**Quiz Website**

**Our plan:**

Here we are going to make a website which have a page where a user can add come questions and add options to that question and after that It will able to share a link and user with the link can give the quiz and each question should have a particular time set by the questioner.

**Requirements and Executions:**

**1. Project Setup**

**Front-end:** HTML, CSS, JavaScript (React.js or Vanilla JS)

**Back-end:** Node.js with Express.js (or any other backend framework)

**Database:** MongoDB or MySQL for storing questions, user data, and quiz results

**Authentication:** JWT (JSON Web Token) for user authentication

**2. Core Features**

**User Authentication**

**Sign-Up/Login:** Allow users to sign up and log in to create and take quizzes.

**Profile Management:** Users can view and manage their profile, including quizzes created and taken.

**Question Creation Page**

**Form for Adding Questions:**

- Input field for the question text.

- Options (minimum of 2, maximum of 4 or more) with radio buttons for the correct answer.

- Input for setting a time limit for each question.

- Submit button to save the question to the database.

**Option to Add More Questions:** Users should be able to add multiple questions to a quiz.

**Quiz Link Generation**

- After creating a quiz, the user should be able to generate a unique link that can be shared with others.

- The link will direct to a quiz-taking page.

**Quiz-Taking Page**

**- Display Questions One by One:** Each question should be displayed individually with the set time limit.

**- Timer Functionality:** The timer should count down for each question. Once time is up, the question should be automatically submitted or skipped.

**- Submit Quiz:** At the end of the quiz, answers should be submitted and recorded.

**Result Calculation and Display**

**- Score Calculation:** Based on correct answers, calculate the score.

**- Result Page:** Display the score and correct answers after the quiz is completed.

**Database Schema Example**

**- Users Table:**

- `userID`, `username`, `email`, `password`

**- Quizzes Table:**

- `quizID`, `creatorID`, `quizTitle`

**- Questions Table:**

- `questionID`, `quizID`, `questionText`, `options`, `correctOption`, `timeLimit`

**- Results Table:**

- `resultID`, `quizID`, `userID`, `score`, `dateTaken`

**3. Technology Stack**

**Front-End:**

- HTML/CSS for structure and styling.

- JavaScript for handling logic and DOM manipulation.

- React.js (optional) for a more dynamic and scalable front-end.

**Back-End:**

- Node.js with Express.js to handle API requests.

- MongoDB (with Mongoose) for storing data.

**Deployment:**

- Host on platforms like Heroku, Netlify, or Vercel.

**4. Additional Features**

**- Leaderboards:** Display top scores for each quiz.

- Quiz Categories Organize quizzes by categories or tags.

- Mobile Responsiveness Ensure the website is accessible on all devices.

**- Email Notifications:** Notify users when their quiz is taken and results are available.

**5. Workflow**

1. User registers → User logs in → User creates a quiz → User shares the quiz link.

2. Other users access the link → Take the quiz → Submit → View results.

**6. Implementation Steps**

1. Setup project environment (Initialize npm, set up database, create project structure).

2. Create basic routes (for user registration, login, quiz creation, and quiz taking).

3. Design and implement the front-end (HTML/CSS layout and JS/React.js functionality).

4. Develop back-end logic (Express.js routes, JWT authentication, database interactions).

5. Integrate timer functionality.

6. Testing (Ensure all features work correctly).

7. Deploy.

Colour palletts:- https://colorhunt.co/palette/fff2d7ffe0b5f8c794d8ae7e