GOPALAN COLLEGE
OF ENGINEERING AND MANAGEMENT
Whitefield, Bangalore

DEPARTMENT OF
COMPUTER SCIENCE
ENGINEERING.

SEPM PROJECT TITLE

Bookshop Management system

SUBJECT:Software Engineering & Project Management
SUBJECT CODE: BCS501

Submitted By

Name	USN
1.HIMANI	(1GD23CS038)
2. THANISHASRI S S	(1GD23CS111)
3.VINAYAKA.V	(1GD23CS121)

Faculty In-charge
Dr. Manoj Challa
Professor
Department Of computer science &
Engineering

PHASE-2 SYSTEMDESIGN

High-level Design

1. Architecture:

Monolithic Web Application: The system is built as a single, self-contained unit where all components (UI, business logic, data access) are managed in one codebase.

Technology Stack:Backend: Python with the Flask framework.

Database: SQLite for development (easily swappable with PostgreSQL/MySQL for production).

ORM: SQLAlchemy for database interaction, abstracting SQL queries.

Frontend: Server-rendered HTML with Jinja2 templating, styled with TailwindCSS for responsiveness, and Alpine.js for minor client-side interactivity.

Three-Tier Structure: The architecture is logically separated into three main layers:

Presentation Layer: (Templates) Renders the user interface.

Logic/Application Layer: (Flask Routes) Handles business logic and processes user requests.

Data Layer: (SQLAlchemy Models) Manages data and interaction with the database.

Modular Design with Blueprints: The application is organized into modular components called Blueprints (admin, cashier, auth), making the codebase easier to manage, scale, and debug.

Stateless Communication: The server is stateless; each HTTP request from the client contains all the information needed to be processed. User state is managed via secure, server-side sessions handled by Flask-Login.

2. Data Flow:

User Interaction: A user (e.g., Cashier) interacts with the web interface in their browser. Actions like searching for a book or adding an item to the cart trigger events.

HTTP Request: The browser sends an HTTP request to the Flask server. For page loads, it's a standard GET request. For data-intensive actions like book searches or checkout, it's an asynchronous fetch (AJAX) request to a specific API endpoint (e.g., /api/books/search).

Routing and Processing: Flask receives the request and routes it to the appropriate Python function (the "view"). This function processes the request, validates input, and interacts with the data layer.

Database Interaction: The view function uses SQLAlchemy models to query the database (e.g., Book.query.filter(...)) or to create/update records (e.g., db.session.add(...), db.session.commit()).

HTTP Response: The Flask server formulates a response. For an API call, it sends data back in JSON format. For a regular page request, it renders a Jinja2 template with dynamic data, sending back a complete HTML page.

UI Update: The browser receives the response. If it's HTML, the page is rendered or reloaded. If it's JSON, JavaScript on the page uses the data to dynamically update the UI without a full page refresh (e.g., showing search results).

3. Database Schema:

Users Table: Stores login and role information.id (Primary Key), username, password_hash, role (Admin/Cashier).

Books Table: The core inventory catalog

Sales Table: A header record for each transaction.id (Primary Key), timestamp, user_id (Foreign Key to Users), total_amount.

SaleItems Table: A junction table linking Books to Sales.id (Primary Key), sale_id (Foreign Key to Sales), book_id (Foreign Key to Books), quantity, price_per_item.

Key Relationships:One-to-Many: One User can process many Sales.One-to-Many: One Sale can be composed of many SaleItems.

One-to-Many: One Book can be included in many SaleItems.

Low-level Designs

UML(Unified Modeling Language) Diagrams

Actors:

Cashier: An employee responsible for handling sales and customer interactions.

Manager: An employee with administrative access, responsible for inventory, reporting, and employee management.

Customer: An individual who purchases books.

Use Case:

1. Cashier Use Cases

Process Sale: This is the core function. The cashier initiates a new sale, adds books to a cart, applies discounts, and processes payment.

Generate Receipt: After a sale is complete, the system automatically prints or emails a receipt.

Search for Book: The cashier can search the inventory by title, author, or ISBN to check stock availability and price.

View Stock Levels: The cashier can check the current quantity of a specific book.

Handle Returns: The cashier processes a book return, updating the inventory and potentially issuing a refund or store credit.

Log In/Log Out: The cashier authenticates to access the system.

2. Manager Use Case:

All Cashier Use Cases: The Manager can perform all the actions of a cashier.

Manage Inventory: The manager can add new books to the database, update existing book information (e.g., price, quantity), and delete records. This often involves updating stock after a new shipment arrives.

Generate Reports: The manager can generate various reports, such as daily/weekly sales reports, bestsellers, or low-stock reports.

Manage Employees: The manager can add new employees, update their details, and deactivate employee accounts.

Manage Suppliers: The manager can add, update, or delete supplier information.

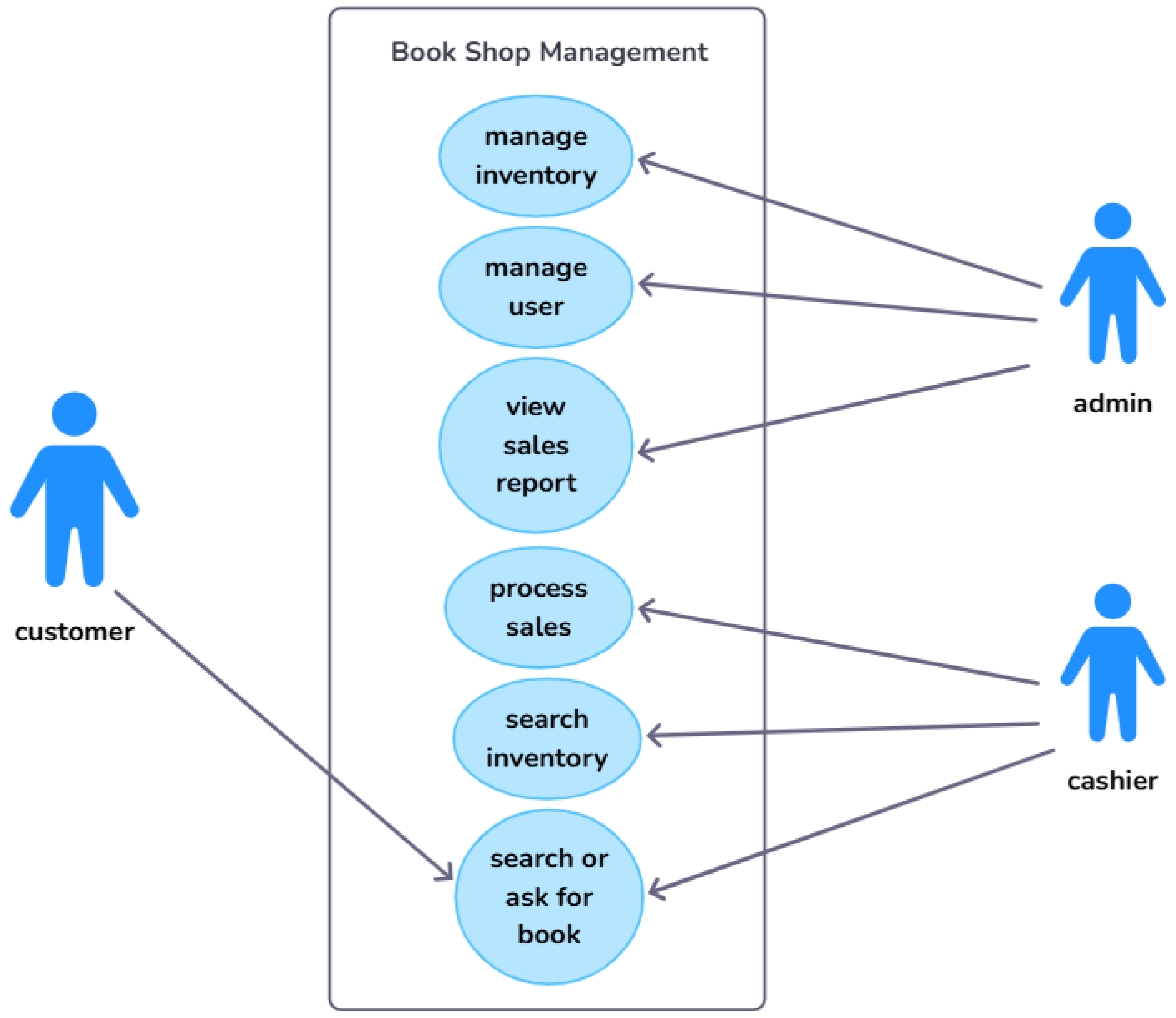
Create Purchase Order: The manager can generate a purchase order for new stock from a supplier.

3. Customer Use Case:

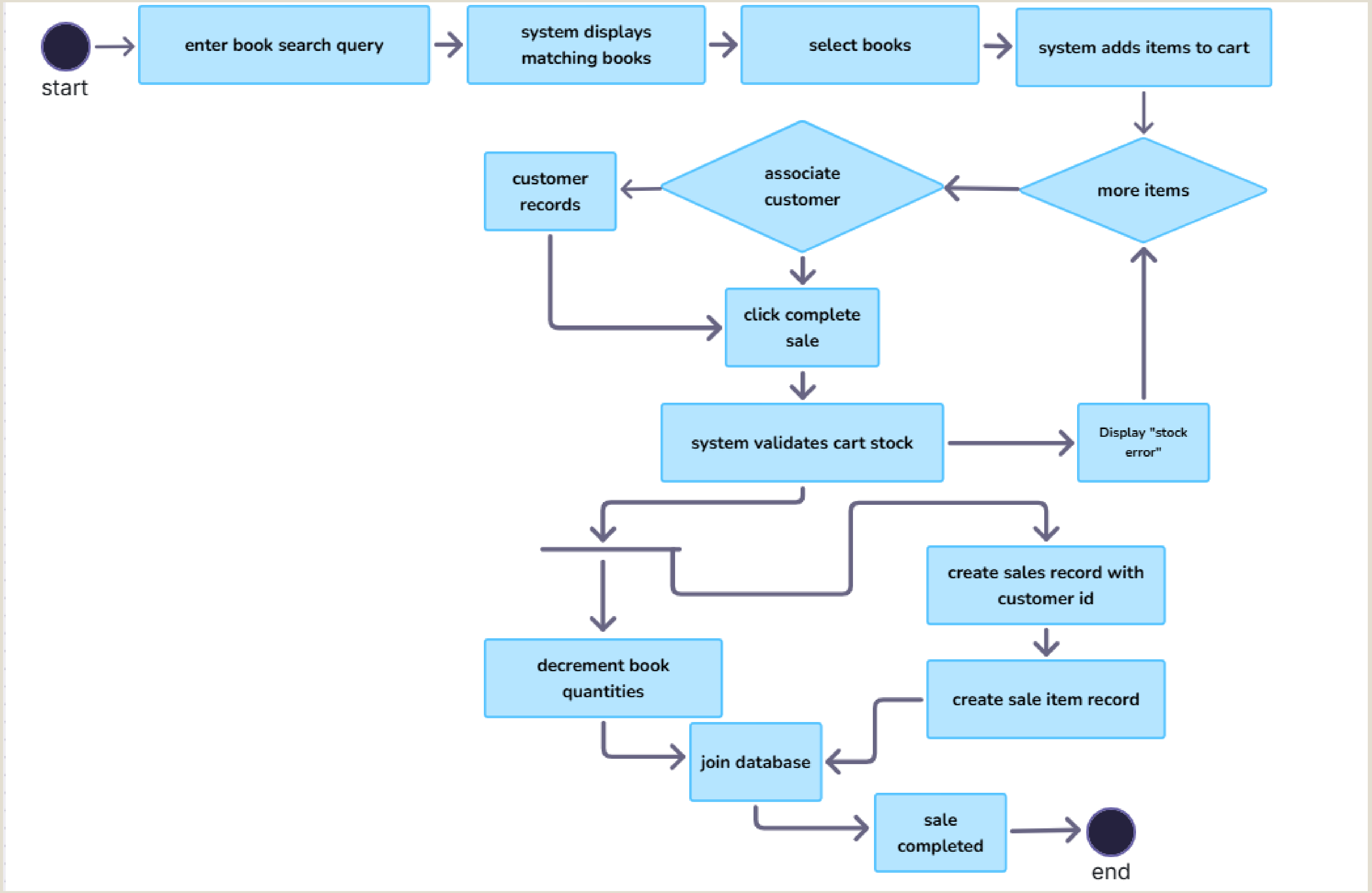
Browse Books: The customer can look through the available books (in a self-service kiosk or online portal, if the system supports it).

Search for Books: The customer can search for a specific book.

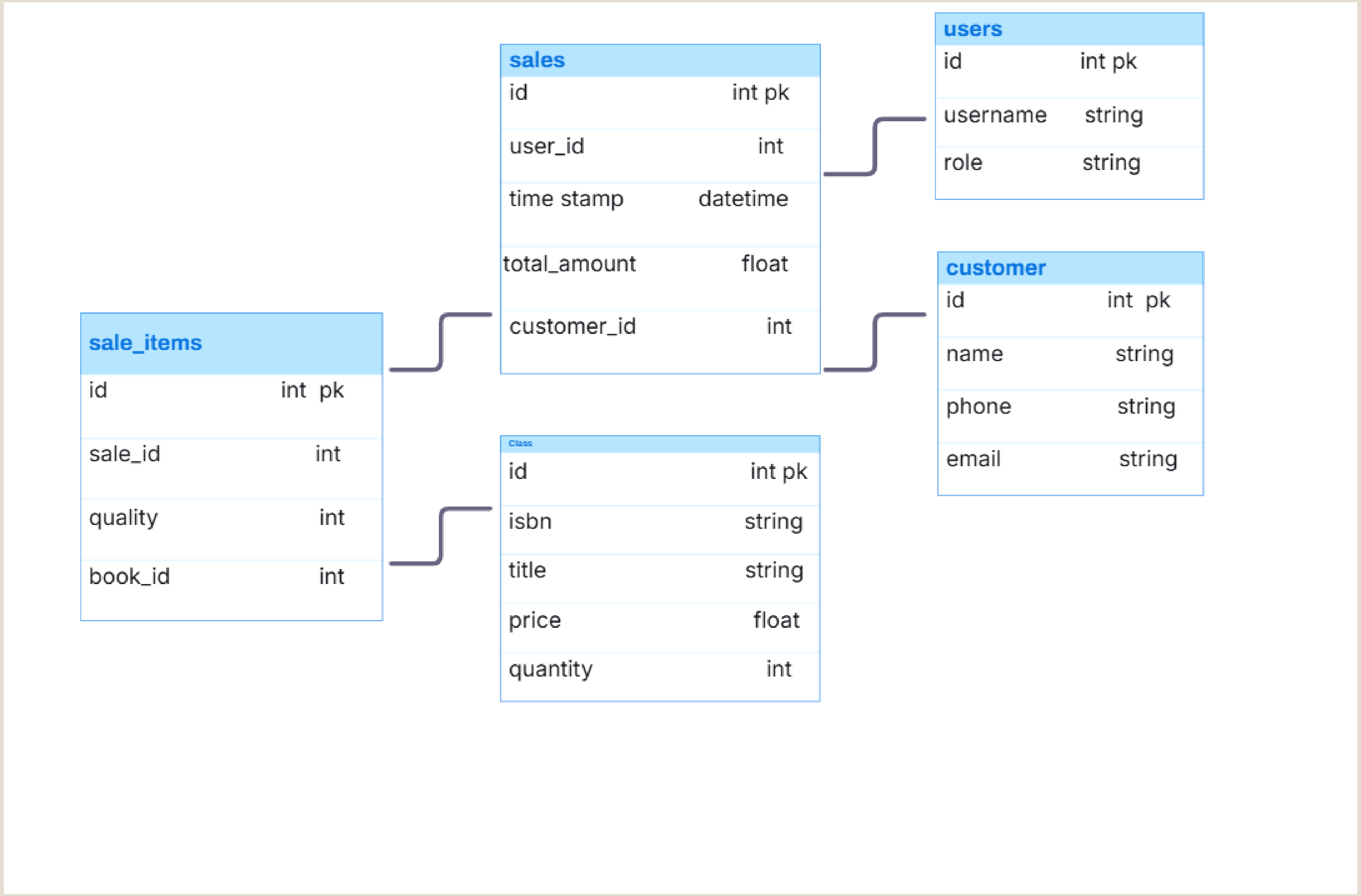
Purchase Books: The customer makes a purchase, usually through the **Cashier's** Process Sale use case.


Book Shop Management
manage inventory
manage user
view sales report
process sales
search inventory
search or ask for book
customer
admin
cashier

Activity Diagram:



Class Diagram:



sales

id	int pk
user_id	int
time stamp	datetime
total_amount	float
customer_id	int

users

id	int pk
username	string
role	string

customer

id	int pk
name	string
phone	string
email	string

sale_items

id	int pk
sale_id	int
quality	int
book_id	int

Class

id	int pk
isbn	string
title	string
price	float
quantity	int

THANK YOU
ANY QUERIES…!