

GARAGE MANAGEMENT SYSTEM

By

Aryan Singh

as.aryansingh2004@gmail.com

Abstract

The **Garage Management System (GMS)** is designed to streamline operations within auto repair shops, helping businesses manage customer data, appointments, services, billing, and feedback efficiently. By leveraging Salesforce's powerful platform, GMS ensures a seamless workflow and enhances customer satisfaction by providing easy access to essential information, automating processes, and improving communication. The system includes custom objects like Customer Details, Appointments, Service Records, and Billing Feedback, along with automated flows, validation rules, and reports. The integration of Salesforce's robust features ensures real-time updates, accurate billing, and automated alerts, making GMS a valuable tool for any garage looking to optimize its services and customer interactions. This project highlights the importance of digital transformation in garage management by improving efficiency and customer experience.

INDEX

Topics		Page no
Introduction	-----	01
Object Creation	-----	02
Tabs	-----	03
Lightning App	-----	03
Fields	-----	04
Validation Rule	-----	05
Profile	-----	05
Role Heirarchy	-----	06
User	-----	07
Sharing Setting	-----	08
Flow	-----	09
Apex Trigger	-----	14
Report	-----	14
Dashboard	-----	16

INTRODUCTION

The Garage Management System (GMS) is a helpful tool for auto repair shops. It makes it easier for them to provide great service, work more efficiently, and keep customers happy. With an easy-to-use interface and strong features, GMS helps garages succeed in a competitive market and ensures a smooth experience for both customers and employees. This project is built using Salesforce.

Task 1: Object Creation

So for this particular project we create Four Objects named

1. Customer Details Object
2. Appointment Object
3. Service records Object
4. Billing details and feedback Object

Steps for creating the Customer Details object:-

To create an object:

1. From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.
 1. Enter the label name >> Customer Details
 2. Plural label name >> Customer Details
 3. 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.

Steps for Creating the Appointment Object:-

1. From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.
 1. Enter the label name >> Appointment
 2. Plural label name >> Appointments
 3. Enter Record Name Label and Format
Record Name >> Appointment Name
Data Type >> Auto Number
Display Format >> app-{000}
Starting number >> 1
2. Click on Allow reports and Track Field History,
3. Allow search >> Save.

Steps for creating Service Record Object:-

1. From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.
 1. Enter the label name >> Service records
 2. Plural label name >> Service records
 3. Enter Record Name Label and Format
Record Name >> Service records Name

Data Type >> Auto Number
Display Format >> ser-{000}
Starting number >> 1

2. Click on Allow reports and Track Field History,
3. Allow search >> Save.

Steps for Creating Billing details and feedback Object

1. From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.
 1. Enter the label name >> Billing details and feedback
 2. Plural label name >> Billing details and feedback
 3. Enter Record Name Label and Format
Record Name >> Billing details and feedback Name
Data Type >> Auto Number
Display Format >> bill-{000}
Starting number >> 1
2. Click on Allow reports and Track Field History,
3. Allow search >> Save.

Task 2: Tabs

In this project we created 4 tabs for each objects.

Steps for creating Tab:(Customer Details)

1. Go to setup page >> type Tabs in Quick Find bar >> click on tabs >> New (under custom object tab)
2. Select Object (Customer Details) >> Select the tab style >> Next (Add to profiles page) keep it as default >> Next (Add to Custom App) uncheck the include tab.
3. Make sure that the Append tab to users' existing personal customizations is checked.
4. Click save.

Similar for other objects.

Task 3: Lightning App

For this project we created a Lightning App page named- "Garage Management Application".

Steps for Creating Lightning App: -

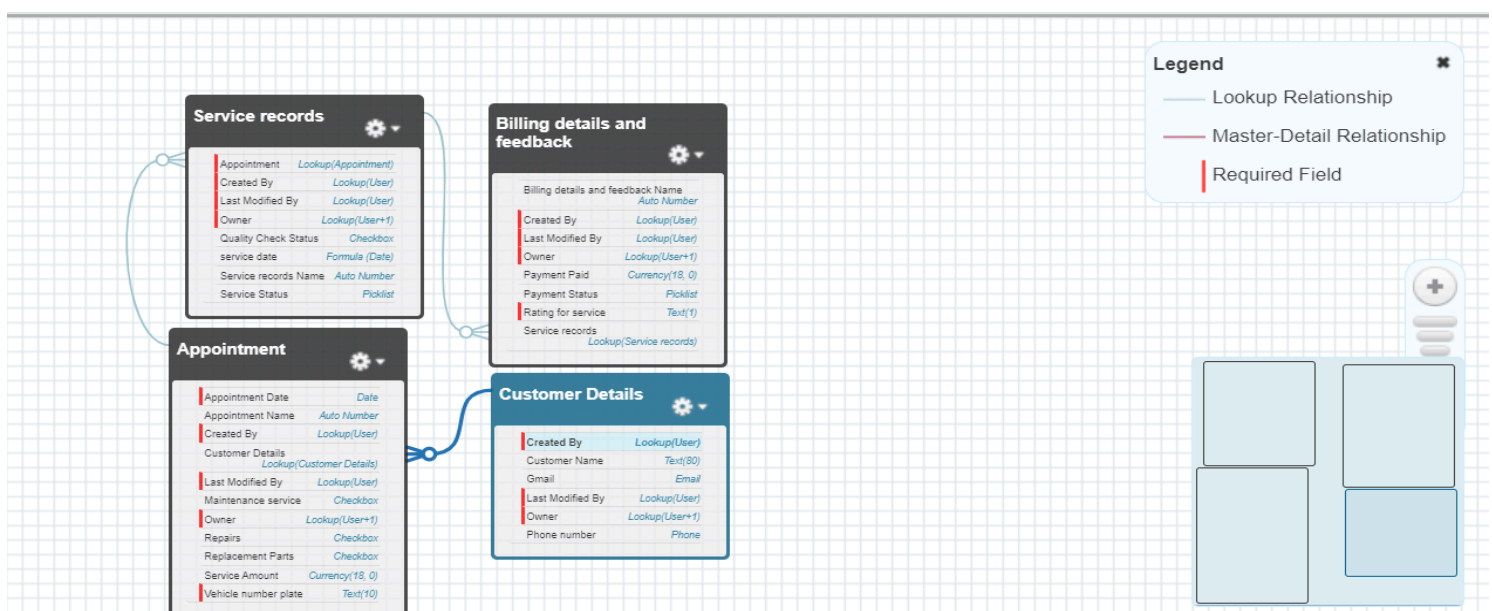
1. Go to setup page >> search “app manager” in quick find >> select “app manager” >> click on New lightning App.
2. Fill the app name in app details as Garage Management Application >> Next >> (App option page) keep it as default >> Next >> (Utility Items) keep it as default >> Next.
3. To Add Navigation Items:
4. Select the items (Customer Details, Appointments, Service records, Billing details and feedback, Reports and Dashboards) from the search bar and move it using the arrow button >> Next.
5. To Add User Profiles: Search profiles (System administrator) in the search bar >> click on the arrow button >> save & finish.

Task 4: Fields

Steps for creating fields: -

1. Go to setup >> click on Object Manager >> type object name(Customer Details) in search bar >> click on the object.
2. Now click on “Fields & Relationships” >> New
3. Select Data Type as a “Phone”
4. Click on next.
5. Fill the Above as following: • Field Label: Phone number • Field Name: gets auto generated • Click on Next >> Next >> Save and new.

For this project many types of field here is the image of all fields.



Task 5: Validation Rule

In this project we created 3 Validation Rule.

Steps for creating Validation rule: -

1. Go to the setup page >> click on object manager >> From drop down click edit for Appointment object.
2. Click on the validation rule >> click New.
3. Enter the Rule name as "Vehicle".
4. Insert the Error Condition Formula as: - NOT(REGEX(Vehicle_number_plate__c,"[A-Z]{2} [0-9]{2}[A-Z]{2} [0-9]{4}"))
5. Enter the Error Message as "Please enter valid number", select the Error location as Field and select the field as "Vehicle number plate", and click Save.

Similarly, we created 2 another validation rule for service record object and Billing details and feedback Object.

Task 6: Profile

For this project we mainly created a two profiles: - • Manager profile • Sales person profile

Steps to create a new profile:

1. Go to setup >> type profiles in quick find box >> click on profiles >> clone the desired profile (Standard User) >> enter profile name (Manager) >> Save.
2. While still on the profile page, then click Edit.
3. Select the Custom App settings as default for the Garage management.
4. Scroll down to Custom Object Permissions and Give access permissions for Appointments, Billing details and feedback, service records and customer details objects as mentioned in the below diagram.

Custom Object Permissions													
	Basic Access				Data Administration			Basic Access				Data Administration	
	Read	Create	Edit	Delete	View All	Modify All		Read	Create	Edit	Delete	View All	Modify All
Appointments	✓	<input type="checkbox"/>	✓	✓	✓	✓	Customer Details	✓	<input type="checkbox"/>	✓	✓	✓	✓
Billing details and feedback	✓	<input type="checkbox"/>	✓	✓	✓	✓	Service records	✓	<input type="checkbox"/>	✓	✓	✓	✓

5. Changing the session times out after should be " 8 hours of inactivity".
6. Change the password policies as mentioned :
7. User passwords expire in should be " never expires ".
8. Minimum password length should be " 8 ", and click save.

Sales Person Profile: -

1. Go to setup >> type profiles in quick find box >> click on profiles >> clone the desired profile (Salesforce Platform User) >> enter profile name (sales person) >> Save.
2. While still on the profile page, then click Edit.
3. Select the Custom App settings as default for the Garage management.
4. Scroll down to Custom Object Permissions and Give access permissions for Appointments, Billing details and feedback, service records and customer details objects as mentioned in the below diagram.

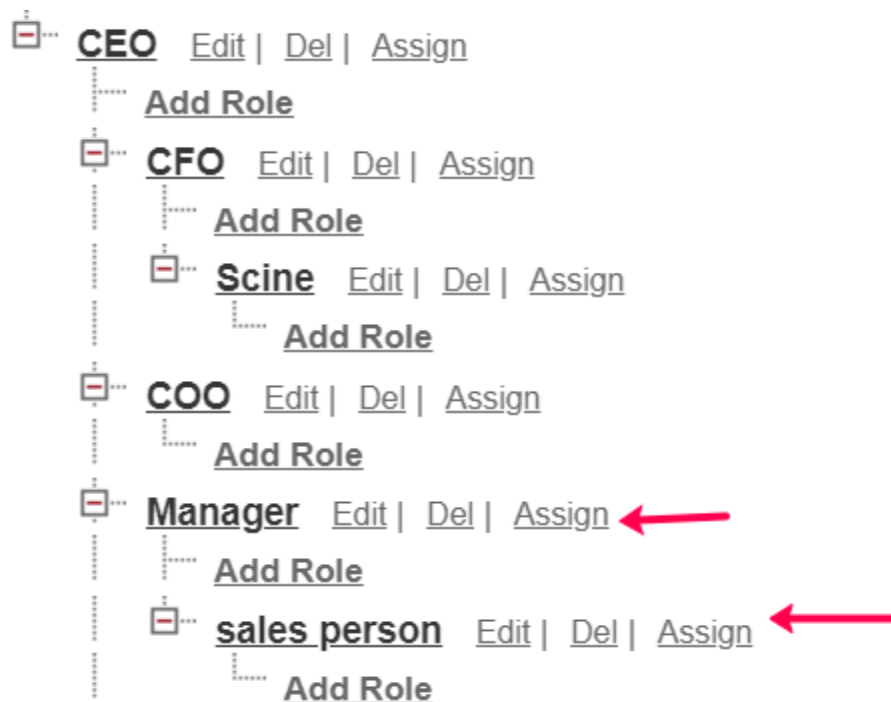
Custom Object Permissions													
	Basic Access				Data Administration			Basic Access				Data Administration	
	Read	Create	Edit	Delete	View All	Modify All		Read	Create	Edit	Delete	View All	Modify All
Appointments	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Customer Details	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Billing details and feedback	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Service records	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

5. And click save

Task 7: Role & Role Hierarchy

For this project we created two Roles: -

1. Manager Role Under CEO
2. Sales Person Role Under Manager



Task 8: User

Steps for creating User:

1. Go to setup >> type users in quick find box >> select users >> click New user.
2. Fill in the fields

First Name : Niklaus

Last Name : Mikaelson

Alias : Give a Alias Name

Email id : Give your Personal Email id

Username : Username should be in this form: text@text.text

Nick Name : Give a Nickname

Role : Manager

User licence : Salesforce

Profiles : Manager

New User

Help for this

User Edit

Save Save & New Cancel

General Information

First Name: Niklaus

Last Name: Mikaelson

Alias: nmika

Email:

Username: Mikaelson@Niklaus

Nickname: nik

Title:

Company:

Department:

Division:

Role: Manager

User License: Salesforce

Profile: Manager

Active: ☒

Marketing User: ☐

Offline User: ☐

Knowledge User: ☐

Flow User: ☐

Service Cloud User: ☐

Site.com Contributor User: ☐

Site.com Publisher User: ☐

WDC User: ☐

Data.com User Type: --None--

Save.

3. Save.

Repeat the steps and create another user using

- Role: sales person
- User licence: Salesforce Platform
- Profile: sales person

Task 9: Sharing Settings

Steps for creating Sharing setting:

1. Go to setup >> type users in quick find box >> select Sharing Settings >> click Edit.
2. Change the OWD setting of the Service records Object to private as shown in fig.

Sharing Settings

Object	Private	Public Read/Write	Public Read/Write
Work Plan Template	Private	Private	✓
Work Step Template	Private	Private	✓
Work Type	Private	Private	✓
Work Type Group	Public Read/Write	Private	✓
Appointment	Public Read/Write	Private	✓
Billing details and feedback	Public Read/Write	Private	✓
Customer Details	Public Read/Write	Private	✓
Environment	Public Read/Write	Private	✓
Laptop	Public Read/Write	Private	✓
Service records	Private	Private	✓
SessionData	Public Read/Write	Private	✓

User Visibility Settings

Portal User Visibility ☐ Site User Visibility ☐

Other Settings

Standard Report Visibility ☒ Manual User Record Sharing ☐ Manager Groups ☐

Minimize the number of roles created, which improves performance by cutting down processing loads ☒ Grant site users access to related cases ☒ Secure guest user access ☐ Require permission to view record names in lookup fields ☐

Save Cancel

3. Click on save and refresh.
4. Scroll down a bit, Click new on Service records sharing Rules.

Service records Sharing Rules

New Recalculate

No sharing rules specified.

6. Give the Label name as “ Sharing setting”
7. Rule name is auto populated.
8. In step 3 : Select which records to be shared, members of “ Roles ” >> “ Sales person”
9. In step 4: share with, select “ Roles ” >> “ Manager ”
10. In step 5 : Change the access level to “ Read / write ”.
11. Click on save.

Sharing Settings

You can use sharing rules only to grant wider access to data, not to restrict access.

Step 1: Rule Name

Label Rule Name Description

Step 2: Select your rule type

Rule Type ☒ Based on record owner ☐ Based on criteria

Step 3: Select which records to be shared

Service records: owned by members of

Step 4: Select the users to share with

Share with

Step 5: Select the level of access for the users

Access Level

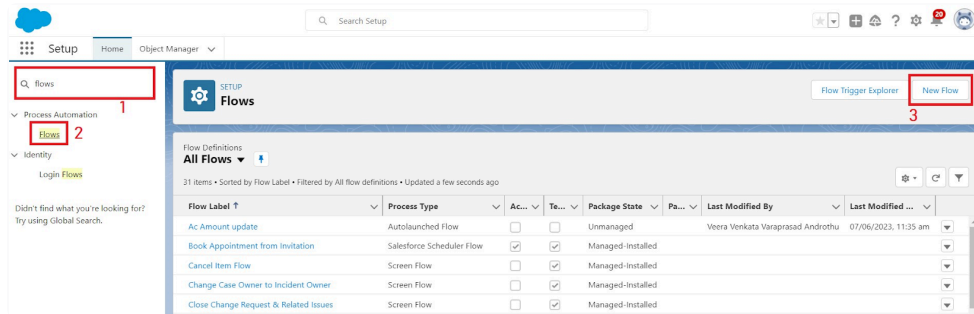
Save Cancel

Task 10: Flows

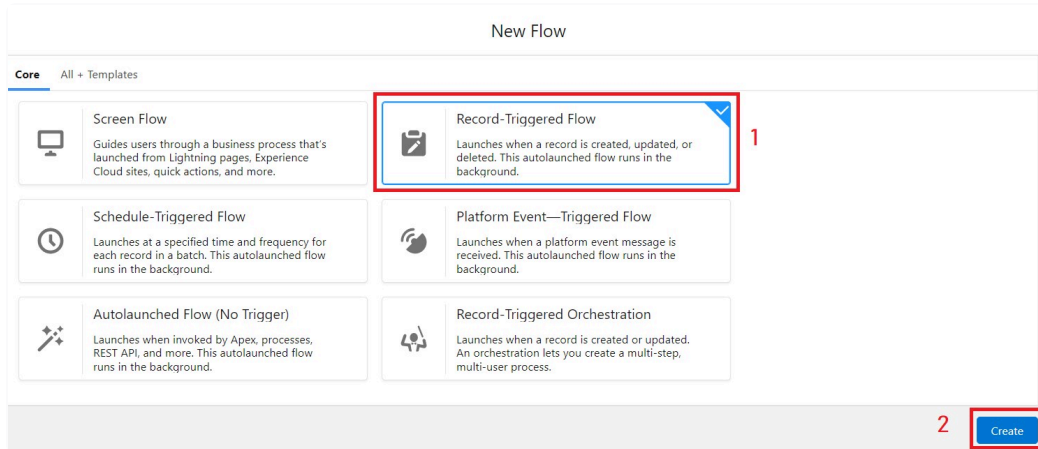
In this Project We use the Record Triggered Flow.

Steps for creating the Flow:-

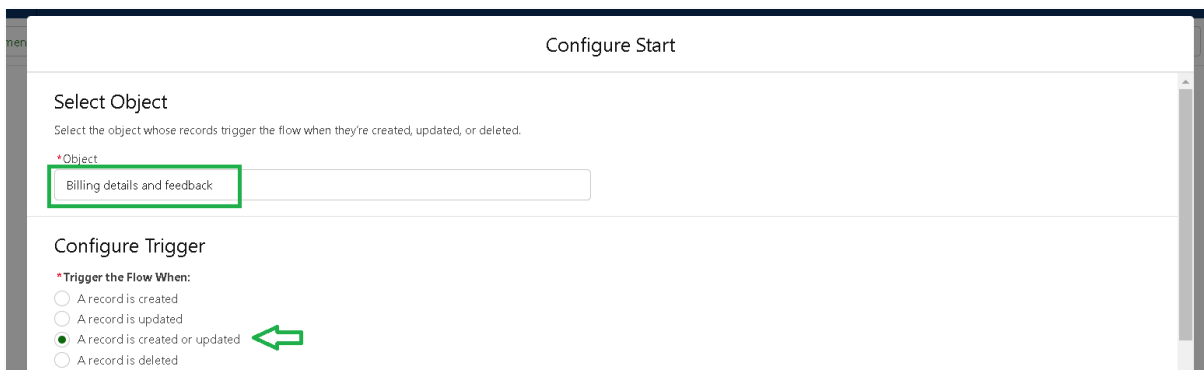
1. Go to setup >> type Flow in quick find box >> Click on the Flow and Select the New Flow.



2. Select the Record-triggered flow and Click on Create.



3. Select the Object as “Billing details and feedback” in the Drop down list.
4. Select the Trigger Flow when: “A record is Created or Updated”.
5. Select the Optimize the flow for: “Actions and Related Records” and Click on Done.



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.

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements
None

***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.

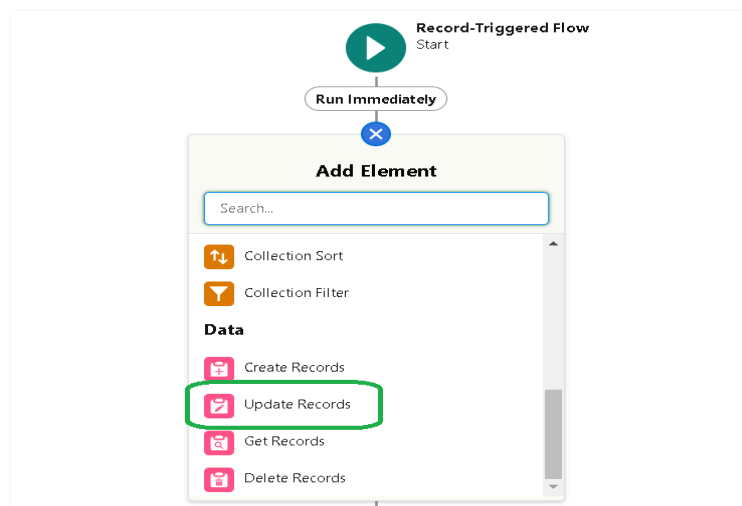
Actions and Related Records

Update any record and perform actions, like send an email. This more flexible flow runs *after* the record is saved to the database.

☐ Include a Run Asynchronously path to access an external system after the original transaction for the triggering record is successfully committed

Cancel Done

- Under the Record-triggered Flow Click on “+” Symbol and In the Drop down List select the “Update records Element”.



- Give the Label Name : Amount Update
- Api name : is auto populated

Edit Update Records

Update Salesforce records using values from the flow.

*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)

Cancel Done

Set Filter Conditions

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

Field	Operator	Value
Payment_Status__c	Equals	Completed

+ Add Condition

Set Field Values for the Billing details and feedback Record

Field	Value
Payment_Paid__c	\$Record > Service records > Appointment > Service A...

+ Add Field

Cancel Done

9. Set a filter condition : All Conditions are met(AND)

10. Field : Payment_Status__c

11. Operator : Equals

12. Value : Completed

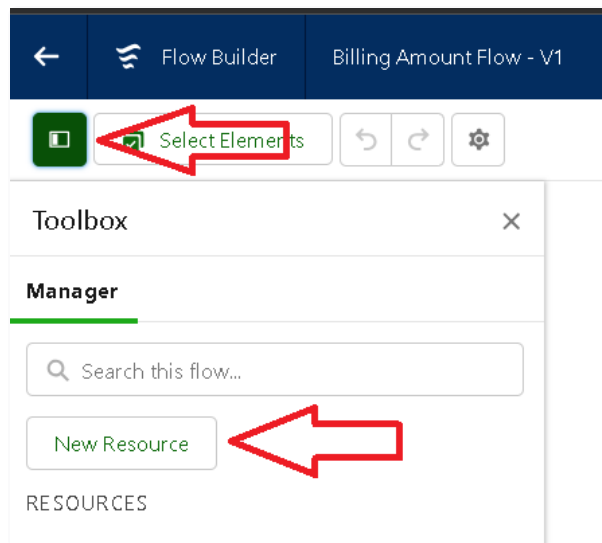
13. And Set Field Values for the Billing details and feedback Record

14. Field : Payment_Paid__c

15. Value : {!\$Record.Service_records__r.Appointment__r.Service_Amount__c}

16. Click On Done.

17. Before creating another Element. Create a New Resource form Toolbox form top left.



18. Click on the New Resource, And select Variable.

19. Select the resource type as text template.

20. Enter the API name as “ alert”.

21. Change the view as Rich Text ? View to Plain Text.

22. In body field paste the syntax that given below.

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 provided by our garage management team. Your prompt payment is greatly appreciated, and it helps us continue to provide top-notch services to you and all our valued customers.

Amount paid : {!\$Record.Payment_Paid__c}

Thank you for Coming .

23. Click done.

The screenshot shows the 'Edit Text Template' interface. The 'API Name' field is highlighted with a red box and contains the text 'alert'. The 'Body' field contains the text 'Dear {!\$Record.Service_records__r.Appointment__r.Customer_Name__r.Name},'. A red arrow points to the 'Body' field. The 'View as Plain Text' dropdown is highlighted with a red box. The 'Done' button is highlighted with a red box and a red arrow.

24. Now Click on Add Element , select Action.

25. Their action bar will be opened in that search for “ send email ” and click on it.

26. Give the label name as “ Email Alert”

27. API name will be auto populated.

28. Enable the body in set input values for the selected action.

29. Select the text template that created , Body : {!alert}

30. Include recipient address list select the email form the record.

31. RecipientAddressList: {!\$Record.Service_records__r.Appointment__r.Customer_Name__r.Gmail__c}

32. Include subject as “ Thank You for Your Payment - Garage Management”.

33. Click done.

Edit Action

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.

*Label: *API Name:

Description:

Set Input Values for the Selected Action

A₀ Body: ☒ Include

A₀ Email Template ID: ☐ Don't Include

☒ Log Email on Send ☐ Don't Include

Edit Action

A₀ Recipient Address List: ☒ Include

A₀ Recipient ID: ☐ Don't Include

A₀ Related Record ID: ☐ Don't Include

☒ Rich-Text-Formatted Body: ☐ Don't Include

A₀ Sender Email Address: ☐ Don't Include

A₀ Sender Type: ☐ Don't Include

A₀ Subject: ☒ Include

34. Click on save. Give the Flow label , Flow Api name will be autopopulated.

35. And click save, and click on activate.

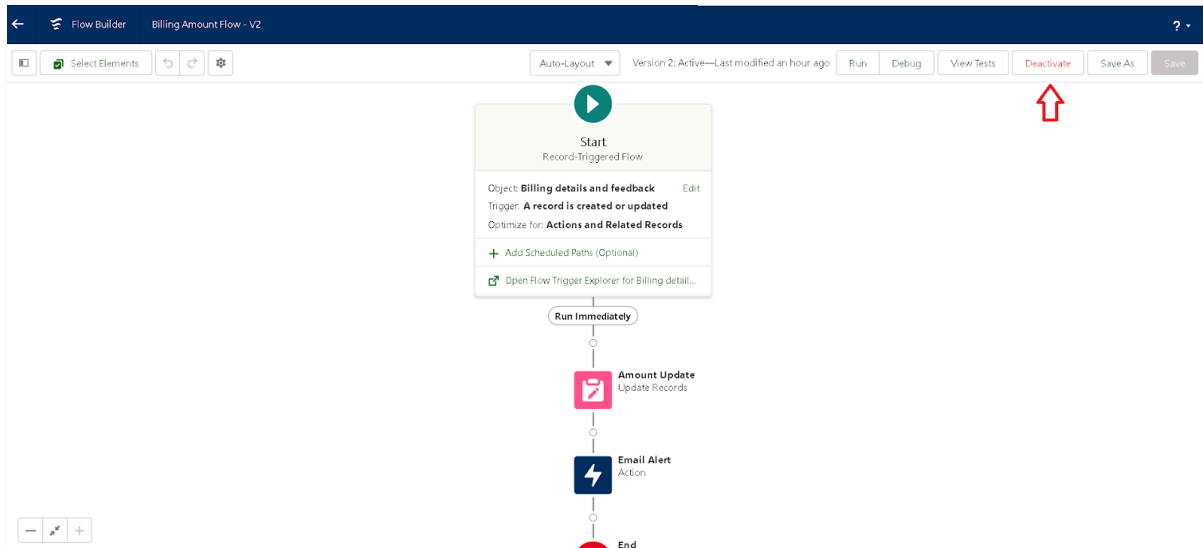
Record-Triggered Flow

Save as

*Flow Label: *Flow API Name:

Description:

Show Advanced



Task 11: Apex Trigger

This use case works for Amount Distribution for each Service the customer selected for there Vehicle.

Steps for creating Apex Trigger are:-

1. Login to the respective trailhead account and navigate to the gear icon in the top right corner.
2. Click on the Developer console. Now you will see a new console window.
3. In the toolbar, you can see FILE. Click on it and navigate to new and create New apex class.
4. Name the class as "AmountDistributionHandler".

```

AmountDistribution.appt  AmountDistributionHandler.apxc * A
Code Coverage: None  API Version: 58  Go To

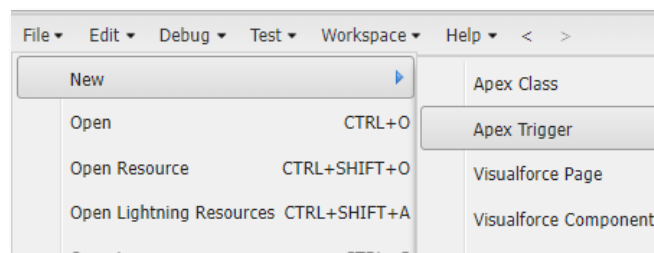
1 public class AmountDistributionHandler {
2
3     public static void amountDist(list<Appointment__c> listApp){
4         list<Service_records__c> serList = new list <Service_records__c>();
5
6         for(Appointment__c app : listApp){
7             if(app.Maintenance_service__c == true && app.Repairs__c == true && app.Replacement_Parts__c == true){
8                 app.Service_Amount__c = 10000;
9             }
10            else if(app.Maintenance_service__c == true && app.Repairs__c == true){
11                app.Service_Amount__c = 5000;
12            }
13            else if(app.Maintenance_service__c == true && app.Replacement_Parts__c == true){
14                app.Service_Amount__c = 8000;
15            }
16            else if(app.Repairs__c == true && app.Replacement_Parts__c == true){
17                app.Service_Amount__c = 7000;
18            }
19            else if(app.Maintenance_service__c == true){
20                app.Service_Amount__c = 2000;

```

```
AmountDistribution.apxt | AmountDistributionHandler.apxc *
Code Coverage: None | API Version: 58
12 }
13 else if(app.Maintenance_service__c == true && app.Replacement_Parts__c == true){
14     app.Service_Amount__c = 8000;
15 }
16 else if(app.Repairs__c == true && app.Replacement_Parts__c == true){
17     app.Service_Amount__c = 7000;
18 }
19 else if(app.Maintenance_service__c == true){
20     app.Service_Amount__c = 2000;
21 }
22 else if(app.Repairs__c == true){
23     app.Service_Amount__c = 3000;
24 }
25 else if(app.Replacement_Parts__c == true){
26     app.Service_Amount__c = 5000;
27 }
28
29 }
30 }
31 }
```

Steps for creating Trigger Handler:-

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

A screenshot of the 'New Apex Trigger' dialog box. It has a title bar with a close button. Inside, there are two fields: 'Name:' with a text input field, and 'sObject:' with a dropdown menu. At the bottom right, there is a 'Submit' button.

Syntax For creating trigger :

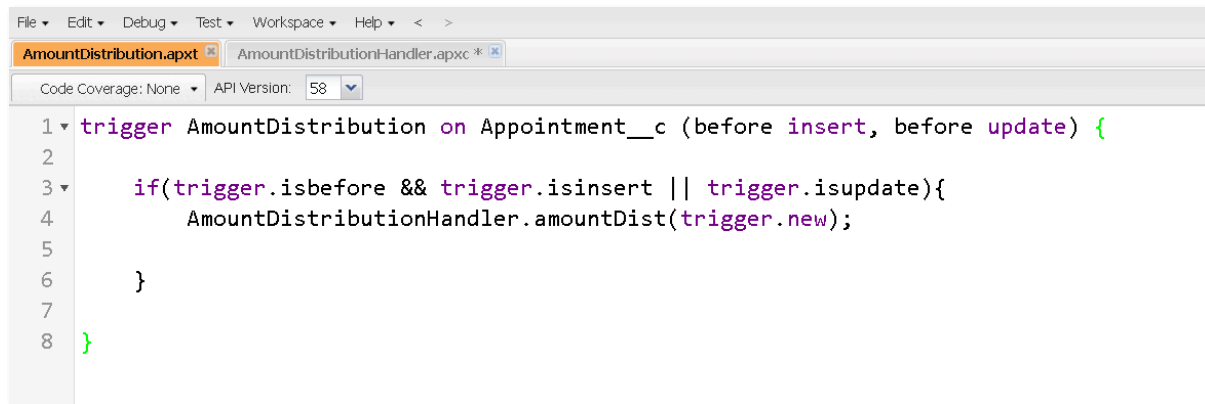
The syntax for creating trigger is :

Trigger [trigger name] on [object name](Before/After event)

```
{  
}
```

In this project , trigger is called whenever the particular records sum exceed the threshold i.e minimum business requirement value. Then the code in the trigger will get executed.

1. Handler for the Appointment Object



Task 11: Report

Steps for creating the Report folder:-

1. Click on the app launcher and search for reports.
2. Click on the report tab, click on new folder.
3. Give the Folder label as "Garage Management Folder", Folder unique name will be auto populated.
4. Click save.

Steps for Sharing a Report folder:-

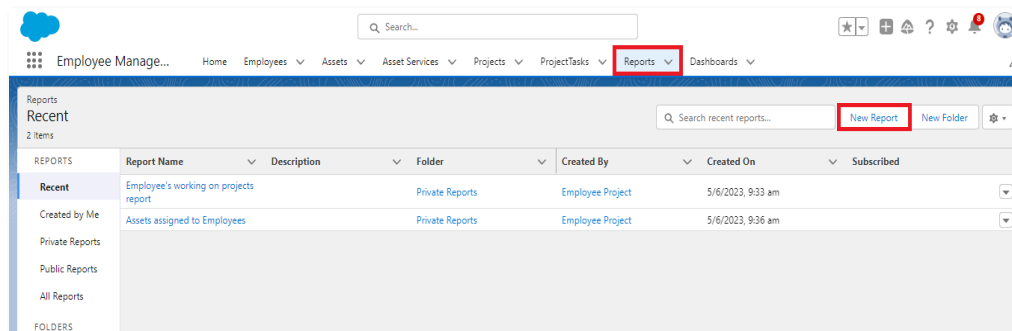
1. Go to the app >> click on the reports tab.
2. Click on the All folder , click on the Drop down arrow for Garage Management folder, and Click on share.
3. Select the share with as "roles", in name field search for "manager", give "view" as access for that role.
4. Then click share, and click on Done.

Steps for Creating a Report Type:-

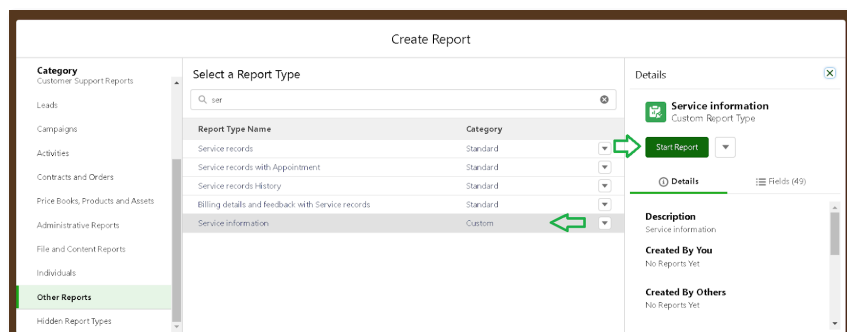
1. Go to setup >> type users in quick find box >> select Report Type >> click on Continue.
2. Click on new custom report type.
3. Select the Primary object as “ Customer details” .
4. Give the Report type Label as “ Service information ”
5. Report type Name is autopopulated.
6. Keep the Description as same.
7. Select Store in Category as “ other Reports ”
8. Select the deployment status as “ Deployed ”, click on Next.
9. now , Click on Related object box.
10. Click on Select Object, choose Appointment Object as shown in fig.
11. Again Click to relate another object.
12. And select the related object as “ service records”.
13. Repeat the process and select the related object as “ Billing details and feedback”.
14. And click on save.

Steps for Creating A Report:-

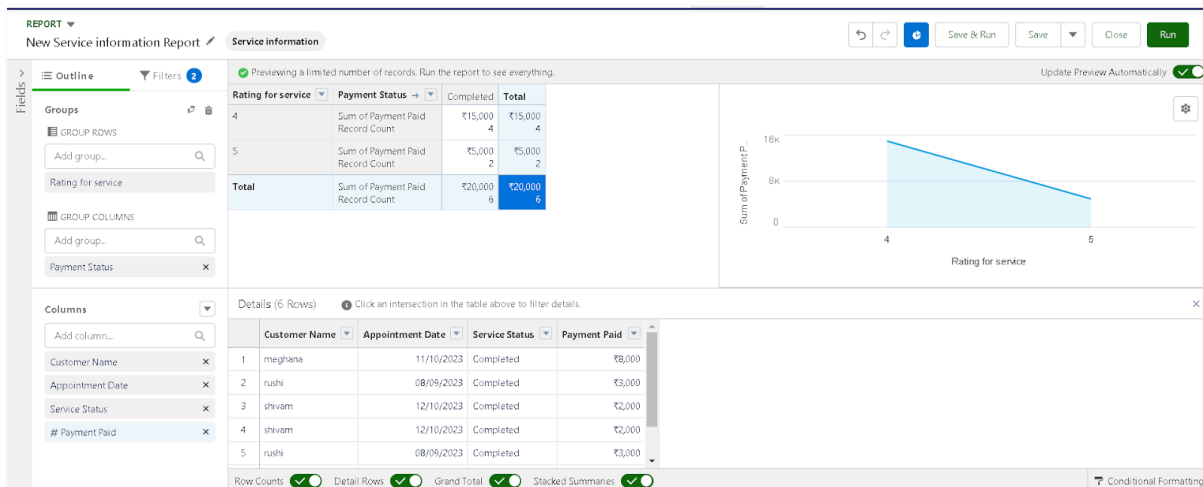
1. Go to the app >> click on the reports tab
2. Click New Report.



3. Select the Category as other reports, search for Service Information, select that report, click on it. And click on start report.



4. Their outline pane is opened already, select the fields that mentioned below in column section.
 1. Customer name
 2. Appointment Date
 3. Service Status
 4. Payment paid
5. Remove the unnecessary fields.
6. Select the fields that mentioned below in GROUP ROWS section.
 1. Rating for Service
7. Select the fields that mentioned below in GROUP ROWS section.
 1. Payment Status
8. Click on Add Chart , Select the Line Chart.
9. Click on save, Give the report Name : New Service information Report
10. Report unique Name is auto populated.
11. Select the folder the created and Click on save.



The 'Save Report' dialog box is shown with the following fields:

- Report Name:** New Service information Report (indicated by a green arrow)
- Report Unique Name:** New_Service_information_Report_oVu
- Report Description:** (Empty text area)
- Folder:** Garage Management Folder (indicated by a green arrow)

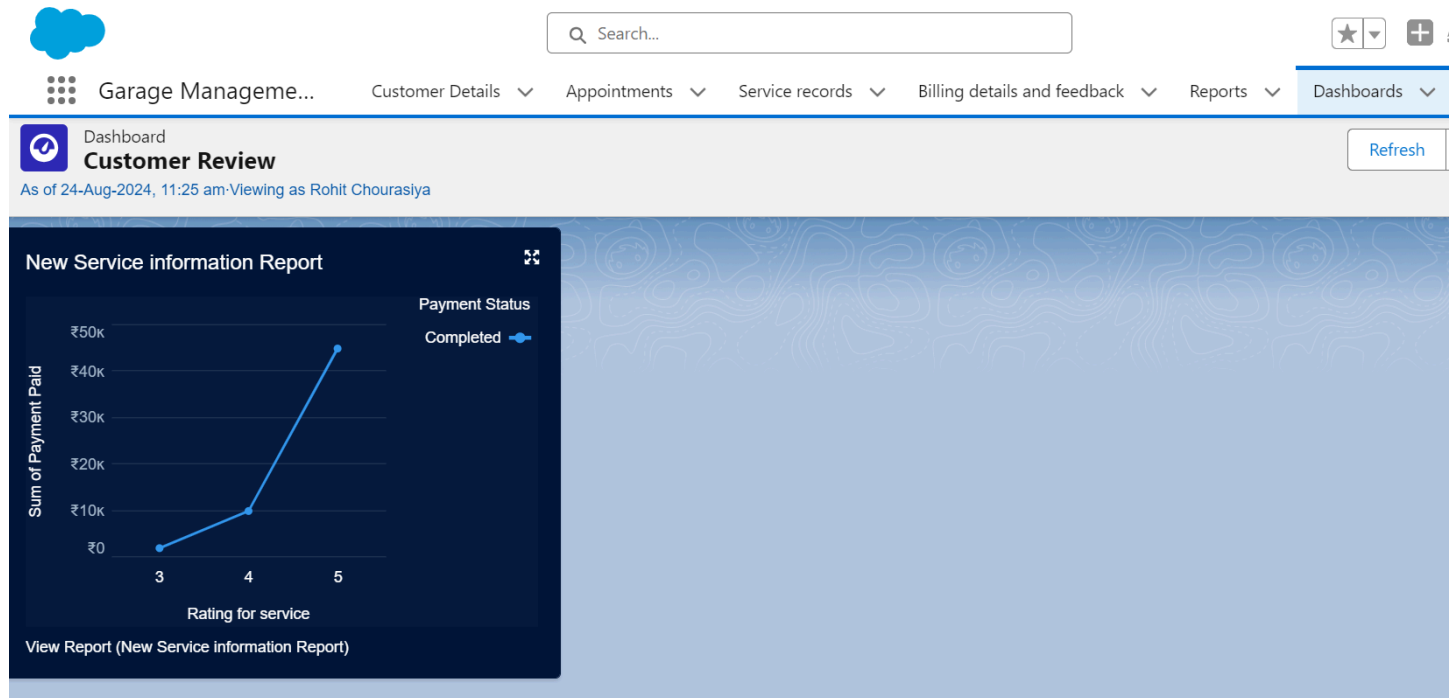
Buttons at the bottom: Cancel, Save.

Task 12: Dashboard

Dashboard is simply a visual representation of the Report.

Steps for creating the Dashboard are:-

1. Go to the app >> click on the Dashboards tabs.
2. Give a Name and select the folder that created, and click on create.
3. Select add component.
4. Select a Report and click on select.
5. Select the Line Chart. Change the theme.
6. Click Add then click on Save and then click on Done.
7. Preview is shown below
8. After that Click on Subscribe on top right.
9. Set the Frequency as “ weekly ”.
10. Set a day as monday.
11. And Click on save.



Thank You