*Thank you for applying for the position of **Full-Stack Web Developer** at **Printful Bits**. We'd be happy to continue the further selection process with you and look forward to receiving your take on our technical task below.*

Instructions

1. Before you send us your task or any additional personal information, make sure you've reviewed our **recruitment process privacy policy**.

2. Please let us know when we can expect your task (applicants usually send us their tasks in **3 business days**).

Task

You need to develop a simple quiz system, where a user can enter their name, choose a test, and see their results.

You need to develop the system using Vue.js with ES6 (or newer JavaScript syntax). You can submit it in plain vanilla JS, React, or Angular, but we highly prefer Vue.js. The design is not limited - you are allowed to use Material, Tailwind, Bootstrap, etc., whichever suits best for you to achieve the most beautiful result. Remember, that a nice look and feel makes the first impression of your work.

The test consists of 3 different views:

1. Homepage view where the user inputs their name and chooses one of the tests.
2. Test question view where each question has:
    - The text of the question (always);
    - a picture of the question (sometimes);
    - at least two possible answers (but there's no limit to the max number of answers). Always exactly one of the answers is correct.

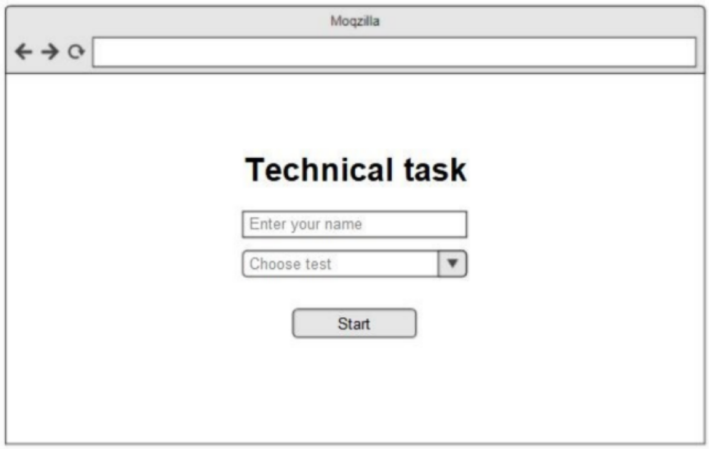
3. Results view where the user sees their results and can either retry the test or get back to the homepage view.

Detailed view descriptions:

1. Homepage view

The user enters their name and chooses one of the tests.

If the user has forgotten to input their name or choose a test, they can't access the next page, and an error message appears.

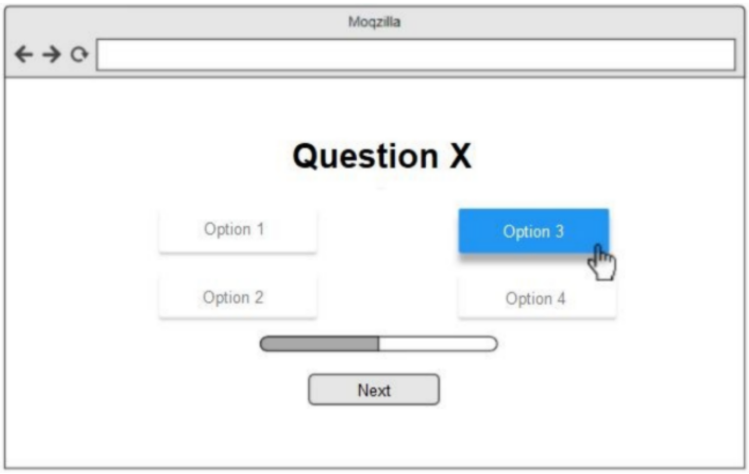


2. Test questions view

The user takes the test. They choose one of the possible answers and go to the next question. Unless an answer has been chosen, the user can't move on to the next question.

There's a dynamic progress bar that shows the user's progress. It changes depending on which questions the user is on.

For the questions with images place the image between question and answers.



3. Results view

This view needs to show the name user entered in the homepage view, and the final results - how many questions there were, and how many of them the user answered correctly.

The *Retry* button will restart the quiz while the *Next quiz* button will lead to the home screen.

Technical requirements (Front-end):

1. You are expected to use Vue.js with ES6 or newer JavaScript. (It will be ok also with vanilla JS, React, or Angular but we prefer Vue.js)
2. The design has to be responsive:
    - if the width of the screen is wide enough, the answers have to appear in 2 columns (as in the example above);
    - if the width of the screen is not wide enough to show 2 columns (for example, on the mobile phone), then all answers need to appear below each other).
3. When it comes to sizes, fonts, colors, spaces, etc., feel free to improvise. The examples in the previous steps are just schematic sketches of the layout. You need to pick the text color, fonts, background color, button colors, etc. You can use existing CSS frameworks like Material, Bootstrap, Tailwind, etc. Remember, that we will appreciate your UI/UX skills as well.

Technical requirements (Back-end):

Please use our custom quiz API for accessing quizzes, questions and requesting correct answers. Documentation (Postman) on how to use it can be found here:

https://bit.ly/2UxVDMI

Please implement your own lightweight backend. As for Printful Bits projects we use either Laravel or Node.js, we would prefer, that also your backend would be implemented using one of these 2 technologies.

Remember, that:

1. The tests, questions, answers, and test results for every test need to be stored in the back-end.
2. The data should be kept in the database of your choice.

3. After submitting each answer, a request needs to be sent to the back-end, where it's determined whether the answer is correct or not.

What to submit:

You need to submit all of the code with a readme file where you concisely describe the architecture of your chosen solution and particular reasons for that. The database tables need to be saved in a SQL file or .json (MongoDB).

You need to submit an assignment in a GIT repository of your choice (e.g., gitlab.com, github.com, etc.) instead of an archived folder (e.g., zip). Please submit us a link to your GIT repository. Depending on the repository you might also need to send us data to access your assignment.

Thank you for participating and good luck!