Phase 6: User Interface Development

The user interface provides access to the custom objects and functionality built for the EV Charging CRM. This was primarily achieved using the declarative Lightning App Builder and a custom Lightning Web Component (LWC) for dynamic booking functionality.

Lightning App Builder: Custom App & Home Page

A dedicated Lightning App was created to provide a focused experience for users managing EV charging operations.
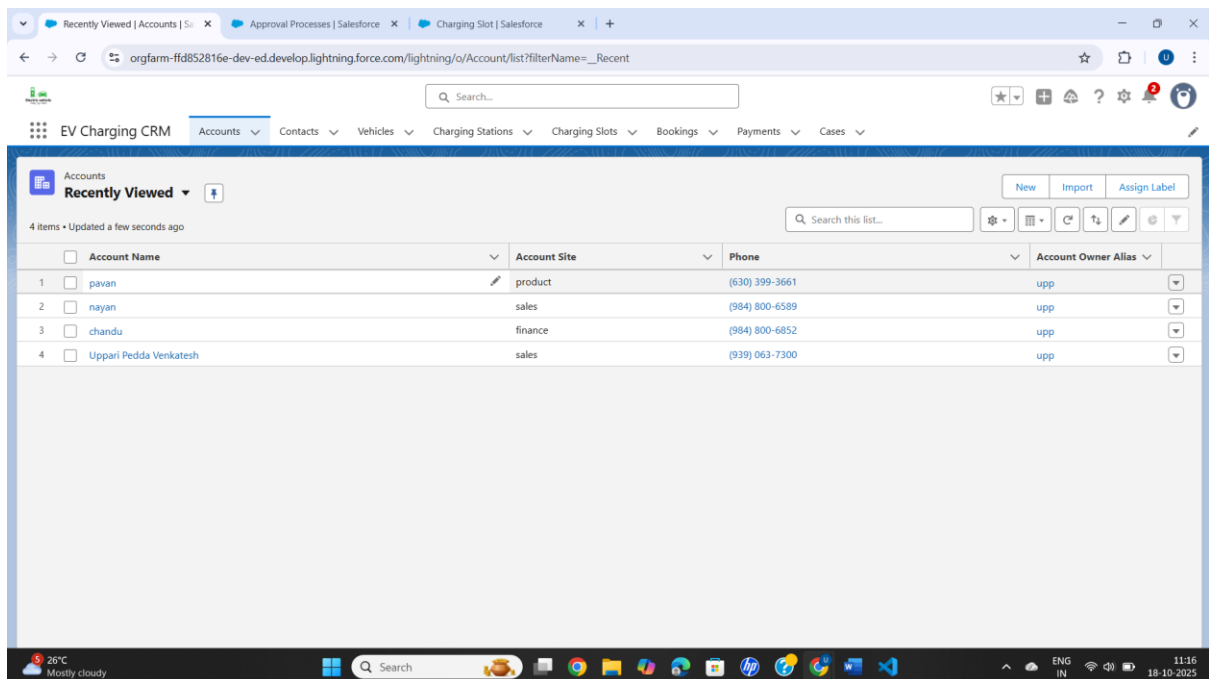
1. **Custom App: "EV Charging CRM"**

   o **Creation:** Created via **Setup -> App Manager -> New Lightning App**.

   o **Configuration:**

      ▪ **App Name:** EV Charging CRM

      ▪ **Navigation Style:** Standard Navigation

      ▪ **Tabs Added:** Home, Accounts, Contacts, Vehicles, Charging Stations, Charging Slots, Bookings, Payments, Cases, Invoices (and others as needed). Tabs were ordered logically based on the business process flow (e.g., Accounts/Contacts first, then Stations/Slots, followed by Bookings/Payments).

      ▪ **User Profiles:** Assigned to relevant profiles (System Administrator, Standard User, potentially custom profiles like Station Manager).

   o **Purpose:** Provides a central hub for users to access all EV charging-related objects and features.

2. **Custom Home Page: "EV Charging Home Page"**

   o **Creation:** Created via **Setup -> User Interface -> Lightning App Builder -> New -> Home Page**.

   o **Template:** Standard Home Page template selected.

- o **Components Added:**

  - ▪ **Dashboard Component:** Added to the main region. Configured to display the EV Charging Overview dashboard created in Phase 9, showing key KPIs.

  - ▪ **Rich Text Component:** Added to the sidebar region. Configured with "Quick Links" (using the link tool) pointing to frequently used actions, such as the 'New Booking' page or specific list views.

  - o **Activation:** Activated and assigned as the **App Default** specifically for the "EV Charging CRM" app.

  - o **Purpose:** Provides users with an immediate overview of key metrics and quick access to common tasks upon entering the EV Charging CRM app.

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

A custom LWC was developed to handle the dynamic process of selecting a station and then viewing/selecting only the available slots for that specific station.

- **Component Name:** bookASlot

- **Purpose:** Provides an intuitive interface for users to create new Booking__c records.

- **Key Features:**

  - **Dynamic Slot Fetching:** Uses two lightning-combobox elements. The first ("Select a Station") is populated on load using a @wire call to the bookingController.getStations() Apex method. When a station is selected, the handleStationChange JavaScript function calls the bookingController.getAvailableSlots(stationId) Apex method *imperatively* to fetch only the slots marked 'Available' for the selected station. The second combobox ("Select an Available Slot") is then populated with these results and enabled.

  - **User Input:** Includes a lightning-input of type datetime for selecting the booking start time.

  - **Record Creation:** Uses the createRecord wire adapter from lightning/uiRecordApi in the handleBookNow JavaScript function to create the new Booking__c record with the selected Station (via Slot), Slot, Start Time, and a default Status of 'Confirmed'.

  - **User Feedback:** Includes a lightning-spinner for loading states and uses ShowToastEvent to display success or error messages to the user.

- **Placement:** The LWC is made available for lightning__AppPage and lightning__HomePage targets (defined in bookASlot.js-meta.xml) and was added to the "EV Charging Home Page" using the Lightning App Builder.

Screenshot 1 — bookASlot.js

```
1  import { LightningElement, wire, track } from 'lwc';
2  import { ShowToastEvent } from 'lightning/platformShowToastEvent';
3
4  // 1. Import Apex methods
5  import getStations from '@salesforce/apex/bookingController.getStations';
6  import getAvailableSlots from '@salesforce/apex/bookingController.getAvailableSlots';
7
8  // 2. Import schema for creating the record
9  import { createRecord } from 'lightning/uiRecordApi';
10 import BOOKING_OBJECT 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 comboboxes
```

Salesforce metadata XML IntelliSense is now available.

Screenshot 2 — bookASlot.js-meta.xml

```
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>
```

Salesforce metadata XML IntelliSense is now available.

Screenshot 3 — bookASlot.html

```
1   <template>
2       <lightning-card title="Book a Charging Slot" icon-name="custom:custom14">
3           <div class="slds-m-around_medium">
29              <lightning-input
34                  required>
35              </lightning-input>
36
37              <lightning-button
38                  label="Book Now"
39                  variant="brand"
40                  onclick={handleBookNow}
41                  disabled={isBookDisabled}
42                  class="slds-m-top_medium">
43              </lightning-button>
44          </div>
45      </lightning-card>
46  </template>
```

Salesforce metadata XML IntelliSense is now available.