ANNAMACHARYA INSTITUTE OF TECHNOLOGY AND SCIENCES, KADAPA

Project Report on

SALESFORCE – Garage Management system

BY

MITTA PAVITHRA

(23HM5A0422)

pavithrareddymitta@gmail.com

Table of Contents

Project Overview

The Garage Management System (GMS) is a CRM application built on Salesforce to handle vehicle service workflows such as customer management, appointment bookings, service tracking, billing, and feedback. The system automates repetitive tasks, improves record accuracy, and enhances customer communication.

The Garage Management System (GMS) is a cloud-based Customer Relationship Management (CRM) application built using the Salesforce platform. It is designed to automate and digitize the daily operations of automotive service centers and repair garages. Traditional garage operations involve heavy paperwork, manual scheduling, unstructured billing, and poor visibility into customer service history. This project aims to overcome these limitations by offering a centralized and automated platform that enhances efficiency, data accuracy, and customer satisfaction.

The system manages the complete vehicle service lifecycle — from customer registration and appointment booking to service tracking, billing, and customer feedback. It supports multiple roles such as Admin, Manager, and Technician with controlled access permissions to ensure secure and organized data flow. By leveraging Salesforce features such as custom objects, flows, validation rules, Apex classes, and dashboards, GMS ensures streamlined workflows, reduces manual errors, and enables informed decision-making through real-time reports.

Key business challenges addressed by GMS include:

- **Efficient appointment scheduling and service tracking:** Staff can schedule, manage, and monitor appointments and service records for each customer in real-time, reducing wait times and improving resource allocation.
- **Automated service cost calculations:** Based on selected services (e.g., maintenance, repairs, or part replacements), the system automatically calculates service costs through Apex logic, ensuring billing accuracy and transparency.
- **Real-time performance insights:** Custom dashboards and dynamic reports allow managers to visualize service trends, revenue generation, and customer feedback, empowering data-driven decision-making.
- **Enhanced customer engagement:** Automated email notifications and follow-ups, including personalized thank-you messages post-payment, help in building long-term customer relationships and encouraging return visits.

Project Objectives

The Garage Management System was developed with the aim of modernizing and automating the daily operations of an automobile service center. Below are the elaborated objectives that guided the development of this Salesforce-based solution:

- **Digitize Service Operations:** Eliminate manual paperwork by digitizing all customer, appointment, and service records using custom Salesforce objects. This allows for faster record keeping, easier access to historical data, and better tracking of each customer interaction.
- **Enable Real-Time Tracking:** Empower garage staff and managers to view and update customer appointments, service status, quality checks, and billing information in real-time from any device. This enhances coordination and operational transparency.
- **Minimize Errors Through Automation:** Reduce manual calculation errors and operational oversights by implementing automated Flows, Apex triggers, and validation rules. These automations ensure that service charges, billing, and data updates are accurate and consistent.
- **Enhance Data Security & Access Control:** Define precise role hierarchies and profiles to manage user permissions and ensure that sensitive information is only accessed by authorized personnel. Sharing rules, OWD settings, and field-level security further enforce safe data practices.
- **Provide Actionable Insights:** Use Salesforce Reports and Dashboards to convert raw service and billing data into actionable metrics. These help managers track performance trends, identify bottlenecks, and make strategic decisions.
- **Boost Customer Satisfaction:** Enhance customer engagement through timely service updates, automated thank-you emails, and feedback collection mechanisms. This creates a professional and responsive service experience.
- **Ensure Scalability and Maintainability:** Design the system with modularity and scalability in mind, allowing for the addition of new features such as AI-based recommendations, chatbot interfaces, or external integrations in future versions.
- **Support Role-Based Usability:** Tailor the user experience for different roles such as Managers, Salespersons, and Admins, ensuring that each user only sees relevant data and has access to the tools they need.

Phase 1: Requirement Analysis & Planning

Understanding Business Requirements:

- The garage requires a way to register customer details, schedule appointments, and track service types.
- Payments must be calculated automatically based on selected services.
- Managers need an overview of operations via reports and dashboards.
- Feedback and rating collection should be enabled for quality control.

Project Scope and Objectives:

- Create custom Salesforce objects to hold structured data for: Customer Details, Appointments, Service Records, Billing, and Feedback.
- Implement logic using Apex Triggers and Flows to automate payment calculations and update service statuses dynamically.
- Enable secure data access by defining roles (Manager, Salesperson, Admin) and assigning object-level and field-level permissions.
- Build relationships among objects to support a clear, hierarchical data model suitable for future reporting.

Data & Security Model Design:

An efficient and secure data model was essential to support reporting, automation, and access control. The object relationships were designed as follows:

- **Customer Details** (Parent) → **Appointment** (Child)
- **Appointment** (Parent) → **Service Records** (Child)
- **Service Records** (Parent) → **Billing & Feedback** (Child)

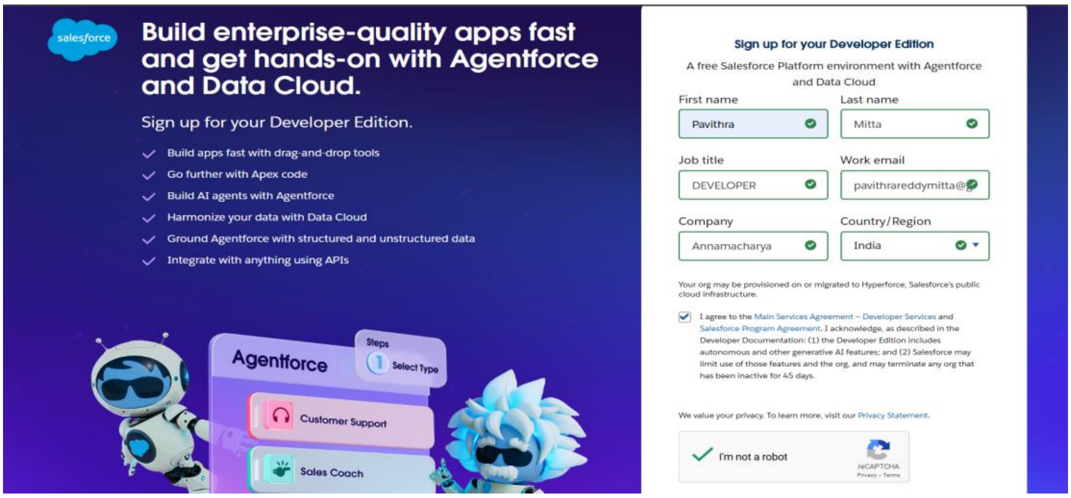
This lookup-based structure ensures data integrity while maintaining the flexibility to scale. Security was enforced using the following Salesforce mechanisms:

- Profile-based access control for defining object permissions
- Role Hierarchies to manage record visibility across different roles
- Organization-Wide Defaults (OWD) and Sharing Rules to control access to sensitive records, such as limiting access to service and billing records only to assigned staff

Phase 2: Salesforce Backend Development & Configurations

Environment Setup:

- Developer account created via [https://developer.salesforce.com/]
- Basic org setup and branding applied.



Build enterprise-quality apps fast and get hands-on with Agentforce and Data Cloud.

Sign up for your Developer Edition.

- ✓ Build apps fast with drag-and-drop tools
- ✓ Go further with Apex code
- ✓ Build AI agents with Agentforce
- ✓ Harmonize your data with Data Cloud
- ✓ Ground Agentforce with structured and unstructured data
- ✓ Integrate with anything using APIs

Sign up for your Developer Edition

A free Salesforce Platform environment with Agentforce and Data Cloud

First name Pavithra	Last name Mitta
Job title DEVELOPER	Work email pavithrareddymitta@
Company Annamacharya	Country/Region India

Your org may be provisioned on or migrated to Hyperforce, Salesforce's public cloud infrastructure.

☑ I agree to the Main Services Agreement – Developer Services and Salesforce Program Agreement. I acknowledge, as described in the Developer Documentation: (1) the Developer Edition includes autonomous and other generative AI features; and (2) Salesforce may limit use of those features and the org, and may terminate any org that has been inactive for 45 days.

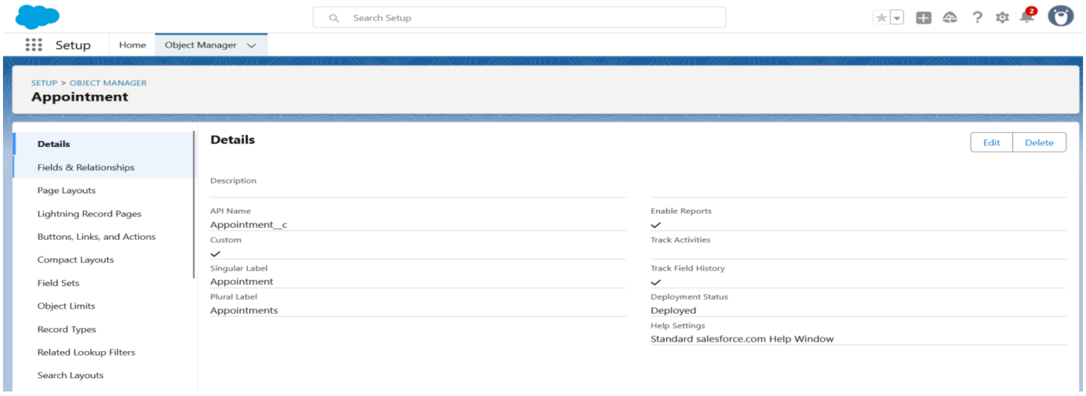
We value your privacy. To learn more, visit our Privacy Statement.

✓ I'm not a robot

Fig: Signing Up for Developer Org

Custom Objects Created:

- Customer Details
- Appointment (Auto-number format: app-{000})
- Service Records (Auto-number format: ser-{000})
- Billing Details and Feedback (Auto-number format: bill-{000})



Appointment

Details

Description

API Name Appointment__c	Enable Reports ✓
Custom ✓	Track Activities
Singular Label Appointment	Track Field History ✓
Plural Label Appointments	Deployment Status Deployed
	Help Settings Standard salesforce.com Help Window

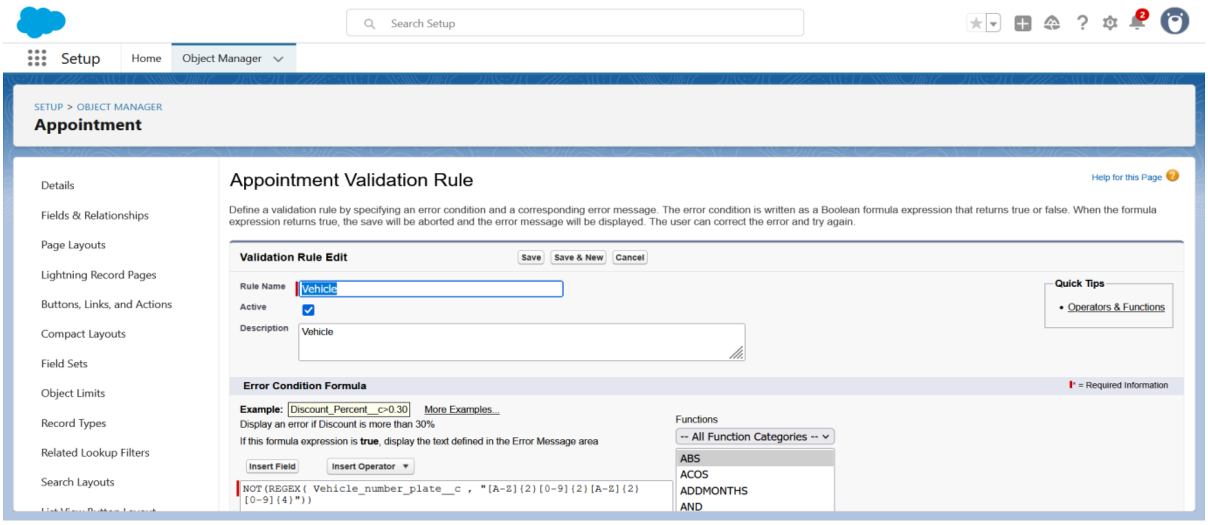
Fig: Object Creation

Field Configurations:

- Customer Details: Phone (Phone), Gmail (Email)
- Appointment: Date, Vehicle Plate (Text/Regex), Service Checkboxes (Maintenance, Repair, Replacement)
- Service Records: Quality Check Status (Checkbox), Service Status (Picklist)
- Billing: Payment Paid (Currency), Rating (Text, Required), Payment Status (Picklist)

Validation Rules:

- Vehicle Plate Format (e.g., AP01AB1234)
- Rating (Only values 1–5)



Appointment Validation Rule

Define a validation rule by specifying an error condition and a corresponding error message. The error condition is written as a Boolean formula expression that returns true or false. When the formula expression returns true, the save will be aborted and the error message will be displayed. The user can correct the error and try again.

Validation Rule Edit

Rule Name: Vehicle

Active: ☑

Description: Vehicle

Error Condition Formula

= Required Information

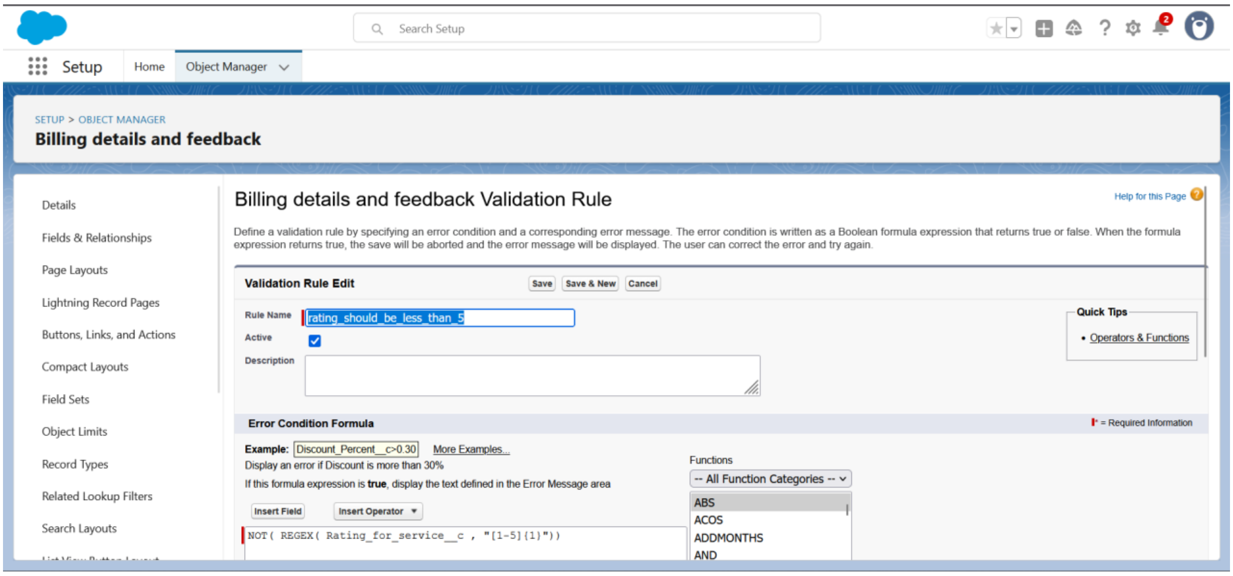
Example: Discount_Percent__c>0.30 More Examples...

Display an error if Discount is more than 30%

If this formula expression is **true**, display the text defined in the Error Message area

```
NOT(REGEX( Vehicle_number_plate__c , "[A-Z]{2}[0-9]{2}[A-Z]{2}
[0-9]{4}"))
```

Fig: Validation Rule for Vehicle Plate Format



Billing details and feedback Validation Rule

Define a validation rule by specifying an error condition and a corresponding error message. The error condition is written as a Boolean formula expression that returns true or false. When the formula expression returns true, the save will be aborted and the error message will be displayed. The user can correct the error and try again.

Validation Rule Edit

Rule Name: rating_should_be_less_than_5

Active: ☑

Description:

Error Condition Formula

= Required Information

Example: Discount_Percent__c>0.30 More Examples...

Display an error if Discount is more than 30%

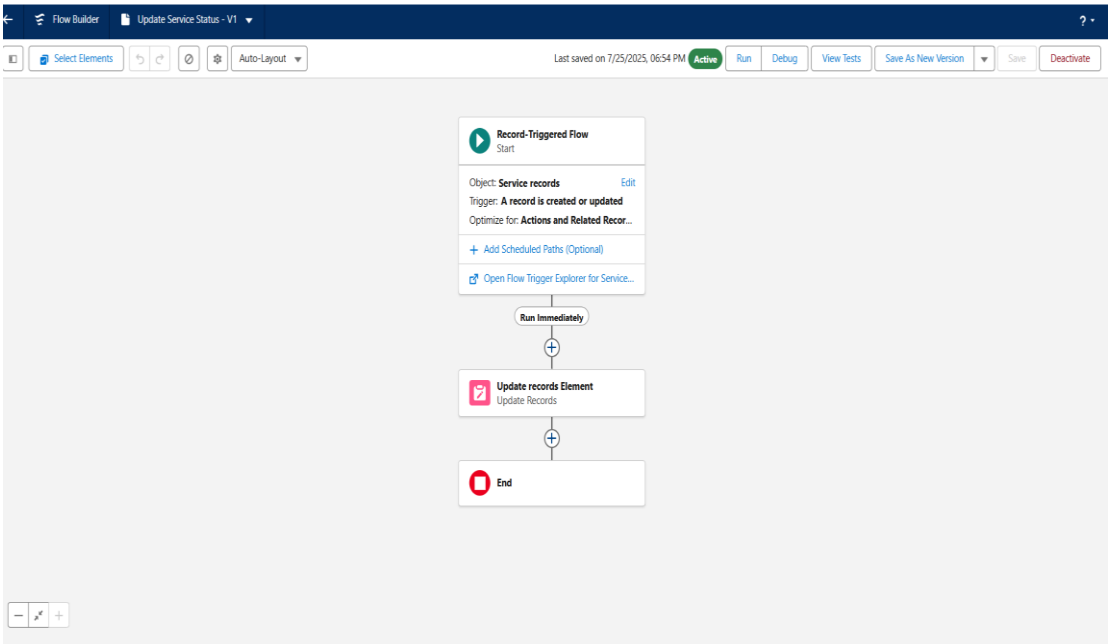
If this formula expression is **true**, display the text defined in the Error Message area

```
NOT( REGEX( Rating_for_service__c , "[1-5]{1}"))
```

Fig: Validation Rule for Rating

Automation:

- **Flows:**
  - Update Payment after Status = Completed
  - Update Service Status after Quality Check is true



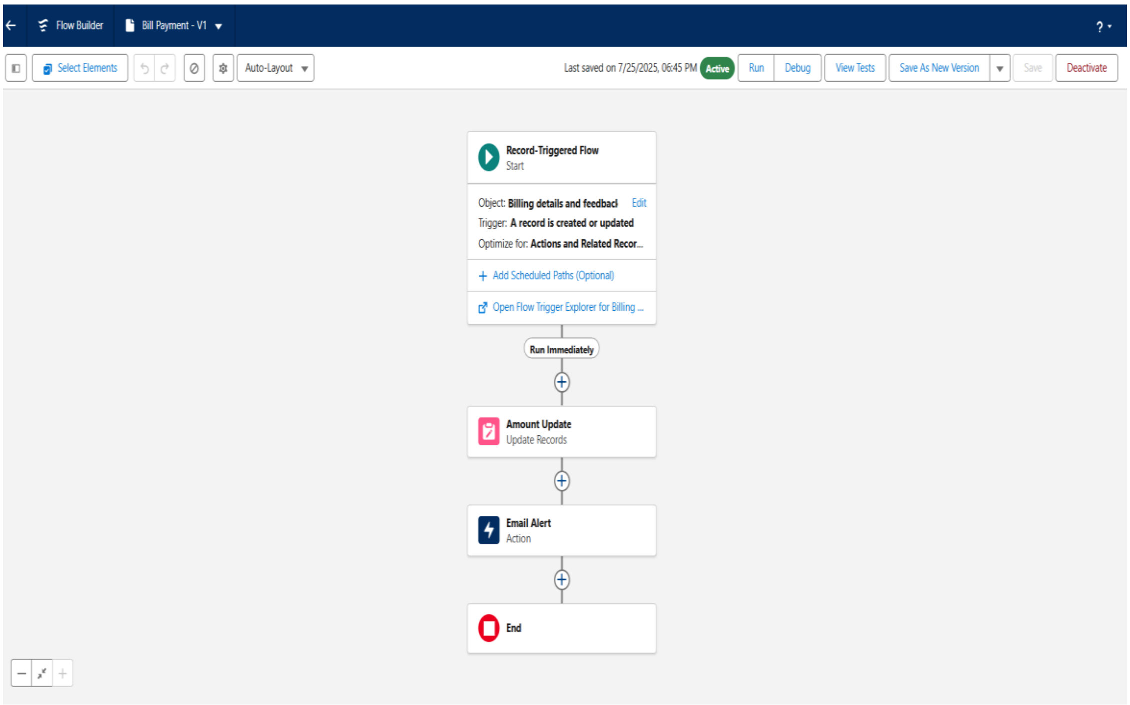
Last saved on 7/25/2025, 06:54 PM

Object: Service records

Trigger: A record is created or updated

Optimize for: Actions and Related Recor...

Fig: Service Record Flow



Last saved on 7/25/2025, 06:45 PM

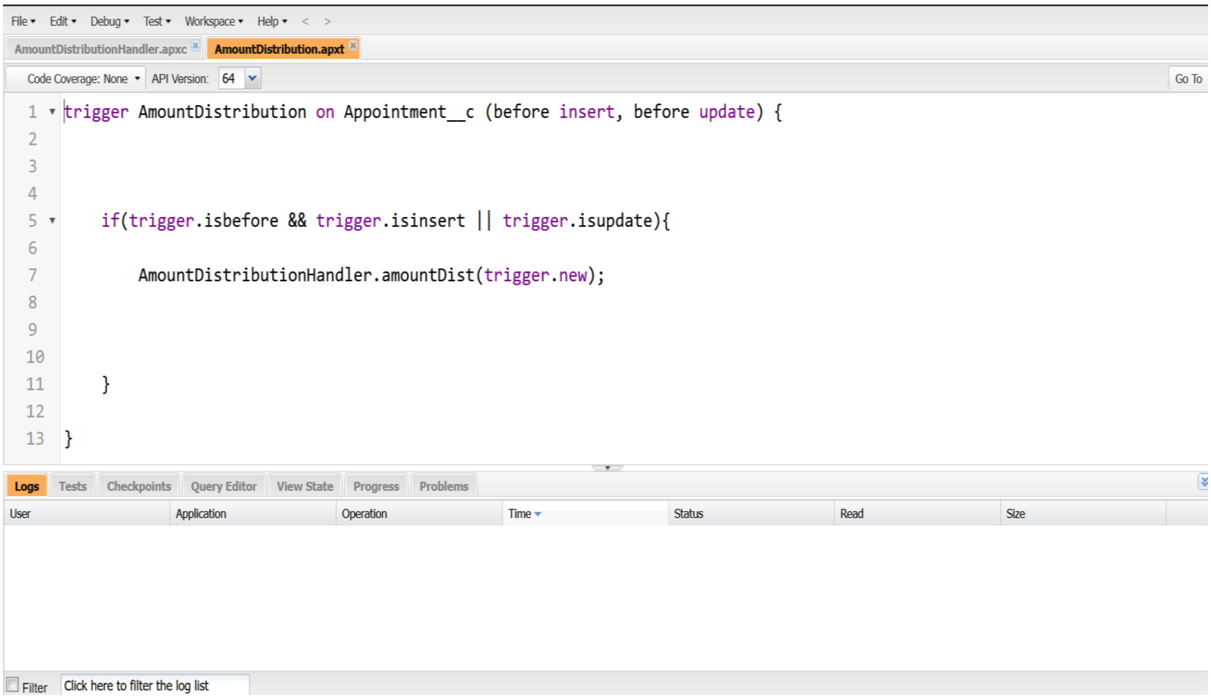
Object: Billing details and feedbacl

Trigger: A record is created or updated

Optimize for: Actions and Related Recor...

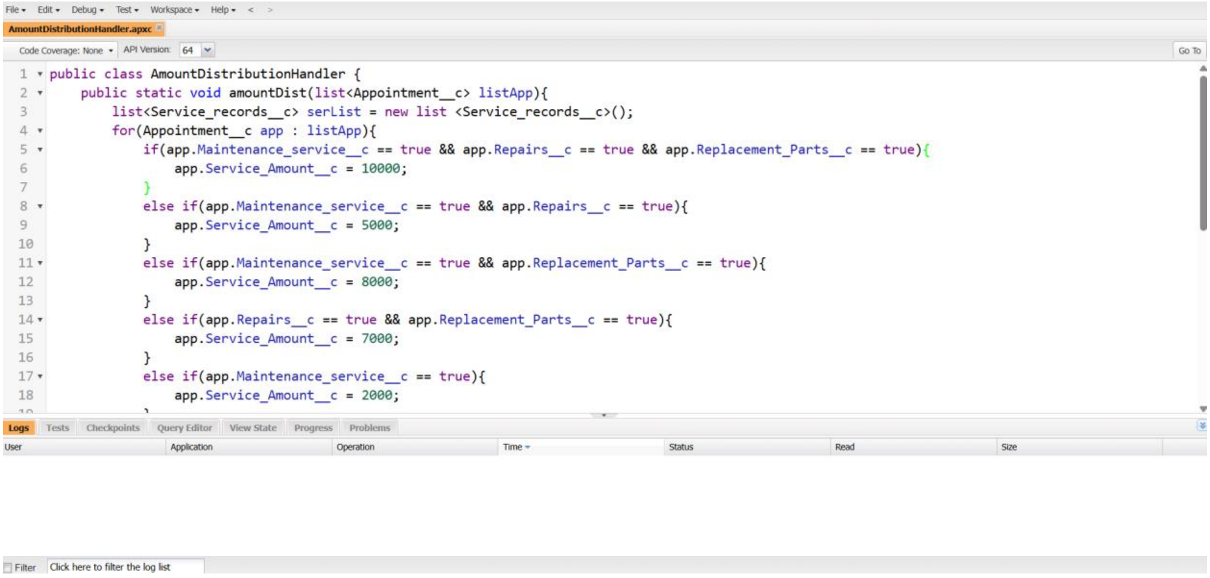
Fig: Billing Amount Flow

- **Apex Trigger:**
  - Trigger: AmountDistribution (before insert/update on Appointment)
  - Handler: AmountDistributionHandler class calculating service amount.



```
trigger AmountDistribution on Appointment__c (before insert, before update) {



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

        AmountDistributionHandler.amountDist(trigger.new);



    }

}
```

Fig: Amount Distribution Trigger



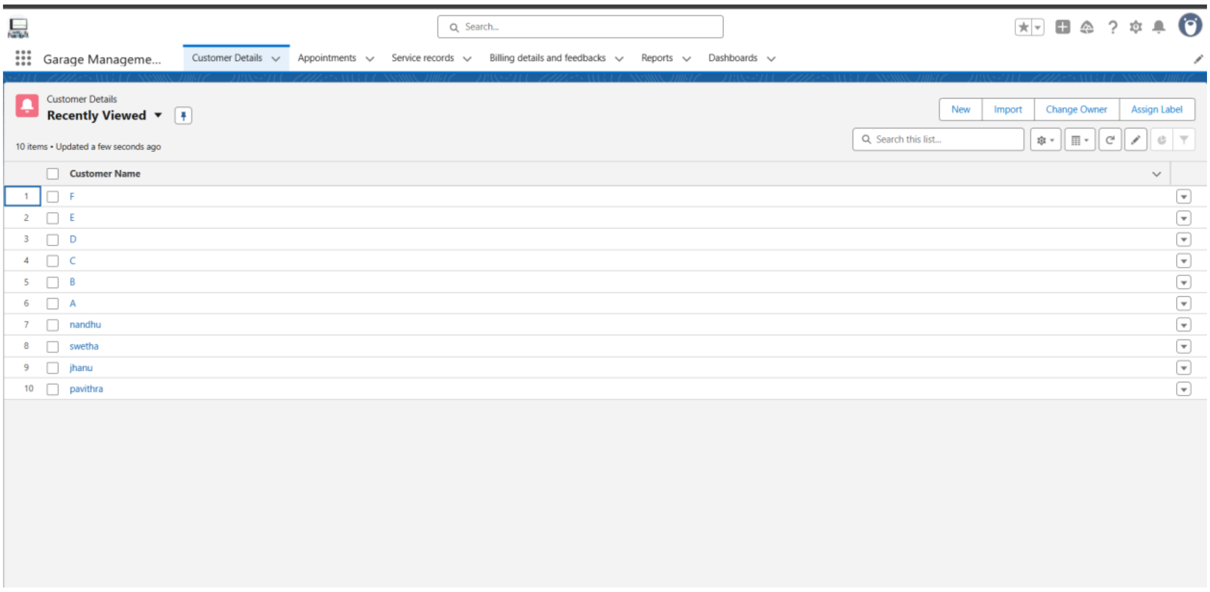
```
public class AmountDistributionHandler {
    public static void amountDist(list<Appointment__c> listApp){
        list<Service_records__c> serList = new list <Service_records__c>();
        for(Appointment__c app : listApp){
            if(app.Maintenance_service__c == true && app.Repairs__c == true && 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;
            }
            else if(app.Maintenance_service__c == true && app.Replacement_Parts__c == true){
                app.Service_Amount__c = 8000;
            }
            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;
```

Fig: Amount Distribution Handler Class

Phase 3: UI/UX Development & Customization

Lightning App Creation:

- App Name: Garage Management Application
- Tabs: Customer Details, Appointments, Service Records, Billing, Reports, Dashboards.



Customer Details
Recently Viewed

10 items • Updated a few seconds ago

	Customer Name
1	F
2	E
3	D
4	C
5	B
6	A
7	nandhu
8	swetha
9	jhanu
10	pavithra

Fig: Garage Management Application

Tabs Setup:

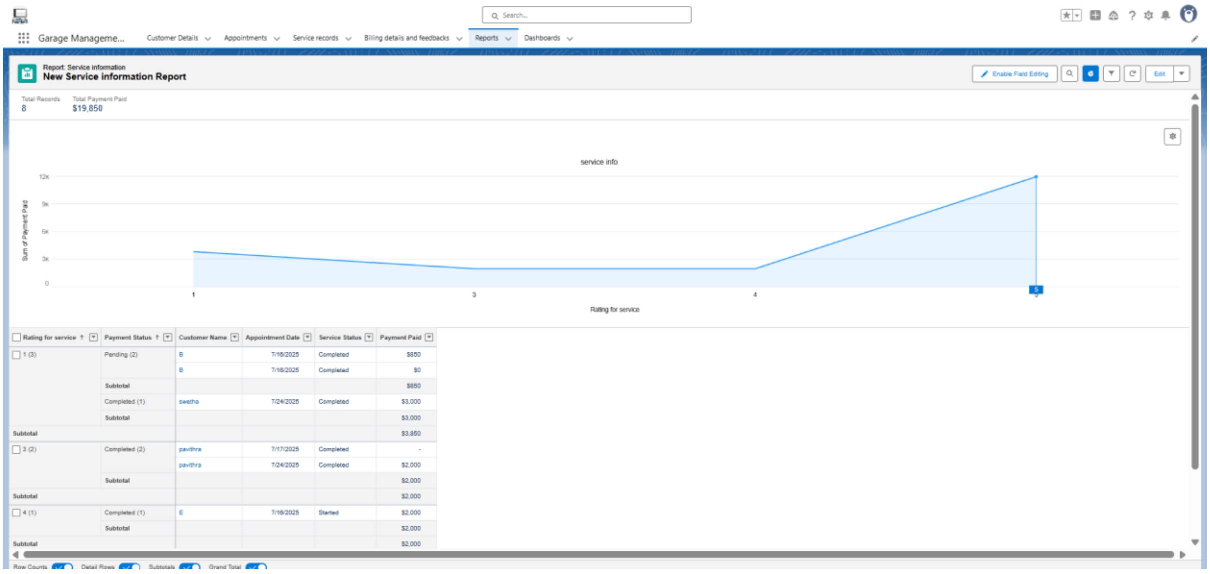
- Created custom tabs for all objects with default tab styles

Page Layouts & Dynamic Forms:

- Custom layouts for each object
- Required fields and checkboxes placed prominently
- Dynamic visibility used for service selections

Reports:

- Custom Report Type: Customer → Appointment → Service Record → Billing
- Fields: Appointment Date, Payment Paid, Service Status
- Grouped by: Rating of service, Payment Status



Report: Service information

New Service information Report

Total Records 8

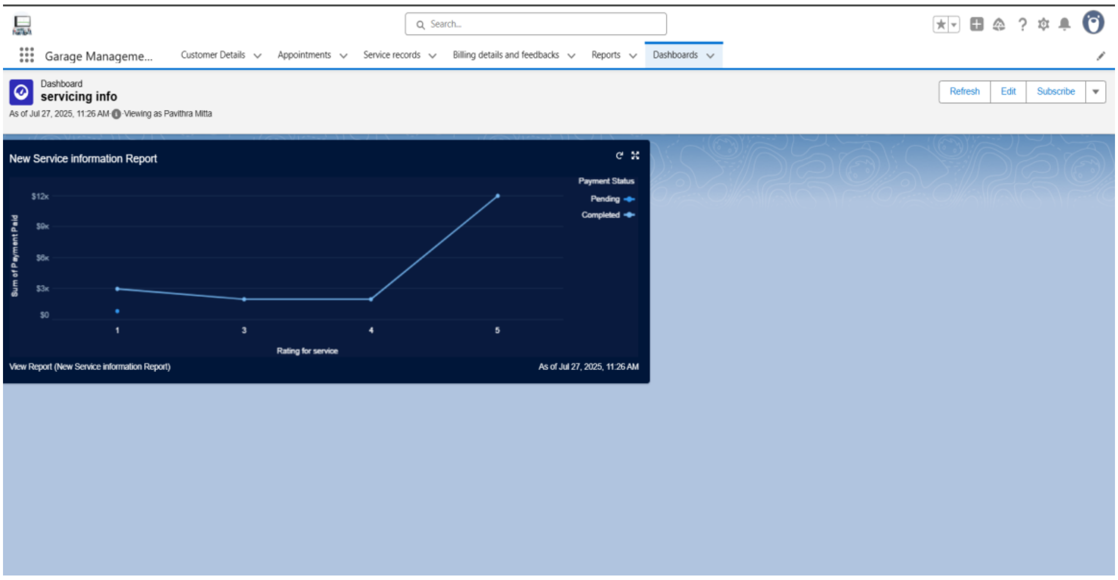
Total Payment Paid $19,850

Rating for service	Payment Status	Customer Name	Appointment Date	Service Status	Payment Paid
□ 1 (3)	Pending (2)	B	7/16/2025	Completed	$850
		B	7/16/2025	Completed	$0
	Subtotal				$850
	Completed (1)	swetha	7/24/2025	Completed	$3,000
	Subtotal				$3,000
Subtotal					$3,850
□ 3 (2)	Completed (2)	pavithra	7/17/2025	Completed	-
		pavithra	7/24/2025	Completed	$2,000
	Subtotal				$2,000
Subtotal					$2,000
□ 4 (1)	Completed (1)	E	7/16/2025	Started	$2,000
	Subtotal				$2,000
Subtotal					$2,000

Fig: Report

Dashboards:

- Folder: Service Rating Dashboard
- Chart: Line chart displaying ratings and payments over time
- Subscription: Weekly report email every Monday



Dashboard

servicing info

As of Jul 27, 2025, 11:26 AM · Viewing as Pavithra Mitta

New Service information Report

View Report (New Service information Report)

As of Jul 27, 2025, 11:26 AM

Fig: Dashboards

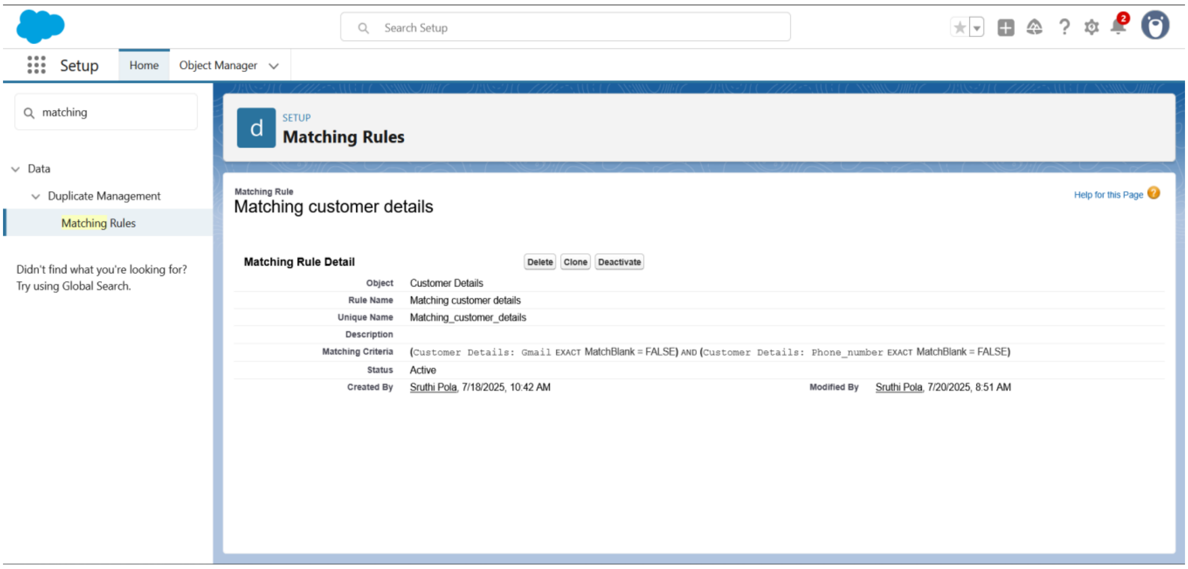
Phase 4: Data Migration, Testing & Security

Data Migration:

- Used Salesforce UI for manual data entry (10+ records per object)

Matching & Duplicate Rules:

- Matching Rule: Gmail & Phone for Customer Details
- Duplicate Rule: Triggered on record creation

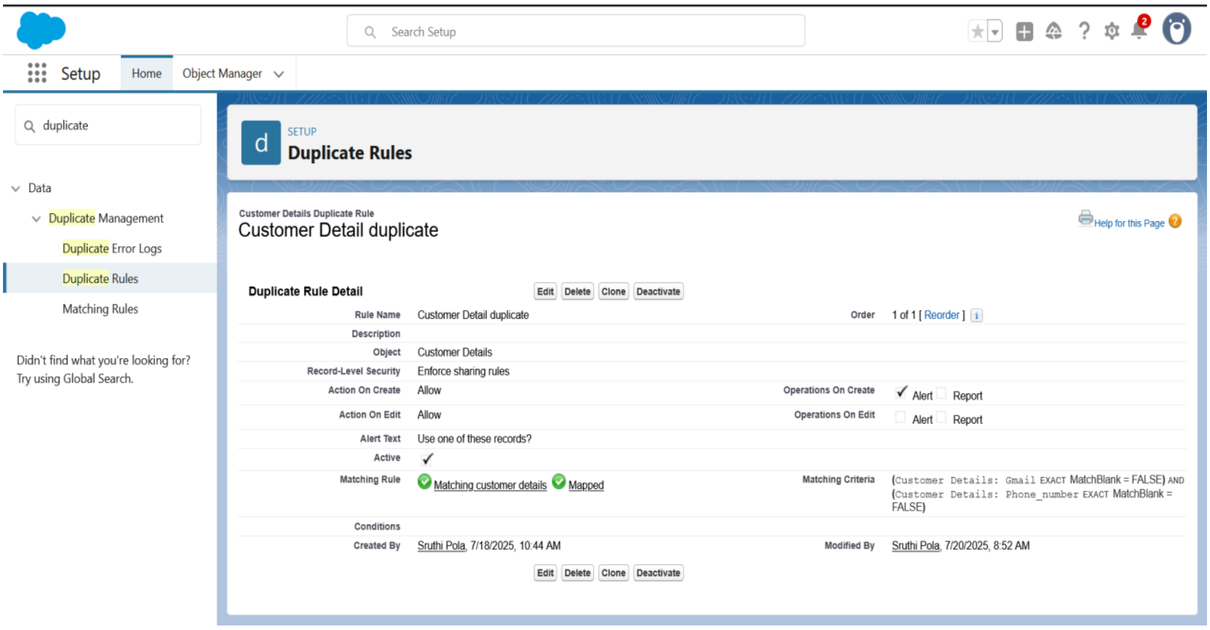


Matching Rule
Matching customer details

Matching Rule Detail

Object	Customer Details
Rule Name	Matching customer details
Unique Name	Matching_customer_details
Description	
Matching Criteria	(Customer Details: Gmail EXACT MatchBlank = FALSE) AND (Customer Details: Phone_number EXACT MatchBlank = FALSE)
Status	Active
Created By	Sruthi Pola, 7/18/2025, 10:42 AM
Modified By	Sruthi Pola, 7/20/2025, 8:51 AM

Fig: Matching Customer Details



Customer Details Duplicate Rule
Customer Detail duplicate

Duplicate Rule Detail

Rule Name	Customer Detail duplicate	Order	1 of 1 [Reorder]
Description			
Object	Customer Details		
Record-Level Security	Enforce sharing rules		
Action On Create	Allow	Operations On Create	☑ Alert ☐ Report
Action On Edit	Allow	Operations On Edit	☐ Alert ☐ Report
Alert Text	Use one of these records?		
Active	✓		
Matching Rule	✓ Matching customer details ✓ Mapped	Matching Criteria	(Customer Details: Gmail EXACT MatchBlank = FALSE) AND (Customer Details: Phone_number EXACT MatchBlank = FALSE)
Conditions			
Created By	Sruthi Pola, 7/18/2025, 10:44 AM	Modified By	Sruthi Pola, 7/20/2025, 8:52 AM

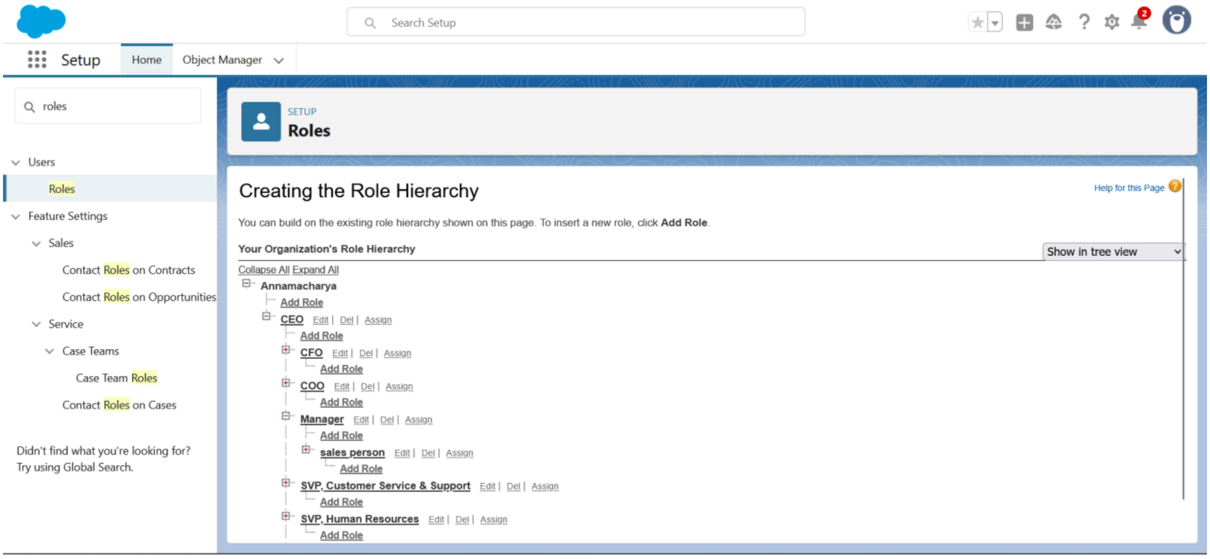
Fig: Customer Detail Duplicate

Field History Tracking:

- Enabled on all custom objects

Security Settings:

- OWD: Private for Service Records
- Sharing Rule: Salesperson → Manager (Read/Write)
- Public Group: Sales Team
- Profiles: Manager, Salesperson with appropriate permissions
- Roles: CEO > Manager > Salesperson (Role hierarchy)



Didn't find what you're looking for? Try using Global Search.

Creating the Role Hierarchy

You can build on the existing role hierarchy shown on this page. To insert a new role, click **Add Role**.

Your Organization's Role Hierarchy

```
Annamacharya
  Add Role
  CEO  Edit | Del | Assign
    Add Role
    CFO  Edit | Del | Assign
      Add Role
    COO  Edit | Del | Assign
      Add Role
    Manager  Edit | Del | Assign
      Add Role
      sales person  Edit | Del | Assign
        Add Role
    SVP, Customer Service & Support  Edit | Del | Assign
      Add Role
    SVP, Human Resources  Edit | Del | Assign
      Add Role
```

Fig: Roles

Testing & Q&A:

- Manual tests conducted for each feature.
- Sample Inputs and Expected Outputs documented for:
  - Record creation
  - Validation errors
  - Flow updates
  - Trigger execution

Phase 5: Deployment, Documentation & Maintenance

Deployment Strategy:

Since the Garage Management System was developed within a Salesforce Developer Org, traditional deployment processes such as change sets or managed packages were not required during the initial build. However, for production or sandbox deployments in a real-world environment, the system is designed to support structured deployment using Salesforce Change Sets, Unmanaged Packages, or Salesforce DevOps Center tools. These options ensure safe migration of metadata components, including custom objects, flows, Apex classes, profiles, and validation rules, from one org to another. A well-documented deployment strategy ensures future scalability and maintainability of the solution in enterprise environments.

System Maintenance:

To ensure the application continues to function as expected, routine system monitoring and maintenance practices have been put in place. Debug logs are enabled in the org to help monitor flow executions, Apex errors, and data updates. These logs assist developers and administrators in troubleshooting issues quickly. Field history tracking is also enabled for critical objects to monitor changes made by different users. Scheduled reviews of flows and triggers are recommended to optimize performance over time.

Documentation & Training:

Comprehensive documentation was created throughout the project to support future users and administrators of the application. This includes:

- **Labeled Screenshots:** Each step of the configuration, including object creation, field setup, validation rules, flows, and dashboards, has been documented with screenshots for easy reference.
- **Object Relationship Diagram:** A clear visual representation of the relationships between objects (Customer → Appointment → Service Record → Billing & Feedback) helps users and developers understand data flow.

- **Testing Summary:** All major test cases for record creation, trigger functionality, flow automation, and validation rule enforcement have been compiled and summarized.
- **Admin Training Material:** A concise guide is included to support new Salesforce users or garage employees in navigating the app, performing updates, and accessing reports.

Conclusion

The Garage Management System brings structure and efficiency to daily garage tasks. It automates appointment handling, payments, and improves customer satisfaction through real-time updates.

The Garage Management System (GMS) built using Salesforce CRM offers a robust, cloud-based solution to the complex needs of automobile service centers. Through the digital transformation of garage operations, the system effectively replaces traditional, error-prone manual processes with automated and structured workflows.

From the moment a customer's details are captured, GMS ensures seamless handling of appointment bookings, service type selection, billing, and post-service follow-up. The use of validation rules minimizes incorrect data entry, while Apex triggers and flows automate the calculation of service charges and update statuses without manual effort.

By leveraging Salesforce's built-in capabilities such as object customization, field-level security, and real-time analytics, this system empowers garage staff to focus more on customer service rather than administrative tasks. It also provides managers with valuable operational insights through custom reports and dashboards, helping in identifying trends, optimizing resources, and improving decision-making.

In addition, the system enhances customer satisfaction by sending personalized thank-you emails upon successful payment and collecting structured feedback to improve service quality. Security is maintained via well-structured role hierarchies and sharing rules, ensuring that sensitive information is accessible only to the right personnel.

Overall, this Garage Management System proves to be a scalable, secure, and maintainable solution that not only supports current business needs but also lays a strong foundation for future enhancements like AI integration, chatbot support, and mobile accessibility. It successfully showcases how Salesforce can be utilized to create an industry-specific CRM that improves efficiency, accuracy, and customer engagement. It simplifies appointment booking, cost computation, service tracking, and billing—while enhancing customer engagement through emails and feedback mechanisms. The system is secure, user-friendly, and scalable.

Future Enhancements

- Integration with WhatsApp for appointment reminders
- AI-based recommendation engine for service packages
- Voice-enabled service requests using Einstein Voice
- Mobile Lightning App for customers