МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ОСТРОЗЬКА АКАДЕМІЯ"
Кафедра економіко-математичного моделювання та інформаційних технологій

ЗВІТ
Лабораторна робота №5
Створення скелетону проекту з Redux і React Router

здобувачки вищої освіти
першого (бакалаврського) рівня
третього року навчання групи КН-31
спеціальності 122 Комп'ютерні науки
ОПП «Комп'ютерні науки»

Лайтер Ярини Семенівни
(прізвище, ім'я, по батькові)

База лабораторної - кафедра економіко-математичного моделювання та інформаційних технологій

Керівник лабораторної від університету

_____
(науковий ступінь, учене звання керівника)

_____
(Власне ім'я та ПРІЗВИЩЕ)

Члени комісії:

_____

Острог - 2023

Зміст

Опис

Тема: Створення скелетону проекту з Redux та React Router
Мета: Ознайомити студентів зі створенням додатків на React з використанням Redux та редукторів (reducers)

Завдання

1. Створіть новий проект React за допомогою Vite або будь-якого іншого способу, який вам подобається та встановіть необхідні залежності:
    1. redux
    2. react-redux
    3. react-router-dom

2. Створіть структуру каталогів для вашого проекту. Наприклад:

    src/ components/ reducers/ actions/ containers/ App.jsx main.jsx

3. Створіть основний компонент "App.jsx", який буде кореневим компонентом вашого додатку. В цьому компоненті ви повинні використовувати React Router для налаштування маршрутів. Створіть два простих маршрути, наприклад, "/home" і "/about".

4. Створіть редуктори (reducers) та дії (actions) для вашого Redux-стору. Наприклад, створіть редуктор для збереження стану додатку та дію для оновлення цього стану.

5. Підключіть Redux до вашого додатку за допомогою react-redux та налаштуйте його так, щоб компонент "App.jsx" мав доступ до Redux-стору.

6. В компоненті "App.jsx" виводьте дані з Redux-стору, наприклад, виведіть поточний стан додатку на сторінці.
   Створіть компоненти для сторінок "/home" та "/about", які будуть відображати різні дані на кожній сторінці.
   Впевніться, що при переході між маршрутами дані в Redux-сторі залишаються незмінними, і вони правильно оновлюються та виводяться на відповідних сторінках.
   Переконайтеся, що ваш додаток працює на локальному сервері та може бути запущений та перевірений в браузері.
   Додайте стилізацію до свого проекту за допомогою CSS або будь-якого іншого інструменту стилізації, який вам подобається.
   Завершіть проект та переконайтеся, що він працює на вашому локальному сервері, та всі функції працюють коректно.

Виконання

1. **Створення нового проекту React:**

Створюємо проект. Вводимо в терміналі комаду npm init vite@latest та підключаємо залежності:

```
PS D:\OA_3\REACT\lab5>  npm init vite@latest
√ Project name: ... lab5_vite
√ Select a framework: » React
√ Select a variant: » JavaScript

Scaffolding project in D:\OA_3\REACT\lab5\lab5_vite...

Done. Now run:

  cd lab5_vite
  npm install
  npm run dev

PS D:\OA_3\REACT\lab5> cd lab5_vite
PS D:\OA_3\REACT\lab5\lab5_vite> npm install redux react-redux react-router-dom

added 280 packages, and audited 281 packages in 23s

97 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Рис 1.1 Виконання команд

2. **Створюємо структуру каталогу**:

Створюємо нові папочки:

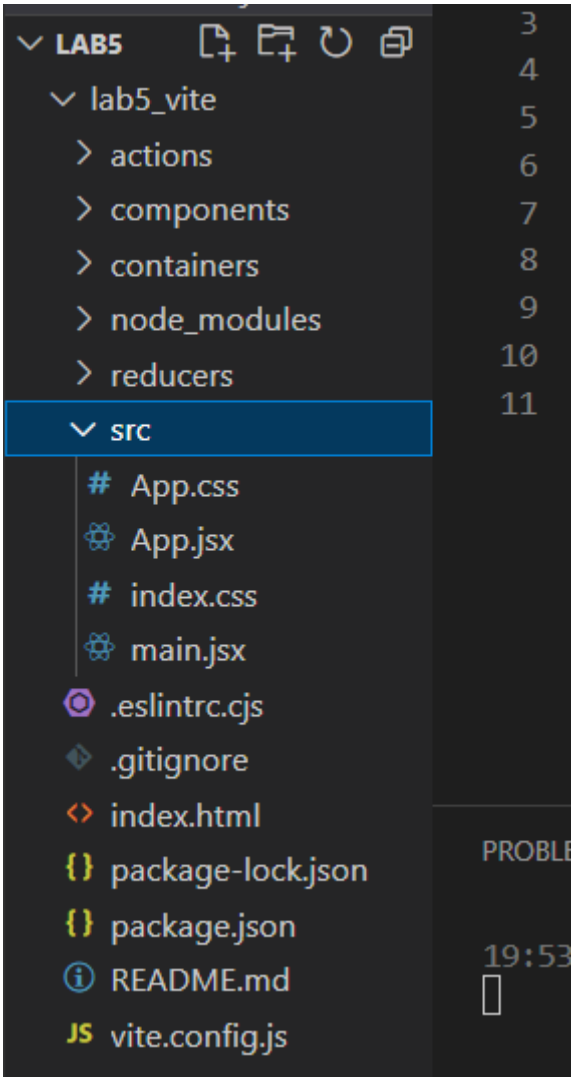

Рис 2. 1 Створення структури проекту

3. **Створюємо компонент "App.jsx"та два простих маршрути "/home" і "/about": в**

Створюємо компоненти Home.jsx та About.jsx:

lab5_vite > components > Home.jsx > ...

```jsx
import React from 'react';

function Home() {
  return (
    <div>
      <h1>Welcome to the Home Page</h1>
      <p>This is the home page of our application.</p>
    </div>
  );
}

export default Home;
```

Рис 3.1 Код сторінки Home.jsx

Рис 3.2 Код сторінки About.jsx

В компоненті App.jsx використовуємо React Router для налаштування маршрутів та створюємо два простих маршрути "/home" і "/about":

Рис 3.3 Код сторінки App.jsx

В результаті отримуємо:

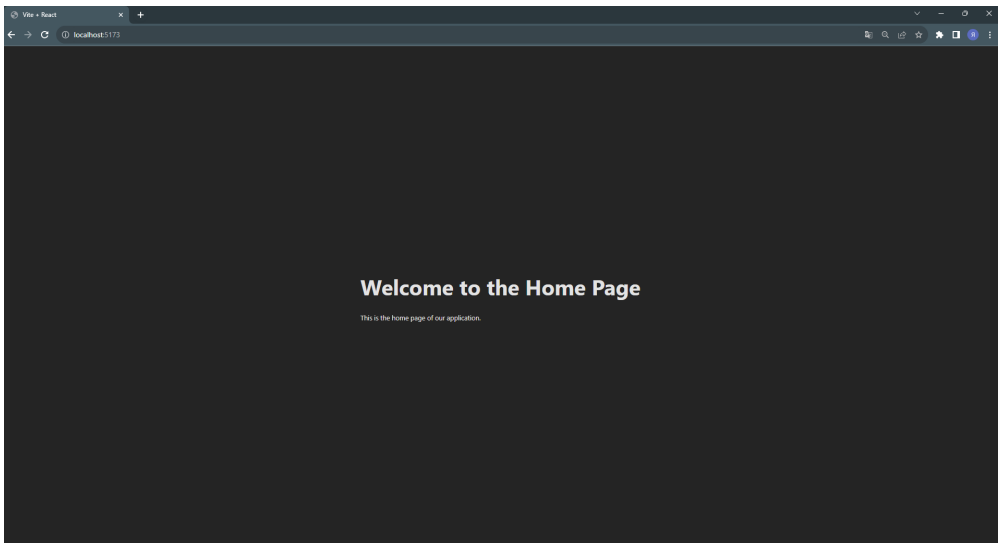

Рис 3.4. Результат програми

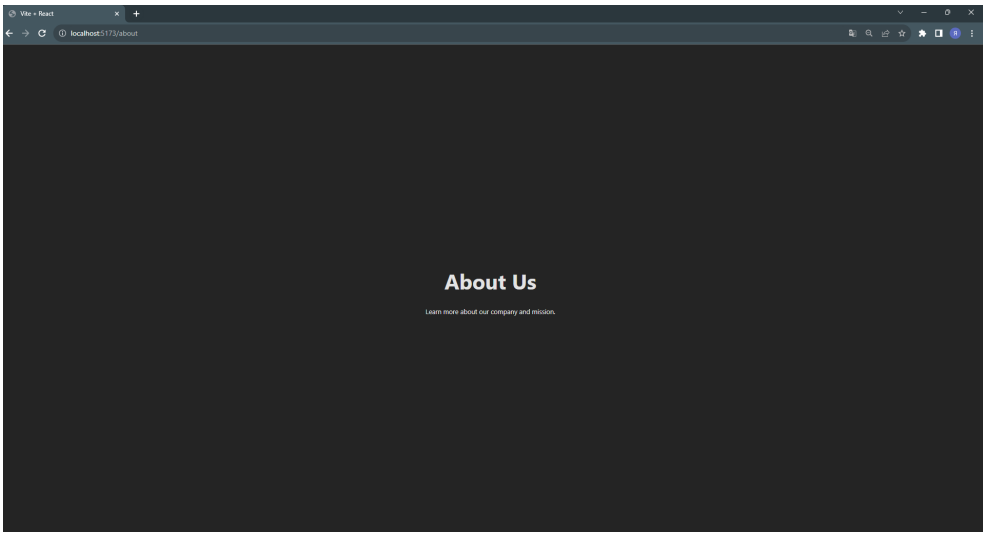

Рис 3.5. Результат програми

4. **Створюємо редуктори (reducers) та дії (actions):**

   Створюємо редуктор appReducer.jsx, який відповідає за лічильник counter. Коли відбувається дія INCREMENT або DECREMENT, appReducer змінює відповідну частину стану додатку:

Рис 4.1 Код сторінки appReducer.jsx

Та створюємо *appActions.jsx*, який відповідає дії, які використовуються в компонентах або діяльностях для відправки запитів до Redux-стору. Коли дія відправляється, відповідний редуктор обробляє цю дію і вносить відповідні зміни в стан додатку*:*

Рис 4.2 Код сторінки appActions.jsx

5. **Підключаємо Redux до додатку за допомогою react-redux:**

Підключаємо відповідні бібліотеки у файлі main.jsx, щоб додаток функціонував:

lab5_vite > src > main.jsx

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import { Provider } from 'react-redux';
import store from '../containers/store';
import App from './App.jsx';
import './index.css';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
);
```

Рис 5.1 Код компонента main.jsx

6. **Виводимо дані з Redux-стору та створюємо компоненти для сторінок "/home","/about" та “/todo”:**

Спочатку створимо компоненти з різними даними в них:

lab5_vite > components > Home.jsx > ...

```jsx
import React from 'react';
import { Link } from 'react-router-dom';
import { useSelector, useDispatch } from 'react-redux';
import { increment, decrement } from '../actions/appActions';
import './Home.css';

function Home() {
  const appState = useSelector((state) => state.app);
  const dispatch = useDispatch();

  const handleIncrement = () => {
    dispatch(increment());
  };

  const handleDecrement = () => {
    dispatch(decrement());
  };

  return (
    <div className="home-container">
      <h1>Welcome to the Home Page</h1>
      <p>Current App State:  {appState.counter}</p>
      <div className="button-container">
        <button onClick={handleIncrement} className="increment-button">Increment</button>
        <button onClick={handleDecrement} className="decrement-button">Decrement</button>
      </div>
      <nav>
        <ul className="nav-links">
          <li>
            <Link to="/about" className="nav-link">About</Link>
          </li>
          <li>
            <Link to="/todo" className="nav-link">Todo</Link>
          </li>
        </ul>
      </nav>
    </div>
  );
}

export default Home;
```

Рис 6.1 Код компонента Home.jsx

lab5_vite > components > About.jsx > ...

```jsx
import React from 'react';
import { Link } from "react-router-dom";
import './Home.css';

function About() {
  return (
    <div className="about-page">
      <h1>About Us</h1>
      <p>
        Welcome to our About page.
      </p>
      <p>
        We are a fantastic team of developers dedicated to creating amazing web applications.
      </p>
      <p>
        If you want to learn more about our work, feel free to explore our projects in the "Home" section.
      </p>
      <nav>
        <ul className="nav-links">
          <li>
            <Link to="/" className="nav-link">Home</Link>
          </li>
          <li>
            <Link to="/todo" className="nav-link">Todo</Link>
          </li>
        </ul>
      </nav>
    </div>
  );
}

export default About;
```

Рис 6.2 Код компонента About.jsx

Рис 6.3 Код компонента Todo.jsx

Далі викликаємо всі компоненти в *App.jsx*:

```
lab5_vite > src > App.jsx > ...
 2  import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
 3  import Home from '../components/Home';
 4  import About from '../components/About';
 5  import Todo from '../components/Todo';
 6
 7  function App() {
 8    return (
 9      <Router>
10        <Routes>
11          <Route path="/" element={<Home />} />
12          <Route path="/about" element={<About />} />
13          <Route path="/todo" element={<Todo />} />
14        </Routes>
15      </Router>
16    );
17  }
18
19  export default App;
```

Рис 6.4 Код компонента App.jsx

При переході між маршрутами дані в Redux-сторі залишаються незмінними, і вони правильно оновлюються та виводяться на відповідних сторінках (а зокрема на сторінці Home, About). На сторінці Todo дані не будуть зберігатися і при кожному вході на сторінку все буде “перезагружатися”\повертатися в початковий стан.

7. Результати з стилями та перевірками:

Додаток працює на локальному сервері та був запущений та перевірений в браузері.
Додано стилізацію до свого проекту за допомогою CSS.

В результаті отримуємо:

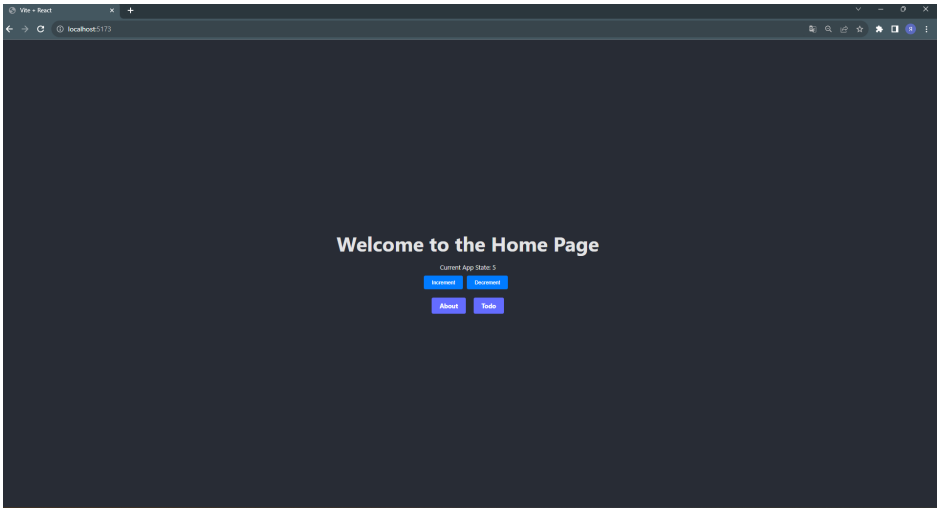

Рис 7.1 Результат запуску програми

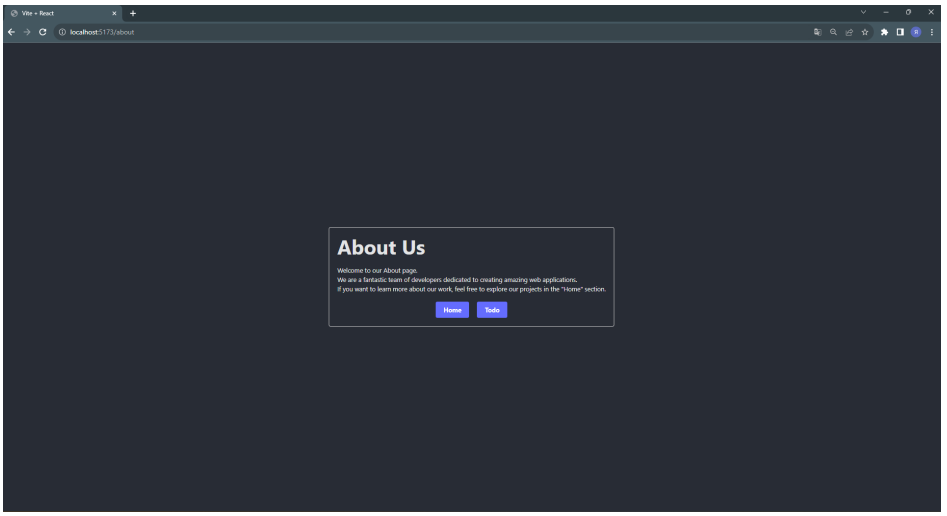

Рис 7.2 Результат запуску програми

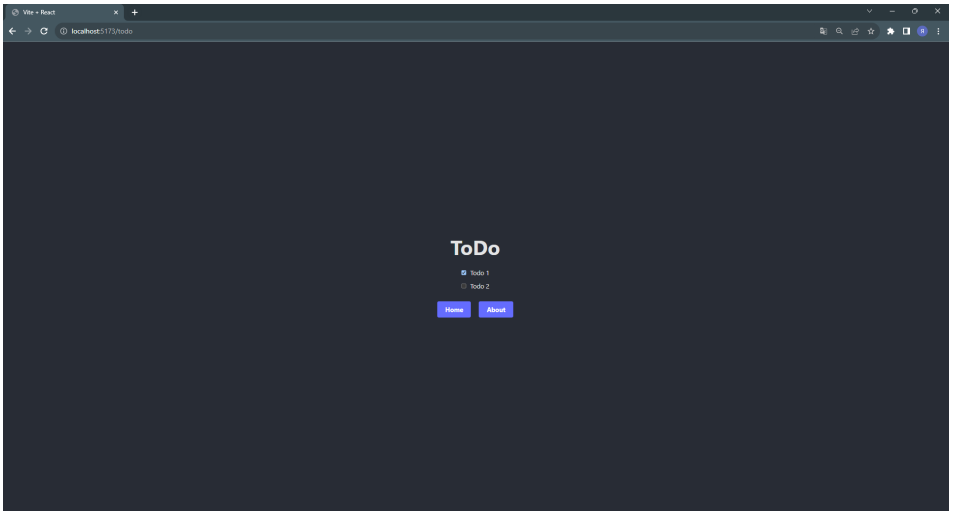

Рис 7.3 Результат запуску програми

Висновок:

На цій лабораторній роботі ми створили проект React з використанням Vite, налаштували структуру каталогів, підключили Redux та React Router. Окрім того, створили компоненти для сторінок "/home", "/about" та "/todo", які відображають різні дані на кожній сторінці.

Ми переконалися, що дані в Redux-сторі залишаються незмінними при переході між маршрутами, і вони правильно оновлюються та виводяться на відповідних сторінках.

Також ми додали стилізація до проекту для поліпшення зовнішнього вигляду та завершено проект, переконавшись у його працездатності на локальному сервері.