APPLICATION TO MAKE THE GAS FILLING SATATION EASY USING CRM (DEVELOPER)

FullName : Amara Chandrika
Roll.No : 22A51A0567
College Email Id : 22a51a0567@adityatekkali.edu.in

ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT
TEKKALI, SRIKAKULAM

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

[TITLE OF THE PROJECT]

APPLICATION TO MAKE FILLING GAS EASY USING CRM

OBJECTIVES:

The primary objective of this project is to **modernize and digitize gas station operations** through a **cloud-based CRM solution** developed on the Salesforce platform. Traditional methods of fuel booking, inventory tracking, and customer service are often fragmented, manual, and error-prone. This application addresses those challenges by delivering a **centralized, automated, and intuitive system** tailored to the specific needs of gas filling stations.

Key Goals:

- **Simplify Booking & Tracking:**
  Empower customers with an easy-to-use interface to book fuel refills and track their requests in real-time.
- **Automate Order & Inventory Management:**
  Streamline backend operations for staff with automated order processing, inventory alerts, and workflow approvals.
- **Enhance Decision-Making with Reports:**
  Provide administrators with real-time dashboards, insightful reports, and operational metrics to support data-driven decisions.
- **Reduce Manual Errors & Speed Up Service:**
  Leverage Salesforce's low-code automation tools such as Flows, Validation Rules, and Process Builders to minimize human error and improve efficiency.
- **Scalable & Secure CRM Platform:**
  Ensure role-based access, audit trails, and scalability to support multiple store branches or locations in the future.

PROJECT OVERVIEW:

The **Gas Filling Store CRM Application** is a comprehensive, cloud-based solution designed to **streamline and modernize the gas filling process** for both customers and store operators. Developed on the Salesforce CRM platform, this application enhances customer engagement, optimizes internal workflows, and improves service efficiency.

By leveraging Salesforce's powerful CRM capabilities—including automation flows, custom objects, role-based access control, and real-time reporting—the application addresses the unique operational challenges of gas filling stations. The goal is to provide a **user-friendly**, **feature-rich**, and **scalable** platform that not only digitizes booking and inventory management but also delivers a seamless experience for all stakeholders.

Why this project?

Imagine you own a gas station and manage hundreds of customers every day. You need to track who filled fuel, how much fuel is available, how much money was collected, and when to order more fuel from suppliers. Doing this manually can be messy, error-prone, and time-consuming.

This CRM solves all these problems by digitizing the gas station operations. It helps the staff handle bookings, payments, and inventory, while managers get real-time dashboards and reports to make decisions instantly."

[Phase-1]

Requirement Analysis & Planning:

Understanding Business Requirements

The Gas Filling Store CRM application aims to streamline the fuel booking, supply tracking, and customer interaction process by leveraging Salesforce CRM. The primary users — station staff, customers, and management — all face challenges that are resolved through this centralized, automated system.

User Needs:

- Gas Station Staff need a simplified way to manage fuel supply, orders, and customer data.
- Customers expect a fast and transparent booking experience with multiple payment options.
- Administrators require real-time dashboards to monitor fuel usage, inventory, and operations.

Problems to Be Solved:

- Manual bookings and inventory tracking are error-prone and inefficient.
- Delayed order management due to a lack of automation.
- Limited operational visibility for administrators.
- No centralized system to view customer transactions, fuel inventory, and supplier performance.
- Repetitive tasks cause wasted time and human error.

Defining Project Scope and Objectives

This project focuses on digitizing and automating key processes at a gas filling station using Salesforce's powerful low-code tools and CRM capabilities.

Scope Includes:

- Designing custom objects: Fuel Details, Supplier, Buyer, and Gas Station.
- Enabling full customer lifecycle tracking — from fuel booking to payment.
- Creating automated workflows (Flows, Process Builder) for inventory and order updates.
- Implementing dashboards and reports for performance monitoring.
- Configuring security via roles, profiles, and sharing rules.

Project Objectives:

- Simplify the fuel booking process with digital interfaces.
- Automate fuel inventory management and supplier coordination.
- Provide real-time insights through customizable dashboards.
- Reduce manual intervention and improve customer service speed.
- Ensure secure access to sensitive business data.

Data Model and Security Model

Data Model (Custom Objects & Relationships):

S.No	Object Name	Fields (Sample)
1	**Fuel Details**	Fuel Supplied (Number), Supplier Name (Lookup), Gas Station (Lookup)
2	**Supplier**	Sum of Fuel Supplied (Roll-up from Fuel Details)
3	**Gas Station**	Fuel Supplied (Roll-up), Fuel Price/Litre, Fuel Used (Roll-up), Fuel Available
4	**Buyer**	Customer Info, Fuel Filled, Vehicle Type (Picklist), Payment Mode (Picklist)

Relationships:

- Buyer ↔ Fuel Details (Lookup or Related List)
- Supplier ↔ Fuel Details (Master-Detail)
- Gas Station ↔ Fuel Details (Master-Detail)

Security Model:

- **Profiles** and **Permission Sets** to control object-level access.
- **Role Hierarchy** to manage record visibility across staff, managers, and admin.
- **Field-Level Security (FLS)** to protect sensitive data like payment info.
- **Sharing Rules** to extend visibility for collaborative roles.
- **Validation Rules** to ensure data quality (e.g., fuel quantity limits, payment entry)

[Phase-2]

Salesforce Development – Backend & Configurations

In this phase, the backend structure, customizations, and automation logic were implemented to build a fully functional and secure CRM for the gas filling station. This phase involved setting up the development environment, customizing Salesforce objects and fields, and building automation through Flows, Reports, and Apex code.

1. Setup Environment & DevOps Workflow

- 🔧 **Sandbox/Developer Org Setup:**
  A Salesforce Developer Org was configured for building and testing the application.
- 🔧 **App Creation:**
  - ✓ Created a **Lightning App** named **"GAS STATION"**, which included all custom object tabs (Supplier, Gas Station, Buyer, Fuel Details).
- 🔧 **User & Role Configuration:**
  - ✓ Set up **three profiles**: Manager, Sales Executive, and Sales Person.
  - ✓ Implemented **Role Hierarchy**:
    - Manager → Sales Executive → Sales Person.
  - ✓ **Permission Set (P1)** created to grant additional access (e.g., Read & Create on Fuel Details).
- 🔧 **OWD & Sharing Settings:**
  - ✓ Organization-Wide Defaults were configured:
    - Gas Station & Supplier objects are set to **Public Read-Only**.
  - ✓ Folder sharing (e.g., **Fuel Estimation Folder**) was configured with role-based access.

2. Customization of Objects, Fields, and Validation Rules

- 🔧 **Custom Objects:**
  - ✓ Supplier, Gas Station, Buyer, and Fuel Details were designed to represent the business data.
- 🔧 **Field Customization:**
  - ✓ Created **lookup relationships** between Fuel Details → Supplier and Gas Station.
  - ✓ Added **Roll-Up Summary fields** to calculate:
    - Total fuel supplied.
    - Fuel available at the gas station.
  - ✓ Created **Formula fields** (e.g., Customer Name, Amount Paid).
  - ✓ Picklist fields like **Vehicle Type** and **Mode of Payment** were configured.
- 🔧 **Validation Rules:**
  - ✓ For example, ensuring fuel price per liter in Gas Station is **greater than 50** before saving.

3. Automation (Flows, Workflow Rules, Approval Processes)

- 🔧 **Flows:**
  - ✓ A **Record-Triggered Flow** was created on the Buyer object:
    - Triggered **on record creation or update**.
    - Sends a **Thank You Email** to customers including details like fuel quantity, vehicle type, and amount paid.
- 🔧 **Workflow Rules & Process Builder:**
  - ✓ Auto-updates inventory or triggers approval processes when thresholds are reached.
- 🔧 **Reports & Dashboards:**
  - ✓ Reports (e.g., “Amount Range”) and **Dashboard components** were built to visualize fuel consumption and payments.

4. Apex Classes and Triggers

- 🔧 **Apex Class:**

  Created FuelRecordHandler to enforce business rules:

  - ✓ Prevent deletion of Fuel Details if Fuel Supplied > 500.
  - ✓ Validate that Fuel Price/Liter > 50 in the Gas Station object.

- 🔧 **Triggers:**

  - ✓ **Before Delete Trigger:** On Fuel Details to call beforeDeleteInfo().
  - ✓ **Before Insert Trigger:** On Gas Station to validate fuel price using beforeDeleteGas().

- 🔧 **Asynchronous Apex (if applicable):**

  While no explicit asynchronous class was required, this can be extended in the future for:

  - ✓ Large data processing (e.g., bulk inventory updates).
  - ✓ Scheduled batch jobs to monitor stock levels.

[Phase-3]

UI/UX Development & Customization

The UI/UX development phase focused on making the **GAS STATION CRM** application visually structured, user-friendly, and efficient for day-to-day operations. This involved customizing the Lightning experience, page layouts, and dashboards to deliver an intuitive interface for all user profiles.

1. Lightning App Setup via App Manager

- A **Lightning App** named **"GAS STATION"** was created through the **App Manager**.
- Key features:
  - ✓ **Custom Tabs:** Supplier, Gas Station, Buyer, and Fuel Details.
  - ✓ **App Branding:** Added a custom color and icon to reflect the Gas Station theme.
  - ✓ Optimized for **Salesforce Lightning Experience** for a modern UI.

2. Page Layouts & Dynamic Forms

- **Customized Page Layouts** for each object (Supplier, Gas Station, Buyer, Fuel Details):
  - ✓ Grouped fields logically (e.g., Customer details, Vehicle details, Fuel details).
  - ✓ Adjusted the placement of roll-up summary and formula fields for better visibility.
- **Dynamic Forms** enabled on record pages to:
  - ✓ Show/hide fields based on conditions (e.g., display Payment Mode only if Amount Paid > 0).
  - ✓ Improve performance and reduce clutter for end users.

3. User Management

- **Profiles** (Manager, Sales Executive, Sales Person) were assigned custom page layouts and permissions for controlled access.
- **Role Hierarchy** ensures that:
  - ✓ **Manager** can view all records of Sales Executive and Sales Person.
  - ✓ **Sales Executive** can view records of Sales Person.
- **Permission Set (P1)** applied for additional access (e.g., creating and reading Fuel Details).

4. Reports and Dashboards

- **Report Folder:***Fuel Estimation*.
  - ✓ Created a report **"Amount Range"** with:
    - Columns: Fuel Filled in Vehicle, Amount Paid.
    - Group Rows: Fuel Available in Bunk, Customer Name.
    - **Conditional Formatting:** Highlighted Amount Paid in the range of 1000–5000.
- **Dashboard Folder:***Amount Estimation Dashboard*.
  - Created dashboard components linked to the "Amount Range" report for quick analysis.

5. Lightning Pages

- **Lightning Record Pages** were customized for better navigation:
  - ✓ Added components like **Related Lists**, **Highlights Panel**, and **Charts**.
  - ✓ Used a two-column layout for quick access to both Buyer and Fuel details.
  - ✓ Assigned custom record pages per profile for **Manager** and **Sales Executive**.

[Phase-4]

Data Migration, Testing & Security

1. Data Migration

Tools & approach

Activity	Tool	Used For	Notes
Prototype / small loads	**Data Import Wizard**	**Buyers, simple lookups**	**UI-driven, basic validation**
Bulk loads & reruns	**Data Loader**	**Fuel Details, historical transactions**	**Upsert with External IDs**
Sequencing	**—**	**Supplier → Gas Station → Fuel Details → Buyer**	**Respect M-D & lookup dependencies**
Verification	**Reports / SOQL**	**Spot-check record counts and key fields**	**Pre/post-load reconciliation**

Key steps

- Field mapping sheets prepared for every object (source → Salesforce API name).
- De-dup & cleanse source files (normalize phone/email, numeric ranges).
- Load order enforced to satisfy lookups (Supplier & Gas Station before Fuel Details; Gas Station before Buyer for roll-ups).
- Post-load validations: run exception reports, compare roll-up totals vs. source.

Rollback/Recovery plan

- Export pre-load snapshots.
- Use Data Loader to delete by ID if required, then re-upsert corrected data.

2. Security (Profiles, Roles, Permission Sets, Sharing)

Profiles (object-level & FLS)

- **Manager, Sales Executive, Sales Person**: CRUD tailored per object (e.g., Sales Person cannot edit Supplier).
- Sensitive fields (e.g., **Fuel_Price_per_Litre__c**, **Amount_Paid__c**) restricted via **FLS** for non-manager roles.

Role Hierarchy

Manager
└── Sales Executive
└── Sales Person

- Manager sees Sales Executive & Sales Person records.
- Sales Executive sees Sales Person records.

Permission Set: P1

- Grants **Read & Create** on **Fuel Details** object to selected users without altering base profiles.

OWD & Sharing Rules

- **OWD**: Supplier & Gas Station = **Public Read-Only**.
- **Buyer, Fuel Details** = Private (if configured) with **role-based sharing** for visibility up the hierarchy.

3. Governance: Field History, Duplicate & Matching Rules

Field History Tracking

- Enabled on:
  - ✓ **Gas_Station__c**: Fuel_Price_per_Litre__c, Fuel_Available_in_Bunk__c
  - ✓ **Fuel_Details__c**: Fuel_Supplied__c
  - ✓ **Buyer__c**: Amount_Paid__c, Fuel_Filled_in_Vehicle__c
- Purpose: Audit pricing changes, stock movements, and financial totals.

Duplicate & Matching Rules

- **Buyer__c Matching Rule**: Match on **(Email OR Phone)**.
- **Duplicate Rule**: Block or allow with alert; report duplicates for manager review.

4. Automation & Programmatic Logic to Test

- **Flows**: Record-triggered Flow on **Buyer** (send receipt email).
- **Reports/Dashboards**:
  - ✓ Report "**Amount Range**" with conditional formatting (Sum of Amount Paid: 1000–5000).
  - ✓ Dashboard in folder **Amount Estimation Dashboard**.
- **Apex**:
  - ✓ **FuelRecordHandler** class:
    - beforeDeleteInfo(List<Fuel_Details__c>) → block deletion if Fuel_Supplied__c > 500
    - beforeDeleteGas(List<Gas_Station__c>) → enforce Fuel_Price_per_Litre__c > 50
  - ✓ **Triggers**:
    - before delete on **Fuel_Details__c**
    - before insert on **Gas_Station__ca**

5.Test Classes (Apex)

Targets

- ≥ **75% org-wide coverage** (100% on handler & triggers preferred).
- Bulk, positive, negative, and mixed DML scenarios.

4.6 Functional Test Cases (samples)

#	Feature	Scenario	Input	Expected Output
1	Buyer Flow	Create Buyer	Valid email, Fuel_Filled=10	Email sent with receipt details
2	Conditional Formatting	Report “Amount Range”	Amount Paid sums between 1000–5000	Rows highlighted per rule
3	Trigger Validation	Insert Gas_Station with price 40	Fuel_Price_per_Litre__c=40	DML fails with validation error
4	Trigger Validation	Delete Fuel_Details with 600 liters	Fuel_Supplied__c=600	Deletion blocked
5	Roll-ups	Create Fuel_Details for Supplier	Multiple child rows	Supplier.Sum_of_Fuel_Supplied updated
6	Sharing	Sales Exec views Sales Person record	Login as Sales Exec	Record visible
7	Duplicate Rule	Create Buyer with existing Phone	Same Phone	Duplicate blocked/alert shown

Non-functional tests

- **Bulk load** 200 Fuel_Details rows to confirm trigger bulk safety.
- **Security**: Verify FLS/CRUD prevents low-profile users from editing restricted fields.
- **UAT**: Manager validates dashboards, flows, and reports against business scenarios.

[Phase-5]

Deployment, Documentation & Maintenance

This phase focuses on moving the developed solution from the development environment to production, documenting key configurations, and ensuring ongoing maintenance and troubleshooting strategies.

1.Deployment Strategy

Deployment Method:

- 🔧 **Change Sets:**
  - ✓ All custom objects (Supplier, Gas Station, Buyer, Fuel Details), fields, validation rules, flows, Apex classes, triggers, and dashboards were included in an **Outbound Change Set**.
  - ✓ This change set was deployed from the **Developer Sandbox** to the **Production Org**.
- 🔧 **Steps Followed:**
  - ✓ Created an **Outbound Change Set** in the sandbox.
  - ✓ Added all custom components (Objects, Flows, Reports, Dashboards, Apex Classes, Triggers, Permission Sets).
  - ✓ Validated the change set in production to ensure no errors.
  - ✓ Executed the deployment during a low-traffic period to minimize disruption.
- 🔧 **Alternative Methods (Future Scope):**
  - ✓ **Salesforce CLI (SFDX):** For automated deployments and CI/CD.
  - ✓ **ANT Migration Tool:** For handling large deployments and scripts.

2. Maintenance and Monitoring

System Maintenance:

- 🔧 **User Management:**
  - ✓ Periodic review of profiles and permission sets to ensure proper access control.
  - ✓ Addition of new users or roles as required by business operations.
- 🔧 **Data Integrity:**
  - ✓ Regular review of **field history tracking** for critical fields like Fuel Price, Fuel Available, and Amount Paid.
  - ✓ Duplicate management with matching rules and duplicate rules.
- 🔧 **Reports & Dashboards:**
  - ✓ Scheduled refresh of dashboards (e.g., **Amount Estimation Dashboard**).
  - ✓ Monthly review of data trends and KPI performance metrics.

Monitoring Activities:

- 🔧 Apex **debug logs** are used to monitor trigger and class executions for any anomalies.
- 🔧 Continuous monitoring of **flow failures** and email alerting on error events.
- 🔧 Scheduled health checks for large data volumes (Fuel Details and Buyer transactions).

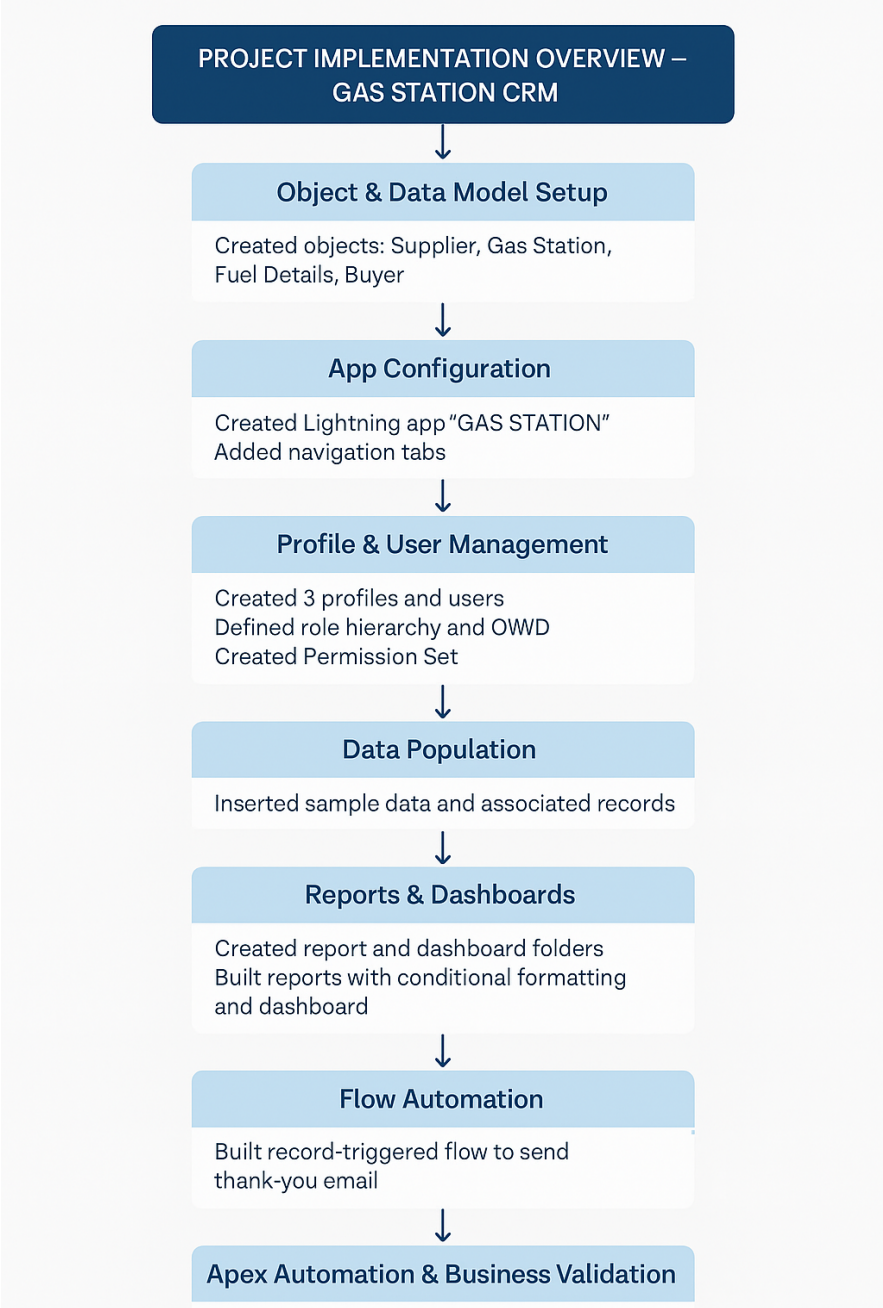
3 .Troubleshooting Approach

Common Troubleshooting Steps:

- 🔧 **Flows & Automations:**
  - ✓ Check the **Flow Error Logs** in the setup if a record-triggered flow fails (e.g., Buyer email flow).
  - ✓ Validate that all required fields have proper values.
- 🔧 **Apex & Triggers:**
  - ✓ Use **Debug Logs** to trace trigger events if records are blocked during insert or delete (e.g., Fuel Details exceeding 500 liters).
  - ✓ Modify or disable automation temporarily if it is causing unexpected data issues.
- 🔧 **Data Issues:**
  - ✓ Use **Reports** to find missing or incorrect data.
  - ✓ Correct bulk data using **Data Loader** or **Data Import Wizard**.
- 🔧 **Permissions & Security Errors:**
  - ✓ Review **OWD, role hierarchy, and field-level security** if users face access-related issues.
  - ✓ Adjust permission sets (P1) to grant specific rights.

Project Implementation Overview – GAS STATION CRM

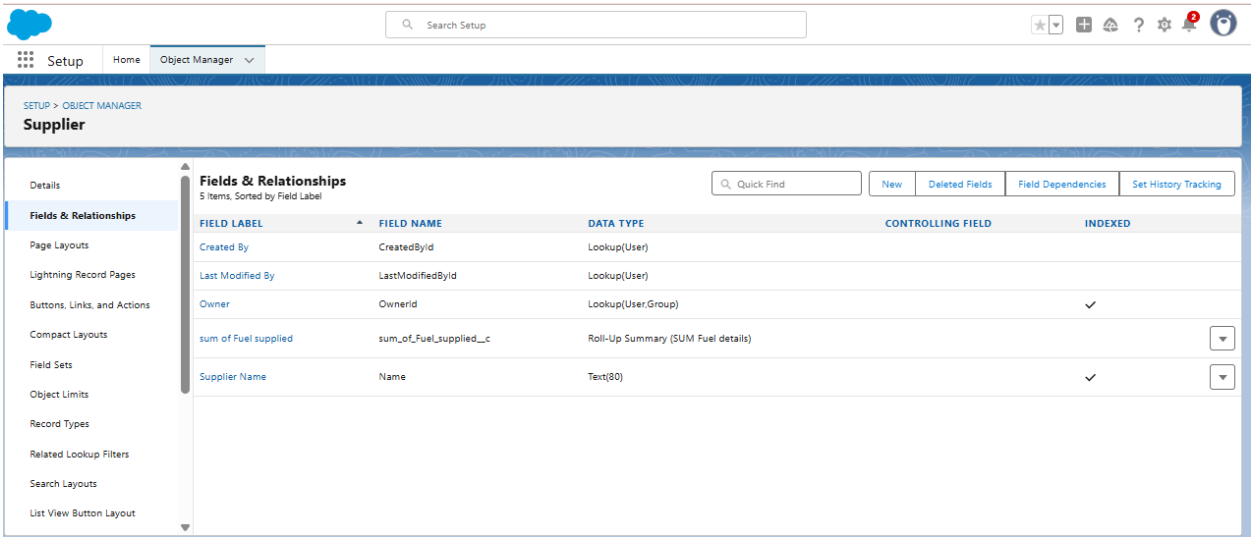
This Salesforce CRM project was designed and implemented by following a structured approach that covers object modeling, security, automation, reporting, and customization.

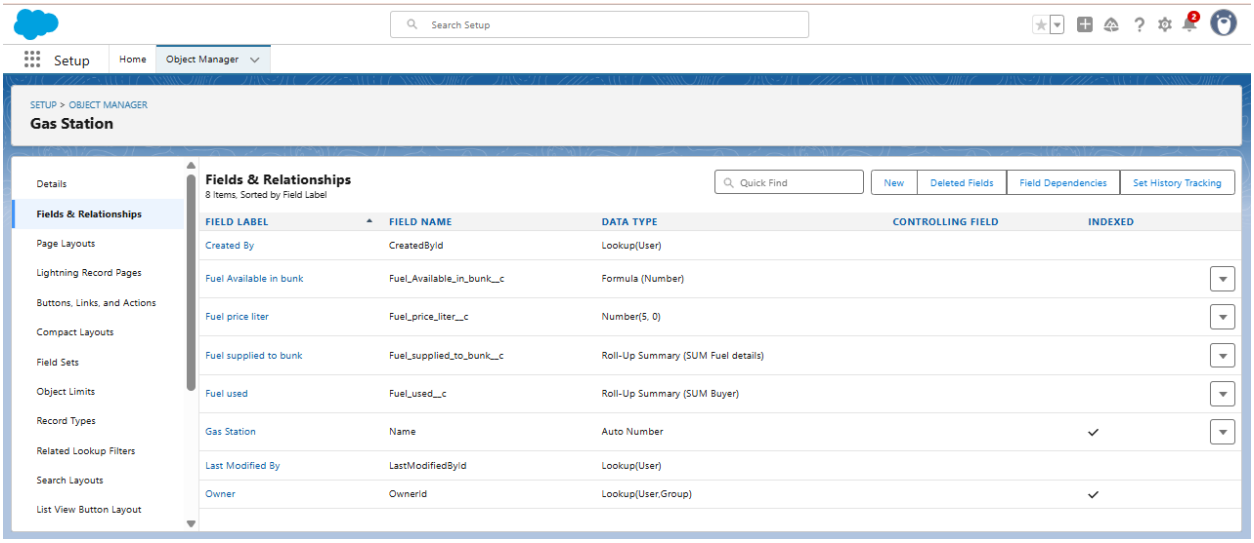


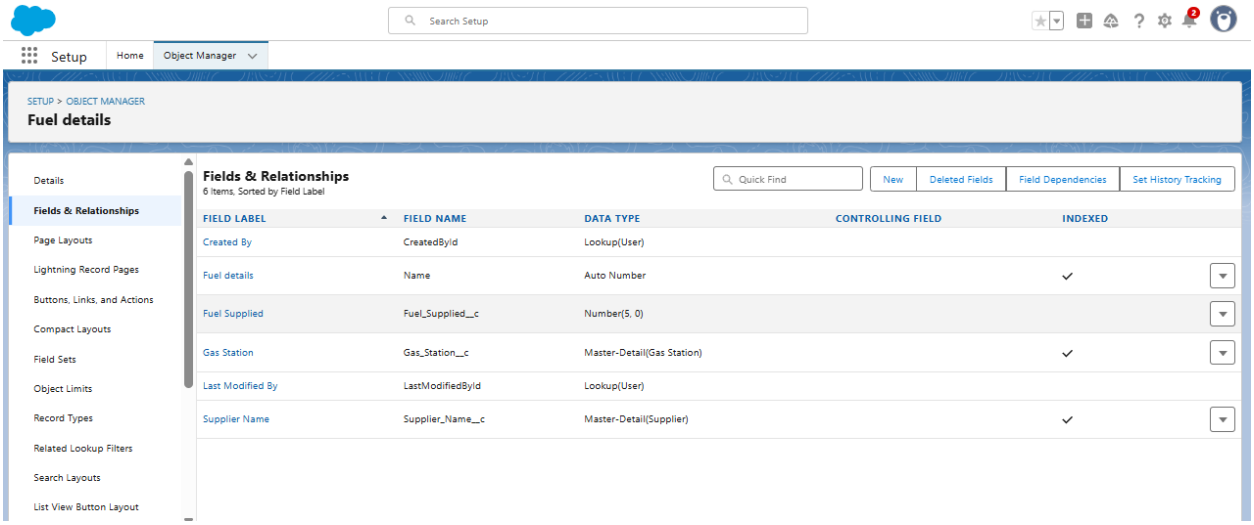
1. Object & Data Model Setup

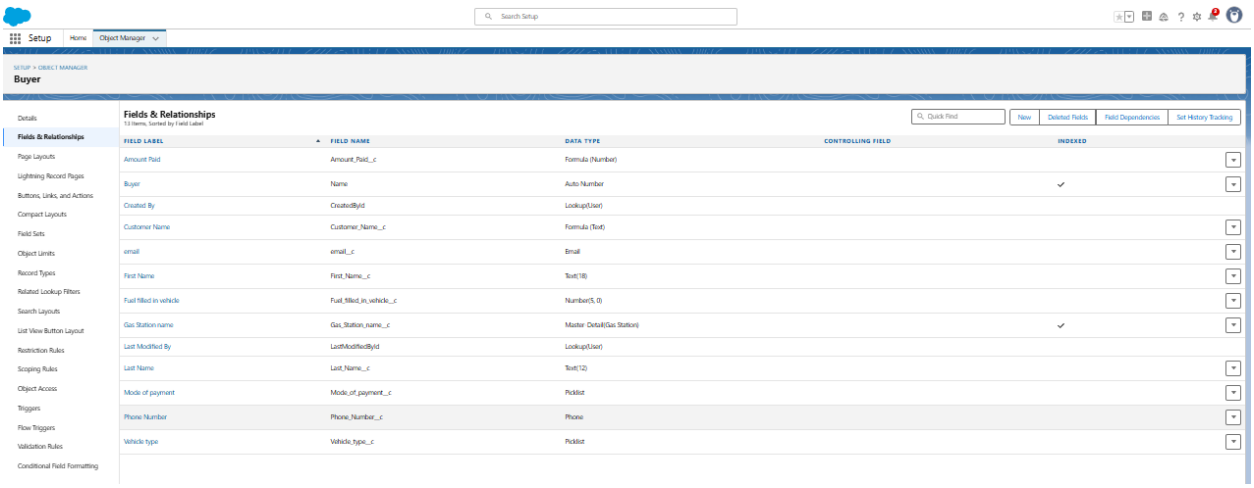
- Created **custom objects**:
  - ✓ Supplier
  - ✓ Gas Station
  - ✓ Fuel Details
  - ✓ Buyer
- Added relevant **fields**, including:
  - ✓ Lookup fields for relationships (e.g., Fuel Details → Supplier & Gas Station)
  - ✓ Roll-up summaries for fuel metrics
  - ✓ Picklists for Vehicle Type and Mode of Payment
  - ✓ Formulas for dynamic calculations like Customer Name and Amount Paid

Objects and their respective Fields



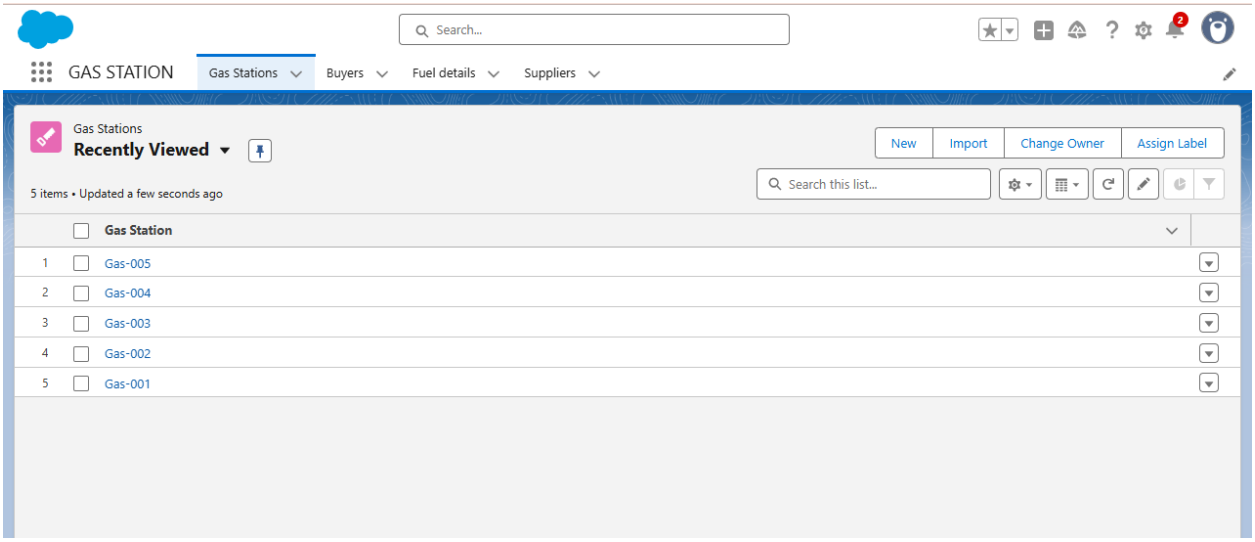

Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Gas Station
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Fields & Relationships
8 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
FIELD LABEL
FIELD NAME
DATA TYPE
CONTROLLING FIELD
INDEXED
Created By
CreatedById
Lookup(User)
Fuel Available in bunk
Fuel_Available_in_bunk__c
Formula (Number)
Fuel price liter
Fuel_price_liter__c
Number(5, 0)
Fuel supplied to bunk
Fuel_supplied_to_bunk__c
Roll-Up Summary (SUM Fuel details)
Fuel used
Fuel_used__c
Roll-Up Summary (SUM Buyer)
Gas Station
Name
Auto Number
Last Modified By
LastModifiedById
Lookup(User)
Owner
OwnerId
Lookup(User,Group)


Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Fuel details
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
Fields & Relationships
6 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
FIELD LABEL
FIELD NAME
DATA TYPE
CONTROLLING FIELD
INDEXED
Created By
CreatedById
Lookup(User)
Fuel details
Name
Auto Number
Fuel Supplied
Fuel_Supplied__c
Number(5, 0)
Gas Station
Gas_Station__c
Master-Detail(Gas Station)
Last Modified By
LastModifiedById
Lookup(User)
Supplier Name
Supplier_Name__c
Master-Detail(Supplier)


Setup
Home
Object Manager
Buyer
Fields & Relationships
Amount Paid
Amount_Paid__c
Formula (Number)
Buyer
Name
Auto Number
Created By
CreatedById
Lookup(User)
Customer Name
Customer_Name__c
Formula (Text)
email
email__c
Email
First Name
First_Name__c
Text(18)
Fuel filled in vehicle
Fuel_filled_in_vehicle__c
Number(5, 0)
Gas Station name
Gas_Station_name__c
Master-Detail(Gas Station)
Last Modified By
LastModifiedById
Lookup(User)
Last Name
Last_Name__c
Text(12)
Mode of payment
Mode_of_payment__c
Picklist
Phone Number
Phone_Number__c
Phone
Vehicle type
Vehicle_type__c
Picklist

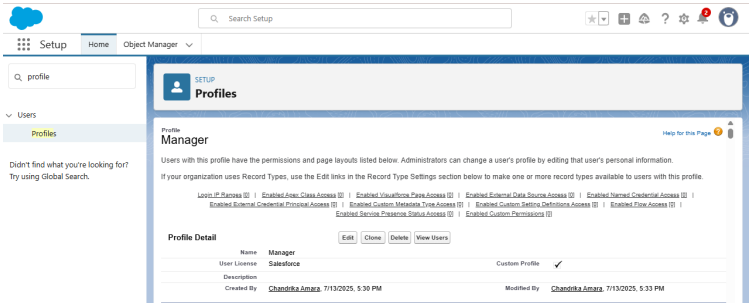
2. App Configuration

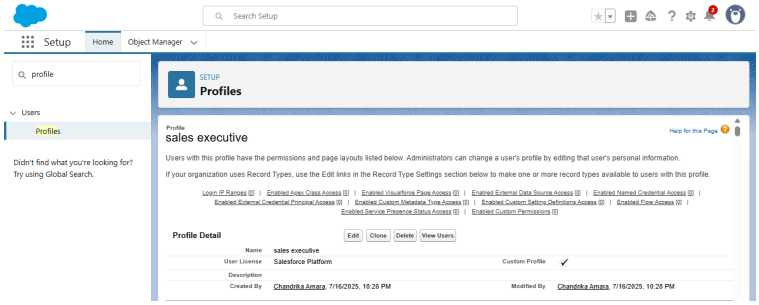
- Created a **Lightning App** named: **“GAS STATION”**
- Added navigation tabs for key objects:
  - ✓ Supplier
  - ✓ Gas Station
  - ✓ Buyer
  - ✓ Fuel Details

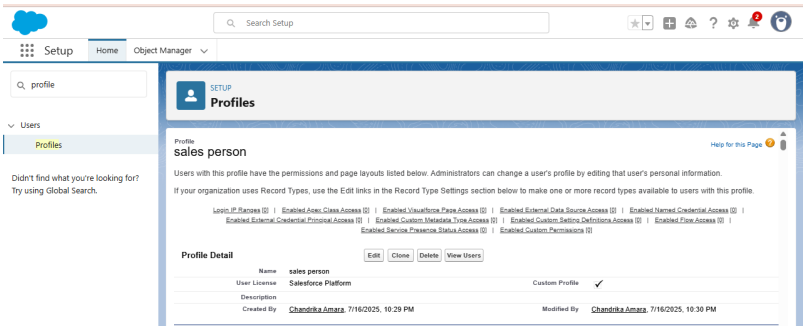


3. Profile & User Management

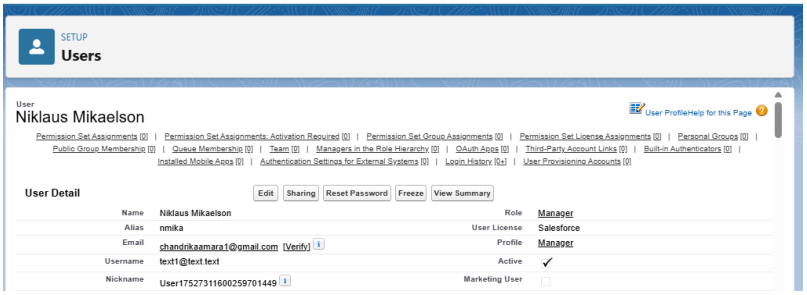
- Created **3 custom profiles**:
  - ✓ Manager
  - ✓ Sales Executive
  - ✓ Sales Person

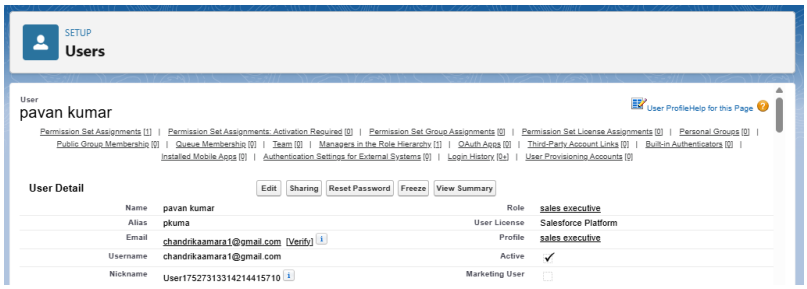


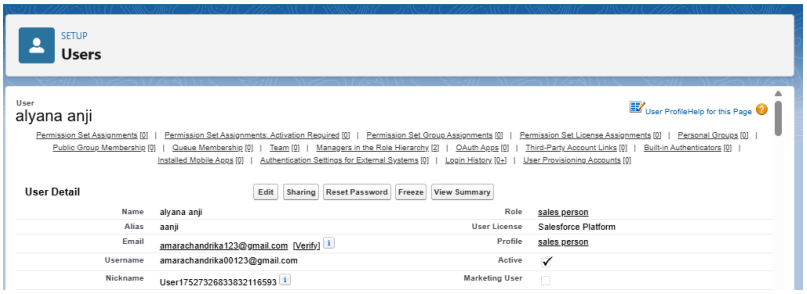




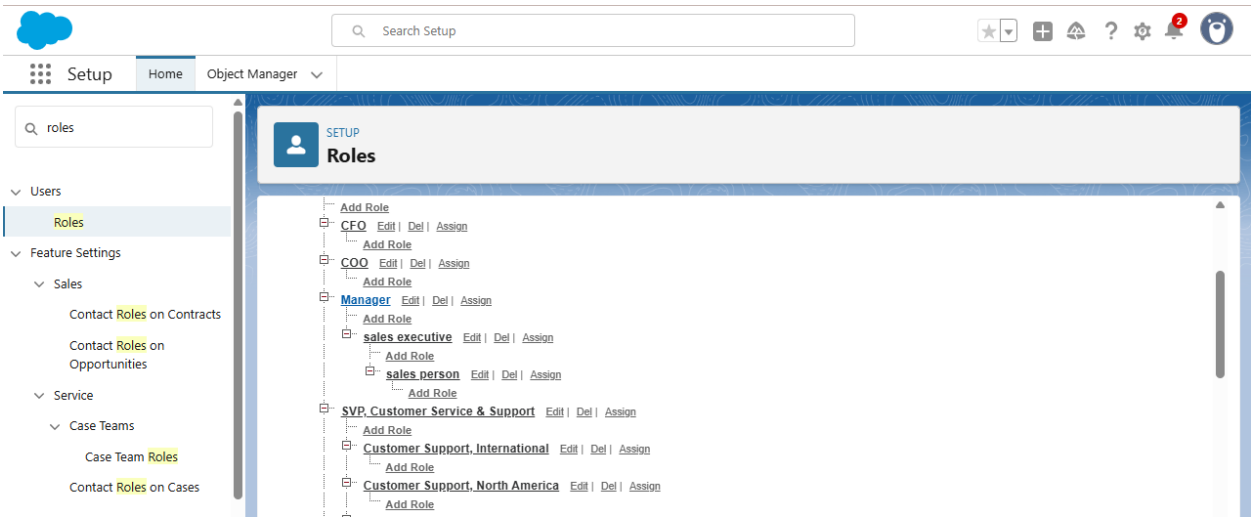
- Created **three users** and assigned each to a corresponding profile
  - ✓ **Manager**-->Niklaus Michelson
  - ✓ **Sales Executive** -->pavan markup
  - ✓ **Sales Person**--->alyana anti



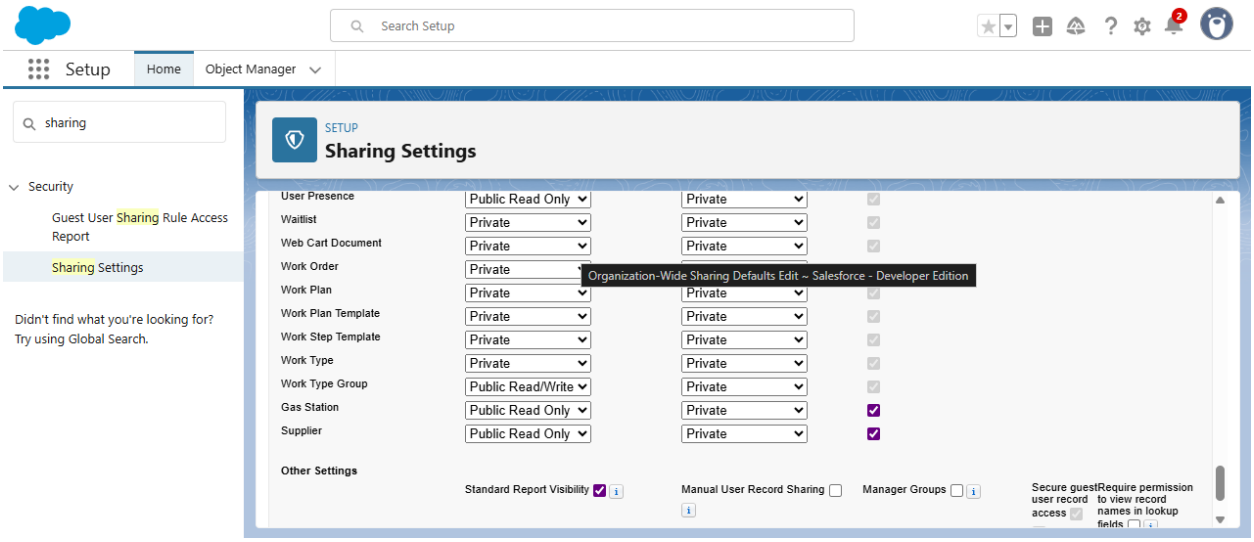




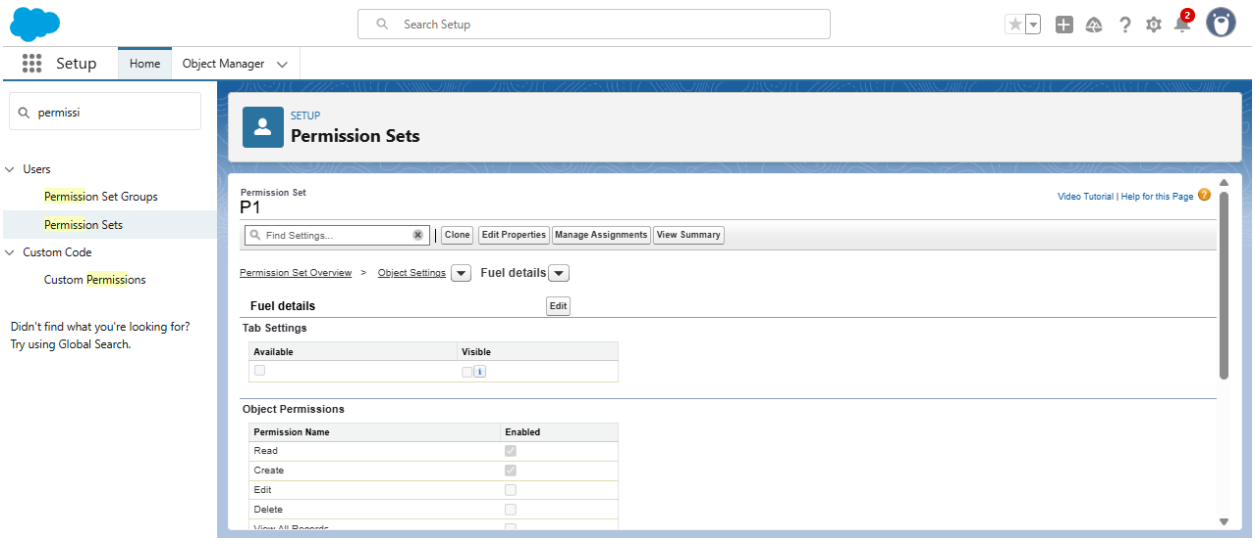
- Defined a **role hierarchy**:
  - ✓ Manager can access records of Sales Executives and Sales Persons
  - ✓ Sales Executive can view records of Sales Persons



- Configured **OWD (Organization-Wide Defaults)**:
  - ✓ Set **Public Read-Only** for Gas Station and Supplier objects

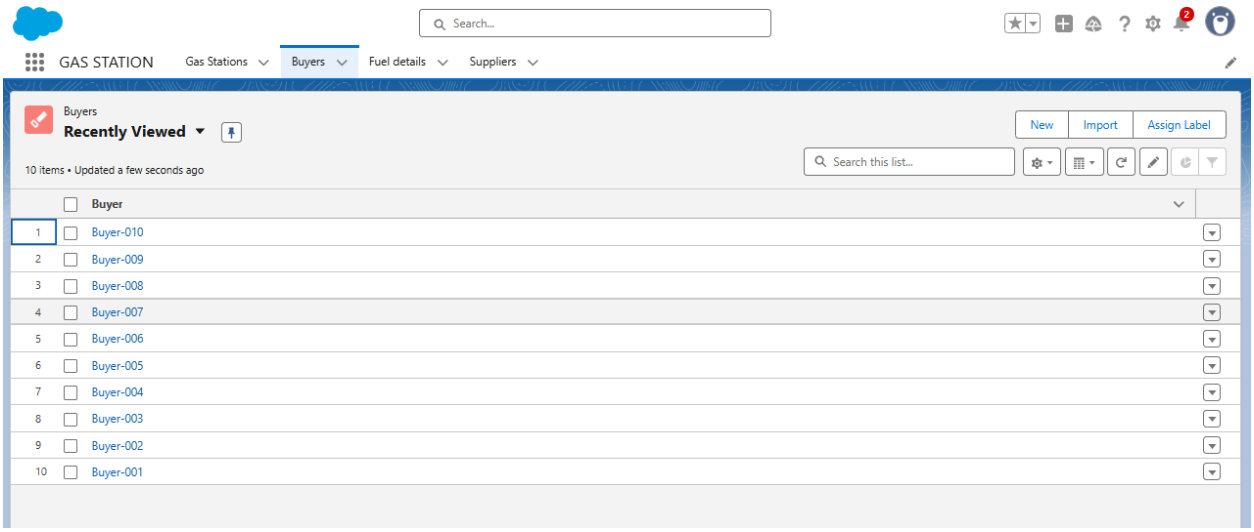


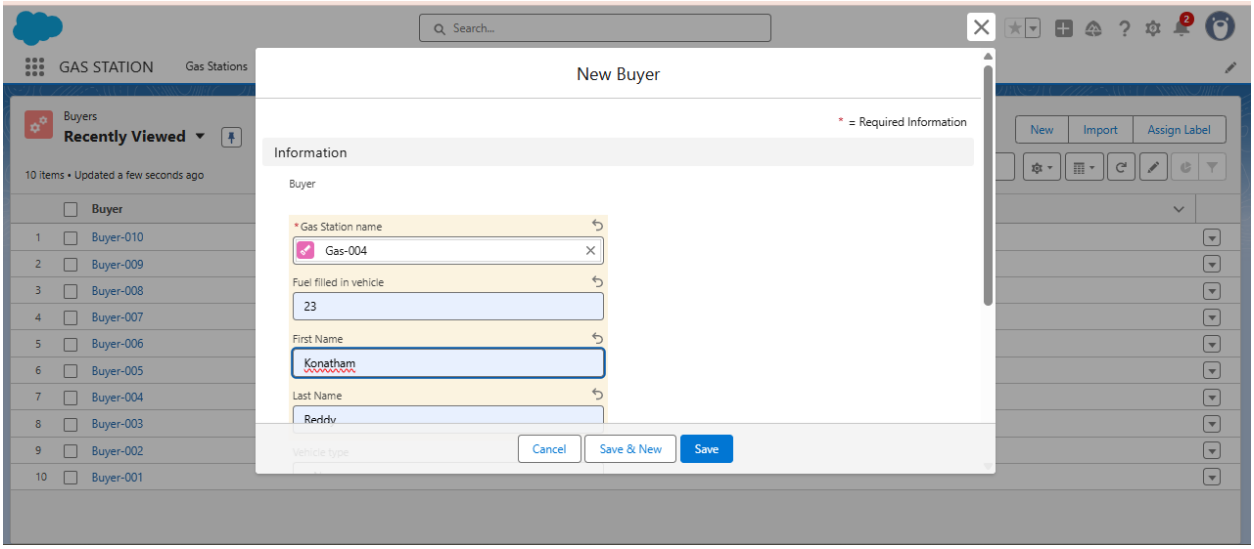
- Created a **Permission Set (P1)**:
  - ✓ Granted **Read** and **Create** access for Fuel Details object



4. Data Population

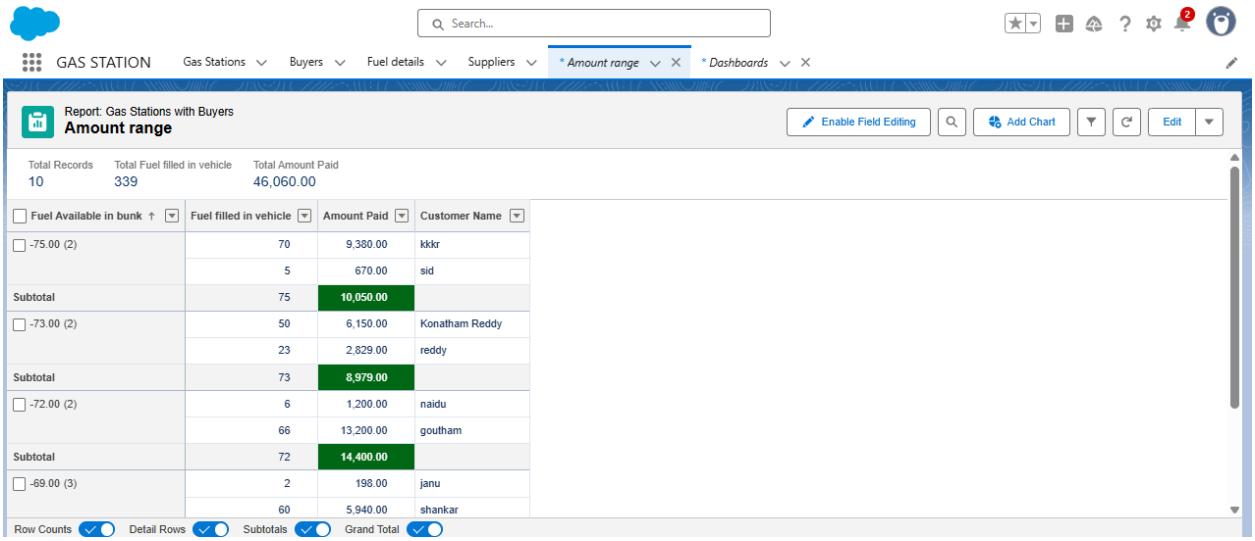
- Inserted sample data:
  - ✓ 10+ records in the Buyer object
  - ✓ Associated buyers with relevant fuel data and gas stations



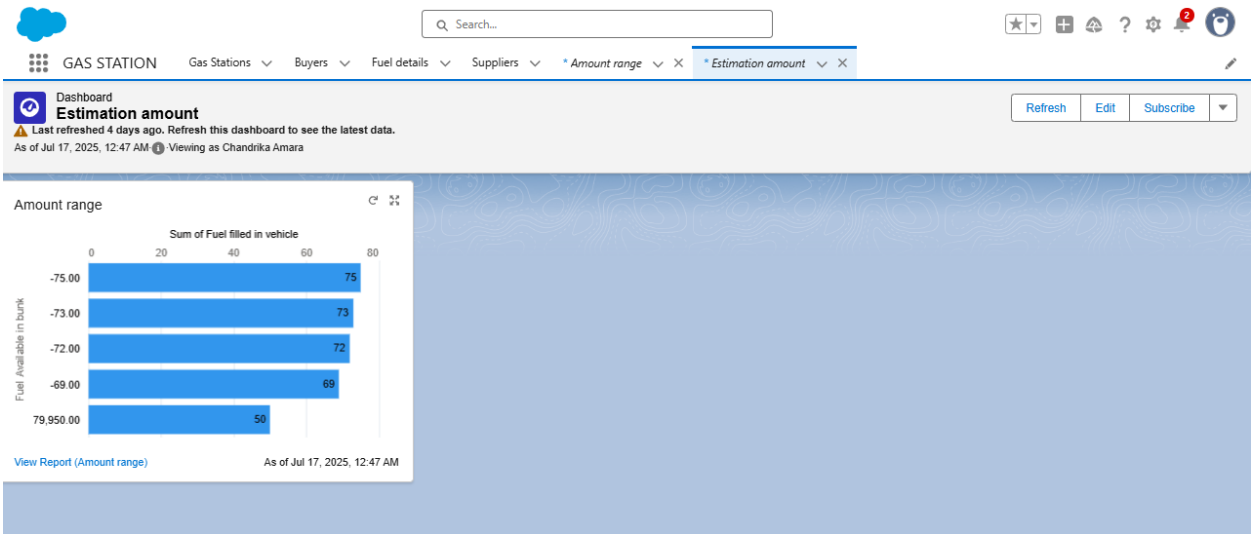


5. Reports & Dashboards

- Created a **Report Folder**: “Fuel Estimation”
- Created a **Report**: “Amount Range”
  - ✓ Report Type: *Gas Station with Buyers*
  - ✓ Displayed: Fuel Filled, Amount Paid, Fuel Available, Customer Name
  - ✓ Applied **Conditional Formatting** on Amount Paid (range: 1000–5000)

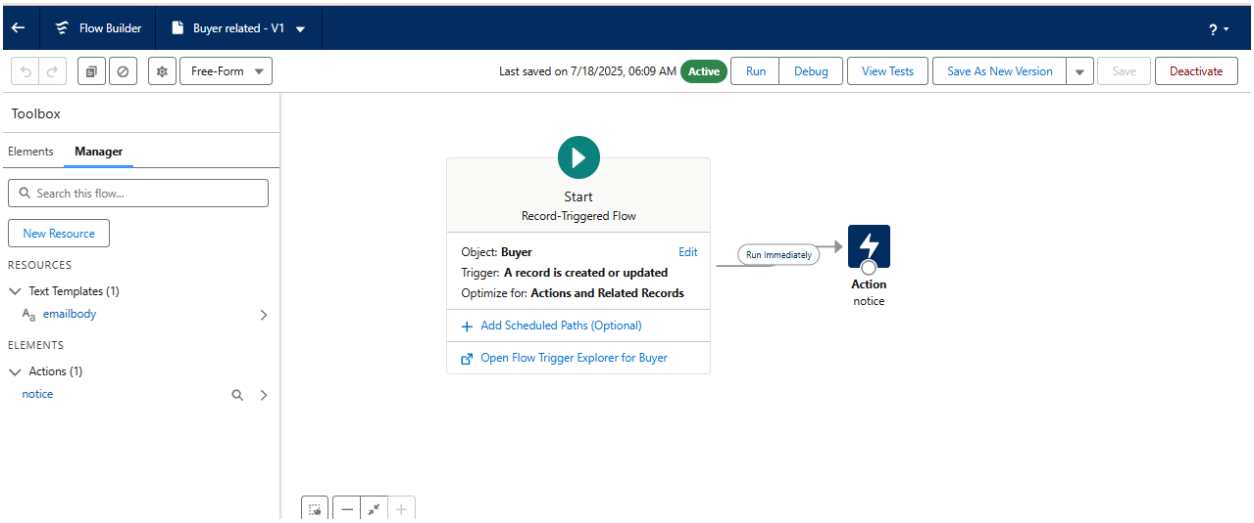


- Created a **Dashboard Folder**: “Amount Estimation Dashboard”
- Built a **Dashboard**:
  - ✓ Added the “Amount Range” report as a dashboard component



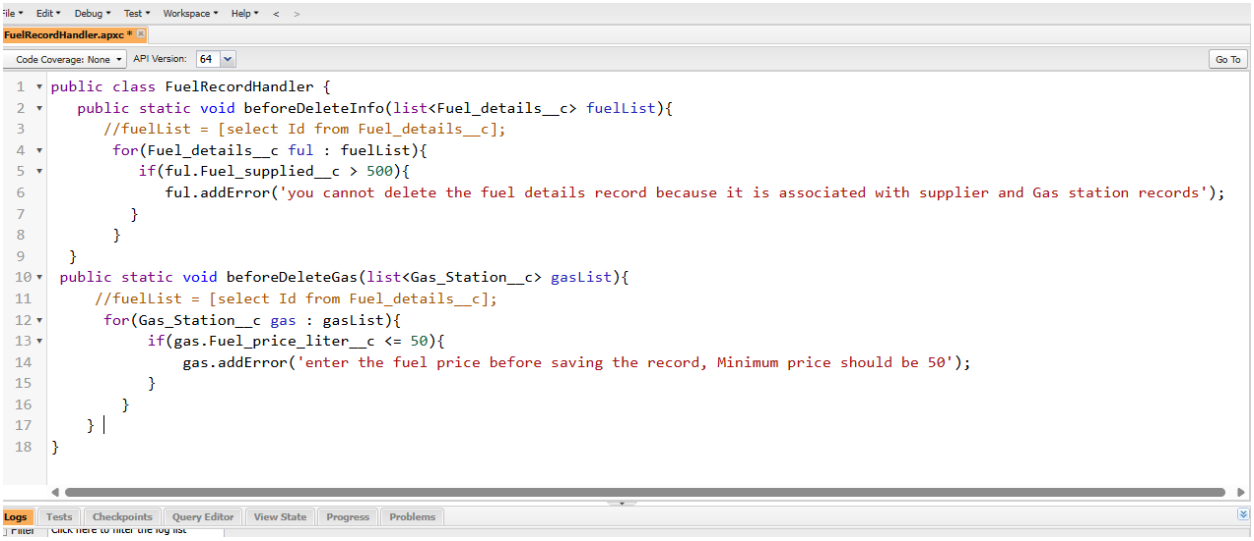
6. Flow Automation

- Built a **Record-Triggered Flow** on the Buyer object:
  - ✓ Triggered on **Create or Update**
  - ✓ Sent a **thank-you email** to the buyer using a **Text Template**
  - ✓ Email included details like the amount paid, fuel intake, and vehicle type



7. Apex Automation & Business Validation

- Created an **Apex Class**: FuelRecordHandler
  - ✓ **Fuel Details**:
    - Prevents deletion if fuel supplied > 500
  - ✓ **Gas Station**:
    - Validates fuel price must be > 50 before saving




```
public class FuelRecordHandler {
   public static void beforeDeleteInfo(list<Fuel_details__c> fuelList){
      //fuelList = [select Id from Fuel_details__c];
       for(Fuel_details__c ful : fuelList){
          if(ful.Fuel_supplied__c > 500){
             ful.addError('you cannot delete the fuel details record because it is associated with supplier and Gas station records');
         }
       }
  }
 public static void beforeDeleteGas(list<Gas_Station__c> gasList){
    //fuelList = [select Id from Fuel_details__c];
      for(Gas_Station__c gas : gasList){
           if(gas.Fuel_price_liter__c <= 50){
               gas.addError('enter the fuel price before saving the record, Minimum price should be 50');
           }
        }
    }
}
```

- Created **Triggers**:
  - ✓ On Fuel Details: before delete → calls handler to validate fuel quantity
  - ✓ On Gas Station: before insert → checks price constraint

File Edit Debug Test Workspace Help < >

FuelRecordHandler.apxc * | beforeDelete.apxt | beforeInsert.apxt

Code Coverage: None | API Version: 64

```
1  trigger beforeDelete on Fuel_details__c  (before Delete) {
2
3      if(trigger.isbefore && trigger.isDelete){
4
5          FuelRecordHandler.beforeDeleteInfo(trigger.old);
6
7      }
8
9  }
```

File Edit Debug Test Workspace Help < >

FuelRecordHandler.apxc * | beforeDelete.apxt | beforeInsert.apxt

Code Coverage: None | API Version: 64

```
1  trigger beforeInsert on Gas_Station__c (before insert ) {
2
3      if(trigger.isbefore && trigger.isinsert){
4
5              FuelRecordHandler.beforeDeleteGas(trigger.new);
6
7      }
8  }
```

8. Business Logic Enforcement

- Ensured that data integrity and business rules are enforced through:
  - ✓ Apex triggers for critical operations
  - ✓ Formula fields for auto-calculation
  - ✓ Roll-up summaries to aggregate fuel values
  - ✓ Flow for personalized communication

ADVANTAGES

1. Centralized Data Management

- **Unified view** of Suppliers, Gas Stations, Buyers, and Fuel Details in one system.
- Eliminates paperwork and manual tracking of fuel supply and customer data.

2. Real-Time Inventory Tracking

- Live monitoring of **fuel availability**, **fuel usage**, and **price fluctuations**.
- Automatic roll-up summaries for fuel supplied and consumed help prevent shortages.

3. Improved Operational Efficiency

- Automated **fuel booking**, **task creation**, and **approval processes** streamline daily operations.
- Triggers and flows reduce manual intervention and ensure smooth workflow.

4. Enhanced Security & Access Control

- Role hierarchy (Manager > Sales Executive > Sales Person) ensures **data visibility is restricted** based on user profiles.
- **OWD, Sharing Rules, Permission Sets** ensure controlled access to records.

5. Better Customer Engagement

- Maintains full **buyer profiles**, including vehicle type, fuel filled, and payment mode.
- Improves service with faster checkouts and accurate customer data.

6. Data-Driven Decision Making

- **Dashboards & Reports** like "Amount Range" provide visual insights into business metrics.
- Managers can track fuel demand, revenue, and sales patterns with ease.

7. Automated Compliance & Auditing

- **Field history tracking**, validation rules, and duplicate rules ensure data integrity.
- Helps maintain audit logs and data consistency for regulatory compliance.

8. Scalability & Customization

- Built on Salesforce, the system can scale with new stations, suppliers, and customer base.
- Easily customizable to add more objects, flows, and automation.

9. Modern & Intuitive UI

- Lightning App with **dynamic forms**, custom layouts, and responsive pages provides a user-friendly interface.
- Future-ready with optional **LWC components** for enhanced interactivity.

10. Easy Maintenance & Deployment

- **Change sets** for deployment and test classes for validation ensure safe updates.
- System is easy to maintain with proper documentation and DevOps practices
-

Conclusion

The **Gas Station CRM Application**, developed entirely on the Salesforce platform, has successfully transformed traditional gas filling station operations into a **digitized, automated, and customer-centric system**. By leveraging the robust features of Salesforce CRM — including **custom objects, validation rules, flows, Apex triggers, reports, and dashboards** — this solution ensures seamless management of **fuel bookings, inventory tracking, supplier coordination, and customer service**.

Key achievements of this project include:

- **Centralized Data Management:** All critical data related to fuel supply, buyers, and gas stations is stored and managed in a single, secure system.
- **Automation:** Record-triggered flows and Apex triggers minimize manual tasks, reduce errors, and enforce strict business rules (e.g., fuel price validations and deletion restrictions).
- **Advanced Reporting & Dashboards:** Real-time insights into fuel availability, payment trends, and customer transactions help managers make informed business decisions.
- **Role-Based Access & Security:** Profiles, roles, and permission sets ensure data is accessed only by authorized users, maintaining **data integrity and security**.
- **Scalable Design:** The data model and security configurations are designed to handle future expansion, including more stations, suppliers, or advanced features.

This CRM solution is not just a digital tool but a **business accelerator**, improving efficiency, reducing operational overhead, and enhancing the overall **customer experience** at the gas station.