PROJECT REPORT

#1 TO- DO LIST APPLICATION(EASY)

#2 CONTACT MANAGER (INTERMEDIATE)

Submitted by: Manit Srivastava
Kiet Group of Colleges
Date of Submission: [29.10.2024]

1. Introduction

This combined project report covers the development of two web applications—a To-Do List and a Contact Manager. The To-Do List project, created using HTML, CSS, and JavaScript, is designed to help users manage their daily tasks efficiently, while the Contact Manager, built using React and CSS, is a more advanced application enabling users to manage their contact information. Each project serves as an opportunity to explore different front-end development skills, ranging from basic HTML and JavaScript to more complex React-based development.

Technologies Used:
- To-Do List: HTML, CSS, JavaScript
- Contact Manager: React, CSS

2. Project 1: To-Do List Application(EASY LEVEL)

2.1 Overview

The To-Do List application is a simple yet functional web-based project developed using HTML, CSS, and JavaScript. The primary objective of this application is to provide users with an intuitive tool to manage their daily tasks. Users can create new tasks, mark them as complete, edit them if necessary, and delete tasks that are no longer relevant. The application also incorporates local storage, allowing users to save their task lists in their browser and access them even after closing and reopening the application.

2.2 Requirements

Functional Requirements:
- Add, delete, edit, and mark tasks as complete or incomplete.
- Store data in local storage for persistence across sessions.

Non-Functional Requirements:
- Intuitive user interface with responsive design.
- Fast loading and optimal performance.

2.3 System Design

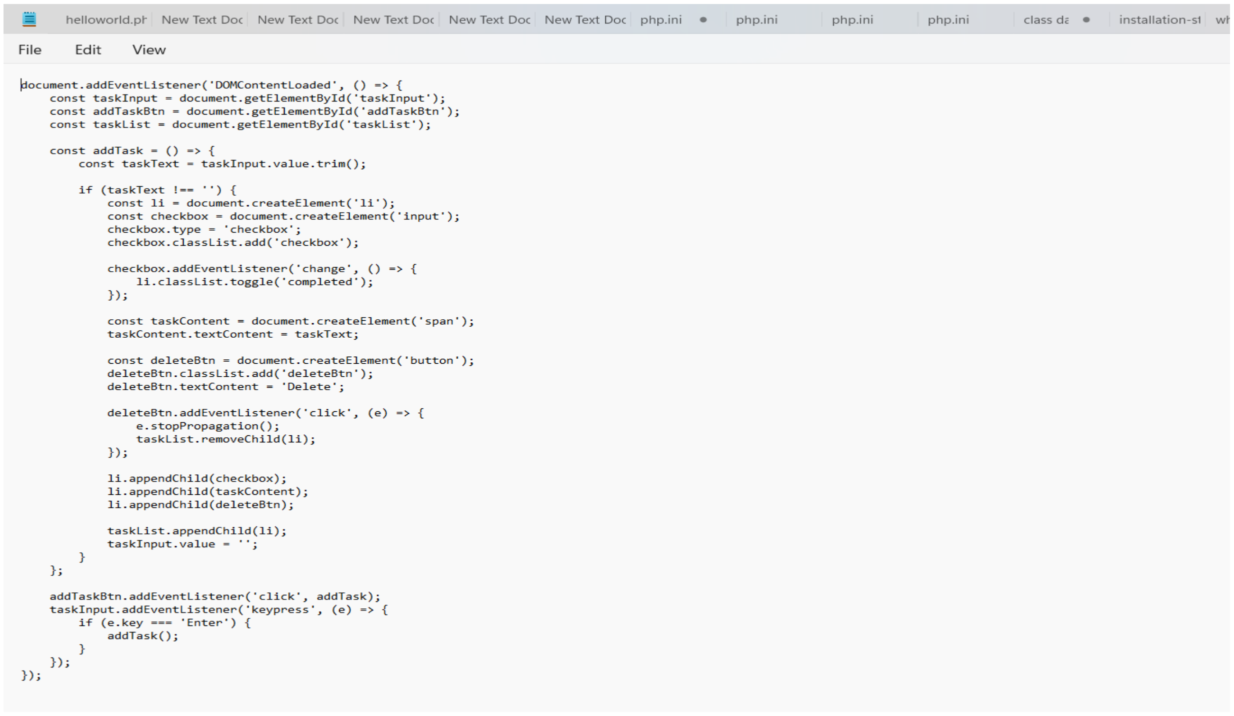
UI Design:
The user interface is simple and user-friendly, featuring input fields for tasks, buttons for adding and deleting tasks, and options to mark tasks as complete. Each task is displayed in a list format.

Data Flow:
Tasks are dynamically added to the DOM using JavaScript. Data persistence is handled using the browser's local storage, enabling data retention across sessions.

2.4 Implementation

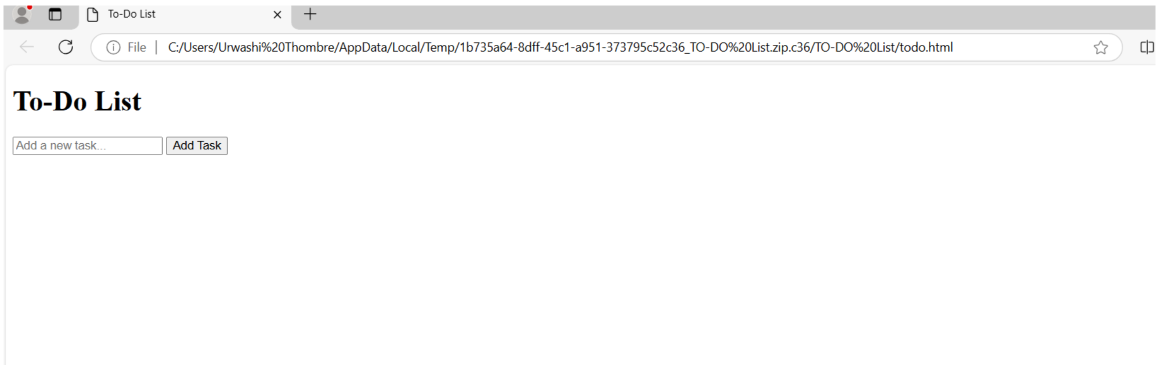
HTML: The HTML structure consists of input fields and buttons for managing tasks.
CSS: CSS is used to style the task list, buttons, and layout.
JavaScript: JavaScript provides the core functionality for adding, editing, and deleting tasks, as well as managing local storage to persist data.

2.4 CODE



```
document.addEventListener('DOMContentLoaded', () => {
    const taskInput = document.getElementById('taskInput');
    const addTaskBtn = document.getElementById('addTaskBtn');
    const taskList = document.getElementById('taskList');

    const addTask = () => {
        const taskText = taskInput.value.trim();

        if (taskText !== '') {
            const li = document.createElement('li');
            const checkbox = document.createElement('input');
            checkbox.type = 'checkbox';
            checkbox.classList.add('checkbox');

            checkbox.addEventListener('change', () => {
                li.classList.toggle('completed');
            });

            const taskContent = document.createElement('span');
            taskContent.textContent = taskText;

            const deleteBtn = document.createElement('button');
            deleteBtn.classList.add('deleteBtn');
            deleteBtn.textContent = 'Delete';

            deleteBtn.addEventListener('click', (e) => {
                e.stopPropagation();
                taskList.removeChild(li);
            });

            li.appendChild(checkbox);
            li.appendChild(taskContent);
            li.appendChild(deleteBtn);

            taskList.appendChild(li);
            taskInput.value = '';
        }
    };

    addTaskBtn.addEventListener('click', addTask);
    taskInput.addEventListener('keypress', (e) => {
        if (e.key === 'Enter') {
            addTask();
        }
    });
});
```

2.7 Output


To-Do List
File | C:/Users/Urwashi%20Thombre/AppData/Local/Temp/1b735a64-8dff-45c1-a951-373795c52c36_TO-DO%20List.zip.c36/TO-DO%20List/todo.html
To-Do List
Add a new task...
Add Task

2.5 Testing

Testing included verifying all functional requirements. Key test cases include adding a new task, editing a task, deleting a task, marking a task as complete, and confirming data persistence through local storage.

2.6 Challenges and Solutions

Challenges included ensuring data persistence and managing the UI state between updates. The local storage API provided a solution for data persistence, while JavaScript event listeners enabled real-time UI updates.

2.7 Conclusion

The To-Do List application is a simple yet functional web-based project developed using HTML, CSS, and JavaScript. The primary objective of this application is to provide users with an intuitive tool to manage their daily tasks. Users can create new tasks, mark them as complete, edit them if necessary, and delete tasks that are no longer relevant. The application also incorporates local storage, allowing users to save their task lists in their browser and access them even after closing and reopening the application.

3. Project 2: Contact Manager Application(Intermediate level)

3.1 Overview

The Contact Manager project is an intermediate-level application created with React, enabling users to create, edit, delete, and search contacts. This application is designed to provide an organized and efficient way of managing contact information.

3.2 Requirements

Functional Requirements:
- Add, view, edit, and delete contact entries.
- Search functionality to filter contacts.

Non-Functional Requirements:
- Responsive design compatible with various devices.
- Interactive and user-friendly UI.

3.3 System Design

UI Design:
The UI includes a search bar, a contact list, and a form for adding or editing contacts. Each contact is displayed with its details, and actions are available to update or delete entries.

Component Architecture:
The application is structured using React components such as ContactList, ContactForm, and SearchBar, with data flow managed through state and props.

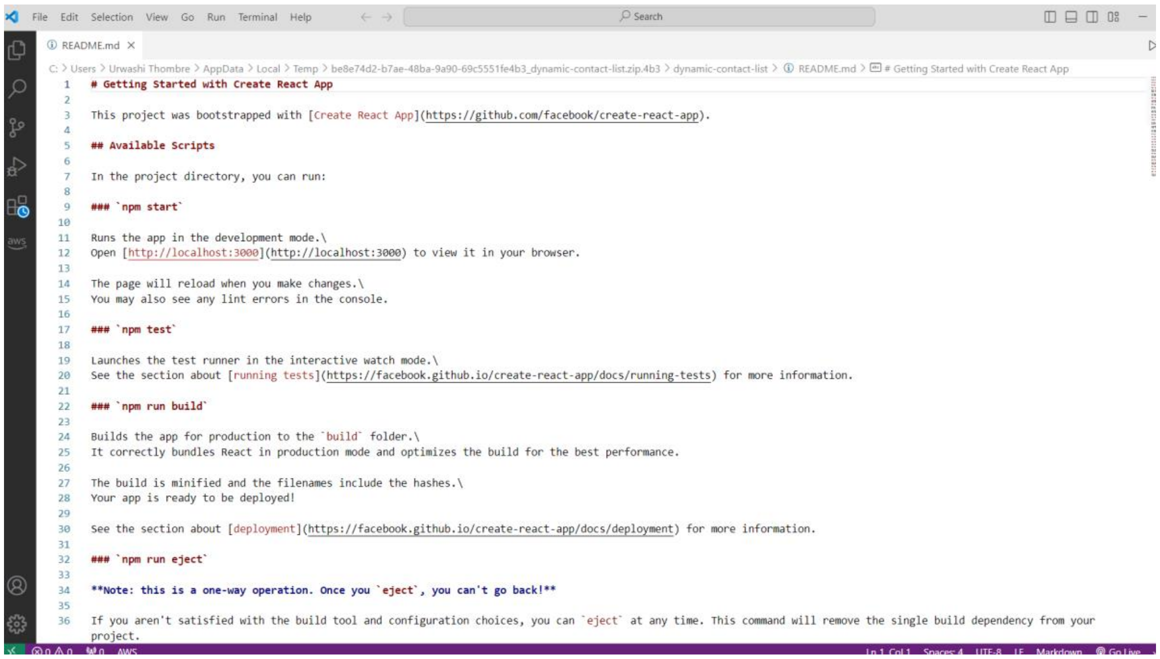
Data Flow:
React's state management is used to manage contact data and UI updates across components.

3.4 Implementation

React Components:
Each feature is encapsulated within a component, making the application modular and manageable. Key components include ContactList, ContactForm, and SearchBar.

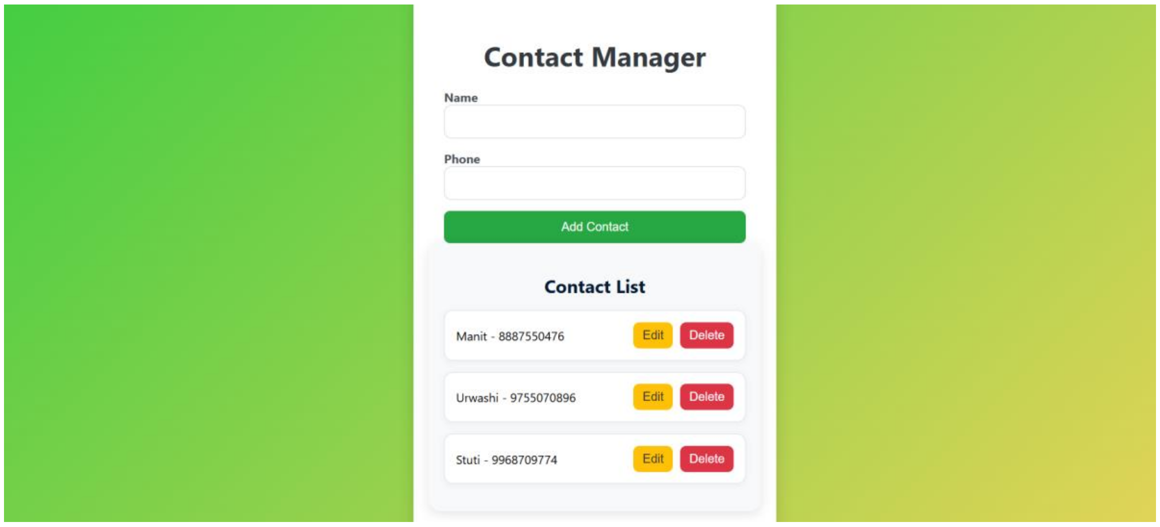
CSS Styling:
CSS styles are applied to achieve a responsive design and visually appealing layout for an interactive user experience.

3.5 CODE


File Edit Selection View Go Run Terminal Help
Search
README.md
C: > Users > Urwashi Thombre > AppData > Local > Temp > be8e74d2-b7ae-48ba-9a90-69c5551fe4b3_dynamic-contact-list.zip.4b3 > dynamic-contact-list > README.md > # Getting Started with Create React App

```
# Getting Started with Create React App

This project was bootstrapped with [Create React App](https://github.com/facebook/create-react-app).

## Available Scripts

In the project directory, you can run:

### `npm start`

Runs the app in the development mode.\
Open [http://localhost:3000](http://localhost:3000) to view it in your browser.

The page will reload when you make changes.\
You may also see any lint errors in the console.

### `npm test`

Launches the test runner in the interactive watch mode.\
See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more information.

### `npm run build`

Builds the app for production to the `build` folder.\
It correctly bundles React in production mode and optimizes the build for the best performance.

The build is minified and the filenames include the hashes.\
Your app is ready to be deployed!

See the section about [deployment](https://facebook.github.io/create-react-app/docs/deployment) for more information.

### `npm run eject`

**Note: this is a one-way operation. Once you `eject`, you can't go back!**

If you aren't satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your
project.
```

3.6 Output



Manit - 8887550476
Urwashi - 9755070896
Stuti - 9968709774

3.7 Testing

Testing involved validating CRUD operations and the search functionality. Test cases included creating, updating, deleting, and searching contacts to ensure smooth functionality and state management.

3.8 Challenges and Solutions

Challenges included managing state between components and ensuring data consistency across the application. React’s state and props allowed efficient handling of these challenges, enabling robust data flow and reactivity.

3.9 Conclusion

The Contact Manager project provided hands-on experience with React, especially in state management and component-based architecture. The project served as a valuable learning experience in creating scalable, intermediate-level applications.

4. Combined Conclusion

The development of the To-Do List application using HTML, CSS, and JavaScript, alongside the Contact Manager application built with React and CSS, showcases the versatility and power of modern web technologies in creating user-centric applications.

The To-Do List application exemplifies the fundamental principles of web development, where a simple yet effective interface allows users to manage tasks efficiently. By employing HTML for structure, CSS for styling, and JavaScript for dynamic interactivity, the application provides a straightforward user experience, emphasizing task management's importance in everyday life.

In contrast, the Contact Manager application, developed using React and CSS, illustrates the advantages of a component-based architecture. This application not only streamlines contact management but also enhances user interaction through advanced features and state management. The use of React enables efficient updates and re-renders, ensuring a smooth experience when adding, editing, or deleting contacts.

Together, these projects highlight the critical skills in front-end development, including UI design, user experience considerations, and efficient coding practices. They serve as foundational projects that can be further expanded to include advanced functionalities such as data persistence, user authentication, and integration with backend services. Overall, these applications provide a strong basis for future exploration and development in web technologies, emphasizing the importance of both functionality and design in creating successful web applications.

5. Future Scope

Future improvements for the To-Do List could include adding features like deadline reminders, priority levels, and cloud storage. For the Contact Manager, potential enhancements might include integration with external APIs, contact import/export, and additional search filters.Both applications have significant potential for further enhancement and scalability:

- Data Persistence: Integrating a backend service to store tasks and contacts would allow users to access their data across different devices. This could be achieved using RESTful APIs or technologies like Firebase, enabling data storage and retrieval.
- User Authentication: Implementing user authentication features would personalize the experience, allowing multiple users to securely manage their tasks and contacts. This can include OAuth integration or custom user management.
- For the To-Do List application, features like due dates, reminders, and categorization of tasks could enhance functionality.
- The Contact Manager could benefit from features like grouping contacts, importing/exporting contact lists, and adding additional information (e.g., addresses, social media links).

6. REFERENCES

- Flanagan, D. (2020). *JavaScript: The Definitive Guide*. O'Reilly Media.
  This book provides comprehensive coverage of JavaScript, including fundamental concepts and advanced features.

- Marcotte, E. (2011). *Responsive Web Design*. A Book Apart.
  A guide to creating responsive designs that work on various devices, which can help improve user experience.

- Hodges, S. (2018). *React: Up & Running: Building Web Applications in JavaScript*. O'Reilly Media.
  This book offers insights into building applications using React, covering both basic and advanced topics.