

# Payment Tracking in Banking

## Project Overview:

The Payment Tracking in Banking CRM is a Salesforce-based solution designed to track loan repayment management for banks. Traditionally, banks relied on manual processes to track EMI schedules, overdue payments, and loan recovery, which led to errors, delays, and poor customer experience. This CRM resolves those issues by providing centralized loan and payment records, automated reminders for customers, recovery workflows, dashboards. It leverages Salesforce automation tools like validation rules, workflows. Creating custom objects for Loan, Payment and recovery provide a structure way to manage financial data and secure sharing rules, field level security protect sensitive customer and financial information. The system ensures timely payment tracking, efficient collections, and transparent communication with customers through email.

## Objectives:

- To build a secure, automated Salesforce CRM that centralizes loan, payment and recovery management while Improving customer experience through timely remainders.
- Enabling role-based access so managers, officers, and agents can securely view only the data relevant to them.
- Providing real-time dashboards and reports for monitoring loan status, overdue amounts, and recovery performance.
- Ensuring customer communication via email alerts for due dates, approvals, and recovery updates.
- Supporting business efficiency by minimizing manual work, improving compliance, and enhancing customer satisfaction.

## **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.

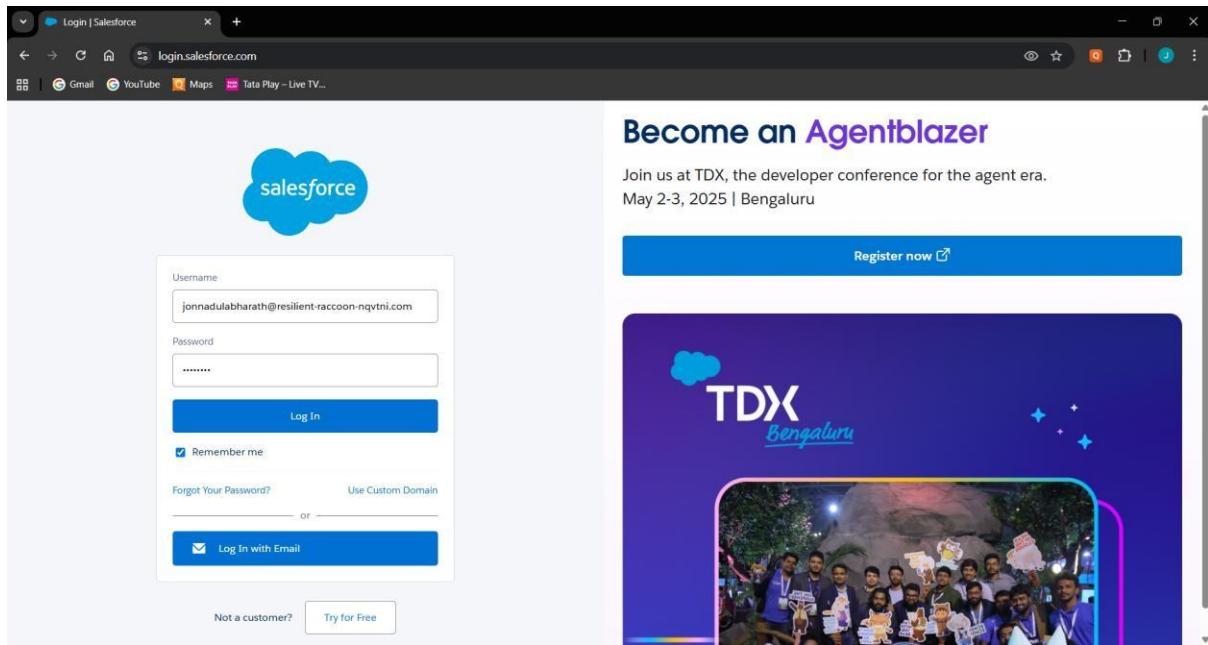
### **AppExchange Exploration**

- Research payment reminder.
- Identify ready-made apps to speed up implementation.

## Phase 2: Org Setup & Configuration

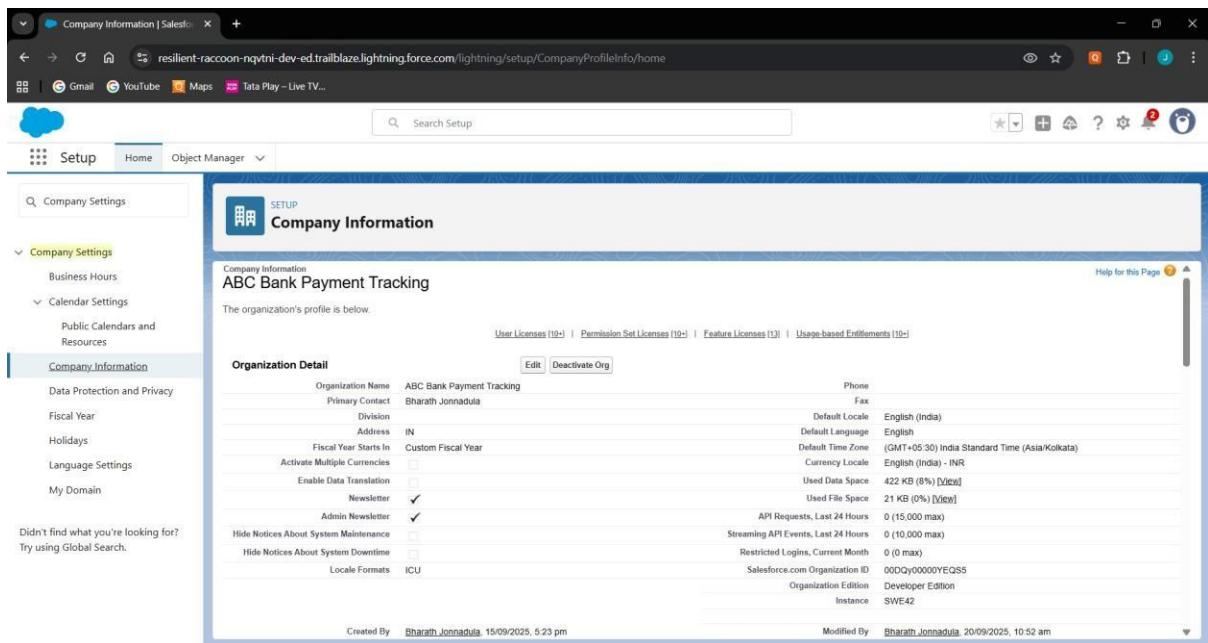
### Salesforce Editions

- 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.

The screenshot shows the Salesforce Setup interface for 'Business Hours'. The left sidebar under 'Company Settings' has 'Business Hours' selected. The main content area is titled 'Business Hours' and contains three steps: Step 1. Business Hours Name (Bank Working Hours, Active), Step 2. Time Zone ((GMT+05:30) India Standard Time (Asia/Kolkata)), and Step 3. Business Hours (a grid showing daily work hours from 9:00 am to 6:00 pm, with a '24 hours' checkbox checked for each day). A note at the top says: 'Select the days and hours that your support team is available. These hours, when associated with escalation rules, determine the times at which cases can escalate. If you enter blank business hours for a day, that means your organization does not operate on that day.'

## Fiscal Year Settings

- Use standard Jan–Dec fiscal year or custom if needed.
- Align with financial reporting requirements.

The screenshot shows the Salesforce Setup interface for 'Fiscal Year'. The left sidebar under 'Company Settings' has 'Fiscal Year' selected. The main content area is titled 'Fiscal Year' and contains sections for 'Custom Fiscal Years' (New button) and 'Custom Fiscal Year Names' (Action column: Edit | Replace, Field Label column: Quarter Prefix, Period Prefix, Quarter Name, Period Name). A note at the top says: 'This page allows you to define and edit custom fiscal years, including the names used in reports and forecasts. Click the New button to define a new fiscal year. Click Edit to edit a previously defined fiscal year.'

## User Setup & Licenses

- Create Loan Officer, Recovery Agent, and Manager users.
- Assign appropriate licenses (Salesforce / Salesforce Platform).

The screenshot shows the Salesforce Setup interface under the 'Users' section. A new user record is being created for 'Branch Manager'. The 'General Information' section includes fields for First Name (Branch), Last Name (Manager), Alias (manager), Email (bankmanager.officer@bank), Username (bankmanager.officer@bank), Nickname (manager), Title (empty), Company (empty), Department (empty), and Division (empty). On the right side, the 'Role' dropdown is set to '<None Specified>', 'User License' to 'Salesforce', 'Profile' to 'Manager Profile', and 'Active' is checked. Other optional fields like Marketing User, Offline User, Knowledge User, Flow User, Service Cloud User, Site.com Contributor User, Site.com Publisher User, WDC User, and Data.com User Type are listed below.

The screenshot shows the Salesforce Setup interface under the 'Users' section. A new user record is being created for 'Recovery Agent'. The 'General Information' section includes fields for First Name (Recovery), Last Name (Agent), Alias (recovery), Email (recovery.officer@bankdemo), Username (recovery.officer@bankdemo), Nickname (recovery agent), Title (empty), Company (empty), Department (empty), and Division (empty). On the right side, the 'Role' dropdown is set to '<None Specified>', 'User License' to 'Salesforce Platform', 'Profile' to 'Standard Platform User', and 'Active' is checked. Other optional fields are listed on the right.

The screenshot shows the Salesforce Setup interface under the 'Users' section. A new user record is being created for 'Loan Officer'. The 'General Information' section includes fields for First Name (Loan), Last Name (Officer), Alias (loan), Email (loan.officer@bankdemo.cor), Username (loan.officer@bankdemo.cor), Nickname (loanofficer), Title (empty), Company (empty), Department (empty), and Division (empty). On the right side, the 'Role' dropdown is set to '<None Specified>', 'User License' to 'Salesforce Platform', 'Profile' to 'Standard Platform User', and 'Active' is checked. Other optional fields are listed on the right.

## Login Access Policies

- Set session timeout to 30 minutes.
- Enable login hours (8 AM–8 PM) for users.

The screenshot shows the 'Session Settings' page in the Salesforce Setup interface. The URL is <https://resilient-raccoon-nqytni-dev-ed.trailblaze.lightning.force.com/lightning/setup/SecuritySession/home>. The left sidebar shows 'Session Management' and 'Session Settings' selected. The main content area is titled 'Session Settings' and includes sections for 'Session Timeout' (set to 30 minutes), 'Session Settings' (with various checkboxes like 'Lock sessions to the IP address from which they originated'), and 'Extended use of IE11 with Lightning Experience' (warning message about the end of support). A note at the bottom states: "AS OF DECEMBER 31, THE EXTENDED PERIOD HAS ENDED, AND USE OF INTERNET EXPLORER 11 (IE 11) WITH LIGHTNING EXPERIENCE IS NO LONGER SUPPORTED. ISSUES WITH PERFORMANCE OR FUNCTIONALITY" followed by a link.

## 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.

The screenshot shows the VS Code editor with an Apex class named 'LoanDetailsController.cls'. The code defines a static method 'getRecord' that queries a 'Loan\_\_c' record based on its ID. The code editor has syntax highlighting for Apex and shows line numbers. The left sidebar shows the project structure under 'PAYMENTTRACKINGINBANKING' with files like 'loanSummary.html', 'loanSummary.js', 'loanSummary.js-meta.xml', 'LoanDetailsController.cls', and 'paymentChild.htm'. The bottom status bar indicates the file is 'Activating Extensions...'.

## Phase 3: Data Modeling & Relationships

### Standard & Custom Objects

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

The image displays two side-by-side screenshots of the Salesforce Object Manager setup screen. Both screenshots show the 'Details' tab for a custom object, with the left sidebar collapsed.

**Loan Object Setup:**

- API Name:** Loan\_\_c
- Singular Label:** Loan
- Plural Label:** Loans

**Object Settings (Right Panel):**

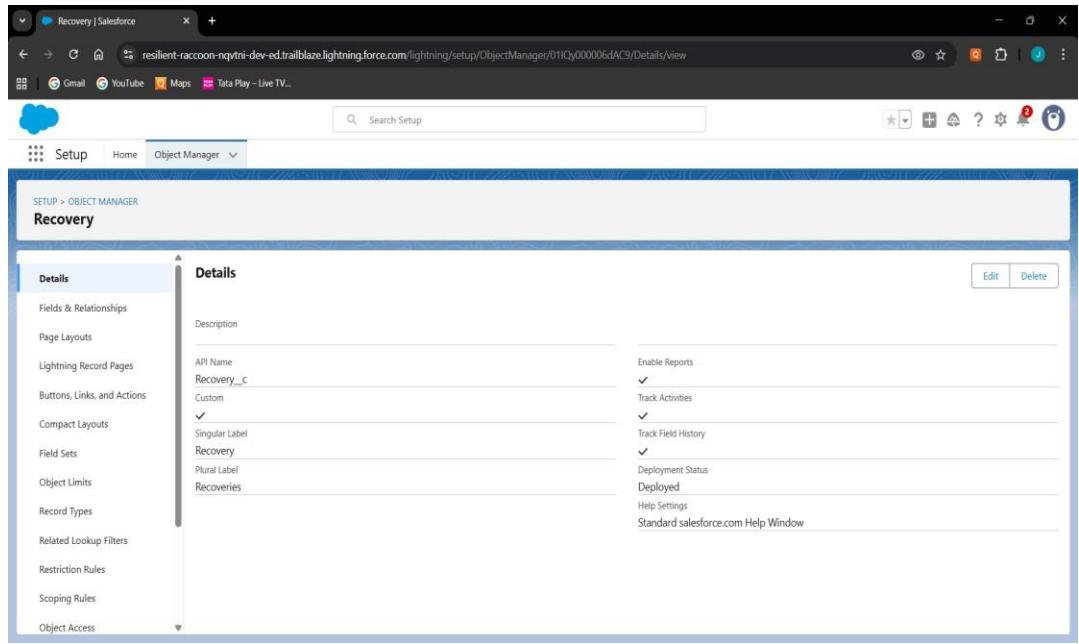
- Enable Reports: ✓
- Track Activities: ✓
- Track Field History: ✓
- Deployment Status: Deployed
- Help Settings: Standard salesforce.com Help Window

**Payment Object Setup:**

- API Name:** Payment\_\_c
- Singular Label:** Payment
- Plural Label:** Payments

**Object Settings (Right Panel):**

- Enable Reports: ✓
- Track Activities: ✓
- Track Field History: ✓
- Deployment Status: Deployed
- Help Settings: Standard salesforce.com Help Window



## 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).

Fields & Relationships					
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED	
Amount Recovered	Amount_Recovered__c	Currency(16, 2)			
Assigned Agent	Assigned_Agent__c	Lookup(User)			
Created By	CreatedById	Lookup(User)			
Last Modified By	LastModifiedById	Lookup(User)			
Loan	Loan__c	Lookup(Loan)			
Owner	OwnerId	Lookup(User, Group)			
Recovery Date	Recovery_Date__c	Date			
Recovery Number	Name	Auto Number			
Status	Status__c	Picklist			

## Record Types

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

The screenshot shows the Salesforce Object Manager interface for the 'Loan' object. The left sidebar lists various configuration options: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types (which is selected and highlighted in blue), Related Lookup Filters, Search Layouts, List View Button Layout, and Restriction Rules. The main content area is titled 'Record Types' and displays a table with two items: 'Business Loan' and 'Retail Loan'. The table columns are 'RECORD TYPE LABEL', 'DESCRIPTION', 'ACTIVE', and 'MODIFIED BY'. Both records were modified by 'Bharath Jonnadula' on 20/09/2025, 5:10 pm.

RECORD TYPE LABEL	DESCRIPTION	ACTIVE	MODIFIED BY
Business Loan		✓	Bharath Jonnadula, 20/09/2025, 5:10 pm
Retail Loan		✓	Bharath Jonnadula, 20/09/2025, 5:10 pm

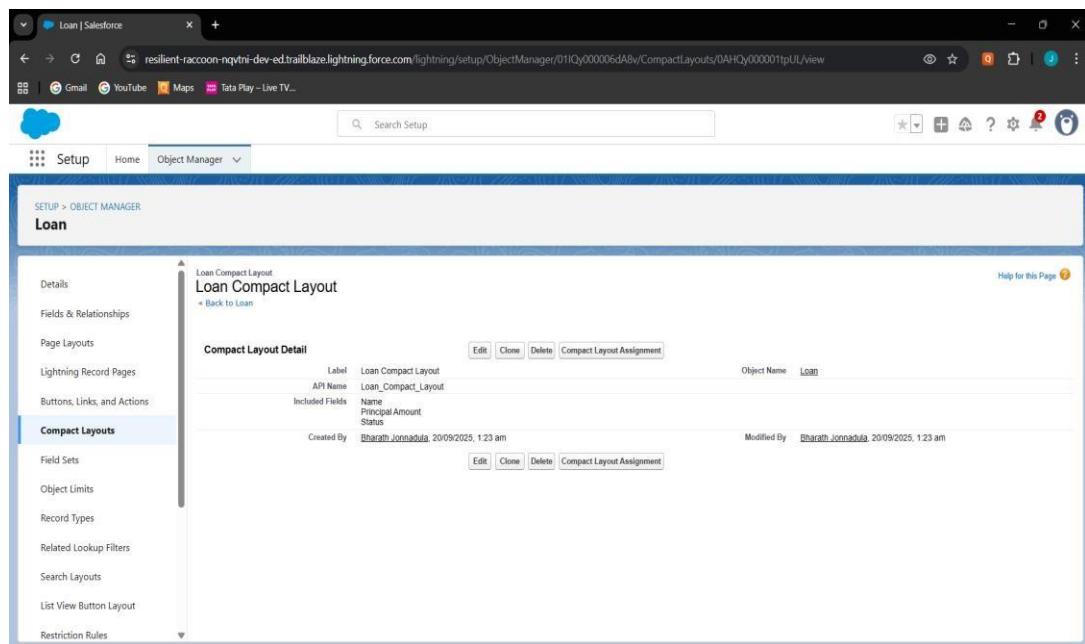
## Page Layouts

- 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.

The screenshot shows the Salesforce Object Manager interface for the 'Loan' object, specifically the 'Page Layouts' section. The left sidebar includes 'Details', 'Fields & Relationships', Page Layouts (which is selected and highlighted in blue), Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, and Restriction Rules. The main content area is titled 'Loan Layout' and shows the configuration for the 'Loan' page layout. It includes tabs for 'Fields', 'Buttons', 'Quick Actions', 'Mobile & Lightning', 'Actions', 'Expanded Lookups', 'Search', 'Links', and 'Derkont Private'. A 'Layout Properties' panel is open, showing fields like 'Name', 'Save Form', 'Status', 'Record Type', and 'Record ID'. Below the layout properties, there are sections for 'Loan Sample', 'Highlights Panel', 'Quick Actions In the Salesforce Classic Publisher', and 'Salesforce Mobile and Lightning Experience Actions'.

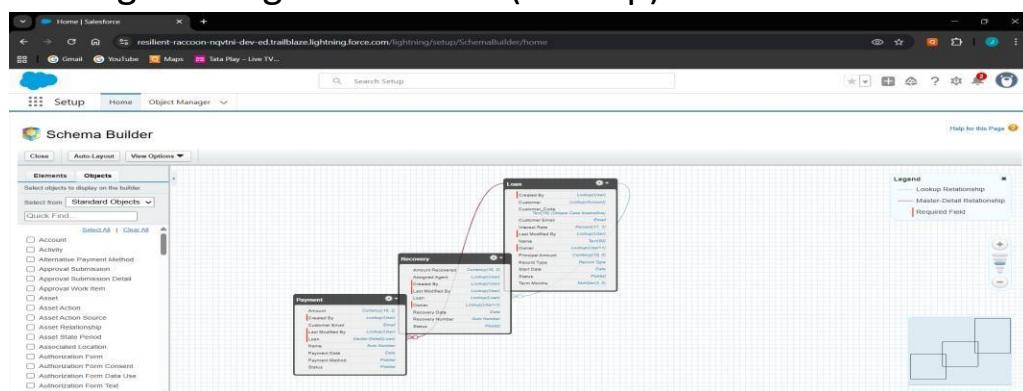
## Compact Layouts

- 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**

- Connect to external payment system using Salesforce Connect.

## Phase 4: Process Automation (Admin)

### Validation Rules

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

The image displays two screenshots of the Salesforce Object Manager interface, specifically showing the 'Validation Rules' section for the 'Loan' and 'Payment' objects.

**Loan Validation Rules:**

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Interest_Rate_Range	Interest Rate	Interest Rate must be between 0 and 100.	✓	Bharath Jonnadula, 20/09/2025, 11:44 am
Principal_Positive	Principal Amount	Principal Amount must be greater than zero.	✓	Bharath Jonnadula, 20/09/2025, 11:44 am
Term_Positive	Term Months	Loan term (Term Months) must be greater than zero.	✓	Bharath Jonnadula, 20/09/2025, 11:43 am

**Payment Validation Rules:**

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Payment_Amount_Positive	Amount	Payment Amount must be greater than zero.	✓	Bharath Jonnadula, 20/09/2025, 11:59 am
Payment_Date_must_be_on	Payment Date	Payment Date cannot be earlier than the Loan Start Date.	✓	Bharath Jonnadula, 20/09/2025, 11:56 am
Payment_Status_required	Status	Please select a Payment Status.	✓	Bharath Jonnadula, 20/09/2025, 12:00 pm

Rule Name	Error Location	Error Message	Active	Modified By
AssignedAgent_Required	Assigned Agent	Assigned Agent must be selected when status is Assigned.	✓	Bharath Jonnadula, 20/09/2025, 12:04 pm
AssignedAgent_Required_ForAssigned	Top of Page	Please select an Assigned Agent when status is Assigned.	✓	Bharath Jonnadula, 20/09/2025, 12:13 pm
Recovery_Amount_Positive	Amount Recovered	Recovery Amount must be greater than zero.	✓	Bharath Jonnadula, 20/09/2025, 12:03 pm
RecoveryDate_NotFuture	Recovery Date	Recovery Date cannot be in the future.	✓	Bharath Jonnadula, 20/09/2025, 12:04 pm

## Workflow Rules

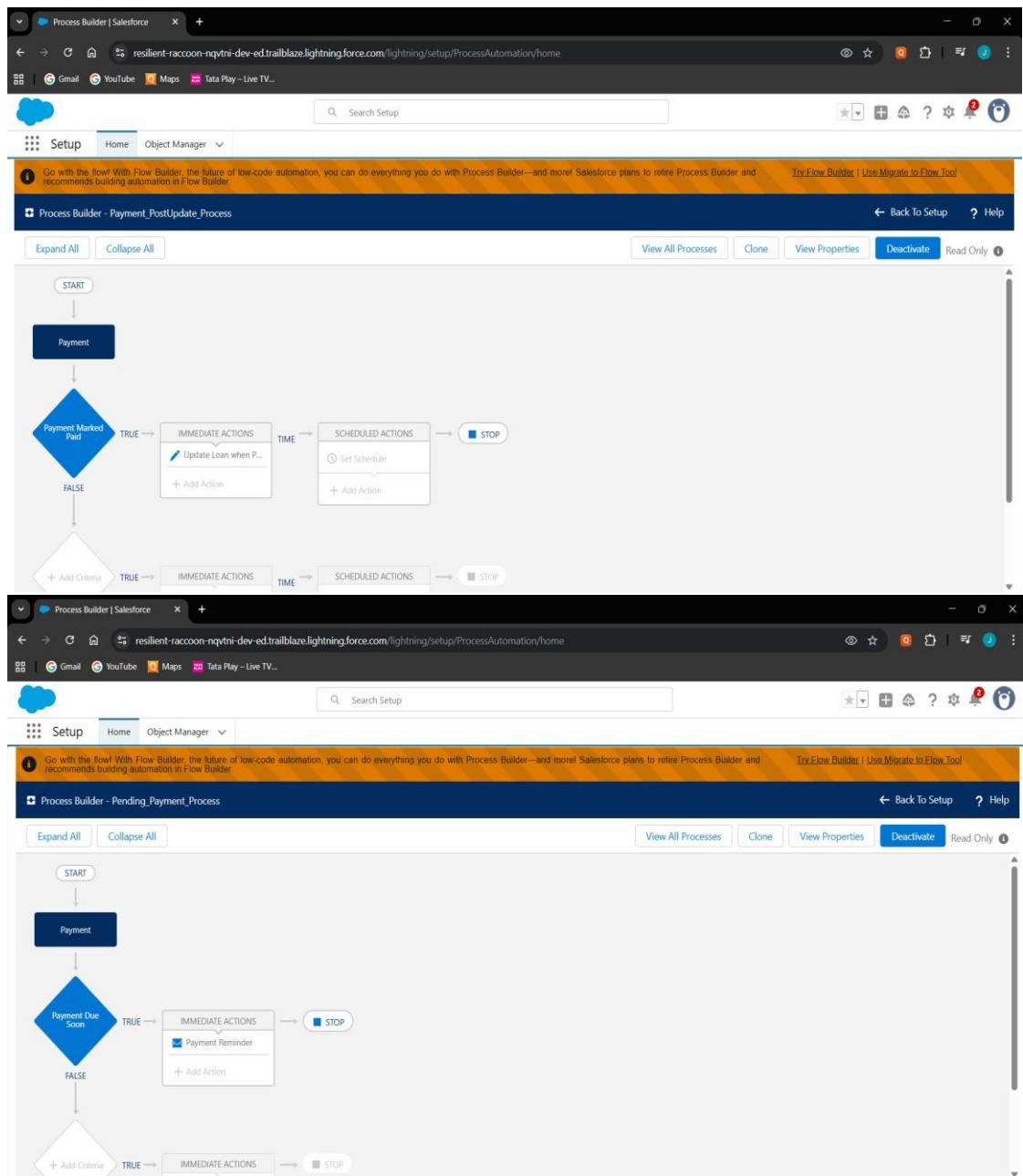
- Rule: Pending\_Payment\_Reminder.
- Sends reminder email for pending payments.

Action	Rule Name	Description	Object	Active
Edit   Del   Deactivate	Payment_Overage_Rule	created, and any time it's edited to subsequently meet criteria	Payment	✓
Edit   Del   Deactivate	Pending_Payment_Reminder		Payment	✓

## 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.

The screenshot shows the Salesforce Setup interface under the Approval Processes section. A specific approval process named "Loan: Loan\_High\_Amount\_Approval" is being edited. The process definition detail includes:

- Process Name:** Loan\_High\_Amount\_Approval
- Unique Name:** Loan\_High\_Amount\_Approval
- Description:** Loan: Principal Amount GREATER THAN 50000
- Entry Criteria:** Loan: Principal Amount GREATER THAN 50000
- Record Editability:** Administrator ONLY
- Active:** checked
- Next Automated Approver Determined By:** (empty)
- Approval Assignment Email Template:** (empty)
- Initial Submitters:** Loan Owner
- Created By:** Bharath.Jonnadula, 20/09/2025, 4:08 pm
- Modified By:** Bharath.Jonnadula, 22/09/2025, 10:49 am
- Allow Submitters to Recall Approval Requests:** unchecked

The "Initial Submission Actions" section contains two entries:

Action	Type	Description
Record Lock		Lock the record from being edited
Email Alert		Loan Submitted Email

## Flow Builder

- Record-Triggered: Marks payments overdue automatically.
- Record-Triggered: Creates Recovery record when overdue.

## Mark Payments Overdue

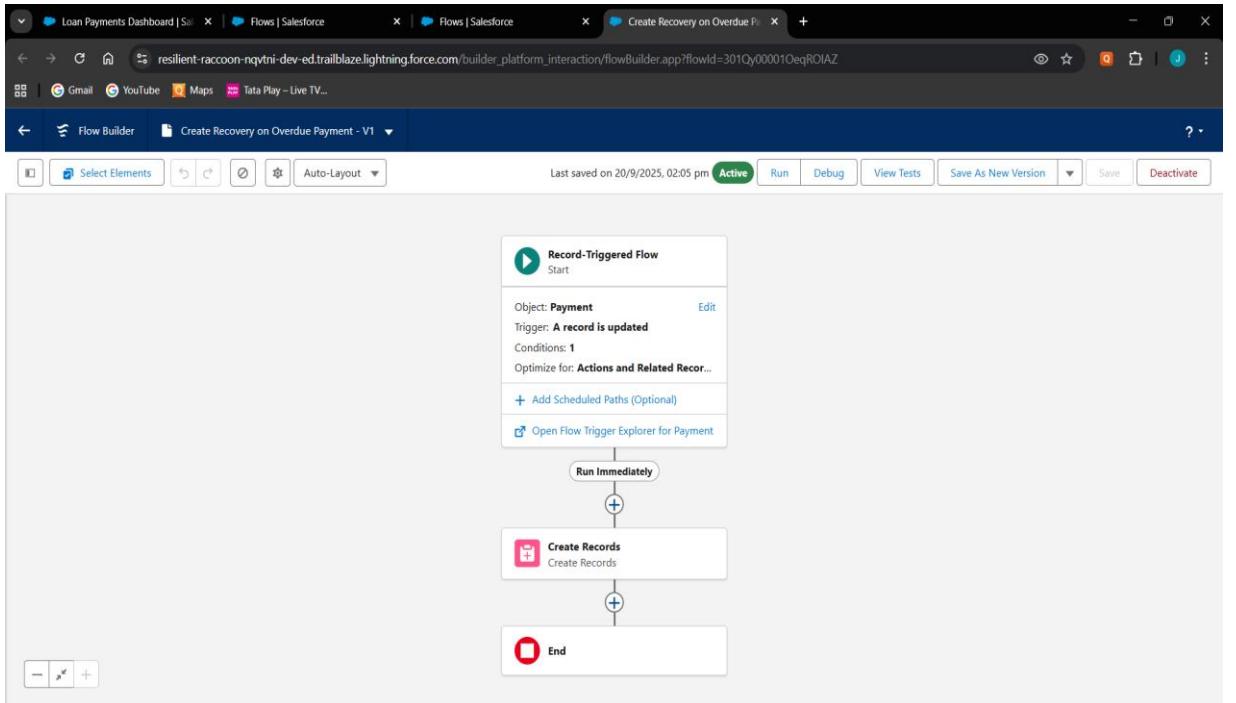
The screenshot shows the Salesforce Flow Builder interface with a flow titled "Mark Payments Overdue - V1". The flow starts with a "Record-Triggered Flow" start element for the "Payment" object, triggered by "A record is created or updated". The flow then branches into two parallel paths:

- The top path leads to a "Run Immediately" action.
- The bottom path leads to a "Mark Payment Overdue" update records action, which then connects to an "End" element.

On the right side, there are configuration panels for the start element:

- Configure Start:**
  - Select Object:** Payment
  - Trigger the Flow When:**
    - A record is created or updated
    - A record is updated
    - A record is deleted
- Set Entry Conditions:**
  - Condition Requirements:** All Conditions Are Met (AND)

## Create Recovery on Overdue



## Email Alerts

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

The screenshot shows the Salesforce Setup interface under the "Email Alerts" section. The page title is "Email Alerts". The main content area displays a table of existing email alerts:

Action	Description	Email Template Name	Object	Last Modified Date
Edit   Del	Loan_Approved_Email	Loan_Approved_Template	Loan	20/09/2025
Edit   Del	Loan_Rejected_Email	Loan_Rejected_Template	Loan	20/09/2025
Edit   Del	Loan_Submitted_Email	Loan_Submitted_Template	Loan	20/09/2025
Edit   Del	Payment_Reminder_Email	Payment_Reminder_Template	Payment	20/09/2025
Edit   Del	Send_Payment_Reminder_Email	Payment_Reminder_Template	Payment	20/09/2025

## Field Updates

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

The screenshot shows the Salesforce Setup interface under the 'Field Updates' section. The title bar reads 'Field Updates | Salesforce'. The main content area is titled 'All Workflow Field Updates' with a sub-header 'Field updates allow you to automatically change a field value to one that you specify. Field updates are actions associated with workflow rules and approval processes.' Below this is a search bar with 'View: All Workflow Field Updates' and 'Edit | Create New View'. A navigation bar at the bottom includes links for A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, and Other. The main table lists the following field updates:

Action	Name	Field to Update	Operation	Value	Last Modified Date
Edit   Del	Changes the case priority to high.	Case: Priority	Value	High	15/09/2025
Edit   Del	Mark_Payment_Overdue	Payment: Status	Value	Overdue	20/09/2025
Edit   Del	Mark_Payment_Paid	Payment: Status	Value	Paid	20/09/2025
Edit   Del	Set_Loan_Status_Approved	Loan: Status	Value	Approved	20/09/2025
Edit   Del	Set_Loan_Status_Closed	Loan: Status	Value	Closed	20/09/2025
Edit   Del	Set_Loan_Status_Rejected	Loan: Status	Value	Rejected	20/09/2025
Edit   Del	Set_Recovery_Status_Completed	Recovery: Status	Value	Completed	20/09/2025

The screenshot shows the 'Field Update Detail' page for 'Mark\_Payment\_Overdue'. The title bar includes tabs for 'Loan Payments Dashboard | Salesforce', 'Field Updates | Salesforce', and 'validate loan - V1'. The main content area is titled 'Mark\_Payment\_Overdue' with sub-sections 'Rules Using This Field Update', 'Approval Processes Using This Field Update', and 'Entitlement Processes Using This Field Update'. The 'Field Update Detail' section shows the following details:

Name	Mark_Payment_Overdue
Unique Name	Mark_Payment_Overdue
Description	Set Payment Status _c to Overdue when past due date
Object	Payment
Field to Update	Payment: Status
Field Data Type	Picklist
Re-evaluate Workflow Rules after Field Change	<input type="checkbox"/>
New Field Value	Overdue

Below this are sections for 'Rules Using This Field Update', 'Approval Processes Using This Field Update', and 'Entitlement Processes Using This Field Update'.

## **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.

#### **LoanHandler.cls**

```
public class LoanHandler {

    public static void beforeInsert(List<Loan__c>
newLoans) {
        for (Loan__c loan : newLoans) {
            if (loan.Status__c == null) {
                loan.Status__c = 'Pending Approval';
            }
        }
    }

    public static void beforeUpdate(List<Loan__c>
newLoans, Map<Id, Loan__c> oldMap) {
        for (Loan__c loan : newLoans) {
            Loan__c oldLoan = oldMap.get(loan.Id);
            if (loan.Principal_Amount__c != oldLoan.Principal_Amount__c) {
                loan.Status__c = 'Modified';
            }
        }
    }

    public static void afterInsert(List<Loan__c> newLoans)
    {
        System.debug('New Loans inserted: ' + newLoans);
    }
}
```

```
public static void afterUpdate(List<Loan__c>
newLoans, Map<Id, Loan__c> oldMap) {
    System.debug('Loans updated: ' + newLoans);
}
```

```
public static void afterDelete(List<Loan__c> oldLoans)
{
    System.debug('Loans deleted: ' + oldLoans);
}
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows the project structure under "PAYMENTTRACKINGINBANKING".
- CODE** view: Displays the Apex class `LoanHandler.cls` with its code.
- TERMINAL** view: Shows the command-line output of running the project.
- STATUS BAR**: Shows the file path "D:\PaymentTrackinginBanking", line 15, column 45, and other status indicators.

```
1  public class LoanHandler {
2
3      public static void beforeInsert(List<Loan__c> newLoans) {
4          for (Loan__c loan : newLoans) {
5              if (loan.Status__c == null) {
6                  loan.Status__c = 'Pending Approval';
7              }
8          }
9      }
10
11     public static void beforeUpdate(List<Loan__c> newLoans, Map<Id, Loan__c> oldMap) {
12         for (Loan__c loan : newLoans) {
13             Loan__c oldLoan = oldMap.get(loan.Id);
14             if (loan.Principal_Amount__c != oldLoan.Principal_Amount__c) {
15                 loan.Status__c = 'Modified';
16             }
17         }
18     }
19 }
```

```
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking>sf project deploy start --source-dir force-app/main/default/classes --target-org DevOrg
* History restored

Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.
```

## **PaymentHandler.cls**

```
public class PaymentHandler {  
  
    public static void beforeInsert(List<Payment__c> newPayments) {  
        for (Payment__c p : newPayments) {  
            if (p.Status__c == null) p.Status__c = 'Pending';  
        }  
    }  
  
    public static void beforeUpdate(List<Payment__c> newPayments,  
        Map<Id, Payment__c> oldMap) {  
        for (Payment__c p : newPayments) {  
            Payment__c oldP = oldMap.get(p.Id);  
            if (p.Amount__c != oldP.Amount__c) {  
                p.Status__c = 'Revised';  
            }  
        }  
    }  
  
    public static void afterInsert(List<Payment__c> newPayments) {  
        updateLoanStatus(newPayments);  
    }  
  
    public static void afterUpdate(List<Payment__c> newPayments,  
        Map<Id, Payment__c> oldMap) {  
        updateLoanStatus(newPayments);  
    }  
  
    public static void afterDelete(List<Payment__c> oldPayments) {  
        updateLoanStatus(oldPayments);  
    }  
  
    private static void updateLoanStatus(List<Payment__c> payments) {  
        Set<Id> loanIds = new Set<Id>();  
        for (Payment__c p : payments) if (p.Loan__c != null)  
            loanIds.add(p.Loan__c);  
        if (loanIds.isEmpty()) return;  
    }  
}
```

```

Map<Id, Integer> totalPayments = new Map<Id, Integer>();
for (AggregateResult ar : [
    SELECT Loan__c loanId, COUNT(Id) totalCount
    FROM Payment__c WHERE Loan__c IN :loanIds
    GROUP BY Loan__c
]) {
    totalPayments.put((Id) ar.get('loanId'), (Integer) ar.get('totalCount'));
}

Map<Id, Integer> paidPayments = new Map<Id, Integer>();
for (AggregateResult ar : [
    SELECT Loan__c loanId, COUNT(Id) paidCount
    FROM Payment__c WHERE Loan__c IN :loanIds AND Status__c
    = 'Paid'
    GROUP BY Loan__c
]) {
    paidPayments.put((Id) ar.get('loanId'), (Integer) ar.get('paidCount'));
}

List<Loan__c> loansToUpdate = new List<Loan__c>();
for (Id lid : loanIds) {
    Integer total = totalPayments.containsKey(lid) ?
totalPayments.get(lid) : 0;
    Integer paid = paidPayments.containsKey(lid) ?
paidPayments.get(lid) : 0;

    if (total > 0 && total == paid) {
        loansToUpdate.add(new Loan__c(Id = lid, Status__c =
'Closed'));
    } else {
        loansToUpdate.add(new Loan__c(Id = lid, Status__c =
'Active'));
    }
}
if (!loansToUpdate.isEmpty()) update loansToUpdate;

```

```
}
```

```
}
```

The screenshot shows the Salesforce IDE interface. The left sidebar displays the project structure under 'PAYMENTTRACKINGINBANKING'. The main editor window shows the code for the PaymentHandler.cls class. The code handles insertions and updates for Payment\_\_c objects, setting initial status and revising amounts if necessary. The bottom status bar indicates the file is at line 67, column 1, with 4 spaces, using UTF-8 encoding.

```
1 public class PaymentHandler {
2
3     public static void beforeInsert(List<Payment__c> newPayments) {
4         for (Payment__c p : newPayments) {
5             if (p.Status__c == null) p.Status__c = 'Pending';
6         }
7     }
8
9     public static void beforeUpdate(List<Payment__c> newPayments, Map<Id, Payment__c> oldMap) {
10        for (Payment__c p : newPayments) {
11            Payment__c oldP = oldMap.get(p.Id);
12            if (p.Amount__c != oldP.Amount__c) {
13                p.Status__c = 'Revised';
14            }
15        }
16    }
17}
```

## RecoveryHandler.cls

```
public class RecoveryHandler {

    public static void beforeInsert(List<Recovery__c> newRecoveries) {
        for (Recovery__c r : newRecoveries) {
            if (r.Status__c == null) r.Status__c = 'Assigned';
        }
    }

    public static void beforeUpdate(List<Recovery__c> newRecoveries,
                                   Map<Id, Recovery__c> oldMap) {
        for (Recovery__c r : newRecoveries) {
            Recovery__c oldR = oldMap.get(r.Id);
            if (r.Amount_Recovered__c != oldR.Amount_Recovered__c) {
                r.Status__c = 'Updated';
            }
        }
    }
}
```

```

public static void afterInsert(List<Recovery__c> newRecoveries) {
    System.debug('New Recoveries: ' + newRecoveries);
}

public static void afterUpdate(List<Recovery__c> newRecoveries,
Map<Id, Recovery__c> oldMap) {
    System.debug('Updated Recoveries: ' + newRecoveries);
}

public static void afterDelete(List<Recovery__c> oldRecoveries) {
    System.debug('Deleted Recoveries: ' + oldRecoveries);
}

```

The screenshot shows the Visual Studio Code (VS Code) interface with the following details:

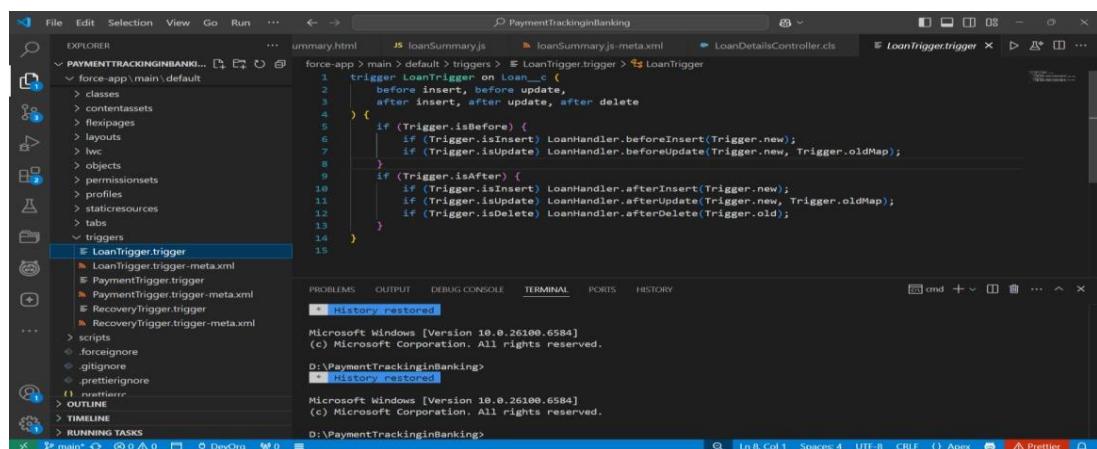
- File Explorer:** On the left, it shows the project structure under "PAYOUTTRACKINGINBANKING". The "classes" folder contains several Apex classes: PaymentController.cls, PaymentHandler.cls, PaymentHandlerTest.cls, PaymentHelper.cls, PaymentReminderQueue.cls, PaymentReminderQueue.cls-meta.xml, PaymentSearchDemo.cls, PaymentSearchDemo.cls-meta.xml, PaymentSummaryController.cls, PaymentSummaryController.cls-meta.xml, RecoveryCallout.cls, RecoveryHandler.cls, RecoveryHandler.cls-meta.xml, RecoveryHandlerTest.cls, RecoveryHelper.cls, Test\_PaymentHandlers.cls, and Test\_RecoveryHandler.cls.
- Code Editor:** The main area displays the Apex code for the RecoveryHandler class. It includes methods for beforeInsert, beforeUpdate, and afterDelete. The code handles updating the Status\_\_c field to 'Assigned' for new recoveries and updating the Status\_\_c field to 'Updated' if the Amount\_Recovered\_\_c value has changed.
- Terminal:** At the bottom, the terminal window shows the command `sf project deploy start --source-dir force-app/main/default/classes` being run, followed by the message "\* History restored".
- Bottom Status Bar:** The status bar at the bottom provides information about the file (main.sfdx), the current line (Ln 11, Col 39), and the encoding (UTF-8).

## Apex Triggers

- Implemented triggers for Loan, Payment, and Recovery to handle automation like updating statuses and roll-ups.

### LoanTrigger.trigger

```
trigger LoanTrigger on Loan__c (  
    before insert, before update,  
    after insert, after update, after delete  
) {  
  
    if (Trigger.isBefore) {  
  
        if (Trigger.isInsert) LoanHandler.beforeInsert(Trigger.new);  
  
        if (Trigger.isUpdate) LoanHandler.beforeUpdate(Trigger.new,  
            Trigger.oldMap);  
  
    }  
  
    if (Trigger.isAfter) {  
  
        if (Trigger.isInsert) LoanHandler.afterInsert(Trigger.new);  
  
        if (Trigger.isUpdate) LoanHandler.afterUpdate(Trigger.new,  
            Trigger.oldMap);  
  
        if (Trigger.isDelete) LoanHandler.afterDelete(Trigger.old);}}}
```

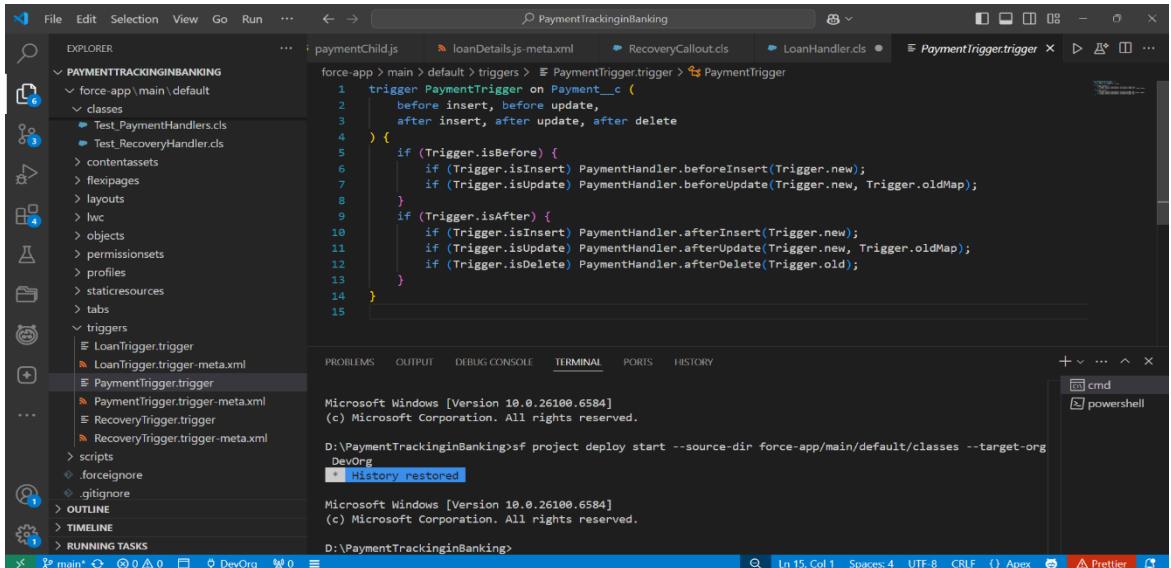


The screenshot shows the Visual Studio Code (VS Code) interface with the following details:

- Editor:** The main editor area displays the Apex code for the `LoanTrigger.trigger` trigger.
- Terminal:** Below the editor, the terminal window shows the output of running the code in a Windows environment. It includes the following text:
  - Microsoft Windows [Version 10.0.26100.6584]
  - (c) Microsoft Corporation. All rights reserved.
  - D:\PaymentTrackinginBanking> History restored
  - Microsoft Windows [Version 10.0.26100.6584]
  - (c) Microsoft Corporation. All rights reserved.
  - D:\PaymentTrackinginBanking> History restored
- Explorer:** On the left side, the Explorer view shows the project structure, including the `force-app/main/default/triggers` folder which contains the `LoanTrigger.trigger` file.
- Bottom Status Bar:** The status bar at the bottom shows the current file is `main*`, the line number is `Ln 8, Col 1`, and the character count is `Spaces: 4`.

## PaymentTrigger.trigger

```
trigger PaymentTrigger on Payment__c (
    before insert, before update,
    after insert, after update, after delete
) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert) PaymentHandler.beforeInsert(Trigger.new);
        if (Trigger.isUpdate)
            PaymentHandler.beforeUpdate(Trigger.new, Trigger.oldMap);
    }
    if (Trigger.isAfter) {
        if (Trigger.isInsert) PaymentHandler.afterInsert(Trigger.new);
        if (Trigger.isUpdate)
            PaymentHandler.afterUpdate(Trigger.new, Trigger.oldMap);
        if (Trigger.isDelete) PaymentHandler.afterDelete(Trigger.old);
    }
}
```



The screenshot shows the Salesforce Dev Console interface. The top navigation bar includes File, Edit, Selection, View, Go, Run, and various icons. The title bar says "PaymentTrackinginBanking". The left sidebar (EXPLORER) shows the project structure under "PAYMENTTRACKINGINBANKING": force-app > main > default > triggers > PaymentTrigger.trigger > PaymentTrigger. Below it are classes like Test\_PaymentHandlers.cls and Test\_RecoveryHandler.cls, and various metadata files such as contentassets, flexipages, lwc, objects, permissionsets, profiles, staticresources, tabs, and triggers (LoanTrigger.trigger, PaymentTrigger.trigger, RecoveryTrigger.trigger). The right pane displays the Apex trigger code. At the bottom, the terminal window shows the command "D:\PaymentTrackinginBanking>sf project deploy start --source-dir force-app/main/default/classes --target-org DevOrg" and the message "History restored". The status bar at the bottom indicates "Ln 15, Col 1" and "Spaces: 4 - CRLF".

## RecoveryTrigger.trigger

trigger RecoveryTrigger on Recovery\_\_c (

    before insert, before update,

    after insert, after update, after delete

) {

    if (Trigger.isBefore) {

        if (Trigger.isInsert) RecoveryHandler.beforeInsert(Trigger.new);

        if (Trigger.isUpdate)

            RecoveryHandler.beforeUpdate(Trigger.new, Trigger.oldMap);

    }

    if (Trigger.isAfter) {

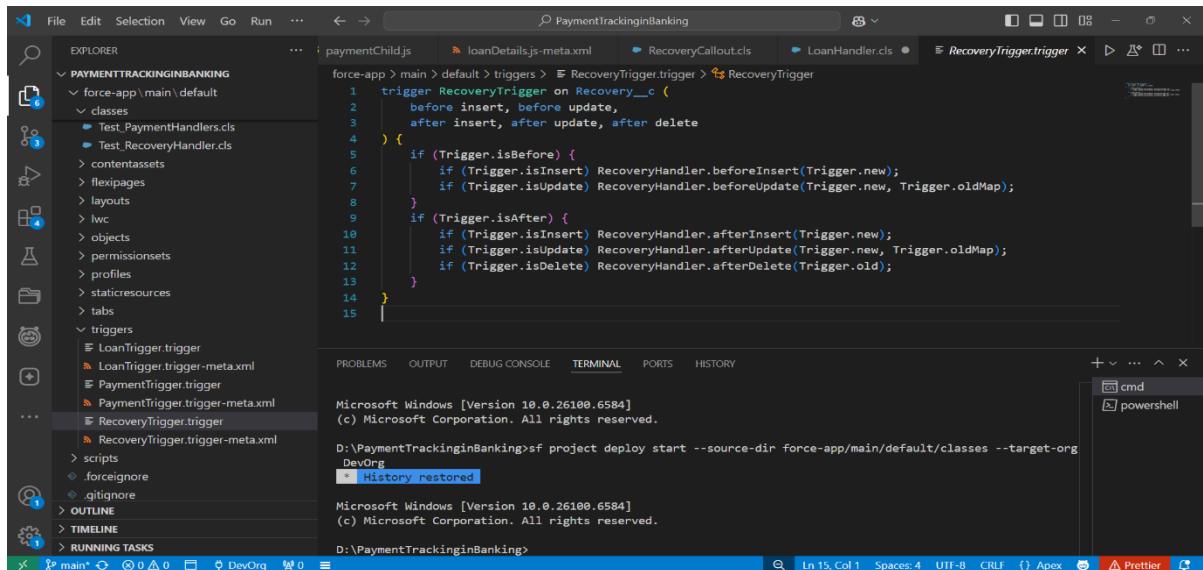
        if (Trigger.isInsert) RecoveryHandler.afterInsert(Trigger.new);

        if (Trigger.isUpdate)

            RecoveryHandler.afterUpdate(Trigger.new, Trigger.oldMap);

        if (Trigger.isDelete) RecoveryHandler.afterDelete(Trigger.old);

}



The screenshot shows the Salesforce IDE interface with the following details:

- Editor Tab:** Displays the Apex trigger code for `RecoveryTrigger`. The code handles various trigger events based on the `Trigger.isBefore` and `Trigger.isAfter` conditions.
- Terminal Tab:** Shows the command-line output of a deployment process:
  - Windows version information: Microsoft Windows [Version 10.0.26100.6584]
  - Deployment command: `D:\PaymentTrackinginBanking>sf project deploy start --source-dir force-app/main/default/classes --target-org DevOrg`
  - Deployment status: `+ History restored.`
  - Windows version information again: Microsoft Windows [Version 10.0.26100.6584]
  - Deployment command again: `D:\PaymentTrackinginBanking>`

## 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.

The screenshot shows the VS Code interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "PAYMENTTRACKINGINBANKING". It includes "force-app/main/default/classes" containing "Test\_PaymentHandlers.cls" and "Test\_RecoveryHandler.cls", and "triggers" containing "RecoveryTrigger.trigger" and "RecoveryTrigger.trigger-meta.xml".
- Code Editor:** The main editor window displays the "RecoveryTrigger.trigger" file. The code implements a trigger on the "Recovery\_\_c" object with logic for both "before insert" and "before update" events. It uses if statements to check if the trigger is before or after the event, and then calls methods on the "RecoveryHandler" class based on the specific event type (insert, update, delete).
- Terminal:** Below the code editor is a terminal window showing two command-line sessions. The first session is in a directory named "D:\PaymentTrackinginBanking>" and shows the message "\* History restored". The second session is in a directory "D:\PaymentTrackinginBanking>" and also shows "\* History restored". Both sessions are running on Microsoft Windows [Version 10.0.26100.6584] and (c) Microsoft Corporation. All rights reserved.
- Bottom Status Bar:** The status bar at the bottom shows the file path "D:\PaymentTrackinginBanking>", line number "Ln 15, Col 1", spaces count "Spaces: 4", encoding "UTF-8", and file type "Apex". There are also icons for Prettier and other development tools.

## SOQL & SOSL

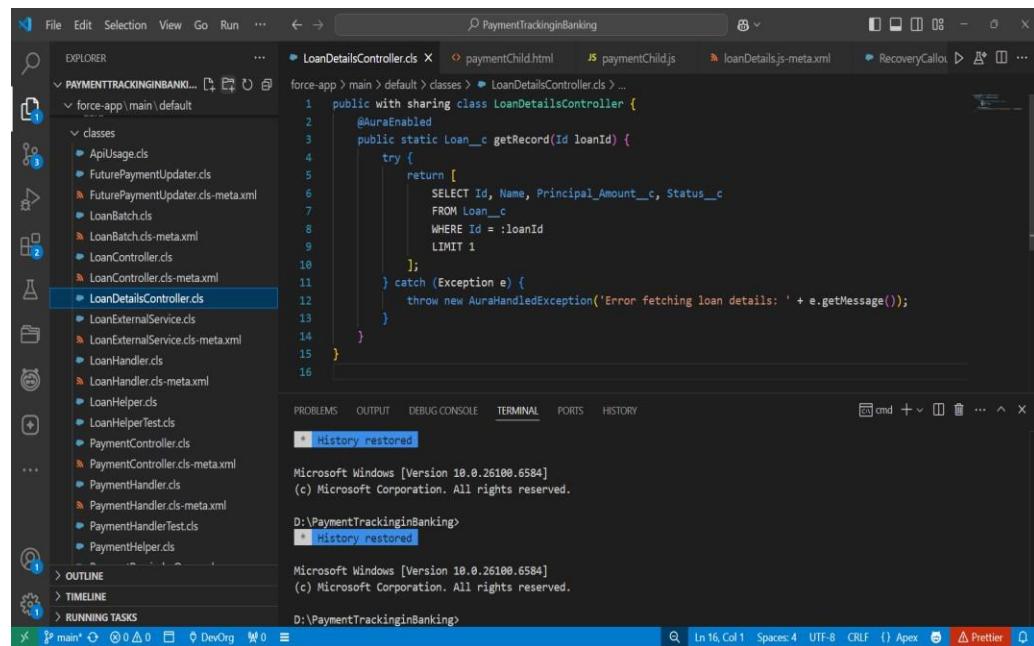
- Used SOQL in classes to query Loans, Payments, and Recoveries.
- Implemented simple search functionality to fetch customer records.

### LoanDetailsController.cls

```
public with sharing class LoanController {  
    @AuraEnabled(cacheable=true)  
    public static Loan__c getLoan(String loanId) {  
        return [SELECT Name, Principal_Amount__c FROM  
        Loan__c WHERE Id = :loanId LIMIT 1];  
    }  
}
```

```
@AuraEnabled
```

```
public static Boolean markLoanClosed(String loanId) {  
    try {  
        Loan__c L = [SELECT Id, Status__c FROM Loan__c WHERE  
        Id = :loanId LIMIT 1];  
        L.Status__c = 'Closed';  
        update L;  
        return true;  
    } catch (Exception e) {  
        throw new AuraHandledException('Cannot close loan: ' +  
        e.getMessage());  
    }  
}
```



## PaymentsController.cls

```
public class PaymentController {  
    @AuraEnabled(cacheable=true)  
    public static List<Payment__c> getPayments(Id loanId) {  
        if (loanId == null) return new List<Payment__c>();  
        return [  
            SELECT Name, Amount__c, Status__c,  
            Payment_Date__c  
            FROM Payment__c  
            WHERE Loan__c = :loanId  
            ORDER BY Payment_Date__c DESC  
        ];  
    }  
}
```

The screenshot shows the Visual Studio Code (VS Code) interface with the following details:

- Editor:** The main editor area displays the `PaymentController.cls` file.
- Explorer:** The left sidebar shows the project structure under `PAYMENTTRACKINGINBANKING`, including files like `paymentChild.js`, `loanDetails.js-meta.xml`, `RecoveryCallout.cls`, `LoanHandler.cls`, and `PaymentController.cls`.
- Terminal:** The bottom right terminal window shows the command-line output of a deployment process:

```
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking>sf project deploy start --source-dir force-app/main/default/classes --target-org DevOrg
+ History restored

Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

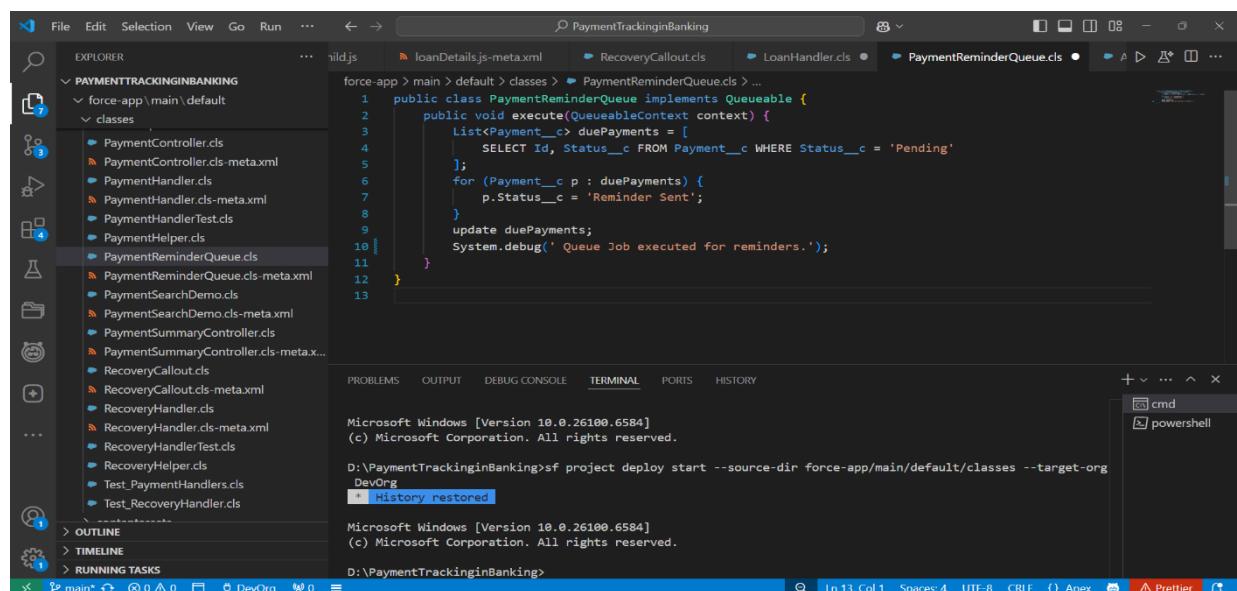
D:\PaymentTrackinginBanking>
```

## Collections: List, Set, Map

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

### PaymentReminderQueue.cls

```
public class PaymentReminderQueue implements Queueable {
    public void execute(QueueableContext context) {
        List<Payment__c> duePayments = [
            SELECT Id, Status__c FROM Payment__c WHERE
            Status__c = 'Pending'
        ];
        for (Payment__c p : duePayments) {
            p.Status__c = 'Reminder Sent';
        }
        update duePayments;
        System.debug(' Queue Job executed for reminders.');
    }
}
```



## Control Statements

- Implemented IF conditions for validation.

### RecoveryHandler.cls

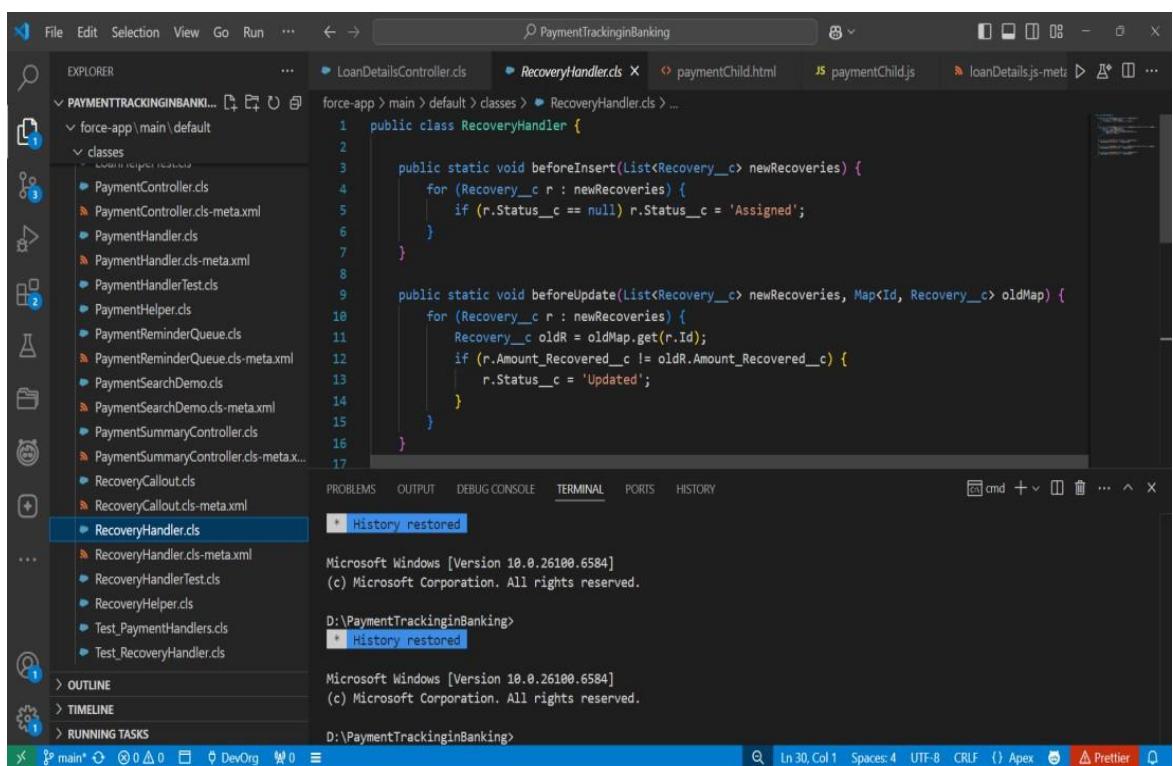
```
public class RecoveryHandler {  
  
    public static void beforeInsert(List<Recovery__c> newRecoveries) {  
  
        for (Recovery__c r : newRecoveries) {  
  
            if (r.Status__c == null) r.Status__c = 'Assigned';  
  
        }  
  
    }  
  
  
    public static void beforeUpdate(List<Recovery__c> newRecoveries,  
        Map<Id, Recovery__c> oldMap) {  
  
        for (Recovery__c r : newRecoveries) {  
  
            Recovery__c oldR = oldMap.get(r.Id);  
  
            if (r.Amount_Recovered__c != oldR.Amount_Recovered__c) {  
  
                r.Status__c = 'Updated';  
  
            }  
  
        }  
  
    }  
  
  
    public static void afterInsert(List<Recovery__c> newRecoveries) {  
  
        System.debug('New Recoveries: ' + newRecoveries);  
  
    }  
}
```

```

public static void afterUpdate(List<Recovery__c> newRecoveries,
Map<Id, Recovery__c> oldMap) {
    System.debug('Updated Recoveries: ' + newRecoveries);
}

public static void afterDelete(List<Recovery__c> oldRecoveries) {
    System.debug('Deleted Recoveries: ' + oldRecoveries);
}

```



```

1  public class RecoveryHandler {
2
3      public static void beforeInsert(List<Recovery__c> newRecoveries) {
4          for (Recovery__c r : newRecoveries) {
5              if (r.Status__c == null) r.Status__c = 'Assigned';
6          }
7      }
8
9      public static void beforeUpdate(List<Recovery__c> newRecoveries, Map<Id, Recovery__c> oldMap) {
10         for (Recovery__c r : newRecoveries) {
11             Recovery__c oldR = oldMap.get(r.Id);
12             if (r.Amount_Recovered__c != oldR.Amount_Recovered__c) {
13                 r.Status__c = 'Updated';
14             }
15         }
16     }
17 }

```

The screenshot shows the VS Code interface with the following details:

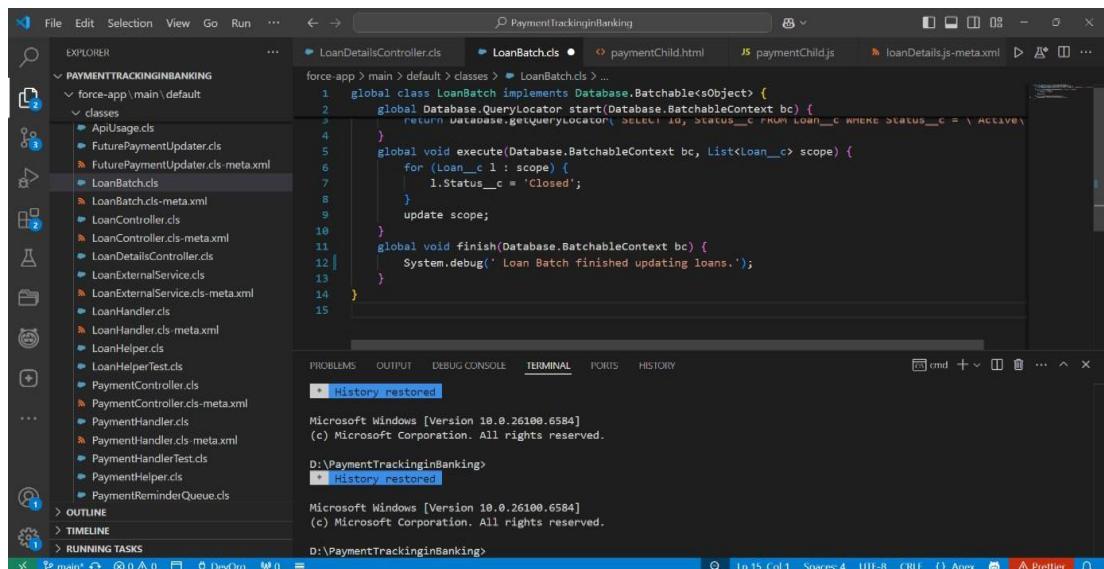
- File Explorer:** Shows the project structure under "PAYOUTTRACKINGINBANKING". The "classes" folder contains several Apex classes like PaymentController.cls, PaymentHandler.cls, etc.
- Editor:** The main pane displays the "RecoveryHandler.cls" file with the provided Apex code.
- Terminal:** The bottom terminal window shows command-line history for two separate runs on a Windows 10 system.

## Batch Apex

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

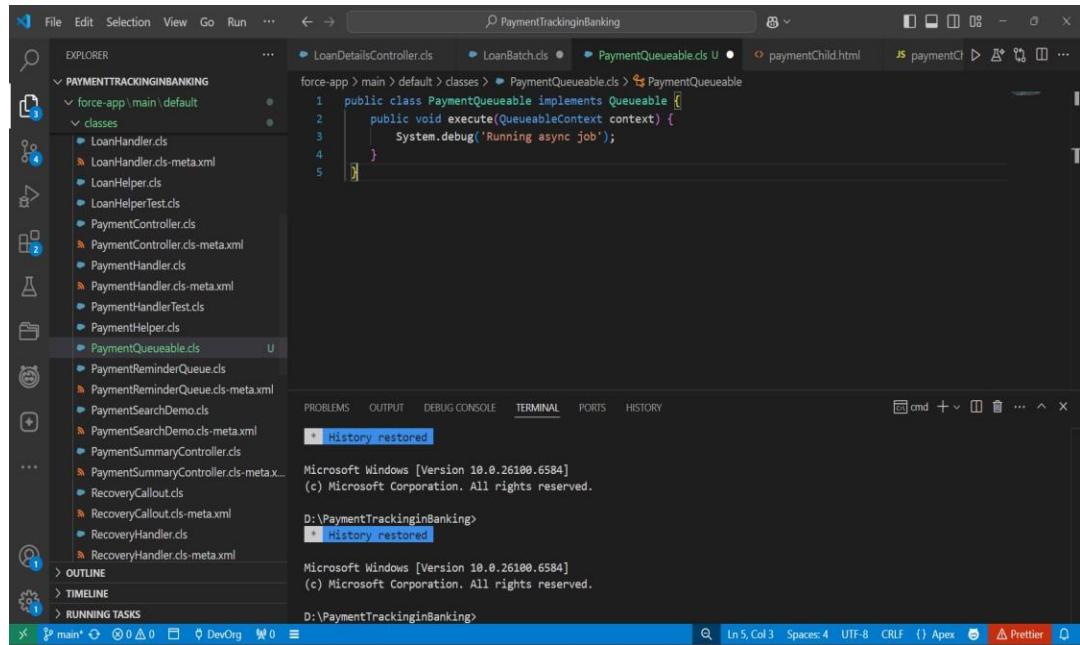
### LoanBatch.cls

```
global class LoanBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc)
    {
        return Database.getQueryLocator('SELECT Id, Status__c FROM
Loan__c WHERE Status__c = \'Active\'');
    }
    global void execute(Database.BatchableContext bc, List<Loan__c>
scope) {
        for (Loan__c l : scope) {
            l.Status__c = 'Closed';
        }
        update scope;
    }
    global void finish(Database.BatchableContext bc) {
        System.debug(' Loan Batch finished updating loans.');
    }
}
```



## Queueable Apex

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



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PAYMENTTRACKINGINBANKING". The "classes" folder contains several files, with "PaymentQueueable.cls" currently selected.
- Code Editor:** Displays the code for "PaymentQueueable.cls":

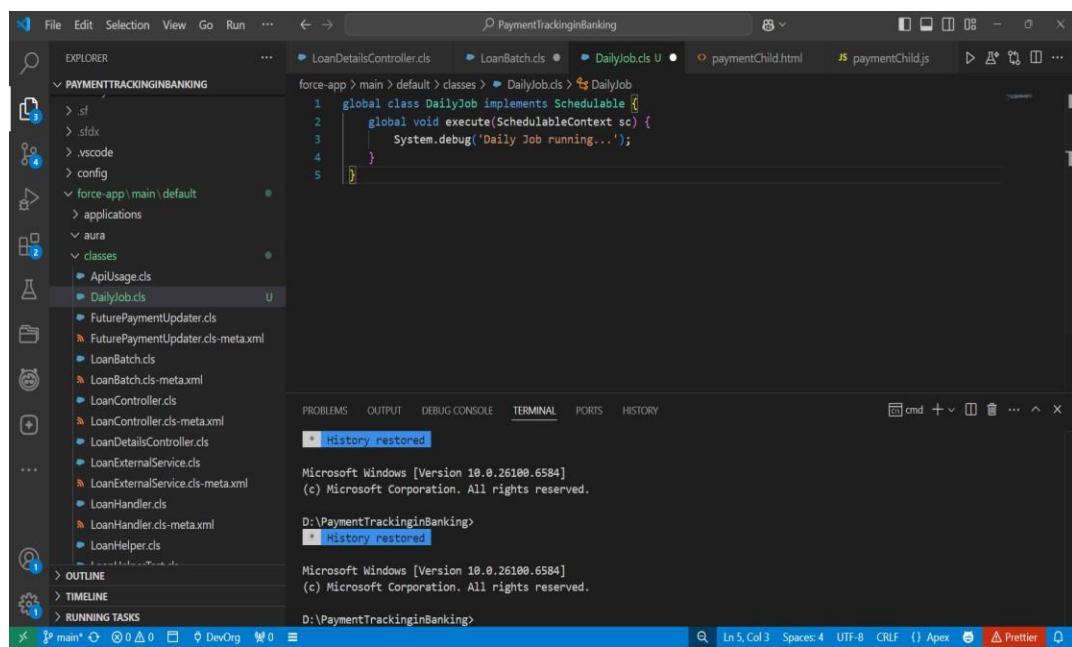
```
1  public class PaymentQueueable implements Queueable {
2      public void execute(QueueableContext context) {
3          System.debug('Running async job');
4      }
}
```
- Terminal:** Shows the command line history:

```
* History restored
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking
* History restored
```
- Status Bar:** Shows the file path "D:\PaymentTrackinginBanking", line "Ln 5, Col 3", spaces "Spaces: 4", encoding "UTF-8", and line endings "CRLF". It also indicates the file is "Apex".

## Scheduled Apex

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



The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PAYMENTTRACKINGINBANKING". The "classes" folder contains several files, with "DailyJob.cls" currently selected.
- Code Editor:** Displays the code for "DailyJob.cls":

```
1  global class DailyJob implements Schedulable {
2      global void execute(SchedulableContext sc) {
3          System.debug('Daily Job running...');
4      }
}
```
- Terminal:** Shows the command line history:

```
* History restored
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking
* History restored
```
- Status Bar:** Shows the file path "D:\PaymentTrackinginBanking", line "Ln 5, Col 3", spaces "Spaces: 4", encoding "UTF-8", and line endings "CRLF". It also indicates the file is "Apex".

## 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.

### **LoanHelper.cls**

```
public class LoanHelper {  
  
    public static void beforeInsert(List<Loan__c> newLoans) {  
  
        try {  
  
            for (Loan__c loan : newLoans) {  
  
                if (loan.Amount__c <= 0) {  
  
                    throw new PaymentTrackingException('Loan Amount must  
be greater than 0.');                }  
  
            }  
  
        } catch (Exception ex) {  
  
            String msg = ErrorHandler.logError(ex,  
'LoanHelper.beforeInsert');  
  
            ErrorHandler.notifyAdmin('Loan Insert Error', msg);  
  
            throw ex;  
  
        }  
  
    }  
}
```

```

1  public class LoanHelper {
2      public static void beforeInsert(List<Loan__c> newLoans) {
3          try {
4              for (Loan__c loan : newLoans) {
5                  if (loan.Amount__c <= 0) {
6                      throw new PaymentTrackingException('Loan Amount must be greater than 0.');
7                  }
8              }
9          } catch (Exception ex) {
10             String msg = ErrorHandler.logError(ex, 'LoanHelper.beforeInsert');
11             ErrorHandler.notifyAdmin('Loan Insert Error', msg);
12             throw ex;
13         }
14     }
15 }
16

```

TERMINAL

```

Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking>sf project deploy start --source-dir force-app/main/default/classes --target-org
DevOrg
+ History restored

Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking>

```

## PaymentHelper.cls

```

public class PaymentHandler {
    public static void beforeInsert(List<Payment__c> newPayments) {
        try {
            for (Payment__c pay : newPayments) {
                if (pay.Payment_Date__c == null) {
                    throw new PaymentTrackingException('Payment Date is required.');
                }
            }
        } catch (Exception ex) {
            String msg = ErrorHandler.logError(ex,
'PaymentHandler.beforeInsert');
            ErrorHandler.notifyAdmin('Payment Insert Error', msg);
            throw ex;
        }
    }

    public static void afterUpdate(List<Payment__c> updatedPayments)
    {
        try {

```

```

        for (Payment__c pay : updatedPayments) {
            if (pay.Status__c == 'Overdue') {
                System.debug('Payment ' + pay.Id + ' is overdue. Creating
recovery...');
            }
        }
    } catch (Exception ex) {
        String msg = ErrorHandler.logError(ex,
'PaymentHandler.afterUpdate');
        ErrorHandler.notifyAdmin('Payment Update Error', msg);
    }
}
}

```

The screenshot shows the Salesforce Dev Console interface. On the left is the Explorer sidebar with project files like LoanBatch.cls, PaymentHelper.cls, and Test\_PaymentHandlers.cls. The main area displays the code for PaymentHelper.cls:

```

1  public class PaymentHandler {
2      public static void beforeInsert(List<Payment__c> newPayments) {
3          try {
4              for (Payment__c pay : newPayments) {
5                  if (pay.Payment_Date__c == null) {
6                      throw new PaymentTrackingException('Payment Date is required.');
7                  }
8              }
9          } catch (Exception ex) {
10             String msg = ErrorHandler.logError(ex, 'PaymentHandler.beforeInsert');
11             ErrorHandler.notifyAdmin('Payment Insert Error', msg);
12             throw ex;
13         }
14     }
15
16     public static void afterUpdate(List<Payment__c> updatedPayments) {
17         try {

```

Below the code, the terminal window shows deployment logs:

```

Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking>sf project deploy start --source-dir force-app/main/default/classes --target-org
DevOrg
[*] History restored

Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking>
```

## Test Classes

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

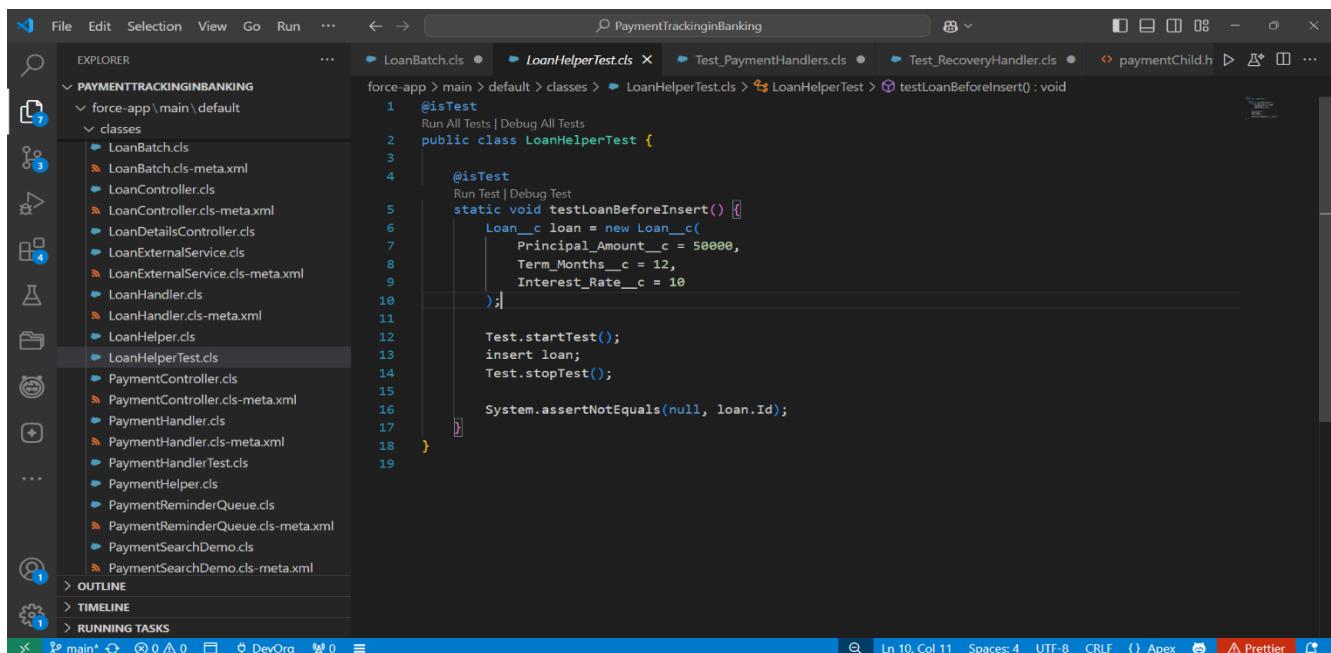
### LoanHelperTest.cls

```
@isTest
public class LoanHelperTest {

    @isTest
    static void testLoanBeforeInsert() {
        Loan__c loan = new Loan__c(
            Principal_Amount__c = 50000,
            Term_Months__c = 12,
            Interest_Rate__c = 10
        );

        Test.startTest();
        insert loan;
        Test.stopTest();

        System.assertEquals(null, loan.Id);
    }
}
```



The screenshot shows the Salesforce IDE interface with the following details:

- File Path:** force-app/main/default/classes/LoanHelperTest.cls
- Code Editor Content:**

```
1  @isTest
2  public class LoanHelperTest {
3
4      @isTest
5      static void testLoanBeforeInsert() {
6          Loan__c loan = new Loan__c(
7              Principal_Amount__c = 50000,
8              Term_Months__c = 12,
9              Interest_Rate__c = 10
10         );
11
12         Test.startTest();
13         insert loan;
14         Test.stopTest();
15
16         System.assertEquals(null, loan.Id);
17     }
18 }
```
- Explorer View:** Shows the project structure under PAYMENTTRACKINGINBANKING, including files like LoanBatch.cls, LoanController.cls, PaymentController.cls, etc.
- Toolbar:** Standard IDE toolbar with icons for File, Edit, Selection, View, Go, Run, etc.
- Status Bar:** Shows the current file (main), line count (Ln 10, Col 11), spaces (Spaces: 4), encoding (UTF-8), CRLF, Apex, and Prettier buttons.

## **Test\_PaymentHandlers.cls**

```
@IsTest
```

```
private class Test_PaymentHandlers {
```

```
    @IsTest static void testPaymentInsertAndLoanClose() {
```

```
        Loan__c loan = new Loan__c(Name='TestLoan',  
        Principal_Amount__c = 1000.00, Status__c='Active');  
        insert loan;
```

```
        Payment__c p1 = new Payment__c(Name='P1', Loan__c = loan.Id,  
        Payment_Date__c = Date.today().addDays(1), Amount__c = 500,  
        Status__c='Pending');
```

```
        Payment__c p2 = new Payment__c(Name='P2', Loan__c = loan.Id,  
        Payment_Date__c = Date.today().addDays(2), Amount__c = 500,  
        Status__c='Pending');
```

```
        insert new List<Payment__c>{p1,p2};
```

```
        p1.Status__c = 'Paid';
```

```
        p2.Status__c = 'Paid';
```

```
        update new List<Payment__c>{p1,p2};
```

```
        Loan__c refreshed = [SELECT Id, Status__c FROM Loan__c WHERE  
        Id = :loan.Id];
```

```
        System.assertEquals('Closed', refreshed.Status__c);
```

```
}
```

```
@IsTest static void testPaymentOverdueBeforeInsert() {
```

```
    Loan__c loan = new Loan__c(Name='L2',  
    Principal_Amount__c=1000.00);  
    insert loan;
```

```

        Payment__c p = new Payment__c(Name='past', Loan__c =
loan.Id, Payment_Date__c = Date.today().addDays(-2), Amount__c
= 100, Status__c=null);
        insert p;
        Payment__c got = [SELECT Id, Status__c FROM Payment__c
WHERE Id = :p.Id];
        System.assertEquals('Overdue', got.Status__c);
    }
}

```

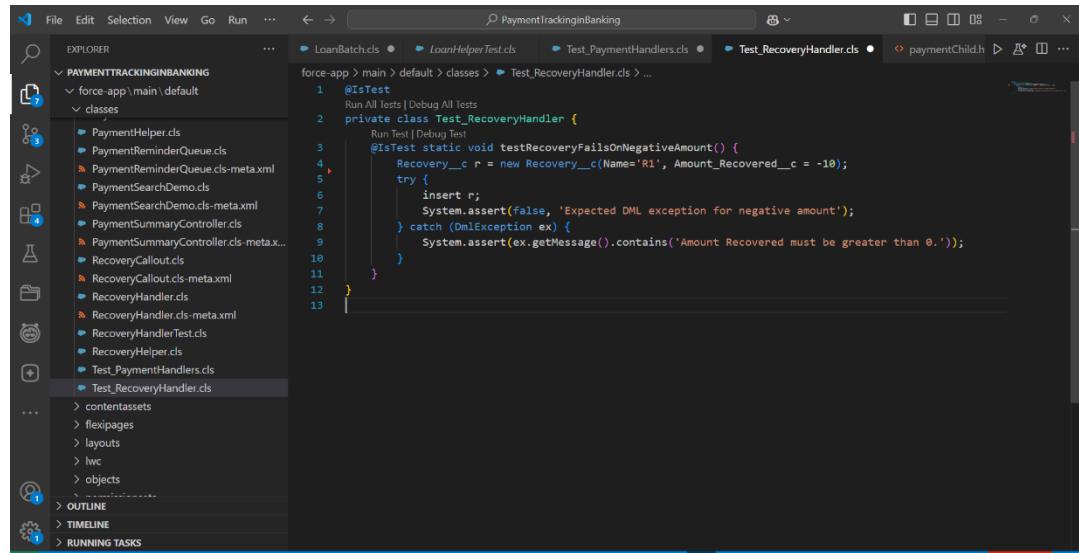
```

1  @IsTest
2  private class Test_PaymentHandlers {
3      @IsTest static void testPaymentInsertAndLoanClose() {
4          Loan__c loan = new Loan__c(Name='TestLoan', Principal_Amount__c = 1000.00, Status__c='Active');
5          insert loan;
6
7          Payment__c p1 = new Payment__c(Name='P1', Loan__c = loan.Id, Payment_Date__c = Date.today().addDays(-2));
8          Payment__c p2 = new Payment__c(Name='P2', Loan__c = loan.Id, Payment_Date__c = Date.today().addDays(-2));
9
10         insert new List<Payment__c>{p1,p2};
11
12         p1.Status__c = 'Paid';
13         p2.Status__c = 'Paid';
14         update new List<Payment__c>{p1,p2};
15
16         Loan__c refreshed = [SELECT Id, Status__c FROM Loan__c WHERE Id = :loan.Id];
17         System.assertEquals('Closed', refreshed.Status__c);
18     }
19
20     @IsTest static void testPaymentOverdueBeforeInsert() {
21         Loan__c loan = new Loan__c(Name='L2', Principal_Amount__c=1000.00);
22         insert loan;
23
24         Payment__c p = new Payment__c(Name='past', Loan__c = loan.Id, Payment_Date__c = Date.today().addDays(-2));
25
26         insert p;
27     }
}

```

## Test\_RecoveryHandlers.cls

```
@IsTest
private class Test_RecoveryHandler {
    @IsTest static void testRecoveryFailsOnNegativeAmount() {
        Recovery__c r = new Recovery__c(Name='R1',
        Amount_Recovered__c = -10);
        try {
            insert r;
            System.assert(false, 'Expected DML exception for negative
amount');
        } catch (DmlException ex) {
            System.assert(ex.getMessage().contains('Amount Recovered
must be greater than 0.'));
        }
    }
}
```



```
}
```

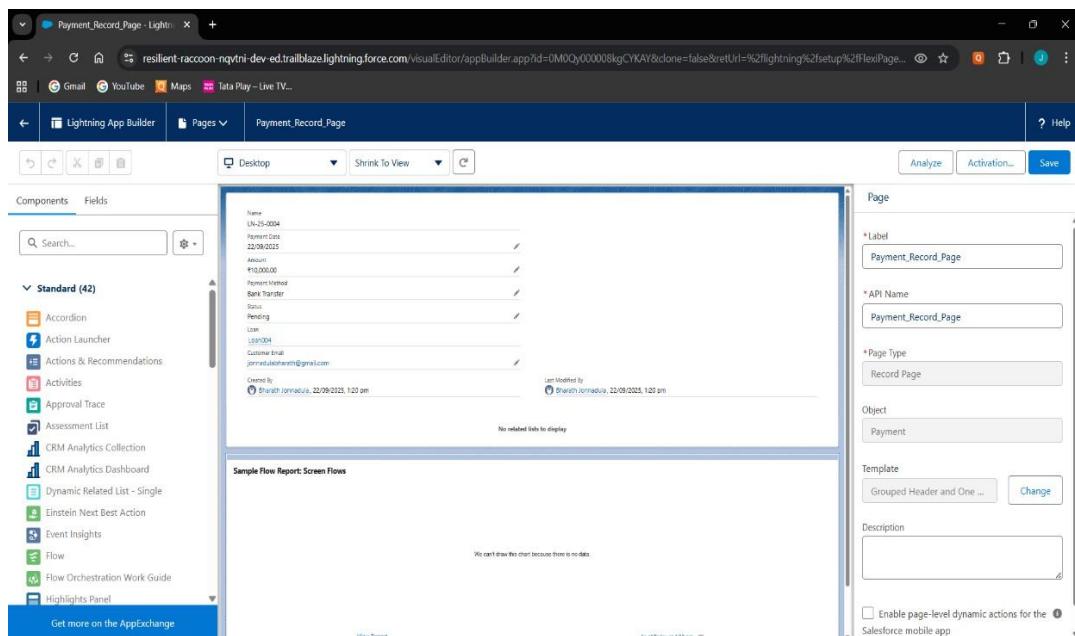
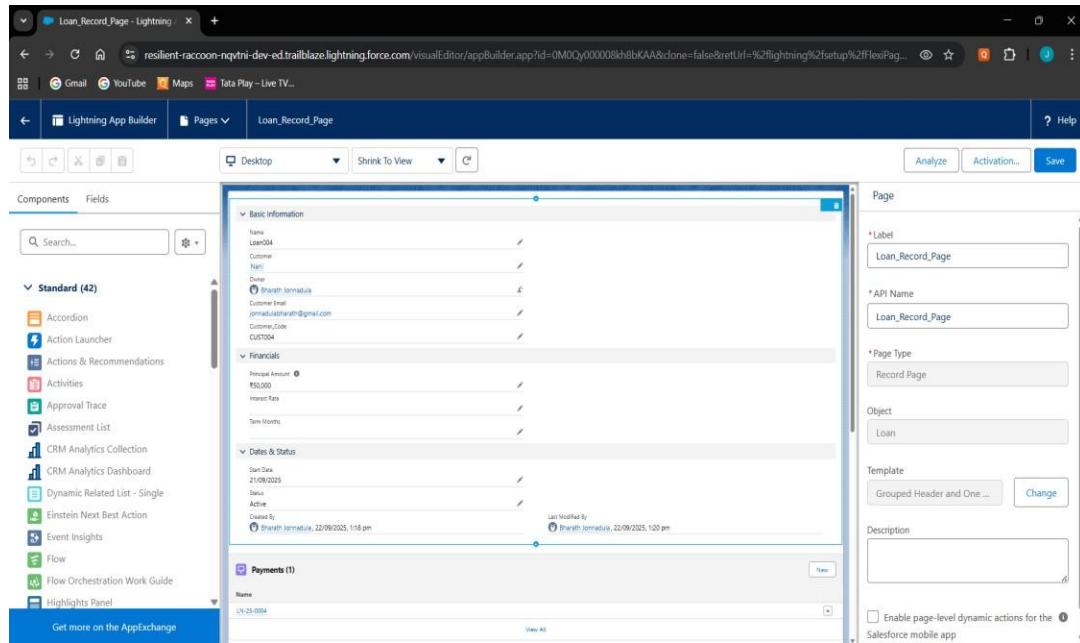
## Asynchronous Processing

- Combined Batch, Queueable, Scheduled, and Future methods for async operations.

## Phase 6: User Interface Development

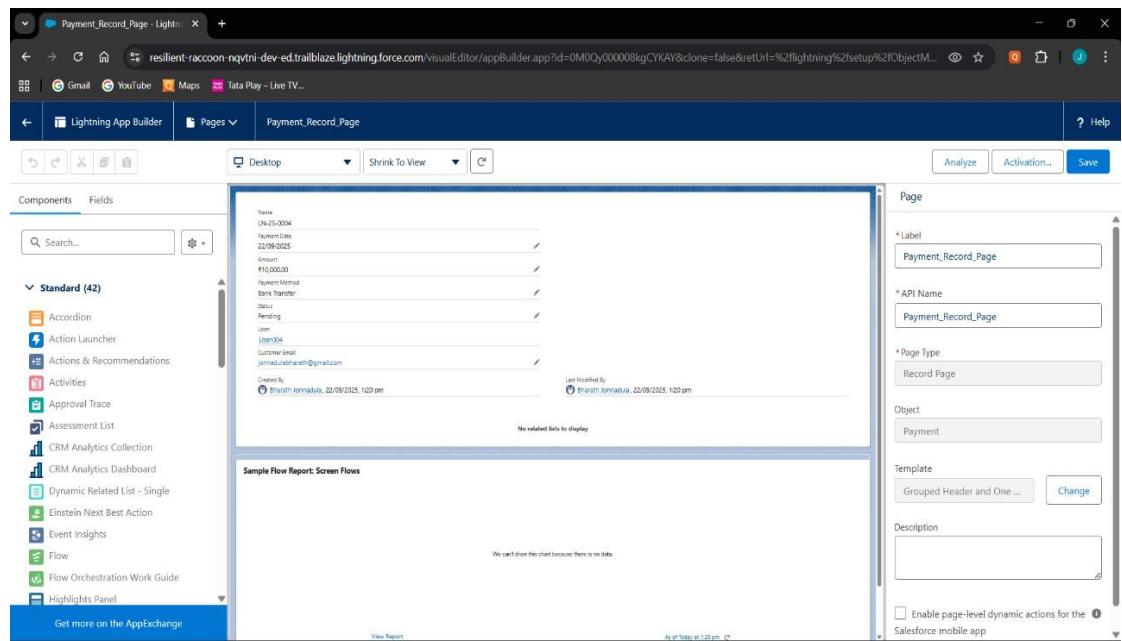
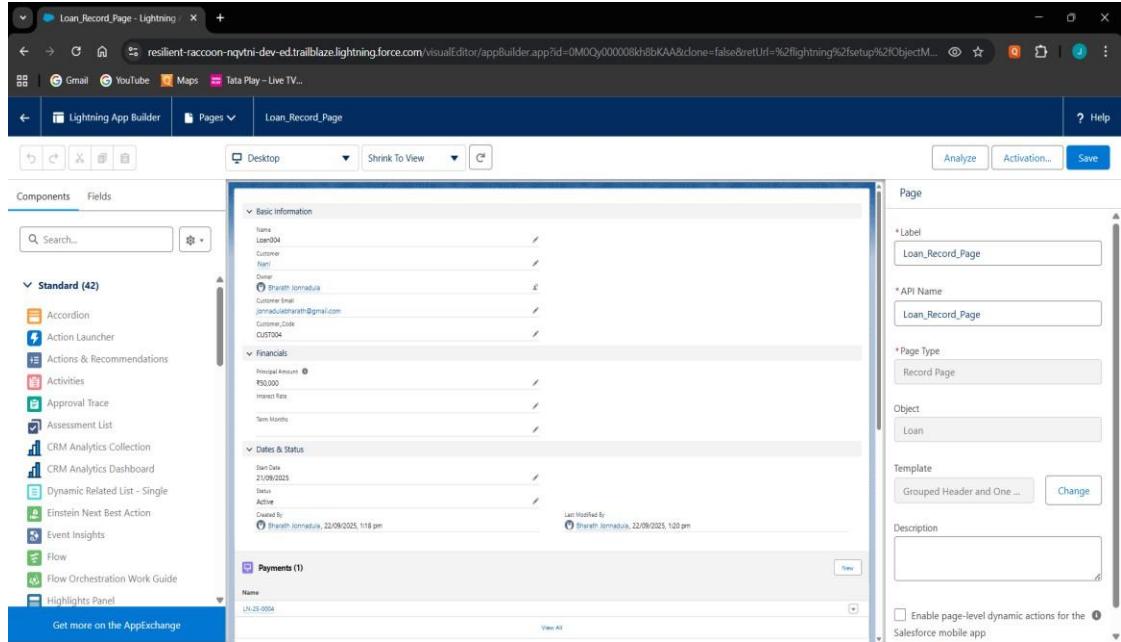
### Lightning App Builder

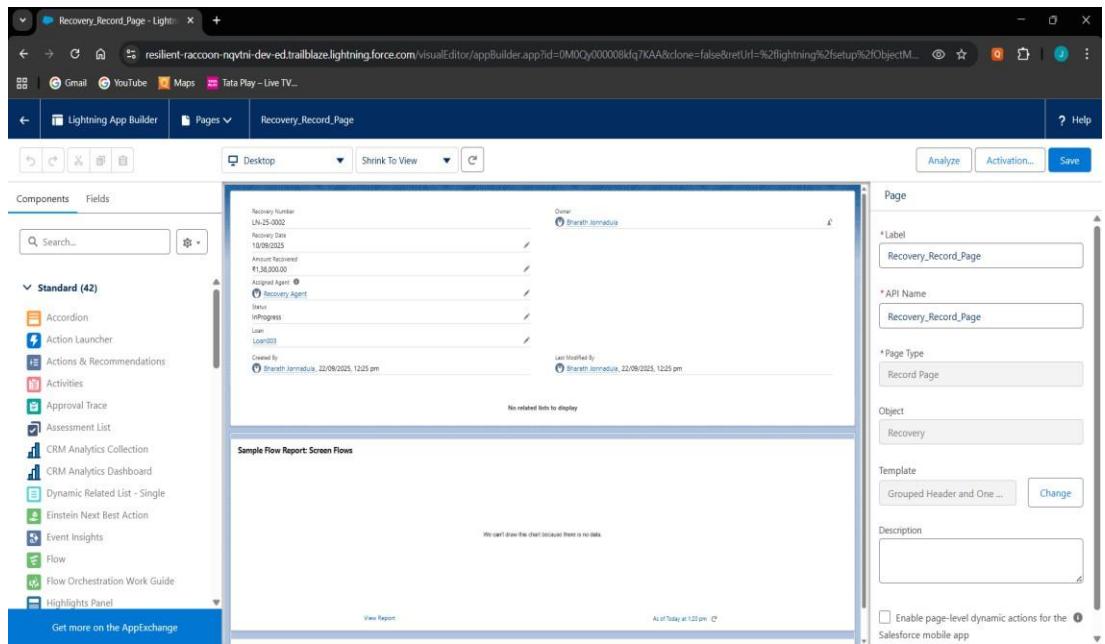
- Created a custom Loan\_Record\_Page, Payment\_Home\_Page, Payment\_Record\_Page, Recovery\_record\_Page



## Record Pages

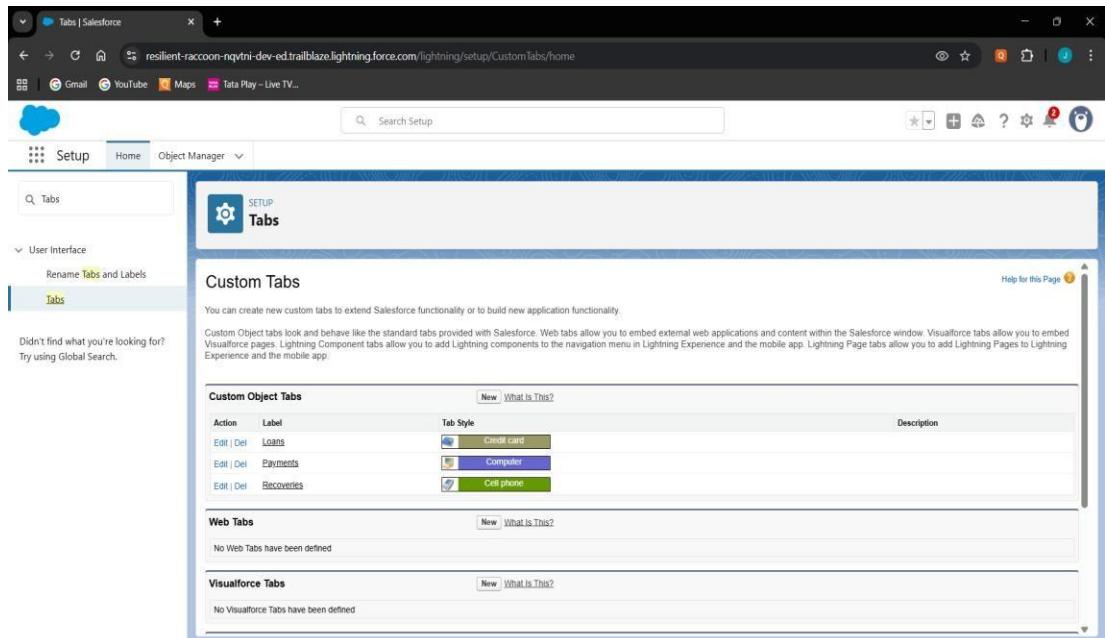
- Built custom Record Pages for Loan and Payment objects using Lightning App Builder.





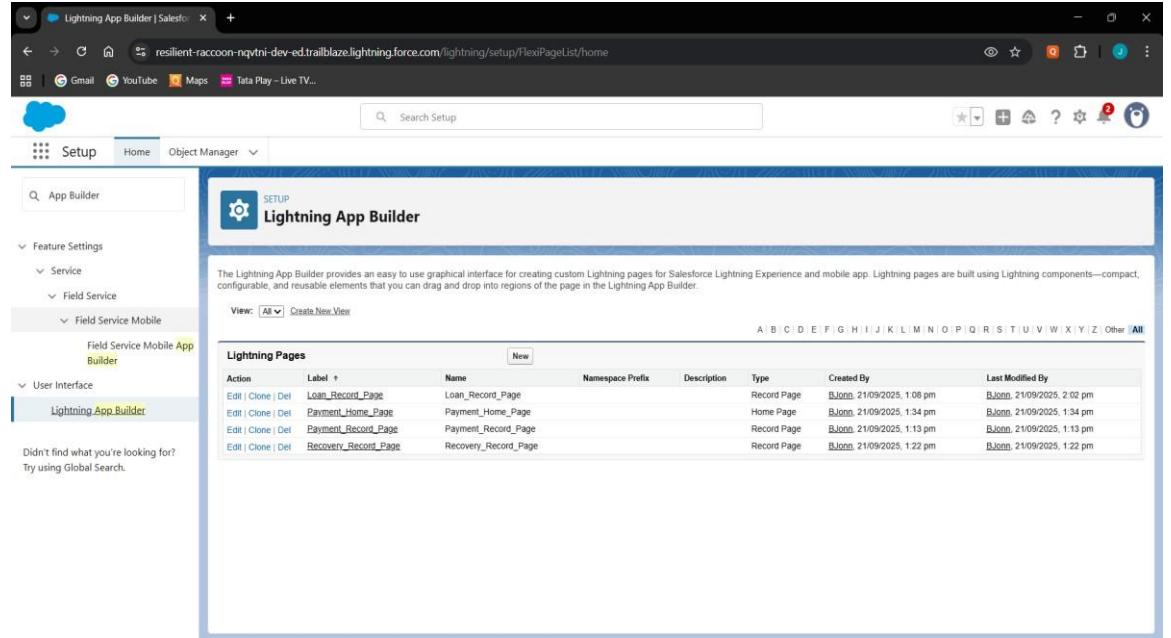
## Tabs

- Added custom tabs for Loan, Payment, and Recovery objects.



## Home Page Layouts

- Customized the Home Page with components like Recent Loans, Pending Payments

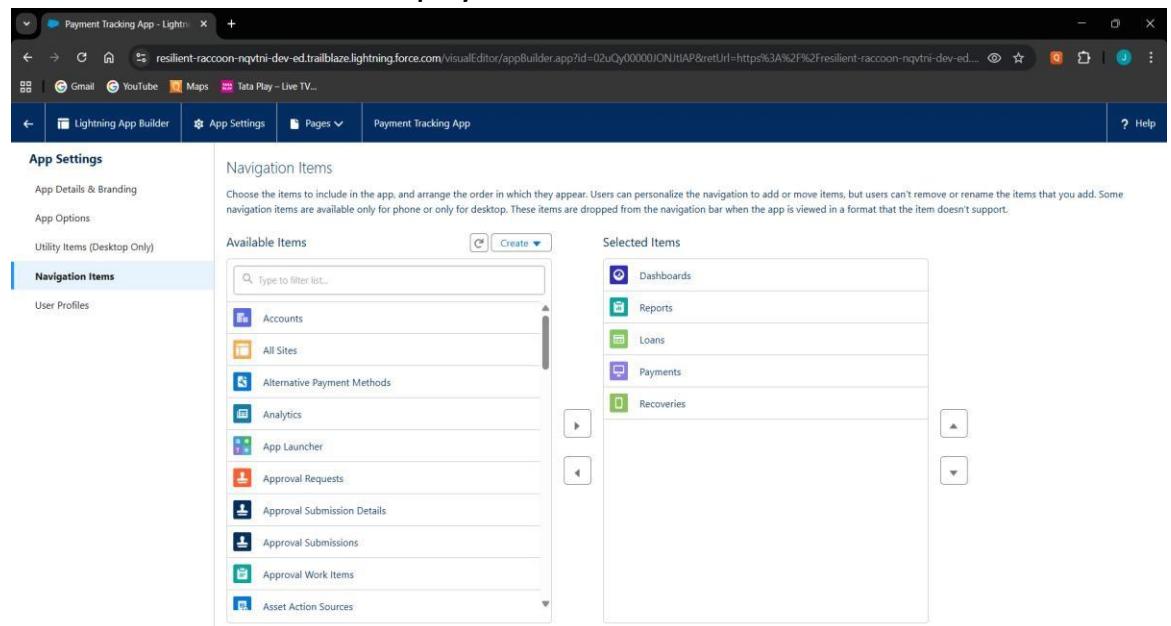


The screenshot shows the Salesforce Lightning App Builder interface. On the left, there's a sidebar with navigation links: Setup, Home, Object Manager, App Builder, Feature Settings, Service, Field Service, Field Service Mobile, Field Service Mobile App Builder, User Interface, and Lightning App Builder. The 'Lightning App Builder' link is highlighted. Below the sidebar, a search bar says 'Search Setup'. The main area has a title 'Lightning App Builder' with a gear icon. A sub-header says 'The Lightning App Builder provides an easy to use graphical interface for creating custom Lightning pages for Salesforce Lightning Experience and mobile app. Lightning pages are built using Lightning components—compact, configurable, and reusable elements that you can drag and drop into regions of the page in the Lightning App Builder.' Below this, there's a 'View' dropdown set to 'All' and a 'Create New View' button. A table titled 'Lightning Pages' lists several pages:

Action	Label	Name	Namespace Prefix	Description	Type	Created By	Last Modified By
Edit   Clone   Del	Loan_Record_Page	Loan_Record_Page			Record Page	Bjorn	21/09/2025, 1:08 pm
Edit   Clone   Del	Payment_Home_Page	Payment_Home_Page			Home Page	Bjorn	21/09/2025, 1:34 pm
Edit   Clone   Del	Payment_Record_Page	Payment_Record_Page			Record Page	Bjorn	21/09/2025, 1:13 pm
Edit   Clone   Del	Recovery_Record_Page	Recovery_Record_Page			Record Page	Bjorn	21/09/2025, 1:22 pm

## Utility Bar

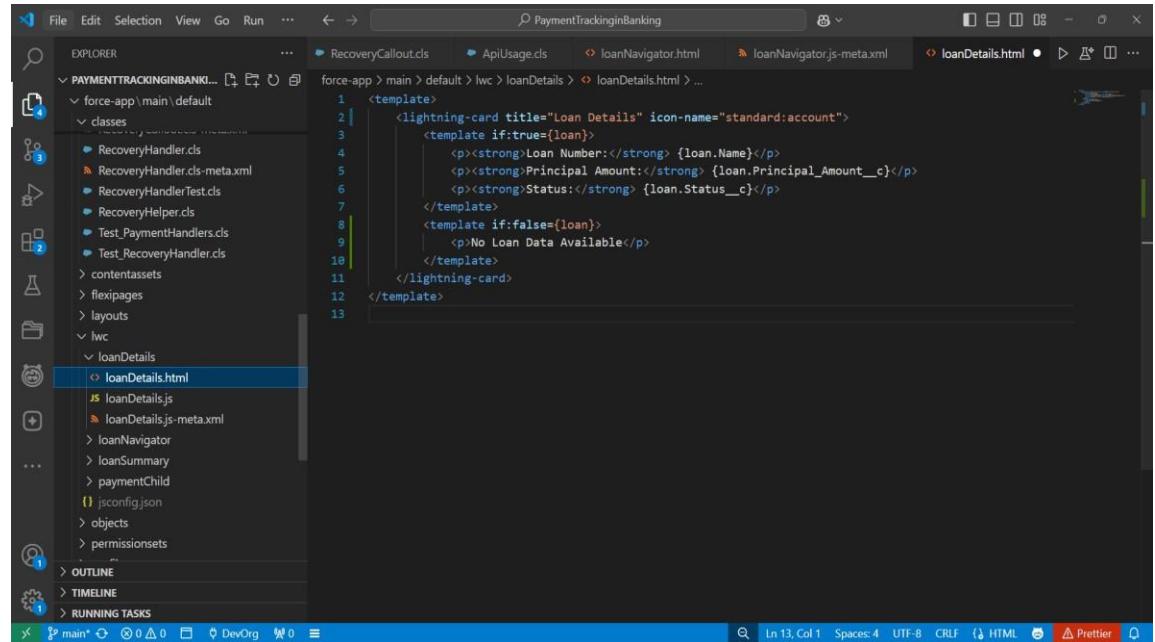
- Switch between loan & payment records.



The screenshot shows the 'App Settings' screen for a 'Payment Tracking App'. At the top, there are tabs for 'Lightning App Builder', 'App Settings', 'Pages', and 'Payment Tracking App'. The 'App Settings' tab is active. In the main area, there's a section for 'Navigation Items' with the sub-section 'Navigation Items'. It says: 'Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename the items that you add. Some navigation items are available only for phone or only for desktop. These items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.' Below this, there are two columns: 'Available Items' and 'Selected Items'. The 'Available Items' column contains a list of items with icons: Accounts, All Sites, Alternative Payment Methods, Analytics, App Launcher, Approval Requests, Approval Submission Details, Approval Submissions, Approval Work Items, and Asset Action Sources. The 'Selected Items' column contains items: Dashboards, Reports, Loans, Payments, and Recoveries. There are arrows between the two columns to move items between them.

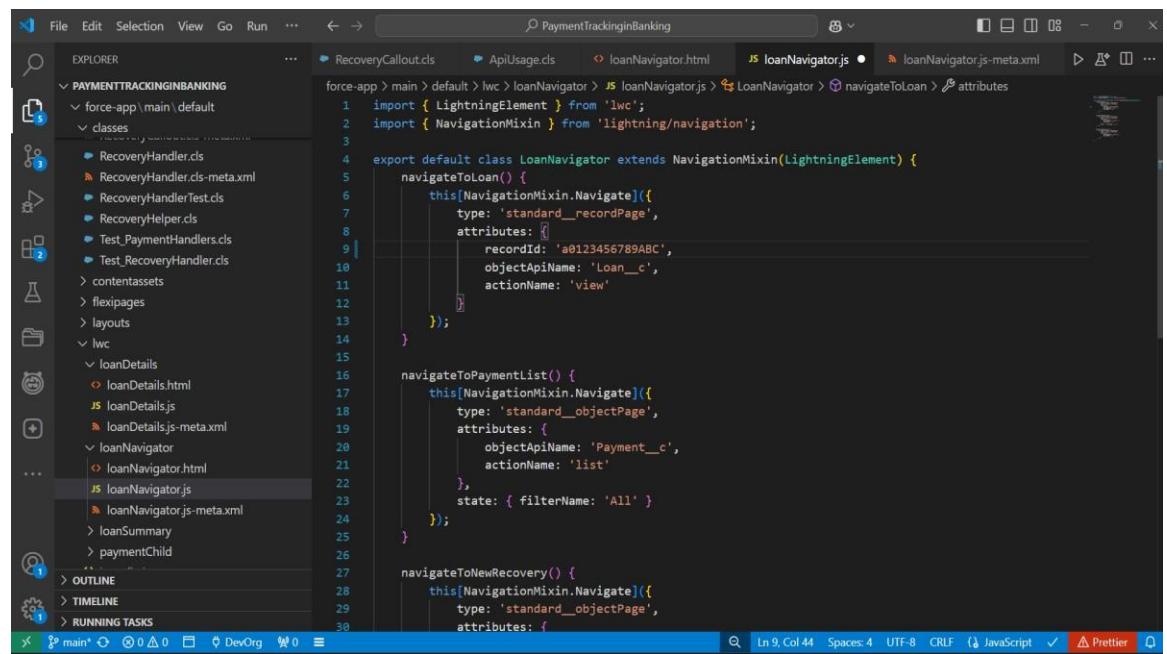
# LWC (Lightning Web Components)

- Built LWC components (loanSummary, loanDetails, paymentChild).



The screenshot shows the VS Code interface with the title bar "PaymentTrackinginBanking". The Explorer sidebar on the left shows a project structure under "PAYMENTTRACKINGINBANKING". In the center, the code editor displays the template for an LWC component named "loanDetails.html". The template uses the `<lightning-card>` element to display loan details, including the loan number, principal amount, and status. It includes conditional logic for when loan data is available or not.

```
<template>
  <lightning-card title="Loan Details" icon-name="standard:account">
    <template if:true={loan}>
      <p><strong>Loan Number:</strong> {loan.Name}</p>
      <p><strong>Principal Amount:</strong> {loan.Principal_Amount__c}</p>
      <p><strong>Status:</strong> {loan.Status__c}</p>
    </template>
    <template if:false={loan}>
      <p>No Loan Data Available</p>
    </template>
  </lightning-card>
</template>
```



The screenshot shows the VS Code interface with the title bar "PaymentTrackinginBanking". The Explorer sidebar on the left shows a project structure under "PAYMENTTRACKINGINBANKING". In the center, the code editor displays a JavaScript class named "loanNavigator.js". The class extends the "NavigationMixin" and defines three methods: "navigateToLoan", "navigateToPaymentList", and "navigateToNewRecovery". Each method uses the `this[NavigationMixin.Navigate]` method to perform navigation based on specific parameters like record ID, object API name, and action name.

```
import { LightningElement } from 'lwc';
import { NavigationMixin } from 'lightning/navigation';

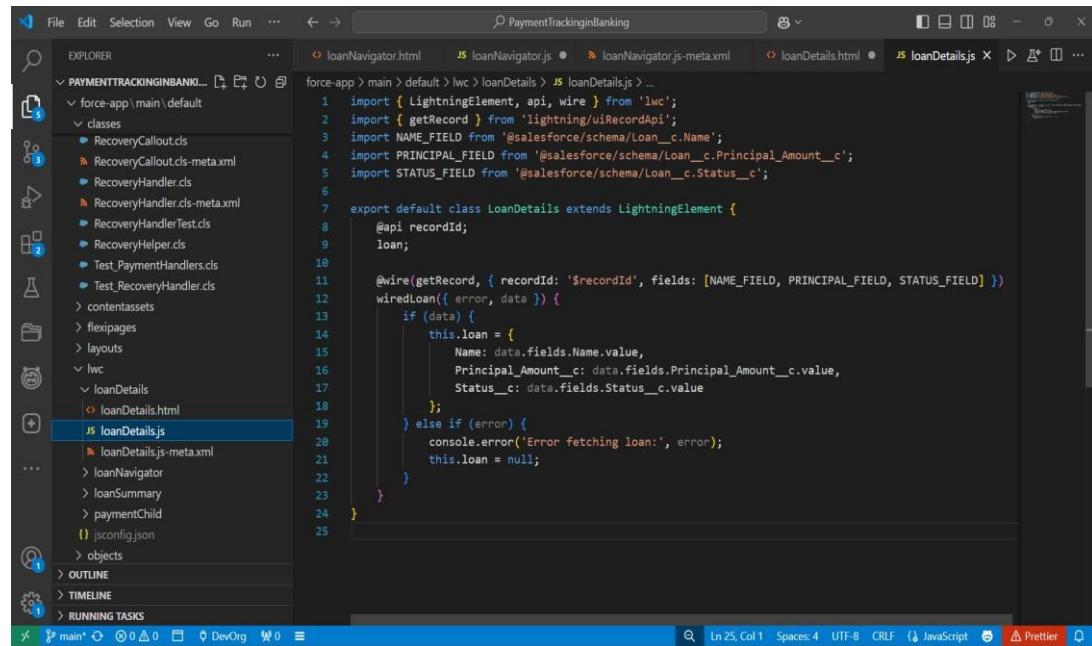
export default class LoanNavigator extends NavigationMixin(LightningElement) {
  navigateToLoan() {
    this[NavigationMixin.Navigate]({
      type: 'standard__recordPage',
      attributes: {
        recordId: 'a0123456789ABC',
        objectApiName: 'Loan__c',
        actionPerformed: 'view'
      }
    });
  }

  navigateToPaymentList() {
    this[NavigationMixin.Navigate]({
      type: 'standard__objectPage',
      attributes: {
        objectApiName: 'Payment__c',
        actionPerformed: 'list'
      },
      state: { filterName: 'All' }
    });
  }

  navigateToNewRecovery() {
    this[NavigationMixin.Navigate]({
      type: 'standard__objectPage',
      attributes: {
        objectApiName: 'Recovery__c',
        actionPerformed: 'new'
      }
    });
  }
}
```

## Wire Adapters

- Used @wire(getRecord) to fetch Loan fields directly in LWC.



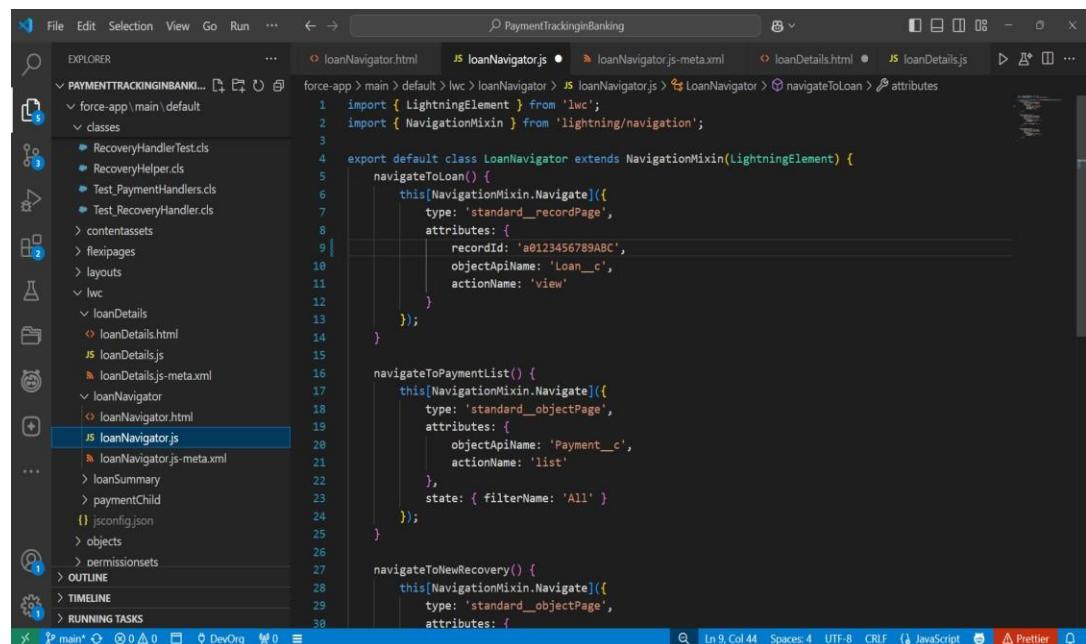
```
import { LightningElement, api, wire } from 'lwc';
import { getRecord } from 'lightning/uiRecordApi';
import NAME_FIELD from '@salesforce/schema/Loan__c.Name';
import PRINCIPAL_FIELD from '@salesforce/schema/Loan__c.Principal_Amount__c';
import STATUS_FIELD from '@salesforce/schema/Loan__c.Status__c';

export default class LoanDetails extends LightningElement {
    @api recordId;
    loan;

    @wire(getRecord, { recordId: '$recordId', fields: [NAME_FIELD, PRINCIPAL_FIELD, STATUS_FIELD] })
    wiredLoan({ error, data }) {
        if (data) {
            this.loan = {
                Name: data.fields.Name.value,
                Principal_Amount__c: data.fields.Principal_Amount__c.value,
                Status__c: data.fields.Status__c.value
            };
        } else if (error) {
            console.error('Error fetching loan:', error);
            this.loan = null;
        }
    }
}
```

## Navigation Service

- Implemented NavigationMixin to let users navigate from Payment → Loan detail page.
- Improves user experience with one-click record redirection.



```
import { LightningElement } from 'lwc';
import { NavigationMixin } from 'lightning/navigation';

export default class LoanNavigator extends NavigationMixin(LightningElement) {
    navigateToLoan() {
        this[NavigationMixin.Navigate]({
            type: 'standard__recordPage',
            attributes: {
                recordId: 'a0123456789ABC',
                objectApiName: 'Loan__c',
                actionName: 'view'
            }
        });
    }

    navigateToPaymentList() {
        this[NavigationMixin.Navigate]({
            type: 'standard__objectPage',
            attributes: {
                objectApiName: 'Payment__c',
                actionName: 'list'
            },
            state: { filterName: 'All' }
        });
    }

    navigateToNewRecovery() {
        this[NavigationMixin.Navigate]({
            type: 'standard__objectPage',
            attributes: {
                objectApiName: 'RecoveryCallout__c'
            }
        });
    }
}
```

## Phase 7: Integration & External Access

### Named Credentials

- Used to securely store API endpoints + authentication details for external systems.

The screenshot shows the 'Named Credentials' setup page in Salesforce. The 'BankAPI' credential is selected. The configuration includes:

- Label:** BankAPI
- Name:** BankAPI
- URL:** https://api.mybank.com
- Enabled for Callouts:** Checked
- Authentication:** External Credential, BankAPI\_Credential
- Callout Options:** Generate Authorization Header (checked), Allow Formulas in HTTP Header (unchecked), Allow Formulas in HTTP Body (unchecked)

### External Services

- Lets Salesforce call external APIs using schema (Swagger/OpenAPI).

The screenshot shows the 'External Services' setup page in Salesforce. The 'BankPaymentService' integration is selected. The configuration includes:

- Service name:** BankPaymentService
- Creation source:** From API specification
- Description:** External Banking API for Loan Validation and Payments
- Type:** OpenApi
- File Name:** BankAPI\_OpenAPI.json
- Named credentials:** BankAPI

**Operations:**

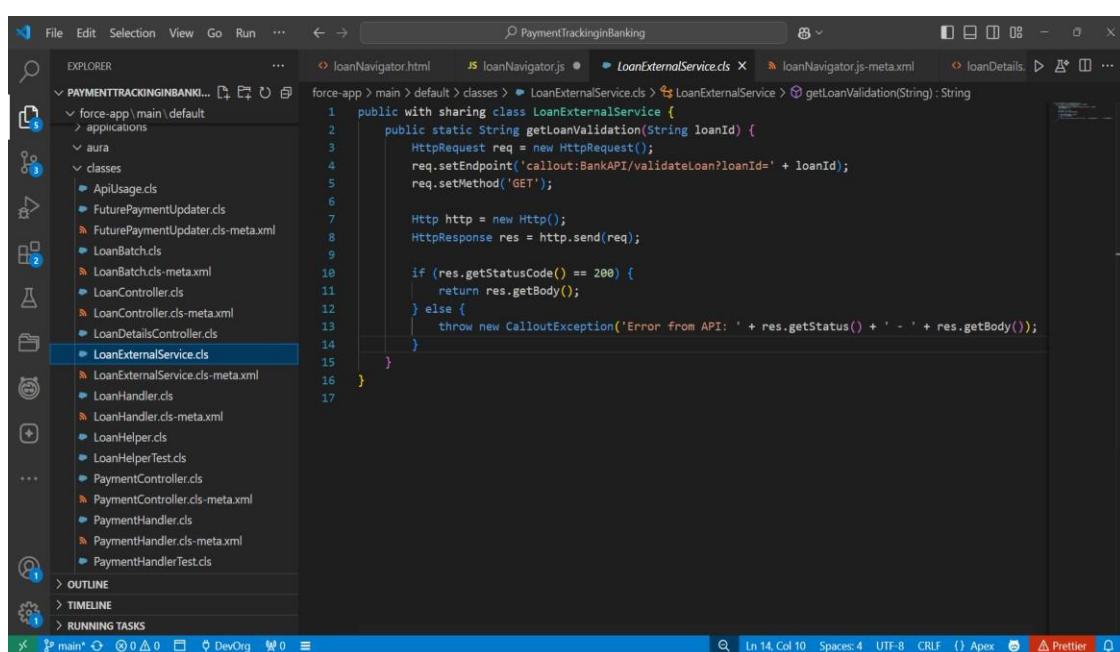
Operation Name ↑	Description	Input parameters	Output parameters
getValidateLoan	Validate Loan	loanId	responseCode, 200, default
postMakePayment	Make a Payment	loanId, amount	responseCode, 200, default

## Web Services (REST/SOAP)

- Allow external system to fetch Loan details.

### LoanExternalService.cls

```
public with sharing class LoanExternalService {  
  
    public static String getLoanValidation(String loanId) {  
  
        HttpRequest req = new HttpRequest();  
  
        req.setEndpoint('callout:BankAPI/validateLoan?loanId=' + loanId);  
  
        req.setMethod('GET');  
  
        Http http = new Http();  
  
        HttpResponse res = http.send(req);  
  
        if (res.getStatusCode() == 200) {  
  
            return res.getBody();} else {  
  
            throw new CalloutException('Error from API: ' + res.getStatus() + ' - ' + res.getBody());}}}
```

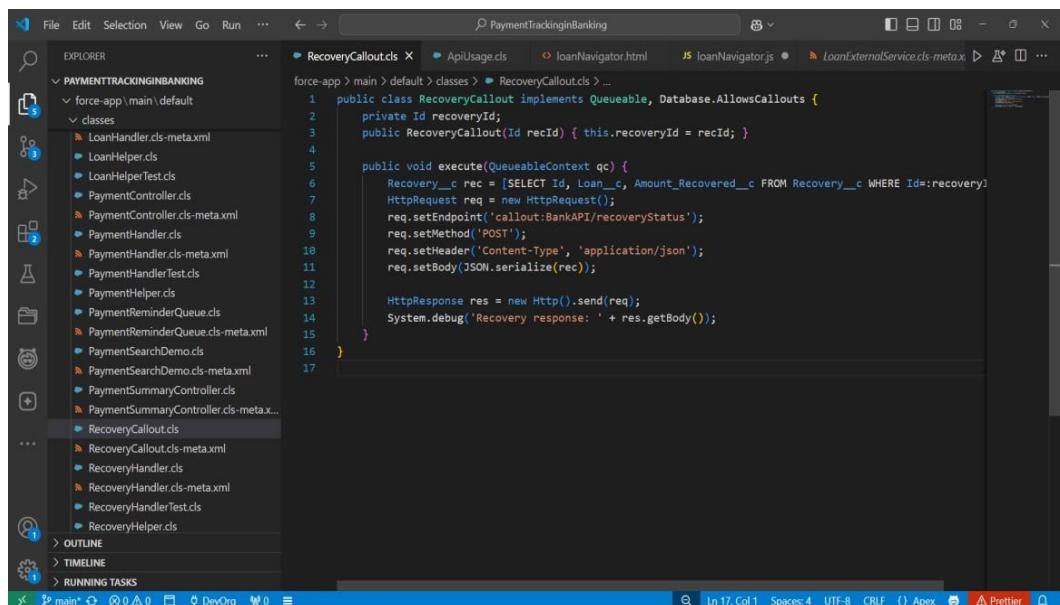


## Callouts

- Used when Salesforce sends data to external services.

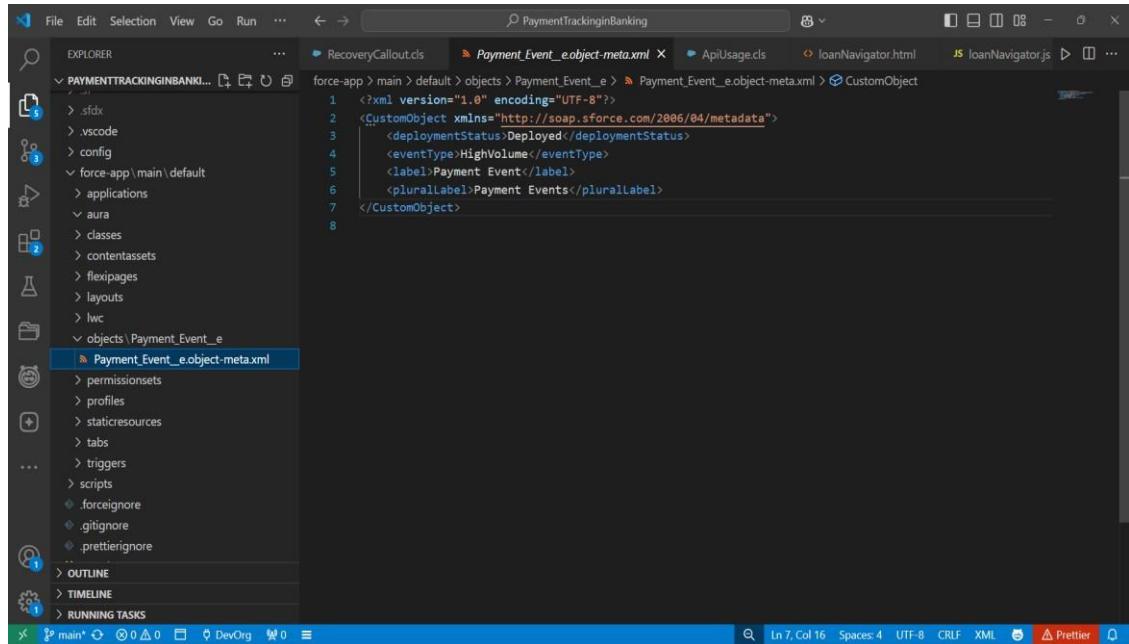
### RecoveryCallout.cls

```
public class RecoveryCallout implements Queueable,  
    Database.AllowsCallouts {  
  
    private Id recoveryId;  
  
    public RecoveryCallout(Id recId) { this.recoveryId = recId; }  
  
    public void execute(QueueableContext qc) {  
  
        Recovery__c rec = [SELECT Id, Loan__c, Amount_Recovered__c  
        FROM Recovery__c WHERE Id=:recoveryId];  
  
        HttpRequest req = new HttpRequest();  
  
        req.setEndpoint('callout:BankAPI/recoveryStatus');  
        req.setMethod('POST');  
  
        req.setHeader('Content-Type', 'application/json');  
  
        req.setBody(JSON.serialize(rec));  
  
        HttpResponse res = new Http().send(req);  
  
        System.debug('Recovery response: ' + res.getBody());}  
}
```



## Platform Events

- Publish event when a Payment is Overdue (integrates with external collectors).



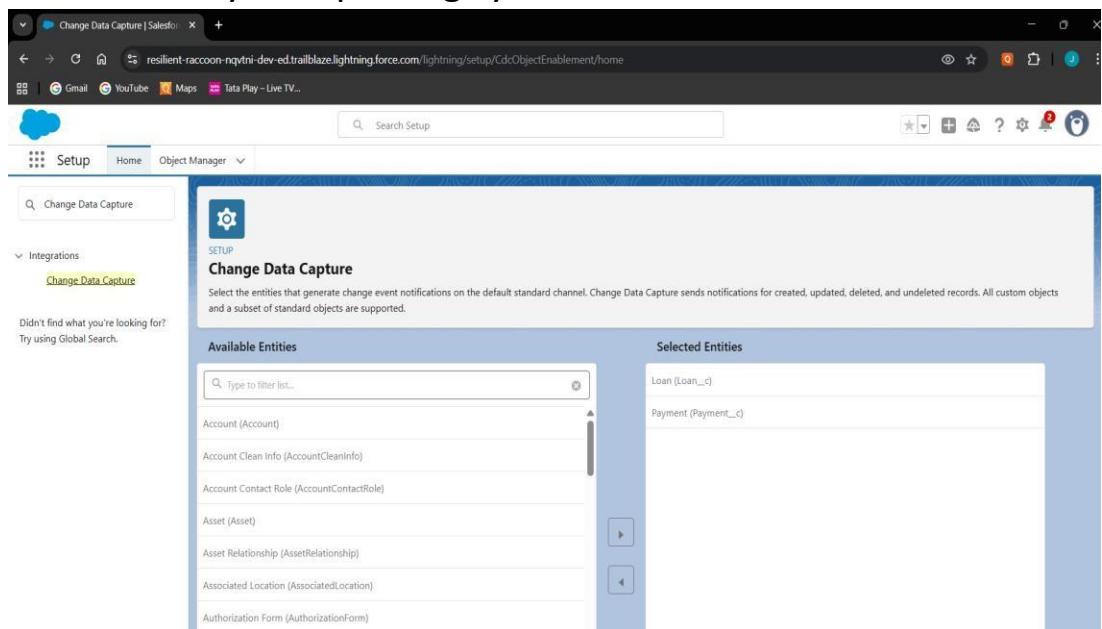
The screenshot shows the VS Code interface with the following details:

- File Explorer:** On the left, the "PAYMENTTRACKINGINBANKING" folder is expanded, showing subfolders like ".sfdx", ".vscode", "config", "force-app\main\default", "aura", "classes", "contentassets", "flexipages", "layouts", "lwc", and "objects\Payment\_Event\_e".
- Code Editor:** The main editor area displays the XML file "Payment\_Event\_e\_object-meta.xml".
- Status Bar:** At the bottom, it shows "In 7, Col 16 Spaces 4 UTF-8 CRLF XML" and a "Prettier" icon.

```
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <deploymentStatus>Deployed</deploymentStatus>
    <eventType>HighVolume</eventType>
    <label>Payment Event</label>
    <pluralLabel>Payment Events</pluralLabel>
</CustomObject>
```

## Change Data Capture (CDC)

- Streams real-time changes of Salesforce records to external systems.
- Any update in Payment\_c record can be pushed automatically to reporting system.

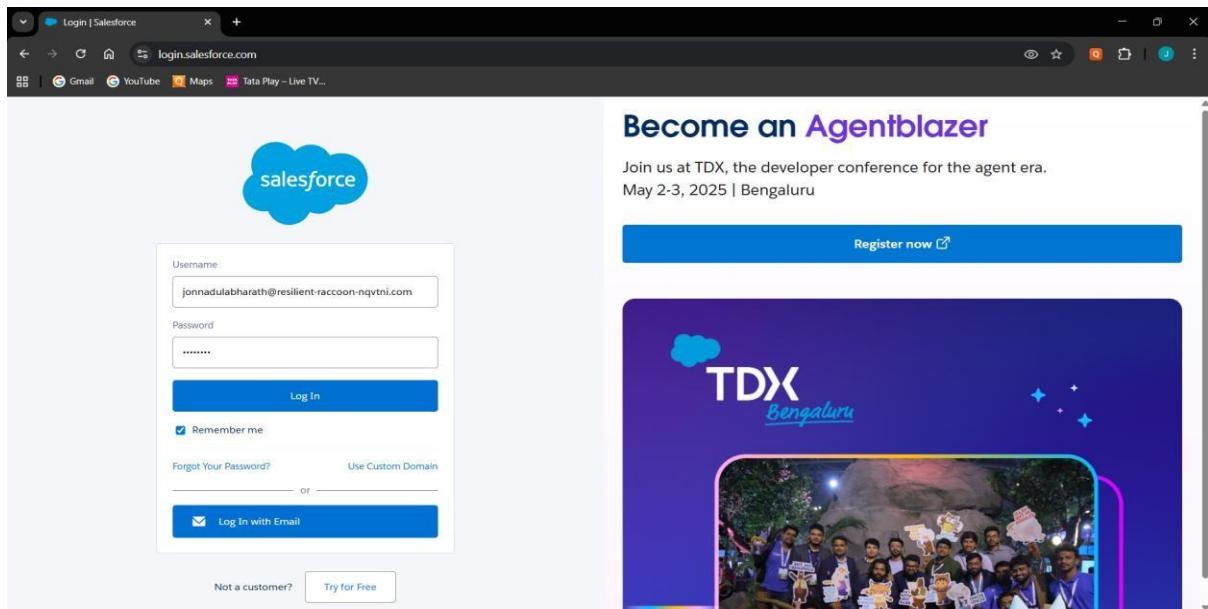


The screenshot shows the Salesforce Setup page for Change Data Capture:

- Page Header:** "Change Data Capture | Salesforce" and the URL "resilient-raccoon-nqvtni-dev-ed.lightning.force.com/lightning/setup/CdcObjectEnablement/home".
- Left Navigation:** "Setup" tab selected, "Integrations" section, "Change Data Capture" link.
- Content Area:**
  - Section Title:** "Change Data Capture".
  - Description:** "Select the entities that generate change event notifications on the default standard channel. Change Data Capture sends notifications for created, updated, deleted, and undeleted records. All custom objects and a subset of standard objects are supported."
  - Available Entities:** A list of entities including Account, Account Clean Info, Account Contact Role, Asset, Asset Relationship, Associated Location, and Authorization Form.
  - Selected Entities:** A list showing "Loan (Loan\_\_c)" and "Payment (Payment\_\_c)".

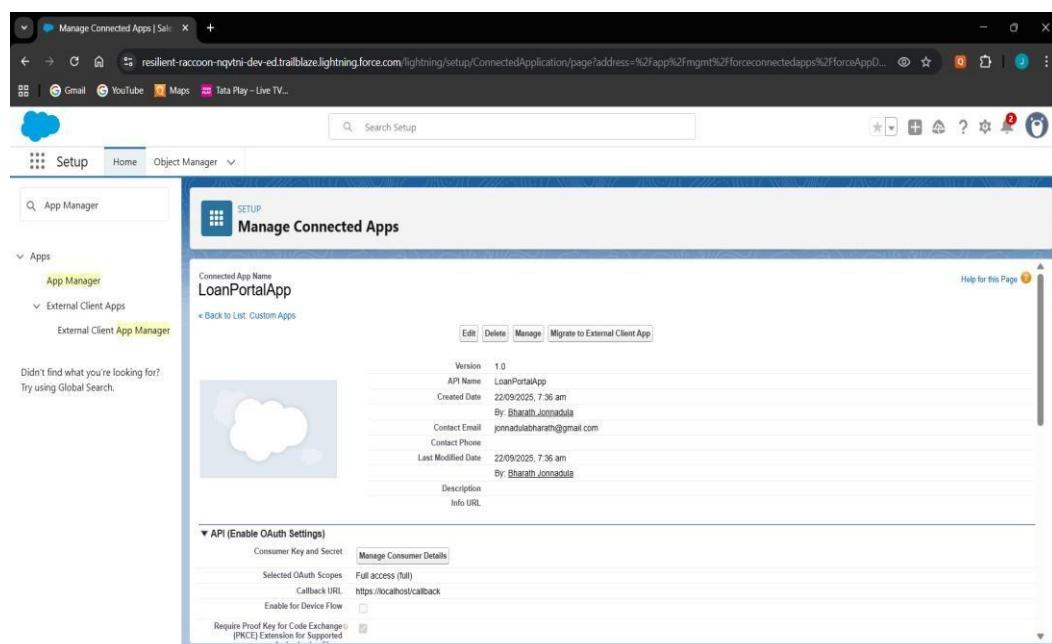
## Salesforce Connect

- Provides real-time view of external data without storing in Salesforce.



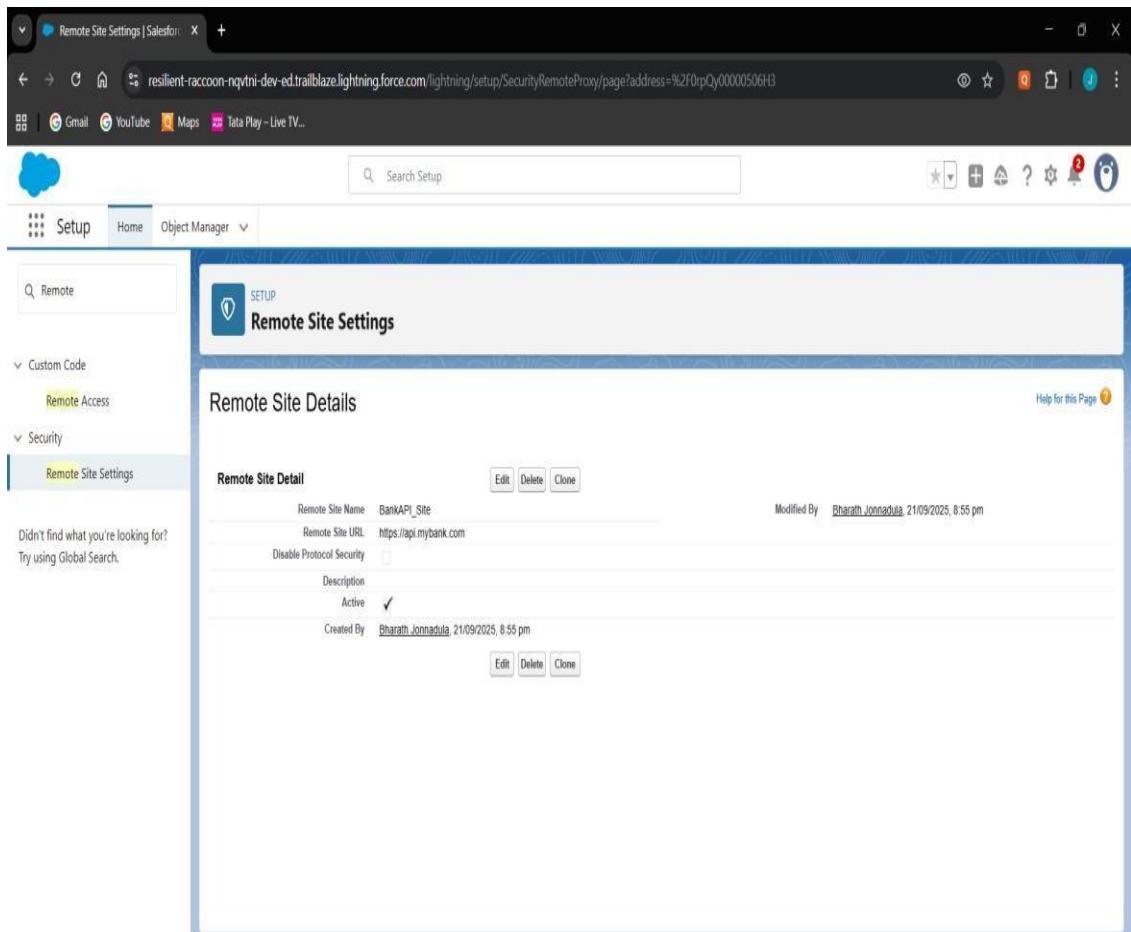
## OAuth & Authentication

- Enables secure access for external apps to Salesforce APIs.



## Remote Site Settings

- Required to allow outbound callouts to untrusted domains.
- Add `https://api.bank.com` as a Remote Site Setting before making callouts.



## Phase 8: Data Management & Deployment

### Data Import Wizard

- Used Data Import Wizard to insert sample data into Loan, Payment, and Recovery objects.

Basic Information

Name	Loan001
Customer	Ravindra Babu
Owner	Bharath Jonnadula
Customer Email	ravindrababu.jonnadula@gmail.com
Customer Code	CUST001

Payments

Name	LN-25-0001
Payment Date	24/09/2025
Amount	₹10,000.00
Payment Method	Bank Transfer
Status	Pending
Loan	Loan001
Customer Email	ravindrababu.jonnadula@gmail.com

Last Modified By  
Bharath Jonnadula, 22/09/2025, 12:19 pm

No related lists to display

Sample Flow Report: Screen Flows

The screenshot shows a Salesforce Lightning page for a Recovery record. The record details are as follows:

- Recovery Number: LN-25-0001
- Recovery Date: 05/09/2025
- Amount Recovered: ₹90,000.00
- Assigned Agent: Recovery Agent
- Status: InProgress
- Loan: Loan002
- Created By: Bharath Jonnadula, 22/09/2025, 12:24 pm
- Last Modified By: Bharath Jonnadula, 22/09/2025, 12:24 pm

The page also displays a message: "No related lists to display". Below the main content, there is a section titled "Sample Flow Report: Screen Flows".

## Data Loader

- Data Loader for bulk upload, but not required since records were limited.

## Data Export & Backup

- Verified Data Export option in Setup for backup.

The screenshot shows the "Data Export" setup page under the "Monthly Export Service". The configuration includes:

- Export File Encoding: ISO-8859-1 (General US & Western European, ISO-LATIN-1)
- Include images, documents, and attachments: Unchecked
- Include Salesforce Files and Salesforce CRM Content document versions: Unchecked
- Replace carriage returns with spaces: Checked

The "Exported Data" section allows selecting data types to include in the export. The "Include all data" checkbox is checked, and the following data types are selected:

- Contract
- Asset
- Lead
- BusinessProcess
- Campaign
- CaseContactRole
- ContentDocumentLink
- ContractContactRole
- Order
- Account
- Partner
- NotificationMember
- CampaignMember
- CaseHistory2
- ContentVersion
- EmailDisclaimer
- OrderItem
- Contact
- Product2
- UserRole
- Case
- CaseSolution
- ContentVersionMap
- EmailMessage

## VS Code & SFDX

- Used VS Code with SFDX CLI for writing and deploying Apex Classes, Triggers, and LWC.
- Successfully pushed/pulled metadata between local project and Salesforce Org



The screenshot shows the VS Code interface with the Terminal tab selected. The terminal window displays the following command-line session:

```
Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking>
* History restored

Microsoft Windows [Version 10.0.26100.6584]
(c) Microsoft Corporation. All rights reserved.

D:\PaymentTrackinginBanking>sf project deploy start --source-dir force-app/main/default/classes --target-org DevOrg
```

The terminal also shows a dropdown menu for selecting between 'cmd' and 'powershell' as the terminal type.

## Phase 9: Reporting, Dashboards & Security Review

### Reports (Tabular, Summary, Matrix, Joined)

- Created different report formats (Grouped using Amount, Payment Date, Status and added columns Loan:Name, Payment:Name, Status, Customer Email)

The screenshot shows the Salesforce Report Builder interface. The report is titled "Loans with Payments". The left sidebar displays "Fields" for grouping by "Amount", "Payment Date", and "Status". The main area shows a table with three distinct sections: one for "Overdue" loans (2 records), one for "Pending" loans (2 records), and one for "Subtotal" (1 record). The table includes columns for "Amount", "Payment Date", "Status", "Loan: Name", "Payment: Name", "Status", and "Customer Email". A preview message at the top says, "Previewing a limited number of records. Run the report to see everything." At the bottom, there are buttons for "Row Counts", "Detail Rows", "Subtotals", and "Grand Total".

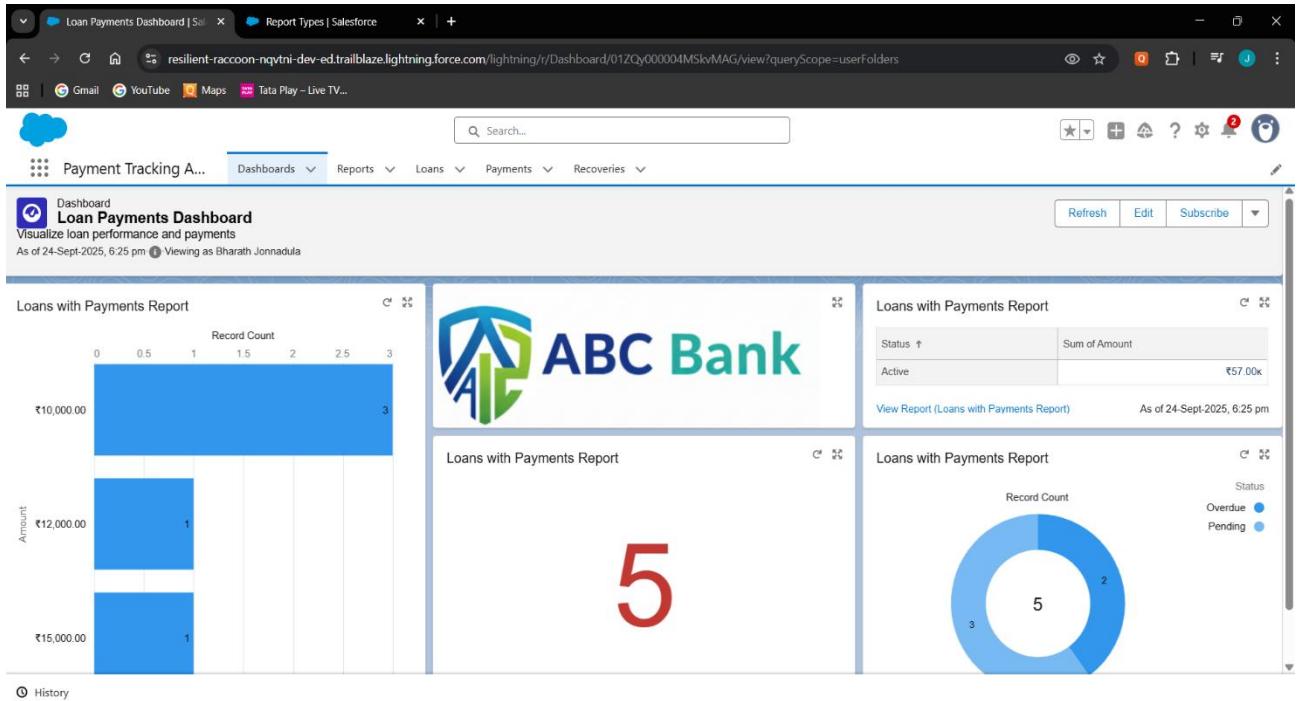
### Report Types

- Defined a Custom Report Type with `Loan__c` as the primary object and `Payment__c`.

The screenshot shows the Salesforce Setup interface under the "Custom Report Types" section. The left sidebar has "Feature Settings" expanded, with "Report Types" selected. The main area is titled "Loans with Payments" and shows a table of fields. The table has two sections: "Loans" and "Payments". The "Loans" section includes fields like "Loan ID", "Owner", "Name", "Record Type", "Created Date", "Created By", "Last Modified Date", "Last Modified By", and "Last Activity Date". The "Payments" section includes fields like "Payment ID", "Name", "Created Date", "Created By", "Last Modified Date", "Last Activity Date", and "Payment Date". There are buttons for "Sections", "New Section", "Lookup Fields", "Close", and "Save".

## Dashboards

- Built a Payment Tracking Dashboard that included widgets like Loan vs Paid Amount.
- Connected each widget to source reports.



## Profiles

- Created Manager, Loan Officer, Recovery agent profiles.

The screenshot shows the Salesforce Setup page for creating a new profile:

- The left sidebar shows "Profiles" under the "Users" category.
- The main area is titled "Profile Edit" for "Manager Profile".
- Fields include "Name" (Manager Profile), "User License" (Salesforce), and "Description".
- A "Custom Profile" checkbox is checked.
- "Custom App Settings" section shows checkboxes for various apps:
  - All Tabs (standard\_\_AllTabSet)
  - Analytics Studio (standard\_\_Insights)
  - App Launcher (standard\_\_AppLauncher)
  - Approvals (standard\_\_Approvals)
  - Automation (standard\_\_FlowsApp)
  - Payment Tracking App (Payment\_Tracking\_App)
  - Playground Starter (trifidips\_Playground\_Starter)
  - Sales (standard\_\_Lightning\_Sales)
  - Sales (standard\_\_Sales)
  - Sales Cloud Mobile (standard\_\_SalesCloudMobile)

Loan Payments Dashboard | Sales | Profiles | Salesforce

resilient-raccoon-nqvtni-dev-ed.trailblaze.lightning.force.com/lightning/setup/EnhancedProfiles/page?address=%2F00eQy00000H54YL%2Fe%3FretURL%3D%252F00eQy...

Setup Home Object Manager

Profiles

Users Profiles

Didnt find what you're looking for?  
Try using Global Search.

SETUP Profiles

Profile Edit  
Loan Officer Profile

Set the permissions and page layouts for this profile.

Profile Edit

Name: Loan Officer Profile

User License: Salesforce

Description:

Custom Profile:

Custom App Settings

	Visible	Default	Visible	Default	
All Tabs (standard__AllTabSet)	<input checked="" type="checkbox"/>	<input type="radio"/>	Payment Tracking App (Payment_Tracking_App)	<input checked="" type="checkbox"/>	<input type="radio"/>
Analytics Studio (standard__Insights)	<input checked="" type="checkbox"/>	<input type="radio"/>	Playground Starter (trhdtips__Playground_Starter)	<input checked="" type="checkbox"/>	<input type="radio"/>
App Launcher (standard__AppLauncher)	<input checked="" type="checkbox"/>	<input type="radio"/>	Sales (standard__LightningSales)	<input checked="" type="checkbox"/>	<input type="radio"/>
Approvals (standard__Approvals)	<input checked="" type="checkbox"/>	<input type="radio"/>	Sales (standard__Sales)	<input type="checkbox"/>	<input checked="" type="radio"/>
Automation (standard__FlowsApp)	<input checked="" type="checkbox"/>	<input type="radio"/>	Sales Cloud Mobile (standard__SalesCloudMobile)	<input checked="" type="checkbox"/>	<input type="radio"/>
Bolt Solutions (standard__LightningBolt)	<input checked="" type="checkbox"/>	<input type="radio"/>	Sales Console	<input checked="" type="checkbox"/>	<input type="radio"/>

Help for this Page

Loan Payments Dashboard | Sales | Profiles | Salesforce

resilient-raccoon-nqvtni-dev-ed.trailblaze.lightning.force.com/lightning/setup/EnhancedProfiles/page?address=%2F00eQy00000H54YL%2Fe%3FretURL%3D%252F00eQy...

Setup Home Object Manager

Profiles

Users Profiles

Didnt find what you're looking for?  
Try using Global Search.

SETUP Profiles

Profile Edit  
Recovery Agent Profile

Set the permissions and page layouts for this profile.

Profile Edit

Name: Recovery Agent Profile

User License: Salesforce

Description:

Custom Profile:

Custom App Settings

	Visible	Default	Visible	Default	
All Tabs (standard__AllTabSet)	<input checked="" type="checkbox"/>	<input type="radio"/>	Payment Tracking App (Payment_Tracking_App)	<input checked="" type="checkbox"/>	<input type="radio"/>
Analytics Studio (standard__Insights)	<input checked="" type="checkbox"/>	<input type="radio"/>	Playground Starter (trhdtips__Playground_Starter)	<input checked="" type="checkbox"/>	<input type="radio"/>
App Launcher (standard__AppLauncher)	<input checked="" type="checkbox"/>	<input type="radio"/>	Sales (standard__LightningSales)	<input checked="" type="checkbox"/>	<input type="radio"/>
Approvals (standard__Approvals)	<input checked="" type="checkbox"/>	<input type="radio"/>	Sales (standard__Sales)	<input type="checkbox"/>	<input checked="" type="radio"/>
Automation (standard__FlowsApp)	<input checked="" type="checkbox"/>	<input type="radio"/>	Sales Cloud Mobile (standard__SalesCloudMobile)	<input checked="" type="checkbox"/>	<input type="radio"/>
Bolt Solutions (standard__LightningBolt)	<input checked="" type="checkbox"/>	<input type="radio"/>	Sales Console	<input checked="" type="checkbox"/>	<input type="radio"/>

Help for this Page

## Roles

- Built the role hierarchy of Branch Manager, Loan Officer and Recovery Agent.

The screenshot shows the Salesforce Setup Roles page. The left sidebar has a 'Users' section with 'Roles' selected. The main content area is titled 'Creating the Role Hierarchy' and displays a tree view of roles under 'ABC Bank Payment Tracking'. The hierarchy includes:

- Branch Manager (Edit | Del | Assign)
- Loan Officer (Edit | Del | Assign)
- Recovery Agent (Edit | Del | Assign)
- User (Edit | Del | Assign)
- VP International Sales (Edit | Del | Assign)
- VP Marketing (Edit | Del | Assign)
- Customer Support, International (Edit | Del | Assign)
- Customer Support, North America (Edit | Del | Assign)
- Installation & Repair Services (Edit | Del | Assign)

A 'Help for this Page' link is visible in the top right corner.

## Users

- Created users (Branch Manager, Loan Officer, Recovery Agent).

The screenshot shows the Salesforce Setup Users page. The left sidebar has a 'Users' section with 'Users' selected. The main content area is titled 'All Users' and displays a list of users:

Action	Full Name	Alias	Username	Role	Active	Profile
<input type="checkbox"/>   Edit   Login	Agent_Recovery	recovery	recovery_officer@bankdemo.com		<input checked="" type="checkbox"/>	Standard Platform User
<input type="checkbox"/>   Edit	Chatter Expert	Chatter	chatty.0000y0000y0c5map.lcpdyxgqg@chatter.salesforce.com		<input checked="" type="checkbox"/>	Chatter Free User
<input type="checkbox"/>   Edit	Jonnadula Bharath	BJon	jonnadulabharath@resilient-raccoon-nqvtni.com		<input checked="" type="checkbox"/>	System Administrator
<input type="checkbox"/>   Edit   Login	Manager_Branch	manager	bankmanager_officer@bankdemo.com		<input checked="" type="checkbox"/>	Manager Profile
<input type="checkbox"/>   Edit   Login	Officer_Loan	loan	loan.officer@bankdemo.com		<input checked="" type="checkbox"/>	Standard Platform User

A navigation bar at the bottom includes links for A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, and Other.

## Permission Sets

- Create a Permission Set for payment tracking access.

The screenshot shows the Salesforce Setup interface under the 'Permission Sets' section. A permission set named 'Payment Team Access' is displayed. The 'Permission Set Overview' section includes fields for Description (left blank), License (left blank), Session Activation Required (unchecked), and Permission Set Groups Added To (0). The API Name is 'Payment\_Team\_Access' and the Namespace Prefix is 'Bharath\_jonnadula'. The 'Created By' and 'Last Modified By' fields show 'Bharath\_jonnadula' with their respective dates and times. The 'Apps' section lists several settings: 'Assigned Apps' (Settings that specify which apps are visible in the app menu), 'Assigned Connected Apps' (Settings that specify which connected apps are visible in the app menu), 'Object Settings' (Permissions to access objects and fields, and settings such as tab availability), 'App Permissions' (Permissions to perform app-specific actions, such as "Manage Call Centers"), and 'Apex Class Access' (Permissions to execute Apex classes).

## OWD & Sharing Rules

- Set custom objects (Loan, Payment, Recovery) to Private.

The screenshot shows the Salesforce Setup interface under the 'Sharing Settings' section. The 'Sharing Settings' page displays a table of sharing rules for various objects. The columns are 'Object' (listing objects like Quick Link Usage, Rebate Payout Snapshot, Scorecard, Seller, Service Contract, Streaming Channel, Tableau Host Mapping, Thanks, User Provisioning Request, Web Cart Document, Work Order, Work Plan, Work Plan Template, Work Step Template, Loan, Loan Agent Assignment, Payment, and Recovery), 'Default' (listing sharing levels like Private, Public Read/Write, Public Read Only, or Controlled by Parent), and 'Sharing Rule' (indicated by a checkmark icon). Most objects have 'Private' as the default sharing level, except for some like Rebate Payout Snapshot, Scorecard, and others which are 'Controlled by Parent'.

Object	Default	Sharing Rule
Quick Link Usage	Private	
Rebate Payout Snapshot	Private	
Scorecard	Private	
Seller	Private	
Service Contract	Private	
Streaming Channel	Public Read/Write	
Tableau Host Mapping	Public Read Only	
Thanks	Public Read Only	
User Provisioning Request	Private	
Web Cart Document	Private	
Work Order	Private	
Work Plan	Private	
Work Plan Template	Private	
Work Step Template	Private	
Loan	Private	
Loan Agent Assignment	Controlled by Parent	
Payment	Controlled by Parent	
Recovery	Private	

## Sharing Settings

- Set custom objects to private(Loan, Payment, Recovery).

The screenshot shows the Salesforce Sharing Settings page for the 'Loan' object. The 'Manage sharing settings for' dropdown is set to 'Loan'. In the 'Default Sharing Settings' section, under 'Organization-Wide Defaults', the 'Object' is 'Loan', 'Default Internal Access' is 'Private', and 'Default External Access' is 'Private'. The 'Grant Access Using Hierarchies' checkbox is checked. In the 'Other Settings' section, 'Manager Groups' is unchecked, 'Secure guest user record access' is checked, and 'Require permission to view record names in lookup fields' is unchecked.

The screenshot shows the Salesforce Sharing Settings page for the 'Payment' object. The 'Manage sharing settings for' dropdown is set to 'Payment'. In the 'Default Sharing Settings' section, under 'Organization-Wide Defaults', the 'Object' is 'Payment', 'Default Internal Access' is 'Controlled by Parent', and 'Default External Access' is 'Controlled by Parent'. The 'Grant Access Using Hierarchies' checkbox is unchecked. In the 'Other Settings' section, 'Manager Groups' is unchecked, 'Secure guest user record access' is checked, and 'Require permission to view record names in lookup fields' is unchecked.

**Sharing Settings**

This page displays your organization's sharing settings. These settings specify the level of access your users have to each others' data. Go to [Background Jobs](#) to monitor the progress of a change to an organization-wide default or a parallel sharing recalculation.

Manage sharing settings for: **Recovery**

Disable External Sharing Model

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Recovery	Private	Private	<input checked="" type="checkbox"/>

**Other Settings**

Manager Groups	<input type="checkbox"/>
Secure guest user record access	<input checked="" type="checkbox"/>
Require permission to view record names in lookup fields	<input type="checkbox"/>

[Organization-Wide Defaults Help](#)

[Other Settings Help](#)

**Sharing Rules**

## Field Level Security

- Restricted interest rate to view only for manager.

**Users**

Force.com - Free User  
Gold Partner User  
Identity User  
Loan Officer Profile  
Manager Profile  
Marketing User  
Minimum Access - API Only Integrations  
Minimum Access - Salesforce  
Partner App Subscription User  
Partner Community Login User  
Partner Community User  
Read Only  
Recovery Agent Profile  
Salesforce API Only System Integrations  
Silver Partner User  
Solution Manager  
Standard Platform User

## Session settings

- Set session timeout for 30 minutes.

The screenshot shows the Salesforce Setup interface. In the left sidebar, under 'Session Management', 'Session Settings' is selected. The main content area is titled 'Session Settings' and contains the following sections:

- Session Timeout**: A dropdown menu set to '30 minutes'. There are two checkboxes: 'Disable session timeout warning popup' (unchecked) and 'Force logout on session timeout' (checked).
- Session Settings**: A list of configuration options:
  - Lock sessions to the IP address from which they originated
  - Lock sessions to the domain in which they were first used
  - Terminate all of a user's sessions when an admin resets that user's password
  - Force reload after Login-As-User
  - Require HttpOnly attribute
  - Use POST requests for cross-domain sessions
  - Enforce login IP ranges on every request
  - When embedding a Lightning application in a third-party site, use a session token instead of a session cookie.
- Extended use of IE11 with Lightning Experience**: A note stating "\*\*EXTENDED USE OF IE11 WITH LIGHTNING EXPERIENCE HAS NOW ENDED\*\*" and "AS OF DECEMBER 31, THE EXTENDED PERIOD HAS ENDED, AND USE OF INTERNET EXPLORER 11 (IE 11) WITH LIGHTNING EXPERIENCE IS NO LONGER SUPPORTED. ISSUES WITH PERFORMANCE OR FUNCTIONALITY".

## Login IP Ranges

- Configured login IP ranges for each profile.

The screenshot shows the Salesforce Setup interface. In the left sidebar, under 'Users', 'Profiles' is selected. The main content area is titled 'Manager Profile' and contains the following sections:

- Profile**: A note stating "Users with this profile have the permissions and page layouts listed below. Administrators can change a user's profile by editing that user's personal information. If your organization uses Record Types, use the Edit links in the Record Type Settings section below to make one or more record types available to users with this profile."
- Login IP Ranges**: A table with one row:

Action	IP Start Address	IP End Address	Description
Edit   Del	49.37.153.221	49.37.153.221	
- Page Layouts**: A table showing page layout assignments for various object types:

Standard Object Layouts	Global	Individual
Email Application	Not Assigned	Invoice Layout [View Assignment]
Home Page Layout	DE Default [View Assignment]	Invoice Line Layout [View Assignment]
Access	Access Layout [View Assignment]	Lead Layout [View Assignment]

Loan Payments Dashboard | Sales | Profiles | Salesforce

resilient-raccoon-nqvtni-dev-ed.trailblaze.lightning.force.com/lightning/setup/EnhancedProfiles/page?address=%2F00eQy00000H54YL

Setup Home Object Manager

Search Setup

Profile

Users Profiles

Loan Officer Profile

Profile Description: Loan Officer Profile

Users with this profile have the permissions and page layouts listed below. Administrators can change a user's profile by editing that user's personal information.

If your organization uses Record Types, use the Edit links in the Record Type Settings section below to make one or more record types available to users with this profile.

Login IP Ranges [1] Enabled Apex Class Access [1] Enabled Visualforce Page Access [0] Enabled External Data Source Access [0] Enabled Named Credential Access [0] Enabled External Credential Principal Access [0]

**-Login IP Ranges**

Action	IP Start Address	IP End Address	Description
Edit   Del	49.37.153.221	49.37.153.221	

Description:  Created By: Bharath.Jonnadula, 15/09/2025, 7:34 pm Modified By: Bharath.Jonnadula, 22/09/2025, 11:07 am

**Page Layouts**

Standard Object Layouts	Global	Individual
Email Application	Not Assigned [View Assignment]	Invoice [View Assignment]
Home Page Layout	DE Default [View Assignment]	Invoice Line [View Assignment]
Access	Access Layout [View Assignment]	Lead [View Assignment]

Help for this Page ?

Log In Ranges Help ?

Loan Payments Dashboard | Sales | Profiles | Salesforce

resilient-raccoon-nqvtni-dev-ed.trailblaze.lightning.force.com/lightning/setup/EnhancedProfiles/page?address=%2F00eQy00000H50eP

Setup Home Object Manager

Search Setup

Profile

Users Profiles

Recovery Agent Profile

Profile Description: Recovery Agent Profile

Users with this profile have the permissions and page layouts listed below. Administrators can change a user's profile by editing that user's personal information.

If your organization uses Record Types, use the Edit links in the Record Type Settings section below to make one or more record types available to users with this profile.

Login IP Ranges [1] Enabled Apex Class Access [1] Enabled Visualforce Page Access [0] Enabled External Data Source Access [0] Enabled Named Credential Access [0] Enabled External Credential Principal Access [0]

**-Login IP Ranges**

Action	IP Start Address	IP End Address	Description
Edit   Del	49.37.153.221	49.37.153.221	

Description:  Created By: Bharath.Jonnadula, 15/09/2025, 7:35 pm Modified By: Bharath.Jonnadula, 22/09/2025, 11:07 am

**Page Layouts**

Standard Object Layouts	Global	Individual
Email Application	Not Assigned [View Assignment]	Invoice [View Assignment]
Home Page Layout	DE Default [View Assignment]	Invoice Line [View Assignment]
Access	Access Layout [View Assignment]	Lead [View Assignment]

Help for this Page ?

Log In Ranges Help ?

## Audit Trail

- Enabled Setup Audit Trail to track all admin-level changes.

The screenshot shows the 'View Setup Audit Trail' page in the Salesforce Setup interface. The page title is 'View Setup Audit Trail'. It displays a table of audit entries with columns for Date, User, Source Namespace Prefix, Action, Section, and Delegate User. The table lists 20 entries from September 2025, primarily related to user management actions like deleting profiles, logging in as users, and changing password status.

Date	User	Source Namespace Prefix	Action	Section	Delegate User
25/09/2025, 2:13:30 pm IST	jonnadulabharath@resilient-raccoon-nqvtni.com		Deleted profile Manager	Manage Users	jonnadulabharath@resilient-raccoon-nqvtni.com
24/09/2025, 8:02:42 pm IST	y22cs071@rvnj.ac.in		Logged in using Login-As access for Manager Test	Manage Users	jonnadulabharath@resilient-raccoon-nqvtni.com
24/09/2025, 7:46:54 pm IST	y22cs071@rvnj.ac.in		Set new password for user Manager Test	Manage Users	
24/09/2025, 7:46:32 pm IST	Automated Process		For user y22cs071@rvnj.ac.in, the User Verified Email status changed to verified	Manage Users	
24/09/2025, 7:37:50 pm IST	Automated Process		For user y22cs071@rvnj.ac.in, the User Verified Email status changed to verified	Manage Users	
24/09/2025, 7:37:12 pm IST	y22cs071@rvnj.ac.in		Reset password for user Manager Test	Manage Users	jonnadulabharath@resilient-raccoon-nqvtni.com
24/09/2025, 7:36:49 pm IST	y22cs071@rvnj.ac.in		Logged in using Login-As access for Manager Test	Manage Users	jonnadulabharath@resilient-raccoon-nqvtni.com
24/09/2025, 7:36:33 pm IST	jonnadulabharath@resilient-raccoon-nqvtni.com		Changed Administrators Can Log in as Any User from off to on	Manage Users	
24/09/2025, 7:35:20 pm IST	jonnadulabharath@resilient-raccoon-nqvtni.com		Created new user Manager Test	Manage Users	
24/09/2025, 7:30:05 pm IST	jonnadulabharath@resilient-raccoon-nqvtni.com		Created new user Manager Test	Manage Users	

## Phase 10: Quality Assurance Testing

### Test Case 1:

- **Use case:** Verify Loan doesn't allow negative amounts.
- **Input:** Creating Loan with principal amount=-1000.

A screenshot of a software application window titled "Payment Tracking A...". The main interface shows a list of loans under the heading "Recently Viewed". On the right, a modal dialog box is open for "Basic Information". In the "Financials" section, the "Principal Amount" field contains the value "-1000". Below the dialog, a toolbar includes "Import", "Change Owner", and "Assign Label" buttons.

- **Expected Result:** Error should appear.
- **Output:**

A screenshot of the same software application window as the previous image. The modal dialog now displays an error message: "Principal Amount must be greater than zero." The "Principal Amount" field still contains "-1000". A red border highlights the error message. A modal alert box at the bottom center says "We hit a snag." with the sub-instruction "Review the following fields • Principal Amount".

## Test Case 2:

- **Use case:** Verify Loan record can be created successfully.
- **Input:**

Name: Ravindra Babu,

Email: [ravindrababu.jonnadula@gmail.com](mailto:ravindrababu.jonnadula@gmail.com)

Interest rate: 10%, Principal Amount = Rs.100000

The screenshot shows the Salesforce Lightning interface for creating a new loan record. The 'Name' field is populated with 'Loan001'. The 'Customer' field is set to 'Ravindra Babu'. In the 'Financials' section, the 'Principal Amount' is set to '₹1,00,000' and the 'Interest Rate' is set to '10.0%'. The 'Term Months' field is set to '12'. At the bottom right, there are 'Cancel', 'Save & New', and a large blue 'Save' button. On the left, a sidebar shows a list of recently viewed loans, including 'Ravindra Babu', 'Loan007', 'Loan002', 'Loan005', and 'Loan003'. A top navigation bar includes links for 'Payment Tracking A...', 'Dashboards', and 'Loans'.

- **Expected Result:** Loan Record created successfully.
- **Output:**

The screenshot shows the Salesforce Lightning interface displaying a success message: 'Loan "Ravindra Babu" was created.' The page shows the 'Basic Information' and 'Financials' sections of the loan record. The 'Basic Information' section includes fields for Name ('Ravindra Babu'), Customer ('Ravindra Babu'), Owner ('Bharath Jonnadula'), and other details like Customer Email and Customer Code. The 'Financials' section shows the Principal Amount as '₹1,00,000', Interest Rate as '10.0%', and Term Months as '12'. The top navigation bar includes links for 'Payment Tracking A...', 'Dashboards', 'Reports', 'Loans', 'Payments', and 'Recoveries'.

### Test Case 3:

- **Use case:** Verify Payment record links correctly to Loan.

- **Input:**

Loan: Loan001,

Amount: Rs.10000,

Email: [ravindrababu.jonnadula@gmail.com](mailto:ravindrababu.jonnadula@gmail.com),

Status: Pending, Due date: 25-09-2025

The screenshot shows the 'Information' form for creating a new payment record. The form fields are as follows:

- Payment Date: 25/09/2025
- Amount: ₹10,000.00
- Payment Method: Cash
- Status: Pending
- Loan: Loan001
- Customer Email: ravindrababu.jonnadula@gmail.com

At the bottom of the form are three buttons: 'Cancel', 'Save & New', and 'Save'.

- **Expected Result:** Payment Record created successfully.

- **Output:**

The screenshot shows the newly created payment record details. The payment record is identified by the ID LN-25-0008. The record includes the following information:

- Name: LN-25-0008
- Payment Date: 25/09/2025
- Amount: ₹10,000.00
- Payment Method: Cash
- Status: Pending
- Loan: [Loan001](#)
- Customer Email: ravindrababu.jonnadula@gmail.com
- Created By: Bharath Jonnadula, 25/09/2025, 6:03 pm
- Last Modified By: Bharath Jonnadula, 25/09/2025, 6:03 pm

A green success message at the top right of the page states: "Payment 'LN-25-0008' was created."

## Test Case 4:

- **Use case:** Payment due mail.

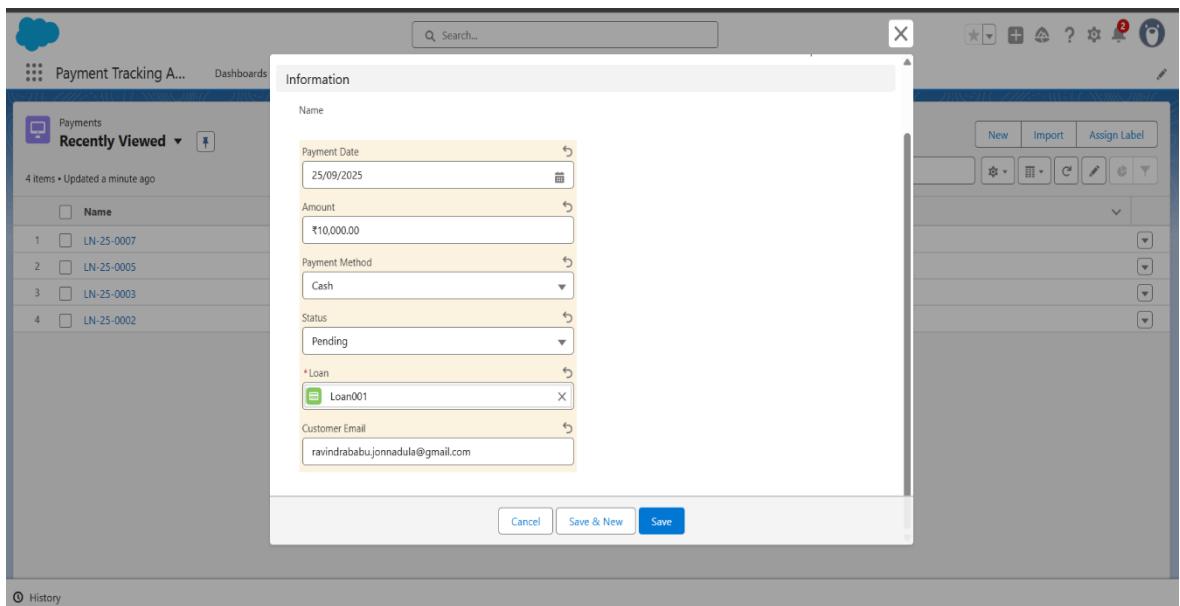
- **Input:**

Loan: Loan001,

Amount: Rs.10000,

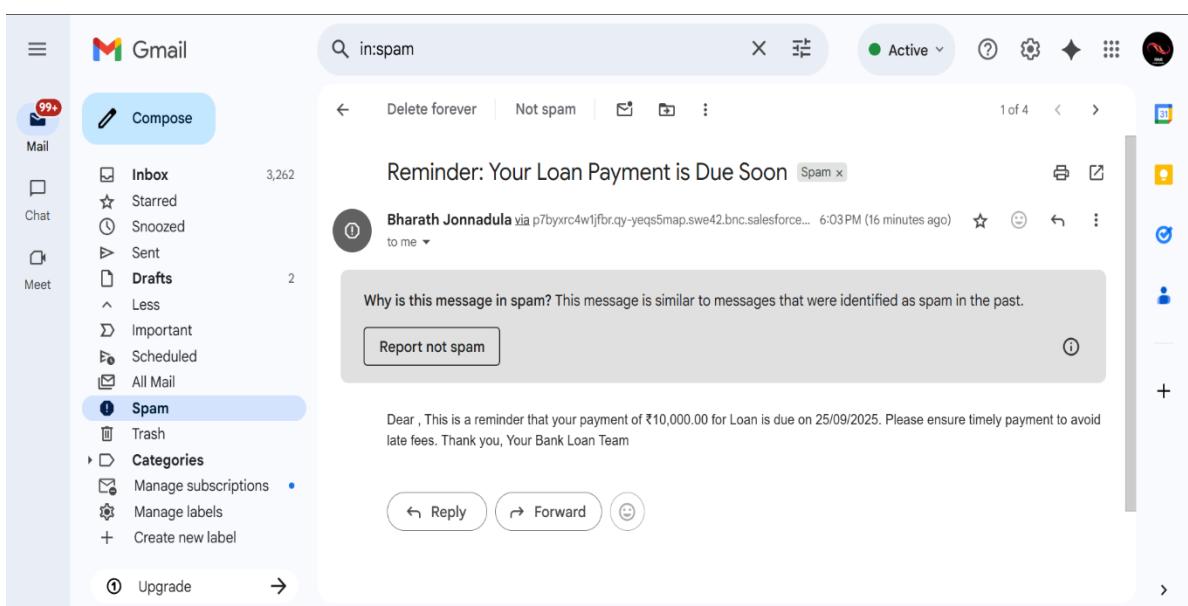
Email: [ravindrababu.jonnadula@gmail.com](mailto:ravindrababu.jonnadula@gmail.com),

Status: Pending, Due date: 25-09-2025



- **Expected Result:** Payment Remainder Email.

- **Output:**



## Test Case 5:

- **Use case:** Verify Loans with Payments Report.
- **Input:** Loans, Payment data

Loan ID	Name	Principal Amount	Interest Rate	Term Months
Loan001	Ravindra Babu	₹1,00,000	10.09%	12
Loan002	Pardha Praneeth	₹1,00,000	9.09%	24

- **Expected Result:** Report should show Loan: Name, Payment: Name, Status, Email, Groupby : Amount.
- **Output:**

Amount	Payment Date	Status	Loan: Name	Payment: Name	Status	Customer Email
₹10,00,000 (3)	02/09/2025 (1)	Overdue (1)	Loan002	LN-25-0002	Active	pardhapraneethjonnadula@gmail.com
		Subtotal				
	24/09/2025 (1)	Pending (1)	Loan007	LN-25-0007	Active	jonnadulabharath@gmail.com
		Subtotal				
	25/09/2025 (1)	Pending (1)	Loan001	LN-25-0008	Active	ravindrababu.jonnadula@gmail.com
		Subtotal				
	₹12,00,000 (1)	Overdue (1)	Loan003	LN-25-0003	Active	praneethjp03@gmail.com
		Subtotal				

## Test Case 6:

- **Use case:** Verify dashboard updates after new payment.
- **Input:** Loans, Payment data

The image shows two side-by-side screenshots of a software application titled "Payment Tracking A...". Both screenshots have a top navigation bar with tabs: Dashboards, Reports, Loans, Payments, and Recoveries. The "Loans" tab is selected in both.

**Screenshot 1 (Left): Basic Information for Loan001**

Name	Loan001
Customer	Ravindra Babu
Owner	Bharath Jonnadula
Customer Email	ravindrababu.jonnadula@gmail.com
Customer Code	Cusi001

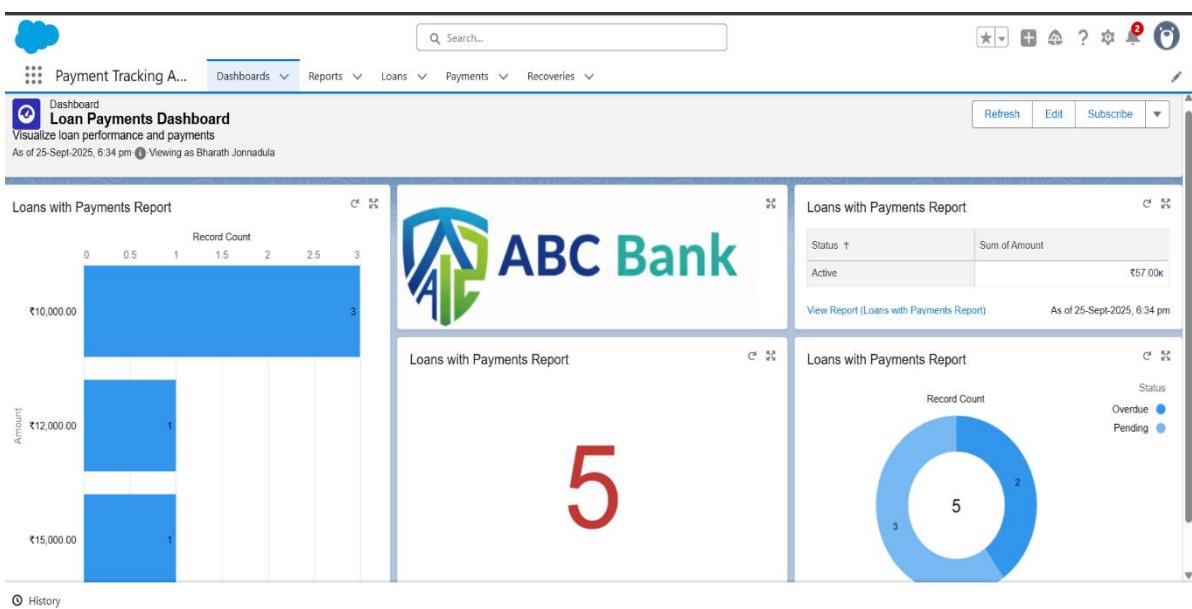
**Screenshot 2 (Right): Basic Information for Loan002**

Name	Loan002
Customer	Pardha Praneeth
Owner	Bharath Jonnadula
Customer Email	pardhapraneeth.jonnadula@gmail.com
Customer Code	CLUST002

**Financials Section (Both Screenshots)**

Principal Amount	₹1,00,000
Interest Rate	10.0%
Term Months	12

- 
- **Expected Result:** Loan and Payment summary dashboard.
- **Output:**



## Test Case 7:

- **Use case:** Verify Recovery record links correctly to Loan.

- **Input:**

Recovery Date: 25-09-2025,

Assigned Agent: Recovery Agent,

Status: Assigned,

Loan: Loan001

The screenshot shows the 'New Recovery' form. The 'Information' section includes fields for Recovery Number, Recovery Date (set to 25/09/2025), Amount Recovered (empty), Assigned Agent (set to 'Recovery Agent'), Status (set to 'Assigned'), and Loan (set to 'Loan001'). The 'Owner' field is populated with 'Bharath Jonnadula'. At the bottom of the form are three buttons: 'Cancel', 'Save & New', and 'Save'.

- **Expected Result:** Recovery Record created successfully.

- **Output:**

The screenshot shows the newly created Recovery record. A green success message at the top right of the screen states 'Recovery "LN-25-0003" was created.' The main area displays the record details: Recovery Number (LN-25-0003), Recovery Date (25/09/2025), Assigned Agent (Recovery Agent), Status (Assigned), and Loan (Loan001). The 'Created By' field shows 'Bharath Jonnadula, 25/09/2025, 6:43 pm' and the 'Last Modified By' field also shows 'Bharath Jonnadula, 25/09/2025, 6:43 pm'. Below the record, a note states 'No related lists to display'. At the bottom left, there is a link to 'Sample Flow Report: Screen Flows'.

## Test Case 8:

- **Use case:** Verify Duplicate prevention for Loan records.
- **Input:**

Name: Ravindra Babu,

Email: [ravindrababu.jonnadula@gmail.com](mailto:ravindrababu.jonnadula@gmail.com)

Interest rate: 10%, Principal Amount = Rs.100000

The screenshot shows the Salesforce interface for creating a new loan record. The 'Name' field is populated with 'Loan001'. The 'Customer' field shows 'Ravindra Babu'. In the 'Financials' section, the 'Principal Amount' is set to ₹1,00,000 and the 'Interest Rate' is set to 10.0%. The 'Save' button is visible at the bottom right of the form.

- **Expected Result:** Error blocking the duplicates.

- **Output:**

The screenshot shows the same loan creation process as above, but now with an error message. A red box with the text 'We hit a snag.' and 'Review the errors on this page.' is displayed. Below it, a list of errors states: 'duplicate value found: Customer\_Code\_\_c duplicates value on record with id: a00Qy00001IDo3lAD'. The 'Save' button is still present at the bottom.

## Test Case 9:

- **Use case:** Verify Data Export feature works properly.
- **Input:** Loans, Payments, Recovery

The screenshot shows the Salesforce Setup interface with the 'Data' tab selected. On the left, there's a sidebar with various setup categories like Data Objects, Data Integration Metrics, and Duplicate Management. The 'Data Export' section is highlighted. The main content area is titled 'SETUP Data Export'. It lists numerous data objects as checkboxes, with several items checked, including 'Loan\_\_c' and 'Payment\_\_c'. A vertical scrollbar is visible on the right side of the content area.

- **Expected Result:** Receive a mail regarding data export.
- **Output:**

The screenshot shows an email inbox with one message from 'Do not reply <noreply@salesforce.com>' with the subject 'Your Organization Data Export has completed - ABC Bank Payment Tracking'. The message body contains a link: <https://resilient-raccoon-nqvtni-dev-ed.trailblaze.my.salesforce.com/ui/setup/export/DataExportPage/d>. The email was sent at 19:10 (0 minutes ago). The inbox also shows other messages and icons for search, filters, and more.

## Test Case 10:

- **Use case:** Payment remainder email before due date.

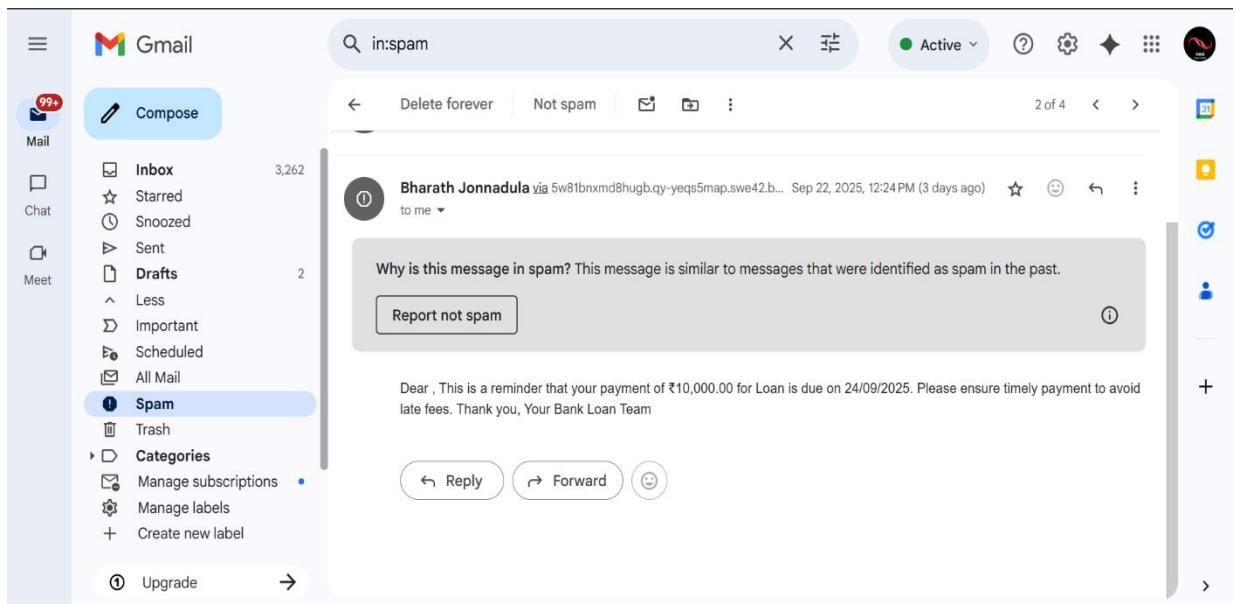
- **Input:**

Payment Status: Pending

Payment Date: Two days before payment

- **Expected Result:** Receive a mail .

- **Output:**



## **Conclusion:**

The Payment Tracking in Banking CRM successfully demonstrates how Salesforce can be leveraged to automate banking workflows, integrate approvals, and enhance payment recovery efficiency. With features such as custom objects (Loan, Payment, Recovery), automated flows, validation rules, approval processes, and interactive dashboards, the system addresses critical gaps in traditional banking operations. By implementing this solution, banks can ensure timely collections, reduced operational errors, and higher transparency. The project also highlights how Salesforce CRM can be adapted for industry-specific use cases like loan management and financial services. Overall, the solution provides a scalable foundation for banks to modernize customer repayment processes and drive better financial outcomes.