# Assignment for NET Developer: API Aggregation

## Objective:

Your task is to develop an API aggregation service that consolidates data from multiple external APIs and provides a unified endpoint to access the aggregated information. The goal is to create a scalable and efficient system that fetches data from various sources, aggregates it, and delivers it through a single API interface.

## Requirements:

* Develop a .NET-based API aggregation service using ASP.NET Core.
* Design and implement an architecture that allows for easy integration of new APIs.
* Implement functionality to fetch data from at least three different external APIs simultaneously. You can choose any APIs you prefer (e.g., weather, news, social media, etc.).
* Create an API endpoint that allows users to retrieve the aggregated data.
* Provide the option to filter and sort the aggregated data based on specified parameters (e.g., date, relevance, category).
* Apply proper error handling and implement a fallback mechanism in case any of the external APIs are unavailable.
* Write unit tests to ensure the reliability and correctness of the codebase.
* Document the API aggregation service, including its endpoints, input/output formats, and any necessary setup or configuration instructions.

## Optional:

* Utilize caching techniques to optimize performance and minimize redundant API calls.
* Utilize parallelism to decrease response times.
* Create an API endpoint to retrieve request statistics. Specifically, return for each API the total number of requests and the average response time, grouped in performance buckets (e.g. fast <100ms, average 100-200ms, slow > 200ms – the values are indicative feel free to tweak). For the purposes of the exercise source data can be stored in memory.

## Evaluation Criteria:

Your solution will be evaluated based on the following criteria:

* Successful integration of multiple external APIs and retrieval of aggregated data.
* Adherence to coding best practices, including code structure, readability, and maintainability.
* Efficient use of asynchronous programming and scalability considerations.
* Effective error handling and fallback mechanism for API failures.
* Completeness and correctness of unit tests to ensure code reliability.
* Quality and clarity of documentation provided.
* (Optional) Proper implementation of requests statistics feature.
* (Optional) Performance optimizations, such as caching, to improve response times.

Please note that this assignment is designed to assess your skills in API integration, asynchronous programming, error handling, authentication, and documentation.

Sample APIs that you can use for an API aggregation assignment:

1. OpenWeatherMap API: https://openweathermap.org/api

This API provides weather data for various locations around the world. You can request weather data by city, ZIP code, or geographic coordinates.

1. Twitter API: https://developer.twitter.com/en/docs/twitter-api

The Twitter API allows you to access Twitter's data, including tweets, users, and trends. You can use this API to aggregate tweets based on specific search terms, user handles, or hashtags.

1. News API: https://newsapi.org/

The News API provides access to headlines and articles from various news sources around the world. You can use this API to aggregate news articles based on specific keywords, topics, or sources.

1. Spotify Web API: https://developer.spotify.com/documentation/web-api/

The Spotify Web API allows you to access Spotify's music data, including albums, tracks, playlists, and artists. You can use this API to aggregate music based on specific search terms, genres, or artists.

1. GitHub API: https://docs.github.com/en/rest

The GitHub API provides access to GitHub's data, including repositories, issues, and pull requests. You can use this API to aggregate information about specific users or repositories.

Note that these are just examples, and you can use any other APIs that you find suitable for your assignment.

Please provide your solution in a GitHub or GitLab or similar repository.