**paskaitos užduotis**

sukurkite aplikaciją kurioje galėtumėte atvaizduoti savo nuotaiką. bus tekstas ir 3 mygtukai paspaudus primą mygtuką tekstas pasikeis į "Laimingas" paspaudus antrą į "Liūdnas" trečią į "Piktas". Mygtukai ir tekstas yra komponentai

**4 paskaitos užduotis**

puslapyje yra 2 mygtukai posts ir comments. paspaudus posts atsivaizduoja postai, o paspaudus comments atsivaizduoja komentarai. Informaciją imti iš jsonplaceholder ir daryti requestam naudojam axios

vartotojai turi matyti kada yra daromas requestas (loading)

**6 paskaitos užduotis**

**Task Description:** Create a React component that fetches and displays a random quote from an API when the component is mounted.

**Requirements:**

1. Create a functional component named RandomQuote.
2. Use the useEffect hook to fetch a random quote from a free public API (e.g., the "<https://api.quotable.io/random>" API) when the component is mounted. (kai užsikrauna)
3. Display the fetched quote along with the author's name on the screen.
4. Handle any loading or error states and display appropriate messages to the user.

**Bonus (Optional):**

1. Add a button that allows the user to fetch a new random quote without refreshing the entire page.

**8-9 paskaitų savarankiška užduotis**

Build a simple todo list app with Node.js and React. The app should allow users to create, read, and delete todo items.

Endpoints:

/todos (GET): Get all todo items.

/todos (POST): Create a new todo item.

/todos/:id (DELETE): Delete a todo item by ID.

Components:

TodoList component: Displays a list of todo items.

TodoItem component: Displays a single todo item.

CreateTodo component: Allows users to create a new todo item.

Additional challenges:

Add the ability to filter and sort the todo list.

Add the ability to add due dates and priorities to todo items.

Add the ability to mark todo items as complete and incomplete.

**10 paskaitos užduotis**

1.Naudodami “React Router”, sukurkite SPA(Single page application), kuri turės šiuos “pagrindinius” komponentus:

a.Header;

b.Footer;

2.Header dalyje turi būti:

a.Logotipas;

b.Meniu su nuorodomis atskirus puslapių komponentus:

i.Pagrindinis, kuriame bus full width ir height paveikslėlis;

ii.Apie mus, kuriame bus puslapio pavadinimas, nuotrauka ir tekstas;

iii.Naujienos, kuriame bus is [API](https://jsonplaceholder.typicode.com/posts) (https://jsonplaceholder.typicode.com/posts) kraunama dvi naujienos su antrašte ir tekstu;

iv.Paslaugos, kuriame bus 3 kortelės su paslaugos pavadinimu ir tekstu po ja;

v.Atlikti darbai, pateikiamas 8 nuotraukos (4 stulpeliai);

vi.Kontaktai, kontaktų forma (vardas, el. paštas, žinutė, ją pateikus atsiranda užrašas su padėka už pateikimą);

3.Footer dalyje pateikiamas meniu (eilutėje) ir “Visos teisės saugomos”

**11 paskaitos užduotis**

In this project, you'll create an interactive Pokémon data retrieval app that allows users to specify the Pokémon they want information about. You'll make API calls to the "PokeAPI" (<https://pokeapi.co/>) to fetch data and display it in a user-friendly format. Here's a breakdown of the components you need to create and the fields you should display:

**Components to Create:**

**Input Component:** Develop a user interface where users can enter the name or ID of the Pokémon they want information about. This can be a simple HTML form with a text input field and a submit button, or it can be an input prompt in a command-line application.

**Display Component:** Create a section in your app where the retrieved Pokémon data will be displayed. This could be a section on a web page, a pop-up modal, or a well-structured console output. Make sure it's user-friendly and easy to read.

**API to Call:**

You will be making API requests to the "PokeAPI" (<https://pokeapi.co/>). This API provides comprehensive data about Pokémon, including their names, abilities, types, and much more. Use the base URL for the PokeAPI, which is <https://pokeapi.co/api/v2/>.

**Fields to Display:**

The information you retrieve from the API can be quite extensive, but you should aim to display at least the following fields:

**Pokémon Name:** Display the name of the Pokémon the user specified.

**Types:** Indicate the type or types of the Pokémon, such as Water, Fire, Electric, etc.

**Height and Weight:** Include the height and weight of the Pokémon.

**Image:** If you choose to enhance your project, you can include an image of the Pokémon to make it visually appealing.

**14 paskaitos užduotis**

**A shopping cart modal**

This modal would be used to display the items that a user has added to their shopping cart. It would also allow the user to remove items from their cart, update the quantities of items in their cart, and proceed to checkout.

The modal could be triggered when the user clicks on a shopping cart icon in the header of the website. It would then be rendered over the top of the main content of the website, using a React portal.

The modal would contain a table listing the items in the user's cart, along with their quantities and prices. There would also be buttons for removing items from the cart, updating quantities, and proceeding to checkout.

**16 paskaitos užduotis**

**Task:** Create a custom button component that allows you to listen for hover events from a parent component.

**Steps:**

1. Create a new component called HoverableButton.
2. Use the forwardRef function to forward the ref to the underlying button element.
3. Export the HoverableButton component.
4. In the parent component, create a ref to the HoverableButton component.
5. Render the HoverableButton component and pass the ref to it.
6. To listen for hover events on the button element, use the ref.current property.

ant ref kintamojo panaudosite addEventListener funkciją, kad uždėti eventus iš tėvinio komponento

**11 paskaitos užduotis**