ಬೆಂಗಳೂರು
ನಗರ ವಿಶ್ವವಿದ್ಯಾಲಯ

BENGALURU
CITY UNIVERSITY

ಆಗು ನೀ ಅನಿಕೇತನ
BE BOUNDLESS

BANGALORE CITY UNIVERSITY

BANGALORE – 560001

PROJECT ENTITED

ONLINE QUIZ SYSTEM

A Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF COMPUTER APPLICATION

FOR THE ACADEMIC YEAR 2024-2025

SUBMITTED BY

ABINAV (U18GD22S0035)

Under the Esteemed Guidance of

Mrs. Shwetha B.R.

HOD, Department of Computer Science

Mr. Abhishekh Mathapati

Assistant Professor, Department of Computer Science

Project carried out at

SB COLLEGE OF MANAGEMENT STUDIES

SB COLLEGE OF MANAGEMENT STUDIES

5TH PHASE YELAHANKA, BANAGALORE-64

Certificate

This is to certify that this is a Bonafede report of the project work entitled **"Online Quiz System"** that is being submitted by **ABINAV (U18GD22S0035)**. A final year student of BCA, Bengaluru City University, in partial fulfillment of the requirement for the award of BCA during the year 2024-25.

Asst.Prof. Abhishekh Mathapati
Internal Guide

Asst.Prof. Shwetha B.R
HOD, Dept. of CS

Prof. B. Parvathi B Devi
Principal

ACKNOWLEDGEMENT

I would like to acknowledge all those whose guidance and encouragement has made us do what is done so far. We avail ourselves of the opportunity to express our deep sense of gratitude and sincere thanks to our Department of Computer Science which has always been a tremendous source of inspiration.

Our sincere gratitude to the principal **Prof. B. Parvathi B Devi,** for all the help rendered.

I am whole-heartedly thankful to **Prof. Shwetha B.R** Head of the Department of BCA, SB College of management studies, for allowing me to carry out this project.

I also have great pleasure in acknowledging a sense of gratitude to my project guide **Assistant Prof. Abhishekh Mathapati,** project guide, immense help and valuable advice.

I also thank all those members of the BCA Dept. Who has contributed directly or indirectly to the successful completion of this project.

I also thank our friends for their valuable suggestions and constructive criticisms.
I am thankful to them for their co-operation in the successful completion of my project.

DECLARATION

I am a student of final year BCA, SB College of management studies, hereby declare that this project work entitled **"Online Quiz System"** has been carried out by me only.

I hereby state that this project work is not submitted to any college or university for the award of any certificate/ Degree.

Place: Bengaluru
Date:

SUBMITTED BY:
ABINAV (U18GD22S0035)

ABSTRACT

The **Online Quiz System** is an advanced web-based application developed using the latest technologies, including **React.js, Firebase, and JavaScript**, designed to provide an engaging and interactive quiz experience for users. This innovative platform allows users to participate in a variety of quizzes, track their scores, and compete on a real-time **leaderboard**, ensuring a fun and competitive learning environment.

Registered users can seamlessly create profiles, attempt multiple quizzes, and monitor their progress, while administrators can efficiently manage quiz content and analyze user performance. The system's robust backend, powered by **Firebase Firestore**, ensures secure data storage, real-time updates, and seamless user authentication.

The **Online Quiz System** not only enhances learning but also fosters engagement through features such as **dynamic score tracking, game history, and user rankings**. With a modern, intuitive UI and a highly responsive design, the platform delivers a smooth and enjoyable experience for users across all devices.

We believe that this state-of-the-art online quiz platform will provide an exciting, educational, and competitive experience, supported by a well-structured backend and a dedicated team ready to assist users with any queries or improvements.

CONTENTS

INTRODUCTION

➢ Introduction To Project:

The **Online Quiz System** is a cutting-edge web application developed using **React.js and Firebase**. It is designed to provide an engaging and interactive platform for users to test their knowledge through a variety of quizzes while competing with others in real-time. This system offers a seamless and efficient way for users to participate in quizzes, track their progress, and view their rankings on a **dynamic leaderboard**.

Built with **modern web technologies**, the platform ensures a **fast, responsive, and intuitive** user experience across all devices. **React.js** powers the front-end, delivering a smooth and interactive UI, while **Firebase Firestore** provides a **secure, scalable, and real-time** backend to handle user authentication, quiz data, and score management.

The **Online Quiz System** integrates key features such as **user authentication, quiz history tracking, real-time score updates, and performance analytics** to enhance the overall experience. By leveraging the latest web technologies, this system ensures robustness, scalability, and maintainability, making it an **ideal solution** for learners, educators, and trivia enthusiasts alike.

➢ Purpose Of the Project

The **primary purpose** of the **Online Quiz System** is to provide an interactive and engaging platform for users to test their knowledge, compete with others, and track their progress in real-time. By leveraging modern **web technologies**, the system aims to deliver a **seamless, efficient, and accessible** quiz-taking experience.

The **specific objectives** of the project are as follows:

- **Enhance User Engagement**: Provide an interactive and intuitive interface that makes participating in quizzes a fun and rewarding experience.
- **Increase Accessibility**: Allow users to access quizzes anytime and from any device, ensuring maximum flexibility.
- **Ensure Data Security**: Implement robust authentication and database security measures to protect user data and maintain system integrity.
- **Encourage Continuous Improvement**: Gather user feedback to enhance the platform, introduce new features, and refine the quiz experience over time.

➢ Existing System

In the current educational landscape, traditional **quiz and assessment systems** largely rely on **pen-and-paper methods** or **basic online quizzes**, which often lack interactivity, real-time tracking, and engagement. Many existing systems are either too simplistic or too complex, making them **inaccessible or inefficient** for general users. Below are some key characteristics of the **existing quiz systems**:

For Users (Students & Participants):

- **Manual Quiz Taking**: Users often take quizzes using printed question papers or simple online forms without interactive elements.
- **Limited Online Presence**: Some platforms offer online quizzes, but they are often **not optimized for real-time scoring** or user engagement.
- **No Real-time Leaderboard**: Existing systems **lack real-time ranking** or performance comparison with other participants.
- **Basic User Profiles**: Most traditional quiz systems do not allow users to **track progress, maintain history, or personalize their experience**.

For Quiz Administrators & Educators:

- **Manual Question Management**: Quiz creation and question distribution are often done manually, leading to inefficiencies.
- **Static Score Calculation**: Many systems **lack automated real-time scoring**, requiring manual evaluation in some cases.
- **Limited Security Measures**: Without **authentication and anti-cheating mechanisms**, users can **manipulate quiz results** or access answers easily.
- **No Centralized Dashboard**: There is **no unified system** for managing quizzes, tracking scores, and monitoring user participation.

➢ Proposed System

The proposed **Online Quiz System** overcomes the limitations of traditional quiz-taking methods by leveraging modern web technologies like **React.js, Firebase, and JavaScript** to create an interactive and automated platform. It enhances accessibility, engagement, and efficiency, allowing users to take quizzes anytime while receiving real-time feedback and leaderboard updates. The system offers a seamless, user-friendly interface with secure authentication, ensuring a smooth experience for students, professionals, and educators.

For administrators, the system simplifies quiz management with automated scoring, real-time analytics, and dynamic content control. With built-in anti-cheating measures, progress tracking, and gamification elements, the **Online Quiz System** provides a **fair, engaging, and data-driven** approach to learning and assessments, making it a **modern solution for digital education and skill evaluation**.

SYSTEM ANALYSIS

➢ Software Development Lifecycle (SDLC) Model

The development of the **Online Quiz System** follows the **Agile Software Development Lifecycle (SDLC) model** due to its flexibility, iterative nature, and continuous improvement approach. Agile allows for frequent updates based on user feedback, ensuring a highly interactive and engaging quiz platform. This model emphasizes **collaboration, adaptability, and user satisfaction**, making it ideal for modern web applications like our **quiz system**.

Phases of Agile SDLC

1. **Requirement Gathering and Analysis**
   - Identify system requirements from educators and quiz takers.
   - Define objectives such as real-time scoring, user authentication, and leaderboard functionality.
2. **Planning**
   - Develop a roadmap outlining key features and sprint cycles.
   - Assign development tasks and set project milestones.
3. **Design**
   - Create wireframes for the user interface.
   - Define system architecture with **React.js for the frontend and Firebase for the backend**.
4. **Development**
   - Implement core features like quiz creation, user authentication, and score tracking.
   - Follow **continuous integration (CI) practices** for smooth development.
5. **Testing**
   - Conduct **unit, integration, and user acceptance testing**.
   - Ensure smooth performance across devices and browsers.
6. **Deployment**
   - Launch the platform for users via a **secure cloud-based hosting** service.
   - Ensure database connectivity and feature stability.
7. **Maintenance**
   - Monitor system performance and resolve any bugs.
   - Continuously update features based on user feedback for an enhanced experience.

➢ System Architecture

The **Online Quiz System** follows a **client-server architecture**, ensuring a **clear separation between the front-end and back-end** for better performance, security, and scalability. The system is built using **React.js for the front end, Firebase for the back end, and Firestore as the database** to enable real-time updates and seamless interactions.

Components of the System Architecture

1. Client-Side (Front-End)

- **Technology:** React.js
- **Description:** The front end provides an interactive and responsive user interface that allows users to take quizzes, view scores, and interact with the leaderboard in real-time.
- **Key Components:**
  - **Home Page:** Displays available quizzes.
  - **Quiz Interface:** Loads questions dynamically and records answers.
  - **Leaderboard Page:** Displays top scores and player rankings.
  - **Profile Page:** Allows users to view their past performances and update details.

---

2. Server-Side (Back-End)

- **Technology:** Firebase Functions
- **Description:** The backend is responsible for handling user authentication, storing quiz data, processing scores, and managing the leaderboard.
- **Key Modules:**
  - **Authentication Module:** Handles user registration and login via Firebase Authentication.
  - **Leaderboard Module:** Fetches and ranks top players based on performance.

---

3. Database

- **Technology:** Firestore (NoSQL Cloud Database)
- **Description:** A real-time cloud database that stores quiz questions, user details, and leaderboard rankings.
- **Key Collections:**
  - **Users:** Stores user profiles, including their username and quiz history.
  - **Quizzes:** Stores quiz questions, answers, and difficulty levels.
  - **Scores:** Records user scores and timestamps.
  - **Leaderboard:** Maintains rankings based on high scores.

FEASIBILITY REPORT

A **preliminary investigation** examines the feasibility of the **Online Quiz System**, assessing its technical, operational, and economic viability. The main objective is to determine whether developing the system is **practical, cost-effective, and beneficial** for users. This includes evaluating the possibility of future enhancements, ensuring smooth operation, and debugging existing functionalities. Any system is feasible if given unlimited resources and time. The **feasibility study** includes the following aspects:

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

➢ Technical Feasibility

The **Online Quiz System** leverages **React.js, Firebase, and Firestore**, ensuring it meets modern technological standards.

- **Does the required technology exist?**
  - Yes, React.js and Firebase are widely used, scalable, and well-supported for developing real-time web applications.
- **Can the system handle the required data load?**
  - Yes, Firestore provides cloud-based, scalable storage with real-time syncing, ensuring efficient data handling.
- **Will the system provide fast responses regardless of user load?**
  - Yes, Firebase's cloud infrastructure ensures low latency and optimal performance under varying loads.
- **Can the system be upgraded in the future?**
  - Yes, it is designed with modularity and scalability, making future updates and new feature additions seamless.
- **Is security and data integrity guaranteed?**
  - Yes, Firebase Authentication, Firestore security rules, and encryption mechanisms ensure secure data access and storage.

➢ Operational Feasibility

For a system to be successful, it must meet the **operational needs** of its users.

- **Will users (students and instructors) support and use the system?**
  - Yes, the system is designed based on user requirements, ensuring it meets their expectations.

- **Will the system be intuitive and effective?**
  - Yes, the **React.js** front end provides a smooth user experience, and Firebase ensures real-time performance.
- **Is there a risk of resistance from users?**
  - No, since the system is designed to **enhance learning, engagement, and accessibility**, resistance is unlikely.

The **Online Quiz System** is built with usability in mind, ensuring a **smooth transition** for users and minimizing adoption barriers.

➢ Economic Feasibility

The system must be a **worthwhile investment** by offering benefits that outweigh its costs.

- **Development Costs:**
  - Costs include initial **software development, hosting, and database setup**, but these are minimal due to Firebase's cost-effective pricing model.
- **Operational Costs:**
  - Expenses include **server hosting and maintenance**, but Firebase's pay-as-you-go model keeps costs manageable.
- **Long-Term Benefits:**
  - Increased **student engagement**, automated quiz evaluation, and improved learning experiences make it a valuable educational tool.

The **Online Quiz System** is **economically viable**, as the benefits of **enhanced learning, automated grading, and real-time performance** far outweigh the costs.

SYSTEM REQUIREMENTS SPECIFICATION

Hardware Requirements

The Coffee Ordering App requires a high-end system for optimal performance during development and testing. The following hardware specifications are recommended:

Component	Specification
Processor	12th Gen Intel(R) Core (TM) i5-12700H @ 2.70GHz
RAM	16.00 GB
Cache	1 MB or more
Hard Disk	500 GB SSD recommended for primary partition
Graphics Card	NVIDIA GeForce GTX 1650 or equivalent
Display	15.6" Full HD (1920 x 1080)

Software Requirements

Component	Specification
Operating System	Windows 11 Pro ©2023
Front End Framework	React Js
Back End Framework	Node.js, Express.js
Database	Firebase
Package Manager	NPM
Server	Local development server using Node.js
Version Control	Git, GitHub
Mobile Emulators	Android Emulator (part of Android Studio)
Other Tools	Postman (for API testing), Firebase (for authentication and database)

SYSTEM DESIGN

➢ Overview:

The **Online Quiz System** is a **web-based** platform developed to provide users with an interactive and efficient way to participate in quizzes. The system consists of a **frontend built with React.js**, a **backend powered by Node.js and Express.js**, and a **Firebase database** for storing user data, quizzes, and results. It follows a **three-tier architecture**, ensuring **scalability, security, and maintainability**.

System Architecture

The system is divided into **three main layers**:

- **Presentation Layer**
  - **User Interface:** Built with React.js for a responsive and interactive experience.
  - **Admin Panel:** A web-based interface for quiz management, user monitoring, and report generation.
- **Business Logic Layer**
  - **Backend Server:** Node.js with Express.js handles **quiz management, user authentication, result processing, and database interactions**.
- **Data Access Layer**
  - **Database:** Firebase stores **user profiles, quiz questions, answers, scores, and reports**, ensuring structured data management.

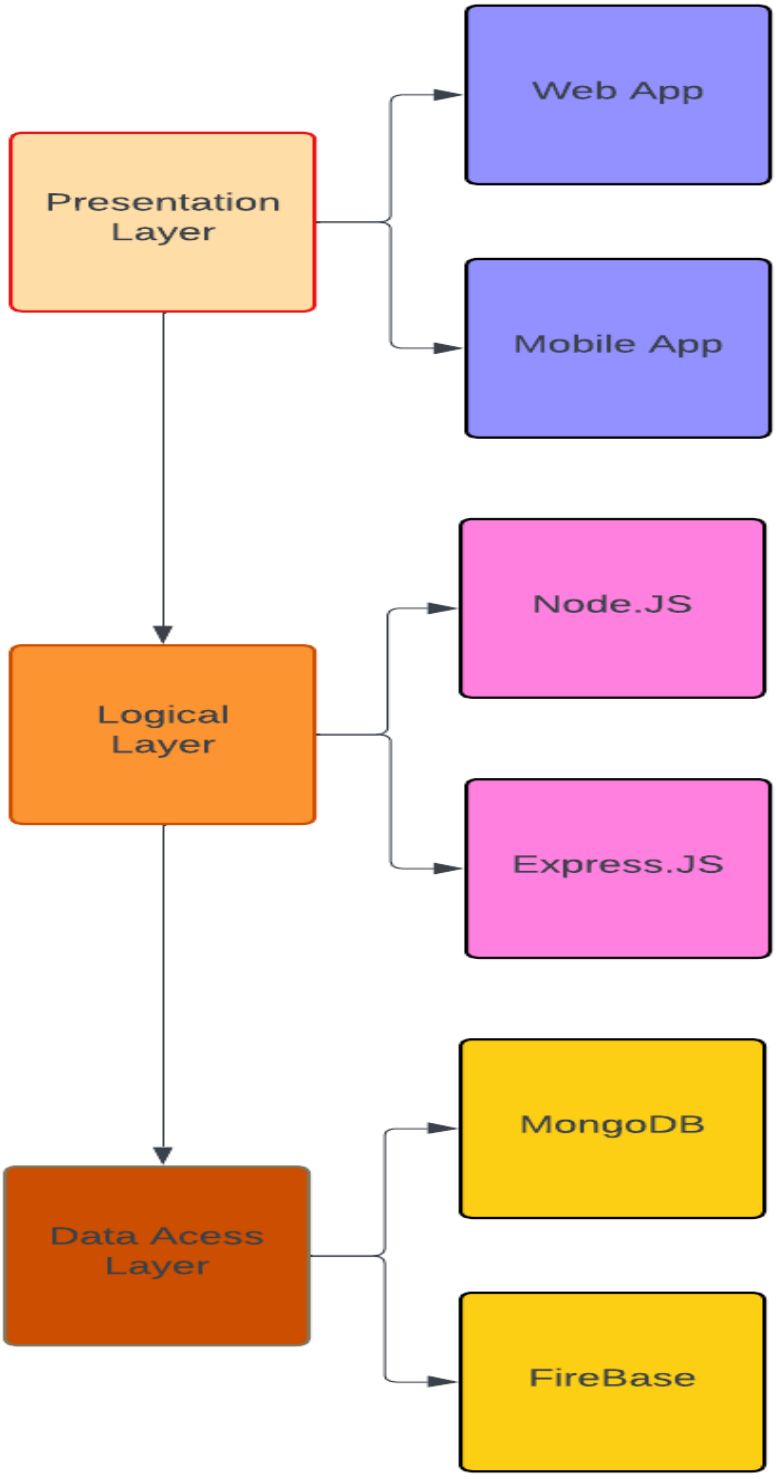
➢ Module design:

A well-structured **software design** ensures:

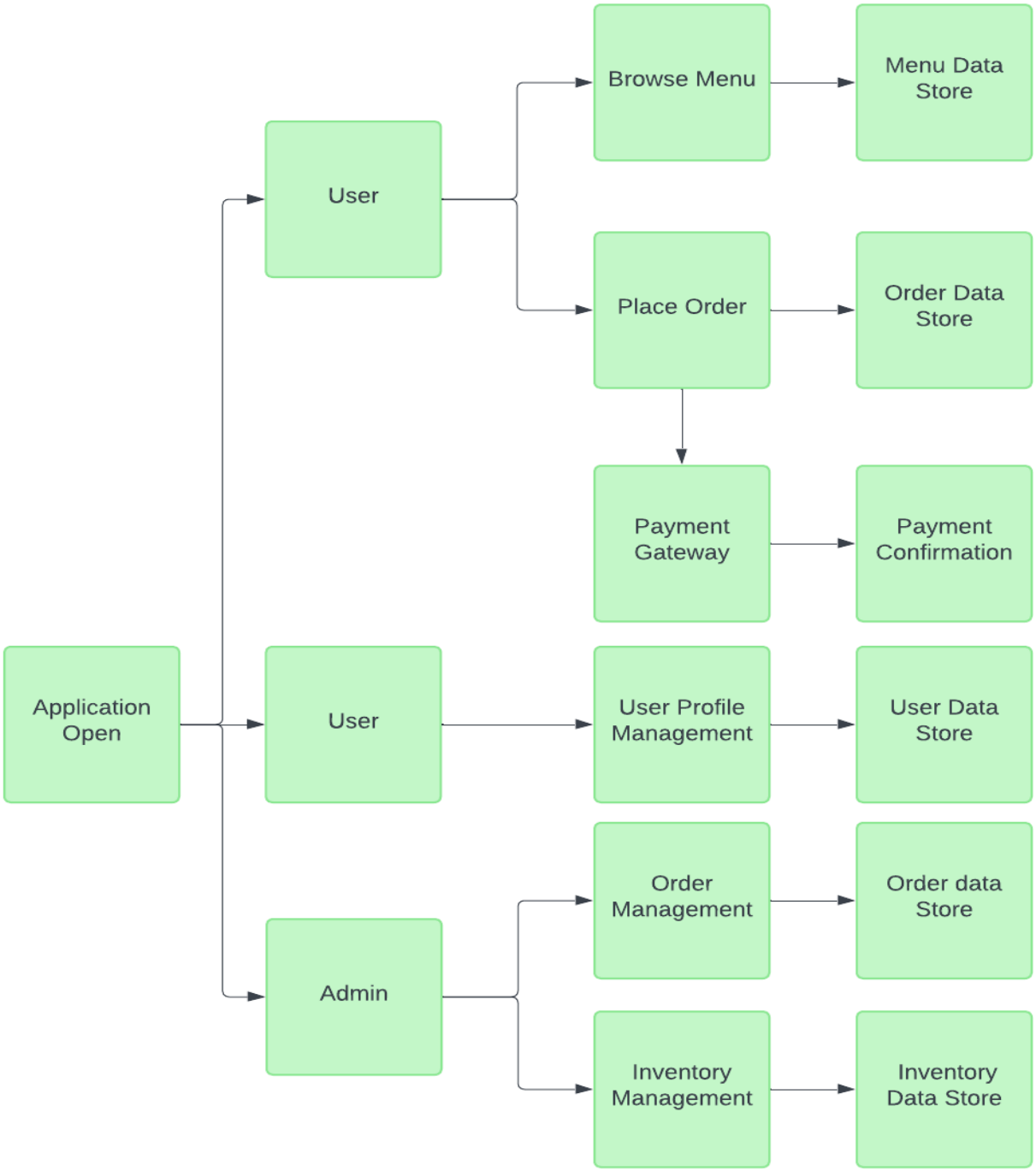
- **Efficiency** in quiz management and result processing.
- **Scalability** to support a large number of users.
- **Security** to protect user data and quiz integrity.

The **Online Quiz System** follows a **modular approach**, ensuring independent functionality for each component while maintaining **seamless integration**, leading to a **robust and high-performing system**.

➢ System Design:



➢ Data Flow Diagram:



SYSTEM SECURITY

The Online Quiz System app requires robust security features to protect user data, ensure secure transactions, and maintain overall system integrity. This section outlines the essential security measures to be implemented in the system.

➢ Security Features:

1. Authentication and Authorization

- **OAuth2 Implementation: Secure user authentication using OAuth2 to ensure safe access to the system.**
- **Role-Based Access Control (RBAC): Restrict actions based on roles (e.g., students, teachers, admins) to prevent unauthorized modifications.**

2. Data Encryption

- **HTTPS Enforcement: Encrypt data in transit between the client and server using SSL/TLS.**
- **Encryption at Rest: Protect sensitive stored data (e.g., user credentials, quiz results) with AES-256 encryption.**

3. Secure Communication

- **JWT Tokens: Use JSON Web Tokens (JWT) for secure, stateless authentication between client and server.**
- **Secure WebSocket (WSS): Encrypt real-time quiz interactions to prevent data interception.**

4. Input Validation and Sanitization

- **Client-Side Validation: Reduce server load by validating inputs before submission.**
- **Server-Side Validation: Prevent SQL injection, XSS (Cross-Site Scripting), and CSRF (Cross-Site Request Forgery) by sanitizing all inputs.**

5. Database Security

- **Parameterized Queries: Use prepared statements to prevent SQL injection.**
- **Access Controls: Enforce least privilege access to database resources, limiting access per user role.**

6. Logging and Monitoring

- **Activity Logs: Maintain detailed logs of quiz attempts, login attempts, and admin actions for auditing.**
- **Intrusion Detection Systems (IDS): Monitor traffic for malicious activity and potential breaches.**

7. Regular Security Audits

- **Penetration Testing: Conduct periodic tests to identify vulnerabilities and fix security gaps.**
- **Code Reviews: Ensure adherence to secure coding practices by reviewing the codebase regularly.**

8. Backup and Recovery

- **Regular Data Backups: Store encrypted backups in secure cloud storage to prevent data loss.**
- **Disaster Recovery Plan: Maintain a failover strategy to restore services in case of system failures.**

Language Support:

The **Online Quiz System** is developed using **Javascript and React** for the front end, ensuring a **scalable and high-performance web application**. The back end is built using **Node.js with Express.js**, providing an efficient and asynchronous environment for handling quiz operations.

What Is React-JS?

React JS is a popular JavaScript framework used to build mobile applications. It allows developers to create cross-platform apps using a single codebase, leveraging the power of React for dynamic and efficient UI development.

Code Behind in React-JS

In React Native, the logic of the app can be separated from the UI components using a pattern like the code-behind method. This involves organizing the code in separate files where the business logic is managed independently from the presentation layer. This enhances code readability, maintainability, and modularity.

Connections:

Connections in the Online Quiz System app is managed using libraries like Axios or Fetch API to communicate with the backend services. These connections are used to send requests and receive responses, interacting with the server to fetch or update data.

Commands:

Commands in the Coffeehouse app, like database commands, are actions that interact with the backend services. These commands can be POST, GET, PUT, or DELETE requests that handle various operations such as adding a new order, updating user information, or retrieving menu items. The responses are then processed and displayed within the app's UI components.

CODING

➢ App.jsx:

```
➢ import { BrowserRouter as Router, Routes, Route, Navigate } from "react-
  router-dom";
➢ import { useState, useEffect } from "react";
➢ import { auth } from "../firebaseConfig";
➢ import { onAuthStateChanged } from "firebase/auth";
➢ import { getFirestore } from "firebase/firestore"; // ✅ Import Firestore
➢ import { AuthProvider } from "./context/authContext";
➢ import Navbar from "./components/Navbar";
➢ import Home from "./pages/Home";
➢ import Login from "./pages/Login";
➢ import Register from "./pages/Register";
➢ import Quiz from "./pages/Quiz";
➢ import Leaderboard from "./pages/Leaderboard";
➢ import Lobby from "./pages/Lobby";
➢ import GameRoom from "./pages/GameRoom";
➢ import "./styles/global.css";
➢
➢ function App() {
➢   const [user, setUser] = useState(null);
➢   const [loading, setLoading] = useState(true);
➢   const db = getFirestore(); // ✅ Initialize Firestore here
➢
➢   useEffect(() => {
➢     const unsubscribe = onAuthStateChanged(auth, (currentUser) => {
➢       setUser(currentUser);
➢       setLoading(false);
➢     });
➢     return () => unsubscribe();
➢   }, []);
➢
➢   if (loading) {
➢     return <div className="loading-screen">Loading...</div>;
➢   }
➢
➢   return (
➢     <AuthProvider>
➢       <Router>
➢         <Navbar user={user} />
➢         <Routes>
```

```
            <Route path="/" element={user ? <Home /> : <Navigate to="/login" />} />
            <Route path="/quiz" element={user ? <Quiz db={db} user={user} /> : <Navigate to="/login" />} />
            <Route path="/leaderboard" element={<Leaderboard db={db}/>} />
            <Route path="/lobby" element={user ? <Lobby db={db} user={user} /> : <Navigate to="/login" />} />
            <Route path="/gameroom/:roomId" element={user ? <GameRoom db={db} user={user} /> : <Navigate to="/login" />} />

            <Route path="/login" element={user ? <Navigate to="/" /> : <Login />} />
            <Route path="/register" element={user ? <Navigate to="/" /> : <Register />} />
          </Routes>
        </Router>
      </AuthProvider>
    );
  }

  export default App;
```

➢ **Src/Component/ Navbar.jsx:**

```
import { useState, useEffect } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { signOut } from 'firebase/auth';
import { auth, db } from '../../firebaseConfig';
import { doc, getDoc, onSnapshot } from 'firebase/firestore';
import '../styles/navbar.css';

const Navbar = ({ user }) => {
  const [dropdownOpen, setDropdownOpen] = useState(false);
  const [favoriteCategory, setFavoriteCategory] = useState('');
  const [username, setUsername] = useState('');
  const [highestScore, setHighestScore] = useState('None');
  const navigate = useNavigate();

  useEffect(() => {
    let unsubscribe;

    if (user) {
      const userRef = doc(db, "users", user.uid);
```

```
    const fetchUserData = async () => {
      try {
        const userSnap = await getDoc(userRef);
        if (userSnap.exists()) {
          const userData = userSnap.data();
          setUsername(userData.username || "None");
          setFavoriteCategory(userData.favoriteCategory || "None");
          setHighestScore(userData.highestScore || "None");
        } else {
          console.warn("User document not found in Firestore.");
        }
      } catch (error) {
        console.error("Error fetching user data:", error);
      }
    };


    fetchUserData();


    unsubscribe = onSnapshot(userRef, (docSnap) => {
      if (docSnap.exists()) {
        const updatedData = docSnap.data();
        setUsername(updatedData.username || "None");
        setFavoriteCategory(updatedData.favoriteCategory || "None");
        setHighestScore(updatedData.highestScore || "None");
      }
    });
  }

  return () => unsubscribe && unsubscribe();
}, [user]);

const toggleDropdown = () => setDropdownOpen(!dropdownOpen);

const handleLogout = async () => {
  try {
    await signOut(auth);
    navigate('/login');
  } catch (error) {
    console.error("Logout error:", error);
  }
};

return (
```

```
    <nav className="navbar">
      <div className="nav-links">
        <Link to="/">Home</Link>
        <Link to="/quiz">Play Quiz</Link>
        <Link to="/leaderboard">Leaderboard</Link>
        <Link to="/lobby">Multiplayer</Link>

        <div className="profile-section">
          {user ? (
            <div className="profile-menu">
              <button className="profile-btn" onClick={toggleDropdown}>
                <img src={user.photoURL || '/default-avatar.png'} alt="Profile"
className="profile-avatar" />
                <span className='profile-text'>Profile</span>
              </button>

              {dropdownOpen && (
                <div className="dropdown-menu">
                  <ul>
                    <li>Username: {username}</li>
                    <li onClick={handleLogout}>Logout</li>
                  </ul>
                </div>
              )}
            </div>
          ) : (
            <div className="auth-links">
              <Link to="/login">Login</Link>
              <Link to="/register">Register</Link>
            </div>
          )}
        </div>
      </div>
    </nav>
  );
};

export default Navbar;
```

- **Src/Component/ Home.jsx:**

```
import { Link, useNavigate } from 'react-router-dom';
import '../styles/home.css';
```

```
➢
➢  const categories = [
➢    { id: 9, name: 'General Knowledge' },
➢    { id: 21, name: 'Sports' },
➢    { id: 23, name: 'History' },
➢    { id: 17, name: 'Science & Nature' },
➢    { id: 11, name: 'Movies' },
➢  ];
➢
➢  const Home = () => {
➢      const navigate = useNavigate();
➢
➢      const handleCategorySelect = (categoryId) => {
➢          navigate(`/quiz?category=${categoryId}`);
➢      };
➢
➢      return (
➢        <div className="home-container">
➢          <h1>Welcome to the Online Quiz System</h1>
➢          <p>Test your knowledge and compete with others!</p>
➢
➢          {/* Category Selection Buttons */}
➢          <div className="category-buttons">
➢            {categories.map((cat) => (
➢              <button key={cat.id} onClick={() =>
   handleCategorySelect(cat.id)} className="category-btn">
➢                {cat.name}
➢              </button>
➢            ))}
➢          </div>
➢
➢          <Link to="/leaderboard" className="leaderboard-btn">View
   Leaderboard</Link>
➢        </div>
➢      );
➢  };
➢
➢  export default Home;
➢
```

➢ Src/Component/ Quiz.jsx:

```
➢  import { useState, useEffect } from 'react';
➢  import { useSearchParams, useNavigate } from 'react-router-dom';
```

```
import { fetchQuizQuestions } from '../api/quizAPI';
import { doc, getDoc, setDoc, updateDoc } from 'firebase/firestore';
import '../styles/quiz.css';

const decodeEntities = (html) => {
  const txt = document.createElement("textarea");
  txt.innerHTML = html;
  return txt.value;
};

const Quiz = ({ db, user }) => {
  const [questions, setQuestions] = useState([]);
  const [currentQuestion, setCurrentQuestion] = useState(0);
  const [score, setScore] = useState(0);
  const [loading, setLoading] = useState(true);
  const [showPopup, setShowPopup] = useState(false); //  ◇  State for the
modal
  const [searchParams] = useSearchParams();
  const navigate = useNavigate(); //  ◇  Used for navigation

  const category = searchParams.get('category') || 'Random';

  useEffect(() => {
    const loadQuestions = async () => {
      setLoading(true);
      const quizData = await fetchQuizQuestions(10, category);
      if (quizData.length > 0) {
        const formattedQuestions = quizData.map(q => ({
          question: decodeEntities(q.question),
          answers: shuffleArray([...q.incorrect_answers,
q.correct_answer].map(decodeEntities)),
          correctAnswer: decodeEntities(q.correct_answer)
        }));
        setQuestions(formattedQuestions);
      }
      setLoading(false);
    };

    loadQuestions();
  }, [category]);

  const shuffleArray = (array) => array.sort(() => Math.random() - 0.5);

  const handleAnswer = (selectedAnswer) => {
    if (selectedAnswer === questions[currentQuestion].correctAnswer) {
```

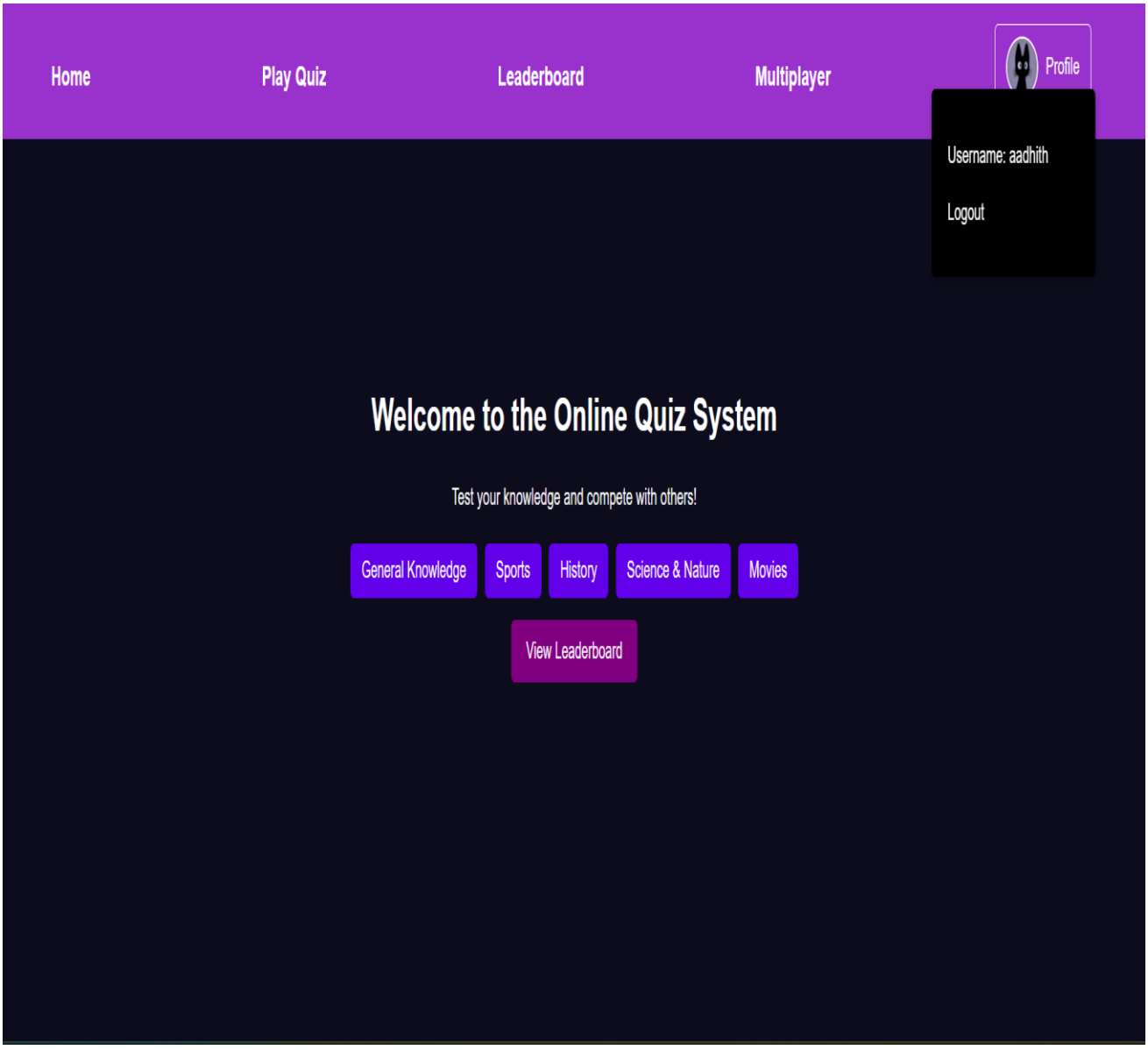
```
        setScore(score + 1);
      }

      if (currentQuestion < questions.length - 1) {
        setCurrentQuestion(currentQuestion + 1);
      } else {
        handleQuizCompletion();
      }
    };

    const handleQuizCompletion = async () => {
      setShowPopup(true); // Show modal

      if (user) {
        const userRef = doc(db, "users", user.uid);
        const userSnap = await getDoc(userRef);

        let username = "Anonymous"; // Default name

        if (userSnap.exists()) {
          const userData = userSnap.data();
          username = userData.username || "Anonymous"; // Get username from
   Firestore
        }

        const scoreRef = doc(db, "scores", user.uid);
        const scoreSnap = await getDoc(scoreRef);

        if (scoreSnap.exists()) {
          const scoreData = scoreSnap.data();
          await updateDoc(scoreRef, {
            score: scoreData.score + score, // 🔥 Add latest score to total
            gamesPlayed: (scoreData.gamesPlayed || 0) + 1,
            username, // Ensure correct username is stored
          });
        } else {
          await setDoc(scoreRef, {
            username,
            score: score,
            gamesPlayed: 1,
          });
        }
      }
    };

```

```
  // ◇ Restart Quiz
  const restartQuiz = () => {
    setScore(0);
    setCurrentQuestion(0);
    setShowPopup(false);
  };

  // ◇ Return to Home
  const returnHome = () => {
    navigate('/'); // Redirects to home page
  };

  return (
    <div className="quiz-container">
      {loading ? (
        <p>⏳ Loading questions...</p>
      ) : questions.length > 0 ? (
        <div>
          <h1>Quiz - {category}</h1>
          <h2>{questions[currentQuestion].question}</h2>
          <div className="answers">
            {questions[currentQuestion].answers.map((answer, index) => (
              <button key={index} className="answer" onClick={() =>
handleAnswer(answer)}>
                {answer}
              </button>
            ))}
          </div>
        </div>
      ) : (
        <p>⚠ No questions available. Try again later.</p>
      )}

      {/* ◇ Pop-up Modal */}
      {showPopup && (
        <div className="popup-overlay">
          <div className="popup-content">
            <h2>Quiz Completed 🎉</h2>
            <p>Your Score: {score} / 10</p>
            <button className="btn-restart"
onClick={restartQuiz}>Restart</button>
            <button className="btn-home" onClick={returnHome}>Return
Home</button>
          </div>
        </div>
```

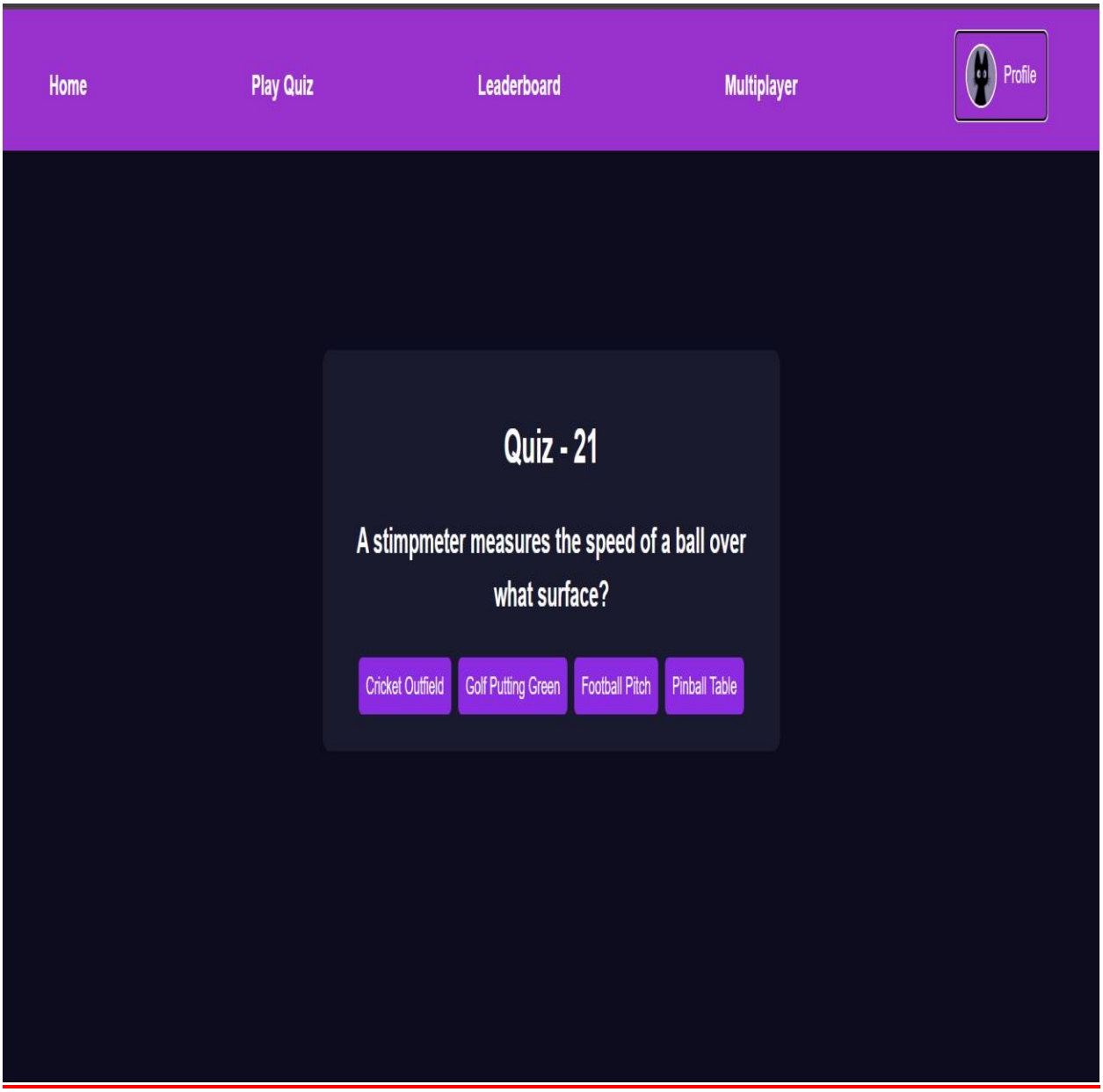
```
➢         )}
➢       </div>
➢     );
➢   };
➢
➢   export default Quiz;
➢
```

OUTPUT SCREENS

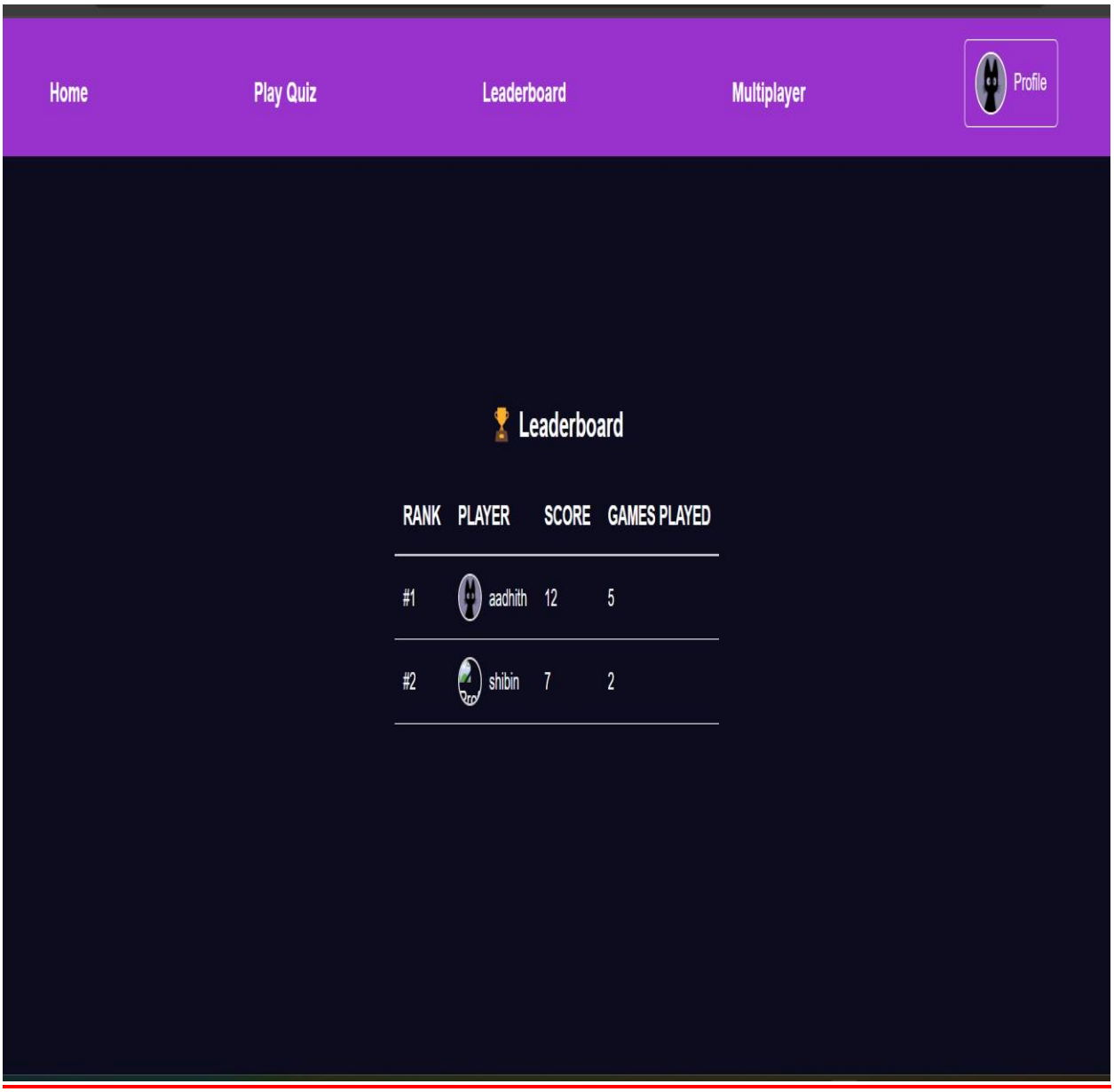
- **Home Page:**



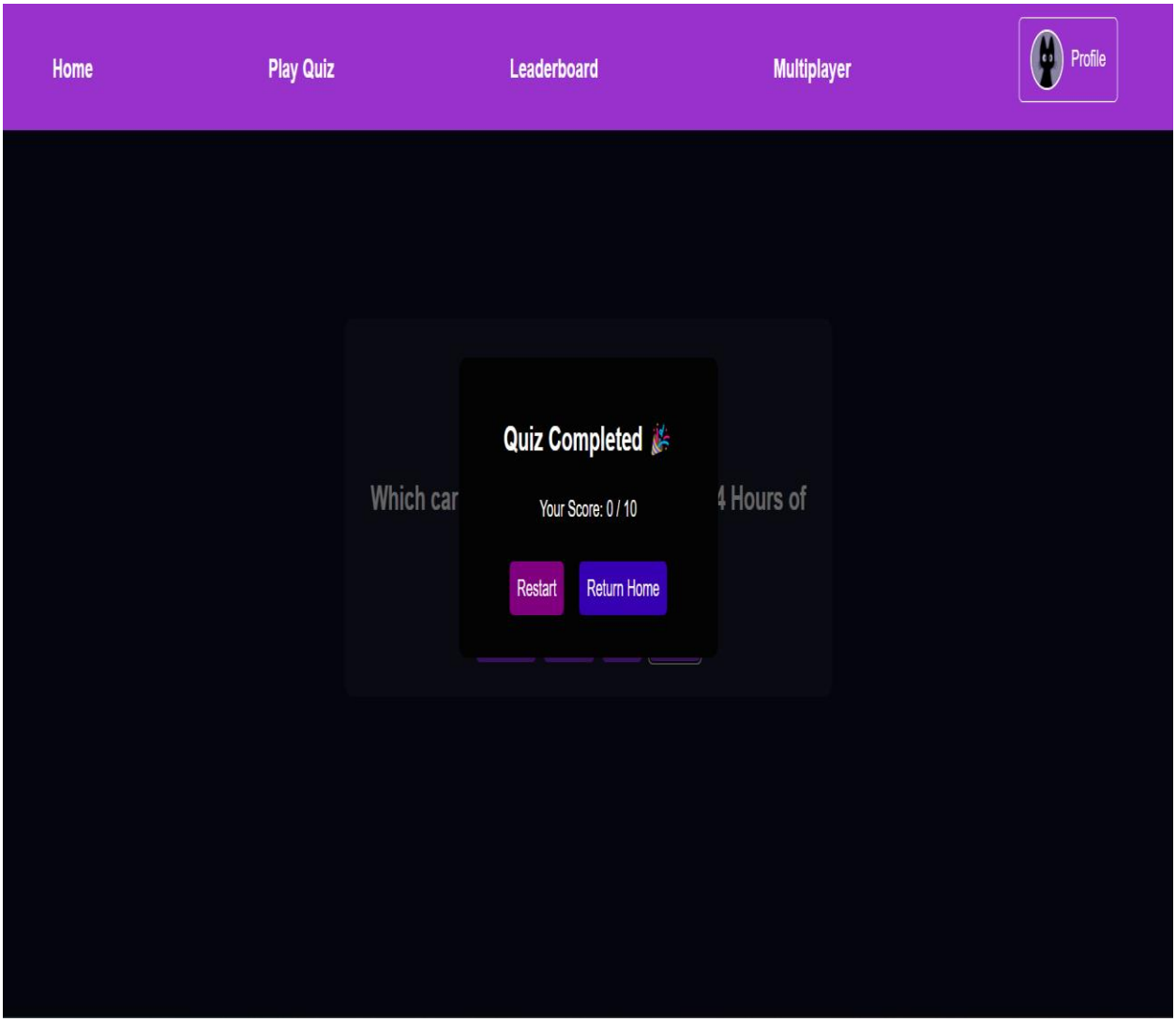
➢ Quiz Page:


Home
Play Quiz
Leaderboard
Multiplayer
Profile
Quiz - 21
A stimpmeter measures the speed of a ball over what surface?
Cricket Outfield
Golf Putting Green
Football Pitch
Pinball Table

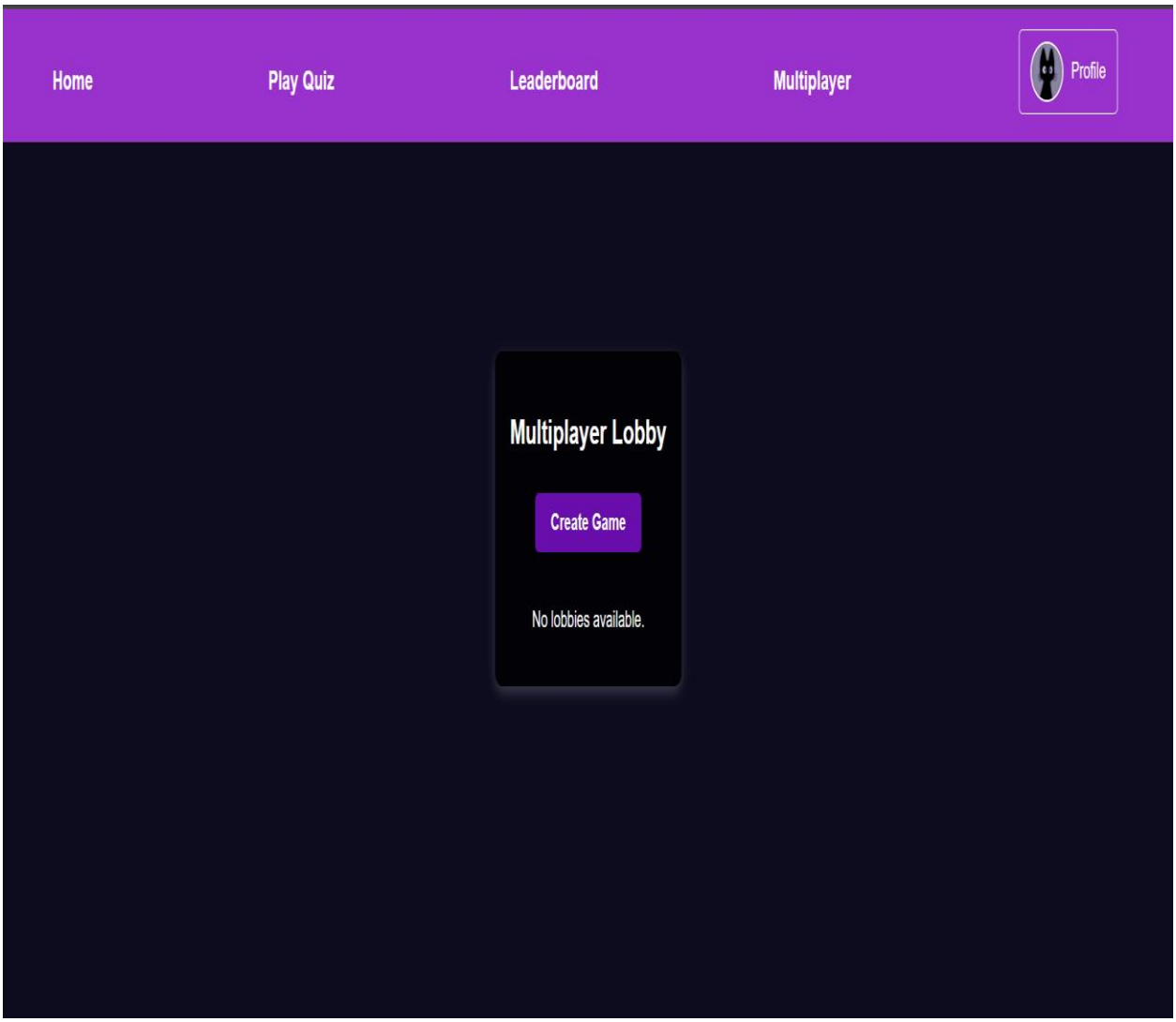
➢ LeaderBoard Page:



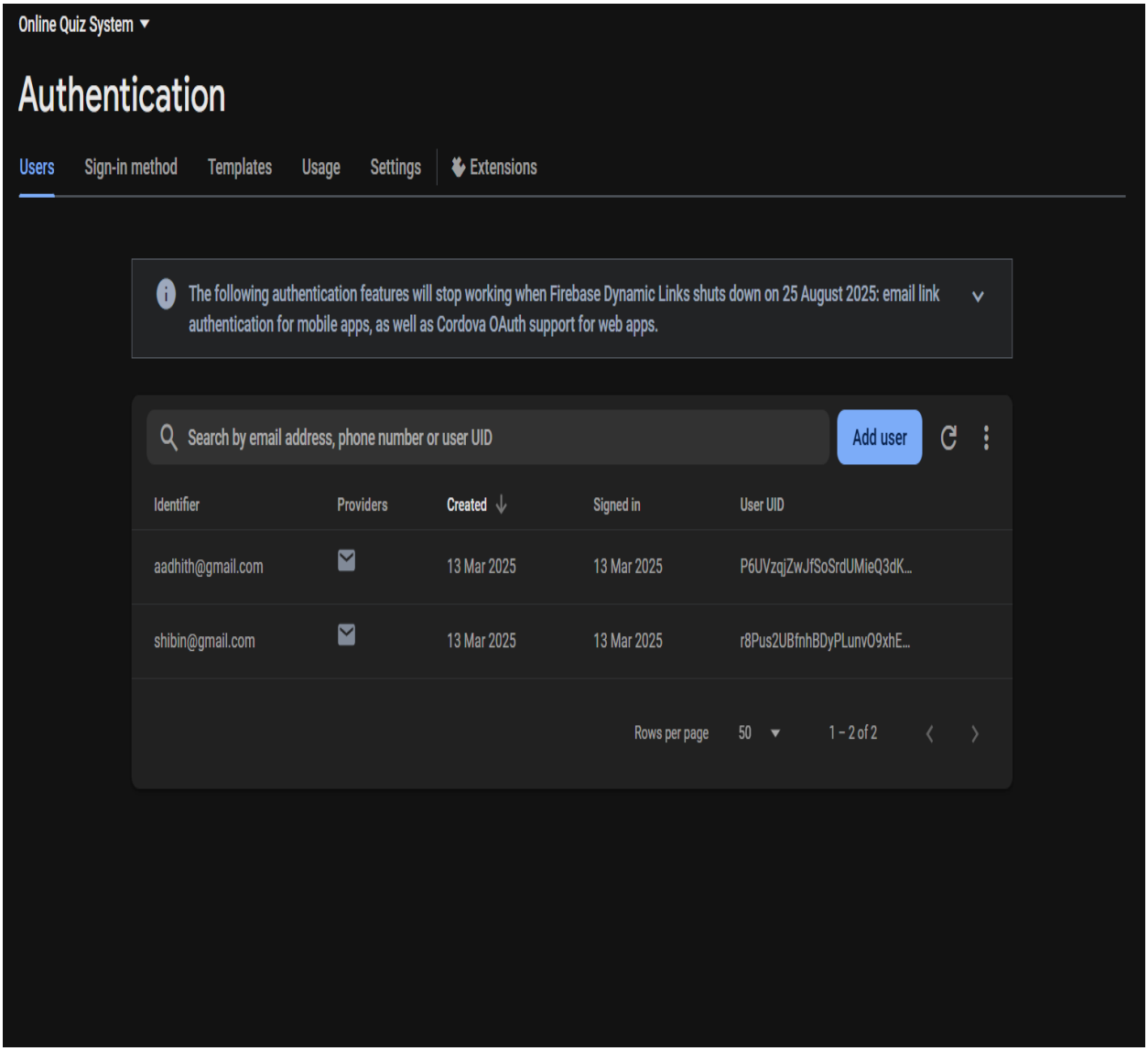
➢ Quiz Result Pop Up:



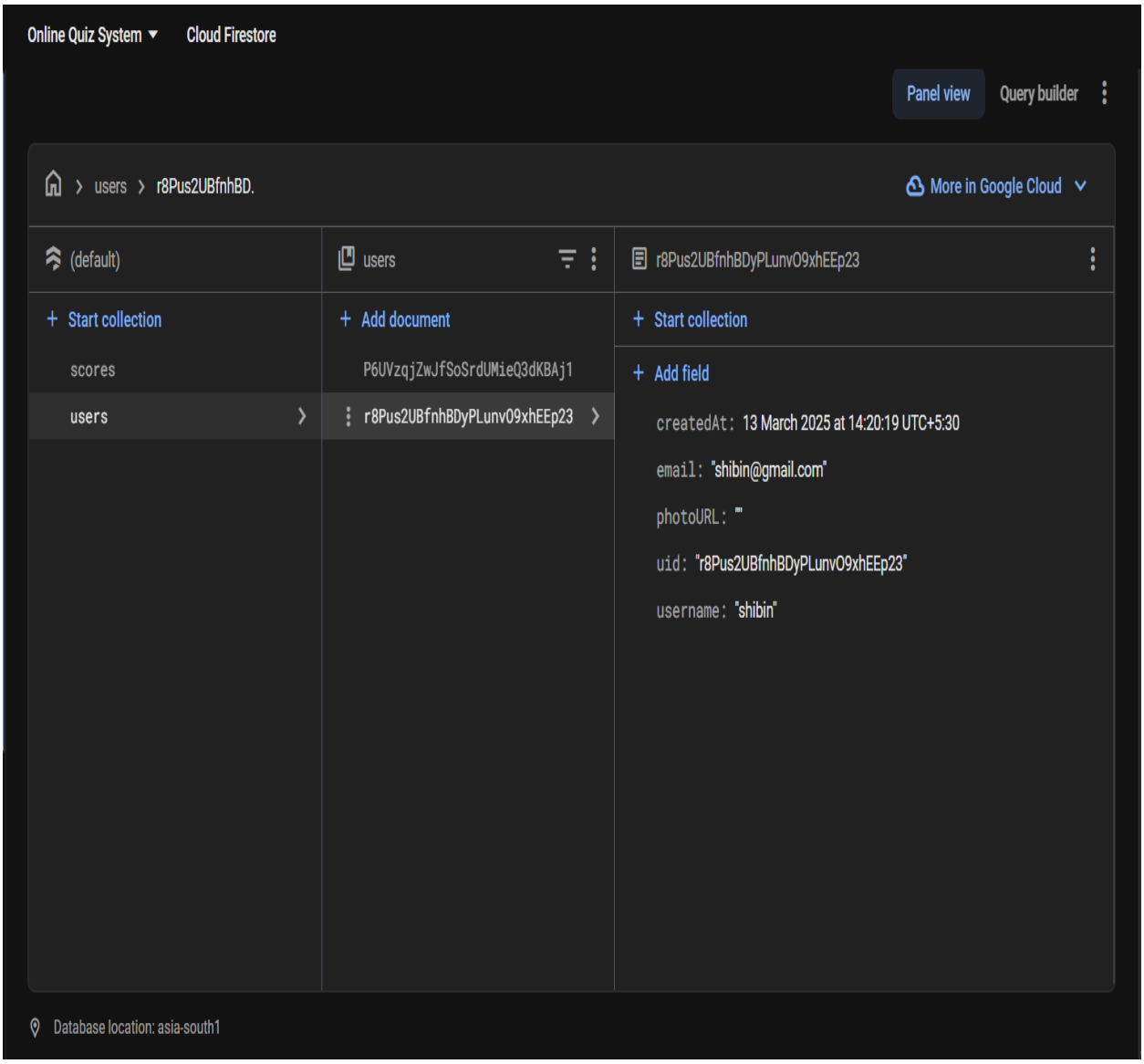
➢ Multiplayer Page:



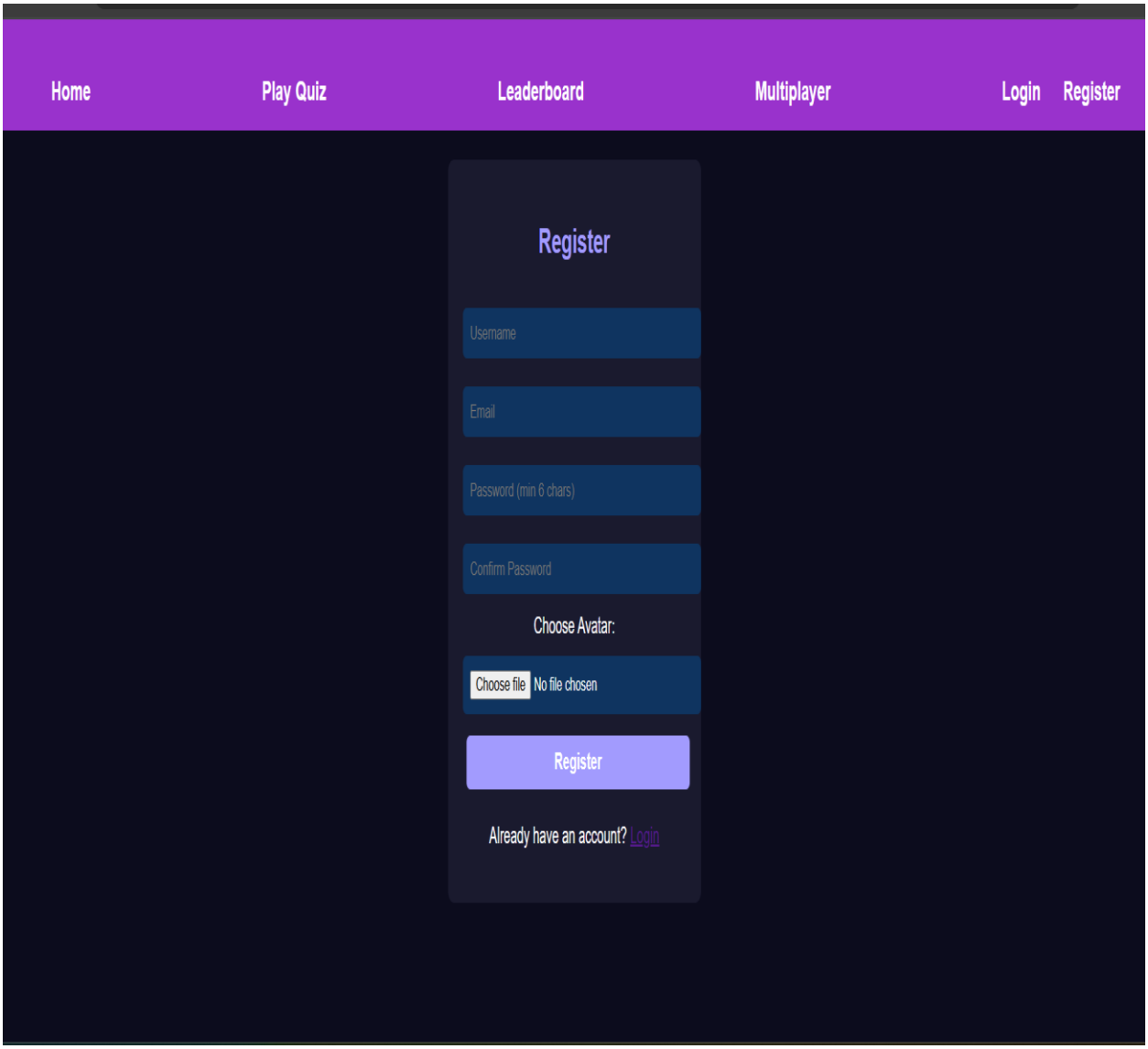
➢ Firebase Authentication Database:



➢ Firebase User Database:



➢ Registration Page:



CONCLUSION

The **Online Quiz System** is a comprehensive solution designed to enhance the quiz-taking and assessment experience. Through meticulous **software engineering practices**, we have ensured that the system is **robust, scalable, and user-friendly** for both administrators and participants.

The architecture leverages modern technologies like **JavaScript, React, Node.js, and Firebase**, ensuring a seamless and efficient performance across multiple platforms. By integrating **security best practices, modular design, and a structured development lifecycle**, the system is well-equipped to handle large-scale quiz operations with **high reliability and security**.

In conclusion, the **Online Quiz System** is positioned to effectively meet educational and assessment needs, delivering a **seamless, secure, and interactive** quiz experience. With **continuous updates, testing, and user feedback integration**, the system will remain adaptable to evolving requirements and technological advancements, ensuring long-term success.

FUTURE ENHANCEMENT

The **Online Quiz System** has vast potential for future enhancements to further improve user experience and engagement. One key area of expansion is the **introduction of multiplayer quiz modes**, allowing users to **compete in real-time quiz battles, host live quiz events, and engage in team-based competitions**. This will foster a more **interactive and engaging** learning environment.

Additional improvements include **profile customization**, enabling users to **edit their profiles, set avatars, track quiz progress, and earn achievements**. Expanding the **quiz hosting capabilities** by allowing **custom quiz themes, scheduling options, and automated reminders** will enhance flexibility. Furthermore, integrating **AI-driven question recommendations, enhanced analytics, voice-based interactions, and support for wearable devices** will keep the platform innovative and future-proof.

BIBLIOGRAPHY

➢ Web Resources:

❖ For React JS Documentation: https://react.dev/

❖ For JavaScript Documentation: https://developer.mozilla.org/en-US/docs/Web/JavaScript

❖ For Android Studio: https://developer.android.com/studio

❖ For Visual Studio Code: https://code.visualstudio.com/