PropertyMatch- Real Estate CRM

Project Overview

PropertyMatch – Real Estate CRM is a Salesforce-based application that helps real estate companies manage properties, brokers, and customer interactions in one place. It allows users to search and filter properties by city, price, bedrooms, and bathrooms, and view details like images, broker information, and location on Google Maps. Administrators can create and assign properties to brokers, while automated email alerts keep brokers informed of new assignments. The system also includes reports and dashboards for insights, validation rules and duplicate checks for data quality, and security controls like profiles, roles, and sharing rules. Overall, PropertyMatch makes property management more efficient, transparent, and user-friendly.

Project Objectives

The objective of this Real Estate CRM is to provide a centralized platform for managing properties, brokers, and client interactions. The system enables users to filter and search properties by city, price, bedrooms, and bathrooms, improving the customer experience by offering quick and accurate results. By allowing administrators to create properties and assign brokers, and by automating broker notifications through email alerts, the project ensures faster communication and reduced manual follow-up. Integration with Google Maps adds transparency by showing exact property locations. Overall, this solution brings business value by improving customer management, streamlining the property search and assignment process, enhancing broker accountability, and ensuring better operational efficiency in the real estate workflow.

Phase 1: Problem Understanding & Industry Analysis

1. Requirement Gathering

- Buyers need property search by **city, price, bedrooms, bathrooms**.
- Display **property details, images, broker info, and map location**.
- Admins should **create properties and assign brokers**.
- **Email alerts** must notify brokers on assignment.
- Navigation: **Home Page, Property Records, Broker Records**.

2. Stakeholder Analysis

- **Primary:** Buyers (search/filter properties), Brokers (assigned properties, email alerts), Admins (manage properties/brokers).
- **Secondary:** Salesforce Platform (data & automation), Email Services (notifications).

3. Business Process Mapping

- *Current (Manual):* Scattered property info, no map visibility, no broker automation.
- *Proposed (Automated):* LWC-based search filters → Property Records → Broker assignment → Google OAuth 2.0 maps → Email alerts.

4. Industry-Specific Use Case Analysis

- Existing apps: generic property listings.
- Gaps: lack of Salesforce-native solution with **map integration + broker automation**.
- Solution: Custom Salesforce LWC app with filters, maps, and email alerts.

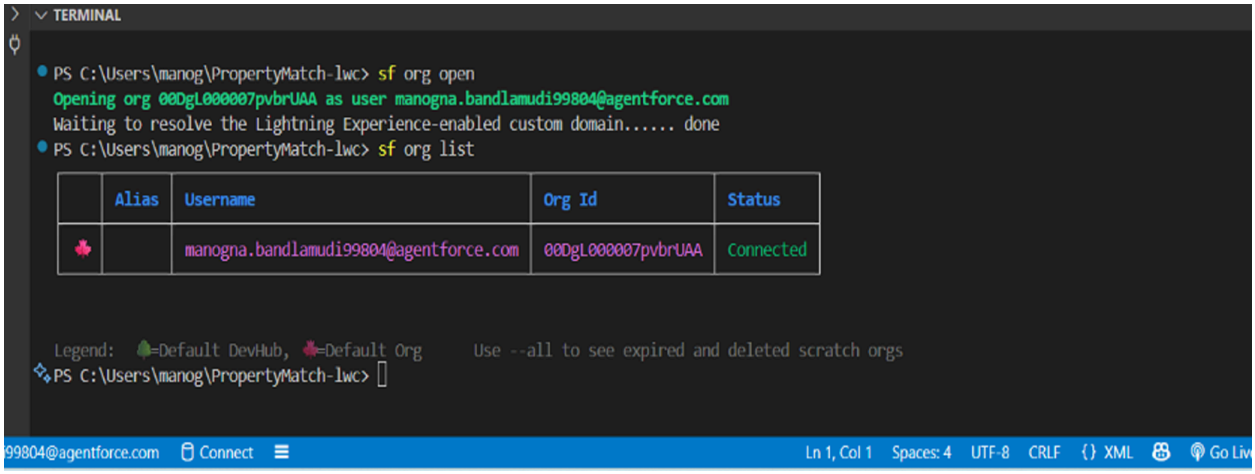
5. AppExchange Exploration

- Explored property apps.
- Found missing features (Google maps, broker email automation).
- Decision: **Custom build on Salesforce CRM**.

Phase 2: Org Setup & Configuration

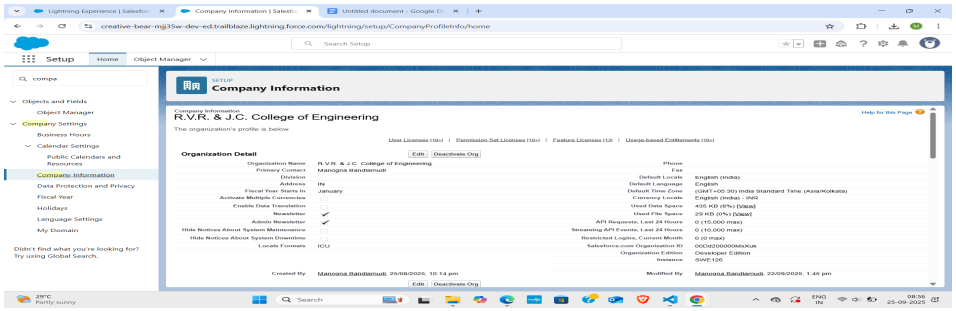
1. Salesforce Edition & Org Setup

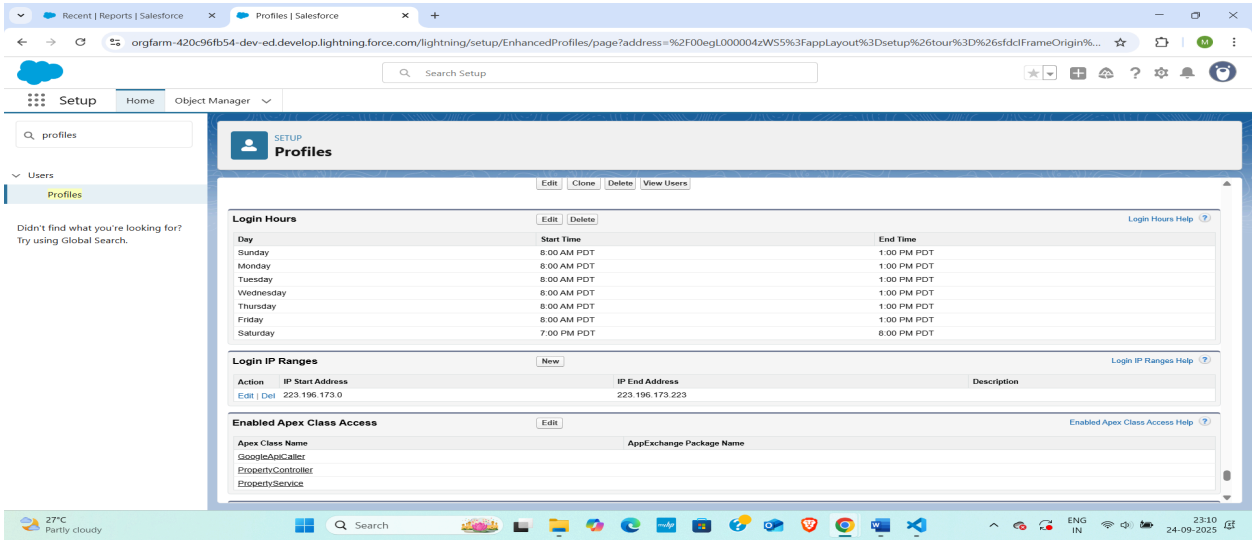
- Created a **Developer Org** for the Dreamhouse LWC project.
- Used it for building, testing, and deploying CRM components.



2. Company Profile Setup

- Navigation: **Setup → Company Information → Edit**.
- Updated organization details (Name, Default Time Zone).
- Purpose: Ensure correct org identity and time settings for automation.



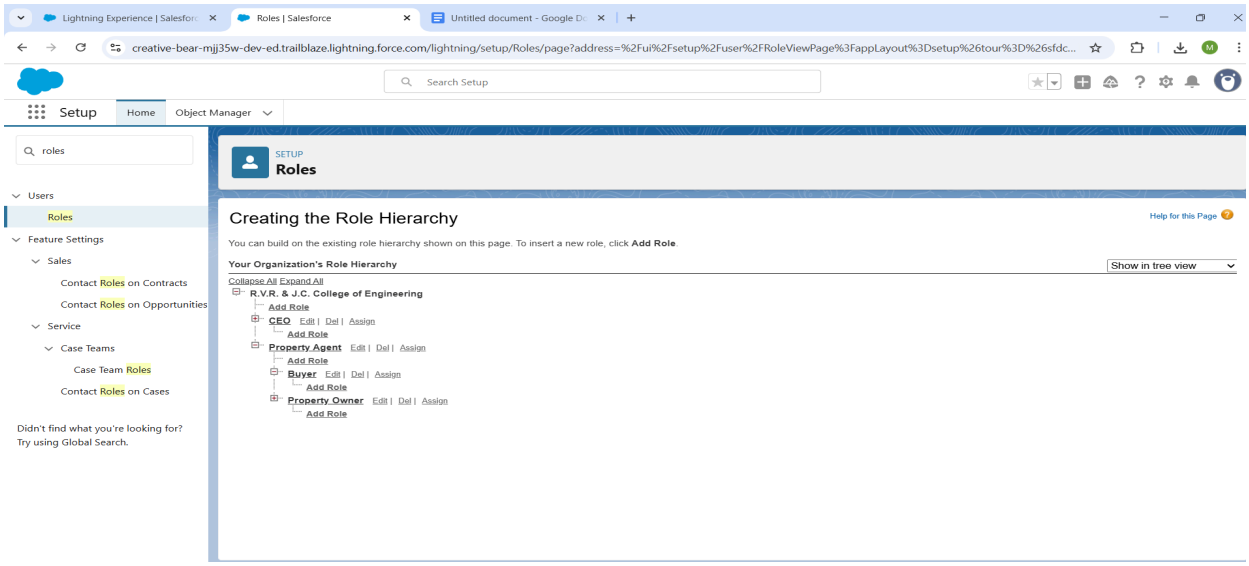


3. Business Hours & Holidays

- Navigation: **Setup → Business Hours → New**.
- Configured business hours (09:00 – 18:00).
- Added holidays (Independence Day, Republic Day).
- Purpose: Time-based automations respect working days.

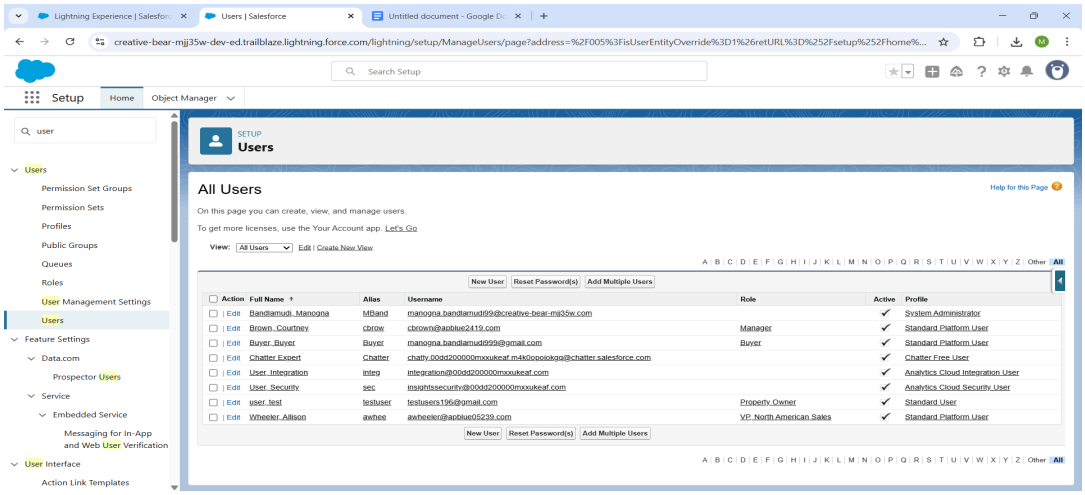
4. Role Hierarchy

- Your final roles will look like:
    - Property Agency Manager(RVRJCCE)
        - Property Agent
            - Buyer
            - Property Owner



5. User Setup & Licenses

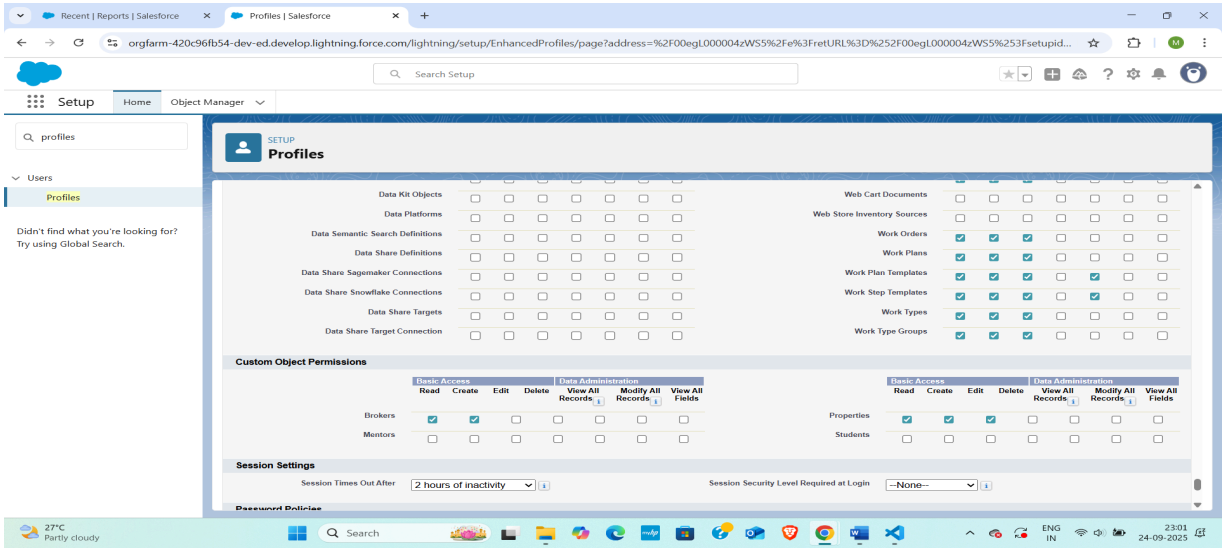
- Navigation: **Setup → Users → New User**.
- Created different users:
  - **Admin** → manages properties and brokers.
  - **Broker** → receives property assignments and email alerts.
  - **Property Manager** → manages records and assignments.
- Assigned profiles and licenses to each user.



6. Configure Profiles (Object Permissions)

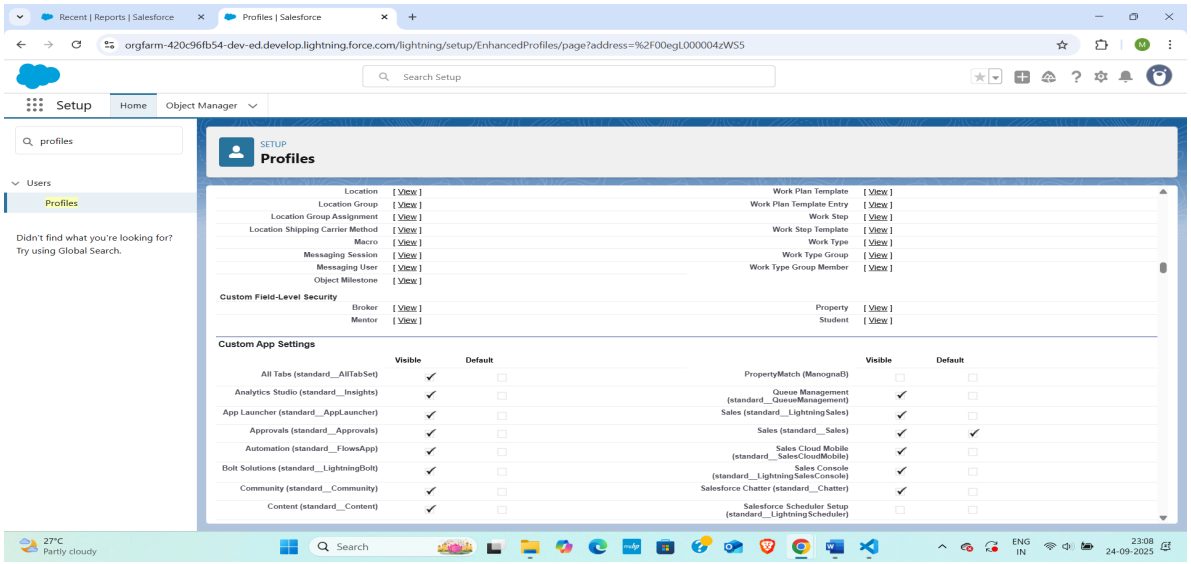
Quick Find → Profiles → open a profile (e.g., Standard User), Scroll to Object Settings → select object Property__c → click Edit. Grant permissions: Read, Create, Edit (as required), Click Save

- For System Administrator Profile: Full access enabled by default.
- For Standard User Profile:
    - Object Permissions (after creating objects): Read, Create, Edit on Property, Buyer, and Visit objects.
    - **Field-Level Security** to allow key fields visible/editable (after objects are created).



7. Login Access Policies

- Navigation: **Setup** → **Login Access Policies**.
- Enabled admin login as other users for testing.



8. Sandbox / Deployment

- Used **VS Code with SFDX** for source push and pull.
- Deployment steps:
  - Edited components in VS Code.
  - Command: `sfdx force:source:push` to deploy changes.

Phase 3: Data Modeling & Relationships

1. Custom Objects Created

- Navigation: **Setup → Object Manager → Create → Custom Object**
- Objects:
  - **Property__c** → Stores property details (city, state, price, bedrooms, bathrooms, latitude, longitude, broker, image).
  - **Broker__c** → Stores broker details (name, email, phone, assigned properties).

2. Custom Fields (Property__c)

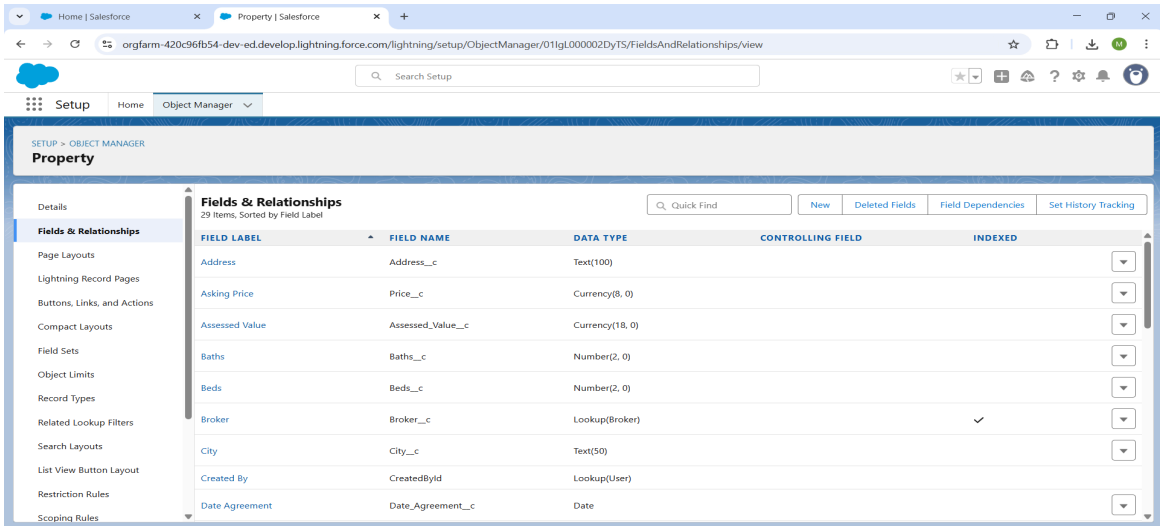
- City__c (Text) – City name.
- State__c (Text) – State name.
- Price__c (Currency) – Property price.
- Bedrooms__c (Number) – No. of bedrooms.
- Bathrooms__c (Number) – No. of bathrooms.
- Latitude__c / Longitude__c (Number) – Coordinates for map display.
- Image_URL__c (Text) – Store property image link.
- Broker__c (Lookup → Broker__c) – Assigned broker.

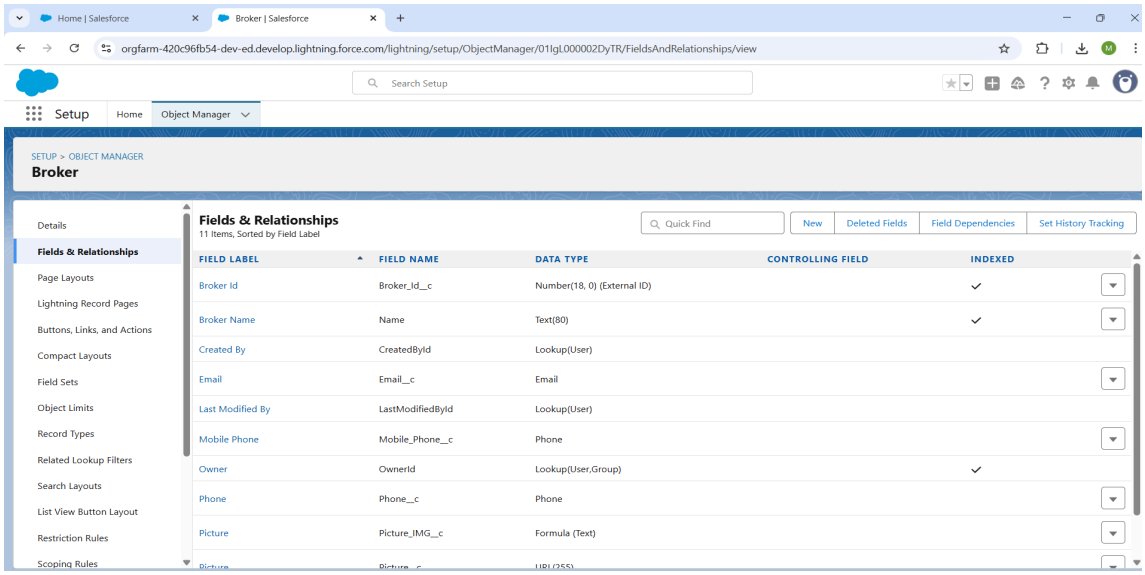
Custom Fields (Broker__c)

- Email__c (Email) – Broker's email (used in email alerts).
- Phone__c (Phone) – Contact number.
- Assigned_Properties (Related List from Property__c).

3. Relationships

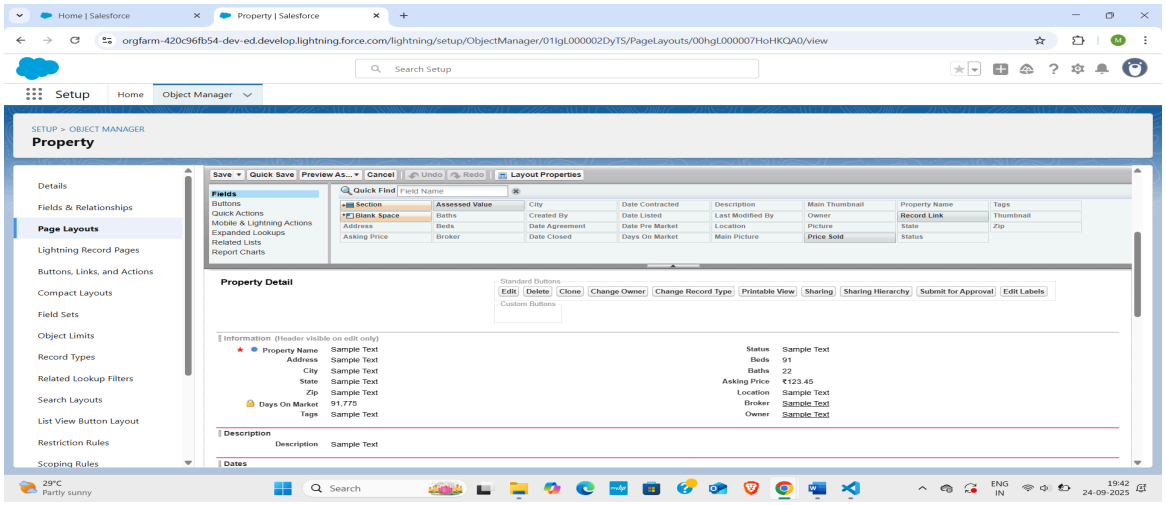
- **Lookup:** Property__c → Broker__c (each property linked to one broker).
- **Master-Detail:** Not required, as broker deletion should not delete properties.

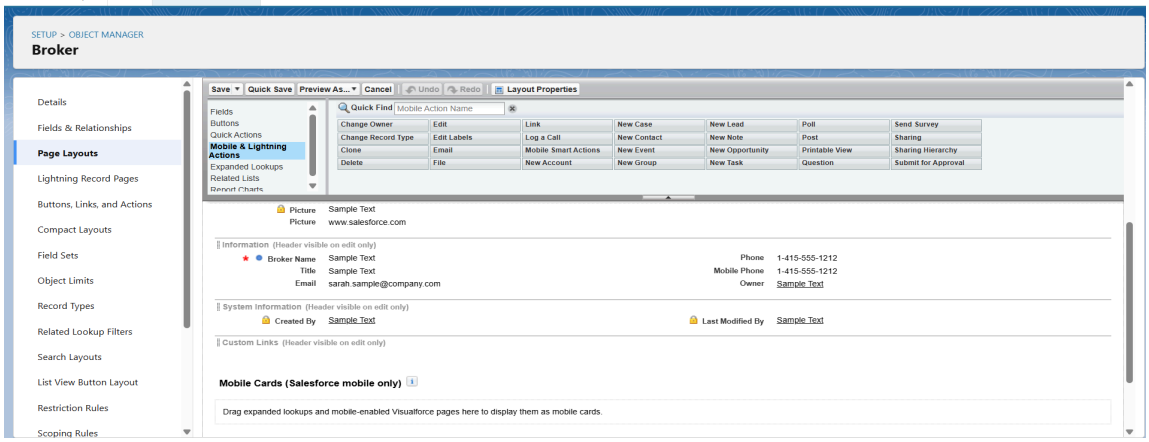




4. Page Layouts

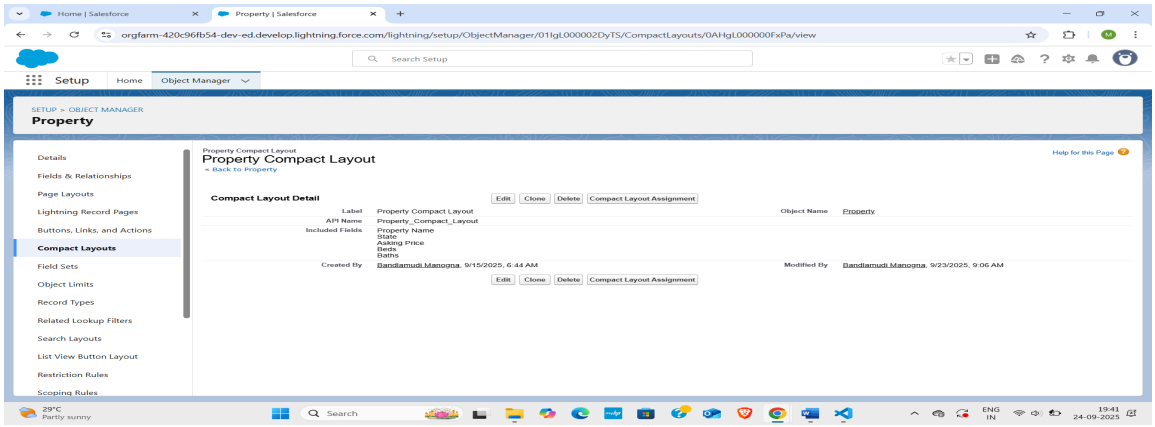
- Navigation: **Setup → Object Manager → [Object] → Page Layouts**
- **Property Layout**: Show all fields + Related List (Broker).
- **Broker Layout**: Show contact info + Related List (Assigned Properties).

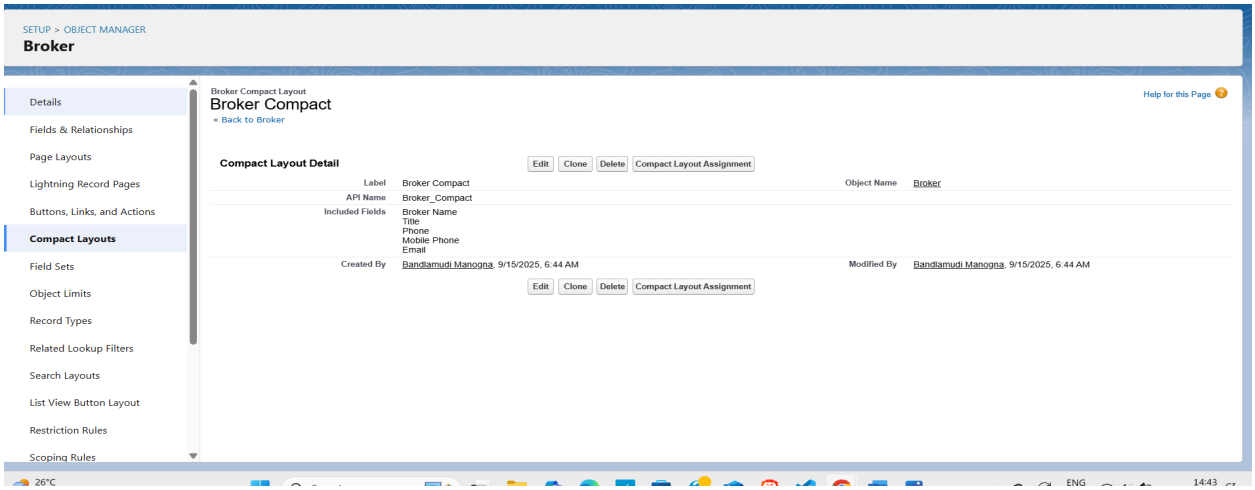




5. Record Types & Compact Layouts

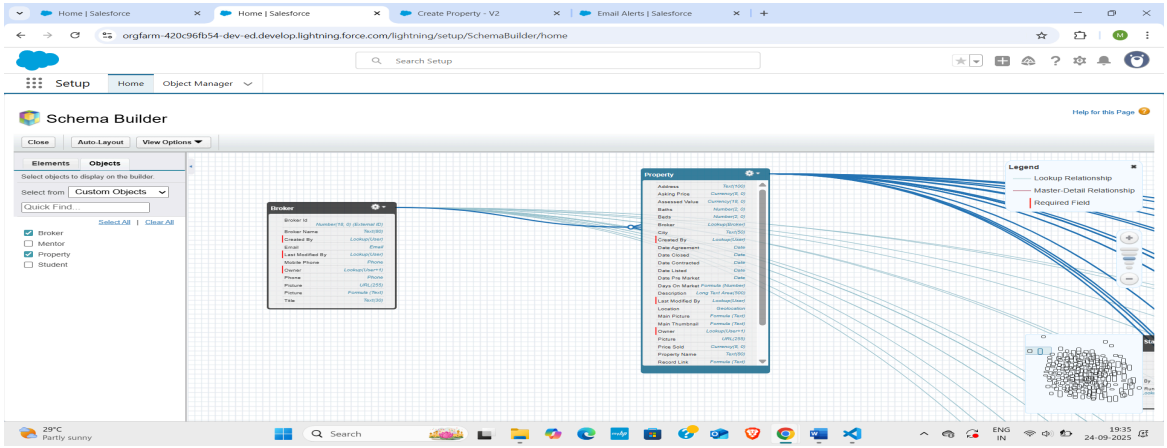
- Created default Record Type for Property__c.
- Compact Layout: Display key fields (City, Price, Bedrooms, Bathrooms) in highlights panel.





6. Schema Builder

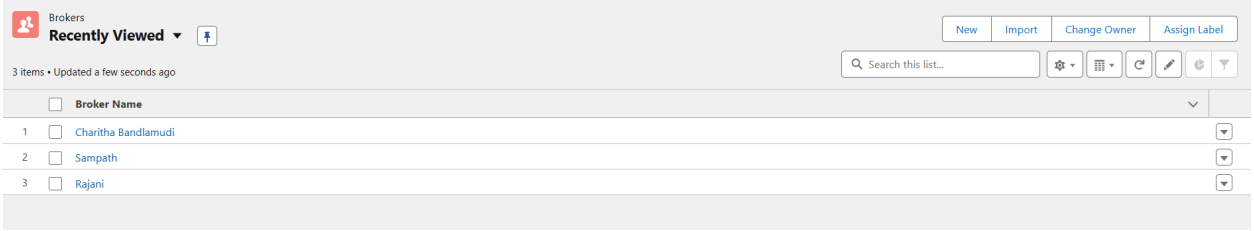
- Navigation: **Setup → Schema Builder**.
- Verified relationship between Property__c and Broker__c.



7. Add Sample Records

- **Properties:**
  - Property 1 → City: Hyderabad, Price: 50,00,000, Bedrooms: 3, Bathrooms: 2, Status: Available
  - Property 2 → City: Bangalore, Price: 75,00,000, Bedrooms: 4, Bathrooms: 3, Status: Available

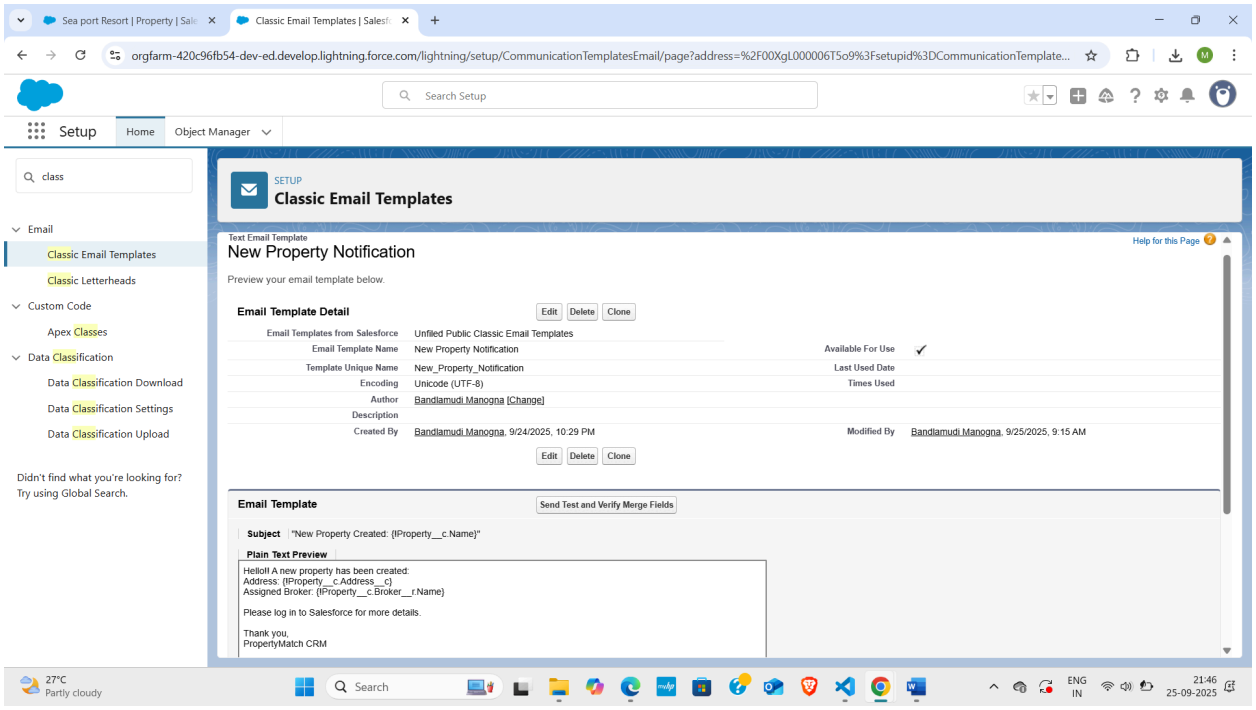
- **Brokers:**
    - Broker 1 → Name: charitha, Email: charitha@propertymatch.com, Phone: 9876543210
    - Broker 2 → Name: sampath, Email: sampath@propertymatch.com, Phone: 9876501234
- **Link Property to Broker:**
    - Property 1 (Hyderabad) → Assigned to Broker 1 (charitha)
    - Property 2 (Bangalore) → Assigned to Broker 2 (sampath)



Phase 4: Process Automation (Admin)

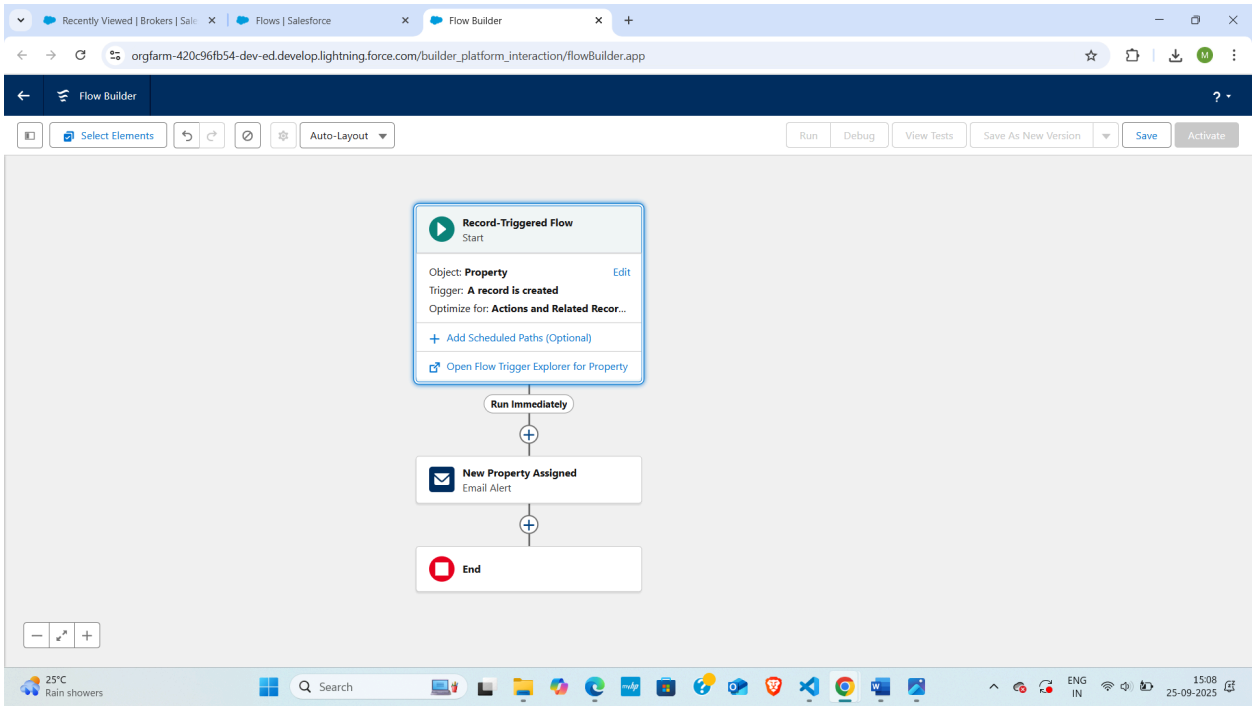
1. Email Alerts to Brokers

- **Purpose:** Notify brokers automatically when a property is assigned.
- **Navigation:**
    - Setup → Email Alerts → New Email Alert.
    - Choose object **Property__c**.
    - Selected **Broker Email__c** field.
    - Attached template: “New Property Assigned – Details Included”.
- **How it works:** When a Property is created with Broker__c field, an email is triggered to that broker.



2. Flow – Trigger Email on Property Assignment

- **Purpose:** Automate broker notification without manual effort.
- **Navigation:**
    - Setup → Flows → New Flow → Record-Triggered Flow.
    - Object: **Property__c**.
    - Trigger: When record is created or updated.
    - Condition: Broker__c field is not null.
    - Action: **Send Email Alert** using created template.
- **How it works:** Whenever a property is assigned to a broker, the broker instantly receives an email with property details.



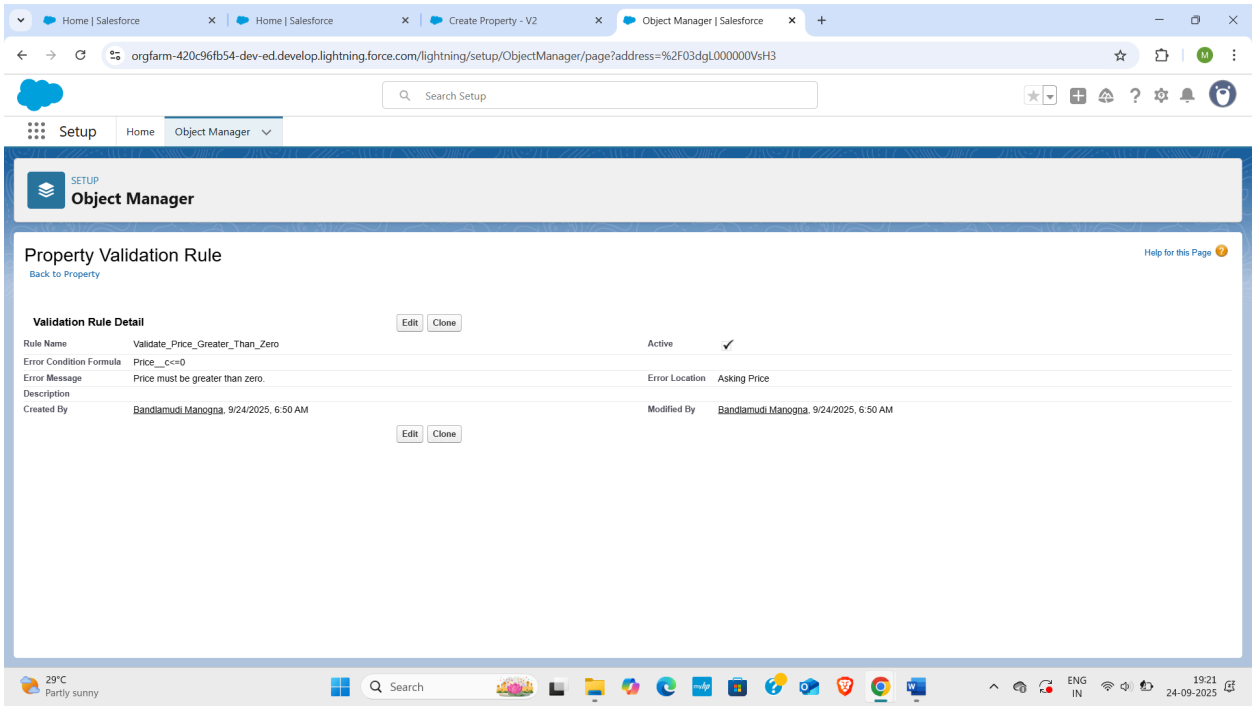
3. Validation Rules

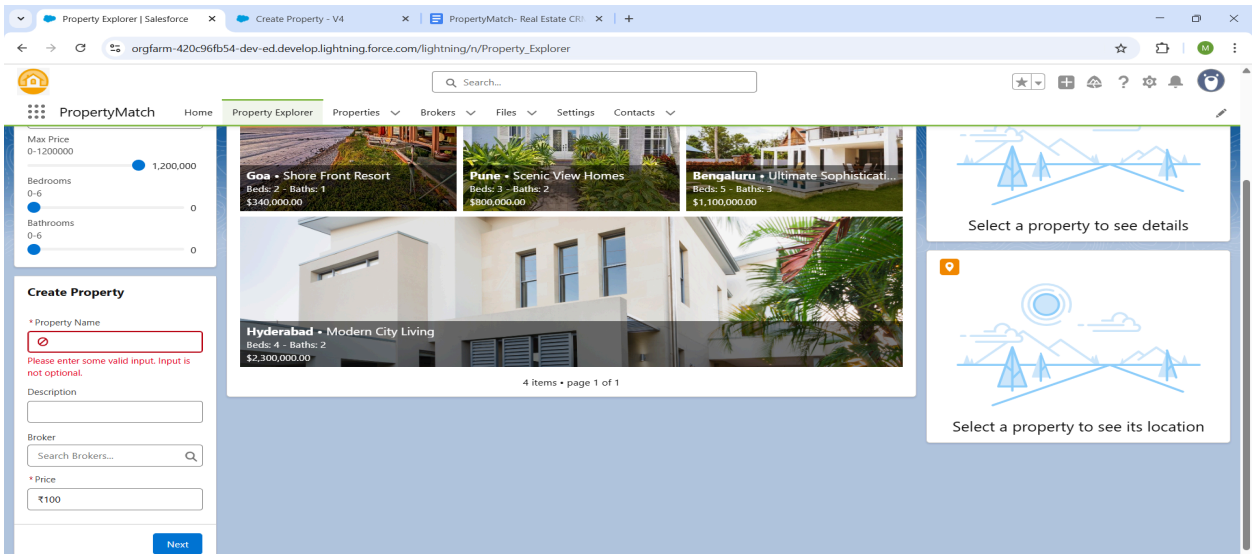
- **Purpose:** Ensure important property details are always filled.
- **Navigation:**

  Setup → Object Manager → Property__c → Validation Rules → New.

- **Rules Created:**
  - `ISBLANK(Price__c)` → Error: “Price must be greater than zero.”
  - `ISBLANK(Name__c)`→ Error: “Please enter some valid input. Input is not optional.”

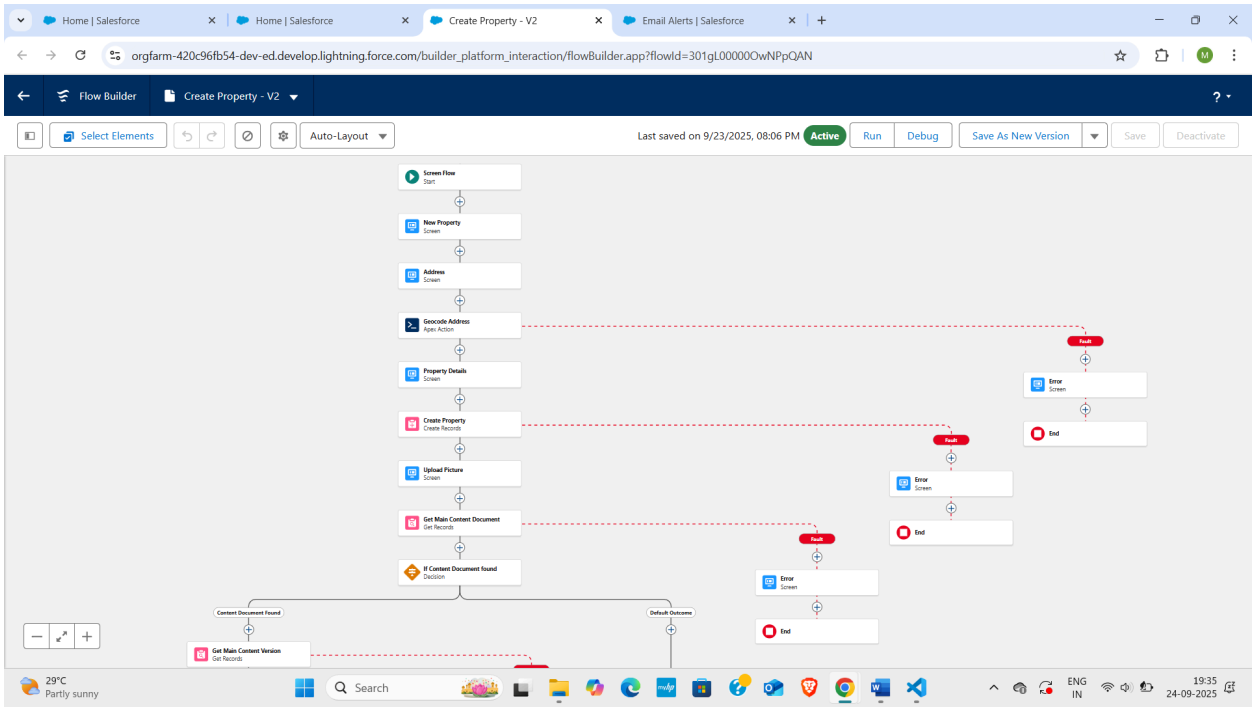
How it works: Prevents saving incomplete property records, ensuring data quality. Give an error page saying we hit a sang, review error on this page and the error details.

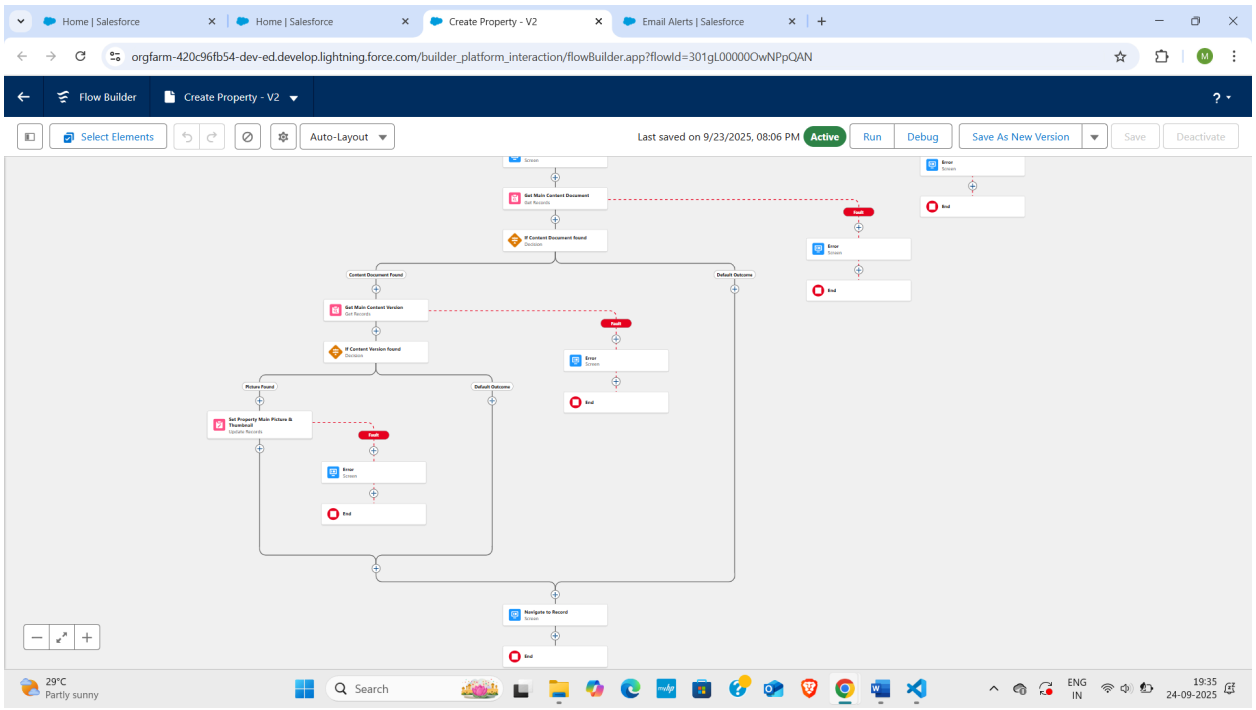




3. Screen Flow:

In the **Property Explorer tab**, **Screen Flows** were implemented to simplify property creation by guiding administrators through step-by-step input of fields such as city, price, bedrooms, bathrooms, and broker assignment, picture upload. This ensures that all required details are captured in an organized manner, reducing errors and improving the property onboarding process.


Home | Salesforce
Home | Salesforce
Create Property - V2
Email Alerts | Salesforce
orgfarm-420c96fb54-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=301gL00000OwNPpQAN
Flow Builder
Create Property - V2
Select Elements
Auto-Layout
Last saved on 9/23/2025, 08:06 PM
Active
Run
Debug
Save As New Version
Save
Deactivate
Screen Flow
Start
New Property
Screen
Address
Screen
Geocode Address
Apex Action
Property Details
Screen
Create Property
Create Records
Upload Picture
Screen
Get Main Content Document
Get Records
If Content Document found
Decision
Content Document Found
Default Outcome
Get Main Content Version
Get Records
Fault
Error
Screen
End
29°C
Partly sunny
Search
ENG
IN
19:35
24-09-2025

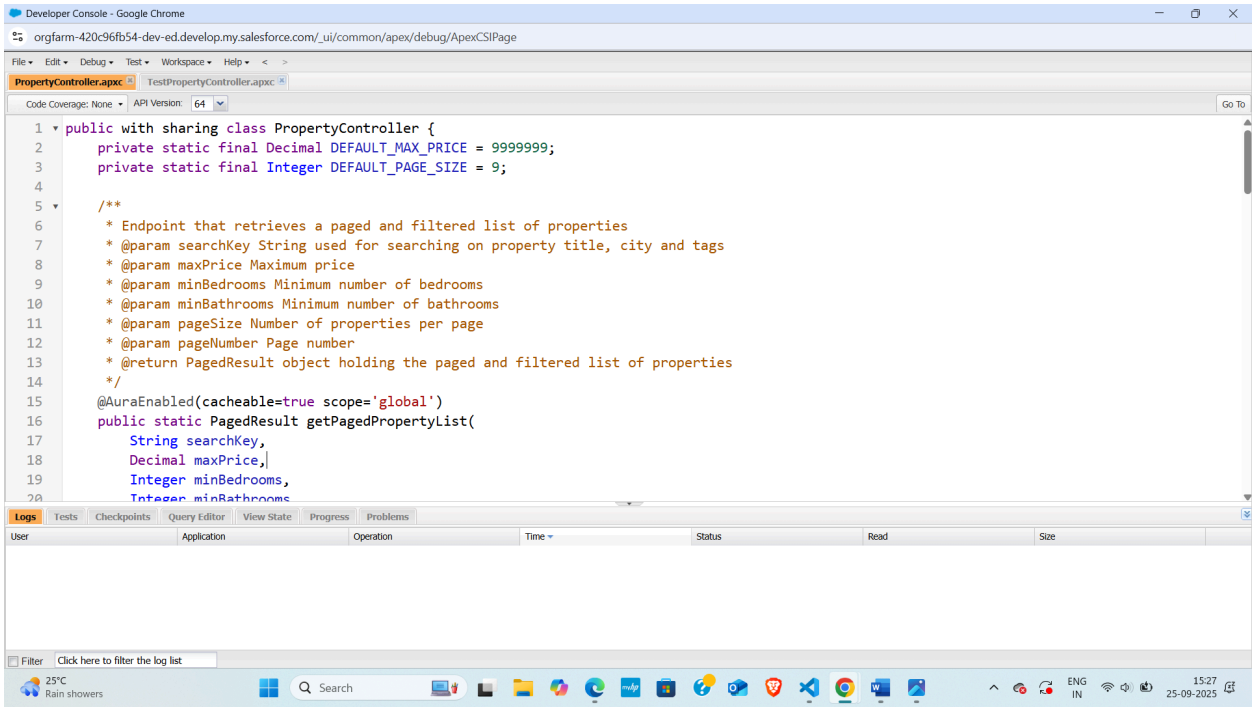

Home | Salesforce
Home | Salesforce
Create Property - V2
Email Alerts | Salesforce
orgfarm-420c96fb54-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=301gL00000OwNPpQAN
Flow Builder
Create Property - V2
Select Elements
Auto-Layout
Last saved on 9/23/2025, 08:06 PM
Active
Run
Debug
Save As New Version
Save
Deactivate
Get Main Content Document
Get Records
If Content Document found
Decision
Content Document Found
Default Outcome
Get Main Content Version
Get Records
If Content Version found
Decision
Picture Found
Default Outcome
Set Property Main Picture & Thumbnail
Update Records
Fault
Error
Screen
End
Navigate to Record
Screen
29°C
Partly sunny
Search
ENG
IN
19:35
24-09-2025

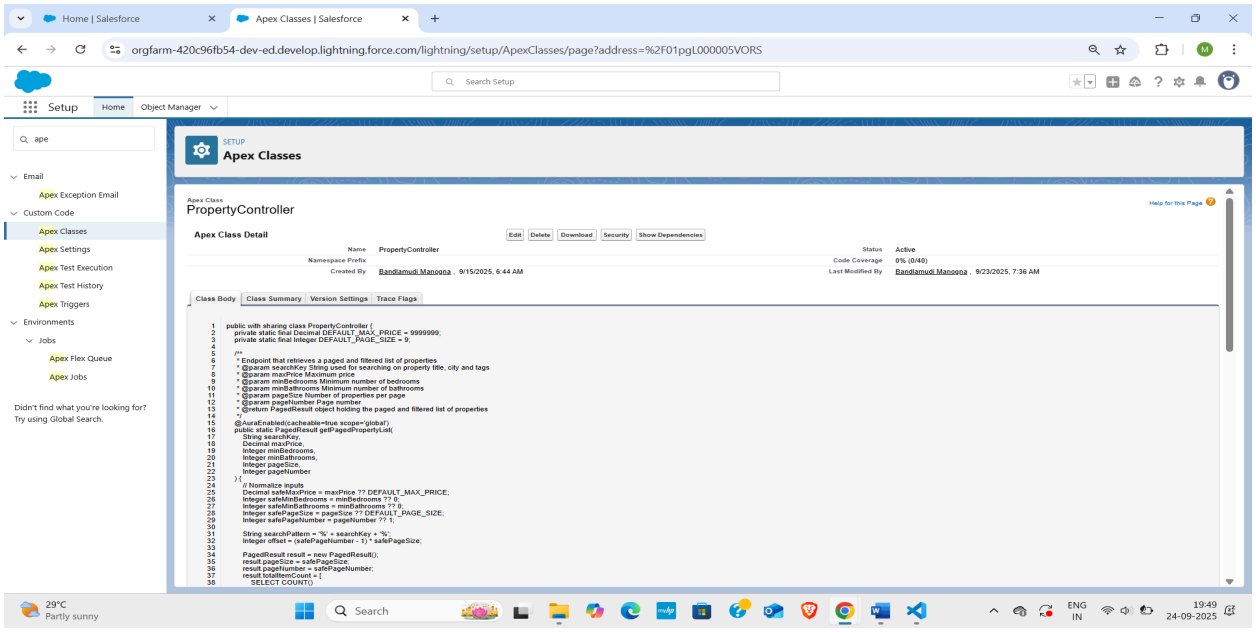
Phase 5: Apex Programming (Developer)

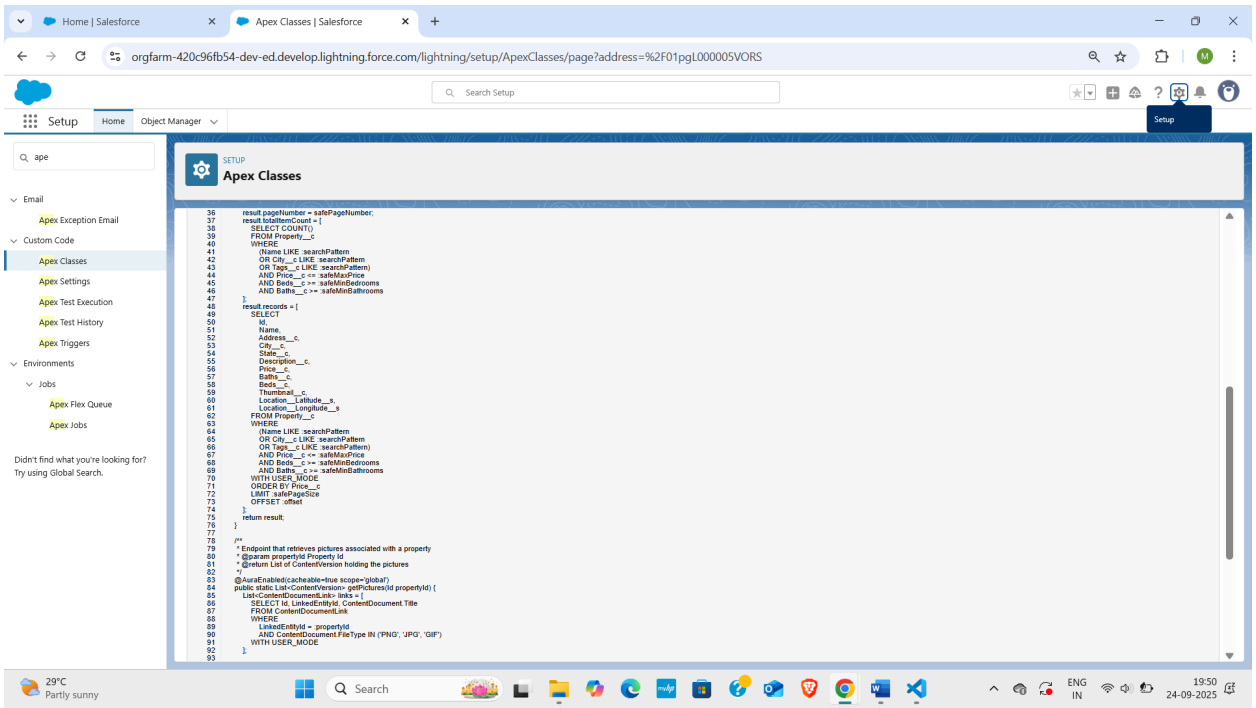
1. Apex Classes Created

a) PropertyController.cls

- **Purpose:** Fetch property records from `Property__c` based on filters.
- **Implementation Steps:**
  - Navigation: **Setup → Apex Classes → New → PropertyController.cls**.
  - Added methods for fetching property data.





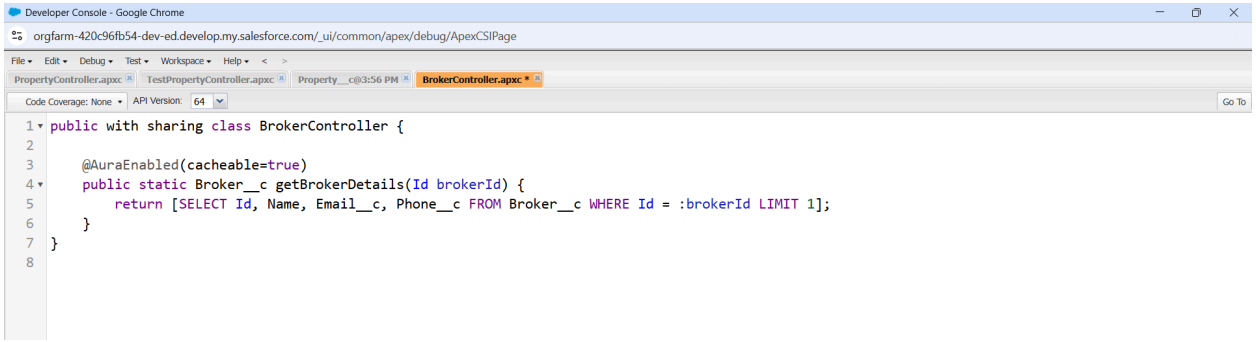


- **How it works:**
    - Accepts filter inputs from LWC (searchKey, maxPrice, bedrooms, bathrooms).

- Dynamically builds SOQL query.
- Returns a list of matching properties.
- Used `@AuraEnabled(cacheable=true)` so LWC can call efficiently.

b) BrokerController.cls

- **Purpose:** Fetch broker details for display and email alerts.
- **Implementation Steps:**
  - Navigation: **Setup → Apex Classes → New → BrokerController.cls**.
- **Code Logic:**



- **How it works:**
  - LWC uses this to display broker details when a property record is opened.

2. Test Classes

Why required:

- Salesforce requires **minimum 75% code coverage** for deployment.
- Validates that Apex methods work with sample data.

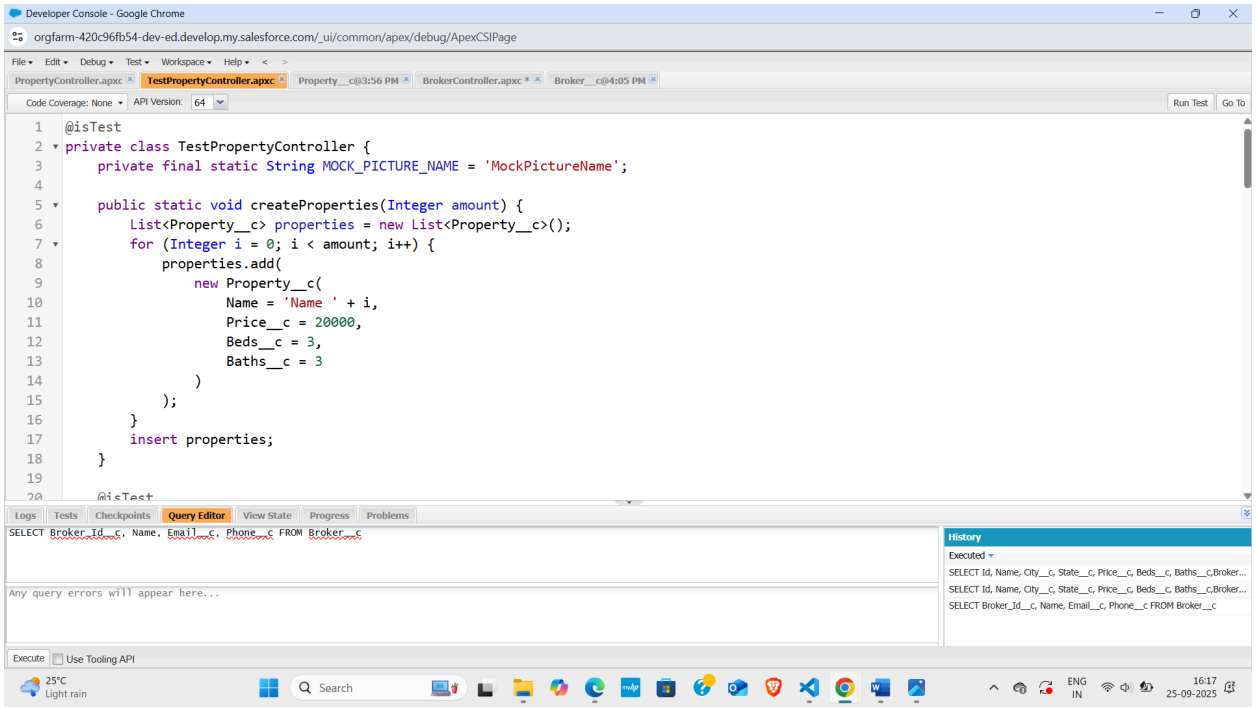
- **PropertyControllerTest.cls**

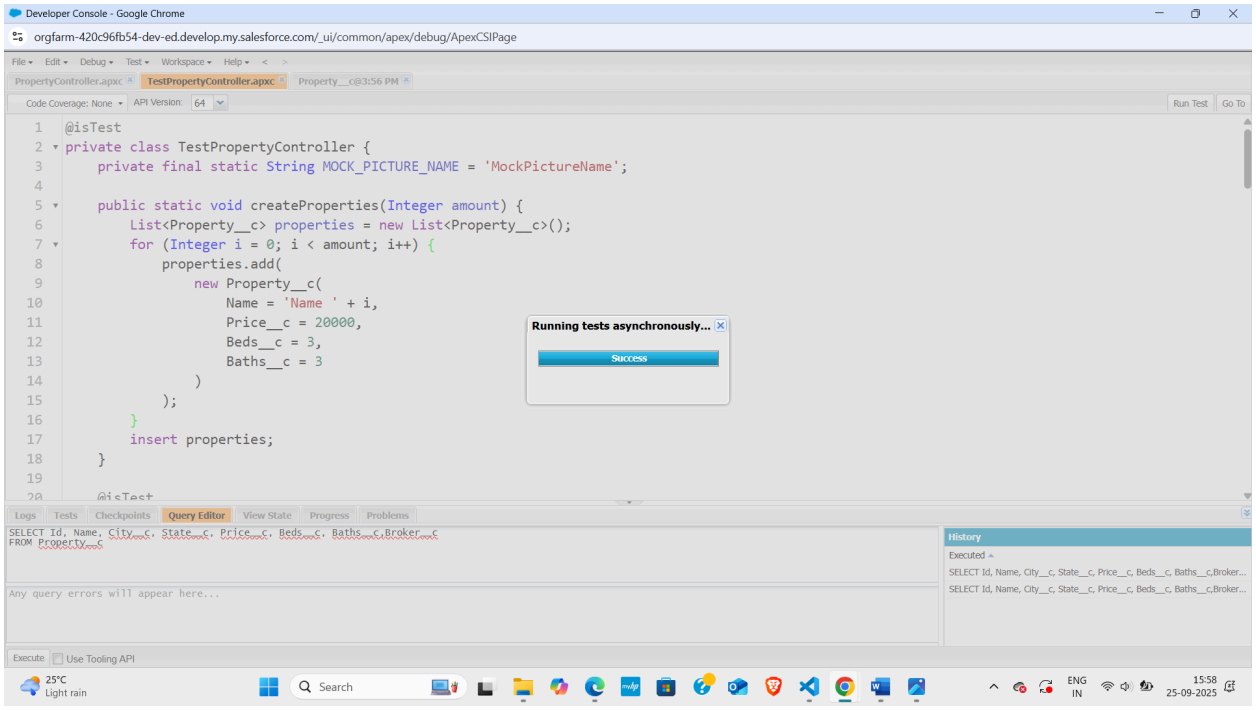
Steps to create Test Class:

1. Click **Setup → Apex Classes → New**.
2. Copy and paste the test class code below.
3. Click **Save**.

Test Logic Overview:

- Creates sample Property and Broker records.
- Tests property filtering and broker fetching.





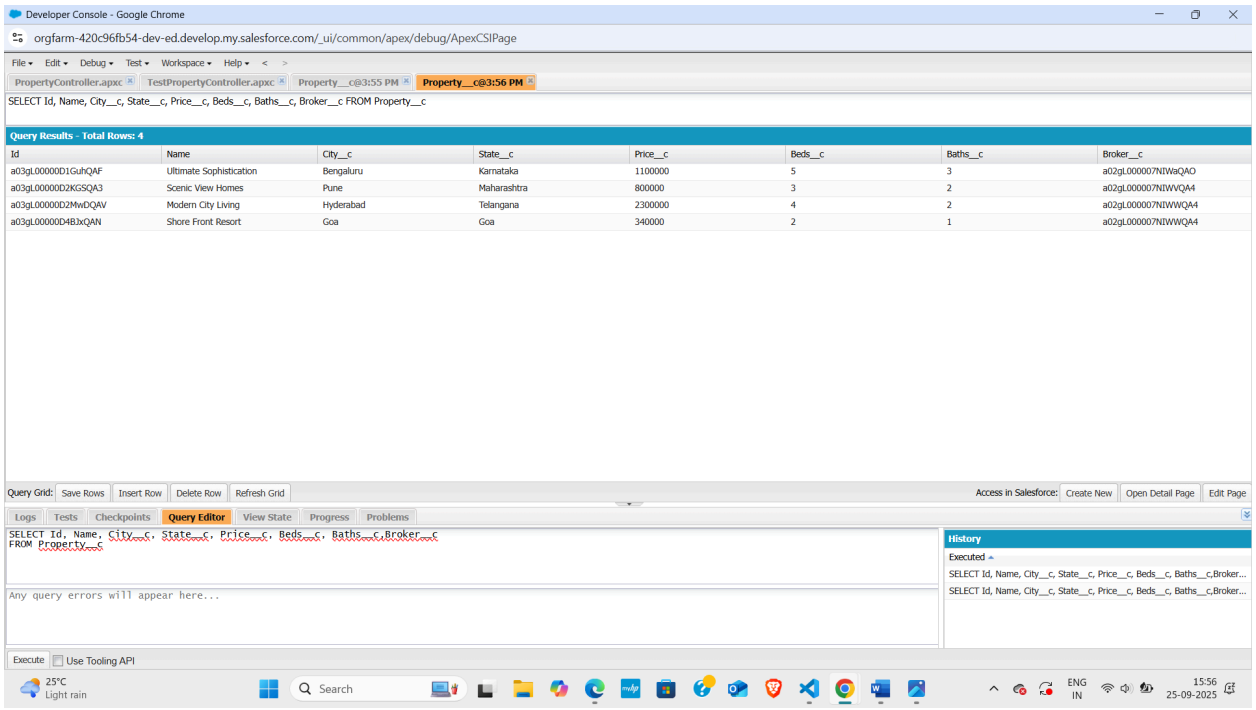
- **BrokerControllerTest.cls**

- **How it works:**
    - Creates test records in memory (not visible in org).
    - Runs Apex methods with test data.

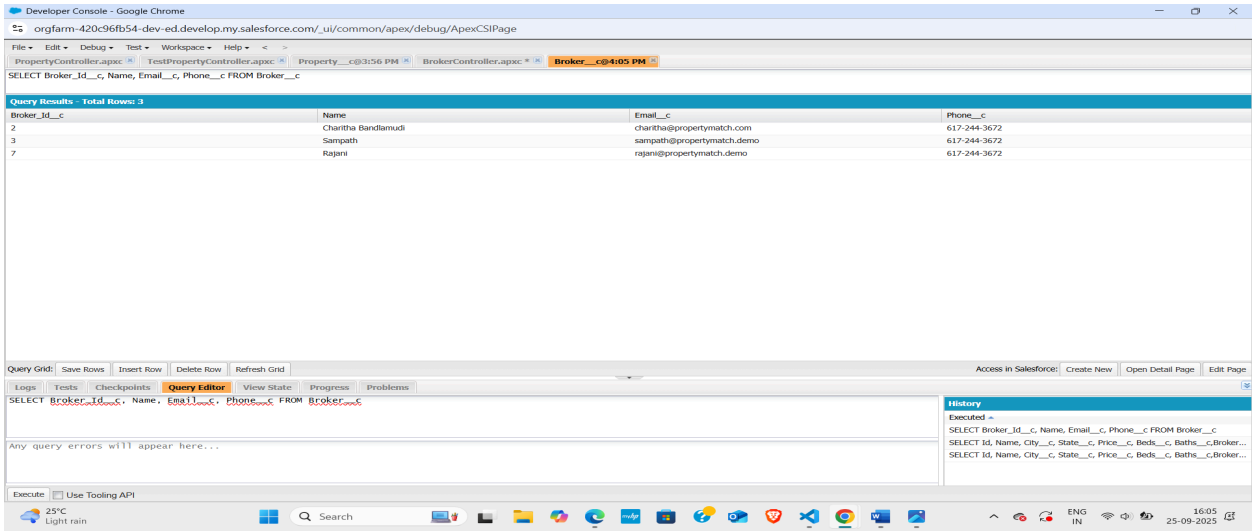
- Asserts results to confirm correct behavior.
- Ensures >75% coverage for deployment.

3. SOQL Queries Used

Fetching properties with filters:

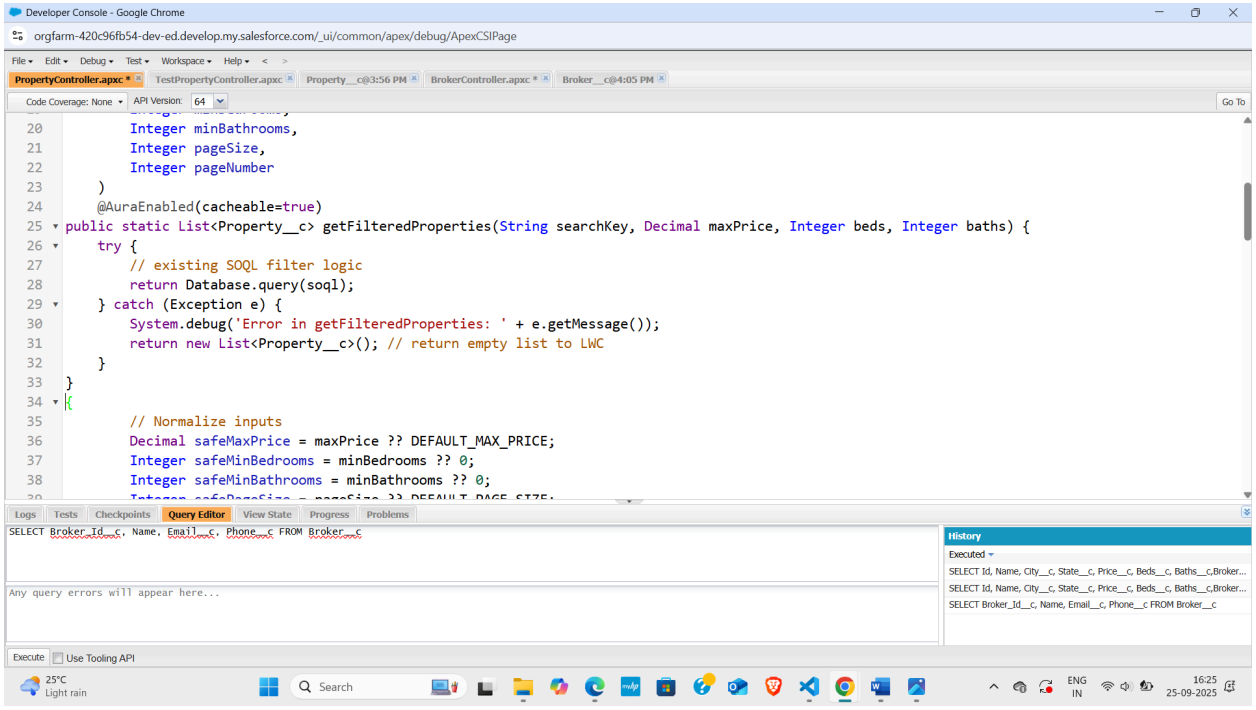


Fetching broker details:



Exception Handling

- **Why used:**
    - To ensure Apex methods don't break when invalid filters or missing data are passed from LWC.
    - Helps log errors and show user-friendly messages.
- **Implementation in PropertyController:**



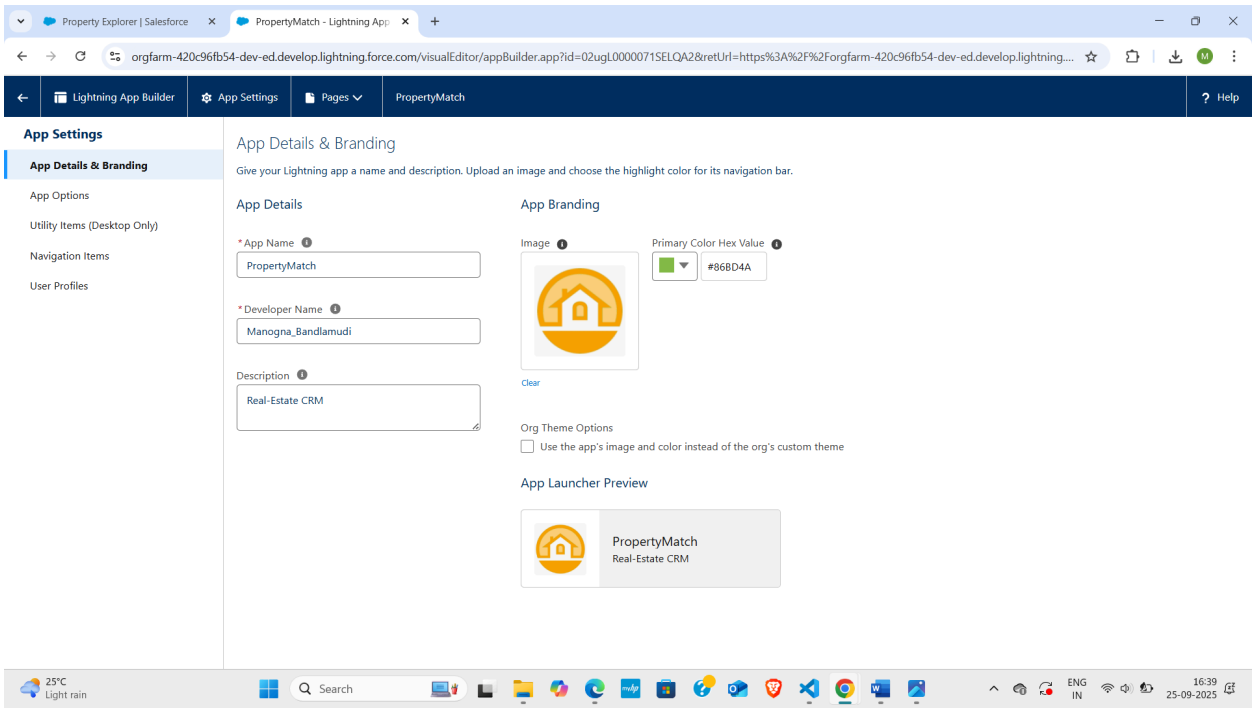
- **How it works:**
    - If invalid input/query error occurs, the system does not break.
    - Returns a safe empty list instead of crashing the LWC.

Phase 6: User Interface Development

1. Lightning App Setup

- Navigation: **Setup → App Manager → Lightning App Builder**.
- Created **PropertyMatch App** with tabs:
  - **Home**
  - **Properties**
  - **Brokers**

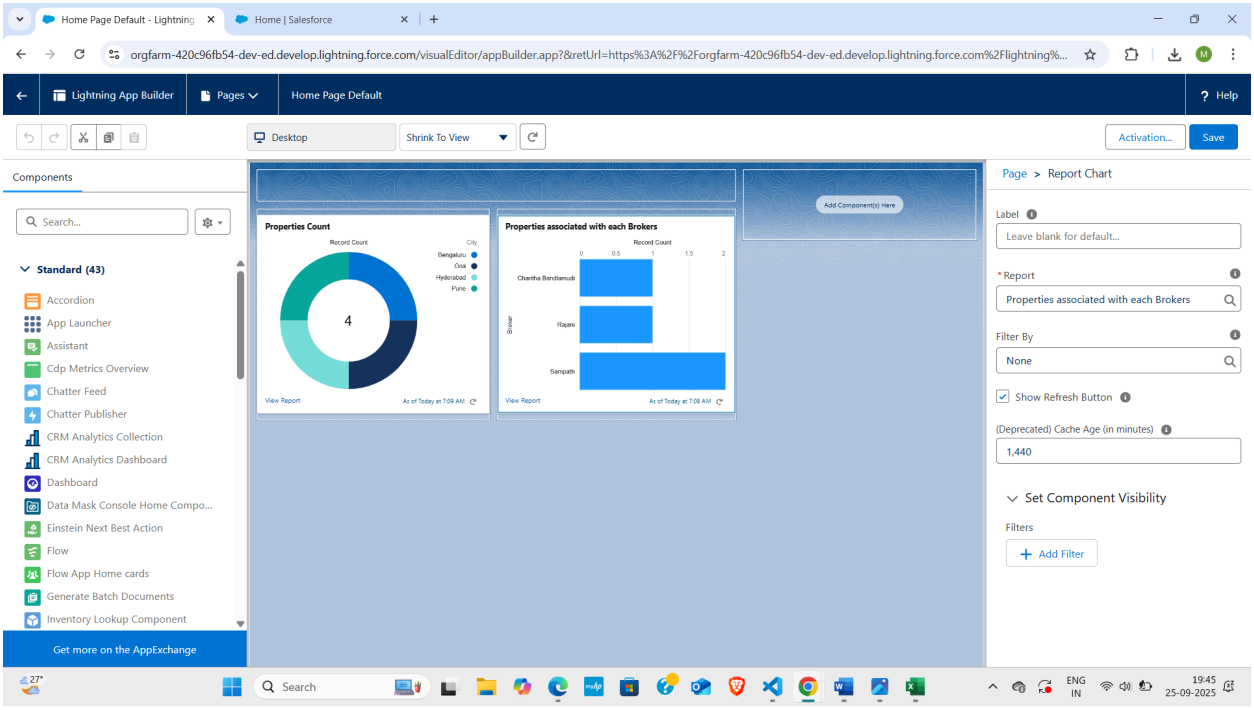
Why: Centralized navigation for all users (admins, brokers, buyers).



2. Home Page (Reports + Dashboard + Navigation)

- **Component Used:** Standard Dashboard + Report Chart.

- **How it works:** Displays property count by city in a donut chart (Bengaluru, Goa, Hyderabad, Pune).
- **Steps:**
    1. Go to **App Builder** → **Home Page**.
    2. Added **Report Chart Component** linked to “Properties Report”.
    3. Saved → Activated as org-wide default Home Page.
- **Screenshot:** Properties Report showing city-wise distribution.



3. Property Explorer Page (Custom LWC)

- **LWCs Used:** `propertyFilter`, `propertyTileList`, `propertyMap`.
- **How it works:**
    1. `propertyFilter` → captures inputs (search key, max price, bedrooms, bathrooms).

2. Sends filters to **Apex (PropertyController)**.

3. `propertyTitleList` → displays property cards with price, city, beds, baths, and image.

4. Clicking card → shows details and location.

5. `propertyMap` → displays property markers on Google Maps.

4. LWC Components & How They Work

a) propertyFilter LWC

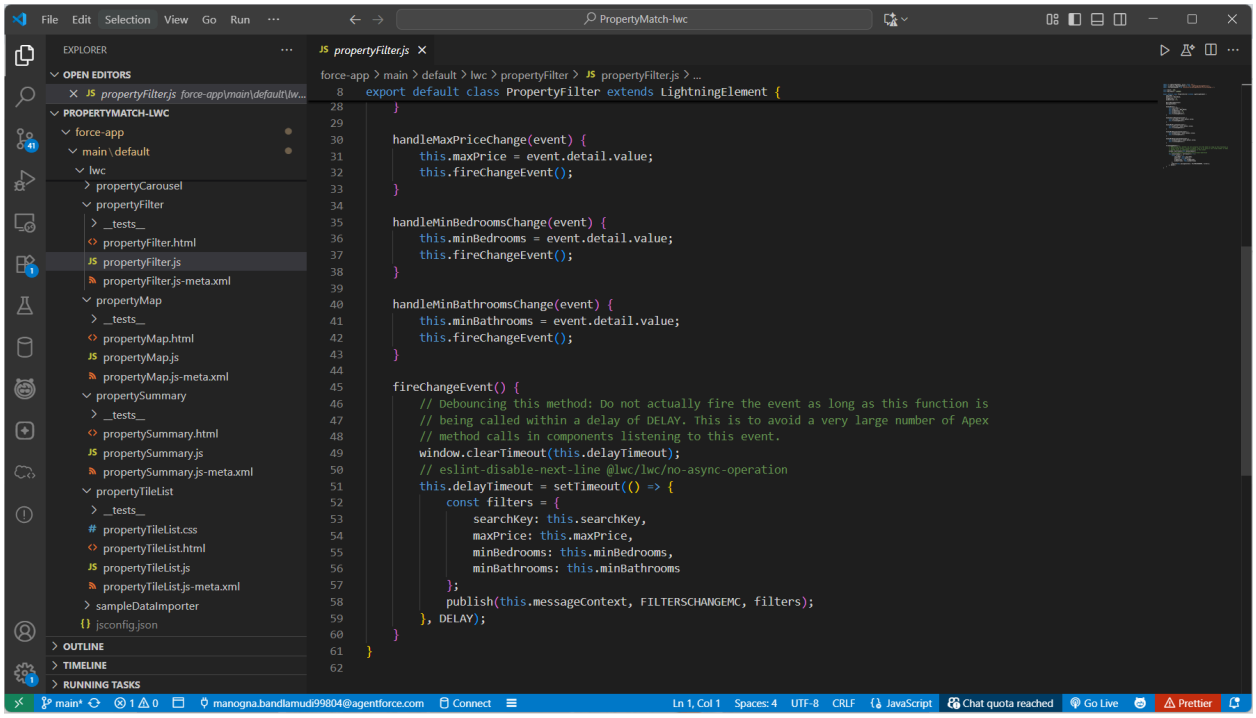
- **Purpose:** Capture user filter input (search key, max price, bedrooms, bathrooms).

- **How it works:**

    - HTML: Input fields (text, number).

    - JS: Stores user inputs and calls Apex method `getFilteredProperties()`.

    - On submit, it sends filters to PropertyList LWC.
    - Deploy your changes to your org by by right-clicking the file and selecting `SFDX: Deploy This Source to Org`



b) propertyList LWC

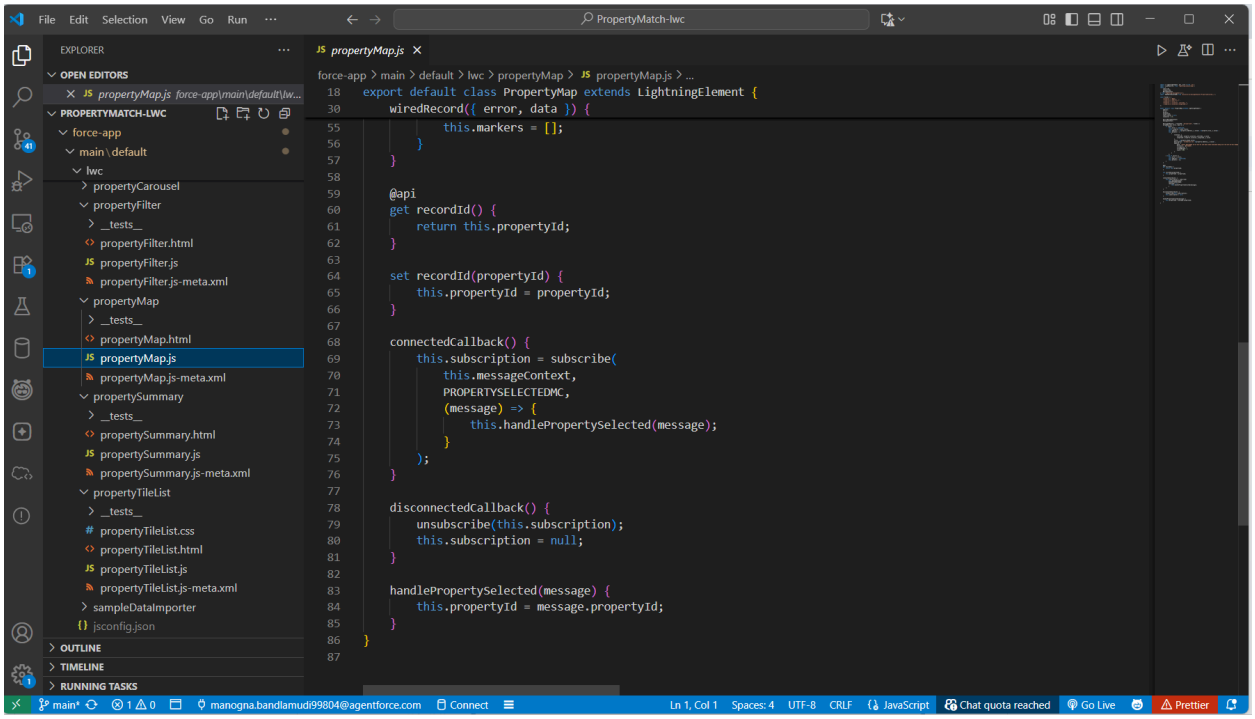
- **Purpose:** Display properties that match filters.
- **How it works:**
  - Calls `PropertyController.getFilteredProperties()` with filter values.
  - Displays property tiles (name, image, city, price, bedrooms, bathrooms).

c) propertyTile LWC

- **Purpose:** Individual card for each property.
- **How it works:**
  - Receives property record from propertyList.
  - Displays image, price, city, bedrooms, bathrooms.
  - On click → navigate to Property Record Page.

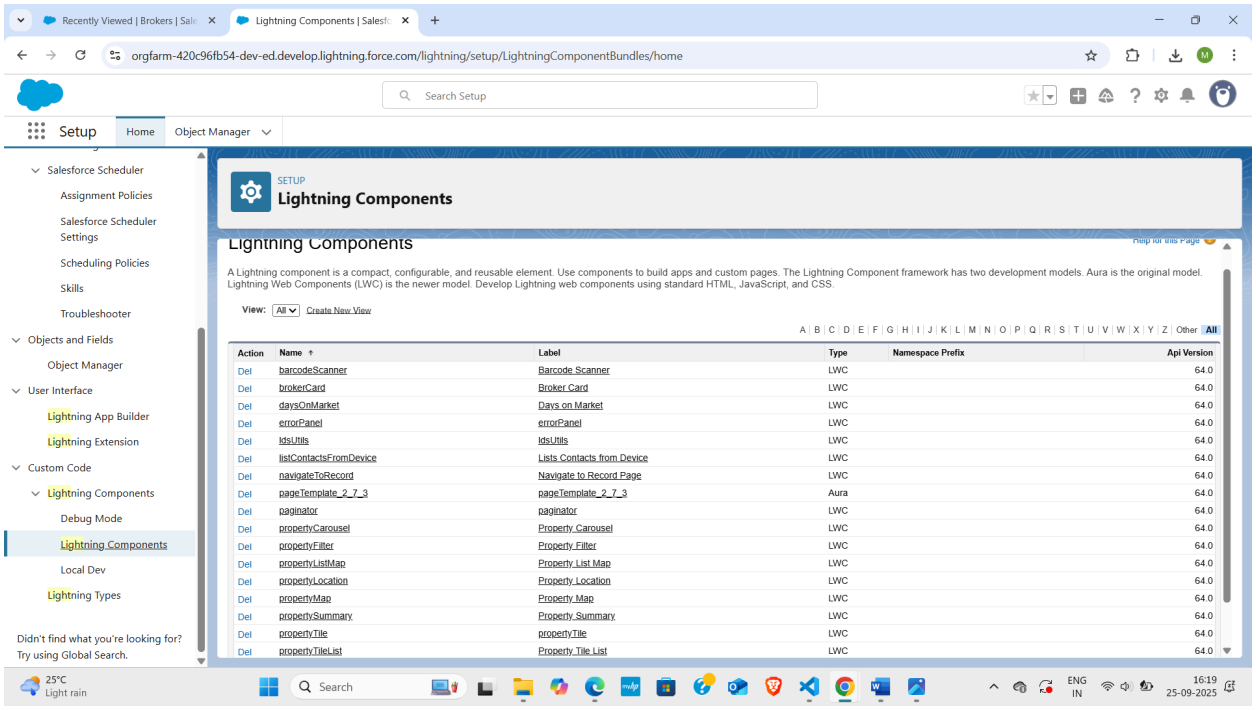
d) propertyMap LWC

- **Purpose:** Show property location on map.
- **How it works:**
    - Uses a lightning-map base component.
    - Takes Latitude__c and Longitude__c from Property__c.
    - Requires **Google OAuth 2.0 key** for map integration.



e) brokerCard LWC

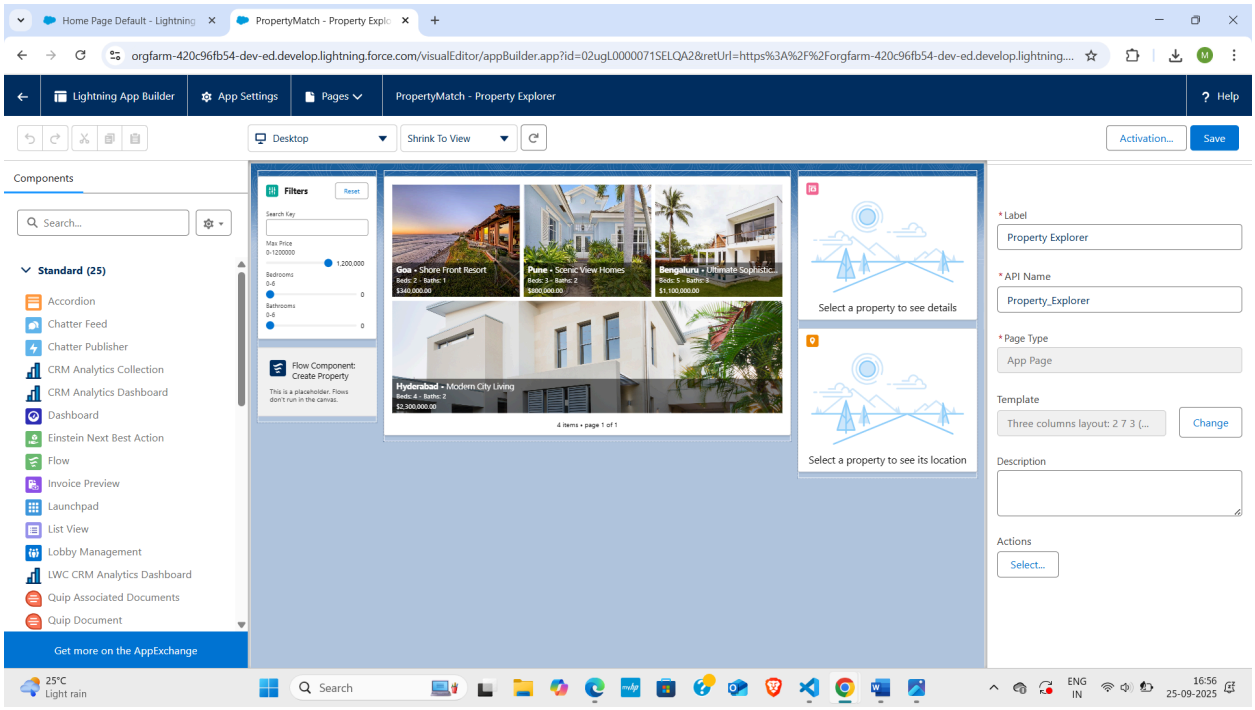
- **Purpose:** Show broker details when property record is opened.
- **How it works:**
    - Calls `BrokerController.getBrokerDetails()` using property's Broker__c.
    - Displays broker name, email, phone.



- **Steps:**

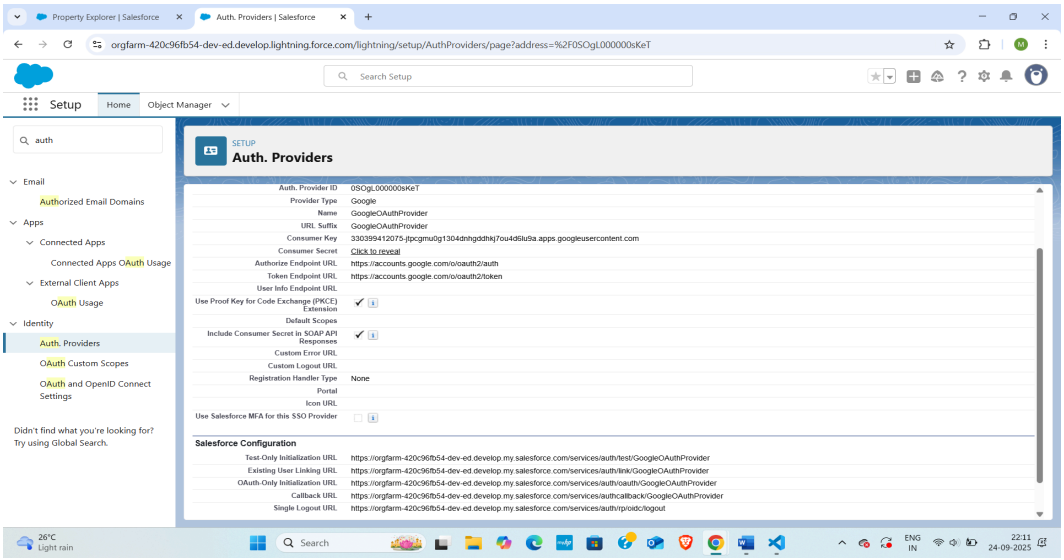
1. Created custom **Property Explorer Lightning Page**.
2. Dropped LWCs into layout (filters on left, list center, map/details on right).
3. Added **Create Property Form** section for admins.

- **Screenshot:** Property Explorer showing filters, property list, and create property tab.



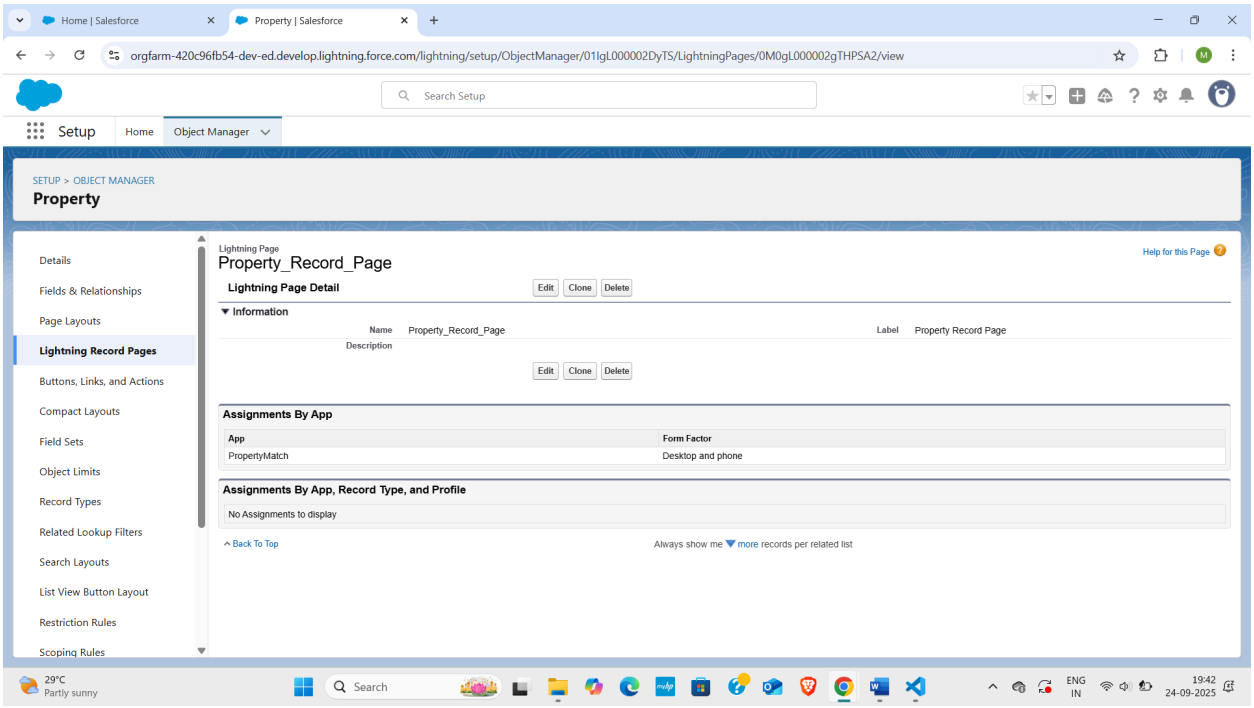
3. How Filters & Location Were Implemented

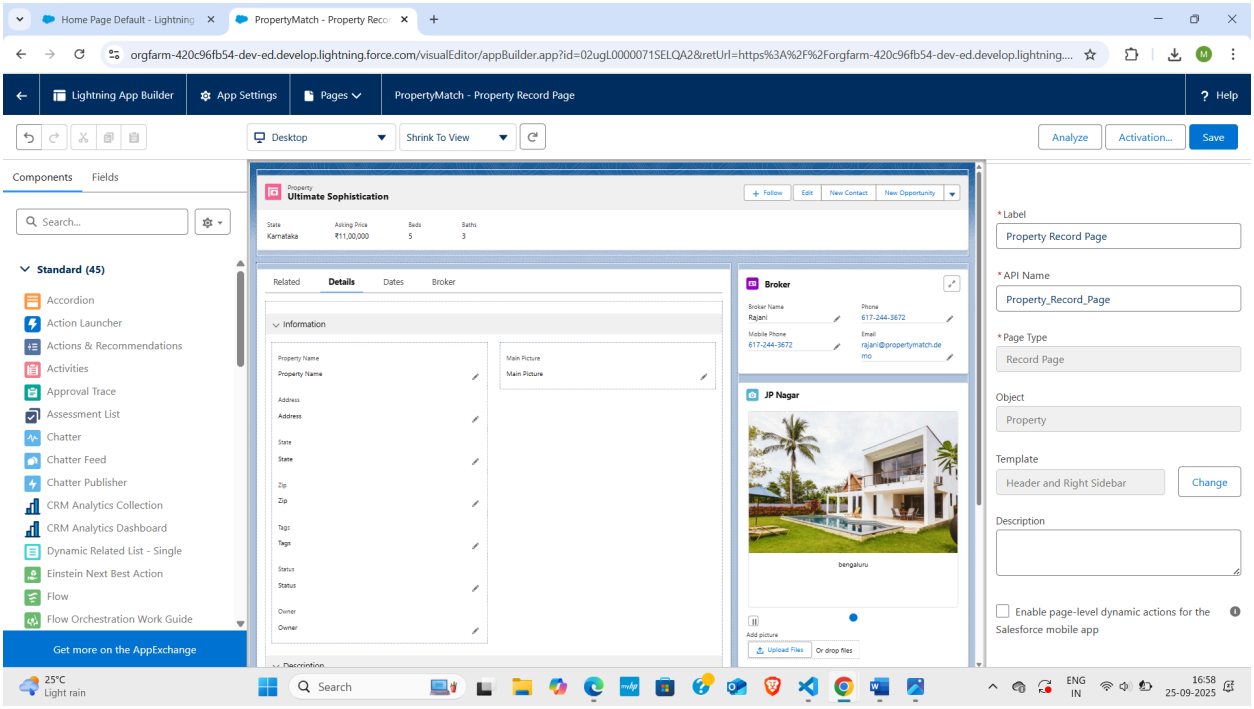
- **Filters (Search, Max Price, Bedrooms, Bathrooms):**
  - Implemented in **propertyFilter LWC**.
  - Passed to **PropertyController Apex** → SOQL query filters properties.
  - Results displayed in **propertyList LWC** with tiles.
- **Location (Map):**
  - Implemented in **propertyMap LWC** using `lightning-map`.
  - Uses **Latitude__c & Longitude__c** fields from Property__c.
  - Integrated with **Google OAuth 2.0** for map rendering.



6. Property Record Page

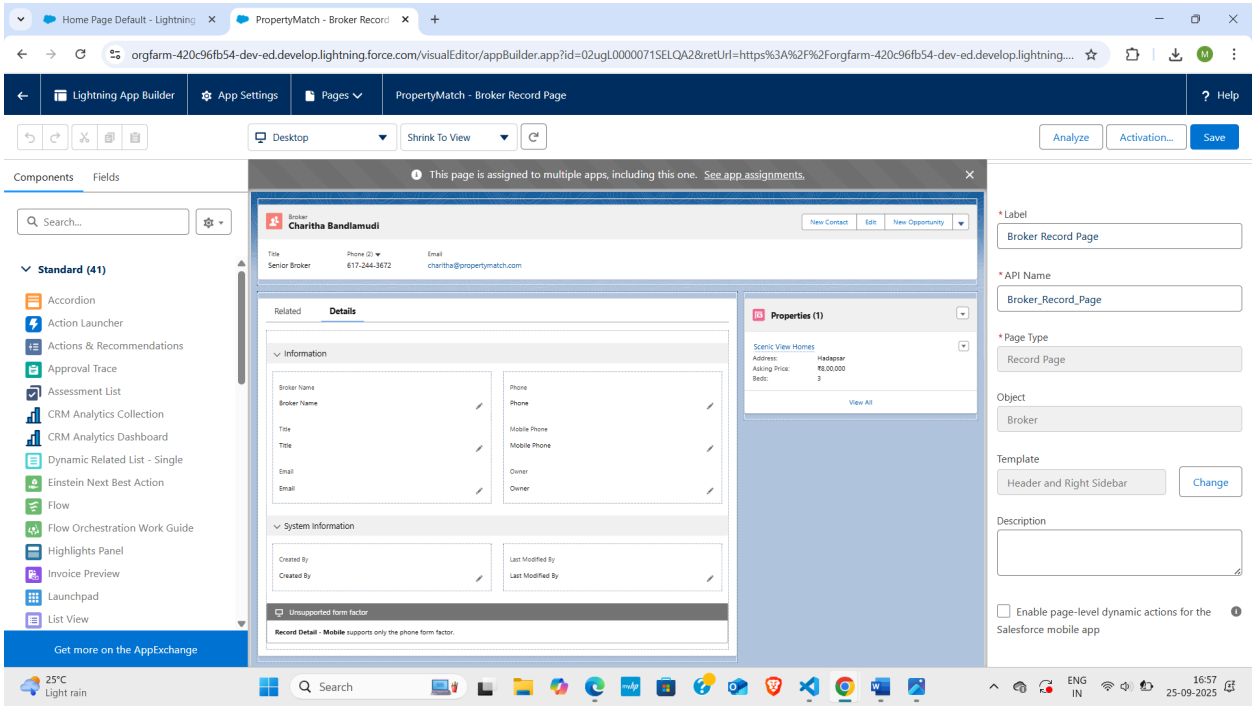
- **LWCs Used:** `propertyTile`, `brokerCard`
- **How it works:**
    1. Displays full property details (name, address, state, zip, status, description, image).
    2. Shows assigned broker info (`brokerCard`) with email and phone.
- **Steps:**
    1. Navigation: **Setup → Object Manager → Property__c → Lightning Record Pages → New Record Page**.
    2. Added LWC components into the property details section.
- **Screenshot:** Property record page with broker details and map.





7. Broker Record Page

- **Components Used:** `brokerCard`, Related List (Properties).
- **How it works:**
    1. Displays broker details (name, title, phone, email).
    2. Shows related properties assigned to the broker.
- **Steps:**
    1. Navigation: **Setup → Object Manager → Broker__c → Lightning Record Pages**.
    2. Added **brokerCard LWC** and **Properties Related List**.
- **Screenshot:** Broker record page with related property.



8. Tabs and Navigation

- **Tabs Created:**
    - **Home** → Default Home Page with reports.

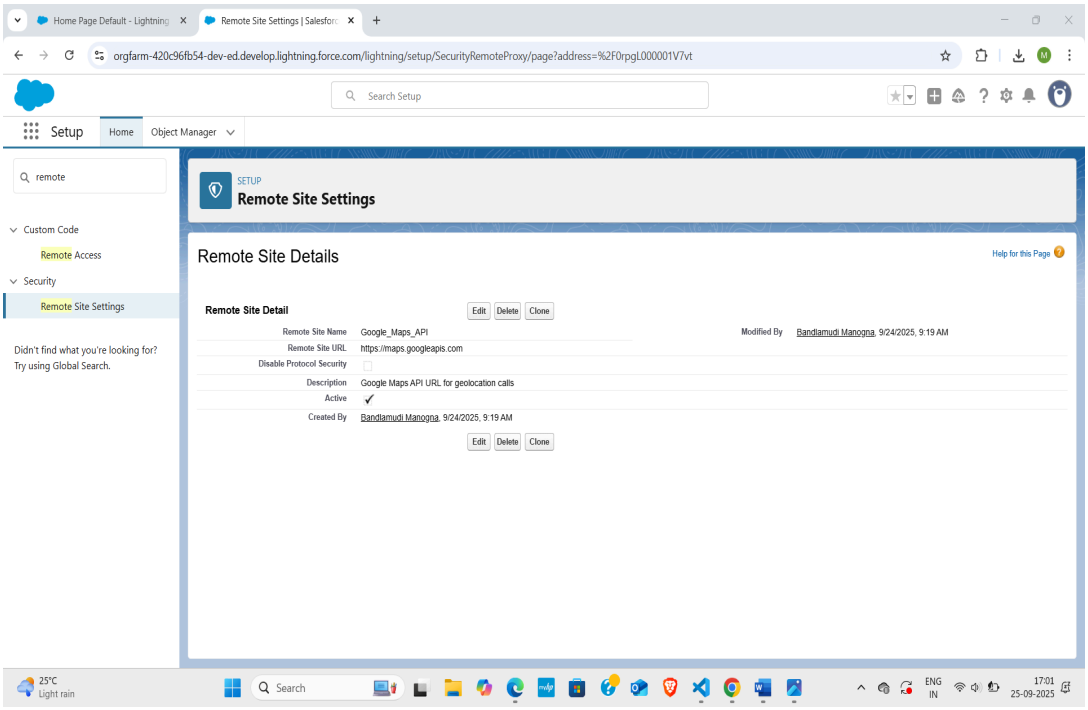
- **Property Explorer** → Custom navigation item with filters + search.
- **Properties** → Standard tab for Property__c records.
- **Brokers** → Standard tab for Broker__c records.

- **How it all connects:**
  - User lands on **Home Page** (reports).
  - Goes to **Property Explorer** → filters/search → clicks property.
  - Opens **Property Record Page** → sees broker info + map.
  - Navigates to **Broker Tab** → sees broker details + assigned properties.

Phase 7: Integration & External Access

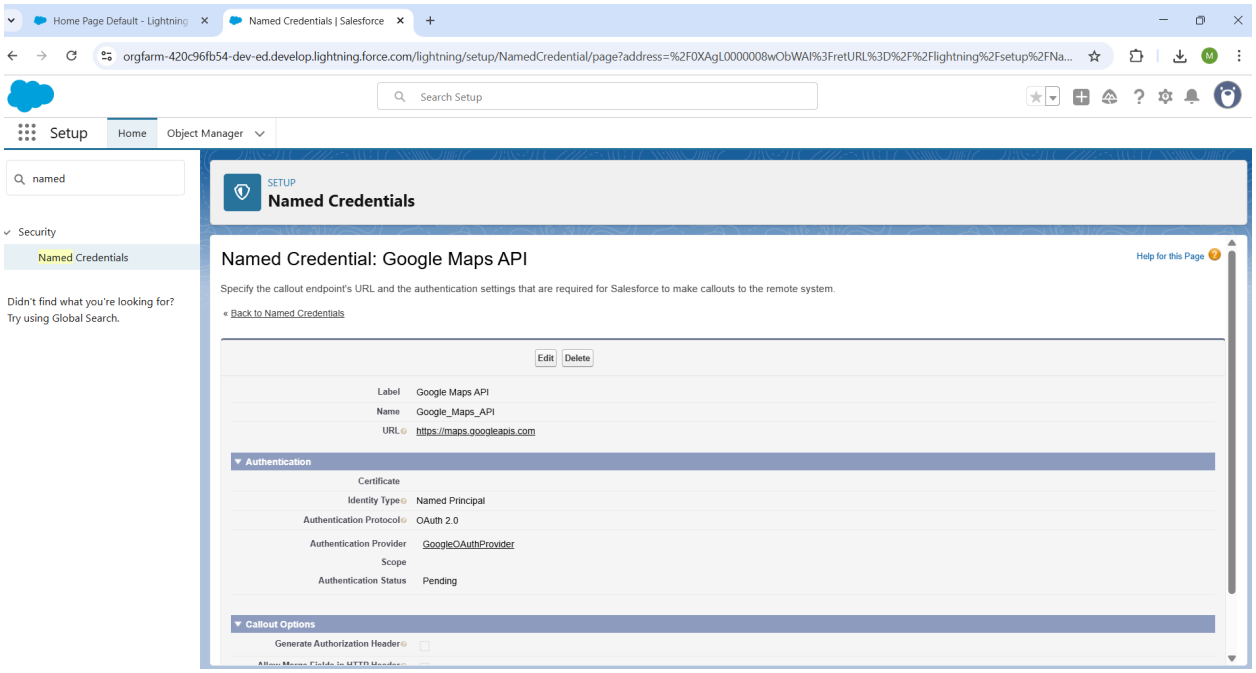
1. Remote Site Settings

- **Use Case:**
  - Needed to allow Salesforce org to make external callouts to Google Maps API.
- **Implementation Steps:**
  - Navigation: **Setup** → **Remote Site Settings** → **New Remote Site**.
  - Added Google Maps API base URL.
  - Enabled active status.
- **How it works:**
  - Allows Apex/LWC to send requests to Google Maps securely.



2. Named Credentials

- **Use Case:**
    - To securely store and authenticate credentials (API key or OAuth token) for Google API integration.
- **Implementation Steps:**
    - Navigation: **Setup → Named Credentials → New Named Credential**.
    - Added Google OAuth 2.0 endpoint with client ID, secret, and token.
    - Used in Apex for map integration.
- **How it works:**
    - Keeps credentials safe, avoids hardcoding in Apex.



3. OAuth & Authentication

- **Use Case:**
    - Required for **Google Maps OAuth 2.0 integration** to fetch map data and render property locations.
- **Implementation Steps:**
    - Registered project in **Google Cloud Console**.
    - Enabled Maps JavaScript API + Places API.
    - Generated **OAuth 2.0 Client ID & Secret**.
    - Configured in Salesforce under **Auth Providers + Named Credential**.
- **How it works:**

- When a user views a property map, Salesforce makes authenticated requests to Google API.

4. Web Services (REST Callouts)

- **Use Case:**
  - To fetch and display property locations on Google Map within Salesforce using coordinates.
- **Implementation Steps:**
  - Configured callout to Google Maps API endpoint.
  - Used `lightning-map` LWC component which consumes the coordinates.
- **How it works:**
  - Apex or LWC sends REST requests → Google API responds with map data → displayed in components.

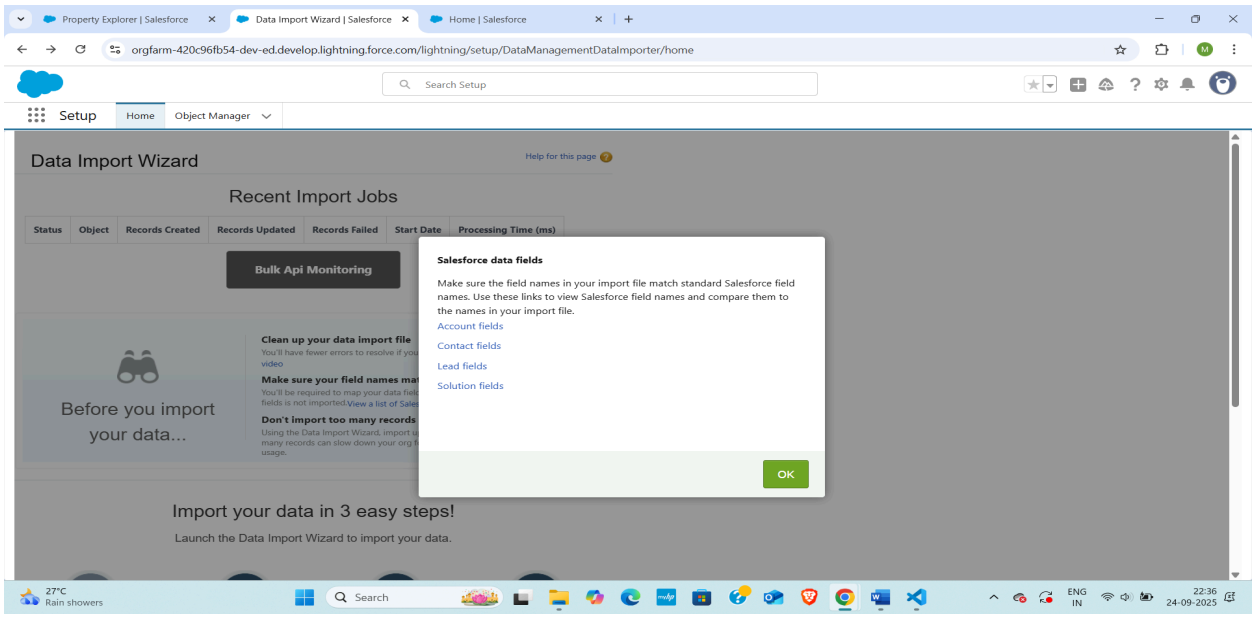
Phase 8: Data Management & Deployment

1. Data Import Wizard

- **Use Case:**
  - Used to upload initial property and broker records into the org.
  - Suitable for small data sets like test property listings.
- **Implementation Steps:**
  - Navigation: **App Launcher** → **Data Import Wizard**.
  - Selected **Property__c** object.

- Uploaded CSV with property details (City, State, Price, Bedrooms, Bathrooms, Latitude, Longitude, Broker).
- Mapped fields and imported records.

- **How it works:**
  - Created sample data used for testing filters and map display.

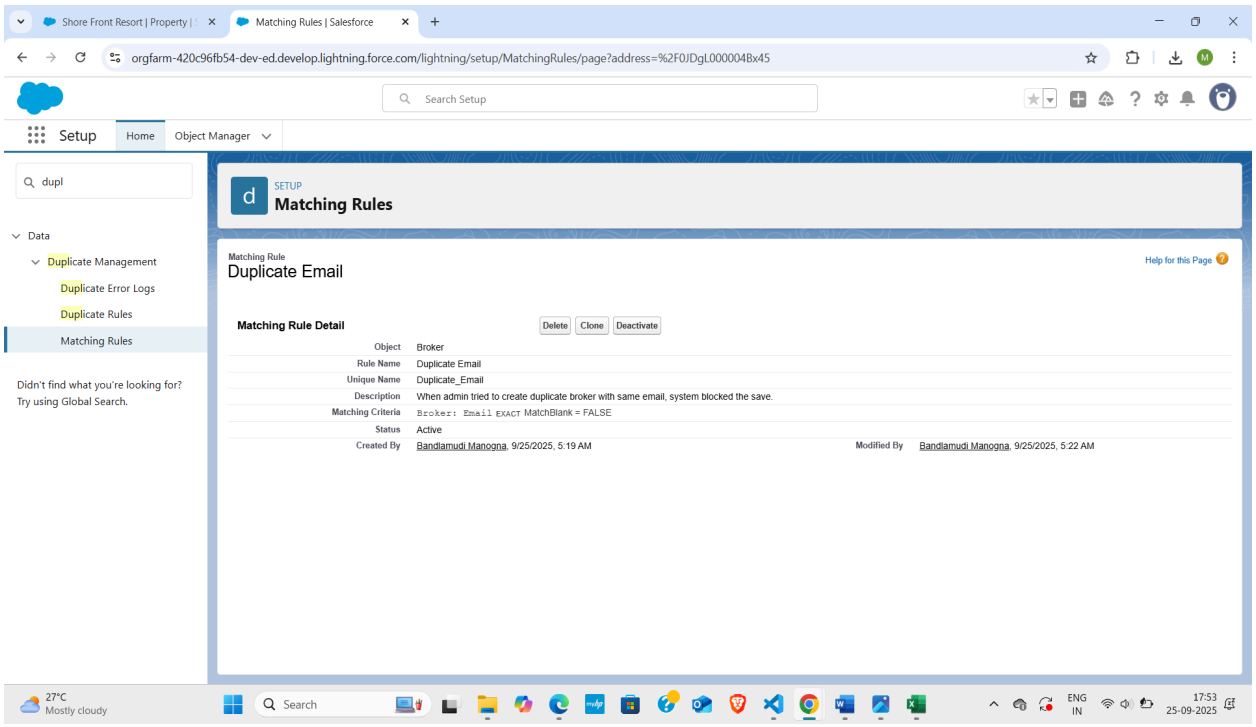


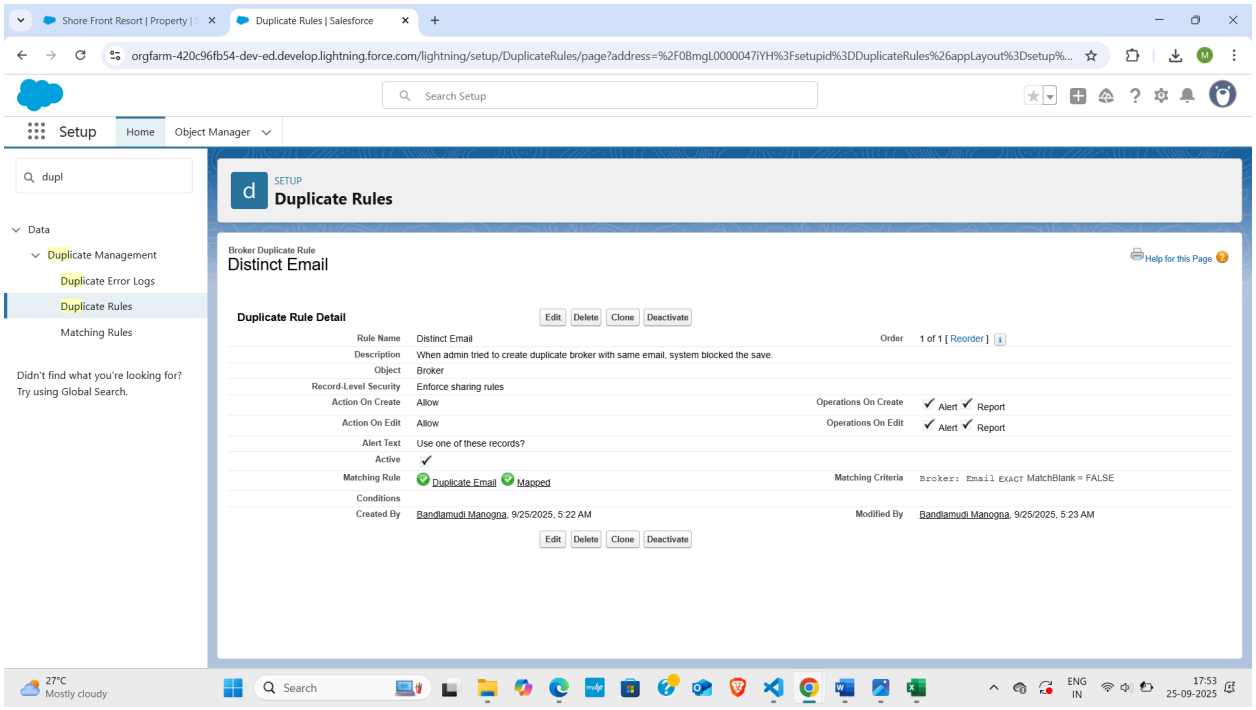
2. Duplicate Rules

- **Use Case:**
  - Prevented duplicate broker records with the same email address.

- **Implementation Steps:**
  - Navigation: **Setup** → **Duplicate Rules** → **New Rule**.
  - Selected **Broker__c** object.
  - Matching rule: Email__c must be unique.

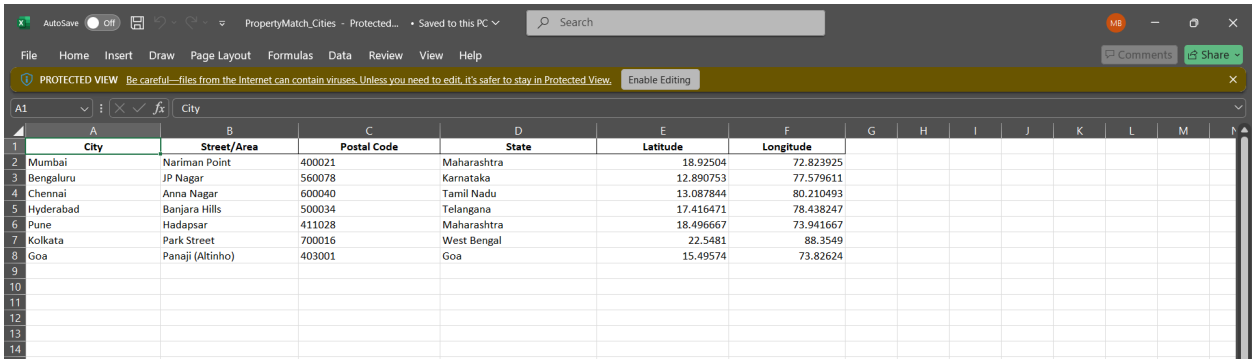
- **How it works:**
    - When the admin tried to create a duplicate broker with the same email, the system blocked the save.





3. Data Export & Backup

- **Use Case:**
    - For backup of property and broker records.
- **Implementation Steps:**
    - Navigation: **Setup** → **Data Export** → **Schedule Export**.
    - Selected **Property__c** and **Broker__c** objects.
    - Scheduled weekly exports.
- **How it works:**
    - Backup files stored as CSVs for safety in case of data loss.



	A	B	C	D	E	F
1	City	Street/Area	Postal Code	State	Latitude	Longitude
2	Mumbai	Nariman Point	400021	Maharashtra	18.92504	72.823925
3	Bengaluru	JP Nagar	560078	Karnataka	12.890753	77.579611
4	Chennai	Anna Nagar	600040	Tamil Nadu	13.087844	80.210493
5	Hyderabad	Banjara Hills	500034	Telangana	17.416471	78.438247
6	Pune	Hadapsar	411028	Maharashtra	18.496667	73.941667
7	Kolkata	Park Street	700016	West Bengal	22.5481	88.3549
8	Goa	Panaji (Altinho)	403001	Goa	15.49574	73.82624

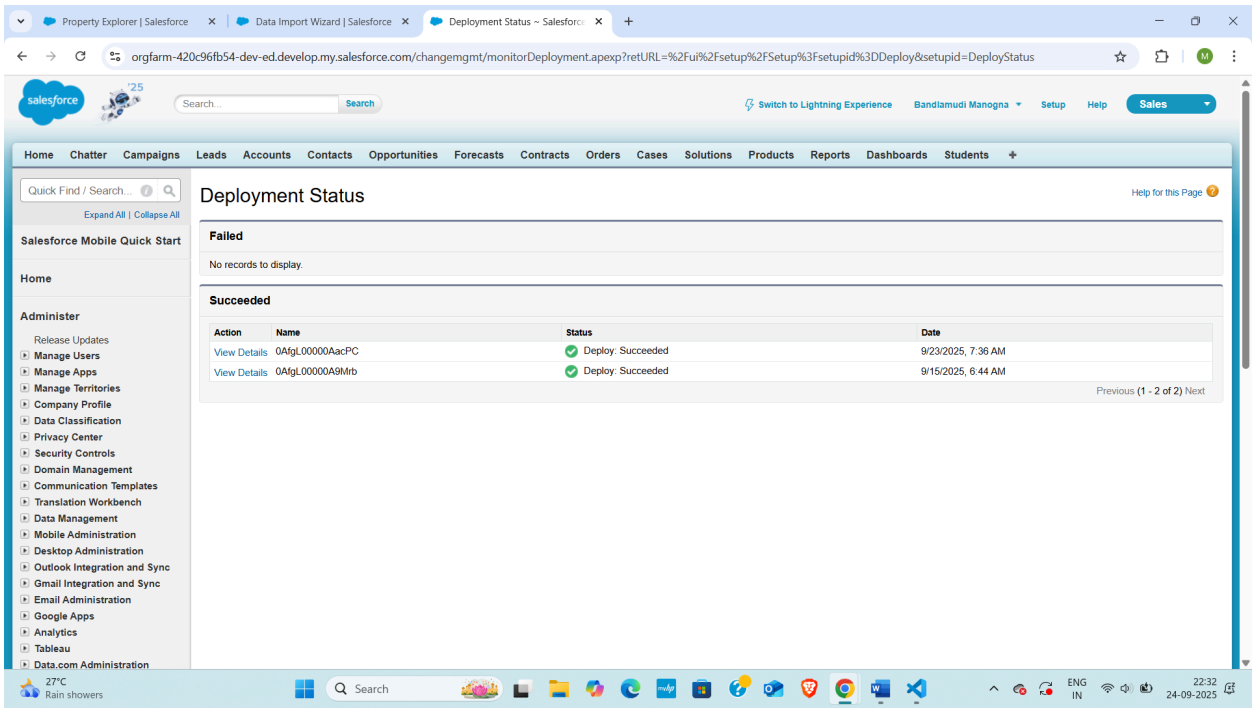
4. VS Code & SFDX

- **Use Case:**
    - Used for building LWCs, Apex classes, and deploying metadata.
- **Implementation Steps:**
    - Open project in **VS Code**.
    - Developed LWCs (propertyFilter, propertyList, propertyTile, propertyMap, brokerCard).

Deployed to org with: `sfdx force:source:push`

Retrieved metadata from org with: `sfdx force:source:pull`

- **How it works:**
    - Enabled smooth development → test → deployment cycle.



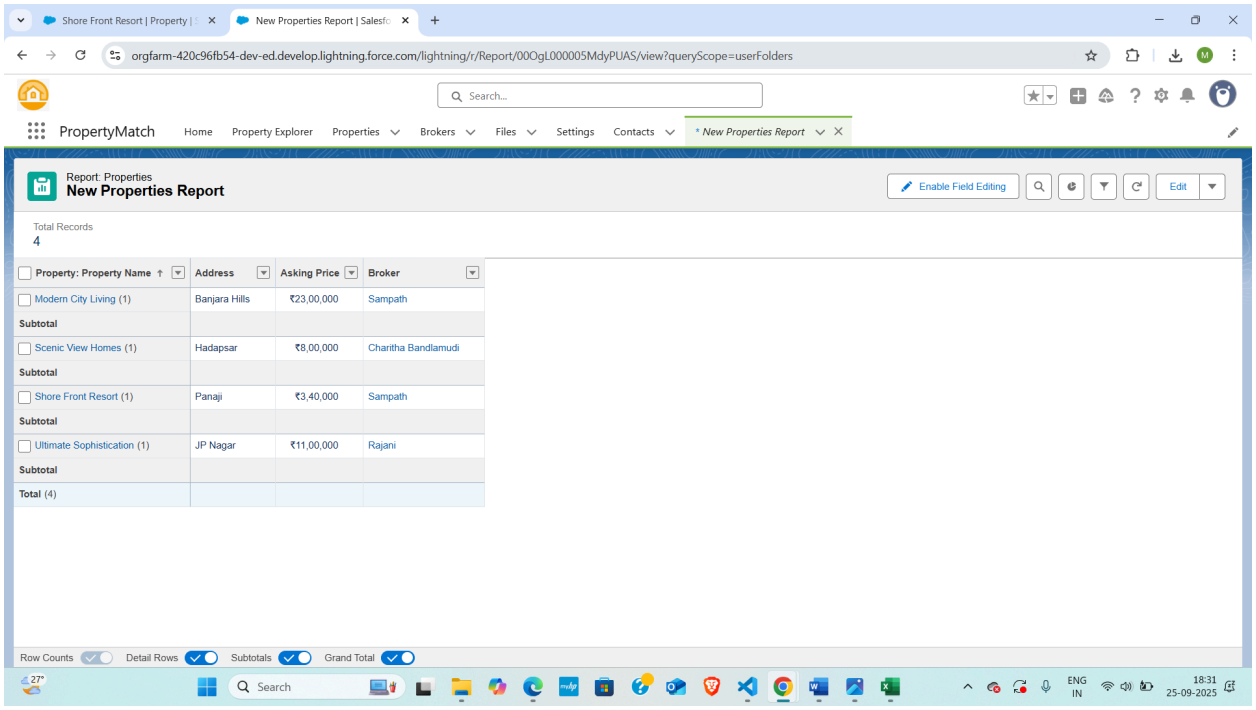
Phase 9: Reporting, Dashboards & Security Review

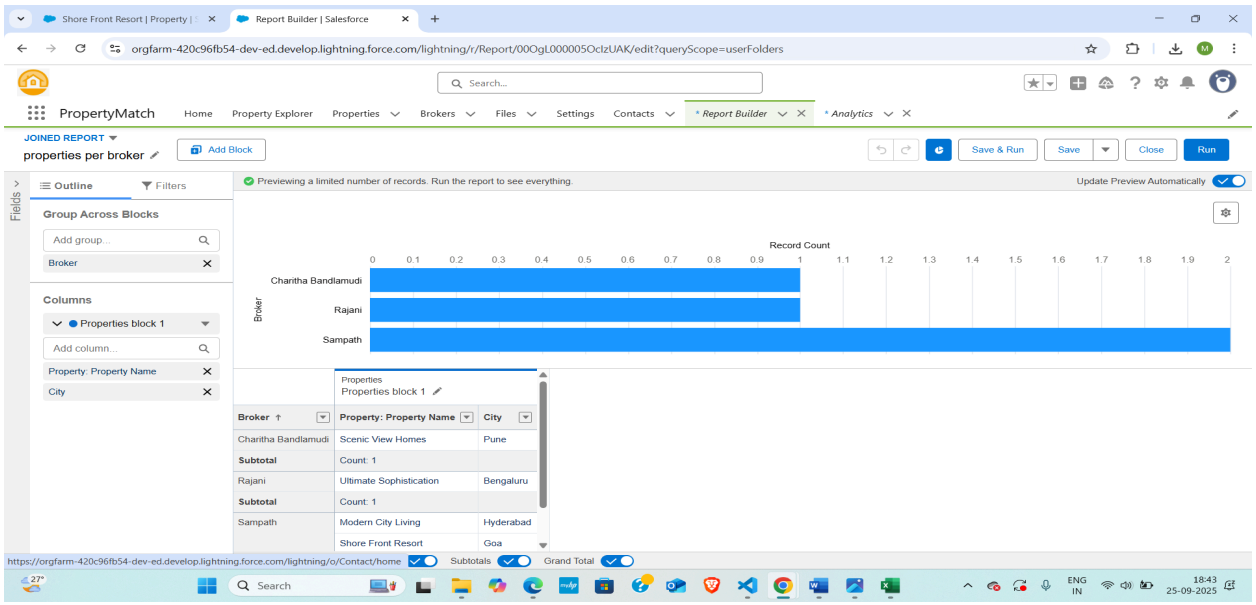
1. Reports

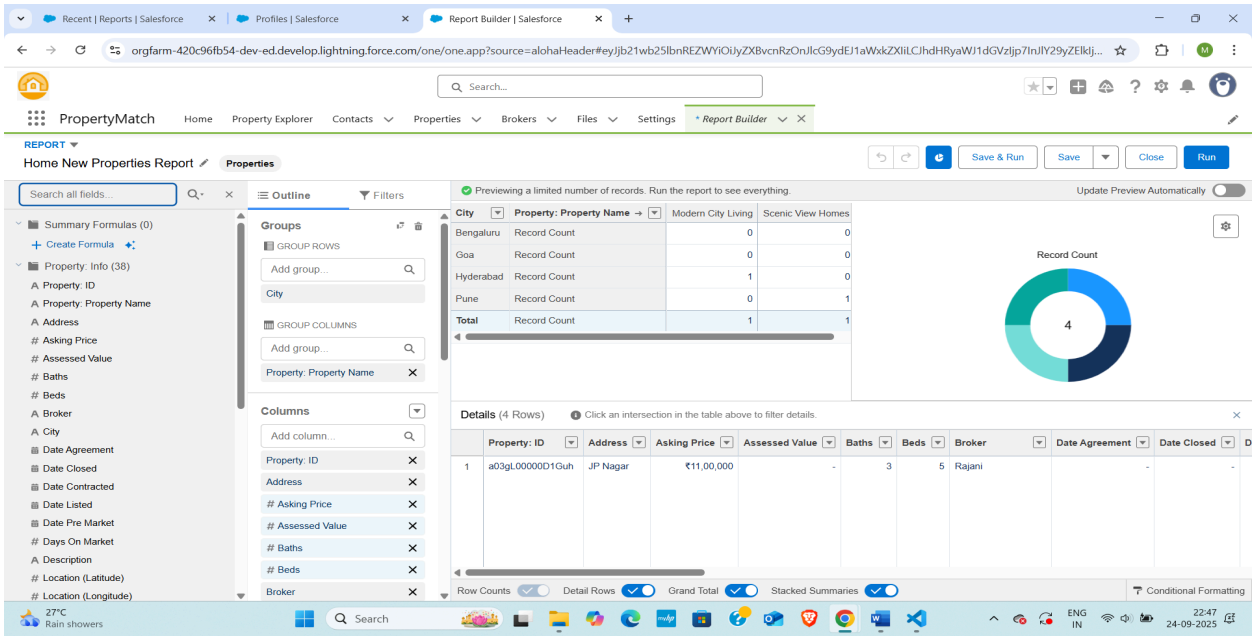
- **Use Case:** To analyze property data (price ranges, city-wise distribution, broker assignments).
- **Types Used:**
    - **Tabular Report:** Simple list of all properties with city, price, bedrooms, bathrooms.

    - **Joined Report:** List of each broker name with the number of properties they are assigned to.

- **Implementation Steps:**
    - Navigation: **App Launcher** → **Reports** → **New Report**.
    - Choose a custom report type for **Properties with Brokers**.
- **How it works:** Displays insights into availability and broker workload.
- **Use Case:** To join **Property__c** and **Broker__c** objects in one report.
- **Implementation Steps:**
    - Navigation: **Setup** → **Report Types** → **New Custom Report Type**.
    - Primary Object: Property__c.
    - Related Object: Broker__c.
- **How it works:** Allows reports combining property details with broker assignments.

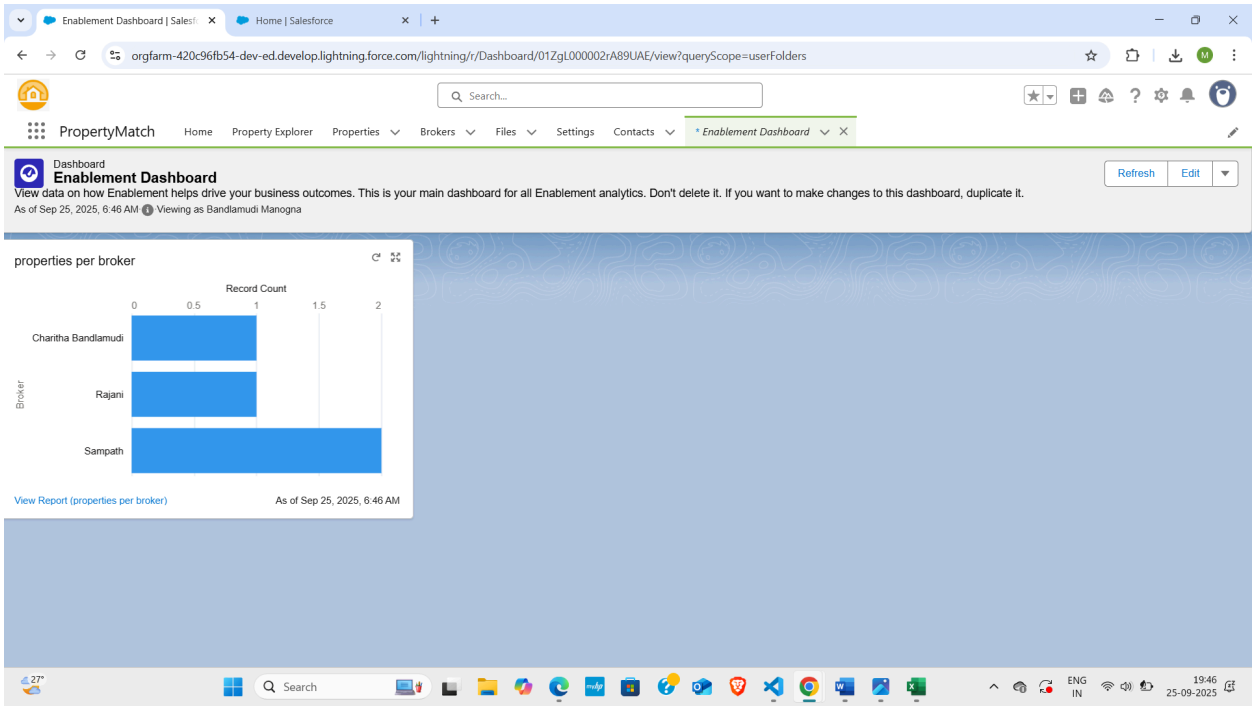

Shore Front Resort | Property
New Properties Report | Salesfo
orgfarm-420c96fb54-dev-ed.develop.lightning.force.com/lightning/r/Report/00OgL000005MdyPUAS/view?queryScope=userFolders
Search...
PropertyMatch
Home
Property Explorer
Properties
Brokers
Files
Settings
Contacts
* New Properties Report
Report: Properties
New Properties Report
Enable Field Editing
Edit
Total Records
4
Property: Property Name
Address
Asking Price
Broker
Modern City Living (1)
Banjara Hills
₹23,00,000
Sampath
Subtotal
Scenic View Homes (1)
Hadapsar
₹8,00,000
Charitha Bandlamudi
Subtotal
Shore Front Resort (1)
Panaji
₹3,40,000
Sampath
Subtotal
Ultimate Sophistication (1)
JP Nagar
₹11,00,000
Rajani
Subtotal
Total (4)
Row Counts
Detail Rows
Subtotals
Grand Total
Search
ENG
IN
18:31
25-09-2025


Shore Front Resort | Property
Report Builder | Salesforce
orgfarm-420c96fb54-dev-ed.develop.lightning.force.com/lightning/r/Report/00OgL000005OclzUAK/edit?queryScope=userFolders
Search...
PropertyMatch
Home
Property Explorer
Properties
Brokers
Files
Settings
Contacts
* Report Builder
* Analytics
JOINED REPORT
properties per broker
Add Block
Save & Run
Save
Close
Run
Fields
Outline
Filters
Group Across Blocks
Add group...
Broker
Columns
Properties block 1
Add column...
Property: Property Name
City
Previewing a limited number of records. Run the report to see everything.
Update Preview Automatically
Record Count
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
1.9
2
Broker
Charitha Bandlamudi
Rajani
Sampath
Properties
Properties block 1
Broker
Property: Property Name
City
Charitha Bandlamudi
Scenic View Homes
Pune
Subtotal
Count: 1
Rajani
Ultimate Sophistication
Bengaluru
Subtotal
Count: 1
Sampath
Modern City Living
Hyderabad
Shore Front Resort
Goa
https://orgfarm-420c96fb54-dev-ed.develop.lightning.force.com/lightning/o/Contact/home
Subtotals
Grand Total
Search
ENG
IN
18:43
25-09-2025



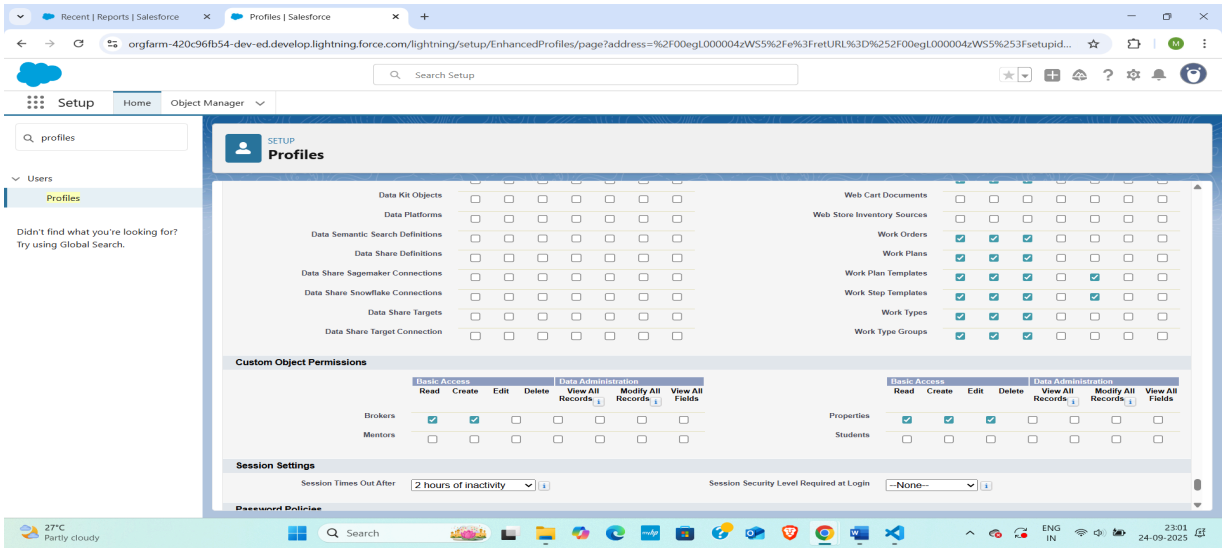
2. Dashboards

- **Use Case:** Visualize property and broker data for management.
- **Widgets Created:**
  - Pie Chart → Property distribution by city.
  - Bar Chart → No. of properties per broker.
- **Implementation Steps:**
  - Navigation: **App Launcher → Dashboards → New Dashboard**.
  - Added charts using saved reports.
- **How it works:** Quick, visual overview of CRM performance.



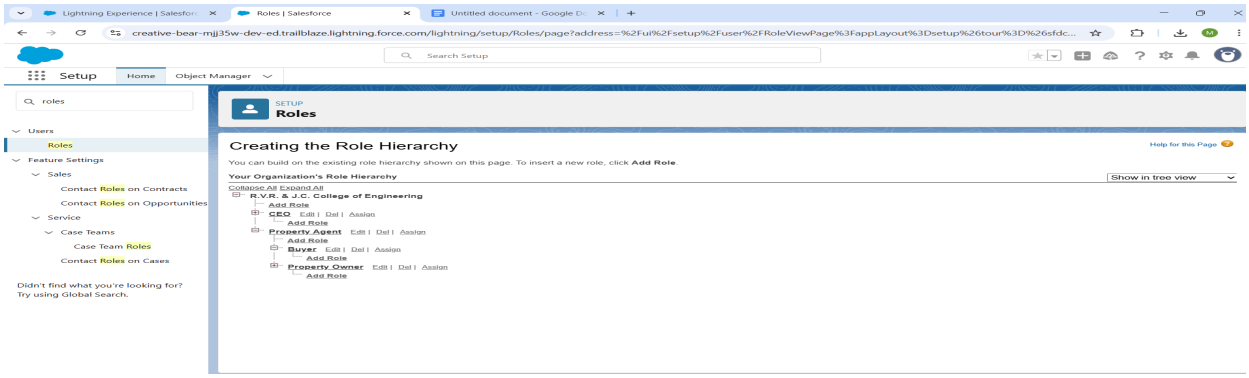
3. Profiles

- **Use Case:** Define access levels.
- **Profiles Created:**
    - **System Administrator** → Full access.
    - **Property Manager** → Manage properties and brokers.
    - **Broker User** → View only assigned properties.
- **How it works:** Restricts what each role can do in CRM.



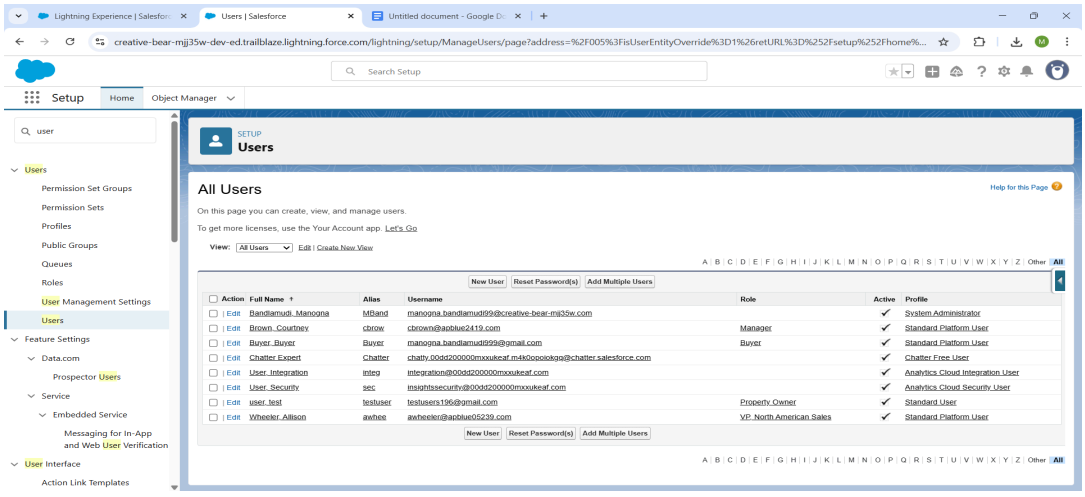
4. Roles & Role Hierarchy

- **Use Case:** Data visibility based on hierarchy.
- **Roles Created:**
  - **Admin** (top).
  - **Property Manager** (middle).
  - **Broker** (bottom).
- **Implementation Steps:**
  - Setup → Roles → New Role.
  - Assigned users to roles.
- **How it works:** Managers can see data owned by brokers below them.



5. Users

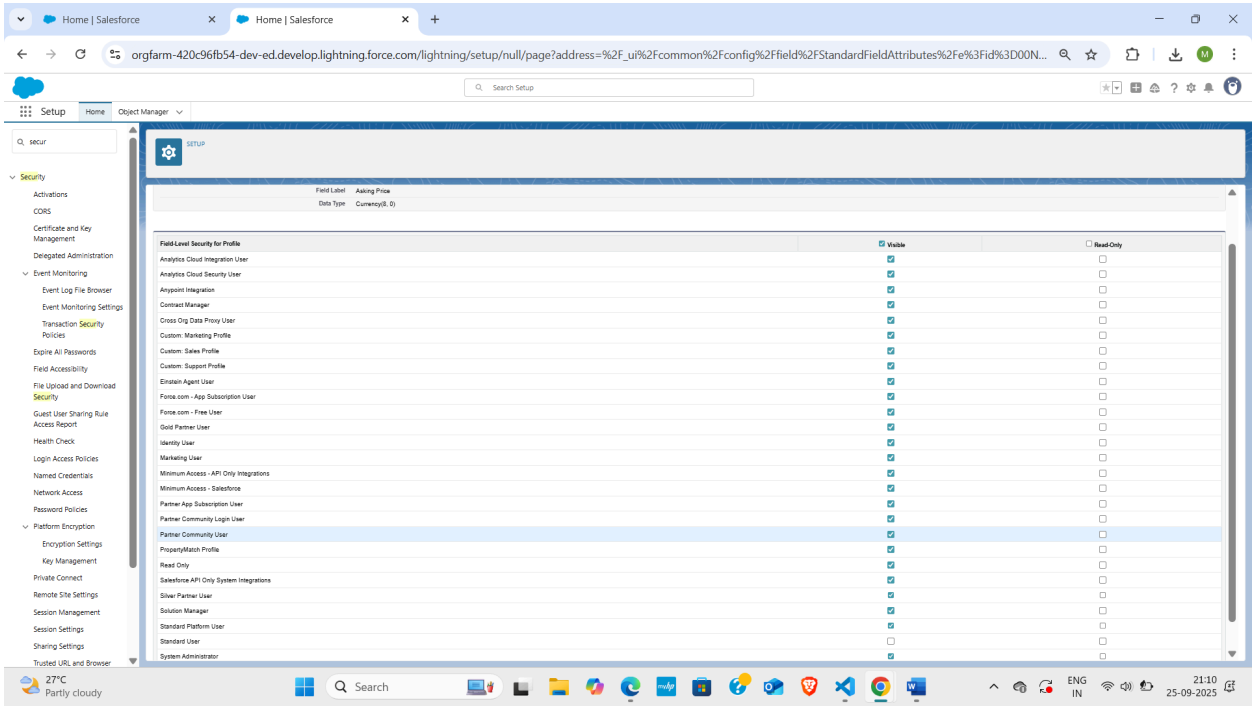
- **Use Case:** Create different users for testing security & sharing.
- **Implementation Steps:**
  - Setup → Users → New User.
  - Created: Admin, 1 Manager, 2 Broker users.
- **How it works:** Tested record visibility & email alerts.



6. Field-Level Security (FLS)

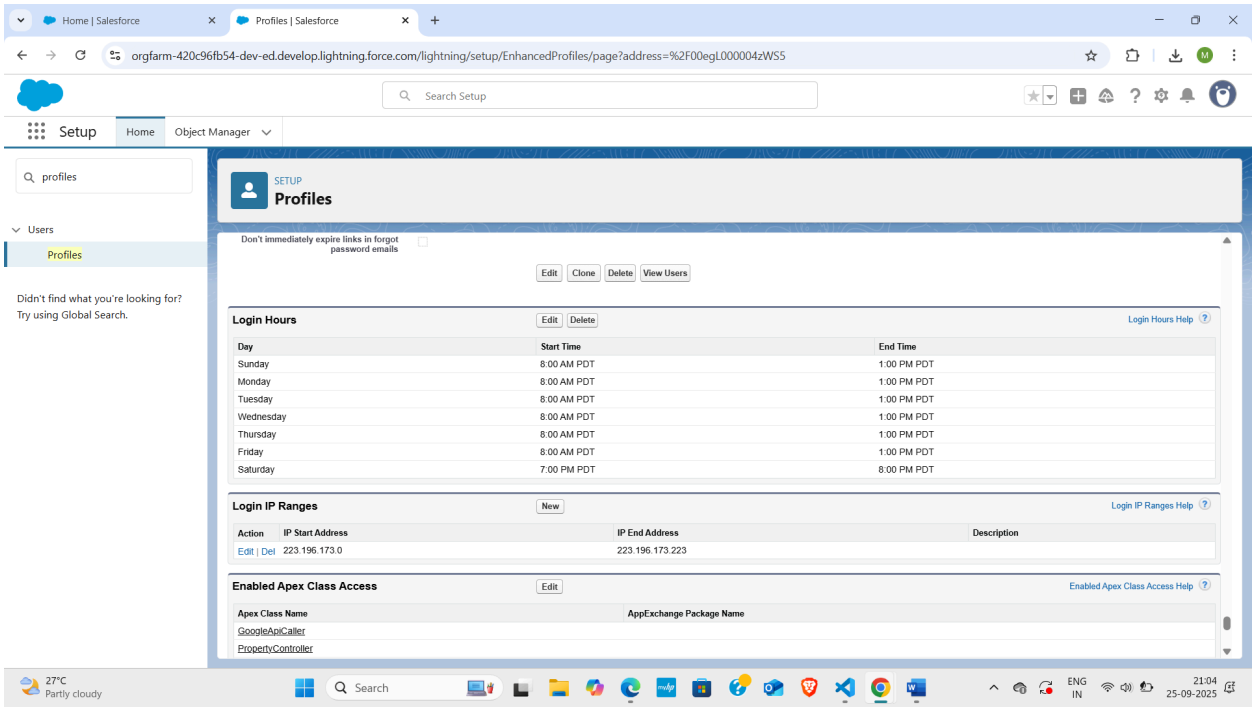
- **Use Case:** Restrict sensitive fields.

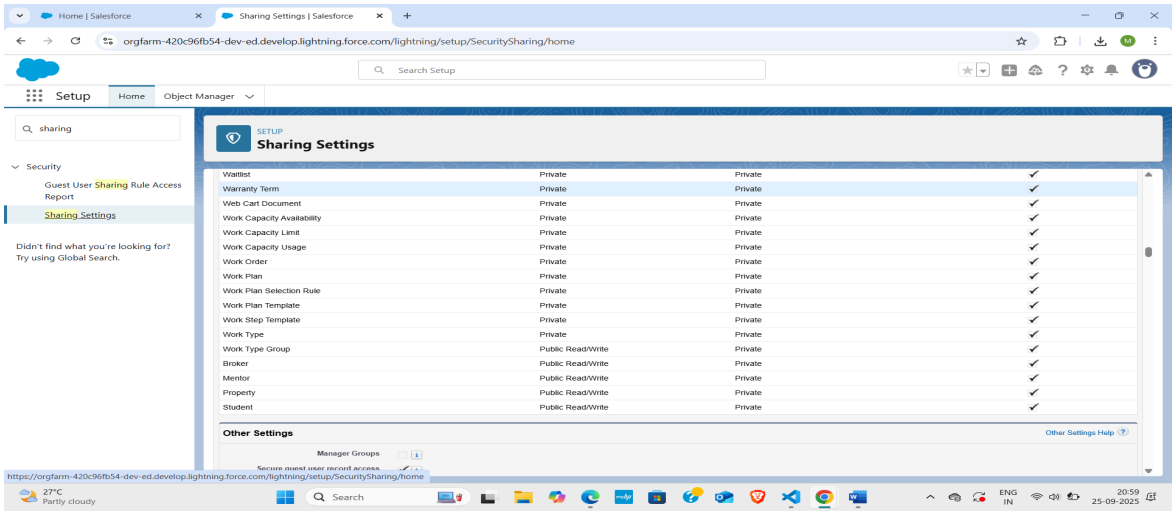
- **Example:** Hide `Price__c` from brokers, visible only to managers/admin.
- **Implementation Steps:**
    - Setup → Object Manager → Property__c → Fields & Relationships → Set Field-Level Security.
- **How it works:** Brokers can see property details but not price field, hide from standard users.



7. Session Settings & Login IP Ranges

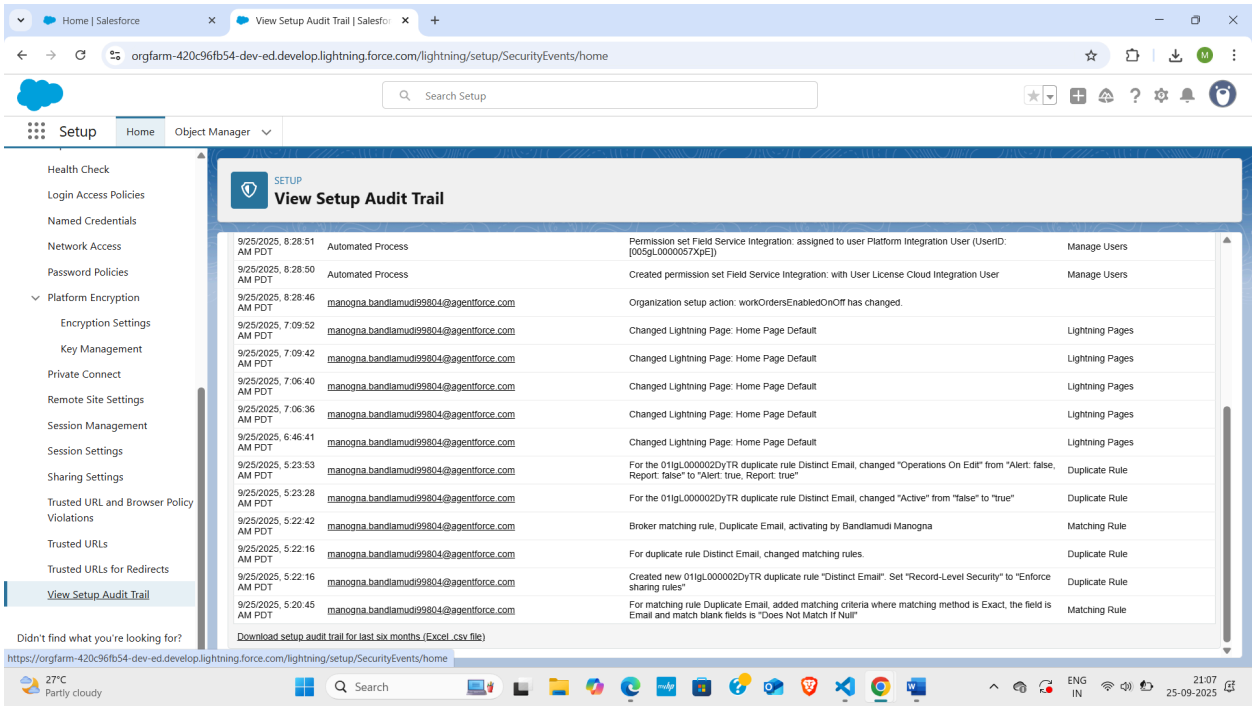
- **Use Case:** Extra security for admin access.
- **Implementation Steps:**
    - Setup → Session Settings → Reduced session timeout.
    - Setup → Profiles → Login IP Ranges → Allowed only office IP.
- **How it works:** Prevents unauthorized logins.





8. Audit Trail

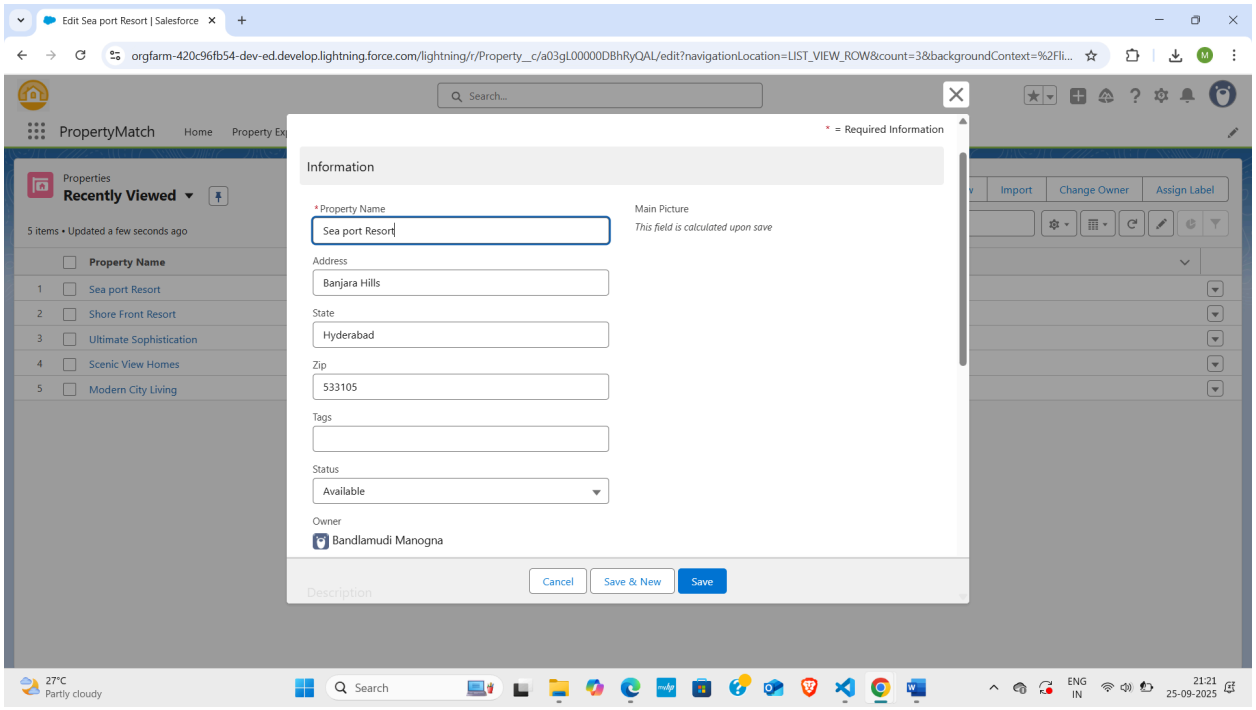
- **Use Case:** Track admin changes.
- **Implementation Steps:**
  - Setup → View Setup Audit Trail.
- **How it works:** Logged all setup/configuration changes for review.

- 

Phase 10: Quality Assurance Testing

Test Case 1: Property Record Creation

- **Use Case / Scenario:** Create a new property record with required details.
- **Test Steps (Input):**
  - Navigate to **Properties** → **New**.
  - Enter City = “Hyderabad”, State = “Telangana”, Price = 50,00,000, Bedrooms = 3, Bathrooms = 2.
  - Assign to Broker.
- **Expected Result:**
  - Property record should save successfully and appear in list view.
- **Actual Result:**
  - Property saved successfully, record visible under Properties tab.



Test Case 2: Email Notification to Broker on Property Assignment

Use Case / Scenario:

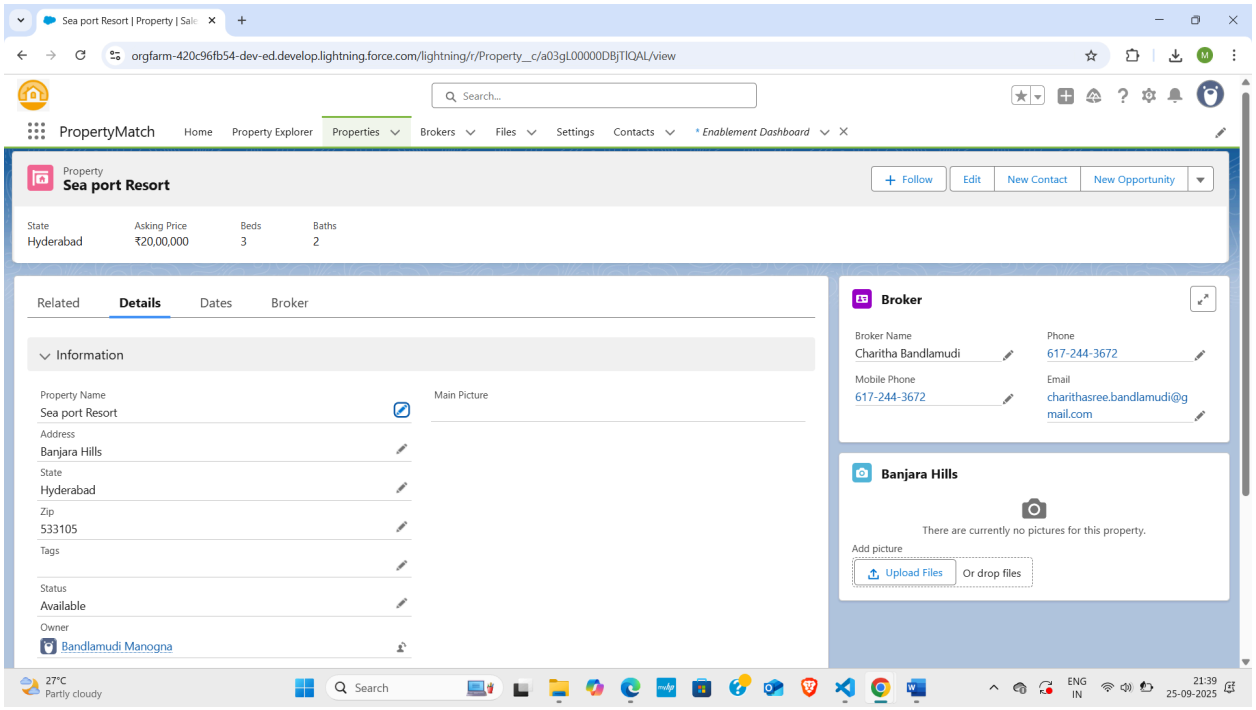
When a new property record is created and assigned to a broker, the system should automatically send an email notification to the broker with property details.

Test Steps (with Input):

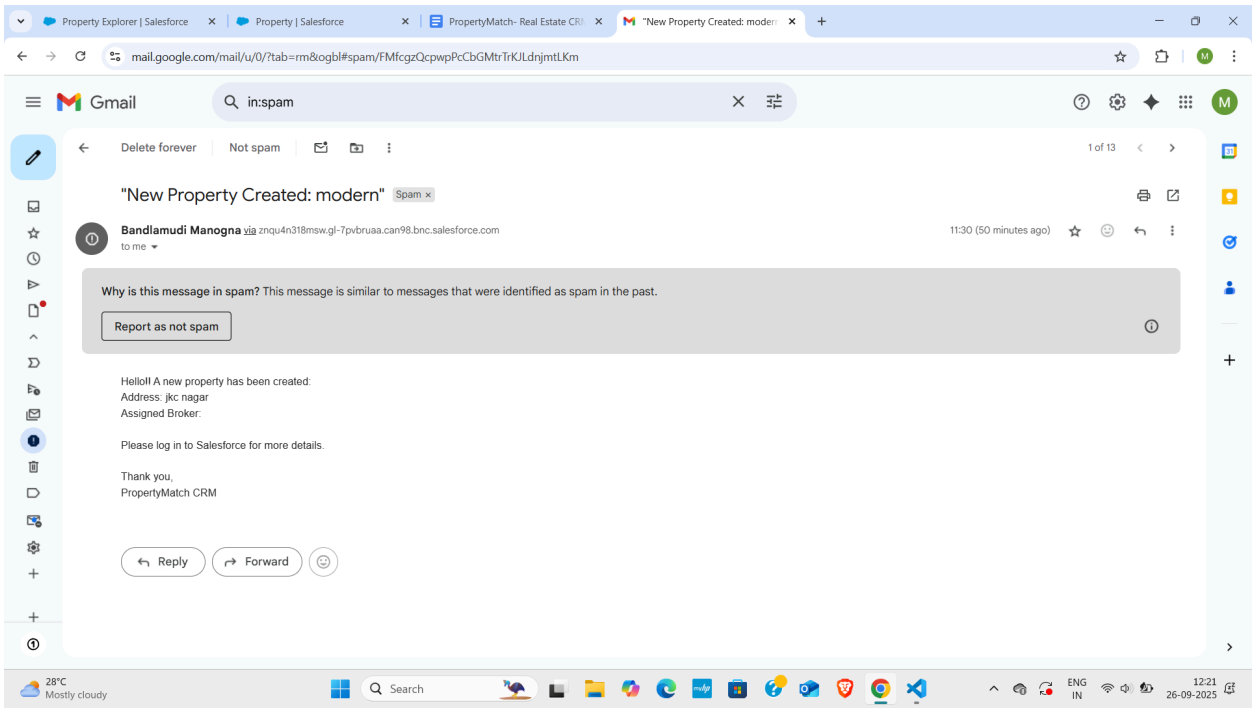
1. Navigate to **Properties Tab → New Property**.
2. Enter property details:
3. Click **Save**.

Expected Result:

- A property record is created in Salesforce.
- An **email is triggered automatically** to the assigned broker.
- Email contains property name, price, city, and assigned broker details.



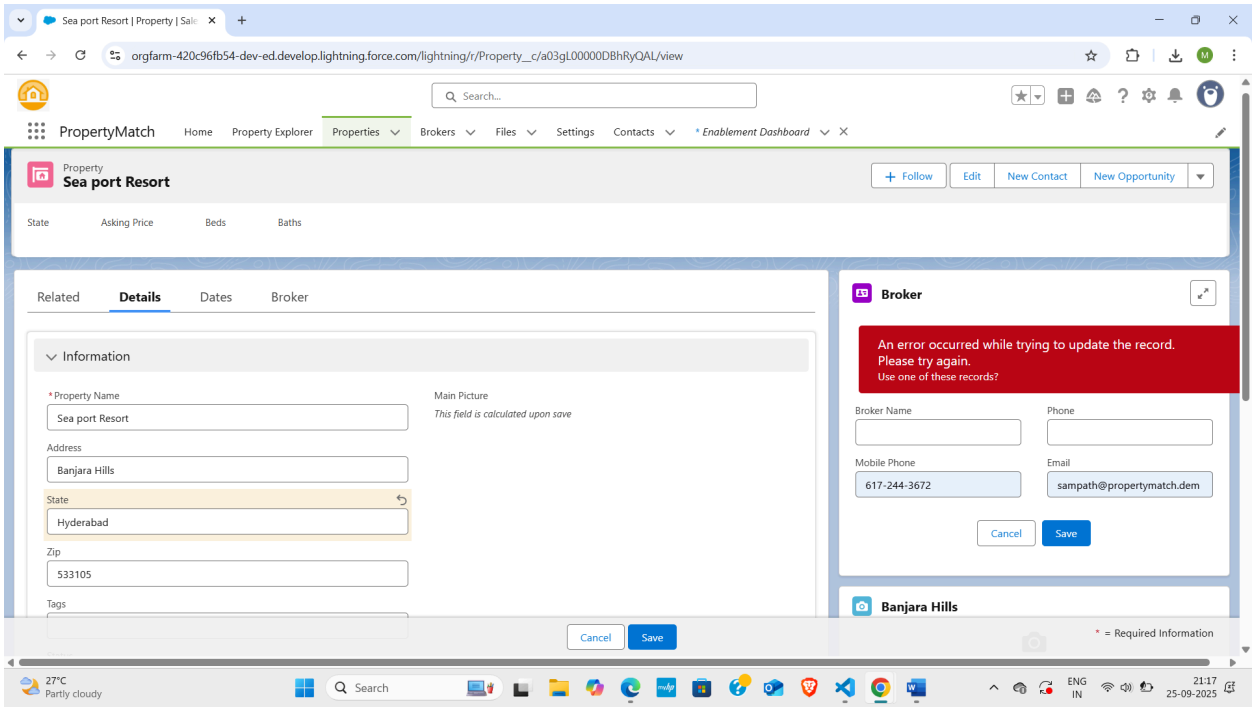
Actual Result (Obtained Email):



Status: Test Passed – Email triggered successfully with correct property details.

Test Case 3: Duplicate Rule (Broker Email)

- **Use Case / Scenario:** Prevent duplicate broker creation.
- **Test Steps (Input):**
    - Try creating a new Broker with Email = “sampath@propertymatch.com” (already exists).
- **Expected Result:**
    - Error preventing duplicate broker.
- **Actual Result:**
    - Error displayed.

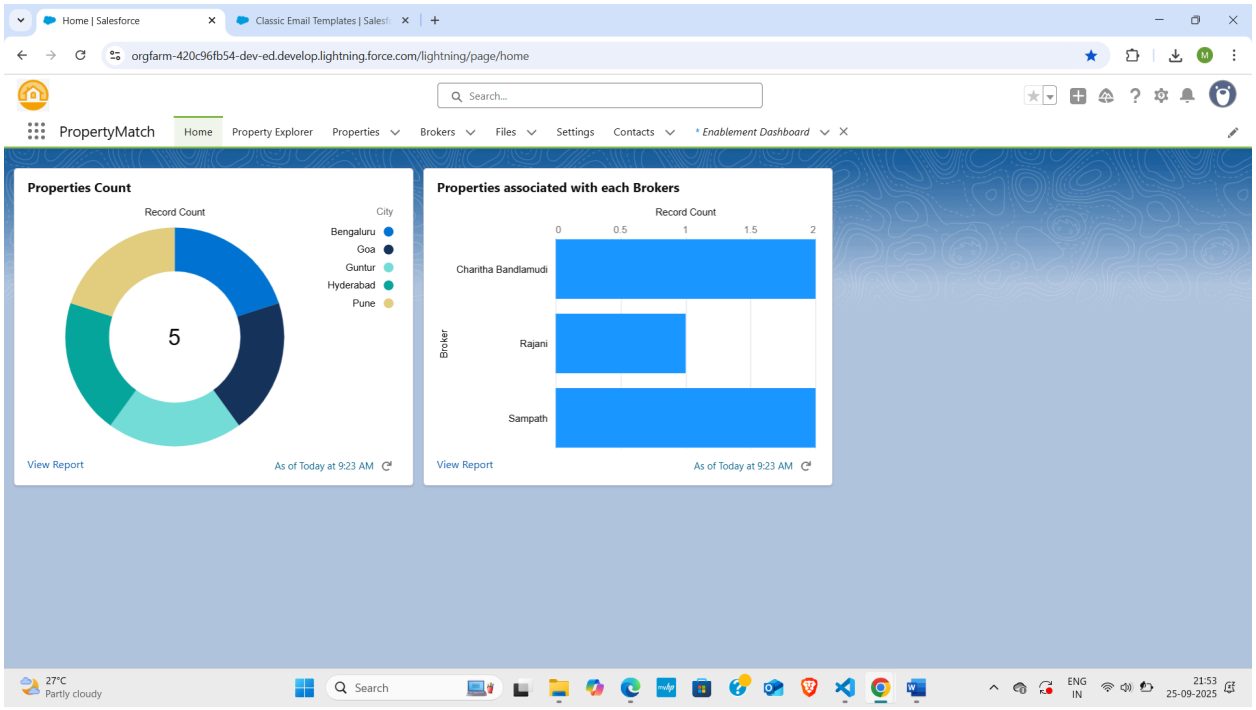


Test Case 4: Report & Dashboard Update

- **Use Case:** When a new Property and Broker are created, the “Properties count” and “Properties assigned to each broker's” report and dashboard should reflect

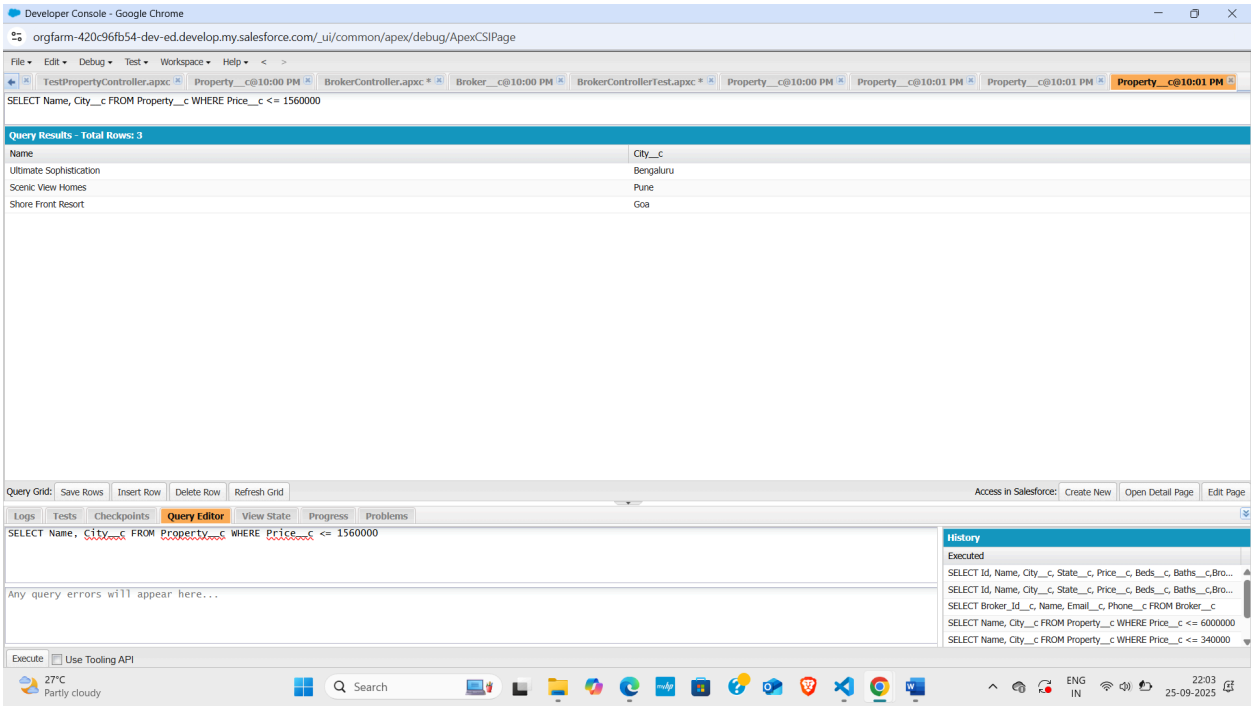
the increased count.

- **Steps:** Create new Property = *Sea port Resort, Hyderabad* assigned to *Charitha Bandlamudi*.
- **Expected Result:** Report shows increased property count for Hyderabad; Dashboard chart updates automatically.
- **Actual Result:** Counts updated successfully



Test Case 5: SOQL Query Execution

- **Use Case:** Validate SOQL returns correct filtered property list.
- **Steps:** Run query → `SELECT Name, City__c FROM Property__c WHERE Price__c <= 1560000`.
- **Expected Result:** Returns all properties ≤ 1560,000.
- **Actual Result:** Correct filtered list returned.



Conclusion

The **PropertyMatch – Real Estate CRM** project successfully addressed the key challenges in the real estate domain by providing a centralized, Salesforce-based platform for managing properties, brokers, and client interactions.

Through the use of **custom objects (Property, Broker)**, **Lightning Web Components**, and **Apex Controllers**, the application enabled users to filter properties, view detailed records, and visualize locations using **Google Maps integration with OAuth 2.0**. Administrators could seamlessly create and assign properties, while **flows and email alerts** ensured that brokers were instantly notified of new assignments.

The project also implemented strong **data management practices** (Data Import Wizard, Data Loader, Duplicate Rules, Data Export), along with robust **security and sharing settings** (Profiles, Roles, OWD, Sharing Rules, Field-Level Security), ensuring compliance and controlled data access. **Reports and Dashboards** provided actionable insights into property distribution, broker performance, and city-wise trends.

Comprehensive **testing and quality assurance** validated all features, from record creation to automation, integration, and security, guaranteeing system reliability and user satisfaction.

In conclusion, **PropertyMatch – Real Estate CRM** delivers a scalable, secure, and user-friendly solution that enhances operational efficiency for administrators, improves visibility for brokers, and provides an intuitive property search experience for end-users. This implementation not only fulfills the initial requirements but also establishes a strong foundation for future enhancements such as advanced analytics, AI-driven property recommendations, and mobile-first access.

Future Enhancements

In the future, this Real Estate CRM can be enhanced with advanced features such as integrating a chatbot for instant customer assistance, AI-based property recommendations based on user preferences, and predictive analytics for pricing trends. Mobile app support and integration with third-party payment or booking systems can also be included to make the platform more interactive, user-friendly, and business-ready.