**FIELD SERVICE WORKORDER OPTIMIZATION**

**Project Title:** Field Service WorkOrder Optimization

**College Name:** Ideal Institute of Technology

**Team ID:** LTVIP2025TMID31104

**Team Size:** 4

**Team Members Details:**

| **Name** | **Email ID** | **Role** |
| --- | --- | --- |
| Hem Shankar Trinath Pitani | pitanihemanth9@gmail.com | Team Leader |
| Erothu Venkata Siva Sai | erothusadhivik@gmail.com | Team Member |
| Doddi Durga Harshith | sridurgaharshith@gmail.com | Team Member |
| Dadala Gangadhar | bunnygangadhar8@gmail.com | Team Member |

**1. INTRODUCTION**

**1.1 Project Overview:** Field Service WorkOrder Optimization*,* is built on Salesforce to improve how work orders are assigned and managed. It helps managers assign the right technician automatically and track work progress using custom apps, reports, and automation tools.

**1.2 Purpose:** The purpose of this project to make field service work faster and easier by automating technician assignments, sending email alerts, and providing reports. This saves time, reduces errors, and keeps both technicians and customers informed.

**2. IDEATION PHASE**

**2.1 Problem Statement**

Creating a problem statement from the customer’s perspective is essential for designing solutions that address their needs. The Customer Problem Statement template helps the team focus on customer challenges in field service work order processes, fostering empathy and enabling the development of solutions that enhance efficiency and satisfaction.

A screenshot of a computer

AI-generated content may be incorrect.

**Example:**

Chart, treemap chart

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem Statement (PS)** | **I am (Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | A field service technician | Complete work orders efficiently and on time | I receive work orders with incomplete information or last-minute schedule changes. | The current scheduling system lacks real-time updates and integration with field data. | Frustrated and stressed, as I cannot plan my day effectively or meet customer expectations. |
| PS-2 | A customer awaiting field service | Get timely updates on when a technician will arrive to resolve my issue. | I receive vague or no communication about the technicians arrival time. | The work order system does not provide real-time tracking or automated notifications | Anxious and dissatisfied, as I have to adjust my schedule without clear information. |

**2.2 Empathy Map Canvas**

The Empathy Map Canvas is a visual tool designed to capture insights into the behaviors, attitudes, and needs of users. It enables teams to deeply understand their users, ensuring that solutions are tailored to address real problems from the user’s perspective.

For the Field Service WorkOrder Optimization, our primary users are:

* Field Service Managers
* Field Technicians
* Customers awaiting service

**Purpose of empathy mapping**

* Gain a deeper understanding of users’ goals, challenges, and behaviors to design an effective work order optimization platform.
* Identify pain points and opportunities to enhance operational efficiency and user satisfaction.
* Align the solution with the needs and workflows of managers, technicians, and customers.

The Empathy Map includes:

**Field Service Managers :**

* **Pains**: Frustrated by scheduling errors, stressed about meeting KPIs, and overwhelmed by manual processes.
* **Gains**: Desire a system that automates scheduling, provides real-time insights, and reduces customer complaints.
* **Goals**: Improve operational efficiency, reduce costs by 10%, and increase customer satisfaction by 25%.

**Field Technicians :**

* **Pains**: Frustrated by inefficient workflows, stressed about mismatched assignments, and annoyed by lack of real-time data.
* **Gains**: Want a single, user-friendly app for all job details, clear schedules, and fewer disruptions.
* **Goals**: Complete jobs faster, reduce downtime by 15%, and improve customer interactions.

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**2.3 Brainstorming**

Brainstorming fosters a collaborative environment where team members can freely contribute creative ideas to optimize field service work order processes. The goal is to prioritize innovative, out-of-the-box solutions that enhance efficiency, reduce costs, and improve customer satisfaction. This template guides your team through the ideation and prioritization process, whether in-person or remote, to develop actionable solutions for work order optimization.

Reference: <https://www.mural.co/templates/brainstorm-and-idea-prioritization>

**Step-1: Team Gathering, Collaboration and Select the Problem Statement**

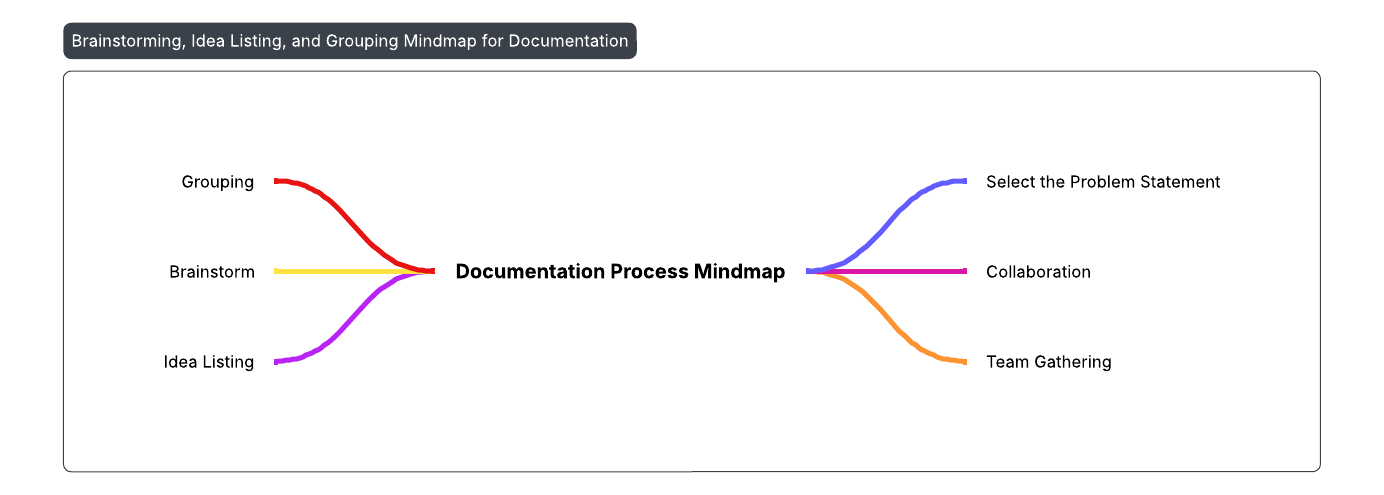
A close-up of a logo

AI-generated content may be incorrect.

* **Objective**: Assemble the team, define roles, and select a clear problem statement to address inefficiencies in field service work order management.
* **Problem Statement**: Current field service work order processes suffer from delays, miscommunication, and inefficient resource allocation, leading to increased operational costs and reduced customer satisfaction.
* **Activities**:
  + Gather cross-functional team members (e.g., field technicians, dispatchers, IT specialists, and customer service representatives).
  + Use collaborative tools (e.g., Mural, Miro) for virtual or hybrid brainstorming sessions.
  + Define the scope: Optimize scheduling, routing, and task allocation for field service work orders.
  + Assign a facilitator to guide discussions and a scribe to document ideas.

**Step-2: Brainstorm, Idea Listing and Grouping**

* **Objective:** Generate a high volume of ideas to address the problem statement, then group similar ideas for clarity.
* **Process:**
  + Conduct a 15–20 minute brainstorming session where all ideas are welcome, no matter how unconventional.
  + Use a timer to keep the session focused and encourage rapid idea generation.
  + List ideas on a digital whiteboard or sticky notes (virtual or physical).
  + **Group similar ideas into categories such as:**
    - **Scheduling Optimization**: Automated scheduling, real-time updates
    - **Routing Efficiency**: GPS-based routing, traffic-aware algorithms.
    - **Resource Allocation**: Technician skill matching, inventory management.
    - **Customer Communication**: Automated notifications, customer portals.
    - **Data Analytics**: Predictive maintenance, performance dashboards.



**Step-3: Idea Prioritization**

* **Objective**: Evaluate and prioritize ideas based on feasibility, impact, and alignment with project goals.
* **Process**:
  + Use a prioritization matrix to score ideas (e.g., Impact vs. Effort or Cost vs. Benefit).
  + Assign scores (1–4) for each idea based on:
    - **Impact**: How significantly does it improve efficiency or customer satisfaction? (1 = Low, 4 = High)
    - **Feasibility**: How practical is it to implement within 6–12 months? (1 = Difficult, 4 = Easy)
    - **Cost**: What is the estimated cost of implementation? (1 = High, 4 = Low)

**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey Map**

| **Stage** | **Customer Action** | **Customer Experience** | | **System Support (Salesforce)** | |
| --- | --- | --- | --- | --- | --- |
| **1. Service Request** | Customer raises a service request | | Wants quick and clear acknowledgment | WorkOrder created in Salesforce (WorkOrder\_\_c) |
| **2. Waiting** | Customer waits for technician assignment | | May feel anxious or uncertain | Apex class auto-assigns technician based on skill/location |
| **3. Notification** | Receives technician assignment email | | Feels informed and confident | Email sent using AssigningEmail class after Assignment creation |
| **4. Service Delivery** | Technician visits and resolves issue | | Gets service done with minimal delay | Technician record linked with Assignment & WorkOrder updates status |
| **5. Resolution Update** | Customer receives resolution email | | Feels assured service is completed | CompletionMail class sends email when WorkOrder is marked Resolved |
| **6. Feedback/Closure** | Customer may share feedback or close case | | Has a complete service experience | WorkOrder reports and dashboards show performance metrics |

**3.2 Solution Requirement**

* The solution requires creating custom Salesforce objects (Technician, WorkOrder, Assignment) and automating technician assignment using Apex triggers and classes.
* It also includes email notifications, reports, dashboards, and scheduled data cleanup for efficient field service management.

Solution Requirements are of two types:

1. Functional Requirements
2. Non-Functional Requirements

**Functional Requirements:**

| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
| --- | --- | --- |
| FR-1 | Technician Management | Create Technician Update Technician Delete Technician |
| FR-2 | WorkOrder Creation | Create WorkOrder Set Priority, Service Type, and Location |
| FR-3 | Technician Assignment | Auto-assign Technician based on Skills & Location |
| FR-4 | Email Notifications | Send email on Technician Assignment Send email when WorkOrder is Resolved |
| FR-5 | Status Management | Update WorkOrder Status Set Completion Date |
| FR-6 | Reporting & Dashboard | Generate Reports View Dashboard Metrics |

**Non-Functional Requirements:**

.

|  |  |  |
| --- | --- | --- |
| FR No. | Non-Functional Requirement | Description |
| NFR-1 | Usability | User-friendly interface for managing Technicians and WorkOrders |
| NFR-2 | Security | Role-based access control and secure data handling |
| NFR-3 | Reliability | Consistent and error-free automation of assignments and email alerts |
| NFR-4 | Performance | Quick loading and execution of Apex triggers and reports |
| NFR-5 | Availability | System should be accessible 99.9% of the time during business hours |
| NFR-6 | Scalability | Can support increasing number of WorkOrders and Technicians without degradation |

**3.3 Data Flow Diagram**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. For this project, the DFD illustrates how work orders are created, processed, assigned to technicians, and how status updates are managed.

DFD Level 0:

1. WorkOrder is created by admin or customer.  
2. System checks technician availability and auto-assigns if matched.  
3. Assignment is stored and email notification is sent.  
4. Technician updates work status.

5. Completed orders are monitored and removed after 30 days via batch process.

**User Stories**

Use the below template to list all the user stories for the product.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **User Type** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance Criteria** | **Priority** | **Release** |
| Admin | Technician Management | FSW-1 | As an admin, I can add and manage technician records including location and skill set. | Technician record is created and updated | High | Sprint-1 |
| Admin | WorkOrder Creation | FSW-2 | As an admin, I can create work orders with service type, location, and priority. | Work order is visible in the system | High | Sprint-1 |
| System | Auto Assignment | FSW-3 | As a system, I want to auto-assign a technician based on availability, skill, and location. | Technician is auto-assigned on creation | High | Sprint-1 |
| Technician | Email Notification | FSW-4 | As a technician, I should receive email alerts upon assignment. | Technician receives notification email | Medium | Sprint-2 |
| Technician | Status Update | FSW-5 | As a technician, I can update the status of work orders. | Work order status changes to In-progress/Resolved | High | Sprint-2 |
| Manager | Dashboard | FSW-6 | As a manager, I can view work order stats in dashboards. | Dashboard displays current status metrics | Medium | Sprint-2 |

**3.4 Technology Stack**

The solution is developed on the Salesforce cloud platform, which includes a Lightning user interface, Apex-based backend logic, and standard Salesforce cloud infrastructure. It enables automatic assignment of field technicians for service requests based on criteria like skill set, availability, and location. External email APIs and Apex triggers handle.

**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Component | Description | Technology |
| 1 | User Interface | Users interact with system through Salesforce Lightning Experience | Salesforce Lightning, HTML |
| 2 | Application Logic-1 | Custom logic for assignment and notifications | Apex Classes & Triggers |
| 3 | Application Logic-2 | Auto-scheduling & batch job for data clean-up | Apex Batch & Scheduler |
| 4 | Database | Stores WorkOrders, Technicians, Assignments | Salesforce Custom Objects |
| 5 | Cloud Database | Hosted on Salesforce infrastructure | Salesforce Cloud |
| 6 | File Storage | Stores metadata and attachments | Salesforce File Storage |
| 7 | External API-1 | Send email alerts for assignment | Salesforce Messaging API |
| 8 | Infrastructure (Server / Cloud) | Application hosted in the cloud | Salesforce Platform (Cloud) |

**Table-2: Application Characteristics:**

|  |  |  |
| --- | --- | --- |
| S.No | Characteristics | Technology |
| 1 | Open-Source Frameworks | Salesforce Lightning Framework, Apex (proprietary but extensible) |
| 2 | Security Implementations | Field-level security, role-based access control, and encrypted communication |
| 3 | Scalable Architecture | Cloud-based architecture with multi-tenant scalability on Salesforce |
| 4 | Availability | Hosted on Salesforce Cloud with 99.9% SLA |
| 5 | Performance | Uses optimized triggers and batch jobs for bulk processing and high-speed execution |

**4. PROJECT DESIGN**

**4.1 Problem Solution Fit** Field service teams face delays and errors due to manual work order assignment and lack of real-time updates. This project solves the problem by automating technician assignment, sending email alerts, and providing dashboards for better tracking and faster service delivery.

Problem–Solution Fit means you have:

1. Field service operations face delays due to manual technician assignment and lack of real-time updates.
2. Technicians and customers are often not informed promptly, causing poor service experiences.

The Problem-Solution Fit for Field Service Work Order Optimization focuses on identifying critical challenges faced by field service operations and ensuring the proposed solution effectively addresses these issues. It enables businesses to align technological solutions with customer and operational needs, enhancing efficiency and adoption. The template helps identify behavioral patterns, optimize workflows, and improve communication strategies to solve urgent, frequent, or costly problems in field service management.

**Purpose:**

Field service teams face operational inefficiencies due to:

* Manual work order assignment
* Unoptimized technician scheduling
* Poor visibility into service status
* Communication delays between technicians and management
* Lack of automation in technician notifications and work tracking

These lead to:

* Increased service delays
* Low technician productivity
* Poor customer satisfaction

Implement a custom Salesforce solution using:

* **Custom Objects:** Technician, WorkOrder, Assignment
* **Automated Workflows:** Apex triggers/classes for auto-assignment and email notifications
* **Lightning App Interface:** Centralized access to WorkOrders and Assignments
* **Reports & Dashboards:** Real-time insights into technician performance and work order status
* **Scheduled Apex Jobs:** Periodic cleanup of outdated assignment

| **Pain Point** | **Solution Feature** |
| --- | --- |
| Manual technician assignment | Apex Trigger + Class (WorkOrderTrigger, WorkOrderClass) |
| No notifications to technicians | AssigningEmail class with AssignmentTrigger |
| No customer update after resolution | CompletionMail class in WorkOrderTrigger |
| Unmanaged old data | Batch + Scheduled Apex (RecordDeletions, ScheduleClass) |
| No analytics | Custom Reports and Dashboards in Lightning App |

### **Why Problem–Solution Fit is Important**

* It ensures the solution you're building actually solves a real, frequent, and painful problem.
* It helps avoid wasting time and resources on features that users don’t need.
* It increases adoption by aligning with how users already work (e.g., email alerts, mobile access).
* It builds trust among stakeholders by delivering clear, useful results—like faster technician assignments and better tracking.
* For your project, it proves that automating field service tasks solves real issues and improves efficiency.
* **Example:**

In field service operations, managers used to manually assign technicians, which caused delays and errors. Your Salesforce solution automatically assigns technicians based on their skills and availability. This directly solves the problem, saves time, and improves customer satisfaction — proving a strong problem–solution fit.

**Problem–Solution Fit Checklist:**

* **Is the problem frequent, urgent, or costly?**
* Is the problem clearly defined?
* Is the solution directly addressing the problem?
* Does the solution align with user behavior?
* can the solution scale with business growth?

**4.2 Proposed Solution**

The project team shall fill the following information in the proposed solution template to outline the Field Service WorkOrder Optimization project, addressing inefficiencies in work order management and enhancing operational efficiency.

**Problem to Be Solved**

* Manual assignment of work orders leads to delays and errors
* Technician skills, availability, and location are not considered in assignment decisions
* Slow and inefficient scheduling causes delayed service delivery
* Technicians and customers are not notified promptly about updates or assignments
* Managers lack real-time visibility into work order progress and technician status
* Old and completed records are not cleaned up, cluttering the system
* Poor communication and tracking lead to reduced customer satisfaction

### **Solution Description**

The solution is a Salesforce-based system designed to automate and streamline field service operations. It uses custom objects (Technician, WorkOrder, Assignment) along with Apex triggers and classes to automatically assign technicians based on skill, location, and availability. Email notifications are sent to both technicians and customers for real-time communication. Managers can track progress using reports and dashboards, and outdated records are cleaned using scheduled Apex jobs. This ensures faster service, fewer errors, and improved satisfaction for all users involved.

|  |  |  |
| --- | --- | --- |
| S.No. | Parameter | Description |
| 1 | Problem Statement (Problem to be solved) | Manual allocation of field service technicians leads to delays, errors, and inefficient service. There is a need for an automated system that intelligently assigns technicians based on skills, location, and availability to handle service work orders efficiently. |
| 2 | Idea / Solution description | The project provides a Salesforce-based solution that automates work order creation, technician assignment, and notification processes. It utilizes custom objects (Technician, WorkOrder, Assignment), Apex logic, and automation tools to dynamically assign the most suitable technician to each work order. Additionally, it includes scheduled cleanup of outdated records and real-time dashboards for tracking performance. |
| 3 | Novelty / Uniqueness | The solution intelligently maps technicians to work orders using logic that matches skillsets, location, and availability—minimizing manual intervention. Real-time email notifications, automatic data clean-up via Apex Scheduler, and dynamic dashboards provide a comprehensive field service management experience within Salesforce. |
| 4 | Social Impact / Customer Satisfaction | Faster and more accurate technician assignment leads to quicker issue resolution and improved customer satisfaction. It reduces manual errors and increases accountability. Technicians receive timely assignments and updates, enhancing their productivity. |
| 5 | Business Model (Revenue Model) | The solution can be offered as a Salesforce app or consulting service to organizations with large field teams (telecom, utilities, IT services). Revenue can be generated through licensing, subscription, or one-time implementation fees. |
| 6 | Scalability of the Solution | Built on the Salesforce platform, the solution is scalable to handle large volumes of work orders and technician records. Additional features such as route optimization, mobile support, or AI-driven dispatching can be integrated to enhance functionality. |

**4.3 Solution Architecture**

* Salesforce Developer Org

The **Solution Architecture** for the Field Service WorkOrder Optimization project is designed to address key inefficiencies in field service management—especially around technician assignment, scheduling delays, and communication gaps. It utilizes Salesforce’s customizable platform, automation tools, and reporting capabilities to build a scalable, cloud-native solution.

* Automate and optimize technician assignment based on skills, availability, and location
* Provide real-time visibility into work order status for stakeholders
* Improve operational efficiency using Apex triggers, workflows, and batch jobs
* Enable technician and customer communication through automated email notifications
* Create insightful dashboards and reports for monitoring service performance

### **Architecture Overview:**

This project uses **Salesforce** as a centralized cloud platform. The system architecture includes:

* **Custom Objects:** Technician, WorkOrder, Assignment
* **Lightning App:** Unified interface for managers to view/manage assignments
* **Apex Classes & Triggers:**
  + WorkOrderClassfor auto-assignment logic
  + AssigningEmailto notify technicians
  + CompletionMail to notify customers upon resolution
* **Batch Apex & Scheduled Class:**
  + Record Deletions to clean old records
  + ScheduleClassfor monthly cleanup
* **Reports & Dashboards:**
  + Technician performance, WorkOrder completion, and status tracking

**Features:**

|  |  |
| --- | --- |
| **Automated Technician Assignment** | Improves dispatch efficiency and reduces manual errors |

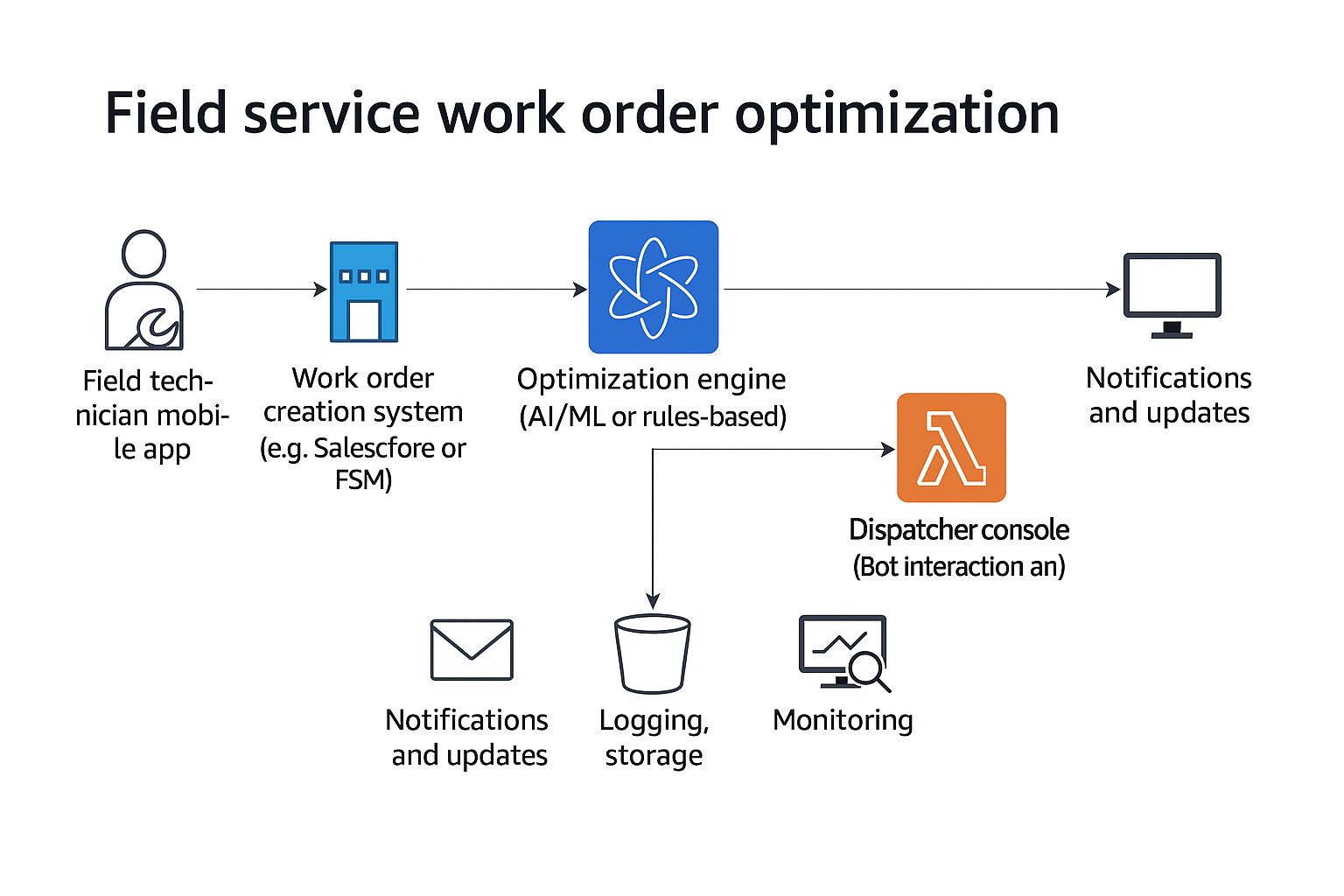
|  |  |
| --- | --- |
| **Email Alerts to Technicians & Customers** | Enhances communication and satisfaction |

|  |  |
| --- | --- |
| **Scheduled Data Cleanup** | Ensures system remains optimized and clutter-free |

|  |  |
| --- | --- |
| **Reports & Dashboards** | Provides operational insights and tracks KPIs |

|  |  |
| --- | --- |
| **Custom App Interface** | Easy navigation and usability for admin and technicians |

**Example - Solution Architecture Diagram:**



*Figure 1: Architecture and data flow of the voice patient diary sample application*

### **4.3.1.Architecture Layers**

| **Layer** | **Components / Technologies** |
| --- | --- |
| **Presentation Layer** | Lightning App, Tabs (WorkOrder, Technician, Assignment), Reports & Dashboards |
| **Business Logic Layer** | Apex Classes (WorkOrderClass, AssigningEmail, CompletionMail), Triggers (WorkOrderTrigger, AssignmentTrigger) |
| **Data Access Layer** | SOQL Queries, Custom Object Relationships (Lookup Fields), Formula Fields |
| **Data Layer** | Custom Objects: Technician, WorkOrder, Assignment; Standard Object: User |
| **Integration Layer** | Email Messaging via Apex (Messaging.sendEmail), Scheduled Jobs (Apex Scheduler) |

### 

### **4.3.2. Key Components**

* Technician (Custom Object)
* WorkOrder (Custom Object)
* Assignment (Custom Object)
* Lightning App
* WorkOrderClass (Apex Class)
* AssigningEmail (Apex Class)
* CompletionMail (Apex Class)
* RecordDeletions (Apex Class)
* ScheduleClass (Apex Class)
* WorkOrderTrigger (Apex Trigger)
* AssignmentTrigger (Apex Trigger)
* Lookup Fields
* Formula Fields
* Technician Profile
* Reports
* Dashboards
* Email Notifications
* Batch Job
* Scheduled Apex

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

* **Week 1**: Salesforce Setup, Creation of Technician, WorkOrder, and Assignment Objects with Fields
* **Week 2**: Tabs, Lightning App, Page Layouts, Picklists, and Email Templates
* **Week 3**: Validation Rules, Formula Fields, Lookup Relationships, Process Automations
* **Week 4**: Apex Triggers & Classes, Testing (UAT), Reports, Dashboards, and Final Documentation

### **Project Objectives**

* Automate the field service work order assignment lifecycle
* Improve technician allocation efficiency and work order transparency
* Minimize manual assignment work through automation and triggers
* Deliver a scalable, secure, and user-friendly Salesforce Lightning solution

### **Project Scope**

**In Scope:**

* Work order creation, status updates, and technician assignments
* Location and skill-based technician mapping
* Email notifications for assignments and completions
* Role-based access through custom profiles (Technician)
* Reports and dashboards on technician performance and work status
* Scheduled cleanup of completed assignments

**Out of Scope:**

* Integration with third-party dispatching or GPS systems
* Payment gateway or billing system integrations

**Milestones and Deliverables:**

| **Milestone** | **Description** | **Duration** |
| --- | --- | --- |
| Requirement Gathering | Identify user needs, technician workflow, and assignment logic | 2 Days |
| Design Phase | Create ERD, architecture diagrams, and wireframes | 2 Days |
| Development Phase | Create custom objects, fields, triggers, automation flows | 5 Days |
| Testing Phase | Perform functional, unit, and UAT testing | 2 Days |
| Deployment & Training | Deploy to production and train users/admins | 1 Day |
| Project Documentation | Prepare and finalize all project-related documents | 1 Day |

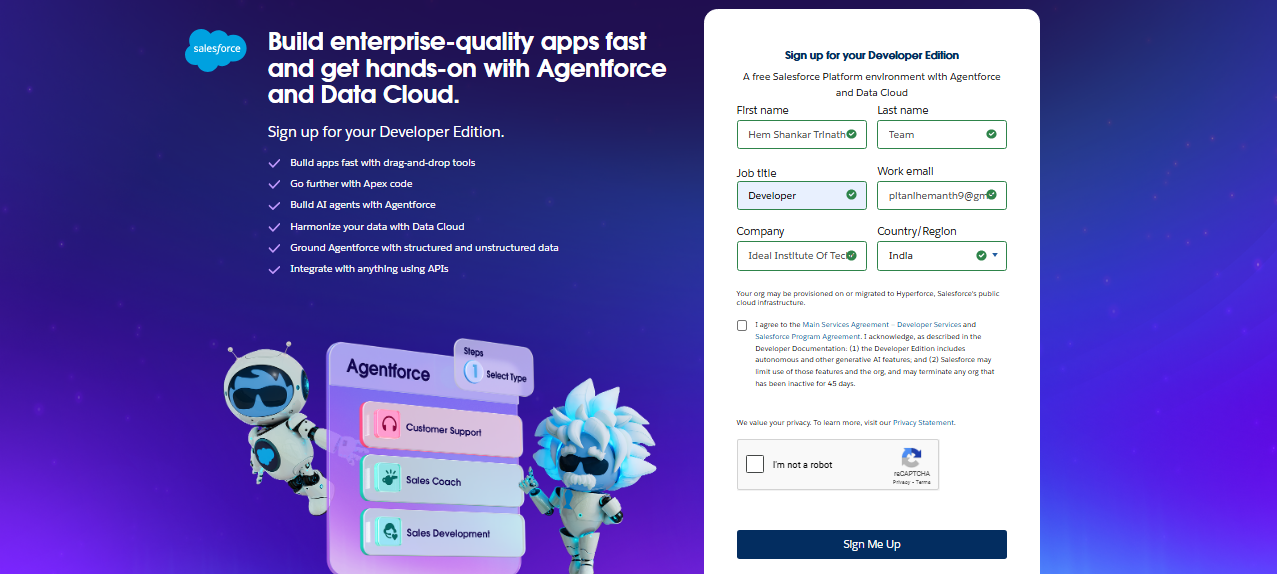
### **Tools and Technologies Used**

* Project Management Tools: Trello / Excel / Salesforce Agile Accelerator
* Development Platform: Salesforce Lightning Platform
* Version Control: Salesforce Change Sets
* Communication: Email, Zoom (for stakeholder meetings)
* Documentation: Microsoft Word, PowerPoint

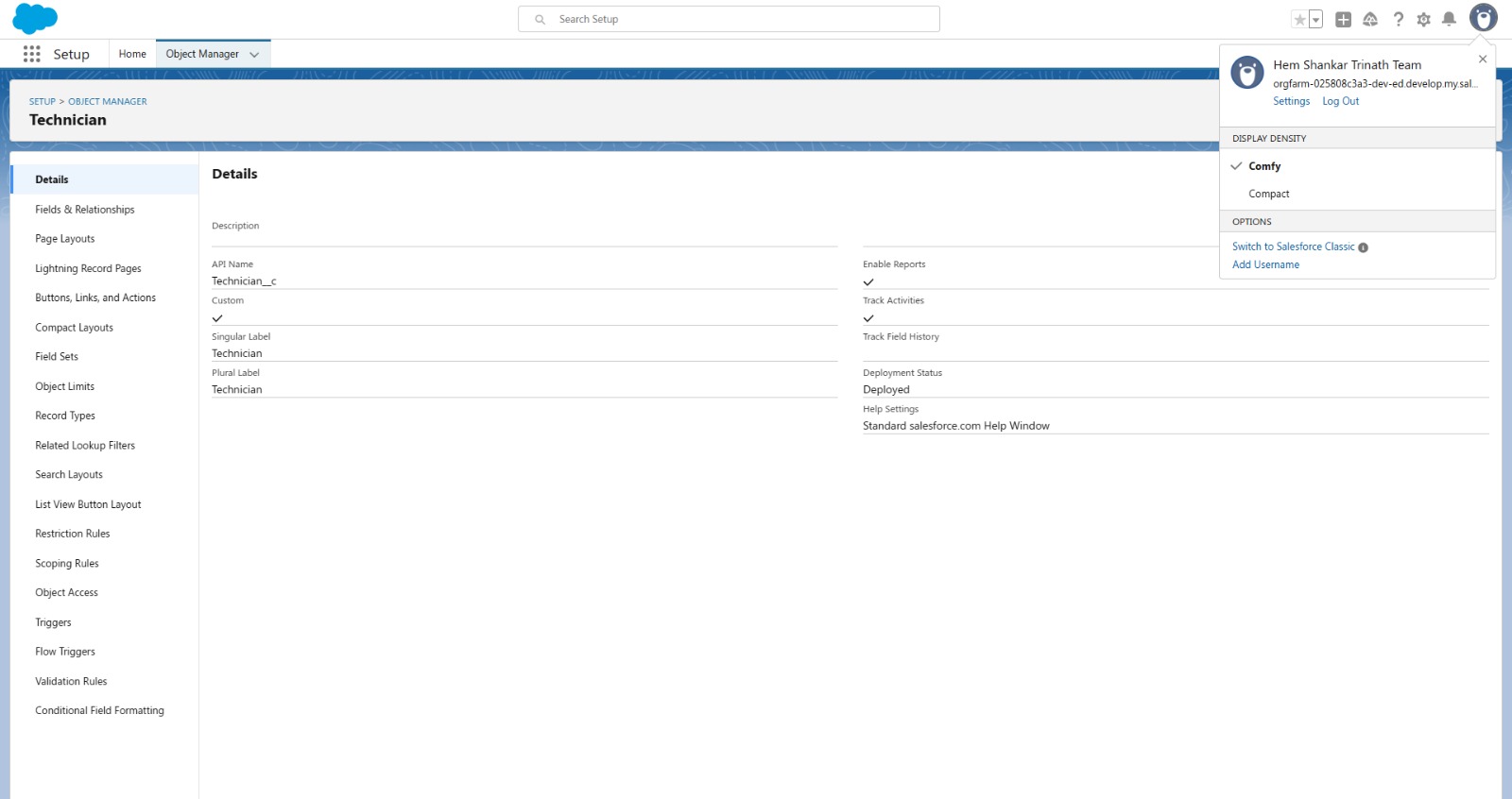
**6. PROJECT DEVELOPMENT PHASE**

**6.1 Development Activities**

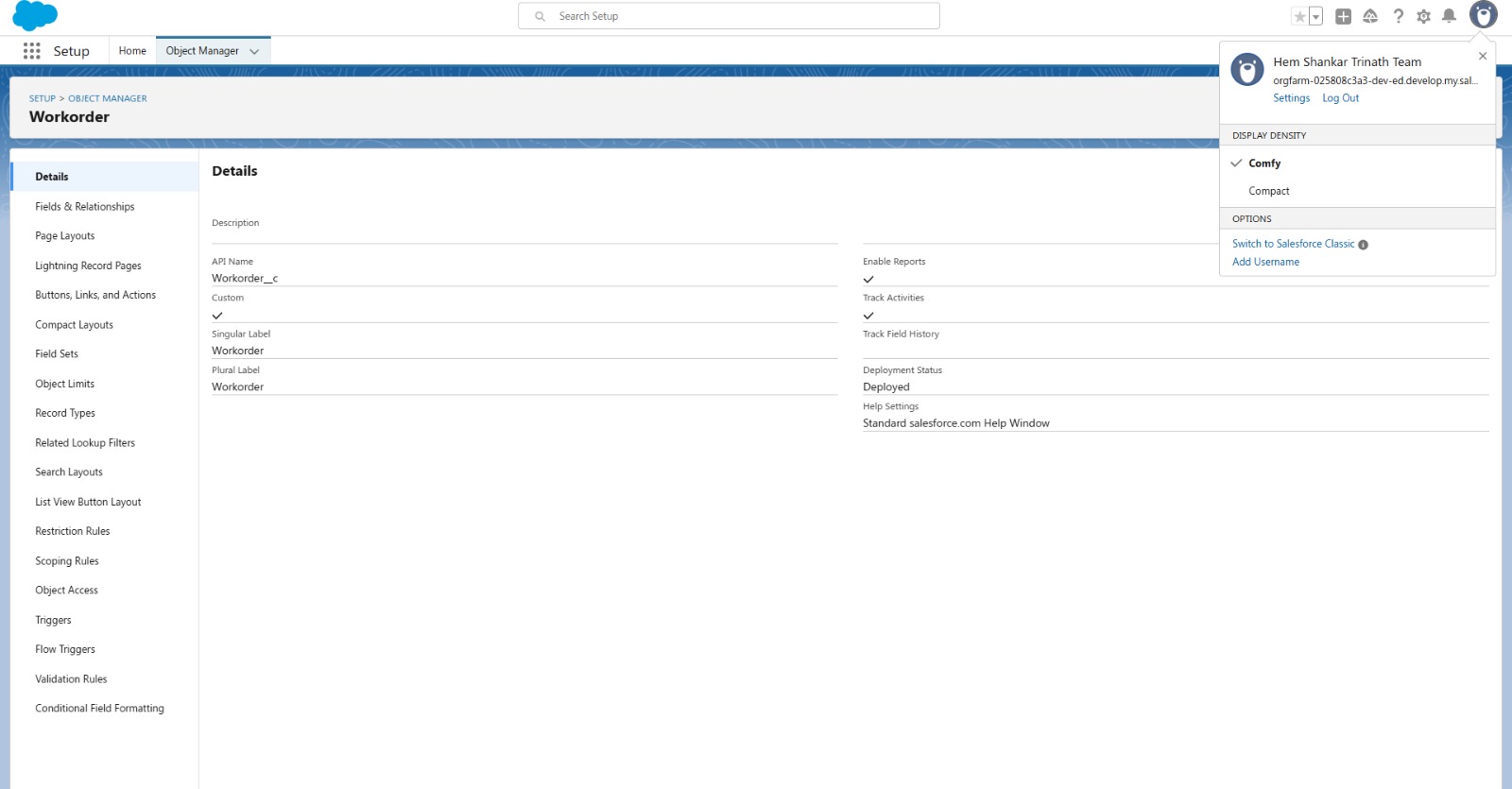
* Created Developer Org and set up objects: **Technician, WorkOrder, Assignment**



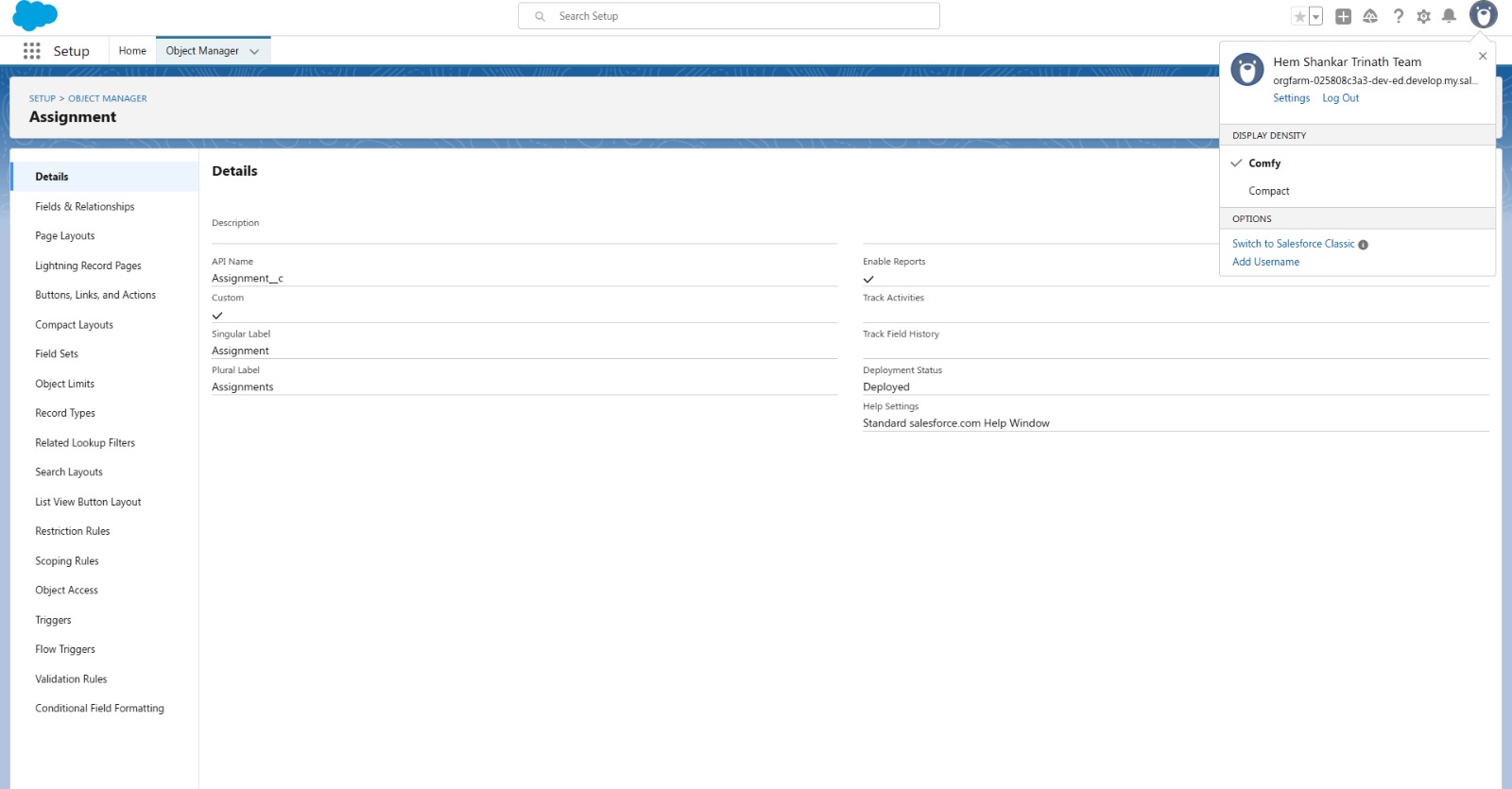
**Technician Object:**

****

**Workorder Object:**

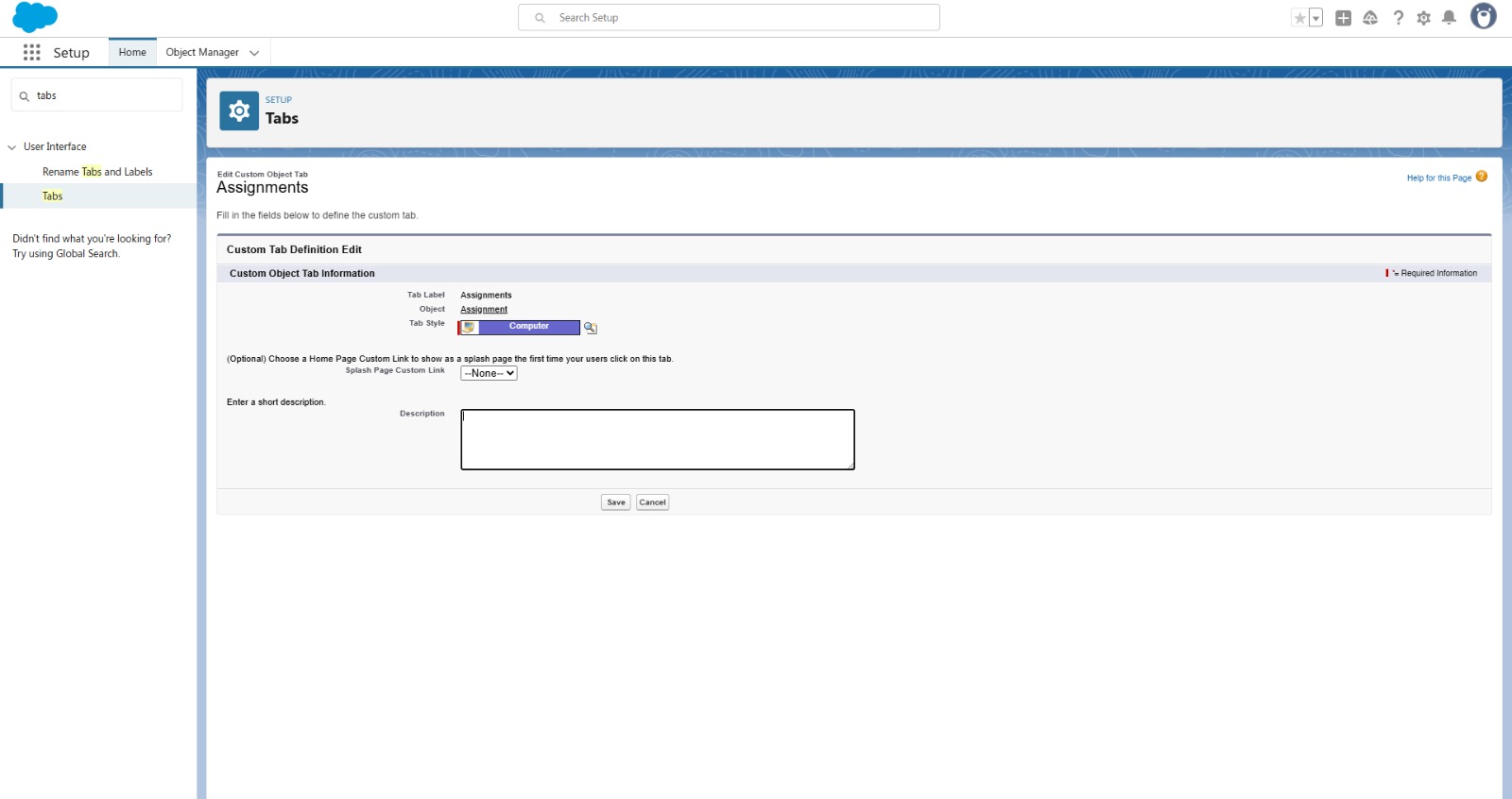
****

**AssignmentObject:**

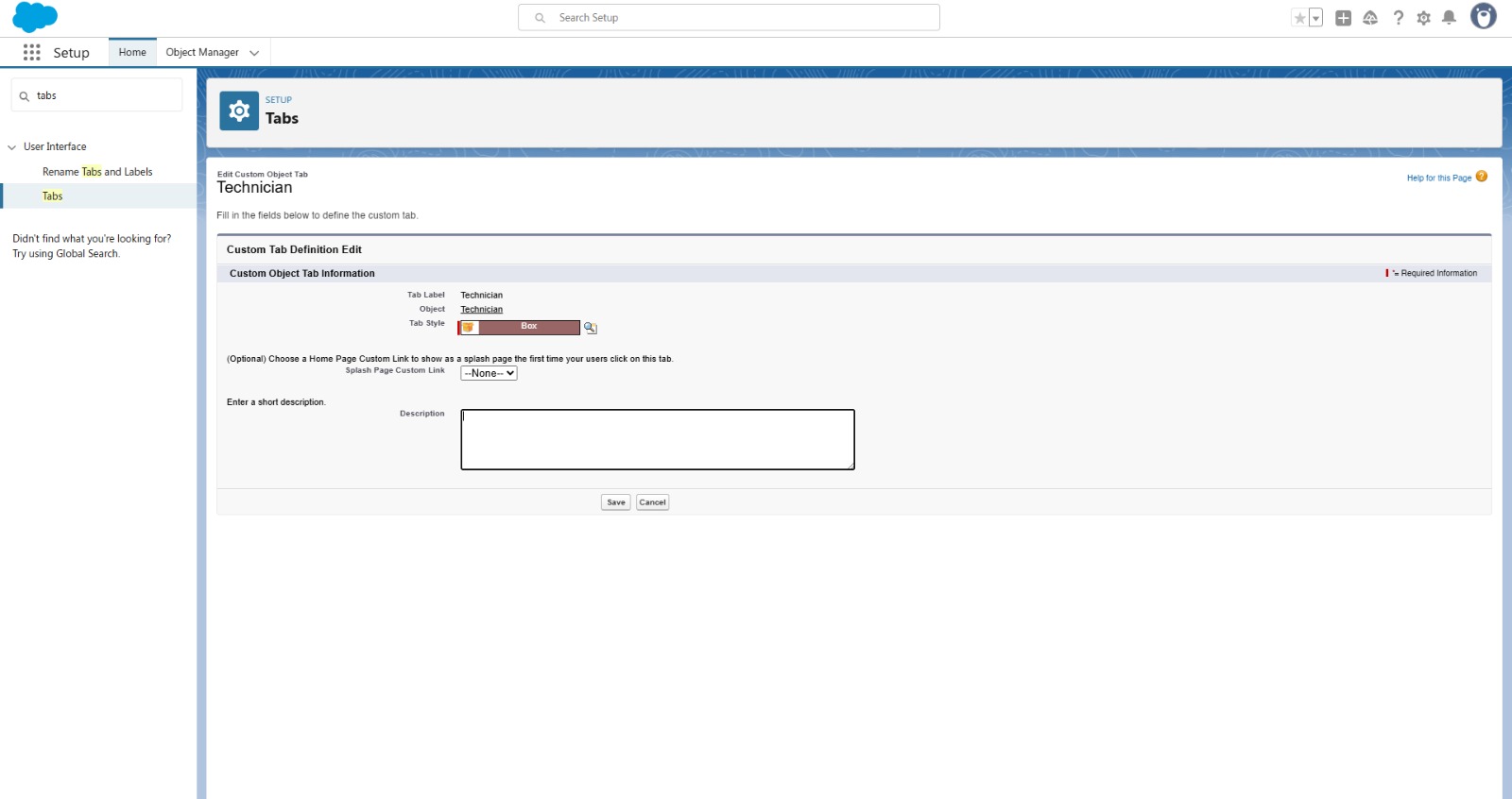
****

* Designed tabs and Lightning App for navigation and UI

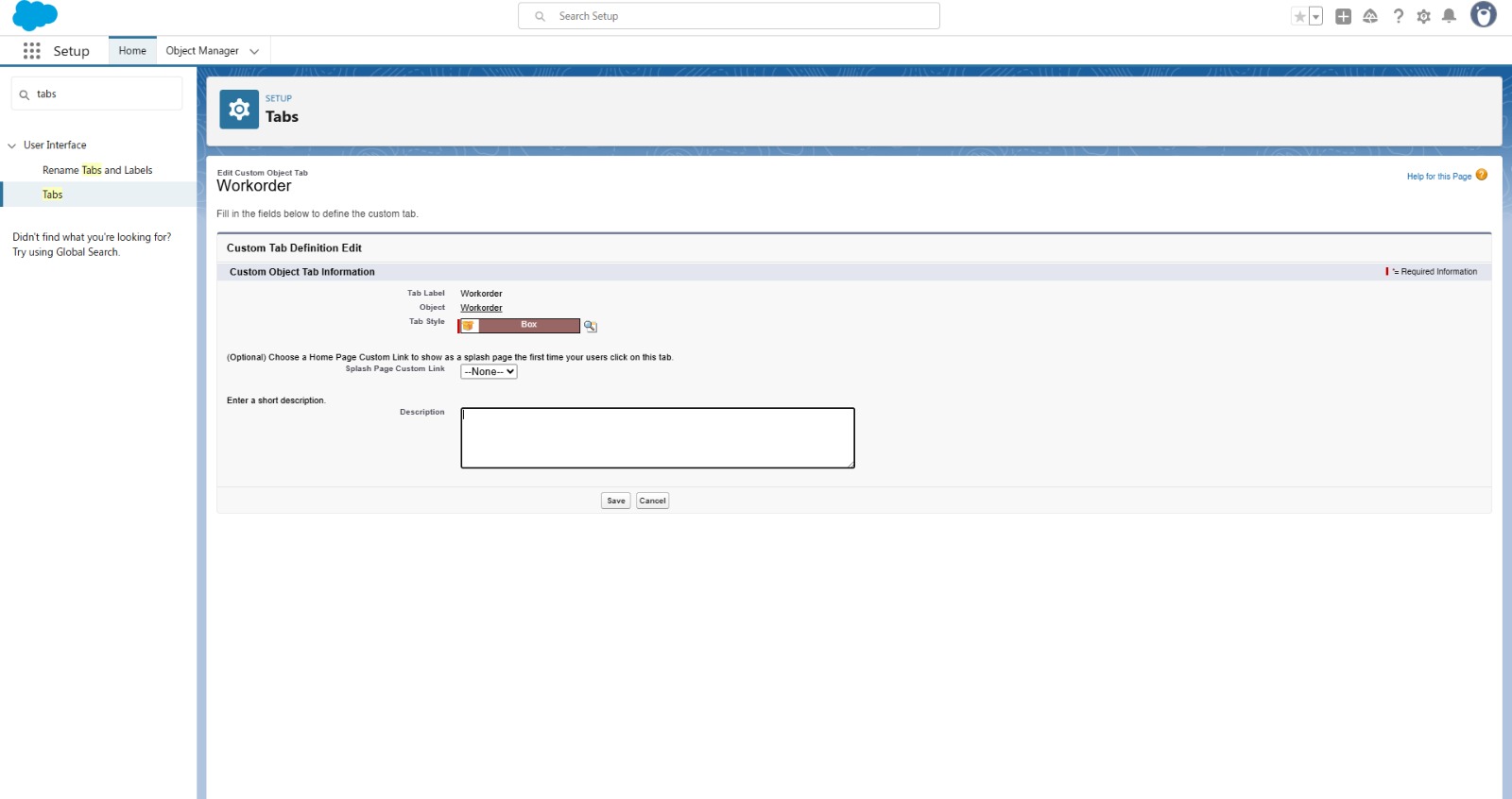
**Assignment Tab:**

****

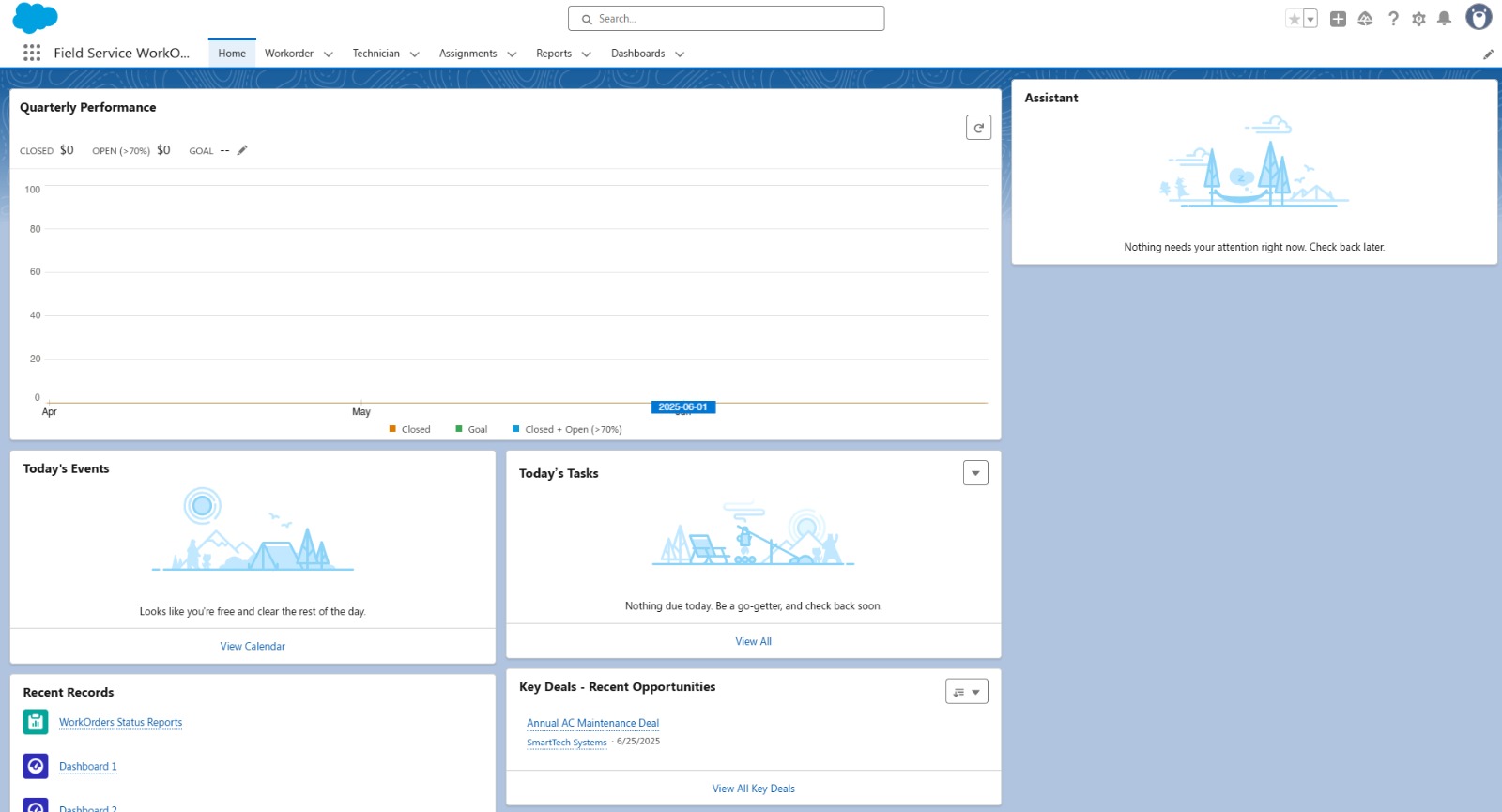
**Technician Tab:**



**Workorder Tab:**

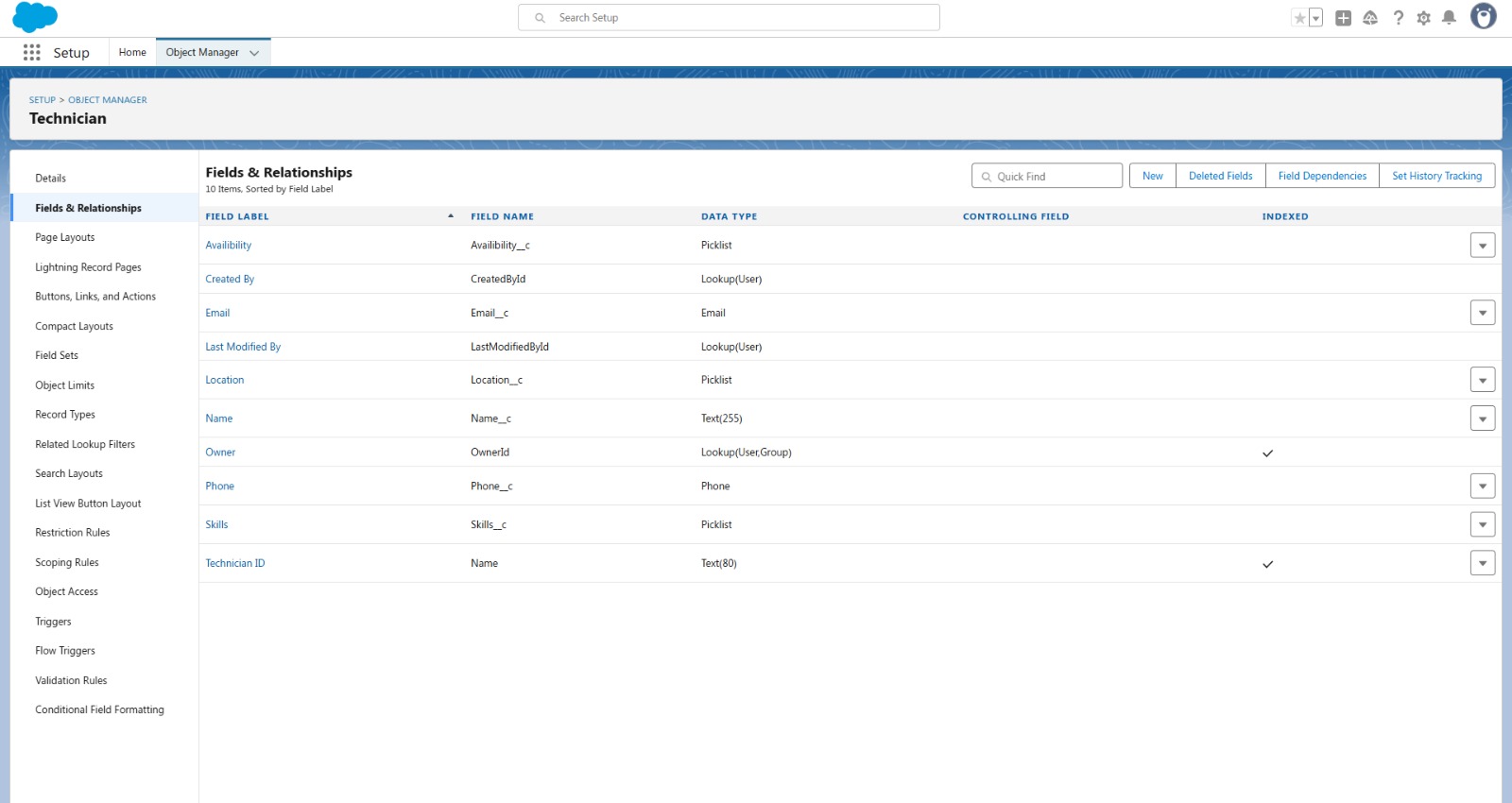
****

**Field Service WorkOrder Lightning App:**

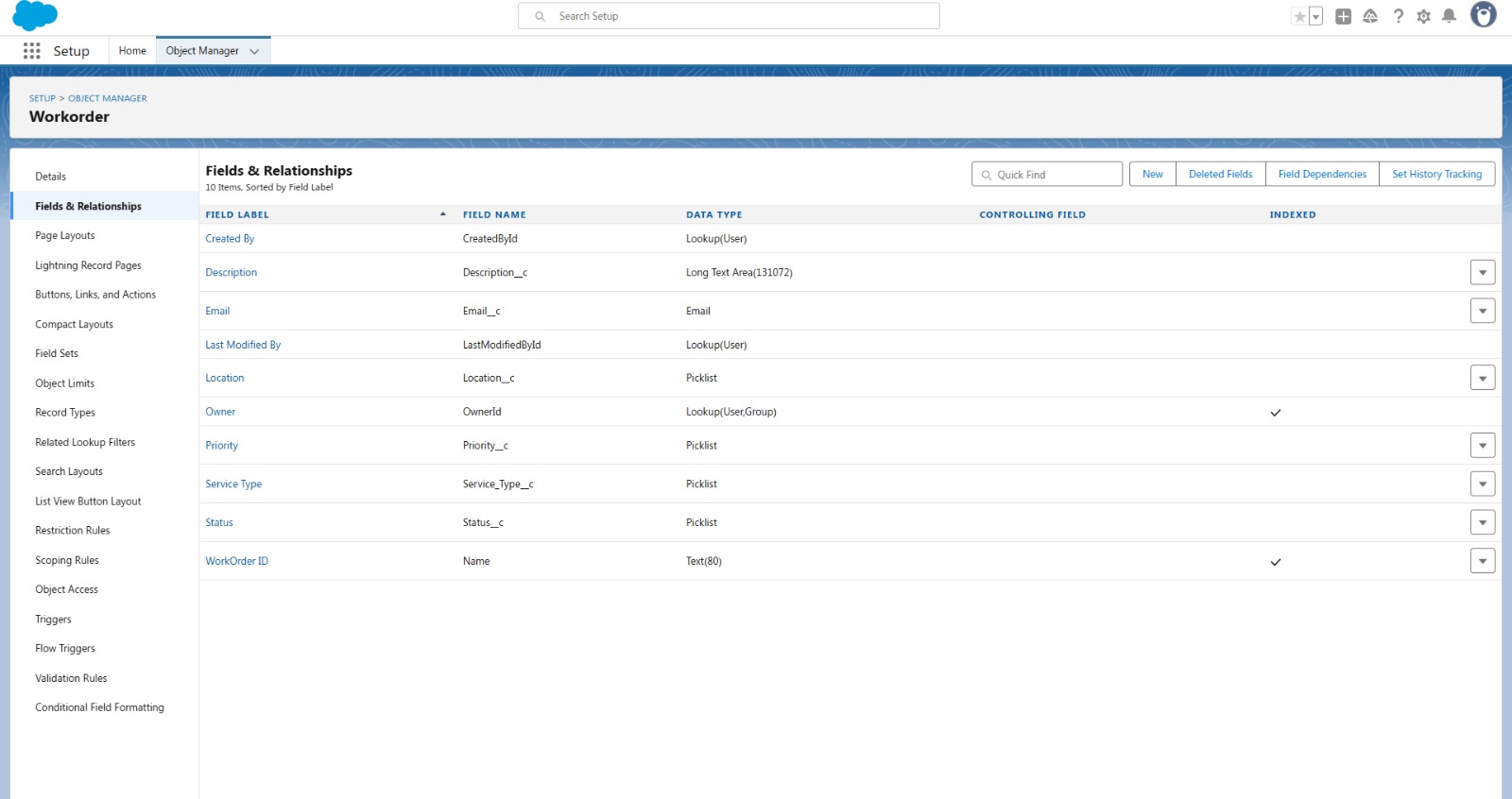
****

* Configured fields and Relationship to ensure data integrity

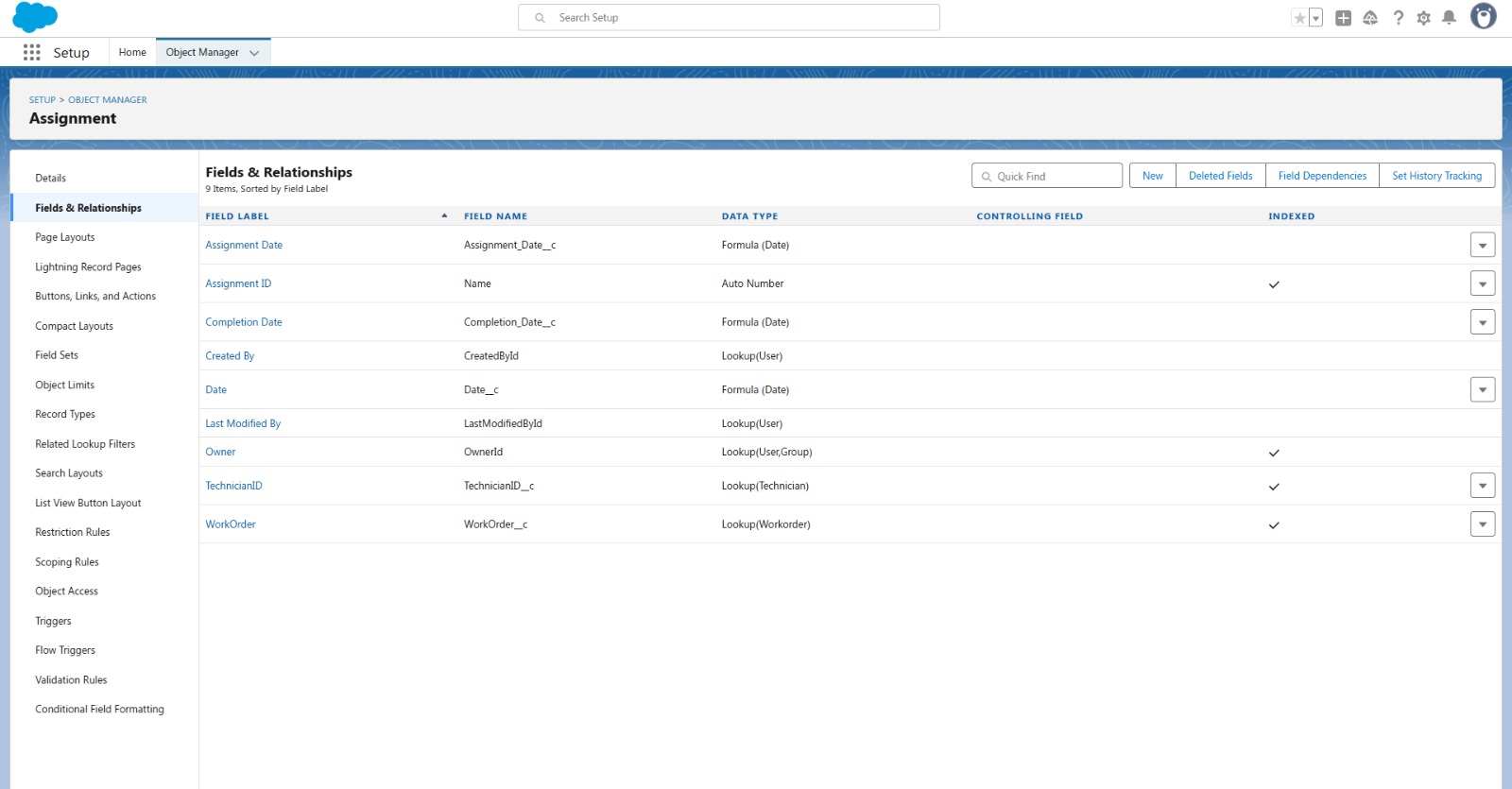
**Fields of Technician object:**

****

**Fields of Workorder Object:**

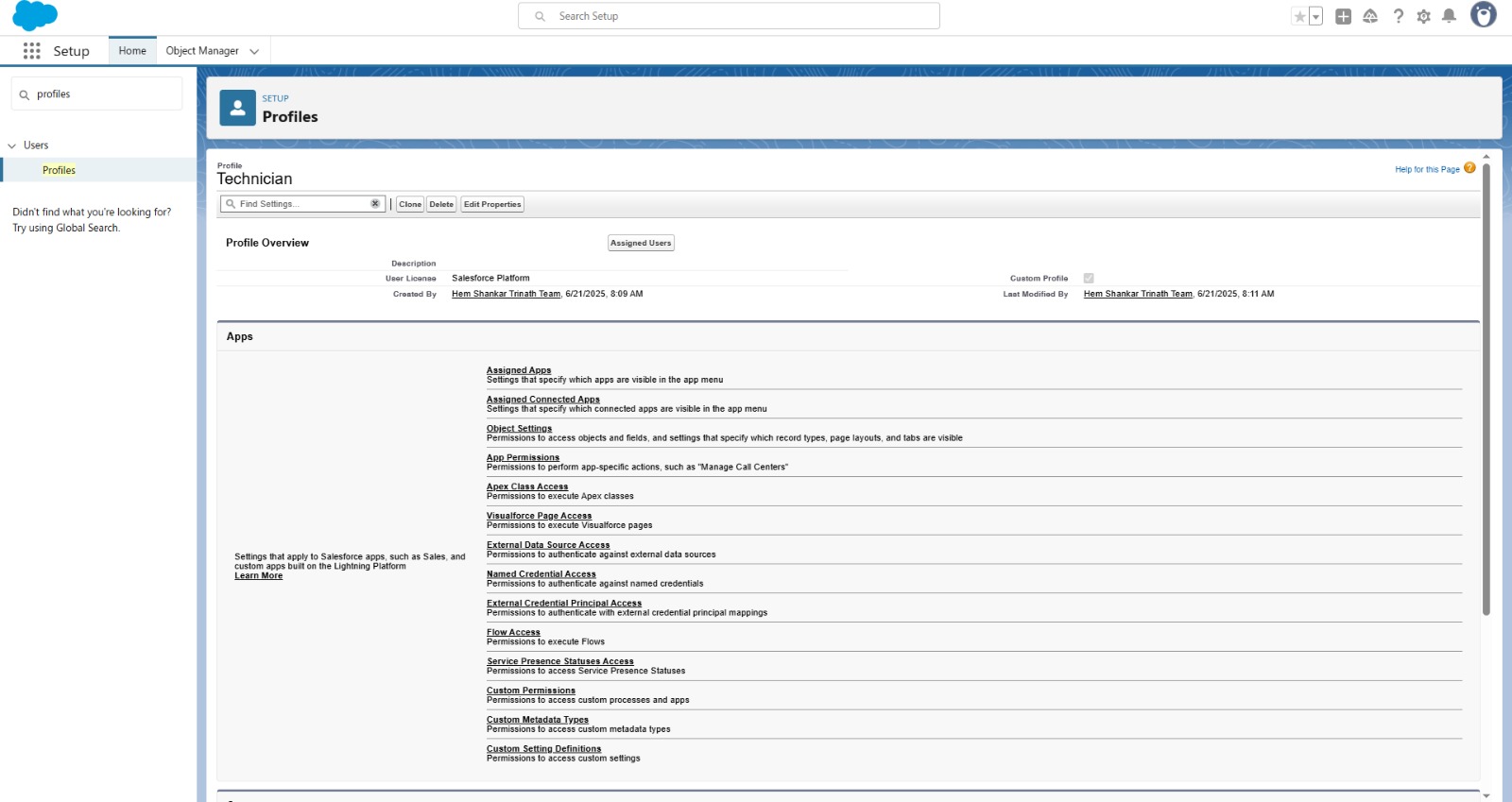
****

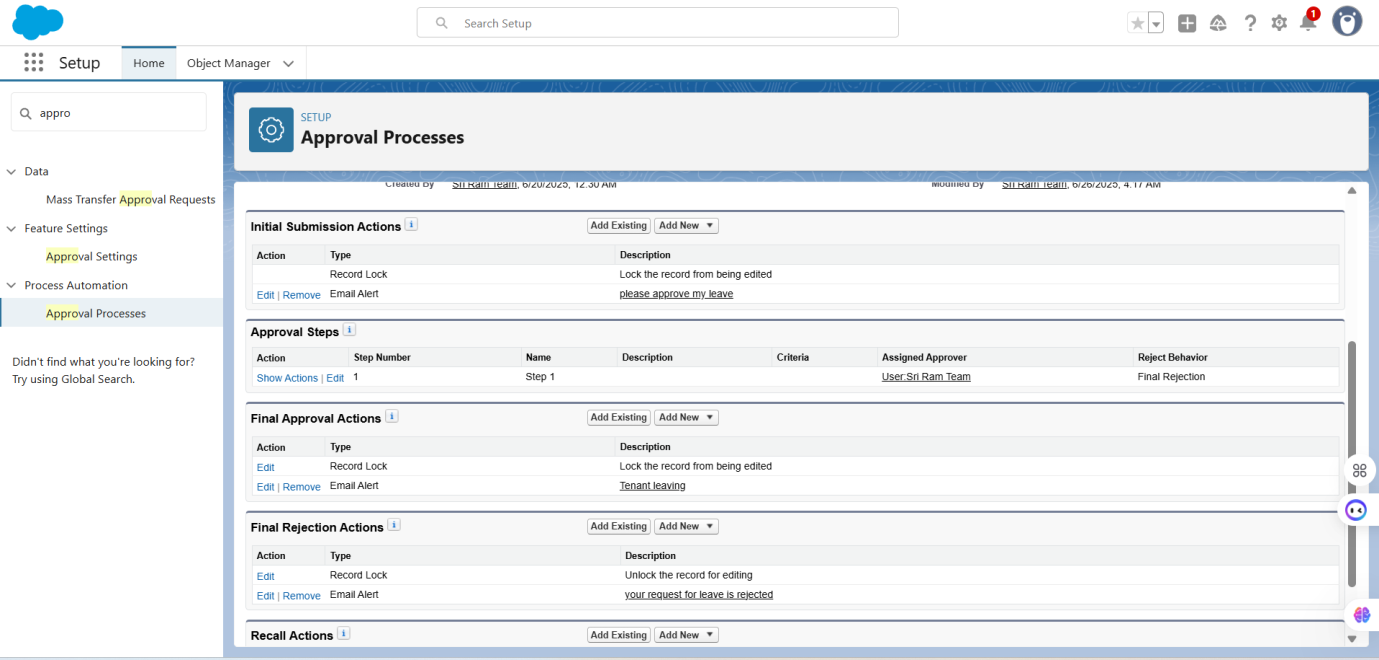
**Fields of Assignment object:**

****

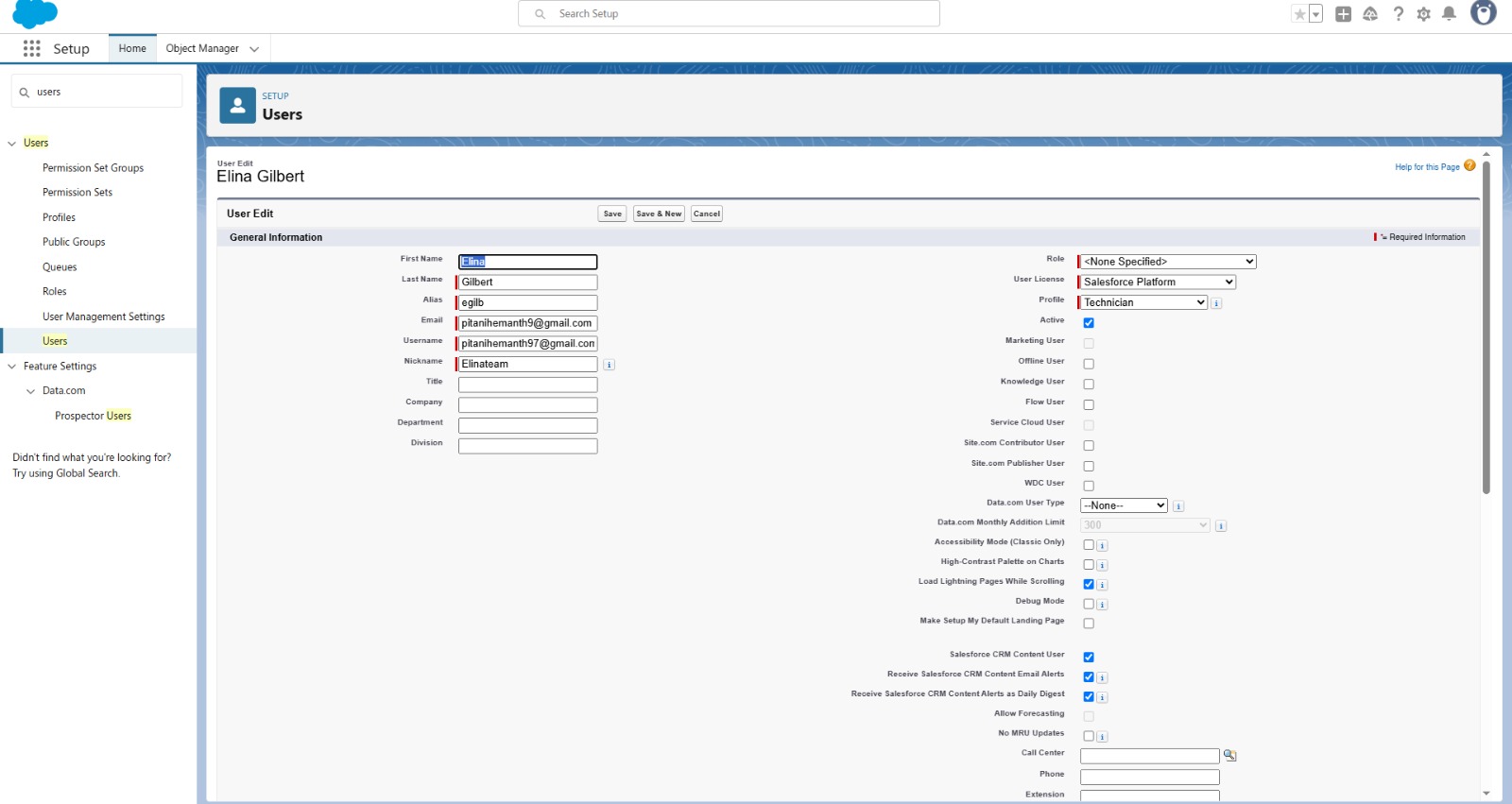
* Created custom profiles with role-based access and added users with appropriate permissions to manage WorkOrders and Assignments securely.

**Technician profile:**



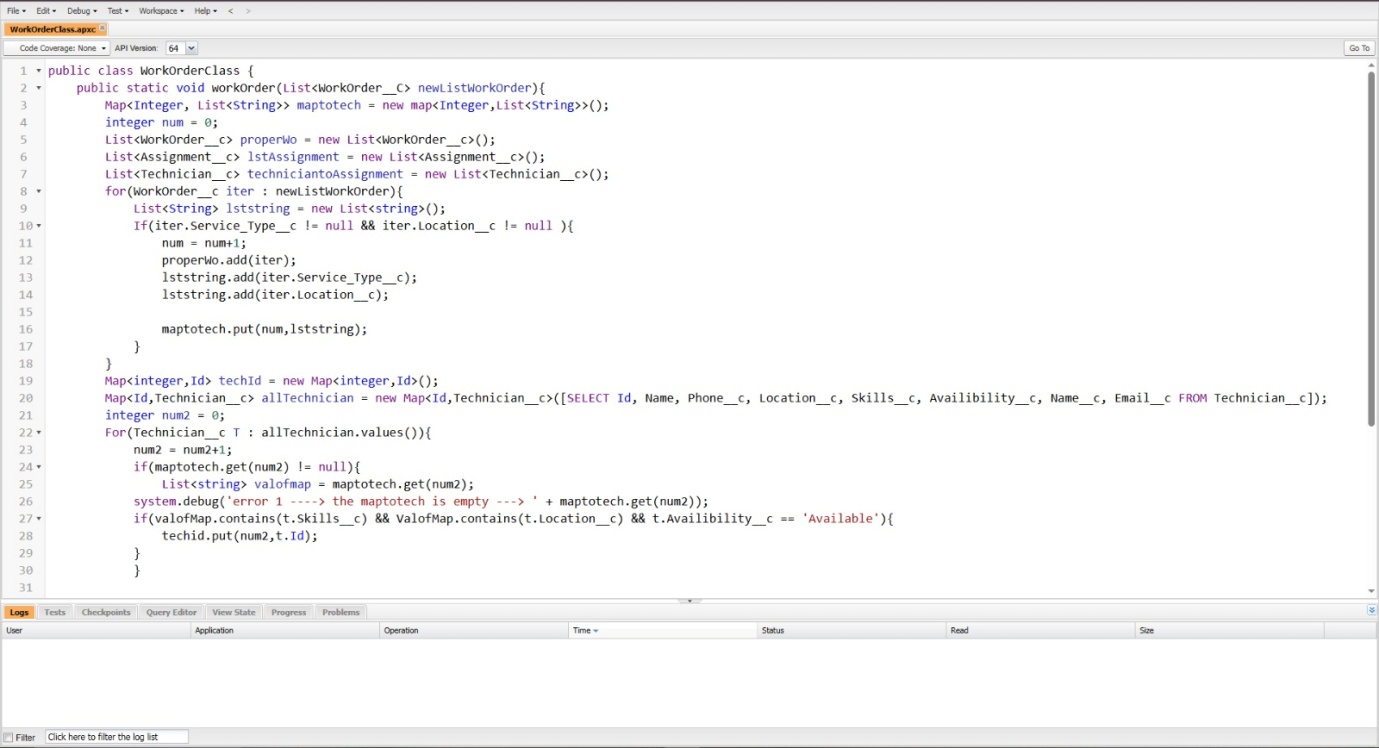


**User:**

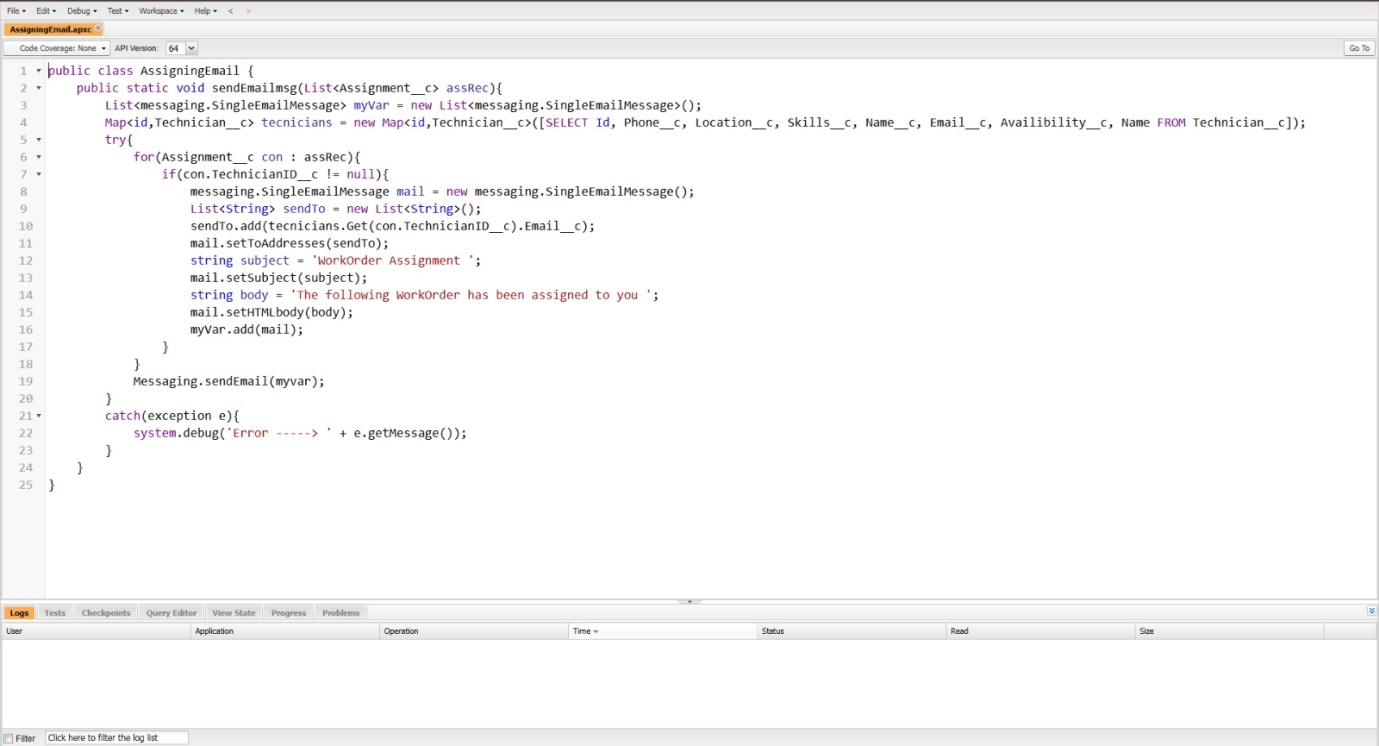


* Developed multiple Apex Classes to automate WorkOrder assignment (WorkOrderClass), send technician notifications (AssigningEmail, CompletionMail), and manage record cleanup through batch and scheduled jobs (RecordDeletions, ScheduleClass).

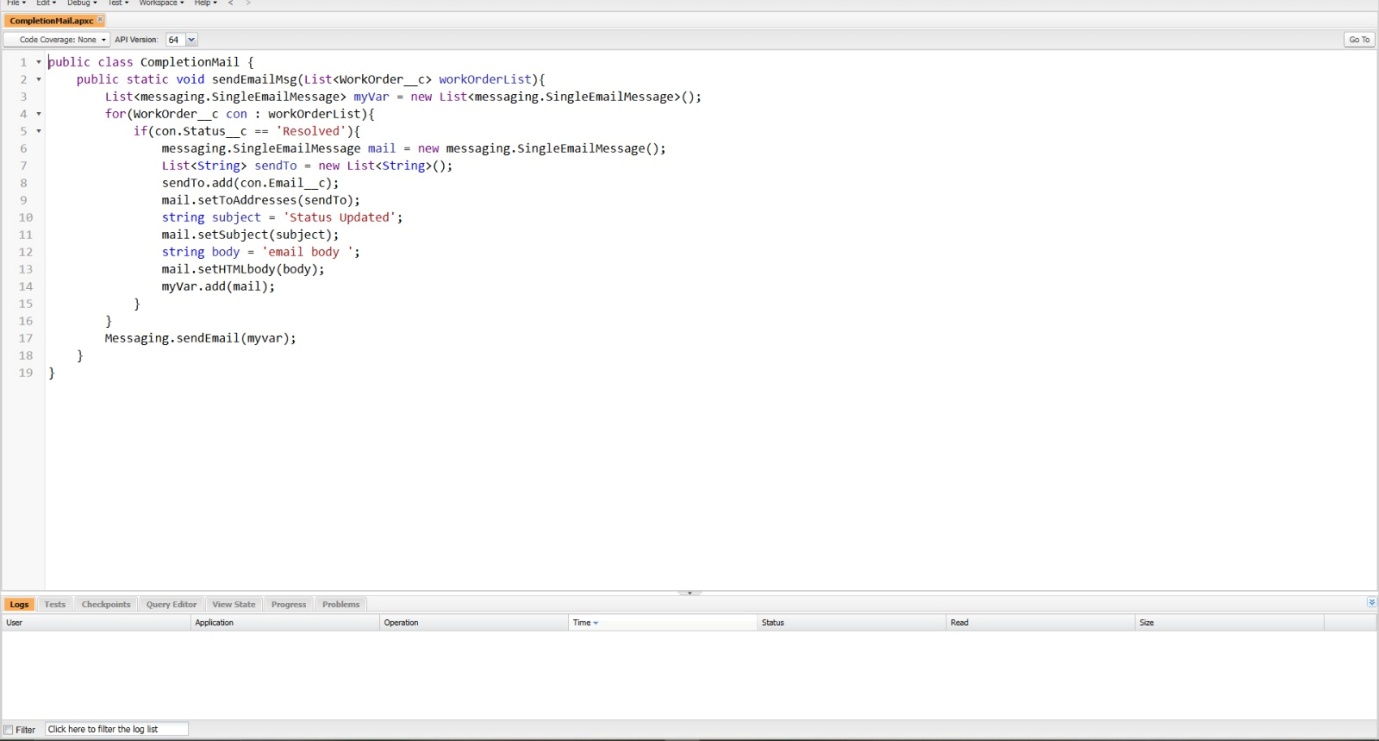
WorkOrderClass:



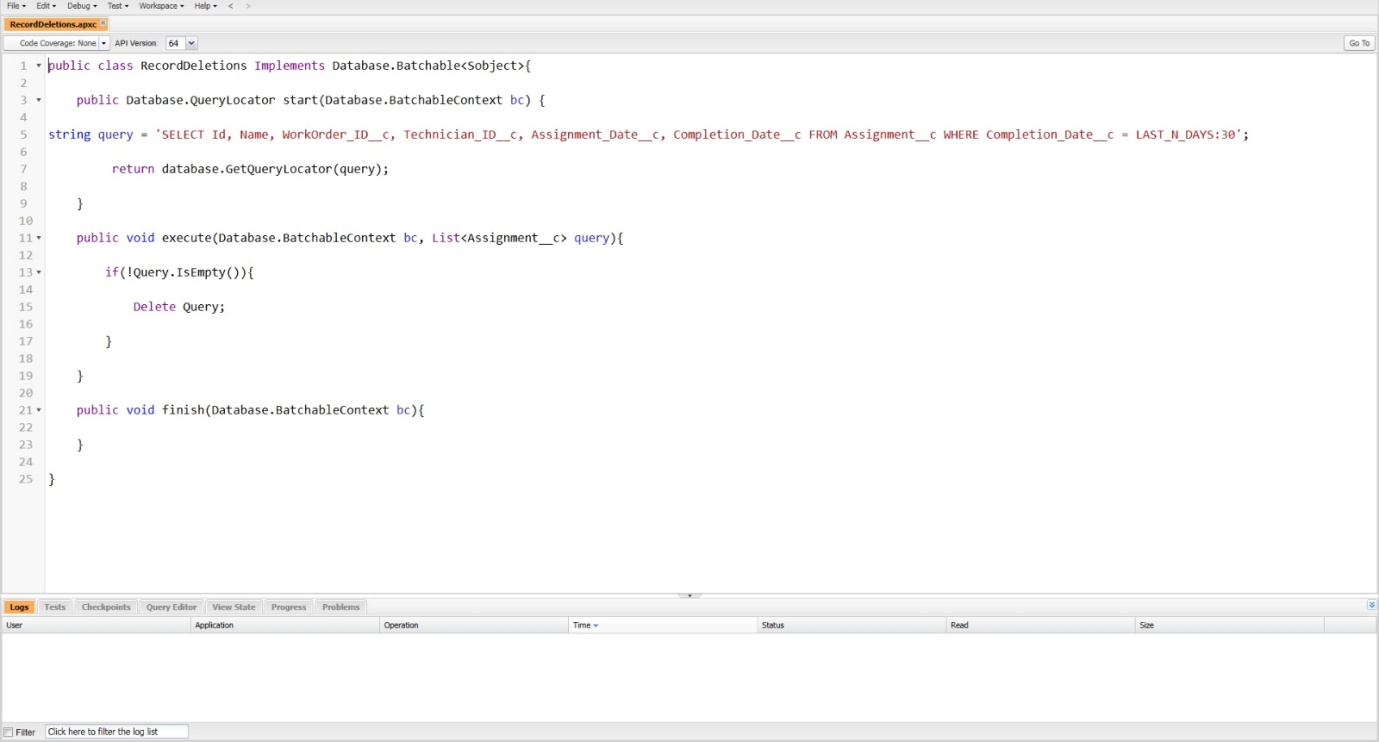
AssigningEmail:



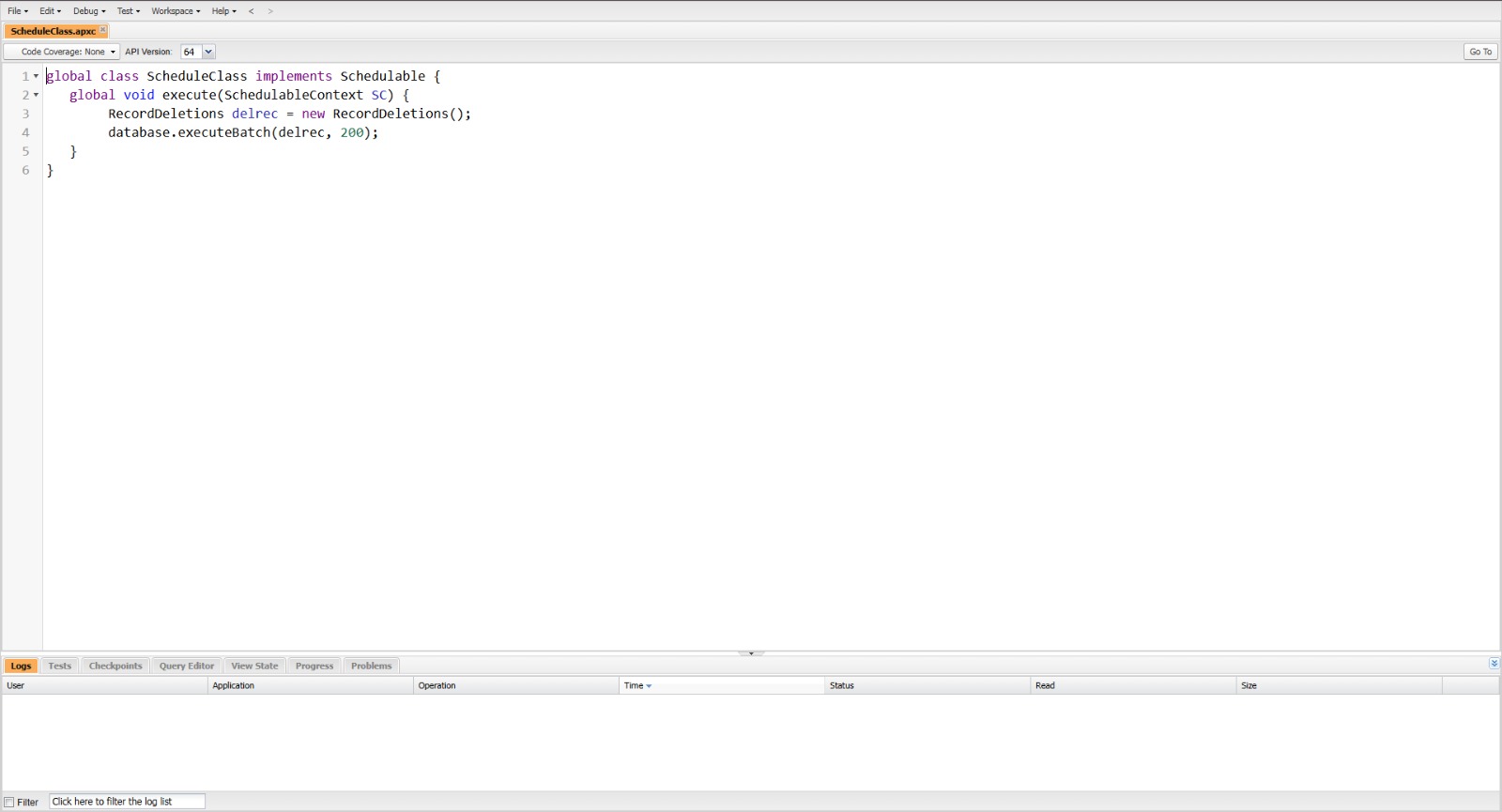
Completion mail:



RecordDeletions:

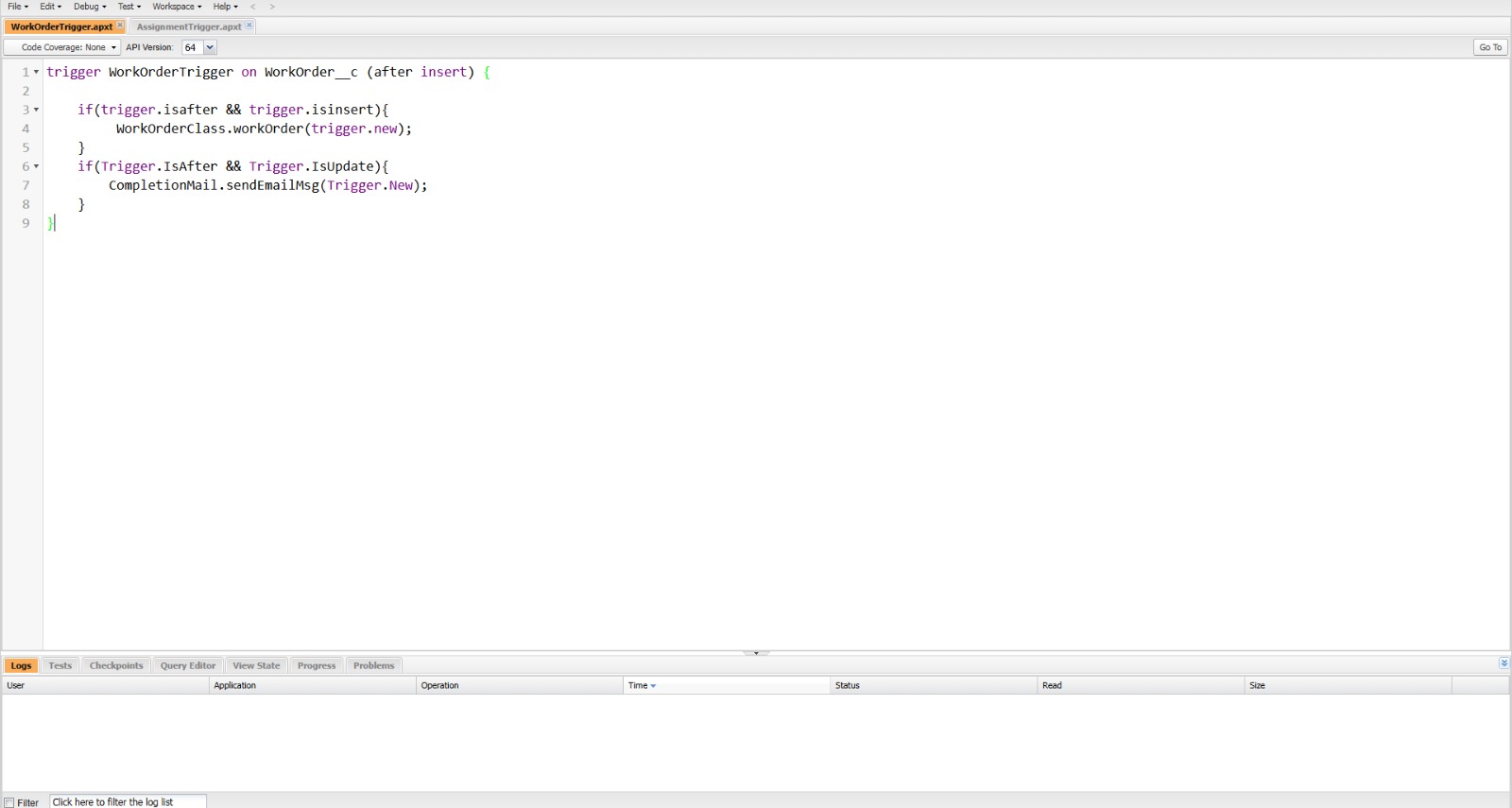


Schedule class:



* Implemented Apex Trigger to automatically assign WorkOrders to available Technicians based on matching location and skill criteria.

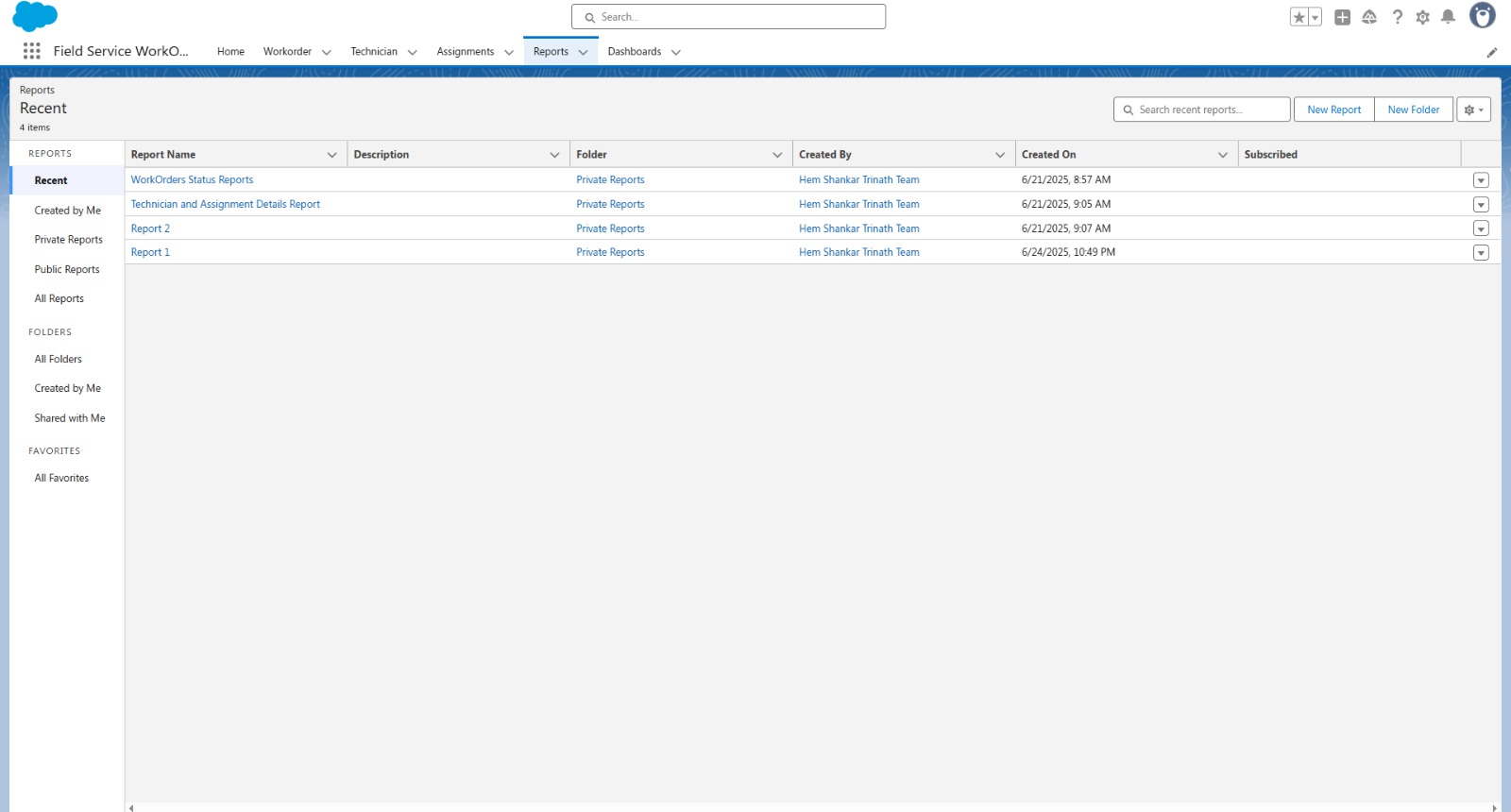
Workorder trigger:



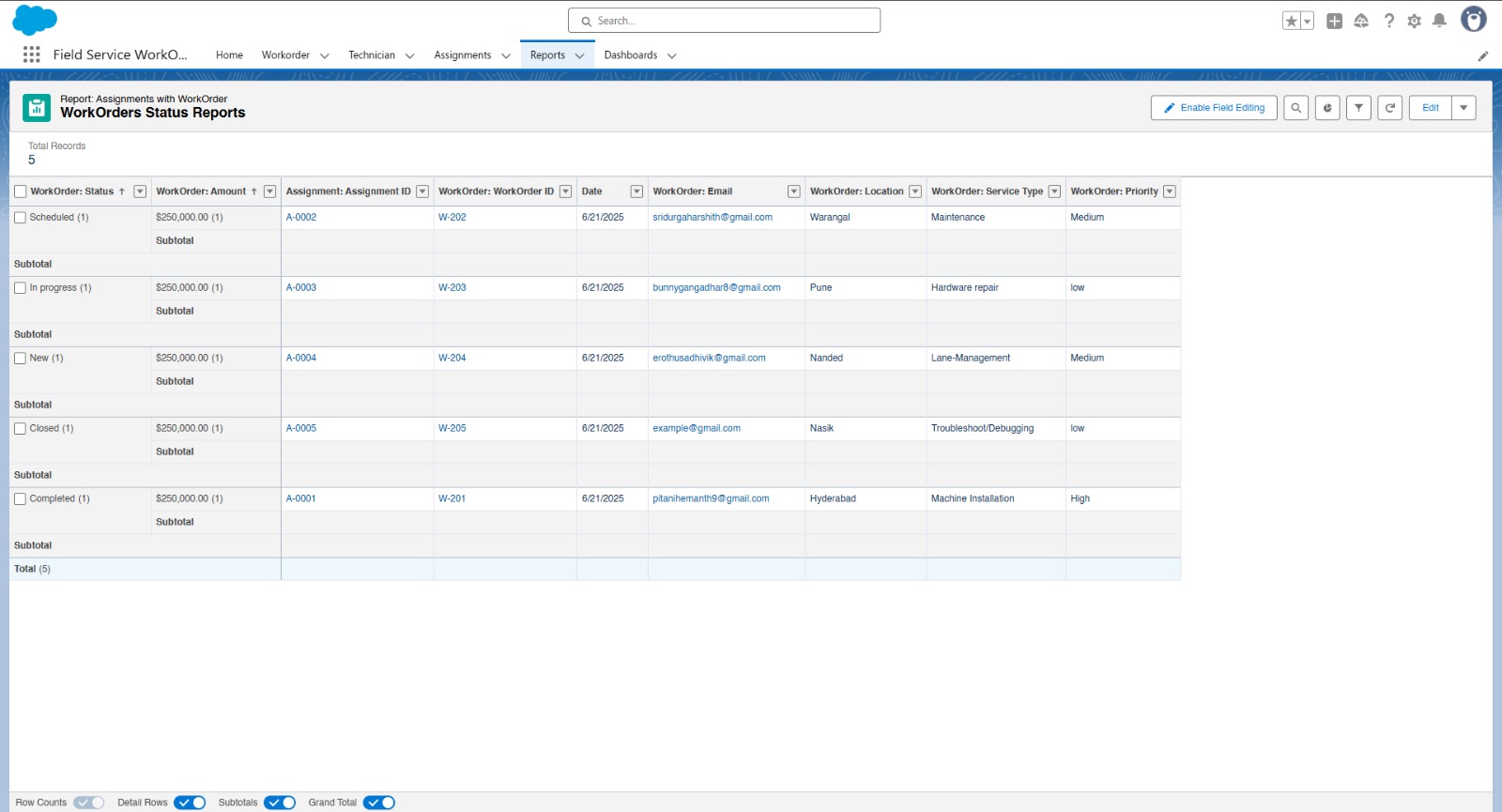
Assignment trigger:



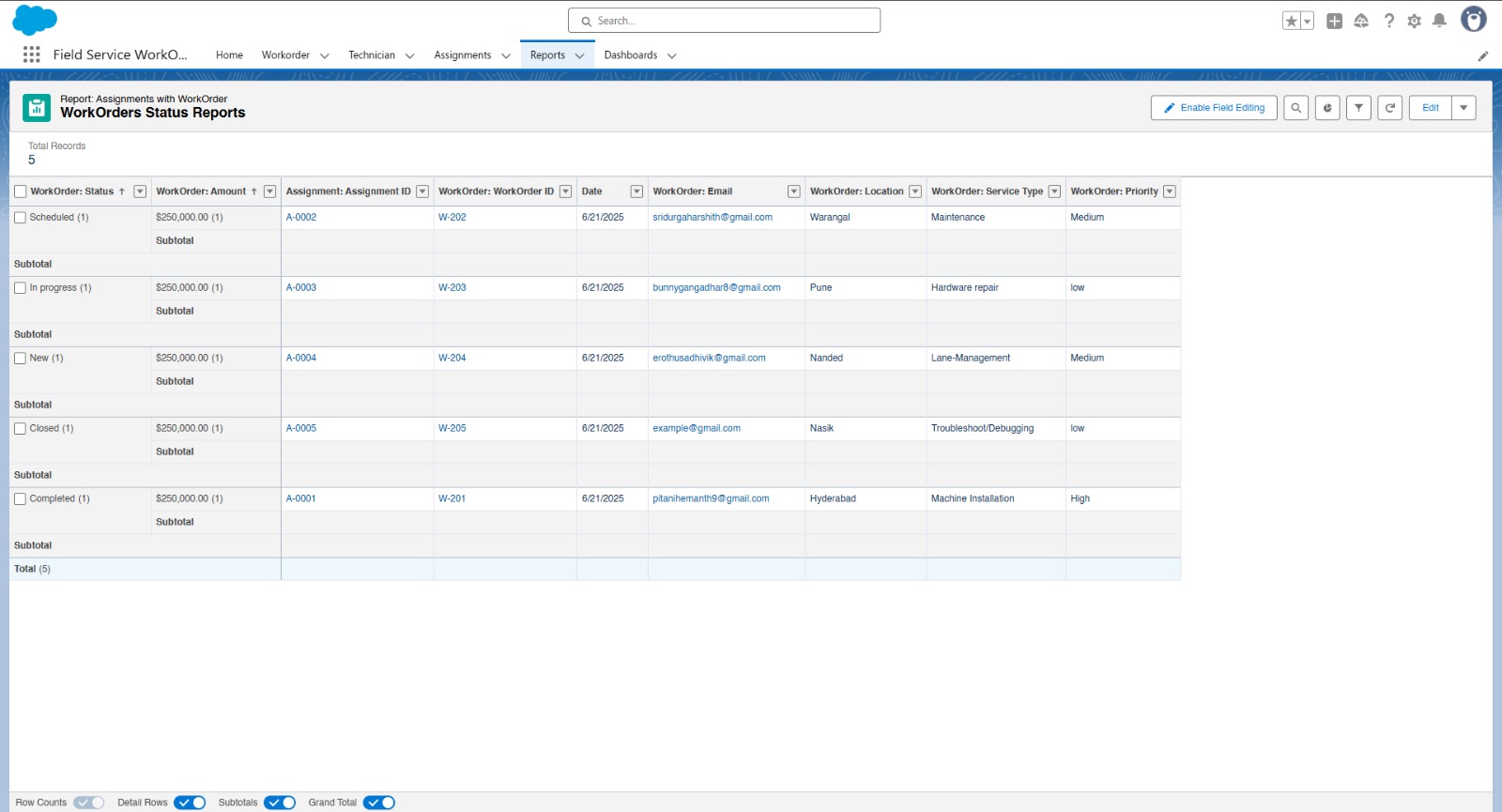
* Created Reports and Dashboards to monitor WorkOrder statuses, technician assignments, and service performance, enabling real-time visibility and data-driven decision-making.



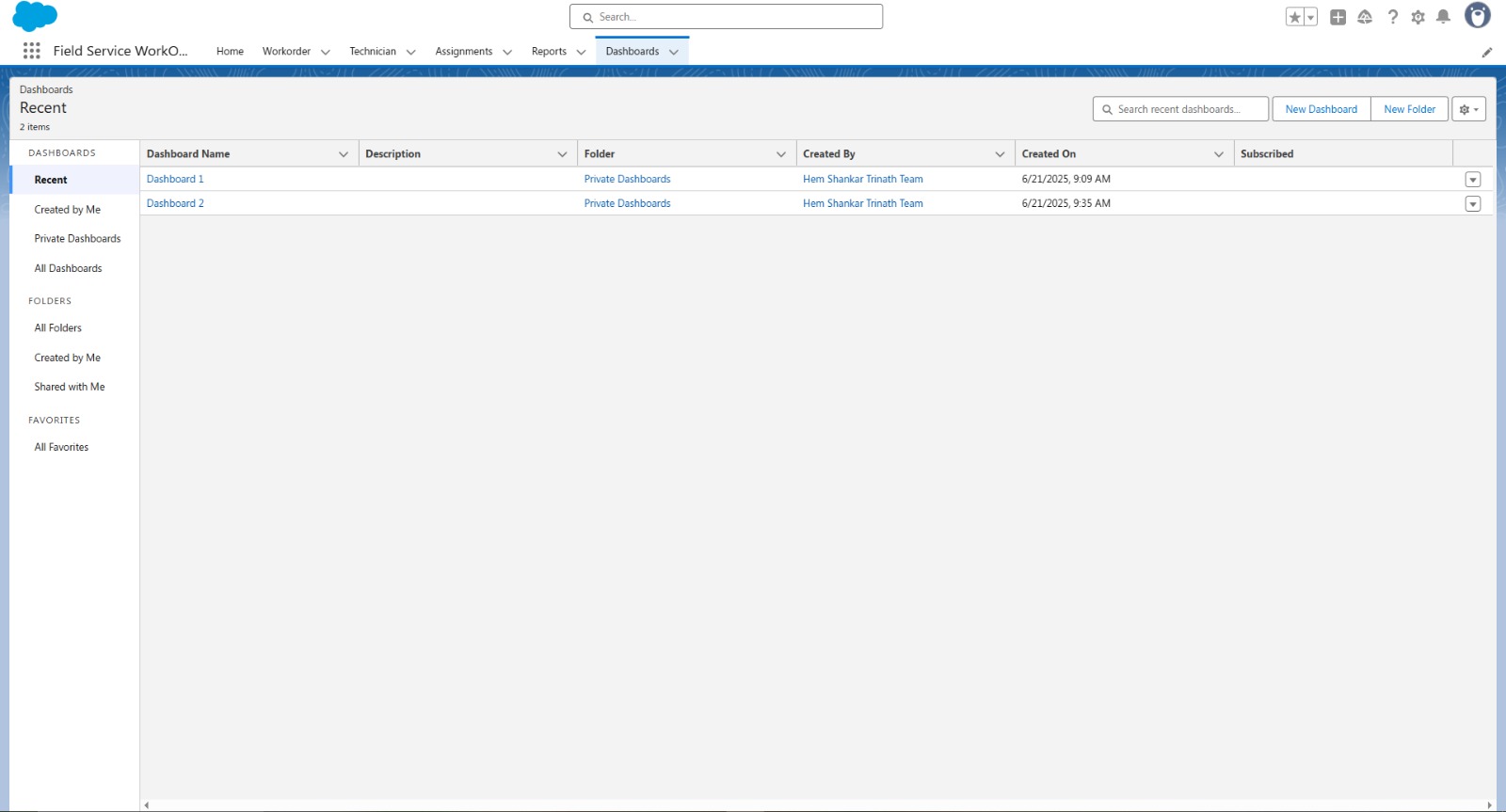
WorkOrders Status Reports:



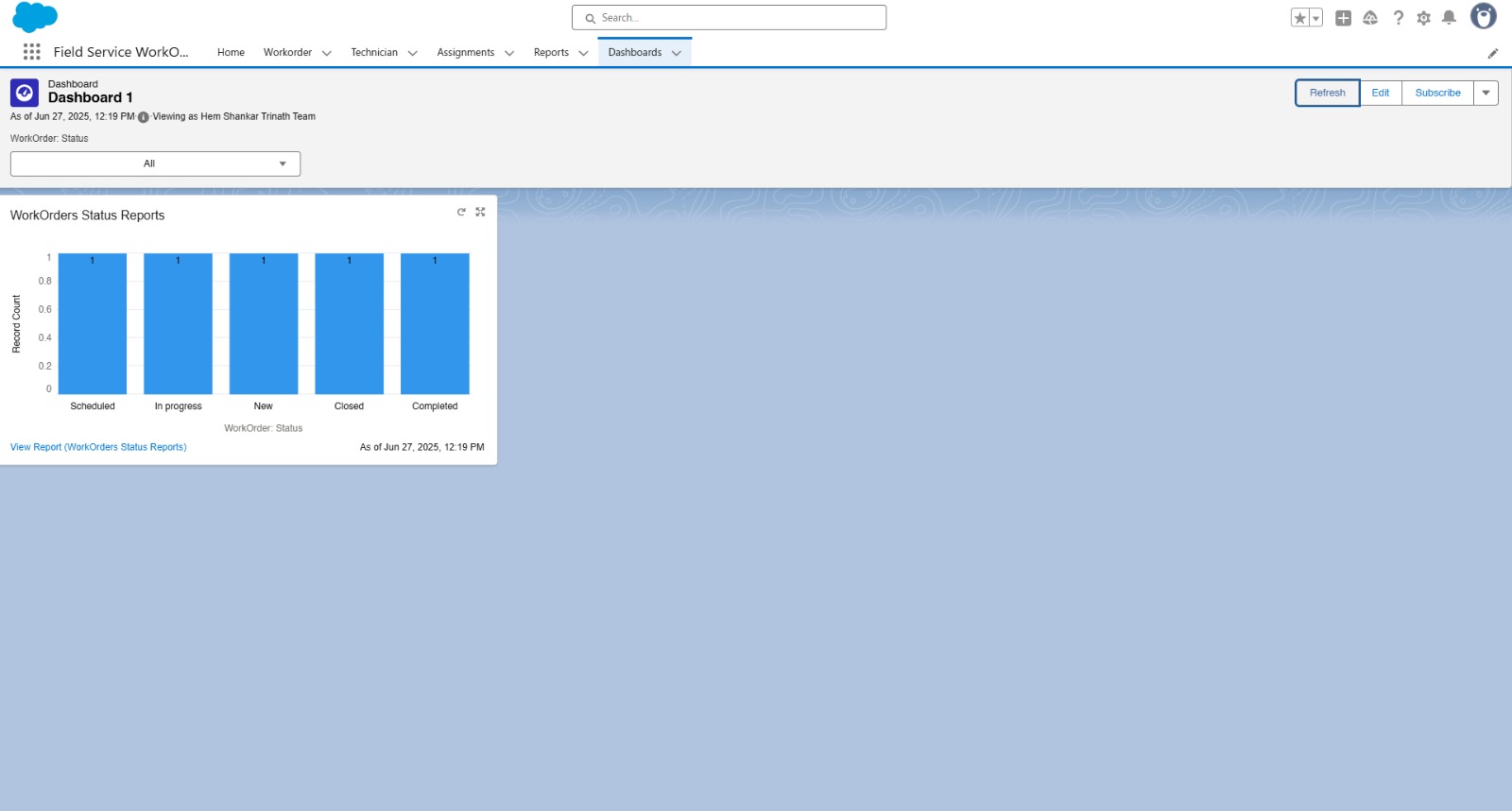
Technician and Assignment Details Reports:



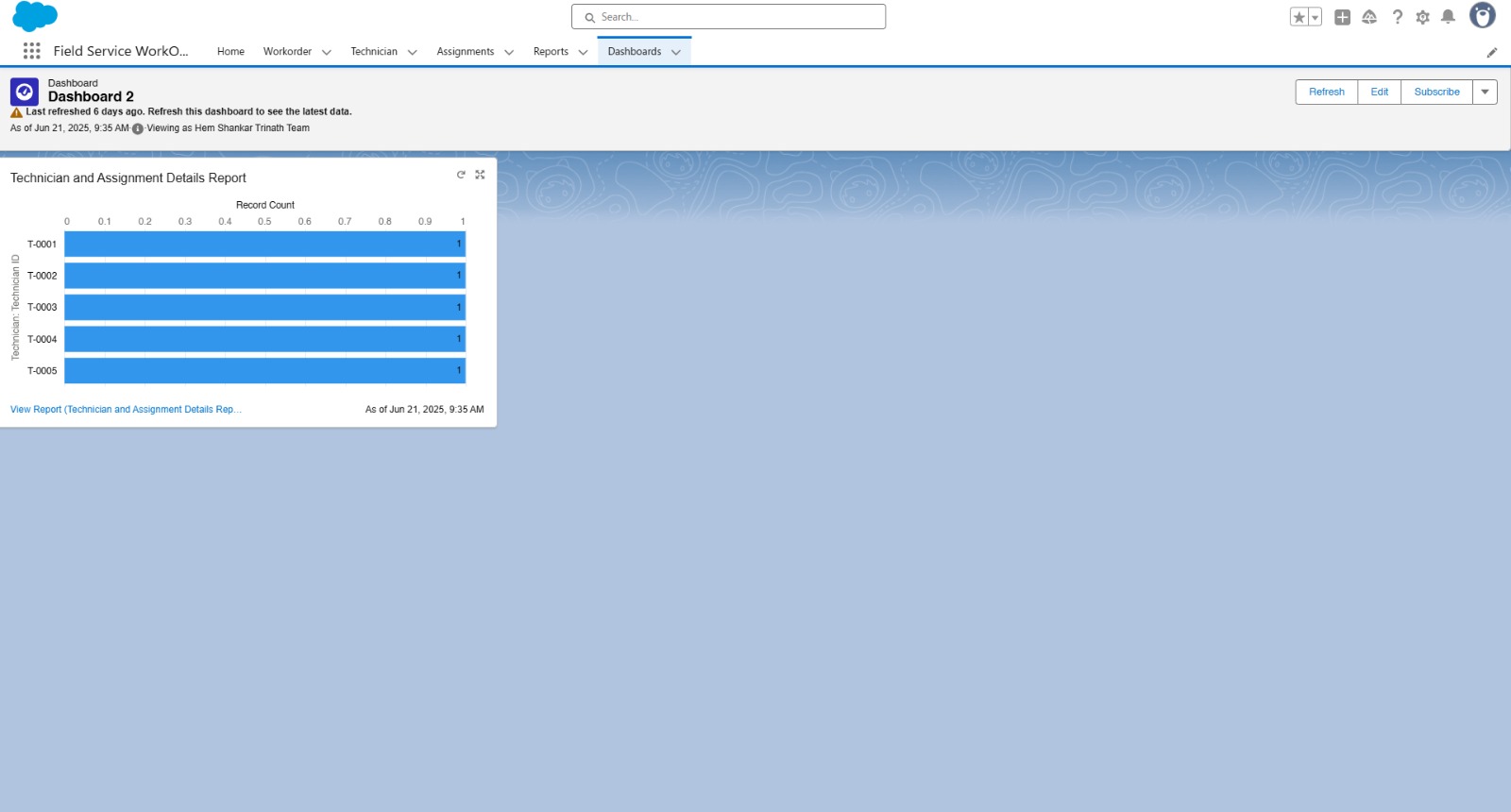
Dashboards



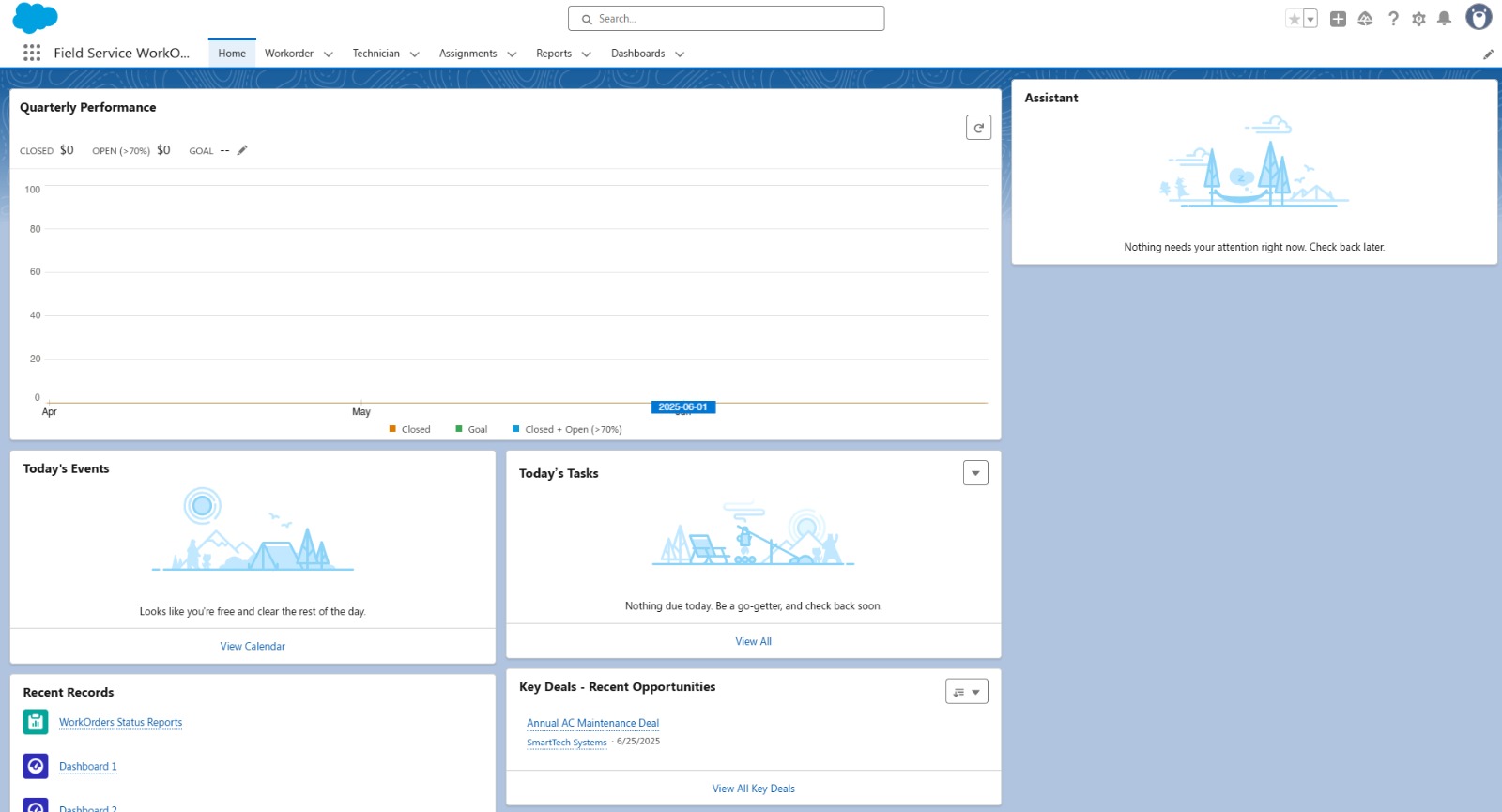
Dashboard 1:



Dashboard 2:



● Designed a user-friendly interface using Lightning App, custom tabs, and page layouts to provide seamless access to WorkOrders, Technicians, and Assignments.



**6.2 Integration and Testing**

### **Integration Activities**

Once all custom objects (Technician, WorkOrder, and Assignment) and automation logic were developed, integration was carried out by linking them through **Lookup Relationships** to ensure seamless data flow. Each **Assignment** was linked to a **WorkOrder** and a **Technician**, enabling automatic allocation and tracking.

Apex Triggers and Classes were integrated to handle real-time actions like technician assignment and completion notifications. Email templates were connected to workflow actions to alert technicians when work orders were assigned or completed. Scheduled Apex and batch jobs were also integrated to perform periodic cleanup of resolved work orders after 30 days.

The entire automation flow—**Triggers, Flows, Email Alerts, and Scheduled Apex**—was reviewed and tested to ensure they worked cohesively and reflected actual field service processes.

### **Testing Approach**

Testing was conducted in the **Salesforce Sandbox** environment to avoid affecting live data. A structured approach was followed:

* **Unit Testing**: Each component—custom objects, validation rules, flows, and Apex classes—was tested individually to confirm expected behavior.
* **Integration Testing**: Object interactions and automated processes were tested together to ensure correct data flow. For example, creating a WorkOrder triggered technician assignment and email notification as expected.
* **Functional Testing**: Core functions such as WorkOrder creation, status updates, technician assignment, and completion tracking were tested using real-world scenarios.
* **System Testing**: Full end-to-end testing was done to verify the overall workflow, field access control, and system behavior under normal conditions.
* **Performance Testing**: Key operations such as email sending, dashboard refresh, and batch deletions were tested for speed and reliability.
* **User Acceptance Testing (UAT)**: Admin and Technician users reviewed the system interface and functionality. Their feedback was used to fine-tune layouts and improve usability.

### **Outcome**

The integration and testing phase ensured that the Field Service Work Order Optimization solution was robust, efficient, and deployment-ready. All automation components worked together smoothly, data consistency was maintained, and the user interface met expectations. The final system delivered accurate work order assignment, real-time technician alerts, and streamlined performance, in line with project objectives.

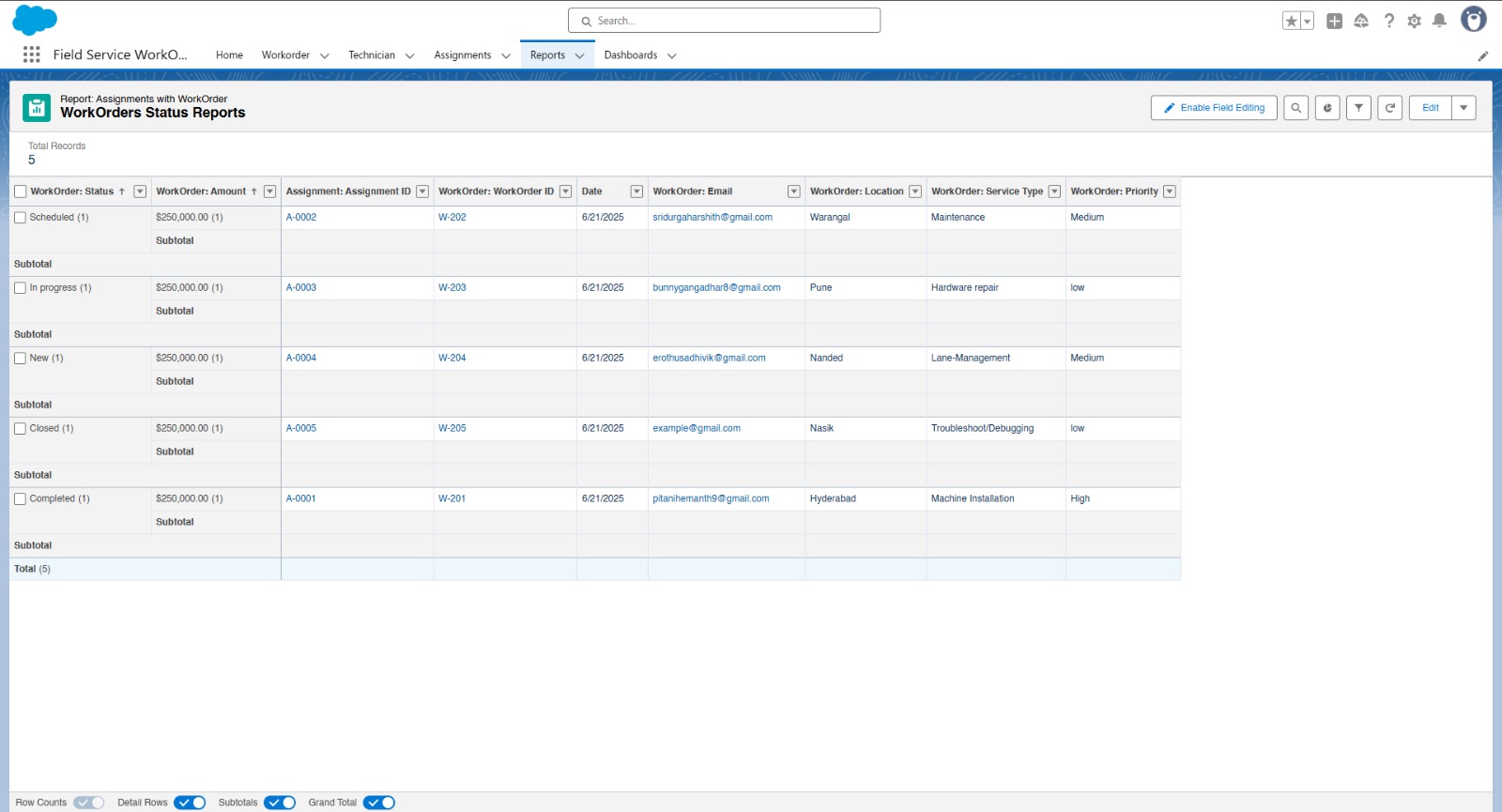
### **Key Highlights**

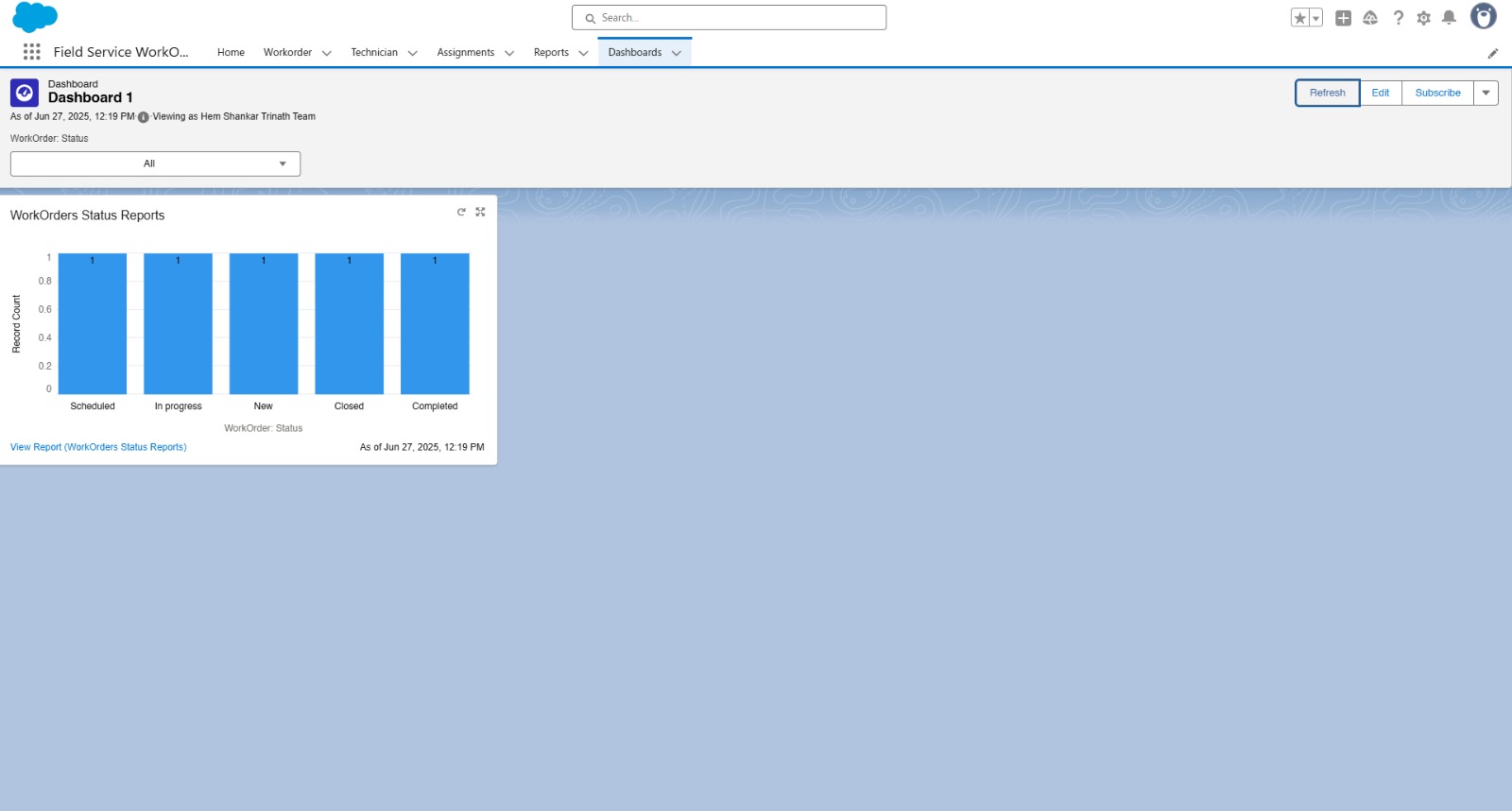
* Integrated all custom objects (Technician, WorkOrder, Assignment) using lookup relationships
* Connected email templates with triggers for technician alerts
* Conducted manual and automated testing to validate all core features and flows

**7. FUNCTIONAL AND PERFORMANCE TESTING**

**7.1 Performance Testing**

* **WorkOrder creation** and **technician auto-assignment** via Apex Trigger
* **Email alerts** sent upon WorkOrder assignment and completion
* **Scheduled Apex batch job** deleting resolved assignments older than 30 days
* Loading of **real-time dashboards** and **custom reports**





### **Testing Focus Areas**

Performance testing was focused on validating key business operations most commonly used by end users. These included:

* **WorkOrder creation** and automatic technician assignment
* Execution of **Flows and Apex Triggers** (e.g., assignment logic, completion alerts)
* **Scheduled Apex jobs** for deleting old assignments (e.g., after 30 days)
* Loading and refreshing **dashboards and custom reports** with real-time metrics

### **Approach and Tools Used**

Testing was conducted in the **Salesforce Sandbox** using test records designed to mimic real-world usage. Manual test cases were executed and monitored using:

* **Debug Logs**
* **Developer Console**
* **System Overview Tools** (to monitor governor limits, heap size, etc.)

Test scenarios simulated high-usage patterns, such as:

* Bulk creation of WorkOrders
* Trigger-based mass assignment and email alerts
* Running scheduled jobs on large datasets
* Dashboard refreshes with dynamic filters applied

**Key performance indicators monitored:**

* Page load and save times for records
* Execution duration of Flows and Apex logic
* Completion time of scheduled batch jobs
* Email delivery time after trigger events
* Report and dashboard loading latency

### **Results and Observations**

The system handled standard and moderate workload scenarios efficiently:

* WorkOrders were created and assigned in real time
* Apex Triggers and Flows executed with no visible delays
* Email alerts were sent to Technicians promptly upon assignment
* Scheduled Apex jobs completed on schedule and processed data as expected
* Dashboards and reports loaded within seconds, even with applied filters

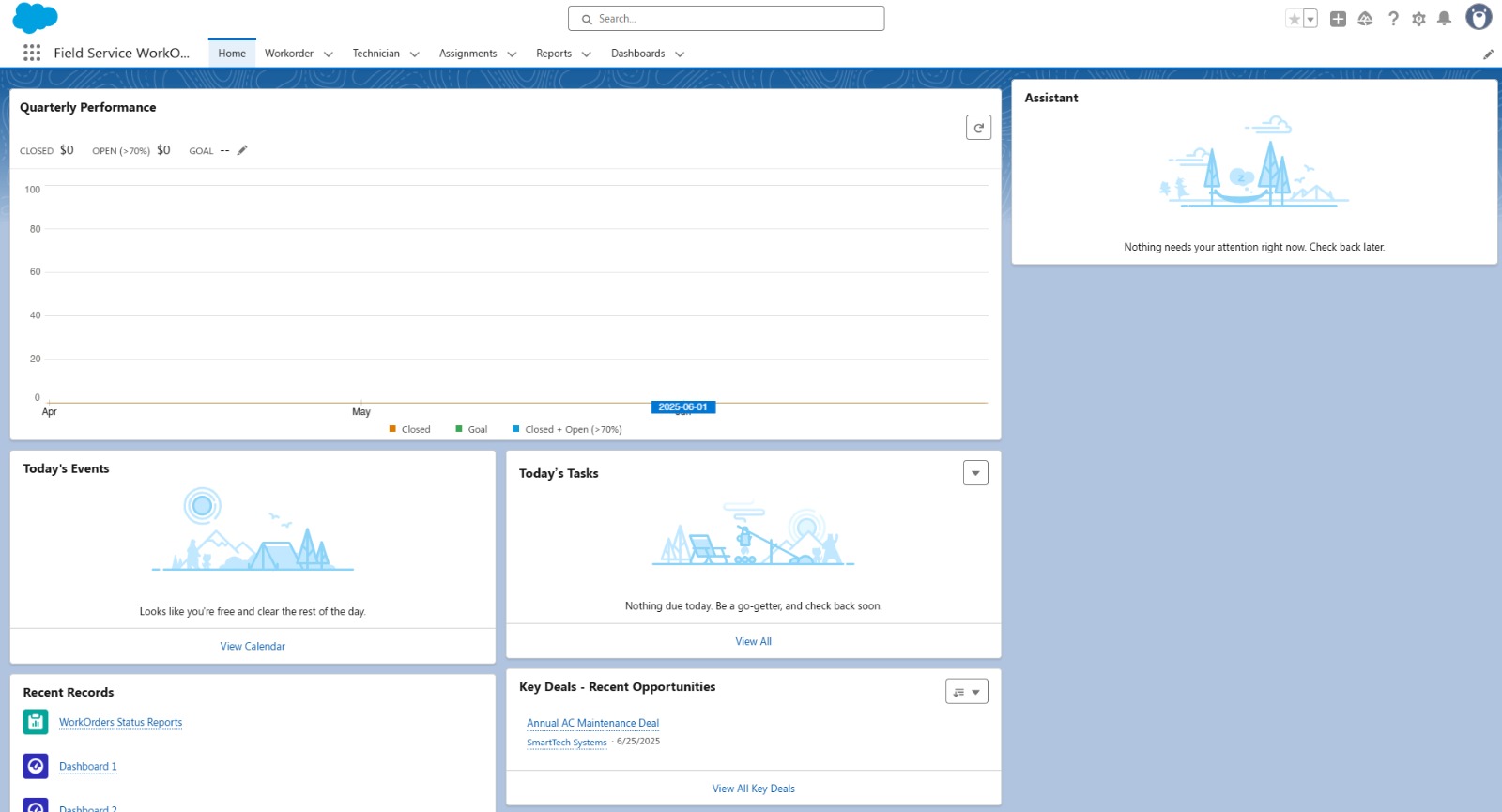
Minor delays were observed during **bulk WorkOrder imports**, which were resolved by:

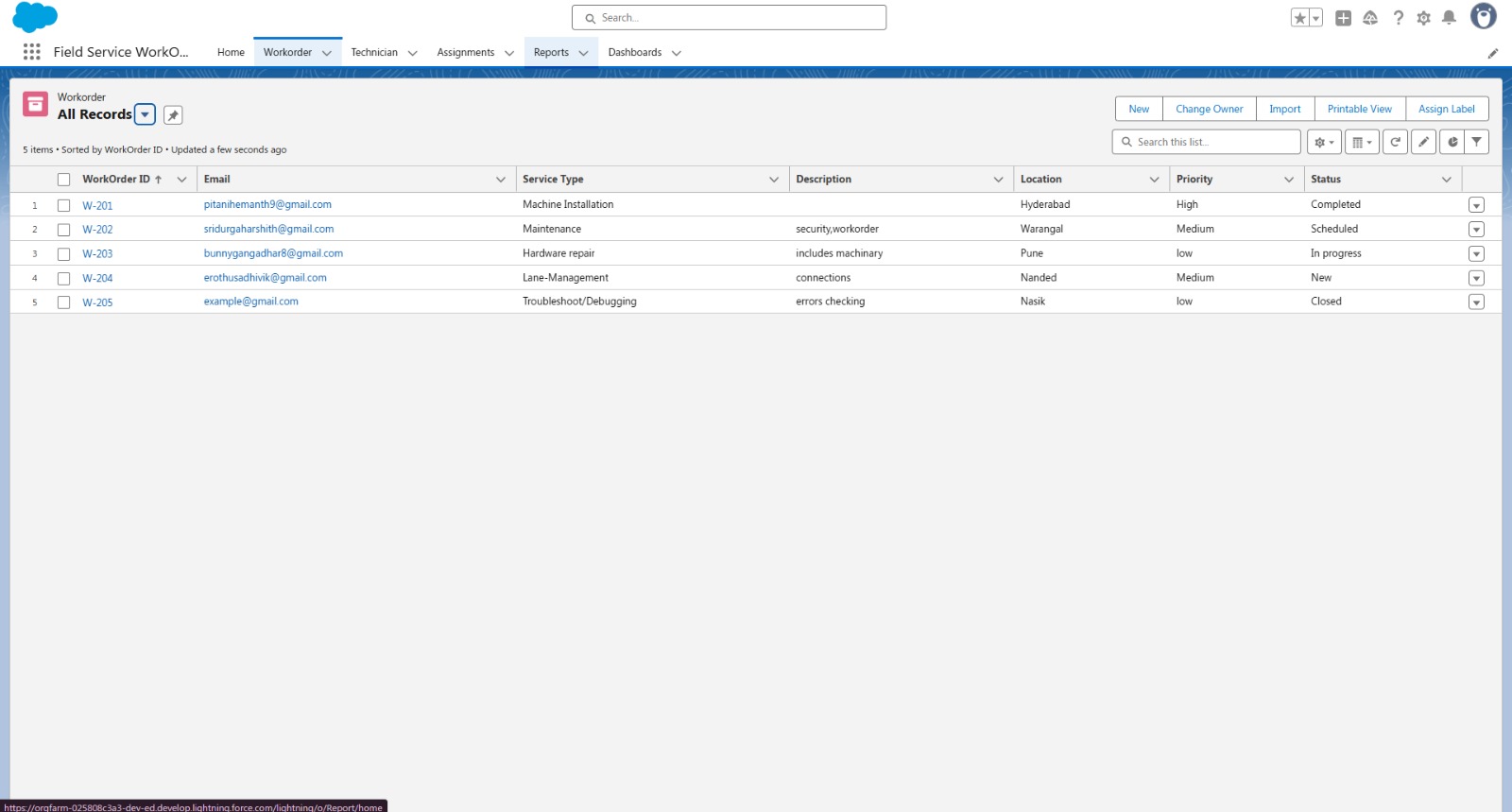
* **Optimizing Flows** and formula conditions
* **Bulkifying Apex classes** and implementing governor limit best practices

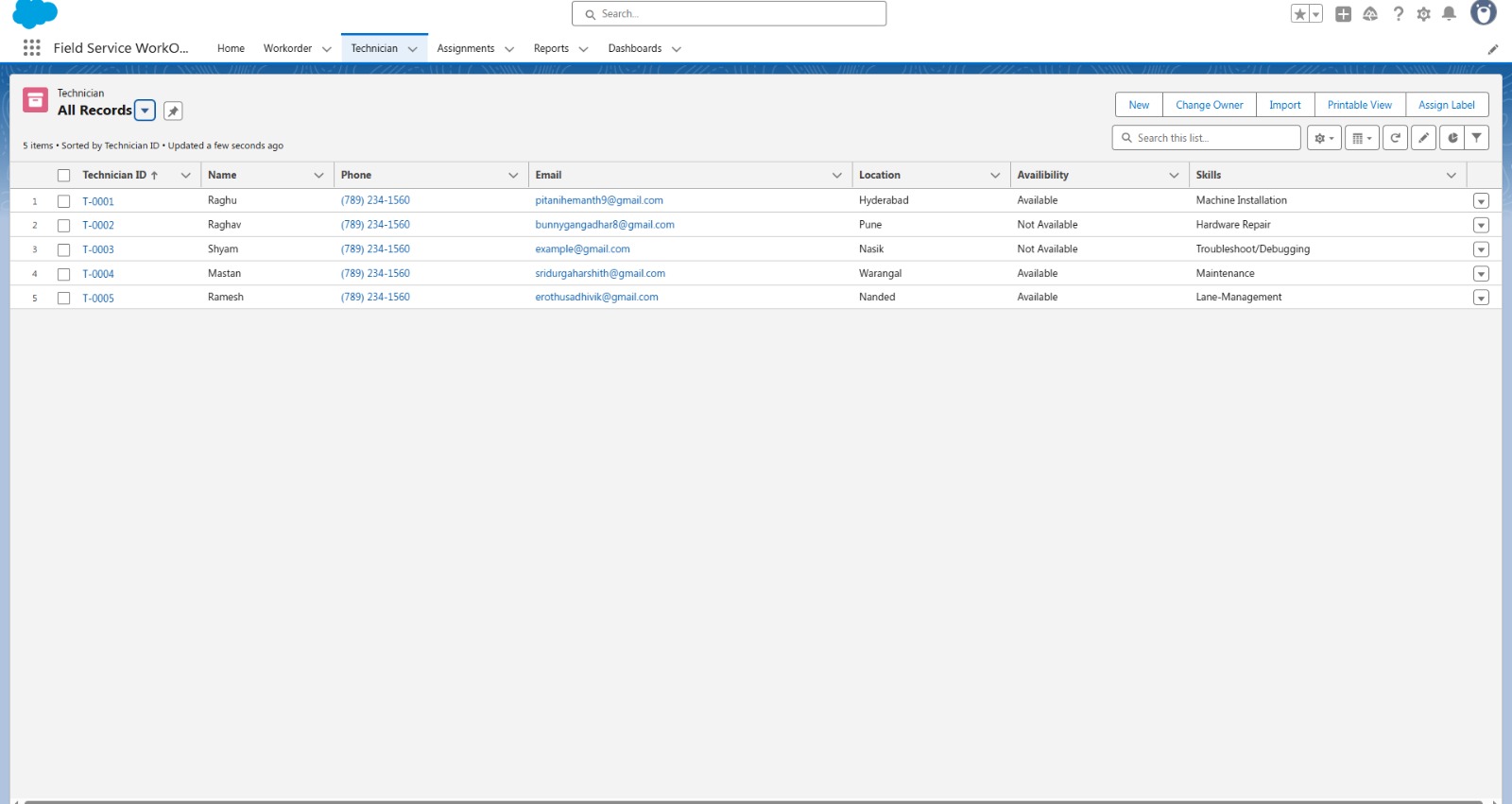
These enhancements helped ensure that the system is **scalable**, **responsive**, and aligned with best practices for enterprise-grade performance.

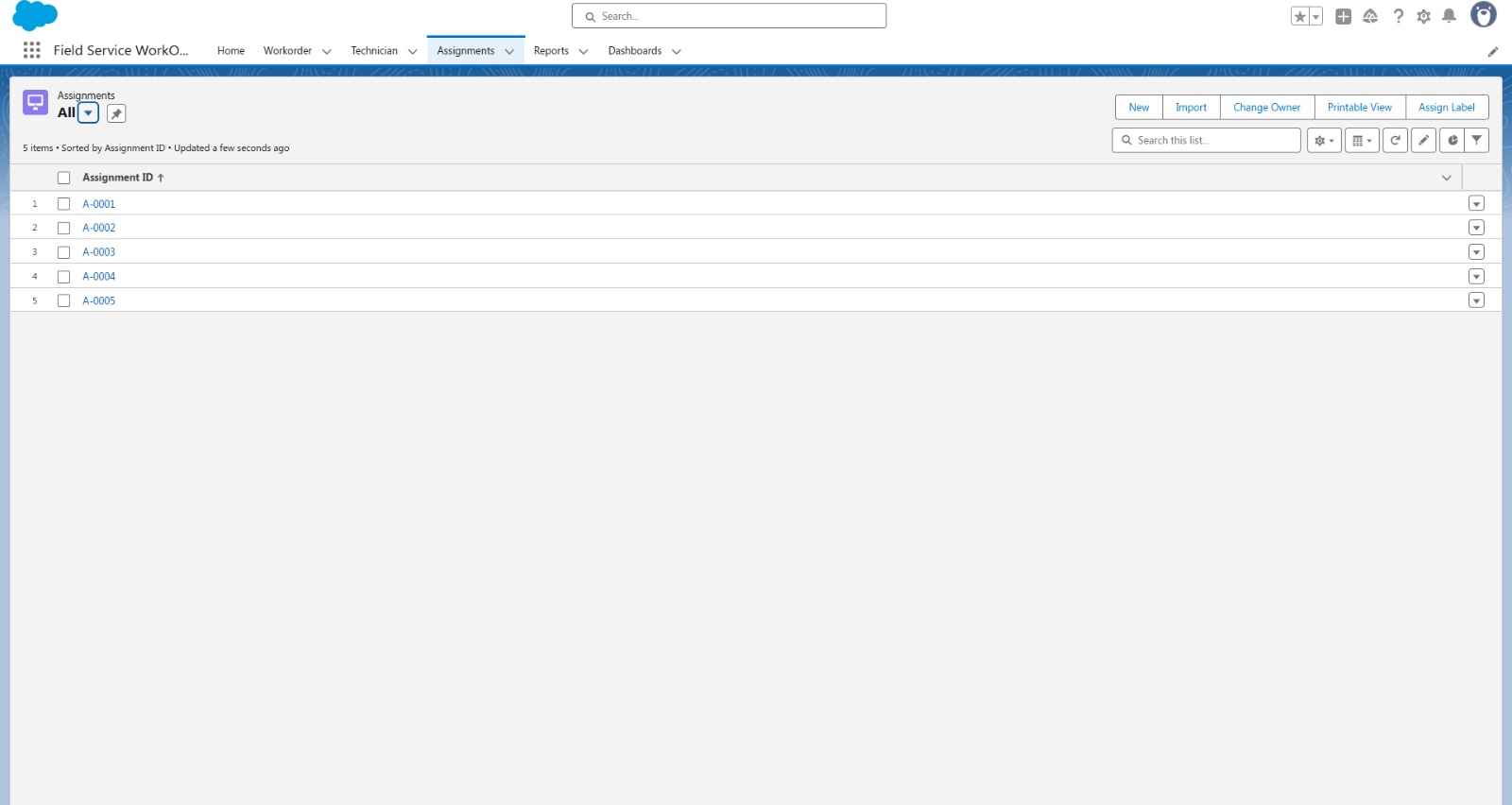
**8. RESULTS**

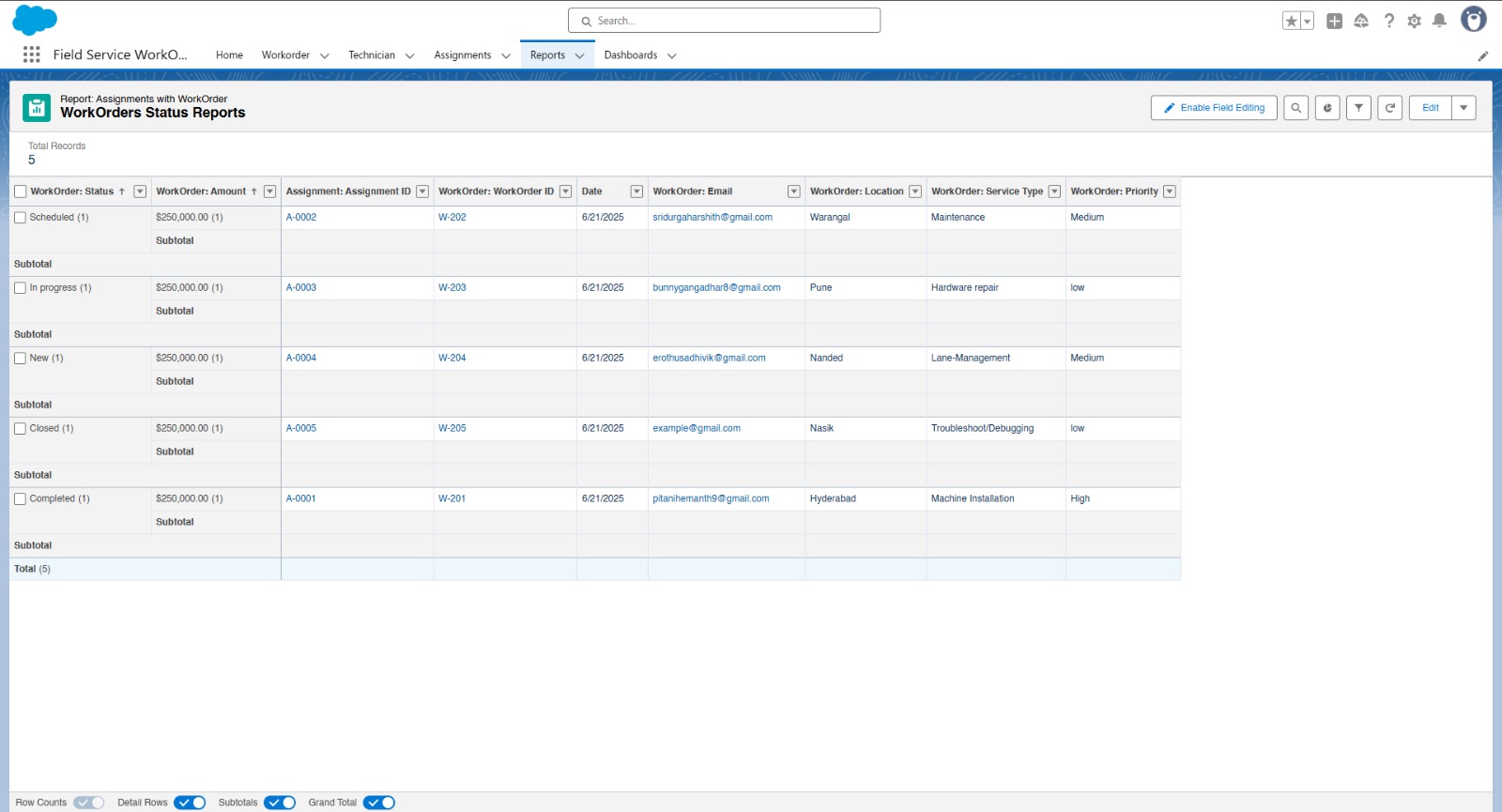
**8.1 Output Screenshots**

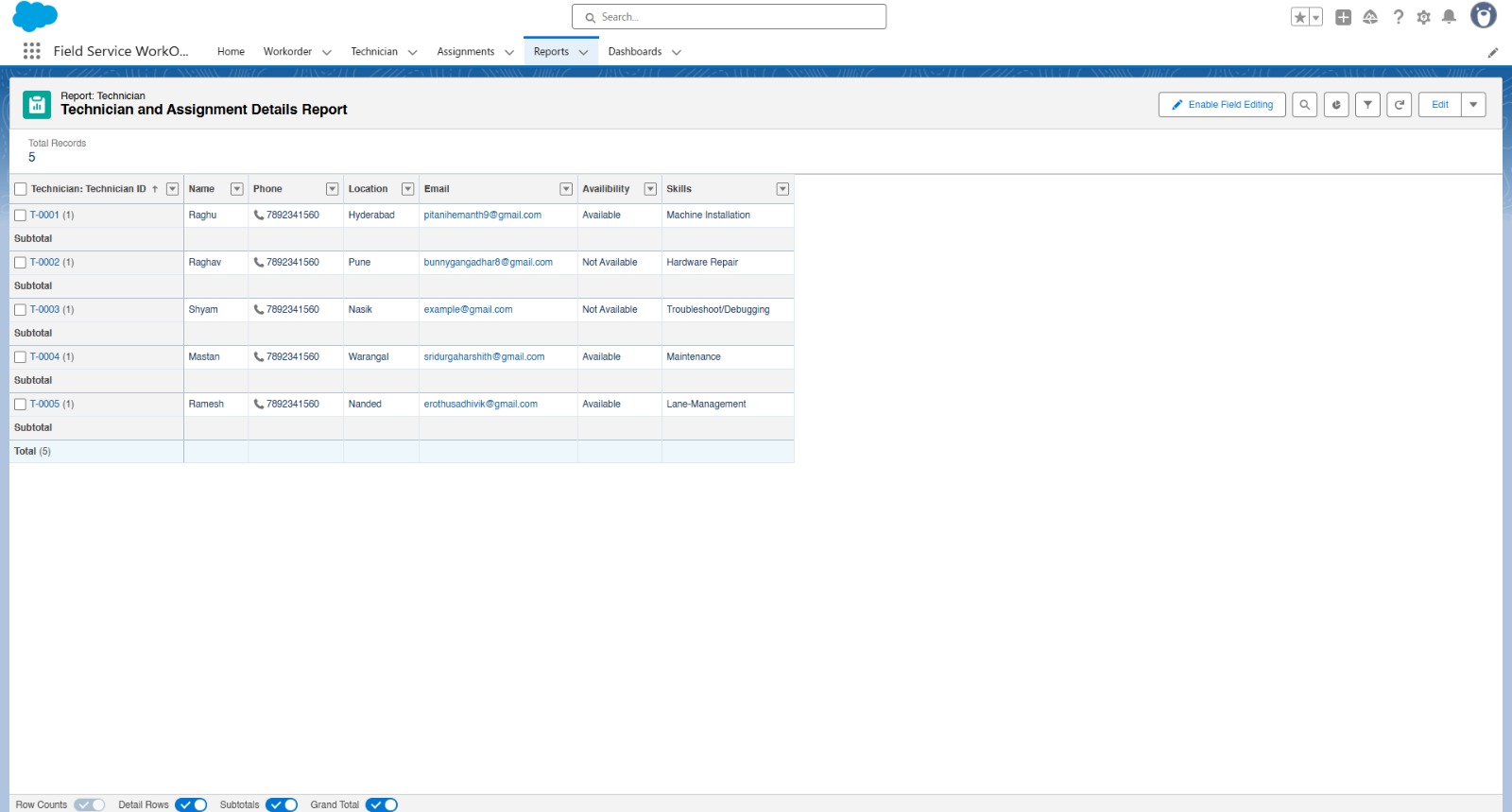


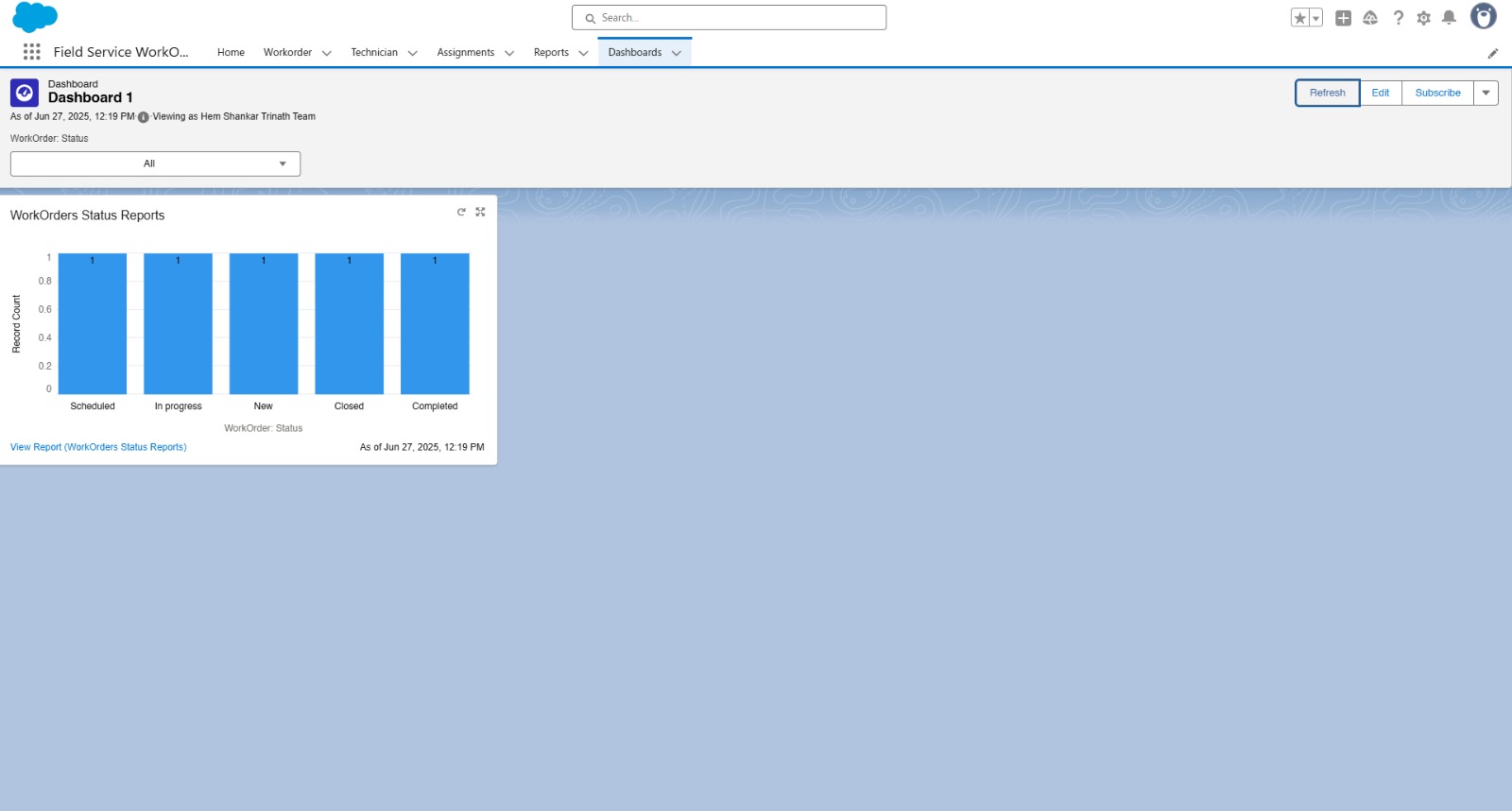


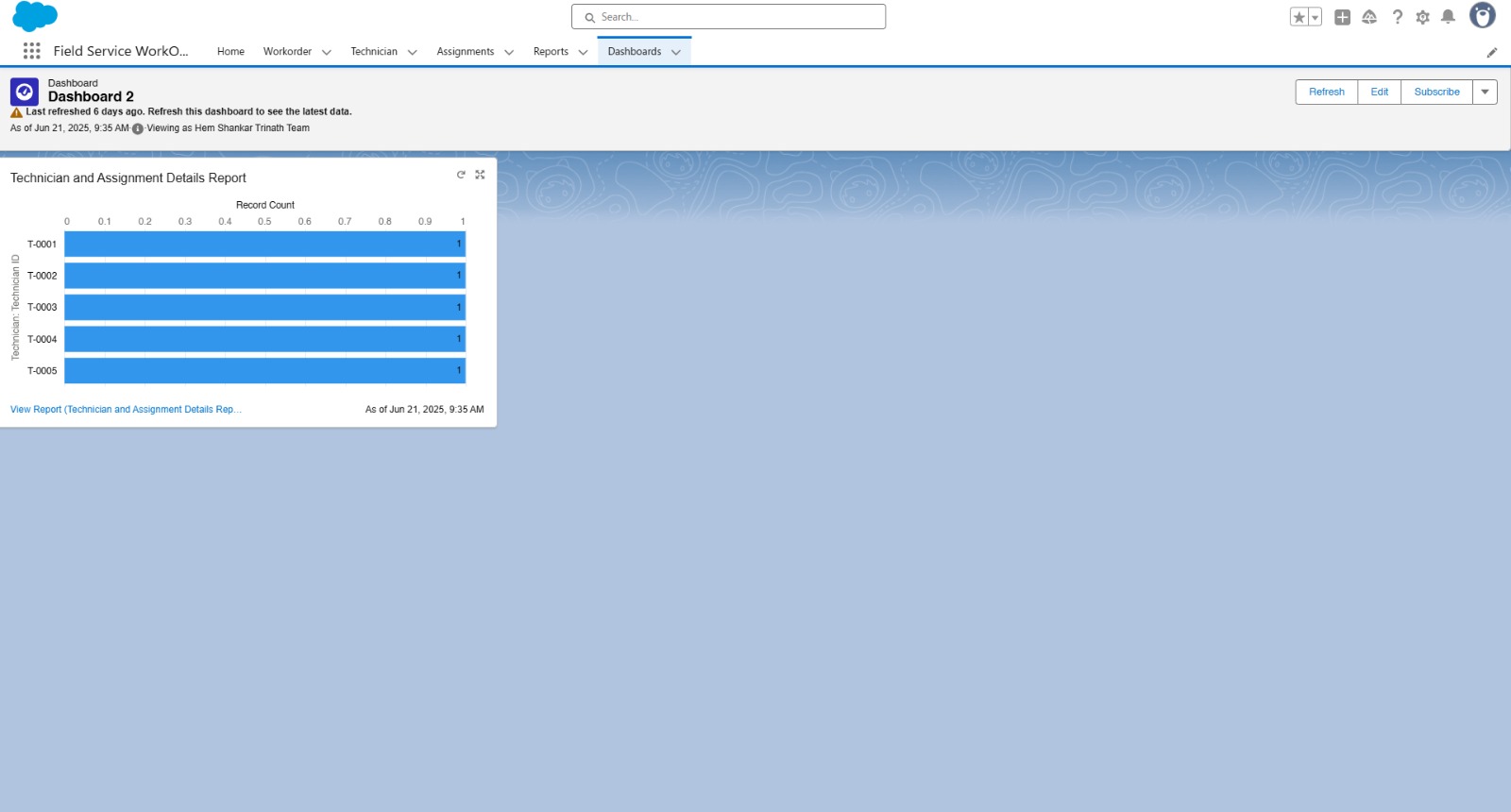












**9. ADVANTAGES & DISADVANTAGES**

### **Advantages**

● **Automation of WorkOrder Assignment**  
The system automatically assigns technicians to work orders based on location, skills, and availability, minimizing manual intervention and improving operational efficiency.

● **Centralized Service Data Management**  
All technician, work order, and assignment records are stored securely in Salesforce, enabling better data accuracy and traceability.

● **Real-Time Notifications**  
Email alerts are sent to technicians upon assignment and work order completion, ensuring timely communication and quick response.

● **Role-Based Access Control**  
Custom profiles like "Technician" ensure that users can only access relevant objects and fields, enhancing data security.

● **Scalable and Customizable Platform**  
Built on Salesforce Lightning, the system can be extended with new objects, flows, and automation features as business needs evolve.

● **Improved Service Insights**  
Dynamic reports and dashboards provide real-time visibility into technician workload, work order statuses, and service coverage areas.

● **User-Friendly Interface**  
The use of Lightning App, custom tabs, and guided layouts makes it easy for admins and technicians to navigate and manage records.

**Disadvantages**

● **Salesforce Platform Dependency**  
The solution relies on Salesforce, requiring a license and platform access for all users.

● **Initial Setup and Configuration Overhead**  
Creating custom objects, Apex classes, and automation logic takes initial time and planning.

● **No Native Offline Functionality**  
Internet access is required to use the system; field agents may face issues in low-connectivity zones.

● **Learning Curve for Non-Technical Users**  
Users unfamiliar with Salesforce may require training to use dashboards, reports, or update records.

● **Bulk Data Import Limitations**  
Migrating large sets of technician or work order data may require tools like Data Loader and careful field mapping.

**10. CONCLUSION:**

The **Field Service Work Order Optimization** system has been successfully designed and developed on the Salesforce platform to automate and streamline technician assignments and service management workflows. By leveraging custom objects, Apex logic, and scheduled automation, the system replaces manual processes with dynamic, real-time functionality.

The solution meets all business and functional requirements—ensuring proper technician allocation, automated notifications, assignment tracking, and real-time performance reporting. It has been rigorously tested for functional accuracy and performance efficiency. The system is reliable, scalable, and ready for deployment in a real-world service environment.

**11. FUTURE SCOPE**

● Integrate real-time mapping or GPS tracking for technician routing  
● Build advanced analytics dashboards for service performance KPIs  
● Enable mobile-responsive Lightning pages for field accessibility  
● Integrate chatbot support for customer or technician inquiries  
● Expand with QR code scanning for work order updates onsite

**12. APPENDIX**

**Source Code:** Apex classes and triggers (WorkOrderClass, AssignmentEmail, CompletionMail, RecordDeletions, ScheduleClass & WorkOrderTrigger, AssignmentTrigger)

**Apex Class**: WorkOrderClass

public class WorkOrderClass {  
    public static void workOrder(List<WorkOrder\_\_C> newListWorkOrder){  
        Map<Integer, List<String>> maptotech = new map<Integer,List<String>>();  
        integer num = 0;  
        List<WorkOrder\_\_c> properWo = new List<WorkOrder\_\_c>();  
        List<Assignment\_\_c> lstAssignment = new List<Assignment\_\_c>();  
        List<Technician\_\_c> techniciantoAssignment = new List<Technician\_\_c>();  
        for(WorkOrder\_\_c iter : newListWorkOrder){  
            List<String> lststring = new List<string>();   
            If(iter.Service\_Type\_\_c != null && iter.Location\_\_c != null ){  
                num = num+1;  
                properWo.add(iter);  
                lststring.add(iter.Service\_Type\_\_c);  
                lststring.add(iter.Location\_\_c);  
  
                maptotech.put(num,lststring);  
            }   
        }  
        Map<integer,Id> techId = new Map<integer,Id>();  
        Map<Id,Technician\_\_c> allTechnician = new Map<Id,Technician\_\_c>([SELECT Id, Name, Phone\_\_c, Location\_\_c, Skills\_\_c, Availibility\_\_c, Name\_\_c, Email\_\_c FROM Technician\_\_c]);  
        integer num2 = 0;  
        For(Technician\_\_c T : allTechnician.values()){  
            num2 = num2+1;  
            if(maptotech.get(num2) != null){  
                List<string> valofmap = maptotech.get(num2);  
            system.debug('error 1 ----> the maptotech is empty ---> ' + maptotech.get(num2));  
            if(valofMap.contains(t.Skills\_\_c) && ValofMap.contains(t.Location\_\_c) && t.Availibility\_\_c == 'Available'){  
                techid.put(num2,t.Id);  
            }  
            }  
             
        }  
        integer num3 = 0;  
        For(WorkOrder\_\_c W : properWo){  
            num3 = num3 + 1;  
            Assignment\_\_c A = new Assignment\_\_c();  
            A.WorkOrder\_ID\_\_c = W.Id;  
            A.Technician\_ID\_\_c = techid.get(num3);  
            lstAssignment.add(A);  
        }  
        If(!lstAssignment.IsEmpty()){  
            insert lstAssignment;  
        }  
    }  
}

**Apex Class**: AssignmentEmail

public class AssigningEmail {  
    public static void sendEmailmsg(List<Assignment\_\_c> assRec){  
        List<messaging.SingleEmailMessage> myVar = new List<messaging.SingleEmailMessage>();  
        Map<id,Technician\_\_c> tecnicians = new Map<id,Technician\_\_c>([SELECT Id, Phone\_\_c, Location\_\_c, Skills\_\_c, Name\_\_c, Email\_\_c, Availibility\_\_c, Name FROM Technician\_\_c]);  
        try{  
            for(Assignment\_\_c con : assRec){  
                if(con.Technician\_ID\_\_c != null){     
                    messaging.SingleEmailMessage mail = new messaging.SingleEmailMessage();  
                    List<String> sendTo = new List<String>();  
                    sendTo.add(tecnicians.Get(con.Technician\_ID\_\_c).Email\_\_c);  
                    mail.setToAddresses(sendTo);  
                    string subject = 'WorkOrder Assignment ';  
                    mail.setSubject(subject);  
                    string body = 'The following WorkOrder has been assigned to you ';  
                    mail.setHTMLbody(body);  
                    myVar.add(mail);  
                }  
            }  
            Messaging.sendEmail(myvar);     
        }  
        catch(exception e){  
            system.debug('Error -----> ' + e.getMessage());  
        }  
    }  
}

**Apex Class:** CompletionMail

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

**Apex Class:** RecordDeletions

public class RecordDeletions Implements Database.Batchable<Sobject>{

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

string query = 'SELECT Id, Name, WorkOrder\_ID\_\_c, Technician\_ID\_\_c, Assignment\_Date\_\_c, Completion\_Date\_\_c FROM Assignment\_\_c WHERE Completion\_Date\_\_c = LAST\_N\_DAYS:30';

         return database.GetQueryLocator(query);

    }

    public void execute(Database.BatchableContext bc, List<Assignment\_\_c> query){

        if(!Query.IsEmpty()){

            Delete Query;

        }

    }

    public void finish(Database.BatchableContext bc){

    }

}

**Apex Class:** ScheduleClass

global class ScheduleClass implements Schedulable {  
   global void execute(SchedulableContext SC) {  
        RecordDeletions delrec = new RecordDeletions();  
        database.executeBatch(delrec, 200);  
   }  
}

**Apex Triggers**

**Apex trigger:** WorkOrderTrigger

trigger WorkOrderTrigger on WorkOrder\_\_c (after insert) {

    if(trigger.isafter && trigger.isinsert){

        WorkOrderClass.workOrder(trigger.new);

    }

}

**Apex trigger:** AssignmentTrigger

trigger AssignmentTrigger on Assignment\_\_c (after insert) {  
    if(Trigger.IsAfter && Trigger.IsInsert){  
        AssigningEmail.sendEmailmsg(Trigger.New);  
    }  
}

**Dataset Link:** Not Applicable

**GitHub & Project Demo Link:**

[https://github.com/Trinath0928/Field-Service-WorkOrder-Optimization](Final%20Report%20of%20Lease%20Management%20System.docx)

[**https://youtu.be/-0N5ShB1WyA?si=fsqPN-RAUKtO58UF**](Final%20Report%20of%20Lease%20Management%20System.docx)