Field Service WorkOrder Optimization

1. Project Overview

The Field Service Work Order Optimization System is designed to improve operations for companies specializing in installation and repair services. Utilizing a robust database and intelligent algorithms, the system assigns work orders to skilled technicians based on location, availability, and expertise.

The system prioritizes tasks strategically to maximize efficiency and minimize delays. Automated communication ensures real-time updates for technicians, fostering seamless coordination. Additionally, analytics tools deliver actionable insights to support continuous improvement.

This solution aims to enhance operational efficiency, reduce costs, and improve customer satisfaction in the dynamic field service industry.

2. Objectives

Business Goals:

Operational Goals:

- Optimize Task Scheduling
  - Automate work order assignments to minimize delays and improve service delivery times.
  - Allocate tasks to the most qualified and available technicians for maximum efficiency.
- Improve Resource Allocation
  - Assign tasks based on technicians' expertise and proximity to the work location.
  - Minimize idle time and maximize productivity with better resource utilization.
- Enhance Communication and Coordination
  - Use real-time updates to ensure seamless communication between dispatch teams and field technicians.
  - Enable technicians to update work orders on-the-go via mobile access.
- Deliver Excellent Customer Experiences
  - Ensure timely service completion to improve customer ratings and reduce complaints.
  - Provide customers with real-time updates on technician arrivals and service schedules.
- Enable Strategic Decision-Making
  - Leverage analytics to monitor key performance indicators (KPIs) and identify bottlenecks.
  - Use data insights to refine workflows and continuously improve service quality.
- Support Growth and Scalability
  - Design the system to handle increasing workloads as the business grows.
  - Ensure seamless integration with enterprise systems to meet evolving needs.

Key Outcomes:

- Efficient Task Assignments:
  - Reduce the average time required to assign work orders.
  - Increase the percentage of tasks assigned to the most suitable technicians.

- **Reduced Operational Costs:**
    - Lower fuel and travel expenses with optimized task scheduling and routing.
    - Decrease administrative overhead through automation.
- **Enhanced Technician Productivity:**
    - Boost the number of tasks completed per technician per day.
    - Reduce downtime and enhance resource utilization.
- **Improved Customer Satisfaction:**
    - Achieve higher customer ratings with timely and reliable service.
    - Minimize complaints related to delayed or incomplete tasks.
- **Data-Driven Improvements:**
    - Track metrics such as resolution time and first-time fix rates.
    - Identify trends and areas for improvement with detailed analytics.
- **Scalable Operations:**
    - Maintain system efficiency with increasing workloads.
    - Adapt to changing business needs with flexible integration options.

3. Salesforce Key Features and Concepts Utilized

1. **Custom Objects and Relationships**

    a. **Work Order**, **Technician**, and **Assignment** objects created to model the field service process.

    b. Relationships:

    i. Lookup relationships between Work Order → Technician and Assignment → Work Order/Technician for linking related records.

2. **Data Modeling and Fields**

    a. Custom fields added to capture essential information:

    i. **Work Order**: Status, Priority, Service Type, and Description.

    ii. **Technician**: Availability, Skills, Location, and Contact Details.

    iii. **Assignment**: Assignment Date, Completion Date, Technician ID, and Work Order ID.

3. **UI Customization**

a. **Tabs**: Custom tabs created for easy access to Work Order, Technician, and Assignment records.

b. **Lightning App**: Consolidated these tabs into a unified interface for streamlined navigation.

4. Automation with Apex

   a. **Apex Triggers**:

      i. Automated status updates (e.g., updating Work Order status when Assignment is marked completed).

      ii. Ensured Technician availability is updated after an Assignment is completed.

   b. **Apex Classes**:

      i. Implemented business logic for assigning technicians based on location, availability, and skills.

      ii. Utility methods for efficient operations and system integration.

5. Reports and Dashboards

   a. Created comprehensive **Reports**:

      i. Monitor open Work Orders by status and priority.

      ii. Technician performance metrics like task completion rate.

      iii. Assignments completed within specified time frames.

   b. Built **Dashboards** for real-time visualization of KPIs, including workload distribution, service efficiency, and customer satisfaction metrics.

6. Standard Salesforce Features

   a. **Profiles and Roles**: Defined access levels to secure sensitive data and limit access based on user roles (e.g., Dispatcher, Technician).

   b. **Record Ownership**: Used the Owner field to track accountability for Work Orders and Assignments.

   c. **Chatter**: Enabled team collaboration on Work Order records for updates and discussions.

7. Picklists and Validation

   a. Standardized input with picklists for fields like Status, Priority, Service Type, and Technician Availability.

   b. Added validation rules to ensure data consistency (e.g., mandatory fields before completing a Work Order).

8. Analytics and Metrics

   a. **Custom Dashboards**:

      i. Track ongoing Assignments, high-priority Work Orders, and technician workload.

      ii. KPIs like first-time resolution rate and average response time visualized effectively.

9. Mobile Accessibility

   a. Leveraged Salesforce’s mobile-ready features for technicians to access Assignments and update statuses in the field.

10. Security and Compliance

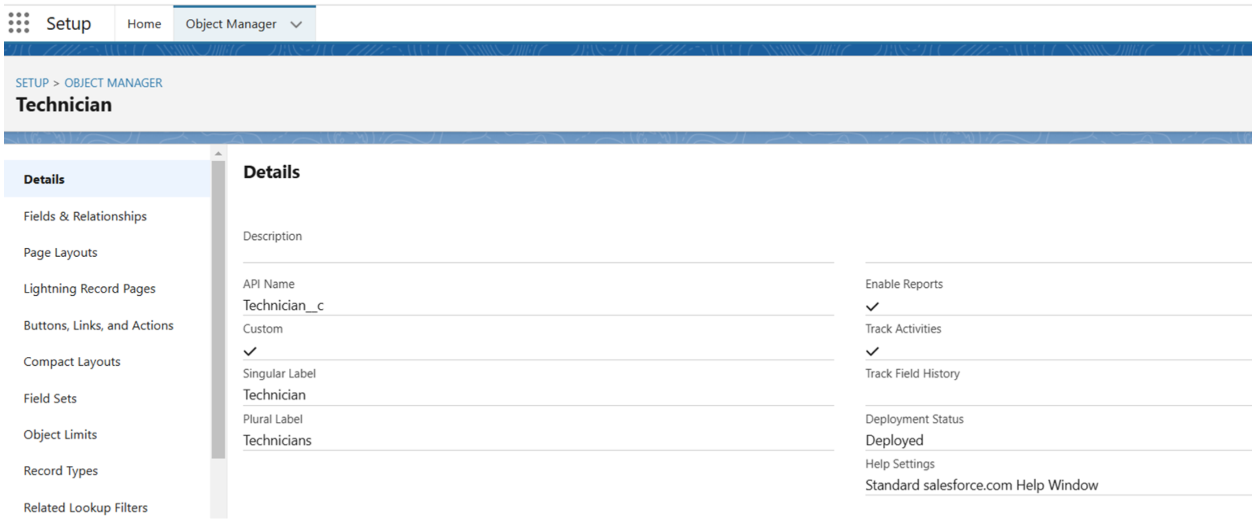
   a. Implemented role-based access control to secure records based on profiles.

   b. Used field-level security to protect sensitive data like customer contact details.

4. Detailed Steps to Solution Design

Create Objects From Spreadsheet

1. **Create Technician Object:**The **Technician** object stores critical details about field technicians, including their availability, skills, and location. It features key fields such as Technician ID, Name, Email, Phone, Availability (Picklist), Location (Picklist), and Skills (Picklist). Relationships are established with the Work Order and Assignment objects via lookup fields to link technicians to specific tasks and assignments.



Technician

Details

Description

API Name
Technician__c

Custom
✓

Singular Label
Technician

Plural Label
Technicians

Enable Reports
✓

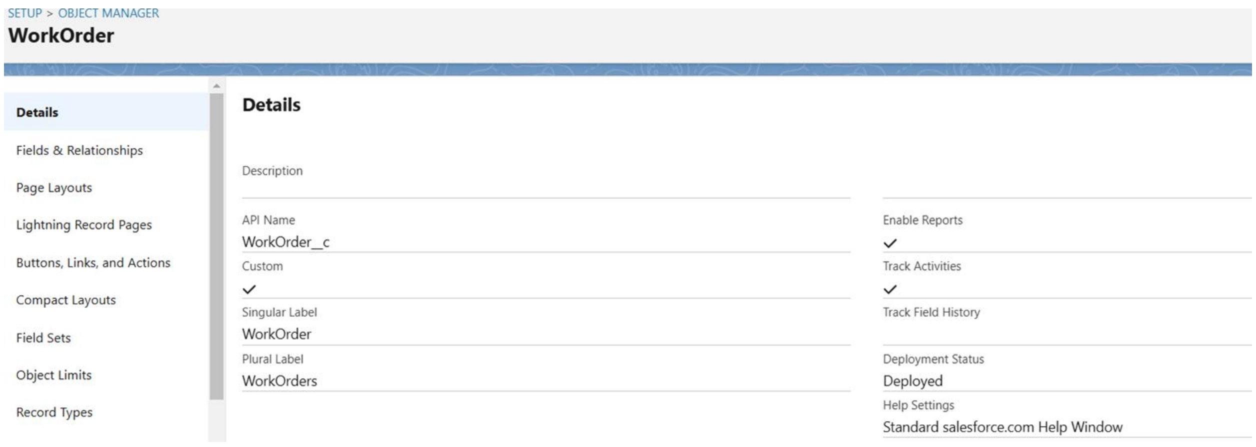
Track Activities
✓

Track Field History

Deployment Status
Deployed

Help Settings
Standard salesforce.com Help Window

1. **Create WorkOrder Object:**The **WorkOrder** object represents service tasks, such as installations or repairs. It includes key fields like Work Order ID (Auto Number), Date, Description, Status (Picklist), Priority (Picklist), Service Type (Picklist), Location, and Email. Relationships are established with Technician and Assignment objects to link tasks to the appropriate resources.



WorkOrder

Details

Description

API Name
WorkOrder__c

Custom
✓

Singular Label
WorkOrder

Plural Label
WorkOrders

Enable Reports
✓

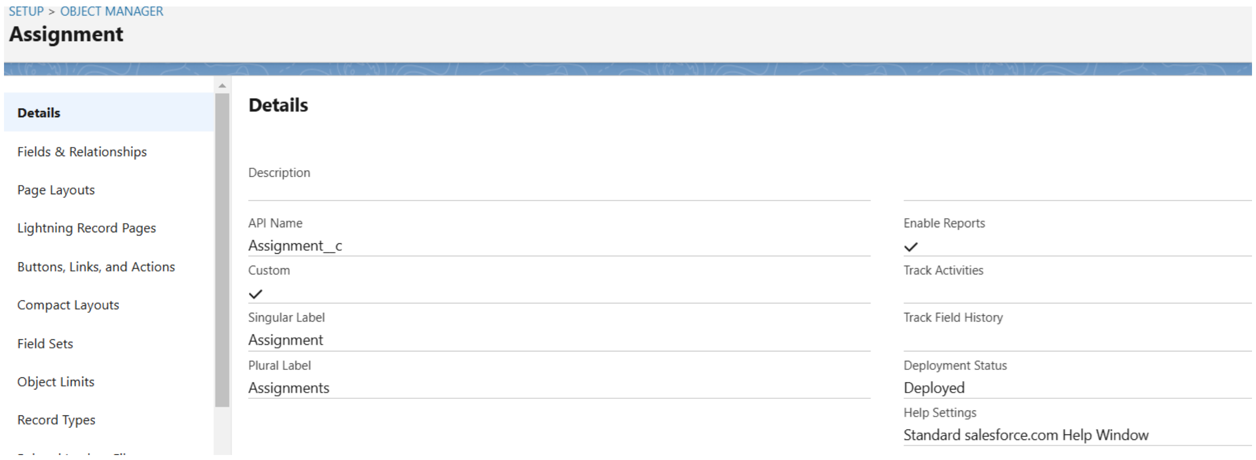
Track Activities
✓

Track Field History

Deployment Status
Deployed

Help Settings
Standard salesforce.com Help Window

1. **Create Assignment Object:**The **Assignment** object tracks the allocation of technicians to specific work orders. It includes fields like Assignment ID (Auto Number), Assignment Date, Completion Date, Technician ID (Lookup), and Work Order ID (Lookup). This object links technicians to their assigned tasks, ensuring proper tracking of service completion.



SETUP > OBJECT MANAGER

Assignment

- Details
- Fields & Relationships
- Page Layouts
- Lightning Record Pages
- Buttons, Links, and Actions
- Compact Layouts
- Field Sets
- Object Limits
- Record Types

Details

Description

API Name	Assignment__c
Custom	✓
Singular Label	Assignment
Plural Label	Assignments
Enable Reports	✓
Track Activities	
Track Field History	
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

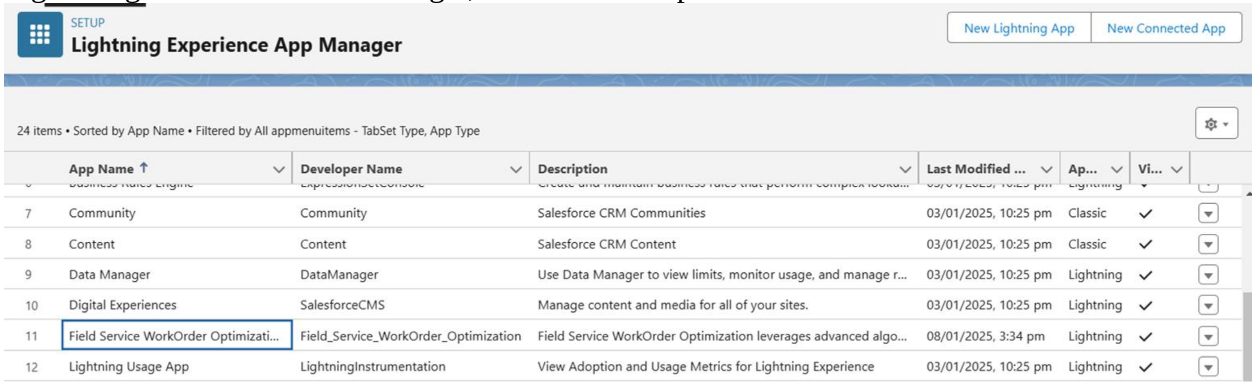
Create A Custom Tab

1. **Assignment Object:**The **Assignment** tab is created to provide easy access to Assignment records. It allows users to view, manage, and track technician assignments for work orders. The tab is added to a **Lightning App** for seamless integration into the user interface, ensuring efficient navigation and task management.

1. **WorkOrder and Technician Object:**The **Work Order** and **Technician** tabs are **default created** as these objects are imported from a spreadsheet.

Create The Lightning App

A **Lightning App** is created to consolidate the **Work Order**, **Technician**, and **Assignment** tabs into a unified interface. This app provides users with easy access to all relevant records and ensures streamlined navigation across different objects. It enhances the user experience by organizing tasks and data in a single, efficient workspace.



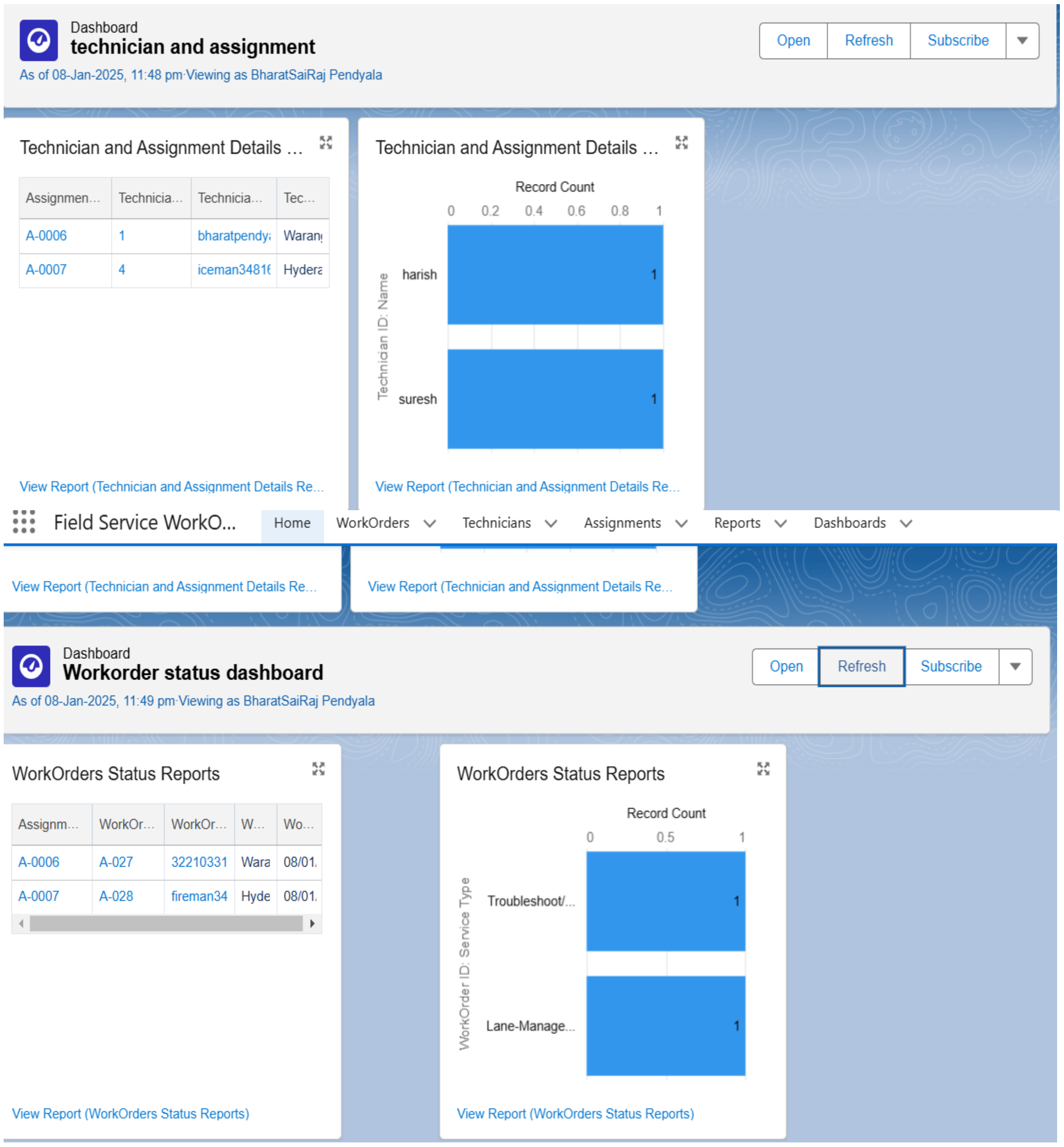
SETUP

Lightning Experience App Manager

New Lightning App | New Connected App

24 items • Sorted by App Name • Filtered by All appmenuitems - TabSet Type, App Type

	App Name	Developer Name	Description	Last Modified ...	Ap...	Vi...
7	Community	Community	Salesforce CRM Communities	03/01/2025, 10:25 pm	Classic	✓
8	Content	Content	Salesforce CRM Content	03/01/2025, 10:25 pm	Classic	✓
9	Data Manager	DataManager	Use Data Manager to view limits, monitor usage, and manage r...	03/01/2025, 10:25 pm	Lightning	✓
10	Digital Experiences	SalesforceCMS	Manage content and media for all of your sites.	03/01/2025, 10:25 pm	Lightning	✓
11	Field Service WorkOrder Optimizati...	Field_Service_WorkOrder_Optimization	Field Service WorkOrder Optimization leverages advanced algo...	08/01/2025, 3:34 pm	Lightning	✓
12	Lightning Usage App	LightningInstrumentation	View Adoption and Usage Metrics for Lightning Experience	03/01/2025, 10:25 pm	Lightning	✓


Dashboard
technician and assignment
As of 08-Jan-2025, 11:48 pm·Viewing as BharatSaiRaj Pendyala

Technician and Assignment Details ...

Assignmen...	Technicia...	Technicia...	Tec...
A-0006	1	bharatpendy	Waran
A-0007	4	iceman3481	Hydera

View Report (Technician and Assignment Details Re...

Dashboard
Workorder status dashboard
As of 08-Jan-2025, 11:49 pm·Viewing as BharatSaiRaj Pendyala

WorkOrders Status Reports

Assignm...	WorkOr...	WorkOr...	W...	Wo...
A-0006	A-027	32210331	Wara	08/01.
A-0007	A-028	fireman34	Hyde	08/01.

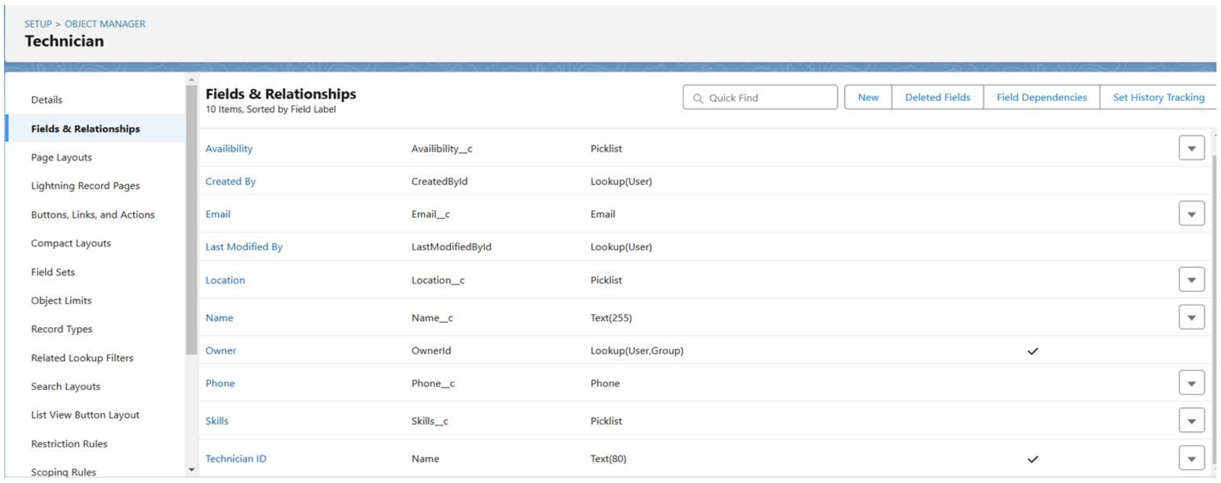
View Report (WorkOrders Status Reports)

WorkOrders Status Reports
Record Count
0
0.5
1
WorkOrder ID: Service Type
Troubleshoot/...
1
Lane-Manage...
1
View Report (WorkOrders Status Reports)

Create Fields and Relationships

1. **Technician**: Fields like Technician ID, Skills, Availability, and Location. Lookup to Work Order and Assignment.
2. **Work Order**: Fields like Work Order ID, Status, Priority, and Service Type. Lookup to Technician and Assignment.
3. **Assignment**: Fields like Assignment ID, Assignment Date, and Completion Date. Lookup to Technician and Work Order.

1. Technician:


SETUP > OBJECT MANAGER
Technician

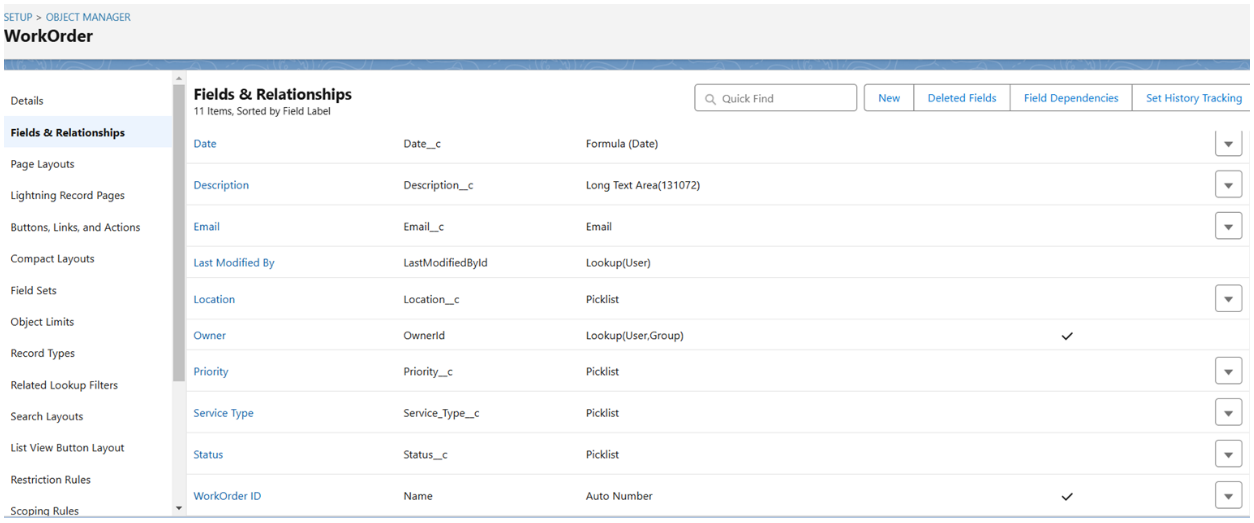
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules

Fields & Relationships
10 Items, Sorted by Field Label

Quick Find | New | Deleted Fields | Field Dependencies | Set History Tracking

Availibility	Availibility__c	Picklist	
Created By	CreatedById	Lookup(User)	
Email	Email__c	Email	
Last Modified By	LastModifiedById	Lookup(User)	
Location	Location__c	Picklist	
Name	Name__c	Text(255)	
Owner	OwnerId	Lookup(User,Group)	✓
Phone	Phone__c	Phone	
Skills	Skills__c	Picklist	
Technician ID	Name	Text(80)	✓

1. WorkOrder:


SETUP > OBJECT MANAGER
WorkOrder

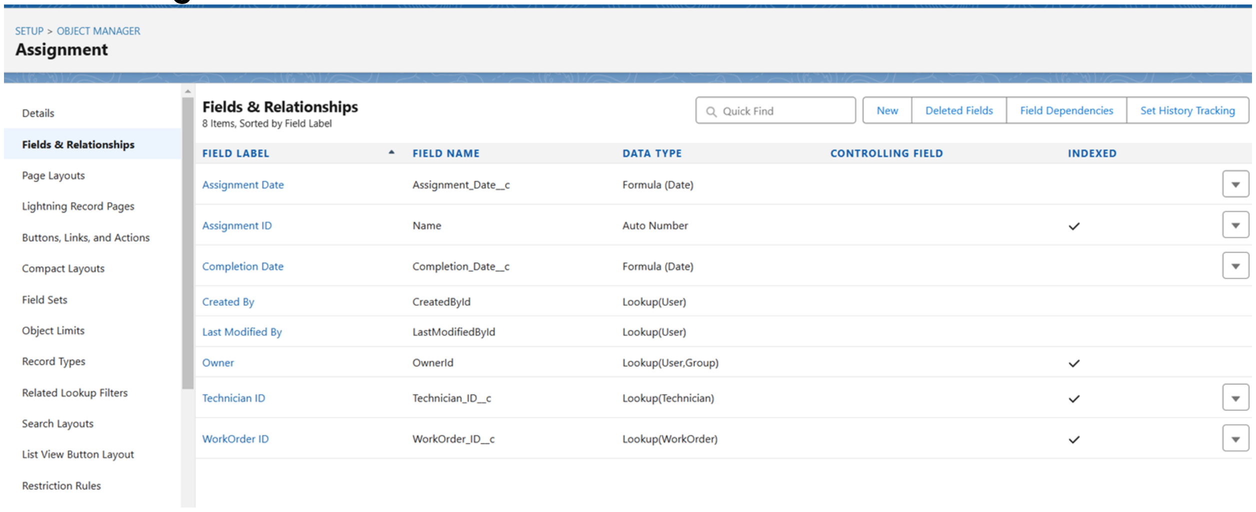
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules

Fields & Relationships
11 Items, Sorted by Field Label

Quick Find | New | Deleted Fields | Field Dependencies | Set History Tracking

Date	Date__c	Formula (Date)	
Description	Description__c	Long Text Area(131072)	
Email	Email__c	Email	
Last Modified By	LastModifiedById	Lookup(User)	
Location	Location__c	Picklist	
Owner	OwnerId	Lookup(User,Group)	✓
Priority	Priority__c	Picklist	
Service Type	Service_Type__c	Picklist	
Status	Status__c	Picklist	
WorkOrder ID	Name	Auto Number	✓

1. Assignment


SETUP > OBJECT MANAGER
Assignment

Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules

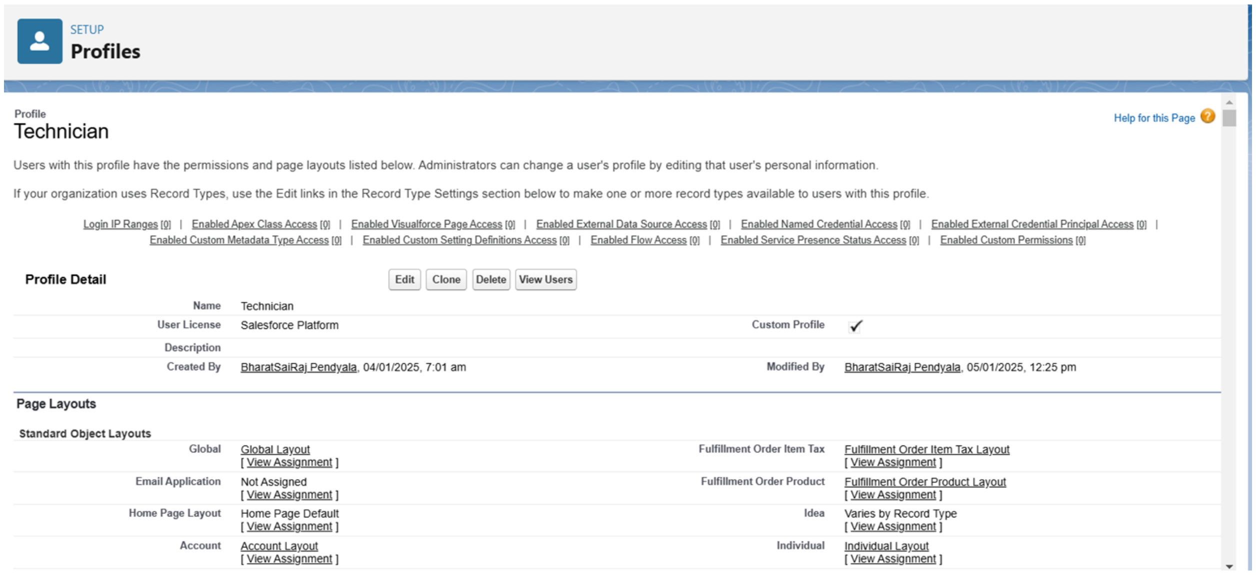
Fields & Relationships
8 Items, Sorted by Field Label

Quick Find | New | Deleted Fields | Field Dependencies | Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Assignment Date	Assignment_Date__c	Formula (Date)		
Assignment ID	Name	Auto Number		✓
Completion Date	Completion_Date__c	Formula (Date)		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Technician ID	Technician_ID__c	Lookup(Technician)		✓
WorkOrder ID	WorkOrder_ID__c	Lookup(WorkOrder)		✓

Create Technician Profile

The **Technician Profile** defines permissions and access for technicians in the Salesforce Org. It grants visibility to relevant objects like **Work Order** and **Assignment**, allowing technicians to view, update, and complete their tasks. Access is restricted to only records assigned to them, ensuring data security and streamlined operations.



SETUP
Profiles

Profile
Technician

Help for this Page

Users with this profile have the permissions and page layouts listed below. Administrators can change a user's profile by editing that user's personal information.

If your organization uses Record Types, use the Edit links in the Record Type Settings section below to make one or more record types available to users with this profile.

Login IP Ranges [0] | Enabled Apex Class Access [0] | Enabled Visualforce Page Access [0] | Enabled External Data Source Access [0] | Enabled Named Credential Access [0] | Enabled External Credential Principal Access [0] | Enabled Custom Metadata Type Access [0] | Enabled Custom Setting Definitions Access [0] | Enabled Flow Access [0] | Enabled Service Presence Status Access [0] | Enabled Custom Permissions [0]

Profile Detail — Edit | Clone | Delete | View Users

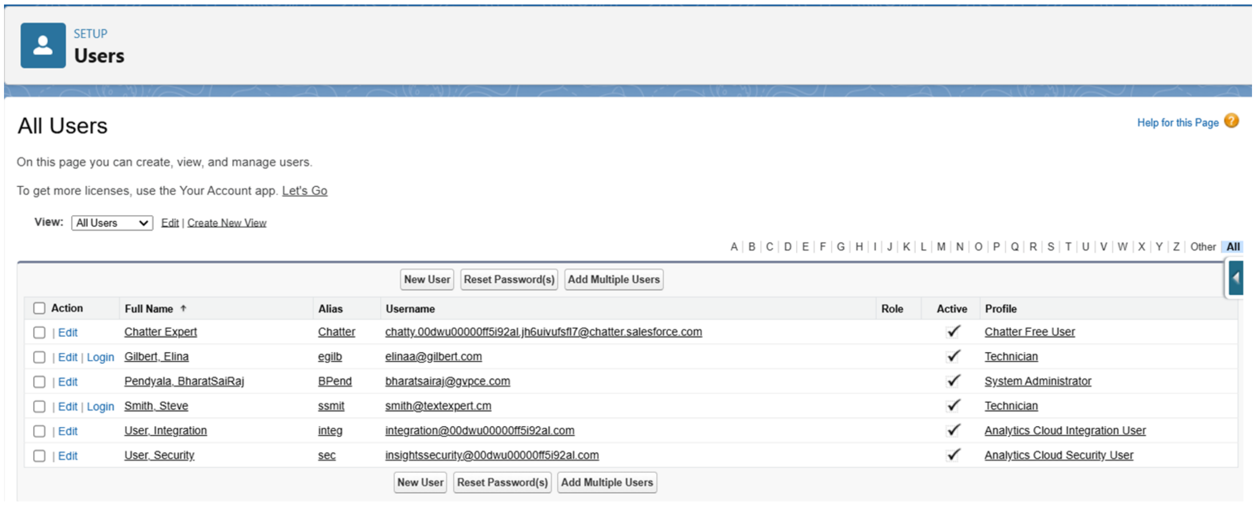
Name	Technician		
User License	Salesforce Platform	Custom Profile	✓
Description			
Created By	BharatSaiRaj Pendyala, 04/01/2025, 7:01 am	Modified By	BharatSaiRaj Pendyala, 05/01/2025, 12:25 pm

Page Layouts

Standard Object Layouts

Global	Global Layout [View Assignment]	Fulfillment Order Item Tax	Fulfillment Order Item Tax Layout [View Assignment]
Email Application	Not Assigned [View Assignment]	Fulfillment Order Product	Fulfillment Order Product Layout [View Assignment]
Home Page Layout	Home Page Default [View Assignment]	Idea	Varies by Record Type [View Assignment]
Account	Account Layout [View Assignment]	Individual	Individual Layout [View Assignment]

Create Users



SETUP
Users

All Users

Help for this Page

On this page you can create, view, and manage users.

To get more licenses, use the Your Account app. Let's Go

View: All Users | Edit | Create New View

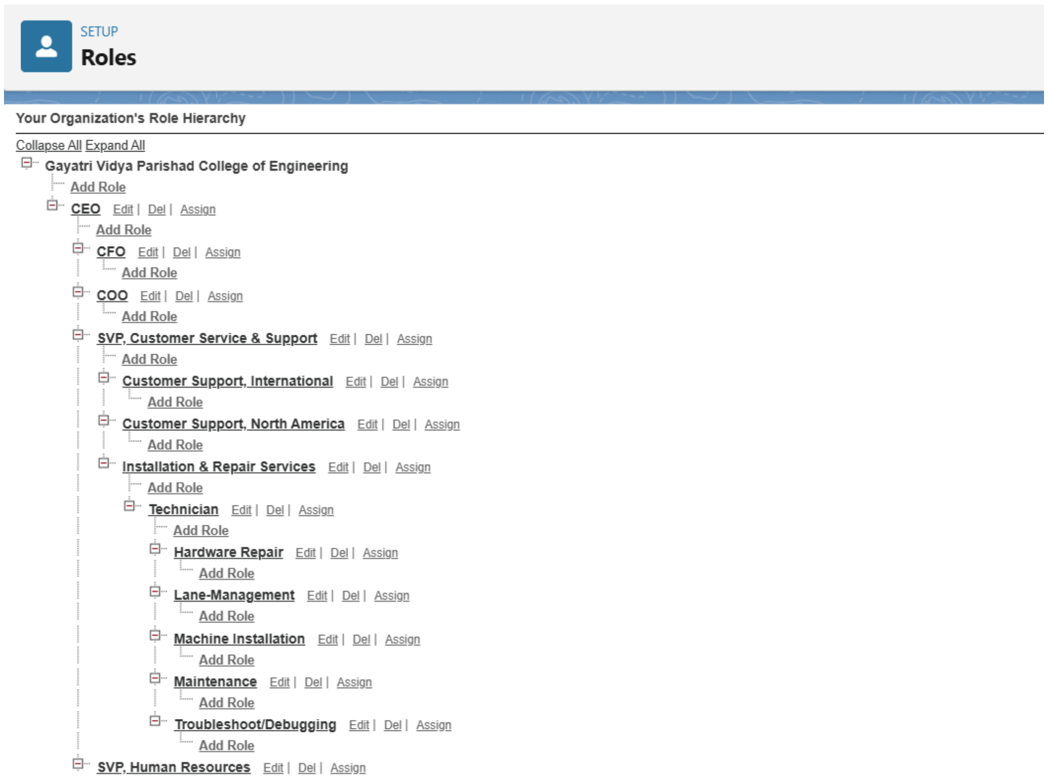
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

New User | Reset Password(s) | Add Multiple Users

Action	Full Name ↑	Alias	Username	Role	Active	Profile
| Edit	Chatter Expert	Chatter	chatty.00dwu00000ff5i92al.jh6uivufsfl7@chatter.salesforce.com		✓	Chatter Free User
| Edit | Login	Gilbert, Elina	egilb	elinaa@gilbert.com		✓	Technician
| Edit	Pendyala, BharatSaiRaj	BPend	bharatsairaj@gvpce.com		✓	System Administrator
| Edit | Login	Smith, Steve	ssmit	smith@textexpert.cm		✓	Technician
| Edit	User, Integration	integ	integration@00dwu00000ff5i92al.com		✓	Analytics Cloud Integration User
| Edit	User, Security	sec	insightssecurity@00dwu00000ff5i92al.com		✓	Analytics Cloud Security User

New User | Reset Password(s) | Add Multiple Users

Create Technician Role under Services and Maintainance and create roles under Technician and Assign it to Users:


SETUP
Roles

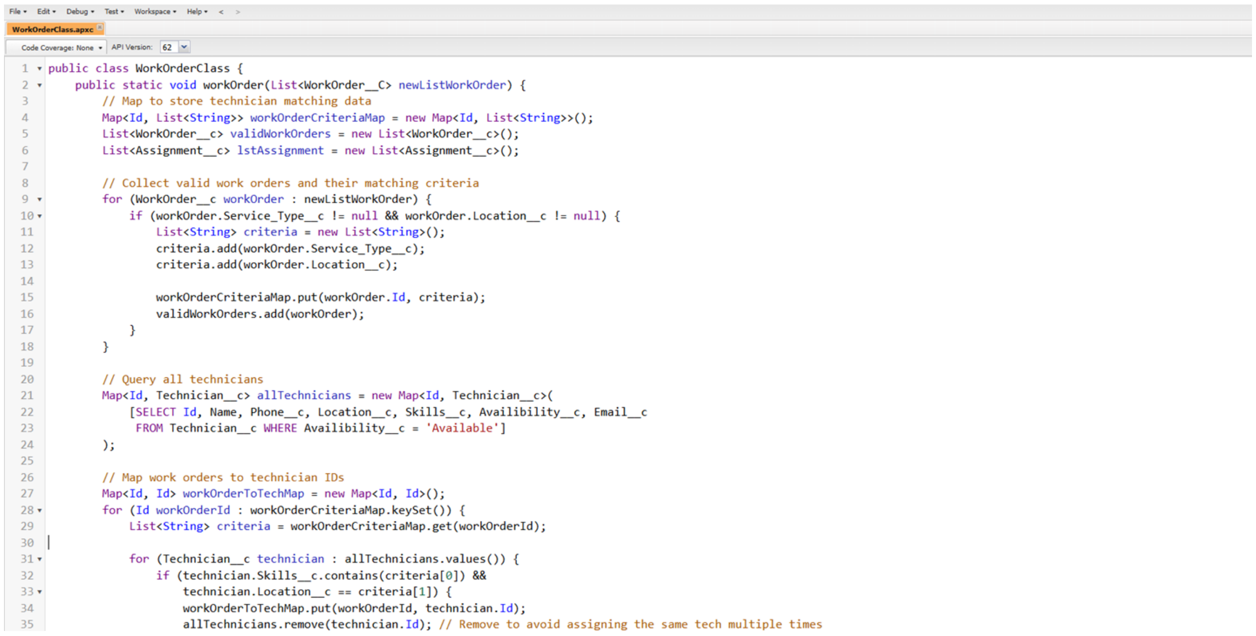
Your Organization's Role Hierarchy

Collapse All Expand All

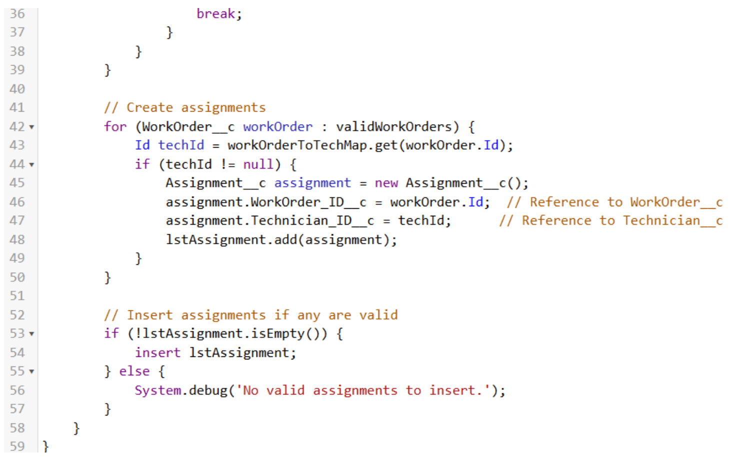
- Gayatri Vidya Parishad College of Engineering
  - Add Role
  - CEO Edit | Del | Assign
    - Add Role
    - CFO Edit | Del | Assign
      - Add Role
    - COO Edit | Del | Assign
      - Add Role
    - SVP, Customer Service & Support Edit | Del | Assign
      - Add Role
      - Customer Support, International Edit | Del | Assign
        - Add Role
      - Customer Support, North America Edit | Del | Assign
        - Add Role
      - Installation & Repair Services Edit | Del | Assign
        - Add Role
        - Technician Edit | Del | Assign
          - Add Role
          - Hardware Repair Edit | Del | Assign
            - Add Role
          - Lane-Management Edit | Del | Assign
            - Add Role
          - Machine Installation Edit | Del | Assign
            - Add Role
          - Maintenance Edit | Del | Assign
            - Add Role
          - Troubleshoot/Debugging Edit | Del | Assign
            - Add Role
    - SVP, Human Resources Edit | Del | Assign

Create Apex Classes & Triggers

1. WorkOrder Class:

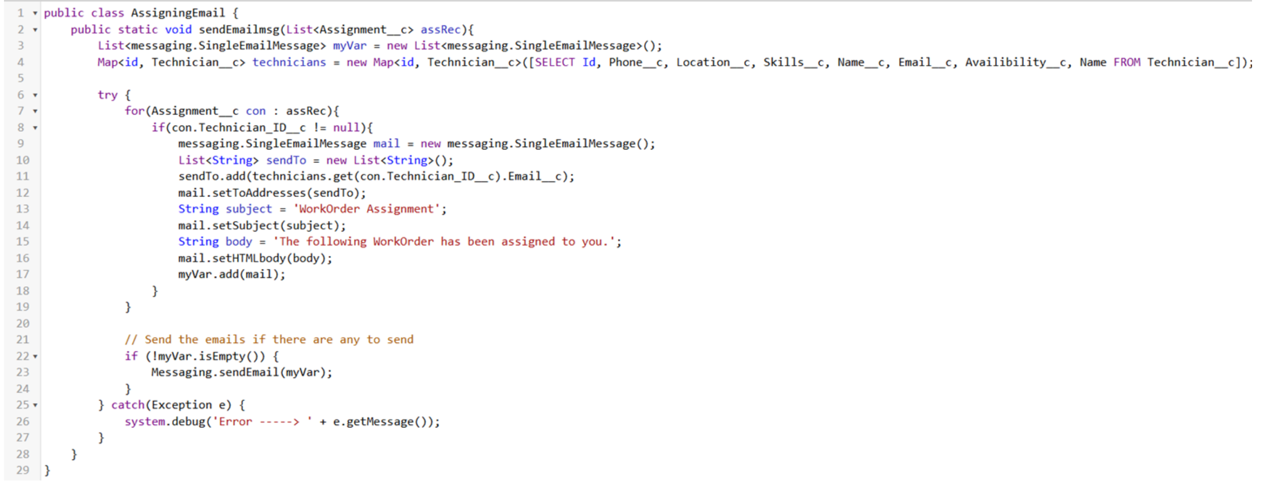

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
WorkOrderClass.apxc
Code Coverage: None ▾ API Version: 62

```apex
public class WorkOrderClass {
    public static void workOrder(List<WorkOrder__C> newListWorkOrder) {
        // Map to store technician matching data
        Map<Id, List<String>> workOrderCriteriaMap = new Map<Id, List<String>>();
        List<WorkOrder__c> validWorkOrders = new List<WorkOrder__c>();
        List<Assignment__c> lstAssignment = new List<Assignment__c>();

        // Collect valid work orders and their matching criteria
        for (WorkOrder__c workOrder : newListWorkOrder) {
            if (workOrder.Service_Type__c != null && workOrder.Location__c != null) {
                List<String> criteria = new List<String>();
                criteria.add(workOrder.Service_Type__c);
                criteria.add(workOrder.Location__c);

                workOrderCriteriaMap.put(workOrder.Id, criteria);
                validWorkOrders.add(workOrder);
            }
        }

        // Query all technicians
        Map<Id, Technician__c> allTechnicians = new Map<Id, Technician__c>(
            [SELECT Id, Name, Phone__c, Location__c, Skills__c, Availibility__c, Email__c
             FROM Technician__c WHERE Availibility__c = 'Available']
        );

        // Map work orders to technician IDs
        Map<Id, Id> workOrderToTechMap = new Map<Id, Id>();
        for (Id workOrderId : workOrderCriteriaMap.keySet()) {
            List<String> criteria = workOrderCriteriaMap.get(workOrderId);

            for (Technician__c technician : allTechnicians.values()) {
                if (technician.Skills__c.contains(criteria[0]) &&
                    technician.Location__c == criteria[1]) {
                    workOrderToTechMap.put(workOrderId, technician.Id);
                    allTechnicians.remove(technician.Id); // Remove to avoid assigning the same tech multiple times
```



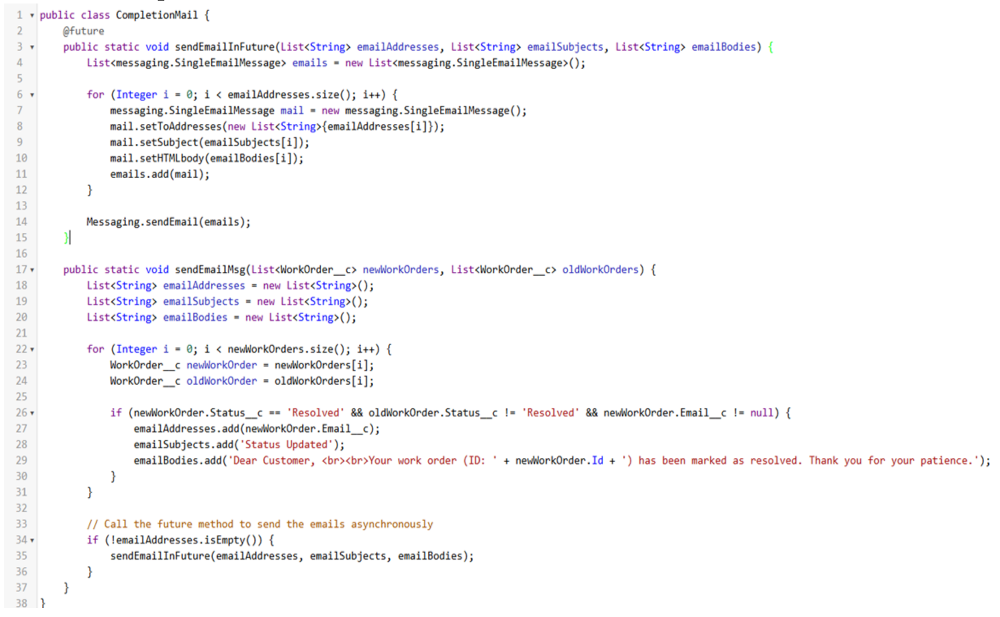
```
                    break;
                }
            }
        }

        // Create assignments
        for (WorkOrder__c workOrder : validWorkOrders) {
            Id techId = workOrderToTechMap.get(workOrder.Id);
            if (techId != null) {
                Assignment__c assignment = new Assignment__c();
                assignment.WorkOrder_ID__c = workOrder.Id;  // Reference to WorkOrder__c
                assignment.Technician_ID__c = techId;      // Reference to Technician__c
                lstAssignment.add(assignment);
            }
        }

        // Insert assignments if any are valid
        if (!lstAssignment.isEmpty()) {
            insert lstAssignment;
        } else {
            System.debug('No valid assignments to insert.');
        }
    }
}
```

1. AssigningEMail Class :



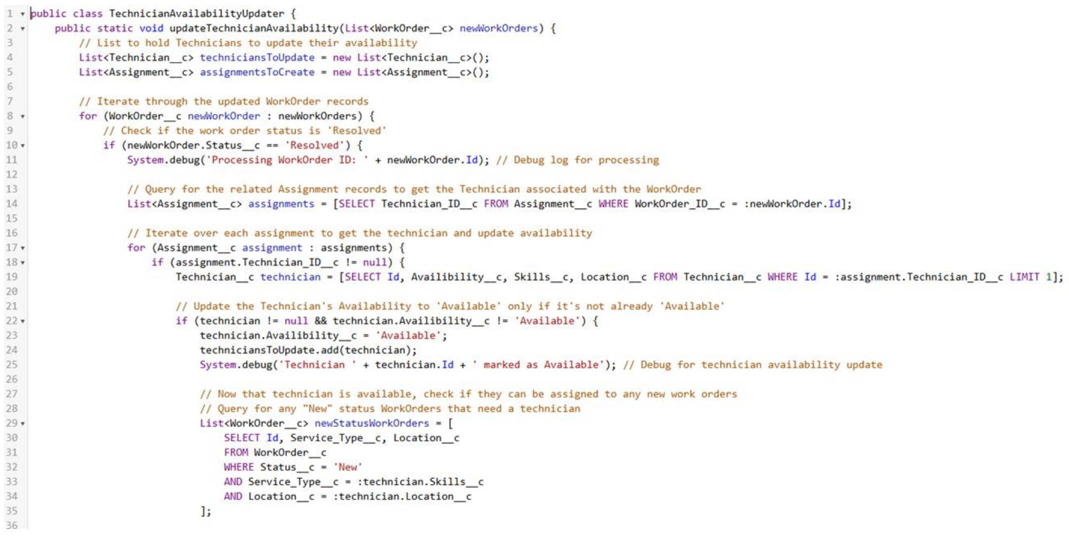
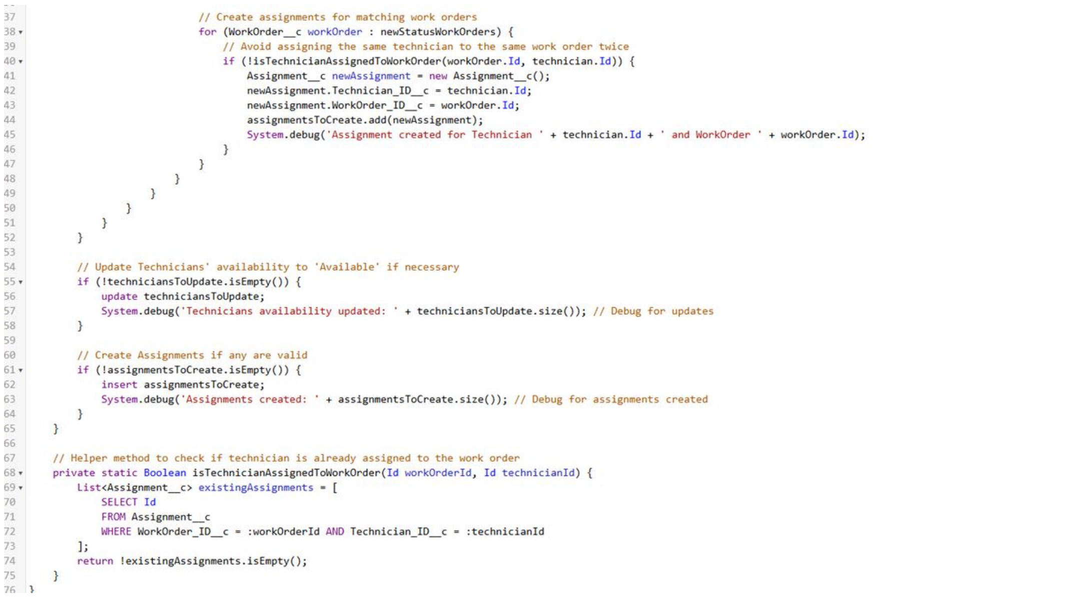
```
public class AssigningEmail {
    public static void sendEmailmsg(List<Assignment__c> assRec){
        List<messaging.SingleEmailMessage> myVar = new List<messaging.SingleEmailMessage>();
        Map<id, Technician__c> technicians = new Map<id, Technician__c>([SELECT Id, Phone__c, Location__c, Skills__c, 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(technicians.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);
                }
            }

            // Send the emails if there are any to send
            if (!myVar.isEmpty()) {
                Messaging.sendEmail(myVar);
            }
        } catch(Exception e) {
            system.debug('Error -----> ' + e.getMessage());
        }
    }
}
```

1. CompletionMail Class:

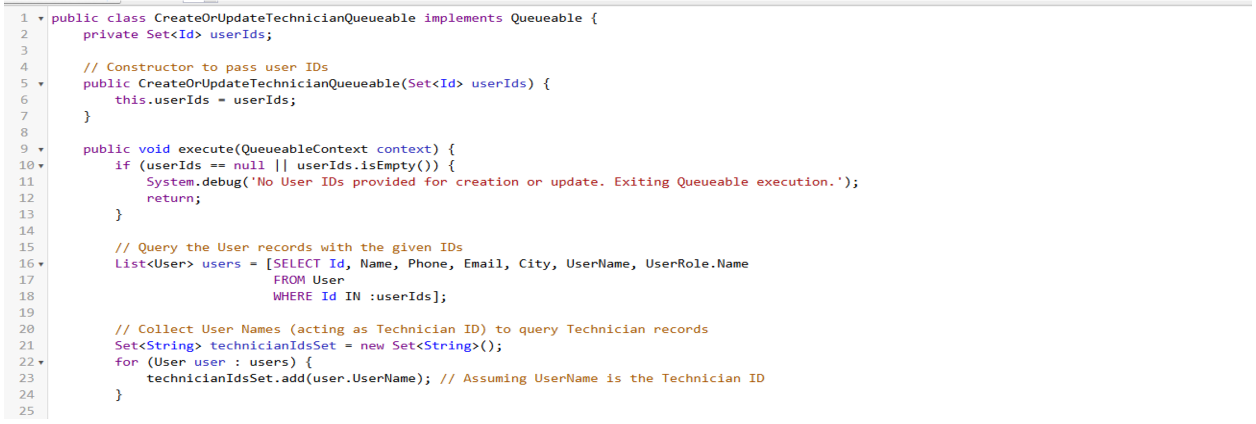


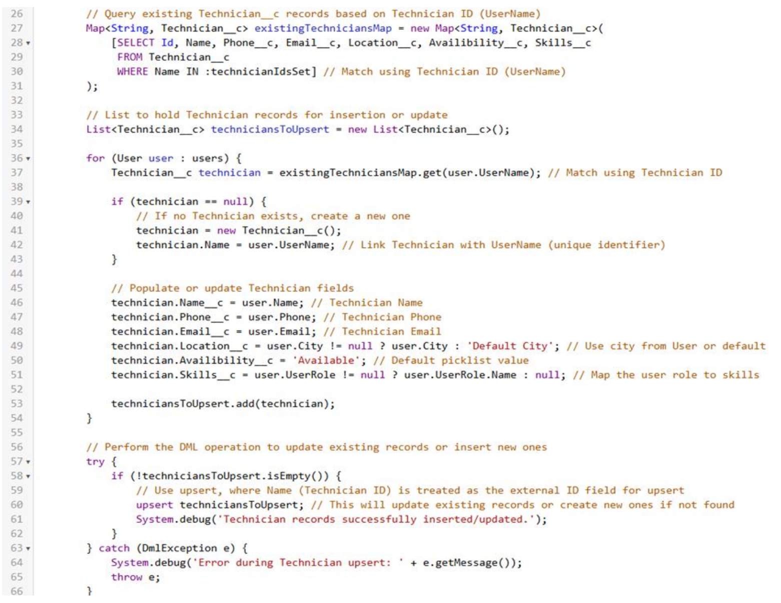
```
public class CompletionMail {
    @future
    public static void sendEmailInFuture(List<String> emailAddresses, List<String> emailSubjects, List<String> emailBodies) {
        List<messaging.SingleEmailMessage> emails = new List<messaging.SingleEmailMessage>();

        for (Integer i = 0; i < emailAddresses.size(); i++) {
            messaging.SingleEmailMessage mail = new messaging.SingleEmailMessage();
            mail.setToAddresses(new List<String>{emailAddresses[i]});
            mail.setSubject(emailSubjects[i]);
            mail.setHTMLbody(emailBodies[i]);
            emails.add(mail);
        }

        Messaging.sendEmail(emails);
    }

    public static void sendEmailMsg(List<WorkOrder__c> newWorkOrders, List<WorkOrder__c> oldWorkOrders) {
        List<String> emailAddresses = new List<String>();
        List<String> emailSubjects = new List<String>();
        List<String> emailBodies = new List<String>();

        for (Integer i = 0; i < newWorkOrders.size(); i++) {
            WorkOrder__c newWorkOrder = newWorkOrders[i];
            WorkOrder__c oldWorkOrder = oldWorkOrders[i];

            if (newWorkOrder.Status__c == 'Resolved' && oldWorkOrder.Status__c != 'Resolved' && newWorkOrder.Email__c != null) {
                emailAddresses.add(newWorkOrder.Email__c);
                emailSubjects.add('Status Updated');
                emailBodies.add('Dear Customer, <br><br>Your work order (ID: ' + newWorkOrder.Id + ') has been marked as resolved. Thank you for your patience.');
            }
        }

        // Call the future method to send the emails asynchronously
        if (!emailAddresses.isEmpty()) {
            sendEmailInFuture(emailAddresses, emailSubjects, emailBodies);
        }
    }
}
```

TechnicianAvailabilityUpdater Class:


```
public class TechnicianAvailabilityUpdater {
    public static void updateTechnicianAvailability(List<WorkOrder__c> newWorkOrders) {
        // List to hold Technicians to update their availability
        List<Technician__c> techniciansToUpdate = new List<Technician__c>();
        List<Assignment__c> assignmentsToCreate = new List<Assignment__c>();

        // Iterate through the updated WorkOrder records
        for (WorkOrder__c newWorkOrder : newWorkOrders) {
            // Check if the work order status is 'Resolved'
            if (newWorkOrder.Status__c == 'Resolved') {
                System.debug('Processing WorkOrder ID: ' + newWorkOrder.Id); // Debug log for processing

                // Query for the related Assignment records to get the Technician associated with the WorkOrder
                List<Assignment__c> assignments = [SELECT Technician_ID__c FROM Assignment__c WHERE WorkOrder_ID__c = :newWorkOrder.Id];

                // Iterate over each assignment to get the technician and update availability
                for (Assignment__c assignment : assignments) {
                    if (assignment.Technician_ID__c != null) {
                        Technician__c technician = [SELECT Id, Availibility__c, Skills__c, Location__c FROM Technician__c WHERE Id = :assignment.Technician_ID__c LIMIT 1];

                        // Update the Technician's Availability to 'Available' only if it's not already 'Available'
                        if (technician != null && technician.Availibility__c != 'Available') {
                            technician.Availibility__c = 'Available';
                            techniciansToUpdate.add(technician);
                            System.debug('Technician ' + technician.Id + ' marked as Available'); // Debug for technician availability update

                            // Now that technician is available, check if they can be assigned to any new work orders
                            // Query for any "New" status WorkOrders that need a technician
                            List<WorkOrder__c> newStatusWorkOrders = [
                                SELECT Id, Service_Type__c, Location__c
                                FROM WorkOrder__c
                                WHERE Status__c = 'New'
                                AND Service_Type__c = :technician.Skills__c
                                AND Location__c = :technician.Location__c
                            ];

```

```
                            // Create assignments for matching work orders
                            for (WorkOrder__c workOrder : newStatusWorkOrders) {
                                // Avoid assigning the same technician to the same work order twice
                                if (!isTechnicianAssignedToWorkOrder(workOrder.Id, technician.Id)) {
                                    Assignment__c newAssignment = new Assignment__c();
                                    newAssignment.Technician_ID__c = technician.Id;
                                    newAssignment.WorkOrder_ID__c = workOrder.Id;
                                    assignmentsToCreate.add(newAssignment);
                                    System.debug('Assignment created for Technician ' + technician.Id + ' and WorkOrder ' + workOrder.Id);
                                }
                            }
                        }
                    }
                }
            }
        }

        // Update Technicians' availability to 'Available' if necessary
        if (!techniciansToUpdate.isEmpty()) {
            update techniciansToUpdate;
            System.debug('Technicians availability updated: ' + techniciansToUpdate.size()); // Debug for updates
        }

        // Create Assignments if any are valid
        if (!assignmentsToCreate.isEmpty()) {
            insert assignmentsToCreate;
            System.debug('Assignments created: ' + assignmentsToCreate.size()); // Debug for assignments created
        }
    }

    // Helper method to check if technician is already assigned to the work order
    private static Boolean isTechnicianAssignedToWorkOrder(Id workOrderId, Id technicianId) {
        List<Assignment__c> existingAssignments = [
            SELECT Id
            FROM Assignment__c
            WHERE WorkOrder_ID__c = :workOrderId AND Technician_ID__c = :technicianId
        ];
        return !existingAssignments.isEmpty();
    }
}
```

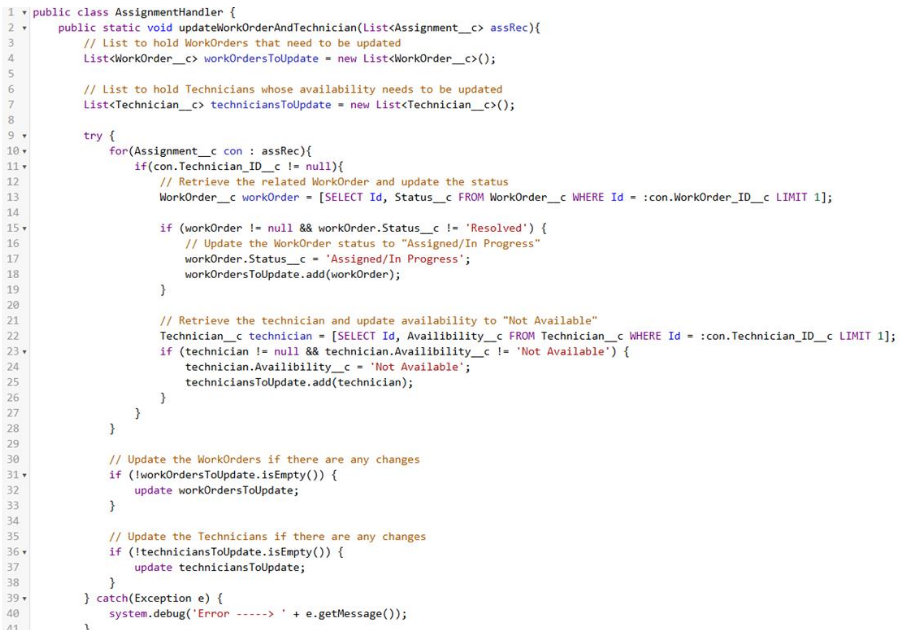
1. CreateOrUpdateTechnicianQueueable Class:


```
public class CreateOrUpdateTechnicianQueueable implements Queueable {
    private Set<Id> userIds;

    // Constructor to pass user IDs
    public CreateOrUpdateTechnicianQueueable(Set<Id> userIds) {
        this.userIds = userIds;
    }

    public void execute(QueueableContext context) {
        if (userIds == null || userIds.isEmpty()) {
            System.debug('No User IDs provided for creation or update. Exiting Queueable execution.');
            return;
        }

        // Query the User records with the given IDs
        List<User> users = [SELECT Id, Name, Phone, Email, City, UserName, UserRole.Name
                            FROM User
                            WHERE Id IN :userIds];

        // Collect User Names (acting as Technician ID) to query Technician records
        Set<String> technicianIdsSet = new Set<String>();
        for (User user : users) {
            technicianIdsSet.add(user.UserName); // Assuming UserName is the Technician ID
        }

```



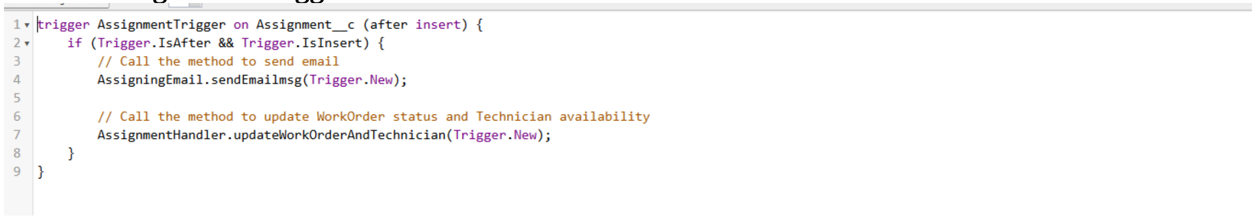
```apex
        // Query existing Technician__c records based on Technician ID (UserName)
        Map<String, Technician__c> existingTechniciansMap = new Map<String, Technician__c>(
            [SELECT Id, Name, Phone__c, Email__c, Location__c, Availibility__c, Skills__c
             FROM Technician__c
             WHERE Name IN :technicianIdsSet] // Match using Technician ID (UserName)
        );

        // List to hold Technician records for insertion or update
        List<Technician__c> techniciansToUpsert = new List<Technician__c>();

        for (User user : users) {
            Technician__c technician = existingTechniciansMap.get(user.UserName); // Match using Technician ID

            if (technician == null) {
                // If no Technician exists, create a new one
                technician = new Technician__c();
                technician.Name = user.UserName; // Link Technician with UserName (unique identifier)
            }

            // Populate or update Technician fields
            technician.Name__c = user.Name; // Technician Name
            technician.Phone__c = user.Phone; // Technician Phone
            technician.Email__c = user.Email; // Technician Email
            technician.Location__c = user.City != null ? user.City : 'Default City'; // Use city from User or default
            technician.Availibility__c = 'Available'; // Default picklist value
            technician.Skills__c = user.UserRole != null ? user.UserRole.Name : null; // Map the user role to skills

            techniciansToUpsert.add(technician);
        }

        // Perform the DML operation to update existing records or insert new ones
        try {
            if (!techniciansToUpsert.isEmpty()) {
                // Use upsert, where Name (Technician ID) is treated as the external ID field for upsert
                upsert techniciansToUpsert; // This will update existing records or create new ones if not found
                System.debug('Technician records successfully inserted/updated.');
            }
        } catch (DmlException e) {
            System.debug('Error during Technician upsert: ' + e.getMessage());
            throw e;
        }
```

1. **AssingmentHandler Class**:



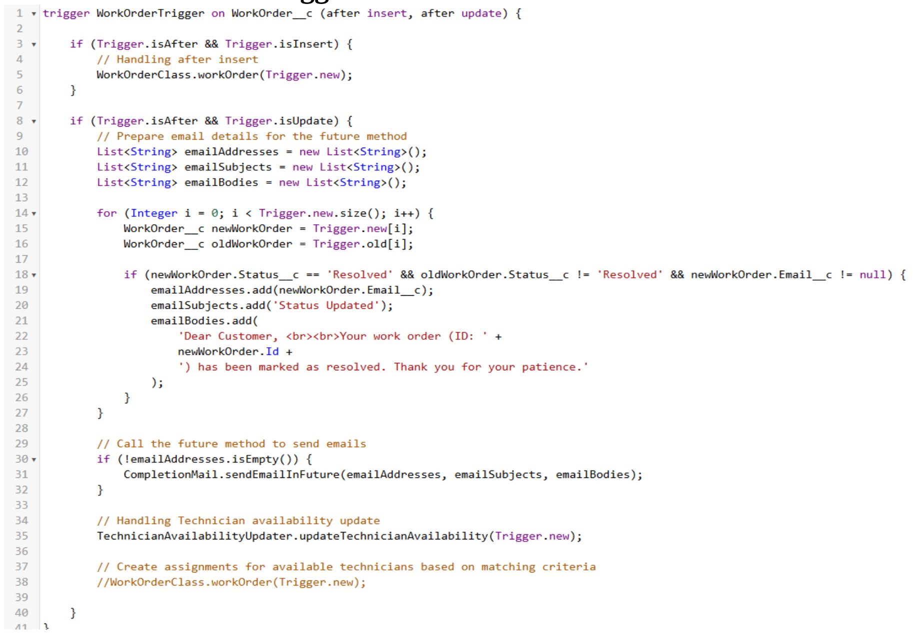
```apex
public class AssignmentHandler {
    public static void updateWorkOrderAndTechnician(List<Assignment__c> assRec){
        // List to hold WorkOrders that need to be updated
        List<WorkOrder__c> workOrdersToUpdate = new List<WorkOrder__c>();

        // List to hold Technicians whose availability needs to be updated
        List<Technician__c> techniciansToUpdate = new List<Technician__c>();

        try {
            for(Assignment__c con : assRec){
                if(con.Technician_ID__c != null){
                    // Retrieve the related WorkOrder and update the status
                    WorkOrder__c workOrder = [SELECT Id, Status__c FROM WorkOrder__c WHERE Id = :con.WorkOrder_ID__c LIMIT 1];

                    if (workOrder != null && workOrder.Status__c != 'Resolved') {
                        // Update the WorkOrder status to "Assigned/In Progress"
                        workOrder.Status__c = 'Assigned/In Progress';
                        workOrdersToUpdate.add(workOrder);
                    }

                    // Retrieve the technician and update availability to "Not Available"
                    Technician__c technician = [SELECT Id, Availibility__c FROM Technician__c WHERE Id = :con.Technician_ID__c LIMIT 1];
                    if (technician != null && technician.Availibility__c != 'Not Available') {
                        technician.Availibility__c = 'Not Available';
                        techniciansToUpdate.add(technician);
                    }
                }
            }

            // Update the WorkOrders if there are any changes
            if (!workOrdersToUpdate.isEmpty()) {
                update workOrdersToUpdate;
            }

            // Update the Technicians if there are any changes
            if (!techniciansToUpdate.isEmpty()) {
                update techniciansToUpdate;
            }
        } catch(Exception e) {
            system.debug('Error -----> ' + e.getMessage());
        }
```

1. **Assignment Trigger**



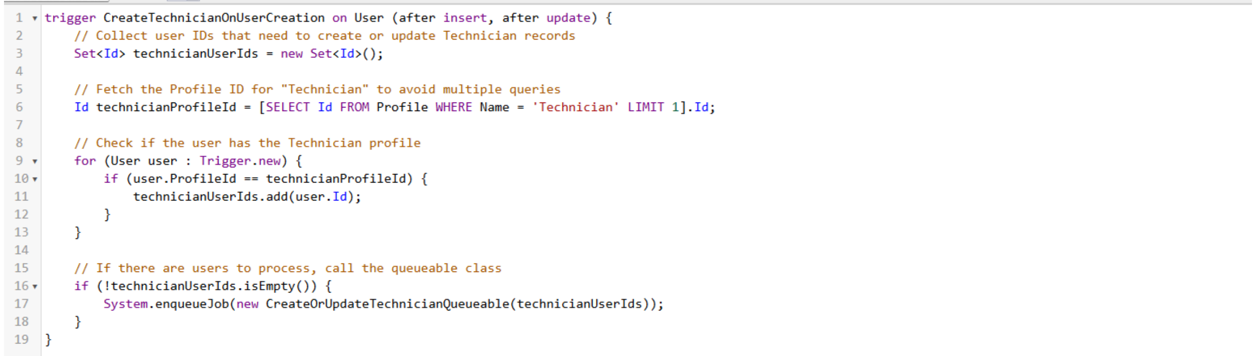
```apex
trigger AssignmentTrigger on Assignment__c (after insert) {
    if (Trigger.IsAfter && Trigger.IsInsert) {
        // Call the method to send email
        AssigningEmail.sendEmailmsg(Trigger.New);

        // Call the method to update WorkOrder status and Technician availability
        AssignmentHandler.updateWorkOrderAndTechnician(Trigger.New);
    }
}
```

1. **WorkOrder Trigger**:



```
trigger WorkOrderTrigger on WorkOrder__c (after insert, after update) {

    if (Trigger.isAfter && Trigger.isInsert) {
        // Handling after insert
        WorkOrderClass.workOrder(Trigger.new);
    }

    if (Trigger.isAfter && Trigger.isUpdate) {
        // Prepare email details for the future method
        List<String> emailAddresses = new List<String>();
        List<String> emailSubjects = new List<String>();
        List<String> emailBodies = new List<String>();

        for (Integer i = 0; i < Trigger.new.size(); i++) {
            WorkOrder__c newWorkOrder = Trigger.new[i];
            WorkOrder__c oldWorkOrder = Trigger.old[i];

            if (newWorkOrder.Status__c == 'Resolved' && oldWorkOrder.Status__c != 'Resolved' && newWorkOrder.Email__c != null) {
                emailAddresses.add(newWorkOrder.Email__c);
                emailSubjects.add('Status Updated');
                emailBodies.add(
                    'Dear Customer, <br><br>Your work order (ID: ' +
                    newWorkOrder.Id +
                    ') has been marked as resolved. Thank you for your patience.'
                );
            }
        }

        // Call the future method to send emails
        if (!emailAddresses.isEmpty()) {
            CompletionMail.sendEmailInFuture(emailAddresses, emailSubjects, emailBodies);
        }

        // Handling Technician availability update
        TechnicianAvailabilityUpdater.updateTechnicianAvailability(Trigger.new);

        // Create assignments for available technicians based on matching criteria
        //WorkOrderClass.workOrder(Trigger.new);

    }
}
```

1. **CreateTechnicianOnUserCreation Trigger**

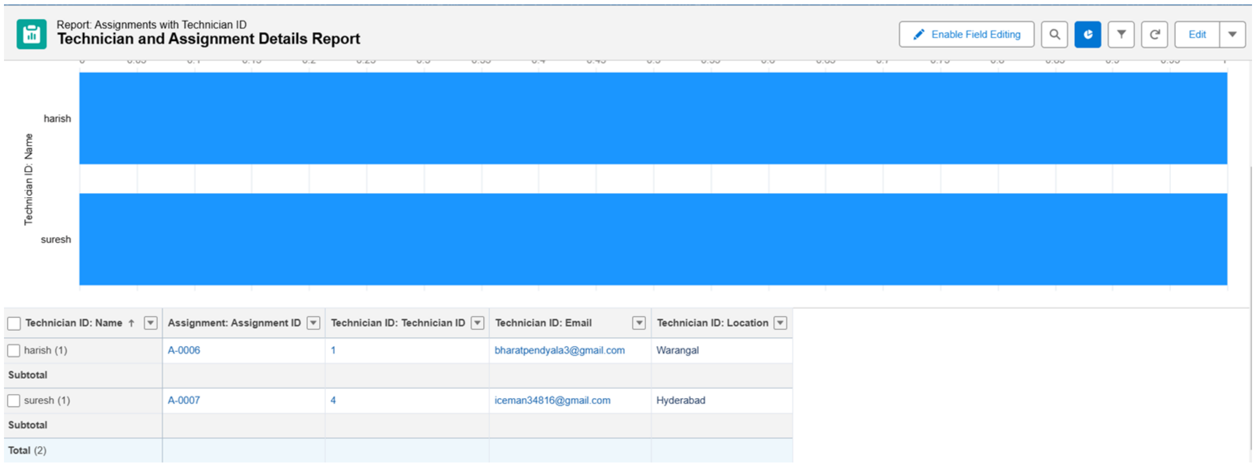


```
trigger CreateTechnicianOnUserCreation on User (after insert, after update) {
    // Collect user IDs that need to create or update Technician records
    Set<Id> technicianUserIds = new Set<Id>();

    // Fetch the Profile ID for "Technician" to avoid multiple queries
    Id technicianProfileId = [SELECT Id FROM Profile WHERE Name = 'Technician' LIMIT 1].Id;

    // Check if the user has the Technician profile
    for (User user : Trigger.new) {
        if (user.ProfileId == technicianProfileId) {
            technicianUserIds.add(user.Id);
        }
    }

    // If there are users to process, call the queueable class
    if (!technicianUserIds.isEmpty()) {
        System.enqueueJob(new CreateOrUpdateTechnicianQueueable(technicianUserIds));
    }
}
```

Create Reports And Dashboards:

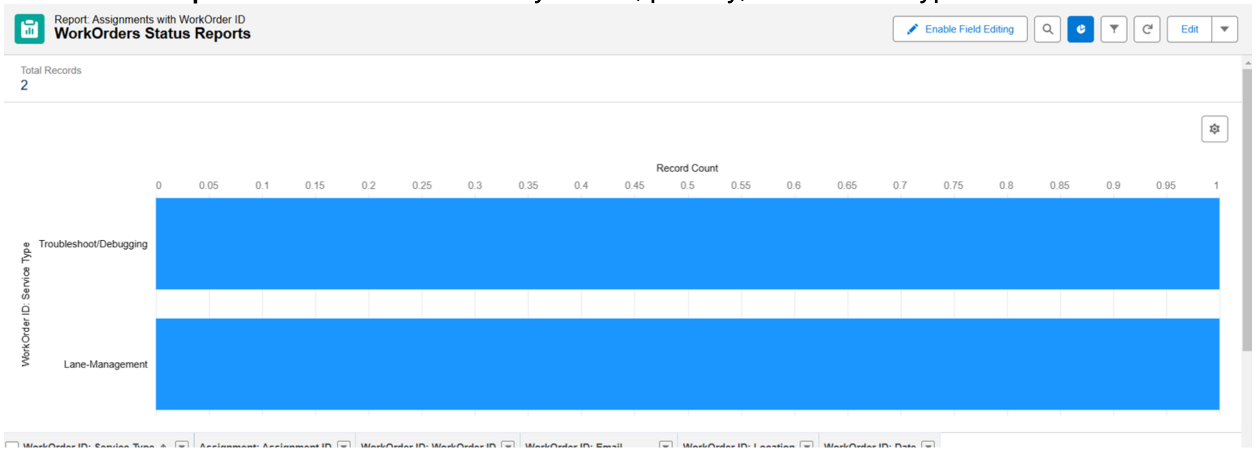
Reports:

1. **Technician and Assignment Details Report**: Monitors technician assignments, including dates and completion status.


Report: Assignments with Technician ID
Technician and Assignment Details Report
Enable Field Editing
Edit
Technician ID: Name
harish
suresh

Technician ID: Name ↑	Assignment: Assignment ID	Technician ID: Technician ID	Technician ID: Email	Technician ID: Location
harish (1)	A-0006	1	bharatpendyala3@gmail.com	Warangal
Subtotal				
suresh (1)	A-0007	4	iceman34816@gmail.com	Hyderabad
Subtotal				
Total (2)				

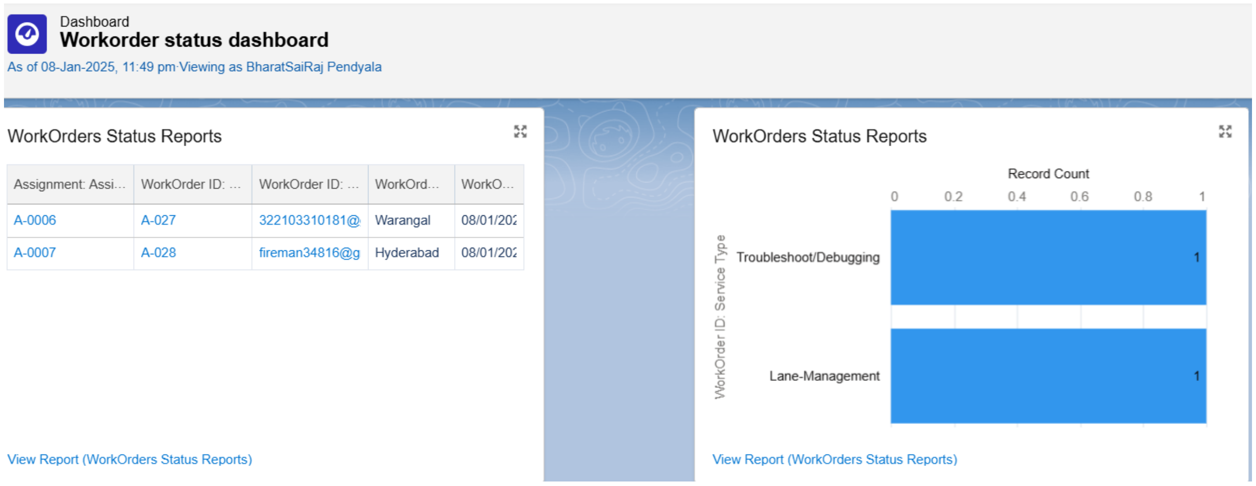
Work Order Report: Tracks work orders by status, priority, and service type.



WorkOrder ID: Service Type ↑	Assignment: Assignment ID	WorkOrder ID: WorkOrder ID	WorkOrder ID: Email	WorkOrder ID: Location	WorkOrder ID: Date

Dashboards:

1. **Work Order Status Dashboard:**

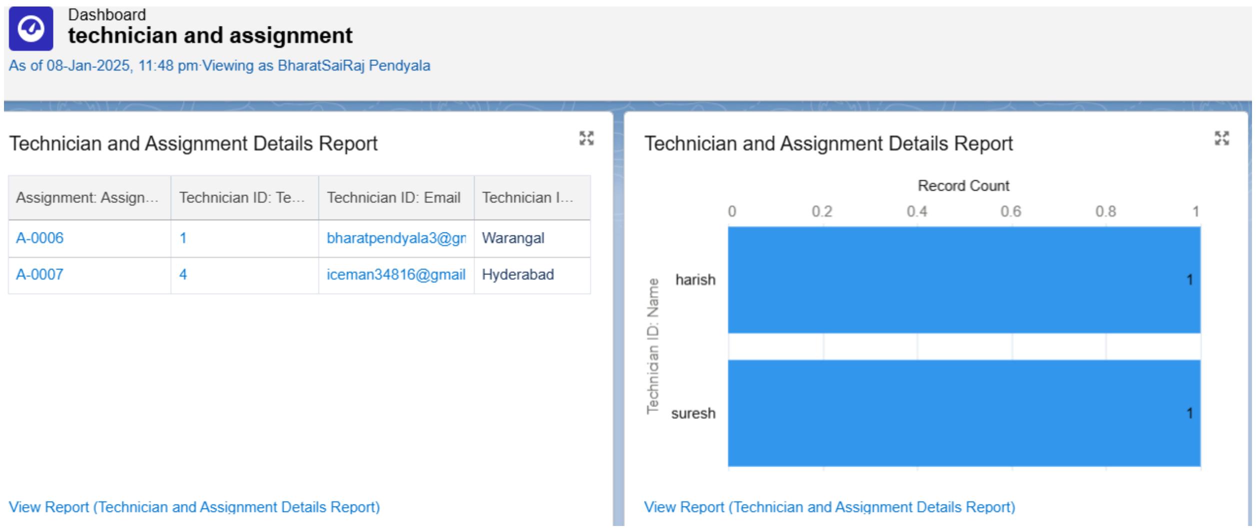

Dashboard
Workorder status dashboard
As of 08-Jan-2025, 11:49 pm·Viewing as BharatSaiRaj Pendyala
WorkOrders Status Reports
Record Count
0
0.2
0.4
0.6
0.8
1
WorkOrder ID: Service Type
Troubleshoot/Debugging
1
Lane-Management
1
View Report (WorkOrders Status Reports)

WorkOrders Status Reports

Assignment: Assi...	WorkOrder ID: ...	WorkOrder ID: ...	WorkOrd...	WorkO...
A-0006	A-027	322103310181@	Warangal	08/01/202
A-0007	A-028	fireman34816@g	Hyderabad	08/01/202

View Report (WorkOrders Status Reports)

1. **Technician and Assignment Details Dashboard:**


Dashboard
technician and assignment
As of 08-Jan-2025, 11:48 pm·Viewing as BharatSaiRaj Pendyala

Technician and Assignment Details Report

Assignment: Assign...	Technician ID: Te...	Technician ID: Email	Technician I...
A-0006	1	bharatpendyala3@gr	Warangal
A-0007	4	iceman34816@gmail	Hyderabad

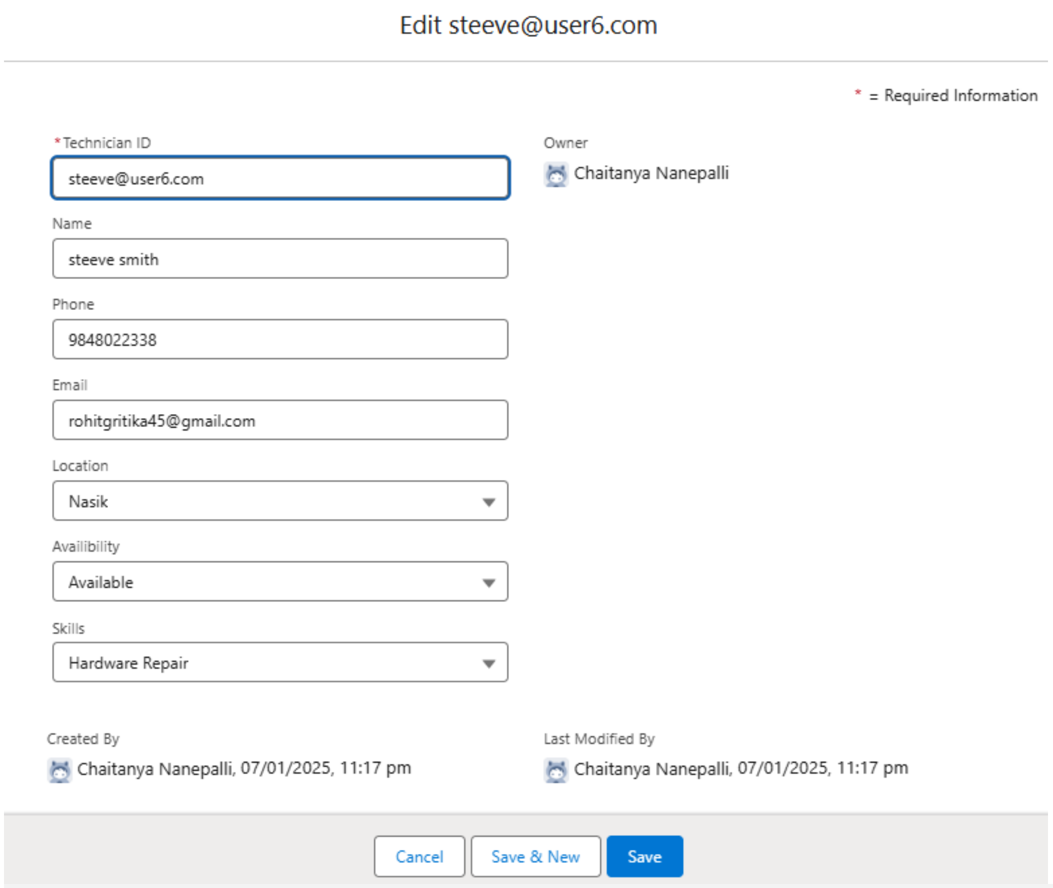
View Report (Technician and Assignment Details Report)

Technician and Assignment Details Report

View Report (Technician and Assignment Details Report)

Testing and Validation:

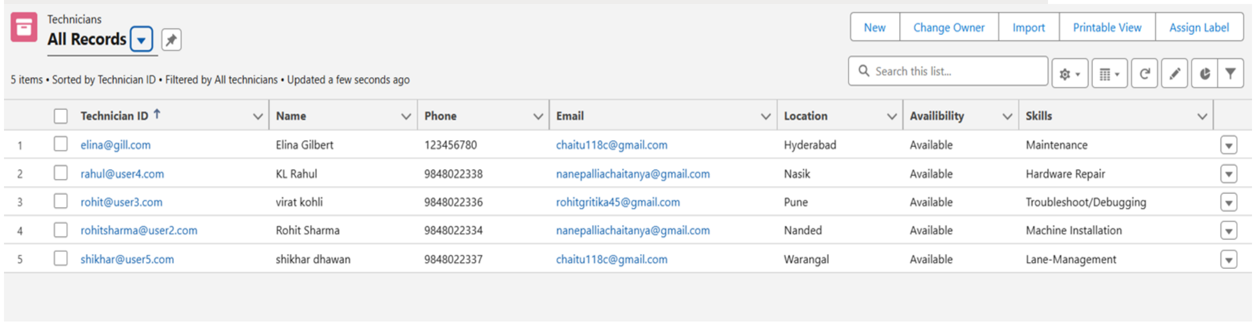
1. Create users under Technician profile such that a record is created in Technician Object with the Users details:(or)
   Create users under Technician Profile directly:



Edit steeve@user6.com

* = Required Information

*Technician ID	steeve@user6.com
Owner	Chaitanya Nanepalli
Name	steeve smith
Phone	9848022338
Email	rohitgritika45@gmail.com
Location	Nasik
Availibility	Available
Skills	Hardware Repair
Created By	Chaitanya Nanepalli, 07/01/2025, 11:17 pm
Last Modified By	Chaitanya Nanepalli, 07/01/2025, 11:17 pm

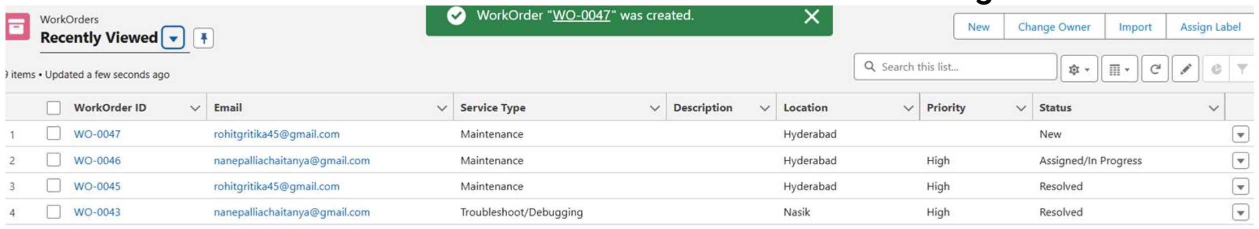


Technicians
All Records

5 items • Sorted by Technician ID • Filtered by All technicians • Updated a few seconds ago

	Technician ID	Name	Phone	Email	Location	Availibility	Skills
1	elina@gill.com	Elina Gilbert	123456780	chaitu118c@gmail.com	Hyderabad	Available	Maintenance
2	rahul@user4.com	KL Rahul	9848022338	nanepalliachaitanya@gmail.com	Nasik	Available	Hardware Repair
3	rohit@user3.com	virat kohli	9848022336	rohitgritika45@gmail.com	Pune	Available	Troubleshoot/Debugging
4	rohitsharma@user2.com	Rohit Sharma	9848022334	nanepalliachaitanya@gmail.com	Nanded	Available	Machine Installation
5	shikhar@user5.com	shikhar dhawan	9848022337	chaitu118c@gmail.com	Warangal	Available	Lane-Management

1. **Create WorkOrder Record which is the work that Customer assigns us:**



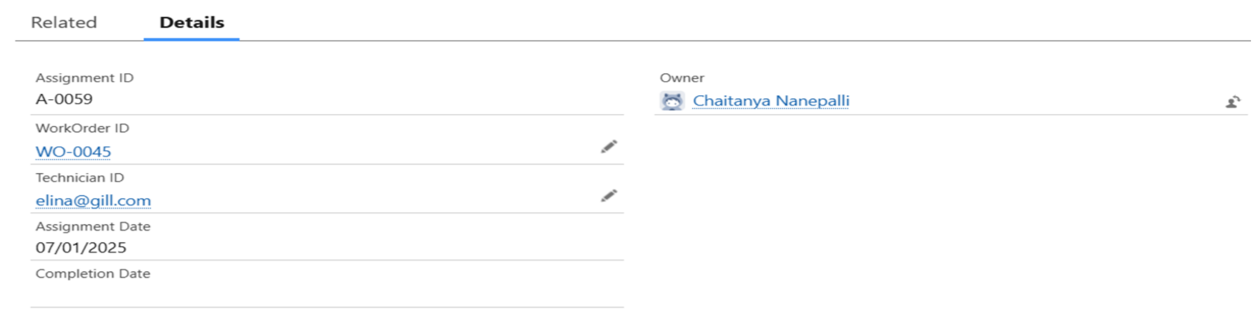
WorkOrders
Recently Viewed

WorkOrder "WO-0047" was created.

items • Updated a few seconds ago

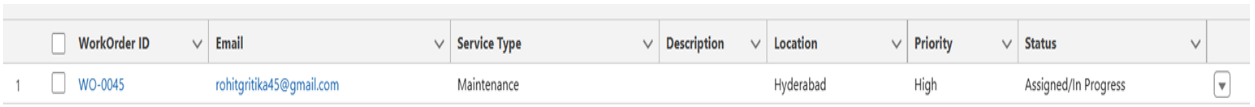
	WorkOrder ID	Email	Service Type	Description	Location	Priority	Status
1	WO-0047	rohitgritika45@gmail.com	Maintenance		Hyderabad		New
2	WO-0046	nanepalliachaitanya@gmail.com	Maintenance		Hyderabad	High	Assigned/In Progress
3	WO-0045	rohitgritika45@gmail.com	Maintenance		Hyderabad	High	Resolved
4	WO-0043	nanepalliachaitanya@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved

Case 1:If the work order is assigned and there is a available technician then it assigns to then it assigns to the technician

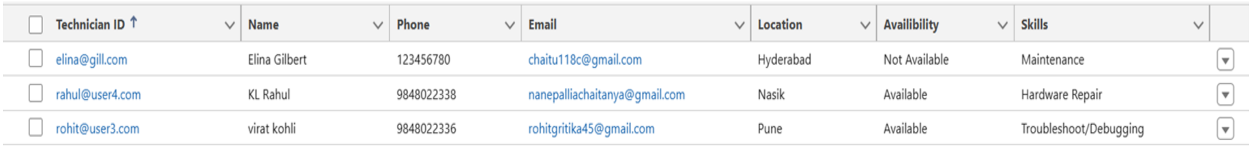


Related | **Details**

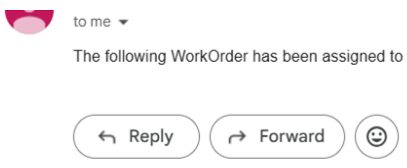
Assignment ID	A-0059
WorkOrder ID	WO-0045
Technician ID	elina@gill.com
Assignment Date	07/01/2025
Completion Date	
Owner	Chaitanya Nanepalli



	WorkOrder ID	Email	Service Type	Description	Location	Priority	Status
1	WO-0045	rohitgritika45@gmail.com	Maintenance		Hyderabad	High	Assigned/In Progress



Technician ID ↑	Name	Phone	Email	Location	Availibility	Skills
elina@gill.com	Elina Gilbert	123456780	chaitu118c@gmail.com	Hyderabad	Not Available	Maintenance
rahul@user4.com	KL Rahul	9848022338	nanepalliachaitanya@gmail.com	Nasik	Available	Hardware Repair
rohit@user3.com	virat kohli	9848022336	rohitgritika45@gmail.com	Pune	Available	Troubleshoot/Debugging

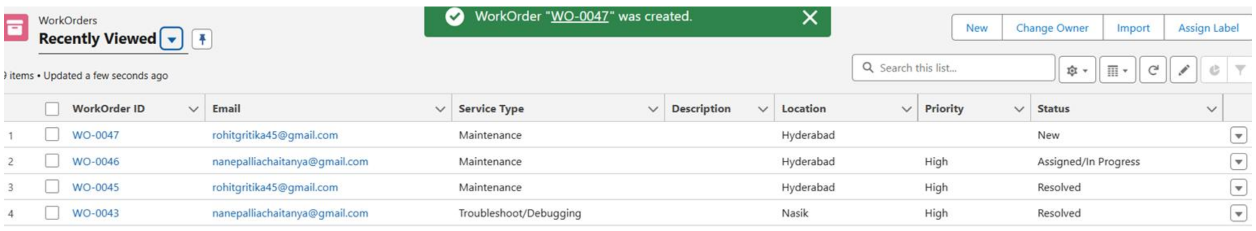


to me

The following WorkOrder has been assigned to you.

When a work order requirement matches with the technician Elina, it is assigned to her, an assignment record is created, and an email notification is sent to inform her about the assigned work. Upon assignment, the work order status is updated to **Assigned/In Progress**, and the technician's availability is updated to **Not Available**.

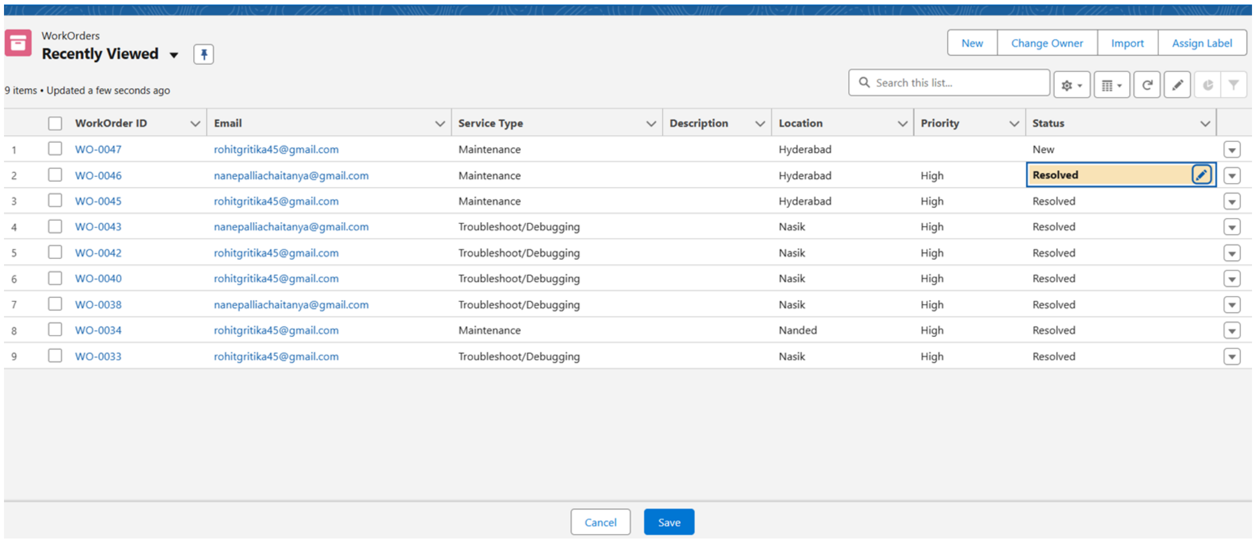
Case 2:If a work order is created and no matching technician is currently available



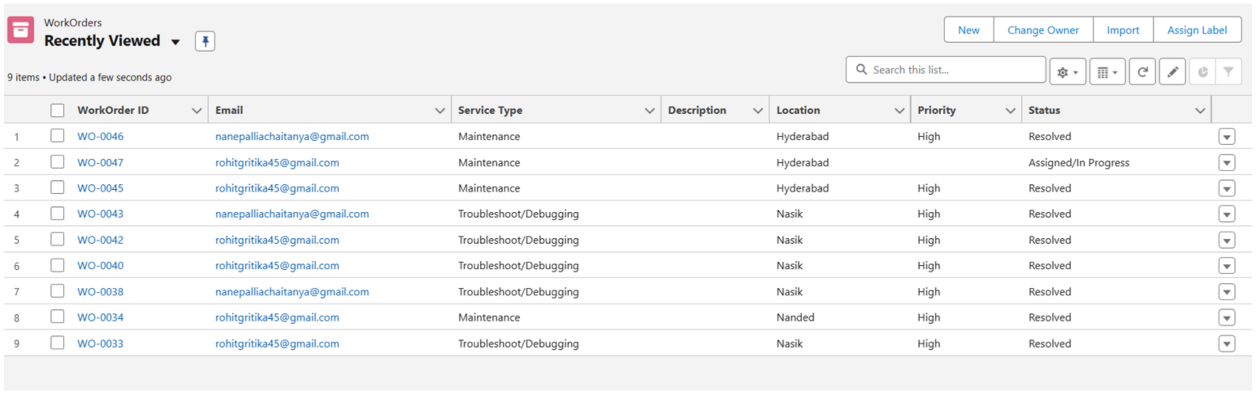
WorkOrder "WO-0047" was created.

items • Updated a few seconds ago

	WorkOrder ID	Email	Service Type	Description	Location	Priority	Status
1	WO-0047	rohitgritika45@gmail.com	Maintenance		Hyderabad		New
2	WO-0046	nanepalliachaitanya@gmail.com	Maintenance		Hyderabad	High	Assigned/In Progress
3	WO-0045	rohitgritika45@gmail.com	Maintenance		Hyderabad	High	Resolved
4	WO-0043	nanepalliachaitanya@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved


WorkOrders
Recently Viewed
9 items • Updated a few seconds ago

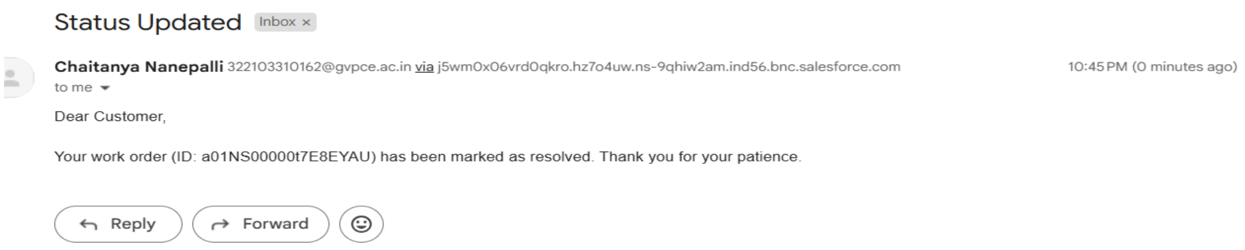
	WorkOrder ID	Email	Service Type	Description	Location	Priority	Status
1	WO-0047	rohitgritika45@gmail.com	Maintenance		Hyderabad		New
2	WO-0046	nanepalliachaitanya@gmail.com	Maintenance		Hyderabad	High	Resolved
3	WO-0045	rohitgritika45@gmail.com	Maintenance		Hyderabad	High	Resolved
4	WO-0043	nanepalliachaitanya@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
5	WO-0042	rohitgritika45@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
6	WO-0040	rohitgritika45@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
7	WO-0038	nanepalliachaitanya@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
8	WO-0034	rohitgritika45@gmail.com	Maintenance		Nanded	High	Resolved
9	WO-0033	rohitgritika45@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved


WorkOrders
Recently Viewed
9 items • Updated a few seconds ago

	WorkOrder ID	Email	Service Type	Description	Location	Priority	Status
1	WO-0046	nanepalliachaitanya@gmail.com	Maintenance		Hyderabad	High	Resolved
2	WO-0047	rohitgritika45@gmail.com	Maintenance		Hyderabad		Assigned/In Progress
3	WO-0045	rohitgritika45@gmail.com	Maintenance		Hyderabad	High	Resolved
4	WO-0043	nanepalliachaitanya@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
5	WO-0042	rohitgritika45@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
6	WO-0040	rohitgritika45@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
7	WO-0038	nanepalliachaitanya@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
8	WO-0034	rohitgritika45@gmail.com	Maintenance		Nanded	High	Resolved
9	WO-0033	rohitgritika45@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved

If a work order is created and no matching technician is currently available, the work order remains in a pending state. Once a suitable technician becomes available, the system automatically assigns the work order to them, creates an assignment record, sends an email notification to the technician, updates the work order status to **Assigned/In Progress**, and sets the technician's availability to **Not Available**.

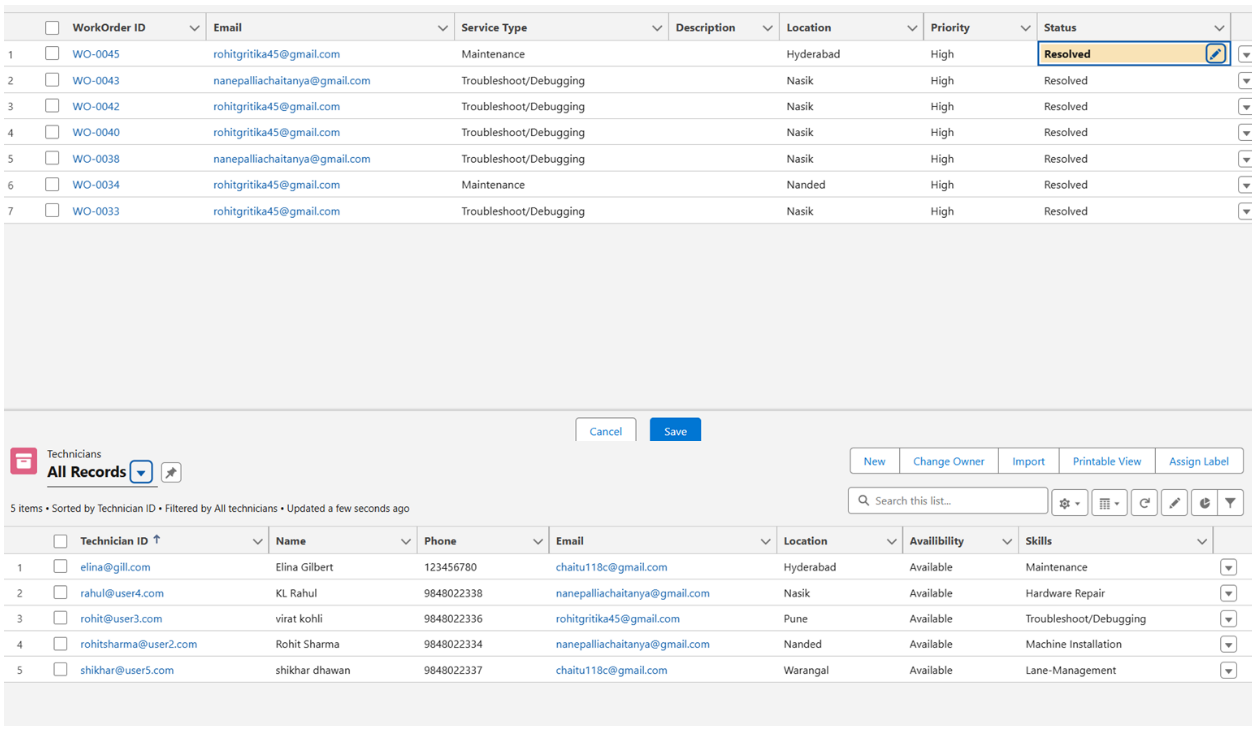
3.If the work order is completed:


Status Updated

Chaitanya Nanepalli 322103310162@gvpce.ac.in via j5wm0x06vrd0qkro.hz7o4uw.ns-9qhiw2am.ind56.bnc.salesforce.com 10:45 PM (0 minutes ago)
to me

Dear Customer,

Your work order (ID: a01NS00000t7E8EYAU) has been marked as resolved. Thank you for your patience.



		WorkOrder ID	Email	Service Type	Description	Location	Priority	Status
1		WO-0045	rohitgritika45@gmail.com	Maintenance		Hyderabad	High	Resolved
2		WO-0043	nanepalliachaitanya@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
3		WO-0042	rohitgritika45@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
4		WO-0040	rohitgritika45@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
5		WO-0038	nanepalliachaitanya@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved
6		WO-0034	rohitgritika45@gmail.com	Maintenance		Nanded	High	Resolved
7		WO-0033	rohitgritika45@gmail.com	Troubleshoot/Debugging		Nasik	High	Resolved

5 items • Sorted by Technician ID • Filtered by All technicians • Updated a few seconds ago

		Technician ID ↑	Name	Phone	Email	Location	Availibility	Skills
1		elina@gill.com	Elina Gilbert	123456780	chaitu118c@gmail.com	Hyderabad	Available	Maintenance
2		rahul@user4.com	KL Rahul	9848022338	nanepalliachaitanya@gmail.com	Nasik	Available	Hardware Repair
3		rohit@user3.com	virat kohli	9848022336	rohitgritika45@gmail.com	Pune	Available	Troubleshoot/Debugging
4		rohitsharma@user2.com	Rohit Sharma	9848022334	nanepalliachaitanya@gmail.com	Nanded	Available	Machine Installation
5		shikhar@user5.com	shikhar dhawan	9848022337	chaitu118c@gmail.com	Warangal	Available	Lane-Management

If the work is completed, the technician updates the work order status to **Resolved**. Upon this update, the system automatically changes the technician's availability status to **Available**.