Field Service WorkOrder Optimization

1. Project Overview

The Field Service Work Order Optimization System is designed to streamline operations for companies providing installation and repair services. By leveraging a robust database and intelligent algorithms, the system efficiently assigns work orders to skilled technicians based on factors such as location, availability, and expertise.

A prioritization algorithm ensures that tasks are assigned strategically, maximizing efficiency and reducing delays. Automated communication features keep technicians informed in real-time, fostering seamless coordination. Additionally, analytics tools provide actionable insights to drive continuous improvement.

This solution aims to enhance operational efficiency, reduce costs, and elevate customer satisfaction in the fast-paced field service industry.

2. Objectives

Business Goals:

- **Enhance Operational Efficiency**
  - Streamline work order assignment to reduce delays and improve service delivery times.
  - Optimize technician utilization by matching tasks to the most suitable and available resources.
- **Reduce Operational Costs**
  - Minimize travel time and fuel consumption through location-based task allocation.
  - Lower administrative overhead with automated scheduling and communication tools.
- **Improve Customer Satisfaction**
  - Ensure timely and accurate service delivery to enhance the customer experience.
  - Provide customers with real-time updates on service schedules and technician arrivals.
- **Maximize Resource Utilization**
  - Leverage technicians' specialized skills by assigning appropriate tasks based on expertise.
  - Increase technician productivity with better planning and reduced idle time.
- **Drive Continuous Improvement**
  - Utilize analytics to identify performance bottlenecks and areas for improvement.

    - Implement data-driven decisions to refine workflows and enhance service quality.
- **Support Scalability**
    - Enable the system to handle growing workloads and accommodate future business expansion.
    - Integrate seamlessly with other business systems to support evolving needs.

Specific Outcomes:

- **Improved Task Assignment Efficiency**
    - Reduction in average time taken to assign work orders to technicians.
    - Increased percentage of tasks assigned to the most suitable technicians based on skills and proximity.
- **Reduced Travel Time and Costs**
    - Decrease in average technician travel time per work order.
    - Reduction in fuel costs and vehicle wear-and-tear due to optimized route planning.
- **Higher Customer Satisfaction**
    - Increased customer ratings and positive feedback due to timely service.
    - Reduction in customer complaints related to delayed or missed appointments.
- **Increased Technician Productivity**
    - Higher number of completed work orders per technician per day.
    - Reduced idle time for technicians, resulting in better resource utilization.
- **Enhanced Communication**
    - Faster technician response times through real-time notifications and updates.
    - Improved coordination between dispatch teams and field technicians.
- **Data-Driven Decision Making**
    - Availability of detailed analytics for monitoring key performance indicators (KPIs), such as average resolution time and first-time fix rates.
    - Insights into trends and bottlenecks to facilitate continuous improvement.
- **Cost Savings**
    - Measurable reduction in operational expenses, including administrative overhead and logistical costs.
    - Improved profitability through more efficient use of resources.
- **Scalable Operations**
    - Ability to handle increased volumes of work orders without compromising efficiency or quality.
    - Seamless integration with existing and future enterprise systems.

3. Salesforce Key Features and Concepts Utilized

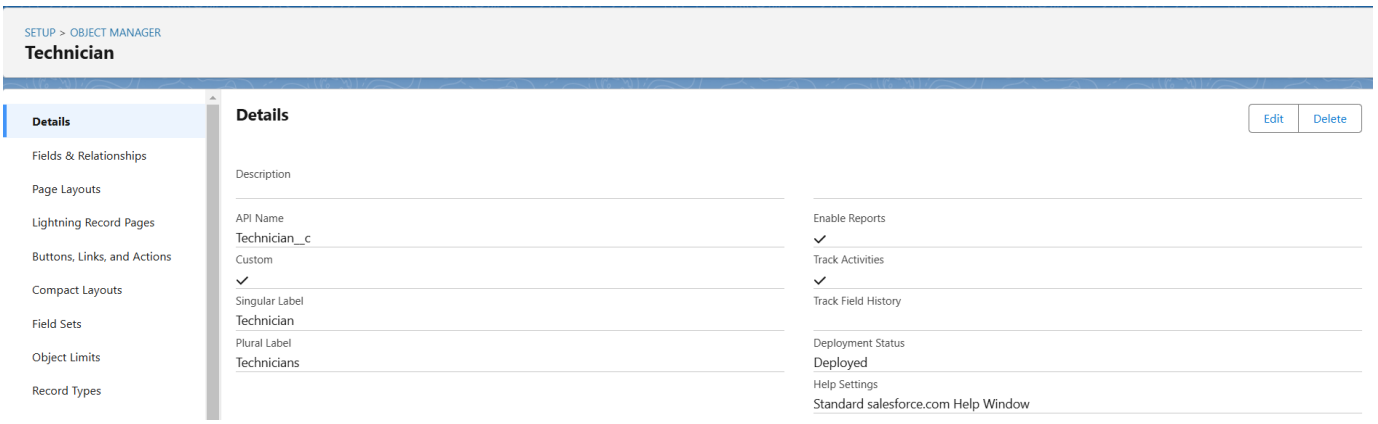
- **Custom Objects and Relationships**
  - **Work Order**, **Technician**, and **Assignment** objects created to model the field service process.
  - Relationships:
    - Lookup relationships between Work Order → Technician and Assignment → Work Order/Technician for linking related records.
- **Data Modeling and Fields**
  - Custom fields added to capture essential information:
    - **Work Order**: Status, Priority, Service Type, and Description.
    - **Technician**: Availability, Skills, Location, and Contact Details.
    - **Assignment**: Assignment Date, Completion Date, Technician ID, and Work Order ID.
- **UI Customization**
  - **Tabs**: Custom tabs created for easy access to Work Order, Technician, and Assignment records.
  - **Lightning App**: Consolidated these tabs into a unified interface for streamlined navigation.
- **Automation with Apex**
  - **Apex Triggers**:
    - Automated status updates (e.g., updating Work Order status when Assignment is marked completed).
    - Ensured Technician availability is updated after an Assignment is completed.
  - **Apex Classes**:
    - Implemented business logic for assigning technicians based on location, availability, and skills.
    - Utility methods for efficient operations and system integration.
- **Reports and Dashboards**
  - Created comprehensive **Reports**:
    - Monitor open Work Orders by status and priority.
    - Technician performance metrics like task completion rate.
    - Assignments completed within specified time frames.
  - Built **Dashboards** for real-time visualization of KPIs, including workload distribution, service efficiency, and customer satisfaction metrics.
- **Standard Salesforce Features**
  - **Profiles and Roles**: Defined access levels to secure sensitive data and limit access based on user roles (e.g., Dispatcher, Technician).
  - **Record Ownership**: Used the Owner field to track accountability for Work Orders and Assignments.

    - **Chatter**: Enabled team collaboration on Work Order records for updates and discussions.
- **Picklists and Validation**
    - Standardized input with picklists for fields like Status, Priority, Service Type, and Technician Availability.
    - Added validation rules to ensure data consistency (e.g., mandatory fields before completing a Work Order).
- **Analytics and Metrics**
    - **Custom Dashboards**:
        - Track ongoing Assignments, high-priority Work Orders, and technician workload.
        - KPIs like first-time resolution rate and average response time visualized effectively.
- **Mobile Accessibility**
    - Leveraged Salesforce's mobile-ready features for technicians to access Assignments and update statuses in the field.
- **Security and Compliance**
    - Implemented role-based access control to secure records based on profiles.
    - Used field-level security to protect sensitive data like customer contact details.

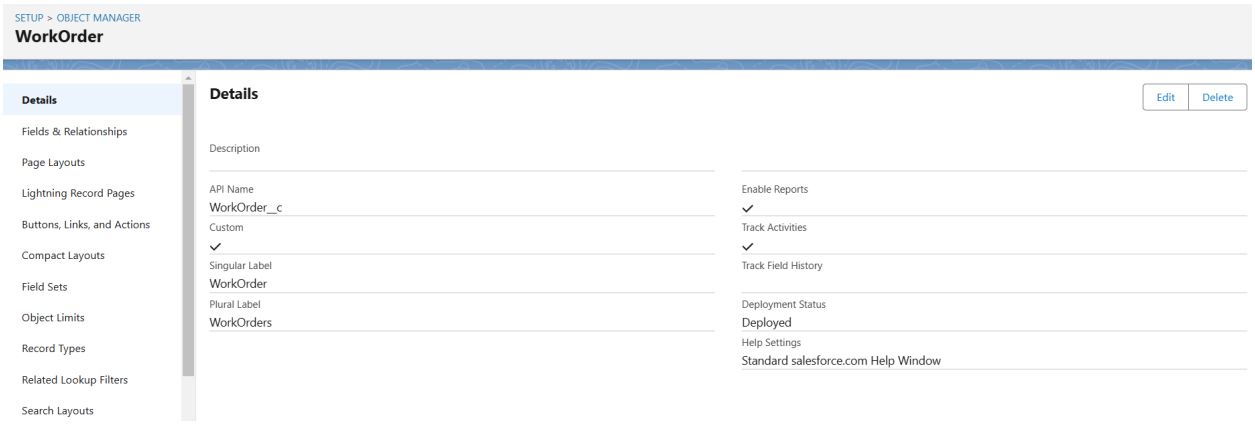
4. Detailed Steps to Solution Design

Create Objects From Spreadsheet

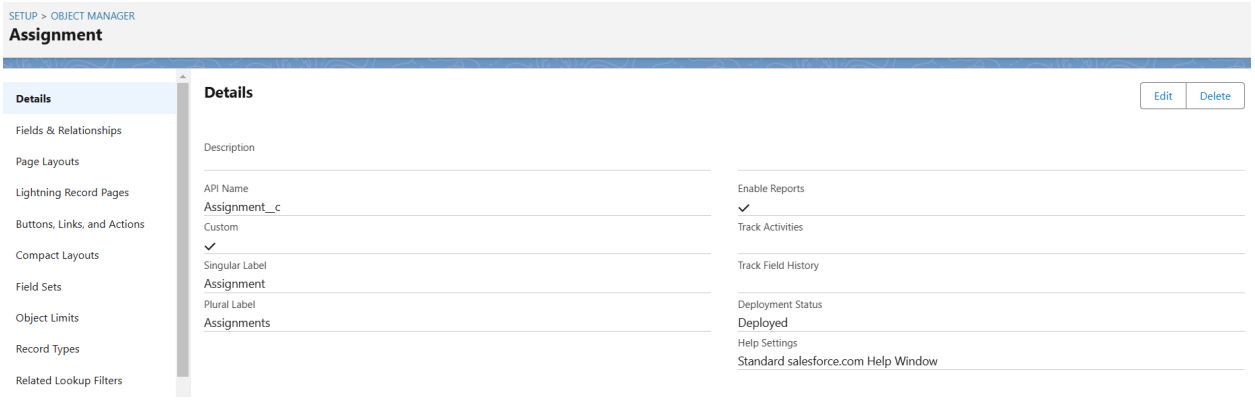
- **Create Technician Object:** The **Technician** object stores critical details about field technicians, including their availability, skills, and location. It features key fields such as Technician ID, Name, Email, Phone, Availability (Picklist), Location (Picklist), and Skills (Picklist). Relationships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



- **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.



- **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.



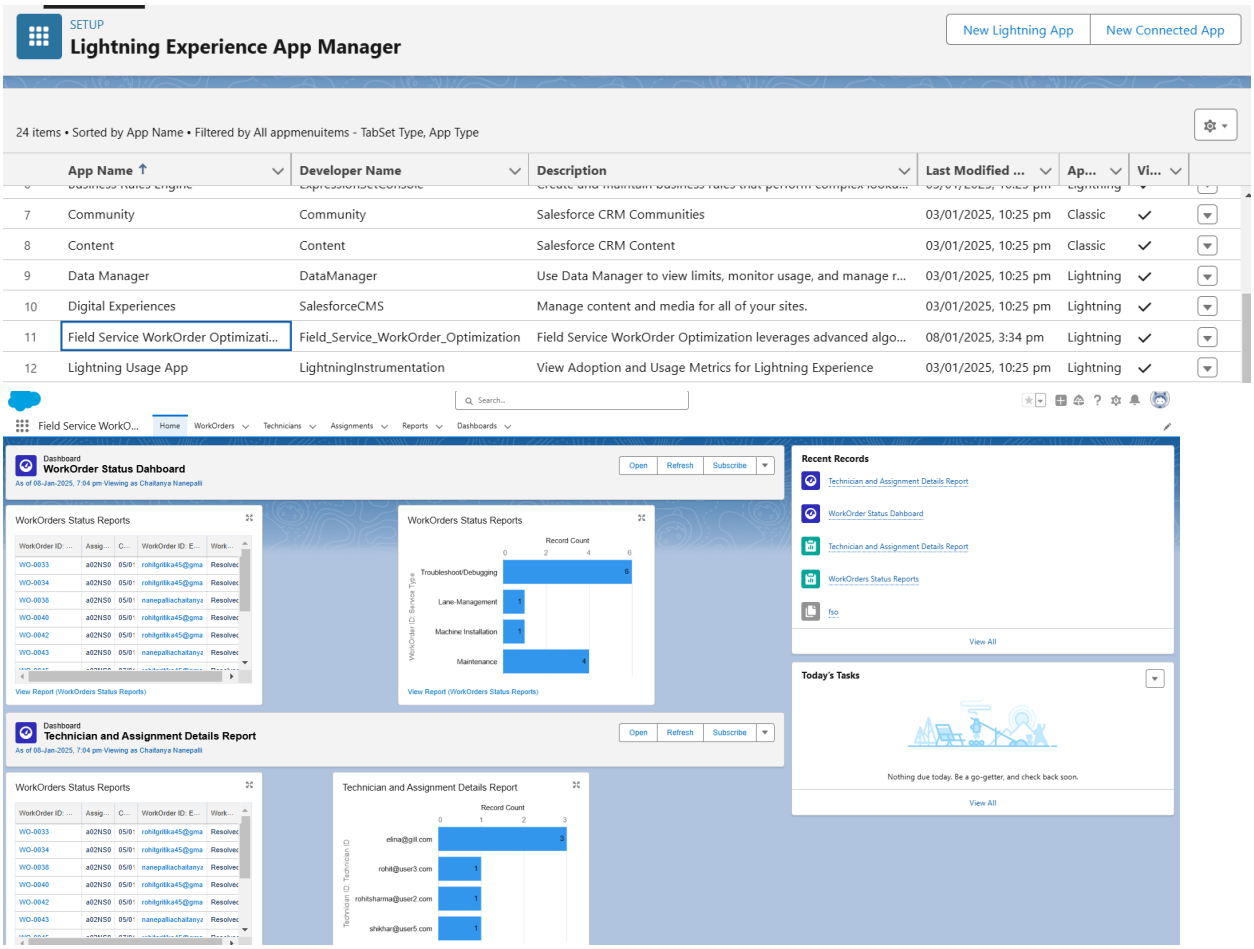
Create A Custom Tab

- **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.

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

Create The Lightning App

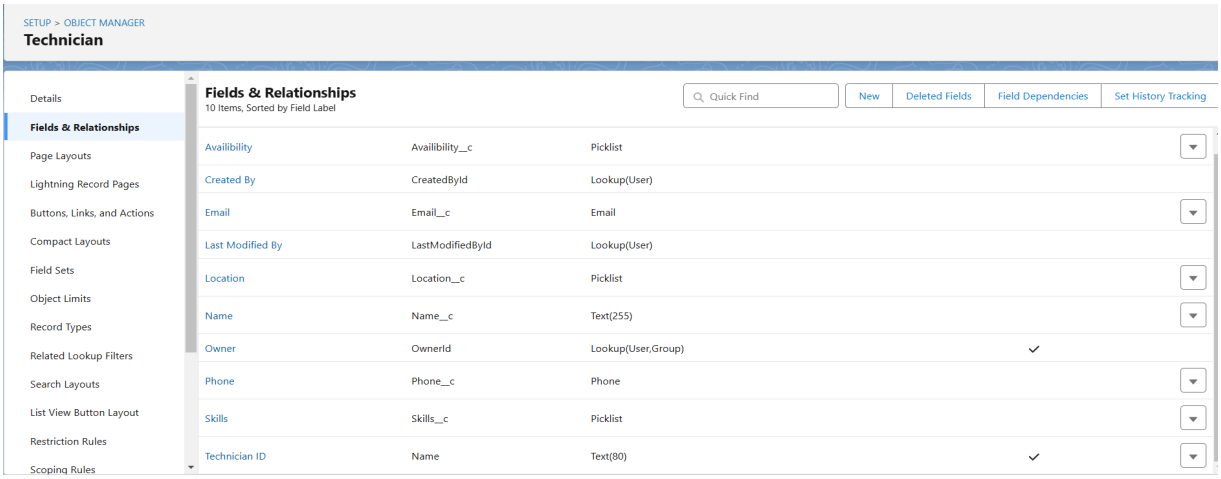
A **Lightning App** is created to consolidate the **Work Order**, **Technician**, and **Assignment** tabs into a unified interface. This app provides users with easy access to all relevant records and ensures streamlined navigation across different objects. It enhances the user experience by organizing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



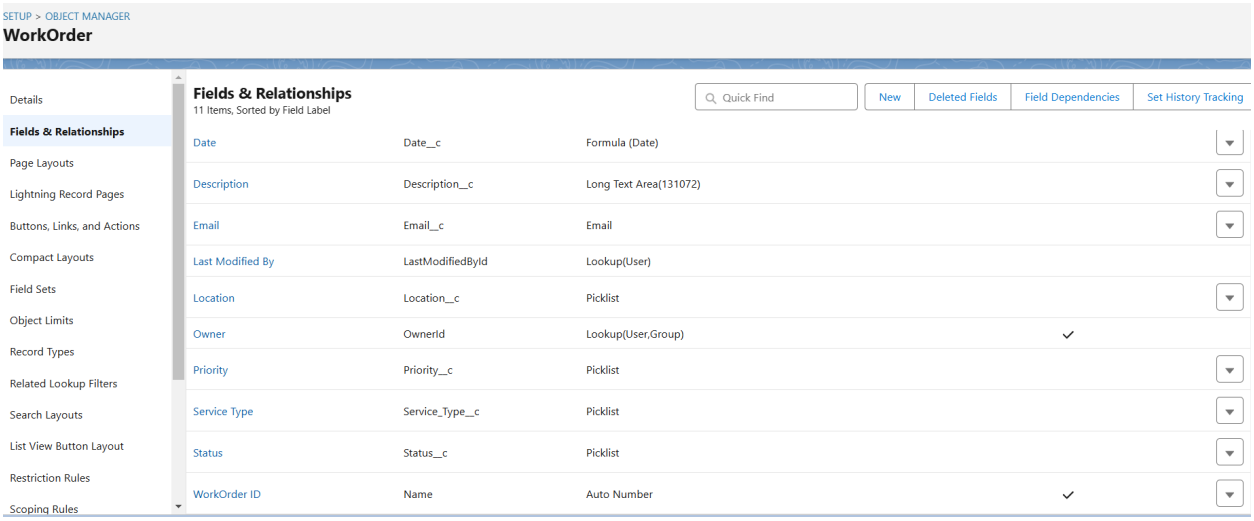
Create Fields and Relationships

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

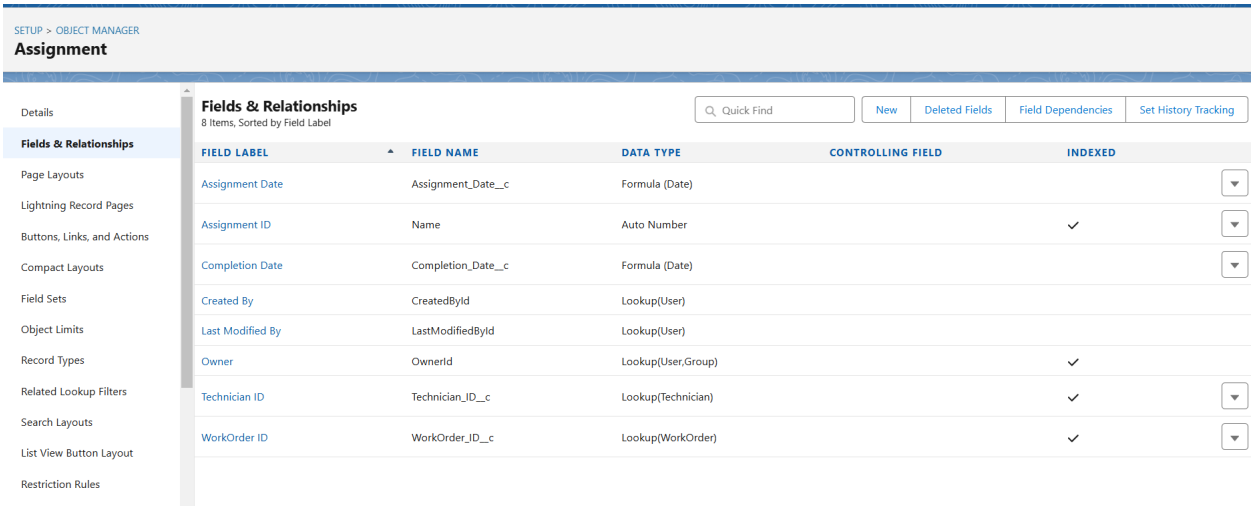
- **Technician:**



- **WorkOrder:**

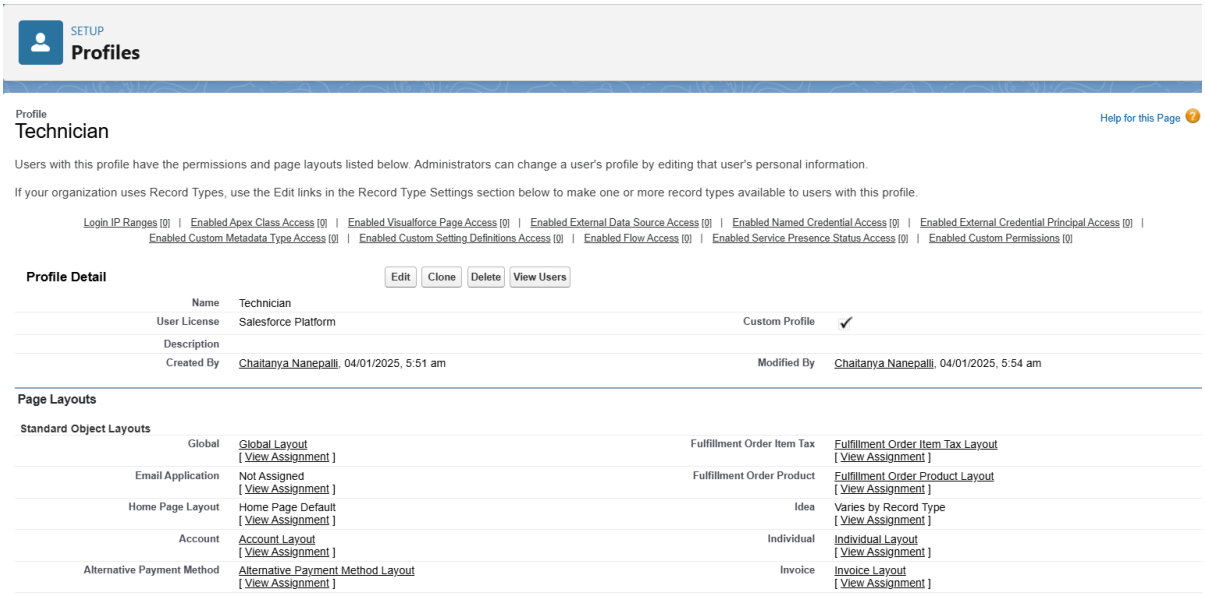


- **Assignment**

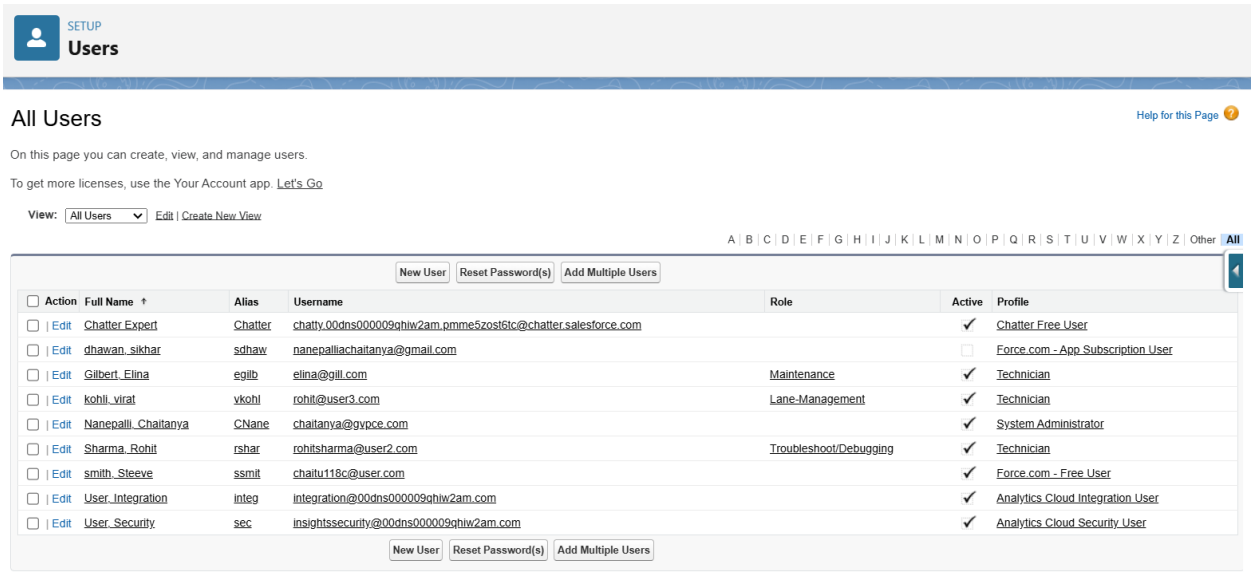


Create Technician Profile

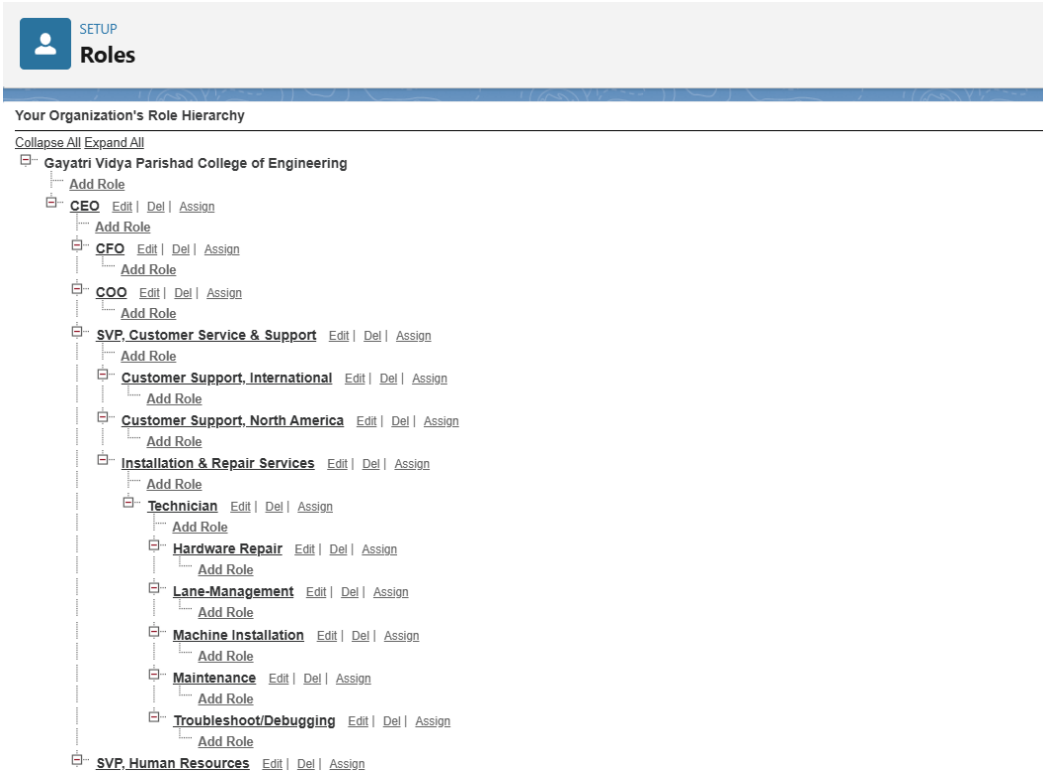
The **Technician Profile** defines permissions and access for technicians in the Salesforce Org. It grants visibility to relevant objects like **Work Order** and **Assignment**, allowing technicians to view, update, and complete their tasks. Access is restricted to only records assigned to them, ensuring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



Create Users

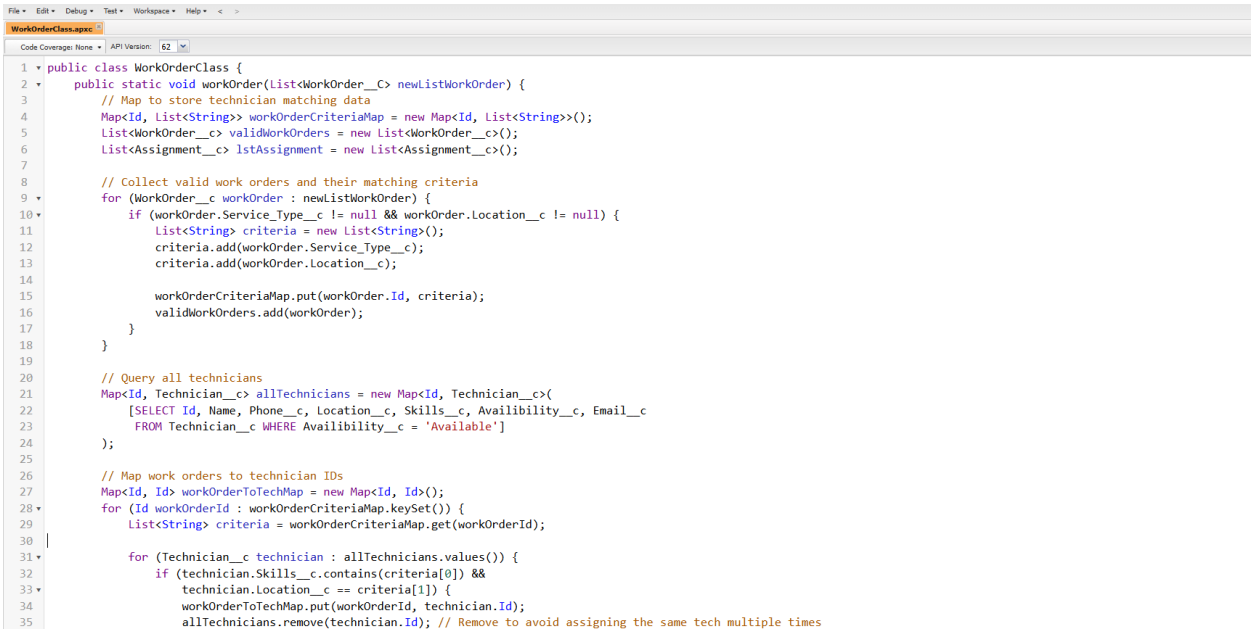


Create Technician Role under Services and Maintainance and create roles under Technician and Assign it to Users:



Create Apex Classes & Triggers

- WorkOrder Class:



```
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.');
        }
    }
}
```

- 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());
        }
    }
}
```

- 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();
    }
}
```

- # 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
        }

```

```
        // 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;
        }
```

- **AssingmentHandler Class**:

```
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());
        }
```

- **Assignment Trigger**

```
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);
    }
}
```

- **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);

    }
}
```

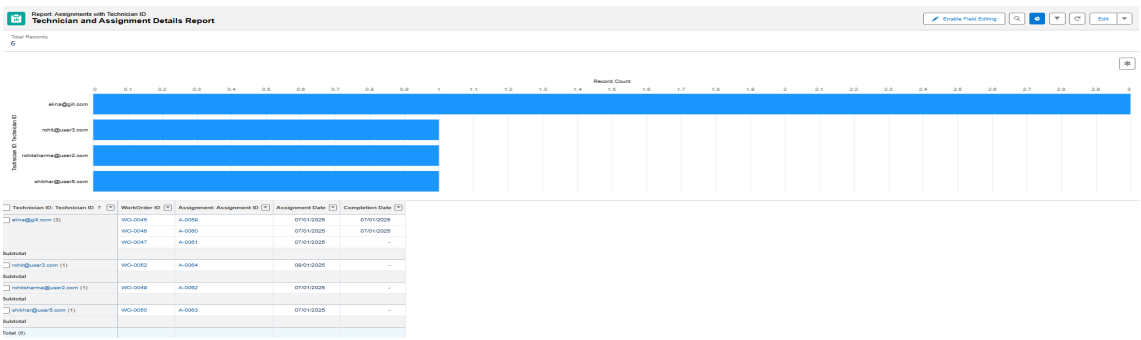
- **CreateTechnicianOnUserCreation Trigger**

```
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));
    }
}
```

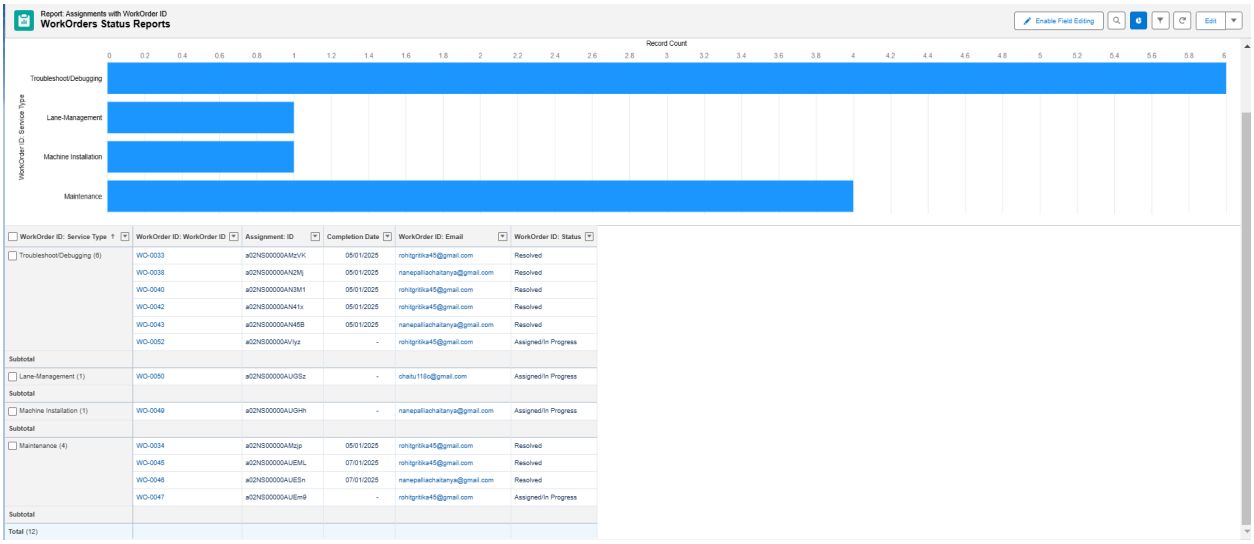
Create Reports And Dashboards:

Reports:

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



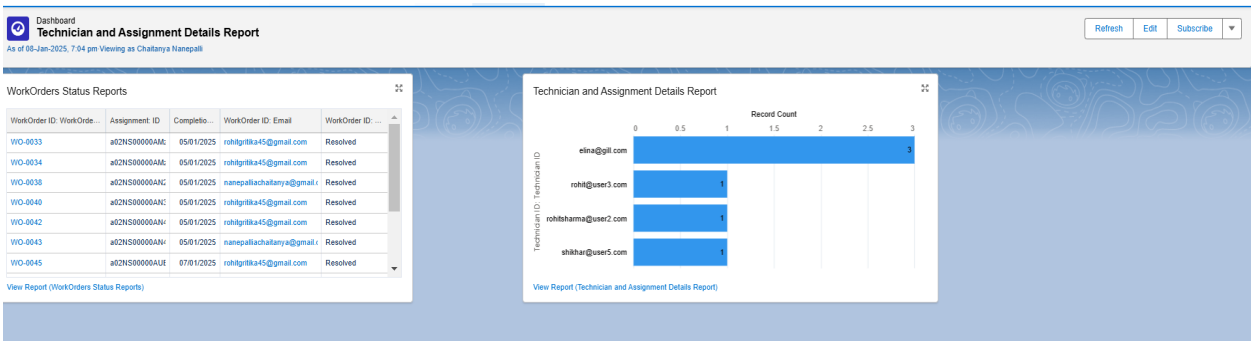
Work Order Report: Tracks work orders by status, priority, and service type.



Dashboards:

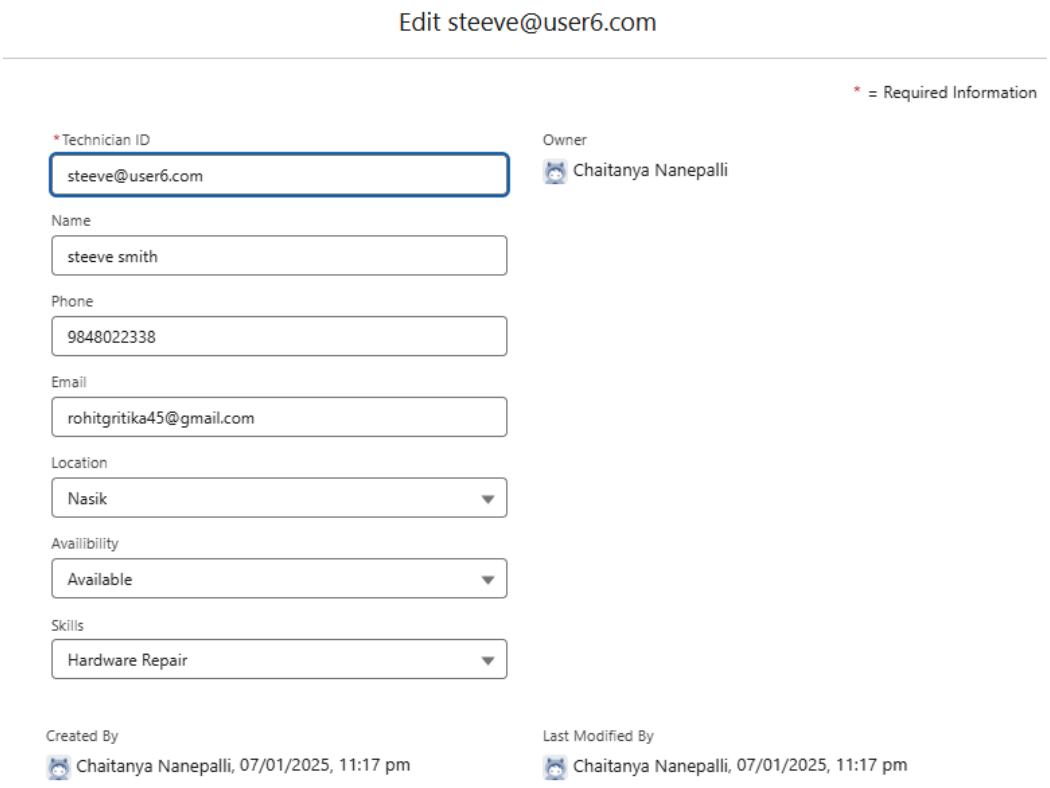
- **Work Order Status Dashboard:**

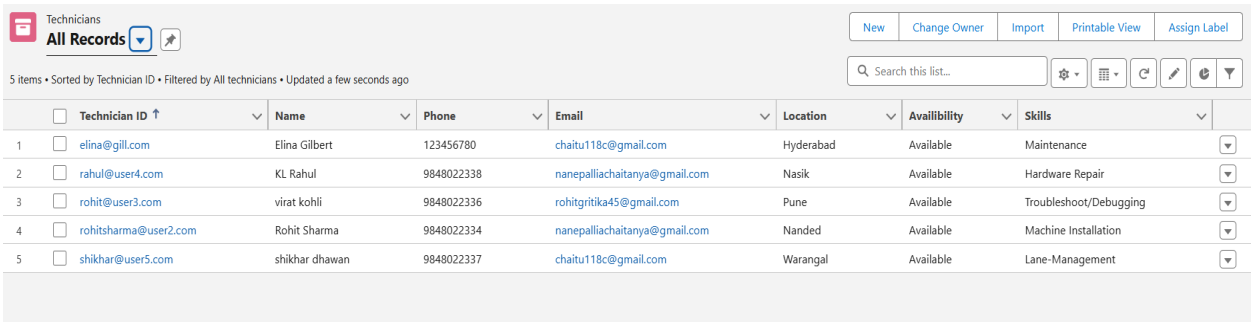
- **Technician and Assignment Details Dashboard:**



Testing and Validation:

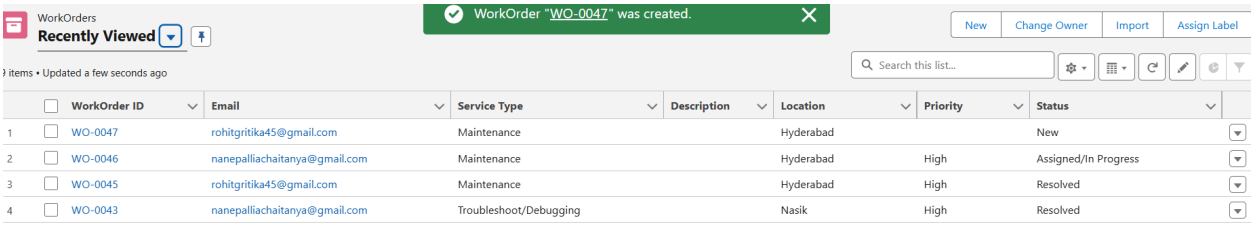
1. **Create users under Technician profile such that a record is created in Technician Object with the Users details:(or)**
   Create users under Technician Profile directly:





	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

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



	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 workorder is assigned and there is a available technician then it assigns to then it assigns to the technician

Related **Details**

Assignment ID
A-0059

Owner
Chaitanya Nanepalli

WorkOrder ID
WO-0045

Technician ID
elina@gill.com

Assignment Date
07/01/2025

Completion Date

Created By
Chaitanya Nanepalli, 07/01/2025, 10:44 pm

Last Modified By
Chaitanya Nanepalli, 07/01/2025, 10:44 pm

	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

WorkOrder Assignment Inbox ×

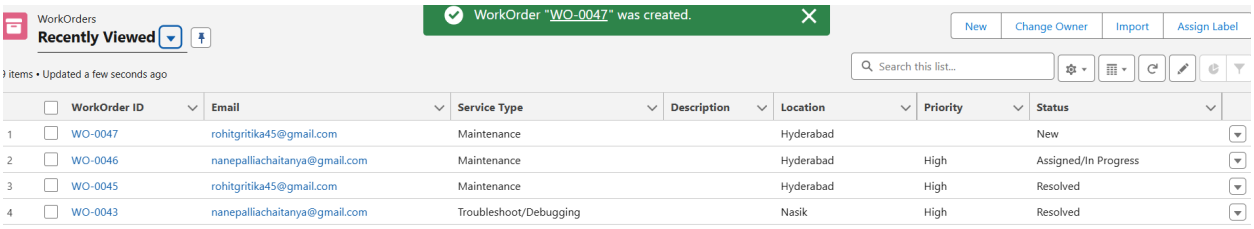
Chaitanya Nanepalli via lbbw8g8gnpi329jw.7zaa.ns-9qhiw2am.ind56.bnc.salesforce.com 10:44 PM (3 minutes ago)
to me

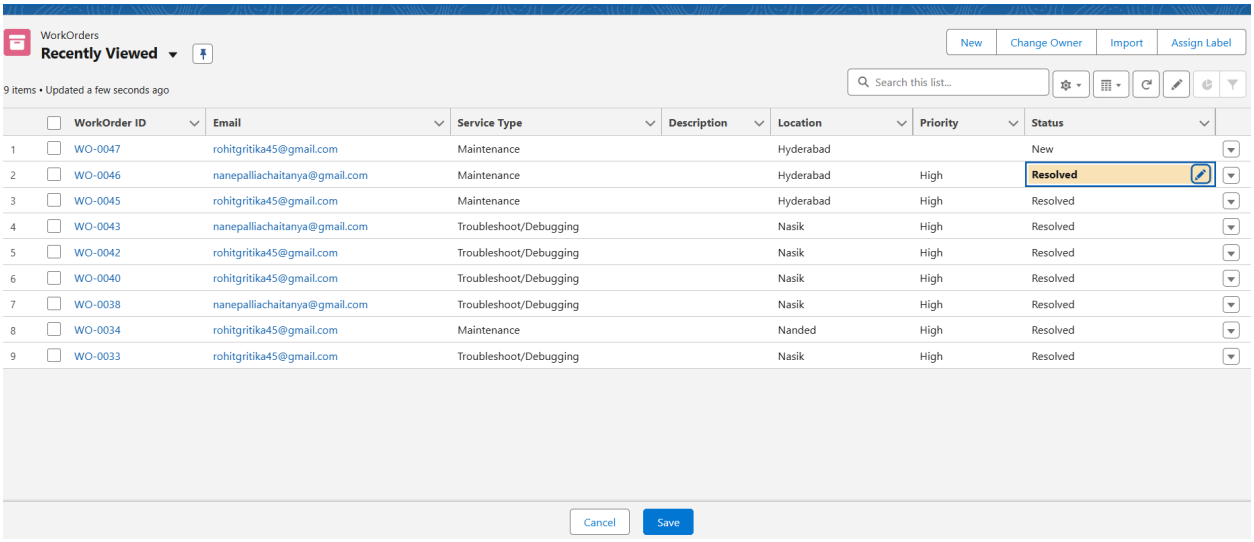
The following WorkOrder has been assigned to you.

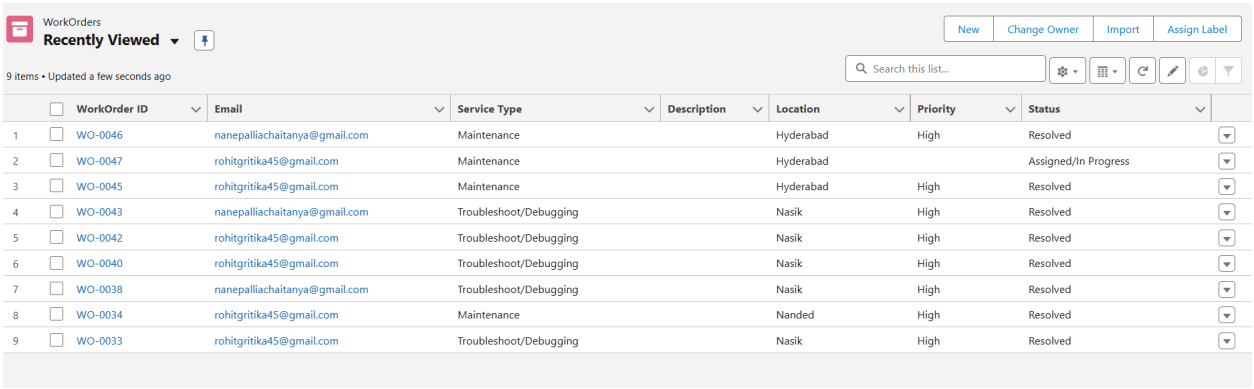
Reply Forward

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

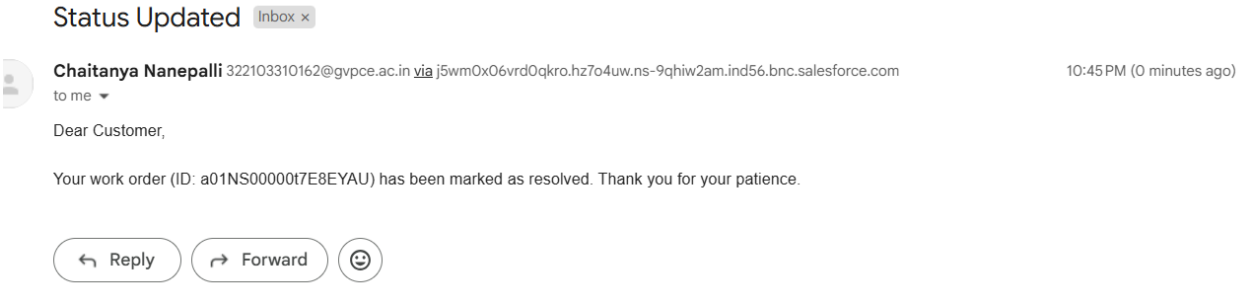


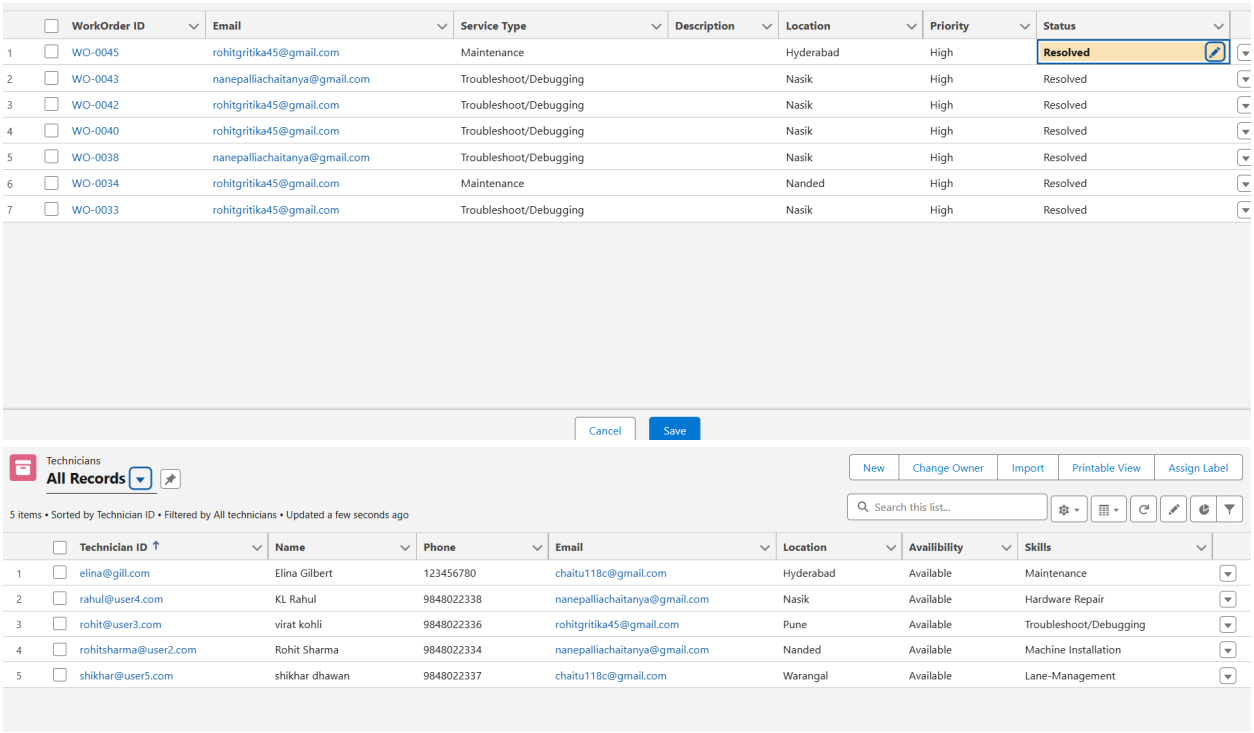




If a work order is created and no matching technician is currently available, the work order remains in a pending state. Once a suitable technician becomes available, the system automatically assigns the work order to them, creates an assignment record, sends an email notification to the technician, 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:





	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

Cancel Save

	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**.