Задание

- Написать тесты для своего веб-приложения (клиентской его части)
- Тестирование: react-testing-libraly

AddTodoForm.tsx

```
import React from "react";
import { render, fireEvent } from "@testing-library/react";
import { AddTodoForm } from "./AddTodoForm";

describe("AddTodoForm", () => {
  it("renders input and button", () => {
    const { getByText, getByPlaceholderText } = render(<AddTodoForm addTodo={() => {}} />);
    expect(getByPlaceholderText("Enter new todo")).toBeInTheDocument();
    expect(getByText("Add Todo")).toBeInTheDocument();
  });

  it("allows users to input text", () => {
    const { getByPlaceholderText } = render(<AddTodoForm addTodo={() => {}} />);
    const input = getByPlaceholderText("Enter new todo") as HTMLInputElement;
    fireEvent.change(input, { target: { value: "Test todo" } });
    expect(input.value).toBe("Test todo");
  });

  it("calls addTodo function on form submission", () => {
    const addTodoMock = jest.fn();
    const { getByPlaceholderText, getByText } = render(<AddTodoForm addTodo={addTodoMock} />);
    const input = getByPlaceholderText("Enter new todo") as HTMLInputElement;
    const button = getByText("Add Todo");

    fireEvent.change(input, { target: { value: "Test todo" } });
    fireEvent.click(button);

    expect(addTodoMock).toHaveBeenCalledWith("Test todo");
  });

  it("clears input after form submission", () => {
    const addTodoMock = jest.fn();
    const { getByPlaceholderText, getByText } = render(<AddTodoForm addTodo={addTodoMock} />);
    const input = getByPlaceholderText("Enter new todo") as HTMLInputElement;
    const button = getByText("Add Todo");

    fireEvent.change(input, { target: { value: "Test todo" } });
    fireEvent.click(button);

    expect(input.value).toBe("");
  });
});
```

App.tsx

```
import React from "react";
import { render, fireEvent } from "@testing-library/react";
import App from "./App";

describe("App", () => {
  it("renders the initial todos", () => {
    const { getByText } = render(<App />);
    const todoTexts = ["Купить хлеб", "Помыть посуду", "Пойти в парк"];
    todoTexts.forEach(text => {
      expect(getByText(text)).toBeInTheDocument();
    });
  });

  it("allows adding a new todo", () => {
    const { getByPlaceholderText, getByText, queryByText } = render(<App />);
    const newTodoText = "Приготовить обед";
    const input = getByPlaceholderText("Enter new todo");
    const addButton = getByText("Add Todo");

    fireEvent.change(input, { target: { value: newTodoText } });
    fireEvent.click(addButton);

    expect(getByText(newTodoText)).toBeInTheDocument();
    expect(queryByText("")).toBeNull();
  });

  it("allows toggling a todo's completion status", () => {
    const { getByText } = render(<App />);
    const todoText = "Помыть посуду";
    const todoCheckbox = getByText(todoText).querySelector("input[type='checkbox']");

    fireEvent.click(todoCheckbox);

    expect(todoCheckbox).toBeChecked();
  });
});
```

TodoListItem.tsx

```
import React from "react";
import { render, fireEvent } from "@testing-library/react";
import { TodoListItem } from "./TodoListItem";

const mockToggleComplete = jest.fn();

const sampleTodo = {
  text: "Sample Todo",
  complete: false
};

describe("TodoListItem", () => {
  it("renders todo text correctly", () => {
    const { getByText } = render(
      <TodoListItem todo={sampleTodo} toggleComplete={mockToggleComplete} />
    );
    expect(getByText(sampleTodo.text)).toBeInTheDocument();
  });

  it("calls toggleComplete function when checkbox is clicked", () => {
    const { getByLabelText } = render(
      <TodoListItem todo={sampleTodo} toggleComplete={mockToggleComplete} />
    );
    const checkbox = getByLabelText(sampleTodo.text);
    fireEvent.click(checkbox);
    expect(mockToggleComplete).toHaveBeenCalledWith(sampleTodo);
  });

  it("renders todo as complete if complete property is true", () => {
    const completeTodo = { ...sampleTodo, complete: true };
    const { getByText } = render(
      <TodoListItem todo={completeTodo} toggleComplete={mockToggleComplete} />
    );
    const todoElement = getByText(completeTodo.text);
    expect(todoElement).toHaveClass("complete");
  });

  it("renders todo as incomplete if complete property is false", () => {
    const incompleteTodo = { ...sampleTodo, complete: false };
    const { getByText } = render(
      <TodoListItem todo={incompleteTodo} toggleComplete={mockToggleComplete} />
    );
    const todoElement = getByText(incompleteTodo.text);
    expect(todoElement).not.toHaveClass("complete");
  });
});
```

TodoList.tsx

```
import React from "react";
import { render, fireEvent } from "@testing-library/react";
import { TodoList } from "./TodoList";

const mockToggleComplete = jest.fn();

const sampleTodos = [
  { text: "Todo 1", complete: false },
  { text: "Todo 2", complete: true },
  { text: "Todo 3", complete: false }
];

describe("TodoList", () => {
  it("renders a list of todos", () => {
    const { getByText } = render(
      <TodoList todos={sampleTodos} toggleComplete={mockToggleComplete} />
    );
    sampleTodos.forEach(todo => {
      expect(getByText(todo.text)).toBeInTheDocument();
    });
  });

  it("calls toggleComplete function when a todo is clicked", () => {
    const { getByText } = render(
      <TodoList todos={sampleTodos} toggleComplete={mockToggleComplete} />
    );
    const todoText = sampleTodos[0].text;
    const todoElement = getByText(todoText);
    fireEvent.click(todoElement);
    expect(mockToggleComplete).toHaveBeenCalledWith(todoText);
  });
});
```