SE3040 – Application Frameworks
BSc (Hons) in Information Technology
Specialized in Software Engineering
3rd Year
Faculty of Computing
SLIIT

2024 – Assignment02

Assignment Title: Development of a React Frontend Application Using NASA APIs

Overview:

In this assignment, you are required to develop a creative frontend application using React functional components. The application will consume data from NASA's public APIs available at NASA API portal. This project aims to showcase your skills in front-end development, integration with external APIs, and application deployment.

Start Date:

April 15, 2024

Deadline:

May 5, 2024, Midnight (Git Classroom will be disabled afterwords)

Objectives:

- To develop a React application with a strong emphasis on functional components.
- To integrate and utilize data effectively from NASA's APIs.
- 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**:
    - **Frontend**: React (with functional components)
    - **Language**: JavaScript
    - **CSS Framework**: Choose any modern CSS framework such as Bootstrap, Tailwind CSS, or Material-UI to enhance the application's usability.
    - **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)
- **Session Management**: Implement user session management.
- **Version Control**: Use Git for version control and regularly commit your code to GitHub from the beginning of the project.

2. **API Integration**:
    - Utilize at least two different endpoints from NASA's APIs. Possible API choices include Mars Rover Photos, Astronomy Picture of the Day, or the Earth imagery APIs.
3. **Functional Requirements**:
    - A user should be able to view daily or historical astronomy-related data.
    - Incorporate user authentication for accessing personalized features (optional).
    - Display data dynamically based on user input or interactions.
4. **Testing**:
    - Conduct both unit and integration tests. Use testing frameworks like Jest and React Testing Library.
    - Ensure responsiveness and cross-browser compatibility.
5. **Documentation**:
    - 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.
6. **Submission**:
    - Submit the GitHub repository link containing all source code, tests, and documentation.
    - GitHub Classroom: https://classroom.github.com/a/V1F4A3D5
    - 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 NASA APIs is integrated and displayed correctly.
- **API Integration and Data Handling (2 Marks)**
    - Effective use of at least two different NASA API 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, 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.