

2025 – Assignment 02

Assignment Title: Development of a React Frontend Application Using REST Countries API

Overview:

In this assignment, you are required to develop a **creative frontend application** using **React** functional components. The application must **consume data from the REST Countries API**. This project aims to showcase your skills in front-end development, API integration, application design, and deployment.

Start Date:

April 04, 2025

Deadline:

May 4, 2025, Midnight (Git Classroom will be disabled afterwards)

Objectives:

- To **develop a React application with a strong emphasis on functional components.**
- To **integrate and utilize data effectively from the REST Countries API.**
- To enhance usability through a sophisticated **CSS framework.**
- To **manage user sessions effectively**, with the option to **develop a separate REST API for user management.**
- To maintain a robust version control system through regular **git commits.**
- To **deploy the application** on a suitable hosting platform.
- To **perform comprehensive testing** across the application.

Requirements:

1. Technology Stack:

- done** • **Frontend:** **React** (with functional components)
- done** • **Language:** **JavaScript**
- done** • **CSS Framework:** Choose any modern CSS framework such as Bootstrap, **Tailwind CSS**, or **Material-UI** to enhance the application's usability.
- done** • **Backend** (Optional): You may choose to **develop a separate REST API for user management.**

- **Hosting:** The application should be hosted on a platform. (try to find a free solution)
- done • **Session Management:** Implement user session management.
- done • **Version Control:** Use Git for version control and regularly commit your code to GitHub from the beginning of the project.

2. API Integration:

You must use data from the [REST Countries API](#) to build your application.

What you need to do:

- Use **at least four different API endpoints** from the REST Countries API.
 - Example endpoints you can use:
 - done ▪ GET /all – to get a list of all countries.
 - done ▪ GET /name/{name} – to search a country by its name.
 - done ▪ GET /region/{region} – to get countries from a specific region.
 - done ▪ GET /alpha/{code} – to get full details using a country code.
- Fetch and show useful data such as:
 - done ◦ Country name
 - done ◦ Population
 - done ◦ Region
 - done ◦ Languages
 - done ◦ Flag
 - done ◦ Capital
- Let users interact with the app using:
 - done ◦ A **search bar** to find countries by name.
 - done ◦ A **dropdown menu** or **filter buttons** to filter by region or language.
- done • When users interact, the country list should **update dynamically** (without refreshing the page).

Functional Requirements:

Your app should allow the user to:

- **View details about a country**, including:
 - done ◦ Name, Capital, Region, Population, Flag, and Languages.

- **Search and filter:**

done ○ Users should be able to **search for a country by name**.

done ○ Users should be able to **filter countries by region or language**.

done • **Click on a country** to view more details (optional but recommended).

done • (Optional) Add **login functionality**:

- If you want, you can add user authentication so users can log in and access custom features (like favorite countries).

3. Testing:

- Conduct both **unit and integration tests**. Use testing frameworks like **Jest and React Testing Library**.

done • Ensure **responsiveness and cross-browser compatibility**.

4. Documentation:

done • **Document the application setup, build process, and usage instructions** in a **README file on GitHub**.

- **Provide a brief report discussing the chosen APIs, any challenges faced, and how they were resolved**.

5. Submission:

- Submit the GitHub repository link containing all source code, tests, and documentation.
- GitHub Classroom: <https://classroom.github.com/a/mNaxAqQD>
- Provide the URL of the hosted application in the README file.

Evaluation Criteria:

- Correctness and functionality of the application.
- Creativity and design implementation.
- Code quality, including readability and use of best practices.
- Completeness of documentation.
- Regularity and informativeness of git commit.
- Thoroughness of testing.

Additional Notes:

- You are encouraged to explore advanced React features and hooks.
- Consider security best practices, particularly in how you handle API keys and user data.

Marking Guide: Total 20 Marks

1. Functionality and Correctness (8 Marks)

- **Application works as expected without any errors (4 Marks)**

- ✓ • The application meets all the functional requirements specified in the assignment.
- ✓ • All features are functional and data from the REST Countries APIs is integrated and displayed correctly.

- **API Integration and Data Handling (2 Marks)**

- ✓ • Effective use of at least four different REST Countries endpoints.
- ✓ • Correct parsing, handling, and display of API data.

- **User Session Management (2 Marks)**

- ✓ • Implementation of session management, ensuring that the user state is preserved during the session.

2. Design and Usability (4 Marks)

- **Use of CSS Framework (2 Marks)**

- ✓ • Effective and aesthetic use of a CSS framework to enhance the application's usability and visual appeal.

- **Responsive Design (2 Marks)**

- ✓ • The application is responsive and provides a consistent experience across different devices and screen sizes.

3. Code Quality and Best Practices (4 Marks)

- **Code Organization and Readability (2 Marks)**

- ✓• Code is well-organized, properly commented on, and easy to read.
- ✓• Consistent naming conventions and code style.

- **Use of Git (Version Control) (2 Marks)**

- ✓• Regular and meaningful git commits with clear messages.
- ✓• Maintenance of a clean and organized repository.

4. Documentation and Reporting (2 Marks)

- **Quality of README and Documentation (2 Mark)**

- ✓• Comprehensive README file with clear setup, build, and run instructions.
- ✓• Documentation covers all aspects of the application, including how to use the APIs.

5. Testing (2 Marks)

- **Implementation of Tests (2 Marks)**

- ✓• Comprehensive unit and integration tests are provided.
- ✓• Tests cover critical functionalities and components of the application.