Implementing Asynchronous API Calls in Blazor Applications

Objective: by the end of this activity, you will be able to manage asynchronous API calls in a Blazor application, display loading states, handle errors gracefully, and display fetched data in a table.

Step 1: Configure a Blazor Application in Visual Studio Code

Set up the Blazor application as the foundation for this activity.

Instructions:

1. Open Visual Studio Code and navigate to a folder where you want to set up your projects.
2. Open the terminal in VS Code using Ctrl + ~.
3. Create a new front-end project by running: dotnet new blazor -n AsyncApi This creates a Blazor WebAssembly project in a folder named "AsyncAPI."
4. Open both folders in separate VS Code windows using: code AsyncAPI
5. Open the Program.cs file and Register HttpClient for API calls by adding: builder.Services.AddHttpClient();
6. Run the application to confirm it works: dotnet run This command serves the application and opens it in a browser. Verify that the default Blazor app loads correctly.
7. Create a new Razor component named FetchUsers.razor in the Pages folder.

Step 2: Define the User Class

Create a class to define the structure of the user data fetched from the API.

Instructions:

1. Add a folder named Models in the root of the project.
2. Create a file named User.cs in the Models folder.
3. Define properties for the user object, including Id, Name, Email, and Address.

Step 3: Implement Asynchronous AP1 Calls

Fetch user data using asynchronous methods.

Instructions:

1. In FetchUsers.razor, inject the HttpClient service.
2. Create a method to fetch data from https://jsonplaceholder.typicode.com/users using asynchronous methods (Task and await).
3. Store the retrieved data in a list of User objects.

Step 4: Add Loading and Error States

Handle loading states and errors gracefully during the API call.

Instructions:

1. Create a bool property to track the loading state.
2. Add a string property to store error messages.
3. Update the fetch method to:
    - Show a loading indicator while the data is being fetched.
    - Display an error message if the API call fails.

Step 5: Display Retrieved Data in a Table

Render the fetched user data in a table format.

Instructions:

1. Add a button labeled Fetch New Users to trigger the API call.
2. Display the retrieved user data in a table.
3. Ensure the table dynamically updates when new data is fetched.

Step 6: Test Your Application

Verify that the application works as expected for all scenarios.

Instructions:

1. Start your application using: dotnet run
2. Navigate to http://localhost:5000/fetchusers in your browser
3. Click the Fetch New Users button:
    a. While fetching data, a loading message is displayed.

b. After fetching data successfully, a table with user data is displayed.

c. If an error occurs, an error message is displayed.