**TITLE:** **CODTECH IT SOLUTIONS INTERNSHIP**

Task Documentation: “**Quiz Application**” Using CSS, HTML, JAVASCRIPT.

**INTERN INFORMATION:**

**Name:** Nikhil Thipparthi

**ID:** ICOD6447

**INTRODUCTION**

Welcome to the documentation for our Quiz Application! This document serves as a guide for understanding the structure, functionality, and implementation of our web-based quiz application developed using CSS, HTML, and JavaScript.

The purpose of our Quiz Application is to provide users with an immersive and enjoyable way to test their knowledge on a wide range of topics. Whether it's for educational purposes, entertainment, or assessment, our application offers a seamless and intuitive interface for conducting quizzes.

**Features:**

* User-Friendly Interface: Our application boasts a clean and intuitive user interface, making it easy for users to navigate through the quiz and select their answers.
* Dynamic Question Generation: With the ability to dynamically generate questions from a predefined dataset, our application ensures a diverse and customizable quiz experience every time.
* Real-Time Feedback: Users receive instant feedback upon submitting their answers, allowing them to track their progress and learn from their mistakes.
* Scoring System: The application calculates and displays the user's score in real-time, providing a sense of achievement and motivation to continue participating in the quiz.
* Responsive Design: Designed with responsiveness in mind, our application adapts seamlessly to different devices and screen sizes, ensuring a consistent user experience across desktops, tablets, and mobile devices.

**IMPLEMENTATION**

* To implement a Quiz Application using CSS, HTML, and JavaScript, you can follow these steps

1. **HTML Structure:**

Define the structure of your quiz application using HTML. This includes elements for questions, options, buttons, and any other necessary components.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Quiz Application</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="quiz-container">

<div id="question"></div>

<div id="options"></div>

<button id="submit-btn">Submit</button>

</div>

<script src="script.js"></script>

</body>

</html>

1. **CSS Styling:**

Style your quiz application using CSS to enhance its visual appeal and layout.

.quiz-container {

width: 80%;

margin: 0 auto;

text-align: center;

}

#options {

margin-top: 20px;

}

button {

padding: 10px 20px;

margin-top: 20px;

cursor: pointer;

}

1. **JavaScript Functionality:**

Implement the logic and functionality of your quiz application using JavaScript. This includes loading questions, handling user input, calculating scores, and displaying results.

const quizData = [

{

question: "What is the capital of France?",

options: ["Paris", "Berlin", "London", "Madrid"],

answer: "Paris"

},

// Add more questions here...

];

const questionElement = document.getElementById('question');

const optionsElement = document.getElementById('options');

const submitButton = document.getElementById('submit-btn');

let currentQuestion = 0;

let score = 0;

function loadQuestion() {

const currentQuizData = quizData[currentQuestion];

questionElement.textContent = currentQuizData.question;

optionsElement.innerHTML = "";

currentQuizData.options.forEach(option => {

const button = document.createElement('button');

button.textContent = option;

button.addEventListener('click', selectOption);

optionsElement.appendChild(button);

});

}

function selectOption(e) {

const selectedOption = e.target.textContent;

const currentQuizData = quizData[currentQuestion];

if (selectedOption === currentQuizData.answer) {

score++;

}

currentQuestion++;

if (currentQuestion < quizData.length) {

loadQuestion();

} else {

showResults();

}

}

function showResults() {

questionElement.textContent = `You scored ${score} out of ${quizData.length} questions.`;

optionsElement.innerHTML = "";

submitButton.style.display = 'none';

}

loadQuestion();

**CODE EXPLAINATION**

**HTML Structure:**

<div class="listcontainer">: Acts as the main container for the to-do list application, wrapping everything in a visually appealing background.

<div class="todo-app">: Encloses the to-do list's title, input area, and list itself, providing a centered, stylized container for the app components.

<h2>To-Do List</h2>: The title for the application.

Input Row (<div class='row'>): Contains the text input field and the Enter button. Users can type their task here and add it to the list.

<ul id="List-container">: The unordered list where tasks will be displayed as list items (<li>).

**CSS Styling:**

The CSS styles define the look and feel of the to-do list, applying a gradient background, styling the input fields, buttons, and tasks.

Key styling includes:

* Global styles are applied to set margin, padding, and font settings.
* The application is centered on the page with a maximum width and padding for aesthetics.
* Input fields and buttons are styled for a seamless interface, with hover effects for interactivity.
* The .listcontainer and .todo-app are styled to center the content and apply specific background colors and paddings.
* Tasks (<li> elements) have distinctive styles, with completed tasks being visually different to provide clear feedback on their status.
* Tasks (<li> elements) and other components like the input box and buttons have specific styles for appearance, hover effects, and when a task is marked as completed.

**JavaScript Functionality:**

The JavaScript adds dynamic behavior to the to-do list, covering task addition, completion marking, and deletion.

**Adding Tasks (add function):**

Checks if the input field (inputactivity) is empty. If not, it proceeds; otherwise, it alerts the user to enter a task.

Creates a new list item (<li>) and sets its content to the value entered in the input field.

Appends a close button (<span>) to each task for the removal functionality, with a click event listener that hides the task on click.

Clears the input field after adding the task to the list.

**Marking Tasks as Completed:**

Utilizes event delegation by adding a click event listener to the list container (inputlist). When a task is clicked, the 'checked' class is toggled on the task, changing its appearance to indicate completion.

**Removing Tasks:**

The close button (<span> with '×') added to each task allows users to remove tasks from the list.

Initially set up in the add function and further facilitated through a click event listener that sets the task's display style to "none", effectively hiding it.

**USAGE**

To use the Quiz Application implemented with CSS, HTML, and JavaScript, follow these steps:

* Download the Files: If you haven't already, download the HTML, CSS, and JavaScript files containing the implementation of the Quiz Application.
* Open the HTML File: Navigate to the directory where you've saved the files and open the HTML file in a web browser of your choice.
* Start the Quiz: Once the HTML file is opened in the browser, you'll see the Quiz Application interface. The first question will be displayed along with the options to choose from.
* Answer the Questions: Read the question and select the option you believe is correct by clicking on the corresponding button.
* Submit Answers: After selecting your answer, click the "Submit" button to proceed to the next question. The application will provide instant feedback on whether your answer was correct or incorrect.
* Continue with the Quiz: Repeat steps 4 and 5 for each subsequent question until you've completed the quiz.
* View Results: Once you've answered all the questions, the application will display your final score, showing the number of correct answers out of the total number of questions.
* Try Again: If you wish to retake the quiz, simply refresh the page in your browser to start over from the beginning.
* Customization (Optional): If you want to customize the quiz content, styling, or functionality, you can do so by modifying the HTML, CSS, and JavaScript files accordingly.

**CONCLUSION**

In conclusion, the Quiz Application developed using CSS, HTML, and JavaScript provides an engaging and interactive platform for users to test their knowledge on various topics. With its user-friendly interface, dynamic question generation, real-time feedback, and responsive design, the application offers a seamless quiz experience across different devices and screen sizes.

By following the steps outlined in the documentation, developers can easily implement and customize the Quiz Application to suit their specific requirements. Whether it's for educational purposes, entertainment, or assessment, this application offers a versatile solution for creating captivating quiz experiences.

We hope this documentation has provided you with the necessary guidance to understand, implement, and utilize the Quiz Application effectively. Thank you for considering our application, and we look forward to seeing the creative quizzes you develop with it.

**OUTPUT**







