***ShopSense*  
Mall Sales and Inventory Management Software**

**Project Overview**

This project is designed to develop a comprehensive mall sales and inventory management software using Java (Spring Boot), Python, and Svelte. The software aims to streamline inventory tracking, sales processing, and reporting. It will use Agile Software Development Life Cycle (SDLC) to ensure iterative and flexible development.

**Tech Stack**

**Backend**

* **Java (Spring Boot):** Core backend functionalities including API development, database interactions, and business logic.
* **Python:** Data analytics, reporting, machine learning, and automation tasks.

**Frontend**

* **Svelte:** Interactive, fast, and modern user interfaces.

**Database**

* **MySQL/PostgreSQL:** Relational database for data storage.

**Other Tools**

* **Docker:** Containerization for deployment.
* **Postman:** API testing.
* **Swagger UI:** API documentation.
* **Git:** Version control.

**Key Features**

**Inventory Management**

* Track stock levels in real-time.
* Categorize products based on type, supplier, and price.
* Automated restocking alerts.

**Sales Tracking**

* Record sales transactions.
* Generate receipts and invoices.
* Monitor sales trends.

**Reporting and Analytics**

* Generate visual reports (Python with Matplotlib/Seaborn).
* Predict demand using machine learning (Python with Scikit-learn).

**User Management**

* Role-based access control (e.g., admin, cashier, manager).
* Secure login and registration (Spring Security).

**Interactive Dashboard**

* Real-time updates and visualizations (Python Dash integrated with Svelte).

**Project Workflow**

**1. Planning Phase**

* **Gather Requirements:** Collaborate with stakeholders to define features.
* **Define Sprint Goals:** Break down tasks into manageable sprints.
* **Tool Setup:** Configure development environments for Java, Python, and Svelte.

**2. Development Phase**

**Backend**

1. **Set Up Spring Boot Project:**
   * Use Spring Initialize with dependencies (Spring Web, Spring Data JPA, Spring Security, etc.).
2. **Database Design:**
   * Design schema for products, sales, users, etc.
3. **API Development:**
   * Create RESTful endpoints for inventory, sales, and user management.
4. **Integrate Python:**
   * Set up Flask/Fast API microservices for analytics and reporting.
   * Use RabbitMQ/Kafka for communication between Java and Python.

**Frontend**

1. **Set Up Svelte Project:**
   * Configure Svelte for dynamic UI.
2. **Connect to Backend:**
   * Use Fetch API or Axios to consume Spring Boot APIs.
3. **Interactive Dashboards:**
   * Embed Python Dash apps into Svelte components.

**3. Testing Phase**

* Unit Testing (JUnit/Mockito for Java, Pytest for Python).
* API Testing (Postman).
* End-to-End Testing (Svelte + Cypress).

**4. Deployment Phase**

* Containerize the application using Docker.
* Set up CI/CD pipeline (GitHub Actions/Jenkins).
* Deploy to AWS/GCP/Azure.

**System Architecture**

Frontend (Svelte) → Backend (Spring Boot) → Database (MySQL/PostgreSQL)

↘ ↗

Python Services (Analytics, ML)

* **Svelte:** User interface, interacts with backend APIs.
* **Spring Boot:** Central business logic and data handling.
* **Python Services:** Handles analytics, machine learning, and automation tasks.
* **Database:** Stores all data securely.

**Module Details**

**1. Inventory Module**

* **Backend:**
  + Spring Boot APIs for CRUD operations on products.
  + Python scripts for inventory analysis.
* **Frontend:**
  + Svelte components for adding/editing products.
  + Visual indicators for low-stock products.

**2. Sales Module**

* **Backend:**
  + APIs for recording and retrieving sales transactions.
  + Python scripts for sales trend analysis.
* **Frontend:**
  + Svelte components for point-of-sale (POS) interface.

**3. Reporting Module**

* **Backend:**
  + Python for generating charts and predictions.
  + Flask/Fast API for serving analytics APIs.
* **Frontend:**
  + Embed Python Dash apps into Svelte UI.

**Integration Points**

**1. Java ↔ Python**

* **REST API Integration:** Java calls Python APIs for analytics.
* **Message Queues:** Use RabbitMQ or Kafka for asynchronous communication.

**2. Backend ↔ Frontend**

* Use REST APIs to connect Spring Boot with Svelte.
* Handle authentication using JWT or OAuth.

**3. Data Sharing**

* Use JSON for exchanging data between components.
* Store generated reports in the database for UI access.
* **├   
  Structure  
  src/ main/java/com/**

controller

-InventoryController.java

- SalesController.java

- UserController.java

Service

- InventoryService.java

- SalesService.java

- UserService.java

repository

- InventoryRepository.java

- SalesRepository.java

- UserRepository.java

model

- Product.java

- Sale.java

- User.java

dto

- InventoryDTO.java

- SaleDTO.java

- UserDTO.java

config

- SecurityConfig.java

- SwaggerConfig.java

- AppConfig.java

exception

- ResourceNotFoundException.java

- InvalidInputException.java

- GlobalExceptionHandler.java

**├ src/ main/python/**

analytics # Contains modules for data analysis and reporting

- \_\_init\_\_.py

- sales\_analysis.py

- inventory\_analysis.py

- report\_generator.py

ml\_models # Contains machine learning models and utilities

- \_\_init\_\_.py

- demand\_prediction.py

-model\_training.py

api # Contains Flask/FastAPI API endpoints for analytics

- \_\_init\_\_.py

- analytics\_api.py

- reporting\_api.py

utils # Utility functions and helpers used across the project

- \_\_init\_\_.py

- data\_loader.py

- data\_visualization.py

config # Configuration settings

- \_\_init\_\_.py

- settings.py

tests # Unit tests for the Python modules

- \_\_init\_\_.py

- test\_sales\_analysis.py

- test\_inventory\_analysis.py

- test\_reporting\_api.py

-

README.md # Project documentation or description file

**Documentation**

* **API Docs:** Use Swagger UI for backend API documentation.
* **Code Docs:** Use Javadoc and Python docstrings.
* **User Manual:** Create a detailed guide for system users.

**Conclusion**

This project combines the strengths of Java, Python, and Svelte to build a robust and feature-rich mall management software. By following Agile SDLC, the team can deliver iterative improvements and adapt to changing requirements efficiently.