**LIBRARY MANAGEMENT SYSTEM**

**Library Management System in Salesforce:**

This project helps you understand **Salesforce Admin & Development** (without integration). It involves managing books, borrowers, lending records, and overdue tracking in a library system.

**Project Scope:**

The **Library Management System (LMS)** will include the following features:  
✅ Book Inventory Management  
✅ Borrower Registration & Management  
✅ Book Lending & Return Tracking  
✅ Overdue Fine Calculation  
✅ Role-Based Access Control  
✅ Reports and Dashboards

**Step-by-Step Implementation Plan:**

1. Create Custom Objects (Covers: Objects, Fields, Relationships)

Go to Setup → Object Manager → Create Custom Object

**Objects & Fields:**

|  |  |  |
| --- | --- | --- |
| **BOOK** | | |
| **FIELD NAME** | **API NAME** | **DATATYPE** |
| Name | Name | Text/Standard field |
| ISBN Number | ISBN\_Number\_\_c | Text, Unique |
| Author | Author\_\_c | Text |
| Publisher | Publisher\_\_c | Text |
| Genre | Genre\_\_c | Picklist(Fiction, Non-Fiction, Science, History) |
| Availability Status | Availability\_Status\_\_c | Picklist(Available, Borrowed, Reserved) |
| Shelf Location | Shelf\_Location\_\_c | Text |
| Number of Copies | Number\_of\_Copies\_\_c | Number |

|  |  |  |
| --- | --- | --- |
| **BORROWER** | | |
| **FIELD NAME** | **API NAME** | **DATATYPE** |
| Name | Name | Text/Standard field |
| Borrower ID | Borrower\_ID\_\_c | Text, Unique |
| Email | Email\_\_c | Email |
| Phone | Phone\_\_c | Phone |
| Membership Type | Membership\_Type\_\_c | Picklist(Regular, Premium) |
| Status | Status\_\_c | Picklist(Active, Inactive) |

|  |  |  |
| --- | --- | --- |
| **BOOK LENDING** | | |
| **FIELD NAME** | **API NAME** | **DATATYPE** |
| Lending ID | Name | AutoNumber/Standard field |
| Borrower | Borrower\_\_c | Lookup to Borrower |
| Book | Book\_\_c | Lookup to Book |
| Issue Date | Issue\_Date\_\_c | Date |
| Due Date | Due\_Date\_\_c | Date |
| Return Date | Return\_Date\_\_c | Date |
| Fine Amount | Fine\_Amount\_\_c | Currency |
| Status | Status\_\_c | Picklist(Issued, Returned, Overdue) |

2. Setup Profiles & Permission Sets (Covers: Security, Profiles, Permission Sets)

Go to Setup → Profiles → Create Custom Profiles

📌 Roles & Profiles:

✅ **Librarian Profile** → Full Access (Can Add, Update, Delete Books & Manage Lending)  
✅ **Member Profile** → Limited Access (Can Only View Books & Borrow Books)

**3. Automate Processes with Flows & Validation Rules**

(Covers: Process Automation, Flows, Validation Rules)

1️⃣ **Validation Rules:**

**A) Ensure Due Date is after Issue Date:** Due\_Date\_\_c > Issue\_Date\_\_c  
**B) Prevent duplicate ISBN Numbers:** Set **ISBN\_Number\_\_c** as a **Unique Field**  
**C) Prevent Book Lending if Status is "Borrowed":**

IF(Availability\_Status\_\_c = "Borrowed", TRUE, FALSE)

**2️⃣ Approval Process for Leave Requests:**

Go to Setup → Approval Process → New Approval Process

✅ **Entry Criteria:** Status\_\_c = "Issued"  
✅ **Step 1:** Librarian Approves or Rejects the Borrow Request  
✅ **Final Step:** If Approved → Update Status\_\_c to "Issued"

**3️⃣ Record-Triggered Flow for Welcome Email:**

Triggered when a Lending Record is Updated  
✅ Condition: If Return\_Date\_\_c > Due\_Date\_\_c → Calculate Fine Amount  
✅ Fine Calculation Formula:

IF(Return\_Date\_\_c > Due\_Date\_\_c, (Return\_Date\_\_c - Due\_Date\_\_c) \* 2, 0)

(Assuming $2 fine per day of delay)

**4. Reports & Dashboards (Covers: Reports, Dashboards):**

📌 **Reports:**  
A) **Book Inventory Report** (Lists all books, grouped by Availability Status)  
B) **Overdue Books Report** (Lists books not returned on time with fine amounts)  
C) **Borrower Activity Report** (Shows books borrowed per member)

📌 **Dashboards:**  
📊 **Library Performance Dashboard** (Total Books Borrowed, Overdue Books, Total Fine Collected)

**5. Trigger to prevent duplicate Borrower ID**

**trigger PreventDuplicateBorrowerID on Borrower\_\_c (before insert, before update) {**

**Set<String> existingIDs = new Set<String>();**

**for (Borrower\_\_c b : [SELECT Borrower\_ID\_\_c FROM Borrower\_\_c]) {**

**existingIDs.add(b.Borrower\_ID\_\_c);**

**}**

**for (Borrower\_\_c newB : Trigger.new) {**

**if (existingIDs.contains(newB.Borrower\_ID\_\_c)) {**

**newB.addError('Borrower ID must be unique.');**

**}**

**}**

**}**

6. **Batch class to update Book Availability Status**

**Scenario:** If all copies of a book are borrowed, **Availability Status** should update to "Borrowed".

**global class UpdateBookStatusBatch implements Database.Batchable<sObject> {**

**global Database.QueryLocator start(Database.BatchableContext BC) {**

**return Database.getQueryLocator('SELECT Id, Number\_of\_Copies\_\_c FROM Book\_\_c');**

**}**

**global void execute(Database.BatchableContext BC, List<Book\_\_c> bookList) {**

**for (Book\_\_c book : bookList) {**

**Integer issuedCopies = [SELECT COUNT() FROM Book\_Lending\_\_c WHERE Book\_\_c = :book.Id AND Status\_\_c = 'Issued'];**

**book.Availability\_Status\_\_c = (issuedCopies >= book.Number\_of\_Copies\_\_c) ? 'Borrowed' : 'Available';**

**}**

**update bookList;**

**}**

**global void finish(Database.BatchableContext BC) {**

**System.debug('Batch Job Completed.');**

**}**

**}**

**7. Lightning Web Components (LWC) for User Interface**

📌 **Employee List LWC** (Displays books with search & filter options)  
📌 **Borrow Book LWC** (Allows members to request book borrowing)

**📌 LWC: Book List Component**