# Project Definition and Software Requirements Specification (SRS)

Project: SmartTechMart — Tech Convenience Store 4.0

Author: Generated by ChatGPT  
Date: 2025-10-08  
Version: 1.0

## Lab 1 – Project Initialization & Requirements Definition

This document contains the Project Definition and the Software Requirements Specification (SRS) for the mini-project: SmartTechMart — a retail application for a technology convenience store (Tech Store 4.0).

## Project Definition

### 1. Project Title

SmartTechMart — Tech Convenience Store 4.0

### 2. Background / Context

Small technology retail and convenience stores increasingly need digital tools to manage stock, sales, customer loyalty, and online presence. SmartTechMart is a modern retail management application designed for small/medium tech convenience stores to handle point-of-sale (POS), inventory, supplier orders, basic e-commerce features, and analytics in a lightweight, affordable system.

### 3. Objectives

- Provide a simple, reliable POS for in-store sales (barcode scanning, quick checkout).  
- Manage inventory with real-time stock levels, reorder alerts, and purchase orders.  
- Support customer records, basic loyalty points, and promotional campaigns.  
- Offer management dashboards and daily/weekly/monthly sales reports.  
- Enable online catalog browsing and click-and-collect for customers (optional MVP feature).

### 4. Scope

In scope:

- POS module (sales, returns).  
- Inventory management (products, categories, stock levels, suppliers).  
- User accounts and role-based access (Admin, Manager, Cashier).  
- Basic reporting and exports (CSV/PDF).  
- Simple promotions and discount rules.  
- Backup and data export.

Out of scope (for Lab 1 / MVP):

- Full-scale e-commerce checkout/payment gateway integrations beyond basic QR/card payments.  
- Advanced CRM features or marketing automation.  
- Multi-store synchronization (only single store supported initially).

### 5. Stakeholders

- Store Owner / Admin: project sponsor, receives management reports and admin controls.  
- Store Manager: daily operations, inventory management, ordering.  
- Cashier: operates POS for sales.  
- Customers: shop in-store and browse catalog for click-and-collect.  
- Developers & QA: implement and test the system.

### 6. Constraints

- Budget and timeline constraints for a semester mini-project.  
- Use of open-source libraries and low-cost hosting.  
- Must run in modern browsers; optional mobile-friendly layout.  
- Data privacy rules (basic compliance with local data protection).

### 7. Assumptions

- The store has an internet connection (some offline caching possible).  
- Barcode scanner available (keyboard wedge or USB HID).  
- Payment handling will be manual (card terminal) or via simple QR for MVP.

### 8. Deliverables

- Project Definition document and SRS (this file).  
- Basic working prototype: POS + Inventory + Reports.  
- Source code in a GitHub repository with README and instructions.  
- Demo video or screenshots and test data.

### 9. High-Level Timeline (suggested)

- Week 1: Project definition, SRS, repo setup, initial UI mockups.  
- Week 2–3: Basic data model and inventory module implementation.  
- Week 4: POS flows and checkout.  
- Week 5: Reporting, user roles, testing.  
- Week 6: Buffer, documentation, demo preparation.

## Software Requirements Specification (SRS)

### 1. Introduction

1.1 Purpose

This SRS describes the functional and non-functional requirements for SmartTechMart. It is intended for the development team, the project stakeholders, and the testers.

1.2 Document Conventions

Requirement IDs use the format FR-### for functional requirements and NFR-### for non-functional requirements.

1.3 Intended Audience and Reading Suggestions

Stakeholders, developers, testers, and documentation authors should read this document. Technical staff should focus on the functional and data sections.

### 2. Overall Description

2.1 Product Perspective

SmartTechMart is a standalone retail management web application (progressive web app recommended) that can run on a local server or cloud host. It integrates with hardware devices such as barcode scanners and receipt printers, and optionally can integrate with payment terminals and simple QR payment gateways.

2.2 Product Functions (high-level)

- Manage product catalog and inventory.  
- Process sales and returns via POS.  
- Generate sales and inventory reports.  
- Support user accounts and role-based permissions.  
- Manage suppliers and purchase orders.  
- Support promotions/discounts and simple loyalty.

2.3 User Classes and Characteristics

- Admin: full access to system settings and user management.  
- Manager: access to inventory, orders, reports.  
- Cashier: access to POS and basic customer lookup.  
- Customer: interacts with optional catalog interface.

2.4 Operating Environment

Modern web browsers (Chrome, Edge, Firefox, Safari). Server: Linux or Windows hosting with Node.js / Python / PHP stack (team choice).

2.5 Design and Implementation Constraints

- Use open-source components.  
- Keep UI responsive and simple.  
- Data persistence via relational database (SQLite/Postgres) for the MVP.

2.6 Assumptions and Dependencies

- Availability of barcode scanner hardware and a network connection for cloud-hosted deployments.  
- Third-party payment gateways not required for MVP.

### 3. Specific Requirements

3.1 Functional Requirements (FR)

1. FR-001 — User Authentication & Authorization

Allow users to register (Admin only), login, reset password; support roles: Admin, Manager, Cashier. Priority: High.

1. FR-002 — Product Catalog Management

Add, edit, delete products with fields: SKU, name, description, category, cost price, selling price, barcode, image, supplier. Priority: High.

1. FR-003 — Inventory Tracking

Track stock levels per product; record stock-in and stock-out events; show low-stock alerts and reorder thresholds. Priority: High.

1. FR-004 — Point-of-Sale (POS) Sales

Process sales with multiple items, apply discounts, accept payment type (cash/card/QR), generate receipt. Priority: High.

1. FR-005 — Returns & Refunds

Support processing returns and generating refund records. Priority: Medium.

1. FR-006 — Supplier & Purchase Orders

Create supplier records and purchase orders, receive shipments to update inventory. Priority: Medium.

1. FR-007 — Promotions & Discounts

Configure percentage or fixed discounts by product or cart-level promotions. Priority: Medium.

1. FR-008 — Customer Records & Loyalty

Store customer info, track purchases, and apply loyalty points for discounts. Priority: Low/Optional.

1. FR-009 — Reporting & Analytics

Generate daily sales, inventory valuation, best-selling products reports; export CSV/PDF. Priority: High.

1. FR-010 — Data Backup & Export

Allow manual export and backup of database; import CSV for product bulk upload. Priority: Medium.

1. FR-011 — Audit Log

Record user actions (login, sales, inventory adjustments) for auditing. Priority: Medium.

1. FR-012 — Offline Mode (Optional)

Cache recent catalog and allow POS transactions offline with later sync. Priority: Low/Optional.

### 3.2 Non-Functional Requirements (NFR)

1. NFR-001 — Performance

System must handle up to 20 concurrent cashier sessions and respond to POS item scan within 1 second under typical load.

1. NFR-002 — Availability

System availability 99.5% (excluding maintenance windows).

1. NFR-003 — Security

Passwords must be hashed (e.g., bcrypt). Sensitive data in transit must be encrypted via HTTPS. Access control enforced by roles.

1. NFR-004 — Usability

UI should be intuitive; cashier workflow must allow checkout in under 60 seconds for typical transactions.

1. NFR-005 — Scalability

Design should allow future multi-store support and modular integration.

1. NFR-006 — Maintainability

Codebase should follow standard conventions, documented API, and automated tests for core flows.

1. NFR-007 — Localization

Support for English and local language (Vietnamese) in UI strings (optional).

### 4. Data Requirements

Main entities: Product, Category, Supplier, Customer, User, Sale, SaleItem, PurchaseOrder, InventoryTransaction.  
Data retention: sales records retained for at least 2 years; backups created regularly.  
Personal customer data must follow privacy rules and not be shared without consent.

### 5. Use Cases (examples)

**UC-01: Process a Sale (Actor: Cashier)**

Preconditions: Cashier is logged in; products loaded in catalog.

Main flow:  
1. Cashier scans product barcode or searches product.  
2. System adds item to sale with price and quantity.  
3. Cashier applies discounts if any.  
4. Cashier selects payment type and records payment.  
5. System completes transaction, updates inventory, and prints/sends receipt.

Postconditions: Sale recorded, inventory decreased, receipt generated.

**UC-02: Add New Product (Actor: Manager/Admin)**

Preconditions: Manager logged in.

Main flow:  
1. Manager opens product creation form.  
2. Enters SKU, name, prices, category, supplier, and reorder threshold.  
3. Saves product. System validates required fields and creates product record.

Postconditions: Product available in catalog with initial stock (if provided).

**UC-03: Create Purchase Order (Actor: Manager)**

Preconditions: Low-stock alert or manual order request.

Main flow:  
1. Manager creates purchase order to supplier.  
2. On receipt, manager marks PO as received and inventory is updated.

Postconditions: Inventory increased, PO closed or partially fulfilled.

### 6. UI / UX Notes

- Keep POS screen minimal: large buttons, quick search, support keyboard shortcuts.  
- Inventory screens: filters, bulk import/export, low-stock highlights.  
- Responsive layout that works on tablets and desktops.  
- Provide clear error messages and confirmations for destructive actions.

### 7. Acceptance Criteria

- Core POS flow (sell, receipt, inventory update) works end-to-end in demo environment.  
- Inventory CRUD operations function as specified.  
- Reports generate accurate summaries for a provided test dataset.  
- Role-based access prevents unauthorized actions.  
- Basic security measures (HTTPS, password hashing) implemented for the demo.

### 8. Appendix

Glossary:

- SKU: Stock Keeping Unit.  
- POS: Point of Sale.  
- PO: Purchase Order.  
- MVP: Minimum Viable Product.

References:  
- Standard SRS templates (IEEE recommended practice simplified).

1

2

3

4

5