

# Salesforce Project Documentation: EV Charging CRM

## Project Documentation Overview

Project documentation in a Salesforce CRM implementation serves as a comprehensive record of the application's purpose, design, development, and deployment. It ensures that business requirements are clearly defined and aligned with the system's functionality. This document acts as a blueprint for developers and admins, guiding consistent development and future scalability. It supports user training, facilitates troubleshooting, and ensures smooth maintenance by detailing every component and its logic.

---

## Project Overview

This project is an EV Charging CRM built on the Salesforce platform, designed to efficiently manage a growing network of Electric Vehicle (EV) charging stations. The primary challenge this CRM solves is the lack of a centralized system for tracking station status, managing real-time slot availability, handling customer bookings, and processing payments. The CRM provides a comprehensive solution for customers to find and reserve charging times, for station managers to monitor their assets, and for admins to oversee the entire operation and generate key reports.

## Objectives

The main goal of building this CRM is to streamline all business processes related to the EV charging network and enhance the end-user experience. The project links these goals to business value by:

- **Improving customer management** by tracking customer vehicles, booking history, and support cases.
  - **Streamlining the booking process** through an intuitive interface and real-time availability tracking.
  - **Providing analytics** on station utilization, revenue, and customer behavior to drive business decisions.
- 

## Phase 1: Problem Understanding & Industry Analysis

### Business Problem

The primary challenges addressed by this project are:

- **Station Management:** Tracking the location, operational status (operational, maintenance), and specifications of all charging stations.
- **Slot Availability:** Real-time monitoring and display of individual charging slot availability within each station.
- **Bookings:** Allowing customers to find available slots, reserve charging times, and manage their bookings.
- **Payments:** Securely processing payments for charging sessions.
- **User Management:** Handling different user types (Customers, Station Managers, Admins).
- **Reporting:** Generating insights on station utilization and revenue.

## Stakeholders

- **Admin:** Oversees the entire system, manages configurations, and needs full access and reporting.
- **Customer:** The end-user who books slots, manages vehicles, and makes payments.
- **Station Manager:** Responsible for one or more stations; monitors slot status and local bookings.
- **Technician:** Handles maintenance and repairs; needs access to station/slot details and cases.

## Business Process Flow

1. **Discovery:** Customer finds nearby stations and checks slot availability.
2. **Booking:** Customer selects a station, slot, and time, then confirms.
3. **Allocation:** The system marks the slot as 'Reserved'.
4. **Session:** Customer arrives and starts the charging session.
5. **Calculation:** Session ends. System calculates the cost.
6. **Payment:** Customer pays via the integrated gateway; system records the transaction.
7. **Update:** System marks the slot as 'Available' again.

## Industry Analysis & AppExchange

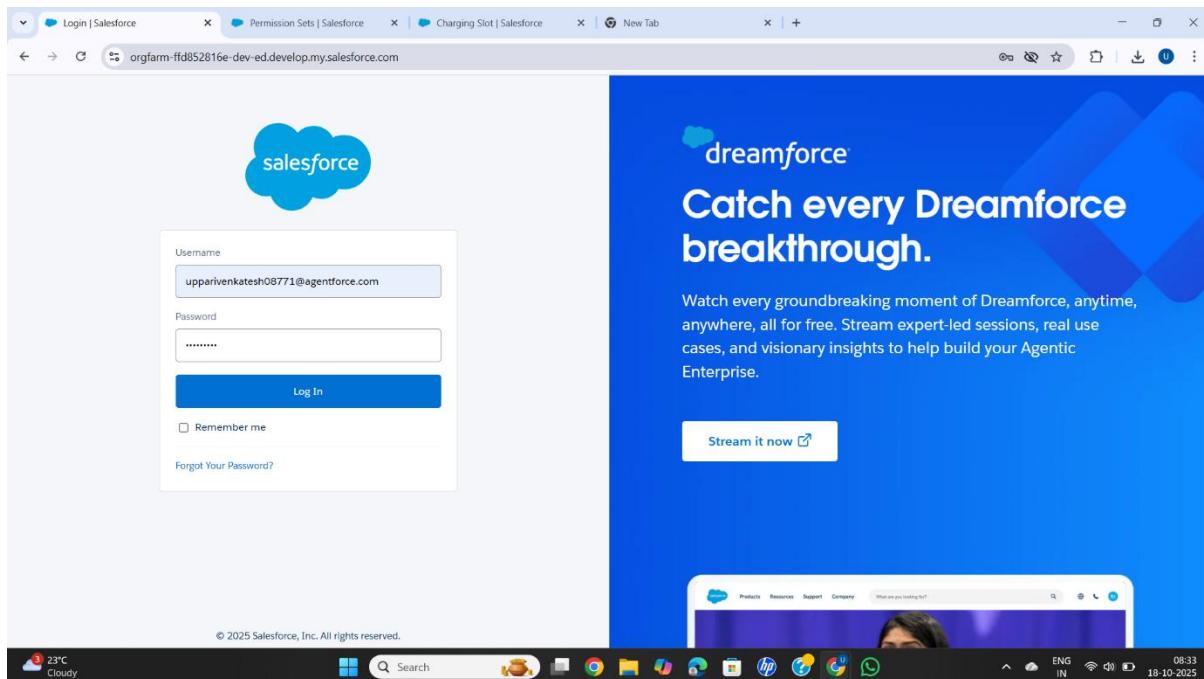
The EV charging market is rapidly expanding. Key competitors in the Indian market include Tata Power EZ Charge, ChargeGrid, Ather Grid, and Statiq. This CRM can differentiate through superior reliability tracking, a seamless booking experience, and advanced analytics. Functionality was inspired by AppExchange categories like Field Service Management (for technician dispatch), Booking/Scheduling Apps, and Subscription Billing/Payment Apps (like Stripe Connectors).

---

## Phase 2: Org Setup & Configuration

### Salesforce Edition

- **Edition Used:** Developer Edition.



### Company Profile & Business Hours

- **Company Information:** The company profile was set up with the name "EV Charging CRM Solutions Pvt Ltd" and the appropriate address, time zone, and locale.
- **Business Hours:** Default business hours for support cases were set to 9:00 AM to 6:00 PM, Monday-Friday, IST.

The screenshot shows the 'Company Information' page in the Salesforce Setup. The organization's name is 'EV Charging CRM Solutions Pvt Ltd'. The 'Organization Detail' section includes fields like Organization Name (EV Charging CRM Solutions Pvt Ltd), Primary Contact (Uppari Pedda Venkatesh), Division (United States), Fiscal Year Starts In (January), and various checkboxes for newsletter and admin newsletter. The 'Phone' and 'Fax' sections show default locales as English (United States). The 'Default Time Zone' is (GMT+07:00) Pacific Daylight Time (America/Los\_Angeles). The 'Currency Locale' is English (United States) - USD. The 'Used Data Space' is 396 KB (0%) [View], 'Used File Space' is 40 KB (0%) [View], and 'API Requests, Last 24 Hours' is 0 (15,000 max). The 'Streaming API Events, Last 24 Hours' is 0 (10,000 max). The 'Restricted Logins, Current Month' is 0 (0 max). The 'Salesforce.com Organization ID' is 00Dg000000DLb1u and the 'Organization Edition' is Developer Edition. The 'Instance' is CAN98. The page was created by OrgFarm EPIC on 10/5/2025, 7:57 PM and modified by Uppari Pedda Venkatesh on 10/17/2025, 8:06 PM.

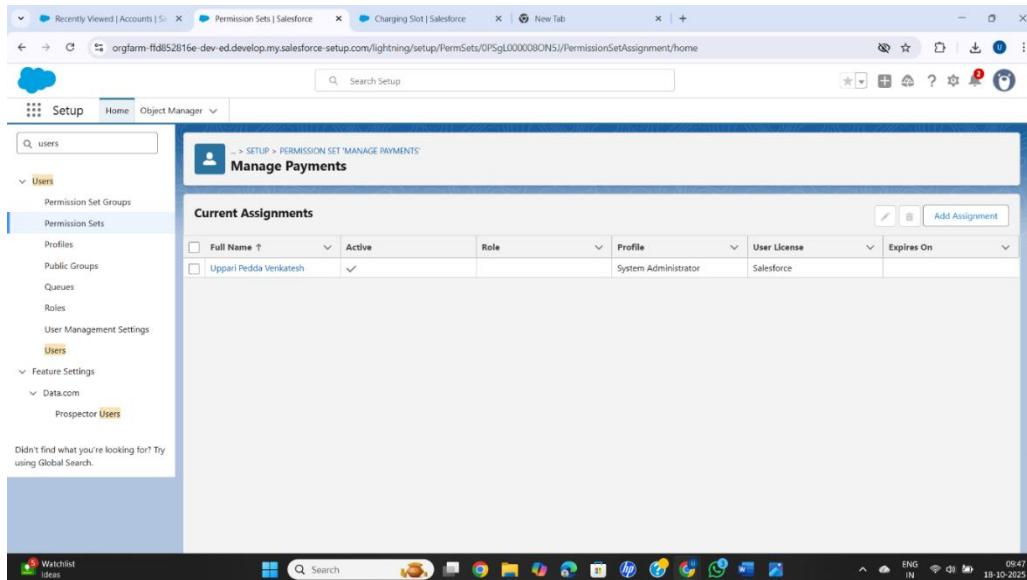
The screenshot shows the 'Business Hours' page in the Salesforce Setup. The 'Organization Business Hours' section allows selecting days and hours of availability. The 'Business Hours Detail' table lists the days of the week with their respective start and end times: Sunday (8:00 AM to 6:00 PM), Monday (8:00 AM to 6:00 PM), Tuesday (8:00 AM to 6:00 PM), Wednesday (8:00 AM to 6:00 PM), Thursday (8:00 AM to 6:00 PM), Friday (8:00 AM to 6:00 PM), and Saturday (8:00 AM to 6:00 PM). The 'Time Zone' is (GMT+05:30) India Standard Time (Asia/Colombo). The 'Default Business Hours' checkbox is unchecked. The page was created by Uppari Pedda Venkatesh on 10/17/2025, 8:11 PM and last modified by Uppari Pedda Venkatesh on 10/17/2025, 8:11 PM.

## User Setup & Roles

- User Creation:** Standard users were created to represent key stakeholders, including System Administrator, Station Manager, and Customer Support.
- Role Hierarchy:** A vertical role hierarchy was defined to enable managers to view records owned by their subordinates. The structure includes CEO, Operations Manager, Station Manager, and Station Staff.

## Permission Sets

- **Manage Payments:** A Permission Set was created to grant additional "Manage Payments" access (e.g., Edit/Delete on the Payment object) to specific finance users without changing their profile.

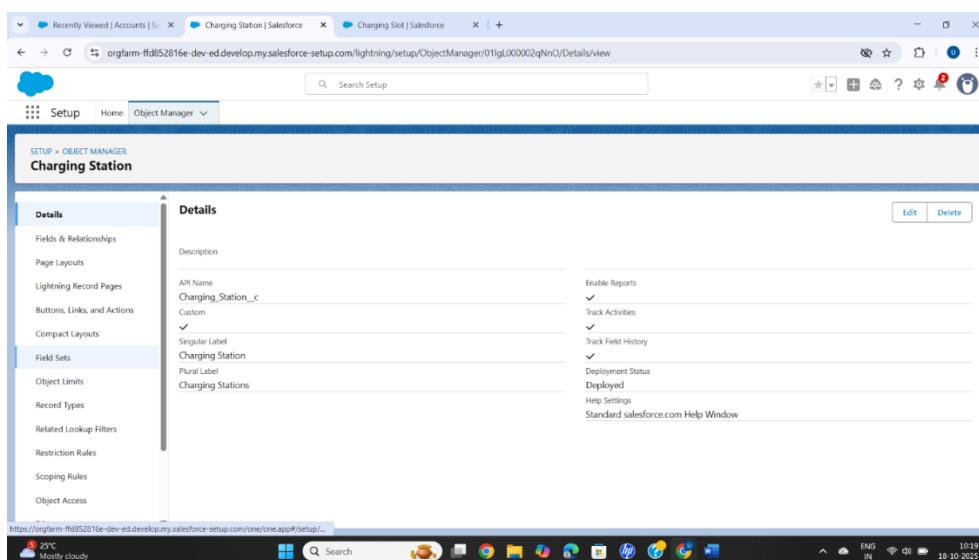


## Phase 3: Data Modeling & Relationships

### Custom Objects

The core data model for the CRM consists of the following custom objects:

- **Charging Station (Charging\_Station\_\_c):** Represents a physical charging station location.



- **Charging Slot (Charging\_Slot\_\_c):** Represents an individual charging point within a Station. This has a Master-Detail relationship to the Charging Station.

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes tabs for Recently Viewed, Accounts, and Charging Slot. The main title is "Charging Slot | Salesforce". Below the title, the URL is "orgfarm-ffd852816e-dev-ed.develop.my.salesforce-setup.com/lightning/setup/ObjectManager/01lgL000002qNqb/Details/view". The page header has a search bar labeled "Search Setup" and various browser icons. The main content area is titled "Charging Slot" under "SETUP > OBJECT MANAGER". On the left, there's a sidebar with a "Details" tab selected and a list of configuration options: Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Restriction Rules, Scoping Rules, and Object Access. The main panel displays the "Details" section for the "Charging Slot" object. It shows the API Name as "Charging\_Slot\_\_c" and the Singular Label as "Charging Slot". The right side of the panel contains sections for "Enable Reports", "Track Activities", "Track Field History", "Deployment Status" (set to "Deployed"), and "Help Settings" (set to "Standard salesforce.com Help Window"). At the bottom of the page, there's a status bar showing the weather (25°C, Mostly cloudy), system icons, and the date/time (18-10-2025).

- **Booking (Booking\_\_c):** Records a customer's reservation of a specific Slot. It has lookups to Account, Contact, Slot, and Vehicle.

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes tabs for Recently Viewed, Accounts, and Booking. The main title is "Booking | Salesforce". Below the title, the URL is "orgfarm-ffd852816e-dev-ed.develop.my.salesforce-setup.com/lightning/setup/ObjectManager/01lgL000002qODB/Details/view". The page header has a search bar labeled "Search Setup" and various browser icons. The main content area is titled "Booking" under "SETUP > OBJECT MANAGER". On the left, there's a sidebar with a "Details" tab selected and a list of configuration options: Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Restriction Rules, Scoping Rules, Object Access, and Triggers. The main panel displays the "Details" section for the "Booking" object. It shows the API Name as "Booking\_\_c" and the Singular Label as "Booking". The right side of the panel contains sections for "Enable Reports", "Track Activities", "Track Field History", "Deployment Status" (set to "Deployed"), and "Help Settings" (set to "Standard salesforce.com Help Window"). At the bottom of the page, there's a status bar showing the weather (25°C, Mostly cloudy), system icons, and the date/time (18-10-2025).

- **Payment (Payment\_\_c):** Records financial transactions related to Bookings. It has a lookup to Booking.

- **Vehicle (Vehicle\_\_c):** Represents a customer's electric vehicle, linked via lookup to Account/Contact.

**Vehicle**

**Details**

Description

API Name: Vehicle\_\_c  
Custom ✓  
Singular Label: Vehicle  
Plural Label: Vehicles

Enable Reports ✓  
Track Activities  
Track Field History ✓  
Deployment Status: Deployed  
Help Settings: Standard salesforce.com Help Window

- **Case (Case\_\_c):** Represents customer issues after charging.

**Case**

**Details**

Description

API Name: Case\_\_c  
Custom ✓  
Singular Label: Case  
Plural Label: Cases

Enable Reports ✓  
Track Activities ✓  
Track Field History ✓  
Deployment Status: Deployed  
Help Settings: Standard salesforce.com Help Window

## Key Fields & Relationships

- **Charging Station (Charging\_Station\_\_c):**
  - Location\_\_c (Geolocation)
  - Status\_\_c (Picklist: Operational, Maintenance, Offline)

**Charging Station**

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address_c	Long Text Area(500)		
Available Slots	Available_Slots_c	Formula (Number)		
Contact Person	Contact_Person_c	Text(100)		
Contact Phone	Contact_Phone_c	Phone		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Location	Location_c	Text(25)		
Occupied Slots	Occupied_Slots_c	Number(3, 0)		
Operating Hours	Operating_Hours_c	Text(50)		

- **Slot (Charging\_Slot\_\_c):**

- Station\_\_c (Master-Detail to Station)
- Availability\_\_c (Picklist: Available, Unavailable, Reserved) - This is updated by automation.
- Hourly\_Rate\_\_c (Currency) - Used for payment calculation.

**Charging Slot**

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Last Maintenance Date	Last_Maintenance_Date_c	Date		
Last Modified By	LastModifiedById	Lookup(User)		
Occupied Slots	Occupied_Slots_c	Number(3, 0)		
Owner	OwnerId	Lookup(User,Group)		
Power Rating	Power_Rating_c	Number(16, 2)		
Slot Health	Slot_Health_c	Picklist		
Slot Number	Name	Auto Number		
Station	Charging_Station_c	Lookup(Charging Station)		
Total Slots	Total_Slots_c	Number(3, 0)		

- **Booking (Booking\_\_c):**

- Slot\_\_c (Lookup to Slot)
- Contact\_\_c (Lookup to Contact)
- Start\_Time\_\_c (Date/Time)

- **Status\_\_c** (Picklist: Pending, Confirmed, Completed, Cancelled) - This field triggers automations.
- **Total\_Price\_\_c** (Currency)
- **Payment\_Status\_\_c** (Picklist: Unpaid, Paid, Failed)
- **Reminder\_Sent\_\_c** (Checkbox) - Used by the reminder flow.

**Fields & Relationships**

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Booking Number	Name	Auto Number		✓
Booking Status	Booking_Status__c	Picklist		
Charging Slot	Charging_Slot__c	Lookup(Charging Slot)		✓
Charging Station	Charging_Station__c	Lookup(Charging Station)		✓
Contact	Contact__c	Lookup(Contact)		✓
Cost	Cost__c	Currency(16, 2)		
Created By	CreatedById	Lookup(User)		
Customer	Customer__c	Lookup(Contact)		✓
Duration	Duration__c	Formula (Number)		

## • Payment (Payment\_\_c):

- **Booking\_\_c** (Lookup to Booking)
- **Amount\_\_c** (Currency)
- **Status\_\_c** (Picklist: Pending, Succeeded, Failed)
- **Transaction\_ID\_\_c** (Text, External ID) - For linking to the payment gateway

**Fields & Relationships**

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Contact	Contact__c	Lookup(Contact)		✓
Created By	CreatedById	Lookup(User)		
Customer	Customer__c	Lookup(Contact)		✓
Invoice URL	Invoice_URL__c	URL(255)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User Group)		✓
Payment Date	Payment_Date__c	Date		
Payment Mode	Payment_Mode__c	Picklist		
Payment Number	Name	Auto Number		✓
Transaction ID	Transaction_ID__c	Text(50)		

- **Vehicle (Vehicle\_\_c):**

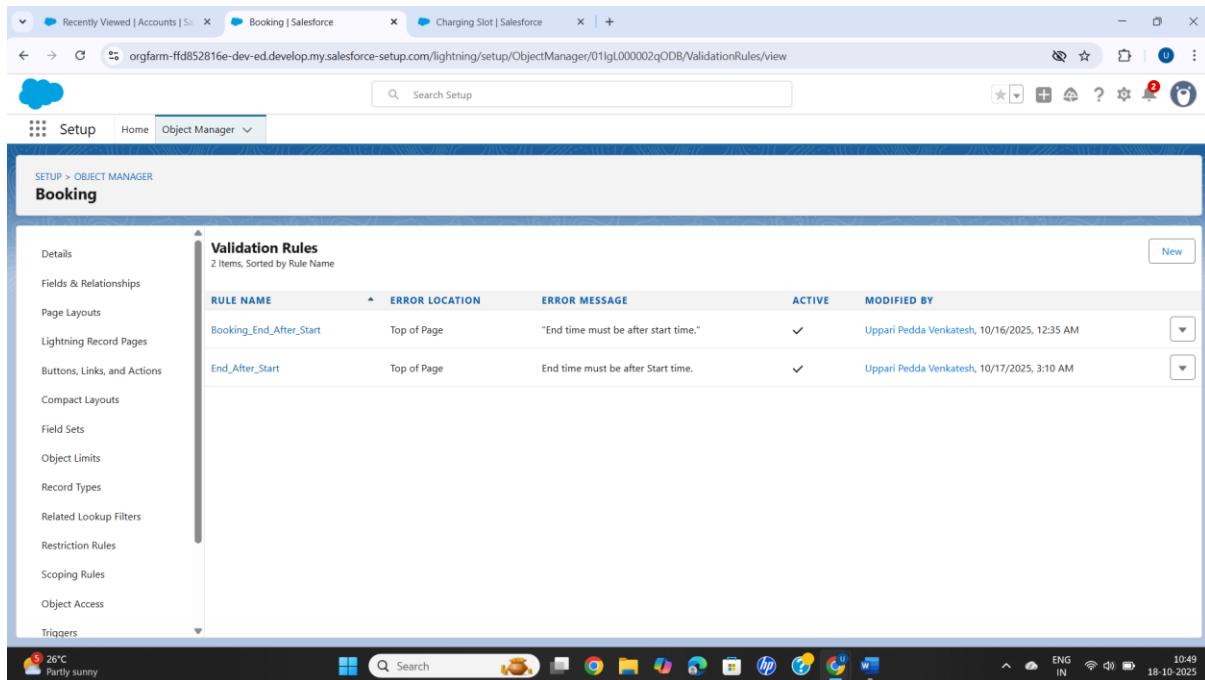
- **Vehicle\_Identifier\_\_c** (Text, Name Field - e.g., License Plate)
- **Contact\_\_c** (Lookup to Contact)

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Battery Capacity	Battery_Capacity__c	Number(5, 2)		
Connector Type	Connector_Type__c	Picklist		
Contact	Contact__c	Lookup(Contact)		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		
Vehicle Name	Name	Text(80)		
Vehicle Number	Vehicle_Number__c	Text(20)		
Vehicle Type	Vehicle_Type__c	Picklist		

## Phase 4: Process Automation (Admin)

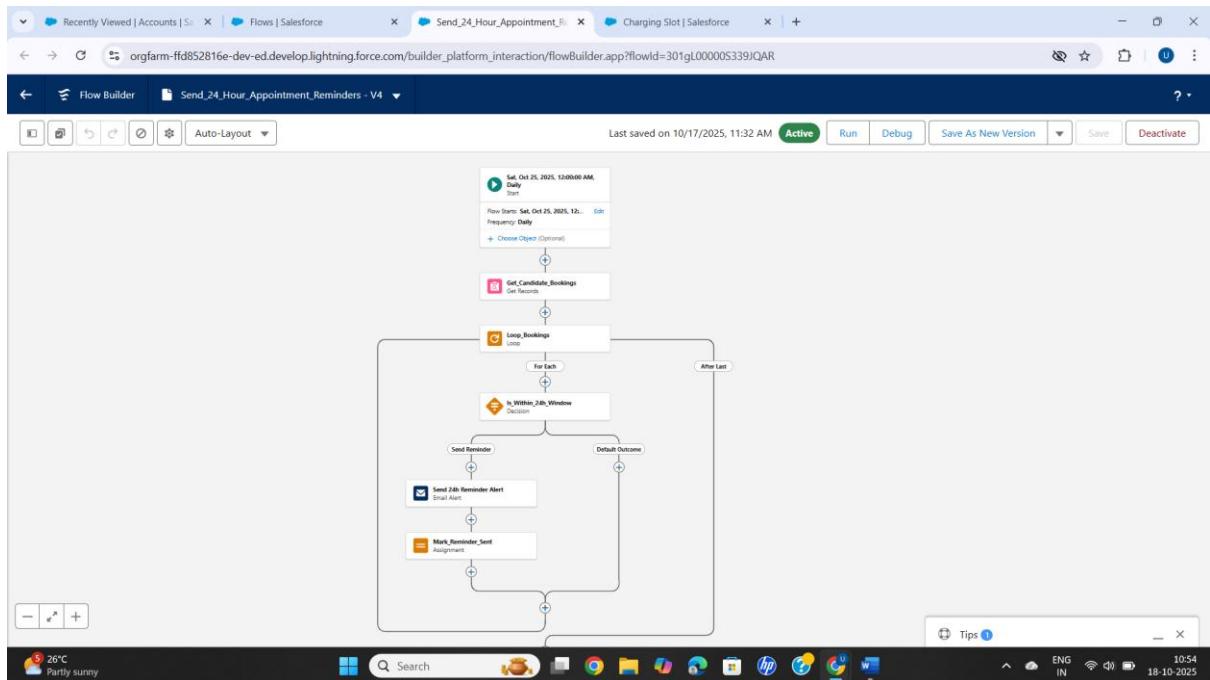
### Validation Rules

- **Use Case:** Prevent Booking End Time Before Start Time.
- **Object:** Booking\_\_c.
- **Error Condition Formula:** End\_Time\_\_c < Start\_Time\_\_c.
- **Error Message:** "Booking End Time cannot be earlier than the Start Time."



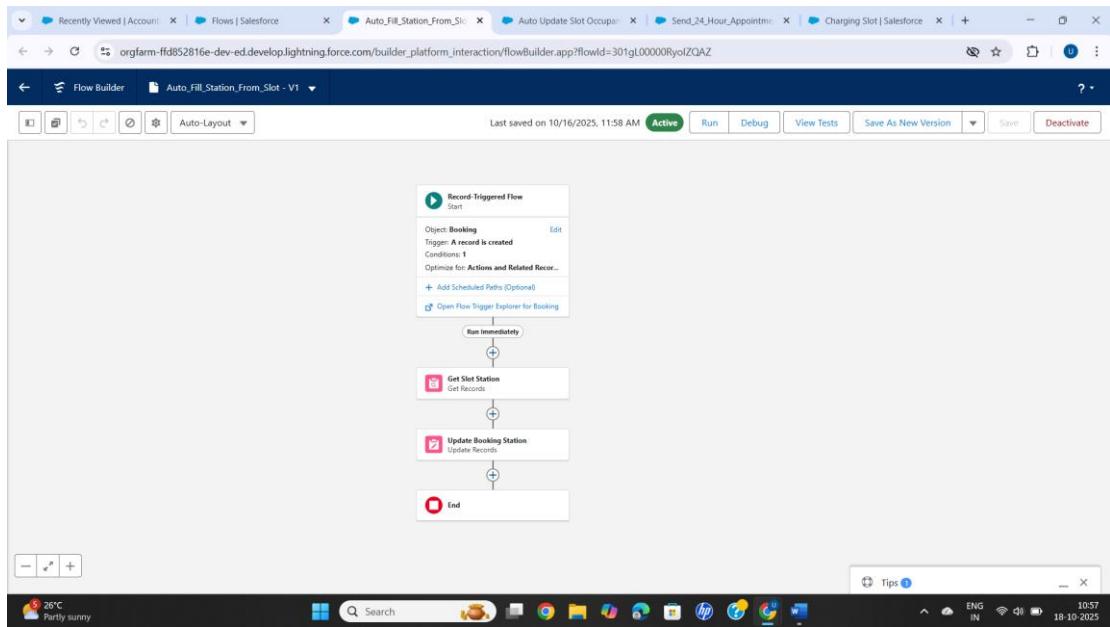
## Scheduled Flow: Send 24-Hour Appointment Reminders

- **Flow Type:** Schedule-Triggered Flow.
- **Trigger:** Runs on a defined schedule (e.g., Hourly).
- **Logic:**
  1. **Get Records:** Finds Booking\_\_c records where Start\_Time\_\_c is 24 hours from now AND Reminder\_Sent\_\_c is false.
  2. **Loop:** Iterates through the found Bookings.
  3. **Send Email Alert:** Uses a standard Email Alert action (BookingReminder\_24h\_EmailAlert) to notify the customer.
  4. **Update Records:** After the loop, performs a single update to set the Reminder\_Sent\_\_c checkbox to true for all processed records.



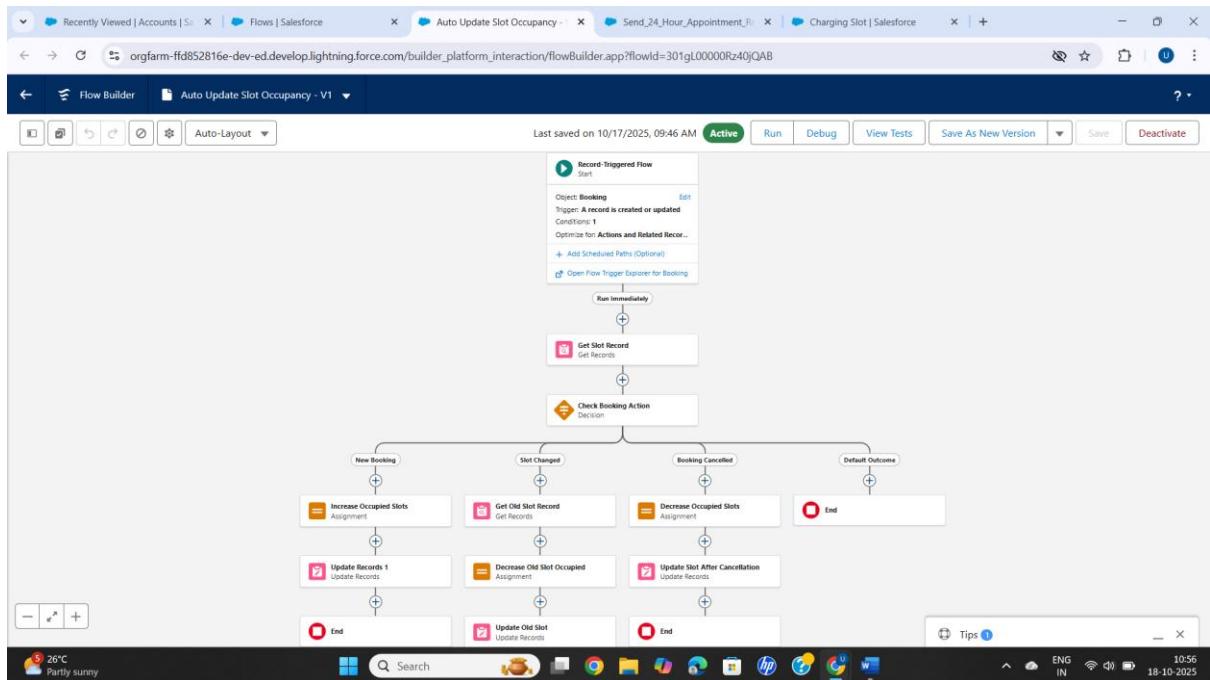
## Record-Triggered Flow: Auto Update Slot Occupancy

- Flow Type:** Record-Triggered Flow (After Save).
- Object:** Booking\_\_c.
- Entry Conditions:** Runs when Status\_\_c is changed to 'Confirmed' or 'Cancelled'.
- Logic:**
  - Get Records:** Retrieves the related Slot\_\_c record.
  - Decision:** Checks if the Booking status is 'Confirmed' or 'Cancelled'.
  - Update Records (Confirmed):** If 'Confirmed', updates the Slot's Availability\_\_c to 'Unavailable'.
  - Update Records (Cancelled):** If 'Cancelled', updates the Slot's Availability\_\_c to 'Available'.



## Screen Flow: Guided Station Creation

- **Flow Type:** Screen Flow.
- **Logic:**
  - Screen 1:** Collects Station\_\_c details (Name, Address).
  - Create Records:** Creates the new Station\_\_c record.
  - Screen 2:** Asks "How many Slots to create for this Station?".
  - Loop:** Loops from 1 to the number entered.
  - Assignment (Inside Loop):** Adds new Slot record variables to a collection.
  - Create Records:** After the loop, performs a single create action to insert all Slot\_\_c records in the collection.
  - Screen 3:** Displays a confirmation message.



## Phase 5: Apex Programming (Developer)

### Apex Trigger: BookingTrigger

- **Object:** Booking\_\_c.
- **Events:** after insert, after update.
- **Purpose:** This trigger is "logic-less." It delegates all processing to the BookingTriggerHandler class, which is a best practice.
- **Logic:**
  - On after insert, it calls `BookingTriggerHandler.handleAfterInsert(Trigger.new)`.
  - On after update, it calls `BookingTriggerHandler.handleAfterUpdate(Trigger.new, Trigger.oldMap)`.

```

1 * trigger BookingsPreventOverlap on Bookings__c (before insert, before update) {
2     // Map slotId -> list of incoming bookings for that slot (in this transaction)
3     Map<Id, List<Bookings__c>> slotToIncoming = new Map<Id, List<Bookings__c>>();
4     for (Bookings__c b : Trigger.new) {
5         if (b.Charging_Slot__c == null) continue;
6         // skip cancelled bookings (adjust based on your Status picklist)
7         if (b.Status__c == 'Cancelled') continue;
8         if (!slotToIncoming.containsKey(b.Charging_Slot__c)) {
9             slotToIncoming.put(b.Charging_Slot__c, new List<Bookings__c>());
10        }
11        slotToIncoming.get(b.Charging_Slot__c).add(b);
12    }
13    if (slotToIncoming.isEmpty()) return;
14
15    // Query existing (persisted) bookings in those slots (exclude cancelled)
16    Set<Id> slotIds = slotToIncoming.keySet();
17    List<Bookings__c> existing = [
18        SELECT Id, Start_Time__c, End_Time__c, Charging_Slot__c, Status__c
19        FROM Bookings__c
20        WHERE Charging_Slot__c IN :slotIds

```

**Logs**

User	Application	Operation	Time	Status	Read	Size
Uppari Peda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:05:43 AM	Success	Unread	523 bytes

Filter Click here to filter the log list

## Apex Class: BookingTriggerHandler

This class contains the core logic for booking processing:

- **handleAfterInsert(List<Booking\_\_c> newBookings):**
  - **Purpose:** Updates the related Slot\_\_c to 'Unavailable' when a new Booking is 'Confirmed'.
  - **Logic:** Collects Slot\_\_c IDs from confirmed Bookings into a Set<Id>, queries for those slots, updates their Availability\_\_c field, and performs a single DML update. It includes a try-catch block for exception handling.
- **handleAfterUpdate(List<Booking\_\_c> newBookings, Map<Id, Booking\_\_c> oldMap):**
  - **Purpose:** Handles status changes on existing Bookings.
  - **Logic:**
    1. Iterates through newBookings and compares the new Status\_\_c with the oldMap version.
    2. If Status changes to 'Confirmed', it updates the related Slot to 'Unavailable'.
    3. If Status changes to 'Cancelled', it updates the related Slot to 'Available'.

- Crucially, if Status changes to 'Confirmed', it calls the asynchronous future method PaymentGatewayIntegration.processPayment(...) to handle the payment callout.

The screenshot shows the Salesforce Developer Console interface. The code editor tab is set to 'BookingTriggerHandler.apxc'. The code is as follows:

```

1 * public class BookingTriggerHandler {
2
3     // (Your handleAfterInsert method is already here...)
4     // public static void handleAfterInsert(List<Booking__c> newBookings) { ... }
5
6
7     // --- ADD THIS NEW METHOD ---
8     public static void handleAfterUpdate(List<Booking__c> newBookings, Map<Id, Booking__c> oldMap) {
9
10        for (Booking__c newBooking : newBookings) {
11            // Get the old version of the record
12            Booking__c oldBooking = oldMap.get(newBooking.Id);
13
14            // Check if the status *just changed* to 'Confirmed'
15            if (newBooking.Status__c == 'Confirmed' && oldBooking.Status__c != 'Confirmed') {
16
17                // Fire and forget! Call the future method.
18                // We pass the ID and a price.
19                // (You must have a price field on your booking, like Total_Price__c)
20                PaymentGatewayIntegration.processPayment(
21
22
23
24
25
26
27
28
29
2

```

The logs section shows two entries:

User	Operation	Time	Status	Read	Size
Uppari Pedda Venkatesh	ApexTestHandler	10/18/2025, 11:05:57 AM	Success	Unread	2.2 KB
Uppari Pedda Venkatesh	ApexTestHandler	10/18/2025, 11:05:43 AM	Success	Unread	523 bytes

The system tray at the bottom indicates the date as 18-10-2025 and the time as 11:08.

## Test Classes: BookingTrigger\_Test

- Purpose:** To verify the trigger and handler functionality and achieve >75% code coverage for deployment.
- Key Components:**
  - @isTest Annotation: Marks the class as test code.
  - @testSetup Method: Creates common test data (Accounts, Stations, Slots) for all test methods.
  - Test Methods:** Each method tests a specific scenario (e.g., testAfterInsertTrigger\_UpdatesSlot, testAfterUpdate\_CancelFreesSlot).
  - Arrange, Act, Assert:** Test methods set up data, perform an action (like DML), and use System.assertEquals() to verify the outcome is correct.
  - Test.startTest() and Test.stopTest() are used to reset governor limits around the action being tested.

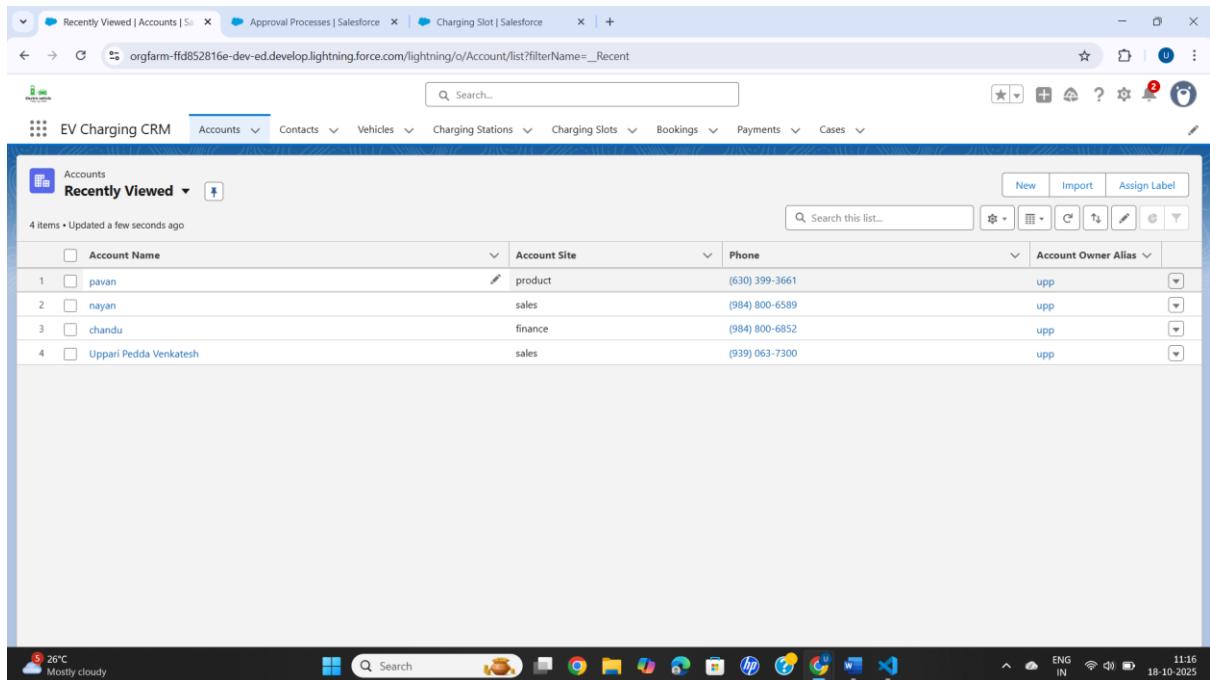
User	Application	Operation	Time	Status	Read	Size
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:10:29 AM	Success	Unread	2.19 KB
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:10:29 AM	Success	Unread	523 bytes
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:05:57 AM	Success	Unread	2.2 KB
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:05:43 AM	Success	Unread	523 bytes

Filter Click here to filter the log list

## Phase 6: User Interface Development

### Lightning App Builder: Custom App & Home Page

- **Custom App: "EV Charging CRM"**
  - A dedicated Lightning App was created to provide users with a focused experience.
  - **Tabs Added:** Includes Home, Accounts, Contacts, Vehicles, Charging Stations, Charging Slots, Bookings, and Payments, ordered logically.
  - **Profiles:** Assigned to System Administrator and other relevant profiles like Station Manager.



- **Custom Home Page: "EV Charging Home Page"**
  - A custom Home Page was designed using the Lightning App Builder.
  - **Components:**

- **Dashboard Component:** Added to the main region to display the "EV Charging Overview" dashboard.

- **Rich Text Component:** Used in the sidebar to provide "Quick Links" to common actions.

- **Activation:** This page was activated and assigned as the App Default for the "EV Charging CRM" app.

```

force-app > main > default > lwc > bookASlot > bookASlot.js -->
1 import { LightningElement, wire, track } from 'lwc';
2 import { ShowToastEvent } from 'lightning/platform/showToastEvent';
3
4 // 1. Import Apex methods
5 import getAvailableSlots from '@salesforce/apex/bookingController.getStations';
6 import getAvailableSlots from '@salesforce/apex/bookingController.getAvailableSlots';
7
8 // 2. Import schema for creating the record
9 import { Booking__c } from 'lightning/uiRecordApi';
10 import { FORCECOMSTEPPED } from '@salesforce/schema/Booking__c';
11 import { STATION_FIELD } from '@salesforce/schema/Booking__c.Charging_Station__c';
12 import { SLOT_FIELD } from '@salesforce/schema/Booking__c.Charging_Slot__c';
13 import { START_TIME_FIELD } from '@salesforce/schema/Booking__c.Start_Time__c';
14 import { STATUS_FIELD } from '@salesforce/schema/Booking__c.Status__c';
15
16
17 export default class BookASlot extends LightningElement {
18     // Form field values
19     selectedStationId = '';
20     selectedSlotId = '';
21     startTime;
22
23     // Options for checkboxes
24 }

```

```

<force-app> > main > default > lwc > bookASlot > bookASlot-meta.xml --> LightningComponentBundle
1 <xml version='1.0' encoding='UTF-8'>
2 <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3     <apiVersion>59.0</apiVersion>
4     <isExposed>true</isExposed>
5     <targets>
6         <target>lightning_AppPage</target>
7         <target>lightning_HomePage</target>
8     </targets>
9 </LightningComponentBundle>

```

## Lightning Web Component (LWC): "Book a Slot" Form

- **Component Name:** bookASlot.
- **Purpose:** Provides a dynamic form for users to create new Booking\_\_c records, as standard record creation cannot dynamically filter slots based on a station.

- **Key Features:**
    - **Dynamic Slot Fetching:**
      1. A lightning-combobox for "Select a Station" is populated on load using a @wire call to an Apex method (`bookingController.getStations()`).
      2. When a station is selected, the `handleStationChange` JavaScript function **imperatively** calls another Apex method (`bookingController.getAvailableSlots(stationId)`).
      3. The second combobox, "Select an Available Slot," is then populated with only the available slots for the chosen station.
    - **Record Creation:** On submission, the `handleBookNow` function uses the `createRecord` wire adapter to create the new `Booking__c` record with the selected slot and start time.
    - **User Feedback:** Uses lightning-spinner during loading and `ShowToastEvent` to display success or error messages.
  - **Placement:** The LWC was added to the "EV Charging Home Page" using the Lightning App Builder.
- 

## Phase 7: Integration & External Access

This phase simulates a callout to a third-party payment gateway when a booking is confirmed.

### Configuration for Callouts

- **Remote Site Settings:** A Remote Site named `Payment_API_Mock` was created to whitelist the mock API domain (<https://httpbin.org>) for Apex callouts.
- **Named Credentials (Preferred Method):**
  - A Named Credential named `Payment_API` was created, pointing to the same URL (<https://httpbin.org>).
  - This is the preferred method as it separates the endpoint URL and authentication from the code.

- **Authentication:** Set to Anonymous and No Authentication as the mock API is public.

```

1  public class PaymentGatewayIntegration {
2      public static void processPayment(Id bookingId, Decimal amount) {
3          Id = bookingId;
4          Payment_Status__c = 'Paid' // (Assuming you have this field)
5          update booking;
6      } else {
7          // Handle error
8          System.debug('Payment Failed: ' + res.getStatus());
9      }
10     } catch (System.CalloutException e) {
11         // Handle callout exception
12         System.debug('Callout error: ' + e.getMessage());
13     }
14 }

```

```

1  /*
2   * This class handles all external payment processing.
3   * It uses a @future method to make a callout,
4   * which is required when invoked from a trigger.
5   */
6  public class PaymentGatewayIntegration {
7
8      // This method runs in the background.
9      @future(callout=true)
10     public static void processPayment(Id bookingId, Decimal amount) {
11
12         // 1. Build the HTTP Request
13         HttpRequest req = new HttpRequest();
14
15         // 2. Set the Endpoint using the Named Credential
16         // 'callout:Payment_API' is the magic part.
17         // '/post' is the specific path on httpbin.org we want to send to.
18         req.setEndpoint('callout:Payment_API/post');
19         req.setMethod('POST');
20         req.setHeader('Content-Type', 'application/json;charset=UTF-8');
21
22     }
23 }

```

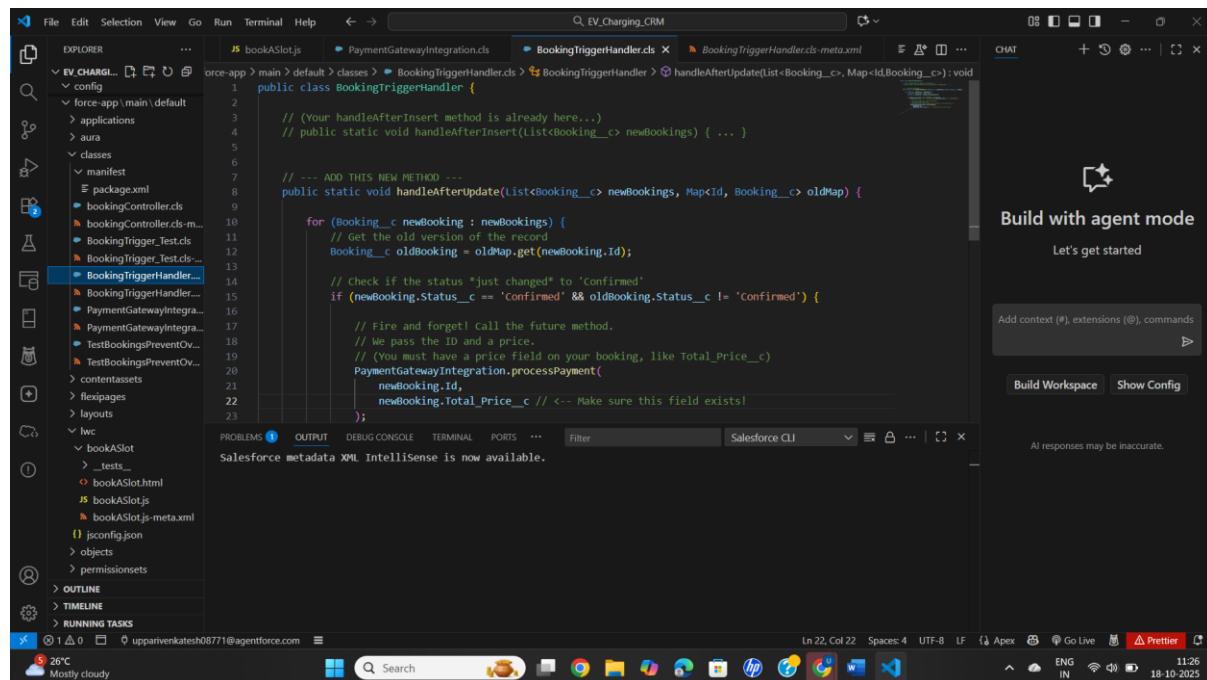
User	Application	Operation	Time	Status	Read	Size
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:24:12 AM	Success	Unread	525 bytes
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:24:12 AM	Success	Unread	2.2 KB
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:10:29 AM	Success	Unread	2.19 KB
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:10:29 AM	Success	Unread	523 bytes
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:05:57 AM	Success	Unread	2.2 KB
Uppari Pedda Venkatesh	Unknown	ApexTestHandler	10/18/2025, 11:05:43 AM	Success	Unread	523 bytes

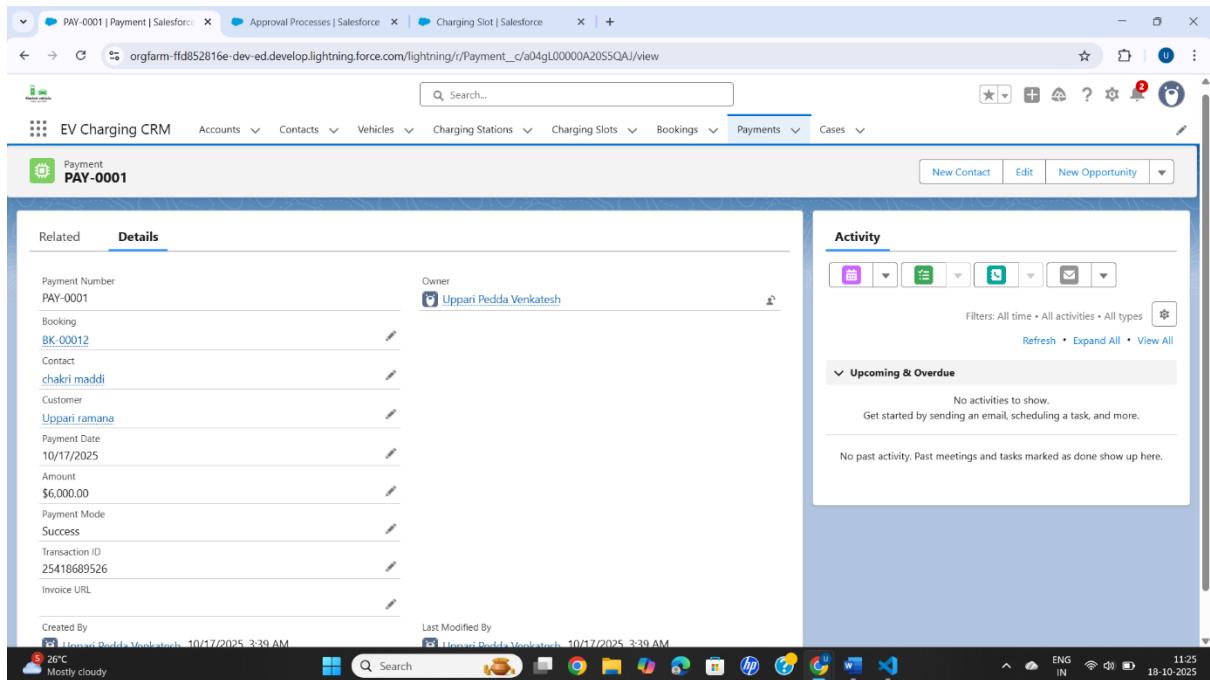
## Apex Callout Example (Mock Payment Processing)

- **Apex Class: PaymentGatewayIntegration**

- **Purpose:** Simulates sending booking details to an external payment processor.
- **Method:** processPayment(Id bookingId, Decimal amount).

- **Annotation:** @future(callout=true) - This is required for two reasons:
    1. It allows the method to make external callouts.
    2. It runs the logic asynchronously, which is required when a callout is initiated from a trigger.
  - **Logic:**
    1. Constructs an HttpRequest.
    2. Sets the endpoint using the Named Credential: callout:Payment\_API/post.
    3. Sets the method to POST and Content-Type to application/json.
    4. Builds a JSON body with the booking ID and amount.
    5. Uses Http.send(req) to send the request.
    6. Checks the response status code (expecting 200).
    7. If successful, performs a DML update on the original Booking\_\_c record to set Payment\_Status\_\_c to 'Paid'.
    8. Uses a try-catch block to handle System.CalloutException errors.





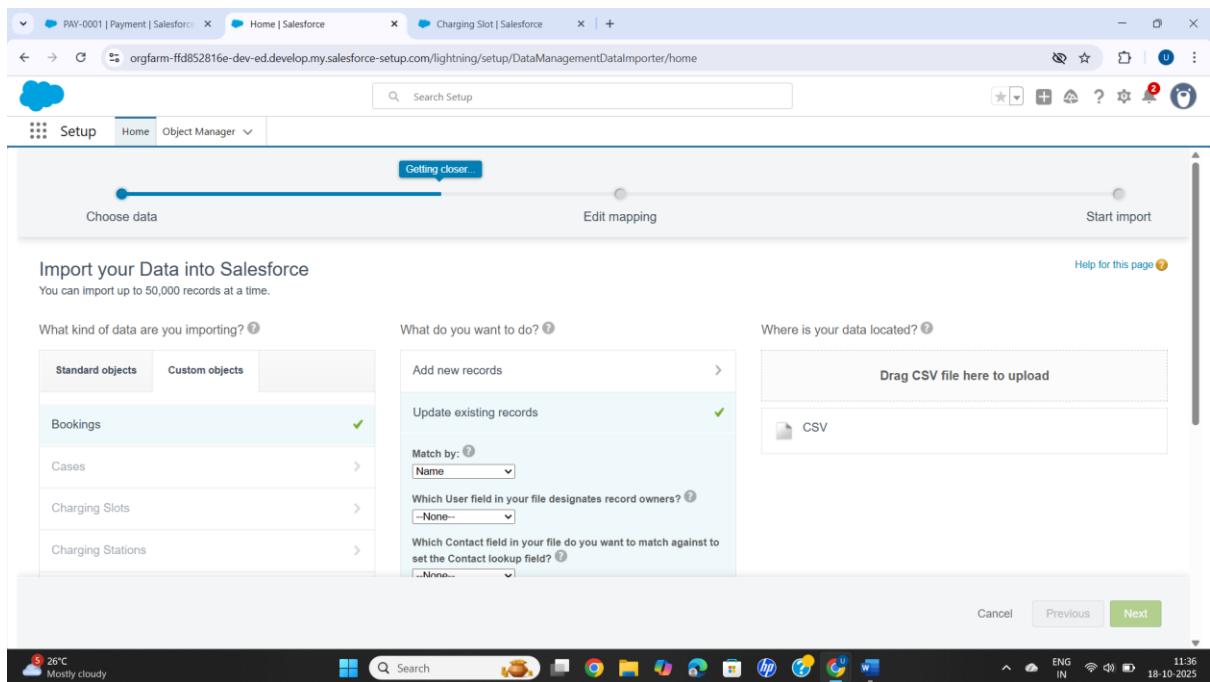
- **Trigger Handler Modification:**

- The BookingTriggerHandler.handleAfterUpdate method was modified.
- It checks if the Booking\_\_c status changed to 'Confirmed'. If it did, it calls PaymentGatewayIntegration.processPayment(...) to initiate the asynchronous callout.

## Phase 8: Data Management & Deployment

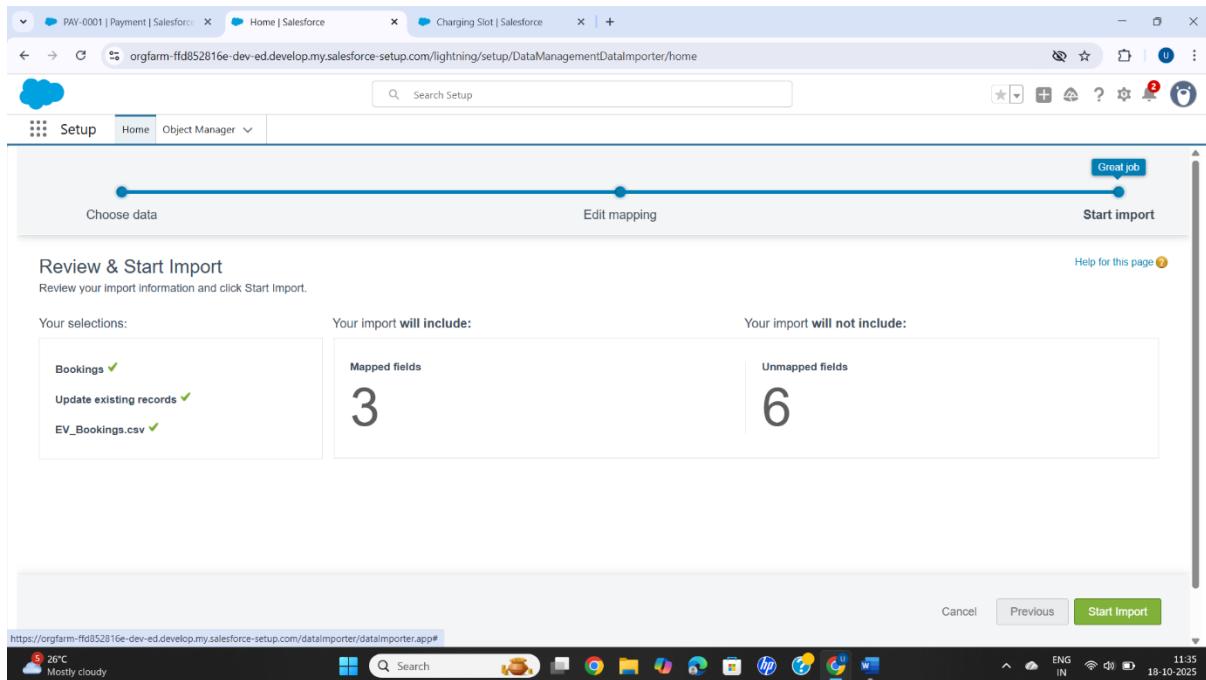
### Data Import Wizard (Initial Data Upload)

- **Use Case:** The Data Import Wizard was used for the initial upload of foundational Account and Contact records.
- **Process:**
  1. A CSV file (e.g., sample\_accounts.csv) was prepared.
  2. The wizard was launched via Setup.
  3. The 'Accounts' object and 'Add new records' action were selected.
  4. Fields were mapped, and the import was initiated.



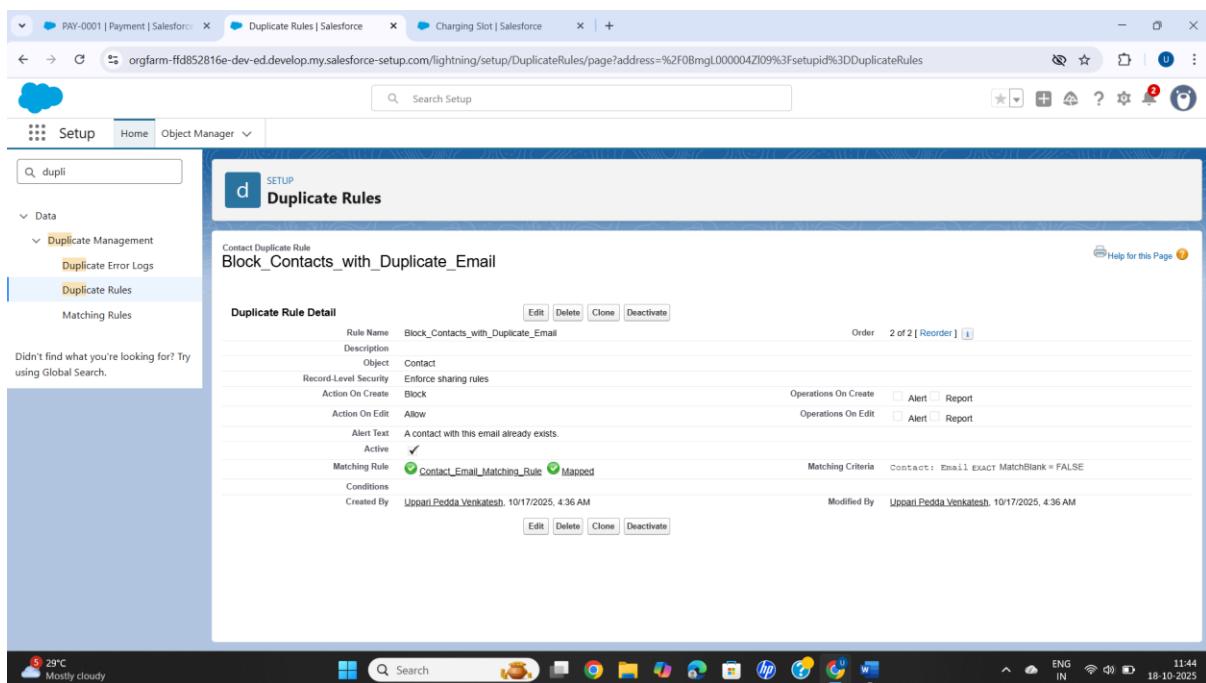
## Duplicate Rules

- **Purpose:** To maintain data quality by preventing users from creating duplicate Contact records based on email.
- **Setup:**
  1. **Matching Rule:** A rule named `Contact_Email_Matching_Rule` was created on the Contact object to find records where the Email field is an exact match.
  2. **Duplicate Rule:** A rule named `Block_Contacts_with_Duplicate_Email` was created.
  3. **Action:** The Action on Create was set to **Block**.
  4. **Alert:** An alert text "A contact with this email already exists" was configured.
  5. Both rules were activated.



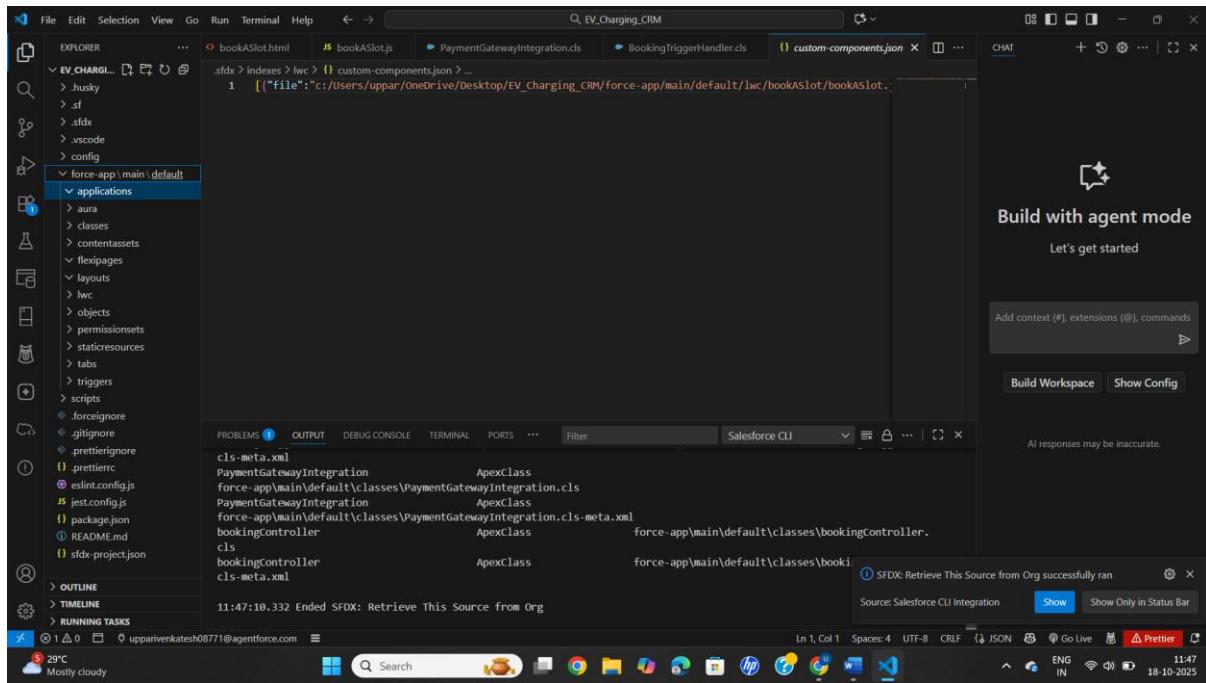
## Deployment Methods

- **SFDX (Salesforce Developer Experience via VS Code):**
  - This was the method used for deploying metadata (objects, fields, code, flows) from the developer org to a target org.

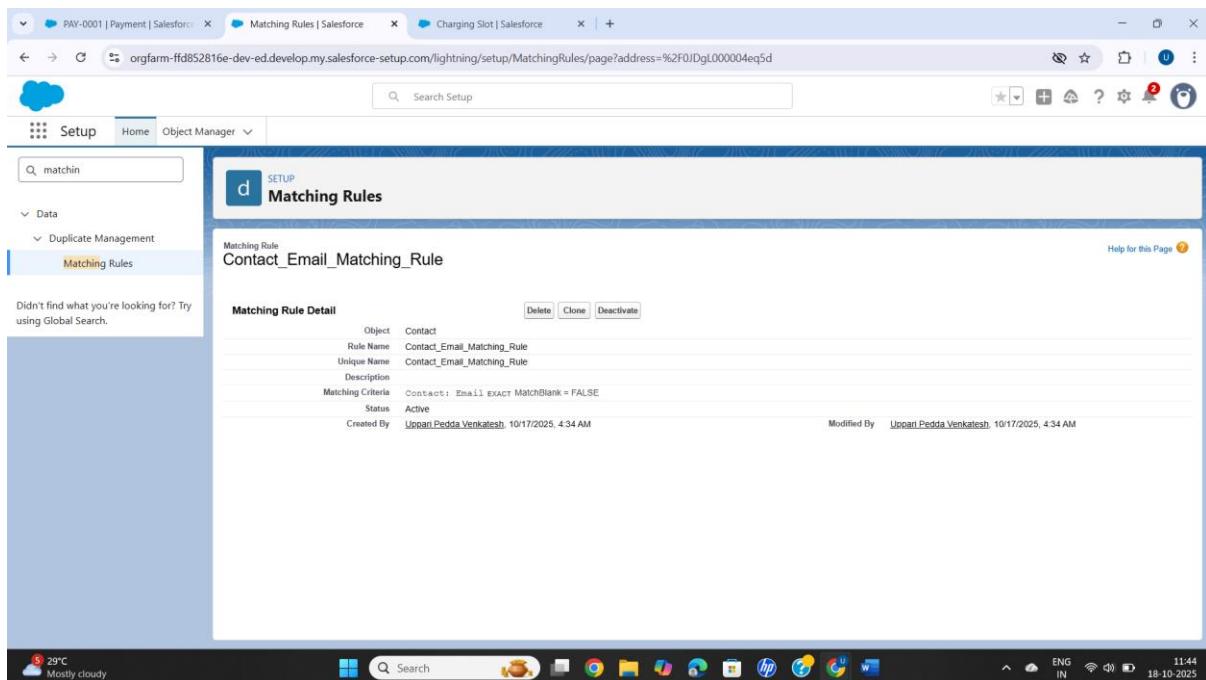


- **Process:**

1. **Authorize Target Org:** Connected VS Code to the destination org using SFDX: Authorize an Org.



2. **Retrieve (if needed):** Ensured the manifest/package.xml file included all components to be deployed.
3. **Deploy:** Right-clicked the project folder and selected SFDX: Deploy Source to Org to push the metadata.



## Phase 9: Reporting, Dashboards & Security Review

### Reports

The screenshot shows a Salesforce Lightning interface with the following details:

- Report Name:** New Bookings Report
- Total Records:** 1
- Total Total Price:** \$6,000
- Booking Details:**

Booking: Booking Number	Charging Station	Charging Slot	Start Time	Status	Total Price
BK-00012	-	-	10/17/2025, 12:00 PM	-	\$6,000

- **Daily Bookings:**

- A report on the Booking object, grouped by Created Date (by Calendar Day).
- **Purpose:** To track the volume of new bookings each day.

- **Revenue by Station:**

- A report on the Booking object, filtered for Status = 'Completed'.
- It is grouped by Station and includes a SUM on the Total\_Price\_c field.
- **Purpose:** To analyze revenue generation per station.

The screenshot shows a Salesforce Lightning interface with the following details:

- Report Name:** Slot Utilization by Station
- Total Records:** 3
- Station Data:**

Station	Availability	Available	Total
STN-0001	Record Count	1	2
STN-0002	Record Count	1	1
<b>Total</b>	<b>Record Count</b>	<b>2</b>	<b>3</b>
- Details View:** Shows a table for Charging Slot: Slot Number with four entries: SLOT-001, SLOT-003, SLOT-002, and SLOT-004.

- **Slot Utilization:**

- A report on the Slot object, grouped first by Station and then by Availability\_\_c.
- **Purpose:** To visualize the current status (e.g., Available, Unavailable) of slots across all stations.

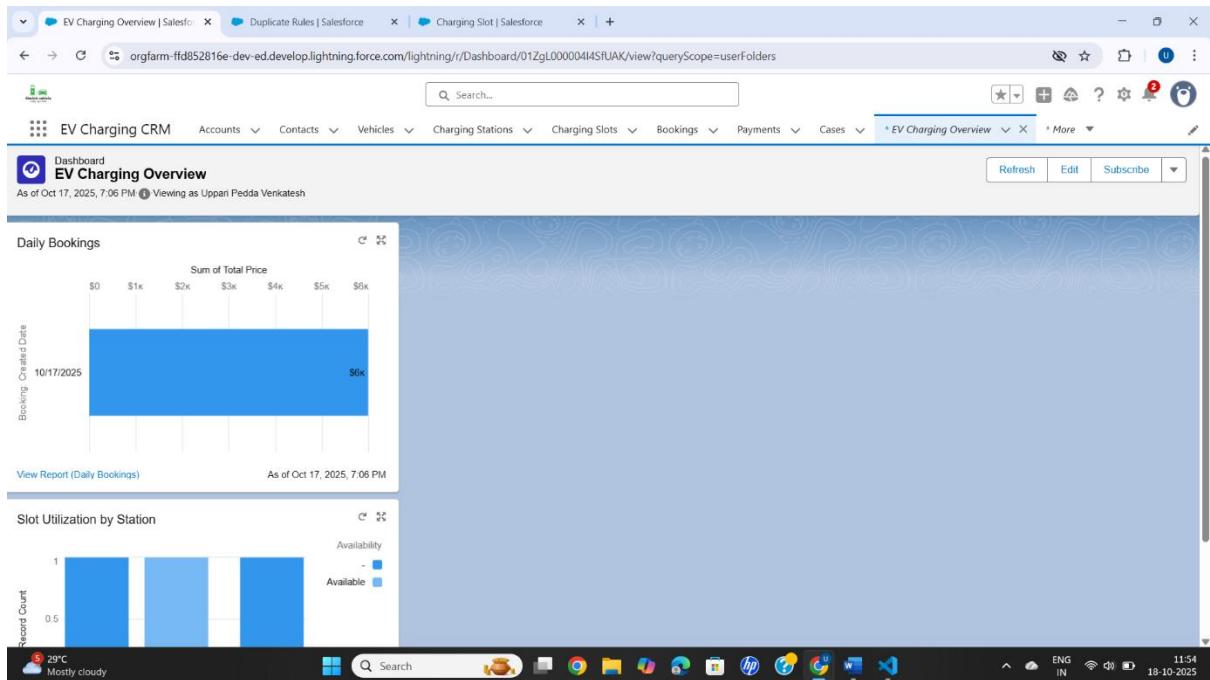
The screenshot shows the Salesforce Reports interface. The top navigation bar includes tabs for Recent, Reports, and Sales. Below the navigation is a search bar and a toolbar with various icons. The main content area displays a table titled 'Slot Utilization by Station'. The table has columns for Report Name, Description, Folder, Created By, Created On, and Subscribed. There are five items listed:

Report Name	Description	Folder	Created By	Created On	Subscribed
New Bookings Report		Private Reports	Uppari Pedda Venkatesh	10/17/2025, 5:28 AM	
Slot Utilization by Station		Private Reports	Uppari Pedda Venkatesh	10/17/2025, 5:35 AM	
Revenue by Station		Private Reports	Uppari Pedda Venkatesh	10/17/2025, 5:32 AM	
Daily Bookings		Public Reports	Uppari Pedda Venkatesh	10/17/2025, 5:30 AM	
Sample Flow Report: Screen Flows	Which flows run, what's the status of each interview, and how long do users take to complete the screens?	Public Reports	Automated Process	10/5/2025, 7:57 PM	

The left sidebar contains navigation links for Reports, Folders, and Favorites. The bottom of the screen shows the Windows taskbar with various pinned apps and system status indicators.

## Dashboard

- **Dashboard Name:** EV Charging Overview.
- **Purpose:** To provide a high-level, at-a-glance view of key performance indicators.
- **Components:**
  - Total Bookings Today:** A Metric component using the Daily Bookings report (filtered for Today).
  - Revenue by Station:** A Bar Chart component using the Revenue by Station report.
  - Overall Slot Utilization:** A Gauge or Chart component using the Slot Utilization report to show the ratio of available vs. unavailable slots.



## Security Review

A layered security model was configured:

- **Organization-Wide Defaults (OWD):**
  - **Booking:** Private
  - **Payment:** Private
  - **Station:** Public Read Only
  - **Slot:** Controlled by Parent (inherits read-only access from Station via the Master-Detail relationship).
- **Role Hierarchy:**
  - A vertical hierarchy (CEO -> Operations Manager -> Station Manager) was built to allow managers to view their subordinates' records.
- **Sharing Rules:**
  - **Use Case:** A criteria-based sharing rule named "Share Bookings within Station" was created.
  - **Purpose:** To allow all users in the "Station Staff" role to see all other bookings for their *same* station, even though the OWD for Booking is Private.

The screenshot shows the 'Sharing Settings' page in the Salesforce Setup. The left sidebar has a search bar and navigation links for Security, Guest User Sharing Rule Access, Report, and Sharing Settings. A note says 'Didn't find what you're looking for? Try using Global Search.' The main area displays sections for 'Booking Sharing Rules', 'Case Sharing Rules', 'Charging Slot Sharing Rules', 'Charging Station Sharing Rules', 'Payment Sharing Rules', and 'Vehicle Sharing Rules'. Each section has 'New' and 'Recalculate' buttons. The 'Charging Slot Sharing Rules' section includes an 'Action' column with 'Edit | Det' and a 'Criteria' row: '(Charging Slot: Total slots EQUALS 100) AND (Charging Slot: Slot Number CONTAINS pending)'. The 'Shared With' column lists 'Role\_Station Staff' and the 'Access Level' is 'Read Only'. The bottom status bar shows weather (26°C, Partly sunny), system icons, and the date/time (10/10/2025, 10:42 AM).

- **Access Level:** Read Only.
- **Profiles & Permission Sets:**
  - **Profiles:** A custom profile Station Staff was cloned from Standard User to define base-level object permissions (e.g., Read/Create/Edit on Bookings).
  - **Permission Sets:** Used to grant *additional* access. For example, the Manage Payments permission set grants Edit/Delete on the Payment object to specific users without changing their profile.

The screenshot shows the 'Activations' page in the Salesforce Setup. The left sidebar includes 'Quick Find' and links for Setup Home, Salesforce Go, Service Setup Assistant, Commerce Setup Assistant, Hyperforce Assistant, Release Updates, Salesforce Mobile App, Lightning Usage, Optimizer, Sales Cloud Everywhere, Administration (Users, Data, Email), and Platform Tools (Subscription Management, Apps, Feature Settings). The main area has a 'Login IP' section with a table showing logins for user 'uparvenkatesh08771@agentforce.com' with IP addresses 117.213.146.33, 139.167.130.38, 117.192.50.39, and 157.50.79.126. The 'Activated Client Browsers' section shows browser agent information for the same user with entries for Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36 and Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36. The bottom status bar shows weather (29°C, Mostly cloudy), system icons, and the date/time (10/10/2025, 12:02 PM).

- **Field-Level Security (FLS):**

- Used to restrict access to specific fields. For example, the Hourly\_Rate\_\_c field on the Slot object was made Read-Only for the Station Staff profile.

The screenshot shows the Salesforce Setup interface with the 'Manage Payments' permission set selected. The left sidebar shows the navigation menu under 'Users'. The main area displays the 'Current Assignments' table, which contains one row for 'Uppari Pedda Venkatesh' assigned to the 'System Administrator' profile with the 'Salesforce' user license. The table has columns for Full Name, Active, Role, Profile, User License, and Expires On.

Full Name	Active	Role	Profile	User License	Expires On
Uppari Pedda Venkatesh	✓		System Administrator	Salesforce	

## Quality Assurance Testing:

Test cases were prepared for each feature to ensure functionality aligns with requirements.

### Test Case 1: Validation Rule

- **Use Case / Scenario:** Prevent Booking End Time Before Start Time.
- **Test Steps:**
  1. Navigate to the Booking tab and click "New".
  2. Select a Contact and a Slot.
  3. Enter Start\_Time\_\_c = 10/20/2025, 10:00 AM.
  4. Enter End\_Time\_\_c = 10/20/2025, 9:00 AM (i.e., before the start time).
  5. Click "Save".
- **Expected Result:** A validation error message appears: "Booking End Time cannot be earlier than the Start Time.". The record is not saved.

- **Actual Result:** (Screenshot) As expected, the error message appeared at the top of the page, and the save was prevented.

### Test Case 2: Record-Triggered Flow

- **Use Case / Scenario:** Slot availability updates to 'Unavailable' when a Booking is 'Confirmed'.
- **Test Steps:**
  1. Create a new Charging\_Slot\_\_c record. Verify its Availability\_\_c is 'Available'.
  2. Create a new Booking\_\_c record associated with the Slot from Step 1.
  3. Set the Status\_\_c field to 'Confirmed'.
  4. Click "Save".
- **Expected Result:** The flow triggers. The related Charging\_Slot\_\_c record's Availability\_\_c field is automatically updated from 'Available' to 'Unavailable'.
- **Actual Result:** (Screenshot) As expected, the flow ran, and the Slot's availability was correctly set to 'Unavailable'.

### Test Case 3: Apex Callout (Asynchronous)

- **Use Case / Scenario:** Payment Status updates to 'Paid' after a Booking is confirmed.
- **Test Steps:**
  1. Create a new Booking\_\_c record with Status\_\_c = 'Pending' and Payment\_Status\_\_c = 'Unpaid'.
  2. Update the Booking\_\_c record's Status\_\_c from 'Pending' to 'Confirmed'.
  3. Click "Save".
  4. Navigate to Setup -> Apex Jobs to monitor the future method.
- **Expected Result:** The BookingTrigger fires, and the PaymentGatewayIntegration future method is queued and runs

successfully. The Booking\_\_c record's Payment\_Status\_\_c field is updated from 'Unpaid' to 'Paid'.

- **Actual Result:** (Screenshot) As expected, the Apex job completed, and the Payment\_Status\_\_c field on the Booking record was successfully updated to 'Paid'.
- 

## Conclusion

This project successfully delivered a comprehensive EV Charging CRM on the Salesforce platform. It addresses all core business requirements, from customer-facing booking and payment processing to internal station management and security. The application provides a scalable foundation for managing charging operations, automating key processes like payment callouts and reminders , and delivering critical business insights through reports and dashboards. The use of platform-native tools and a logic-less trigger framework ensures the system is both robust and maintainable for future enhancements.