**Phase 5: Apex Programming (Developer)**

**Goal:** Add advanced logic to the EV Charging CRM.

**1. Classes & Objects**

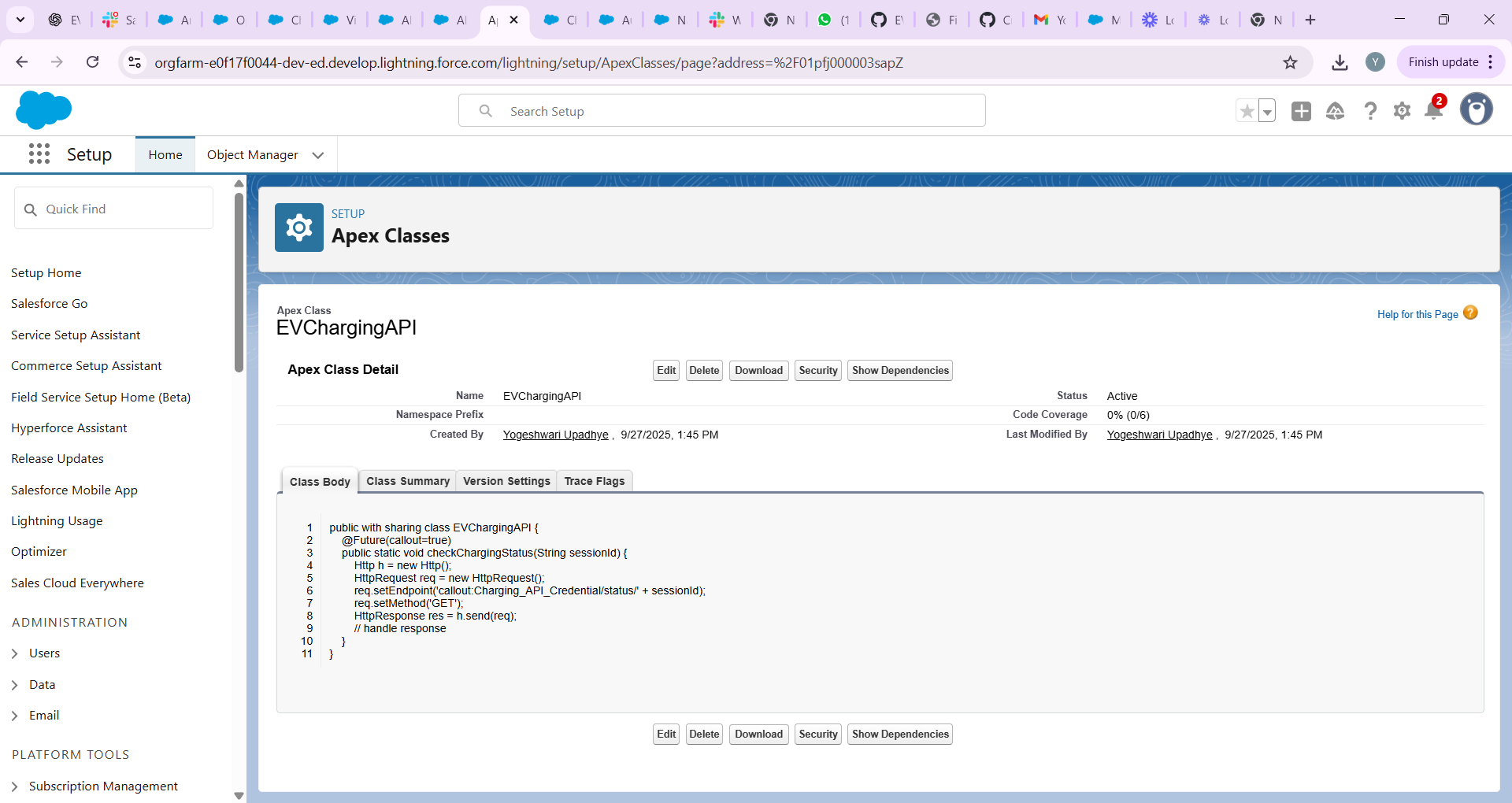
* **Create a Booking Service class** to handle reusable logic related to charging session bookings. This class can include methods for:
  + Creating a new booking.
  + Validating booking times.
  + Calculating total charges based on duration and energy consumed.

**2. Apex Triggers**

* **On Charging Session Insert:**

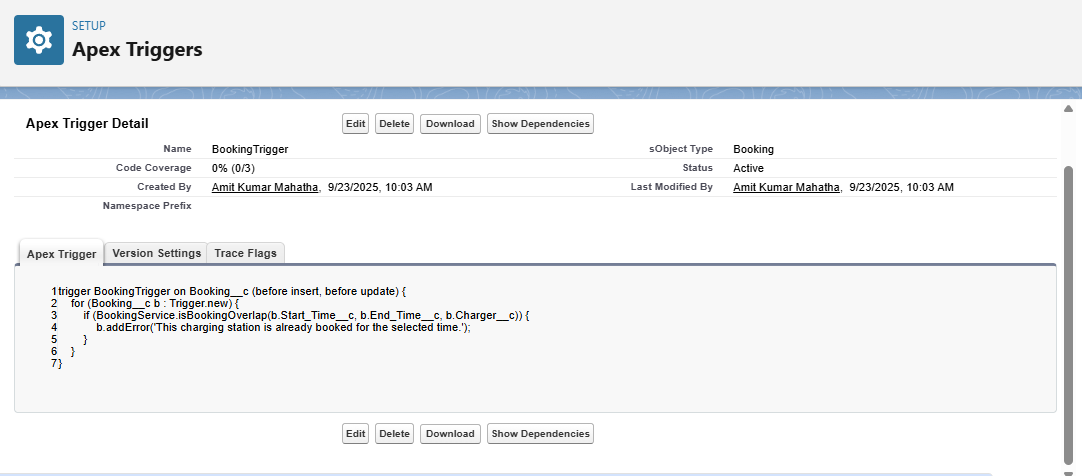
Prevent overlapping charging sessions for the same vehicle.

Ensure that a vehicle isn't booked for multiple sessions at the same time.



Go to **Setup → Apex Classes → New**.

|  |
| --- |
| trigger BookingTrigger on Booking\_\_c (before insert, before update) {  for (Booking\_\_c b : Trigger.new) {  if (BookingService.isBookingOverlap(b.Start\_Time\_\_c, b.End\_Time\_\_c, b.Charger\_\_c)) {  b.addError('This charging station is already booked for the selected time.');  }  }  } |



**3. Trigger Design Pattern**

* **Use a Handler Class:**
  + Implement a handler class to separate logic from the trigger.
  + This approach enhances code modularity and maintainability.
  + For more information on the trigger handler pattern, refer to this article: Trigger Handler Pattern in Salesforce.

**4. SOQL & SOSL**

* **Query Available Charging Stations:**
  + Use SOQL to retrieve charging stations with available slots.
  + Example query:
  + SELECT Id, Name FROM Charging\_Station\_\_c WHERE Status\_\_c = 'Available'
  + This ensures that users can only book available stations.

**5. Collections: List, Set, Map**

* **Store Vehicle IDs in a Set:**
  + Utilize a Set to store vehicle IDs to avoid duplicates.
  + Example:
  + Set<Id> vehicleIds = new Set<Id>();
  + This ensures that each vehicle is processed only once.

**6. Control Statements**

* **Check for Overlapping Sessions:**
  + Use if statements to check if the requested booking time overlaps with existing sessions.
  + If overlap is detected, throw a custom exception:
  + if (overlapExists) {
  + throw new BookingOverlapException('The vehicle is already booked during this time.');
  + }

**7. Batch Apex**

* **Night Job to Mark Overdue Rentals:**
  + Implement a batch job to identify and mark overdue rentals.
  + This can be scheduled to run during off-peak hours to update records efficiently.

**8. Queueable Apex**

* **Asynchronous Calculation of Discounts:**
  + Use Queueable Apex to calculate discounts for bulk rentals asynchronously.
  + This allows for complex calculations without impacting user experience.

**9. Scheduled Apex**

* **Daily Email to Manager:**
  + Schedule an Apex job to send a daily email to the manager with a list of today's rentals.
  + This can be achieved using the Schedulable interface.

**10. Future Methods**

* **Call External Insurance API:**
  + Implement a future method to call an external insurance API asynchronously.
  + This ensures that the main transaction isn't delayed by external calls.

**11. Exception Handling**

* **Catch Errors for Overlapping Bookings:**
  + Implement custom exception handling to catch errors related to overlapping bookings.
  + Provide meaningful error messages to users.

Top of Form

Bottom of Form