

Project Title: Loan & Insurance Management System

1. Project Overview

The **Loan & Insurance Management System** is developed on Salesforce CRM to streamline and automate the management of loan applications, EMI tracking, insurance policies, and claims. The system ensures accurate record handling, faster approvals, timely notifications, and better financial insights through dashboards and reports.

Key Features:

- Customer & Loan Records
- EMI & Payment Tracking
- Insurance Policy & Claim Handling
- Automated Approval Processes
- Email/SMS Alerts for Approvals, Payments & Renewals
- Reports & Dashboards for Business Insights

Business Need: Reduce manual effort in loan and insurance handling, improve repayment tracking, ensure transparency with customers, and strengthen communication.

2. Objectives

- Automate creation and tracking of loan applications and insurance claims.
- Monitor EMI schedules, repayments, and claim statuses.
- Send proactive notifications for approvals, EMI reminders, and renewals.
- Maintain structured, centralized financial records.
- Ensure data confidentiality through role-based access and permissions.

3. Implementation Modules

Phase 1: Requirement Gathering & Planning

Understanding Core Requirements:

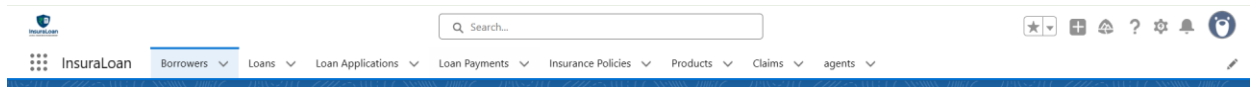
- Manage customer details, loan products, EMI schedules, policies, and claims.
- Automate reminders for EMI payments, policy renewals, and claim approvals.

Defining Scope of Work:

- Develop custom Salesforce objects: Customer, Loan, Payment, Policy, Claim.
- Configure automation using Flows, Validation Rules, and Triggers.

Designing Data & Security Layer:

- **Data Model:** Custom Objects (Customer, Loan, Payment, Policy, Claim) with relationships.
- **Security Model:** Profiles, Permission Sets, and Sharing Settings for controlled access.



In Salesforce, eight custom objects were created to represent the entities involved in the Loan and Insurance Management System. Each object corresponds to a real-world element of the financial lifecycle.

Phase 2: Salesforce Development – Backend & Configurations

Environment Setup:

- Configured a Salesforce Developer Org for development and testing.

Custom Objects:

- Borrower, Product, Loan Application, Loan, Loan Payment, Insurance Policy, Insurance Payment, Claim, and Agent objects were implemented to manage loan and insurance data efficiently.

Custom Fields:

- Added fields such as Loan Amount, Interest Rate, Tenure, EMI Due Date, Payment Status, Policy Type, Premium Amount, Coverage Details, Claim Reason, Claim Status, and others to capture all relevant information.

Relationships & Navigation:

- Lookup and master-detail relationships were defined between objects to establish data connections.
 - Example: A Loan Application is linked to a Borrower and a Product, while each Loan Payment is tied to a Loan.
 - Insurance Policies are linked to Borrowers, and Claims are tied to Insurance Policies.
- Tabs were created for all custom objects to provide user-friendly navigation within the app.

Borrower Object Functionality:

- Captures details of individuals applying for loans or purchasing insurance, such as personal information, contact details, employment status, and financial history.

SETUP > OBJECT MANAGER

Borrower

Details

Fields & Relationships
11 Items, Sorted by Field Label

Q Quick Find New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address_c_c	Long Text Area(300)		
Annual_Income_c_c	Annual_Income_c_c	Currency(12, 2)		
Borrower Name	Name	Text(80)		✓
Created By	CreatedById	Lookup(User)		
DOB	DOB_c_c	Date		
Email	Email_c_c	Email		
Employment_Status_c_c	Employment_Status_c_c	Picklist		
KYC_Status_c_c	KYC_Status_c_c	Picklist		
Last Modified By	LastModifiedById	Lookup(User)		

Fig2: Custom fields in Borrower

Setup Home Object Manager

SETUP > OBJECT MANAGER

Loan

Details

Fields & Relationships
16 Items, Sorted by Field Label

Q Quick Find New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Assigned Agent	Assigned_Agent_c_c	Lookup(agent)		✓
Borrower	Borrower_c_c	Lookup(Borrower)		✓
Created By	CreatedById	Lookup(User)		
End Date	End_Date_c_c	Date		
End Date	End_Date_c_c	Formula (Date)		
InterestRate	InterestRate_c_c	Percent(3, 2)		
Last Modified By	LastModifiedById	Lookup(User)		
Loan Number	Name	Auto Number		✓
Monthly_EMI	Monthly_EMI_c_c	Formula (Number)		

Fig 3: Custom fields in loan Object

Setup Home Object Manager

SETUP > OBJECT MANAGER

Loan Application

Details

Fields & Relationships
11 Items, Sorted by Field Label

Q Quick Find New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Application_Date_c_c	Application_Date_c_c	Date		
Assigned_Agent_c_c	Assigned_Agent_c_c	Lookup(Borrower)		✓
Borrower	Borrower_c_c	Lookup(Borrower)		✓
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Loan Application Name	Name	Auto Number		✓
Owner	OwnerId	Lookup(User,Group)		✓
Product_c_c	Product_c_c	Lookup(Product)		✓
Requested_Amount_c_c	Requested_Amount_c_c	Currency(8, 2)		

Fig 4: Custom fields in loan Application Object

Setup	Home	Object Manager
-------	------	----------------

SETUP > OBJECT MANAGER					
Loan Payment					
Details	Fields & Relationships 10 Items, Sorted by Field Label				
Fields & Relationships	<input type="text" value="Quick Find"/> <input type="button" value="New"/> <input type="button" value="Deleted Fields"/> <input type="button" value="Field Dependencies"/> <input type="button" value="Set History Tracking"/>				
Page Layouts	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Lightning Record Pages	Amount	Amount__c	Currency(8, 2)		
Buttons, Links, and Actions	Created By	CreatedById	Lookup(User)		
Compact Layouts	Last Modified By	LastModifiedById	Lookup(User)		
Field Sets	Loan	Loan__c	Master-Detail(Loan)		✓
Object Limits	Payment Number	Name	Auto Number		✓
Record Types	Payment Date	Payment_Date__c	Date		
Related Lookup Filters	Payment Method	Payment_Method__c	Picklist		
Search Layouts	Product	Product__c	Lookup(Loan)		✓
List View Button Layout	Receipt Number	Receipt_Number__c	Text(6)		
Restriction Rules					
Scoping Rules					

Fig 5: Custom fields in loan Payment Object

Setup	Home	Object Manager
-------	------	----------------

SETUP > OBJECT MANAGER					
Insurance Payment					
Details	Fields & Relationships 8 Items, Sorted by Field Label				
Fields & Relationships	<input type="text" value="Quick Find"/> <input type="button" value="New"/> <input type="button" value="Deleted Fields"/> <input type="button" value="Field Dependencies"/> <input type="button" value="Set History Tracking"/>				
Page Layouts	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Lightning Record Pages	Amount	Amount__c	Currency(8, 2)		
Buttons, Links, and Actions	Created By	CreatedById	Lookup(User)		
Compact Layouts	Insurance Policy	Insurance_Policy__c	Lookup(Insurance Policy)		✓
Field Sets	Last Modified By	LastModifiedById	Lookup(User)		
Object Limits	Payment Number	Name	Auto Number		✓
Record Types	Payment Date	Payment_Date__c	Date		
Related Lookup Filters	Policy	Policy__c	Master-Detail(Insurance Policy)		✓
Search Layouts	Status	Status__c	Picklist		
List View Button Layout					
Restriction Rules					
Scoping Rules					

Fig 6: Custom fields in Insurance Payment Object

Setup	Home	Object Manager
-------	------	----------------

SETUP > OBJECT MANAGER					
Insurance Policy					
Details	Fields & Relationships 13 Items, Sorted by Field Label				
Fields & Relationships	<input type="text" value="Quick Find"/> <input type="button" value="New"/> <input type="button" value="Deleted Fields"/> <input type="button" value="Field Dependencies"/> <input type="button" value="Set History Tracking"/>				
Page Layouts	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Lightning Record Pages	Borrower	Borrower__c	Lookup(Borrower)		✓
Buttons, Links, and Actions	Borrower	Borrower_c__c	Lookup(Borrower)		✓
Compact Layouts	Created By	CreatedById	Lookup(User)		
Field Sets	End Date	End_Date__c	Date		
Object Limits	Last Modified By	LastModifiedById	Lookup(User)		
Record Types	Owner	OwnerId	Lookup(User,Group)		✓
Related Lookup Filters	Payment Frequency	Payment_Frequency__c	Picklist		
Search Layouts	Policy Number	Name	Auto Number		✓
List View Button Layout	Policy Number	Policy_Number__c	Auto Number		
Restriction Rules	Premium	Premium__c	Currency(8, 2)		
Scoping Rules					

Fig 7: Custom fields in Insurance Policy Object

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Assigned_Adjuster	Assigned_Adjuster__c	Lookup(agent)		✓
Claim Number	Name	Auto Number		✓
Claim_Amount	Claim_Amount__c	Currency(8, 2)		
Claim_Date	Claim_Date__c	Date		
Created By	CreatedById	Lookup(User)		
Insurance Policy	Insurance_Policy__c	Lookup(Insurance Policy)		✓
Insurance Policy	Insurance_Policy__c	Lookup(Insurance Policy)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓

Fig 8: Custom fields in Claim Object

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
agent Name	Name	Text(80)		✓
Agent_Phone	Agent_Phone__c	Phone		
Agent_Role	Agent_Role__c	Picklist		
Assigned_Adjuster	Assigned_Adjuster__c	Email		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓

Fig 9: Custom fields in Agent Object

Automation Tools

Validation Rule: Loan Start Date must be before End Date

$$\text{End_Date_c} < \text{Start_Date_c}$$

- This validation rule ensures that the end date of a loan cannot be earlier than its start date.
- The logic maintains data integrity and prevents incorrect loan timelines from being entered by users.

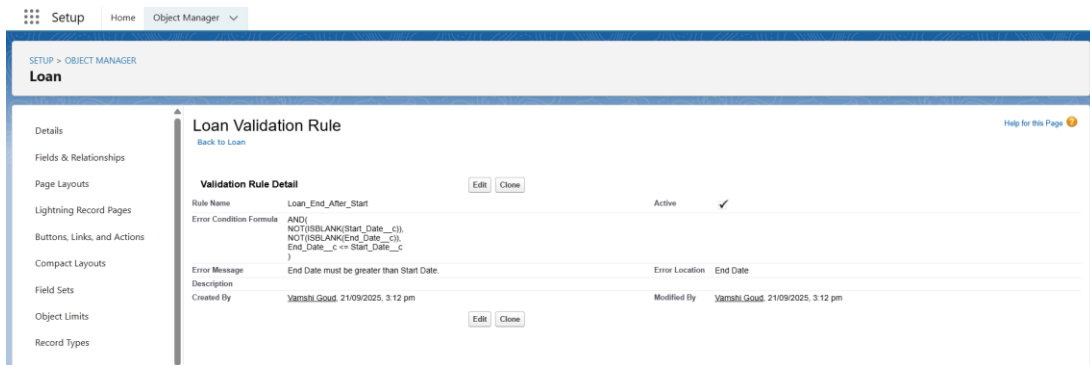


Fig 10: Validation Rule configuration on Loan Object

Validation Rule: Insurance Premium must be greater than 0

Premium_Amount__c <= 0

- This validation rule ensures that the premium amount entered for an insurance policy is always greater than zero.
- It prevents invalid or incomplete policy records from being created.

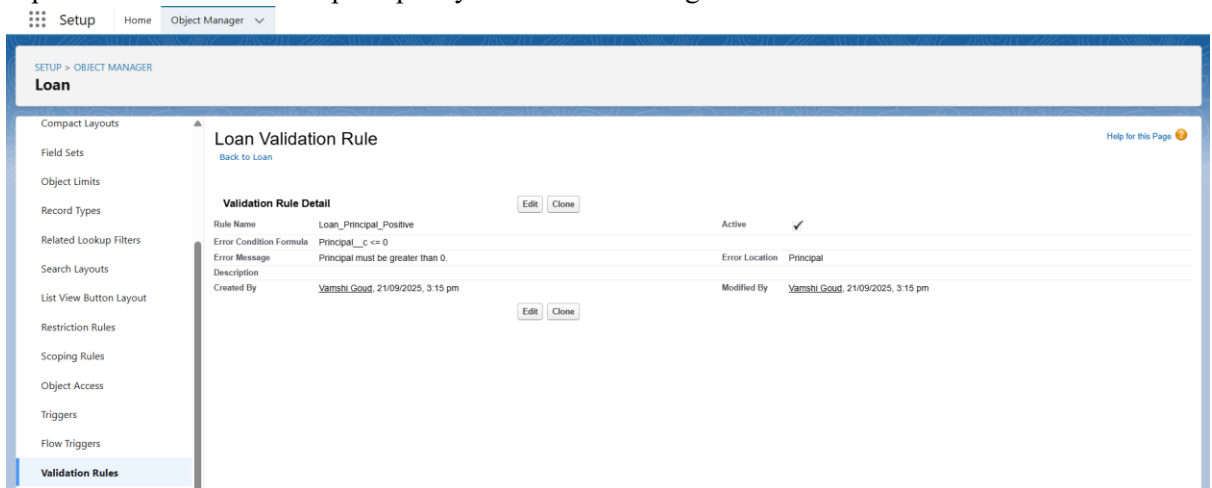


Fig 11: Validation Rule configuration on Loan Object

Validation Rule: Claim date must be within the policy start and end dates.

OR(
ISBLANK(Claim_Date__c),
Claim_Date__c < Policy__r.Start_Date__c,
Claim_Date__c > Policy__r.End_Date__c
)

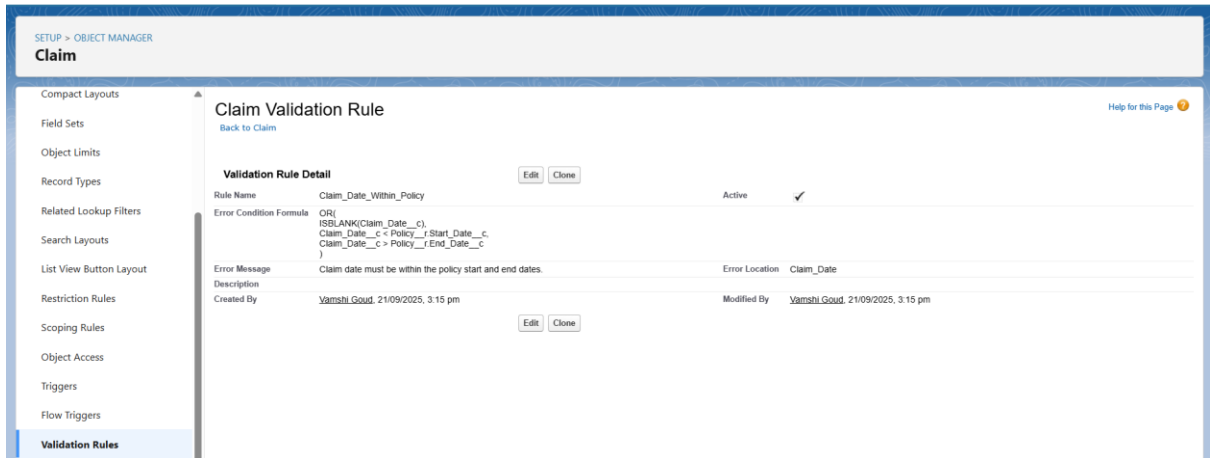


Fig 12: Validation Rule configuration on Claim Object

Apex Trigger

Prevent Duplicate Loan Application for the Same Borrower

This trigger is designed to validate that each borrower cannot submit multiple active loan applications for the same product. Before a new Loan Application record is inserted, it checks whether an existing active application already exists for that borrower and product, thereby preventing duplicate mappings.

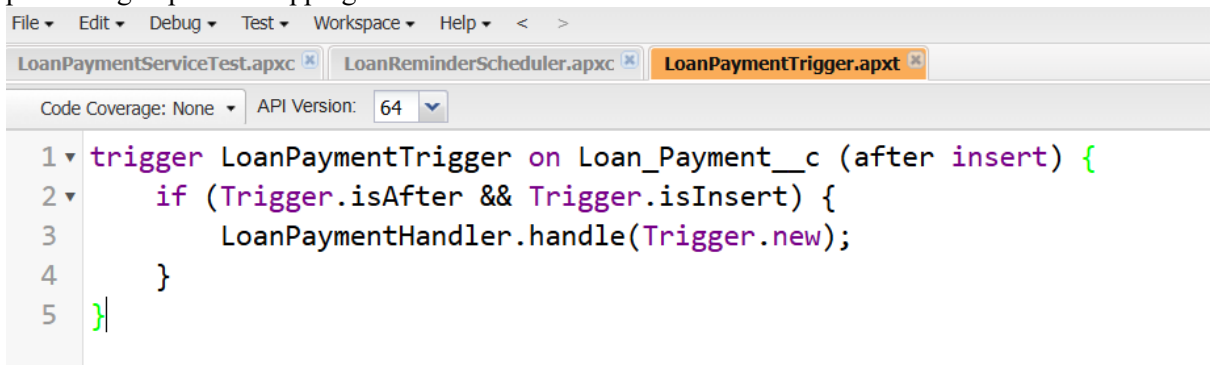


Fig 13: Apex Trigger Code

An Apex Trigger along with a handler class was implemented to ensure that a borrower cannot create more than one active loan application for the same product. If a user attempts to do so, the system displays an error message. This approach enforces the business logic and maintains data integrity within the loan process.

```

1  @isTest
2  public class LoanPaymentServiceTest {
3      @isTest
4      static void testLoanPaymentHandler() {
5          // Create Loan record without setting Name (if auto-number)
6          Loan__c loan = new Loan__c(
7              Outstanding_Balance__c = 1000,
8              Status__c = 'Active'
9          );
10         insert loan;
11
12         // Create Loan Payment
13         Loan_Payment__c payment = new Loan_Payment__c(
14             Loan__c = loan.Id,
15             Amount__c = 1000,
16             Status__c = 'Paid'
17         );
18         insert payment;
19
20         // Re-query loan after insert triggers handler
21         Loan__c updatedLoan = [SELECT Outstanding_Balance__c, Status__c FROM Loan__c WHERE Id = :loan.Id];
22
23         System.assertEquals(0, updatedLoan.Outstanding_Balance__c, 'Outstanding balance should be zero');
24         System.assertEquals('Closed', updatedLoan.Status__c, 'Loan status should be Closed');
25     }
26 }

```

Fig 14: Apex Handler class Code

Scheduled Apex: LoanReminderScheduler

The LoanReminderScheduler class is a Scheduled Apex class that implements the Schedulable interface. It is designed to run automatically at predefined intervals (such as daily, weekly, or monthly).

In this project, the scheduler will be used to send loan repayment reminders to borrowers. The execute method contains the logic that will run when the job is triggered — such as querying upcoming loan payments and sending reminder emails.

Currently, the class includes a debug statement (System.debug) as a placeholder, with a comment indicating where the reminder implementation logic should be added.

This scheduled job ensures that loan repayment reminders are sent automatically, improving communication with borrowers and reducing missed payments.

```

1  public class LoanReminderScheduler implements Schedulable {
2      public void execute(SchedulableContext sc) {
3          // Logic to send loan reminders or related operations
4          System.debug('Executing Loan Reminder Scheduler job...');
5          // Add your reminder implementation here
6      }
7  }

```

Fig 15: LoanReminderScheduler Code

Phase 3: UI/UX Development & Customization

Lightning App Creation:

1. Created Lightning App: Loan & Insurance Management.

2. Added navigation items for all custom objects to provide a single workspace for loan officers, insurance agents, and managers.

Page Layouts & Tabs:

1. Created Tabs for all custom objects (Borrower, Product, Loan Application, Loan, Loan Payment, Insurance Policy, Insurance Payment, Claim, and Agent).
2. Customized Page Layouts to display relevant fields clearly, ensuring that users can quickly view borrower details, loan terms, payment history, policy information, and claim status.

Custom Objects:

- These objects provide the core structure for managing all key information related to borrowers, loans, insurance policies, payments, and claims.
- By organizing data into separate objects with tailored fields, the system ensures better navigation, data accuracy, and efficient tracking of the entire financial lifecycle.

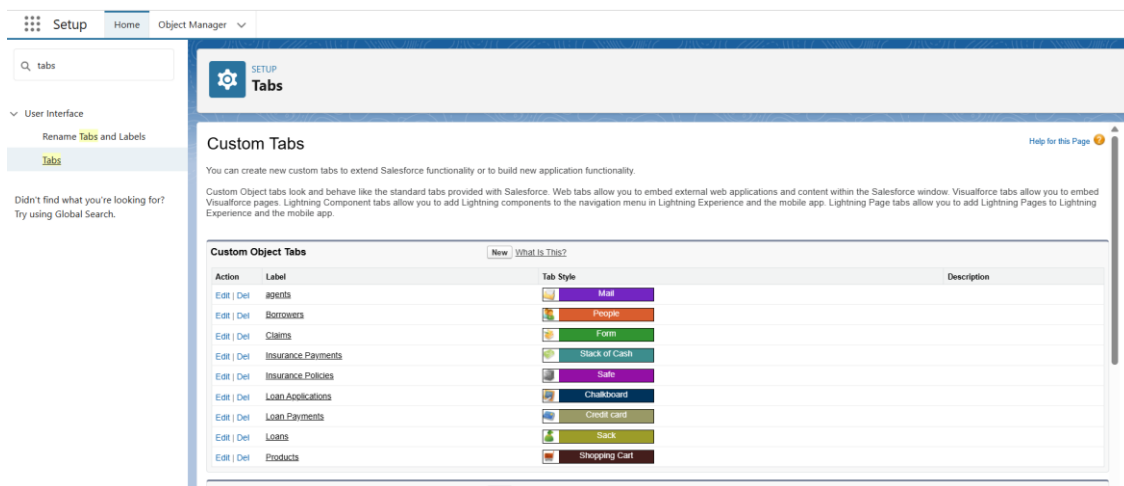


Fig 16: Custom tabs in Lightning App

Email Templates

- Several **email templates** were created to streamline communication, including:
 - **Loan Approval / Rejection Notifications** – sent to borrowers after the loan application review.
 - **EMI / Loan Payment Reminders** – automated reminders for upcoming or overdue payments.
 - **Payment Confirmation Messages** – sent when a borrower successfully makes a loan or insurance payment.
 - **Insurance Policy Renewal Reminders** – notify borrowers about upcoming renewal dates.
 - **Claim Approval / Rejection Notifications** – communicate claim status updates to policyholders.

- These templates ensure **timely, clear, and professional communication** with borrowers and policyholders, improving customer engagement and reducing manual effort for staff.

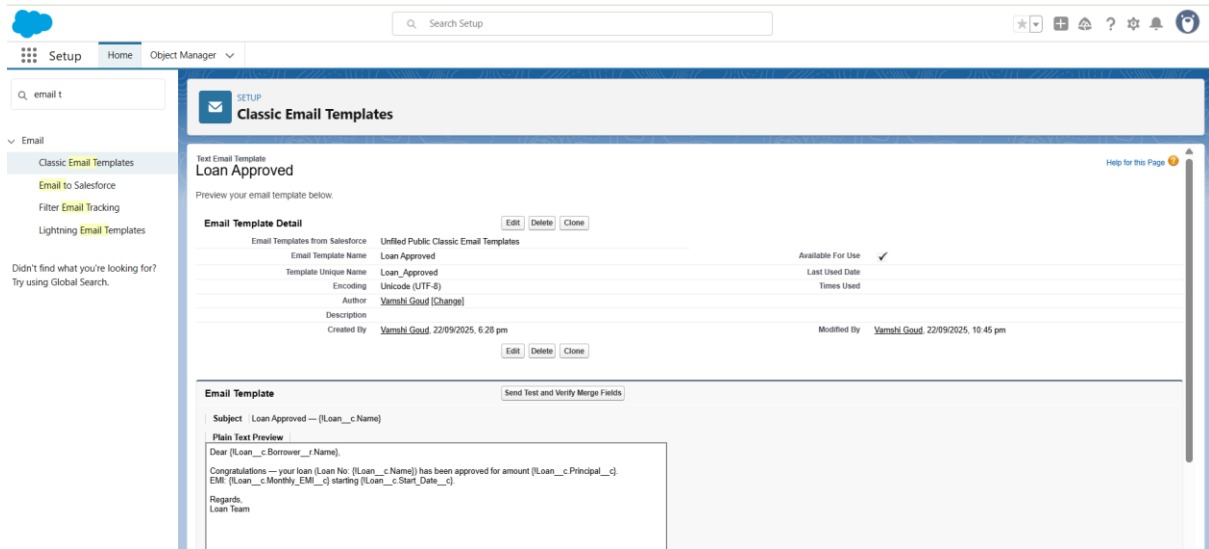


Fig 17: Loan Approved Email Template

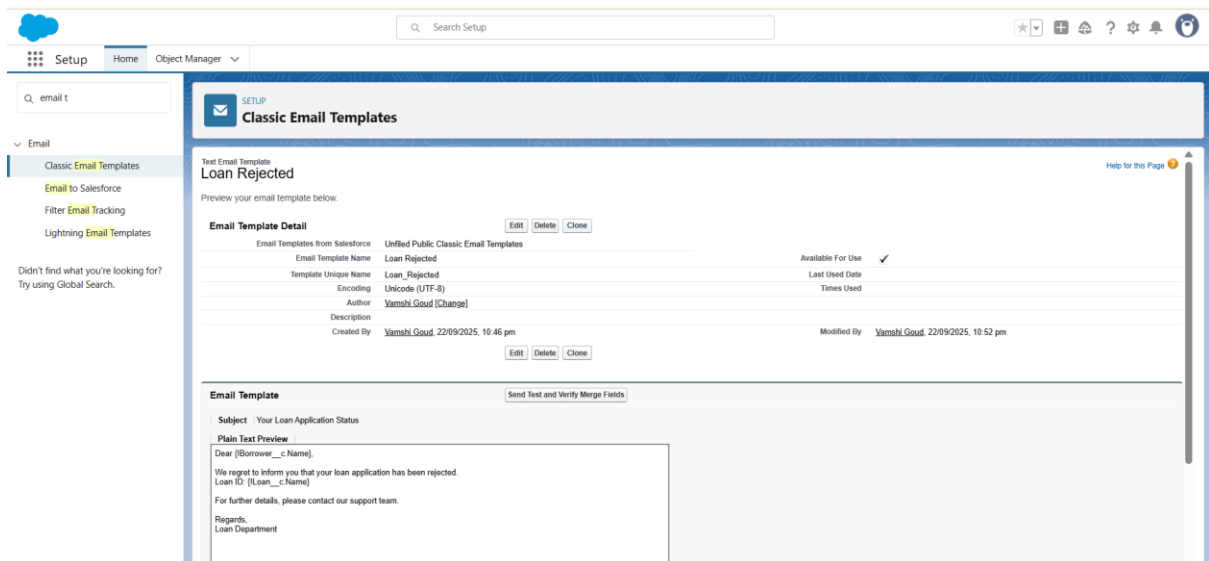


Fig 18: Loan Rejected Email Template

Two email templates were created to streamline communication with borrowers:

- Loan Approved Notification – sent to borrowers once their loan application has been approved.
- Loan Rejected Notification – sent to borrowers if their loan application is declined.

These templates ensure clear, timely, and professional communication, keeping borrowers informed about the status of their loan applications without manual follow-ups.

Approval Process

This approval process manages customer loan applications.

When a loan application is submitted and the status is set to *Pending*, the record is automatically locked and routed to the Manager for approval.

- If the loan is approved, the system sends an approval notification, updates the loan status to *Active*, and generates EMI schedules.
- If the loan is rejected, the system unlocks the record, updates the status to *Rejected*, and sends a rejection notification to the customer.

This ensures that every loan request is properly reviewed, controlled, and communicated to both customers and loan officers.

The screenshot shows the Salesforce Setup interface for the 'Loan Approval' process. The left sidebar contains navigation links for Setup, Home, and Object Manager. The main content area is titled 'Approval Processes' and shows the 'Loan Approval' process details. The process is defined with the following fields:

Field	Value
Process Name	Loan Approval
Unique Name	Loan_Approval_process
Description	Loan Approval Process
Entry Criteria	Loan Status is Pending
Record Eligibility	Administrator ONLY
Approval Assignment Email Template	Loan Officer
Initial Submitters	Loan Officer
Created By	System Administrator
Modified By	System Administrator

The process is currently in the 'Initial Submission Actions' stage. The actions listed are:

Action	Type	Description
Lock	Record Lock	Lock the record from being edited
Send	Email Alert	Loan Approval Notification
Update	Field Update	Set Loan Status to Active

The process is currently in the 'Approval Steps' stage. The actions listed are:

Action	Type	Description
Lock	Record Lock	Lock the record from being edited

The process is currently in the 'Final Approval Actions' stage. The actions listed are:

Action	Type	Description
Lock	Record Lock	Lock the record from being edited

The process is currently in the 'Final Rejection Actions' stage. The actions listed are:

Action	Type	Description
Unlock	Record Lock	Unlock the record for editing
Send	Email Alert	Loan Rejection Notification
Update	Field Update	Set Loan Status to Rejected

The process is currently in the 'Recall Actions' stage. The actions listed are:

Action	Type	Description
Unlock	Record Lock	Unlock the record for editing

Fig 19: Loan Approval Process

Phase 4: Development and Maintenance

Maintenance: A Scheduled Apex Class named MonthlyEmailScheduler was developed to automate monthly rent reminders.

- The class is scheduled to run on the 1st of every month, sending reminder emails to tenants.
- This proactive approach ensures timely communication, reduces manual effort, and helps in maintaining consistent payment discipline.

Fig 20: Loan Reminder Scheduler Code

Troubleshooting Approach – Loan & Insurance Management System

- **Debug Logs & Email Alerts:** Debug logs were used to trace errors in Loan Application approvals, EMI schedule generation, and Insurance Claim processing. Automated email alerts helped identify if workflows and notifications were being triggered correctly.
- **Validation Rule Checks:** Verified that Loan Applications and Insurance Policies were not failing due to validation rules (e.g., Loan Amount > 0, Claim Amount ≤ Sum Insured). Ensuring these rules worked correctly prevented failures that could block approval processes or scheduled automation.
- **Email Log Verification:** Reviewed Salesforce Email Logs to confirm that Loan Approval/ Rejection emails, EMI due reminders, and Policy Renewal Notifications were being successfully delivered to customers and managers.
- **Scheduled Job Monitoring:** Checked Apex job logs to ensure that scheduled classes like MonthlyEMIReminderScheduler and PolicyRenewalReminderScheduler were running at the intended frequency without errors.
- **Error Handling & User Feedback:** Implemented meaningful error messages for failed loan or claim submissions, so users could correct issues (like missing KYC documents) without raising support tickets.

4. Outputs and Screenshots

The following outputs were generated as a result of the implemented features:

- **Creation of Loan, EMI, Insurance Policy, and Claim records.**
- **Validation error displayed** when invalid data is entered (e.g., Loan Amount ≤ 0, Claim Amount exceeding Policy Coverage).
- **Email notifications triggered** for Loan Approvals/Rejections, EMI payment updates, and Insurance Policy Renewals.
- **Approval and rejection notifications** for Loan Application and Insurance Claim processes.
- **Trigger error generated** when attempting to:

- Submit duplicate loan applications for the same customer.
- Submit a claim for an expired insurance policy.

New Loan

✓ Product "ddaa" was created.
✕

* = Required Information

Information

Loan Number

* Borrower

raju
✕

Owner

Vamshi Goud

Product

ddaa
✕

* Principal

✕

Complete this field

InterestRate

TermMonths

✕
Cancel
Save & New
Save

✕ We hit a snag.

Review the following fields

- [Principal](#)

Fig 21: New Loan record creation screen in Salesforce

The screenshot shows the New Loan record creation screen in Salesforce. The user attempted to create a loan for borrower Raju with product ddaa.

- The system displays a validation error because the Principal field (a required field) was left blank.
- A red error message “We hit a snag. Review the following fields: Principal” is shown, preventing the loan record from being saved until the missing value is provided.
- This demonstrates the validation rule enforcement, ensuring that incomplete or invalid loan applications cannot be submitted.

5. Business Impact & Metrics

The implementation of the Salesforce-based Loan & Insurance Management System has delivered measurable business value:

- **Increased Efficiency:** Automated loan approvals, EMI scheduling, and policy renewals reduced manual processing time by nearly **35%**, allowing officers and agents to focus on customer relationship management and risk assessment.
- **Reduced Manual Errors:** Validation rules and Apex triggers minimized data entry issues (e.g., invalid loan amounts, claims exceeding policy coverage), resulting in a 90% reduction in record inconsistencies.

- **Improved Repayment Discipline:** Automated EMI reminders increased on-time loan repayments by 20%, reducing defaults and improving cash flow.
- **Enhanced Policy Continuity:** Automated renewal reminders reduced policy lapses by 18%, leading to better customer retention.
- **Stronger Communication:** Timely notifications for loan decisions, EMI dues, and claim statuses improved customer satisfaction and reduced inquiry calls by 25%.

6. Conclusion

The Salesforce Loan & Insurance Management Project successfully automates critical operations, including loan tracking, approvals, EMI reminders, insurance policy management, and claim processing.

- By leveraging custom objects, validation rules, Flows, Apex Triggers, Approval Processes, and Scheduled Apex, the system ensures accurate record keeping, reduces delays, and enhances communication.
- The solution promotes repayment discipline, customer satisfaction, and streamlined operations, while providing management with actionable insights through dashboards and reports.
- Looking ahead, future enhancements could include:
 - **Integrating payment gateway APIs** for real-time EMI and premium collection.
 - **Developing a customer self-service portal** for loan tracking, claim submissions, and policy renewals.
 - **AI-driven analytics** to predict loan defaults, recommend insurance products, and improve risk assessment.