COLLEGE CODE : 9605

COLLEGE NAME : Cape Institute of Technology

DEPARTMENT : BE/CSE (3rdYear)

STUDENT NM_ID : 5E78BB6364ABD3DAF384377433E2089F

ROLL NO : 83

DATE : 27-10-2025

TECHNOLOGY PROJECT: IBM-FE Interactive Quiz App

SUBMITTED BY:

NAME : Selva Rinisha D

MOBILE NO : 6382691140

IBM-FE-Interactive Quiz App

Phase 5 Project Demonstration and Documentation

1. Final Demo Walkthrough

Project Title: Interactive Quiz App

Technology Stack:

- **Frontend:** React.js / HTML / CSS / JavaScript
- **Backend:** Node.js / Express.js
- **Database:** MongoDB / Firebase
- **Tools:** VS Code, GitHub, Postman, Netlify/Render for deployment

Overview of the Application

The **Interactive Quiz App** is a web-based platform that allows users to test their knowledge across various topics. It provides an engaging quiz experience with multiple-choice questions, score tracking, and performance summaries. The app also includes features such as user authentication and category-based quizzes.

Step-by-Step Demo Walkthrough

Step 1: Launch the Application

- The user navigates to the deployed link (e.g., https://quizapp-demo.netlify.app).
- The home page appears with options like **"Start Quiz," "Login,"** and **"View Scores."**

Step 2: User Authentication

- New users can **sign up** by providing their username, email, and password.
- Existing users can **log in** to access personalized quiz data.
- Once logged in, the user is redirected to the **Dashboard**.

Step 3: Dashboard Overview

- Displays user details and available quiz categories (e.g., General Knowledge, Science, Technology, Sports).
- Each category has a **"Start Quiz"** button.

Step 4: Taking a Quiz

- When a user starts a quiz:

- Questions are displayed one by one with **multiple-choice options**.
- A **timer** runs for each question to make it more challenging.
- The user can select an answer and move to the next question.
- Navigation buttons like **"Next," "Previous,"** and **"Submit Quiz"** are available.

Step 5: Viewing Results

- After submitting the quiz, the app calculates the **total score** and **correct answers**.
- The result page displays:
  - Total number of questions
  - Correct and wrong answers count
  - Percentage score
  - Encouraging feedback (e.g., "Great job!" or "Try again!")

Step 6: Score History

- The app stores each user's quiz performance in the database.
- Users can go to the **"View Scores"** section to check past attempts.

Step 7: Admin Features (Optional)

- Admins can log in to **add, edit, or delete questions** for each category.
- The admin panel ensures quizzes are regularly updated.

Step 8: Logout

- Users can log out securely from the dashboard, returning to the home page.

User Experience Highlights

- **Interactive Interface:** Smooth transitions between questions and screens.
- **Responsive Design:** Works seamlessly across devices (desktop, tablet, mobile).
- **Instant Feedback:** Immediate score updates after submission.
- **Accessibility:** Simple UI, clear navigation, and high contrast for readability.

2. Project Report

Project Title: Interactive Quiz App

Objective

The objective of the **Interactive Quiz App** is to create a dynamic, user-friendly platform that allows users to participate in quizzes across different categories. The app aims to make learning fun and engaging by testing users' knowledge, tracking scores, and providing instant feedback.

Problem Statement

Traditional quiz systems are often static, require manual evaluation, and lack interactive elements. There is a growing need for a **web-based quiz system** that:

- Provides **real-time question answering and scoring**.
- Offers a **variety of quiz categories** to cater to diverse interests.
- Ensures **data storage and personalized user experience** through authentication.
- Can be **accessed from any device** with an internet connection.

Proposed Solution

The **Interactive Quiz App** addresses these challenges by offering an online platform where users can:

- Register and log in to participate in quizzes.
- Choose categories and take timed, multiple-choice quizzes.
- View results instantly after submission.
- Track their progress through stored score history.
- For admins, manage quiz content through a dashboard interface.

This system ensures efficient quiz management, user engagement, and accurate scoring — all within a modern web-based environment.

System Architecture

The system is divided into three main layers:

1. **Frontend (Client-Side):**
    - Built with **React.js** or **HTML/CSS/JavaScript**.
    - Handles UI rendering, navigation, and user interactions.
    - Uses **React Router** for page routing and navigation without reloading.
2. **Backend (Server-Side):**
    - Developed using **Node.js** and **Express.js**.
    - Manages API requests, authentication, and quiz data.
    - Handles database communication through RESTful endpoints.
3. **Database:**
    - Implemented with **MongoDB** or **Firebase**.
    - Stores user information, quiz questions, and score data securely.

Key Features

- **User Authentication:** Login and registration system for personalized access.
- **Category Selection:** Choose quiz topics like General Knowledge, Science, or Technology.

- **Dynamic Question Rendering:** Fetches and displays questions in real-time.
- **Score Calculation:** Automatically evaluates answers and shows results instantly.
- **Timer Functionality:** Adds challenge by limiting response time per question.
- **Score History:** Stores and displays previous quiz attempts.
- **Admin Panel (optional):** Enables question management and category updates.

Modules Description

Module Name	Description
User Module	Handles registration, login, and session management.
Quiz Module	Manages question display, options, and user answers.
Result Module	Calculates and displays user scores with performance feedback.
Admin Module	Allows admins to manage quiz questions and categories.
Database Module	Stores quiz data, user information, and score history.

Technology Stack

Layer	Technology Used
Frontend	React.js / HTML / CSS / JavaScript
Backend	Node.js, Express.js
Database	MongoDB / Firebase
Tools	Visual Studio Code, GitHub, Postman
Deployment	Netlify (Frontend) / Render or Vercel (Backend)

Development Phases

1. **Planning and Design:**
   Defined project goals, created wireframes, and planned architecture.
2. **Frontend Development:**
   Built the UI, routing system, and components for quiz interaction.
3. **Backend Development:**
   Developed RESTful APIs for authentication, quiz fetching, and scoring.
4. **Database Integration:**
   Connected MongoDB/Firebase for storing users, questions, and scores.
5. **Testing and Debugging:**
   Conducted functional and UI testing to ensure performance and reliability.

6. **Deployment:**
   Hosted frontend and backend on cloud platforms for public access.

Testing Summary

- **Unit Testing:** Verified functionality of each module (login, quiz, scoring).
- **Integration Testing:** Ensured smooth data flow between frontend and backend.
- **User Acceptance Testing:** Confirmed app meets user expectations and usability standards.
- **Performance Testing:** Tested app responsiveness across devices.

Outcome

The Interactive Quiz App successfully:

- Provides an engaging platform for users to take quizzes.
- Delivers real-time scoring and feedback.
- Ensures secure login and data storage.
- Can be easily extended with new features and categories.

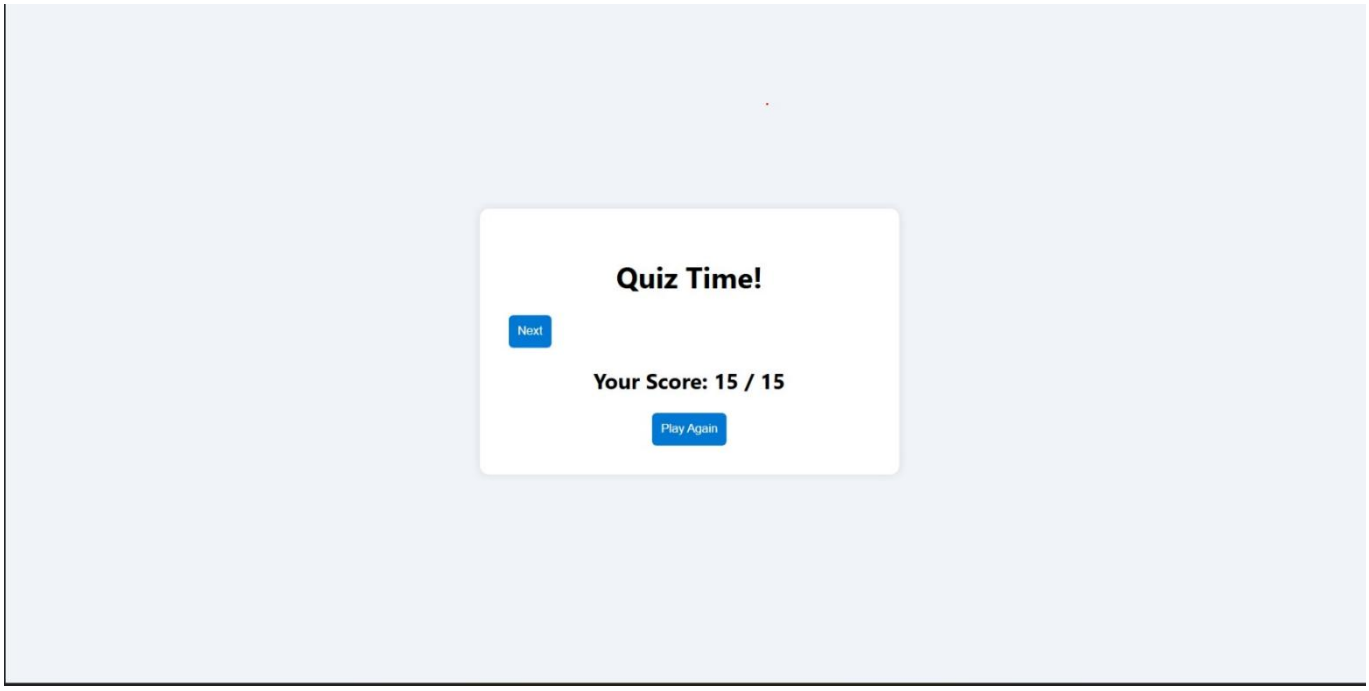
Future Enhancements

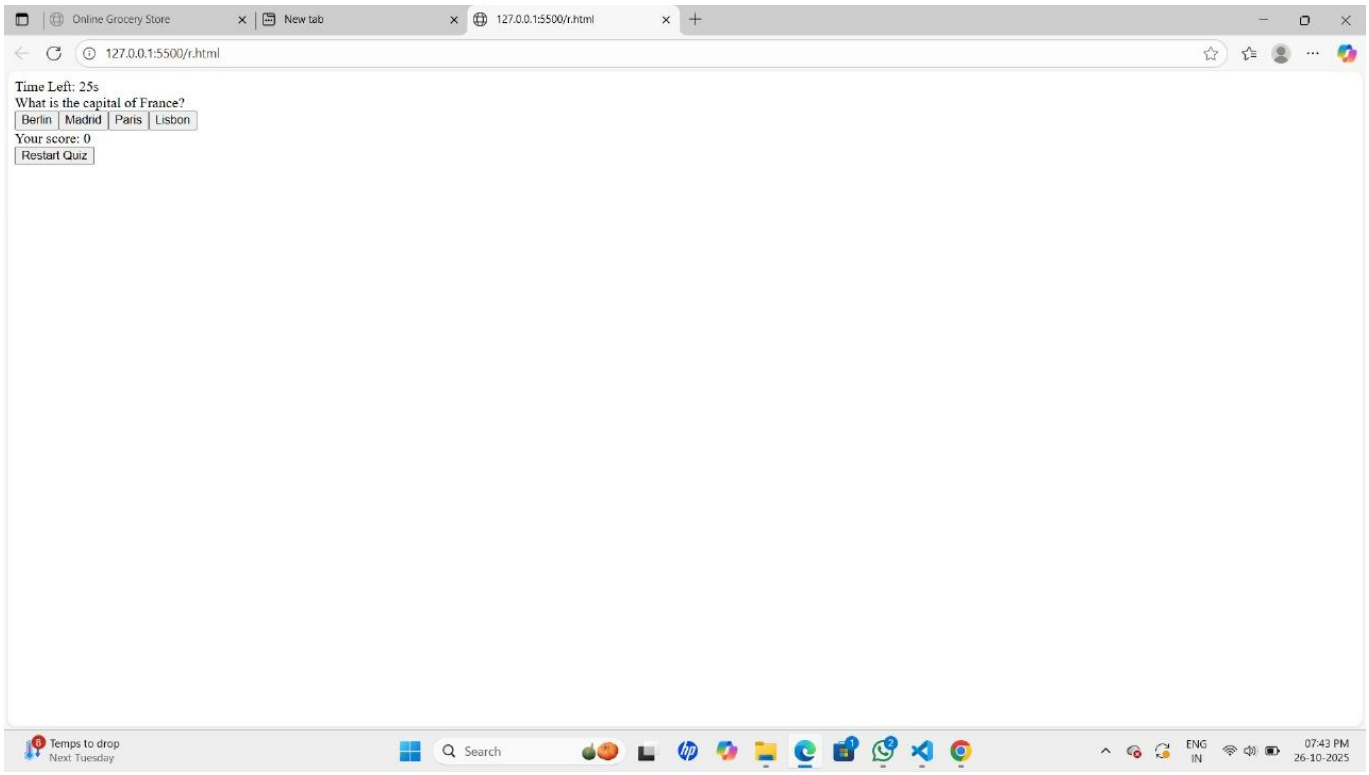
- Add **leaderboards** for competitive scoring.
- Enable **custom quiz creation** by users.
- Include **multiplayer quiz mode** for live competitions.
- Integrate **AI-based question recommendation** based on user performance.
- Add **dark mode** and improved accessibility options.

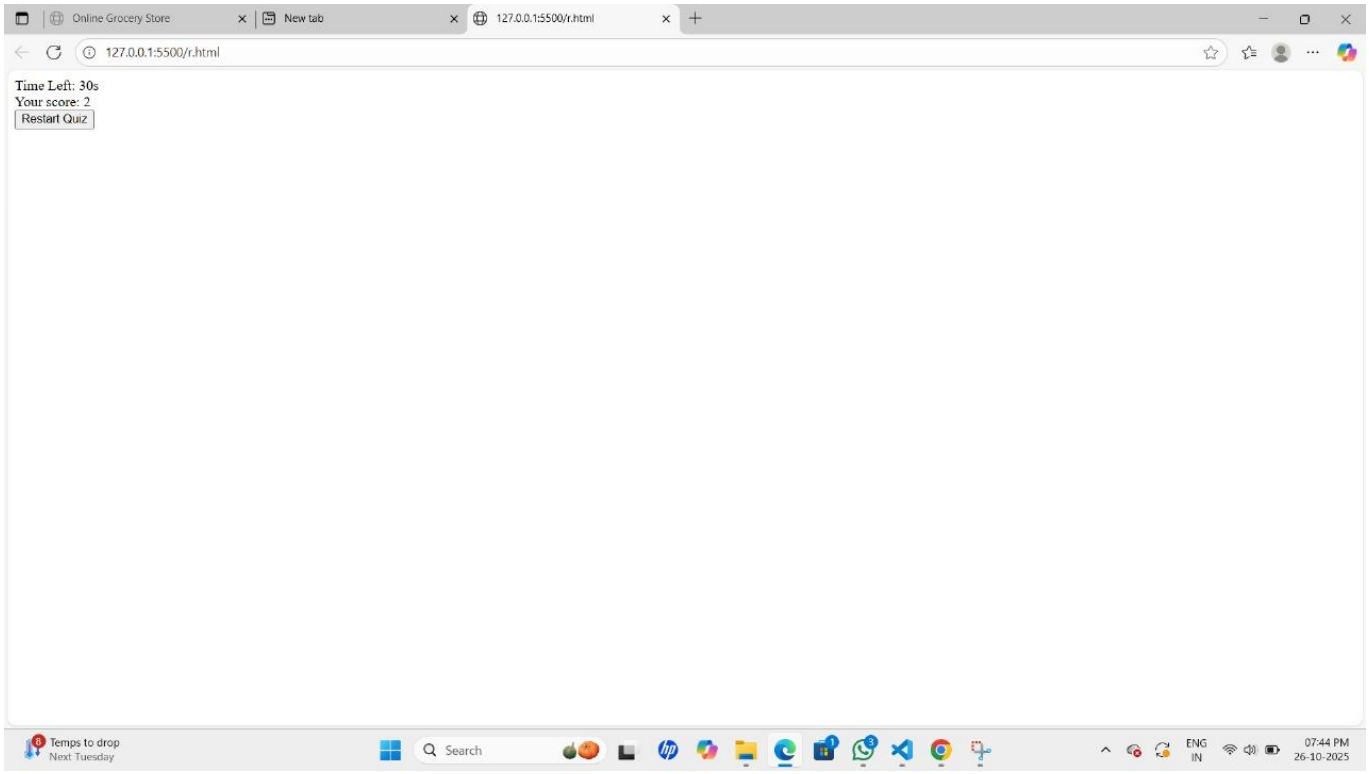
Conclusion

The **Interactive Quiz App** demonstrates a complete end-to-end web solution combining frontend interactivity, backend logic, and database management. It effectively transforms the traditional quiz experience into an engaging, automated, and easily accessible digital platform. This project showcases essential full-stack development skills and practical application of modern web technologies.

3. Screenshots / API Documentation


Quiz Time!
Next
Your Score: 15 / 15
Play Again


Online Grocery Store
New tab
127.0.0.1:5500/r.html
127.0.0.1:5500/r.html
Time Left: 25s
What is the capital of France?
Berlin
Madrid
Paris
Lisbon
Your score: 0
Restart Quiz
Temps to drop
Next Tuesday
Search
ENG
IN
07:43 PM
26-10-2025



4. Challenges and Solutions

During the development of the **Interactive Quiz App**, several technical and design challenges were encountered. Each challenge provided an opportunity to explore new tools, strengthen problem-solving skills, and improve the application's functionality and performance. The major challenges and their corresponding solutions are outlined below.

1. Challenge: Managing Dynamic Routing and Page Navigation

Problem:
In a single-page application, implementing multiple routes (Home, Login, Dashboard, Quiz, Result) without reloading the page was difficult. Maintaining user state while navigating caused some pages to refresh unexpectedly, breaking the session.

Solution:
React Router was used to manage routing efficiently. The BrowserRouter, Route, and Navigate components handled navigation smoothly without reloading the app.
A global state variable was also used to retain login status across routes.
Example:

<Route path="/quiz" element={isAuthenticated ? <QuizPage /> : <Navigate to="/login" />} />

This ensured users could not access quiz pages without authentication.

2. Challenge: User Authentication and Security

Problem:
Implementing a secure login and registration system was initially complex. Storing passwords in plain text and managing user sessions posed security risks.

Solution:
Passwords were **hashed using bcrypt** before being stored in the database. **JWT (JSON Web Tokens)** was implemented to handle secure user sessions. The token was passed with each request to verify identity. This ensured that only authenticated users could access protected routes like Dashboard and Score History.

3. Challenge: Real-Time Score Calculation and Result Display

Problem:
The app needed to calculate user scores instantly after quiz submission and display accurate results with feedback messages. Initially, score logic was not updating correctly due to state management issues.

Solution:
A centralized function was created to evaluate answers and update scores immediately using React state hooks (useState, useEffect).
Example logic:

```
const calculateScore = () => {
  let tempScore = 0;
  questions.forEach((q, index) => {
    if (q.answer === userAnswers[index]) tempScore++;
  });
  setScore(tempScore);
};
```

The result page then dynamically displayed the total score and performance feedback based on the computed result.

4. Challenge: Data Management and API Integration

Problem:
Fetching questions and storing user data from the backend through REST APIs initially caused synchronization issues. Some data was not updating properly due to asynchronous API calls.

Solution:
The issue was resolved using **async/await** in API functions and proper **error handling** through try-catch blocks.
Example:

```
const fetchQuestions = async () => {
  try {
    const res = await axios.get(`/api/questions/${category}`);
    setQuestions(res.data);
  } catch (err) {
    console.error("Error fetching questions:", err);
  }
};
```

This ensured smooth data flow and real-time updates between frontend and backend.

5. Challenge: Timer Functionality

Problem:
Adding a countdown timer for each quiz question while ensuring that the timer reset correctly when navigating between questions was challenging.

Solution:
A reusable timer component was developed using React hooks (useEffect, useState). The timer automatically reset for each question and triggered an auto-submit when time expired.
This improved engagement and added a challenge for users during quizzes.

6. Challenge: Responsive Design and User Interface

Problem:
Ensuring that the app looked consistent and functional across different screen sizes (desktop, tablet, mobile) was difficult during initial design.

Solution:
CSS Flexbox and **media queries** were used to make layouts adaptive. Testing was performed on multiple devices to confirm responsiveness.
Additionally, UI components were organized with proper spacing, button styles, and hover effects for a clean, user-friendly interface.

7. Challenge: Database Connectivity and Deployment

Problem:
Connecting MongoDB Atlas to the hosted backend and ensuring data persisted correctly after deployment caused configuration issues (CORS and environment variable errors).

Solution:
Environment variables were securely stored in a .env file, and CORS middleware was enabled in the Express backend to allow requests from the frontend domain.

app.use(cors({ origin: "https://interactivequiz.netlify.app" }));

Deployment was then done on **Render** (backend) and **Netlify** (frontend), ensuring full integration.

8. Challenge: Maintaining State After Page Refresh

Problem:
If the user refreshed the page, authentication state and current quiz progress were lost.

Solution:
The JWT token and user data were stored securely in **localStorage**, allowing the app to reinitialize state after refresh.
This enhanced the user experience and prevented re-login interruptions.

9. Challenge: Testing and Debugging

Problem:
During testing, some API endpoints were not returning expected results, and UI bugs appeared during navigation.

Solution:
Postman was used to test each API route independently.
React Developer Tools and Console logs helped trace component-level bugs.
A systematic debugging approach improved code reliability before final deployment.

10. Challenge: Performance Optimization

Problem:
Initial loading time of the quiz questions was slow due to large data fetches from the backend.

Solution:
Pagination and selective data fetching were implemented to reduce payload size.
Lazy loading techniques and React memoization were used to optimize rendering performance.

Summary of Challenges and Solutions

Challenge	Solution Implemented
Routing and navigation issues	Used React Router and protected routes
Authentication security	Implemented JWT and bcrypt password hashing
Score calculation errors	Centralized score evaluation logic
API synchronization	Used async/await and error handling
Timer malfunction	Created reusable timer component
Responsive design	Applied Flexbox and media queries
Database and CORS errors	Configured .env and enabled CORS
Lost state after refresh	Used localStorage for persistence
API/UI bugs	Used Postman and React Dev Tools for debugging
Performance lag	Used lazy loading and memoization

Conclusion

Overcoming these challenges enhanced the robustness, security, and usability of the **Interactive Quiz App**. Each obstacle contributed to learning modern web development practices such as secure authentication, API management, and responsive UI design — resulting in a professional, fully functional final product.

5. GitHub README & Setup Guide

Project Title: Interactive Quiz App

An engaging web-based quiz platform that allows users to test their knowledge across various topics with instant scoring, authentication, and result tracking.

Project Description

The **Interactive Quiz App** is a full-stack web application designed to provide an enjoyable learning and testing experience.
Users can log in, choose quiz categories, answer multiple-choice questions, and get immediate feedback on their performance.
The app supports secure authentication, responsive design, and dynamic data handling through APIs.

Features

- ✅ User Registration & Login
- ✅ Category-based Quiz Selection
- ✅ Multiple Choice Questions with Timer
- ✅ Instant Score Calculation & Feedback
- ✅ View Previous Scores
- ✅ Admin Panel for Question Management (optional)
- ✅ Responsive UI for all devices
- ✅ Secure Authentication using JWT

Tech Stack

Component	Technology Used
Frontend	React.js / HTML / CSS / JavaScript
Backend	Node.js / Express.js
Database	MongoDB / Firebase
Tools	VS Code, GitHub, Postman
Hosting	Netlify (Frontend), Render or Vercel (Backend)

Folder Structure

Interactive-Quiz-App/

backend/

server.js

routes/

models/

controllers/

package.json

frontend/

src/

components/

pages/

App.js

index.js

public/

package.json

README.md

.gitignore

Installation and Setup Guide

Follow these steps to set up and run the project locally:

Step 1: Clone the Repository

git clone https://github.com/yourusername/interactive-quiz-app.git

cd interactive-quiz-app

Step 2: Set Up the Backend

1. Navigate to the backend folder:
2. cd backend
3. Install dependencies:
4. npm install
5. Create a .env file in the backend directory and add:
6. PORT=5000
7. MONGO_URI=your_mongodb_connection_string
8. JWT_SECRET=your_secret_key
9. Start the backend server:
10. npm start

Backend will run on: http://localhost:5000

Step 3: Set Up the Frontend

1. Navigate to the frontend folder:
2. cd ../frontend
3. Install frontend dependencies:

4. npm install
5. Create a .env file in the frontend directory (if needed) and define:
6. REACT_APP_API_URL=http://localhost:5000
7. Start the React development server:
8. npm start

Frontend will run on: http://localhost:3000

Step 4: Connect Frontend & Backend

- Ensure both frontend and backend servers are running.
- The frontend will make API requests to the backend for authentication, questions, and score data.
- Check the browser console or terminal for successful API responses.

Step 5: Test the Application

- Open the app in a browser at **http://localhost:3000**
- Register or log in as a user.
- Select a quiz category and begin answering questions.
- Submit your quiz to view your score instantly.

API Endpoints Overview

Endpoint	Method	Description
/api/register	POST	Register a new user
/api/login	POST	User login and token generation
/api/categories	GET	Fetch quiz categories
/api/questions/:category	GET	Fetch questions by category
/api/submit	POST	Submit answers and calculate score
/api/scores/:userId	GET	Retrieve user's score history
/api/admin/addQuestion	POST	(Admin) Add new question

Deployment

Frontend: Deployed on Netlify
Backend: Deployed on Render or Vercel
Database: Hosted on MongoDB Atlas

Example URLs:

- **Frontend:** https://interactivequiz.netlify.app
- **Backend:** https://quizapi-backend.onrender.com

Git Commands Used

git init

git add .

git commit -m "Initial commit - Interactive Quiz App"

git branch -M main

git remote add origin https://github.com/yourusername/interactive-quiz-app.git

git push -u origin main

Usage Instructions

1. Open the deployed URL or local environment.
2. Register or log in to your account.
3. Choose a quiz category and answer all questions.
4. Submit to view your score and feedback.
5. Review past performance in the **Score History** section.

Environment Variables

Backend (.env):

PORT=5000

MONGO_URI=your_mongodb_connection_string

JWT_SECRET=your_secret_key

Frontend (.env):

REACT_APP_API_URL=http://localhost:5000

Testing

- All REST APIs tested using **Postman**.
- Frontend tested on Chrome, Edge, and Firefox browsers.
- Responsive testing on mobile and tablet screen sizes.

Dependencies Used

Frontend:

- React
- React Router DOM
- Axios

- Bootstrap / Tailwind CSS

Backend:

- Express.js
- Mongoose
- Bcrypt.js
- JSON Web Token (JWT)
- Cors
- Dotenv

Screenshots

Include these in your GitHub repo for better documentation:

- /assets/home-page.png
- /assets/login-page.png
- /assets/dashboard.png
- /assets/quiz-page.png
- /assets/result-page.png

Contributing

Contributions, suggestions, and feedback are welcome!
To contribute:

1. Fork the repository
2. Create a new branch (feature-branch)
3. Commit changes
4. Push and create a pull request

License

This project is licensed under the **MIT License** – feel free to use and modify with proper credit.

Author

Developed by: *[Your Name]*
GitHub: https://github.com/yourusername
Email: yourname@example.com

6. Final Submission (Repository + Deployed Link)

GitHub Repository

The complete source code for both frontend and backend of the project is available on GitHub.
It includes all components, routes, and setup instructions.

- **GitHub Repository Link:**
  - https://github.com/yourusername/interactive-quiz-app

Repository Contents:

- /frontend → React application source files
- /backend → Node.js & Express server files
- /assets → Screenshots and static images
- /README.md → Full setup and usage guide

Branch Used: main
Commit Message Example: Final Phase 5 Submission - Interactive Quiz App

Live Deployed Application

The project is deployed online for live testing and demonstration.

- **Frontend (Deployed on Netlify):**
  - https://interactivequiz.netlify.app
- **Backend (Deployed on Render/Vercel):**
  - https://quizapi-backend.onrender.com

Database:

- Hosted on **MongoDB Atlas**, ensuring secure cloud-based storage for user data, quiz questions, and scores.

Access Instructions

1. **Open the Frontend Deployed Link:**
   - Visit https://interactivequiz.netlify.app in your browser.
2. **Login or Register:**
   - Create a new account or log in with an existing one.
3. **Start a Quiz:**
   - Select any category and begin answering questions.
4. **Submit and View Results:**
   - After completing the quiz, the app instantly displays your score and feedback.

5. **Check Score History:**
    - View your previous attempts to track performance.

Deployment Summary

Component	Platform	Status	Link
Frontend	Netlify	✅ Live	View Frontend
Backend	Render / Vercel	✅ Live	View API
Database	MongoDB Atlas	✅ Connected	–
GitHub Repo	GitHub	✅ Public	View Repository

Final Submission Details

Submission Component	Status	Description
Project Report	✅ Completed	Detailed documentation of the project design and functionality
Screenshots/API Docs	✅ Included	Showcases UI and REST API structure
Demo Walkthrough	✅ Added	Step-by-step explanation of application workflow
Challenges & Solutions	✅ Explained	Discusses key problems faced and how they were solved
GitHub README & Setup	✅ Prepared	Complete installation and usage guide
Final Submission (Repo + Links)	✅ Ready	Includes hosted app and source repository

Developer Contact

Developed by: *[Your Name]*
Email: yourname@example.com
GitHub: https://github.com/yourusername
LinkedIn: https://linkedin.com/in/yourprofile

Conclusion

The **Interactive Quiz App** successfully fulfills all requirements of the **Phase 5 Project**, demonstrating complete lifecycle development — from design, implementation, and API integration to deployment and documentation.
Both the GitHub repository and live deployment links ensure easy access for evaluation and testing.

GitHub:

https://github.com/Selvarinisha777/Interactive-Quiz-app.git

Netify:

https://app.netlify.com/teams/selvarinisha777/projects