

## **Programs to practice in the Lab**

**Topics Covered:** Promise and Async / Await.

Reading Data from APIs - Weather Data

---

**Question 1:** Write a function that returns a promise, and the promise should be rejected with an error if it takes longer than a specified timeout duration (e.g., 3 seconds).

**Question 2:** Design and implement a JavaScript program that simulates the order lifecycle of an online food delivery application using callbacks first and then change it to Promises followed by Async / Await.

### **System Requirements**

The system must perform the following steps sequentially using callbacks:

```
Place Order
    Accept customerName, restaurantName, and orderItems
    Simulate order placement delay (2 seconds)
Validate Order
    Check whether:
        At least one item is ordered
        Restaurant is open ("OpenKitchen" is open, others are closed)
        If validation fails, stop the process and return an error
Prepare Food
    Simulate food preparation time (3 seconds)
Assign Delivery Partner
    Assign a random delivery partner ID
    Simulate delay (1 second)
Deliver Order
    Display the estimated delivery message after 2 seconds
```

### **Execution Flow:**

```
Placing order...
Order placed successfully
Validating order...
Order validated
Food is being prepared...
Food prepared
Assigning delivery partner...
Delivery partner assigned: DP102
Out for delivery...
Order delivered successfully to Suhail
```

**Use callback error-first pattern: `callback(error, result)`**

**Working of Promise Chaining**

**Usage of Async function with Await**

**Question 3:** Develop a web page using HTML, CSS, and JavaScript that fetches user data from the JSONPlaceholder API and displays it in a tabular format.

API URL: <https://jsonplaceholder.typicode.com/users>

Use the Fetch API to retrieve data. Display the following details for each user: Name, Username, Email, City, Company Name. Show a loading message while data is being fetched. Handle errors such as network failure gracefully.

**Question 4:** Create a weather application that allows users to enter a **city name** and displays the **current weather details** using the OpenWeatherMap API.

API URL:

[https://api.openweathermap.org/data/2.5/weather?q={CITY\\_NAME}&appid={API\\_KEY}&units=metric](https://api.openweathermap.org/data/2.5/weather?q={CITY_NAME}&appid={API_KEY}&units=metric)

Display: Temperature (°C), Weather condition (Clear, Rain, etc.), Humidity, Wind speed. Use async/await for API calls and display an appropriate error message for invalid city names.

**Question 5:** Develop a web application that fetches **5-day weather forecast data** from OpenWeatherMap and visualizes the **temperature trend** using **Chart.js**.

API URL:

[https://api.openweathermap.org/data/2.5/forecast?q={CITY\\_NAME}&appid={API\\_KEY}&units=metric](https://api.openweathermap.org/data/2.5/forecast?q={CITY_NAME}&appid={API_KEY}&units=metric)

Fetch forecast data at 3-hour intervals, extract date and temperature values, Plot a **bar chart**, **line chart** showing temperature vs time. Ensure the chart updates dynamically based on the city entered.

**Question 6:** Create a web page that fetches and displays all public repositories of a given GitHub user.

API URL: <https://api.github.com/users/{USERNAME}/repos>

Display: Repository Name, Description, Programming Language, Repository URL. **Repositories should be displayed as cards or a list and Handle API errors properly.**

####@##@##