Aggregator API – Project Documentation

# 1. Project Overview

The Aggregator API is a .NET 8 Web API designed to fetch and combine data from multiple external providers and present aggregated results.  
Main goals:  
- Aggregate data from multiple external sources  
- Track response times and categorize requests as fast, average, or slow  
- Use in-memory caching to improve performance  
- Provide a clear API for clients to fetch both aggregated data and statistics

# 2. Architecture

Components:  
1. Controllers  
 - AggregationController: Handles HTTP requests for aggregated data and statistics.  
 - Endpoints:  
 - GET /aggregate?from=&to=&limit= – fetch aggregated items  
 - GET /stats – fetch API statistics  
2. Services  
 - AggregationService: Core service to fetch data from providers, combine results, and cache them.  
 - Uses IMemoryCache to store aggregated results temporarily.  
3. Providers  
 - Implement the IExternalProvider interface  
 - Fetch data from external APIs: OpenMeteo, HackerNews, SpaceflightNews  
4. Domain  
 - AggregatedItem: Represents a single fetched item  
 - AggregationContext: Holds request parameters like From, To, and Limit  
 - ApiStatistics: Tracks response times and counts requests by category  
5. Caching  
 - IMemoryCache is used to store aggregated results  
 - Reduces response time for repeated requests

# 3. Data Flow

Client (Swagger / HTTP) -> AggregationController -> AggregationService -> IExternalProviders -> ApiStatistics -> Response  
IMemoryCache is used in AggregationService to store and retrieve cached results.

# 4. Providers

OpenMeteo – Weather data  
HackerNews – Tech and news stories  
SpaceflightNews – Articles about spaceflight  
Each provider implements FetchAsync() and converts raw data into AggregatedItem.

# 5. Statistics

ApiStatistics tracks:  
- TotalRequests  
- AverageResponseMs  
- FastCount (<100ms)  
- AverageCount (100-200ms)  
- SlowCount (>200ms)

Example output:  
{  
 "OpenMeteo": {"totalRequests": 1, "averageResponseMs": 349, "fastCount": 0, "averageCount": 0, "slowCount": 1},  
 "HackerNews": {"totalRequests": 1, "averageResponseMs": 1562, "fastCount": 0, "averageCount": 0, "slowCount": 1},  
 "Spaceflight": {"totalRequests": 1, "averageResponseMs": 713, "fastCount": 0, "averageCount": 0, "slowCount": 1}  
}

# 6. Caching

IMemoryCache stores aggregated results to improve performance.  
Cached responses reduce external API calls and response times.

# 7. Example API Usage

Fetch Aggregated Items:  
GET /aggregate?from=2025-08-01&to=2025-08-16&limit=5  
Response example:  
[{"Id":"1","Title":"SpaceX launches new satellite","Source":"Spaceflight","PublishedAt":"2025-08-15T12:30:00Z"},  
{"Id":"2","Title":"HackerNews trending story","Source":"HackerNews","PublishedAt":"2025-08-15T10:00:00Z"}]

Fetch Statistics:  
GET /stats  
Response example:  
{  
 "OpenMeteo": {"totalRequests":3,"averageResponseMs":120,"fastCount":1,"averageCount":1,"slowCount":1},  
 "HackerNews": {"totalRequests":3,"averageResponseMs":200,"fastCount":0,"averageCount":2,"slowCount":1},  
 "Spaceflight": {"totalRequests":3,"averageResponseMs":100,"fastCount":2,"averageCount":1,"slowCount":0}  
}

# 8. Technology Stack

- Backend: .NET 8 Web API  
- Dependency Injection: Built-in .NET DI  
- Caching: IMemoryCache  
- Unit Testing: xUnit + Moq  
- API Testing: Swagger   
- External APIs: OpenMeteo, HackerNews, SpaceflightNews

# 9. Notes

- Aggregation respects From, To, and Limit parameters  
- Statistics are updated with each request  
- Cached results improve performance and reduce response times  
- Providers are modular and easily extendable