Software Requirements Specification (SRS) for Import Export ERP System

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document outlines the functional and non-functional requirements for an Import Export ERP (Enterprise Resource Planning) system. The system is designed to facilitate import and export operations for sellers. Users register as sellers, manage products, orders, and reports, and perform profile management. The system emphasizes secure authentication and efficient data management for import-export activities, enhanced with AI, ML, and chatbot functionalities for intelligent insights, predictive analytics, and interactive user support.

The primary goals are:

- Enable user registration and login with redirection to seller dashboard.
- Provide dashboards for sellers to manage products, orders, reported products, and profiles.
- Support order placement and reporting from external sources (e.g., via API or manual entry, out of scope for user modules).
- Integrate AI for automated data analysis, ML for predictive modeling (e.g., demand forecasting), and chatbots for real-time assistance.
- Use fake data for testing and demonstration purposes where real data integration is not required (e.g., initial reported products).

1.2 Scope

The system includes:

- User registration and authentication for sellers.
- Seller dashboards to manage products, orders, reported products, and profiles.
- Product management (add, update, delete products).
- Order management (create, view, update orders).
- Reported product management (view and handle reports on products/orders).
- Profile management (update or delete user profiles).
- AI-driven analytics for insights on products and orders.
- ML models for forecasting and optimization.
- Chatbot integration for user queries and support.
- Database integration for storing user, product, order, and report data.

- **Fake data** population for development and testing.

Out of scope:

- Advanced payment processing (e.g., integration with payment gateways).
- Real-time shipping tracking.
- Multi-language support.
- Mobile responsiveness beyond Bootstrap basics.
- External buyer registration or interfaces.
- Training of ML models (assumed pre-trained or simple implementations).
- External AI/ML services (e.g., cloud APIs; use in-house implementations).

1.3 Definitions, Acronyms, and Abbreviations

- **ERP**: Enterprise Resource Planning.
- **PK**: Primary Key.
- **FK**: Foreign Key.
- **MVC2**: Model-View-Controller design pattern (version 2, emphasizing servlet-based controllers).
- **JSP**: JavaServer Pages.
- **JDBC**: Java Database Connectivity.
- **Port ID**: Unique identifier for users (e.g., a port-related code for import-export context).
- **AI**: Artificial Intelligence.
- **ML**: Machine Learning.

1.4 References

- Technologies: HTML, CSS, Bootstrap (for frontend); Servlet, JDBC (for middleware); MySQL (for backend). Additional integrations for AI/ML/chatbots via Java-compatible libraries (e.g., Weka for ML, simple rule-based chatbots in JSP/Servlet).
- Design Pattern: MVC2.
- Database Schema: As defined in Section 3.4.

1.5 Overview

The document is structured as follows:

- Section 2: Overall description of the product.
- Section 3: Specific requirements.

- Section 4: Supporting information.

2. Overall Description

2.1 Product Perspective

This ERP system is a web-based application for managing import-export transactions for sellers. It focuses on sellers (exporters/importers managing inventory and orders from external buyers), augmented with AI for intelligent processing, ML for data-driven decisions, and chatbots for enhanced user interaction.

2.2 Product Functions

- **User Registration:** Collect port_id (PK), password, location, name, email. Redirect to login upon success.
- **User Login:** Authenticate using port_id and password. Redirect to seller dashboard.
- **Seller Dashboard:**
    - Product Management: Add, update, delete products.
    - Order Management: View, update orders (orders assumed from external buyers).
    - Reported Product Management: View and handle reported products (using fake data initially).
    - Profile Management: Update or delete profile.
    - AI Insights: Generate automated reports and anomalies detection.
    - ML Predictions: Forecast demand, optimize inventory.
    - Chatbot Assistance: Interactive query handling for dashboard features.
- **Data Management:** Use MySQL for persistence, with JDBC for connectivity.

2.3 User Classes and Characteristics

- **Sellers:** Experienced in import-export, need tools for inventory and order tracking, enhanced by AI/ML insights and chatbot support.
- **Administrators:** Not explicitly defined; assumed no admin role in this version.

2.4 Operating Environment

- **Frontend:** Web browsers (Chrome, Firefox) supporting HTML5, CSS3, Bootstrap 5+.
- **Middleware:** Java Servlets running on Apache Tomcat or similar.
- **Backend:** MySQL 8+ database server.
- **Development Tools:** Eclipse or IntelliJ for Java development.
- **AI/ML Environment:** Java-based libraries for integration.

2.5 Design and Implementation Constraints

- **Design Pattern**: MVC2 (Model: Database/JDBC with AI/ML logic; View: JSP/HTML/CSS/Bootstrap with chatbot interfaces; Controller: Servlets).
- **Technologies**: Strictly adhere to specified stack; no additional frameworks (e.g., no Spring). AI/ML implemented via basic Java algorithms or libraries like Weka; chatbots as rule-based systems in servlets/JSP.
- **Security**: Passwords hashed in database; basic session management via servlets.
- **Fake Data**: Populate tables with sample entries for testing (e.g., reported products, ML training data).

2.6 Assumptions and Dependencies

- Users have valid email for registration verification (though not implemented).
- Port_id is unique and generated/validated during registration.
- Fake data will be inserted via SQL scripts during setup.
- Orders and reports can originate from external sources (e.g., manual entry by sellers).
- AI/ML features use historical data from the database; no real-time external data feeds.
- Chatbots handle predefined queries; advanced NLP out of scope.

3. Specific Requirements

3.1 External Interface Requirements

- **User Interfaces**: Web pages using JSP for dynamic content, styled with Bootstrap for responsiveness.
    - Registration Page: Form fields for port_id, password, location, name, email.
    - Login Page: Fields for port_id, password.
    - Seller Dashboard: Tabs/sections for products, orders, reports, profile, AI insights, ML forecasts, chatbot widget.
- **Hardware Interfaces**: None (web-based).
- **Software Interfaces**: JDBC for MySQL connectivity; potential API calls for chatbot logic.
- **Communication Interfaces**: HTTP/HTTPS for web access.

3.2 Functional Requirements

3.2.1 User Registration

- **Input**: port_id (string, unique PK), password (string, min 8 chars), location (string), name (string), email (valid format).
- **Process**: Validate uniqueness of port_id and email. Insert into users table. On success, redirect to login.jsp.
- **Output**: Success message or error (e.g., "Port ID already exists").
- **Error Handling**: Validate fields; prevent SQL injection via prepared statements.

3.2.2 User Login

- **Input**: port_id, password.
- **Process**: Authenticate against users table. On success, set session and redirect to seller_dashboard.jsp.
- **Output**: Dashboard or error message.
- **Error Handling**: Invalid credentials redirect back to login with message.

3.2.3 Seller Dashboard - Product Management

- **Functions**: Add (form for product_name, description, quantity, price; auto-set seller_port_id, created_at/updated_at), Update/Delete (list products, edit/delete via ID).
- **Input/Output**: Forms and tables displaying products from product table.

3.2.4 Seller Dashboard - Order Management

- **Functions**: View orders (filtered by seller_port_id), Update status (e.g., pending, shipped, delivered).
- **Input/Output**: Table of orders; form to update status.

3.2.5 Seller Dashboard - Reported Product Management

- **Functions**: View reported products (list from reported_products table; use fake data initially). Handle reports (e.g., mark resolved).
- **Input/Output**: Table of reports; basic actions.

3.2.6 Seller Dashboard - Profile Management

- **Functions**: Update (edit location, name, email, password), Delete (remove user and cascade related data).
- **Input/Output**: Form pre-filled with user data.

3.2.7 Seller Dashboard - AI Insights

- **Functions:** Use AI algorithms to analyze data (e.g., detect anomalies in orders or products). Generate summaries or alerts.
- **Input/Output:** Dashboard section displaying AI-generated insights, based on database queries.

3.2.8 Seller Dashboard - ML Predictions

- **Functions:** Apply ML models (e.g., simple regression) to forecast demand, suggest inventory levels based on historical orders.
- **Input/Output:** Forms to input parameters; output charts or predictions integrated via JSP.

3.2.9 Seller Dashboard - Chatbot Assistance

- **Functions:** Embed a chatbot widget for natural language queries (e.g., "Show my recent orders"). Rule-based responses pulling from database.
- **Input/Output:** Chat interface in dashboard; responses via servlet-processed queries.

3.3 Non-Functional Requirements

- ***Performance:*** **Page load < 2 seconds; handle up to 100 concurrent users; AI/ML computations < 5 seconds.**
- ***Security:*** **Encrypt passwords; access control; secure AI/ML data handling.**
- **Reliability**: 99% uptime; backup database weekly.
- **Usability**: Intuitive UI with Bootstrap; error messages in red; responsive chatbot.
- **Scalability**: Design for easy addition of features (e.g., more AI/ML models).

3.4 Database Requirements

- **Users Table**:
  - port_id (VARCHAR(50), PK)
  - password (VARCHAR(255))
  - location (VARCHAR(100))
  - name (VARCHAR(100))
  - email (VARCHAR(100))
- **Product Table**:
  - product_id (INT, AUTO_INCREMENT, PK)
  - seller_port_id (VARCHAR(50), FK references users.port_id)
  - product_name (VARCHAR(100))

- description (TEXT)
- quantity (INT)
- price (DECIMAL(10,2))
- created_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
- updated_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP)

- **Orders Table**:
  - order_id (INT, AUTO_INCREMENT, PK)
  - buyer_id (VARCHAR(50))
  - seller_port_id (VARCHAR(50), FK references users.port_id)
  - order_date (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
  - total_amount (DECIMAL(10,2))
  - status (ENUM('pending', 'shipped', 'delivered', 'cancelled'))
  - delivery_address (VARCHAR(200))
- **Reported Products Table:**
  - report_id (INT, AUTO_INCREMENT, PK)
  - product_id (INT, FK references product.product_id)
  - reporter_id (VARCHAR(50))
  - reason (TEXT)
  - reported_at (TIMESTAMP, DEFAULT CURRENT_TIMESTAMP)
  - status (ENUM('open', 'resolved'))