QUIZ CREATOR PROJECT

**1. Introduction**

The purpose of this project is to develop a **web-based Quiz Creator and Player application** using **HTML, CSS, JavaScript, and Firebase Firestore**.

The application allows an **administrator** to create quiz questions with multiple options, store them in Firebase, and later retrieve them for users to attempt. Users can play the quiz, answer questions, and view their scores at the end.

This project demonstrates how **frontend technologies** can be combined with a **cloud database (Firebase Firestore)** to create an interactive and persistent application.

**2. Objectives**

* To design a **user-friendly interface** for creating and playing quizzes.
* To implement **dynamic quiz options** (not limited to 4 choices).
* To integrate **Firebase Firestore** for storing and retrieving quiz data.
* To enable real-time quiz management with minimal backend coding.
* To demonstrate the use of **modular JavaScript** and Firebase SDK.

**3. Technologies Used**

1. **HTML5** – for structuring the web application.
2. **CSS3** – for styling and providing a clean user interface.
3. **JavaScript (ES6)** – for implementing the quiz logic and handling user interactions.
4. **Firebase Firestore** – for storing quiz questions and answers in the cloud.
5. **Firebase Hosting (optional)** – for deploying the application online.

**4. System Design**

**4.1 Architecture**

* **Frontend**: Built using HTML, CSS, and JavaScript.
* **Database**: Firebase Firestore (cloud NoSQL database).
* **Workflow**:
  + Admin creates a question → Saves to Firestore.
  + Player fetches questions → Plays quiz → Gets score.

**4.2 Modules**

1. **Admin Module**
   * Add a new quiz question.
   * Add unlimited options dynamically.
   * Select the correct answer.
   * Save questions to Firestore.
   * View recently added questions.
2. **Player Module**
   * Enter username.
   * Fetch quiz questions from Firestore.
   * Attempt quiz with multiple options.
   * View score after completing the quiz.

**5. Implementation**

**5.1 Admin Workflow**

1. Enter the question text.
2. Add multiple options using **“+ Add Option”** button.
3. Select the correct option from the dropdown.
4. Click **Save Question** to store in Firebase.
5. The question appears in the **Existing Questions** list.

**5.2 Player Workflow**

1. Enter username and click **Start Quiz**.
2. Questions are fetched from Firestore.
3. Select an answer and click **Next**.
4. At the end, click **Finish** to view the score.

**6. Advantages**

* **Dynamic Options**: Not restricted to 4 choices.
* **Cloud Storage**: Questions are permanently saved in Firebase.
* **Simple UI**: Easy to use for both admin and players.
* **No Backend Required**: Firebase acts as backend + database.

**7. Limitations**

* Currently does not store **user scores/leaderboard** in Firebase (can be added).
* Requires **internet connection** to access Firebase.
* Basic authentication (admin login) is not implemented.

**8. Future Enhancements**

* Add **leaderboard** to store scores of all users.
* Implement **user authentication** (Admin login).
* Add **timer for each question**.
* Allow **categories/levels of quizzes**.
* Provide **export/import quiz feature**.

**9. Conclusion**

This project demonstrates how a simple **quiz application** can be built using only **frontend technologies** and **Firebase**. It is lightweight, scalable, and suitable for online learning platforms, training programs, or fun quizzes.

**10. References**

1. Firebase Official Documentation
2. Mozilla Developer Network (MDN) – HTML, CSS, JavaScript
3. Project source code developed for seminar (Quiz Creator with Firebase).