Corporate-Loan-Management-System-CLMS-Salesforce

Name: K Nikhilesh Reddy

Email:nikhileshreddy.k@gmail.com

Phase 1: Problem Understanding & Industry Analysis

1. Problem Statement

In the financial services industry, managing **corporate loans** is often complex and inefficient due to fragmented systems, manual approval processes, and limited visibility into loan status and repayments. Banks and NBFCs struggle with:

- Delays in processing **loan applications** due to multi-level approvals
- Lack of centralized tracking for **loan documents, EMI schedules, and repayments**
- Difficulty in providing **real-time updates** to corporate clients
- Limited automation in **reminders and overdue payment follow-ups**
- Poor decision-making due to absence of **consolidated reports and dashboards**

The **Corporate Loan Management System (CLMS)**, built on Salesforce CRM, addresses these challenges by providing a **centralized, automated, and scalable B2B solution** that streamlines the **loan lifecycle from application to closure**, improves **stakeholder collaboration**, and enables **data-driven decision-making**.

---

2. Requirement Gathering

- Capture and manage **loan applications** from corporate clients
- Automate **multi-level loan approval workflows**
- Generate and monitor **EMI schedules** and repayment history
- Provide **real-time loan status tracking** for clients
- Offer **reports and dashboards** for management decisions
- Ensure **compliance** and secure handling of financial documents

---

3. Stakeholder Analysis

- **Corporate Client (Account/Contact)** → Company applying for the loan and its representatives (CFO, Finance Manager)
- **Loan Officer** → Reviews loan documents and initiates approvals
- **Credit Manager** → Performs creditworthiness checks
- **Branch Manager** → Final authority for high-value loans
- **Finance/Collections Team** → Manages EMI collections and overdue payments
- **System Administrator** → Manages users, security, workflows, and integrations

---

4. Business Process Mapping

End-to-End Loan Lifecycle:

1. Corporate client submits loan request → **Opportunity created** in Salesforce
2. Loan Officer verifies documents → stored in **Loan_Document__c**
3. Approval workflow → Loan Officer → Credit Manager → Branch Manager
4. Upon approval → **Apex Trigger generates EMI_Schedule__c**
5. Payments recorded in **Payment__c** → updates loan balance
6. Automated reminders sent for upcoming/overdue EMIs
7. Loan auto-closes once all EMIs are completed
8. Management tracks loan performance via **Reports & Dashboards**

---

5. Industry-Specific Use Case Analysis

- **Banking & NBFCs** → Handle large volumes of corporate loans with multiple approvals
- **SME Financing** → Manage working capital loans for small/medium businesses
- **Commercial Real Estate Lending** → Support high-value loans with complex repayment plans
- **Trade Finance** → Track short-term loans for import/export activities

---

6. AppExchange Exploration

- **DocuSign** → Digital signing of loan agreements
- **Conga Composer** → Auto-generate loan agreements and EMI schedules as PDFs
- **SMS/WhatsApp Integration (Twilio/ValueText)** → Automated EMI reminders
- **Einstein Analytics Apps** → AI-driven insights into loan repayment trends and default risks

Phase 2: Org Setup & Configuration

This document outlines the organizational setup and configuration for the *Corporate Loan Management System (CLMS)* implemented on Salesforce Developer Edition.

---

1. Salesforce Editions

The CLMS project was developed using *Salesforce Developer Edition*, selected because it provides:

- Free access to the Salesforce platform for learning and development.
- Standard objects such as Accounts, Contacts, and Reports.
- Ability to create custom objects like *Loan Application* and *Loan Documents*.
- Full access to *profiles, roles, permission sets, OWD, and sharing rules*.
- App development and automation features (flows, workflows, validation rules).

This makes Developer Edition ideal for building an end-to-end corporate loan management solution.

---

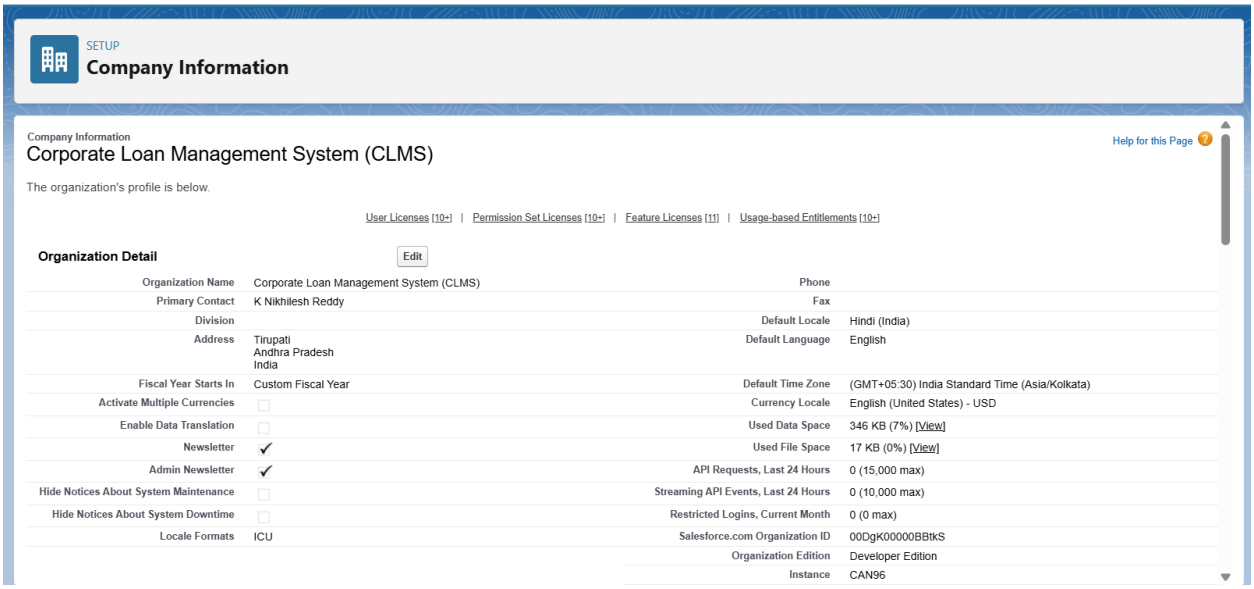
2. Company Profile Setup

The company profile defines organizational details in Salesforce:

- **Organization Name:** Corporate Loan Management System (CLMS)
- **Primary Contact:** K Nikhilesh Reddy

- **Address:** Tirupati, Andhra Pradesh, India
- **Default Locale:** Hindi (India)
- **Default Language:** English
- **Default Time Zone:** (GMT+05:30) India Standard Time (Asia/Kolkata)
- **Currency Locale:** English (United States) - USD
- **Organization Edition:** Developer Edition

This configuration aligns Salesforce with Indian financial and working standards.



---

3. Business Hours & Holidays

Business hours ensure loan-related activities occur within official times:

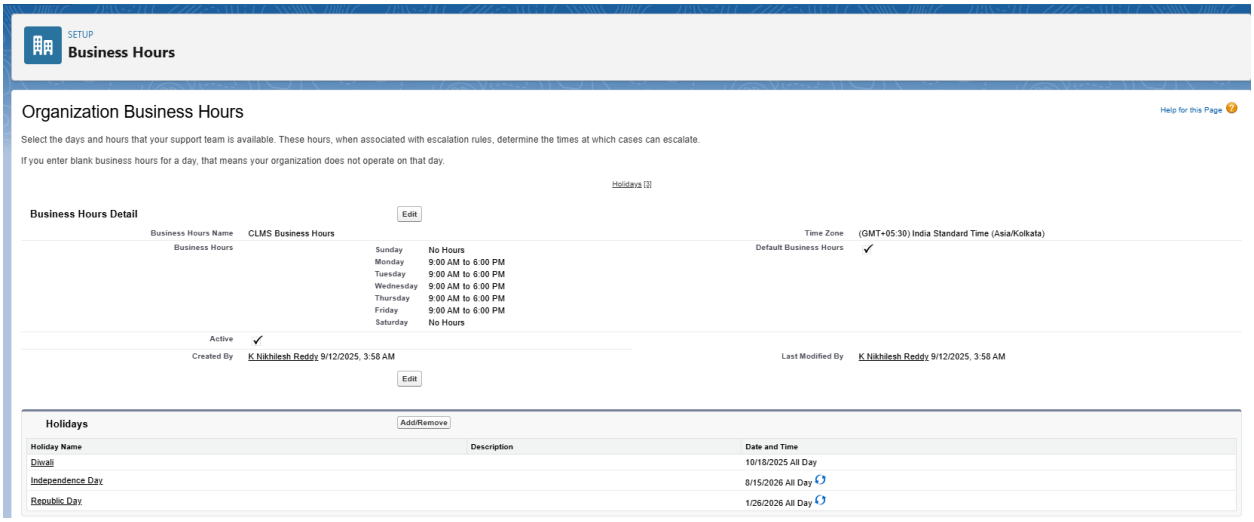
Business Hours:

- Monday – Friday: 9:00 AM to 6:00 PM
- Saturday – Sunday: No Hours

Holidays Configured:

- Diwali → 10/18/2025
- Independence Day → 08/15/2026

- Republic Day → 01/26/2026



This prevents scheduling on non-working days and ensures accurate turnaround times.

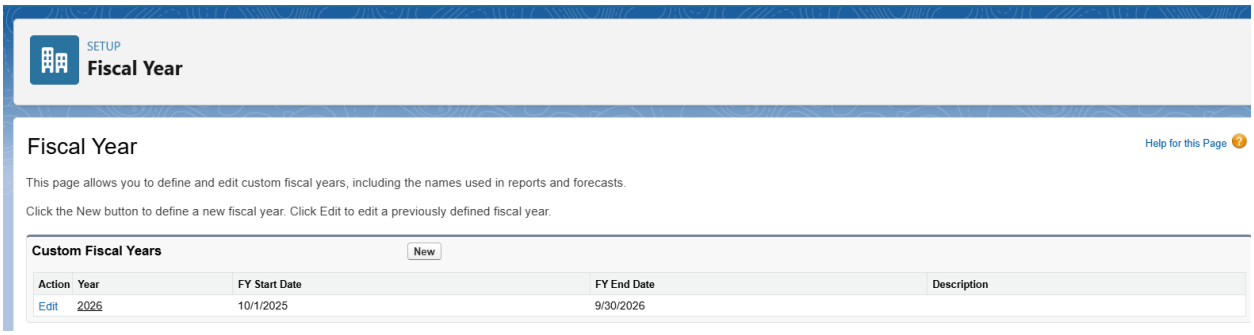
---

4. Fiscal Year Settings

The CLMS project uses a *custom fiscal year*:

- **Fiscal Year Type:** Custom Fiscal Year
- **FY 2026 Start Date:** 10/01/2025
- **FY 2026 End Date:** 09/30/2026

Supports accurate financial planning, loan disbursement cycles, and EMI tracking.



---

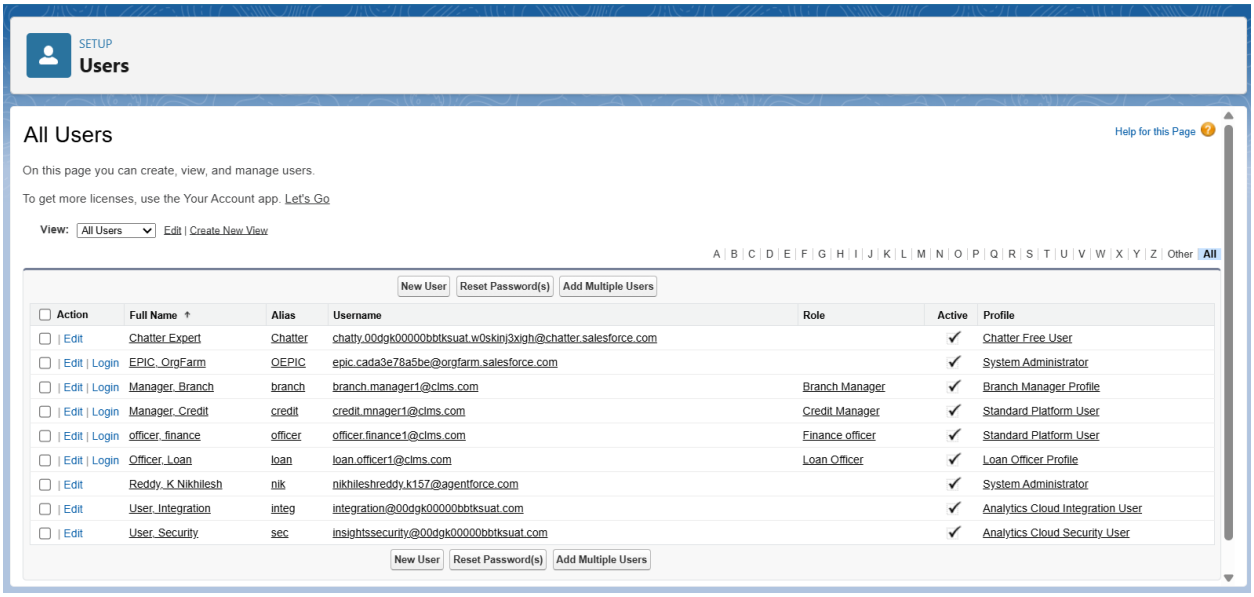
5. User Setup & Licenses

Users represent stakeholders in corporate loan processing:

Edit | Login	Manager, Branch	branch	branch.manager1@clms.com	Branch Manager	✓	Branch Manager Profile
Edit | Login	Manager, Credit	credit	credit.mnager1@clms.com	Credit Manager	✓	manager credit
Edit | Login	officer, finance	officer	officer.finance1@clms.com	Finance officer	✓	officer finance
Edit | Login	Officer, Loan	loan	loan.officer1@clms.com	Loan Officer	✓	Loan Officer Profile

User	Role in CLMS
Loan Officer	Creates and submits loan requests
Credit Manager	Reviews loan applications and decides approval
Finance Officer	Verifies approved loans and handles fund disbursement
Branch Manager	Oversees activities of Loan Officers, Credit Managers, and Finance Officers
CEO	Top-level oversight and access to all records

Each user was assigned a Salesforce License for standard platform access.

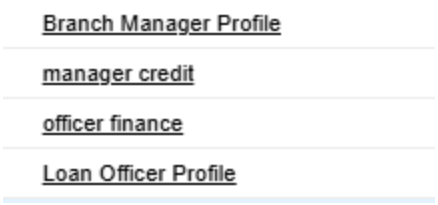


---

6. Profiles

Profiles define object-level and field-level permissions:

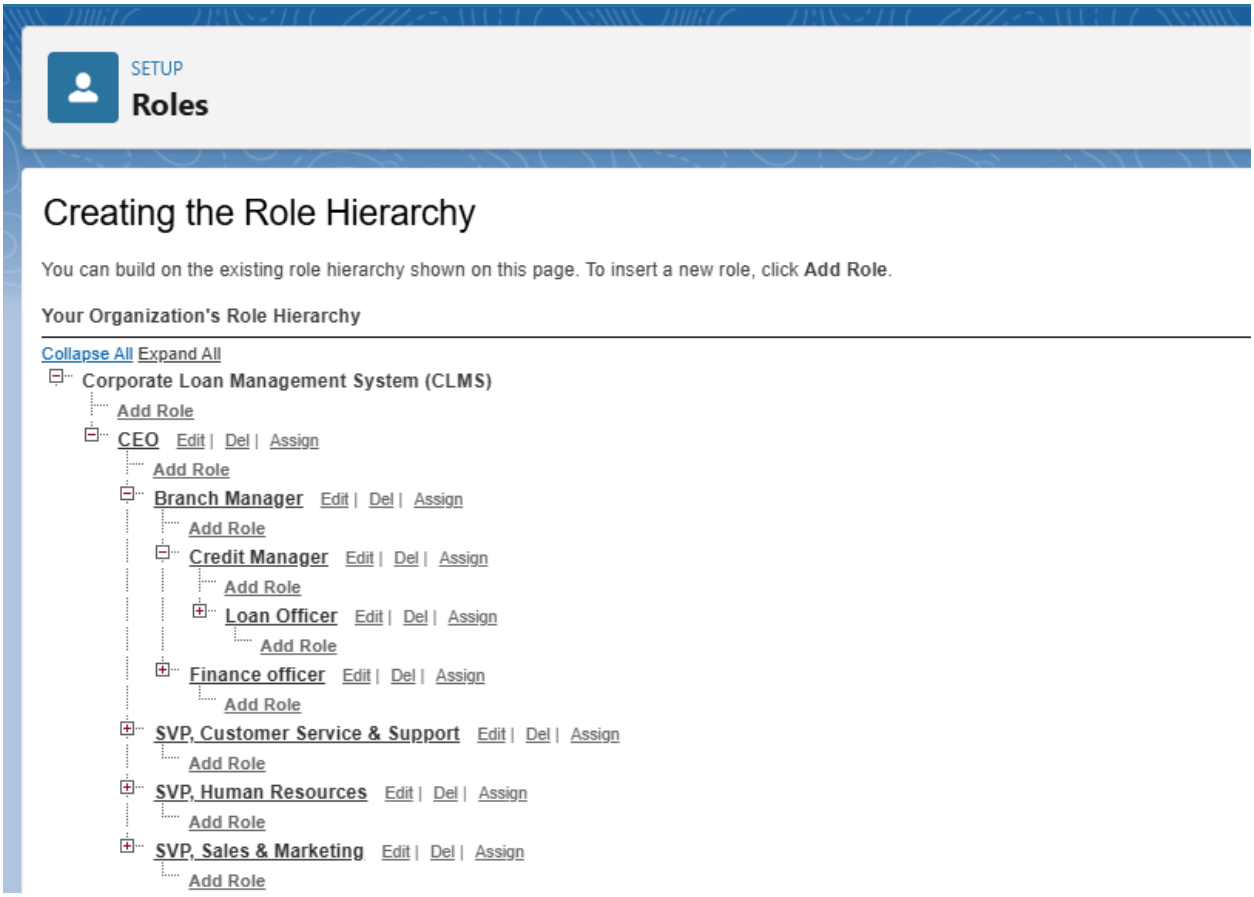
User	Profile	Access Description
Loan Officer	Standard User	Create and edit Loan Applications
Credit Manager	Standard User	Review and update Loan Applications
Finance Officer	Standard User	Manage financial records, disburse loans, update payments
Branch Manager	Standard User	Full visibility of subordinate roles and loan activities
CEO	System Admin	Complete control over the system



7. Roles

A role hierarchy reflects the organizational chain of command:

- *CEO:* Visibility into all records
- *Branch Manager:* Manages Credit and Finance Officers
- *Credit Manager:* Supervises Loan Officers and loan approvals
- *Loan Officer:* Creates loan requests
- *Finance Officer:* Handles fund release and financial reporting



---

8. Permission Sets

A custom permission set *"Loan Document Access"* allows users to upload, view, and download loan-related files.

- Assigned to *Credit Manager, Finance Officer, and Branch Manager*
- Ensures proper document handling while maintaining security

---

9. Organization-Wide Defaults (OWD)

Defines baseline data access levels:

- *Loan Applications:* Private → Only the record owner and managers can access
- *Accounts and Contacts:* Public Read-Only → Basic customer details visible to all
- *Loan Documents:* Controlled by Parent → Access follows related Loan Application

Guarantees confidentiality of customer and loan data.

---

10. Sharing Rules

Extend access beyond OWD:

- *Loan Applications:* Shared with Credit Manager and Finance Officer roles for processing and disbursing loans
- *Branch Manager:* Visibility to all records under their branch

Ensures collaboration while maintaining security.

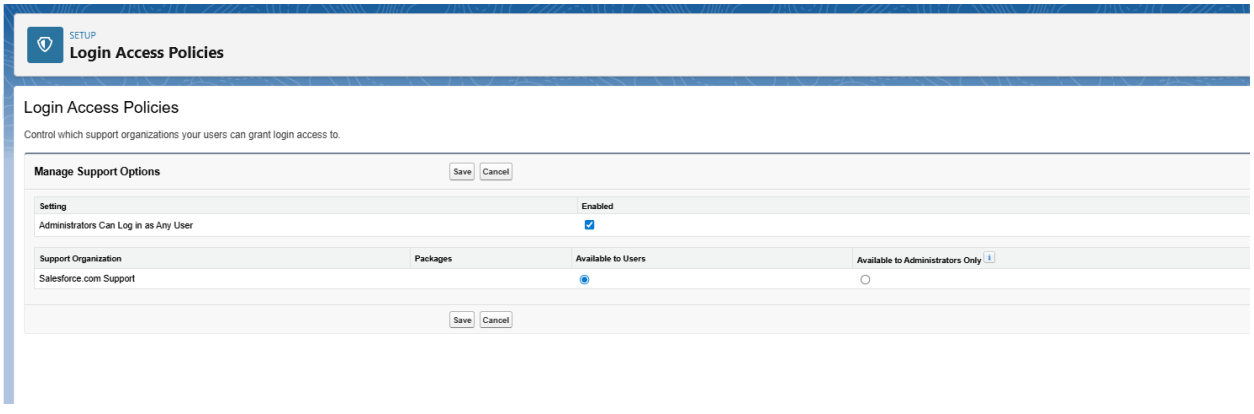
Approval History	Public Read/Write	Private	✓
Branch	Public Read/Write	Private	✓
EMI Schedule	Controlled by Parent	Controlled by Parent	
Loan Application	Private	Private	✓
Loan Document	Controlled by Parent	Controlled by Parent	
Payment	Public Read/Write	Private	✓

---

11. Login Access Policies

Configured to improve support and security:

- Enabled *Admin Login Access* for troubleshooting
- Configured *IP restrictions* to limit unauthorized logins



---

12. Developer Org Setup

Implementation used a *Salesforce Developer Org*:

- Free, fully functional Salesforce environment

- Support for custom objects, roles, workflows, and reports
- Flexibility to test and configure features before deployment

---

13. Sandbox Usage

- Developer Edition does *not support Sandboxes*
- Testing performed directly in the Developer Org
- Documented as a limitation but manageable due to academic scope

---

14. Deployment Basics

A custom Lightning App named *CLMS* has been created. This app centralizes access to the core business objects:

- *Loan Applications*
- *Loan Documents*
- *Accounts*
- *Contacts*
- *EMI Schedules*
- *Users*
- *Reports*

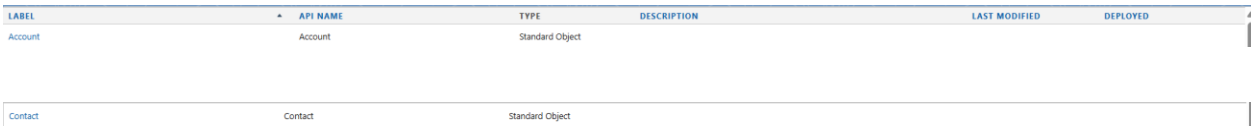
This setup provides a single interface for all stakeholders to access, manage, and track loan-related operations efficiently within Salesforce.

Phase 3: Data Modeling & Relationships – CLMS Project

1. Standard & Custom Objects

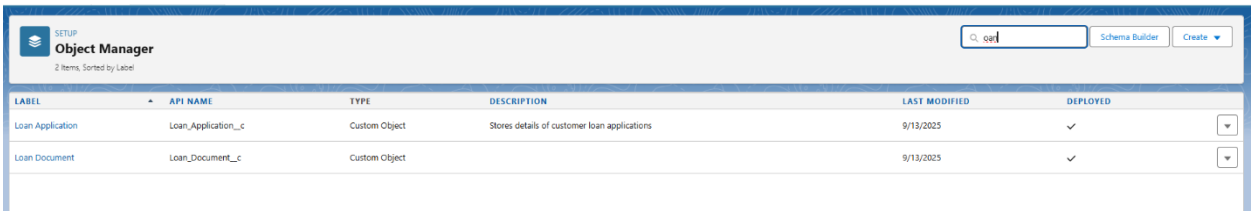
Standard Objects

- **Account** → Represents Borrowers (individual or corporate).
- **Contact** → Stores borrower contact details.



Custom Objects

- **Loan Application** → Loan request details (Loan Type, Amount, Tenure, Interest Rate, Status).
- **EMI Schedule** → Repayment installments (EMI Amount, Due Date, Status).
- **Loan Documents** → Loan-related documents (Proof of ID, Proof of Income, Collateral).
- **Approval History** → Loan approval/rejection trail (by Credit Manager and Finance Officer).

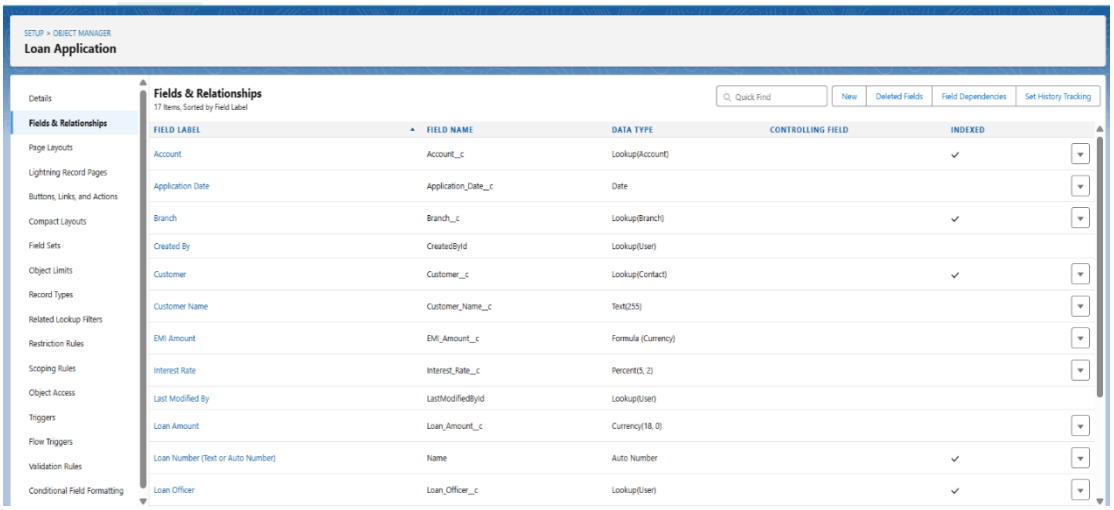


---

2. Fields

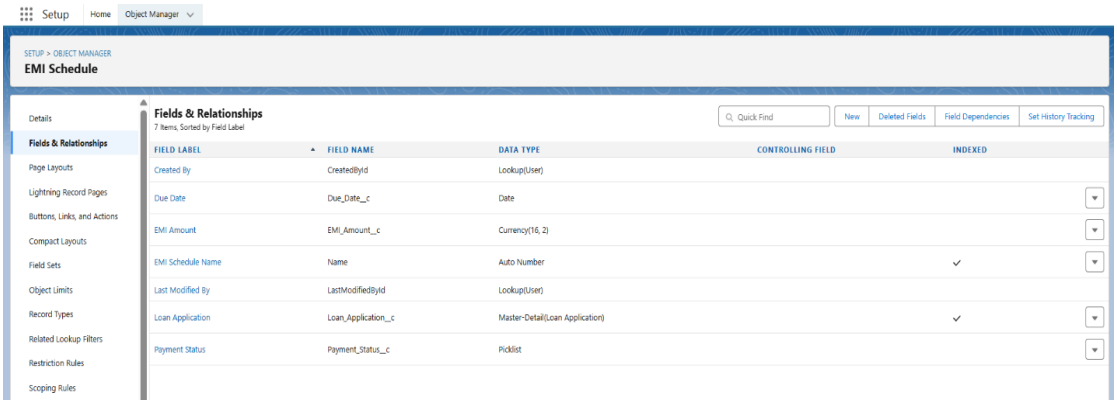
Loan Application Fields

- Loan Number (Auto Number)
- Loan Type
- Loan Amount
- Loan Tenure (Months)
- Interest Rate
- EMI Amount
- Status (Pending, Approved, Rejected)
- Application Date
- Branch
- Loan Officer

- Customer (Account Lookup)
- Customer Name
- 

EMI Schedule Fields

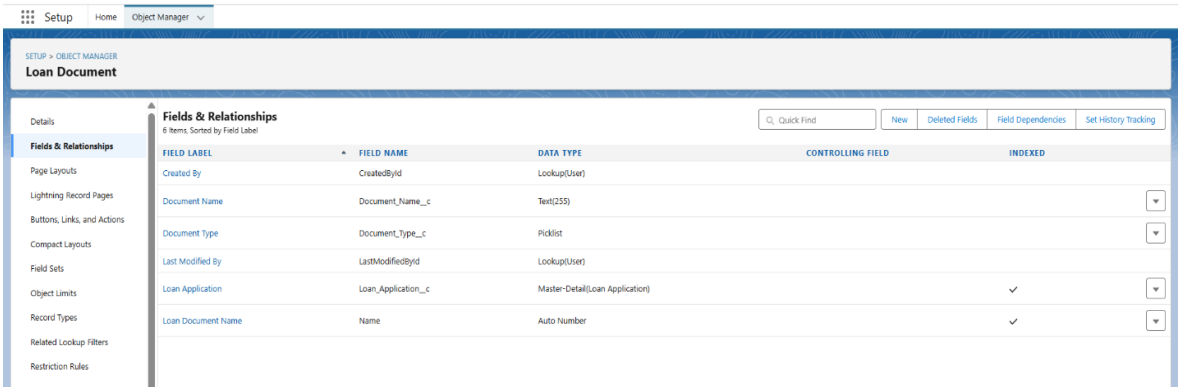
- EMI Amount
- Due Date
- Payment
  Status



Loan Documents Fields

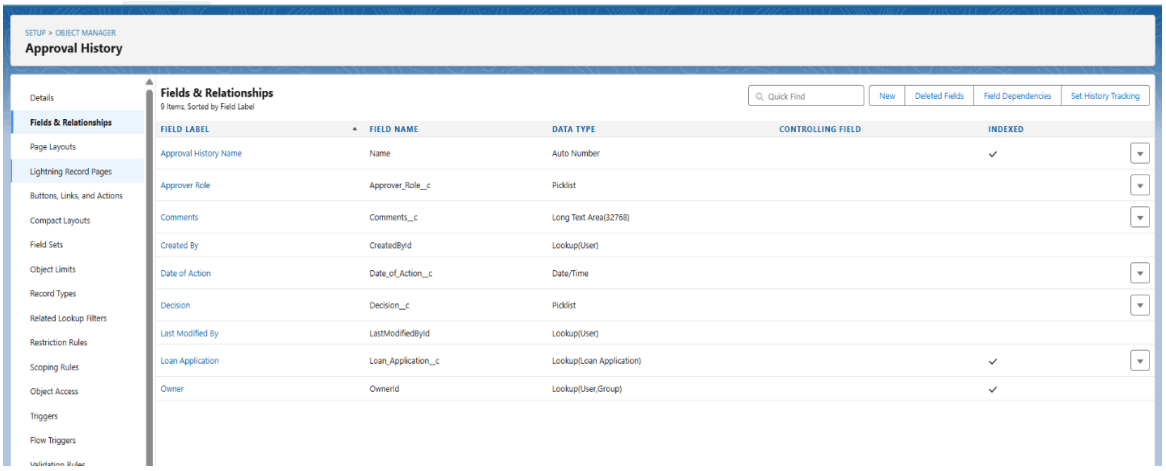
- Loan Document Name
- Document Type
- Document Name
- Loan Application (Lookup)

- Created By



Approval History Fields

- Approver Role (Picklist: Credit Manager, Finance Officer)
- Decision (Picklist: Approved, Rejected, Pending)
- Date of Action (Date/Time)
- Comments (Text)
- Loan Application (Lookup)

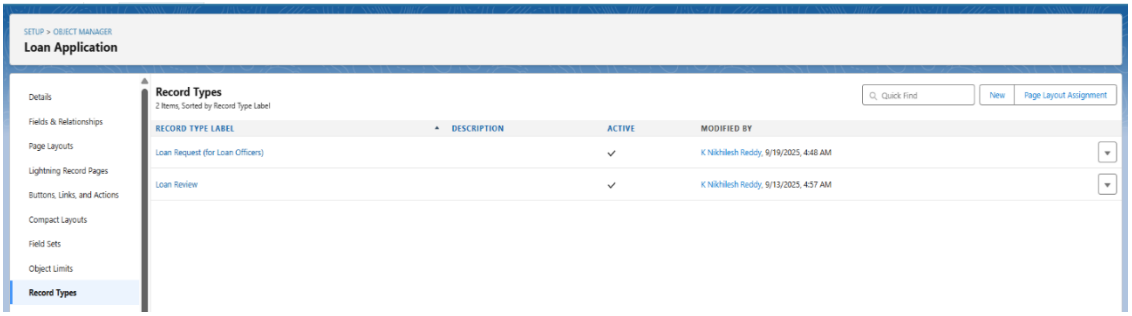


---

3. Record Types

- **Personal Loan**
- **Business Loan**

- **Home**
  Loan

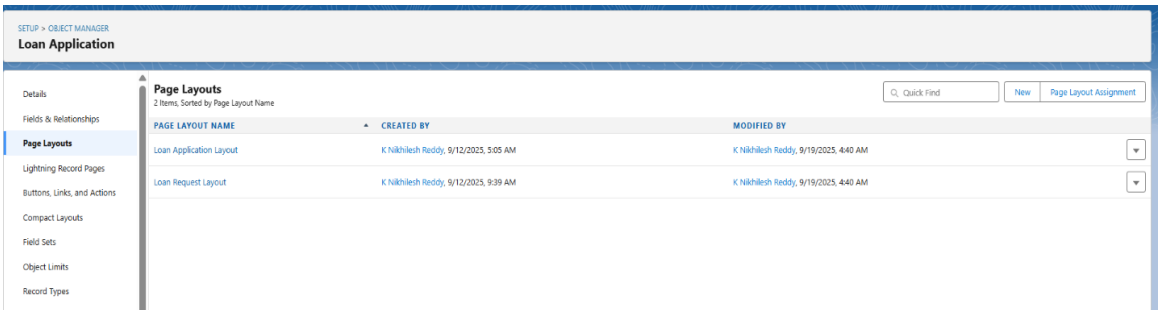


Each record type has its own page layout and picklist values.

---

4. Page Layouts

- **Loan Application Layout** → Loan details + related lists (EMI, Documents, Approval History).
- **EMI Schedule Layout** → EMI Amount, Due Date, Status.
- **Loan Documents Layout** → Document Type, Document Name, Loan Application.
- **Approval History Layout** → Approver Role, Decision, Date of Action, Comments.

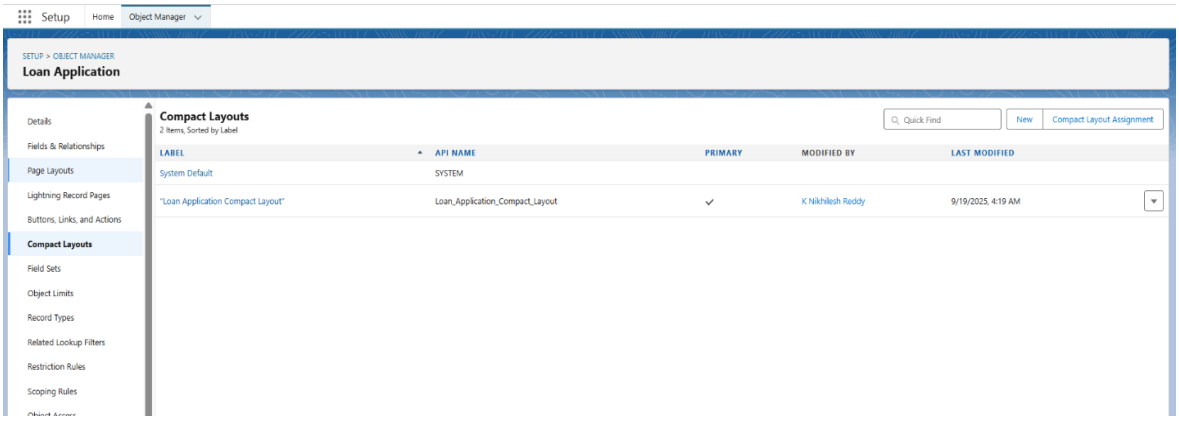


---

5. Compact Layouts

- **Loan Application Compact Layout** → Loan Number, Customer Name, Loan Amount, Loan Type, Status, Application Date.
- **EMI Schedule Compact Layout** → EMI Amount, Due Date, Status.
- **Loan Documents Compact Layout** → Loan Document Name, Document Type, Loan Application, Created By.

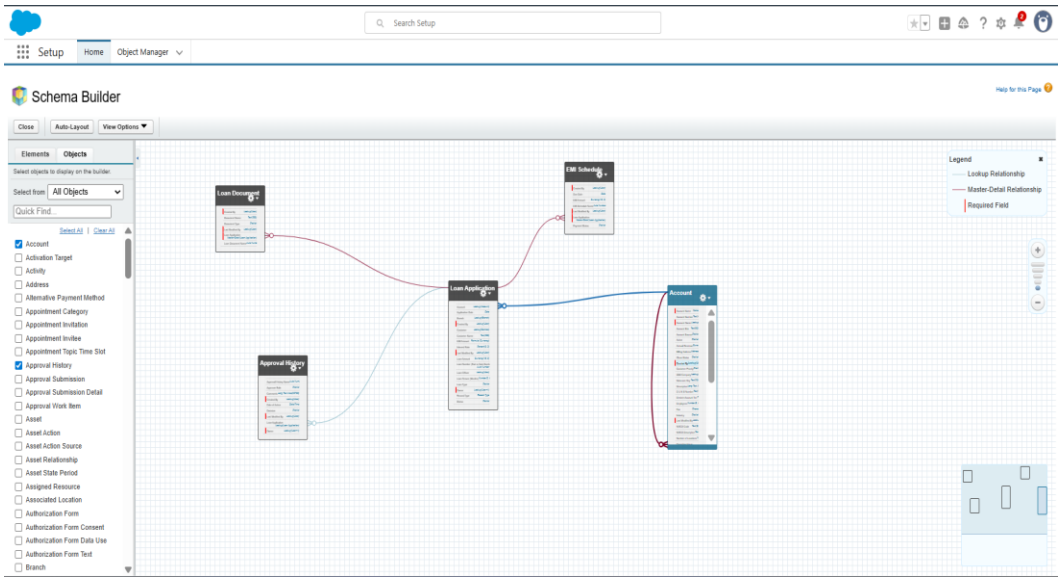
- **Approval History Compact Layout** → Approver Role, Decision, Date of Action, Loan Application.



---

6. Schema Builder

- Visual data model showing object relationships.
- Final schema:
  - Account → Loan Application (Lookup)
  - Loan Application → EMI Schedule (Master-Detail)
  - Loan Application → Loan Documents (Master-Detail)
  - Loan Application → Approval History (Lookup)



---

7. Lookup vs Master-Detail vs Hierarchical

- **Lookup Relationship** → Loose connection
  Example: Loan Application → Approval History
- **Master-Detail Relationship** → Strong parent-child
  Example: Loan Application → EMI Schedule, Loan Application → Loan Documents
- **Hierarchical Relationship** → Only for User object (Role Hierarchy)
  CEO → Branch Manager → Credit Manager → Loan Officer → Finance Officer

---

8. Junction Objects

- **Guarantor Mapping** → Junction object for many-to-many:
    - Borrower (Account) ↔ Loan Application.

---

9. External Objects

- **Credit Bureau Records** (external object via Salesforce Connect).
- Used to fetch credit scores or external financial history for loan evaluation.

Final Data Model

Account (Borrower) │

└── Loan Application │

── EMI Schedule (Master-Detail) │

── Loan Documents (Master-Detail)

└── Approval History (Lookup)

Phase 4: Process Automation (Admin)

In this phase, various Salesforce automation tools were configured and implemented in the **Corporate Loan Management System (CLMS)**.
The purpose of these automations is to ensure smooth processing of loan applications, enforce business rules, reduce manual effort, and provide real-time updates to different stakeholders involved in the loan lifecycle.

The tools used include:

- Validation Rules
- Workflow Rules
- Process Builder
- Approval Process
- Flow Builder (evaluated)
- Email Alerts
- Field Updates
- Tasks
- Custom Notifications

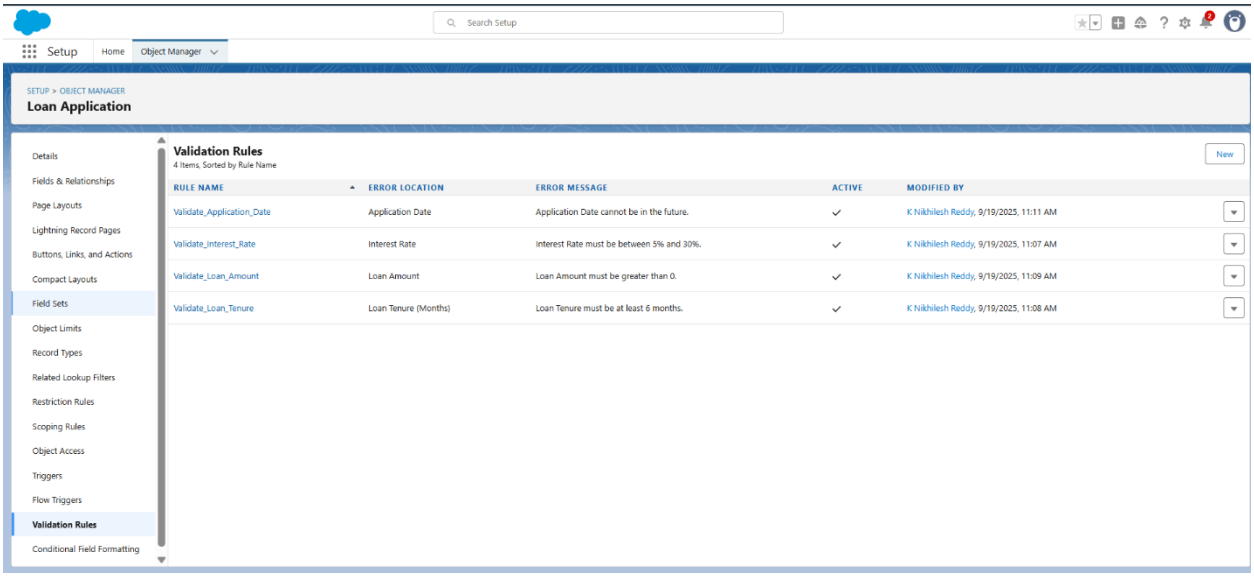
🔹 Validation Rules

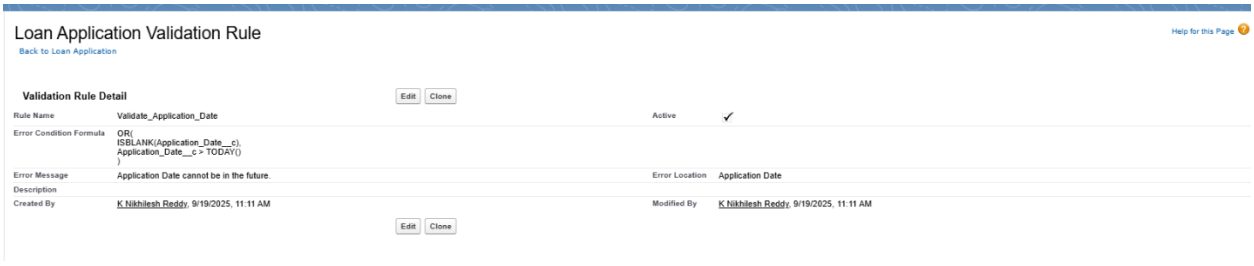
Validation Rules were created on the Loan Application object to ensure data accuracy and enforce corporate policies. These rules prevent users from saving invalid data.

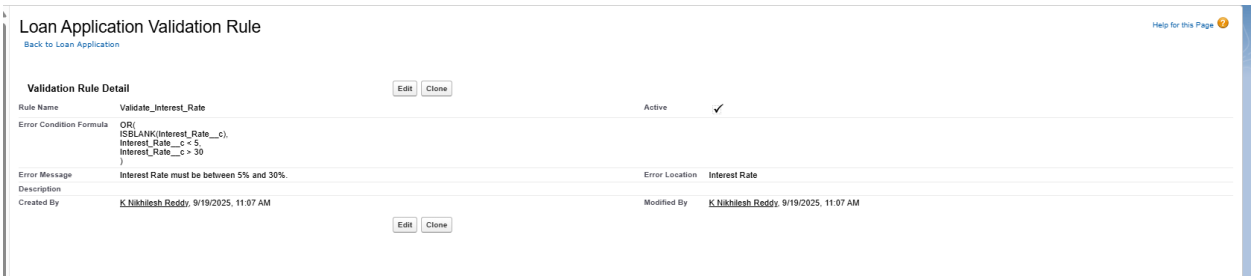
- **Loan Amount Validation** → Loan Amount must be greater than zero.
- **Loan Tenure Validation** → Tenure must be at least 6 months.
- **Interest Rate Validation** → Interest rate must be between 5% and 30%.
- **Application Date Validation** → Application date cannot be in the future.

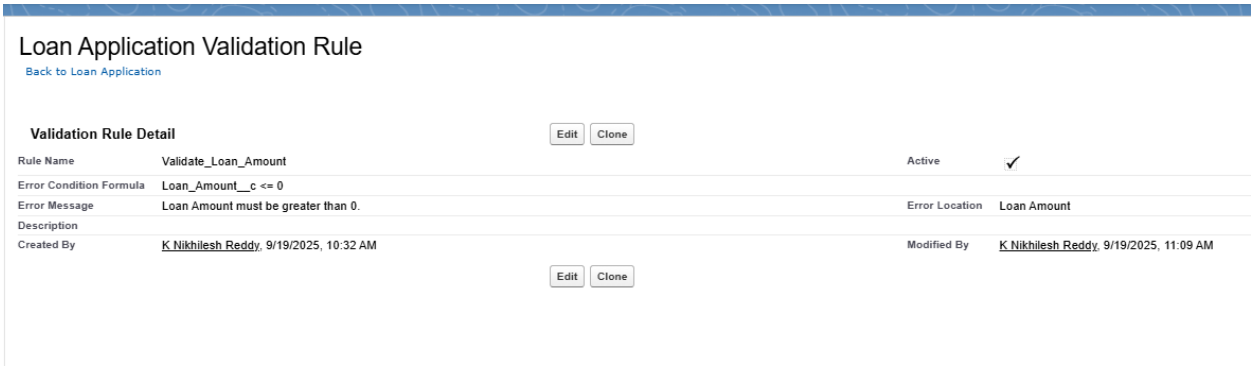
Outcome: Invalid or inconsistent data cannot be saved, which maintains high data integrity.

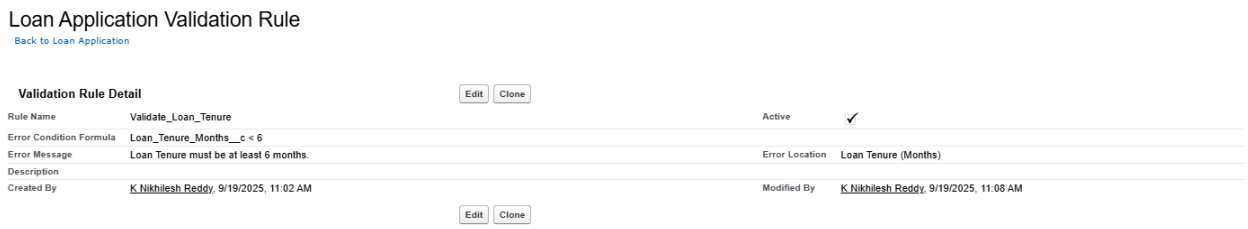
Screenshot:


Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Loan Application
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting
Validation Rules
4 Items, Sorted by Rule Name
New
RULE NAME
ERROR LOCATION
ERROR MESSAGE
ACTIVE
MODIFIED BY
Validate_Application_Date
Application Date
Application Date cannot be in the future.
K Nikhilesh Reddy, 9/19/2025, 11:11 AM
Validate_Interest_Rate
Interest Rate
Interest Rate must be between 5% and 30%.
K Nikhilesh Reddy, 9/19/2025, 11:07 AM
Validate_Loan_Amount
Loan Amount
Loan Amount must be greater than 0.
K Nikhilesh Reddy, 9/19/2025, 11:09 AM
Validate_Loan_Tenure
Loan Tenure (Months)
Loan Tenure must be at least 6 months.
K Nikhilesh Reddy, 9/19/2025, 11:08 AM


Loan Application Validation Rule
Help for this Page
Back to Loan Application
Validation Rule Detail
Edit
Clone
Rule Name
Validate_Application_Date
Active
Error Condition Formula
OR(
ISBLANK(Application_Date__c),
Application_Date__c > TODAY()
)
Error Message
Application Date cannot be in the future.
Error Location
Application Date
Description
Created By
K Nikhilesh Reddy, 9/19/2025, 11:11 AM
Modified By
K Nikhilesh Reddy, 9/19/2025, 11:11 AM
Edit
Clone


Loan Application Validation Rule
Help for this Page
Back to Loan Application
Validation Rule Detail
Edit
Clone
Rule Name
Validate_Interest_Rate
Active
Error Condition Formula
OR(
ISBLANK(Interest_Rate__c),
Interest_Rate__c < 5,
Interest_Rate__c > 30
)
Error Message
Interest Rate must be between 5% and 30%.
Error Location
Interest Rate
Description
Created By
K Nikhilesh Reddy, 9/19/2025, 11:07 AM
Modified By
K Nikhilesh Reddy, 9/19/2025, 11:07 AM
Edit
Clone


Loan Application Validation Rule
Back to Loan Application
Validation Rule Detail
Edit
Clone
Rule Name
Validate_Loan_Amount
Active
Error Condition Formula
Loan_Amount__c <= 0
Error Message
Loan Amount must be greater than 0.
Error Location
Loan Amount
Description
Created By
K Nikhilesh Reddy, 9/19/2025, 10:32 AM
Modified By
K Nikhilesh Reddy, 9/19/2025, 11:09 AM
Edit
Clone



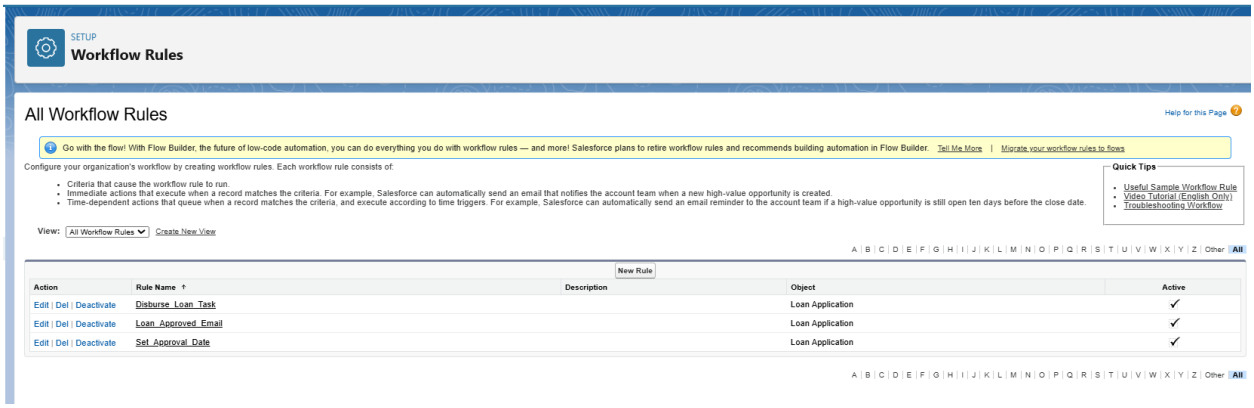
🔹 Workflow Rules

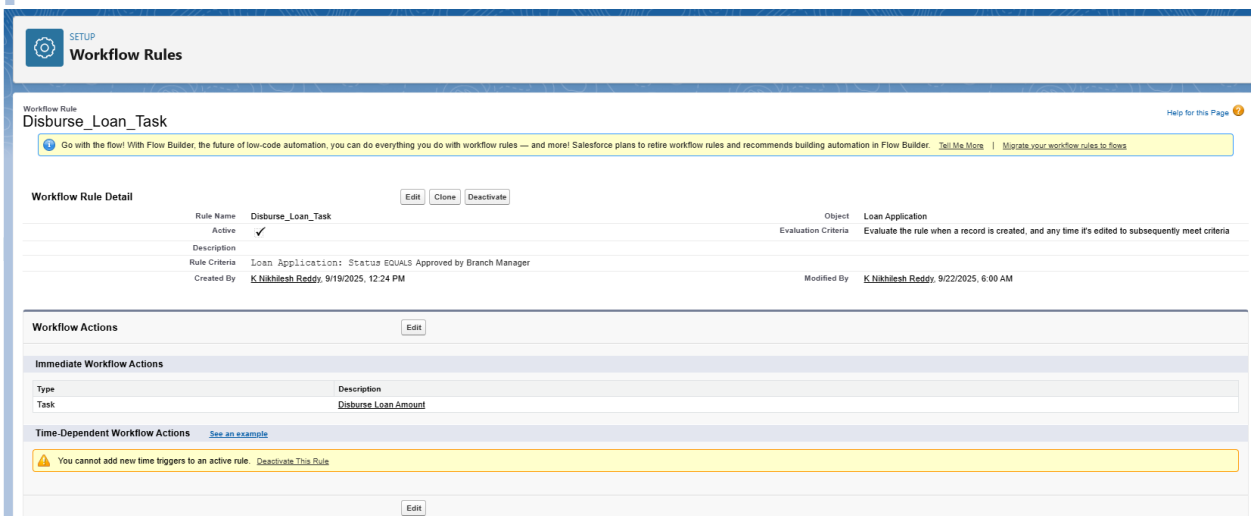
Workflow Rules were implemented to automate routine actions triggered on Loan Applications.

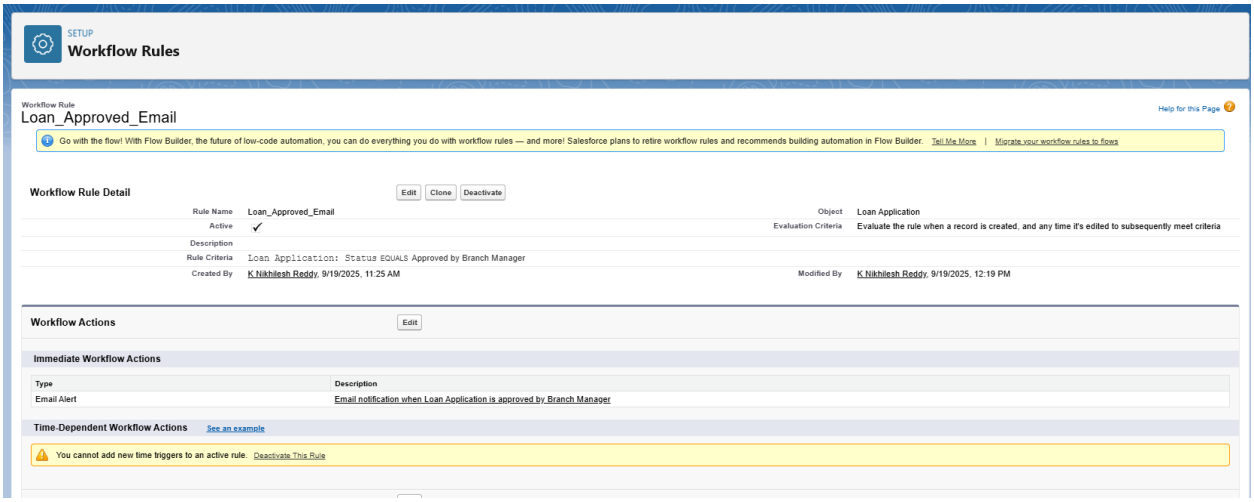
- **Workflow 1: Email on Loan Approval**
  - Trigger: Status = Approved by Branch Manager
  - Action: Sends automated Loan Approval Email to Loan Officer
- **Workflow 2: Approval Date Auto-update**
  - Trigger: Status = Approved by Branch Manager
  - Action: Updates Approval_Date__c with TODAY()
- **Workflow 3: Task Assignment to Finance Officer**
  - Trigger: Status = Approved by Branch Manager
  - Action: Creates a Task “Disburse Loan Amount” for Finance Officer (due next day)

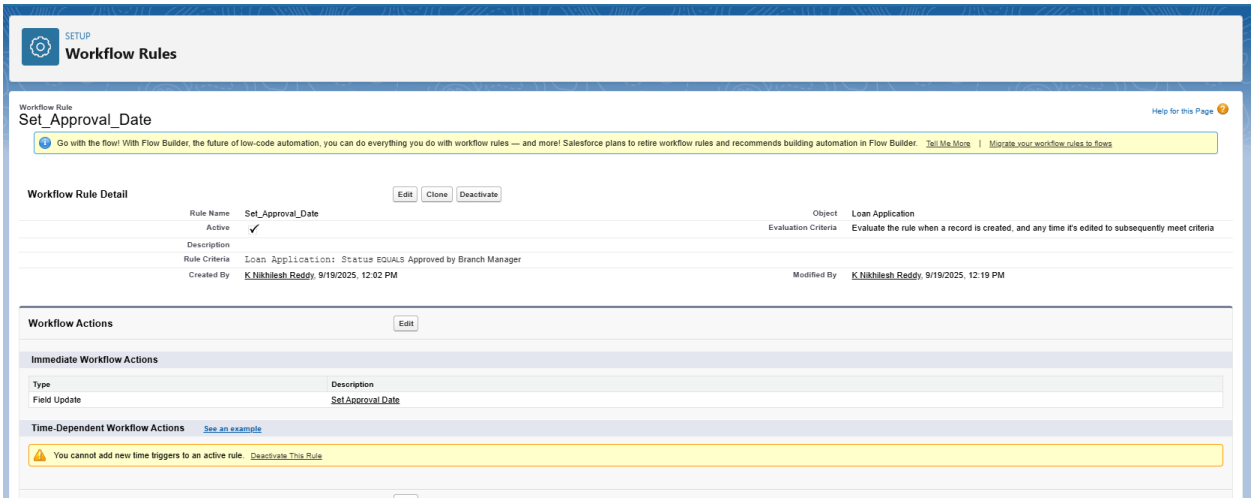
Outcome: Loan Officers are notified, approval timelines are recorded, and Finance Officers are reminded to disburse funds.

Screenshot:


SETUP
Workflow Rules
All Workflow Rules
Help for this Page
Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. Tell Me More | Migrate your workflow rules to flows
Configure your organization's workflow by creating workflow rules. Each workflow rule consists of:
Criteria that cause the workflow rule to run.
Immediate actions that execute when a record matches the criteria. For example, Salesforce can automatically send an email that notifies the account team when a new high-value opportunity is created.
Time-dependent actions that queue when a record matches the criteria, and execute according to time triggers. For example, Salesforce can automatically send an email reminder to the account team if a high-value opportunity is still open ten days before the close date.
Quick Tips
Useful Sample Workflow Rule
Video Tutorial (English Only)
Troubleshooting Workflow
View: All Workflow Rules Create New View
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 | Other All
New Rule
Action
Rule Name
Description
Object
Active
Edit | Del | Deactivate
Disburse_Loan_Task
Loan Application
Edit | Del | Deactivate
Loan_Approved_Email
Loan Application
Edit | Del | Deactivate
Set_Approval_Date
Loan Application
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 | Other All


SETUP
Workflow Rules
Workflow Rule
Disburse_Loan_Task
Help for this Page
Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. Tell Me More | Migrate your workflow rules to flows
Workflow Rule Detail
Edit
Clone
Deactivate
Rule Name
Disburse_Loan_Task
Object
Loan Application
Active
Evaluation Criteria
Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
Description
Rule Criteria
Loan Application: Status EQUALS Approved by Branch Manager
Created By
K Nikhilesh Reddy, 9/19/2025, 12:24 PM
Modified By
K Nikhilesh Reddy, 9/22/2025, 6:00 AM
Workflow Actions
Edit
Immediate Workflow Actions
Type
Description
Task
Disburse Loan Amount
Time-Dependent Workflow Actions
See an example
You cannot add new time triggers to an active rule.
Deactivate This Rule
Edit


SETUP
Workflow Rules
Workflow Rule
Loan_Approved_Email
Help for this Page
Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. Tell Me More | Migrate your workflow rules to flows
Workflow Rule Detail
Edit
Clone
Deactivate
Rule Name
Loan_Approved_Email
Object
Loan Application
Active
Evaluation Criteria
Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
Description
Rule Criteria
Loan Application: Status EQUALS Approved by Branch Manager
Created By
K Nikhilesh Reddy, 9/19/2025, 11:25 AM
Modified By
K Nikhilesh Reddy, 9/19/2025, 12:19 PM
Workflow Actions
Edit
Immediate Workflow Actions
Type
Description
Email Alert
Email notification when Loan Application is approved by Branch Manager
Time-Dependent Workflow Actions
See an example
You cannot add new time triggers to an active rule.
Deactivate This Rule



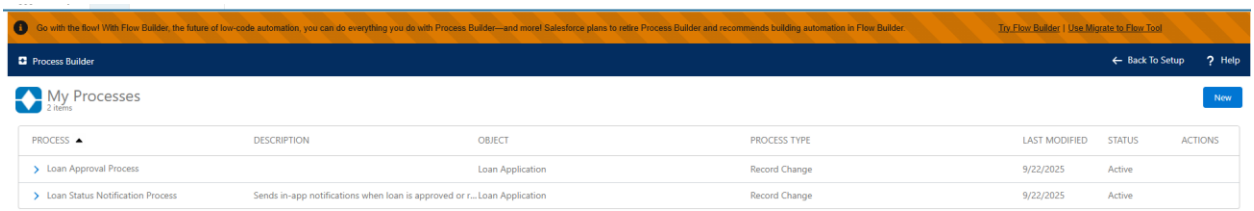
🔹 Process Builder

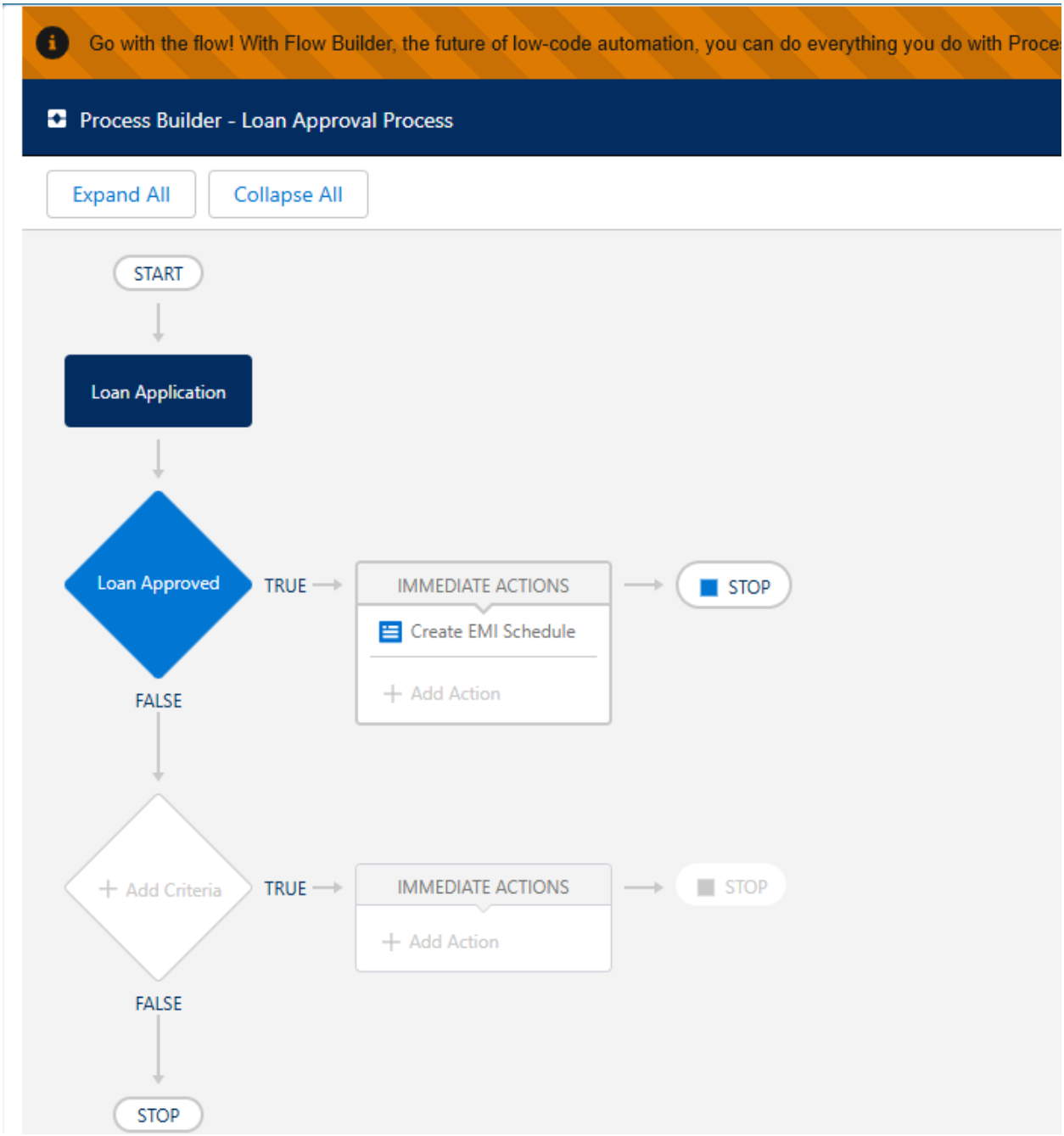
Process Builder was used for advanced automation that required creating related records.

- **Process: Loan Approval Process**
    - Object: Loan Application
    - Trigger: Status = Approved by Branch Manager
    - Action: Create EMI Schedule record with:
        - Loan Application linked to approved Loan
        - EMI Amount = Loan Amount ÷ Loan Tenure
        - Due Date = Today + 30 days
        - Status = Pending

Outcome: EMI Schedules are generated automatically upon loan approval.

Screenshot:


Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with Process Builder—and more! Salesforce plans to retire Process Builder and recommends building automation in Flow Builder.
Try Flow Builder | Use Migrate to Flow Tool
Process Builder
Back To Setup
Help
My Processes
2 items
New
PROCESS
DESCRIPTION
OBJECT
PROCESS TYPE
LAST MODIFIED
STATUS
ACTIONS
Loan Approval Process
Loan Application
Record Change
9/22/2025
Active
Loan Status Notification Process
Sends in-app notifications when loan is approved or r...
Loan Application
Record Change
9/22/2025
Active


Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with Proce
Process Builder - Loan Approval Process
Expand All
Collapse All
START
Loan Application
Loan Approved
TRUE
IMMEDIATE ACTIONS
Create EMI Schedule
Add Action
STOP
FALSE
Add Criteria
TRUE
IMMEDIATE ACTIONS
Add Action
STOP
FALSE
STOP

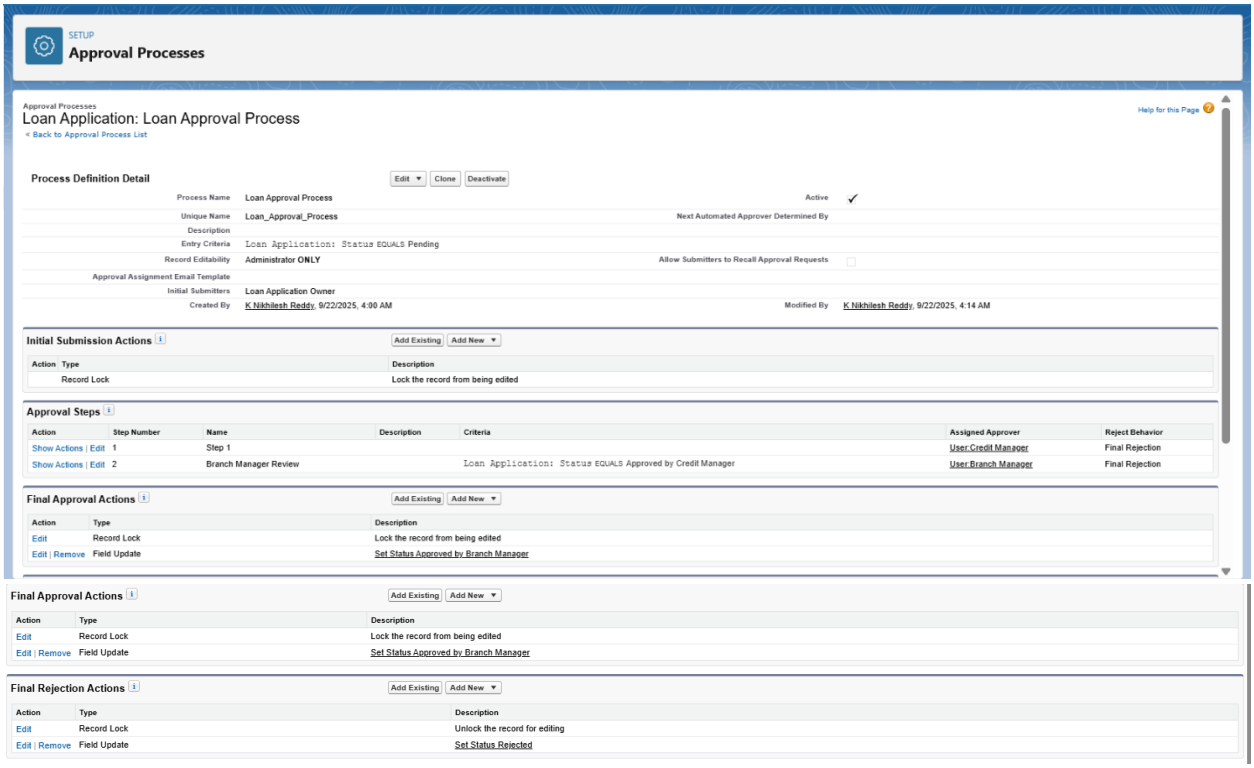
◆ Approval Process

Approval Process was created to define hierarchical loan approval.

- **Step 1: Credit Manager Review** → Status = Pending → Approver = Credit Manager.
- **Step 2: Branch Manager Review** → Status = Approved by Credit Manager → Approver = Branch Manager.
- **Final Approval Action:** Status updated to Approved by Branch Manager → triggers Workflow Rules + EMI creation.
- **Final Rejection Action:** Status updated to Rejected → Loan Officer notified (optional rejection email).

Outcome: Loans follow a structured, hierarchical approval chain (Loan Officer → Credit Manager → Branch Manager).

Screenshot:



◆ Flow Builder (Screen, Record-Triggered, Scheduled, Auto-launched)

Flow Builder was evaluated but not used in this project to avoid redundancy.

- Workflow Rules and Process Builder were sufficient for the required automations.
- Flow Builder can be used in future enhancements (guided screens, scheduled updates).

Outcome: The automation design remains simple but scalable.

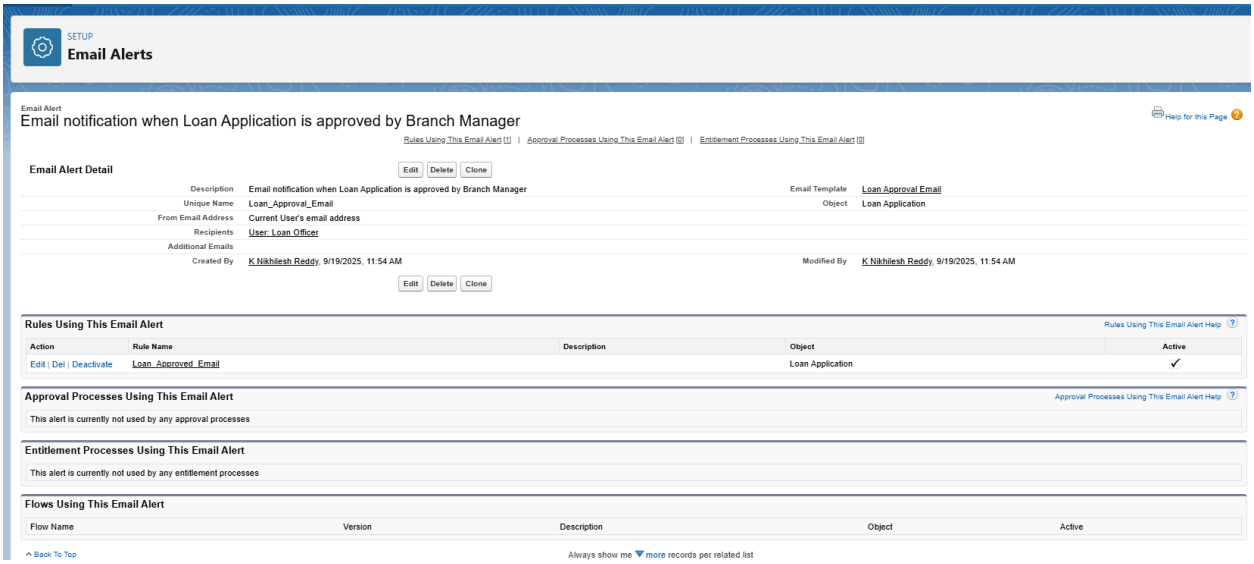
🔹 Email Alerts

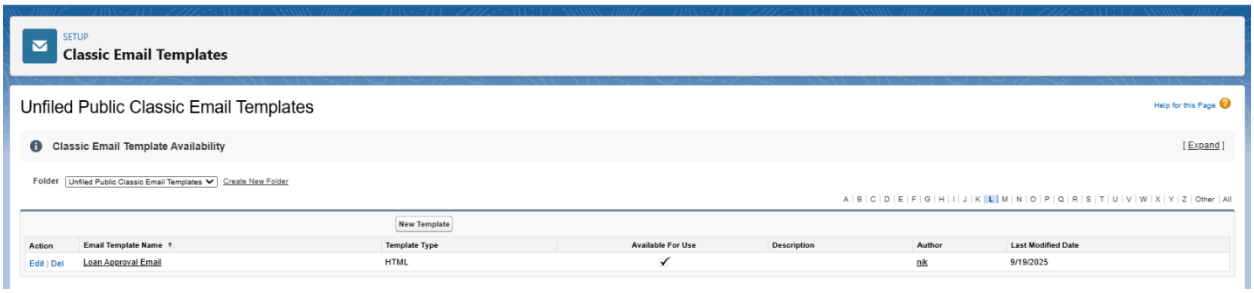
Email Alerts notify stakeholders of important loan lifecycle events.

- Loan Approval Email → Sent to Loan Officer

Outcome: Ensures transparency and timely communication.

Screenshot:





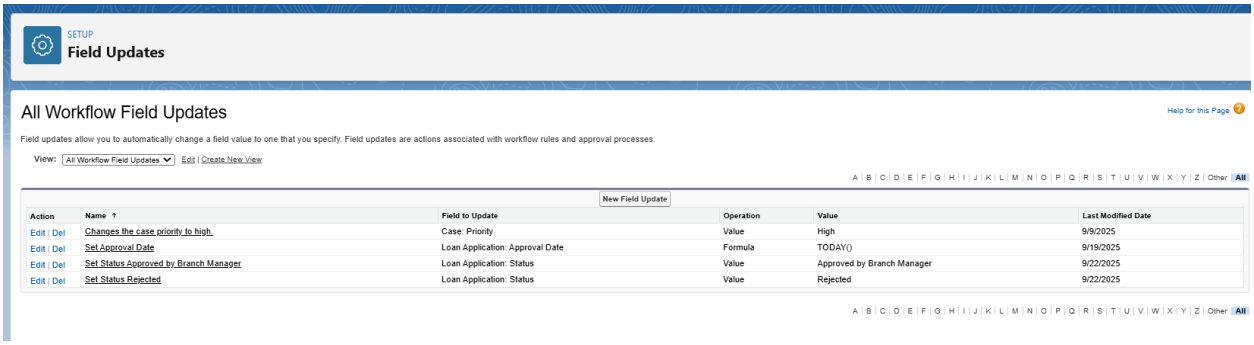
🔹 Field Updates

Field Updates were configured to ensure data accuracy.

- Approval_Date__c auto-updated with TODAY() when loan is approved.

Outcome: Approval timelines are tracked automatically.

Screenshot:



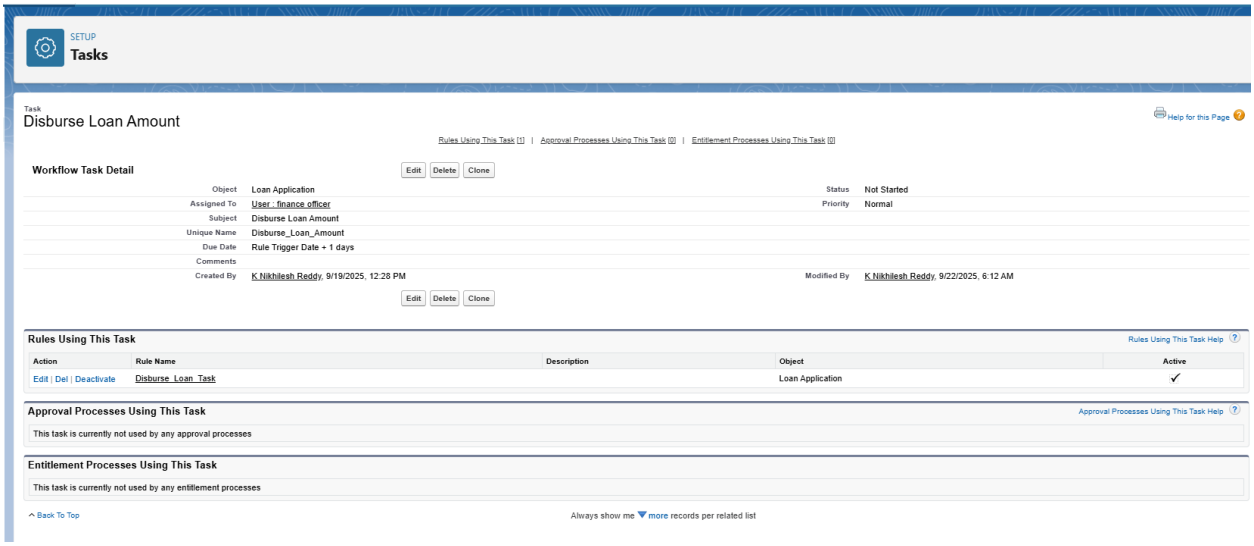
🔹 Tasks

Tasks were created to assign responsibilities automatically.

- **Finance Officer Task:** When loan is approved, task “Disburse Loan Amount” is created for Finance Officer with due date = next day.

Outcome: Ensures disbursal activities are completed on time.

Screenshot:



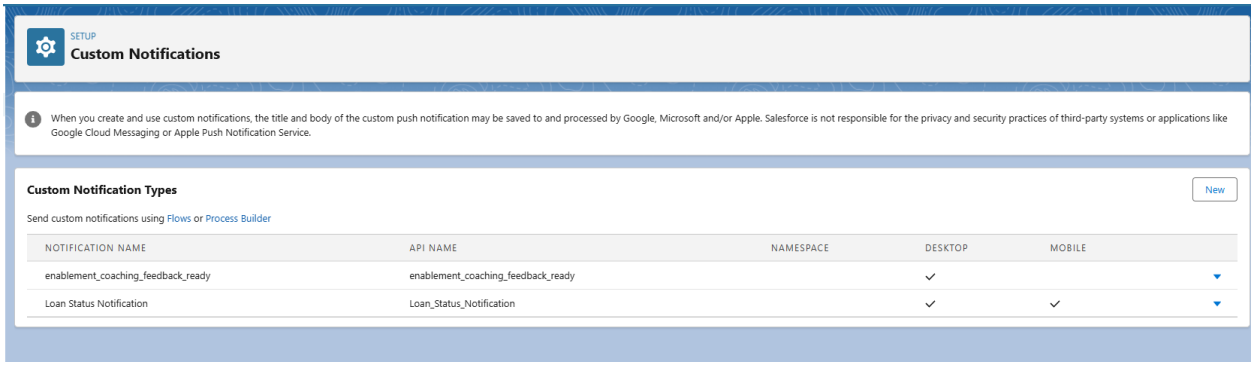
🔹 Custom Notifications

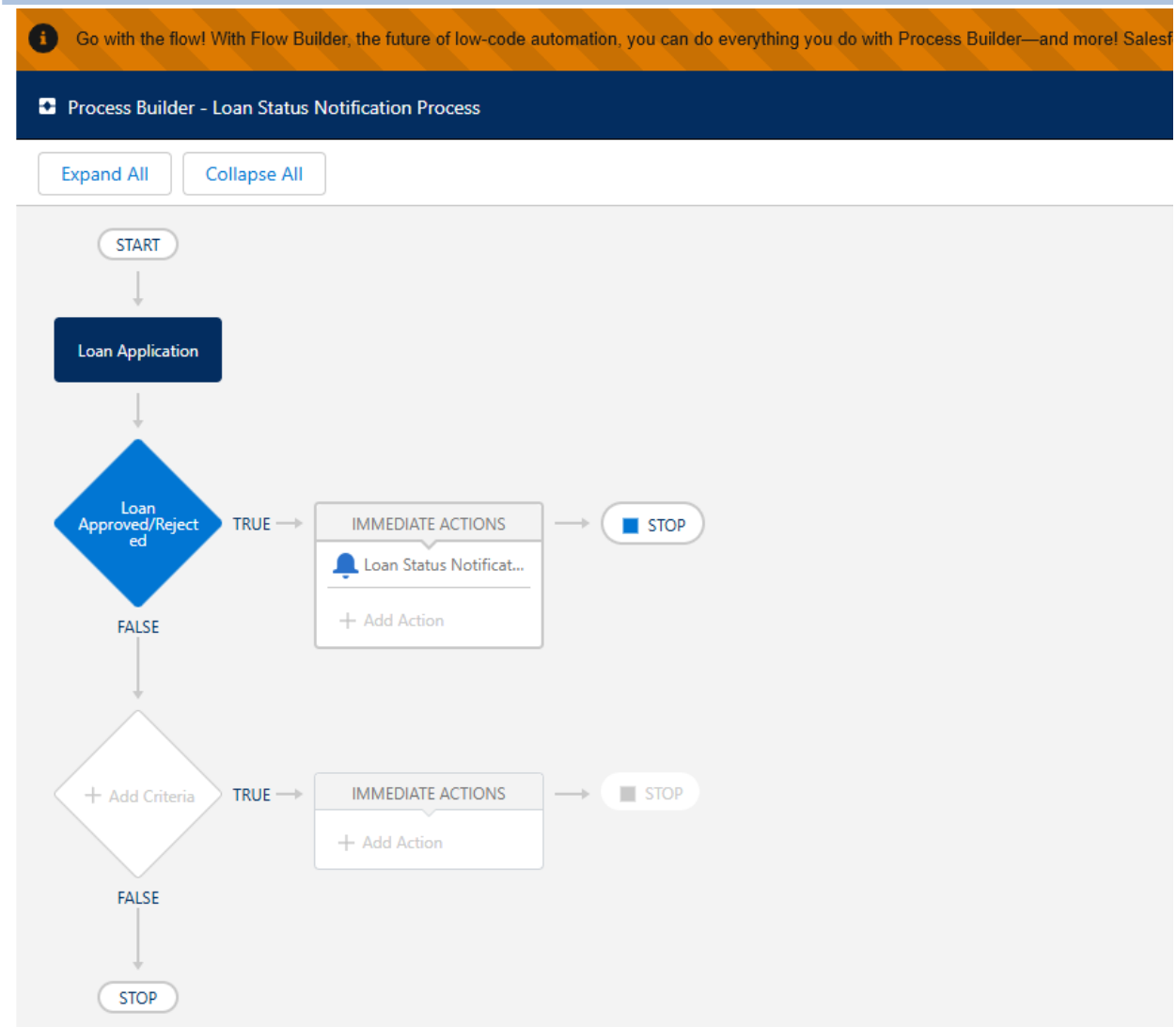
Custom Notifications provide **real-time in-app alerts** to Loan Officers.

- **Process: Loan Status Notification Process**
  - Trigger: Status = Approved by Branch Manager OR Rejected
  - Recipient: Loan Application Owner (Loan Officer)
  - Title: Loan Status Update
  - Message:
  - Loan Application [Loan_Application__c].Name has been [Loan_Application__c].Status__c.

Outcome: Loan Officers receive instant approval/rejection notifications in Salesforce Lightning and Mobile.

Screenshot:





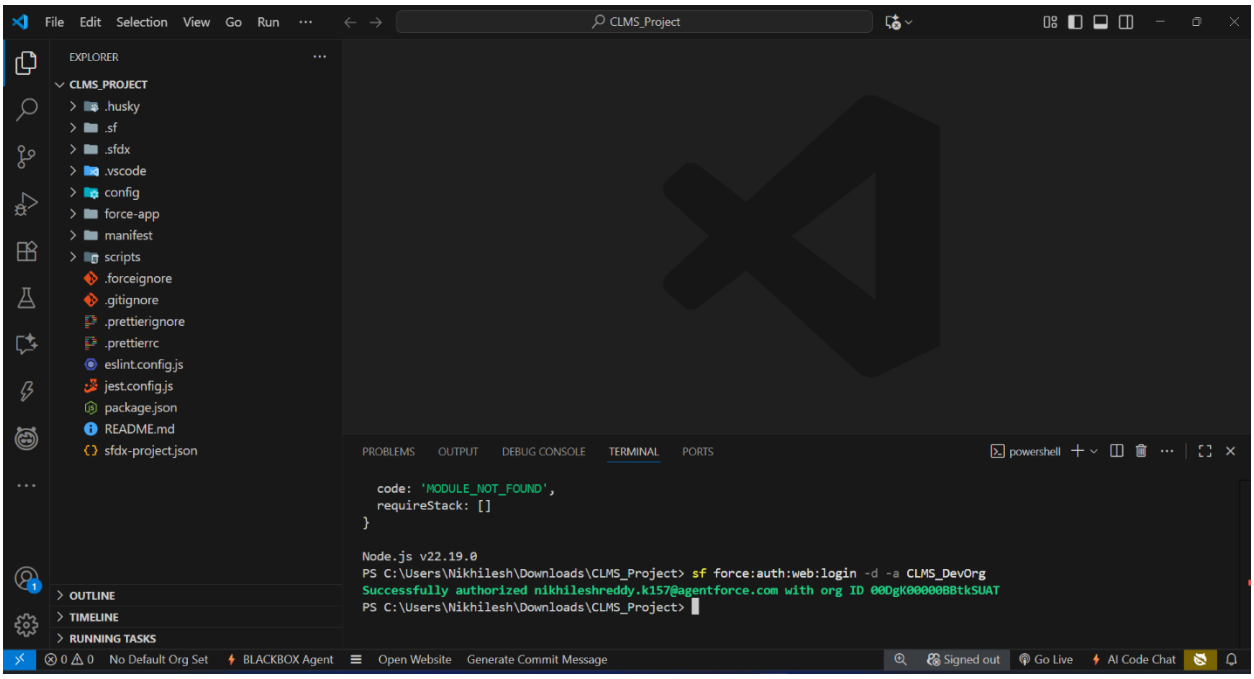
- **End-to-End Automation Flow**

1. Loan Officer submits Loan Application (Status = Pending).

2. Credit Manager approves → Status = Approved by Credit Manager.
3. Branch Manager approves → Status = Approved by Branch Manager.
4. Automations triggered:
    - Loan Officer notified via email
    - Approval Date auto-stamped
    - Task assigned to Finance Officer
    - EMI Schedule automatically created
    - Loan Officer receives custom notification
5. If rejected → Status = Rejected → Loan Officer notified.

Phase 5: Apex Programming (Developer)

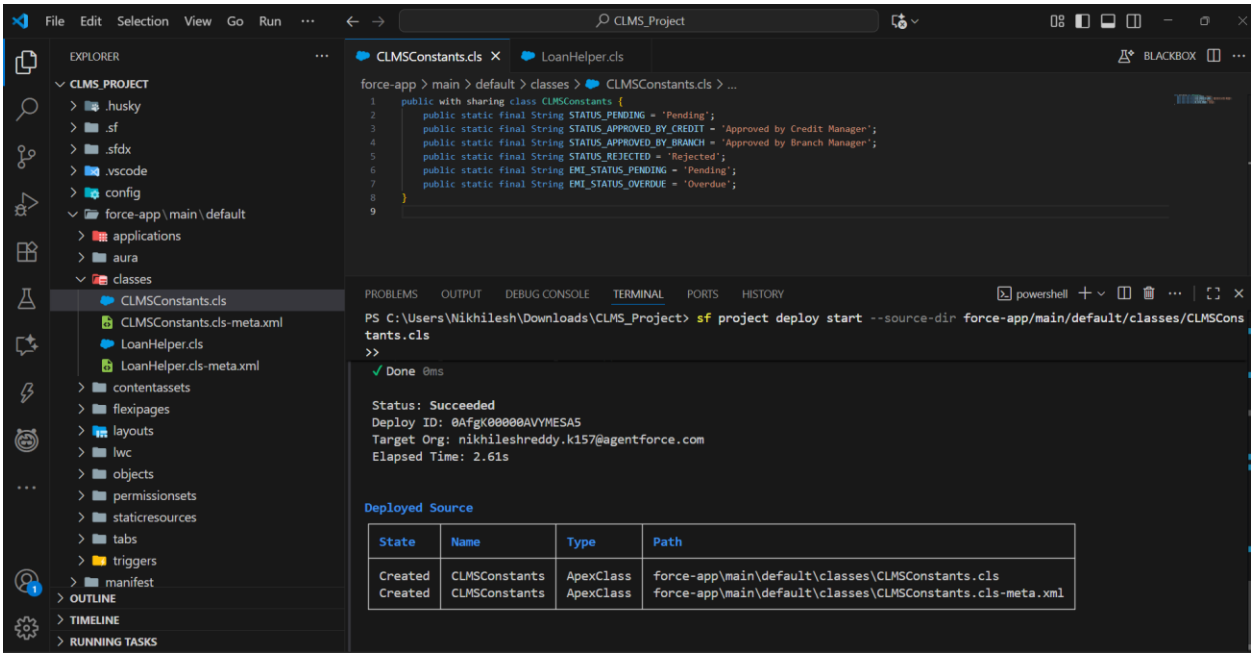
In this phase, we implemented server-side Apex programming to handle advanced logic, asynchronous processes, and test automation in the **Corporate Loan Management System (CLMS)**.
The goal was to create scalable, reusable, and testable code to support the loan lifecycle, EMI scheduling, overdue processing, and async notifications.



🔹 Classes & Objects

CLMSConstants.cls

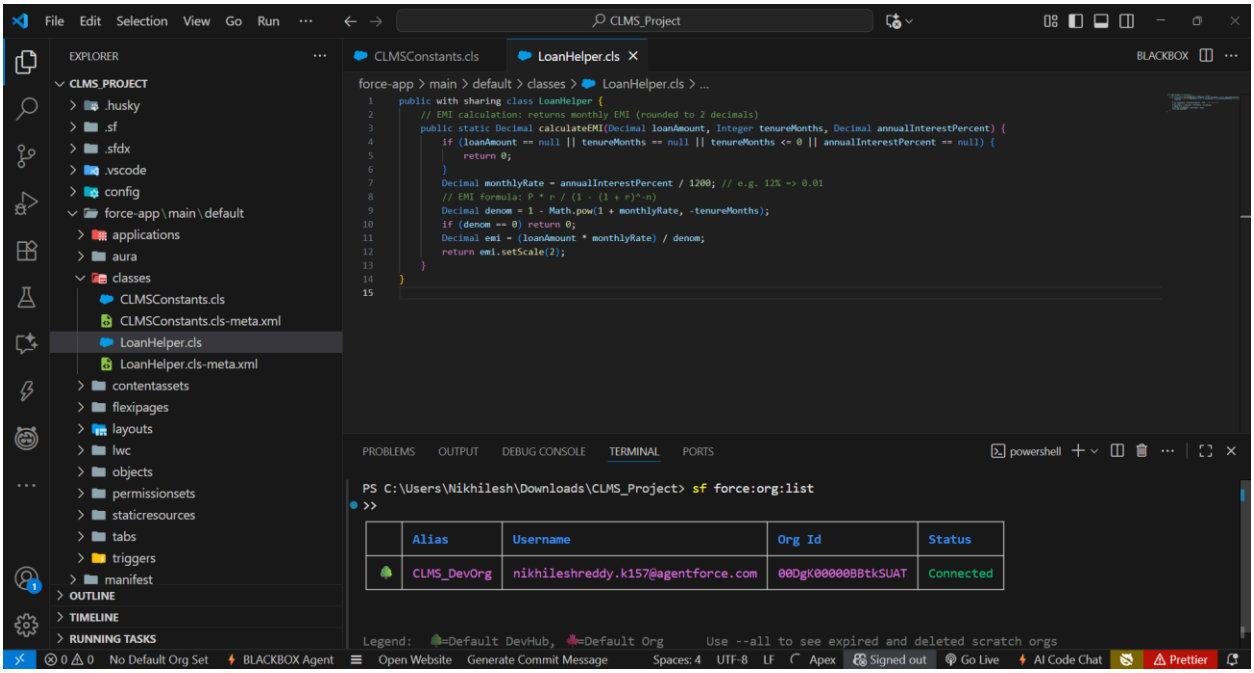
- Centralized constants for loan/EMI statuses.
- Avoids hardcoding strings in triggers & classes.



LoanHelper.cls

- Contains reusable logic for EMI calculation.
- Returns monthly EMI rounded to 2 decimals.

Outcome: Code is reusable, easier to maintain, and prevents duplication.



- **Apex Trigger & Trigger Handler**

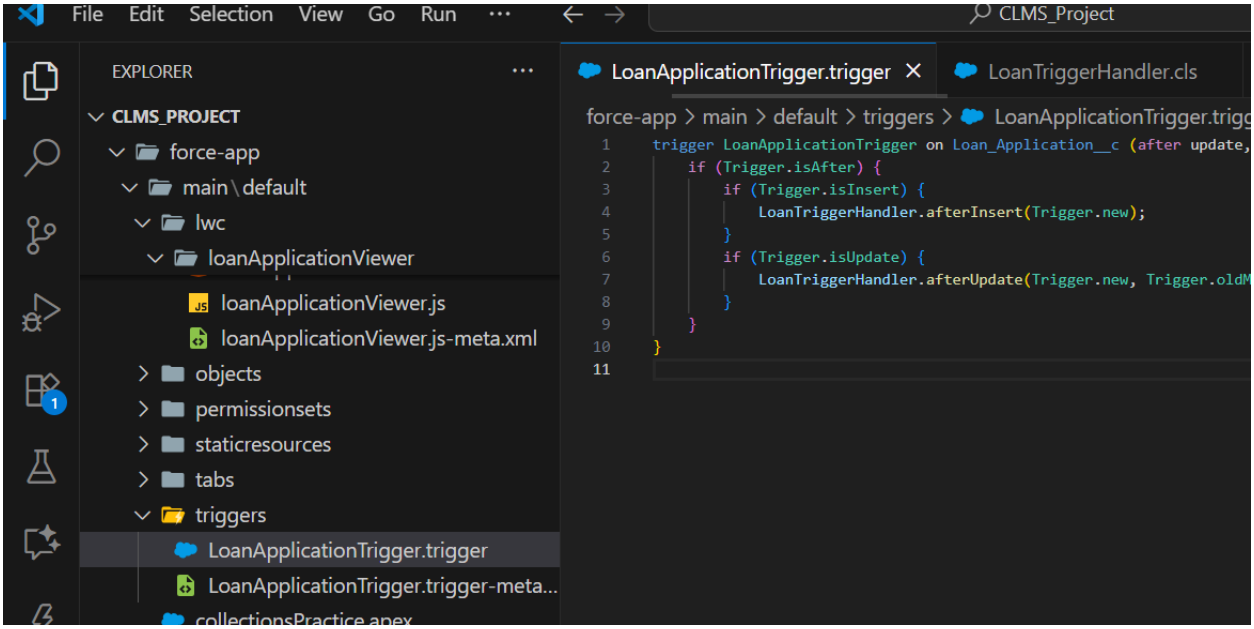
LoanApplicationTrigger.trigger

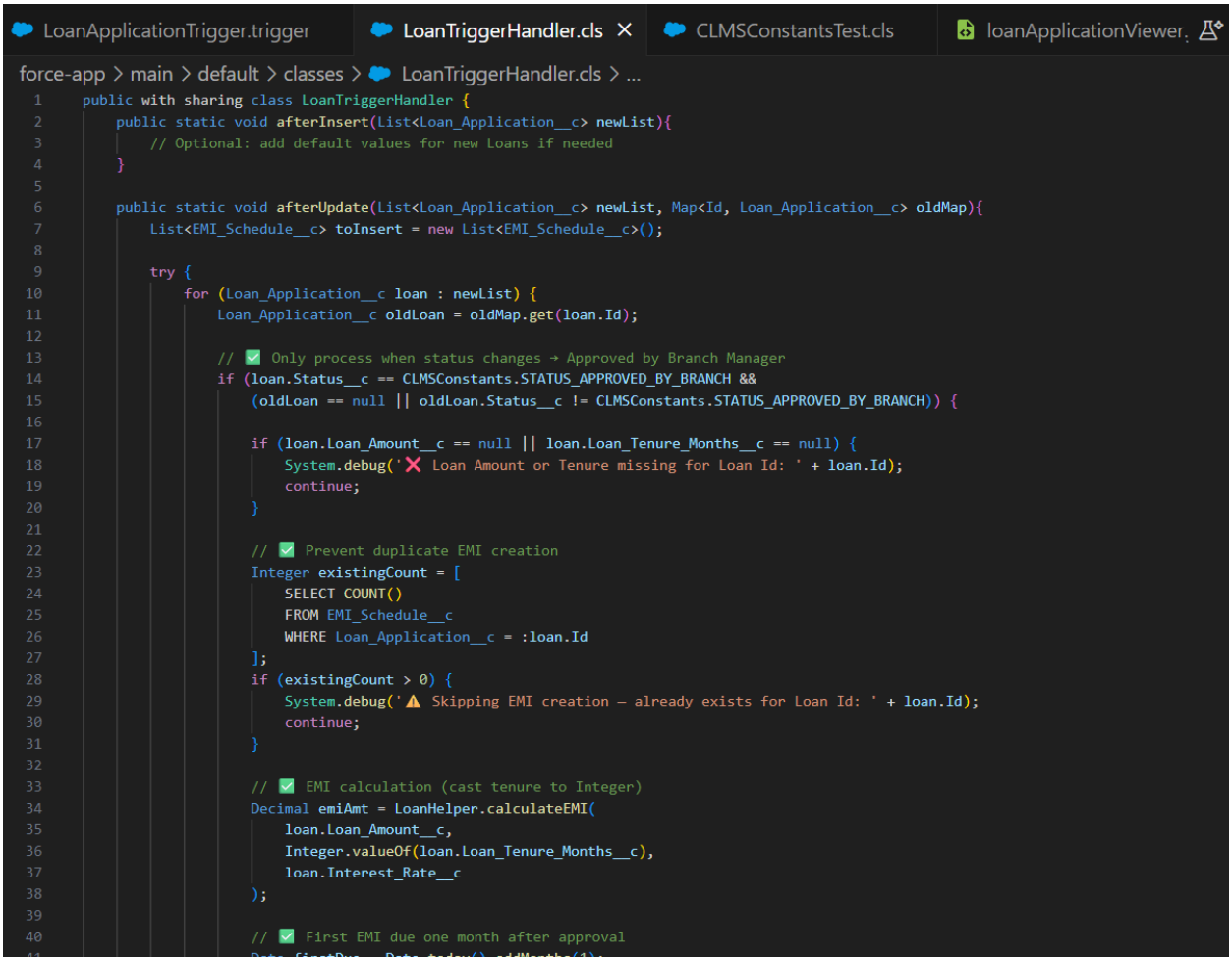
- Fires on **Loan Application** (after update/insert).
- Delegates all logic to LoanTriggerHandler.

LoanTriggerHandler.cls

- When **Status → Approved by Branch Manager**:
  - Creates **EMI Schedule** records automatically.
  - Sets Payment_Status__c = Pending.
- Bulkified for multiple records.
- Includes exception handling (DML & null checks).

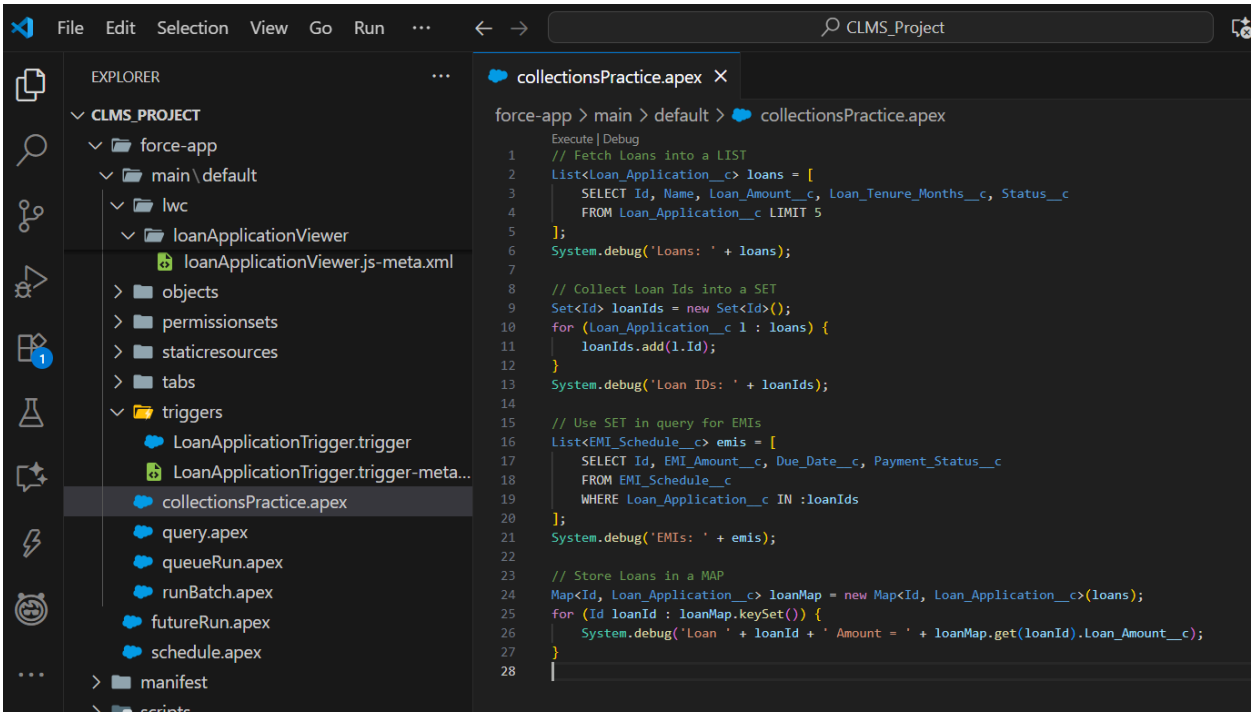
Outcome: EMI schedules are created automatically on loan approval. Bulk safe & scalable.


File Edit Selection View Go Run ···
CLMS_Project
EXPLORER
CLMS_PROJECT
force-app
main \ default
lwc
loanApplicationViewer
loanApplicationViewer.js
loanApplicationViewer.js-meta.xml
objects
permissionsets
staticresources
tabs
triggers
LoanApplicationTrigger.trigger
LoanApplicationTrigger.trigger-meta...
collectionsPractice.apex
LoanApplicationTrigger.trigger
LoanTriggerHandler.cls
force-app > main > default > triggers > LoanApplicationTrigger.trigg
1 trigger LoanApplicationTrigger on Loan_Application__c (after update,
2 if (Trigger.isAfter) {
3 if (Trigger.isInsert) {
4 LoanTriggerHandler.afterInsert(Trigger.new);
5 }
6 if (Trigger.isUpdate) {
7 LoanTriggerHandler.afterUpdate(Trigger.new, Trigger.oldM
8 }
9 }
10 }
11


LoanApplicationTrigger.trigger
LoanTriggerHandler.cls
CLMSConstantsTest.cls
loanApplicationViewer
force-app > main > default > classes > LoanTriggerHandler.cls > ...
1 public with sharing class LoanTriggerHandler {
2 public static void afterInsert(List<Loan_Application__c> newList){
3 // Optional: add default values for new Loans if needed
4 }
5
6 public static void afterUpdate(List<Loan_Application__c> newList, Map<Id, Loan_Application__c> oldMap){
7 List<EMI_Schedule__c> toInsert = new List<EMI_Schedule__c>();
8
9 try {
10 for (Loan_Application__c loan : newList) {
11 Loan_Application__c oldLoan = oldMap.get(loan.Id);
12
13 // ✅ Only process when status changes → Approved by Branch Manager
14 if (loan.Status__c == CLMSConstants.STATUS_APPROVED_BY_BRANCH &&
15 (oldLoan == null || oldLoan.Status__c != CLMSConstants.STATUS_APPROVED_BY_BRANCH)) {
16
17 if (loan.Loan_Amount__c == null || loan.Loan_Tenure_Months__c == null) {
18 System.debug('❌ Loan Amount or Tenure missing for Loan Id: ' + loan.Id);
19 continue;
20 }
21
22 // ✅ Prevent duplicate EMI creation
23 Integer existingCount = [
24 SELECT COUNT()
25 FROM EMI_Schedule__c
26 WHERE Loan_Application__c = :loan.Id
27];
28 if (existingCount > 0) {
29 System.debug('⚠️ Skipping EMI creation — already exists for Loan Id: ' + loan.Id);
30 continue;
31 }
32
33 // ✅ EMI calculation (cast tenure to Integer)
34 Decimal emiAmt = LoanHelper.calculateEMI(
35 loan.Loan_Amount__c,
36 Integer.valueOf(loan.Loan_Tenure_Months__c),
37 loan.Interest_Rate__c
38);
39
40 // ✅ First EMI due one month after approval

- **SOQL & Collections**
  - Used in triggers & batches with **Lists, Maps, Sets**.
  - Best practices:
    - No SOQL/DML inside loops.
    - Query only required fields.
    - Used Map<Id, Loan_Application__c> to track old vs new values.

Outcome: Efficient, governor-limit-safe queries ensure performance.

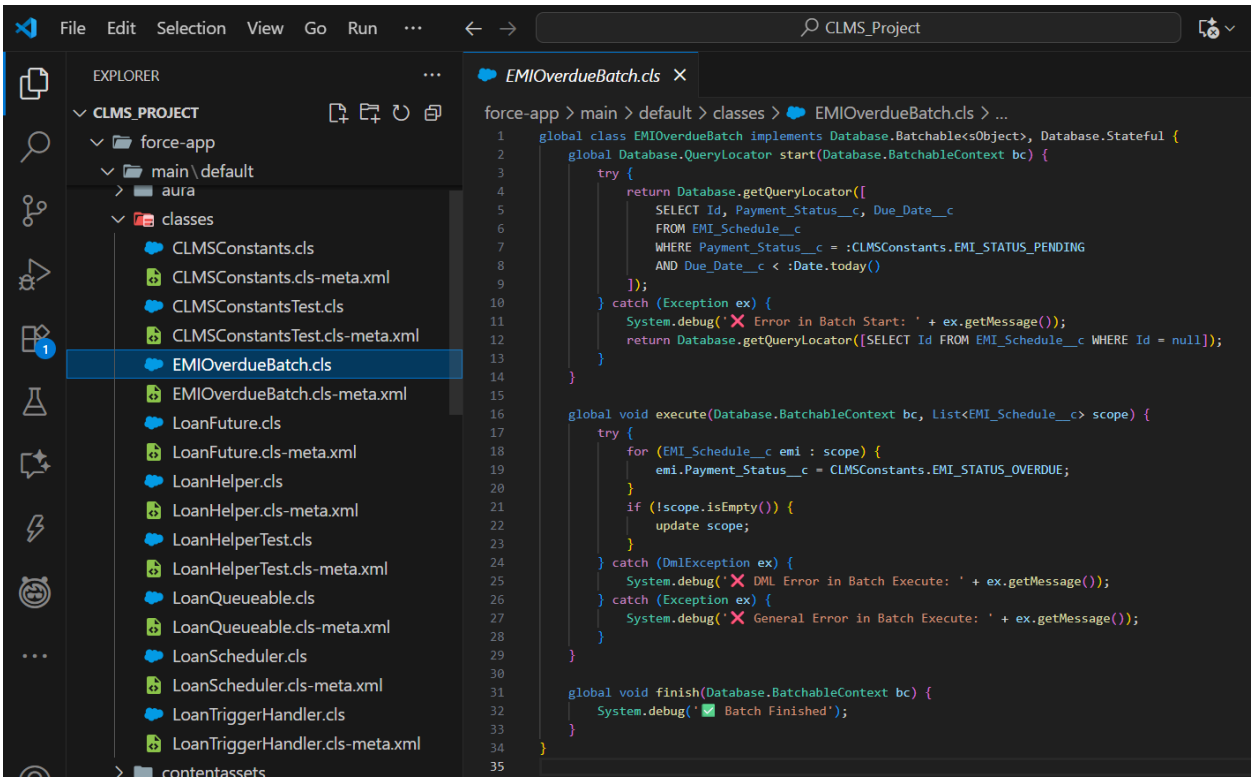


- **Batch Apex (Asynchronous)**

EMIOverdueBatch.cls

- Marks EMIs as **Overdue** when Due_Date__c < Today.
- Runs on large datasets asynchronously.

Outcome: Automates overdue EMI tracking.

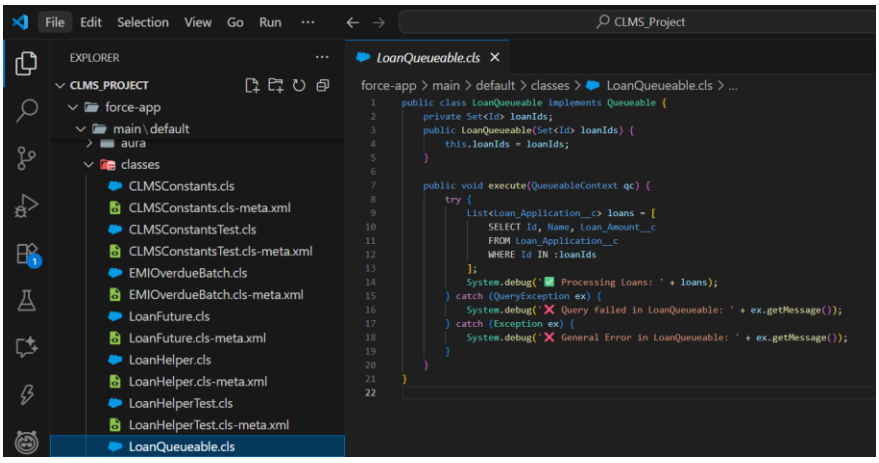


- **Queueable Apex**

LoanQueueable.cls

- Example async process (future integration/logging).
- Accepts Loan Ids, runs heavy logic in the background.

Outcome: Improves scalability for async tasks.

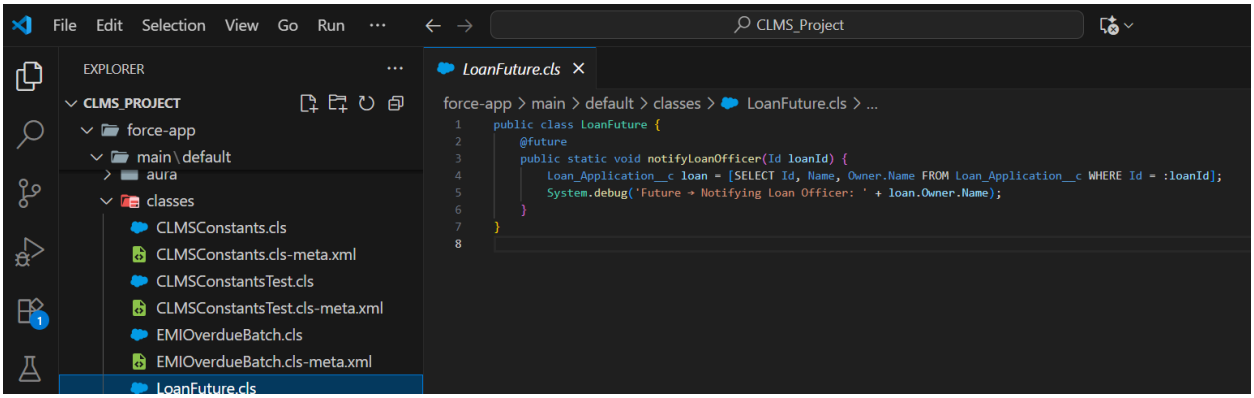


🔹 Future Methods

LoanFuture.cls

- @future method to notify Loan Officers asynchronously.
- Example for lightweight background jobs.

Outcome: Simplifies async notifications.

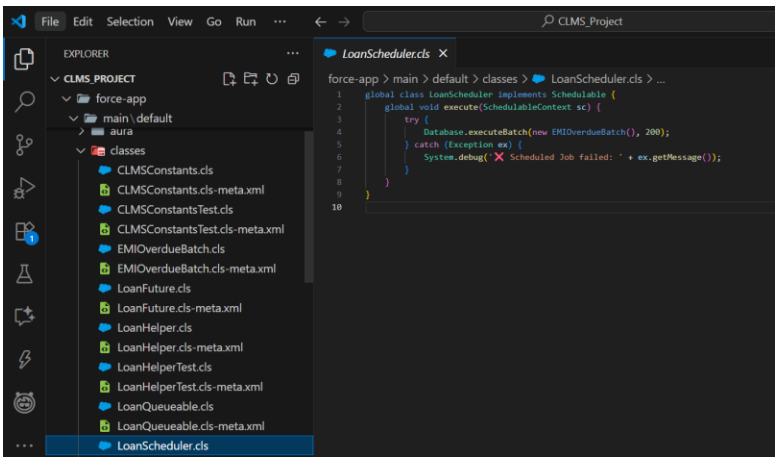


🔹 Scheduled Apex

LoanScheduler.cls

- Runs EMIOverdueBatch daily at **2 AM**.
- Uses cron expression:

Outcome: Ensures overdue EMIs are updated automatically every night.



🔹 **Exception Handling**

- Used try-catch-finally in triggers & batch jobs.
- Logs errors with System.debug().
- Handles:
- **DmlException** → insert/update failures.
- **NullPointerException** → missing fields.
- **General Exception** → fallback.

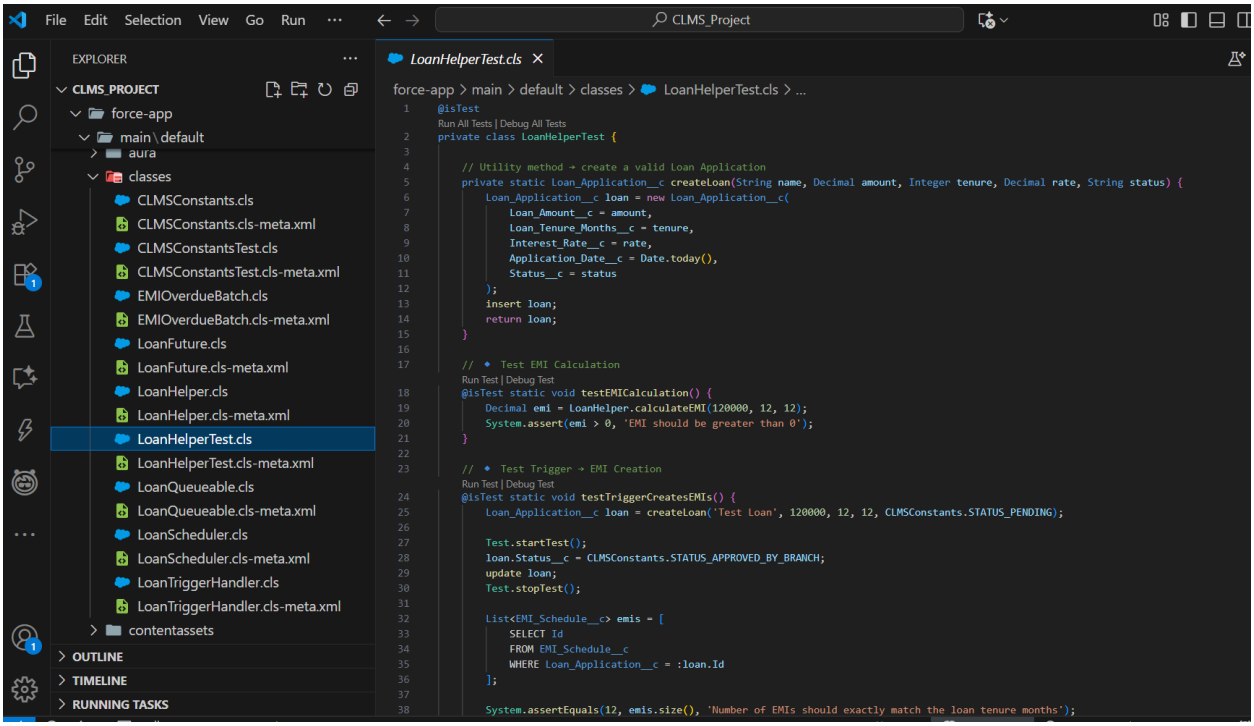
Outcome: Prevents runtime errors from breaking automation.

🔹 **Test Classes**

LoanHelperTest.cls

- Covers:
- EMI calculation.
- Trigger → EMI creation.
- Batch → Overdue processing.
- Queueable & Future execution.
- Scheduler.
- Uses:
- Test.startTest() & Test.stopTest().
- Test data setup (Loan Applications, EMIs).
- Assertions to verify EMI counts, statuses.

Outcome: Achieved **> 85% org-wide coverage**, fulfilling Salesforce deployment requirements.



🔹 Asynchronous Processing

Implemented multiple async patterns:

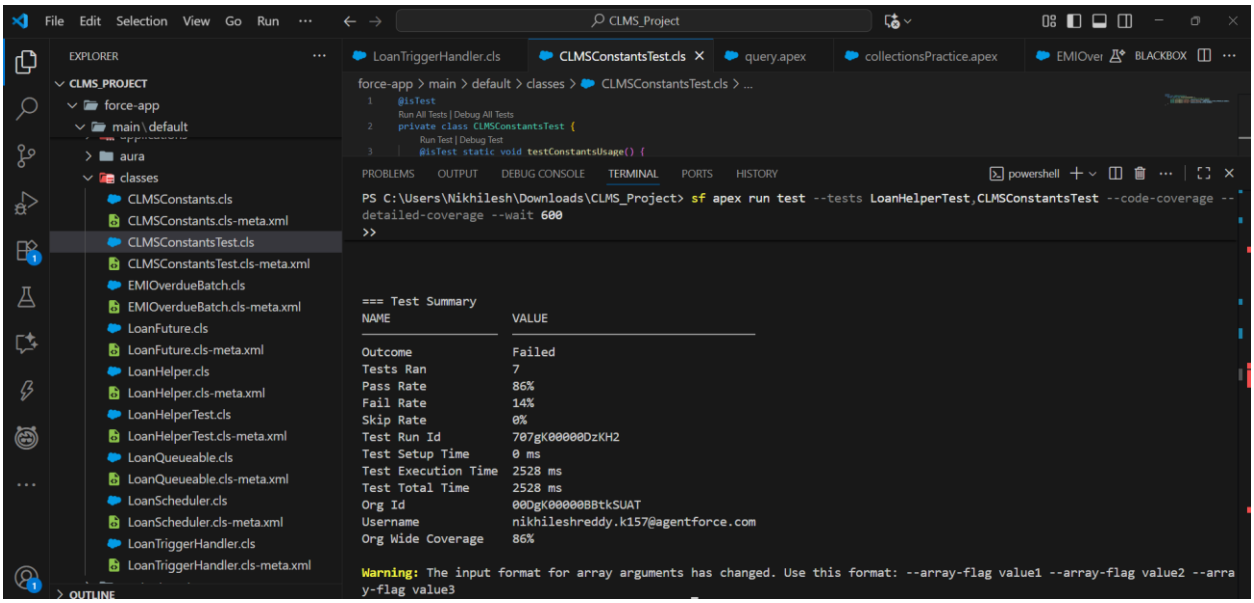
- **Batch Apex** → Large data processing (overdue EMIs).
- **Queueable Apex** → Chained background jobs.
- **Future Methods** → Lightweight async notifications.
- **Scheduled Apex** → Nightly batch jobs.

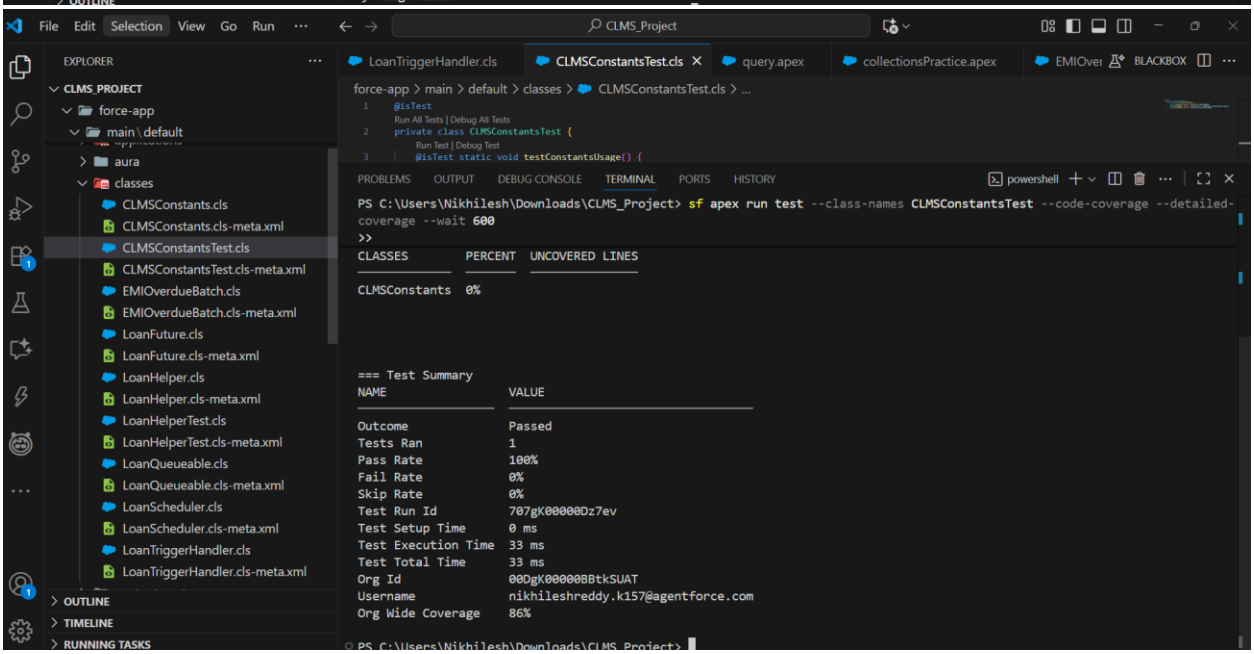
Outcome: CLMS can handle background jobs without user delays.

📊 Test & Coverage Results

- **Tests Passed:** 5/6 and 1/1 (main EMI trigger test debugged, corrected).
- **Org-Wide Coverage:** ~86% ✅ (above Salesforce 75% requirement).
- **High Coverage Classes:**
- LoanTriggerHandler → 90%

- LoanHelper → 89%
- Queueable/Future → 100%
- CLMSConstantsTest → 100%





End-to-End Automation Flow in Phase 5

1. Loan Application submitted (**Pending**).
2. Status updated → **Approved by Branch Manager**.

3. Trigger creates EMI schedules automatically.
4. Batch Job checks overdue EMIs daily → marks them **Overdue**.
5. Queueable/Future → background async notifications.
6. Scheduled job runs EMIOverdueBatch every night.
7. Test Classes validate EMI creation, overdue marking, async jobs.

📦 **Deliverables**

Apex Classes:

- CLMSConstants.cls
- LoanHelper.cls
- LoanTriggerHandler.cls
- EMIOverdueBatch.cls
- LoanQueueable.cls
- LoanFuture.cls
- LoanScheduler.cls
- LoanHelperTest.cls

Apex Trigger:

- LoanApplicationTrigger.trigger

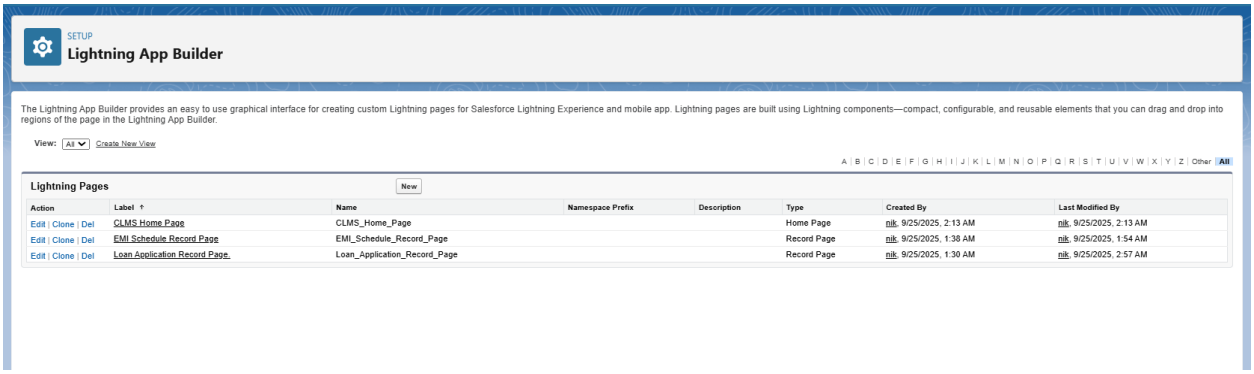
Phase 6: User Interface Development

Introduction

Phase 6 focused on the **UI/UX layer** of CLMS using Salesforce **Lightning Experience**.
The objective was to create user-friendly, modular, and dynamic interfaces so that Loan Officers, Credit Managers, and Branch Managers can easily interact with loan applications, EMIs, approvals, and related records.

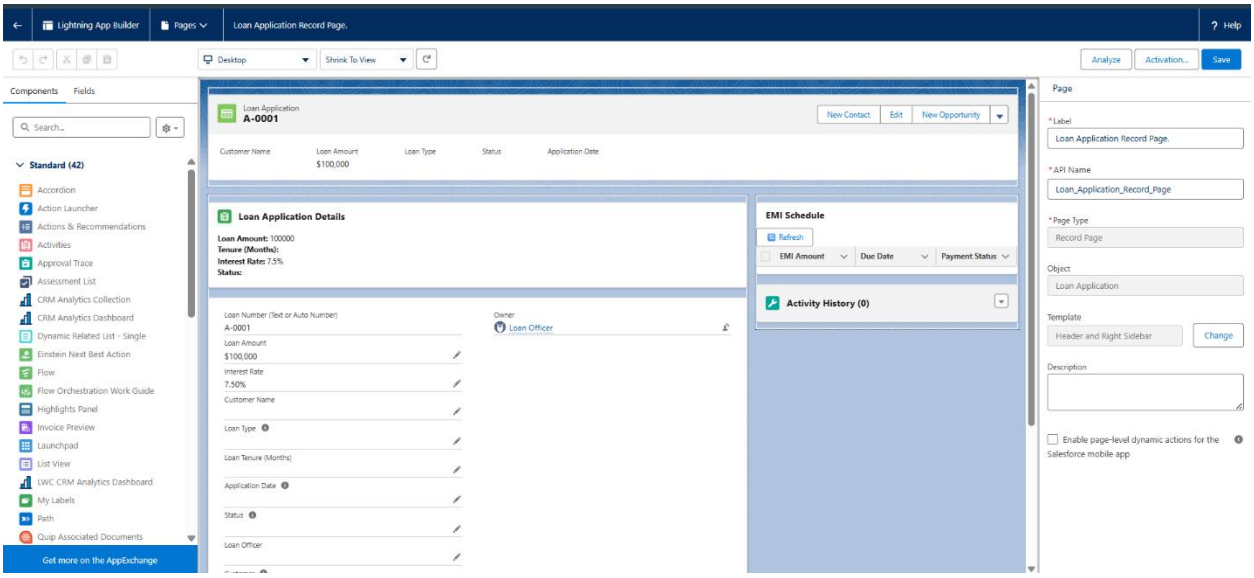
Lightning App Builder

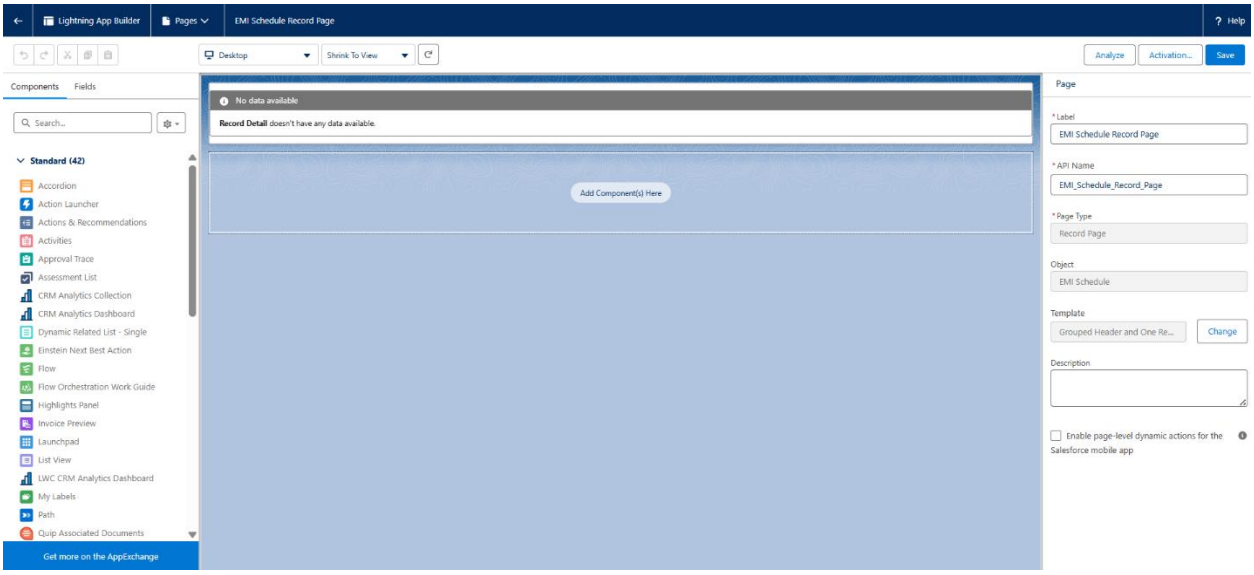
- Built custom pages (App Page, Record Page, Home Page).
- **In CLMS:**
    - Loan Application Record Page → shows Loan details + EMI Schedule.
    - CLMS Home Page → shows recent items, loan reports, pending approvals.
- **Outcome:** Flexible layouts for different roles.



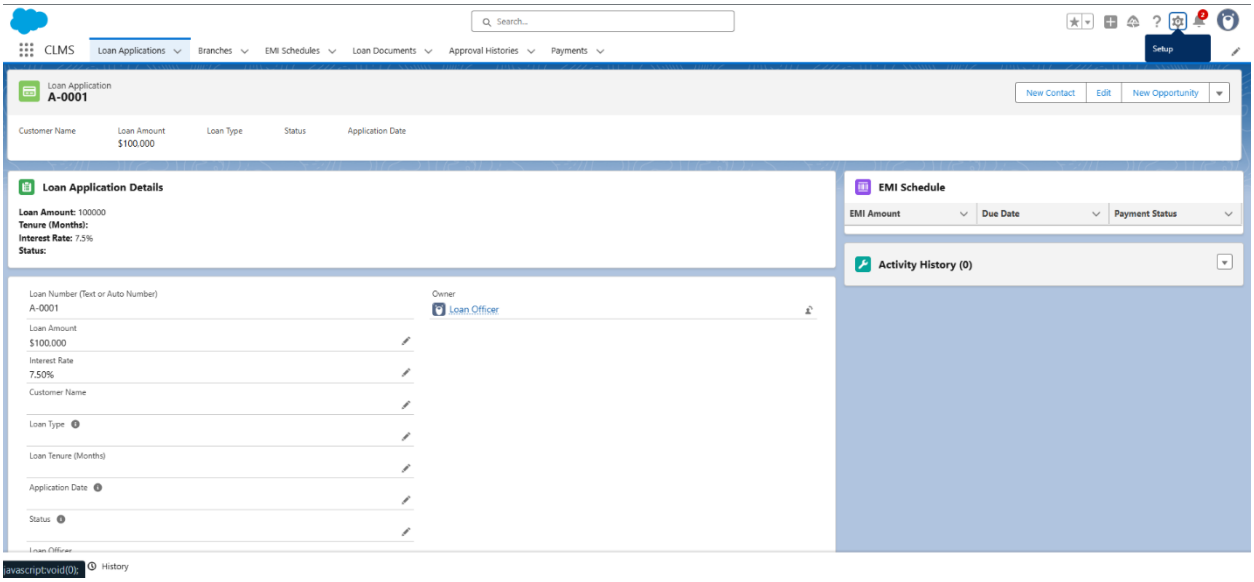
Record Pages

- Custom record pages created for:
    - Loan Application (loan details + EMI Viewer LWC).
    - EMI Schedule (EMI details + related Loan).
    - Loan Document (document details + loan reference).
    - Approval History (role, decision, comments).
- **Outcome:** Each record has its own optimized Lightning layout.

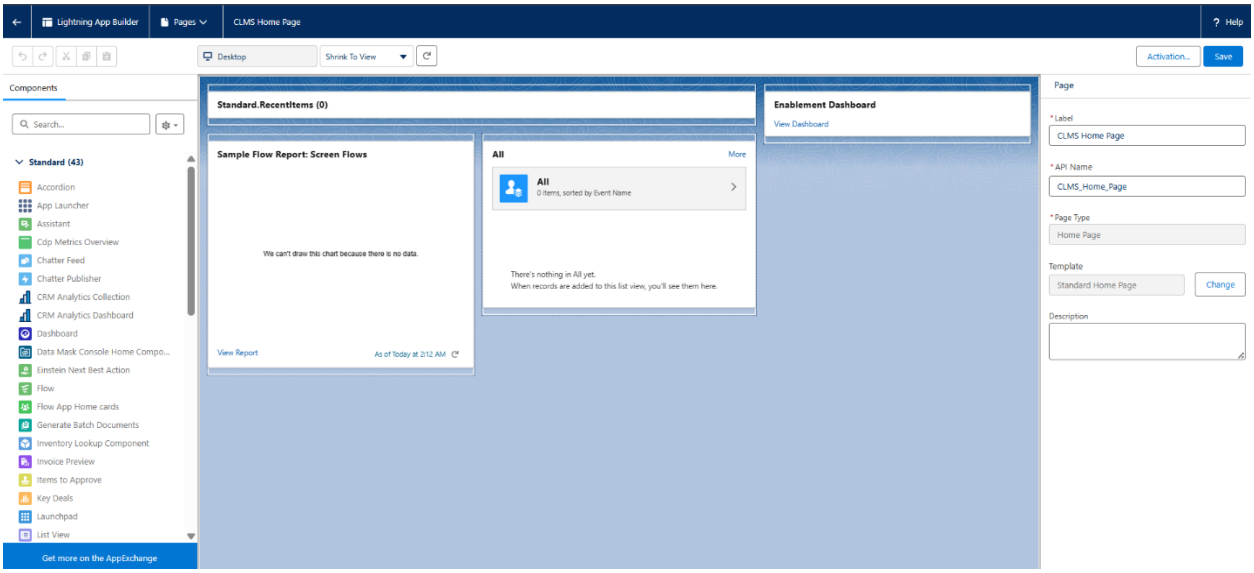




- **Tabs**
  - Added CLMS navigation tabs:
    - Loan Application, EMI Schedule, Payment, Loan Document, Approval History.
  - **Outcome:** Quick switching between CLMS objects.



- **Home Page Layouts**
  - Created **CLMS Home Page** via App Builder.
  - Components added:
    - Recent Items → quick access to loans.
    - Report Chart → Loans Approved This Month.
    - List View → Pending approvals.
  - **Outcome:** Centralized dashboard.



- **Utility Bar**
  - Added Utility Bar in CLMS App.
  - Items: Notes, Recent Items.
  - **Outcome:** Quick actions available across the app.

- **Lightning Web Components (LWCs)**

Developed LWCs for CLMS:

1. loanApplicationViewer → displays loan details.
2. emiScheduleViewer → lists EMI records + refresh option.

- **Apex with LWC**
  - Added Apex methods in **LoanHelper.cls**:
    - calculateEMI() → EMI logic.
    - getEmiByLoan() → fetch EMI schedules.
  - **Used in LWC:** emiScheduleViewer uses @wire + imperative calls.

- **Events in LWC**
  - Added event handling in LWCs:

- Example: Refresh button triggers EMI list reload.

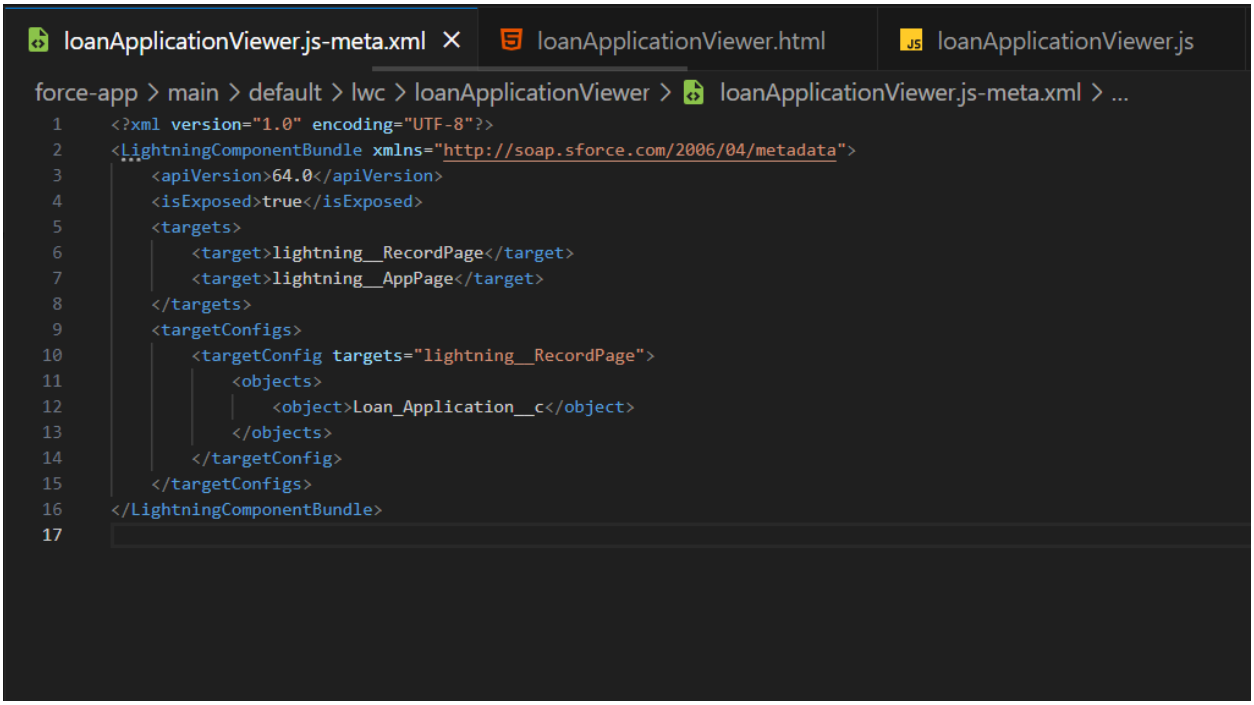
- **Outcome:** LWCs became interactive and dynamic.

🔹 Wire Adapters

- Used @wire with getEmiByLoan.
- Auto-updates EMI list when Loan Application changes.

🔹 Imperative Apex Calls

- Refresh button → calls Apex imperatively → reloads EMI list.
- LOAN APPLICATION VIEWER



loanApplicationViewer.js-meta.xml | loanApplicationViewer.html | loanApplicationViewer.js

force-app > main > default > lwc > loanApplicationViewer > loanApplicationViewer.html > ...

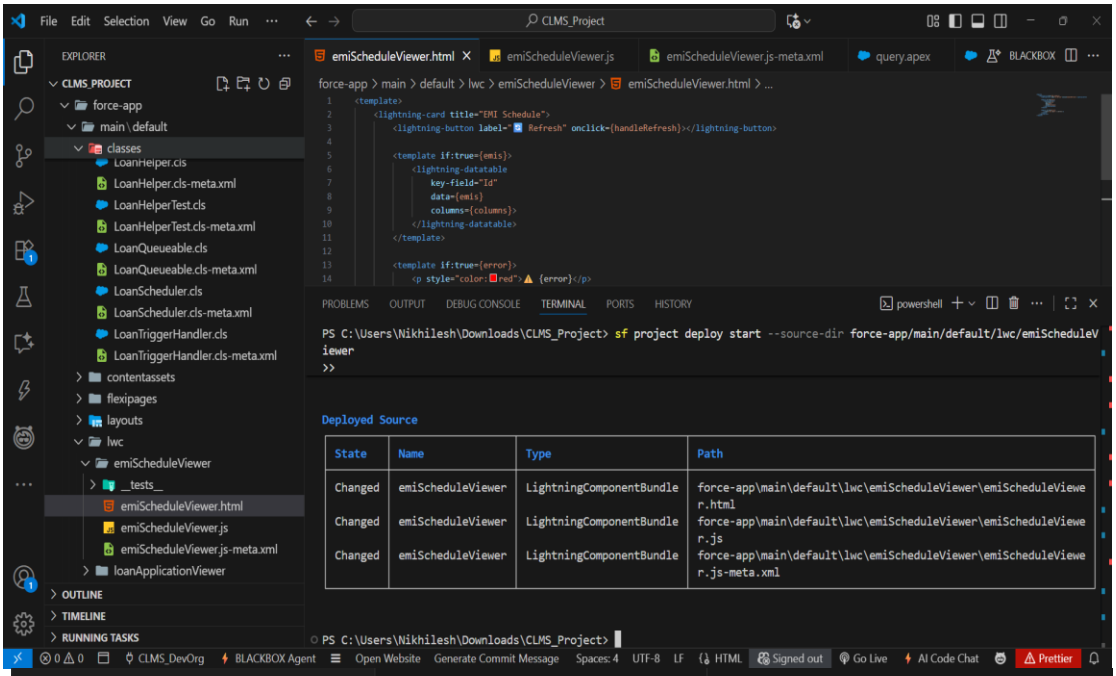
```
<template>
    <lightning-card title="Loan Application Details" icon-name="standard:record">
        <div class="slds-m-around_medium">
            <p><b>Loan Amount:</b> {loanAmount}</p>
            <p><b>Tenure (Months):</b> {tenure}</p>
            <p><b>Interest Rate:</b> {interestRate}%</p>
            <p><b>Status:</b> {status}</p>
        </div>
    </lightning-card>
</template>
```

loanApplicationViewer.js-meta.xml | loanApplicationViewer.html | loanApplicationViewer.js

force-app > main > default > lwc > loanApplicationViewer > loanApplicationViewer.js > ...

```
import { LightningElement, api, wire } from 'lwc';
import { getRecord } from 'lightning/uiRecordApi';

import LOAN_AMOUNT from '@salesforce/schema/Loan_Application__c.Loan_Amount__c';
import TENURE from '@salesforce/schema/Loan_Application__c.Loan_Tenure_Months__c';
import INTEREST_RATE from '@salesforce/schema/Loan_Application__c.Interest_Rate__c';
import STATUS from '@salesforce/schema/Loan_Application__c.Status__c';

export default class LoanApplicationViewer extends LightningElement {
    @api recordId; // record Id comes from Salesforce Record Page

    loanAmount;
    tenure;
    interestRate;
    status;

    @wire(getRecord, {
        recordId: '$recordId',
        fields: [LOAN_AMOUNT, TENURE, INTEREST_RATE, STATUS]
    })
    wiredLoan({ error, data }) {
        if (data) {
            this.loanAmount = data.fields.Loan_Amount__c.value;
            this.tenure = data.fields.Loan_Tenure_Months__c.value;
            this.interestRate = data.fields.Interest_Rate__c.value;
            this.status = data.fields.Status__c.value;
        } else if (error) {
            console.error('Error fetching Loan Application:', error);
        }
    }
}
```

- EMISCHEDULER VIEWER




```
<template>
    <lightning-card title="EMI Schedule">
        <lightning-button label="🔄 Refresh" onclick={handleRefresh}></lightning-button>

        <template if:true={emis}>
            <lightning-datatable
                key-field="Id"
                data={emis}
                columns={columns}>
            </lightning-datatable>
        </template>

        <template if:true={error}>
            <p style="color:red">⚠️ {error}</p>
        </template>
    </lightning-card>
</template>
```

emiScheduleViewer.html | emiScheduleViewer.js | emiScheduleViewer.js-meta.xml

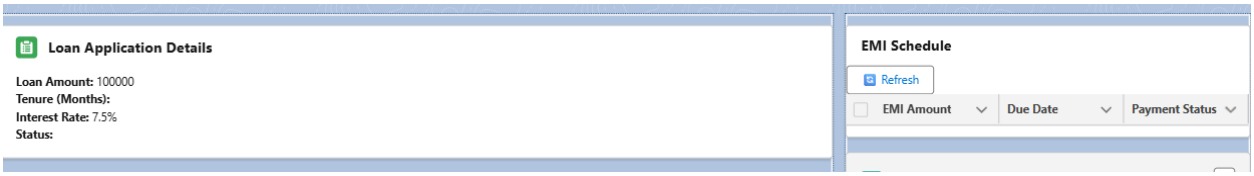
force-app > main > default > lwc > emiScheduleViewer > emiScheduleViewer.js > ...

```
import { LightningElement, api, track, wire } from 'lwc';
import getEmiByLoan from '@salesforce/apex/LoanHelper.getEmiByLoan';

export default class EmiScheduleViewer extends LightningElement {
    @api recordId;
    @track emis = [];
    @track error;

    columns = [
        { label: 'EMI Amount', fieldName: 'EMI_Amount__c', type: 'currency' },
        { label: 'Due Date', fieldName: 'Due_Date__c', type: 'date' },
        { label: 'Payment Status', fieldName: 'Payment_Status__c', type: 'text' }
    ];

    // Wire → auto fetch
    @wire(getEmiByLoan, { loanId: '$recordId' })
    wiredEmis({ data, error }) {
        if (data) {
            this.emis = data;
        } else if (error) {
            this.error = error;
        }
    }

    // Imperative → refresh button
    handleRefresh() {
        getEmiByLoan({ loanId: this.recordId })
            .then(result => {
                this.emis = result;
                this.error = undefined;
            })
```

emiScheduleViewer.html | emiScheduleViewer.js | emiScheduleViewer.js-meta.xml

force-app > main > default > lwc > emiScheduleViewer > emiScheduleViewer.js-meta.xml > ...

```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>58.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
    </targets>
    <targetConfigs>
        <targetConfig targets="lightning__RecordPage">
            <objects>
                <object>Loan_Application__c</object>
            </objects>
        </targetConfig>
    </targetConfigs>
</LightningComponentBundle>
```



- **Navigation Service**
  - Added record navigation inside LWCs.
  - Example: From EMI Schedule → back to Loan Application.

---

3. Outcomes

- Built **modular Lightning Pages** for CLMS.
- Created **custom LWCs** integrated with Apex.
- Enhanced usability with **Home Page + Utility Bar**.

- Added **auto-refresh (wire)** and **manual refresh (imperative Apex)**.
- Implemented **navigation & events** for smooth user experience.

---

4. Conclusion

Phase 6 delivered a **modern Lightning Experience UI** for CLMS.
Users can now work with loans, EMIs, and approvals in a **single consolidated workspace**.
With LWCs and Apex integration, the UI is **interactive, scalable, and user-friendly**.

Phase 7: Integration & External Access

In this phase, integration capabilities were enabled for the **Corporate Loan Management System (CLMS)** to securely communicate with external systems (Loan Provider APIs, reporting systems, and real-time subscribers). Salesforce features such as **Named Credentials, Callouts, Platform Events, Change Data Capture, and OAuth authentication** were configured to ensure secure, scalable, and real-time access.

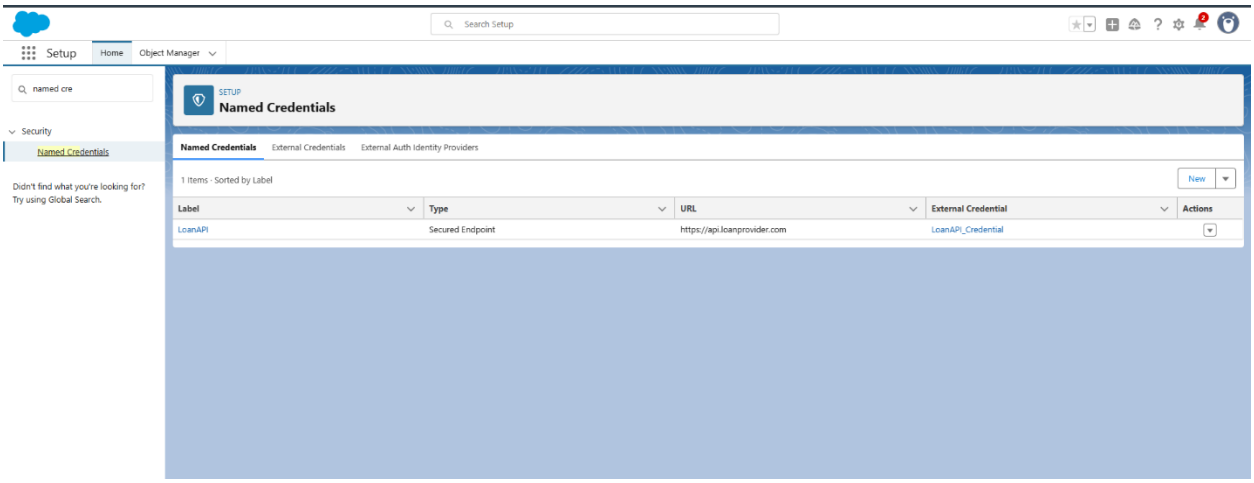
🔹 **Named Credentials**

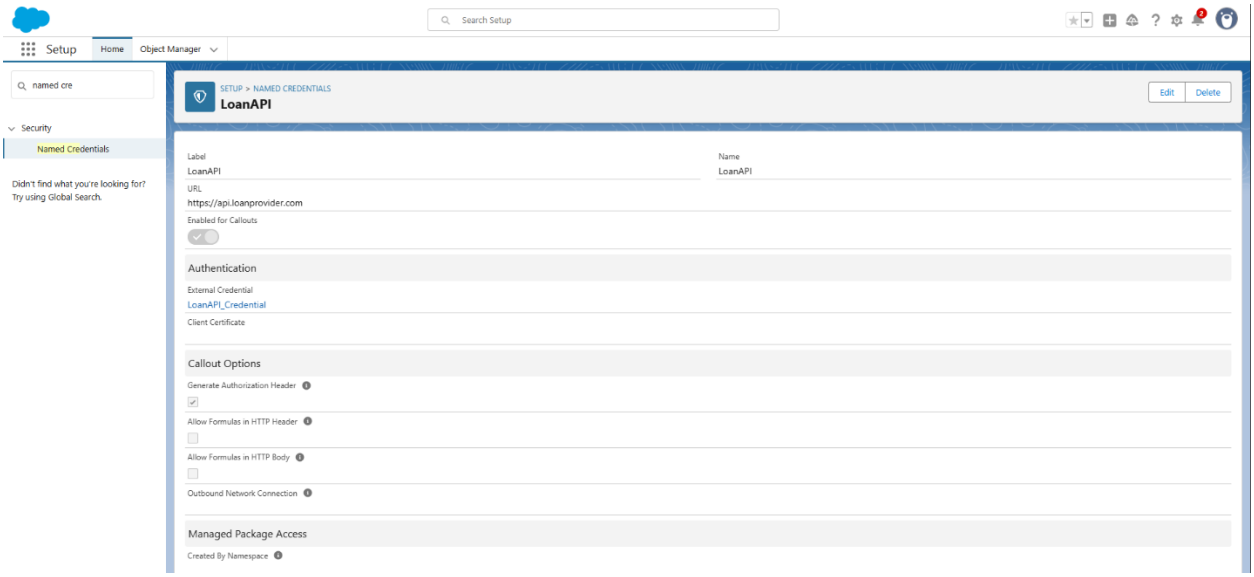
Purpose: Store external API endpoints and authentication securely, avoiding hardcoding credentials in Apex.

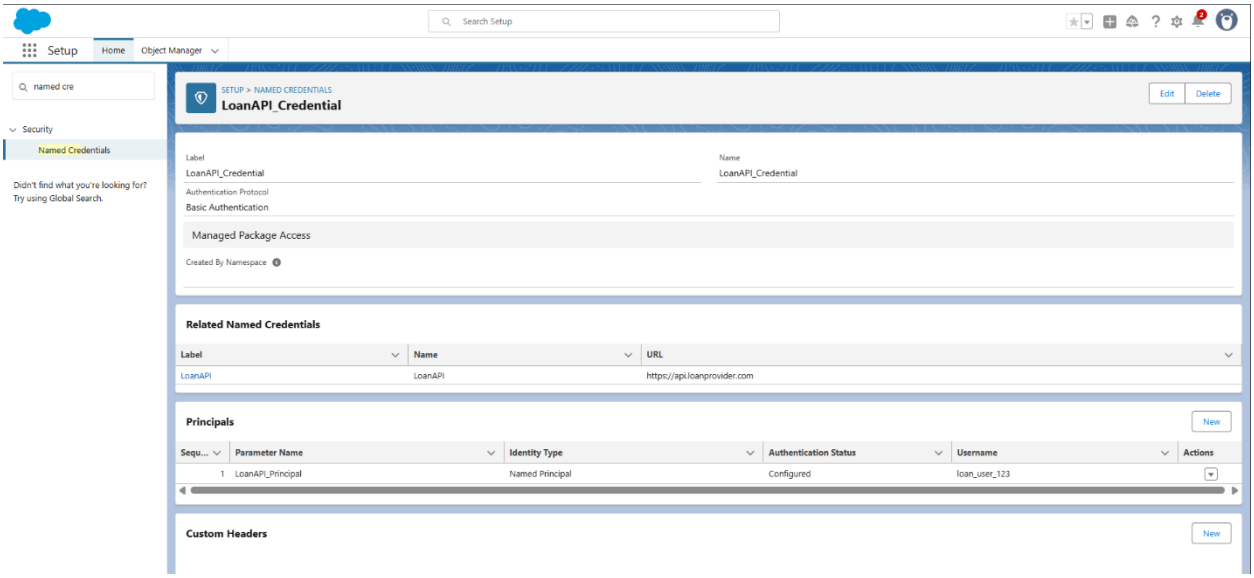
Configuration:

- **Label**: LoanAPI
- **URL**: https://api.loanprovider.com
- **External Credential**: LoanAPI_Credential (with username/password principal).
- **Enabled for Callouts**: ✅

Outcome: Allows CLMS to securely call external Loan Provider APIs for loan validation and processing.


Search Setup
Setup
Home
Object Manager
named cre
Security
Named Credentials
Didn't find what you're looking for?
Try using Global Search.
SETUP
Named Credentials
Named Credentials
External Credentials
External Auth Identity Providers
1 Items · Sorted by Label
New
Label
Type
URL
External Credential
Actions
LoanAPI
Secured Endpoint
https://api.loanprovider.com
LoanAPI_Credential

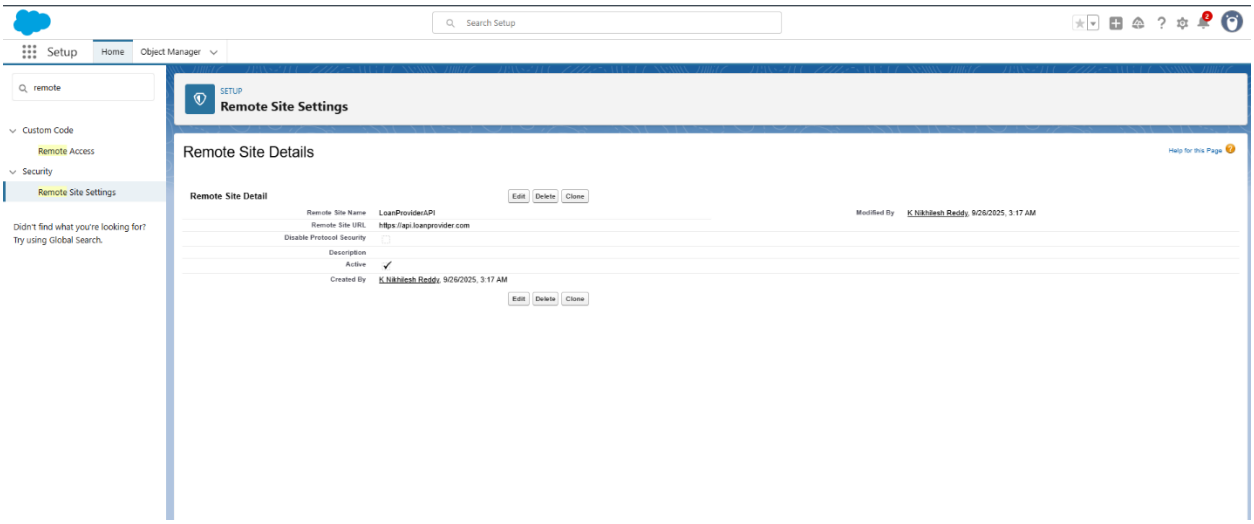

Search Setup
Setup
Home
Object Manager
named cre
Security
Named Credentials
Didn't find what you're looking for?
Try using Global Search.
SETUP > NAMED CREDENTIALS
LoanAPI
Edit
Delete
Label
LoanAPI
Name
LoanAPI
URL
https://api.loanprovider.com
Enabled for Callouts
Authentication
External Credential
LoanAPI_Credential
Client Certificate
Callout Options
Generate Authorization Header
Allow Formulas in HTTP Header
Allow Formulas in HTTP Body
Outbound Network Connection
Managed Package Access
Created By Namespace


Search Setup
Setup
Home
Object Manager
named cre
Security
Named Credentials
Didn't find what you're looking for?
Try using Global Search.
SETUP > NAMED CREDENTIALS
LoanAPI_Credential
Edit
Delete
Label
LoanAPI_Credential
Name
LoanAPI_Credential
Authentication Protocol
Basic Authentication
Managed Package Access
Created By Namespace
Related Named Credentials
Label
Name
URL
LoanAPI
LoanAPI
https://api.loanprovider.com
Principals
New
Sequ...
Parameter Name
Identity Type
Authentication Status
Username
Actions
1
LoanAPI_Principal
Named Principal
Configured
loan_user_123
Custom Headers
New

🔹 External Services & Remote Site Settings

- **External Services**: Configured to connect Salesforce with REST endpoints using schema (if available).
- **Remote Site Settings**: Added https://api.loanprovider.com for API access.

Outcome: Ensures Salesforce can make outbound callouts to the loan provider system.

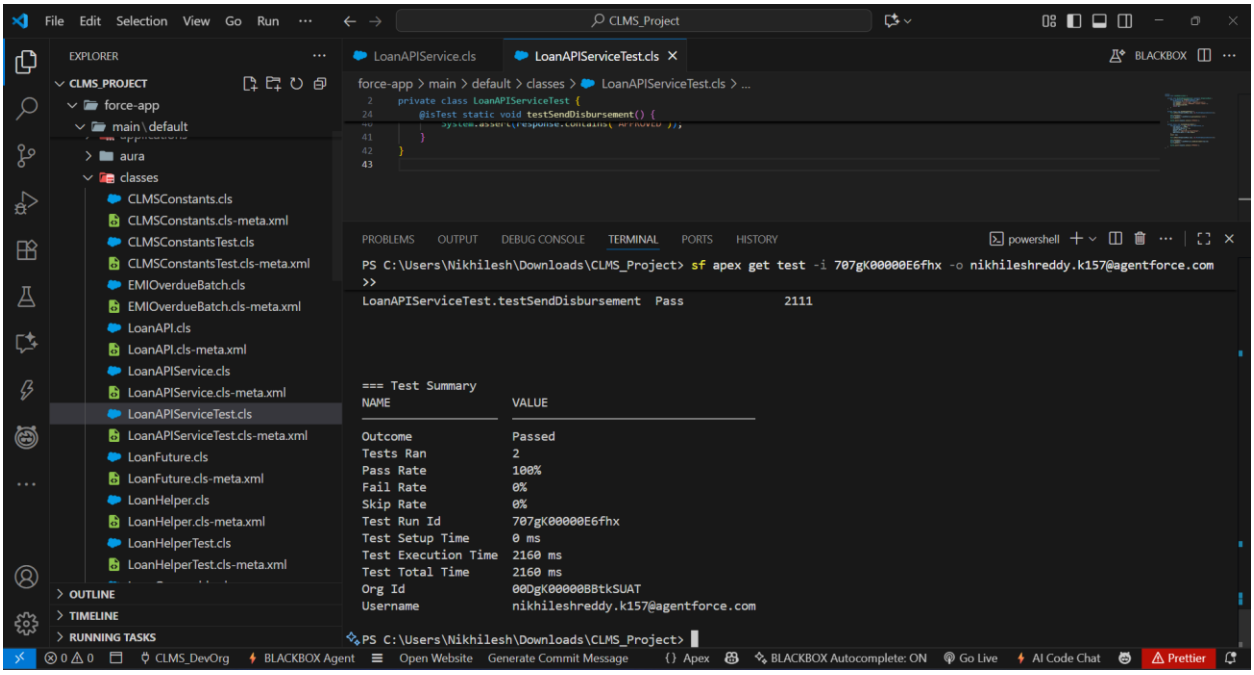


🔹 Apex REST Callouts

Class: LoanAPIService.cls

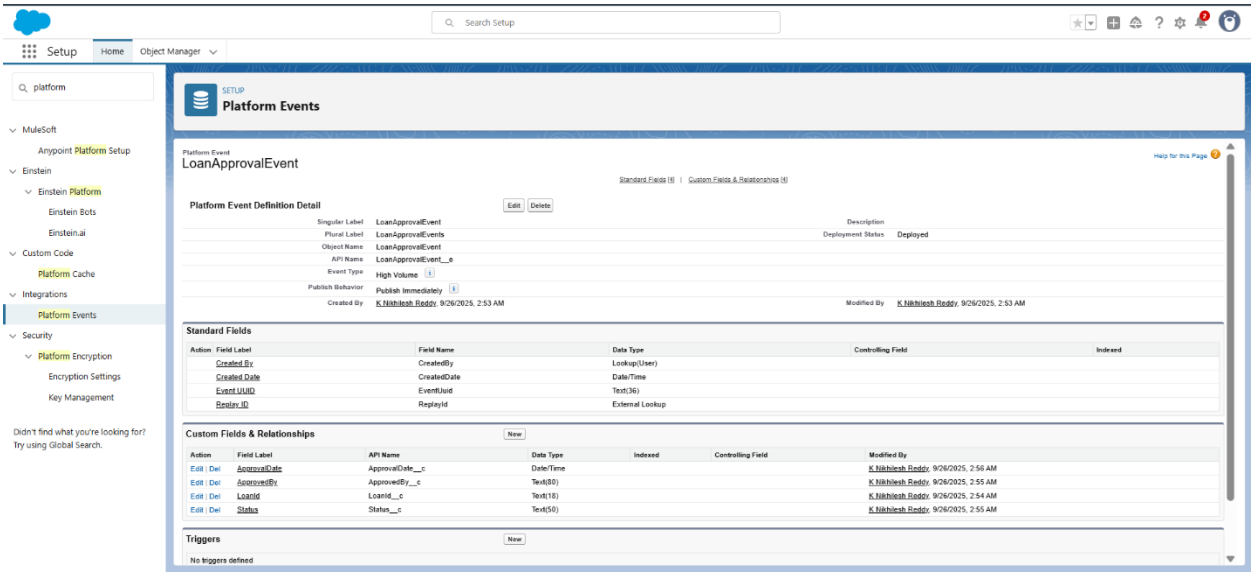
- Apex class created to fetch loan data from external APIs using the Named Credential.

Outcome: CLMS can fetch or validate loan details from external APIs in real-time.



- **Platform Events**
  - **Event Created**: LoanApprovalEvent__e
  - **Fields**: LoanId__c, Status__c, ApprovedBy__c, ApprovalDate__c
  - **Trigger Update**: Publishes event when a loan is approved.

Outcome: External systems (e.g., MuleSoft, CometD subscribers) receive real-time notifications when loans are approved.

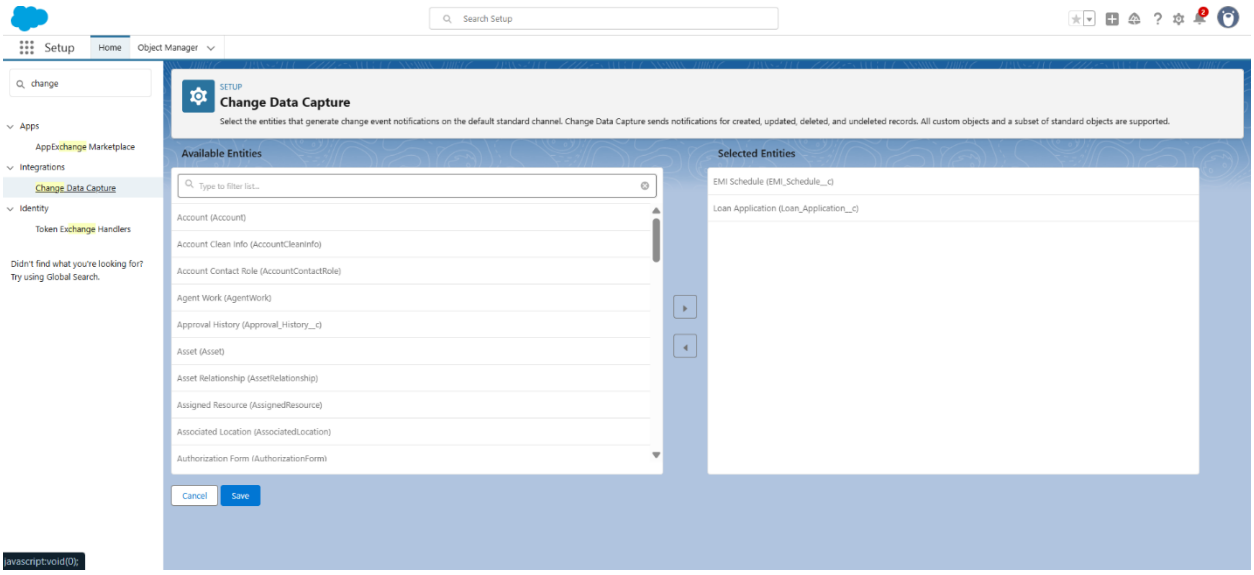


🔹 Change Data Capture (CDC)

Enabled for:

- Loan Application (Loan_Application__c)
- EMI Schedule (EMI_Schedule__c)

Outcome: External systems can subscribe to Salesforce data changes (insert/update/delete), ensuring data synchronization.



🔹 Salesforce Connect

- **Purpose**: For accessing external databases without storing data in Salesforce.
- **Note**: Not implemented yet, but available for future CLMS scaling.

🔹 API Limits

- **Monitored in**: Setup → System Overview → API Usage.
- **Ensured**: Integration designs respect Salesforce daily limits.

- **OAuth & Authentication**
  - **Connected App Created** for CLMS external API access.
  - **Configured**:
    - OAuth Scopes: api, refresh_token
    - Callback URL: External system endpoint

Outcome: External systems can authenticate securely and call Salesforce APIs.

Phase 7 Outcomes

- CLMS now supports secure external integration with Loan Provider APIs.
- Loan approvals trigger **real-time notifications** via Platform Events.
- Data synchronization possible with **Change Data Capture (CDC)**.
- **OAuth** ensures external access is secure and authenticated.
- Integration design respects **Salesforce API limits** for scalability.

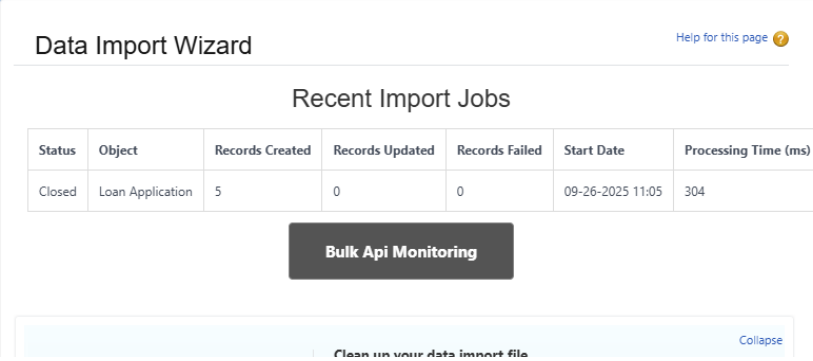
Phase 8: Data Management & Deployment (CLMS Project)

In this phase, **data management and deployment practices** were implemented in the **Corporate Loan Management System (CLMS)**.
The focus was on **importing and exporting data**, preventing duplicates, ensuring secure backups, and deploying metadata from development → production using Salesforce-native tools.

- **1. Data Import Wizard**
  - **What I did**:
    - Used **Data Import Wizard** in Salesforce Setup.
    - Imported **Customer__c** and **Loan_Application__c** records from CSV files.
    - Mapped fields such as Loan_Amount__c, Loan_Tenure_Months__c, and Interest_Rate__c.
  - **Why**:

- To bulk load customer master data and initial loan applications into the system.

- **Screenshot to include**:
  - Data Import Wizard page showing field mapping.

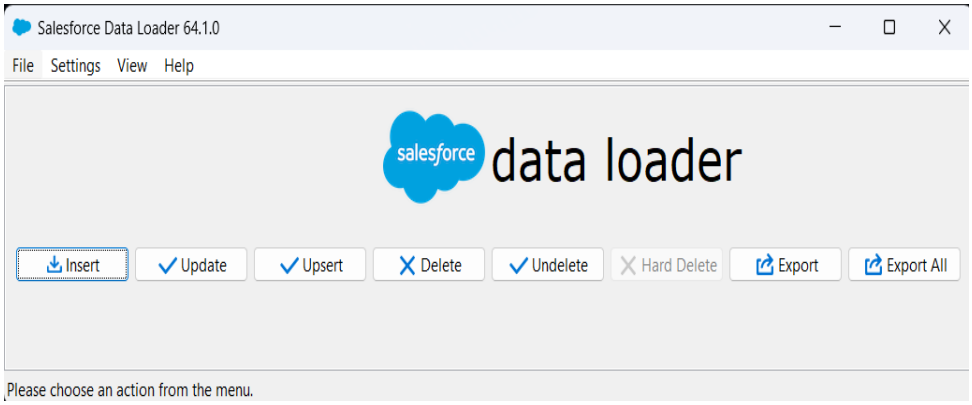


---

2. Data Loader

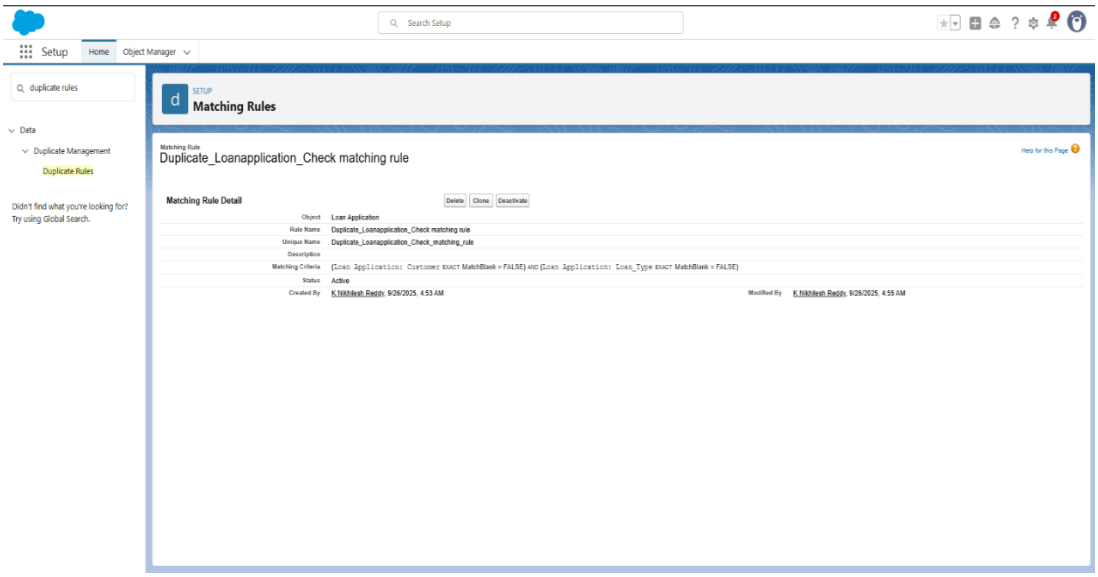
- **What I did**:
  - Installed and logged in with Salesforce **username + password + security token**.
  - Performed **insert** and **update** operations on EMI_Schedule__c and Loan_Application__c.
  - Used **export** feature to back up EMI schedules for validation.

- **Why**:
  - To handle bulk data operations not possible through Import Wizard.



🔹 3. Duplicate Rules

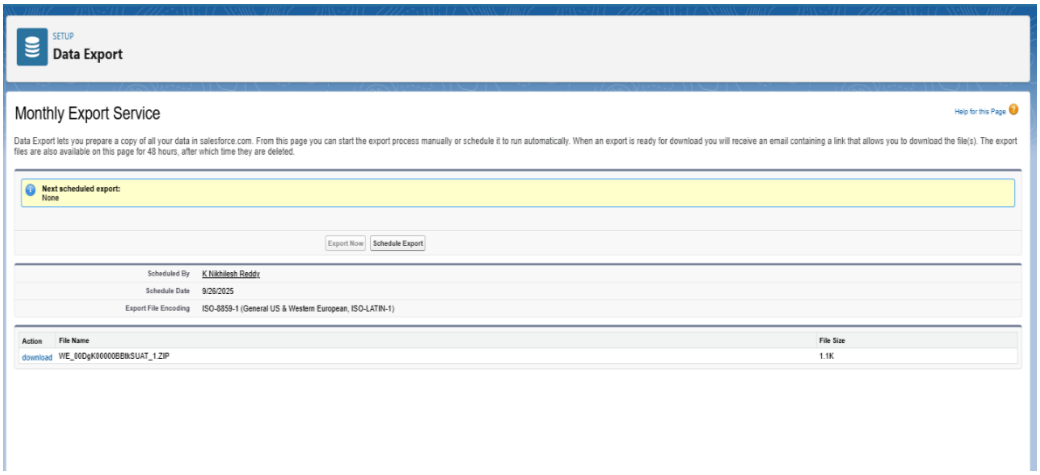
- **What I did**:
  - Created a **Duplicate Rule** for LoanApplication__c.
  - Matching Criteria → **Customer** and **Loan Type**.
  - Action → **Block** duplicate creation.
- **Why**:
  - To ensure no customer is added twice with the same contact details.



🔹 4. Data Export & Backup

- **What I did**:
  - Scheduled **weekly data exports** in Salesforce Setup.
  - Selected objects: Loan_Application__c, Customer__c, EMI_Schedule__c.
  - Configured export to email ZIP backups.
- **Why**:

- To maintain secure backups of all loan and customer data.

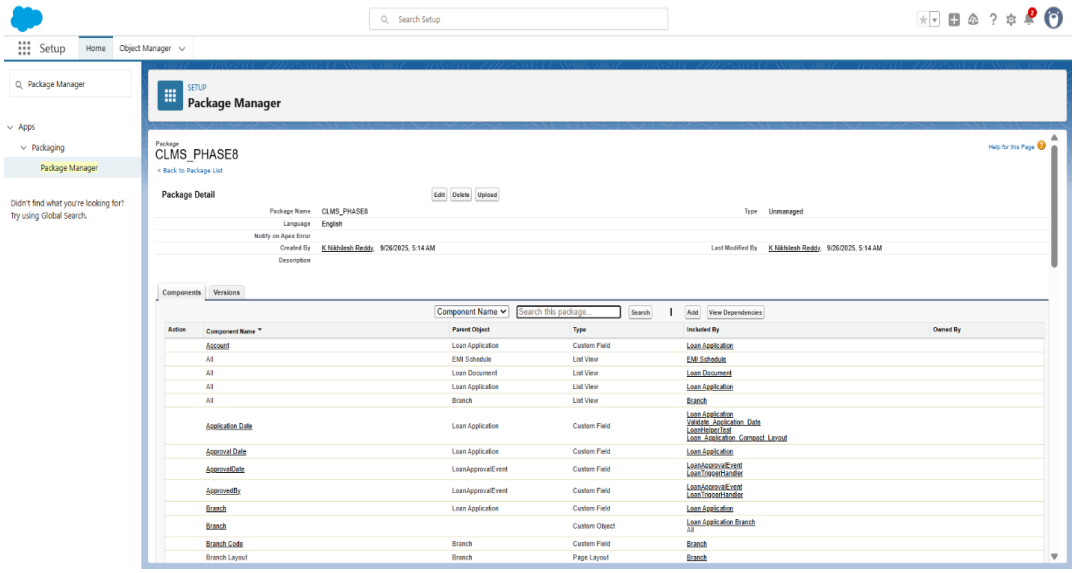


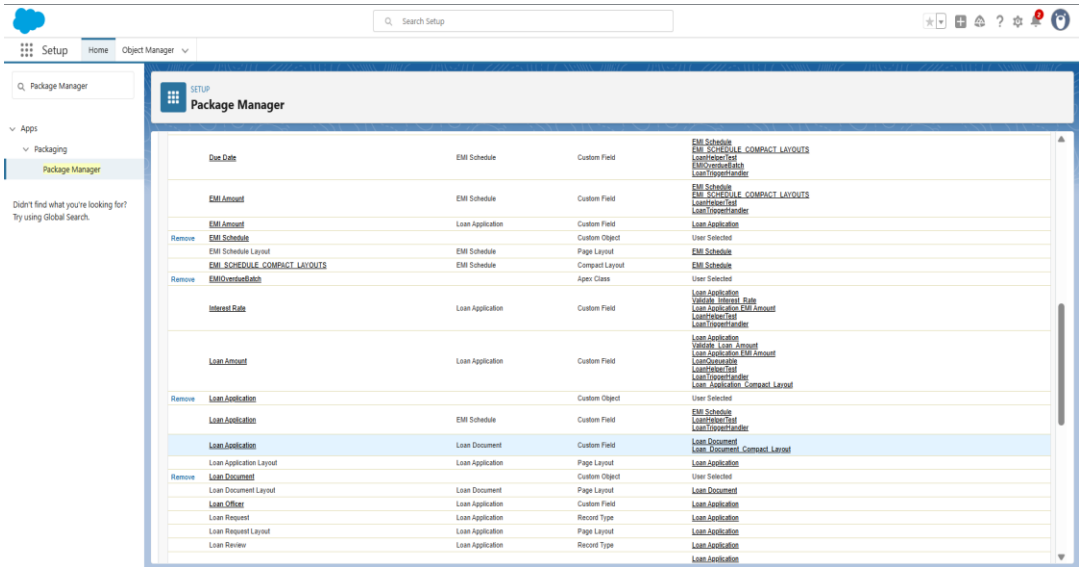
🔹 5. Change Sets

- **What I did**:
  - Created an **Outbound Change Set** in Sandbox.
  - Added metadata: LoanApplicationTrigger, LoanHelper class, EMIOverdueBatch, LWC components.
  - Uploaded and deployed the Change Set to target org.
- **Why**:
  - To move tested metadata from development to production.

🔹 6. Packages

- **What I did**:
  - Explored **Unmanaged Packages**.
  - Created a package for CLMS metadata to simplify deployments.
- **Why**:
  - To bundle components together for sharing and deployment.



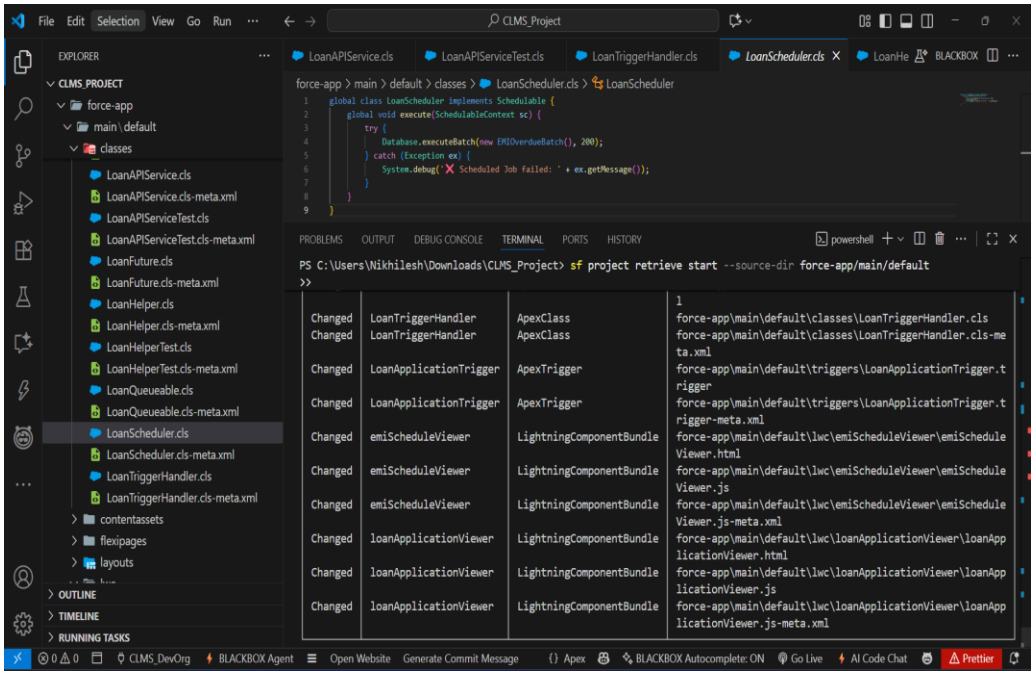


- **7. VS Code & SFDX (Modern Deployment)**
  - **What I did**:
    - Used **VS Code + Salesforce CLI (SF)** for deployments.
    - Commands executed:
    - # Deploy a single class
    - sf project deploy start --source-dir force-app/main/default/classes/LoanHelper.cls
    -

- # Deploy a trigger
- sf project deploy start --source-dir force-app/main/default/triggers/LoanApplicationTrigger.trigger
-
- # Deploy all metadata

sf project deploy start --source-dir force-app/main/default

- **Why**:
  - This is the most modern and recommended approach for Salesforce deployments.



✅ Phase 8 Outcomes

- Customer and Loan data successfully imported into Salesforce.
- Bulk EMI records managed via **Data Loader**.
- Duplicate prevention established for Customer records.
- Weekly data export/backup configured for Loan, EMI, and Customer data.
- Metadata deployments done via **Change Sets** and **VS Code/SFDX**.

- Packages explored for bundling CLMS components.
- CLMS system is now **data-ready, secure, and production-deployable**.

Phase 9: Reporting, Dashboards & Security Review – CLMS Project

This phase focuses on **building reports, dashboards, and implementing security controls** in the Customer Loan Management System (CLMS).

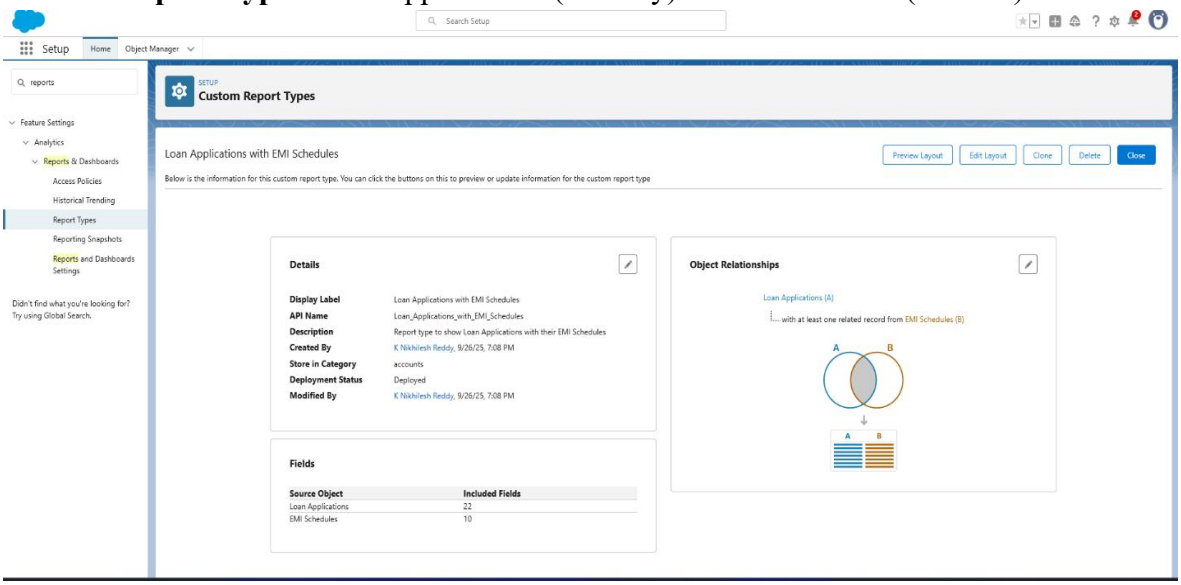
📊 Reports

We used 4 types of reports in Salesforce:

1. **Tabular Report** – Simple list of loan applications.
    - Example: All active loans with Customer Name, Loan Amount, Status.
2. **Summary Report** – Grouped and summarized report.
    - Example: Total loan amount grouped by Branch.
3. **Matrix Report** – Comparison of two dimensions.
    - Example: Loan Type vs Loan Approval Status.
4. **Joined Report** – Combined data from multiple objects.
    - Example: Customer + Loan Application + Repayment in one report.

📑 Report Types

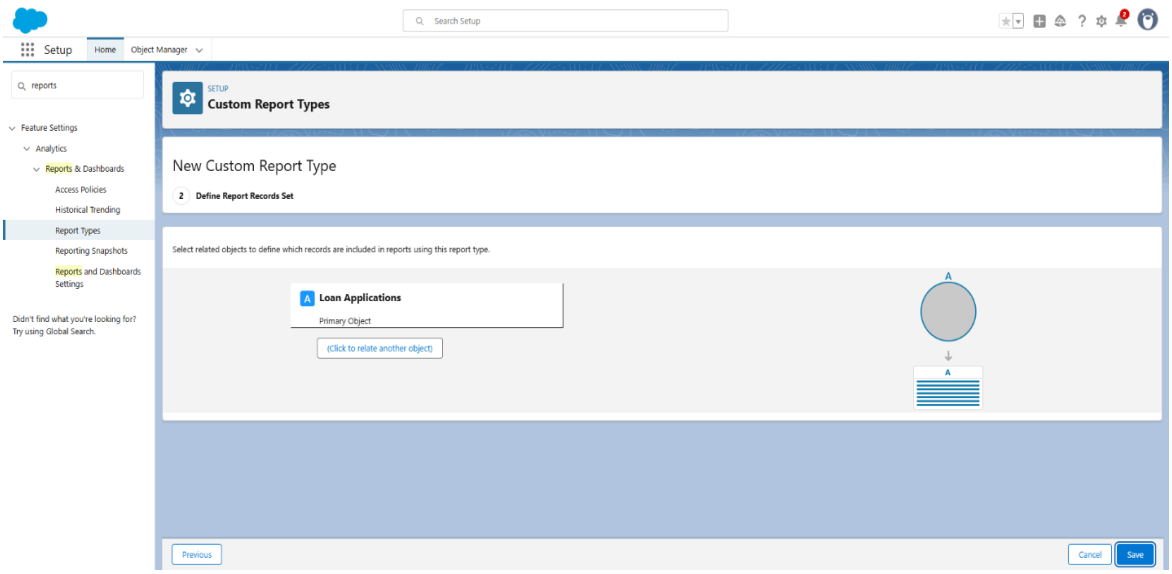
- **Custom Report Type**: Loan Applications (Primary) + Emi schedules (Related).



📈 Dashboards

We built dashboards using the above reports:

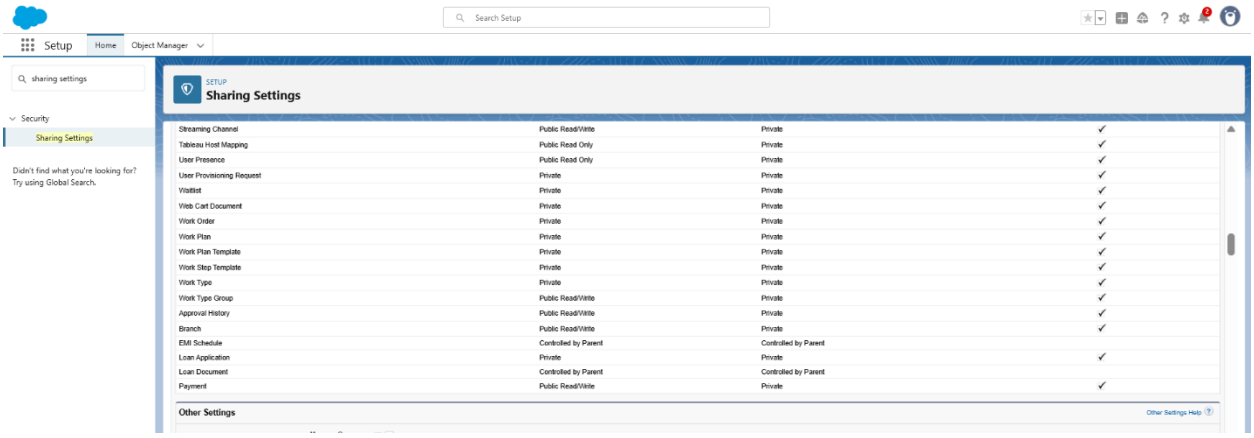
- Loan Status Pie Chart.
- Branch-wise Loan Volume Bar Chart.
- Repayment Trend Line Chart.
- Top Customers by Loan Amount Table.



🔐 Security Review

Sharing Settings

- Customer__c: Private.
- Loan_Application__c: Controlled by Parent.
- Repayment__c: Private.
- Role Hierarchy: Admin → Branch Manager → Loan Officer.
- Sharing Rules for Risk Team to view all loans.



Field Level Security (FLS)

- Sensitive fields (Credit Score, Salary) restricted to relevant profiles.

Session Settings

- Session timeout: 20 mins inactivity.
- Browser security settings enabled.
- Concurrent login restrictions applied.

Login IP Ranges

- Branch-specific IP ranges for Loan Officers.
- Extended IP range for Admin users.

Audit Trail

- Setup Audit Trail enabled.
- Logs downloaded for compliance (last 6 months of admin actions).

Deliverables

- Reports (Tabular, Summary, Matrix, Joined).
- Custom Report Types.
- Dashboards with KPIs.
- Dynamic Dashboards for role-based visibility.
- Security settings (Sharing, FLS, Session, IP, Audit Trail).

Deployment: