**SOCIAL MEDIA DATA AGGREGATION**

Designing the Social Media Aggregation API involved creating a scalable and efficient backend service using Node.js and Express. The core goal was to build an API that aggregates data from social media platforms like Twitter and Instagram, while ensuring robustness and performance.

* **Approach to Design and Integration**

The RESTful principles guiding the layout of the API resulted in well-defined endpoints for aggregated and platform-specific statistics. I created fictitious services that returned specified data in order to mimic interactions with external platforms. Without depending on real-time data from external APIs, this method allowed me to concentrate on internal logic and response handling.

* **Challenges and Solutions**

Rate restriction was a difficult concept to implement. I was able to overcome the initial difficulty of managing request limitations by tracking and restricting requests based on IP addresses using an in-memory cache. By striking a balance between efficacy and simplicity, this method prevented abuse of the API.   
There were challenges in testing the API as well, especially with timing in rate-limiting tests. To ensure consistent test results and dependable performance, I adjusted Jest timeouts and improved the rate-limiting logic.

* **Role of AI Tools**

The development process was greatly facilitated by AI tools, which offered code snippets, debugging advice, and optimization recommendations. This assistance expedited progress and freed me up to concentrate on project strategy, including architectural and design choices.   
All things considered, the project proved to be a useful practice for designing APIs, emphasizing the significance of efficient caching, rate limitation, and modular code structure. The development process was further expedited and made more insightful by the application of AI techniques.