**Functional Requirements**

**🧭 Command-Line Interface**

* **The prototype must be executable via CLI.**
* **Primary command: export**
* **Optional arguments:** 
  + **--success or --error to simulate response type**
  + **--logfile <filename> to specify log output**

**Example Usage:**

**Shell**

**dotnet run export --success**

**Show more lines**

**📦 Mocked Persistence Layer**

* **Product and Product Group data must be loaded from external sources (e.g., JSON, YAML).**
* **Data access must be abstracted via interfaces:** 
  + **IProductRepository**
  + **IProductGroupRepository**
* **CLI must support configuration file input for data paths and logging options.**

**Each Product includes:**

* **Product Code (PIDXCode)**
* **Product Name**
* **Product Family**

**Each Product Group includes:**

* **Group Name**
* **List of assigned Product Codes**

**📄 ADMLOD Export File Generation**

* **The export file must follow DTN ADMLOD format:** 
  + **Header: VERSION;2.4;**
  + **Export command: EXPORT;PGRP;;Y;**
  + **One MERGE;PGRP;...; line per group**
  + **One MERGE;PRODUCT;...; line per product**
  + **One MERGE;PGRPPROD;...; line per group-product assignment**
* **File is saved to a configurable directory (default: MockFiles/adm\_export.txt)**

**🚚 FTP Upload Simulation**

* **FTP upload is simulated by writing the export file locally.**
* **Upload behavior is controlled via CLI flags.**
* **Simulated failures must be logged and handled gracefully.**

**🔁 Dynamic DTN Response Generation**

* **Response feedback must be generated dynamically using a response generator service:** 
  + **IResponseGenerator**
* **Responses must reflect validation results of export commands.**
* **CLI output must distinguish between successes and errors.**

**📋 Logging**

* **All actions (generation, upload, response) must be logged.**
* **Logs must include:** 
  + **Severity levels (INFO, WARN, ERROR, SUCCESS)**
  + **Timestamps**
  + **Correlation IDs**
* **Logs are written to console and optionally to a file.**

**❗ Error Handling**

* **Simulate and log:** 
  + **Missing required fields**
  + **Invalid command formats**
  + **FTP upload failures**
  + **Validation errors in dynamic responses**
* **Categorize errors (validation, FTP, parsing)**
* **Implement retry logic for recoverable operations**

**3. Non-Functional Requirements**

* **Language: C#**
* **Framework: .NET 6 or later**
* **Platform: Windows and cross-platform**
* **All integrations (FTP, DTN) are mocked**
* **Configuration must be externalized (e.g., appsettings.json)**
* **Code must be modular and testable via dependency injection**

**4. Architecture & Module Responsibilities**

| **Module/Class** | **Responsibility** |
| --- | --- |
| **Program.cs** | **CLI parsing, orchestration** |
| **IProductRepository** | **Interface for product data access** |
| **IProductGroupRepository** | **Interface for group data access** |
| **AdmLodFileGenerator.cs** | **Generates ADMLOD export content** |
| **FtpClientMock.cs** | **Simulates FTP upload** |
| **IResponseGenerator** | **Interface for dynamic response generation** |
| **Logger.cs** | **Handles structured logging** |

**5. User Stories / Use Cases**

| **As a...** | **I want to...** | **So that...** |
| --- | --- | --- |
| **CLI User** | **Trigger an export operation** | **I can generate and test ADMLOD files** |
| **Developer** | **Simulate FTP upload and dynamic response** | **I can demonstrate the integration flow** |
| **QA/Tester** | **Simulate and observe error scenarios** | **I can verify error handling and logging** |
| **Stakeholder** | **Review logs and outputs** | **I can validate the prototype’s behavior** |

**6. Acceptance Criteria**

* **Export file is generated with correct structure and mock data.**
* **FTP upload is simulated and logged.**
* **Response feedback is dynamically generated and reflects command validity.**
* **CLI supports flexible testing via flags.**
* **All services are injected and testable in isolation.**
* **Logs include severity, timestamps, and correlation IDs.**

**7. 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.**