## 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.

---

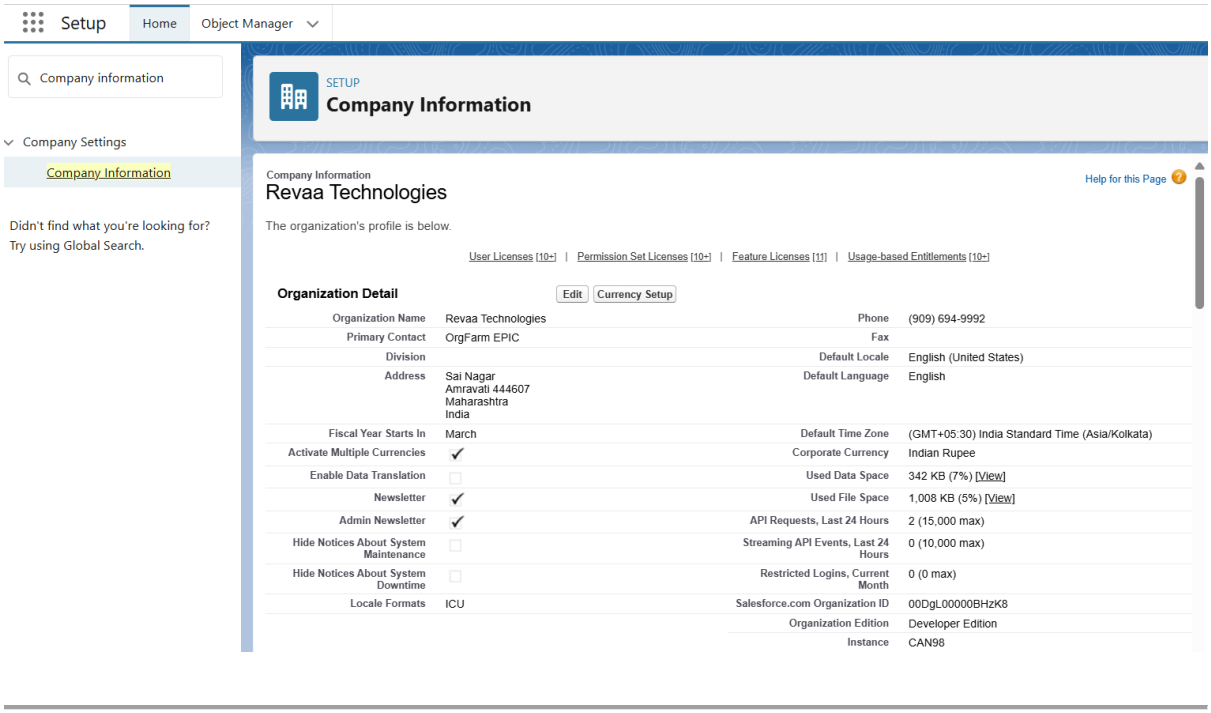
## Step 1: Salesforce Dev Org Setup

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:
    - Setup → My Domain → Choose a domain → Check availability → Save & Deploy.

---

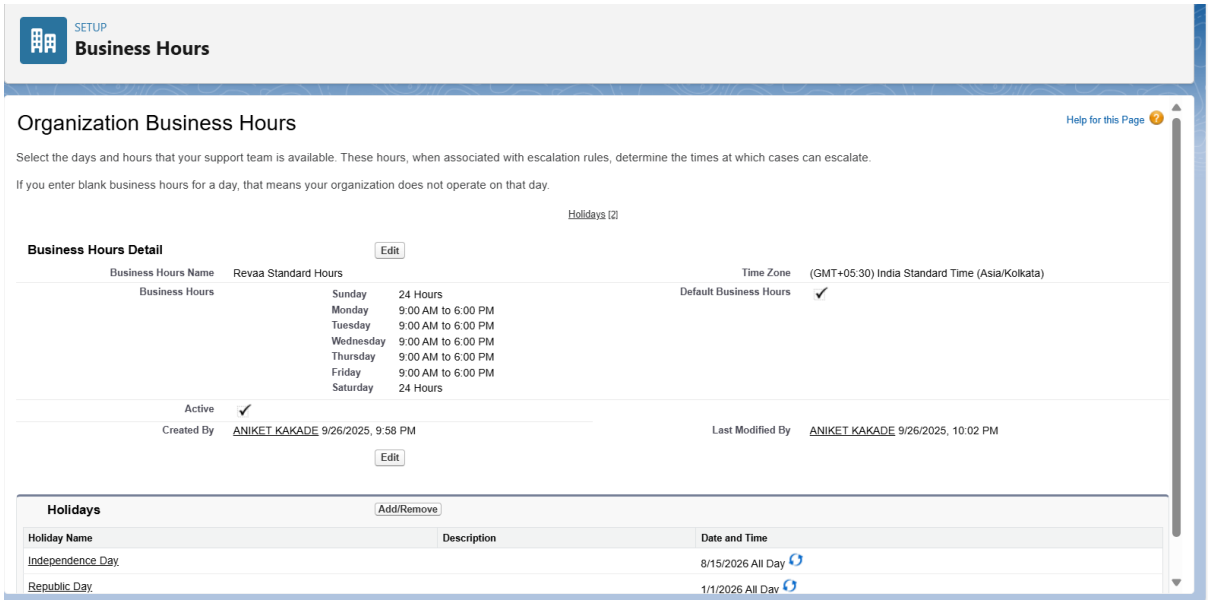
## Step 2: Company Profile Setup

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




## Step 3: Business Hours & Holidays

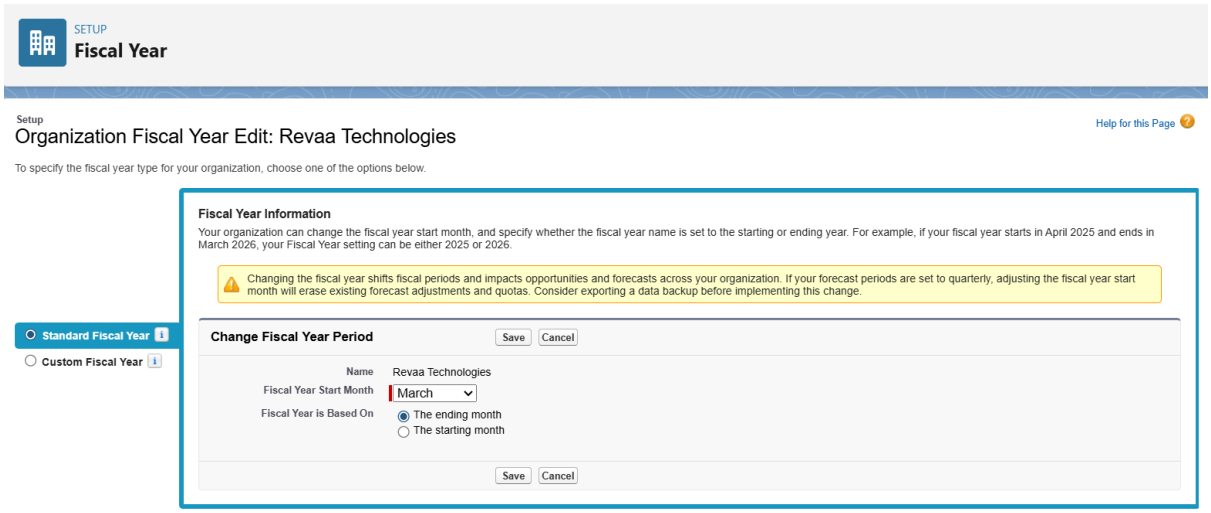
1. Setup → Business Hours → New:
    - Business Hours Name: Revaa Standard Hours
    - Days: Monday – Friday
    - Hours: 9:00 AM – 6:00 PM
    - Save.
2. Setup → Holidays → New:
    - Enter national holidays (e.g., Diwali, Independence Day).
    - **Screenshot:**




---

# Step 4: Fiscal Year Settings

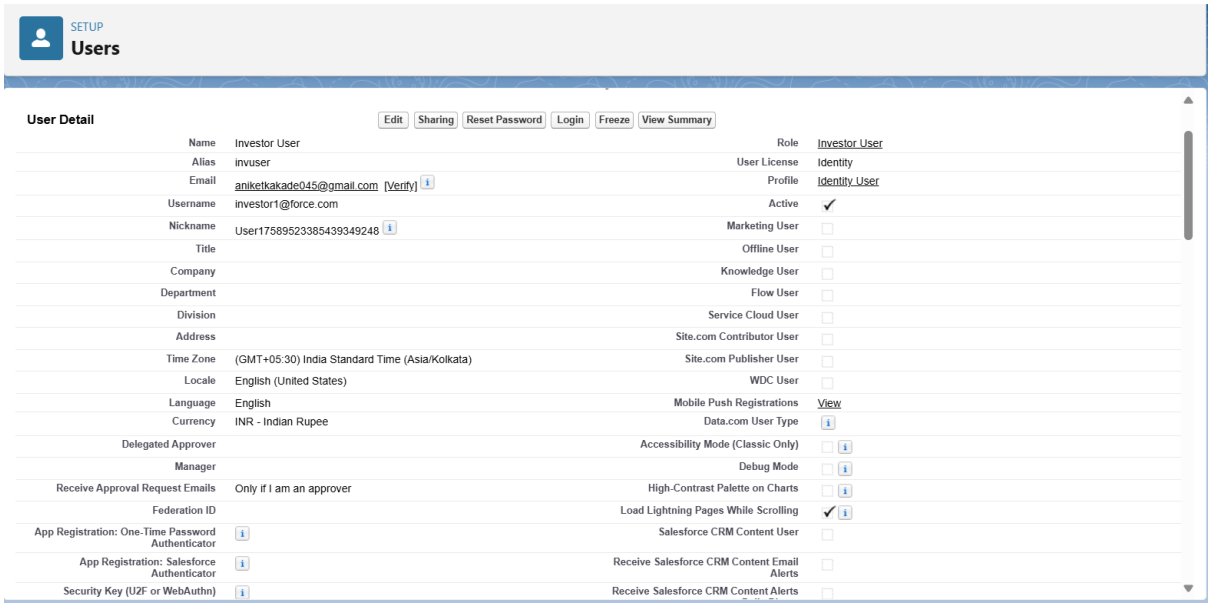
1. Setup → Fiscal Year → Activate Standard Fiscal Year.
2. If using custom fiscal periods (optional), create them here.
    - **Screenshot:**




---

## 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:**




---

## Step 6: Profiles

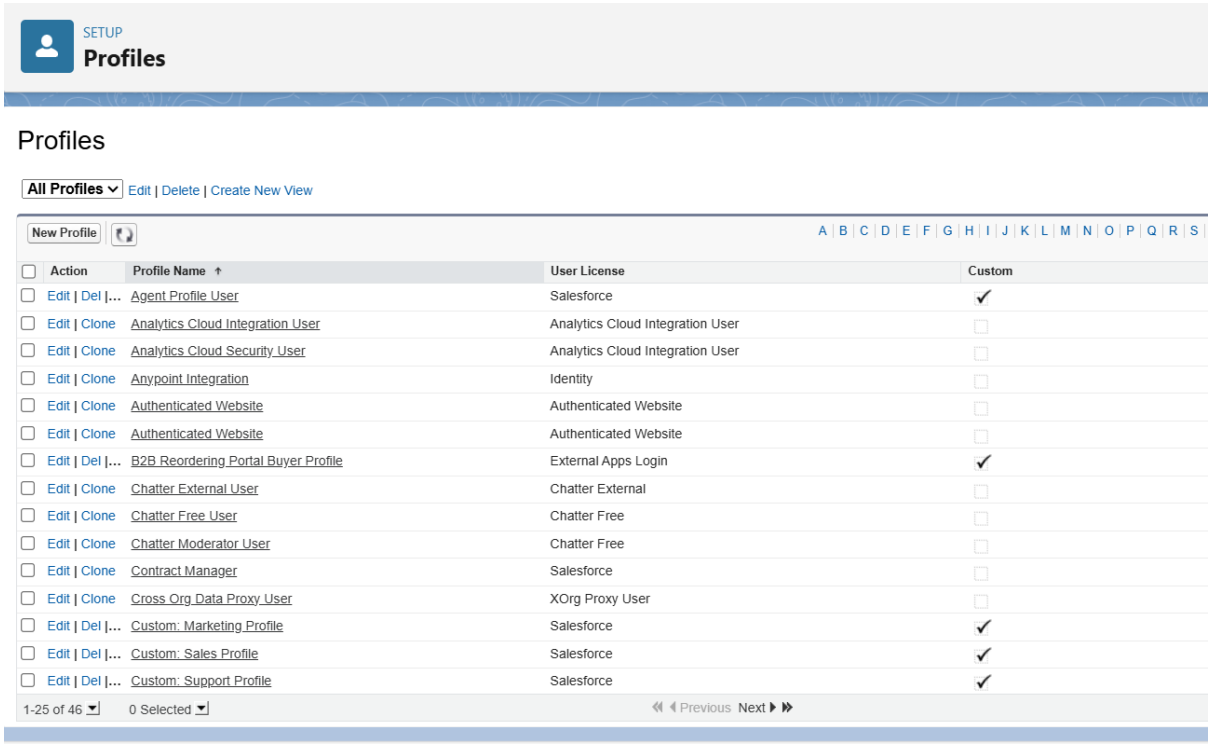
### 6.1 Default Profiles

- **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:**




---

## Step 7: Roles

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

CEO

├── CFO

│ └── Finance Analyst (optional)

└── COO

└── Operations Manager

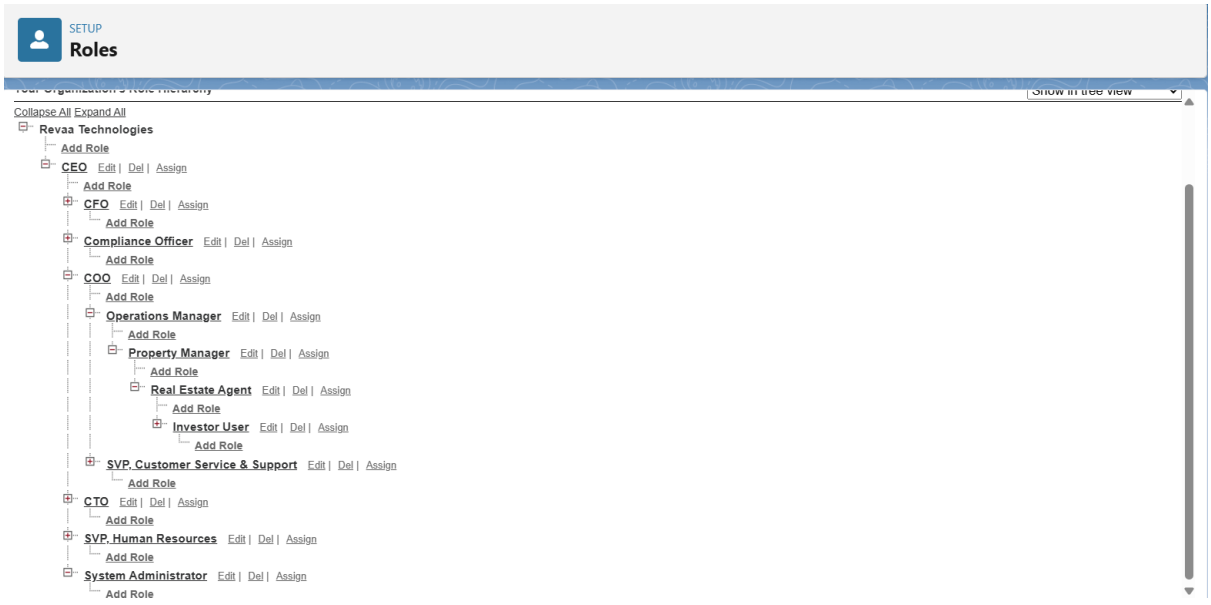
└── Property Manager

└── Agent

└── Investor User

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

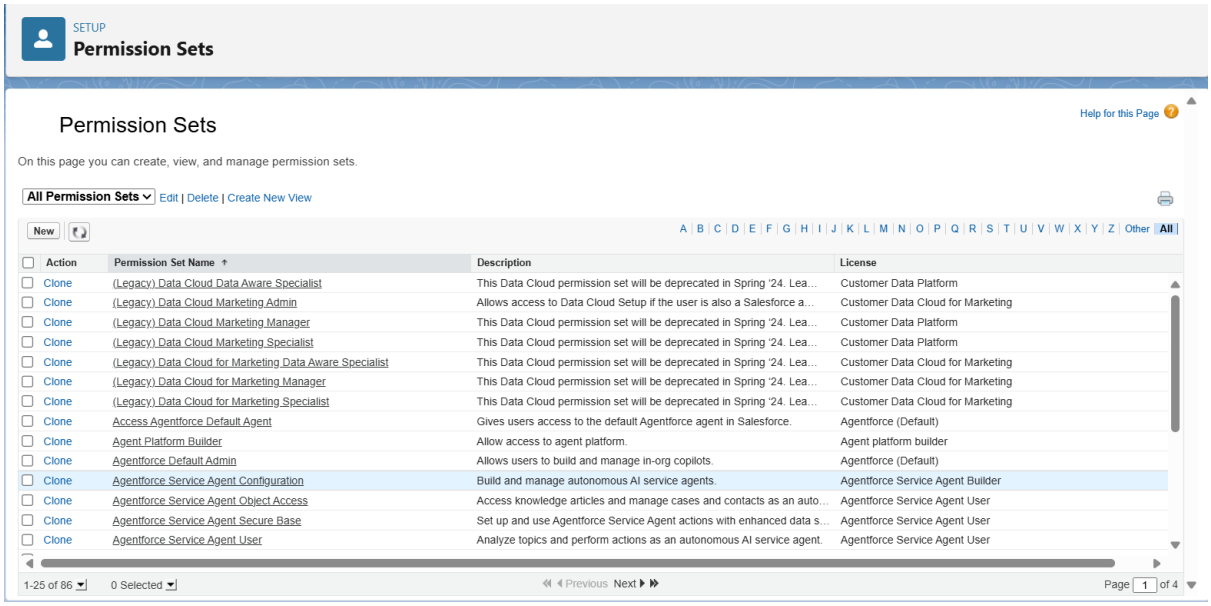
- **Screenshot:**




## 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.

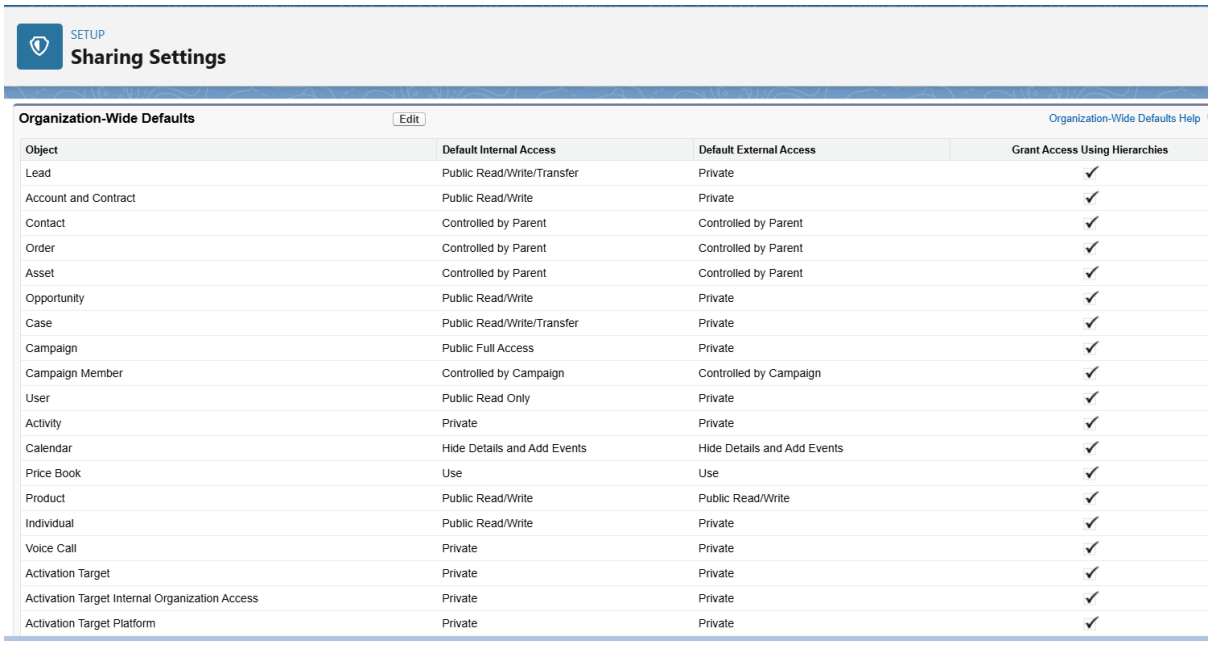
- **Screenshot:**



## 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:**



SETUP
Sharing Settings

Organization-Wide Defaults | Edit | Organization-Wide Defaults Help

| 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.

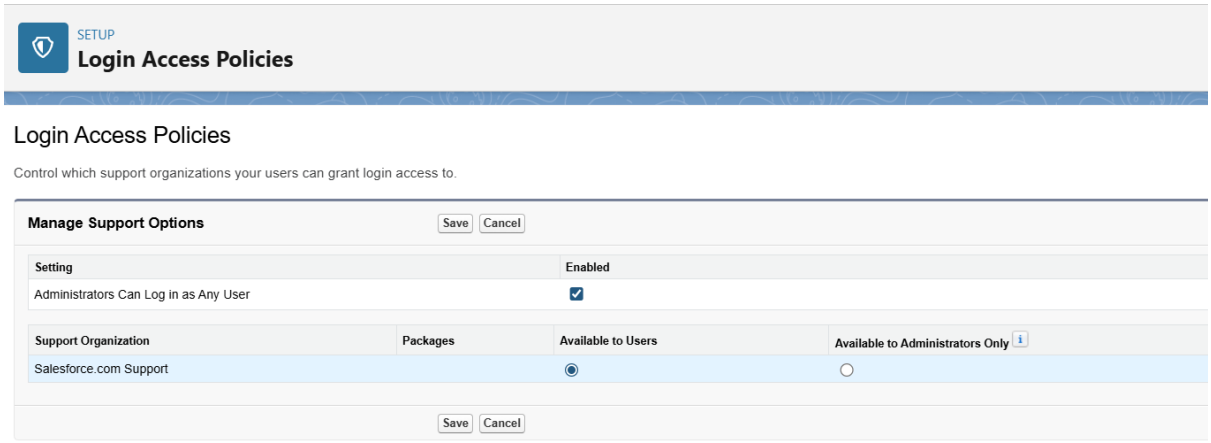
---

## Step 11: Login Access Policies

1. Setup → Login Access Policies → Enable:

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

- **Screenshot:**




---

## 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

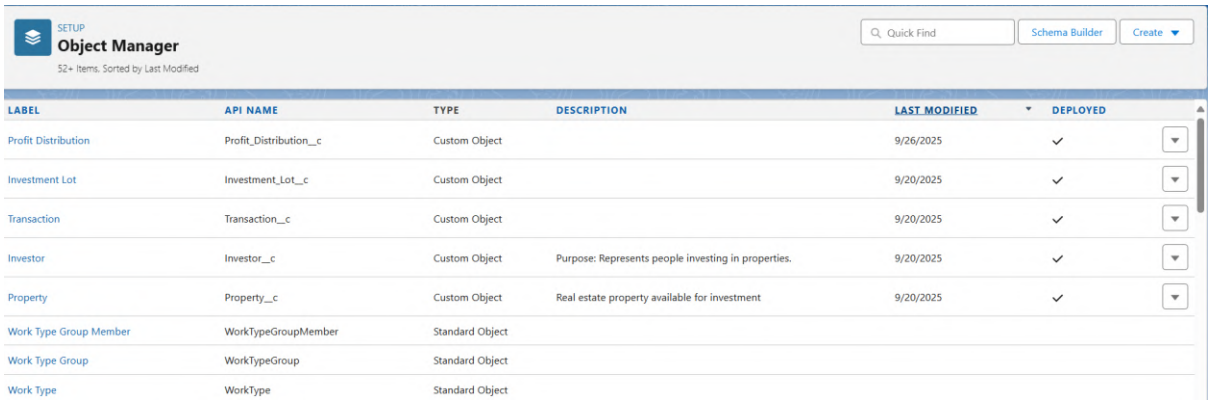
Phase 3 focuses on **defining the Salesforce data model** for the Revaa platform. This includes creation of **custom objects, fields, relationships, page layouts, record types, compact layouts, and junction objects**. Proper data modeling ensures the application can **track investors, properties, investments,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




---

## 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**

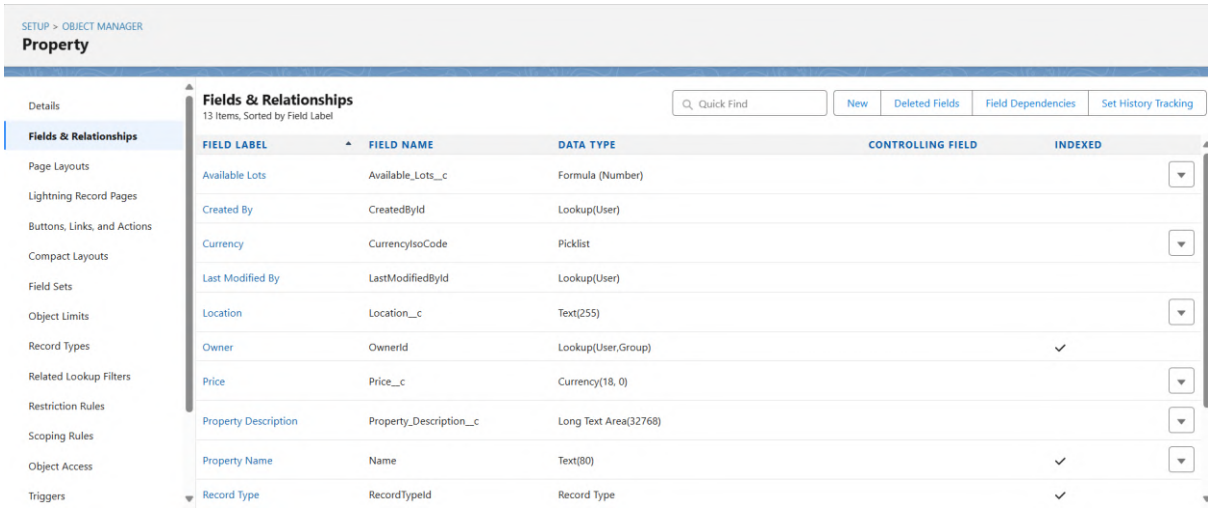
- Transaction Number (Auto Number, e.g., TX-{0001}) – Record Name
- Investor (Lookup → Investor__c)
- Property (Lookup → Property__c)
- Lots Purchased (Number)
- Investment Amount (Currency, formula: Lots_Purchased__c * (Property__r.Total_Value__c / Property__r.Total_Lots__c))
- Ownership % (Formula: (Lots_Purchased__c / Property__r.Total_Lots__c) * 100)

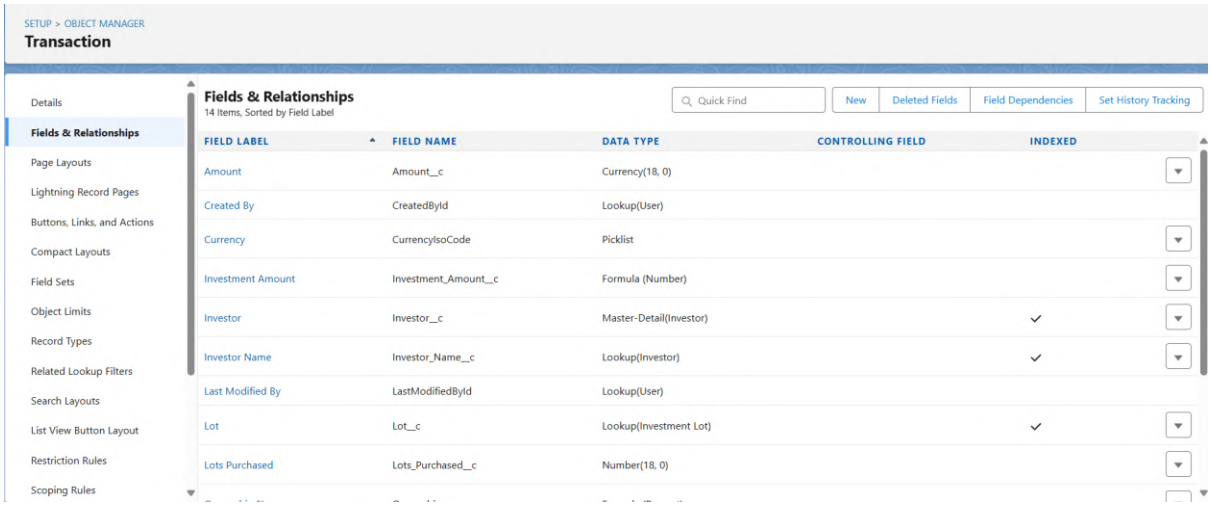
**ProfitDistribution__c**

- 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

## 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

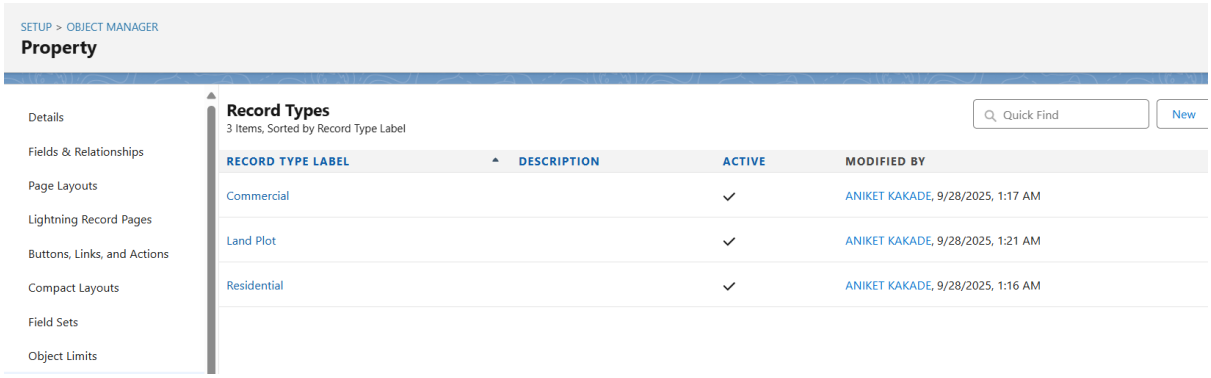
Quick Find | New | Deleted Fields | Field Dependencies | Set History Tracking

| 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) | | ✓ |

## 4. Record Types

### Record Types Created for Property Object :

- Residential
- Commercial
- Land Plot




---

## 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)

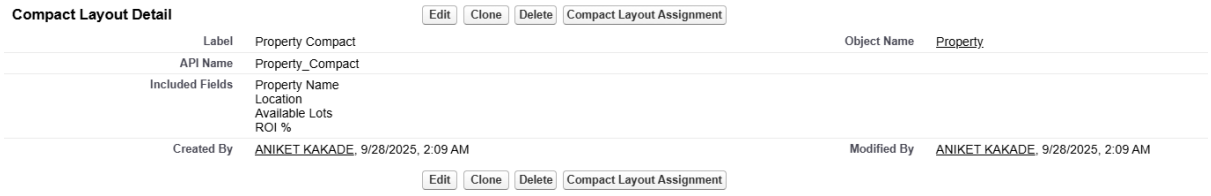
---

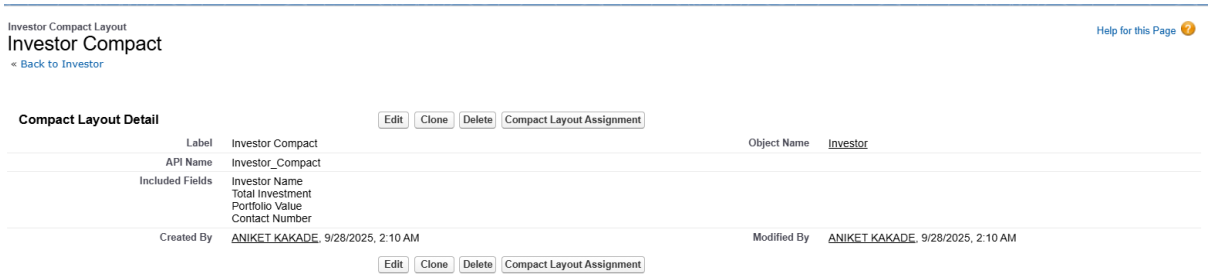
# 6. Compact Layouts

## 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 :



Transaction Compact Layout

Transactions Compact

« Back to Transaction

**Compact Layout Detail** Edit | Clone | Delete | Compact Layout Assignment

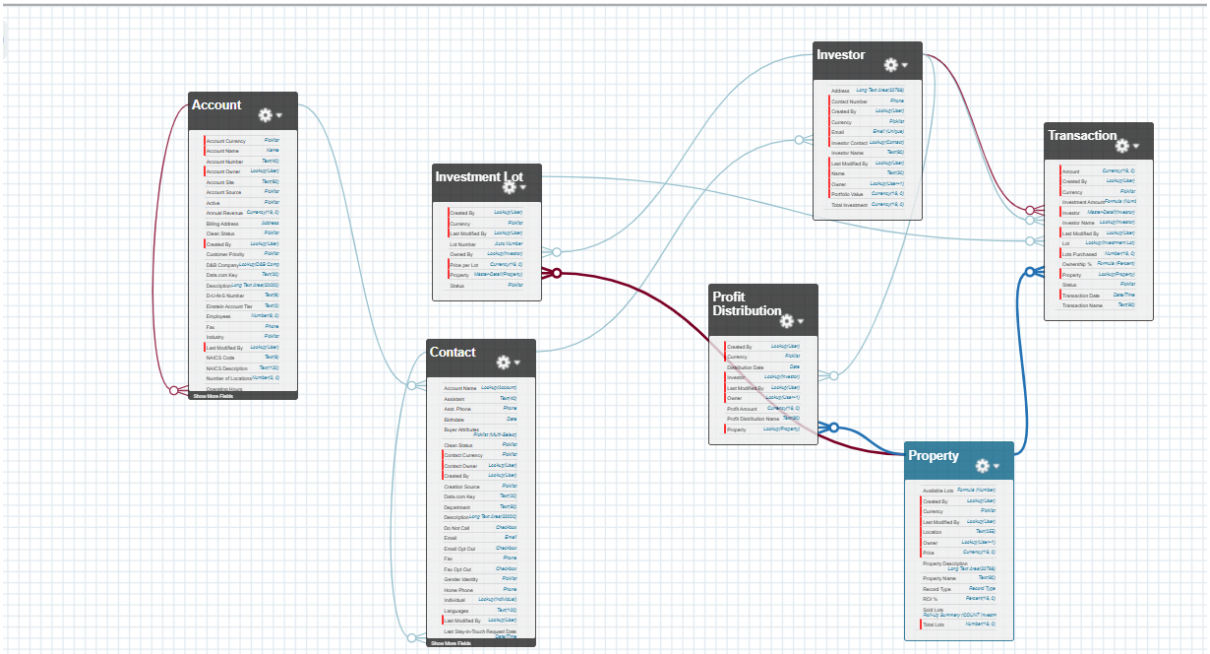
| | | | |
|---|---|---|---|
| Label | Transactions Compact | Object Name | Transaction |
| API Name | Transactions_Compact | | |
| Included Fields | Transaction Name<br>Investor<br>Property<br>Lots Purchased | | |
| Created By | ANIKET KAKADE, 9/28/2025, 2:12 AM | Modified By | ANIKET KAKADE, 9/28/2025, 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: (Lots_Purchased__c / 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

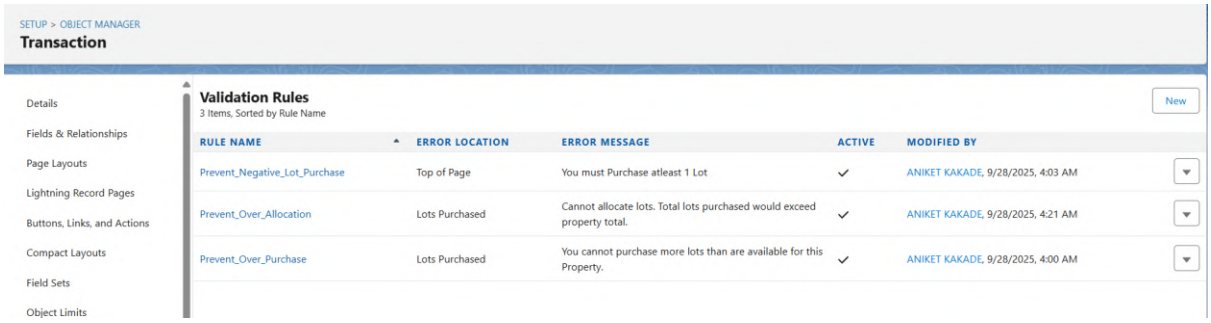
We created multiple validation rules to ensure **data integrity** across objects.

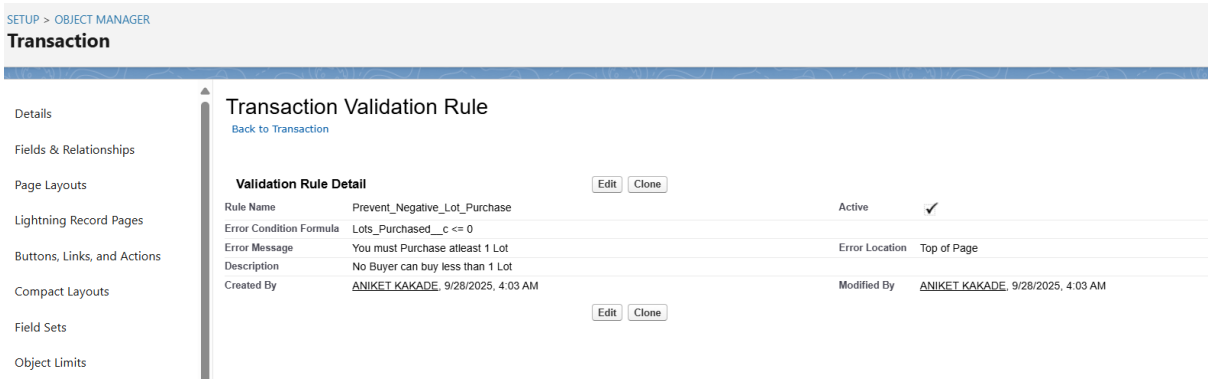
### 4.1.1 Ownership Percentage Required

- **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.



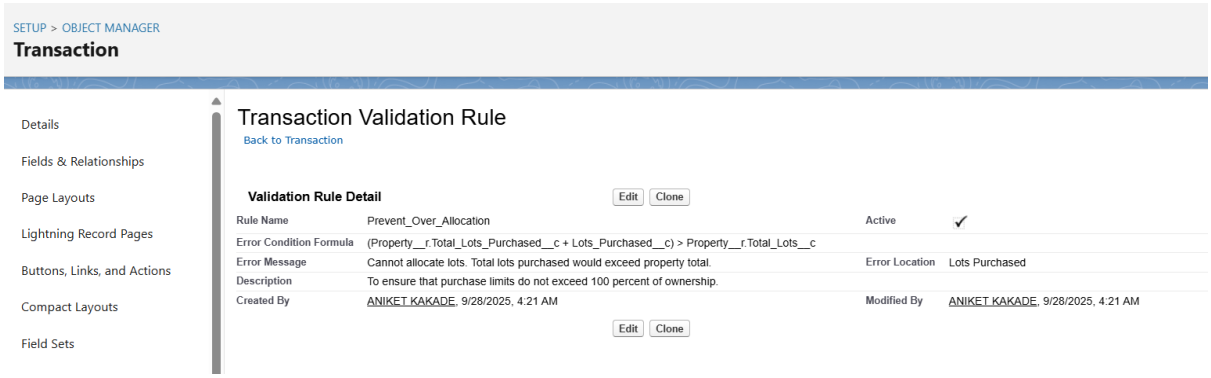

---

### 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.



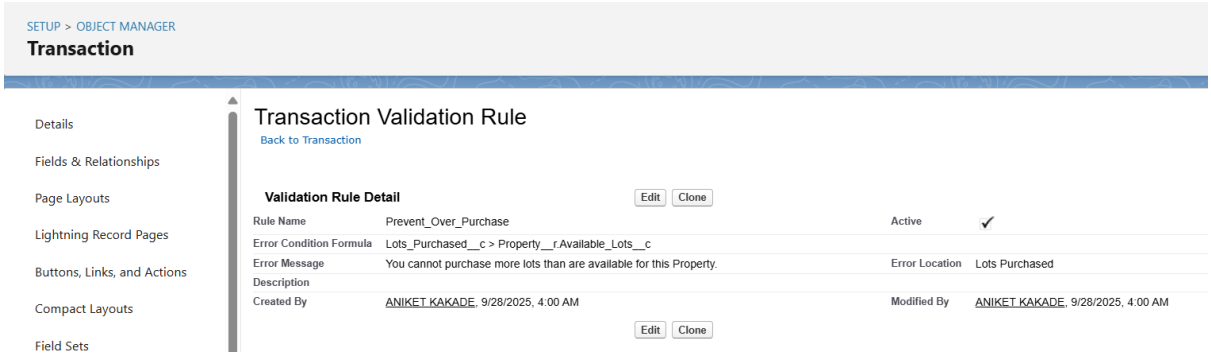

---

### 4.1.3 Profit Amount Required for Profit Distribution

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

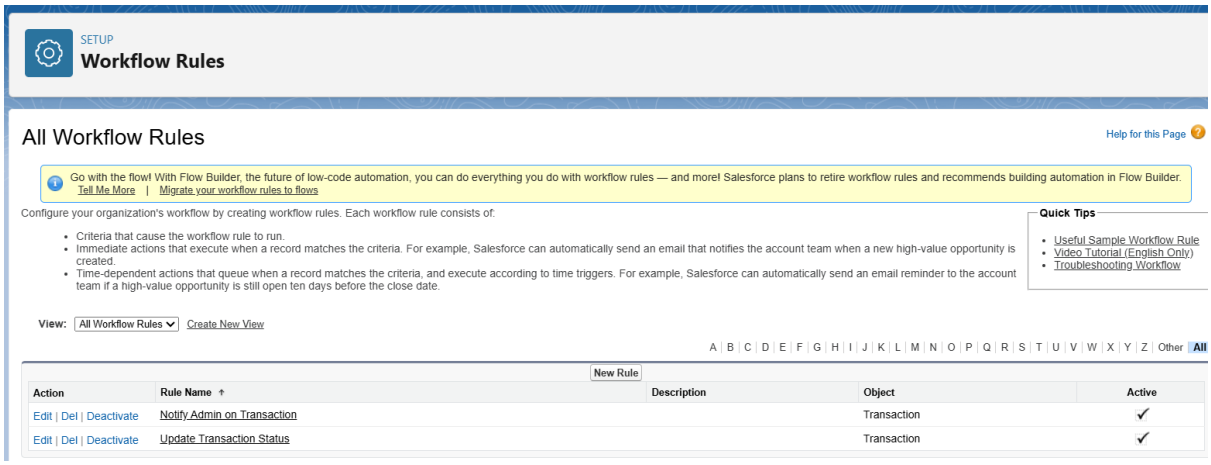
ISPICKVAL(Transaction_Type__c, "Profit Distribution") && ISBLANK(Profit_Amount__c)

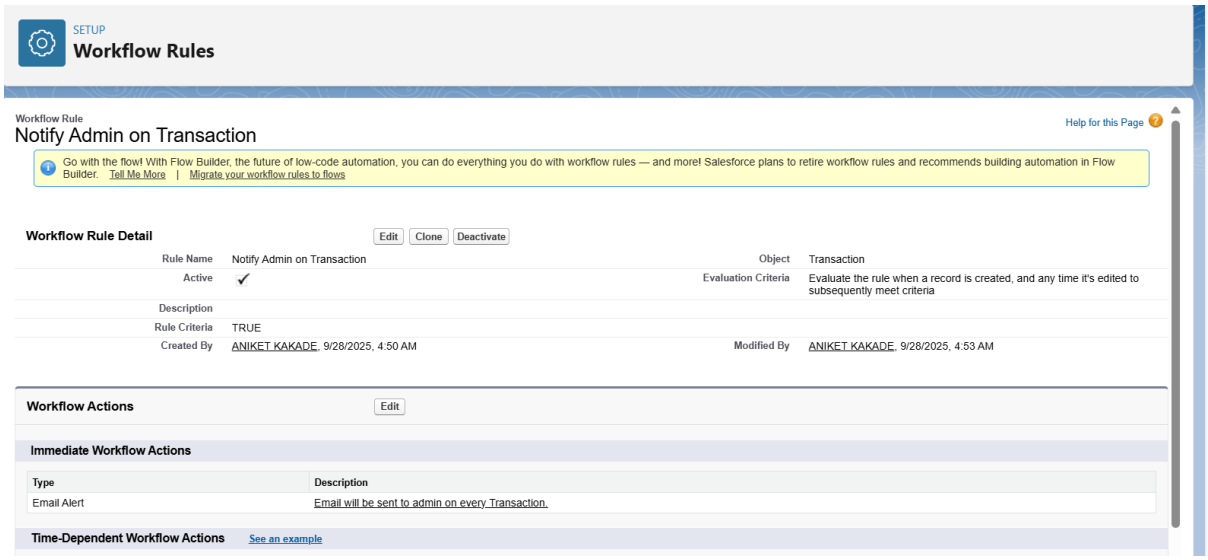
- **Error Message:** Profit Amount must be entered before distributing profit.

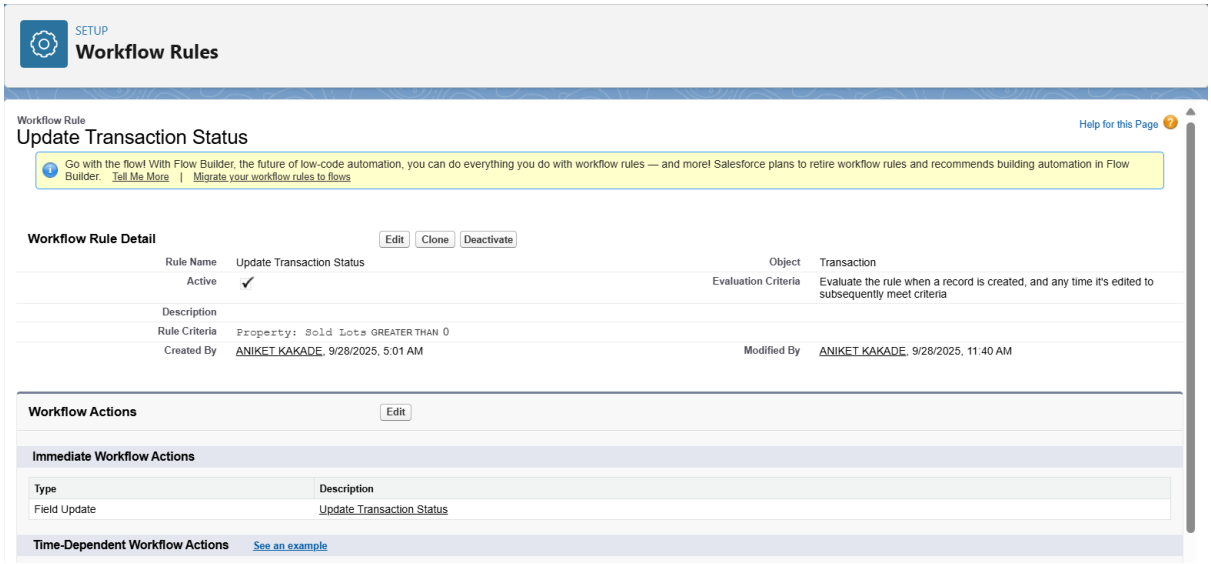



---

# 4.2 Workflow Rules



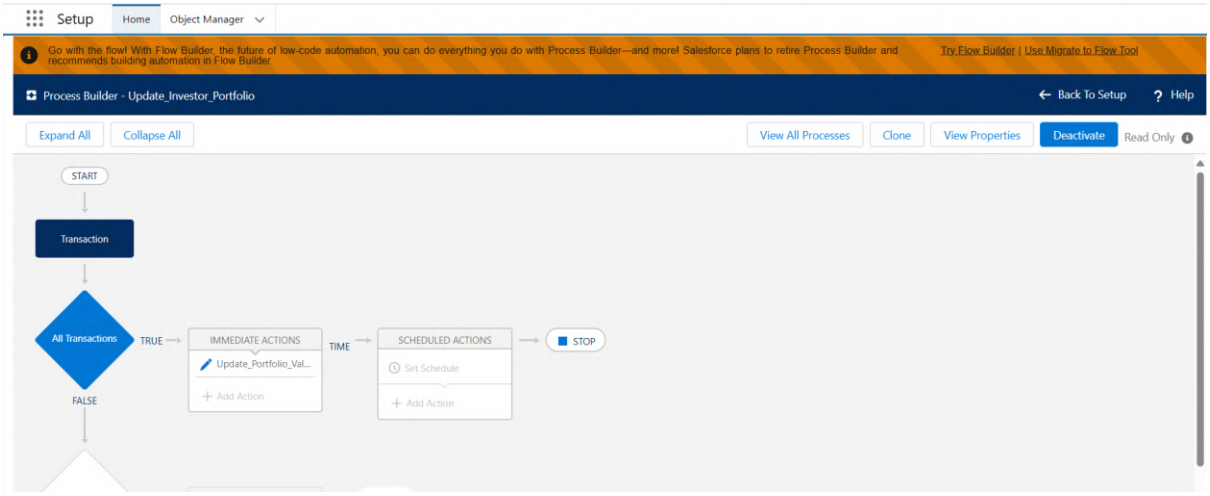
## 🔹 4.3 Process Builder

### 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



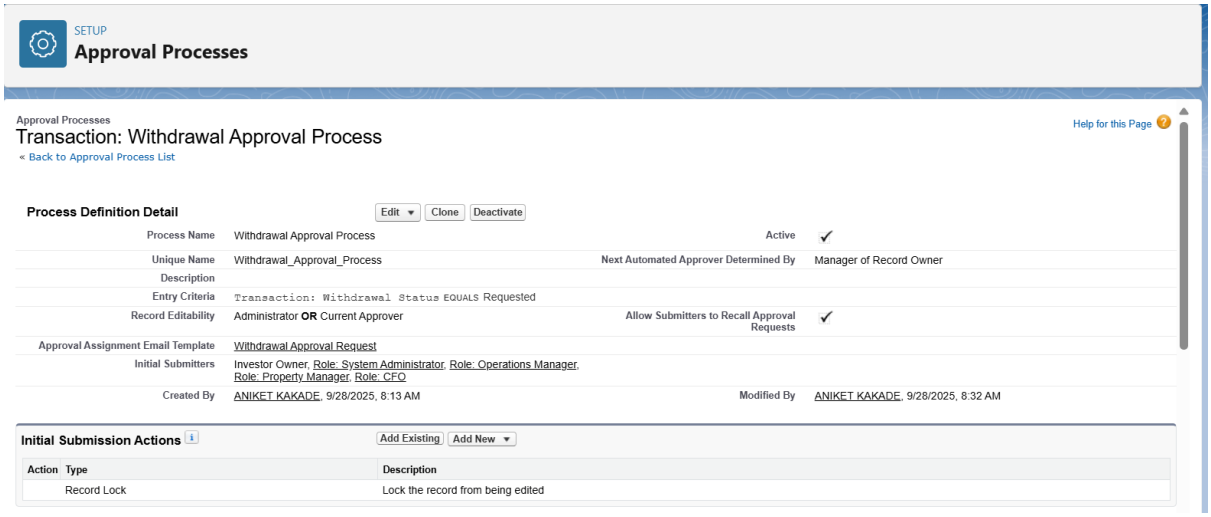

## 🔹 4.4 Approval Process

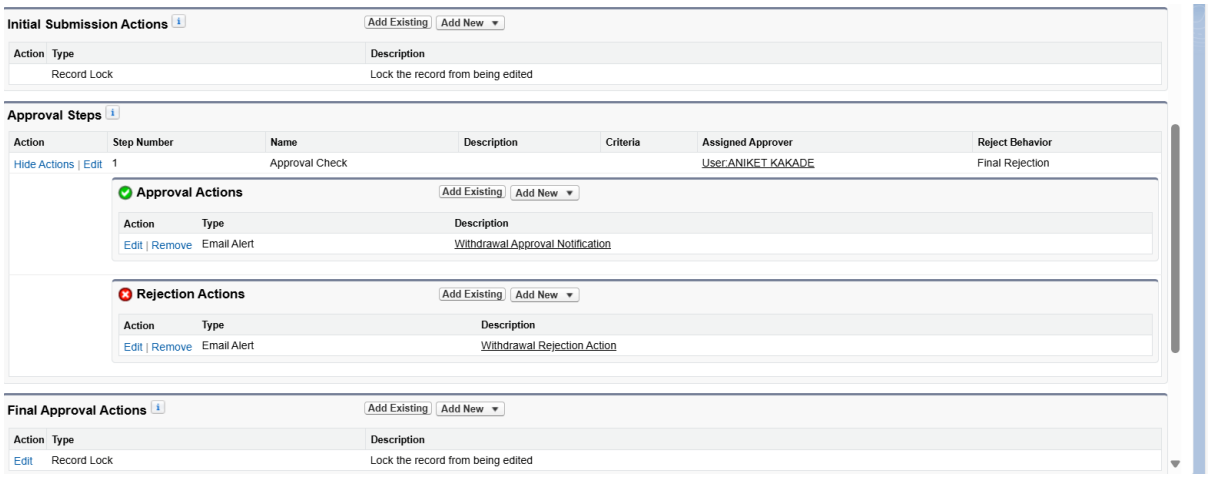
### 4.4.1 Withdrawal Approval Workflow

- **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
« Back to Approval Process List
Help for this Page
Process Definition Detail
Edit
Clone
Deactivate
Process Name
Withdrawal Approval Process
Active
Unique Name
Withdrawal_Approval_Process
Next Automated Approver Determined By
Manager of Record Owner
Description
Entry Criteria
Transaction: Withdrawal Status EQUALS Requested
Record Editability
Administrator OR Current Approver
Allow Submitters to Recall Approval Requests
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

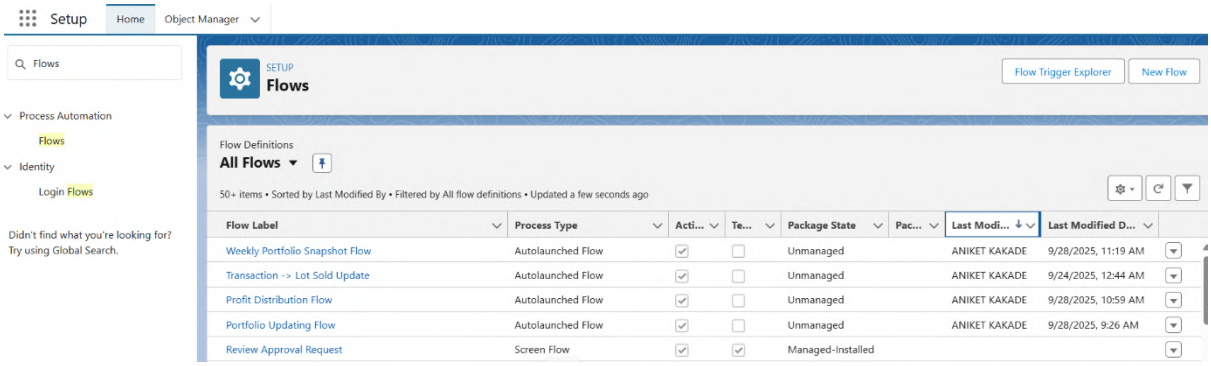

Initial Submission Actions
Add Existing
Add New
Action
Type
Description
Record Lock
Lock the record from being edited
Approval Steps
Action
Step Number
Name
Description
Criteria
Assigned Approver
Reject Behavior
Hide Actions | Edit
1
Approval Check
User:ANIKET KAKADE
Final Rejection
Approval Actions
Add Existing
Add New
Action
Type
Description
Edit | Remove
Email Alert
Withdrawal Approval Notification
Rejection Actions
Add Existing
Add New
Action
Type
Description
Edit | Remove
Email Alert
Withdrawal Rejection Action
Final Approval Actions
Add Existing
Add New
Action
Type
Description
Edit
Record Lock
Lock the record from being edited

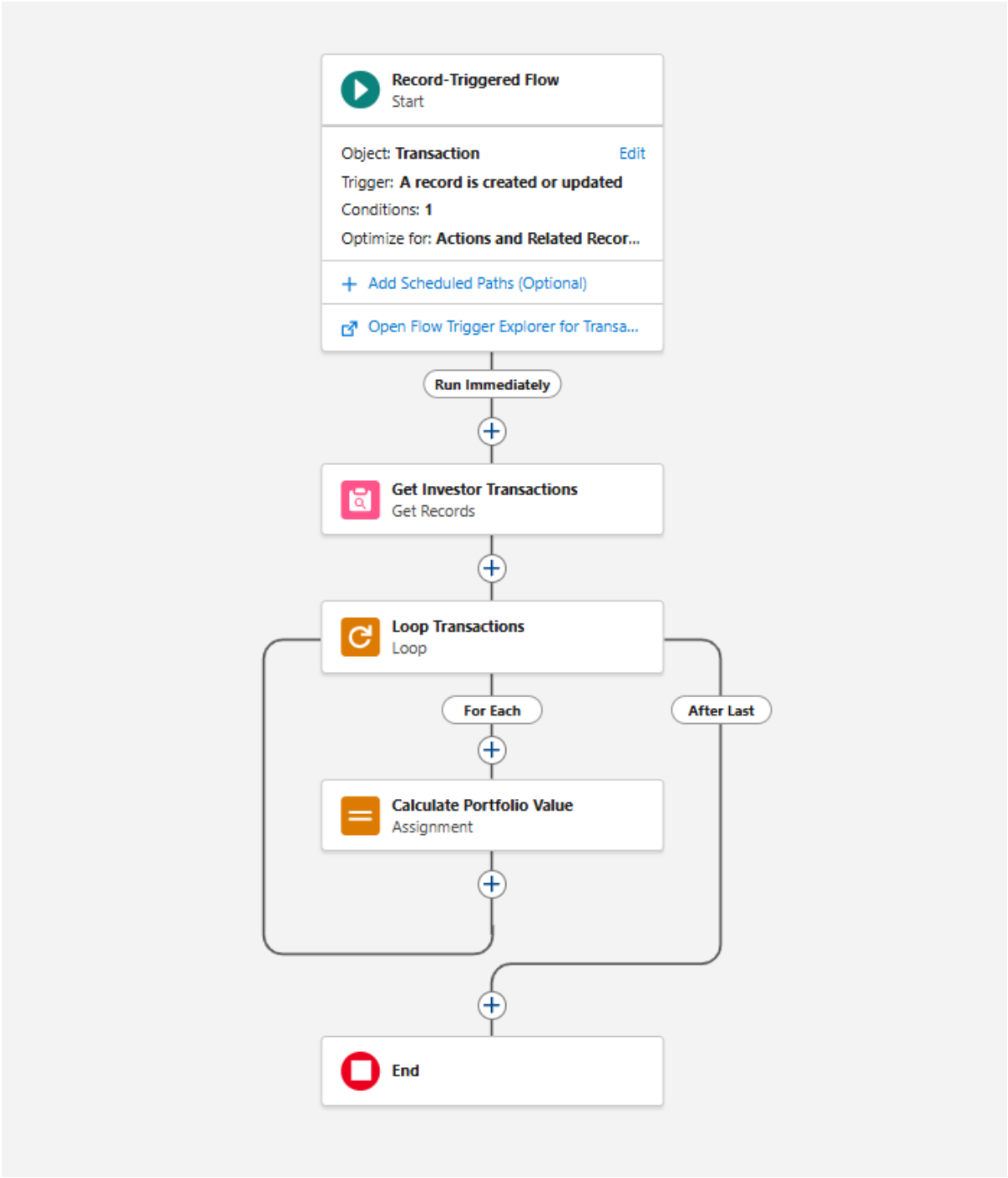
## 🔹 4.5 Record-Triggered Flow

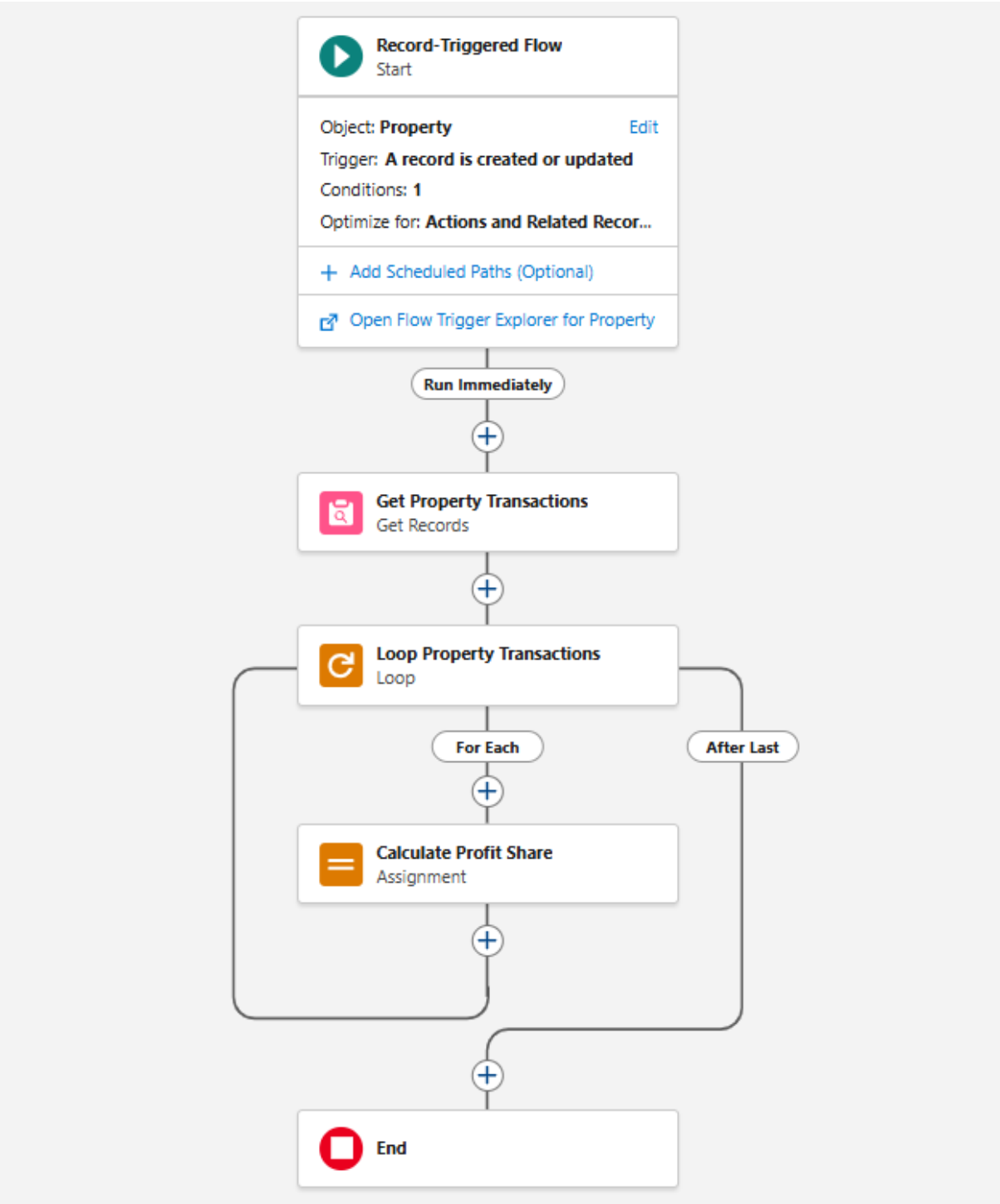
### 4.5.1 Profit Distribution Flow

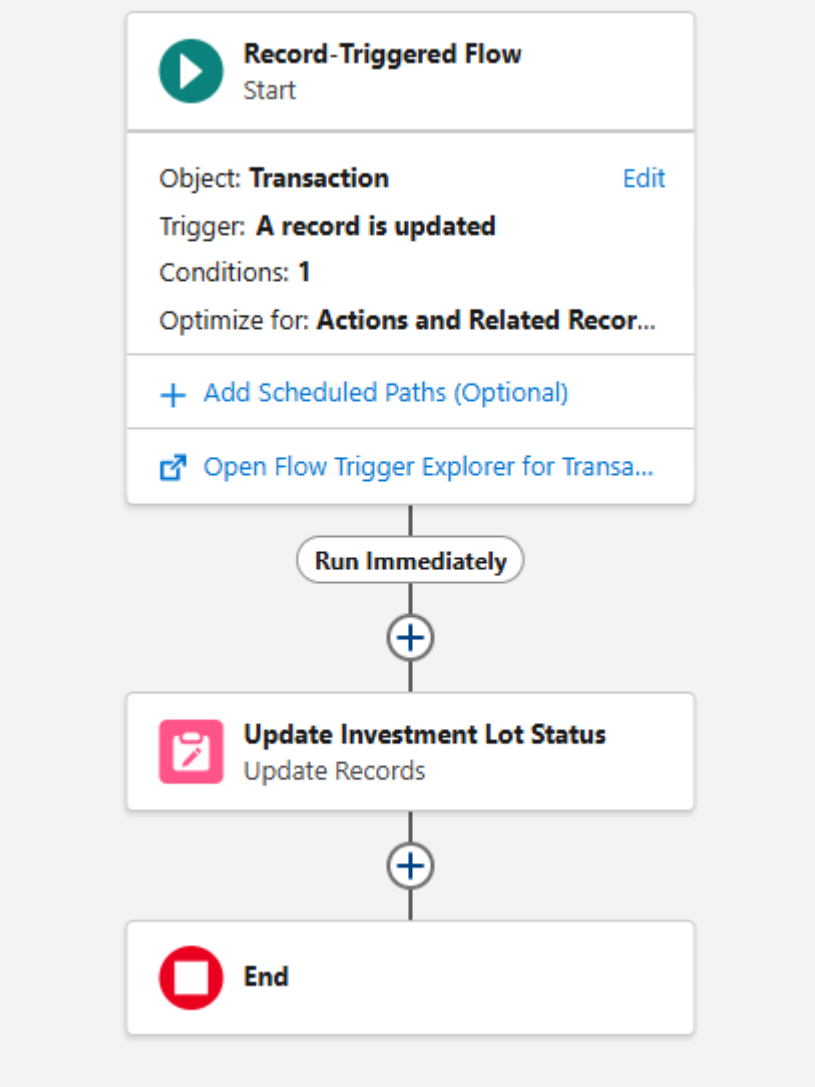
- **Flow Name:** Profit_Distribution_Flow
- **Trigger:** On Property__c (when Profit_Amount__c is updated)
- **Steps:**

1. **Get Records:** Fetch all Transaction__c linked to Property.
2. **Loop:** Loop over each Transaction.
3. **Assignment:**
4. {!Loop_Transactions.Profit_Share__c} =
5. ({!Loop_Transactions.Ownership_Percent__c} * {!$Record.Profit_Amount__c}) / 100
6. **Update Records:** Update each Transaction with Profit_Share__c.




Record-Triggered Flow
Start
Object: Transaction
Edit
Trigger: A record is created or updated
Conditions: 1
Optimize for: Actions and Related Recor...
Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Transa...
Run Immediately
Get Investor Transactions
Get Records
Loop Transactions
Loop
For Each
After Last
Calculate Portfolio Value
Assignment
End


Record-Triggered Flow
Start
Object: Property
Edit
Trigger: A record is created or updated
Conditions: 1
Optimize for: Actions and Related Recor...
Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Property
Run Immediately
Get Property Transactions
Get Records
Loop Property Transactions
Loop
For Each
After Last
Calculate Profit Share
Assignment
End



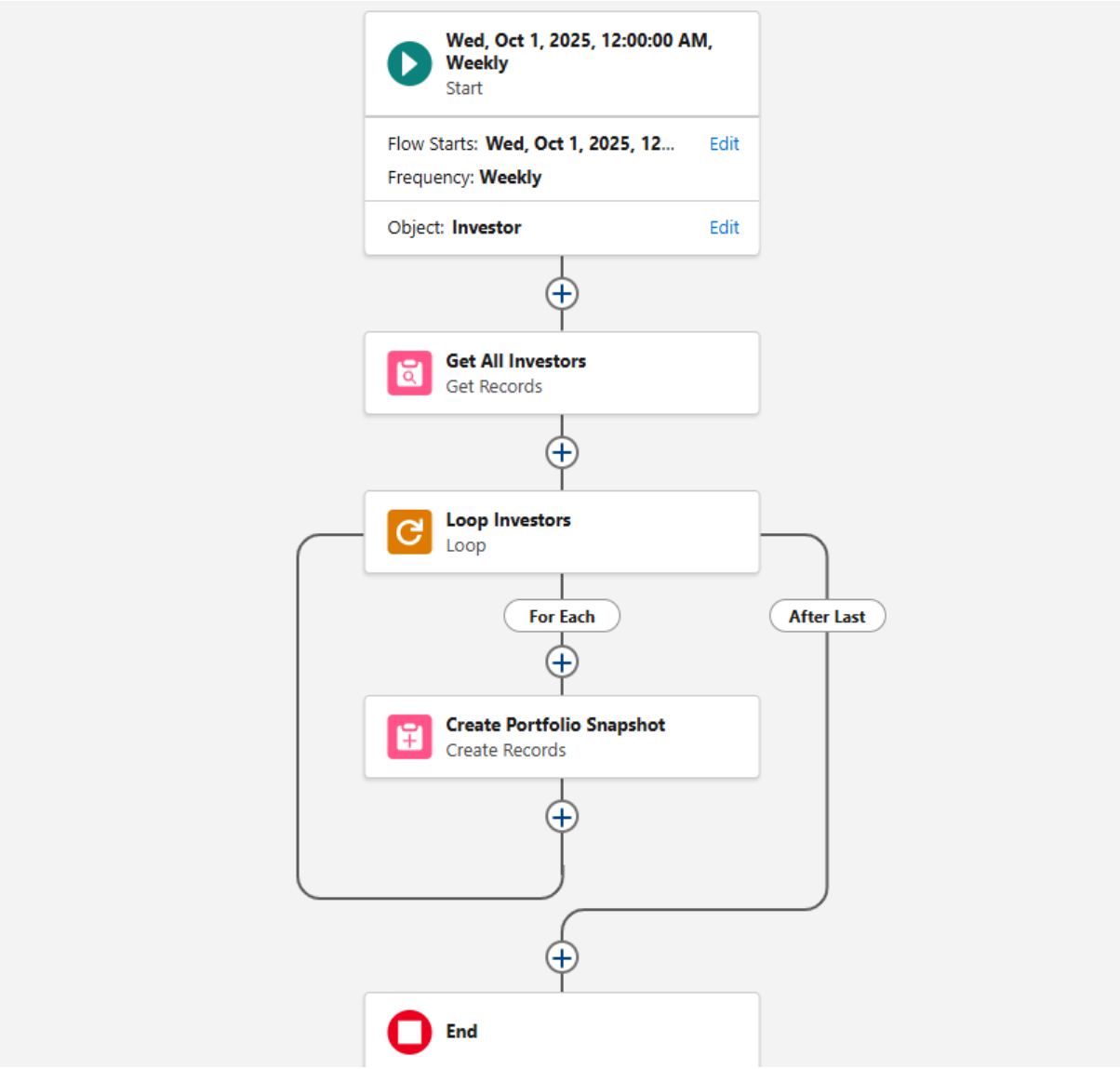

---

## 🔹 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}




---

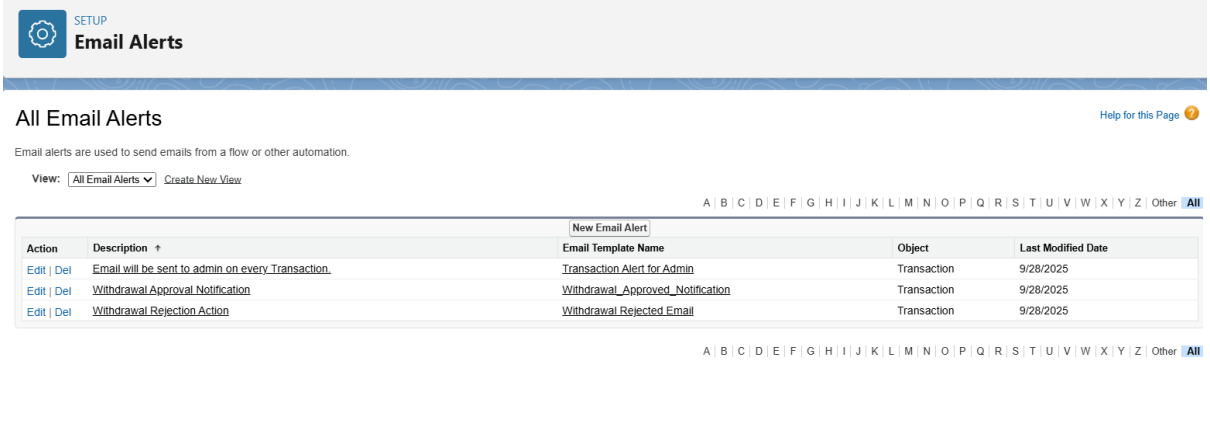
## 🔹 4.7 Email Alerts

### 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.




---

## ✅ 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 **Revaa**.

# 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.

---

## 1️⃣ Classes & Objects

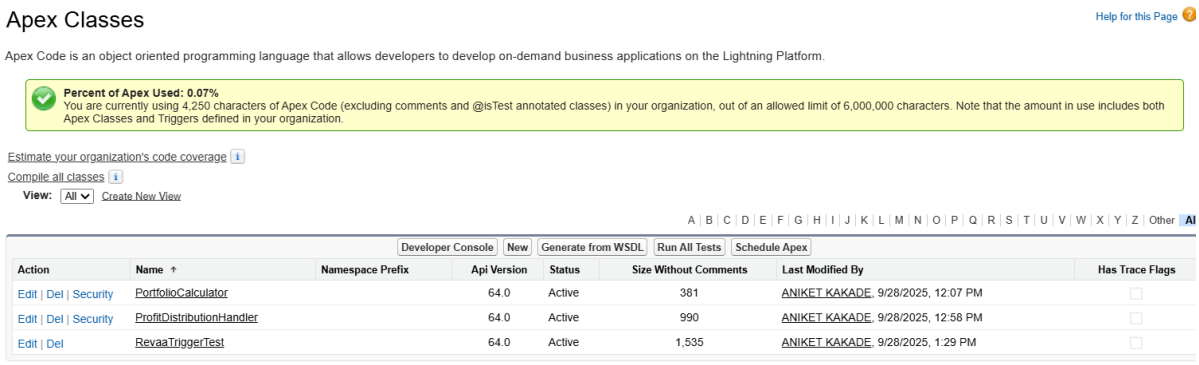
### Custom Objects used in Apex:

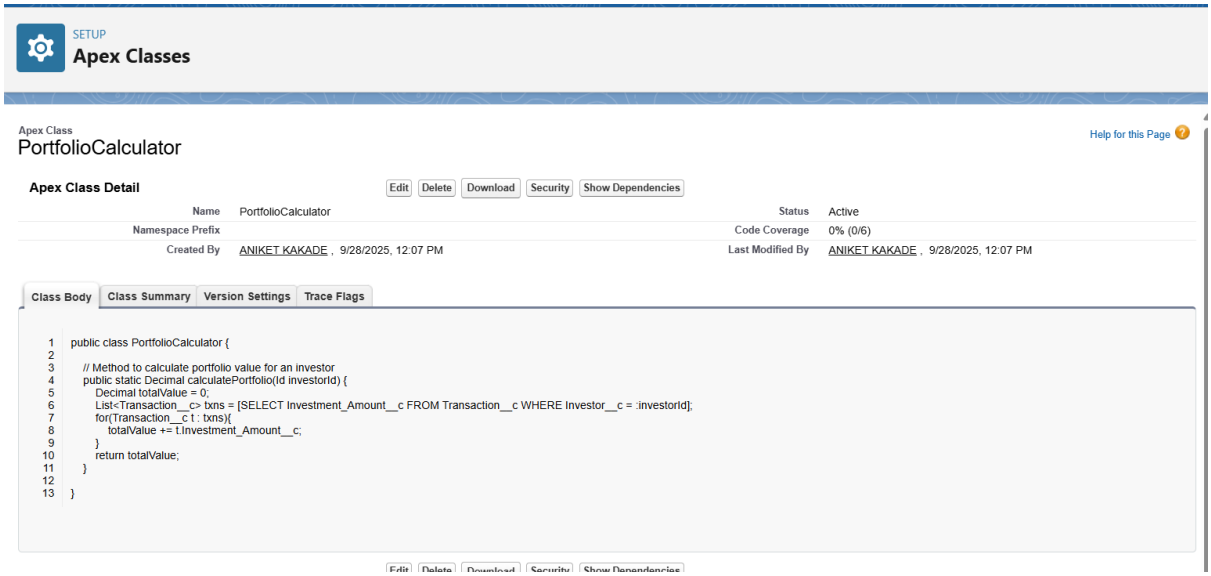
- **Property__c** – Represents real estate properties.
- **Investor__c** – Represents investors.
- **Transaction__c** – Tracks investments made by investors.
- **Investment_Lot__c** – Tracks individual lots purchased per transaction.

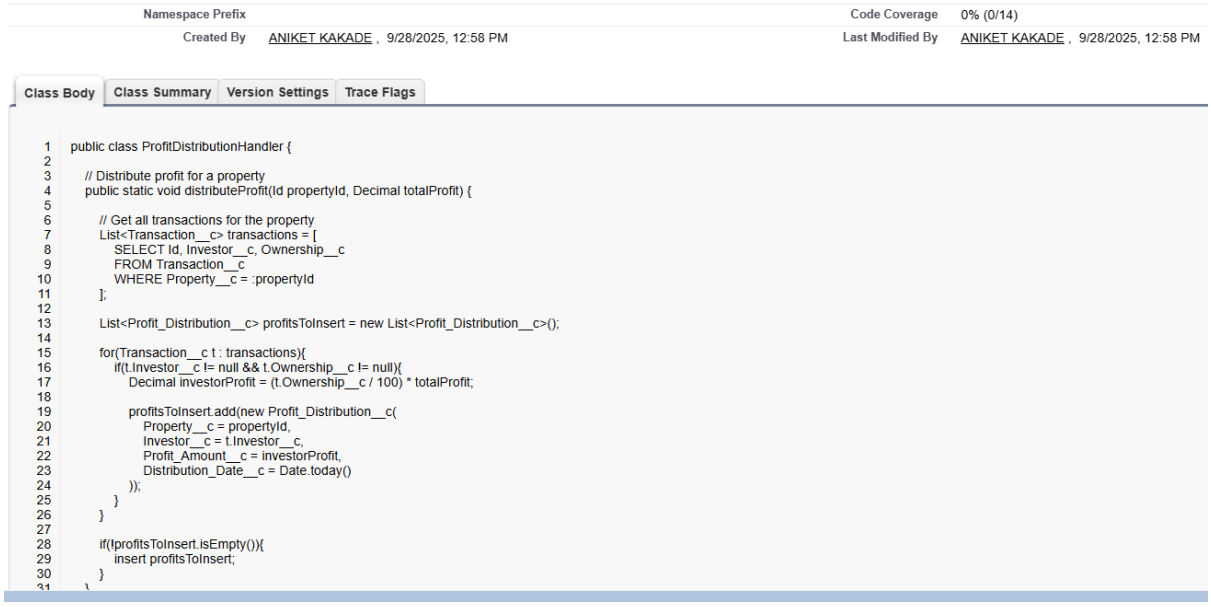
### Apex Classes created:

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.

---






---

## 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.

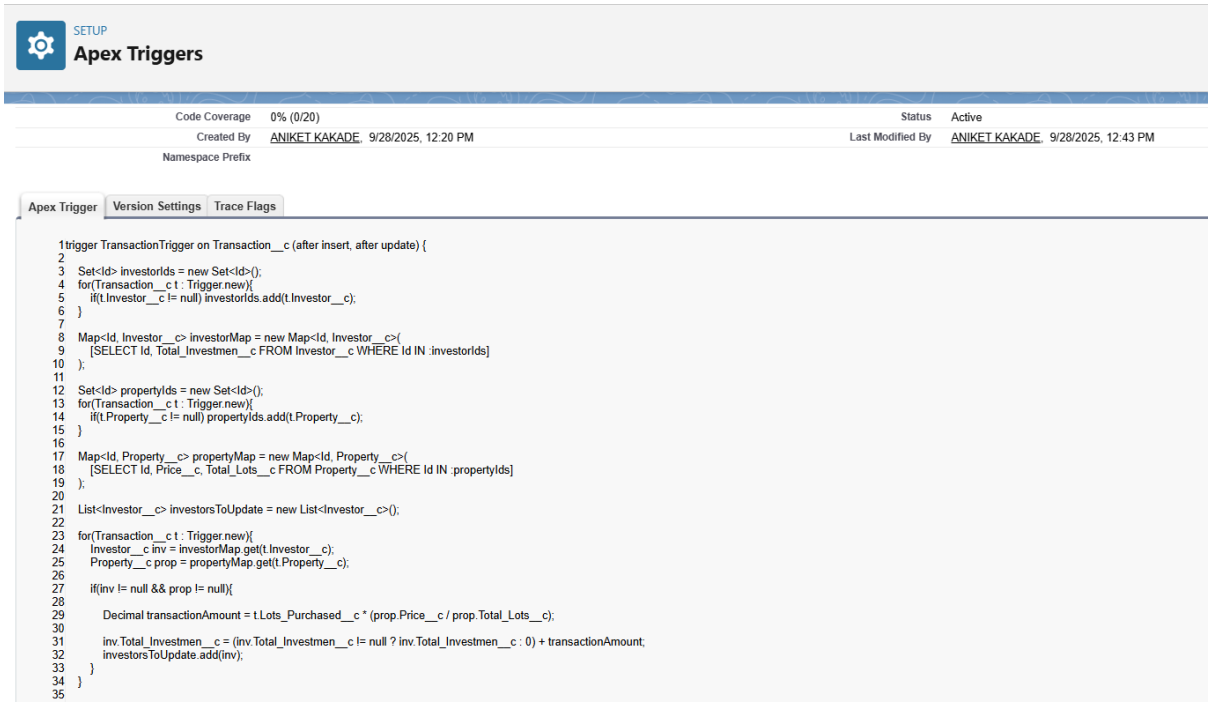
**Trigger Code:**

```
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;
```

}




---

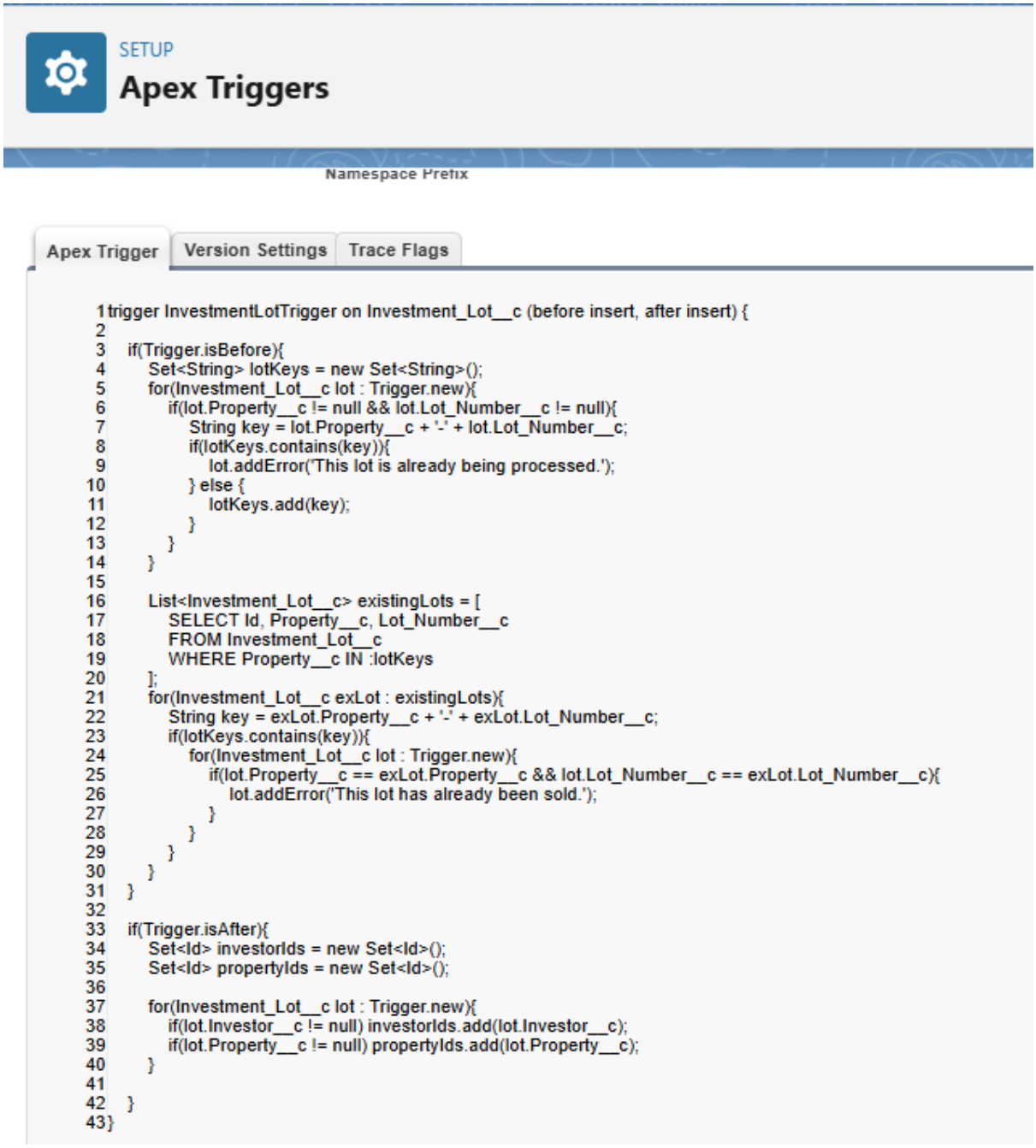
### Trigger 2: InvestmentLotTrigger

- **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:**

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

**Screenshot:**




---

## 3 Trigger Design Pattern

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

**Screenshot:** *(Handler class structure showing methods for triggers)*

---

## 4️⃣ 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)*

---

## 5️⃣ 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.

---

## 6️⃣ Control Statements

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

---

## 7️⃣ 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.

---

## 8️⃣ 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

```
@IsTest
public class RevaaTriggerTest {

    @IsTest
    static void testTransactionTrigger() {
        Property__c prop = new Property__c(
            Name = 'Test Prop',
            Price__c = 1000000,
            Total_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_Investmen__c FROM Investor__c WHERE Id = :inv.Id];

        Decimal expectedInvestment = 10 * (prop.Price__c / prop.Total_Lots__c);
        System.assertEquals(expectedInvestment, inv.Total_Investmen__c, 'Investor Total Investment should be updated correctly.');
    }

    @IsTest
    static void testInvestmentLotTrigger() {
        Property__c prop = new Property__c(
            Name = 'Lot Test',
            Price__c = 500000,
            Total_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;

        Integer lotCount = [SELECT COUNT() FROM Investment_Lot__c WHERE Property__c = :prop.Id];
        System.assertEquals(1, lotCount, 'One lot should be linked to the property.');
    }
```

---

## Expected Outcomes for Phase 5

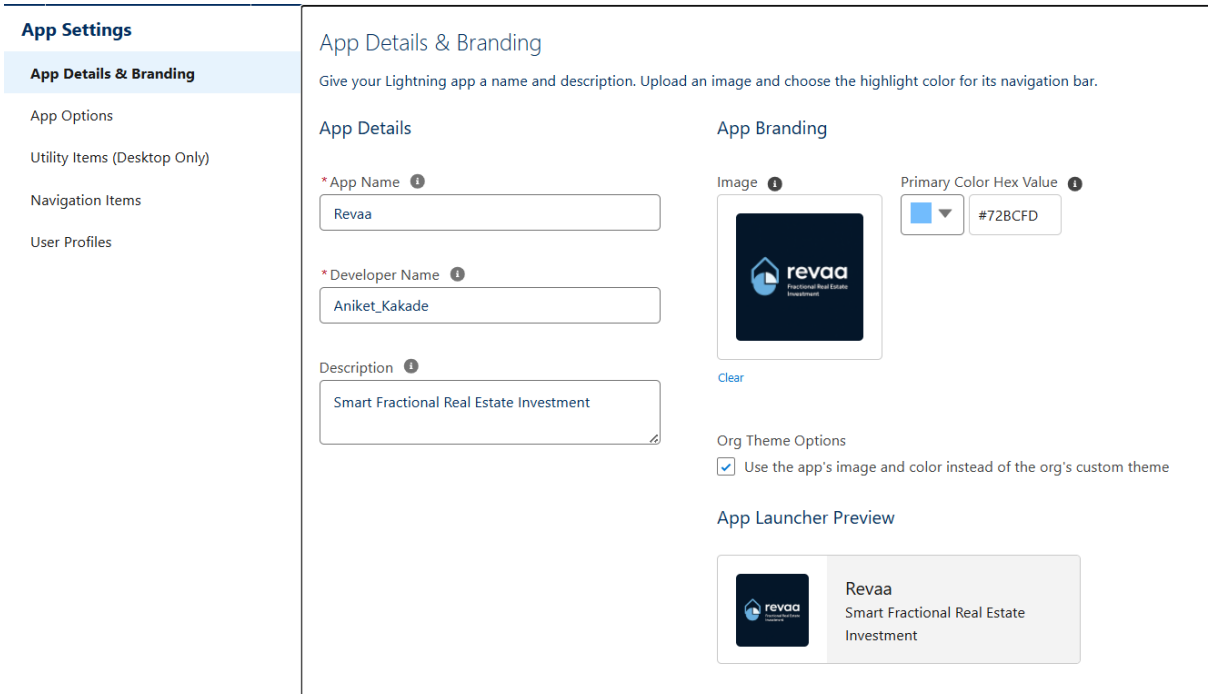
1. Investor portfolio updates automatically on every new transaction.
2. Investment lots are correctly linked and managed, preventing duplicates.
3. Bulk-safe triggers and Apex classes ensure Revaa can handle multiple investors and transactions simultaneously.
4. Test coverage > 75%, ready for deployment.

# 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.

---




## 1️⃣ Lightning App Builder

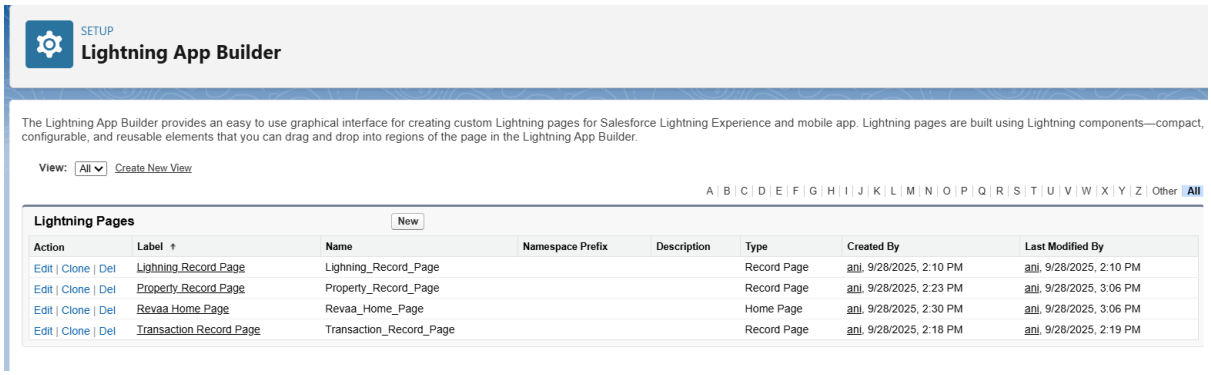
**Purpose:** Create apps, pages, and customize layouts to match user roles and workflows.

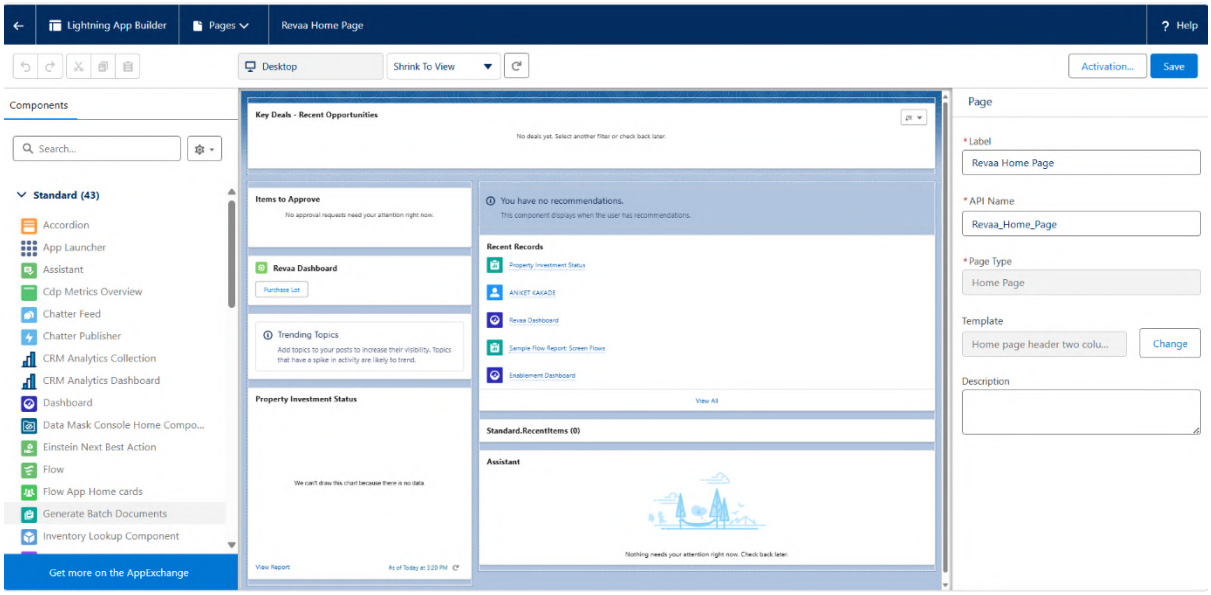
**Steps:**

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:**




---

## 2 Record Pages

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

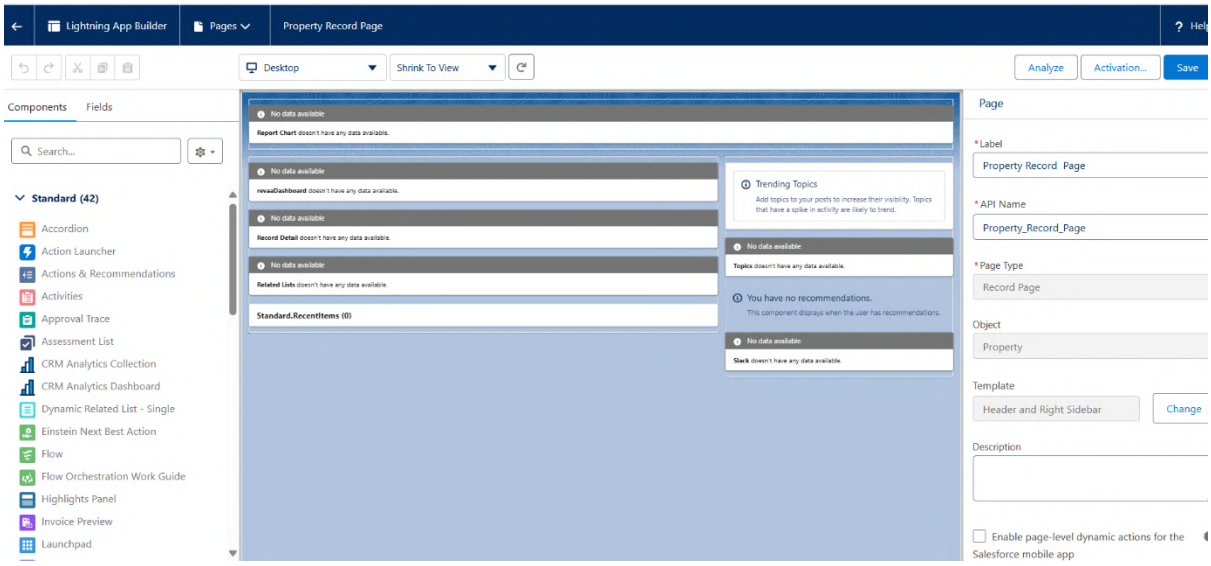
**Steps:**

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:**




---

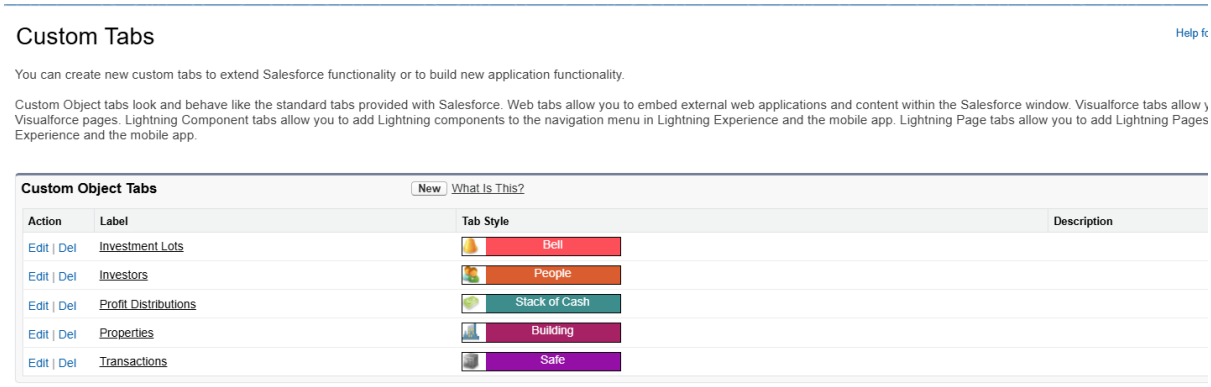
## 3 Tabs

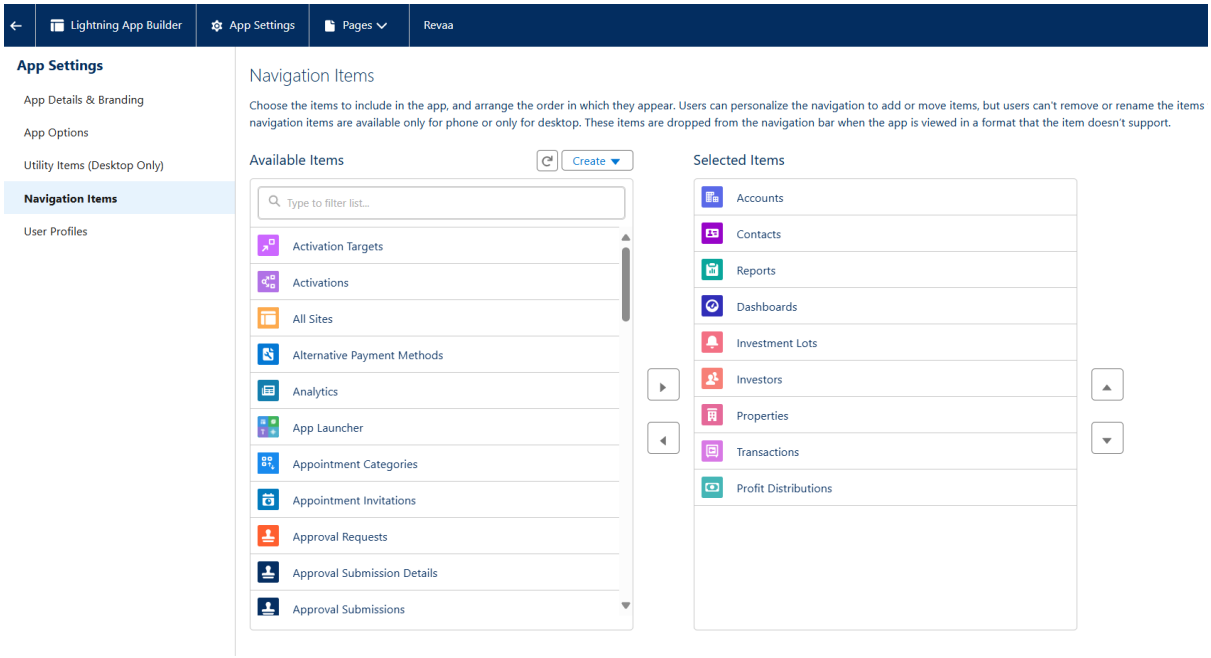
**Purpose:** Make navigation easier across custom objects and reports.

**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:**




## 4 Home Page Layouts

**Purpose:** Provide quick insights, metrics, and shortcuts.

**Steps:**

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

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

---

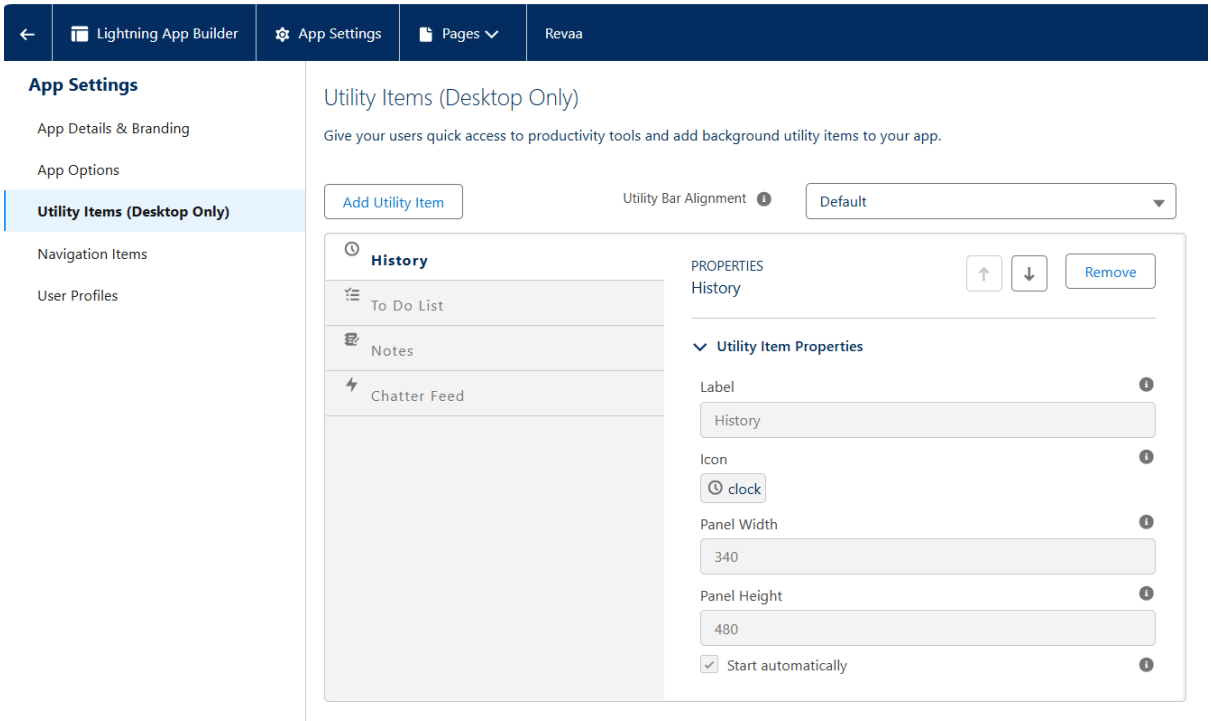
## 5 Utility Bar

**Purpose:** Add quick actions accessible from any page.

**Steps:**

1. Setup → **App Manager → Revaa 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:**



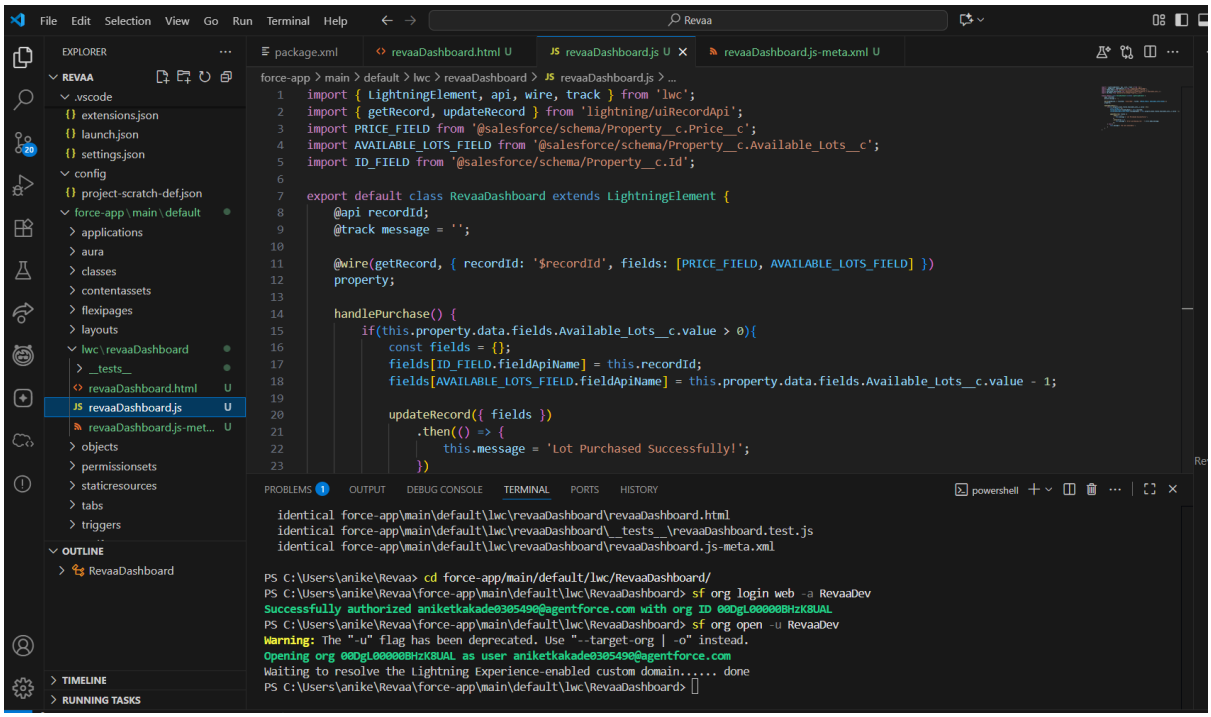
## 6️⃣ Lightning Web Components (LWC)

**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:
    - RevaaDashboard.html → UI template
    - RevaaDashboard.js → Logic, Apex calls, and data handling
    - 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:
    - Right-click component folder → **SFDX: Deploy Source to Org**
11. Add LWC to record page or app page via **Lightning App Builder**.

**Screenshot:**




---

## 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}

---

## 8️⃣ 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}

---

## 9️⃣ 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}

---

## 1️⃣0️⃣ 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.

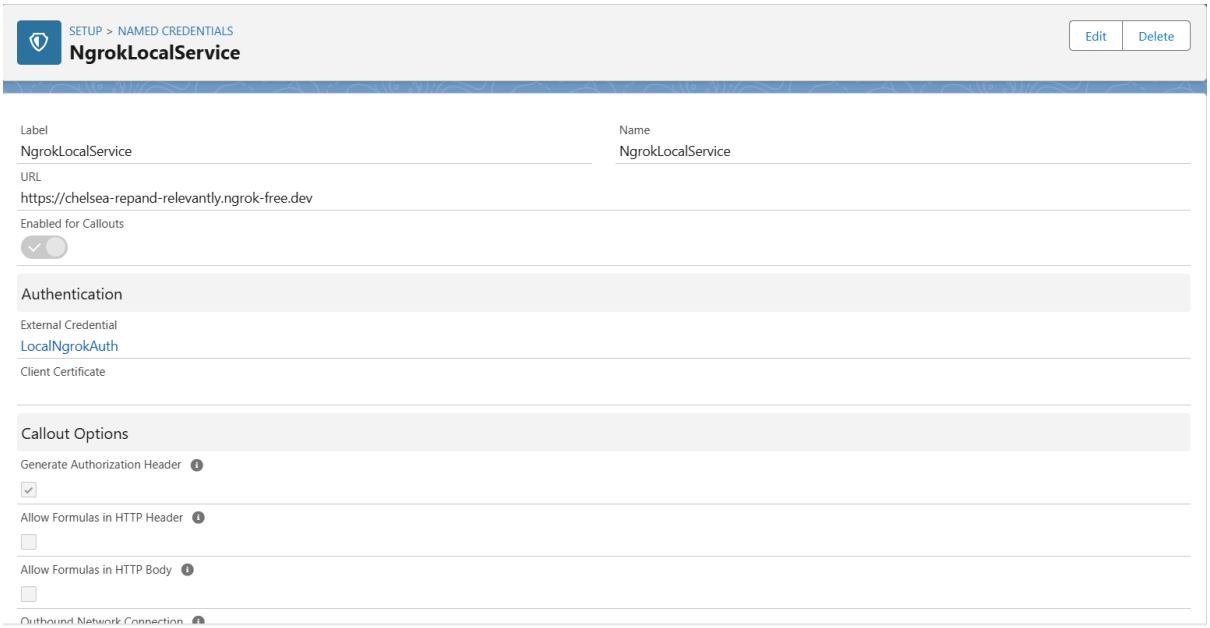
---

## 1️⃣ 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:
    - **Label:** RevaaLocalEndpoint
    - **Name:** RevaaLocalEndpoint
    - **URL:** https://<ngrok-generated-url> *(Example: https://chelsea-repand-relevantly.ngrok-free.dev)*
    - **Identity Type:** Anonymous or Named Principal (depending on authentication needs)
    - **Authentication Protocol:** Choose as per your external service (e.g., OAuth 2.0 or Password Authentication)
4. Click **Save**.




**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.

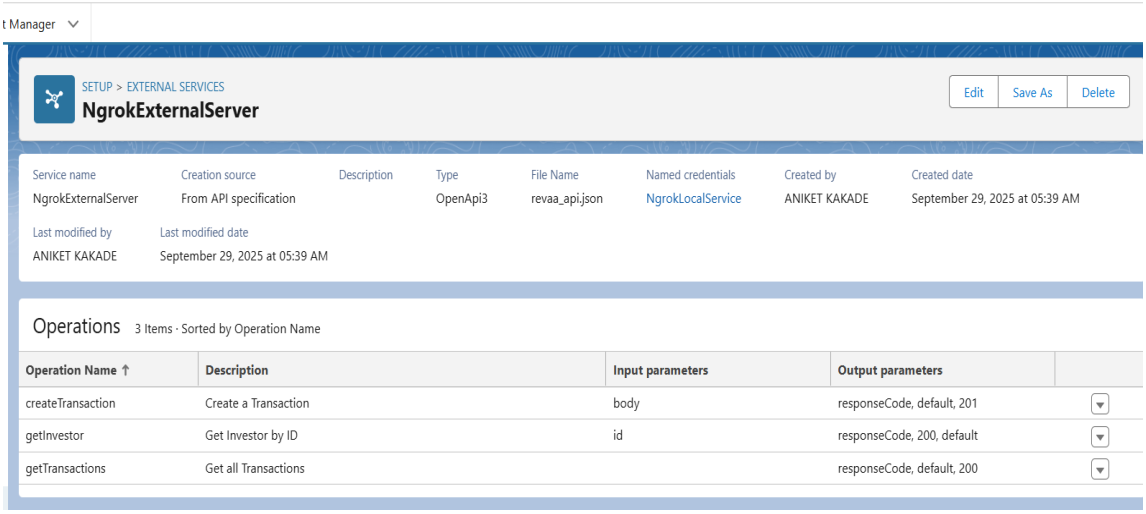
---

## 2️⃣ 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**.




**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());
    }
```


**Critical Notes:**

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

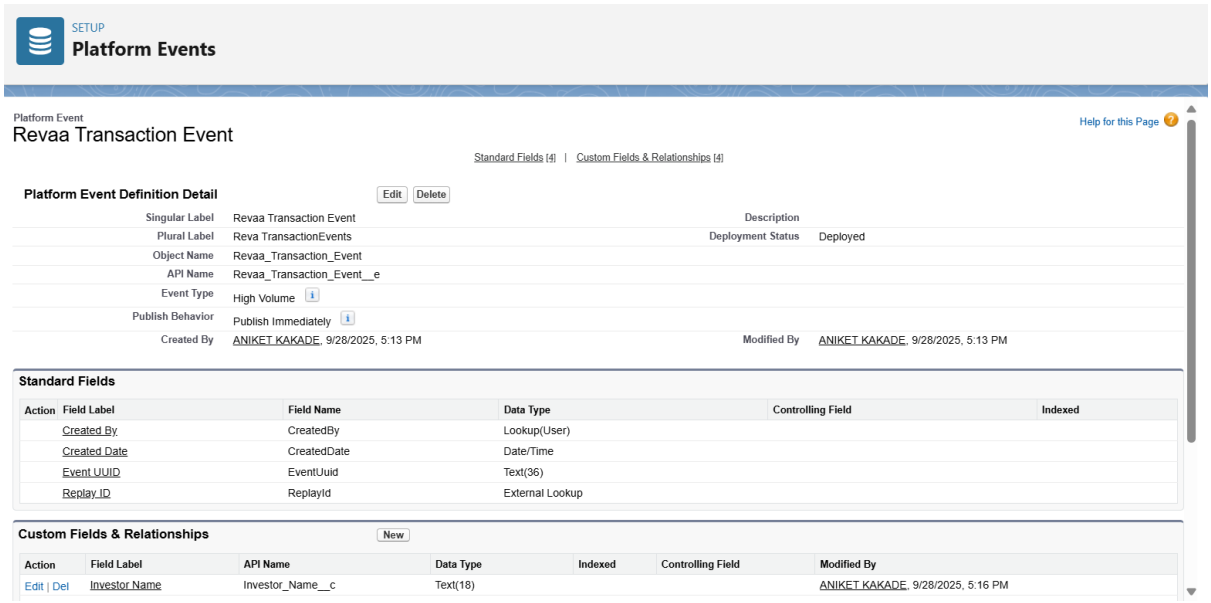
---

## 4️⃣ 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:
    - **Label:** Transaction_Approved
    - **API Name:** Transaction_Approved__e
    - **Fields:** Investor__c (Lookup), Transaction__c (Lookup), Amount__c (Number)
3. Click **Save**.
   **Use Case:** Trigger updates or notifications when a transaction is approved.




---

## 5️⃣ Change Data Capture (CDC)

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

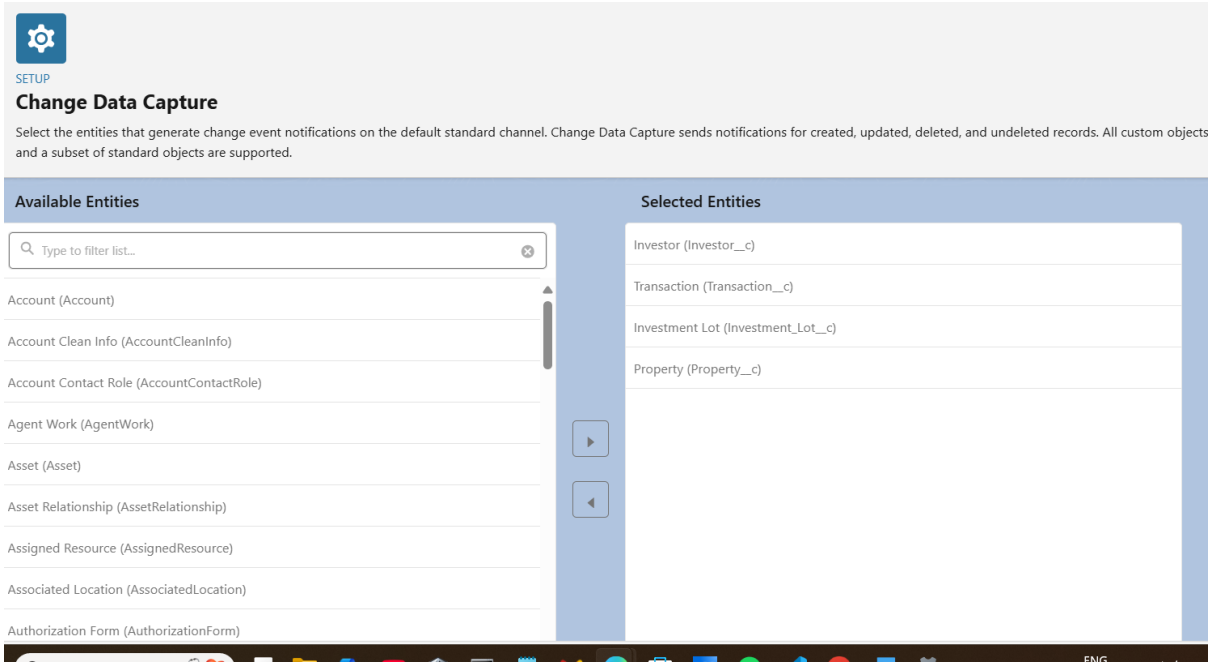
**Steps to Configure:**

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

---

## 7️⃣ API Limits & Authentication

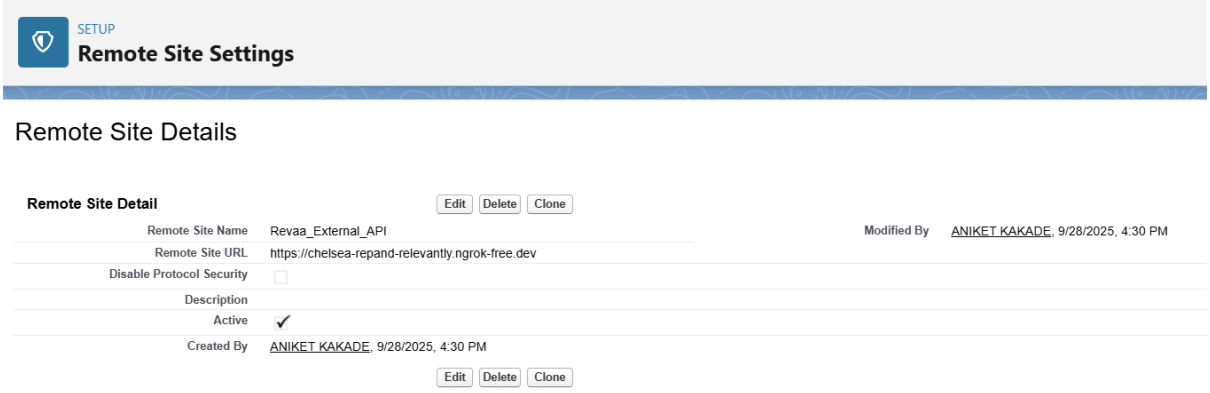
- Monitor API consumption at **Setup → System Overview → API Usage**.
- For OAuth or token-based authentication, always create a **Connected App** if required.
- Named Credential handles authentication for Flows, Apex, and External Services.

---

## 8️⃣ 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**.




---

## ✅ 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:**

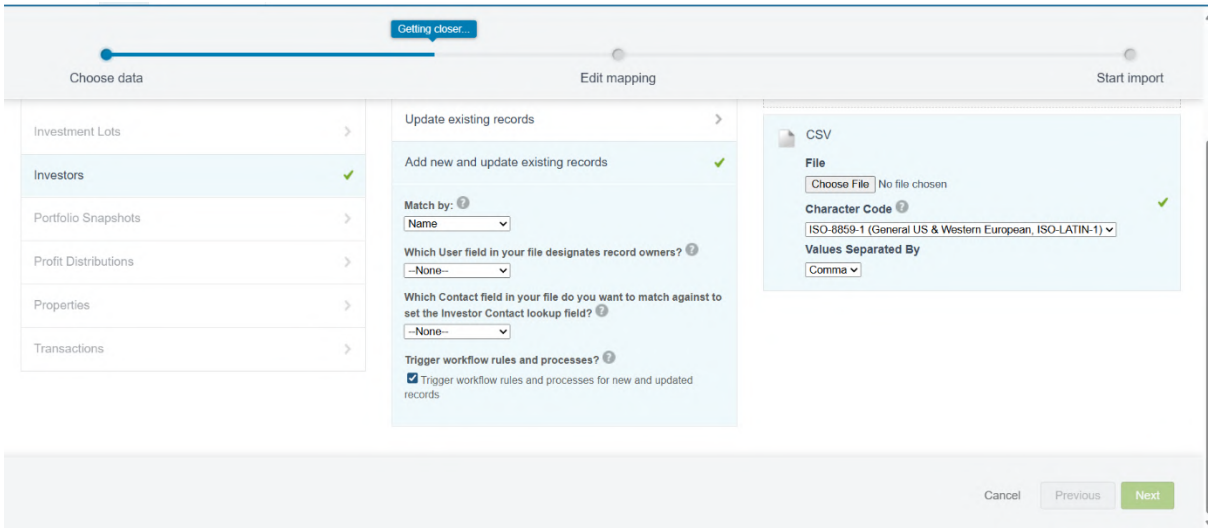
1. Navigate to: **Setup → Data → Data Import Wizard**.
2. Click **Launch Wizard**.
3. Select **Standard or Custom Object** (e.g., Investor__c).
4. Choose **Add New Records** or **Update Existing Records** depending on the use case.
5. Upload the CSV file.
    - Example: Investor__c.csv for investors.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

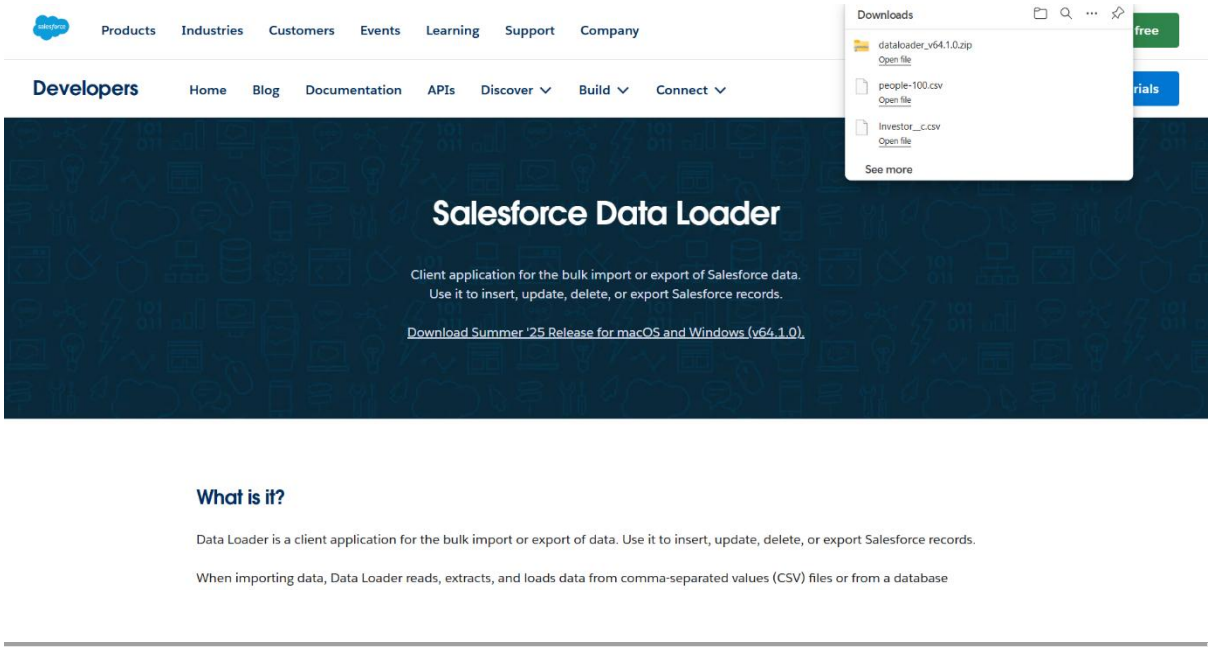
**Steps:**

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.







---

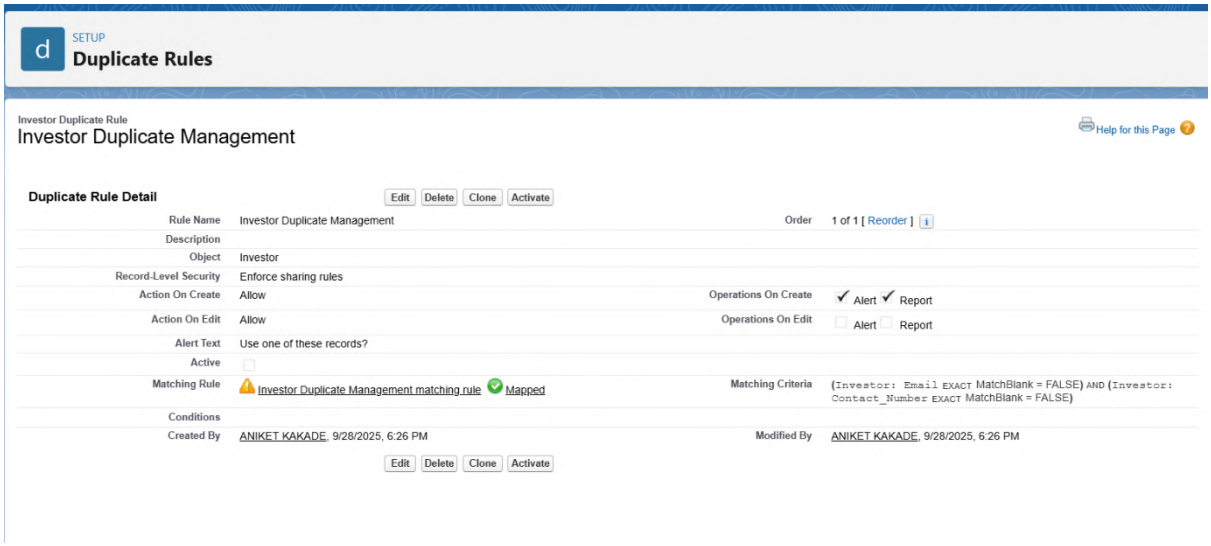
## 3. Duplicate Rules

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:**



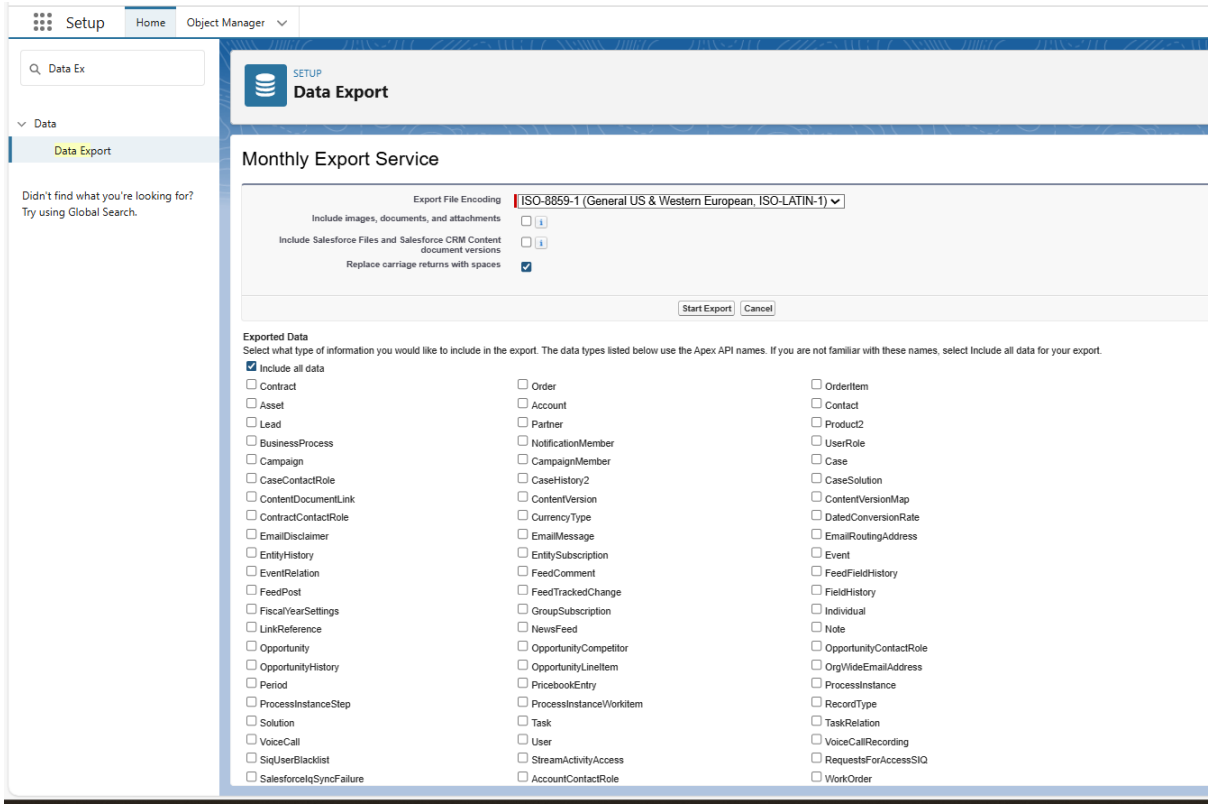

---

## 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.




---

## 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**.

---

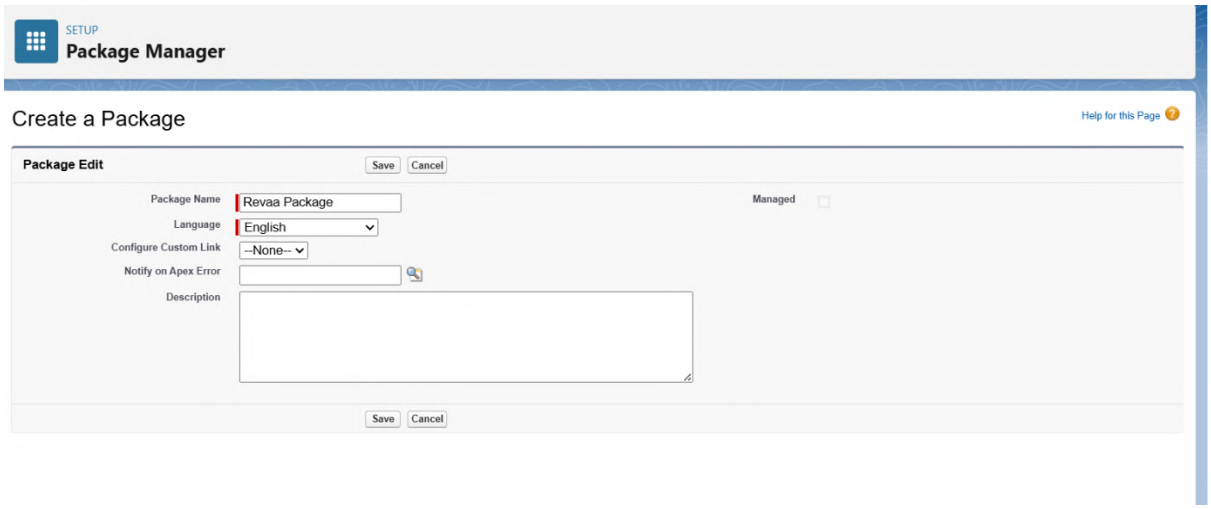
## 6. Unmanaged vs Managed Packages

- **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:
4. ant retrieve
5. ant deploy
6. Validate deployment before pushing to production.

---

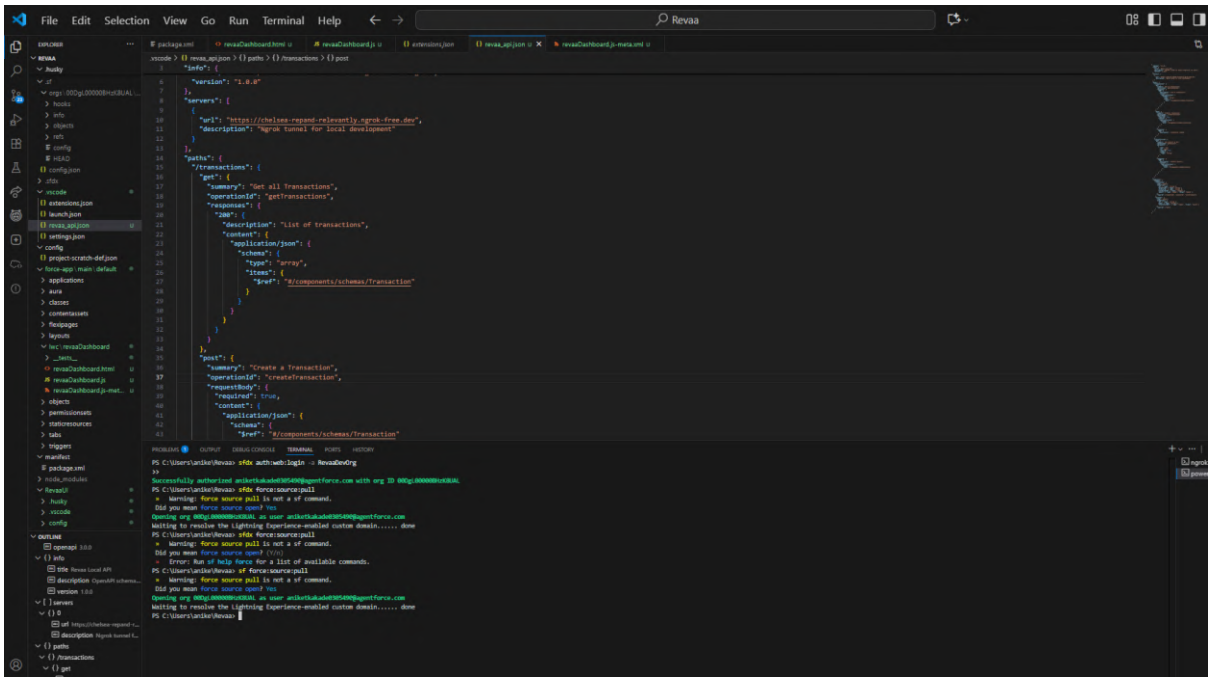
## 8. VS Code & SFDX

**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.




---

**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.

---

## 1️⃣ 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 |

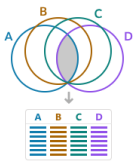
### Object Relationships

Investment Lots (A)

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

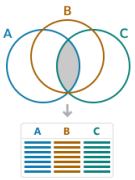
| Source Object | Included Fields |
|---|---|
| Investors | 16 |
| Transactions | 19 |
| Duplicate Record Items | 8 |

### Object Relationships

Investors (A)

with at least one related record from Transactions (B)

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



## 2️⃣ 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 |

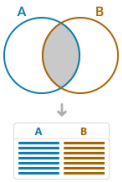
### Fields

| Source Object | Included Fields |
|---|---|
| Transactions | 20 |
| Activities | 36 |

### Object Relationships

Transactions (A)

with at least one related record from Activities (B)




## Prperty 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** | Prperty Transactions Report |
| **API Name** | Prperty_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 |

### Fields

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

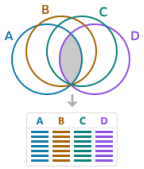
### Object Relationships

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)



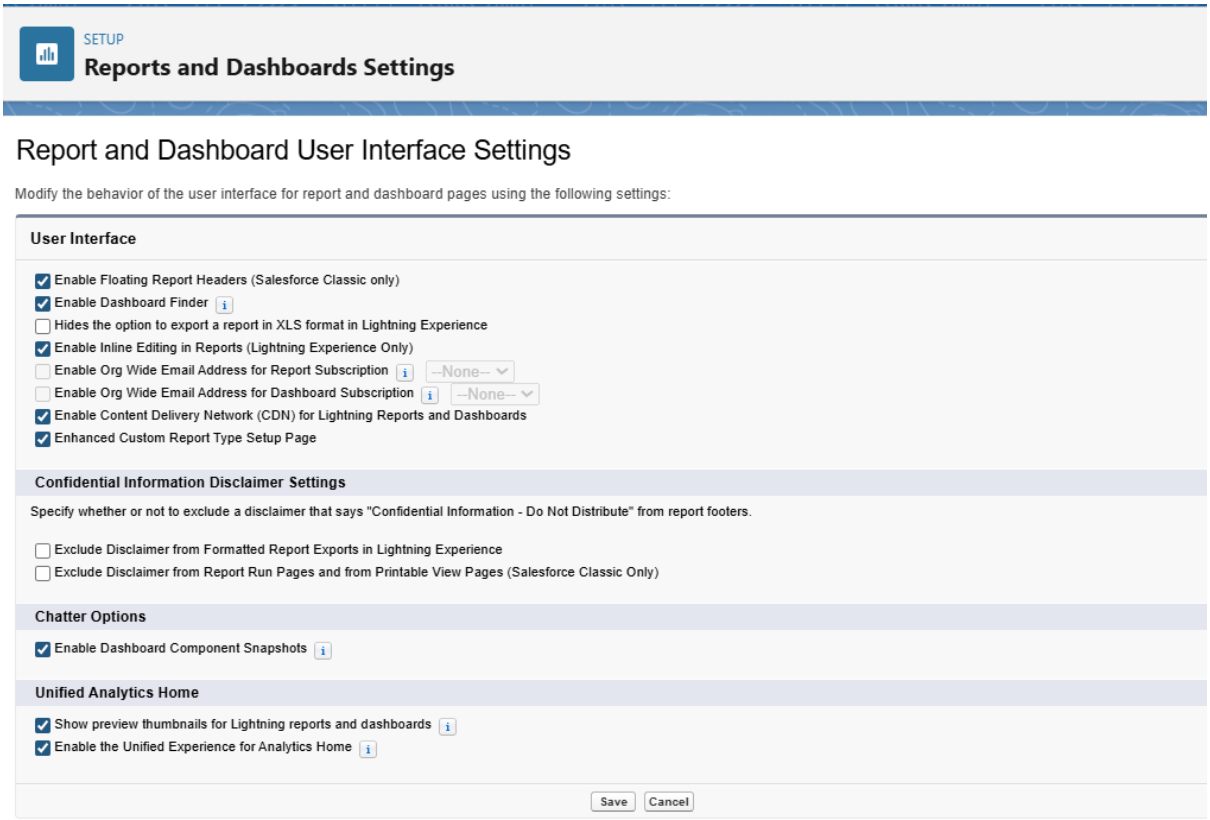
## 3️⃣ Dashboards

**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).




---

## Security Review

### A. Field-Level Security

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

---

## 5️⃣ 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.