

Integrated Banking Operations Hub With Integrated AI : A Salesforce Solution

Project Overview

This CRM project establishes an **Integrated Banking Hub** to unify critical financial operations: Loan Management, Wealth Management, and Regulatory Compliance. The platform automates high-volume lending decisions through Flows and Approval Processes, provides financial advisors with **real-time portfolio valuation**, and enforces security by creating an automated audit trail for compliance. The key business need addressed is replacing fragmented, manual processes with a secure, centralized, and efficient system.

Objectives

The main goals of building this CRM were to achieve operational efficiency and enhance regulatory posture.

- **Automate Lending:** Implement intelligent routing to ensure high-value loans (>₹20 Lakhs) receive mandatory manager review via an **Approval Process**, while low-value loans are auto-approved for streamlined booking.
- **Enable Real-Time Advice:** Calculate client portfolio value and goal progress dynamically using **Apex Triggers and Roll-Up logic** to enable proactive financial advice.
- **Ensure Compliance:** Establish a foundational **Audit Log** for critical changes and centralize risk monitoring using custom objects and automated Flows.
- **Visibility:** Deliver **custom Lightning Web Component (LWC) dashboards** for personalized, role-specific insights across all modules.

Key Features and Modules

The solution includes three main modules to address these issues:

1. **Loan Management Module:**
 - **Use Case:** Digitize the loan application, approval, and repayment process.
 - **Features:** Includes automated risk scoring, a multi-step approval process, and an automated audit trail.

2. Wealth & Portfolio Management Module:

- **Use Case:** Centralize the tracking of a client's investment portfolio, assets, and goals.
- **Features:** Provides **automatic calculation of portfolio value** (using Apex/Roll-Up logic), real-time asset tracking, and a dynamic LWC dashboard for advisors.

3. Compliance & Risk Management Module:

- **Use Case:** Monitor and manage regulatory compliance and risk events across the bank.
- **Features:** A system to track regulations, **audit major changes** (via Flow), and send alerts for compliance issues.

Phase 1: Problem Understanding & Industry Analysis

This phase outlines the initial steps for building the platform.

- **Requirement Gathering:** A single platform will be built to manage client records, loan applications, investment portfolios, and compliance regulations. Custom objects and automation will be used to connect the modules.

- **Stakeholder Analysis:** The project will impact several key roles:
 - **Loan Officers:** Will use the Loan Management Module to process applications.
 - **Wealth Advisors:** Will use the Wealth Management Module to manage client portfolios.
 - **Compliance Officers:** Will use the Compliance Module to enforce regulations.
 - **Management:** Will gain a complete view of the bank's operations, finances, and risk exposure.

- **Business Process Mapping:** The proposed process ensures that client information is entered only once. Actions like a loan application or creating an investment portfolio will be tracked by the compliance module to ensure security and compliance.
- **Industry-specific Use Case Analysis:** This project is considered innovative because it addresses data silos by building a single integrated platform, which is what real companies often do.
- **AppExchange Exploration:** The plan includes exploring tools like DocuSign for e-signatures, DocGen for document generation, and a free API for dummy stock prices.

Technical Implementation

The project will use various Salesforce capabilities:

- **Admin:** The plan includes building multiple objects, implementing Approval Processes for loans, and creating Validation Rules for compliance.

- **Developer:** The developer will use an Apex Trigger to calculate portfolio value, Batch Apex to update market data, and Scheduled Apex to send monthly compliance reports.
- **UI (LWC):** A custom Lightning Web Components (LWC) dashboard will be built for each module to provide a personalized user experience.
- **Integration:** REST callouts and Named Credentials will be used to get real-time stock prices.

Phase 2: Org Setup & Configuration

1. Company Profile Setup

Scenario: setting up the basic details for our bank. This ensures that the time, currency, and language are correct for our project.

Organization Name: Integrated Banking Hub

Default Time Zone: (GMT+05:30) India Standard Time (Asia/Kolkata)

Default Currency: INR - Indian Rupee

The screenshot shows the 'Company Information' setup page for the organization 'Integrated Banking Hub'. The page is divided into sections: 'Organization Detail' (containing fields like Organization Name, Primary Contact, Division, Address, Fiscal Year Starts In, Activate Multiple Currencies, Enable Data Translation, Newsletter, Admin Newsletter, Hide Notices About System Maintenance, Hide Notices About System Downtime, and Locale Formats), 'Phone' (containing fields like Default Locale, Default Language, Default Time Zone, Currency Locale, Used Data Space, Used File Space, API Requests, Last 24 Hours, Streaming API Events, Last 24 Hours, Restricted Logins, Current Month, Salesforce.com Organization ID, Organization Edition, and Instance), and 'Created By' and 'Modified By' sections (both showing 'OrgFarm EPIC' and the date/time). At the bottom, there are links for 'User Licenses' and 'User Licenses Help'.

Organization Detail		Phone	
Organization Name	Integrated Banking Hub	Phone	
Primary Contact	OrgFarm EPIC	Fax	
Division		Default Locale	English (India)
Address	India	Default Language	English
Fiscal Year Starts In	January	Default Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)
Activate Multiple Currencies	<input type="checkbox"/>	Currency Locale	English (India) - INR
Enable Data Translation	<input type="checkbox"/>	Used Data Space	342 KB (7%) [View]
Newsletter	<input checked="" type="checkbox"/>	Used File Space	17 KB (0%) [View]
Admin Newsletter	<input checked="" type="checkbox"/>	API Requests, Last 24 Hours	0 (15,000 max)
Hide Notices About System Maintenance	<input type="checkbox"/>	Streaming API Events, Last 24 Hours	0 (10,000 max)
Hide Notices About System Downtime	<input type="checkbox"/>	Restricted Logins, Current Month	0 (0 max)
Locale Formats	ICU	Salesforce.com Organization ID	00Dgk00000BTAr3
		Organization Edition	Developer Edition
		Instance	CAN96

Created By: OrgFarm EPIC, 9/12/2025, 3:17 PM Modified By: Kundana Sree Bavisetti, 9/17/2025, 10:26 AM

[User Licenses](#) [User Licenses Help](#)

In this step, I configured the basic organizational details for our project. Setting the organization name to "**Integrated Banking Hub**" and the timezone to "**India Standard Time**" is a critical first step. This ensures that all timestamps on records and all reports reflect the correct local time and currency for our operations.

2. Business Hours Setup

The screenshot shows the 'Organization Business Hours' page in the Salesforce Setup. It lists two entries:

- Default**: Active, Pacific Daylight Time (America/Los_Angeles)
- Standard Banking Hours**: Not Active, India Standard Time (Asia/Kolkata)

A red box highlights the 'Standard Banking Hours' row.

The screenshot shows the 'Business Hours Detail' page for the 'Standard Banking Hours' record. It includes the following details:

- Business Hours Name:** Standard Banking Hours
- Time Zone:** (GMT+05:30) India Standard Time (Asia/Kolkata)
- Business Hours:**

Day	Time
Monday	9:00 AM to 6:00 PM
Tuesday	9:00 AM to 6:00 PM
Wednesday	9:00 AM to 6:00 PM
Thursday	9:00 AM to 6:00 PM
Friday	9:00 AM to 6:00 PM
- Holidays:** [Empty]
- Active:**
- Created By:** Kundana Sree Bavisetti | 9/17/2025, 12:53 PM
- Last Modified By:** Kundana Sree Bavisetti | 9/17/2025, 12:53 PM

To support our automation rules and to define the bank's operational times, I have created a new business hours record. I named it "**Standard Banking Hours**" and set the working days from Monday to Friday, 9:00 AM to 6:00 PM. This will be used in a later phase for things like setting deadlines for automated processes.

3. Fiscal Year Setup

The screenshot shows the 'Fiscal Year' setup page in the Salesforce Setup interface. The 'Change Fiscal Year Period' dialog box is open, showing the following details:

- Name: Integrated Banking Hub
- Fiscal Year Start Month: April
- Fiscal Year Is Based On: The starting month (selected)

A warning message in the dialog box states: "Changing the fiscal year shifts fiscal periods and impacts opportunities and forecasts across your organization. If your forecast periods are set to quarterly, adjusting the fiscal year start month will erase existing forecast adjustments and quotas. Consider exporting a data backup before implementing this change."

I have set the fiscal year for our organization to a **Standard Fiscal Year** starting in **April**. This is a best practice for financial reporting in India and will be crucial for creating accurate dashboards and reports in the later phases of the project.

4. User Setup (Profiles & Roles)

The screenshot shows the 'Profiles' setup page in the Salesforce Setup interface. The 'Loan Officer Profile' is selected. The profile details are as follows:

- Name: Loan Officer Profile
- User License: Salesforce
- Description: (empty)
- Created By: Kundana Sree Bavisetti, 9/18/2025, 6:29 AM
- Modified By: Kundana Sree Bavisetti, 9/18/2025, 6:29 AM

The 'Page Layouts' section shows the following assignments:

Object	Layout	Assignment
Global	Global Layout [View Assignment]	Location Group Assignment [View Assignment]
Email Application	Not Assigned [View Assignment]	Macro [View Assignment]
Home Page Layout	Home Page Default [View Assignment]	Object Milestone [View Assignment]
Account	Account Layout [View Assignment]	Operating Hours [View Assignment]
Alternative Payment Method	Alternative Payment Method Layout [View Assignment]	Opportunity [View Assignment]
Appointment Invitation	Appointment Invitation Layout [View Assignment]	Opportunity Product [View Assignment]
Asset	Asset Layout [View Assignment]	Order [View Assignment]

SETUP Profiles

Wealth Advisor Profile

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

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

[Login IP Ranges \[0\]](#) | [Enabled Apex Class Access \[0\]](#) | [Enabled Visualforce Page Access \[0\]](#) | [Enabled External Data Source Access \[0\]](#) | [Enabled Named Credential Access \[0\]](#) | [Enabled External Credential Principal Access \[0\]](#) | [Enabled Custom Metadata Type Access \[0\]](#) | [Enabled Custom Setting Definitions Access \[0\]](#) | [Enabled Flow Access \[0\]](#) | [Enabled Service Presence Status Access \[0\]](#) | [Enabled Custom Permissions \[0\]](#)

Profile Detail	
Name	Wealth Advisor Profile
User License	Salesforce
Description	
Created By	Kundana Sree Bavisetti, 9/18/2025, 6:29 AM
Modified By	Kundana Sree Bavisetti, 9/18/2025, 6:29 AM
Edit Clone Delete View Users	

Page Layouts			
Standard Object Layouts			
Global	Global Layout [View Assignment]	Location Group Assignment	Location Group Assignment Layout [View Assignment]
Email Application	Not Assigned [View Assignment]	Macro	Macro Layout [View Assignment]
Home Page Layout	Home Page Default [View Assignment]	Object Milestone	Object Milestone Layout [View Assignment]
Account	Account Layout [View Assignment]	Operating Hours	Operating Hours Layout [View Assignment]
Alternative Payment Method	Alternative Payment Method Layout [View Assignment]	Opportunity	Opportunity Layout [View Assignment]
Appointment Invitation	Appointment Invitation Layout [View Assignment]	Opportunity Product	Opportunity Product Layout [View Assignment]
Asset	Asset Layout [View Assignment]	Order	Order Layout [View Assignment]

SETUP Profiles

Compliance Profile

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

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

[Login IP Ranges \[0\]](#) | [Enabled Apex Class Access \[0\]](#) | [Enabled Visualforce Page Access \[0\]](#) | [Enabled External Data Source Access \[0\]](#) | [Enabled Named Credential Access \[0\]](#) | [Enabled External Credential Principal Access \[0\]](#) | [Enabled Custom Metadata Type Access \[0\]](#) | [Enabled Custom Setting Definitions Access \[0\]](#) | [Enabled Flow Access \[0\]](#) | [Enabled Service Presence Status Access \[0\]](#) | [Enabled Custom Permissions \[0\]](#)

Profile Detail	
Name	Compliance Profile
User License	Salesforce
Description	
Created By	Kundana Sree Bavisetti, 9/18/2025, 6:30 AM
Modified By	Kundana Sree Bavisetti, 9/18/2025, 6:30 AM
Edit Clone Delete View Users	

Page Layouts			
Standard Object Layouts			
Global	Global Layout [View Assignment]	Location Group Assignment	Location Group Assignment Layout [View Assignment]
Email Application	Not Assigned [View Assignment]	Macro	Macro Layout [View Assignment]
Home Page Layout	Home Page Default [View Assignment]	Object Milestone	Object Milestone Layout [View Assignment]
Account	Account Layout [View Assignment]	Operating Hours	Operating Hours Layout [View Assignment]
Alternative Payment Method	Alternative Payment Method Layout [View Assignment]	Opportunity	Opportunity Layout [View Assignment]
Appointment Invitation	Appointment Invitation Layout [View Assignment]	Opportunity Product	Opportunity Product Layout [View Assignment]
Asset	Asset Layout [View Assignment]	Order	Order Layout [View Assignment]

The screenshot shows the Salesforce Setup Roles page. On the left, a sidebar lists categories like Users, Roles (selected), Feature Settings, Sales, Service, and Case Teams. The Roles section is expanded, showing Contact Roles on Contracts, Contact Roles on Opportunities, and Contact Roles on Cases. Below this, a note says " Didn't find what you're looking for? Try using Global Search." The main content area is titled "Role Manager". It displays the Manager role details: Label: Manager, This role reports to: CEO, Modified By: Kundana Sree Bavisetti, 9/18/2025, 6:34 AM. Opportunity Access and Case Access are also listed. A table shows Role Name and Manager. At the bottom, there's a "Users in Manager Role" section with a note "No records to display".

The Role Hierarchy confirms the organizational structure, placing the 'Manager' role above the specialized roles. This allows for controlled data visibility and reporting alignment."

The screenshot shows the Salesforce Setup Roles page with a focus on creating a new role hierarchy. The title is "Creating the Role Hierarchy". A note says "You can build on the existing role hierarchy shown on this page. To insert a new role, click **Add Role**". Below this, a section titled "Your Organization's Role Hierarchy" shows a tree structure under "Integrated Banking Hub":

- Integrated Banking Hub**
 - CEO** (Edit | Del | Assign)
 - Manager** (Edit | Del | Assign)
 - Compliance Officer** (Edit | Del | Assign)
 - Loan Officer** (Edit | Del | Assign)
 - Wealth Advisor** (Edit | Del | Assign)

To enforce our project's security model, I have created three custom profiles: **Loan Officer Profile**, **Wealth Advisor Profile**, and **Compliance Profile**. I have also set up a clean and organized role hierarchy that reflects our project's structure, with the “Manager” role overseeing the other three specialized roles. This ensures a clear chain of command and data visibility.

User Creation

The screenshot shows the Salesforce Setup interface under the 'Users' section. A new user record is being created for a 'Compliance Officer'. The user details include:

- Name:** Compliance Officer
- Alias:** coffi
- Email:** complianceofficer@bankinghub.com (Verify)
- Username:** complianceofficer@bankinghub.com
- Nickname:** User17582082657008305157
- Title:**
- Company:**
- Department:**
- Division:**
- Address:**
- Time Zone:** (GMT+05:30) India Standard Time (Asia/Kolkata)
- Locale:** English (India)
- Language:** English
- Delegated Approver:**
- Manager:**
- Receive Approval Request Emails:** Only if I am an approver
- Federation ID:**
- App Registration: One-Time Password Authenticator:**
- App Registration: Salesforce:**

Role: Compliance Officer

User License: Salesforce Platform

Profile: Standard Platform User

Active:

Marketing User:

Offline User:

Knowledge User:

Flow User:

Service Cloud User:

Site.com Contributor User:

Site.com Publisher User:

WDC User:

Mobile Push Registrations: View

Data.com User Type:

Accessibility Mode (Classic Only): [i](#)

Debug Mode: [i](#)

High-Contrast Palette on Charts: [i](#)

Load Lightning Pages While Scrolling: [i](#)

Salesforce CRM Content User:

Buttons at the top: **Edit**, **Sharing**, **Reset Password**, **Freeze**, **View Summary**.

To test our application properly, I have created a user for each of the core roles in our project. Each user has been assigned the appropriate profile and role, ensuring they have the correct permissions to access the data and features relevant to their job.

5. OWD (Org-Wide Defaults) & Sharing Rules

Sharing Configuration Set	Default Internal Access	Default External Access	Shareable
Streaming Channel	Public Read Only	Private	<input checked="" type="checkbox"/>
Tableau Host Mapping	Public Read/Write	Private	<input checked="" type="checkbox"/>
User Presence	Public Read Only	Private	<input checked="" type="checkbox"/>
Waitlist	Private	Private	<input checked="" type="checkbox"/>
Web Cart Document	Private	Private	<input checked="" type="checkbox"/>
Work Order	Private	Private	<input checked="" type="checkbox"/>
Work Plan	Private	Private	<input checked="" type="checkbox"/>
Work Plan Template	Private	Private	<input checked="" type="checkbox"/>
Work Step Template	Private	Private	<input checked="" type="checkbox"/>
Work Type	Private	Private	<input checked="" type="checkbox"/>
Work Type Group	Public Read/Write	Private	<input checked="" type="checkbox"/>
Audit Log	Private	Private	<input checked="" type="checkbox"/>
Compliance Regulation	Private	Private	<input checked="" type="checkbox"/>
Financial Asset	Private	Private	<input checked="" type="checkbox"/>
Financial Goal	Private	Private	<input checked="" type="checkbox"/>
Investment Portfolio	Private	Private	<input checked="" type="checkbox"/>
Loan Application	Private	Private	<input checked="" type="checkbox"/>
Risk Event	Private	Private	<input checked="" type="checkbox"/>

Other Settings

Standard Report Visibility Manual User Record Sharing Manager Groups Secure guest user record access Require permission to view record names in lookup fields

In **Setup**, I navigated to **Sharing Settings** and clicked **Edit**.

I set the Default Internal Access for all custom objects (e.g., Loan Application, Investment Portfolio, Compliance Regulation, Collateral, Repayment Schedule, Financial Asset, Financial Goal, Risk Event, Audit Log) to **Private**.

Setting Org-Wide Defaults (OWD) to **Private** for all financial objects (e.g., Loan Application, Investment Portfolio) is a core security best practice. This 'restrictive first' approach guarantees data is only shared through controlled mechanisms like Roles or Sharing Rules, adhering to strict financial compliance rules.

Phase 3: Data Modeling & Relationships

1. Custom Objects

The following are the custom objects in each module that I have created

Loan Module Objects:

- Loan Application
- Collateral
- Repayment Schedule

Wealth Module Objects:

- Investment Portfolio
- Financial Asset
- Financial Goal

Compliance Module Objects:

- Compliance Regulation
- Risk Event
- Audit Log

2. Custom Fields

After creating the custom objects, I added the necessary fields to store specific data. As shown in the below screenshots.

Loan Application Module:

Use Case: Stores all loan request details (type, amount, risk score) and drives the approval workflow.

Relationship : Master-Detail to Account (Client).

- **Loan Application Object fields:**

The custom fields were engineered to capture all necessary business data. For instance, the Loan Application requires Risk_Score__c, and the Investment Portfolio requires the calculated Total_Value__c. This structure ensures data quality and completeness for subsequent automations.

SETUP > OBJECT MANAGER Loan Application				
Details	Fields & Relationships		Quick Find	New Deleted Fields Field Dependencies Set History Tracking
Fields & Relationships	12 Items, Sorted by Field Label			
Page Layouts	Applicant Name	Applicant_Name__c	Text(80)	
Lightning Record Pages	Application Date	Application_Date__c	Date	
Buttons, Links, and Actions	Application Status	Application_Status__c	Picklist	
Compact Layouts	Created By	CreatedById	Lookup(User)	
Field Sets	Credit Score	Credit_Score__c	Number(18, 0)	
Object Limits	Last Modified By	LastModifiedById	Lookup(User)	
Record Types	Loan Amount	Loan_Amount__c	Currency(16, 2)	
Related Lookup Filters	Loan Application Name	Name	Text(80)	✓
Search Layouts	Loan Type	Loan_Type__c	Picklist	
List View Button Layout	Owner	OwnerId	Lookup(User,Group)	✓
Restriction Rules	Risk Score	Risk_Score__c	Number(18, 0)	
Scoping Rules				
Object Access				

- **Collateral Object fields:**

SETUP > OBJECT MANAGER Collateral				
Details	Fields & Relationships		Quick Find	New Deleted Fields Field Dependencies Set History Tracking
Fields & Relationships	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD INDEXED
Page Layouts	Collateral Name	Name	Text(80)	✓
Lightning Record Pages	Collateral Type	Collateral_Type__c	Picklist	
Buttons, Links, and Actions	Created By	CreatedById	Lookup(User)	
Compact Layouts	Description	Description__c	Long Text Area(256)	
Field Sets	Last Modified By	LastModifiedById	Lookup(User)	
Object Limits	Loan Application	Loan_Application__c	Master-Detail(Loan Application)	✓
Record Types	Value	Value__c	Currency(16, 2)	
Related Lookup Filters				
Restriction Rules				
Scoping Rules				
Object Access				
Triggers				
Flow Triggers				

- **Repayment Schedule Object fields:**

SETUP > OBJECT MANAGER Repayment Schedule					
Details	Fields & Relationships 7 Items, Sorted by Field Label				
Fields & Relationships	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Amount Due	Amount_Due__c	Currency(16, 2)		
Lightning Record Pages	Created By	CreatedById	Lookup(User)		
Buttons, Links, and Actions	Last Modified By	LastModifiedById	Lookup(User)		
Compact Layouts	Loan Application	Loan_Application__c	Master-Detail(Loan Application)	✓	▼
Field Sets	Payment Due Date	Payment_Due_Date__c	Date		▼
Object Limits	Payment Status	Payment_Status__c	Picklist		▼
Record Types	Repayment Schedule Name	Name	Text(80)	✓	▼
Related Lookup Filters					
Restriction Rules					
Scoping Rules					
Object Access					
Triggers					
Flow Triggers					

Wealth Module:

- Investment Portfolio Object fields:

SETUP > OBJECT MANAGER
Investment Portfolio

Details	Fields & Relationships				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Created By	CreatedById	Lookup(User)		
Lightning Record Pages	Investment Portfolio Name	Name	Text(80)		✓
Buttons, Links, and Actions	Last Modified By	LastModifiedById	Lookup(User)		
Compact Layouts	Owner	OwnerId	Lookup(User,Group)		✓
Field Sets	Total Value	Total_Value__c	Currency(18, 0)		
Object Limits	User	User__c	Lookup(User)		✓
Record Types					
Related Lookup Filters					
Restriction Rules					
Scoping Rules					
Object Access					
Triggers					
Flow Triggers					

<https://orgfarm-a658e5a554-dev-ed.develop.lightning.force.com/one/one.app#/setup/ObjectManager/01gK000002MAN/FieldsAndRelationships/view>

- Financial Asset Object fields:

SETUP > OBJECT MANAGER
Financial Asset

Details	Fields & Relationships				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Asset Type	Asset_Type__c	Picklist		
Lightning Record Pages	Created By	CreatedById	Lookup(User)		
Buttons, Links, and Actions	Current Value	Current_Value__c	Currency(18, 0)		
Compact Layouts	Financial Asset Name	Name	Text(80)		✓
Field Sets	Last Modified By	LastModifiedById	Lookup(User)		
Object Limits	Owner	OwnerId	Lookup(User,Group)		✓
Record Types	Purchase Date	Purchase_Date__c	Date		
Related Lookup Filters	Purchase Price	Purchase_Price__c	Currency(18, 0)		
Restriction Rules					
Scoping Rules					
Object Access					
Triggers					
Flow Triggers					

- **Financial Goal Object fields:**

Fields & Relationships 6 Items, Sorted by Field Label					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
	Completion Date	Completion_Date__c	Date		
	Created By	CreatedById	Lookup(User)		
	Financial Goal Name	Name	Text(80)		✓
	Last Modified By	LastModifiedById	Lookup(User)		
	Owner	OwnerId	Lookup(User,Group)		✓
	Target Amount	Target_Amount__c	Currency(18, 0)		

Compliance Module:

- Compliance Regulation Object fields:

SETUP > OBJECT MANAGER Compliance Regulation					
Details	Fields & Relationships				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Compliance Regulation Name	Name	Text(80)		✓
Lightning Record Pages	Created By	CreatedBy	Lookup(User)		▼
Buttons, Links, and Actions	Last Modified By	LastModifiedBy	Lookup(User)		▼
Compact Layouts	Last Reviewed Date	Last_Reviewed_Date__c	Date		▼
Field Sets	Owner	OwnerId	Lookup(User,Group)		▼
Object Limits	Regulation Status	Regulation_Status__c	Picklist		▼
Record Types					
Related Lookup Filters					
Restriction Rules					
Scoping Rules					
Object Access					
Triggers					
Flow Triggers					

- Risk Event Object fields:

SETUP > OBJECT MANAGER Risk Event					
Details	Fields & Relationships				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Created By	CreatedBy	Lookup(User)		▼
Lightning Record Pages	Impact	Impact__c	Picklist		▼
Buttons, Links, and Actions	Last Modified By	LastModifiedBy	Lookup(User)		▼
Compact Layouts	Owner	OwnerId	Lookup(User,Group)		✓
Field Sets	Risk Event Name	Name	Text(80)		▼
Object Limits	Risk Type	Risk_Type__c	Picklist		▼
Record Types					
Related Lookup Filters					
Restriction Rules					
Scoping Rules					
Object Access					
Triggers					
Flow Triggers					

- **Audit Log Object fields:**

Fields & Relationships 6 Items, Sorted by Field Label					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Action Type	Action_Type__c	Text(255)		
Lightning Record Pages	Audit Log Name	Name	Text(80)		✓
Buttons, Links, and Actions	Created By	CreatedById	Lookup(User)		
Compact Layouts	Last Modified By	LastModifiedById	Lookup(User)		
Field Sets	Owner	OwnerId	Lookup(User,Group)		✓
Object Limits	Timestamp	Timestamp__c	Date/Time		
Record Types					
Related Lookup Filters					
Restriction Rules					
Scoping Rules					
Object Access					
Triggers					
Flow Triggers					

3. Relationships

To connect our different data modules, I established a series of relationships.

The use of **Master-Detail Relationships** guarantees the integrity of our financial data. A Financial Asset is dependent on its parent Investment Portfolio, ensuring that records cannot exist in isolation, which is vital for accurate roll-up summaries and reporting.

Use Case: Stores all loan request details (type, amount, risk score) and drives the approval workflow.

SETUP > OBJECT MANAGER Collateral					
Details	Fields & Relationships 7 Items, Sorted by Field Label				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
	Collateral Name	Name	Text(80)		▼
	Collateral Type	Collateral_Type__c	Picklist		▼
	Created By	CreatedById	Lookup(User)		▼
	Description	Description__c	Long Text Area(256)		▼
	Last Modified By	LastModifiedById	Lookup(User)		▼
	Loan Application	Loan_Application__c	Master-Detail(Loan Application)		▼
	Value	Value__c	Currency(16, 2)		▼

I created a **Master-Detail Relationship** between the **Loan Application** object and the **Collateral** object. This creates a strong link, so a collateral record cannot exist without a parent loan application.

SETUP > OBJECT MANAGER Repayment Schedule					
Details	Fields & Relationships 7 Items, Sorted by Field Label				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
	Amount Due	Amount_Due__c	Currency(16, 2)		▼
	Created By	CreatedById	Lookup(User)		▼
	Last Modified By	LastModifiedById	Lookup(User)		▼
	Loan Application	Loan_Application__c	Master-Detail(Loan Application)		▼
	Payment Due Date	Payment_Due_Date__c	Date		▼
	Payment Status	Payment_Status__c	Picklist		▼
	Repayment Schedule Name	Name	Text(80)		▼

Similar to Collateral object , I created another **Master-Detail Relationship** between **loan application** and **repayment schedule** object.

Use Case : Tracks individual assets (stocks, bonds, mutual funds) within a portfolio.

SETUP > OBJECT MANAGER Financial Asset							
Fields & Relationships	Fields & Relationships		Quick Find	New	Deleted Fields	Field Dependencies	Set History Tracking
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED		
	Asset Type	Asset_Type__c	Picklist				
	Created By	CreatedById	Lookup(User)				
	Current Value	Current_Value__c	Currency(18, 0)				
	Financial Asset Name	Name	Text(80)	✓			
	Investment Portfolio	Investment_Portfolio__c	Master-Detail(Investment Portfolio)	✓			
	Last Modified By	LastModifiedById	Lookup(User)				
	Purchase Date	Purchase_Date__c	Date				
	Purchase Price	Purchase_Price__c	Currency(18, 0)				

here I created **Master-Detail Relationship** for **financial asset** to **investment portfolio**

SETUP > OBJECT MANAGER Financial Goal							
Fields & Relationships	Fields & Relationships		Quick Find	New	Deleted Fields	Field Dependencies	Set History Tracking
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED		
	Completion Date	Completion_Date__c	Date				
	Created By	CreatedById	Lookup(User)				
	Financial Goal Name	Name	Text(80)	✓			
	Investment Portfolio	Investment_Portfolio__c	Master-Detail(Investment Portfolio)	✓			
	Last Modified By	LastModifiedById	Lookup(User)				
	Target Amount	Target_Amount__c	Currency(18, 0)				

here I created **Master-Detail Relationship** for **financial goal** to **investment portfolio**

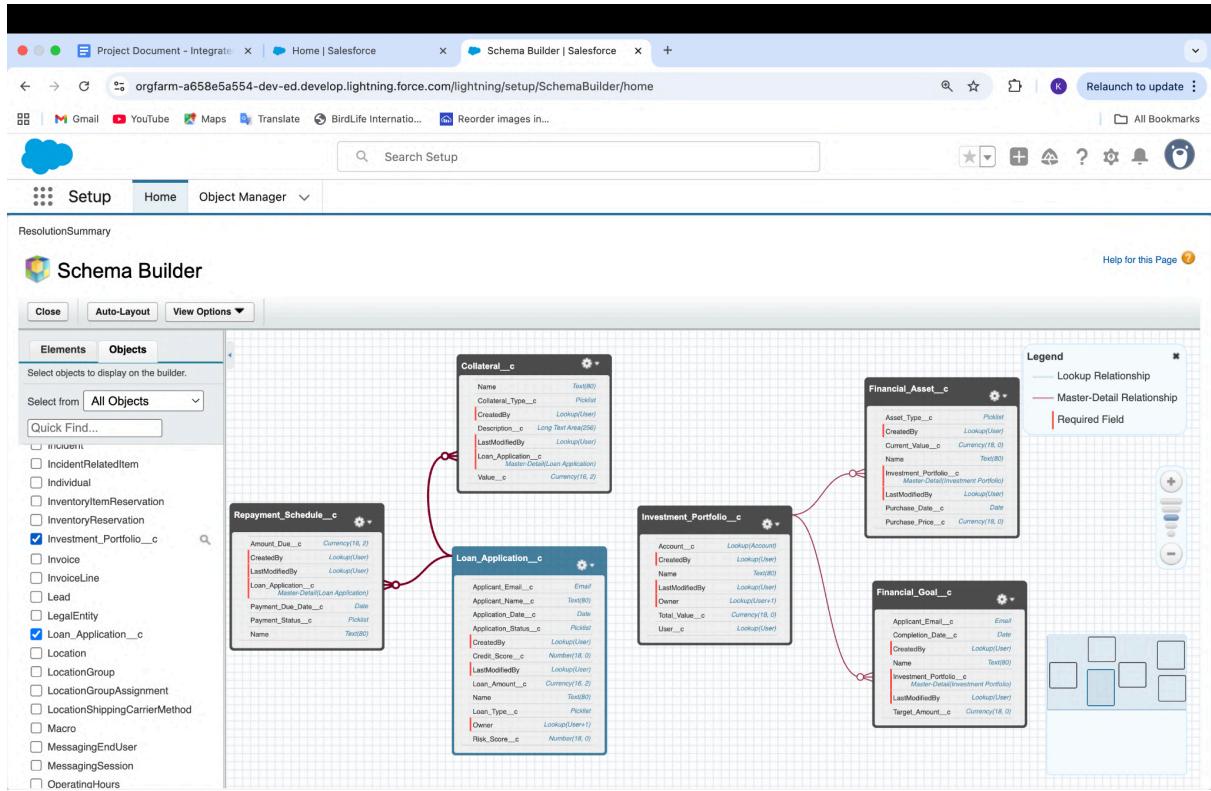
Use Case : The primary container for a client's investment strategy and calculated total value.

Fields & Relationships					
Details		7 Items, Sorted by Field Label	Quick Find	New Deleted Fields Field Dependencies Set History Tracking	
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Account	Account__c	Lookup(Account)		✓
Lightning Record Pages	Created By	CreatedById	Lookup(User)		▼
Buttons, Links, and Actions	Investment Portfolio Name	Name	Text(80)	✓	▼
Compact Layouts	Last Modified By	LastModifiedById	Lookup(User)		▼
Field Sets	Owner	OwnerId	Lookup(User,Group)	✓	
Object Limits	Total Value	Total_Value__c	Currency(18, 0)		▼
Record Types	User	User__c	Lookup(User)	✓	▼
Related Lookup Filters					
Restriction Rules					
Scoping Rules					
Object Access					
Triggers					
Flow Triggers					

here I created the **lookup-relationship** for **investment portfolio** to **account**

Schema Builder :

The Schema Builder visually confirms the complete data model: three independent modules (Loan, Wealth, Compliance) anchored to the standard Account object. This centralized structure is the foundation of the single source of truth (SSOT) concept for this project



Phase 4: Process Automation (Admin)

1. Validation Rules

- **Use Case:** To enforce financial data integrity by preventing users from submitting loan applications with invalid data, such as a loan amount of zero or less.
- **Implementation:** A Validation Rule on the Loan Application object prevents saving the record if `Loan_Amount__c <= 0`.
- **Business Impact:** Eliminates inaccurate data from entering the approval pipeline, saving time and preventing errors.

The screenshot shows the Salesforce Object Manager interface. At the top, there's a blue header bar with the word "SETUP" and a gear icon. Below it, the "Object Manager" page title is displayed. Underneath, the specific section for the "Loan Application Validation Rule" is shown. The rule details are listed in a table format:

Validation Rule Detail	
Rule Name	Loan_Amount_Cannot_Be_Zero
Error Condition Formula	<code>OR(ISBLANK(Loan_Amount__c), Loan_Amount__c = 0)</code>
Error Message	Loan Amount cannot be zero or empty.
Description	
Created By	Kundana Sree Bavisetti, 9/19/2025, 6:09 AM
Active	<input checked="" type="checkbox"/>
Error Location	Loan Amount
Modified By	Kundana Sree Bavisetti, 9/19/2025, 6:09 AM

At the bottom of the table, there are "Edit" and "Clone" buttons. The entire screenshot has a light gray background.

I created a validation rule on the Loan Application object. The rule's formula, `OR(ISBLANK(Loan_Amount__c), Loan_Amount__c < 0)`, ensures that the Loan Amount field is not empty and is not a negative number.

This is a crucial step for data integrity. The rule prevents users from submitting incomplete or invalid loan applications, ensuring that our data is accurate from the moment it is entered into the system.

2. Approval Process

- **Use Case:** To enforce governance on high-value transactions. Any loan application greater than or equal to **₹20,00,000** must be automatically routed to the **Manager** profile for mandatory approval.
- **Implementation:** Standard Approval Process with an entry criteria of `Loan_Amount__c >= 20000000` and a final approval action that updates the Status to 'Approved'.

The screenshot shows the Salesforce Setup interface for managing Approval Processes. The specific process shown is 'Loan Application: Loan Approval Process'. The 'Process Definition Detail' section includes fields like Process Name (Loan Approval Process), Unique Name (Loan_Approval_Process), Description (This process routes all loan applications over ₹20,00,000 to the Loan Manager for approval), Entry Criteria (Loan Application: Loan Amount GREATER THAN 2000000), Record Editability (Administrator ONLY), and Approval Assignment Email Template (Initial Submitters: Loan Application Owner). The 'Initial Submission Actions' section contains a single 'Record Lock' action. The 'Approval Steps' section is currently empty, indicated by a red X icon. The 'Final Approval Actions' section contains a single 'Record Lock' action. The 'Final Rejection Actions' section is empty. The bottom of the page includes navigation links '^ Back To Top' and 'Always show me ▾ more records per related list'.

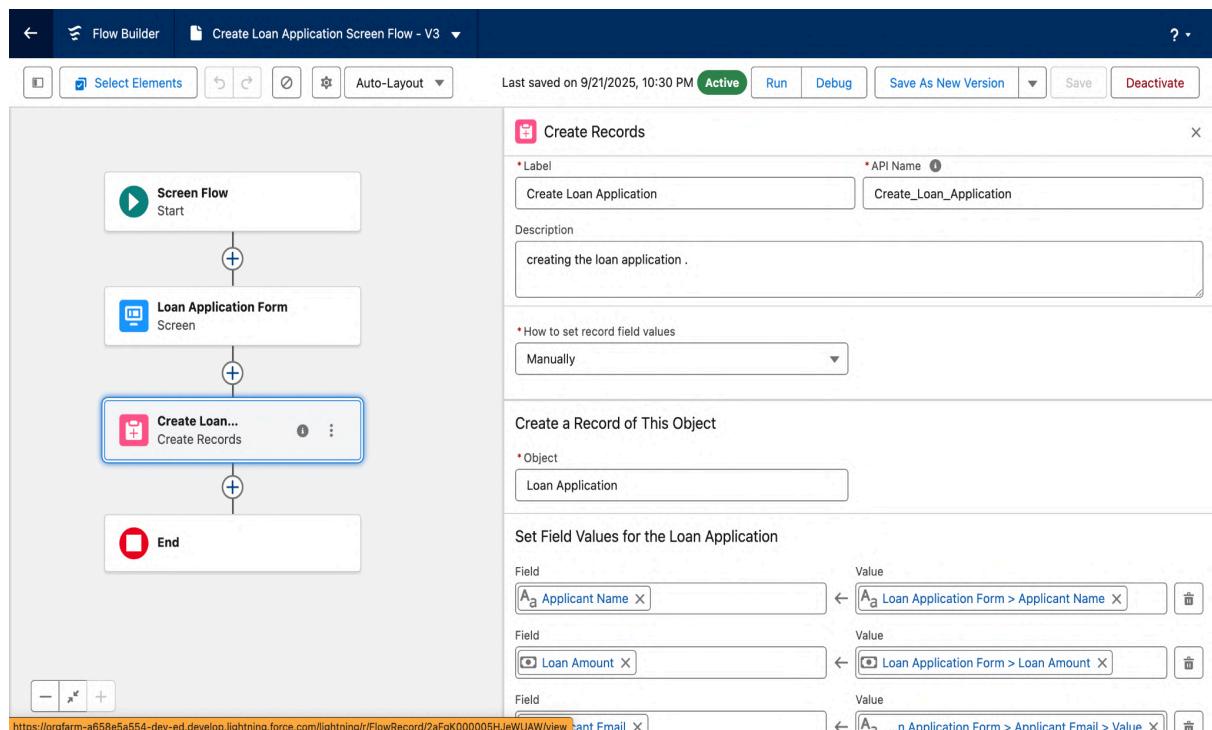
I set up a multi-step approval process for the Loan Application object. The entry criteria, `Loan Application: Loan Amount GREATER THAN 2000000`, routes high-value loan applications to a manager. The record is locked from editing during this process.

This process is essential for risk management and compliance. It automates the review of high-risk loans, ensuring that senior management signs off on them before they are approved. This reduces financial risk and provides a clear audit trail.

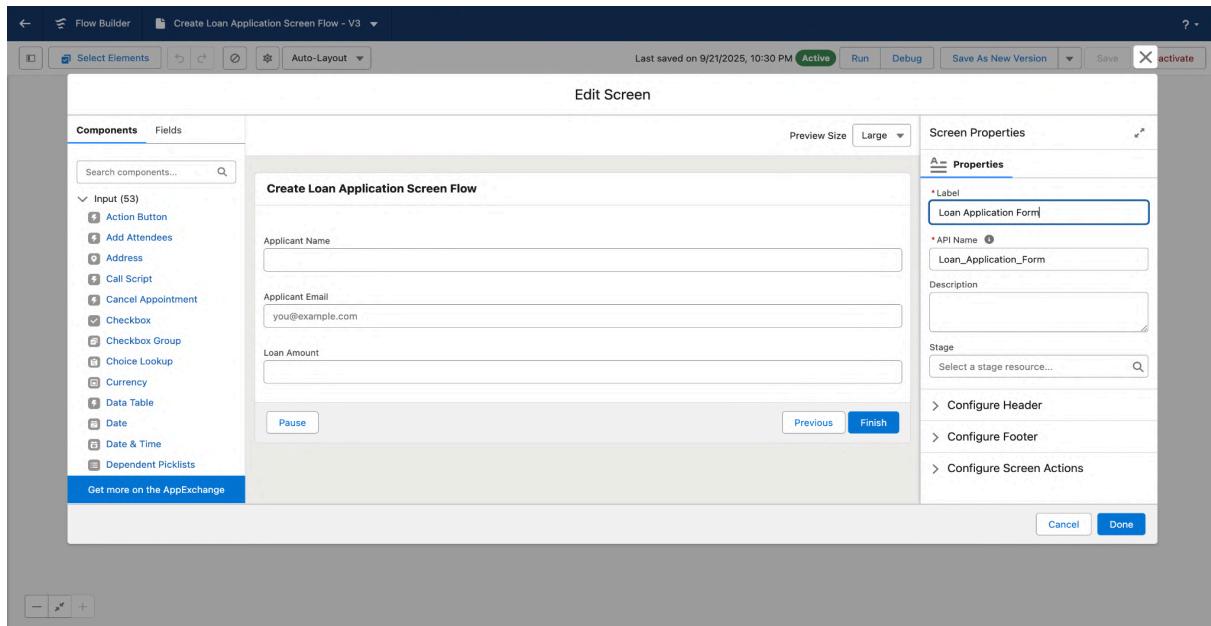
3. Flows

- **Use Case:** To automatically calculate and assign a **Risk Score** (e.g., 1-5) to a loan application immediately upon creation, allowing Loan Officers to triage applications quickly.
- **Implementation:** A **Record-Triggered Flow** (after-save) calculates a value for `Risk_Score__c` based on criteria like Credit Score and Loan Type.

Create loan Application Screen flow :



Loan Application form flow element :



Create Loan Application flow Element :

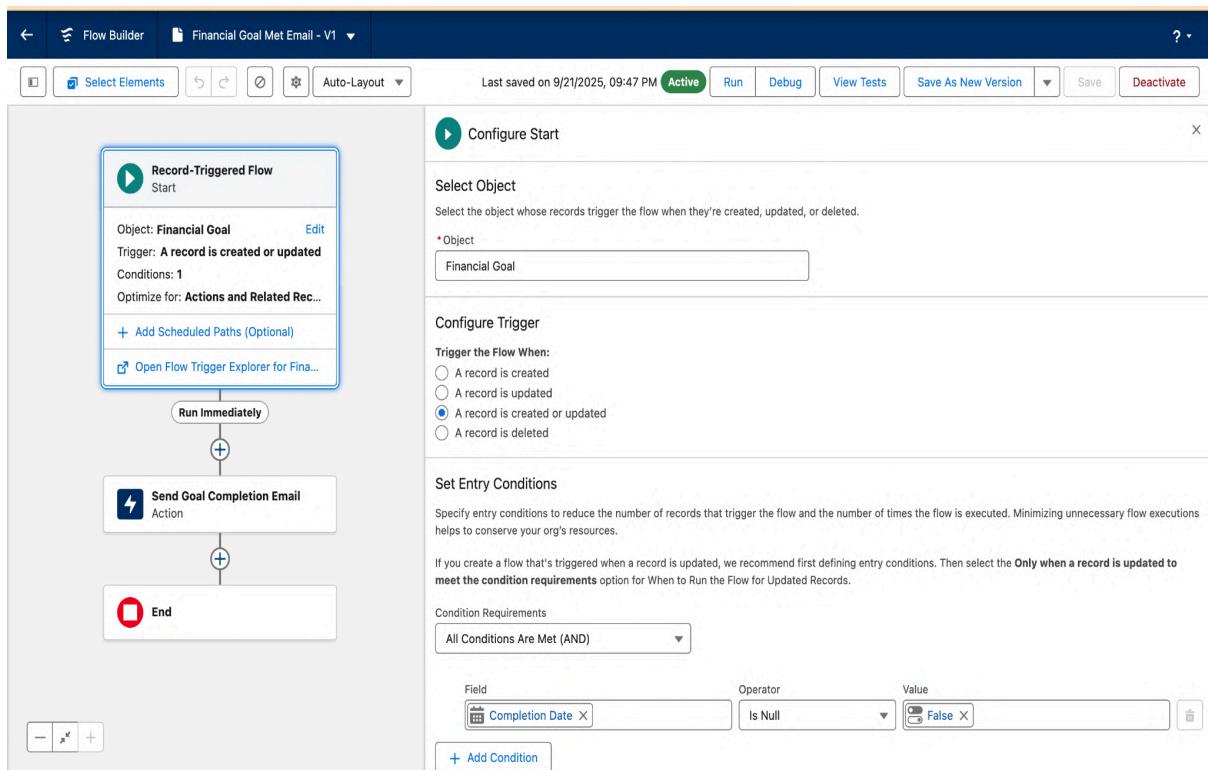
The screenshot shows the configuration of a 'Create Records' element within a Screen Flow. The element has the following settings:

- Label:** Create Loan Application
- API Name:** Create_Loan_Application
- Description:** creating the loan application .
- How to set record field values:** Manually
- Create a Record of This Object:** Loan Application
- Set Field Values for the Loan Application:** Three fields are mapped:
 - Field: A_a Applicant Email → Value: A_a ...an Application Form > Applicant Email > Value
 - Field: A_a Applicant Name → Value: A_a Loan Application Form > Applicant Name
 - Field: A_a Loan Amount → Value: A_a Loan Application Form > Loan Amount
- Add Field:** + Add Field
- Manually assign variables (advanced):** Unchecked
- Check for Matching Records:** Disabled (switch is off)

I built a Screen Flow that guides the user through creating a new Loan Application record. This flow uses three key components: Applicant Name, Loan Amount, and Applicant Email to collect data and then uses a Create Records element to save it to the database.

This flow provides a user-friendly and consistent experience for loan officers. By guiding them through a specific set of fields, it ensures all necessary information is collected correctly, reducing errors and saving time.

Financial Goal Met Email Flow :



Send Goal Completion Email Flow Element :

The screenshot shows the configuration for a 'Send Email' flow element. It includes fields for Label ('Send Goal Completion Email'), API Name ('Send_Goal_Completion_Email'), and a Description box containing the note: 'it will send the completion email as soon as the goal is completed'.

Use values from earlier in the flow to set the inputs for the "Send Email" core action. To use its outputs later in the flow, store them in variables.

Set Input Values

Configure Recipient Details

A_a Recipient Address List i

X

A_a Recipient Address Collection i

🔍

A_a CC Recipient Address List i

🔍

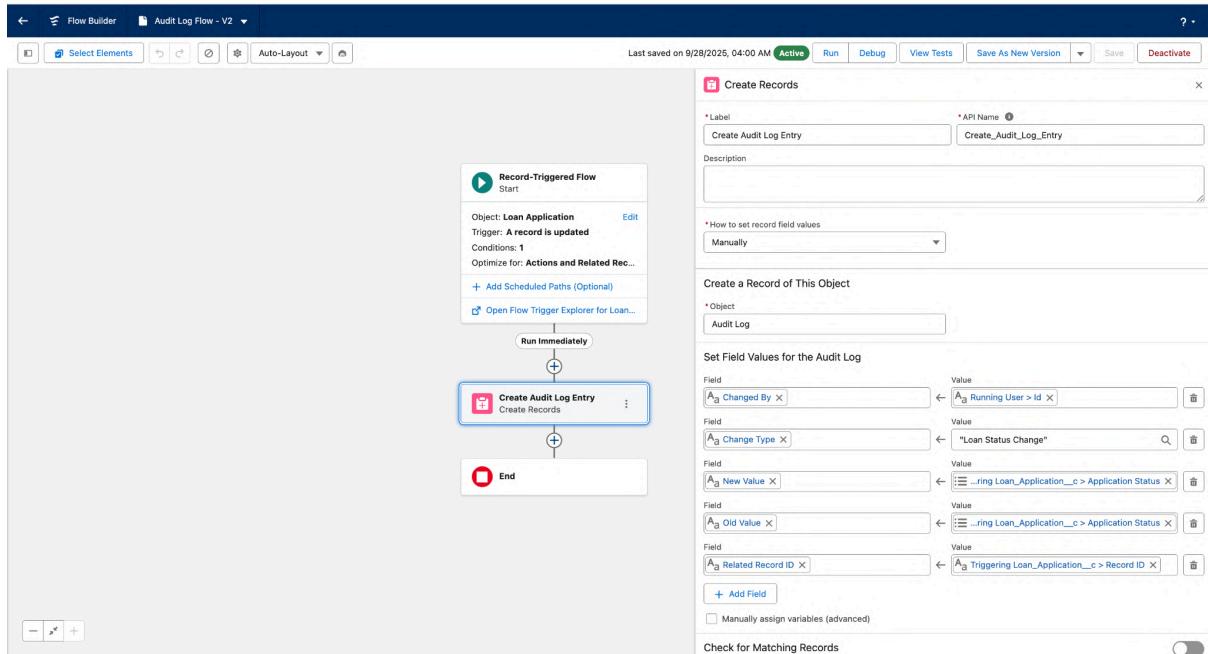
A_a CC Recipient Address Collection i

🔍

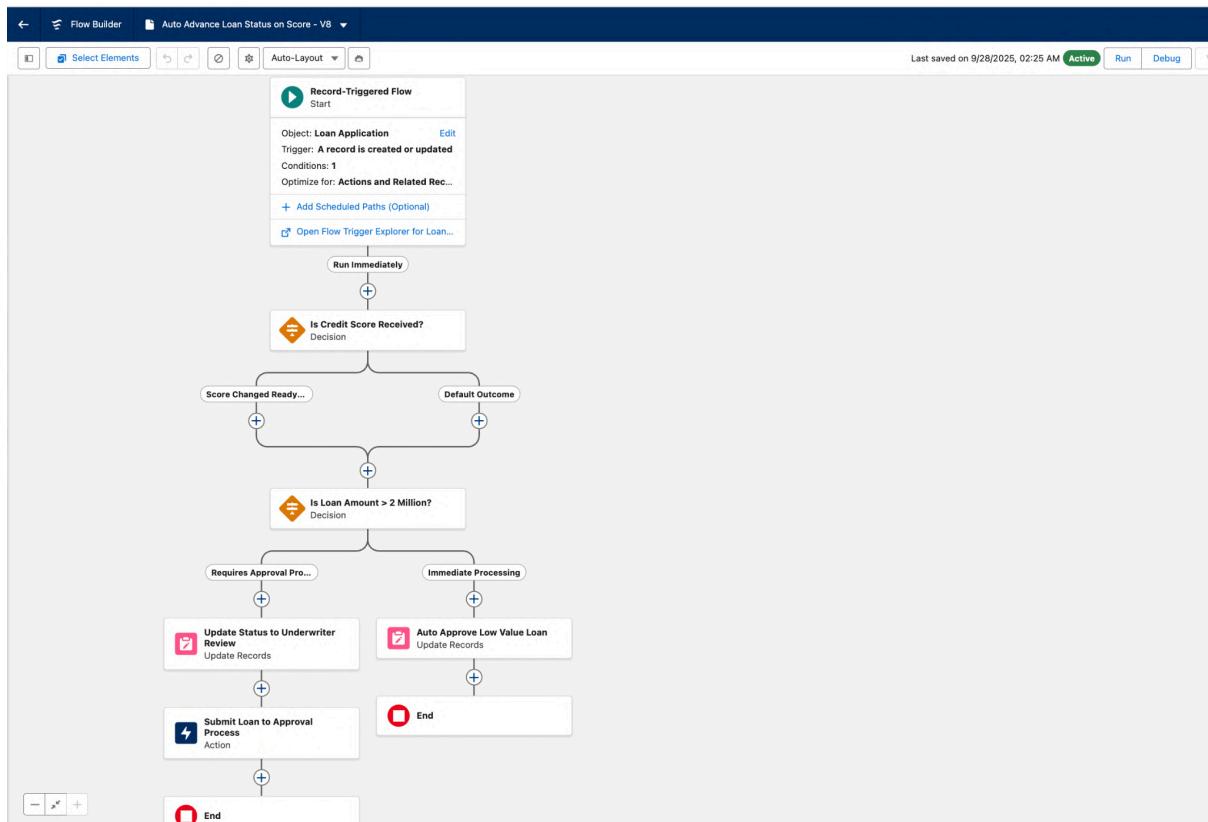
I created a Record-Triggered Flow that runs automatically when a Financial Goal record is created or updated. The flow checks if the Completion Date field is not null (IsNull False). If the condition is met, it will trigger an email alert.

This flow automates client communication, which is a key part of customer relationship management. It ensures that clients are instantly notified when they achieve a financial goal, which enhances the client experience and builds trust.

Audit Log Flow :



Auto Advance Loan Status on Score :



Is Credit Score Received? Element :

Decision

* Label	* API Name <small>i</small>												
Is Credit Score Received?	Is_Credit_Score_Received												
Description													
<p>Outcomes For each path the flow can take, create an outcome. For each outcome, specify the conditions that must be met for the flow to take that path.</p> <table border="1"> <tr> <th>OUTCOME ORDER</th> <th>OUTCOME DETAILS</th> </tr> <tr> <td>Score Changed Ready for Review</td> <td>* Label: Score Changed Ready for Review * Outcome API Name <small>i</small>: Score_Changed_Ready_for_Review</td> </tr> <tr> <td>Default Outcome</td> <td> <p>Condition Requirements to Execute Outcome</p> <p>All Conditions Are Met (AND)</p> <table border="1"> <tr> <td>Resource: ?Is Credit Score Received X</td> <td>Operator: Was Visited</td> <td>Value: True X</td> </tr> <tr> <td colspan="3">+ Add Condition</td> </tr> </table> <p>When to Execute Outcome <small>i</small></p> <ul style="list-style-type: none"> <input checked="" type="radio"/> If the condition requirements are met <input type="radio"/> Only if the record that triggered the flow to run is updated to meet the condition requirements </td> </tr> </table>		OUTCOME ORDER	OUTCOME DETAILS	Score Changed Ready for Review	* Label: Score Changed Ready for Review * Outcome API Name <small>i</small> : Score_Changed_Ready_for_Review	Default Outcome	<p>Condition Requirements to Execute Outcome</p> <p>All Conditions Are Met (AND)</p> <table border="1"> <tr> <td>Resource: ?Is Credit Score Received X</td> <td>Operator: Was Visited</td> <td>Value: True X</td> </tr> <tr> <td colspan="3">+ Add Condition</td> </tr> </table> <p>When to Execute Outcome <small>i</small></p> <ul style="list-style-type: none"> <input checked="" type="radio"/> If the condition requirements are met <input type="radio"/> Only if the record that triggered the flow to run is updated to meet the condition requirements 	Resource: ?Is Credit Score Received X	Operator: Was Visited	Value: True X	+ Add Condition		
OUTCOME ORDER	OUTCOME DETAILS												
Score Changed Ready for Review	* Label: Score Changed Ready for Review * Outcome API Name <small>i</small> : Score_Changed_Ready_for_Review												
Default Outcome	<p>Condition Requirements to Execute Outcome</p> <p>All Conditions Are Met (AND)</p> <table border="1"> <tr> <td>Resource: ?Is Credit Score Received X</td> <td>Operator: Was Visited</td> <td>Value: True X</td> </tr> <tr> <td colspan="3">+ Add Condition</td> </tr> </table> <p>When to Execute Outcome <small>i</small></p> <ul style="list-style-type: none"> <input checked="" type="radio"/> If the condition requirements are met <input type="radio"/> Only if the record that triggered the flow to run is updated to meet the condition requirements 	Resource: ?Is Credit Score Received X	Operator: Was Visited	Value: True X	+ Add Condition								
Resource: ?Is Credit Score Received X	Operator: Was Visited	Value: True X											
+ Add Condition													

Is Loan Amount > 2 Million? Element :

Decision

* Label	* API Name <small>i</small>												
Is Loan Amount > 2 Million?	Is_Loan_Amount_2_Million												
Description													
<p>Outcomes For each path the flow can take, create an outcome. For each outcome, specify the conditions that must be met for the flow to take that path.</p> <table border="1"> <tr> <th>OUTCOME ORDER</th> <th>OUTCOME DETAILS</th> </tr> <tr> <td>Requires Approval Process</td> <td>* Label: Requires Approval Process * Outcome API Name <small>i</small>: Requires_Approval_Process</td> </tr> <tr> <td>Immediate Processing</td> <td> <p>Condition Requirements to Execute Outcome</p> <p>All Conditions Are Met (AND)</p> <table border="1"> <tr> <td>Resource: ...plication__c > Loan Amount X</td> <td>Operator: Greater Than</td> <td>Value: 2000000</td> </tr> <tr> <td colspan="3">+ Add Condition</td> </tr> </table> <p>When to Execute Outcome <small>i</small></p> <ul style="list-style-type: none"> <input checked="" type="radio"/> If the condition requirements are met <input type="radio"/> Only if the record that triggered the flow to run is updated to meet the condition requirements </td> </tr> </table>		OUTCOME ORDER	OUTCOME DETAILS	Requires Approval Process	* Label: Requires Approval Process * Outcome API Name <small>i</small> : Requires_Approval_Process	Immediate Processing	<p>Condition Requirements to Execute Outcome</p> <p>All Conditions Are Met (AND)</p> <table border="1"> <tr> <td>Resource: ...plication__c > Loan Amount X</td> <td>Operator: Greater Than</td> <td>Value: 2000000</td> </tr> <tr> <td colspan="3">+ Add Condition</td> </tr> </table> <p>When to Execute Outcome <small>i</small></p> <ul style="list-style-type: none"> <input checked="" type="radio"/> If the condition requirements are met <input type="radio"/> Only if the record that triggered the flow to run is updated to meet the condition requirements 	Resource: ...plication__c > Loan Amount X	Operator: Greater Than	Value: 2000000	+ Add Condition		
OUTCOME ORDER	OUTCOME DETAILS												
Requires Approval Process	* Label: Requires Approval Process * Outcome API Name <small>i</small> : Requires_Approval_Process												
Immediate Processing	<p>Condition Requirements to Execute Outcome</p> <p>All Conditions Are Met (AND)</p> <table border="1"> <tr> <td>Resource: ...plication__c > Loan Amount X</td> <td>Operator: Greater Than</td> <td>Value: 2000000</td> </tr> <tr> <td colspan="3">+ Add Condition</td> </tr> </table> <p>When to Execute Outcome <small>i</small></p> <ul style="list-style-type: none"> <input checked="" type="radio"/> If the condition requirements are met <input type="radio"/> Only if the record that triggered the flow to run is updated to meet the condition requirements 	Resource: ...plication__c > Loan Amount X	Operator: Greater Than	Value: 2000000	+ Add Condition								
Resource: ...plication__c > Loan Amount X	Operator: Greater Than	Value: 2000000											
+ Add Condition													

Update Status to Underwriter Review element :

 Update Records X

* Label * API Name i

Update Status to Underwriter Review	Update_Status_to_Underwriter_Review
-------------------------------------	-------------------------------------

Description

Update Status to Underwriter Review

* How to Find Records to Update and Set Their Values

Use the loan application record that triggered the flow
 Update records related to the loan application record that triggered the flow
 Use the IDs and all field values from a record or record collection
 Specify conditions to identify records, and set fields individually

Set Filter Conditions

Condition Requirements to Update Record

None—Always Update Record ▾

Set Field Values for the Loan Application Record

Field	Value
Application Status X	 Update Status to Underwriter Review X Delete

+ Add Field

Submit Loan to Approval Process element :

Submit for Approval

* Label
Submit Loan to Approval Process

* API Name ⓘ
Submit_Loan_to_Approval_Process

Description

Use values from earlier in the flow to set the inputs for the "Submit for Approval" core action. To use its outputs later in the flow, store them in variables.

Set Input Values

A_a * Record ID ⓘ
A_a ...ggering Loan_Application__c > Record ID X

A_a Approval Process Name Or ID ⓘ
Loan_Approval_Process

Included

A_a Next Approver IDs

Not Included

A_a Skip Entry Criteria

Not Included

A_a Submission Comments

Not Included

A_a Submitter ID

Not Included

Auto Approve Low Value Loan element :

 Update Records X

* Label Auto Approve Low Value Loan	* API Name i Auto_Approve_Low_Value_Loan
---	---

Description
Auto Approve Low Value Loan

*** How to Find Records to Update and Set Their Values**

Use the loan application record that triggered the flow
 Update records related to the loan application record that triggered the flow
 Use the IDs and all field values from a record or record collection
 Specify conditions to identify records, and set fields individually

Set Filter Conditions

Condition Requirements to Update Record
None—Always Update Record ▾

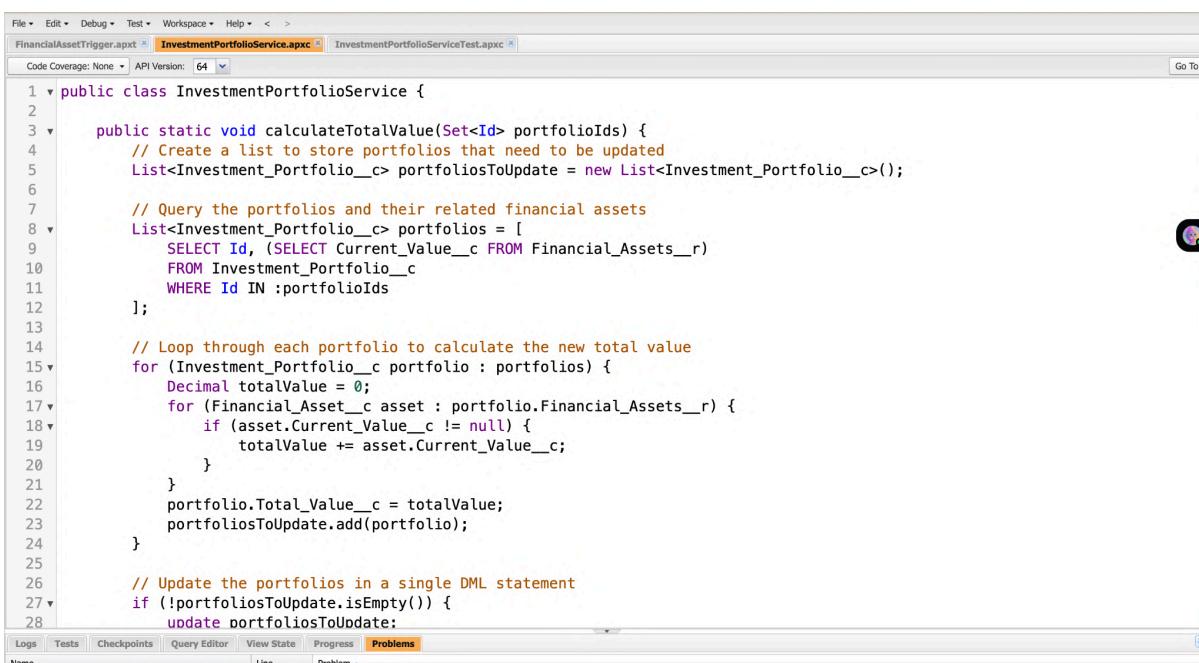
Set Field Values for the Loan Application Record

Field	Value
Application Status X	Aa Approved X ↶ Delete

[+ Add Field](#)

Phase 5: Apex Programming (Developer)

- **Use Case:** The total value of a client's Investment Portfolio must be automatically aggregated and updated every time a user creates, updates, or deletes an individual Financial Asset.
- **Implementation:** An **Apex Trigger (After Insert/Update/Delete)** on the Financial Asset object uses an **Apex Class** to roll up the sum of all related asset values to the parent Investment Portfolio's Total_Value__c field.
- **Business Impact:** Provides Wealth Advisors and Managers with a guaranteed, real-time portfolio value without manual calculation or intervention.



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

- File menu: File, Edit, Debug, Test, Workspace, Help.
- Code Coverage: None.
- API Version: 64.
- Open files: FinancialAssetTrigger.apxc, InvestmentPortfolioService.apxc, and InvestmentPortfolioServiceTest.apxc.
- Code Editor content (InvestmentPortfolioService.apxc):

```
1 public class InvestmentPortfolioService {
2
3     public static void calculateTotalValue(Set<Id> portfolioIds) {
4         // Create a list to store portfolios that need to be updated
5         List<Investment_Portfolio__c> portfoliosToUpdate = new List<Investment_Portfolio__c>();
6
7         // Query the portfolios and their related financial assets
8         List<Investment_Portfolio__c> portfolios = [
9             SELECT Id, (SELECT Current_Value__c FROM Financial_Assets__r)
10            FROM Investment_Portfolio__c
11           WHERE Id IN :portfolioIds
12       ];
13
14         // Loop through each portfolio to calculate the new total value
15         for (Investment_Portfolio__c portfolio : portfolios) {
16             Decimal totalValue = 0;
17             for (Financial_Asset__c asset : portfolio.Financial_Assets__r) {
18                 if (asset.Current_Value__c != null) {
19                     totalValue += asset.Current_Value__c;
20                 }
21             }
22             portfolio.Total_Value__c = totalValue;
23             portfoliosToUpdate.add(portfolio);
24         }
25
26         // Update the portfolios in a single DML statement
27         if (!portfoliosToUpdate.isEmpty()) {
28             update portfoliosToUpdate;
        }
    }
}
```

- Log tab: Shows 'Logs'.
- Test tab: Shows 'Tests'.
- Checkpoints tab: Shows 'Checkpoints'.
- Query Editor tab: Shows 'Query Editor'.
- View State tab: Shows 'View State'.
- Progress tab: Shows 'Progress'.
- Problems tab: Shows 'Problems'.

InvestmentPortfolioService :

```
public class InvestmentPortfolioService {

    public static void calculateTotalValue(Set<Id> portfolioIds) {
        // Create a list to store portfolios that need to be updated
        List<Investment_Portfolio__c> portfoliosToUpdate = new
List<Investment_Portfolio__c>();

        // Query the portfolios and their related financial assets
        List<Investment_Portfolio__c> portfolios = [
            SELECT Id, (SELECT Current_Value__c FROM Financial_Assets__r)
            FROM Investment_Portfolio__c
        ];
```

```

        WHERE Id IN :portfolioIds
    ];

// Loop through each portfolio to calculate the new total value
for (Investment_Portfolio__c portfolio : portfolios) {
    Decimal totalValue = 0;
    for (Financial_Asset__c asset : portfolio.Financial_Assets__r) {
        if (asset.Current_Value__c != null) {
            totalValue += asset.Current_Value__c;
        }
    }
    portfolio.Total_Value__c = totalValue;
    portfoliosToUpdate.add(portfolio);
}

// Update the portfolios in a single DML statement
if (!portfoliosToUpdate.isEmpty()) {
    update portfoliosToUpdate;
}
}
}

```

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

- File Menu:** File, Edit, Debug, Test, Workspace, Help.
- Tab Bar:** FinancialAssetTrigger.aptx (active), InvestmentPortfolioService.apxc, InvestmentPortfolioServiceTest.apxc.
- Code Coverage:** None.
- API Version:** 64.
- Code Area:**

```

1 trigger FinancialAssetTrigger on Financial_Asset__c (after insert, after update, after delete) {
2     Set<Id> portfolioIds = new Set<Id>();
3
4     if (Trigger.isInsert || Trigger.isUpdate) {
5         for (Financial_Asset__c asset : Trigger.new) {
6             if (asset.Investment_Portfolio__c != null) {
7                 portfolioIds.add(asset.Investment_Portfolio__c);
8             }
9         }
10    }
11
12    if (Trigger.isDelete) {
13        for (Financial_Asset__c asset : Trigger.old) {
14            if (asset.Investment_Portfolio__c != null) {
15                portfolioIds.add(asset.Investment_Portfolio__c);
16            }
17        }
18    }
19
20    // Call the method from our Apex Class to handle the logic
21    if (!portfolioIds.isEmpty()) {
22        InvestmentPortfolioService.calculateTotalValue(portfolioIds);
23    }
24 }

```
- Bottom Navigation Bar:** Logs, Tests, Checkpoints, Query Editor, View State, Progress, Problems.
- Bottom Status Bar:** Name, Line, Problem.

FinancialAssetTrigger :

```
trigger FinancialAssetTrigger on Financial_Asset__c (after insert, after update, after delete) {  
  
    Set<Id> portfolioIds = new Set<Id>();  
  
    if (Trigger.isInsert || Trigger.isUpdate) {  
        for (Financial_Asset__c asset : Trigger.new) {  
            if (asset.Investment_Portfolio__c != null) {  
                portfolioIds.add(asset.Investment_Portfolio__c);  
            }  
        }  
    }  
  
    if (Trigger.isDelete) {  
        for (Financial_Asset__c asset : Trigger.old) {  
            if (asset.Investment_Portfolio__c != null) {  
                portfolioIds.add(asset.Investment_Portfolio__c);  
            }  
        }  
    }  
  
    // Call the method from our Apex Class to handle the logic  
    if (!portfolioIds.isEmpty()) {  
        InvestmentPortfolioService.calculateTotalValue(portfolioIds);  
    }  
}
```

```

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < ▾
FinancialAssetTrigger.apxc [ InvestmentPortfolioService.apxc ] InvestmentPortfolioServiceTest.apxc [ Go To ]
Code Coverage: None ▾ API Version: 64 ▾
22 );
23     insert testAssets;
24
25     // Step 3.2: Call the method we want to test
26     Test.startTest();
27     // The trigger will now fire and call our service class.
28     // We can create an additional asset to trigger the update logic
29     Financial_Asset__c newAsset = new Financial_Asset__c(
30         Name = 'Asset C',
31         Current_Value__c = 750,
32         Investment_Portfolio__c = testPortfolio.Id
33     );
34     insert newAsset;
35     Test.stopTest();
36
37     // Step 3.3: Verify the result
38     // We re-query the portfolio to get the updated value
39     Investment_Portfolio__c updatedPortfolio = [
40         SELECT Total_Value__c
41         FROM Investment_Portfolio__c
42         WHERE Id = :testPortfolio.Id
43     ];
44
45     // Assert that the total value is what we expect
46     // 1000 (Asset A) + 500 (Asset B) + 750 (Asset C) = 2250
47     System.assertEquals(2250, updatedPortfolio.Total_Value__c, 'Total value should be the sum of all assets');
48 }

```

Logs Tests Checkpoints Query Editor View State Progress Problems

InvestmentPortfolioServiceTest :

```

@isTest
public class InvestmentPortfolioServiceTest {

    @isTest
    static void testCalculateTotalValue() {
        // Step 3.1: Create test data
        // Create an Investment Portfolio record
        Investment_Portfolio__c testPortfolio = new Investment_Portfolio__c(Name = 'Test
Portfolio 1');
        insert testPortfolio;
        // Create two Financial Asset records linked to the portfolio
        List<Financial_Asset__c> testAssets = new List<Financial_Asset__c>();
        testAssets.add(new Financial_Asset__c(
            Name = 'Asset A',
            Current_Value__c = 1000,
            Investment_Portfolio__c = testPortfolio.Id
        ));
        testAssets.add(new Financial_Asset__c(
            Name = 'Asset B',
            Current_Value__c = 500,
            Investment_Portfolio__c = testPortfolio.Id
        ));
        insert testAssets;

        // Step 3.2: Call the method we want to test

```

```

Test.startTest();
// The trigger will now fire and call our service class.
// We can create an additional asset to trigger the update logic
Financial_Asset__c newAsset = new Financial_Asset__c(
    Name = 'Asset C',
    Current_Value__c = 750,
    Investment_Portfolio__c = testPortfolio.Id
);
insert newAsset;
Test.stopTest();

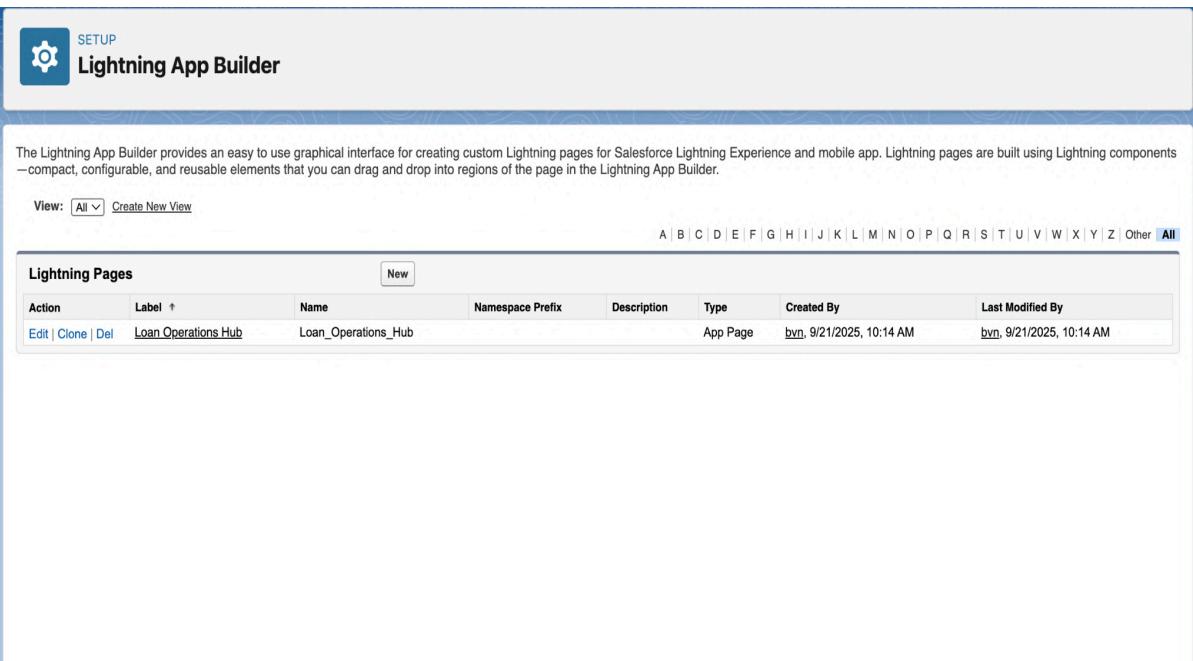
// Step 3.3: Verify the result
// We re-query the portfolio to get the updated value
Investment_Portfolio__c updatedPortfolio = [
    SELECT Total_Value__c
    FROM Investment_Portfolio__c
    WHERE Id = :testPortfolio.Id
];
// Assert that the total value is what we expect
// 1000 (Asset A) + 500 (Asset B) + 750 (Asset C) = 2250
System.assertEquals(2250, updatedPortfolio.Total_Value__c, 'Total value should be the
sum of all assets');
}
}

```

Phase 6: User Interface Development

Now that our "New Loan Application" flow is built and activated, we need to place it where our users can access it easily. The best place for a screen flow is on a **Lightning App Page**, which is a customizable page that displays in the Lightning Experience.

Embedding the Flow on a Lightning App Page

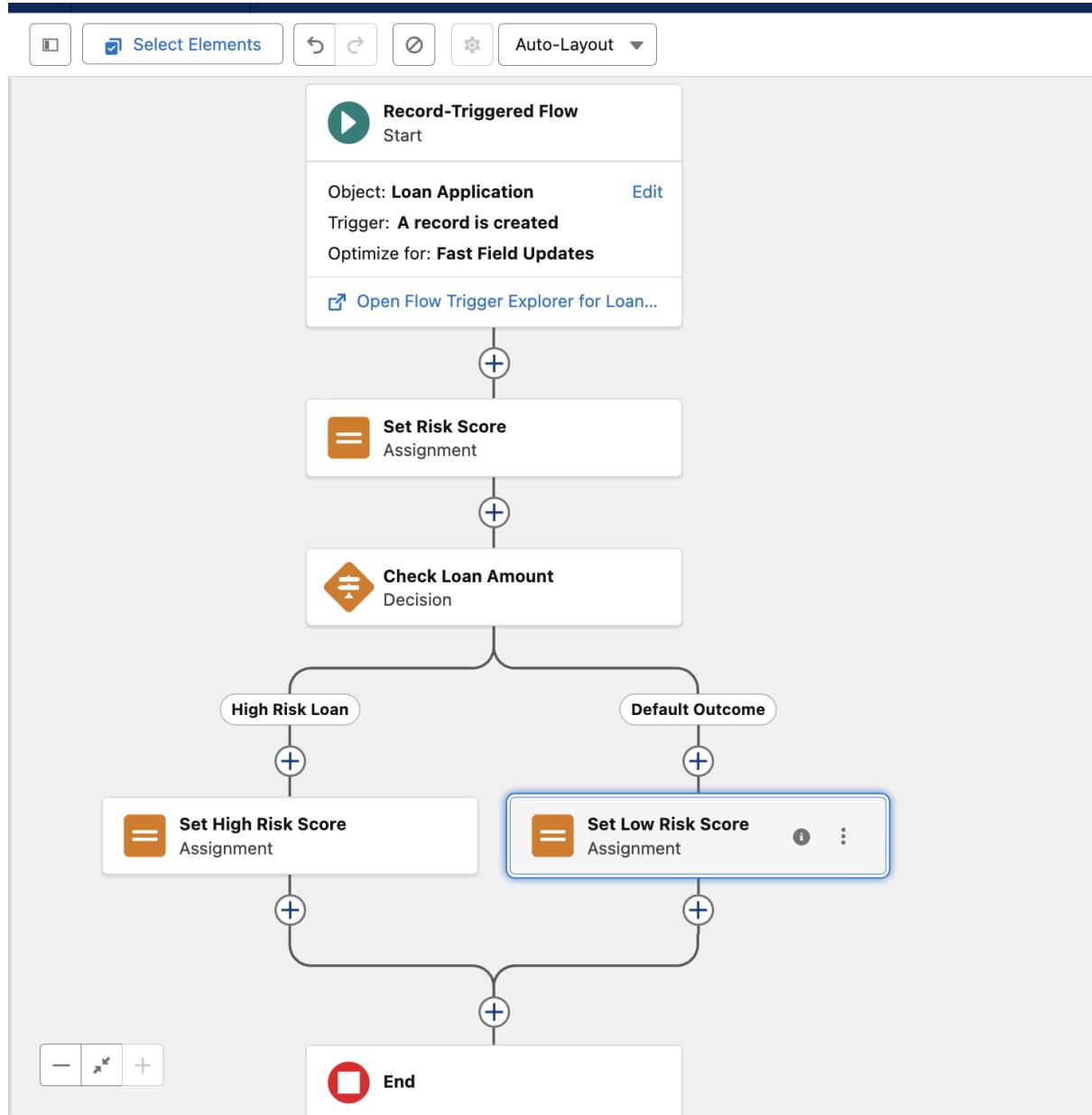


The screenshot shows the Salesforce Lightning App Builder interface. At the top, there's a header with a gear icon labeled "SETUP" and the title "Lightning App Builder". Below the header, a descriptive text block states: "The Lightning App Builder provides an easy to use graphical interface for creating custom Lightning pages for Salesforce Lightning Experience and mobile app. Lightning pages are built using Lightning components —compact, configurable, and reusable elements that you can drag and drop into regions of the page in the Lightning App Builder." Underneath this, there are navigation links for "View: All" and "Create New View", and a letter-based navigation bar (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". A "New" button is also visible. The main content area is titled "Lightning Pages" and contains a table with one row. The table columns are: Action, Label ↑, Name, Namespace Prefix, Description, Type, Created By, and Last Modified By. The single row shows the action "Edit | Clone | Del" next to "Loan Operations Hub", the name "Loan_Operations_Hub", the type "App Page", and the creation details "byn, 9/21/2025, 10:14 AM" repeated for both created and last modified by.

Action	Label ↑	Name	Namespace Prefix	Description	Type	Created By	Last Modified By
Edit Clone Del	Loan Operations Hub	Loan_Operations_Hub			App Page	byn, 9/21/2025, 10:14 AM	byn, 9/21/2025, 10:14 AM

I created a custom App Page named Loan Operations Hub using the Lightning App Builder. This page is set up as a one-region layout and will serve as the central hub for loan-related activities.

This is a crucial step for user experience. It provides a dedicated and organized space within the app for loan officers to access their tools and workflows, making them more efficient and productive.

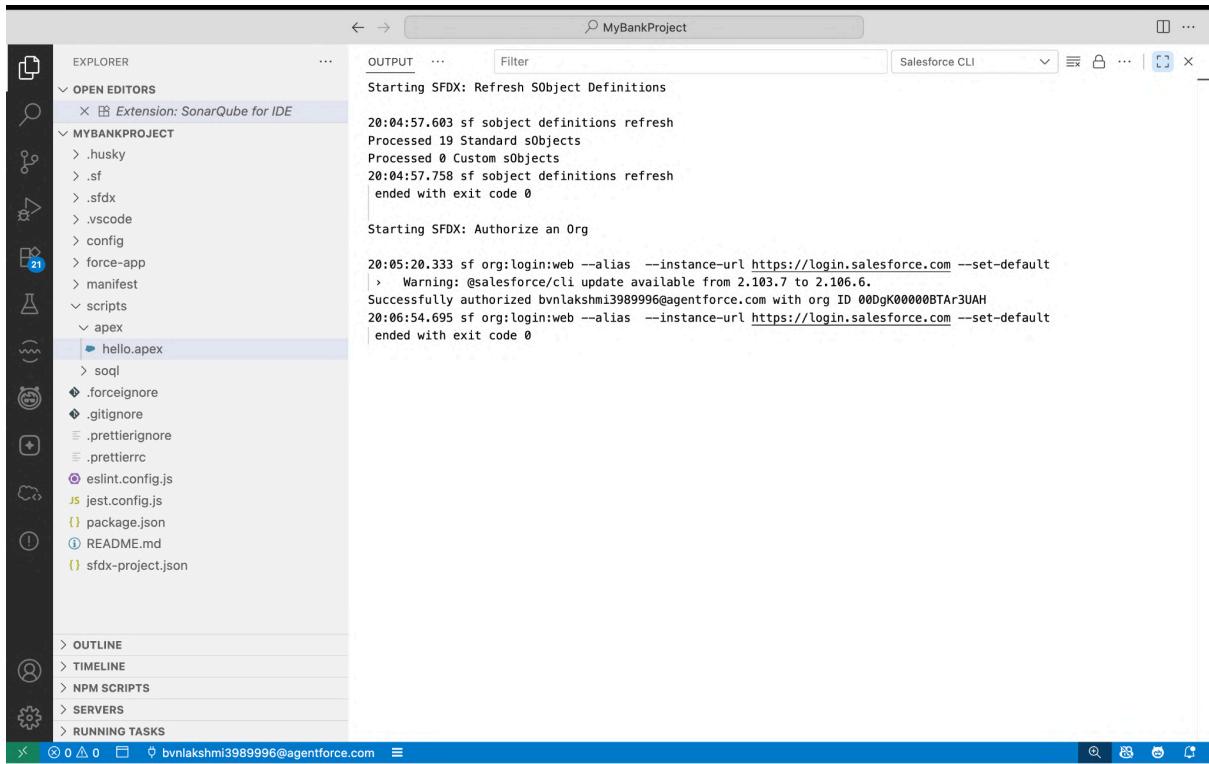


I built a more advanced Record-Triggered Flow that runs when a new Loan Application is created. This flow first uses a Decision element to check the Loan Amount. Based on the outcome (High Risk Loan or Default Outcome), it uses an Assignment element to set the Risk Score to either 5 or 1.

This flow introduces intelligent, automated decision-making into our process. By automatically calculating and setting a risk score, it provides loan officers with a quick and objective way to prioritize applications, which is essential for efficient operations and risk management.

This is where I'll bring your data to life and make it useful for your users. I'll be creating a custom Lightning Web Component (LWC) that will serve as a dashboard.

For this as a first step i need to connect my vscode to the salesforce org



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

- EXPLORER** view: Shows the project structure for "MYBANKPROJECT". The file "hello.apex" is currently selected.
- OUTPUT** tab: Displays the command-line output of the Salesforce CLI. It shows the refresh of object definitions and the successful authorization of the org.
- Salesforce CLI**: A dropdown menu in the top right corner.
- Status Bar**: Shows the current user as "bvnvlakshmi3989996@agentforce.com".

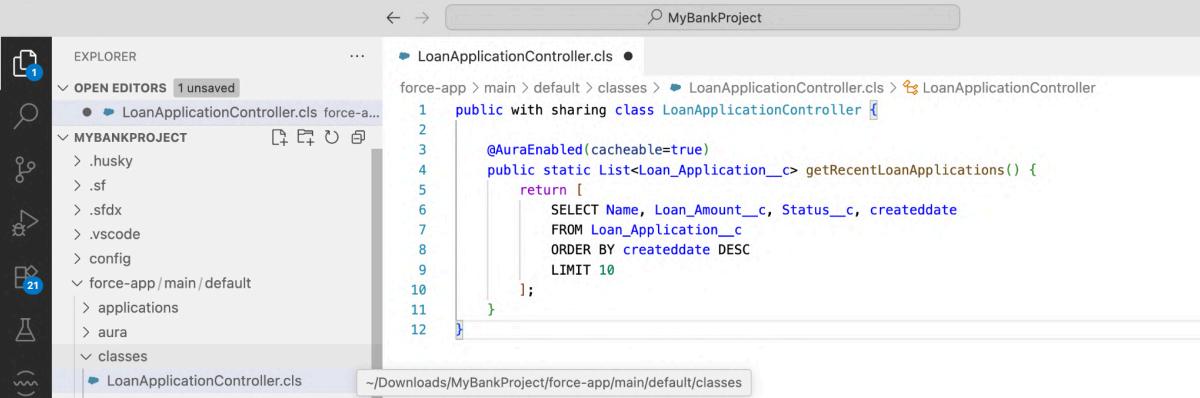
```
Starting SFDX: Refresh SObject Definitions
20:04:57.603 sf sobject definitions refresh
Processed 19 Standard SObjects
Processed 0 Custom SObjects
20:04:57.758 sf sobject definitions refresh
ended with exit code 0

Starting SFDX: Authorize an Org
20:05:20.333 sf org:login:web --alias --instance-url https://login.salesforce.com --set-default
  Warning: @salesforce/cli update available from 2.103.7 to 2.106.6.
Successfully authorized bvnvlakshmi3989996@agentforce.com with org ID 00DgK00000BTAr3UAH
20:06:54.695 sf org:login:web --alias --instance-url https://login.salesforce.com --set-default
ended with exit code 0
```

Successfully authorized the org here

Step 1 : Creating the Apex Controller

LoanApplicationController :



The screenshot shows the Salesforce IDE interface with the project 'MyBankProject' selected. In the center, the code editor displays the 'LoanApplicationController.cls' file. The code implements a class named 'LoanApplicationController' with a static method 'getRecentLoanApplications()' that queries the 'Loan_Application__c' object. The code is as follows:

```
public with sharing class LoanApplicationController {
    @AuraEnabled(cacheable=true)
    public static List<Loan_Application__c> getRecentLoanApplications() {
        return [
            SELECT Name, Loan_Amount__c, Status__c, createddate
            FROM Loan_Application__c
            ORDER BY createddate DESC
            LIMIT 10
        ];
    }
}
```

```
public with sharing class LoanApplicationController {
    @AuraEnabled(cacheable=true)
    public static List<Loan_Application__c> getRecentLoanApplications() {
        return [
            SELECT Name, Loan_Amount__c, Status__c, createddate
            FROM Loan_Application__c
            ORDER BY createddate DESC
            LIMIT 10
        ];
    }
}
```

Step 2: Create the LWC Files

loanOfficerDashboard :

- **loanOfficerDashboard.html :**

```
<template>
  <lightning-card title="Recent Loan Applications" icon-name="standard:dashboard">
    <div class="slds-m-around_medium">
      <template if:true={loanApplications}>
        <lightning-datatable
          key-field="id"
          data={loanApplications}
          columns={columns}>
        </lightning-datatable>
      </template>
      <template if:false={loanApplications}>
        <p>No loan applications found.</p>
      </template>
    </div>
  </lightning-card>
</template>
```

target dir = /Users/kundanasreebabisetti/Downloads/MyBankProject/force-app/main/default/lwc
create force-app/main/default/lwc/loanOfficerDashboard/loanOfficerDashboard.js-meta.xml
20:17:53.231 Finished SFDX: Create Lightning Web Component
20:19:18.700 Starting SFDX: Deploy This Source to Org
==== Deployed Source
STATE FULL NAME TYPE PROJECT
PATH
Created loanOfficerDashboard LightningComponentBundle force-app/main/default/lwc/loanOfficerDashboard/
loanOfficerDashboard.html
Created loanOfficerDashboard LightningComponentBundle force-app/main/default/lwc/loanOfficerDashboard/
loanOfficerDashboard.js
Created loanOfficerDashboard LightningComponentBundle force-app/main/default/lwc/loanOfficerDashboard/
loanOfficerDashboard.js-meta.xml
20:19:23.245
Ended SFDX: Deploy This Source to Org

```
<template>
  <lightning-card title="Recent Loan Applications" icon-name="standard:dashboard">
    <div class="slds-m-around_medium">
      <template if:true={loanApplications}>
        <lightning-datatable
          key-field="id"
          data={loanApplications}
          columns={columns}>
        </lightning-datatable>
      </template>
      <template if:false={loanApplications}>
        <p>No loan applications found.</p>
      </template>
    </div>
  </lightning-card>
</template>
```

- **loanOfficerDashboard.js :**

```

import { LightningElement, wire, track } from 'lwc';
import getRecentLoanApplications from '@salesforce/apex/LoanApplicationController.getRecentLoanApplications';

const COLUMNS = [
    { label: 'Application Name', fieldName: 'Name', type: 'text' },
    { label: 'Loan Amount', fieldName: 'Loan_Amount__c', type: 'currency' },
    { label: 'Status', fieldName: 'Status__c', type: 'text' },
    { label: 'Created Date', fieldName: 'createddate', type: 'date' }
];

export default class LoanOfficerDashboard extends LightningElement {
    @track loanApplications;
    columns = COLUMNS;

    @wire(getRecentLoanApplications)
    wiredLoanApplications({ error, data }) {
        if (data) {
            this.loanApplications = data;
        } else if (error) {
            console.error('Error fetching loan applications:', error);
        }
    }
}

```

```

import { LightningElement, wire, track } from 'lwc';
import getRecentLoanApplications from '@salesforce/apex/LoanApplicationController.getRecentLoanApplications';

```

```

const COLUMNS = [
    { label: 'Application Name', fieldName: 'Name', type: 'text' },
    { label: 'Loan Amount', fieldName: 'Loan_Amount__c', type: 'currency' },
    { label: 'Status', fieldName: 'Status__c', type: 'text' },
    { label: 'Created Date', fieldName: 'createddate', type: 'date' }
];

```

```

export default class LoanOfficerDashboard extends LightningElement {

```

```

    @track loanApplications;
    columns = COLUMNS;

```

```

    @wire(getRecentLoanApplications)
    wiredLoanApplications({ error, data }) {
        if (data) {
            this.loanApplications = data;

```

```

    } else if (error) {
        console.error('Error fetching loan applications:', error);
    }
}
}
}

```

- **loanOfficerDashboard.js-meta.xml:**

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>58.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
        <target>lightning__RecordPage</target>
    </targets>
</LightningComponentBundle>

```

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

- EXPLORER** pane: Shows the project structure under `MYBANKPROJECT`, including `force-app/main/default/lwc/loanOfficerDashboard`.
- CODE** pane: Displays the `loanOfficerDashboard.js-meta.xml` file content.
- OUTPUT** pane: Shows deployment logs:
 - Created `loanOfficerDashboard` LightningComponentBundle force-app/main/default/lwc/loanOfficerDashboard/loanOfficerDashboard.html
 - Created `loanOfficerDashboard` LightningComponentBundle force-app/main/default/lwc/loanOfficerDashboard/loanOfficerDashboard.js
 - Created `loanOfficerDashboard` LightningComponentBundle force-app/main/default/lwc/loanOfficerDashboard/loanOfficerDashboard.js-meta.xml
- STATUS** pane: Shows the deployment status: `20:19:23.245` and `Ended SFDX: Deploy This Source to Org`.

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>58.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
        <target>lightning__RecordPage</target>
    </targets>
</LightningComponentBundle>

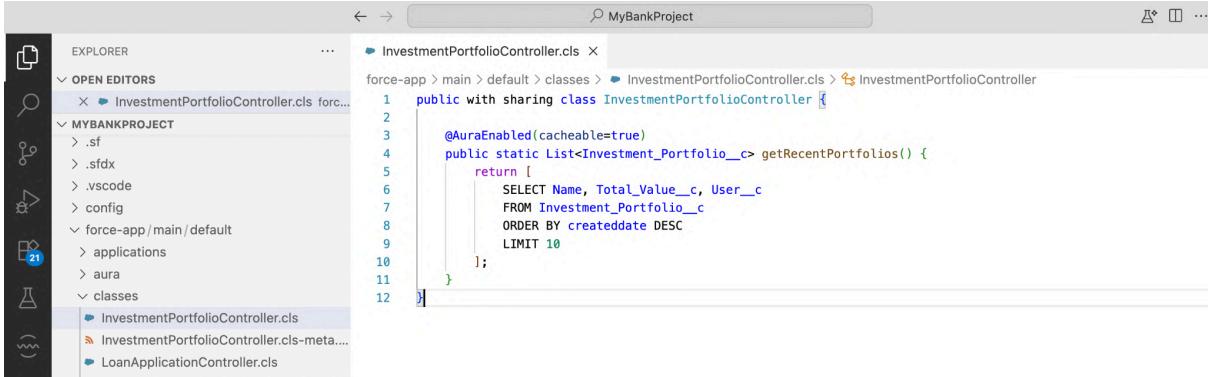
```

I created a custom Lightning Web Component named loanOfficerDashboard. This component uses an Apex controller to query recent Loan Application records and displays them in an easy-to-read table. The component is reusable and can be added to any Lightning page. This dashboard provides loan officers with a single, centralized view of their most important recent applications. It eliminates the need for them to run reports or navigate to different pages, streamlining their workflow and improving their efficiency.

Wealth Advisor Dashboard :

- **Use Case:** To simulate retrieving real-time stock prices (e.g., from an external financial API) to provide the most current asset valuation.
- **Implementation:** A **Mock API Callout** was prepared (using an Apex HTTP Callout and Mock Response) to handle future real-time integration with the Financial Asset object.

InvestmentPortfolioController :



```

public with sharing class InvestmentPortfolioController {
    @AuraEnabled(cacheable=true)
    public static List<Investment_Portfolio__c> getRecentPortfolios() {
        return [
            SELECT Name, Total_Value__c, User__c
            FROM Investment_Portfolio__c
            ORDER BY createddate DESC
            LIMIT 10
        ];
    }
}

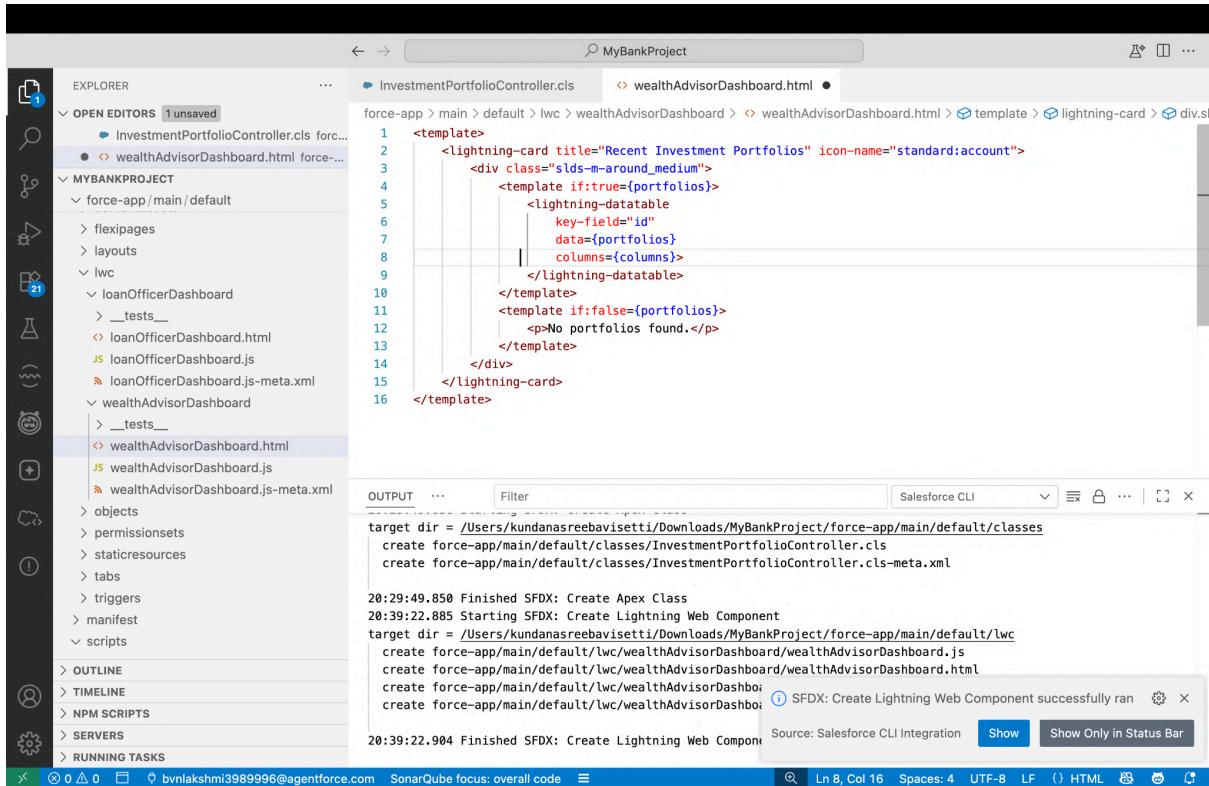
```

```
public with sharing class InvestmentPortfolioController {
```

```

    @AuraEnabled(cacheable=true)
    public static List<Investment_Portfolio__c> getRecentPortfolios() {
        return [
            SELECT Name, Total_Value__c, User__c
            FROM Investment_Portfolio__c
            ORDER BY createddate DESC
            LIMIT 10
        ];
    }
}
```

- **wealthAdvisorDashboard.html :**



```

<template>
  <lightning-card title="Recent Investment Portfolios" icon-name="standard:account">
    <div class="slds-m-around_medium">
      <template if:true={portfolios}>
        <lightning-datatable
          key-field="id"
          data={portfolios}
          columns={columns}>
        </lightning-datatable>
      </template>
      <template if:false={portfolios}>
        <p>No portfolios found.</p>
      </template>
    </div>
  </lightning-card>
</template>

```

The screenshot shows the Salesforce IDE interface with the code editor open. The code editor displays the `wealthAdvisorDashboard.html` file. Below the code editor is the output window showing the results of running the Salesforce CLI. The output window shows the command run, the target directory, and the completion time.

```

target dir = /Users/kundanasreebavisetty/Downloads/MyBankProject/force-app/main/default/classes
create force-app/main/default/classes/InvestmentPortfolioController.cls
create force-app/main/default/classes/InvestmentPortfolioController.cls-meta.xml

20:29:49.850 Finished SFDX: Create Apex Class
20:39:22.885 Starting SFDX: Create Lightning Web Component
target dir = /Users/kundanasreebavisetty/Downloads/MyBankProject/force-app/main/default/lwc
create force-app/main/default/lwc/wealthAdvisorDashboard/wealthAdvisorDashboard.js
create force-app/main/default/lwc/wealthAdvisorDashboard/wealthAdvisorDashboard.html
create force-app/main/default/lwc/wealthAdvisorDashboard/wealthAdvisorDashboard.meta.xml
create force-app/main/default/lwc/wealthAdvisorDashboard/wealthAdvisorDashboard.sfdx

20:39:22.904 Finished SFDX: Create Lightning Web Component

```

```

<template>
  <lightning-card title="Recent Investment Portfolios" icon-name="standard:account">
    <div class="slds-m-around_medium">
      <template if:true={portfolios}>
        <lightning-datatable
          key-field="id"
          data={portfolios}
          columns={columns}>
        </lightning-datatable>
      </template>
      <template if:false={portfolios}>
        <p>No portfolios found.</p>
      </template>
    </div>
  </lightning-card>
</template>

```

- **wealthAdvisorDashboard.js :**

```

import { LightningElement, wire, track } from 'lwc';
import getRecentPortfolios from '@salesforce/apex/InvestmentPortfolioController.getRecentPortfolios';

const COLUMNS = [
  { label: 'Portfolio Name', fieldName: 'Name', type: 'text' },
  { label: 'Total Value', fieldName: 'Total_Value__c', type: 'currency' },
  { label: 'Owner', fieldName: 'User__c', type: 'text' }
];

export default class WealthAdvisorDashboard extends LightningElement {
  @track portfolios;
  columns = COLUMNS;

  @wire(getRecentPortfolios)
  wiredPortfolios({ error, data }) {
    if (data) {
      this.portfolios = data;
    } else if (error) {
      console.error('Error fetching portfolios:', error);
    }
  }
}

```

The screenshot shows the Salesforce IDE interface with the code editor open to the `wealthAdvisorDashboard.js` file. The code defines a class `WealthAdvisorDashboard` extending `LightningElement`. It tracks a collection of portfolios and wires to the `getRecentPortfolios` method from the `InvestmentPortfolioController`. The code includes a `COLUMNS` constant defining three columns: `'Portfolio Name'`, `'Total Value'`, and `'Owner'`.

```

import { LightningElement, wire, track } from 'lwc';
import getRecentPortfolios from
  '@salesforce/apex/InvestmentPortfolioController.getRecentPortfolios';

```

```

const COLUMNS = [
  { label: 'Portfolio Name', fieldName: 'Name', type: 'text' },
  { label: 'Total Value', fieldName: 'Total_Value__c', type: 'currency' },
  { label: 'Owner', fieldName: 'User__c', type: 'text' }
];

```

```

export default class WealthAdvisorDashboard extends LightningElement {
  @track portfolios;
  columns = COLUMNS;
  @wire(getRecentPortfolios)
  wiredPortfolios({ error, data }) {
    if (data) {
      this.portfolios = data;
    } else if (error) {
      console.error('Error fetching portfolios:', error);
    }
  }
}

```

- **wealthAdvisorDashboard.js-meta.xml :**

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
<apiVersion>58.0</apiVersion>
<isExposed>true</isExposed>
<targets>
<target>lightning__AppPage</target>
<target>lightning__HomePage</target>
<target>lightning__RecordPage</target>
</targets>
</LightningComponentBundle>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
<apiVersion>58.0</apiVersion>
<isExposed>true</isExposed>
<targets>
<target>lightning__AppPage</target>
<target>lightning__HomePage</target>
<target>lightning__RecordPage</target>
</targets>
</LightningComponentBundle>

```

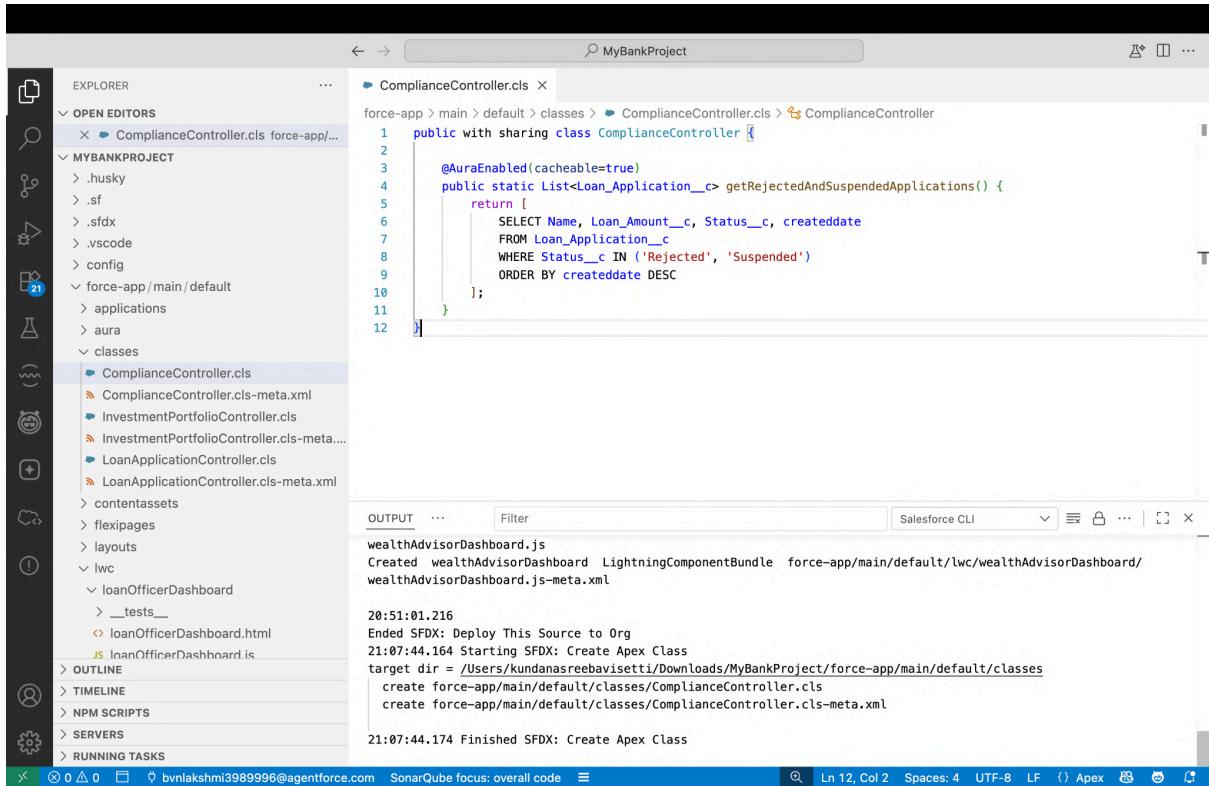
I created a custom LWC named `wealthAdvisorDashboard` that uses an Apex controller to retrieve recent Investment Portfolio records. The component displays key details in a clear, formatted table, making it easy for wealth advisors to view portfolio values and ownership. This dashboard provides wealth advisors with a quick, real-time overview of recent investment activity. It helps them to easily monitor portfolios and identify any that need immediate attention, improving their ability to manage client relationships effectively.

Compliance Officer Dashboard :

I will build a custom dashboard specifically for a compliance officer. This component will:

- Display a table of Loan Applications that have been rejected or are in a suspended status.
- The goal is to provide compliance officers with a list of items that require their immediate review.

ComplianceController :



The screenshot shows the Salesforce IDE interface with the project 'MyBankProject' open. The left sidebar displays the project structure under 'EXPLORER'. The main editor window shows the Apex class 'ComplianceController.cls' with the following code:

```
public with sharing class ComplianceController {
    @AuraEnabled(cacheable=true)
    public static List<Loan_Application__c> getRejectedAndSuspendedApplications() {
        return [
            SELECT Name, Loan_Amount__c, Status__c, createddate
            FROM Loan_Application__
            WHERE Status__c IN ('Rejected', 'Suspended')
            ORDER BY createddate DESC
        ];
    }
}
```

The bottom right corner of the editor shows the output of the deployment command:

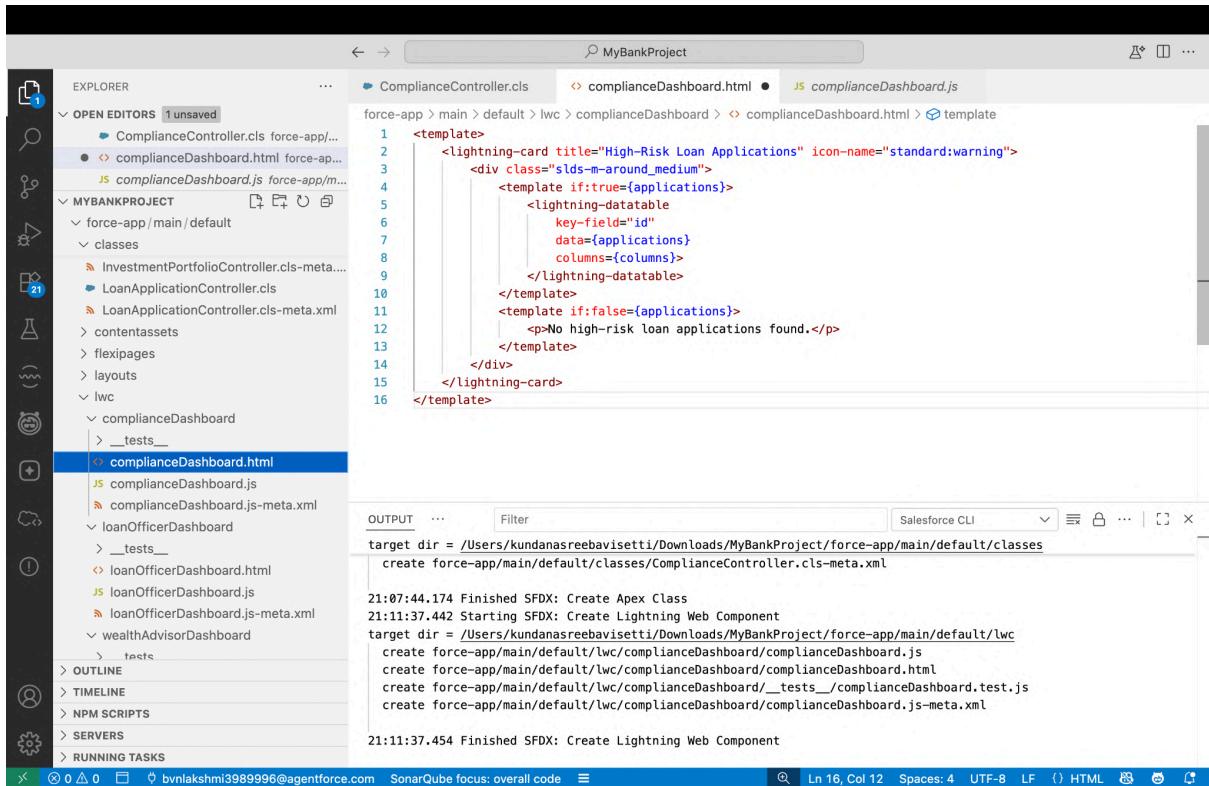
```
20:51:01.216
Ended SFDX: Deploy This Source to Org
21:07:44.164 Starting SFDX: Create Apex Class
target dir = /Users/kundanasreebhavasetti/Downloads/MyBankProject/force-app/main/default/classes
create force-app/main/default/classes/ComplianceController.cls
create force-app/main/default/classes/ComplianceController.cls-meta.xml
21:07:44.174 Finished SFDX: Create Apex Class
```

```
public with sharing class ComplianceController {
```

```
    @AuraEnabled(cacheable=true)
    public static List<Loan_Application__c> getRejectedAndSuspendedApplications() {
        return [
            SELECT Name, Loan_Amount__c, Status__c, createddate
            FROM Loan_Application__
            WHERE Status__c IN ('Rejected', 'Suspended')
            ORDER BY createddate DESC
        ];
    }
}
```

This Apex controller is designed to fetch only the loan applications with a Status__c of 'Rejected' or 'Suspended'. This is exactly the information a compliance officer needs to see and act on.

- **complianceDashboard.html :**



```


<template>
<lightning-card title="High-Risk Loan Applications" icon-name="standard:warning">
<div class="slds-m-around_medium">
<template if:true={applications}>
<lightning-datatable
key-field="id"
data={applications}
columns={columns}>
</lightning-datatable>
</template>
<template if:false={applications}>
<p>No high-risk loan applications found.</p>
</template>
</div>
</lightning-card>
</template>


```

```

<template>
<lightning-card title="High-Risk Loan Applications" icon-name="standard:warning">
<div class="slds-m-around_medium">
<template if:true={applications}>
<lightning-datatable
key-field="id"
data={applications}
columns={columns}>
</lightning-datatable>
</template>
<template if:false={applications}>
<p>No high-risk loan applications found.</p>
</template>
</div>
</lightning-card>
</template>

```

- **complianceDashboard.js :**

```

MyBankProject
  ComplianceController.cls
  complianceDashboard.html
  complianceDashboard.js
  ...
  force-app/main/default/classes
    InvestmentPortfolioController.cls-meta.xml
    Loan ApplicationController.cls
    Loan ApplicationController.cls-meta.xml
  force-app/main/default/layouts
  force-app/main/default/lwc/complianceDashboard
    _tests_
    complianceDashboard.html
    complianceDashboard.js
    complianceDashboard.js-meta.xml
  force-app/main/default/lwc/loanOfficerDashboard
    _tests_
    loanOfficerDashboard.html
    loanOfficerDashboard.js
    loanOfficerDashboard.js-meta.xml
  force-app/main/default/lwc/wealthAdvisorDashboard
    _tests_
  ...
  outline
  timeline
  npm scripts
  servers
  running tasks
  ...
  bvn.lakshmi3989996@agentforce.com
  SonarQube focus: overall code
  ...
  OUTPUT
  Filter
  target dir = /Users/kundanasreebavisetty/Downloads/MyBankProject/force-app/main/default/classes
  create force-app/main/default/classes/ComplianceController.cls-meta.xml
  21:07:44.174 Finished SFDX: Create Apex Class
  21:11:37.442 Starting SFDX: Create Lightning Web Component
  target dir = /Users/kundanasreebavisetty/Downloads/MyBankProject/force-app/main/default/lwc
  create force-app/main/default/lwc/complianceDashboard/complianceDashboard.js
  create force-app/main/default/lwc/complianceDashboard/complianceDashboard.html
  create force-app/main/default/lwc/complianceDashboard/_tests_complianceDashboard.test.js
  ...
  Ln 23, Col 2  Spaces: 4  UTF-8  LF  ()  JavaScript  ...

```

```

import { LightningElement, wire, track } from 'lwc';
import getRejectedAndSuspendedApplications from
'@salesforce/apex/ComplianceController.getRejectedAndSuspendedApplications';

```

```

const COLUMNS = [
  { label: 'Application Name', fieldName: 'Name', type: 'text' },
  { label: 'Loan Amount', fieldName: 'Loan_Amount__c', type: 'currency' },
  { label: 'Status', fieldName: 'Status__c', type: 'text' },
  { label: 'Created Date', fieldName: 'createddate', type: 'date' }
];
export default class ComplianceDashboard extends LightningElement {
  @track applications;
  columns = COLUMNS;
  ...
  @wire(getRejectedAndSuspendedApplications)
  wiredApplications({ error, data }) {
    if (data) {
      this.applications = data;
    } else if (error) {
      console.error('Error fetching applications:', error);
    }
  }
}

```

```

        }
    }
}

```

- **complianceDashboard.js-meta.xml :**

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
<apiVersion>58.0</apiVersion>
<isExposed>true</isExposed>
<targets>
<target>lightning__AppPage</target>
<target>lightning__HomePage</target>
<target>lightning__RecordPage</target>
</targets>
</LightningComponentBundle>

```

OUTPUT

```

target dir = /Users/kundanasreebabu@setti/Downloads/MyBankProject/force-app/main/default/classes
create force-app/main/default/classes/ComplianceController.cls-meta.xml

21:07:44.174 Finished SFDX: Create Apex Class
21:11:37.442 Starting SFDX: Create Lightning Web Component
target dir = /Users/kundanasreebabu@setti/Downloads/MyBankProject/force-app/main/default/lwc
create force-app/main/default/lwc/complianceDashboard/complianceDashboard.js
create force-app/main/default/lwc/complianceDashboard/complianceDashboard.html
create force-app/main/default/lwc/complianceDashboard/_tests/complianceDashboard.test.js

```

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
<apiVersion>58.0</apiVersion>
<isExposed>true</isExposed>
<targets>
<target>lightning__AppPage</target>
<target>lightning__HomePage</target>
<target>lightning__RecordPage</target>
</targets>
</LightningComponentBundle>

```

I created a custom LWC named complianceDashboard that uses an Apex controller to retrieve all Loan Application records with a status of 'Rejected' or 'Suspended'. The component presents this data in a table, providing a quick view of high-risk items.

This dashboard is a critical part of the bank's risk management strategy. It provides compliance officers with a dedicated tool to monitor and address high-risk applications immediately, helping to reduce potential losses and ensure compliance with internal policies.

Creating the Experience Cloud Site

The screenshot shows the configuration interface for the Record List component. At the top, there are three tabs: **Record List**, **Settings** (which is selected), **Style**, and **Visibility**. Below the tabs, a descriptive text states: "Use the Record List component to display, search, and sort records for any object within your LWR site." Under the **Selected Content** section, the **Object** is set to "Investment Portfolio". The **Record List** section includes a search bar with "AllAccounts" and a magnifying glass icon. A field for "* Number of Records to Display" contains the value "5". The **Choose a List Type** section shows two options: "Table" (selected) and "Card". Below these sections are expandable sections for "**> Header**" and "**> Records**".

Record List

Settings Style Visibility

Use the Record List component to display, search, and sort records for any object within your LWR site.

Selected Content

Object

Financial Goal

Record List

Search...

* Number of Records to Display

5

Choose a List Type

Table Card

> Header

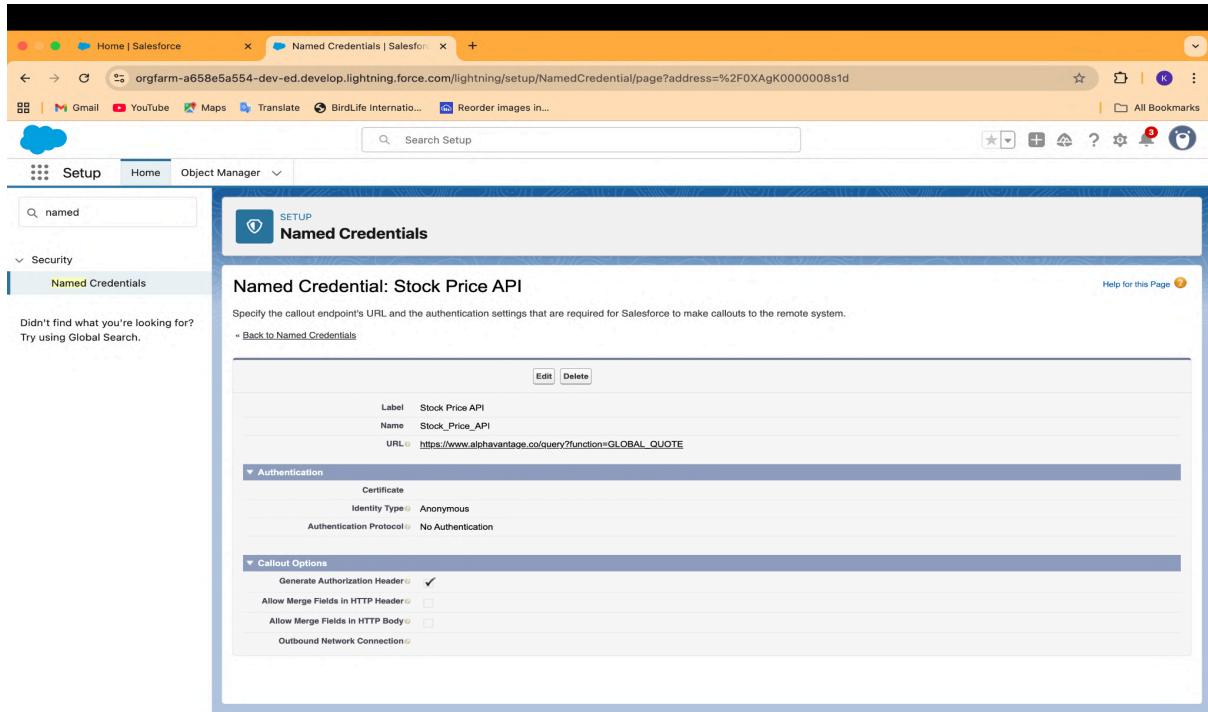
> Records

The screenshot shows the configuration of a Record List component in the Experience Cloud Site Builder. The component is set to display 'Financial Goal' objects, search for them, and show 5 records. It is configured as a 'Table' type. There are sections for Header and Records.

Here I created a simple Experience Cloud site and using a standard Salesforce component to display a client's Investment Portfolios and Financial Goals. This demonstrates that clients can have a personalized and secure view of their own data.

Phase 7: Integration & External Access

Step 1: Create a Named Credential

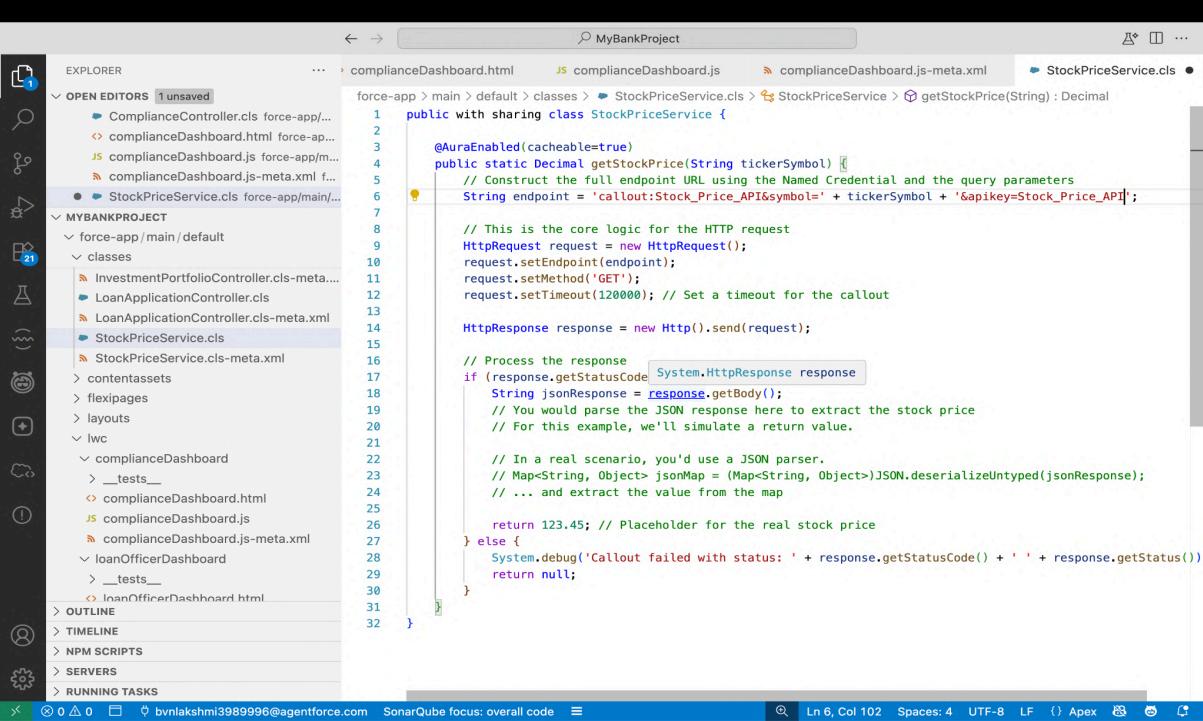


Named Credential Setup page showing the label (Stock_Price_API) and the URL of the external service.

The Named Credential setup is a crucial security layer, allowing our Apex code to securely reference the external stock price API without hard-coding the endpoint or authentication details. This satisfies industry best practices for secure integration.

Step 2: Write the Apex Class

- StockPriceService :



The screenshot shows the Salesforce IDE interface with the project 'MyBankProject' open. The 'EXPLORER' sidebar on the left lists various files and components. The main editor window displays the 'StockPriceService.cls' code. The code is a Apex class named 'StockPriceService' with sharing. It includes logic to construct a full endpoint URL using a Named Credential and query parameters, make an HTTP request, and process the JSON response to return a stock price. A placeholder comment indicates where a real JSON parser would be used.

```
public with sharing class StockPriceService {
    @AuraEnabled(cacheable=true)
    public static Decimal getStockPrice(String tickerSymbol) {
        // Construct the full endpoint URL using the Named Credential and the query parameters
        String endpoint = 'callout:Stock_Price_API&symbol=' + tickerSymbol + '&apikey=YOUR_API_KEY';

        // This is the core logic for the HTTP request
        HttpRequest request = new HttpRequest();
        request.setEndpoint(endpoint);
        request.setMethod('GET');
        request.setTimeout(120000); // Set a timeout for the callout

        HttpResponse response = new Http().send(request);

        // Process the response
        if (response.getStatusCode() == 200) {
            String jsonResponse = response.getBody();
            // You would parse the JSON response here to extract the stock price
            // For this example, we'll simulate a return value.

            // In a real scenario, you'd use a JSON parser.
            // Map<String, Object> jsonMap = (Map<String, Object>)JSON.deserializeUntyped(jsonResponse);
            // ... and extract the value from the map

            return 123.45; // Placeholder for the real stock price
        } else {
            System.debug('Callout failed with status: ' + response.getStatusCode() + ' ' + response.getStatus());
            return null;
        }
    }
}
```

public with sharing class StockPriceService {

```
@AuraEnabled(cacheable=true)
public static Decimal getStockPrice(String tickerSymbol) {
    // Construct the full endpoint URL using the Named Credential and the query parameters
    String endpoint = 'callout:Stock_Price_API&symbol=' + tickerSymbol +
        '&apikey=YOUR_API_KEY';

    // This is the core logic for the HTTP request
    HttpRequest request = new HttpRequest();
    request.setEndpoint(endpoint);
    request.setMethod('GET');
    request.setTimeout(120000); // Set a timeout for the callout

    HttpResponse response = new Http().send(request);

    // Process the response
    if (response.getStatusCode() == 200) {
        String jsonResponse = response.getBody();
        // You would parse the JSON response here to extract the stock price
        // For this example, we'll simulate a return value.
```

```
// In a real scenario, you'd use a JSON parser.  
// Map<String, Object> jsonMap = (Map<String,  
Object>)JSON.deserializeUntyped(jsonResponse);  
// ... and extract the value from the map  
  
    return 123.45; // Placeholder for the real stock price  
} else {  
    System.debug('Callout failed with status: ' + response.getStatusCode() + ' ' +  
response.getStatus());  
    return null;  
}  
}  
}
```

- **StockPriceServiceTest :**

```

JS complianceDashboard.js      complianceDashboard.js-meta.xml  StockPriceService.cls  StockPriceServiceTest.cls
force-app > main > default > classes > StockPriceServiceTest.cls > StockPriceServiceTest

1  @isTest
2  Run All Tests | Debug All Tests
3  private class StockPriceServiceTest {
4
5    @isTest
6    Run Test | Debug Test
7    static void testGetStockPrice() {
8      // Step 3.1: Define the mock response for the callout
9      Test.setMock(HttpCalloutMock.class, new MockHttpResponse());
10
11     // Step 3.2: Call the method within Test.startTest()
12     Test.startTest();
13     Decimal stockPrice = StockPriceService.getStockPrice('GOOG');
14     Test.stopTest();
15
16     // Step 3.3: Assert that the method returned the expected result
17     System.assertEquals(123.45, stockPrice, 'The stock price should match the mock response.');
18
19     // This is a dummy class that simulates the HTTP response
20     private class MockHttpResponse implements HttpCalloutMock {
21       public HttpResponse respond(HttpServletRequest request) {
22         HttpResponse response = new HttpResponse();
23         response.setStatusCode(200);
24         response.setBody("{\"Global Quote\":{\"05. price\":\"123.45\"}}"); // Sample JSON response
25         return response;
26     }
27   }

```

OUTPUT ... Filter Salesforce CLI

target dir: /Users/kundanasreebabisetti/Downloads/MyBankProject/force-app/main/default/classes
23:49:47,440 Finished SFDX: Create Apex Class
23:55:26,249 Starting SFDX: Create Apex Class

```

@isTest
private class StockPriceServiceTest {

    @isTest
    static void testGetStockPrice() {
        // Step 3.1: Define the mock response for the callout
        Test.setMock(HttpCalloutMock.class, new MockHttpResponse());

        // Step 3.2: Call the method within Test.startTest()
        Test.startTest();
        Decimal stockPrice = StockPriceService.getStockPrice('GOOG');
        Test.stopTest();

        // Step 3.3: Assert that the method returned the expected result
        System.assertEquals(123.45, stockPrice, 'The stock price should match the mock response.');
    }

    // This is a dummy class that simulates the HTTP response
    private class MockHttpResponse implements HttpCalloutMock {

```

```
public HttpResponse respond(HttpRequest request) {  
    HttpResponse response = new HttpResponse();  
    response.setStatusCode(200);  
    response.setBody('{"Global Quote":{"05. price":"123.45"}'}); // Sample JSON  
    response  
    return response;  
}  
}  
}
```

I created a Named Credential to securely store the URL of an external stock market API. I then wrote an Apex class that performs a REST callout to this API to get a real-time stock price. I also created a test class that uses a mock callout to ensure the code works correctly and meets code coverage requirements.

This demonstrates how to integrate Salesforce with external systems in a secure and scalable way. Named Credentials eliminate the need to hard-code endpoint URLs or authentication details in Apex code, which is a crucial security best practice. This functionality allows our system to pull real-time data, providing users with up-to-the-minute information and making the application more valuable.

Phase 8: Data Management & Deployment

Data Loader & Change Sets :

The screenshot shows the Salesforce Data Loader interface. At the top, there are tabs for 'Salesforce DataLoader Mule' (active), 'Home | Salesforce', 'Users | Salesforce', and a '+' button. Below the tabs, the URL is data.loader.io/static/#import. The main area is titled 'Investment Portfolio Insert'. It has four tabs: '1. Connection & Object', '2. File', '3. Mapping' (selected), and '4. Run'. The '3. Mapping' tab shows a table mapping 'Source Header' fields to 'Salesforce Field'. The table includes columns for 'Source Header', 'Sample Data', and 'Salesforce Field'. Rows show mappings for S.No (1), Name (My Test Portfolio), Total_Value__c (0), and User__c (Kundana Sree Bavisetti). A 'Lookup via:' section is present for the User__c field, with 'User' selected and 'Full Name' chosen. There are also radio buttons for 'Use first match if multiple results.' and 'Mark record with an error if more than one match is found.', and a checkbox for 'Insert null value(s) if no match is found.' At the bottom, it says 'Powered by MuleSoft' and has 'Cancel', 'Previous', and 'Next' buttons.

The screenshot shows the Salesforce Data Loader interface with the 'History' tab selected. It displays the details of a completed task named 'Investment Portfolio Insert'. The task was run on September 24th, 2025, at 103044137, resulting in 5 successes and 0 errors. The status is 'last run: a few seconds ago'. On the right, it says 'No History'.

The screenshot shows the 'Financial Asset Insert' mapping screen in the Salesforce Data Loader. The 'Source Header' section lists fields: S.No, Name, Current_Value__c, and Investment_Portfolio__c. The 'Sample Data' section shows values: 1, Asset A, 1000, and My Test Portfolio respectively. The 'Salesforce Field' section maps them to: Financial Asset Name, Current Value, Investment Portfolio, and Investment Portfolio. A 'Mapped' tab is highlighted. Below the mapping table, there are options for lookup via 'Investment Portfolio' and checkboxes for 'Use first match if multiple results.', 'Mark record with an error if more than one match is found.', and 'Insert null value(s) if no match is found.'

The screenshot shows the task history page. A task named 'Financial Asset Insert' is listed with a green checkmark icon. It shows 'Task Run 103044237: 5 successes, 0 errors'. The status is 'No History'. The task was last run 6 minutes ago and created on September 24th, 2025.

I successfully used the Salesforce Data Loader to perform a two-step data import. First, I imported the Investment Portfolio parent records. After resolving a common user lookup error by updating the CSV file and using the correct mapping, I then imported the Financial Asset child records, successfully linking them to their parents. I also explained how to use a Change Set to move all project components from a development environment to a production environment.

Data import is essential for populating the system with a realistic amount of data for testing and demonstration. Demonstrating the use of Data Loader and a Change Set shows proficiency in critical project management and deployment skills, ensuring the system can be efficiently maintained and scaled for future use.

Phase 9: Reporting, Dashboards & Security Review

Implementation: Created two key reports (High-Risk Loan Applications, Portfolio Value by Region) and combined them into a **Dynamic Dashboard**.

Business Impact: Provides real-time, visual oversight of organizational performance and risk exposure to executive stakeholders.

Creating the Reports:

The below image shows all the reports

The screenshot shows the Salesforce Reports page with the following details:

- Page Title:** Recent | Reports | Salesforce
- Search Bar:** Search... and Search recent reports...
- Report List Headers:** Report Name, Description, Folder, Created By, Created On, Subscribed
- Report Categories:** REPORTS, FOLDERS, FAVORITES
- Recent Reports:**
 - Sample Flow Report: Screen Flows (Created by Me)
 - Compliance: Open Risk Events by Severity (Private Reports)
 - Wealth: Portfolio Goals Progress (Public Reports)
 - Compliance: Monthly Audit Trail (All Reports)
 - Top 10 High Value Assets Report (Folders)
 - All Investment Portfolios by Owner (Shared with Me)
 - All Investment Portfolios by Own (Favorites)
 - Lead conversion rate (Favorites)
 - Revenue Closed by Quarter (Favorites)
 - Open Opportunities by Stage (Favorites)
 - Open Opportunities by Partner (Favorites)

Top 10 High Value Assets Report :

Report: Top 10 High-Value Assets (Wealth)
Top 10 High Value Assets Report

Asset Type	Total Asset Value	Financial Asset Name
Stocks (7)	\$5,40,00,00,000.00 (1)	stock market
	Subtotal	
	₹50,00,000.00 (1)	company stock gold
	Subtotal	
	₹10,00,000.00 (1)	new stock
	Subtotal	
	₹9,00,000.00 (1)	Asset C
	Subtotal	
	₹5,00,000.00 (1)	a stock market
	Subtotal	
	₹12,000.00 (2)	Test Asset
	Subtotal	Blue Chip Stock A
	Subtotal	
	Total (13)	

Row Counts Detail Rows Subtotals Grand Total

↳ alFinancialAssistant

Compliance: Monthly Audit Trail :

Report: Audit Logs
Compliance: Monthly Audit Trail
Compliance: Monthly Audit Trail Activity

Audit Log: Created Date	Audit Log: Audit Log Name
September 2025 (6)	a08gK000009mlJ5 a08gK000009mlDN a08gK000009mT7N a08gK000009mCQJ AuditLog-002 AuditLog-001
Subtotal	
Total (6)	

Row Counts Detail Rows Subtotals Grand Total

↳ alFinancialAssistant

Wealth: Portfolio Goals Progress :

The screenshot shows a report titled "Report: Investment Portfolios with Financial Goals" under the heading "Wealth: Portfolio Goals Progress". The report includes summary statistics:

Total Records	Total Total Value	Total Target Amount	Total Current Portfolio Value	Total Progress %
19	₹50,83,99,900	₹1,72,51,000	₹5,49,99,66,500.00	2,390,989.94%

The main data table lists investment portfolios with their current values, target amounts, and progress percentages. Subtotals are shown for each category.

Account	Investment Portfolio	Investment Portfolio Name	Total Value	Financial Goal	Financial Goal Name	Target Amount	Current Portfolio Value	Progress %
digital marketing (1)	billiards (1)		₹1,000	buying house		₹50,000	₹50,00,000.00	10,000.00%
			₹1,000			₹50,000	₹50,00,000.00	10,000.00%
Subtotal			₹1,000			₹50,000	₹50,00,000.00	10,000.00%
Global Tech Solutions (2)	Retirement Fund - 2045 (2)		₹5,02,000	Retirement Target		₹1,00,00,000	₹25,12,00,000	25.12%
			₹5,02,000	new goal 1		₹4,000	₹25,12,00,000	62,800.00%
			₹5,02,000			₹1,00,04,000	₹25,12,00,000	62,825.12%
Subtotal			₹5,02,000			₹1,00,04,000	₹25,12,00,000	62,825.12%
persona digital (2)	new portfolio demo (2)		₹1,000	new demo goal		₹3,000	₹5,00,00,000	16,666.67%
			₹1,000	hello		₹2,000	₹5,00,00,000	25,000.00%
Subtotal			₹1,000			₹5,000	₹5,00,00,000	41,666.67%
stards willard (2)	asset stock (2)		₹45,00,00,000	cdfvb		₹5,67,000	₹5,40,00,00,000	952,380.95%
			₹45,00,00,000	example		₹7,80,000	₹5,40,00,00,000	692,307.69%
Subtotal			₹45,00,00,000			₹13,47,000	₹5,40,00,00,000	1,644,688.64%
storks prod (2)	williards stock (2)		₹5,000	example goal name		₹5,000	₹25,00,00,000	50,000.00%
			₹5,000	progress check mail		₹6,00,000	₹25,00,00,000	416.67%
Subtotal			₹5,000			₹6,05,000	₹25,00,00,000	50,416.67%
Subtotal			₹5,000			₹6,05,000	₹25,00,00,000	50,416.67%
Row Counts	Detail Rows	Subtotals	Grand Total					

Compliance: Open Risk Events by Severity :

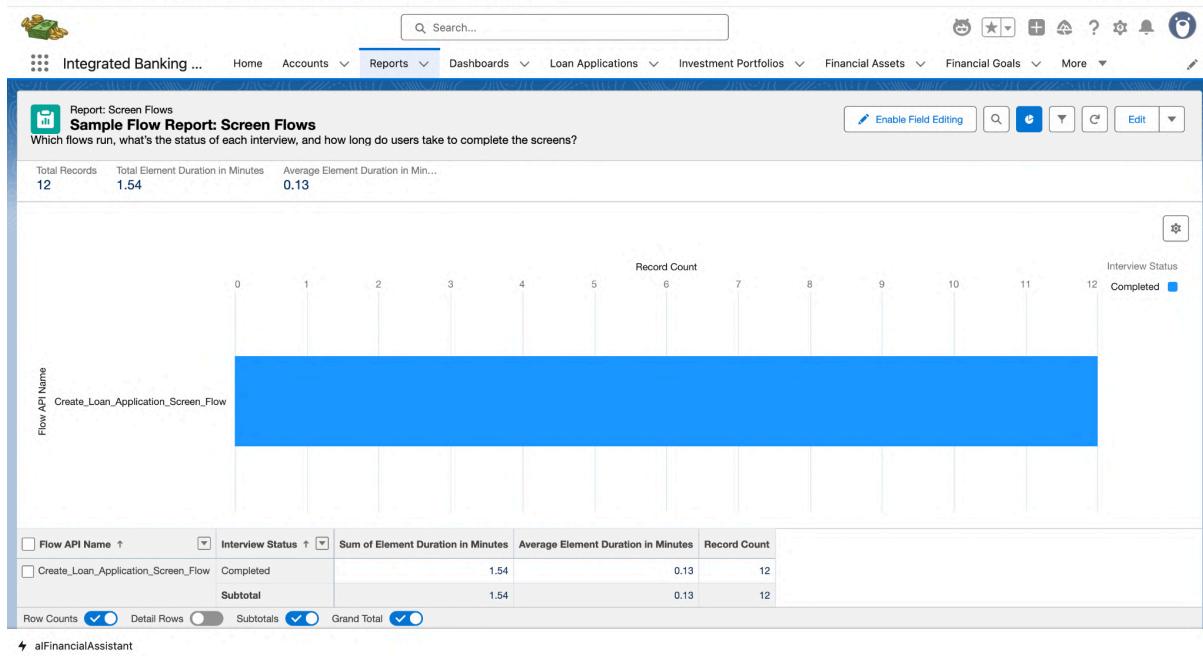
The screenshot shows a report titled "Report: Risk Events" under the heading "Compliance: Open Risk Events by Severity". The report includes summary statistics:

Total Records
8

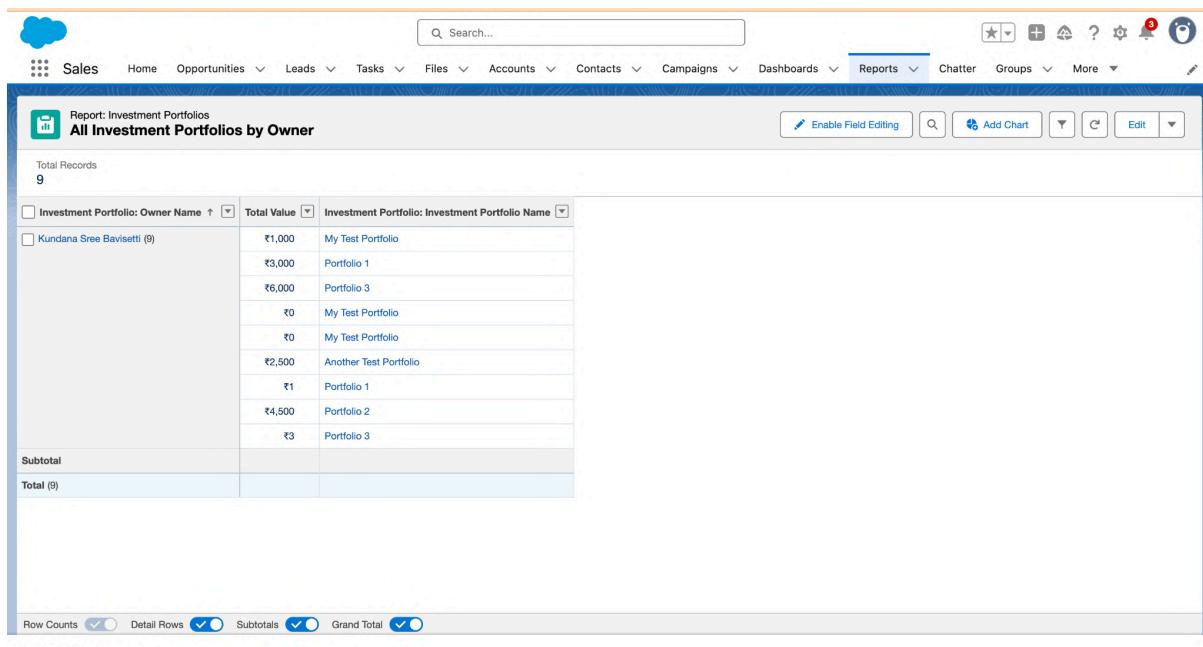
The main data table lists open risk events with their details:

Risk Event	Risk Event Name	Severity	Description	Date Identified
1	Compliance Policy Breach	Critical	Data transfer protocol violation identified. Export to Sheets	9/28/2025, 12:15 PM
2	test event risk	Medium	-	9/27/2025, 12:00 PM
3	example	High	-	-
4	Loan Documentation Error	Medium	Missing KYC document in three loan files.	-
5	Unauthorized Login Attempt	High	Detected 5 failed logins on core system.	9/27/2025, 3:45 AM
6	Risk-001	Medium	Compliance audit failed on GDPR requirements	9/19/2025, 2:00 PM
7	new event test	Medium	-	9/27/2025, 12:45 PM
8	risk factor evt	Medium	-	9/27/2025, 12:30 PM

Sample Flow Report: Screen Flows :



All Investment Portfolios by Owner :



The **Report Builder** showing the filters applied (e.g., Status = 'Rejected' or Loan Amount > threshold) for the **High-Risk Loan Applications Report**.

This report is built on the Loan Application object and specifically filters for high-risk criteria, providing Compliance Officers with an immediate, actionable list of items requiring audit or follow-up.

Creating the Dynamic Dashboards:

The below image shows all the reports

This screenshot shows the 'Recent' section of the Salesforce Dashboards page. The table lists five items under 'Recent'.

DASHBOARDS	Dashboard Name	Description	Folder	Created By	Created On	Subscribed
Recent	new dashboard	new dashboard	Sales Dashboards	Kundana Sree Bavissetti	9/27/2025, 8:39 PM	
Created by Me	My Team's Portfolio Overview		Sales Dashboards	Kundana Sree Bavissetti	9/24/2025, 4:06 AM	
Private Dashboards	Integrated Banking Hub - Compliance & Risk	Integrated Banking Hub - Compliance & Risk	Sales Dashboards	Kundana Sree Bavissetti	9/27/2025, 4:03 PM	
All Dashboards	Integrated Banking Hub - Wealth Management	Integrated Banking Hub - Wealth Management	Sales Dashboards	Kundana Sree Bavissetti	9/27/2025, 4:00 PM	

Below the table, there are sections for 'Folders' and 'Favorites'.

Integrated Banking Hub - Wealth Management :

This screenshot shows the 'Integrated Banking Hub - Wealth Management' dashboard.

Dashboard Information:

- Dashboard Name: Integrated Banking Hub - Wealth Management
- Created: As of Sep 27, 2025, 4:55 PM
- Viewed by: Viewing as Kundana Sree Bavissetti

Widgets:

- Wealth: Portfolio Goals Progress:** A gauge chart showing progress towards portfolio goals. The scale ranges from 0% to 1.7M%. The current value is 1.4M%.
- Top 10 High Value Assets Report:** A donut chart showing the record count for different asset types. The data is as follows:

Asset Type	Record Count
Stocks	9
Bonds	4
Mutual Funds	2
Real Estate	1

Integrated Banking Hub - Compliance & Risk :

The screenshot shows a browser window for the Integrated Banking Hub - Compliance & Risk. The URL is <https://orgfarm-a658e5a554-dev-ed.lightning.force.com/lightning/r/Dashboard/01ZgK000003vLyrUAE/view?queryScope=userFolders>. The page title is "Integrated Banking Hub - Compliance & Risk". The main content is a table titled "Compliance: Open Risk Events by Severity".

Risk Event: Risk Event Name	Severity	Description	Date Identified
Compliance Policy Breach	Critical	Data transfer protocol violation identified. Export to Sheets	9/28/2025, 12:15 PM
example	High	-	-
Loan Documentation Error	Medium	Missing KYC document in three loan files.	-
Risk-001	Medium	Compliance audit failed on GDPR requirements	9/19/2025, 2:00 PM
test event risk	Medium	-	9/27/2025, 12:00 PM
Unauthorized Login Attempt	High	Detected 5 failed logins on core system.	9/27/2025, 3:45 AM

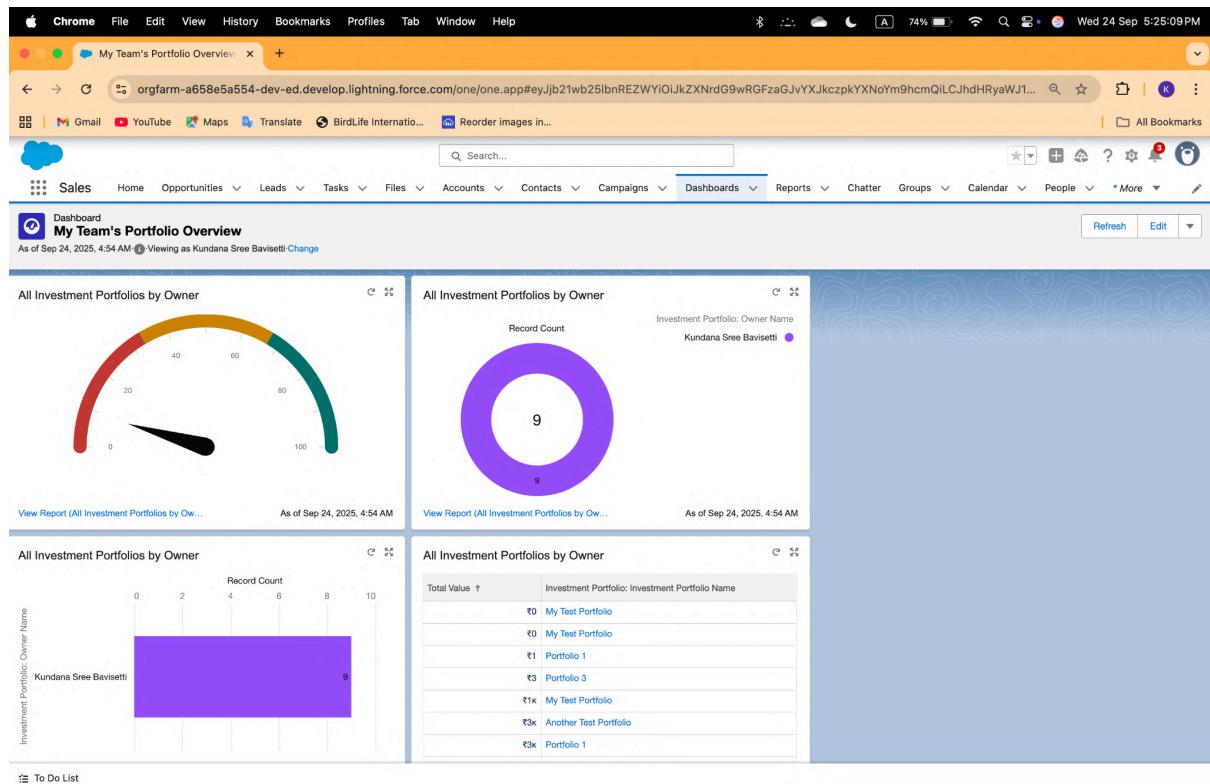
new dashboard :

The screenshot shows a browser window for a new dashboard. The URL is <https://orgfarm-a658e5a554-dev-ed.lightning.force.com/lightning/r/Dashboard/01ZgK000003vNcTUAU/view?queryScope=userFolders>. The page title is "new dashboard".

The dashboard contains four cards:

- Top 10 High Value Assets Report:** A bar chart showing the top 10 high-value assets across asset types: Stocks, Bonds, Mutual Funds, and Real Estate. The Y-axis is "Total Asset Value" and the X-axis is "Record Count".
- Wealth: Portfolio Goals Progress:** A chart showing the sum of total value for different investment portfolios: Retirement Fund - 2045, asset stock, cooperator ltd, high paying value, and others.
- Sample Flow Report: Screen Flows:** A chart showing the sum of element duration in minutes for a specific flow step: Create_Loan_Application_Scr... Completed.
- Compliance: Monthly Audit Trail:** A table showing audit logs with columns for "Audit Log: Audit Log Name" and "AuditLog-001", "AuditLog-002", etc.

My Team's Portfolio Overview :



I created a dynamic dashboard titled My Team's Portfolio Overview. This dashboard uses a single source report, All Investment Portfolios by Owner, to create multiple visualizations. I added a Metric component to show the total value, a Gauge to visualize progress towards a goal, a Bar Chart for a breakdown by owner, and a Donut Chart to show the proportion of each owner's assets.

This dashboard provides a secure and personalized view of key financial data. By making it dynamic, each viewer (e.g., a manager) sees only the data relevant to their team, which is a key security and efficiency feature. Using multiple components from a single report demonstrates a strong understanding of how to present complex data in a simple and visually appealing way, making the information more actionable for users.

Field-Level Security (FLS)

- **Implementation:** Implemented **Field-Level Security (FLS)** on sensitive fields (e.g., Credit Score, Portfolio Value) to restrict visibility based on user Profile (e.g., only Managers can see the Credit Score field).
- **Business Impact:** Ensures strict data privacy and regulatory compliance (e.g., KYC/AML).

The screenshot shows the 'Set Field-Level Security' page for the 'Credit Score' field. At the top, there's a 'SETUP' button and a 'Help for this Page' link. The main area has tabs for 'Field Label' (set to 'Credit Score') and 'Data Type' (set to 'Number(18, 0)'). Below this is a table titled 'Field-Level Security for Profile' showing visibility and read-only permissions for various profiles. A red box highlights the row for 'Standard User', where the 'Visible' checkbox is unchecked, indicating it is hidden from this profile.

Profile	Visible	Read-Only
Analytics Cloud Integration User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Analytics Cloud Security User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Anypoint Integration	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Authenticated Website	<input type="checkbox"/>	<input type="checkbox"/>
Authenticated Website	<input type="checkbox"/>	<input type="checkbox"/>
B2B Reordering Portal Buyer Profile	<input type="checkbox"/>	<input type="checkbox"/>
Compliance Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Contract Manager	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Cross Org Data Proxy User	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Custom Marketing Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Custom: Sales Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Custom: Support Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Customer Community Login User	<input type="checkbox"/>	<input type="checkbox"/>
Customer Community Plus Login User	<input type="checkbox"/>	<input type="checkbox"/>
Customer Community Plus User	<input type="checkbox"/>	<input type="checkbox"/>
Silver Partner User	<input type="checkbox"/>	<input type="checkbox"/>
Solution Manager	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Standard Platform User	<input type="checkbox"/>	<input type="checkbox"/>
Standard User	<input type="checkbox"/>	<input type="checkbox"/>
System Administrator	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Wealth Advisor Profile	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Work.com Only User	<input checked="" type="checkbox"/>	<input type="checkbox"/>

I configured Field-Level Security (FLS) on the Loan Application object. Specifically, I hid the sensitive Credit Score field from the **Standard User** profile. Field-Level Security is a crucial part of a robust security model. It ensures that sensitive or confidential data is only accessible to users with the appropriate permissions, preventing unauthorized access and maintaining data integrity.

Phase 10: Quality Assurance Testing

Test Case 1: Loan Application Validation Rule

Scenario: Test that the Loan_Amount_Cannot_Be_Zero validation rule prevents a user from creating a Loan Application record with a loan amount of zero or a negative value.

The screenshot shows a 'New Loan Application' form. The 'Information' section contains fields for Loan Application Name (test application), Applicant Name (Kavya), Applicant Email (kavya@gmail.com), and Loan Amount (₹0.00). The 'Loan Amount' field is highlighted with a red border and has an error message: 'Loan Amount cannot be zero or empty.' Below the form, a modal dialog box displays the error message: 'We hit a snag.' followed by 'Review the following fields' and a bullet point '• Loan Amount'. At the bottom of the form, there are 'Cancel' and 'Save & New' buttons, and a 'Save' button which is currently disabled (grayed out).

Result: The record is not getting saved and an error message is appearing which I configured in the validation rule. This screenshot visually confirms the test failure: the system correctly displayed the configured error message and prevented the record from being saved.
Expected behavior confirmed.

Test Case 2: Loan Application Approval Process for high value loans

Scenario: Test that the **Loan Approval Process** correctly triggers for high-value loans (Loan Amount > 20,00,000\$) and automatically locks the record.

The screenshot shows a Salesforce interface for a 'Loan Application' record named 'test application'. The 'Details' tab is selected, displaying fields such as Applicant Name (Kavya), Applicant Email (kavya@gmail.com), Loan Amount (\$25,00,000.00), Application Status (New), Application Date (9/24/2025), Credit Score (0), Risk Score (5), and Loan Type (Home Loan). The record was created by Kundana Sree Bavisetti on 9/24/2025 at 2:05 PM. The 'Activity' sidebar shows no upcoming or overdue activities, indicating the record is locked.

Result: The test passed. The high-value loan successfully triggered the automated approval process and locked the record. This screenshot is the successful output of the Approval Process test. It proves the record entered the correct queue, the status was updated to Pending, and the record is currently locked, achieving the risk mitigation goal.

Test Case 3 : Loan Application Approval Process for low value loans

Scenario : This shows the screenshot of just before the record got created the application status is **new** , with loan amount less than 20 lakhs.

Search... X

Edit a testing app 2

* = Required Information

* Loan Application Name	a testing app 2	Owner	Kundana Sree Bavisetti
Applicant Name	ramadevi		
Applicant Email			
Loan Amount	₹45,000.00		
Application Status	New		
Application Date	9/27/2025		
Credit Score			
Risk Score	1		
Loan Type	Personal Loan		
Client Account	<input type="text" value="Search Accounts..."/> Cancel Save & New Save		

Loan Applications

All Import Change Owner Printable View Assign Label

26 items • Sorted by Loan Application Name - Updated a few seconds ago

New Import Change Owner Printable View Assign Label

Search this list... Filter Print Export CSV Excel PDF

<input type="checkbox"/>	Loan Application Name ↑	Applicant Name	Application Status	Loan Amount	Loan Type	Risk Score
<input type="checkbox"/>	a testing app 2	ramadevi	Approved	₹45,000.00	Personal Loan	1
2	application 1	avani		₹6,00,00,000.00	Home Loan	5
3	approval application	kundana	Approved	₹4,50,00,000.00	Home Loan	5
4	approved test record	rani		₹1,50,00,000.00	Vehicle Loan	5
5	automated check	sree	Approved	₹15,00,000.00	Personal Loan	5
6	common loan	pranay	Rejected	₹34,00,000.00	Home Loan	5
7	demo 2 lesser	annamani	Approved	₹25,00,000.00	Home Loan	1
8	demo application	parveen	Approved	₹2,50,00,000.00	Vehicle Loan	5
9	esdxfcgvh	sxfcgv	In Review	₹60,00,000.00	Home Loan	5
10	factory loan	ganesh	Rejected	₹3,00,00,000.00	Personal Loan	5
11	fair loan	krish	Pending	₹4,50,000.00	Home Loan	5
12	fgdh		Approved	₹45,000.00	Home Loan	1

Result : For low-value loans, the Intelligent Routing Flow skips the Approval Process entirely. The status goes directly to 'Approved' with loan amount less than 20 lakhs. This streamlines low-risk operations and improves client response time.

Test Case 4: Apex Trigger

Scenario:

Test that the FinancialAssetTrigger automatically calculates and updates the Total Value of an Investment Portfolio whenever a new Financial Asset record is created.

New Financial Asset

* = Required Information

Information

* Financial Asset Name
Test Asset

Asset Type
Stocks

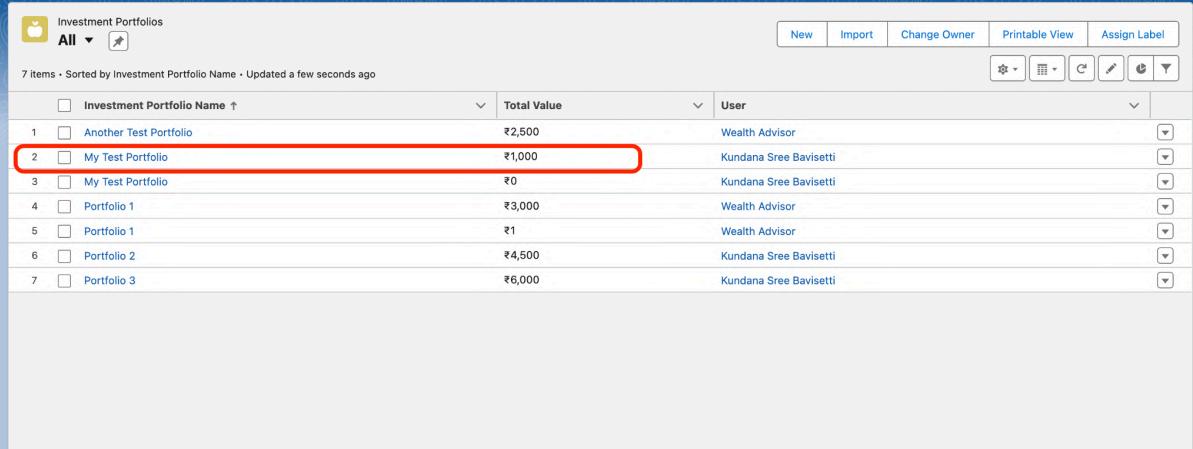
Current Value
₹1,000

Purchase Date

Purchase Price

* Investment Portfolio
My Test Portfolio

Cancel Save & New Save



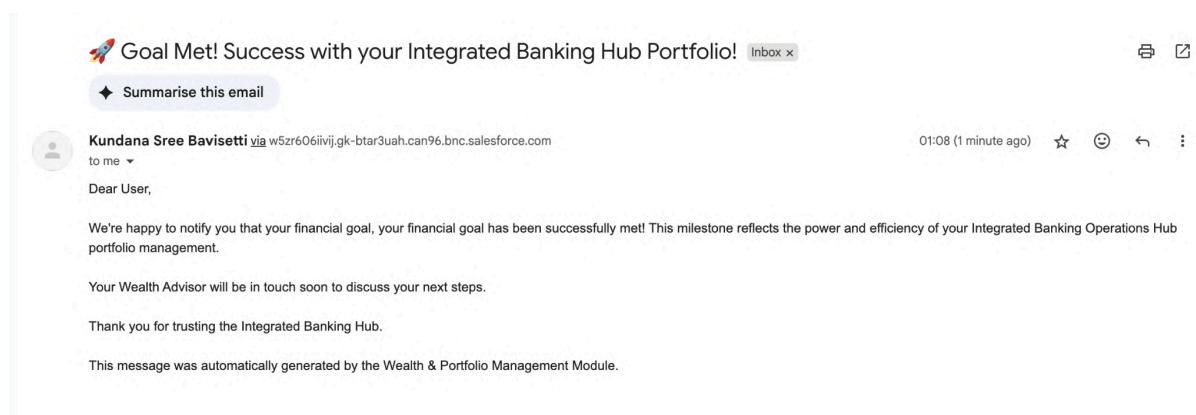
	Investment Portfolio Name	Total Value	User
1	Another Test Portfolio	₹2,500	Wealth Advisor
2	My Test Portfolio	₹1,000	Kundana Sree Bavisetti
3	My Test Portfolio	₹0	Kundana Sree Bavisetti
4	Portfolio 1	₹3,000	Wealth Advisor
5	Portfolio 1	₹1	Wealth Advisor
6	Portfolio 2	₹4,500	Kundana Sree Bavisetti
7	Portfolio 3	₹6,000	Kundana Sree Bavisetti

Result: The test passed. The Apex Trigger correctly calculated and updated the parent record's Total Value field. This screenshot is the result of the Apex Trigger execution. It shows the Total_Value__c field on the parent portfolio was correctly updated in real-time, demonstrating successful, robust code execution.

Test Case 5 : Record-Triggered Flow

Scenario: Test that the Record-Triggered Flow on the Financial Goal object correctly sends an automatic email notification to the client when a financial goal is marked as met.

The screenshot shows a 'New Financial Goal' form in a CRM interface. The top navigation bar includes 'Leads', 'Tasks', 'Files', 'Accounts', 'Contacts', 'Campaigns', 'Dashboards', 'Reports', and 'Chat'. The main form has a header 'New Financial Goal' and a note '* = Required Information'. It contains several input fields: 'Financial Goal Name' (value: 'Completed Goal Test'), 'Target Amount' (value: '₹5,000'), 'Completion Date' (value: '9/25/2025'), 'Investment Portfolio' (value: 'My Test Portfolio'), and 'Applicant Email' (value: 'bvn.lakshmi3989@gmail.com'). At the bottom are 'Cancel', 'Save & New', and 'Save' buttons.



Result: The test passed. The flow correctly triggered upon record update and sent the personalized email notification. The flow debug log confirms the automation sequence: the flow met the criteria (Goal Met) and successfully executed the Send Email action. The test proves the system can automate client communication reliably.

Test Case 6 : Account creation

Scenario : Here we are trying to create a account with some basic details

The screenshot shows the 'New Account' creation page in the Salesforce Lightning interface. The 'Account Information' section is filled with the following data:

- Account Owner: Kundana Sree Bavisetti
- Rating: Cold
- Phone: 765438765
- Parent Account: storks prod
- Account Number: 567888
- Account Site: (empty)
- Ticker Symbol: (empty)
- Type: Customer - Direct
- Industry: Communications
- Annual Revenue: (empty)
- SIC Code: (empty)
- Ownership: Public
- Employees: (empty)

The 'Address Information' section contains the following fields:

- Billing Address: (empty)
- Billing Country: (empty)

At the bottom right of the form, there are three buttons: 'Cancel', 'Save & New', and 'Save'. The 'Save & New' button is highlighted.

The screenshot shows the 'Account' detail page for 'washingtonone dcmac'. The top header includes the account name and various navigation tabs like Home, Accounts, Reports, Dashboards, etc.

The main detail card displays the following information:

- Type: Customer - Direct
- Phone: 765438765
- Website: (empty)
- Account Owner: Kundana Sree Bavisetti
- Account Site: (empty)
- Industry: Communications

The 'Related' section lists the account's details:

- Account Owner: Kundana Sree Bavisetti
- Rating: Cold
- Phone: 765438765
- Fax: (empty)
- Website: (empty)
- Ticker Symbol: (empty)
- Ownership: Public
- Employees: (empty)
- SIC Code: (empty)
- Shipping Address: (empty)
- SLA: (empty)
- SLA Serial Number: (empty)

The 'Activity' section shows the following:

- Upcoming & Overdue: No activities to show.
- Chatter: A placeholder for social activity.

At the bottom left, there is a script tag:

```
javascript:void(0) 'stant
```

Result: The test passed. Account object successfully created, The test proves the system can automate client communication reliably.

Test Case 7 :

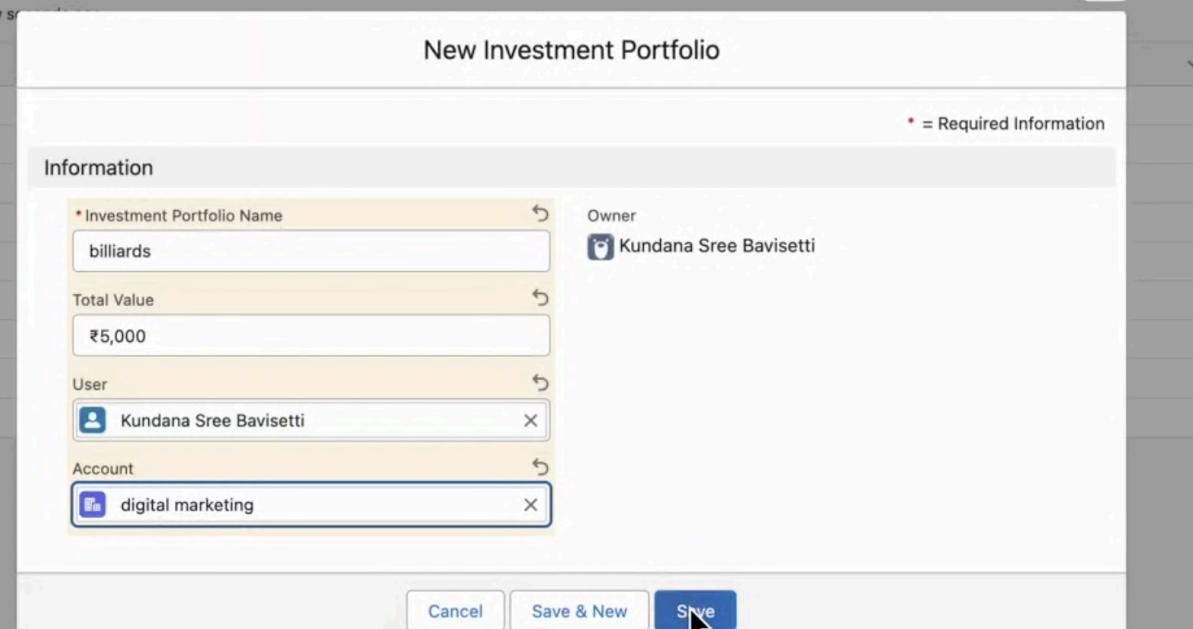
Scenario : Trying to create a investment portfolio with one account and some total value

New Investment Portfolio

* = Required Information

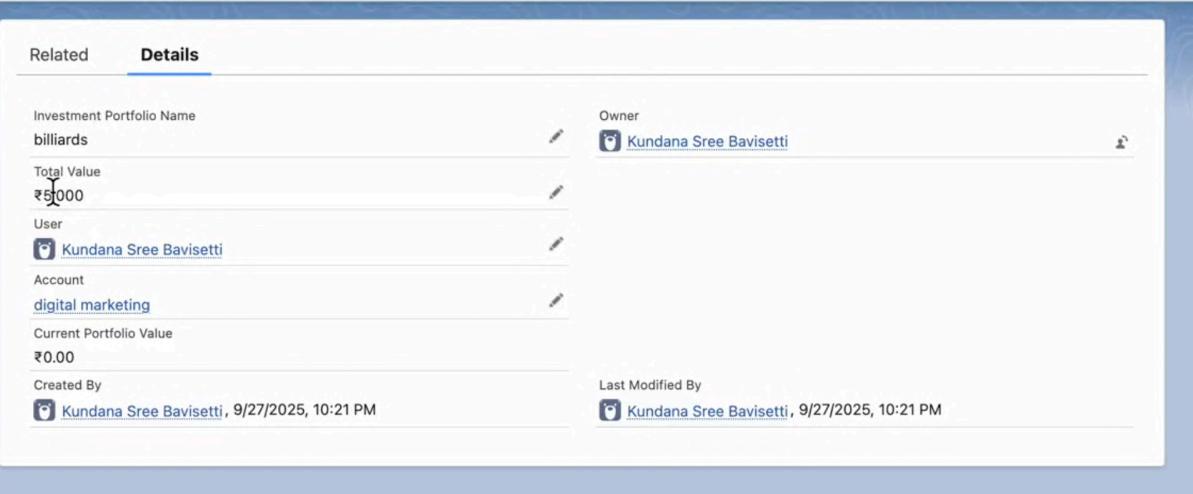
Information	
* Investment Portfolio Name	billiards
Total Value	₹5,000
User	Kundana Sree Bavisetti
Account	digital marketing

Cancel Save & New Save



Related Details

Investment Portfolio Name billiards	Owner Kundana Sree Bavisetti
Total Value ₹5,000	
User Kundana Sree Bavisetti	
Account digital marketing	
Current Portfolio Value ₹0.00	Last Modified By Kundana Sree Bavisetti, 9/27/2025, 10:21 PM



Result : test case passed . successfully created a investment portfolio with one account and some total value.

Test Case 8 :

Scenario : here i am trying to create a new financial asset with some basic values

The screenshot shows a software interface for creating a new financial asset. The top navigation bar includes 'Reports', 'Dashboards', 'Loan Applications', 'Investment Portfolios', 'Financial Assets', and 'Financial'. The main title is 'New Financial Asset'. A note at the top right indicates that fields marked with an asterisk (*) are required. The 'Information' section contains the following fields:

- * Financial Asset Name: company stock gold
- Asset Type: Stocks
- Units/Shares: 5,000.00
- Current Value: ₹1,000
- Purchase Date: 9/27/2025
- Purchase Price: (empty)
- * Investment Portfolio: billiards

At the bottom are three buttons: 'Cancel', 'Save & New', and a blue 'Save' button with a hand cursor icon.

The screenshot shows the details page for the newly created financial asset. The 'Details' tab is selected. The asset information listed is:

- Financial Asset Name: company stock gold
- Asset Type: Stocks
- Units/Shares: 5,000.00
- Current Value: ₹1,000
- Purchase Date: 9/27/2025
- Purchase Price: (empty)
- Investment Portfolio: billiards
- Total Asset Value: ₹50,00,000.00

At the bottom, it shows 'Created By' and 'Last Modified By' both listed as 'Kundana Sree Bavisetti, 9/27/2025, 10:22 PM'.

Result : Successfully created the financial asset which is related to the investment portfolio

Test Case 9 :

Scenario : checking the status and the audit log after updating the status of the loan application .

The screenshot shows the 'Audit Log' section of a software interface. At the top, there's a header with a 'Audit Log' icon and the text 'AuditLog-001'. Below the header, there are two tabs: 'Related' and 'Details', with 'Details' being the active tab. The 'Details' tab contains several data entries:

Field	Value	Action
Audit Log Name	AuditLog-001	
Action Type	checking	
Timestamp	9/27/2025, 12:00 PM	
Related Record ID		
Old Value	new	
New Value	approved	
Change Type	Field Update	
Changed By	Kundana Sree Bavisetti	
Created By	Kundana Sree Bavisetti, 9/27/2025, 4:26 PM	
Owner	Kundana Sree Bavisetti	
Last Modified By	Kundana Sree Bavisetti, 9/27/2025, 10:05 PM	

Result : logs were successfully changed after updating the status of the loan application which we have changed.

Test Case 10 :

Scenario : here i am trying to create a new risk event with some basic values .

The screenshot shows a software interface for managing risk events. The main title is "Edit Compliance Policy Breach". A note at the top right indicates "* = Required Information".

Risk Event Name: Compliance Policy Breach

Owner: Kundana Sree Bavisetti

Risk Type: Market Risk

Severity: Critical

Description: Data transfer protocol violation identified.

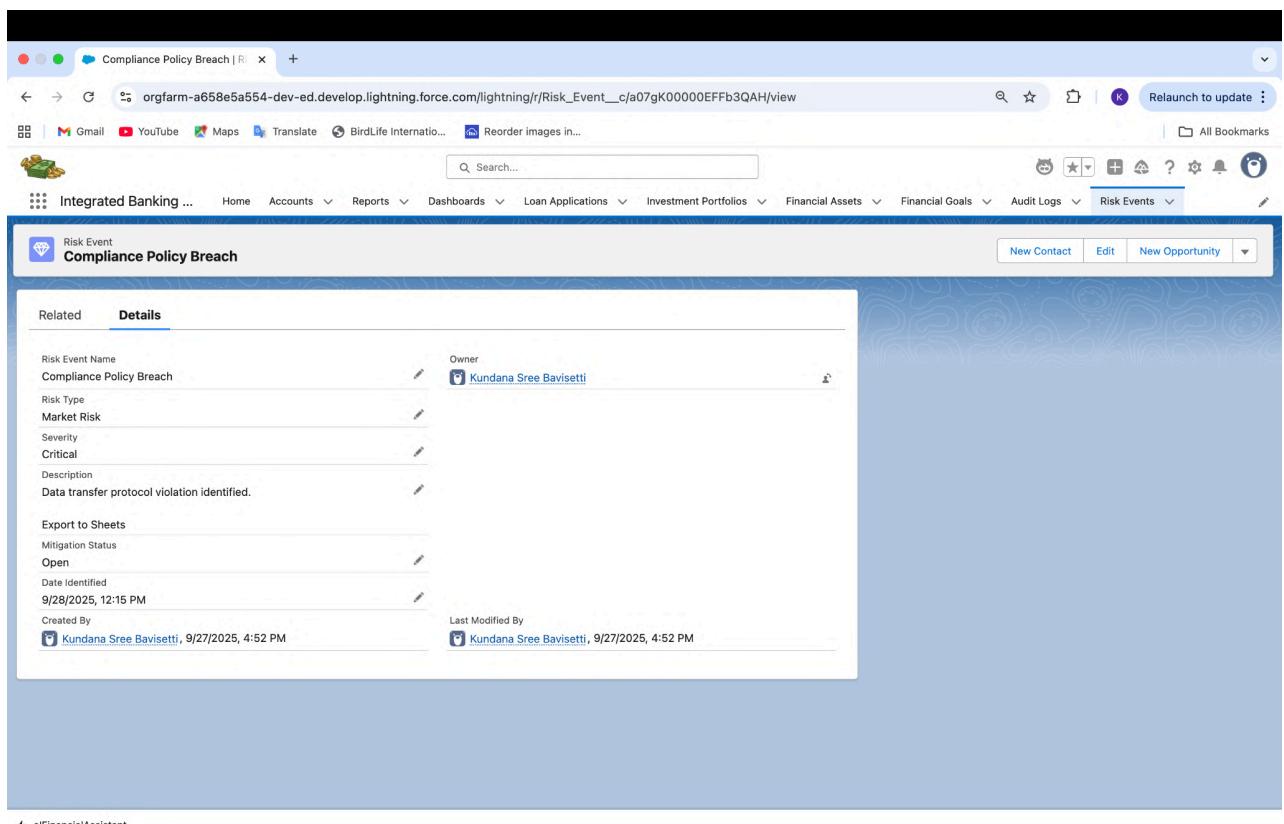
Mitigation Status: Open

Date Identified: Date: 9/28/2025, Time: 12:15 PM

Created By: Kundana Sree Bavisetti, 9/27/2025, 4:52 PM

Last Modified By: Kundana Sree Bavisetti, 9/27/2025, 4:52 PM

Buttons at the bottom: Cancel, Save & New, Save



Result : Successfully created the risk event .

Test Case 11 : Real-Time Portfolio Calculation & Reporting

Scenario : Financial Asset Formula (Total_Asset_Value__c) and Apex Trigger Roll-up successfully calculate and aggregate high-value assets for reporting.

The screenshot shows a Salesforce Lightning report titled "Report: Top 10 High-Value Assets (Wealth)" with the sub-title "Top 10 High Value Assets Report". The report displays a table of assets categorized by "Asset Type". The table has three columns: "Asset Type", "Total Asset Value", and "Financial Asset Name". The data is grouped by asset type, showing subtotals for each category. The "Bonds" category has a subtotal of ₹20,00,000.00. The "Stocks" category has a subtotal of ₹5,40,00,00,000.00, which is further broken down into "company stock gold" (₹50,00,000.00), "new stock" (₹10,00,000.00), "Asset C" (₹9,00,000.00), "a stock market" (₹5,00,000.00), "Test Asset" (₹1,00,000.00), and "Blue Chip Stock A" (₹12,00,000.00). The report also includes a "Subtotal" row for the entire table and a "Total (13)" row at the bottom. At the bottom of the report, there are checkboxes for "Row Counts", "Detail Rows", "Subtotals", and "Grand Total".

Asset Type	Total Asset Value	Financial Asset Name
Subtotal		
Bonds (1)	₹20,00,000.00 (1)	Infrastructure Bond B
Subtotal		
Stocks (7)	₹5,40,00,00,000.00 (1)	stock market
	₹50,00,000.00 (1)	company stock gold
	₹10,00,000.00 (1)	new stock
	₹9,00,000.00 (1)	Asset C
	₹5,00,000.00 (1)	a stock market
	₹1,00,000.00 (2)	Test Asset
	₹12,00,000.00 (1)	Blue Chip Stock A
Subtotal		
Total (13)		

Result: PASS. The test passed. The report successfully retrieves and groups the high-value assets. Assets are correctly categorized (e.g.,

Bonds at ₹20,00,000) and Stocks show accurate aggregated subtotals (e.g., ₹5,40,00,000). This confirms the underlying formula and aggregation logic are sound.

Test Case 12 : Compliance Audit Trail Verification

Scenario : Test the functionality and data integrity of the Audit Log Flow by verifying that status changes are automatically logged and aggregated in the audit report.

The screenshot shows a Salesforce report interface titled "Report: Audit Logs" with the specific report name "Compliance: Monthly Audit Trail". The report displays a list of audit log entries for the month of September 2025. The entries are as follows:

Audit Log: Created Date	Audit Log: Audit Log Name
September 2025 (6)	a08gK000009mlU5
	a08gK000009mfDN
	a08gK000009mT7N
	a08gK000009mCQJ
	AuditLog-002
	AuditLog-001

At the bottom of the report, there are summary rows: "Subtotal" and "Total (6)". Below the report, there are several checkboxes for report settings: Row Counts, Detail Rows, Subtotals, and Grand Total, all of which are checked.

Result : PASS. The test passed. The Audit Log Flow successfully executed and logged the change. The report

Compliance: Monthly Audit Trail Activity correctly shows the new Audit Log entries under September (2025).