## Task 1: Simulated Weather Checker

**Objective**: Build an app that provides random weather information for a city entered by the user, with a simulated delay.

**Requirements:**

1. **UI:** An input field for the city name, a button labeled "Get Weather", and a display area for the weather information.
2. **Functionality**:

* When the user enters a city and clicks "Get Weather," a function (getWeather) should simulate an API call with a delay (e.g., 2 seconds) using setTimeout.
* The function should return a promise that resolves with random weather information, like a temperature between 15°C and 30°C, and a random weather condition (sunny, cloudy, rainy).
* After the delay, display the simulated weather information on the screen.

1. **Error Handling**: Randomly simulate errors (e.g., 1 in 5 times) to practice error handling with promises.

**Learning Goals:**

* Using promises and setTimeout to simulate asynchronous operations.
* Handling user input, promises, and errors.
* Basic DOM manipulation to display data.

# Task 2: Random Quote Generator with Local Data

**Objective**: Create an app that displays a random motivational quote when the user clicks a button.

**Requirements**:

1. **UI**: A button labeled "Get Quote" and a display area for the quote.
2. **Functionality**:
   * Define an array of quotes.
   * When the user clicks the "Get Quote" button, a function (getQuote) should return a promise that resolves after a delay (e.g., 1 second) with a randomly selected quote from the array.
   * Display the selected quote in the designated area.
3. **Bonus**: Add an option to mark quotes as favorites and display them separately.

**Learning Goals**:

* Using setTimeout to simulate an asynchronous delay with promises.
* Practicing random selection from local data and promise resolution.
* Basic DOM manipulation for dynamic content updates.

# Task 3: To-Do List with Delayed Task Loading

**Objective**: Create a basic to-do list app with simulated loading of initial tasks.

**Requirements**:

1. **UI**: An input field to add a new task, an "Add Task" button, and a list to display tasks.
2. **Functionality**:
   * Create a function fetchTasks that simulates loading initial tasks from a “server” by returning a promise that resolves with a list of tasks after a delay (e.g., 2 seconds).
   * When the page loads, call fetchTasks to populate the task list with initial data (e.g., "Buy groceries", "Walk the dog").
   * Allow users to add new tasks to the list using the input field. New tasks should be appended to the list immediately.
3. **Bonus**: Allow users to mark tasks as completed by clicking on them, with completed tasks styled differently.

**Learning Goals**:

* Practicing promise-based functions with simulated data loading.
* Using DOM manipulation for dynamic lists.
* Basic CRUD operations (Create, Read) in JavaScript.