DIY Projects

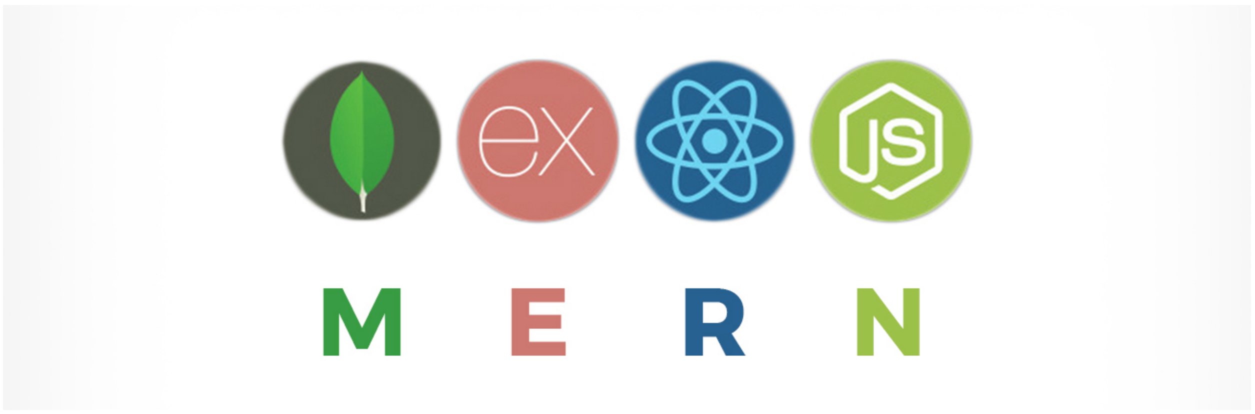
Quiz Maker-MERN

Expected time to finish **2-3 Days**

By **Ankit Das**

Introduction

We will create a simple quiz app that allows you to create and share your own quiz link. We will be using the MERN stack for this application, along with Mongo Atlas as our cloud database.



You can use either NodeJS or Django for the backend, where you have to create a local server and React to create the front-end.

Tech Stack

- **HTML/CSS**
- **JavaScript**
- **NodeJS (Express framework) or Django**
- **React**

Resources

MERN Stack

- MERN Stack: https://linktr.ee/codingclubiitg
- MERN Stack Crash Course Tutorial - YouTube

Extra Resources

- Intro to MongoDB Atlas in 10 mins | Jumpstart
- Mongoose v6.8.0: SubDocuments
- Tutorial v6.4.5 | React Router
- MERN Stack Full Tutorial & Project | Complete All-in-One Course | 8 Hours
- React: Full Modern React Tutorial - YouTube

Resources for Django

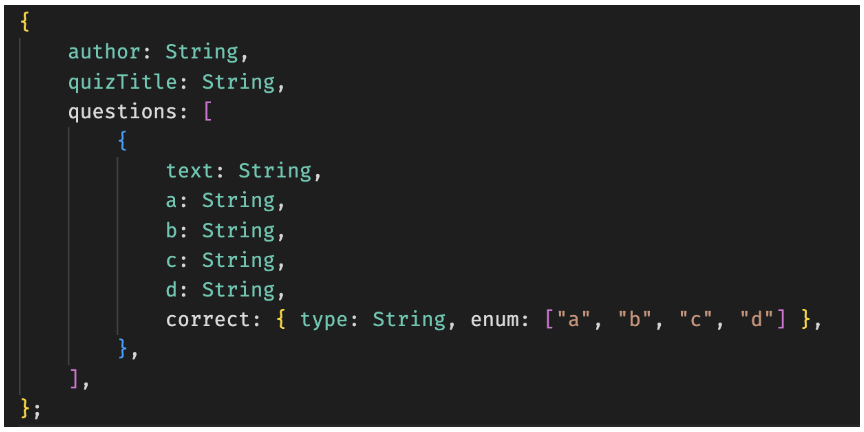
- Django Integration With MongoDB Tutorial
- React + Django To-Do App | Django REST Framework (Reference for React integration)

Implementation

(The schemas are given just for reference purposes, you can design your own)

A user can 1. Create, and 2. Respond to quizzes. We will handle the creation part first.

Define your Quiz Schema similar to the one given below:



```
{
    author: String,
    quizTitle: String,
    questions: [
        {
            text: String,
            a: String,
            b: String,
            c: String,
            d: String,
            correct: { type: String, enum: ["a", "b", "c", "d"] },
        },
    ],
};
```

The schema plays a major role in how our application is structured. You can make it as complex as you want, but for now, we will work with this simple schema.

To make a new Quiz, we need to design our front end to send data in the format of our quiz schema. Making such a system in React is reasonably easy.

The next step is to create routes for creating new quizzes with the data we receive from our front-end application and save it in Mongo Atlas. Also, create routes for fetching a particular quiz data.

The routes will look something like this:

POST /quiz

GET /quiz/:id

With this, our creation part is done. Again this is a very simple schema, and you can redesign the schema later.

Next, we will handle responding to a quiz. Using the unique id, we will fetch the correct quiz document from our backend, map over the questions, and render the question data. Based on the answer selected by the user and the correct answer, we can count and display the score. This is the basic implementation.

Next Steps

Add Authentication

You can set up a database for users, implement a login/signup feature and save user responses and the quizzes they have created.

User Authentication in Web Apps (Passport.js, Node, Express)

Add Complex Answer Options

Try to add different answer formats, such as integer answers, strings, etc., instead four options for every question.

Submission

Form Link: https://forms.gle/sHmvzS1gT4LBxXa28