• Smart Event Management & Ticketing System

• Problem Statement

Event organizers face multiple challenges:
- Ticket sales are fragmented across different channels (online, offline, sponsors).
- No centralized system to manage seat allocation, VIP access, cancellations, and refunds.
- Sponsors don't get proper visibility into performance .
- Attendees lack a smooth digital experience (QR tickets, real-time updates, group discounts).

To solve this, we want a system to:
- Centralize event, ticket, and sponsor management.
- Automate approval workflows for VIP tickets and cancellations.
- Enable digital payments and QR-based entry passes.
- Provide real-time dashboards for ticket sales, sponsorship revenue, and attendee engagement.

Phase 1: Problem Understanding & Industry Analysis

1 Requirement Gathering

- Capture event details (type, date, location, capacity).
- Manage ticket categories (General, VIP, Early Bird, Group Discounts).
- Automate seat allocation and group bookings.
- Track sponsorship deals and benefits.
- Send digital tickets/QR codes to attendees.
- Enable refunds and cancellations with approval workflows.
- Provide dashboards for ticket sales, revenue, and attendance.

2 Stakeholder Analysis

- Event Organizers: Need event setup, sponsor management, ticket monitoring.
- Attendees: Need smooth booking, payments, QR entry, and refund options.
- Sponsors: Need visibility into event reach audience size, and engagement.
- Administrators (CRM Managers): Ensure secure access, data integrity, and reporting.

3 Business Process Mapping

Current:
- Tickets sold via third-party sites or offline.
- Manual sponsor agreement tracking.
- Attendees face delays in receiving confirmation.
- No unified dashboard for revenue analysis.

Proposed:
- Centralized Event, Ticket, Sponsor objects.
- Automated VIP approvals and refund workflows.
- QR-code ticketing + calendar sync for attendees.
- Dashboards for organizers & sponsors to track revenue, seats, and engagement.

4 Industry-specific Use Case Analysis

- Concerts & Music Festivals VIP, General, Early Bird ticketing.
- Corporate Conferences Multi-track event scheduling, sponsor branding.
- Sports Events Seat allocation and bulk ticket sales.
- Weddings & Private Events Guest management with QR invitations.

5 AppExchange Exploration

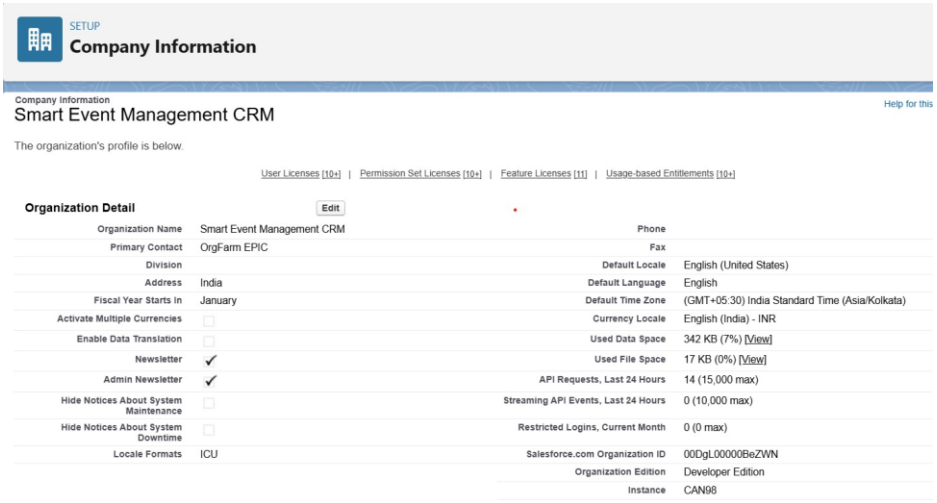
- Explore apps for payment gateways (Razorpay).
- Explore apps for QR code generation.
- Check Event Management accelerators available on AppExchange and customize instead of reinventing everything.

Phase 2: Org Setup & Configuration

- This phase focused on setting up and configuring the Salesforce Developer Org for the Smart Event Management Project. The goal was to establish a solid foundation for the CRM system, ensuring proper organizational structure, security, and access controls. The configuration was carried out entirely using Salesforce's declarative (point-and-click) tools without coding.

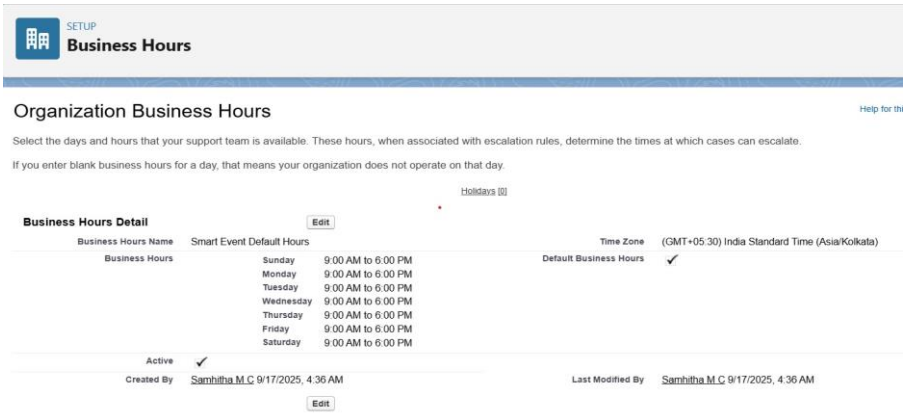
Steps Completed in Phase 2:

1. Company Profile Setup

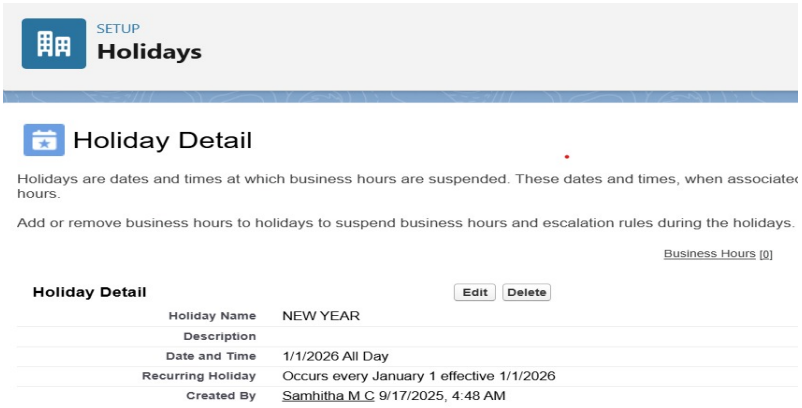


- Configured company information, business hours, and holidays.

2. Business Hours

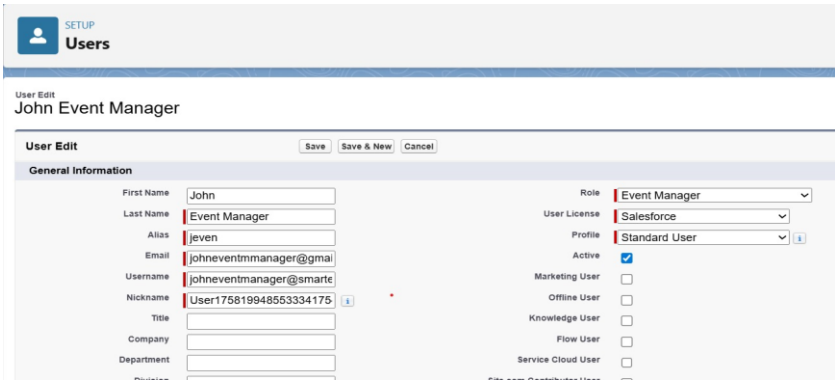


3. Holidays

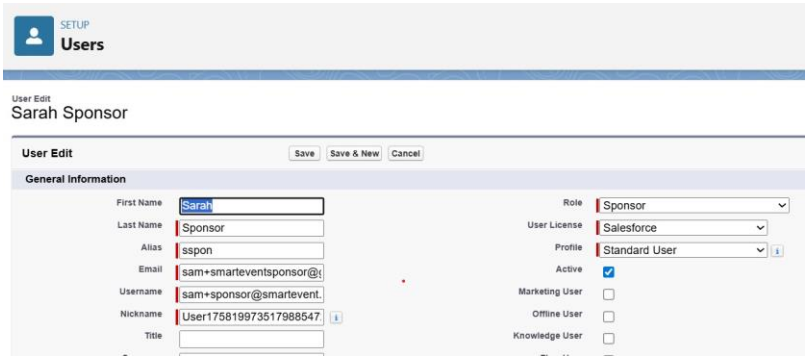


4. User Setup

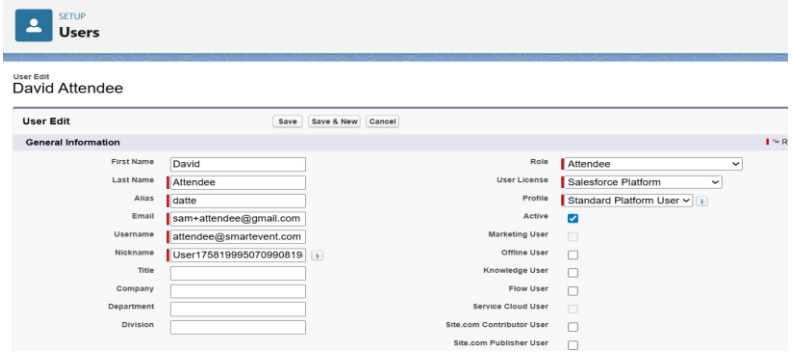
- Created multiple users with different roles (Event Manager, Sponsor, Attendee).



- Event Manager



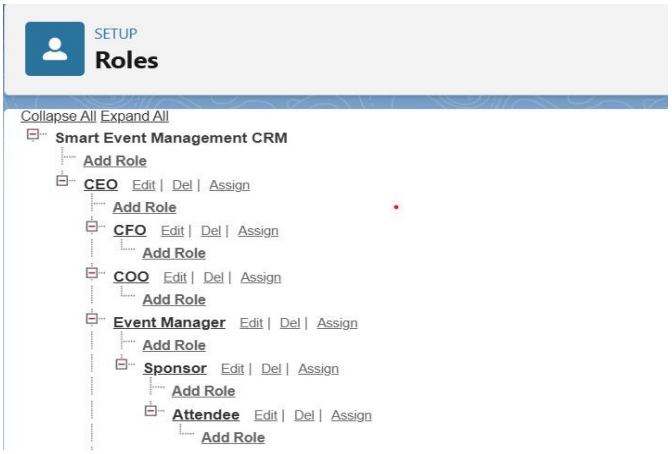
- Sponsor



- Attendee

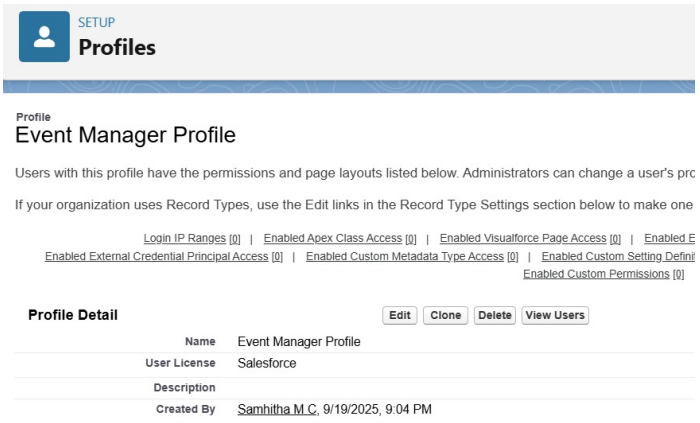
5. Roles

- Defined role hierarchy to manage data visibility across the organization.

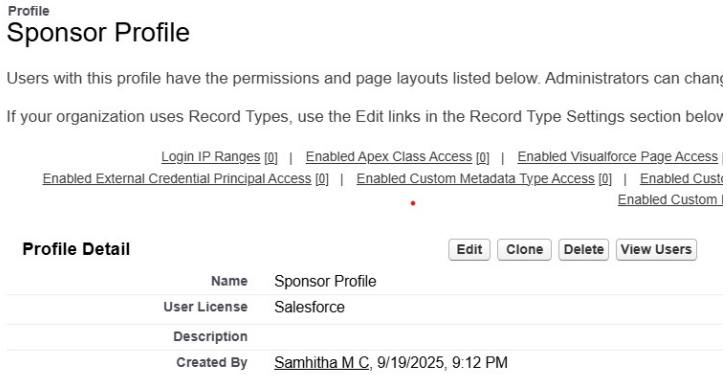


6. Profiles

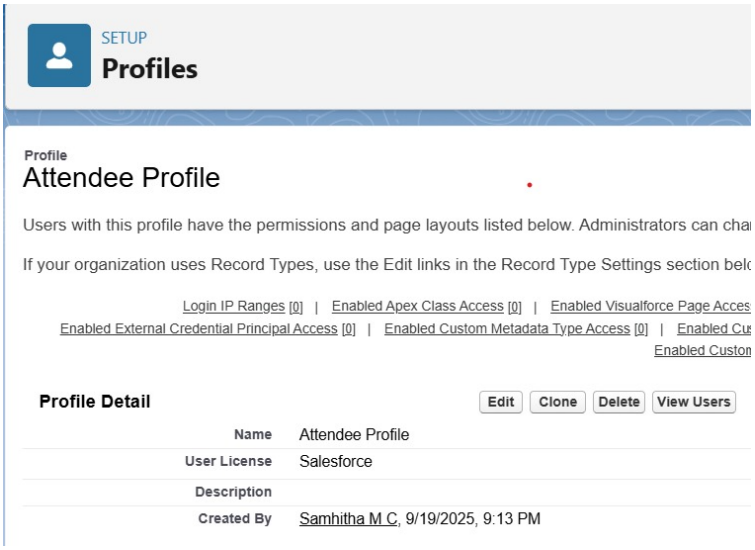
- Created and customized profiles by cloning standard profiles and adjusting object-level permissions.



- Event Manager Profile



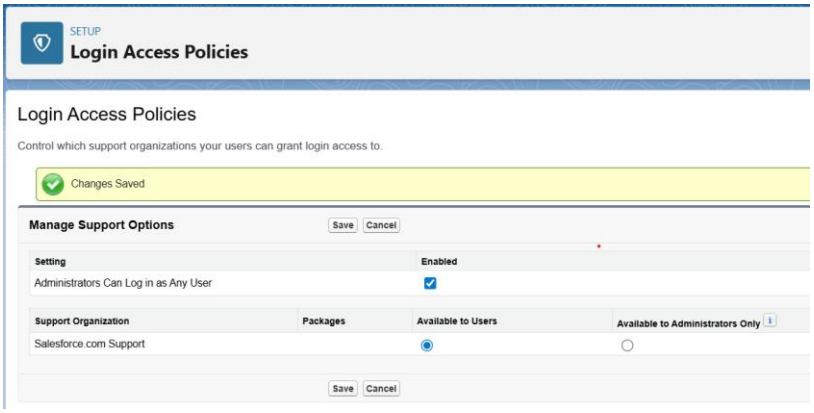
- Sponsor Profile



- Attendee Profile

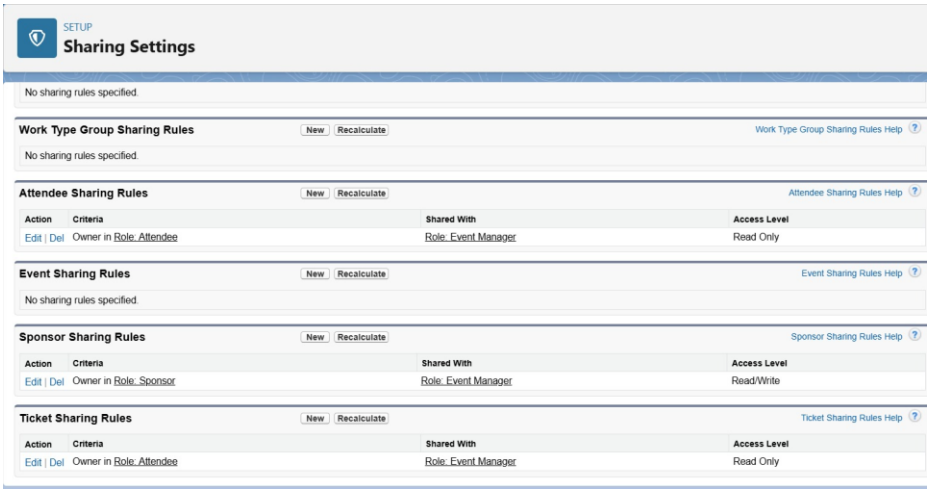
7. Login Access Policies

- Enabled admin login access for troubleshooting and testing.



8. Sharing Rules

- Created exceptions to OWD (e.g., Event Manager access to Attendees, Sponsors, and Tickets).



9. Organization wide Defaults(OWD)

- Configured baseline record access (Event = Public Read Only, Attendee/Sponsor/Ticket = Private).

Attendee	Private	Private	✓
Event	Public Read Only	Public Read Only	✓
Sponsor	Private	Private	✓

10. Permission Sets

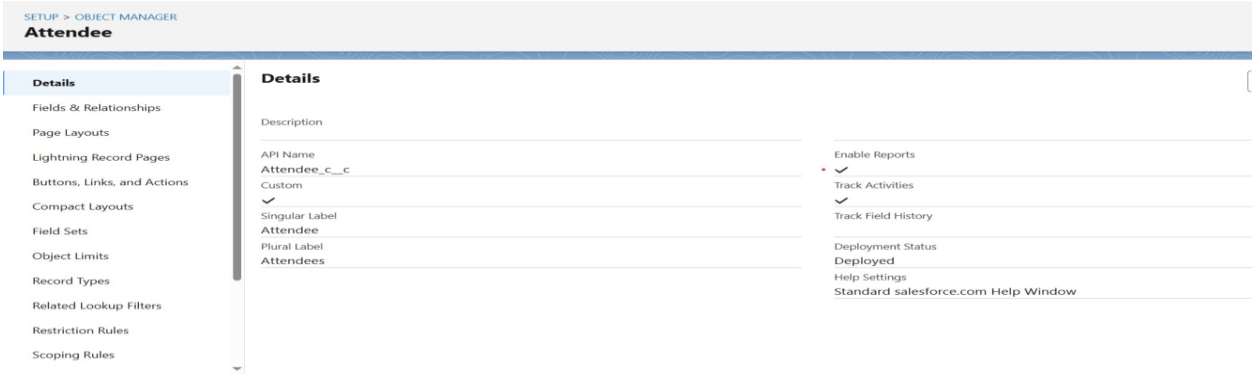
- Assigned extra permissions without modifying base profiles.

Phase 3: Data Modeling and Relationships

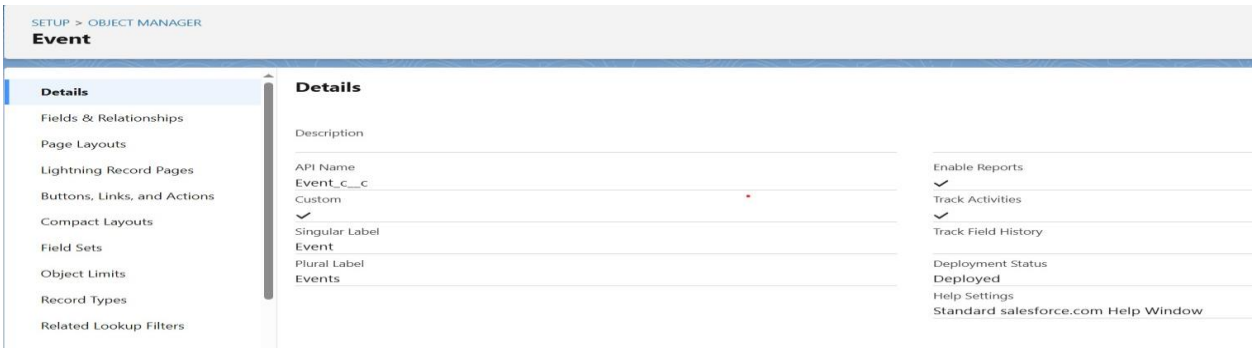
Goal: Build the core data structure to manage events, sponsors, attendee and tickets.

1. Standard and Custom Objects
    - Account: To store information.
    - Contact: To store contact details.

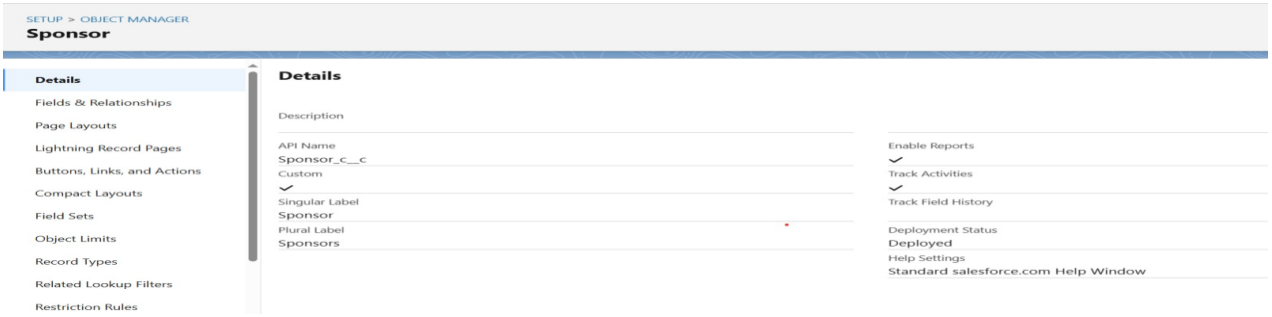
Custom Object:



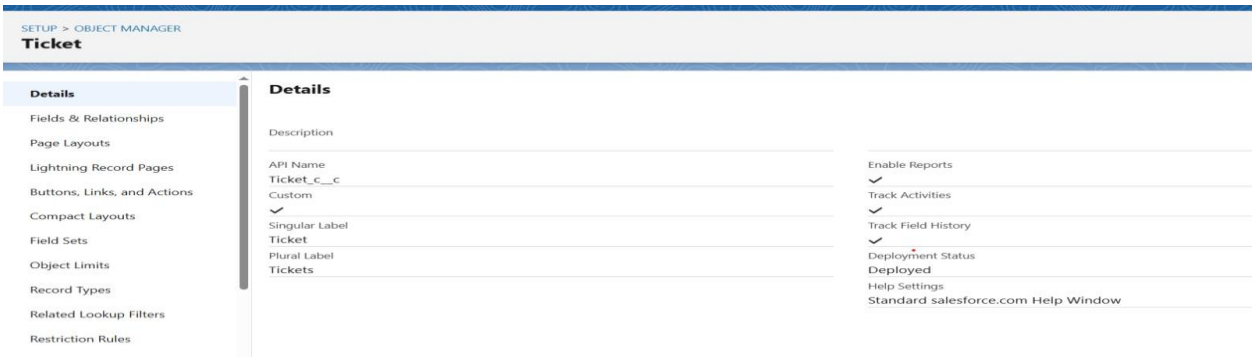
i. Attendee



ii. Event



iii. Sponsor



iv. Ticket

2. Fields

Event_Status__c

- Purpose: Tracks the current state of the event lifecycle.
- Values: *Planning, Registration Open, Active (In Progress), Completed, Canceled.* (This field is crucial for automation and reporting.)

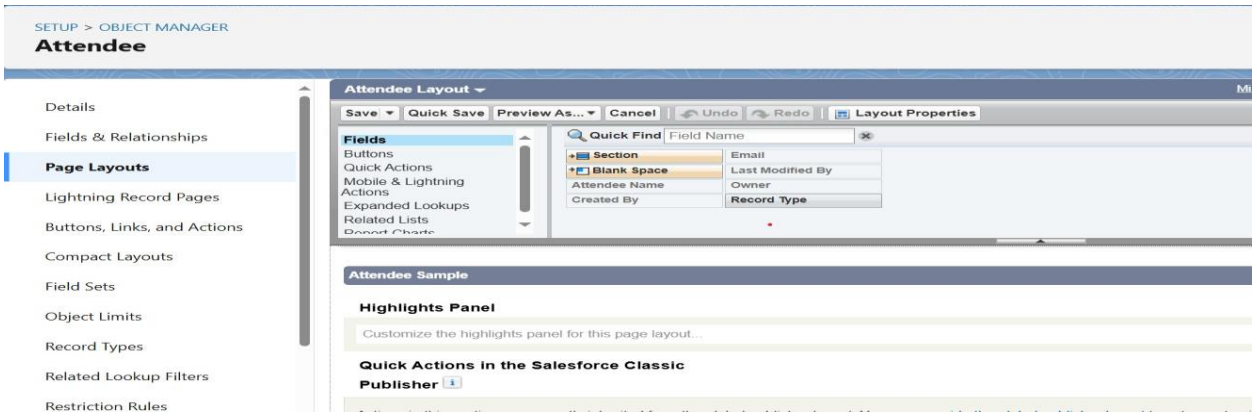
Ticket_Type__c

- Purpose: Tracks the specific access level purchased or assigned.
- Values: *General Admission, VIP, Speaker, Sponsor, Employee.* (Affects Page Layout visibility for related details, like seating assignment.)

3. Record Types

    - Record Types allow you to offer different business processes values, and page layouts to different users based on their profile. They are often used on Objects like Lead, Opportunity, or Case, and custom objects.

4. Page Layout

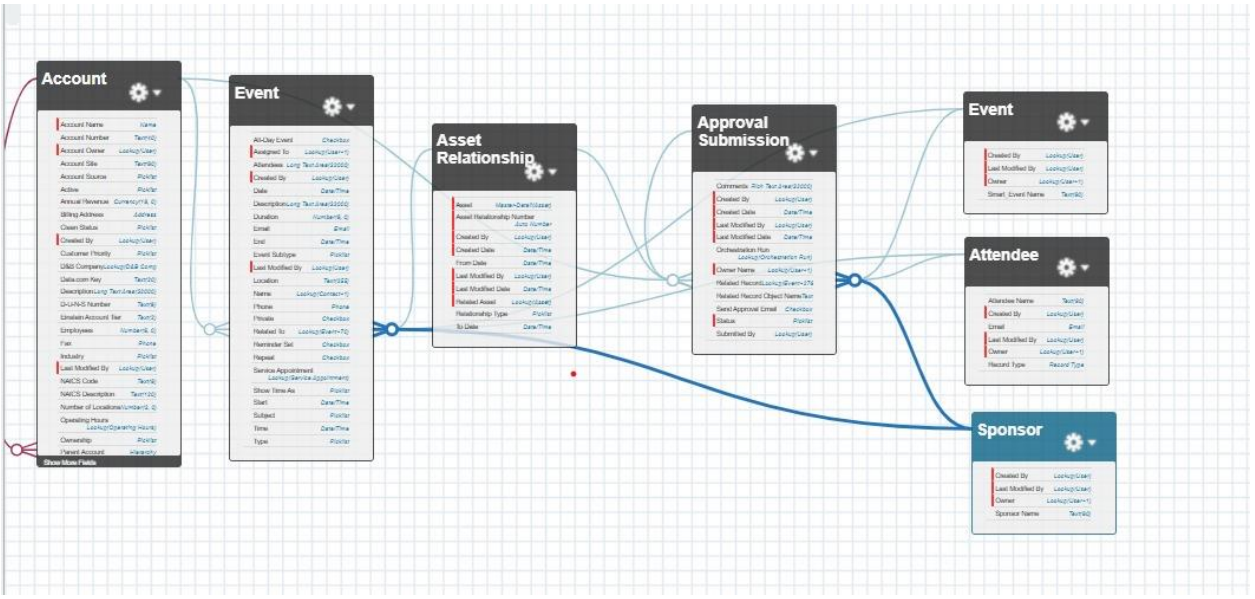


    - Page Layouts are assigned to specific **Profiles** and **Record Types**.

5. Custom Layout

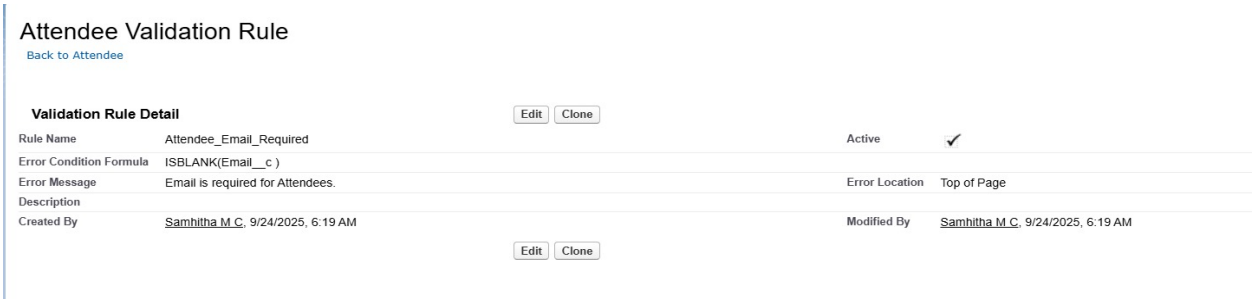
    - The Project compact layout is configured to show the project name, event manager name, sponsors and tickets in list.

6. Schema builder



Phase 4: Process Automation (Admin)

7. Validation Rules:

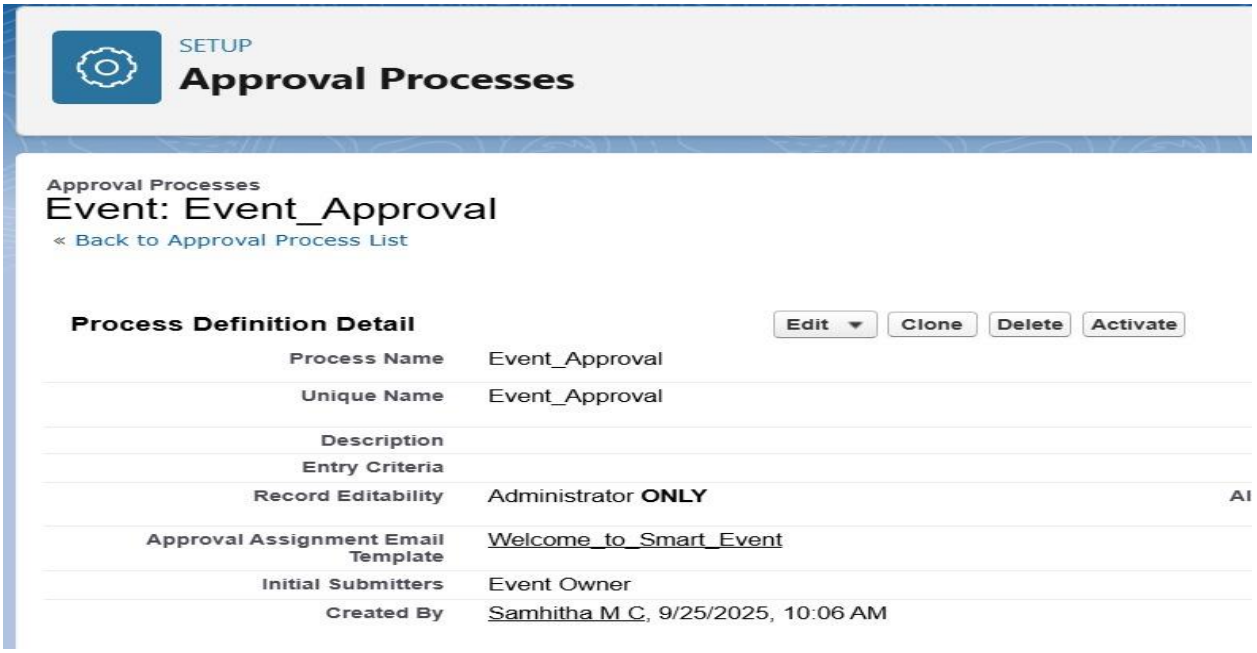


- A validation rule has been created on the attendee. It prevents user from blank email Id.

8. Workflow Rules:
    - This is a legacy automation tool. All new automations for this project are being built in Flow builder for better performance.

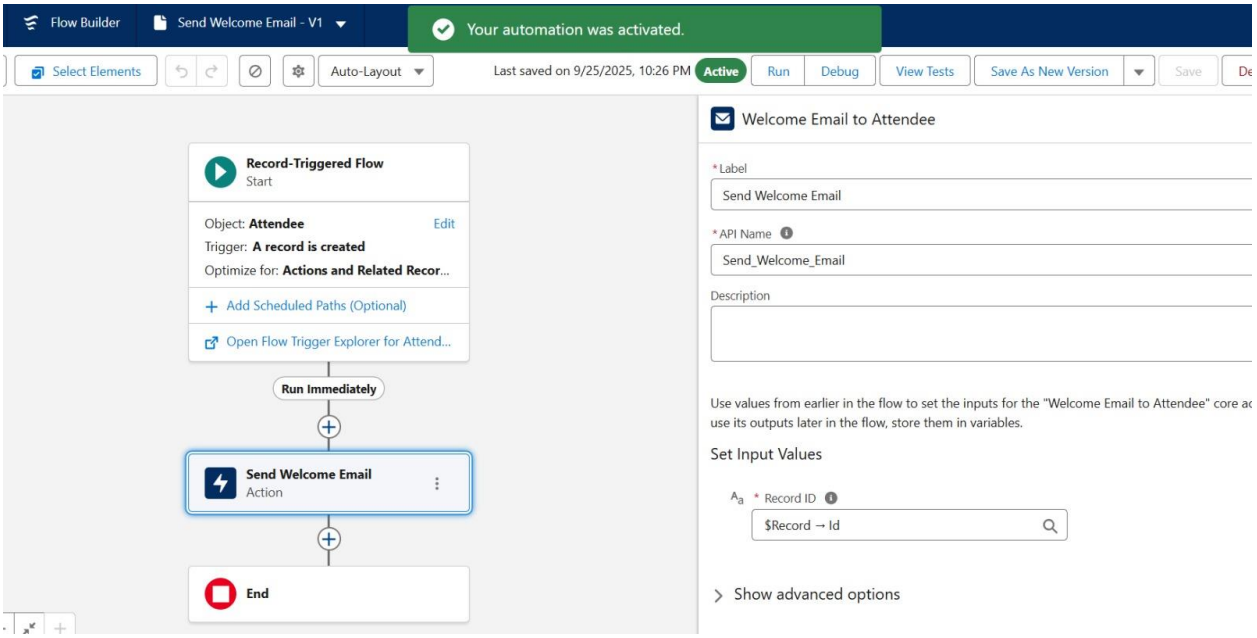
9. Process Builders:
    - Used for all record triggered automations in this project.

10. Approval Process:



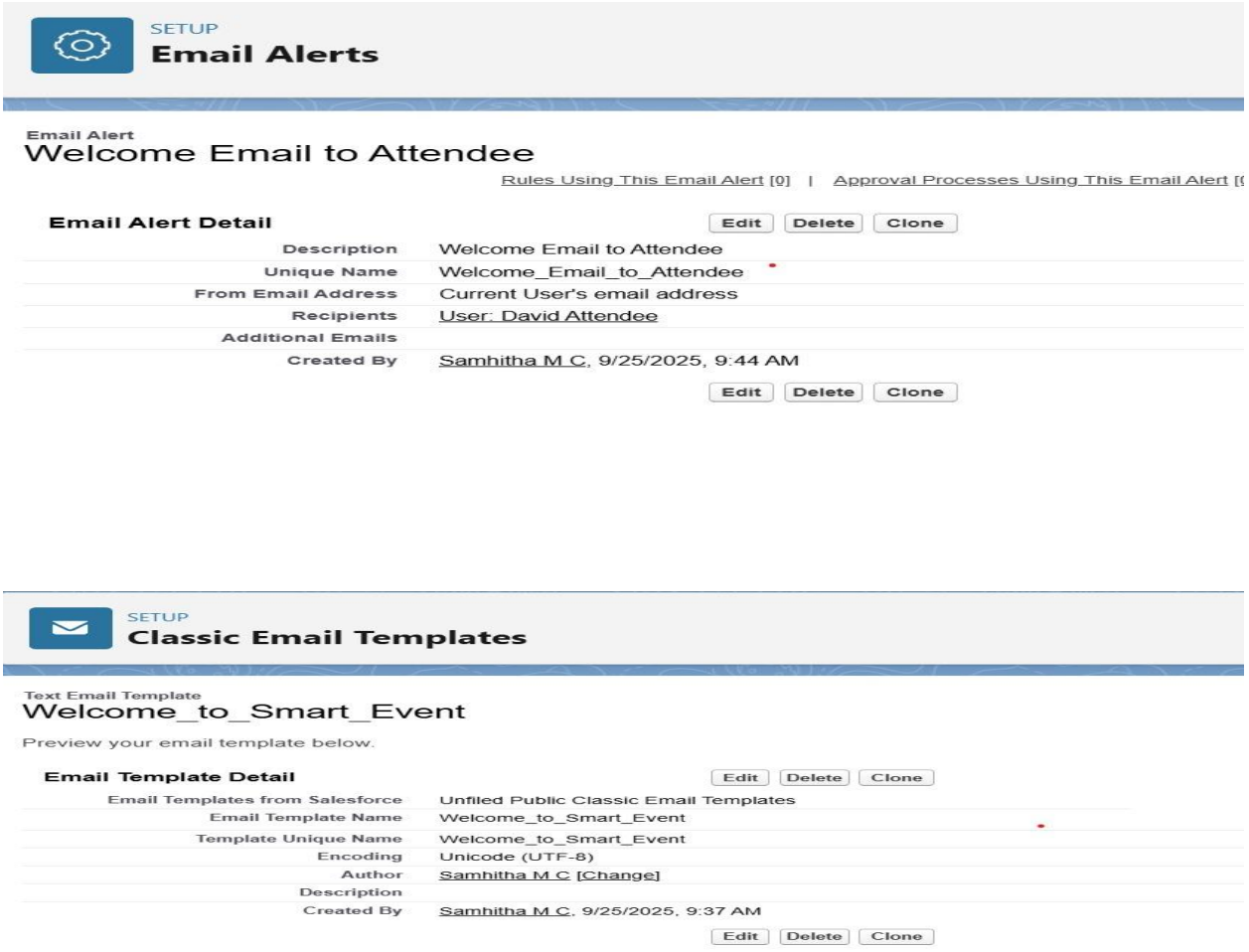
- Approval process is required for this project has it requires more team members

11. Flow Builder



- Record-Triggered Flow: Runs automatically when a attendee is created . This single flow is the cover of the project's automation.

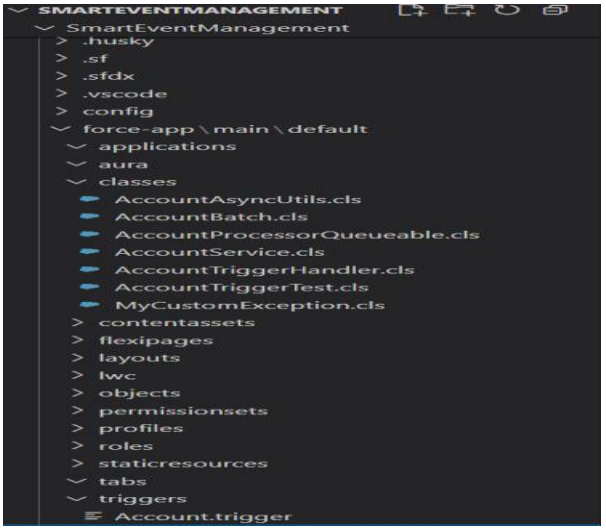
12. Email alerts



Phase 5: Apex Programming (Developer)

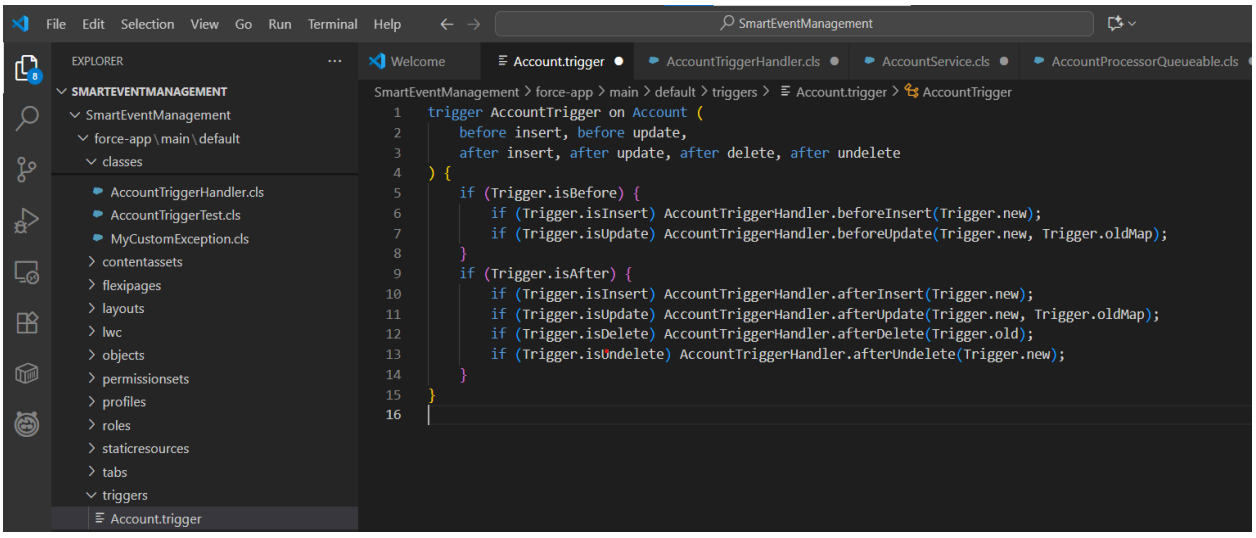
1. Classes and Objects

- A Class is a blueprint for an object. It defines variables (attributes) and methods (actions).
- AccountAsyncUtils.cls: Likely a utility class with helper methods, possibly for asynchronous operations.
- AccountBatch.cls: A Batch Apex class, a blueprint for a job object that processes large data sets.
- AccountProcessorQueueable.cls: A Queueable Apex class, a blueprint for an asynchronous job object.
- AccountService.cls: A service layer class that holds the main business logic for the Account object.
- AccountTriggerHandler.cls: The class that contains the bulkified logic for the Account trigger.
- AccountTriggerTest.cls: A test class (ends with Test.cls) used to verify the functionality of other Account-related classes.
- MyCustomException.cls: A blueprint for a custom error object that can be thrown and caught in your code.



In the context of Apex programming, an Object is an instance created from a class. These objects do not exist as files in your directory; they are created and used at runtime within the code

2. Apex Triggers (before/after insert/update/delete)



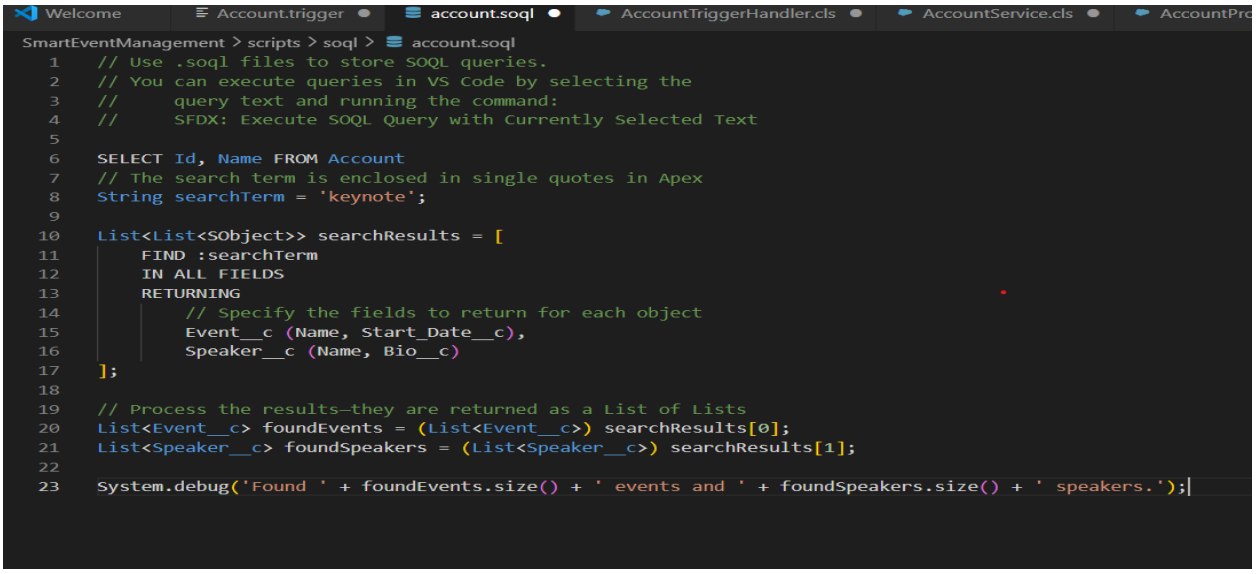
- Apex Triggers are code blocks that execute before or after a Data Manipulation Language (DML) event (insert, update, delete, undelete) occurs on a Salesforce object record.

3. Trigger Design Pattern

- A trigger framework or pattern is a way to structure your Apex code to make it scalable, maintainable

4. SOQL and SOSL

- SOQL (Salesforce Object Query Language): Used to retrieve records from a *single* object or multiple related objects based on criteria. It's similar to SELECT in SQL.
- SOSL (Salesforce Object Search Language): Used to search for text across *multiple* objects and fields simultaneously. It's better for text-based searches when you don't know the exact object.



5. Collections: List, Set, Map

- Used a Set<String> in trigger handler to efficiently store keys.

6. Control Statement

- The trigger handler uses if statements to iterate and check the conditions.

7. Asynchronous Processing (Batch, Queueable, Future Methods)

- Scheduled Apex is a form of asynchronous processing. The use of Batch or Queueable Apex isn't explicitly mentioned as a required step, but they are often the next logical step if the logic within your scheduled job needs to process millions of records.

8. Exception Handling

- Using try, catch, and finally blocks to gracefully handle runtime errors (exceptions) without halting the entire transaction or showing a cryptic error message to the user.

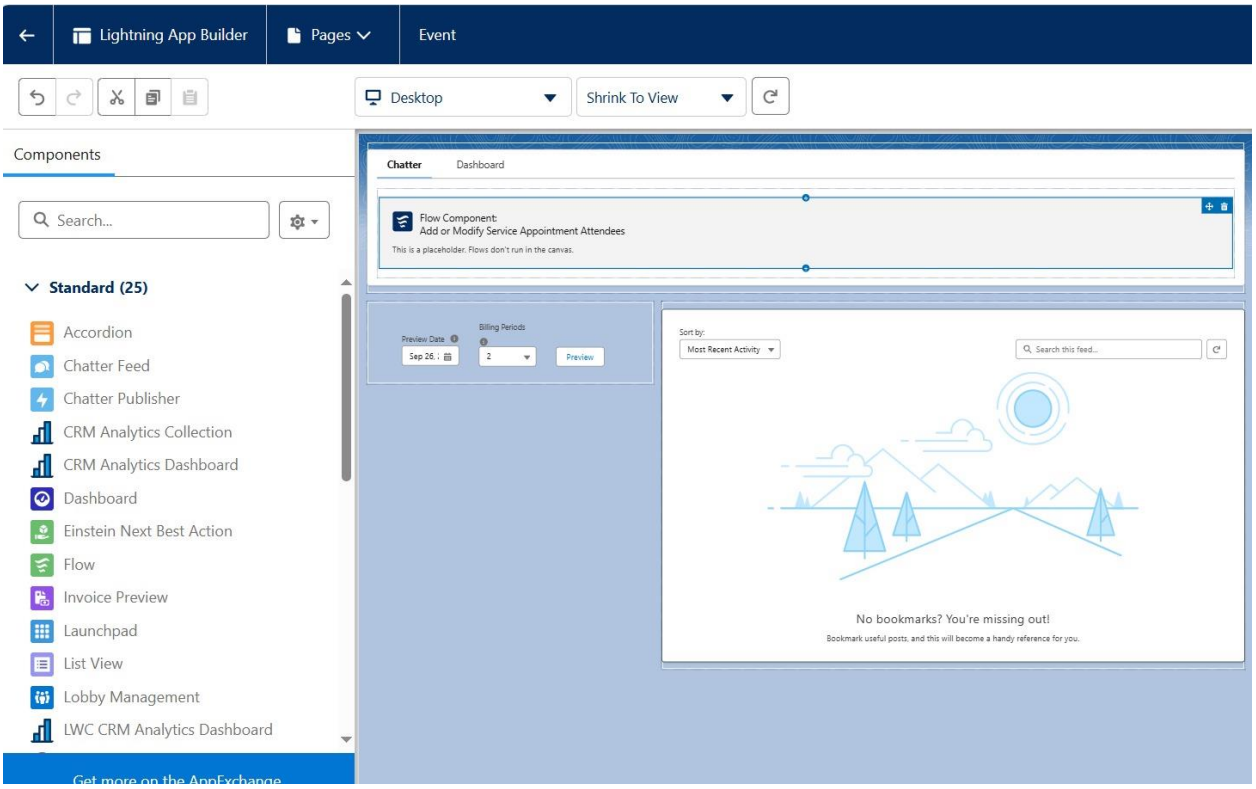
9. Test Classes

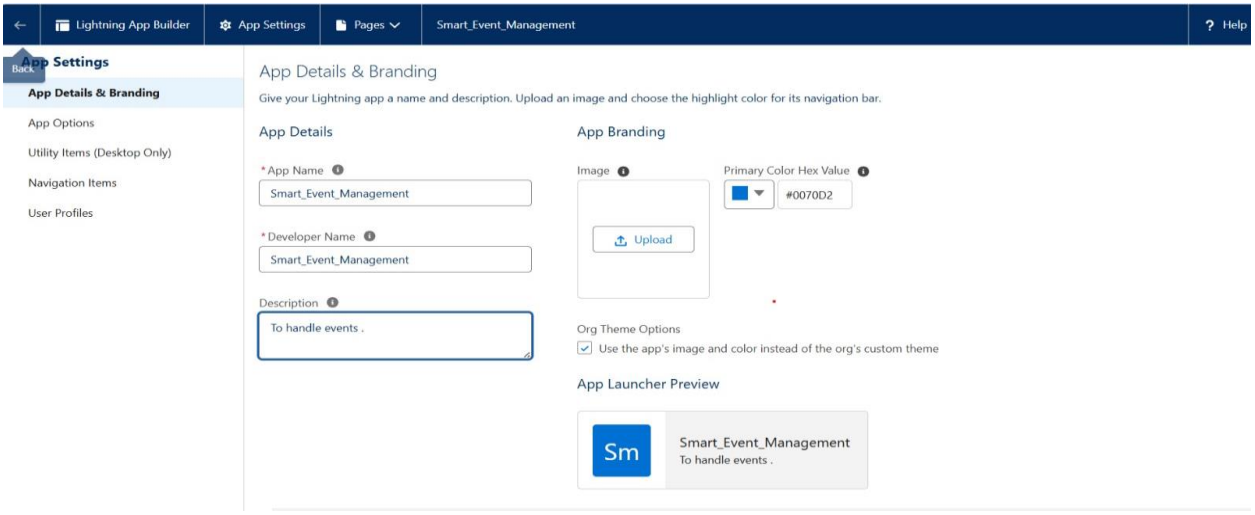
```
@isTest
private class AccountTriggerTest {
    @isTest static void testBeforeInsertAndQueueableAndBatch() {
        // Create test data
        List<Account> accts = new List<Account>();
        for (Integer i=0; i<5; i++) {
            accts.add(new Account(Name = '  TestAcct ' + i + '  ')); // spaces to be trimmed
        }

        Test.startTest();
            insert accts; // triggers run, queueable enqueued
            // run batch within test context
            Database.executeBatch(new AccountBatch(), 100);
        Test.stopTest();

        // Collect inserted ids
        List<Id> ids = new List<Id>();
        for (Account a : accts) ids.add(a.Id);

        // Verify names were trimmed
        for (Account a : [SELECT Name, Description FROM Account WHERE Id IN :ids]) {
            System.assertEquals(a.Name, a.Name.trim(), 'Name should be trimmed by trigger/service');
            // Description should include 'processed' because queueable updated it during Test.stopTest()
            System.assert(a.Description != null && a.Description.contains('processed'), 'Queueable should have updated Description');
        }
    }
}
```

Phase 6: User Interface Development

1 . Lightning App Builder

- Used the lighting app builder to create app named “Smart Event Management”





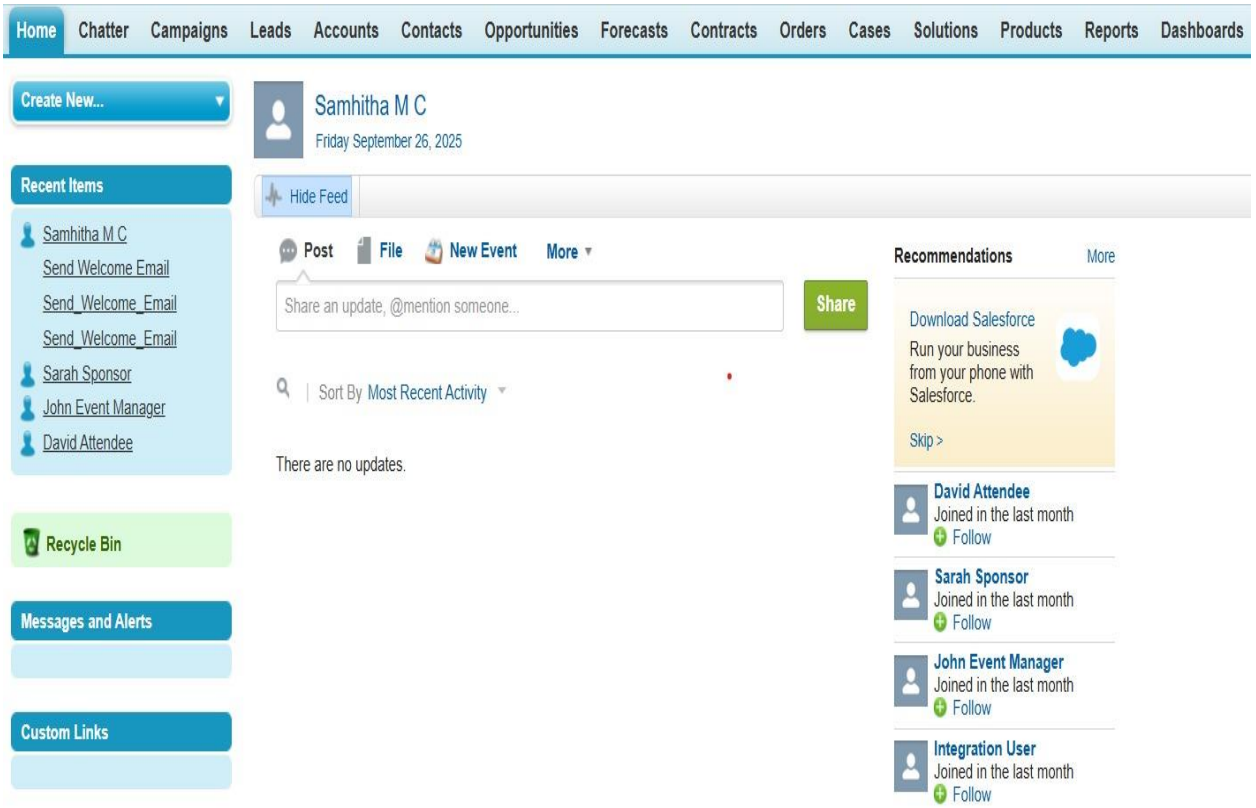
2. Record Pages

- Event and Booking record pages can be customized to display important event details like participants, venue, and schedule.

3. Tabs

- Custom tabs can be created for modules like Events, Bookings, Sponsors to allow easy navigation inside the Smart Event Manager app.
- Tabs help segment functionalities so that users can quickly access required features like event reports or registration details.

4. Home Page Layouts



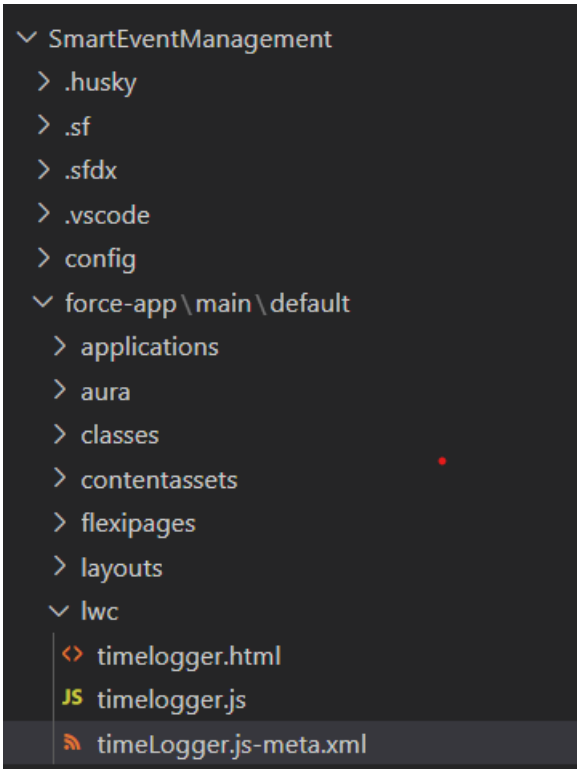
5. Utility Bar

- The utility bar can include a Recent Items component so event managers can quickly access frequently used records like active events.

6. LWC

SmartEventManagement > force-app > main > default > lwc > timelogger.html > ...

```
<template>
  <lightning-card title="Log Billable Time">
    <div class="slds-p-around_medium">
      <lightning-input label="Hours" type="number" value={hours} onchange={handleHoursChange}></lightning-input>
      <lightning-textarea label="Description" value={description} onchange={handleDescChange}></lightning-textarea>
      <div class="slds-m-top_small">
        <lightning-button label="Log Time" onclick={handleLogTime} variant="brand"></lightning-button>
      </div>
    </div>
  </lightning-card>
</template>

```



v. File directory which includes LWC file

- LWCs provide reusable UI blocks that ensure faster load times and responsive design for event operations.

7. Apex with LWC

- Apex classes can handle backend logic like saving registrations, calculating total revenue, or generating participant certificates.

8. Events in LWC

- Custom events can notify parent components, e.g., once an attendee has been successfully registered.

9. Imperative Apex calls

- Imperative calls can be used for on-demand actions like booking an event ticket when the “Register” button is clicked.
- They allow error handling and confirmation messages (e.g., “Registration Successful”) directly in the UI.

10 . Navigate service

- Navigating service is not used in this component , but a future enhancement could be to add a button that navigates service can redirect users to a list of all events, specific event records, or reports after completing an action.

Phase 7: Integration and External Access

- For the **Smart Event Management Project**, we are not implementing Phase 7 at this stage because the current scope is focused on building core features. External integrations and advanced access configurations can be added in future phases when the application requires connections to outside systems.

1. Named Credentials

- Provide a secure way to store authentication details (like usernames, passwords, tokens) for calling external systems.
- Simplify integration by avoiding hardcoding credentials inside Apex code.

2. External Services

- Allow Salesforce to directly consume APIs from external systems using a declarative setup.
- Useful for connecting event management with third-party apps like payment gateways or ticketing platforms**.**

3. Web Services (REST/SOAP)

- Salesforce can expose its data as REST or SOAP services for other systems to consume.
- It can also consume external services for tasks like syncing event attendee data from partner systems.

4. Callouts

- Apex callouts allow Salesforce to send HTTP requests to external services (e.g., sending attendee info to an email service).
- Callouts are essential for real-time communication with APIs like payment processing or SMS notifications.

5. Platform Events

- Used for event-driven architectures, enabling apps inside and outside Salesforce to communicate asynchronously.
- For example, an “Event Registered” platform event could notify other systems about new registrations instantly.

6. Change Data Capture (CDC)

- Publishes real-time events whenever data changes in Salesforce (e.g., a new booking is created).
- Helps external systems stay synchronized without the need for constant API polling.

7. Salesforce Connect

- Allows Salesforce to access data stored in external databases without importing it into Salesforce.
- Ideal for cases where event data is maintained in another system but needs to be viewed in Salesforce

8. API Limits

- Understanding and optimizing API usage is crucial for integrations that involve high volumes of transactions.

9. OAuth & Authentication

- OAuth provides a secure way for Salesforce to authenticate with external services.
- It ensures only authorized systems and users can access sensitive event-related data.

10. Remote Site Settings

- Before making a callout, the external service URL must be whitelisted in Salesforce Remote Site Settings**.**
- This acts as a safeguard against unauthorized or accidental callouts to untrusted domains