Phase 3: Data Modeling & Object Creation

Event Registration & Attendee Management

Overview

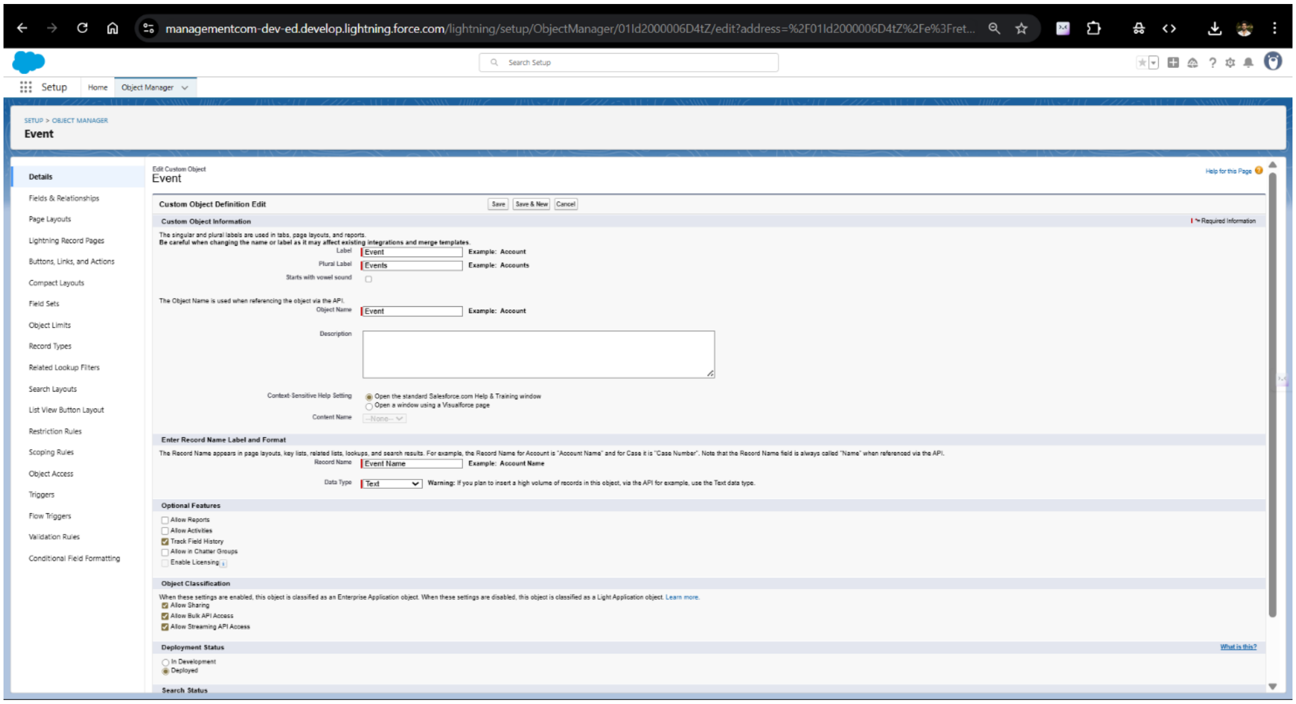
This phase covers the complete design and creation of the custom objects, fields, relationships, and page layouts for the Event Registration & Attendee Management system. Accurately modeling the data ensures the app meets business requirements, is easy to use, and supports scalability and automation.

---

1. Custom Object Creation

1.1 Create "Event" Object

- In Setup, go to **Object Manager** → *Create* → *Custom Object*.
- Enter:
  - Label: Event
  - Plural Label: Events
  - Object Name: Event__c
  - Record Name: Event Name (Text)
  - Enable Reports and Activities
- Click **Save**.



Edit Custom Object
Event

Custom Object Definition Edit

Custom Object Information

| = Required Information

The singular and plural labels are used in tabs, page layouts, and reports.
Be careful when changing the name or label as it may affect existing integrations and merge templates.

Label: Event **Example: Account**

Plural Label: Events **Example: Accounts**

Starts with vowel sound: □

The Object Name is used when referencing the object via the API.

Object Name: Event **Example: Account**

Description:

Context-Sensitive Help Setting: ● Open the standard Salesforce.com Help & Training window ○ Open a window using a Visualforce page

Content Name: --None--

Enter Record Name Label and Format

The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" and for Case it is "Case Number". Note that the Record Name field is always called "Name" when referenced via the API.

Record Name: Event Name **Example: Account Name**

Data Type: Text **Warning:** If you plan to insert a high volume of records in this object, via the API for example, use the Text data type.

Optional Features

□ Allow Reports
□ Allow Activities
☑ Track Field History
□ Allow in Chatter Groups
□ Enable Licensing

Object Classification

When these settings are enabled, this object is classified as an Enterprise Application object. When these settings are disabled, this object is classified as a Light Application object. Learn more.

☑ Allow Sharing
☑ Allow Bulk API Access
☑ Allow Streaming API Access

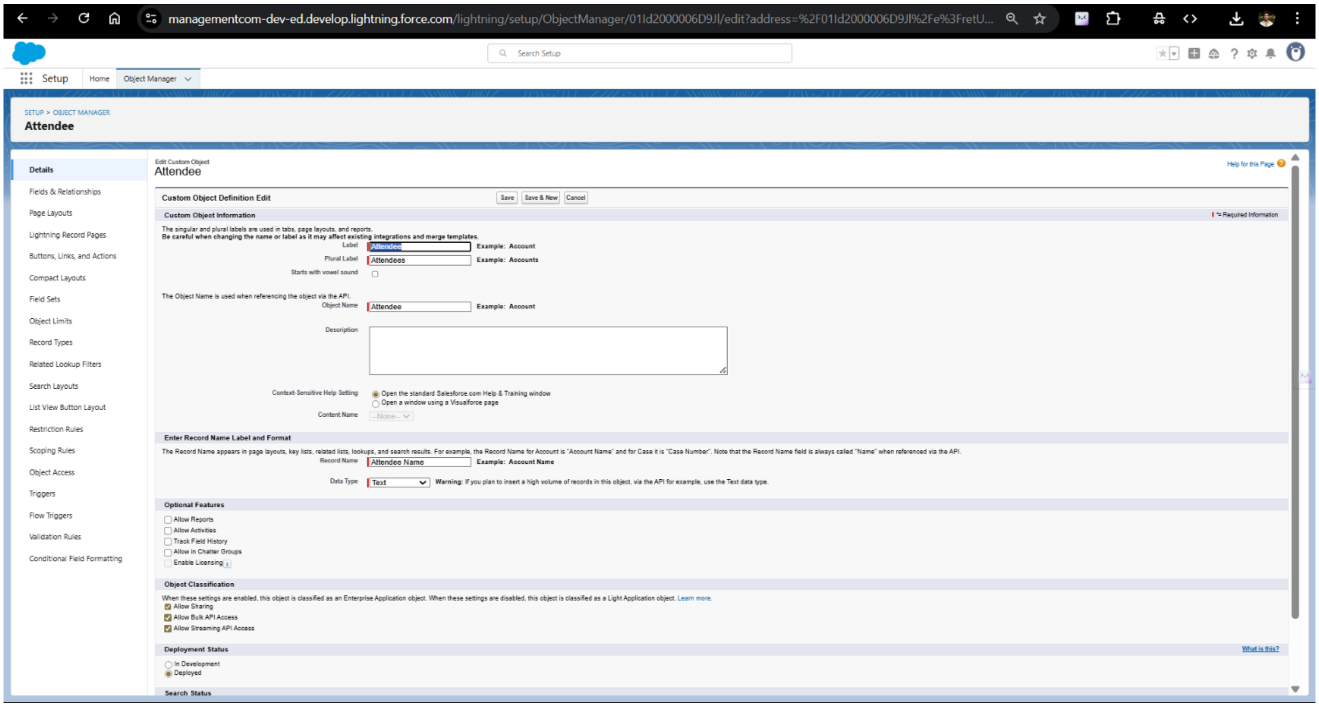
Deployment Status

○ In Development
● Deployed

Search Status

1.2 Create "Attendee" Object

- Repeat the above steps.
- Enter:
    - Label: Attendee
    - Plural Label: Attendees
    - Object Name: Attendee__c
    - Record Name: Attendee Name (Text)
    - Enable Reports and Activities
- Click **Save.**



Edit Custom Object

Attendee

Custom Object Definition Edit

Custom Object Information

| = Required Information

The singular and plural labels are used in tabs, page layouts, and reports.
Be careful when changing the name or label as it may affect existing integrations and merge templates.

Label: Attendee — **Example: Account**

Plural Label: Attendees — **Example: Accounts**

Starts with vowel sound: □

The Object Name is used when referencing the object via the API.

Object Name: Attendee — **Example: Account**

Description:

Context-Sensitive Help Setting: ● Open the standard Salesforce.com Help & Training window
○ Open a window using a Visualforce page

Content Name: --None--

Enter Record Name Label and Format

The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" and for Case it is "Case Number". Note that the Record Name field is always called "Name" when referenced via the API.

Record Name: Attendee Name — **Example: Account Name**

Data Type: Text — **Warning:** If you plan to insert a high volume of records in this object, via the API for example, use the Text data type.

Optional Features

□ Allow Reports
□ Allow Activities
□ Track Field History
□ Allow in Chatter Groups
□ Enable Licensing

Object Classification

When these settings are enabled, this object is classified as an Enterprise Application object. When these settings are disabled, this object is classified as a Light Application object. Learn more.

☑ Allow Sharing
☑ Allow Bulk API Access
☑ Allow Streaming API Access

Deployment Status

○ In Development
● Deployed

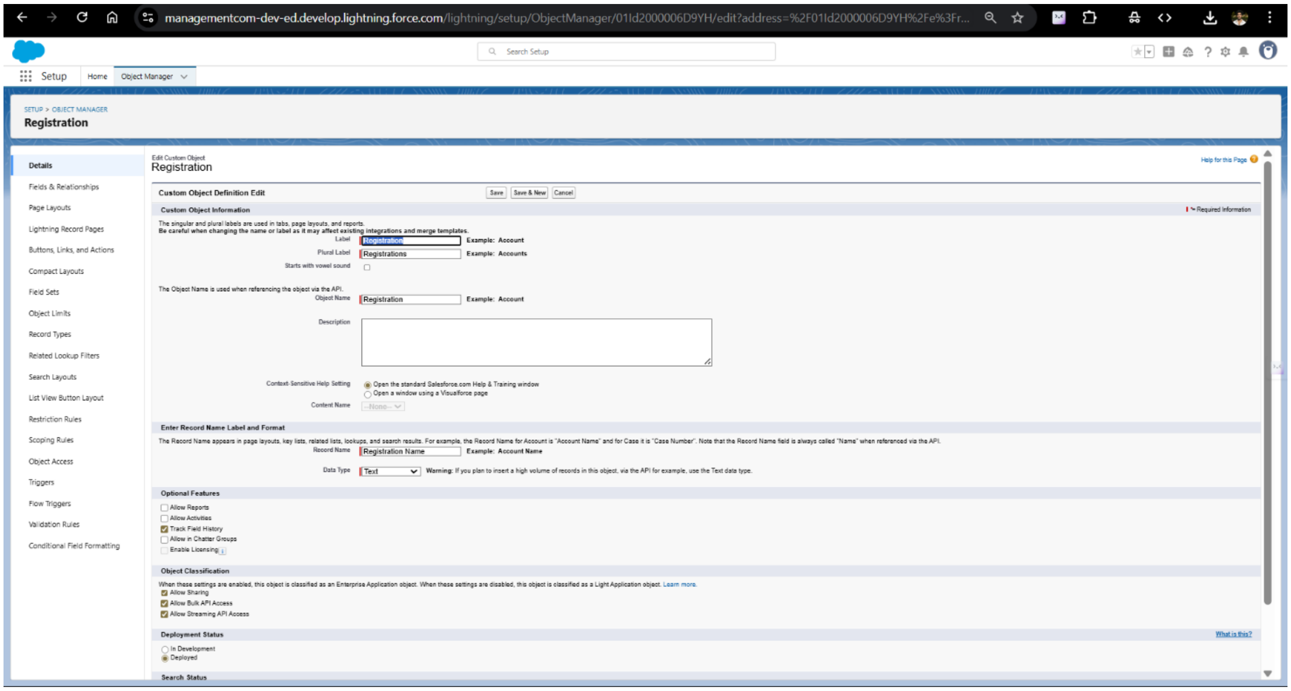
Search Status

1.3 Create "Registration" Object

- Repeat the custom object wizard.
- Enter:
    - Label: Registration
    - Plural Label: Registrations
    - Object Name: Registration__c
    - Record Name: Registration Number (Auto Number or Text)

  - Enable Reports and Activities

- Click **Save**.



Edit Custom Object
Registration

Custom Object Definition Edit

Custom Object Information — | = Required Information

The singular and plural labels are used in tabs, page layouts, and reports.
Be careful when changing the name or label as it may affect existing integrations and merge templates.

Label: Registration — **Example: Account**
Plural Label: Registrations — **Example: Accounts**
Starts with vowel sound: □

The Object Name is used when referencing the object via the API.
Object Name: Registration — **Example: Account**

Description:

Context-Sensitive Help Setting: ● Open the standard Salesforce.com Help & Training window ○ Open a window using a Visualforce page
Content Name: --None--

Enter Record Name Label and Format

The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" and for Case it is "Case Number". Note that the Record Name field is always called "Name" when referenced via the API.
Record Name: Registration Name — **Example: Account Name**
Data Type: Text — **Warning:** If you plan to insert a high volume of records in this object, via the API for example, use the Text data type.

Optional Features

□ Allow Reports
□ Allow Activities
☑ Track Field History
□ Allow in Chatter Groups
□ Enable Licensing

Object Classification

When these settings are enabled, this object is classified as an Enterprise Application object. When these settings are disabled, this object is classified as a Light Application object. Learn more.
☑ Allow Sharing
☑ Allow Bulk API Access
☑ Allow Streaming API Access

Deployment Status

○ In Development
● Deployed

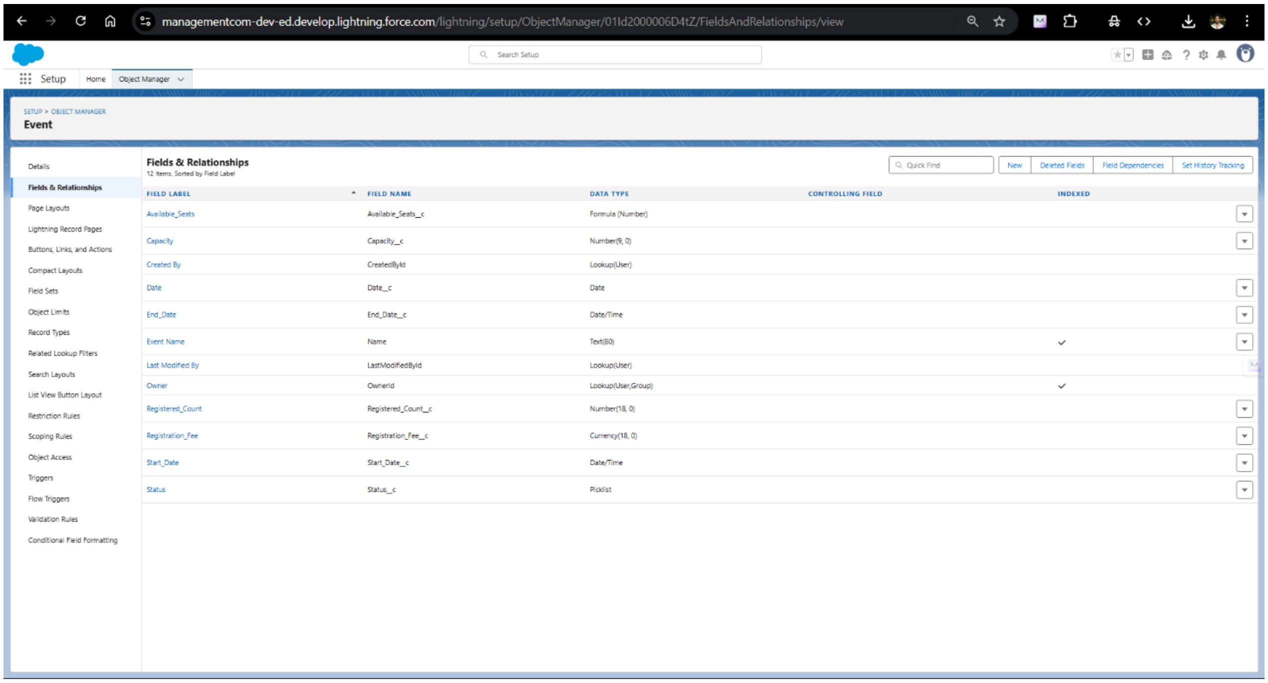
Search Status

---

2. Add Fields to Objects

2.1 Event__c Fields

- **Start_Date__c** (Date/Time)
- **End_Date__c** (Date/Time)
- **Venue__c** (Text or Lookup to Venue)
- **Capacity__c** (Number, 0 decimals, default 0)
- **Registration_Fee__c** (Currency)
- **Status__c** (Picklist: Draft; Published; Closed; Cancelled)
- **Registered_Count__c** (Number, 0 decimals)
- **Available_Seats__c** (Formula): Capacity__c - Registered_Count__c
- **Registration_Open__c** (Checkbox)



Event

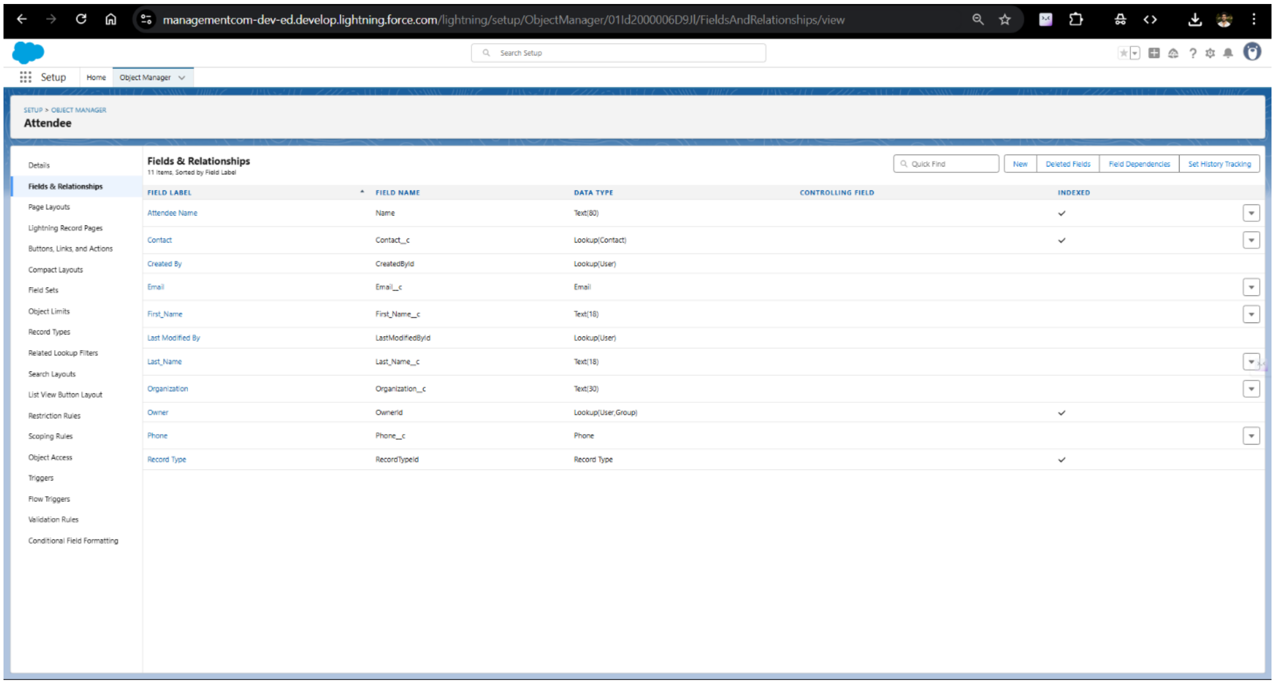
Fields & Relationships

12 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Available_Seats	Available_Seats__c	Formula (Number)		
Capacity	Capacity__c	Number(9, 0)		
Created By	CreatedById	Lookup(User)		
Date	Date__c	Date		
End_Date	End_Date__c	Date/Time		
Event Name	Name	Text(80)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Registered_Count	Registered_Count__c	Number(18, 0)		
Registration_Fee	Registration_Fee__c	Currency(18, 0)		
Start_Date	Start_Date__c	Date/Time		
Status	Status__c	Picklist		

2.2Attendee__c Fields

- **First_Name__c** (Text)
- **Last_Name__c** (Text)
- **Email__c** (Email)
- **Phone__c** (Phone)
- **Organization__c** (Text)
- **Contact__c** (Lookup to Contact, optional)



Attendee

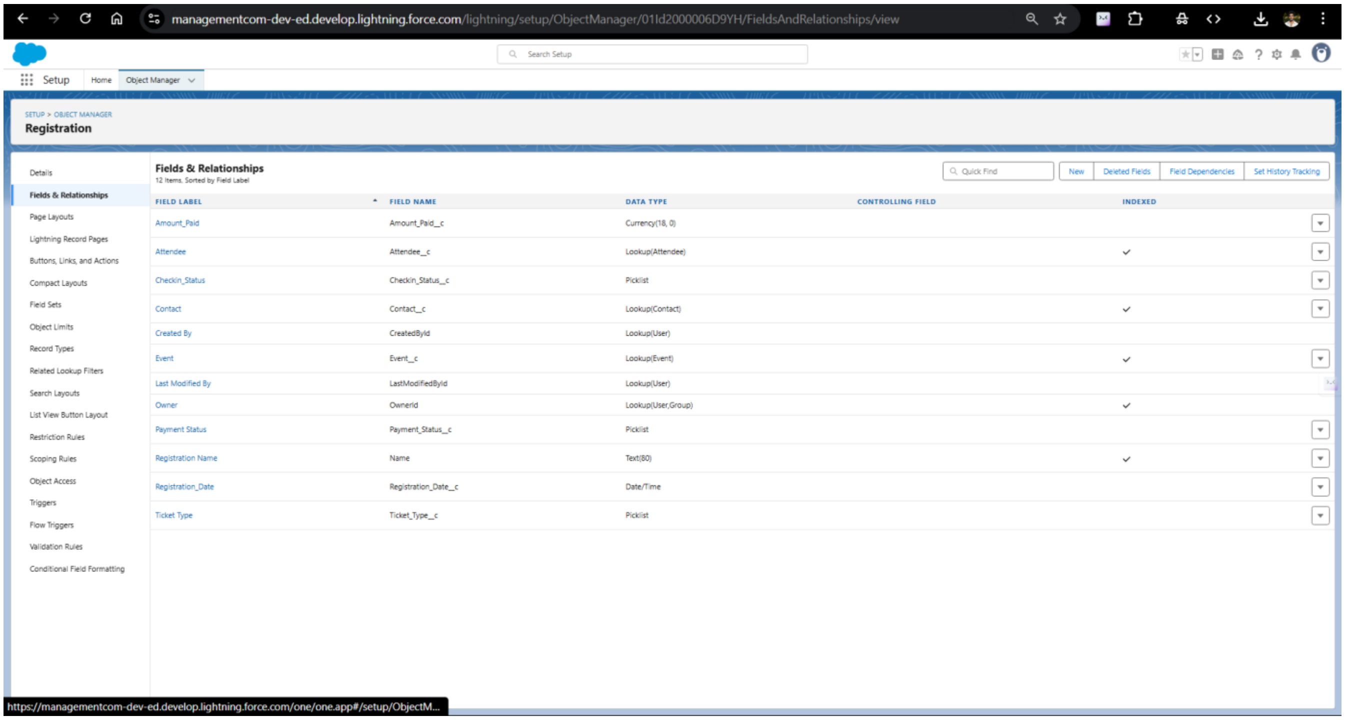
Fields & Relationships

11 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Attendee Name	Name	Text(80)		✓
Contact	Contact__c	Lookup(Contact)		✓
Created By	CreatedById	Lookup(User)		
Email	Email__c	Email		
First_Name	First_Name__c	Text(18)		
Last Modified By	LastModifiedById	Lookup(User)		
Last_Name	Last_Name__c	Text(18)		
Organization	Organization__c	Text(30)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		
Record Type	RecordTypeId	Record Type		✓

2.3 Registration__c Fields

- **Event__c** (Lookup to Event__c, required)
- **Attendee__c** (Lookup to Attendee__c, required)
- **Registration_Date__c** (Date/Time)
- **Ticket_Type__c** (Picklist)
- **Payment_Status__c** (Picklist: Pending, Paid, Failed, Refunded)
- **Amount_Paid__c** (Currency)
- **Checkin_Status__c** (Picklist: Not Checked-in, Checked-in)



Fields & Relationships
12 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Amount_Paid	Amount_Paid__c	Currency(18, 0)		
Attendee	Attendee__c	Lookup(Attendee)		✓
Checkin_Status	Checkin_Status__c	Picklist		
Contact	Contact__c	Lookup(Contact)		✓
Created By	CreatedById	Lookup(User)		
Event	Event__c	Lookup(Event)		✓
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Payment Status	Payment_Status__c	Picklist		
Registration Name	Name	Text(80)		✓
Registration_Date	Registration_Date__c	Date/Time		
Ticket Type	Ticket_Type__c	Picklist		

---

3. Set up Relationships & Related Lists

- On **Registration__c**, both Event__c and Attendee__c are required lookup fields.
- On the **Event__c** page layout, ensure the “Registrations” related list is visible.
- On the **Attendee__c** page layout, ensure the “Registrations” related list is visible.

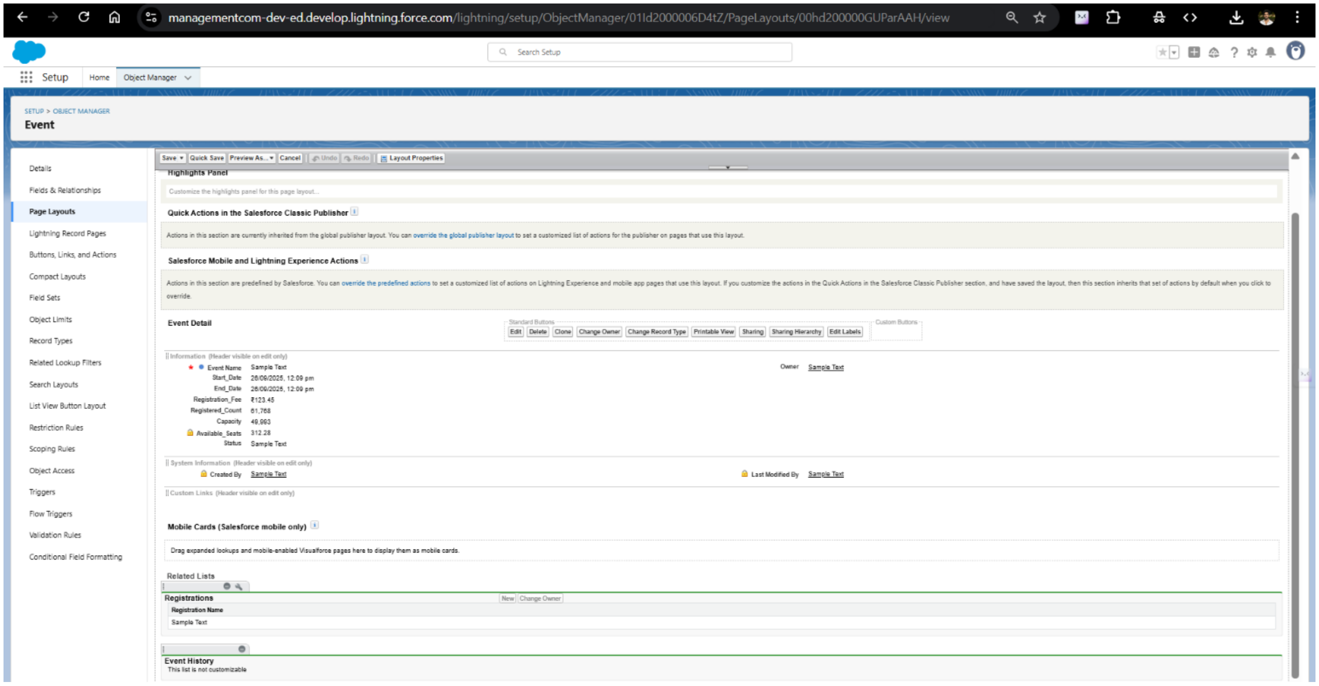
---

4. Page Layouts & Compact Layouts

- For each object (Event__c, Registration__c, Attendee__c):
  - Edit the Page Layout.
  - Move most important fields to the top (e.g., Name, Start Date, Status for Event).

- Add related lists (e.g., Registrations).
- Save and assign the default Compact Layout to prioritize key fields (Name, Status, Date).

➢ Event Object



Highlights Panel

Customize the highlights panel for this page layout...

Quick Actions in the Salesforce Classic Publisher

Actions in this section are currently inherited from the global publisher layout. You can override the global publisher layout to set a customized list of actions for the publisher on pages that use this layout.

Salesforce Mobile and Lightning Experience Actions

Actions in this section are predefined by Salesforce. You can override the predefined actions to set a customized list of actions on Lightning Experience and mobile app pages that use this layout. If you customize the actions in the Quick Actions in the Salesforce Classic Publisher section, and have saved the layout, then this section inherits that set of actions by default when you click to override.

Event Detail

Information (Header visible on edit only)

Event Name	Sample Text	Owner	Sample Text
Start_Date	26/09/2025, 12:09 pm		
End_Date	26/09/2025, 12:09 pm		
Registration_Fee	₹123.45		
Registered_Count	61,768		
Capacity	49,993		
Available_Seats	312.28		
Status	Sample Text		

System Information (Header visible on edit only)

Created By	Sample Text	Last Modified By	Sample Text

Custom Links (Header visible on edit only)

Mobile Cards (Salesforce mobile only)

Drag expanded lookups and mobile-enabled Visualforce pages here to display them as mobile cards.

Related Lists

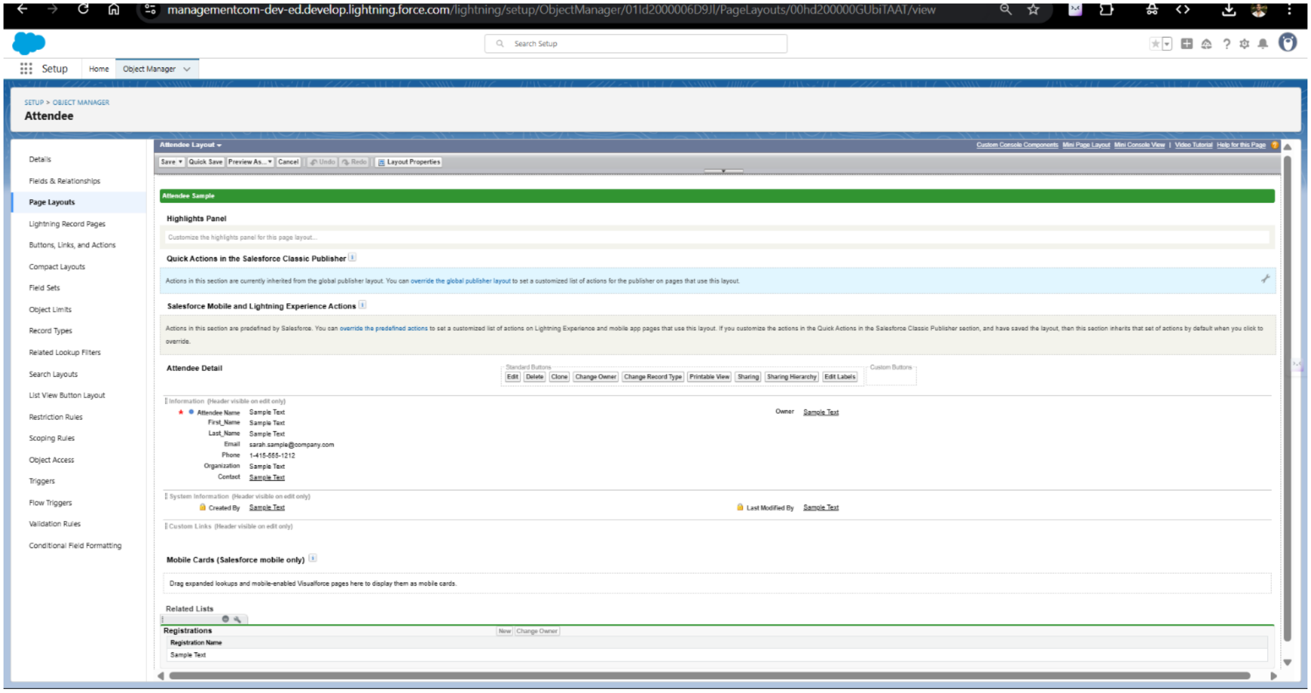
Registrations

Registration Name
Sample Text

Event History

This list is not customizable

> Attendee Object



Attendee Sample

Highlights Panel

Customize the highlights panel for this page layout...

Quick Actions in the Salesforce Classic Publisher

Actions in this section are currently inherited from the global publisher layout. You can override the global publisher layout to set a customized list of actions for the publisher on pages that use this layout.

Salesforce Mobile and Lightning Experience Actions

Actions in this section are predefined by Salesforce. You can override the predefined actions to set a customized list of actions on Lightning Experience and mobile app pages that use this layout. If you customize the actions in the Quick Actions in the Salesforce Classic Publisher section, and have saved the layout, then this section inherits that set of actions by default when you click to override.

Attendee Detail

Information (Header visible on edit only)

Attendee Name	Sample Text	Owner	Sample Text
First_Name	Sample Text		
Last_Name	Sample Text		
Email	sarah.sample@company.com		
Phone	1-415-555-1212		
Organization	Sample Text		
Contact	Sample Text		

System Information (Header visible on edit only)

Created By	Sample Text	Last Modified By	Sample Text

Custom Links (Header visible on edit only)

Mobile Cards (Salesforce mobile only)

Drag expanded lookups and mobile-enabled Visualforce pages here to display them as mobile cards.

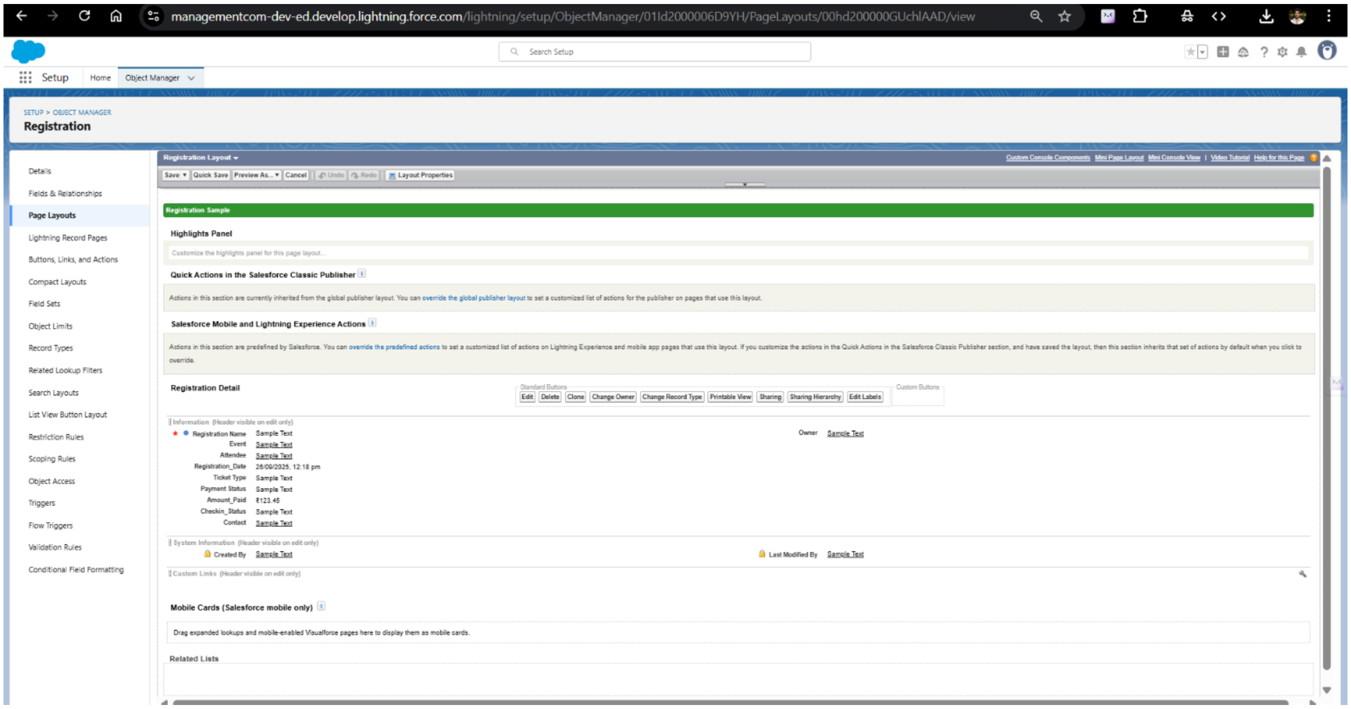
Related Lists

Registrations

Registration Name
Sample Text

>

> Registration Object



Registration Sample

Highlights Panel

Customize the highlights panel for this page layout...

Quick Actions in the Salesforce Classic Publisher

Actions in this section are currently inherited from the global publisher layout. You can override the global publisher layout to set a customized list of actions for the publisher on pages that use this layout.

Salesforce Mobile and Lightning Experience Actions

Actions in this section are predefined by Salesforce. You can override the predefined actions to set a customized list of actions on Lightning Experience and mobile app pages that use this layout. If you customize the actions in the Quick Actions in the Salesforce Classic Publisher section, and have saved the layout, then this section inherits that set of actions by default when you click to override.

Registration Detail

Information (Header visible on edit only)

Registration Name	Sample Text	Owner	Sample Text
Event	Sample Text		
Attendee	Sample Text		
Registration_Date	26/09/2025, 12:18 pm		
Ticket Type	Sample Text		
Payment Status	Sample Text		
Amount_Paid	₹123.45		
Checkin_Status	Sample Text		
Contact	Sample Text		

System Information (Header visible on edit only)

Created By	Sample Text	Last Modified By	Sample Text

Custom Links (Header visible on edit only)

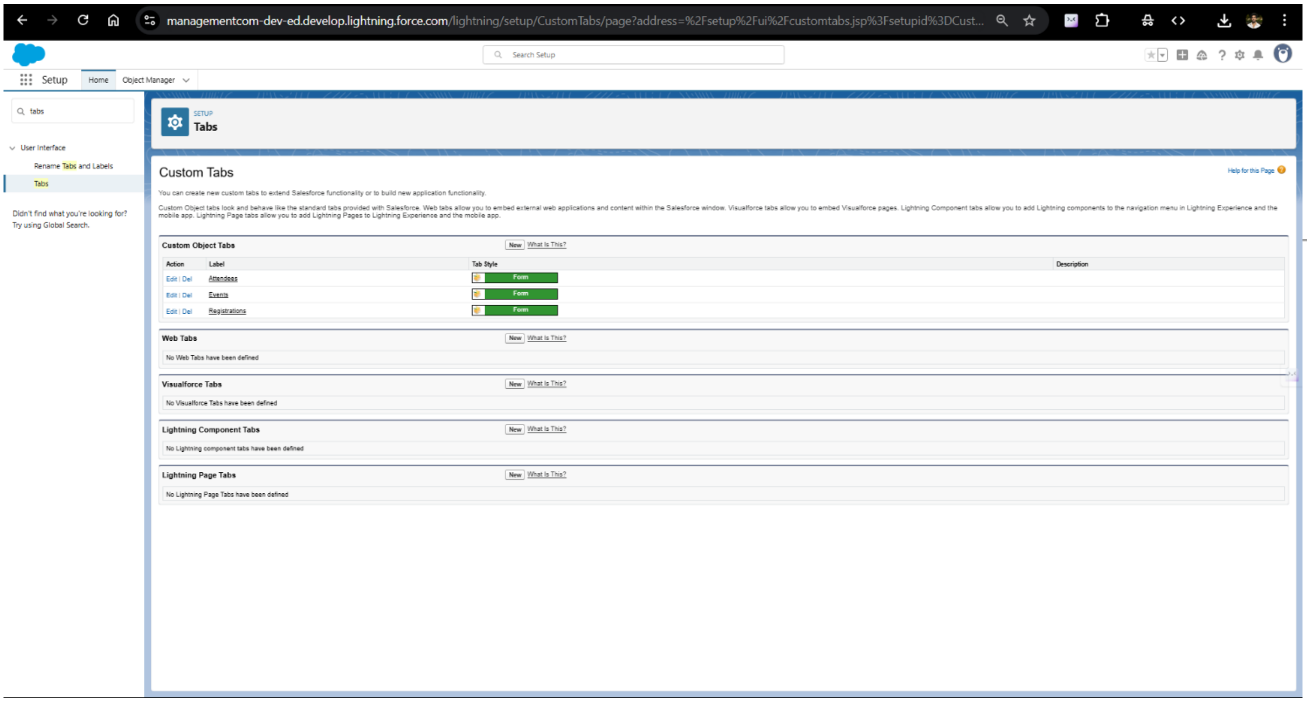
Mobile Cards (Salesforce mobile only)

Drag expanded lookups and mobile-enabled Visualforce pages here to display them as mobile cards.

Related Lists

5. Custom Tabs

- Setup → Tabs → New → Custom Object Tab.
- Create tabs for "Event", "Attendee", and "Registration" objects.
- Choose icons/styles; save each tab.
- Check tab visibility in profiles/permission sets.


managementcom-dev-ed.develop.lightning.force.com/lightning/setup/CustomTabs/page?address=%2Fsetup%2Fui%2Fcustomtabs.jsp%3Fsetupid%3DCust...
Search Setup
Setup
Home
Object Manager
tabs
User Interface
Rename Tabs and Labels
Tabs
Didn't find what you're looking for? Try using Global Search.
SETUP
Tabs
Help for this Page
New
What Is This?

Custom Tabs

You can create new custom tabs to extend Salesforce functionality or to build new application functionality.

Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.

Custom Object Tabs

Action	Label	Tab Style	Description
Edit | Del	Attendees	Form	
Edit | Del	Events	Form	
Edit | Del	Registrations	Form	

Web Tabs

No Web Tabs have been defined

Visualforce Tabs

No Visualforce Tabs have been defined

Lightning Component Tabs

No Lightning component tabs have been defined

Lightning Page Tabs

No Lightning Page Tabs have been defined

6. Object Security & Profile Permissions

- Confirm that Event Admin/Organizer profiles have:
  - Read, Create, Edit, Delete on Event, Attendee, Registration objects.
- Set **Tab Visibility** to Default On.
- Other profiles (e.g., Attendee) get minimal or read-only access.

---

7. Test Data Entry

- Switch to your Event Management app (App Launcher).
- Create a sample Event, Attendee, Registration.
- Confirm field relationships, required field enforcement, and that related lists display as intended.

Phase 4: Business Logic, Validation & Automation

Overview

This phase implements critical business rules, validation, and process automation to ensure data integrity, correct user behavior, and a seamless event registration workflow. Tools include validation rules, formulas, record-triggered flows, and lightning email templates—prioritizing declarative solutions unless Apex is necessary.

---

1. Validation Rules & Formulas

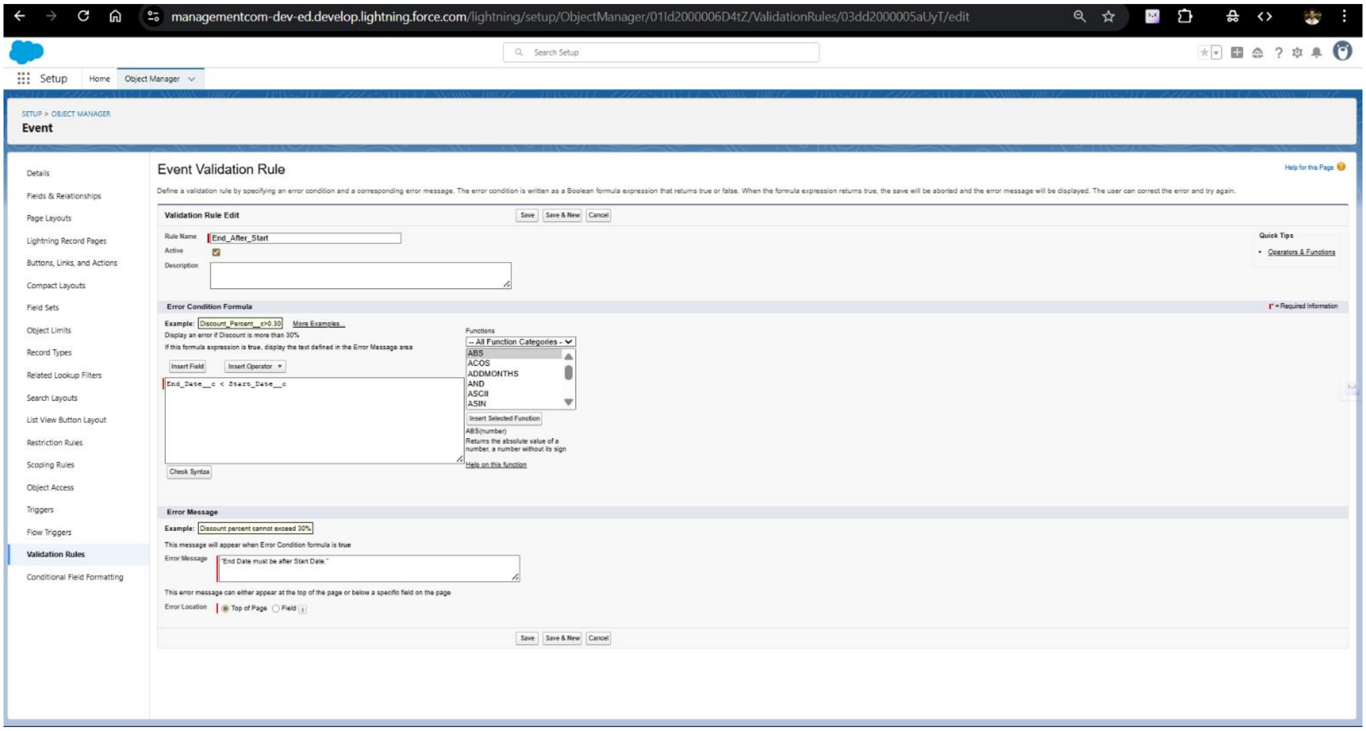
1.1 Event End Date Must Be After Start Date

- **Location:** Object Manager → Event__c → Validation Rules → New.
- **Rule Name:** End_After_Start
- **Error Condition Formula:**

text

End_Date__c < Start_Date__c

- **Error Message:** "End Date must be after Start Date."
- **Usage:** Prevents user error in scheduling events.



SETUP > OBJECT MANAGER

Event

Event Validation Rule

Define a validation rule by specifying an error condition and a corresponding error message. The error condition is written as a Boolean formula expression that returns true or false. When the formula expression returns true, the save will be aborted and the error message will be displayed. The user can correct the error and try again.

Validation Rule Edit

Rule Name: End_After_Start

Active: ☑

Description:

= Required Information

Error Condition Formula

Example: Discount_Percent__c>0.30

Display an error if Discount is more than 30%

If this formula expression is true, display the text defined in the Error Message area

```
End_Date__c < Start_Date__c
```

ABS(number)
Returns the absolute value of a number, a number without its sign

Error Message

Example: Discount percent cannot exceed 30%

This message will appear when Error Condition formula is true

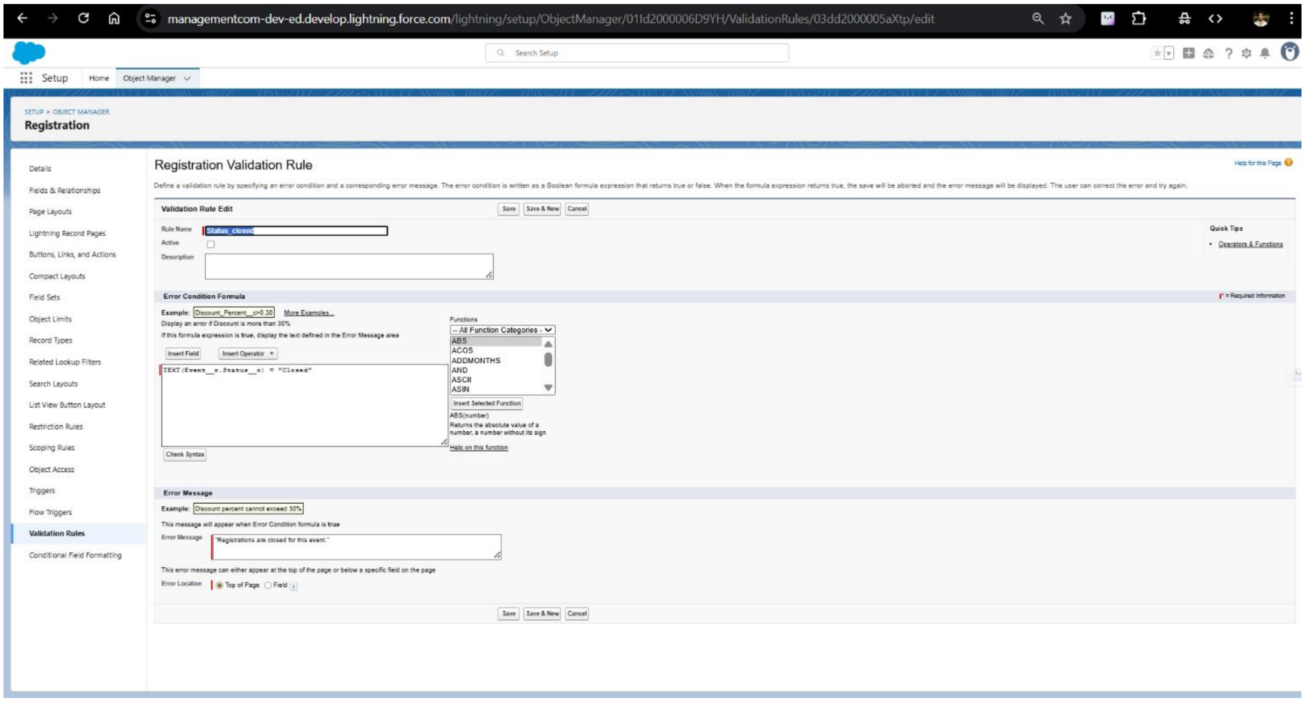
Error Message: "End Date must be after Start Date."

This error message can either appear at the top of the page or below a specific field on the page

Error Location: ● Top of Page ○ Field

1.2 Registration Prohibited When Event Closed

- **Location:** Object Manager → Registration__c → Validation Rules → New.
- **Formula:** Event__r.Status__c = "Closed"
- **Error Message:** "Registrations are closed for this event."
- **Usage:** Ensures no registrations when event is not accepting new attendees.



SETUP > OBJECT MANAGER

Registration

Registration Validation Rule

Define a validation rule by specifying an error condition and a corresponding error message. The error condition is written as a Boolean formula expression that returns true or false. When the formula expression returns true, the save will be aborted and the error message will be displayed. The user can correct the error and try again.

Validation Rule Edit

Rule Name: Status_closed

Active: □

Description:

= Required Information

Error Condition Formula

Example: Discount_Percent__c>0.30

Display an error if Discount is more than 30%

If this formula expression is true, display the text defined in the Error Message area

```
TEXT(Event__r.Status__c) = "Closed"
```

ABS(number)
Returns the absolute value of a number, a number without its sign

Error Message

Example: Discount percent cannot exceed 30%

This message will appear when Error Condition formula is true

Error Message: "Registrations are closed for this event."

This error message can either appear at the top of the page or below a specific field on the page

Error Location: ● Top of Page ○ Field

1.3 Formula Fields

- **Available_Seats__c** on Event__c:

text

Capacity__c - Registered_Count__c

- **Usage:** Provides real-time feedback on event capacity to users/admins and automations.

2.1 Flow: Update Registered_Count__c

Purpose

- Automatically maintain a running total of confirmed registrations per event and enforce event closure when full.

Type

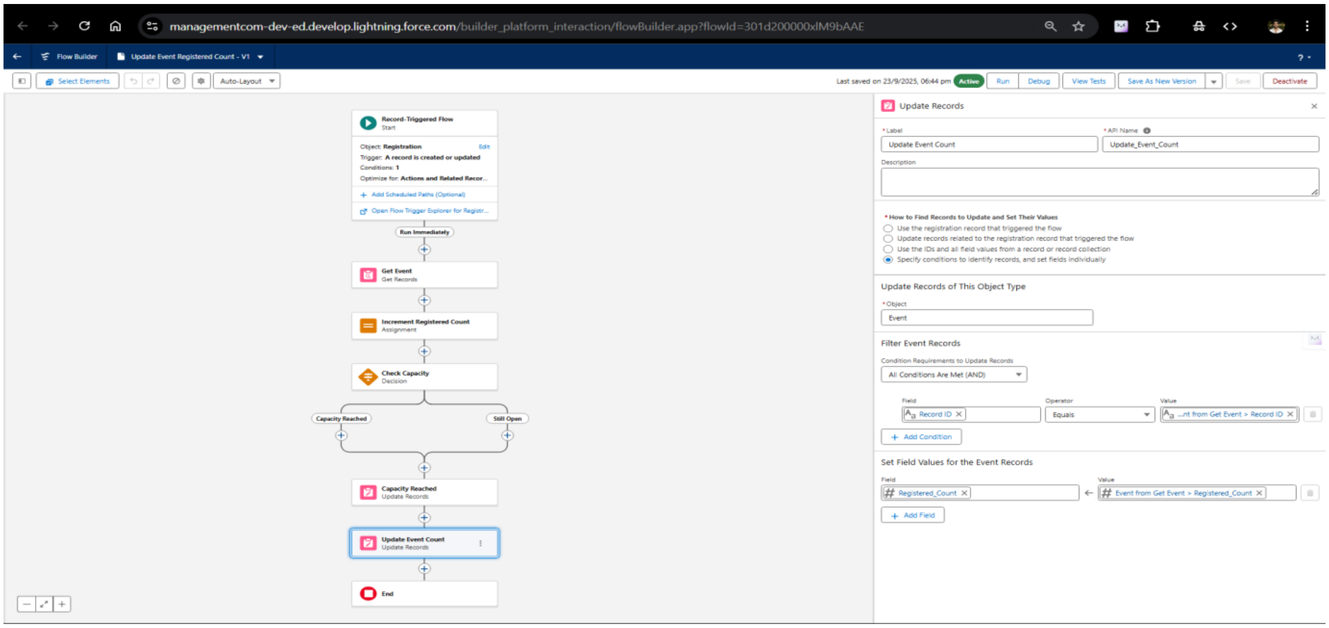
- Record-Triggered Flow

Trigger:

- When a Registration__c record is created, updated, or deleted.

Logic:

1. If Payment_Status__c transitions to "Paid" (from another status), increment Event__c.Registered_Count__c by one.
2. If Payment_Status__c changes from "Paid" to "Refunded" or if the registration is deleted, decrement Event__c.Registered_Count__c.
3. If Registered_Count__c is now greater than or equal to Capacity__c, update Event__c.Status__c to "Closed" and (optionally) alert an admin.



managementcom-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=301d200000xIM9bAAE

Last saved on 23/9/2025, 06:44 pm

Object: Registration
Trigger: A record is created or updated
Conditions: 1
Optimize for: Actions and Related Recor...

*Label: Update Event Count

*API Name: Update_Event_Count

Description:

*How to Find Records to Update and Set Their Values

○ Use the registration record that triggered the flow
○ Update records related to the registration record that triggered the flow
○ Use the IDs and all field values from a record or record collection
● Specify conditions to identify records, and set fields individually

Update Records of This Object Type

*Object: Event

Filter Event Records

Condition Requirements to Update Records: All Conditions Are Met (AND)

Field	Operator	Value
Record ID	Equals	...nt from Get Event > Record ID

Set Field Values for the Event Records

Field		Value
Registered_Count	←	Event from Get Event > Registered_Count

2.2 Flow: Email Confirmation

Purpose

- Instantly notifies the attendee of their successful registration and/or payment for transparency and engagement.

Type

- Record-Triggered Flow
  (Object: Registration__c, After Insert or Payment_Status__c update)

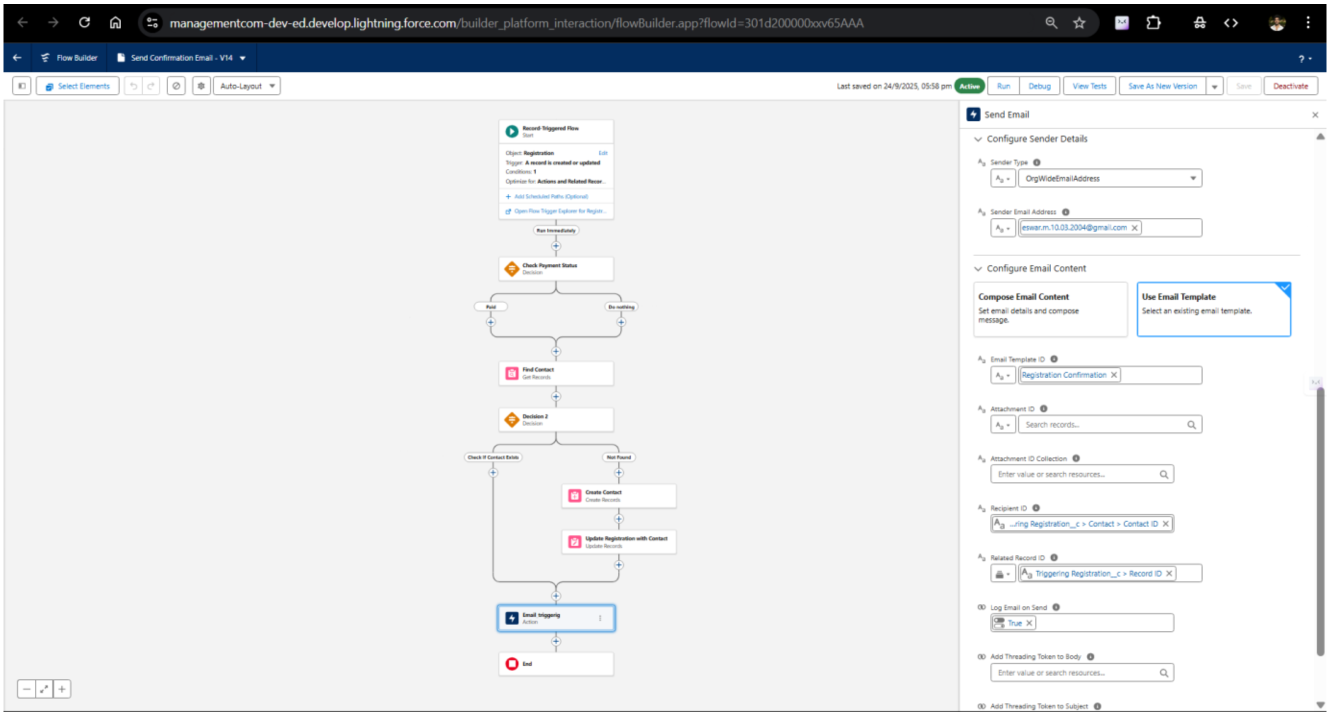
Process Breakdown

Trigger:

- New Registration__c is created, or Payment_Status__c is set to "Paid".

Logic:

1. Gather information from Registration__c, Attendee__c, and Event__c.
2. Use a Lightning Email Template—populating necessary fields with merge data.
3. Send the template-based email to the attendee's email address.



managementcom-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=301d200000xxv65AAA

Flow Builder · Send Confirmation Email - V14

Last saved on 24/9/2025, 05:58 pm · Active

Record-Triggered Flow
Start
Object: Registration
Trigger: A record is created or updated
Conditions: 1
Optimize for: Actions and Related Recor...

Run Immediately → Check Payment Status (Decision) → Paid / Do nothing → Find Contact (Get Records) → Decision 2 (Decision) → Check If Contact Exists / Not Found → Create Contact (Create Records) → Update Registration with Contact (Update Records) → Email triggerig (Action) → End

Send Email

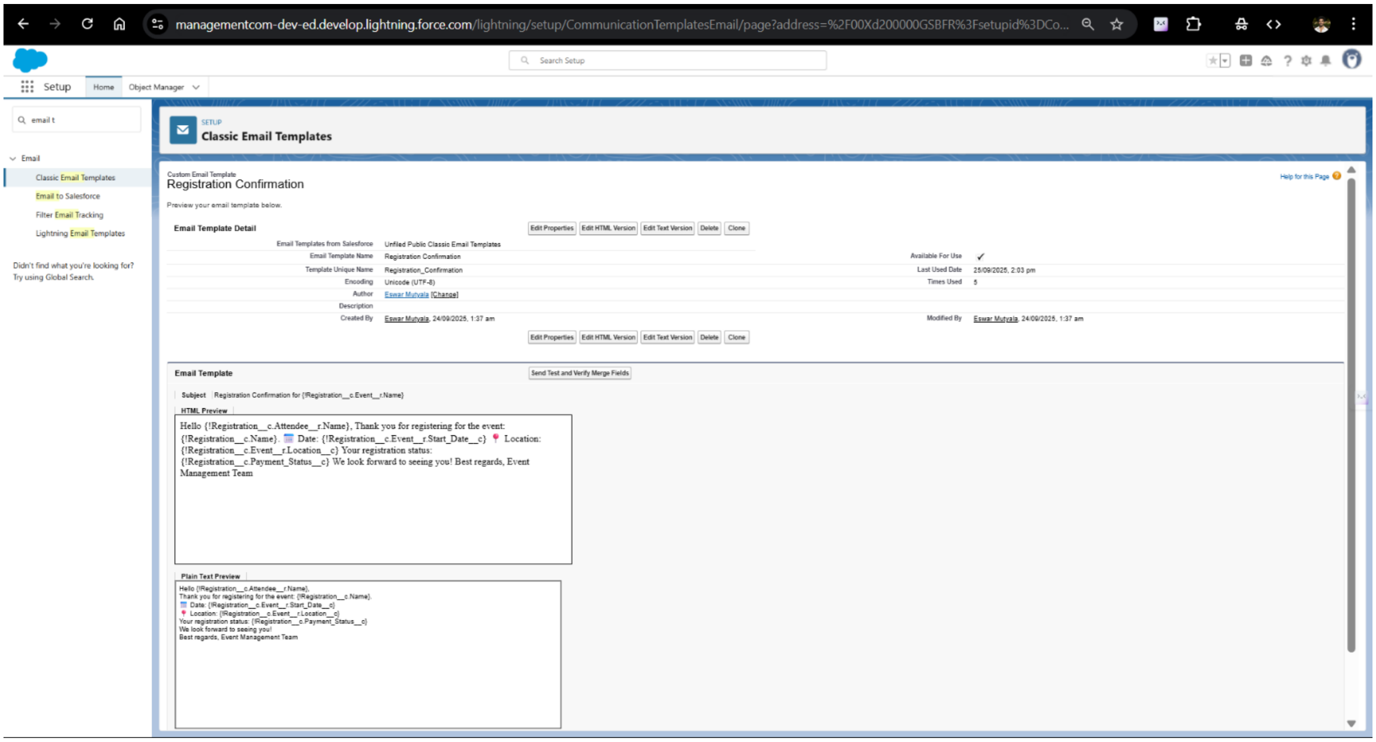
Configure Sender Details
Sender Type: OrgWideEmailAddress
Sender Email Address: eswar.m.10.03.2004@gmail.com

Configure Email Content
Compose Email Content — Set email details and compose message.
Use Email Template — Select an existing email template.
Email Template ID: Registration Confirmation
Attachment ID: Search records...
Attachment ID Collection: Enter value or search resources...
Recipient ID: ...ring Registration__c > Contact > Contact ID
Related Record ID: Triggering Registration__c > Record ID
Log Email on Send: True
Add Threading Token to Body: Enter value or search resources...
Add Threading Token to Subject

3. Email Templates & Notifications

3.1 Setup

- **Path:** Setup → Email Templates → New Email Template (Lightning).
- **Template:** Registration Confirmation
- **Merge Fields:** Event Name, Attendee First/Last Name, Date, etc.
- **Use:** Referenced by confirmation flow for automatic attendee notification.


managementcom-dev-ed.develop.lightning.force.com/lightning/setup/CommunicationTemplatesEmail/page?address=%2F00Xd200000GSBFR%3Fsetupid%3DCo...
Search Setup
Setup
Home
Object Manager
email t
Email
Classic Email Templates
Email to Salesforce
Filter Email Tracking
Lightning Email Templates
Didn't find what you're looking for? Try using Global Search.
SETUP
Classic Email Templates
Help for this Page
Edit Properties
Edit HTML Version
Edit Text Version
Delete
Clone
Send Test and Verify Merge Fields

Custom Email Template
Registration Confirmation

Preview your email template below.

Email Template Detail

Email Templates from Salesforce	Unfiled Public Classic Email Templates		
Email Template Name	Registration Confirmation	Available For Use	✓
Template Unique Name	Registration_Confirmation	Last Used Date	25/09/2025, 2:03 pm
Encoding	Unicode (UTF-8)	Times Used	5
Author	Eswar Mutyala [Change]		
Description			
Created By	Eswar Mutyala, 24/09/2025, 1:37 am	Modified By	Eswar Mutyala, 24/09/2025, 1:37 am

Email Template

Subject Registration Confirmation for {!Registration__c.Event__r.Name}

HTML Preview

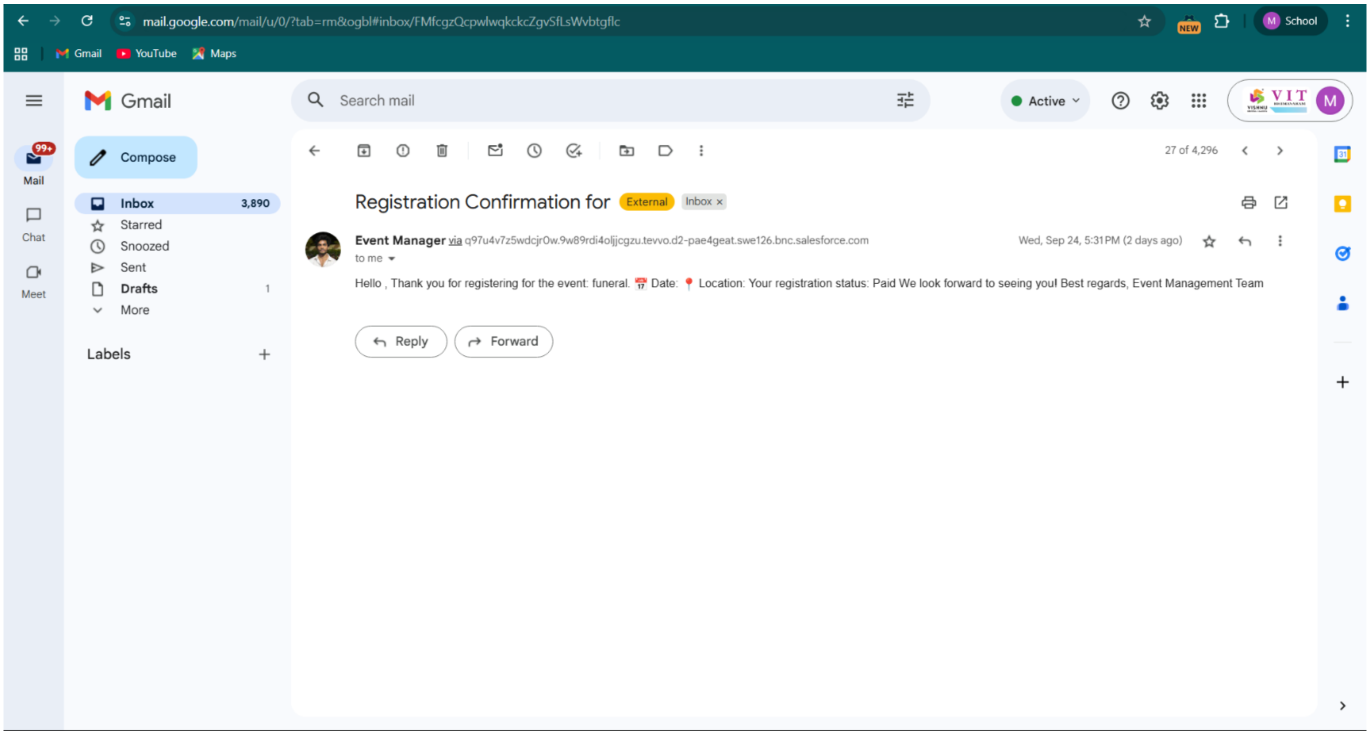
Hello {!Registration__c.Attendee__r.Name}, Thank you for registering for the event: {!Registration__c.Name}. 📅 Date: {!Registration__c.Event__r.Start_Date__c} 📍 Location: {!Registration__c.Event__r.Location__c} Your registration status: {!Registration__c.Payment_Status__c} We look forward to seeing you! Best regards, Event Management Team

Plain Text Preview

```
Hello {!Registration__c.Attendee__r.Name},
Thank you for registering for the event: {!Registration__c.Name}.
📅 Date: {!Registration__c.Event__r.Start_Date__c}
📍 Location: {!Registration__c.Event__r.Location__c}
Your registration status: {!Registration__c.Payment_Status__c}
We look forward to seeing you!
Best regards, Event Management Team
```

4. Testing & Iteration

- **Test Cases:**
  - Create event, register attendee (capacity available): should succeed, confirmation email sent.
  - Register when event is full: should block registration, with error.
  - Change payment status to Paid: Registered_Count__c should increment.
  - Refund/Cancel registration: Registered_Count__c should decrement.
- **Reporting:** Ensure Available_Seats__c and Registered_Count__c always reflect accurate numbers.
- **Iterate on flows/rules as needed after initial tests.



Registration Confirmation for External Inbox

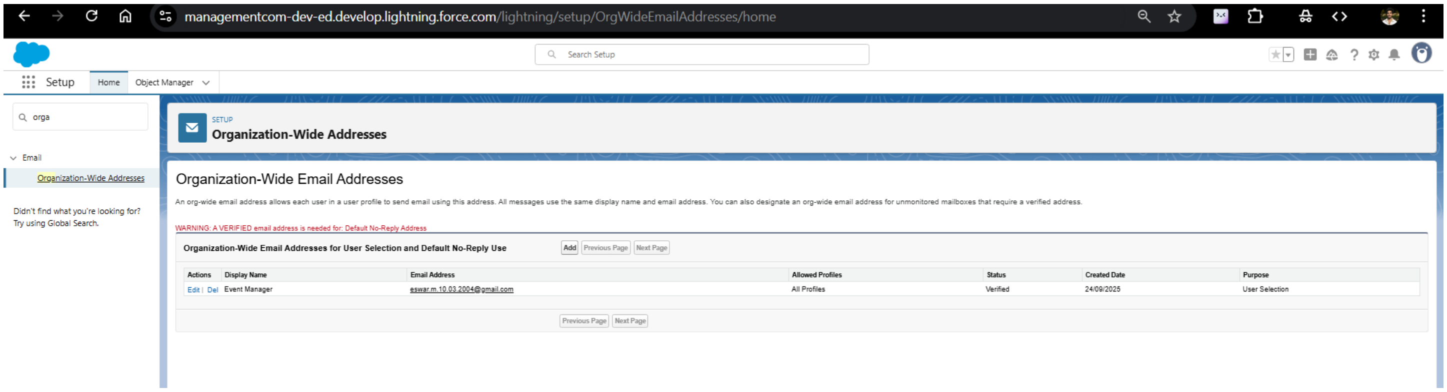
Event Manager via q97u4v7z5wdcjrOw.9w89rdi4oljjcgzu.tevvo.d2-pae4geat.swe126.bnc.salesforce.com
to me

Wed, Sep 24, 5:31 PM (2 days ago)

Hello , Thank you for registering for the event: funeral. 📅 Date: 📍 Location: Your registration status: Paid We look forward to seeing you! Best regards, Event Management Team

5. Security & Permissions (in context)

- Confirm flow and field security for each profile:
  - Only Event Admins can change status or capacity.
  - Attendees can only register (not directly change event data).
- Check OWD/sharing settings support automation.



managementcom-dev-ed.develop.lightning.force.com/lightning/setup/OrgWideEmailAddresses/home
Search Setup
Setup
Home
Object Manager
orga
Email
Organization-Wide Addresses
Didn't find what you're looking for? Try using Global Search.
SETUP

Organization-Wide Addresses

Organization-Wide Email Addresses

An org-wide email address allows each user in a user profile to send email using this address. All messages use the same display name and email address. You can also designate an org-wide email address for unmonitored mailboxes that require a verified address.

WARNING: A VERIFIED email address is needed for: Default No-Reply Address

Organization-Wide Email Addresses for User Selection and Default No-Reply Use

Actions	Display Name	Email Address	Allowed Profiles	Status	Created Date	Purpose
Edit | Del	Event Manager	eswar.m.10.03.2004@gmail.com	All Profiles	Verified	24/09/2025	User Selection

Phase 5: Apex Programming

RegistrationController

Overview

In this phase, we implement the core backend logic for event registration using Apex. The main class, RegistrationController, is exposed as an AuraEnabled method to support Lightning Web Components (LWCs) or other UI integrations by creating and managing Attendee and Registration records programmatically.

Apex Class: RegistrationController

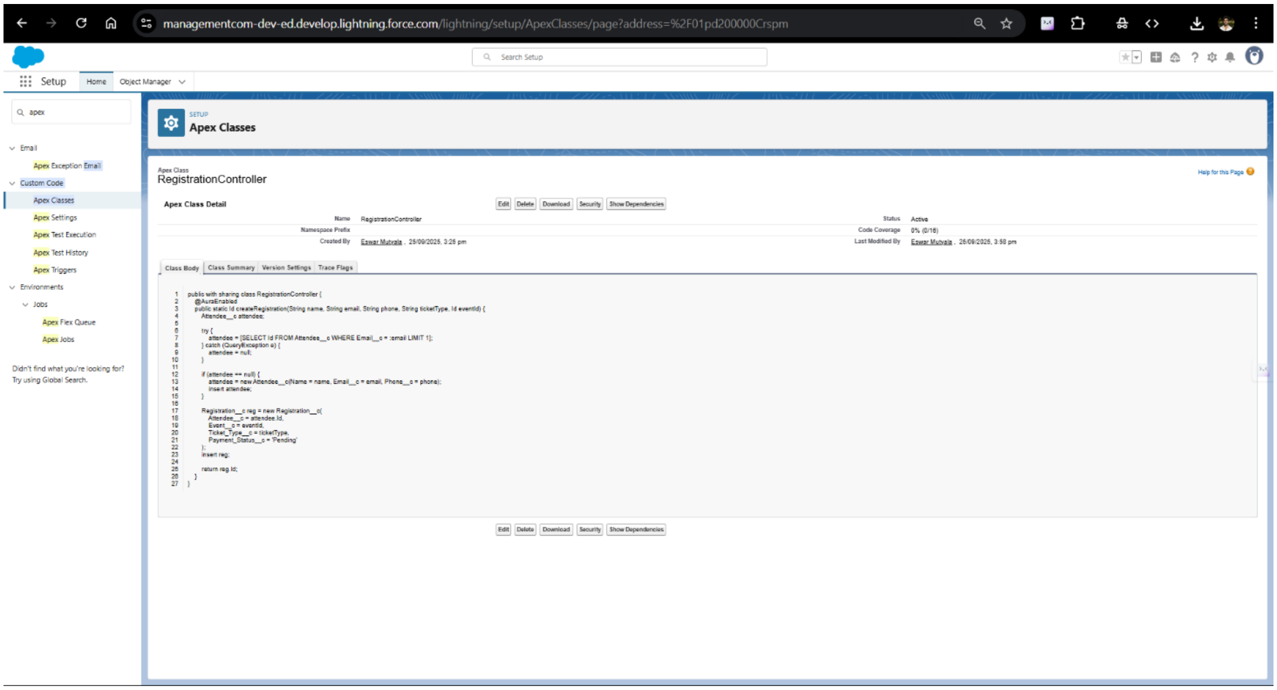
Purpose

- This Apex class handles attendee registrations in code.
- It finds or creates an Attendee record based on email.
- Creates a Registration record associated with the Event and Attendee.
- Sets the initial registration Payment Status as 'Pending'.
- Returns the Registration record Id for further use by the UI or calling components.

Code Summary

```
public with sharing class RegistrationController {
  @AuraEnabled
  public static Id createRegistration(String name, String email, String phone, String ticketType, Id eventId) {
    Attendee__c attendee;
    try {
      attendee = [SELECT Id FROM Attendee__c WHERE Email__c = :email LIMIT 1];
    } catch (QueryException e) {
      attendee = null;
    }
    if (attendee == null) {
      attendee = new Attendee__c(Name = name, Email__c = email, Phone__c = phone);
      insert attendee;
    }
    Registration__c reg = new Registration__c(
      Attendee__c = attendee.Id,
      Event__c = eventId,
      Ticket_Type__c = ticketType,
      Payment_Status__c = 'Pending'
    );
    insert reg;
    return reg.Id;
  }
}
```

Apex Class(RegistrationController)



Apex Class
RegistrationController

Apex Class Detail

Name	RegistrationController	Status	Active
Namespace Prefix		Code Coverage	0% (0/16)
Created By	Eswar Mutyala , 25/09/2025, 3:26 pm	Last Modified By	Eswar Mutyala , 25/09/2025, 3:58 pm

```
public with sharing class RegistrationController {
    @AuraEnabled
    public static Id createRegistration(String name, String email, String phone, String ticketType, Id eventId) {
        Attendee__c attendee;

        try {
            attendee = [SELECT Id FROM Attendee__c WHERE Email__c = :email LIMIT 1];
        } catch (QueryException e) {
            attendee = null;
        }

        if (attendee == null) {
            attendee = new Attendee__c(Name = name, Email__c = email, Phone__c = phone);
            insert attendee;
        }

        Registration__c reg = new Registration__c(
            Attendee__c = attendee.Id,
            Event__c = eventId,
            Ticket_Type__c = ticketType,
            Payment_Status__c = 'Pending'
        );
        insert reg;

        return reg.Id;
    }
}
```

Usage in Project

- Called by Lightning Web Component (registrationForm) to create registrations from user input.
- Enables registration processing through Lightning UI and automates record creation.
- Forms the basis for further Apex enhancements such as capacity checking or payment integration

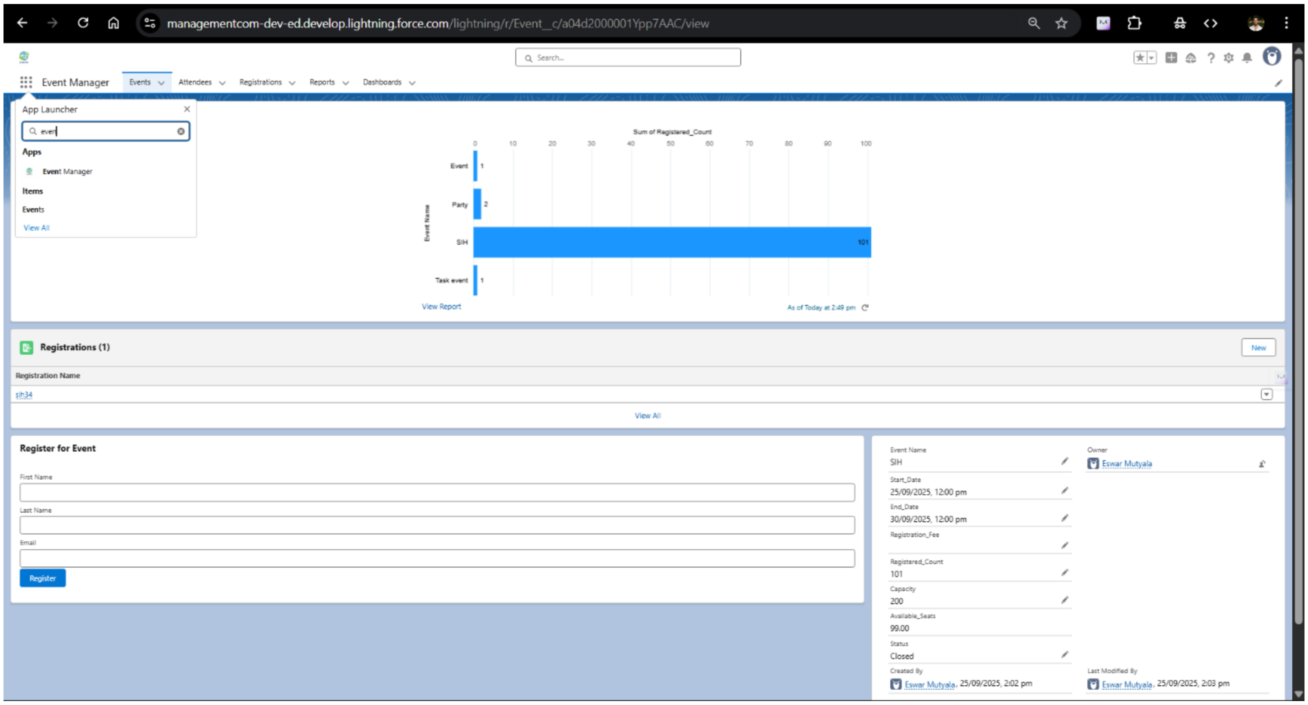
Phase 6: User Interface Development

Overview

In this phase, I developed a user-friendly Lightning Experience interface enabling event management and attendee registration via intuitive navigation and interactive components. The process involved creating a dedicated Lightning App, customizing record pages, building Lightning Web Components integrated with Apex, and configuring tabs and utility features to enhance usability.

1. Created Lightning App - Event Manager

- Navigated to Setup → App Manager → New Lightning App.
- Defined the app as "Event Manager," with appropriate branding and standard navigation.
- Added custom object tabs for Event__c, Registration__c, and Attendee__c, along with Reports to the app's navigation.
- Assigned app visibility to my System Administrator profile.
- Saved and activated the app for use.



Registration Name
sih34

First Name

Last Name

Email

Event Name
SIH

Owner
Eswar Mutyala

Start_Date
25/09/2025, 12:00 pm

End_Date
30/09/2025, 12:00 pm

Registration_Fee

Registered_Count
101

Capacity
200

Available_Seats
99.00

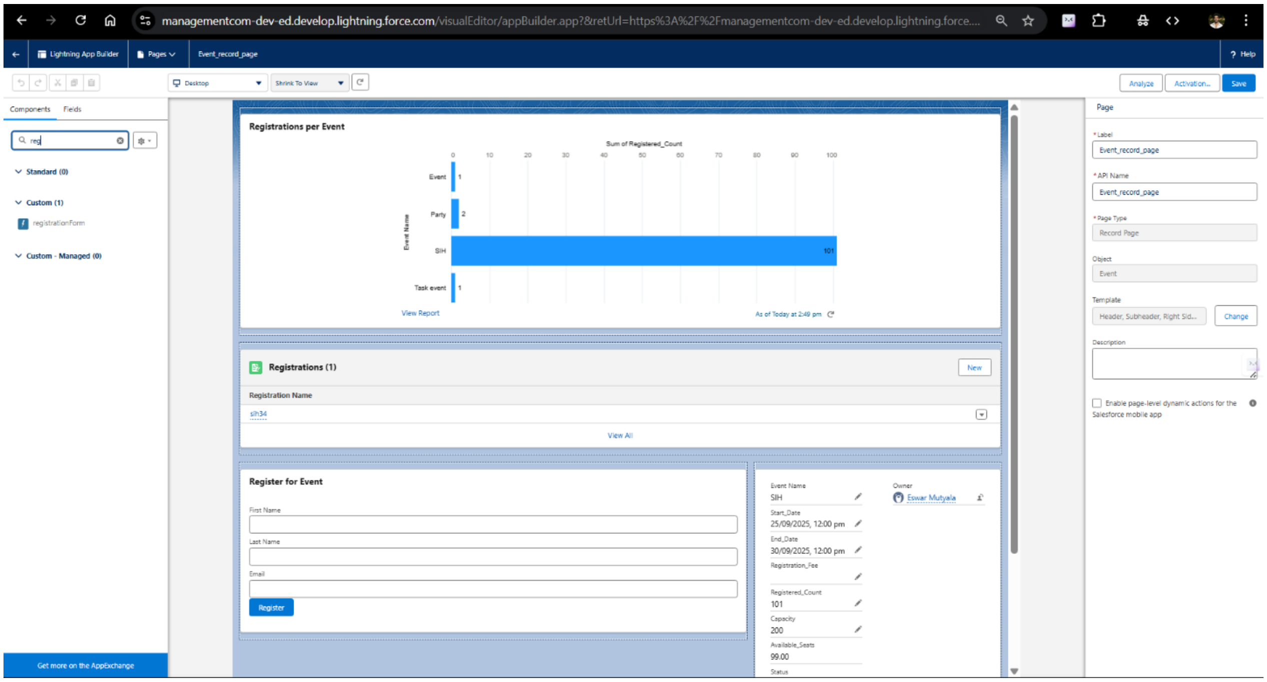
Status
Closed

Created By
Eswar Mutyala, 25/09/2025, 2:02 pm

Last Modified By
Eswar Mutyala, 25/09/2025, 2:03 pm

2. Customized Lightning Record Page for Event__c

- Opened an Event record and selected **Edit Page** in the Lightning App Builder.
- Added the **Record Detail** component to display key event fields including Name, Capacity, Status, and Dates.
- Added a **Related List - Single** to show related Registrations, allowing quick access to attendees linked to the event.
- Incorporated a chart component to visually represent Registered_Count__c versus Capacity__c.
- Saved and activated this page layout; assigned as the default Lightning Record Page for the Event object.



Registration Name: sih34

First Name

Last Name

Email

Event Name: SIH

Owner: Eswar Mutyala

Start_Date: 25/09/2025, 12:00 pm

End_Date: 30/09/2025, 12:00 pm

Registration_Fee

Registered_Count: 101

Capacity: 200

Available_Seats: 99.00

Status

* Label: Event_record_page

* API Name: Event_record_page

* Page Type: Record Page

Object: Event

Template: Header, Subheader, Right Sid...

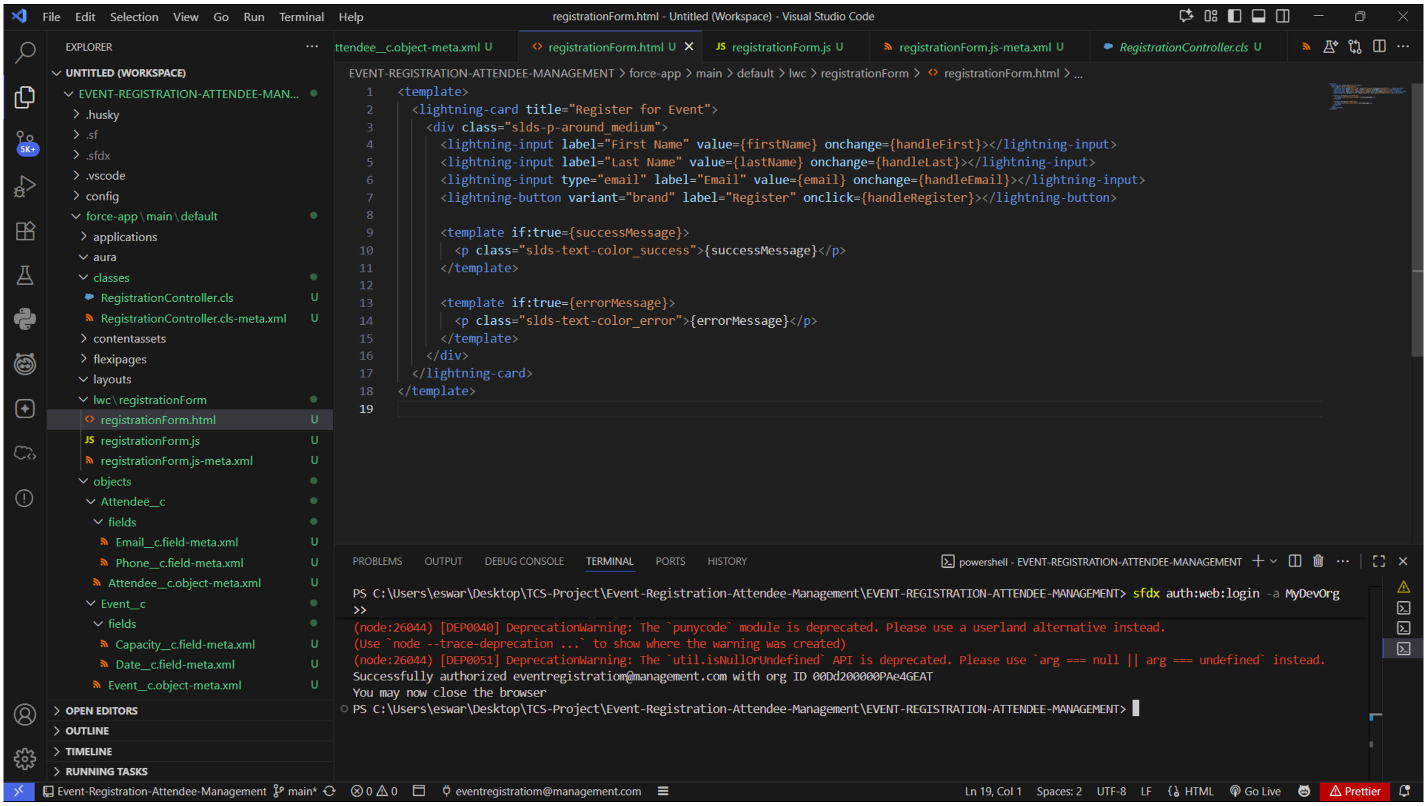
Description

□ Enable page-level dynamic actions for the Salesforce mobile app

3. Developed Lightning Web Component: registrationForm

- Created the registrationForm LWC using Salesforce CLI and VS Code.
- The component UI includes:
  - Inputs for Full Name, Email, Phone.
  - Ticket Type dropdown with options Standard, VIP, Student.
  - Submit button for registration.
- Managed state and input handling in registrationForm.js.
- Implemented imperative Apex calls to backend RegistrationController class to:
  - Find or create Attendee records.
  - Create a linked Registration record with Payment_Status__c defaulted to 'Pending'.
- Handled success and error feedback via Lightning toast notifications.
- Placed the registrationForm component on the Event Lightning Record Page using App Builder.
- Activated changes, enabling inline attendee registration from the Event page.

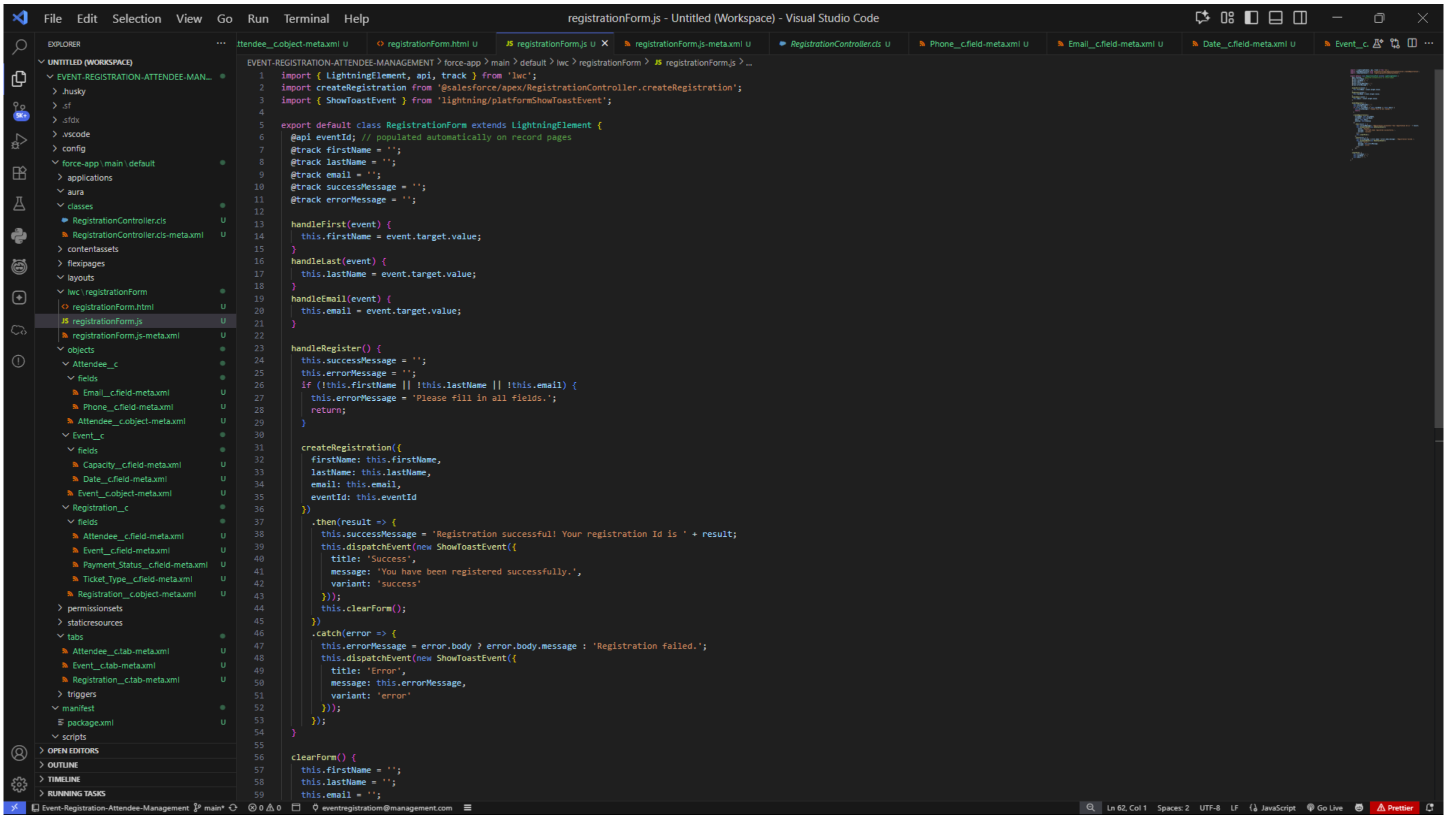
registrationForm.html



```html
<template>
  <lightning-card title="Register for Event">
    <div class="slds-p-around_medium">
      <lightning-input label="First Name" value={firstName} onchange={handleFirst}></lightning-input>
      <lightning-input label="Last Name" value={lastName} onchange={handleLast}></lightning-input>
      <lightning-input type="email" label="Email" value={email} onchange={handleEmail}></lightning-input>
      <lightning-button variant="brand" label="Register" onclick={handleRegister}></lightning-button>

      <template if:true={successMessage}>
        <p class="slds-text-color_success">{successMessage}</p>
      </template>

      <template if:true={errorMessage}>
        <p class="slds-text-color_error">{errorMessage}</p>
      </template>
    </div>
  </lightning-card>
</template>

```

```
PS C:\Users\eswar\Desktop\TCS-Project\Event-Registration-Attendee-Management\EVENT-REGISTRATION-ATTENDEE-MANAGEMENT> sfdx auth:web:login -a MyDevOrg
>>
(node:26044) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
(node:26044) [DEP0051] DeprecationWarning: The `util.isNullOrUndefined` API is deprecated. Please use `arg === null || arg === undefined` instead.
Successfully authorized eventregistratiom@management.com with org ID 00Dd200000PAe4GEAT
You may now close the browser
PS C:\Users\eswar\Desktop\TCS-Project\Event-Registration-Attendee-Management\EVENT-REGISTRATION-ATTENDEE-MANAGEMENT>
```

registrationForm.js



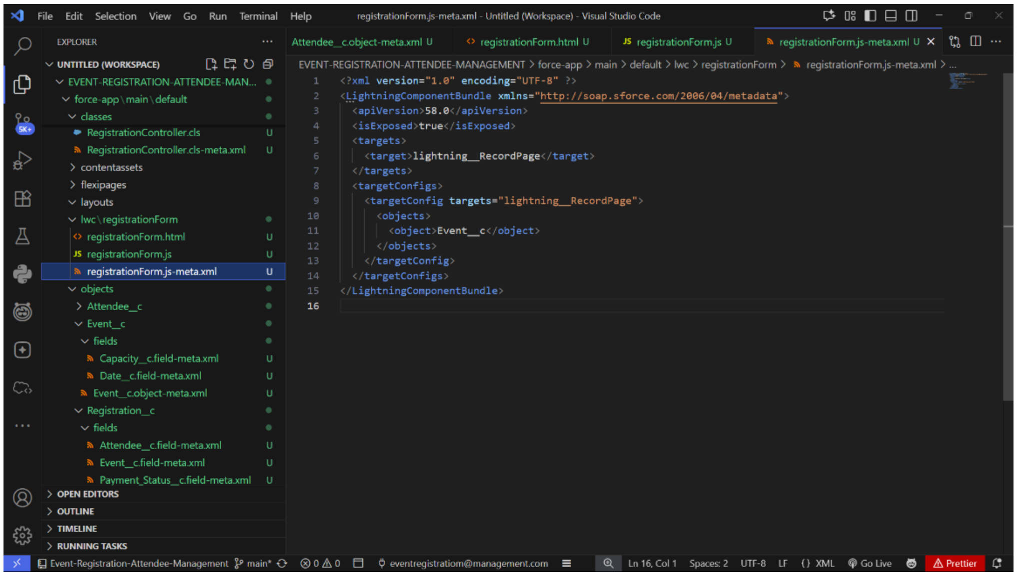
```javascript
import { LightningElement, api, track } from 'lwc';
import createRegistration from '@salesforce/apex/RegistrationController.createRegistration';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';

export default class RegistrationForm extends LightningElement {
  @api eventId; // populated automatically on record pages
  @track firstName = '';
  @track lastName = '';
  @track email = '';
  @track successMessage = '';
  @track errorMessage = '';

  handleFirst(event) {
    this.firstName = event.target.value;
  }
  handleLast(event) {
    this.lastName = event.target.value;
  }
  handleEmail(event) {
    this.email = event.target.value;
  }

  handleRegister() {
    this.successMessage = '';
    this.errorMessage = '';
    if (!this.firstName || !this.lastName || !this.email) {
      this.errorMessage = 'Please fill in all fields.';
      return;
    }

    createRegistration({
      firstName: this.firstName,
      lastName: this.lastName,
      email: this.email,
      eventId: this.eventId
    })
      .then(result => {
        this.successMessage = 'Registration successful! Your registration Id is ' + result;
        this.dispatchEvent(new ShowToastEvent({
          title: 'Success',
          message: 'You have been registered successfully.',
          variant: 'success'
        }));
        this.clearForm();
      })
      .catch(error => {
        this.errorMessage = error.body ? error.body.message : 'Registration failed.';
        this.dispatchEvent(new ShowToastEvent({
          title: 'Error',
          message: this.errorMessage,
          variant: 'error'
        }));
      });
  }

  clearForm() {
    this.firstName = '';
    this.lastName = '';
    this.email = '';
```

4. Custom Objects & Fields: SFDX Source-Driven Development

Instead of relying solely on Object Manager, I defined and managed all core objects and fields as metadata using Salesforce DX and Visual Studio Code:

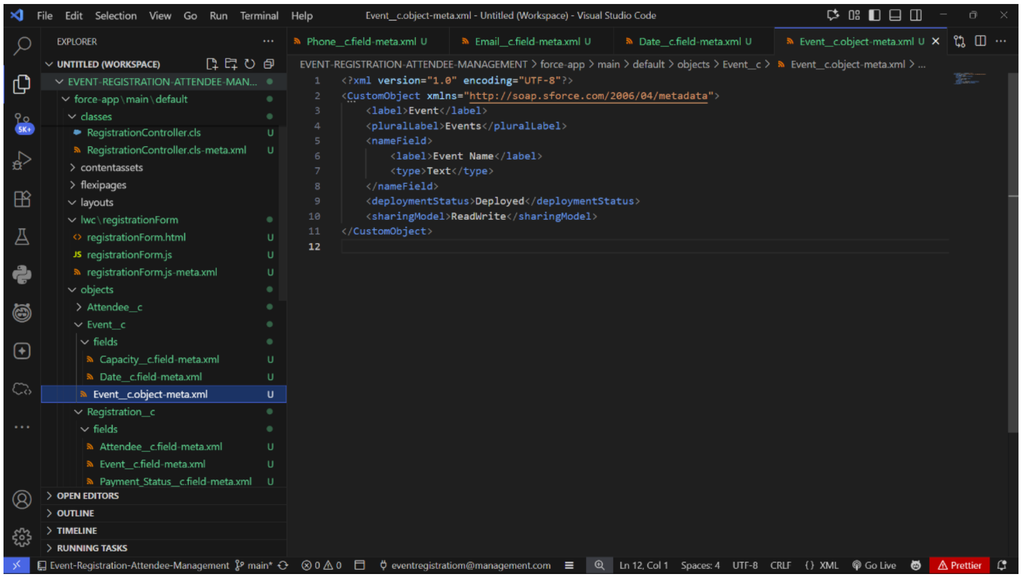
- Created folder structure for force-app/main/default/objects/
- Defined metadata XML for:
    - **Event__c** (including fields like Name, Date__c, Capacity__c)
    - **Attendee__c** (fields: Name, Email__c, Phone__c)
    - **Registration__c** (fields: Attendee__c, Event__c, Ticket_Type__c, Payment_Status__c)

Registration:


registrationForm.js-meta.xml - Untitled (Workspace) - Visual Studio Code
Attendee__c.object-meta.xml U
registrationForm.html U
registrationForm.js U
registrationForm.js-meta.xml U
EVENT-REGISTRATION-ATTENDEE-MANAGEMENT > force-app > main > default > lwc > registrationForm > registrationForm.js-meta.xml > ...

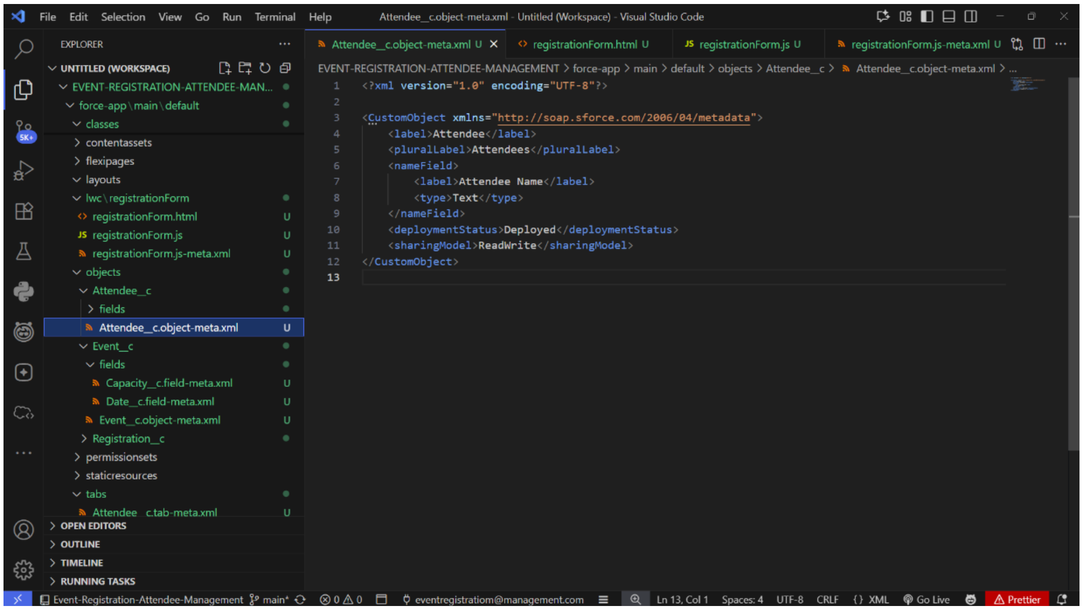
```xml
<?xml version="1.0" encoding="UTF-8" ?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>58.0</apiVersion>
  <isExposed>true</isExposed>
  <targets>
    <target>lightning__RecordPage</target>
  </targets>
  <targetConfigs>
    <targetConfig targets="lightning__RecordPage">
      <objects>
        <object>Event__c</object>
      </objects>
    </targetConfig>
  </targetConfigs>
</LightningComponentBundle>
```

Event:-


Event__c.object-meta.xml - Untitled (Workspace) - Visual Studio Code
Phone__c.field-meta.xml U
Email__c.field-meta.xml U
Date__c.field-meta.xml U
Event__c.object-meta.xml U
EVENT-REGISTRATION-ATTENDEE-MANAGEMENT > force-app > main > default > objects > Event__c > Event__c.object-meta.xml > ...

```xml
<?xml version="1.0" encoding="UTF-8"?>
<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>Event</label>
    <pluralLabel>Events</pluralLabel>
    <nameField>
        <label>Event Name</label>
        <type>Text</type>
    </nameField>
    <deploymentStatus>Deployed</deploymentStatus>
    <sharingModel>ReadWrite</sharingModel>
</CustomObject>
```

Attendee:-


File Edit Selection View Go Run Terminal Help
Attendee__c.object-meta.xml - Untitled (Workspace) - Visual Studio Code
EXPLORER
UNTITLED (WORKSPACE)
EVENT-REGISTRATION-ATTENDEE-MAN...
force-app\main\default
classes
contentassets
flexipages
layouts
lwc\registrationForm
registrationForm.html
registrationForm.js
registrationForm.js-meta.xml
objects
Attendee__c
fields
Attendee__c.object-meta.xml
Event__c
fields
Capacity__c.field-meta.xml
Date__c.field-meta.xml
Event__c.object-meta.xml
Registration__c
permissionsets
staticresources
tabs
Attendee__c.tab-meta.xml
OPEN EDITORS
OUTLINE
TIMELINE
RUNNING TASKS
Attendee__c.object-meta.xml
registrationForm.html
registrationForm.js
registrationForm.js-meta.xml
EVENT-REGISTRATION-ATTENDEE-MANAGEMENT > force-app > main > default > objects > Attendee__c > Attendee__c.object-meta.xml > ...
Event-Registration-Attendee-Management main* 0 0 eventregistratiom@management.com Ln 13, Col 1 Spaces: 4 UTF-8 CRLF XML Go Live Prettier

```xml
<?xml version="1.0" encoding="UTF-8"?>

<CustomObject xmlns="http://soap.sforce.com/2006/04/metadata">
    <label>Attendee</label>
    <pluralLabel>Attendees</pluralLabel>
    <nameField>
        <label>Attendee Name</label>
        <type>Text</type>
    </nameField>
    <deploymentStatus>Deployed</deploymentStatus>
    <sharingModel>ReadWrite</sharingModel>
</CustomObject>
```

Each individual field and object was defined in a .object-meta.xml or .field-meta.xml file, all tracked by source control for easy deployment.

Phase 7: Integration & External Access

Event Registration & Attendee Management — Student Project

Objective

The objective of this phase was to enable integration of the Salesforce Event Registration system with external services to add value beyond Salesforce's native functionality. While the core application provides powerful features, linking to external systems and tools helps automate workflows and improve user efficiency.

7.1 Implemented Feature: Custom Link for External Event Lookup

To demonstrate a straightforward, user-initiated external integration, we implemented a **custom button** on the Event record page to facilitate quick event-related searches on an external platform.

- **Component:** Custom Button — Event Web Search
- **Purpose:**
  Provides event managers and users with one-click access to external web resources (e.g., Google or another event information site) about the event, helping validate or enrich event details easily.
- **Implementation Details:**

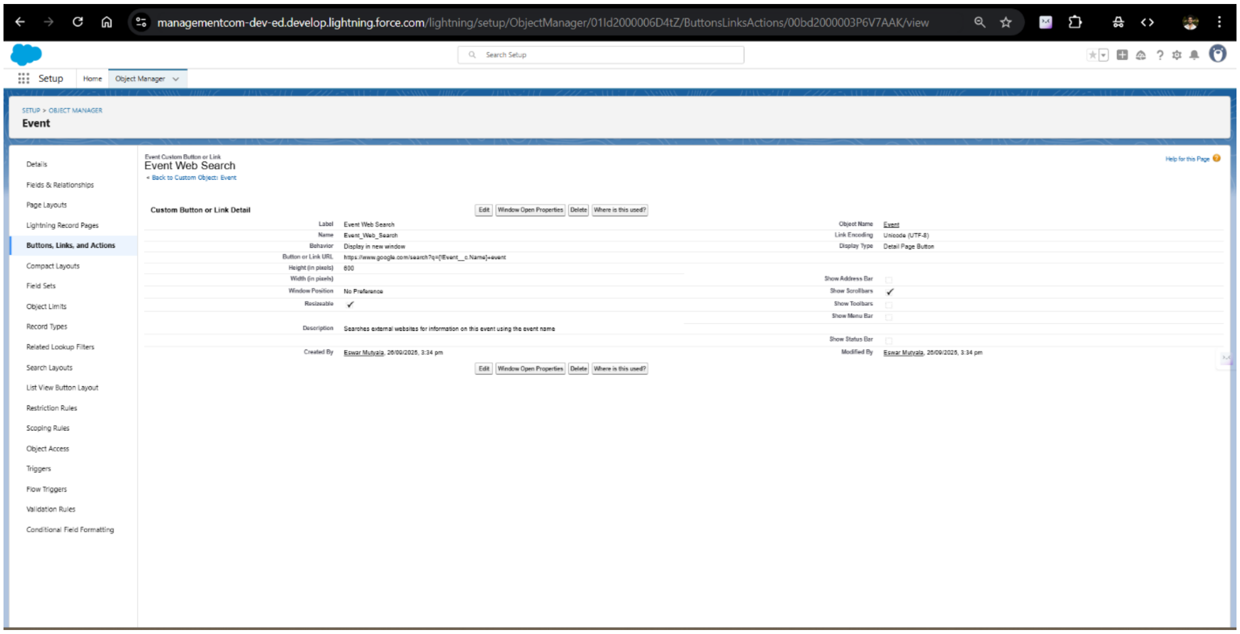
- **Object:** Event__c
- **Display Type:** Detail Page Button
- **Behavior:** Opens the external link in a new browser tab/window
- **Content Source:** Defined as a URL with dynamic merge field
- **URL Formula:**

text

https://www.google.com/search?q={!Event__c.Name}+event

- This URL leverages the merge field {!Event__c.Name} to dynamically include the current event's name into the search query, making the button context-aware.

Users can click this button from any Event record page to instantly search related event information externally, streamlining the workflow and reducing manual lookup time.



Event Custom Button or Link

Event Web Search

« Back to Custom Object: Event

Custom Button or Link Detail

Label	Event Web Search	Object Name	Event
Name	Event_Web_Search	Link Encoding	Unicode (UTF-8)
Behavior	Display in new window	Display Type	Detail Page Button
Button or Link URL	https://www.google.com/search?q={!Event__c.Name}+event		
Height (in pixels)	600		
Width (in pixels)		Show Address Bar	□
Window Position	No Preference	Show Scrollbars	✓
Resizeable	✓	Show Toolbars	□
		Show Menu Bar	□
Description	Searches external websites for information on this event using the event name		
		Show Status Bar	□
Created By	Eswar Mutyala, 26/09/2025, 3:34 pm	Modified By	Eswar Mutyala, 26/09/2025, 3:34 pm

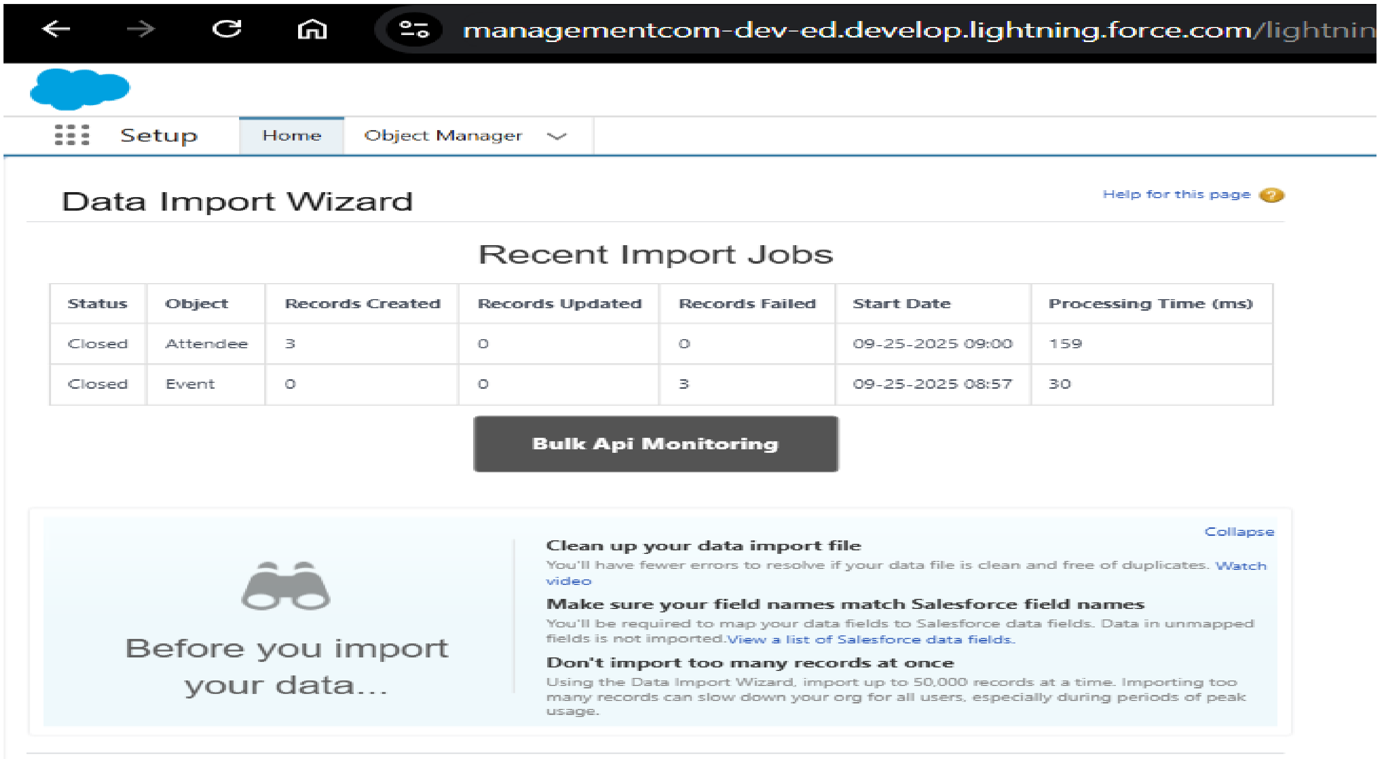
7.2 Future Enhancements

To evolve into a full enterprise-grade integration platform, future phases can include:

- Seamless payment gateway API integration (e.g., Stripe, PayPal) for live event ticketing.
- Real-time availability and pricing updates via external ERP or ticketing systems.
- Automated background checks or attendee validation services integrated via Apex callouts.
- Bidirectional data synchronization with marketing automation platforms.
- Enhanced event analytics via external BI tools using Salesforce Connect or REST APIs.

Phase 8 – Data Management & Deployment

1. **Data Import Wizard (Attendees & Events)**
   - Prepared **CSV files** with sample data for testing:
     - **Events.csv** → Event Name, Date, Capacity.
     - **Attendees.csv** → Attendee Name, Email.
   - Used **Data Import Wizard** (Setup → Data → Data Import Wizard) to load **Event__c** and **Attendee__c** records into Salesforce.
   - Verified imported data by checking the respective object tabs.


managementcom-dev-ed.develop.lightning.force.com/lightnin
Setup
Home
Object Manager
Help for this page
Bulk Api Monitoring
Collapse
Before you import your data...

Data Import Wizard

Recent Import Jobs

Status	Object	Records Created	Records Updated	Records Failed	Start Date	Processing Time (ms)
Closed	Attendee	3	0	0	09-25-2025 09:00	159
Closed	Event	0	0	3	09-25-2025 08:57	30

Clean up your data import file
You'll have fewer errors to resolve if your data file is clean and free of duplicates. Watch video

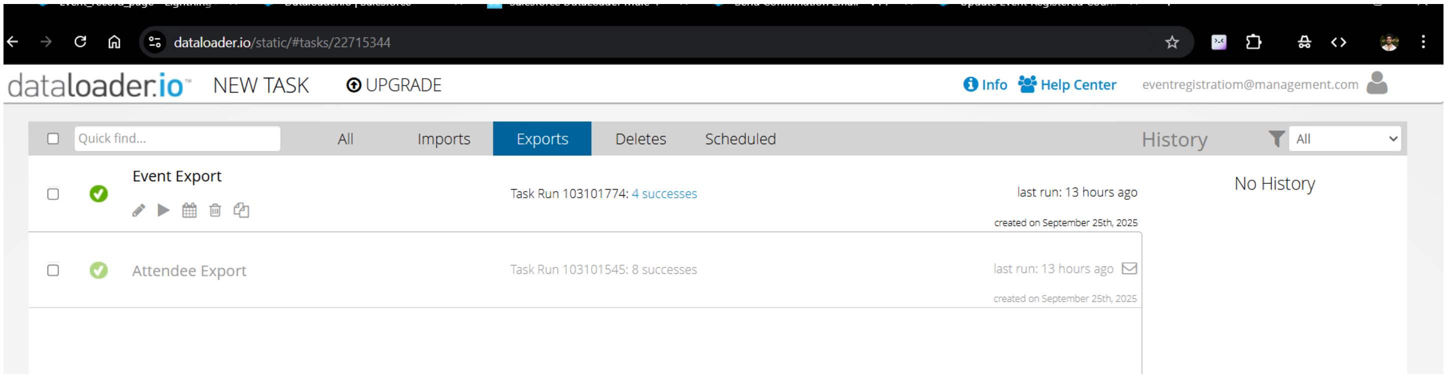
Make sure your field names match Salesforce field names
You'll be required to map your data fields to Salesforce data fields. Data in unmapped fields is not imported.View a list of Salesforce data fields.

Don't import too many records at once
Using the Data Import Wizard, import up to 50,000 records at a time. Importing too many records can slow down your org for all users, especially during periods of peak usage.

   -

2. **Data Loader (Registrations)**
   - Since **Registration__c** object had lookup relationships to **Event__c** and **Attendee__c**, used **Data Loader** for import.
   - Steps followed:
     - Exported Event__c and Attendee__c records to get their **record IDs**.
     - Mapped those IDs into **Registrations.csv** → Columns: Event__c, Attendee__c, Amount_Paid__c.
     - Imported registrations using **Insert** operation in Data Loader.

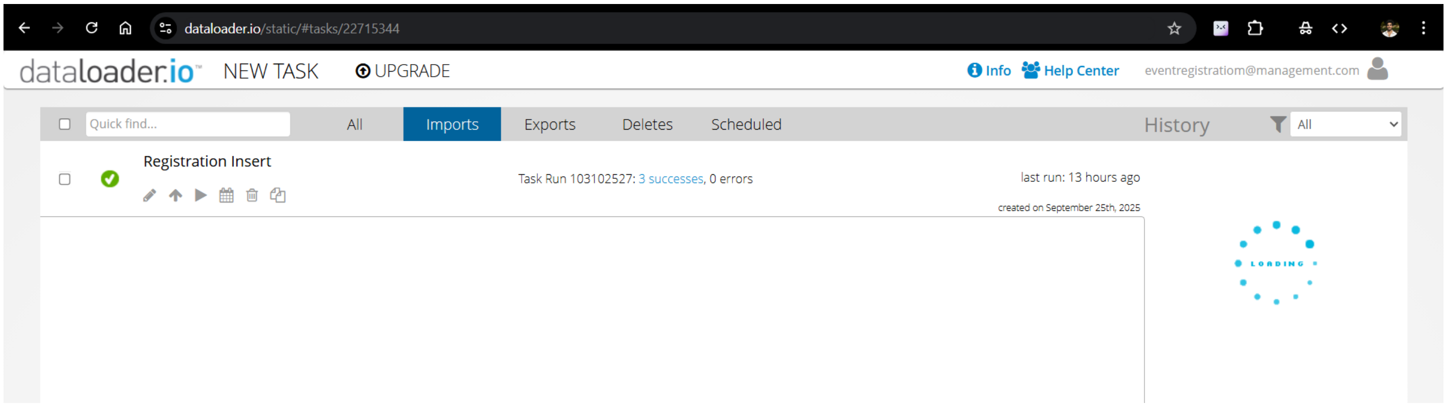
- Validated that registrations were properly linked to Events and Attendees.



Event Export — Task Run 103101774: 4 successes — last run: 13 hours ago — created on September 25th, 2025

Attendee Export — Task Run 103101545: 8 successes — last run: 13 hours ago — created on September 25th, 2025

No History



Registration Insert — Task Run 103102527: 3 successes, 0 errors — last run: 13 hours ago — created on September 25th, 2025

3. **Duplicate Rules (Planned for Production)**
   - Identified potential need for duplicate prevention (e.g., same attendee registering multiple times for the same event).
   - Documented plan to configure **Duplicate Rules** on Attendee__c (based on Email) and Registration__c (based on Attendee + Event combination).
4. **Data Export & Backup**
   - Used **Data Export Service** (Setup → Data Export) to take a backup of all records for Events, Attendees, and Registrations after test data import.
   - Exported files were stored locally as a backup for recovery.

Phase 9: Reporting, Dashboards & Security Review

Objective

The objective of this phase was twofold:

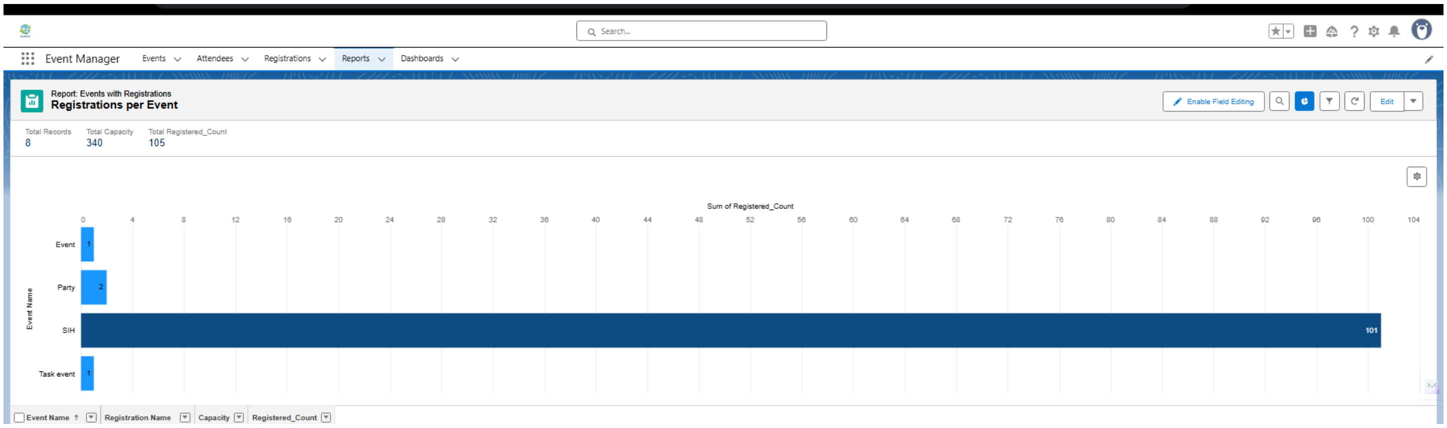
1. To transform the raw data from the **Event Management application** into actionable insights through **reports and dashboards**.
2. To review and confirm the **application's security settings**, ensuring that event data (registrations, revenue, and attendee status) is protected and accessible only to authorized users.

9.1 Reports

Reports in Salesforce were used to query event-related data, apply filters, group results, and provide summaries. For this project, three key reports were created to help **Event Managers and Organizers** monitor the success of events.

Report 1: Registrations by Event

- **Purpose:** To provide a quick overview of how many attendees have registered for each event.
- **Report Type:** Summary Report based on the custom report type **Events with Registrations**.
- **Configuration:** Grouped by *Event Name* and summarized by the **count of Registration records**.
- **Visualization:** A **Bar Chart** was added for an at-a-glance view of registrations across different events.



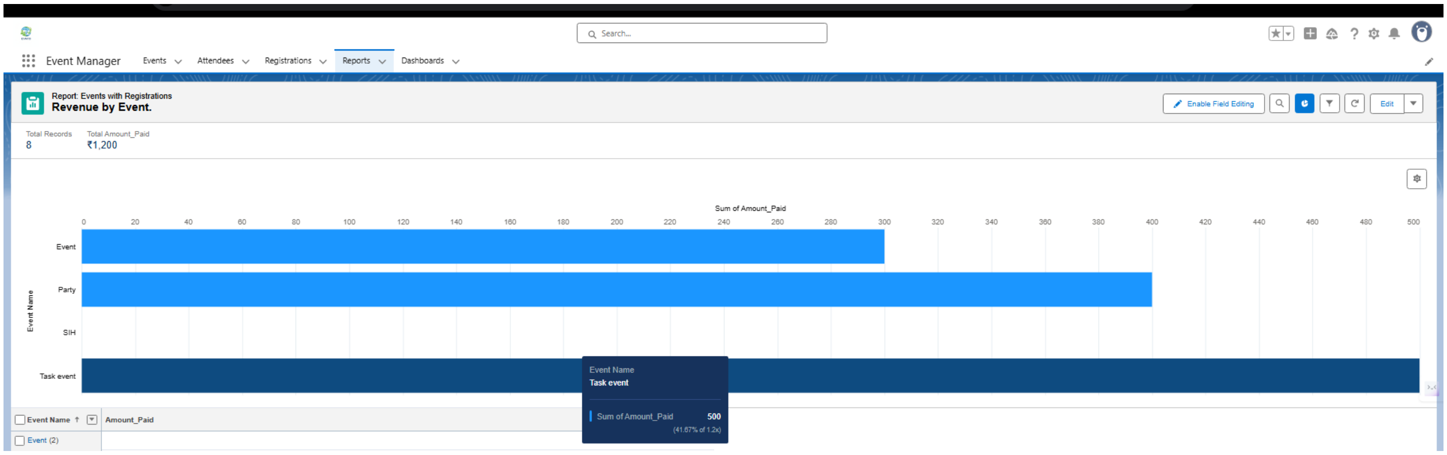
Report: Events with Registrations
Registrations per Event

Total Records 8
Total Capacity 340
Total Registered_Count 105

Event Name ↑	Registration Name	Capacity	Registered_Count

Report 2: Revenue by Event

- **Purpose:** To measure the total revenue generated from registrations for each event.
- **Report Type:** Summary Report using **Events with Registrations**.
- **Configuration:** Grouped by *Event Name*, with a summary field calculating the **SUM of Amount_Paid__c**.
- **Visualization:** A **Number Widget** was added to highlight total revenue per event.



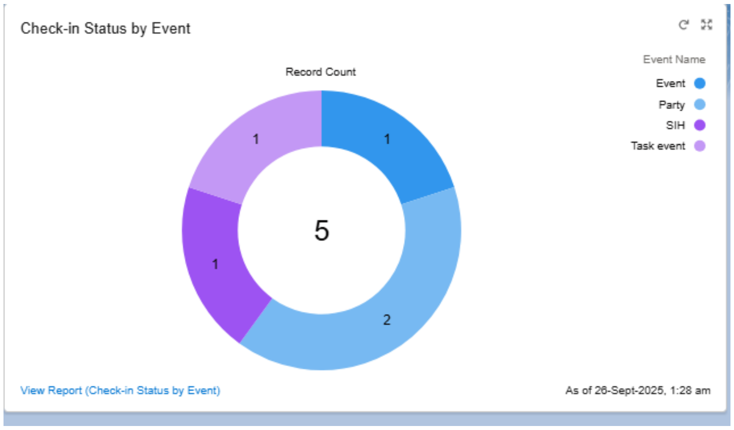
Report: Events with Registrations
Revenue by Event.

Total Records: 8
Total Amount_Paid: ₹1,200

Event Name ↑	Amount_Paid
Event (2)	

Report 3: Check-in Status by Event

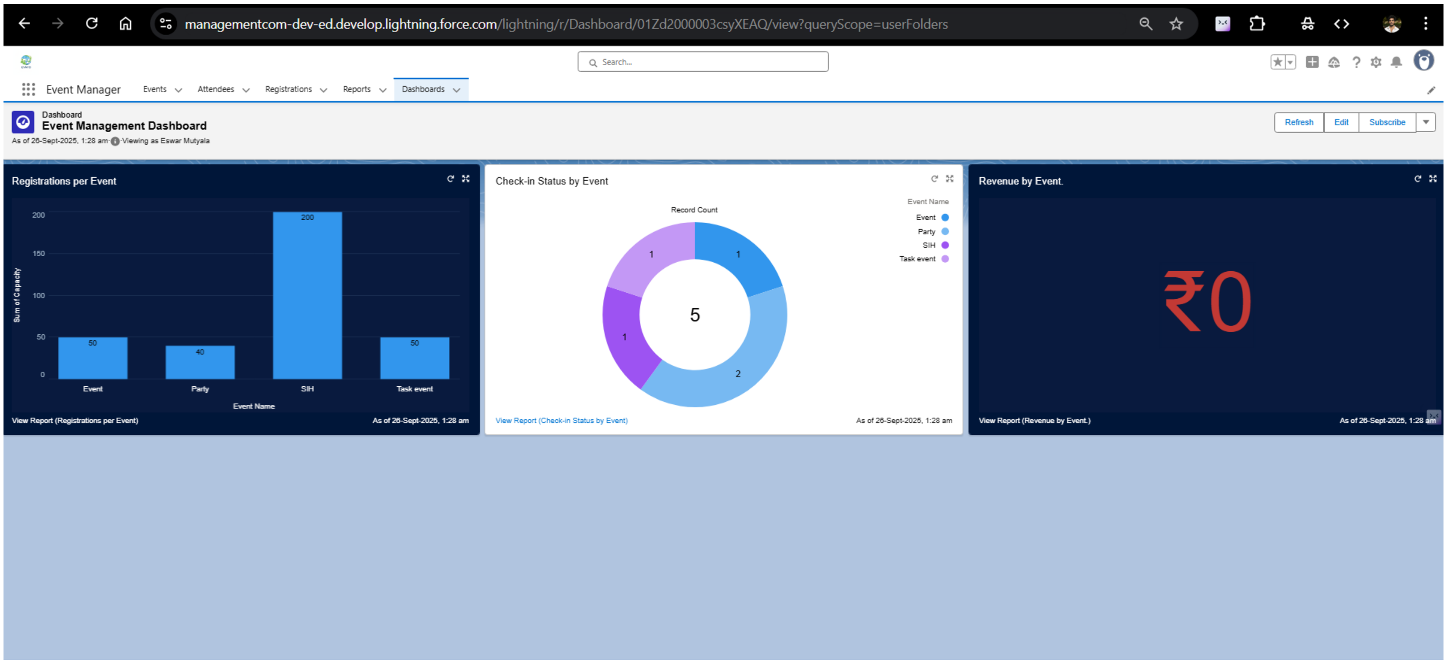
- **Purpose:** To analyze attendee participation by tracking who checked in vs. who did not.
- **Report Type:** Matrix Report using **Events with Registrations**.
- **Configuration:** Grouped by *Event Name* and then by *Check_in_Status__c*.
- **Visualization:** A **Donut Chart** was used to clearly show the proportion of attendees who checked in vs. those who did not.



9.2 Dashboard

Event Management Dashboard

- **Purpose:** To consolidate event performance metrics into a single, visual, and easy-to-read dashboard.
- **Components Added:**
  1. **Bar Chart** → Registrations by Event.
  2. **Number Widget** → Revenue by Event.
  3. **Donut Chart** → Check-in Status by Event.
- **Outcome:** Event Managers and Organizers can now instantly see **event popularity, financial performance, and attendee engagement** without manually analyzing records.



managementcom-dev-ed.develop.lightning.force.com/lightning/r/Dashboard/01Zd2000003csyXEAQ/view?queryScope=userFolders

Dashboard
Event Management Dashboard
As of 26-Sept-2025, 1:28 am · Viewing as Eswar Mutyala

9.3 Security Review

Security settings were reviewed to ensure proper access control and data protection.

- **Sharing Settings:**
  - Org-Wide Defaults (OWD) were set to **Private** for custom objects (*Event, Registration, Attendee*).
  - Sharing Rules were documented to allow Organizers access only to their assigned events.
- **Field-Level Security (FLS):**
  - Sensitive fields such as *Amount_Paid__c* were restricted to Event Managers and Admins.
  - Attendees had read-only access to event details but not to financial data.

- **Session Settings & Login Policies:**
    - Default Salesforce session timeout and password policies were applied for added security.
    - No external login IP restrictions were applied in development, but they were noted as a best practice for production.
- **Audit Trail:**
    - The Salesforce Audit Trail feature was reviewed to ensure tracking of configuration changes within the org.