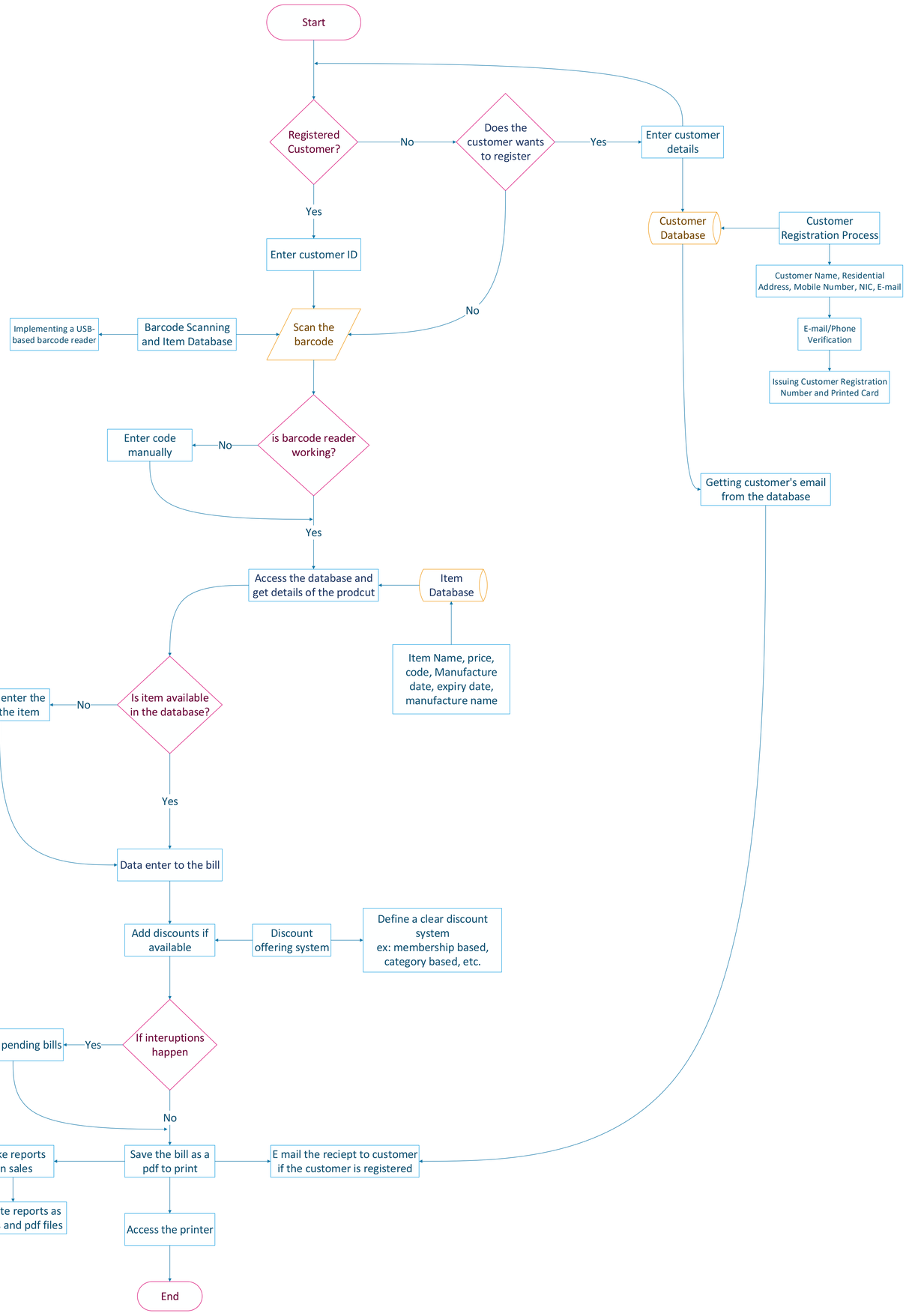
Step-by-Step Development Plan

1. Backend (Java + Spring Boot)
Set up a Spring Boot project.
Create a REST API for handling requests.
Implement authentication & cashier/customer roles.
Add logic for scanning items, discounts, saving bills, etc.

2. Database (PostgreSQL)
Design the schema (tables for customers, bills, items, etc.).
Set up relations (e.g., one customer → multiple bills).
Implement CRUD operations (Create, Read, Update, Delete).

3. Connect Backend & Database
Use Spring Data JPA to communicate with PostgreSQL.
Write repository classes to fetch & update data.
Implement ACID transactions to prevent billing errors.

4. Frontend (React.js)
Design UI for cashiers (scan items, apply discounts).
Integrate frontend with backend using Axios.
Implement real-time updates (like paused/resumed bills).



For Multiple Counters

Database Adjustments (Concurrency & Transactions)
Use ACID-compliant databases like PostgreSQL or MySQL to prevent billing conflicts.
Implement optimistic locking to avoid race conditions when two cashiers access the same product stock.
Use database transactions to ensure bills don't get partially saved if an error occurs mid-process.

Real-Time Updates
WebSocket / Firebase for instant stock updates across counters.
Event-driven messaging (RabbitMQ/Kafka) to sync live inventory across counters.

Unique Bill IDs for Each Counter
Prefix the bill ID with the counter number (e.g., C1-0001, C2-0001) to avoid duplicate bill numbers.

Centralized User & Discount Management
All counters should fetch customer data from a shared database, not local storage.
Discounts should apply dynamically based on shared customer data.

Tech Stack

Frontend: React.js (for cashier UI)
Backend: Spring Boot (REST API, authentication, business logic)
Database: PostgreSQL / MySQL (centralized storage)
Real-Time Updates: WebSockets / Firebase / Redis Pub-Sub
Authentication: JWT-based login for cashiers
PDF Generation: iText / Apache PDFBox
Email Service: JavaMail API / SendGrid

Why This Works Well for Multiple Counters

Centralized customer & item database for all counters.
Web-based → No installation required on cashier PCs.
Real-time stock updates → Prevents double billing issues.
Scalable → Can later integrate admin panels, cashier management, etc.