Create a Quiz App I Nest.js learning assignment

2024-01-05

The scope of work:

In this task, your main objective is building the backend for a quiz application.

The API should have the following capabilities:

- Create a new quiz. Teachers should be able to create a new quiz by providing the quiz name along with a list of questions and their corresponding answers.
- Get questions for a quiz. Students should be able to fetch the questions for a specific quiz.
- Check answers. Students should be able to submit their answers for a quiz and receive the results. The results should include two numbers: the maximum possible points and the points obtained.

The quiz questions can be of different types:

- Single Correct Answer: The question will have multiple provided answers, but only one of them is correct. For example, "What is the capital of France? Options: (a) London, (b) Paris, (c) Rome, (d) Madrid."
- Multiple Correct Answers: The question will have multiple provided answers, and more than one can be correct. For example, "Which of the following programming languages are object-oriented? Options: (a) Java, (b) C, (c) Python, (d) Ruby."
- Sorting: The question will have several provided answers that need to be correctly sorted. For example, "Arrange the following events in chronological order. Options: (a) Declaration of Independence, (b) World War II, (c) First Moon Landing."
- Plain Text Answer: The question will require students to provide their own answer as plain text. The teacher will provide the correct answer, but students won't see it. The answer validation should be user-friendly, considering minor differences in capitalization, and punctuation. For example, "What is the famous phrase from Star Wars? Your Answer: _____."

Other Requirements:

- Implement a GraphQL-based API using NestJS and TypeScript.
- Cover the relevant parts of the code with unit tests.
- Use the PostgreSQL database and Docker to run it.
- No authentication is required.
- Provide instructions on how to run the application.
- Provide examples of GraphQL queries and mutations for each operation.
- All quizzes, including questions and answers, must be created within a single GraphQL mutation and a corresponding database transaction. Avoid splitting this operation into three separate mutations.
- Import database config from .env file.
- Make frequent commits following the commit naming conventions described in this guide

Nice to have:

- Error handling, providing meaningful error messages.
- Input validation.
- Follow documentation, avoid reinventing.
- Keep the code as simple as possible.