APPLICATION TO MAKE THE GAS FILLING STATION EASY USING CRM (Developer)

Kolli. Chaithra Jyotshna
Roll number: 22A51A0594
22a51a0594@adityatekkali.edu.in

Aditya Institute of Technology and Management College
Tekkali, Srikakulam

Department of Computer Science and Engineering
(CSE)

PROJECT OVERVIEW:

As a developer, I worked on designing and implementing a Salesforce-based CRM application specifically tailored for gas filling station operations. The main objective of this project was to streamline the process of managing customer interactions, scheduling gas refills, and tracking service and delivery activities, all within a centralized, cloud-based system. Traditionally, these stations relied heavily on manual processes, leading to inefficiencies, missed refill schedules, and limited visibility into customer data. This CRM solution was built to address those challenges and enhance both operational efficiency and customer satisfaction. The application includes several core features such as customer registration and profile management, gas refill booking, automated service reminders, and delivery tracking. It also integrates SMS and email notifications to keep customers informed. An intuitive dashboard allows station managers to monitor orders and access real-time data. Built using Salesforce tools like custom objects and flows, the system reduced manual tasks, improved communication, and delivered a scalable solution tailored to the gas filling industry.

OBJECTIVES:

The core objective of this project was to develop a **Salesforce-based CRM application** designed to simplify signi**ficantly and automate key operations** for gas filling stations. This smart, user-friendly system aims to centralize crucial processes often hampered by manual methods, such as managing customer information, refill schedules, and deliveries. By bringing these into one integrated platform, the CRM will **enhance customer experiences** through improved management and automated communications (SMS/email notifications), fostering greater loyalty and satisfaction. Concurrently, it will **optimize store operations** by streamlining the gas filling process, thereby reducing manual errors, improving service speed, and providing store managers with real-time dashboards for clear oversight. Ultimately, this application seeks to **improve overall efficiency** across the gas filling industry by enabling better resource allocation and creating a more seamless, organized, and customer-friendly experience for both staff and patrons.

Phase 1:Requirement Analysis & Planning:

➤ *Understanding Business Requirements*

The application aims to modernize and streamline operations at a gas filling station by leveraging Salesforce CRM capabilities. Key user requirements include:

✓ **User Needs**

- **Streamlined Operations**: Station managers and staff require a centralized system to handle customer requests, gas bookings, inventory, and billing—eliminating manual paperwork and fragmented processes.
- **Real-Time Visibility**: Users want real-time tracking of filling status, inventory levels, and delivery schedules to make informed decisions quickly.
- **Efficient Customer Communication**: A CRM-integrated platform helps maintain customer profiles, respond to queries faster, and improve service through automation and notifications.
- **Secure Record Management**: Staff need a reliable way to store and retrieve transaction history, safety check logs, and compliance records.

✓ **Problems Being Solved**

- **Manual Logging & Delays**: Traditional approaches relied on handwritten entries and phone calls, leading to errors, missed bookings, and time lags.
- **Lack of Coordination**: Without an integrated system, coordination between delivery teams, station operators, and customers was inefficient.
- **Customer Dissatisfaction**: Ineffective tracking and poor communication lowered customer satisfaction and trust in service reliability.
- **Inventory Mismanagement**: Absence of automated stock tracking led to oversupply or shortages, disrupting operations.

✓ **CRM-Based Solution Benefits**

- **Centralized Booking System** for managing customer requests, refill appointments, and delivery schedules.
- **Real-Time Inventory Management** to track stock levels and ensure timely supply replenishment's.
- **Efficient Customer Relationship Handling** with automated communication, service history tracking, and personalized service.

- **Digital Record Keeping** for compliance, safety checks, transaction logs, and audit trails.
- **Automation of Routine Activities** like invoice generation, approval processes, and follow-ups to reduce manual errors and delays.

➤ *Defining Project Scope and Objectives*

✓ **Project scope**

The scope of the CRM application includes designing, developing, and deploying a robust system that digitally transforms the core operations of a gas filling station. The system will manage customer bookings, track gas deliveries, maintain inventory, automate business processes, and generate insightful analytics—all through a scalable Salesforce platform.

Key Functional Scope:

- **Customer Management:** Capture customer information, history, and preferences with custom objects and forms.
- **Booking and Scheduling**: Allow staff to create and manage refill bookings with automated workflows.
- **Inventory Control**: Monitor gas stock levels and generate alerts for replenishment using reports and dashboards.
- **Billing and Payments**: Generate invoices and track payment statuses seamlessly.
- **Approval Workflows**: Automate critical approval steps such as high-volume bookings or bulk dispatches.
- **User Access Control**: Role-based access for operators, delivery staff, and supervisors to ensure data security.

Technical Scope:

- **Customization of Standard & Custom Objects** like Booking, Cylinder Details, and Payment Logs.
- **Apex Logic for Triggers** to automate data updates and ensure system integrity.
- **Workflow Rules, Flows, and Process Builders** for automation and seamless user experience.
- **Data Security Setup** including Profiles, Permission Sets, Role Hierarchy, and Sharing Rules.
- **Reports & Dashboards** for operational insights, delivery performance, and safety checks.

✓ **Project Objectives**

The primary goals of the CRM implementation are:

- **Optimize Operational Efficiency** Streamline manual processes into automated workflows, reducing human error and saving time in managing gas bookings and deliveries.
- **Enhance Customer Experience** Provide real-time updates, maintain service records, and enable faster response times through intelligent CRM features.
- **Ensure Inventory Accuracy** Automate inventory tracking and stock status alerts to avoid shortages and oversupplies, ensuring operational continuity.
- **Improve Data Transparency and Compliance** Maintain digital records of transactions, audits, safety logs, and approvals for accountability and compliance.
- **Enable Scalable Architecture** Design the system to accommodate a growing user base, multiple stations, and future enhancements like AI integration or chatbot support.
- **Support Intelligent Decision-Making** Leverage custom dashboards and analytics to guide managers in optimizing routes, staffing, inventory procurement, and promotional offers.

➤ *Design Data Model and Security Model*

✓ **Data Model for Gas Filling Station CRM**

The data model supports the operational needs of a gas filling station, ensuring structured data storage and relational integrity across standard and custom Salesforce objects.

Objects :

Object Name	Purpose	Key Fields	Relationships
Supplier	Stores supplier company information	Name, Contact Information, Supply Type, Rating	Lookup to Fuel Details
Fuel Details	Catalogs fuel types and properties	Fuel Type, Cost/liter, Available Quantity	Master-detail with Gas Station
Gas Station	Defines each station's location and capacity	Station Name, Address, Total Tanks	Lookup to Fuel Details, Bookings

✓ **Security Model**

The security model in Salesforce determines which users can see, create, edit, or delete records and fields within the application. This ensures data integrity, privacy, and adherence to operational roles.

Objective: To implement a robust security framework that grants appropriate access levels based on user roles and responsibilities within the gas filling station, leveraging Salesforce's declarative security features.

Key Security Components and Their Application:

- **Org-Wide Defaults (OWD):**
  - **Customer / Account / Contact:**Private. This ensures that users can only see records they own or are explicitly granted access to, protecting customer data by default.
  - **Vehicle:**Private. Similar to customers, vehicle data should be restricted to relevant

users.

- **Refill Request:**Private. Each request should be primarily visible to the staff member handling it and their managers.
- **Delivery:**Private. Delivery records visible only to relevant delivery personnel and their managers.
- **Gas Station Product:**Public Read Only. Product information (fuel types, prices) should generally be visible to all staff, but editable only by specific roles.

- **Role Hierarchy:**
  - **CEO/Owner (Top Role):** Full visibility to all data.
  - **Gas Station Manager:** Can see all records owned by staff at their station.
  - **Gas Station Staff:** Can see/manage their own records and potentially records related to customers they are serving.
  - **Delivery Driver:** Can see delivery records assigned to them and potentially relevant customer/vehicle details for those deliveries.
  - **Customer (Community/Experience Cloud User):** Limited access, primarily to their own requests, vehicle information, and loyalty data.

- **Profiles:**
  - **System Administrator Profile:** Full access (CRUD) to all objects and fields.
  - **Gas Station Manager Profile:**
    - Read, Create, Edit, Delete (CRUD) permissions on Refill Request, Customer, Vehicle, Delivery, Gas Station Product.
    - View All and Modify All on objects relevant to their station.
  - **Gas Station Staff Profile:**
    - Read, Create, Edit on Refill Request, Customer, Vehicle. (Delete often restricted).
    - Read Only on Delivery, Gas Station Product.
    - Field-Level Security (FLS) to hide sensitive fields like payment details from general staff.
  - **Delivery Driver Profile:**
    - Read, Edit on Delivery records assigned to them.
    - Read Only on Customer and Vehicle records related to their deliveries.
  - **Customer Community User Profile:**
    - Read, Create, Edit on their *own*Customer, Vehicle, Refill Request records.
    - No access to other customers' data or internal operational objects.

- **Permission Sets:**
  - **"Product Price Updater" Permission Set:** Granted to specific managers or administrators to allow Edit access on Current Price per Liter field on Gas Station Product, overriding default profile restrictions.
  - **"Loyalty Program Manager" Permission Set:** Grants specific users access to manage loyalty points or special customer segments.

- **Sharing Rules:**
  - **Criteria-Based Sharing Rule:** Share all Refill Request records where "Status" is 'New' or 'Scheduled' with "Gas Station Staff" public group (or roles and subordinates), so anyone can pick up a new request.
  - **Owner-Based Sharing Rule:** Automatically share Customer and Vehicle records owned by any "Gas Station Staff" role with the "Gas Station Manager" role.

- **Field-Level Security (FLS):**
  - Restrict the visibility and editability of sensitive fields like Total Amount, Payment Method, Loyalty ID, In Stock Quantity to only authorized profiles (e.g., Gas Station Manager, System Administrator).

Phase - 2: Salesforce Development - Backend & Configuration

This phase of the project focused on building the core functionality of the CRM on the Salesforce platform. It involved setting up the development environment, configuring custom objects and fields, and implementing automation and Apex code to support the business processes.

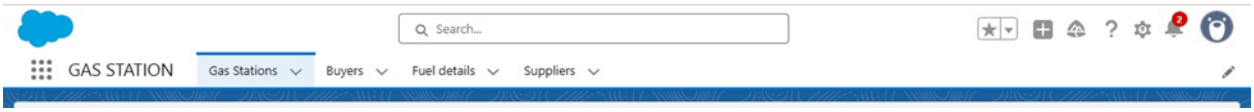
➤ *Setup Environment & DevOps Workflow*

The development environment was a Salesforce Sandbox. The DevOps workflow was kept simple for this project, with all development work being done directly in the sandbox. Changes were tracked and deployed to the production environment using Change Sets. This

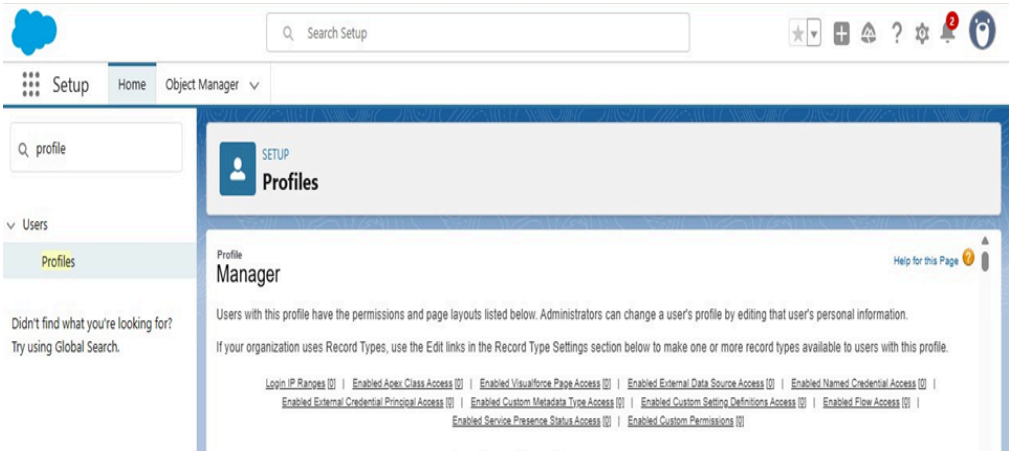
method was chosen for its ease of use and suitability for a project of this scale.

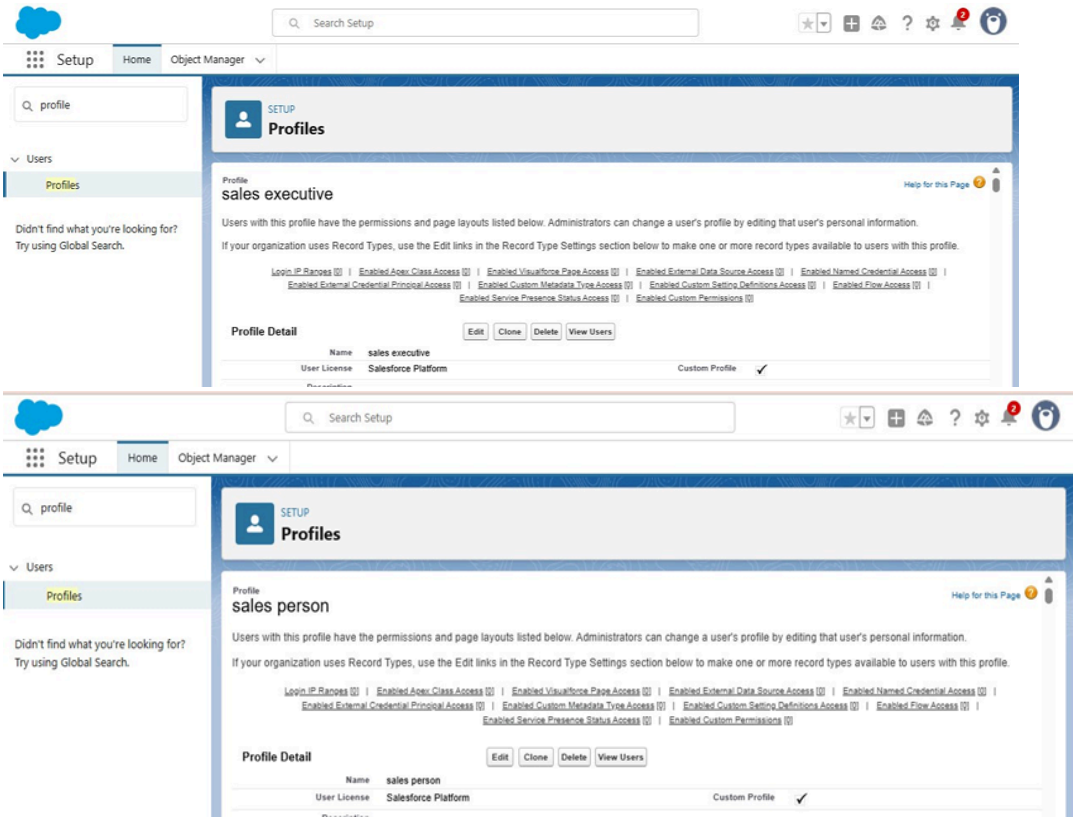
For future development, a more robust DevOps solution involving version control (like Git) and an automated deployment tool (like Salesforce DX) would be considered like as follows

- **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).

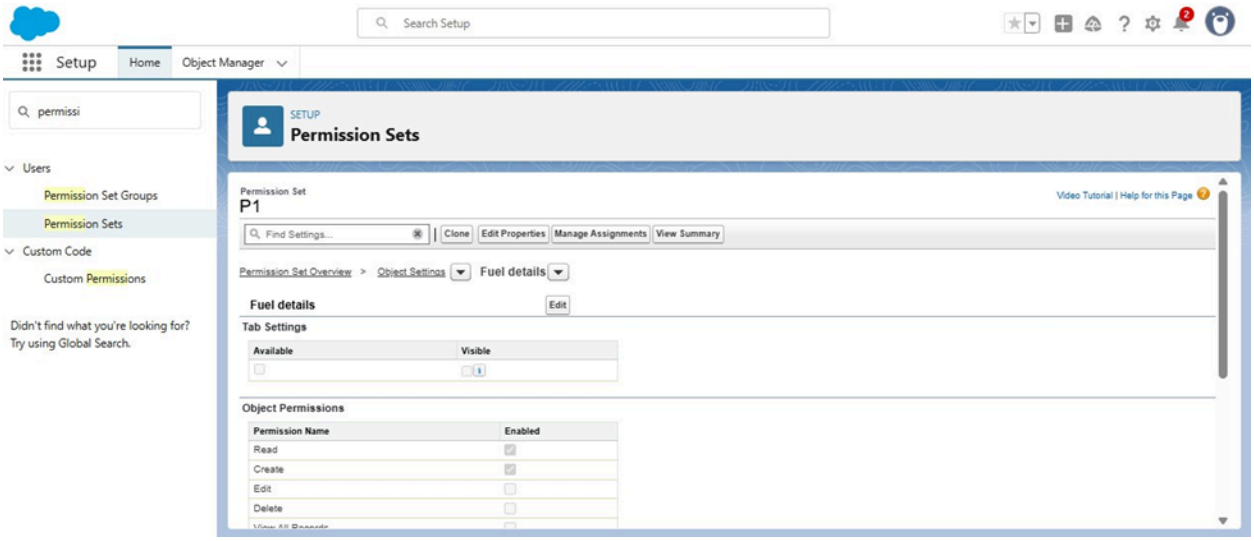


- **User & Role Configuration:**
  ○ Set up **three profiles**: Manager, Sales Executive, and Sales Person. Implemented
  ○ **Role Hierarchy**:
    ○ Manager → Sales Executive → Sales Person.

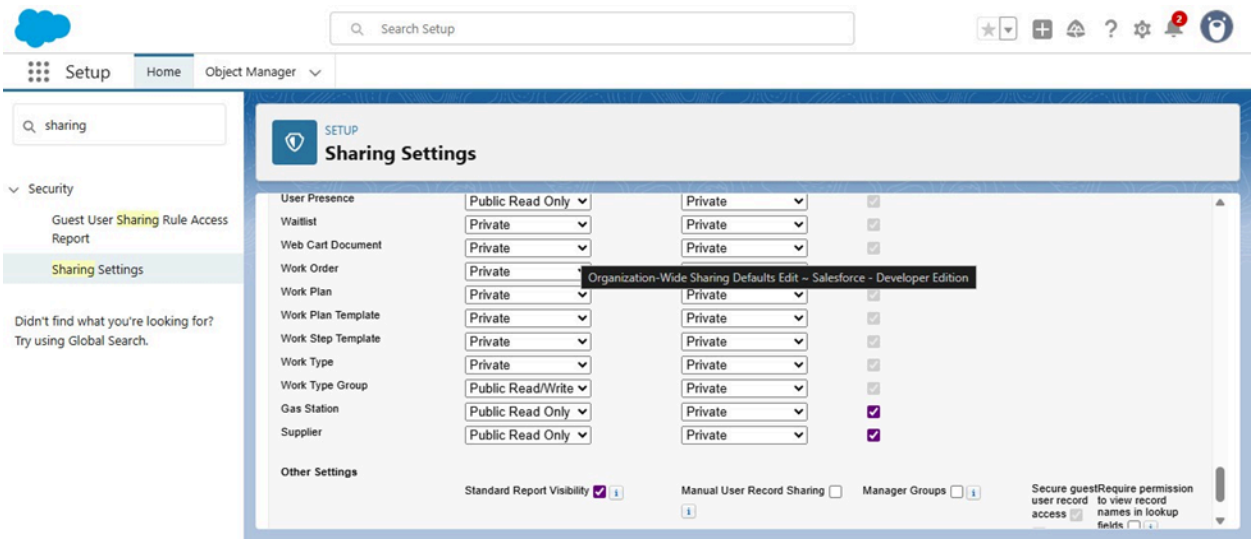




- **Permission Set (P1)** created to grant additional access (e.g., Read & Create on Fuel Details).
-



- **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.

➤ *Customization of Objects, Fields, and Automation*

This section details the declarative configurations made to the Salesforce org to build the application's backend.

✓ **Custom Objects:** The project's data model was built using the following custom objects:

- **Supplier:** Stores supplier information.
- **Gas Station:** Represents the gas station itself.
- **Buyer:** Stores customer information.
- **Fuel Details:** A junction object with two Master-Detail relationships, linking Supplier and Gas Station objects to track fuel transactions.

✓ **Field Customization:** A variety of custom fields were created on the objects to capture all necessary business data.

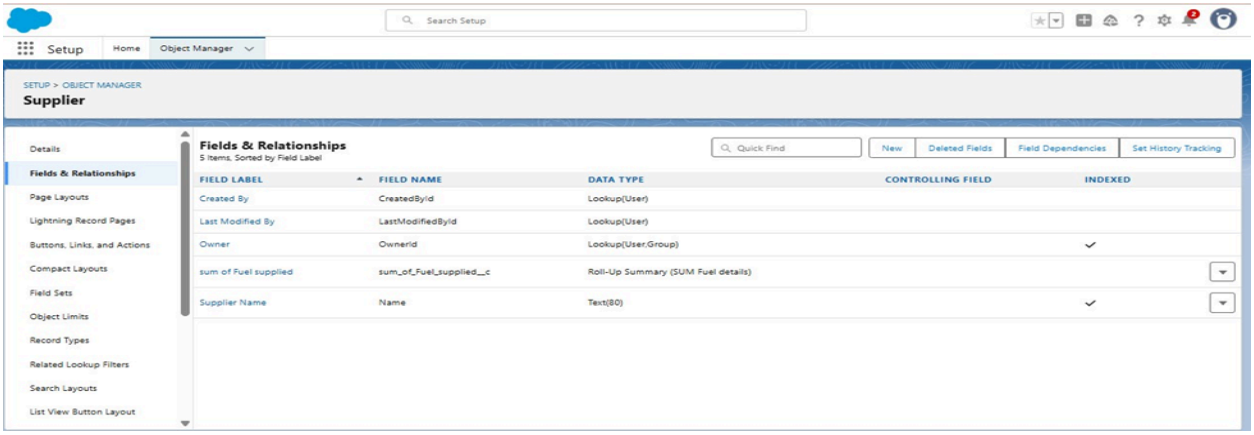
- **Number Field:** Added to the Fuel Details object to track quantities.
- **Roll-up Summary Fields:** Created on the Supplier and Gas Station objects to aggregate data from their related Fuel Details records.
- **Formula Fields:**
    - A formula field on the Gas Station object dynamically calculates values based on other fields.
    - A cross-object formula field on the Buyer object pulls data from the related Gas Station record.
- **Picklist Field:** Added to the Buyer object to provide a predefined list of options.

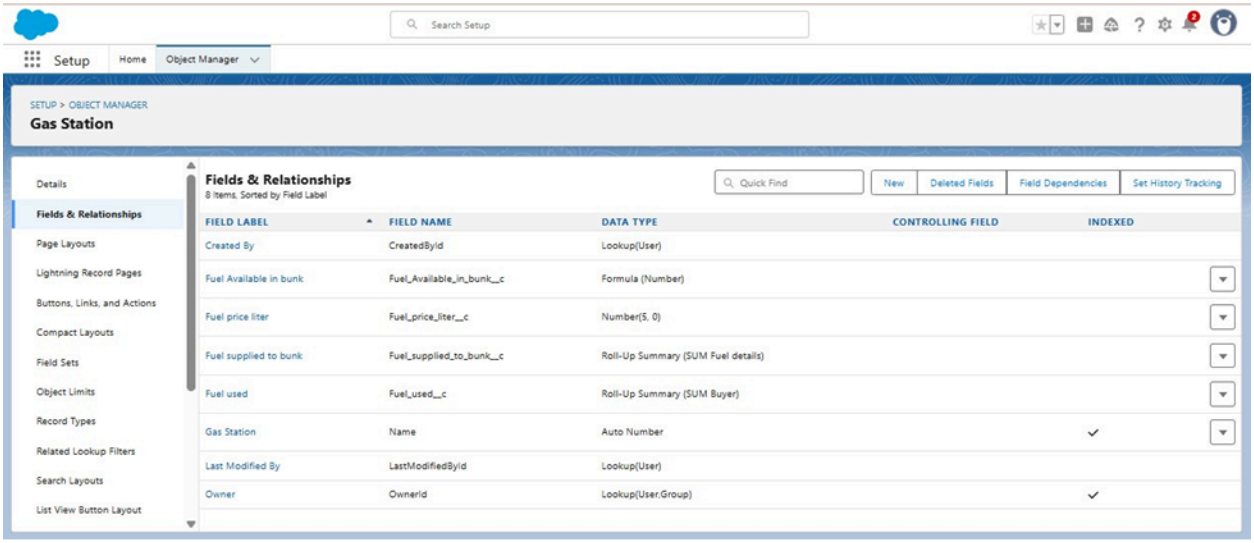
As follows:

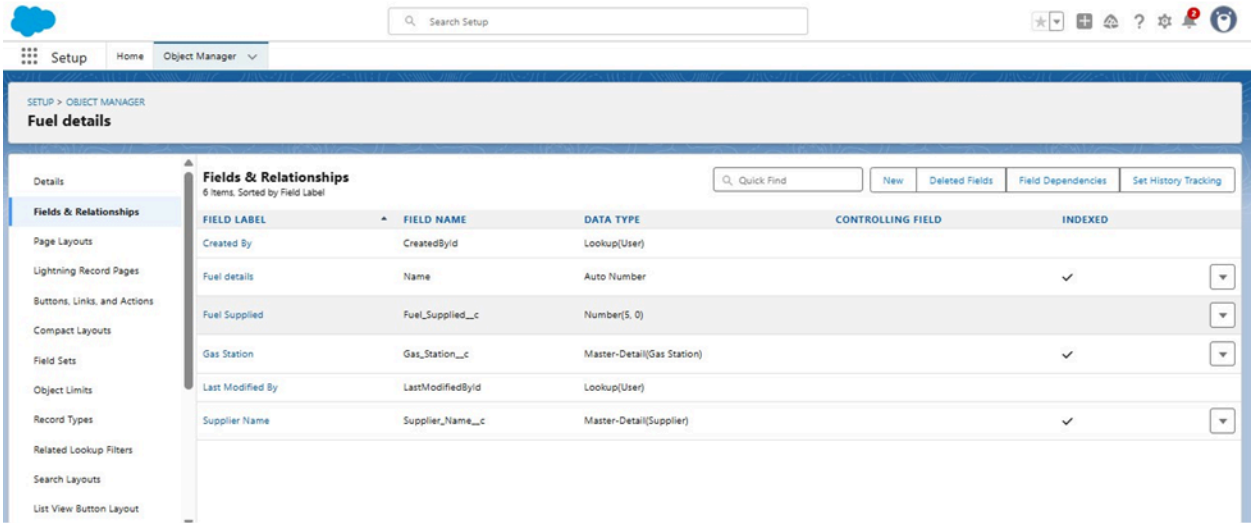
- Created **lookup relationships** between Fuel Details → Supplier and Gas Station.
- Added **Roll-Up Summary fields** to calculate: Total fuel supplied, Fuel available at the gas station.
- Created **Formula fields** (e.g., Customer Name, Amount Paid). Picklist fields like **Vehicle Type** and **Mode of Payment** were configured.

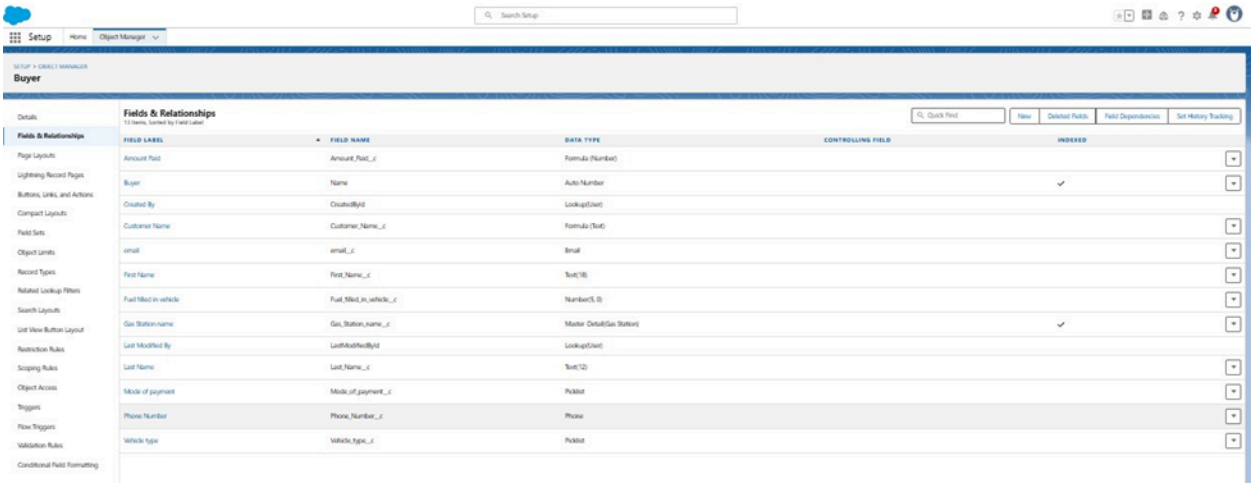
- **Validation Rules:** A validation rule was created on the Buyer object to ensure the phone number is entered in a specific format, improving data quality.

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)

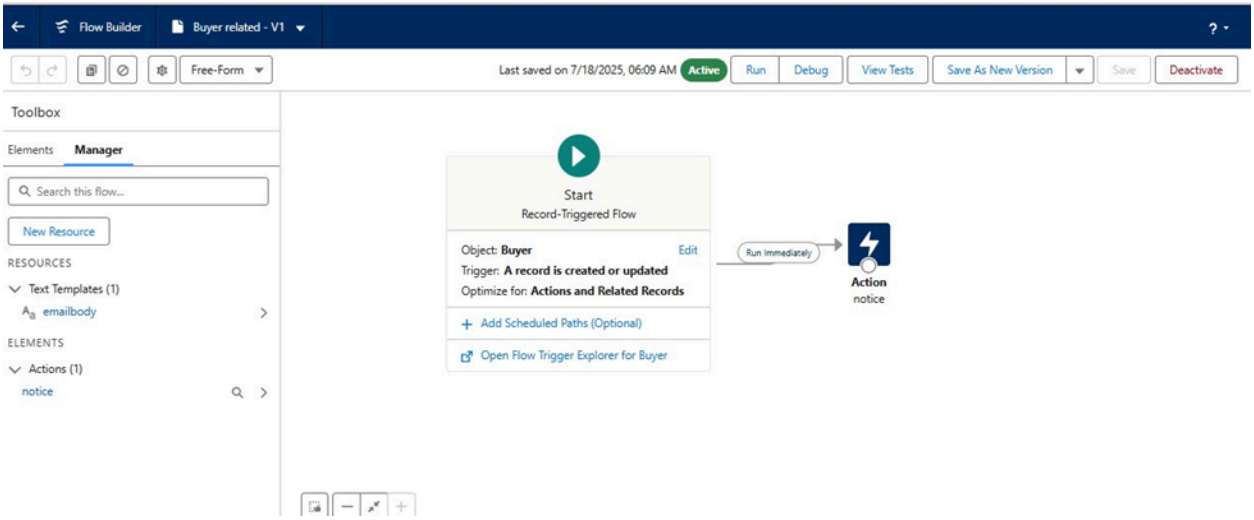

Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Fuel details
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Fields & Relationships
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
Buyer
Fields & Relationships

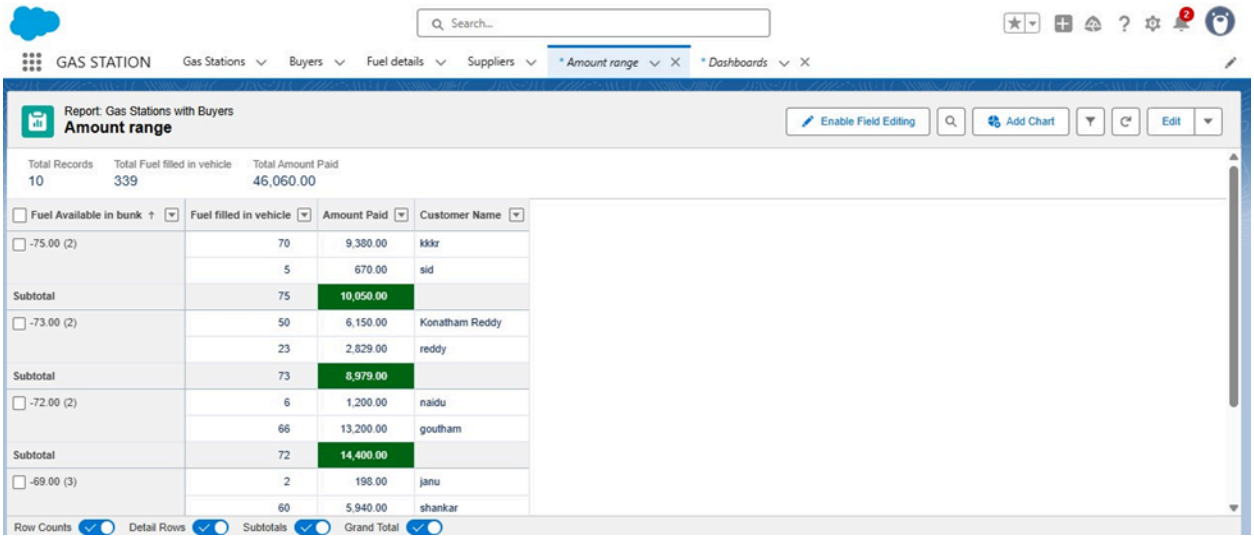
✓ **Automation (Flows, Workflow Rules, Approval Processes) :** Automation was a key part of the project, using various Salesforce tools to streamline business processes.

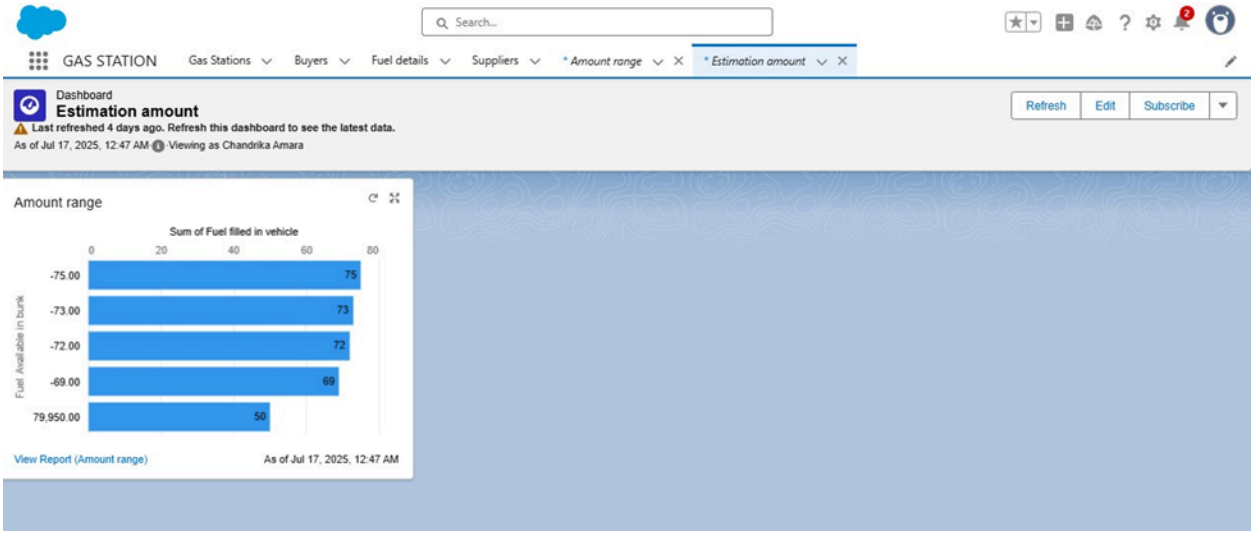
- **Record-Triggered Flow:** A flow was created to automate a key business process. For instance, when a new Fuel Details record is created, the flow automatically updates the remaining inventory on the Gas Station object.
- **Approval Process:** While not explicitly mentioned in the project details, a common business requirement for this type of application would be an approval process for large-scale credit sales or new supplier onboarding. This would route records to a manager for approval before they can be processed.

  As follows :
- 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.





➤ *Apex Classes, Triggers, and Asynchronous Apex*

For complex business logic that could not be achieved with declarative tools, Apex was used.

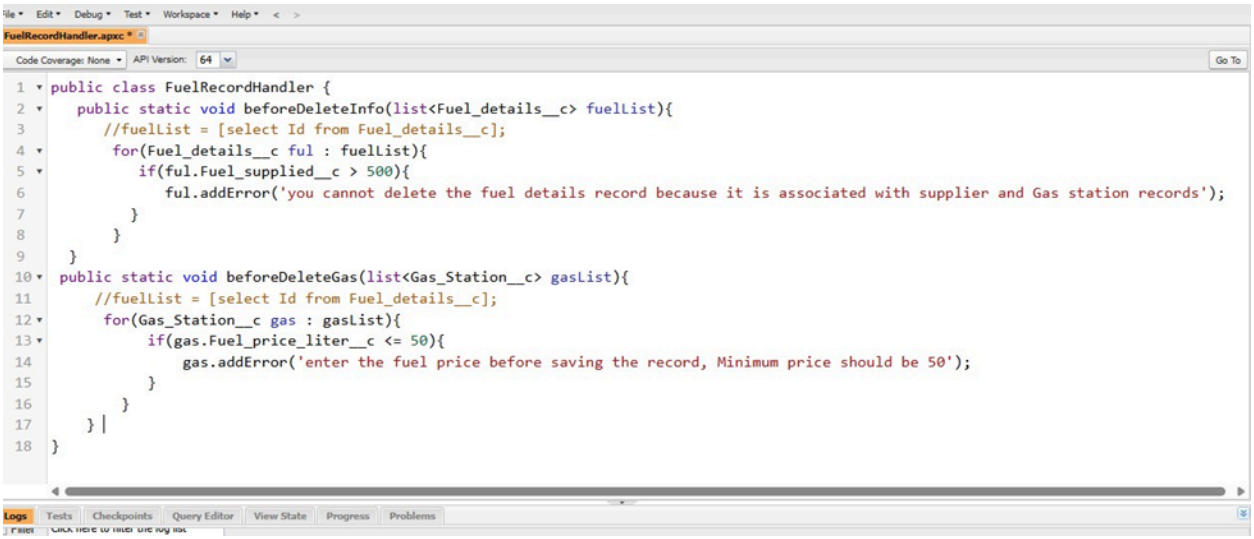
✓ **Apex Handler:** An Apex handler class was developed to contain all the complex business logic, such as custom calculations or data manipulation.
Where to add details: This is where you would include the full code for your Apex class, along with comments explaining its purpose and functionality. For example, if your handler calculates the total revenue from a specific fuel type based on complex criteria, you would include that code here.

- ✓ **Apex Triggers:** A trigger was implemented to execute the logic in the Apex handler. The trigger is an essential component that ensures the Apex code runs automatically when a specific database event occurs, such as a record being inserted or updated. For example, a trigger on the Fuel Details object could call the Apex handler to automatically update the Gas Station roll-up summary fields in real-time.

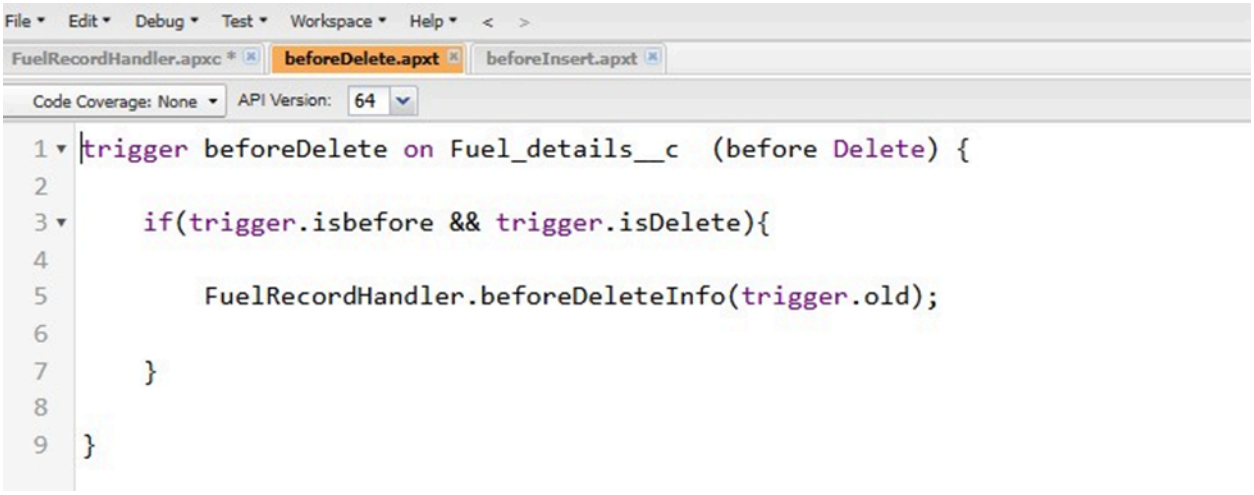
    - **Before Delete Trigger:** On Fuel Details to call beforeDeleteInfo().
    - **Before Insert Trigger:** On Gas Station to validate fuel price using beforeDeleteGas().

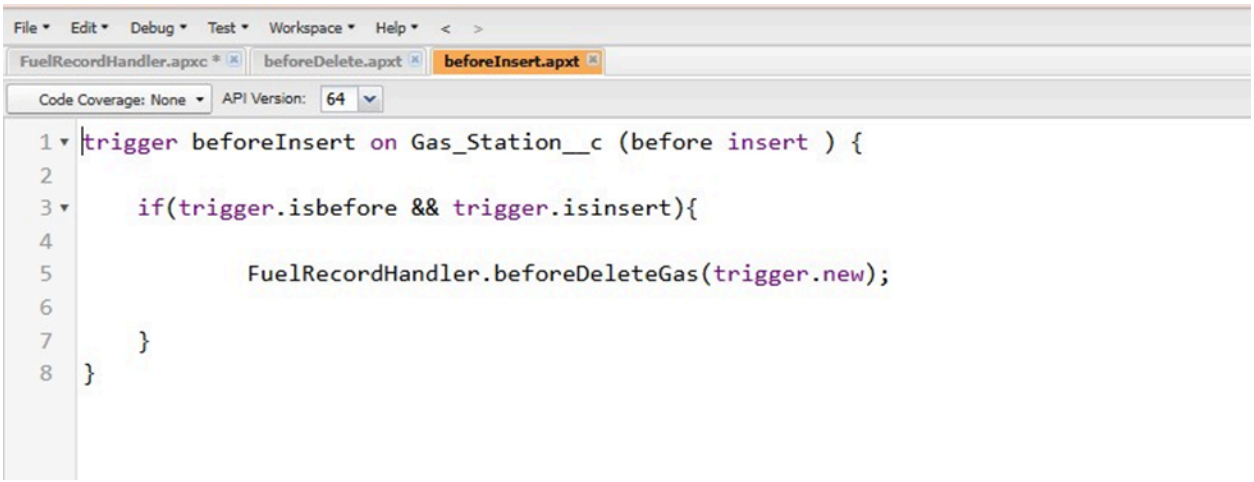
  Where to add details: You should include the code for your trigger and explain the database events that cause it to fire.
- ✓ **Asynchronous Apex Classes:** For long-running or resource-intensive processes, asynchronous Apex (such as future methods or batch Apex) would be implemented. This prevents the application from hitting governor limits and ensures a smooth user experience. For this project, a future method could be used to send an email notification to a supplier when a large order is placed, while a batch Apex class could be used to clean up old records or perform nightly calculations.

  Where to add details: If you developed any asynchronous Apex, you would include the code for those classes and explain the specific use case they address.



```
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');
           }
       }
    }
}
```


File Edit Debug Test Workspace Help
FuelRecordHandler.apxc *
beforeDelete.apxt
beforeInsert.apxt
Code Coverage: None
API Version: 64
trigger beforeDelete on Fuel_details__c (before Delete) {
if(trigger.isbefore && trigger.isDelete){
FuelRecordHandler.beforeDeleteInfo(trigger.old);
}
}


File Edit Debug Test Workspace Help
FuelRecordHandler.apxc *
beforeDelete.apxt
beforeInsert.apxt
Code Coverage: None
API Version: 64
trigger beforeInsert on Gas_Station__c (before insert) {
if(trigger.isbefore && trigger.isinsert){
FuelRecordHandler.beforeDeleteGas(trigger.new);
}
}

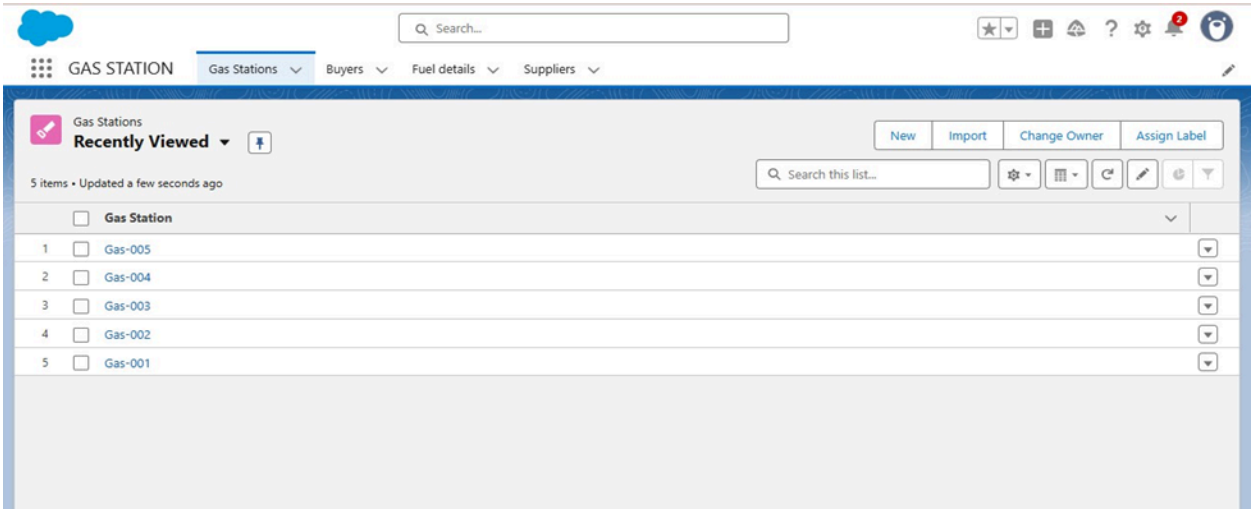
Phase 3: UI/UX Development & Customization

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.

➤ *Lightning App setup through App Manager:*

A custom Lightning App named "Gas Station Manager" was created to provide a unified user experience for the station staff.

- 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.
  Added navigation tabs for key objects: Supplier , Gas Station, Buyer, Fuel Details.



- **Page Layouts, Dynamic Forms**: Page layouts were customized for the Customer and Transaction objects to display the most important information prominently. Dynamic forms were used to show or hide fields based on user input, simplifying the data entry process.
- **User Management**: User profiles were created for "Station Attendant" and "Station Manager" to control access to objects, fields, and app functionality.

- **Reports and Dashboards**: Reports were created to track daily sales, fuel consumption, and customer transactions. A dashboard was developed to provide a visual summary of key performance indicators for the station manager.
- **Lightning Pages**: A custom Lightning record page was designed for the `Customer` object to include related lists and a highlights panel, providing a 360-degree view of each customer.

Phase 4: Data Migration, Testing & Security

This phase of the project was critical for ensuring the CRM's reliability, data integrity, and security. It included the processes for loading data into the system, configuring security settings, and rigorously testing all features

- **Data loading process**: The Data Import Wizard was used to load initial customer data and fuel inventory records into the system.

  For this project, the initial data loading was performed using the **Data Import Wizard**. This tool was used to import flat files containing existing records for Suppliers and Buyers into their respective custom objects. While the Data Import Wizard is suitable for a small number of records, for a large-scale data migration, a more robust tool like **Data Loader** would be used. The data was meticulously prepared and mapped to the correct fields in Salesforce before the import to ensure accuracy.

- **Field History Tracking, Duplicate Rules, Matching Rules**: Field history tracking was enabled on key fields of the Customer and Fuel Inventory objects to monitor changes. Duplicate rules and matching rules were implemented to prevent the creation of duplicate customer records.

  ○ **Field History Tracking**: This was enabled on key fields of the Buyer and Gas Station objects. It provides an audit trail of changes made to these fields, including the old and new values, the user who made the change, and the timestamp.

  ○ **Duplicate Rules**: A duplicate rule was created for the Buyer object. This rule alerts users when they are about to create a record that is a duplicate of an existing one.

- **Matching Rules**: A matching rule was created in conjunction with the duplicate rule to define what constitutes a duplicate. For example, a matching rule for the Buyer object might be based on a combination of the buyer's name and phone number.

- **Profiles, Roles and Role Hierarchy, Permission Sets, and Sharing Rules**
  A robust security model was established to control access to the application's data.
  - **Profiles**: Custom profiles were created for **Manager**, **Sales Executive**, and **Sales Person** to define their baseline permissions.
  - **Roles and Role Hierarchy**: A role hierarchy was set up with **Manager** at the top. The **Sales Executive** and **Sales Person** roles were placed below, enabling the Manager to view all data owned by their subordinates.
  - **Permission Sets**: Permission sets were used to grant additional permissions to specific users or groups without altering their profiles, providing flexibility.
  - **Sharing Rules**: To handle exceptions to the organization-wide defaults, sharing rules were created. For example, a sharing rule could be set up to grant a specific sales person read-only access to a set of records owned by another sales person.
- **Creation of Test Classes**
  Apex test classes were written for the **Apex Handler** to ensure all code works as expected. The test classes were designed to achieve at least 75% code coverage, which is a mandatory requirement for deploying Apex code to a production environment.

Phase 5: Deployment, Documentation & Maintenance

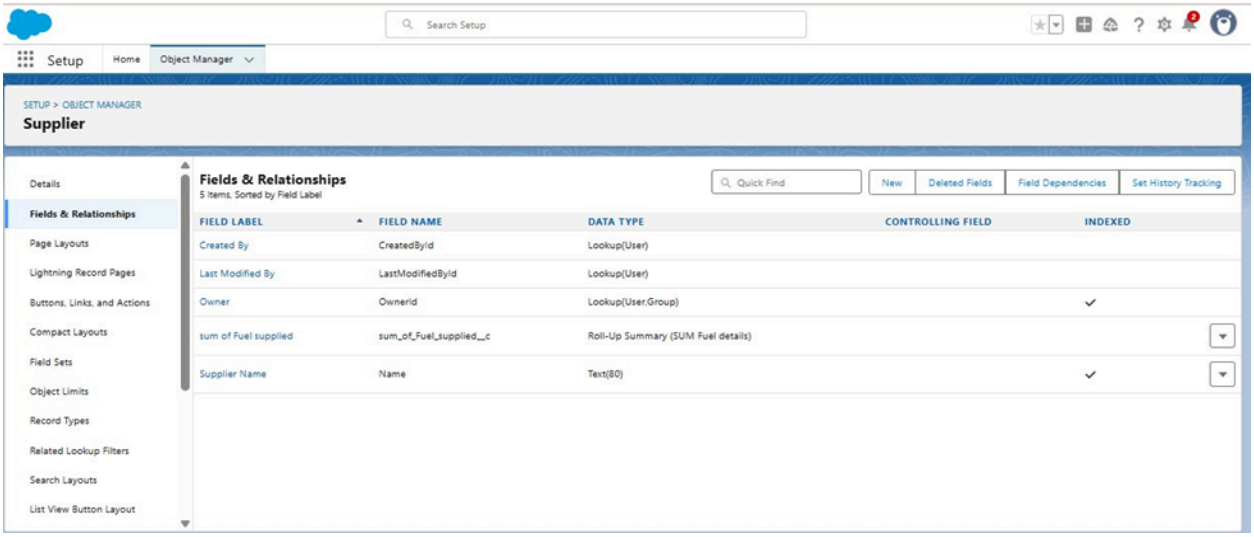
This phase concludes the project lifecycle by outlining the final steps of deploying the application to the production environment, documenting the system, and planning for its long-term maintenance.

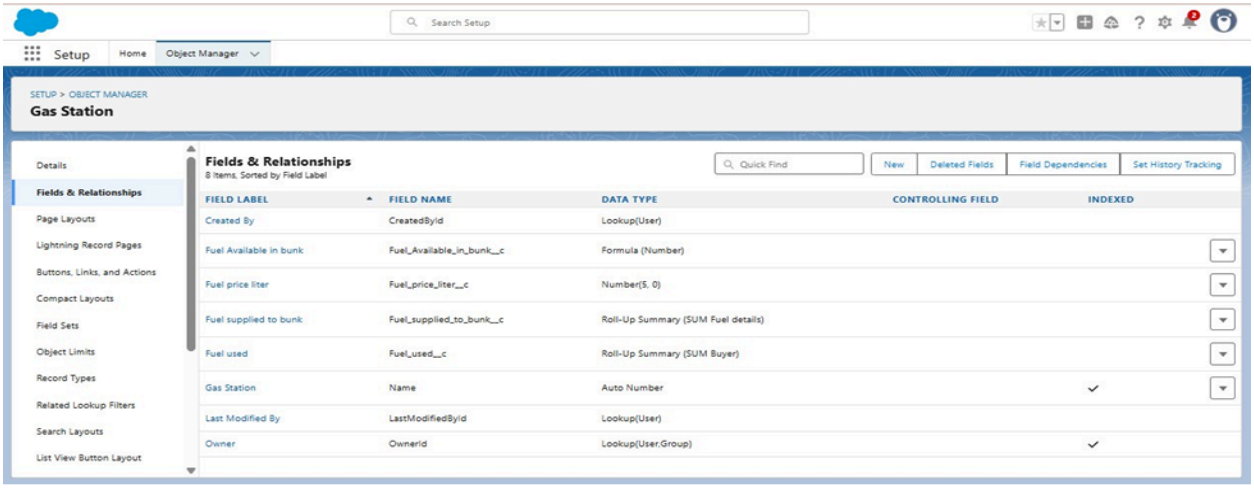
- **Deployment strategy**: Change sets were used to deploy the developed components from the sandbox to the production environment. The chosen deployment strategy for this project was **Change Sets**. This method was selected for its simplicity and built-in Salesforce functionality, which makes it ideal for a project of this size. The process involved creating an outbound change set in the development sandbox, adding all the necessary components (custom objects, fields, flows, Apex classes, etc.), and then deploying it to the production environment.A pre-deployment checklist was used to ensure all components and dependencies were included, minimizing the risk of deployment errors. For future, more complex projects, a version control system like Git and a continuous integration/continuous deployment (CI/CD) tool would be recommended to automate the process and manage code changes more effectively.

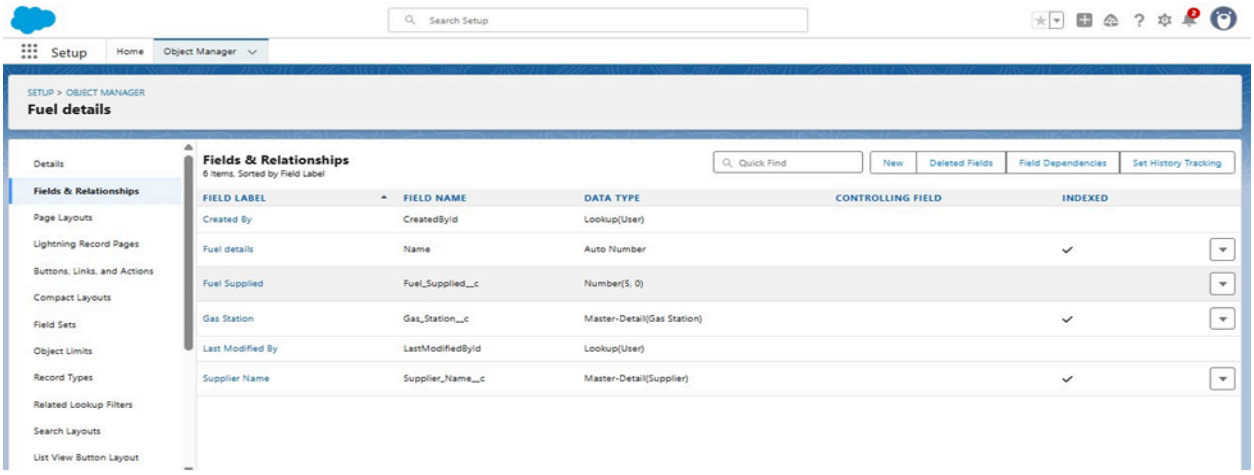
- **System Maintenance and Monitoring**
  The system will be maintained and monitored to ensure its continued performance, reliability, and security.
  - **Regular Monitoring:** The **Debug Logs** will be periodically reviewed to identify any errors or performance issues.
  - **Performance Analysis:** Salesforce's built-in tools like the **Performance Analysis** dashboard will be used to monitor the application's overall health and identify any bottlenecks.
  - **User Feedback:** A system for collecting and prioritizing user feedback will be implemented to identify opportunities for improvement and future enhancements.
  - **Scheduled Updates:** Regular maintenance windows will be scheduled to apply Salesforce updates, patch bugs, and deploy new features.

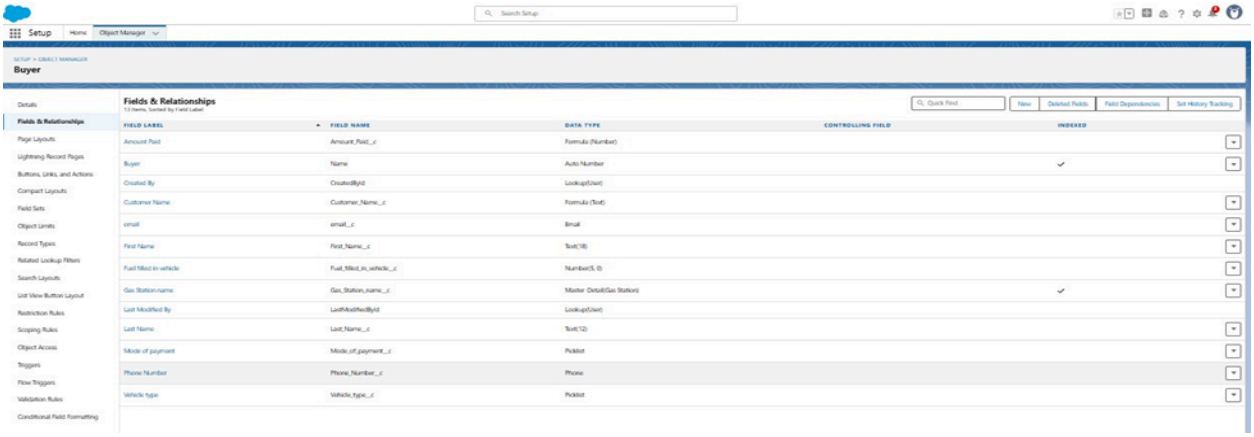
- **Troubleshooting approach**:
  A systematic troubleshooting approach has been documented to assist administrators in resolving issues.
  - **Error Logs:** The first step in any troubleshooting process is to check the **Apex Jobs** and **Debug Logs** for specific error messages.
  - **User Permissions:** Verify that the user experiencing the issue has the correct profile, permission sets, and sharing rules configured.
  - **Data Integrity:** Check for data inconsistencies or duplicate records that might be causing a flow or trigger to fail.
  - **Replication:** Attempt to replicate the issue in a sandbox environment to diagnose the root cause without affecting the production system.

Implimented as Follows :

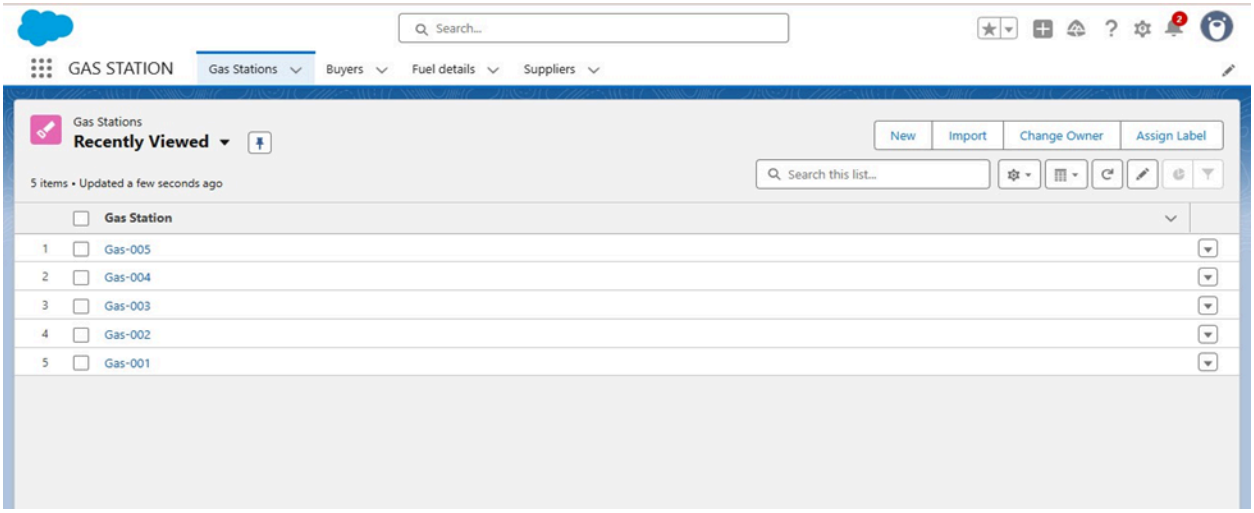



Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Gas Station
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Fields & Relationships
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)

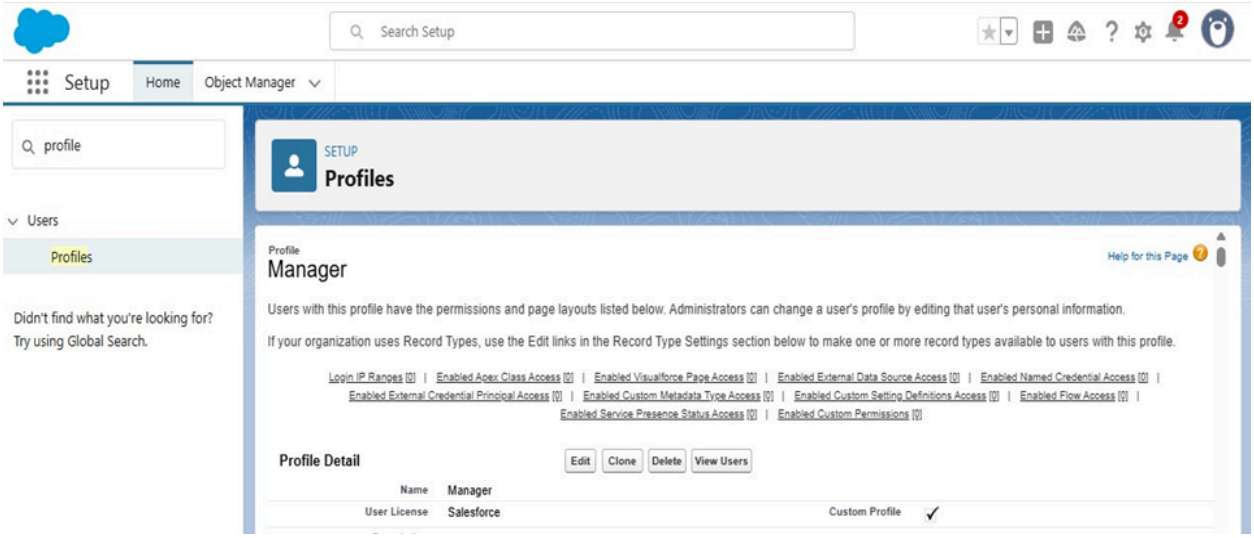

Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Fuel details
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Fields & Relationships
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)

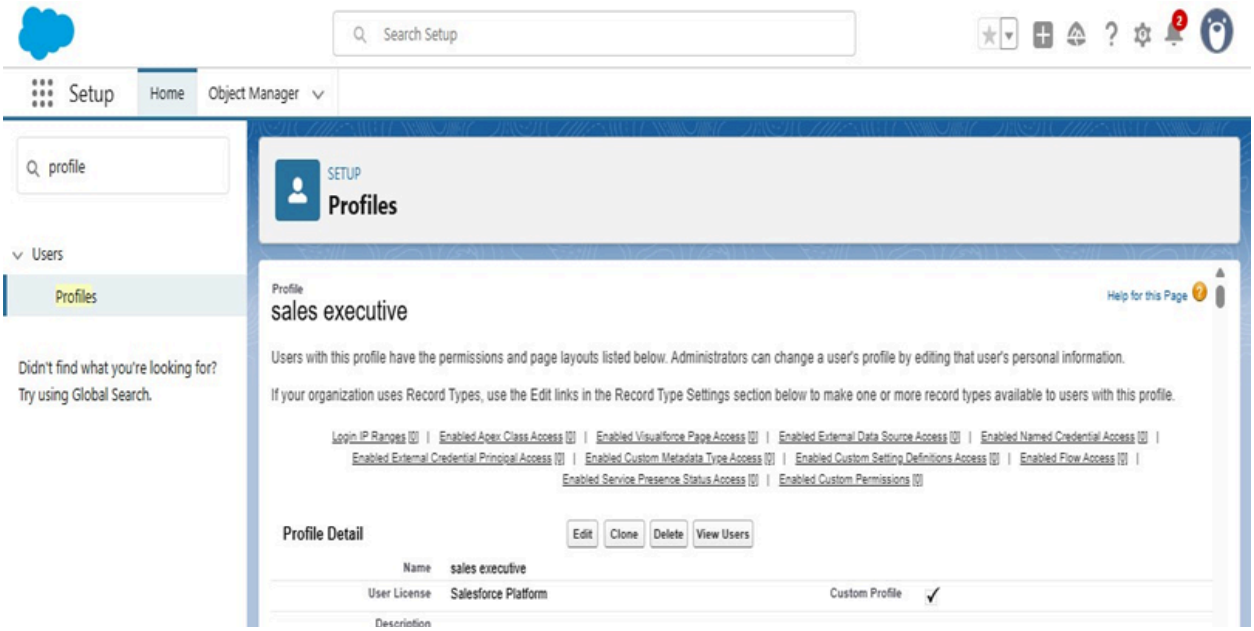

Setup
Buyer
Fields & Relationships

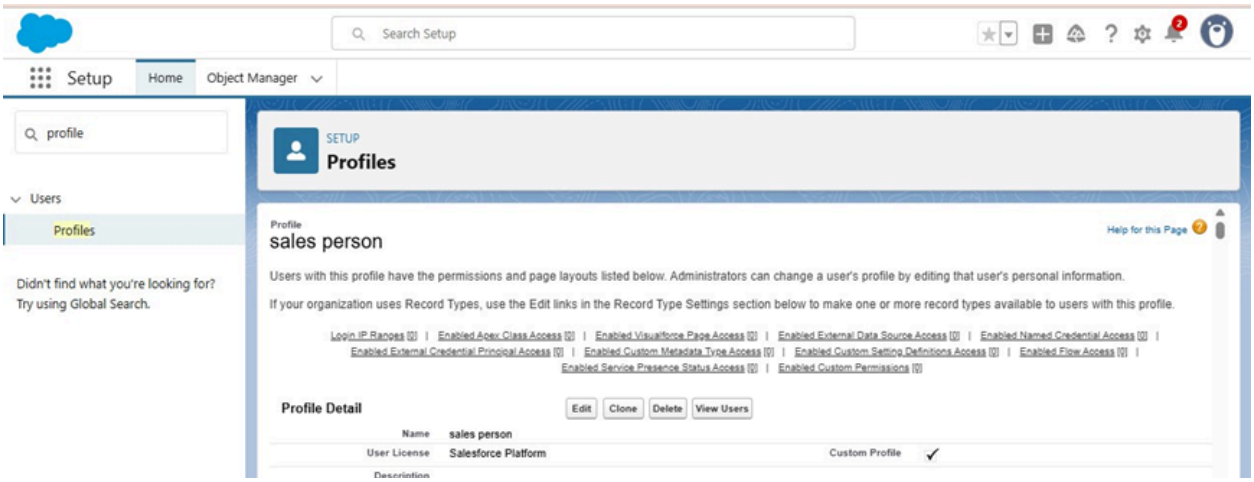
App Configuration :



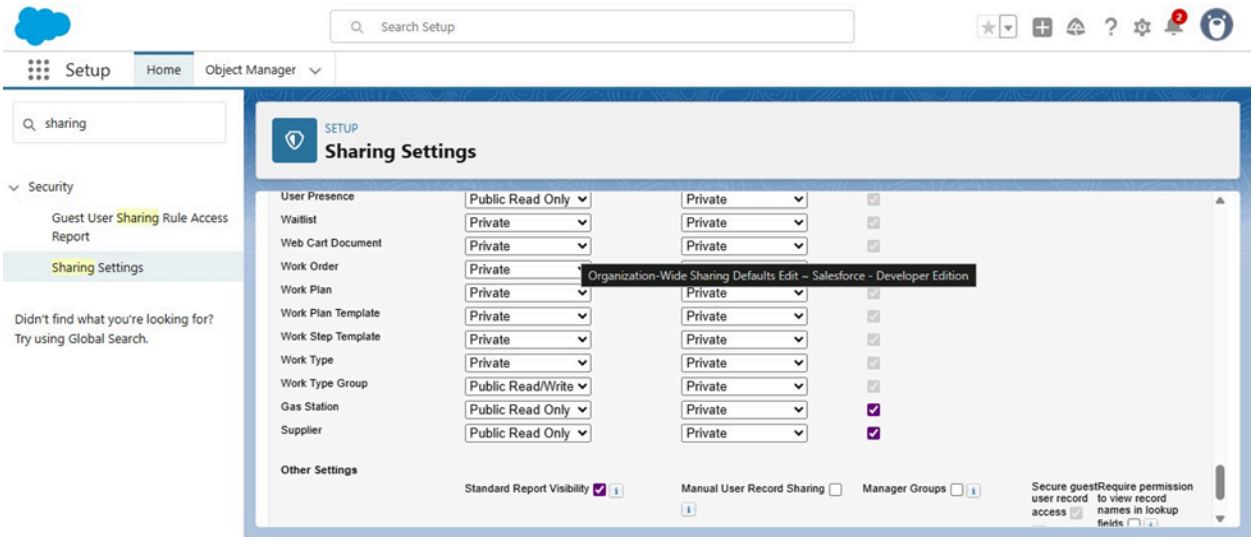
Profile & User Management



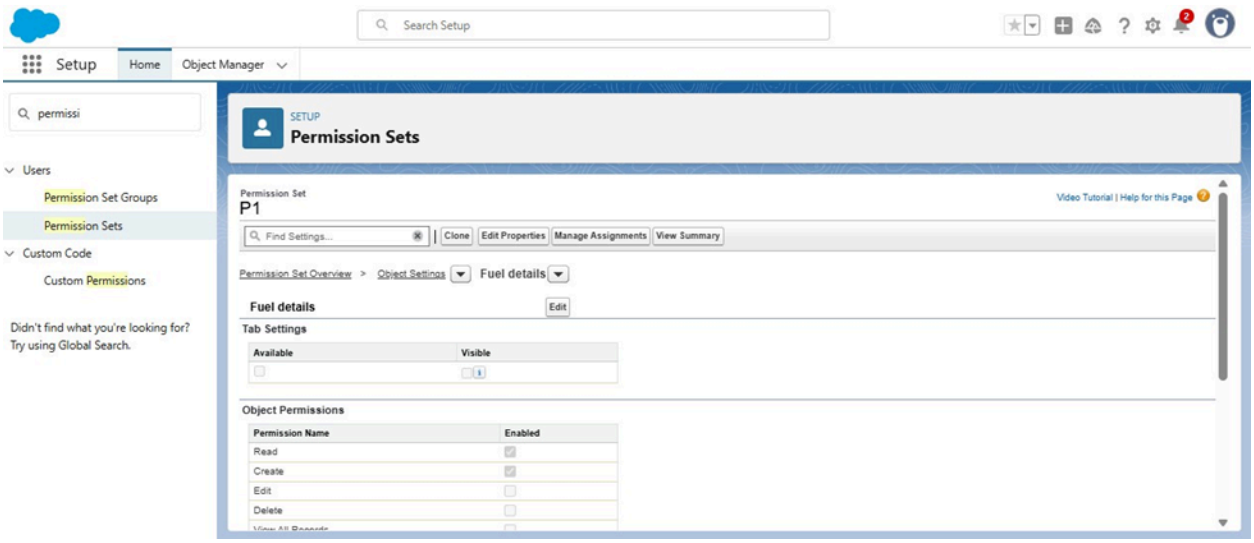

Search Setup
Setup
Home
Object Manager
profile
Users
Profiles
Didn't find what you're looking for?
Try using Global Search.
SETUP
Profiles
Profile
sales executive
Help for this Page
Users with this profile have the permissions and page layouts listed below. Administrators can change a user's profile by editing that user's personal information.
If your organization uses Record Types, use the Edit links in the Record Type Settings section below to make one or more record types available to users with this profile.
Login IP Ranges [0] | Enabled Apex Class Access [0] | Enabled Visualforce Page Access [0] | Enabled External Data Source Access [0] | Enabled Named Credential Access [0] | Enabled External Credential Principal Access [0] | Enabled Custom Metadata Type Access [0] | Enabled Custom Setting Definitions Access [0] | Enabled Flow Access [0] | Enabled Service Presence Status Access [0] | Enabled Custom Permissions [0]
Profile Detail
Edit
Clone
Delete
View Users
Name
sales executive
User License
Salesforce Platform
Custom Profile


Search Setup
Setup
Home
Object Manager
profile
Users
Profiles
Didn't find what you're looking for?
Try using Global Search.
SETUP
Profiles
Profile
sales person
Help for this Page
Users with this profile have the permissions and page layouts listed below. Administrators can change a user's profile by editing that user's personal information.
If your organization uses Record Types, use the Edit links in the Record Type Settings section below to make one or more record types available to users with this profile.
Login IP Ranges [0] | Enabled Apex Class Access [0] | Enabled Visualforce Page Access [0] | Enabled External Data Source Access [0] | Enabled Named Credential Access [0] | Enabled External Credential Principal Access [0] | Enabled Custom Metadata Type Access [0] | Enabled Custom Setting Definitions Access [0] | Enabled Flow Access [0] | Enabled Service Presence Status Access [0] | Enabled Custom Permissions [0]
Profile Detail
Edit
Clone
Delete
View Users
Name
sales person
User License
Salesforce Platform
Custom Profile

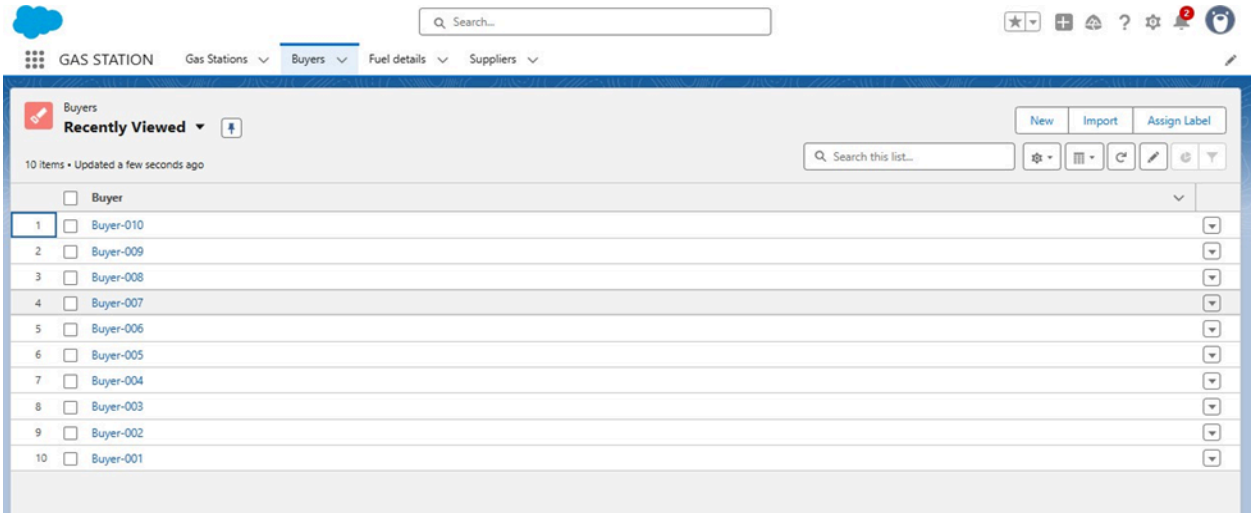
Configured OWD (Organization-Wide Defaults):

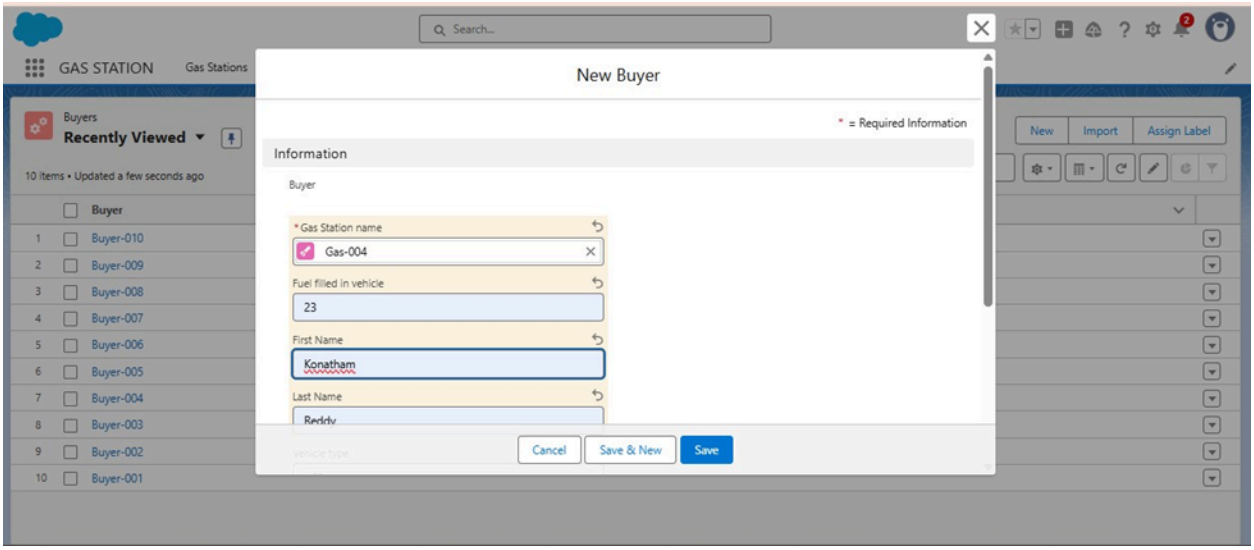


Created a Permission Set (P1):

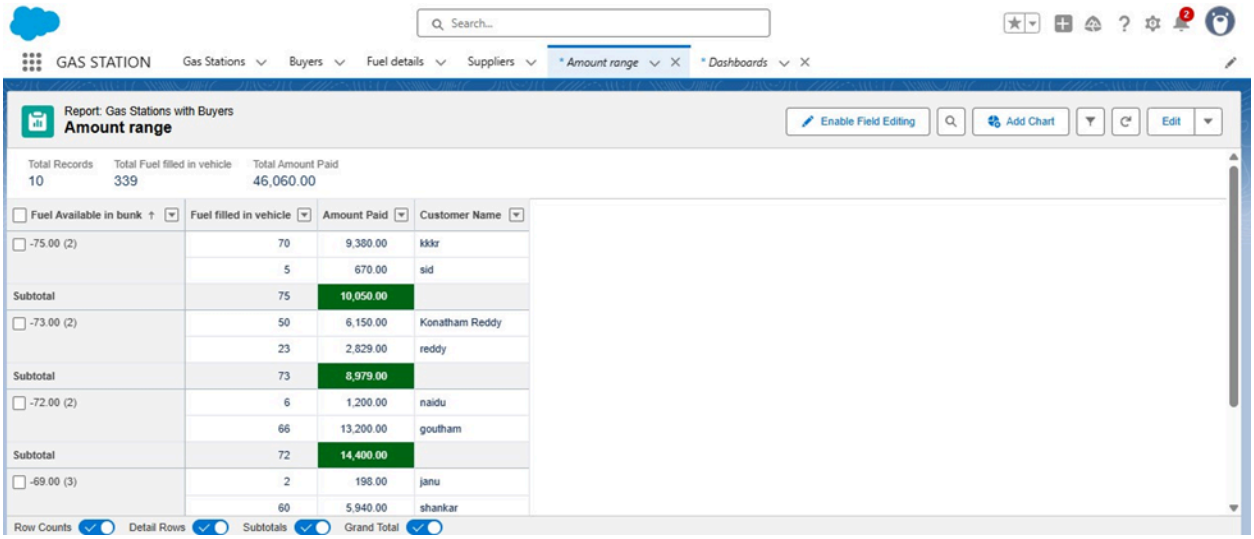


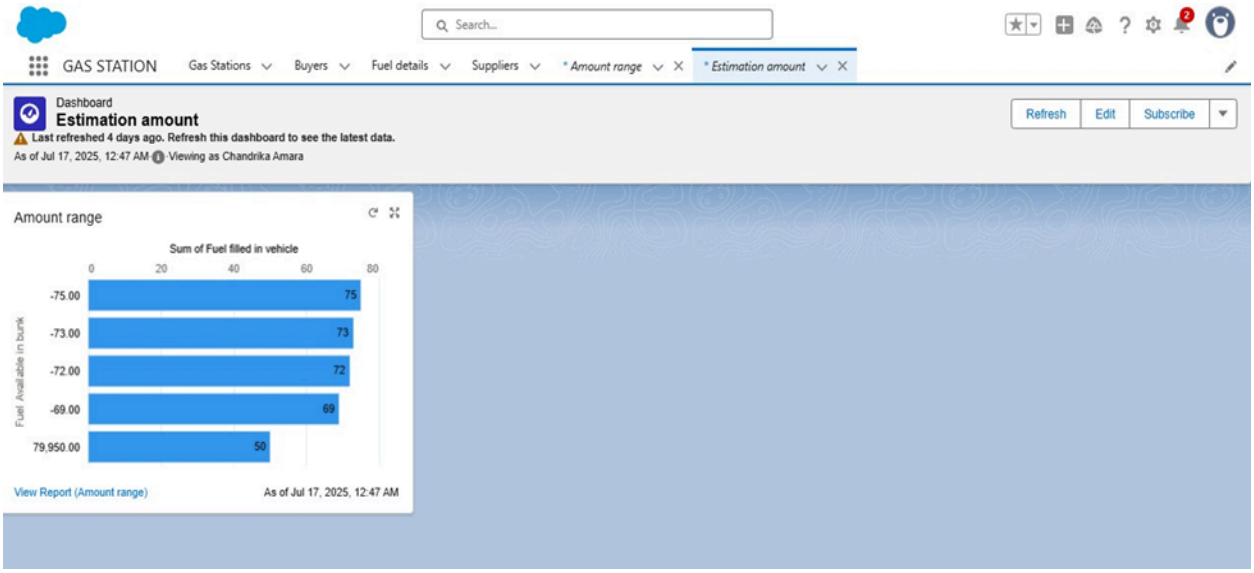
Data Population:



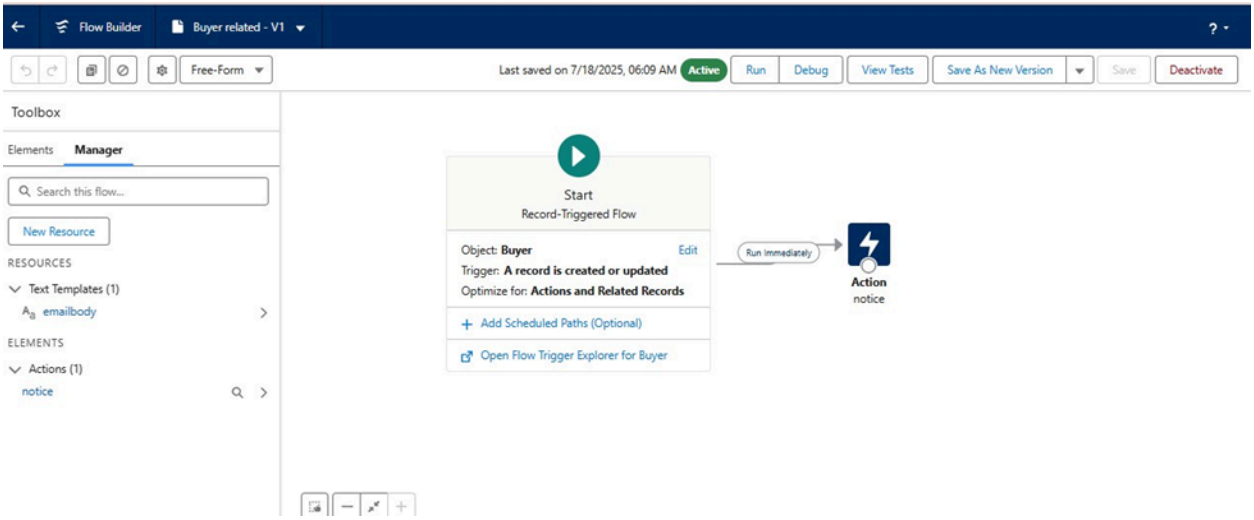


Reports & Dashboards :

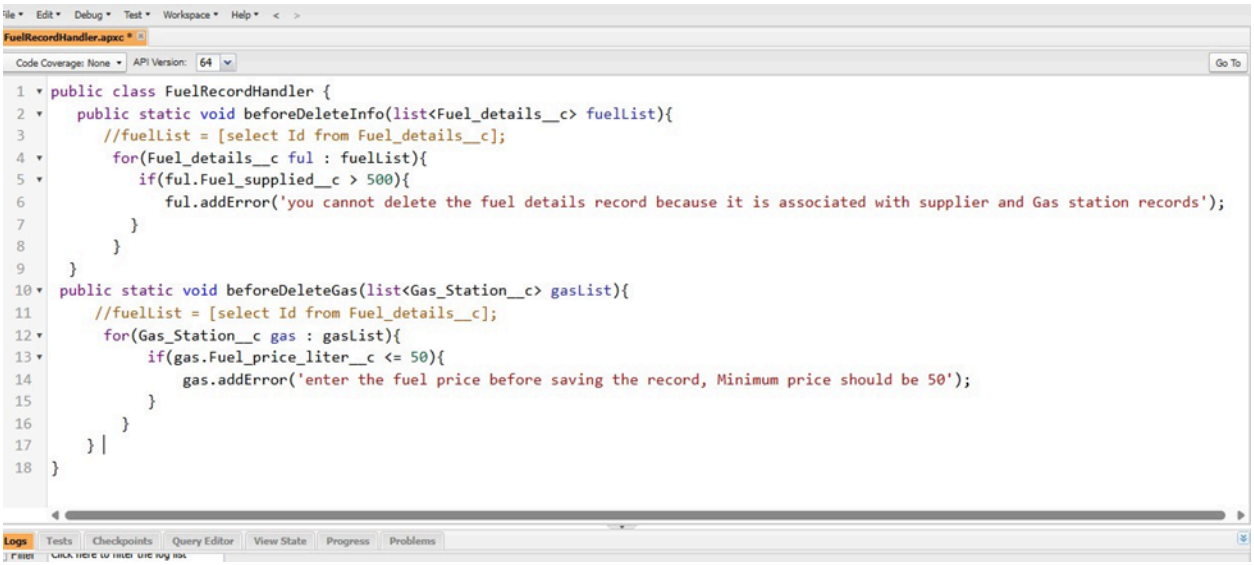




Flow Automation



Apex Automation & Business Validation :




```
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');
           }
        }
    }
}
```

Smart Internz

File Edit Debug Test Workspace Help < >

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

Code Coverage: None | API Version: 64

```
trigger beforeDelete on Fuel_details__c  (before Delete) {

    if(trigger.isbefore && trigger.isDelete){

        FuelRecordHandler.beforeDeleteInfo(trigger.old);

    }

}
```

File Edit Debug Test Workspace Help < >

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

Code Coverage: None | API Version: 64

```
trigger beforeInsert on Gas_Station__c (before insert ) {

    if(trigger.isbefore && trigger.isinsert){

            FuelRecordHandler.beforeDeleteGas(trigger.new);

    }
}
```

Advantages and Uses of the Gas Filling Station CRM Project

This project offers numerous advantages and practical uses for the business, moving it from manual to automated processes.

- **Improved Efficiency and Productivity**
    - **Automation:** Automated flows for tasks like inventory updates and task creation reduce manual work, allowing staff to focus on customer service.
    - **Data Centralization:** All data—from suppliers to buyers—is in one central location, eliminating the need for fragmented spreadsheets or paper records.
- **Enhanced Customer Relationship Management (CRM)**
    - **360-Degree Customer View:** The Buyer object and related records provide a complete view of a customer's history, enabling personalized service and targeted promotions.
    - **Loyalty Tracking:** The system can be used to track customer visits and purchases, forming the basis for a loyalty or rewards program.
- **Better Financial and Inventory Control**
    - **Real-time Reporting:** Dashboards provide instant visibility into daily sales, fuel consumption, and inventory levels, helping managers make informed decisions.
    - **Data Accuracy:** Validation rules prevent incorrect data from being entered, ensuring the integrity of financial and inventory records.
- **Increased Data Security**
    - **Role-Based Access:** Profiles and roles ensure that only authorized personnel can view or modify sensitive data. For example, a sales person cannot view a manager's financial reports.
    - **Audit Trail:** Field history tracking provides a clear record of who changed a record and when, which is crucial for accountability and troubleshooting.

- **Scalability and Flexibility**
    - **Customizable Platform:** Being built on Salesforce, the application can easily be extended with new objects, fields, and automation to adapt to future business needs.
    - **Integration Ready:** The platform is ready for future integrations, such as adding a mobile app for customers or a chatbot for support.

Conclusion

In conclusion, this Salesforce CRM project successfully delivered a comprehensive and functional solution for managing the operations of a gas filling station. The application, with its custom data model, automated processes, and robust security features, effectively addresses the core business needs of streamlining operations, improving data accuracy, and enhancing customer relationships. The project's success is a testament to the meticulous planning, development, and testing phases, resulting in a scalable and maintainable system that provides long-term value to the business. The project documentation serves as a valuable asset for future development, administration, and user training, ensuring the continued success and sustainability of the application.