10. Quiz App

Creating a Quiz App using the MEAN stack (MongoDB, Express.js, Angular, Node.js) is a great project. It involves both the creation of quizzes and the ability for users to take those quizzes. Below is a high-level overview of how to get started and some code snippets to guide you:

Project Setup and Structure

Set up a new project folder and structure for your Quiz App. Install the required Node.js packages and create a basic Angular application.

```
# Create a new Angular application

            ng new quiz-app
```

- ## Backend (Node.js & Express.js)

Create the backend of your Quiz App using Node.js and Express.js.

Installation of Packages

Install the necessary packages for Express.js, Mongoose (for MongoDB), and other dependencies.

```
npm install express mongoose cors
```

Setting up Express.js

Create your Express.js server, set up middleware, and handle routes.

- **javascript**

```
// server.js


const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');


const app = express();


// Middleware

app.use(express.json());

app.use(cors());


// Database connection

mongoose.connect('mongodb://localhost/quiz-app', {

  useNewUrlParser: true,

  useUnifiedTopology: true,

  useCreateIndex: true,
});

// Define Mongoose models for Quiz, Question, and User data
const Quiz = mongoose.model('Quiz', {
  title: String,
  questions: [{ type: mongoose.Schema.Types.ObjectId, ref:
'Question' }],
});

const Question = mongoose.model('Question', {
  text: String,
  options: [String],
  correctOption: Number,
});

const User = mongoose.model('User', {
  username: String,
  score: Number,
});

// Routes
app.get('/api/quizzes', async (req, res) => {
  const quizzes = await Quiz.find();

  res.json(quizzes);

});


app.get('/api/quizzes/:id', async (req, res) => {

  const { id } = req.params;

  const quiz = await Quiz.findById(id).populate('questions');

  res.json(quiz);

});
```

// Create similar routes for submitting quiz answers and user score tracking

- Frontend (Angular)

Create the frontend of your Quiz App using Angular. Design the user interface, implement quiz-taking functionality, and score tracking.

Design and UI

Design the user interface for your Quiz App using Angular components, templates, and styles.

Quiz Taking

Create components and forms for users to take quizzes, display questions, and allow them to select answers.

- **typescript**

// quiz-take.component.ts

```
import { Component } from '@angular/core';
import { QuizService } from './quiz.service';

@Component({
  selector: 'app-quiz-take',
  templateUrl: './quiz-take.component.html',
})
export class QuizTakeComponent {
  quiz: any;
  answers: number[] = [];

  constructor(private quizService: QuizService) {}

  loadQuiz(quizId: string) {
    this.quizService.getQuiz(quizId).subscribe((data) => {
      this.quiz = data;
    });
  }

  submitAnswers() {
    // Calculate the score and save it to the backend
  }
}
```

MongoDB

Create a MongoDB database to store quiz, question, and user data.

Scoring

Implement a scoring system in the Angular component to calculate and display the user's score after submitting answers.

Putting It All Together

Integrate the frontend and backend by making API requests from Angular components to Node.js routes. Ensure that you handle quiz-taking, answer submission, and score tracking properly.

Building a Quiz App is a fun project, and you can expand it with additional features such as user authentication, leaderboard tracking, and support for various question types (e.g., multiple choice, true/false, short answer).