

# **Quiz Application**

**A PROJECT REPORT  
for  
Mini Project-I (K24MCA18P)  
Session (2024-25)**

**Submitted by**

**Rishu Agarwal  
202410116100168  
Rahul  
202410116100157  
Riya  
202410116100170**

**Submitted in partial fulfilment of the  
Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATION**

**Under the Supervision of  
Ms. Divya Singhal  
Assistant Professor**



**Submitted to**

**DEPARTMENT OF COMPUTER APPLICATIONS  
KIET Group of Institutions, Ghaziabad  
Uttar Pradesh-201206**

**(DECEMBER- 2024)**

## **CERTIFICATE**

Certified that **Rishu Agarwal 202410116100168, Rahul 202410116100157, Riya 202410116100170** have carried out the project work having “**Quiz Application**” (**Mini Project-I, K24MCA18P**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Ms. Divya Singhal**  
**Assistant Professor**  
**Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**

**Dr. Arun Kr. Tripathi**  
**Dean**  
**Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**

## **Quiz Application**

**Rishu Agarwal**

**Rahul**

**Riya**

## **ABSTRACT**

The Quiz Application project is an interactive, user-friendly platform developed using the MERN stack (MongoDB, Express.js, React.js, Node.js). It is designed to help users evaluate and enhance their knowledge in various technical domains, including HTML, CSS, JavaScript, React, MongoDB, and Redux. The application begins with a landing page showcasing multiple quiz options and a login button. To ensure secure access, users must register and log in before attempting any quiz. The registration process collects essential information such as the user's name, email, and password, which is securely stored in the database. Once logged in, users can select a quiz of their choice and proceed to answer multiple-choice questions. The quiz interface features two navigation buttons, "Submit" and "Skip," allowing users to control their quiz flow. Upon quiz completion, a result page provides detailed feedback, displaying a table of questions alongside the user's answers and correct answers. Additionally, the application calculates the user's score and indicates their pass or fail status. Users are also given the option to attempt more quizzes, seamlessly returning to the homepage. This project focuses on creating a scalable, secure, and engaging platform for learning and self-assessment. It leverages the robustness of the MERN stack to ensure smooth operation, a dynamic user experience, and efficient data management. The inclusion of detailed performance tracking fosters continuous learning, making the application suitable for students and professionals alike. The Quiz Application stands as a practical solution for skill enhancement, bridging the gap between knowledge acquisition and self-evaluation in a modern, technology-driven environment.

# ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Ms. Divya Singhal** for her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Dean, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Rishu Agarwal**

**Rahul**

**Riya**

# TABLE OF CONTENTS

	Certificate	ii
	Abstract	iii
	Acknowledgements	iv
	Table of Contents	v
1	Introduction	6-8
	1.1 Project Discription	6
	1.2 Project Scope	6
	1.3 Project Overview	7-8
2	Feasibility Study	9-10
	1.1 Technical	9
	1.2 Operational	9
	1.3 Behavioral	10
3	Project Objective	11
4	Hardware and Software Requirement	12
5	Project Flow	13-24
6	Project Outcome	25-31
7	References	32

# Chapter 1

## Introduction

### 1.1 Project Description

The project is a **Quiz Application** developed using the **MERN (MongoDB, Express, React, Node.js) stack**. It provides an interactive platform for users to test their knowledge on various topics such as **HTML, CSS, JavaScript, React, MongoDB, and Redux**.

The application starts with a homepage displaying multiple quiz options. Users are required to **register and log in** before accessing any quiz. Once logged in, users can choose a quiz and attempt multiple-choice questions. Each question screen includes a **"Submit" button** to lock in the answer and a **"Skip" button** to move to the next question.

After completing the quiz, users can review their answers in a table format with three columns:

- **Given Question**
- **User's Answer**
- **Correct Answer**

A final result page shows the **user's score** and whether they have passed or failed the quiz. Users can then opt to attempt another quiz by navigating back to the homepage.

This application aims to provide a comprehensive and engaging learning experience for students or professionals looking to strengthen their foundational knowledge.

### 1.2. Project Scope

The quiz application has a broad scope in both educational and practical scenarios:

1. **Educational Tool:** It helps students reinforce their understanding of programming languages and technologies through hands-on testing.
2. **Interview Preparation:** Users preparing for technical interviews can practice with topic-specific quizzes.
3. **Customizable Content:** The application can be expanded to include more topics or specialized quizzes.

4. **Performance Analysis:** Users can evaluate their strengths and weaknesses by reviewing their answers and scores. 6
5. **Scalable Architecture:** Built using the MERN stack, the application can easily handle a growing user base and additional features like timed quizzes or leaderboard systems.

## 1.3 Project Overview

The Quiz Application is a dynamic and interactive platform designed to facilitate knowledge testing and learning in various technical domains. Developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), the application provides a secure, user-friendly environment for users to evaluate their skills in technologies like HTML, CSS, JavaScript, React, MongoDB, and Redux.

### *Key Features*

1. **Homepage Interface:**
  - Displays multiple quiz options based on different technical topics.
  - Includes a login button for secure user authentication.
2. **User Authentication:**
  - Ensures that only registered users can access the quizzes.
  - Redirects unregistered users attempting to access quizzes to the registration page.
  - Collects user details such as name, email, and password for secure storage in the database.
3. **Quiz Flow:**
  - After logging in, users can select a quiz of their choice.
  - The quiz interface displays multiple-choice questions with "Submit" and "Skip" buttons, enabling smooth navigation.
4. **Result Generation:**
  - Upon completion of a quiz, users are presented with a detailed result page that includes:
    - A table showing each question, the user's selected answers, and the correct answers.
    - A calculated score and an indication of whether the user passed or failed the quiz.
5. **Continuous Engagement:**
  - Offers users the ability to return to the homepage and attempt more quizzes.

## Technology Stack

- **MongoDB:** Used to store user data, quiz questions, answers, and results.
- **Express.js:** Provides a backend framework to manage APIs and handle server-side logic.
- **React.js:** Enables a responsive and intuitive user interface.
- **Node.js:** Acts as the server runtime for processing requests and delivering data efficiently.

## Objectives

- To provide a platform for users to test and improve their knowledge in technical fields.
- To ensure secure user authentication and data management.
- To create a seamless and engaging user experience with detailed feedback on performance.





## Chapter-2

### Feasibility Study

This section examines whether the technology and tools used in the project are adequate for its successful implementation.

#### 2.1 Technical Feasibility

- **Technology Stack:** The project is built using the MERN stack, a proven and robust technology for developing full-stack web applications.
- **Scalability:** MongoDB allows for easy storage and retrieval of data, ensuring that the application can scale as more users and quizzes are added.
- **Cross-Platform Accessibility:** React.js ensures a responsive design, making the application accessible on desktops, tablets, and mobile devices.
- **Security:** The use of JWT for authentication ensures secure user data and prevents unauthorized access to quizzes and results.
- **Performance:** Node.js and Express.js provide a fast and efficient backend framework to handle concurrent requests seamlessly.

Overall, the tools and technologies used make the project technically feasible.

#### 2.2 Operational Feasibility

Operational feasibility evaluates whether the application meets user needs and is practical to operate.

- **User Registration and Login:** The system ensures that only authenticated users can access quizzes, which is both practical and secure.
- **Ease of Use:** The intuitive interface allows users to navigate through the application with minimal effort. The skip and submit options provide flexibility in answering questions.
- **Feedback Mechanism:** The result page provides detailed feedback, helping users learn from their mistakes and track progress.
- **Support for Multiple Quizzes:** Users can choose topics of their interest, catering to diverse learning needs

## 2.3 Behavioral Feasibility

Behavioral feasibility assesses the willingness of users to adapt to and use the application.

- **User Engagement:** The application's interactive design, with multiple-choice questions and instant feedback, promotes active participation.
- **Motivation:** The pass/fail results and the ability to track performance encourage users to take more quizzes and improve their knowledge.
- **Learning Outcomes:** Users will find value in the application as it helps them enhance their skills in technologies critical to modern development practices.
- **Adaptability:** The registration and login process is simple and standard, reducing the learning curve for new users.

# Chapter-3

## Project Objective

The quiz application project aims to provide an engaging and educational platform for users to test and improve their technical knowledge. This objective is achieved by implementing key features that cater to user convenience, learning, and performance tracking.

Specific objectives include:

**1. Knowledge Evaluation:**

- Enable users to assess their understanding of core technologies like HTML, CSS, JavaScript, React, MongoDB, and Redux.

**2. Interactive Learning Platform:**

- Incorporate multiple-choice quizzes with features such as skip and submit buttons for a smooth user experience.
- Provide instant feedback by displaying correct and incorrect answers after quiz completion.

**3. Secure User Management:**

- Implement robust authentication using JWT to ensure data security and a personalized experience for users.

**4. User-Friendly Interface:**

- Design an intuitive interface that allows users to navigate through quizzes and view results effortlessly.

**5. Detailed Performance Tracking:**

- Present a comprehensive results table showing the user's answers, correct answers, and final score, offering insights into their performance.

**6. Continuous Engagement:**

- Encourage users to attempt more quizzes by providing seamless navigation back to the homepage.

**7. Scalable Architecture:**

- Build the application using the MERN stack to accommodate future enhancements, additional quiz topics, and a growing user base.

**8. Real-World Application:**

- Support students and professionals in their learning journey, whether for academic purposes, skill improvement, or interview preparation.

# Chapter-4

## Hardware and Software Requirement

### Hardware Requirements

- **Processor:** Intel i3 or higher
- **RAM:** Minimum 4GB (8GB recommended for smooth performance)
- **Storage:** At least 500MB of available space for local database and application files
- **Internet Connection:** Required for accessing the application (if hosted on a server)

### Software Requirements

- **Frontend:** React.js
  - Used for building the user interface and dynamic components.
- **Backend:** Node.js and Express.js
  - Handles server-side operations and API endpoints.
- **Database:** MongoDB
  - Stores user information, quiz questions, and results.
- **Authentication:** JWT (JSON Web Tokens)
  - Ensures secure login and session management.
- **Development Tools:**
  - **Code Editor:** Visual Studio Code
  - **Version Control:** Git and GitHub
  - **Postman:** For testing APIs
- **Hosting (Optional):**
  - **Frontend:** Deployed on platforms like Vercel or Netlify
  - **Backend:** Hosted on services like Heroku or AWS

# Chapter-5

## Project Flow

The Quiz Application follows a well-structured flow to ensure a seamless user experience. Below is the step-by-step project flow:

### 1. Landing Page (Homepage)

- Displays the interface with multiple quiz options:
  - HTML, CSS, JavaScript, React, MongoDB, Redux.
- Includes a **Login** button for secure user access.
- If a user clicks on a quiz without logging in, the system automatically redirects to the **Registration Page**.

### 2. User Authentication

#### a. Registration

- Users provide the following details:
  - Name
  - Email
  - Password
- The system validates the input and stores the data securely in the database.

#### b. Login

- Registered users enter their email and password.
- The system verifies the credentials and grants access.
- If credentials are invalid, an error message is displayed.

### 3. Quiz Selection

- After logging in, users return to the homepage.
- Users select any quiz topic from the available options.

## 4. Quiz Attempt

### a. Quiz Interface

- The quiz begins with multiple-choice questions displayed one at a time.
- Two navigation buttons are available:
  - **Submit**: Saves the selected answer and moves to the next question.
  - **Skip**: Skips the current question without answering and moves to the next question.

### b. Question Progression

- Users can navigate through all the questions until the quiz is completed.

## 5.Result Page

### a. Detailed Feedback

- After completing the quiz, the system displays a **Result Table** containing:
  - Given Question
  - User's Selected Answer
  - Correct Answer

### b. Score Calculation

- The system calculates the user's total score based on correct answers.
- Displays whether the user has **passed** or **failed** the quiz.

## 6. Post-Result Actions

### a. Attempt More Quizzes

- Users can click the **Attempt More Quizzes** button, which redirects them to the homepage to select another quiz.

## **b. Logout Option**

- Users can log out to end their session securely.

## **Technical Flow (Backend)**

### **1. Authentication Flow:**

- Registration and login data are processed through secure API endpoints.
- Passwords are encrypted using hashing algorithms for security.

### **2. Quiz Data Management:**

- Quiz questions and answers are fetched dynamically from the database.
- User responses are stored for result processing.

### **3. Result Processing:**

- User responses are compared against correct answers stored in the database.
- A pass/fail status is determined based on predefined scoring criteria.



## Flow Chart Diagram :

**Flowchart** is a diagrammatic representation of sequence of logical steps of a program. Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.

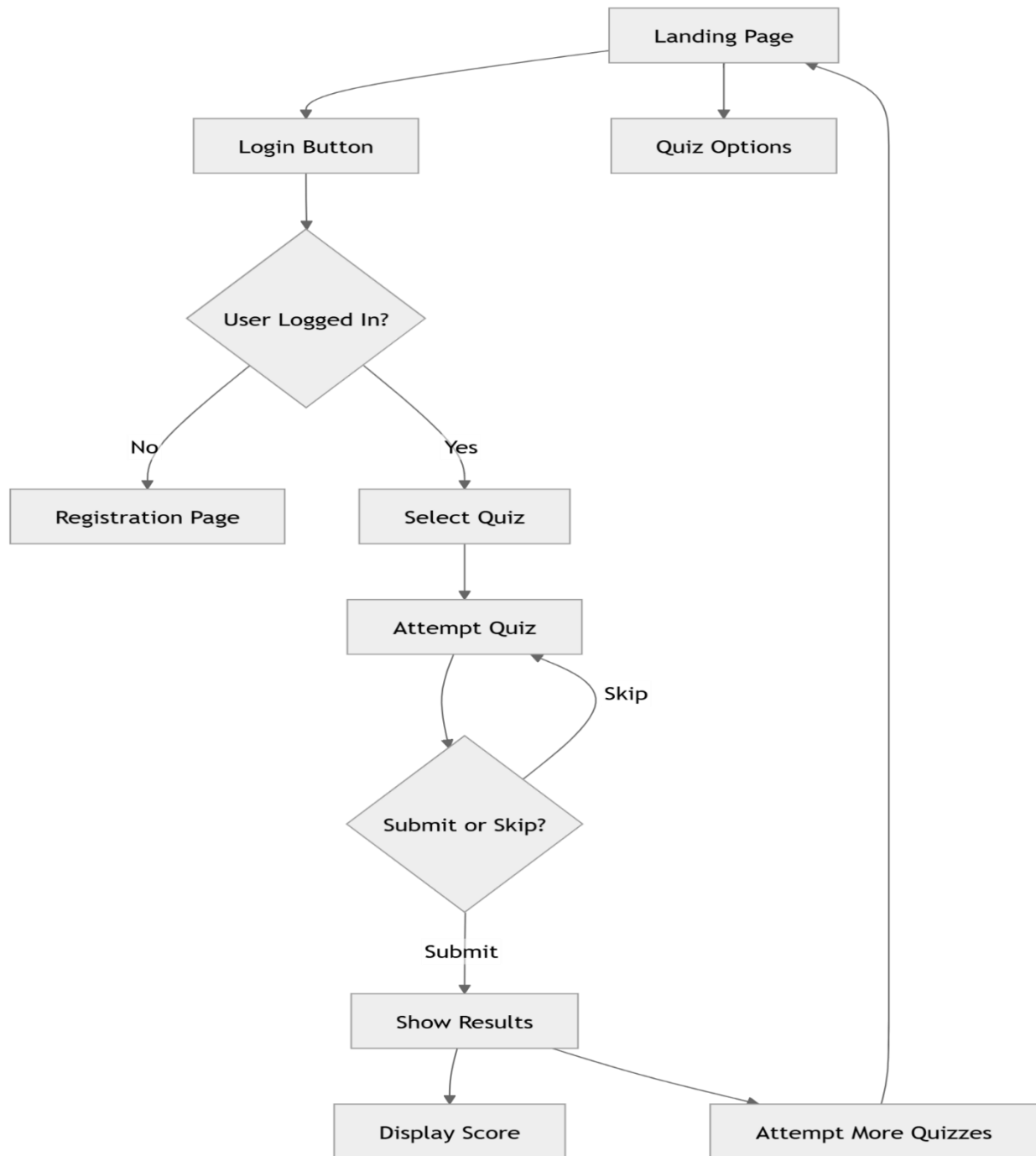


Fig.No.5.1

# 1. Landing Page

- **Description:** The user starts at the homepage, which displays the available quiz options (e.g., HTML, CSS, JavaScript, etc.) and a login button.

# 2. Login Button

- **Description:** If the user clicks the login button, the system checks whether the user is already logged in or not.

# 3. User Logged In?

- **Decision Point:** This diamond-shaped decision node determines if the user is logged in or not:
  - **No:** If the user is not logged in, they are redirected to the **Registration Page**.
  - **Yes:** If the user is logged in, they can proceed to select a quiz.

# 4. Registration Page

- **Description:**
  - New users must register by providing their details (name, email, and password).
  - Once registered, users can log in to access quizzes.

# 5. Select Quiz

- **Description:** After logging in, users return to the homepage, where they can select any quiz topic from the available options.

# 6. Attempt Quiz

- **Description:** The quiz starts, presenting one multiple-choice question at a time.

# 7. Submit or Skip?

- **Decision Point:** The user decides how to proceed for each question:
  - **Skip:** The user skips the current question and moves to the next one.

- **Submit:** The user selects an answer, submits it, and moves to the next question.

## 8. Show Results

- **Description:** After the user completes all the questions, the system displays a detailed results table.
  - The table includes the given questions, the user's answers, and the correct answers.

## 9. Display Score

- **Description:** The system calculates the user's score and shows whether they passed or failed.

## 10. Attempt More Quizzes

- **Description:**
  - The user is given the option to attempt more quizzes by returning to the homepage.
  - If the user clicks this option, the system loops back to the landing page, completing the cycle.

## Entity Relationship Diagram :

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modelling, the database structure is portrayed as a diagram called an entity-relationship diagram.
- 

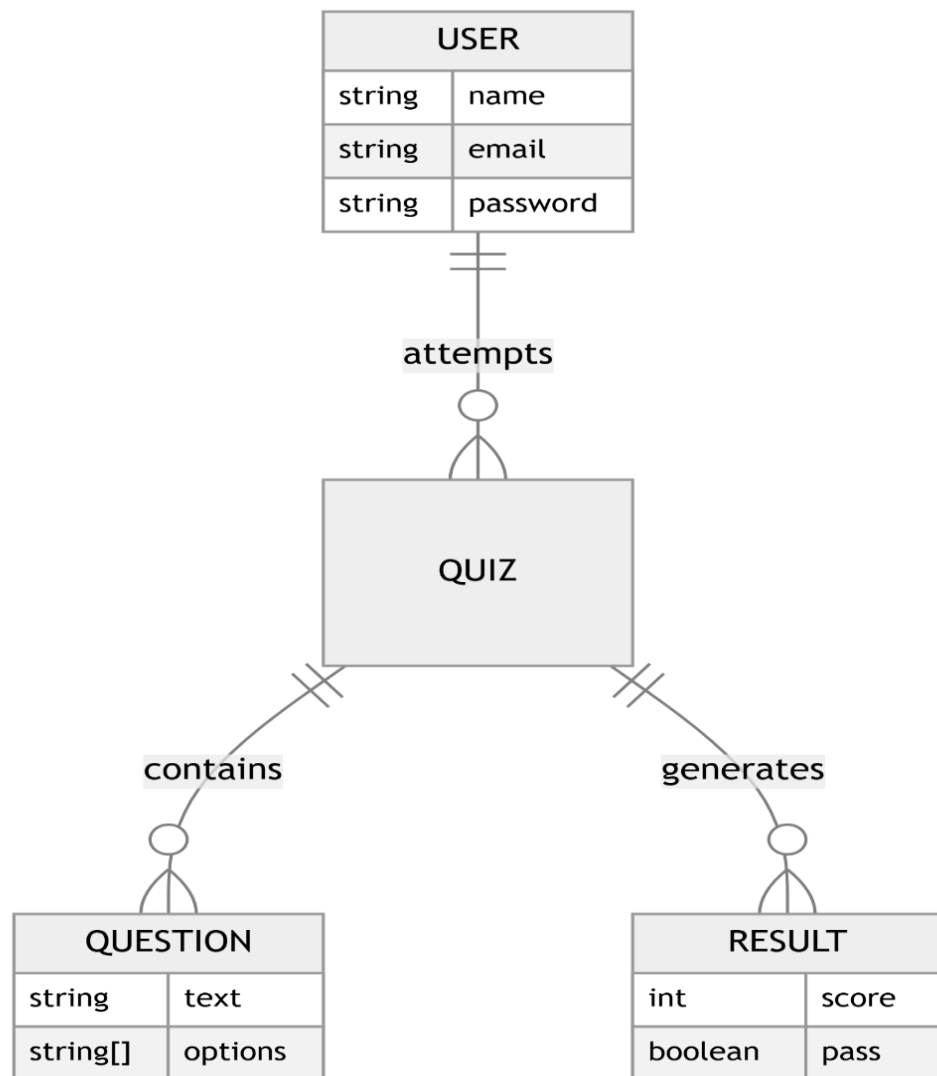


Fig.No.5.2

# 1. USER Entity

- **Attributes:**
  - name (string): The name of the user.
  - email (string): The user's unique email address.
  - password (string): The user's password for authentication.
- **Relationship:**
  - The **USER** entity has a "one-to-many" relationship with the **QUIZ** entity via the attempts relationship. This indicates that a user can attempt multiple quizzes.

# 2. QUIZ Entity

- **Description:** Represents the quiz that users attempt.
- **Relationships:**
  - **Contains Questions:**
    - A quiz includes multiple questions, forming a "one-to-many" relationship between the **QUIZ** and **QUESTION** entities.
  - **Generates Results:**
    - A quiz generates results for the user, forming a "one-to-one" or "one-to-many" relationship between the **QUIZ** and **RESULT** entities.

# 3. QUESTION Entity

- **Attributes:**
  - text (string): The text of the question.
  - options (string[]): A list of answer options for the question.
- **Relationship:**
  - Each question is part of a quiz, forming a "many-to-one" relationship with the **QUIZ** entity.

# 4. RESULT Entity

- **Attributes:**
  - score (int): The score the user achieved in the quiz.
  - pass (boolean): A value indicating whether the user passed or failed the quiz.
- **Relationship:**

- Each result corresponds to a quiz attempt, forming a "one-to-one" or "one-to-many" relationship with the **QUIZ** entity.

## Relationship Explanation

- **USER Attempts QUIZ:**
  - A user can attempt multiple quizzes, but each quiz attempt is tied to a specific user.
- **QUIZ Contains QUESTIONS:**
  - Each quiz comprises multiple questions.
- **QUIZ Generates RESULT:**
  - After a quiz is completed, a result is generated for that specific attempt.

## Flow Representation

This diagram reflects the logical flow of data in the quiz application:

1. **Users** register and log in.
2. They select a **quiz** to attempt.
3. The quiz contains multiple **questions** for the user to answer.
4. Upon completion, the system generates a **result** displaying the score and pass/fail status.

## 3.5 Sequence Diagram

Purpose of a Sequence Diagram

To model high-level interaction among active objects within a system.

To model interaction among objects inside a collaboration realizing a use case.

It either models' generic interactions or some certain instances of interaction.

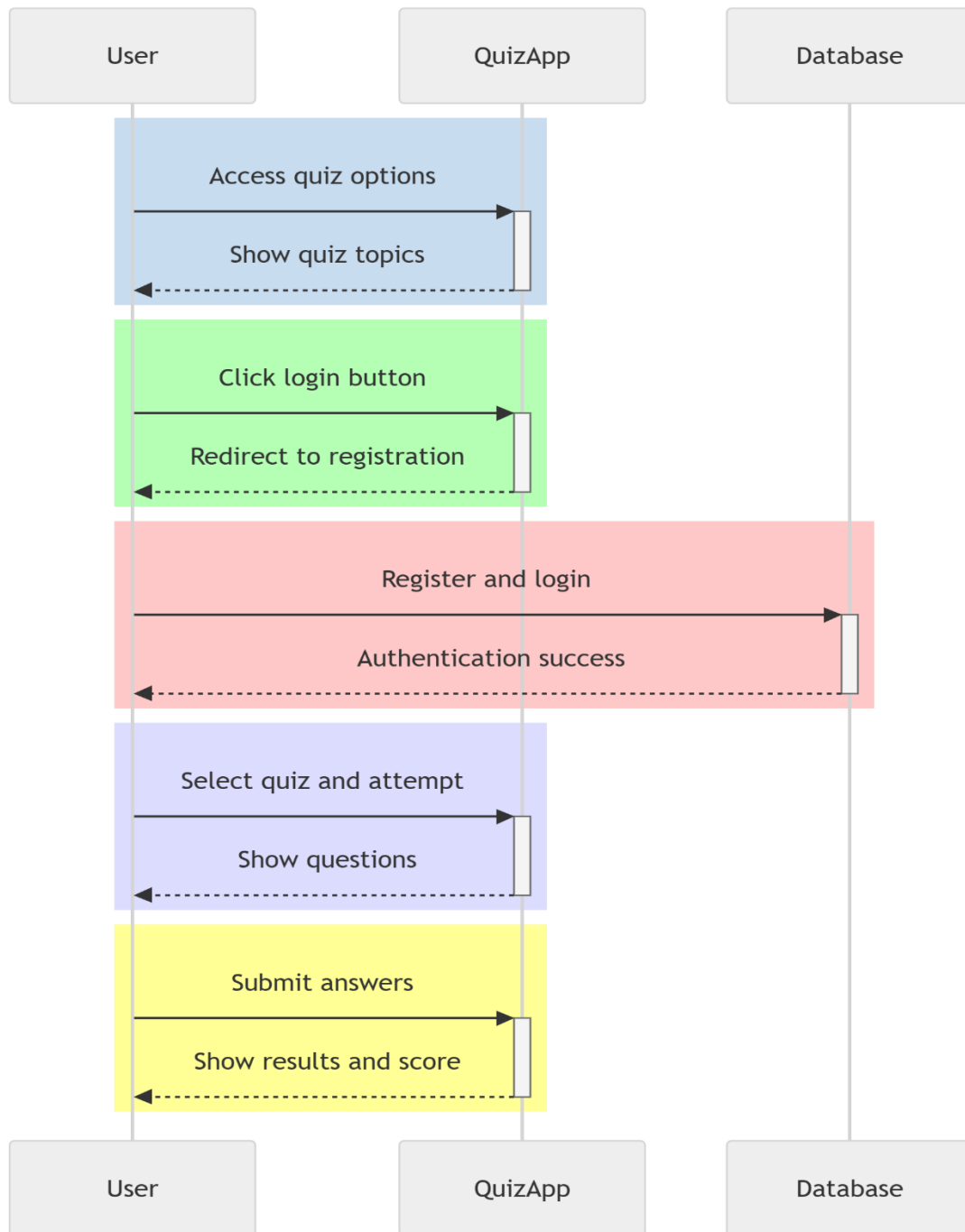


Fig.No.5.3

## 1. Access Quiz Options (Blue Section)

- **User Action:**
  - The user accesses the quiz application and sees available quiz topics.
- **QuizApp Response:**
  - The application fetches and displays the list of available quizzes to the user.

## 2. Login or Registration (Green Section)

- **User Action:**
  - The user clicks on the "Login" button.
- **QuizApp Response:**
  - If the user is not already registered, the application redirects them to the registration page.
- **Database Interaction:**
  - User credentials (name, email, password) are submitted to the database.

## 3. Authentication (Pink Section)

- **User Action:**
  - The user completes the registration and logs in with valid credentials.
- **QuizApp Response:**
  - The application checks the credentials against the database for authentication.
- **Database Interaction:**
  - Authentication is validated, and upon success, the user is allowed access to quiz functionalities.

## 4. Select and Attempt Quiz (Purple Section)

- **User Action:**
  - The user selects a quiz to attempt and begins answering questions.

## 5. Submit Answers and View Results (Yellow Section)

- **User Action:**
  - The user submits their answers at the end of the quiz.
- **QuizApp Response:**
  - The application calculates the user's score and determines whether they passed or failed.
- **Database Interaction:**
  - The submitted answers and the resulting score are saved in the database.
- **Final Output:**
  - The user views their results and the system displays the score along with the option to attempt more quizzes.



## Overall Flow

1. **User Journey:** The user starts from the landing page, logs in, and interacts with the quiz system.
2. **QuizApp Role:** Acts as the middle layer, facilitating communication between the user and the database.
3. **Database Role:** Stores and retrieves data for user registration, authentication, quiz content, and results.

# Chapter-6

## Project Outcome

The quiz application delivers several key outcomes, showcasing its effectiveness and usability in enhancing users' knowledge and providing a seamless experience. Below are the major outcomes of the project:

### 1. Interactive Learning Platform:

The application offers an engaging platform for users to test their knowledge in technologies like HTML, CSS, JavaScript, React, MongoDB, and Redux through interactive quizzes.

### 2. Streamlined User Experience:

- The project successfully integrates **user registration and login functionality**, ensuring that each user's progress and data are securely managed.
- The intuitive interface allows users to easily navigate through various quiz options, attempt questions, and view results.

### 3. Dynamic Quiz System:

- Users can answer multiple-choice questions with options to skip or submit, enhancing flexibility during the quiz attempt.
- Immediate feedback through result tables provides clarity on the user's performance, displaying the given question, user's answer, and the correct answer.

### 4. Performance Tracking:

- A dedicated results page summarizes the user's quiz performance, highlighting their **score** and **pass/fail status**, which motivates them to improve and try again.
- The ability to revisit the homepage and attempt more quizzes encourages continuous learning and practice.

### 5. Efficient Data Handling:

- The backend effectively stores and manages user data (name, email, password) and quiz responses in a **MongoDB database**, ensuring data integrity and security.

### 6. Comprehensive Technology Demonstration:

- This project demonstrates the efficient use of the **MERN (MongoDB, Express.js, React.js, Node.js)** stack, showcasing skills in both frontend and backend development.

- It highlights proficiency in creating dynamic web applications with secure authentication mechanisms.

## 7. Scalability and Reusability:

- The modular design of the application allows for easy addition of new quiz topics or features in the future, making it scalable.
- The architecture can be reused for other similar projects, saving development time and effort.

## 8. Real-World Application:

- The project addresses a common real-world need for interactive learning tools, making it a valuable resource for students and professionals looking to strengthen their technical skills.
- It sets a foundation for potential enhancements, such as time-bound quizzes, leaderboard systems, or personalized question recommendations.

## Front Page :

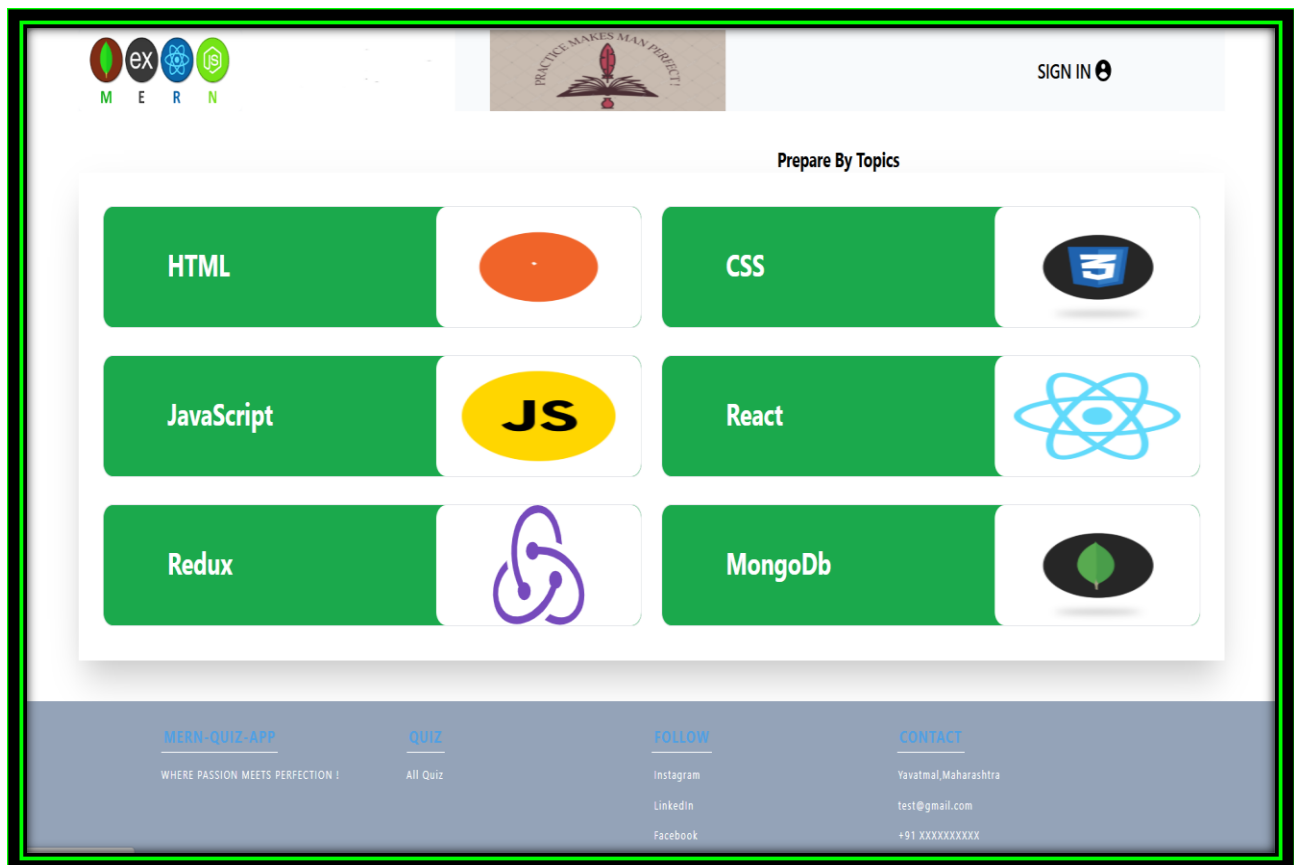


Fig.No.6.1

Welcome to the Quiz Application! Test your knowledge with quizzes on **HTML**, **CSS**, **JavaScript**, **React**, **MongoDB**, and **Redux**.

To get started, **register or log in**, and choose a quiz of your interest. Answer multiple-choice questions with the option to skip or submit.

View your detailed results, including your score and pass/fail status. Ready to sharpen your skills? **Click Login to Begin!**

## Registration Page :

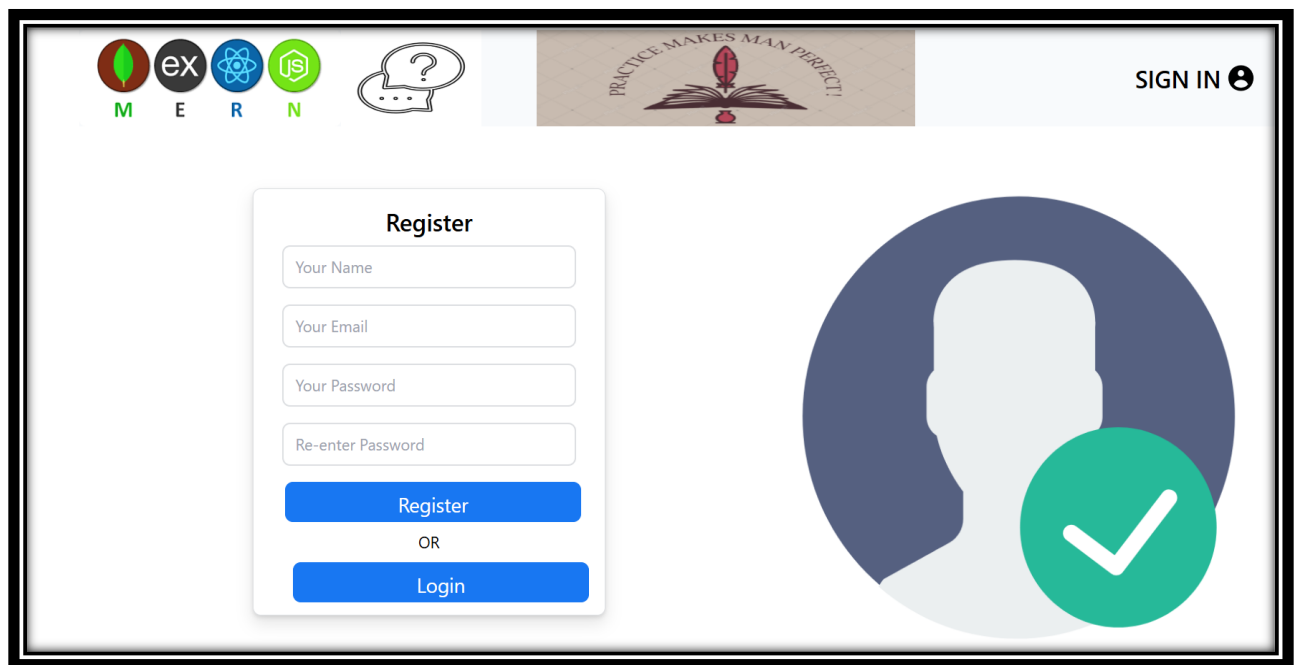
A screenshot of a web application's registration page. The page has a light gray background. At the top, there is a navigation bar with several icons: a green leaf, a black circle with 'ex', a blue atom, a green square with 'JS', and a speech bubble with a question mark. To the right of these icons is a banner with the text 'PRACTICE MAKES MAN PERFECT!' and an illustration of an open book with a red quill. Further right is a 'SIGN IN' button with a user icon. The main content area features a 'Register' form on the left and a large circular graphic on the right. The form has four input fields: 'Your Name', 'Your Email', 'Your Password', and 'Re-enter Password'. Below these fields are two blue buttons: 'Register' and 'Login', separated by the text 'OR'. The circular graphic on the right shows a gray silhouette of a person's head and shoulders, with a large green checkmark overlaid on it.

Fig.No.6.2

Welcome to the **Registration Page!** Please provide your **Name**, **Email**, and **Password** to create an account.

Registration is quick and secure, and your data will be safely stored in our database.

Once registered, you can log in and access quizzes of your choice.

## Login Page :

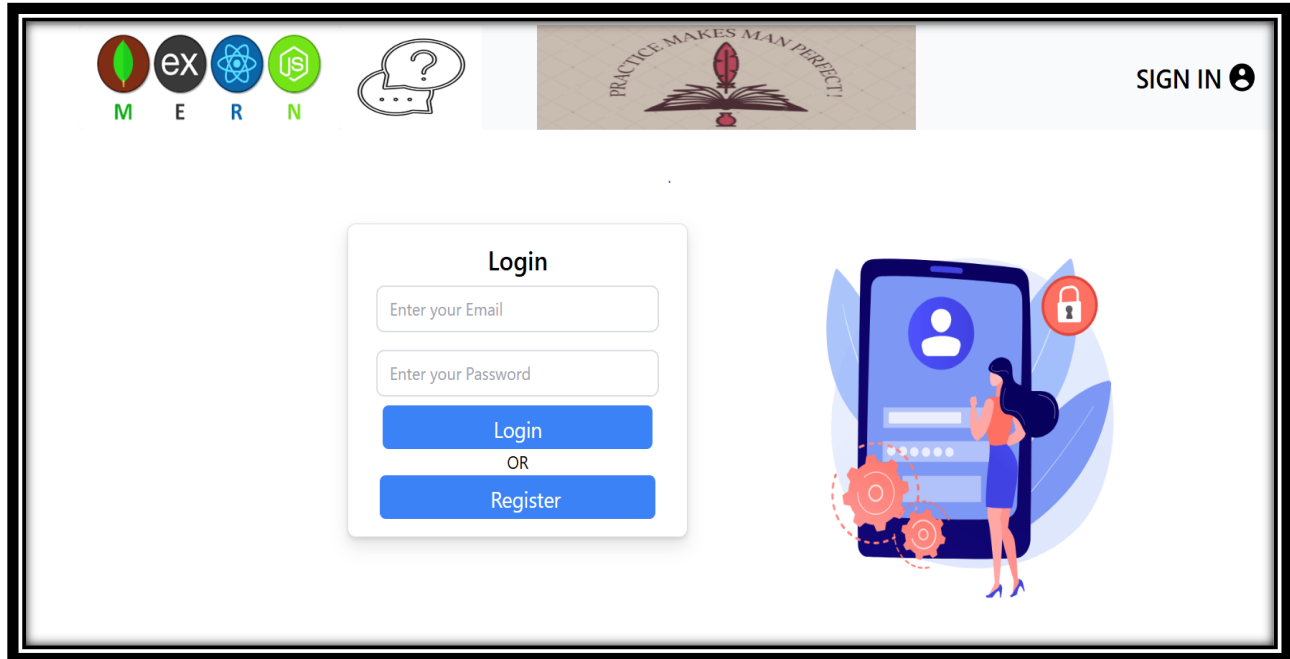


Fig.No-6.3

Welcome back to the **Login Page!** Please enter your **Email** and **Password** to access your account. If you don't have an account yet, click **Register** to create one. Once logged in, you'll be able to attempt quizzes on various topics like **HTML**, **CSS**, **JavaScript**, and more.

## Attempt Quiz

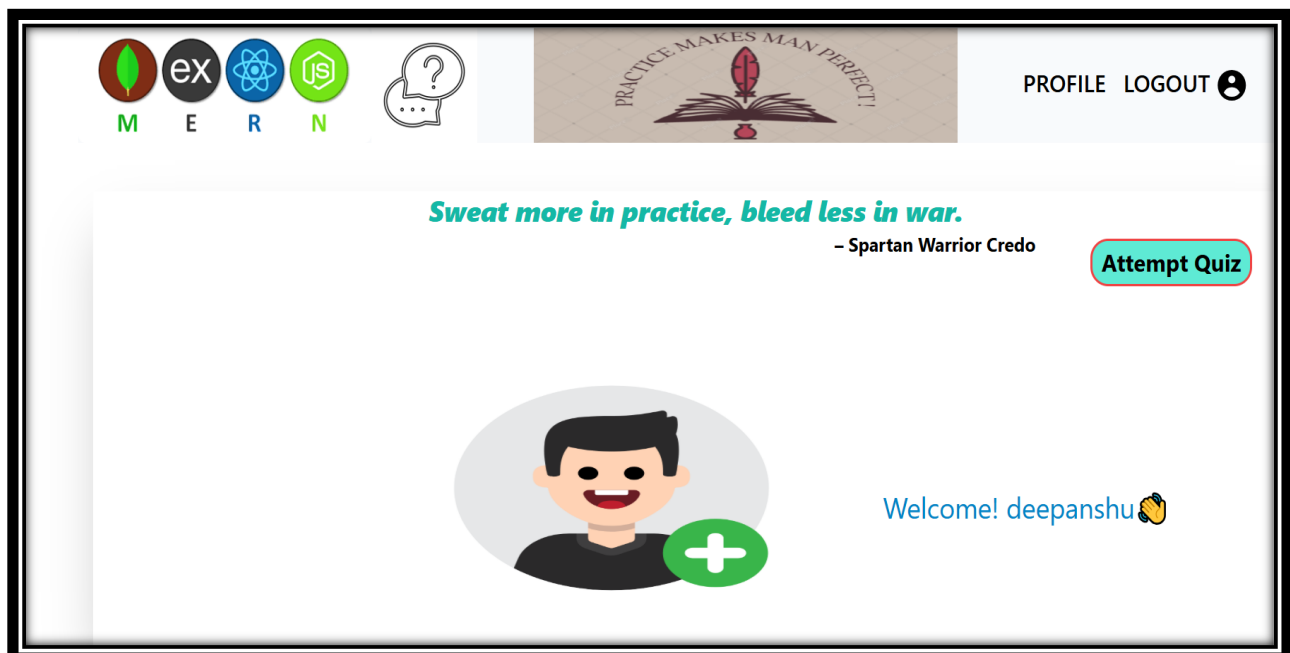


Fig.No.6.4

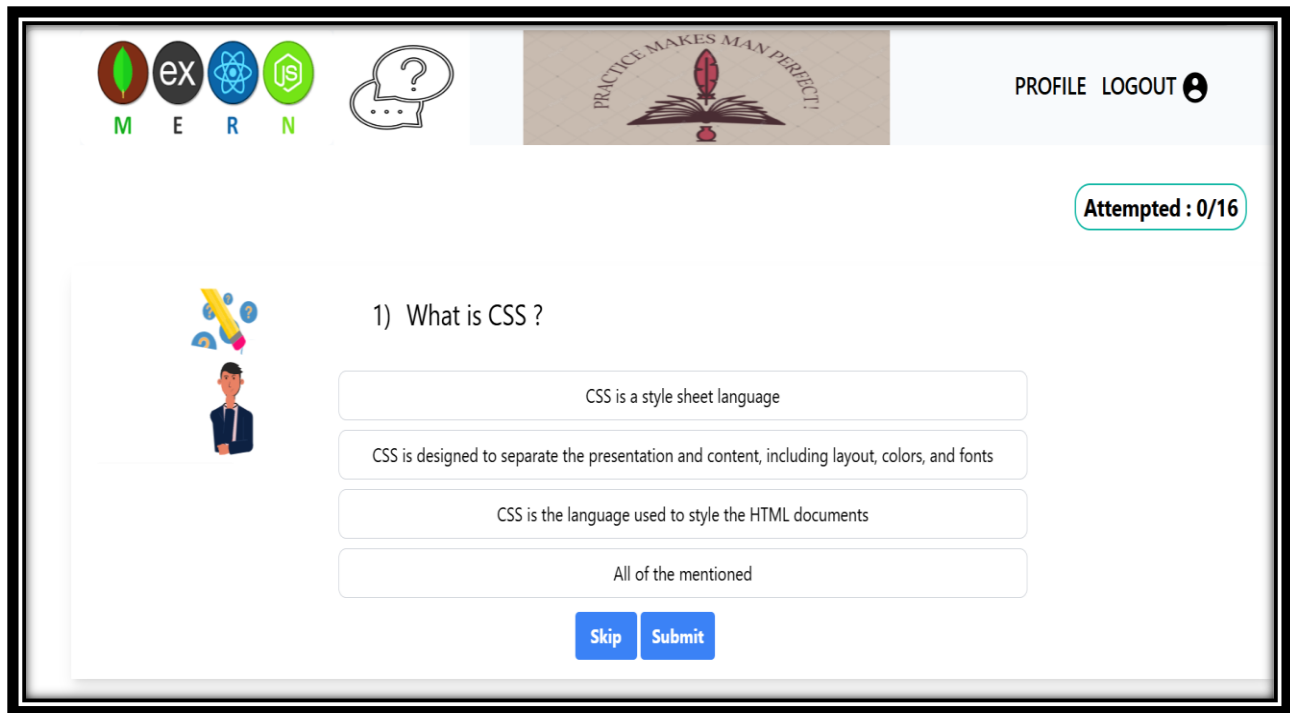


Fig.No.6.5

Welcome to the **Attempted Quiz Page!** You've successfully logged in and are ready to test your knowledge.

Here, you can choose from a variety of quiz topics like **HTML**, **CSS**, **JavaScript**, **React**, **MongoDB**, and **Redux**.

Once you select a quiz, you'll be presented with multiple-choice questions. You can choose your answers and move to the next question.

If you're unsure about a question, you can click the **Skip** button to move forward without answering.

At any point, you can **Submit** your answers to finish the quiz.

After completing the quiz, you'll see a detailed results page, showing the questions, your answers, and the correct answers.

You'll also receive a **final score** to help you track your progress.

Based on your score, you'll know whether you've **passed** or **failed** the quiz.

If you're ready for more, click **Attempt More Quizzes** to return to the quiz options.

Keep testing your skills and improving with each quiz!

   <span>PROFILE LOGOUT </span>		
QUESTIONS	USER ANSWER	CORRECT ANSWER
1) What is CSS ?	CSS is a style sheet language	All of the mentioned
2) Which of the following tag is used to embed css in html page?	<style>	<style>
3) Which of the following CSS selectors are used to specify a group of elements?	CSS	class
4) Which of the following has introduced text, list, box, margin, border, color, and background properties?	larawell	CSS
5) Which of the following CSS framework is used to create a responsive design?	class	bootstrap
6) Which of the following CSS selector is used to specify a rule to bind a particular unique element?	font-weight: bold	id
7) Which of the following CSS property is used to make the text bold?	font-style	font-weight: bold
8) Which of the following CSS style property is used to specify an italic text?	-o-	font-style

Fig.No.6.6

9) Which of the following are the CSS Extension Prefixes for Webkit?	CSS level 2	-webkit
10) Which of the following is the first CSS specification to become an official W3C Recommendation?	grayscale()	CSS level 1
11) Which of the following function defines a linear gradient as a CSS image?	grayscale()	linear-gradient()
12) Which of the following CSS property can be used to set the image as a border instead of the border style?	image()	border-image-source
13) Which of the following CSS property defines the different properties of all four sides of an element's border in a single declaration?	border-image-source	border-width
14) Which of the following CSS property sets the font size of text?	padding	font-size
15) Which of the following is not the property of the CSS box model?	size	color
16) Which of the following CSS property sets what kind of line decorations are added to an element, such as underlines, overlines, etc?	margin	text-decoration-line
	text-style	
Final Marks		

Fig.No.6.7

On this page, you can review the results of your quiz attempt.

A table will display each **question** you answered, alongside your **selected answer** and the **correct answer**.

This allows you to compare your choices with the correct responses and learn from any mistakes.

You can review your performance and gain insights into areas where you need improvement.

## Result Page :



Fig.No.6.8

The **Result page** shows your final score after completing the quiz.

It will indicate whether you **passed** or **failed** based on your performance.

You can view your **overall score** and compare it with the passing criteria.

The page also provides a **detailed breakdown** of correct and incorrect answers for further analysis.

Finally, you can click on the "**Attempt More Quizzes**" button to return to the quiz options page.



## References

1. **MongoDB Official Documentation**  
<https://www.mongodb.com/docs/>
2. **ReactJS Documentation** – Official React documentation to guide you through building user interfaces.  
<https://reactjs.org/docs/getting-started.html/>
3. **Node.js Documentation**  
<https://nodejs.org/en/docs/>
4. **Express.js Documentation**  
<https://expressjs.com/en/starter/installing.html/>
5. **JWT Authentication in Node.js**  
<https://www.digitalocean.com/community/tutorials/>
6. **Building a Full-Stack Quiz Application**  
<https://www.section.io/engineering-education/mern-stack-tutorial/>
7. **Understanding the MongoDB Aggregation Framework**  
<https://www.mongodb.com/docs/manual/aggregation/>
8. **React Router for Navigation**  
<https://reactrouter.com/>
9. **MERN Stack Authentication Tutorial**  
<https://www.tutorialsbuddy.com/mern-stack-authentication-tutorial/>
10. **Using Redux in React**  
<https://redux.js.org/introduction/getting-started/>
11. **Creating a Responsive UI with Material-UI**  
<https://mui.com/getting-started/usage/>
12. **MongoDB User Authentication and Security**  
<https://www.mongodb.com/solutions/atlas-security>
13. **Building a Quiz Application with React and Node.js**  
<https://www.freecodecamp.org/news/how-to-build-a-quiz-app-with-react-and-node/>

