Department of Information Technology

UNIVERSITY INSTITUTE OF TECHNOLOGY HIMACHAL PRADESH UNIVERSITY, SHIMLA

Vocational-Training

Todo App Using React.js

Batch 2021-25

Submitted By: Project Guide:

Divay Sharma Dr. Balvir Singh Thakur

B.Tech. 5th Semester

Roll No.: *17212610031*

CERTIFICATE

This is to certify that the project entitled “Todo App using React.js” submitted in partial

fulfilment of the degree of B.Tech. INFORMATION TECHNOLOGY ENGINEERING (CSE) to the University Institute of Technology (UIT), HPU, Shimla, is an authentic and original work carried out by **Divay Sharma** under Roll no. **17212610031**.

The matter embodied in this project is genuine work done by the student and has not been submitted whether to this University or to any other University/Institute for the fulfilment of the requirements of any course of study.

Project Guide:

Dr. Balvir Singh Thakur

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who have supported and guided us throughout the completion of this project.

Firstly, we would like to thank our mentors and advisors, Dr. Balvir Singh Thakur Sir, for their invaluable guidance and support. Their expertise and insights were essential to the success of this project.

We would like to extend our special thanks to each other for our teamwork and dedication throughout the project. We acknowledge the valuable contributions each of us made, and appreciate the collaborative spirit that helped us overcome challenges and achieve our goals.

Finally, we would like to thank our friends and family for their encouragement and support. Their love and support gave us the strength and motivation to persevere and complete this project.

We are grateful for the opportunity to have worked on this project together and for the learning experience it has provided. We hope our findings will be of benefit to others who are interested in this field of research. Thank you.

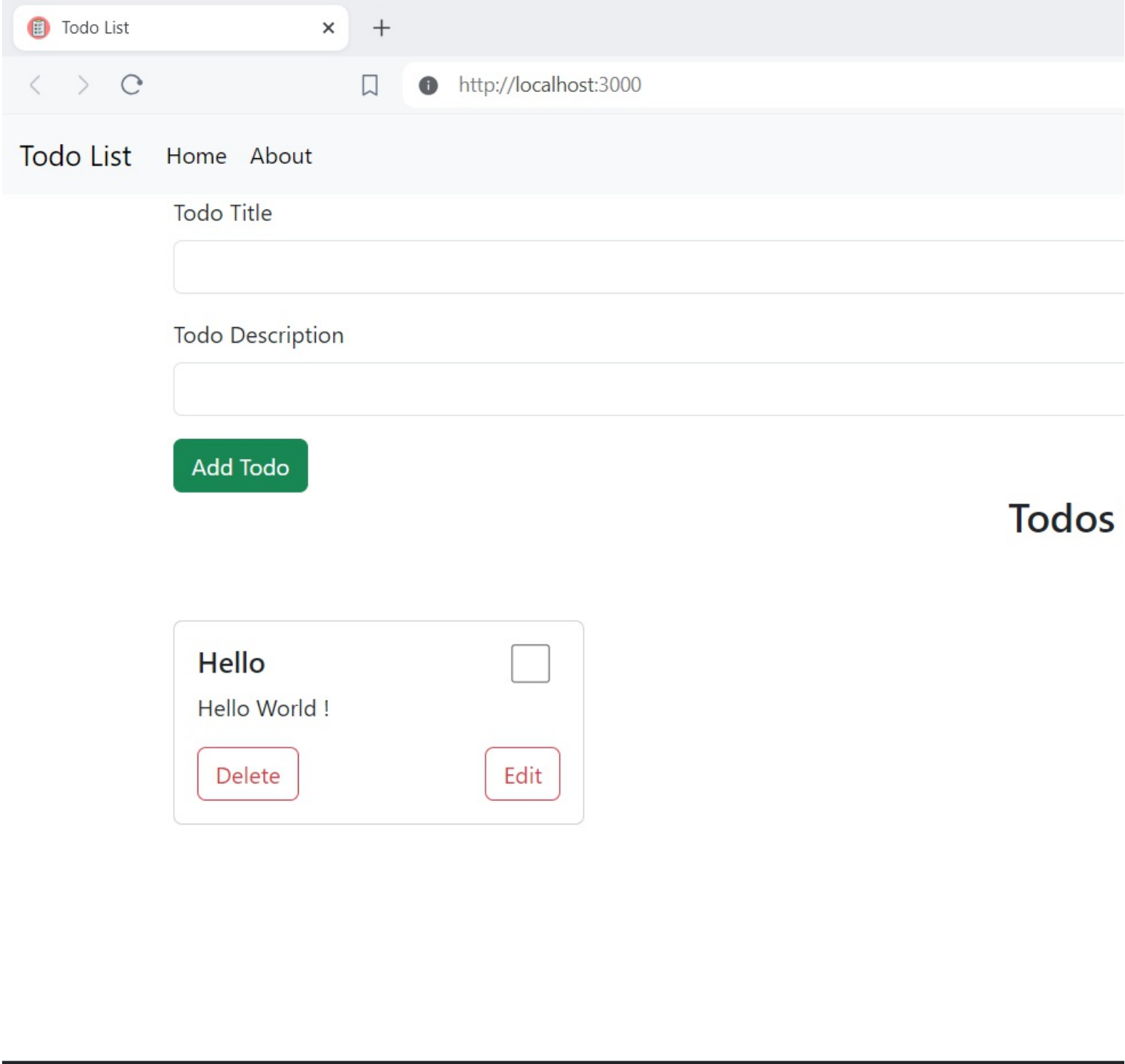
ABSTRACT

- App Component (App.js)
- Acts as the main component that sets up the application structure.
- Uses React Router (BrowserRouter, Routes, Route) for navigation.
- Includes a Header component for the application header.
- Includes a List component for displaying and managing todos.
- Includes a Footer component for the application footer.
- List Component (List.js)
- Manages the state of todos using the useState hook.
- Retrieves initial data from local storage using getData and sets it to the state using useEffect.
- Renders an AddTodo component for adding new todos.
- Renders a Todos component for displaying the list of todos based on the search criteria.
- AddTodo Component (AddTodo.js)
- Manages the state for the todo being edited using editItem.
- Allows adding new todos using the handletodos function.
- Handles editing existing todos using the handleEditValue function.
- Todos Component (Todos.js)
- Receives a list of todos and functions for handling deletion and editing.
- Maps through the todos array and renders individual TodoItems.
- TodoItems Component (TodoItems.js)
- Represents an individual todo item.
- Displays the title, description, and a checkbox for marking completion.
- Provides buttons for deleting and editing the todo.
- About Component (About.js)
- Presumably, this component represents information about the application or its creators. However, the code for this component is not provided.

TABLE OF CONTENT

Introduction

In summary, this code creates a basic Todo List application with the ability to add, edit, and delete tasks. The application uses React for the front-end and relies on local storage to persist todo data. Additionally, it implements React Router for navigation between different views.



App Component (App.js):

1. Dependencies and Imports:

- React, useState: Imported from 'react'.
- Footer: Imported from './myComponents/Footer'.
- BrowserRouter, Routes, Route: Imported from 'react-router-dom'.
- List, Header, About: Imported from corresponding files.

2. State:

- searchTodo: State variable managed by useState hook to store the search query for filtering todos.

3. Components:

- <Header>: Component rendered at the top of the page, responsible for displaying the header and accepting search input.

- <Routes> and <Route>: Components from React Router for defining the application routes.
- <List>: Component rendered based on the current route, responsible for managing the todo list.
- <About>: Component rendered when the '/about' route is active.
- <Footer>: Component rendered at the bottom of the page.

4. Functionality:

- Manages routing using React Router.
- Passes down searchTodo state to the List component.
- Provides a header with a search input field.
- Renders either the list of todos or the about page based on the route.

List Component (List.js):

1. Dependencies and Imports:

- React, useState, useEffect: Imported from 'react'.
- nanoid: Imported for generating unique IDs.
- AddTodo, Todos: Imported from corresponding files.

2. State:

- editItem: State variable to manage the todo item being edited.
- editId: State variable to store the ID of the todo item being edited.
- todos: State variable to store the list of todos.

3. Functions:

- handleonDelete: Function to delete a todo item.
- getData: Function to retrieve todo list data from local storage.
- handletodos: Function to add a new todo item to the list.
- handleEditItem: Function to set the ID of the todo item being edited.
- handleEditValue: Function to update the values of a todo item being edited.

4. Effect Hooks:

- Uses useEffect to initialize the todos state with data from local storage.
- Uses useEffect to update local storage whenever the todos state changes.

5. Component Structure:

- Renders the <AddTodo> component for adding new todos.
- Renders the <Todos> component for displaying the list of todos.

Todos Component (Todos.js):

1.Props:

- todos: Array of todo items to be displayed.
- handleonDelete: Function to delete a todo item.
- handleEditItem: Function to set the todo item being edited.
- editItem: Todo item being edited.

2.Component Structure:

- Maps over the todo items and renders individual <TodoItems> components for each todo.
- Handles editing and deletion of todo items.

TodoItems Component (TodoItems.js):

1. Props:

- todo: Individual todo item to be displayed.
- handleEditItem: Function to set the todo item being edited.
- handleonDelete: Function to delete a todo item.
- todos: Array of all todo items.

2. State:

- isChecked: State variable to manage the checked status of the todo item.

3.Functionality:

- Displays the title and description of the todo item.
- Provides checkboxes for marking todo items as checked.

- Allows users to delete and edit todo items.

Code Snippets

```
import Footer from "./myComponents/Footer";
import React ,{useState} from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import List from "./myComponents/List";
import Header from "./myComponents/Header"
import About from "./About"
function App() {
   const [searchTodo, setSearchTodo] = useState("");


  return (
    <Router>
      <div className="App">
        <Header handleSearchTodo={setSearchTodo}  />
        <Routes>
          <Route path="/" element={<List searchTodo={searchTodo} />} />
          <Route path="/about" element={<About />} />
        </Routes>
        <Footer />
      </div>
    </Router>
  );
}
export default App;
import React, { useState, useEffect } from 'react'
import { nanoid } from "nanoid";
import AddTodo from './AddTodo';
import Todos from './Todos';
function List({ searchTodo }) {
   const [editItem, setEditItem] = useState(null);
   const [editId, setEditId] = useState(null);
   const [todos, setTodos] = useState([]);




   useEffect(() => {
      setTodos(getData())
   }, [])
```

```
// delete button
const handleonDelete = (key) => {
    const newtodo = todos.filter((todo) => todo.id !== key);
    setTodos(newtodo);
};

//  get data from local storage
const getData = () => {
    let list = localStorage.getItem("todo-list");
    if (list) {
        return JSON.parse(localStorage.getItem("todo-list"));
    } else {
        return [];
    }
};
// add data to local variable
useEffect(() => {
    localStorage.setItem("todo-list", JSON.stringify(todos));
}, [todos]);
// what to add in todo
const handletodos = ({ title, description }) => {
    const newtodo = {
        id: nanoid(),
        title: title,
        description: description,



    };
    const newtodos = [...todos, newtodo];
    setTodos(newtodos);

};

// to find id
const handleEditItem = (id) => {
    setEditId(id)
};
// to change values
useEffect(() => {
```

```
    if (editId) {

        const intialObj = todos.filter((obj) => obj.id === editId);
        setEditItem(intialObj[0])
    }
}, [editId]);



const handleEditValue = ({ title, desc }) => {

    const index = todos.findIndex((obj) => obj.id === editId);
    const newtodo = {
        id: editId,
        title: title,
        description: desc ,
        checked : false

    }
    const newTodos = [...todos];
    newTodos.splice(index, 1, newtodo)
    setTodos(newTodos)
    setEditId(null)
    setEditItem(null)
}


return (
    <>
        <AddTodo
            handletodos={handletodos}
            setEditItem={setEditItem}
            editItem={editItem}
            handleEditValue={handleEditValue}

        />
        <Todos
            todos={todos.filter((todo) => todo.title.toLowerCase().includes(searchTodo.toLowerCase()))}
```

```
            handleonDelete={handleonDelete}
            handleEditItem={handleEditItem}
            editItem={editItem}
            setEditItem={setEditItem}
          />
      </>
    )
}
export default List
import React from 'react'
import TodoItems from './TodoItems'
const Todos = ({ todos, handleonDelete, handleEditItem, editItem}) => {
  return (
    <div className="container">
      <h3 className='text-center'>Todos List</h3>
      <div className="row">
        {todos.length === 0 ? "no todo to display" :
          todos.map((todo) => {
            return <TodoItems key={todo.id}
            todos={todos}
              todo={todo} handleonDelete={handleonDelete} handleEditItem={handleEditItem} editItem={editItem} />
          })
        }
      </div>
    </div>
  )
}
export default Todos
import React, { useState } from "react";
import "../App.css";
const TodoItems = ({ todos, todo, handleEditItem, handleonDelete }) => {
  const { title, id, description } = todo;
  const [isChecked, setIsChecked] = useState(false);

  const handleChecked = (e) => {
    setIsChecked(!isChecked);
```

```
  const index = todos.findIndex((obj) => obj.id === id);
  let updatedTodo = { ...todos[index] };
  updatedTodo["isChecked"] = !isChecked;
  const newTodos = [...todos];
  newTodos.splice(index, 1, updatedTodo);
  localStorage.setItem("todo-list", JSON.stringify(newTodos));

  // newTodos.splice(index, 1, newtodo)
};
return (
  <div className="col-sm-4 mt-5 todoDiv ">
    <div className="card" style={{ width: "18rem" }}>
      <div className="card-body">
        <input
          value={isChecked}
          type="checkBox"
          className="checkbox"
          onClick={handleChecked}
        />
        <h5 className="card-title " id={id}>
          {title}
        </h5>
        <p className="card-text">{description}</p>
        <div className="button_div">
          <button
            type="button"
            className="btn btn-outline-danger "
            onClick={() => {
              handleonDelete(id);
            }}
          >
            Delete
          </button>
          <button
            type="button"
            className="btn btn-outline-danger  "
            onClick={() => {
              handleEditItem(id);
            }}
```

```
            >
              Edit
            </button>
          </div>
        </div>
      </div>
    </div>
  );
};

export default TodoItems;
```

Conclusion

Your React Todo List application effectively manages todo items, allows users to add, edit, and delete items, and provides routing for navigating between different views. The use of React Router simplifies navigation, and the application's structure is modular and well-organized.