GARAGE MANAGEMENT SYSTEM

Garage Management System addresses multiple operational challenges, streamlines workflows, enhances data visibility, and automates repetitive processes.Each feature and configuration is described with additional examples, process explanations, and use-case scenarios to provide a comprehensive understanding.

The Garage Management System is a valuable tool for automotive repair facilities, helping them deliver top-notch service, increase operational efficiency, and build lasting customer relationships. With its user-friendly interface and powerful features, GMS empowers garages to thrive in a competitive market while ensuring a seamless and satisfying experience for both customers and staff.

Project Overview:

The Garage Management CRM is a custom-built solution developed on the Salesforce platform to modernize and automate vehicle servicing operations. The system centralizes critical garage functions such as “customer management”, “appointment scheduling”, “vehicle tracking”, and “service history maintenance”. It enables garage staff to efficiently handle mechanic assignments, service requests, billing, and customer feedback in one place. By using “custom objects”, “flows”, and “validation rules”, the CRM reduces manual workload and increases operational accuracy.

Automation is a key strength of the system, with features like “automatic service record creation”, “billing updates”, and “email notifications” triggered through flows. Role-based access control ensures that data is securely handled by the right users, such as Managers and Salespersons. The CRM also supports decision-making through “real-time reports and dashboards”, offering insights into customer satisfaction, billing status, and service trends. The user interface is intuitive and tailored using Lightning App Builder with dynamic layouts. Ultimately, the system transforms traditional paper-based workflows into an efficient, digital-first process that improves both service delivery and business growth.

Objectives:

The primary objectives of building this Garage Management CRM are:

• To improve customer management by maintaining detailed vehicle and service histories:

All customer interactions, vehicle details, and service records are stored in one place.

This supports better follow-ups and personalized service.

• To streamline appointment bookings, mechanic assignments, and job tracking:

Appointments are booked with proper time management, and mechanics are auto-assigned. This reduces overlap, delays, and confusion.

• To implement automation for tasks like service creation, approvals, and notifications:

Flows and triggers automate routine processes like service record generation and email alerts. This saves time and minimizes human error.

• To enhance service quality and operational efficiency, leading to better customer satisfaction and business growth:

Faster workflows and improved communication increase customer trust. This results in higher retention and supports business expansion.

Phase 1: Requirement Analysis & Planning

Understanding Business Requirements

- The garage operates with multiple departments handling bookings, repairs, billing, and customer follow-ups.
- Previously, these processes were managed using physical registers or spreadsheets, leading to miscommunication, scheduling conflicts, and data loss.
- The business required a centralized CRM to efficiently manage customer data, service appointments, mechanic assignments, billing, and feedback.
- By digitizing these functions, the goal was to improve service delivery speed, ensure data consistency, and enhance the overall customer experience through automation.

Defining Project Scope and Objectives

- The project scope includes core modules such as “Customer Management”, “Appointment Scheduling”, “Service Tracking”, “Billing & Payments”, and “Customer Feedback”.
- It also covers automation features like service record creation, email alerts after service, and approval processes for high-value bills.
- The CRM is designed to streamline job allocation, track performance, and maintain detailed records for transparency and audit.
- Excluded from scope are external payment gateway integration and mobile app development, which are planned as future enhancements.

Design Data Model and Security Model

Five custom objects were designed to capture business needs:

• Customer_Details__c: Holds contact and vehicle information.

- Appointment__c: Manages bookings and scheduling.
- Service_Records__c: Tracks work done and service outcomes.
- Billing_Feedback__c: Combines payment status and customer satisfaction data.

Relationships were created between these objects using “lookup fields”, ensuring data integrity and logical flow. The “security model” includes a clearly defined role hierarchy: “Admin > Manager > Salesperson”, with custom profiles granting appropriate object-level and field-level access. “Permission sets” were used to extend access for specific tasks like billing approvals. The system uses “private OWD settings” with “sharing rules” to allow role-based data sharing between salespersons and managers.

Phase 2: Salesforce Development - Backend & Configurations

Setup Environment

A Salesforce Developer Org was created by signing up at developer.salesforce.com/signup.

The signup form required details like Name, Email, Role (Developer), College Name, Country, PIN Code, and a custom Username in email format.

After submission, a verification email was received and used to activate the account.

Upon verification, a new password was set and a security question answered to complete account setup.

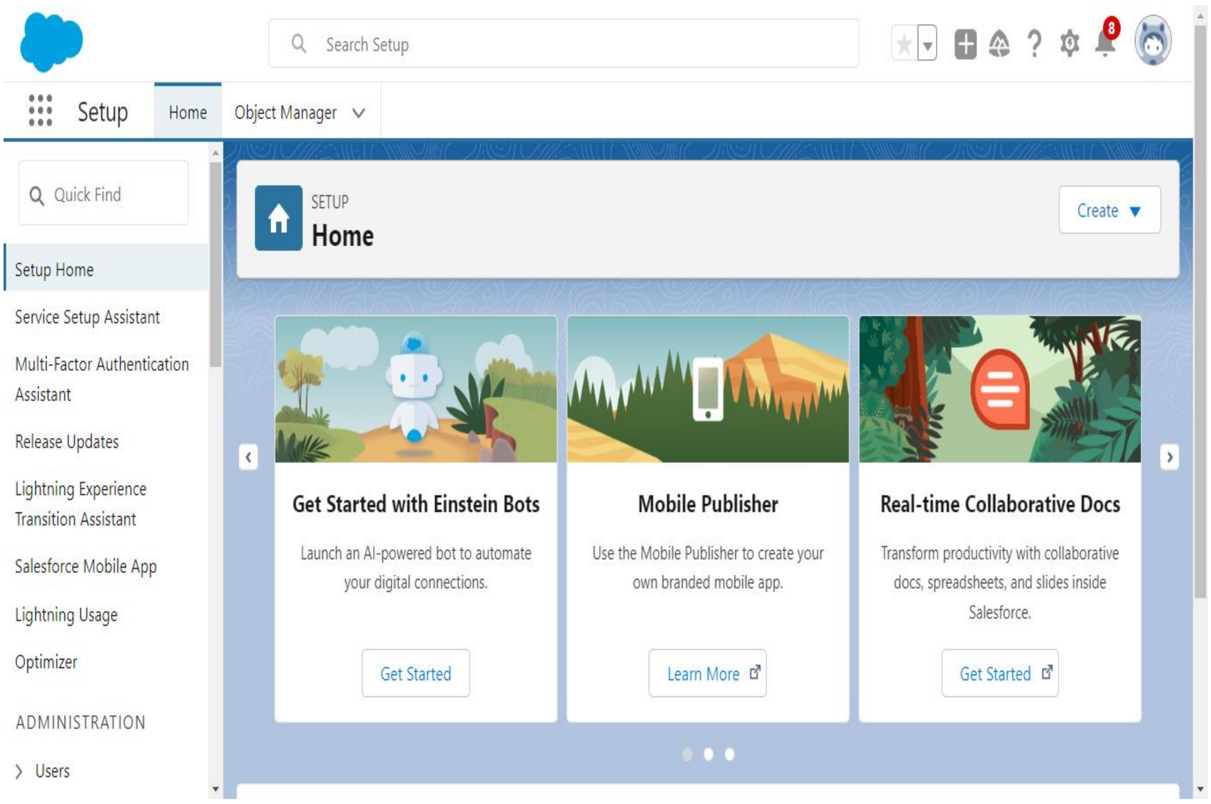
The user was then redirected to the Salesforce Setup page (Lightning Experience).

This org served as the development and testing environment for the entire Garage Management CRM project.

All custom objects, fields, flows, triggers, and validation rules were configured within this org.

No separate Sandbox or production environment was used due to the project’s academic/demo scope.

For versioning and deployment simulation, best practices like organizing metadata and modular testing were followed.


Search Setup

Setup | Home | Object Manager

Quick Find

Setup Home
Service Setup Assistant
Multi-Factor Authentication Assistant
Release Updates
Lightning Experience Transition Assistant
Salesforce Mobile App
Lightning Usage
Optimizer

ADMINISTRATION

Users

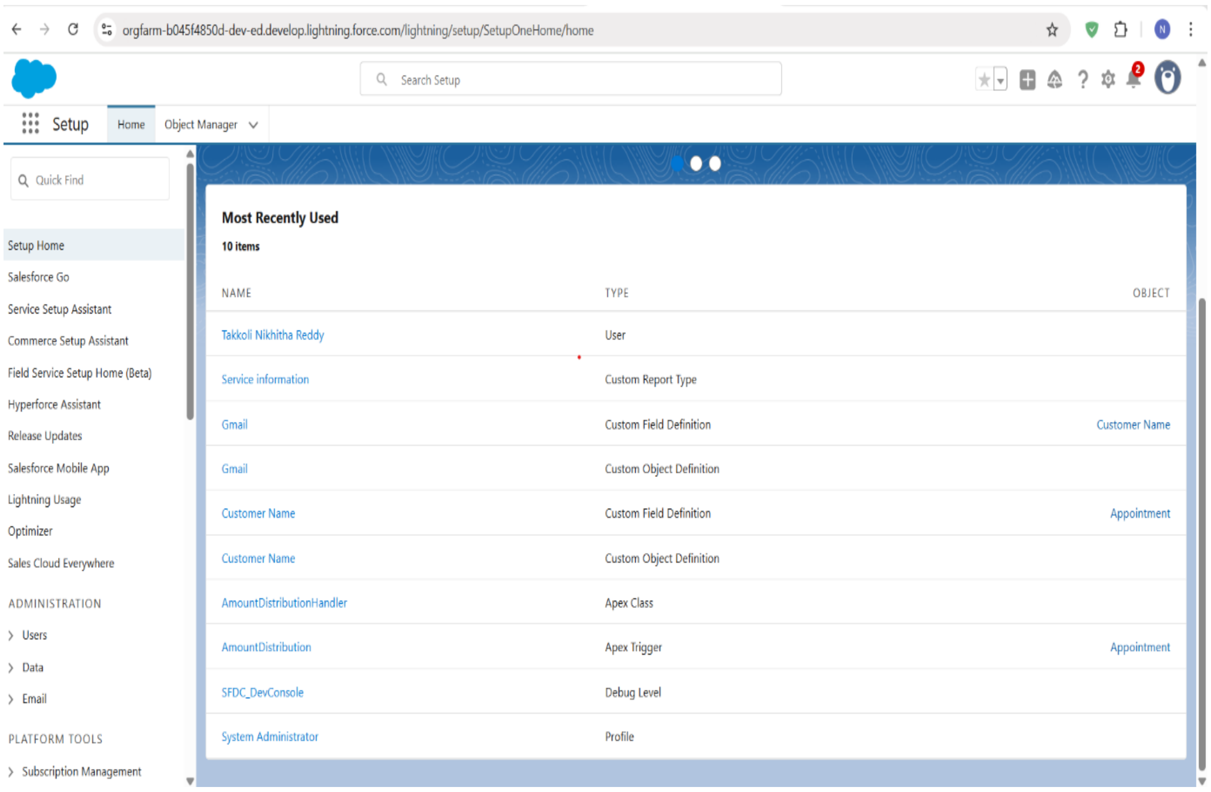
SETUP
Home

Create

Get Started with Einstein Bots
Launch an AI-powered bot to automate your digital connections.
Get Started

Mobile Publisher
Use the Mobile Publisher to create your own branded mobile app.
Learn More

Real-time Collaborative Docs
Transform productivity with collaborative docs, spreadsheets, and slides inside Salesforce.
Get Started


orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/lightning/setup/SetupOneHome/home

Search Setup

Setup | Home | Object Manager

Quick Find

Setup Home
Salesforce Go
Service Setup Assistant
Commerce Setup Assistant
Field Service Setup Home (Beta)
Hyperforce Assistant
Release Updates
Salesforce Mobile App
Lightning Usage
Optimizer
Sales Cloud Everywhere

ADMINISTRATION
Users
Data
Email

PLATFORM TOOLS
Subscription Management

Most Recently Used
10 items

NAME	TYPE	OBJECT
Takkoli Nikhitha Reddy	User	
Service information	Custom Report Type	
Gmail	Custom Field Definition	Customer Name
Gmail	Custom Object Definition	
Customer Name	Custom Field Definition	Appointment
Customer Name	Custom Object Definition	
AmountDistributionHandler	Apex Class	
AmountDistribution	Apex Trigger	Appointment
SFDC_DevConsole	Debug Level	
System Administrator	Profile	

Change Sets or Unmanaged Packages can be used in real scenarios for DevOps and deployment to other environments.

Customization

- **Custom Objects Information:**

1. Customer_Details__c

This object captures essential customer information including Name, Phone Number, Email, and Vehicle Details such as Vehicle Number, Brand, and Model. It serves as the parent record for customer-related operations. Other objects like `Appointment__c` reference this object using a Lookup relationship, ensuring that all appointments and service records are linked to the appropriate customer. This object supports the identification of repeat customers and helps maintain service history by customer and vehicle.

For Example to create any custom object we need to follow few steps. To create customer details object here is the process:

1) From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.
    - Enter the label name >> Customer Details
    - Plural label name >> Customer Details
    - Enter Record Name Label and Format
        - Record Name >> Customer Name
        - Data Type >> Text
2) Click on Allow reports and Track Field History,
3) Allow search >> Save.

2. Appointment__c

This object is used to manage service booking details. Key fields include Appointment Date, Appointment Status, and checkboxes like Maintenance Service, Repairs, and Replacement Parts. It has a Lookup to `Customer_Details__c`, connecting each appointment to the respective customer. It also has a Lookup to `Service_Records__c`, allowing users to track service delivery based on the appointment. Automation such as mechanic assignment and service amount calculation is triggered when this record is created or updated.

3. Service_Records__c

This object logs the actual service provided to the customer's vehicle. It contains fields like Service Type, Service Date, Mechanic Assigned, and Service Completed (checkbox). It is linked via a Lookup to Appointment__c, allowing traceability from service back to the appointment and customer. It also plays a key role in automation — when a service is marked completed, this can trigger flows to update billing or customer feedback components.

4. Billing_Feedback__c

A combined object that manages both payment and customer feedback. Fields include Billing Amount, Payment Status, Payment Paid, Feedback Rating, and Customer Comments. It connects to Service_Records__c through a Lookup field, allowing billing and feedback to be traced back to the specific service performed. Flows use this object to send automated thank-you emails and update payment information when service is complete.

- **Field Types**

…Lookup Fields…

Used to establish relationships between different objects.

Example Use Cases:

- Appointment__c has a lookup to Customer_Details__c to link each appointment to a specific customer.
- Service_Records__c has a lookup to Appointment__c for tracking service against bookings.
- Billing_Feedback__c has a lookup to Service_Records__c for linking feedback and billing to completed services.

These fields help maintain data integrity and enable cross-object reporting and automation.

…Checkbox Fields…

Used for binary selections like Yes/No or True/False.

Example Use Cases:

- Service_Completed__c on Service_Records__c indicates whether the job is done.
- Bill_Paid__c on Billing_Feedback__c flags if the customer has cleared their dues.

These are commonly used in flows, visibility rules, and status tracking.

…Date Fields…

Capture important calendar-based inputs.

Example Use Cases:

- Appointment_Date__c on Appointment__c for scheduling services.
- Service_Date__c on Service_Records__c to record when service was performed.

Date fields are used in reports, automations, and reminders.

…Currency Fields…

Store financial amounts like charges and payments.

Example Use Cases:

- Service_Amount__c on Appointment__c to store total billing based on selected services.
- Payment_Paid__c on Billing_Feedback__c to track actual payment made.

Currency fields support invoice generation, revenue analytics, and approval flows.

…Text Fields…

Used to enter and store free-form text.

Example Use Cases:

- Vehicle_Number__c on Customer_Details__c stores vehicle registration details.
- Mechanic_Notes__c on Service_Records__c records any technician comments.

These fields offer flexibility for detailed data capture.

…Picklist Fields…

Allow users to choose from a set list of values.

Example Use Cases:

- Service_Type__c (e.g., Oil Change, Engine Repair) on Service_Records__c.
- Appointment_Status__c (e.g., Scheduled, Completed) on Appointment__c.
- Feedback_Rating__c (e.g., Excellent, Good, Poor) on Billing_Feedback__c.

Picklists support consistent data, filters, and conditional visibility.

…Formula Fields…

Auto-calculate values based on logic and other field inputs.

Example Use Cases:

- Final_Amount__c formula that calculates total cost including tax.
- Status_Display__c to show a user-friendly status message (e.g., "Pending Payment") based on logic.

Formula fields reduce manual effort and provide real-time computed values across records.

- **Validation Rules:**

  I. Vehicle Number Plate Validation (`Appointment__c`)

This rule ensures that the Vehicle Number Plate entered by the user follows as standardized Indian format (e.g., `AP31AB1234`). It uses a REGEX formula to match the format `[A-Z]{2}[0-9]{2}[A-Z]{2}[0-9]{4}`.If the pattern doesn't match, an error message is shown:

"Please enter valid number."

The rule prevents invalid or inconsistent vehicle numbers from being saved, which helps with accurate tracking and reporting.

II. Service Rating Validation (`Billing_Feedback__c`)

This rule ensures that the Feedback Rating entered by the customer is a number between 1 and 5, using the REGEX pattern `[1-5]{1}`.If an invalid value is entered, the system throws an error:

"Rating should be from 1 to 5."

This enforces consistency in feedback data, allowing the business to measure service satisfaction accurately in reports and dashboards.Both rules were implemented at the field level and prevent records from being saved if invalid data is entered. They contribute to data standardization, improve report accuracy, and ensure reliable input for automation and decision-making processes.

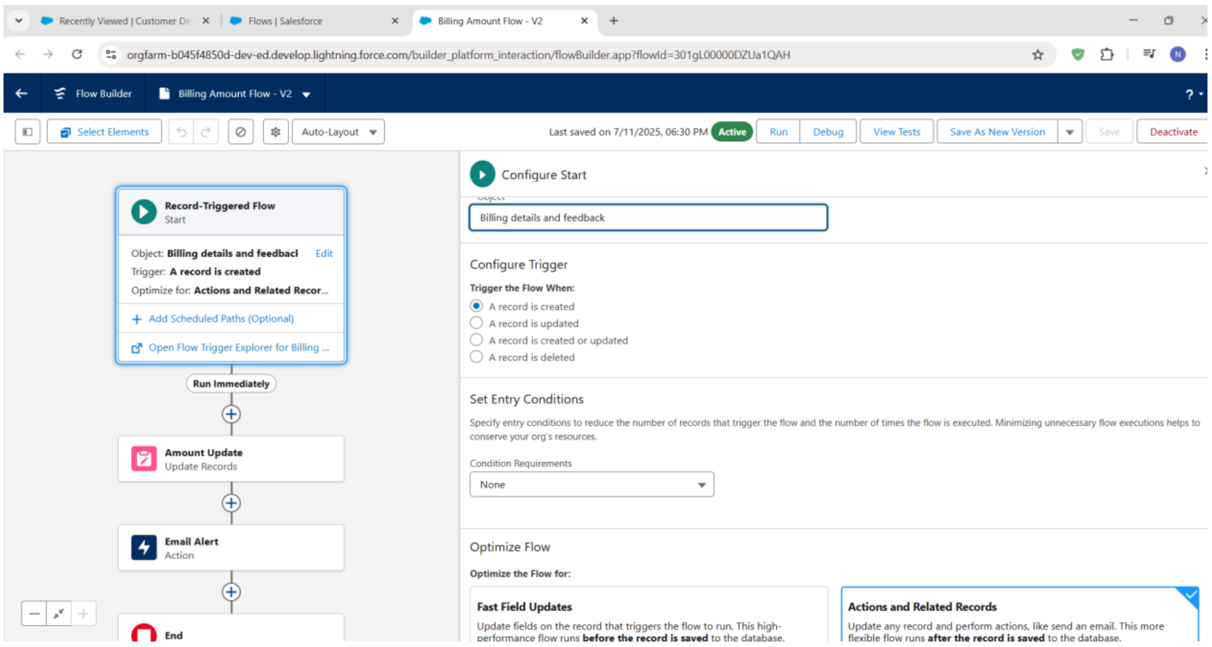
- **Automation Flows :**

A flow is a powerful tool that allows you to automate business processes, collect and update data, and guide users through a series of screens or steps. Flows are built using a visual interface and can be created without any coding knowledge.

Two record-triggered flows were developed to automate key business operations and reduce manual effort in the Garage Management CRM:

1. Billing Amount Flow (`Billing_Feedback__c`)

This flow is triggered when a billing record is created or updated. If the `Payment_Status__c` field equals "Completed", it updates the `Payment_Paid__c` field with the actual service amount (fetched from related Appointment via Service Record). It also sends a thank-you email to the customer using a Text Template and the customer's email stored in the record. This enhances customer experience and provides automated post-service communication.


Configure Start

Object: Billing details and feedback

Configure Trigger

Trigger the Flow When:
- (●) A record is created
- ( ) A record is updated
- ( ) A record is created or updated
- ( ) A record is deleted

Set Entry Conditions

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

Condition Requirements: None

Optimize Flow

Optimize the Flow for:

Fast Field Updates — Update fields on the record that triggers the flow to run. This high-performance flow runs **before the record is saved** to the database.

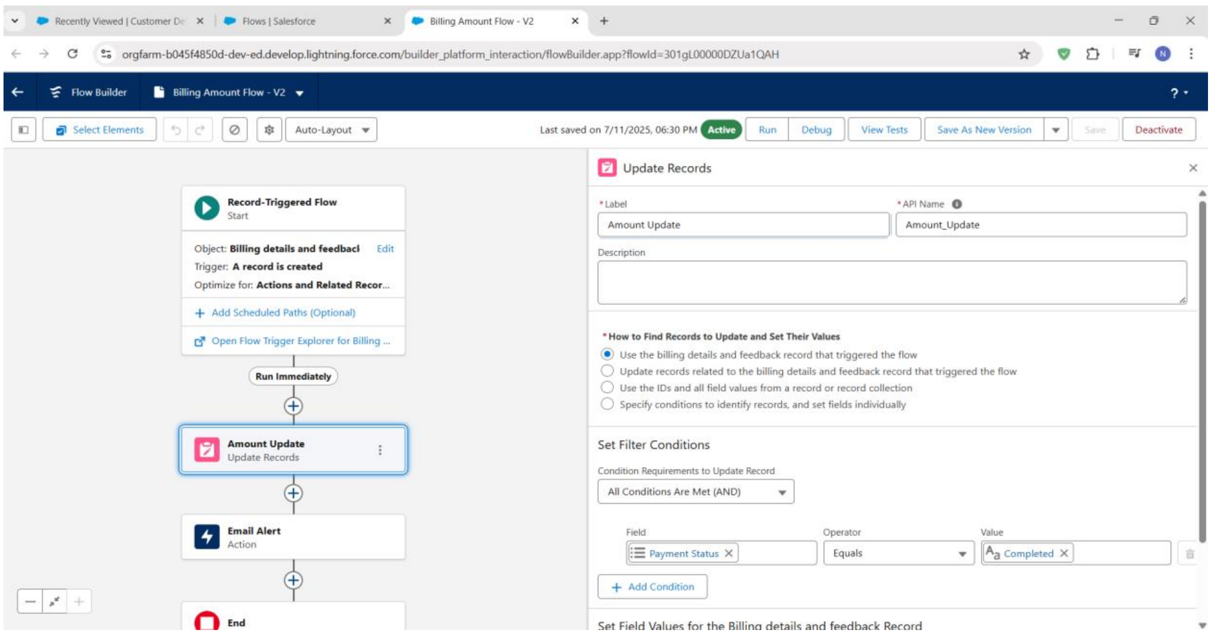
[✓ Actions and Related Records] — Update any record and perform actions, like send an email. This more flexible flow runs **after the record is saved** to the database.

Record-Triggered Flow (Start) — Object: Billing details and feedbacl; Trigger: A record is created; Optimize for: Actions and Related Recor...

Run Immediately → Amount Update (Update Records) → Email Alert (Action) → End

2. Update Service Status Flow (`Service_Records__c`)

This flow is triggered when a service record is created or updated. If the `Quality_Check_Status__c` field is marked True, the flow updates the `Service_Status__c` field to Completed. This ensures that only services which have passed quality review are marked as finished, supporting better operational accuracy.Both flows are optimized for Actions and Related Records, making them efficient and responsive. These automations improve data consistency, save time, and support customer satisfaction and service integrity.


Update Records

*Label: Amount Update

*API Name: Amount_Update

Description:

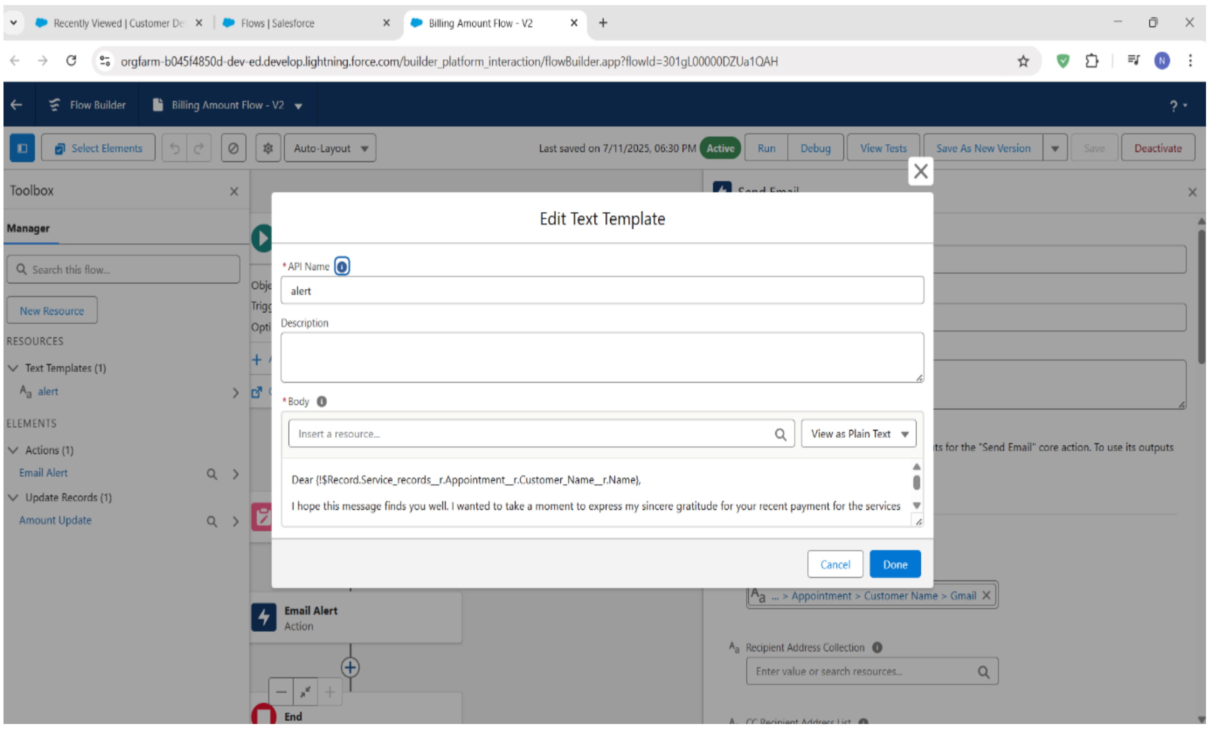
*How to Find Records to Update and Set Their Values
- (●) Use the billing details and feedback record that triggered the flow
- ( ) Update records related to the billing details and feedback record that triggered the flow
- ( ) Use the IDs and all field values from a record or record collection
- ( ) Specify conditions to identify records, and set fields individually

Set Filter Conditions

Condition Requirements to Update Record: All Conditions Are Met (AND)

Field	Operator	Value
Payment Status	Equals	Completed

Set Field Values for the Billing details and feedback Record



Edit Text Template

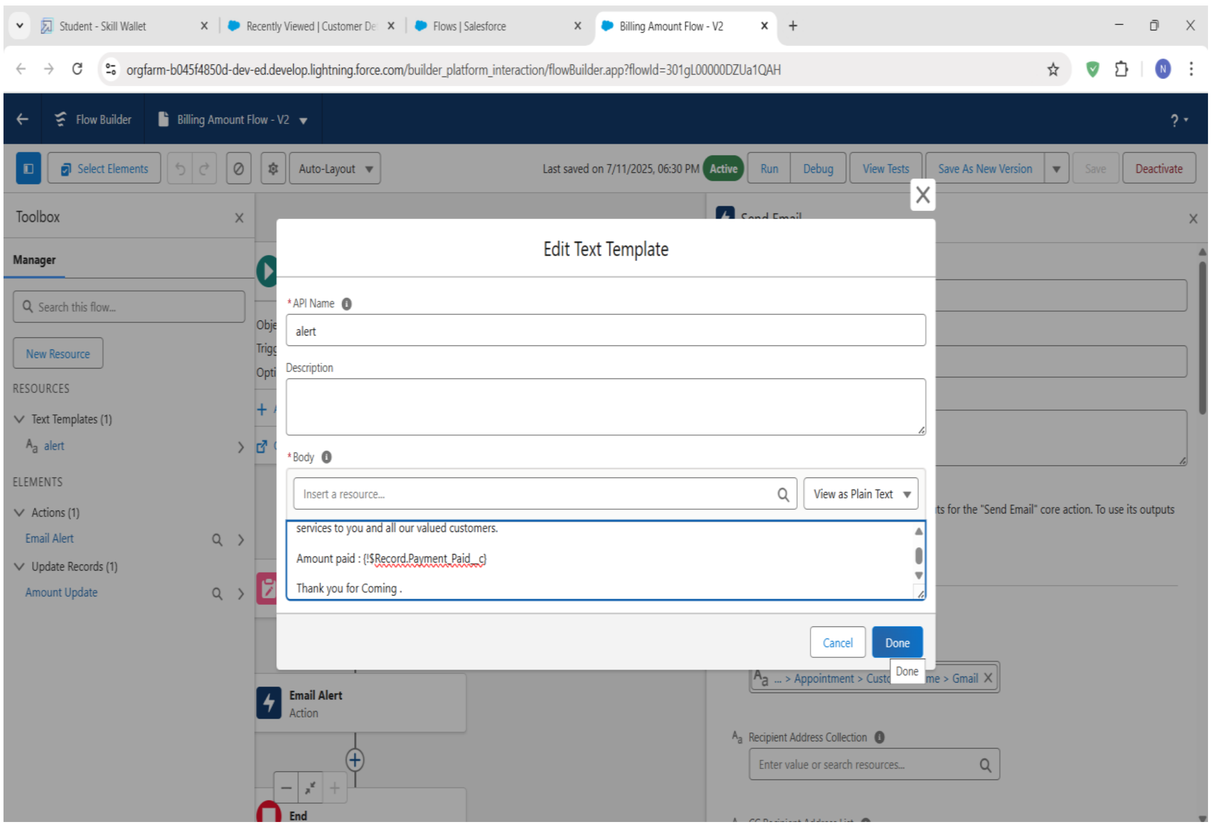
*API Name: alert

Description

*Body

Dear {!$Record.Service_records__r.Appointment__r.Customer_Name__r.Name},

I hope this message finds you well. I wanted to take a moment to express my sincere gratitude for your recent payment for the services



Edit Text Template

*API Name: alert

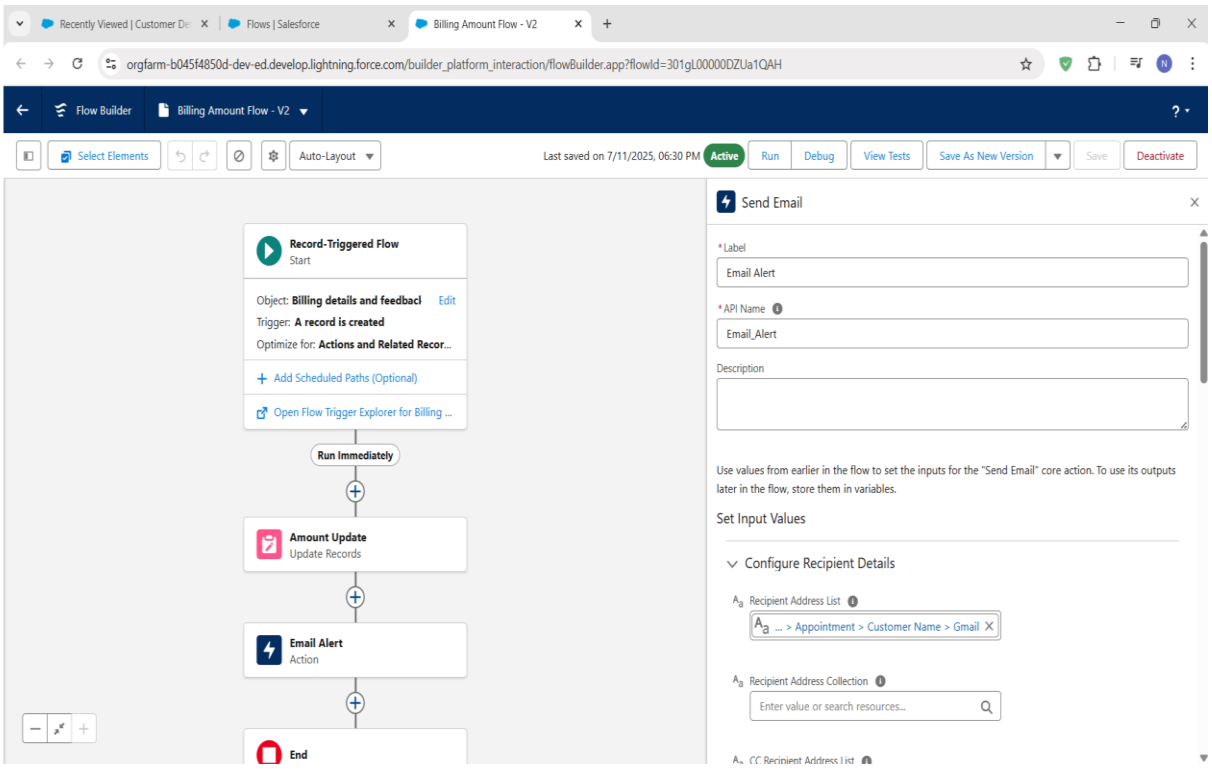
Description

*Body

services to you and all our valued customers.

Amount paid : {!$Record.Payment_Paid__c}

Thank you for Coming .



Recently Viewed | Customer De
Flows | Salesforce
Billing Amount Flow - V2
orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=301gL00000DZUa1QAH
Flow Builder
Billing Amount Flow - V2
Select Elements
Auto-Layout
Last saved on 7/11/2025, 06:30 PM
Active
Run
Debug
View Tests
Save As New Version
Save
Deactivate
Record-Triggered Flow
Start
Object: Billing details and feedbacl
Edit
Trigger: A record is created
Optimize for: Actions and Related Recor...
Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Billing ...
Run Immediately
Amount Update
Update Records
Email Alert
Action
End

Send Email

*Label
Email Alert

*API Name
Email_Alert

Description

Use values from earlier in the flow to set the inputs for the "Send Email" core action. To use its outputs later in the flow, store them in variables.

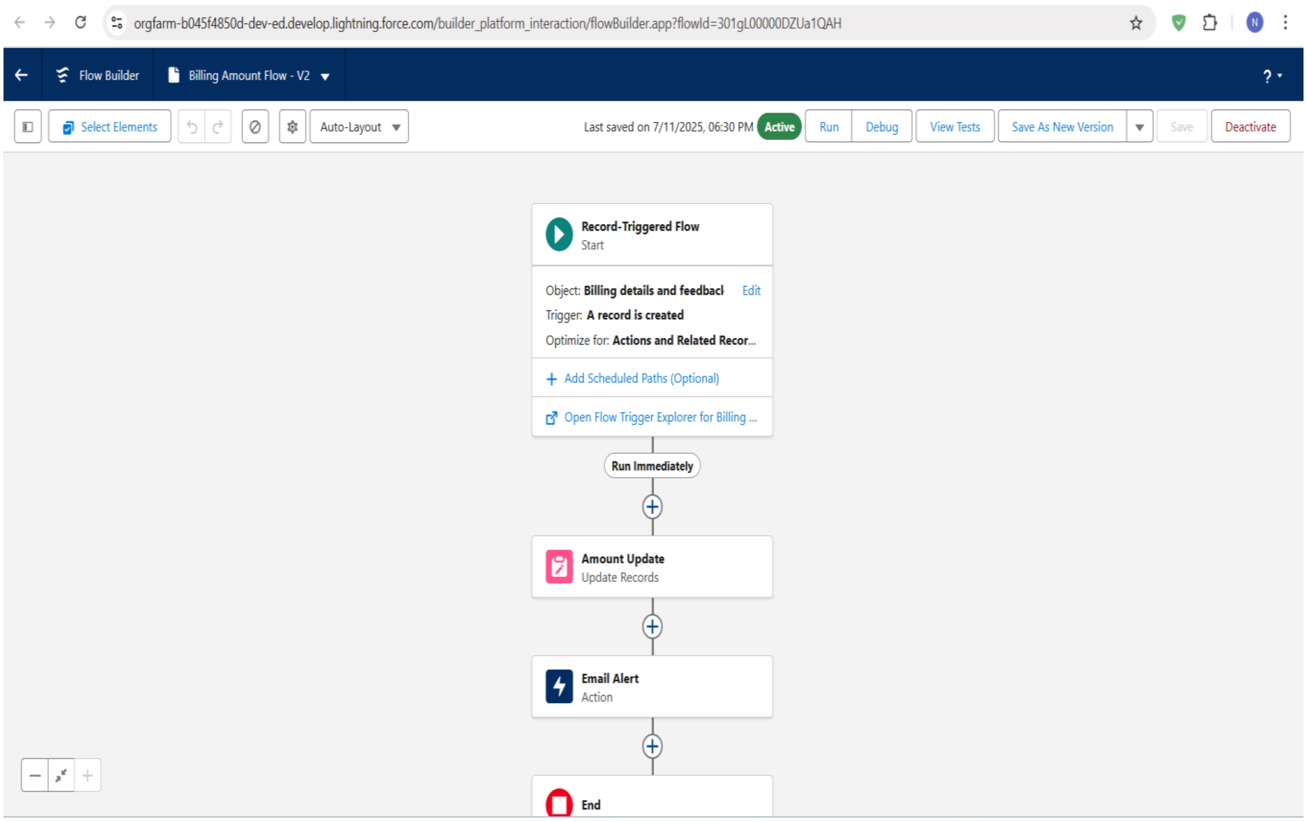
Set Input Values

Configure Recipient Details

Recipient Address List
... > Appointment > Customer Name > Gmail

Recipient Address Collection
Enter value or search resources...

CC Recipient Address List



orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=301gL00000DZUa1QAH
Flow Builder
Billing Amount Flow - V2
Select Elements
Auto-Layout
Last saved on 7/11/2025, 06:30 PM
Active
Run
Debug
View Tests
Save As New Version
Save
Deactivate
Record-Triggered Flow
Start
Object: Billing details and feedbacl
Edit
Trigger: A record is created
Optimize for: Actions and Related Recor...
Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Billing ...
Run Immediately
Amount Update
Update Records
Email Alert
Action
End

Apex Development

- Apex is Salesforce's programming language used to write custom backend logic.
- In this project, an Apex class named `AmountDistributionHandler` was created.
- It calculates the Service Amount for each appointment based on selected services.
- The services include Maintenance, Repairs, and Replacement Parts.
- Different combinations of these services result in different billing amounts.
- The class has a method `amountDist()` that receives a list of `Appointment__c` records.
- This logic is called using a before insert/update trigger on the Appointment object.
- The automation ensures accurate, consistent billing without manual data entry.
- It helps enforce business rules and simplifies future maintenance.

Apex enables advanced automation that goes beyond what Flows or Process Builder can do.

- The Apex class AmountDistributionHandler is responsible for calculating and setting the Service_Amount__c field on the Appointment__c object based on the types of services selected (like Maintenance, Repairs, and Replacement Parts).
- The amountDist method checks the combination of services (Maintenance, Repairs, Replacement Parts).Based on these combinations, it sets the total Service_Amount__c for each Appointment__c.

For example, you can have a trigger run before an object's records are inserted into the database, after records have been deleted, or even after a record is restored from the Recycle Bin. You can define triggers for top-level standard objects that support triggers, such as a Contact or an Account, some standard child objects, such as a CaseComment, and custom objects. To define a trigger, from the object management settings for the object whose triggers you want to access go to triggers.

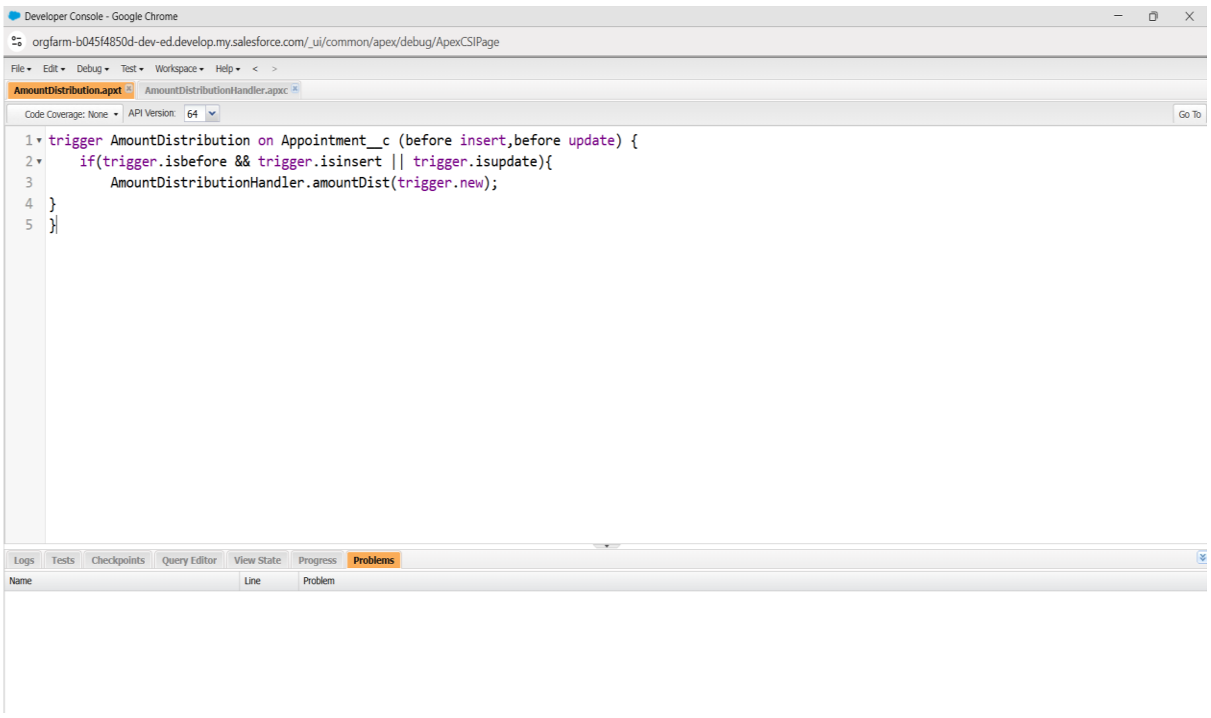
There are primarily two types of Apex Triggers:

> **Before Trigger:** This type of trigger in Salesforce is used either to update or validate the values of a record before they can be saved into the database. So, basically, the before trigger validates the record first and then saves it. Some criteria or code can be set to check data before it gets ready to be inserted into the database.
>
> **After Trigger:** This type of trigger in Salesforce is used to access the field values set by the system and affect any change in the record. In other words, the after trigger makes changes to the value from the data inserted in some other record.

Apex Handler and Trigger

- An Apex class named `AmountDistributionHandler` was developed to automatically calculate and assign service charges based on the combination of services selected in an appointment — including Maintenance Service, Repairs, and Replacement Parts.

- The class contains a method `amountDist()` which uses conditional logic to set a fixed amount in the `Service_Amount__c` field depending on which service checkboxes are marked as `true`.
- A corresponding Apex Trigger `AmountDistribution` was created on the `Appointment__c` object and set to fire before insert and before update.
- This trigger ensures that whenever a new appointment is created or an existing one is modified, the billing logic is executed automatically via the handler.
- This approach eliminates manual data entry for service pricing, ensures consistency in billing, and enforces standard business rules.
- It also enhances automation by integrating logic directly at the object level, making the system smarter and more reliable during real-time operations.


Developer Console - Google Chrome
orgfarm-b045f4850d-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
AmountDistribution.apxt
AmountDistributionHandler.apxc
Code Coverage: None
API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

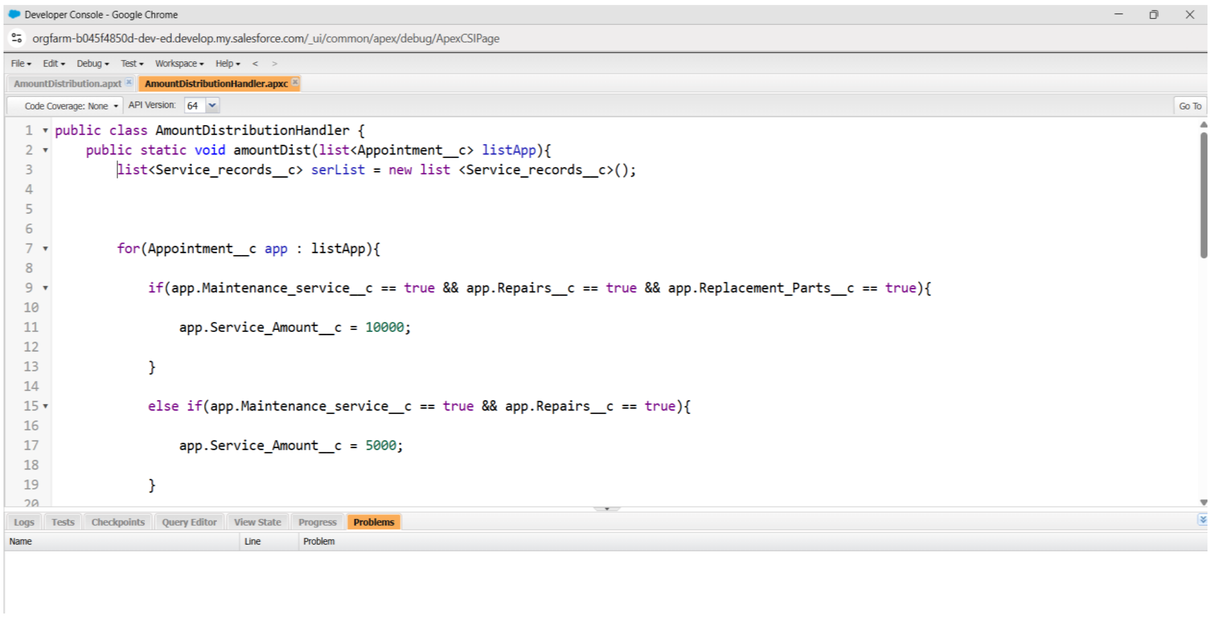
```
trigger AmountDistribution on Appointment__c (before insert,before update) {
    if(trigger.isbefore && trigger.isinsert || trigger.isupdate){
        AmountDistributionHandler.amountDist(trigger.new);
}
}
```

To create a new trigger :

1. While still in the trailhead account, navigate to the gear icon in the top right corner.
2. Click on developer console and you will be navigated to a new console window.
3. Click on File menu in the tool bar, and click on new? Trigger.
4. Enter the trigger name and the object to be triggered.
5. Name : AmountDistribution
6. sObject : Appointment__c

Handler for Appointment Object:

- Trigger Name: AmountDistribution
- Object: Appointment__c
- Events: before insert, before update
- Purpose: To invoke AmountDistributionHandler.amountDist() to handle logic like splitting or calculating amounts for appointments.



Developer Console - Google Chrome
orgfarm-b045f4850d-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
AmountDistribution.apxt
AmountDistributionHandler.apxc
Code Coverage: None
API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

```
public class AmountDistributionHandler {
    public static void amountDist(list<Appointment__c> listApp){
        list<Service_records__c> serList = new list <Service_records__c>();



        for(Appointment__c app : listApp){

            if(app.Maintenance_service__c == true && app.Repairs__c == true && app.Replacement_Parts__c == true){

                app.Service_Amount__c = 10000;

            }

            else if(app.Maintenance_service__c == true && app.Repairs__c == true){

                app.Service_Amount__c = 5000;

            }
```

Phase 3: UI/UX Development & Customization

This phase focused on building a user-friendly, visually consistent, and functional user interface (UI) for the Garage Management CRM using Salesforce Lightning tools. The goal was to ensure that both garage staff and managers could easily navigate, interact with, and manage records in the system.

Lightning App Setup:

An app is a collection of items that work together to serve a particular function. In Lightning Experience, Lightning apps give your users access to sets of objects, tabs, and other items all in one convenient bundle in the navigation bar.

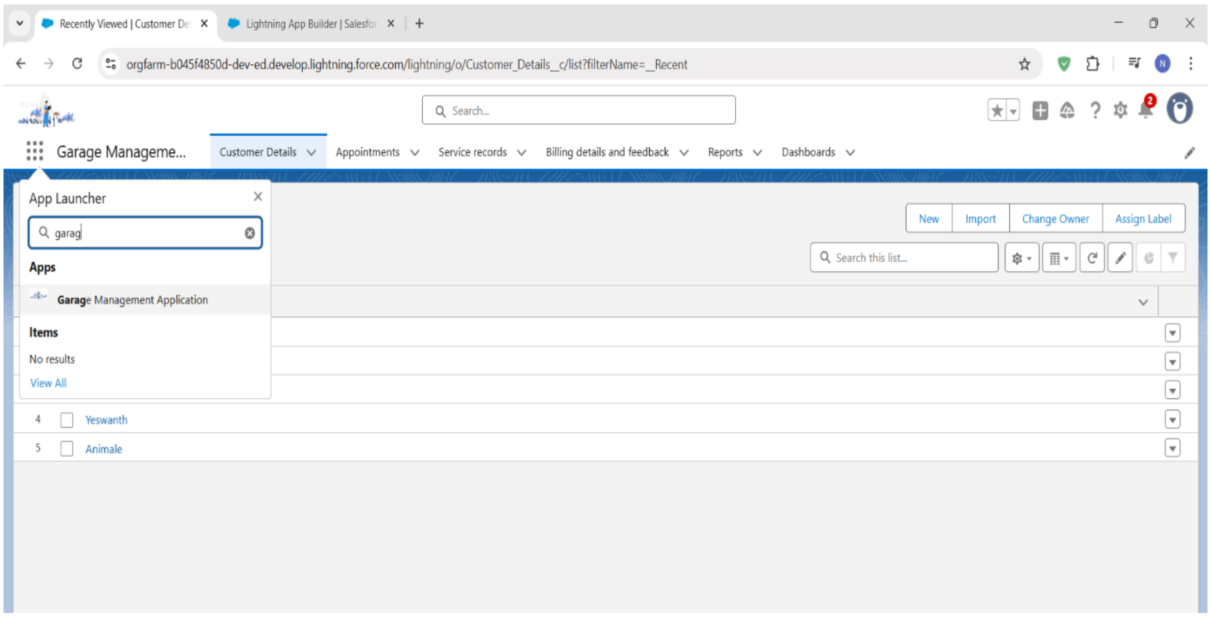
Lightning apps let you brand your apps with a custom colour and logo. You can even include a utility bar and Lightning page tabs in your Lightning app. Members of your org can work more efficiently by easily switching between apps.

A custom Lightning App named Garage Management was created using the App Manager in Salesforce.

This app served as the central entry point for users working on customer details, appointments, services, and billing. Key configurations included:

App visibility restricted to specific profiles (Manager and Sales Person).

The Garage Management app simplifies navigation by grouping all related modules into a single interface.


Recently Viewed | Customer De
Lightning App Builder | Salesfo
orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/lightning/o/Customer_Details__c/list?filterName=__Recent
Search...
Garage Manageme...
Customer Details
Appointments
Service records
Billing details and feedback
Reports
Dashboards
App Launcher
garag
Apps
Garage Management Application
Items
No results
View All
New
Import
Change Owner
Assign Label
Search this list...
4 Yeswanth
5 Animale

Four custom object tabs were created:

* Customer Details
* Appointments
* Service Records
* Billing Details and Feedback
* Each tab was added via the Custom Tabs section and linked to the Garage Management app.

User Management – Profiles and access:

Manager Profile

* Cloned from Standard User.
* Assigned as the default profile for senior users.
* Granted full Create, Read, Edit, Delete (CRED) access to all custom objects.
* Session timeout set to 8 hours, and password policy set to "Never Expires".
* Garage Management was set as the default app upon login.

Sales Person Profile

* Cloned from Salesforce Platform User.

* Given limited object permissions (typically Read, Create, Edit).

* Excluded from sensitive fields like internal comments or full billing configuration.

* Garage Management app set as default for easy access.

3.Page Layouts & Dynamic Forms

Each custom object was configured with a tailored page layout to ensure users see only relevant fields:

Fields like Billing Amount and Service Feedback were placed logically and grouped using sections.

* For the Appointments and Service Records objects:

* Dynamic Forms were used to show/hide fields based on field values.

* Example: The "Feedback" section only appears after a service is marked as completed.

* Related records like Service Records within an Appointment were displayed using Related Lists.

4. Lightning Record Pages

Using the Lightning App Builder, custom record pages were built for each object:

* Highlights Panels: Displayed key fields like Customer Name, Appointment Date, Service Status at the top.

* Tabbed Layouts: Helped divide form content into tabs like "Service Info", "Billing Details", "Feedback".

* Related Lists & Related Record Components: Displayed associated records (e.g., all service records under an appointment).

* Mobile responsiveness was also ensured for use on Salesforce mobile app.

These custom pages offered a cleaner and more intuitive user experience, reducing training time.

Reports and Dashboards

Reports give you access to your Salesforce data. You can examine your Salesforce data in almost infinite combinations, display it in easy-to-understand formats, and share the resulting insights with others. Before building, reading, and sharing reports, review these reporting basics.

Dashboards help you visually understand changing business conditions so you can make decisions based on the real-time data you've gathered with reports. Use dashboards to help users identify trends, sort out quantities, and measure the impact of their activities. Before building, reading, and sharing dashboards, review these dashboard basics.

- **Reports:**

A custom report was created using Report Builder with the following configuration:

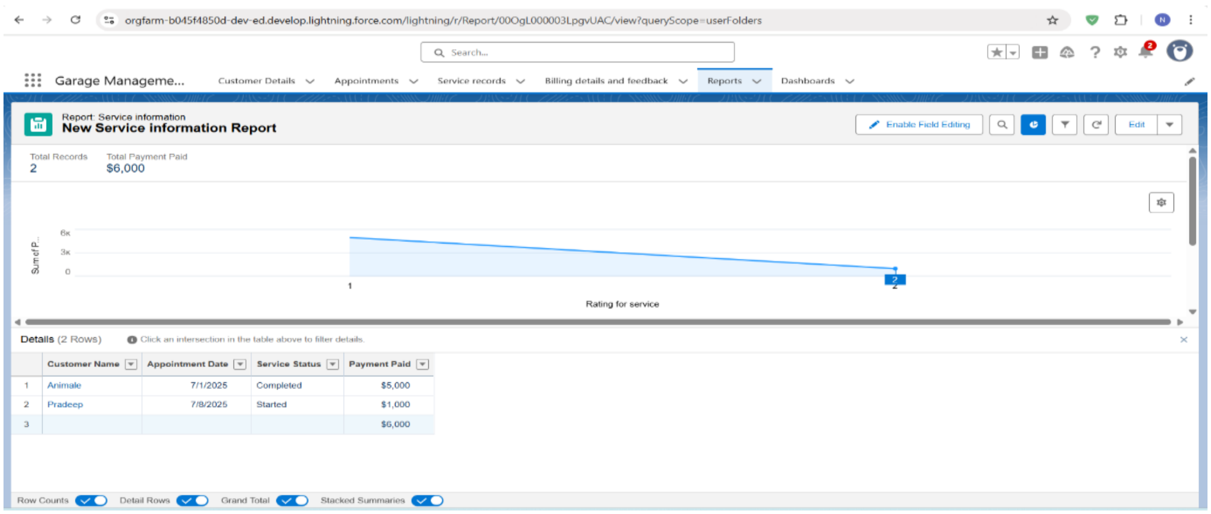
Report Type: "Service Information" under Other Reports

Selected Fields:

- Customer Name
- Appointment Date
- Service Status
- Payment Paid

Group Rows:

- Rating for Service
- Payment Status
- Chart: A Line Chart was added to visualize how payment and service ratings correlate over time.
- Purpose: Help managers monitor customer satisfaction and identify bottlenecks or issues in billing or service quality.



Report: Service information
New Service information Report

Total Records 2
Total Payment Paid $6,000

Details (2 Rows) Click an intersection in the table above to filter details.

	Customer Name	Appointment Date	Service Status	Payment Paid
1	Animale	7/1/2025	Completed	$5,000
2	Pradeep	7/8/2025	Started	$1,000
3				$6,000

- **Dashboards:**

A dashboard was created using the above report with the following setup:

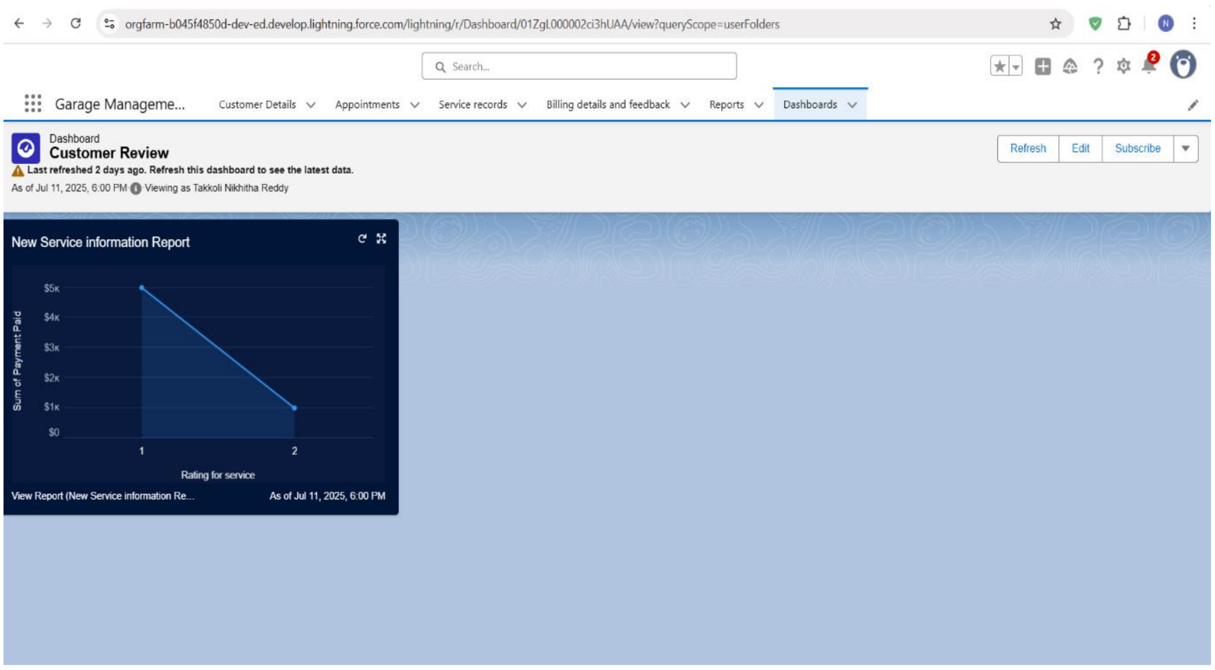
- Component Type: Line Chart

- Theme: Customized for visibility
- Title: “Garage Service Report”

Subscription:

- Scheduled Weekly
- Sent every Monday
- Helps management track weekly performance and customer trends

This dashboard serves as a decision-making tool for business stakeholders to evaluate service delivery and identify improvement areas.



orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/lightning/r/Dashboard/01ZgL000002ci3hUAA/view?queryScope=userFolders

Dashboard
Customer Review
Last refreshed 2 days ago. Refresh this dashboard to see the latest data.
As of Jul 11, 2025, 6:00 PM Viewing as Takkoli Nikhitha Reddy

- **Lightning Pages:**

Lightning Pages are custom record page layouts that define what users see when they open a record (e.g., an appointment or a service record). You customized these pages using the Lightning App Builder to make them user-friendly and role-specific.

1. Display Key Fields Prominently

For example, the `Appointment__c` Lightning Page displays fields like Appointment Date, Vehicle Number, and Status in the **highlights panel** for quick reference.

2. Organize Details with Tabs

Pages like `Service_Records__c` are organized into tabs (e.g., Details, Related, Notes) so that users can easily access relevant sections without scrolling too much.

3. Include Related Lists and Lookups

You used Lightning Pages to show related records like services under appointments or billing linked to a service record using related list components.

4. Support Role-Specific UX

   The layout helps Sales Persons or Managers view only what's necessary for their job (e.g., Managers see more reporting-related fields, Sales Persons see only service and appointment data).

5. Integrate with Automation

   Flows or quick actions (e.g., "Send Payment Confirmation") can be added to Lightning Pages, enabling users to trigger automated tasks directly from the page.

6. Improve Overall Navigation

   By using Lightning Pages for each object, you created a clean and modern UI for navigating garage operations.

Phase 4: <u>Data Migration, Testing & Security</u>

This phase focuses on ensuring secure data handling, accurate role-based access, and functional testing of all major CRM features. While the project was developed on a Salesforce Developer Org (sandbox-like environment), key production-like configurations were implemented to mirror real-world behavior.

Data Migration

No bulk data import was performed using Data Import Wizard or Data Loader in this project. Instead, "manual data entry" was used to create realistic sample records across all custom objects:

- Customer_Details__c
- Appointment__c
- Service_Records__c
- Billing_Feedback__c

Creating records manually allowed for better testing of validations, flows, triggers, and relationship mappings in a controlled way.

Matching Rules and Duplicate Rule

To maintain clean and non-redundant customer data, both a Matching Rule and a Duplicate Rule were created:

Matching Rule

- Created on the `Customer_Details__c` object
- Checks for duplicate entries based on “Gmail” and “Phone Number” fields
- Matching method used: “Exact Match”
- Ensures two customers with the same contact information are not accidentally added.

Duplicate Rule

- Uses the above matching rule
- Prevents saving duplicate customer records
- Error message prompts users to review the existing data

Both rules were activated to enforce real-time data quality during record creation or updates.

Roles and Role Hierarchy

A role in Salesforce defines a user's visibility access at the record level. Roles may be used to specify the types of access that people in your Salesforce organization can have to data. Simply put, it describes what a user could see within the Salesforce organization.

Salesforce uses a role hierarchy to determine record access.

Users at higher levels in the hierarchy have greater access to records owned by or shared with users lower in the hierarchy.

The role hierarchy is often used in combination with OWD settings to grant different levels of access.

A structured role hierarchy was configured to simulate team-based access control. The roles created were:

1.Manager (Top-level)

2.Sales Person 1 (Reports to Manager)

- This hierarchy enables record access inheritance, where a manager can view and control the records created by the salespeople under them. It also supports future sharing logic or approvals that depend on role levels.

Sharing Settings

Salesforce allows you to configure sharing settings to control how records are accessed and shared within your organization. These settings are crucial for maintaining data security and privacy. Salesforce provides a variety of tools and mechanisms to define and enforce sharing rules.

To secure service-related data and allow selective visibility:

The OWD settings define the default level of access for all objects within your Salesforce org.

OWD settings include Private, Public Read-Only, Public Read/Write, and Controlled by Parent.

OWD settings can be configured for each standard and custom object.

The OWD (Organization-Wide Default) for the `Service_Records__c` object was set to Private, meaning records are only visible to the record owner and users above them in the role hierarchy.

Sharing rules are used to extend access to records for users who meet specific criteria.

They can be used to grant read-only or read-write access to records owned by other users.

Manual Sharing:

Administrators and record owners can manually share specific records with other users or groups.

A Sharing Rule (Sharing setting) was created:

- Source: Records owned by users in the Sales Person role
- Target: Shared with the Manager role
- Access Level: Read/Write
- Purpose: Managers can supervise and manage service records without changing OWD to public

This configuration enhances record-level security while maintaining operational transparency for supervisors.

Profiles & Permission Management

Profiles and permission sets allow administrators to specify object-level and field-level permissions for users.

Profiles are typically used to grant general object and field access, while permission sets can be used to extend those permissions to specific users.

A profile is a group/collection of settings and permissions that define what a user can do in salesforce. Profile controls "Object permissions, Field permissions, User permissions, Tab settings, App settings, Apex class access, Visualforce page access, Page layouts, Record Types, Login hours & Login IP ranges. You can define profiles by the user's job function. For example System Administrator, Developer, Sales Representative.

One type of the profile is Custom Profiles.

Custom ones defined by us.

They can be deleted if there are no users assigned with that particular one.

Two custom profiles were created by cloning standard Salesforce profiles:

Manager Profile

- Cloned from: Standard User
- Full access to all custom objects
- Assigned to Manager users

- Default app set to: Garage Management
- Custom password policy: never expires, 8-character minimum

Sales Person Profile

- Cloned from: Salesforce Platform User
- Assigned to Sales team users
- Object permissions limited to create, read, and edit (no delete)
- Sharing and flows used to manage extended access

These profiles ensure role-specific visibility, enforce field-level security, and restrict access to only required components.

Apex Trigger and Handler Execution

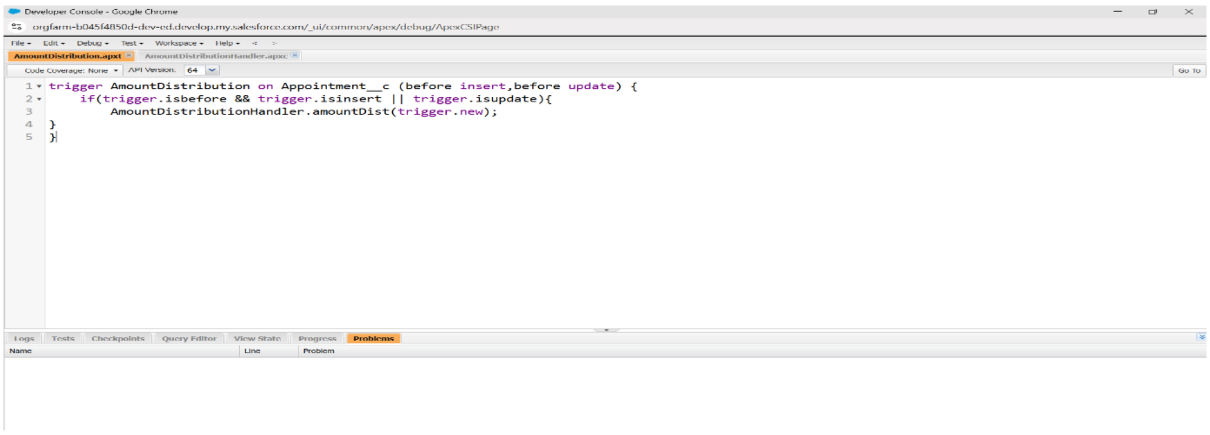
To automate billing logic:

- An Apex Class named `AmountDistributionHandler` was created.
- A Trigger named `AmountDistribution` was set on the `Appointment__c` object.
- The trigger runs before insert and before update.

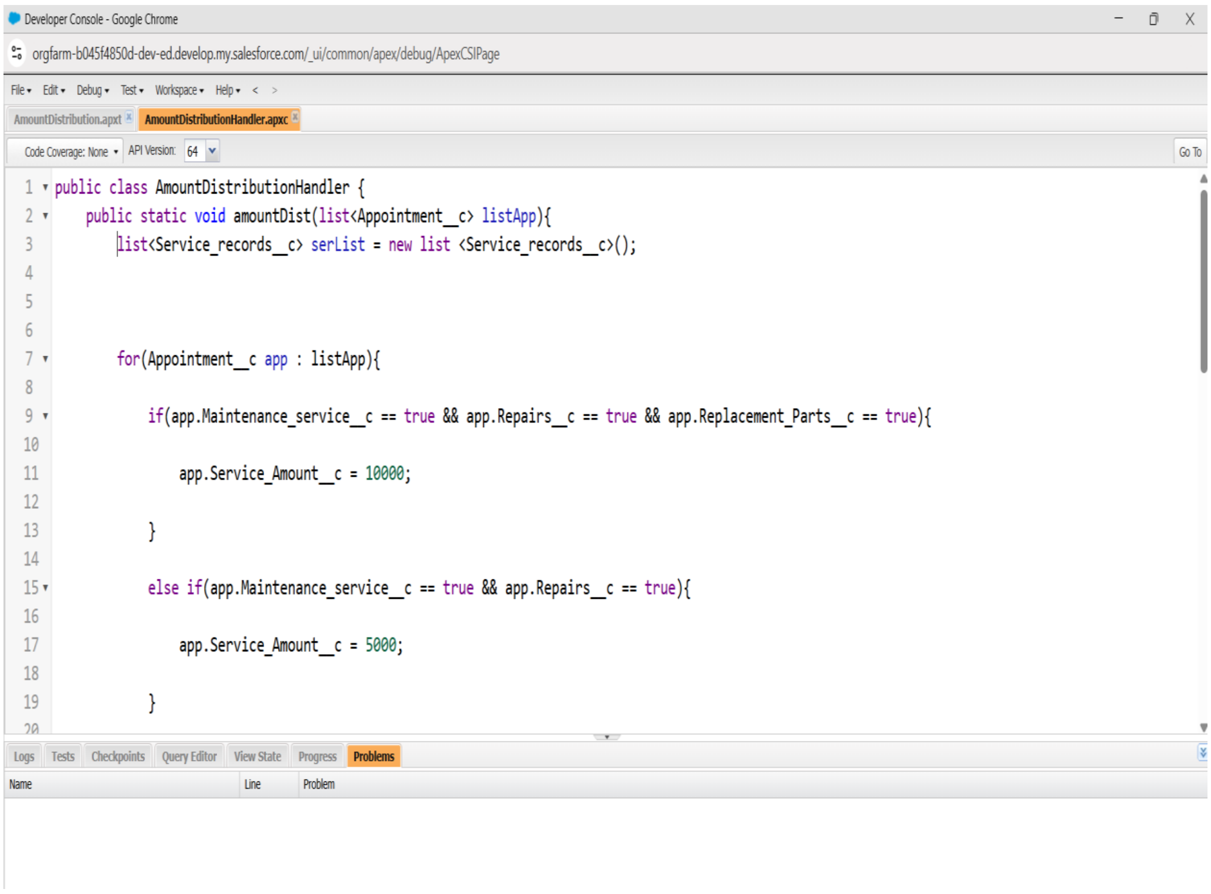
It invokes logic from the handler to assign billing amounts automatically based on service checkboxes like:

- Maintenance
- Repairs
- Replacement Parts

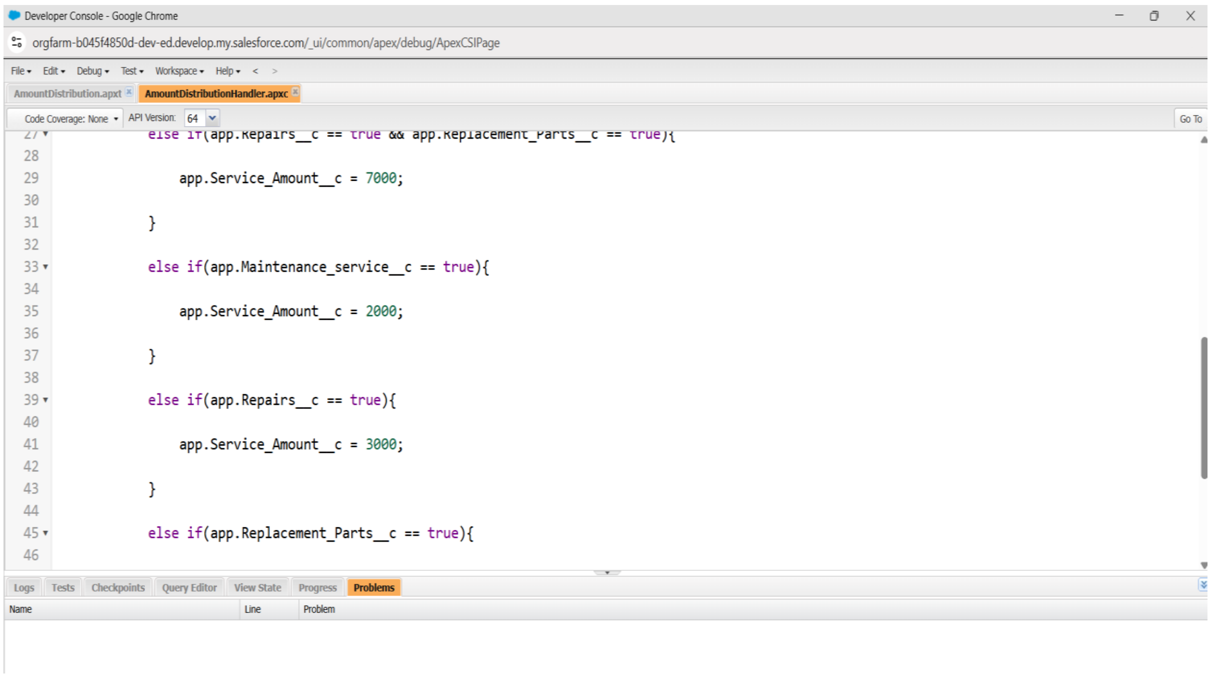
This ensures service charges are assigned consistently and without manual intervention.


Developer Console - Google Chrome
orgfarm-b045f4850d-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
AmountDistribution.apxt
AmountDistributionHandler.apxc
Code Coverage: None
API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

```
trigger AmountDistribution on Appointment__c (before insert,before update) {
    if(trigger.isbefore && trigger.isinsert || trigger.isupdate){
        AmountDistributionHandler.amountDist(trigger.new);
}
}
```


Developer Console - Google Chrome
orgfarm-b045f4850d-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help < >
AmountDistribution.apxt
AmountDistributionHandler.apxc
Code Coverage: None
API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

```
public class AmountDistributionHandler {
    public static void amountDist(list<Appointment__c> listApp){
        list<Service_records__c> serList = new list <Service_records__c>();



        for(Appointment__c app : listApp){

            if(app.Maintenance_service__c == true && app.Repairs__c == true && app.Replacement_Parts__c == true){

                app.Service_Amount__c = 10000;

            }

            else if(app.Maintenance_service__c == true && app.Repairs__c == true){

                app.Service_Amount__c = 5000;

            }
```


Developer Console - Google Chrome
orgfarm-b045f4850d-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help < >
AmountDistribution.apxt
AmountDistributionHandler.apxc
Code Coverage: None
API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

```
            else if(app.Repairs__c == true && app.Replacement_Parts__c == true){

                app.Service_Amount__c = 7000;

            }

            else if(app.Maintenance_service__c == true){

                app.Service_Amount__c = 2000;

            }

            else if(app.Repairs__c == true){

                app.Service_Amount__c = 3000;

            }

            else if(app.Replacement_Parts__c == true){

```

Testing and Feature Validation

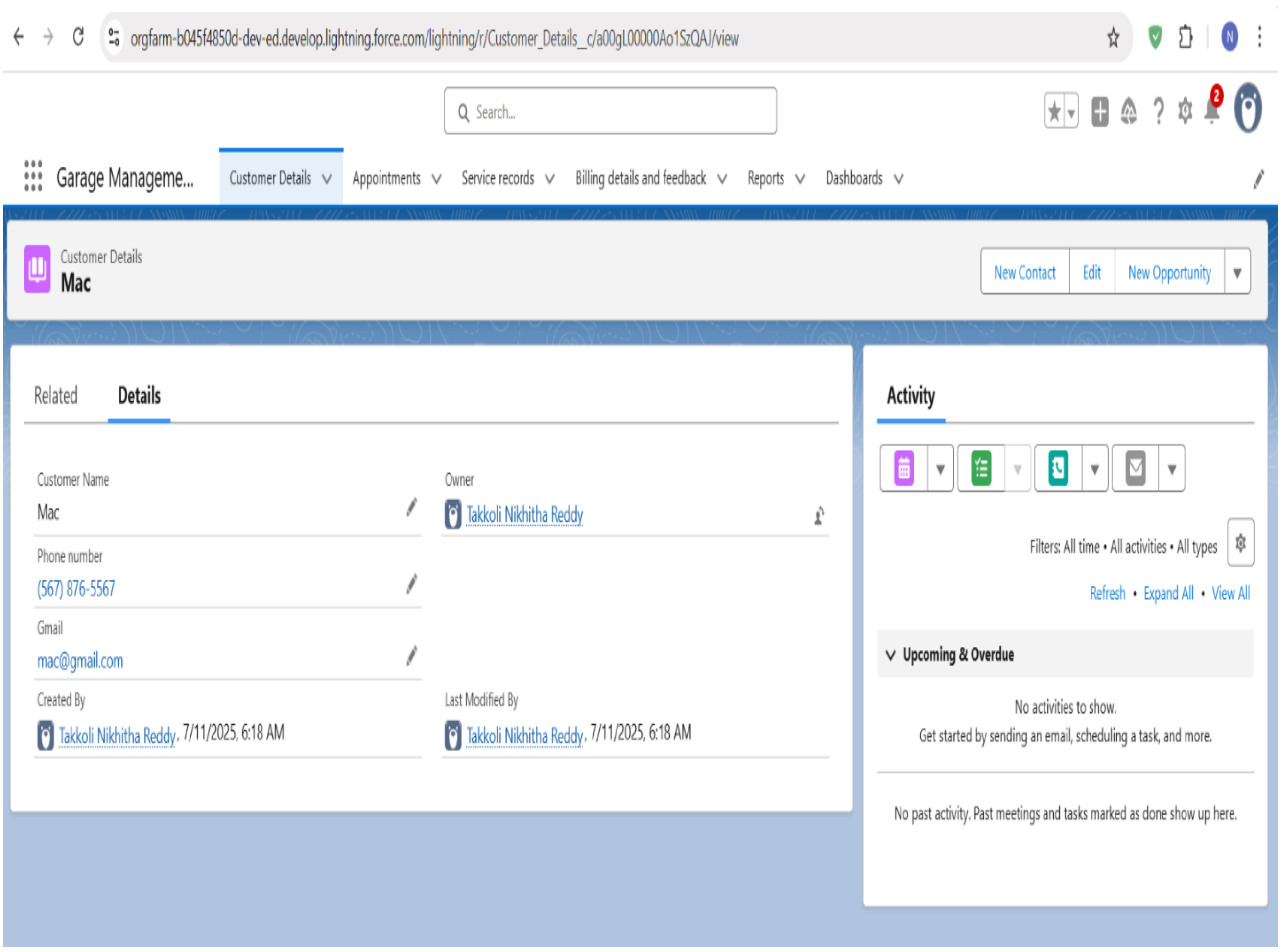
To verify the functionality of all features and ensure business rules were correctly enforced, manual testing was performed on every component:

Tested Features:

- Appointment creation and modification
- Approval Process simulation
- Flow-triggered updates (e.g., Payment confirmation, Service status updates)
- Trigger-based billing automation
- Validation rules (e.g., Vehicle number plate format, Rating between 1–5)
- Reports & Dashboard data visibility

The below screenshots shows the details about the custom objects…

Customer Details


orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/lightning/r/Customer_Details__c/a00gL00000Ao1SzQAJ/view
Search...
Garage Manageme...
Customer Details
Appointments
Service records
Billing details and feedback
Reports
Dashboards
New Contact
Edit
New Opportunity
Related
Details
Activity
Filters: All time • All activities • All types
Refresh • Expand All • View All
Upcoming & Overdue

Customer Details
Mac

Customer Name
Mac

Owner
Takkoli Nikhitha Reddy

Phone number
(567) 876-5567

Gmail
mac@gmail.com

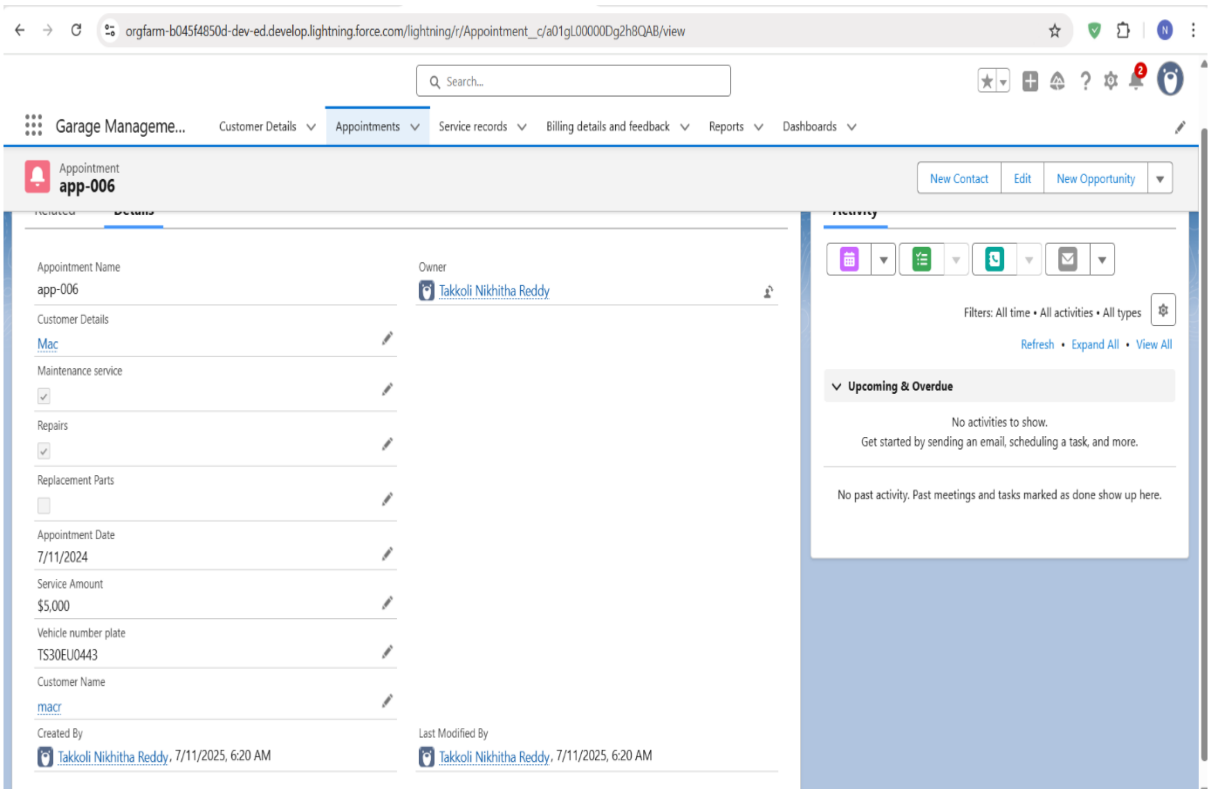
Created By
Takkoli Nikhitha Reddy, 7/11/2025, 6:18 AM

Last Modified By
Takkoli Nikhitha Reddy, 7/11/2025, 6:18 AM

No activities to show.
Get started by sending an email, scheduling a task, and more.

No past activity. Past meetings and tasks marked as done show up here.

Appointments


orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/lightning/r/Appointment__c/a01gL00000Dg2h8QAB/view

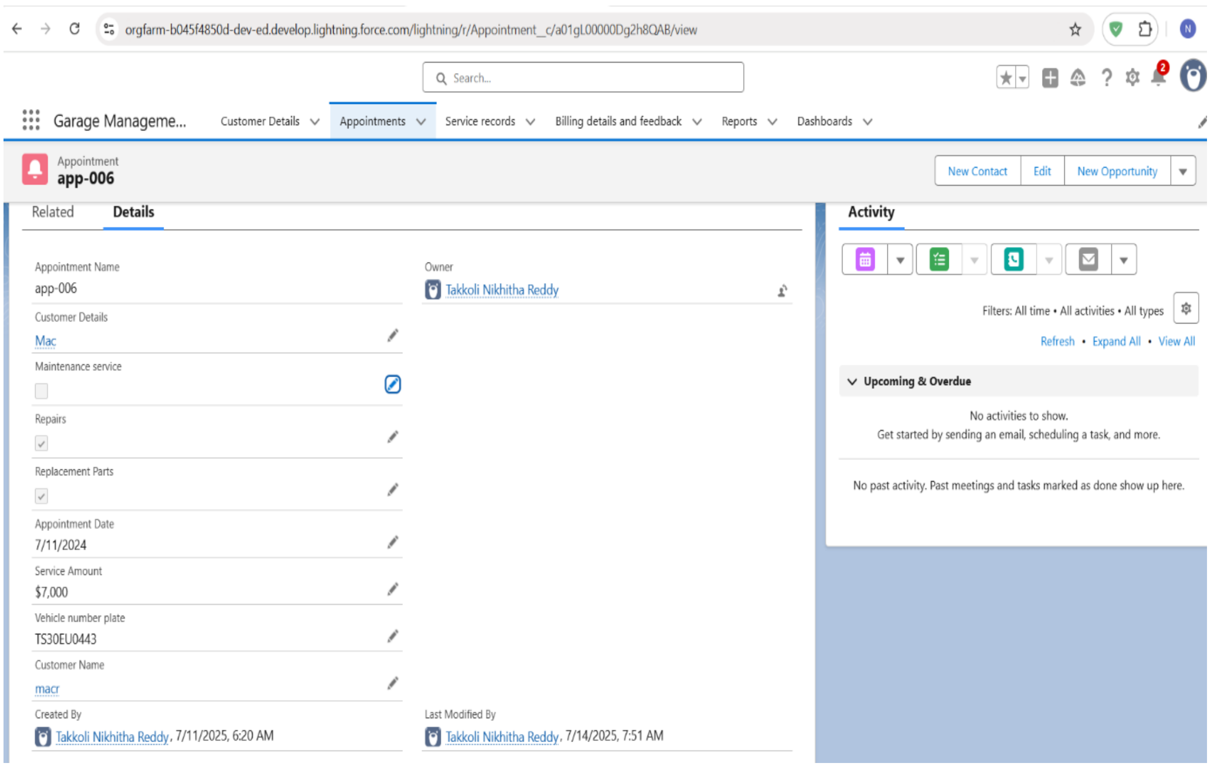
Appointment
app-006

Appointment Name	app-006
Owner	Takkoli Nikhitha Reddy
Customer Details	Mac
Maintenance service	✓
Repairs	✓
Replacement Parts	
Appointment Date	7/11/2024
Service Amount	$5,000
Vehicle number plate	TS30EU0443
Customer Name	macr
Created By	Takkoli Nikhitha Reddy, 7/11/2025, 6:20 AM
Last Modified By	Takkoli Nikhitha Reddy, 7/11/2025, 6:20 AM

Filters: All time • All activities • All types

No activities to show.
Get started by sending an email, scheduling a task, and more.

No past activity. Past meetings and tasks marked as done show up here.


orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/lightning/r/Appointment__c/a01gL00000Dg2h8QAB/view

Appointment
app-006

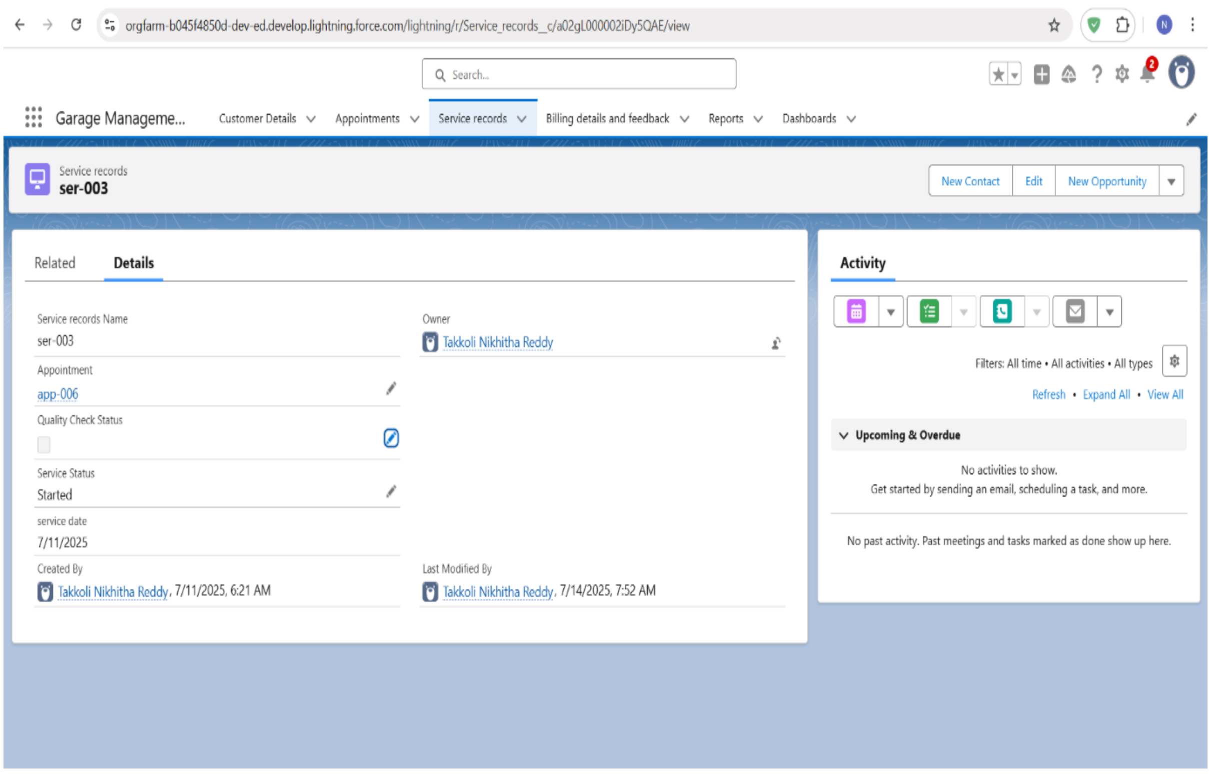
Appointment Name	app-006
Owner	Takkoli Nikhitha Reddy
Customer Details	Mac
Maintenance service	
Repairs	✓
Replacement Parts	✓
Appointment Date	7/11/2024
Service Amount	$7,000
Vehicle number plate	TS30EU0443
Customer Name	macr
Created By	Takkoli Nikhitha Reddy, 7/11/2025, 6:20 AM
Last Modified By	Takkoli Nikhitha Reddy, 7/14/2025, 7:51 AM

Filters: All time • All activities • All types

No activities to show.
Get started by sending an email, scheduling a task, and more.

No past activity. Past meetings and tasks marked as done show up here.

Service Records

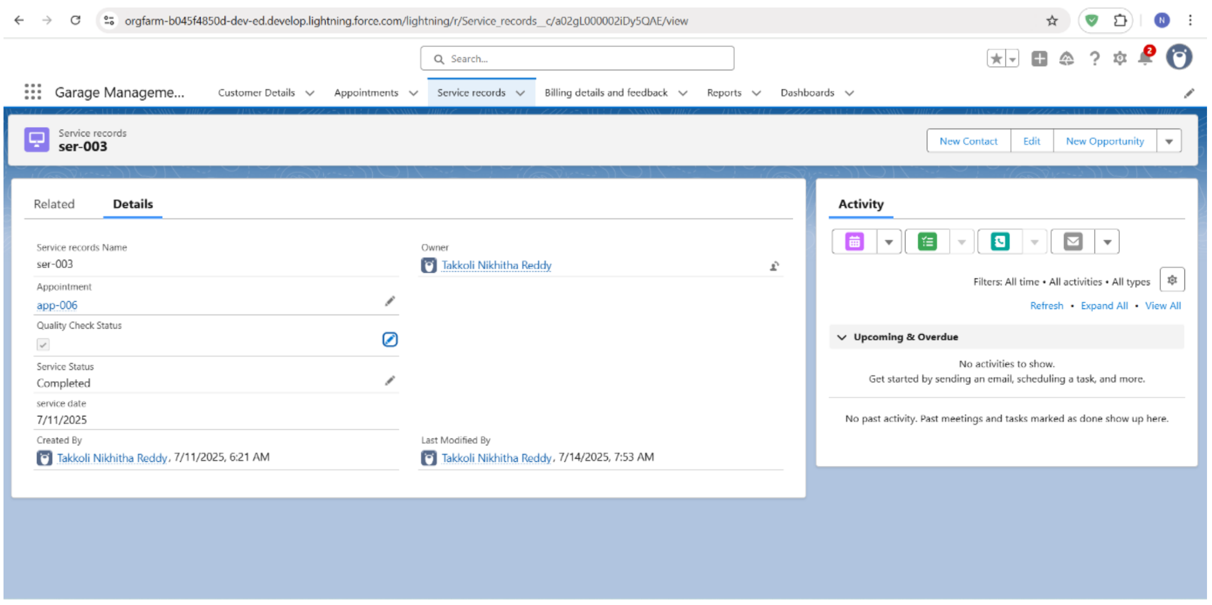


orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/lightning/r/Service_records__c/a02gL000002iDy5QAE/view

Service records
ser-003

Service records Name: ser-003
Owner: Takkoli Nikhitha Reddy
Appointment: app-006
Quality Check Status: ☐
Service Status: Started
service date: 7/11/2025
Created By: Takkoli Nikhitha Reddy, 7/11/2025, 6:21 AM
Last Modified By: Takkoli Nikhitha Reddy, 7/14/2025, 7:52 AM

Garage Manageme...
Customer Details
Appointments
Service records
Billing details and feedback
Reports
Dashboards
New Contact
Edit
New Opportunity
Related
Details
Activity
Filters: All time • All activities • All types
Refresh • Expand All • View All
Upcoming & Overdue
No activities to show.
Get started by sending an email, scheduling a task, and more.
No past activity. Past meetings and tasks marked as done show up here.



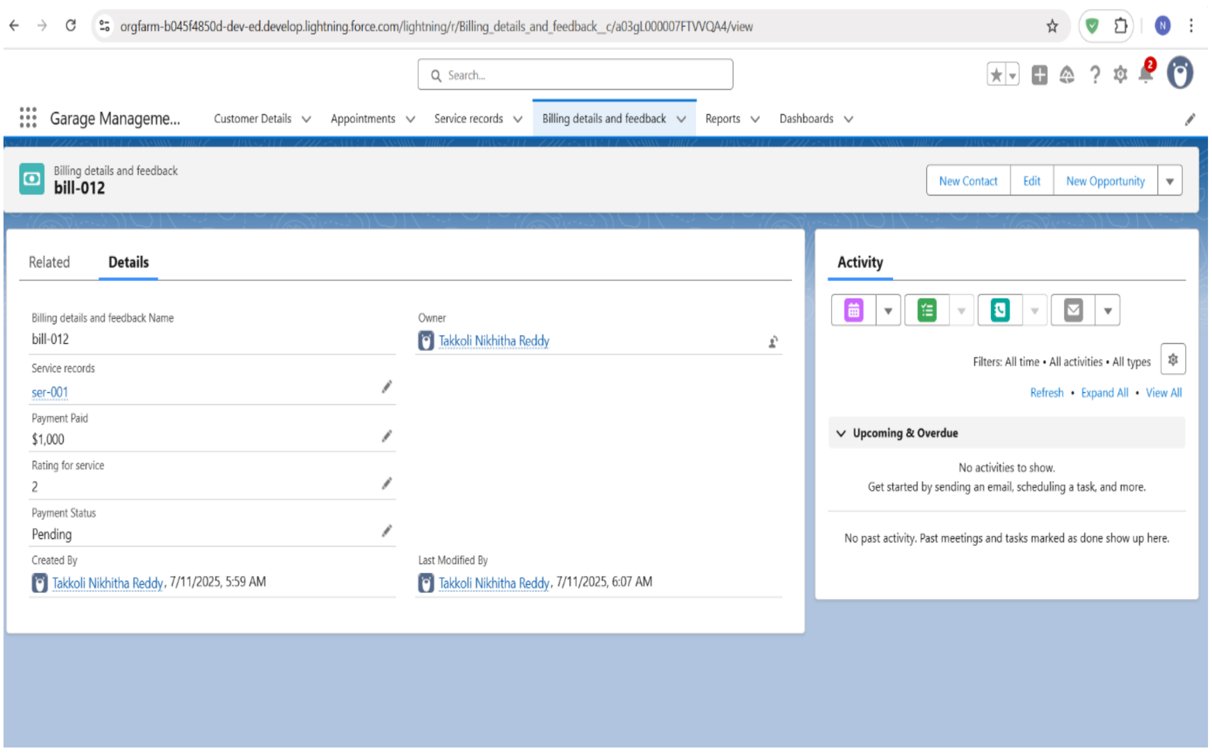
orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/lightning/r/Service_records__c/a02gL000002iDy5QAE/view

Service records
ser-003

Service records Name: ser-003
Owner: Takkoli Nikhitha Reddy
Appointment: app-006
Quality Check Status: ✓
Service Status: Completed
service date: 7/11/2025
Created By: Takkoli Nikhitha Reddy, 7/11/2025, 6:21 AM
Last Modified By: Takkoli Nikhitha Reddy, 7/14/2025, 7:53 AM

Garage Manageme...
Customer Details
Appointments
Service records
Billing details and feedback
Reports
Dashboards
New Contact
Edit
New Opportunity
Related
Details
Activity
Filters: All time • All activities • All types
Refresh • Expand All • View All
Upcoming & Overdue
No activities to show.
Get started by sending an email, scheduling a task, and more.
No past activity. Past meetings and tasks marked as done show up here.

Billing Details and Feedback



orgfarm-b045f4850d-dev-ed.develop.lightning.force.com/lightning/r/Billing_details_and_feedback__c/a03gL000007FTVVQA4/view
Search...
Garage Manageme...
Customer Details
Appointments
Service records
Billing details and feedback
Reports
Dashboards
New Contact
Edit
New Opportunity
Related
Details
Activity
Filters: All time • All activities • All types
Refresh • Expand All • View All
Upcoming & Overdue

Billing details and feedback
bill-012

Billing details and feedback Name
bill-012

Owner
Takkoli Nikhitha Reddy

Service records
ser-001

Payment Paid
$1,000

Rating for service
2

Payment Status
Pending

Created By
Takkoli Nikhitha Reddy, 7/11/2025, 5:59 AM

Last Modified By
Takkoli Nikhitha Reddy, 7/11/2025, 6:07 AM

No activities to show.
Get started by sending an email, scheduling a task, and more.

No past activity. Past meetings and tasks marked as done show up here.

Phase 5: <u>Deployment, Documentation & Maintenance</u>

Deployment Strategy

Since the entire project was developed and tested within a Salesforce Developer Org, no formal deployment to production was required. However, in real-world scenarios, Change Sets or Unmanaged Packages would be used to deploy components from a Sandbox to Production or another org. This ensures that metadata such as objects, fields, flows, Apex classes, and validation rules can be securely and consistently transferred.

System Maintenance and Monitoring

The Garage Management CRM system is designed to be scalable and easy to maintain. Regular maintenance includes reviewing automation flows, checking error logs, validating user access, and updating picklist values or form layouts based on operational changes. Admins can monitor flow executions and Apex errors using Debug Logs, Email Alerts, and the Flow Error Email Notifications provided by Salesforce.

Troubleshooting Approach

A structured troubleshooting method is in place to resolve issues quickly.

Debug logs are used to trace errors in Flows or Apex triggers. Validation failures and automation misfires are diagnosed by reviewing the error messages shown during record creation or updates. In addition, a Log__c custom object can be optionally created in future to store error details and improve system transparency. Documentation of all configuration steps and logic helps with quick issue identification and resolution during handover or enhancement.

Additional Points

1. Custom Lightning App:

Created a Lightning app named Garage Management that groups all custom objects and tabs for easy access.

2. Custom Tabs for Navigation:

Tabs were created for all major custom objects — Customer Details, Appointments, Service Records, Billing & Feedback — improving user navigation.

3. Custom Profiles:

Two profiles were implemented — Manager and Sales Person — each with specific object-level and field-level permissions.

4. Role Hierarchy Setup:

Created a role structure where Manager is at the top, and two Sales Person roles report under it to control data visibility.

5. Validation Rules:

Implemented validation on:

Vehicle Number Format

Feedback Rating

6. Flows for Automation:

Two record-triggered flows:

Automatically update payment and send thank-you emails.

Automatically update service status after quality check.

7. Apex Trigger with Handler Class:

Apex trigger on Appointment__c calculates service charges dynamically using a handler class based on selected service types.

8. Matching & Duplicate Rules:

Configured Matching and Duplicate Rules on Customer_Details__c to prevent duplicate records based on Gmail and Phone Number.

9. Sharing Rules & Private OWD:

Service_Records__c object set to Private, with sharing rules allowing Managers to see records owned by Sales Persons.

10. Custom Report & Dashboard:

Built a report grouped by Rating and Payment Status, and created a line chart dashboard subscribed weekly by managers.

11. User Adoption: creating records

To create a record in the follow objects follow these steps

1. Click on the app launcher located at the left side of the screen.
2. Search for "**Garage Management**" and click on it.
3. Click on the "**Consumer details** tab".
4. Click on new and fill the details, and click save.

Now, Create the Appointment Record

1. Click on the "**Appointment** tab".
2. Enter the customer details as created, while entering Appointment Date enter the date less than the created date.
3. Match the validation while entering the vehicle number plate.
4. Select the services you need.
5. Click on save to see the Service Amount.

Now, Create a service Record

1. Click on the "**Service record** tab".
2. Enter the Appointment, and started is selected as default.
3. Click on save.
4. Open the record and click on Quality check status as true and click on save.

Conclusion

The Garage Management CRM project successfully streamlined vehicle service operations using the Salesforce platform.

Key modules like customer handling, appointment booking, service tracking, billing, and feedback were implemented with automation, validation, and role-based access.

Custom objects, flows, triggers, and reports were configured to improve efficiency, accuracy, and user experience.

The solution reduces manual work, enhances customer communication, and provides real-time visibility to managers.

With scope for future enhancements like chatbot integration or AI-based service suggestions, the system is scalable and ready for real-world adoption.