Name: Yash Sangwan

Roll.no. : 2300290120287

Year: 2nd

Objective : MLSA Internship PROJECT REPORT

Project Report: Simple To-Do List Web App

1. Introduction

This report documents a simple to-do list web application built using HTML, CSS, and JavaScript. The application allows users to create, manage, and mark tasks as completed.

2. Technologies Used

- **HTML: Defines the structure and content of the web page.**
- **CSS: Styles the visual presentation of the web page.**
- **JavaScript: Provides interactive functionality to the web page.**

3. Features

- **Add tasks: Users can enter a task description and add it to the list.**
- **Mark tasks as completed: Users can click on a task to mark it as completed, visually striking it through and lowering its opacity.**
- **Task persistence: The application utilizes local storage to save tasks, allowing them to persist even after the page is refreshed.**
- **Task deletion: Users can delete individual tasks using a dedicated button.**

4. Code Breakdown

- **HTML:**
    - **Defines the overall layout of the application with a container element.**
    - **Includes elements for the title, input field, button, and task list.**
    - **Links to the external CSS file.**
- **CSS:**
    - **Styles various aspects of the application, including:**
        - **Background colors and fonts**
        - **Input field and button appearance**
        - **Task list formatting**
        - **Visual cues for completed tasks**

        - **Delete button styling**

- **JavaScript:**
    - **Runs after the page loads (DOMContentLoaded event).**
    - **Retrieves references to HTML elements like input field, button, and task list.**
    - **Loads previously saved tasks from local storage (if any).**
    - **Defines functions for:**
        - **Rendering the task list based on stored tasks.**
        - **Adding new tasks with unique IDs and completion status.**
        - **Clearing the input field after adding a task.**
        - **Creating and styling individual list items for each task.**
        - **Marking tasks as completed on clicking the task itself.**
        - **Deleting tasks using the delete button and updating the list accordingly.**
        - **Saving the updated task list back to local storage.**

5. Conclusion

This simple to-do list web app demonstrates the power of combining HTML, CSS, and JavaScript to create functional and interactive web applications

Project Report: Weather App

1. Introduction

This report documents a weather application built using HTML, CSS, and JavaScript.

2. Technologies Used

- **HTML: Defines the structure and content of the web page.**
- **CSS: Styles the visual presentation of the web page.**
- **JavaScript: Provides interactive functionality and fetches data from an external API.**

3. Features

- **User Input: The user can enter a city name in the provided input field.**
- **Weather Data Retrieval: Upon clicking the "Get Weather" button, the application fetches weather data for the entered city using an OpenWeatherMap API.**
- **Weather Display: If the city is found, the application displays the retrieved weather information, including:**
    - **City name**

    - **Temperature (in Celsius)**
    - **Weather description**
    - **Feels-like temperature**
- **Error Handling: If the entered city is not found in the API response, the application displays an error message.**

4. Code Breakdown

- **HTML:**
    - **Defines the overall layout with a container element.**
    - **Includes elements for the title, input field, button, weather information section, and error message.**
    - **Links to the external CSS file.**
- **CSS:**
    - **Styles various aspects of the application, including:**
        - **Background colors and fonts**
        - **Input field and button appearance**
        - **Layout of weather information elements**
        - **Styling for the error message**
- **JavaScript:**
    - **Runs after the page loads (DOMContentLoaded event).**
    - **Retrieves references to HTML elements like input field, button, weather information elements, and error message.**
    - **Defines the API key used to access OpenWeatherMap data.**
    - **Defines functions for:**
        - **Getting user input from the city name field.**
        - **Handling the "Get Weather" button click event.**
            - **Trims leading and trailing spaces from the user input.**
            - **Checks for an empty input field.**
            - **Fetches weather data using the getWeatherData function (asynchronous).**
            - **Displays weather data or an error message based on the response.**
        - **Fetching weather data for a city (getWeatherData function - asynchronous).**

- **Builds a URL for the OpenWeatherMap API with the entered city and API key.**
- **Fetches data from the API using the fetch API.**
- **Checks for successful response (status code 200).**
- **Parses the JSON response and returns the data.**

- **Displaying weather information (displayWeatherData function).**
  - **Extracts relevant information from the weather data object (city name, temperature, weather description, feels like).**
  - **Updates the respective HTML elements with the retrieved data.**
  - **Shows the weather information section and hides the error message.**
- **Displaying an error message (displayError function).**
  - **Hides the weather information section and shows the error message.**

5. Conclusion

This weather application demonstrates the ability to fetch data from an external API, parse it, and display relevant information to the user. It incorporates basic error handling to provide feedback when a city is not found.