Software Requirements Specification (SRS)

Java Spring Boot Microservices Project

Document Version: 1.0

Date: May 13, 2025

Timeline: 10-Day Implementation Plan

---

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document outlines the requirements for a comprehensive Java-based e-commerce platform consisting of multiple microservices. The project is designed to test and demonstrate proficiency across a wide range of Java and Spring technologies.

1.2 Scope

The system will consist of four primary microservices (Employee, Product, Order, and Notification) that communicate via REST APIs and event-driven messaging. The project incorporates Java 11 features, Spring Boot, Hibernate/JPA, Kafka messaging, and Google Cloud Platform integrations.

1.3 Technology Stack

- Java 11
- Spring Boot 2.7.x
- Spring Framework (Core, MVC, Data)
- Hibernate/JPA
- Maven & Gradle
- Kafka
- GCP Spanner & BigQuery
- JUnit 5 & Mockito
- Git

---

2. System Architecture

2.1 High-Level Architecture

The system follows a microservices architecture pattern with the following components:

Show Image

2.2 Component Descriptions

2.2.1 Employee Service

HR management system handling employee data, departments, and payroll information.

2.2.2 Product Service

Inventory and catalog management system with product details, pricing, and availability.

2.2.3 Order Service

Order processing system handling customer orders, payment processing, and order fulfillment.

2.2.4 Notification Service

Event-driven notification system delivering alerts via various channels based on system events.

2.3 Database Architecture

- Employee Service: JPA/Hibernate with relational database
- Product Service: GCP Spanner
- Order Service: JPA/Hibernate with relational database
- Analytics: BigQuery

2.4 Communication Patterns

- Synchronous: REST APIs (RestTemplate, WebClient, Feign)
- Asynchronous: Kafka messaging

---

3. Detailed Requirements

3.1 Employee Service Requirements

3.1.1 Domain Model

- Employee entity with proper OOP principles (inheritance, encapsulation, etc.)
- Department entity with one-to-many relationship to employees
- Address as a component class
- Payroll as an associated entity

3.1.2 API Endpoints

- `GET /api/employees` - List all employees with pagination
- `GET /api/employees/{id}` - Get employee details
- `POST /api/employees` - Create a new employee
- `PUT /api/employees/{id}` - Update employee details
- `DELETE /api/employees/{id}` - Remove an employee
- `GET /api/employees/department/{departmentId}` - List employees by department
- `GET /api/departments` - List all departments

3.1.3 Technical Requirements

- Implement proper exception handling with custom exceptions
- Use Java 11 Stream API for data processing
- Implement File I/O for report generation (PDF, CSV)
- Configure Spring profiles for dev, test, and prod environments
- Proper DTO pattern implementation
- Comprehensive unit and integration testing

3.2 Product Service Requirements

3.2.1 Domain Model

- Product entity with variants
- Category hierarchy (tree structure)
- Inventory tracking
- Price history

3.2.2 API Endpoints

- `GET /api/products` - List all products with filtering and pagination

- `GET /api/products/{id}` - Get product details
- `POST /api/products` - Create a new product
- `PUT /api/products/{id}` - Update product details
- `DELETE /api/products/{id}` - Remove a product
- `GET /api/products/category/{categoryId}` - List products by category
- `GET /api/categories` - List all categories

3.2.3 Technical Requirements

- Integration with GCP Spanner
- Implementation of caching strategies
- BigQuery integration for analytics
- Custom native queries for complex reporting

3.3 Order Service Requirements

3.3.1 Domain Model

- Order entity with line items
- Order status tracking
- Payment information
- Shipping details

3.3.2 API Endpoints

- `GET /api/orders` - List all orders with filtering
- `GET /api/orders/{id}` - Get order details
- `POST /api/orders` - Create a new order
- `PUT /api/orders/{id}/status` - Update order status
- `POST /api/orders/{id}/payment` - Process payment for an order
- `GET /api/orders/customer/{customerId}` - List orders by customer

3.3.3 Technical Requirements

- Transaction management
- Integration with Product Service via RestTemplate/WebClient

- Kafka producer for order events
- Proper concurrency handling
- Idempotent API design

3.4 Notification Service Requirements

3.4.1 Supported Notification Types

- Order status updates
- Inventory alerts
- Employee onboarding notifications
- System alerts

3.4.2 Technical Requirements

- Kafka consumer implementation
- Notification template system
- Delivery channel abstraction (email, SMS, etc.)
- Retry mechanisms for failed notifications

---

4. Cross-Cutting Concerns

4.1 Security

- Basic authentication for service-to-service communication
- Proper input validation
- CORS configuration

4.2 Logging and Monitoring

- Centralized logging
- Performance metrics
- Health check endpoints

4.3 Error Handling

- Consistent error responses
- Global exception handlers
- Proper HTTP status code usage

4.4 Testing Requirements

- Unit tests with JUnit 5
- Mock testing with Mockito
- Integration tests
- API tests

---

5. Implementation Plan

5.1 Phase 1: Foundation & Employee Service (Days 1-3)

Day 1: Project Setup & Core Java

- **Morning**:
  - Set up development environment
  - Create Git repository with proper branching strategy
  - Initialize Employee Service with Spring Boot
  - Implement core domain models using Java 11 OOP concepts
- **Afternoon**:
  - Implement custom exceptions and handling strategy
  - Create utility classes using Stream API and Lambda expressions
  - Set up Maven build configuration with proper dependency management

Day 2: Employee Service Development

- **Morning**:
  - Develop REST controllers with proper HTTP verbs and status codes
  - Implement Spring IoC, DI patterns with appropriate bean scopes
  - Configure application properties and profiles
- **Afternoon**:
  - Create service layer with business logic
  - Implement repository layer with Spring Data JPA
  - Set up Hibernate entity mappings and relationships

Day 3: Testing & Completion of Employee Service

- **Morning**:

    - Implement JUnit 5 tests for all layers
    - Use Mockito for service and controller testing
    - Add integration tests
- **Afternoon**:
    - Implement advanced File I/O operations for report generation
    - Add custom query methods using JPA and native queries
    - Document API endpoints

5.2 Phase 2: Product Service & Database Integration (Days 4-5)

Day 4: Product Service & GCP Integration

- **Morning**:
    - Initialize Product Service with Gradle
    - Configure Spring Data to work with GCP Spanner
    - Implement domain models and repositories
- **Afternoon**:
    - Create REST controllers and service layer
    - Implement Hibernate second-level caching
    - Configure lazy loading strategies

Day 5: Advanced Database Features

- **Morning**:
    - Implement native queries for complex reporting
    - Set up BigQuery integration for analytics
    - Create data migration utilities using NIO
- **Afternoon**:
    - Add exception handling with Spring's @ControllerAdvice
    - Implement custom validators
    - Set up comprehensive testing suite

5.3 Phase 3: Microservices Architecture & Kafka (Days 6-8)

Day 6: Microservices Communication

- **Morning**:

    - Set up service discovery
    - Implement inter-service communication using RestTemplate
    - Add WebClient for reactive API calls
- **Afternoon**:
    - Configure Feign clients
    - Implement circuit breaker patterns
    - Set up API gateway routing

Day 7: Kafka Integration

- **Morning**:
    - Set up Kafka environment
    - Create producers and consumers using CLI
    - Implement message schemas
- **Afternoon**:
    - Integrate Spring Kafka
    - Configure JSON serialization/deserialization
    - Implement error handling for Kafka consumers

Day 8: Order & Notification Services

- **Morning**:
    - Develop Order Service with REST endpoints
    - Implement order processing workflow
    - Create order repository with transaction management
- **Afternoon**:
    - Develop Notification Service
    - Configure Kafka listeners
    - Implement notification delivery mechanisms

5.4 Phase 4: Advanced Features & Project Completion (Days 9-10)

Day 9: Advanced Features

- **Morning**:
    - Implement advanced Hibernate mappings

    - Configure caching strategies
    - Add performance monitoring
- **Afternoon**:
    - Implement functional interfaces for business rules
    - Add comprehensive logging
    - Create data export features using Java I/O streams

Day 10: Final Integration & Testing

- **Morning**:
    - End-to-end testing across all services
    - Performance testing and optimization
    - Fix any remaining issues
- **Afternoon**:
    - Finalize documentation
    - Prepare demonstration scenarios
    - Create project README with setup instructions

---

6. Knowledge Assessment Coverage

6.1 Java Fundamentals

- Java 11 OOP Concepts (inheritance, polymorphism, encapsulation, abstraction)
- Exception Handling (custom exceptions, try-with-resources)
- File I/O (Streams, Readers/Writers, NIO, Post-Java 8 APIs)
- Collections Framework
- Lambda Expressions & Functional Interfaces
- Stream API

6.2 Git & Build Tools

- Git Workflow (clone, branch, commit, merge)
- Maven Configuration & Dependency Management
- Gradle Configuration & Usage

6.3 Spring Framework & Spring Boot

- Dependency Injection & Inversion of Control
- Bean Configuration & Lifecycle
- Properties & Profiles
- Spring Boot Auto-configuration
- Spring Boot DevTools
- REST Controller Implementation
- Exception Handling in Spring Boot

6.4 Database & ORM

- Hibernate Entity Mapping
- JPA Repository Pattern
- Spring Data JPA
- Hibernate Internals (caching, lazy loading, etc.)
- Native Queries
- GCP Spanner Integration
- BigQuery Basics

6.5 Microservices

- Service Communication (RestTemplate, WebClient, Feign)
- Event-Driven Architecture
- Kafka Basics (Producer/Consumer)
- Spring Kafka Integration
- JSON Message Processing

6.6 Testing

- JUnit 5 Features
- Mockito for Unit Testing
- Integration Testing
- Test Coverage

---

7. Deliverables

7.1 Source Code

- Properly structured Git repository
- Documentation (JavaDoc, README, etc.)
- Complete build configuration

7.2 Testing Artifacts

- Unit tests
- Integration tests
- Test reports

7.3 Documentation

- API documentation
- Architecture documentation
- Setup instructions

---

8. Success Criteria

8.1 Functional Criteria

- All services function as specified
- Services properly communicate with each other
- Event-driven messaging works correctly

8.2 Technical Criteria

- Code demonstrates proper usage of all required technologies
- Tests cover critical functionality
- Services are properly documented
- Code follows best practices and patterns

---

9. Appendix

9.1 Technical Topic Coverage Matrix

Topic	Implementation Area
Java 11 OOP Concepts	Domain models in all services
File I/O	Report generation in Employee Service
Exception Handling	Custom exceptions, global handlers
Git Workflow	Repository structure and branching strategy
Collections, Lambda, Stream API	Utility classes, data processing
RESTful Services	All service controllers
Maven & Gradle	Employee Service (Maven), Product Service (Gradle)
Spring DI, IoC, Beans	Service layers across all components
Spring Boot Basics	All services configuration
CRUD API Layers	All services
Hibernate + Spring Data JPA	Employee and Order services
Exception Handling in Spring Boot	@ControllerAdvice in all services
Spring Boot Testing	Test suites for all services
Microservices Communication	Service-to-service interactions
Kafka Basics	Message producers and consumers
Spring Kafka Integration	Notification service
GCP Spanner	Product service data store
Hibernate Internals	Caching and optimization strategies
Native Queries + BigQuery	Reporting features

9.2 Mini-Project Coverage

9.2.1 Mini Project 1: Employee Service

Comprehensive HR management system demonstrating:

- CRUD operations
- File handling
- OOP principles
- Spring Boot fundamentals
- Exception handling
- Testing

9.2.2 Mini Project 2: Order + Notification Services

Event-driven order processing system demonstrating:

- Kafka messaging
- Microservices communication
- Transaction management
- Event handling