**Technical / Architecture Specification**

**Project Title: DTN ADMLOD Export Prototype  
Prepared by: Jordan Eisenback  
Date: October 5, 2025**

**1. Overview**

**This document describes the technical architecture and design for a modular, testable C# command-line prototype that simulates the DTN ADMLOD export process. The prototype:**

* **Generates ADMLOD-compliant export files using data from a mocked persistence layer.**
* **Simulates FTP upload behavior.**
* **Dynamically generates DTN-like response feedback based on export command validation.**
* **Uses dependency injection to decouple CLI orchestration from business logic and data access.**

**2. Technology Stack**

| **Component** | **Details** |
| --- | --- |
| **Language** | **C#** |
| **Framework** | **.NET 6.0 or later** |
| **Project Type** | **Console Application** |
| **Dependencies** | **.NET Standard Library** |
| **Configuration** | **appsettings.json (paths, flags)** |
| **File I/O** | **Local file system (mocked FTP)** |

**3. Architecture Overview**

**🔧 Core Modules**

| **Module/Class** | **Responsibility** |
| --- | --- |
| **Program.cs** | **CLI parsing, orchestration** |
| **IProductRepository** | **Interface for product data access** |
| **IProductGroupRepository** | **Interface for group data access** |
| **JsonProductRepository.cs** | **Loads product data from JSON** |
| **JsonGroupRepository.cs** | **Loads group data from JSON** |
| **AdmLodFileGenerator.cs** | **Generates ADMLOD export content** |
| **FtpClientMock.cs** | **Simulates FTP upload** |
| **IResponseGenerator** | **Interface for dynamic response generation** |
| **MockResponseGenerator.cs** | **Generates response based on command validation** |
| **Logger.cs** | **Structured logging to console and file** |

**🧠 Dependency Injection**

**All services are injected via constructors to support modularity and testability. A simple DI container or manual wiring in Program.cs is used to register:**

* **Repositories**
* **Response generator**
* **Logger**

**4. Data Flow**

**Plain Text**

**User triggers export via CLI**

**↓**

**Program.cs parses arguments and initializes services**

**↓**

**Repositories load product and group data from JSON**

**↓**

**AdmLodFileGenerator generates export file**

**↓**

**FtpClientMock simulates upload (success or failure)**

**↓**

**MockResponseGenerator generates dynamic response**

**↓**

**Logger logs all actions and errors**

**↓**

**Console and optional log file show results**

**Show more lines**

**5. File Structure**

**/AdmLodPrototype**

**Program.cs**

**AdmLodFileGenerator.cs**

**FtpClientMock.cs**

**Logger.cs**

**/Repositories**

**IProductRepository.cs**

**IProductGroupRepository.cs**

**JsonProductRepository.cs**

**JsonGroupRepository.cs**

**/Responses**

**IResponseGenerator.cs**

**MockResponseGenerator.cs**

**/MockFiles**

**products.json**

**productgroups.json**

**adm\_export.txt**

**export.log**

**6. Error Simulation**

| **Scenario** | **Trigger Mechanism** |
| --- | --- |
| **Missing Fields** | **Invalid entries in JSON** |
| **FTP Failure** | **CLI flag --error** |
| **Invalid Command Format** | **Malformed data in repository** |
| **Response Errors** | **Generated dynamically by MockResponseGenerator** |

**7. Logging & Observability**

* **All actions and errors are logged with:** 
  + **Severity (INFO, WARN, ERROR, SUCCESS)**
  + **Timestamps**
  + **Correlation IDs**
* **Logs are written to console and optionally to a file via --logfile flag.**

**8. Extensibility & Maintainability**

* **Modular Design: Each responsibility is encapsulated in a separate class.**
* **Mock Data: Easily extendable via JSON files.**
* **Error Handling: Centralized and categorized.**
* **Configuration: Externalized via appsettings.json.**
* **Testability: All services mockable via interfaces.**

**9. Security**

* **No real credentials or sensitive data are used.**
* **All file operations are local and sandboxed.**

**10. Build & Run Instructions**

**Shell**

**dotnet build**

**dotnet run export --success**

**dotnet run export --error --logfile export.log**

**Show more lines**

**11. Glossary**

* **ADMLOD: Automated Data Management Load, DTN’s batch file protocol.**
* **FTP: File Transfer Protocol, simulated in this prototype.**
* **Mocked Persistence Layer: Externalized data source accessed via repository interfaces.**
* **Dynamic Response: Simulated DTN feedback generated based on export command validation.**
* **Dependency Injection: Technique for injecting services into modules to improve modularity and testability.**