Smart-IMS Requirements Document

CS4092 Final Project – Shivam Sinay Kharangate

Project Overview

Smart-IMS is an AI-powered inventory management system that enables natural language queries to interact with inventory data through LLM-generated SQL queries.

Data Requirements

Core Data Entities

1. **Categories**
   - Purpose: Organize products into logical groups
   - Fields:
     - ID (Primary Key)
     - Name (Electronics, Clothing, Home & Garden, Sports, Books)

2. **Products**
   - Purpose: Catalog of manageable items
   - Fields:
     - ID (Primary Key)
     - Name
     - Category ID (Foreign Key)
     - Price (Float)
     - Reorder Level (Integer threshold for low stock alerts)

3. **Warehouses**
   - Purpose: Physical storage locations
   - Fields:
     - ID (Primary Key)
     - Location (Address/Name of warehouse)

4. **Inventory**
   - Purpose: Track product quantities at specific warehouses
   - Fields:
     - Product ID + Warehouse ID (Composite Primary Key)
     - Quantity (Current stock level)

5. **Suppliers**
   - Purpose: Vendors contact information

- Fields:
    - ID (Primary Key)
    - Name
    - Contact (Email/Phone)

Data Relationships

- Products belong to Categories (Many-to-One)
- Inventory links Products to Warehouses (Many-to-Many)
- Each Inventory record represents stock level of a specific product at a specific warehouse

Use Cases

1. Natural Language Inventory Queries
    - **User Story:** "As a user, I want to ask inventory questions in plain English"
    - **Examples:**
        - "Show me all products with their categories and stock levels"
        - "Which items are below their reorder level?"
        - "What's the total inventory value per warehouse?"
        - "Find all electronics products in the main warehouse"

2. Stock Level Management
    - **User Story:** " As an inventory manager, I need to monitor and update stock levels"
    - **Functions:**
        - View current inventory across all warehouses
        - Add/update inventory quantities
        - Track products below reorder thresholds
        - Generate low stock alerts

3. Multi-Warehouse Operations
    - **User Story:** "As a business owner, I need visibility across multiple locations"
    - **Functions:**
        - Compare stock levels between warehouses
        - Calculate total inventory value per location
        - Transfer tracking between warehouses

4. Reporting and Analytics
    - **User Story:** "As management, I need inventory insights for decision making"
    - **Functions:**
        - Inventory valuation reports
        - Category-wise stock analysis
        - Reorder recommendations
        - Warehouse utilization metrics

5. Administrative Operations
    - **User Story:** "As an admin, I need direct database access for maintenance"
    - **Functions:**
        - Execute raw SQL queries
        - Database schema inspection
        - Bulk data operations
        - System health monitoring

Technical Requirements

1. AI Integration
    - Natural language processing via Ollama (Gemma 3 model)
    - Text-to-SQL conversion capability
    - Model Context Protocol (MCP) for LLM-database bridge

2. Database Requirements
    - PostgreSQL for data persistence
    - Support for complex JOINs and aggregations
    - ACID compliance for inventory transactions
    - Foreign key constraints for data integrity

3. API Requirements
    - RESTful endpoints for all operations
    - Natural language query processing
    - Direct SQL execution capability
    - JSON response format
    - CORS support for web interface

4. User Interface Requirements
    - Web-based interface for testing and demonstration
    - Natural language query input
    - Tabular result display
    - Quick action buttons for common operations
    - Real-time API status monitoring

Performance Requirements

- Sub-second response time for simple queries
- Support for concurrent users
- Efficient query execution on multi-table JOINs
- Real-time inventory updates

Security Requirements

- Environment-based database credentials
- SQL injection protection via parameterized queries
- API input validation
- Database connection pooling

Integration Requirements

- Ollama LLM service integration
- MCP client-server architecture
- PostgreSQL database connectivity
- FastAPI web framework
- Cross-platform compatibility (Windows/Linux/Mac)

References

Microsoft Visual Studio Code – GitHub Copilot: Utilized for data collection for project deliverables to assess content information and summarize notes for better understanding.