# Salesforce Project Implementation – Smart Bookstore Management System

## Project Title

**Smart Bookstore Management System**

## Problem Statement

A local or online bookstore manages hundreds of books across various genres and receives multiple orders daily. Inventory, order tracking, and customer management are handled manually, leading to: - Stock mismatches and accidental sale of out-of-stock books. - No automated low-stock alerts. - Delayed communication with customers regarding orders and deliveries. - Lack of centralized insights into sales and customer history.

## Objectives

* Automate book inventory management with real-time updates.
* Streamline customer order processing and validation.
* Send automatic notifications for orders and deliveries.
* Maintain a customer database with purchase history.
* Provide dashboards and reports for insights.
* Ensure scalability for future e-commerce expansion.

## Scope

* **Objects:** Book, Order, Customer, Supplier, Order Line.
* **Automation:** Validation Rules, Flows, Email Alerts.
* **Apex Development:** Triggers, Batch Apex, Future Methods.
* **UI:** Lightning Record Pages, LWCs (Book Search, Order Placement).
* **Integrations:** Web-to-Lead, REST API (Payment Gateway).
* **Reports & Dashboards:** Sales vs Stock, Customer Trends.

## Phases with Detailed Process

### **Phase 1: Requirement Gathering & Analysis**

* **Conduct Meetings:**
  + Meet bookstore owner to understand business goals.
  + Discuss daily workflows with staff.
  + Collect customer expectations like fast updates and easy ordering.
* **Identify Pain Points:** Stock mismatches, manual delays, lack of reporting.
* **Document Requirements:** Draft requirement specification document.
* **Industry Study:** Research similar CRM bookstore solutions for ideas.

### **Phase 2: Organization Setup**

* Create Salesforce Developer Org and Sandbox for testing.
* Set up **Company Profile** with bookstore info.
* Create **Users** (Manager, Sales Staff, Support).
* Define **Roles** (Manager > Sales > Staff) and assign profiles.
* Configure **Permission Sets** for specific tasks (e.g., stock editing).
* Set **OWD** (e.g., Orders private, Books read-only).
* Add **Sharing Rules** so managers can access all records.
* Configure **Login IP Ranges** for security.

### **Phase 3: Data Modeling**

* Create Custom Objects: **Book, Order, Customer, Supplier, Order Line.**
* Define Fields:
  + Book: ISBN, Genre, Price, Stock.
  + Order: Order Date, Status, Payment.
  + Customer: Name, Email, Purchase History.
* Define Relationships:
  + Order → Customer (Lookup).
  + Order Line → Book (Master-Detail).
  + Book → Supplier (Lookup).
* Design **Page Layouts** for Books (show stock, price), Orders (show items), Customers (show purchase history).
* Use **Schema Builder** for visual confirmation.

### **Phase 4: Automation**

* **Validation Rules:** Prevent stock < 0, order without a customer.
* **Flows:**
  + Auto-reduce stock when order is confirmed.
  + Auto-send email on order confirmation.
  + Trigger alert when stock < 10.
* **Approval Process:** For supplier restock requests.
* **Email Alerts:** Notify manager when restock needed.

### **Phase 5: Apex Development**

* **Triggers:**
  + Update stock after order creation.
  + Restrict order if stock unavailable.
* **Batch Apex:** Generate daily low-stock report.
* **Future Methods:** Send asynchronous notification emails.
* **Exception Handling:** Gracefully handle errors when stock data missing.
* **Test Classes:** Achieve >75% coverage.

### **Phase 6: User Interface (UI)**

* **Lightning Pages:**
  + Book Record Page with stock banner.
  + Order Page with Order Lines component.
* **LWC Components:**
  + Book Search with filters by genre, price.
  + Order Placement form linked to payment.
  + Low Stock Alert popup for managers.
* **Utility Bar:** Quick access to Sales Dashboard.
* **Navigation Service:** Link LWCs with record details.

### **Phase 7: Integration**

* **Web-to-Lead:** Capture book inquiries from website.
* **Payment Gateway Integration:** REST API callout to process payments.
* **OAuth Authentication:** Secure API calls.
* **Named Credentials:** Store API login info securely.
* **Remote Site Settings:** Allow external callouts.
* Optional: Supplier system integration for auto-restocking.

### **Phase 8: Data Management & Deployment**

* **Data Import Wizard:** Import initial book and customer data.
* **Data Loader:** Bulk upload orders and suppliers.
* **Duplicate Rules:** Avoid duplicate customer records.
* **Change Sets:** Deploy metadata from Sandbox to Production.
* **Backup:** Weekly data export for safety.

### **Phase 9: Reporting & Dashboards**

* Reports:
  + Top-Selling Books.
  + Monthly Sales Trends.
  + Low-stock Items.
* Dashboards:
  + Sales vs Stock.
  + Genre-wise Sales Trends.
  + Customer Loyalty Activity.
* Schedule automatic report emails to manager.

### **Phase 10: Final Demo & Testing**

* **End-to-End Flow Test:** Add Book → Place Order → Stock Update → Email.
* **System Integration Testing:** Payment API and Web-to-Lead.
* **Report Validation:** Check sales and stock trends.
* **UAT:** Bookstore staff test real-life scenarios.
* **Presentation & Documentation:** Demo to stakeholders, handover guide.

## Expected Outcomes

* Minimized stock mismatches and order delays.
* Faster order processing with real-time updates.
* Improved customer satisfaction via automated communication.
* Insights into sales trends and customer loyalty.
* Scalable system for future e-commerce growth.

## Salesforce Detailed Process (High-Level)

* **Admin Side:**
  + Configure users, roles, profiles, security.
  + Build data model and automation (Flows, Rules).
  + Create reports and dashboards.
* **Developer Side:**
  + Write Apex triggers, batch jobs, test classes.
  + Build LWCs for Book Search, Orders, Alerts.
  + Handle REST API integrations.
* **Deployment Side:**
  + Use Change Sets/SFDX for migration.
  + Perform UAT and Production Release.

This document now gives **complete detailed actions for all phases** of the Smart Bookstore Management System Salesforce project.