2024 – Assignment02

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

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

**Chosen APIs:**

1. **Near-Earth Object (NEO) API**: This API provides data about near-Earth objects, including their orbits, sizes, and close approaches to Earth. It helps in tracking potentially hazardous asteroids and understanding their characteristics.

[https://api.nasa.gov/neo/rest/v1/feed?start\_date=${astFullDate}&end\_date=${astFullDate}&api\_key=${api\_key}](https://api.nasa.gov/neo/rest/v1/feed?start_date=$%7bastFullDate%7d&end_date=$%7bastFullDate%7d&api_key=$%7bapi_key%7d)

1. **Astronomy Picture of the Day (APOD) API**: This API offers the daily astronomy picture along with a brief explanation written by a professional astronomer. It provides stunning images of celestial objects and phenomena, fostering interest and appreciation for space exploration.

[https://api.nasa.gov/planetary/apod?api\_key=yOPGtFFJvRbP0DddzPAkSTr5frBFNZW2EJPb9jkK&date=${date}](https://api.nasa.gov/planetary/apod?api_key=yOPGtFFJvRbP0DddzPAkSTr5frBFNZW2EJPb9jkK&date=$%7bdate%7d)

1. **Earth Observatory Natural Event Tracker (EONET) API**: EONET tracks natural events worldwide and provides information about them, including wildfires, hurricanes, earthquakes, and more. It helps in monitoring and understanding natural disasters and their impact on the environment.

<https://eonet.gsfc.nasa.gov/api/v2.1/events>

**Challenges Faced:**

* **Integrating Multiple APIs:** Integrating data from multiple APIs while ensuring efficient data retrieval and processing can be complex. Handling asynchronous requests and managing responses from different APIs posed a challenge.
* **User Authentication and Authorization:** Implementing secure user authentication and authorization mechanisms, such as JWT tokens, to protect sensitive user data and restrict access to authorized users.
* **Handling API Rate Limits and Errors:** Dealing with API rate limits, unexpected errors, and downtime while ensuring a seamless user experience required careful error handling and caching strategies.

**Resolutions:**

* Asynchronous Handling: Utilized modern JavaScript features like async/await and Promises to handle asynchronous operations efficiently, ensuring smooth data retrieval and processing.
* JWT Tokens: Implemented JSON Web Tokens (JWT) for secure user authentication and authorization, providing users with access to protected resources based on their roles and permissions.
* Error Handling Middleware: Developed error handling middleware to intercept and respond to errors gracefully, providing meaningful error messages and status codes to clients.
* Caching Mechanisms: Implemented caching mechanisms to cache API responses and reduce the number of requests made to external APIs, thereby mitigating API rate limits and improving application performance.
* Environment Variables: Stored sensitive information, such as API keys and database credentials, securely as environment variables, minimizing the risk of exposure and unauthorized access to sensitive data.