Software Developer Course Assessment Quantitative Assessment Practice

Course Name: Programming with JavaScript

Current Week: (2025/7/21)

Introduction:

The purpose of this assessment is to help us understand how the class is doing in terms of the review material that we have covered during the previous couple of weeks. The only purpose of this assessment is for us to improve our approach to review and ensure that what we're currently doing is an effective strategy. Completion of this assessment is mandatory - if you don't submit a solution, it will be marked as incomplete. If you do submit a solution, it will be marked as complete, as you will receive full marks.

Again, the goal here is to help you all in the best way that we can, so please do be honest when answering the questions related to how long it took, which resources you used, etc. And please ensure that you do your own work – don't just copy off a friend to get it done, earnestly do your best with it. If you can't get it completely working, give us what you have. While it will be graded, the grade will not count against you, it's just a way for us to see where everybody is, and to know which concepts, if any, we, as a class, may be struggling with.

Deadline: You will have until the end of the day on **Thursday July 31,2025 (4:00pm)** to submit your assessment solutions. Please ensure you answer all the questions outlined in the instructions portion of this document as well in your submission.

Instructions: Your name: ____________________

You are allowed to complete the assessment problems below in whatever way you can but please answer the following questions/points as part of your submission:

1. How many hours did it take you to complete this assessment? (Please keep try to keep track of how many hours you have spent working on each individual part of this assessment as best you can - an estimation is fine; we just want a rough idea.)

   Answer: [your answers should be formatted like this]...

2. What online resources do you use? (My lectures, YouTube, Stack Overflow etc.)
3. Did you need to ask any of your friends in solving the problems. (If yes, please mention name of the friend. They must be amongst your classmates.)

4. Did you need to ask questions to any of your instructors? If so, how many questions did you ask (or how many help sessions did you require)?
5. Rate (subjectively) the difficulty of Making this all! from your own perspective, and whether you feel confident that you can solve a similar but different problem requiring some of the same techniques in the future now that you've completed this one.

Objective:

In this assignment, you will build a **React application** that retrieves country data from a **public API**, filters it based on the **starting letter** of neighboring countries, and displays the data in a structured format.

This assignment is **similar to one of the practice questions,** but with modifications specific to a **React-based implementation**. Moreover, no UI screen shots are given here. Use your creative imagination and come up with a visually pleasant design.

API Endpoint:

You will fetch country data from the following API:
🔗 **URL:** https://restcountries.com/v3.1/all

Carefully analyze the API response to extract the required details for each country.

Tasks & Requirements

1. Fetch Country Data

- Retrieve country data from the given API endpoint.
- Extract and store the following details from each country's response:
    - **Official Name**
    - **Capital Name**
    - **Flag Image**
    - **List of Neighboring Countries** (Bordering countries)

---

2. App Component Structure

The main App component should contain the following elements:

- **A heading:** "Neighboring Countries"
- **Two buttons:**
    - "NEIGHBORS STARTING WITH A"
    - "NEIGHBORS STARTING WITH I"

Each button will trigger the filtering of country data based on its neighbors' names.

---

3. Filtering & Displaying Data

When a user clicks on a button, the app should:

- **"NEIGHBORS STARTING WITH A"**
    - Display all countries that have **at least one neighboring country** whose name starts with the letter **"A"**.
- **"NEIGHBORS STARTING WITH I"**
    - Display all countries that have **at least one neighboring country** whose name starts with the letter **"I"**.

Data Presentation:
Each country should be displayed **neatly** and **clearly structured**, showing:

1. **Official Country Name**
2. **Capital Name**
3. **Flag Image**
4. **List of Neighboring Countries**

---

4. Data Fetching Methods

You may use any of the following methods to fetch API data:

- XMLHttpRequest
- Fetch API with .then()
- Async/Await (Recommended for better readability)

Make sure to **store the fetched data in a React state variable** so that it can be dynamically passed to components.

---

5. Component Structure

Your app should have **three components**:

(a) App.jsx (Main Component)

- Displays the **title** and **two buttons**.
- Fetches and stores country data.
- Calls respective components on button clicks.
- Follow class styling methods here to style this page.

(b) NeighborsA.jsx (For Neighbors Starting with "A")

- **Receives filtered country data** (countries having at least one neighbor starting with "A").
- Displays relevant information in a well-structured format. (Follow class styling methods here too)

(c) NeighborsI.jsx (For Neighbors Starting with "I")

- **Receives filtered country data** (countries having at least one neighbor starting with "I").
- Displays relevant information in a well-structured format. (Follow class styling methods here too)

---

6. Additional Requirements

1. Use **React state** and **event handlers** to manage and display data dynamically.
2. Ensure **no page refresh** occurs when clicking buttons (use React state).
3. Write **clean, modular, and well-commented** code.
4. The UI should be **visually appealing and responsive** by using the class discussion techniques.

Submission Instructions

Ensure the following before submission:

a) App.jsx fetches data and manages state properly.
b) NeighborsA.jsx and NeighborsI.jsx components are correctly implemented.
c) Buttons filter and display data dynamically without refreshing the page.
d) Code follows our followed practices in class and is well-commented.