**Assignment Title: Building a Language Switcher with React's useContext Hook**

**Objective**:

To develop a multi-language switcher in a React application using the **useContext** hook, enhancing understanding of state management and internationalization in web applications.

Scenario:

You are developing a website that needs to support multiple languages (e.g., English, Spanish, French). The website has multiple components, and the language selection should be globally accessible to all components without prop drilling.

Target Skills:

* Understanding and implementing Context API and **useContext** hook in React.
* Building a scalable structure for internationalization in React apps.
* Enhancing component reusability and state management practices.

Assignment Details:

1. **Project Setup**:
   * Initialize a new React application.
   * Create a simple layout with at least three different components (e.g., Header, Content, Footer).
2. **Language Data Setup**:
   * Prepare language data files or objects for at least three languages (e.g., **en.json**, **es.json**, **fr.json**).
   * Each file should contain key-value pairs for various UI elements.
3. **Creating a Language Context**:
   * Use **React.createContext()** to create a Language Context.
   * Explain the structure and purpose of this Context in your project.
4. **Language Provider Component**:
   * Develop a Language Provider component that encapsulates the state and function to change the language.
   * This component should read and switch between different language data files based on user selection.
5. **Implementing useContext for Language Switching**:
   * In your main App component and other child components, use the **useContext** hook to access and change the current language.
   * Ensure that language changes reflect across all components dynamically.
6. **User Interface for Language Selection**:
   * Implement a language selection control (e.g., dropdown or buttons) that allows users to switch languages.
   * This control should update the global language state.
7. **Assignment Deliverables**:
   * **Source Code**: Submit via GitHub repository.
   * **Documentation**:
     + **README.md** in the repository with setup instructions.
     + A report explaining:
       - How **useContext** was used for managing language state.
       - Challenges faced during development and their solutions.
       - Benefits of using **useContext** in this scenario.
8. **Evaluation Criteria**:
   * Correct implementation of the Context API and **useContext** hook.
   * Functionality of the language switcher.
   * Code quality and documentation.

Submission Instructions:

* GitHub repository link for the project.
* A detailed PDF report discussing the implementation and learning outcomes.

Deadline:

[tomorrow ☺]

Additional Challenge (Optional):

* Add support for right-to-left languages and demonstrate the UI's adaptability to these changes.

This assignment provides a practical scenario for understanding the use of React's **useContext** hook in a real-world application, focusing on the development of a feature common in modern web applications: multi-language support.

**For the assignment of building a language switcher in a React application using the useContext hook, you will need to create several components and additional elements. Here's a detailed breakdown:**

**1. Language Data Files**

* **Purpose**: Store key-value pairs for UI text in different languages (e.g., English, Spanish, French).
* **Structure**: JSON files or JavaScript objects (e.g., **en.json**, **es.json**, **fr.json**) containing translations for each piece of text displayed on the UI.

**2. Language Context**

* **Creation**: Use **React.createContext()** to create a new context, named **LanguageContext**.
* **Content**: This context should hold the current language state and a function to update this state.

**3. Language Provider Component**

* **Functionality**: Manages the language state and provides a function to change the language.
* **State Management**: Holds the state for the current language and the method to update it (e.g., **setCurrentLanguage**).
* **Data Loading**: Responsible for loading the appropriate language data based on the current language state.

**4. Main App Component**

* **Role**: The root component that wraps all other components.
* **Context Integration**: Wraps the entire application or relevant parts within the **LanguageProvider** to provide language data to all child components.

**5. Language Selector UI Component**

* **Type**: A dropdown menu or button group for language selection.
* **Behavior**: On selection, it should update the global language state via the context.

**6. Individual UI Components**

* **Examples**: Header, Content, Footer.
* **Context Consumption**: Each component should use the **useContext** hook to access the current language and the associated translations.
* **Dynamic Text Rendering**: The text displayed in these components should change based on the current language.

**7. Additional Components (Optional)**

* **Right-to-Left (RTL) Layout Component**: If implementing RTL support, create a component to handle layout changes.
* **Theme Component**: For advanced scenarios, integrate a theme manager using the same context pattern.

**Key Implementation Details:**

* **Context API**: Central to managing and distributing the application's language state.
* **useContext Hook**: Used in individual components to access and set the language state.
* **State Management**: The Language Provider should encapsulate the logic for changing languages and loading the respective language data.
* **Modularity and Reusability**: Design components to be reusable and independent, relying on the context for data rather than props drilling.

By structuring your application with these components, you will create a scalable and maintainable solution for language switching in React apps, demonstrating an effective use of the Context API and the **useContext** hook.