Project Report – Event Registration App in Salesforce

# Phase 2 – Organization Setup and Configuration

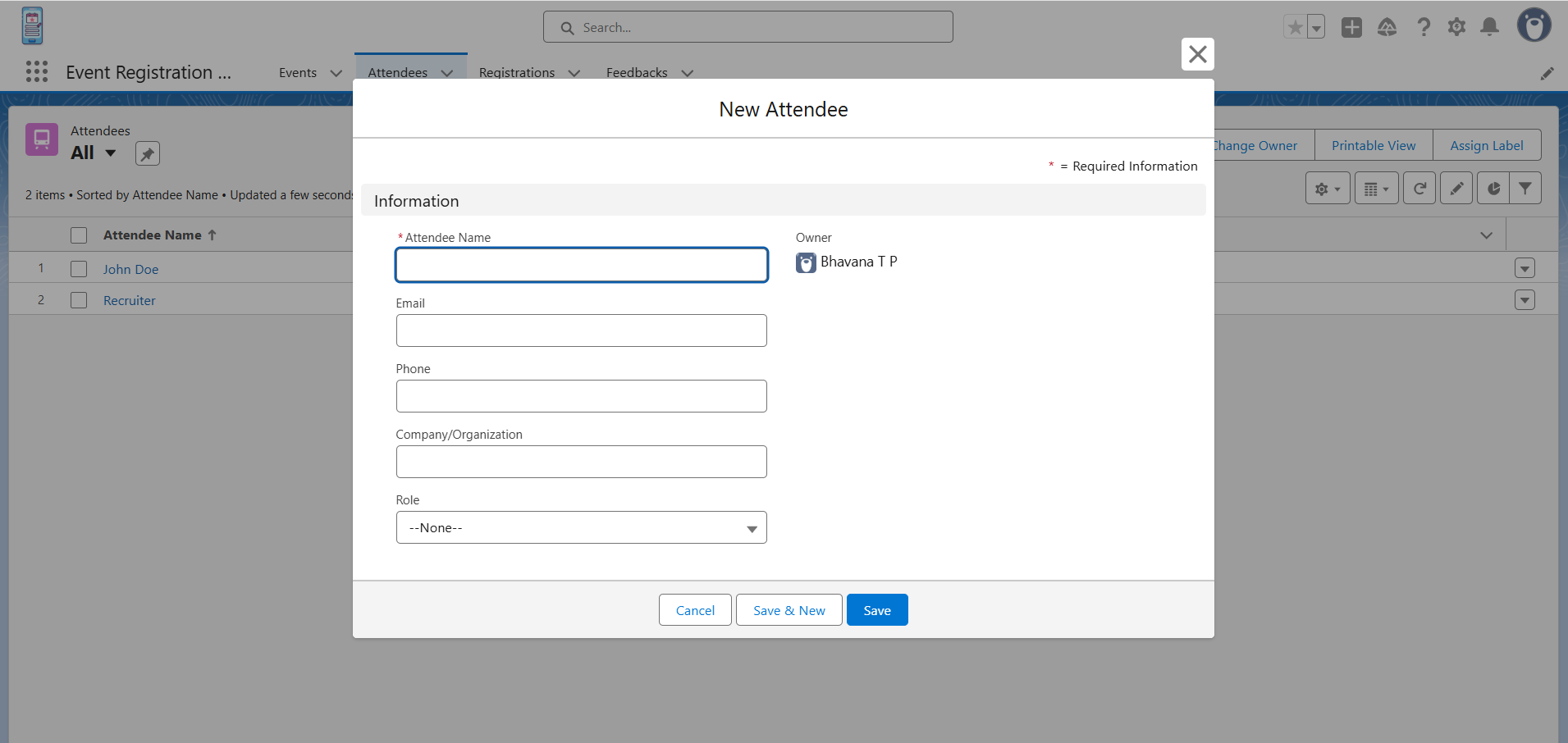
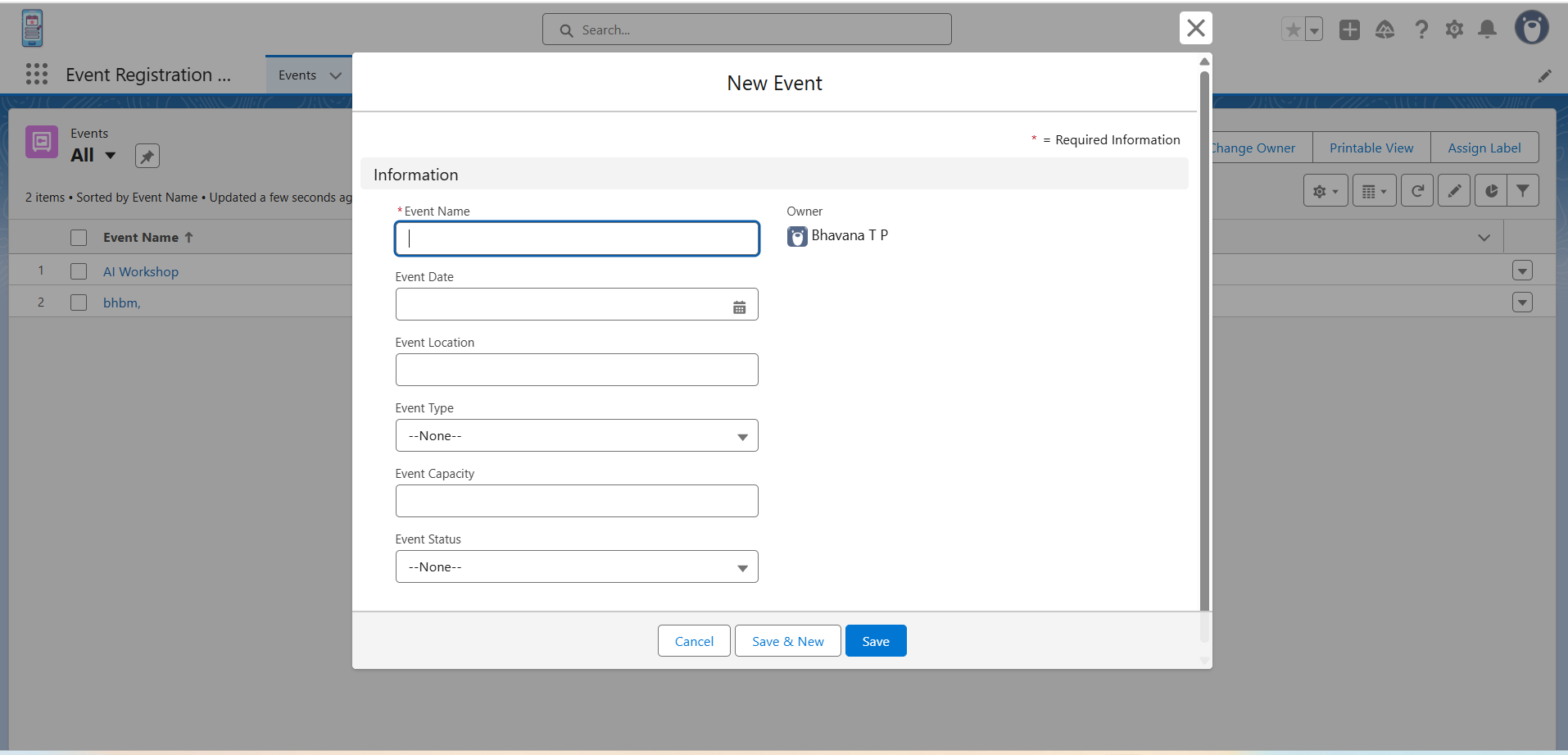
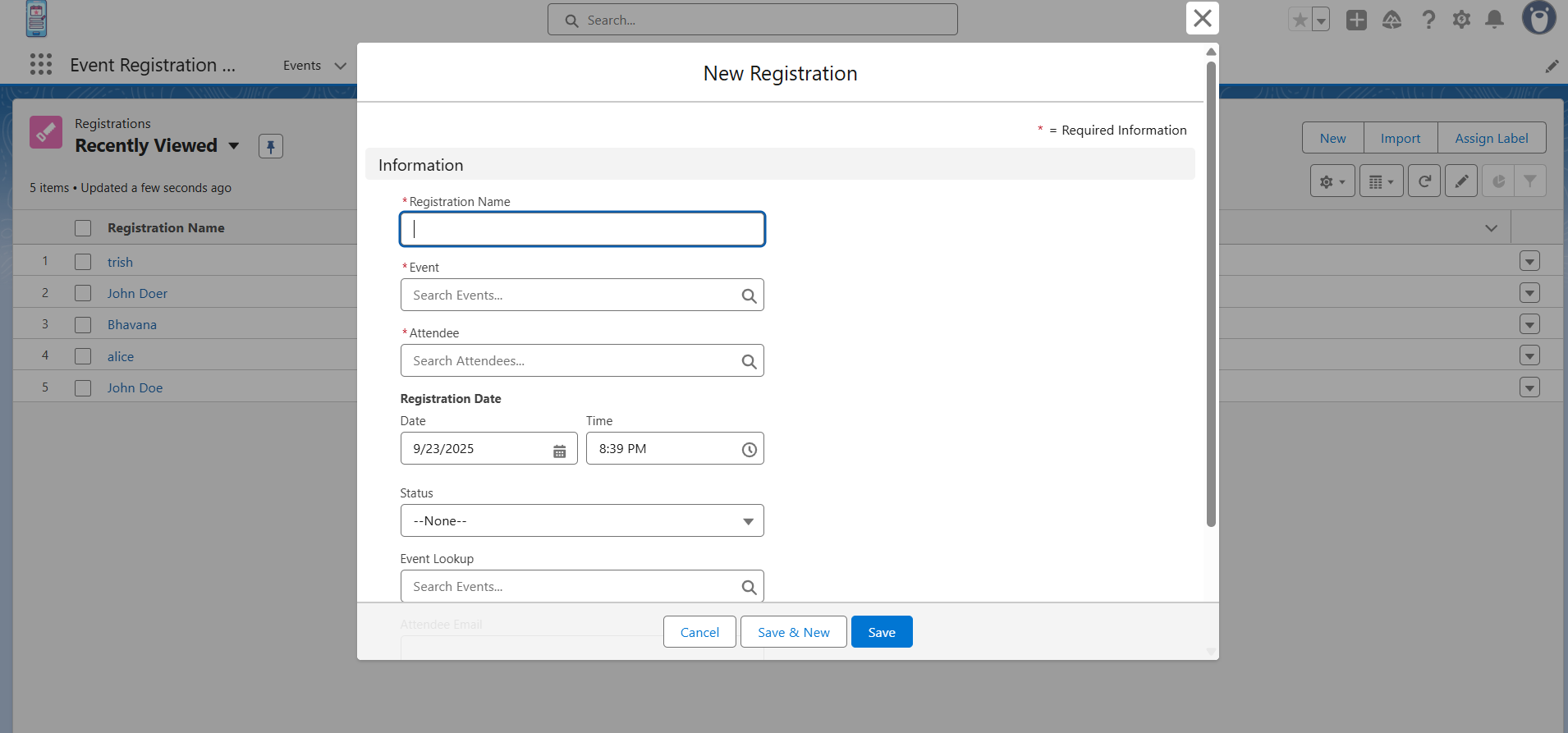
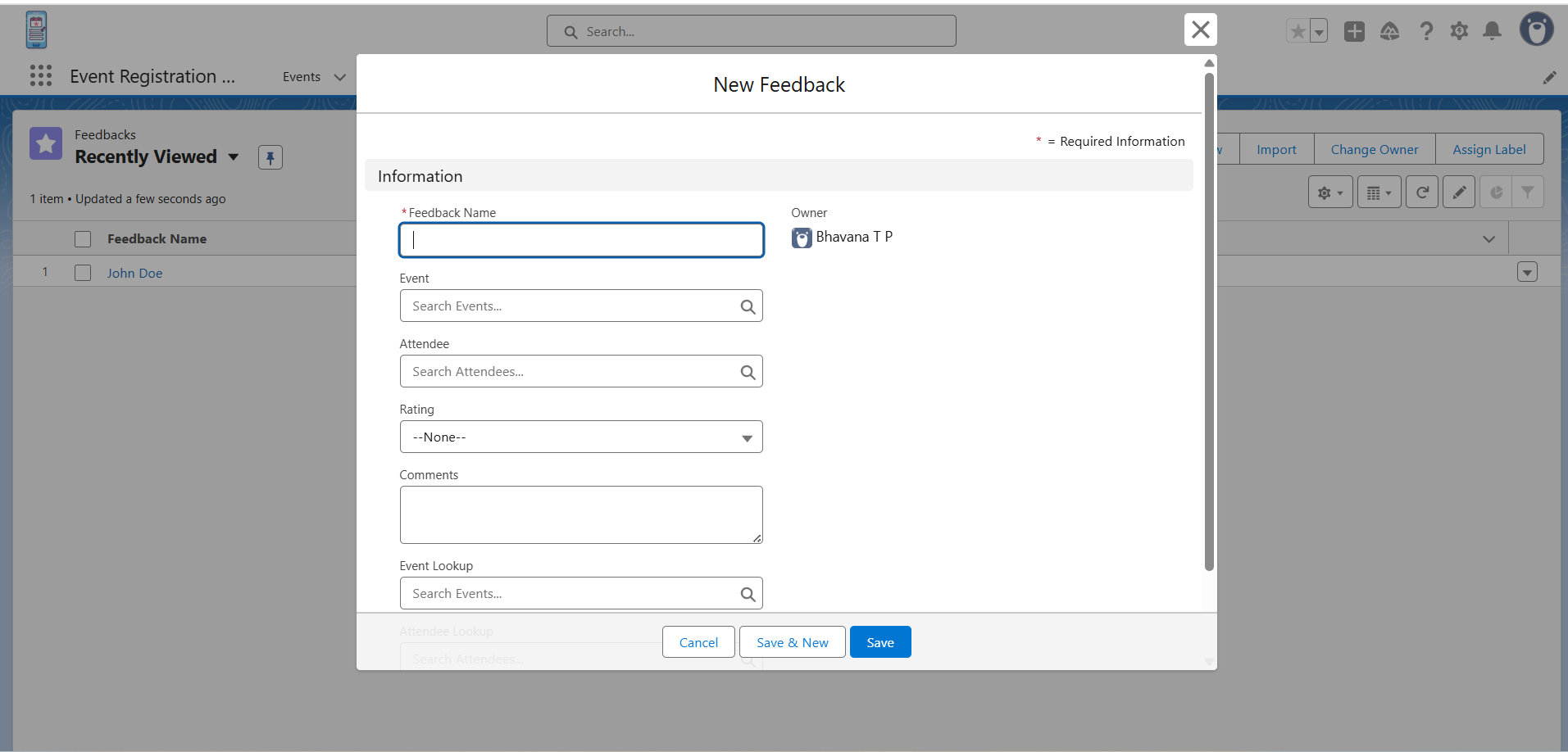
## Objective

Establish a Salesforce environment where the Event Registration App can be developed and deployed.

## Tasks Performed

* - Created Salesforce Developer Org and configured company profile.
* - Created user roles: Administrator, Organizer, Attendee.
* - Configured profiles and permissions for each role.

## Outcome

Salesforce environment was set up with role-based access ensuring correct permissions for admins, organizers, and attendees.

# Phase 3 – Data Modeling and Relationship

## Objective

Design a scalable and normalized data structure to support event registration, attendee management, and feedback collection.

## Tasks Performed

* - Created custom objects: Event\_\_c, Attendee\_\_c, Registration\_\_c, Feedback\_\_c.
* - Added key fields such as Event Name, Event Date, Event Capacity, Attendee Email, Registration Status, Feedback Rating.
* - Defined relationships: Event ↔ Registration, Attendee ↔ Registration, Event ↔ Feedback, Attendee ↔ Feedback.

## Outcome

A well-structured relational model was implemented in Salesforce ensuring data integrity

# Phase 4 – Process Automation (Admin)

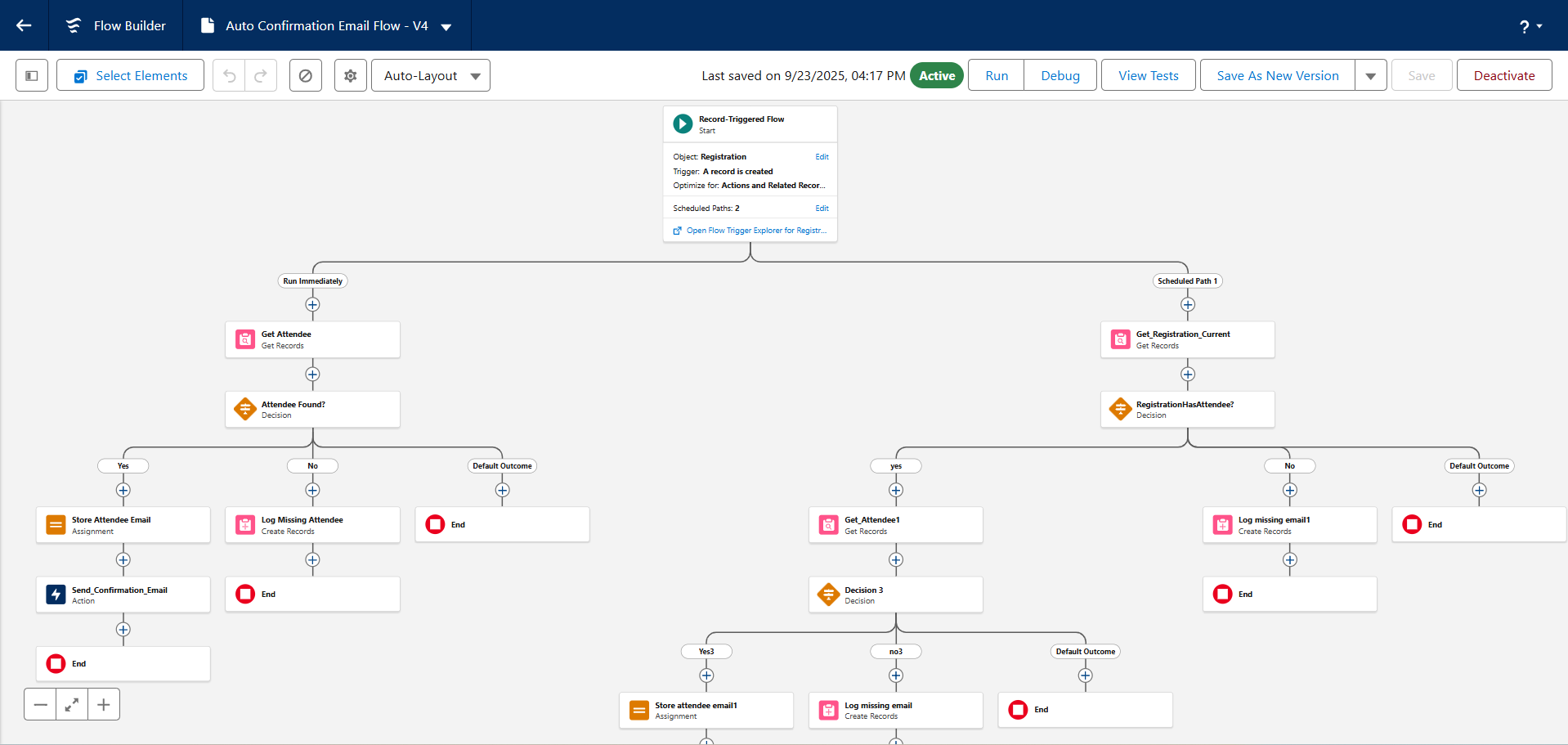
## Objective

Automate repetitive processes such as confirmations, reminders, and validations.

## Tasks Performed

* - Built Record-Triggered Flow to send confirmation email when a registration is created.
* - Built Scheduled Flow to send reminder emails 1 day before Event Date.
* - Created Validation Rule to prevent overbooking.
* - Created Flow to auto-update Registration status when cancelled.

## Outcome

Automation ensured timely communication, reduced manual work, and maintained data accuracy.

# Phase 5 – Apex Programming (Developer)

## Objective

Implement advanced business logic beyond declarative tools.

## Tasks Performed

* - Created RegistrationTrigger to auto-waitlist attendees if Event capacity is reached.
* - Created RegistrationAfterInsert trigger to call Apex class for confirmation emails.
* - Developed RegistrationHandler class to send programmatic emails.
* - Created EventAfterUpdate trigger to auto-create Feedback records once Event is marked completed.

## Outcome

Apex logic ensured robust, scalable automation and advanced event handling.

**Trigger: Prevent Overbooking & Auto-Waitlist**

trigger RegistrationTrigger on Registration\_\_c (before insert, before update) {

// Collect Event IDs from incoming registrations

Set<Id> eventIds = new Set<Id>();

for (Registration\_\_c reg : Trigger.new) {

if (reg.Event\_\_c != null) {

eventIds.add(reg.Event\_\_c);

}

}

// Query Events with their current registration count

Map<Id, Event\_\_c> eventMap = new Map<Id, Event\_\_c>(

[SELECT Id, Name, Capacity\_\_c,

(SELECT Id FROM Registrations\_\_r)

FROM Event\_\_c

WHERE Id IN :eventIds]

);

// Apply logic

for (Registration\_\_c reg : Trigger.new) {

if (reg.Event\_\_c != null && eventMap.containsKey(reg.Event\_\_c)) {

Event\_\_c ev = eventMap.get(reg.Event\_\_c);

Integer currentCount = ev.Registrations\_\_r.size();

if (currentCount >= ev.Capacity\_\_c) {

// Event is full → auto waitlist

reg.Status\_\_c = 'Waitlisted';

} else {

// Seats available → mark registered

reg.Status\_\_c = 'Registered';

}

}

}

}

**Trigger: Call Email Logic After Insert**

trigger RegistrationAfterInsert on Registration\_\_c (after insert) {

for (Registration\_\_c reg : Trigger.new) {

if (reg.Status\_\_c == 'Registered') {

RegistrationHandler.sendConfirmationEmail(reg);

}

}

}

**Apex Class: Send Confirmation Email**

public class RegistrationHandler {

public static void sendConfirmationEmail(Registration\_\_c reg) {

if (reg.Attendee\_\_c == null || reg.Event\_\_c == null) return;

// Fetch Attendee and Event details

Attendee\_\_c att = [SELECT Id, Name, Email\_\_c FROM Attendee\_\_c WHERE Id = :reg.Attendee\_\_c LIMIT 1];

Event\_\_c ev = [SELECT Id, Name, Event\_Date\_\_c, Event\_Location\_\_c FROM Event\_\_c WHERE Id = :reg.Event\_\_c LIMIT 1];

// Construct email

Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();

email.setToAddresses(new String[] {att.Email\_\_c});

email.setSubject('Registration Confirmed: ' + ev.Name);

email.setPlainTextBody(

'Hi ' + att.Name + ',\n\n' +

'Your registration for "' + ev.Name + '" is confirmed.\n\n' +

'Date: ' + String.valueOf(ev.Event\_Date\_\_c) + '\n' +

'Location: ' + ev.Event\_Location\_\_c + '\n\n' +

'See you there!\nEvent Team'

);

Messaging.sendEmail(new Messaging.SingleEmailMessage[] {email});

}

}

**Trigger: Call Email Logic After Insert**

trigger RegistrationAfterInsert on Registration\_\_c (after insert) {

for (Registration\_\_c reg : Trigger.new) {

if (reg.Status\_\_c == 'Registered') {

RegistrationHandler.sendConfirmationEmail(reg);

}

}

}

# Phase 6 – User Interface Development

## Objective

Provide a user-friendly interface for managing Events, Attendees, Registrations, and Feedback.

## Tasks Performed

* - Built Lightning App 'Event Registration App' with tabs for all objects.
* - Customized Page Layouts with relevant fields and related lists.
* - Configured Lightning Record Pages with dynamic components.

## Outcome

Delivered an intuitive UI for admins and organizers to manage the event lifecycle.

# Phase 7 – Integration and External Access

## Objective

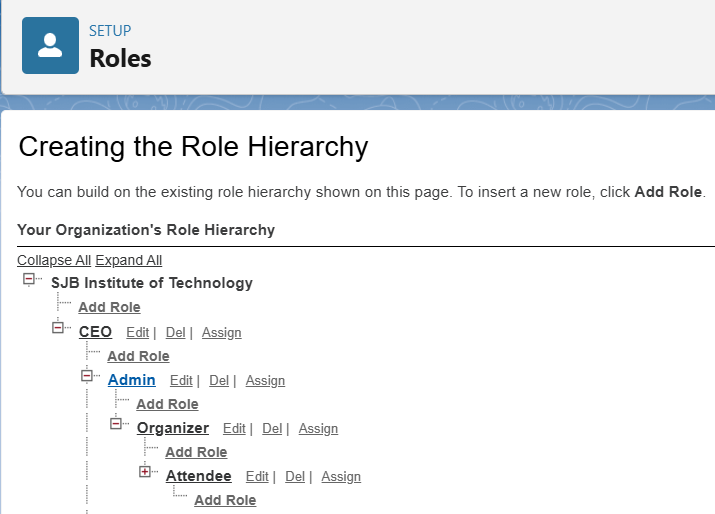
Allow external participants to interact with the system and provide communication tools.

## Tasks Performed

* - Configured Salesforce Experience Cloud site for external attendees.
* - Enabled Salesforce Email integration for organizer communication.

## Outcome

External users could self-register and organizers could communicate seamlessly.



# Phase 8 – Data Management and Deployment

## Objective

Ensure clean data handling and successful deployment.

## Tasks Performed

* - Imported sample Events and Attendees via Data Import Wizard.
* - Validated mandatory fields and consistent relationships.
* - Deployed app configurations using Change Sets.
* - Created data backup and cleanup strategy.

## Outcome

Successful deployment with clean data and reliable processes.