**User Stories – DTN ADMLOD Export Prototype (v2)**

**🟢 Export Trigger**

**As a** command-line user  
**I want to** trigger an export operation via a CLI command  
**So that** I can generate an ADMLOD export file for Products and Product Groups using repository-backed mock data

**Acceptance Criteria:**

* CLI accepts export command
* Export file is generated in /MockFiles
* Log confirms operation start and completion

**🟢 ADMLOD File Generation**

**As a** developer  
**I want to** generate an ADMLOD-compliant export file from mock Products and Product Groups  
**So that** the file structure matches DTN requirements and can be used for demonstration or testing

**Acceptance Criteria:**

* File includes header, export command, and MERGE lines
* Format matches ADMLOD specification
* File saved to /MockFiles/adm\_export.txt

**🟢 FTP Upload Simulation**

**As a** developer or tester  
**I want to** simulate an FTP upload by writing the export file to a local directory and logging the action  
**So that** I can demonstrate the integration flow without connecting to a real FTP server

**Acceptance Criteria:**

* File is written locally
* Upload action is logged
* FTP failure can be simulated via CLI flag

**🟢 Dynamic DTN Response Generation**

**As a** developer or QA  
**I want to** generate DTN-like responses dynamically based on export command validation  
**So that** I can simulate realistic feedback without relying on static files

**Acceptance Criteria:**

* Response generator evaluates each command
* Success and error messages are generated dynamically
* Response output is logged and displayed

**🟢 Response Parsing**

**As a** user  
**I want to** parse the generated DTN response after the simulated upload  
**So that** I can see which export commands would succeed or fail in a real scenario

**Acceptance Criteria:**

* Response is parsed
* Success and error counts are displayed
* Log includes parsing results

**🟢 Error Scenario Simulation**

**As a** tester  
**I want to** simulate error scenarios (e.g., missing fields, FTP failure, invalid command format)  
**So that** I can verify that the prototype handles and logs errors appropriately

**Acceptance Criteria:**

* Errors are triggered via repository data or CLI flags
* Errors are categorized (validation, FTP, parsing)
* Logs include severity and correlation ID

**🟢 Logging**

**As a** stakeholder  
**I want to** see a log of all actions and errors in the console and optionally in a log file  
**So that** I can review the prototype’s behavior and troubleshoot issues

**Acceptance Criteria:**

* Logs include timestamps, severity, and correlation ID
* Console and file logging are supported
* Log file path can be specified via CLI

**🟢 CLI Options**

**As a** command-line user  
**I want to** specify options (e.g., response type, log file location)  
**So that** I can control the prototype’s behavior for different test scenarios

**Acceptance Criteria:**

* CLI supports --success, --error, and --logfile flags
* Invalid options are handled gracefully
* Help message is shown for missing/invalid input

**🟢 Review Output**

**As a** stakeholder or QA  
**I want to** review the generated export file and the parsed response output  
**So that** I can confirm the prototype meets the business and technical requirements

**Acceptance Criteria:**

* Export file matches ADMLOD format
* Response output reflects command validation
* Logs and files are accessible for review

**🟢 Mock Data & Response Configuration**

**As a** QA or developer  
**I want to** configure mock product data and response logic via external files  
**So that** I can simulate different business scenarios and edge cases

**Acceptance Criteria:**

* Data is loaded via repository interfaces from JSON or other formats
* Response logic is configurable via service injection
* Changes are reflected in export and parsing behavior