Dynamic Mapping System

# 1. Introduction

Overview: General introduction to the API, its purpose, and main functionalities.  
Architecture Summary: High-level description of the architecture (e.g., MVC, microservices, REST, or GraphQL), including any third-party libraries or services used.  
Authentication: Describe the authentication/authorization mechanisms (e.g., OAuth2, JWT).

The provided solution is a RESTful API that represent an extensible dynamic mapping engine/system using .Net 8  
The API is capable of mapping data between our internal DIRS21 C# data models and external data models used by our partners.  
It handles the conversion of our internal DIRS21 models to partner-specific formats and the mapping of incoming partner data to our internal DIRS21 models.

Architecture: This API consists of four layers.  
- The Data Access Layer (DAL – Class Library)  
- The Business Logic Layer (BLL – Class Library)  
- The Data Models Layer (DataModels – Class Library)  
- The RESTful API project (DynamicMapping – Web API)

# 2. Getting Started

Prerequisites: Tools and dependencies required to run or consume the API.  
Installation: Steps to set up the API locally or on a server.  
Configuration: Key configuration settings (e.g., environment variables, database connections).  
Running the API: Instructions for starting the API and any setup commands.  
Example Requests: Show simple API requests to help users quickly get started.

# 3. API Endpoints

Overview: Describe how the endpoints are structured (e.g., RESTful resources, versioning strategy).  
Detailed List of Endpoints:  
 - Endpoint: HTTP Method (GET/POST/PUT/DELETE).  
 - URL: The path to access the resource.  
 - Description: What this endpoint does.  
 - Parameters: List of query, path, and body parameters, including data types and required fields.  
 - Request/Response: Examples of request payloads and expected responses (success and error).  
 - Authentication: Whether authentication is required, and if so, how.  
 - Errors: Common error codes and messages.

# 4. Key Classes and Methods

Class Overview: Describe key classes used in the API (e.g., controllers, services, models).  
 - Class Name: Description of the class.  
 - Methods: List of key methods, their arguments, and return values.  
 - Usage Example: Example of how the class or method is used within the API.  
Error Handling: Description of how errors are handled within the system.

# 5. Data Models

Database Schema: High-level view of the database tables or document structure if using NoSQL.  
Model Definitions: Explain data models/entities used by the API (attributes, relationships).  
Validation Rules: Define validation rules for models.

# 6. Extending the API

Architecture Considerations: High-level architectural principles to keep in mind when extending the API.  
How to Add a New Endpoint:  
 - Step-by-step instructions on how to add a new API endpoint.  
 - Include details about routing, controller creation, service methods, etc.  
Adding New Models: How to extend the database schema and data models.  
Custom Middleware: Explain how to add or extend middleware (e.g., logging, authentication).

# 7. Deployment

Deployment Environments: How to deploy the API (e.g., production, staging).  
CI/CD: Overview of any continuous integration/continuous deployment processes used.  
Monitoring: List tools used for monitoring and how to set them up (e.g., error tracking, performance monitoring).

# 8. Testing

Unit Testing: Describe how the API is tested at the unit level (key testing frameworks and examples).  
Integration Testing: Explain integration tests and how they are set up.  
Test Coverage: Recommendations for ensuring coverage when extending the system.

# 9. Changelog

Keep a detailed changelog of API changes, such as new features, breaking changes, and bug fixes.