

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

ЗВІТ
Лабораторна робота №4
Використання REACT REDUCE

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. Створення компонентів:	4
3. Визначаємо дії (actions) та створюємо редуктиви (reducers):	8
4. Підключаємо Redux до компонентів та реалізуємо функціональності:	9
Висновок:	12

Опис

Тема: Використання react redux

Мета: Ознайомити студентів зі створенням додатків на React з використанням Redux та редукторів (reducers)

Завдання

Налаштування проекту:

1. Створіть новий проект React за допомогою Create React App або іншого інструменту на ваш вибір.
2. Встановіть бібліотеку Redux та react-redux для управління станом додатку.

Створення компонентів:

1. Створіть компонент TodoList, який буде відображати список завдань.
2. Створіть компонент TodoItem, який буде відображати окреме завдання.
3. Створіть компонент TodoForm, який буде відповідати за додавання нових завдань.

Визначення дій (actions) та Створення редукторів (reducers):

1. Створіть дії (actions) для додавання нового завдання, видалення завдання та позначення завдання як виконаного.
2. Створіть редуктор для обробки дій, пов'язаних з завданнями (наприклад, додавання, видалення, позначення як виконаного).
3. Підключіть цей редуктор до свого Redux-сторю.

Підключення Redux до компонентів та Реалізація функціональності:

1. Використовуйте бібліотеку react-redux для підключення компонентів TodoList, TodoItem, та TodoForm до Redux-сторю.
2. Доступ до стану і дій має бути забезпечений через властивості та дії компонентів.
3. Додайте можливість додавати нові завдання через TodoForm.
4. Дозвольте користувачам позначати завдання як виконані та видаляти завдання через TodoItem.
5. Оновлюйте список завдань після кожної дії.

Стилізація:

1. Застосуйте стилізацію за допомогою CSS або CSS-препроцесора за своїм вибором, щоб додаток виглядав привабливо.

Додаткова функціональність (необов'язково):

Додайте можливість редагування завдань.

Зберігайте стан додатку в локальному сховищі для збереження даних між сеансами.

Виконання

1. Створення нового проекту React:

Створюємо проект. Вводимо в терміналі команду `npm create vite@latest`:

```
PS D:\OA_3\lab4> npm init vite@latest
✓ Project name: ... lab4_vite
✓ Select a framework: » React
✓ Select a variant: » JavaScript

Scaffolding project in D:\OA_3\lab4\lab4_vite...

Done. Now run:

  cd lab4_vite
  npm install
  npm run dev
```

Рис 1.1 Виконання команди

```
PS D:\OA_3\lab4> cd lab4_vite
PS D:\OA_3\lab4\lab4_vite> npm install redux

added 271 packages, and audited 272 packages in 19s

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

found 0 vulnerabilities
PS D:\OA_3\lab4\lab4_vite> npm install react-redux

added 6 packages, and audited 278 packages in 4s

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

found 0 vulnerabilities
```

Рис 1.2 Підключення бібліотек

2. Створення компонентів:

ToDoList, який відображає список завдань:

```

lab4_vite > src > components > TodoList.jsx > ...
1  import React from 'react';
2  import TodoItem from './components/TodoItem';
3
4  const TodoList = () => {
5    const todos = [
6      { id: 1, title: '№1', completed: false },
7      { id: 2, title: '№2', completed: false },
8      { id: 3, title: '№3', completed: false },
9      { id: 4, title: '№4', completed: false },
10     { id: 5, title: '№5', completed: false },
11   ];
12
13   return (
14     <ul className='list-group'>
15       {todos.map((todo) => (
16         <TodoItem id={todo.id} title={todo.title} completed={todo.completed} />
17       ))}
18     </ul>
19   );
20 };
21
22 export default TodoList;

```

Рис 2. 1 Створення компонента TodoList.jsx

TodoItem, який відображає окреме завдання:

```

lab4_vite > src > components > TodoItem.jsx > ...
1  import React from 'react';
2
3  const TodoItem = ({ id, title, completed }) => {
4    return (
5      <li className={`list-group-item ${completed && 'list-group-item-success'}>
6        <div className='d-flex justify-content-between'>
7          <span className='d-flex align-items-center'>
8            <input type='checkbox' className='mr-3' checked={completed}></input>
9            {title}
10          </span>
11          <button className='btn btn-danger'><i class="bi bi-trash"></i> Delete</button>
12        </div>
13      </li>
14    );
15  };
16
17 export default TodoItem;

```

Рис 2. 2 Створення компонента TodoItem.jsx

TodoForm, який відповідає за додавання нових завдань:

```

lab4_vite > src > components > TodoForm.jsx > TodoForm
1  import React, { useState } from 'react';
2
3  const TodoForm = () => {
4    const [value, setValue] = useState('');
5
6    const onSubmit = (event) => {
7      event.preventDefault();
8      console.log('user entered: ' + value);
9    };
10
11    return (
12      <form onSubmit={onSubmit} className='form-inline mt-3 mb-3'>
13        <label className='sr-only'>Name</label>
14        <input
15          type='text'
16          className='form-control mb-2 mr-sm-2'
17          placeholder='write...'
18          value={value}
19          onChange={(event) => setValue(event.target.value)}
20        ></input>
21
22        <button type='submit' className='btn btn-primary mb-2'>
23          <i class="bi bi-plus-lg"></i> Submit
24        </button>
25      </form>
26    );
27  };
28
29  export default TodoForm;

```

Рис 2.3 Створення компонента *TodoForm.jsx*

TotalCompleteItems.jsx, який показує загальну кількість виконаних компонентів:

```

lab4_vite > src > components > TotalCompleteItems.jsx > ...
1  import React from 'react';
2
3  const TotalCompleteItems = () => {
4    return <h4 className='mt-3'>Total Complete Items: 5</h4>;
5  };
6
7  export default TotalCompleteItems;

```

Рис 2.4 Створення компонента *TotalCompleteItems.jsx*

App.jsx, який відповідає за виклик компонентів:

```

lab4_vite > src > App.jsx > ...
1  import React from 'react';
2  import 'bootstrap/dist/darkly/bootstrap.min.css';
3  import 'bootstrap-icons/font/bootstrap-icons.css';
4  import TodoList from './components/TodoList';
5  import TodoForm from './components/TodoForm';
6  import TotalCompleteItems from './components/TotalCompleteItems';
7
8  const App = () => {
9      return (
10         <div className='container bg-dark p-4 mt-5'>
11             <h1 className='text-center'>My Todo List</h1>
12             <TodoForm />
13             <TodoList />
14             <TotalCompleteItems />
15         </div>
16     );
17 };
18
19 export default App;

```

Рис 2.5 Створення компонента App.jsx

main.jsx, який який імпортує бібліотеки та компоненти і візуалізує додаток:

```

lab4_vite > src > main.jsx > ...
1  import React from 'react';
2  import { createRoot } from 'react-dom/client';
3  import './index.css';
4  import App from './App';
5  import store from './redux/store';
6  import { Provider } from 'react-redux';
7
8  const root = createRoot(document.getElementById('root'));
9
10 root.render(
11     <React.StrictMode>
12         <Provider store={store}>
13             <App />
14         </Provider>
15     </React.StrictMode>
16 );

```

Рис 2.6 Код компонента main.jsx

В результаті отримуємо:

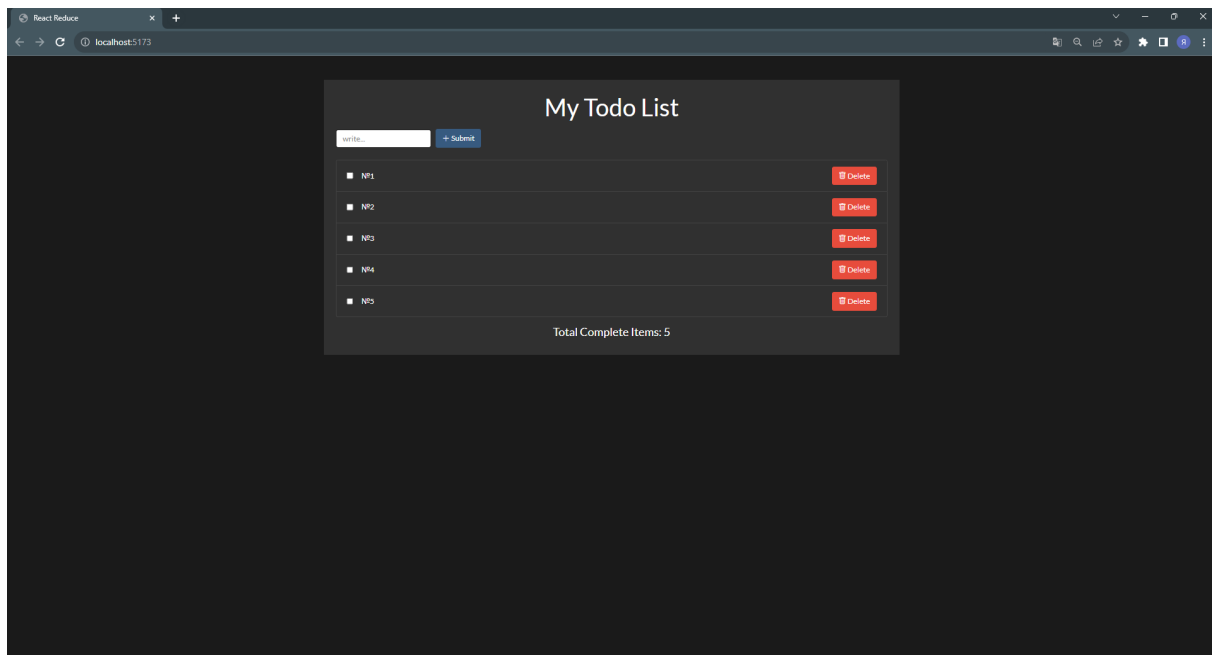


Рис 2.7. Результат програми

3. Визначаємо дії (actions) та створюємо редуктиви (reducers):

Створюємо редуктор для обробки дій, пов'язаних з завданнями (додавання, видалення, позначення як виконаного). У файлі *todoSlice.jsx* є три редуктори, які представляють три різні дії: *addTodo*, *toggleComplete* і *deleteTodo*. Кожен редуктор виконує відповідну дію, змінюючи стан згідно з параметрами *action*:

```
lab4_vite > src > redux > todoSlice.jsx > ...
1 import { createSlice } from '@reduxjs/toolkit';
2 import { nanoid } from 'nanoid';
3
4 export const todoSlice = createSlice({
5   name: 'todos',
6   initialState: [],
7   reducers: {
8     addTodo: (state, action) => {
9       const todo = {
10         id: nanoid(),
11         title: action.payload.title,
12         completed: false,
13       };
14       state.push(todo);
15     },
16     toggleComplete: (state, action) => {
17       const index = state.findIndex((todo) => todo.id === action.payload.id);
18       state[index].completed = action.payload.completed;
19     },
20     deleteTodo: (state, action) => {
21       return state.filter((todo) => todo.id !== action.payload.id);
22     },
23   },
24 });
25
26 export const { addTodo, toggleComplete, deleteTodo } = todoSlice.actions;
27
28 export default todoSlice.reducer;
```

Рис 3. 1 Код сторінки *todoSlice.jsx*

Підключаємо цей редуктор до нашого Redux-стору.

```
lab4_vite > src > redux > store.jsx > ...
1  import { configureStore } from '@reduxjs/toolkit';
2  import todoReducer from './todoSlice';
3
4  export default configureStore({
5    reducer: {
6      todos: todoReducer,
7    },
8  });
9
```

Рис 3.2 Код сторінки store.jsx

4. Підключаємо Redux до компонентів та реалізуємо функціональність:

Використали бібліотеку react-redux для підключення компонентів TodoList, TodoItem, та TodoForm до Redux-стору.

TodoList, який відображає список завдань та Оновлюйте список завдань після кожної дії:

```
lab4_vite > src > components > TodoList.jsx > ...
1  import React, { useEffect } from 'react';
2  import TodoItem from './components/TodoItem';
3  import { useSelector, useDispatch } from 'react-redux';
4
5  const TodoList = () => {
6    const dispatch = useDispatch();
7    const todos = useSelector((state) => state.todos);
8
9    useEffect(() => {
10     }, [dispatch]);
11
12    return (
13      <ul className='list-group'>
14        {todos.map((todo) => (
15          <TodoItem id={todo.id} title={todo.title} completed={todo.completed} />
16        ))}
17      </ul>
18    );
19  };
20
21  export default TodoList;
```

Рис 4.1 Код компонента TodoList.jsx

TodoItem, який дозволяє користувачам позначати завдання як виконані та видаляти їх:

```

lab4_vite > src > components > TodoItem.jsx > [x] TodoItem
4
5  const TodoItem = ({ id, title, completed }) => {
6    const dispatch = useDispatch();
7
8    const handleCheckboxClick = () => {
9      dispatch(toggleComplete({ id, completed: !completed }));
10   };
11
12   const handleDeleteClick = () => {
13     dispatch(deleteTodo({ id }));
14   };
15
16   return (
17     <li className={`list-group-item ${completed && 'list-group-item-dark'}>
18       <div className='d-flex justify-content-between'>
19         <span className='d-flex align-items-center'>
20           <input
21             type='checkbox'
22             className='mr-3'
23             checked={completed}
24             onChange={handleCheckboxClick}
25           />
26           {title}
27         </span>
28         <button onClick={handleDeleteClick} className='btn btn-danger'>
29           <i class="bi bi-trash"></i> Delete
30         </button>
31       </div>
32     </li>
33   );
34 };
35
36 export default TodoItem;
37

```

Рис 4. 2 Код сторінки *TodoItem.jsx*

TodoForm, який відповідає за додавання нових завдань:

```

lab4_vite > src > components > TodoForm.jsx > ...
5   const TodoForm = () => {
6     const [value, setValue] = useState('');
7     const dispatch = useDispatch();
8
9     const onSubmit = (event) => {
10      event.preventDefault();
11      if (value) {
12        dispatch(addTodo({ title: value }));
13        setValue('');
14      }
15    };
16
17    return (
18      <form onSubmit={onSubmit} className='form-inline mt-3 mb-3'>
19        <label className='sr-only'>Name</label>
20        <input
21          type='text'
22          className='form-control mb-2 mr-sm-2'
23          placeholder='Add todo...'
24          value={value}
25          onChange={(event) => setValue(event.target.value)}
26        />
27
28        <button type='submit' className='btn btn-primary mb-2'>
29          <i class="bi bi-plus-lg"></i> Submit
30        </button>
31      </form>
32    );
33  };
34
35  export default TodoForm;

```

Рис 4.3 Створення компонента *TodoForm*

TotalCompleteItems.jsx, який показує загальну кількість виконаних компонентів:

```

lab4_vite > src > components > TotalCompleteItems.jsx > ...
1   import React from 'react';
2   import { useSelector } from 'react-redux';
3
4   const TotalCompleteItems = () => {
5     const todos = useSelector((state) =>
6       state.todos.filter((todo) => todo.completed === true)
7     );
8
9     return <h4 className='mt-3 text-center'>Total complete items: {todos.length}</h4>;
10  };
11
12  export default TotalCompleteItems;

```

Рис 4.4 Створення компонента *TotalCompleteItems.jsx*

В результаті отримуємо:

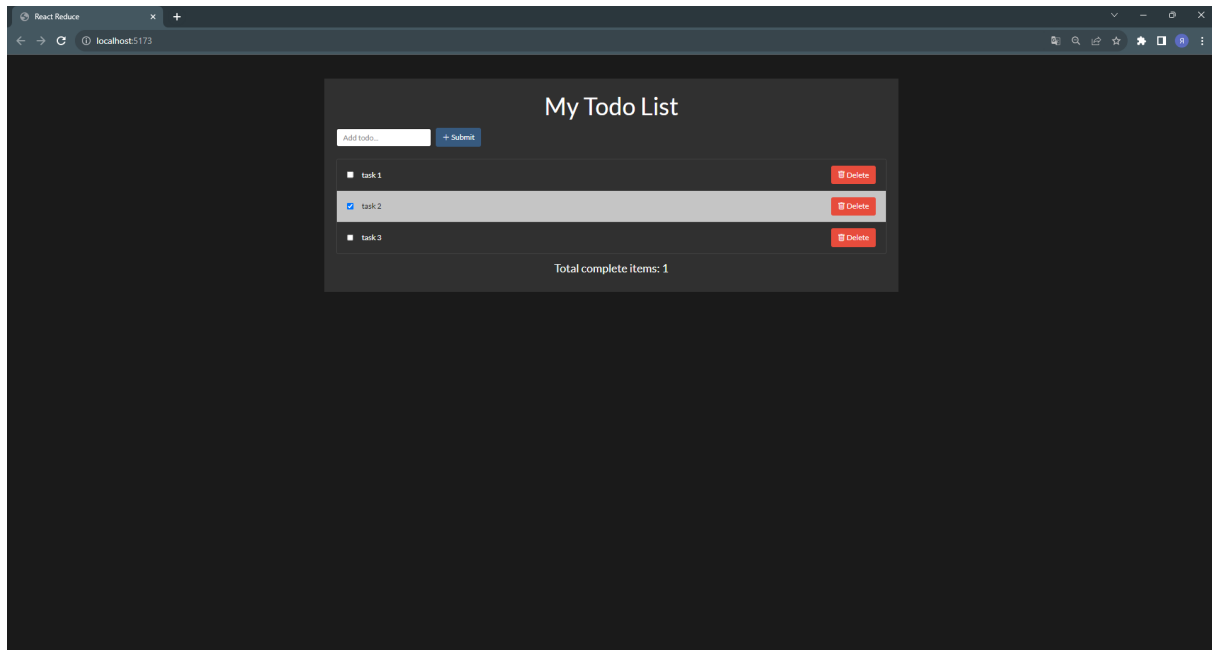


Рис 4.5. Результат програми

Висновок:

На цій лабораторній роботі ми успішно реалізували додаток на React, який використовує Redux для керування станом. Ми створили редуктори для обробки різних дій, таких як додавання, видалення та оновлення елементів.

Крім того, ми застосували стилізацію за допомогою CSS (а саме bootstrap) для кращого вигляду нашого додатка.