

Test Task for The Candidate

The essence of the task

You have to develop a simple test system in which the user enters their name, chooses a test, executes it, and at the end sees their result.

You have to use C# .NET , MONGO/COSMOS DB, HTML, CSS, JavaScript. C# code should be object-oriented. The design solution is up to you - you can express yourself freely.

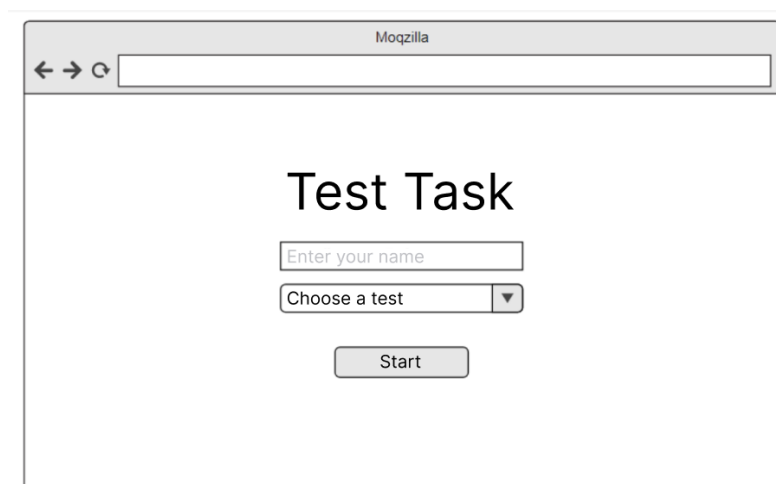
The test consists of 3 different views:

- 1) Homepage - the user enters their name and chooses one of the tests in the database.
- 2) Test question view - each question has answer options. One of them is correct.
- 3) Result view - the user sees their result.

Detailed description of views

1) Homepage

The user enters their name and selects a test. If no word is entered or no test is selected, the user cannot go to the next view - an error message must be displayed.



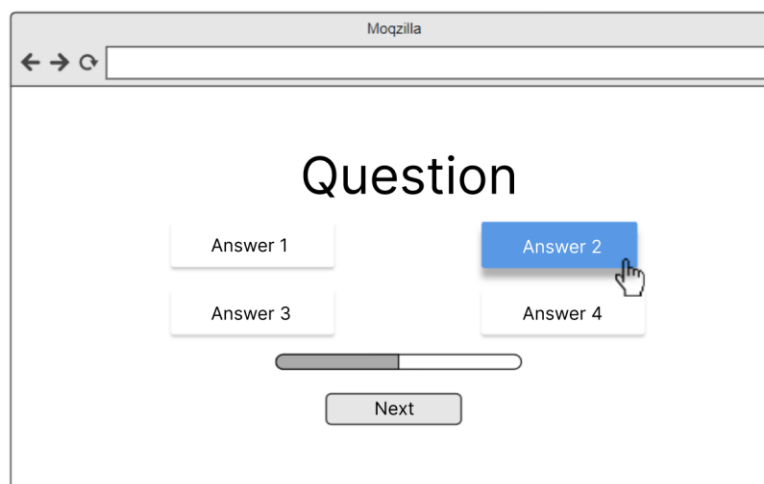
The screenshot shows a web browser window titled "Mozilla". The address bar is empty. The main content area displays the title "Test Task" in a large, bold font. Below the title, there is a text input field with the placeholder text "Enter your name". Below that is a dropdown menu with the text "Choose a test" and a downward arrow. At the bottom, there is a "Start" button.

2) Test question view

The user is completing the test. Each test is composed of a sequence of questions, minimum 3. The user chooses one answer option and goes to the next question. If no answer option is chosen, the user cannot go to the next question.

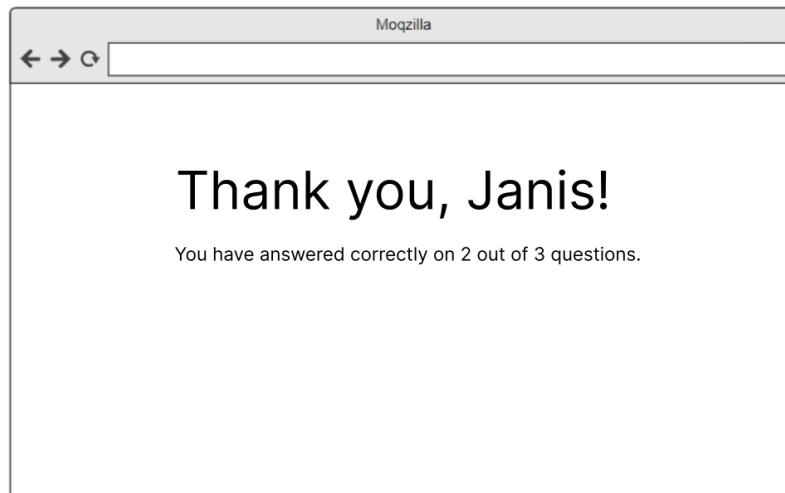
Important: the question can have a variable number of answers. One question may have 2 answer options, another, for example, 11.

A dynamic progress bar (progress bar) is displayed according to the question's sequence number.



3) Result View

In this view, it is necessary to display again the user's name, which they entered on the homepage, as well as the total result - how many questions there were in total and how many of them the user answered correctly.



Technical conditions

Design

Sizes, fonts, colors, distances – a free opportunity for improvisation. The visuals in the previous steps are only wireframes of the design layout.

C#

1. The code must be object-oriented.
2. Add comments to your code.
3. It is not allowed to use any frameworks. The code structure you have to build by your own, but Composer and third-party libraries can be used.
4. Unit testing code must be included for each method/unit.
5. Attention should also be paid to code formatting, style, and good programming practices and security.

Database

1. COSMOS DB or MONGO DB can be used.
2. There can be several tests in the database at the same time.
3. An unlimited number of questions are possible for each test.

4. Each question has 2 - n answer options. Respectively, the maximum number of answer options is not strictly defined. One of the answer options is correct. The rest are wrong.
5. Each time the user answers a question, this fact must be saved in the database - which user answered which test, which question, which answer.
6. The final result should be stored in the database for each time the test is completed - it is necessary to know which user completed which test and how many questions were answered correctly.
7. It must be assumed that the database will "grow" very large over time, so the relevant indexes must be assembled so that queries are optimal.

What has to be submitted

All code must be submitted with accompanying documentation of the preparatory steps to be taken on the server where this code will be tested (what are the parameters to be configured, etc.). The database tables have to be saved in a dump file.

Additional points will be given if the task is submitted not in an archived (eg .zip) form, but placed on a GIT repository (eg gitlab.com, github.com, etc., absolutely free choice). In this case, a link to the GIT repository has to be sent. Depending on the chosen repository - if necessary, you must also specify the access data, how it can be accessed.