AI QUIZIFY

A Minor Project Report Submitted
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

By

SHAGUN
(UNIVERSITY ROLLNO: 2300290140167)
SAKSHI GUPTA
(UNIVERSITY ROLLNO: 2300290140154)
SARTHAK GUPTA
(UNIVERSITY ROLLNO: 2300290140159)
SHIVANSH DUBEY
(UNIVERSITY ROLLNO: 2300290140175)

Under the Supervision of

Dr. VIPIN KUMAR
Associate Professor
KIET Group of Institutions

Submitted to

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

DECLARATION

We hereby declare that the work presented in this report entitled **"AI QUIZIFY"**, was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, We shall be fully responsible and answerable.

SHAGUN (2300290140167)

SAKSHI GUPTA (2300290140154)

SARTHAK GUPTA (2300290140159)

SHIVANSH DUBEY (2300290140175)

ACKNOWLEDGEMENTS

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

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

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

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

SHAGUN (2300290140167)

SAKSHI GUPTA (2300290140154)

SARTHAK GUPTA (2300290140159)

SHIVANSH DUBEY (2300290140175)

CERTIFICATE

Certified that **Shagun (2300290140167), Sakshi Gupta (2300290140154), Sarthak Gupta (2300290140159), Shivansh Dubey (2300290140175)** have carried out the project work having **AI QUIZIFY**. (**Mini-Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU),Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

SHAGUN (2300290140167)

SAKSHI GUPTA (2300290140154)

SARTHAK GUPTA (2300290140159)

SHIVANSH DUBEY (2300290140175)

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Date:

Dr. VIPIN KUMAR
AssociateProfessor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner

Signature of External Examiner

Dr. Arun Kumar Tripathi
Head, Department of Computer Application
KIET Group of Institutions,
Ghaziabad

ABSTRACT

AI Quizify is an intelligent quiz platform designed to make learning interactive and engaging. Built using the MERN stack and styled with Tailwind CSS, it offers users a seamless experience to test and enhance their knowledge. With features like user authentication, customizable quiz topics, and dynamic scoring, AI Quizify caters to learners of all levels.

At the heart of the platform is its ability to generate quizzes dynamically using AI, ensuring a fresh and personalized experience for every user. Users can explore topics, attempt quizzes, and instantly view results, all within a user-friendly interface. The dashboard provides insights into performance, helping users track their progress effectively.

AI Quizify is more than a quiz app; it's a tool to make learning smarter. By combining modern web development techniques and AI capabilities, it delivers an efficient and engaging solution for educational challenges. Whether for students or professionals, AI Quizify empowers users to learn, assess, and grow in an interactive way.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Project Description

The project, *AI-Based Quiz Using MERN Stack and Tailwind CSS,* is a modern platform for conducting interactive quizzes tailored to programming topics. It is designed to help users enhance their knowledge of programming languages, including C, C++, Java, JavaScript, Node.js, and Python. The platform provides essential features such as secure user authentication, customizable quizzes, real-time scoring, and a dashboard to track progress.

The application is built using the MERN stack:

- **MongoDB** is used as the database to store user profiles, quiz questions, and scores.
- **Express.js and Node.js** handle server-side logic, routing, and API requests, ensuring efficient backend operations.
- **React.js** powers the frontend, creating a dynamic and interactive user experience with reusable components.

Tailwind CSS is utilized for designing a responsive and visually appealing user interface, ensuring the platform meets modern design standards.

The workflow is simple yet effective. Users create an account or log in securely, choose a programming topic, and attempt a quiz. Questions are fetched dynamically from the backend, and scores are displayed in real time upon submission. The dashboard maintains a history of quiz attempts, allowing users to monitor their progress over time.

The platform is accessible, scalable, and user-friendly, providing an engaging learning experience. While the current system does not include difficulty levels or adaptive questioning, it sets the foundation for future enhancements, such as leaderboards, advanced analytics, and mobile app integration.

In conclusion, this project is an innovative and efficient solution for self-paced learning and skill assessment in programming, addressing the needs of students and professionals alike. With its robust technology stack and intuitive design, it has the potential to evolve into a comprehensive education tool.

1.2 Project Scope

The scope of the project includes:

1. **User Management:** Secure login and signup features.
2. **Quiz Selection:** Allowing users to choose quizzes on programming topics.
3. **Scoring System:** Real-time evaluation of quiz answers and displaying scores.
4. **Dashboard:** Providing a user profile and history of quiz attempts.

The platform targets a wide range of users, from students preparing for exams to professionals looking to refine their programming skills. With its modular design, the project is scalable and adaptable, allowing future enhancements such as advanced analytics and a mobile-friendly version.

The application not only serves as a learning tool but also as an assessment system for educational institutions and training programs, making it versatile and impactful in various domains.

1.3 Future Scope

This project has significant potential for expansion and improvement:

1. **Difficulty Levels:** Adding options for quizzes categorized as easy, medium, and hard.
2. **Adaptive Questioning:** Adjusting the difficulty of questions based on user performance.
3. **Leaderboards:** Introducing competitive rankings to encourage active participation.
4. **Analytics:** Providing detailed performance metrics to help users identify strengths and weaknesses.
5. **Mobile Application:** Developing a mobile app for better accessibility and convenience.
6. **Multilingual Support:** Offering quizzes in multiple languages to reach a global audience.

With these enhancements, the project can become a comprehensive platform for both learning and skill assessment, appealing to a broader audience.

1.4 Identification of Need

In the digital age, self-paced learning tools are essential for skill development. Traditional quizzes are static and lack customization. This project addresses these gaps by creating an interactive, AI-driven quiz platform.

The need for such a platform arises from:

- **Limited Tools:** Existing platforms do not offer customizable quizzes or personalized assessments.
- **Skill Enhancement:** Users require a system that adapts to their learning pace and preferences.
- **Convenience:** Online access to topic-specific quizzes eliminates geographical and temporal barriers.

By catering to these needs, the project aims to bridge the gap between conventional assessment methods and modern digital solutions.

1.5 Problem Statement

The absence of an engaging, user-friendly quiz system for programming topics creates challenges for learners. Current platforms either lack flexibility or fail to track progress effectively.

This project resolves these issues by:

- Creating an intuitive interface for easy navigation.
- Offering customizable, topic-specific quizzes.
- Storing user performance data securely for future analysis.
- Enabling real-time scoring and feedback.

The system empowers users to test their skills, learn at their own pace, and track improvements over time.

1.6 Software/Technology Used in Project

VSCODE

Figure 1.1 : Vscode Logo

- **Description: VS Code is a lightweight and powerful source code editor with extensions for various programming languages.**
- **Role in Project: Used as the primary development environment for coding, debugging, and integration of frontend and backend modules.**

MongoDB

Figure 1.2:Mongodb Logo

- **Description**: MongoDB is a NoSQL database that stores data in a JSON-like format, making it highly flexible and scalable. It is used in this project to manage user profiles, quiz questions, and user scores.
- **Why MongoDB**: Its ability to handle large datasets and dynamic schema is ideal for storing quiz data, which can vary by topic or format.

Express.js

Figure 1.3:Express Js Logo

- **Description**: Express.js is a backend framework for Node.js that simplifies the development of web applications. It is used to create APIs and handle server-side routing efficiently.
- **Why Express.js**: It provides a simple and lightweight solution to manage backend logic and communicate with the database.

React.js

Figure 1.4:React Js Logo

- **Description**: React.js is a JavaScript library for building interactive and dynamic user interfaces. It powers the frontend of the application, ensuring responsiveness and a smooth user experience.
- **Why React.js**: React's component-based architecture makes it easy to develop reusable UI elements, enhancing development speed and consistency.

Node.js

Figure 1.5:Node Js Logo

- **Description**: Node.js is a runtime environment that enables server-side execution of JavaScript. It forms the core of the backend, handling requests, processing data, and ensuring fast responses.
- **Why Node.js**: Its event-driven and non-blocking architecture makes it ideal for real-time applications like quizzes.

Tailwind CSS

Tailwind CSS

Figure 1.6:Tailwind css Logo

- **Description**: Tailwind CSS is a utility-first CSS framework that helps create beautiful and responsive designs quickly. It provides pre-defined classes that speed up the styling process.
- **Why Tailwind CSS**: It allows developers to build custom, modern interfaces without writing extensive CSS from scratch.

1.6.1 Non-Functional Requirements

1. **Performance:** Handles concurrent users without delays.
2. **Usability:** Simple and intuitive design for all user types.
3. **Scalability:** Supports increasing numbers of users and quizzes.
4. **Security:** Protects sensitive user data and quiz results.

1.6.2 Functional Requirements

1. Users can create accounts and log in securely.
2. Quiz questions are fetched dynamically based on selected topics.
3. Users can submit answers and view scores instantly.
4. The dashboard displays quiz history and scores.

1.7 Project Schedule

The *AI-Based Quiz Using MERN Stack and Tailwind CSS* project is divided into five key phases, each with a set duration and clear tasks:

1. **Requirement Analysis (2 Weeks)**
   This phase involves gathering project requirements, defining the project scope, and planning the approach for development. It ensures a clear understanding of the project goals and features.

2. **Frontend Development (2 Weeks)**
   In this phase, the user interface is built using **React.js** and **Tailwind CSS**. It includes designing UI components, implementing user authentication, and ensuring the platform is responsive and user-friendly.

3. **Backend Development (2 Weeks)**
   The backend is developed using **Node.js**, **Express.js**, and **MongoDB**. This includes creating APIs to handle quiz data, user profiles, and storing information securely in the database.

4. **Integration and Testing (1 Week)**
   This phase integrates the frontend and backend, followed by testing to ensure that all functionalities are working as expected. Bugs are fixed, and performance optimizations are applied.

5. **Deployment (1 Week)**
   The final phase involves deploying the application to a live environment, performing final checks, and launching the platform for user access.

CHAPTER 2

FEASIBILITY STUDY

2.1 Introduction

The feasibility study is an essential part of the initial planning phase for any project. It assesses the practicality of the project from various perspectives—technical, economic, and operational—to determine if the project can be successfully implemented within the given constraints of time, budget, and resources. For the **AI-Based Quiz Using MERN Stack and Tailwind CSS**, the feasibility study evaluates these factors to ensure the project is viable, both from a development and operational standpoint.

This project aims to create an interactive, web-based quiz platform focused on programming languages such as C, C++, Java, JavaScript, Node.js, and Python. The system allows users to log in, select quiz topics, take quizzes, and view their performance via real-time scoring. The platform is built using the **MERN stack** (MongoDB, Express.js, React.js, and Node.js) and **Tailwind CSS** for the frontend. These technologies ensure a modern, responsive user interface and a robust, scalable backend capable of handling multiple users and providing a seamless experience.

The feasibility study is crucial for identifying potential challenges, evaluating the project's overall viability, and ensuring it meets the user needs while staying within budget and time limits. It ensures that the technologies and methodologies selected are suitable for the project, that the project is financially feasible, and that it can be smoothly implemented and maintained once deployed.

The study evaluates the following key aspects:

- **Technical Feasibility**: This examines whether the required technology stack (MERN stack and Tailwind CSS) is suitable for building the system and whether the development team has the necessary technical expertise to implement the project.

- **Economic Feasibility**: This aspect assesses whether the project's benefits justify the costs involved, including development, hosting, and maintenance. Given the use of open-source technologies, the project is expected to be cost-effective.**Operational Feasibility**: Operational feasibility focuses on how well the project can be integrated into the real-world environment and whether it will function effectively for the end-users. It looks at ease of use, scalability, and ongoing maintenance.

2.2 Main Aspects

The feasibility study for the *AI-Based Quiz Using MERN Stack and Tailwind CSS* focuses on three main aspects: **technical feasibility**, **economic feasibility**, and **operational feasibility**. Each of these aspects plays a vital role in ensuring the project is viable, practical, and effective.

2.2.1 Technical Feasibility

Technical feasibility determines if the technology needed for the project is available, reliable, and suitable for the project's requirements. For this AI-based quiz platform, we are using the **MERN stack** (MongoDB, Express.js, React.js, Node.js), which is a widely used, modern technology stack for building web applications. It is known for its flexibility, scalability, and extensive developer support. MongoDB will be used for data storage, Express.js and Node.js for backend API development, and React.js for building a dynamic frontend.

These technologies are all open-source, have active communities, and offer a wealth of resources to support development. The team is proficient in these technologies, ensuring that the project is technically feasible. The stack supports the creation of responsive, high-performance applications capable of handling high traffic loads, which is essential for the quiz platform.

2.2.2 Economic Feasibility

Economic feasibility evaluates whether the project's cost is justified by the benefits. This includes assessing the development costs, resources needed, and potential return on investment. The primary cost components for this project include the costs associated with **cloud hosting**, developer time, and any necessary tools or third-party services. Since the core technologies (MongoDB, Node.js, React.js, Tailwind CSS) are open-source, licensing costs are minimal, making the project cost-effective.

The platform is expected to generate significant value through user engagement, especially in educational sectors where interactive learning tools are in demand. By offering a free-to-use platform with a potential premium model in the future (e.g., advanced quiz features, premium quizzes), the project can generate revenue, ensuring long-term sustainability.

2.2.3 Operational Feasibility

Operational feasibility examines whether the project can be effectively implemented and maintained. For the AI-Based Quiz platform, operational feasibility assesses how easy it will be for users to interact with the system and whether the platform can scale effectively to accommodate a growing user base.

The platform is designed to be **user-friendly**, with intuitive navigation and clear instructions. By utilizing **React.js** for frontend development, the system ensures that users can have a responsive and seamless experience across different devices. Additionally, the system is built to be scalable, allowing for increased usage without significant redesigns or performance issues.

From an administrative standpoint, the system will be easy to maintain, with manageable server requirements and simple database updates. Since most of the development stack is built on widely adopted technologies, ongoing operational support and maintenance can be handled efficiently.

2.3 Benefits

The *AI-Based Quiz Using MERN Stack and Tailwind CSS* project offers several significant benefits, which contribute to its attractiveness and long-term viability.

- **Enhanced Learning Experience**: Users can engage in interactive quizzes on programming languages like **C, C++, Java, JavaScript, Node.js, and Python**, improving their knowledge and skills.
- **Scalability**: The system is scalable and can easily be expanded to include new quiz topics, user features, or increased traffic.
- **Cost-Effective**: The use of open-source technologies minimizes licensing costs, making the project financially viable while delivering high-quality results.
- **Real-Time Feedback**: The platform provides immediate scoring and feedback to users, enhancing the learning process by allowing users to learn from their mistakes instantly.
- **User Engagement**: By offering customizable quizzes, the platform keeps users engaged and motivated to return regularly to test their knowledge.

These benefits make the project a valuable learning tool for users while ensuring the project remains cost-effective and sustainable in the long run.

2.4 SRS (Software Requirements Specification)

The **Software Requirements Specification (SRS)** is a critical document that defines the functional and non-functional requirements for the project. It provides detailed guidelines on how the software should function and the standards it should meet.

- **Functional Requirements**: These outline the core features of the AI-Based Quiz platform. Key functionalities include user login and signup, quiz creation and selection, real-time scoring, and a user dashboard to track progress and scores. Users can select quizzes based on their preferred programming languages and topics, answer questions, and see their results immediately.
- **Non-Functional Requirements**: These describe how the system should perform, focusing on scalability, security, performance, and user-friendliness. The system must be able to handle a high number of users simultaneously, with fast response times. Security is a priority, ensuring that user data is protected, especially during the authentication process.
- **System Requirements**: The platform will be web-based and will run on modern browsers such as Chrome, Firefox, and Edge. The user interface will be responsive, ensuring compatibility with both desktop and mobile devices.

CHAPTER 3

DESIGN AND PLANNING

3.1 INTRODUCTION

The AI Quizify project is designed to provide an engaging and interactive quiz platform where users can participate in quizzes generated by AI. The system aims to offer a seamless experience for users to log in, select subjects, answer questions, and view scores in real-time. The design phase of this project involves understanding the requirements, creating the system architecture, and ensuring that the design meets both user needs and technical specifications.

3.2 SYSTEM ANALYSIS

System analysis involves gathering detailed information about the AI Quizify application to understand its operations and the relationships between different components. This process ensures that all requirements are documented and analyzed before moving on to the design phase. The system is analyzed through the following steps:

- Information Gathering: Collect data on the functional requirements for the AI Quizify platform.
- Identification of Needs: Determine the essential features, including user authentication, quiz generation, score tracking, and leaderboard functionality.
- System Planning and Initial Investigation: Outline the project scope, objectives, and user goals.

- Feasibility Study: Analyze the technical feasibility of using AI for quiz generation, database integration, and real-time score tracking.

The primary goal of system analysis is to understand the operations and data flow within the system and identify any potential challenges.

3.3 SDLC

Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.

SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a Software Product.

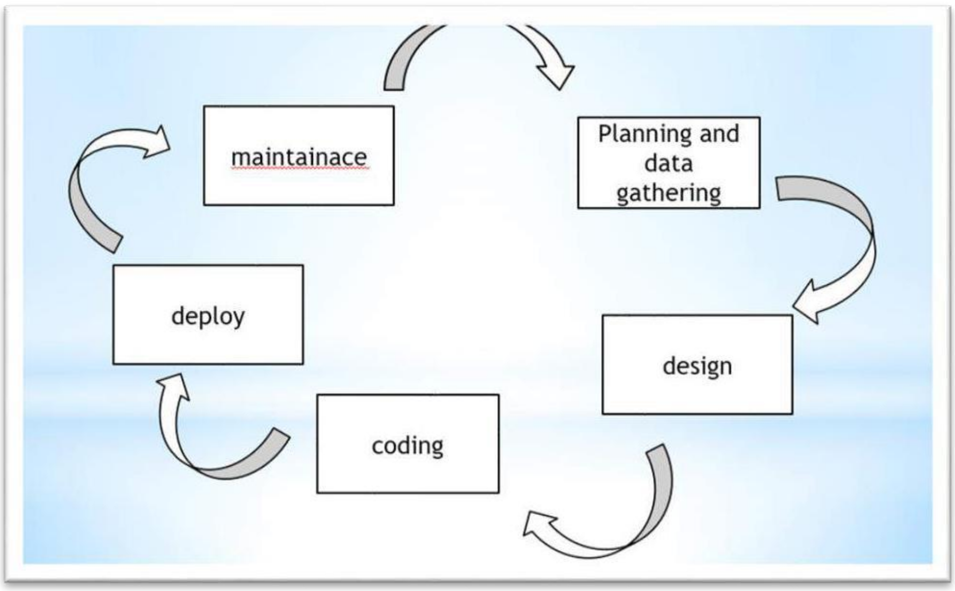


Figure 3.1: SDLC Phases

SDLC Phases

Given below are the various phases:

- Requirement gathering and analysis
- Design
- Implementation or coding

- Testing
- Deployment
- Maintenance

The development of AI Quizify follows the Software Development Life Cycle (SDLC), which is crucial for ensuring the quality and functionality of the software. The SDLC phases include:

1. **Requirement Gathering and Analysis**: Understanding user needs, such as quiz topics, login preferences, and scoreboards.
2. **Design**: Developing the system architecture and detailed designs for the front-end and back-end components.
3. **Implementation**: Coding the application, including user authentication, quiz generation, and score tracking features.
4. **Testing**: Testing the application to identify and resolve bugs or issues.
5. **Deployment**: Deploying the application on a server or cloud platform.
6. **Maintenance**: Regular updates and bug fixes to ensure the system continues to operate smoothly.

3.4 USE CASE DIAGRAM

Use-case diagrams model the behavior of a system and help to capture the requirements of the system. Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.
A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high level functionality of a system and also tells how the user handles a system.
Purposes of a use case diagram given below:
1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

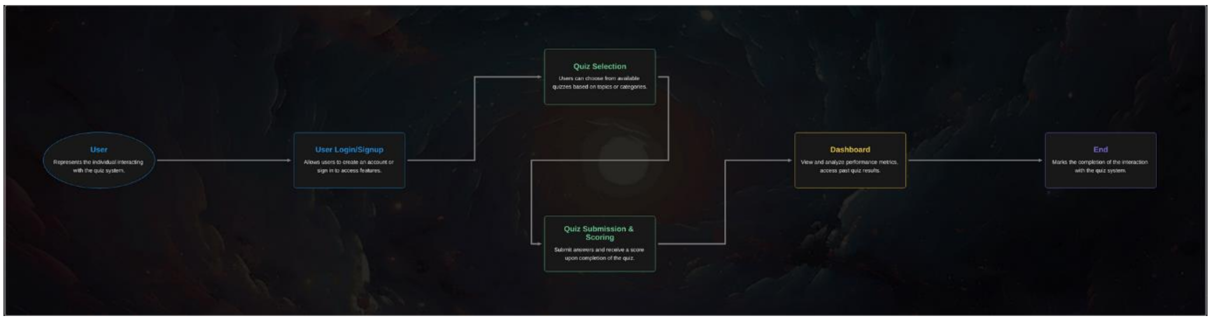

User
Represents the individual interacting with the quiz system.
User Login/Signup
Allows users to create an account or sign in to access features.
Quiz Selection
Users can choose from available quizzes based on topics or categories.
Quiz Submission & Scoring
Submit answers and receive a score upon completion of the quiz.
Dashboard
View and analyze performance metrics, access past quiz results.
End
Marks the completion of the interaction with the quiz system.

Figure 3.2:UseCase Diagram

3.5 Data Flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both. It shows how data enters and leaves the system, what changes the information, and where data is stored. The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

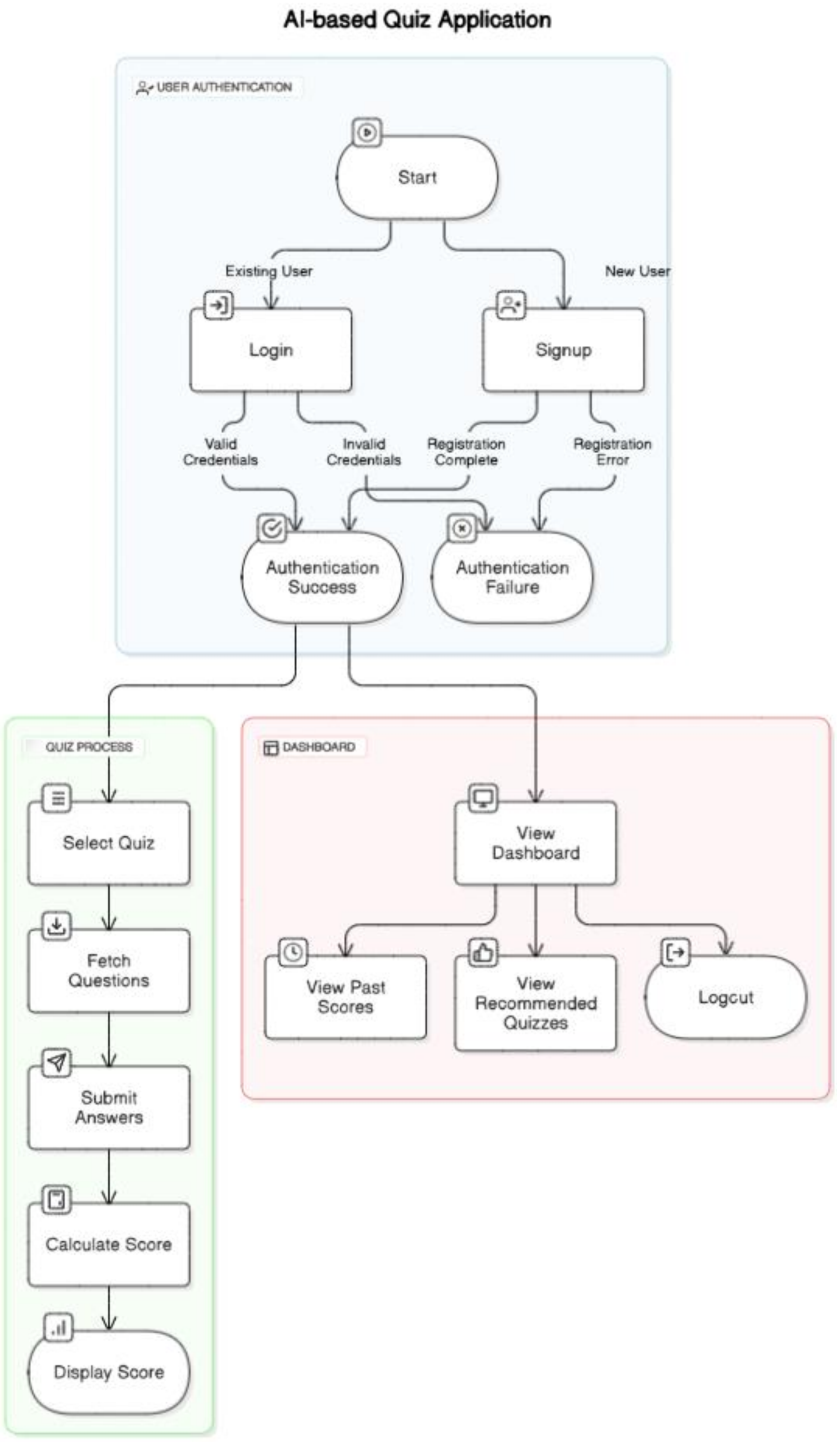


Figure 3.3:DFD

3.6 Flow Chart

A flowchart is a diagram that depicts a process, system or computer algorithm. They are widely used in multiple fields to document, study, plan, improve and communicate often complex processes in clear, easy-to-understand diagrams. Flowcharts, sometimes spelled as flow charts, use rectangles, ovals, diamonds and potentially numerous other shapes to define the type of step, along with connecting arrows to define flow and sequence.

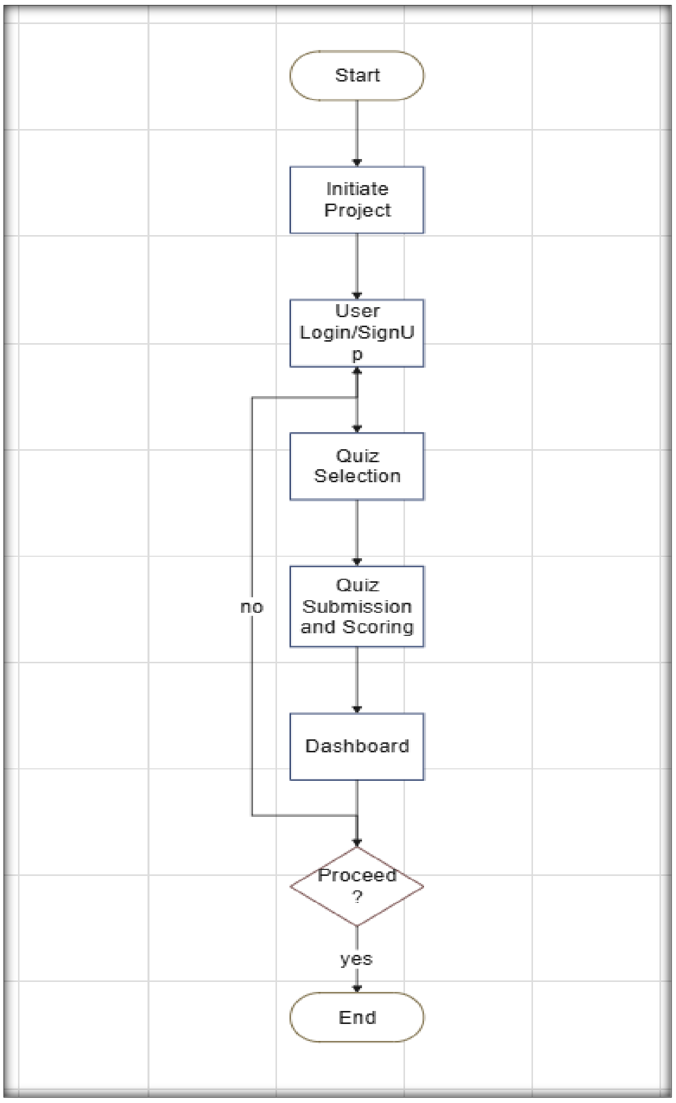


Figure 3.4:Mongodb Logo

CHAPTER 4

REPORT

4.1 GIST

AI Quizify is a web application designed to allow users to sign up, log in, select a quiz subject, and attempt quizzes with AI-generated questions. The system prioritizes user security with an authentication process using email and password. Real-time score tracking ensures a dynamic experience, and users can see top scores on a leaderboard. To enhance the experience, the application allows users to view and share their progress and answers with friends or peers. The AI-driven quiz generation ensures that the questions are relevant and challenging.

4.2 SOME SNIPPETS

4.2.1 WEBSITE

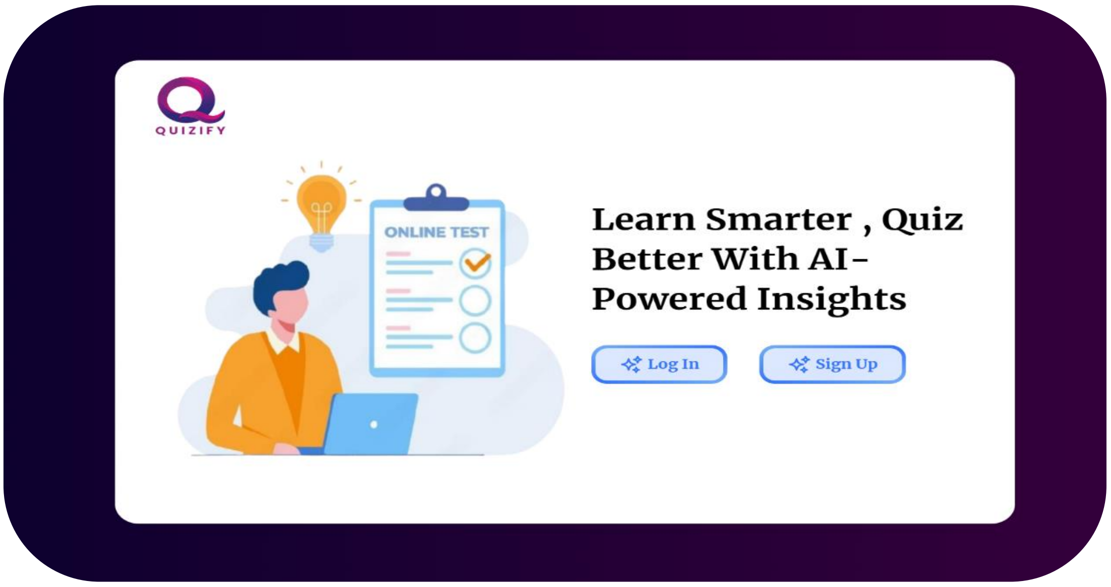


Figure 4.1:Website Landing

- **Signup Page:** The Signup page of AI Quizify where users authenticate themselves via email and password.

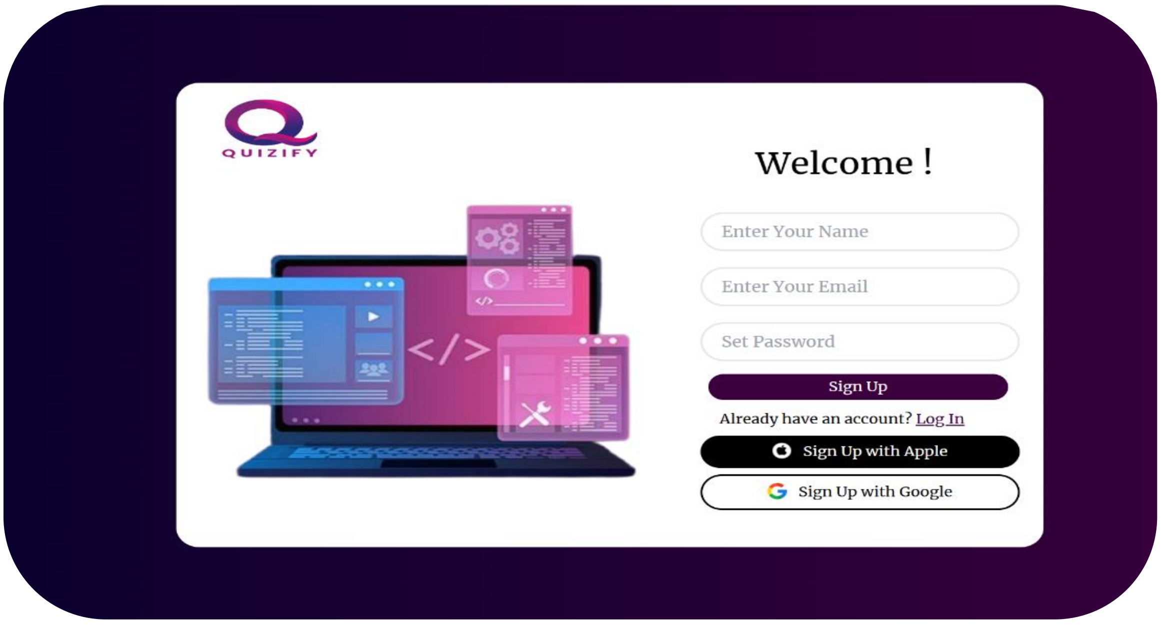


Figure 4.2:Signup page

➢ **Login Page:** The login page of AI Quizify where users authenticate themselves via email and password.

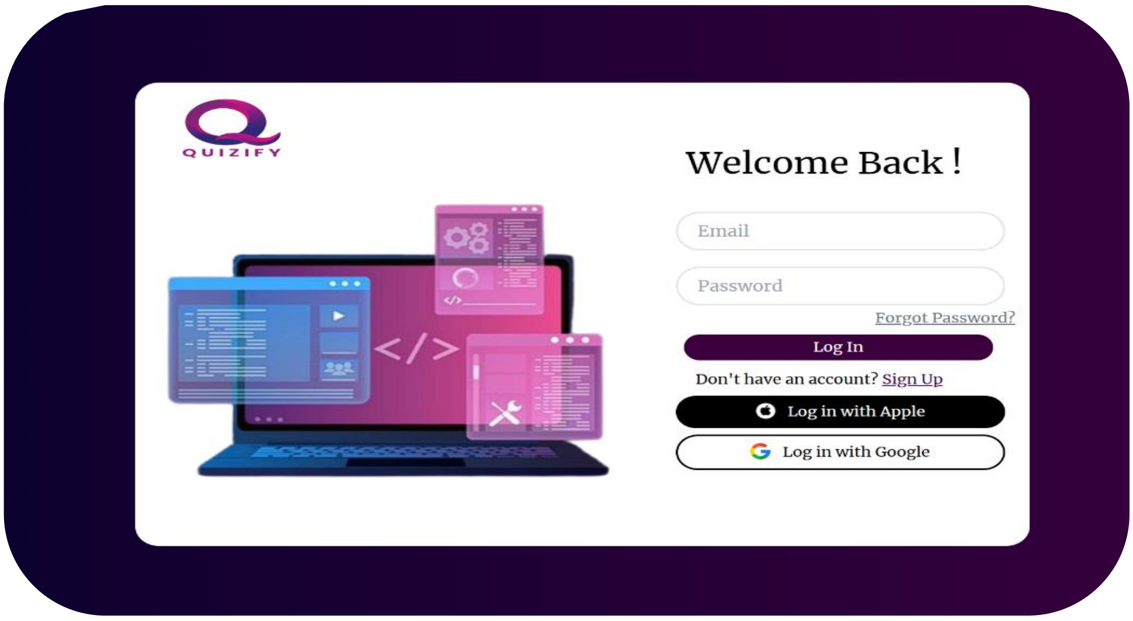
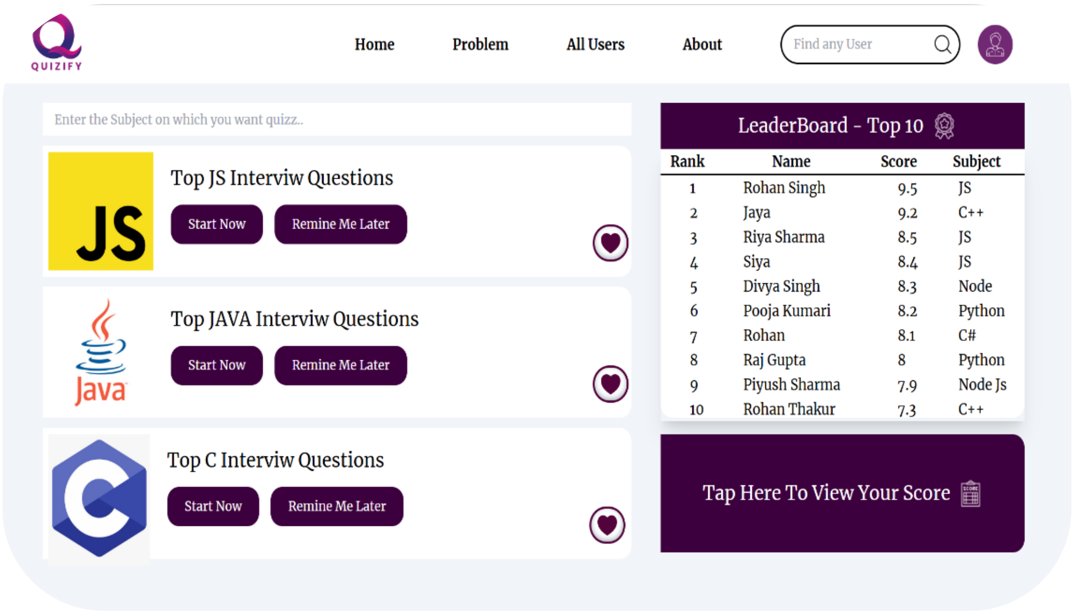


Figure 4.3:Login Page

➢ **Quiz Selection:** Users can choose the subject of their quiz from a dropdown menu.

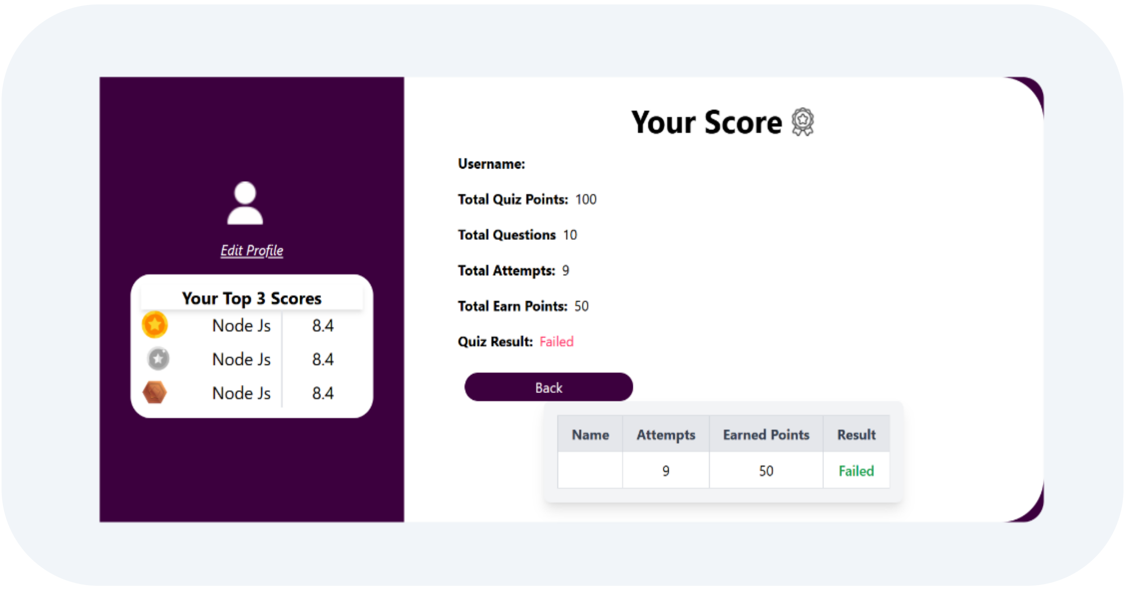


LeaderBoard - Top 10

Rank	Name	Score	Subject
1	Rohan Singh	9.5	JS
2	Jaya	9.2	C++
3	Riya Sharma	8.5	JS
4	Siya	8.4	JS
5	Divya Singh	8.3	Node
6	Pooja Kumari	8.2	Python
7	Rohan	8.1	C#
8	Raj Gupta	8	Python
9	Piyush Sharma	7.9	Node Js
10	Rohan Thakur	7.3	C++

Figure 4.4:Quiz Selection Page

➢ **Scoring:** The leaderboard displays the top scorers for each subject.



Node Js	8.4
Node Js	8.4
Node Js	8.4

Your Score

Username:

Total Quiz Points: 100

Total Questions 10

Total Attempts: 9

Total Earn Points: 50

Quiz Result: Failed

Name	Attempts	Earned Points	Result
	9	50	Failed

Figure 4.5:Scoring Page

4.2.2 DATABASE

The backend of AI Quizify uses MongoDB to store user information, quiz results, and leaderboard data. All the user data is securely stored and accessed efficiently. The database schema includes collections for users, quizzes, and scores.

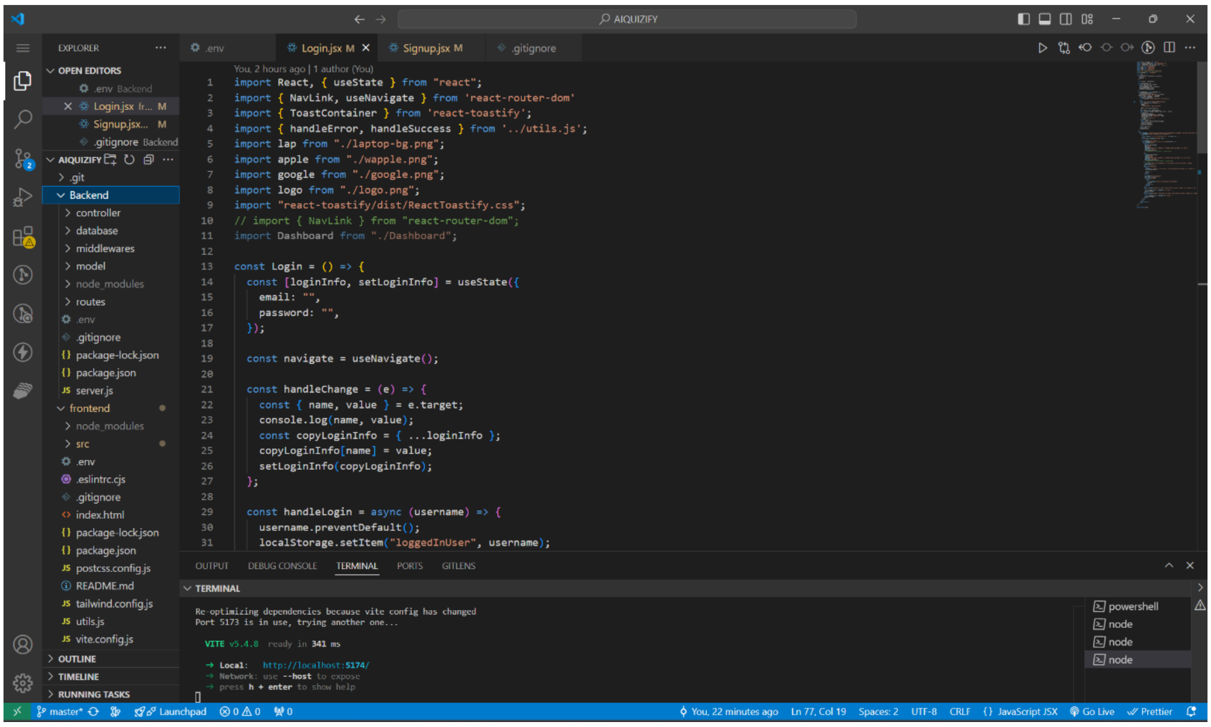
4.2.3 TECHNOLOGIES USED

- **Frontend:** The application's user interface is built using Angular for seamless interaction and responsiveness**.**

- **Backend:** The backend server is powered by Node.js and Express, while data is stored and retrieved using MongoDB.

- **AI Integration:** The AI-driven quiz generation is implemented through an integration with GPT.

4.2.4 VISUAL STUDIO CODE



```
import React, { useState } from "react";
import { NavLink, useNavigate } from 'react-router-dom'
import { ToastContainer } from 'react-toastify';
import { handleError, handleSuccess } from '../utils.js';
import lap from "./laptop-bg.png";
import apple from "./wapple.png";
import google from "./google.png";
import logo from "./logo.png";
import "react-toastify/dist/ReactToastify.css";
// import { NavLink } from "react-router-dom";
import Dashboard from "./Dashboard";

const Login = () => {
  const [loginInfo, setLoginInfo] = useState({
    email: "",
    password: "",
  });

  const navigate = useNavigate();

  const handleChange = (e) => {
    const { name, value } = e.target;
    console.log(name, value);
    const copyLoginInfo = { ...loginInfo };
    copyLoginInfo[name] = value;
    setLoginInfo(copyLoginInfo);
  };

  const handleLogin = async (username) => {
    username.preventDefault();
    localStorage.setItem("loggedInUser", username);
```

```
Re-optimizing dependencies because vite config has changed
Port 5173 is in use, trying another one...

  VITE v5.4.8  ready in 341 ms

  ➜  Local:   http://localhost:5174/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

Figure 4.6: Visual Studio Window

CHAPETR 5

CODING

This chapter contains some codes of the project. The goal of the coding is to translate the design of the system into code in a given programming language. For a given design, the aim of this phase is to implement the design in the best possible manner. The coding phase affects both testing and maintenance profoundly.

Some Codes are as Written below:

Signup.jsx

```
import React, { useState } from 'react'
import { NavLink, useNavigate } from 'react-router-dom'
import { ToastContainer, toast } from 'react-toastify';
import { handleError, handleSuccess } from '../utils';

import lap from "./laptop-bg.png";
import apple from "./wapple.png";
import google from "./google.png";
import logo from "./logo.png";
import "react-toastify/dist/ReactToastify.css";
// import { NavLink } from 'react-router-dom';
import Dashboard from './Dashboard';
```

```
const Signup = () => {
  const [signupInfo, setSignupInfo] = useState({
    name: '',
    email: '',
    password: ''
})

const navigate = useNavigate();
const handleChange = (e) => {
    const { name, value } = e.target;
    console.log(name, value);
    const copySignupInfo = { ...signupInfo };
    copySignupInfo[name] = value;
    setSignupInfo(copySignupInfo);
}
console.log('signup info->',signupInfo);

const handleSignup = async (e) => {
    e.preventDefault();
    const { name, email, password } = signupInfo;
    if (!name || !email || !password) {
        return handleError('name, email and password are required')
    }
    try {
        const url = `http://localhost:4004/auth/signup`;
        const response = await fetch(url, {
            method: "POST",
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(signupInfo)
        });
        const result = await response.json();
        const { success, message, error } = result;
        if (success) {
            handleSuccess(message);
            setTimeout(() => {
                navigate('/login')
            }, 1000)
        } else if (error) {
            const details = error?.details[0].message;
            handleError(details);
        } else if (!success) {
            handleError(message);
        }
        console.log(result);
    } catch (err) {
        handleError(err);
    }
}
  return (
    <div className="w-screen h-screen bg-gradient-to-r from-[#07012d] to-[#3c003e]  flex
place-items-center justify-center font-classy">

    <div className="w-3/5 h-3/4 bg-white flex rounded-3xl ">
    <div>
    <img className=" ml-6 mt-3 w-40 absolute h-24 " src={logo} alt="" />
    </div>
      <img src={lap} className="mt-[70px]" alt="" />
      <div className="pt-20 -ml-8">
           <h1 className="text-4xl text-center -ml-14">Welcome !</h1>
           <div className="mt-10" >
             <input
               onChange={handleChange}
               type='text'
               name='name'
              className="border-[2px] rounded-full  w-[330px] border-gray-200 p-2 px-5 text-
lg "
              placeholder="Enter Your Name"
              value={signupInfo.name}
            />
            <input
             onChange={handleChange}
             type='email'
             name='email'
              className="border-[2px] rounded-full  w-[330px] border-gray-200 p-2 px-5 mt-5
text-lg "
              placeholder="Enter Your Email"
              value={signupInfo.email}
            />
            <input
               onChange={handleChange}
               type='password'
               name='password'
              className="border-[2px] rounded-full  w-[330px] border-gray-200 p-2 px-5 mt-5
text-lg "
              placeholder="Set Password"
              value={signupInfo.password}
            />
             <NavLink to="/Dashboard">
            <button className=" w-[310px] ml-2 text-white p-1 hover:bg-[#57015a] bg-
[#3c003e] mt-4 rounded-full" onClick={handleSignup} >
```

```
            Sign Up
          </button>
          </NavLink>

          <h3 className="text-s mt-3 ml-5 ">
            Already have an account?{" "}
            <NavLink to="/Login">
            <button >
              {" "}
              <span className="underline cursor-pointer  text-[#3c003e]">
                Log In
              </span>
            </button>
            </NavLink>
          </h3>
          <button className="bg-black flex gap-3 justify-center text-white w-[330px] mt-2
rounded-full p-2">
            {" "}
            <img src={apple} alt="" className="w-5 mt-[1px] " />
            Sign Up with Apple
          </button>
          <button className="border-2 border-black flex gap-3 justify-center text-black w-
[330px] mt-2 rounded-full p-2">
            {" "}
            <img src={google} alt="" className="w-5 mt-[1px] " />
            Sign Up with Google
          </button>
        </div>
      </div>

    </div>
    <ToastContainer />
    </div>
  )
}

export default Signup
```

Login.jsx

```
import React, { useState } from "react";
import { NavLink, useNavigate } from 'react-router-dom'
import { ToastContainer } from 'react-toastify';
import { handleError, handleSuccess } from '../utils.js';
import lap from "./laptop-bg.png";
import apple from "./wapple.png";
import google from "./google.png";
import logo from "./logo.png";
import "react-toastify/dist/ReactToastify.css";
// import { NavLink } from "react-router-dom";
import Dashboard from "./Dashboard";

const Login = () => {
  const [loginInfo, setLoginInfo] = useState({
    email: "",
    password: "",
  });

  const navigate = useNavigate();

  const handleChange = (e) => {
    const { name, value } = e.target;
    console.log(name, value);
    const copyLoginInfo = { ...loginInfo };
    copyLoginInfo[name] = value;
    setLoginInfo(copyLoginInfo);
  };

  const handleLogin = async (username) => {
    username.preventDefault();
    localStorage.setItem("loggedInUser", username);
  console.log("User logged in:", username);
    const { email, password } = loginInfo;
    if (!email || !password) {
      return handleError("email and password are required");
    }
    try {
      const url = `http://localhost:4004/auth/login`;
      const response = await fetch(url, {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(loginInfo),
      });
      const result = await response.json();
      const { success, message, jwtToken, name, error } = result;
      if (success) {
        handleSuccess(message);
        localStorage.setItem("token", jwtToken);
        localStorage.setItem("loggedInUser", name);
        setTimeout(() => {
          navigate("/dashboard");
        }, 1000);
      } else if (error) {
        const details = error?.details[0].message;
        handleError(details);
      } else if (!success) {
        handleError(message);
      }
      console.log(result);
    } catch (err) {
      handleError(err);
    }
  };
  return (
    <div className="w-screen h-screen bg-gradient-to-r from-[#07012d] to-[#3c003e]  flex
place-items-center justify-center font-classy ">
      <div className="w-3/5 h-3/4 bg-white flex rounded-3xl ">
        <div>
          <img className=" ml-6 mt-3 w-40 absolute h-24 " src={logo} alt="" />
        </div>
        <img src={lap} className="mt-[70px]" alt="" />

        <div className="pt-20 -ml-8">
          <h1 className="text-4xl text-center -ml-14">Welcome Back !</h1>
          <div className="mt-10">
            <input
              onChange={handleChange}
              type="email"
              name="email"
              className="border-[2px] rounded-full  w-[330px] border-gray-200 p-2 px-5 text-lg
"
              placeholder="Email"
              value={loginInfo.email}
              // onChange={(e) => (user.email = e.target.value)}
            />
            <input
               onChange={handleChange}
               type='password'
               name='password'
              className="border-[2px] rounded-full  w-[330px] border-gray-200 p-2 px-5 mt-5
text-lg "
              placeholder="Password"
              value={loginInfo.password}
              // onChange={(e) => (user.password = e.target.value)}
            />
            <h6 className="underline text-gray-500 mt-1 cursor-pointer hover:text-gray-
700  ml-[200px]">
              Forgot Password?{" "}
            </h6>
            <NavLink to="/Dashboard">
              <button
                className=" w-[310px] ml-2 text-white p-1 hover:bg-[#57015a] bg-[#3c003e] mt-
2 rounded-full"
                onClick={handleLogin}
              >
                Log In{" "}
              </button>
            </NavLink>

            <h3 className="text-s mt-3 ml-5 ">
              Don't have an account?{" "}
              <NavLink to="/Signup">
                <button>
                  {" "}
                  <span className="underline cursor-pointer text-[#3c003e]">
                    Sign Up
                  </span>
                </button>{" "}
              </NavLink>
            </h3>
            <button className="bg-black flex gap-3 justify-center text-white w-[330px] mt-2
rounded-full p-2">
              {" "}
              <img src={apple} alt="" className="w-5 mt-[1px] " />
              Log in with Apple
            </button>
            <button className="border-2 border-black flex gap-3 justify-center text-black w-
[330px] mt-2 rounded-full p-2">
              {" "}
              <img src={google} alt="" className="w-5 mt-[1px] " />
              Log in with Google
            </button>
          </div>
        </div>
      </div>
      <ToastContainer />
    </div>
  );
};

export default Login;
```

Quiz.jsx

```
import React, { useEffect, useState } from 'react';
import { MoveNextQuestion, MovePrevQuestion } from './hooks/FetchQuestion.js';
import { PushAnswer } from './hooks/setResult.js';
import { Navigate } from 'react-router-dom'
import { useSelector, useDispatch } from 'react-redux';
import Questions from './Questions.jsx';

const Quiz = () => {
  const [check, setChecked] = useState(undefined);

  const result = useSelector((state) => state.result.result);
  const { queue, trace } = useSelector((state) => state.questions);
  const dispatch = useDispatch();

  useEffect(() =>{
    console.log(result);
  })

  function onNext() {
    if (trace < queue.length) {
      /** increase the trace value by one using MoveNextAction */
      dispatch(MoveNextQuestion());

      /** insert a new result in the array.  */
      if (result.length <= trace) {
        dispatch(PushAnswer(check));
      }
    }
    setChecked(undefined);
  }

  function onPrev() {
    if (trace > 0) {
      /** decrease the trace value by one using MovePrevQuestion */
      dispatch(MovePrevQuestion());
    }
  }

  function onChecked(check) {
    setChecked(check);
  }
  if(result.length && result.length >= queue.length){
    return <Navigate to={'/Score'} replace={true}></Navigate>
}

  return (
    <div className="w-screen flex justify-center bg-slate-100 items-center h-screen">
      <div className="flex flex-col w-3/4 h-[500px] text-white p-16 rounded-r-3xl bg-
[#3c003e] gap-3 ml-6 text-xl">
        <h1 className="text-2xl font-bold">Question 1 : </h1>

        <Questions onChecked={onChecked} />
        <div className="flex gap-4">
          {trace > 0 ? (
            <button
              className="bg-white hover:text-[#3c003e] hover:scale-105 transition-transform
duration-150 text-black p-2 w-32 text-lg font-bold rounded-2xl"
              onClick={onPrev}
            >
              Prev
            </button>
          ) : null}
          <button
            className="bg-white hover:text-[#3c003e] hover:scale-105 transition-transform
duration-150 text-black p-2 w-32 text-lg font-bold rounded-2xl"
            onClick={onNext}
          >
            Next
          </button>
        </div>
      </div>
    </div>
  );
};

export default Quiz;
```

Score.jsx

```
import React, { useEffect, useState } from "react";
import gold from "./1st.png";
import silver from "./2nd.png";
import bronze from "./3rd.png";
import user2 from "./user2.png";
import top from "./top3.png";
import Dashboard from "./Dashboard";
import { useNavigate } from "react-router-dom";
import { useDispatch, useSelector } from "react-redux";
import {
  attempts_Number,
  earnPoints_Number,
  flagResult,
} from "./helper/helper.js";
import { resetAllAction } from "./redux/question_reducer";
import { resetResultAction } from "./redux/result_reducer";
import { handleError, handleSuccess } from "../utils";

import ScoreTable from "./ScoreTable.jsx";

const Score = () => {
  const dispatch = useDispatch();
  const [loggedInUser, setLoggedInUser] = useState("");
  const {
    questions: { queue, answers },
    result: { result, userId },
  } = useSelector((state) => state);

  // useEffect(() => {
  //   setLoggedInUser(localStorage.getItem("loggedInUser"));
  // }, []);
  useEffect(() => {
    const user = localStorage.getItem("loggedInUser");
    if (user) {
      setLoggedInUser(user);
    } else {
      console.error("No user found in localStorage");
    }
  }, []);

  const totalPoints = queue.length * 10;
  const attempts = attempts_Number(result);
  const earnPoints = earnPoints_Number(result, answers, 10);
  const flag = flagResult(totalPoints, earnPoints);

  const navigate = useNavigate();
  const onBack = (e) => {
    localStorage.removeItem("token");
    localStorage.removeItem("loggedInUser");
    handleSuccess("User Loggedout");
    dispatch(resetAllAction());
    dispatch(resetResultAction);
    setTimeout(() => {
      navigate("/Dashboard");
    }, 1000);
```

```
 };

 return (
   <div className="w-screen flex justify-center bg-slate-100 items-center h-screen">
     <div className="w-3/4 h-[500px] flex rounded-r-3xl bg-[#3c003e]">
       {/* Profile Section */}
       <div className="w-1/3 text-white text-center flex flex-col gap-4 justify-center items-
center">
         <div className="flex items-center">
           <img src={user2} className="w-12 " alt="User" />
           <h1 className="text-5xl font-bold cursor-pointer ml-4">
             {loggedInUser}
           </h1>
         </div>
         <h1 className="cursor-pointer underline hover:scale-105 transition-transform
duration-200 italic underline-offset-2">
           Edit Profile
         </h1>
         <div className="bg-white text-black p-3 rounded-3xl w-72">
           <h3 className="text-xl font-bold shadow-md">Your Top 3 Scores</h3>
           <div className="flex flex-col gap-3 text-xl">
             <table>
               <tbody>
                 <tr>
                   <td>
                     <img src={gold} alt="Gold" className="w-8" />
                   </td>
                   <td className="w-24 border-r-2">Node Js</td>
                   <td className="w-24">8.4</td>
                 </tr>
                 <tr>
                   <td>
                     <img src={silver} alt="Silver" className="w-10" />
                   </td>
                   <td className="w-24 border-r-2">Node Js</td>
                   <td className="w-24">8.4</td>
                 </tr>
                 <tr>
                   <td>
                     <img src={bronze} alt="Bronze" className="w-8" />
                   </td>
                   <td className="w-24 border-r-2">Node Js</td>
                   <td className="w-24">8.4</td>
                 </tr>
               </tbody>
             </table>
```

```
          </div>
        </div>
      </div>

      {/* Score Section */}
      <div className="bg-white w-[70%] rounded-r-[50px] flex flex-col items-center p-4">
        <div className="flex items-center gap-2 justify-center">
          <h1 className="text-4xl font-bold text-center mt-3">Your Score</h1>
          <img src={top} className="h-8 mt-4" alt="Top" />
        </div>

        {/* Score Details */}
        <div className="w-full flex flex-col items-start gap-4 mt-4 pl-12">
          <div className="flex gap-2">
            <span className="font-bold">Username:</span>
            <span>{loggedInUser}</span>
          </div>
          <div className="flex gap-2">
            <span className="font-bold">Total Quiz Points:</span>
            <span>{totalPoints || 0}</span>
          </div>
          <div className="flex gap-2">
            <span className="font-bold">Total Questions</span>
            <span>{queue.length || 0}</span>
          </div>
          <div className="flex gap-2">
            <span className="font-bold">Total Attempts:</span>
            <span>{attempts || 0}</span>
          </div>
          <div className="flex gap-2">
            <span className="font-bold">Total Earn Points:</span>
            <span>{earnPoints || 0}</span>
          </div>
          <div className="flex gap-2">
            <span className="font-bold">Quiz Result:</span>
            <span style={{ color: `${flag ? "#2aff95" : "#ff2a66"}` }}>
              {flag ? "Passed" : "Failed"}
            </span>
          </div>
          <button
            className="w-[200px] ml-2 text-white p-1 hover:bg-[#57015a] bg-[#3c003e] mt-2
rounded-full"
            onClick={onBack}
          >
            Back
          </button>
        </div>
        <div>
          <ScoreTable></ScoreTable>
        </div>
      </div>
    </div>
  </div>
);
};

export default Score;
```

CHAPTER 6

TESTING

6.1 Introduction

6.1.1 Testing Objectives

The following are the testing objectives:

-Testing is a process of executing a program with the intent of finding an error.

-A good test case is one that has a high probability of finding an as-yet-undiscovered error

-successful test is one that uncovers an as yet undiscovered error.

6.1.2 Testing Principles

The basic principles that guide software testing are as follows:

- -All tests should be traceable to customer requirements.

- -Tests should be planned long before testing begins.
- -The parate principle applies to software testing**.**

Pareto principle states that 80 percent of all errors uncovered during testing will likely be traceable to 20 percent of all program components.

Testing should begin “in the small “and progress toward testing “in the large.”

Exhaustive testing is not possible.

6.2 Level Of Testing

There are different level softesting

->UnitTesting

->IntegrationTesting

->SystemTesting

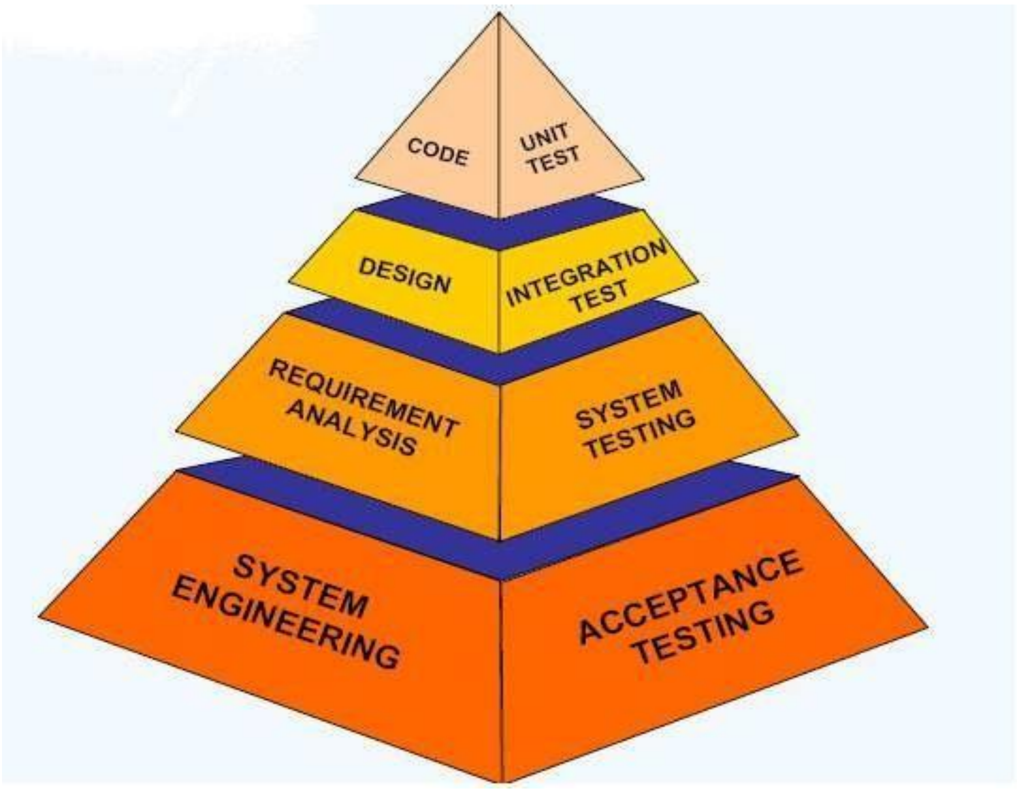


Figure5.1: Testing pyramid

Figure:6.1 Testing Levels

6.2.1 Unit testing

Unittestingfocusesverificationeffort onthesmallestunitofsoftwaredesign, the module. The important control parts are tested to uncover with in theboundary of the module. The module interface is tested to ensure that the information properly flows into and out of the program unit and boundary conditions are tested to ensure that the modules operate properly at boundaries established to limit or restrict processing. Test date is provided through testingscreens.

6.2.2 Integration testing

IntegratingtestingisasystematictechniqueforconstructingProgram structurewhile conducting tests to uncover error associates with interfacing. The objective isto take unit modules and built a program structure that has been directed by design.

• IntegrationTestingwilltestwhetherthemodulesworkwelltogether.

• Thiswillcheckwhetherthedesigniscorrect.

6.2.3 System testing

System testing is the process of testing the completed software as a part of the environmentitwascreatedfor.Itisdonetoensurethatalltherequirementsspec ifiedbythe customer are met. System testing involves functional testing and performance testing.

• SystemTesting willcontain thefollowing testing:

- FunctionalTesting.
- PerformanceTesting.

• FunctionTestingwilltesttheimplementation ofthebusinessneeds.

• PerformanceTestingwilltestthenon-functionalrequirementsofthesystemlikethespeed, load etc.

CHAPTER7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

AI Quizify is an advanced and interactive learning platform that uses AI to create personalized quizzes for users.

Leveraging the MERN stack and Tailwind CSS, it ensures a smooth, responsive, and visually appealing experience. The platform allows users to select from a variety of customizable quiz topics, ensuring relevance and engagement. With dynamic scoring and instant feedback, AI Quizify offers real-time performance insights, helping users identify areas for improvement. Whether you're a student looking to reinforce your knowledge or a professional aiming to test your expertise, AI Quizify is an effective tool for interactive learning.

Its intelligent quiz generation ensures fresh and challenging content every time, making learning both efficient and enjoyable.

The dashboard’s performance tracking further motivates users to keep improving, making AI Quizify an ideal educational tool for learners at all levels.

7.2 FUTURE SCOPE

- **More Topics**: Add quizzes on advanced subjects and professional courses.
- **Multilingual**: Offer quizzes in multiple languages for global users.
- **Custom Quizzes**: Use AI to adapt quizzes to user strengths and weaknesses.
- **Group Quizzes**: Enable team-based quizzes for collaborative learning.
- **Gamify It**: Add rewards, badges, and leader boards for fun.
- **Mobile App**: Create Android and iOS apps for on-the-go access.
- **Detailed Feedback**: Give clear, specific feedback on quiz performance.
- **School Integration**: Connect with schools' learning systems.
- **Voice Quizzes**: Add voice-enabled quizzes for hands-free use.
- **Live Competitions**: Host real-time quiz challenges for friendly competition.

7.3 BIBILOGRAPHY

- **MERN Stack Development**
- Books like *Learning Node.js Development* by Vohra and *Full-Stack React Projects* by Eisenberg helped us build the app’s backend and frontend.
- Resources like *Mastering MongoDB 4.x* guided us in handling data efficiently.
- **Tailwind CSS Styling**
- Guides such as *Tailwind CSS: A Modern Guide* by Studholme and *Tailwind CSS in Action* by Merta helped in designing a clean and responsive interface.
- **AI and ChatGPT Integration**
- OpenAI’s official *ChatGPT API Documentation* and papers like *Language Models are Few-Shot Learners* gave us the tools to integrate AI-powered quiz generation.
- **Dynamic Quiz Generation Using AI**
- Concepts from *Artificial Intelligence: A Modern Approach* by Russell and Norvig helped us understand AI-driven systems.
- Research articles on adaptive learning systems inspired the app’s personalized quiz experience.
- **AI in Education**
- *Intelligence Unleashed: An Argument for AI in Education* by Luckin et al. showed us how AI can improve learning.
- Books on AI-driven learning, like *AI-Driven Personalization in E-Learning*, offered practical insights.
- **User Experience and Design**
- Classics like *Don’t Make Me Think* by Krug and *The Elements of User Experience* by Garrett guided us in building a user-friendly interface.
- **Collaboration and Version Control**
- For teamwork, resources like *Version Control with Git* by Loeliger and *Pro Git* by Chacon were invaluable for managing our code on GitHub.

There are some websites also helped me in building this software. These are: - https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs
https://www.javatpoint.com/react
https://www.tutorialspoint.com/nodejs/index.htm

GITHUB PROJECT LINK

https://github.com/shagun1021/AiQuizify