A Field Project Report on

QUIZTOPIA

Submitted

In partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE and ENGINEERING

By

D.Lakshmi	231FA04746
T.Bharath	231FA04770
N.Virat	231FA04773
N.Keerthi	231FA04825

Under the Guidance of
Mr.T.Narasimha Rao
Assistant Professor, CSE

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF COMPUTING AND INFORMATICS

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University)
Vadlamudi, Guntur -522213, INDIA.

April, 2025

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the field project entitled *"QUIZTOPIA"* is being submitted by **D.LAKSHMI[231FA04746],T.BHARATH[231FA04770],N.VIRAT[231FA04773]** and**N.KEERTHI[231FA04825]** in partial fulfilment of the requirements for the degree of Bachelor of Technology (B.Tech.) in Computer Science and Engineering at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur District, Andhra Pradesh, India.

This is a bonafide work carried out by the aforementioned students under my guidance and supervision.

[illegible]

Guide

[illegible]

Project Review Committee

[illegible]

HoD, CSE

HoD
Dept. of Computer Science & Engineering
VFSTR Deemed to be University
VADLAMUDI - 522 213
Guntur Dist., A.P., India.

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DECLARATION

Date:19-04-2025

We hereby declare that the work presented in the field project titled "Quiztopia" is the result of our own efforts and investigations.

This project is being submitted under the supervision of **Mr.T.Narasimha Rao, Assistant Professor, CSE** in partial fulfillment of the requirements for the Bachelor of Technology (B.Tech.) degree in Computer Science and Engineering at Vignan's Foundation for Science, Technology and Research (Deemed to be University), Vadlamudi, Guntur, Andhra Pradesh, India.

D.Lakshmi	(231FA04746) D.Lakshmi
T.Bharath	(231FA04770) [illegible]
N.Virat	(231FA04773) N.Virat
N.Keerthi	(231FA04825) N. Keerthi

CONTENTS

1. INTRODUCTION

Our project is a fully functional Interactive Online Quiz Application developed using HTML, CSS, and JavaScript. Designed with both usability and performance in mind, this web-based system provides an engaging platform for users to test their knowledge across a range of topics. The application begins with a secure login page, allowing basic user authentication before accessing the quiz. Once logged in, users are taken to a dynamic quiz interface featuring multiple-choice questions that are presented one at a time. The system incorporates a countdown timer, ensuring that each question must be answered within a specified time limit, adding an element of challenge and excitement. As users progress through the quiz, the application dynamically loads questions, tracks selected answers, and updates the score in real time. Upon completion, users are presented with their final score and performance feedback, offering an immediate sense of accomplishment and areas for improvement.

1.1 Problem Definition:

User Authentication : Some systems have login/signup; others allow guest access.
Quiz Start : Often starts directly from the homepage or embedded in the same page.
Question Randomization : Some quizzes have fixed questions.
Leaderboard & Assessments: Only advanced quiz platforms offer a leaderboard.

1.2 Existing System:

1.User Authentication:

The application includes a secure login system to ensure that only authorized or registered users can access the quiz. This helps maintain user data integrity and enables personalized quiz sessions. Future upgrades could include role-based access for admins and regular users.

2.Dynamic Question Loading:

Questions are dynamically loaded using JavaScript, providing a smooth and interactive quiz experience. This eliminates the need for page reloads and keeps the user engaged throughout the quiz by presenting one question at a time or in a shuffled order.

3.Automatic Timer

A built-in countdown timer is assigned to each quiz or individual question, promoting time efficiency. It challenges users to think and respond quickly while also preventing unfair advantages like prolonged thinking or internet searching.

4.Answer Tracking

Users can navigate between questions freely and change their answers before final submission. This simulates real-world exam scenarios and improves user control and flexibility during the quiz.

5.Instant Score Calculation

After submission, the quiz instantly evaluates user responses and displays the final score. This immediate feedback loop helps users understand their performance and identify areas for improvement.

6.Responsive Design

Built with responsive web design principles, the application works seamlessly across a wide range of devices, including desktops, tablets, and smartphones. This ensures accessibility and usability regardless of screen size or platform.

7.Interactive UI

The user interface is designed to be clean, intuitive, and interactive. Smooth transitions, hover effects, visual cues, and progress indicators enhance the overall user experience and make navigation effortless.

8.Admin Panel

The application includes an admin panel that allows authorized administrators to add, edit, or delete quiz questions. This feature makes the system highly flexible and easy to maintain, enabling quick updates to the question bank and supporting dynamic content management.

1.3 Proposed System:

User Authentication:

- A dedicated login page ensures that only authorized users can access the quiz platform. This feature not only enhances security but also allows personalized user sessions. It serves as the first layer of control, preventing unauthorized access and enabling the storage of individual scores in future enhancements.

Quiz Start:

- Upon successful login, users are directed to a separate quiz page. This clear separation between login and quiz interface helps maintain a smooth user flow and improves the overall user experience. The quiz interface is clean, distraction-free, and optimized for user focus.

Question Randomization:

- To prevent predictability and cheating, the system includes dynamic question randomization. Questions are selected at random from a pool covering various categories, ensuring each quiz attempt is unique. This also allows for broader topic coverage and a fair assessment of knowledge.

Result Display:

- After the quiz is submitted, users are redirected to a dedicated result page. This page displays the final score, the number of correct answers, and optionally, the correct answers for each question. This immediate feedback helps users understand their performance and encourages learning through reflection.

Admin Panel:

- The system includes a robust admin panel that allows administrators to manage quiz content effectively. Admins can:
- Add new questions to expand the quiz database
- Edit existing questions to update outdated content
- Delete irrelevant or incorrect questions
- This panel ensures the quiz content remains current, relevant, and adaptable to various topics or educational needs.
- This proposed system emphasizes user engagement, content flexibility, and administrative control, making it suitable for educational institutions, training programs, or general knowledge platforms. It lays the foundation for future scalability such as user score tracking, leaderboards, and category-based quizzes.

1.4 Literature Review:

The increasing adoption of online assessment platforms has transformed how quizzes and tests are conducted. Traditional paper-based examinations are being replaced by digital solutions that offer efficiency, scalability, and automation. This literature review examines existing studies on quiz portals, online assessment systems, and their impact on education and training.

Online Assessment Systems: Online quiz systems have been widely studied in educational technology research. Studies highlight the following key advantages:

- **Accessibility:** Online quizzes provide learners with flexibility, allowing them to take assessments at their convenience (Kim, 2018).
- **Automated Evaluation:** Immediate feedback and automated grading enhance learning efficiency (Brown & Green, 2020).
- **Data Analytics:** Quiz platforms integrate analytics to assess student performance and provide insights to educators (Miller, 2021).
- The literature review highlights the growing importance of online quiz portals in education and corporate training. By incorporating real-time analytics, security.

The literature strongly supports the growing importance of online quiz portals in formal education, self-paced learning, and **corporate training**. As digital tools become increasingly integrated into learning ecosystems, quiz platforms are evolving from simple testing tools to sophisticated assessment systems that contribute to personalized learning and data-driven decision-making. The incorporation of **real-time analytics, secure assessment environments**, and **interactive design** makes these platforms essential in modern education and training infrastructure.

2.SYSTEM REQUIREMENTS

The system requirements outline the essential components needed for the successful development, deployment, and operation of the **Interactive Online Quiz Application**. These requirements are categorized into three main sections: **Hardware Requirements**, **Software Requirements**, and the **Software Requirements Specification (SRS)**. Each section ensures that the system is compatible with the necessary technical environment and meets the intended performance criteria.

2.1 Minimum Hardware Requirements:

Processor: Intel Core i3 (or equivalent AMD processor)

RAM: 4 GB

Storage: 20 GB of free space

Display: 1024×768 resolution

Network: 10 Mbps Internet connection

Input Devices: Keyboard & Mouse

Recommended Hardware Requirements:

Processor: Intel Core i5/i7 (or equivalent AMD Ryzen)

RAM: 8 GB or higher

Storage: 50 GB free SSD storage

Display: Full HD (1920×1080) resolution

Network: 50 Mbps Internet connection

Input Devices: Keyboard, Mouse, or Touchscreen

2.1 Software Requirements

These are the software dependencies needed for running and developing the system.

For End Users (Client-Side)

Operating System: Windows 10/11, macOS, Linux

Web Browser: Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari (latest versions)

Network: Internet access for online hosting

Operating System: Windows 10/11, macOS, Linux

Text Editor/IDE:

Visual Studio Code

Web Technologies:

Frontend: HTML, CSS, JavaScript

Backend (optional for future enhancements): Node.js, PHP, Python (Django/Flask), or Java (Spring Boot)

Database (optional if storing data): MySQL, PostgreSQL, Firebase, or MongoDB

Web Server: Apache, Nginx, or Node.js for hosting.

2.2 Software Requirements Specification(SRS):

The purpose of this document is to define the software requirements for the Interactive Online Quiz Application. It outlines the functional and non-functional requirements, system features, and design constraints necessary for successful development and deployment.

The application is a web-based quiz platform designed using HTML, CSS, and JavaScript. It allows authenticated users to participate in timed quizzes, view results, and navigate questions dynamically. An admin panel enables management of quiz content, including adding, editing, and deleting questions.

Intended Audience

- Project developers
- Academic evaluators
- System testers
- Future maintainers or contributors

Overall Description:

The system is a standalone web application hosted locally or on a web server. It operates independently without external APIs or third-party backend integration.

Product Functions:

- Secure user login
- Start quiz after login
- Load randomized questions dynamically
- Timer-based quiz progression
- Real-time score tracking
- View final results with correct answers
- Admin panel for question management

User Classes and Characteristics:

- User: Can log in, take quizzes, and view results
- Admin: Can log in and manage quiz content (add/edit/delete questions)

Operating Environment:

- Compatible with modern web browsers (Chrome, Firefox, Edge, Safari)
- Runs on any operating system (Windows, Linux, macOS) with browser support
- Optional local hosting via XAMPP, WAMP, or Live Server

3.SYSTEM DESIGN

3.1. Modules of System:

The system is divided into several functional modules to ensure modularity, maintainability, and ease of development. Each module handles a specific responsibility in the application workflow.

1. User Authentication Module

Purpose: Restricts access to only registered or authorized users.
Functions:

- User login with username and password (or simple input validation).
- Session initiation or local verification.
- Redirects users to the quiz dashboard upon successful login.

2. Quiz Interface Module

Purpose: Serves as the main interface where users take the quiz.
Functions:

- Display of questions and options.
- Navigation buttons (Next, Previous).
- Real-time timer display.
- Option selection and answer saving.

3. Question Management Module

Purpose: Loads and randomizes questions from the question bank.
Functions:

- Fetches questions dynamically from a JSON object or array.
- Randomizes the order of questions and options.
- Categorizes questions if needed (e.g., General, Tech, Math).

4. Timer Module

Purpose: Controls the time allotted for the quiz.
Functions:

- Countdown timer per quiz or per question.
- Automatically submits the quiz when time runs out.
- Warns users when time is about to end (e.g., last 10 seconds).

5. Answer Tracking Module

- Purpose: Records user selections throughout the quiz session.
- Functions:
  - Tracks selected answers for each question.
  - Allows modification of answers before submission.
  - Supports navigation between questions without data loss.

6. Result Evaluation Module

Purpose: Processes submitted answers and calculates scores.
Functions:

- Compares selected answers with correct answers.

- Calculates the total score.
- Identifies correct and incorrect responses.

7. Result Display Module

Purpose: Shows final quiz results to the user.
Functions:

- Displays total score, percentage, and feedback.
- Optionally shows correct answers and explanations.
- Provides a "Try Again" or "Logout" button.

8. Admin Panel Module

Purpose: Enables backend management of quiz content.
Functions:

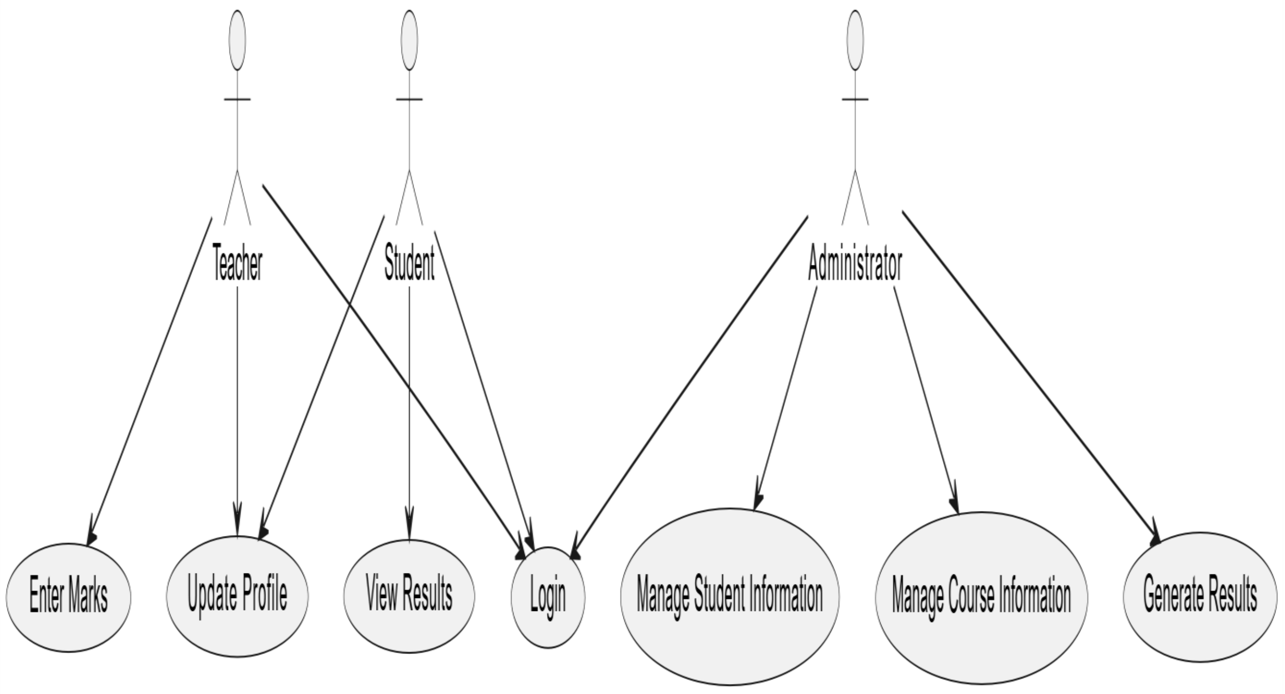
- Admin login (if implemented).
- Add new questions with options and correct answers.
- Edit or update existing questions.
- Delete obsolete or incorrect questions.
- View all quiz content in a table or list format.

Features:

- **Leaderboard Module –** Shows rankings of users based on scores.
- **User Management Module –** Allows registration, profiles, and user history.
- **Feedback Module –** Collects user feedback after the quiz.

3.2. UML Diagrams:

1. Use Case Diagram
The Use Case Diagram shows the different actors (users) and their interactions with the system.
User (Student) – Takes the quiz.
System – Manages quiz, questions, timer, and score calculation.
Use Cases: User Login – User enters valid credentials.
Start Quiz – Begins the quiz with a timer.
Answer Questions – Select answers and navigate between questions.
Submit Quiz – Submit manually or auto-submit on timeout.
View Results – Score is displayed after submission.
View Recent Assessments – Users can check past quiz attempts.


Teacher
Student
Administrator
Enter Marks
Update Profile
View Results
Login
Manage Student Information
Manage Course Information
Generate Results

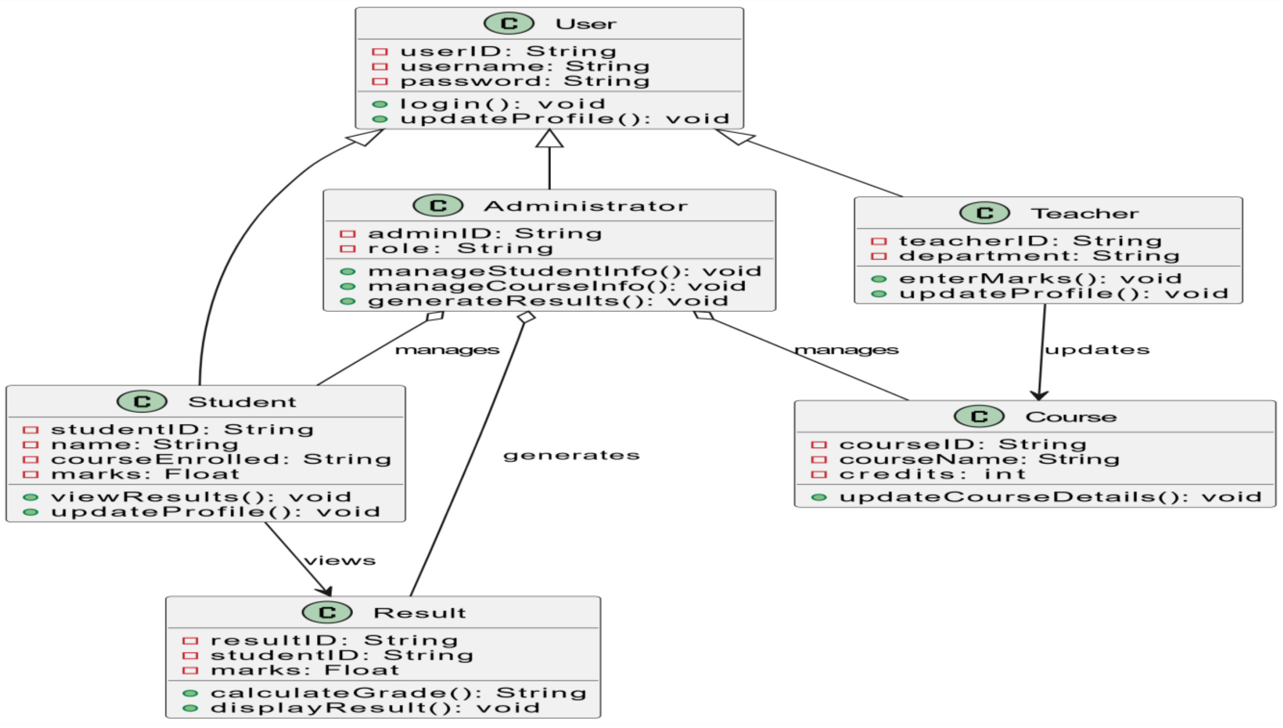

User
userID: String
username: String
password: String
login(): void
updateProfile(): void
Administrator
adminID: String
role: String
manageStudentInfo(): void
manageCourseInfo(): void
generateResults(): void
Teacher
teacherID: String
department: String
enterMarks(): void
updateProfile(): void
manages
manages
updates
Student
studentID: String
name: String
courseEnrolled: String
marks: Float
viewResults(): void
updateProfile(): void
generates
Course
courseID: String
courseName: String
credits: int
updateCourseDetails(): void
views
Result
resultID: String
studentID: String
marks: Float
calculateGrade(): String
displayResult(): void

Fig:3.1.1

Process Flow Diagram:

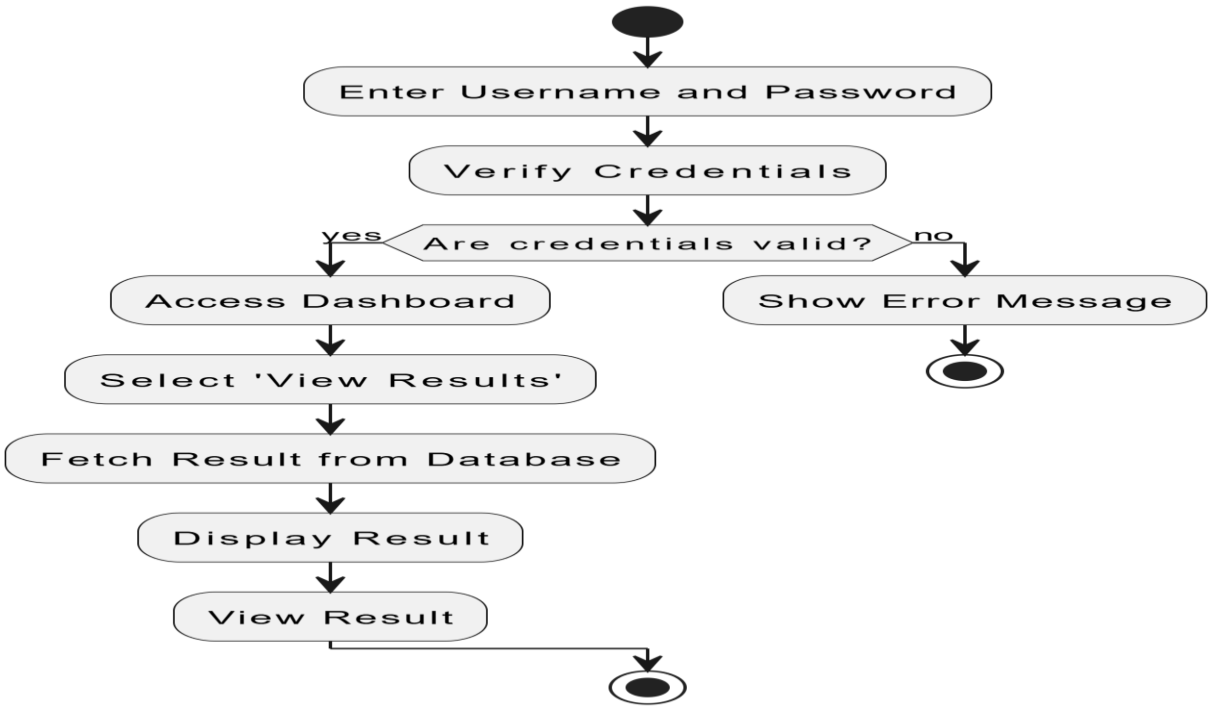


Fig:3.1.2

4.IMPLEMENTATION

4.1Sample Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>Login Page</title>
   <style>
      * { margin: 0; padding: 0; box-sizing: border-box; }
      body { font-family: Arial, sans-serif; display: flex; justify-content: center; align-items: center; height:
100vh; background-color: #f1f1f1; }
      .container {
         width: 80%;
         max-width: 900px;
         min-height: 600px;
         background: #fff;
         padding: 40px;
         border-radius: 10px;
         box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
         position: relative;
      }
      h1 { margin-bottom: 30px; text-align: center; }
      input { padding: 12px; width: 100%; margin: 10px 0; border-radius: 5px; border: 1px solid #ddd; }
      .btn {
         padding: 12px 24px;
         border: none;
         border-radius: 5px;
         cursor: pointer;
         font-size: 16px;
         margin: 5px;
      }
      .btn-login {
         background-color: #4CAF50;
         color: white;
         width: 100%;
      }
      .error { color: red; text-align: center; }
      #answer-container {
```

```
    text-align: left;
    margin: 30px 0;
}
.answer-option {
    display: flex;
    align-items: center;
    margin: 15px 0;
    padding: 15px;
    border: 1px solid #ddd;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.2s;
}
.answer-option input[type="radio"] {
    margin-right: 12px;
    width: auto;
    cursor: pointer;
}
.answer-option span {
    flex: 1;
}
.answer-option:hover {
    background-color: #f5f5f5;
}
.timer {
    position: absolute;
    top: 20px;
    right: 20px;
    font-size: 18px;
    font-weight: bold;
    background: #f8f8f8;
    padding: 10px;
    border-radius: 5px;
}
#question {
    font-size: 20px;
    margin: 20px 0;
}
.button-container {
```

```
            display: flex;
            justify-content: center;
            gap: 10px;
            margin-top: 30px;
        }
        .btn-prev { background-color: #666; color: white; }
        .btn-next { background-color: #4CAF50; color: white; }
        .btn-clear { background-color: #ff9800; color: white; }
        .btn-submit { background-color: #2196F3; color: white; }
        #result {
            text-align: center;
            margin-top: 20px;
            padding: 20px;
            background-color: #f8f8f8;
            border-radius: 5px;
        }
    </style>
</head>
<body>
    <div class="container" id="login-container">
        <h1>Login</h1>
        <div class="error" id="error-message"></div>
        <input type="text" id="username" placeholder="Username">
        <input type="password" id="password" placeholder="Password">
        <input type="text" id="name" placeholder="Enter your name">
        <button class="btn btn-login" onclick="login()">Login</button>
    </div>

    <div class="container" id="quiz-container" style="display:none;">
        <h1>Welcome, <span id="user-name"></span>!</h1>
        <div class="timer" id="timer">Time: 120</div>
        <h2 id="question"></h2>

        <div id="answer-container">
            <label class="answer-option">
                <input type="radio" name="answer" id="answer-0" value="0">
                <span id="answer-text-0"></span>
            </label>
            <label class="answer-option">
            <input type="radio" name="answer" id="answer-1" value="1">
            <span id="answer-text-1"></span>
        </label>
        <label class="answer-option">
            <input type="radio" name="answer" id="answer-2" value="2">
            <span id="answer-text-2"></span>
        </label>
        <label class="answer-option">
            <input type="radio" name="answer" id="answer-3" value="3">
            <span id="answer-text-3"></span>
        </label>
    </div>

    <div class="button-container">
        <button class="btn btn-prev" id="prev-btn" onclick="previousQuestion()">Previous</button>
        <button class="btn btn-next" onclick="nextQuestion()">Next</button>
        <button class="btn btn-clear" onclick="clearResponse()">Clear Response</button>
        <button class="btn btn-submit" onclick="submitQuiz()">Submit Quiz</button>
    </div>
</div>

<div class="container" id="result-container" style="display:none;">
    <div id="result">
        <h2>Your Score: <span id="score"></span> / 20</h2>
    </div>
</div>

<script type="module">
    // Import the functions you need from the SDKs you need
    import { initializeApp } from "https://www.gstatic.com/firebasejs/11.5.0/firebase-app.js";
    import { getDatabase, ref, get, child } from "https://www.gstatic.com/firebasejs/11.5.0/firebase-
database.js";
    import { getAnalytics } from "https://www.gstatic.com/firebasejs/11.5.0/firebase-analytics.js";

    // Your web app's Firebase configuration
    const firebaseConfig = {
        apiKey: "AIzaSyCn7LexHKkY2rAWeUV5bQ6wXKiMcLXDhFQ",
        authDomain: "quiz-portal-25.firebaseapp.com",
        databaseURL: "https://quiz-portal-25-default-rtdb.firebaseio.com",
        projectId: "quiz-portal-25",
        storageBucket: "quiz-portal-25.firebasestorage.app",
        messagingSenderId: "132340467831",
        appId: "1:132340467831:web:2cd303cd6b14d2f98d9e72",
        measurementId: "G-L19DSRCBJ9"
    };

    // Initialize Firebase
    const app = initializeApp(firebaseConfig);
    const analytics = getAnalytics(app);
    const db = getDatabase(app);

    let currentQuestion = 0;
    let score = 0;
    let answers = Array(10).fill(null);
    let timeLeft = 100;
    let timer;
    let loggedIn = false;
    let questions = [];

    window.login = function() {
        const username = document.getElementById("username").value;
        const password = document.getElementById("password").value;
        const name = document.getElementById("name").value;

        if (username === "user" && password === "password" && name !== "") {
            loggedIn = true;
            document.getElementById("user-name").innerText = name;
            document.getElementById("login-container").style.display = "none";
            document.getElementById("quiz-container").style.display = "block";
            startQuiz();
        } else {
            document.getElementById("error-message").innerText = "Invalid credentials or name not
      provided. Please try again.";
        }
    }

    window.startQuiz = function() {
        timer = setInterval(() => {
            if (timeLeft > 0) {
                timeLeft--;
                document.getElementById("timer").innerText = "Time: " + timeLeft;
            } else {
                clearInterval(timer);
                submitQuiz();
            }
        }, 1000);

        loadQuestions();
    }

    window.loadQuestions = function() {
        const dbRef = ref(db);
        get(child(dbRef, 'questions')).then((snapshot) => {
            if (snapshot.exists()) {
                questions = snapshot.val();
                loadQuestion();
            } else {
                console.log("No data available");
            }
        }).catch((error) => {
            console.error(error);
        });
    }

    window.loadQuestion = function() {
        const q = questions[currentQuestion];
        if (q) {
            document.getElementById("question").innerText = q.question;
            for (let i = 0; i < 4; i++) {
                document.getElementById("answer-text-" + i).innerText = q.answers[i];
                document.getElementById("answer-" + i).checked = answers[currentQuestion] === i;
            }
            document.getElementById("prev-btn").style.display = currentQuestion === 0 ? "none" :
"inline-block";
        }
    }
```

```
        window.nextQuestion = function() {
            saveAnswer();
            if (currentQuestion < questions.length - 1) {
                currentQuestion++;
                loadQuestion();
            }
        }

        window.previousQuestion = function() {
            saveAnswer();
            if (currentQuestion > 0) {
                currentQuestion--;
                loadQuestion();
            }
        }

        window.saveAnswer = function() {
            let selectedOption = document.querySelector('input[name="answer"]:checked');
            answers[currentQuestion] = selectedOption ? parseInt(selectedOption.value) : null;
        }

        window.clearResponse = function() {
            document.querySelectorAll('input[name="answer"]').forEach(radio => radio.checked = false);
            answers[currentQuestion] = null;
        }

        window.submitQuiz = function() {
            clearInterval(timer);
            saveAnswer();
            score = answers.filter((ans, i) => questions[i] && ans === questions[i].correct).length;
            document.getElementById("quiz-container").style.display = "none";
            const resultDiv = document.getElementById("result");
            resultDiv.innerHTML = `<h2 style="color: green;">Your quiz has been completed
successfully!</h2>
                        <p>Username: ${document.getElementById("user-name").innerText}</p>
                        <p>Score: ${score} / 10</p>`;
            document.getElementById("result-container").style.display = "block";
        }
    </script>
```

```
</body>
</html>
```

Quiz

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Start Quiz - Quiz Portal</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="home.html">Home</a></li>
                <li><a href="recent-assessments.html">Recent Assessments</a></li>
                <li><a href="start-quiz.html">Start Quiz</a></li>
                <li><a href="sign-out.html">Sign Out</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h1>Welcome to the Quiz Portal</h1>
        <p>To start a new quiz, please read the instructions below:</p>
        <ol>
            <li>Select the quiz you want to take.</li>
            <li>Read each question carefully.</li>
            <li>Select the answer you believe is correct.</li>
            <li>Submit your answers at the end of the quiz.</li>
        </ol>
        <button onclick="startQuiz()">Begin Quiz</button>
    </main>
    <footer>
        <p>&copy; 2023 Quiz Portal. All rights reserved.</p>
    </footer>
    <script>
        function startQuiz() {
            // Logic to start the quiz goes here
            window.location.href = 'start.html'; // Redirect to the quiz page
        }
    </script>
</body>
</html>
```

recent-assessments

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login Page</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.2/css/all.min.css"
integrity="sha512-
SnH5WK+bZxgPHs44uWIX+LLJAJ9/2PkPKZ5QiAj6Ta86w+fsb2TkcmfRyVX3pBnMFcV7oQPJkl9
QevSCWr3W6A==" crossorigin="anonymous" referrerpolicy="no-referrer" />
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500;600;700;800;900
&display=swap" rel="stylesheet" />
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: "Poppins", sans-serif;
        }

        body {
            background-color: #c9d6ff;
            background: linear-gradient(to right, #e2e2e2, #c9d6ff);
            display: flex;
            align-items: center;
            justify-content: center;
            flex-direction: column;
            height: 100vh;
        }

        .container {
            background-color: #fff;
            border-radius: 150px;
            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.35);
            position: relative;
            overflow: hidden;
            width: 768px;
            max-width: 100%;
            min-height: 480px;
        }

        .container p {
            font-size: 14px;
            line-height: 20px;
            letter-spacing: 0.3px;
            margin: 20px 0;
        }

        .container span {
            font-size: 12px;
        }

        .container a {
            color: #333;
            font-size: 13px;
            text-decoration: none;
            margin: 15px 0 10px;
        }

        .container button {
            background-color: #13c755;
            color: #fff;
            padding: 10px 45px;
            border: 1px solid transparent;
            border-radius: 8px;
            font-weight: 600;
            letter-spacing: 0.5px;
            text-transform: uppercase;
            margin-top: 10px;
            cursor: pointer;
        }
```

```
.container button.hidden {
    background-color: transparent;
    border-color: #fff;
}

.container form {
    background-color: #fff;
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    padding: 0 40px;
    height: 100%;
}

.container input {
    background-color: #eee;
    border: none;
    margin: 8px 0;
    padding: 10px 15px;
    font-size: 13px;
    border-radius: 8px;
    width: 100%;
    outline: none;
}

.sign-up, .sign-in {
    position: absolute;
    top: 0;
    height: 100%;
    transition: all 0.6s ease-in-out;
}

.sign-in {
    left: 0;
    width: 50%;
    z-index: 2;
}
```

```
.container.active .sign-in {
    transform: translateX(100%);
}

.sign-up {
    left: 0;
    width: 50%;
    z-index: 1;
    opacity: 0;
}

.container.active .sign-up {
    transform: translateX(100%);
    opacity: 1;
    z-index: 5;
    animation: move 0.6s;
}

@keyframes move {
    0%, 49.99% {
        opacity: 0;
        z-index: 1;
    }
    50%, 100% {
        opacity: 1;
        z-index: 5;
    }
}

.toogle-container {
    position: absolute;
    top: 0;
    left: 50%;
    width: 50%;
    height: 100%;
    overflow: hidden;
    border-radius: 150px;
    z-index: 1000;
    transition: all 0.6s ease-in-out;
```

```
}

.container.active .toogle-container {
    transform: translateX(-100%);
    border-radius: 150px;
}

.toogle {
    background-color: #1fb536;
    height: 100%;
    background: linear-gradient(to right, #1ab14c, #17a82f);
    color: #fff;
    position: relative;
    left: -100%;
    width: 200%;
    transform: translateX(0);
    transition: all 0.6s ease-in-out;
}

.container.active .toogle {
    transform: translateX(50%);
}

.toogle-panel {
    position: absolute;
    width: 50%;
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    padding: 0 30px;
    text-align: center;
    top: 0;
    transform: translateX(0);
    transition: all 0.6s ease-in-out;
}

.toogle-left {
        transform: translateX(-200%);
    }

    .container.active .toogle-left {
        transform: translateX(0);
    }

    .toogle-right {
        right: 0;
        transform: translateX(0);
    }

    .container.active .toogle-right {
        transform: translateX(200%);
    }
</style>
<script type="module">
    // Import the functions you need from the SDKs you need
    import { initializeApp } from "https://www.gstatic.com/firebasejs/11.5.0/firebase-app.js";
    import { getAuth, createUserWithEmailAndPassword, signInWithEmailAndPassword, signOut }
from "https://www.gstatic.com/firebasejs/11.5.0/firebase-auth.js";
    import { getDatabase, ref, set, get, child } from "https://www.gstatic.com/firebasejs/11.5.0/firebase-
database.js";

    // Your web app's Firebase configuration
    const firebaseConfig = {
        apiKey: "AIzaSyCn7LexHKkY2rAWeUV5bQ6wXKiMcLXDhFQ",
        authDomain: "quiz-portal-25.firebaseapp.com",
        databaseURL: "https://quiz-portal-25-default-rtdb.firebaseio.com",
        projectId: "quiz-portal-25",
        storageBucket: "quiz-portal-25.appspot.com",
        messagingSenderId: "132340467831",
        appId: "1:132340467831:web:2cd303cd6b14d2f98d9e72",
        measurementId: "G-L19DSRCBJ9"
    };

    // Initialize Firebase
    const app = initializeApp(firebaseConfig);
    const auth = getAuth(app);
    const db = getDatabase(app);

    window.showView = function(viewId) {
        const views = document.querySelectorAll('.container');
        views.forEach(view => view.classList.add('hidden'));
        document.getElementById(viewId).classList.remove('hidden');
    }

    document.getElementById('sign-up-form').addEventListener('submit', function(event) {
        event.preventDefault();
        const email = document.querySelector('.sign-up input[type="text"][placeholder="Email"]').value;
        const password = document.querySelector('.sign-up
input[type="password"][placeholder="Password"]').value;
        const name = document.querySelector('.sign-up input[type="text"][placeholder="Name"]').value;

        createUserWithEmailAndPassword(auth, email, password)
          .then((userCredential) => {
            // Signed up
            const user = userCredential.user;
            set(ref(db, 'users/' + user.uid), {
              username: name,
              email: email
            });
            alert('Sign-up successful!');
            showView('login');
          })
          .catch((error) => {
            const errorCode = error.code;
            const errorMessage = error.message;
            console.error(`Error [${errorCode}]: ${errorMessage}`);
            alert(`Error: ${errorMessage}`);
          });
    });

    document.getElementById('login-form').addEventListener('submit', function(event) {
        event.preventDefault();
        const email = document.querySelector('.sign-in input[type="text"][placeholder="Email"]').value;
        const password = document.querySelector('.sign-in
input[type="password"][placeholder="Password"]').value;
```

```
            signInWithEmailAndPassword(auth, email, password)
                .then((userCredential) => {
                    // Logged in
                    const user = userCredential.user;
                    get(child(ref(db), `users/${user.uid}`)).then((snapshot) => {
                        if (snapshot.exists()) {
                            const userData = snapshot.val();
                            alert(`Welcome back, ${userData.username}!`);
                            showView('recent-assessments');
                        } else {
                            alert("No user data found!");
                        }
                    }).catch((error) => {
                        console.error(`Error: ${error.message}`);
                        alert(`Error: ${error.message}`);
                    });
                })
                .catch((error) => {
                    const errorCode = error.code;
                    const errorMessage = error.message;
                    console.error(`Error [${errorCode}]: ${errorMessage}`);
                    alert(`Error: ${errorMessage}`);
                });
        });

        window.signOutUser = function() {
            signOut(auth).then(() => {
                alert('Sign-out successful!');
                showView('home');
            }).catch((error) => {
                console.error(`Error: ${error.message}`);
                alert(`Error: ${error.message}`);
            });
        }
    </script>
</head>
<body>
    <div class="container" id="container">
        <div class="sign-up">
            <form id="sign-up-form">
                <h1>Create Account</h1>
                <input type="text" placeholder="Name" />
                <input type="text" placeholder="Email" />
                <input type="password" placeholder="Password" />
                <button type="submit">Sign Up</button>
            </form>
        </div>
        <div class="sign-in">
            <form id="login-form">
                <h1>Sign In</h1>
                <input type="text" placeholder="Email" />
                <input type="password" placeholder="Password" />
                <a href="#">Forgot password</a>
                <button type="submit">Sign In</button>
            </form>
        </div>
        <div class="toogle-container">
            <div class="toogle">
                <div class="toogle-panel toogle-left">
                    <h1>Welcome User!</h1>
                    <p>If you already have an account</p>
                    <button class="hidden" id="login">Sign In</button>
                </div>
                <div class="toogle-panel toogle-right">
                    <h1>Hello, User!</h1>
                    <p>If you don't have an account</p>
                    <button class="hidden" id="register">Sign Up</button>
                </div>
            </div>
        </div>
    </div>
    <script>
        const container = document.getElementById("container");
        const registerbtn = document.getElementById("register");
        const loginbtn = document.getElementById("login");

        registerbtn.addEventListener("click", () => {
            container.classList.add("active");
        });

        loginbtn.addEventListener("click", () => {
            container.classList.remove("active");
        });
    </script>
</body>
</html>
```

Home

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Quiz Portal - Home</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Welcome to the Quiz Portal</h1>
        <nav>
            <ul>
                <li><a href="home.html">Home</a></li>
                <li><a href="recent-assessments.html">Recent Assessments</a></li>
                <li><a href="start-quiz.html">Start Quiz</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h2>Your Learning Journey Starts Here!</h2>
        <p>Explore our quizzes and assessments to enhance your knowledge.</p>
    </main>
    <footer>
        <p>&copy; 2023 Quiz Portal. All rights reserved.</p>
    </footer>
</body>
</html>
```

Admin

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Page</title>
    <link rel="stylesheet" href="styles.css">
    <style>
        .admin-container {
            width: 80%;
            max-width: 900px;
            min-height: 600px;
            background: #fff;
            padding: 40px;
            border-radius: 10px;
            box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
            margin: 50px auto;
        }
        .admin-container h1 {
            text-align: center;
            margin-bottom: 30px;
        }
        .admin-container input, .admin-container textarea {
            width: 100%;
            padding: 12px;
            margin: 10px 0;
            border-radius: 5px;
            border: 1px solid #ddd;
        }
        .admin-container button {
            padding: 12px 24px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-size: 16px;
            margin: 5px;
            background-color: #4CAF50;
            color: white;
        }
        .admin-container .question-list {
            margin-top: 30px;
        }
        .admin-container .question-item {
            padding: 15px;
            border: 1px solid #ddd;
            border-radius: 5px;
            margin-bottom: 10px;
            position: relative;
        }
        .admin-container .question-item button {
            position: absolute;
            top: 10px;
            right: 10px;
            background-color: #f44336;
        }
        .admin-container .question-item button.edit {
            right: 150px; /* Adjusted to create space */
            background-color: #2196F3;
        }
    </style>
</head>
<body>
    <div class="admin-container">
        <h1>Admin Panel</h1>
        <input type="number" id="total-questions" placeholder="Enter total number of questions">
        <button onclick="setTotalQuestions()">Set Total Questions</button>
        <input type="text" id="new-question" placeholder="Enter new question" style="display:none;">
        <textarea id="new-options" placeholder="Enter options separated by commas"
style="display:none;"></textarea>
        <input type="number" id="correct-option" placeholder="Enter correct option number (0-based)"
style="display:none;">
        <button onclick="addQuestion()" style="display:none;" id="add-question-btn">Add
Question</button>
        <div class="question-list" id="question-list"></div>
    </div>


    <script type="module">
        // Import the functions you need from the SDKs you need
        import { initializeApp } from "https://www.gstatic.com/firebasejs/11.5.0/firebase-app.js";
        import { getDatabase, ref, set, get, child } from "https://www.gstatic.com/firebasejs/11.5.0/firebase-
    database.js";

        // Your web app's Firebase configuration
        const firebaseConfig = {
            apiKey: "AIzaSyCn7LexHKkY2rAWeUV5bQ6wXKiMcLXDhFQ",
            authDomain: "quiz-portal-25.firebaseapp.com",
            databaseURL: "https://quiz-portal-25-default-rtdb.firebaseio.com",
            projectId: "quiz-portal-25",
            storageBucket: "quiz-portal-25.firebasestorage.app",
            messagingSenderId: "132340467831",
            appId: "1:132340467831:web:2cd303cd6b14d2f98d9e72",
            measurementId: "G-L19DSRCBJ9"
        };

        // Initialize Firebase
        const app = initializeApp(firebaseConfig);
        const db = getDatabase(app);

        let questions = [];
        let totalQuestions = 0;
        let editIndex = -1;

        window.setTotalQuestions = function() {
            totalQuestions = parseInt(document.getElementById('total-questions').value);
            if (!isNaN(totalQuestions) && totalQuestions > 0) {
              document.getElementById('total-questions').style.display = 'none';
              document.querySelector('button[onclick="setTotalQuestions()"]').style.display = 'none';
              document.getElementById('new-question').style.display = 'block';
              document.getElementById('new-options').style.display = 'block';
              document.getElementById('correct-option').style.display = 'block';
              document.getElementById('add-question-btn').style.display = 'block';
              loadQuestions();
            } else {
              alert('Please enter a valid number of questions.');
            }
        }

        window.addQuestion = function() {
```

```
    const questionText = document.getElementById('new-question').value;
    const optionsText = document.getElementById('new-options').value;
    const correctOption = parseInt(document.getElementById('correct-option').value);

    if (questionText && optionsText && !isNaN(correctOption)) {
        const options = optionsText.split(',').map(option => option.trim());
        const newQuestion = {
            question: questionText,
            answers: options,
            correct: correctOption
        };

        if (editIndex >= 0) {
            questions[editIndex] = newQuestion;
            editIndex = -1;
        } else {
            questions.push(newQuestion);
        }

        displayQuestions();
        clearInputs();
        if (questions.length === totalQuestions) {
            alert('All questions added successfully!');
            saveQuestionsToDatabase();
        }
    } else {
        alert('Please fill in all fields correctly.');
    }
}

window.loadQuestions = function() {
    const dbRef = ref(db);
    get(child(dbRef, 'questions')).then((snapshot) => {
        if (snapshot.exists()) {
            questions = snapshot.val();
            displayQuestions();
        } else {
            console.log("No data available");
        }
    }).catch((error) => {
        console.error(error);
    });
}

window.displayQuestions = function() {
    const questionList = document.getElementById('question-list');
    questionList.innerHTML = '';
    questions.forEach((q, index) => {
        const questionItem = document.createElement('div');
        questionItem.className = 'question-item';
        questionItem.innerHTML = `
            <h3>Question ${index + 1}</h3>
            <p>${q.question}</p>
            <p>Options: ${q.answers.join(', ')}</p>
            <p>Correct Option: ${q.correct}</p>
            <button class="edit" onclick="editQuestion(${index})">Edit</button>
            <button onclick="deleteQuestion(${index})">Delete</button>
        `;
        questionList.appendChild(questionItem);
    });
}

window.clearInputs = function() {
    document.getElementById('new-question').value = '';
    document.getElementById('new-options').value = '';
    document.getElementById('correct-option').value = '';
}

window.saveQuestionsToDatabase = function() {
    set(ref(db, 'questions'), questions)
        .then(() => {
            console.log('Questions saved successfully!');
        })
        .catch((error) => {
            console.error('Error saving questions:', error);
        });
}
```

```
        window.editQuestion = function(index) {
            const question = questions[index];
            document.getElementById('new-question').value = question.question;
            document.getElementById('new-options').value = question.answers.join(', ');
            document.getElementById('correct-option').value = question.correct;
            editIndex = index;
        }

        window.deleteQuestion = function(index) {
            questions.splice(index, 1);
            displayQuestions();
            saveQuestionsToDatabase();
        }
    </script>
</body>
</html>
```

4.2Test Cases:

The **User Authentication Module** was tested to verify whether valid login credentials redirect the user to the quiz page successfully, and whether invalid login attempts are properly rejected with an appropriate error message. Attempts to access the quiz directly via URL without logging in were also tested to confirm access control is in place.

For the **Quiz Module**, test cases included starting the quiz after login, navigating between questions using "Next" and "Previous" buttons, and selecting answers. The system was checked to ensure that user inputs were saved properly when navigating back and forth. Another important test was to let the timer expire to verify if the quiz automatically submits when the time limit is reached.

The **Score Evaluation Module** was tested by submitting different combinations of correct and incorrect answers. The results displayed after submission were compared with the expected outcomes to ensure scoring accuracy. Additionally, the visibility of correct answers post-submission was verified.

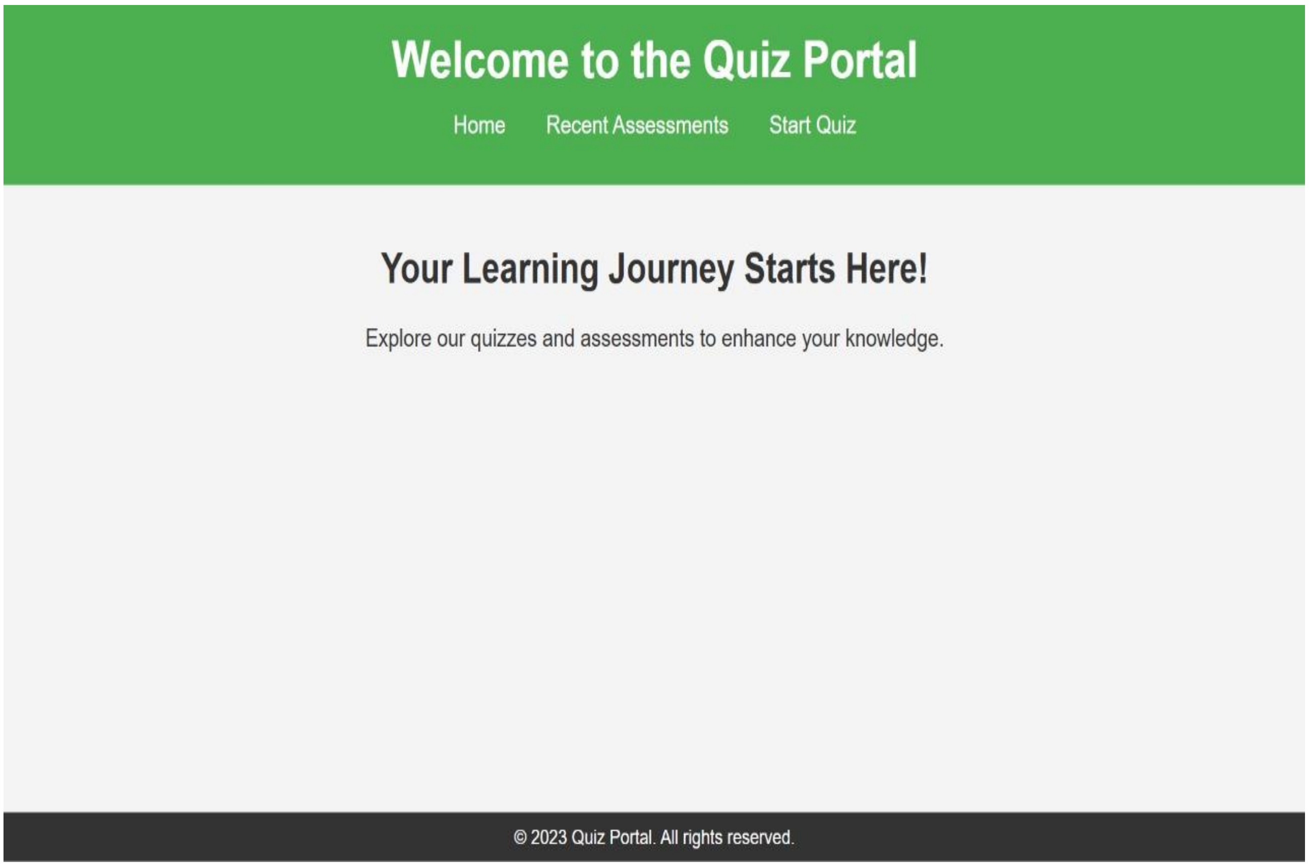
In the **Admin Panel**, test cases included adding new questions, editing existing ones, and deleting questions from the question bank. Each function was tested to ensure the changes were reflected in the quiz module as expected. Validations were also checked to ensure that only properly formatted questions and options were accepted.

Responsive design test cases were conducted by running the application on different devices and screen sizes to ensure that the layout adapted correctly, and the functionality remained consistent across platforms.

These test cases helped validate the core functionality and usability of the application, ensuring it performs as expected under different conditions. The testing phase also highlighted areas for future enhancement, such as user registration, database integration, and extended result analytics.

Test Case ID	Description	Steps	Expected Result
TC001	Login functionality	Enter valid username/password and click login	Redirect to home page
TC002	Add a new question (Admin)	Enter question details and click "Add Question"	Question appears in list
TC003	Start Quiz	Click "Start Quiz" button	First question is displayed
TC004	Submit Quiz	Answer all questions and click "Submit"	Results are displayed
TC005	Timer countdown	Start quiz and observe timer	Timer decreases every second

5.RESULT



Your Learning Journey Starts Here!

Explore our quizzes and assessments to enhance your knowledge.

© 2023 Quiz Portal. All rights reserved.

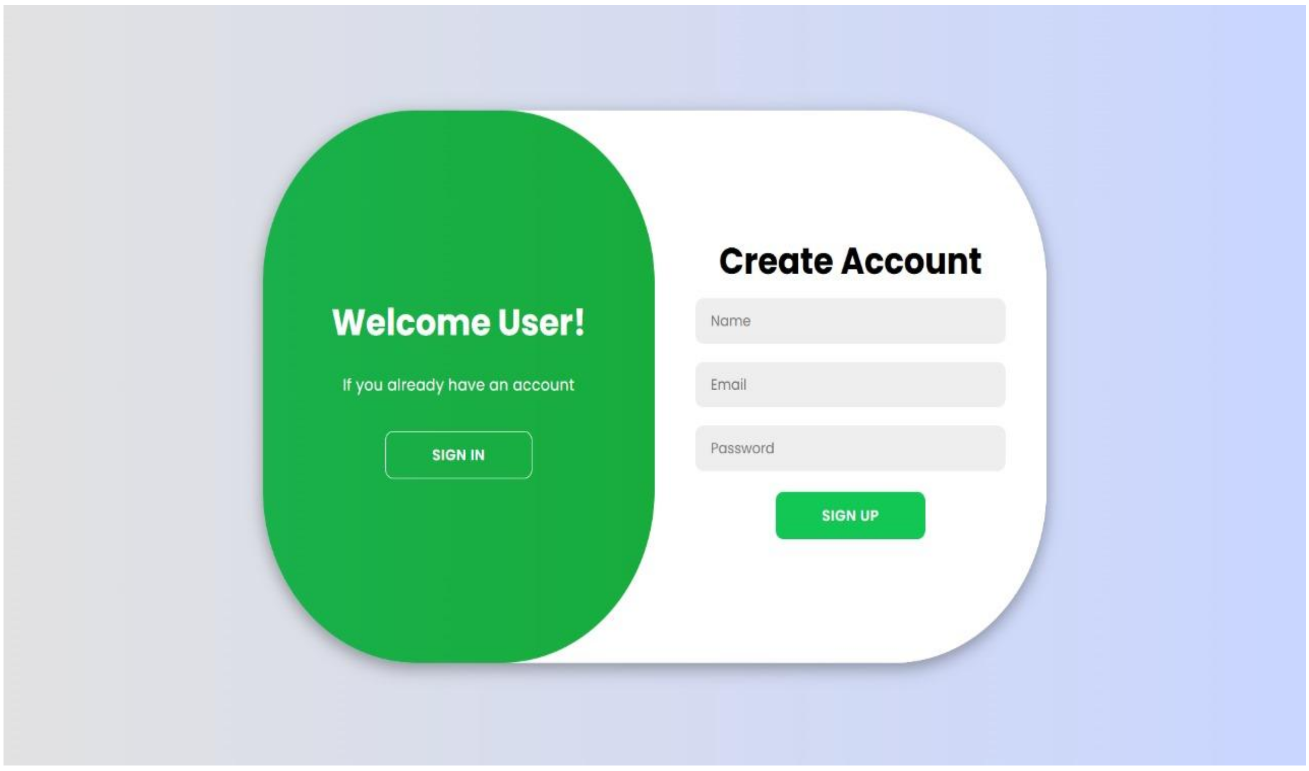


Fig:5.1

5.1Output Screens:

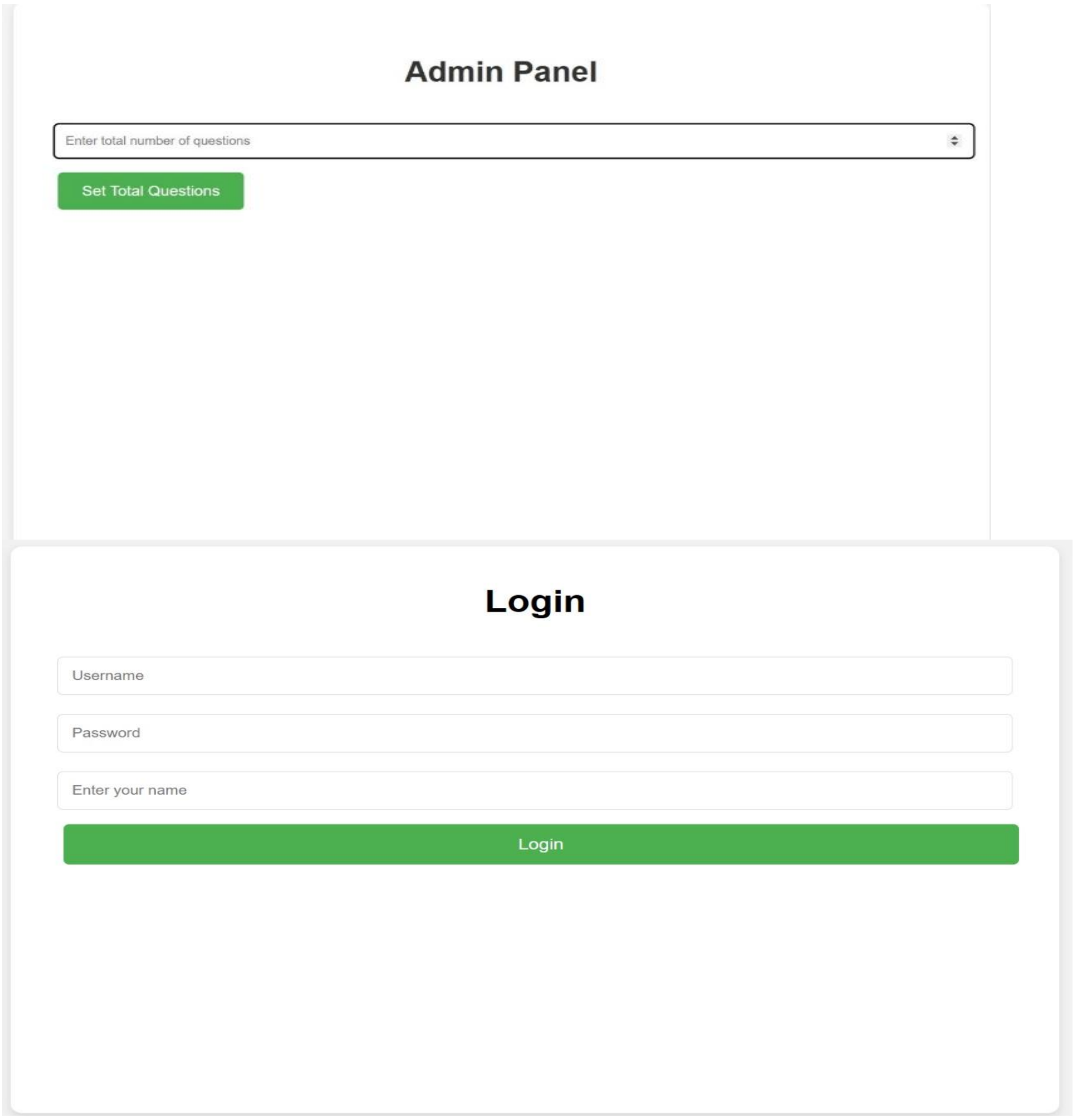
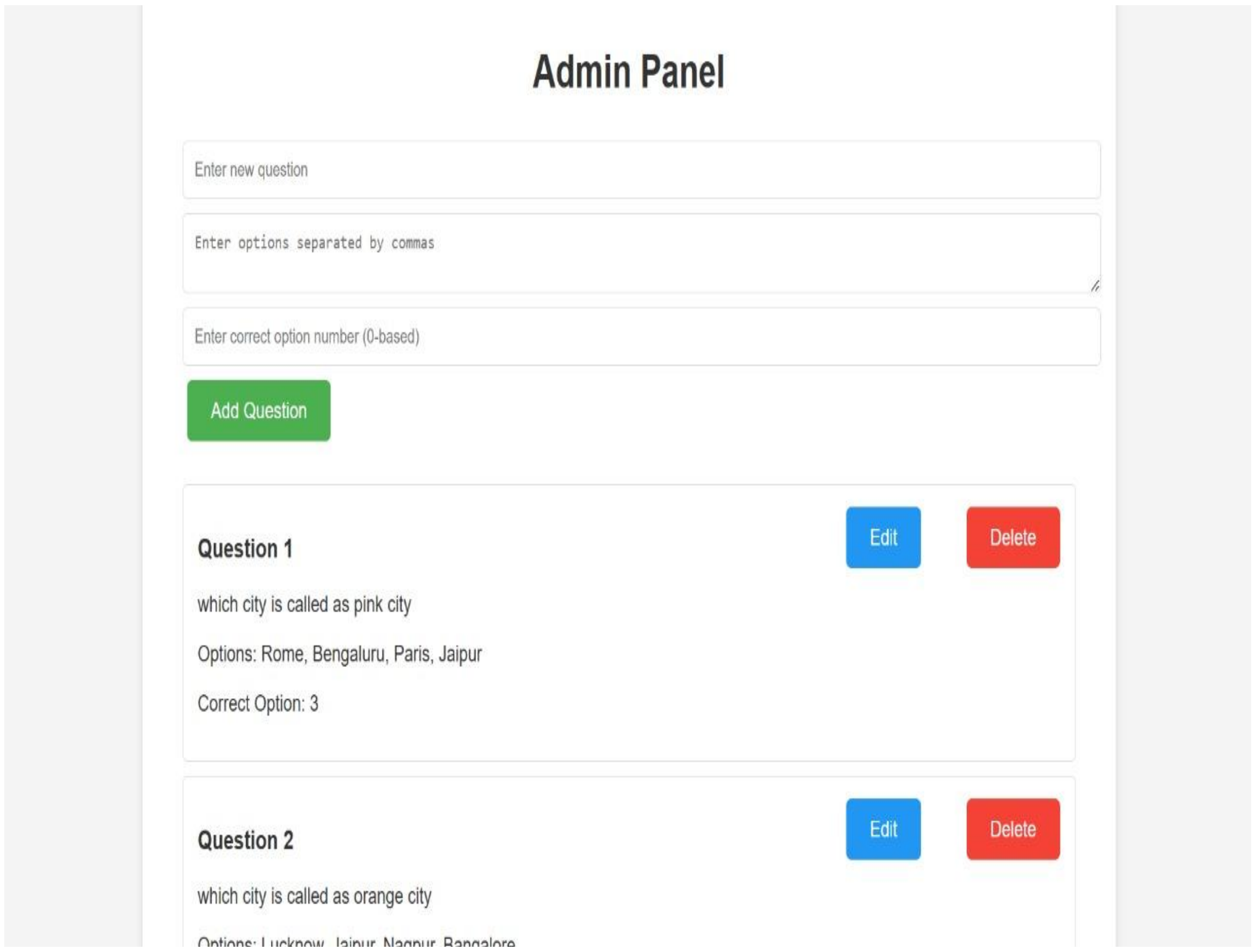


Fig:5.1.1


Admin Panel
Enter new question
Enter options separated by commas
Enter correct option number (0-based)
Add Question
Edit
Delete
Edit
Delete

Question 1

which city is called as pink city

Options: Rome, Bengaluru, Paris, Jaipur

Correct Option: 3

Question 2

which city is called as orange city

Options: Lucknow, Jaipur, Nagpur, Bangalore

Fig:5.2.2

6.CONCLUSION

QuizVerse is a dynamic and interactive online quiz system designed to deliver an engaging and efficient assessment experience. It integrates essential features such as secure **user authentication**, smooth **question navigation**, countdown **timers**, and automatic **result evaluation**, all within a user-friendly interface. The platform is built with a modular structure that supports easy updates and future feature enhancements, making it ideal for educational, training, or self-assessment purposes.

Scalability is a key focus of the system, allowing for seamless integration with **backend databases**, **user management systems**, and **advanced analytics** in future versions. This future-proof architecture ensures that QuizVerse can evolve into a more powerful learning and evaluation tool, capable of supporting larger datasets, diverse question formats, and user performance tracking over time.

With its responsive design, Quiztopia works smoothly across a variety of devices, including desktops, tablets, and smartphones, ensuring accessibility and convenience for users. Whether used in classrooms, corporate training, or casual knowledge testing, the system delivers a reliable and enjoyable quiz-taking experience.

References

Anderson, J. & Rainie, L. (2020). "The Future of Digital Learning: Online Education Trends."Discusses the impact of interactive quizzes in online learning.
Source: Pew Research Center.
Duckett, J. (2014). "HTML & CSS: Design and Build Websites."
Covers front-end development techniques useful for designing QuizVerse..
W3Schools – https://www.w3schools.com/.
Bootstrap Documentation – https://getbootstrap.com/.
GitHub Repositories (Open Source Projects for Reference)
Open Trivia API - https://opentdb.com/.

Project Link:https://github.com/keerthinerella11/keerthi4825.git