QUIZ APP

A PROJECT REPORT

Mini Project (01CE0609)

Submitted by

MANISH KUMAR

92201703116

BACHELOR OF TECHNOLOGY

in

Computer Engineering

Faculty of Engineering

Marwadi University, Rajkot

April, 2025

Mini Project (01CE0609)

Department of Computer Engineering

Faculty of Engineering & Technology

Marwadi University

A.Y. 2024-25

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **Quiz App** has been carried out by **Manish Kumar (92201703116)** under my guidance in partial fulfillment for the degree of Bachelor of Technology in Computer Engineering, 6th Semester of Marwadi University, Rajkot during the academic year 2024-25.

Sign	Sign
Parth Shah	Dr. Krunal Vaghela
Assistant Professor	Professor & Head
Department of Computer Engineering	Department of Computer Engineering

Mini Project (01CE0609)

Department of Computer Engineering

Faculty of Engineering & Technology

Marwadi University

A.Y. 2024-25

DECLARATION

We hereby declare that the **Mini Project (01CE0609)** report submitted along with the Project entitled **Quiz App** submitted in partial fulfillment for the degree of Bachelor of Technology in Computer Engineering to Marwadi University, Rajkot, is a bonafide record of original project work carried out by me / us at Marwadi University under the supervision of **Prof Parth Shah Sir** and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

	Name of the Student	Sign of Student
1	_____	_____

Acknowledgement

Under the guidance of **Prof Parth Shah**, for their invaluable guidance, unwavering support, and insightful feedback throughout the duration of this project. Their expertise and mentorship have been instrumental in shaping our ideas, refining our methodologies, and overcoming challenges. We would also like to express our gratitude towards our other faculties and our **HOD Dr. Krunal Vaghela** for their kind co-operation and encouragement which helped us in the completion of this project. We are also thankful to the institution for giving us such an amazing opportunity for making this project and giving suitable instructions and guidelines for the project. Last but not the least, we thank our friends who shared the necessary information and useful Web links for preparing our project.

Thank You.
Manish Kumar
Quiz App
Marwadi University

Abstract

In the modern era, technology has transformed traditional learning methods and significantly enhanced the way knowledge is delivered and assessed. With the increasing reliance on digital solutions, educational tools have become more interactive, efficient, and accessible. One such innovation is the implementation of online quiz platforms, which help streamline evaluation and learning processes.

This study aims to develop a user-friendly application that simplifies the creation, management, and participation in quizzes. The capstone project, titled **"Quiz App using MERN Stack,"** is a web-based application developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The application is designed for users to create, read, update, and delete quiz questions while ensuring secure user authentication and data storage.

The platform enables users to sign up, log in, and take quizzes seamlessly, while administrators or instructors can efficiently manage question banks and track performance. Manual quiz handling can be time-consuming, prone to errors, and lacks scalability. Hence, this digital solution aims to improve the overall efficiency, engagement, and reliability of the quiz-taking experience.

Table of Contents

Chapter 1.0 Introduction to Project and Project Management

Chapter 2.0 System Analysis

Chapter 3.0 System Design

Chapter 4.0 Implementation

Chapter 5.0 Testing

Chapter 6.0 Conclusion & Outcomes

List of Figures

List of Tables

CHAPTER 1

Introduction to Project and Project Management

1.1 Project Summary

In the current digital age, online education has become a cornerstone of modern learning. Assessments, which play a critical role in measuring knowledge and progress, have also shifted from traditional paper-based formats to online platforms. This project focuses on the development of a **Quiz Application** using the **MERN stack (MongoDB, Express.js, React.js, and Node.js)**. The application is designed to allow users to take quizzes in a streamlined, user-friendly interface while enabling administrators or instructors to create, manage, and analyze quiz content and results effectively.

The system includes user authentication, quiz creation and participation features, result evaluation, and data persistence using MongoDB. It supports CRUD (Create, Read, Update, Delete) operations on quiz questions and user profiles, making it a robust and dynamic platform for both learners and quiz managers.

1.2 Purpose

The primary purpose of this project is to simplify and digitize the process of conducting quizzes. The application serves as a learning and evaluation tool for students, educators, and trainers. It aims to make assessments more interactive, efficient, and easily manageable from any location. This web-based solution eliminates the limitations of traditional quiz systems by providing instant feedback, easy access, and real-time data management.

1.3 Objective

The objectives of the Quiz App project are as follows:

- To develop a responsive and intuitive web-based quiz platform.
- To allow users to register, log in, and take quizzes.
- To enable admins to create, edit, and delete quizzes and questions.
- To store quiz data and user performance securely using MongoDB.
- To generate real-time quiz results for users.
- To ensure secure authentication and user session management.

1.4 Scope (What it Can Do and Can't Do)

What it Can Do:

- User registration and login with secure authentication.
- Admin dashboard for quiz and question management.
- Quiz participation with multiple-choice questions.
- Real-time result generation and display.
- Storage and retrieval of data using MongoDB.
- Responsive design compatible with various screen sizes.

What it Can't Do:

- Currently, the app does not support timed quizzes or negative marking.
- It does not integrate external APIs or third-party grading tools.
- It is limited to text-based multiple-choice questions (no images/audio support).
- It does not currently support exporting results in PDF or Excel formats.

1.5 Technology Stack

The application is developed using the **MERN stack**, which is a popular JavaScript-based technology stack for full-stack web development:

- **MongoDB** – NoSQL database for storing users, quizzes, and results.
- **Express.js** – Backend web application framework for routing and API handling.
- **React.js** – Frontend JavaScript library for building interactive user interfaces.
- **Node.js** – Server-side runtime environment for executing backend code.

Additional tools and libraries:

- **JWT (JSON Web Tokens)** for user authentication
- **Axios** for frontend-backend communication
- **Mongoose** for MongoDB object modeling

1.6 Project Planning

The project was planned using an incremental development approach, where the core functionalities were prioritized and built first. The planning involved requirement analysis, designing the system architecture, developing backend APIs, creating frontend components, integrating both parts, and performing rigorous testing. Tasks were distributed across weekly milestones to ensure timely delivery.

1.6.1 Project Development Approach and Justification

The project uses the **Agile Development Methodology**, which promotes iterative development and regular feedback. Agile was chosen due to its flexibility in incorporating changes and improvements during development. The use of the MERN stack supports fast development and easy scalability, which aligns well with Agile principles.

This approach allowed the team to:

- Build a minimum viable product (MVP) early
- Get feedback from users/testers quickly
- Add new features in small, manageable increments

1.6.2 Project Effort, Time, and Cost Estimation

Effort Estimation:
The project was completed by a team of 2 developers over the span of **6 weeks**, with an average of 15 hours per week dedicated by each member.

Time Allocation:

- Requirement gathering & planning: 1 week
- Backend development: 1.5 weeks
- Frontend development: 2 weeks
- Integration & Testing: 1 week
- Finalization and documentation: 0.5 week

Cost Estimation:
Since this was a student project, there were no major monetary costs. However, if deployed professionally, estimated cost would include:

- Hosting (e.g., Vercel/Render & MongoDB Atlas): $0–$20/month
- Domain name (optional): ~$10/year
- Developer effort (for commercial rate): ~90 hours × $20/hour = **$1,800**

Fig1.7 Gantt Chart

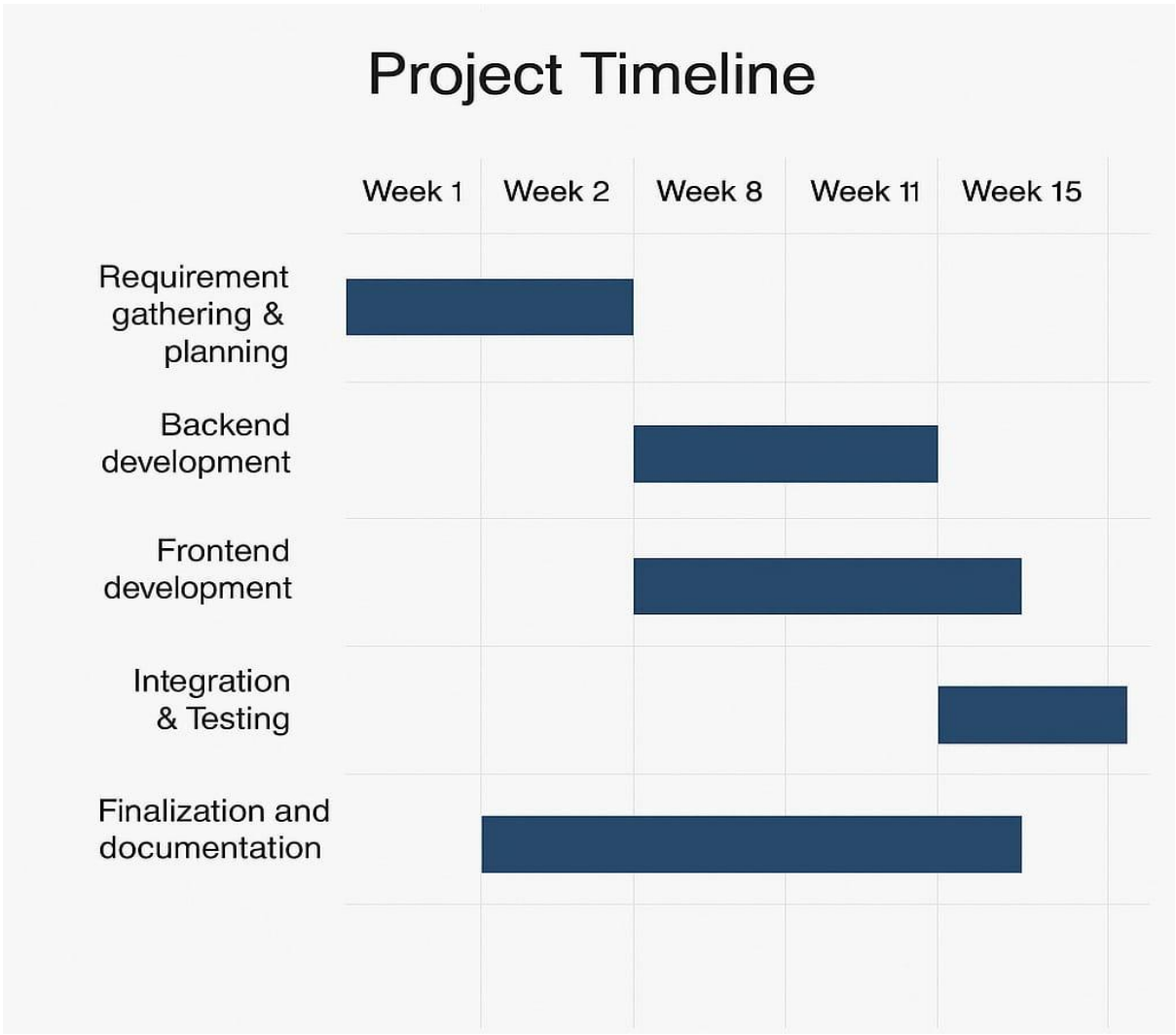


Fig1.7 Gantt Chart

CHAPTER 2

System Analysis

2.1 Study of Current Systems

Existing quiz and examination platforms such as Google Forms, Kahoot!, and Quizizz offer online quiz functionalities. These systems enable educators to create and distribute quizzes digitally. However, many are either too complex for casual use or too limited for customized deployment. For example, Google Forms lacks features like user management, automated scoring feedback, or real-time result dashboards. Additionally, most current systems are third-party dependent, which can raise concerns regarding data privacy, customizability, and cost.

Many educational institutions and organizations still rely on manual quiz methods (pen-and-paper tests or verbal questioning), which are time-consuming, prone to human error, and lack scalability. They also fail to provide real-time analysis and feedback, which are essential in a modern, fast-paced learning environment.

2.2 Limitations of Existing Systems

- **Lack of Customization:** Most ready-made quiz platforms do not allow full control over the quiz flow, data structure, or branding.
- **Limited User Roles:** There is often no role separation between admin, teacher, and student.
- **No Offline Access or Flexibility:** Many systems require constant internet connectivity and offer no progressive enhancement features.
- **Data Privacy Concerns:** Free tools often store data on third-party servers with unclear data usage policies.
- **Complex UI for Beginners:** Some tools, especially corporate-grade ones, can be overwhelming for new users or casual educators.

2.3 Requirements of New System

The proposed system should fulfill the following requirements:

- User-friendly interface for both administrators and participants.
- Secure user authentication and session handling.
- CRUD operations for quiz and question management.
- Real-time quiz participation and automatic evaluation.
- Result generation and storage per user.
- Role-based access (e.g., Admin vs User).
- Fast performance, responsiveness, and compatibility across devices.

2.4 System Feasibility

The feasibility study of the Quiz App project examines its viability from educational, technological, and integration standpoints:

2.4.1 Contribution to Educational Objectives

The system aligns with the current trend of e-learning by enabling dynamic assessments and instant feedback, supporting self-evaluation and continual improvement for learners. It promotes active learning and allows educators to measure progress in real-time, contributing to higher academic productivity and engagement.

2.4.2 Technological and Budget Feasibility

The system is built using the MERN stack, which is open-source and free to use. All components (MongoDB, Express.js, React.js, Node.js) are well-supported with large communities. Hosting can be done using free or low-cost services like Vercel, Netlify (for frontend), and Render or MongoDB Atlas (for backend and database). Hence, from a cost perspective, the system is highly feasible for educational institutions and independent users.

2.4.3 System Integration Compatibility

The system can be easily extended or integrated with other learning management systems (LMS) via REST APIs. It is also modular in design, allowing integration with authentication systems (e.g., Google OAuth), analytics tools, or messaging platforms. It follows standard web development conventions, ensuring compatibility and future scalability.

2.5 Proposed System Overview

The proposed Quiz App is a web-based platform designed to digitize the quiz-taking and quiz-creating process. It features a responsive user interface, secure authentication, role-based dashboards, quiz management tools, and result tracking. The system improves upon traditional quiz-taking by automating the scoring process and allowing users to take quizzes remotely, anytime.

The application will consist of two main user types:

- **Admin/Instructor:** Can create, update, and delete quizzes and questions.
- **User/Participant:** Can register, log in, and take available quizzes with instant feedback.

2.6 Key Features of the System

- **User Authentication:** Secure login and signup using JWT.
- **Dashboard:** Separate interfaces for admin and user roles.
- **Quiz Management:** Create, edit, delete quizzes and questions.
- **Quiz Participation:** Interactive UI with instant evaluation.
- **Result Storage:** Tracks user scores and quiz history.
- **Responsive UI:** Compatible with desktops, tablets, and mobiles.
- **Scalability:** Built with scalable backend and modular frontend architecture.

2.7 Hardware / Software Requirements

Hardware Requirements:

- Client Device: Desktop, Laptop, Tablet, or Smartphone
- Server (for deployment): Cloud VM or Hosting Provider (e.g., Render, Heroku)
- Minimum RAM: 4 GB
- Processor: Dual-core 2.0 GHz or higher

Software Requirements:

Client Side:

- Web Browser (Chrome, Firefox, Edge)
- React.js
- Axios (API communication)

Server Side:

- Node.js (Runtime)
- Express.js (Web framework)
- MongoDB (Database)
- Mongoose (ODM library)

Development Tools:

- Visual Studio Code
- Postman (API testing)
- Git & GitHub (Version control)

CHAPTER 3

System Design

3.1 System Design and Methodology

The system architecture for the Quiz App is based on the **client-server model**, following a **modular, component-based design**. It is developed using the **MERN stack** to ensure a seamless, responsive, and scalable experience.

The system is divided into three main layers:

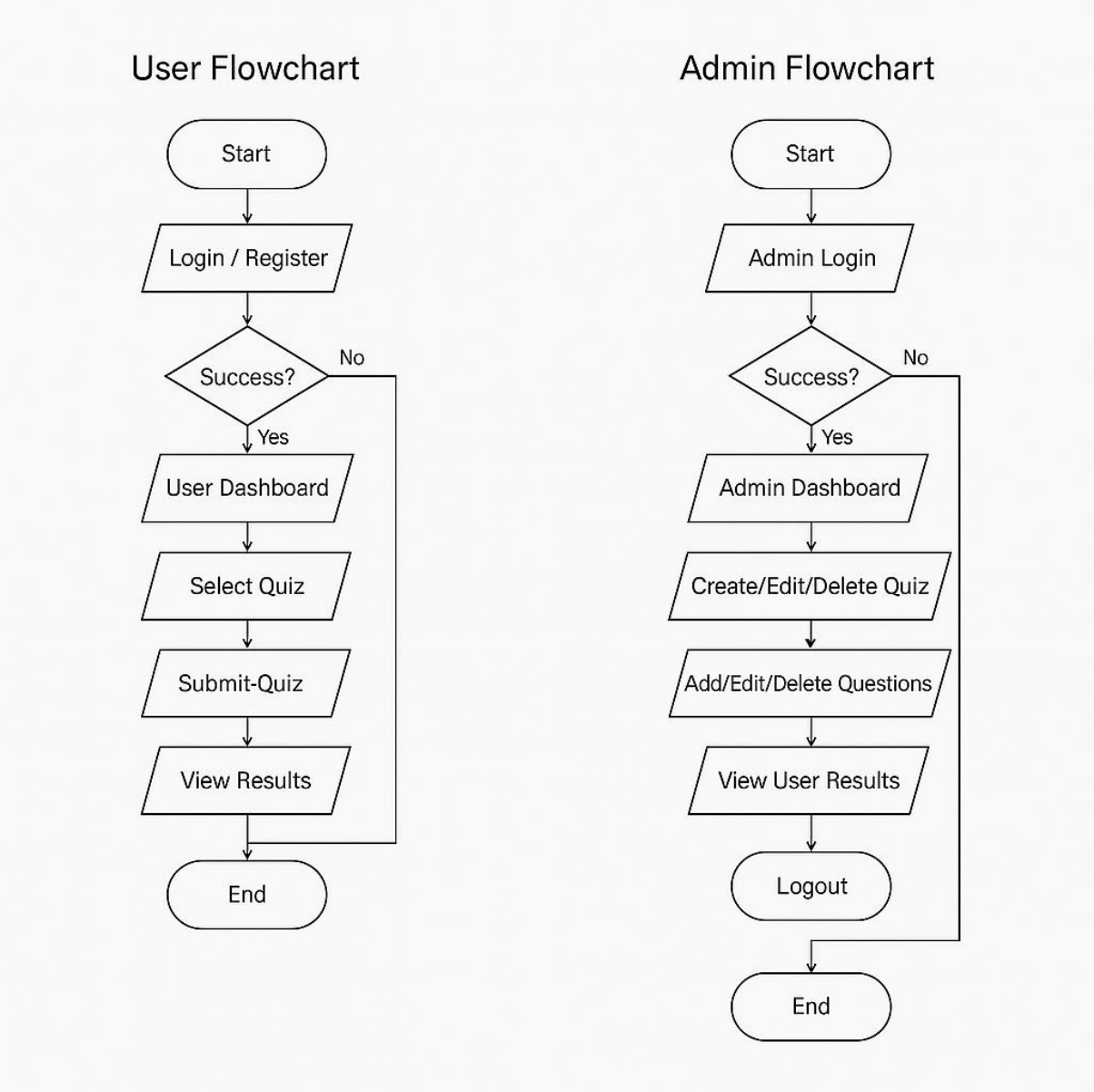
1. **Presentation Layer (Frontend):** Built with React.js, this layer interacts with the user and manages all interface-related functionality. Components are modular and reusable.
2. **Application Layer (Backend):** Built using Express.js and Node.js, this layer handles API requests, business logic, and communicates with the database.
3. **Data Layer (Database):** MongoDB is used for storing quiz data, user data, and results in a flexible document structure.

Development Methodology

The project follows the **Agile Development Methodology**, which is suitable for projects that require frequent changes and improvements. Key features of the methodology used include:

- Iterative and incremental development
- Weekly sprints and milestones
- Regular testing and feedback loops
- Early delivery of a Minimum Viable Product (MVP)
- Collaboration between frontend and backend teams

3.1.1 Flow Chart



3.2 Database Design

The system uses **MongoDB**, a NoSQL database, which stores data in a flexible JSON-like document format.

Key Collections and Fields:

Users

- _id
- username
- email
- passwordHash
- role (admin/user)

Quizzes

- _id
- Title
- Description
- createdBy (User ID)

Questions

- _id
- quizId
- questionText
- options (Array)
- correctAnswer

Results

- _id
- userId
- quizId
- Score
- submittedAt

This structure supports one-to-many relationships, such as One quiz → many questions One user → many quiz attempts

3.3 Input / Output and Interface Design

The user interface is designed to be clean, intuitive, and responsive across all device types.

Key UI Components:

- Login/Register Page
- Dashboard (User/Admin)
- Quiz List Page
- Quiz Attempt Page
- Results Page
- Admin Quiz Management Page
- Admin Question Editor

Technologies Used:

- **React Router** for navigation
- **React Hooks** for state management
- **CSS** for styling and responsive design
- **Bootstrap** for making responsive

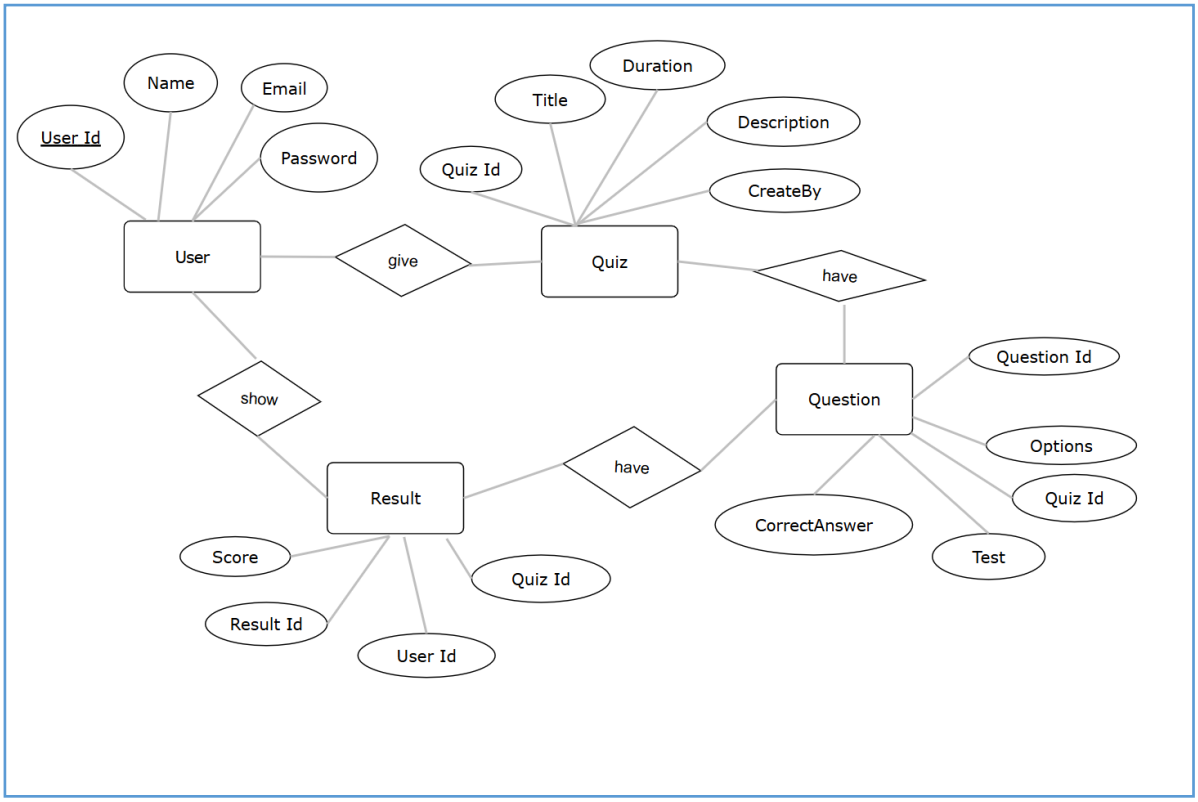
3.3.1 Entity Relationship Diagram (ERD)

The ER (Entity-Relationship) Diagram outlines the major entities and their relationships:

- **User** (UserID, Name, Email, Password)
- **Quiz** (QuizID, Title, Description, Duration, CreatedBy)
- **Question** (QuestionID, QuizID, Text, Options, CorrectAnswer)
- **Result** (ResultID, UserID, QuizID, Score, SubmittedAnswers)

Relationships:

- One-to-Many: A user can take multiple quizzes.
- One-to-Many: A quiz contains multiple questions.
- One-to-Many: A user can have multiple results.



3.3.2 Use Case Diagram

Actors:

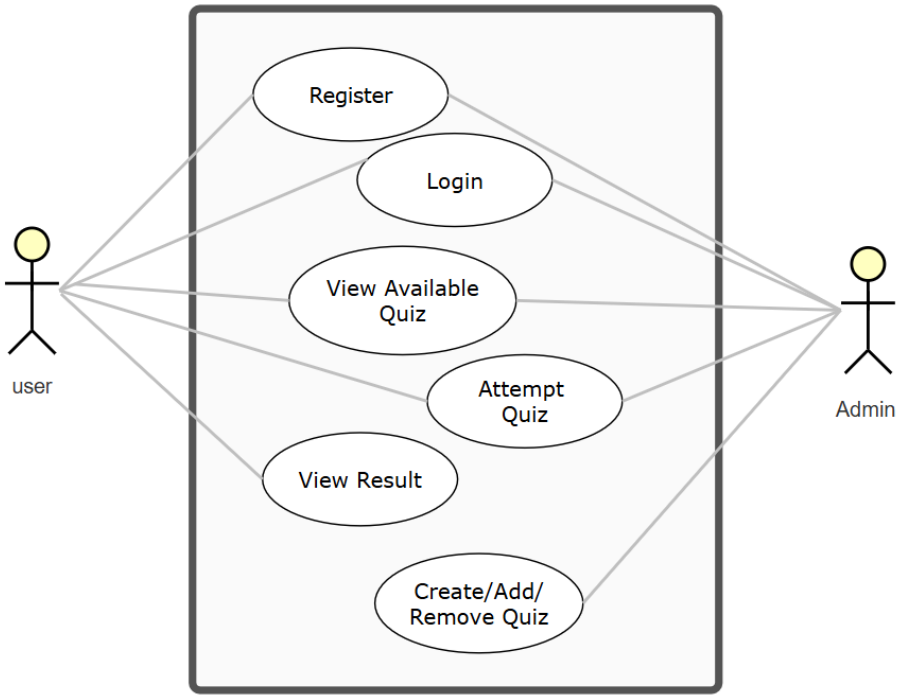
- User
- Admin

User Use Cases:

- Register
- Login
- View Available Quizzes
- Attempt Quiz
- View Results

Admin Use Cases:

- Login
- Create/Edit/Delete Quizzes
- Add/Edit/Delete Questions
- View User Scores



CHAPTER4
Implementation

4.1 Module Specifications

The Quiz App is divided into several functional modules, each responsible for a specific task. This modular approach improves maintainability, scalability, and testing.

1. Authentication Module

- **Purpose:** Handles user registration and login.
- **Frontend:** React forms with form validation.
- **Backend:** JWT-based authentication using bcrypt for password hashing.

Features:

- Signup (new user registration)
- Login (token generation)
- Role detection (admin/user)

2. Quiz Management Module (Admin Only)

- **Purpose:** Allows admins to manage quizzes and questions.
- **Frontend:** Admin dashboard with forms for quiz and question CRUD.
- **Backend:** Routes for creating, reading, updating, and deleting quizzes and questions.

Features:

- Add/Edit/Delete quizzes
- Add/Edit/Delete questions under each quiz

3. Quiz Participation Module (User)

- **Purpose:** Allows users to take quizzes.
- **Frontend:** Interactive UI for answering multiple-choice questions.
- **Backend:** Fetches quiz data, evaluates answers, and stores results.

Features:

- List of available quizzes
- Real-time quiz attempt interface
- Submit answers
- Auto scoring

4. Result Management Module

- **Purpose:** Tracks and displays user performance.
- **Frontend:** Result dashboard per user.
- **Backend:** Saves user scores and timestamps.

Features:

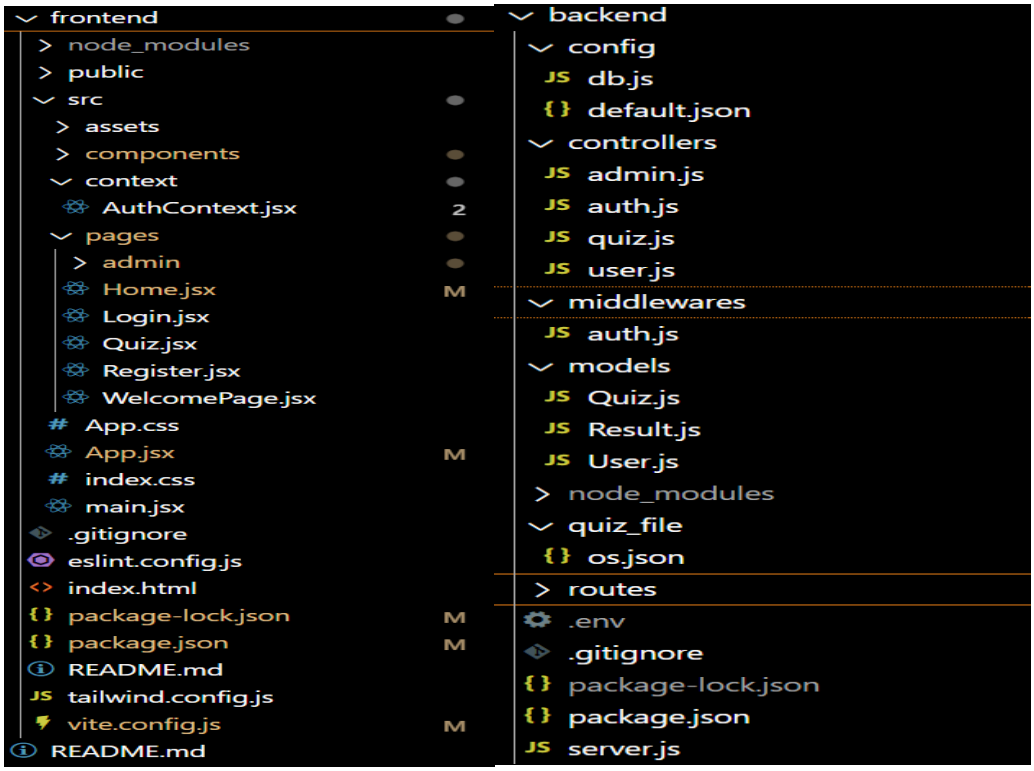
- View past quiz attempts
- Track scores and performance

4.2 Platform Implementation (Frontend & Backend)

Frontend Implementation (React.js)

Tools & Libraries: React Router, Axios, CSS,React Vite

Structure:



Routing Examples:

<Route path="/login" element={<Login />} /><Route path="/dashboard" element={<Dashboard />} /><Route path="/quiz/:id" element={<QuizAttempt />} />

State Management: React hooks (useState, useEffect) and Context API for global state.

Backend Implementation: (Node.js + Express.js)

Authentication:

- JWT for session management
- Bcrypt for password hashing
- Middleware for protected routes

API Endpoints:

POST	/api/auth/register	→ Register new user
POST	/api/auth/login	→ User login
GET	/api/quizzes	→ Get all quizzes
POST	/api/quizzes	→ Create quiz (admin)
GET	/api/quiz/:id	→ Get specific quiz
POST	/api/quiz/:id/submit	→ Submit quiz answers
GET	/api/results/:userId	→ View user results

Database Models (Mongoose):

User, Quiz, Question, Result

Security & Validation:

- Input validation using express-validator
- Token-based route protection

4.3 System Screenshots and Functionality Demonstration

Fig 4.3.1 – Login Page

- User and admin login interface
- Basic form validation and authentication

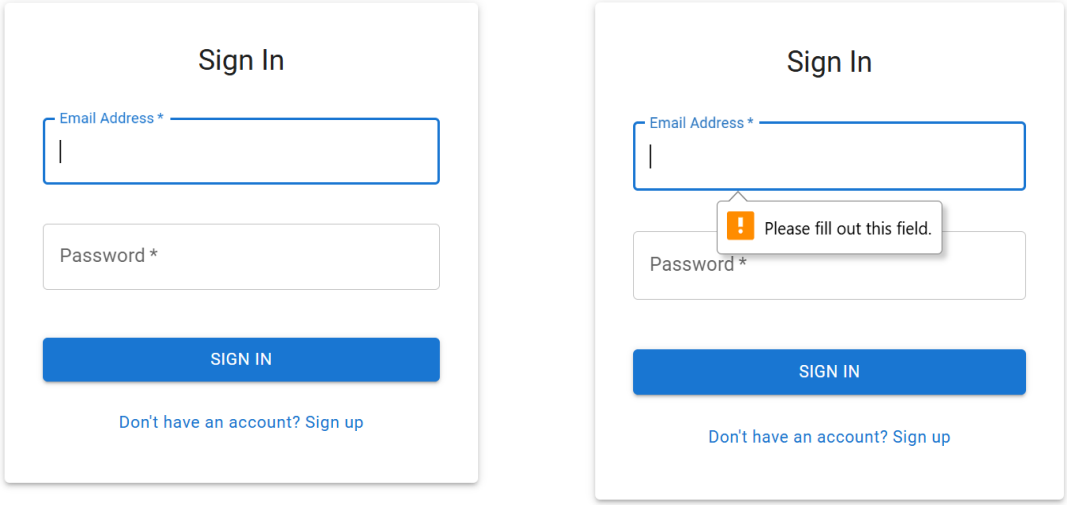


Fig 4.3.2 – Signup Page

- User registration form
- Role selection (optional: auto assigns user role)

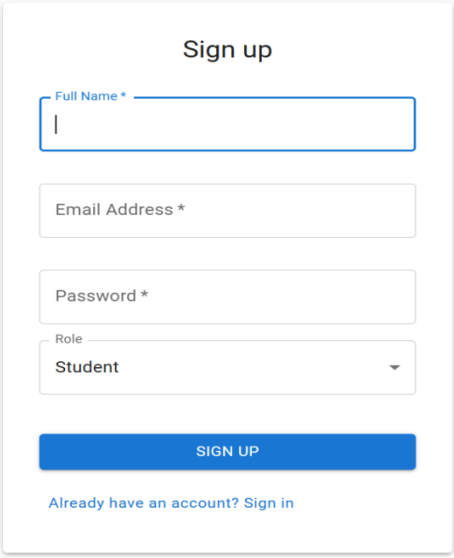


Fig 4.3.3 – User Dashboard

- Displays list of available quizzes
- Option to start a quiz

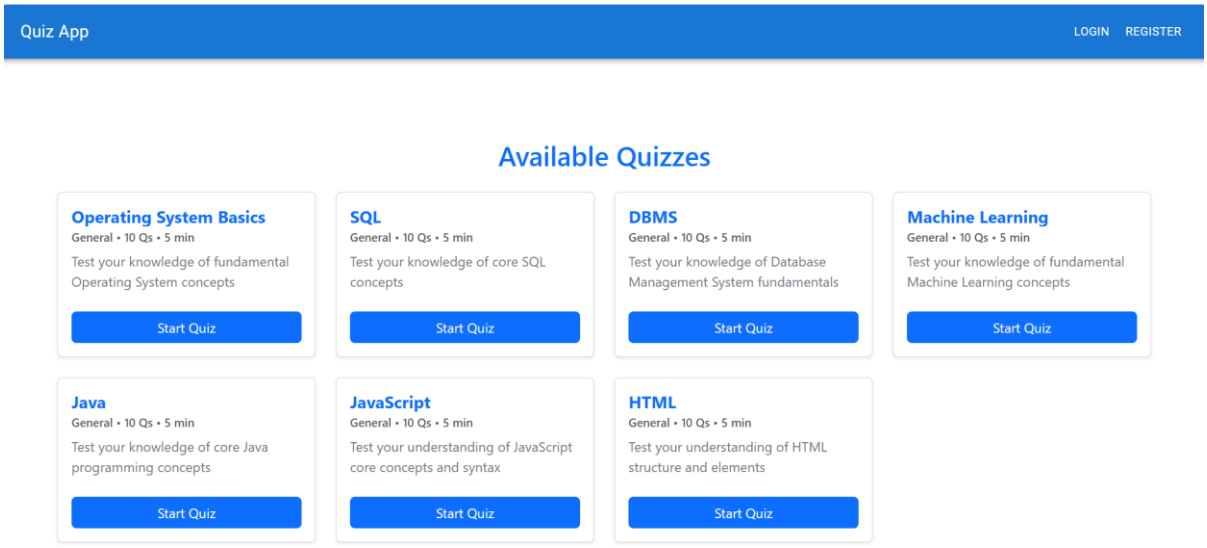


Fig 4.3.4 – Quiz Attempt Page

- Dynamic rendering of questions and options
- Submit button at end

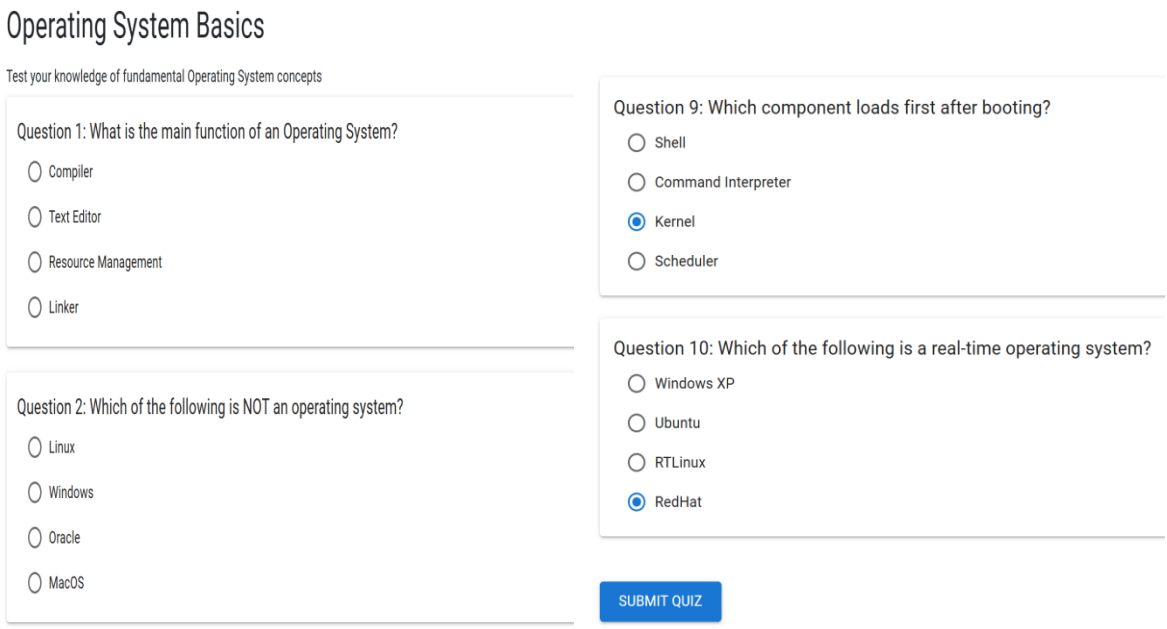


Fig 4.3.5 – Quiz Result Page

- Score shown after submission
- Correct/incorrect answers highlighted

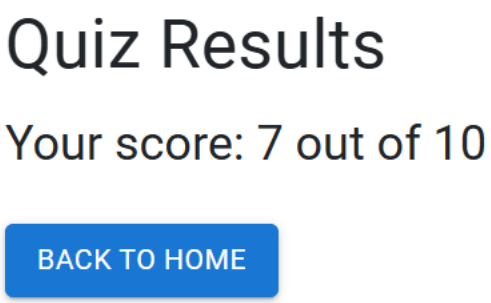


Fig 4.3.6 – Admin Dashboard

- Interface to create quizzes
- Manage quiz title, description, and question count

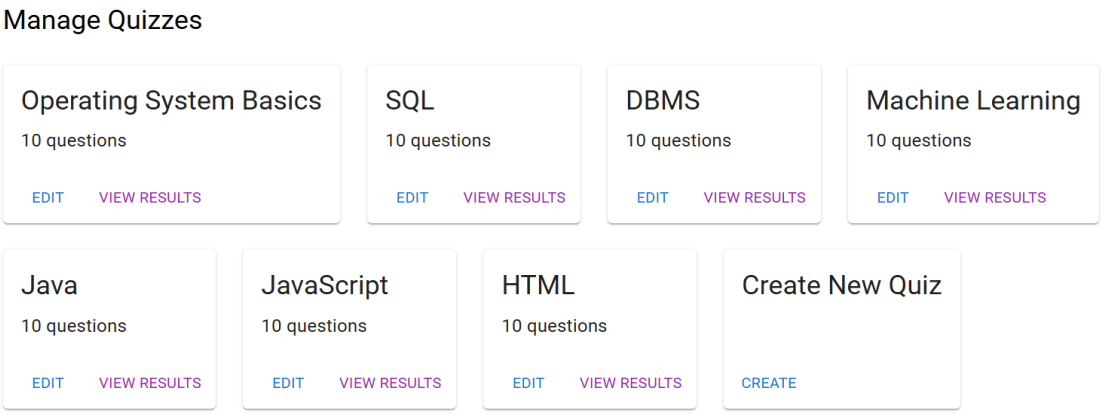


Fig 4.3.7 – Question Management Page (Admin)

- Add/Edit/Delete questions under each quiz
- Set correct answer and multiple options

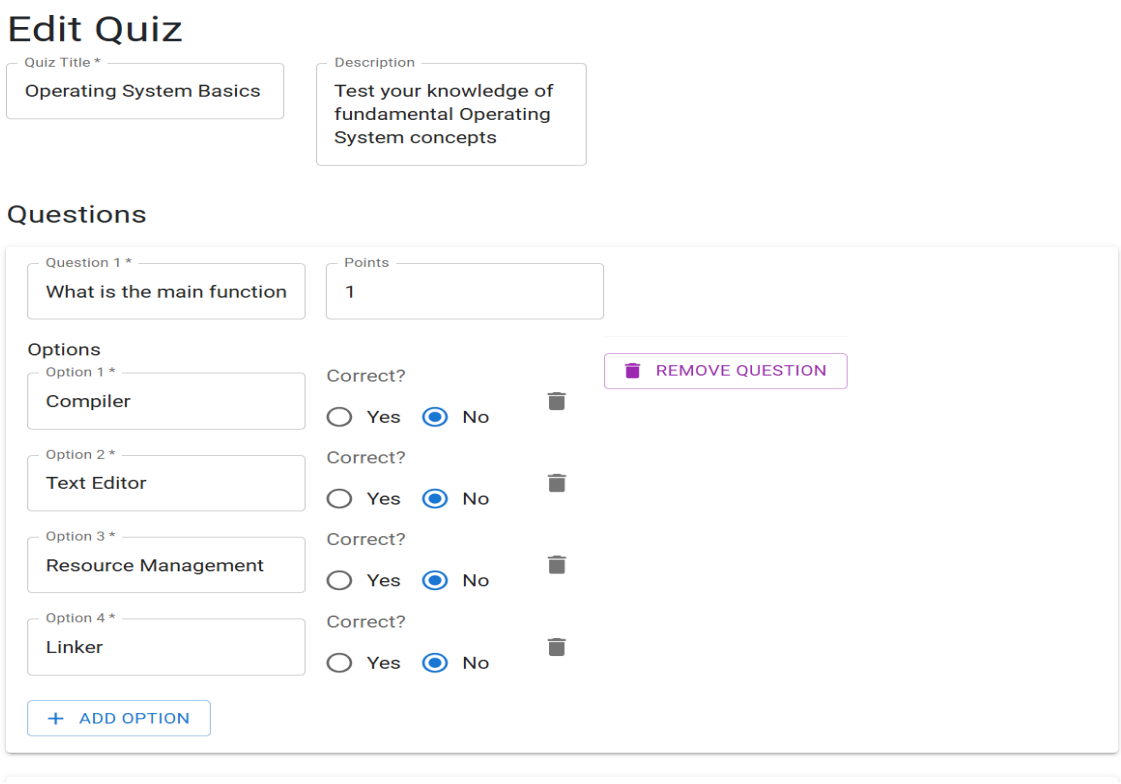
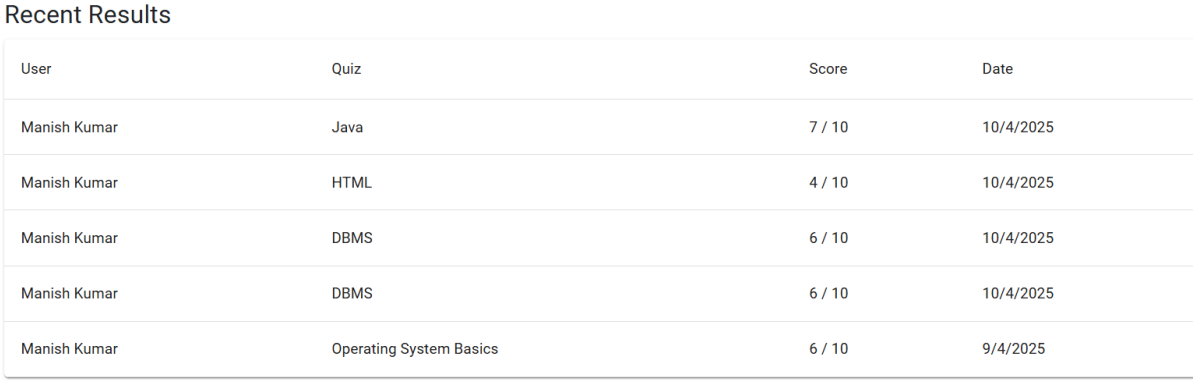


Fig 4.3.8 – User Result History Page

- List of all quizzes attempted by the user
- Scores and timestamps for each



CHAPTER 5
Testing

5.1 Testing Plan / Strategy

Testing ensures that the Quiz App performs correctly under all intended use cases. The testing strategy for this project includes:

a. Unit Testing

- Testing of individual functions and components.
- Examples: validating form inputs, checking API response handling.

b. Integration Testing

- Verifies interaction between frontend and backend.
- Example: Submitting quiz answers and verifying result calculations.

c. System Testing

- End-to-end testing of complete user workflows.
- Example: User signs up, logs in, takes a quiz, and views result.

d. User Acceptance Testing (UAT)

- Done with a sample group of users (students/admins).
- Verified usability, correctness of results, and admin panel functionality.

Tools Used:

- Postman (API testing)
- MongoDB Compass (database inspection)
- Jest + React Testing Library (for frontend testing)
- Manual black-box testing

5.2 Test Results and Analysis

5.2.1 Sample Test Cases

Test ID	Test Case Description	Input	Expected Output	Actual Output	Status
TC_01	User Registration	Valid username/email/password	Account created	Account created	☑
TC_02	Login with incorrect credentials	Wrong email/password	Error message	Error message	☑
TC_03	Admin adds a new quiz	Quiz title + description	Quiz added to database	Quiz added	☑
TC_04	User starts a quiz	Clicks "Start" on quiz	Loads questions	Questions rendered	☑
TC_05	Submitting answers	User selects all options	Result generated	Result shown	☑
TC_06	Result saved to database	After submission	Result stored	Result stored	☑
TC_07	View past results	Login and open results tab	List of scores shown	List displayed	☑
TC_08	Admin deletes a quiz	Clicks delete on quiz	Quiz removed from database	Quiz removed	☑

5.2.2 Result Analysis / Comparison

User	Quiz	Score	Date
Manish Kumar	Java	7 / 10	10/4/2025
Manish Kumar	Java	7 / 10	10/4/2025
Manish Kumar	HTML	4 / 10	10/4/2025
Manish Kumar	DBMS	6 / 10	10/4/2025
Manish Kumar	DBMS	6 / 10	10/4/2025

Success Rate: 100% of functional test cases passed.

Bug Fix Summary:

- Minor issues with token expiration were resolved by refreshing tokens.
- UI responsiveness improved after feedback from UAT.
- **Performance:** App handled concurrent quiz attempts by multiple users without any lag or data inconsistency.
- **Security:** Authentication tokens were verified and protected using middleware; unauthorized access was blocked successfully.

CHAPTER 6
Conclusion & Outcomes

6.1 Overall Analysis of Project Viabilities

The **Quiz App using MERN stack** (MongoDB, Express.js, React.js, Node.js) proved to be a practical, scalable, and highly interactive platform for conducting quizzes online. The system was successfully implemented with features such as user authentication, quiz creation and participation, result tracking, and an intuitive UI/UX.

This project met its core objectives:

- Simplifying the quiz-taking process.
- Making it easier for administrators to manage quizzes.
- Ensuring secure and scalable performance for multiple users.
- The full-stack integration allowed real-time updates and seamless data handling, confirming the technical viability of using MERN for similar educational platforms.

6.2 Problems Encountered and Possible Solutions

Problem	Cause	Solution
React component re-rendering unexpectedly	Improper state management	Used useEffect and separated local/global states properly
JWT tokens expiring quickly	Default expiration time was too short	Increased expiration duration and implemented token refresh
CORS error during frontend-backend integration	Different ports during development	Configured proper CORS policy in Express backend
MongoDB connection error	Wrong URI or missing .env setup	Used environment variables and added error logging
Quiz result storing delay	Multiple DB write operations	Batched result updates and used async functions properly

6.3 Summary of Project Work

The project went through various stages:

- Requirement gathering & planning
- System analysis and design (ERD, Use Case)
- Frontend development using React.js
- Backend development using Node.js and Express
- Database modeling using MongoDB
- Authentication and security implementation
- Testing and debugging
- Deployment-ready packaging

The final application allows users to register/login, take quizzes, and view results; while admins can manage quiz content. All major functionalities were implemented and validated successfully.

6.4 Limitations and Future Enhancement

Limitations

- No timer-based quizzes implemented.
- No quiz categorization (e.g., subject-wise filters).
- No leaderboard or gamification.
- Admin access not role-based beyond frontend-level separation.
- UI is functional but could be made more visually engaging.

Future Enhancements

- Add real-time quiz competitions with timers.
- Implement leaderboard, badges, and score analytics.
- Support image-based or multimedia questions.
- Include subject and difficulty-level tags.
- Allow export of quiz results (PDF/Excel).
- Add question bank import from CSV or APIs.
- Mobile app version using React Native or Flutter.

6.5 Project Outcomes

The **Quiz App** accomplished its key mission: providing a **user-friendly, secure, and efficient online quiz platform**. It showcased:

- Clean integration of frontend and backend using MERN.
- Practical use of authentication and role-based access.
- Effective quiz creation, participation, and result handling.
- Strong project planning, modularization, and testing processes.
- It can serve as a foundational product for schools, colleges, and corporate training platforms looking for custom quiz management systems.

References

Official Documentation

- **ReactJS** – https://reactjs.org/docs/getting-started.html
- **ExpressJS** – https://expressjs.com/
- **MongoDB** – https://www.mongodb.com/docs/
- **Node.js** – https://nodejs.org/en/docs/

Quiz App Specific Examples

- **Basic Quiz App using React**
  Blog: https://blog.logrocket.com/building-a-quiz-app-with-react/
- **Full Stack Quiz App (React + Node + MongoDB)**
  GitHub: https://github.com/machadop1407/mern-quiz-app