Toptal test task: Quiz Builder App

We would like to see how you would create a **simple quiz builder web API**. The API has to either be **GraphQL** or **RESTful**.

The app should allow authenticated users to create their own quizzes and other users to take them and see how many questions they got right.

Estimated time/effort: **10-20 hours**
Expected delivery time: **7 days**

Task scope and expectations

Your task is to build a quiz builder backend API. Users should be able to register and sign in using their email and password. Please note that this application should be able to support multiple different clients, including eg. mobile applications.

Authenticated users need to be able to create a quiz. Every quiz has a title and consists of 1-10 questions. Every question has 1-5 possible answers. Depending on the question type, it can either be a question with a single correct answer, or a question where visitor is expected to select all correct answers (eg. only one answer is correct, or multiple answers need to be selected).

Example of a single correct answer question:

- Moon is a star (Yes / No)

If the question is answered correctly, it is scored as 1 point. If it is skipped, it is scored as 0 points. If it is answered, but incorrectly, then it is counted as -1 point in the test.

Example of a multiple correct answers question:

- Temperature can be measured in (Kelvin / Fahrenheit / Gram / Celsius / Liters)

Every correctly selected answer adds a positive weight, and every wrong selected answer reduces it.

A couple of examples for the question above where the correct answers are Kelvin, Fahrenheit, and Celsius:

- Correctness weights for right options is ⅓ (or 0.333), as there are 3 correct answers.
- Correctness weights for wrong options is ½ (or 0.5), as there are 2 incorrect answers.

- If selected: Kelvin + Fahrenheit, the question can be considered ⅔ correct = 0.66 points.
- If selected: Kelvin + Fahrenheit + Celsius, the question can be considered fully correct = 1 point.
- If selected: Kelvin + Fahrenheit + Gram, the question can be considered: 0.333 + 0.333 - 0.5 = 0.166 points correct (16%).
- If selected all: Kelvin + Fahrenheit + Gram + Celsius + Liters, it is considered: ⅓ + ⅓ + ⅓ - ½ - ½ = 0% correct.
- If selected: Gram + Liter, it can be considered: -½ - ½ = -1 (-100%) correct, which means that this question will actually reduce the total score of the visitor, just like wrongly answered single-answer questions. So in this scenario, it would've been better if the user simply skipped it and didn't select anything.

Once questions are added, the quiz can be published. A published quiz can then be solved and answered by other authenticated users. Every user can solve the same quiz only once. Once solved, they should have access to the total score, represented as total correctness percentage, as well as individual scores they got on each question. They should **not** be able to see which answers were correct, only the score on each question.

Functional requirements

- All API calls (other than authentication endpoints themselves) need to be authenticated.
- API has to be usable by different clients, eg. not just browsers but mobile or desktop applications etc. Please choose the authentication strategy accordingly.
- Every user has the ability to create a quiz.
- A quiz consists of a quiz title and a list of questions.
- Every question contains question text and a list of possible answers.
- Questions can be either with a single or multiple correct answers. It should not be possible to submit multiple answers to a question that has only one correct answer.
- Quiz can only be taken by other users once published.
- Unpublished quizzes can be updated by the author, but can't be taken by other visitors.
- Published quizzes can't be edited anymore by the author, only deleted.
- Other users can take published quizzes and submit their solutions.
- Users should be able to access the list of all the solutions they have submitted to various quizzes.
- Users should also be able to see the solutions of other users **to their own quizzes** so they can get the statistics of how people perform on their quiz.

Technical requirements

- The server component needs to expose a **GraphQL or RESTful API.** It needs to be able to support non-web clients such as mobile apps.
- Be mindful of the edge cases and unexpected scenarios.
- Be mindful of security and data validation.
- It should not be possible to easily guess the quiz URLs, to prevent enumeration attacks where users could get a list of all the quizzes in the system.
- It is expected that certain quizzes can have 100 000+ visitors completing them.

Milestones and task delivery

- The deadline to submit your completed project is **1 week from the moment you received the project requirements**.
- It means project code must be submitted within 1 week from the moment that was delivered to you by email.
- If you schedule your final interview after the 1-week deadline, make sure to submit your completed project and all code to the private repository before the deadline
- Everything that is submitted after the deadline will not be taken into consideration.
- Please do not commit any code at least 12 hours before the meeting time so that it can be reviewed. Anything that is submitted after this time will not be taken into consideration.
- **Short project video** - after you are done, please provide us with a simple and short video recording of the functionality demonstrating the API endpoints and submit it to the GIT repository. Please do not spend too much time on this step. Our goal is to see your project presentation as soon as possible and how you would provide a quick for the client. So, please pick any free or preinstalled software that can help you do this.
- Please join the meeting room for this final interview on time. If you miss your interview without providing any prior notice, your application may be paused for six months.