**Event Management for Institution Services**

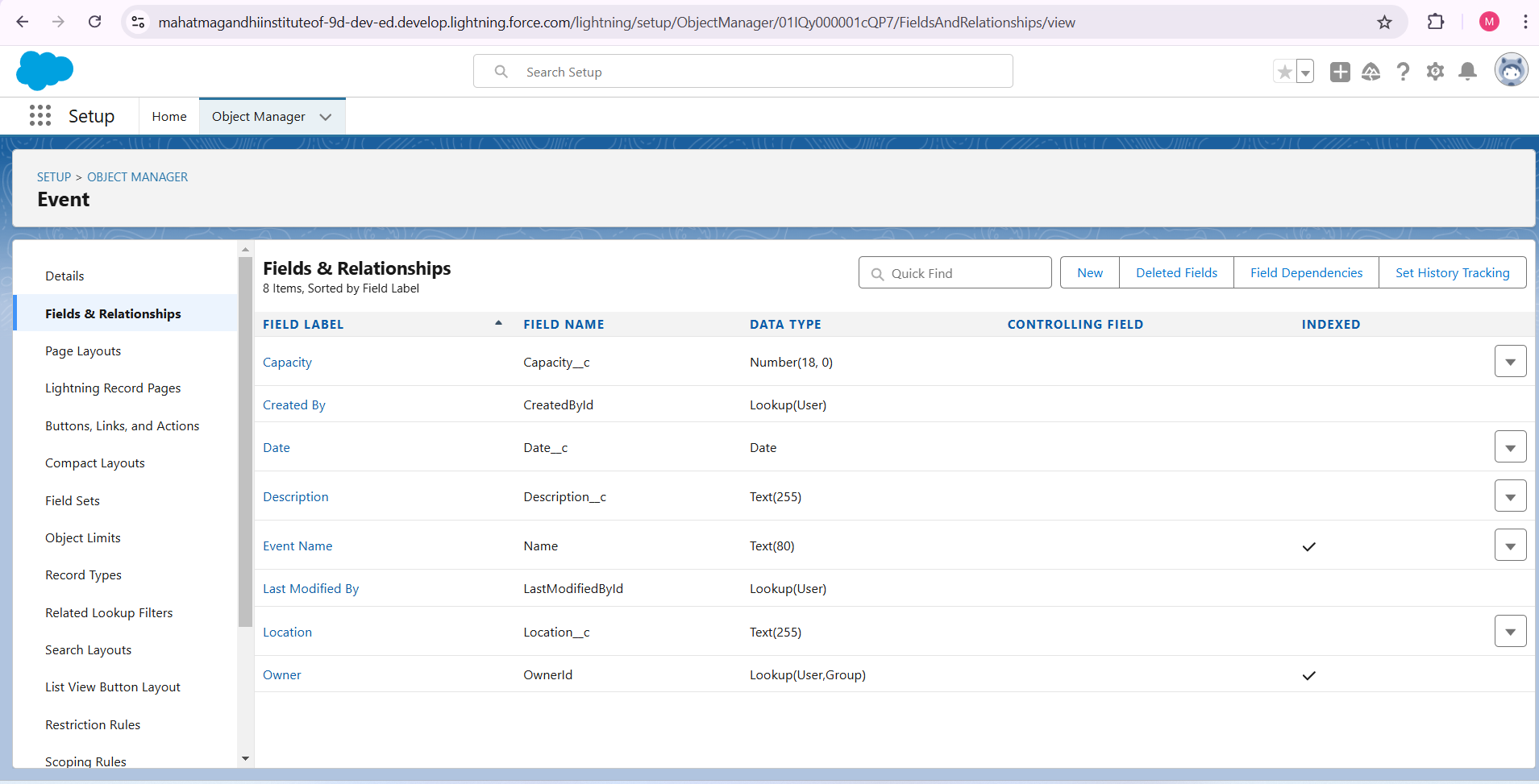
An Event Management System for institutional services is designed to simplify organizing, managing, and tracking events on campus. It can handle various event types, such as workshops, seminars, orientation sessions, counseling sessions, and cultural events, making it a versatile tool for campus administration. Its goal is to improve engagement, communication, and operational efficiency for both students and staff involved in campus events.

### Step 1: ****Setting Up the Custom Objects****

First, you need to create the necessary custom objects to store event and registration data.

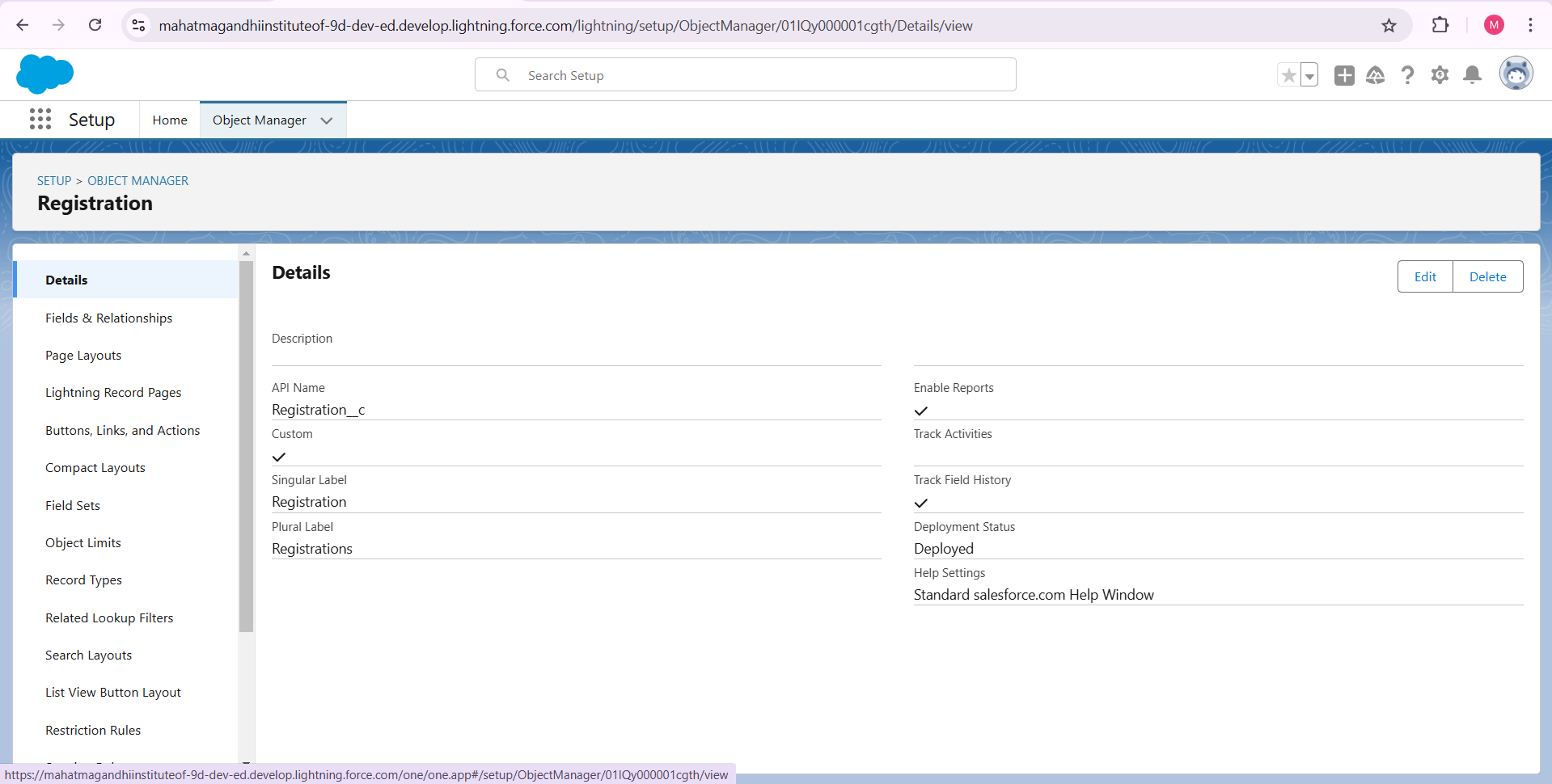
**1.Create an Event Object**

* 1. Go to **Setup** > **Object Manager**.
  2. Click **Create** > **Custom Object**.
  3. Fill in the details:
     + **Label**: Event
     + **Plural Label**: Events
     + **Object Name**: Event
     + **Data Type**: Text (for event name, description)
     + Add custom fields:
       - **Date** (Date/Time type)
       - **Location** (Text)
       - **Capacity** (Number)
     + Enable **Allow Reports** and **Track Field History**.
  4. Click **Save**.



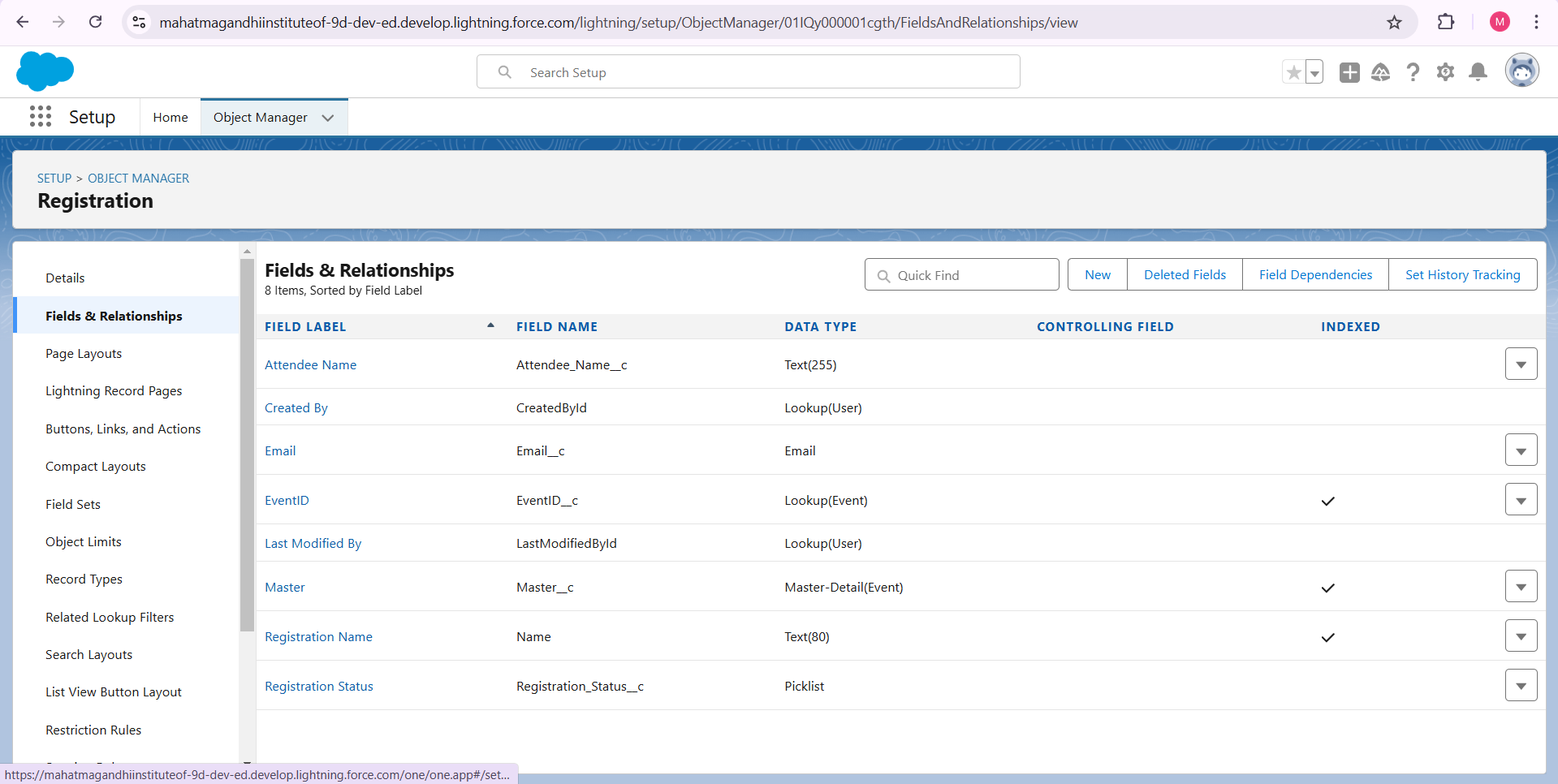
**2.Create a Registration Object**

* 1. Follow the same steps as above but for registration.
  2. Label the object **Registration**.
  3. Add fields like **Event ID** (Lookup to Event object), **Attendee Name** (Text), **Email** (Email), and **Registration Status** (Picklist with values like "Pending," "Confirmed").
  4. Save the object.



### Step 2: ****Creating Relationships Between Objects****

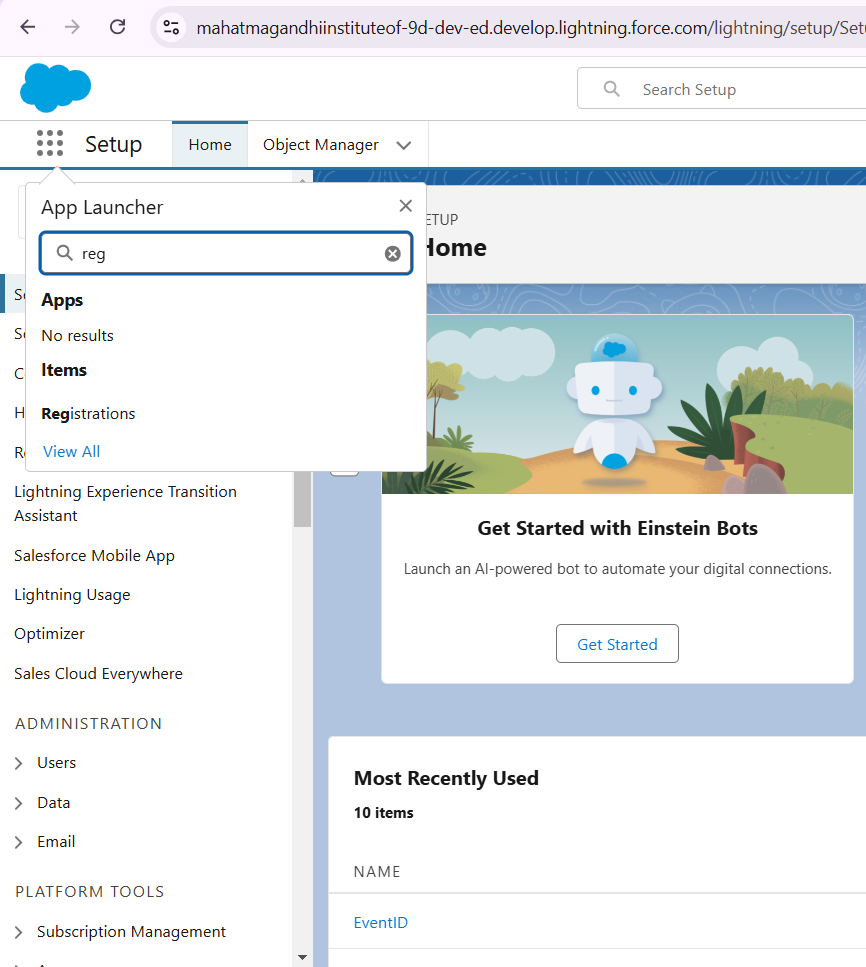
* Add a **Master-Detail Relationship** from Registration to Event. This will allow you to track registrations for specific events.

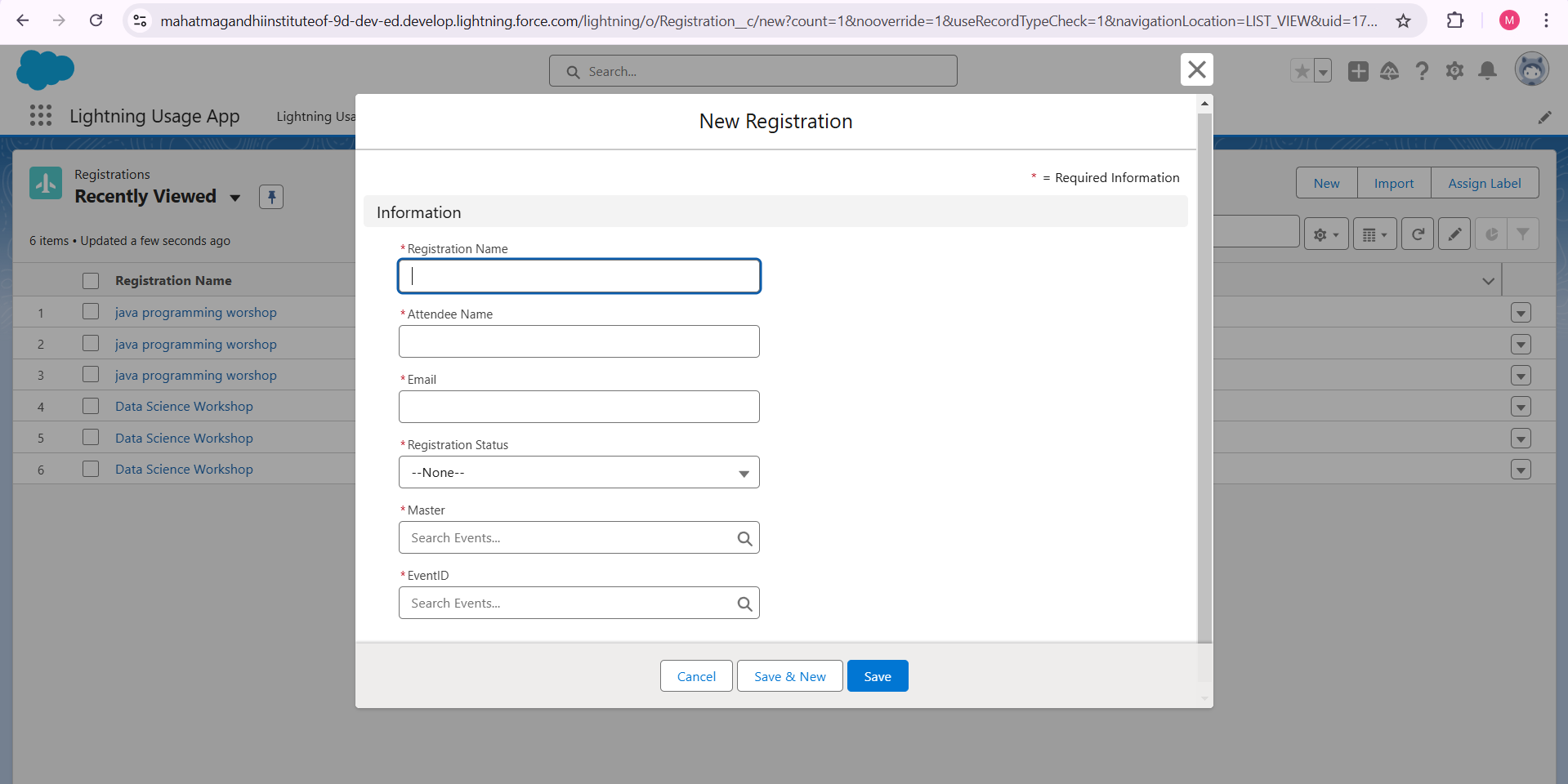


### Step 3: ****Creating a Registration Form****

We will use Salesforce’s UI for users to register for events.

* Go to **App Builder** and create a **Lightning Page** for the event registration form.
* Drag and drop components like fields for **Attendee Name**, **Email**, and **Event**.
* Save and activate the page for the Registration object.





### Step 4: ****Developing Apex Trigger for Capacity Check****

Create an Apex trigger to ensure that an event’s capacity is not exceeded when attendees register.

1. Go to **Developer Console** > **File** > **New** > **Apex Trigger**.

Create a trigger on the **Registration** object to check event capacity:

**Apex code**

trigger CheckCapacity on Registration (before insert) {

List<Event\_\_c> eventList = [SELECT Id, Capacity\_\_c, (SELECT Id FROM Registrations\_\_r)

FROM Event\_\_c WHERE Id IN :Trigger.newMap.keySet()];

for (Event\_\_c event : eventList) {

Integer registeredCount = event.Registrations\_\_r.size();

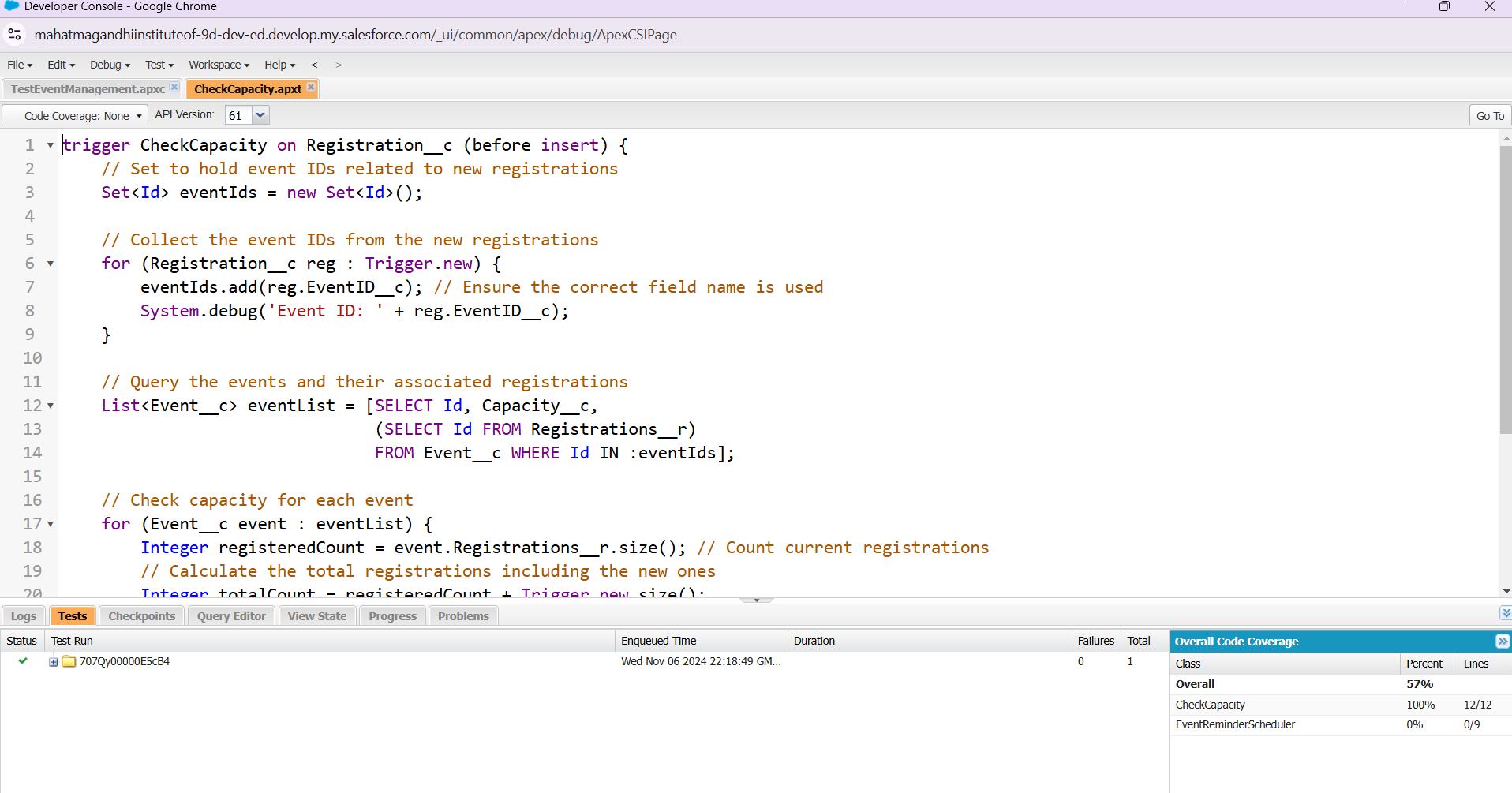
if (registeredCount >= event.Capacity\_\_c) {

Trigger.new[0].addError('Event capacity has been reached. You cannot register.');

}

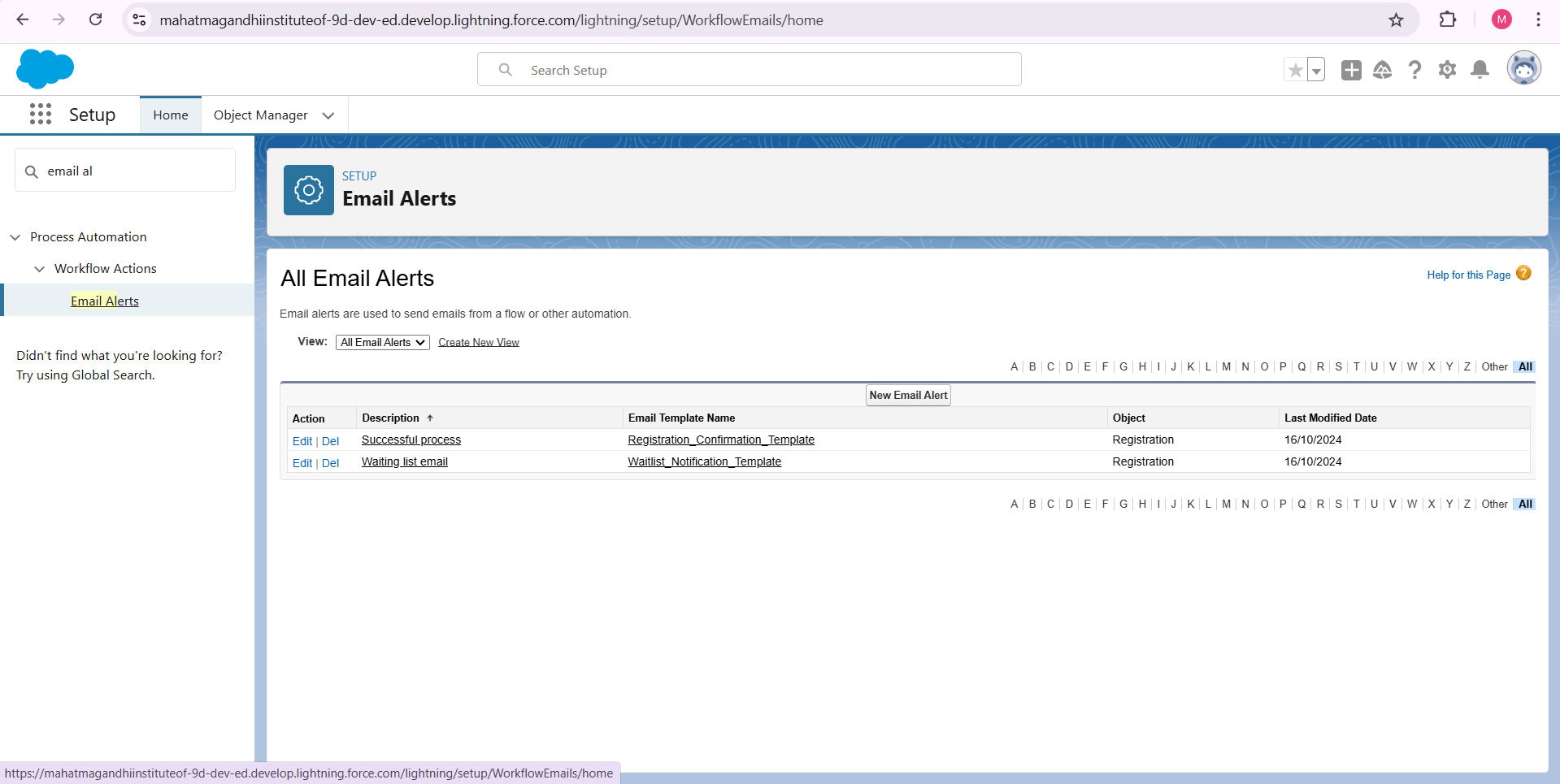
}

}



### Step 5: ****Automating Notifications via Email****

* Use **Email Alerts** to notify attendees of successful registration or if they are on a waitlist.
* Create a **Flow** or **Process Builder**:
  1. Trigger the flow when a **Registration** record is created.
  2. Set up **Email Alerts** with templates for registration confirmation.



### Step 6: ****Adding Scheduled Apex for Event Reminders****

* Create a scheduled class to send reminders to attendees before the event starts.

Go to **Developer Console** > **New** > **Apex Class** and write a scheduled class:

**Apex code**

global class EventReminderScheduler implements Schedulable {

global void execute(SchedulableContext sc) {

List<Registration\_\_c> registrations = [SELECT Attendee\_Email\_\_c, Event\_\_r.Date\_\_c

FROM Registration\_\_c WHERE Event\_\_r.Date\_\_c = :Date.today().addDays(1)];

for (Registration\_\_c reg : registrations) {

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

mail.setToAddresses(new String[]{reg.Attendee\_Email\_\_c});

mail.setSubject('Reminder: Upcoming Event');

mail.setPlainTextBody('Your event "' + reg.Event\_\_r.Name + '" is tomorrow.');

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

}

}

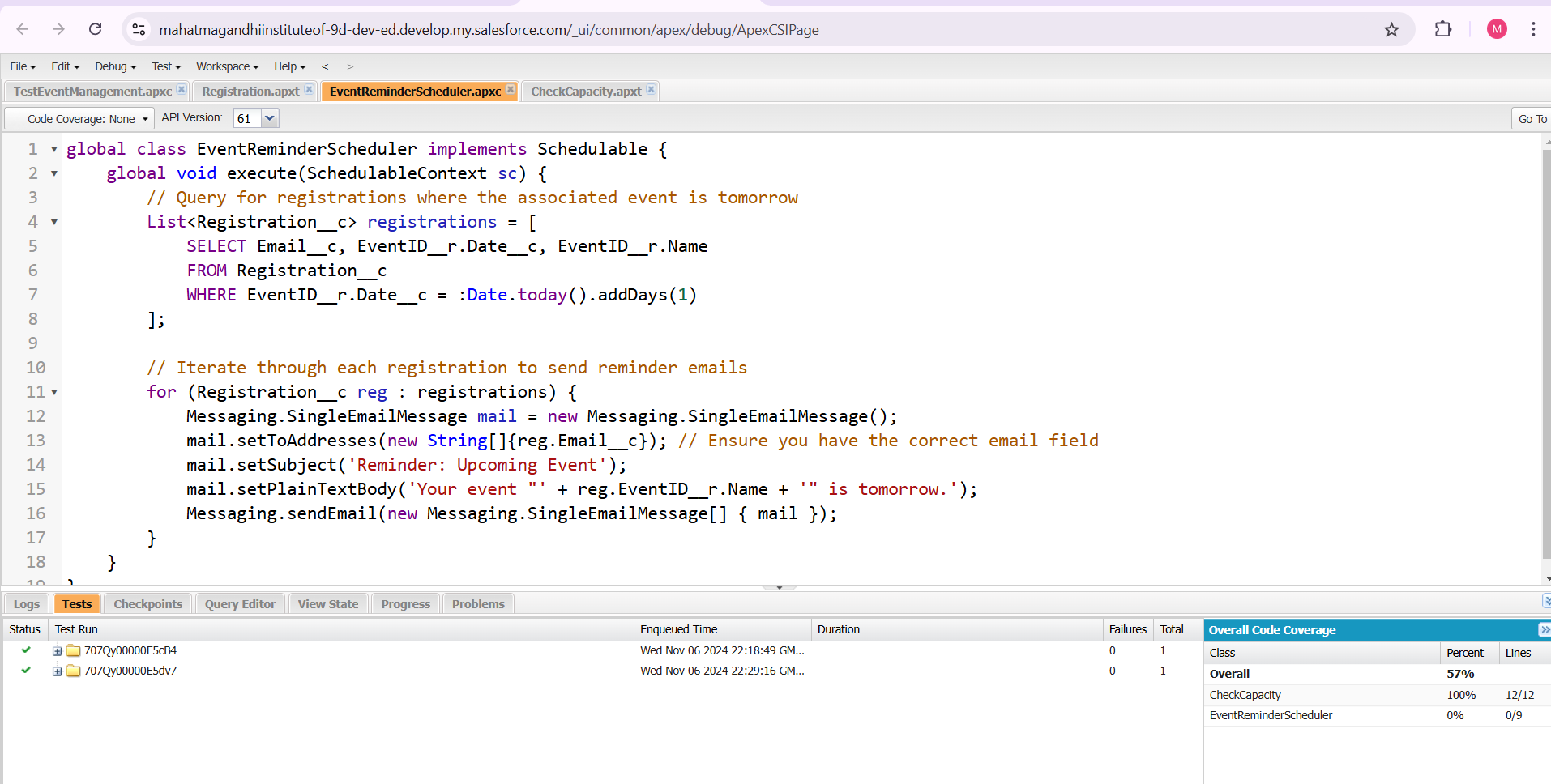
}

Schedule this class to run daily using the following:

**Apex code**

String sch = '0 0 8 \* \* ?'; // Every day at 8 AM

System.schedule('Event Reminder Job', sch, new EventReminderScheduler());



### Step 7: ****Testing the Apex Classes and Triggers****

Testing is essential to ensure full code coverage in Salesforce.

1. Go to **Developer Console** > **File** > **New** > **Apex Test Class**.
2. Write unit tests for the trigger and scheduled classes.

**Apex code**

@isTest

private class TestEventManagement {

static testMethod void testCapacityCheck() {

Event\_\_c event = new Event\_\_c(Name = 'Workshop', Capacity\_\_c = 2);

insert event;

Registration\_\_c reg1 = new Registration\_\_c(Event\_\_c = event.Id, Attendee\_Name\_\_c = 'John', Email\_\_c = 'john@example.com');

Registration\_\_c reg2 = new Registration\_\_c(Event\_\_c = event.Id, Attendee\_Name\_\_c = 'Jane', Email\_\_c = 'jane@example.com');

insert new List<Registration\_\_c>{reg1, reg2};

Registration\_\_c reg3 = new Registration\_\_c(Event\_\_c = event.Id, Attendee\_Name\_\_c = 'Doe', Email\_\_c = 'doe@example.com');

Test.startTest();

try {

insert reg3; // Should fail

} catch (DmlException e) {

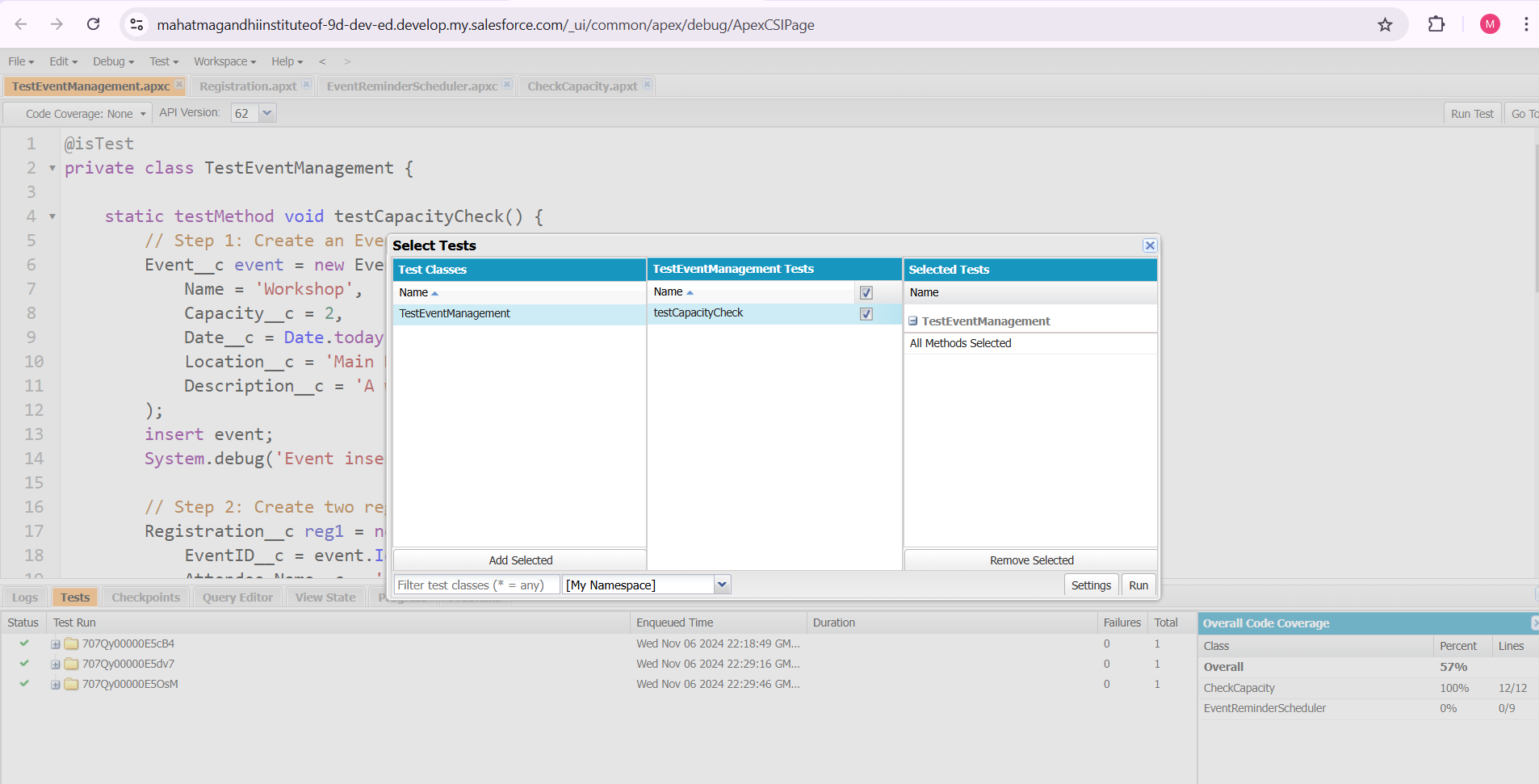
System.assert(e.getMessage().contains('Event capacity has been reached'));

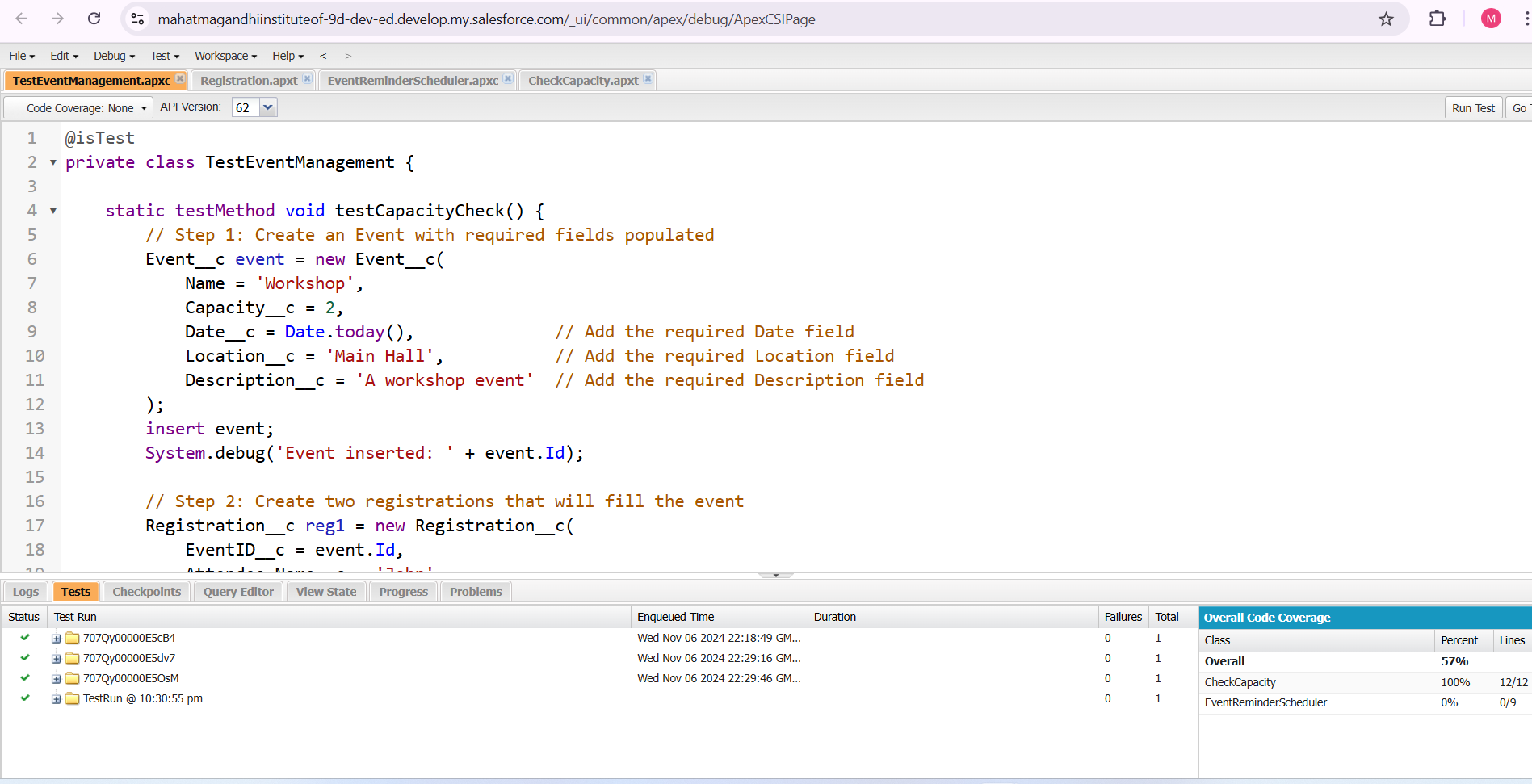
}

Test.stopTest();

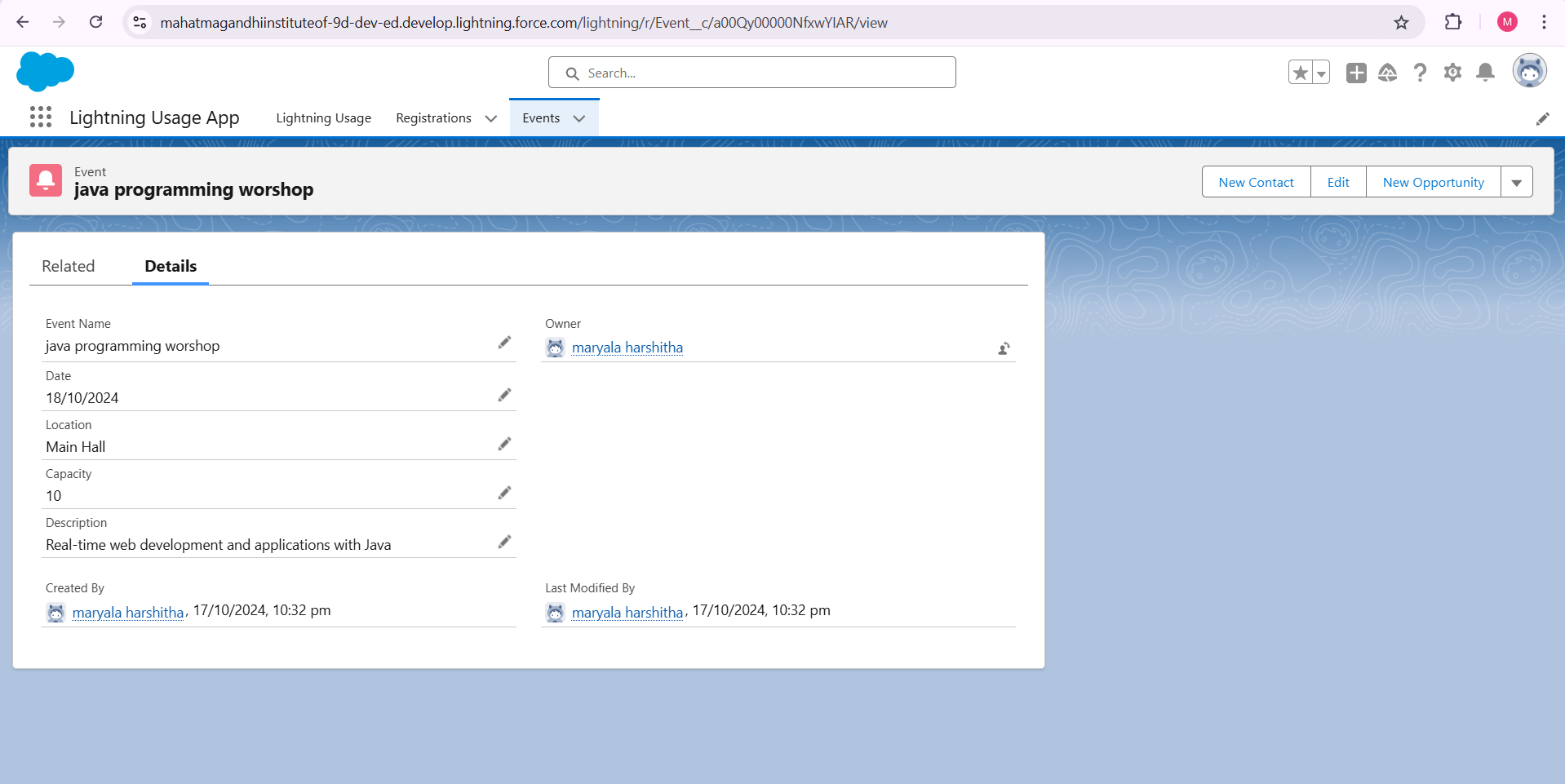
}

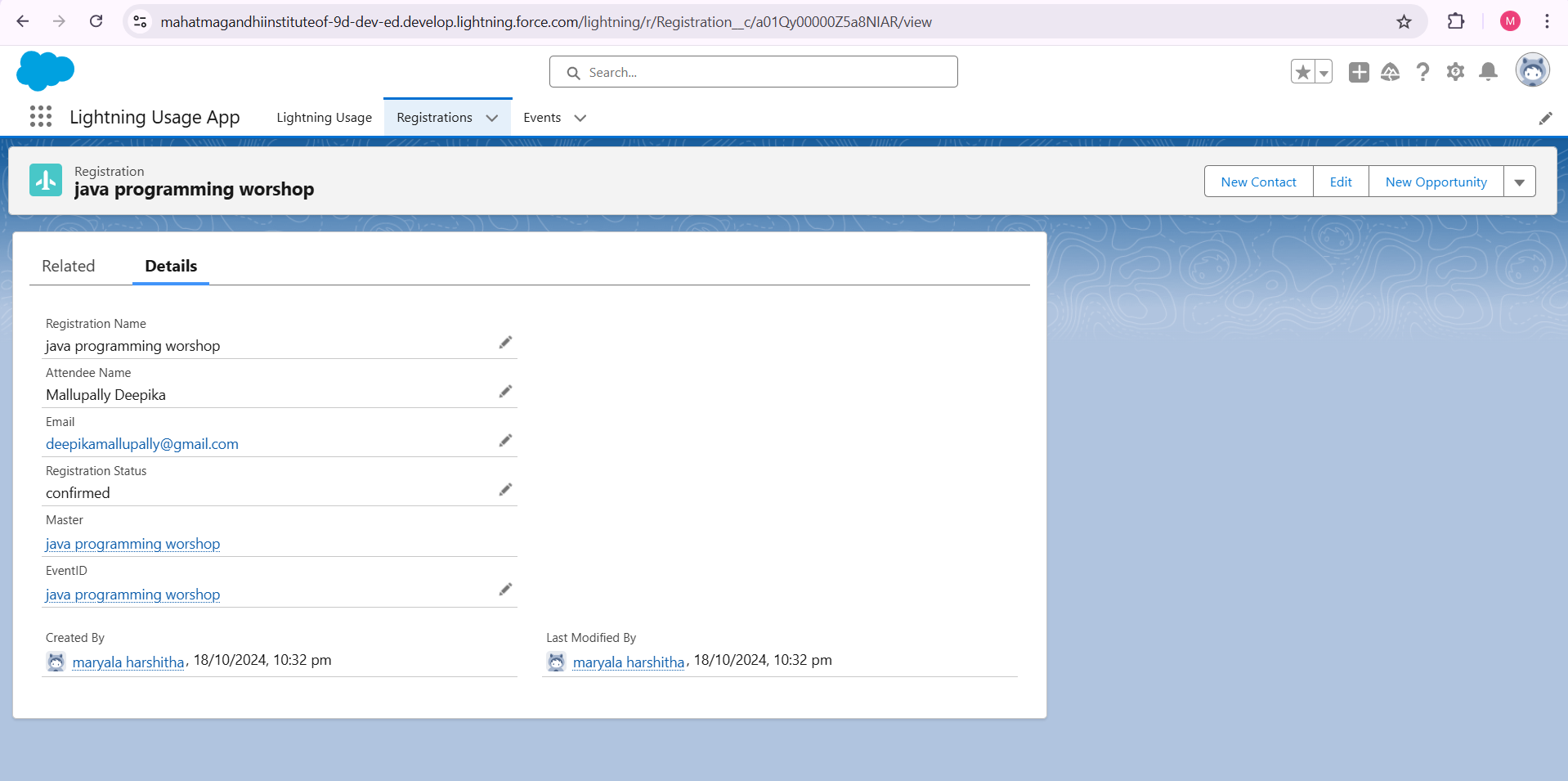
}

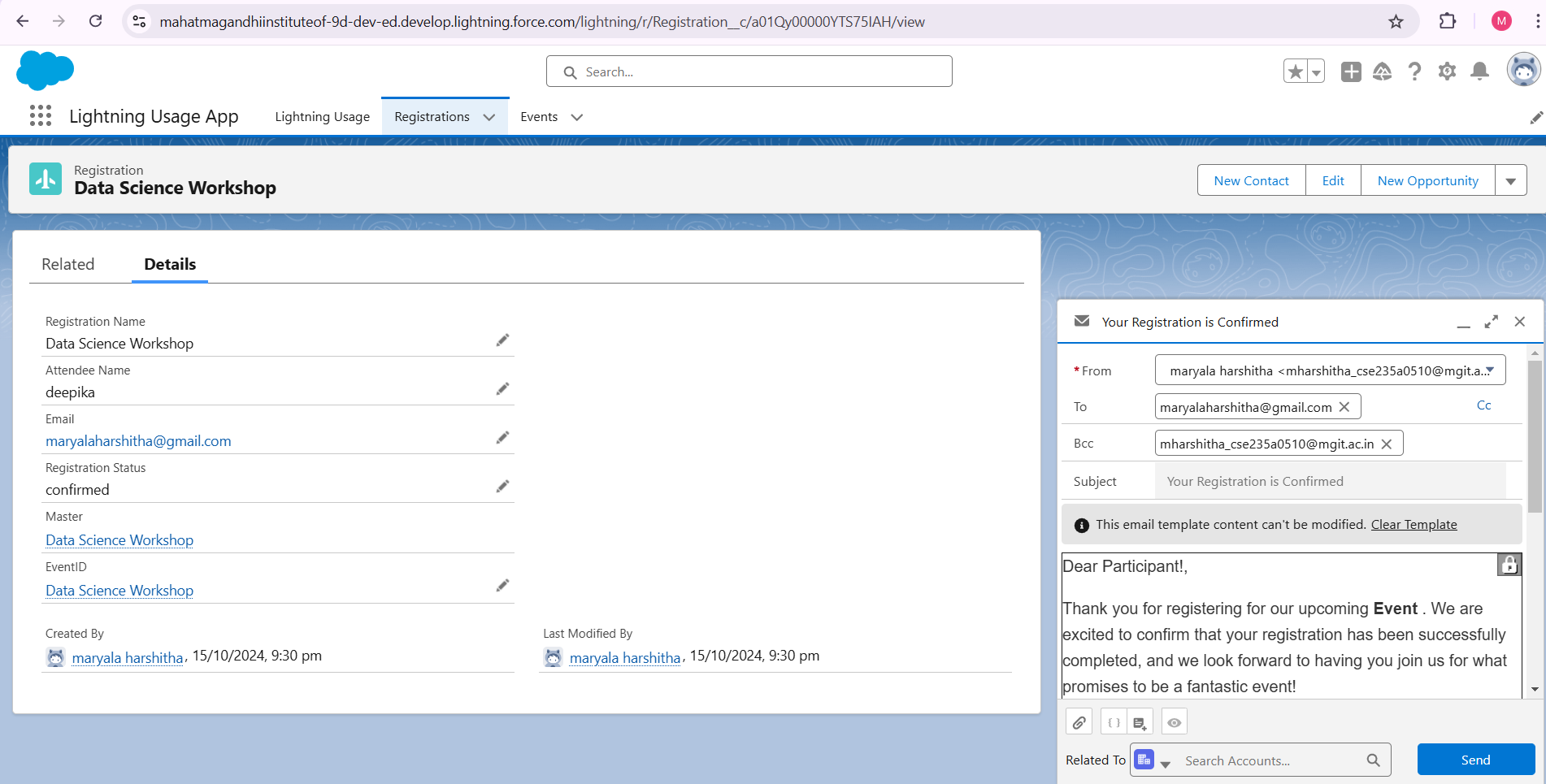


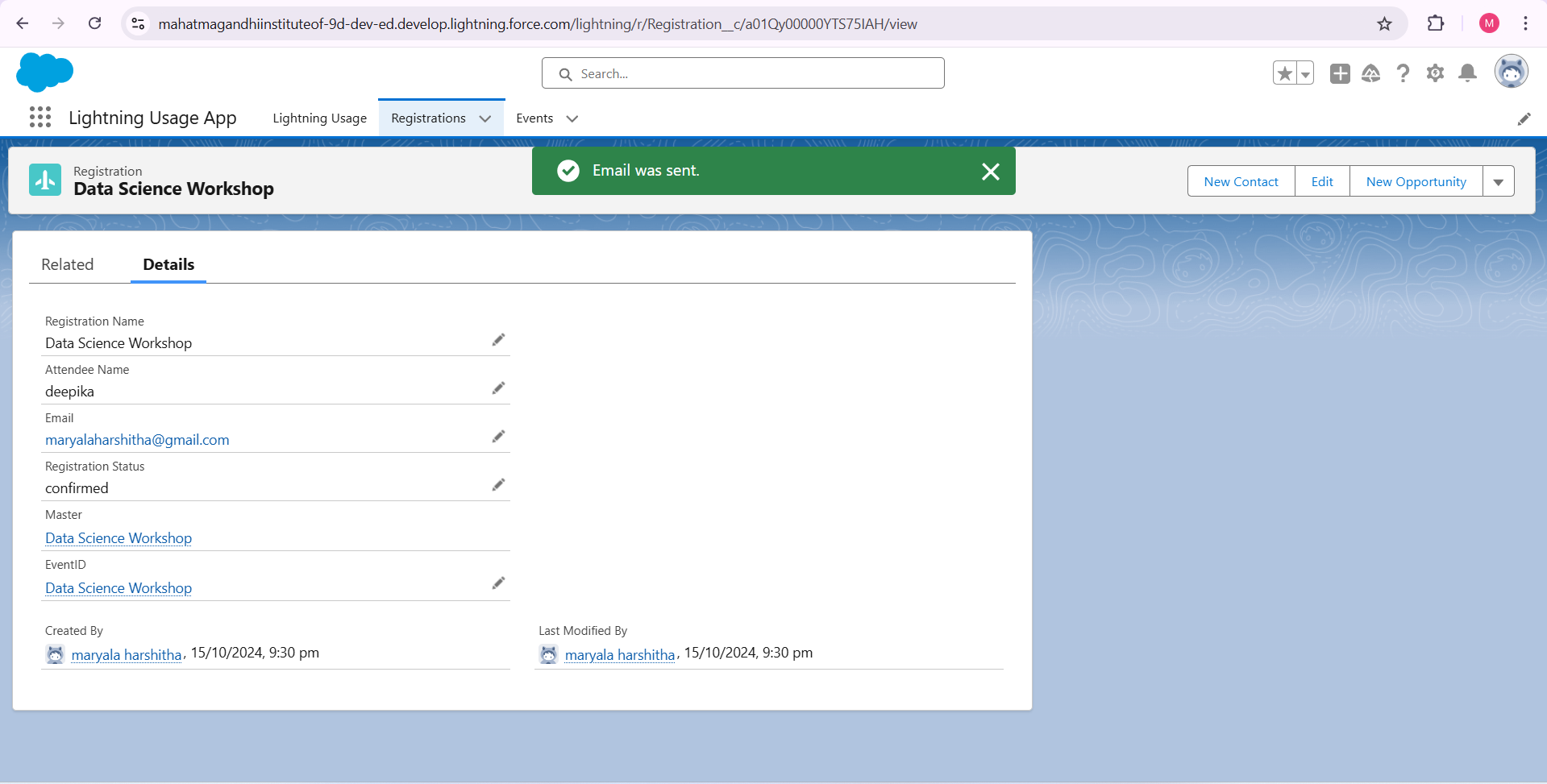


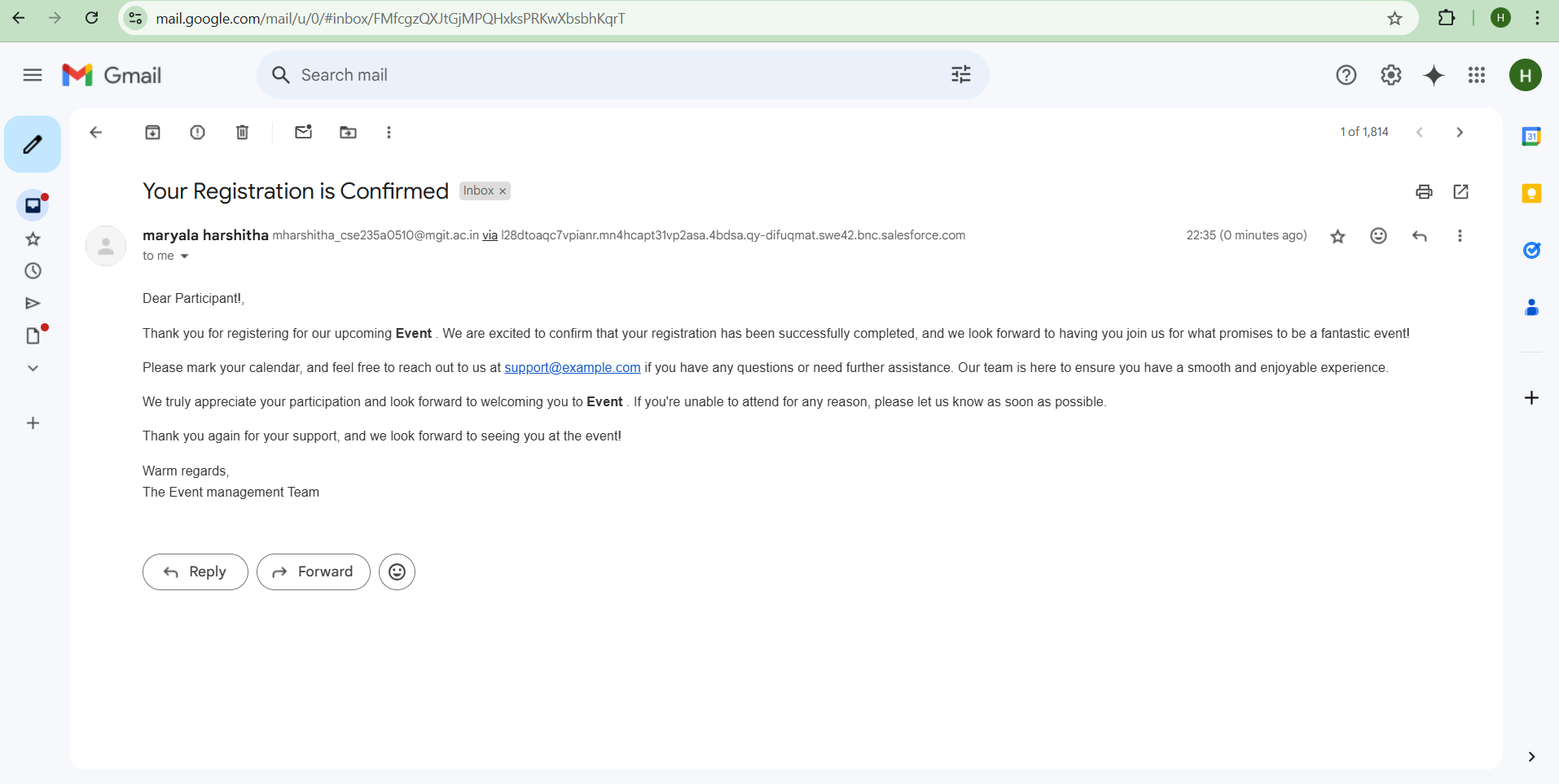
**OUTPUT:-.**





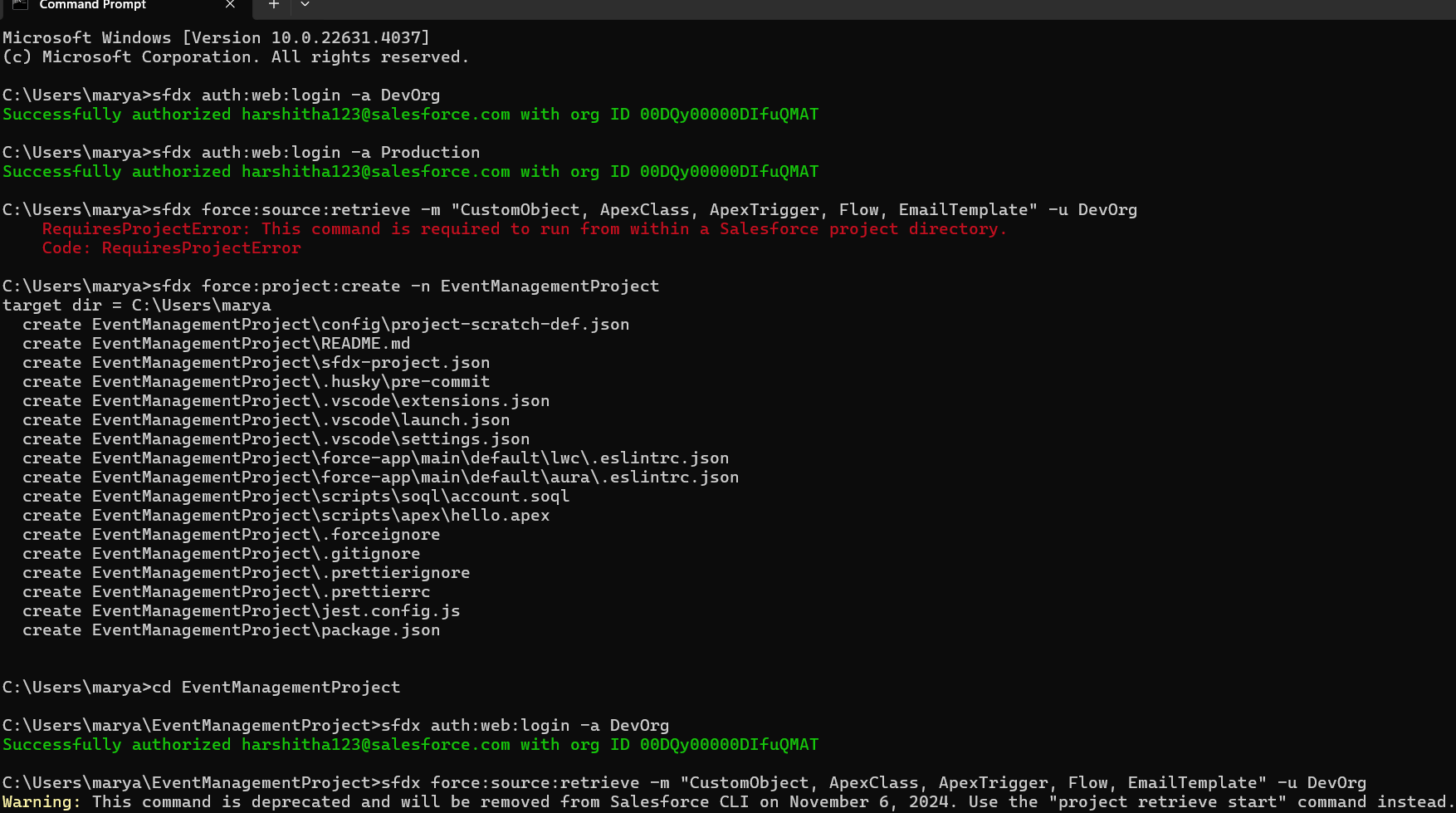


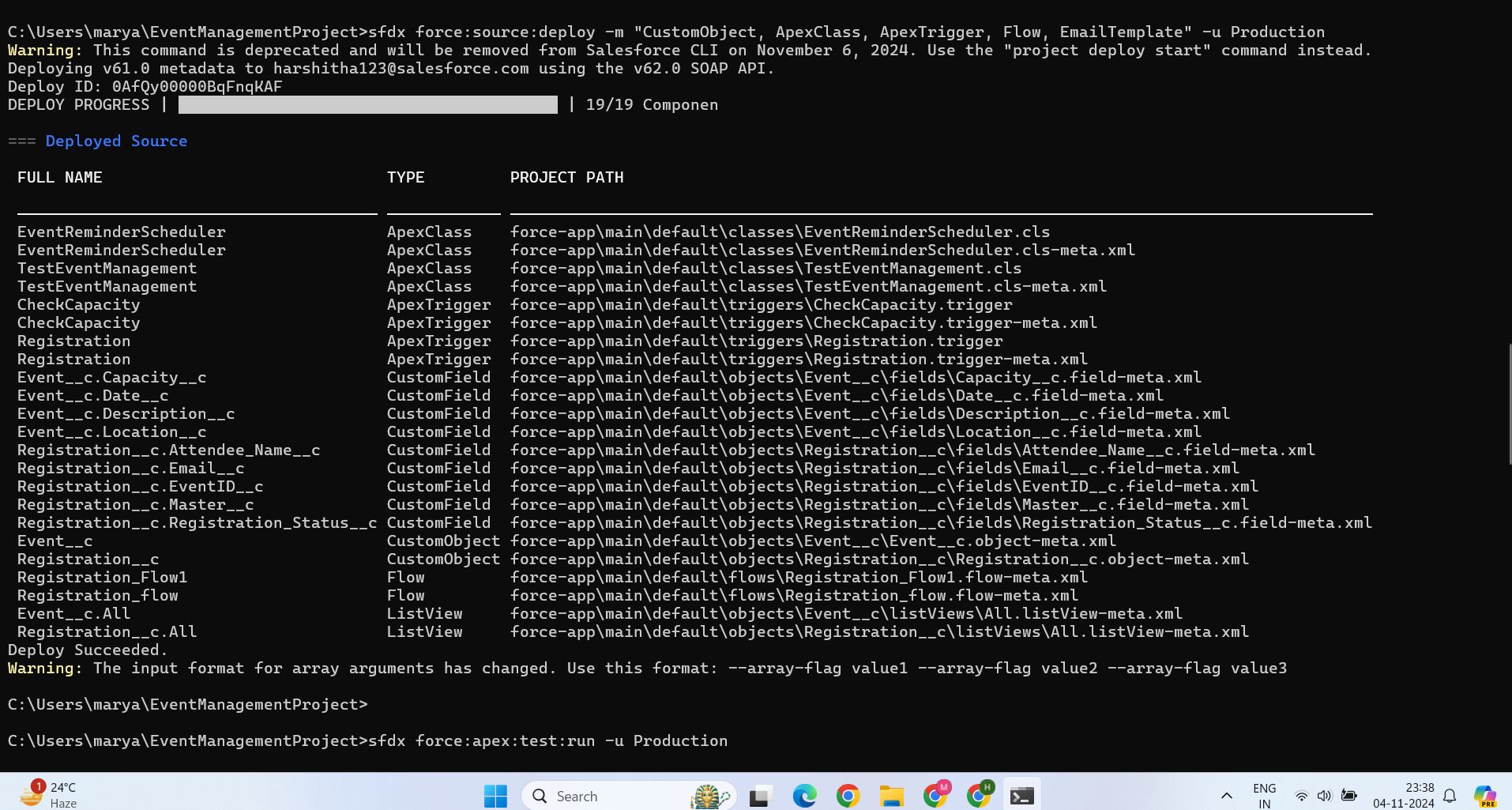


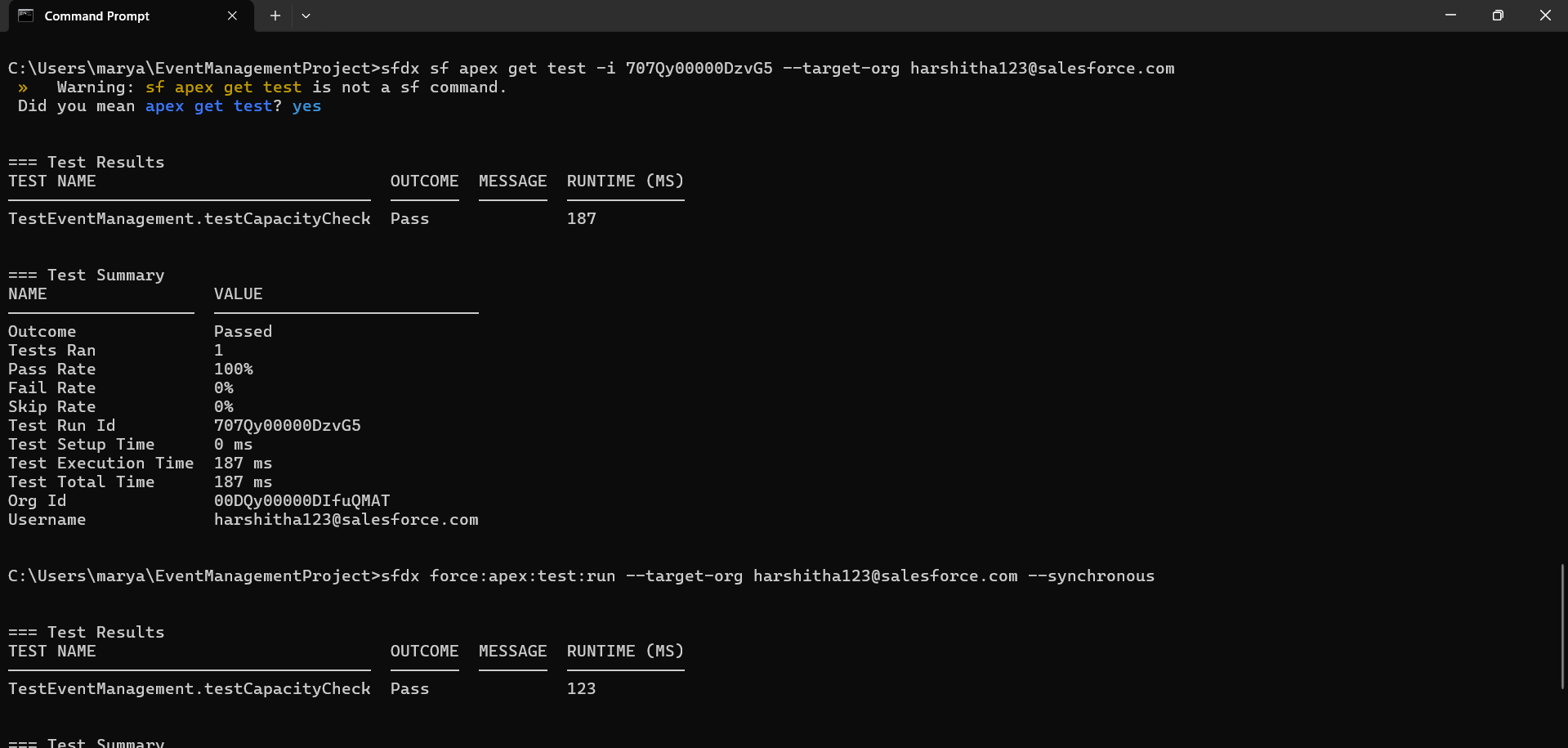


### Step 8: ****Deploying the Project****

* Once everything is working in your sandbox, deploy the project to production using **Change Sets** or **Salesforce DX**.
* Ensure all components, including objects, triggers, classes, and flows, are included in the deployment package.







### Step 9: ****Enhancements****

* Add more features like:
  + Attendee waitlist management.
  + Event feedback forms.
  + Custom reports and dashboards for event analytics.

### Advantages

* **Streamlined Event Organization**: The system centralizes event management, making it easy for administrators to organize workshops, counseling sessions, seminars, and other campus events. Automation reduces manual work, ensuring a smooth setup and reducing human errors.
* **Improved Communication and Engagement**: Integrated notifications, emails, and reminders keep participants informed about event details and updates. This engagement improves attendance rates and overall student and staff satisfaction.
* **Customizable Registration Process**: With custom objects and flows, the registration process can be tailored to specific events, making it more intuitive for different types of activities (workshops, counseling sessions, etc.).
* **Data Security and Privacy**: Built-in Salesforce security measures protect student and staff data, ensuring privacy compliance and secure handling of information.

