequence of characters
	Sstream	Helps in handling string as if it were a stream
	Vector	Used to implement the sequential container vector
	Fstream	Used to perform read and write on files
JavaScript	Node.js	It is an open source environment that can use JavaScript on server
	Express.js	Web application framework for Node.js that provides robust features for web applications
		Used to compile C++ programs in Node.js applications

	Compile-run	
	Cors	Cross Origin Resource Sharing. Facilitates resource sharing across servers
	Body-parser	HTTP request body that accesses underlying information in a URL
	Axios	HTTP library that interacts with API (Application Programming Interface) services
	Dotenv	Loads environment variables from .env file into process.env

3. Requirements

A stable internet connection and the presence of git (Version control system) are inevitable to install and use the web app

3.1. PC requirements:

- Windows 10 OS or greater
- 4 GB RAM or greater
- 10 GB free space or greater
- Compatible in all Linux environments

3.2. Supported web browsers:

- Chrome - versions 63+
- Firefox - versions 58+
- Opera – versions 50+
- Edge - versions 13+
- Safari – versions 9+

4. Guidelines on how to use SciQuizzy

Once the required packages are installed, the web app is ready to be used

- Ensure a stable internet connection
- Install software according to installation instructions
- Use anti-malware software with frequent updates

5. Getting started

5.1. Installation

5.1.1. Setting up the repository

I. Fork the github repository containing the code base. The repository can be found here.
II. Create a folder in the local machine for the web app
III. Convert it into a git repository by typing the following command in the terminal
IV. Clone the repository forked out by typing the command

```
>> git clone <repository link>
```

V. Head into the folder SE by typing the following command in the terminal

```
>> cd SE
```

5.1.2. Setting up the backend

I. Head to the folder containing the backend by typing the command

```
>> cd Backend
```

II. Create a .env file in it similar to the .env.example file
III. Install Node Package Manager by typing the command

```
>> npm init
```

IV. Install the required dependencies by typing the following command in the terminal

```
>> npm install
```

V. Run the server by typing the command

```
>> npm run dev
```

5.1.3. Setting up the frontend

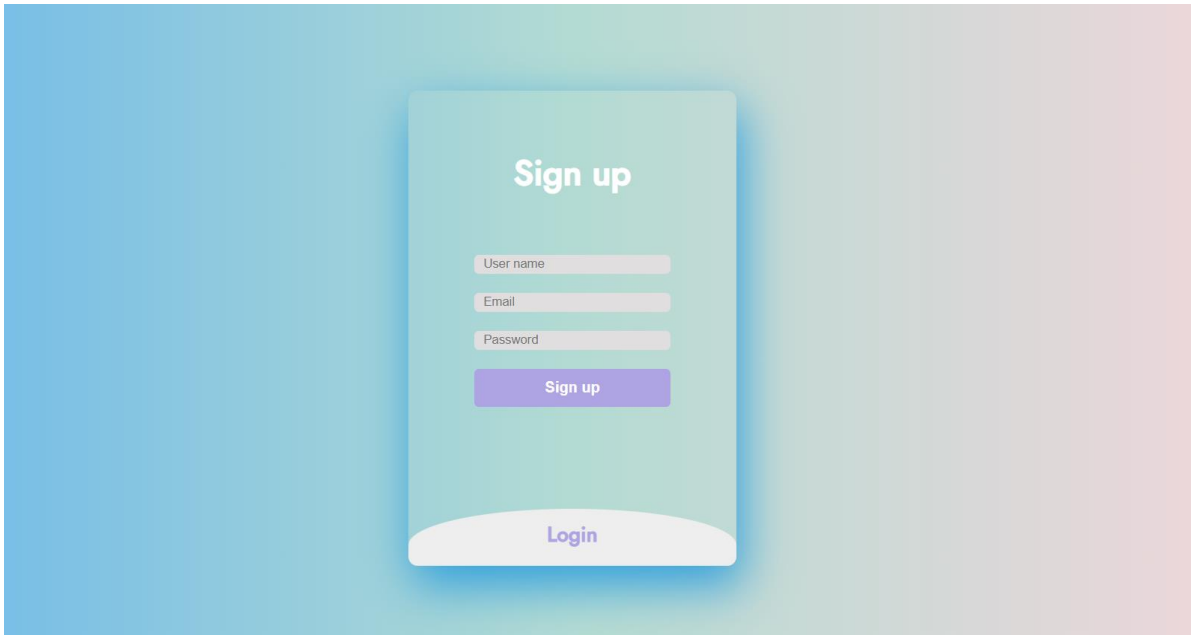
I. After configuring the backend, head to the folder containing the frontend by using the command

```
>> cd ../Frontend
```

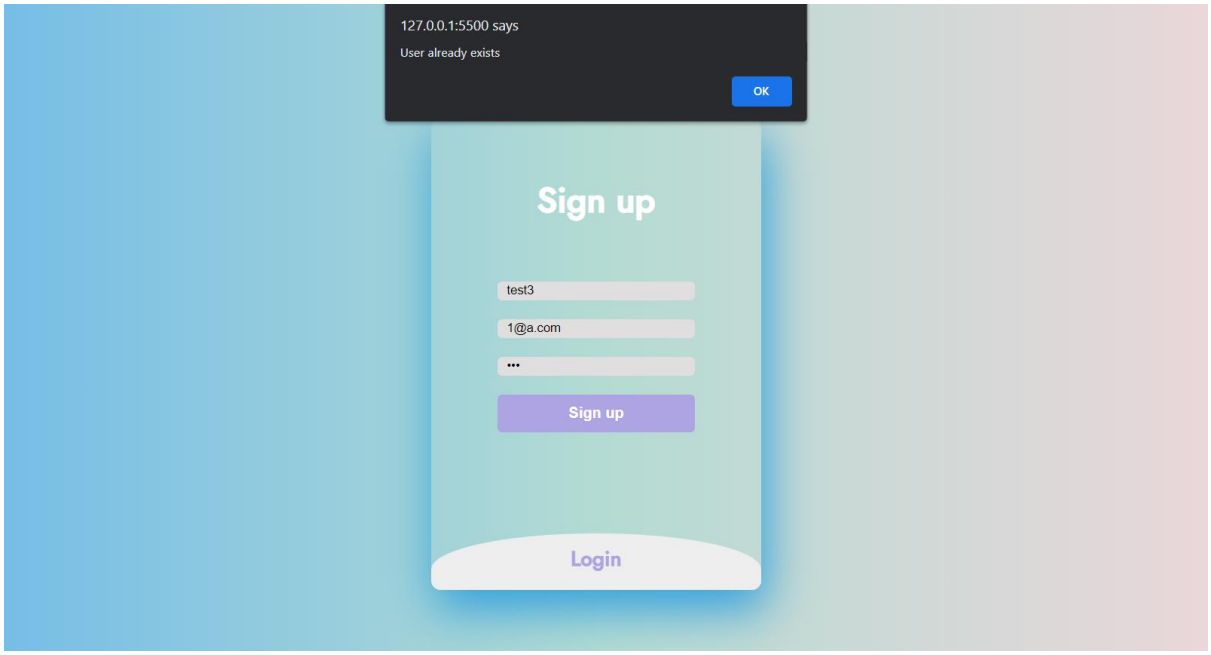
II. Run the login.html file using any of the browsers mentioned above.

5.2. Sign in and Sign up

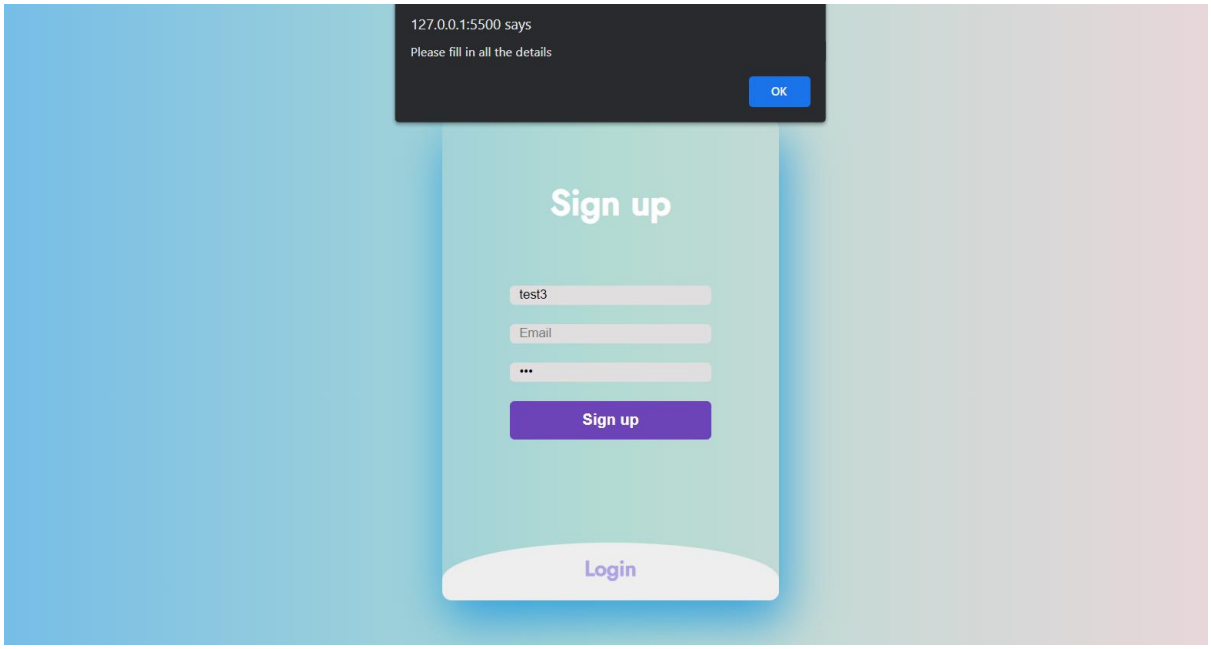
I. New users have to sign up by filling in their email address, username and password



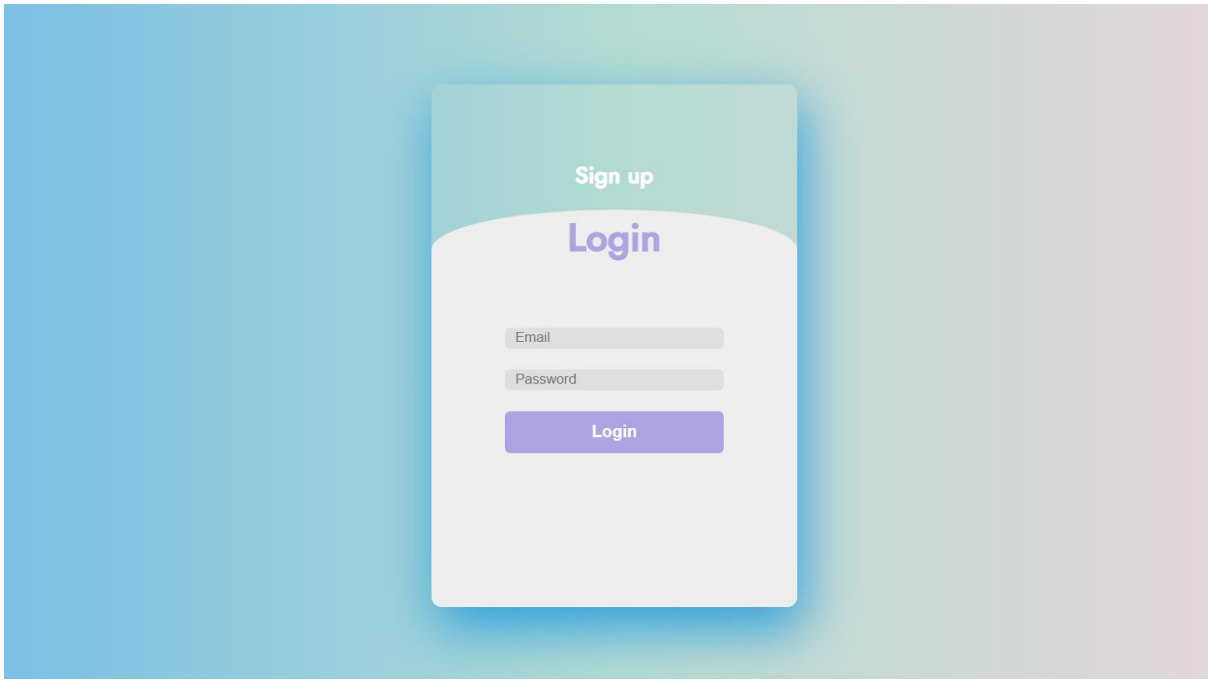
II. If the email id that was entered had already been used by another user, a message will be displayed prompting the user to enter new details again.



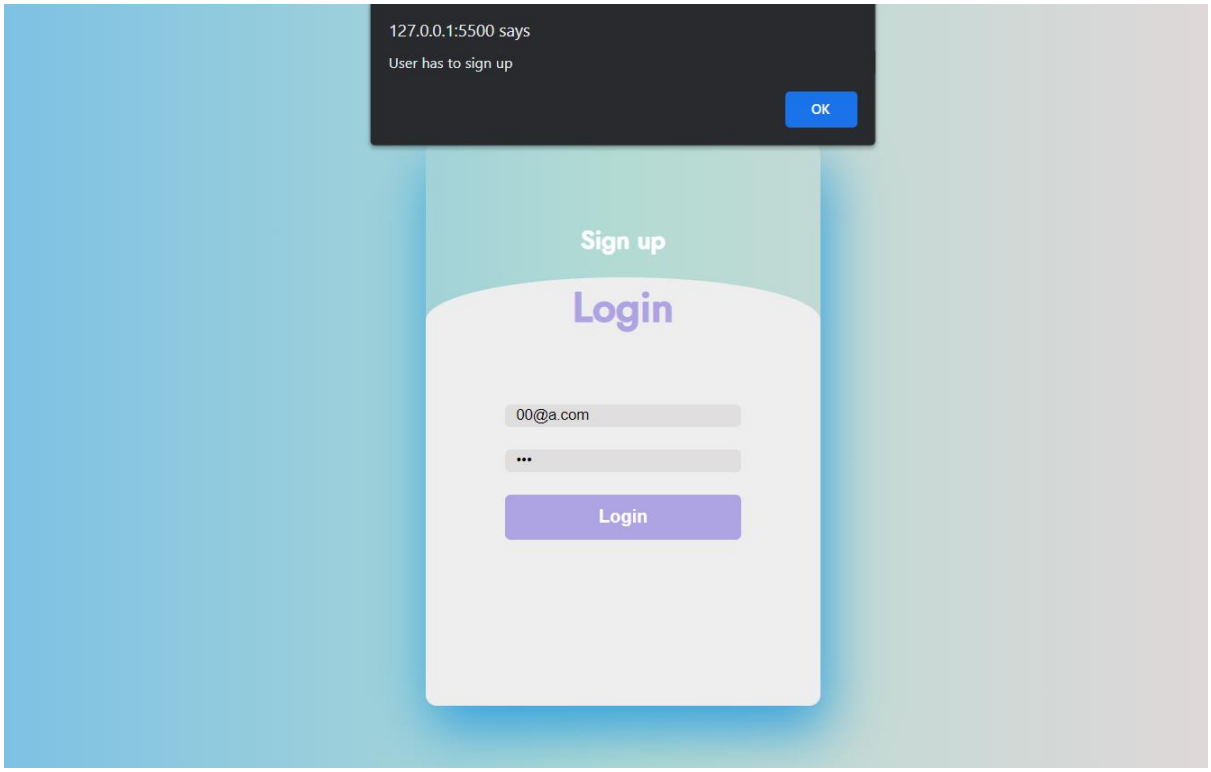
III. If any of the input fields are left empty, a message will be displayed prompting the user to enter all the details again.



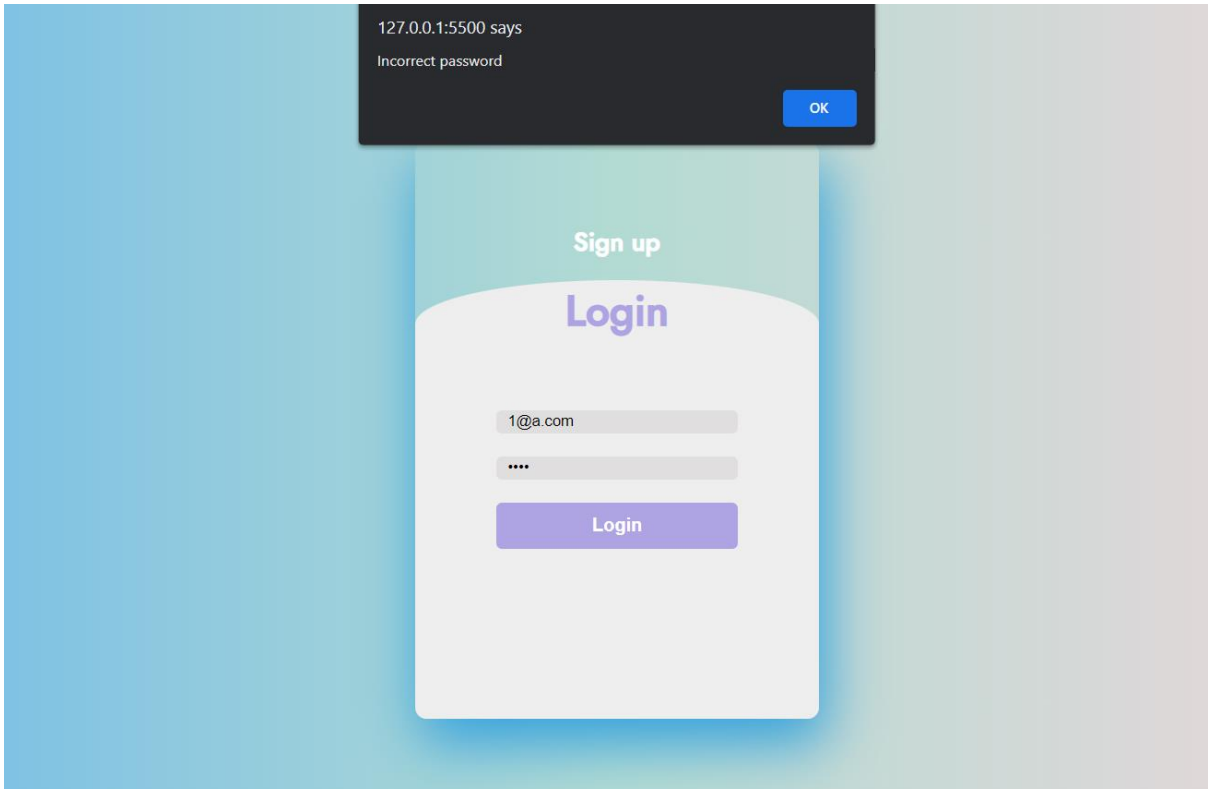
IV. After signing up, enter the email id and password in the login page to attempt the quiz. Users who already signed up can skip step I to III.



V. If incorrect password is entered or if the email id used for signing in is not registered, appropriate messages will be displayed.

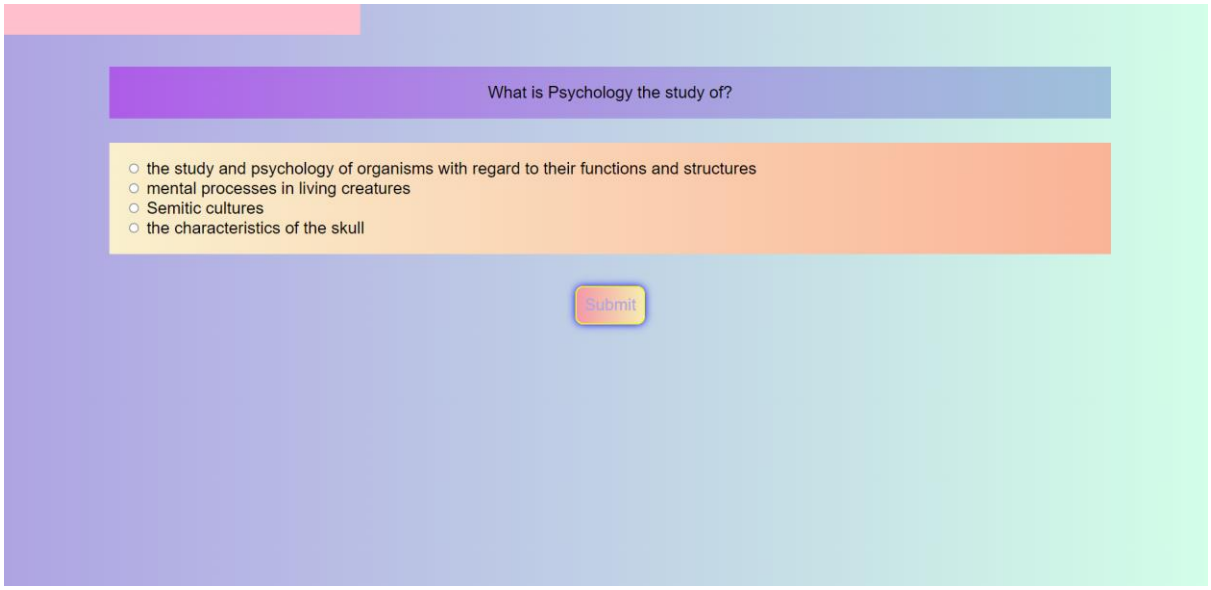


VI. On entering correct details users will be redirected to the page containing the main quiz.



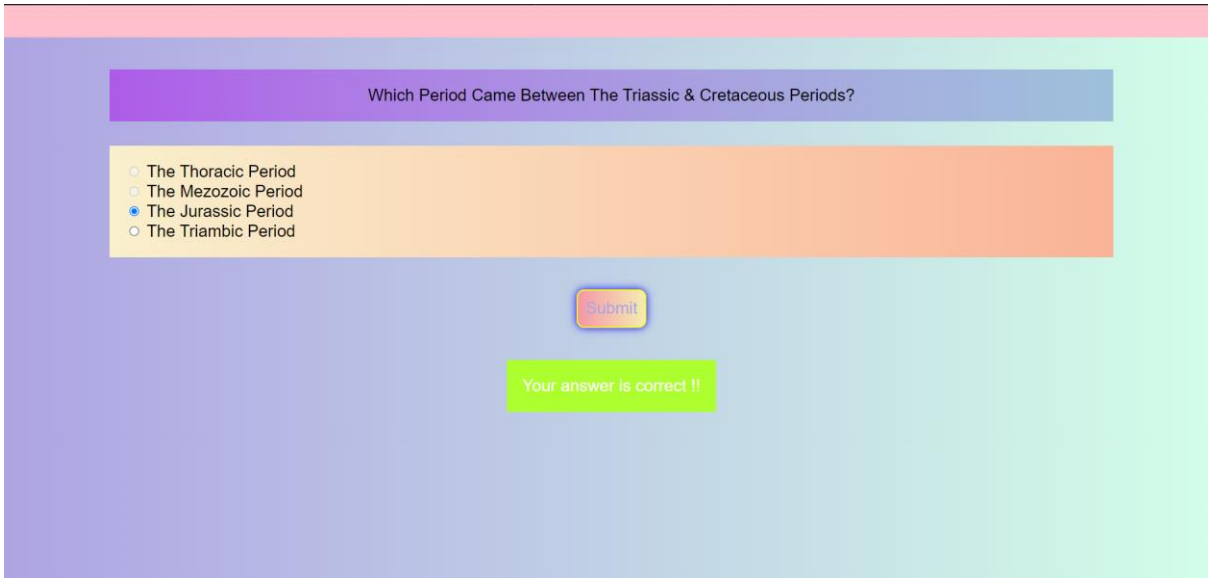
5.3. Attempting the quiz

I. The progress bar on top indicates the amount of time left to attempt a question. Seven seconds will be provided to attempt each question. Each quiz will contain a total of 10 questions.

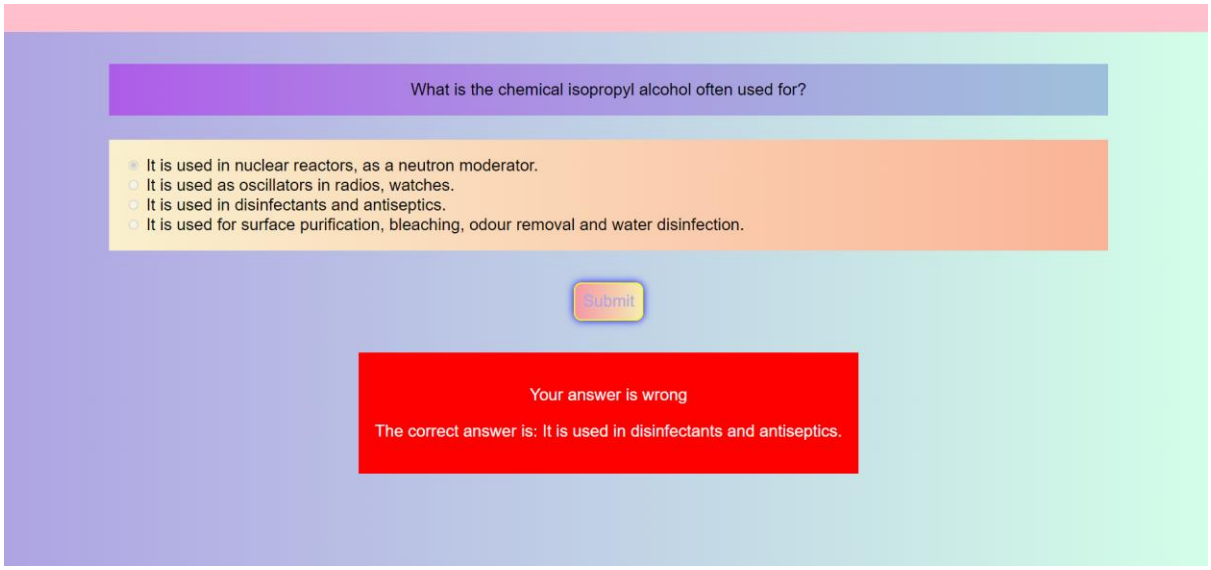


II. After choosing the option press on “Submit” button to verify the answer.

III. If it is correct, message will be displayed in a green box.

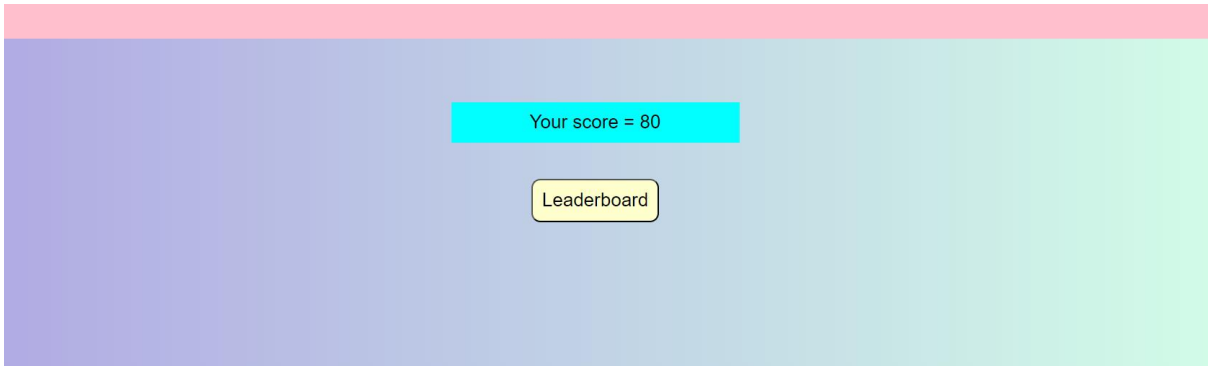


IV. If it is incorrect, the message will be display in a red box along with the correct answer.



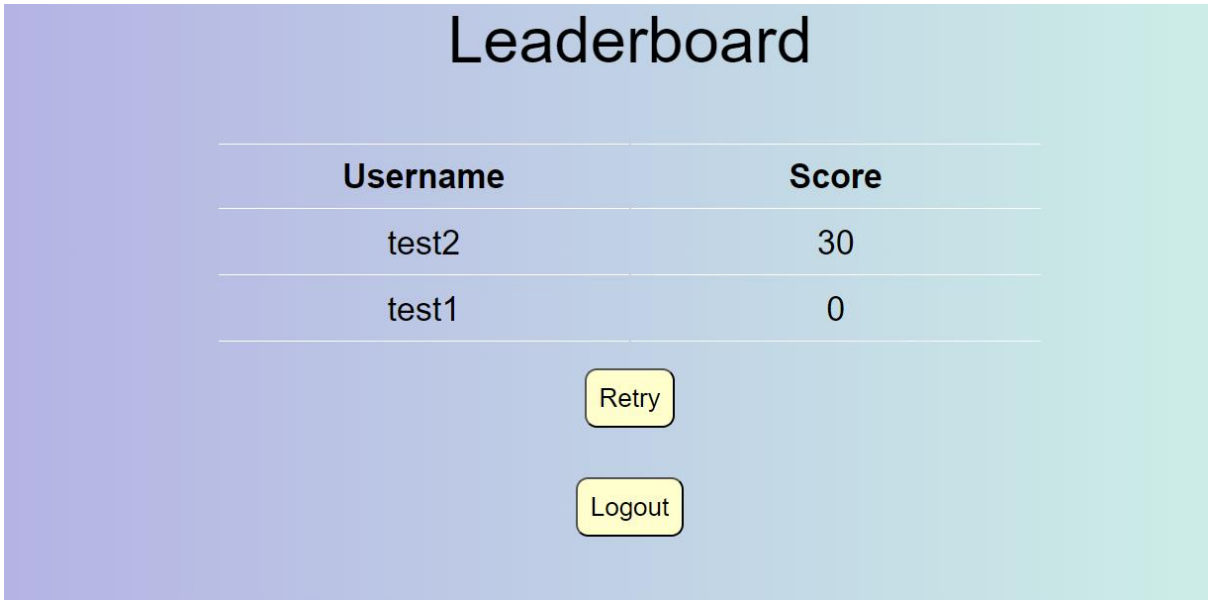
V. Each question carries 10 points which will be credited to the user if the option chosen by him/her is correct. If the question is unanswered or wrong option is chosen, the points won't be credited.

VI. After all the questions are completed, the score obtained by the user will be displayed along with a button that take the user to the leaderboard.



VII. The leaderboard shows the highest score obtained by the users who had used SciQuizzy until then.

VIII. It also contains options to logout of SciQuizzy and to take another quiz.



6. Frequently Asked Questions

1) Should I pay to attempt quizzes?

Ans: No, SciQuizzy is free. Users can attempt as many quizzes as they want for free.

2) Can I change the category of questions?

Ans: No, SciQuizzy is meant for enhancing scientific knowledge by attempting Science quizzes

Future scope and Development

Creating questions for quiz and managing quiz: Users will be able to create custom quizzes by selecting questions from a question bank or creating their own questions. The website will also allow users to browse and take quizzes created by others.

Adding new features: The quiz website can be expanded by adding new features such as a chat room for users to discuss quiz-related topics

Integrating with other platforms: The website can be integrated with other platforms such as social media, e-learning platforms, and job search portals to increase its reach and appeal to a wider audience.

Personalized quizzes: The website can offer personalized quizzes based on user preferences and past performance. This will not only improve the user experience but also provide valuable data for the website's analytics.

Monetization: The website can generate revenue through advertising, sponsorships, or premium subscriptions that offer access to exclusive content or features.

Mobile app: A mobile app version of the website can be developed to cater to the growing number of mobile users. This will increase the accessibility and convenience of the website for users.

Overall, the future scope of a quiz website project is vast, and the website can evolve and adapt to meet the changing needs and expectations of its users. With continuous updates and improvements, the website can remain relevant and engaging for years to come.

Conclusion

The project of a quiz website is a great way to create an engaging and interactive platform for users to improve their knowledge and skills. The website will enable users to create, manage and take quizzes on a variety of topics, with features such as question banks. The future scope of the project is immense, with possibilities such as adding new features, personalized quizzes, integration with other platforms, and monetization.

A quiz website has the potential to be a valuable tool for education, training, and entertainment. It can provide a fun and engaging way for users to learn new things, test their knowledge, and challenge themselves. Overall, the project of a quiz website is a promising venture that can benefit users and the website owner alike.

Key Takeaways

There are several key takeaways from this project that can help improve future projects and team performance:

- Clear goals and expectations: It is important to have clear project goals and expectations communicated to the team. This helped team members to stay focused and aligned towards the same goal.
- Effective communication: Communication is crucial for the success of a team project. Regular check-ins and status updates were held to ensure everyone is on the same page.
- Collaboration and teamwork: Teamwork and collaboration are essential for software projects, as it involves working on complex tasks that require different skills and expertise. All the team members worked together, shared knowledge and provided feedback.
- Agile methodology: Agile methodology helped the team to deliver high-quality software products faster by breaking down projects into smaller, more manageable tasks and prioritizing work based on the needs.
- Continuous integration and testing: Continuous integration and testing helped to ensure that software products are of high quality and meet the requirements. This involved automated testing, frequent code reviews, and continuous integration of code changes.
- Learning and improvement: Team members were encouraged to learn from their experiences and identify areas for improvement.

References

- Software Engineering: A Practitioner's Approach Seventh Edition by Roger S. Pressman
- SOFTWARE ENGINEERING: Ninth Edition by Ian Sommerville
- Lucid chart: Interface design tool
- Black box and white box testing
- Node.js
- JavaScript
- C++

Contribution

Roll.no	Name	Contribution
106121014	Ankita Priya	• Test document • Data flow diagram • Worked on the login system, main quiz section and leaderboard system
106121081	S. Nanthana	• User manual • UML diagrams • Worked on the signup system, main quiz section and leaderboard system
106121117	Sayanti Maiti	• Software requirements specification • UML diagrams • Worked on the login system, signup system and main quiz section
106121135	Suhani Bhardwaj	• Code details document • Entity relationship diagram • Worked on the login system, main quiz section and leaderboard system