# **Project Architecture Overview**

## **1. Introduction**

The Demo project is aimed at implementing a robust .NET Core architecture, leveraging industry-best practices and technologies to achieve a scalable and maintainable solution. The primary goals include the incorporation of JWT token-based authentication, the implementation of the Command Query Responsibility Segregation (CQRS) pattern, and the utilization of Entity Framework for seamless interaction with a SQL database. The project is also containerized using a Docker file, enhancing its portability and deployment efficiency.

## **2. System Components**

### **2.1. Demo**

* Description: The main project module containing .NET 6.0 controllers and related files.
* Key Components:
  + .NET 6.0 Controllers
  + Appsettings
  + RateLimitting

### **2.2. Demo.Core**

* Description: Module responsible for handling Data Transfer Objects (DTOs), ViewModels, interfaces, and command and query implementation for Command Query Responsibility Segregation (CQRS) pattern.
* Key Components:
  + DTOs
  + ViewModels
  + Interfaces
  + Command and Query Handlers

### **2.3. Demo.Infrastructure**

* Description: Module responsible for database-related tasks, including repositories and migration files.
* Key Components:
  + Repositories
  + Migration Files

### **2.4. Demo.Services**

* Description: Module responsible for implementing services and handlers.
* Key Components:
  + Services
  + Handlers

### **2.5. Demo.Test**

* Description: Module containing test cases for controller methods.
* Key Components:
  + Controller Test Cases

## **3. Data Flow**

* Controller → Mediator → Service → Repository :
* The controller sends a request or command to the mediator.The mediator, being the central coordinator, determines the appropriate service to handle the request.The mediator forwards the request to the chosen service.The service layer processes the request, possibly involving interactions with the repository for data operations.The results may be returned through the same path, with the mediator serving as a conduit for responses back to the controller.

## **4. Authentication**

* Authentication Mechanism: JWT token
* Justification: JWT token authentication was chosen due to its statelessness, efficiency, cross-domain compatibility, security features, standardization, ease of integration, support for decentralized authentication, and scalability.

## **5. Database**

* Database Type: SQL
* ORM Framework: Entity Framework
* Justification:SQL database and Entity Framework were chosen for the Demo project due to the relational data model, ensuring data integrity and consistency, and Entity Framework's ORM capabilities, which enhance productivity by simplifying database interactions and facilitating rapid development..

## **6. Rate limiting**

* Rate limiting is implemented to control the rate of incoming requests, preventing abuse and ensuring fair usage of system resources.

### **7. Docker File**

* The Docker file is an integral component of the system, encapsulating the configuration for containerizing the application. It includes specifications for the .NET 6.0 framework and sets up the SQL database, providing a comprehensive and portable deployment environment.

## **8. Conclusion**

The selected architecture for the Demo project offers a robust foundation by combining .NET Core, CQRS pattern, Entity Framework, JWT token authentication, rate limiting, and Docker containerization. This choice ensures scalability, maintainability, and security, while also providing efficient data handling and cross-platform compatibility. The technology stack aligns with industry best practices, fostering developer productivity and positioning the project for future advancements and community support.



