**BMI Calculator**

**BMI.html**

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>BMI Calculator</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<div class="container">

<h1>BMI Calculator</h1>

<div class="calculator">

<label for="height">Height (cm):</label>

<input type="number" id="height" placeholder="Enter your height">

<label for="weight">Weight (kg):</label>

<input type="number" id="weight" placeholder="Enter your weight">

<button onclick="calculateBMI()">Calculate BMI</button>

<p id="result"></p>

</div>

</div>

<script src="script.js"></script>

</body>

</html>

**script.js**

function calculateBMI() {

const height = document.getElementById('height').value;

const weight = document.getElementById('weight').value;

const resultElement = document.getElementById('result');

resultElement.classList.remove('underweight', 'normal', 'overweight', 'obese');

if (height === '' || weight === '') {

resultElement.innerText = "Please fill out all fields.";

return;

}

const heightInMeters = height / 100;

const bmi = (weight / (heightInMeters \* heightInMeters)).toFixed(2);

let resultText = `Your BMI is ${bmi}. `;

let categoryClass = '';

let adviceText = '';

if (bmi < 18.5) {

resultText += "You are underweight.";

categoryClass = 'underweight';

const minWeight = (18.5 \* heightInMeters \* heightInMeters).toFixed(2);

adviceText = `You need to gain at least ${(minWeight - weight).toFixed(2)} kg to reach a normal weight.`;

} else if (bmi >= 18.5 && bmi <= 24.9) {

resultText += "You have a normal weight.";

categoryClass = 'normal';

} else if (bmi >= 25 && bmi <= 29.9) {

resultText += "You are overweight.";

categoryClass = 'overweight';

const maxWeight = (24.9 \* heightInMeters \* heightInMeters).toFixed(2);

adviceText = `You need to lose at least ${(weight - maxWeight).toFixed(2)} kg to reach a normal weight.`;

} else {

resultText += "You are obese.";

categoryClass = 'obese';

const maxWeight = (24.9 \* heightInMeters \* heightInMeters).toFixed(2);

adviceText = `You need to lose at least ${(weight - maxWeight).toFixed(2)} kg to reach a normal weight.`;

}

resultElement.innerText = resultText + (adviceText ? ' ' + adviceText : '');

resultElement.classList.add(categoryClass);

}

**ToDoList**

**App.js**

import React from "react";

import Todolist from "./activity/Todolist";

import "./Todolist.css";

function App(){

return(

<div>

<Todolist/>

</div>

);

}

export default App

**Todolist.js**

import React, { useState } from "react";

function Todolist() {

const [todos, setTodos] = useState([]);

const [task, setTask] = useState("");

const [isEditing, setIsEditing] = useState(null);

const [editedTask, setEditedTask] = useState("");

const addTodo = () => {

if (task.trim()) {

const newTodo = {

task,

date: new Date().toLocaleString(),

};

setTodos([...todos, newTodo]);

setTask("");

}

};

const removeTodo = (index) => {

setTodos(todos.filter((\_, i) => i !== index));

};

const editTodo = (index) => {

setIsEditing(index);

setEditedTask(todos[index].task);

};

const saveEditTodo = (index) => {

const updatedTodos = todos.map((todo, i) =>

i === index

? { ...todo, task: editedTask, date: new Date().toLocaleString() }

: todo

);

setTodos(updatedTodos);

setIsEditing(null);

setEditedTask("");

};

return (

<div className="app">

<center><h1 className="todo-title">ToDo List</h1></center>

<div className="input-container">

<input

type="text"

value={task}

onChange={(e) => setTask(e.target.value)}

placeholder="Enter a task"

/>

<button onClick={addTodo}>Add</button>

</div>

{todos.length > 0 && (

<table className="todo-table">

<thead>

<tr>

<th>S.No</th>

<th>Items</th>

<th>Uploaded/Edited Date</th>

<th>Edit</th>

<th>Delete</th>

</tr>

</thead>

<tbody>

{todos.map((todo, index) => (

<tr key={index}>

<td>{index + 1}</td>

<td>

{isEditing === index ? (

<input

type="text"

value={editedTask}

onChange={(e) => setEditedTask(e.target.value)}

/>

) : (

<span>{todo.task}</span>

)}

</td>

<td>{todo.date}</td>

<td>

{isEditing === index ? (

<button onClick={() => saveEditTodo(index)} className="edit-button">

Save

</button>

) : (

<button onClick={() => editTodo(index)} className="edit-button">

<img src="https://cdn-icons-png.flaticon.com/128/2921/2921222.png" alt="Edit" className="edit-icon" />

</button>

)}

</td>

<td>

<button onClick={() => removeTodo(index)} className="delete-button">

<img src="https://cdn-icons-png.flaticon.com/128/2602/2602768.png" alt="Delete" className="trash-icon" />

</button>

</td>

</tr>

))}

</tbody>

</table>

)}

</div>

);

}

export default Todolist;

**Weather App**

**App.js**

import React from "react";

import Weather from "./activity/Weather";

import "./Weather.css";

function App(){

return(

<div>

<Weather/>

</div>

);

}

export default App

**Weather.js**

import React, { useState, useEffect, useCallback } from "react";

import axios from "axios";

const Weather = () => {

const [city, setCity] = useState("");

const [weatherData, setWeatherData] = useState(null);

const [error, setError] = useState("");

const [currentDateTime, setCurrentDateTime] = useState("");

const [loading, setLoading] = useState(false);

const API\_KEY = "086b2978d25b9357088087991f2a3d58";

useEffect(() => {

const formatDateTime = () => {

const now = new Date();

const options = {

year: "numeric",

month: "long",

day: "numeric",

hour: "2-digit",

minute: "2-digit",

second: "2-digit",

};

return now.toLocaleDateString(undefined, options);

};

setCurrentDateTime(formatDateTime());

const timer = setInterval(() => {

setCurrentDateTime(formatDateTime());

}, 1000);

return () => clearInterval(timer);

}, []);

const getWeather = useCallback(async () => {

if (!city) return;

setLoading(true);

try {

const response = await axios.get(

`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${API\_KEY}&units=metric`

);

setWeatherData(response.data);

setError("");

} catch (error) {

setError("City not found. Please enter a valid city.");

setWeatherData(null);

} finally {

setLoading(false);

}

}, [city, API\_KEY]);

useEffect(() => {

if (weatherData && city) {

const interval = setInterval(() => {

getWeather();

}, 60000);

return () => clearInterval(interval);

}

}, [weatherData, city, getWeather]);

const handleSubmit = (e) => {

e.preventDefault();

getWeather();

};

return (

<div className="weather-container">

<h1 className="title">Weather App</h1>

<form onSubmit={handleSubmit}>

<center>

<input

type="text"

value={city}

onChange={(e) => setCity(e.target.value)}

placeholder="Enter city"

/>

</center>

<button type="submit">Get Weather</button>

</form>

{loading && <p>Loading...</p>}

{error && <p className="error">{error}</p>}

{weatherData && (

<div className="weather-info">

<h2>{weatherData.name}</h2>

<p>Temperature: {weatherData.main.temp}°C</p>

<p>Date & Time: {currentDateTime}</p>

</div>

)}

</div>

);

};

export default Weather;