FIELD SERVICE

WORKORDER OPTIMIZATION

By

MOUNIKA MEKA

21bq1a05e2@vvit.net

ABSTRACT

Objective: The Field Service Work Order Optimization System streamlines operations for a company providing installations and repairs. Utilizing a robust database, the system efficiently matches work orders with skilled technicians based on technicians' location, availability, and skills. The system employs a prioritization algorithm, focusing on assigning tasks to technicians. Automated communication keeps technicians informed, while analytics offer insights for continuous improvement. Overall, this solution maximizes efficiency, reduces operational costs, and improves customer satisfaction in the dynamic realm of field service operations

Key Technologies:

1. **Salesforce Field Service:** Salesforce's comprehensive field service management solution will be the core platform, providing tools for scheduling, dispatching, and real-time communication.
2. **Artificial Intelligence & Machine Learning:** Integrated with Salesforce, AI and ML algorithms will predict service demands, optimize scheduling, and match the right technician to the right job based on skill set, location, and availability.
3. **Predictive Analytics:** Leveraging Salesforce's analytics capabilities, predictive models will forecast service needs and preemptively address potential issues by analyzing historical data and current conditions.
4. **Internet of Things (IoT):** IoT devices will provide real-time data from field equipment, which will be integrated into Salesforce for proactive maintenance and swift response to issues.

Implementation Phases:

1. **Salesforce Field Service:** Salesforce's comprehensive field service management solution will be the core platform, providing tools for scheduling, dispatching, and real-time communication.
2. **Artificial Intelligence & Machine Learning:** Integrated with Salesforce, AI and ML algorithms will predict service demands, optimize scheduling, and match the right technician to the right job based on skill set, location, and availability.
3. **Predictive Analytics:** Leveraging Salesforce's analytics capabilities, predictive models will forecast service needs and preemptively address potential issues by analyzing historical data and current conditions.
4. **Internet of Things (IoT):** IoT devices will provide real-time data from field equipment, which will be integrated into Salesforce for proactive maintenance and swift response to issues.

Potential Challenges:

1. **Data Integration:** Ensuring seamless integration of various data sources and legacy systems.
2. **Change Management:** Managing the transition and ensuring buy-in from all stakeholders.
3. **Scalability:** Ensuring the solution can scale to accommodate growth and increased demand.
4. **Security and Privacy:** Protecting sensitive customer and operational data from breaches.

Measurable Outcomes:

1. Efficiency Metrics
2. Customer Satisfaction
3. Operational Excellence

Functional Requirements:

1. Work Order Management
2. Scheduling and Dispatching
3. Resource Management
4. Mobile Access
5. Customer Communication
6. Analytics and Reporting
7. Integration
8. User Management and Security
9. Maintenance and Support

By fulfilling these functional requirements, the Salesforce Field Service Work Order Optimization project will enhance the efficiency of field operations, improve customer satisfaction, and achieve overall business objectives.

INDEX PAGE

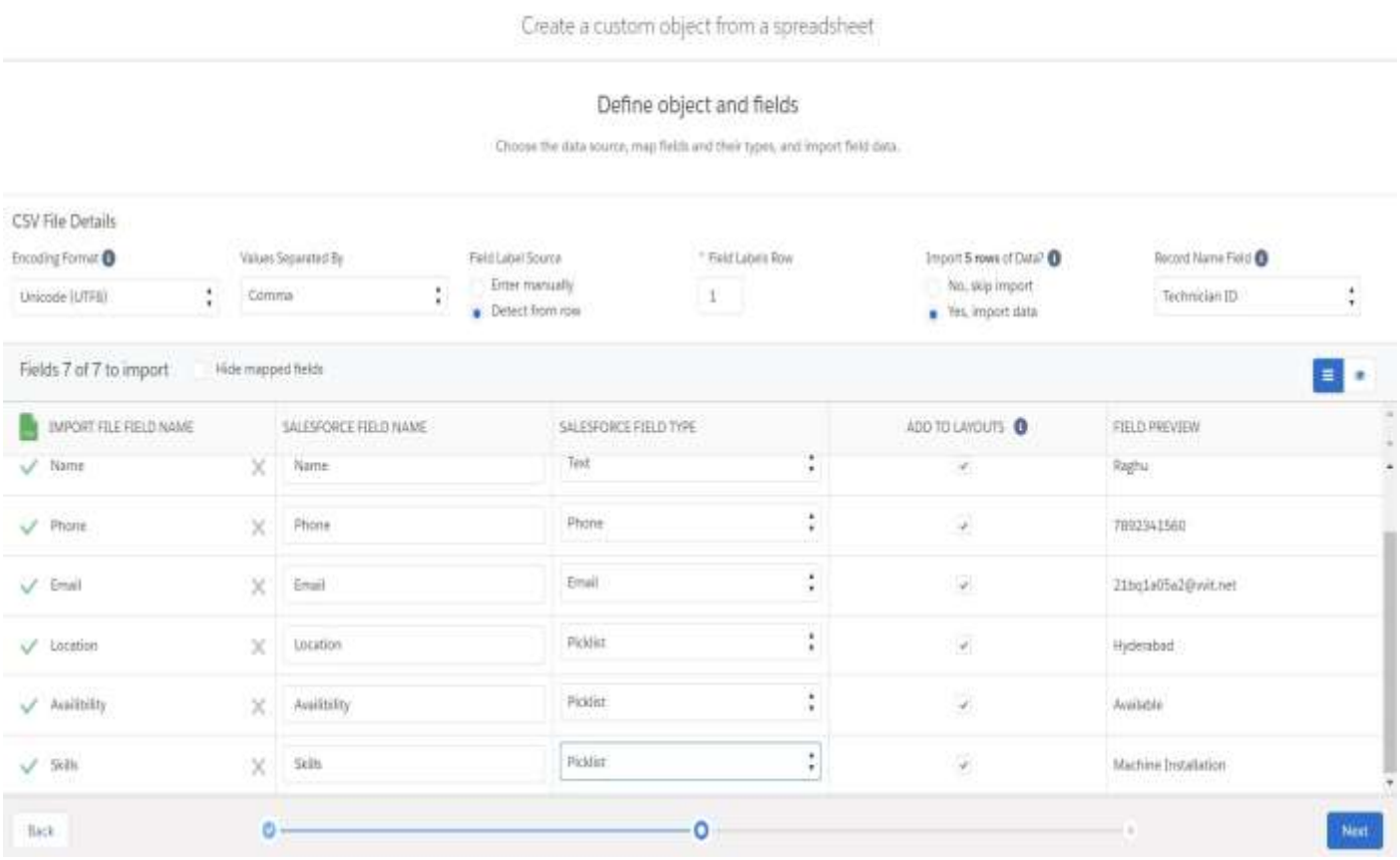
INTRODUCTION

The Field Service Work Order Optimization System streamlines operations for a company providing installations and repairs. Utilizing a robust database, the system efficiently matches work orders with skilled technicians based on technicians' location, availability, and skills. The system employs a prioritization algorithm, focusing on assigning tasks to technicians. Automated communication keeps technicians informed, while analytics offer insights for continuous improvement. Overall, this solution maximizes efficiency, reduces operational costs, and improves customer satisfaction in the dynamic realm of field service operations.

Task 1:

1.1 Create Technician Object:

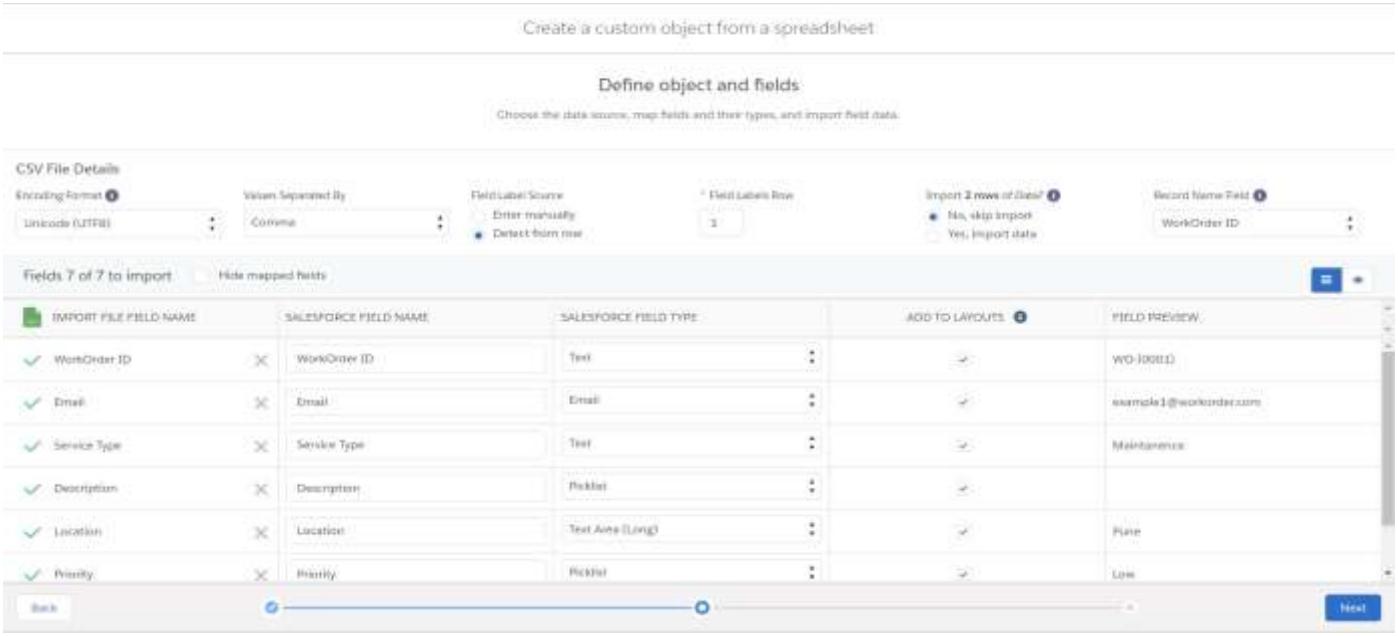
An entity representing field technicians, capturing details like skills, name, location, availability, and contact information for optimized service dispatch.



After creating technician details, the Quick box looks like the below

1.2 Create WorkOrder Object:

An entity tracking service tasks, detailing job requirements, status, assigned technician, and customer information for efficient field operations.

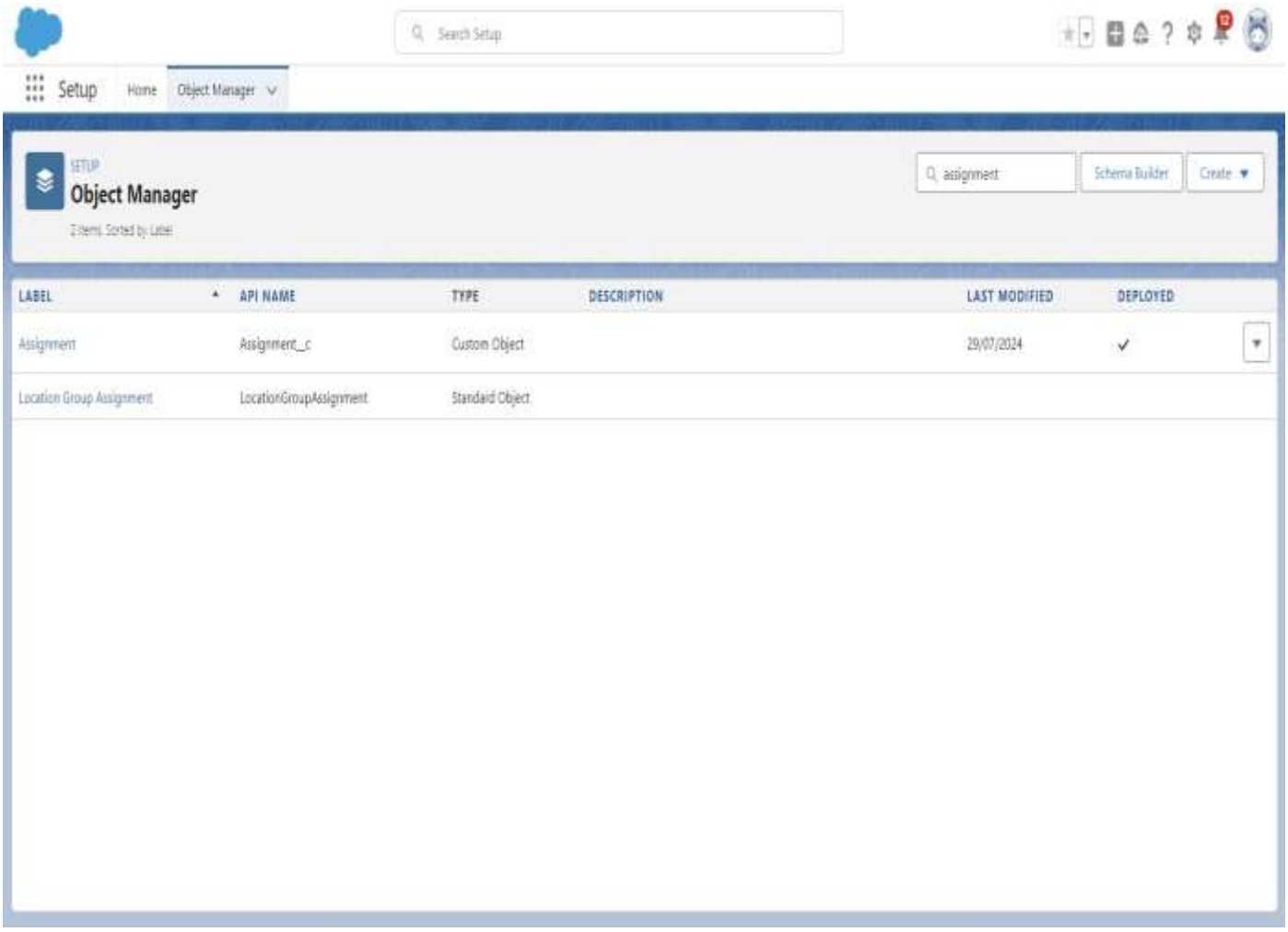


After creating the WorkOrder Custom object it looks like the below

1.3 Create Assignment Object :

An entity linking technicians to work orders, detailing assignment dates, priority, status, and specific tasks for optimized field service.

After creating the Assignment custom object, the object manager bar looks the below



Task 2:

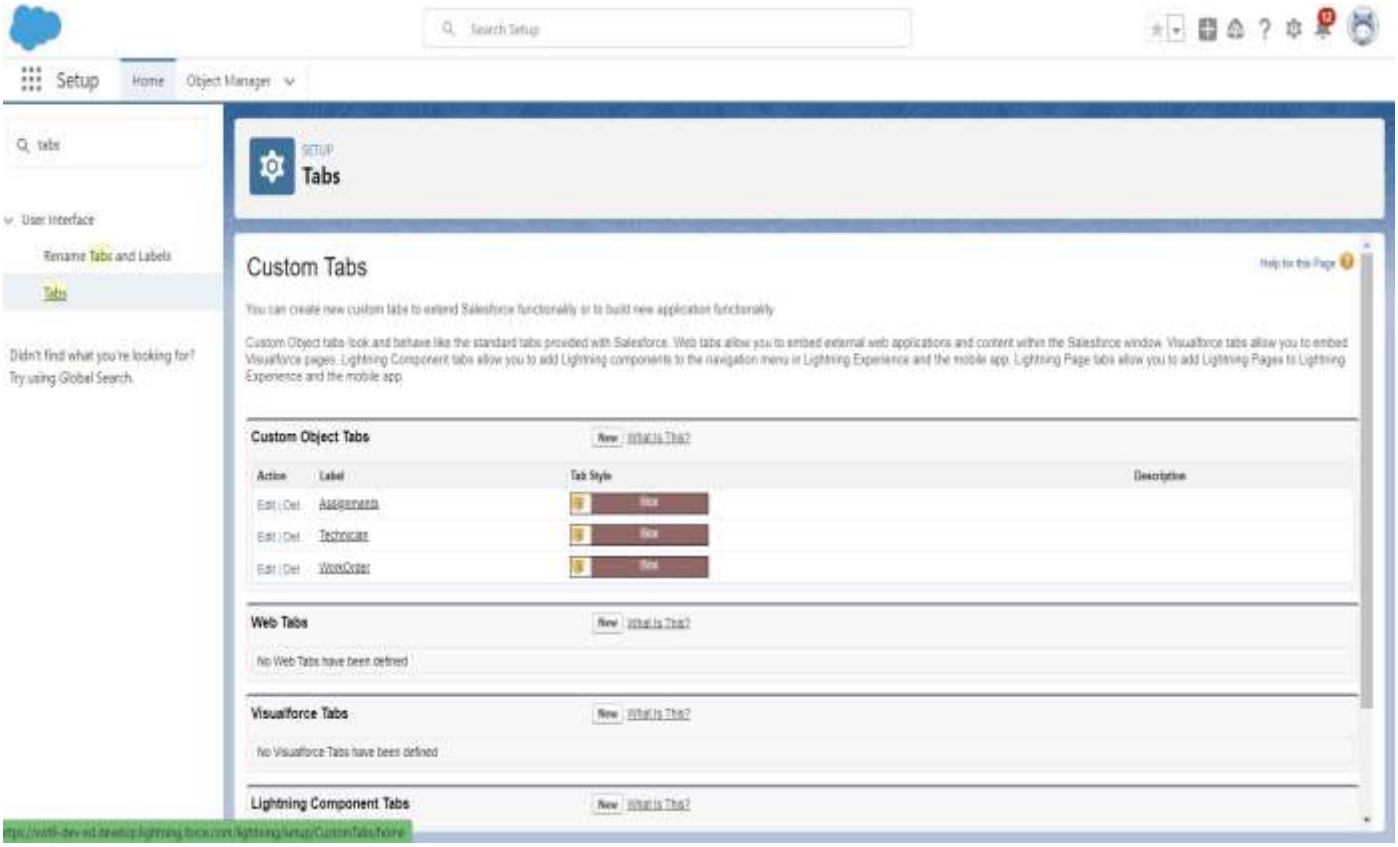
Creating a Custom Tab

A user interface element in Salesforce that provides access to custom objects, records, or web content, enhancing navigation and organization of data within the Salesforce environment. To create a Tab:(Assignment)

1. Go to the setup page --> type Tabs in the Quick Find bar --> click on tabs --> New (under the custom object tab)
2. Select Object(Assignment) --> Select any tab style --> Next (Add to profiles page) keep it as default --> Next (Add to Custom App) keep it as default --> Save.

Note: Tabs for WorkOrder & Technician objects do get created automatically. We do not need to create tabs for those objects.

After following the above steps, the output looks like this:



Task 3 :

Create a Lightning App

To create a lightning app page:

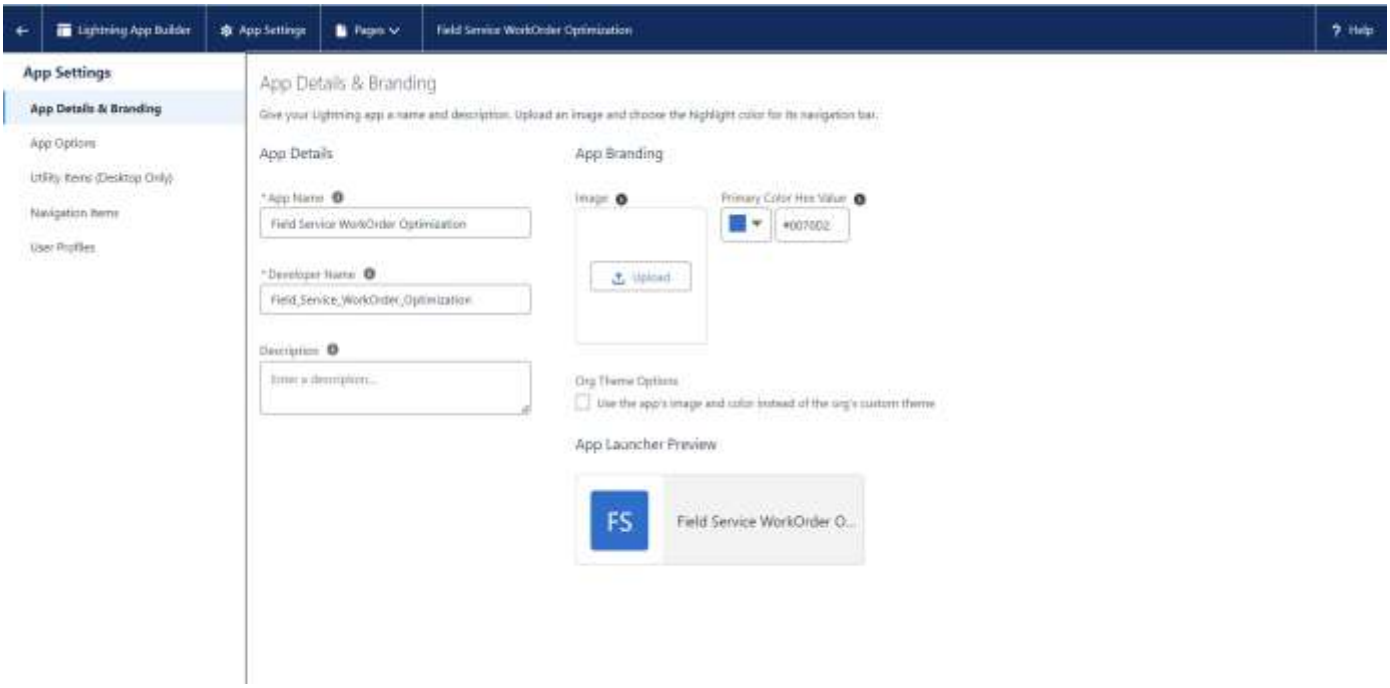
1. Go to the setup page --> search “app manager” in quick find --> select “app manager” --> click on New lightning App.
2. Fill the app name in app details and branding as follow

   App Name : Field Service WorkOrder Optimization

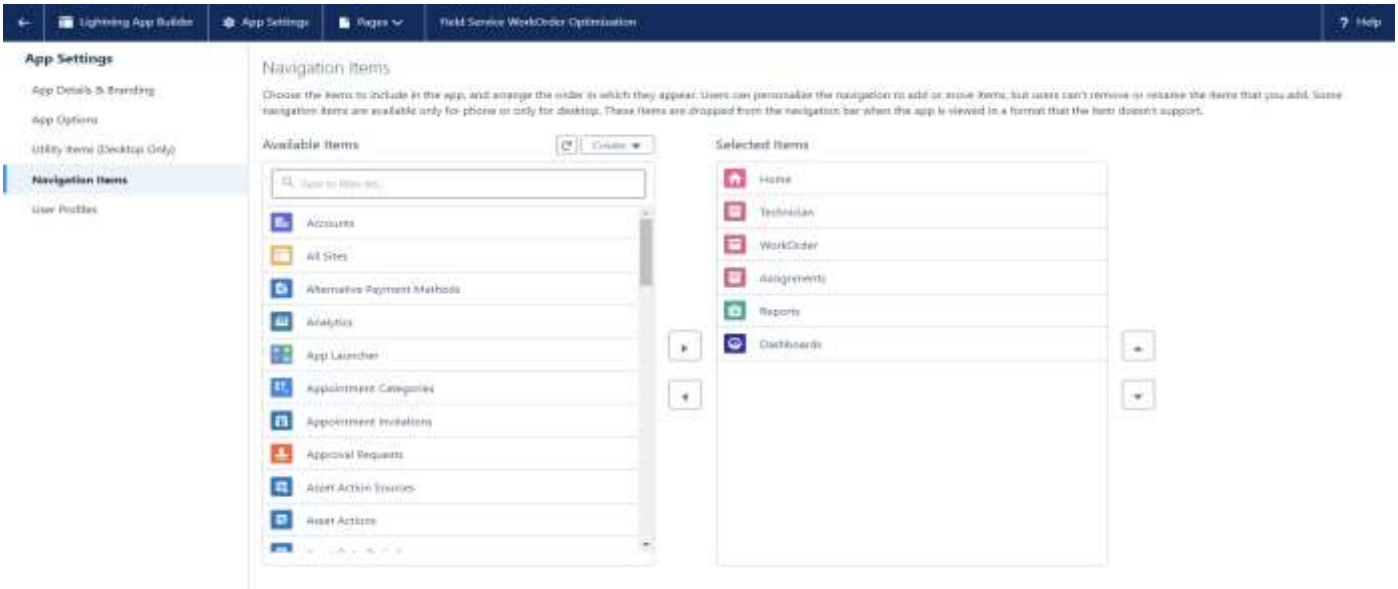
   Developer Name : this will be auto populated

   Description : Give a meaningful description

   Image : optional (if you want to give any image you can, otherwise not mandatory) Primary color hex value : keep this default



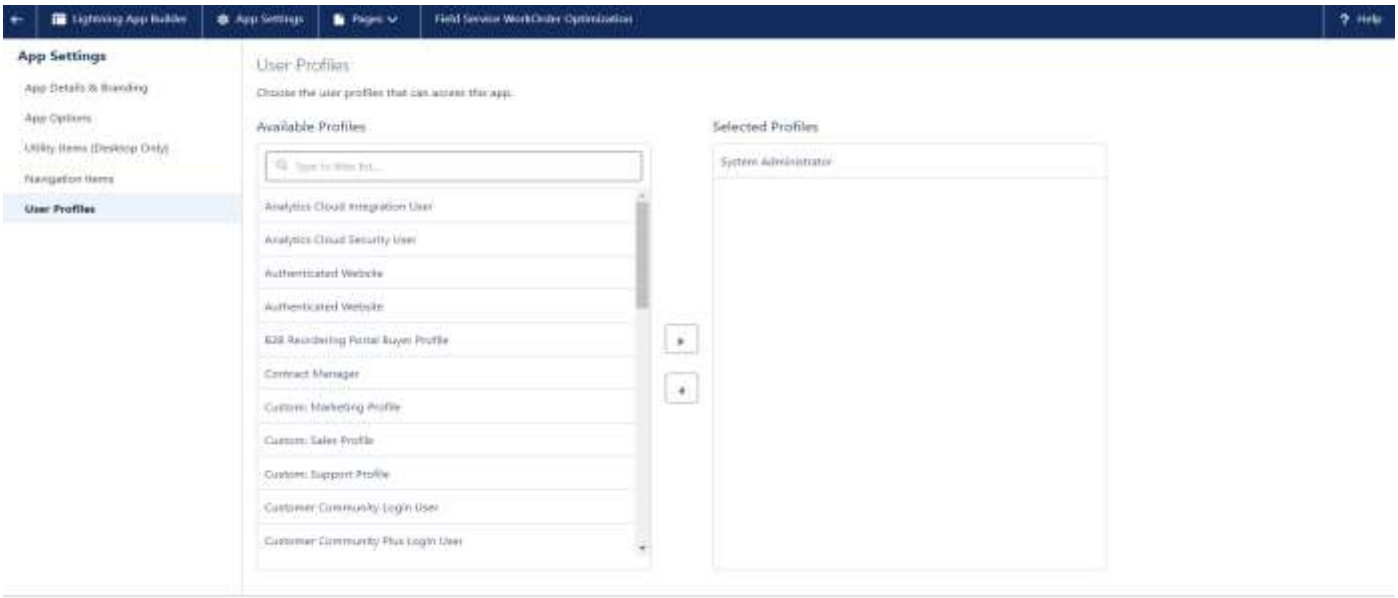
3. Then click Next --> (App option page) keep it as default --> Next --> (Utility Items) keep it as default --> Next
4. To Add Navigation Items:



Search the items in the search bar(Home, WorkOrder, Technician, Assignment, Reports, Dashboard) from the search bar and move it using the arrow button ? Next.

Note: select asset the custom object which we have created in the previous activity. 5. To Add User Profiles:

Search profiles (System administrator) in the search bar --> click on the arrow button --> save & finish.

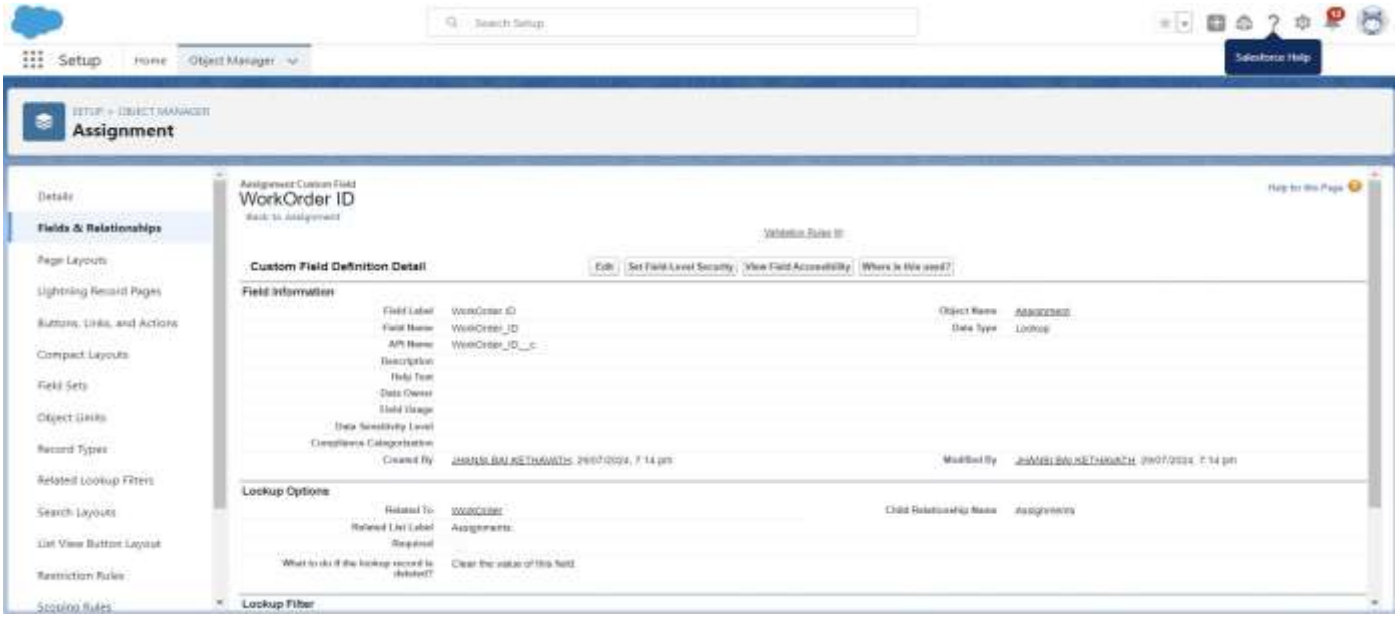


This is the output after completion of following the above procedure.

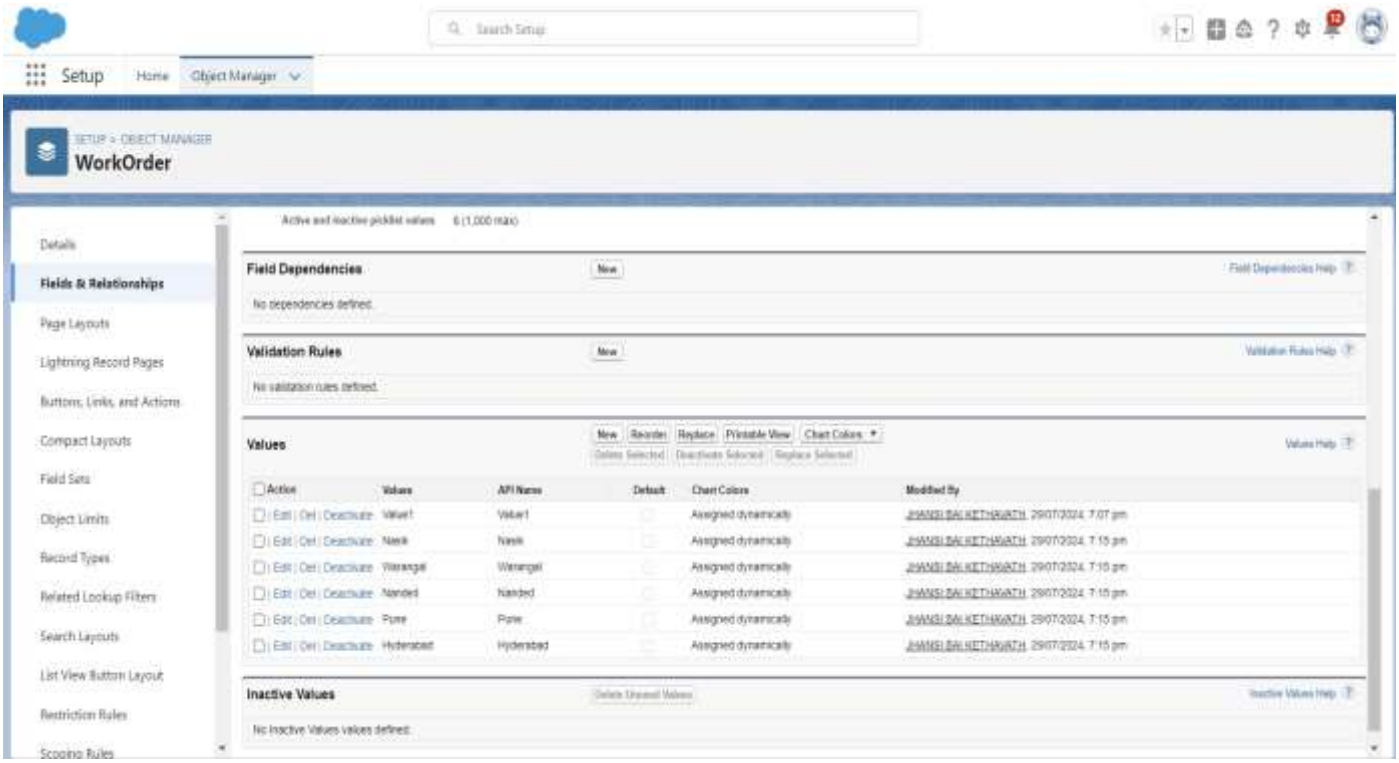
Task 4 :

4.1 Creating Lookup Field in Assignment Object

A lookup field in the Assignment Object establishes a relationship with another object, such as Technicians or Work Orders, enabling users to link and reference related records for improved data organization and relational tracking.



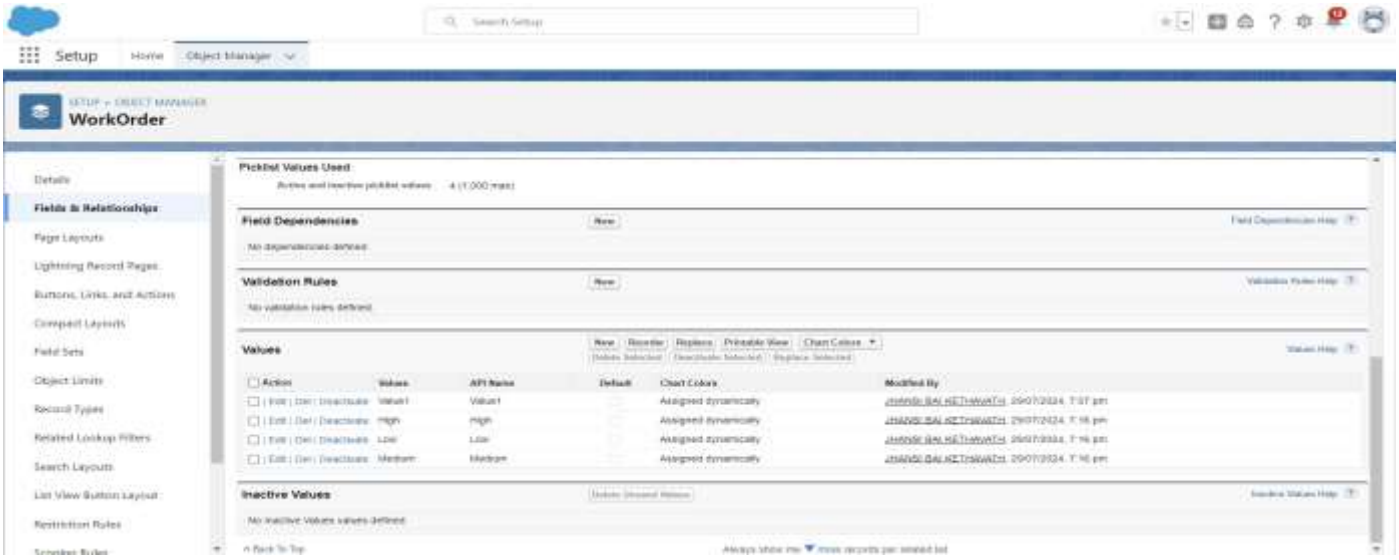
4.2 Manage your picklist values

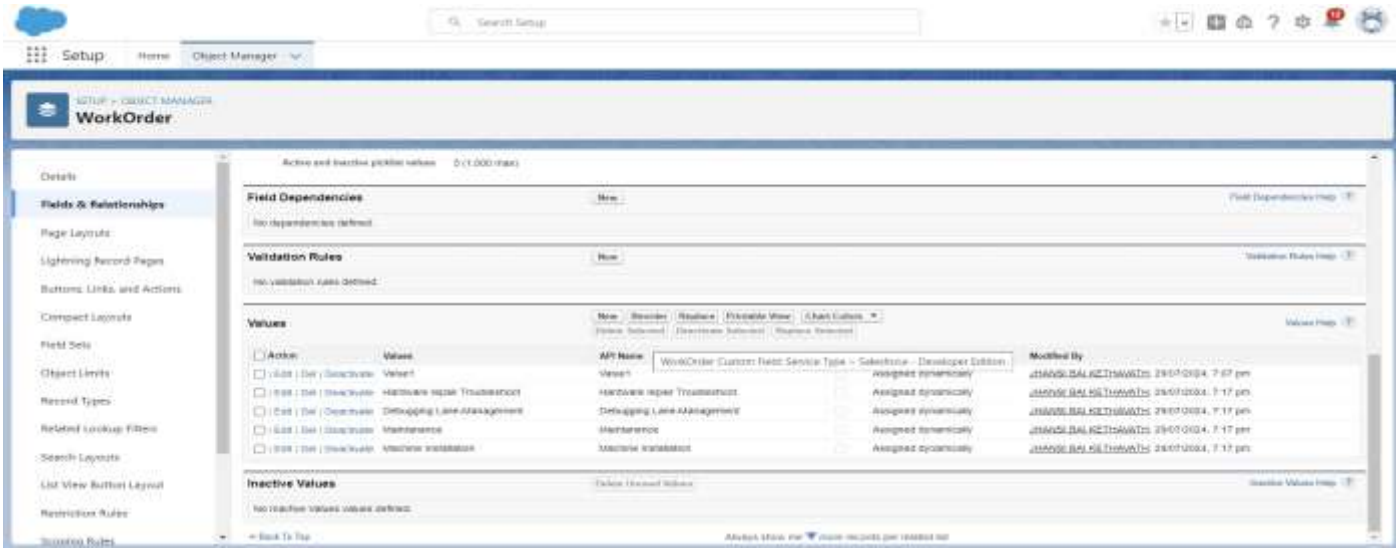


4.3 Manage your picklist values :

Add following values to the respective fields in WorkOrder object:

Field	Values
Priority	High
Service Type	Hardware repair Troubleshoot/Debugging Lane-Management



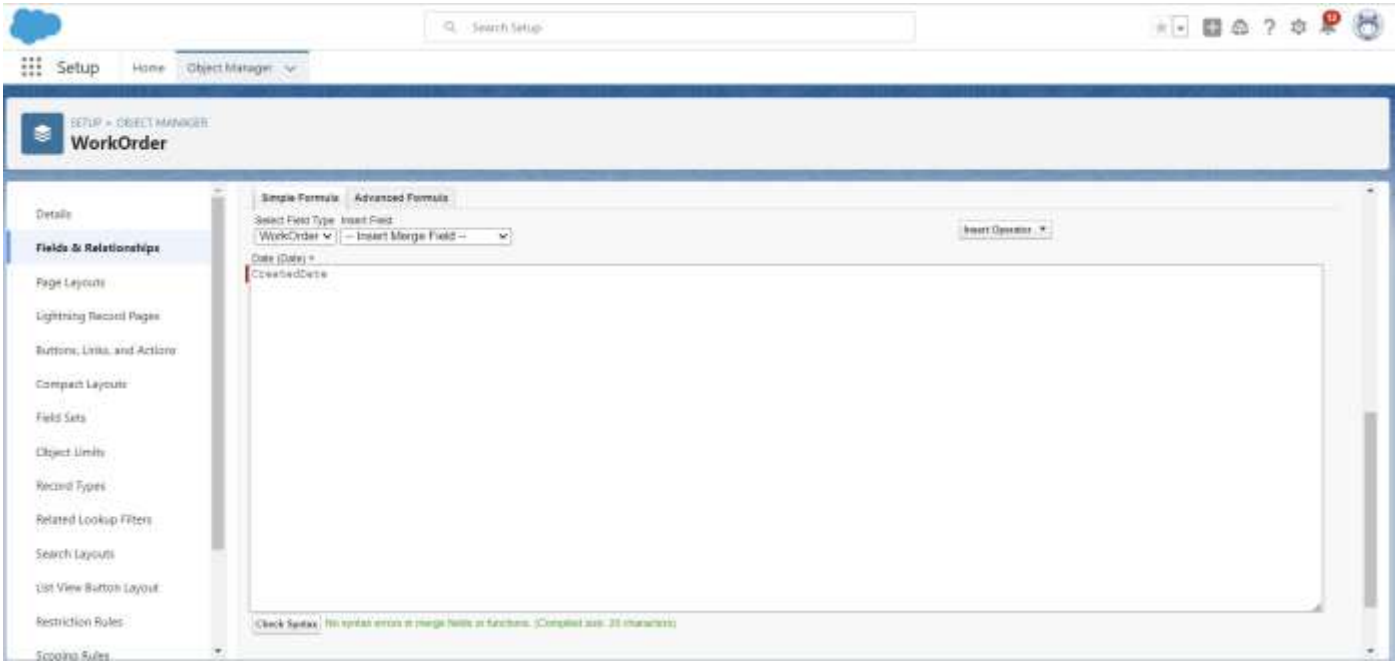


4.4 Creating Formula Field in WorkOrder Object

A formula field in the Work Order Object automatically calculates and displays data based on other fields or custom logic. This feature streamlines data entry, ensures consistency, and provides real-time insights without manual updates.

1. Repeat steps 1 and 2 mentioned in activity 1
2. Select Data type as “Formula” and click Next.
3. Give Field Label and Field Name as “Date” and select formula return type as “Date” and click next.
4. Under Advanced Formula, write the formula and click “Check Syntax”

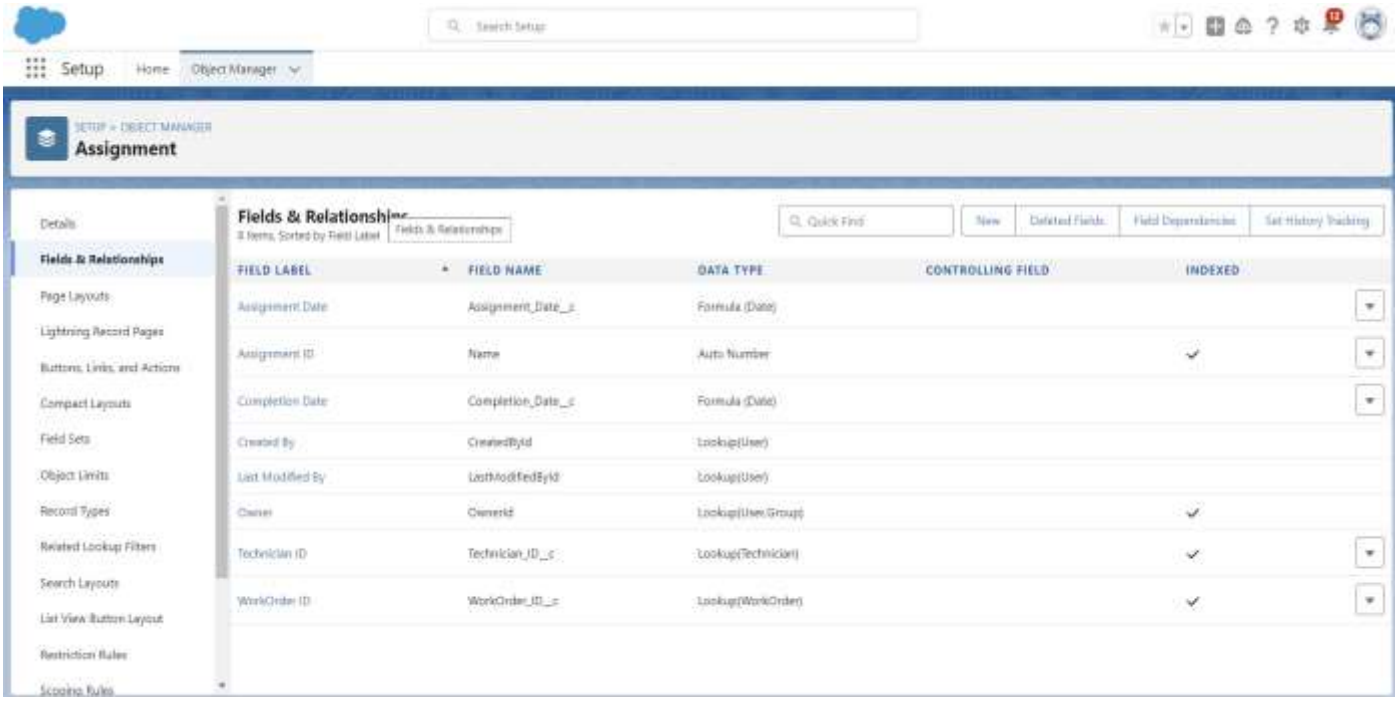
   Formula: CreatedDate
5. Next--> Next--> Save.



4.5 Creating Remaining fields for the respective objects

Now create the remaining fields using the data types mentioned in the table.

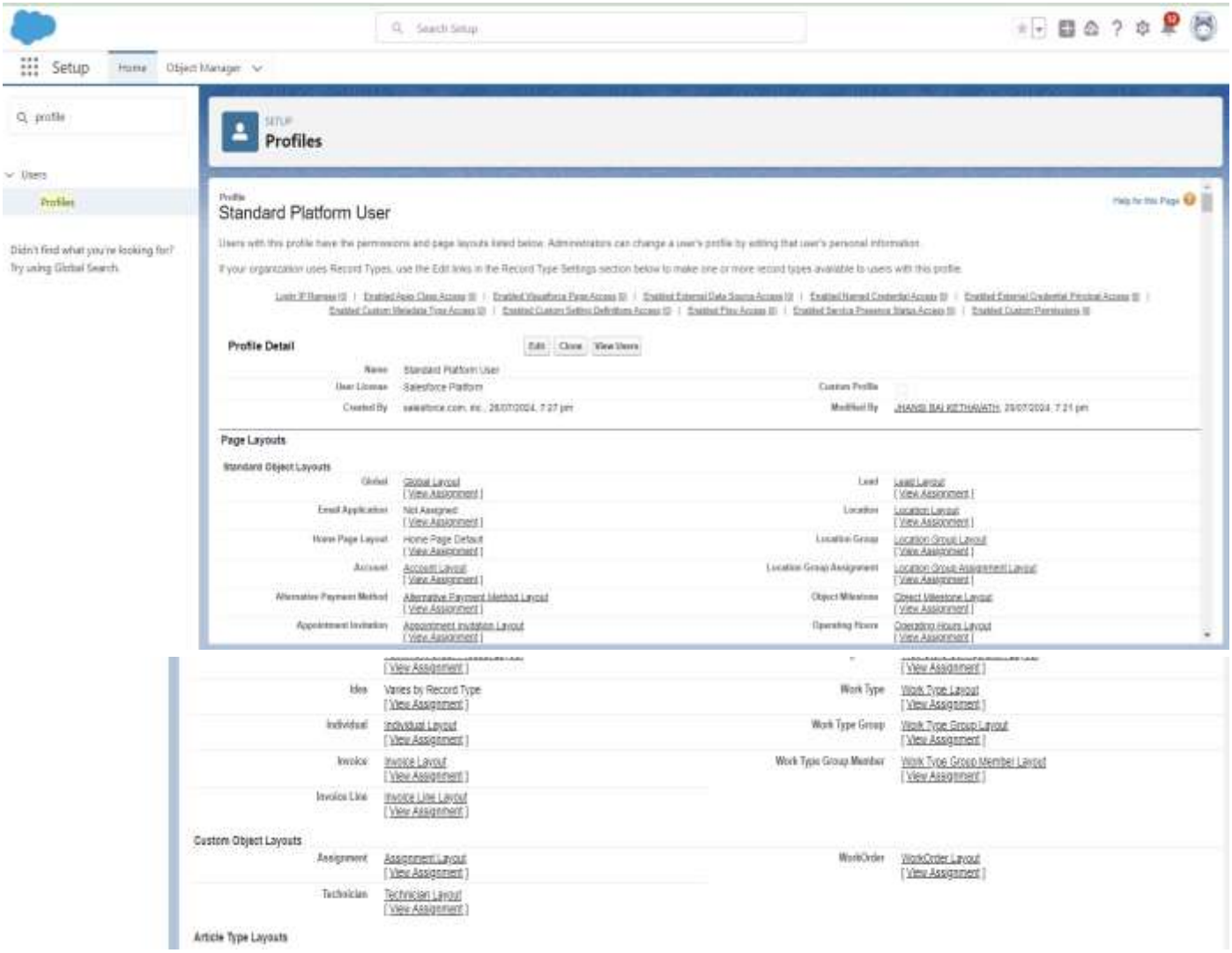
Sl No	Object Name	Field
1	Assignment	<table><tr><th>Field Name</th><th>Datatype</th></tr><tr><td>• Technician ID • Assignment Date • Completion Date</td><td>Lookup(Technician) Formula: return type : Date (WorkOrder_ID__r.Date__c) Formula: return type : Date IF(ISPICKVAL(WorkOrder_ID__r.Status__c , 'Resolved'), WorkOrder_ID__r.LastModifiedDate , NULL)</td></tr></table>



Task 5 :

Technician Profile

1. Go to setup --> type profiles in the quick find box --> click on profiles --> click on new profile.
2. Select ‘Standard Platform User’ for existing profile and give ‘Technician’ for Profile Name and click on Save.
3. While still on the profile page, then click Edit.
4. While still on the profile page, then click Edit.
5. Scroll down and Click on Save.
6. Now from the profile detail page scroll down to custom field level security click on view next to WorkOrder object.
7. Click on Edit, enable the check box for the status field.
8. Click on Save.



Task 6 :

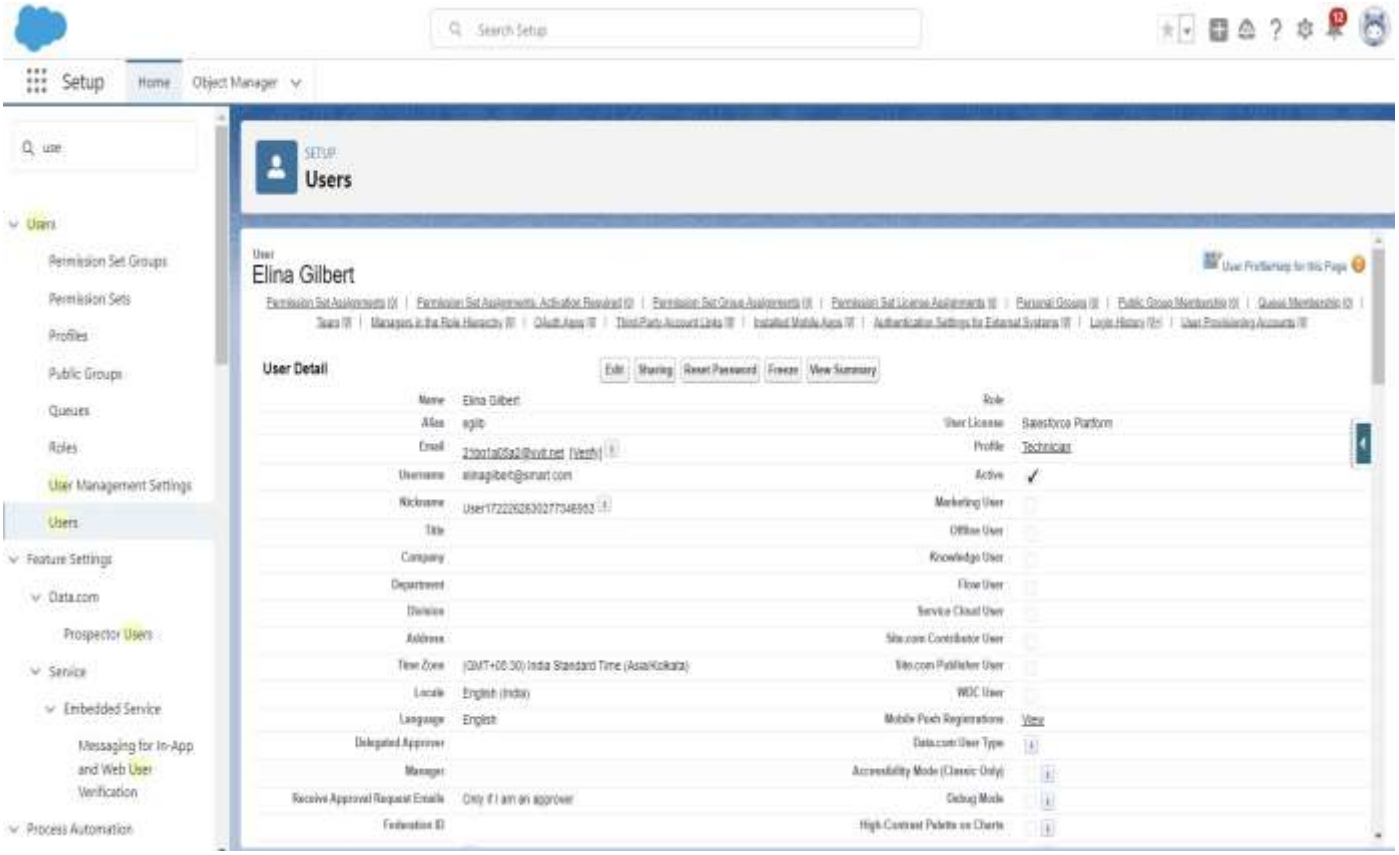
Create User

User is engaged in the Field Service Workforce Optimization Project, utilizing Salesforce to optimize field operations, improve resource management, and enhance customer service through efficient scheduling, real-time tracking, and comprehensive analytics.

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

1. First Name : Elina
2. Last Name : Gilbert
3. Alias : Give an Alias Name
4. Email id : Give your Personal Email id
5. Username : Username should be in this form: text@text.text
6. Nick Name : Give a Nickname
7. Role :
8. User license : Salesforce Platform

9. Profiles : Technician



Task 7 :

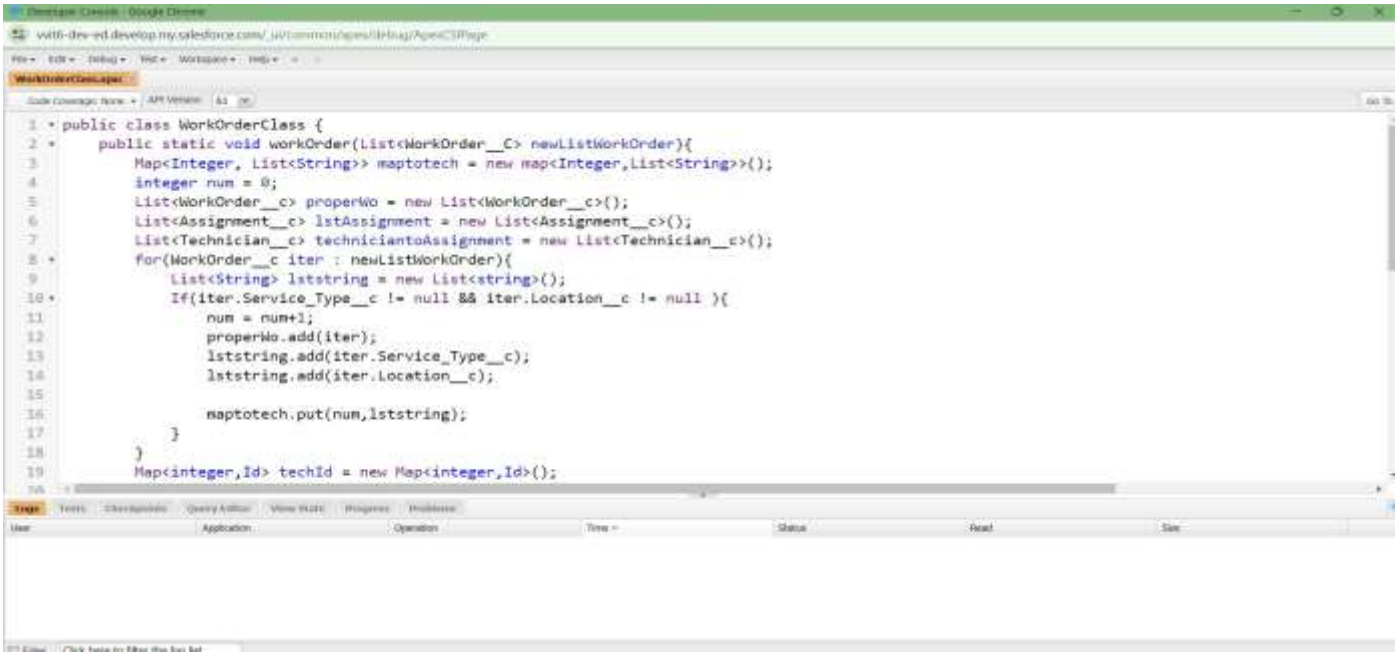
7.1 Create an Apex Class

1. Go to Setup --> Click on the gear icon --> Select Developer Console.
2. Then we can see the Developer console. Click on the developer console and you will navigate to a new console window.
3. To create a new Apex Class follow the below steps: Click on the file --> New --> Apex Class.
4. Give the Apex Class name as “WorkOrderClass”.
5. Click ok.
6. Now write the code logic here
7. **Source Code:**

```
public class WorkOrderClass {
   public static void workOrder(List<WorkOrder__C> newListWorkOrder){
      Map<Integer, List<String>> maptotech = new map<Integer,List<String>>();       integer
num = 0;
      List<WorkOrder__c> properWo = new List<WorkOrder__c>();
      List<Assignment__c> lstAssignment = new List<Assignment__c>();
      List<Technician__c> techniciantoAssignment = new List<Technician__c>();
      for(WorkOrder__c iter : newListWorkOrder){
         List<String> lststring = new List<string>();
         If(iter.Service_Type__c != null && iter.Location__c != null ){
```

```
            num = num+1;
            properWo.add(iter);
            lststring.add(iter.Service_Type__c);
            lststring.add(iter.Location__c);

            maptotech.put(num,lststring);
          }
        }
        Map<integer,Id> techId = new Map<integer,Id>();
        Map<Id,Technician__c> allTechnician = new Map<Id,Technician__c>([SELECT Id, Name,
Phone__c, Location__c, Skills__c, Availibility__c, Name__c, Email__c FROM Technician__c]);
integer num2 = 0;
        For(Technician__c T : allTechnician.values()){
          num2 = num2+1;
          if(maptotech.get(num2) != null){
            List<string> valofmap = maptotech.get(num2);
          system.debug('error 1 ----> the maptotech is empty ---> ' + maptotech.get(num2));
if(valofMap.contains(t.Skills__c) && ValofMap.contains(t.Location__c) && t.Availibility__c ==
'Available'){              techid.put(num2,t.Id);
         }
          }

        }
        integer num3 = 0;
        For(WorkOrder__c W : properWo){
          num3 = num3 + 1;
          Assignment__c A = new Assignment__c();
          A.WorkOrder_ID__c = W.Id;
          A.Technician_ID__c = techid.get(num3);
          lstAssignment.add(A);
        }
        If(!lstAssignment.IsEmpty()){
          insert lstAssignment;
        }
    }
}
```

8. Save the code.(click on file --> Save)



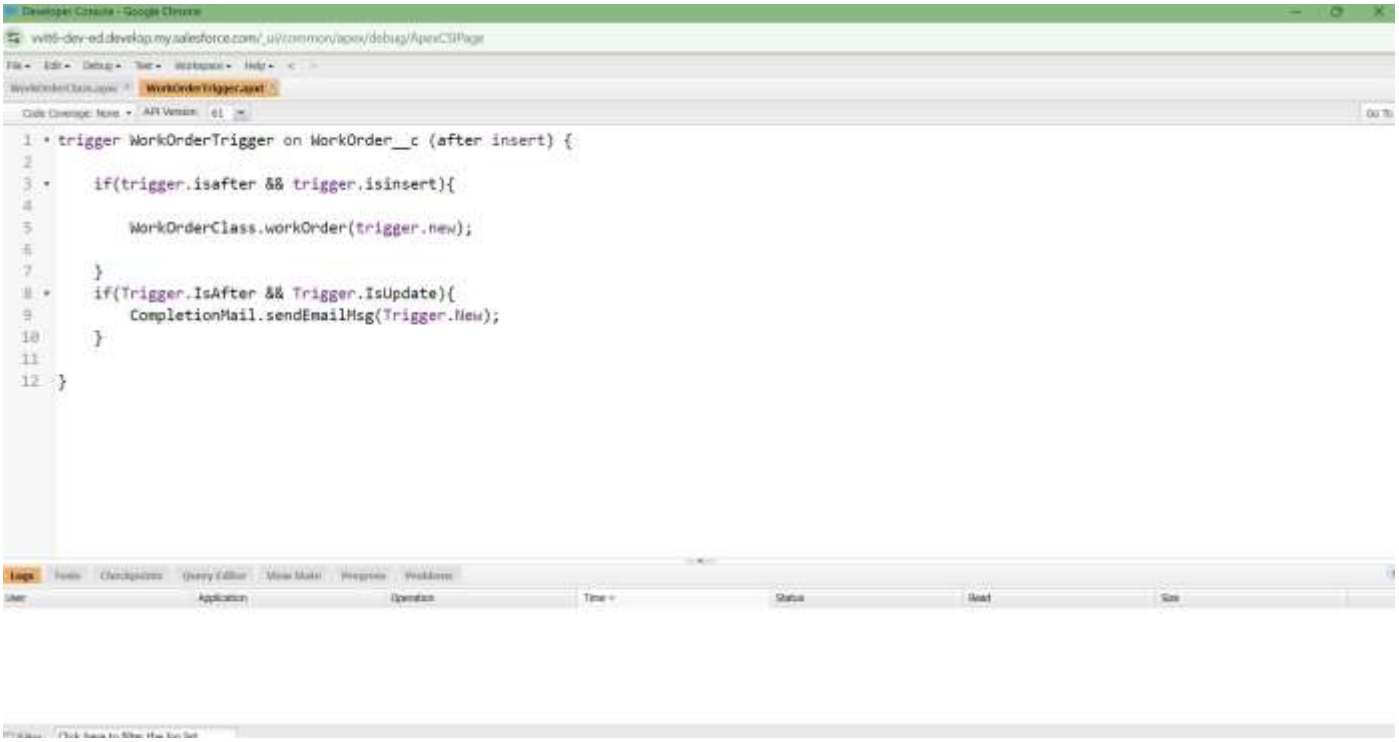
```
public class WorkOrderClass {
    public static void workOrder(List<WorkOrder__C> newListWorkOrder){
        Map<Integer, List<String>> maptotech = new map<Integer,List<String>>();
        integer num = 0;
        List<WorkOrder__c> properWo = new List<WorkOrder__c>();
        List<Assignment__c> lstAssignment = new List<Assignment__c>();
        List<Technician__c> techniciantoAssignment = new List<Technician__c>();
        for(WorkOrder__c iter : newListWorkOrder){
            List<String> lststring = new List<string>();
            If(iter.Service_Type__c != null && iter.Location__c != null ){
                num = num+1;
                properWo.add(iter);
                lststring.add(iter.Service_Type__c);
                lststring.add(iter.Location__c);

                maptotech.put(num,lststring);
            }
        }
        Map<integer,Id> techId = new Map<integer,Id>();
```

7.2 Create an Apex Trigger

1. To create a new Apex Class follow the below steps:
   Click on the file --> New --> Apex Class.
2. Give the Apex Trigger name as “WorkOrderTrigger”, and select “WorkOrder__c” from the dropdown for sObject.
3. Click Submit.
4. Now write the code logic here **Source Code:** trigger WorkOrderTrigger on WorkOrder__c (after insert) {

```
if(trigger.isafter && trigger.isinsert){
    WorkOrderClass.workOrder(trigger.new);
}
}
```

5. Save the code.(click on file --> Save)



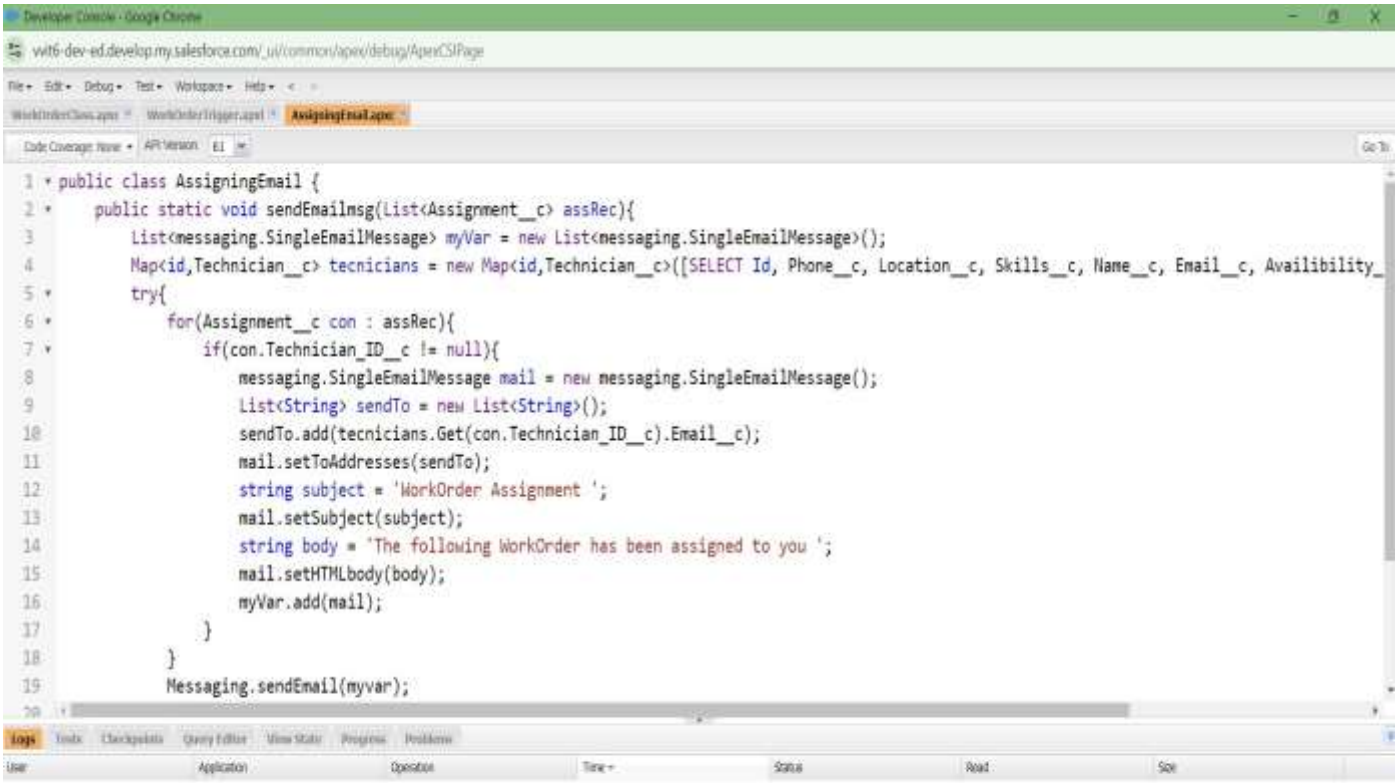
7.3 Create an Apex Class

1. Go to Setup --> Click on the gear icon --> Select Developer Console.
2. Then we can see the Developer console. Click on the developer console and you will navigate to a new console window.
3. To create a new Apex Class follow the below steps: Click on the file --> New --> Apex Class.
4. Give the Apex Class name as “AssigningEmail”.
5. Click ok.
6. Now write the code logic here
7. **Source Code:**

```
public class AssigningEmail {
    public static void sendEmailmsg(List<Assignment__c> assRec){
        List<messaging.SingleEmailMessage> myVar = new
List<messaging.SingleEmailMessage>();
        Map<id,Technician__c> tecnicians = new Map<id,Technician__c>([SELECT Id, Phone__c,
Location__c, Skills__c, Name__c, Email__c, Availibility__c, Name FROM Technician__c]);
        try{
            for(Assignment__c con : assRec){
                if(con.Technician_ID__c != null){
                    messaging.SingleEmailMessage mail = new messaging.SingleEmailMessage();
                    List<String> sendTo = new List<String>();
```

```
                sendTo.add(tecnicians.Get(con.Technician_ID__c).Email__c);
                mail.setToAddresses(sendTo);
                string subject = 'WorkOrder Assignment ';
                mail.setSubject(subject);
                string body = 'The following WorkOrder has been assigned to you ';
                mail.setHTMLbody(body);
                myVar.add(mail);
              }
            }
            Messaging.sendEmail(myvar);
        }
        catch(exception e){
            system.debug('Error -----> ' + e.getMessage());
        }
    }
}
```

8. Save the code.(click on file --> Save)



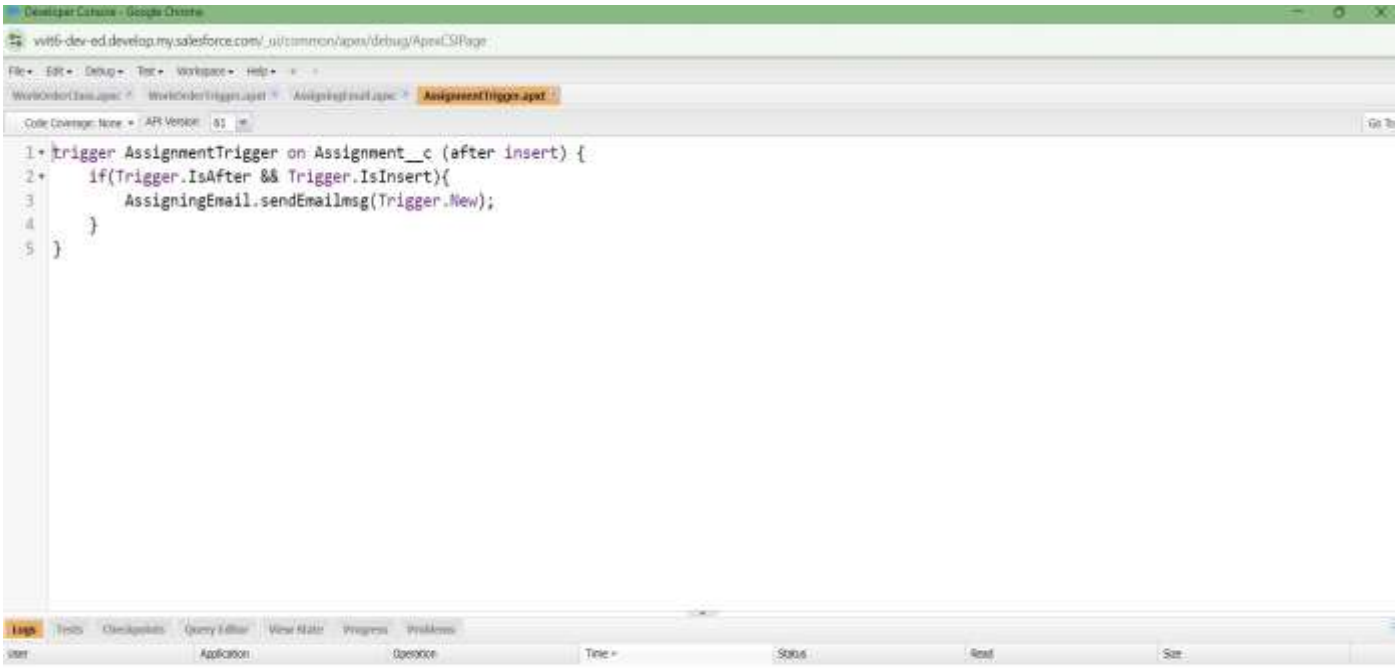
7.4 Create an Apex Trigger

To create a new Apex Class follow the below steps:

1. Click on the file --> New --> Apex Class.
2. Give the Apex Trigger name as “AssignmentTrigger”, and select “Assignment__c” from the dropdown for sObject.
3. Click Submit.
4. Now write the code logic here
5. **Source Code:**

```
trigger AssignmentTrigger on Assignment__c (after insert) {
   if(Trigger.IsAfter && Trigger.IsInsert){
      AssigningEmail.sendEmailmsg(Trigger.New);
   }
}
```

6. Save the code.(click on file --> Save)



7.5 Create an Apex Class

1. Go to Setup --> Click on the gear icon --> Select Developer Console.
2. Then we can see the Developer console. Click on the developer console and you will navigate to a new console window.
3. To create a new Apex Class follow the below steps: Click on the file --> New --> Apex Class.
4. Give the Apex Class name as “CompletionMail”.
5. Click ok.
6. Now write the code logic here
7. **Source Code:**

```
public class CompletionMail {
    public static void sendEmailMsg(List<WorkOrder__c> workOrderList){
        List<messaging.SingleEmailMessage> myVar = new
List<messaging.SingleEmailMessage>();
        for(WorkOrder__c con : workOrderList){
            if(con.Status__c == 'Resolved'){
                messaging.SingleEmailMessage mail = new messaging.SingleEmailMessage();
                List<String> sendTo = new List<String>();
                sendTo.add(con.Email__c);
                mail.setToAddresses(sendTo);
                string subject = 'Status Updated';
```

```
            mail.setSubject(subject);
            string body = 'email body ';
            mail.setHTMLbody(body);
            myVar.add(mail);
          }
        }
        Messaging.sendEmail(myvar);
      }
    }
```

8. Save the code.(click on file --> Save)

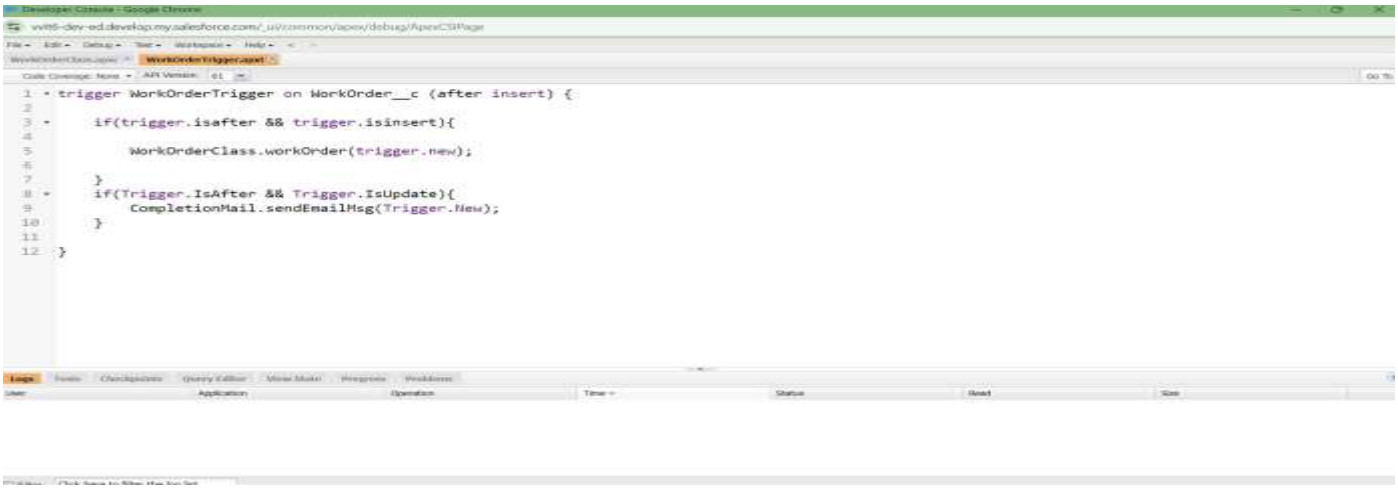
```
public class CompletionMail {
    public static void sendEmailMsg(List<WorkOrder__c> workOrderList){
        List<messaging.SingleEmailMessage> myVar = new List<messaging.SingleEmailMessage>();
        for(WorkOrder__c con : workOrderList){
            if(con.Status__c == 'Resolved'){
                messaging.SingleEmailMessage mail = new messaging.SingleEmailMessage();
                List<String> sendTo = new List<String>();
                sendTo.add(con.Email__c);
                mail.setToAddresses(sendTo);
                string subject = 'Status Updated';
                mail.setSubject(subject);
                string body = 'email body ';
                mail.setHTMLbody(body);
                myVar.add(mail);
            }
        }
        Messaging.sendEmail(myvar);
    }
}
```

7.6 Create an Apex Trigger

1. Click on the file --> Open.
2. A pop up window opens click on Triggers, then select “WorkOrderTrigger” and click on “Open”
3. Now write the code logic here.
4. WorkOrderClass.workOrder(trigger.new);

```
    }
    if(Trigger.IsAfter && Trigger.IsUpdate){
        CompletionMail.sendEmailMsg(Trigger.New);
    }
}
```

5. Save the code.(click on file --> Save)



7.7 Create an Asynchronous Apex Class

Create an Apex Class to Delete all the WorkOrder records which meets the following criteriaL

1. Completed date should be more than 30 days.
2. Status should be 'Resolved'. Create an Apex Class

1. Go to Setup --> Click on the gear icon --> Select Developer Console.
2. Then we can see the Developer console. Click on the developer console and you will navigate to a new console window.
3. To create a new Apex Class follow the below steps: Click on the file --> New --> Apex Class.
4. Give the Apex Class name as "RecordDeletion".
5. Click ok.
6. Now write the code logic here

```
public class RecordDeletions Implements Database.Batchable<Sobject>{
  public Database.QueryLocator start(Database.BatchableContext bc) { string query =
      'SELECT Id, Name, WorkOrder_ID__c, Technician_ID__c,
          Assignment_Date__c, Completion_Date__c FROM Assignment__c WHERE
       Completion_Date__c = LAST_N_DAYS:30';
     return database.GetQueryLocator(query);
  }
  public void execute(Database.BatchableContext bc, List<Assignment__c> query){
    if(!Query.IsEmpty()){
      Delete Query;
    }
  }
  public void finish(Database.BatchableContext bc){
   }
}
```

7. Save the code.(click on file --> Save)

```
public class RecordDeletions Implements Database.Batchable<Sobject>{

    public Database.QueryLocator start(Database.BatchableContext bc) {

string query = 'SELECT Id, Name, WorkOrder_ID__c, Technician_ID__c, Assignment_Date__c, Completion_Date__c FROM Assignment__c WHERE Completion_Date

        return database.GetQueryLocator(query);

    }

    public void execute(Database.BatchableContext bc, list<Assignment__c> query){

        if(!Query.IsEmpty()){

            Delete Query;

        }

    }
```

7.8 Create an Apex Schedule Class

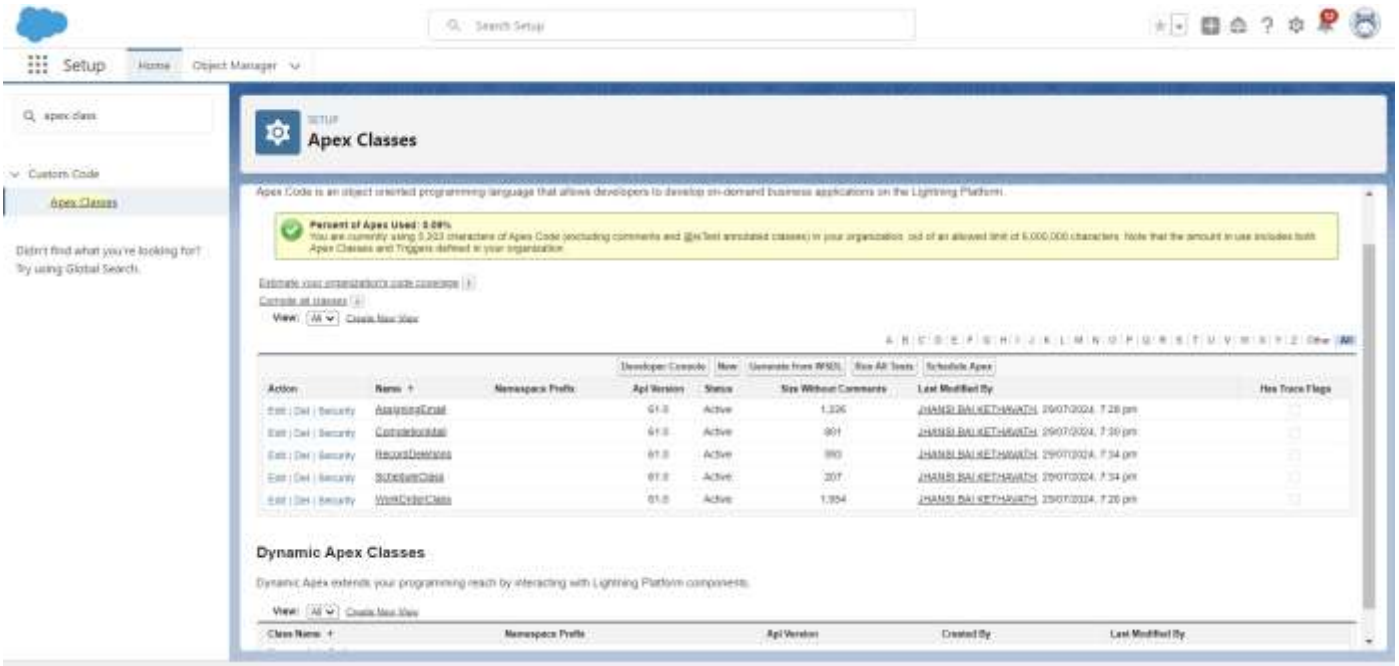
1. Go to Setup --> Click on the gear icon --> Select Developer Console.
2. Then we can see the Developer console. Click on the developer console and you will navigate to a new console window.
3. To create a new Apex Class follow the below steps: Click on the file --> New --> Apex Class.
4. Give the Apex Class name as “ScheduleClass”.
5. Click ok.
6. Now write the code logic here **Source Code:** global class ScheduleClass implements Schedulable {

```
global void execute(SchedulableContext SC) {
    RecordDeletions delrec = new RecordDeletions();
    database.executeBatch(delrec, 200);
  }
}
```

7. Save the code.(click on file ? Save)

7.9 Create a Schedule Apex Schedule the Apex class:

1. From the Setup page search for “Apex Classes” in quick search.
2. Click on “Schedule Apex” as shown below.
3. Click on Schedule Apex and enter the Job name.
4. Job Name : DeleteAssignmentSchedule
5. Apex Class : ScheduleClass (from clicking on lookup icon)
6. Frequency : Monthly
7. Preferred Start Time : Select any time
8. Click Save.



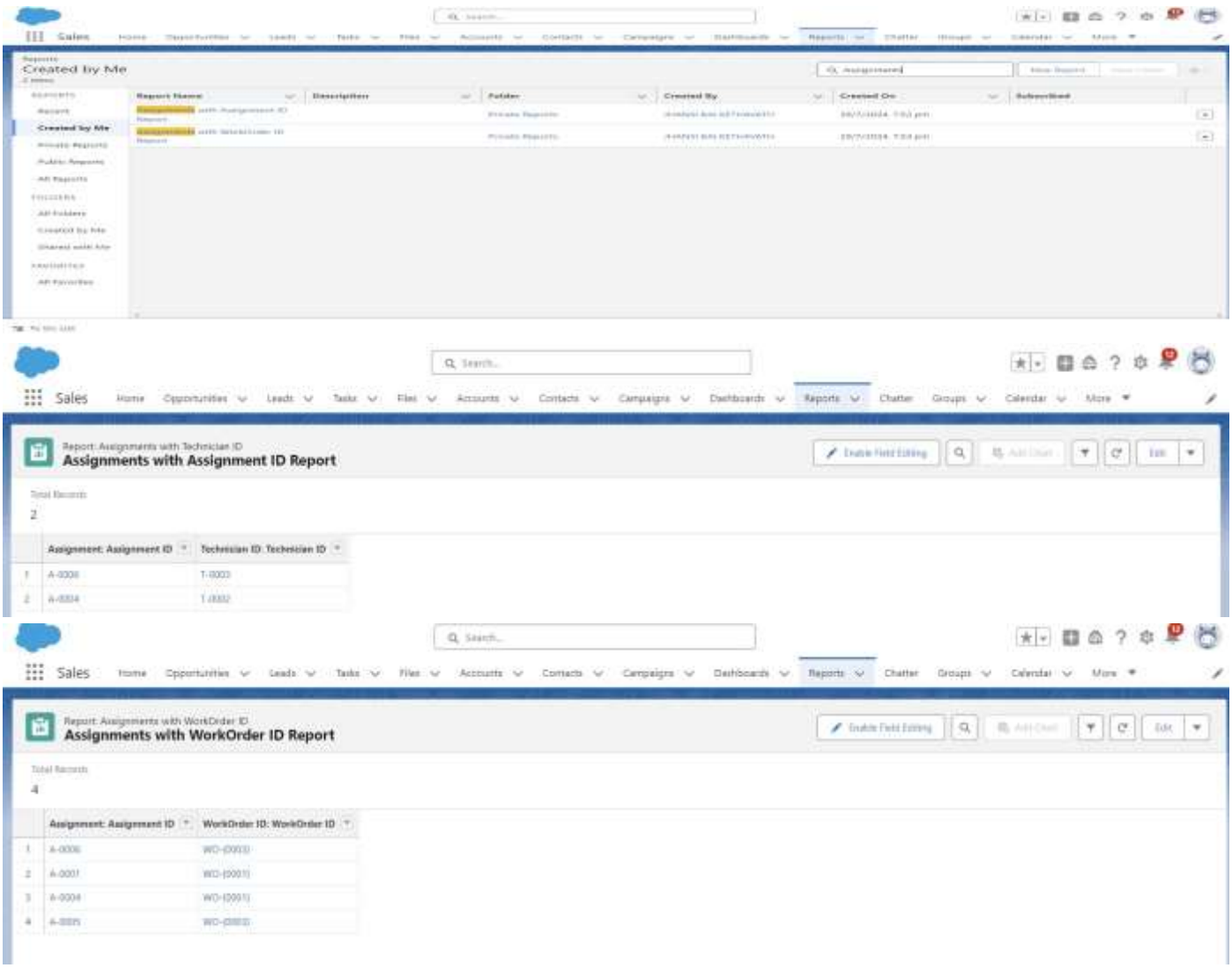
Task 8 :

8.1 Report

1. Go to the app --> click on the reports tab

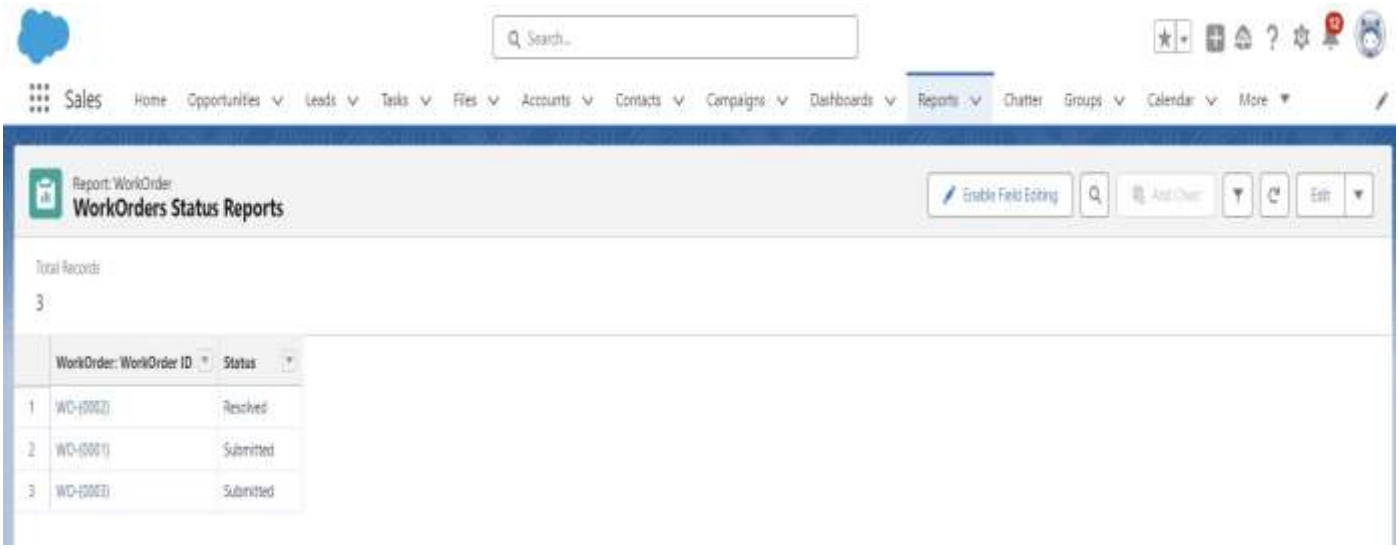
2. Click New Report.
3. Select report type from category or from report type panel or from search panel --> click on start report.
4. Customize your report
5. Add fields from left pane as shown below
6. Grouped by workorder ID
7. Save or run it.

Note: Reports may get varied from the above pictures as the data might be different.

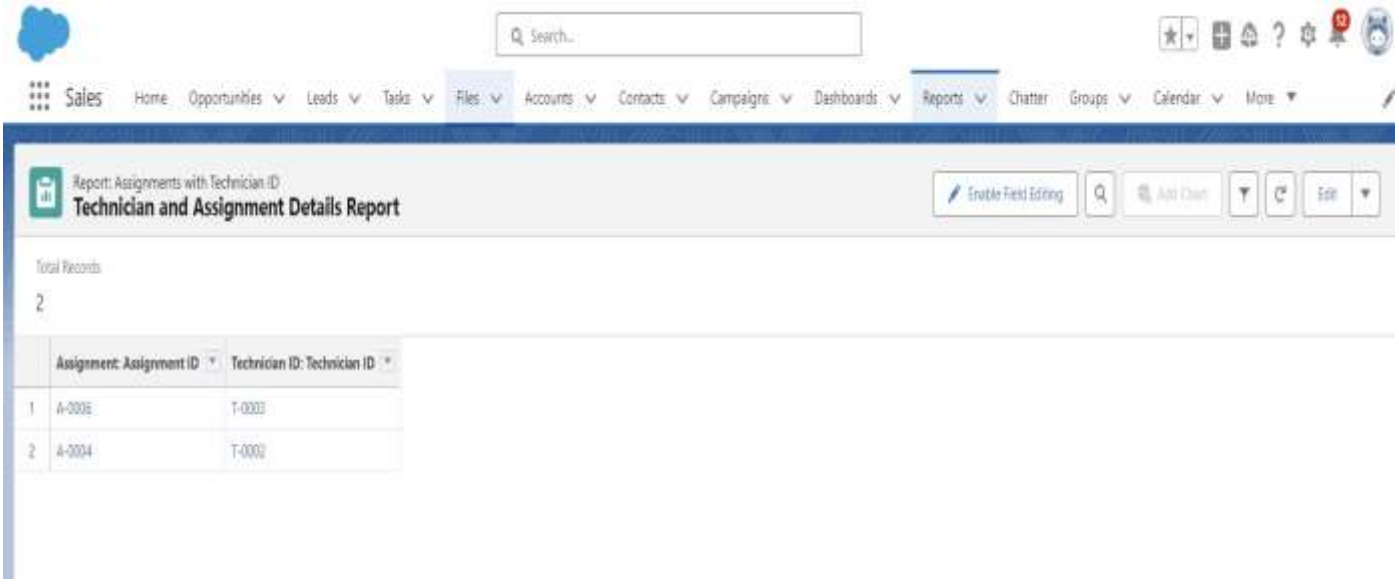


8.2 Create Reports

1. Create a report with report type: “WorkOrders Status Reports”.

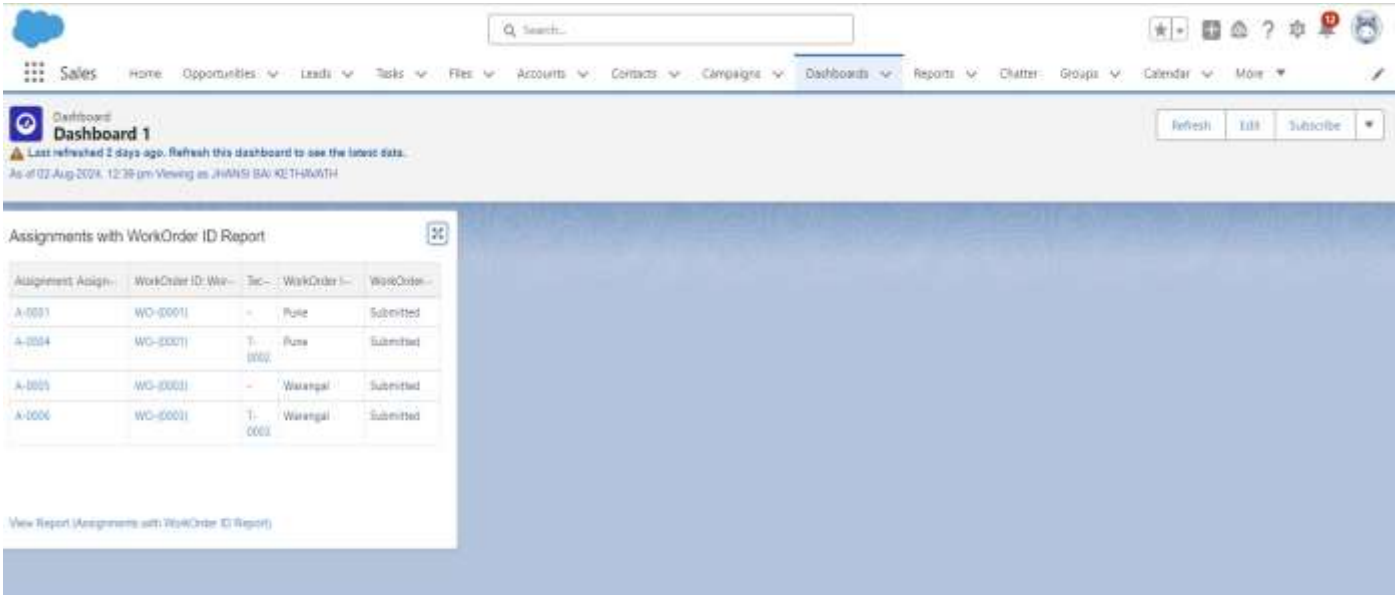


2. Create a report with report type: “Technician and Assignment Details Reports”.



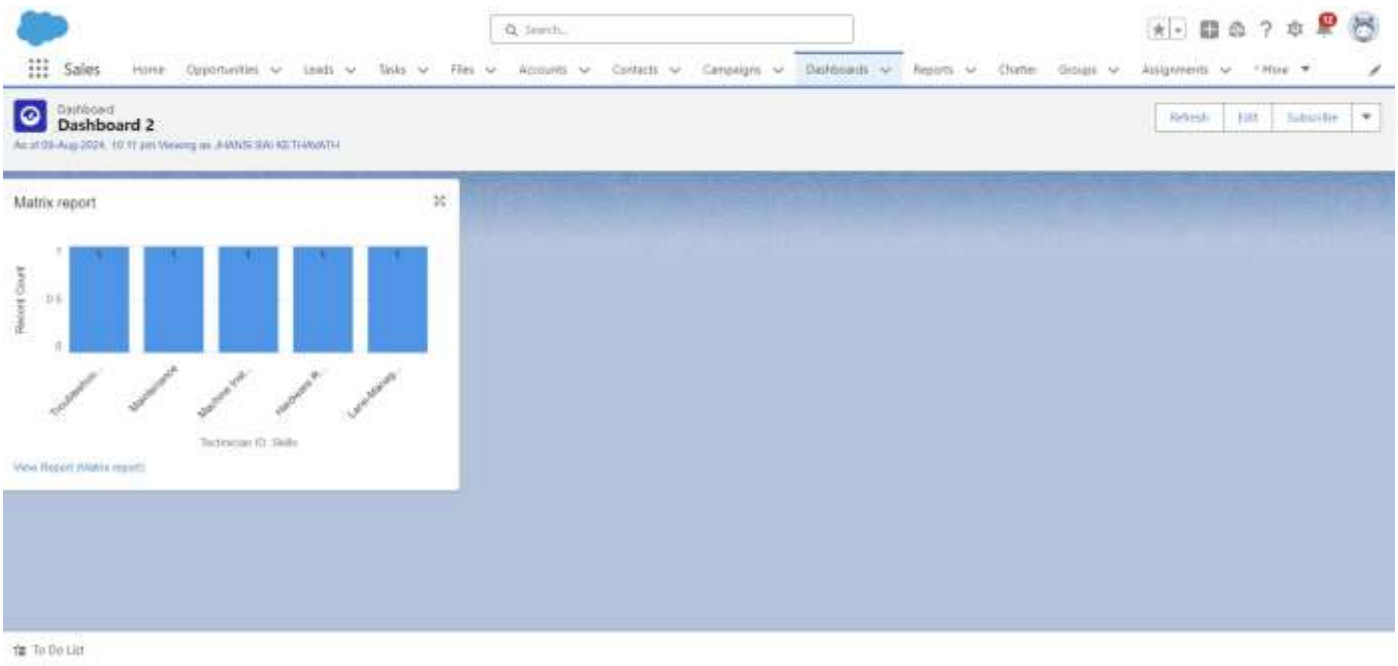
8.3 Dashboard

1. Go to the app --> click on the Dashboards tabs.
2. Give a Name and click on Create.
3. Select add component.
4. Select a Report which we have created in the previous activities and click on select.
5. Click Add then click on Save and then click on Done.



8.4 Create Dashboards

Create another Dashboard as we discussed in activity 3 which shows the details of completed workorder status in a vertical bar graph.




Thank
you