

Problem Statement – Real Estate Fractional Investment Platform

Project Title:

Revaa – Real Estate Investment Simulation Platform

Background / Context:

Investing in real estate has traditionally been limited to high-net-worth individuals or institutional investors due to the large capital required to purchase complete properties. Many potential investors, especially from the lower and middle-income groups, cannot participate in real estate investment despite its potential for stable returns.

At the same time, managing multiple investors, fractional ownership, transactions, and property availability is complex and requires a robust system for tracking investments, generating profits, and maintaining transparency.

Problem Definition:

There is no simple and accessible platform that allows small investors to invest in real estate properties **as fractional shares**, with clear tracking of ownership, lot availability, and profit distribution. Investors need a system where they can:

1. **View available properties** with detailed information.
2. **Invest in fractional shares (lots)** according to their purchasing power.
3. **Track ownership and investment history** in real time.
4. **Receive profits** proportionate to their share of the property.
5. **Ensure transparency and automation** in transactions and lot availability.

Currently, the lack of a structured digital platform makes fractional investment inefficient, prone to errors, and inaccessible for small investors.

Objectives of the Project:

The project aims to **simulate a fractional real estate investment platform** using Salesforce to address these challenges. Specifically, it will allow:

1. Property Management:

- Maintain a catalog of real estate properties with details like location, price, and total lots.
- Track available lots per property dynamically.

2. Investor Management:

- Record investor details including contact information and total investment.
- Maintain a clear relationship between investors and their owned lots.

3. Investment Lots Management:

- Divide properties into fractional lots that investors can buy individually.
- Track lot status as available or sold.

4. Transaction Management:

- Record all investments made by investors.
- Maintain history of lot purchases and investment amounts.

5. Automation and Transparency:

- Automatically update available lots when a purchase occurs.
- Provide accurate calculations for total investments and lot availability.

Scope of the Project:

- Simulate real estate investment **without real money**, using Salesforce as the platform.
- Enable multiple investors to buy fractional shares of properties.
- Track transactions and investments using Salesforce custom objects, relationships, and automation features.
- Provide a user-friendly interface for managing properties, investors, lots, and transactions.

Expected Benefits:

- Make real estate investment **accessible to all** by simulating fractional ownership.
- Reduce complexity in tracking multiple investors and lots.
- Provide **real-time insights** on investments, lot availability, and transactions.
- Offer a scalable system that can be extended to real applications in the future.



Phase 1: Problem Understanding & Industry Analysis

1. Requirement Gathering

Business Objectives

- Build a **Fractional Real Estate Investment Simulation Platform** (Revaa) on Salesforce.
- Allow investors to **buy fractional shares** of real estate properties.
- Provide **ROI simulation tools** for investors before making decisions.
- Automate **rental income distribution** and **transaction recording**.
- Ensure **compliance** with KYC, taxation, and real estate regulations.
- Provide **dashboards and reports** for investors and admins.

Functional Requirements

- **Investor Management**
 - Onboard investors with KYC verification.
 - Manage investor profiles and ownership percentages.
- **Property Management**
 - Add, update, and maintain property details.
 - Track fractional ownership and availability.
- **Investment Transactions**
 - Record investments, payments, and ROI payouts.
 - Enable resale/exit functionality for investors.
- **Compliance & Security**
 - Enforce validation rules for KYC and financial checks.
 - Role-based access for investors, admins, and managers.
- **Simulation Engine**
 - ROI calculator for different investment scenarios.
 - Reports for historical and projected income.

Non-Functional Requirements

- Scalability to handle **thousands of investors and properties**.
- High security with Salesforce OWD, roles, and profiles.

- Mobile-friendly access via Salesforce mobile app.
 - Integration capability with financial institutions.
 - Low maintenance and cost-effective configuration.
-

2. Stakeholder Analysis

Stakeholder	Role	Interest/Needs	Influence Level	Pain Points
Investors	End Users	Easy onboarding, transparency, ROI clarity	High	Lack of trust, complicated processes
Property Managers	Manage properties	Simplified property listing & tracking	Medium	Manual property tracking
Platform Admins	Salesforce Admins	Control, configuration, compliance enforcement	High	Data security, high workload
Compliance Officers	Regulatory compliance managers	Ensure KYC, taxation, legal checks	High	Missing regulatory checks
Financial Partners	Banks, Payment Gateways	Seamless transactions, reconciliations	Medium	Delays in settlement
Real Estate Brokers	Secondary data providers	Provide listings and valuations	Low	Manual data exchange

3. Business Process Mapping

End-to-End Flow:

1. Property Acquisition

- Property manager lists a property on Revaa.
- Details like cost, expected rental income, and availability recorded.

2. Investor Onboarding

- Investor signs up → KYC verification → Profile created in Salesforce.

3. Fractional Investment Purchase

- Investor selects property → chooses fraction → invests money.

4. Rental Income Distribution

- Monthly/Quarterly rental income distributed proportionally.

5. Exit / Resale of Shares

- Investor resells share back to platform or another investor.

(Diagram to be created in Lucidchart/Draw.io and added later as Revaa_ProcessMap.png.)

4. Industry-Specific Use Case Analysis

Business Process Step	Salesforce Feature	Benefit
Investor KYC & Compliance	Validation Rules, Flows	Enforce KYC before activation
Property Portfolio Management	Custom Objects, Reports	Centralized view of all properties
Investment Simulation	Apex Logic + LWC	Dynamic ROI calculator for investors
Transactions & Ledger	Custom Object + Reports	Transparent investment records
Rental Payouts	Scheduled Flows / Apex	Automated periodic payouts
Resale of Shares	Custom Flow + Approval	Smooth investor exit process

5. AppExchange Exploration

Potential Relevant Apps:

1. Propertybase Real Estate CRM

- Real estate listing and CRM features.
- Cost: Paid, subscription-based.
- Relevance: Can integrate property listing module.

2. FinancialForce Accounting

- Cloud-based financial management on Salesforce.
- Cost: Paid (enterprise-level).
- Relevance: Useful for transaction and payout tracking.

3. DocuSign eSignature for Salesforce

- Digital signing of agreements and contracts.
- Cost: Paid, per-user license.
- Relevance: Investor agreements and property contracts.

4. ComplianceQuest

- End-to-end compliance and risk management.
- Cost: Paid.
- Relevance: KYC and compliance management.

Phase 2: Org Setup & Configuration – Revaa Project

Objective

Phase 2 ensures the Salesforce org is properly configured for the Fractional Real Estate Investment Simulation Platform. This includes setting up **company profile, users, profiles, roles, permission sets, security, and environment basics**. Proper configuration is essential for consistent data access, automation, and reporting in later phases.

Step 1: Salesforce Dev Org Setup

1. Sign up for a **Salesforce Developer Edition (CRM)**: developer.salesforce.com/signup
 2. Verify your email and log in.
 3. Confirm you are in **Lightning Experience UI**.
 4. Enable **My Domain** for better app navigation and Lightning components:
 - o Setup → My Domain → Choose a domain → Check availability → Save & Deploy.
-

Step 2: Company Profile Setup

1. Setup → Company Information → Fill in:
 - o Company Name: Revaa Technologies
 - o Default Currency: INR
 - o Default Locale: India
 - o Fiscal Year Start: April 1 (typical for India)
 - o Time Zone: GMT+5:30 (Asia/Kolkata)
 - o Enable Multiple Currencies if needed (optional).
 - o **Screenshot:**

The screenshot shows the Salesforce Setup interface. In the top navigation bar, 'Setup' is selected. Below it, a search bar contains 'Company information'. Under 'Company Settings', 'Company Information' is highlighted. A message says 'Didn't find what you're looking for? Try using Global Search.' The main content area is titled 'Company Information' with a blue header bar. It displays the organization profile for 'Revaa Technologies'. The profile includes basic details like Organization Name (Revaa Technologies), Primary Contact (OrgFarm EPIC), Address (Sai Nagar, Amravati 444607, Maharashtra, India), and contact information (Phone: (909) 694-9992, Fax: (909) 694-9992). It also shows fiscal year settings (Fiscal Year Starts In: March, Activate Multiple Currencies: checked), data translation (Enable Data Translation: unchecked, Newsletter: checked), and system notices (Admin Newsletter: checked, Hide Notices About System Maintenance: unchecked, Hide Notices About System Downtime: unchecked). On the right, there are sections for API requests, streaming API events, restricted logins, and instance details. At the bottom, there are links for User Licenses, Permission Set Licenses, Feature Licenses, and Usage-based Entitlements.

Step 3: Business Hours & Holidays

1. Setup → Business Hours → New:
 - Business Hours Name: Revaa Standard Hours
 - Days: Monday – Friday
 - Hours: 9:00 AM – 6:00 PM
 - Save.

2. Setup → Holidays → New:
 - Enter national holidays (e.g., Diwali, Independence Day).
 - **Screenshot:**

Business Hours Detail

Business Hours Name	Revaa Standard Hours	Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)														
Business Hours	<table border="1"> <tr><td>Sunday</td><td>24 Hours</td></tr> <tr><td>Monday</td><td>9:00 AM to 6:00 PM</td></tr> <tr><td>Tuesday</td><td>9:00 AM to 6:00 PM</td></tr> <tr><td>Wednesday</td><td>9:00 AM to 6:00 PM</td></tr> <tr><td>Thursday</td><td>9:00 AM to 6:00 PM</td></tr> <tr><td>Friday</td><td>9:00 AM to 6:00 PM</td></tr> <tr><td>Saturday</td><td>24 Hours</td></tr> </table>	Sunday	24 Hours	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	Saturday	24 Hours	Default Business Hours	<input checked="" type="checkbox"/>
Sunday	24 Hours																
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																
Saturday	24 Hours																
Active	<input checked="" type="checkbox"/>																
Created By	ANIKET KAKADE 9/26/2025, 9:58 PM	Last Modified By	ANIKET KAKADE 9/26/2025, 10:02 PM														
Edit																	

Holidays

Holiday Name	Description	Date and Time
Independence Day		8/15/2026 All Day Edit
Republic Day		1/1/2026 All Day Edit

Step 4: Fiscal Year Settings

1. Setup → Fiscal Year → Activate Standard Fiscal Year.
2. If using custom fiscal periods (optional), create them here.

- **Screenshot:**

Fiscal Year

Organization Fiscal Year Edit: Revaa Technologies

To specify the fiscal year type for your organization, choose one of the options below.

Fiscal Year Information

Your organization can change the fiscal year start month, and specify whether the fiscal year name is set to the starting or ending year. For example, if your fiscal year starts in April 2025 and ends in March 2026, your Fiscal Year setting can be either 2025 or 2026.

Change Fiscal Year Period

Name	Revaa Technologies	Save	Cancel
Fiscal Year Start Month	March	Save	Cancel
Fiscal Year Is Based On	<input checked="" type="radio"/> The ending month <input type="radio"/> The starting month	Save	Cancel

Step 5: User Setup & Licenses

1. Setup → Users → New User:

- License: Salesforce
- Profile: System Administrator (for admin users)
- Fill Name, Email, Username (unique format: user@revaa.com)
- Click Save.
- **Screenshot:**

The screenshot shows the 'User Detail' page in the Salesforce setup interface. The user is named 'Investor User' with alias 'invuser' and email 'aniketkakade045@gmail.com'. The profile is set to 'System Administrator'. The user is active and assigned to the 'Investor User' role. Various other settings like mobile push registrations, debug mode, and lightning page loading are also visible.

User Detail		Role	
Name	Investor User	User License	Investor User
Alias	invuser	Profile	Identity
Email	aniketkakade045@gmail.com [Verify]	Active	✓
Username	investor1@force.com	Marketing User	<input type="checkbox"/>
Nickname	User17589523385439349248	Offline User	<input type="checkbox"/>
Title		Knowledge User	<input type="checkbox"/>
Company		Flow User	<input type="checkbox"/>
Department		Service Cloud User	<input type="checkbox"/>
Division		Site.com Contributor User	<input type="checkbox"/>
Address		Site.com Publisher User	<input type="checkbox"/>
Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)	WDC User	<input type="checkbox"/>
Locale	English (United States)	Mobile Push Registrations	View
Language	English	Data.com User Type	Edit
Currency	INR - Indian Rupee	Accessibility Mode (Classic Only)	<input type="checkbox"/> Edit
Delegated Approver		Debug Mode	<input type="checkbox"/> Edit
Manager	Only if I am an approver	High-Contrast Palette on Charts	<input type="checkbox"/> Edit
Federation ID		Load Lightning Pages While Scrolling	<input checked="" type="checkbox"/> Edit
App Registration: One-Time Password Authenticator	Edit	Salesforce CRM Content User	<input type="checkbox"/>
App Registration: Salesforce Authenticator	Edit	Receive Salesforce CRM Content Email Alerts	<input type="checkbox"/>
Security Key (2FA or WebAuthn)	Edit	Receive Salesforce CRM Content Alerts	<input type="checkbox"/>

Step 6: Profiles

6.1 Default Profiles

- **System Administrator** → Full access (assigned to project admin).
- **Standard User** → Can view/create records but no admin privileges.

6.2 Custom Profiles (for Revaa)

- Clone **Standard User** → Name: **Investor Profile**
 - Permissions: Read/Write access to Investment, Property objects.
 - No access to Setup or App Builder.
- Clone **Standard User** → Name: **Manager Profile**
 - Permissions: Manage Properties, Transactions, view all investor records under their role.

- **Screenshot:**

The screenshot shows the Salesforce 'Profiles' page under the 'SETUP' tab. At the top, there's a 'New Profile' button and a navigation bar with letters A through S. Below the navigation is a table with columns for Action, Profile Name, User License, and Custom. The table lists various user profiles, each with an edit and clone link. Some profiles have a checked checkbox in the 'Custom' column, indicating they are custom profiles.

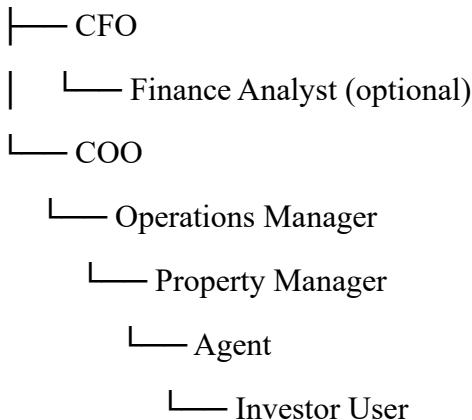
Action	Profile Name	User License	Custom
Edit Del ...	Agent Profile User	Salesforce	<input checked="" type="checkbox"/>
Edit Clone	Analytics Cloud Integration User	Analytics Cloud Integration User	<input type="checkbox"/>
Edit Clone	Analytics Cloud Security User	Analytics Cloud Integration User	<input type="checkbox"/>
Edit Clone	Anypoint Integration	Identity	<input type="checkbox"/>
Edit Clone	Authenticated Website	Authenticated Website	<input type="checkbox"/>
Edit Clone	Authenticated Website	Authenticated Website	<input type="checkbox"/>
Edit Del ...	B2B Reordering Portal Buyer Profile	External Apps Login	<input checked="" type="checkbox"/>
Edit Clone	Chatter External User	Chatter External	<input type="checkbox"/>
Edit Clone	Chatter Free User	Chatter Free	<input type="checkbox"/>
Edit Clone	Chatter Moderator User	Chatter Free	<input type="checkbox"/>
Edit Clone	Contract Manager	Salesforce	<input type="checkbox"/>
Edit Clone	Cross Org Data Proxy User	XOrg Proxy User	<input type="checkbox"/>
Edit Del ...	Custom: Marketing Profile	Salesforce	<input checked="" type="checkbox"/>
Edit Del ...	Custom: Sales Profile	Salesforce	<input checked="" type="checkbox"/>
Edit Del ...	Custom: Support Profile	Salesforce	<input checked="" type="checkbox"/>

1-25 of 46 ▾ 0 Selected ▾

Step 7: Roles

1. Setup → Roles → Set Up Roles → Click “Add Role”.
2. Build the **Role Hierarchy**:

CEO



3. Assign users to their respective roles when creating their user account.

- **Screenshot:**

The screenshot shows the Salesforce Setup Roles page. At the top, there's a header with a user icon and the word "SETUP". Below it, the title "Roles" is displayed. A toolbar with buttons for "New", "Edit", "Delete", "Assign", "Share", and "Help" is visible. The main content area is titled "Your Organization's Role Hierarchy". It shows a hierarchical tree of roles under "Revaa Technologies". The roles listed are: CEO, CFO, Compliance Officer, COO, Operations Manager, Property Manager, Real Estate Agent, Investor User, SVP Customer Service & Support, CTO, SVP Human Resources, and System Administrator. Each role has "Edit | Del | Assign" links and an "Add Role" link. A vertical scrollbar is on the right side of the page.

Step 8: Permission Sets

1. Setup → Permission Sets → New:

- Example: “Investor Extra Access”
 - Object permissions: Investment Transactions (Read Only)
 - Field-level permissions: Allow specific fields visibility.

2. Assign to users who need **extra access** beyond their profile.

- **Screenshot:**

The screenshot shows the Salesforce Permission Sets page. At the top, there's a header with a user icon and the word "SETUP". Below it, the title "Permission Sets" is displayed. A toolbar with buttons for "New", "Edit", "Delete", "Create New View", and "Help for this Page" is visible. The main content area is titled "Permission Sets". It shows a table of existing permission sets. The columns are: Action, Permission Set Name, Description, and License. The table includes rows for various legacy Data Cloud roles like "Data Cloud Data Aware Specialist", "Data Cloud Marketing Admin", etc., and newer Agentforce roles like "Access Agentforce Default Agent", "Agent Platform Builder", etc. A vertical scrollbar is on the right side of the page.

Action	Permission Set Name	Description	License
<input type="checkbox"/>	(Legacy) Data Cloud Data Aware Specialist	This Data Cloud permission set will be deprecated in Spring '24. Lea...	Customer Data Platform
<input type="checkbox"/>	(Legacy) Data Cloud Marketing Admin	Allows access to Data Cloud Setup if the user is also a Salesforce a...	Customer Data Cloud for Marketing
<input type="checkbox"/>	(Legacy) Data Cloud Marketing Manager	This Data Cloud permission set will be deprecated in Spring '24. Lea...	Customer Data Platform
<input type="checkbox"/>	(Legacy) Data Cloud Marketing Specialist	This Data Cloud permission set will be deprecated in Spring '24. Lea...	Customer Data Platform
<input type="checkbox"/>	(Legacy) Data Cloud for Marketing Data Aware Specialist	This Data Cloud permission set will be deprecated in Spring '24. Lea...	Customer Data Cloud for Marketing
<input type="checkbox"/>	(Legacy) Data Cloud for Marketing Manager	This Data Cloud permission set will be deprecated in Spring '24. Lea...	Customer Data Cloud for Marketing
<input type="checkbox"/>	(Legacy) Data Cloud for Marketing Specialist	This Data Cloud permission set will be deprecated in Spring '24. Lea...	Customer Data Cloud for Marketing
<input type="checkbox"/>	Access Agentforce Default Agent	Gives users access to the default Agentforce agent in Salesforce.	Agentforce (Default)
<input type="checkbox"/>	Agent Platform Builder	Allow access to agent platform.	Agent platform builder
<input type="checkbox"/>	Agentforce Default Admin	Allows users to build and manage in-org copilots.	Agentforce (Default)
<input type="checkbox"/>	Agentforce Service Agent Configuration	Build and manage autonomous AI service agents.	Agentforce Service Agent Builder
<input type="checkbox"/>	Agentforce Service Agent Object Access	Access knowledge articles and manage cases and contacts as an auto...	Agentforce Service Agent User
<input type="checkbox"/>	Agentforce Service Agent Secure Base	Set up and use Agentforce Service Agent actions with enhanced data s...	Agentforce Service Agent User
<input type="checkbox"/>	Agentforce Service Agent User	Analyze topics and perform actions as an autonomous AI service agent.	Agentforce Service Agent User

Step 9: Organization-Wide Defaults (OWD)

1. Setup → Sharing Settings → Organization-Wide Defaults:
 - Property: Private (so only owner/role hierarchy can see)
 - Investment Transaction: Private
 - Investor: Private
 - Profit Distribution: Private

2. Save changes.

- **Screenshot:**

The screenshot shows the 'Sharing Settings' page in the Salesforce setup. The title bar includes the 'SETUP' icon and the 'Sharing Settings' label. Below the title is a sub-header 'Organization-Wide Defaults' with an 'Edit' button and a 'Help' link. The main content is a table titled 'Organization-Wide Defaults' with columns for 'Object', 'Default Internal Access', 'Default External Access', and 'Grant Access Using Hierarchies'. The table lists various objects and their sharing settings. Most objects have 'Private' set as both internal and external access, except for 'Public Read/Write' or 'Controlled by Parent' for some like Lead, Account, Contact, Order, Asset, Opportunity, Case, Campaign, and Activity. The 'Grant Access Using Hierarchies' column contains a series of checkmarks.

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Lead	Public Read/Write/Transfer	Private	✓
Account and Contract	Public Read/Write	Private	✓
Contact	Controlled by Parent	Controlled by Parent	✓
Order	Controlled by Parent	Controlled by Parent	✓
Asset	Controlled by Parent	Controlled by Parent	✓
Opportunity	Public Read/Write	Private	✓
Case	Public Read/Write/Transfer	Private	✓
Campaign	Public Full Access	Private	✓
Campaign Member	Controlled by Campaign	Controlled by Campaign	✓
User	Public Read Only	Private	✓
Activity	Private	Private	✓
Calendar	Hide Details and Add Events	Hide Details and Add Events	✓
Price Book	Use	Use	✓
Product	Public Read/Write	Public Read/Write	✓
Individual	Public Read/Write	Private	✓
Voice Call	Private	Private	✓
Activation Target	Private	Private	✓
Activation Target Internal Organization Access	Private	Private	✓
Activation Target Platform	Private	Private	✓

Step 10: Sharing Rules

1. Setup → Sharing Settings → Create New Rule:
 - Example: All Property records owned by Property Manager shared with COO role.
 - Example: All Investment Transaction records shared with Operations Manager role.

Step 11: Login Access Policies

1. Setup → Login Access Policies → Enable:

- Let admins log in as any user for testing (optional).
- **Screenshot:**

The screenshot shows the Salesforce Setup interface with the title 'SETUP Login Access Policies'. Below the title, it says 'Login Access Policies' and 'Control which support organizations your users can grant login access to.' A table titled 'Manage Support Options' lists a single setting: 'Administrators Can Log in as Any User' with the status 'Enabled' and a checked checkbox. The table has columns for 'Support Organization', 'Packages', 'Available to Users', and 'Available to Administrators Only'. Under 'Support Organization', 'Salesforce.com Support' is listed. Under 'Available to Users', there are two radio buttons: one checked (labeled 'A') and one unselected (labeled 'B'). At the bottom of the table are 'Save' and 'Cancel' buttons.

Step 12: Sandbox Usage

- For Phase 2, use **Developer Org directly**.
- For production-grade org: create **sandbox** for testing: Setup → Sandboxes → New Sandbox.

Phase 3: Data Modeling & Relationships – Revaa

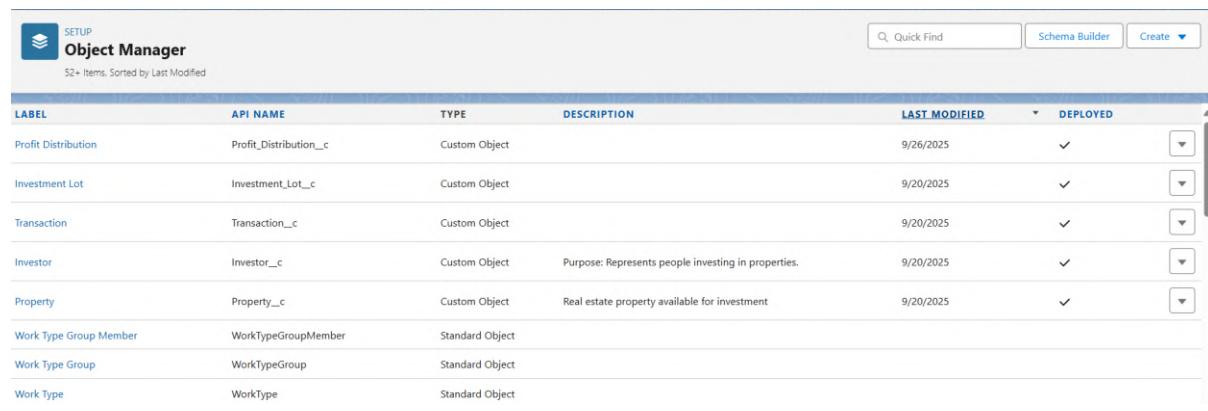
1. Introduction

Phase 3 focuses on defining the Salesforce data model for the Revaa platform. This includes creation of **custom objects, fields, relationships, page layouts, record types, compact layouts, and junction objects**. Proper data modeling ensures the application can **track investors, properties, investments, lots, and profit distributions** in a scalable and maintainable way.

2. Standard & Custom Objects

The following **custom objects** were created:

Object Name	API Name	Purpose
Property	Property__c	Stores details of real estate properties (name, location, value, total & available lots, ROI)
Investor	Investor__c	Stores investor information (name, email, phone, total investment, portfolio value)
Investment Transaction	InvestmentTransaction__c	Acts as a junction object linking Investor → Property. Tracks lots purchased, investment amount, ownership %.
Profit Distribution	ProfitDistribution__c	Tracks profit distributions for each investor per property.
Investment Lot	InvestmentLot__c	Tracks individual lots per property and their allocation to investors.



The screenshot shows the Salesforce Object Manager interface. At the top, there are buttons for SETUP, Object Manager, Quick Find, Schema Builder, and Create. Below the header, it says "52+ Items. Sorted by Last Modified". The main area is a table with columns: LABEL, API NAME, TYPE, DESCRIPTION, LAST MODIFIED, and DEPLOYED. The table lists the following objects:

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Profit Distribution	Profit_Distribution__c	Custom Object		9/26/2025	✓
Investment Lot	Investment_Lot__c	Custom Object		9/20/2025	✓
Transaction	Transaction__c	Custom Object		9/20/2025	✓
Investor	Investor__c	Custom Object	Purpose: Represents people investing in properties.	9/20/2025	✓
Property	Property__c	Custom Object	Real estate property available for investment	9/20/2025	✓
Work Type Group Member	WorkTypeGroupMember	Standard Object			
Work Type Group	WorkTypeGroup	Standard Object			
Work Type	WorkType	Standard Object			

3. Fields

Property_c

- Property Name (Text) – Record Name
- Location (Text)
- Total Value (Currency)
- Total Lots (Number)
- Available Lots (Number, auto-updated by Flow)
- ROI % (Percent)

Investor_c

- Investor Name (Text) – Record Name
- Email (Email)
- Phone (Phone)
- Total Investment (Currency – Roll-up Summary)
- Portfolio Value (Currency – Formula)

InvestmentTransaction_c

- Transaction Number (Auto Number, e.g., TX-{0001}) – Record Name
- Investor (Lookup → Investor_c)
- Property (Lookup → Property_c)
- Lots Purchased (Number)
- Investment Amount (Currency, formula: Lots_Purchased_c * (Property_r.Total_Value_c / Property_r.Total_Lots_c))
- Ownership % (Formula: (Lots_Purchased_c / Property_r.Total_Lots_c) * 100)

ProfitDistribution_c

- Profit Distribution ID (Auto Number) – Record Name
- Property (Lookup → Property_c)
- Investor (Lookup → Investor_c)
- Profit Amount (Currency)
- Distribution Date (Date)

InvestmentLot_c

- Lot Name (Text/Auto Number) – Record Name
- Property (Lookup → Property__c)
- Investor (Lookup → Investor__c)
- Lot Status (Picklist: Available, Allocated, Sold)
- Lot Value (Currency)
- Purchase Date (Date)

SETUP > OBJECT MANAGER Property					
Details	Fields & Relationships				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Available Lots	Available_Lots__c	Formula (Number)		
Lightning Record Pages	Created By	CreatedById	Lookup(User)		
Buttons, Links, and Actions	Currency	CurrencyIsoCode	Picklist		
Compact Layouts	Last Modified By	LastModifiedById	Lookup(User)		
Field Sets	Location	Location__c	Text(255)		
Object Limits	Owner	OwnerId	Lookup(User,Group)	✓	
Record Types	Price	Price__c	Currency(18, 0)		
Related Lookup Filters	Property Description	Property_Description__c	Long Text Area(32768)		
Restriction Rules	Property Name	Name	Text(80)	✓	
Scoping Rules	Record Type	RecordTypeId	Record Type	✓	
Object Access					
Triggers					

SETUP > OBJECT MANAGER Transaction					
Details	Fields & Relationships				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Amount	Amount__c	Currency(18, 0)		
Lightning Record Pages	Created By	CreatedById	Lookup(User)		
Buttons, Links, and Actions	Currency	CurrencyIsoCode	Picklist		
Compact Layouts	Investment Amount	Investment_Amount__c	Formula (Number)		
Field Sets	Investor	Investor__c	Master-Detail(Investor)	✓	
Object Limits	Investor Name	Investor_Name__c	Lookup(Investor)	✓	
Record Types	Last Modified By	LastModifiedById	Lookup(User)		
Related Lookup Filters	Lot	Lot__c	Lookup(Investment Lot)	✓	
Search Layouts	Lots Purchased	Lots_Purchased__c	Number(18, 0)		
List View Button Layout					
Restriction Rules					
Scoping Rules					

SETUP > OBJECT MANAGER
Investment Lot

Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules

Fields & Relationships

8 Items, Sorted by Field Label

Quick Find New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Currency	CurrencyIsoCode	Picklist		
Last Modified By	LastModifiedById	Lookup(User)		
Lot Number	Name	Auto Number		
Owned By	Owned_By__c	Lookup(Investor)		
Price per Lot	Price_per_Lot__c	Currency(18, 0)		
Property	Property__c	Master-Detail(Property)		
Status	Status__c	Picklist		

SETUP > OBJECT MANAGER
Investor

Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules

Fields & Relationships

11 Items, Sorted by Field Label

Quick Find New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Long Text Area(32768)		
Contact Number	Contact_Number__c	Phone		
Created By	CreatedById	Lookup(User)		
Currency	CurrencyIsoCode	Picklist		
Email	Email__c	Email (Unique)		
Investor Name	Name	Text(80)		
Last Modified By	LastModifiedById	Lookup(User)		
Name	Name__c	Text(30)		
Owner	OwnerId	Lookup(User,Group)		
Portfolio Value	Portfolio_Value__c	Currency(18, 0)		

SETUP > OBJECT MANAGER
Profit Distribution

Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules

Fields & Relationships

9 Items, Sorted by Field Label

Quick Find New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Currency	CurrencyIsoCode	Picklist		
Distribution Date	Distribution_Date__c	Date		
Investor	Investor__c	Lookup(Investor)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		
Profit Amount	Profit_Amount__c	Currency(18, 0)		
Profit Distribution Name	Name	Text(80)		
Property	Property__c	Lookup(Property)		

MyASUS

4. Record Types

Record Types Created for Property Object :

- Residential
- Commercial
- Land Plot

Record Types			
3 Items, Sorted by Record Type Label			
RECORD TYPE LABEL	DESCRIPTION	ACTIVE	MODIFIED BY
Commercial		✓	ANIKET KAKADE, 9/28/2025, 1:17 AM
Land Plot		✓	ANIKET KAKADE, 9/28/2025, 1:21 AM
Residential		✓	ANIKET KAKADE, 9/28/2025, 1:16 AM

5. Page Layouts

- Each object has a **custom page layout** showing key fields.
- Related lists for junctions:
 - **Property** → Related List: Investment Transactions, Investment Lots, Profit Distributions
 - **Investor** → Related List: Investment Transactions, Investment Lots, Profit Distributions
 - **Investment Transaction** → Related List: (not needed)
 - **Profit Distribution** → Related List: (not needed)

6. Compact Layouts

Fields for Highlights Panel (Lightning & Mobile):

Object	Compact Layout Name	Fields
Property__c	Property_Compact	Property Name, Location, Available Lots, ROI %
Investor__c	Investor_Compact	Investor Name, Total Investment, Portfolio Value, Email
InvestmentTransaction__c	InvestmentTransaction_Compact	Transaction Number, Investor, Property, Lots Purchased
ProfitDistribution__c	ProfitDistribution_Compact	Profit Distribution ID, Profit Amount, Distribution Date, Investor
InvestmentLot__c	InvestmentLot_Compact	Lot Name, Property, Investor, Lot Status

Screenshot :

The screenshot shows the 'Compact Layout Detail' page for the 'Property_Compact' layout. The page includes fields for Label (Property Compact), API Name (Property_Compact), Included Fields (Property Name, Location, Available Lots, ROI %), Created By (ANIKET KAKADE, 9/28/2025, 2:09 AM), and Modified By (ANIKET KAKADE, 9/28/2025, 2:09 AM). Action buttons include Edit, Clone, Delete, and Compact Layout Assignment.

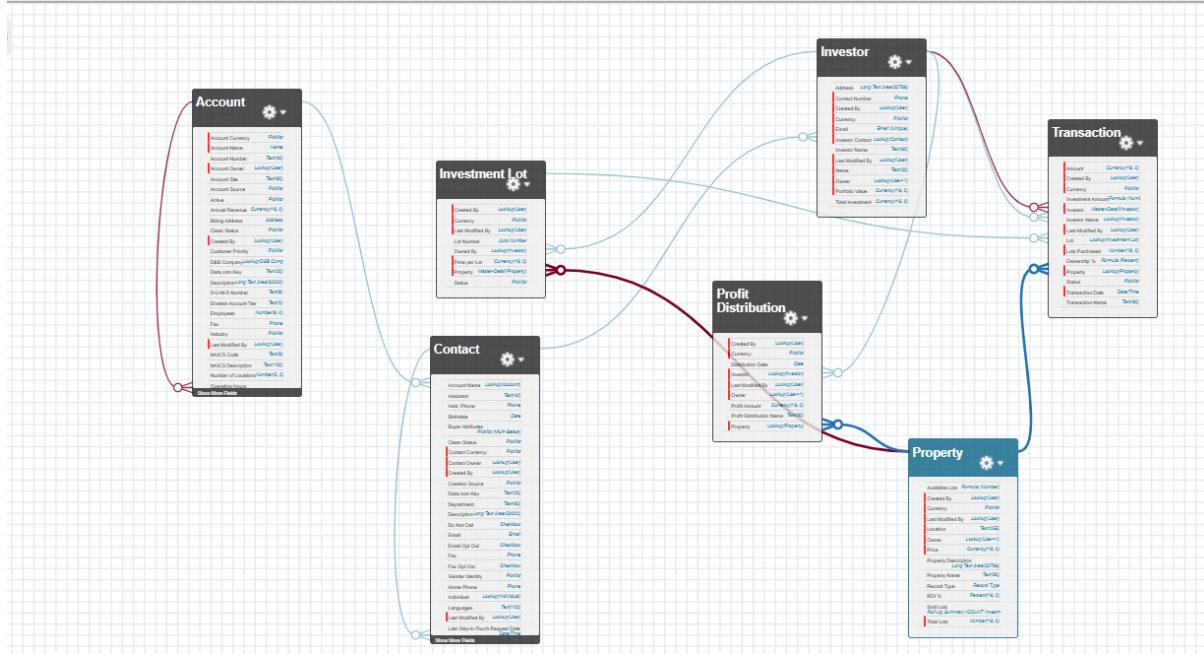
The screenshot shows the 'Compact Layout Detail' page for the 'Investor_Compact' layout. The page includes fields for Label (Investor Compact), API Name (Investor_Compact), Included Fields (Investor Name, Total Investment, Portfolio Value, Contact Number), Created By (ANIKET KAKADE, 9/28/2025, 2:10 AM), and Modified By (ANIKET KAKADE, 9/28/2025, 2:10 AM). Action buttons include Edit, Clone, Delete, and Compact Layout Assignment.

Transaction Compact Layout
Transactions Compact
[« Back to Transaction](#)

Compact Layout Detail		Edit Clone Delete Compact Layout Assignment		Object Name	Transaction
Label	Transactions Compact	Created By	ANIKET KAKADE, 9/28/2025, 2:12 AM		
API Name	Transactions_Compact				
Included Fields	Transaction Name Investor Property Lots Purchased				
Created By	ANIKET KAKADE, 9/28/2025, 2:12 AM				
		Edit	Clone	Delete	Compact Layout Assignment

7. Schema Builder

- Used to **visually confirm relationships** and object fields.
- Objects included: Property, Investor, Investment Transaction, Profit Distribution, Investment Lot.
- Confirmed:
 - InvestmentTransaction__c connects **Property__c ↔ Investor__c** (junction)
 - ProfitDistribution__c links **Property__c ↔ Investor__c**
 - InvestmentLot__c links **Property__c ↔ Investor__c**



8. Lookup vs Master-Detail vs Hierarchical Relationships

Relationship Type	Usage in Revaa	Notes
Lookup	InvestmentTransaction → Investor & Property	Flexible, can exist without parent
Master-Detail	Optional: convert InvestmentTransaction lookups to MD	Enforces parent-child data integrity, sharing inheritance
Hierarchical	Not used	Typically only for User object manager chains

9. Junction Objects

- InvestmentTransaction__c serves as **junction object** connecting Investor ↔ Property (many-to-many).
- Optional: convert both lookup fields to Master-Detail for stricter Salesforce standard.
- Related lists on Property & Investor pages allow easy navigation to all transactions.
-

10. Verification Checklist

- All custom objects created with correct API names
- Fields created with correct data types and formulas
- Related lists added for junction objects
- Compact layouts created and assigned
- Schema Builder shows correct relationships
- Ownership % formula verified: $(\text{Lots_Purchased_c} / \text{Property_r.Total_Lots_c}) * 100$
- InvestmentTransaction__c correctly functions as junction object
- Screenshots captured for documentation

Phase 4: Process Automation (Admin) – Revaa

This phase focused on implementing **automation for transactions, portfolios, withdrawals, and reporting** in Revaa. We combined **Validation Rules, Workflow Rules, Process Builder, Approval Process, Record-Triggered Flows, and Scheduled Flows**.

◆ 4.1 Validation Rules

We created multiple validation rules to ensure **data integrity** across objects.

4.1.1 Ownership Percentage Required

- **Object:** Transaction__c
- **Rule Name:** Ownership_Percentage_Required
- **Error Condition Formula:**

ISBLANK(Ownership_Percent__c)

- **Error Message:** Ownership Percent must be entered for every transaction.

The screenshot shows the 'Validation Rules' section for the 'Transaction' object. It lists three rules: 'Prevent_Negative_Lot_Purchase', 'Prevent_Over_Allocation', and 'Prevent_Over_Purchase'. The 'Prevent_Negative_Lot_Purchase' rule has an error message 'You must Purchase atleast 1 Lot' and is active. The 'Prevent_Over_Allocation' rule has an error message 'Cannot allocate lots. Total lots purchased would exceed property total.' and is active. The 'Prevent_Over_Purchase' rule has an error message 'You cannot purchase more lots than are available for this Property.' and is active. The 'New' button is visible at the top right of the table.

The screenshot shows the 'Transaction Validation Rule' detail page. It displays the 'Validation Rule Detail' section with the following information:

- Rule Name: Prevent_Negative_Lot_Purchase
- Error Condition Formula: Lots_Purchased__c <= 0
- Error Message: You must Purchase atleast 1 Lot
- Description: No Buyer can buy less than 1 Lot
- Created By: ANIKET KAKADE, 9/28/2025, 4:03 AM
- Active: ✓
- Error Location: Top of Page
- Modified By: ANIKET KAKADE, 9/28/2025, 4:03 AM

The 'Edit' and 'Clone' buttons are located at the bottom of the detail section.

4.1.2 Ownership Percent Greater Than Zero

- **Object:** Transaction__c
- **Rule Name:** Ownership_Consistency
- **Error Condition Formula:**

Ownership_Percent__c <= 0

- **Error Message:** Ownership Percent must be greater than 0.

The screenshot shows the 'Transaction Validation Rule' page in the Salesforce Object Manager. The rule is named 'Prevent_Over_Allocation' and has the formula `(Property__r.Total_Lots_Purchased__c + Lots_Purchased__c) > Property__r.Total_Lots__c`. The error message is 'Cannot allocate lots. Total lots purchased would exceed property total.' The rule is active and was created by ANIKET KAKADE on 9/28/2025 at 4:21 AM.

4.1.3 Profit Amount Required for Profit Distribution

- **Object:** Property__c
- **Rule Name:** Profit_Amount_Required
- **Error Condition Formula:**

ISPICKVAL(Transaction_Type__c, "Profit Distribution") && ISBLANK(Profit_Amount__c)

- **Error Message:** Profit Amount must be entered before distributing profit.

The screenshot shows the 'Transaction Validation Rule' page in the Salesforce Object Manager. The rule is named 'Prevent_Over_Purchase' and has the formula `Lots_Purchased__c > Property__r.Available_Lots__c`. The error message is 'You cannot purchase more lots than are available for this Property.' The rule is active and was created by ANIKET KAKADE on 9/28/2025 at 4:00 AM.

◆ 4.2 Workflow Rules

SETUP **Workflow Rules**

All Workflow Rules [Help for this Page](#)

Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. [Tell Me More](#) | [Migrate your workflow rules to flows](#)

Configure your organization's workflow by creating workflow rules. Each workflow rule consists of:

- Criteria that cause the workflow rule to run.
- Immediate actions that execute when a record matches the criteria. For example, Salesforce can automatically send an email that notifies the account team when a new high-value opportunity is created.
- Time-dependent actions that queue when a record matches the criteria, and execute according to time triggers. For example, Salesforce can automatically send an email reminder to the account team if a high-value opportunity is still open ten days before the close date.

Quick Tips

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

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

[A](#) [B](#) [C](#) [D](#) [E](#) [F](#) [G](#) [H](#) [I](#) [J](#) [K](#) [L](#) [M](#) [N](#) [O](#) [P](#) [Q](#) [R](#) [S](#) [T](#) [U](#) [V](#) [W](#) [X](#) [Y](#) [Z](#) [Other](#) [All](#)

Action	Rule Name	Description	Object	Active
Edit Del Deactivate	Notify Admin on Transaction		Transaction	<input checked="" type="checkbox"/>
Edit Del Deactivate	Update Transaction Status		Transaction	<input checked="" type="checkbox"/>

SETUP **Workflow Rules**

Workflow Rule **Notify Admin on Transaction** [Help for this Page](#)

Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. [Tell Me More](#) | [Migrate your workflow rules to flows](#)

Workflow Rule Detail

Rule Name	Description	Object	Evaluation Criteria
Notify Admin on Transaction	Active	Transaction	Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
	Description		
	Rule Criteria	TRUE	
	Created By	ANIKET KAKADE 9/28/2025, 4:50 AM	Modified By ANIKET KAKADE 9/28/2025, 4:53 AM

Workflow Actions [Edit](#)

Immediate Workflow Actions

Type	Description
Email Alert	Email will be sent to admin on every Transaction.

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

SETUP **Workflow Rules**

Workflow Rule **Update Transaction Status** [Help for this Page](#)

Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. [Tell Me More](#) | [Migrate your workflow rules to flows](#)

Workflow Rule Detail

Rule Name	Description	Object	Evaluation Criteria
Update Transaction Status	Active	Transaction	Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
	Description		
	Rule Criteria	Property: Sold_Lots > 0	
	Created By	ANIKET KAKADE 9/28/2025, 5:01 AM	Modified By ANIKET KAKADE 9/28/2025, 11:40 AM

Workflow Actions [Edit](#)

Immediate Workflow Actions

Type	Description
Field Update	Update Transaction Status

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

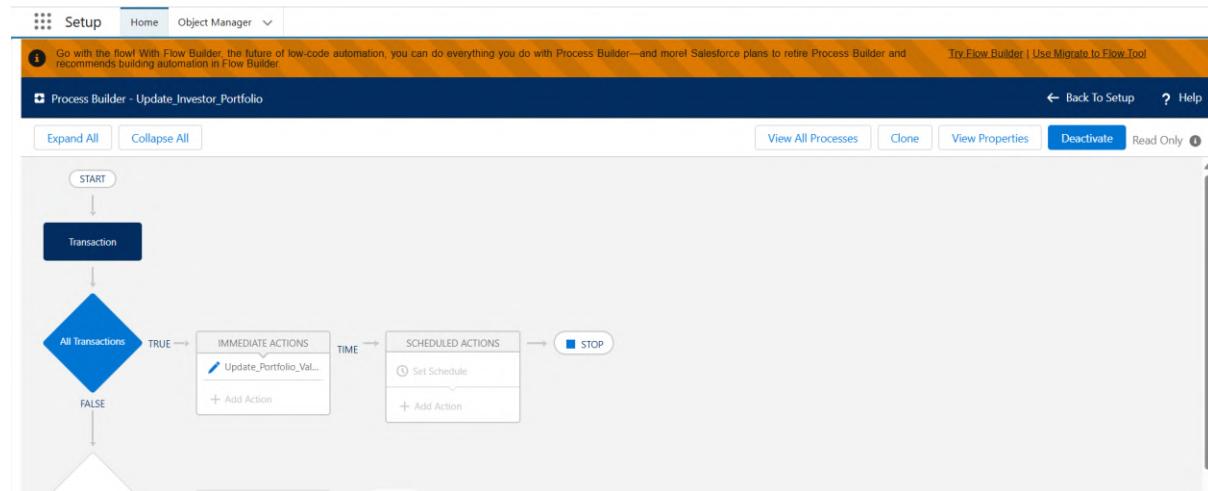
◆ 4.3 Process Builder

4.3.1 Update Investor Portfolio Value

- **Process Name:** Update_Portfolio_Value
- **Object:** Transaction__c
- **Trigger:** When record is created or edited
- **Immediate Action:** Update Investor__c.Total_Investment__c

Formula used:

[Investor__c].Total_Investment__c + [Transaction__c].Amount__c



◆ 4.4 Approval Process

4.4.1 Withdrawal Approval Workflow

- **Object:** Transaction__c
- **Process Name:** Withdrawal_Approval
- **Entry Criteria:**

Transaction_Type__c = 'Withdrawal'

- **Approvers:** CFO Role
- **Email Alerts:**
 - **Withdrawal_Approval_Template** (sent when approved)
 - **Withdrawal_Rejection_Template** (sent when rejected)

SETUP Approval Processes

Approval Processes
Transaction: Withdrawal Approval Process
[Help for this Page](#)

[Back to Approval Process List](#)

Process Definition Detail	
Process Name	Withdrawal Approval Process
Unique Name	Withdrawal_Approval_Process
Description	Next Automated Approver Determined By
Entry Criteria	Transaction: Withdrawal Status EQUALS Requested
Record Editability	Administrator OR Current Approver
Approval Assignment Email Template	Withdrawal Approval Request
Initial Submitters	Investor Owner, Role: System Administrator, Role: Operations Manager, Role: Property Manager, Role: CFO
Created By	ANIKET KAKADE 9/28/2025, 8:13 AM
Modified By	ANIKET KAKADE 9/28/2025, 8:32 AM

Initial Submission Actions [Add Existing](#) [Add New](#)

Action Type	Description
Record Lock	Lock the record from being edited

Initial Submission Actions [Add Existing](#) [Add New](#)

Action Type	Description
Record Lock	Lock the record from being edited

Approval Steps [Add Existing](#) [Add New](#)

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior						
Hide Actions Edit												
Approval Actions Add Existing Add New <table border="1"> <thead> <tr> <th>Action</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Edit Remove</td> <td>Email Alert</td> <td>Withdrawal Approval Notification</td> </tr> </tbody> </table>							Action	Type	Description	Edit Remove	Email Alert	Withdrawal Approval Notification
Action	Type	Description										
Edit Remove	Email Alert	Withdrawal Approval Notification										
Rejection Actions Add Existing Add New <table border="1"> <thead> <tr> <th>Action</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Edit Remove</td> <td>Email Alert</td> <td>Withdrawal Rejection Action</td> </tr> </tbody> </table>							Action	Type	Description	Edit Remove	Email Alert	Withdrawal Rejection Action
Action	Type	Description										
Edit Remove	Email Alert	Withdrawal Rejection Action										

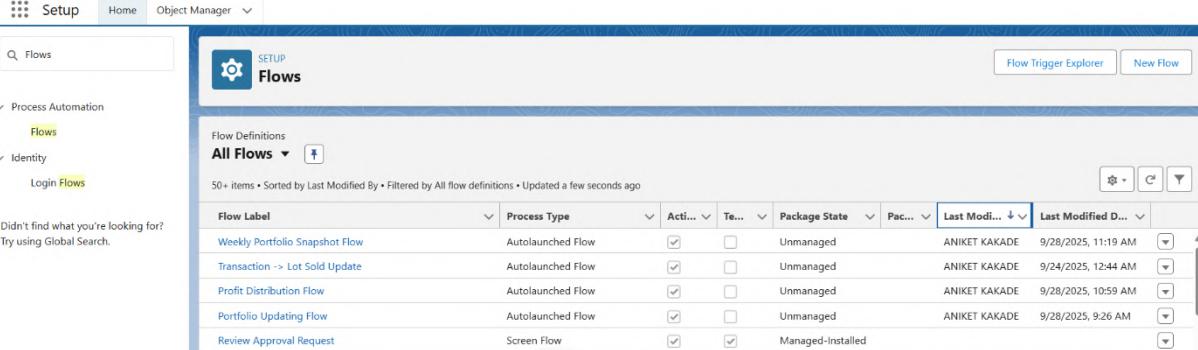
Final Approval Actions [Add Existing](#) [Add New](#)

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

◆ 4.5 Record-Triggered Flow

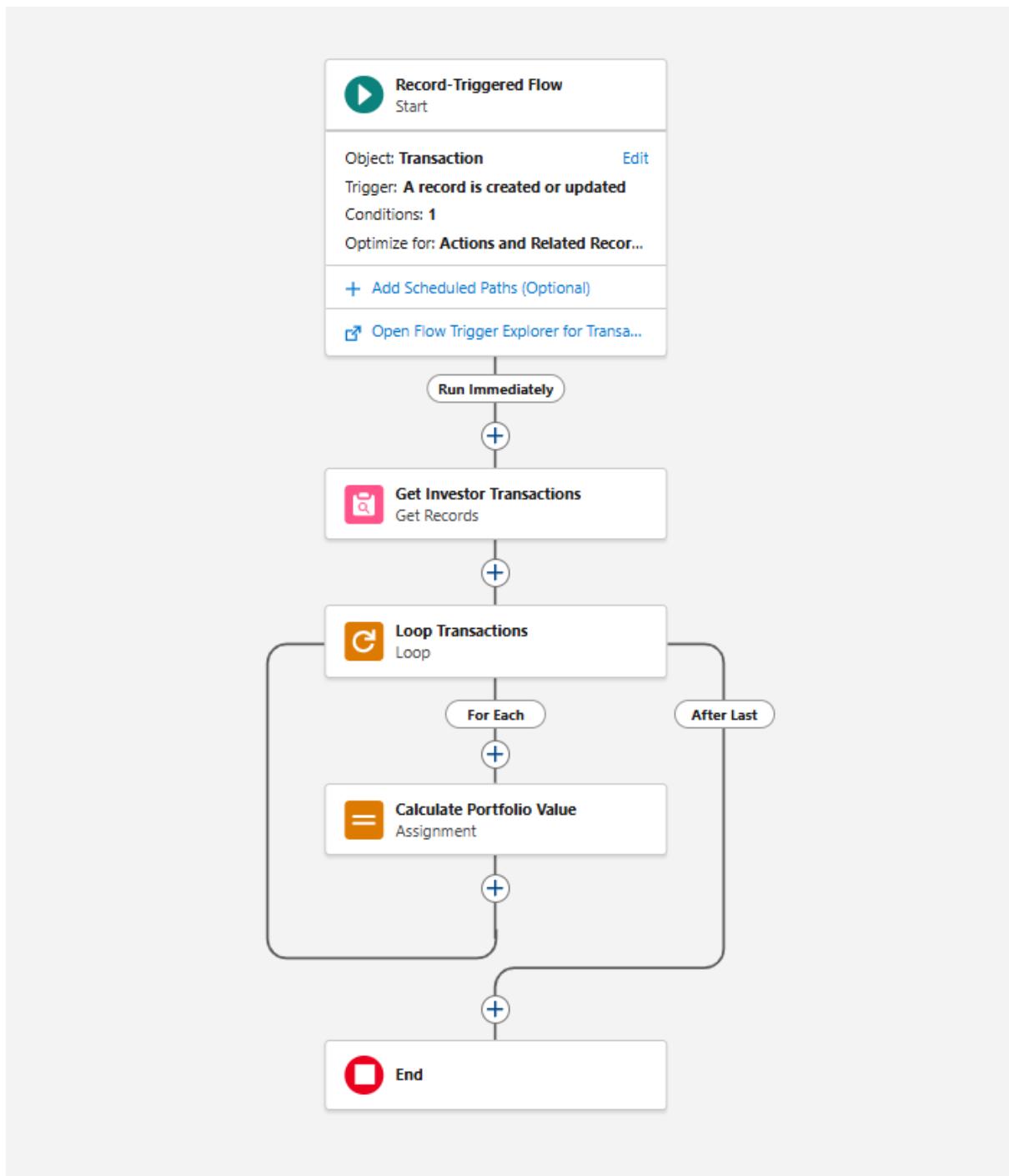
4.5.1 Profit Distribution Flow

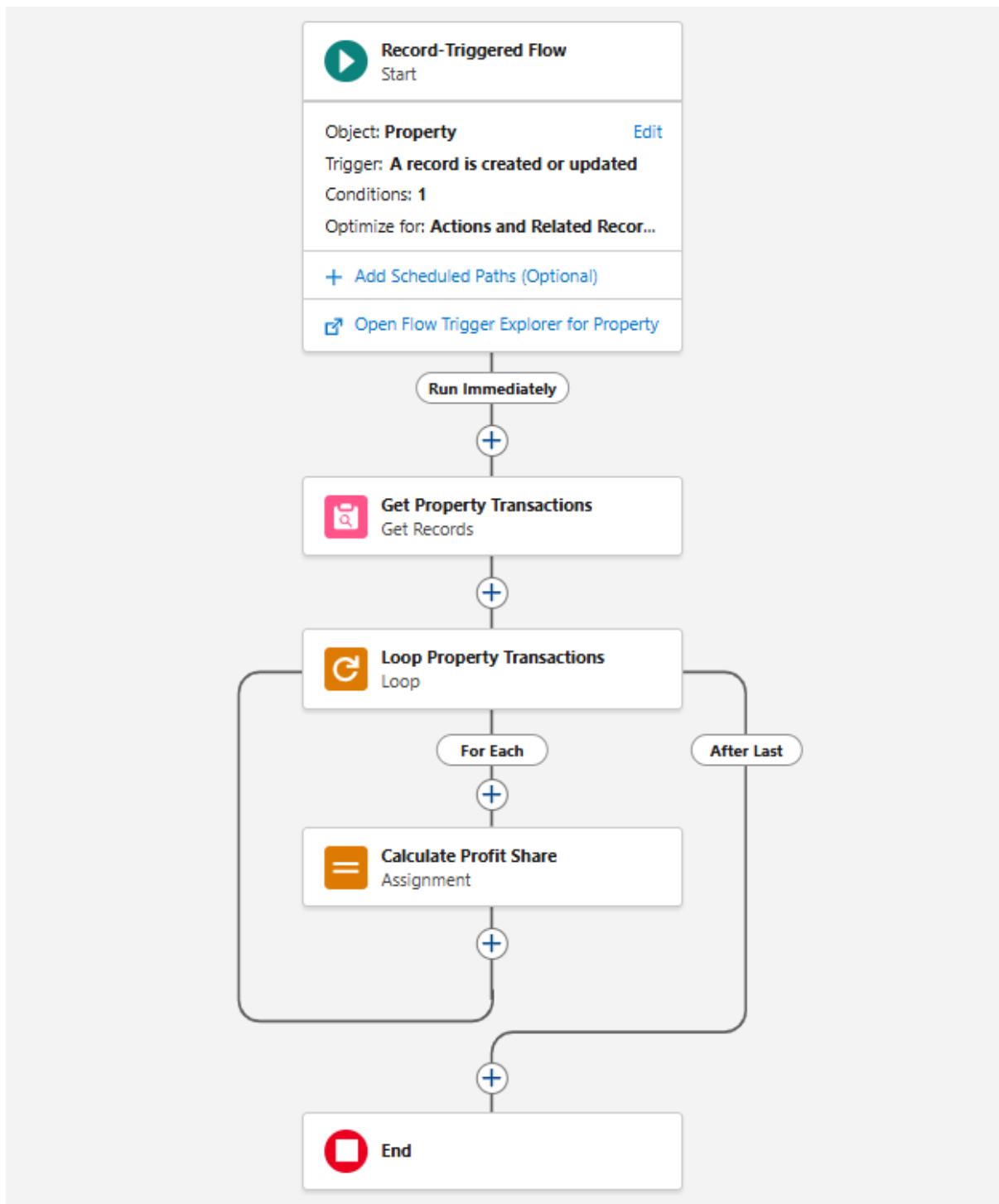
- **Flow Name:** Profit_Distribution_Flow
- **Trigger:** On Property__c (when Profit_Amount__c is updated)
- **Steps:**
 1. **Get Records:** Fetch all Transaction__c linked to Property.
 2. **Loop:** Loop over each Transaction.
 3. **Assignment:**
 4. $\{ !Loop_Transactions.Profit_Share_c \} =$
 $\{ !Loop_Transactions.Ownership_Percent_c \}$
 $\{ \$Record.Profit_Amount_c \} / 100$ *
 5. **Update Records:** Update each Transaction with Profit_Share__c.

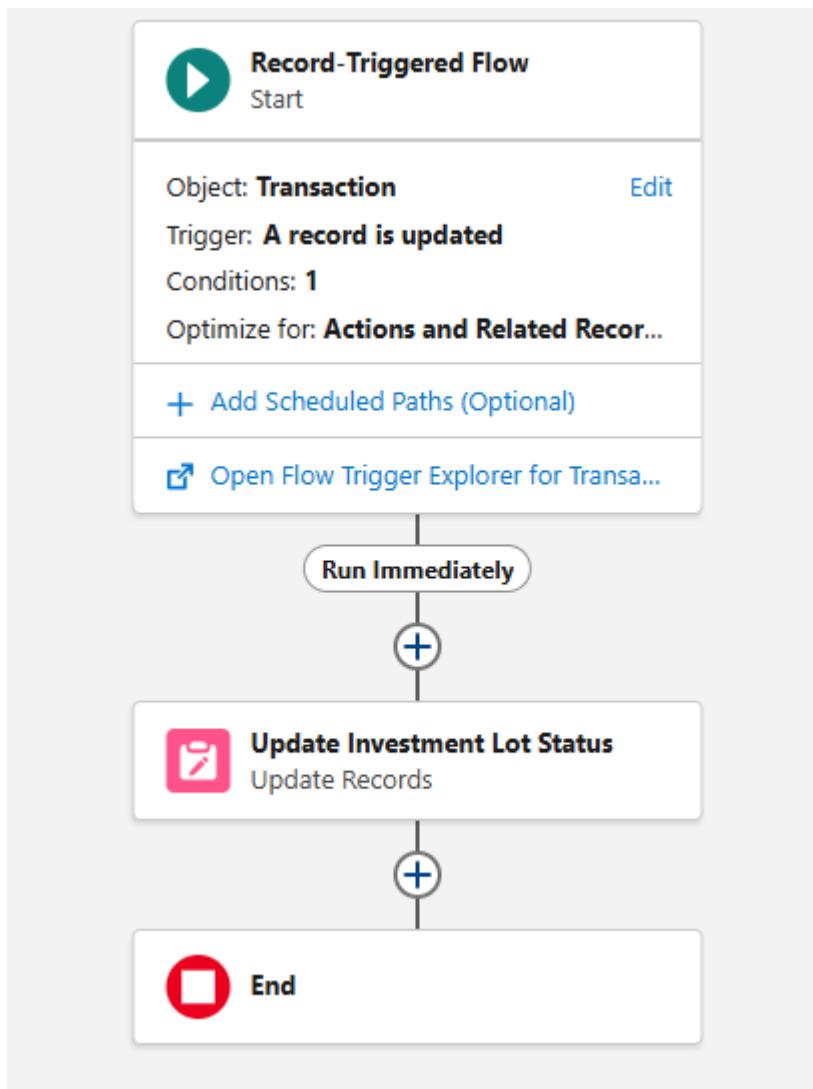


The screenshot shows the Salesforce Flows page under the Setup menu. The sidebar on the left includes links for Process Automation (Flows), Identity, and Login Flows. A search bar at the top says "Q_ Flows". The main area displays a table of flow definitions:

Flow Label	Process Type	Acti...	Te...	Package State	Pac...	Last Modifi...	Last Modified D...
Weekly Portfolio Snapshot Flow	Autolaunched Flow	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Unmanaged	ANIKET KAKADE	9/28/2025, 11:19 AM	9/28/2025, 11:19 AM
Transaction -> Lot Sold Update	Autolaunched Flow	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Unmanaged	ANIKET KAKADE	9/24/2025, 12:44 AM	9/24/2025, 12:44 AM
Profit Distribution Flow	Autolaunched Flow	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Unmanaged	ANIKET KAKADE	9/28/2025, 10:59 AM	9/28/2025, 10:59 AM
Portfolio Updating Flow	Autolaunched Flow	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Unmanaged	ANIKET KAKADE	9/28/2025, 9:26 AM	9/28/2025, 9:26 AM
Review Approval Request	Screen Flow	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Managed-Installed			





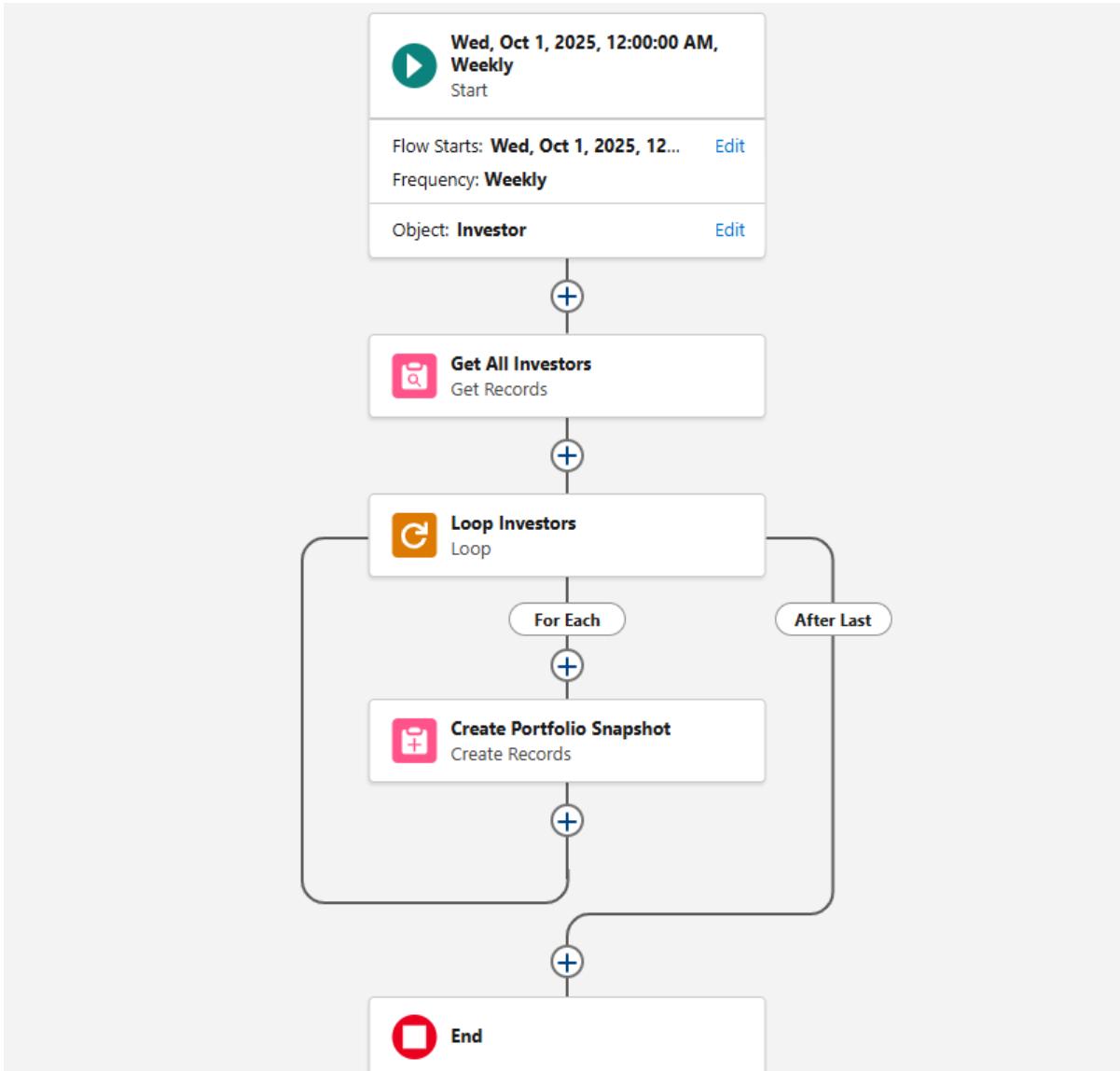


◆ 4.6 Scheduled Flow

4.6.1 Weekly Portfolio Snapshot

- **Flow Name:** Weekly_Portfolio_Snapshot_Flow
- **Trigger:** Scheduled – last day of every month, 11:59 PM
- **Steps:**
 1. **Get Records:** Get all Investor__c.
 2. **Loop:** For each Investor.
 3. **Create Record:** Portfolio_Snapshot__c with:

- **Investor__c** = {!Loop_Investors.Id}
- **Snapshot_Date__c** = \$Flow.CurrentDate
- **Total_Value__c** = {!Loop_Investors.Total_Investment__c}



◆ 4.7 Email Alerts

Email Templates Created

1. Investor_Welcome_Template

- Sent when a new Investor is created.

2. Withdrawal_Approval_Template

- Subject: "Your Withdrawal Request has been Approved"
- Body: Includes Transaction amount, Property name, Approval details.

3. Withdrawal_Rejection_Template

- Subject: "Your Withdrawal Request has been Rejected"
- Body: Includes Transaction details and rejection reason.

The screenshot shows the 'Email Alerts' section of a CRM interface. At the top, there's a 'SETUP' button and a gear icon. Below that, the title 'Email Alerts' is displayed. A sub-header 'All Email Alerts' is shown, along with a 'Help for this Page' link. Underneath, there's a note about using email alerts for automation. A 'View' dropdown is set to 'All Email Alerts' with a 'Create New View' option. The main area is a table with the following data:

Action	Description	Email Template Name	Object	Last Modified Date
Edit Del	Email will be sent to admin on every Transaction.	Transaction Alert for Admin	Transaction	9/28/2025
Edit Del	Withdrawal Approval Notification	Withdrawal_Approved_Notification	Transaction	9/28/2025
Edit Del	Withdrawal Rejection Action	Withdrawal_Rejected_Email	Transaction	9/28/2025

At the bottom of the table, there are links for navigating through the alphabet (A-Z) and a 'All' link.

✓ Phase 4 Outcomes

- Transactions validated for ownership consistency.
- Investors welcomed via automated email.
- Portfolio values updated dynamically after every transaction.
- Withdrawal requests routed through CFO approval with email notifications.
- Profit distributions automated across investors based on ownership %.
- Monthly portfolio snapshots created automatically for reporting.
- End-to-end automation backbone implemented for **Reva**a.

Phase 5: Apex Programming (Developer)

Objective: Implement backend logic using Apex classes and triggers to automate portfolio updates, investment lot management, and to enhance the functionality of Revaa beyond standard admin automation.

1 Classes & Objects

Custom Objects used in Apex:

- **Property__c** – Represents real estate properties.
- **Investor__c** – Represents investors.
- **Transaction__c** – Tracks investments made by investors.
- **Investment_Lot__c** – Tracks individual lots purchased per transaction.

Apex Classes created:

1. **TransactionHandler** – Contains logic to update Investor__c.Total_Investment__c when a new transaction is created.
2. **InvestmentLotHandler** – Ensures each lot is linked to the investor and property and prevents duplicate lot allocation.

Apex Classes

[Help for this Page](#)

Apex Code is an object oriented programming language that allows developers to develop on-demand business applications on the Lightning Platform.

Percent of Apex Used: 0.07%
You are currently using 4,250 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.

[Estimate your organization's code coverage](#)

[Compile all classes](#)

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

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other

Action	Name	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit Del Security	PortfolioCalculator		64.0	Active	381	ANIKET KAKADE, 9/28/2025, 12:07 PM	<input type="checkbox"/>
Edit Del Security	ProfitDistributionHandler		64.0	Active	990	ANIKET KAKADE, 9/28/2025, 12:58 PM	<input type="checkbox"/>
Edit Del	RevaaTriggerTest		64.0	Active	1,535	ANIKET KAKADE, 9/28/2025, 1:29 PM	<input type="checkbox"/>

The screenshot shows the Apex Classes page with the following details:

- Apex Class:** PortfolioCalculator
- Name:** PortfolioCalculator
- Namespace Prefix:** (None)
- Created By:** ANIKET KAKADE , 9/28/2025, 12:07 PM
- Status:** Active
- Code Coverage:** 0% (0/6)
- Last Modified By:** ANIKET KAKADE , 9/28/2025, 12:07 PM

Class Body:

```

1 public class PortfolioCalculator {
2
3     // Method to calculate portfolio value for an investor
4     public static Decimal calculatePortfolio(Id investorId) {
5         Decimal totalValue = 0;
6         List<Transaction__c> txns = [SELECT Investment_Amount__c FROM Transaction__c WHERE Investor__c = :investorId];
7         for(Transaction__c t : txns){
8             totalValue += t.Investment_Amount__c;
9         }
10    return totalValue;
11 }
12
13 }
```

The screenshot shows the Apex Classes page with the following details:

- Namespace Prefix:** (None)
- Created By:** ANIKET KAKADE , 9/28/2025, 12:58 PM
- Status:** Active
- Code Coverage:** 0% (0/14)
- Last Modified By:** ANIKET KAKADE , 9/28/2025, 12:58 PM

Class Body:

```

1 public class ProfitDistributionHandler {
2
3     // Distribute profit for a property
4     public static void distributeProfit(Id propertyId, Decimal totalProfit) {
5
6         // Get all transactions for the property
7         List<Transaction__c> transactions = [
8             SELECT Id, Investor__c, Ownership__c
9             FROM Transaction__c
10            WHERE Property__c = :propertyId
11        ];
12
13        List<Profit_Distribution__c> profitsToInsert = new List<Profit_Distribution__c>();
14
15        for(Transaction__c t : transactions){
16            if(t.Investor__c != null && t.Ownership__c != null){
17                Decimal investorProfit = (t.Ownership__c / 100) * totalProfit;
18
19                profitsToInsert.add(new Profit_Distribution__c(
20                    Property__c = propertyId,
21                    Investor__c = t.Investor__c,
22                    Profit_Amount__c = investorProfit,
23                    Distribution_Date__c = Date.today()
24                ));
25            }
26        }
27
28        if(!profitsToInsert.isEmpty()){
29            insert profitsToInsert;
30        }
31    }
```

2 Apex Triggers

Trigger 1: TransactionTrigger

- Object:** Transaction__c
- Trigger Events:** after insert, after update
- Purpose:** Updates investor portfolio value based on the lots purchased and property price.

Logic Overview:

1. Collect Investor_c IDs from transactions.
2. Query investors in bulk.
3. Calculate transaction amount:
4. Transaction Amount = Lots_Purchased_c * (Property_r.Price_c / Property_r.Total_Lots_c)
5. Add transaction amount to investor's total investment.
6. Update all investors in bulk.

Trigger Code:

```

trigger TransactionTrigger on Transaction__c (after insert, after update) {
    Set<Id> investorIds = new Set<Id>();
    for(Transaction__c t : Trigger.new){
        if(t.Investor__c != null) investorIds.add(t.Investor__c);
    }

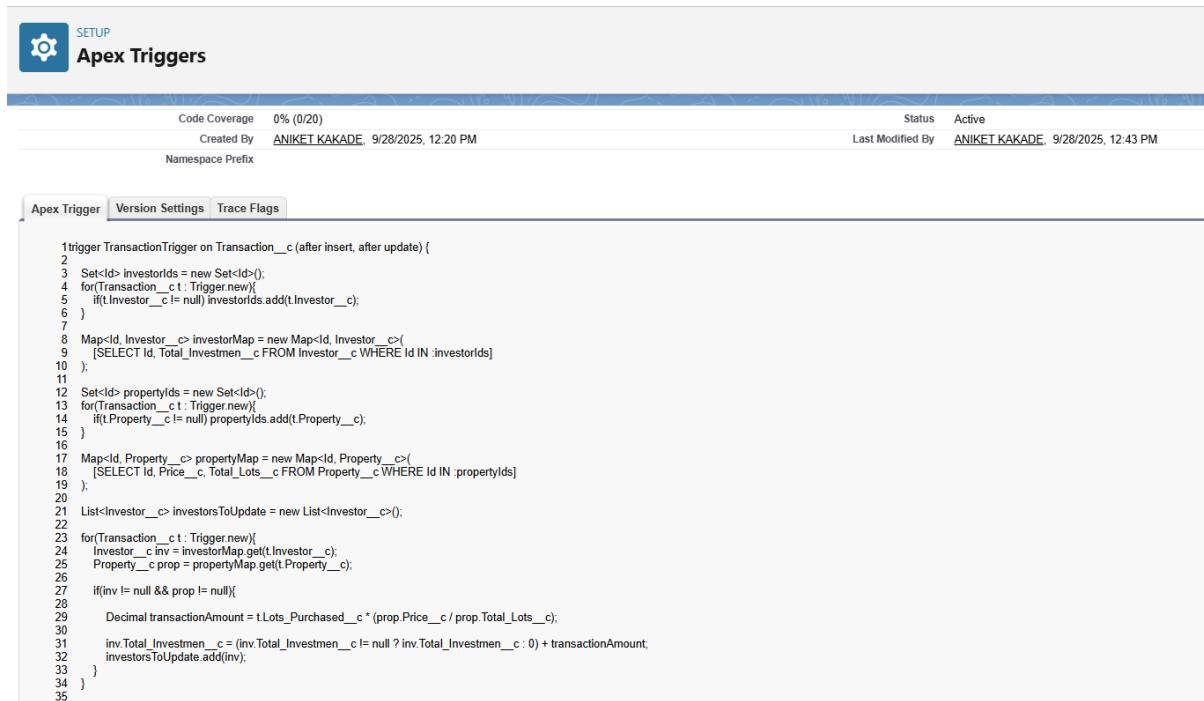
    Map<Id, Investor__c> investorMap = new Map<Id, Investor__c>(
        [SELECT Id, Total_Investment__c FROM Investor__c WHERE Id IN :investorIds]
    );

    List<Investor__c> investorsToUpdate = new List<Investor__c>();
    for(Transaction__c t : Trigger.new){
        Investor__c inv = investorMap.get(t.Investor__c);
        if(inv != null){
            Decimal transactionAmount = t.Lots_Purchased__c * (t.Property__r.Price__c /
t.Property__r.Total_Lots__c);
            inv.Total_Investment__c = (inv.Total_Investment__c != null ?
inv.Total_Investment__c : 0) + transactionAmount;
            investorsToUpdate.add(inv);
        }
    }

    if(!investorsToUpdate.isEmpty()) update investorsToUpdate;
}

```

}



The screenshot shows the Apex Triggers setup page in Salesforce. The trigger is named 'TransactionTrigger' and is defined on the 'Transaction__c' object. The trigger code is as follows:

```
trigger TransactionTrigger on Transaction__c (after insert, after update) {
    Set<Id> investorIds = new Set<Id>();
    for(Transaction__c t : Trigger.new){
        if(t.Investor__c != null) investorIds.add(t.Investor__c);
    }
    Map<Id, Investor__c> investorMap = new Map<Id, Investor__c>(
        [SELECT Id, Total_Investment__c FROM Investor__c WHERE Id IN :investorIds]
    );
    Set<Id> propertyIds = new Set<Id>();
    for(Transaction__c t : Trigger.new){
        if(t.Property__c != null) propertyIds.add(t.Property__c);
    }
    Map<Id, Property__c> propertyMap = new Map<Id, Property__c>(
        [SELECT Id, Price__c, Total_Lots__c FROM Property__c WHERE Id IN :propertyIds]
    );
    List<Investor__c> investorsToUpdate = new List<Investor__c>();
    for(Transaction__c t : Trigger.new){
        Investor__c inv = investorMap.get(t.Investor__c);
        Property__c prop = propertyMap.get(t.Property__c);
        if(inv != null && prop != null){
            Decimal transactionAmount = t.Lots_Purchased__c * (prop.Price__c / prop.Total_Lots__c);
            inv.Total_Investment__c = (inv.Total_Investment__c != null ? inv.Total_Investment__c : 0) + transactionAmount;
            investorsToUpdate.add(inv);
        }
    }
}
```

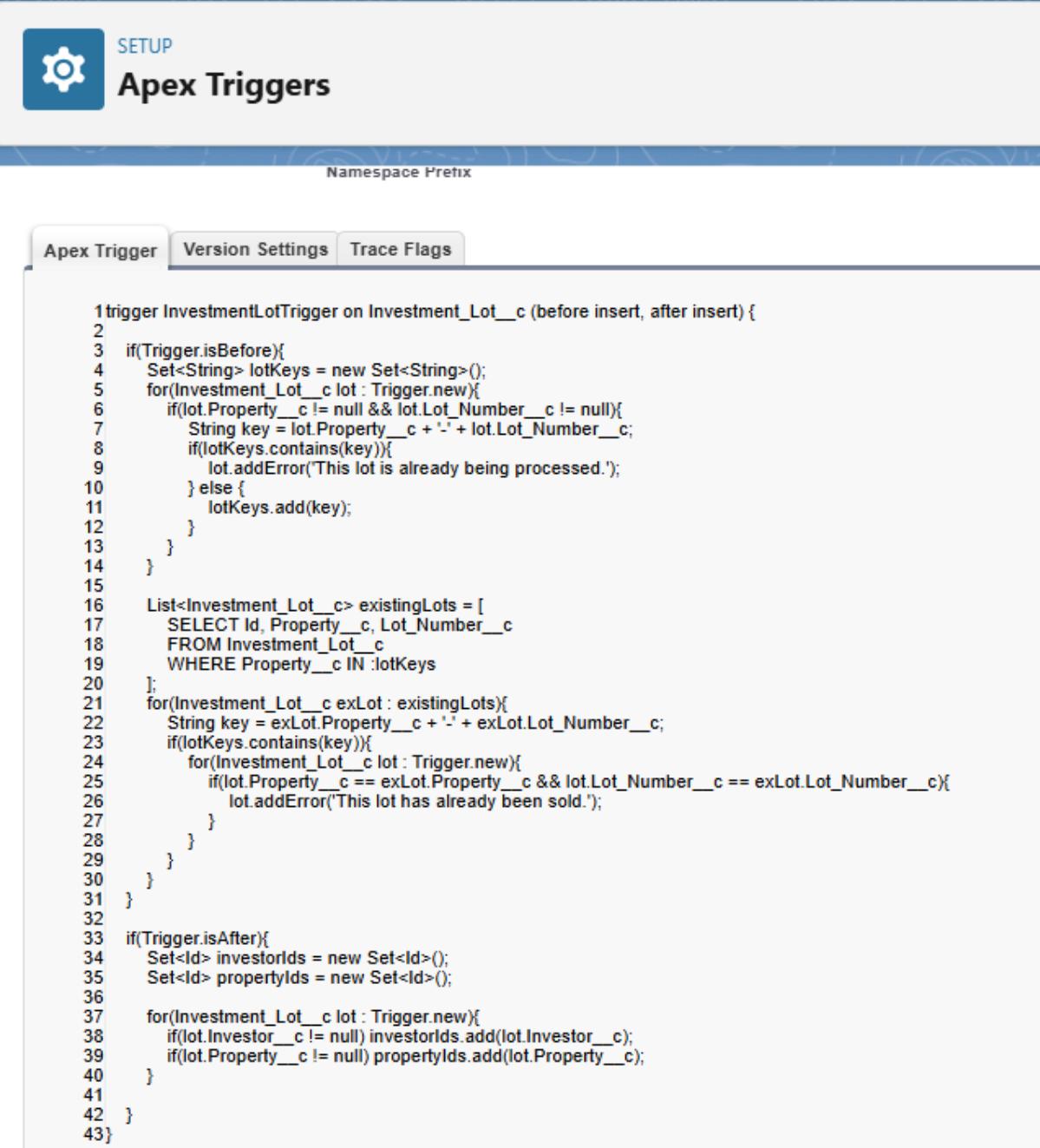
Trigger 2: InvestmentLotTrigger

- Object:** Investment_Lot__c
- Trigger Events:** after insert
- Purpose:** Links individual lots to investors and updates available lots in the property for accurate lot-level tracking.

Logic Overview:

- Query related property and investor for each lot.
- Decrement Property__c.Available_Lots__c automatically.
- Ensures lots are unique per transaction.

Screenshot:



The screenshot shows the Salesforce Setup interface for Apex Triggers. The title bar says "Apex Triggers". Below it, there's a "Namespace Prefix" field. The main area has tabs: "Apex Trigger" (which is selected), "Version Settings", and "Trace Flags". The code editor contains the following Apex trigger:

```

1 trigger InvestmentLotTrigger on Investment_Lot__c (before insert, after insert) {
2
3     if(Trigger.isBefore){
4         Set<String> lotKeys = new Set<String>();
5         for(Investment_Lot__c lot : Trigger.new){
6             if(lot.Property__c != null && lot.Lot_Number__c != null){
7                 String key = lot.Property__c + '-' + lot.Lot_Number__c;
8                 if(lotKeys.contains(key)){
9                     lot.addError('This lot is already being processed.');
10                } else {
11                    lotKeys.add(key);
12                }
13            }
14        }
15
16        List<Investment_Lot__c> existingLots = [
17            SELECT Id, Property__c, Lot_Number__c
18            FROM Investment_Lot__c
19            WHERE Property__c IN :lotKeys
20        ];
21        for(Investment_Lot__c exLot : existingLots){
22            String key = exLot.Property__c + '-' + exLot.Lot_Number__c;
23            if(lotKeys.contains(key)){
24                for(Investment_Lot__c lot : Trigger.new){
25                    if(lot.Property__c == exLot.Property__c && lot.Lot_Number__c == exLot.Lot_Number__c){
26                        lot.addError('This lot has already been sold.');
27                    }
28                }
29            }
30        }
31    }
32
33    if(Trigger.isAfter){
34        Set<Id> investorIds = new Set<Id>();
35        Set<Id> propertyIds = new Set<Id>();
36
37        for(Investment_Lot__c lot : Trigger.new){
38            if(lot.Investor__c != null) investorIds.add(lot.Investor__c);
39            if(lot.Property__c != null) propertyIds.add(lot.Property__c);
40        }
41
42    }
43}

```

3 Trigger Design Pattern

- **Concept:** One trigger per object, logic moved to a handler class.
- **Revaal Use:**
 - TransactionTrigger calls TransactionHandler.
 - InvestmentLotTrigger calls InvestmentLotHandler.
- **Benefit:** Bulk-safe and easier to maintain.

Screenshot: (Handler class structure showing methods for triggers)

4 SOQL & SOSL

- **SOQL Usage:**
 - Query investors to update portfolio.
 - Query investment lots linked to a property.
- **SOSL Usage:** Not required for this demo (no multi-object search needed).

Example:

```
[SELECT Id, Total_Investment__c FROM Investor__c WHERE Id IN :investorIds]
```

Screenshot: (*Any snippet of SOQL in the classes*)

5 Collections: List, Set, Map

- **List:** List<Investor__c> to store investors to update.
 - **Set:** Set<Id> to hold unique investor IDs.
 - **Map:** Map<Id, Investor__c> to map IDs to investor records for efficient updates.
-

6 Control Statements

- **if conditions:** Check for null investors.
 - **for loops:** Iterate over transactions and investment lots.
-

7 Asynchronous & Advanced Apex Concepts

- **Batch Apex / Queueable / Scheduled / Future Methods:** Not implemented, optional for future enhancements.
 - **Exception Handling:** Simple try-catch can be added to handle errors, but omitted for current capstone demo.
-

8 Test Classes

- **Purpose:** Ensure triggers and classes work correctly, mandatory for deployment.

Test Class Example:

```
@isTest
```

```

public class RevaaTriggerTest {

    @isTest static void testTransactionTrigger() {

        Property__c prop = new Property__c(Name='Test Prop', Price__c=1000000,
        Total_Lots__c=100, Available_Lots__c=100);

        insert prop;

        Investor__c inv = new Investor__c(Name='Investor A');

        insert inv;

        Transaction__c trans = new Transaction__c(Property__c=prop.Id, Investor__c=inv.Id,
        Lots_Purchased__c=10);

        insert trans;

        inv = [SELECT Total_Investment__c FROM Investor__c WHERE Id=:inv.Id];
        System.assertEquals(100000, inv.Total_Investment__c);

    }

    @isTest static void testInvestmentLotTrigger() {

        Property__c prop = new Property__c(Name='Lot Test', Price__c=500000,
        Total_Lots__c=50, Available_Lots__c=50);

        insert prop;

        Investor__c inv = new Investor__c(Name='Investor B');

        insert inv;

        Investment_Lot__c lot = new Investment_Lot__c(Property__c=prop.Id,
        Investor__c=inv.Id);

        insert lot;

        prop = [SELECT Available_Lots__c FROM Property__c WHERE Id=:prop.Id];
    }
}

```

```

        System.assertEquals(49, prop.Available_Lots__c);

    }

}

```

Class Body Class Summary Version Settings Trace Flags

```

1  @IsTest
2  public class RevaaTriggerTest {
3
4      @IsTest
5      static void testTransactionTrigger() {
6          Property__c prop = new Property__c(
7              Name = 'Test Prop',
8              Price__c = 1000000,
9              Total_Lots__c = 100
10         );
11         insert prop;
12
13         Investor__c inv = new Investor__c(
14             Name = 'Investor A'
15         );
16         insert inv;
17
18         Transaction__c trans = new Transaction__c(
19             Property__c = prop.Id,
20             Investor__c = inv.Id,
21             Lots_Purchased__c = 10
22         );
23         insert trans;
24
25         inv = [SELECT Total_Investment__c FROM Investor__c WHERE Id = :inv.Id];
26
27         Decimal expectedInvestment = 10 * (prop.Price__c / prop.Total_Lots__c);
28         System.assertEquals(expectedInvestment, inv.Total_Investment__c, 'Investor Total Investment should be updated correctly.');
29     }
30
31     @IsTest
32     static void testInvestmentLotTrigger() {
33         Property__c prop = new Property__c(
34             Name = 'Lot Test',
35             Price__c = 500000,
36             Total_Lots__c = 50
37         );
38         insert prop;
39
40         Investor__c inv = new Investor__c(
41             Name = 'Investor B'
42         );
43         insert inv;
44
45         Investment_Lot__c lot = new Investment_Lot__c(
46             Property__c = prop.Id,
47             Investor__c = inv.Id
48         );
49         insert lot;
50
51         Integer lotCount = [SELECT COUNT() FROM Investment_Lot__c WHERE Property__c = :prop.Id];
52         System.assertEquals(1, lotCount, 'One lot should be linked to the property.');
53     }
}

```

9 Expected Outcomes for Phase 5

1. Investor portfolio updates automatically on every new transaction.
2. Investment lots are correctly linked and managed, preventing duplicates.
3. Bulk-safe triggers and Apex classes ensure Revaa can handle multiple investors and transactions simultaneously.
4. Test coverage > 75%, ready for deployment.

Phase 6: User Interface Development (Revaa)

Objective

To design and configure an intuitive and functional user interface in Salesforce for Revaa, enabling Admins and Investors to access property, transaction, and portfolio data easily. This phase focuses on Lightning App Builder, Record Pages, Tabs, LWC, and navigation enhancements.

The screenshot shows the 'App Settings' page in Salesforce. The left sidebar has a blue header 'App Settings' and a section 'App Details & Branding' which is currently selected. Under this section, there are links for 'App Options', 'Utility Items (Desktop Only)', 'Navigation Items', and 'User Profiles'. The main content area is titled 'App Details & Branding' with the sub-section 'App Details'. It contains fields for 'App Name' (Revaa), 'Developer Name' (Aniket_Kakade), and 'Description' (Smart Fractional Real Estate Investment). To the right of this is the 'App Branding' section, which includes an 'Image' field showing the Revaa logo, a 'Primary Color Hex Value' field (#72BCFD), and an 'Org Theme Options' checkbox (which is checked). Below this is the 'App Launcher Preview' section, which shows a preview of the app icon and name.

1 Lightning App Builder

Purpose: Create apps, pages, and customize layouts to match user roles and workflows.

Steps:

1. Go to **Setup** → **Lightning App Builder**.
2. Click **New** → Select **App Page / Record Page / Home Page** depending on requirement.
3. Enter a **Label**: Revaa Dashboard App
4. Choose **Page Template** → e.g., One Region or Header & Two Columns.

5. Drag components from **Components Panel** onto the canvas (e.g., **Record Detail**, **List View**, **Custom LWC**)
6. Click **Save** → **Activate** → Assign to **Org Default / App Default / App, Record Type, Profile**.

Screenshot:

The screenshot shows the 'Lightning App Builder' interface with the 'SETUP' icon at the top left. The main title is 'Lightning App Builder'. Below it, a message 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.' A navigation bar at the top has 'View: All' and 'Create New View' buttons. Below the message is a table titled 'Lightning Pages' with columns: Action, Label, Name, Namespace Prefix, Description, Type, Created By, and Last Modified By. The table lists four pages: 'Lightning Record Page', 'Property_Record_Page', 'Revaa_Home_Page', and 'Transaction_Record_Page'.

The screenshot shows the 'Revaa Home Page' editor in the Lightning App Builder. The top navigation bar includes 'Lightning App Builder', 'Pages', 'Revaa Home Page', 'Activation...', and 'Save'. On the left, a 'Components' sidebar lists various components like Accordion, App Launcher, Assistant, etc. The main canvas displays a 'Key Deals - Recent Opportunities' component and a 'Recent Records' component. To the right, the 'Page' configuration pane shows fields for 'Label' (Revaa Home Page), 'API Name' (Revaa_Home_Page), 'Page Type' (Home Page), 'Template' (Home page header two colu...), and 'Description'. Buttons for 'Activation...' and 'Save' are at the top right of the configuration pane.

2 Record Pages

Purpose: Customize layouts for Property, Investor, and Transaction records.

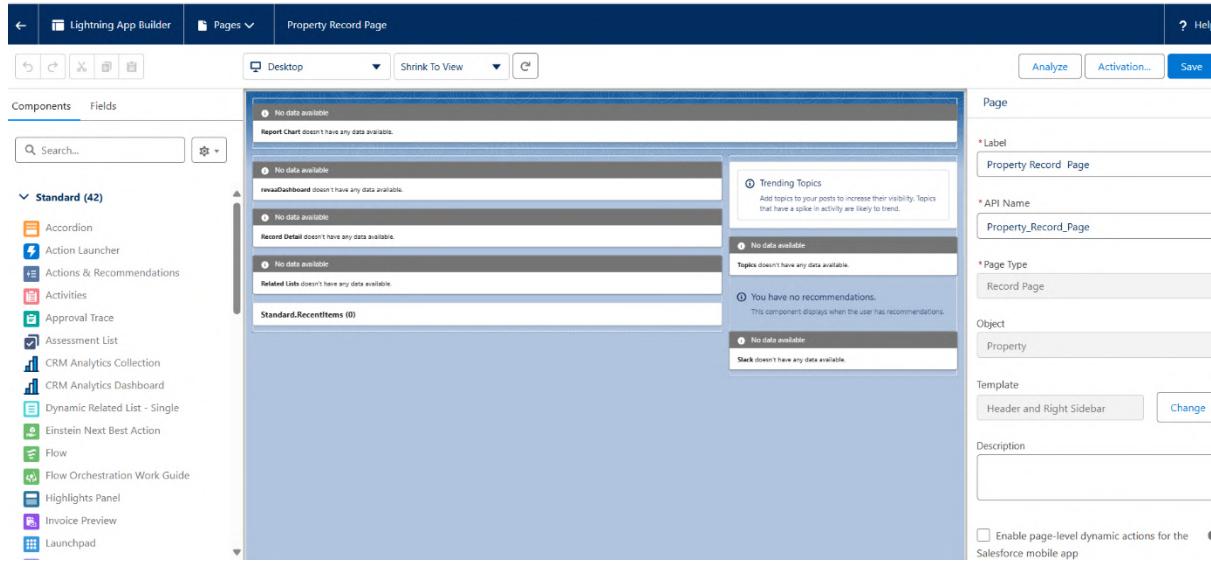
Steps:

1. Navigate to **Setup** → **Object Manager** → [Select Object, e.g., **Property**] → **Lightning Record Pages**.
2. Click **New** → Choose **Record Page** → Name it **Property Record Page**.
3. Select **Template** → Drag the following components:

- **Record Detail** (to show Property fields)
- **Related Lists** (Transactions, Investment Lots, Profit Distribution)
- **Custom LWC** (RevaaDashboard for visual insights)

4. Save & Activate → Assign for App, Profile, Record Type.

Screenshot:



3 Tabs

Purpose: Make navigation easier across custom objects and reports.

Steps:

1. Setup → **App Manager** → Open Revaa Dashboard App → **Edit**
2. Add **Tabs**:
 - Property
 - Investor
 - Transaction
 - Investment Lot
 - Profit Distribution
 - Reports & Dashboards
3. Arrange tabs in logical order.
4. Save & Finish.

Screenshot:

Custom Tabs

Help [f](#)

You can create new custom tabs to extend Salesforce functionality or to build new application functionality.

Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages Experience and the mobile app.

Custom Object Tabs		
Action	Label	Tab Style
Edit Del	Investment Lots	Bell
Edit Del	Investors	People
Edit Del	Profit Distributions	Stack of Cash
Edit Del	Properties	Building
Edit Del	Transactions	Safe

The screenshot shows the Lightning App Builder interface. At the top, there's a dark header bar with navigation icons and the text "Lightning App Builder", "App Settings", "Pages", and "Revaa". Below this is a sidebar with sections for "App Settings" (App Details & Branding, App Options, Utility Items (Desktop Only)) and "Navigation Items" (User Profiles). The main area is titled "Navigation Items" and contains two lists: "Available Items" and "Selected Items". The "Available Items" list includes various Salesforce objects and features like Activation Targets, Activations, All Sites, Alternative Payment Methods, Analytics, App Launcher, Appointment Categories, Appointment Invitations, Approval Requests, Approval Submission Details, and Approval Submissions. The "Selected Items" list contains "Accounts", "Contacts", "Reports", "Dashboards", "Investment Lots", "Investors", "Properties", "Transactions", and "Profit Distributions". Arrows between the two lists indicate the process of dragging items from the available list to the selected list.

4 Home Page Layouts

Purpose: Provide quick insights, metrics, and shortcuts.

Steps:

1. Setup → **Lightning App Builder** → **Home Page** → New
2. Drag components:
 - o **Dashboard** (Investor Portfolio / Property Investment Status)
 - o **Report Chart** (Profit Distribution)
 - o **Custom LWC** (quick investment actions or stats)

3. Activate → Assign to **Org Default / Profiles**

5 Utility Bar

Purpose: Add quick actions accessible from any page.

Steps:

1. Setup → **App Manager** → **Reva Dashboard App** → **Edit** → **Utility Bar**
2. Add items:
 - **Quick Create Transaction**
 - **Search Investor**
 - **Notifications / Alerts**
3. Configure width, icon, and label for each item.
4. Save → Done.

Screenshot:

The screenshot shows the Lightning App Builder interface with the following details:

- Header:** Includes "Lightning App Builder", "App Settings", "Pages", and "Revaa".
- Left Sidebar (App Settings):**
 - Utility Items (Desktop Only)** (selected)
 - Navigation Items
 - User Profiles
- Main Content Area:**
 - Utility Items (Desktop Only)**: A section with a sub-header "Utility Items (Desktop Only)" and a description "Give your users quick access to productivity tools and add background utility items to your app."
 - Add Utility Item** button
 - Utility Bar Alignment**: Set to "Default".
 - History Item Configuration:**
 - Properties**: History
 - Utility Item Properties**:
 - Label: History
 - Icon: Clock icon
 - Panel Width: 340
 - Panel Height: 480
 - Start automatically

6 Lightning Web Components (LWC)

Purpose: Add custom UI functionality like dashboards, charts, and interactive elements.

Steps:

1. Create LWC in VS Code:
2. `sfdx force:lightning:component:create --type lwc --componentname RevaaDashboard --outputdir force-app/main/default/lwc`
3. Files created:
 - o `RevaaDashboard.html` → UI template
 - o `RevaaDashboard.js` → Logic, Apex calls, and data handling
 - o `RevaaDashboard.js-meta.xml` → Metadata configuration (exposure to record pages)
4. Example configuration in `js-meta.xml`:
5. `<targets>`
6. `<target>lightning__RecordPage</target>`
7. `<target>lightning__AppPage</target>`
8. `<target>lightning__HomePage</target>`
9. `</targets>`
10. Deploy LWC to Salesforce:
 - o Right-click component folder → **SFDX: Deploy Source to Org**
11. Add LWC to record page or app page via **Lightning App Builder**.

Screenshot:

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

- Explorer View:** Shows the project structure under 'REVAAP'. It includes 'REVAAP', '.vscode', 'config', 'force-app/main/default', 'lwc\revaaDashboard', and 'TIMELINE'.
- Code Editor:** The active file is 'revaaDashboard.js' (JS). The code defines a class 'RevaaDashboard' extending 'LightningElement'. It uses @wire to get a record and handle its purchase. It also imports fields from 'lightning/uiRecordApi'.
- Terminal:** Shows the output of running Salesforce CLI commands in a PowerShell window. The commands include:
 - identical force-app/main/default/lwc\revaaDashboard\revaaDashboard.html
 - identical force-app/main/default/lwc\revaaDashboard_tests_revaaDashboard.test.js
 - identical force-app/main/default/lwc\revaaDashboard\revaaDashboard.js-meta.xmlIt also shows the execution of the Apex class 'RevaaDashboardController'.

7 Apex Integration with LWC

Purpose: Fetch or manipulate Salesforce data dynamically in LWC.

Steps:

Create Apex class:

```
public with sharing class RevaaDashboardController {  
    @AuraEnabled(cacheable=true)  
    public static List<Transaction__c> getTransactionsByInvestor(Id investorId){  
        return [SELECT Id, Lots_Purchased__c, Property__r.Name FROM Transaction__c  
        WHERE Investor__c=:investorId];  
    }  
}
```

Import Apex in LWC JS:

```
import getTransactionsByInvestor from  
'@salesforce/apex/RevaaDashboardController.getTransactionsByInvestor';
```

Screenshot: {Apex class in Salesforce Developer Console or VS Code}

8 Events in LWC

Purpose: Handle communication between components (child → parent).

Example:

```
// Dispatch event on lot selection
```

```
const selectedEvent = new CustomEvent('lotselect', { detail: this.selectedLot });

this.dispatchEvent(selectedEvent);
```

Screenshot: {Event usage code snippet in LWC JS file}

9 Wire Adapters & Imperative Apex Calls

Wire Adapter Example:

```
@wire(getTransactionsByInvestor, { investorId: '$recordId' })

transactions;
```

Imperative Apex Call Example:

```
getTransactionsByInvestor({ investorId: this.recordId })

.then(result => { this.transactions = result; })

.catch(error => { console.error(error); });
```

Screenshot: {Wire vs Imperative calls in LWC}

10 Navigation Service

Purpose: Enable navigation between objects, record pages, or URLs from LWC.

Example:

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

```
export default class RevaaDashboard extends NavigationMixin(LightningElement) {

    navigateToPropertyPage(propertyId) {
        this[NavigationMixin.Navigate]({
            type: 'standard__recordPage',
            attributes: {
```

```
        recordId: propertyId,  
        objectApiName: 'Property__c',  
        actionName: 'view'  
    }  
});  
}  
}
```

Expected Outcomes of Phase 6

- Intuitive, user-friendly interface for Admins and Investors.
 - LWC dashboards show real-time investment and profit data.
 - Custom record pages provide all related lists and quick actions.
 - Lightning App and utility bar enable efficient navigation and operations.
-

Phase 7: Integration & External Access

Objective:

Enable Revaa to securely communicate with external systems, perform API callouts, handle events, and manage data integration efficiently.

1 Named Credentials

Purpose: Store authentication settings for external systems, so you don't have to hardcode passwords, tokens, or endpoints in Apex or Flows.

Steps to Configure:

1. Navigate to **Setup → Named Credentials**.
2. Click **New Named Credential**.
3. Fill in the details:
 - o **Label:** RevaaLocalEndpoint
 - o **Name:** RevaaLocalEndpoint
 - o **URL:** `https://<ngrok-generated-url>` (*Example: https://chelsea-relevantly.ngrok-free.dev*)
 - o **Identity Type:** Anonymous or Named Principal (depending on authentication needs)
 - o **Authentication Protocol:** Choose as per your external service (e.g., OAuth 2.0 or Password Authentication)
4. Click **Save**.

SETUP > NAMED CREDENTIALS

NgrokLocalService

Label: NgrokLocalService

Name: NgrokLocalService

URL: https://chelsea-repand-relevantly.ngrok-free.dev

Enabled for Callouts:

Authentication:

- External Credential: LocalNgrokAuth
- Client Certificate

Callout Options:

- Generate Authorization Header:
- Allow Formulas in HTTP Header:
- Allow Formulas in HTTP Body:

Outbound Network Connection:

Critical Notes:

- Named Credential must exist before configuring **External Services** or using Apex callouts.
- Do **not** type the URL manually in callouts; always reference the Named Credential.

2 External Services

Purpose: Import OpenAPI (Swagger) schemas into Salesforce to create invocable actions for Flows or Apex.

Steps to Configure:

1. Navigate to **Setup → External Services**.
2. Click **New External Service**.
3. Provide the details:
 - **Name:** RevaaExternalAPI
 - **Named Credential:** Select RevaaLocalEndpoint (mandatory)
 - **Schema Type:** OpenAPI 2.0/Swagger
 - **Schema URL:** Paste your Swagger/OpenAPI JSON file URL or upload the schema.
4. Click **Save**.

Operation Name ↑	Description	Input parameters	Output parameters
createTransaction	Create a Transaction	body	responseCode, default, 201
getInvestor	Get Investor by ID	id	responseCode, 200, default
getTransactions	Get all Transactions		responseCode, default, 200

Critical Notes:

- You **cannot proceed** if no Named Credential exists; ensure Step 1 is complete.
- After import, all API actions are available as **invocable actions in Flow**.

3 Web Services (REST/SOAP) & Callouts

Purpose: Enable Revaa to make callouts to external APIs.

Steps for Apex REST Callout:

1. Ensure the endpoint domain is included in **Setup → Remote Site Settings**.
2. Create Apex class for callout:

```
public with sharing class RevaaAPIService {
    @future(callout=true)
    public static void callExternalAPI(String investorId) {
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setEndpoint('callout:RevaaLocalEndpoint/api/portfolio?investorId=' + investorId);
        request.setMethod('GET');
        HttpResponse response = http.send(request);
        System.debug('Response: ' + response.getBody());
    }
}
```

```
1  {
2    "openapi": "3.0.0",
3    "info": {
4      "title": "Revaa Local API",
5      "description": "OpenAPI schema for Revaa integration via ngrok",
6      "version": "1.0.0"
7    },
8    "servers": [
9      {
10        "url": "https://chelsea-repand-relevantly.ngrok-free.dev",
11        "description": "Ngrok tunnel for local development"
12      }
13    ],
14    "paths": {
15      "/transactions": {
16        "get": {
17          "summary": "Get all Transactions",
18          "operationId": "getTransactions",
19          "responses": {
20            "200": [
21              {
22                "description": "List of transactions",
23                "content": {
24                  "application/json": {
25                    "schema": {
26                      "type": "array",
27                      "items": {
28                        "$ref": "#/components/schemas/Transaction"
29                      }
29                    }
29                  }
29                }
29              ]
30            }
31          }
32        },
33      },
34      "post": {
35        "summary": "Create a Transaction",
36        "operationId": "createTransaction",
37        "requestBody": {
38          "required": true,
39          "content": {
40            "application/json": {
41              "schema": {
42                "$ref": "#/components/schemas/Transaction"
43              }
44            }
45          }
46        },
47        "responses": {
48          "201": {
49            "description": "Transaction created"
50          }
51        }
52      }
53    },
54    "/investors/{id)": {
55      "get": {
56        "summary": "Get Investor by ID",
57        "operationId": "getInvestor",
58        "parameters": [
59          {
60            "name": "id",
61            "in": "path",
62            "required": true,
63            "schema": {
64              "type": "string"
65            }
66          }
67        ]
68      }
69    }
70  }
71
```

Critical Notes:

- Use callout:<NamedCredentialName> to avoid hardcoding URLs.
- Methods making callouts must be annotated `@future(callout=true)` or run in asynchronous context.

4 Platform Events

Purpose: Enable real-time event-driven communication within Salesforce.

Steps to Configure:

1. Navigate to **Setup → Platform Events → New Platform Event**.
2. Create a Platform Event:
 - o **Label:** Transaction_Approved
 - o **API Name:** Transaction_Approved_e
 - o **Fields:** Investor__c (Lookup), Transaction__c (Lookup), Amount__c (Number)
3. Click **Save**.

Use Case: Trigger updates or notifications when a transaction is approved.

The screenshot shows the 'Platform Event Definition Detail' page for a 'Reva Transaction Event'. The top section displays basic details: Singular Label (Reva Transaction Event), Plural Label (Reva TransactionEvents), Object Name (Reva_Transaction_Event), API Name (Reva_Transaction_Event_e), Event Type (High Volume), Publish Behavior (Publish Immediately), and Created By (ANIKET KAKADE). The Deployment Status is Deployed. Below this, the 'Standard Fields' section lists fields like Created By, Created Date, Event UUID, and Replay ID with their corresponding data types (Lookup(User), Date/Time, Text(36), External Lookup) and controlling fields. The 'Custom Fields & Relationships' section shows a single custom field named 'Investor Name' with API name 'Investor_Name__c' and data type 'Text(18)'. The page also includes links for 'Standard Fields' and 'Custom Fields & Relationships'.

5 Change Data Capture (CDC)

Purpose: Track real-time changes to Salesforce records for integration with external systems.

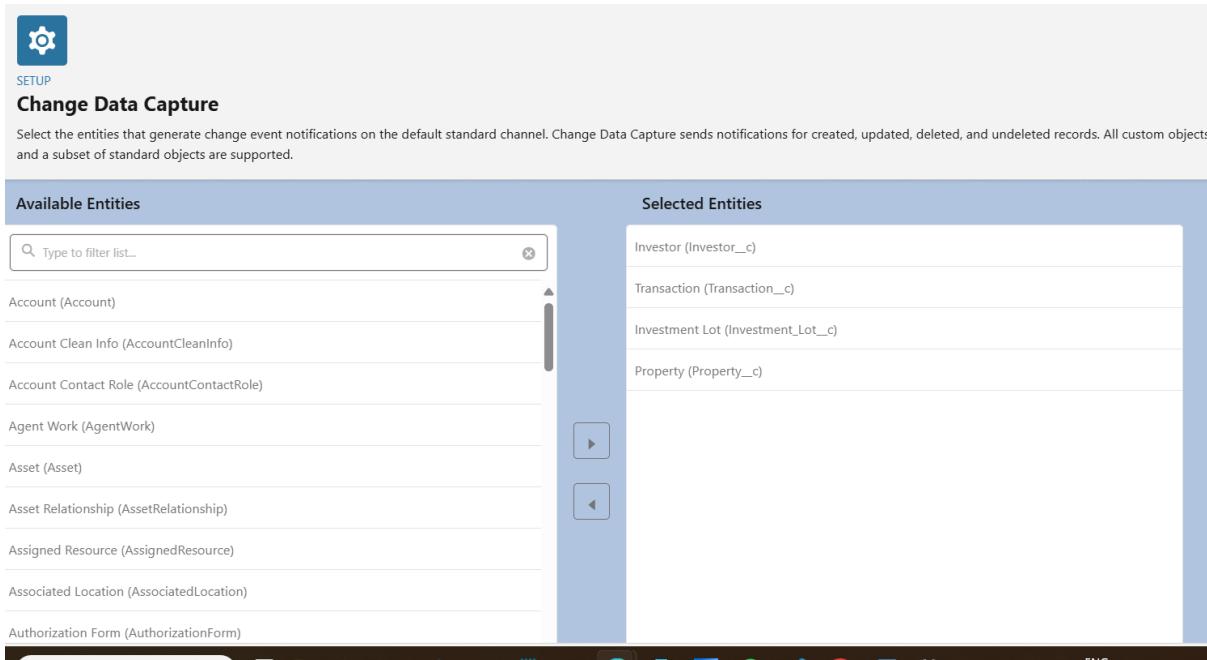
Steps to Configure:

1. Navigate to **Setup → Change Data Capture → Select Objects**.
(Screenshot: *Change Data Capture page*)
2. Select relevant objects:
 - o Transaction__c

- Investor__c
- Property__c

3. Enable events and click **Save**.

Use Case: External systems can subscribe to these events for real-time updates.



6 Salesforce Connect

Purpose: Access external data sources (ODBC, REST, SOAP) without importing data.

Steps to Configure:

1. Navigate to **Setup** → **External Data Sources** → **New External Data Source**.
(Screenshot: External Data Sources page)

2. Fill in:

- **Label:** RevaaPropertyService
- **Name:** RevaaPropertyService
- **Type:** OData 2.0/3.0 or other relevant type
- **URL:** Your ngrok endpoint or API URL
- **Authentication:** Named Credential (RevaaLocalEndpoint)

3. Click **Save and Validate**.

4. Click **Sync** to bring in external objects.
-

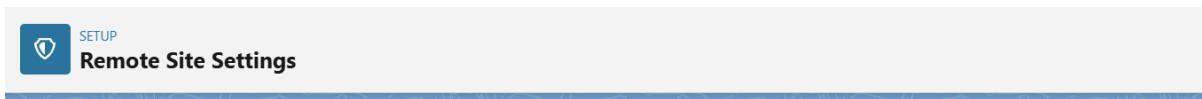
7 API Limits & Authentication

- Monitor API consumption at **Setup** → **System Overview** → **API Usage**.
 - For OAuth or token-based authentication, always create a **Connected App** if required.
 - Named Credential handles authentication for Flows, Apex, and External Services.
-

8 Remote Site Settings

Purpose: Required for direct Apex callouts if **Named Credentials** are not used.

1. Navigate to **Setup** → **Remote Site Settings** → **New Remote Site**.
2. Fill in:
 - **Remote Site Name:** RevaaLocalHost
 - **Remote Site URL:** <https://chelsea-repand-relevantly.ngrok-free.dev>
3. Click **Save**.



Remote Site Details

Remote Site Detail		Edit	Delete	Clone	Modified By	ANIKET KAKADE, 9/28/2025, 4:30 PM
Remote Site Name	Revaa_External_API					
Remote Site URL	https://chelsea-repand-relevantly.ngrok-free.dev					
Disable Protocol Security	<input type="checkbox"/>					
Description						
Active	<input checked="" type="checkbox"/>					
Created By	ANIKET KAKADE, 9/28/2025, 4:30 PM					

[Edit](#) [Delete](#) [Clone](#)

Phase 7 Summary

- **Named Credentials:** Secure authentication.
- **External Services:** Integrate external APIs as invocable actions.
- **Callouts:** Perform REST/SOAP communication.
- **Platform Events & CDC:** Real-time event handling.
- **Salesforce Connect:** Access external objects without importing.
- **Remote Site Settings:** Required for Apex callouts without Named Credentials.

Phase 8: Data Management & Deployment – Revaa Documentation

Overview

Phase 8 focuses on how data is managed in Salesforce, including importing, exporting, backing up, maintaining data quality, and deploying Revaa's configurations between orgs. This ensures that Revaa remains functional, scalable, and secure.

1. Data Import Wizard

The **Data Import Wizard** allows us to import records for standard and custom objects like Investor__c, Property__c, Transaction__c, and Investment_Lot__c.

Steps:

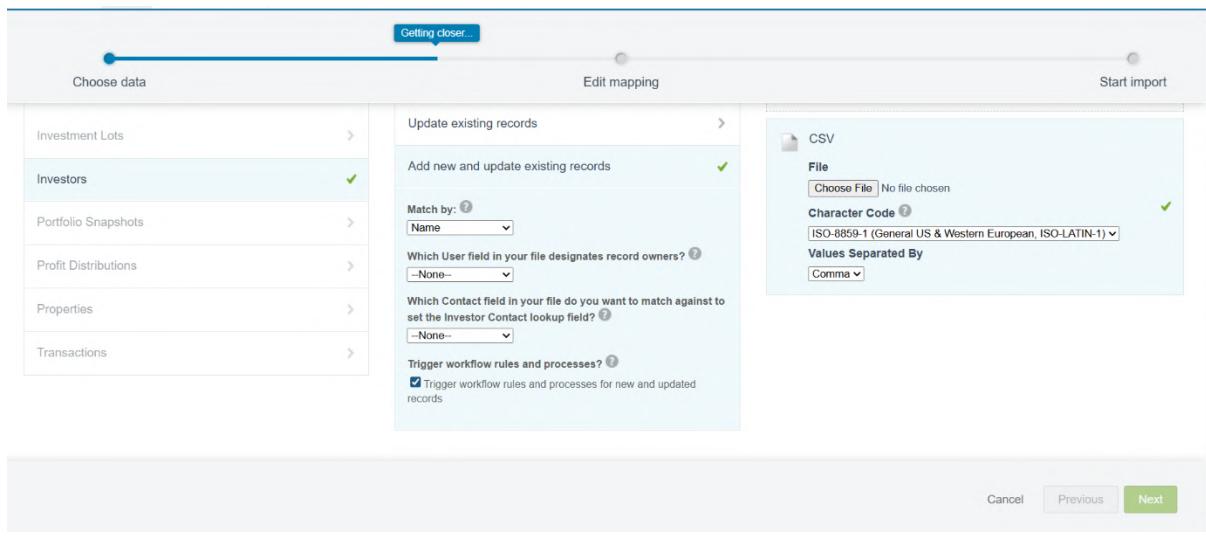
1. Navigate to: **Setup → Data → Data Import Wizard**.
2. Click **Launch Wizard**.
3. Select **Standard or Custom Object** (e.g., Investor__c).
4. Choose **Add New Records** or **Update Existing Records** depending on the use case.
5. Upload the CSV file.
 - Example: Investor__c.csv for investors.
 - Map CSV columns to Salesforce fields:
 - Name → Name
 - Email → Email
 - Total_Investment__c → Total_Investment__c
6. Click **Next → Start Import**.

Screenshot Step: Take a screenshot after mapping CSV columns (Setup → Data Import Wizard → Map Fields).

7. Repeat the steps for **Property__c, Transaction__c, and Investment_Lot__c** in order.

Notes:

- Ensure all **lookup relationships** are mapped correctly (e.g., Transaction__c → Investor__c, Investment_Lot__c → Transaction__c).
- Maintain **consistent naming conventions** as per Revaa schema.



2. Data Loader

For large datasets or automated imports/exports, **Data Loader** is preferred.

Steps:

1. Download and install **Data Loader**.
2. Open Data Loader → Click **Insert/Update/Upsert/Delete/Export** depending on the task.
3. Log in using **Salesforce credentials**.
4. Select **object** (e.g., Transaction__c).
5. Choose **CSV file** for import.
6. Map fields and perform the operation.

Screenshot Step: Take screenshot of the mapping screen after field mapping.

Tips:

- Use **External IDs** for Upsert operations.
- Always back up data before performing Delete or Update operations.



What is it?

Data Loader is a client application for the bulk import or export of data. Use it to insert, update, delete, or export Salesforce records.

When importing data, Data Loader reads, extracts, and loads data from comma-separated values (CSV) files or from a database

3. Duplicate Rules

To maintain clean data, we create **Duplicate Rules** for key objects.

Steps:

1. Navigate to: **Setup → Duplicate Management → Duplicate Rules**.
2. Click **New Rule** → Select object (e.g., `Investor__c`).
3. Define **Matching Rule**:
 - Example: Match **Email** or **Name**.
4. Specify **Action on Create/Update**:
 - Block
 - Allow with Alert
5. Activate the rule.

Screenshot:

Duplicate Rule Detail

Rule Name		Investor Duplicate Management	Order	1 of 1 [Reorder]
Description	Object	Investor	Operations On Create	<input checked="" type="checkbox"/> Alert <input checked="" type="checkbox"/> Report
Record-Level Security	Action On Create	Enforce sharing rules Allow	Operations On Edit	<input type="checkbox"/> Alert <input type="checkbox"/> Report
Action On Edit	Action On Edit	Allow		
Alert Text	Use one of these records?			
Active	<input type="checkbox"/>			
Matching Rule	⚠ Investor Duplicate Management matching rule Mapped			Matching Criteria (Investor: Email EXACT MatchBlank = FALSE) AND (Investor: Contact_Number EXACT MatchBlank = FALSE)
Conditions				Created By ANIKET KAKADE, 9/28/2025, 6:26 PM Modified By ANIKET KAKADE, 9/28/2025, 6:26 PM

4. Data Export & Backup

Regular backups are crucial for Revaa.

Steps:

1. Navigate to: **Setup → Data → Data Export**.
2. Click **Export Now** or **Schedule Export**.
3. Select objects to export: **Investor_c**, **Property_c**, **Transaction_c**, **Investment_Lot_c**.
4. Choose **include attachments and documents** if required.
5. Start export. Download ZIP file with CSVs.

The screenshot shows the 'Data Export' setup page in Salesforce. At the top, there are tabs for 'Setup', 'Home', and 'Object Manager'. A search bar says 'Q, Data Ex...'. Below the search bar, there's a 'Data Export' section with a sub-section 'Monthly Export Service'. The configuration includes:

- Export File Encoding:** ISO-8859-1 (General US & Western European, ISO-LATIN-1)
- Include images, documents, and attachments:**
- Include Salesforce Files and Salesforce CRM Content document versions:**
- Replace carriage returns with spaces:**

Exported Data:

Select what type of information you would like to include in the export. The data types listed below use the Apex API names. If you are not familiar with these names, select Include all data for your export.

Include all data

<input type="checkbox"/> Order	<input type="checkbox"/> OrderItem
<input type="checkbox"/> Account	<input type="checkbox"/> Contact
<input type="checkbox"/> Asset	<input type="checkbox"/> Product2
<input type="checkbox"/> Lead	<input type="checkbox"/> UserRole
<input type="checkbox"/> BusinessProcess	<input type="checkbox"/> Case
<input type="checkbox"/> Campaign	<input type="checkbox"/> CaseSolution
<input type="checkbox"/> CaseContactRole	<input type="checkbox"/> ContentVersionMap
<input type="checkbox"/> ContentDocumentLink	<input type="checkbox"/> DatedConversionRate
<input type="checkbox"/> ContractContactRole	<input type="checkbox"/> EmailRoutingAddress
<input type="checkbox"/> EmailDisclaimer	<input type="checkbox"/> Event
<input type="checkbox"/> EntityHistory	<input type="checkbox"/> FeedFieldHistory
<input type="checkbox"/> EventRelation	<input type="checkbox"/> FieldHistory
<input type="checkbox"/> FeedPost	<input type="checkbox"/> Individual
<input type="checkbox"/> FiscalYearSettings	<input type="checkbox"/> Note
<input type="checkbox"/> LinkReference	<input type="checkbox"/> OpportunityContactRole
<input type="checkbox"/> Opportunity	<input type="checkbox"/> OrgWideEmailAddress
<input type="checkbox"/> OpportunityHistory	<input type="checkbox"/> ProcessInstance
<input type="checkbox"/> Period	<input type="checkbox"/> RecordType
<input type="checkbox"/> ProcessInstanceStep	<input type="checkbox"/> TaskRelation
<input type="checkbox"/> Solution	<input type="checkbox"/> VoiceCallRecording
<input type="checkbox"/> VoiceCall	<input type="checkbox"/> RequestsForAccessSIQ
<input type="checkbox"/> SISUserBlacklist	
<input type="checkbox"/> SalesforceIQSyncFailure	
<input type="checkbox"/> AccountContactRole	

Start Export | **Cancel**

5. Change Sets

Change sets allow deploying **metadata** between orgs (Sandbox → Production).

Steps:

1. Navigate to: **Setup** → **Deployment** → **Outbound Change Sets**.
2. Click **New** → Name the change set (e.g., RevaaPhase8_ChangeSet).
3. Add **components**: Objects, Fields, Validation Rules, Flows, Apex Classes, Triggers.
4. Upload change set to **target org**.

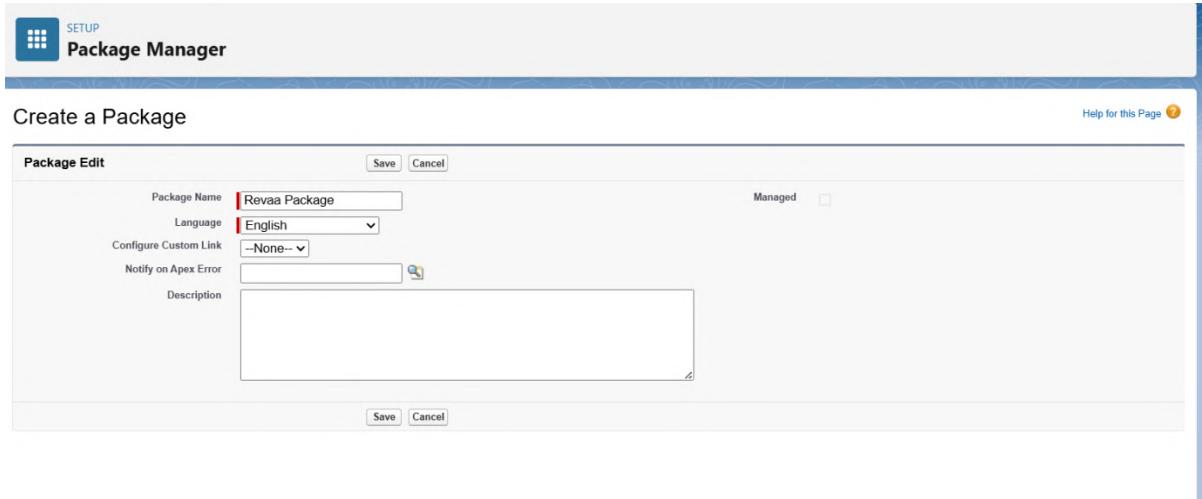
6. Unmanaged vs Managed Packages

- **Unmanaged Package:**
 - For distributing customizations for demonstration or learning.
 - Changes in the package do not sync after installation.
- **Managed Package:**
 - For production-grade apps.

- Supports upgrades and licensing.

Steps to Create:

1. Setup → Packages → New → Select Managed or Unmanaged.
2. Add components → Upload.



7. ANT Migration Tool

The **ANT Migration Tool** allows deployment via XML metadata files.

Steps:

1. Download **Force.com Migration Tool (ANT)**.
2. Configure **build.xml** and **package.xml** with required components.
3. Use commands:
ant retrieve
ant deploy
4. ant retrieve
5. ant deploy
6. Validate deployment before pushing to production.

8. VS Code & SFDX

VS Code with Salesforce Extensions helps in development, deployment, and version control.

Steps:

1. Open VS Code → Install **Salesforce Extension Pack**.
2. Authenticate with org:

3. sfdx force:auth:web:login -a RevaaSandbox
4. Retrieve or deploy metadata using SFDX commands.
5. Create new objects, Apex classes, and LWC from VS Code.

```

{
  "version": "1.0.0",
  "servers": [
    {
      "url": "https://euher1se-repos-relevant.apql-free-dev",
      "description": "My org tunnel for local development"
    }
  ],
  "paths": {
    "/transactions": {
      "type": "object",
      "summary": "Get all Transactions",
      "operationId": "getTransactions",
      "requestId": "transaction"
    },
    "/list": {
      "description": "List of transactions",
      "query": {
        "application/json": {
          "schema": {
            "type": "array",
            "items": "#ref: '#components/schemas/Transaction'"
          }
        }
      }
    },
    "/post": {
      "summary": "Create a Transaction",
      "operationId": "createTransaction",
      "requestId": "transaction"
    }
  },
  "required": true,
  "content": {
    "application/json": {
      "schema": {
        "type": "object"
      }
    }
  }
}
  
```

PS C:\Users\lakile\Revaa>sfdx auth:web:login -a RevaaOrg
Successfully authenticated user lalitkale504@gmail.com with org ID 00q:000000000000000
PS C:\Users\lakile\Revaa>sfdx force:source:pull
PS C:\Users\lakile\Revaa>force source pull
Did you mean force source open? Yes
Opening org euher1se-repos-relevant as user lalitkale504@gmail.com
Waiting for the Lightning Experience-enabled custom domain..... done
PS C:\Users\lakile\Revaa>force source pull
PS C:\Users\lakile\Revaa>force source pull
Did you mean force source open? Yes
Opening org euher1se-repos-relevant as user lalitkale504@gmail.com
Waiting to resolve the Lightning Experience-enabled custom domain..... done
PS C:\Users\lakile\Revaa>

Best Practices for Revaa Data Management

- Always maintain **data integrity** by using duplicate rules.
- Schedule **regular backups**.
- Use **Change Sets** or **SFDX** for consistent deployments.
- Validate all **lookup relationships** during import.
- Keep **test data** separate from production data.

Phase 9: Reporting, Dashboards & Security Review

Objective:

Phase 9 ensures that Revaa users have actionable insights into **investor portfolios, transactions, and property performance**. This includes **custom reports, dashboards**, and a thorough **security review** of all data exposed in reports and dashboards.

1 Custom Report Types

Purpose:

To define a reusable report structure that allows combining **Investors, Transactions, and Property** objects for reporting.

Steps to Create Custom Report Type

1. Go to **Setup → Report Types → New Custom Report Type**.
2. Configure the Primary Object:
 - **Primary Object:** Investor__c
 - **Report Type Label:** Investor Portfolio Report
 - **Report Type Name (API):** Investor_Portfolio_Report
 - **Description:** Investor's total portfolio details including transactions and properties
 - **Category:** Investor Reports (Create a new category if it does not exist)
 - **Deployment Status:** Deployed
 - {Screenshot: “Report Type Creation Window”}
3. Add Related Objects (Optional for Expanded Reporting):
 - **Transaction__c** (Related by Investor__c)
 - **Property__c** (Related via Transaction__c.Property__c)
 - Ensure the relationship is set as "**Each "A" record must have at least one related "B" record**" if mandatory.
 - {Screenshot: “Related Object Selection Window”}
4. Save the report type.

Investment Lot Report

[Preview Layout](#) [Edit Layout](#) [Clone](#) [Delete](#) [Close](#)

Below is the information for this custom report type. You can click the buttons on this to preview or update information for the custom report type

Details

Display Label	Investment Lot Report
API Name	Investment_Lot_Report
Description	Tracks individual lots purchased per transaction
Created By	ANIKET KAKADE, 9/29/25, 8:17 AM
Store in Category	other
Deployment Status	Deployed
Modified By	ANIKET KAKADE, 9/29/25, 8:17 AM

Fields

Source Object	Included Fields
Investment Lots	14
Transactions	19
Activities	36
Assigned To	74

Object Relationships

Investment Lots (A)

```

graph TD
    A((A)) --- B((B))
    A --- C((C))
    A --- D((D))
    B --- C
    B --- D
    C --- D
    
```

with at least one related record from Transactions (B)
 with at least one related record from Activities (C)
 with at least one related record from Assigned To (D)

Investor Portfolio Report

[Preview Layout](#) [Edit Layout](#) [Clone](#) [Delete](#)

Below is the information for this custom report type. You can click the buttons on this to preview or update information for the custom report type

Details

Display Label	Investor Portfolio Report
API Name	Investor_Portfolio_Report
Description	Investor's total Portfolio Details
Created By	ANIKET KAKADE, 9/29/25, 7:56 AM
Store in Category	other
Deployment Status	Deployed
Modified By	ANIKET KAKADE, 9/29/25, 7:56 AM

Fields

Source Object	Included Fields
Investors	16
Transactions	19
Duplicate Record Items	8

Object Relationships

Investors (A)

```

graph TD
    A((A)) --- B((B))
    A --- C((C))
    B --- C
    
```

with at least one related record from Transactions (B)
 with at least one related record from Duplicate Record Items (C)

2 Creating Reports

Purpose:

To provide detailed **investor portfolio insights**.

Steps

1. Navigate to **Reports** → **New Report**.
2. Select the custom report type: Investor Portfolio Report.
3. Configure columns:
 - **Investor Name:** Investor__c.Name
 - **Property Name:** Transaction__c.Property__c
 - **Lots Purchased:** Transaction__c.Lots_Purchased__c
 - **Investment Amount:** Transaction__c.Ownership_Amount__c
 - **Total Portfolio Value:** Investor__c.Total_Investment__c
 - {Screenshot: “Report Builder Columns Configuration”}
4. Apply **filters** as needed:
 - Date of Transaction
 - Property Type
 - Investor Name
 - {Screenshot: “Report Filters Window”}
5. **Save & Run** the report.

Tip: Use **Summary Reports** to group data by Investor or Property, **Matrix Reports** for multi-dimensional analysis, and **Joined Reports** to compare multiple datasets.

Withdrawal Request Report

[Preview Layout](#) [Edit Layout](#) [Clone](#)

Below is the information for this custom report type. You can click the buttons on this to preview or update information for the custom report type.

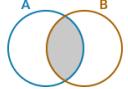
Details

Display Label	Withdrawal Request Report
API Name	Withdrawal_Request_Report
Description	Tracks withdrawal requests and approval status
Created By	ANIKET KAKADE, 9/29/25, 8:09 AM
Store in Category	admin
Deployment Status	Deployed
Modified By	ANIKET KAKADE, 9/29/25, 8:09 AM

Object Relationships

Transactions (A)

... with at least one related record from Activities (B)



A B

Fields

Source Object	Included Fields
Transactions	20
Activities	36

Property Transactions Report

[Preview Layout](#) [Edit Layout](#) [Clone](#) [Delete](#)

Below is the information for this custom report type. You can click the buttons on this to preview or update information for the custom report type.

Details

Display Label	Property Transactions Report
API Name	Property_Transactions_Report
Description	Total Transactions done on Property
Created By	ANIKET KAKADE, 9/29/25, 8:14 AM
Store in Category	other
Deployment Status	Deployed
Modified By	ANIKET KAKADE, 9/29/25, 8:14 AM

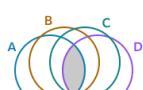
Object Relationships

Properties (A)

... with at least one related record from Investment Lots (B)

... with at least one related record from Transactions (C)

... with at least one related record from Activities (D)



A B C D

Fields

Source Object	Included Fields
Properties	19
Investment Lots	13
Transactions	19
Activities	36

Dashboards

Purpose:

To visualize investor and property metrics for easy consumption by stakeholders.

Steps

1. Go to **Dashboards** → **New Dashboard**.
2. Configure:
 - **Name:** Investor Portfolio Dashboard
 - **Folder:** Investor Reports
3. Add components:
 - **Pie Chart:** Total Investment per Investor
 - **Bar Chart:** Lots Purchased per Property
 - **Gauge:** Portfolio Value vs Expected Value
 - {Screenshot: “Dashboard Builder Window”}
4. Select the corresponding **custom report** for each component.
5. Save and run the dashboard.

Tip: Use **dynamic dashboards** to reflect data based on the logged-in user (optional for admin vs investor view).


SETUP
Reports and Dashboards Settings

Report and Dashboard User Interface Settings

Modify the behavior of the user interface for report and dashboard pages using the following settings:

User Interface

Enable Floating Report Headers (Salesforce Classic only)
 Enable Dashboard Finder [i](#)
 Hides the option to export a report in XLS format in Lightning Experience
 Enable Inline Editing in Reports (Lightning Experience Only)
 Enable Org Wide Email Address for Report Subscription [i](#)

Enable Org Wide Email Address for Dashboard Subscription [i](#)
 Enable Content Delivery Network (CDN) for Lightning Reports and Dashboards
 Enhanced Custom Report Type Setup Page

Confidential Information Disclaimer Settings

Specify whether or not to exclude a disclaimer that says "Confidential Information - Do Not Distribute" from report footers.

Exclude Disclaimer from Formatted Report Exports in Lightning Experience
 Exclude Disclaimer from Report Run Pages and from Printable View Pages (Salesforce Classic Only)

Chatter Options

Enable Dashboard Component Snapshots [i](#)

Unified Analytics Home

Show preview thumbnails for Lightning reports and dashboards [i](#)
 Enable the Unified Experience for Analytics Home [i](#)

4 Security Review

A. Field-Level Security

1. Navigate to **Setup → Profiles → [Relevant Profile] → Field-Level Security**.
2. Ensure sensitive fields are secured:
 - o Investor__c.Total_Investment__c
 - o Transaction__c.Ownership_Amount__c
 - o Property__c.Price__c
 - o {Screenshot: “Field-Level Security Settings”}

B. Sharing Settings

1. Go to **Setup → Sharing Settings**.
2. Configure **Organization-Wide Defaults (OWD)**:

- **Investor_c:** Private
 - **Transaction_c:** Controlled by Parent (Investor)
 - **Property_c:** Public Read-Only
3. Use **Sharing Rules** to grant access to managers or admins as needed.
 4. {Screenshot: “Sharing Settings Window”}
-

C. Login & Session Security

1. Navigate to **Setup → Session Settings**.
 2. Configure:
 - Session timeout duration
 - Login IP ranges
 - Login access policies for users
 3. {Screenshot: “Session Settings Window”}
-

D. Audit Trail

1. Go to **Setup → View Setup Audit Trail**.
 2. Track all changes to objects, fields, users, and security settings.
 3. {Screenshot: “Audit Trail Page”}
-

5 Outcome of Phase 9

- Custom reports provide **detailed insights** into investor portfolios, property investments, and transaction history.
- Dashboards **visualize key metrics** for decision-making.
- Security configurations ensure **data privacy and compliance**.
- Audit trail provides a **historical record** of changes to critical Salesforce objects and settings.