

AgriConnect – Phase 1: Problem Understanding & Industry Analysis

Problem Statement

Agriculture supply chains often face inefficiencies due to:

- Lack of centralized farmer and crop data.

- Manual tracking of buyer inquiries and farmer–buyer transactions.
- Poor coordination between farmers, officers, and buyers.
- No real-time insights into crop availability, yield, or sales.

This leads to missed sales opportunities, delays in advisory services and field visits, and limited decision-making for farmers and buyers.

Therefore, a Salesforce CRM solution is needed to digitize farmer support, buyer engagement, and crop-to-market visibility.

Objectives

1. To centralize farmer, crop, and buyer data in Salesforce.
2. To automate capture (farmer inquiries, buyers).
3. To streamline field visit scheduling .
4. To provide real-time dashboards for records and engagement.
5. To improve farmer–buyer engagement through automated notifications and reporting.

Use Cases

1. **Farmer Management:** Register farmers with details like crops grown, and location. Assign officers.
2. **Crop & Inventory Tracking:** Maintain digital records of crops, and availability.
3. **Visit Scheduling:** Farmers receive emails.
4. **Buyer Inquiry & Deal Closure:** Capture buyer requests, with crop availability
5. **Reporting & Dashboards:** Farmer engagement, crops, officer activity reports.

Stakeholder Analysis

- **Farmers:** Want advisory help, better market access, timely updates.
- **Officers:** Need to track farmers, schedule visits, provide support.
- **Buyers/Distributors:** Need crop availability, pricing, and faster deals.
- **Administrators:** Want dashboards for yield, sales, and performance.

Business Process Mapping (Before Salesforce)

- Farmer calls officer manually Officer notes crops.
- Buyers approach farmers individually Delays in deals.
- No central database Limited visibility into stock & yield.
- Reports prepared manually Time-consuming.

Industry-Specific Use Case Analysis (Agriculture)

- Similar to real estate CRM: crops to buyers.
- visits similar to doctor appointments in healthcare CRM.

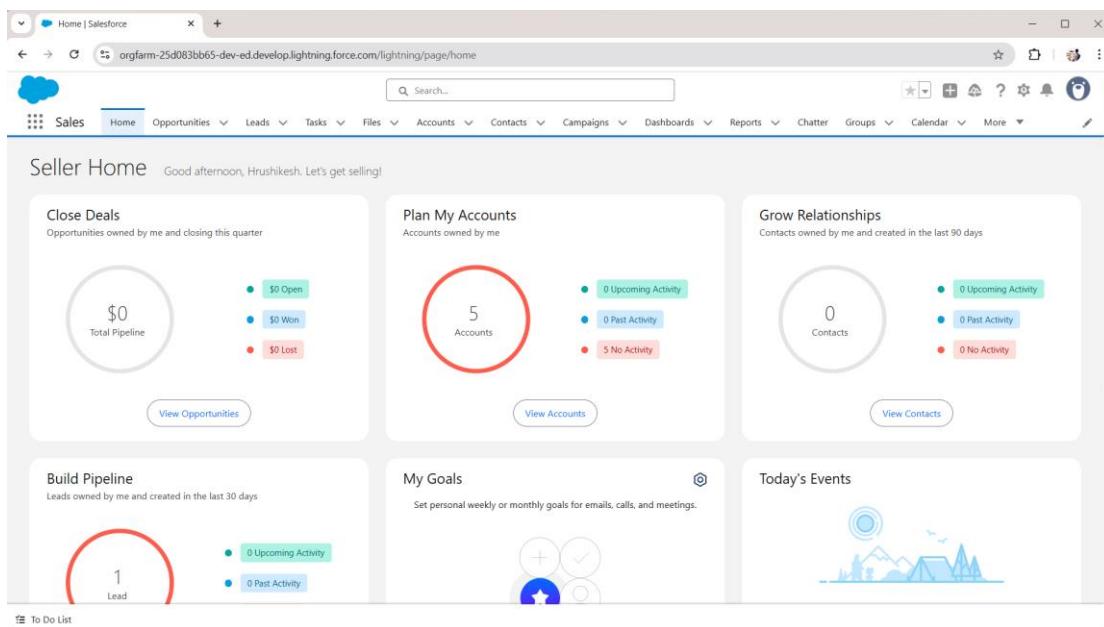
Phase 2: Salesforce Org Setup & Configuration

Project: AgriConnect -Smart Agriculture CRM

Institute: Prof. Ram Meghe Institute of Technology & Research

1. Salesforce Edition

- Developer Org used for implementation



// I HAVE USED SAME ORG THAT WAS USED FOR TRAINING

2. Company Profile Setup

- Organization Name: Prof. Ram Meghe Institute of Technology & Research

- Default Locale: English (India)
- Default Currency: INR
- Default Time Zone: (GMT+05:30) India Standard Time

Process Flow: Setup → Company Information → Edit Company profile

The screenshot shows the Company Information page in the Salesforce Setup. The organization's name is "Prof. Ram Meghe Institute Of Technology and Research". The "Organization Detail" section includes fields like Organization Name, Primary Contact, Division, Address, Fiscal Year Starts In, Activate Multiple Currencies, Enable Data Translation, Newsletter, Admin Newsletter, Hide Notices About System Maintenance, Hide Notices About System Downtime, and Locale Formats (set to ICU). The right side of the page displays various system metrics and settings. The URL in the browser bar is <https://orgfarm-25d083bb65-dev-ed.lightning.force.com/lightning/setup/CompanyProfileInfo/home>.

3. Business Hours & Holidays

- Business Hours: Default (All days, 05:00 AM – 12:00 AM)

Process Flow: Setup → Company Settings → Business

Hours → Define & Assign

Organization Business Hours

Select the days and hours that your support team is available. These hours, when associated with escalation rules, determine the times at which cases can escalate.

If you enter blank business hours for a day, that means your organization does not operate on that day.

Business Hours Edit

Step 1. Business Hours Name

Business Hours Name: Default
Active:

Step 2. Time Zone

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

Step 3. Business Hours

Day	From	To	24 hours
Sunday	05:00 AM	12:00 AM	<input type="checkbox"/>
Monday	05:00 AM	12:00 AM	<input type="checkbox"/>
Tuesday	05:00 AM	12:00 AM	<input type="checkbox"/>
Wednesday	05:00 AM	12:00 AM	<input type="checkbox"/>
Thursday	05:00 AM	12:00 AM	<input type="checkbox"/>
Friday	05:00 AM	12:00 AM	<input type="checkbox"/>
Saturday	05:00 AM	12:00 AM	<input type="checkbox"/>

4. Fiscal Year Settings

Process Flow: Setup → Company Settings → Fiscal Year → Custom Fiscal Year (if required)

Organization Fiscal Year Edit: Prof. Ram Meghe Institute Of Technology and Research

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

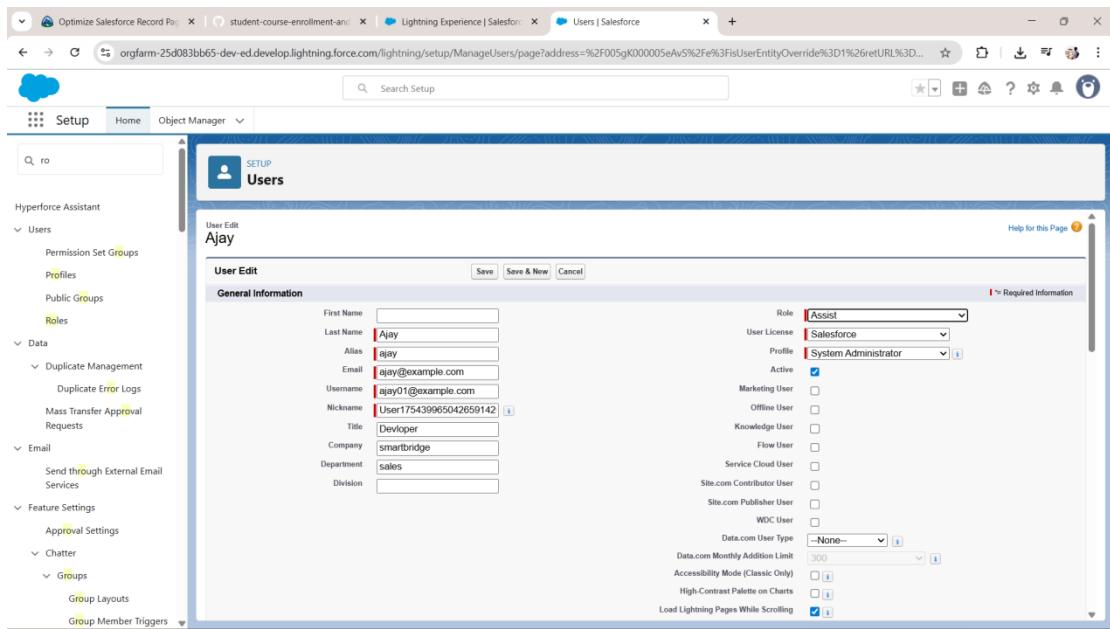
Name: Prof. Ram Meghe Institute Of Technology and Research
Fiscal Year Start Month: January
Fiscal Year is Based On: The ending month

5. User Setup & Licenses

Ajay: License = Salesforce, Profile = System Administrator,

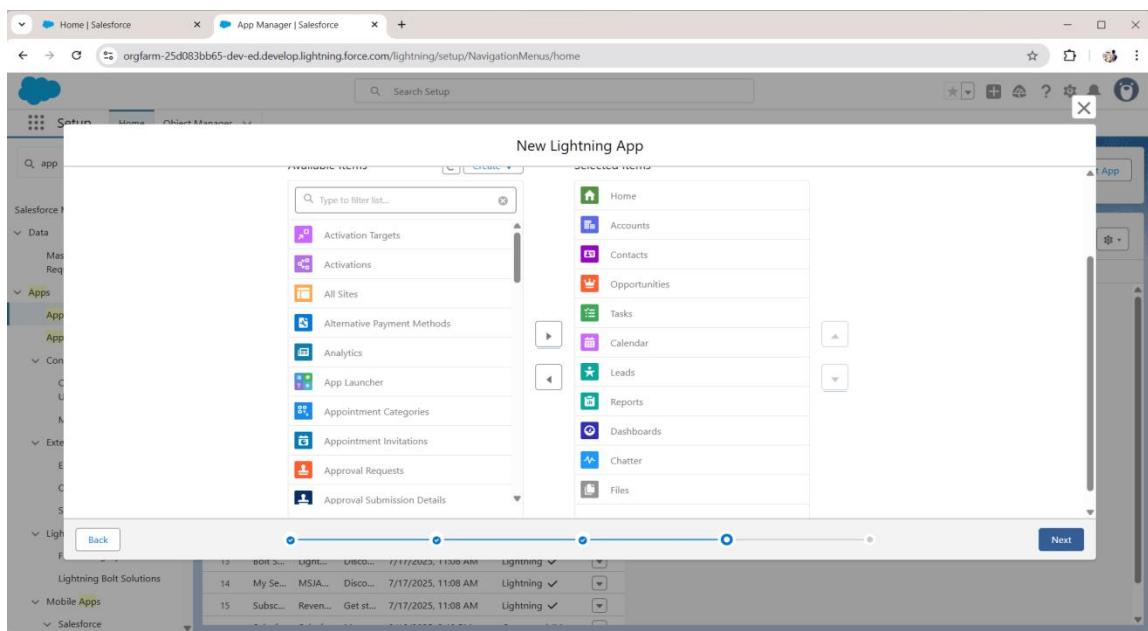
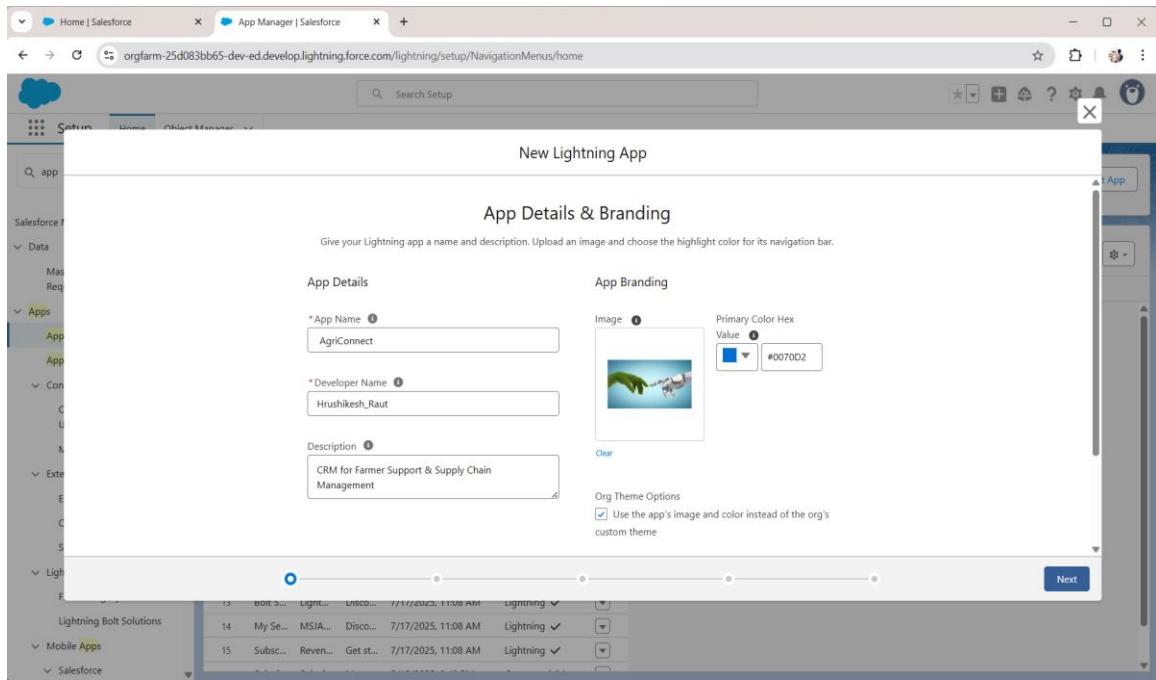
Role = Assist

Process Flow: Setup → Users → New User → Assign License & Profile



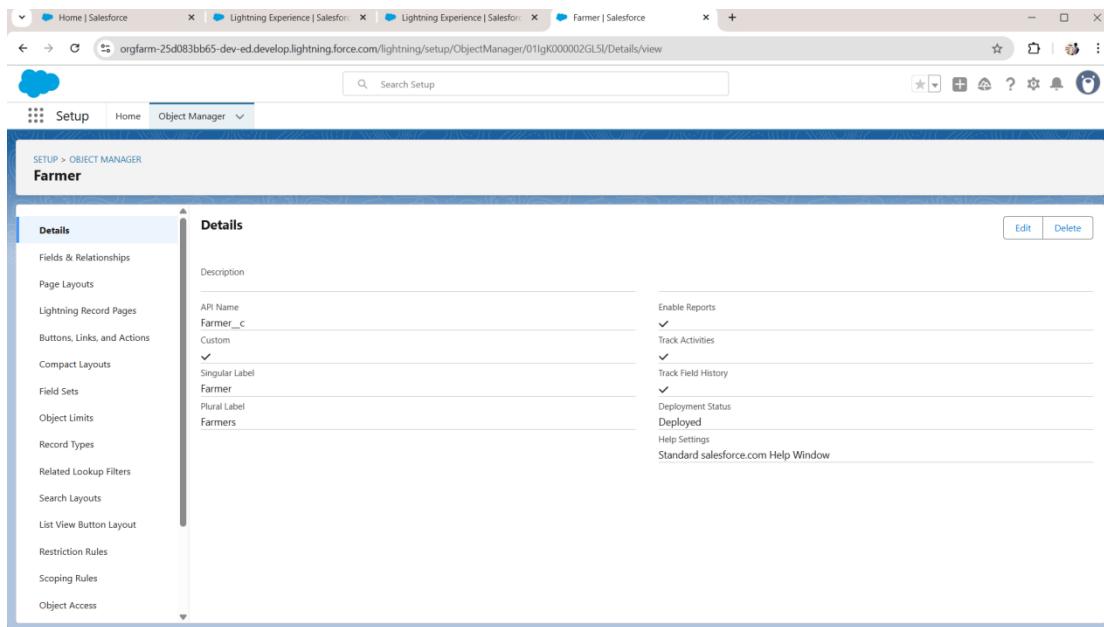
5. App Creation

- Go to App Manager → Click “New Lightning App” → Name the app “AgriConnect” → follow steps (ex : navigation items, next,.. save)



6. Create Objects

- Go to **Object Manager** → Click “Create” → **Custom Object**
- Define object name (Ex: “Farmer”, “Buyer”, “Crop”, “Visit”, “Deal”)



- Add necessary fields (text, date, picklist, etc.)

7. Create Custom fields

- Inside the object, go to **Fields & Relationships**
- Add new fields based on the project requirements
- Fields & Relationship → New → Choose data type → next → label name → Fill requirements → save

SETUP > OBJECT MANAGER

Farmer

Fields & Relationships

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address_c	Text Area(255)		
Created By	CreatedById	Lookup(User)		
crop	crop_c	Picklist		
Farmer Name	Name	Text(80)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone_Number	Phone_Number_c	Phone		

8. Create Custom fields

- Go to **Tabs** in Setup
- Create a new tab for custom object

SETUP

Tabs

Custom Tabs

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 to Lightning Experience and the mobile app.

Custom Object Tabs

Action	Label	Tab Style	Description
Edit Del	Boxes	People	
Edit Del	Leaf	Leaf	
Edit Del	Bank	Bank	
Edit Del	People	People	
Edit Del	Form	Form	
Edit Del	Chip	Chip	
Edit Del	Desk	Desk	
Edit Del	Building Block	Building Block	

Web Tabs

No Web Tabs have been defined

Visualforce Tabs

Phase 3 : Data Modeling & Relationships - Completion Report

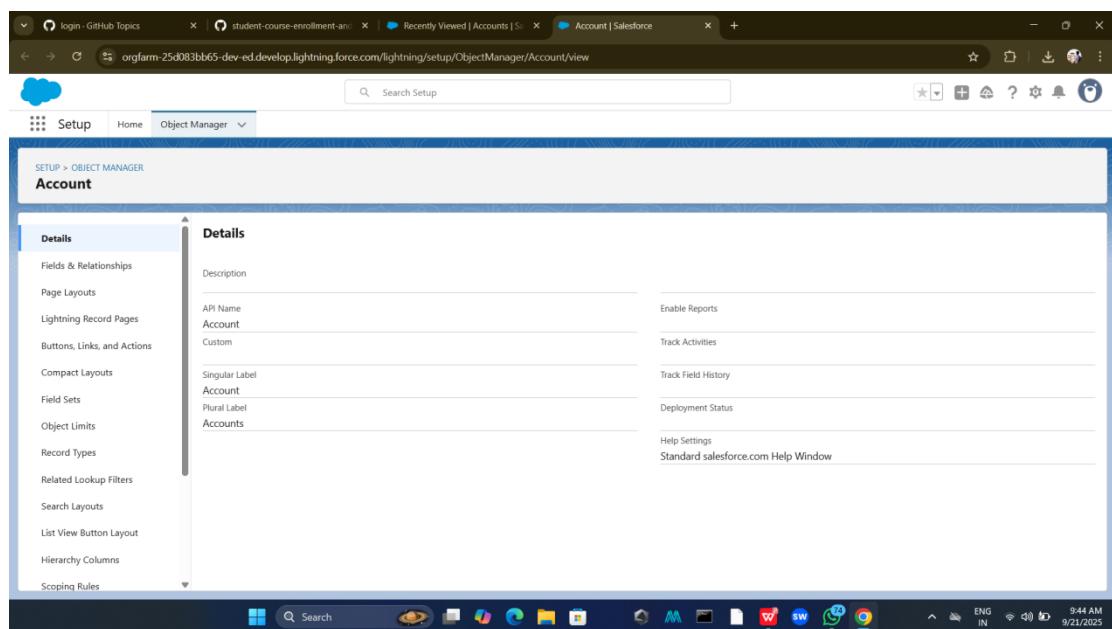
1. Standard & Custom Objects

Process Flow: Setup → Object Manager → New Custom Object → Add Fields. All

objects configured in a similar manner, with respective fields as listed

Standard Objects :

Account (used for buyer or farmer cooperatives)



Opportunity (used for Deals between Farmer & Buyer)

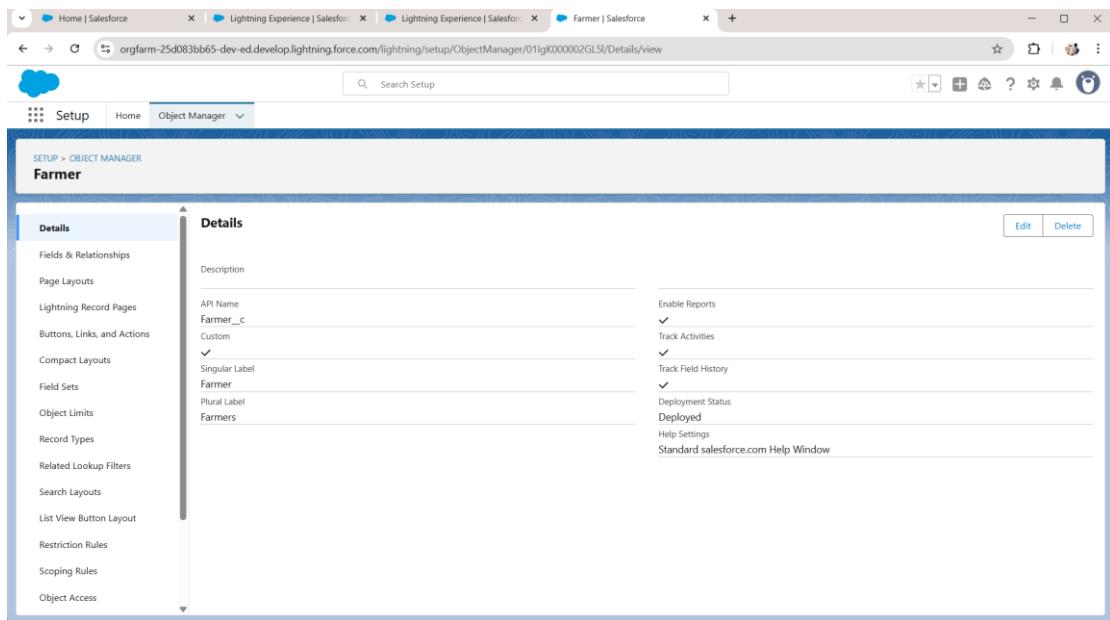
The screenshot shows the Salesforce setup interface for the Opportunity object. The left sidebar lists various configuration options like Fields & Relationships, Page Layouts, and Record Types. The main pane displays the 'Details' section for the Opportunity object, which includes fields for API Name (Opportunity), Singular Label (Opportunity), and Plural Label (Opportunities). It also shows settings for Description, Enable Reports, Track Activities, Track Field History, Deployment Status, Help Settings, and a link to the Standard salesforce.com Help Window.

Contact (used for individual Farmer and Buyer)

The screenshot shows the Salesforce setup interface for the Contact object. The left sidebar lists various configuration options. The main pane displays the 'Details' section for the Contact object, which includes fields for API Name (Contact), Singular Label (Contact), and Plural Label (Contacts). It also shows settings for Description, Enable Reports, Track Activities, Track Field History, Deployment Status, Help Settings, and a link to the Standard salesforce.com Help Window.

Custom Objects :

Farmer_c



The screenshot shows the Salesforce Setup interface under the Object Manager section. A sidebar on the left lists various configuration options like Fields & Relationships, Page Layouts, and Record Types. The main pane displays the 'Details' tab for the 'Farmer' object. It includes fields for Description, API Name (Farmer_c), Singular Label (Farmer), Plural Label (Farmers), and various deployment and reporting settings.

FIELD LABEL	SORTED ASCENDING	SORTFIELD NAME	SORTDATA TYPE
Address		Address__c	Text Area(255)
Created By		CreatedBy	Lookup(User)
crop		crop__c	Picklist
Farmer Name		Name	Text(80)
Last Modified By		LastModifiedBy	Lookup(User)
Owner		OwnerId	Lookup(User,Group)
Phone_Number		Phone_Number__c	Phone

Visits_c

FIELD LABEL SORTED ASCENDING	SORTFIELD NAME	SORTDATA TYPE
Created By	CreatedById	Lookup(User)
Last Modified By	LastModifiedById	Lookup(User)
Lookup	Lookup__c	Lookup(Farmer)
Owner	OwnerId	Lookup(User,Group)
Purpose	Purpose__c	Text(150)
Visit Date	Visit_Date__c	Date
Visit Name	Name	Text(80)
Visiter name	Visiter_name__c	Text(150)

Crop_c

FIELD LABEL SORTED ASCENDING	SORTFIELD NAME	SORTDATA TYPE
Created By	CreatedById	Lookup(User)
crop Name	Name	Text(80)
Harvest Time	Harvest_Time__c	Date
Last Modified By	LastModifiedById	Lookup(User)
Master detail	Master_detail__c	Master-Detail(Farmer)
Prize	Prize__c	Currency(18, 0)
Quantity	Quantity__c	Number(18, 0)

Buyer_c

FIELD LABEL SORTED ASCENDING	SORTFIELD NAME	SORTDATA TYPE
Buyer Name	Name	Text(80)
Contact	Contact__c	Phone
Created By	CreatedBy	Lookup(User)
Last Modified By	LastModifiedBy	Lookup(User)
Owner	OwnerId	Lookup(User,Group)

Deal_c

FIELD LABEL SORTED ASCENDING	SORTFIELD NAME	SORTDATA TYPE
Created By	CreatedById	Lookup(User)
crop	crop__c	Picklist
Deal Name	Name	Text(80)
Farmer	Farmer__c	Lookup(Farmer)
Last Modified By	LastModifiedById	Lookup(User)
Lookup1	Lookup1__c	Lookup(Buyer)
Owner	OwnerId	Lookup(User,Group)
Quantity	Quantity__c	Text(150)
status	status__c	Picklist

2. Page Layouts

Configured for Farmer_c as an example:

Process Flow: Setup → Object Manager → Farmer → Page Layouts →

Farmer Layout

→ Drag fields into sections → Save.

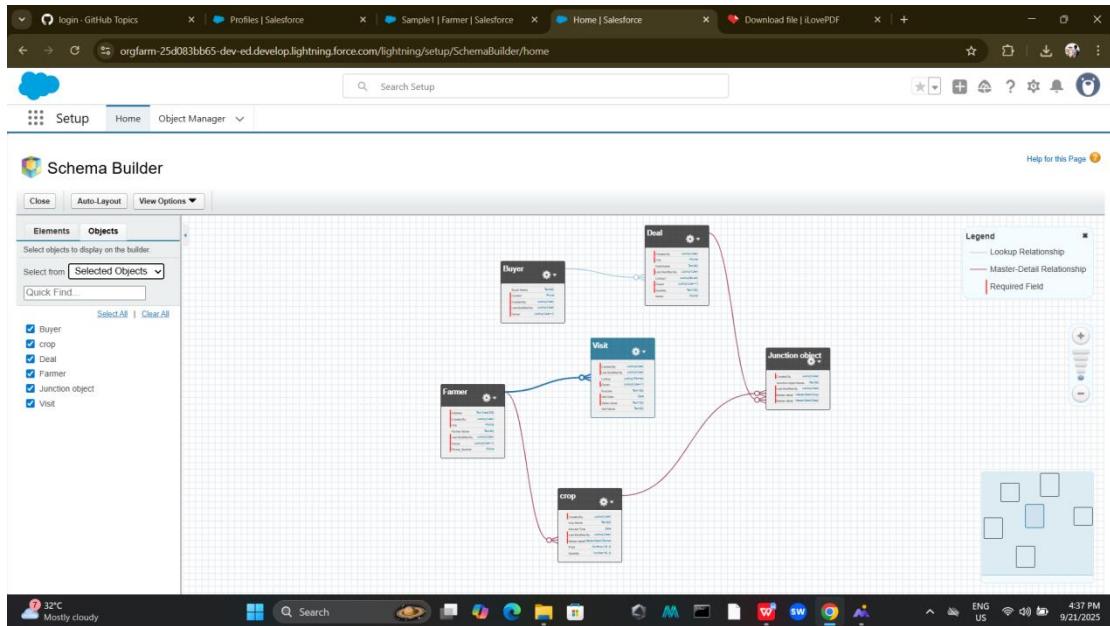
The screenshot shows the Salesforce Setup interface. The left sidebar is titled 'SETUP > OBJECT MANAGER' and lists various options like Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, and Scoping Rules. The 'Page Layouts' option is selected. The main content area is titled 'Page Layouts' and shows a table with one item: 'Farmer Layout' created by Hrushikesh Raut on 9/15/2025 at 3:04 AM, last modified by Hrushikesh Raut on 9/15/2025 at 3:35 AM. The table has columns for 'PAGE LAYOUT NAME', 'CREATED BY', and 'MODIFIED BY'. There are buttons for 'Quick Find', 'New', and 'Page Layout Assignment'. The bottom of the screen shows the Windows taskbar with various icons and the system tray.

Related Lists (visit, deal, crops)

The screenshot shows the 'Layout Properties' dialog box for the 'Farmer Layout'. The 'Fields' tab is selected, displaying fields such as 'crop' (Section), 'Farmer Name' (Section), 'Phone Number' (Text), 'Address' (Text), 'Last Modified By' (Text), and 'Created By' (Text). Below the fields, there are sections for 'Information' (Header visible on edit only), 'System Information' (Header visible on edit only), 'Custom Links' (Header visible on edit only), and 'Mobile Cards (Salesforce mobile only)'. At the bottom of the dialog, there are buttons for 'Save', 'Quick Save', 'Preview As...', 'Cancel', 'Undo', 'Redo', and 'Layout Properties'. The bottom of the screen shows the Windows taskbar with various icons and the system tray.

Other objects (Crop, Visit, Buyer, Deal) had layouts configured similarly with their relevant fields.

3 .Schema Builder



- Farmer (Master) → Crop (Detail).
- Farmer → Visit (Lookup).
- Crop ↔ Buyer (Many-to-Many via Deal)

Phase 4 Completion Report – Process Automation (Admin)

1. Validation Rules

Validation rules were created across different objects:

Steps to Create Validation Rule :

1. Go to Setup → Object Manager → Select the required Object .
2. Navigate to Validation Rules → New Rule.
3. Enter Rule Name and Description.
4. Define the Formula (logic specific to the rule).
5. Enter the Error Message and select Error Location.
6. Save & Activate.

Repeating repeat the steps for each rule as listed above.

Example: Crop Quantity must be greater than 0

Steps:

Setup → Object Manager → Select **Crop** → **Validation Rules**.

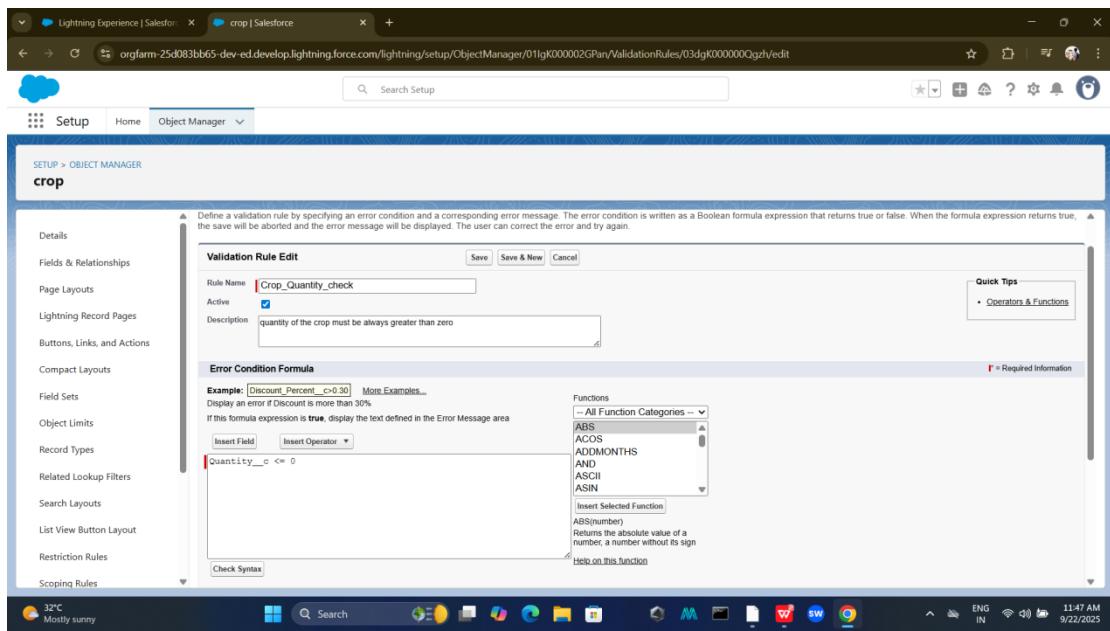
Click **New Rule** → Enter Rule Name: `Crop_Quantity_Check`.

Formula: `Quantity__c <= 0`

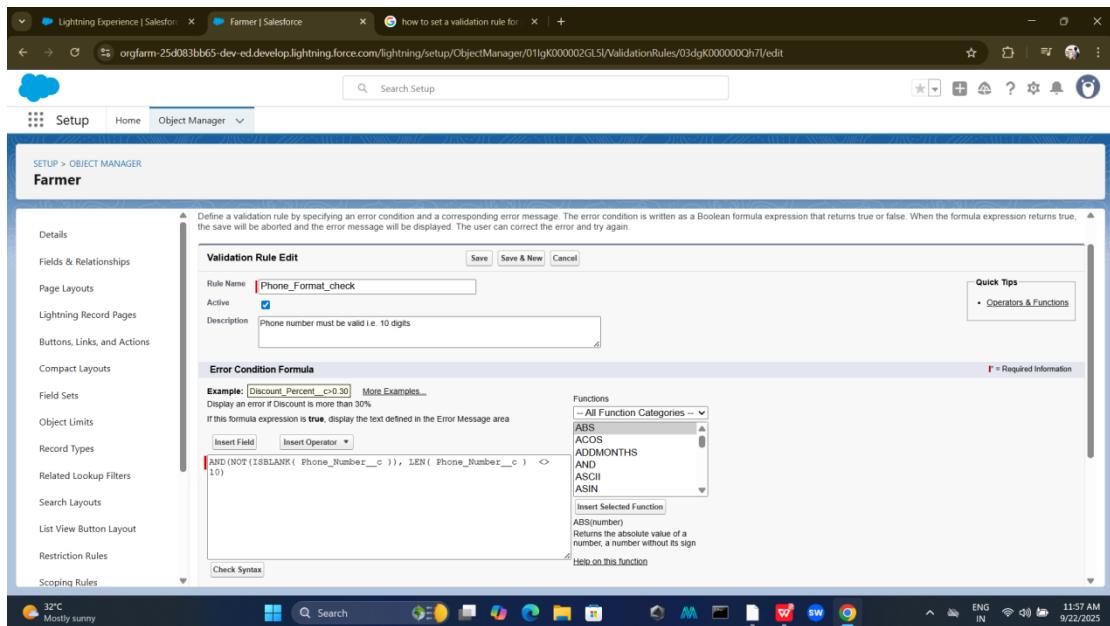
Error Message: “Quantity must be greater than 0.”

Choose where to display → *Top of Page or Field*.

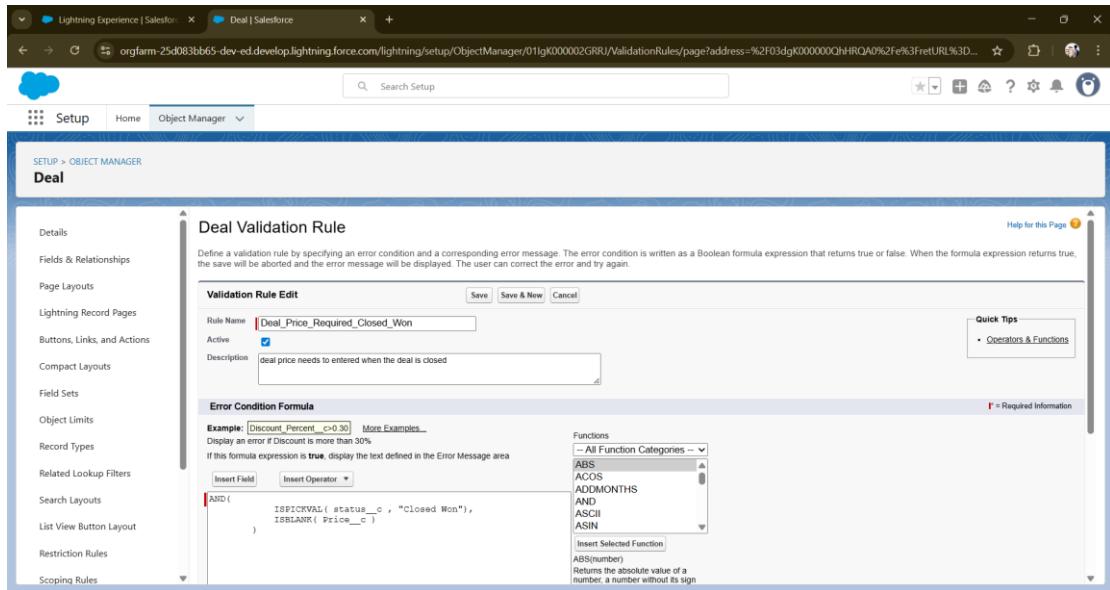
Save & Activate.



Farmer _ c Phone Number Format : (Phone number shouldn't be Null & must have 10 digits)



Deal_c Validation Rule : (Deal Price not empty when Closed Won)



2. WorkFlow Rules

Workflows perform actions when conditions are met.

- Example: Send Email when New Farmer is Created
- Steps:

Setup → Workflow Rules → New Rule.

Object: Farmer

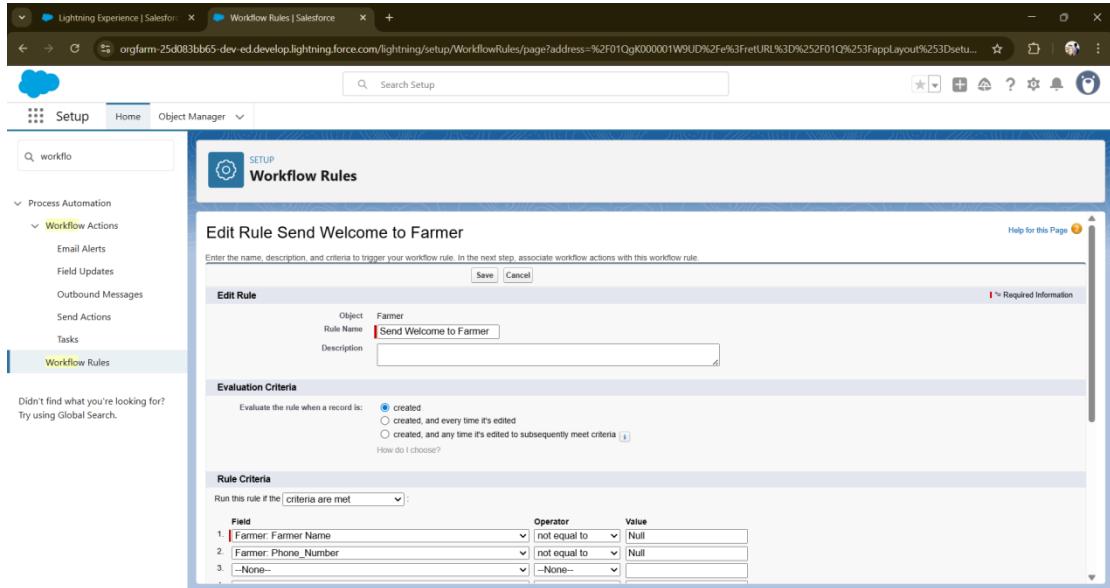
Rule Criteria: Created = True

Add Workflow Action → New Email Alert

Select Email Template → “Welcome Farmer”

Recipient → Farmer Email

Activate Workflow



3. Flow Builder

Flows replace Workflow & Process Builder.

➤ Record-Triggered Flow (Visit Scheduling)

Trigger: When new Visit record is created.

Action: Send email notification to Farmer.

Steps:

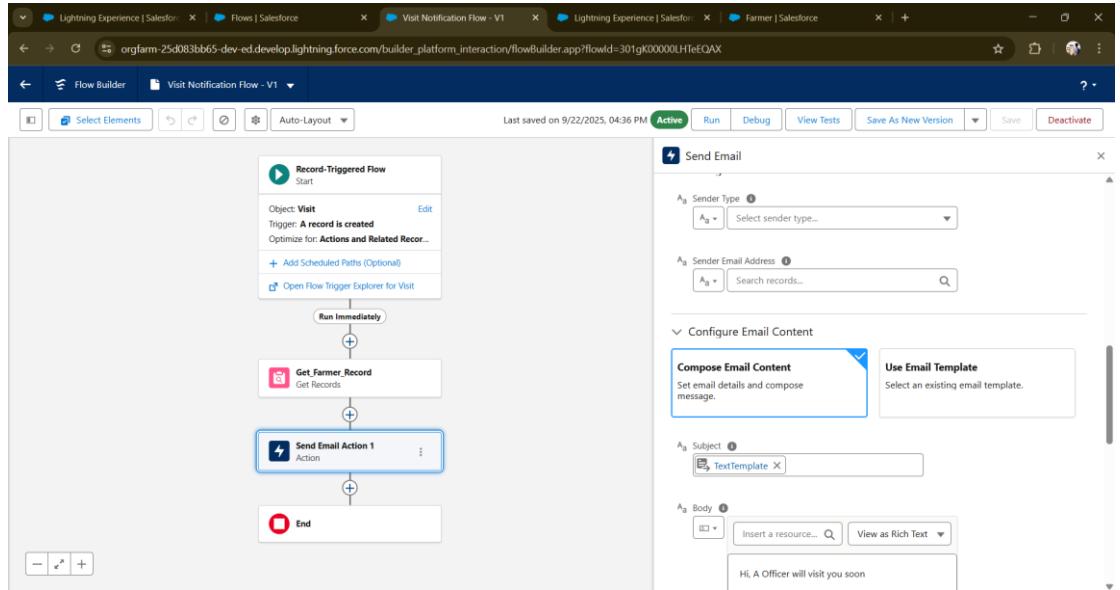
Setup → Flows → New Flow.

Choose **Record-Triggered Flow** → Object = Visit.

Trigger = On Create.

Add **Action** → Email Alert

Save & Activate.



➤ Update Crop Stock after Deal Closure

Steps:

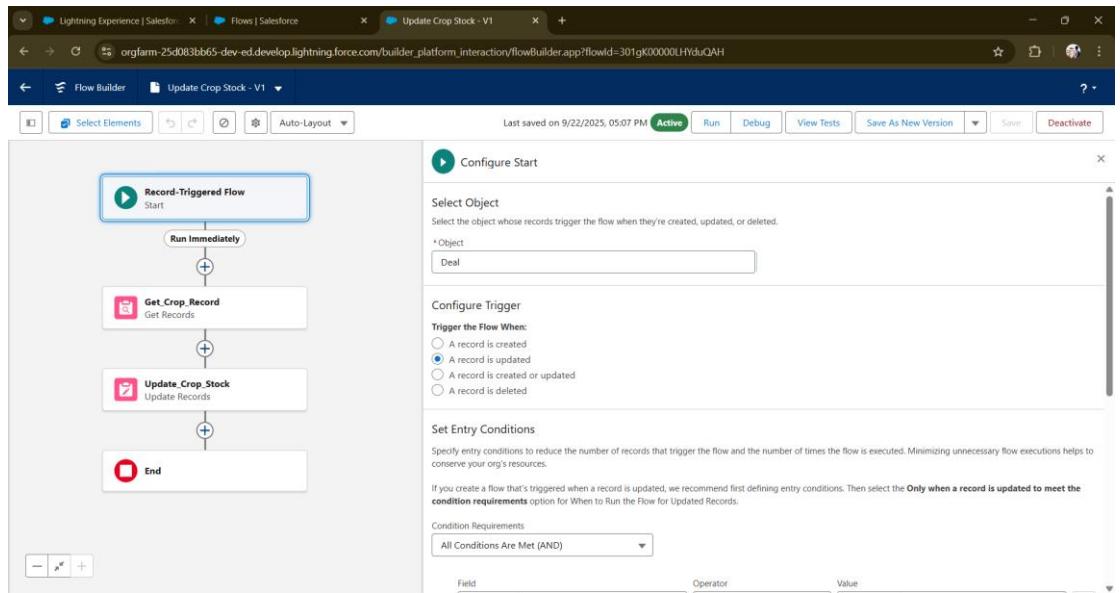
Setup → Flows → New Flow.

Type: **Record-Triggered** → Object = Deal.

Trigger = When Deal Status = “Closed Won.”

Add Update Records → Reduce Crop.Quantity = Crop.Quantity – Deal.Quantity.

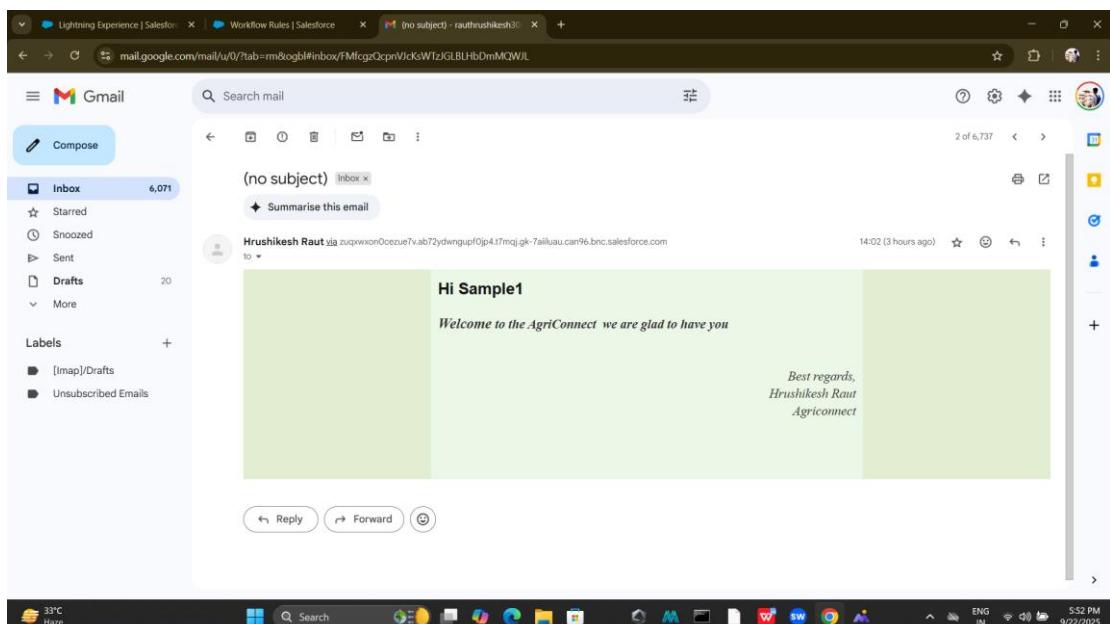
Save & Activate.



4. Email Alerts

Email Alerts were successfully configured and activated :

Example Email : Welcome email for Farmer



Phase 5 Completion Report – Apex Programming (Developer)

1. Classes & Objects

● Create an Apex Class

Steps:

1. In Salesforce → Setup → Quick Find → **Apex Classes** → New.

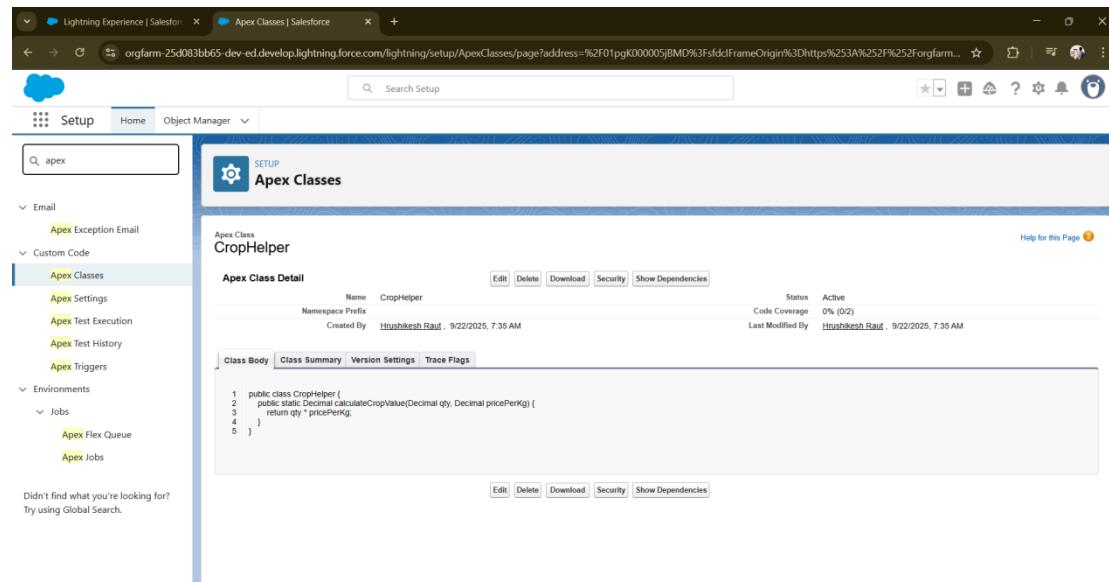
OR Developer Console → File → New → Apex Class.

2. Enter Class Name and click OK.

3. Write Apex code implementing the required logic.

4. Save the class.

Example: Utility class to calculate total crop value.



2. Apex Triggers

Steps to Create Trigger:

1. Developer Console → File → New → Apex Trigger.
2. Select Object
3. Write Trigger code .
4. Save the trigger

✧ **Example :**Let's create a Trigger i.e.

When a Deal is **Closed Won**, automatically update the available Crop quantity.

Steps:

Setup → Apex Triggers → New Trigger.

Select **Deal__c** object.

Add Trigger Code as mentioned in screenshot

Save & Test: Close a Deal → Crop stock reduces

```

1
2 trigger UpdateCropStock on Deal__c (after insert, after update) {
3     List<Crop__c> cropsToUpdate = new List<Crop__c>();
4
5     for (Deal__c d : Trigger.new) {
6         if (d.Status__c == 'Closed Won') {
7             Crop__c c = [SELECT Id, Quantity__c
8                         FROM Crop__c
9                         WHERE Id = :d.Crop__c LIMIT 1];
10            c.Quantity__c = c.Quantity__c - d.Quantity__c;
11            cropsToUpdate.add(c);
12        }
13    }
14
15    if (!cropsToUpdate.isEmpty()) {
16        update cropsToUpdate;
17    }
18 }
19

```

Logs Tests Checkpoints Query Editor View State Progress **Problems**

Name Line Problem

3. SOQL (Salesforce Object Query Language)

---- Used to query data inside Apex.

Example Query:

```

List<Farm__c> farmers = [SELECT Id, Name, Address__c
                           FROM Farmer__c
                           WHERE Address__c = 'North'];

```

‘Queries Can Run in developer console’

- i. Enter Ctrl + E to open window for code
- ii. Enter Query followed by Debug Command
- iii. Click Execute

```

1
2
3 * List<Farmer__c> farmers = [SELECT Id, Name, Address__c
4                               FROM Farmer__c
5                               WHERE Address__c = 'North'];
6 System.debug(farmers);
7
8

```

The screenshot shows the Salesforce Developer Console interface. The top navigation bar includes 'File', 'Edit', 'Debug', 'Test', 'Workspace', and 'Help'. Below the navigation is a toolbar with 'Log executeAnonymous @9/22/2025, 9:22:42 PM' and a 'Logs' tab. The main area is titled 'Execution Log' with tabs for 'Timestamp', 'Event', and 'Details'. A message in the details pane says '[6]DEBUG(Farmer__c:[Id=a04gk00000156yQAA, Name=Rakesh, Address__c=North])'. To the right is a large 'Enter Apex Code' window containing the provided Apex code. At the bottom right of the console are buttons for 'Open Log', 'Execute', and 'Execute Highlighted'.

4. Collections: List, Set, Map

- **List:** Store multiple Farmers

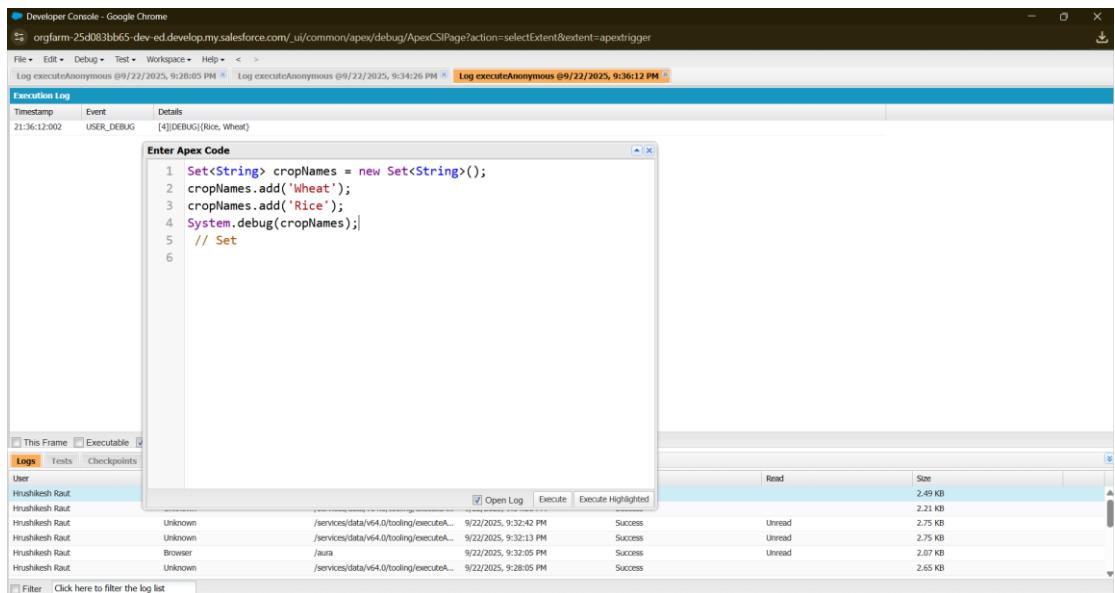
```

1 List<Farmer__c> farmerList = [SELECT Id, Name FROM Farmer__c];
2
3 System.debug(farmerList);
4 // List

```

The screenshot shows the Salesforce Developer Console interface. The top navigation bar includes 'File', 'Edit', 'Debug', 'Test', 'Workspace', and 'Help'. Below the navigation is a toolbar with 'Log executeAnonymous @9/22/2025, 9:40:25 PM' and a 'Logs' tab. The main area is titled 'Execution Log' with tabs for 'Timestamp', 'Event', and 'Details'. A message in the details pane says '[3]DEBUG(Farmer__c:[Id=a04gk0000014ybQAE, Name=Sample1], Farmer__c:[Id=a04gk00000156yQAA, Name=Rakesh], Farmer__c:[Id=a04gk00000156yQAA, Name=Suresh Raj])'. To the right is a large 'Enter Apex Code' window containing the provided Apex code. At the bottom right of the console are buttons for 'Open Log', 'Execute', and 'Execute Highlighted'.

➤ Set: Unique crop names.



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is `orgfarm-25d083bb65-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage?action=selectExtent&extent=apexttrigger`. The title bar says "Developer Console - Google Chrome". The main area is titled "Log executeAnonymous @9/22/2025, 9:36:12 PM". Below it is the "Execution Log" section. A modal window titled "Enter Apex Code" contains the following code:

```

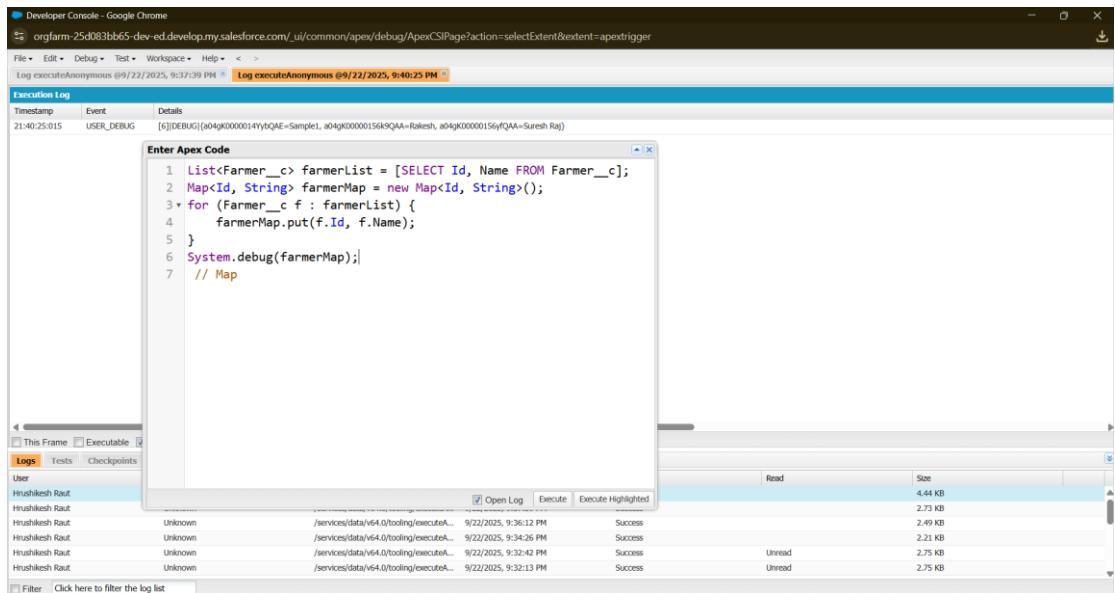
1 Set<String> cropNames = new Set<String>();
2 cropNames.add('wheat');
3 cropNames.add('Rice');
4 System.debug(cropNames);
5 // Set
6

```

The log table shows the following entries:

User	Timestamp	Event	Details	Read	Size
Hrushikesh Raut	21:36:12:002	USER_DEBUG	[4]DEBUG (Rice, Wheat)		2.49 kB
Hrushikesh Raut	Unknown	/services/data/v64.0/tooling/executeA...	9/22/2025, 9:32:42 PM	Success	2.21 kB
Hrushikesh Raut	Unknown	/services/data/v64.0/tooling/executeA...	9/22/2025, 9:32:31 PM	Success	2.75 kB
Hrushikesh Raut	Browser	/aura	9/22/2025, 9:32:05 PM	Success	2.07 kB
Hrushikesh Raut	Unknown	/services/data/v64.0/tooling/executeA...	9/22/2025, 9:28:05 PM	Success	2.65 kB

➤ Map: Match Farmer IDs with Names.



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is `orgfarm-25d083bb65-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage?action=selectExtent&extent=apexttrigger`. The title bar says "Developer Console - Google Chrome". The main area is titled "Log executeAnonymous @9/22/2025, 9:40:23 PM". Below it is the "Execution Log" section. A modal window titled "Enter Apex Code" contains the following code:

```

1 List<Farmer__c> farmerList = [SELECT Id, Name FROM Farmer__c];
2 Map<Id, String> farmerMap = new Map<Id, String>();
3 for (Farmer__c f : farmerList) {
4     farmerMap.put(f.Id, f.Name);
5 }
6 System.debug(farmerMap);
7 // Map

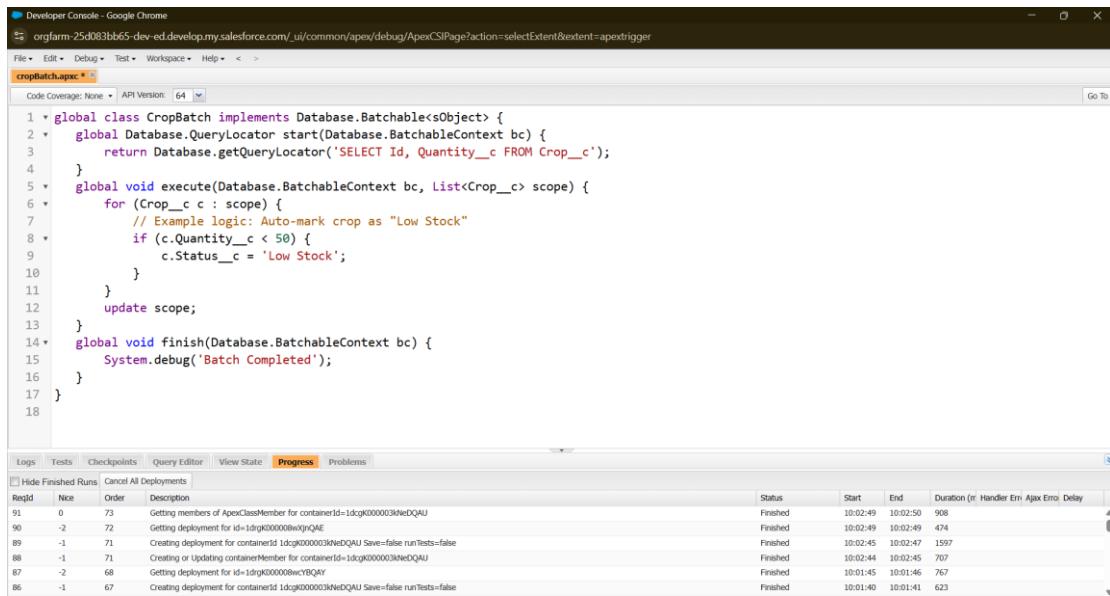
```

The log table shows the following entries:

User	Timestamp	Event	Details	Read	Size
Hrushikesh Raut	21:40:25:015	USER_DEBUG	[6]DEBUG (a04g00000014YyhQAE=Sample1, a04g000000156k9QAA=Rakesh, a04g000000156yfQAA=Suresh Raj)		4.44 kB
Hrushikesh Raut	Unknown	/services/data/v64.0/tooling/executeA...	9/22/2025, 9:36:12 PM	Success	2.49 kB
Hrushikesh Raut	Unknown	/services/data/v64.0/tooling/executeA...	9/22/2025, 9:34:26 PM	Success	2.21 kB
Hrushikesh Raut	Unknown	/services/data/v64.0/tooling/executeA...	9/22/2025, 9:32:42 PM	Success	2.75 kB
Hrushikesh Raut	Unknown	/services/data/v64.0/tooling/executeA...	9/22/2025, 9:32:13 PM	Success	2.75 kB

5. Batch Apex

- ✧ Batch Apex – Weekly Update Crop Reports :



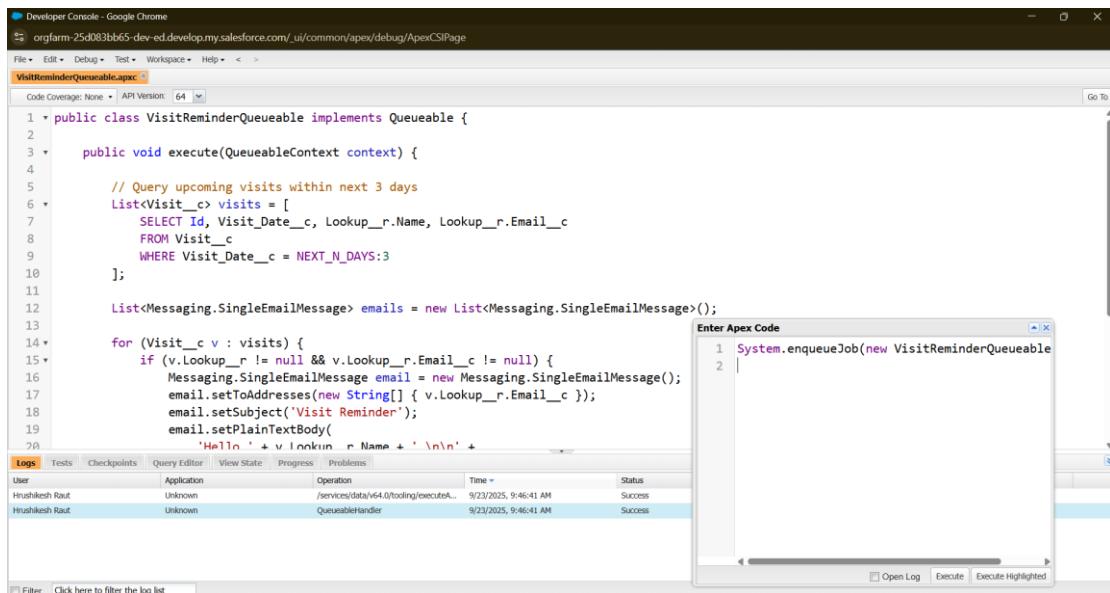
```
global class CropBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id, Quantity__c FROM Crop__c');
    }
    global void execute(Database.BatchableContext bc, List<Crop__c> scope) {
        for (Crop__c c : scope) {
            // Example logic: Auto-mark crop as "Low Stock"
            if (c.Quantity__c < 50) {
                c.Status__c = 'Low Stock';
            }
        }
        update scope;
    }
    global void finish(Database.BatchableContext bc) {
        System.debug('Batch Completed');
    }
}
```

The screenshot shows the developer console with the code editor open. Below the code editor is a progress bar tab labeled 'Progress' which is currently selected. The progress bar displays deployment logs for several runs, each showing a status of 'Finished' with specific start and end times.

ReqId	Nice	Order	Description	Status	Start	End	Duration (m)	Handler	Ajax	Errors	Delay
91	0	73	Getting members of ApexClassMember for containerId=1dgk0000034NeDQAU	Finished	10:02:49	10:02:50	908				
90	-2	72	Getting deployment for id=1drgk000008w0jgjQAE	Finished	10:02:49	10:02:49	474				
89	-1	71	Creating deployment for containerId 1dgk0000034NeDQAU Save=false runTests=false	Finished	10:02:45	10:02:47	1597				
88	-1	71	Creating or Updating containerMember for containerId=1dgk0000034NeDQAU	Finished	10:02:44	10:02:45	707				
87	-2	68	Getting deployment for id=1drgk000008wCYBQAV	Finished	10:01:45	10:01:46	767				
86	-1	67	Creating deployment for containerId 1dgk0000034NeDQAU Save=false runTests=false	Finished	10:01:40	10:01:41	623				

6. Queueable Apex

- ✧ Queueable Apex – Visit Reminder :



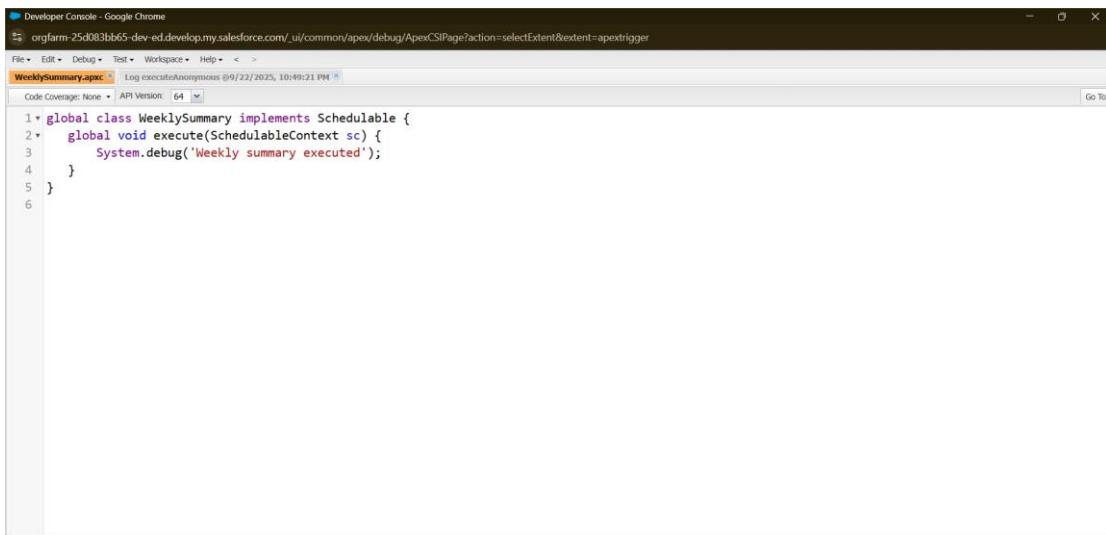
```
public class VisitReminderQueueable implements Queueable {
    public void execute(QueueableContext context) {
        // Query upcoming visits within next 3 days
        List<Visit__c> visits = [
            SELECT Id, Visit_Date__c, Lookup_r.Name, Lookup_r.Email__c
            FROM Visit__c
            WHERE Visit_Date__c = NEXT_N_DAYS:3
        ];
        List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();
        for (Visit__c v : visits) {
            if (v.Lookup_r != null && v.Lookup_r.Email__c != null) {
                Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
                email.setToAddresses(new String[] { v.Lookup_r.Email__c });
                email.setSubject('Visit Reminder');
                email.setPlainTextBody(
                    'Hello ' + v.Lookup_r.Name + '\n\n'
                );
                emails.add(email);
            }
        }
        System.enqueueJob(new VisitReminderQueueable());
    }
}
```

The screenshot shows the developer console with the code editor open. A tooltip for the 'enqueueJob' method is displayed in the bottom right corner. Below the code editor is a log table showing two entries for a user named Hrushikesh Raut. Both entries show a status of 'Success'.

User	Operation	Time	Status
Hrushikesh Raut	/services/data/v64.0/tooling/executeA...	9/23/2025, 9:46:41 AM	Success
Hrushikesh Raut	QueueableHandler	9/23/2025, 9:46:41 AM	Success

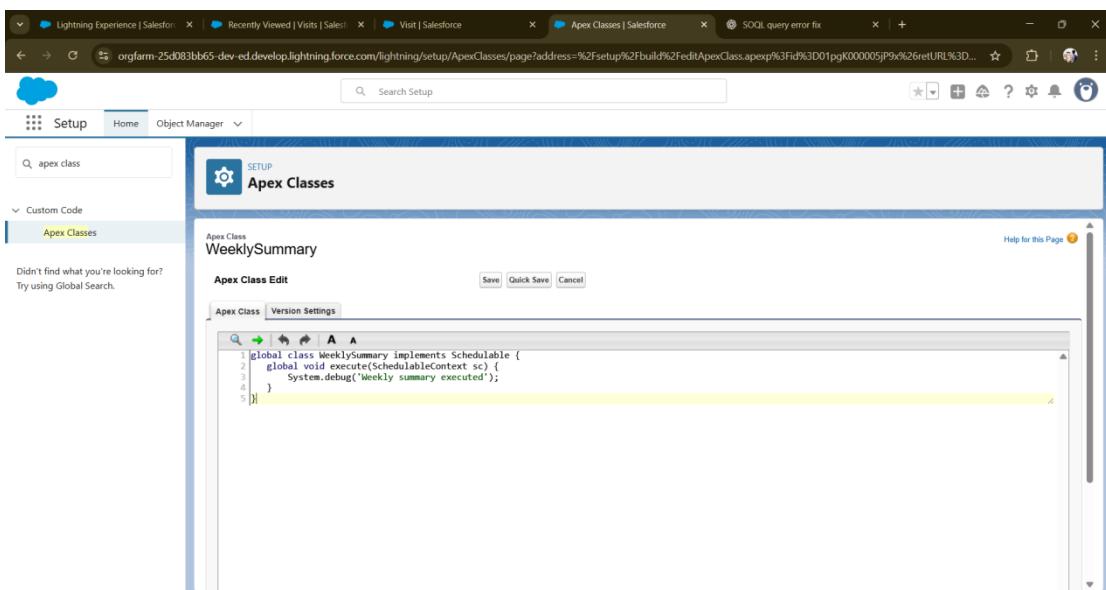
7. Schedule Apex

- ❖ Scheduled Apex – Run Every Week :



```
global class WeeklySummary implements Schedulable {
    global void execute(SchedulableContext sc) {
        System.debug('Weekly summary executed');
    }
}
```

Setup → Apex Classes → Schedule Apex or via Developer Console.



The screenshot shows the Salesforce Setup interface for managing Apex Classes. The left sidebar has 'Custom Code' expanded, with 'Apex Classes' selected. The main area is titled 'Apex Classes' and shows the 'WeeklySummary' class. The code editor contains the following Apex code:

```
global class WeeklySummary implements Schedulable {
    global void execute(SchedulableContext sc) {
        System.debug('Weekly summary executed');
    }
}
```

8. Test Classes

Requires 75% code coverage.

Steps:

Setup → Apex Test Execution → New Test.

Enter the test class name

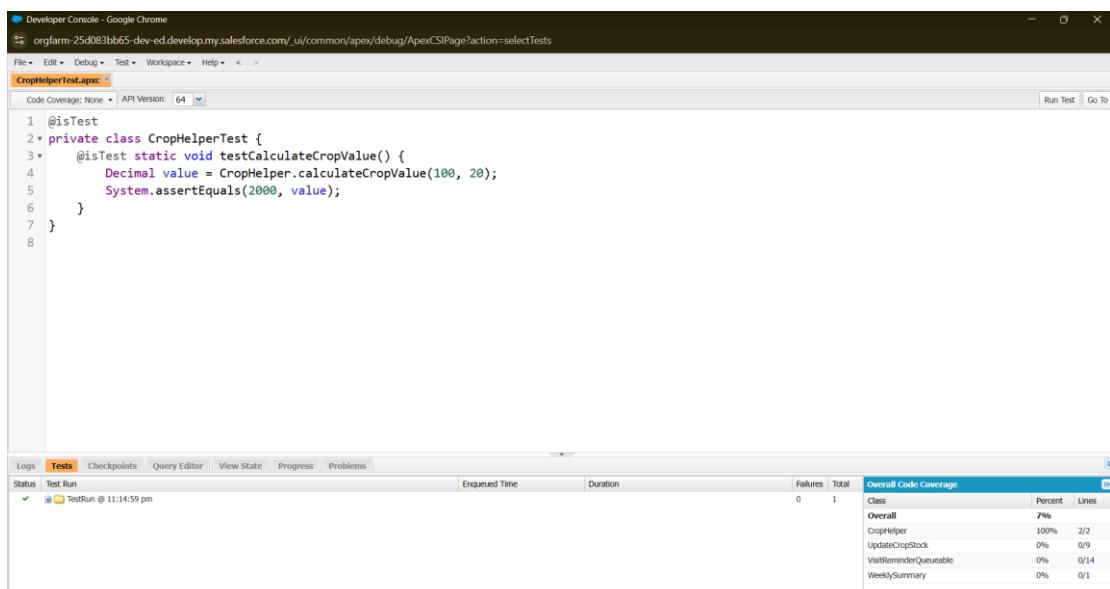
Enter test class code

Save & Run

Steps to run via setup :

- Go to **Setup → Apex Test Execution.**
- Click **Select Tests** → Pick **VisitReminderQueueableTest**.
- Run Test → Wait for result

Example : test for CropHelper (sends remainder for visit)



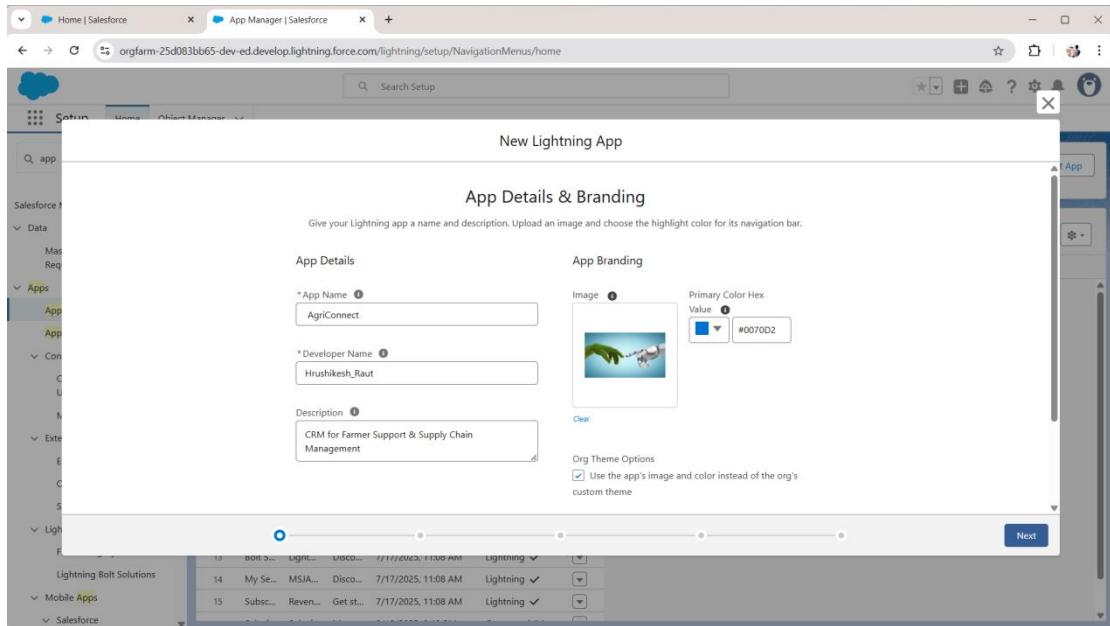
```
Developer Console - Google Chrome
File Edit Debug Test Workspace Help < >
CropHelperTest.apc
Code Coverage: None API Version: 64
Run Test Go To
1 @isTest
2 private class CropHelperTest {
3     @isTest static void testCalculateCropValue() {
4         Decimal value = CropHelper.calculateCropValue(100, 20);
5         System.assertEquals(2000, value);
6     }
7 }
8

Logs Tests Checkpoints Query Editor View State Progress Problems
Status Test Run Enqueued Time Duration Failures Total Overall Code Coverage
Test Run @ 11:14:59 pm 0 1 Class Overall Percent Lines
Overall 7% 2/2
CropHelper 100% 2/2
UpdateCropStock 0% 0/0
VisitReminderQueueable 0% 0/14
WeeklySummary 0% 0/1
```

Phase 6 User Interface Development

❖ Lightning App Builder

- This was the primary tool used for all declarative UI design. Used the Lightning App Builder's drag-and-drop canvas to create and modify the custom pages.



❖ Record Pages

Various Record pages can be created :

Steps:

(Same Steps can be applied to all pages)

1. Lightning App Builder → New → **Record Page** → Object = (select your object)
eg : **Crop_c** → name eg: Crop Record Page.
2. Template: Choose any Template (Ex : **One Region with Right Sidebar** (or Two Columns).
3. Add Components as per requirement into Canvas (Drag & Drop)
4. Example:

Top: **Highlights Panel** (compact layout show Crop Name, Quantity, Price).

Main: **Record Details** (all fields: Variety, Harvest Date, Quantity, Price).

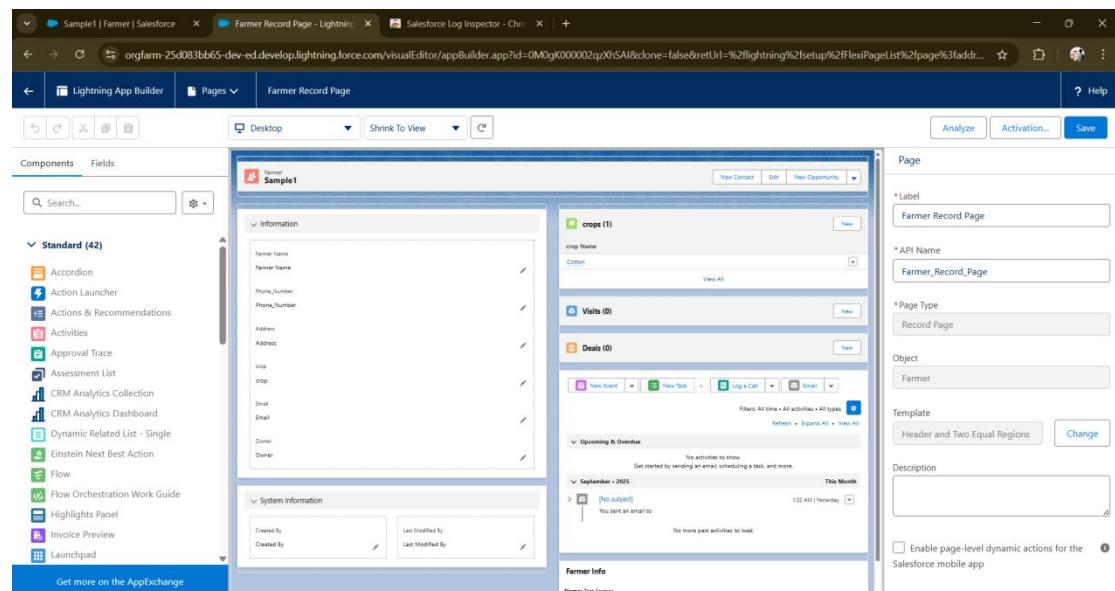
Right sidebar:

Related List — Single: select **Deals** (show which deals reference this crop)

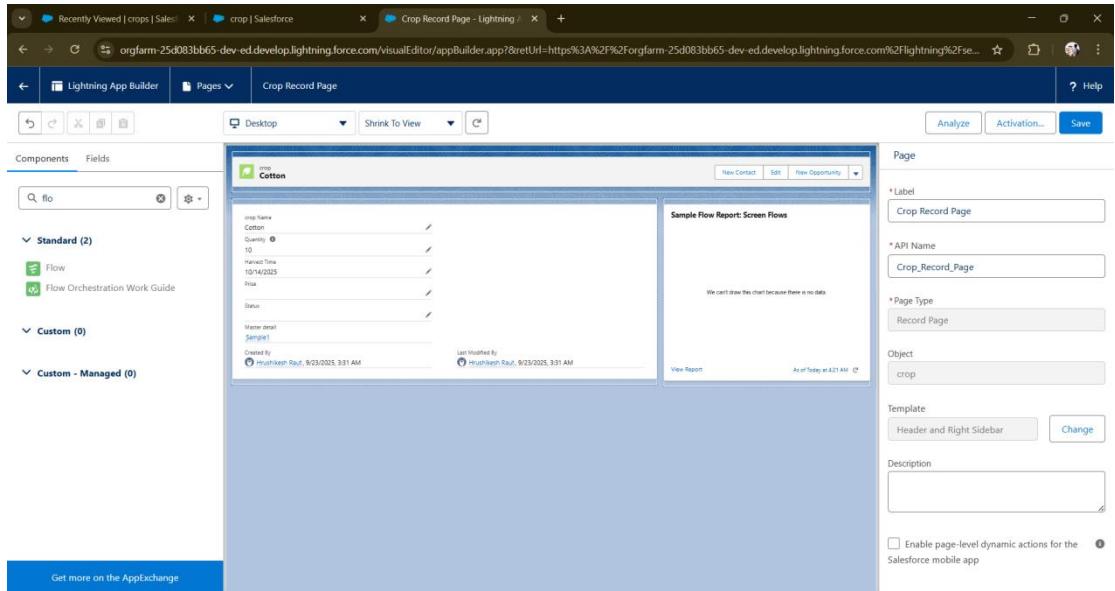
Report Chart: show a small chart (e.g., Crop sales trend or low-stock chart).

5. Save → Activate → assign app & profiles (eg : AgriConnect app & profiles.)

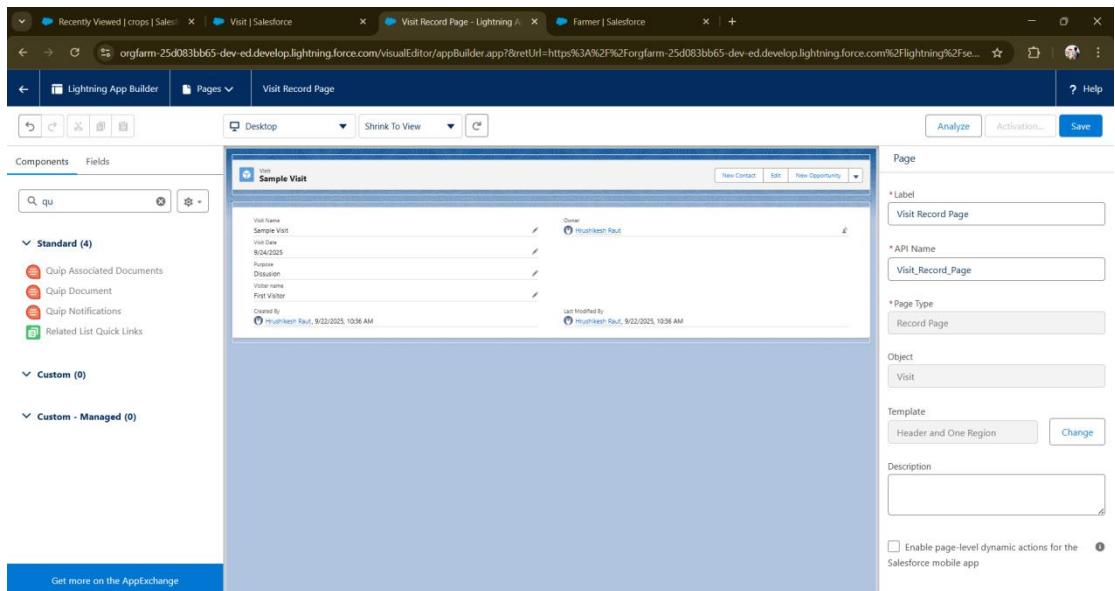
Farmer Record Page



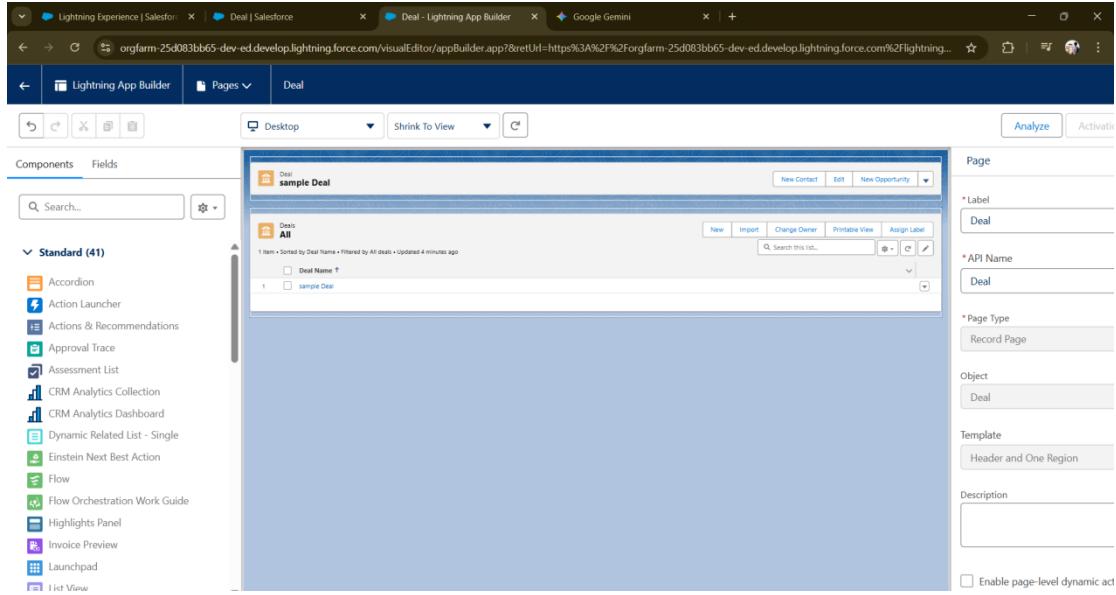
Crop Record Page



Record Page

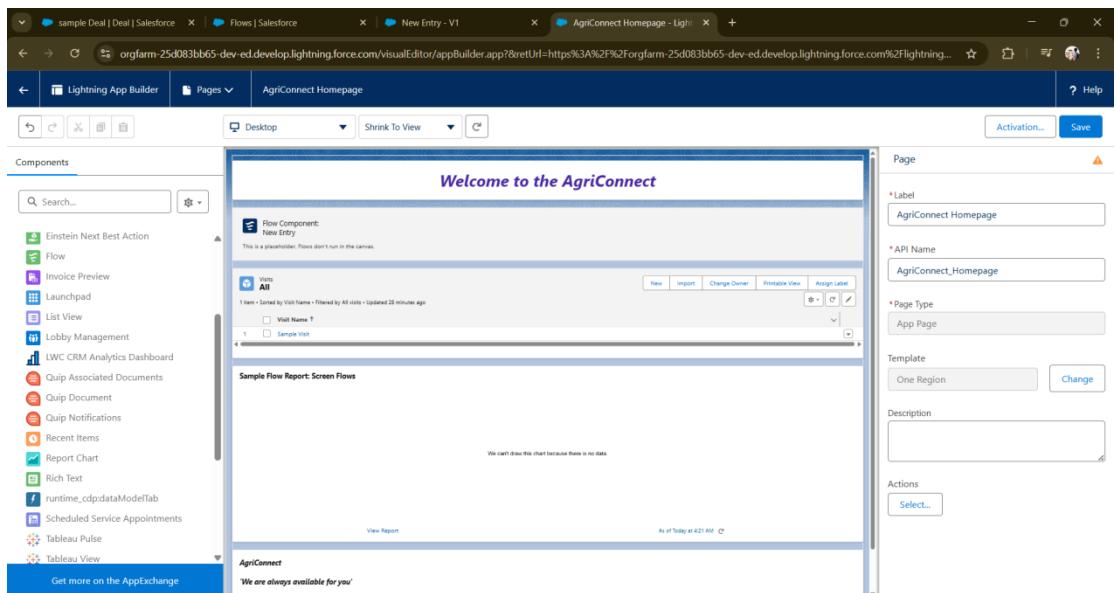


Deal Record Page



AgriConnect HomePage

(This page is a app page not a record page)



❖ *Lightning Web Component (LWC)*

(Use LWC when you need custom UI/UX. Below are small, clear examples you can use in AgriConnect)

Step 1: Install & Open

Install the **Lightning Studio Chrome extension**.

Log in to your Salesforce Org in Chrome.

Open the Lightning Studio extension (you'll see an editor pop up in the org).

□ Step 2: Create a New Component

Click + New → **Lightning Web Component**.

Give your component a name, e.g., farmerCard.

Select where to save → usually under lwc/farmer Card.

The extension will auto-create **3 files**:

farmerCard.html (markup/template)

farmerCard.js (JavaScript logic)

farmerCard.js-meta.xml (exposure config)

□ Step 3 : Write Necessary Code

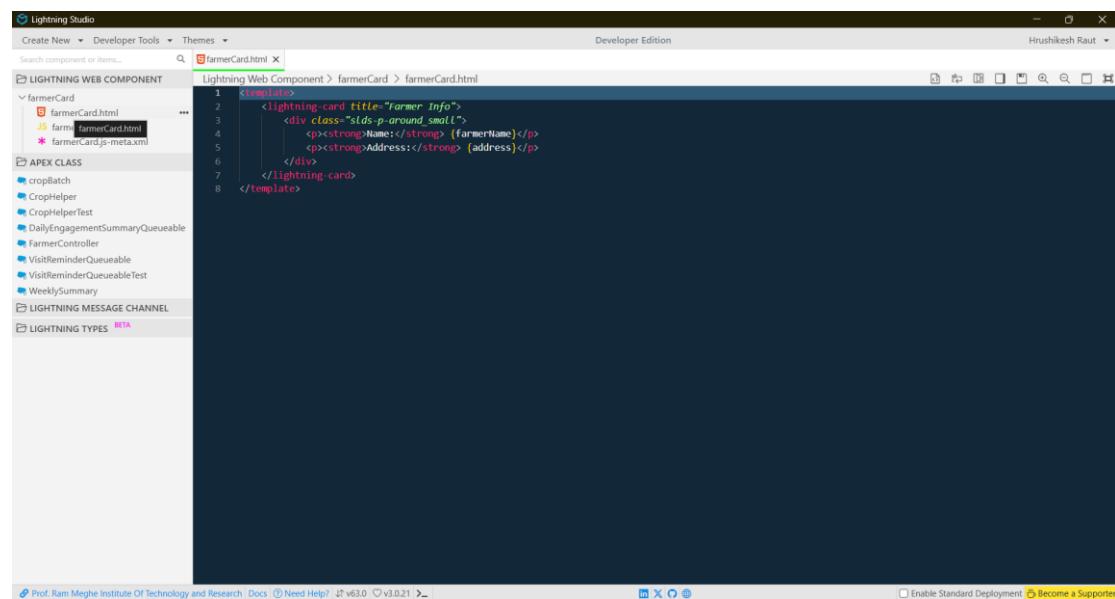
□ Step 4: Save & Deploy

In Lightning Studio, click **Save** → it auto-deploys to your Salesforce org.

If successful, you'll see a confirmation message.

(This saves the setup and configuration of VS studio or Salesforce CLI)

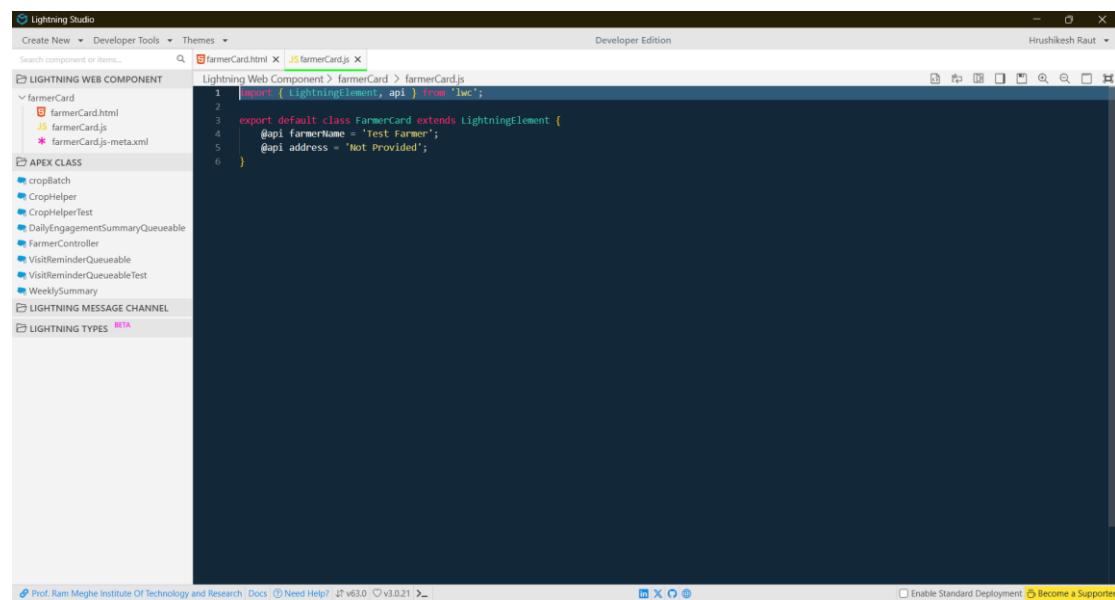
farmerCard.html (markup/template) :



The screenshot shows the Lightning Studio interface with the 'farmerCard.html' component selected. The left sidebar lists components like 'farmerCard', 'APEX CLASS', and 'JS'. The main area displays the component's template code:

```
1 <template>
2   <lightning-card title="Farmer Info">
3     <div class="slds-p-around_smll">
4       <p><strong>Name:</strong> {farmerName}</p>
5       <p><strong>Address:</strong> {address}</p>
6     </div>
7   </lightning-card>
8 </template>
```

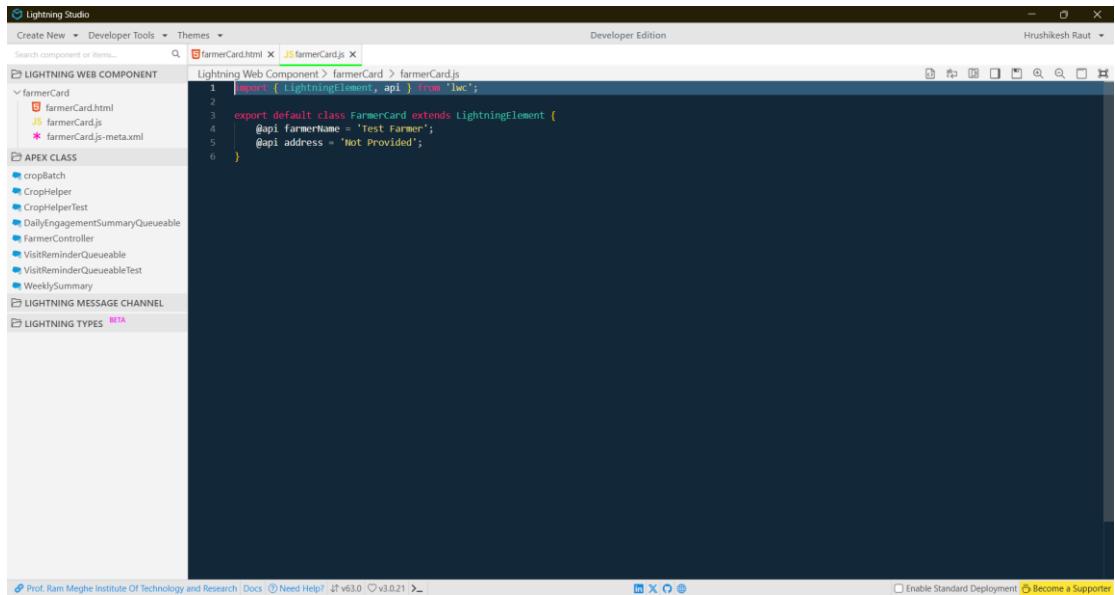
farmerCard.js (JavaScript logic) :



The screenshot shows the Lightning Studio interface with the 'farmerCard.js' component selected. The left sidebar lists components like 'farmerCard', 'APEX CLASS', and 'JS'. The main area displays the component's JavaScript code:

```
1 import { LightningElement, api } from 'lwc';
2
3 export default class FarmerCard extends LightningElement {
4   @api farmerName = 'Test Farmer';
5   @api address = 'Not Provided';
6 }
```

farmerCard.js-meta.xml (exposure config) :



```
Lightning Studio
Create New ▾ Developer Tools ▾ Themes ▾
Search component or items... JS farmerCard.js
Developer Edition Hrushikesh Raut
LIGHTNING WEB COMPONENT
farmerCard
farmerCard.html
JS farmerCard.js
farmerCard.js-meta.xml
APELL CLASS
cropBatch
CropHelper
CropHelperTest
DailyEngagementSummaryQueueable
FarmerController
VisitReminderQueueable
VisitReminderQueueableTest
WeeklySummary
LIGHTNING MESSAGE CHANNEL
LIGHTNING TYPES META
```

```
1 report( LightningElement, api ) from 'lwc';
2
3 export default class FarmerCard extends LightningElement {
4     @api farmerName = 'test Farmer';
5     @api address = 'Not Provided';
6 }
```

Prof. Ram Meghe Institute Of Technology and Research | Docs | Need Help? | v63.0 | v3.0.21 | Enable Standard Deployment | Become a Supporter

(Above is a sample for UI development using LWC)

OR,

❖ Flows

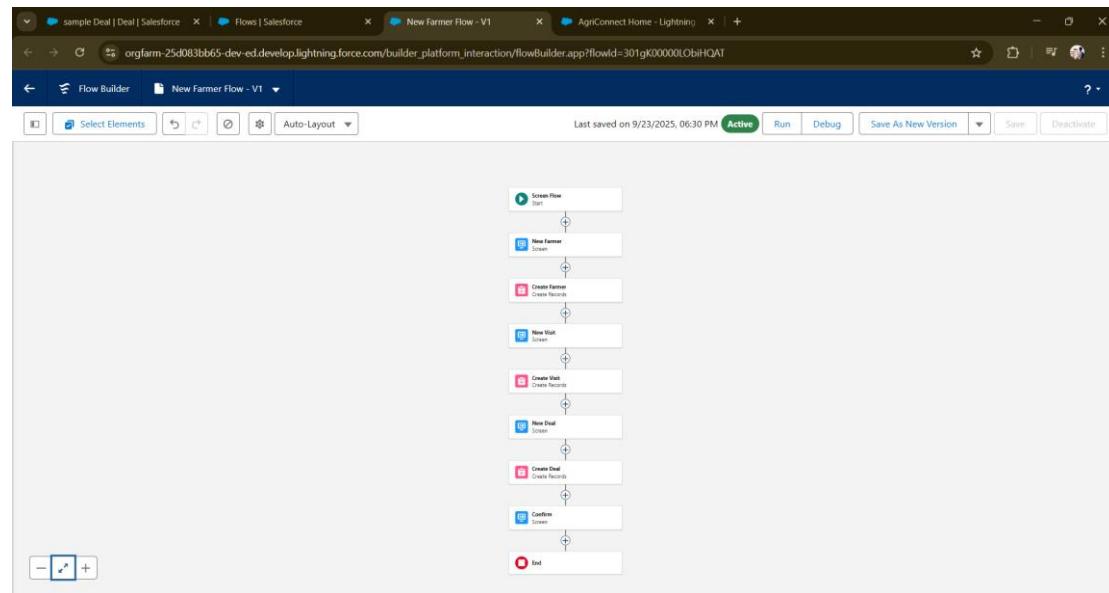
(Flows can be used as an alternative for LWC)

Steps:

1. Navigate to Flows: From the App Launcher, search for Flow
2. Start a New Flow: In the Flows panel, click the New button.
3. Select Flow Type: For Example Choose Screen Flow from the options
4. Add a Screen Element: On the Flow Builder canvas, click the + icon and select the Screen element.
5. Configure the Screen:

6. Give your screen element a unique API Name and a descriptive Label.
7. In the left panel, find the desired components (e.g., Text, Lookup, Email) and drag them onto your screen.
8. For each component, provide a unique API Name and a user-friendly Label.
9. Set components as required if necessary, using the Configure Footer option to customize button labels (like "Next" or "Finish").
10. Add Other Elements (if needed): Use other elements like Create Records or Get Records to interact with Salesforce data and connect them to your screen elements.
11. Save and Activate : Click the Save button in the Flow Builder to save then click Activate

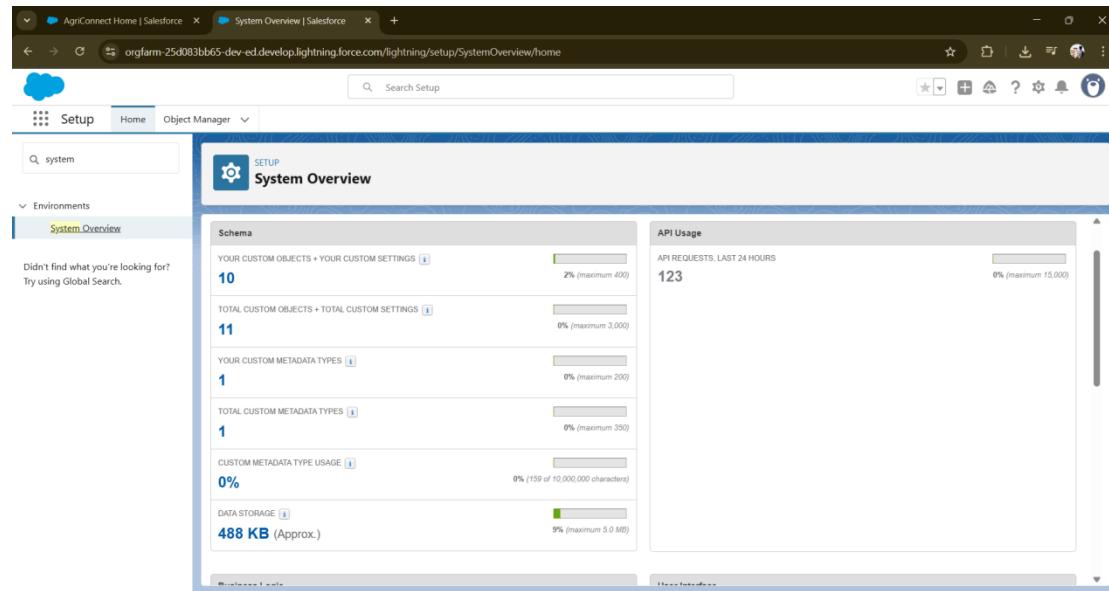
A Screen Flow for new Entry in the records :



Phase 7: Integration & External Access

1. API Limits

- Salesforce tracks the number of **API calls** made by external systems or integrations.
- In **AgriConnect**, all major actions (Farmer Registration, Visits, Buyer related, Deals etc) are handled **inside Salesforce**.
- No external APIs are currently being used in this project, so **API limits are not a concern**. (Below screenshots shows 123 API requests that is because System administrator has some other applications like marketing but our CRM don't)
- You can monitor API usage in **Setup → System Overview**, but no special configuration is required.



Phase 8: Data Management & Deployment

❖ Duplicate Management

Prevent duplicate **Farmer** records (by Email/Phone/Name) and surface potential duplicates during data entry or import.

1) Create the Matching Rule

1. Setup → Quick Find → **Matching Rules** → **New Rule**.

2. **Object:** Farmer

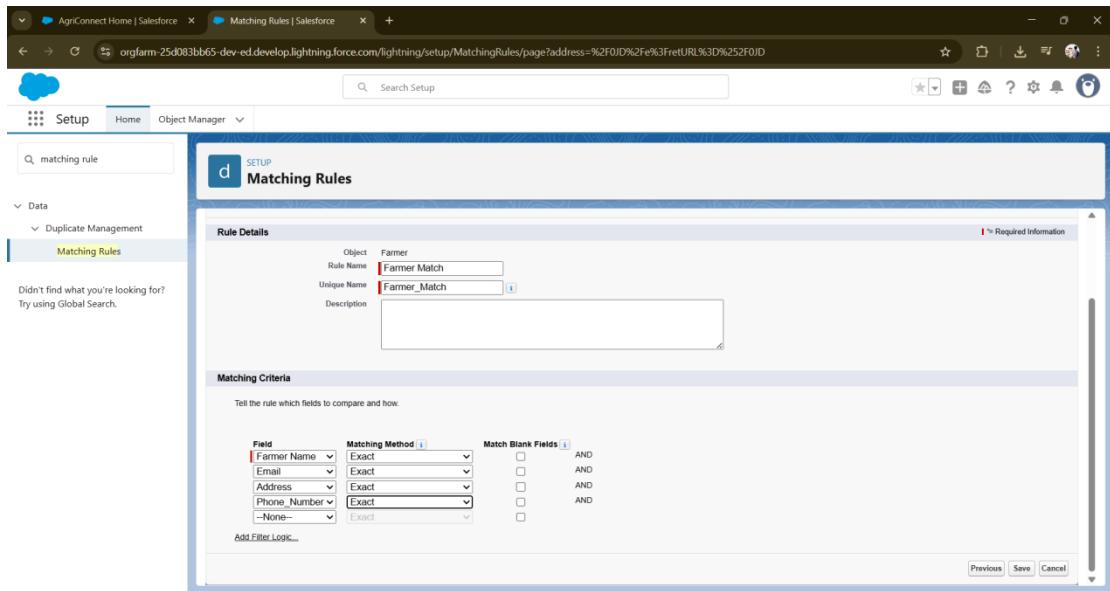
3. **Rule Name:** Farmer Match (or any name).

4. **Add Matching Criteria:**

- Field = **Farmer Name** → Matching Method = **Exact**.
- Field = **Email** → Matching Method = **Exact**.
- Field = **Address** → Matching Method = **Exact**.
- Field = **Phone** → Matching Method = **Exact**.

5. **Save** the rule.

6. Click **Activate** (only active rules can be used by Duplicate Rules).



2) Create the Duplicate Rule

1. Setup → Quick Find → **Duplicate Rules** → New Rule.
2. **Object: Farmer.** (same as that of matching rule)
3. **Rule Label:** Farmer Duplication Rule
4. Under **Matching Rules**, click **Add** and select the ‘Farmer Match’ matching rule you just activated.
5. **Action on Create:** choose **Alert** (start in Alert mode while testing).
6. **Action on Edit:** choose **Alert**.
7. **Save**, then click **Activate**.

Matching Rules

Define how duplicate records are identified.

Compare Farmers With: Farmers

Matching Rule: Farmer Match

Matching Criteria:

```
(Farmer: Name EXACT MatchBlank = FALSE) AND (Farmer: Email EXACT MatchBlank = FALSE) AND (Farmer: Address EXACT MatchBlank = FALSE) AND (Farmer: Phone Number EXACT MatchBlank = FALSE)
```

Field Mapping: Mapped Selected

Add Rule | Remove Rule

Conditions

Optional: specify the conditions a record must meet for the rule to run.

Field	Operator	Value
-None-	-None-	

AND
AND
AND
AND

Add Filter Logic... | Save | Save & New | Cancel

It will look like this after Activation

Duplicate Rule Detail

Rule Name	Farmer Duplicate Rule
Description	
Object	Farmer
Record-Level Security	Enforce sharing rules
Action On Create	Allow
Action On Edit	Allow
Alert Text	Use one of these records?
Active	<input checked="" type="checkbox"/>
Matching Rule	<input checked="" type="checkbox"/> Farmer Match <input checked="" type="checkbox"/> Mapped

Operations On Create

- Alert Report

Operations On Edit

- Alert Report

Matching Criteria

```
(Farmer: Name EXACT MatchBlank = FALSE) AND (Farmer: Email EXACT MatchBlank = FALSE) AND (Farmer: Address EXACT MatchBlank = FALSE) AND (Farmer: Phone Number EXACT MatchBlank = FALSE)
```

Conditions

Created By: Hrushikesh Raut 9/24/2025, 8:41 AM

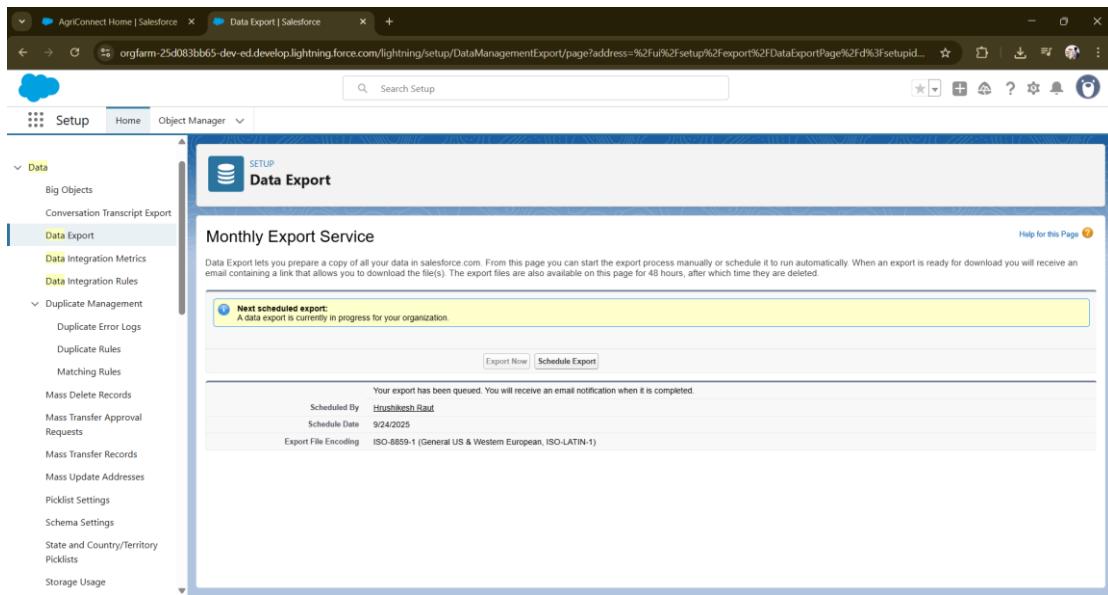
Modified By: Hrushikesh Raut 9/24/2025, 8:41 AM

Edit | Delete | Close | Deactivate

❖ Data Backup

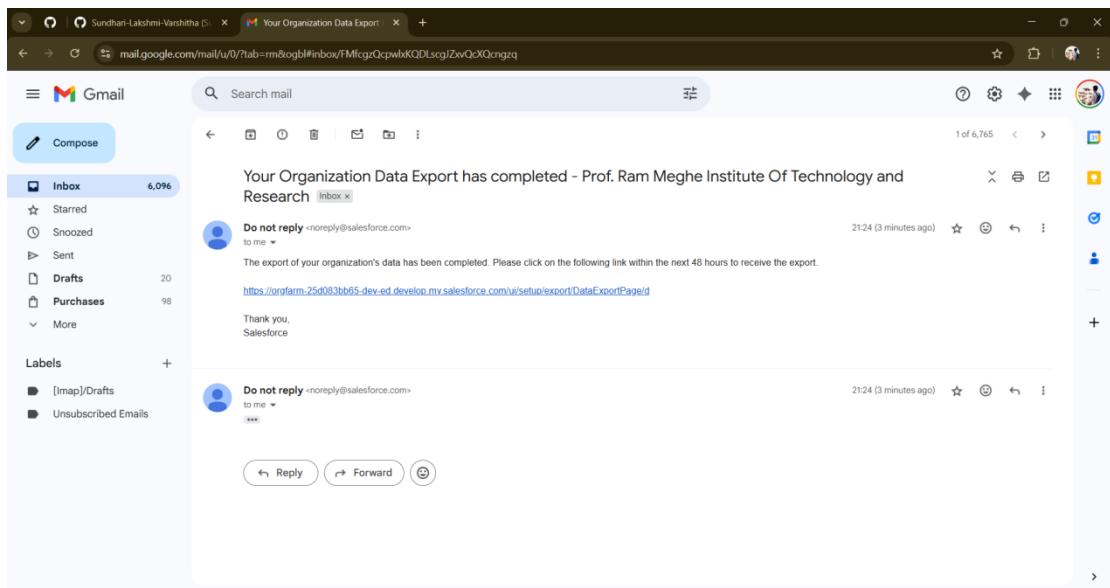
Steps:

- Go to **Setup**.
- In Quick Find, type **Data Export** → click **Data Export**.
- Choose one:
 - **Export Now** → run a one-time backup.
 - **Schedule Export** → set weekly/monthly backups.
- Select the objects you want:
 - **Farmer, Visit, crop, Deal, Buyer** and any standard objects you use (e.g., Users).
- Click **Start Export** (for immediate) or **Schedule Export** (for scheduled).



- Wait → Salesforce emails you when the backup is ready.
- Download the **.zip file** from the export page → extract CSV files.

- Store the backup securely** (encrypted drive, company server, cloud storage).



Extracted ‘CSV’ from zip :

crop_c.csv

Deal_c.csv

Visit_c.csv

Buyer_c.csv

Farmer_c.csv

Phase 9: Reporting Dashboard & Security review

◆ Report Type

Decide which *custom report types* you need beyond standard types (e.g., Farmer with visit or Deal).

Steps: Setup → Report Types → New Custom Report Type

Primary Object: Farmer

Related Object: as per requirement crop/deal/visit (select “Each ‘Farmer’ may or may not have related records”)

Save & Deploy

Why: custom report types let you show object combinations not available in standard report types.

◆ Reports

Reports helps to track effectively in the AgriConnect

A. Create Reports

1. Go to **App Launcher → Reports → New Report.**

2. Select the report type: e.g., **Products** or **Orders**.

3. Add fields:

○ For Farmer Detail Report:

◊ Farmer name

◊ crop

✧ Phone_Number

✧ Address

	Farmer: Farmer Name	crop	Phone_Number	Address
1	Sample1	Cotton	7769052487	xyz lane , village, maharastra, india
2	Rakesh	Cotton	7769052487	North
3	Suresh Raj	Cotton	7769052450	South

○ For New Buyer Report:

✧ Buyer Name

✧ Contact

✧ Created Date

REPORT ▾

New Buyers Report ✓ Buyers

Fields

Outline

Filters 1

Groups

GROUP ROWS

Add group... Q

Columns

Add column... Q

Buyer: Buyer Name X

Contact X

Buyer: Created Date X

Previewing a limited number of records. Run the report to see everything.

Buyer: Buyer Name Contact Buyer: Created Date

1 Sample Buyer 1234567890 9/23/2025

Update Preview Automatically

Save & Run Save Close Run

- o For New Farmer Deal Report :

❖ Farmer Name

❖ Deal Name

❖ Crop

❖ Price

❖ Quantity

❖ status

	Farmer Name	Deal Name	crop	Price	Quantity	status
1	Rakesh	sample Deal	Grapes	\$412	20	Negotiation
2	Suresh Raj	sample Deal 2	Apple	\$320	50	Negotiation
3				\$732	70	

◆ Dashboard

Dashboards provide the visual insights.

A. Create Dashboard

1. Go to App Launcher → Dashboards → New Dashboard.
2. Enter dashboard name, e.g. AgriConnect Dashboard.
3. Add components using your reports as data sources (charts, tables, graphs).

4. Configure components to show key metrics:

- Farmer Details
- Buyer Details
- New Farmer Deals

5. Arrange components for clear visualization.

6. Save the dashboard

Farmers Details Report

Farmer: Farmer Name	crop	Phone_Number	Address
Rakesh	Cotton	7789052487	North
Sample 1	Cotton	7789052487	xyz lane , village, maharastra, india
Suresh Raj	Cotton	7789052450	South

New Buyers Report

Buyer: Buyer Name	Contact	Buyer: Created Date
Buyer 2	9234569890	9/25/2025
Sample Buyer	1234567890	9/23/2025

B. Dashboard Benefits

- Easy visualization about new records.
- Helps identify records and issues quickly.
- Supports faster decision-making for officer.
- Enhances Monitoring, eg. Officer can keep an eye.

New Buyers Report

Buyer: Buyer Name ↑	Contact	Buyer: Created Date
Buyer 2	9234569890	9/25/2025
Sample Buyer	1234567890	9/23/2025

New Farmer Deal Report

Farmer Name ↑	Deal Name	crop	Price	Quantity
Rakesh	sample Deal	Grapes	\$412	20
Suresh Raj	sample Deal 2	Apple	\$320	50

- Security is very essential, specially a persons data. Sharing Settings provide the feature about data sharing like who can see, what can see etc. E.g For a Farmer it is private

Sharing Settings

Object	Sharing Rule 1	Sharing Rule 2	Sharing Rule 3
Work Plan	Private	Private	✓
Work Plan Template	Private	Private	✓
Work Step Template	Private	Private	✓
Work Type	Private	Private	✓
Work Type Group	Public Read/Write	Private	✓
Buyer	Private	Private	✓
crop	Controlled by Parent	Controlled by Parent	✓
Deal	Private	Private	✓
Farmer	Private	Private	✓
Junction object	Controlled by Parent	Controlled by Parent	✓
Mentor	Public Read/Write	Private	✓
Order detail	Public Read/Write	Private	✓
Student	Controlled by Parent	Controlled by Parent	✓
Student_1	Public Read/Write	Private	✓
Visit	Private	Private	✓

Other Settings

- Manager Groups: []
- Secure guest user record access: []
- Require permission to view record names in lookup fields: []