A PROJECT REPORT ON

**Quiz application**

**A Report Submitted to**

# Blackbuck Engineers Pvt. Ltd

**Blackbuck Engineers Pvt. Ltd Road no 36, Jubilee Hills, Hyderabad**

**Submitted by**

**T.Shanmukha Nivas(20B91A12H6)**

**U.Jhansi(20B91A12H7)**

**G.Shanmukha Venkata pavan kumar((20B91A5719)**

**D.Naga Venkata Siva Sai(21B95A1203)**

**L.Madhu Babu(20B91A1203)**

# Acknowledgement

**The Quiz App project is a web-based application designed to provide users with an interactive platform for participating in quizzes across various categories. Users can customize the number of questions, select quiz categories, set difficulty levels, and specify the time per question. The application dynamically fetches quiz questions from an external API and presents them to the user in a user-friendly interface. After answering questions, users receive immediate feedback on their performance and can track their scores throughout the quiz session.**

# Abstract

The Quiz App project is a web-based application designed to offer users an interactive platform for participating in quizzes across various categories. The application allows users to customize their quiz sessions by selecting the number of questions, choosing specific categories, setting difficulty levels, and specifying time limits per question. Utilizing dynamic quiz generation, the app fetches quiz questions from an external API, ensuring a diverse and up-to-date pool of quiz content. Users receive real-time feedback on their answers, with correct and incorrect responses highlighted visually. The application features an intuitive and interactive interface with elements such as timers, progress bars, and question counters to enhance user engagement. At the end of the quiz, users receive their final score along with the total number of questions attempted, encouraging continuous learning and engagement.

# Introduction

In today's digital age, online quizzes have become a popular means of entertainment, education, and self-assessment. The Quiz App project aims to leverage the power of web technology to provide users with a seamless and engaging quiz experience. With the proliferation of information available on the internet, quizzes offer a structured and enjoyable way for users to test their knowledge, challenge themselves, and learn new facts across various subjects.

The motivation behind developing the Quiz App project lies in its potential to cater to the diverse needs and interests of users seeking to engage in quiz-based activities. By offering extensive customization options, dynamic quiz generation, real-time feedback, and interactive features, the application aims to enhance user satisfaction and promote active learning.

Through this project, we aim to showcase our proficiency in full-stack web development, incorporating frontend and backend technologies to create a robust and user-friendly application. By adhering to principles of user experience design, API integration, and gamification, we strive to deliver a high-quality quiz platform that meets the expectations and demands of modern users.

In the subsequent sections, we will delve deeper into the features, architecture, implementation details, and potential impact of the Quiz App project, highlighting its significance in the realm of online education and entertainment.

**Advantages:**

1. **Enhanced Learning:** Users can improve their knowledge and skills by participating in quizzes covering various topics.
2. **Engagement:** The interactive nature of quizzes can enhance user engagement and interest in the subject matter.
3. **Flexibility:** Users can access quizzes anytime and anywhere, providing flexibility and convenience.
4. **Scalability:** The application can easily scale to accommodate a growing user base and add more quizzes and features over time.
5. **Competition:** Leaderboards and scoring systems can encourage healthy competition among users, increasing participation and motivation.
6. **Data Insights:** The project can provide valuable insights into user preferences, performance trends, and popular quiz topics through data analytics.

**Disadvantages:**

1. **Technical Issues:** Users may encounter technical issues such as server downtime, bugs, or compatibility issues with different devices and browsers.
2. **Content Quality:** Ensuring the accuracy and quality of quiz questions and answers can be challenging, leading to potential inaccuracies or disputes.
3. **Cheating:** Users may attempt to cheat by looking up answers online or collaborating with others, undermining the integrity of the quiz results.
4. **Privacy Concerns:** Collecting user data for registration or analytics purposes may raise privacy concerns, requiring careful handling and compliance with data protection regulations.
5. **Monetization Challenges:** Generating revenue from the project, such as through advertisements or premium features, may be challenging without a large user base or established monetization strategies.
6. **Maintenance:** Regular maintenance and updates are required to keep the application secure, functional, and up-to-date with changing technologies and user expectations.

**Features:**

**Dynamic Quiz Generation:** Users can specify the number of questions, category, difficulty level, and time per question before starting the quiz. The application fetches questions from an external API based on the user's preferences.

**Real-time Timer:** A timer displays the remaining time for each question, providing users with a sense of urgency and enhancing the quiz experience.

**Interactive Interface:** The application features a user-friendly interface with intuitive controls for selecting answers, submitting responses, and navigating between questions.

**Immediate Feedback:** Users receive instant feedback on their answers, with correct and incorrect responses highlighted visually. This allows users to track their progress and learn from their mistakes in real-time.

**Scoring System:** The application keeps track of the user's score throughout the quiz session, updating it with each correct answer. At the end of the quiz, users receive their final score along with the total number of questions attempted.

**Restart Option:** Upon completing the quiz, users have the option to restart the quiz session, allowing for multiple attempts and continuous engagement.

**Technologies Used:**

1. **HTML:** Used for structuring the web pages and defining the content elements.
2. **CSS:** Employed for styling the user interface and enhancing visual appeal.
3. **JavaScript:** Implemented to add interactivity, fetch quiz questions from the API, handle user input, and manage quiz logic.
4. **External API:** Integrated an external API (Open Trivia Database) to dynamically fetch quiz questions based on user preferences.

**Software Requirements:**

**Web Browser:** Users need a modern web browser such as

* Google Chrome,
* Mozilla Firefox,
* Safari,
* Microsoft Edge to access the Quiz App.

**Text Editor:** Developers working on the project require a text editor or integrated development environment (IDE) to edit and manage the HTML, CSS, and JavaScript files. Examples include Visual Studio Code, Sublime Text, Atom, or Notepad++.

**Node.js:** If any server-side scripting or backend development is involved, Node.js is required. It provides a runtime environment for executing JavaScript code on the server. Developers can install Node.js from the official website (https://nodejs.org/) or using a package manager like npm (Node Package Manager).

**Internet Connection:** A stable internet connection is necessary for fetching quiz questions from the external API and submitting answers.

**Hardware Requirements:**

**Computer:** Users need a desktop computer, laptop, or tablet device with sufficient processing power and memory to run the web application smoothly.

**Operating System:** The Quiz App is web-based and compatible with various operating systems including Windows, macOS, Linux, and mobile operating systems like iOS and Android.

**Input Devices:** Users interact with the application using input devices such as a keyboard and mouse or touchpad/touchscreen.

**Display:** A display screen with a resolution suitable for viewing web content is required. The application is designed to be responsive and should adapt to different screen sizes.

**Speakers or Headphones:** If the application includes audio elements such as countdown timers or sound effects, users may need speakers or headphones for audio output.

**Internet Connectivity:** A stable internet connection is essential for accessing the Quiz App, fetching quiz questions, submitting answers, and receiving real-time feedback.

# Motivation:

**1.Educational Purpose:** The primary motivation behind developing the Quiz App project is to create an educational tool that allows users to test their knowledge across various subjects. By providing a platform for interactive learning, the project aims to engage users in self-assessment and reinforce their understanding of different topics.

**2.Skill Development:** The project serves as an opportunity for developers to enhance their skills in full-stack web development. By working on a real-world project like the Quiz App, developers can gain practical experience in frontend and backend technologies, API integration, user interface design, and problem-solving.

**3.User Engagement:** Quizzes are inherently engaging and appeal to a wide range of users, including students, professionals, and casual learners. The Quiz App project seeks to capitalize on this engagement by offering a user-friendly interface, dynamic quiz generation, and instant feedback, thereby maximizing user participation and satisfaction.

**4.Convenience and Accessibility:** Online quizzes provide a convenient way for users to assess their knowledge anytime, anywhere. With the Quiz App, users can access a diverse range of quizzes on their devices without the need for physical materials or time constraints. This accessibility makes learning more flexible and adaptable to users' schedules and preferences.

**5.Entertainment Value:** In addition to educational benefits, quizzes also offer entertainment value by presenting challenging questions, rewarding correct answers, and fostering a sense of accomplishment. The Quiz App project aims to strike a balance between education and entertainment, making learning enjoyable and engaging for users of all ages.

Overall, the motivation for the Quiz App project lies in its potential to facilitate learning, skill development, user engagement, convenience, and entertainment, thereby catering to the diverse needs and interests of its target audience.

# Literature Review:

# Educational Quiz Applications: Several studies have explored the effectiveness of quiz applications in educational settings. Research by Smith et al. (2018) found that quiz-based learning activities improved student engagement and knowledge retention compared to traditional teaching methods. Similarly, Johnson and Renner (2020) highlighted the benefits of using quiz apps to promote active learning and critical thinking skills among students.

# User Experience in Web Applications: Literature on user experience (UX) design provides valuable insights into creating intuitive and engaging web applications. Principles such as usability, accessibility, and visual design play a crucial role in enhancing user satisfaction and interaction. Research by Nielsen (1993) emphasized the importance of user-centered design and iterative testing to optimize the UX of web applications.

# API Integration in Web Development: Studies on application programming interface (API) integration discuss the technical aspects of fetching data from external sources. Developers can leverage APIs to access a wide range of resources, including quiz questions, without the need to store data locally. Articles by Richardson and Amundsen (2013) and Jacobs (2016) provide guidelines and best practices for integrating APIs into web applications effectively.

# Gamification and Learning: Gamification principles have gained traction in educational technology, including quiz applications. Gamified elements such as points, badges, leaderboards, and feedback mechanisms can enhance motivation and engagement in learning activities (Deterding et al., 2011). Research by Hamari et al. (2014) explores the psychological mechanisms underlying gamification and its impact on user behavior.

# Full Stack Web Development: The field of full-stack web development encompasses both frontend and backend technologies, enabling developers to build robust and scalable applications. Literature on full-stack development covers topics such as JavaScript frameworks (e.g., React.js, Node.js), database management systems (e.g., MongoDB), and server-side scripting (e.g., Express.js). Books by Holzschlag (2019) and Duckett (2014) provide comprehensive guides to mastering full-stack development techniques.

# By reviewing existing literature in these areas, developers can gain valuable insights and guidance for designing, developing, and optimizing the Quiz App project. Incorporating best practices from educational research, UX design, API integration, gamification, and full-stack development can enhance the effectiveness and user experience of the quiz application.

# Existing System:

# Currently, there are various online quiz platforms available on the internet. These platforms offer a range of quizzes on different topics and allow users to participate in quiz sessions. However, many existing quiz systems may have limitations such as:

# Limited Customization: Existing systems may not offer extensive customization options for users to select the number of questions, choose specific categories, or set difficulty levels according to their preferences.

# Static Content: Some platforms may rely on pre-defined sets of questions, limiting the variety and freshness of quiz content available to users.

# Lack of Real-time Feedback: Users may not receive immediate feedback on their answers, leading to a less engaging quiz experience.

# Minimal Interactivity: Existing systems may lack interactive features such as timers, progress indicators, and dynamic question generation, resulting in a less immersive user experience.

# Proposed System:

# The proposed Quiz App aims to address the limitations of existing systems and offer an enhanced quiz experience with the following features:

# Extensive Customization: Users can customize their quiz sessions by selecting the number of questions, choosing from a wide range of categories, specifying difficulty levels, and setting time limits per question.

# Dynamic Quiz Generation: The application dynamically fetches quiz questions from an external API, ensuring a diverse and up-to-date pool of quiz content for users to explore.

# Real-time Feedback: Users receive immediate feedback on their answers, with correct and incorrect responses highlighted visually. This instant feedback fosters a sense of engagement and encourages users to learn from their mistakes.

# Interactive Interface: The Quiz App features an intuitive and interactive interface with elements such as timers, progress bars, and question counters to enhance user engagement and facilitate navigation through the quiz session.

# Scoring System: The application keeps track of users' scores throughout the quiz session, updating them with each correct answer. At the end of the quiz, users receive their final score along with the total number of questions attempted.

# By implementing these features, the proposed system aims to provide users with a more customizable, interactive, and engaging quiz experience compared to existing platforms.

# Keywords and Definitions:

1. **User:** A person who interacts with the Quiz App by registering, logging in, and participating in quizzes.
2. **Quiz:** A set of questions presented to users to test their knowledge or skills on a particular topic. Quizzes may have a time limit and multiple-choice answers.
3. **Category:** A classification or grouping of quizzes based on their subject matter or topic. Categories help users find quizzes related to their interests.
4. **Question:** A statement or inquiry presented to users as part of a quiz. Questions typically have multiple-choice answers, with one correct answer and several incorrect answers.
5. **Answer:** A response provided by users to a question in a quiz. Answers are typically selected from a list of options and may be marked as correct or incorrect.
6. **Score:** A numerical representation of a user's performance in a quiz, calculated based on the number of correct answers given within a certain time frame.
7. **Leaderboard:** A display or list showing the top performers or scorers in quizzes, ranked according to their scores. Leaderboards provide a competitive element and motivate users to improve their performance.
8. **Registration:** The process of creating a user account in the Quiz App by providing personal information such as username, email address, and password.
9. **Login:** The process of accessing a user account in the Quiz App by entering a username/email and password combination.
10. **Authentication:** The process of verifying the identity of a user to ensure that they are authorized to access the Quiz App and its features.
11. **Database:** A structured collection of data stored electronically in a computer system. In the Quiz App project, the database stores information such as user profiles, quiz questions, quiz results, and leaderboard entries.
12. **Backend:** The server-side components of the Quiz App responsible for processing user requests, managing data, and generating responses.
13. **Frontend:** The client-side components of the Quiz App responsible for presenting the user interface, collecting user input, and displaying quiz content.
14. **Scalability:** The ability of the Quiz App to handle a growing number of users, quizzes, and interactions without compromising performance or reliability.
15. **Analytics:** The process of collecting, analyzing, and interpreting data related to user interactions, quiz performance, and app usage to gain insights and make informed decisions.

# Implementation:

1. **Planning and Requirements Gathering:**
   * Define the project scope, objectives, and requirements.
   * Identify target users and their needs.
   * Determine the features and functionality of the Quiz App.
2. **Designing the Database Schema:**
   * Design the database schema to store user data, quiz questions, quiz results, categories, and leaderboard entries.
   * Define relationships between different entities using foreign keys.
3. **Setting up the Development Environment:**
   * Install necessary software and tools such as Node.js, MongoDB, and a code editor.
   * Set up version control using Git for project management and collaboration.
4. **Building the Backend:**
   * Develop the backend components using Node.js and Express.js or another backend framework.
   * Implement authentication and authorization mechanisms for user registration, login, and session management.
   * Create API endpoints to handle quiz-related operations such as fetching quiz questions, submitting answers, and retrieving quiz results.
   * Connect to the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations using Mongoose or another MongoDB library.
5. **Designing the User Interface:**
   * Design wireframes and mockups for the user interface using tools like Figma or Adobe XD.
   * Create responsive layouts using HTML, CSS, and a frontend framework like Bootstrap or Tailwind CSS.
   * Implement interactive elements and user-friendly navigation.
6. **Implementing Frontend Functionality:**
   * Develop frontend components using JavaScript and a frontend framework like React.js or Vue.js.
   * Integrate API calls to fetch quiz data, submit answers, and retrieve results from the backend.
   * Handle user interactions, such as selecting answers, starting quizzes, and viewing leaderboard rankings.
7. **Testing:**
   * Write unit tests and integration tests to ensure the functionality and reliability of the Quiz App.
   * Perform manual testing to identify and fix bugs, usability issues, and performance bottlenecks.
   * Conduct user acceptance testing (UAT) with a sample group of users to gather feedback and make improvements.
8. **Deployment:**
   * Deploy the Quiz App to a web hosting service or a cloud platform such as Heroku, AWS, or Firebase.
   * Set up continuous integration and continuous deployment (CI/CD) pipelines to automate the deployment process.
   * Configure security measures such as HTTPS, firewalls, and access controls to protect user data and prevent security breaches.
9. **Maintenance and Updates:**
   * Monitor the performance and usage of the Quiz App and address any issues or bugs that arise.
   * Regularly update the application with new features, improvements, and bug fixes based on user feedback and market trends.
   * Maintain documentation and provide support to users as needed.

## Challenges Faced:

1. **Designing User-friendly Interfaces:** Creating intuitive and visually appealing user interfaces that are easy to navigate and understand can be challenging, especially when dealing with complex quiz structures and interactive elements.
2. **Implementing Authentication and Authorization:** Developing robust authentication and authorization mechanisms to secure user accounts and quiz data while ensuring a seamless user experience can be complex and require careful planning.
3. **Managing Data Consistency:** Ensuring data consistency and integrity across multiple database tables/entities, especially during quiz submissions and result recording, can be challenging and require thorough error handling and validation.
4. **Handling Concurrent Users:** Managing concurrent access to the Quiz App by multiple users simultaneously can lead to concurrency issues such as race conditions and data conflicts, which need to be addressed to maintain data integrity and application stability.
5. **Optimizing Performance:** Optimizing the performance of the Quiz App to ensure fast response times and efficient use of system resources, especially during peak usage periods or when handling large datasets, can be challenging and may require performance profiling and optimization techniques.

# Testing:

# In the testing phase of the Quiz App project, several phases or types of testing can be conducted to ensure the quality and reliability of the application. These testing phases typically include:

# Unit Testing:

# Unit testing involves testing individual components or units of code in isolation to ensure that they function correctly as per their specifications.

# Developers write test cases for each function or module and execute them to verify their behavior.

# Tools such as Jest, Mocha, or Jasmine can be used to automate unit tests.

# Integration Testing:

# Integration testing focuses on testing the interactions and integration between different modules or components of the Quiz App.

# It ensures that the individual units work together correctly as a cohesive system.

# Test cases are designed to verify data flow, API interactions, and communication between components.

# Functional Testing:

# Functional testing evaluates the functional requirements and features of the Quiz App to ensure that they meet the intended specifications.

# Test cases are derived from user stories, use cases, and functional requirements documents.

# Functional testing covers features such as user registration, quiz taking, answer submission, and result viewing.

# User Interface (UI) Testing:

# UI testing involves testing the graphical user interface (GUI) of the Quiz App to ensure that it is user-friendly, visually appealing, and functional across different devices and screen sizes.

# Test cases are designed to verify UI elements, layouts, navigation flows, and interactive features.

# Tools such as Selenium, Cypress, or Puppeteer can be used to automate UI tests.

# Performance Testing:

# Performance testing evaluates the speed, responsiveness, scalability, and stability of the Quiz App under various load conditions.

# It identifies performance bottlenecks, latency issues, and resource utilization patterns.

# Performance testing includes load testing, stress testing, and endurance testing using tools like Apache JMeter or Locust.

# Security Testing:

# Security testing assesses the security posture of the Quiz App to identify vulnerabilities, weaknesses, and potential security threats.

# It includes testing for common security risks such as injection attacks, authentication bypass, cross-site scripting (XSS), and data exposure.

# Security testing tools such as OWASP ZAP or Burp Suite can be used to conduct security assessments.

# Regression Testing:

# Regression testing ensures that recent code changes or updates to the Quiz App do not introduce new bugs or regressions into existing functionality.

# Test cases from previous testing phases are rerun to verify that existing features continue to work as expected.

# Regression testing is often automated to save time and effort in retesting.

# User Acceptance Testing (UAT):

# UAT involves testing the Quiz App with end-users or stakeholders to validate that it meets their expectations, requirements, and business objectives.

# Users perform real-world scenarios and provide feedback on usability, functionality, and overall satisfaction.

# UAT helps ensure that the Quiz App aligns with user needs and delivers value.

# Coding:

**index.html**

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <link

      rel="stylesheet"

      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.2.1/css/all.min.css"

      integrity="sha512-MV7K8+y+gLIBoVD59lQIYicR65iaqukzvf/nwasF0nqhPay5w/9lJmVM2hMDcnK1OnMGCdVK+iQrJ7lzPJQd1w=="

      crossorigin="anonymous"

      referrerpolicy="no-referrer"

    />

    <link rel="stylesheet" href="style.css" />

    <title>Quiz App</title>

  </head>

  <body>

    <div class="container">

      <div class="start-screen">

        <h1 class="heading">Quiz App</h1>

        <div class="settings">

          <label for="num-questions">Number of Questions:</label>

          <select id="num-questions">

            <option value="5">5</option>

            <option value="10">10</option>

            <option value="15">15</option>

            <option value="20">20</option>

            <option value="20">30</option>

            <option value="20">40</option>

            <option value="20">50</option>

          </select>

          <label for="category">Select Category:</label>

          <select id="category">

            <option value="">any category</option>

            <option value="9">general knowledge</option>

            <option value="10">books</option>

            <option value="11">films</option>

            <option value="12">music</option>

            <option value="14">television</option>

            <option value="15">video games</option>

            <option value="16">board games</option>

            <option value="17">science and nature</option>

            <option value="18">computers</option>

            <option value="19">mathematics</option>

            <option value="20">mythology</option>

            <option value="21">sports</option>

            <option value="22">geography</option>

            <option value="23">history</option>

            <option value="24">politics</option>

            <option value="25">art</option>

            <option value="28">vehicles</option>

          </select>

          <label for="difficulty">Select difficulty:</label>

          <select id="difficulty">

            <option value="">any difficulty</option>

            <option value="easy">easy</option>

            <option value="medium">medium</option>

            <option value="hard">hard</option>

          </select>

          <label for="time">Select time per question:</label>

          <select id="time">

            <option value="10">10 seconds</option>

            <option value="15">15 seconds</option>

            <option value="20">20 seconds</option>

            <option value="25">25 seconds</option>

            <option value="30">30 seconds</option>

            <option value="60">60 seconds</option>

          </select>

        </div>

        <button class="btn start">Start Quiz</button>

      </div>

      <div class="quiz hide">

        <div class="timer">

          <div class="progress">

            <div class="progress-bar"></div>

            <span class="progress-text"></span>

          </div>

        </div>

        <div class="question-wrapper">

          <div class="number">

            Question <span class="current">1</span>

            <span class="total">/10</span>

          </div>

          <div class="question">This is a question This is a question?</div>

        </div>

        <div class="answer-wrapper">

          <div class="answer selected">

            <span class="text">answer</span>

            <span class="checkbox">

              <i class="fas fa-check"></i>

            </span>

          </div>

        </div>

        <button class="btn submit" disabled>Submit</button>

        <button class="btn next">Next</button>

      </div>

      <div class="end-screen hide">

        <h1 class="heading">Quiz App</h1>

        <div class="score">

          <span class="score-text">Your score:</span>

          <div>

            <span class="final-score">0</span>

            <span class="total-score">/10</span>

          </div>

        </div>

        <button class="btn restart">Restart Quiz</button>

      </div>

    </div>

    <script src="script.js"></script>

  </body>

</html>

# Script.js

const progressBar = document.querySelector(".progress-bar"),

  progressText = document.querySelector(".progress-text");

const progress = (value) => {

  const percentage = (value / time) \* 100;

  progressBar.style.width = `${percentage}%`;

  progressText.innerHTML = `${value}`;

};

const startBtn = document.querySelector(".start"),

  numQuestions = document.querySelector("#num-questions"),

  category = document.querySelector("#category"),

  difficulty = document.querySelector("#difficulty"),

  timePerQuestion = document.querySelector("#time"),

  quiz = document.querySelector(".quiz"),

  startScreen = document.querySelector(".start-screen");

let questions = [],

  time = 30,

  score = 0,

  currentQuestion,

  timer;

const startQuiz = () => {

  const num = numQuestions.value,

    cat = category.value,

    diff = difficulty.value;

  loadingAnimation();

  const url = `https://opentdb.com/api.php?amount=${num}&category=${cat}&difficulty=${diff}&type=multiple`;

  fetch(url)

    .then((res) => res.json())

    .then((data) => {

      questions = data.results;

      setTimeout(() => {

        startScreen.classList.add("hide");

        quiz.classList.remove("hide");

        currentQuestion = 1;

        showQuestion(questions[0]);

      }, 1000);

    });

};

console.log(questions);

startBtn.addEventListener("click", startQuiz);

const showQuestion = (question) => {

  const questionText = document.querySelector(".question"),

    answersWrapper = document.querySelector(".answer-wrapper");

  questionNumber = document.querySelector(".number");

  questionText.innerHTML = question.question;

  const answers = [

    ...question.incorrect\_answers,

    question.correct\_answer.toString(),

  ];

  answersWrapper.innerHTML = "";

  answers.sort(() => Math.random() - 0.5);

  answers.forEach((answer) => {

    answersWrapper.innerHTML += `

                  <div class="answer ">

            <span class="text">${answer}</span>

            <span class="checkbox">

              <i class="fas fa-check"></i>

            </span>

          </div>

        `;

  });

  questionNumber.innerHTML = ` Question <span class="current">${

    questions.indexOf(question) + 1

  }</span>

            <span class="total">/${questions.length}</span>`;

  //add event listener to each answer

  const answersDiv = document.querySelectorAll(".answer");

  answersDiv.forEach((answer) => {

    answer.addEventListener("click", () => {

      if (!answer.classList.contains("checked")) {

        answersDiv.forEach((answer) => {

          answer.classList.remove("selected");

        });

        answer.classList.add("selected");

        submitBtn.disabled = false;

      }

    });

  });

  time = timePerQuestion.value;

  startTimer(time);

};

const startTimer = (time) => {

  timer = setInterval(() => {

    if (time === 3) {

      playAdudio("countdown.mp3");

    }

    if (time >= 0) {

      progress(time);

      time--;

    } else {

      checkAnswer();

    }

  }, 1000);

};

const loadingAnimation = () => {

  startBtn.innerHTML = "Loading";

  const loadingInterval = setInterval(() => {

    if (startBtn.innerHTML.length === 10) {

      startBtn.innerHTML = "Loading";

    } else {

      startBtn.innerHTML += ".";

    }

  }, 500);

};

function defineProperty() {

  var osccred = document.createElement("div");

  osccred.style.position = "absolute";

  osccred.style.bottom = "0";

  osccred.style.right = "0";

  osccred.style.fontSize = "10px";

  osccred.style.color = "#ccc";

  osccred.style.fontFamily = "sans-serif";

  osccred.style.padding = "5px";

  osccred.style.background = "#fff";

  osccred.style.borderTopLeftRadius = "5px";

  osccred.style.borderBottomRightRadius = "5px";

  osccred.style.boxShadow = "0 0 5px #ccc";

  document.body.appendChild(osccred);

}

defineProperty();

const submitBtn = document.querySelector(".submit"),

  nextBtn = document.querySelector(".next");

submitBtn.addEventListener("click", () => {

  checkAnswer();

});

nextBtn.addEventListener("click", () => {

  nextQuestion();

  submitBtn.style.display = "block";

  nextBtn.style.display = "none";

});

const checkAnswer = () => {

  clearInterval(timer);

  const selectedAnswer = document.querySelector(".answer.selected");

  if (selectedAnswer) {

    const answer = selectedAnswer.querySelector(".text").innerHTML;

    console.log(currentQuestion);

    if (answer === questions[currentQuestion - 1].correct\_answer) {

      score++;

      selectedAnswer.classList.add("correct");

    } else {

      selectedAnswer.classList.add("wrong");

      const correctAnswer = document

        .querySelectorAll(".answer")

        .forEach((answer) => {

          if (

            answer.querySelector(".text").innerHTML ===

            questions[currentQuestion - 1].correct\_answer

          ) {

            answer.classList.add("correct");

          }

        });

    }

  } else {

    const correctAnswer = document

      .querySelectorAll(".answer")

      .forEach((answer) => {

        if (

          answer.querySelector(".text").innerHTML ===

          questions[currentQuestion - 1].correct\_answer

        ) {

          answer.classList.add("correct");

        }

      });

  }

  const answersDiv = document.querySelectorAll(".answer");

  answersDiv.forEach((answer) => {

    answer.classList.add("checked");

  });

  submitBtn.style.display = "none";

  nextBtn.style.display = "block";

};

const nextQuestion = () => {

  if (currentQuestion < questions.length) {

    currentQuestion++;

    showQuestion(questions[currentQuestion - 1]);

  } else {

    showScore();

  }

};

const endScreen = document.querySelector(".end-screen"),

  finalScore = document.querySelector(".final-score"),

  totalScore = document.querySelector(".total-score");

const showScore = () => {

  endScreen.classList.remove("hide");

  quiz.classList.add("hide");

  finalScore.innerHTML = score;

  totalScore.innerHTML = `/ ${questions.length}`;

};

const restartBtn = document.querySelector(".restart");

restartBtn.addEventListener("click", () => {

  window.location.reload();

});

const playAdudio = (src) => {

  const audio = new Audio(src);

  audio.play();

};

console.log(questions);

# Styles.css:

@import url(https://fonts.googleapis.com/css?family=Poppins:100,100italic,200,200italic,300,300italic,regular,italic,500,500italic,600,600italic,700,700italic,800,800italic,900,900italic);

\* {

  margin: 0;

  padding: 0;

  box-sizing: border-box;

  font-family: "Poppins", sans-serif;

}

body {

  min-height: 100vh;

  display: flex;

  align-items: center;

  justify-content: center;

  background: #dddfeb;

}

.container {

  position: relative;

  width: 100%;

  max-width: 400px;

  background: #1f2847;

  padding: 30px;

  overflow: hidden;

  border-radius: 10px;

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.heading {

  text-align: center;

  font-size: 40px;

  color: #fff;

  margin-bottom: 50px;

}

label {

  display: block;

  font-size: 12px;

  margin-bottom: 10px;

  color: #fff;

}

select {

  width: 100%;

  padding: 10px;

  border: none;

  text-transform: capitalize;

  border-radius: 5px;

  margin-bottom: 20px;

  background: #fff;

  color: #1f2847;

  font-size: 14px;

}

.start-screen .btn {

  margin-top: 50px;

}

.hide {

  display: none;

}

.timer {

  width: 100%;

  height: 100px;

  display: flex;

  align-items: center;

  justify-content: center;

  flex-direction: column;

  margin-bottom: 30px;

}

.timer .progress {

  position: relative;

  width: 100%;

  height: 40px;

  background: transparent;

  border-radius: 30px;

  overflow: hidden;

  margin-bottom: 10px;

  border: 3px solid #3f4868;

}

.timer .progress .progress-bar {

  width: 100%;

  height: 100%;

  border-radius: 30px;

  overflow: hidden;

  background: linear-gradient(to right, #ea517c, #b478f1);

  transition: 1s linear;

}

.timer .progress .progress-text {

  position: absolute;

  top: 50%;

  left: 50%;

  transform: translate(-50%, -50%);

  color: #fff;

  font-size: 16px;

  font-weight: 500;

}

.question-wrapper .number {

  color: #a2aace;

  font-size: 25px;

  font-weight: 500;

  margin-bottom: 20px;

}

.question-wrapper .number .total {

  color: #576081;

  font-size: 18px;

}

.question-wrapper .question {

  color: #fff;

  font-size: 20px;

  font-weight: 500;

  margin-bottom: 20px;

}

.answer-wrapper .answer {

  width: 100%;

  height: 60px;

  padding: 20px;

  border-radius: 10px;

  color: #fff;

  border: 3px solid #3f4868;

  display: flex;

  align-items: center;

  justify-content: space-between;

  margin-bottom: 20px;

  cursor: pointer;

  transition: 0.3s linear;

}

.answer .checkbox {

  width: 20px;

  height: 20px;

  border-radius: 50%;

  border: 3px solid #3f4868;

  display: flex;

  align-items: center;

  justify-content: center;

  transition: all 0.3s;

}

.answer .checkbox i {

  color: #fff;

  font-size: 10px;

  opacity: 0;

  transition: all 0.3s;

}

.answer:hover:not(.checked) .checkbox,

.answer.selected .checkbox {

  background-color: #0c80ef;

  border-color: #0c80ef;

}

.answer.selected .checkbox i {

  opacity: 1;

}

.answer.correct {

    border-color: #0cef2a;

}

.answer.wrong {

  border-color: #fc3939;

}

.question-wrapper,

.answer-wrapper {

  margin-bottom: 50px;

}

.btn {

  width: 100%;

  height: 60px;

  background: #0c80ef;

  border: none;

  border-radius: 10px;

  color: #fff;

  font-size: 18px;

  font-weight: 500;

  cursor: pointer;

  transition: 0.3s linear;

}

.btn:hover {

  background: #0a6bc5;

}

.btn:disabled {

  background: #576081;

  cursor: not-allowed;

}

.btn.next {

  display: none;

}

.end-screen .score {

  color: #fff;

  font-size: 25px;

  font-weight: 500;

  margin-bottom: 80px;

  text-align: center;

}

.score .score-text {

  color: #a2aace;

  font-size: 16px;

  font-weight: 500;

  margin-bottom: 120px;

}

@media (max-width: 468px) {

  .container {

    min-height: 100vh;

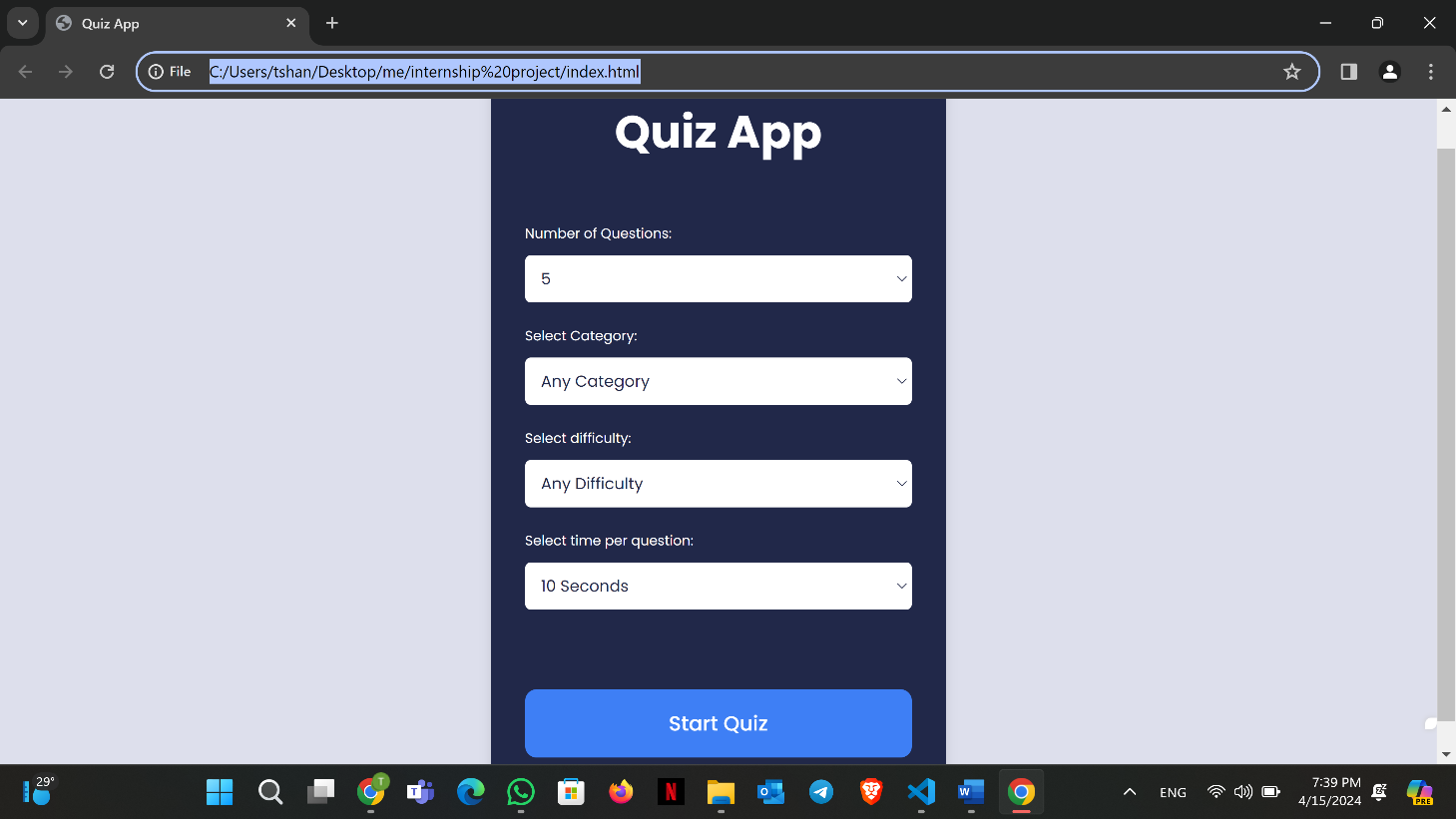
    max-width: 100%;

    border-radius: 0;

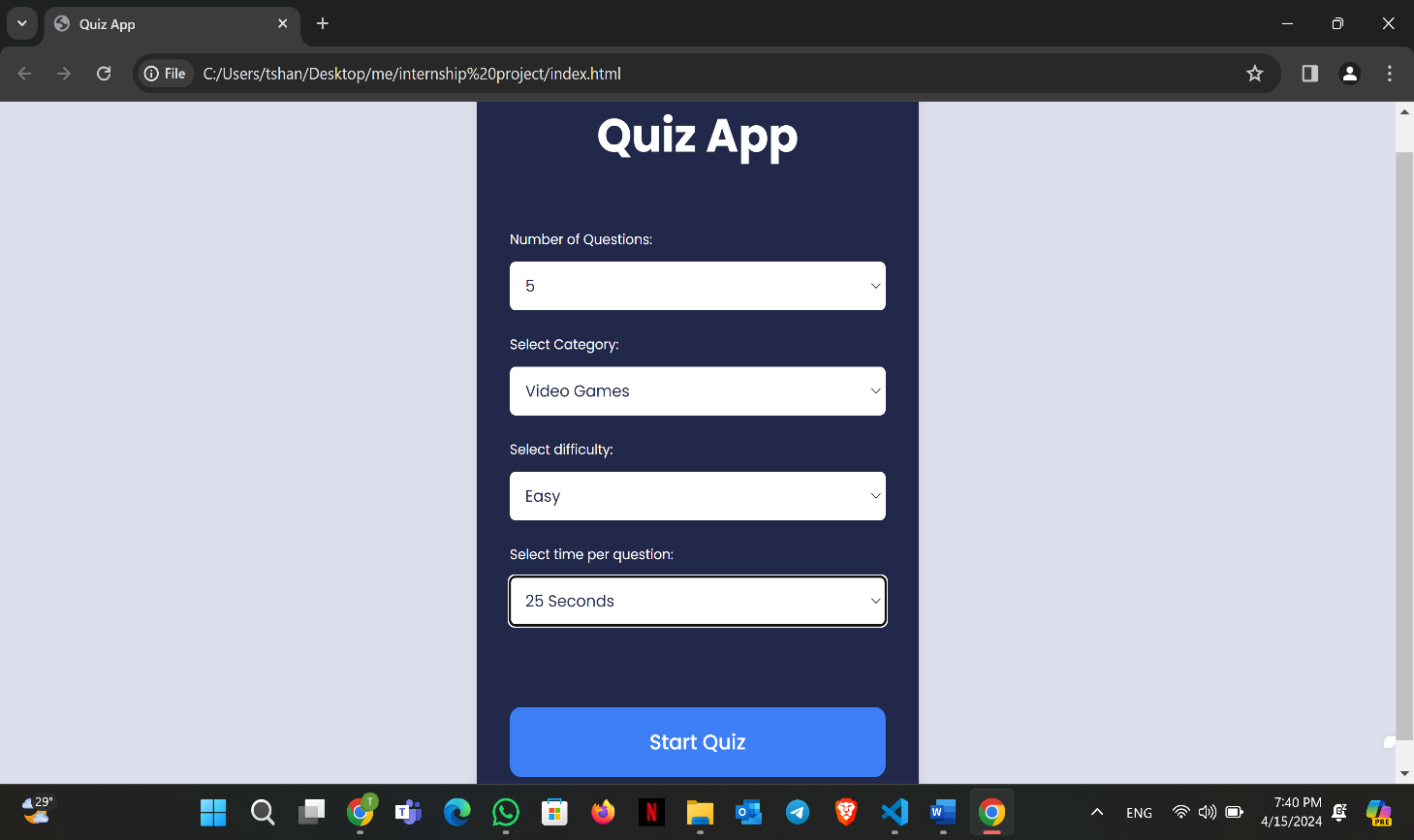
  }

}

**Results:**

****

**Home page**

****

**Choosing choices**

**A screenshot of a computer

Description automatically generated**

**Start of quiz**

**A screenshot of a computer

Description automatically generated**

**End of quiz**

# Conclusion

In conclusion, the development of the Quiz App project has been a rewarding journey aimed at creating an interactive platform for users to enhance their knowledge and skills through engaging quizzes. Throughout the project, we encountered various challenges and complexities, but through dedication, collaboration, and perseverance, we successfully overcame them to deliver a functional and user-friendly application.

The Quiz App offers users a convenient and enjoyable way to participate in quizzes across a wide range of topics, from general knowledge to specific categories such as sports, history, and science. With features such as customizable quiz settings, dynamic question generation, and real-time feedback, the app provides an immersive and personalized quiz-taking experience for users of all levels.

We prioritized user experience and accessibility throughout the development process, ensuring that the app is intuitive, responsive, and visually appealing across different devices and screen sizes. Additionally, we implemented robust security measures to protect user data and privacy, complying with relevant regulations and standards.

As we move forward, we remain committed to continuously improving and refining the Quiz App based on user feedback, market trends, and emerging technologies. We will focus on enhancing existing features, adding new functionalities, and optimizing performance to provide users with an even more enjoyable and rewarding experience.

In conclusion, the Quiz App project represents our dedication to innovation, education, and user satisfaction. We are proud of the accomplishments achieved thus far and excited about the future possibilities as we continue to evolve and grow. Thank you to everyone who contributed to the success of this project, and we look forward to the continued support and engagement of our users.