

Aggregation API Report for Agile Actors

1. Overview

This .NET API asynchronously fetches data from three (3) distinct external API's, combines their results into one unified endpoint with a specific response schema while also logging API request performance statistics. The API is designed to be highly scalable to allow for very easy integration of additional external API with minimal code modifications, as required by the assignment.

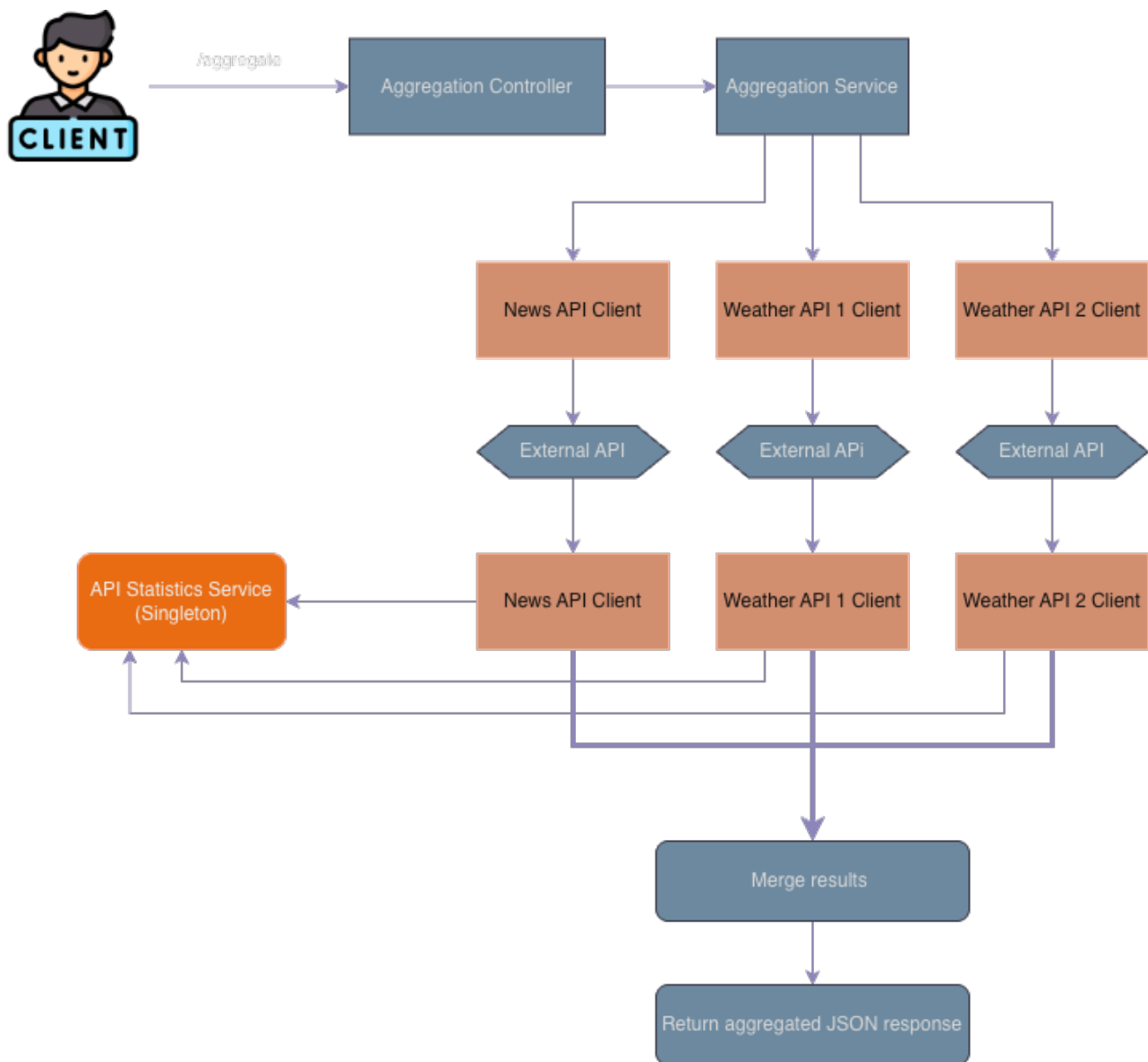
More specifically, the implemented API receives user input (location, news keyword) through the HTTP request query parameters, performs parallel requests to one **(1) news API** and two **(2) weather APIs** to retrieve weather data and news articles for that location and filter news articles based on the news keyword. So in total, this API utilizes three external APIs + plus one utility geocoding API to convert a location's name to latitude and longitude. (Because one of the two weather APIs requires actual geographic coordinates).

The API can be considered RESTful because it is stateless. uses common HTTP status codes and exposes predictable URLs with representations returned as JSON.

2. Technical Implementations Summary

The API implements the following:

- Successful integration of three (3) external APIs and retrieval of aggregated data
 - Weather API 1 : <https://api.openweathermap.org/data/3.0/>
 - Weather API 2 : <https://api.weatherstack.com/>
 - News API: <https://newsapi.org/v2/>
- 1 GET /aggregate endpoint for data aggregation
- 1 GET /stats endpoint for performance statistics
- Console logging messages (information, warnings, errors via ILogger injection)
- Performance logging of requests to external APIs using a C# **concurrent dictionary** to allow for concurrent updates by multiple requests and thread-safe read/writes without the risks of explicit locking. The API statistics service is registered as a singleton service in the container.
- Utilization of appsettings.json file to easily store/update external API configuration.
- Clients that fetch data from external APIs are:
 - **Type**-registered in the container so that each external API is clearly defined and configured (by reading the configuration settings in appsettings.json)
 - Registered as **scoped** services because their functionality is request-oriented, meaning that we want the same request context without leaking state across different requests



3. Endpoints

GET `/aggregation/aggregate`

Accepted Usage:

`/aggregation/aggregate?location=[argument1]`

`/aggregation/aggregate?location=[argument1]&newskeyword=[argument2]`

Example: `/aggregation/aggregate?location=Athens&newskeyword=Greece`

Successful Response schema:

```
{
  "success": true,
  "data": {
    "externalData": [
      {
        "source": "source 1",
        "data": { ... }
      },
      {
        "source": "source 2",
```

```
    "data": { ... }  
  }  
}]}
```

Error Response schema:

```
{  
  "error": "Error message.",  
  "errorCode": "API_UNAVAILABLE"  
}
```

GET /aggregation/stats

Accepted Usage:

/aggregation/stats

Successful Response schema:

```
{  
  "API 1": {  
    "totalRequestCount": 7,  
    "averageResponseTimeMs": 101.71429,  
    "groupedByPerformanceDictionary": {  
      "average": 2,  
      "fast": 5  
    }  
  },  
  "API 2": {  
    "totalRequestCount": 7,  
    "averageResponseTimeMs": 191,  
    "groupedByPerformanceDictionary": {  
      "average": 2,  
      "fast": 2  
    }  
  },  
  "API 3": {  
    "totalRequestCount": 7,  
    "averageResponseTimeMs": 278.14285,  
    "groupedByPerformanceDictionary": {  
      "slow": 4,  
      "average": 3  
    }  
  }  
}
```

Error Response schema:

```
{  
  "error": "Error message.",  
}
```

