

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.

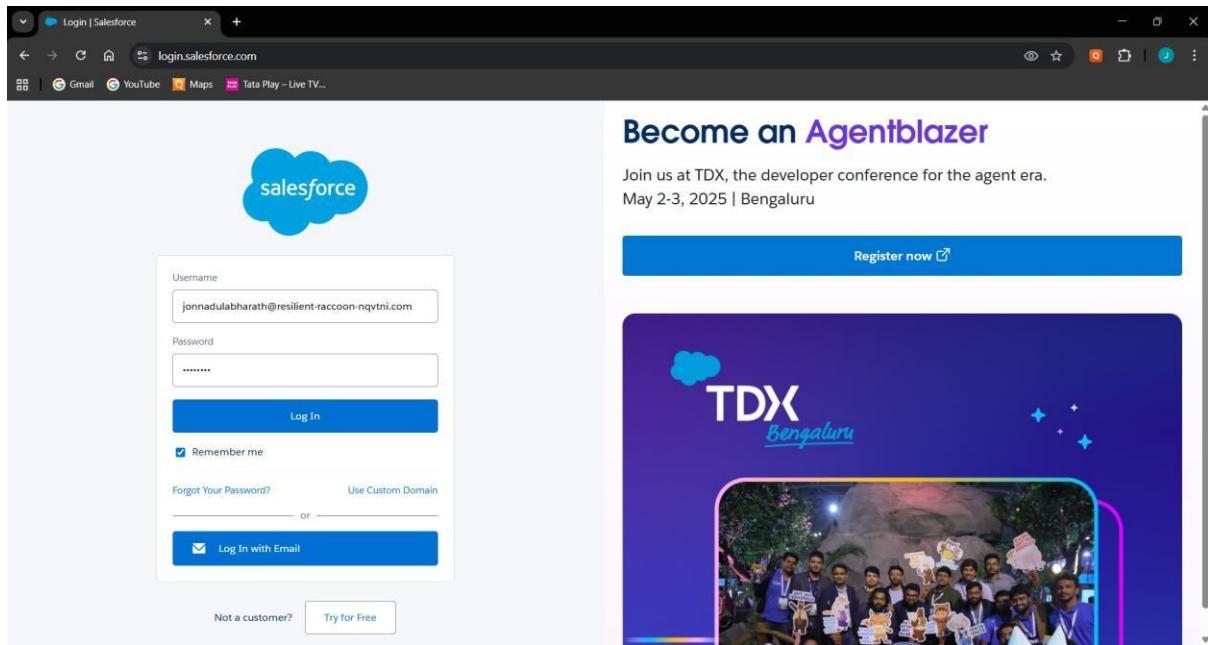
AppExchange Exploration

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

Phase 2: Org Setup & Configuration

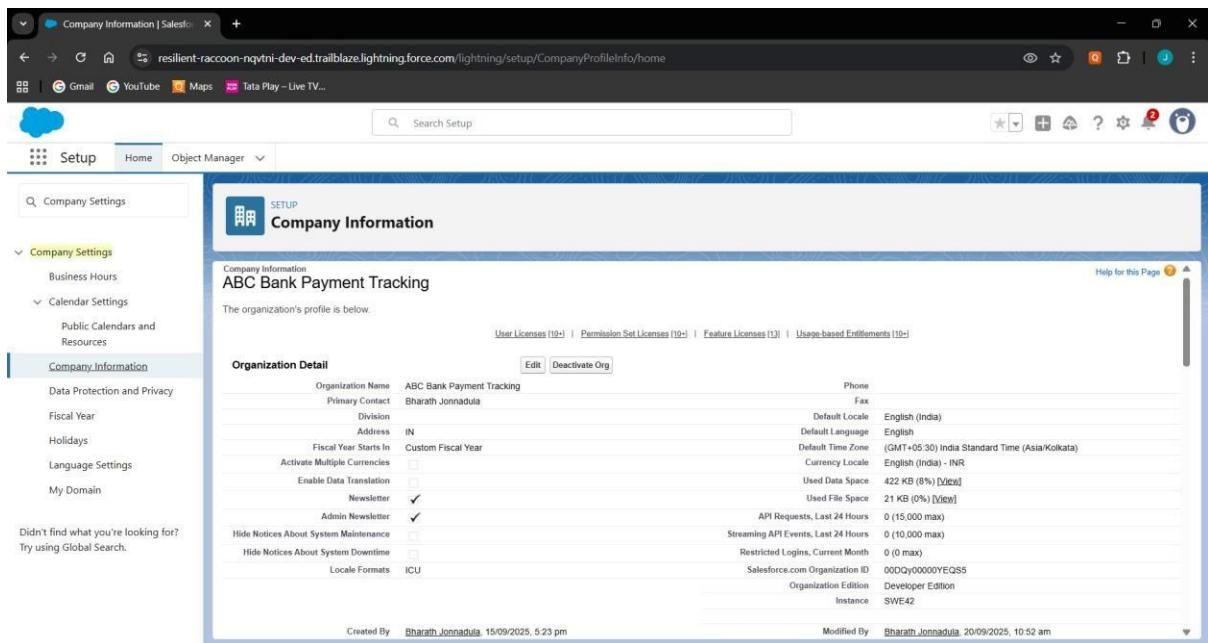
Salesforce Editions

- 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 working 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 includes sections for 'Custom Fiscal Years' (a table showing none defined) and 'Custom Fiscal Year Names' (a table showing columns: Action, Field Label, Edit | Replace, Quarter Prefix, Edit | Replace, Period Prefix, Edit | Replace, Quarter Name, Edit | Replace, 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' tab is selected, displaying fields for First Name ('Branch'), Last Name ('Manager'), Alias ('manager'), Email ('bankmanager.officer@bank'), Username ('bankmanager.officer@bank'), Nickname ('manager'), Title (''), Company (''), Department (''), and Division (''). On the right side, the 'Role' is set to '<None Specified>', 'User License' to 'Salesforce', 'Profile' to 'Manager Profile', and 'Active' is checked. Other optional checkboxes 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 available but unchecked. A note at the bottom indicates a monthly addition limit of 300.

The screenshot shows the Salesforce Setup interface under the 'Users' section. A new user record is being created for 'Loan Officer'. The 'General Information' tab is selected, displaying fields for First Name ('Loan'), Last Name ('Officer'), Alias ('loan'), Email ('loan.officer@bankdemo.cor'), Username ('loan.officer@bankdemo.cor'), Nickname ('loanofficer'), Title (''), Company (''), Department (''), and Division (''). On the right side, the 'Role' is set to '<None Specified>', 'User License' to 'Salesforce Platform', 'Profile' to 'Standard Platform User', and 'Active' is checked. Other optional checkboxes 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 available but unchecked. A note at the bottom indicates a monthly addition limit of 300.

Profiles & Roles

- Create custom profiles (Loan Officer, Manager, Recovery Agent).
- Establish role hierarchy (Manager → Loan Officer / Recovery Agent).

Permission Sets

- Create a Permission Set for payment tracking access.
- Assign Permission Set to Loan Officers and Managers.

The screenshot shows the 'Permission Sets' page in the Salesforce Setup. A permission set named 'Payment Team Access' is selected. The 'Permission Set Overview' section displays details like API Name (Payment_Team_Access), Namespace Prefix, and creation and modification history. The 'Apps' section lists various app settings such as Assigned Apps, Assigned Connected Apps, Object Settings, App Permissions, Apex Class Access, and Visualforce Page Access.

OWD & Sharing Rules

- Set custom objects (Loan, Payment, Recovery) to Private.
- Create rules to share overdue records with Recovery Agents.

The screenshot shows the 'Sharing Settings' page in the Salesforce Setup. It displays a table of sharing rules for various objects. The columns show the object name, sharing rule type (e.g., Private, Public Read Only), and sharing rule status (indicated by checkmarks). Objects listed include Quick Text 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.

Object	Type	Status
Quick Text 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	✓

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 contains sections for 'Session Timeout' (timeout value set to '30 minutes') and 'Session Settings' (checkboxes for locking sessions by IP, domain, or password). A note at the bottom states: "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".

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 from the database. The code editor has syntax highlighting for Apex and shows line numbers. The Explorer sidebar on the left shows project files like 'force-app/main/default/classes' and various Apex classes. The terminal at the bottom shows command-line output related to the deployment process.

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 contains 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):

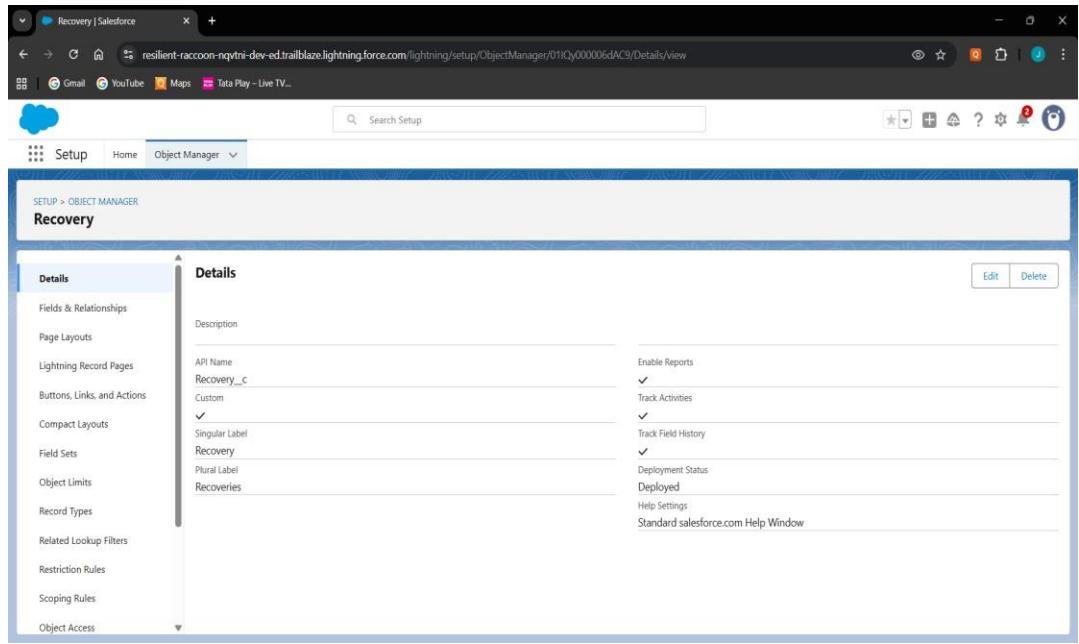
- Enable Reports: ✓
- Track Activities: ✓
- Track Field History: ✓
- Deployment Status: Deployed
- Help Settings: Standard salesforce.com Help Window

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. On the left, a 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 has columns for 'RECORD TYPE LABEL', 'DESCRIPTION', 'ACTIVE', and 'MODIFIED BY'. Both records were modified by 'Bharath Jonnadula' on 20/09/2025, 5:10 pm.

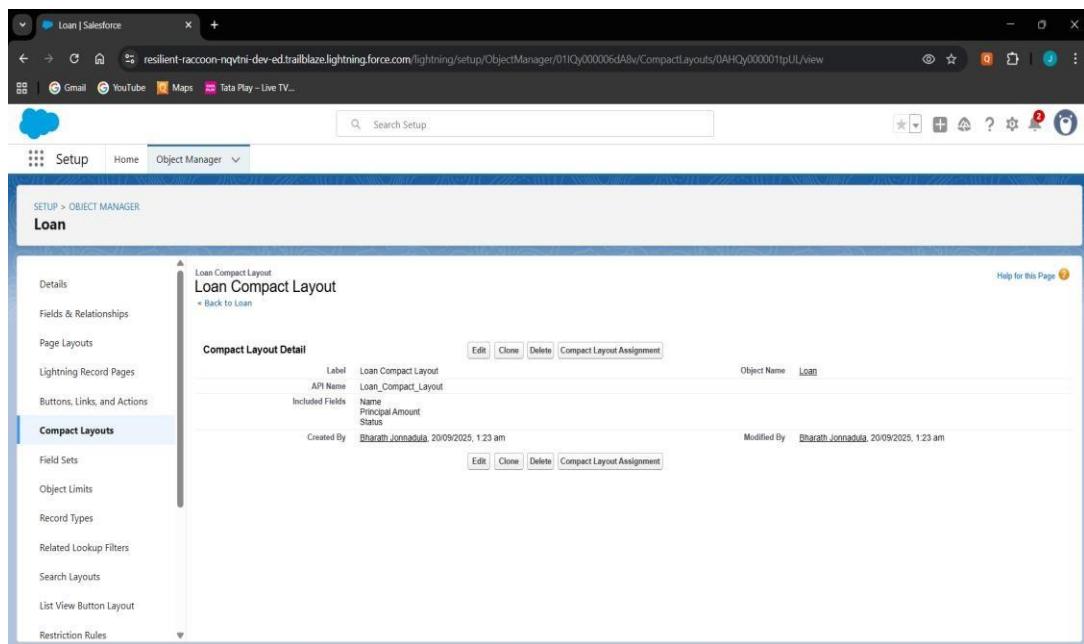
Page Layouts

- 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 editing the 'Loan Layout'. The left sidebar shows the same list of configuration options as the previous screenshot. The main content area is titled 'Loan Layout' and includes tabs for 'Fields', 'Buttons', 'Quick Actions', 'Mobile & Lightning', 'Actions', 'Expanded Lookups', 'Related Lists', 'Details', and 'Record Types'. A 'Layout Properties' section at the top right contains tabs for 'Customize', 'Comments', 'Preview As...', 'Cancel', 'Save', 'Quick Save', 'Photo', and 'Layout Properties'. Below these are sections for 'Loan Sample', 'Highlights Panel', 'Quick Actions In the Salesforce Classic Publisher', and 'Salesforce Mobile and Lightning Experience Actions'. A note at the bottom states: 'Actions in this section are currently inherited from the global publisher layout. You can override the global publisher layout to set a customized list of actions for the publisher on pages that use this layout.'

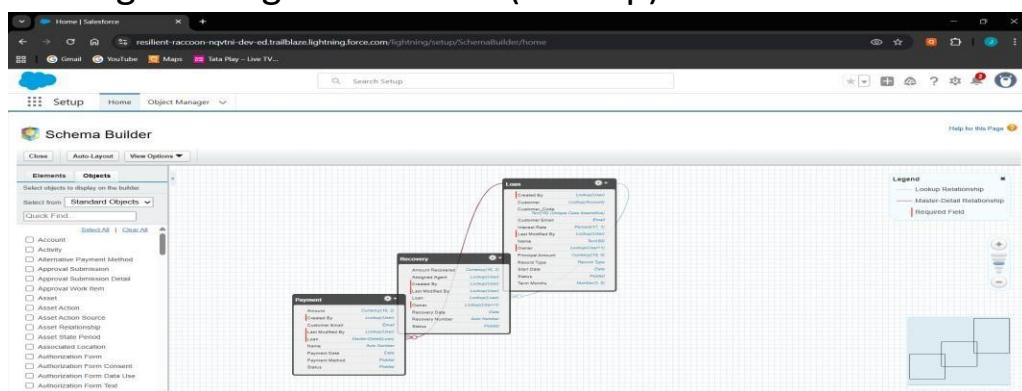
Compact Layouts

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



Schema Builder

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



Lookup vs Master-Detail vs Hierarchical

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

Junction Objects

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

External Objects

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

Phase 4: Process Automation (Admin)

Validation Rules

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

The image displays two screenshots of the Salesforce Object Manager interface, specifically focusing on 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

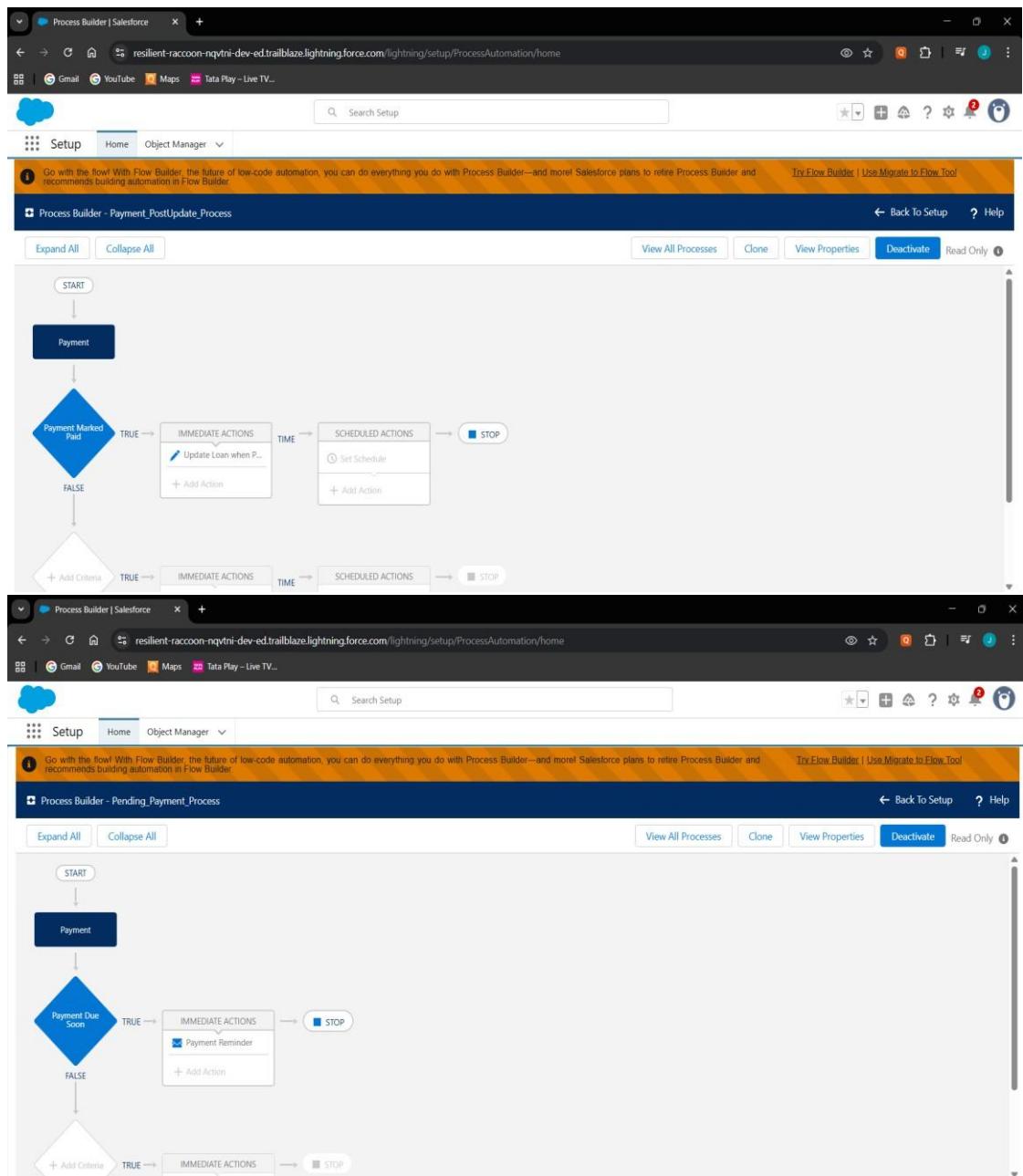
- Rule: Pending_Payment_Reminder.
- Sends reminder email for pending payments.

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. The process is named 'Loan: Loan_High_Amount_Approval'. The 'Process Definition Detail' section 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)
- Allow Submitters to Recall Approval Requests:** unchecked
- Approval Assignment Email Template:** Initial Submitters
- Initial Submitters:** Loan Owner
- Created By:** Bharath.Jonnadula, 20/09/2025, 4:08 pm
- Modified By:** Bharath.Jonnadula, 22/09/2025, 10:49 am

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 for a 'Record-Triggered Flow' on the 'Payment' object. The flow consists of the following steps:

```

graph TD
    Start((Record-Triggered Flow)) --> RunImmediately[Run Immediately]
    RunImmediately --> MarkOverdue[Mark Payment Overdue]
    MarkOverdue --> End([End])
    
```

The 'Configure Start' panel shows:

- Select Object:** Payment
- Trigger:** A record is created or updated
- Conditions:** 2
- Optimize for:** Actions and Related Records

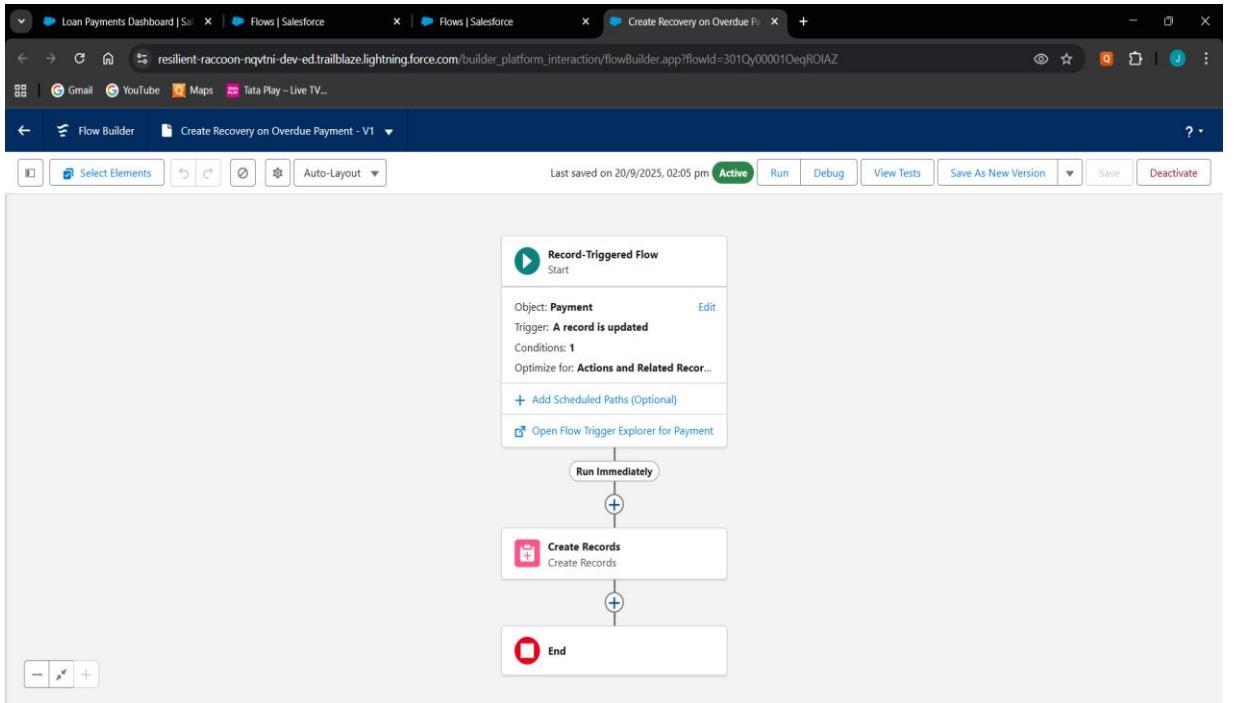
The 'Configure Trigger' panel shows:

- Trigger the Flow When:**
 - A record is created
 - A record is updated
 - A record is created or updated
 - A record is deleted

The 'Set Entry Conditions' panel shows:

- 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" tab. 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' options. 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 configuration:

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:

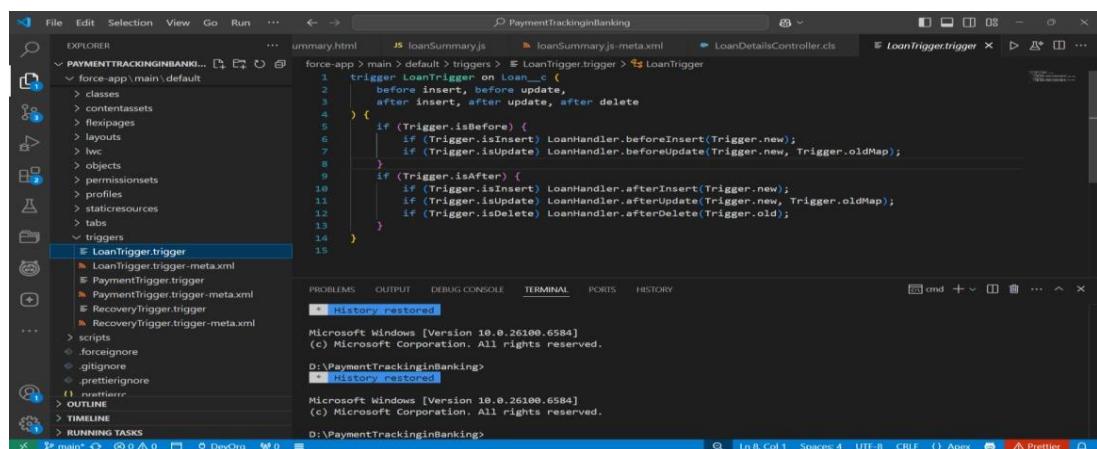
- File Explorer:** On the left, it shows the project structure under "PAYOUTTRACKINGINBANKING". The "classes" folder contains several Apex classes: PaymentController.cls, PaymentHandler.cls, PaymentHandlerTest.cls, PaymentHelper.cls, PaymentReminderQueue.cls, PaymentReminderQueue.cls-meta.xml, PaymentSearchDemo.cls, 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, specifically the beforeInsert, beforeUpdate, and afterDelete methods.
- Terminal:** At the bottom, the terminal window shows the command line output of running the Salesforce CLI (sf) to deploy the project. It includes the command `sf project deploy start --source-dir force-app/main/default/classes` and the response `History restored`.
- Bottom Status Bar:** The status bar at the bottom provides information about the current file ("main"), the number of changes (0), the active developer organization ("DevOrg"), and the file's location and encoding.

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);}}}
```

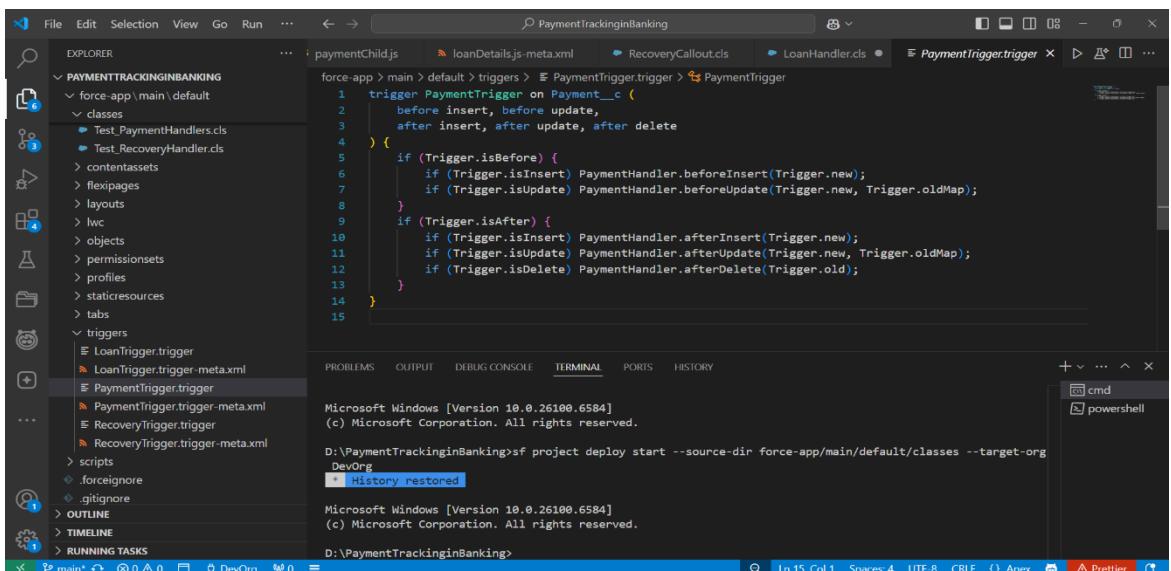


The screenshot shows the Visual Studio Code (VS Code) interface with the following details:

- Explorer View:** Shows the project structure under "force-app/main/default". The "triggers" folder is expanded, and "LoanTrigger.trigger" is selected, indicated by a blue background.
- Editor View:** Displays the Apex code for "LoanTrigger.trigger". The code handles four trigger types: before insert, before update, after insert, and after update. It uses the "LoanHandler" class to perform specific actions based on the trigger type and record status (before or after).
- Terminal View:** Shows the command "D:\PaymentTrackinginBanking> History restored" and "Microsoft Windows [Version 10.0.26100.6584] (c) Microsoft Corporation. All rights reserved." indicating the environment where the code was run.
- Status Bar:** Shows the file name "main", line "Ln 8, Col 1", spaces "Spaces: 4", encoding "UTF-8", and file type "Apex".

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. 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 trigger code. Below the code, the TERMINAL tab shows command-line output for a deployment, and the DEBUG CONSOLE tab shows logs from the browser. The bottom status bar indicates the file is "main*", line 15, column 1, with 0 errors and 0 warnings.

```
force-app > main > default > triggers > PaymentTrigger.trigger > PaymentTrigger
1 trigger PaymentTrigger on Payment__c (
2     before insert, before update,
3     after insert, after update, after delete
4 ) {
5     if (Trigger.isBefore) {
6         if (Trigger.isInsert) PaymentHandler.beforeInsert(Trigger.new);
7         if (Trigger.isUpdate) PaymentHandler.beforeUpdate(Trigger.new, Trigger.oldMap);
8     }
9     if (Trigger.isAfter) {
10        if (Trigger.isInsert) PaymentHandler.afterInsert(Trigger.new);
11        if (Trigger.isUpdate) PaymentHandler.afterUpdate(Trigger.new, Trigger.oldMap);
12        if (Trigger.isDelete) PaymentHandler.afterDelete(Trigger.old);
13    }
14 }
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS HISTORY

```
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>
```

Ln 15, Col 1 Spaces: 4 UTF-8 CRLF ⌂ Apex ⌂ Prettier ⌂

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) {

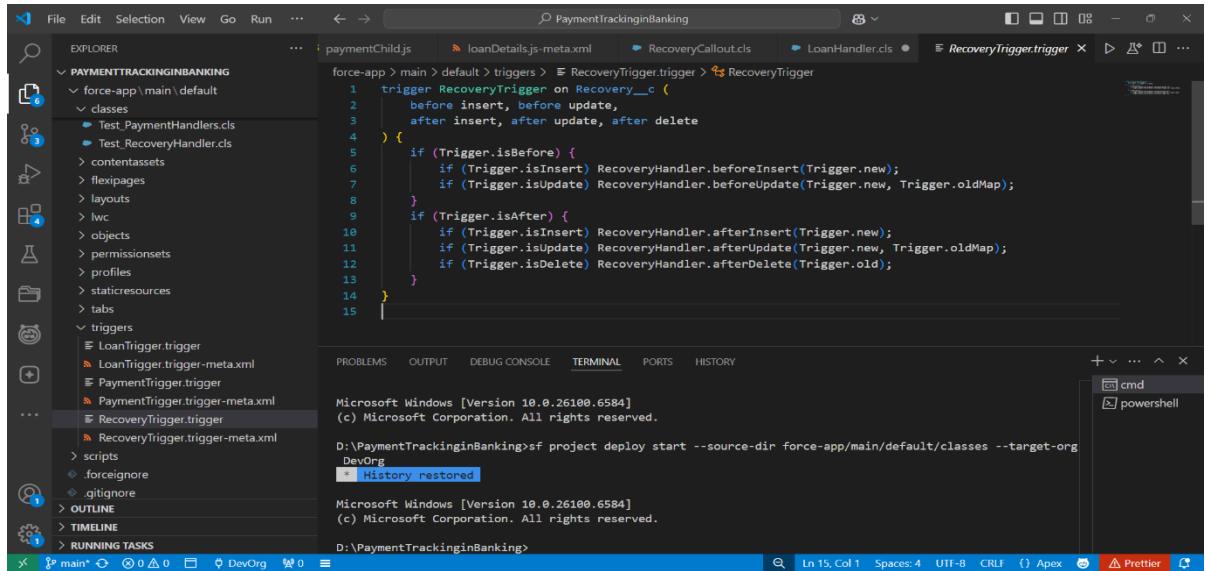
 if (Trigger.isInsert) RecoveryHandler.afterInsert(Trigger.new);

 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 both before and after triggers on the Recovery__c object. It includes logic for insert, update, and delete operations, calling methods like beforeInsert, beforeUpdate, afterInsert, and afterUpdate on the RecoveryHandler class.
- 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 "Ln 15, Col 1" and "Spaces: 4" and "UTF-8". There are also icons for Apex, Prettier, and other development tools.

SOQL & SOSL

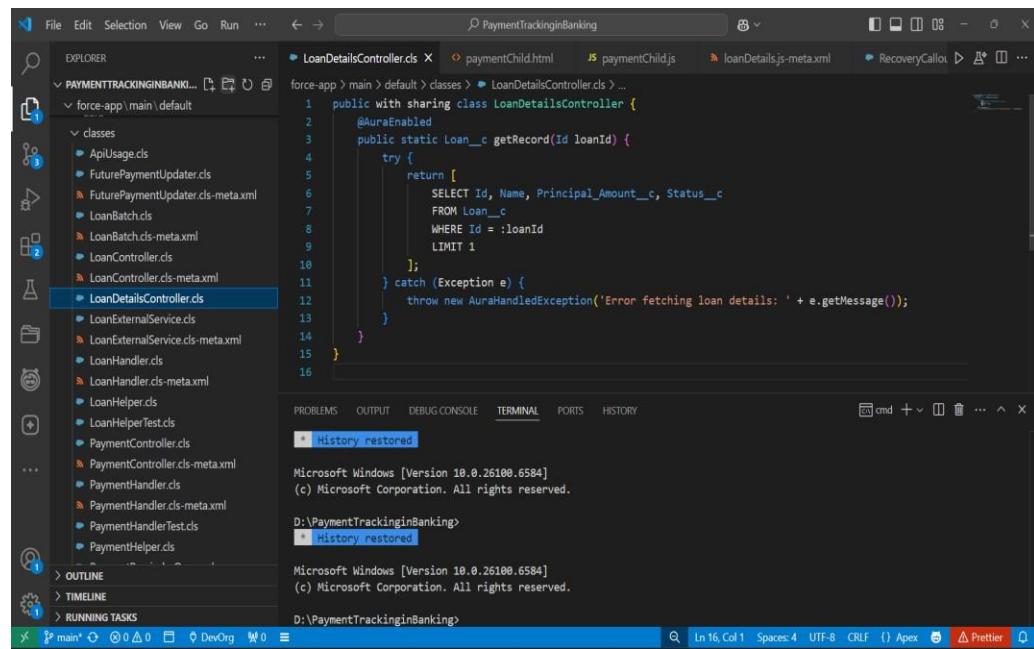
- Used SOQL in classes to query Loans, Payments, and Recoveries.
- Implemented simple search functionality to fetch customer records.

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());  
    }  
}
```



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

- Title Bar:** PaymentTrackinginBanking
- Explorer View:** Shows the file structure of the project. The file `LoanDetailsController.cls` is currently selected.
- Code Editor:** Displays the Apex code for `LoanDetailsController.cls`. The code implements a static method `markLoanClosed` that updates a `Loan__c` record to status 'Closed' if it exists.
- Terminal:** Shows the command history restored on both the Windows host and the Docker container.
- Status Bar:** Shows the current file is `main*`, has 0 errors and 0 warnings, and is connected to `DevOrg`.

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.

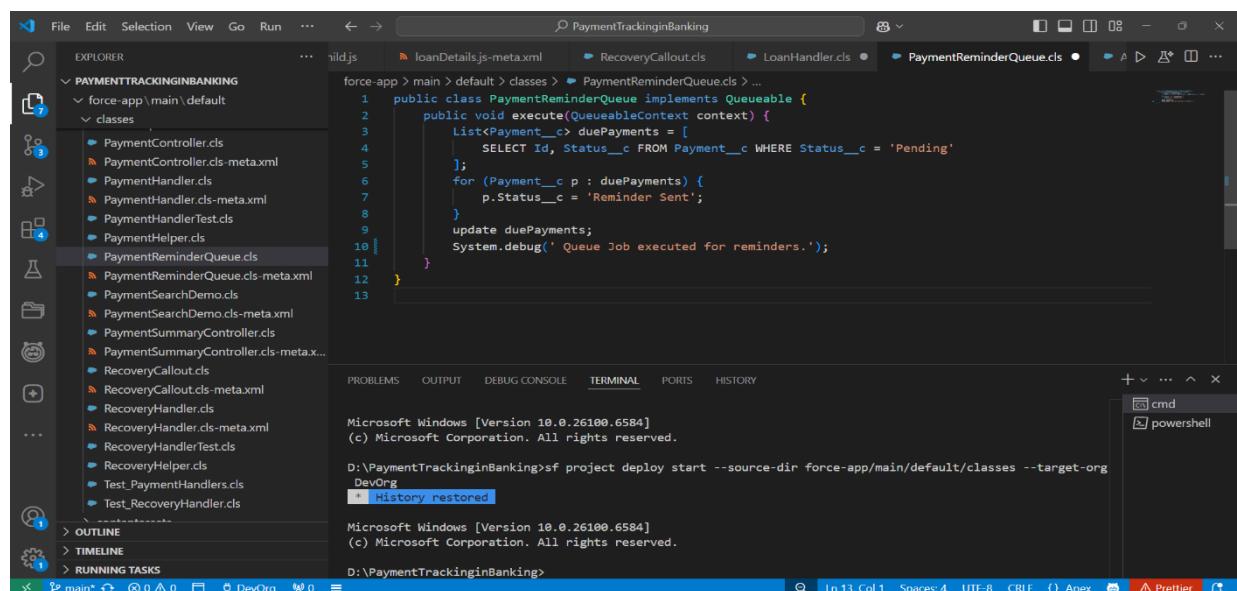
D:\PaymentTrackinginBanking>
```

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.');
    }
}
```



```
1  public class PaymentReminderQueue implements Queueable {
2      public void execute(QueueableContext context) {
3          List<Payment__c> duePayments = [
4              SELECT Id, Status__c FROM Payment__c WHERE
5              Status__c = 'Pending'
6          ];
7          for (Payment__c p : duePayments) {
8              p.Status__c = 'Reminder Sent';
9          }
10         update duePayments;
11         System.debug(' Queue Job executed for reminders.');
12     }
13 }
```

Microsoft Windows [Version 10.0.26100.6584]
© 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]
© Microsoft Corporation. All rights reserved.
D:\PaymentTrackinginBanking>

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);
}

```

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

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

```

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 }

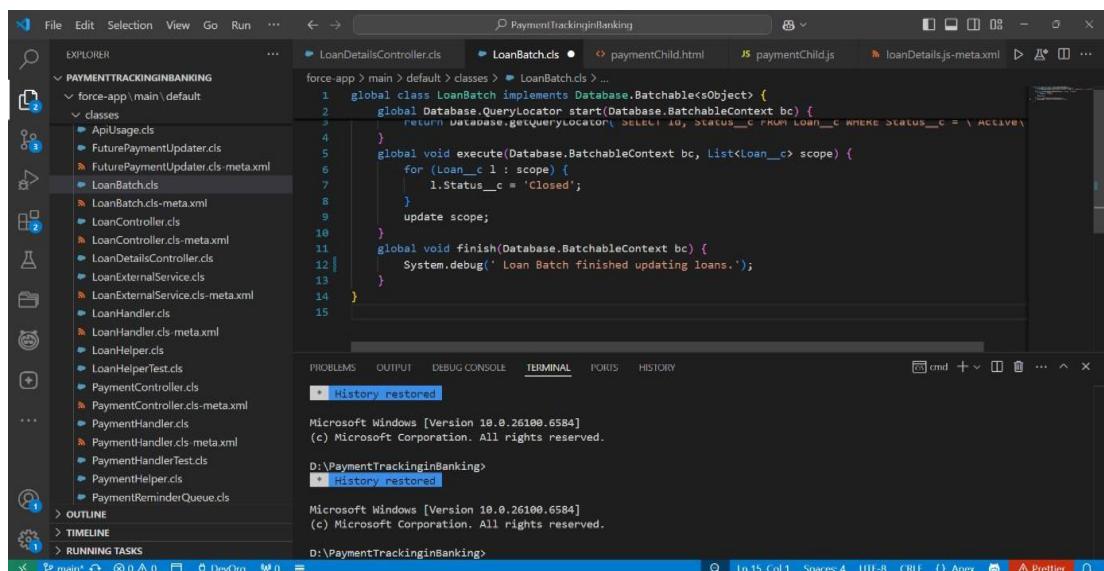
```
- Explorer Sidebar:** Shows the project structure under "PAYOUTTRACKINGINBANKING...".
- Bottom Status Bar:** Displays "main* 0 0 0 DevOrg 0" and "Ln 30, Col 1 Spaces: 4 UTF-8 CRLF {} Apex".

Batch Apex

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

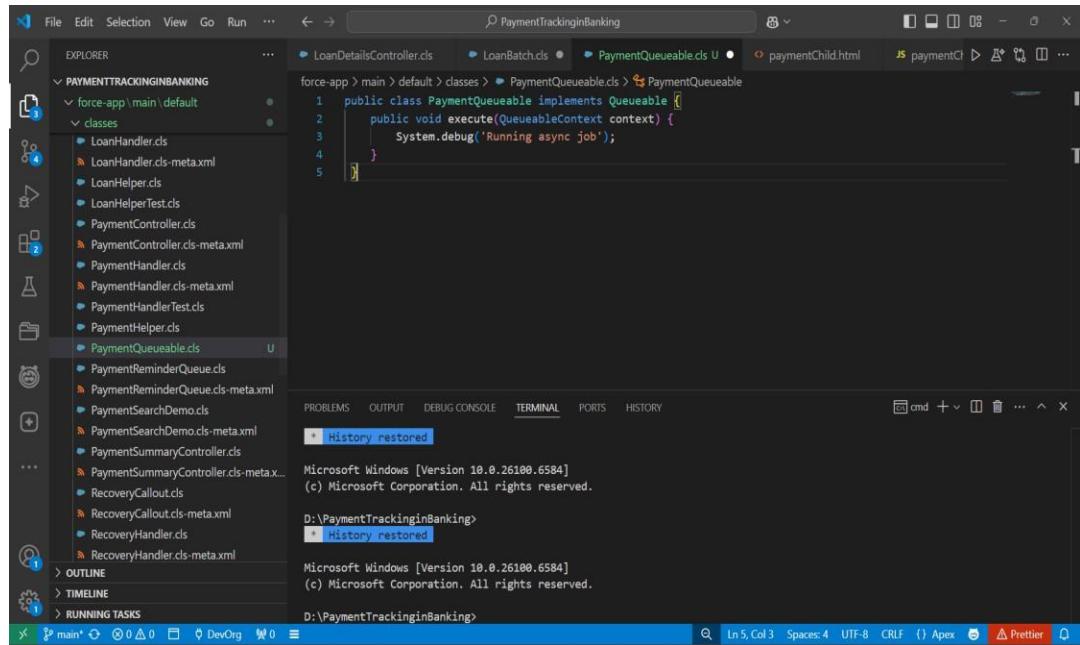
LoanBatch.cls

```
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":

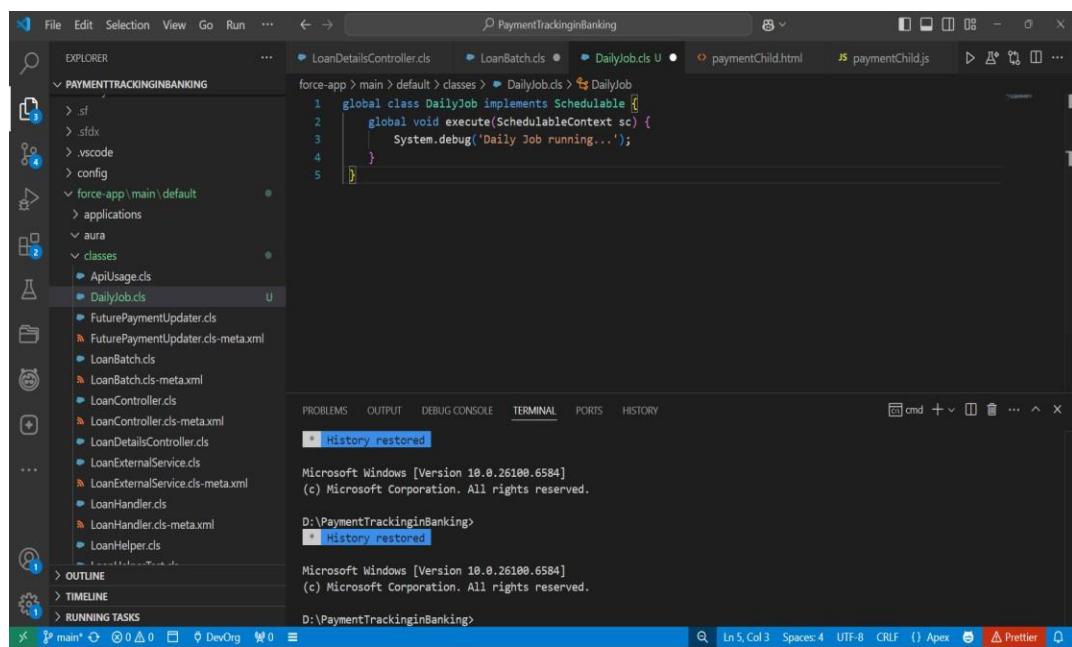
```
1  public class PaymentQueueable implements Queueable {
2      public void execute(QueueableContext context) {
3          System.debug('Running async job');
4      }
6  }
```
- Terminal:** Shows command-line history:

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

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

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      }
5  }
```
- Terminal:** Shows 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 file path "D:\PaymentTrackinginBanking", line "Ln 5, Col 3", spaces "Spaces: 4", encoding "UTF-8", and line endings "CRLF".

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
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS HISTORY

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 structure. The main area displays the `PaymentHelper.cls` code. The code implements two static methods: `beforeInsert` and `afterUpdate`. The `beforeInsert` method checks if the `Payment__c` record has a null `Payment_Date__c` field and throws an exception if so. The `afterUpdate` method logs a message and notifies an admin if an exception occurs during update. Below the code editor is the Terminal pane, which shows the command used to deploy the changes and a history restoration message.

```

force-app > main > default > classes > PaymentHelper.cls > PaymentHandler > beforeInsert(List<Payment__c>) : void
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 {
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);
    }
}
```

```
File Edit Selection View Go Run ... ⏪ PaymenttrackinginBanking
EXPLORER PAYMENTTRACKINGINBANKING force-app/main/default classes
LoanBatch.cls LoanController.cls
LoanController.cls-meta.xml
LoanDetailsController.cls
LoanExternalService.cls
LoanExternalService.cls-meta.xml
LoanHandler.cls
LoanHandler.cls-meta.xml
LoanHelper.cls
LoanHelperTest.cls
PaymentController.cls
PaymentController.cls-meta.xml
PaymentHandler.cls
PaymentHandler.cls-meta.xml
PaymentHandlerTest.cls
PaymentHelper.cls
PaymentReminderQueue.cls
PaymentReminderQueue.cls-meta.xml
PaymentSearchDemo.cls
PaymentSearchDemo.cls-meta.xml
OUTLINE
TIMELINE
RUNNING TASKS
main* 0 0 DevOrg 0 0
Ln 10, Col 11 Spaces: 4 UTF-8 CRLF {} Apex Prettier
```

```
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 }
```

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);
    }
}

```

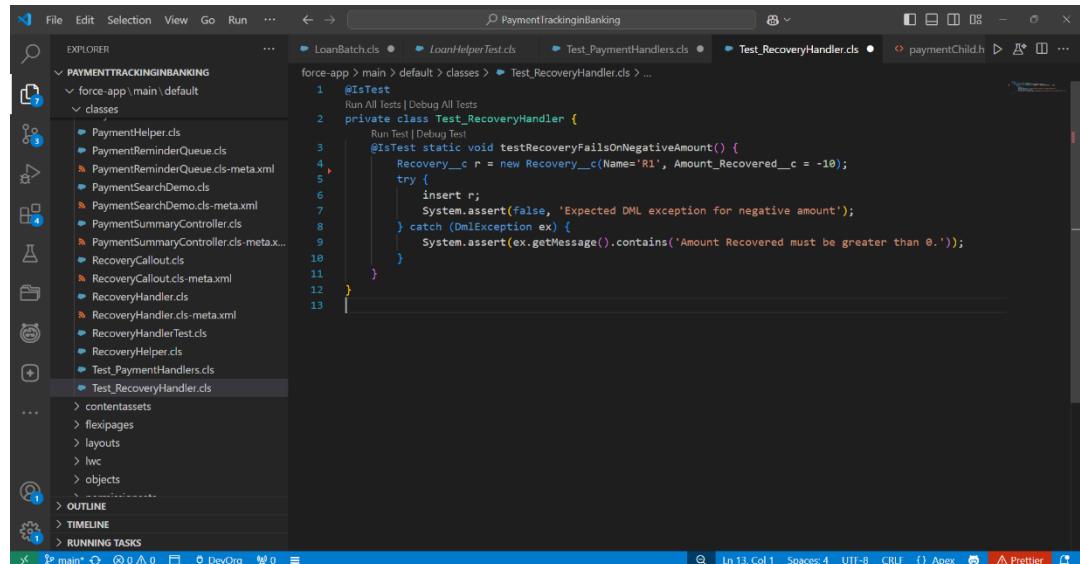
```

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.'));
        }
    }
}
```



```
}
```

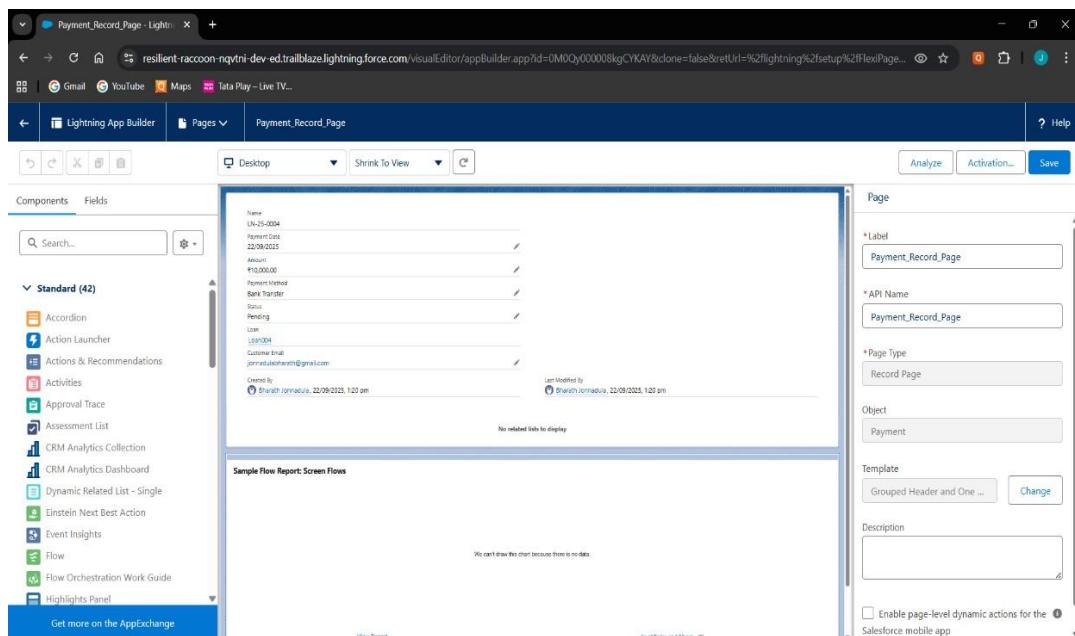
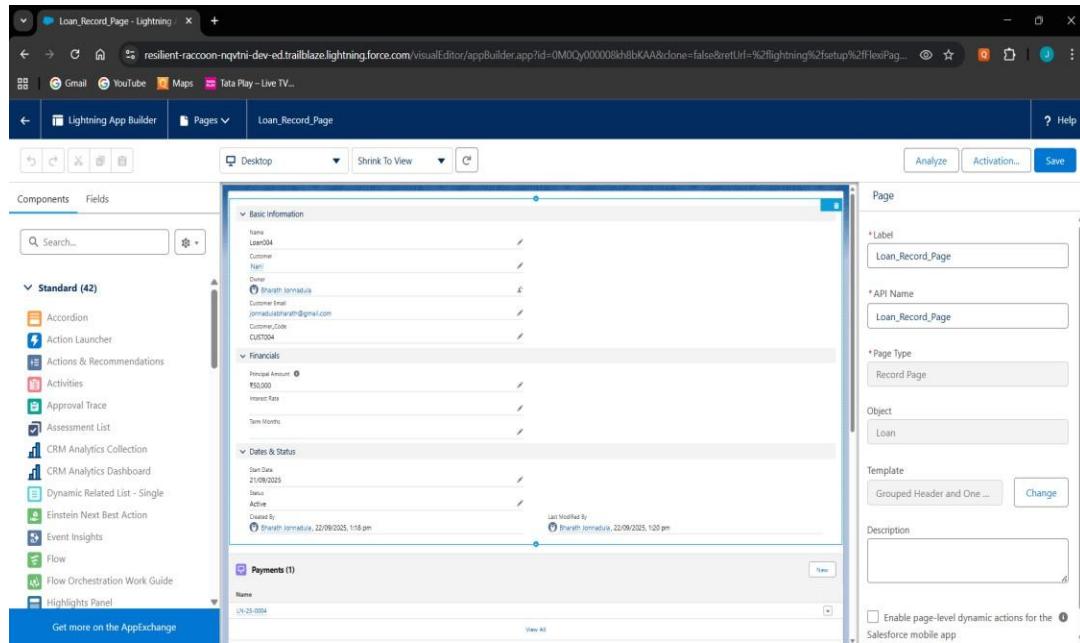
Asynchronous Processing

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

Phase 6: User Interface Development

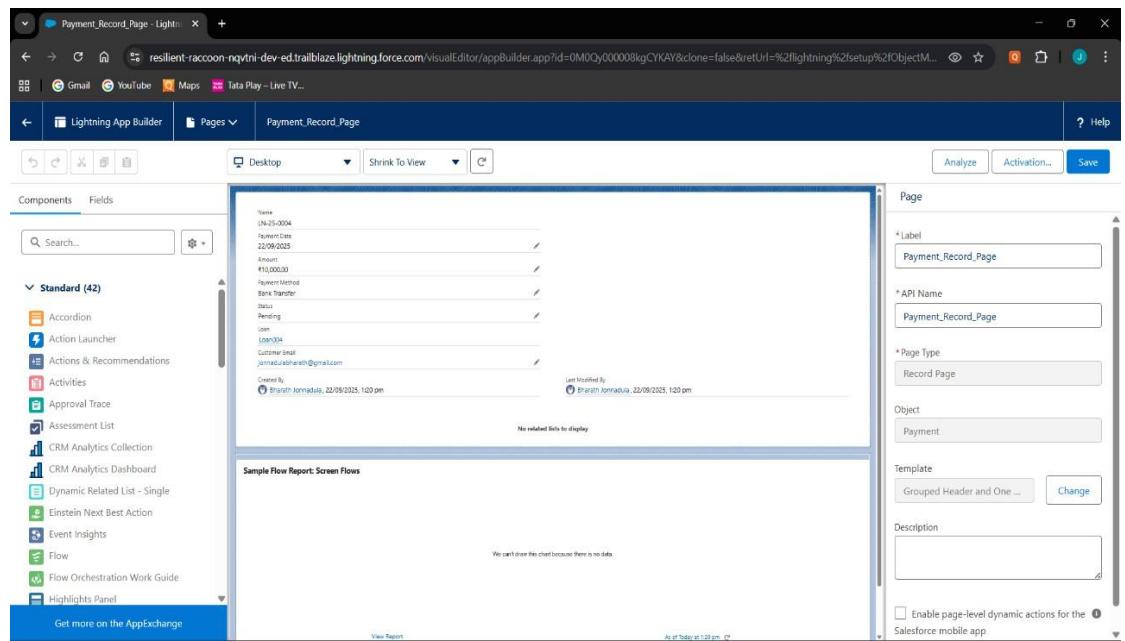
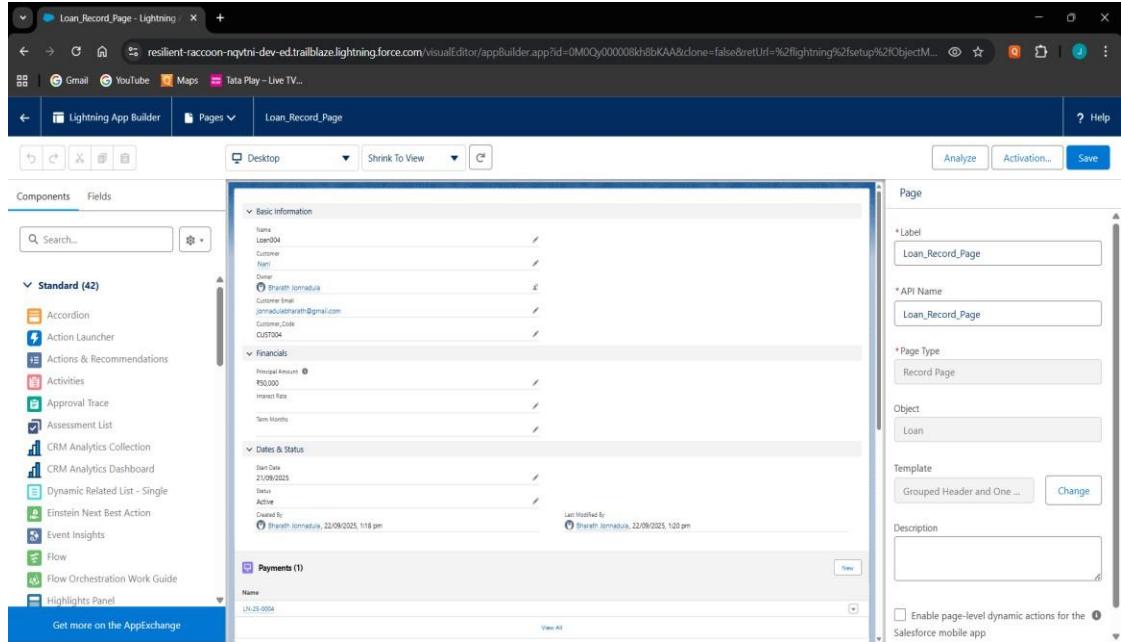
Lightning App Builder

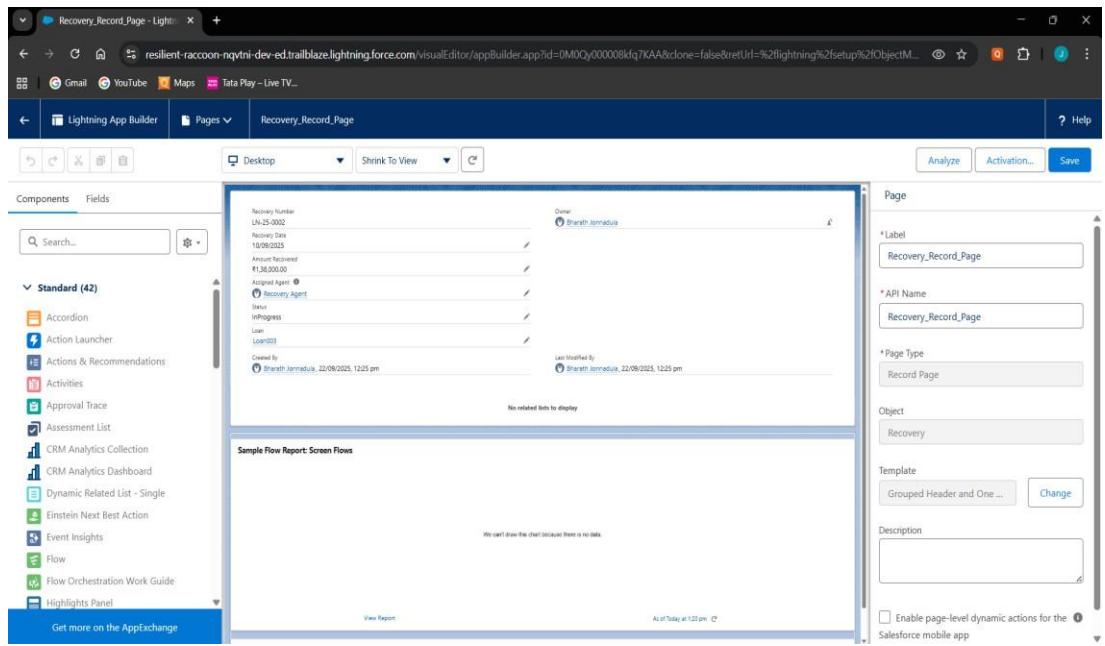
- Created a custom Loan_Record_Page, Payment_Home_Page, Payment_Record_Page, Recovery_record_Page



Record Pages

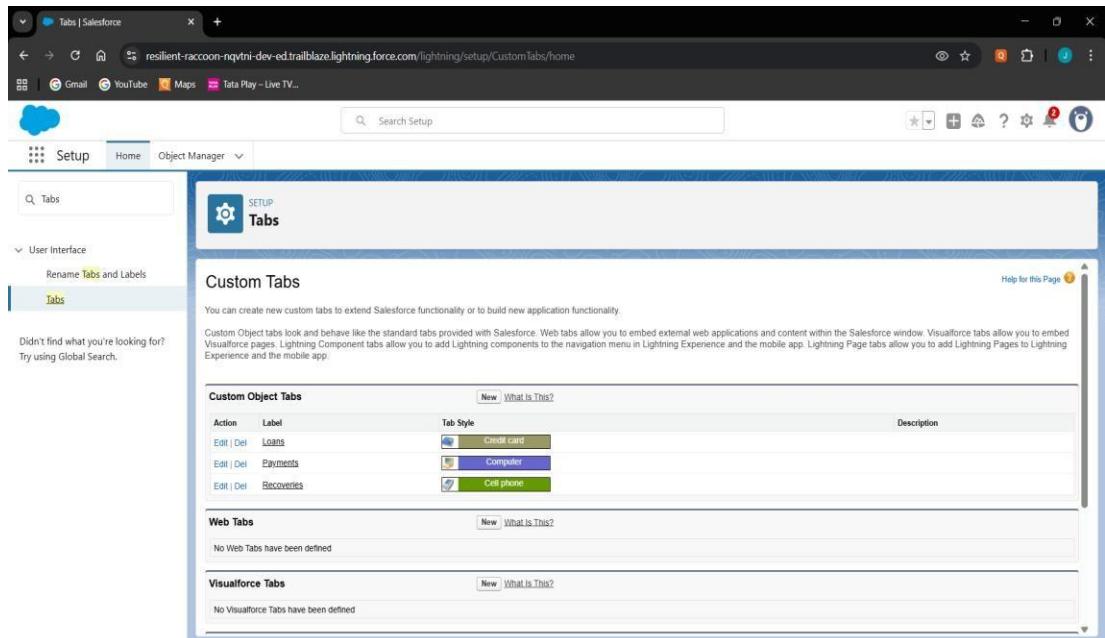
- Built custom Record Pages for Loan and Payment objects using Lightning App Builder.





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 'Feature Settings' expanded, showing 'Service' (Field Service), 'Field Service Mobile', 'Field Service Mobile App Builder', and 'User Interface' (Lightning App Builder selected). Below the sidebar, a message says 'Didn't find what you're looking for? Try using Global Search.' The main area is titled 'Lightning App Builder' and contains a section titled 'Lightning Pages'. It shows a table with the following data:

Action	Label	Name	Namespace Prefix	Description	Type	Created By	Last Modified By
Edit Clone Del	Loan_Record_Page	Loan_Record_Page			Record Page	Bjorn	Bjorn
Edit Clone Del	Payment_Home_Page	Payment_Home_Page			Home Page	Bjorn	Bjorn
Edit Clone Del	Payment_Record_Page	Payment_Record_Page			Record Page	Bjorn	Bjorn
Edit Clone Del	Recovery_Record_Page	Recovery_Record_Page			Record Page	Bjorn	Bjorn

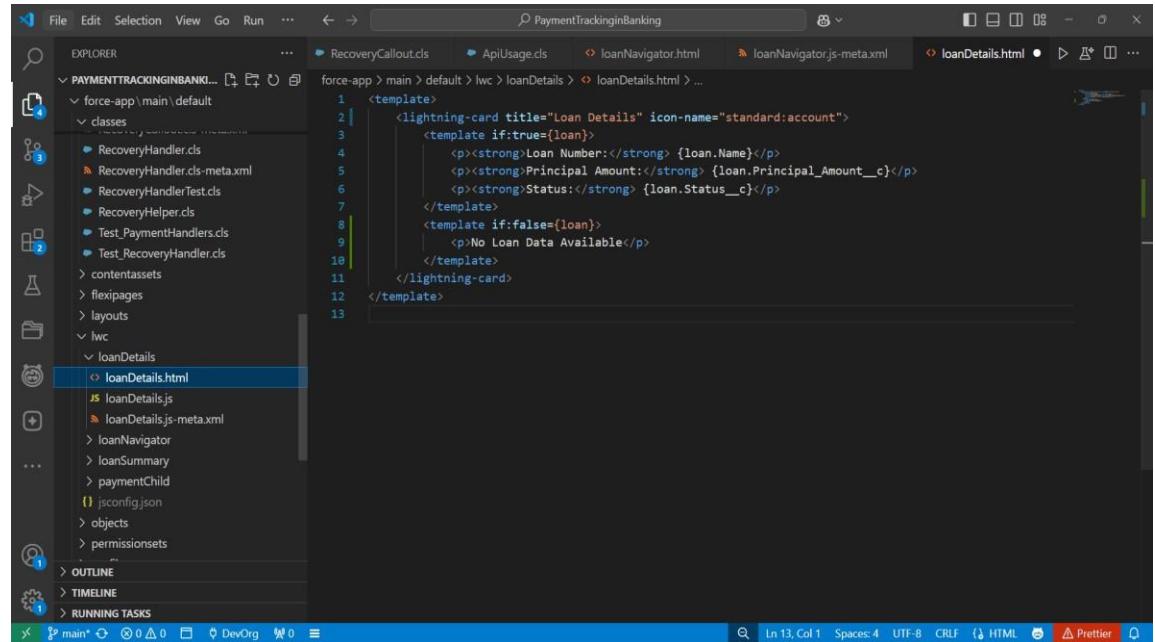
Utility Bar

- Switch between loan & payment records.

The screenshot shows the 'App Settings' page in the Lightning App Builder. The left sidebar has 'Navigation Items' selected. The main area is titled 'Navigation Items' and contains a section titled 'Available Items' with a list of items and a search bar. The 'Selected Items' section on the right lists items with icons: Dashboards, Reports, Loans, Payments, and Recoveries. Arrows on the right side of the 'Selected Items' list allow for reordering.

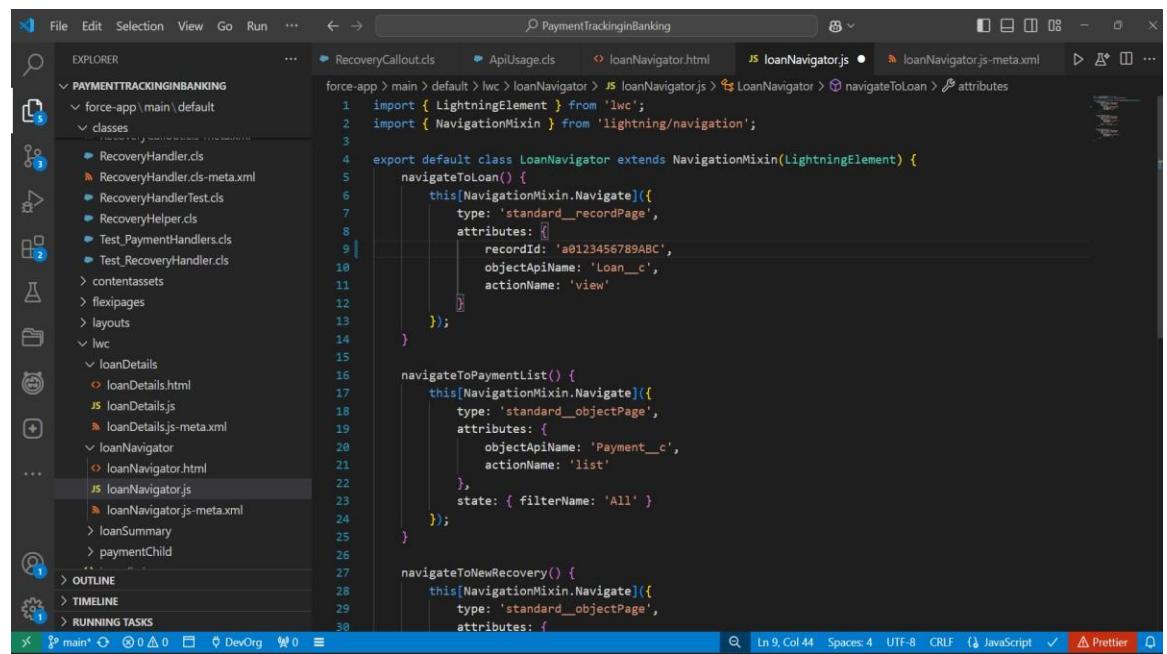
LWC (Lightning Web Components)

- 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 file "loanDetails.html". The code is an LWC template:

```
<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 the file "loanNavigator.js". The code is a JavaScript class:

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

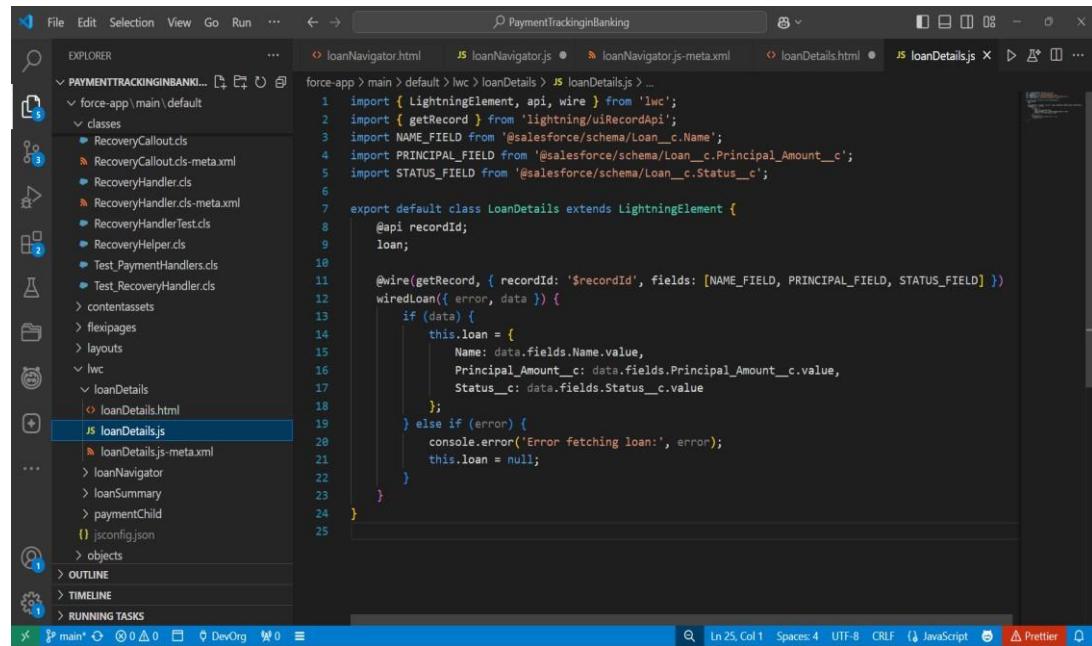
export default class LoanNavigator extends NavigationMixin(LightningElement) {
  navigateToLoan() {
    this[NavigationMixin.Navigate]({
      type: 'standard__recordPage',
      attributes: {
        recordId: 'a0123456789ABC',
        objectApiName: 'Loan__c',
        actionName: 'view'
      }
  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

  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: 'Recovery__c',
        actionName: 'new'
      }
    });
  }
}
```

Wire Adapters

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



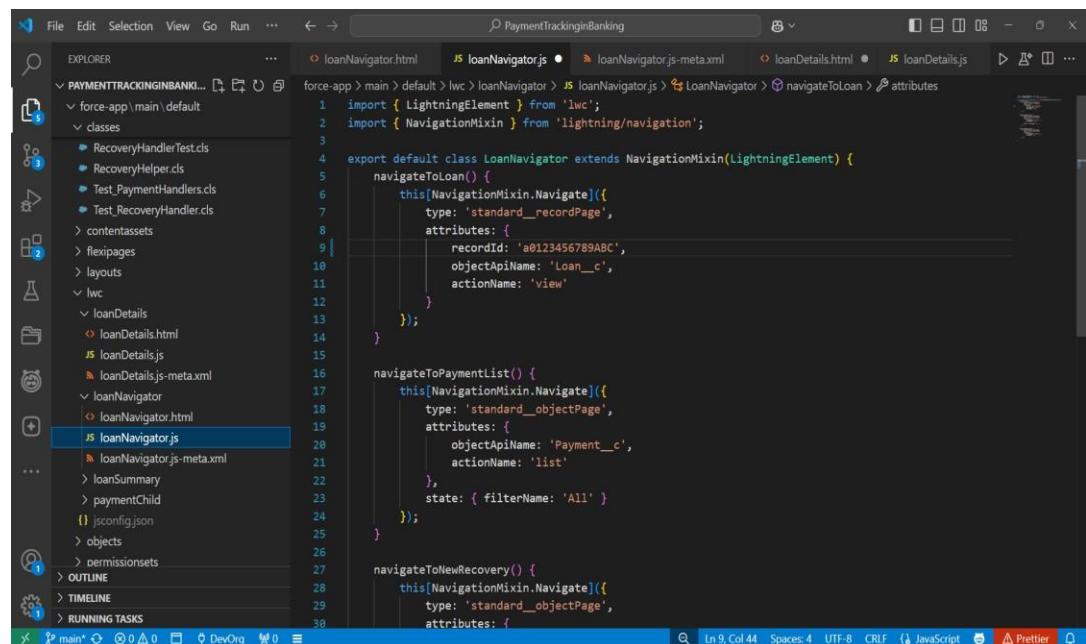
```
import { LightningElement, api, wire } from 'lwc';
import { getRecord } from 'lightning/uiRecordApi';
import NAME_FIELD from '@salesforce/schema/Loan__c.Name';
import PRINCIPAL_FIELD from '@salesforce/schema/Loan__c.Principal_Amount__c';
import STATUS_FIELD from '@salesforce/schema/Loan__c.Status__c';

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:

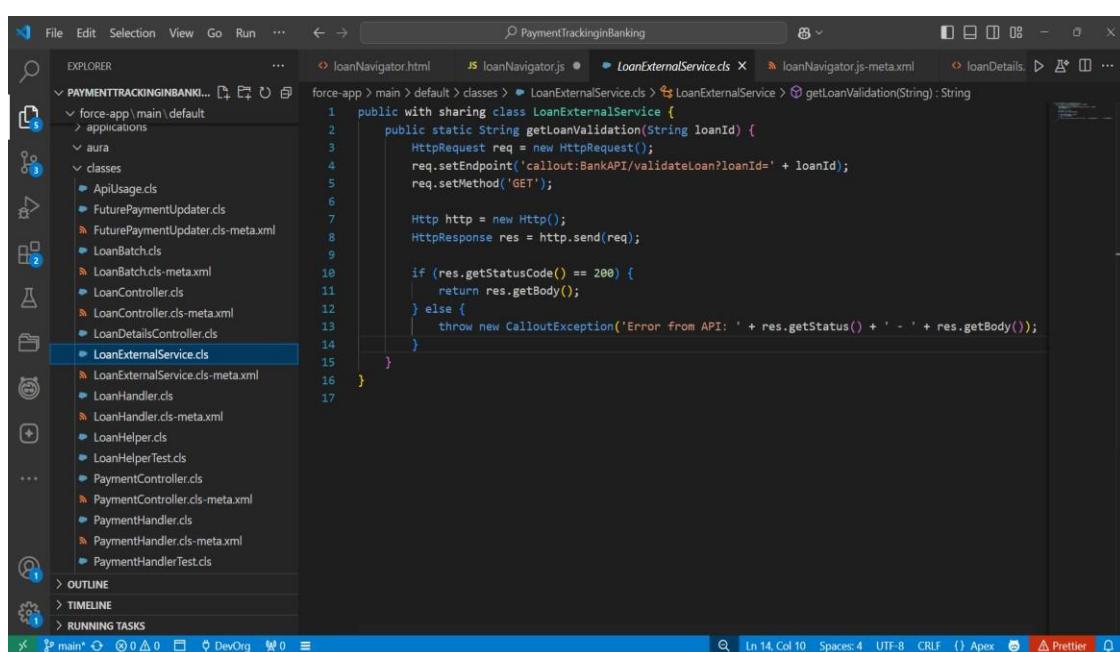
Operation Name ↑	Description	Input parameters	Output parameters
getValidateLoan	Validate Loan	loanId	responseCode, 200, default
postMakePayment	Make a Payment	loanId, amount	responseCode, 200, default

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());}}}
```

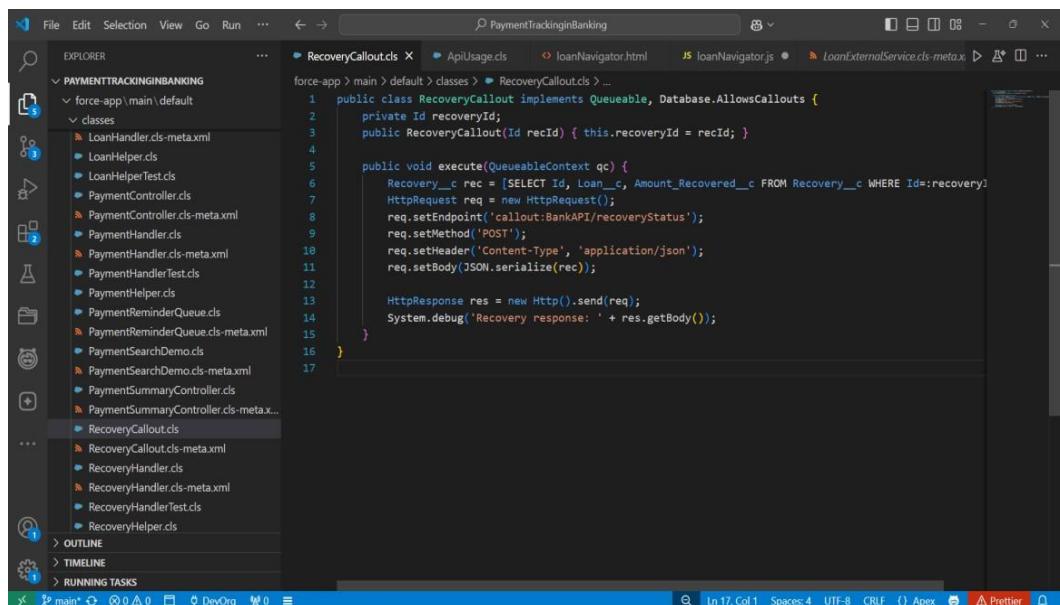


Callouts

- Used when Salesforce sends data to external services.

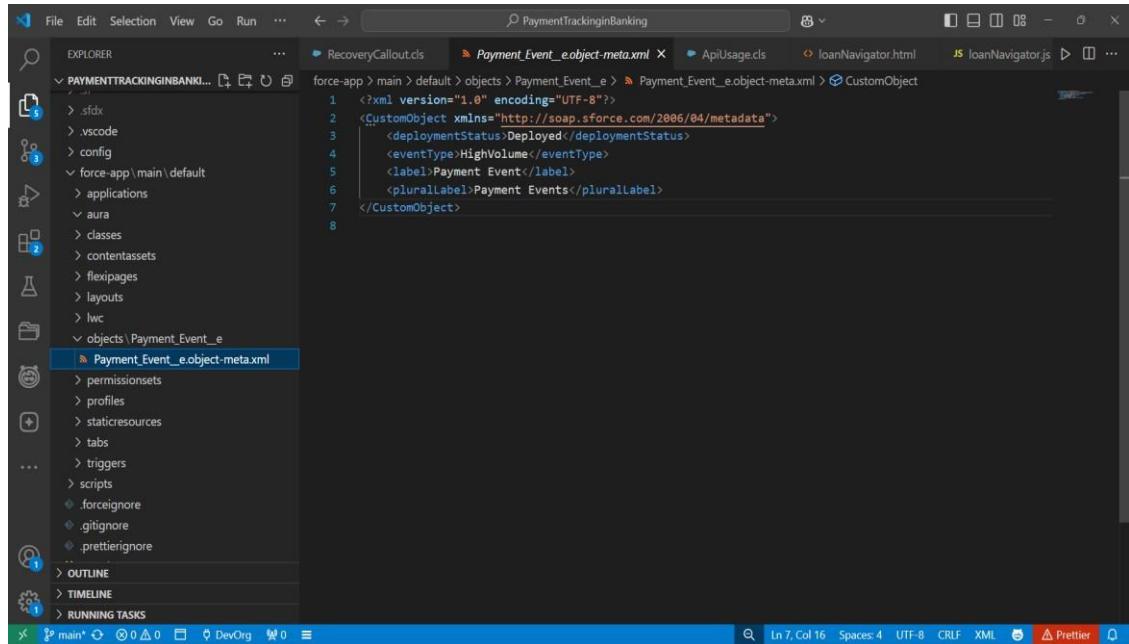
RecoveryCallout.cls

```
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).



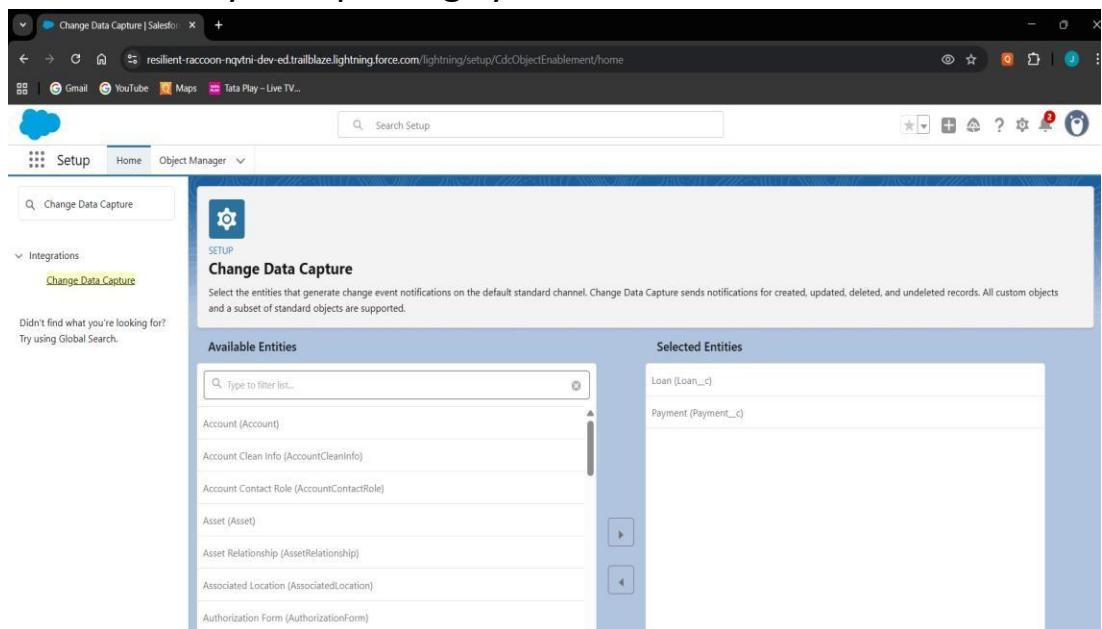
The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows the project structure under "PAYMENTTRACKINGINBANKING".
- Editor:** Displays the XML file "Payment_Event_e_object-meta.xml".
- Bottom Status Bar:** Shows file information like "In 7, Col 16", "Spaces 4", "UTF-8", "CRLF", "XML", and "Prettier".

```
<?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.

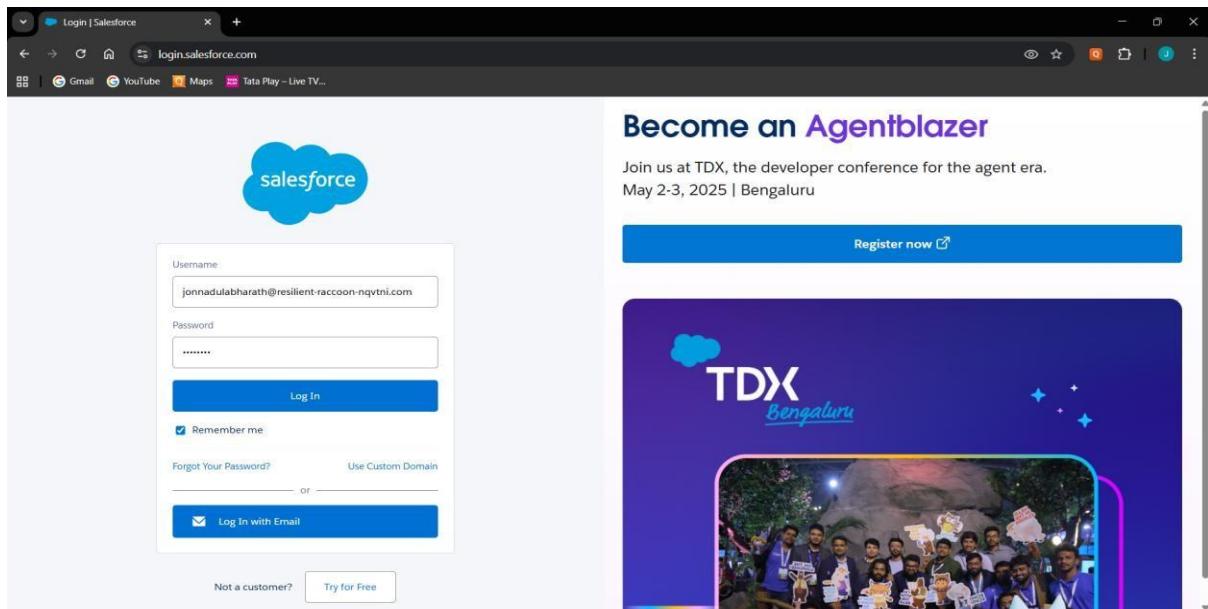


The screenshot shows the Salesforce Setup page for Change Data Capture:

- Page Header:** "Change Data Capture | Salesforce".
- Left Navigation:** "Setup", "Home", "Object Manager".
- Section:** "Integrations" - "Change Data Capture".
- Content:** "Change Data Capture" setup page. It says: "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." It lists "Available Entities" and "Selected Entities".
- Available Entities:** Account, Account Clean Info, Account Contact Role, Asset, Asset Relationship, Associated Location, Authorization Form.
- Selected Entities:** Loan (Loan__c), Payment (Payment__c).

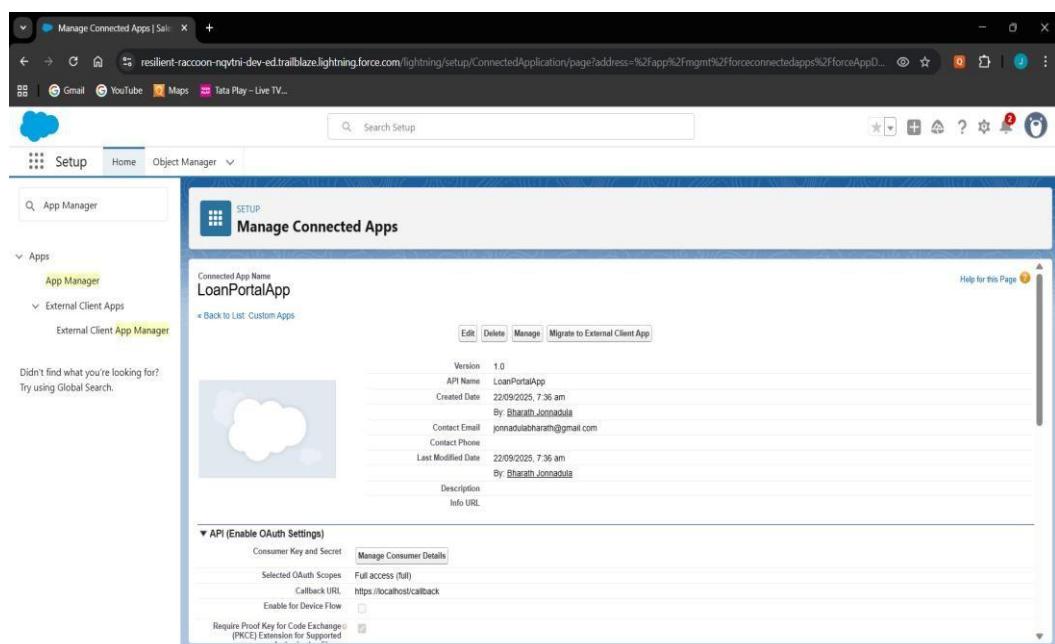
Salesforce Connect

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



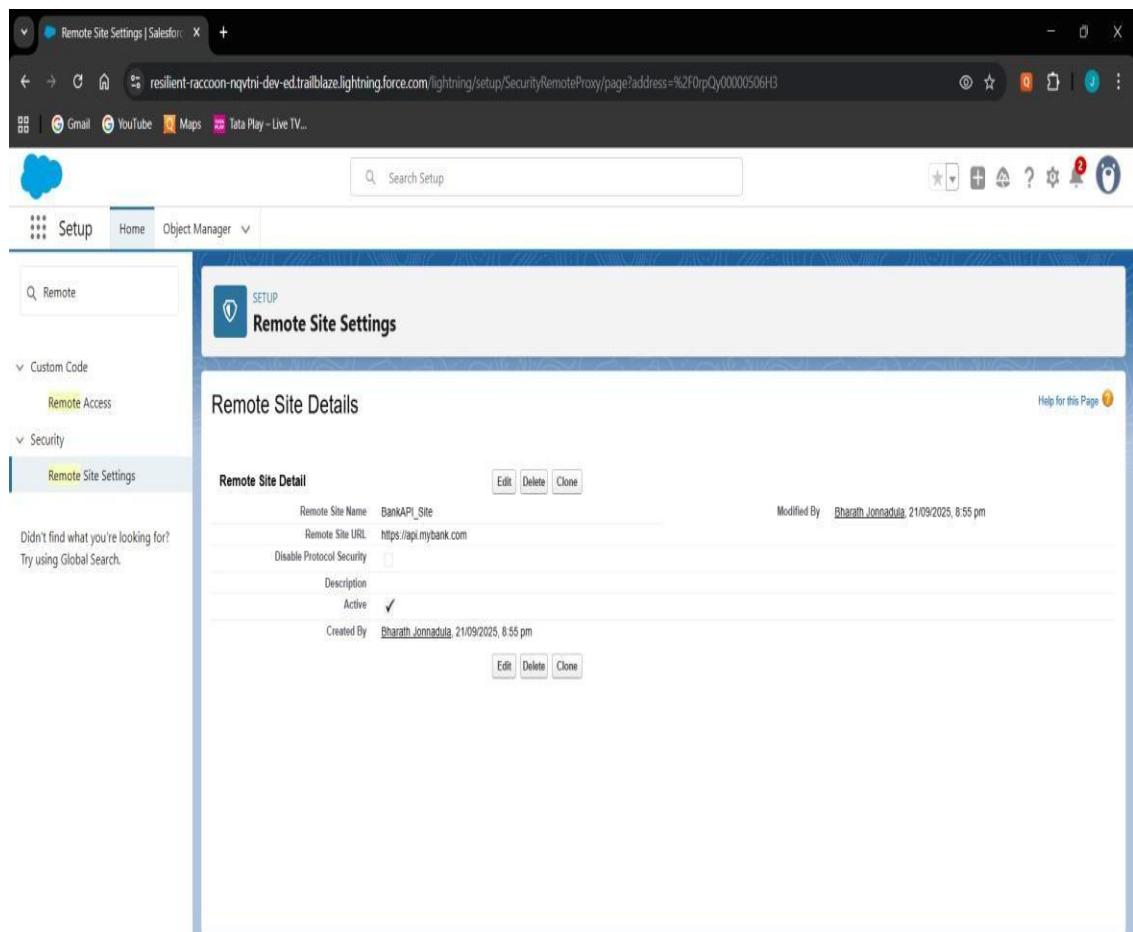
OAuth & Authentication

- 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". The "Customer Email" column contains email addresses such as "pardhapraneethjonnadula@gmail.com", "ravindrababu.jonnadula@gmail.com", and "jonnadulabharath@gmail.com". The "Loan: Name" column lists "Loan002", "Loan001", "Loan007", "Loan003", and "Loan005". The "Payment: Name" column lists "LN-25-0002", "LN-25-0001", "LN-25-0007", "LN-25-0003", and "LN-25-0005". The "Status" column lists "Active" for all entries.

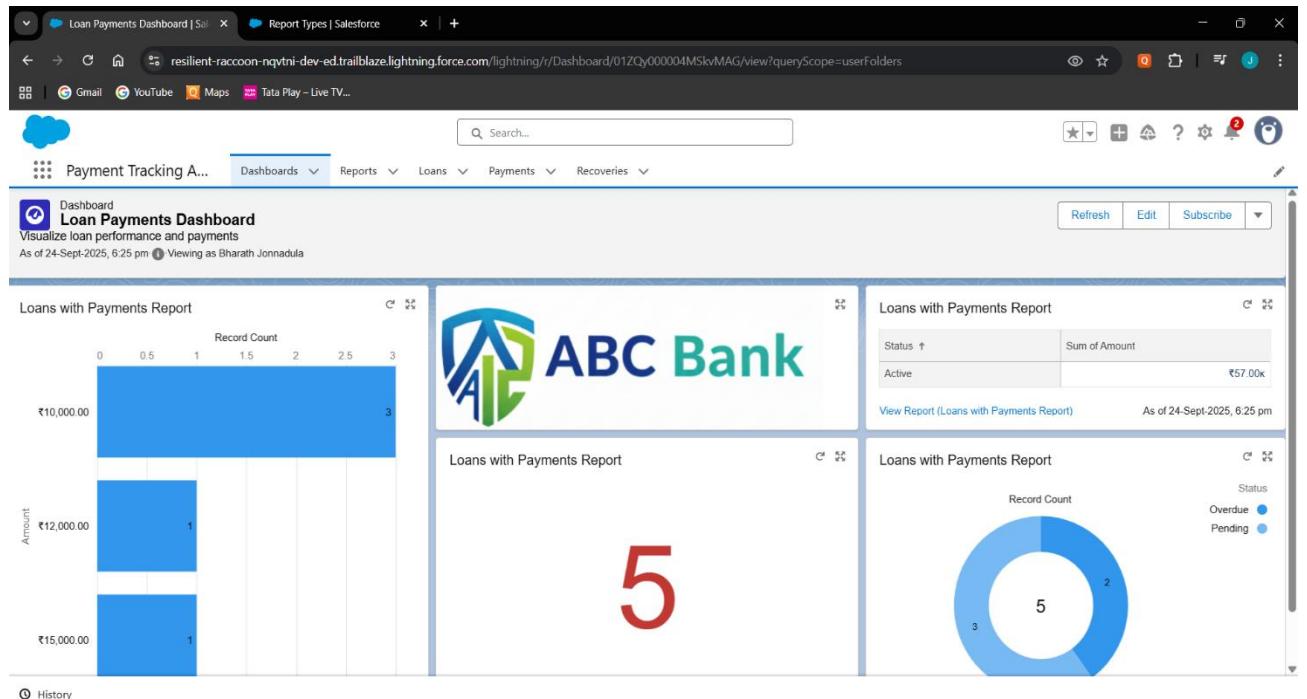
Report Types

- 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 report is titled "Loans with Payments". The left sidebar shows "Feature Settings" and "Analytics" sections, with "Report Types" selected. The main area displays the "Fields" layout for the report. It includes two sections: "Loans" and "Payments". The "Loans" section contains fields like "Loan ID", "Owner", "Name", "Record Type", "Created Date", "Created By", "Last Modified Date", "Last Modified By", "Customer", "Customer Code", "Customer Email", "Interest Rate", and "Last Modified By". The "Payments" section contains fields like "Payment ID", "Name", "Created Date", "Created By", "Last Modified Date", "Last Activity Date", and "Payment Date". The "Save" button is visible at the bottom right of the configuration screen.

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 under the "Profiles" tab. A new profile named "Manager Profile" is being created:

- Profile Edit:** Name is set to "Manager Profile", User License is "Salesforce", and Description is empty. The "Custom Profile" checkbox is checked.
- Custom App Settings:** This 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__LightningSales), Sales (standard__Sales), and Sales Cloud Mobile (standard__SalesCloudMobile). The "Visible" column has checkboxes for most items, while the "Default" column has radio buttons.

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 navigation includes 'Setup' (selected), 'Home', and 'Object Manager'. Under 'Users', 'Roles' is selected. The main content area is titled 'Creating the Role Hierarchy' with a sub-section 'Your Organization's Role Hierarchy'. It displays a hierarchical tree of roles under 'ABC Bank Payment Tracking':

- Branch Manager
 - Loan Officer
 - Recovery Agent
 - Customer Support, International
 - Customer Support, North America
 - Installation & Repair Services
 - User
 - VP, International Sales
 - VP, Marketing

Each node has 'Edit | Del | Assign' buttons. A 'Help for this Page' link is at the top right.

Users

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

The screenshot shows the Salesforce Setup Users page. The left sidebar navigation includes 'Setup' (selected), 'Home', and 'Object Manager'. Under 'Users', 'Users' is selected. The main content area is titled 'All Users' with a sub-section 'On this page you can create, view, and manage users.' It shows a list of users with columns: Action, Full Name, Alias, Username, Role, Active, and Profile. A 'View' dropdown is set to 'All Users'. Buttons for 'New User', 'Reset Password(s)', and 'Add Multiple Users' are at the top. A navigation bar at the bottom includes letters A-Z and a 'All' button. The user list includes:

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.0000y00000ycos5map.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

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' table includes fields for Description (Payment Team Access), License (not specified), Session Activation Required (unchecked), and Permission Set Groups Added To (0). The API Name is 'Payment_Team_Access' and the Namespace Prefix is 'Payment'. The 'Created By' and 'Last Modified By' fields show 'Bharath Jonnadaula' with their respective dates and times. The 'Apps' section lists several settings: Assigned Apps, Assigned Connected Apps, Object Settings, App Permissions, and Apex Class Access.

OWD & Sharing Rules

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

The screenshot shows the Salesforce Setup interface under the 'Sharing Settings' section. A table of sharing rules is displayed, listing objects and their sharing levels. Most objects are set to 'Private', except for 'Rebate Payout Snapshot', 'Scorecard', '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', which are set to 'Controlled by Parent'. A note in the sidebar suggests using Global Search.

Object	Sharing Level	Controlled by Parent
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	Controlled by Parent
Payment	Controlled by Parent	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**

Organization-Wide Defaults

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"/>

Field Level Security

- Restricted interest rate to view only for manager.

Users

Force.com - Free User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Gold Partner User	<input type="checkbox"/>	<input type="checkbox"/>
Identity User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Loan Officer Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Manager Profile	<input type="checkbox"/>	<input type="checkbox"/>
Marketing User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Minimum Access - API Only Integrations	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Minimum Access - Salesforce	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Partner App Subscription User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Partner Community Login User	<input type="checkbox"/>	<input type="checkbox"/>
Partner Community User	<input type="checkbox"/>	<input type="checkbox"/>
Read Only	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Recovery Agent Profile	<input type="checkbox"/>	<input type="checkbox"/>
Salesforce API Only System Integrations	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Silver Partner User	<input type="checkbox"/>	<input type="checkbox"/>
Solution Manager	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Standard Platform User	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Session settings

- Set session timeout for 30 minutes.

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

- Page Header:** Loan Payments Dashboard | Sales, Session Settings | Salesforce
- Search Bar:** Search Setup
- Left Navigation:** Setup, Home, Object Manager, Session Management, Session Settings (selected).
- Content Area:**
 - Session Settings:** Set the session security and session expiration timeout for your organization.
 - Session Timeout:** Timeout Value: 30 minutes (selected), Disable session timeout warning popup (unchecked), Force logout on session timeout (checked).
 - Session Settings:** Lock sessions to the IP address from which they originated (unchecked), Lock sessions to the domain in which they were first used (checked), Terminate all of a user's sessions when an admin resets that user's password (unchecked), Force reload after Login-As-User (checked), Require HttpOnly attribute (unchecked), Use POST requests for cross-domain sessions (unchecked), Enforce login IP ranges on every request (unchecked), When embedding a Lightning application in a third-party site, use a session token instead of a session cookie (unchecked).
 - Extended use of IE11 with Lightning Experience:** **EXTENDED USE OF IE11 WITH LIGHTNING EXPERIENCE HAS NOW ENDED** 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 with the following details:

- Page Header:** Loan Payments Dashboard | Sales, Profiles | Salesforce
- Search Bar:** Search Setup
- Left Navigation:** Setup, Home, Object Manager, Users, Profiles (selected).
- Content Area:**
 - Manager Profile:** Profile users have the permissions and page layouts listed below. Administrators can change a user's profile by editing that user's personal information.
 - Login IP Ranges:** Action: IP Start Address (49.37.153.221), IP End Address (49.37.153.221), Description (Bharath.Jonnadula, 15/09/2025, 7:36 pm), Modified By (Bharath.Jonnadula, 22/09/2025, 11:07 am).
 - Page Layouts:** Standard Object Layouts: Global (Global Layout [View Assignment]), Email Application (Not Assigned [View Assignment]), Home Page Layout (DE Default [View Assignment]), Access (Access Layout [View Assignment]). Individual (Individual Layout [View Assignment]), Invoice (Invoice Layout [View Assignment]), Invoice Line (Invoice Line Layout [View Assignment]), Lead (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: 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 Help \[?\]](#)

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

Description: Bharath_Jonnadula, 15/09/2025, 7:34 pm

Created By: Bharath_Jonnadula

Modified By: Bharath_Jonnadula, 22/09/2025, 11:07 am

[Help for this Page \[?\]](#)

Page Layouts

Standard Object Layouts

Object	Type	Layout	Assignment
Global	Global	Global Layout	[View Assignment]
Email Application	Not Assigned	[View Assignment]	
Home Page Layout	DE Default	[View Assignment]	
Access	Access Layout	[View Assignment]	

Individual Layout: Individual Layout [View Assignment]

Invoice Layout: Invoice Layout [View Assignment]

Invoice Line Layout: Invoice Line Layout [View Assignment]

Lead Layout: Lead Layout [View Assignment]

Created By: Bharath_Jonnadula, 15/09/2025, 7:34 pm

Modified By: Bharath_Jonnadula, 22/09/2025, 11:07 am

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: 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 Help \[?\]](#)

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

Description: Bharath_Jonnadula, 15/09/2025, 7:35 pm

Created By: Bharath_Jonnadula

Modified By: Bharath_Jonnadula, 22/09/2025, 11:07 am

[Help for this Page \[?\]](#)

Page Layouts

Standard Object Layouts

Object	Type	Layout	Assignment
Global	Global	Global Layout	[View Assignment]
Email Application	Not Assigned	[View Assignment]	
Home Page Layout	DE Default	[View Assignment]	
Access	Access Layout	[View Assignment]	

Individual Layout: Individual Layout [View Assignment]

Invoice Layout: Invoice Layout [View Assignment]

Invoice Line Layout: Invoice Line Layout [View Assignment]

Lead Layout: Lead Layout [View Assignment]

Created By: Bharath_Jonnadula, 15/09/2025, 7:35 pm

Modified By: Bharath_Jonnadula, 22/09/2025, 11:07 am

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' with a shield icon. Below it, a sub-header says 'View Setup Audit Trail'. A message indicates that the last 20 entries for the organization are listed. A link to download the audit trail as an Excel (.csv) file is provided. The main content is a table titled 'View Setup Audit Trail' with columns: Date, User, Source Namespace Prefix, Action, Section, and Delegate User. The table lists 20 entries from September 2025, mostly related to user management actions like logging in, changing passwords, and creating users, performed by a user named 'y22cs071@nrjc.ac.in'.

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