VTech - coding challenge v3

Readme

- First of all, thank you for accepting the challenge.
- In this challenge, you will get familiar with Next.js or even more things.
- Please try your best to finish the “Basic requirement”, “Building more functionalities” and “Building Simple CRUD APIs”.
- It would be great if you can finish either of the “Bonus challenges”. If you can finish both of them, it would be perfect!
- Please take no longer than a week to complete the challenge.
- Have fun and happy coding!

Basic requirement

1. Complete a Todo list with Next.js.
2. The list should contain an input field and a list element.
3. After typing some text in the input field, hitting the enter key will add a new item to the list.
4. No need for UI polishing. Just stick to the browser default.
5. Use React hooks as much as possible.
6. Create an API that list of Todos and integrate with Todo list, you may create dummy data for this:

```
Method: GET; Endpoint: /api/todo
Response:
[
{
  id: "string",
  todo: "string",
  isCompleted: "boolean",
  createdAt: "timestamp"
},
{
  id: "string",
  todo: "string",
  isCompleted: "boolean",
  createdAt: "timestamp"
},....
]
```

Building more functionalities

1. Valid Todo list
    a. There should be no “empty” item in the list
2. Unique Todo list
    a. There should be no duplicate item in the list
    b. Adding a duplicate item will warn the users

3. Removable Todo list
    a. There should be a “remove” button next to the item when users hover on it.
    b. The item should be removed from the list by clicking on the “remove” button.
4. Editable Todo list
    a. There should be an “edit” button next to the item when users hover on it.
    b. After clicking on the “edit” button, the input field will have the text of the item and users can edit the item by hitting the “enter” key again.
    c. Use the same input bar as you add a new Todo list.
5. Filter Todo list
    a. When the user starts typing in the input bar, apply a simple text matching to the list of todo and only show the filtered result.
    b. When there is no matched result, please show “No result. Create a new one instead!”
6. Mark as Complete
    a. There should be an “Mark as Complete”/“Mark as Incomplete” button next to the item when users hover on it.
    b. Display the todo as strikethrough text if the todo is marked as completed.

Building Simple CRUD APIs

1. Create APIs that update the todo list and integrate with Todo list.
    a. Create Todo

```
Method: POST; Endpoint: /api/todo
Request Body:
{
  id: "uuid",
  todo: "string",
  isCompleted: "boolean",
  createdAt: "timestamp"
}
Response:   success
```

    b. Update Todo

```
Method: PUT; Endpoint: /api/todo/{id}
Request Body:
{
  id: "uuid",
  todo: "string",
  isCompleted: "boolean",
  createdAt: "timestamp"
}
Response:   success
```

    c. Delete Todo

```
Method: DELETE; Endpoint: /api/todo/{id}
Response:   success
```

Bonus challenges

1. Allow multi users editing to the list by applying a database synchronization
    a. Synchronize the data to a cloud database while the users are editing the todo list.
    b. You might consider using Firebase, Superbase or Fauna as the cloud database.
    c. No user authentication is required.
2. Finish the test in TypeScript or with Flow typing

Getting started

- In order for us to evaluate easily, you might create an online Next.js code sandbox with https://stackblitz.com/edit/nextjs. Or you can pick the way you want to make it but make sure we can view your code and run the application without any hassle.
- When you choose to use any online code sandbox, please save the url somewhere as you might accidentally leave the page and the url might be lost.
- Complete the test and submit the url or any related project file to the person in contact.
- Please also estimate the total hours that you spent on this project. We don't examine based on the hours spent. This metric is useful for us to anchor the level of effort.