

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.

The screenshot shows the 'Company Information' page in the Salesforce setup. The organization's profile is displayed, including its name, address, and various configuration settings. Key details include:

- Organization Name:** Corporate Loan Management System (CLMS)
- Primary Contact:** K Nikhilesh Reddy
- Address:** Tirupati, Andhra Pradesh, India
- Fiscal Year Starts In:** Custom Fiscal Year
- Default Locale:** Hindi (India)
- Default Language:** English
- Default Time Zone:** (GMT+05:30) India Standard Time (Asia/Kolkata)
- Currency Locale:** English (United States) - USD
- Used Data Space:** 346 KB (7%)
- Used File Space:** 17 KB (0%)
- API Requests, Last 24 Hours:** 0 (15,000 max)
- Streaming API Events, Last 24 Hours:** 0 (10,000 max)
- Restricted Logins, Current Month:** 0 (0 max)
- Salesforce.com Organization ID:** 00DgK00000BBtks
- Organization Edition:** Developer Edition
- Instance:** CAN96

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

The screenshot shows the 'Business Hours' setup page under 'SETUP'. It displays the 'Business Hours Detail' section where business hours are defined for each day of the week. The 'CLMS Business Hours' section shows specific times for Monday through Friday, while Saturday and Sunday are listed as 'No Hours'. A 'Time Zone' dropdown is set to '(GMT+05:30) India Standard Time (Asia/Kolkata)'. Below this, the 'Business Hours' section is labeled 'Active' with a checked checkbox. The 'Created By' field shows 'K.Nikhilesh Reddy 9/12/2025, 3:58 AM'. The 'Last Modified By' field shows 'K.Nikhilesh Reddy 9/12/2025, 3:58 AM'. A 'Holidays' section lists three entries: 'Dwali' (Date: 10/18/2025 All Day), 'Independence Day' (Date: 8/15/2026 All Day), and 'Republic Day' (Date: 1/26/2026 All Day). The 'Help for this Page' link is located in the top right corner.

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.

The screenshot shows the 'Fiscal Year' setup page under 'SETUP'. It displays the 'Custom Fiscal Years' section, which lists a single entry: 'Action' (Edit), 'Year' (2026), 'FY Start Date' (10/1/2025), 'FY End Date' (9/30/2026), and 'Description' (empty). A 'New' button is available to add a new fiscal year. The 'Help for this Page' link is located in the top right corner.

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_manager1@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.

The screenshot shows the Salesforce Setup - Users page. At the top, there's a header with a user icon, 'SETUP', and 'Users'. Below it, a sub-header says 'All Users'. A note says 'On this page you can create, view, and manage users.' and 'To get more licenses, use the Your Account app. [Let's Go](#)'. A 'View' dropdown is set to 'All Users'. The main area is a table with columns: Action, Full Name, Alias, Username, Role, Active, and Profile. The table lists several users with their details and assigned profiles like 'Chatter Free User', 'System Administrator', etc.

Action	Full Name	Alias	Username	Role	Active	Profile
Edit	Chatter Expert	Chatter	chatty.00dgk00000bbtksuat.w0skin3xigh@chatter.salesforce.com		✓	Chatter Free User
Edit	EPIC_OrgFarm	OEPIIC	epic.cada3e78a5be@orgfarm.salesforce.com		✓	System Administrator
Edit	Manager_Branch	branch	branch_manager1@clms.com	Branch Manager	✓	Branch Manager Profile
Edit	Manager_Credit	credit	credit_manager1@clms.com	Credit Manager	✓	Standard Platform User
Edit	officer_finance	officer	officer.finance1@clms.com	Finance officer	✓	Standard Platform User
Edit	Officer_Loan	loan	loan.officer1@clms.com	Loan Officer	✓	Loan Officer Profile
Edit	Reddy_K_Nikhlesh	nik	nikhileshreddy.k157@agentforce.com		✓	System Administrator
Edit	User_Integration	integ	integration@00dgk00000bbtksuat.com		✓	Analytics Cloud Integration User
Edit	User_Security	sec	insightssecurity@00dgk00000bbtksuat.com		✓	Analytics Cloud Security User

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

[Branch Manager Profile](#)

[manager credit](#)

[officer finance](#)

[Loan Officer Profile](#)

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

The screenshot shows the Salesforce Setup interface under the 'Roles' section. At the top, there's a blue header bar with the 'SETUP' button and a user icon. Below it, the word 'Roles' is displayed in a large, bold, dark blue font. A sub-header 'Creating the Role Hierarchy' is followed by a descriptive text: 'You can build on the existing role hierarchy shown on this page. To insert a new role, click Add Role.' A horizontal line separates this from the main content. The main content is titled 'Your Organization's Role Hierarchy' and includes 'Collapse All' and 'Expand All' buttons. A tree view shows the following hierarchy:

- Corporate Loan Management System (CLMS)
 - Add Role
 - CEO** Edit | Del | Assign
 - Add Role
 - Branch Manager** Edit | Del | Assign
 - Add Role
 - Credit Manager** Edit | Del | Assign
 - Add Role
 - Loan Officer** Edit | Del | Assign
 - Add Role
 - Finance officer** Edit | Del | Assign
 - Add Role
 - SVP, Customer Service & Support** Edit | Del | Assign
 - Add Role
 - SVP, Human Resources** Edit | Del | Assign
 - Add Role
 - SVP, Sales & Marketing** Edit | Del | Assign
 - Add Role

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

The screenshot shows the 'Login Access Policies' page in the Salesforce Setup. At the top, there's a 'SETUP' button and a 'Login Access Policies' title. Below that, a section titled 'Login Access Policies' with the sub-instruction 'Control which support organizations your users can grant login access to.' Underneath is a 'Manage Support Options' section with a 'Save' and 'Cancel' button. It lists a setting: 'Administrators Can Log in as Any User' with the 'Enabled' checkbox checked. There's also a table for 'Support Organization' with rows for 'Salesforce.com Support' and another row that is partially visible. At the bottom of the page are 'Save' and 'Cancel' buttons.

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

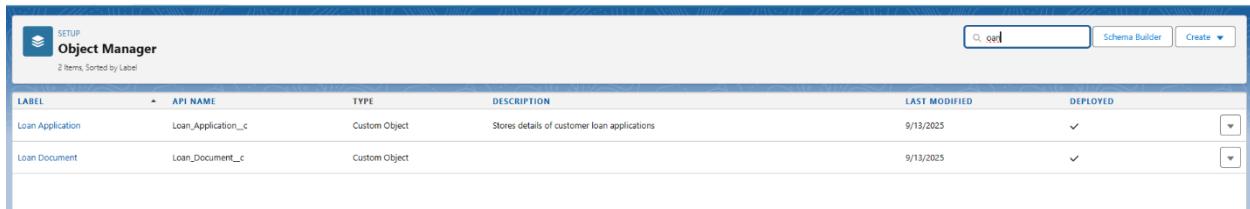
Standard Objects

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

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Account	Account	Standard Object			
Contact	Contact	Standard Object			

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).



The screenshot shows the Salesforce Object Manager interface. At the top, there's a header with 'SETUP' and 'Object Manager'. Below it, a search bar with placeholder 'Search' and buttons for 'Schema Builder' and 'Create'. The main area displays a table with two rows:

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Loan Application	Loan_Application__c	Custom Object	Stores details of customer loan applications	9/13/2025	✓
Loan Document	Loan_Document__c	Custom Object		9/13/2025	✓

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

Fields & Relationships					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Account	Account__c	Lookup(Account)		✓
Lightning Record Pages	Application Date	Application_Date__c	Date		
Buttons, Links, and Actions	Branch	Branch__c	Lookup(Branch)		✓
Compact Layouts	Created By	CreatedById	Lookup(User)		
Field Sets	Customer	Customer__c	Lookup(Contact)		✓
Object Limits	Customer Name	Customer_Name__c	Text(255)		
Record Types	EMI Amount	EMI_Amount__c	Formula(Currency)		
Related Lookup Filters	Interest Rate	Interest_Rate__c	Percent(5, 2)		
Restriction Rules	Last Modified By	LastModifiedById	Lookup(User)		
Scoping Rules	Loan Amount	Loan_Amount__c	Currency(18, 0)		
Object Access	Loan Number (Text or Auto Number)	Name	Auto Number		✓
Triggers	Loan Officer	Loan_Officer__c	Lookup(User)		✓
Flow Triggers					
Validation Rules					
Conditional Field Formatting					

EMI Schedule Fields

- EMI Amount
- Due Date
- Payment Status

Fields & Relationships					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Created By	CreatedById	Lookup(User)		
Lightning Record Pages	Due Date	Due_Date__c	Date		
Buttons, Links, and Actions	EMI Amount	EMI_Amount__c	Currency(16, 2)		
Compact Layouts	EMI Schedule Name	Name	Auto Number		✓
Field Sets	Last Modified By	LastModifiedById	Lookup(User)		
Object Limits	Loan Application	Loan_Application__c	Master-Detail(Loan Application)		
Record Types	Payment Status	Payment_Status__c	Picklist		
Related Lookup Filters					
Restriction Rules					
Scoping Rules					

Loan Documents Fields

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

- Created

By

Fields & Relationships					
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED	
Created By	CreatedById	Lookup(User)			
Document Name	Document_Name__c	Text(255)			
Document Type	Document_Type__c	Picklist			
Last Modified By	LastModifiedById	Lookup(User)			
Loan Application	Loan_Application__c	Master-Detail(Loan Application)			
Loan Document Name	Name	Auto Number			

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)

Fields & Relationships					
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED	
Approval History Name	Name	Auto Number			
Approver Role	Approver_Role__c	Picklist			
Comments	Comments__c	Long Text Area(32768)			
Created By	CreatedById	Lookup(User)			
Date of Action	Date_of_Action__c	Date/Time			
Decision	Decision__c	Picklist			
Last Modified By	LastModifiedById	Lookup(User)			
Loan Application	Loan_Application__c	Lookup(Loan Application)			
Owner	OwnerId	Lookup(User/Group)			

3. Record Types

- Personal Loan
- Business Loan

- **Home**

Loan

The screenshot shows the Salesforce Object Manager interface for the 'Loan Application' object. The left sidebar has a 'Record Types' section selected. The main area displays two record types: 'Loan Request (for Loan Officers)' and 'Loan Review'. The 'Loan Request (for Loan Officers)' record type is active and was modified by K.Nikhilesh Reddy on 9/19/2025, 4:48 AM. The 'Loan Review' record type is also active and was modified by K.Nikhilesh Reddy on 9/19/2025, 4:57 AM.

RECORD TYPE LABEL	DESCRIPTION	ACTIVE	MODIFIED BY
Loan Request (for Loan Officers)		✓	K.Nikhilesh Reddy, 9/19/2025, 4:48 AM
Loan Review		✓	K.Nikhilesh Reddy, 9/19/2025, 4:57 AM

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.

The screenshot shows the Salesforce Object Manager interface for the 'Loan Application' object. The left sidebar has a 'Page Layouts' section selected. The main area displays two page layouts: 'Loan Application Layout' and 'Loan Request Layout'. Both were created by K.Nikhilesh Reddy. The 'Loan Application Layout' was created on 9/12/2025, 5:05 AM, and the 'Loan Request Layout' was created on 9/12/2025, 9:39 AM.

PAGE LAYOUT NAME	CREATED BY	MODIFIED BY
Loan Application Layout	K.Nikhilesh Reddy, 9/12/2025, 5:05 AM	K.Nikhilesh Reddy, 9/19/2025, 4:40 AM
Loan Request Layout	K.Nikhilesh Reddy, 9/12/2025, 9:39 AM	K.Nikhilesh Reddy, 9/19/2025, 4:40 AM

5. Compact Layouts

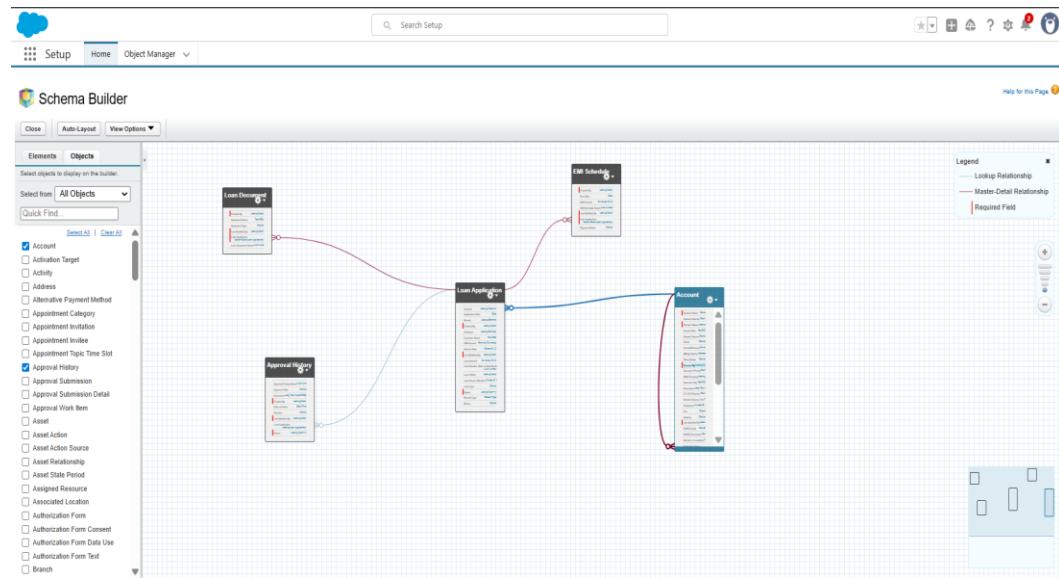
- **Loan Application Compact Layout** → Loan Number, Customer Name, Loan Amount, Loan Type, Status, Application Date.
- **EMI Schedule Compact Layout** → EMI Amount, Due Date, Status.
- **Loan Documents Compact Layout** → Loan Document Name, Document Type, Loan Application, Created By.

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

LABEL	API NAME	PRIMARY	MODIFIED BY	LAST MODIFIED
System Default	SYSTEM		K Nikhilash Reddy	9/19/2025, 4:19 AM
"Loan Application Compact Layout"	Loan_Application_Compact_Layout	✓	K Nikhilash Reddy	9/19/2025, 4:19 AM

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:

Loan Application Validation Rules

The screenshot shows the validation rules for the Loan Application object in the Salesforce Setup interface.

Validation Rules

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

Validation Rule Detail

Rule Name	Validate_Application_Date	Active	✓
Error Condition Formula	OR(ISBLANK(Application_Date__c), Application_Date__c > TODAY())	Error Location	Application Date
Error Message	Application Date cannot be in the future.	Modified By	K Nikhilesh Reddy, 9/19/2025, 11:11 AM
Description			
Created By	K Nikhilesh Reddy, 9/19/2025, 11:11 AM		

Loan Application Validation Rule

Validation Rule Detail

Rule Name	Validate_Interest_Rate	Active	✓
Error Condition Formula	OR(ISBLANK(Interest_Rate__c), Interest_Rate__c < 5, Interest_Rate__c > 30)	Error Location	Interest Rate
Error Message	Interest Rate must be between 5% and 30%.	Modified By	K Nikhilesh Reddy, 9/19/2025, 11:07 AM
Description			
Created By	K Nikhilesh Reddy, 9/19/2025, 11:07 AM		

Loan Application Validation Rule

Validation Rule Detail

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

Loan Application Validation Rule

[Back to Loan Application](#)

Validation Rule Detail	
Rule Name	Validate_Loan_Tenure
Error Condition Formula	Loan_Tenure_Months__c < 6
Error Message	Loan Tenure must be at least 6 months.
Description	
Created By	K.Nikhilesh Reddy, 9/19/2025, 11:02 AM
Modified By	K.Nikhilesh Reddy, 9/19/2025, 11:08 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:

Workflow Rules

All Workflow Rules

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.

View: [All Workflow Rules](#) [Create New View](#)

Quick Tips

- Useful Sample Workflow Rule
- Video Tutorial (English Only)
- Troubleshooting Workflow

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

Workflow Rule

Disburse_Loan_Task

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

Rule Name	Disburse_Loan_Task	Edit Clone Deactivate	
Active	✓	Object	Loan Application
Description		Evaluation Criteria	Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
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](#)

Time-Dependent Workflow Actions [See an example](#)

Workflow Rule Detail

Rule Name	Loan_Approved_Email	Edit Clone Deactivate	
Active	✓	Object	Loan Application
Description		Evaluation Criteria	Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
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](#)

Time-Dependent Workflow Actions [See an example](#)

The screenshot shows the 'Workflow Rules' page in Salesforce. The top navigation bar includes 'SETUP' and 'Workflow Rules'. Below the header, a message encourages users to '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.' A 'Help for this Page' link is also present.

The main section displays the 'Workflow Rule Detail' for 'Set_Approval_Date'. The rule is active and set to trigger on 'Loan Application' when it's created or edited. The evaluation criteria is 'Loan Application: Status EQUALS Approved by Branch Manager'. The rule was created by 'K Nikhilesh Reddy' on 9/19/2025 at 12:02 PM and modified by the same user on 9/19/2025 at 12:19 PM.

The 'Workflow Actions' section shows an immediate field update action named 'SetApprovalDate' on the 'Field Update' type. There is also a note indicating that new time triggers cannot be added to an active 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.

Process Builder

My Processes

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

Try Flow Builder | Use Migrate to Flow Tool Back To Setup Help New

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

The screenshot shows the Salesforce Flow Builder interface. At the top, there's a banner with the message: "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." Below the banner, the title "Process Builder" and "My Processes" are visible. A table lists two processes: "Loan Approval Process" and "Loan Status Notification Process". The "Loan Approval Process" is selected, and its details are shown below the table. The flow diagram for the "Loan Approval Process" starts with a "START" state, followed by a "Loan Application" action. This leads to a decision diamond "Loan Approved". If the answer is "TRUE", it leads to an "IMMEDIATE ACTIONS" section containing "Create EMI Schedule" and "+ Add Action", which then leads to a "STOP" state. If the answer is "FALSE", it leads to another decision diamond "+ Add Criteria". If this second decision is "TRUE", it leads to another "IMMEDIATE ACTIONS" section and a "STOP" state. If it is "FALSE", it leads directly to a final "STOP" state.

◆ 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:

The screenshot shows the 'Approval Processes' page in Salesforce. The top navigation bar includes 'SETUP' and the current page title 'Approval Processes'. Below the title, the specific process is identified as 'Loan Application: Loan Approval Process' with a link to 'Back to Approval Process List'. The main content area is titled 'Process Definition Detail' for the 'Loan Approval Process'. It displays various configuration settings:

- Process Name:** Loan Approval Process
- Unique Name:** Loan_Approval_Process
- Description:** Loan Application: Status EQUALS Pending
- Entry Criteria:** Administrator ONLY
- Record Editability:** Allow Submitters to Recall Approval Requests (unchecked)
- Approval Assignment Email Template:** Loan Application Owner
- Initial Submitters:** Loan Application Owner
- Created By:** K.Nikhilesh Reddy, 9/22/2025, 4:00 AM
- Modified By:** K.Nikhilesh Reddy, 9/22/2025, 4:14 AM

Below this, there are sections for 'Initial Submission Actions', 'Approval Steps', 'Final Approval Actions', and 'Final Rejection Actions', each containing a list of actions with their descriptions and assigned approvers.

◆ 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:

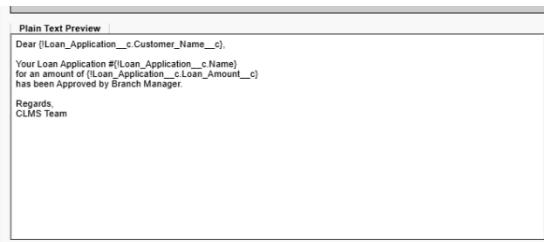
The screenshot displays two main sections of the Salesforce Setup interface:

Email Alerts:

- Email Alert Detail:** Shows details for an email alert named "Loan_Approval_Email". Description: "Email notification when Loan Application is approved by Branch Manager". From Email Address: "Current User's email address". Recipients: "User_Loan Officer". Created By: "K.Nikhilesh Reddy" on 9/19/2025, 11:54 AM. Last Modified By: "K.Nikhilesh Reddy" on 9/19/2025, 11:54 AM.
- Rules Using This Email Alert:** Lists a rule named "Loan_Approved_Email" associated with "Loan Application" objects.
- Approval Processes Using This Email Alert:** States that this alert is currently not used by any approval processes.
- Entitlement Processes Using This Email Alert:** States that this alert is currently not used by any entitlement processes.
- Flows Using This Email Alert:** Shows a flow named "Flow Name" with version "Version".

Classic Email Templates:

- Unfiled Public Classic Email Templates:** Shows a single template named "Loan_Approval_Email" which is an "HTML" template available for use.
- Classic Email Template Availability:** A grid showing the availability of the template across various objects (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 the last modified date (9/19/2025).



◆ 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:

Action	Name	Field to Update	Operation	Value	Last Modified Date
Edit Del	Changes the case priority to high.	Case Priority	Value	High	9/9/2025
Edit Del	Set Approval Date	Loan Application: Approval Date	Formula	TODAY()	9/19/2025
Edit Del	Set Status Approved by Branch Manager	Loan Application: Status	Value	Approved by Branch Manager	9/22/2025
Edit Del	Set Status Rejected	Loan Application: Status	Value	Rejected	9/22/2025

◆ 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:

The screenshot shows the Salesforce Setup Tasks page. At the top, there's a header with a gear icon labeled "SETUP" and "Tasks". Below the header, the title "Task: Disburse Loan Amount" is displayed. Underneath the title, there's a "Workflow Task Detail" section with fields for Object (Loan Application), Assigned To (User - finance officer), Subject (Disburse Loan Amount), Unique Name (Disburse_Loan_Amount), Due Date (Rule Trigger Date + 1 days), and Comments (K Nikhilesh Reddy, 9/19/2025, 12:28 PM). The status is Not Started and Priority is Normal. The "Modified By" field shows K Nikhilesh Reddy, 9/22/2025, 6:12 AM. Below this, there are buttons for Edit, Delete, and Clone.

Below the Workflow Task Detail, there are three sections: "Rules Using This Task", "Approval Processes Using This Task", and "Entitlement Processes Using This Task". Each section has a table with columns for Action, Rule Name, Description, Object, and Status (Active with a checkmark). The "Rules Using This Task" section shows one rule named "Disburse_Loan_Task". The "Approval Processes Using This Task" and "Entitlement Processes Using This Task" sections both show a message stating "This task is currently not used by any approval processes" and "This task is currently not used by any entitlement processes".

At the bottom left, there's a link to "Back To Top". At the bottom right, there's a link to "Always show me more records per related list".

◆ 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:

Custom Notifications

When you create and use custom notifications, the title and body of the custom push notification may be saved to and processed by Google, Microsoft and/or Apple. Salesforce is not responsible for the privacy and security practices of third-party systems or applications like Google Cloud Messaging or Apple Push Notification Service.

Custom Notification Types

Send custom notifications using Flows or Process Builder

New

NOTIFICATION NAME	API NAME	NAMESPACE	DESKTOP	MOBILE
enablement_coaching_feedback_ready	enablement_coaching_feedback_ready		✓	▼
Loan Status Notification	Loan_Status_Notification		✓	✓

Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with Process Builder—and more! Salesf

Process Builder - Loan Status Notification Process

Expand All | Collapse All

```
graph TD; START([START]) --> LoanApplication[Loan Application]; LoanApplication --> ApprovedReject{Loan Approved/Rejected}; ApprovedReject -- TRUE --> ImmediateActions1["IMMEDIATE ACTIONS<br/>Loan Status Notificat...<br/>+ Add Action"]; ImmediateActions1 --> STOP1([STOP]); ApprovedReject -- FALSE --> AddCriteria{+ Add Criteria}; AddCriteria -- TRUE --> ImmediateActions2["IMMEDIATE ACTIONS<br/>+ Add Action"]; ImmediateActions2 --> STOP2([STOP]); AddCriteria -- FALSE --> STOP3([STOP]);
```

◆ End-to-End Automation Flow

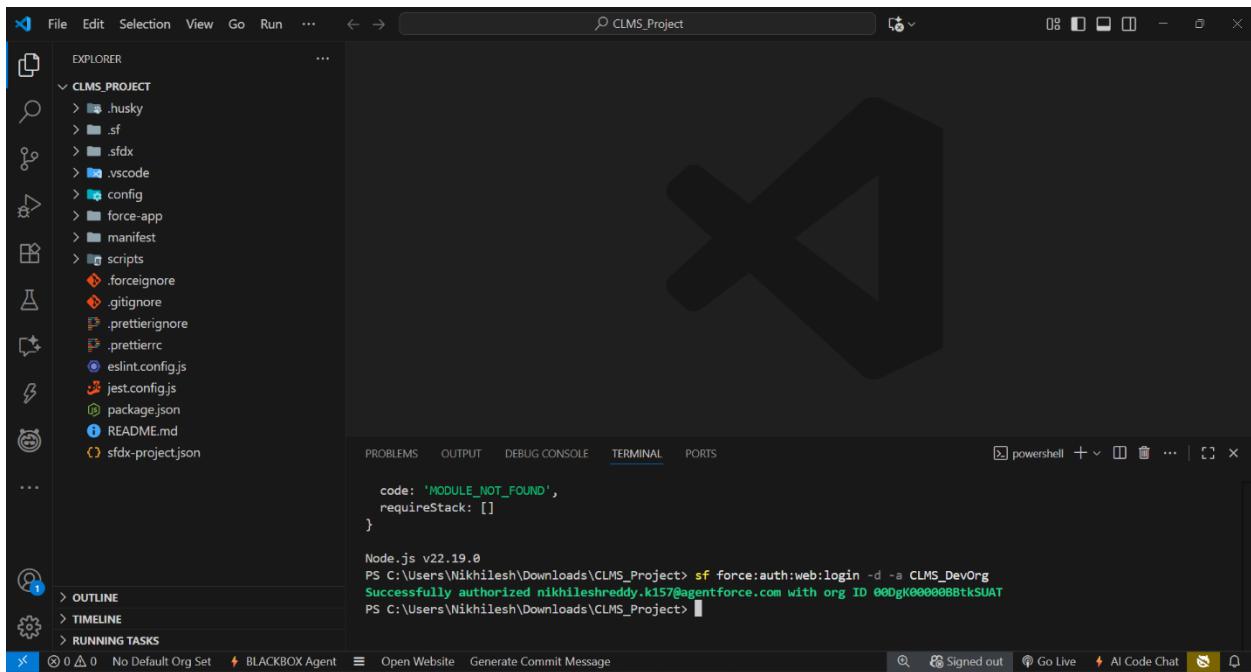
1. Loan Officer submits Loan Application (Status = Pending).

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 hardcoded strings in triggers & classes.

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

- EXPLORER:** Shows the project structure under "CLMS_PROJECT".
- CLMSConstants.cls:** The code editor displays the Apex class definition:

```
1  public with sharing class CLMSConstants {
2      public static final String STATUS_PENDING = 'Pending';
3      public static final String STATUS_APPROVED_BY_CREDIT = 'Approved by Credit Manager';
4      public static final String STATUS_APPROVED_BY_BRANCH = 'Approved by Branch Manager';
5      public static final String STATUS_REJECTED = 'Rejected';
6      public static final String EMI_STATUS_PENDING = 'Pending';
7      public static final String EMI_STATUS_OVERDUE = 'Overdue';
8  }
```

- PROBLEMS:** No problems found.
- OUTPUT:** Shows the deployment command and its success:

```
PS C:\Users\Nikhilesh\Downloads\CLMS_Project> sf project deploy start --source-dir force-app/main/default/classes/CLMSConstants.cls
>>
✓ Done 0ms
```

- STATUS:** Succeeded
- Deploy ID:** 0Afk00000AVYMEAS5
- Target Org:** nikhileshreddy.k157@agentforce.com
- Elapsed Time:** 2.61s
- Deployed Source:** A table showing the deployed files:

State	Name	Type	Path
Created	CLMSConstants	ApexClass	force-app\main\default\classes\CLMSConstants.cls
Created	CLMSConstants	ApexClass	force-app\main\default\classes\CLMSConstants.cls-meta.xml

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.

```

1  public with sharing class LoanHelper {
2      // EMI calculation: return monthly EMI (rounded to 2 decimals)
3      public static Decimal calculateEMI(Decimal loanAmount, Integer tenureMonths, Decimal annualInterestPercent) {
4          if (loanAmount == null || tenureMonths == null || tenureMonths <= 0 || annualInterestPercent == null) {
5              return 0;
6          }
7          Decimal monthlyRate = annualInterestPercent / 1200; // e.g. 12% => 0.01
8          // EMI formula: P * r / (1 - (1 + r)^-n)
9          Decimal denom = 1 - Math.pow(1 + monthlyRate, -tenureMonths);
10         if (denom == 0) return 0;
11         Decimal emi = (loanAmount * monthlyRate) / denom;
12         return emi.setScale(2);
13     }
14 }

```

	Alias	Username	Org Id	Status
CLMS_DevOrg	nikhileshreddy.k157@agentforce.com	00Dgk00000BtkSUAT	Connected	

◆ 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.

CLMS_Project

EXPLORER

- CLMS_PROJECT
 - force-app
 - main\default
 - lwc
 - loanApplicationViewer
 - loanApplicationViewer.js
 - loanApplicationViewer.js-meta.xml
 - objects
 - permissionssets
 - staticresources
 - tabs
 - triggers
 - LoanApplicationTrigger.trigger
 - LoanApplicationTrigger.trigger-meta...
- collectionsPractice.apex

LoanApplicationTrigger.trigger

```
force-app > main > default > triggers > LoanApplicationTrigger.trigger
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.oldMap);
8         }
9     }
10 }
11 }
```

LoanTriggerHandler.cls

CLMSConstantsTest.cls

loanApplicationViewer.

LoanApplicationTrigger.trigger

LoanTriggerHandler.cls

CLMSConstantsTest.cls

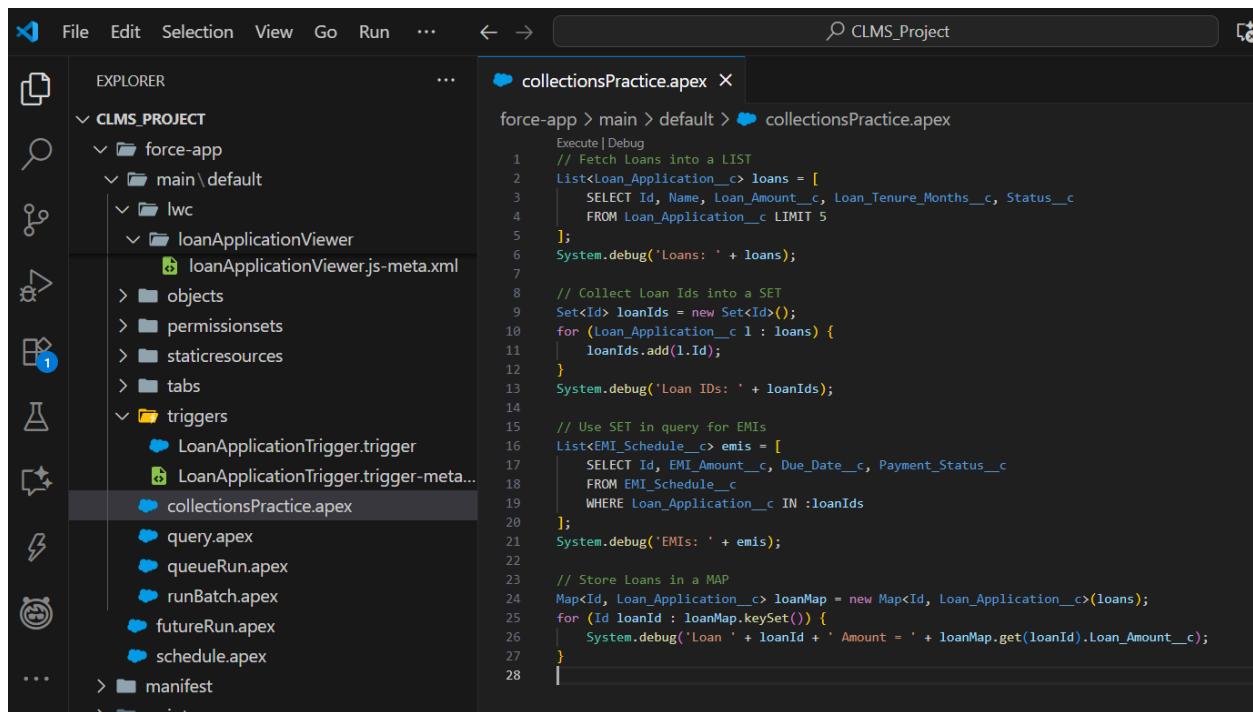
loanApplicationViewer.

```
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 to 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('X 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
41                     Date firstDue = Date.today().addMonths(1);
42
43                     EMI_Schedule__c emiSchedule = new EMI_Schedule__c();
44                     emiSchedule.Loan_Application__c = loan.Id;
45                     emiSchedule.Tenor__c = loan.Loan_Tenure_Months__c;
46                     emiSchedule.Amount__c = emiAmt;
47                     emiSchedule.First_Due__c = firstDue;
48                     toInsert.add(emiSchedule);
49                 }
50             }
51         }
52     }
53 }
```

◆ 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.



The screenshot shows the Salesforce IDE interface. On the left is the Explorer sidebar, which lists the project structure under 'CLMS_PROJECT'. It includes 'force-app' (main\default\lwc\loanApplicationViewer), 'objects', 'permissionsets', 'staticresources', 'tabs', and 'triggers' (LoanApplicationTrigger.trigger, LoanApplicationTrigger.trigger-meta...). In the center-right is the code editor window titled 'collectionsPractice.apex'. The code is as follows:

```
1 // Fetch Loans into a LIST
2 List<Loan_Application__c> loans = [
3     SELECT Id, Name, Loan_Amount__c, Loan_Tenure_Months__c, Status__c
4     FROM Loan_Application__c LIMIT 5
5 ];
6 System.debug('Loans: ' + loans);
7
8 // Collect Loan IDs into a SET
9 Set<Id> loanIds = new Set<Id>();
10 for (Loan_Application__c l : loans) {
11     loanIds.add(l.Id);
12 }
13 System.debug('Loan IDs: ' + loanIds);
14
15 // Use SET in query for EMIs
16 List<EMI_Schedule__c> emis = [
17     SELECT Id, EMI_Amount__c, Due_Date__c, Payment_Status__c
18     FROM EMI_Schedule__c
19     WHERE Loan_Application__c IN :loanIds
20 ];
21 System.debug('EMIs: ' + emis);
22
23 // Store Loans in a MAP
24 Map<Id, Loan_Application__c> loanMap = new Map<Id, Loan_Application__c>(loans);
25 for (Id loanId : loanMap.keySet()) {
26     System.debug('Loan ' + loanId + ' Amount = ' + loanMap.get(loanId).Loan_Amount__c);
27 }
28
```

◆ Batch Apex (Asynchronous)

EMIOverdueBatch.cls

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

Outcome: Automates overdue EMI tracking.

The screenshot shows the Salesforce IDE interface with the CLMS Project selected. The Explorer pane on the left lists the project structure, including the force-app/main/default/classes directory which contains several classes like CLMSConstants.cls, LoanFuture.cls, etc., and the specific class EMIOverdueBatch.cls which is currently selected and highlighted in blue. The code editor on the right displays the Apex code for EMIOverdueBatch.cls:

```


1  global class EMIOverdueBatch implements Database.Batchable<sObject>, Database.Stateful {
2      global Database.QueryLocator start(Database.BatchableContext bc) {
3          try {
4              return Database.getQueryLocator([
5                  SELECT Id, Payment_Status__c, Due_Date__c
6                  FROM EMI_Schedule__c
7                  WHERE Payment_Status__c = :CLMSConstants.EMI_STATUS_PENDING
8                  AND Due_Date__c < :Date.today()
9              ]);
10         } catch (Exception ex) {
11             System.debug('X Error in Batch Start: ' + ex.getMessage());
12             return Database.getQueryLocator([SELECT Id FROM EMI_Schedule__c WHERE Id = null]);
13         }
14     }
15
16     global void execute(Database.BatchableContext bc, List<EMI_Schedule__c> scope) {
17         try {
18             for (EMI_Schedule__c emi : scope) {
19                 emi.Payment_Status__c = CLMSConstants.EMI_STATUS_OVERDUE;
20             }
21             if (!scope.isEmpty()) {
22                 update scope;
23             }
24         } catch (DmlException ex) {
25             System.debug('X DML Error in Batch Execute: ' + ex.getMessage());
26         } catch (Exception ex) {
27             System.debug('X General Error in Batch Execute: ' + ex.getMessage());
28         }
29     }
30
31     global void finish(Database.BatchableContext bc) {
32         System.debug(' Batch Finished');
33     }
34 }


```

◆ 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.

The screenshot shows the Salesforce IDE interface with the CLMS Project selected. The Explorer pane on the left lists the project structure, including the force-app/main/default/classes directory which contains several classes like CLMSConstants.cls, EMIOverdueBatch.cls, etc., and the specific class LoanQueueable.cls which is currently selected and highlighted in blue. The code editor on the right displays the Apex code for LoanQueueable.cls:

```


1  public class LoanQueueable implements Queueable {
2      private Set<Id> loanIds;
3      public LoanQueueable(Set<Id> loanIds) {
4          this.loanIds = loanIds;
5      }
6
7      public void execute(QueueableContext qc) {
8          try {
9              List<Loan_Application__c> loans = [
10                  SELECT Id, Name, Loan_Amount__c
11                  FROM Loan_Application__c
12                  WHERE Id IN :loanIds
13              ];
14              System.debug(' Processing loans: ' + loans);
15          } catch (QueryException ex) {
16              System.debug('X Query failed in LoanQueueable: ' + ex.getMessage());
17          } catch (Exception ex) {
18              System.debug('X General Error in LoanQueueable: ' + ex.getMessage());
19          }
20      }
21  }

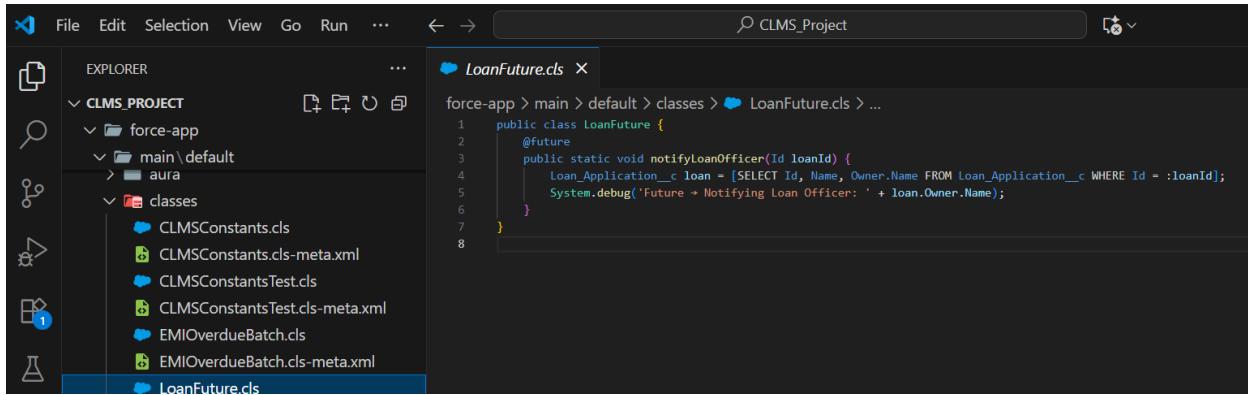

```

◆ Future Methods

LoanFuture.cls

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

Outcome: Simplifies async notifications.



The screenshot shows the Salesforce IDE interface with the project 'CLMS_Project' open. In the Explorer pane on the left, under the 'CLMS_PROJECT' section, there is a 'classes' folder containing several files: CLMSConstants.cls, CLMSConstants.cls-meta.xml, CLMSConstantsTest.cls, CLMSConstantsTest.cls-meta.xml, EMIOverdueBatch.cls, EMIOverdueBatch.cls-meta.xml, and LoanFuture.cls. The 'LoanFuture.cls' file is currently selected and displayed in the code editor on the right. The code in the editor is as follows:

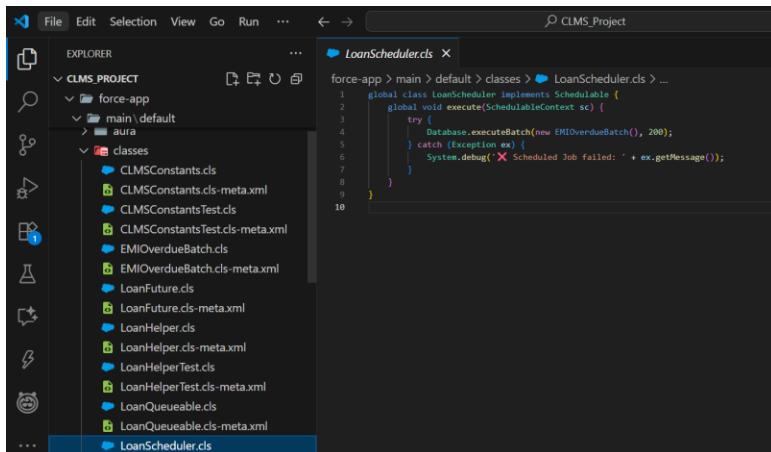
```
1  public class LoanFuture {  
2      @future  
3      public static void notifyLoanOfficer(Id loanId) {  
4          Loan_Application__c loan = [SELECT Id, Name, Owner.Name FROM Loan_Application__c WHERE Id = :loanId];  
5          System.debug('Future - Notifying Loan Officer: ' + loan.Owner.Name);  
6      }  
7  }  
8
```

◆ Scheduled Apex

LoanScheduler.cls

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

Outcome: Ensures overdue EMIs are updated automatically every night.



The screenshot shows the Salesforce IDE interface with the project 'CLMS_Project' open. In the Explorer pane on the left, under the 'CLMS_PROJECT' section, there is a 'classes' folder containing many files, including LoanScheduler.cls. The 'LoanScheduler.cls' file is currently selected and displayed in the code editor on the right. The code in the editor is as follows:

```
1  global class LoanScheduler implements Schedulable {  
2      global void execute(SchedulableContext sc) {  
3          try {  
4              Database.executeBatch(new EMIOverdueBatch(), 200);  
5          } catch (Exception ex) {  
6              System.debug('Scheduled Job failed: ' + ex.getMessage());  
7          }  
8      }  
9  }  
10
```

◆ 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.

The screenshot shows the Salesforce IDE interface. The left sidebar is the Explorer pane, displaying the project structure under 'CLMS_PROJECT'. The main area is the code editor for 'LoanHelperTest.cls', which contains Apex test code. The code includes methods for creating a loan application, calculating EMIs, and testing triggers. The code editor has syntax highlighting and line numbers.

```

1  @isTest
2  Run All Tests | Debug All Tests
3
4  private class LoanHelperTest {
5
6      // Utility method - create a valid loan Application
7      private static Loan_Application__c createLoan(String name, Decimal amount, Integer tenure, Decimal rate, String status) {
8          Loan_Application__c loan = new Loan_Application__c(
9              Loan_Amount__c = amount,
10             Loan_Tenure_Months__c = tenure,
11             Interest_Rate__c = rate,
12             Application_Date__c = Date.today(),
13             Status__c = status
14         );
15         insert loan;
16         return loan;
17     }
18
19     // • Test EMI Calculation
20     @isTest static void testEMICalculation() {
21         Decimal emi = LoanHelper.calculateEMI(120000, 12, 12);
22         System.assert(emi > 0, 'EMI should be greater than 0');
23     }
24
25     // • Test Trigger → EMI Creation
26     @isTest static void testTriggerCreatesEMIs() {
27         Loan_Application__c loan = createLoan('Test Loan', 120000, 12, 12, CLMSConstants.STATUS_PENDING);
28
29         Test.startTest();
30         loan.Status__c = CLMSConstants.STATUS_APPROVED_BY_BRANCH;
31         update loan;
32         Test.stopTest();
33
34         List<EMI_Schedule__c> emis = [
35             SELECT Id
36             FROM EMI_Schedule__c
37             WHERE Loan_Application__c = :loan.Id
38         ];
39
40         System.assertEquals(12, emis.size(), 'Number of EMIs should exactly match the loan tenure months');
41     }

```

◆ 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%

The screenshot displays two separate test runs for the CLMSConstantsTest.cls file within the CLMS_Project workspace in the Salesforce IDE.

Top Test Run (Initial Run):

```

force-app > main > default > classes > CLMSConstantsTest.cls > ...
1  @isTest
2  Run All Tests | Debug All Tests
3  private class CLMSConstantsTest {
4      Run Test | Debug Test
5  }
6  @isTest static void testConstantsUsage() {
7
}

```

Test Summary:

NAME	VALUE
Outcome	Failed
Tests Ran	7
Pass Rate	86%
Fail Rate	14%
Skip Rate	0%
Test Run Id	707gK00000DzKH2
Test Setup Time	0 ms
Test Execution Time	2528 ms
Test Total Time	2528 ms
Org Id	00dgK00000BBtksUAT
Username	nikhileshreddy.k157@agentforce.com
Org Wide Coverage	86%

Warning: The input format for array arguments has changed. Use this format: --array-flag value1 --array-flag value2 --array-flag value3

Bottom Test Run (After Changes):

```

force-app > main > default > classes > CLMSConstantsTest.cls > ...
1  @isTest
2  Run All Tests | Debug All Tests
3  private class CLMSConstantsTest {
4      Run Test | Debug Test
5  }
6  @isTest static void testConstantsUsage() {
7
}

```

Test Summary:

CLASSES	PERCENT	UNCOVERED LINES
CLMSConstants	0%	

Test Summary:

NAME	VALUE
Outcome	Passed
Tests Ran	1
Pass Rate	100%
Fail Rate	0%
Skip Rate	0%
Test Run Id	707gK00000Dz7ev
Test Setup Time	0 ms
Test Execution Time	33 ms
Test Total Time	33 ms
Org Id	00dgK00000BBtksUAT
Username	nikhileshreddy.k157@agentforce.com
Org Wide Coverage	86%

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.

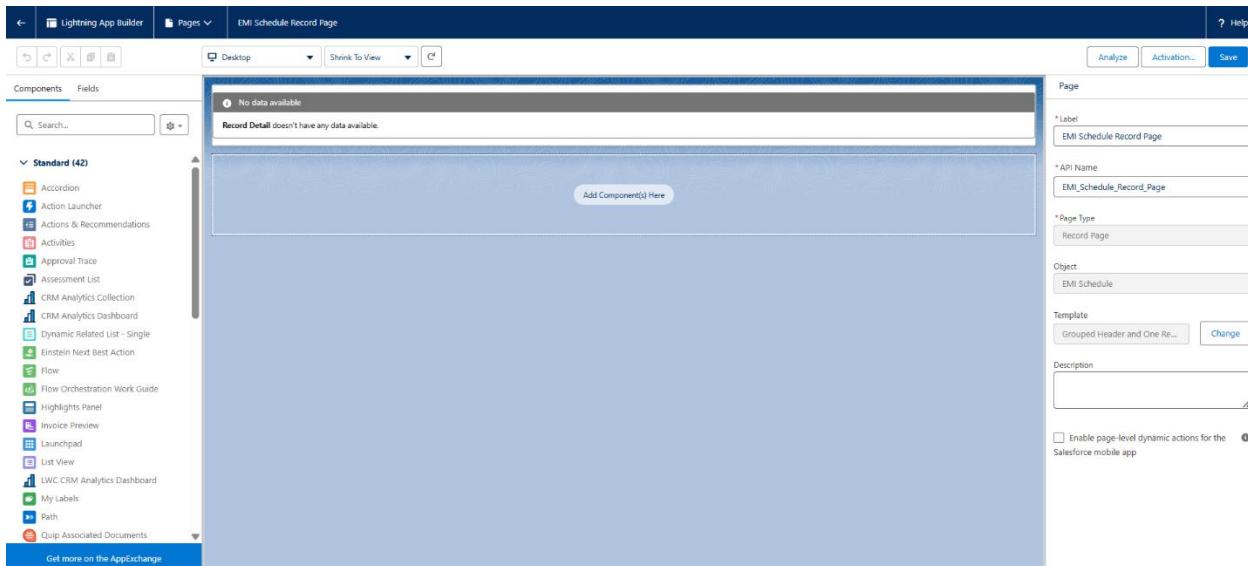
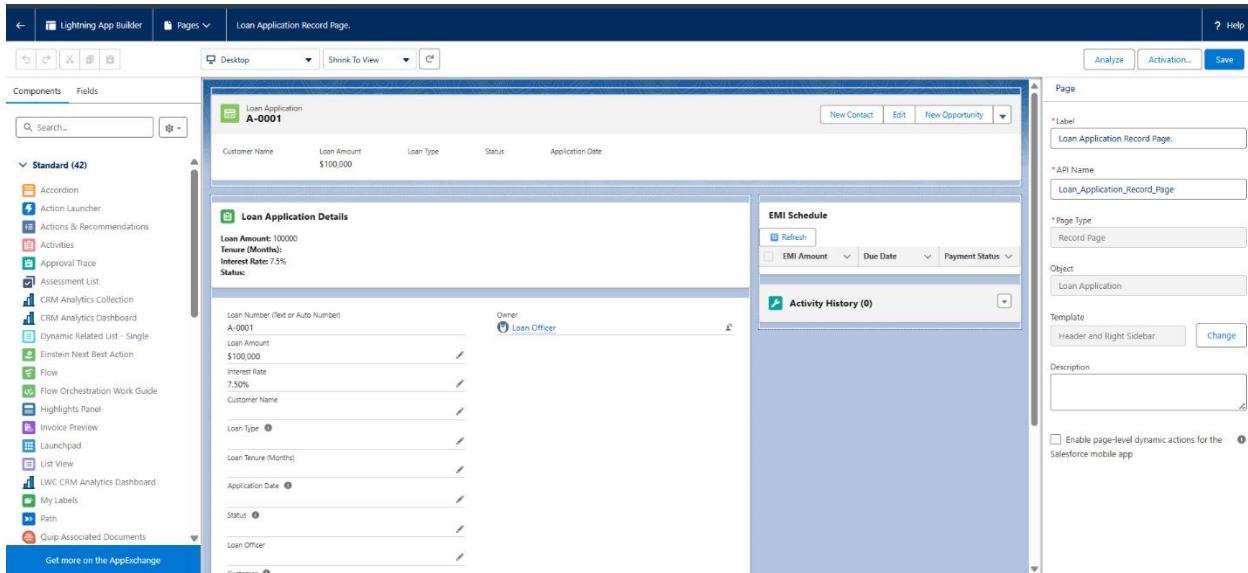


The screenshot shows the Salesforce Lightning App Builder interface. At the top, there's a header with a gear icon labeled "SETUP" and the title "Lightning App Builder". Below the header, a descriptive text explains the purpose of the builder. A navigation bar includes a "View" dropdown set to "All" and a "Create New View" button. A letter navigation bar at the top right lists letters from A to Z, with "All" selected. The main content area is titled "Lightning Pages" and contains a table with three rows of data. The columns are: Action, Label (with a dropdown arrow), Name, Namespace Prefix, Description, Type, Created By, and Last Modified By. The data is as follows:

Action	Label	Name	Namespace Prefix	Description	Type	Created By	Last Modified By
Edit Clone Del	CLMS Home Page	CLMS_Home_Page			Home Page	nib. 9/25/2025, 2:13 AM	nib. 9/25/2025, 2:13 AM
Edit Clone Del	EMI Schedule Record Page	EMI_Schedule_Record_Page			Record Page	nib. 9/25/2025, 1:38 AM	nib. 9/25/2025, 1:54 AM
Edit Clone Del	Loan Application Record Page	Loan_Application_Record_Page			Record Page	nib. 9/25/2025, 1:30 AM	nib. 9/25/2025, 2:57 AM

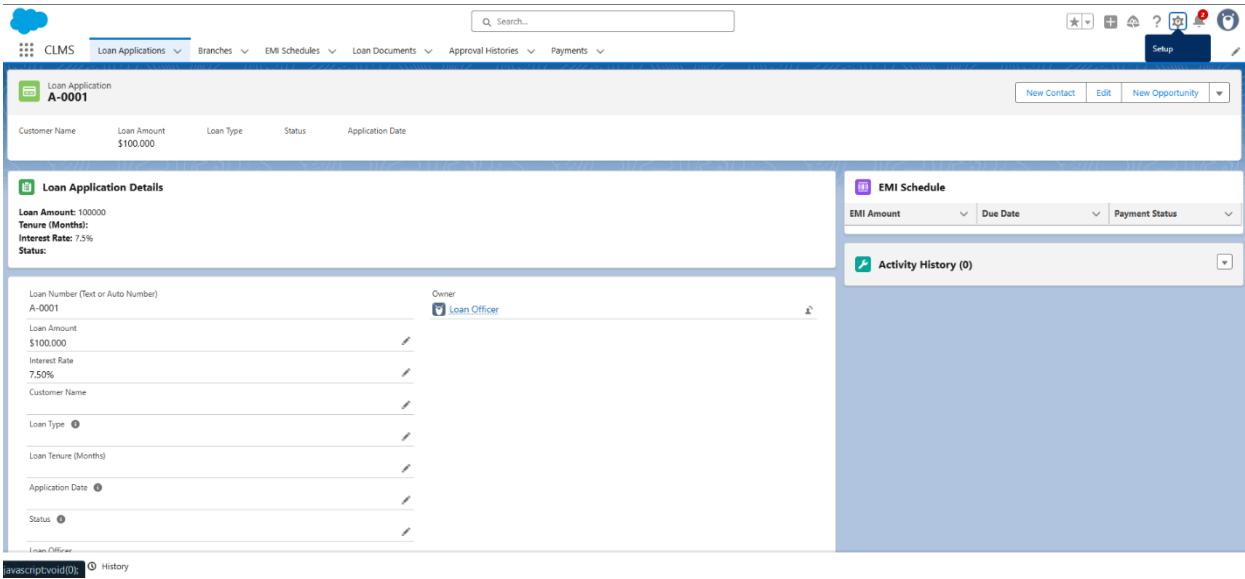
◆ Record Pages

- 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.



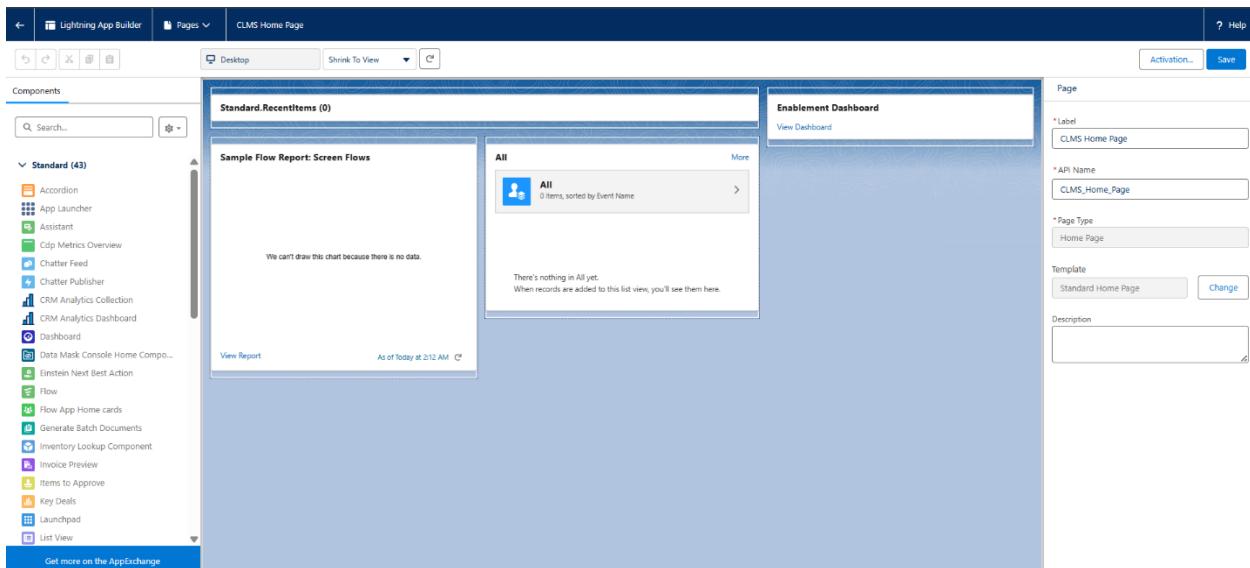
◆ Tabs

- 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:

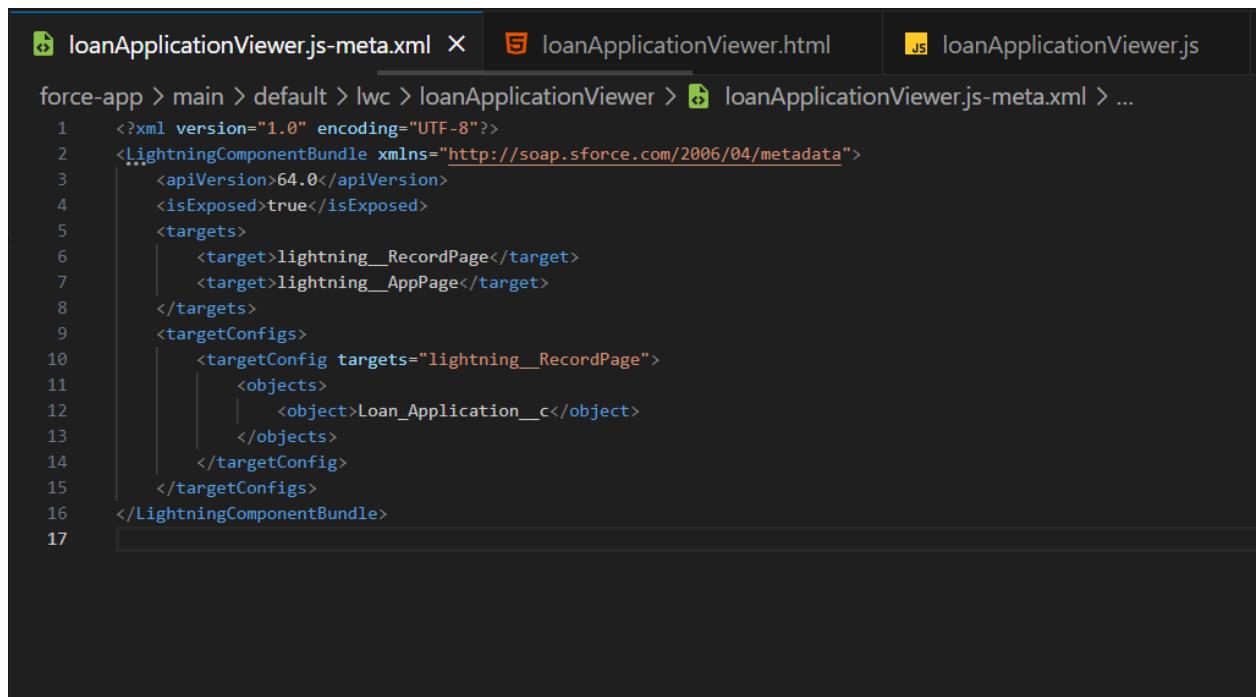
- Example: Refresh button triggers EMI list reload.
 - **Outcome:** LWCs became interactive and dynamic.
-

◆ Wire Adapters

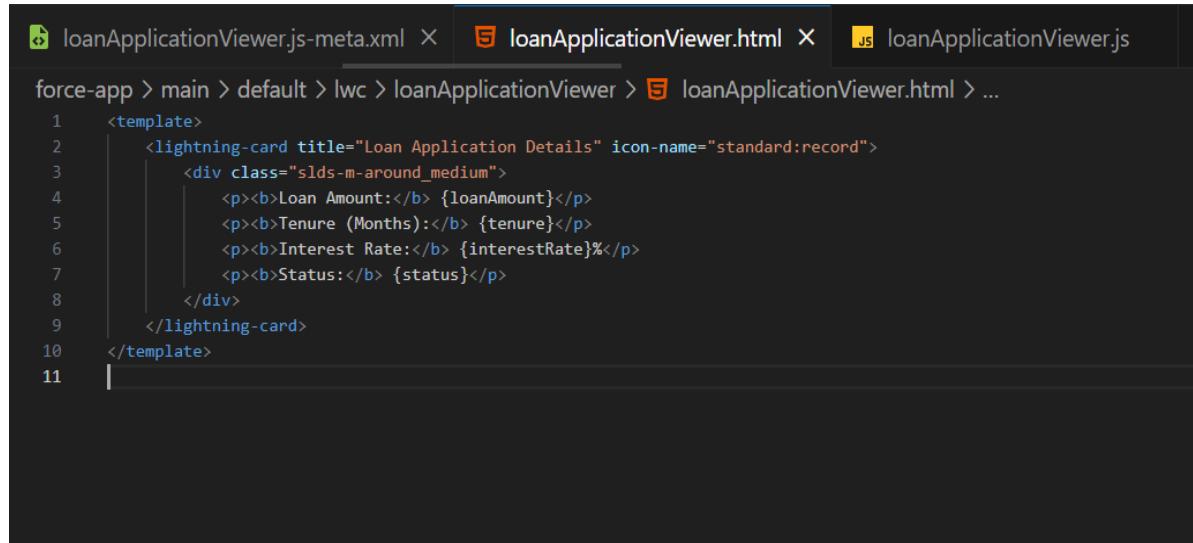
- Used @wire with getEmiByLoan.
 - Auto-updates EMI list when Loan Application changes.
-

◆ Imperative Apex Calls

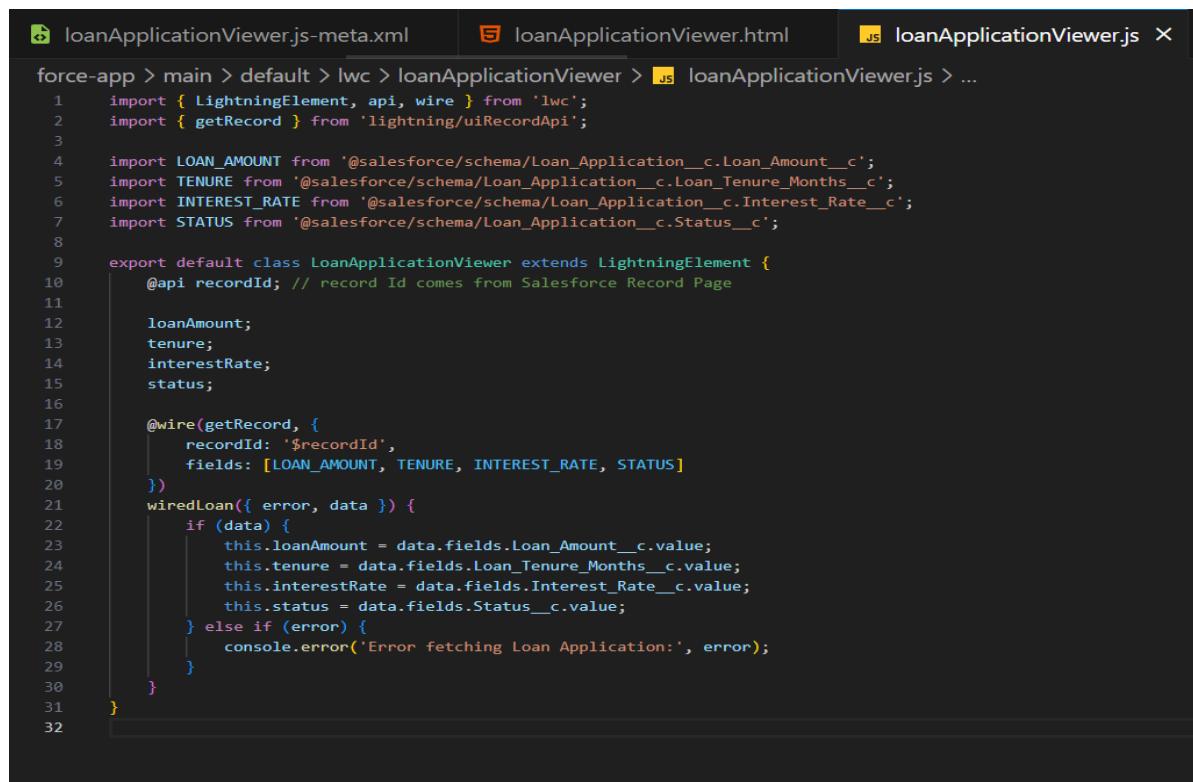
- Refresh button → calls Apex imperatively → reloads EMI list.
- LOAN APPLICATION VIEWER



```
force-app > main > default > lwc > loanApplicationViewer > loanApplicationViewer.js-meta.xml > ...
1  <?xml version="1.0" encoding="UTF-8"?>
2  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3      <apiVersion>64.0</apiVersion>
4      <isExposed>true</isExposed>
5      <targets>
6          <target>lightning__RecordPage</target>
7          <target>lightning__AppPage</target>
8      </targets>
9      <targetConfigs>
10         <targetConfig targets="lightning__RecordPage">
11             <objects>
12                 <object>Loan_Application__c</object>
13             </objects>
14         </targetConfig>
15     </targetConfigs>
16 </LightningComponentBundle>
17
```



```
loanApplicationViewer.js-meta.xml X loanApplicationViewer.html X loanApplicationViewer.js X
force-app > main > default > lwc > loanApplicationViewer > loanApplicationViewer.html > ...
1  <template>
2    <lightning-card title="Loan Application Details" icon-name="standard:record">
3      <div class="slds-m-around_medium">
4        <p><b>Loan Amount:</b> {loanAmount}</p>
5        <p><b>Tenure (Months):</b> {tenure}</p>
6        <p><b>Interest Rate:</b> {interestRate}%</p>
7        <p><b>Status:</b> {status}</p>
8      </div>
9    </lightning-card>
10   </template>
11
```



```
loanApplicationViewer.js-meta.xml X loanApplicationViewer.html X loanApplicationViewer.js X
force-app > main > default > lwc > loanApplicationViewer > loanApplicationViewer.js > ...
1  import { LightningElement, api, wire } from 'lwc';
2  import { getRecord } from 'lightning/uiRecordApi';
3
4  import LOAN_AMOUNT from '@salesforce/schema/Loan_Application__c.Loan_Amount__c';
5  import TENURE from '@salesforce/schema/Loan_Application__c.Loan_Tenure_Months__c';
6  import INTEREST_RATE from '@salesforce/schema/Loan_Application__c.Interest_Rate__c';
7  import STATUS from '@salesforce/schema/Loan_Application__c.Status__c';
8
9  export default class LoanApplicationViewer extends LightningElement {
10    @api recordId; // record Id comes from Salesforce Record Page
11
12    loanAmount;
13    tenure;
14    interestRate;
15    status;
16
17    @wire(getRecord, {
18      recordId: '$recordId',
19      fields: [LOAN_AMOUNT, TENURE, INTEREST_RATE, STATUS]
20    })
21    wiredLoan({ error, data }) {
22      if (data) {
23        this.loanAmount = data.fields.Loan_Amount__c.value;
24        this.tenure = data.fields.Loan_Tenure_Months__c.value;
25        this.interestRate = data.fields.Interest_Rate__c.value;
26        this.status = data.fields.Status__c.value;
27      } else if (error) {
28        console.error('Error fetching Loan Application:', error);
29      }
30    }
31  }
```

- EMISCHEDULER
VIEWER

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

- Explorer View:** Shows the project structure under "CLMS_PROJECT". The "lwc" folder contains "emiScheduleViewer" which has "emiScheduleViewer.html", "emiScheduleViewer.js", and "emiScheduleViewer.js-meta.xml".
- Terminal:** Displays the command: `PS C:\Users\Nikhilesh\Downloads\CLMS_Project> sfdx project deploy start --source-dir force-app/main/default/lwc/emiScheduleViewer`
- Deployed Source Table:** A table showing the status of files deployed:

State	Name	Type	Path
Changed	emiScheduleViewer	LightningComponentBundle	force-app/main/default/lwc/emiScheduleViewer/emiScheduleViewer.html
Changed	emiScheduleViewer	LightningComponentBundle	force-app/main/default/lwc/emiScheduleViewer/emiScheduleViewer.js
Changed	emiScheduleViewer	LightningComponentBundle	force-app/main/default/lwc/emiScheduleViewer/emiScheduleViewer.js-meta.xml
- Code Editor:** The "emiScheduleViewer.html" file is open, showing the following code:


```

1 <template>
2   <lightning-card title="EMI Schedule">
3     <lightning-button label=" Refresh" onclick={handleRefresh}></lightning-button>
4
5     <template if:true={emis}>
6       <lightning-datatable
7         key-field="Id"
8         data={emis}
9         columns={columns}>
10        </lightning-datatable>
11      </template>
12
13      <template if:true={error}>
14        <p style="color: red">▲ {error}</p>
15      </template>
16    </lightning-card>
17  </template>
18
      
```

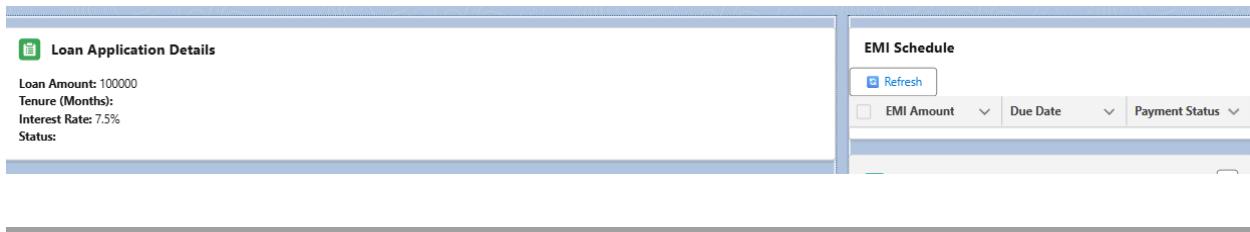
The screenshot shows a code editor interface with three tabs at the top: `emiScheduleViewer.html`, `emiScheduleViewer.js`, and `emiScheduleViewer.js-meta.xml`. The `emiScheduleViewer.js` tab is active, displaying the following LWC component code:

```
force-app > main > default > lwc > emiScheduleViewer > emiScheduleViewer.js > ...
1 import { LightningElement, api, track, wire } from 'lwc';
2 import getEmiByLoan from '@salesforce/apex/LoanHelper.getEmiByLoan';
3
4 export default class EmiScheduleViewer extends LightningElement {
5     @api recordId;
6     @track emis = [];
7     @track error;
8
9     columns = [
10         { label: 'EMI Amount', fieldName: 'EMI_Amount__c', type: 'currency' },
11         { label: 'Due Date', fieldName: 'Due_Date__c', type: 'date' },
12         { label: 'Payment Status', fieldName: 'Payment_Status__c', type: 'text' }
13     ];
14
15     // Wire → auto fetch
16     @wire(getEmiByLoan, { loanId: '$recordId' })
17     wiredEmis({ data, error }) {
18         if (data) {
19             this.emis = data;
20         } else if (error) {
21             this.error = error;
22         }
23     }
24
25     // Imperative → refresh button
26     handleRefresh() {
27         getEmiByLoan({ loanId: this.recordId })
28             .then(result => {
29                 this.emis = result;
30                 this.error = undefined;
31             })
32     }
33 }
```

```

1   <?xml version="1.0" encoding="UTF-8"?>
2   <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3       <apiVersion>58.0</apiVersion>
4       <isExposed>true</isExposed>
5       <targets>
6           <target>lightning__RecordPage</target>
7       </targets>
8       <targetConfigs>
9           <targetConfig targets="lightning__RecordPage">
10             <objects>
11                 <object>Loan_Application__c</object>
12             </objects>
13         </targetConfig>
14     </targetConfigs>
15 </LightningComponentBundle>
16

```



◆ 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.

SETUP

Named Credentials

Named Credentials External Credentials External Auth Identity Providers

1 items - Sorted by Label

Label	Type	URL	External Credential	Actions
LoanAPI	Secured Endpoint	https://api.loanprovider.com	LoanAPI_Credential	<button>New</button>

SETUP > NAMED CREDENTIALS

LoanAPI

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:

SETUP > NAMED CREDENTIALS

LoanAPI_Credential

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:

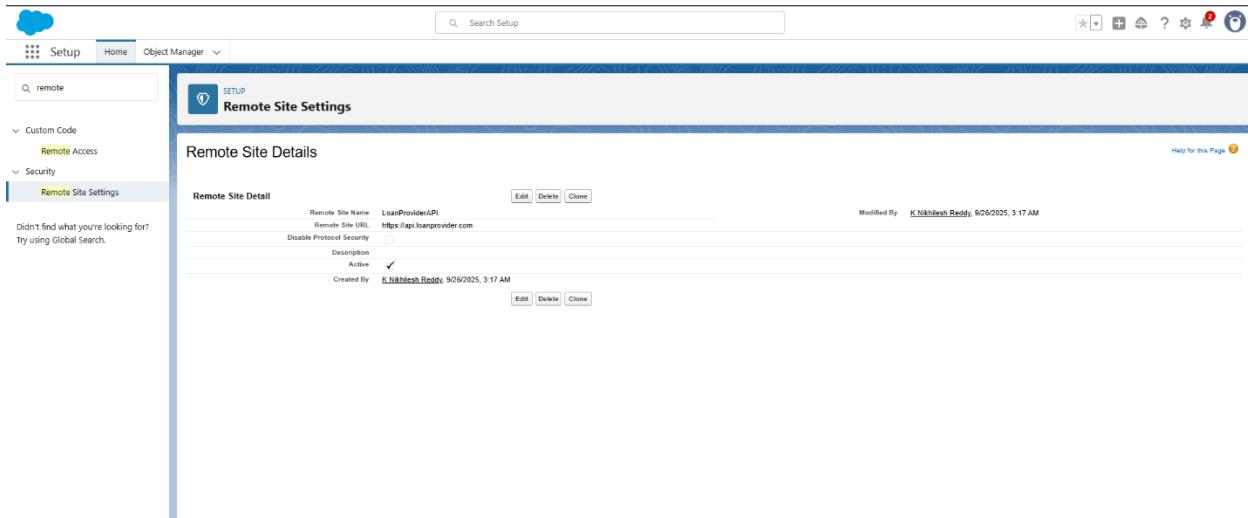
Sequence	Parameter Name	Identity Type	Authentication Status	Username	Actions
1	LoanAPI_Principal	Named Principal	Configured	loan_user_123	<button>New</button>

Custom Headers:

◆ 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.

The screenshot shows the Force.com IDE interface with the project 'CLMS_Project' open. In the Explorer pane, the 'classes' folder contains several files: CLMSConstants.cls, CLMSConstants.cls-meta.xml, CLMSConstantsTest.cls, CLMSConstantsTest.cls-meta.xml, EMIOverdueBatch.cls, EMIOverdueBatch.cls-meta.xml, LoanAPI.cls, LoanAPI.cls-meta.xml, LoanAPIService.cls, LoanAPIService.cls-meta.xml, and LoanAPIServiceTest.cls. The 'LoanAPIServiceTest.cls' file is selected in the code editor. The code editor displays the following Apex class:

```

1  private class LoanAPIServiceTest {
2
3     @isTest static void testSendDisbursement() {
4         // Test logic here
5     }
6
7 }

```

The terminal window shows the command: `sf apex get test -i 707gK0000E6fhx -o nikhilreddy.k157@agentforce.com`. The output indicates the test passed with a value of 2111.

NAME	VALUE
Outcome	Passed
Tests Ran	2
Pass Rate	100%
Fail Rate	0%
Skip Rate	0%
Test Run Id	707gK0000E6fhx
Test Setup Time	0 ms
Test Execution Time	2160 ms
Test Total Time	2160 ms
Org Id	00DgK0000BtksUAT
Username	nikhilreddy.k157@agentforce.com

◆ 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.

The screenshot shows the Salesforce Setup page under the 'Platform Events' section. A new platform event named 'LoanApprovalEvent' is being defined. The 'Standard Fields' section lists the following fields:

Action	Field Label	API Name	Field Name	Data Type	Indexed	Controlling Field	Indirect
Edit	Created By	ApprovedBy__c	CreatedBy	Lookup(User)			
Edit	Created Date	ApprovedBy__c	CreatedDate	Date/Time			
Edit	Event UUID	LoanId__c	EventUUID	Text(36)			
Edit	Replay ID	Status__c	ReplayId	External Lookup			

The 'Custom Fields & Relationships' section shows the following relationships:

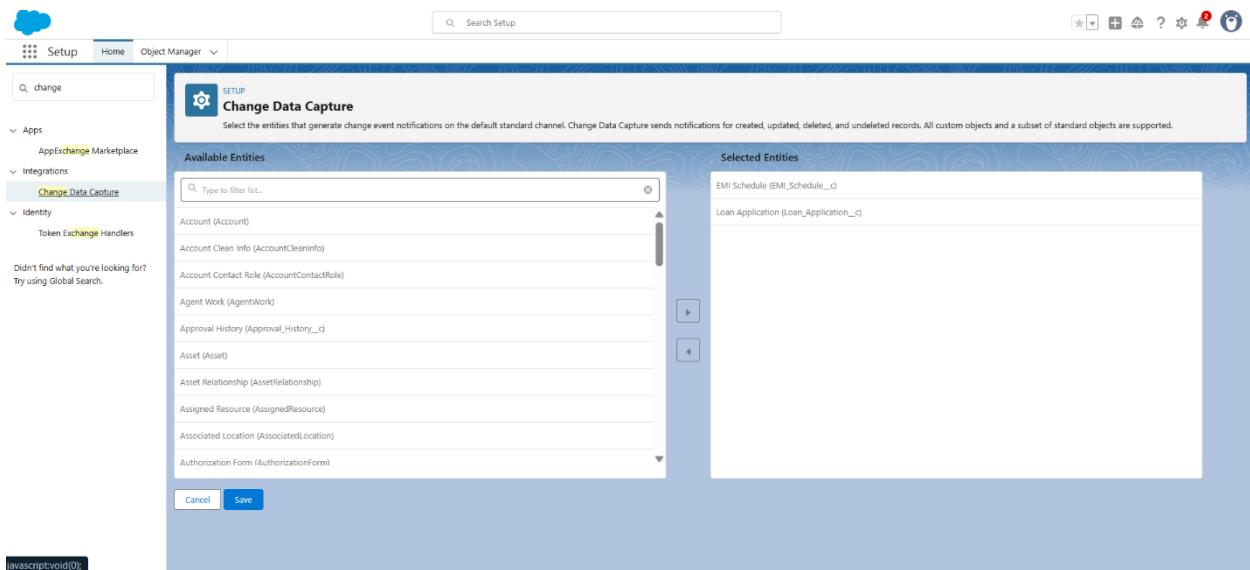
Action	Field Label	API Name	Date Type	Indexed	Controlling Field	Modified By
Edit	Approval Date	ApprovedDate__c	Date/Time			K.Nikhilreddy, 9/26/2025, 2:56 AM
Edit	Approved By	ApprovedBy__c	Text(80)			K.Nikhilreddy, 9/26/2025, 2:55 AM
Edit	Loan ID	LoanId__c	Text(15)			K.Nikhilreddy, 9/26/2025, 2:54 AM
Edit	Status	Status__c	Text(50)			K.Nikhilreddy, 9/26/2025, 2:55 AM

◆ 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**

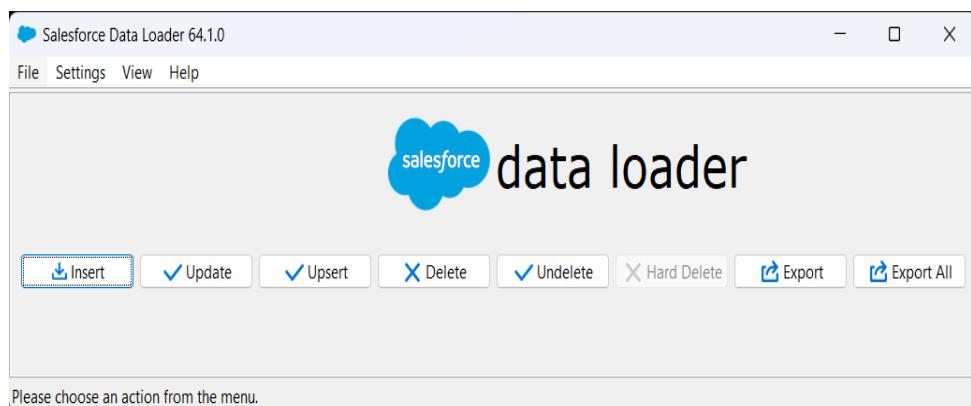
- **What I did:**
 - Used **Data Import Wizard** in Salesforce Setup.
 - Imported **Customer_c** and **Loan_Application_c** records from CSV files.
 - Mapped fields such as **Loan_Amount_c**, **Loan_Tenure_Months_c**, and **Interest_Rate_c**.
- **Why:**

- To bulk load customer master data and initial loan applications into the system.
- **Screenshot to include:**
 - Data Import Wizard page showing field mapping.

The screenshot shows the 'Recent Import Jobs' section of the Data Import Wizard. It includes a table with columns: Status, Object, Records Created, Records Updated, Records Failed, Start Date, and Processing Time (ms). One job is listed: Closed, Loan Application, 5, 0, 0, 09-26-2025 11:05, 304. Below the table is a dark button labeled 'Bulk Api Monitoring'. At the bottom, there's a light blue bar with the text 'Clean up your data import file' and a 'Collapse' link.

◆ 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.



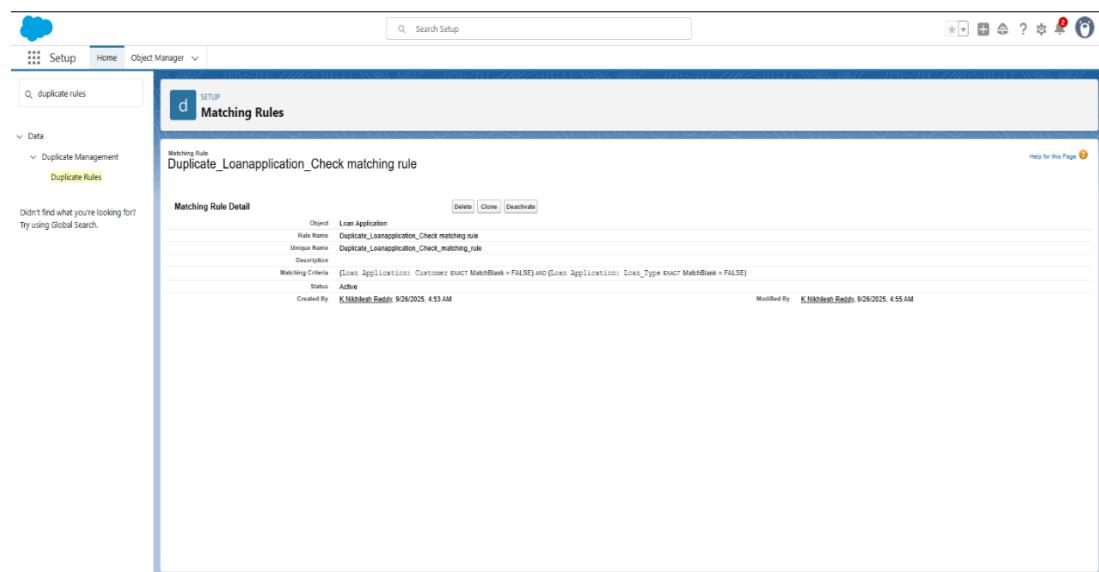
◆ 3. Duplicate Rules

- **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.

The screenshot shows the 'Data Export' page in the Salesforce setup. At the top, there's a header with the Salesforce logo and 'SETUP'. Below it, the page title is 'Data Export'. A sub-header says 'Monthly Export Service'. A note below states: 'Data Export lets you prepare a copy of all your data in salesforce.com. From this page you can start the export process manually or schedule it to run automatically. When an export is ready for download you will receive an email containing a link that allows you to download the file(s). The export files are also available on this page for 48 hours, after which time they are deleted.' There's a yellow banner at the top indicating 'Next scheduled export: None'. Below this are buttons for 'Export Now' and 'Schedule Export'. A table follows, showing a single scheduled export entry:

Scheduled By	K Nikhilesh Reddy	
Scheduled Date	9/26/2025	
Export File Encoding	ISO-8859-1 (General US & Western European, ISO-LATIN-1)	
Action	File Name	File Size
download	WE_00Dgk30000EBkSUA_.ZIP	1.1K

◆ 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.

The screenshot shows the Salesforce Setup interface with the Package Manager selected. A package named "CLMS_PHASE8" is displayed. The "Components" tab is selected, showing a list of components including Account, All, Application Date, Approval Date, ApprovalEvent, Branch, Branch Code, and Branch Layout, all associated with the Loan Application object.

This screenshot shows the same Salesforce Setup interface and package details as the first one. However, the "Components" tab is not selected; instead, the "Versions" tab is selected, displaying a list of package versions. The first version listed is "1.0" with a release date of "9/26/2025, 5:14 AM".

◆ 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.

```

File Edit Selection View Go Run ...
CLMS_Project
PROJECT
force-app
main/default
classes
LoanAPIService.cls
LoanAPIService.cls-meta.xml
LoanAPIServiceTest.cls
LoanAPIServiceTest.cls-meta.xml
LoanFuture.cls
LoanFuture.cls-meta.xml
LoanHelper.cls
LoanHelper.cls-meta.xml
LoanHelperTest.cls
LoanHelperTest.cls-meta.xml
LoanQueueable.cls
LoanQueueable.cls-meta.xml
LoanScheduler.cls
LoanScheduler.cls-meta.xml
LoanTriggerHandler.cls
LoanTriggerHandler.cls-meta.xml
contentassets
flexipages
layouts
OUTLINE
TIMELINE
RUNNING TASKS
...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS HISTORY
powershell + ...
PS C:\Users\Nikhilesh\Downloads\CLMS_Project> sf project retrieve start --source-dir force-app/main/default
>>
Changed LoanTriggerHandler ApexClass
Changed LoanTriggerHandler ApexClass
Changed LoanApplicationTrigger ApexTrigger
Changed LoanApplicationTrigger ApexTrigger
Changed emiScheduleViewer LightningComponentBundle
Changed emiScheduleViewer LightningComponentBundle
Changed emiScheduleViewer LightningComponentBundle
Changed loanApplicationViewer LightningComponentBundle
Changed loanApplicationViewer LightningComponentBundle
Changed loanApplicationViewer LightningComponentBundle
1
force-app/main/default\classes\LoanTriggerHandler.cls
force-app/main/default\classes\LoanTriggerHandler.cls-meta.xml
force-app/main/default\triggers\LoanApplicationTrigger.trigger
force-app/main/default\triggers\LoanApplicationTrigger.trigger-meta.xml
force-app/main/default\lwc\emiScheduleViewer\emiScheduleViewer.html
force-app/main/default\lwc\emiScheduleViewer\emiScheduleViewer.js
force-app/main/default\lwc\loanApplicationViewer\loanApplicationViewer.html
force-app/main/default\lwc\loanApplicationViewer\loanApplicationViewer.js
force-app/main/default\lwc\loanApplicationViewer\loanApplicationViewer.js-meta.xml

```

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).

The screenshot shows the Salesforce Setup interface under the 'Custom Report Types' section. A custom report type named 'Loan Applications with EMI Schedules' is displayed. The 'Details' section includes fields for Display Label, API Name, Description, Created By, Store in Category, Deployment Status, and Modified By. The 'Fields' section shows the source object 'Loan Applications' and included fields (22 total, 10 selected). The 'Object Relationships' section illustrates a relationship between 'Loan Applications' (A) and 'EMI Schedules' (B), indicating that A has at least one related record from B. Buttons for Preview Layout, Edit Layout, Clone, Delete, and Close are visible.

Dashboards

We built dashboards using the above reports:

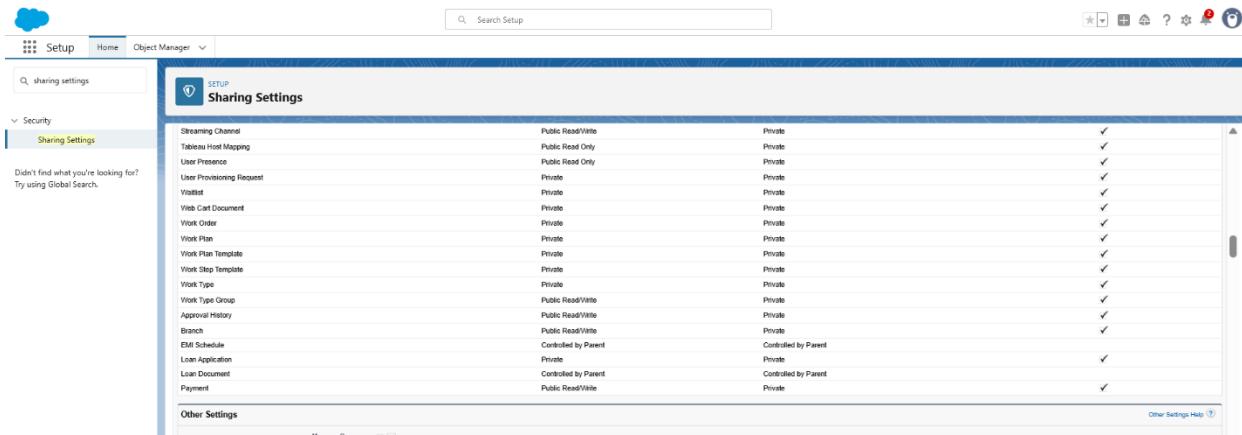
- Loan Status Pie Chart.
- Branch-wise Loan Volume Bar Chart.
- Repayment Trend Line Chart.
- Top Customers by Loan Amount Table.

The screenshot shows the process of defining a new custom report type. Step 2, 'Define Report Records Set', is completed, showing 'Loan Applications' as the primary object. A button labeled '(Click to relate another object)' is available for adding more objects. The bottom right corner features 'Cancel' and 'Save' buttons.

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.



Object	Sharing Rule	Access Level	Action
Streaming Channel	Public Read/Write	Private	✓
Tableau Host Mapping	Public Read Only	Private	✓
User Presence	Public Read Only	Private	✓
User Provisioning Request	Private	Private	✓
Waitlist	Private	Private	✓
Web Cart Document	Private	Private	✓
Work Order	Private	Private	✓
Work Plan	Private	Private	✓
Work Plan Template	Private	Private	✓
Work Step Template	Private	Private	✓
Work Type	Private	Private	✓
Work Type Group	Public Read/Write	Private	✓
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	✓
Other Settings			

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:

