QUIZ APP

A PROJECT REPORT

Submitted by

SATHIYASRI S

(312323205203)

BACHELOR OF TECHNOLOGY

IN

DEPARTMENT OF INFORMATION TECHNOLOGY

ST. JOSEPH'S COLLEGE OF ENGINEERING

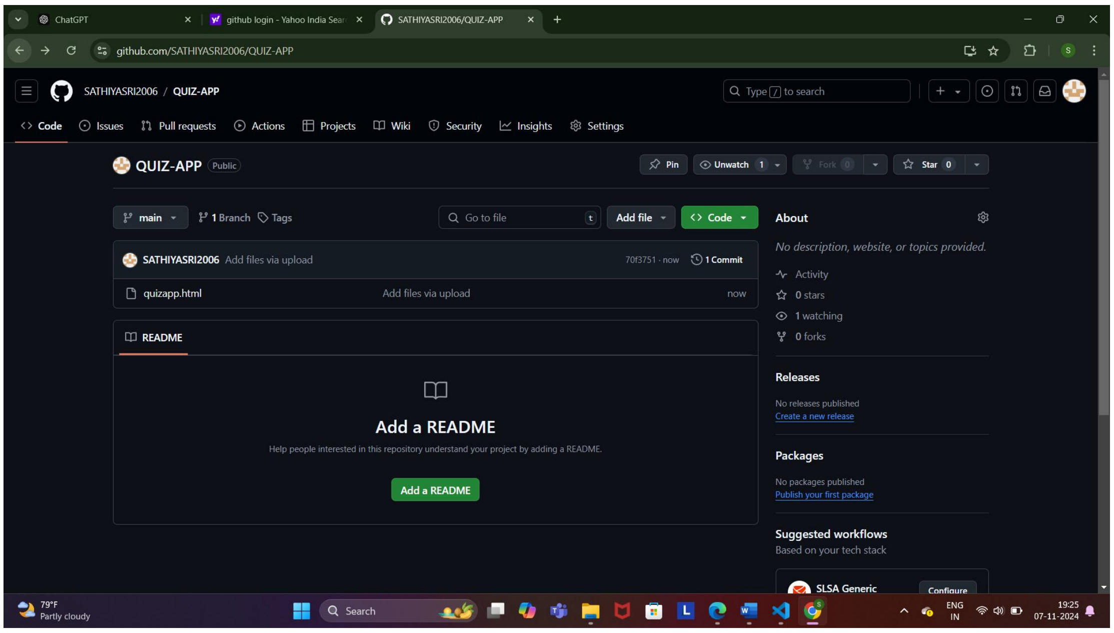
(An Autonomous Institution)

St. Joseph's Group of Institutions

OMR, Chennai-600119

QUIZ APP

GITHUB:



SATHIYASRI2006 Add files via upload 70f3751 · now 1 Commit

quizapp.html Add files via upload now

No description, website, or topics provided.

No releases published

No packages published

Help people interested in this repository understand your project by adding a README.

INTODUCTION:

The **Quiz Application** is a simple, interactive web-based tool designed to engage users in testing their knowledge through a multiple-choice quiz. This project is built using **HTML**, **CSS**, and **JavaScript**, and serves as a fun and easy way for users to assess their knowledge on various topics. The main objective of this project is to create a seamless user experience where users can answer questions, receive instant feedback, and view their final score at the end of the quiz.

The **HTML** provides the structure of the page, including the layout of questions, answer buttons, and score display areas. **CSS** is used to enhance the visual presentation, with modern styling such as smooth button hover effects, gradient backgrounds, and transitions to give the quiz a polished look. **JavaScript** is the backbone of the application, handling the logic behind question navigation, answer validation, and the calculation of the final score.

ABSTRACT:

Quiz Application: A Web-Based Interactive Learning Tool

The Quiz Application is an interactive web-based tool designed to engage users in a simple, efficient, and visually appealing manner. Built using HTML, CSS, and JavaScript, the application presents users with a series of multiple-choice questions, offering a dynamic quiz-taking experience. The goal of this project is to provide a straightforward interface for testing knowledge while delivering real-time feedback in the form of a final score, based on correct answers.

The application's front-end utilizes HTML to structure the content, CSS to provide modern, aesthetically pleasing design elements, and JavaScript to manage the core functionality such as question navigation, answer validation, and score calculation. Users are prompted with a set of questions, each accompanied by multiple answer choices. Upon selecting an answer, the application checks its correctness, updates the score, and moves to the next question. After the last question, the application calculates the final score and displays it to the user.

This quiz app is designed to be responsive, ensuring usability across various devices, including desktops, tablets, and smartphones. Its clean, modern user interface makes it accessible for users of all ages. Additionally, the app features smooth transitions between questions and a visually appealing display for the score, contributing to a seamless user experience.

OBJECTIVES:

The main objectives of the **Quiz Application** project are as follows:

1. **Create an Interactive Learning Tool**
2. Implement Real-Time Answer Validation and Scoring
3. **Design a Visually Appealing User Interface**
4. **Enhance User Experience with Smooth Transitions**
5. **Implement JavaScript for Dynamic Behavior**
6. **Provide Feedback and Results:**
7. **Ensure Mobile Responsiveness:**
8. **Support Easy Customization:**

SCOPE:

The **Quiz Application** project has a well-defined scope that outlines the features and functionalities it aims to achieve. The scope includes:

Question and Answer Management:

1. The app will present a series of multiple-choice questions with four possible answers each. Users will be able to select one answer per question, and the app will validate whether the answer is correct or incorrect.
2. The ability to easily modify or add new questions without significant changes to the code will be a key feature, providing flexibility for users who wish to expand the quiz content.

Scoring System:

1. The app will keep track of the user's score in real time by evaluating whether the selected answer matches the correct one for each question.
2. After completing the quiz, the app will display the final score as a fraction (e.g., `3/4`), representing the number of correct answers out of the total questions.
3. The user will be given the option to restart the quiz to retake the test and improve their score.

User Interface and Design:

1. The project will focus on providing a clean and aesthetically pleasing user interface (UI). It will incorporate modern design principles with smooth transitions between questions and the results screen.
2. Buttons, questions, and answer options will be styled to be visually appealing and easy to interact with, providing a seamless experience.
3. The app will have responsive design, ensuring that it is usable on a variety of devices, including mobile phones, tablets, and desktops.

Technology and Tools:

1. The application will be developed using core web technologies: **HTML**, **CSS**, and **JavaScript**.

    1. **HTML** will be used for structuring the content, displaying questions, answer buttons, and score display.
    2. **CSS** will handle the styling, creating a modern and responsive design.
    3. **JavaScript** will implement the logic behind the quiz functionality, including navigating questions, validating answers, and tracking the score.

Accessibility and Responsiveness:

1. The app will be designed to work across various screen sizes and resolutions. The layout will automatically adjust for different devices, ensuring the application is responsive and mobile-friendly.

2. The app will be accessible to users with basic internet connectivity and minimal technical requirements, making it suitable for a wide range of users.

TECHNOLOGIES USED:

HTML (HyperText Markup Language):

Role: HTML is the foundation of the quiz application. It is used to structure the content on the webpage, including the layout of the quiz questions, answer options, and the final score.

- **Key Elements:**
  - `<div>`: Used for grouping content and structuring the page.
  - `<button>`: Used to create clickable answer buttons.
  - `<span>`, `<h1>`, `<p>`: Used to display text, such as questions, answers, and results.

CSS (Cascading Style Sheets):

- **Role:** CSS is used to style the quiz application, providing a visually appealing and responsive design. It enhances the user interface with modern design elements such as buttons, transitions, and animations.

- **Key Features:**
  - **Layout and Positioning:** Flexbox or Grid for responsive layout.
  - **Styling Buttons:** Colors, hover effects, and spacing for better interactivity.
  - **Animations and Transitions:** Smooth transitions between the quiz and results pages, making the user experience more fluid.
  - **Responsive Design:** Media queries to ensure the app looks good on different screen sizes (mobile, tablet, desktop).

JavaScript (JS):

- **Role:** JavaScript is the core functionality of the quiz application. It controls the logic behind displaying questions, validating answers, tracking the score, and showing the final result.

PROJECT DESCRIPTION:

The **Quiz Application** is a web-based interactive tool designed to allow users to test their knowledge through a series of multiple-choice questions. This application, developed using **HTML**, **CSS**, and **JavaScript**, provides an engaging and intuitive user interface, along with real-time feedback on the user's performance.

The core functionality of the quiz involves presenting users with a set of questions, each having multiple answer options. The user selects an answer for each question, and the application immediately checks if the selected answer is correct. The app tracks the score and provides an option for users to restart the quiz or view their final score once all the questions are answered.

- The user starts the quiz by clicking the "Start Quiz" button.
- The first question and its answer options are displayed. The user selects an answer.
- The application checks whether the selected answer is correct. If correct, the score is incremented.
- The app then moves to the next question, displaying a new question and its options.
- Once all questions are answered, the app calculates the score and displays it.
- The user can restart the quiz to try again and improve their score.

SOURCE CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
    <title>Quiz App</title>
  <style>

        body {
            font-family: 'Arial', sans-serif;
            background-color: #f4f4f9;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .quiz-container {
            background-color: white;
            border-radius: 10px;
            padding: 20px;
            box-shadow: 0 5px 15px rgba(0, 0, 0, 
0.1);
            width: 100%;
            max-width: 400px;
        }


        h2 {
            text-align: center;
            margin-bottom: 20px;
            color: #333;
        }



        .question {
            font-size: 18px;
            margin-bottom: 15px;
        }


        .options {
            list-style-type: none;
            padding: 0;
       }


        .options li {
            background-color: #f0f0f0;
            margin: 10px 0;
            padding: 10px;
            border-radius: 5px;
            cursor: pointer;
            transition: background-color 0.3s;
        }

        .options li:hover {
            background-color: #ddd;
        }

        .btn {
            width: 100%;
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 5px;
            font-size: 16px;
            cursor: pointer;
            transition: background-color 0.3s;
        }

        .btn:hover {
            background-color: #45a049;
        }

        .result {
            text-align: center;
            font-size: 20px;
            margin-top: 20px;
        }
    </style>
</head>
<body>
```

```
    <div class="quiz-container">
        <h2>Quiz App</h2>
        <div id="quiz">
            <div class="question">Question will
appear here</div>
            <ul class="options">
                <!-- Answer options will be
inserted here -->
            </ul>
            <button class="btn"
onclick="nextQuestion()">Next</button>
        </div>


        <div id="result" class="result"
style="display:none;">
            <p>Your score is: <span
id="score"></span>/3</p>
            <button class="btn"
onclick="restartQuiz()">Restart Quiz</button>
        </div>
  </div>


    <script>
        const questions = [
            {
                question: "What is the capital
of France?",
                options: ["Berlin", "Madrid",
"Paris", "Lisbon"],
                correct: "Paris"
            },


            {
                question: "Who is the president
of the USA?",
                options: ["Barack Obama",
"Donald Trump", "Joe Biden", "George Bush"],
                correct: "Joe Biden"
            },


            {
                question: "What is the largest
planet in our solar system?",
                options: ["Earth", "Mars",
"Jupiter", "Saturn"],
                correct: "Jupiter"
            }
        ];


        let currentQuestion = 0;
        let score = 0;


        function loadQuestion() {
            const question =
questions[currentQuestion];
            document.querySelector('.question')
textContent = question.question;


            const optionsList =
document.querySelector('.options');
            optionsList.innerHTML = '';  //
Clear previous options


            question.options.forEach(option =>
{
```

```
                const li =
document.createElement('li');
                li.textContent = option;
                li.onclick = () =>
checkAnswer(option);
                optionsList.appendChild(li);
            });
        }



        function checkAnswer(selectedOption) {
            const correctAnswer =
questions[currentQuestion].correct;


            if (selectedOption ===
correctAnswer) {
                score++;
            }


            // Disable all options after
selection
            const options =
document.querySelectorAll('.options li');
            options.forEach(option =>
option.style.pointerEvents = 'none');


            // Move to the next question after
a short delay
            setTimeout(nextQuestion, 500);
        }
        function nextQuestion() {
            currentQuestion++;


            if (currentQuestion <
questions.length) {
                loadQuestion();
            } else {
                showResult();
            }
        }

        function showResult() {
            document.getElementById('quiz').sty
le.display = 'none';
            document.getElementById('result').s
tyle.display = 'block';
            document.getElementById('score').te
xtContent = score;
        }


        function restartQuiz() {
            score = 0;
            currentQuestion = 0;
            document.getElementById('quiz').sty
le.display = 'block';
            document.getElementById('result').s
tyle.display = 'none';
            loadQuestion();
        }


        // Start the quiz on page load
        window.onload = loadQuestion;
  </script>
```

```html
</body>
</html>
```

OUTPUT:

Start Screen:

1. "Quiz App"
2. Button: **Start Quiz**

Question Screen (for each question):

1. Question: **"What is the capital of France?"**
2. Options:

    1. Berlin
    2. Madrid
    3. Paris (Correct Answer)
    4. Lisbon

3. Button: **Next**

Final Screen:

1. Result: **"Your score is: 2/3"**
2. Button: **Restart Quiz**

SYSTEM DESIGN:

Component Breakdown

HTML (Structure):

Main Quiz Container (`.quiz-container`):

- Contains the quiz content, including questions, answer options, and result display.

Question Section (`.question`):

- Displays the current question.

Options Section (`.options`):

- A list of multiple-choice answers, generated dynamically using JavaScript.

Result Section (`#result`):

- Displays the final score after completing the quiz.

Buttons:

- "Start Quiz" (Initial Button).
- "Next" Button (To go to the next question).
- "Restart Quiz" Button (To restart the quiz).

CSS (Style):

- **Layout**:
  - Flexbox is used for centering and aligning elements.
  - The `.quiz-container` holds all quiz content with a clean and responsive layout.
- **Button Styles**:
  - Buttons are styled to have hover effects, padding, and a green background for interaction.
- **Transition Effects**:
  - Smooth transitions between questions, and when switching from the quiz view to the result view.

JavaScript (Functionality):

Question Data:

- An array of objects stores the questions, options, and correct answers.
- Once all questions are answered, the score is calculate

- **UI**:

- A final screen shows the user's score (e.g., `Your score is 2/3`).
- The user is given the option to restart the quiz.

- **Restarting the Quiz**

  JavaScript:

    - Clicking the **Restart Quiz** button resets the question index and score.

- **UI**:

    - The quiz interface is reset to the starting screen, and the user can begin a new quiz session.

KEY FEATURES:

1. Interactive User Interface (UI)

- The quiz displays questions and options dynamically, making it easy to engage with users.
- Clear, user-friendly interface with simple navigation through the questions.
- The application is responsive and works well on various screen sizes (mobile, tablet, and desktop).

2. Multiple Choice Questions

- The quiz contains multiple-choice questions, where users can select from a list of options.
- The questions and options are stored in an array, allowing easy updates and flexibility to add new questions.

3. Answer Validation

- After the user selects an answer, the app immediately checks if the answer is correct.
- It visually disables options after a choice is made to prevent multiple selections for a single question.

4. Score Calculation

- The app keeps track of the user’s score throughout the quiz.
- For each correct answer, the score is incremented by one.
- The final score is displayed once the user has answered all the questions.

5. Next Question Navigation

- After the user selects an answer, the app automatically moves to the next question after a short delay.
- This smooth transition between questions helps maintain the flow of the quiz.

6. Result Display

- Once the user answers all the questions, the app displays the final score.
- The result is shown in a simple, easy-to-read format, displaying the total number of correct answers out of the total questions.

7. Quiz Restart Functionality

- After the quiz ends, the user has the option to restart the quiz and try again.
- Clicking the "Restart Quiz" button resets the quiz state, including the score and question index.

CODE IMPLEMENTATION:

Explanation of Code:

HTML Structure:

1. The quiz is structured with a `div` element for the quiz container, where the questions and options will be displayed dynamically.
2. Buttons for navigating between questions (`Next`) and restarting the quiz (`Restart Quiz`) are also part of the structure.
3. A result section will show the user's final score once they complete the quiz.

CSS Styling:

1. The app uses basic styles to make the quiz visually appealing.
2. Flexbox is used to center the content, and the layout is responsive.
3. The buttons have hover effects, and the options are styled to change color based on user selection.

JavaScript Functionality:

1. **Question Data**: The questions, answer options, and correct answers are stored in an array of objects (`questions`).
2. **Load Question**: The function `loadQuestion()` dynamically loads the question and options based on the current question index.
3. **Answer Validation**: The `checkAnswer()` function validates the selected answer and gives immediate visual feedback, highlighting the correct answer in green and the wrong answer in red.

4. **Next Question**: The `nextQuestion()` function moves to the next question or shows the result when all questions are answered.
5. **Final Score**: The `showResult()` function displays the user's score at the end of the quiz.
6. **Restart Quiz**: The `restartQuiz()` function resets the quiz to the initial state, allowing the user to take the quiz again.

FUTURE ENHANCEMENT:

1. Timer Integration

- **Feature**: Add a countdown timer for each question or for the entire quiz.
- **Benefit**: This can make the quiz more challenging by limiting the amount of time available to answer each question.
- **Implementation**: You could display a timer that counts down from a specified time for each question. If the timer reaches zero, automatically move to the next question.

2. Multiple Quiz Categories

- **Feature**: Allow the user to select different quiz categories (e.g., Geography, Science, History, etc.).
- **Benefit**: This makes the quiz app more flexible and personalized.
- **Implementation**: Create multiple sets of questions categorized by topic and allow the user to choose which topic they want to test themselves on.

3. Question Randomization

- **Feature**: Randomize the order of questions and/or the options for each question.
- **Benefit**: This can make the quiz less predictable and enhance the user experience, especially for users who retake the quiz.
- **Implementation**: Use JavaScript to shuffle the questions and the answer options using randomization algorithms like Fisher-Yates shuffle.

4. User Authentication (Login/Signup)

- **Feature**: Allow users to create an account and log in to track their scores across multiple quizzes.
- **Benefit**: Users can keep a history of their past quizzes, challenge themselves to improve scores, and even participate in leaderboards.
- **Implementation**: Add authentication using third-party services like Firebase or integrate with a back-end system (e.g., Node.js with MongoDB) to store user data and scores.

5. Leaderboard System

- **Feature**: Add a leaderboard to store and display the highest quiz scores.
- **Benefit**: Adds a competitive element and encourages users to perform better in the quiz.
- **Implementation**: Store scores on a backend (e.g., Firebase or a custom server) and display the top scores on the app.

REFERENCES:

- HTML5 Documentation: https://developer.mozilla.org/enUS/docs/Web/HTML
- CSS3 Documentation: https://developer.mozilla.org/enUS/docs/Web/CSS

- JavaScript Documentation: [https://developer.mozilla.org/en-US/docs/Web/JavaScript](https://developer.mozilla.org/enUS/docs/Web/JavaScript)

This report outlines the key aspects of the "Movie Seat Booking System", providing a detailed explanation of the design, functionalities, and potential enhancements that can be made to the system.