COLLEGE CODE : 9604

COLLEGE NAME : C.S.I INSTITUTE OF TECHNOLOGY

DEPARTMENT : AI & DS

STUDENT NM ID: 332E259F6622205A5184DD2072774F0D

ROLL NO : 960423243006

DATE : 23/09/2025

SUBMITTED BY,

NAME : ASHICK JOSHUA C

MOBILE NO : 9487944311

PHASE 3-MVP IMPLEMENTATION

Project Setup:

To set up the project, you need to create a project folder and organize it with separate files for the app's main logic, styles, and questions database. The basic application structure is built using HTML for the layout, CSS for styling, and JavaScript for interactive logic.

- **HTML Structure:** The index.html file creates the quiz interface, which includes a container for the quiz elements, a timer, a question display area, a section for options, a score display, and a "Restart Quiz" button.
- **CSS Styling:** A styles.css file is linked to the HTML to handle the visual presentation.
- **JavaScript Logic:** A script.js file is linked to manage the quiz's interactive functions, such as storing question data, handling user answers, and managing the score.

Code and Output:

HTML Code

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<title>Quiz App</title>
<link rel="stylesheet" href="styles.css" />
</head>
<body>
<div class="quiz-container">
<div class="timer">Time Left: <span id="time">30</span>s</div>
<div class="question">Question will appear here</div>
<div class="options"></div>
<div class="result">Your score: <span id="score">0</span></div>
<button class="restart-btn">Restart Quiz</button>
</div>
<script src="script.js"></script>
</body>
```

</html>

Output :

The initial output of the app displays a simple interface with placeholders for the quiz content:

- Time Left: 30s
- Question will appear here
- [options will load here]
- Your score: 0
- [Restart Quiz Button]

Core Features Implementation:

The app implements several core features to provide an engaging user experience.

- **Quiz Interface:** The app displays one question at a time with multiple answer options as clickable buttons. It highlights selected answers and disables options after a choice is made.
- **Timer:** A countdown timer can be implemented for each question or for the entire quiz. The app can be configured to automatically move to the next question or end the quiz when the time runs out.
- **Score Tracking:** A score variable is maintained and incremented for each correct answer. The score can be updated dynamically or displayed at the end of the quiz.
- **Navigation & Flow:** The app includes a "Next" button to proceed to the next question and ends the quiz once all questions have been answered or the timer expires.
- **Results & Restart:** Upon completion, the app displays the final score and a summary , along with a "Restart/Replay" button to retake the quiz.

Python Code Example:

A simple Python example demonstrates a functional quiz app.

Python

```
# short_quiz.py
questions = {
"Capital of India? ": "Delhi",
"Father of Electricity? ": "Thomas Edison"
}
score = 0
for q, a in questions.items():
score += input(q).strip().lower() == a.lower()
print(f"Score: {score}/{len(questions)}")
```

Python Output :

When the Python code is run, the output would be:

- Capital of India? Delhi
- Father of Electricity? Edison
- Score: 2

Data Storage (Local State / Database):

The project can use several approaches for data storage, depending on the scale and complexity of the quiz.

- **Local State / Local Storage:** This is suitable for small datasets like user progress, scores, and temporary quiz states.

localStorage is a simple key-value storage option for web apps , though it is not ideal for large or complex data.

- **Local Database:** For larger question sets (thousands of items), a lightweight local database like **SQLite** is recommended.

IndexedDB is a powerful local database option for web apps that can store structured data beyond localStorage limits.

- **Hybrid Approach:** A combined approach is often best. The main question set can be stored in a local database like SQLite, while

localStorage can be used for temporary data like the current quiz state and user scores. This data can then be synced with a remote cloud service (like Firebase or a custom backend) to provide fresh questions.

Testing Core Features:

Thorough testing ensures a smooth user experience.

- **Functional Testing:** Verifies that quizzes load correctly, questions and answers appear as expected, and user input is handled properly. It also checks the accuracy of scoring and core flows like quiz start and navigation.
- **User Interface and Usability Testing:** Ensures UI elements are responsive, the layout is consistent across different devices, and real user gestures (like tap and scroll) are simulated.
- **Compatibility Testing:** Checks the app's performance across different operating systems, browser versions, and network conditions.

- **Performance Testing:** Ensures the app handles quiz data efficiently and loads questions quickly.

- **Error and Edge Case Testing:** Checks the app's behavior with invalid inputs or incomplete quizzes and validates data persistence and recovery from crashes.

Version Control (GitHub):

GitHub is a key tool for project development, allowing you to track changes, collaborate with others, and back up your code.

- **Features:** GitHub uses **repositories** to store code ,

commits to track changes ,

branches for new features or bug fixes , and

pull requests for code review and merging.

- **Best Practices:** It's recommended to make regular commits with clear messages and use a branching strategy to manage different code versions.
- **Tools:** You can use **Git** for local repository management, **GitHub Desktop** for a graphical interface, or the **GitHub CLI** for command-line interactions.

Sources