

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

Зміст	2
Опис	3
Виконання	4
1. Створення нового проекту React:	4
2. Створюємо структуру каталогу:	4
3. Створюємо компонент "App.jsx" та два простих маршрути <code>"/home"</code> і <code>"/about"</code> :	5
4. Створюємо редуктори (reducers) та дії (actions):	7
5. Підключаємо Redux до додатку за допомогою react-redux:	9
6. Виводимо дані з Redux-стору та створюємо компоненти для сторінок <code>"/home"</code> , <code>"/about"</code> та <code>"/todo"</code> :	9
7. Результати з стилями та перевірками:	12
Висновок:	13

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-сторю. 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-сторю.
6. В компоненті "App.jsx" виводьте дані з Redux-сторю, 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 команду `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 папки:

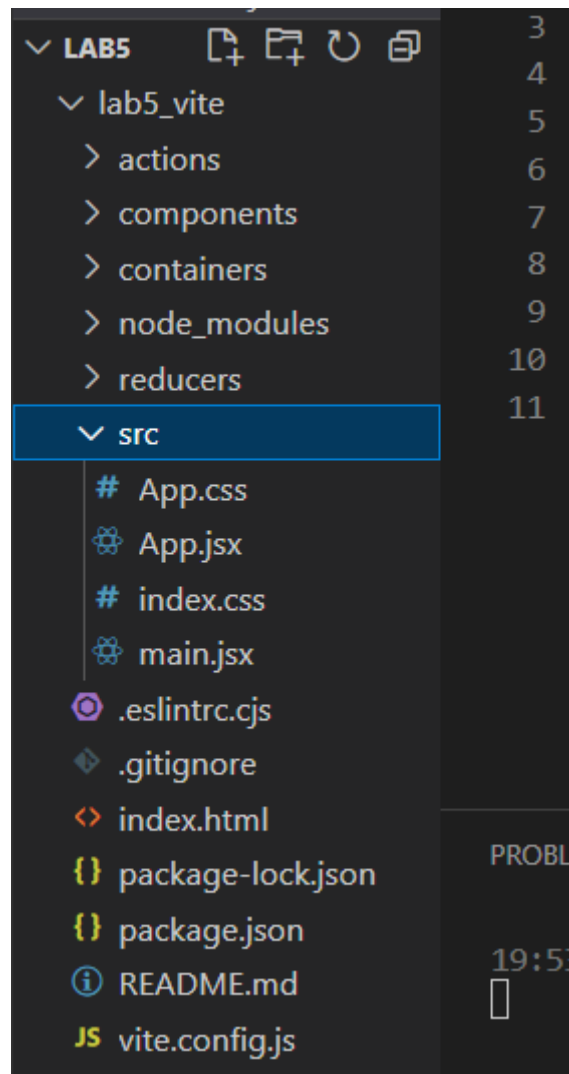


Рис 2. 1 Створення структури проекту

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 > ...
1  import React from 'react';
2
3  function Home() {
4    return (
5      <div>
6        <h1>Welcome to the Home Page</h1>
7        <p>This is the home page of our application.</p>
8      </div>
9    );
10 }
11
12 export default Home;
```

Рис 3.1 Код сторінки Home.jsx

```

lab5_vite > components > About.jsx > ...
1  import React from 'react';
2
3  function About() {
4    return (
5      <div>
6        <h1>About Us</h1>
7        <p>Learn more about our company and mission.</p>
8      </div>
9    );
10 }
11
12 export default About;

```

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"`:

```

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

```

Рис 3.3 Код сторінки App.jsx

В результаті отримуємо:

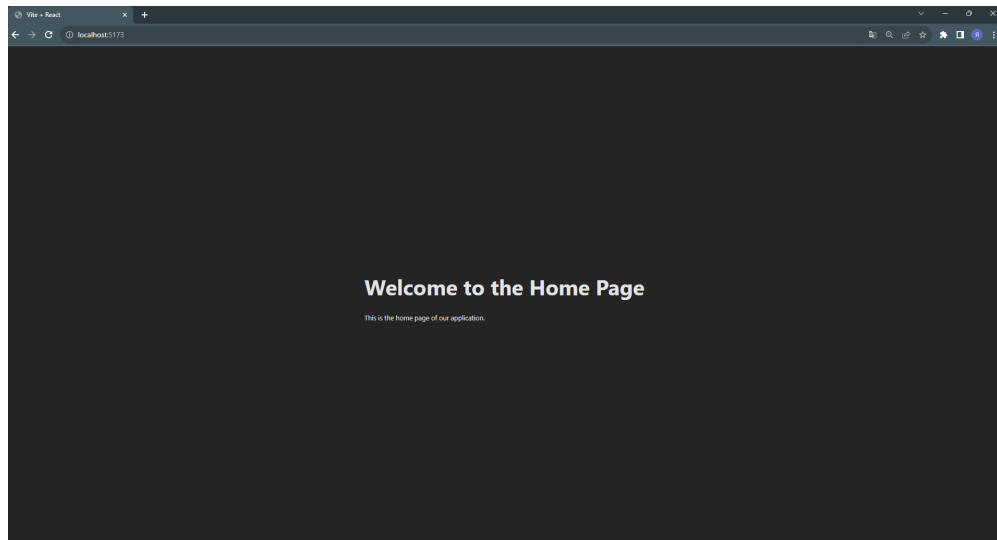


Рис 3.4. Результат програми

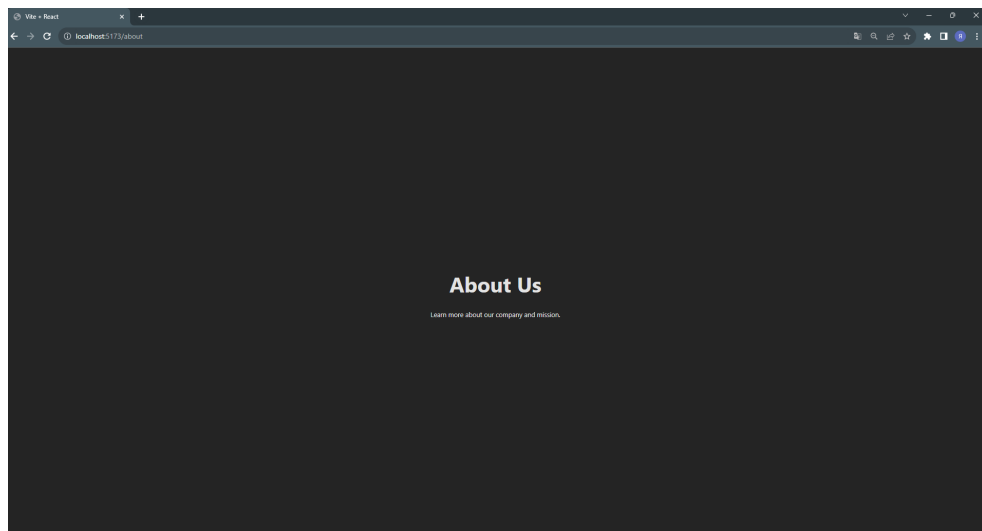


Рис 3.5. Результат програми

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:

```

lab5_vite > reducers > appReducer.jsx > ...
1  import { combineReducers } from 'redux';
2
3  const initialState = {
4    counter: 0,
5  };
6
7  function appReducer(state = initialState, action) {
8    switch (action.type) {
9      case 'INCREMENT':
10       return { ...state, counter: state.counter + 1 };
11      case 'DECREMENT':
12       return { ...state, counter: state.counter - 1 };
13      default:
14       return state;
15    }
16  }
17
18  const rootReducer = combineReducers({
19    app: appReducer,
20  });
21
22  export default rootReducer;

```

Рис 4.1 Код сторінки *appReducer.jsx*

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

```

lab5_vite > actions > appActions.jsx > ...
1  export const updateData = (data) => {
2    return {
3      type: 'COMPLETE',
4      data: data,
5    };
6  };
7
8  export const increment = () => {
9    return {
10     type: 'INCREMENT',
11   };
12 };
13
14 export const decrement = () => {
15   return {
16     type: 'DECREMENT',
17   };
18 };

```

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
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import { Provider } from 'react-redux';
4  import store from '../containers/store';
5  import App from './App.jsx';
6  import './index.css';
7
8  ReactDOM.createRoot(document.getElementById('root')).render(
9    <React.StrictMode>
10     <Provider store={store}>
11       <App />
12     </Provider>
13   </React.StrictMode>
14 );
```

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 > ...
1  import React from 'react';
2  import { Link } from 'react-router-dom';
3  import { useSelector, useDispatch } from 'react-redux';
4  import { increment, decrement } from '../actions/appActions';
5  import './Home.css';
6
7  function Home() {
8    const appState = useSelector((state) => state.app);
9    const dispatch = useDispatch();
10
11    const handleIncrement = () => {
12      dispatch(increment());
13    };
14
15    const handleDecrement = () => {
16      dispatch(decrement());
17    };
18
19    return (
20      <div className="home-container">
21        <h1>Welcome to the Home Page</h1>
22        <p>Current App State: {appState.counter}</p>
23        <div className="button-container">
24          <button onClick={handleIncrement} className="increment-button">Increment</button>
25          <button onClick={handleDecrement} className="decrement-button">Decrement</button>
26        </div>
27        <nav>
28          <ul className="nav-links">
29            <li>
30              <Link to="/about" className="nav-link">About</Link>
31            </li>
32            <li>
33              <Link to="/todo" className="nav-link">Todo</Link>
34            </li>
35          </ul>
36        </nav>
37      </div>
38    );
39  }
40
41  export default Home;

```

Рис 6.1 Код компонента Home.jsx

```

lab5_vite > components > About.jsx > ...
1  import React from 'react';
2  import { Link } from "react-router-dom";
3  import './Home.css';
4
5  function About() {
6    return (
7      <div className="about-page">
8        <h1>About Us</h1>
9        <p>
10          Welcome to our About page.
11        </p>
12        <p>
13          We are a fantastic team of developers dedicated to creating amazing web applications.
14        </p>
15        <p>
16          If you want to learn more about our work, feel free to explore our projects in the "Home" section.
17        </p>
18        <nav>
19          <ul className="nav-links">
20            <li>
21              <Link to="/" className="nav-link">Home</Link>
22            </li>
23            <li>
24              <Link to="/todo" className="nav-link">Todo</Link>
25            </li>
26          </ul>
27        </nav>
28      </div>
29    );
30  }
31
32  export default About;

```

Рис 6.2 Код компонента About.jsx

```

lab5_vite > components > @ Todo.jsx > ...
1  import React from 'react';
2  import { useReducer } from 'react';
3  import reducer from '../reducers/Todoreducer';
4  import { Link } from 'react-router-dom';
5  import './Todo.css';
6
7  const initialTodos = [
8    {
9      id: 1,
10     title: 'Todo 1',
11     complete: false,
12   },
13   {
14     id: 2,
15     title: 'Todo 2',
16     complete: false,
17   },
18 ];
19
20 function Todo() {
21   const [todos, dispatch] = useReducer(reducer, initialTodos);
22
23   const handleComplete = (todo) => {
24     dispatch({ type: 'COMPLETE', id: todo.id });
25   };
26
27   return (
28     <>
29       <div className="container">
30         <h1>ToDo</h1>
31         {todos.map((todo) => (
32           <div key={todo.id}>
33             <label>
34               <input
35                 type="checkbox"
36                 checked={todo.complete}
37                 onChange={() => handleComplete(todo)}
38               />
39               {todo.title}
40             </label>
41           </div>
42         ))}
43         <nav>
44           <ul className="nav-links">
45             <li>
46               <Link to="/" className="nav-link">Home</Link>
47             </li>
48             <li>
49               <Link to="/about" className="nav-link">About</Link>
50             </li>
51           </ul>
52         </nav>
53       </div>
54     </>
55   );
56 }
57
58 export default Todo;

```

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 перевірки:

Додаток працює на локальному сервері та був запущений та перевірений в браузері.

Додано стилізацію до свого проекту за допомогою CSS.

В результаті отримуємо:

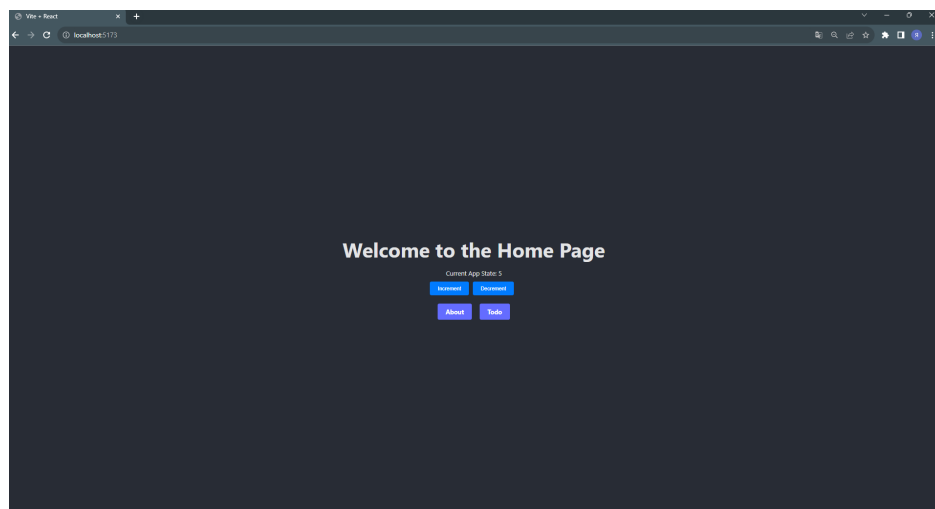


Рис 7.1 Результат запуску програми

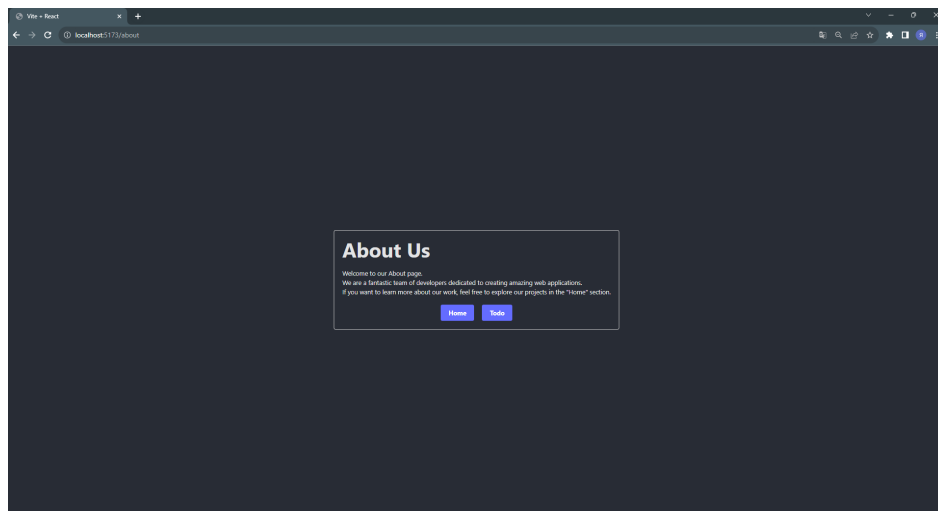


Рис 7.2 Результат запуску програми

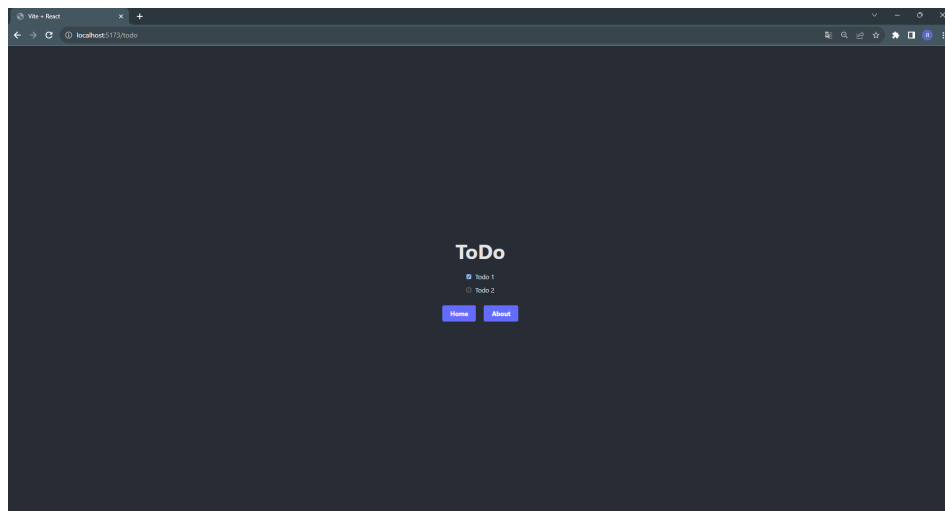


Рис 7.3 Результат запуску програми

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.