**Payment Tracking in Banking**

**Problem statement**

Loan payment tracking in banks is mostly manual, causing delays, errors, and poor customer experience. Overdue payments and recovery processes lack automation and centralized tracking. A Salesforce-based solution is needed to automate tracking, reminders, and role-based access.

**Phase 1: Problem Understanding & Industry Analysis**

**Requirement Gathering**  
 • Identify gaps in manual payment tracking.  
 • Understand issues in sending timely payment reminders.

**Stakeholder Analysis**  
 • Customers, bank staff, recovery agents, and managers.  
 • Focus on their roles and interaction with the payment system.

**Business Process Mapping**  
 • Map repayment lifecycle:

due date → reminder → payment → overdue → recovery.  
 • Visualize processes for loan and EMI repayments.

**Industry-specific Use Case Analysis**  
 • EMI reminders, loan repayments, credit card dues monitoring.  
 • Streamline overdue and recovery workflows.

**AppExchange Exploration**  
 • Research payment reminder.  
 • Identify ready-made apps to speed up implementation.

**Phase 2: Org Setup & Configuration**

**Salesforce Editions** • Use Developer Edition for initial development and testing.  
 • Plan for Enterprise Edition for production banking workflows**.**

**Company Profile Setup** • Set Company Name, Currency (INR/USD), and Time Zone.  
 • Configure organization-wide defaults.

**Business Hours & Holidays** • Define working hours (e.g., Mon–Sat, 9 AM–6 PM).  
 • Add holidays like Independence Day and Diwali.

**Fiscal Year Settings** • Use standard Jan–Dec fiscal year or custom if needed.  
 • Align with financial reporting requirements.

**User Setup & Licenses** • Create Loan Officer, Recovery Agent, and Manager users.  
 • Assign appropriate licenses (Salesforce / Salesforce Platform).

**Profiles & Roles**• Create custom profiles (Loan Officer, Manager, Recovery Agent).  
 • Establish role hierarchy (Manager → Loan Officer / Recovery Agent).

**Permission Sets** • Create a Permission Set for payment tracking access.  
 • Assign Permission Set to Loan Officers and Managers.

**OWD & Sharing Rules** • Set custom objects (Loan, Payment, Recovery) to Private.  
 • Create rules to share overdue records with Recovery Agents.

**Login Access Policies**• Set session timeout to 30 minutes.  
 • Enable login hours (8 AM–8 PM) for users.

**Dev Org & Sandbox Usage** • Use Developer Org for development and testing.  
 • Plan sandbox usage for production-like testing later.

**Deployment Basics** • Use SFDX commands to push/pull metadata.  
 • Prepare for deployment via Change Sets or SFDX when ready.

**Phase 3: Data Modeling & Relationships**

**Standard & Custom Objects**

* Use Account/Contact for Customers.
* Create custom objects: Loan, Payment, Recovery, Loan Agent Assignment (junction).

**Fields**

* Loan: Loan Number (Auto Number), Principal Amount (Currency), Interest Rate (Percent), Term Months (Number), Start Date (Date), Status (Picklist), Customer (Lookup to Account/Contact).
* Payment: Payment Date (Date), Amount (Currency), Payment Method (Picklist: Cash/UPI/Transfer), Status (Picklist: Paid/Overdue), Loan (Master-Detail).
* Recovery: Recovery Date (Date), Amount Recovered (Currency), Status (Picklist: Assigned/InProgress/Completed), Loan (Lookup), Assigned Agent (Lookup to User).
* Loan Agent Assignment (junction): Loan (Master-Detail), Assigned Agent (Lookup to User).

**Record Types**

* Loan: Retail Loan & Business Loan.
* Configure Status picklist values per record type (e.g., Retail = Active/Closed, Business = Active/Defaulted).

**Page Layouts**

* Loan Layout: sections for Basic Info, Financials, Dates & Status. Related Lists = Payments, Recoveries, Loan Agent Assignments.
* Payment Layout: fields for Date, Amount, Method, Status.
* Recovery Layout: fields for Date, Amount Recovered, Status, Assigned Agent.

**Compact Layouts**

* Loan Compact Layout: Loan Number, Principal Amount, Status, Customer.
* Payment Compact Layout: Payment Date, Amount, Status.
* Recovery Compact Layout: Recovery Date, Amount Recovered, Status.

**Schema Builder**

* Visualize Loan, Payment, Recovery, and Loan Agent Assignment objects.
* Confirm relationships: Loan–Payment (Master-Detail), Loan–Recovery (Lookup), Loan–LoanAgentAssignment (Master-Detail), LoanAgentAssignment–User (Lookup).

**Lookup vs Master-Detail vs Hierarchical**

* Payment → Loan = Master-Detail (payments always tied to loans).
* Recovery → Loan = Lookup (independent audit records).
* Loan Agent Assignment → Loan = Master-Detail, → User = Lookup.
* Hierarchical not used (only for User object).

**Junction Objects**

* Loan Agent Assignment to handle many-to-many between Loans and Agents.
* Enables multiple agents handling multiple loans.

**External Objects**

* Concept: Connect to external payment system using Salesforce Connect.
* Example: Show legacy payment history without storing inside Salesforce.

**Phase 4: Process Automation (Admin)**

**Validation Rules**

* Ensure values are valid (Loan/Payment/Recovery > 0).
* Prevent negative amounts and invalid dates.

**Workflow Rules**

* Rule: Pending\_Payment\_Reminder.
* Sends reminder email for pending payments.

**Process Builder**

* Sends email reminders for due payments.
* Updates Loan status when all Payments are paid.
* Creates follow-up task for overdue payments.

**Approval Process**

* Loans > 5 Lakhs require Manager approval.
* Auto email + status update on approval/rejection.

**Flow Builder**

* Scheduled Flow: Sends payment reminders 2 days before due.
* Record-Triggered: Marks payments overdue automatically.
* Record-Triggered: Creates Recovery record when overdue.

**Email Alerts**

* Payment\_Reminder\_Email → Customer.
* Loan\_Submitted, Approved, Rejected → Manager/Officer.

**Field Updates**

* Auto-change Payment.Status to Overdue.
* Auto-close Loan when all Payments are Paid.

**Tasks**

* Task created when Payment is Overdue.
* Assigned to Recovery Agent with high priority.

**Custom Notifications**

* Sends in-app notification for Overdue Payments.

**Phase 5: Apex Programming (Developer)**

**Classes & Objects**

* Created Apex helper classes (LoanHandler, PaymentHandler, RecoveryHandler) to separate business logic from triggers.
* Used object-oriented approach for cleaner, reusable, and modular code.

**Apex Triggers**

* Implemented triggers for Loan, Payment, and Recovery to handle automation like updating statuses and roll-ups.
* Used before/after events for validation and DML updates while avoiding recursion.

**Trigger Design Pattern**

* Applied Trigger Handler Pattern to keep trigger code clean and delegate logic to handler classes.
* Ensured only one trigger per object with proper event handling.

**SOQL & SOSL**

* Used SOQL in classes to query Loans, Payments, and Recoveries.
* Implemented simple search functionality to fetch customer records.

**Collections: List, Set, Map**

* Applied Lists to hold multiple Payments for bulk processing.
* Used Maps for Loan → Payments mapping to handle roll-ups efficiently.

**Control Statements**

* Implemented IF conditions for validation

**Batch Apex**

* Created simple batch class for recalculating loan balances.
* Enabled processing of large datasets asynchronously.

**Queueable Apex**

* Used Queueable Apex for lightweight async operations like notifications.
* Provides better chaining support compared to future methods.

**Scheduled Apex**

* Built a scheduler to run daily overdue payment checks.
* Automated reminders and recovery creation without manual intervention.

**Future Methods**

* Implemented future call for sending async email notifications.

**Exception Handling**

* Created a centralized ErrorHandler class to log and handle exceptions.
* Ensured smooth error reporting without stopping automation.

**Test Classes**

* Wrote test classes for each handler & trigger to validate functionality.

**Asynchronous Processing**

* Combined Batch, Queueable, Scheduled, and Future methods for async operations.
* Ensured the system scales for high-volume data in banking use cases.

**Phase 6: User Interface Development**

**Lightning App Builder**

* Created a custom Banking App with navigation tabs for Loan, Payment, Recovery.
* Assigned the app to custom profiles (Loan Officer, Manager, Recovery Agent).

**Record Pages**

* Built custom record pages for Loan, Payment, and Recovery using Lightning App Builder.
* Added Highlights Panel, Related Lists, and Custom LWCs for better data visibility.

**Tabs**

* Created custom tabs for Loan\_\_c, Payment\_\_c, Recovery\_\_c.
* Added them into the Banking App navigation bar for easy access.

**Home Page Layouts**

* Customized Home Page to show Dashboards, Reports, and Key Loan Metrics.

**Utility Bar**

* Added Notes, Recent Items.
* Ensured visibility across the app for all profiles.

**LWC**

* Developed loanSummary, loanDetails, paymentChild LWCs for dynamic UI.
* Components fetch Loan/Payment/Recovery data via Apex and display in dashboards.

**Apex with LWC**

* Created PaymentSummaryController.cls to fetch Loan & Payment records.
* Integrated Apex with LWCs using @wire adapters and imperative calls.

**Events in LWC**

* Used Custom Events for parent-child communication between LWCs.

**Wire Adapters**

* Used @wire(getRecord) to fetch Loan fields in real-time from UI RecordAPI

**Imperative Apex Calls**

* Used imperative Apex methods (@AuraEnabled) for complex queries and filtering.

**Navigation Service**

* Implemented NavigationMixin to allow programmatic navigation**.**