**📄 Software Requirements Specification (SRS)**

**1. Introduction**

**1.1 Purpose**

This document specifies the software requirements for a **Cloud-Native E-Commerce Order Management & Analytics Platform** deployed on **Microsoft Azure**. The platform integrates cloud storage, data pipelines, ETL processing, microservices, analytics, and role-based access, following cloud-native and modular design principles.

**1.2 Scope**

The system is designed to:

* Support **SaaS/PaaS deployment** on Azure
* Enable ingestion, transformation, and storage of order-related data
* Provide a set of RESTful APIs via FastAPI microservices
* Deliver analytics and visual insights via Azure Databricks
* Enforce access control and ensure secure operations using Azure Key Vault

**1.3 Intended Audience**

* Product Owners
* Development & DevOps Teams
* QA Engineers
* Data Analysts
* Security Engineers

**2. Overall Description**

**2.1 System Architecture**

The platform is developed in **modular phases** for scalability and maintainability:

| **Phase** | **Description** |
| --- | --- |
| 1 | Planning, SRS + HLD + UML |
| 2 | Cloud setup, Data Ingestion (ADLS + ADF) |
| 3 | Data transformation (Databricks) |
| 4 | FastAPI Microservices + SQL VM |
| 5 | Testing & Validation |
| 6 | Analytics & Reporting |
| 7 | Access Control & Monitoring |

**2.2 Deployment Model**

* **Azure PaaS**: Azure Data Lake Storage (ADLS), Azure Data Factory, Azure Databricks
* **Azure IaaS**: Virtual Machine (VM) hosting FastAPI & SQL DB
* **SaaS Design**: Multi-tenant-ready with secure module-level isolation

**3. Functional Requirements**

**Phase 1: Planning & Architecture**

* **FR1.1**: Create Software Requirements Specification (SRS)
* **FR1.2**: Design High-Level Architecture Diagram
* **FR1.3**: Design UML Class Diagram for database (entities: Customers, Orders, Products)

**Phase 2: Cloud Setup & Data Ingestion**

* **FR2.1**: Set up Azure Data Lake Storage (ADLS) with raw and bronze containers
* **FR2.2**: Upload initial datasets to raw container
* **FR2.3**: Create ADF pipeline to move data from raw → bronze container

**Phase 3: ETL & Transformation (Azure Databricks)**

* **FR3.1**: Load data from bronze layer to Databricks
* **FR3.2**: Apply transformations:
  + transformed\_orders: Derived columns total\_amount, order\_month
  + inactive\_customers: Customers with no orders
  + low\_stock\_products: Products with stock\_level < threshold
* **FR3.3**: Store resulting DataFrames in silver container in ADLS

**Phase 4: FastAPI Microservices**

* **FR4.1**: Load silver-layer data into **SQL Database on Azure VM**
* **FR4.2**: Expose FastAPI microservices with:
  + **CRUD endpoints for Product Table**
    - GET /products, POST /products, PUT /products/{id}, DELETE /products/{id}
  + **Specialized endpoints:**
    - GET /orders – Return all order details
    - GET /customer-loyalty – Return customer loyalty analytics
    - POST /inventory-update – Update stock levels
    - GET /low-stock-alerts – Get list of low stock products

**Phase 5: Testing & Validation**

* **FR5.1**: Perform **unit testing** of FastAPI endpoints using pytest
* **FR5.2**: Conduct **manual testing** for data flow and service stability

**Phase 6: Data Analytics & Reporting**

* **FR6.1**: Analyze data in Azure Databricks and generate visual insights:
  + Top 5 cities by revenue
  + Product categories with highest stockouts
  + Average session duration by loyalty status
* **FR6.2**: Push visualized results and final DataFrames to **gold container in ADLS**

**Phase 7: Access Control & Monitoring**

* **FR7.1**: Simulate role-based access (Admin, Analyst, Viewer)
* **FR7.2**: Discuss security practices:
  + Azure Key Vault for storing secrets and credentials
  + Network-level isolation (NSGs, firewalls)

**4. Non-Functional Requirements**

| **Category** | **Requirement** |
| --- | --- |
| **Performance** | REST APIs must respond within 500ms for 95% of requests |
| **Scalability** | Horizontal scaling enabled for microservices |
| **Security** | Use of Azure Key Vault, Role-Based Access |
| **Availability** | 99.9% uptime for APIs |
| **Maintainability** | Microservices must follow modular design for easy updates |
| **Observability** | Integrate Azure Monitor for logging and alerts |

**5. External Interfaces**

**5.1 APIs**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /products | GET/POST/PUT/DELETE | CRUD operations |
| /orders | GET | List all orders |
| /customer-loyalty | GET | Show loyalty metrics |
| /inventory-update | POST | Update product stock |
| /low-stock-alerts | GET | Get products below stock threshold |

**5.2 Storage Interfaces**

* **Azure Data Lake (ADLS)**: Raw → Bronze → Silver → Gold
* **SQL Database**: Product/Customer/Order tables for API consumption

**5.3 Tools & Services**

| **Component** | **Technology** |
| --- | --- |
| Storage | Azure Data Lake Storage (ADLS) |
| ETL | Azure Data Factory (ADF), Azure Databricks |
| Backend | FastAPI on Azure VM |
| Database | SQL Server on Azure VM |
| Visualization | Databricks, Matplotlib/Seaborn |
| Monitoring | Azure Monitor |
| Security | Azure Key Vault |

**6. Data Models**

**6.1 Entity Diagram (Summary)**

* **Customer**: customer\_id, name, email, loyalty\_status, ...
* **Order**: order\_id, customer\_id, order\_date, total\_amount, order\_month, ...
* **Product**: product\_id, name, category, stock\_level, ...

(UML Class Diagram was prepared in Phase 1)

**7. High-Level Architecture**

📦 **Architecture Layers:**

1. **Data Ingestion**: ADLS Raw → Bronze via ADF
2. **Transformation**: Databricks ETL → Silver
3. **API Layer**: FastAPI + SQL VM
4. **Analytics**: Databricks → Gold Layer
5. **Access & Monitoring**: Role Simulation + Key Vault

*Architecture diagram prepared in Phase 1*

**8. Assumptions & Constraints**

* All services are deployed in the same Azure region
* Data volume < 10 million records initially
* APIs will be used internally; no public exposure
* VM specs: 4 vCPUs, 16GB RAM minimum

**9. Future Enhancements**

* Integrate Azure Event Hub for real-time order updates
* Add CI/CD with GitHub Actions or Azure DevOps
* Add Azure AD integration for real access control
* Enable multi-tenant SaaS deployment