Web Programming

Tutorial 12

To begin this tutorial, please create a React project using the `vite` module as instructed in the previous tutorial. When you finish, zip all your source codes (excluding the `node_modules` folder) to submit to this tutorial's submission box.

Activity 1

Goals

- To move the business logic of the flashcards application to a Node.js backend.
- To enable the React front-end to fetch data from the backend.

General guidelines

Create a Node.js (Express) application and put the following dictionary object on it:

```
const dict = {
    "pretty": "xinh đẹp",
    "car": "xe hơi",
    "study": "học tập",
    "life": "cuộc sống",
    "enormous": "to lớn",
    "computer": "máy tính"
};
```

On the server side (Express application), create 2 endpoints:

1. An endpoint which returns the total number of words:

`http://localhost:8000/wordcount`

This endpoint returns the dictionary' size in a JSON object, for example:

`{ "wordcount": 6 }`

2. An endpoint which returns a word and its meaning for a given index:

`http://localhost:8000/getword/:index`

(note the URL parameter named `index` in the above path)

Example of this endpoint's output:

```
{
    "index": 0,
    "word": "pretty",
    "def": "xinh đẹp"
}
```

On the client side (React application), use `fetch` to get the total number of words, then use `fetch` to get any word based on index.

(*) Note: You will have to enable CORS by setting the following header on the server side:

```
res.setHeader('Access-Control-Allow-Origin', '*');
```

Otherwise, the client will be unable to fetch data from the back-end.

To the user, this flashcards application has the same features as the one we built in the last tutorial. However, from the internal, the system now works differently. To sum up, the goal of this activity is to build a simple web application where the front-end replies on the back-end's functionality.

Activity 2

In this activity, you'll try to add gameplay functionality to a the tic-tac-toe game UI that you built in the last tutorial. You should:

- Use function components and the `useState` hook.
- Define the game board's state using an array which has 9 elements.
- Create a method which places either an `X` or an `O` when the user clicks on an empty `Square`. Show an `alert` box when the user clicks on an occupied square.
- Let the `X` player and the `O` player take turn to play.
- Create a button to reset the gameboard (clear everything).
- Store all previous game boards in a state called `history` and let user go back to a certain previous step.
- Detect if a user wins after each move. When a user wins, show an alert box to announce the winner, and reset the game board.