Tours and Travels CRM

Salesforce Documentation

Project Overview

Tours and Travels CRM is a system that enables travel agencies to manage their entire business process in one platform. It includes customer management, bookings, payments, travel packages, employee coordination, and customer feedback. The CRM reduces reliance on manual work and provides automation for activities such as booking confirmations, payment reminders, and feedback collection. It also offers dashboards and reports to monitor revenue, customer satisfaction, and overall business performance. By integrating these features, the system ensures smoother operations, improved communication, and enhanced customer service.

Objectives

The overall objective of the CRM is to automate travel operations and improve efficiency for agencies. It seeks to:

- Provide better customer management by centralizing all customer information and bookings.
- Streamline and automate the booking and payment process to save time and minimize errors.
- Enhance customer satisfaction with instant confirmations, timely reminders, and feedback tracking.

- Enable employees (agents, guides, and finance teams) to collaborate more effectively with role-based access.
- Deliver actionable business insights with reports and dashboards, allowing managers to track revenue, performance, and customer retention.

Overall, the vision is to develop a scalable, Users-friendly, and secure CRM to support both daily operations and long-term growth for travel agencies.

Phase 1: Requirement Analysis & Planning

Understanding Business Requirements

The initial task was to comprehend how travel agencies operate internationally. This involved learning about their day-to-day problems like processing bookings for individuals and groups, managing customer data, monitoring payments, and maintaining effective communication among agents, guides, and customers. I also examined existing manual system issues—such as late updates, lost feedback, and no proper tracking—which underscored the need for an automated CRM solution.

Defining Project Scope & Objectives

The project was to develop a scalable CRM that could be utilized by travel agencies all over the world. Its scope spanned the complete booking process—from customer inquiry to payments and feedback collection. The objectives were well-defined: minimize manual effort, enhance customer satisfaction, and provide managers with insights into revenue and performance through reports and dashboards. One of the essential aspects of the scope was also to ensure role-based access, so various teams (agents, guides, finance team, and admins) only view and deal with what is appropriate to them.

Gathering & Analyzing User Needs

In order to make the system useful, I researched the needs of various Users: customers, travel agents, guides, finance teams, and admins. Customers wished for easy onboarding, package selection, and immediate booking confirmation. Agents and guides required tools to handle bookings, tasks, and communication. Finance staff needed to see payments and reminders, whereas admins wanted reports for measuring performance. I collected these needs through Google Forms, internet research, Salesforce guides, and Trailhead modules, and the design was thus grounded in real-life scenarios.

Identifying Key Salesforce Features & Tools Required

Based on analysis, I enumerated the Salesforce tools that could effectively address these needs. Custom objects such as Booking, Travel Package, Customer, Employee, and Payment would store all key information. Automation features such as Flows, Workflow Rules, and Approval Processes would do repetitive tasks like reminders and confirmations. Apex code (Triggers, Batch Apex, Queueable Apex) would execute more complicated tasks. For the Users interface, Lightning Apps, Dynamic Forms, and LWC components would make it easy to use. On the security front, Profiles, Roles, Permission Sets, and Sharing Rules would provide adequate access.

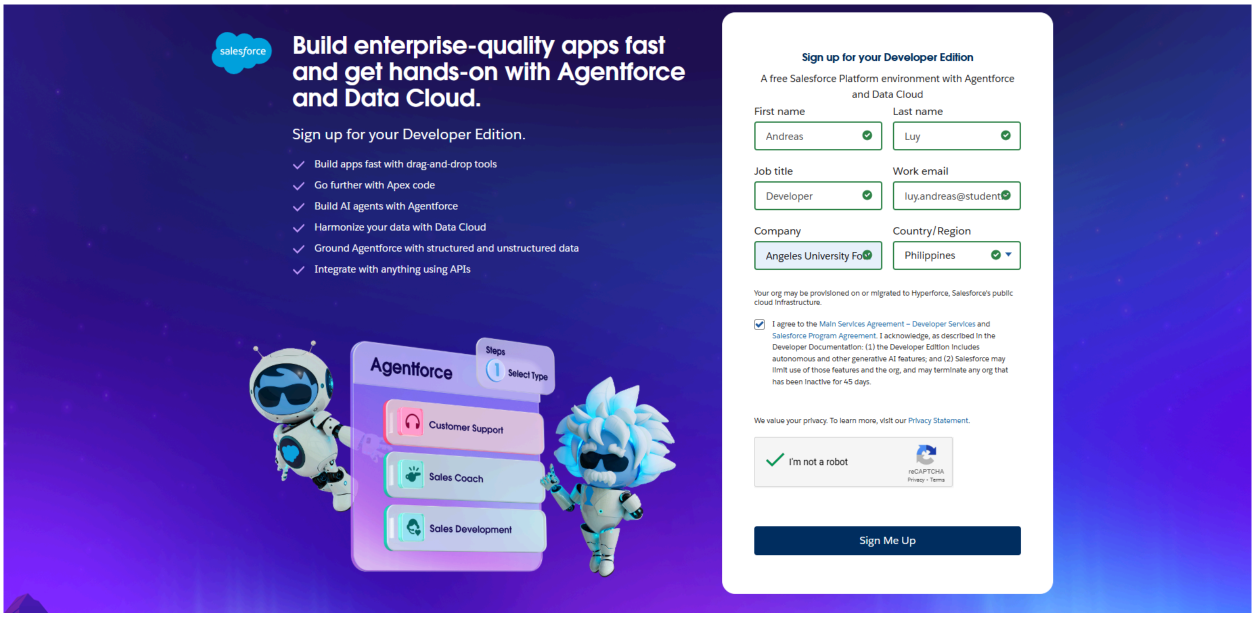
Designing Data Model and Security Model

The data model was created in order to connect important entities to one another: Bookings with Customers, Travel Packages, and Employees, and to monitor payments, feedback, and guest information. This provided a structure where all information related to bookings was logically connected. The security design was based on a straightforward hierarchy—Travel Managers in charge of Agents and Guides. Profiles managed object-level access, and permission sets gave flexibility to grant further access as required. Sensitive information, such as payment details, was limited to finance Users, and sharing rules permitted guides to view only records in relation to their work.

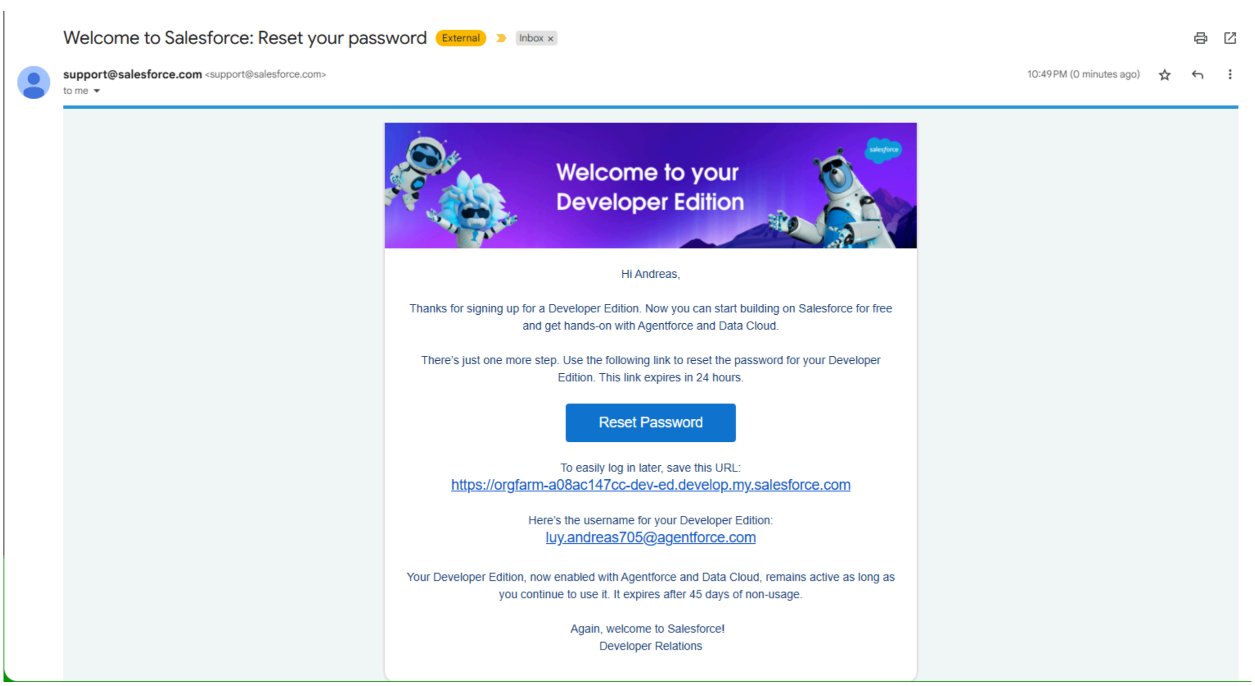
Phase 2: Salesforce Development – Backend & Configurations

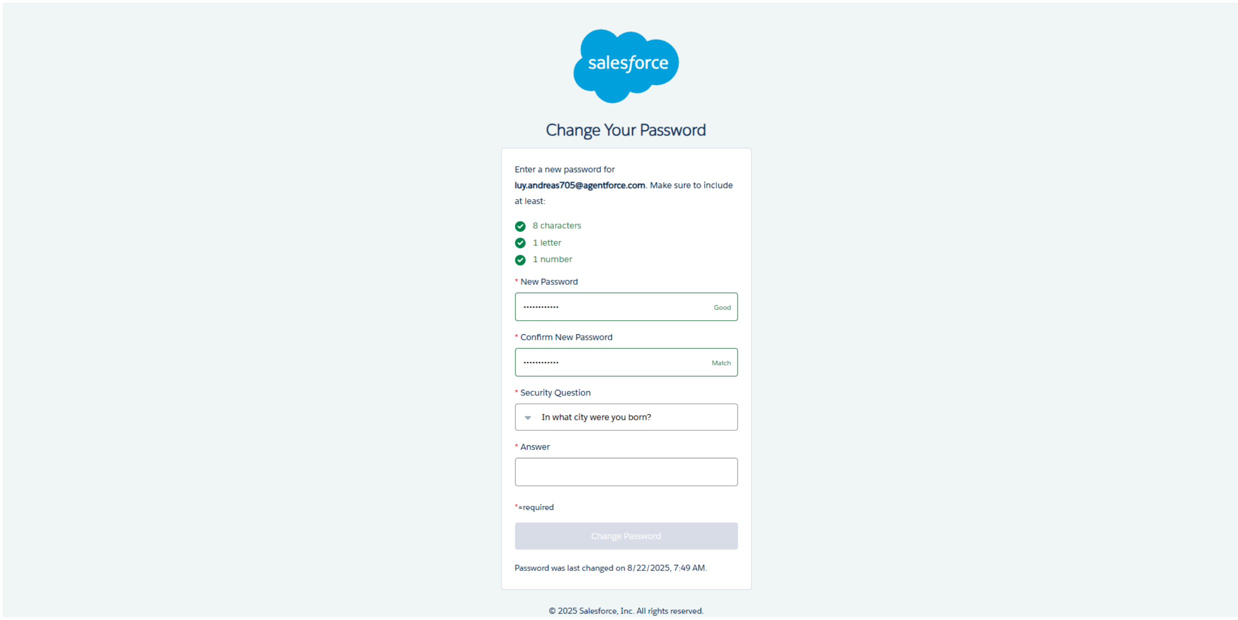
Milestone 1: Salesforce Account

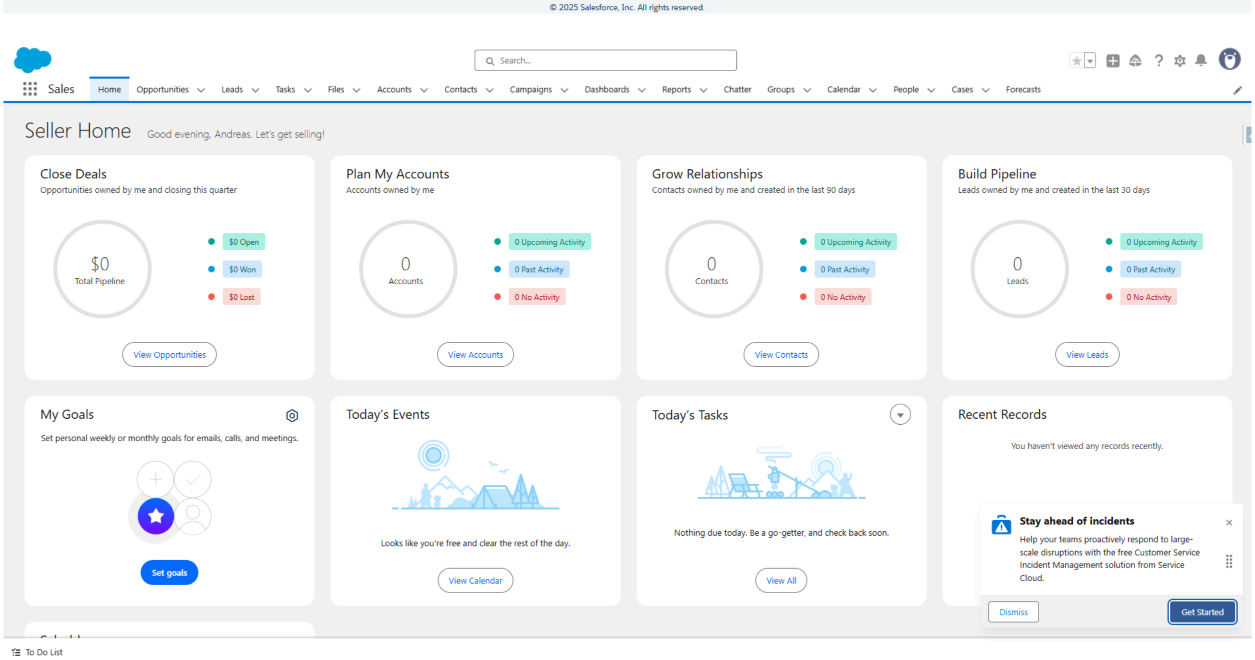
Activity 1: Creating Developer Account



Activity 2: Account Activation





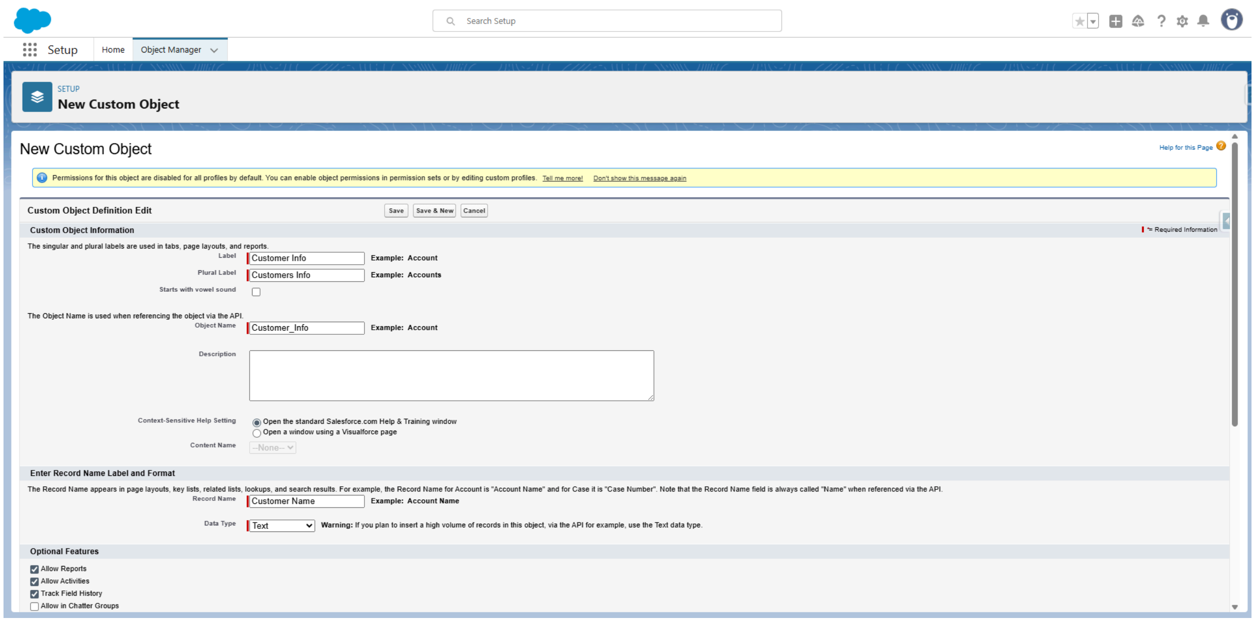


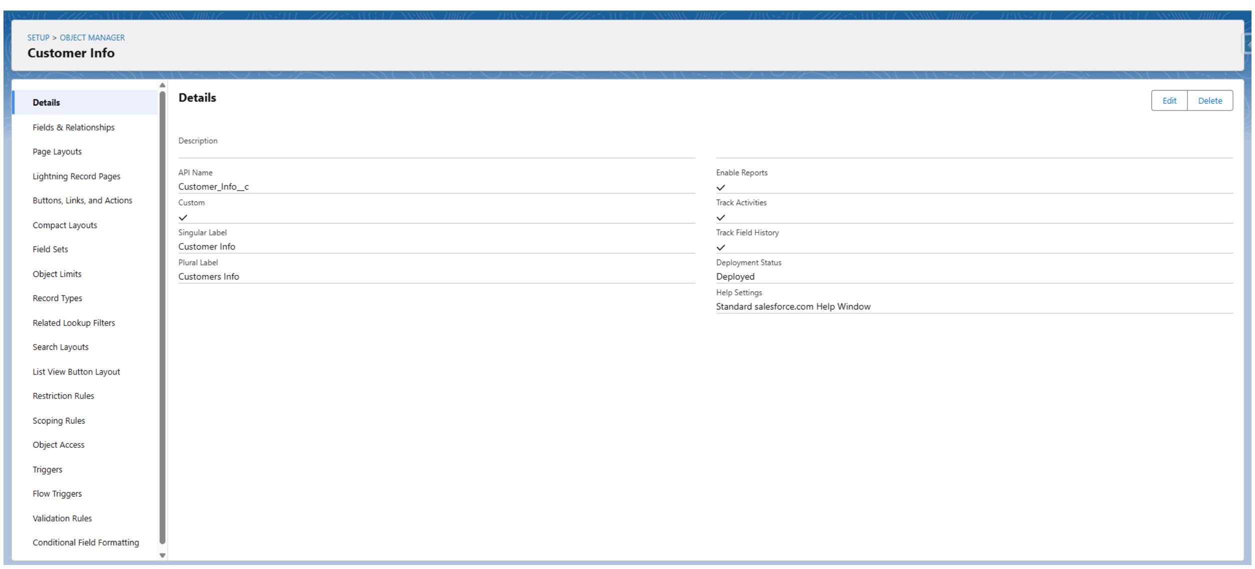
For this milestone, I began by setting up my own account on Salesforce Developer Org. To create my own account, I began by following the steps on the activity. Setting up the developer account was very straightforward and marked my first step to the Salesforce ecosystem. It gave me access to the system where I could start experimenting and exploring different features and functionalities. This is the first step of my learning journey.

Milestone 2: Objects Creation

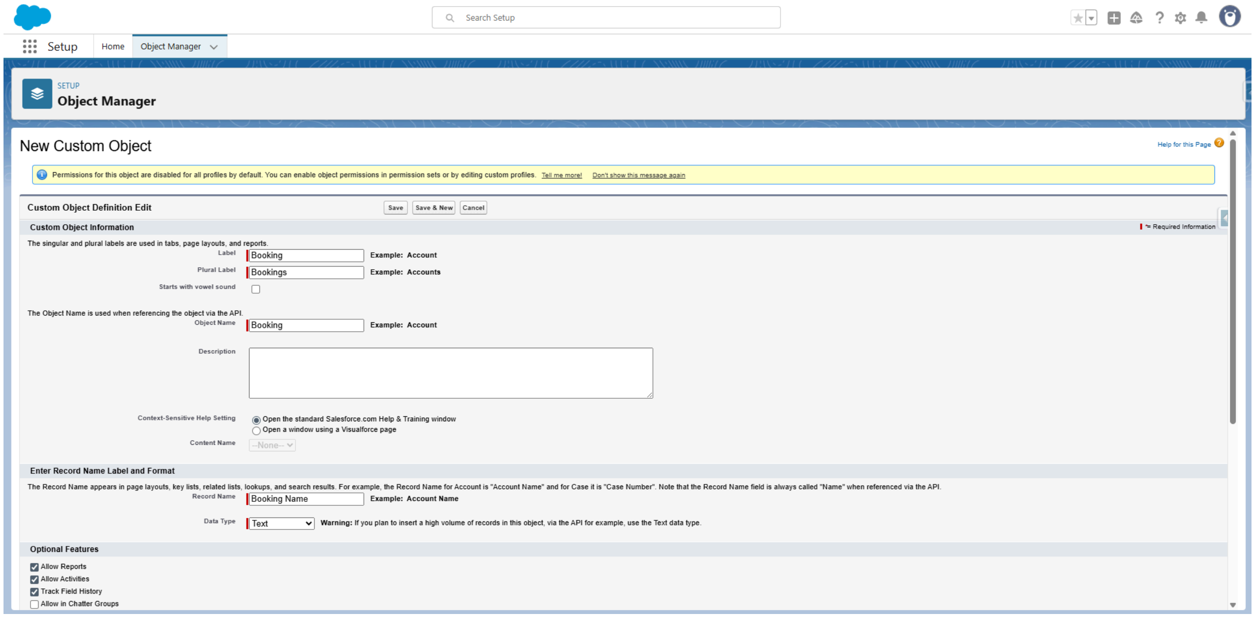
Activity 1: Creating a Customer Info Object

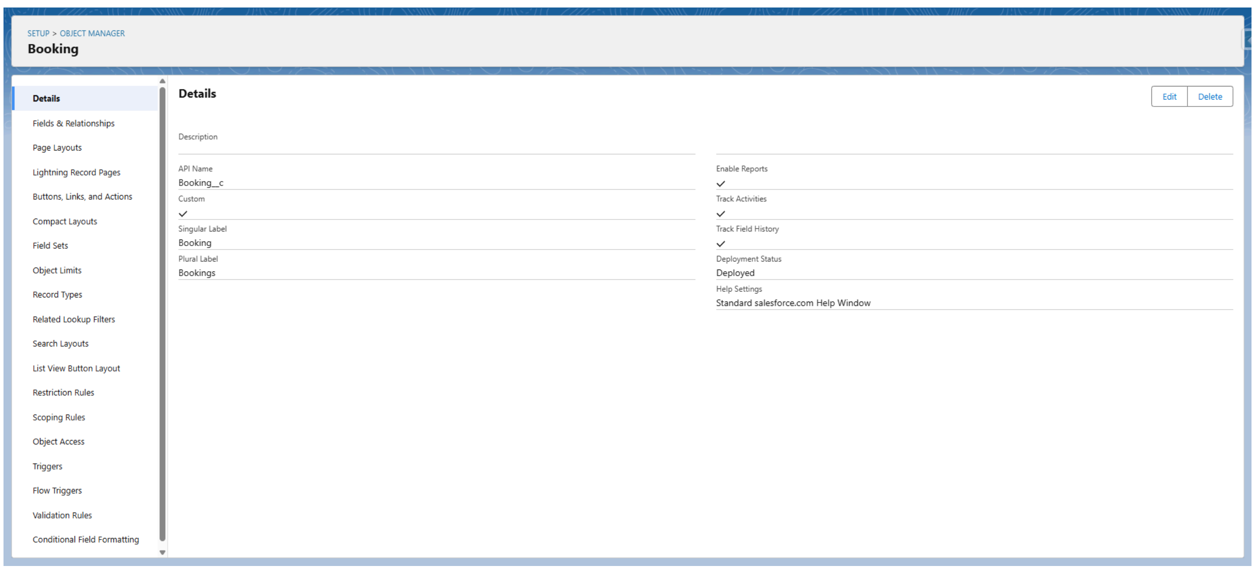
Customer Info Object



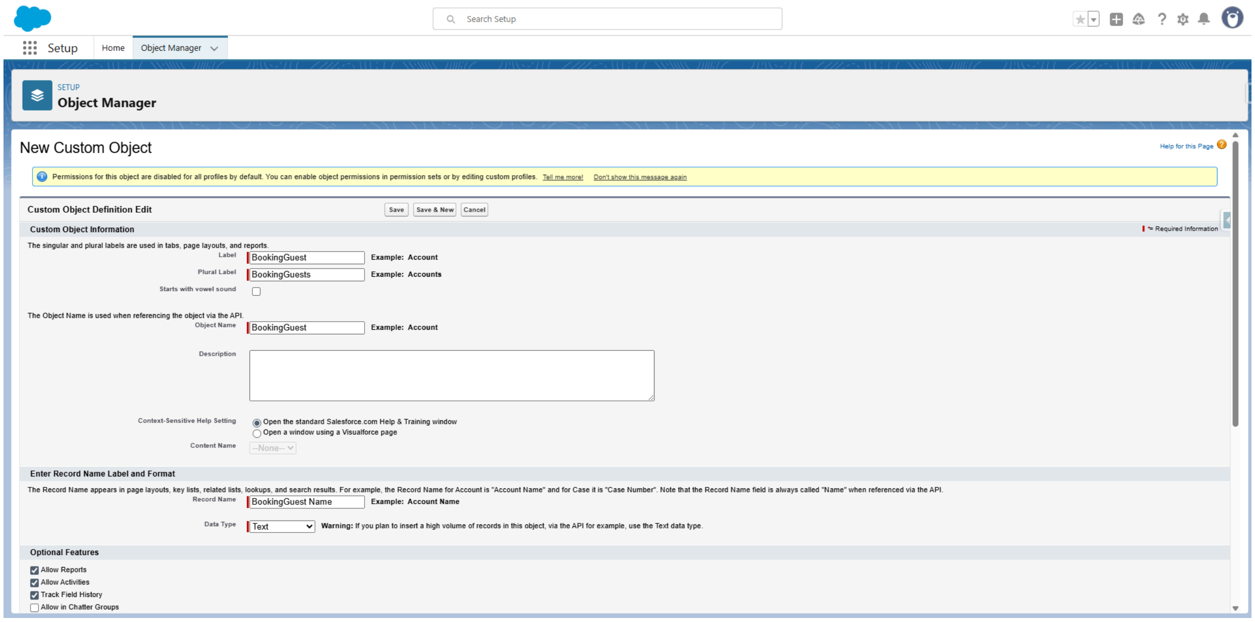


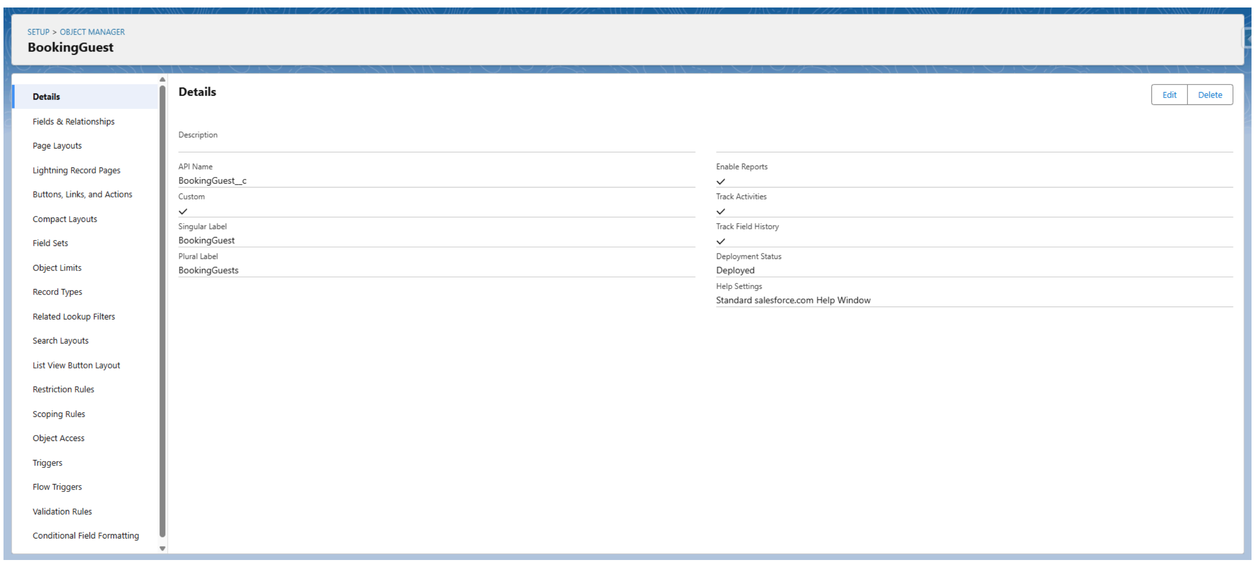
Booking Object



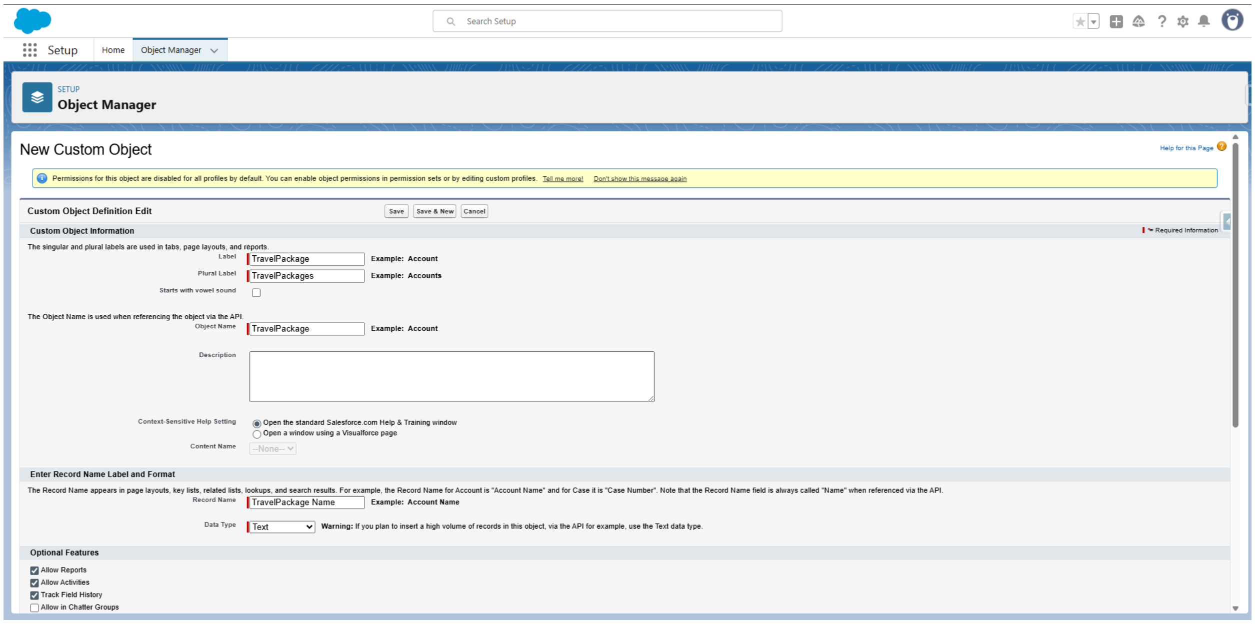


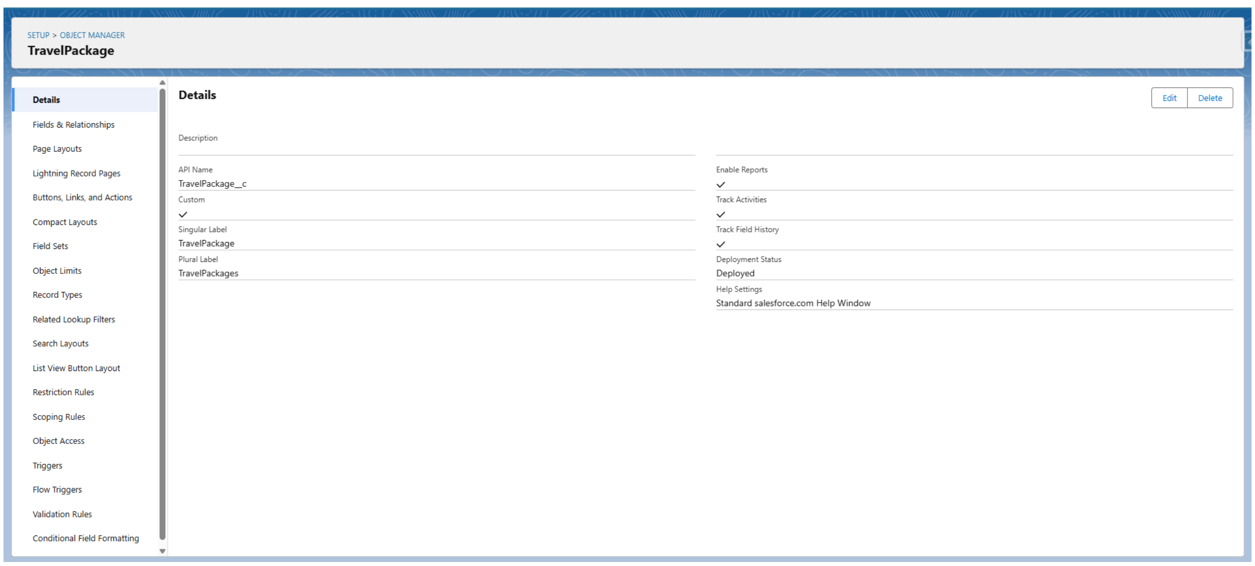
BookingGuest Object



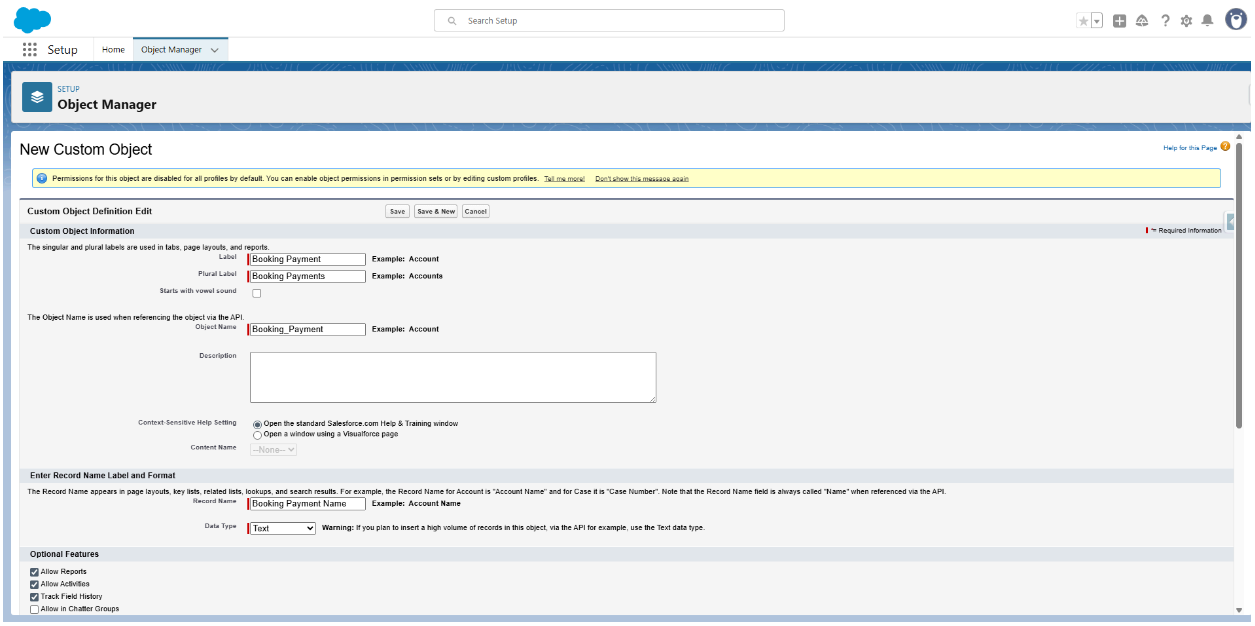


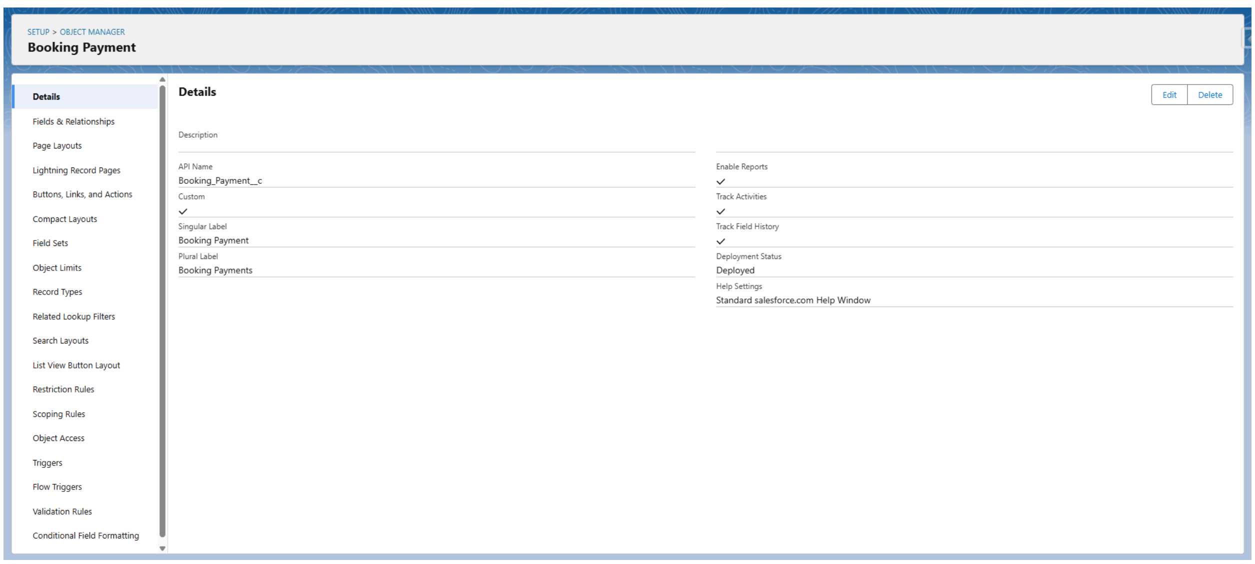
TravelPackage Object



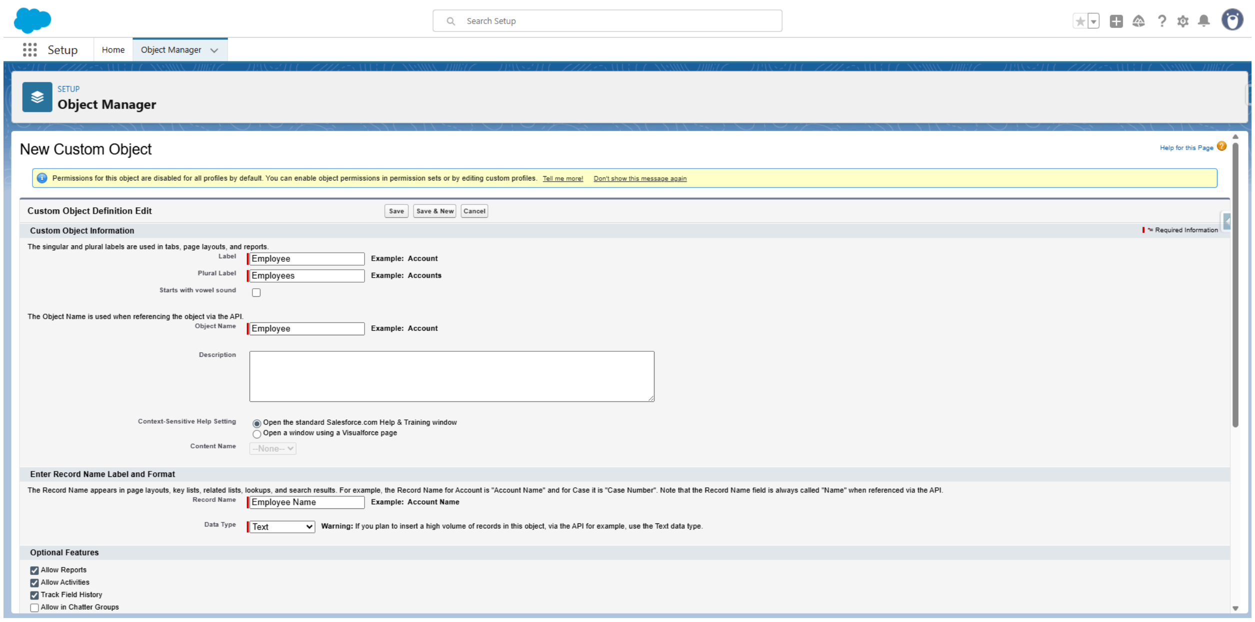


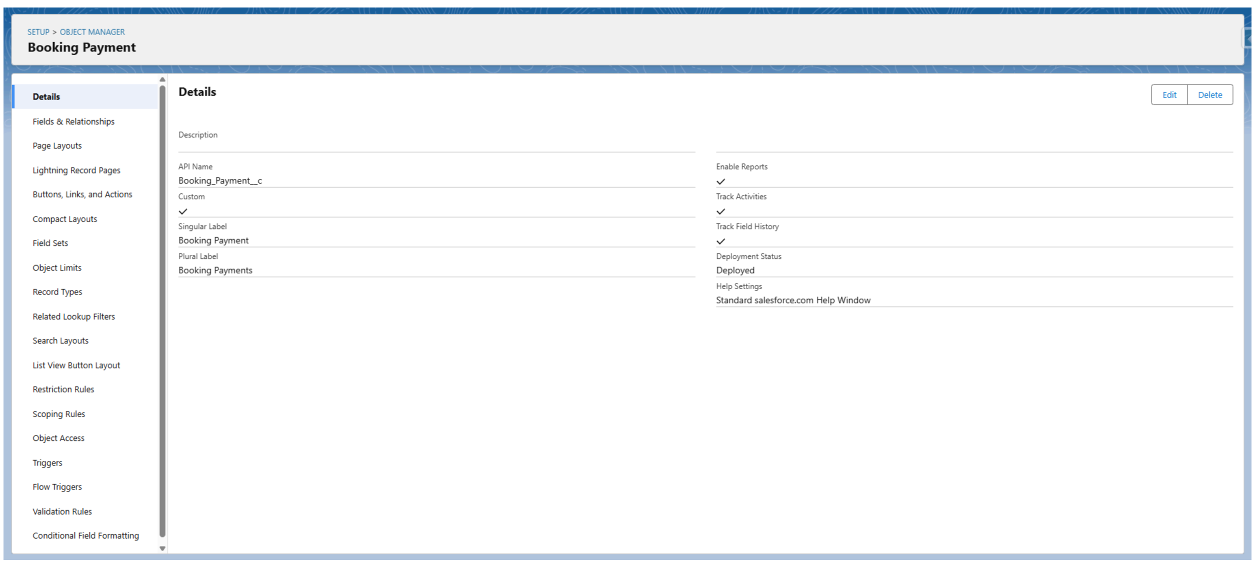
Booking Payment Object



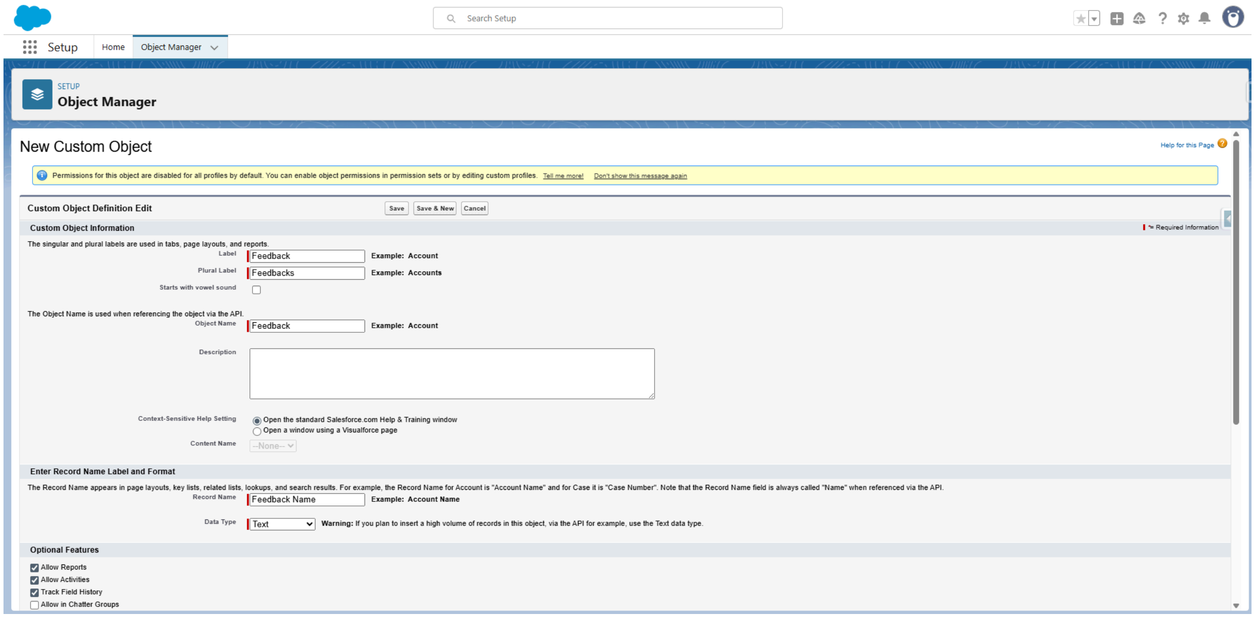


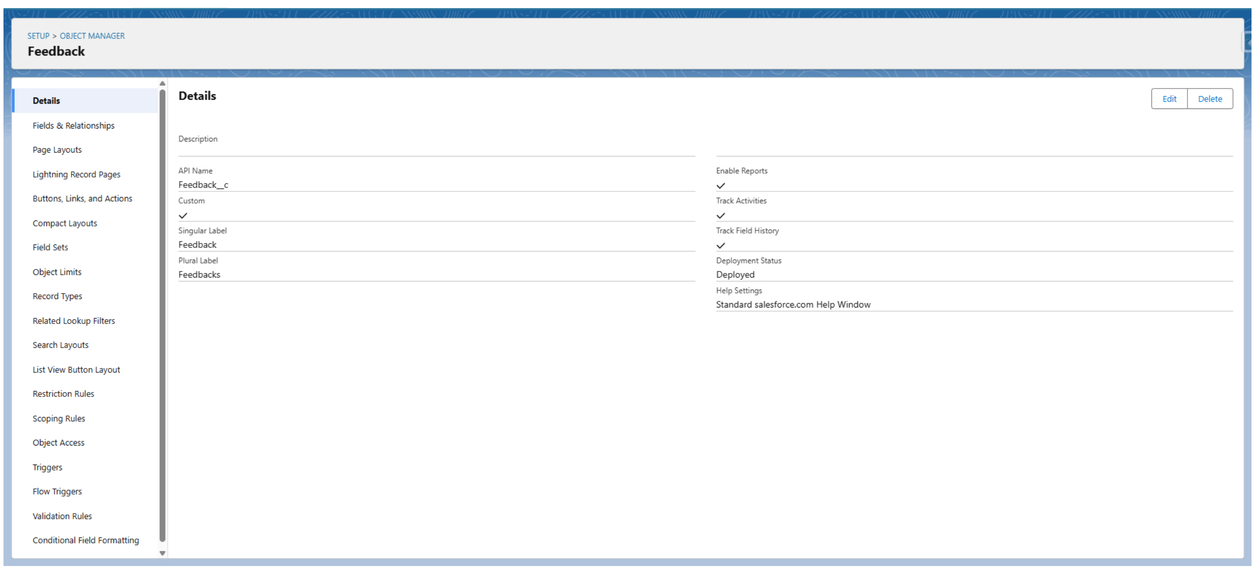
Employee Object





Feedback Object



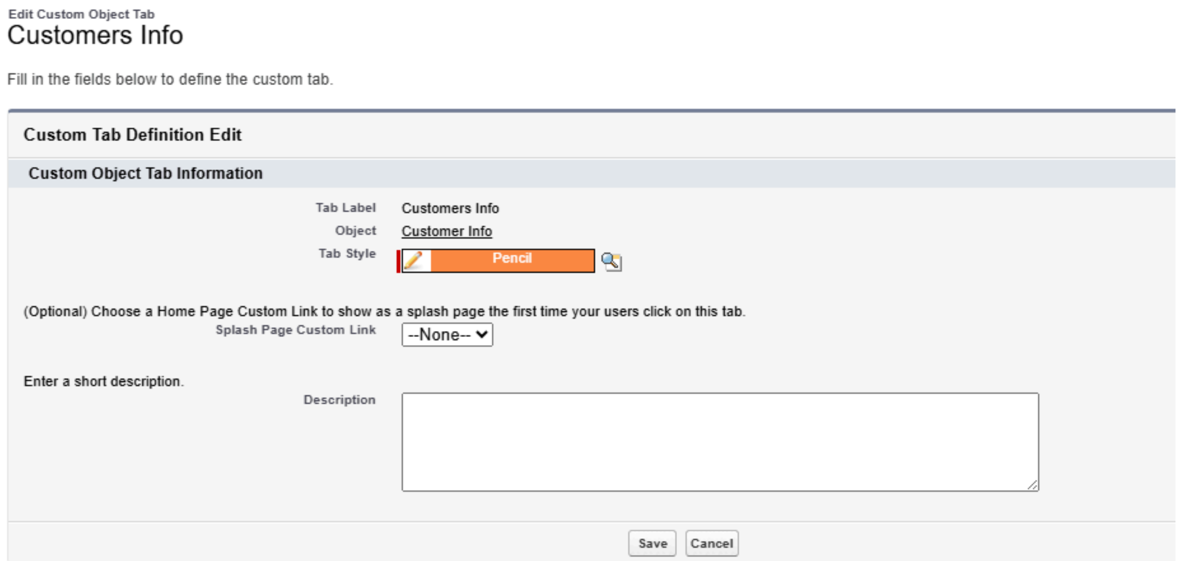


In Salesforce, objects are like the database tables in MySQL. They store the data and act as the foundation of the CRM's data model. Each object helps organize and manage the types of info that is relevant to the Tours and Travels CRM app.

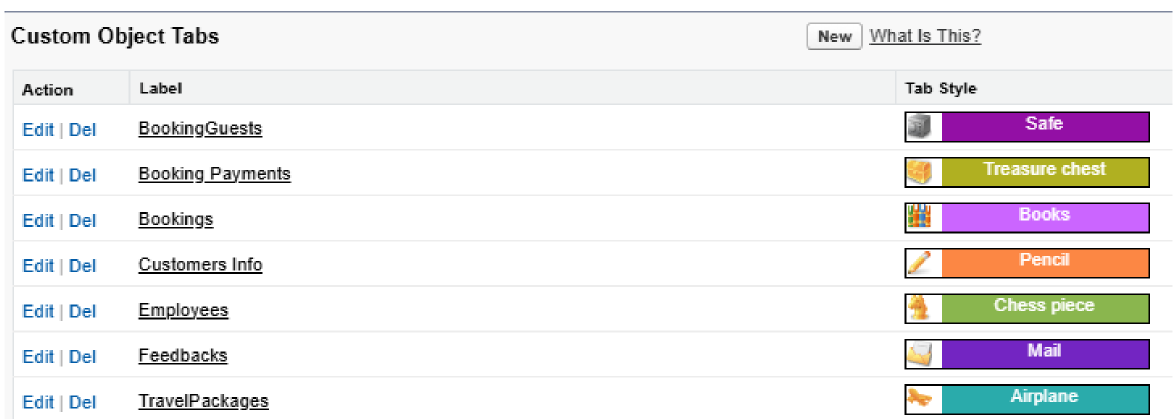
This Milestone helped me understand how Salesforce objects are made and how they form the backbone of the app's data model. By creating the different custom objects, I can somewhat visualize how they could be connected together in the Tours and Travels CRM.

Milestone 3: Tabs

Activity 1: Creating a tab for Customer Info Object



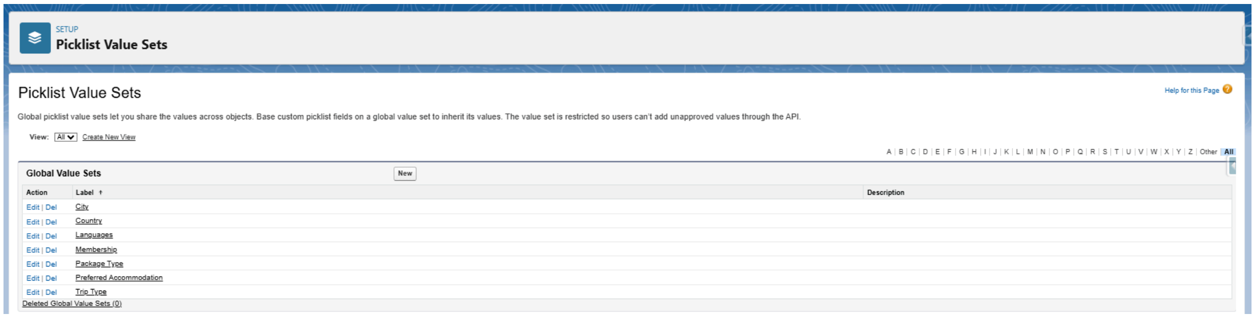
Activity 2: Creating Remaining Tabs



In Salesforce, the Tabs are the main entry points to access the object's data. They appear in the navigation bar of the application. Creating tabs will ensure that Users can easily access what they need. For this milestone, I learned how the creation of tabs brought the objects to life. Until now, the objects only existed in the background, but with tabs, each of them became directly accessible to Users. It took me some time to pick the correct tab style for each object, it was fun though!

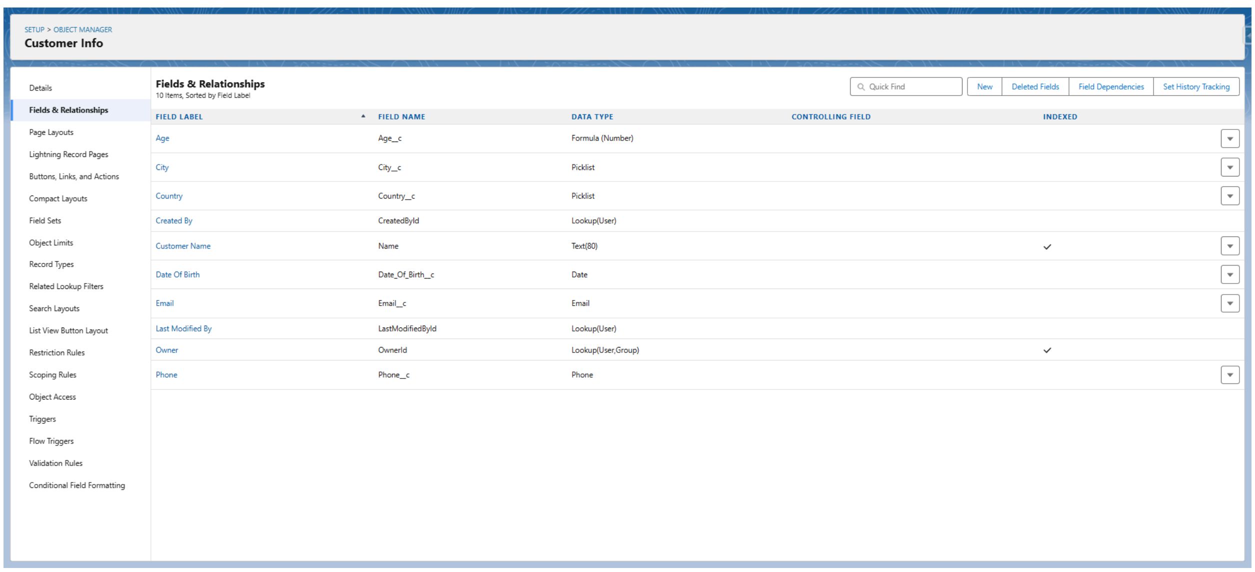
Milestone 4: Fields & Relationships

Picklist Value Sets



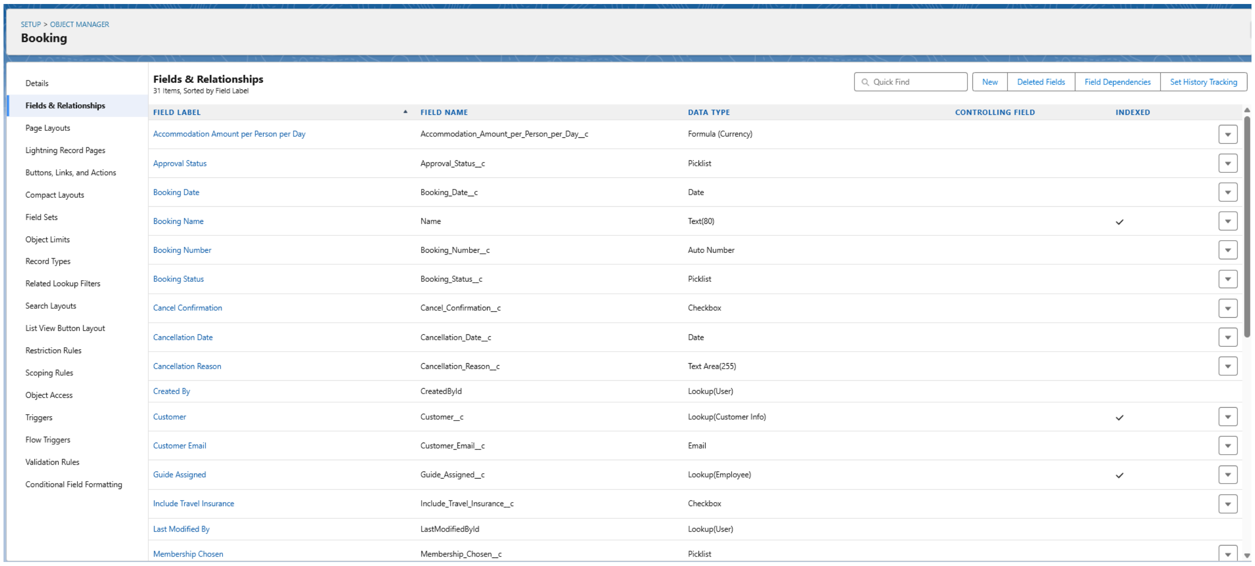
Activity 1: Creation of fields for the Customer Info object

Customer Info Object Fields



Activity 2: Creation of fields for the Booking Object

Booking Object Fields



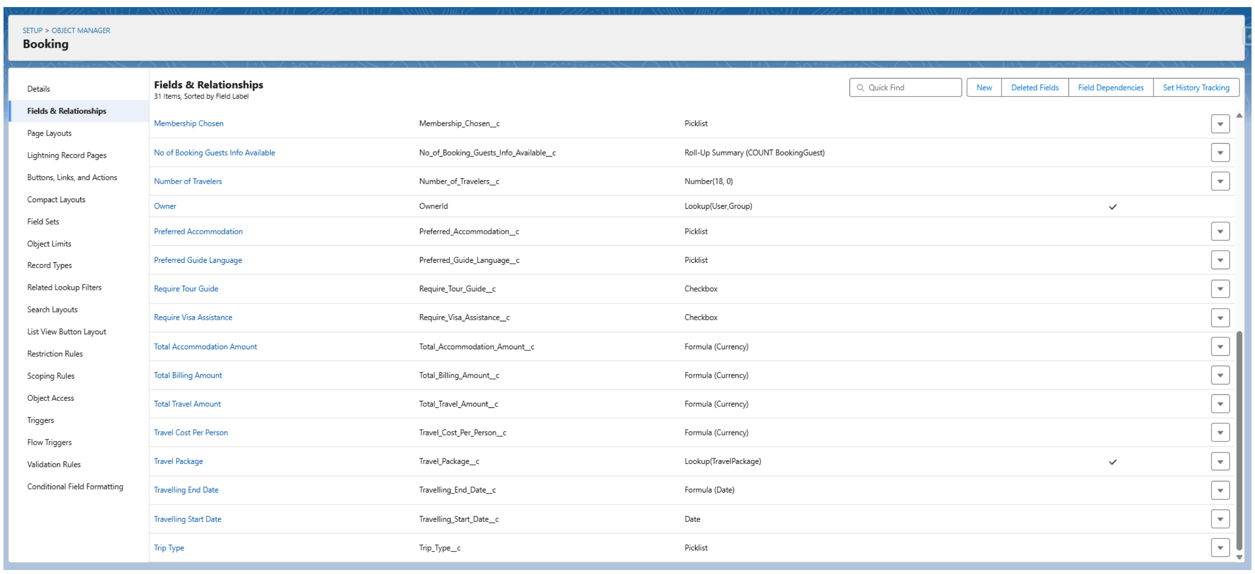
SETUP > OBJECT MANAGER
Booking

Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting

Fields & Relationships
31 Items, Sorted by Field Label

Quick Find | New | Deleted Fields | Field Dependencies | Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Accommodation Amount per Person per Day	Accommodation_Amount_per_Person_per_Day__c	Formula (Currency)		
Approval Status	Approval_Status__c	Picklist		
Booking Date	Booking_Date__c	Date		
Booking Name	Name	Text(80)		✓
Booking Number	Booking_Number__c	Auto Number		
Booking Status	Booking_Status__c	Picklist		
Cancel Confirmation	Cancel_Confirmation__c	Checkbox		
Cancellation Date	Cancellation_Date__c	Date		
Cancellation Reason	Cancellation_Reason__c	Text Area(255)		
Created By	CreatedById	Lookup(User)		
Customer	Customer__c	Lookup(Customer Info)		✓
Customer Email	Customer_Email__c	Email		
Guide Assigned	Guide_Assigned__c	Lookup(Employee)		✓
Include Travel Insurance	Include_Travel_Insurance__c	Checkbox		
Last Modified By	LastModifiedById	Lookup(User)		
Membership Chosen	Membership_Chosen__c	Picklist		



SETUP > OBJECT MANAGER
Booking

Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting

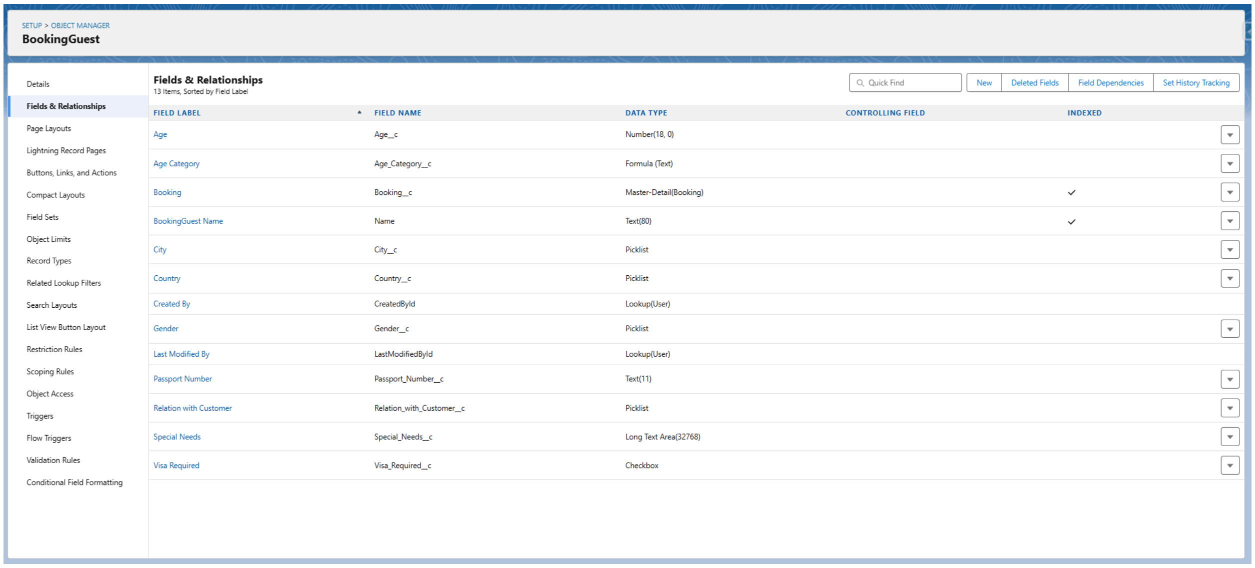
Fields & Relationships
31 Items, Sorted by Field Label

Quick Find | New | Deleted Fields | Field Dependencies | Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Membership Chosen	Membership_Chosen__c	Picklist		
No of Booking Guests Info Available	No_of_Booking_Guests_Info_Available__c	Roll-Up Summary (COUNT BookingGuest)		
Number of Travelers	Number_of_Travelers__c	Number(18, 0)		
Owner	OwnerId	Lookup(User,Group)		✓
Preferred Accommodation	Preferred_Accommodation__c	Picklist		
Preferred Guide Language	Preferred_Guide_Language__c	Picklist		
Require Tour Guide	Require_Tour_Guide__c	Checkbox		
Require Visa Assistance	Require_Visa_Assistance__c	Checkbox		
Total Accommodation Amount	Total_Accommodation_Amount__c	Formula (Currency)		
Total Billing Amount	Total_Billing_Amount__c	Formula (Currency)		
Total Travel Amount	Total_Travel_Amount__c	Formula (Currency)		
Travel Cost Per Person	Travel_Cost_Per_Person__c	Formula (Currency)		
Travel Package	Travel_Package__c	Lookup(TravelPackage)		✓
Travelling End Date	Travelling_End_Date__c	Formula (Date)		
Travelling Start Date	Travelling_Start_Date__c	Date		
Trip Type	Trip_Type__c	Picklist		

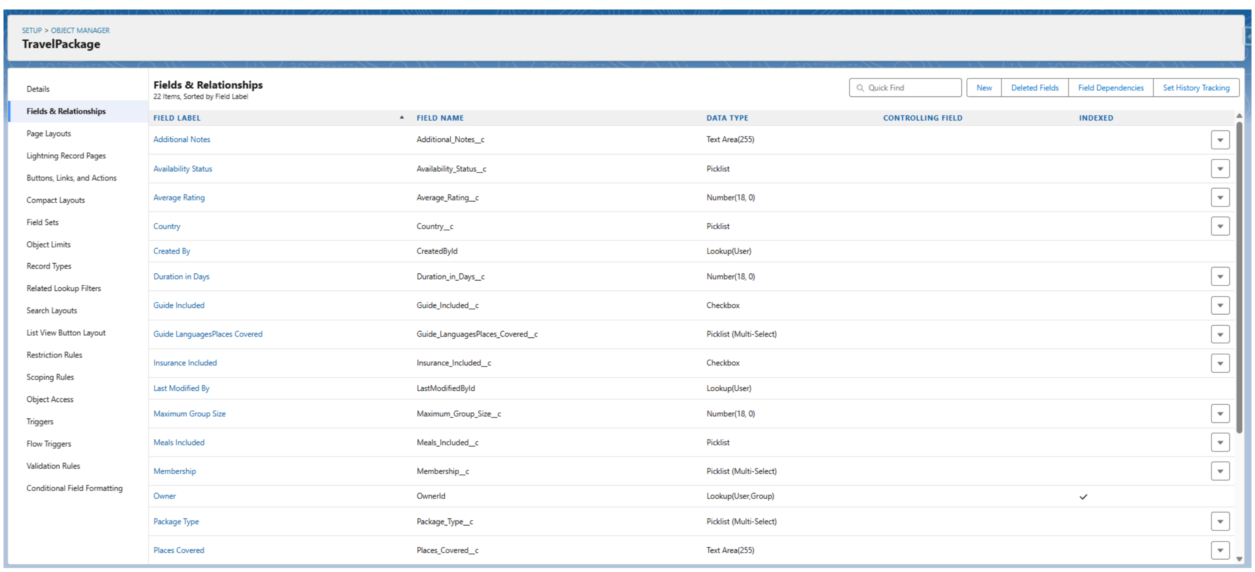
Activity 3: Creation of fields for the Booking Guest Object

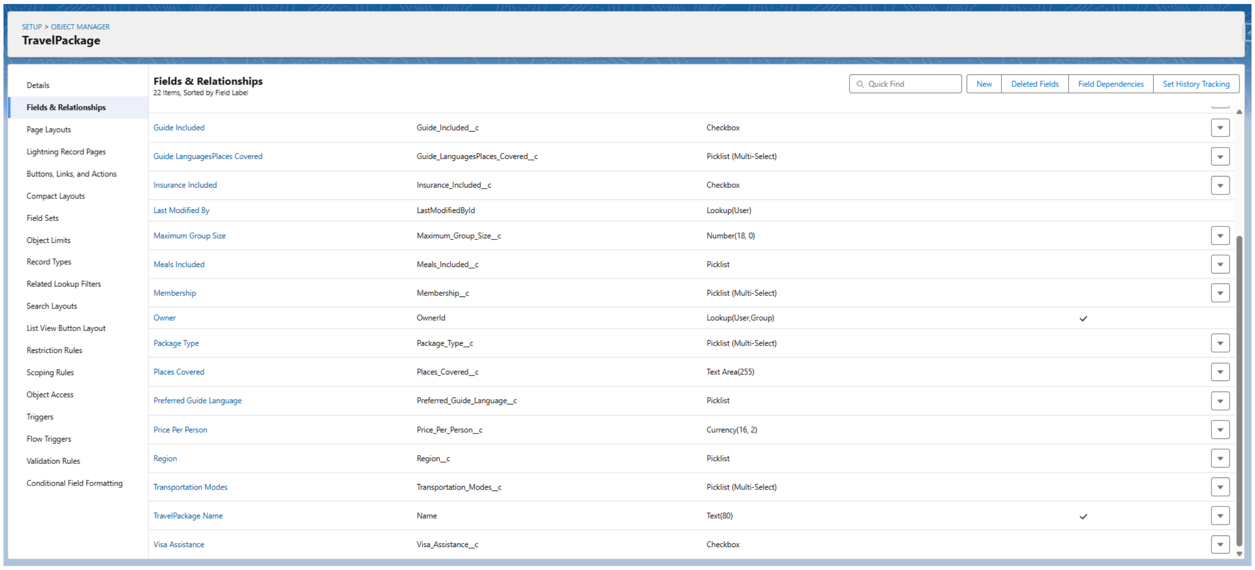
Booking Guest Object Fields



Activity 4: Creation of fields for the TravelPackage Object

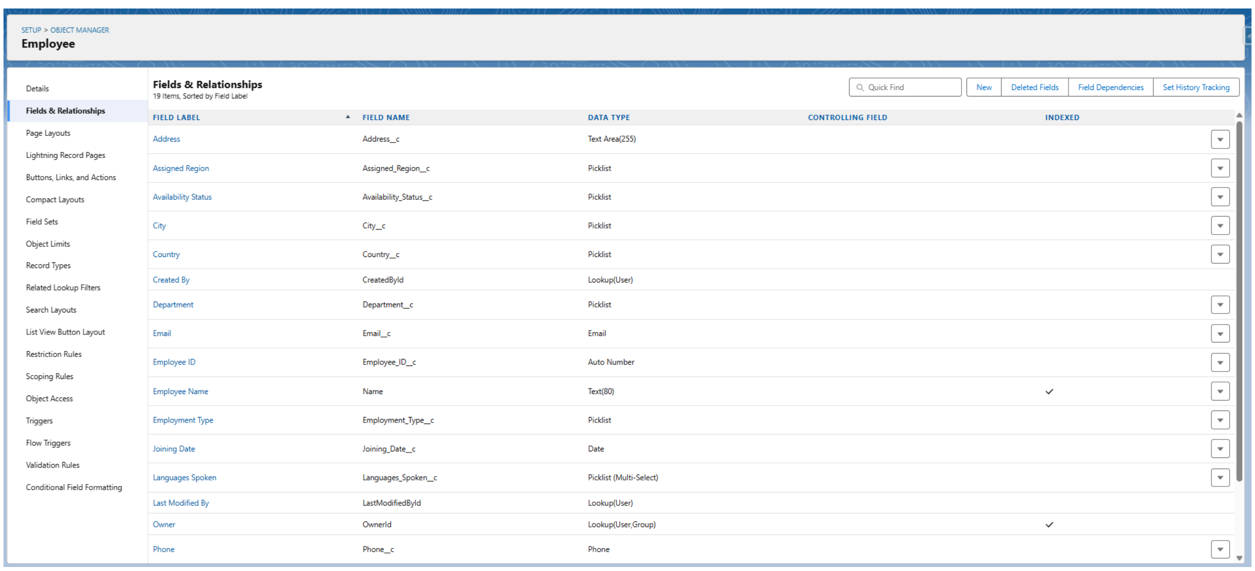
TravelPackage Object Fields

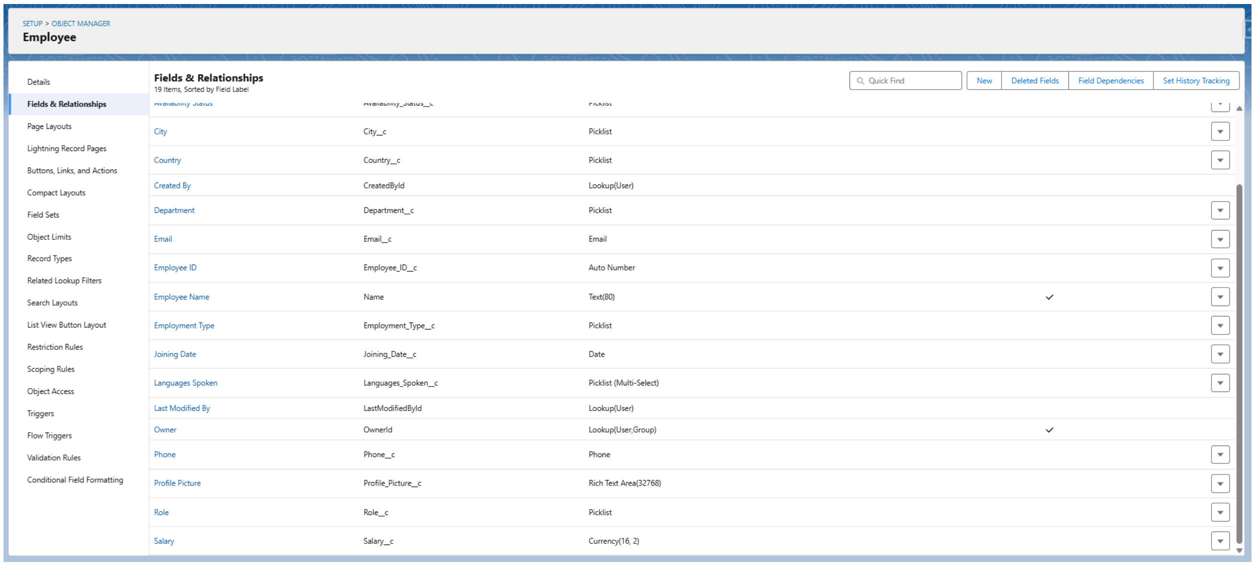




Activity 5: Creation of fields for the Employee Object

Employee Object Fields

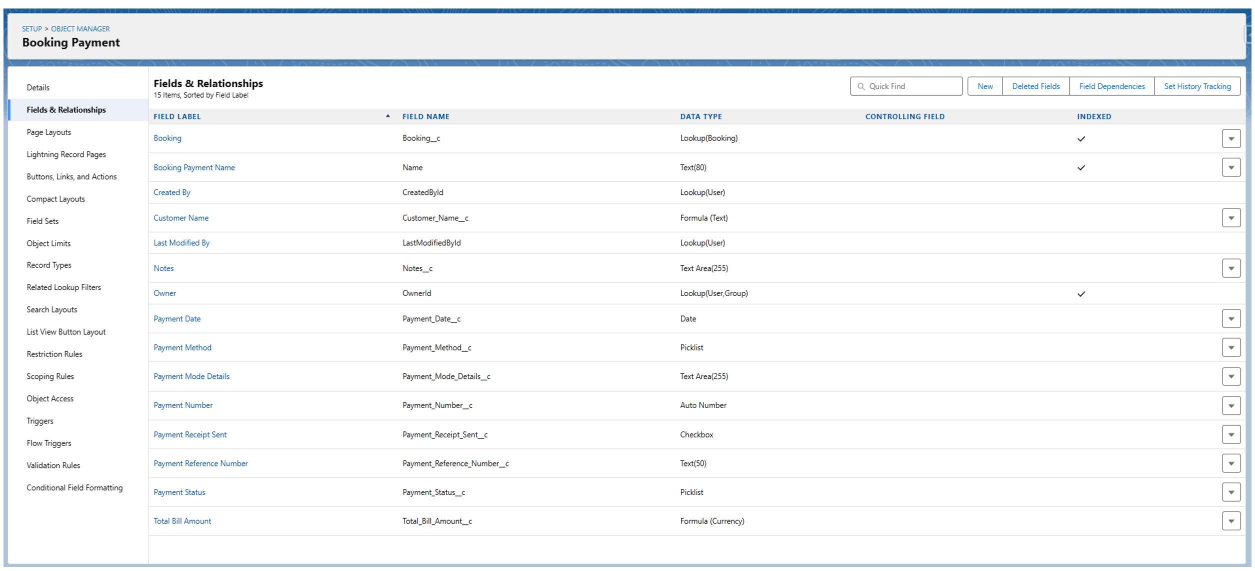




Availability Status	Availability_Status__c	Picklist	
City	City__c	Picklist	
Country	Country__c	Picklist	
Created By	CreatedById	Lookup(User)	
Department	Department__c	Picklist	
Email	Email__c	Email	
Employee ID	Employee_ID__c	Auto Number	
Employee Name	Name	Text(80)	✓
Employment Type	Employment_Type__c	Picklist	
Joining Date	Joining_Date__c	Date	
Languages Spoken	Languages_Spoken__c	Picklist (Multi-Select)	
Last Modified By	LastModifiedById	Lookup(User)	
Owner	OwnerId	Lookup(User,Group)	✓
Phone	Phone__c	Phone	
Profile Picture	Profile_Picture__c	Rich Text Area(32768)	
Role	Role__c	Picklist	
Salary	Salary__c	Currency(16, 2)	

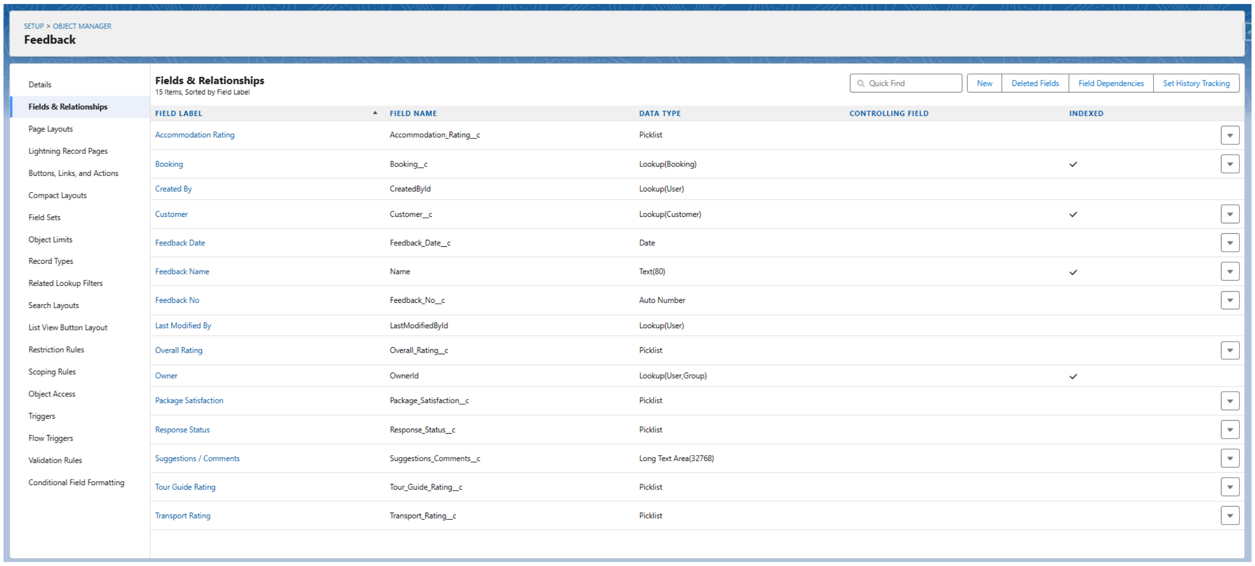
Activity 6: Creation of fields for the Booking Payment Object

Booking Payment Object Fields



FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Booking	Booking__c	Lookup(Booking)		✓
Booking Payment Name	Name	Text(80)		✓
Created By	CreatedById	Lookup(User)		
Customer Name	Customer_Name__c	Formula (Text)		
Last Modified By	LastModifiedById	Lookup(User)		
Notes	Notes__c	Text Area(255)		
Owner	OwnerId	Lookup(User,Group)		✓
Payment Date	Payment_Date__c	Date		
Payment Method	Payment_Method__c	Picklist		
Payment Mode Details	Payment_Mode_Details__c	Text Area(255)		
Payment Number	Payment_Number__c	Auto Number		
Payment Receipt Sent	Payment_Receipt_Sent__c	Checkbox		
Payment Reference Number	Payment_Reference_Number__c	Text(50)		
Payment Status	Payment_Status__c	Picklist		
Total Bill Amount	Total_Bill_Amount__c	Formula (Currency)		

Feedback Object fields


SETUP > OBJECT MANAGER
Feedback

Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting

Fields & Relationships
15 Items, Sorted by Field Label

Quick Find | New | Deleted Fields | Field Dependencies | Set History Tracking

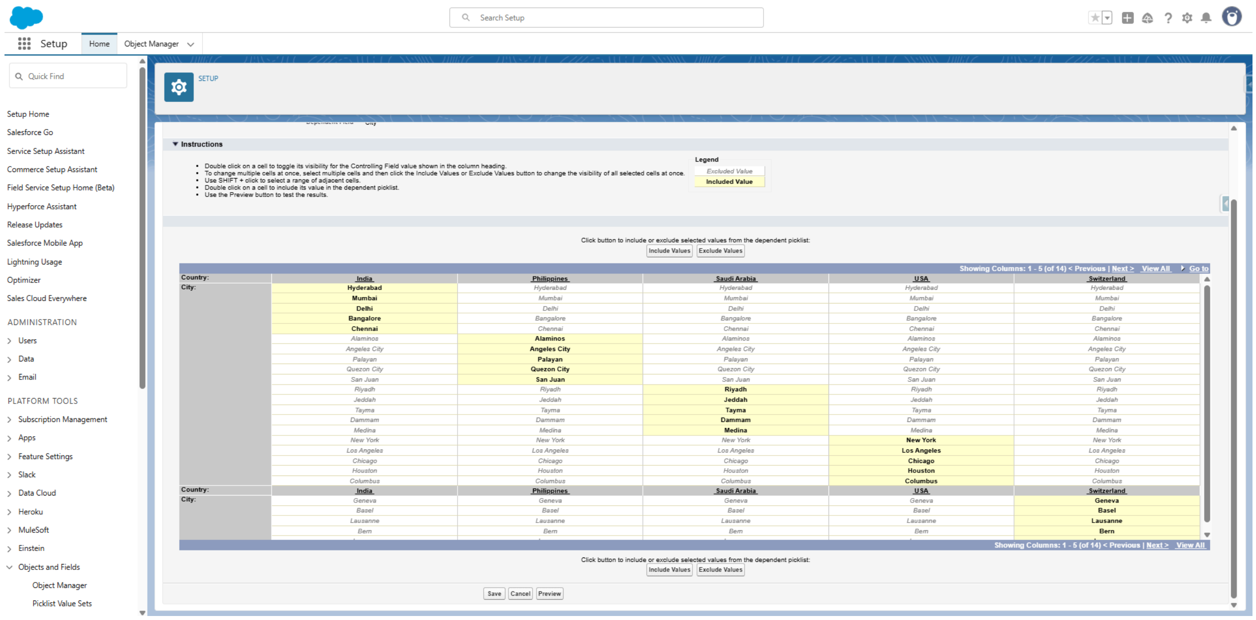
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Accommodation Rating	Accommodation_Rating__c	Picklist		
Booking	Booking__c	Lookup(Booking)		✓
Created By	CreatedById	Lookup(User)		
Customer	Customer__c	Lookup(Customer)		✓
Feedback Date	Feedback_Date__c	Date		
Feedback Name	Name	Text(80)		✓
Feedback No	Feedback_No__c	Auto Number		
Last Modified By	LastModifiedById	Lookup(User)		
Overall Rating	Overall_Rating__c	Picklist		
Owner	OwnerId	Lookup(User,Group)		✓
Package Satisfaction	Package_Satisfaction__c	Picklist		
Response Status	Response_Status__c	Picklist		
Suggestions / Comments	Suggestions_Comments__c	Long Text Area(32768)		
Tour Guide Rating	Tour_Guide_Rating__c	Picklist		
Transport Rating	Transport_Rating__c	Picklist		

In Salesforce, the fields and relationships are like the primary key, unique key, and the foreign key in MySQL. They determine how the data is captured, stored, and connected across the system. I also made use of the Picklist value sets in order to standardize the values across multiple objects. It ensures that the options are consistent and reduces redundancy.

This milestone is very important as it transformed the custom objects into meaningful data. By defining the custom fields and relationships, I built the structure of how information will flow through the CRM. It felt like creating an ERD where I connect the dots between the Customer Info, Bookings, Payments, and Feedback.

Milestone 5: Field Dependencies

Activity 1: Creating a Field Dependency in BookingGuest Object



BookingGuest Field Dependencies

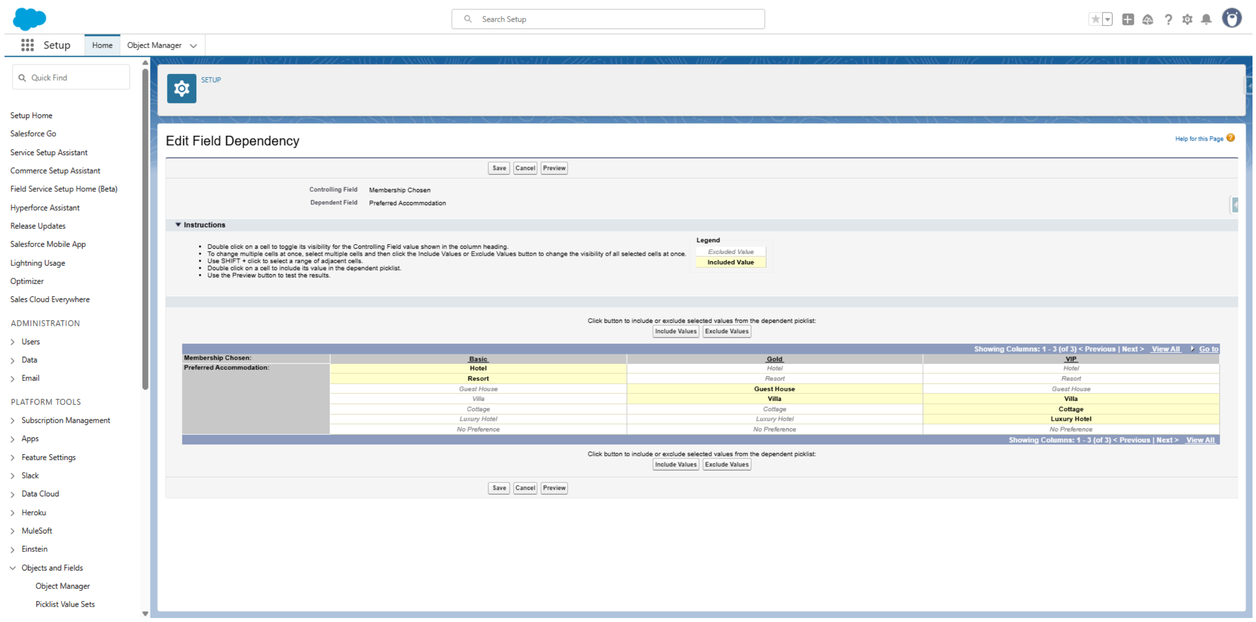
« Back to Custom Object: BookingGuest

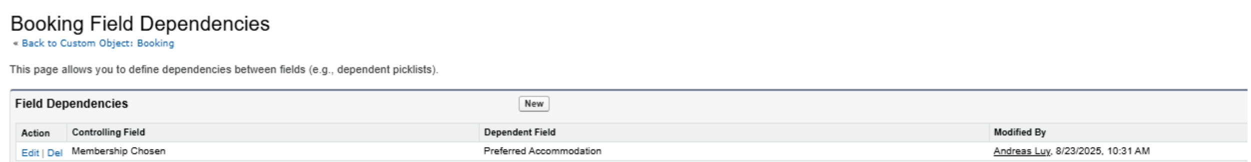
This page allows you to define dependencies between fields (e.g., dependent picklists).

Field Dependencies New

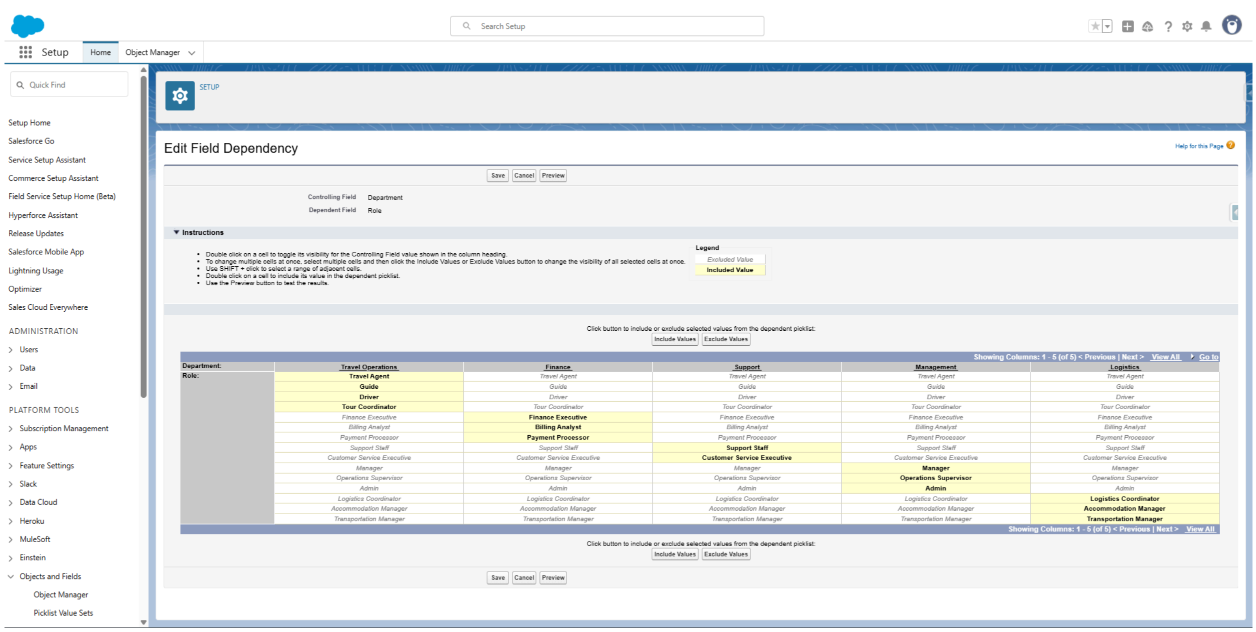
Action	Controlling Field	Dependent Field	Modified By
Edit | Del	Country	City	Andreas Luy, 8/23/2025, 10:29 AM

Activity 2: Creating a Field Dependency in Booking Object





Activity 3: Creating a Field Dependency in Employee Object



Employee Field Dependencies

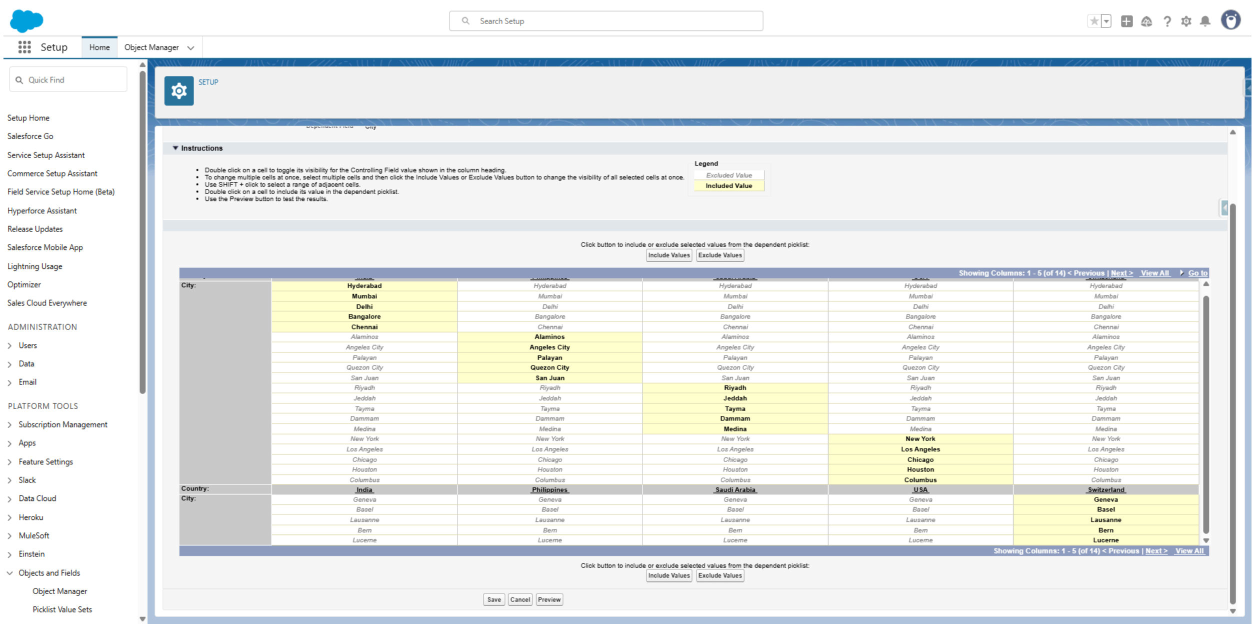
« Back to Custom Object: Employee

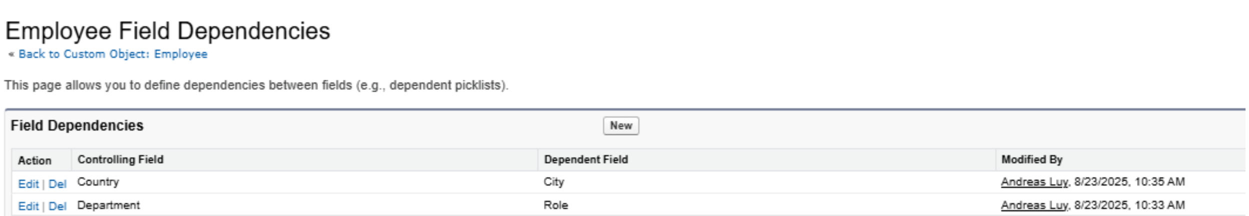
This page allows you to define dependencies between fields (e.g., dependent picklists).

Field Dependencies New

Action	Controlling Field	Dependent Field	Modified By
Edit | Del	Department	Role	Andreas Luy, 8/23/2025, 10:33 AM

Activity 4: Creating another Field Dependency in Employee Object





Employee Field Dependencies

« Back to Custom Object: Employee

This page allows you to define dependencies between fields (e.g., dependent picklists).

Field Dependencies New

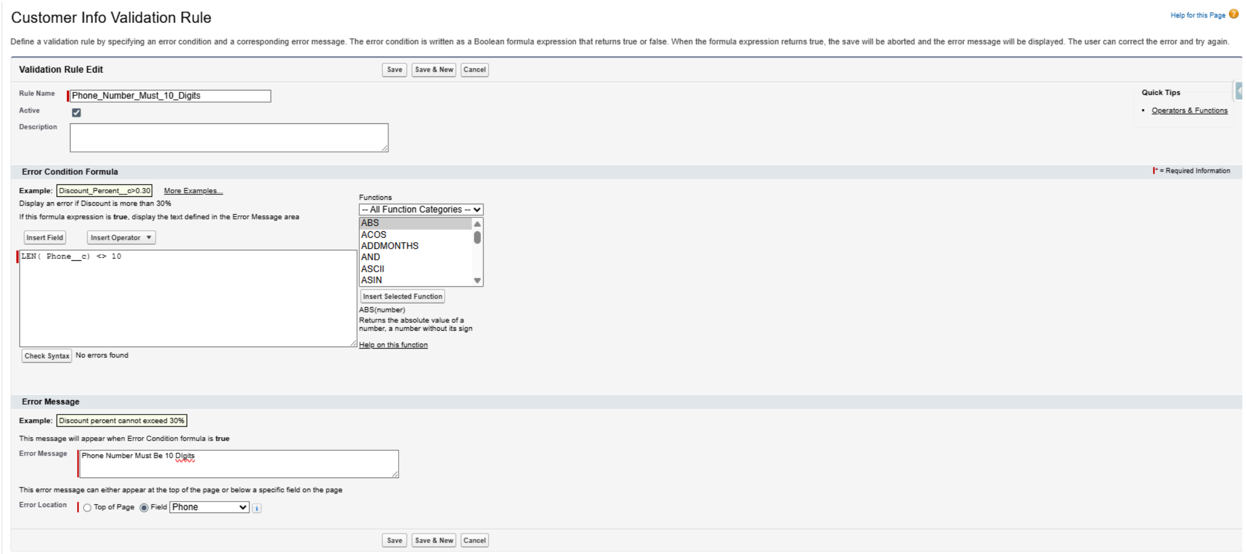
Action	Controlling Field	Dependent Field	Modified By
Edit | Del	Country	City	Andreas Luy, 8/23/2025, 10:35 AM
Edit | Del	Department	Role	Andreas Luy, 8/23/2025, 10:33 AM

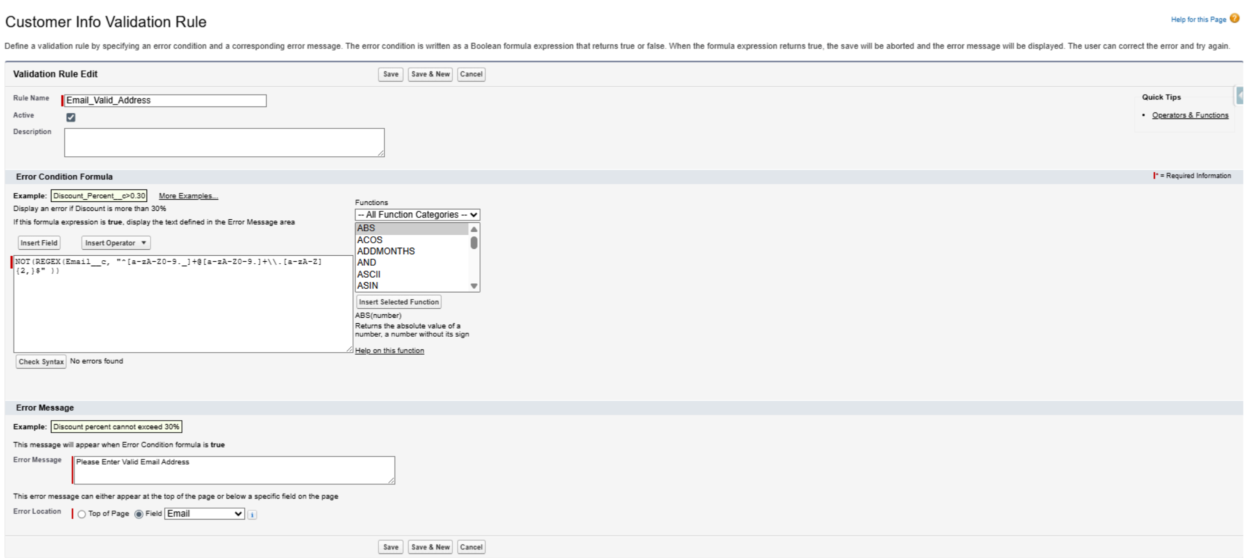
In Salesforce, the field dependencies are used to make the forms more dynamic by controlling which values appear in a dependent picklist that is also based on the controlling field.

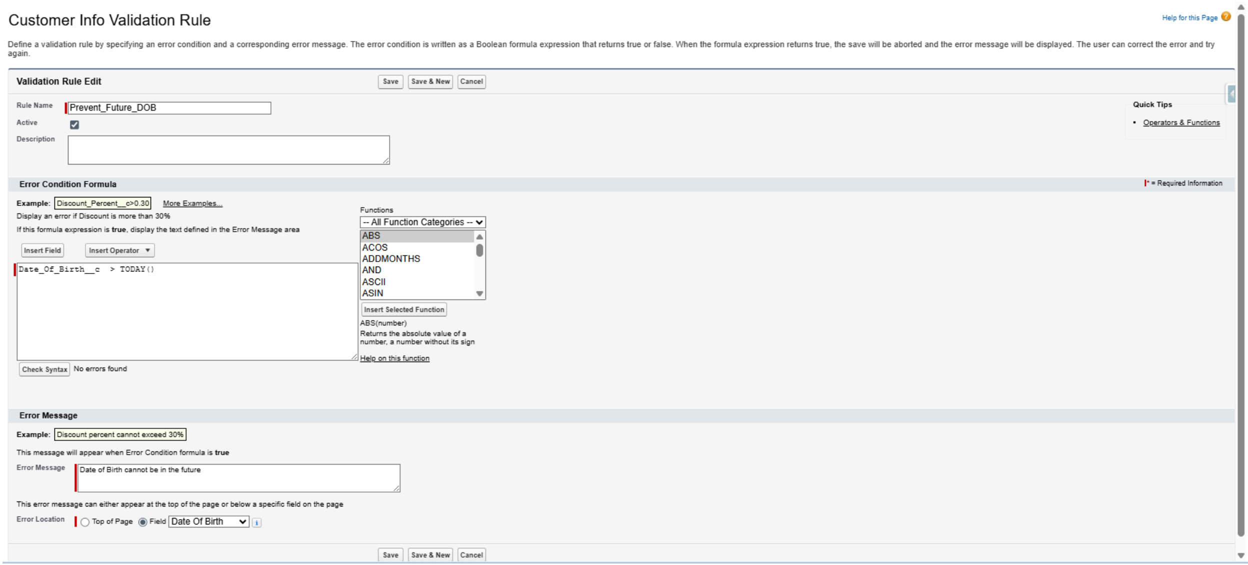
While creating the dependencies across the different objects such as BookingGuest, Booking, and Employee. I noticed that the forms are much cleaner than before. Instead of leaving the Users feel intimidated with how many fields they should enter. It felt like a small change but had a big impact in terms of usability.

Milestone 6: Validation Rules

Activity 1: To create a Validation rules to a Customer Info Object







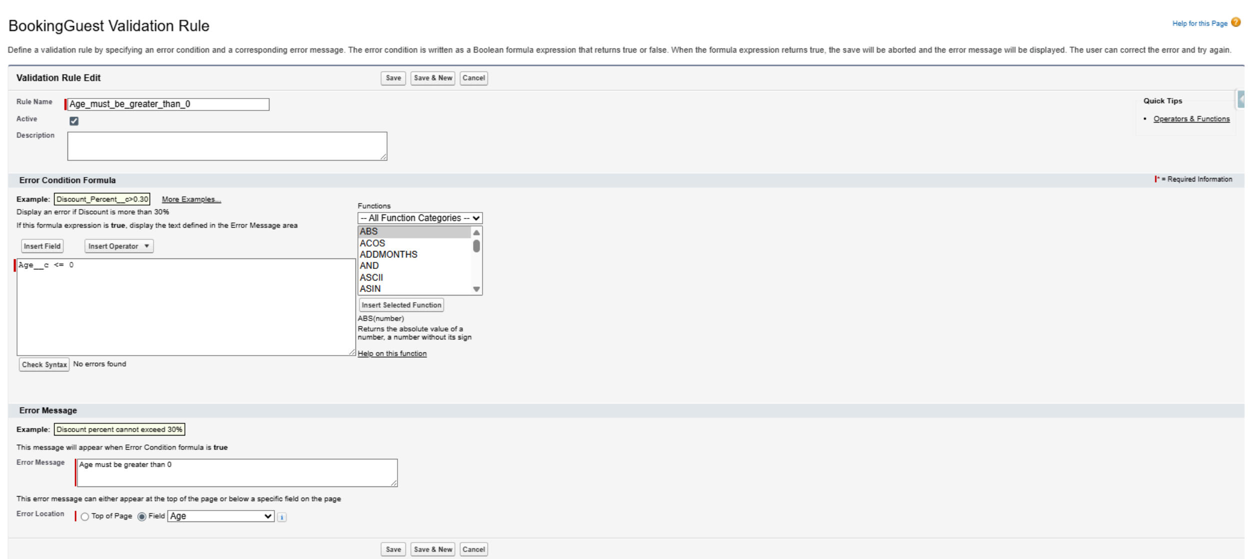
After adding the validations:

Validation Rules
3 Items, Sorted by Rule Name

New

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Email_Valid_Address	Email	Please Enter Valid Email Address	✓	Andreas Luy, 8/23/2025, 10:40 AM
Phone_Number_Must_10_Digits	Phone	Phone Number Must Be 10 DIgits	✓	Andreas Luy, 8/23/2025, 10:39 AM
Prevent_Future_DOB	Date Of Birth	Date of Birth cannot be in the future	✓	Andreas Luy, 8/23/2025, 10:41 AM

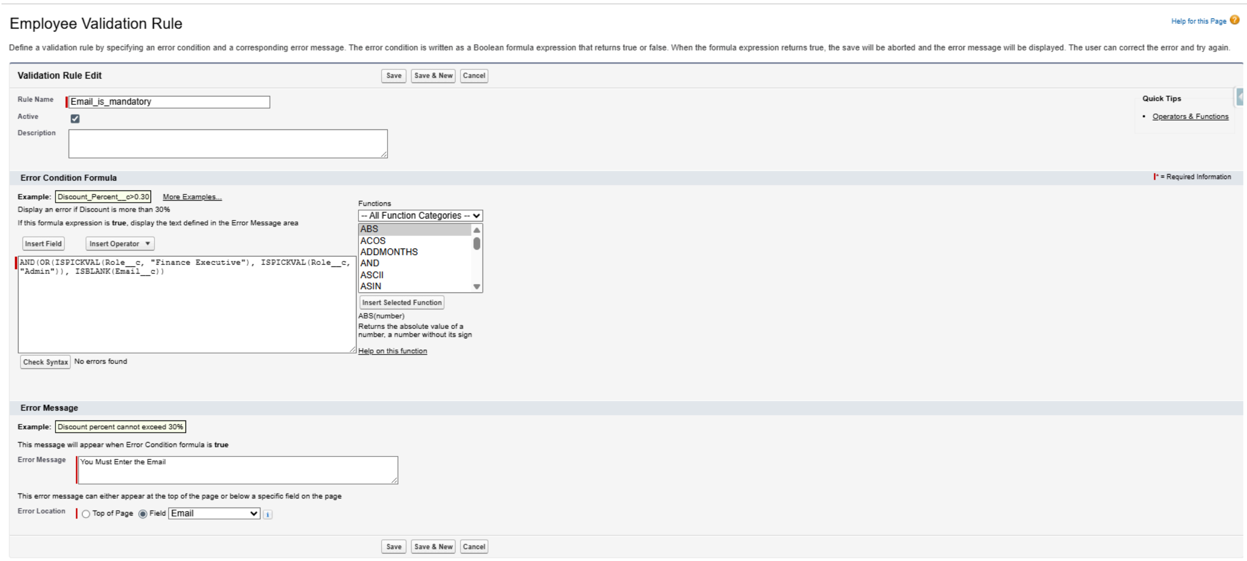
Activity 2: To create a Validation rules to a BookingGuest Object

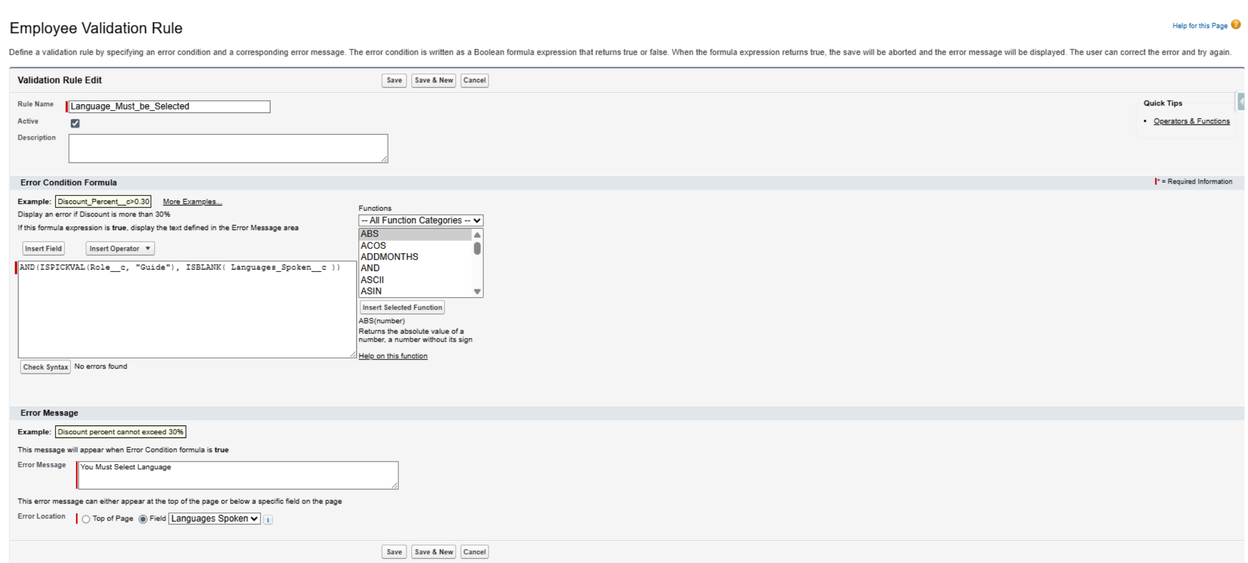


Validation Rules
1 Items, Sorted by Rule Name

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Age_must_be_greater_than_0	Age	Age must be greater than 0	✓	Andreas Luy, 8/23/2025, 10:43 AM

Activity 3: To create a Validation rules to an Employee Object

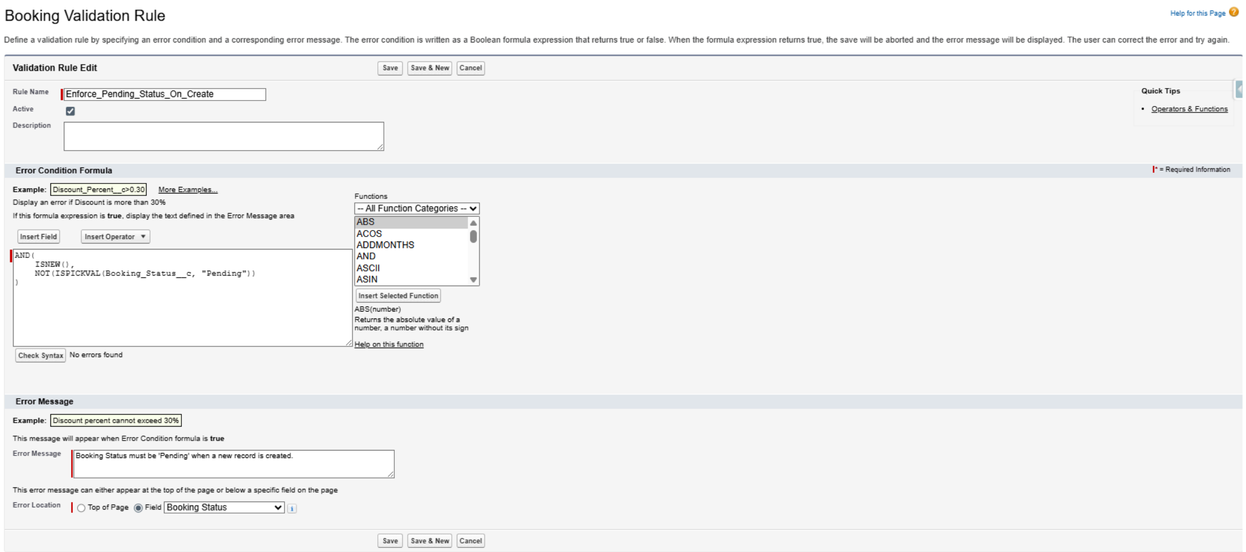




Validation Rules
2 Items, Sorted by Rule Name

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Email_is_mandatory	Email	You Must Enter the Email	✓	Andreas Luy, 8/23/2025, 10:45 AM
Language_Must_be_Selected	Languages Spoken	You Must Select Language	✓	Andreas Luy, 8/23/2025, 10:45 AM

Activity 4: To create a Validation rules to a Booking Object.



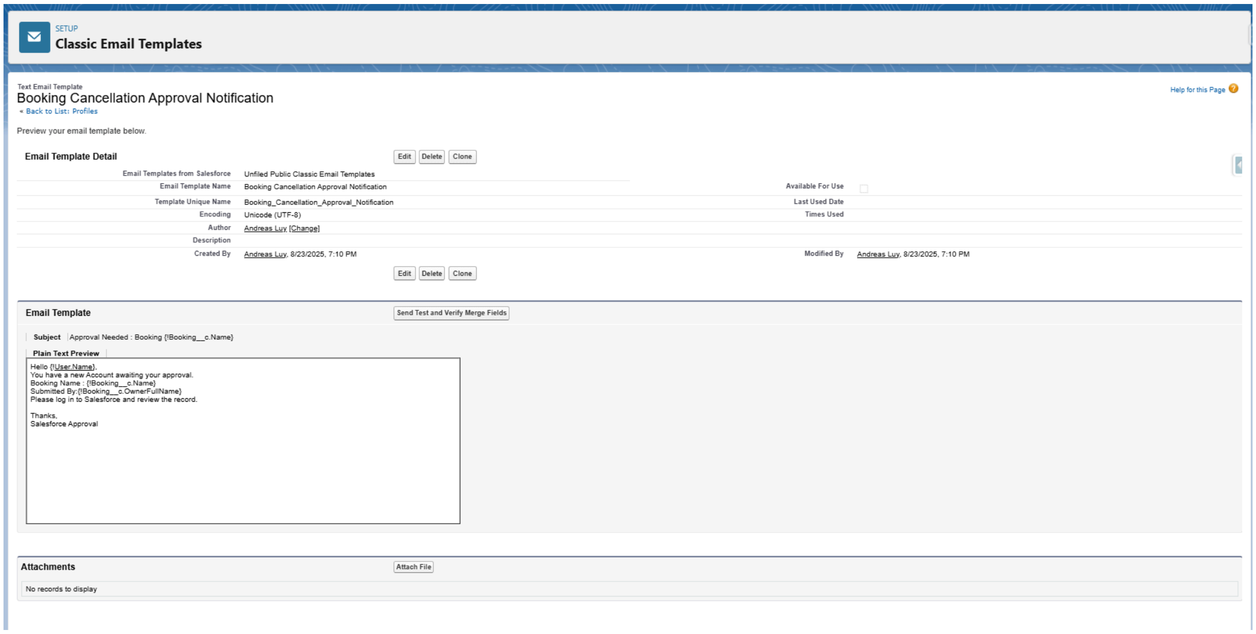
In Salesforce, validation rules can help maintain data integrity by preventing the Users from saving records that do not meet the criteria. Ensuring that the information saved is accurate, complete, and consistent.

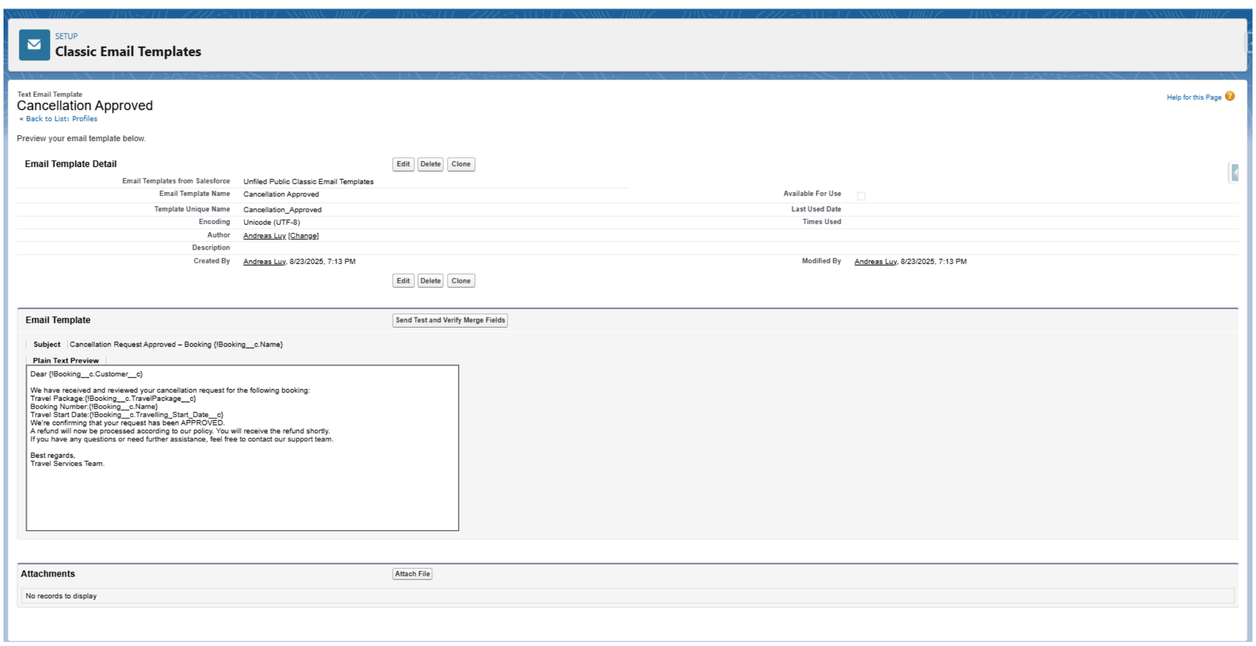
Upon creating the validation rules across different objects. It's important to note that we should always check if the API is correct as it can cause errors.

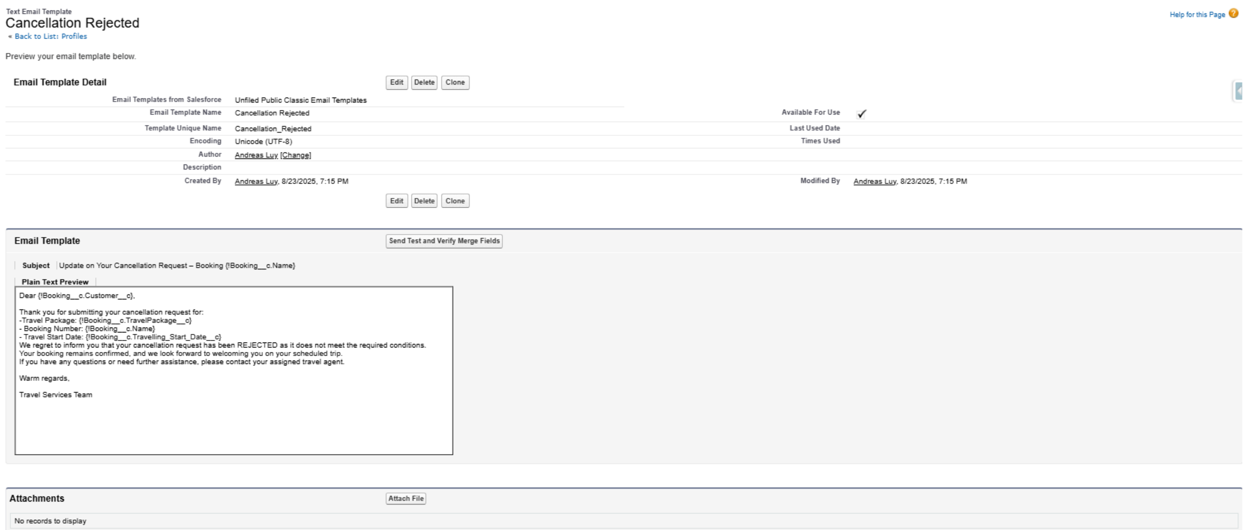
This activity helped me realize how important it is to check and check in order to avoid mistakes like incorrect values.

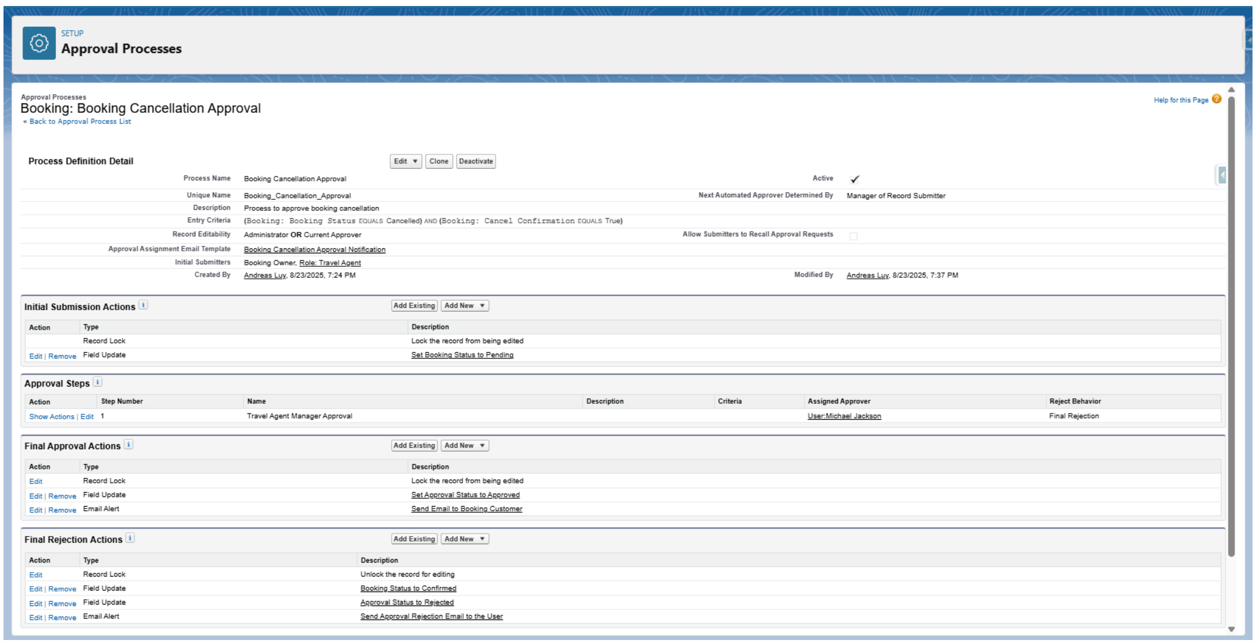
Milestone 7: Approval Process

Activity 1: Create an Approval Process for Booking Cancellation Approval







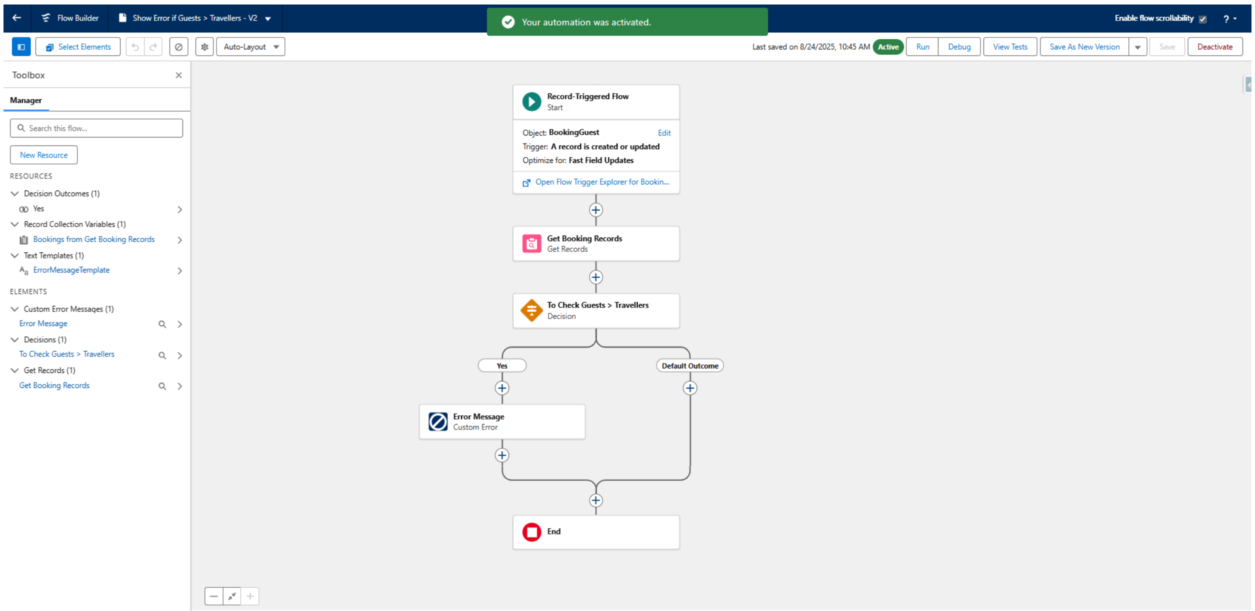


In Salesforce, the approval process can streamline the record by automatically routing the submissions to designed approvers. Ensuring that the critical actions like booking, cancellation, and discounts are going through proper authorization before being finalized.

This milestone helped me understand the difference between creating the approval processes instead of staying in manual follow-up processes.

Milestone 8: Flows

Activity 1

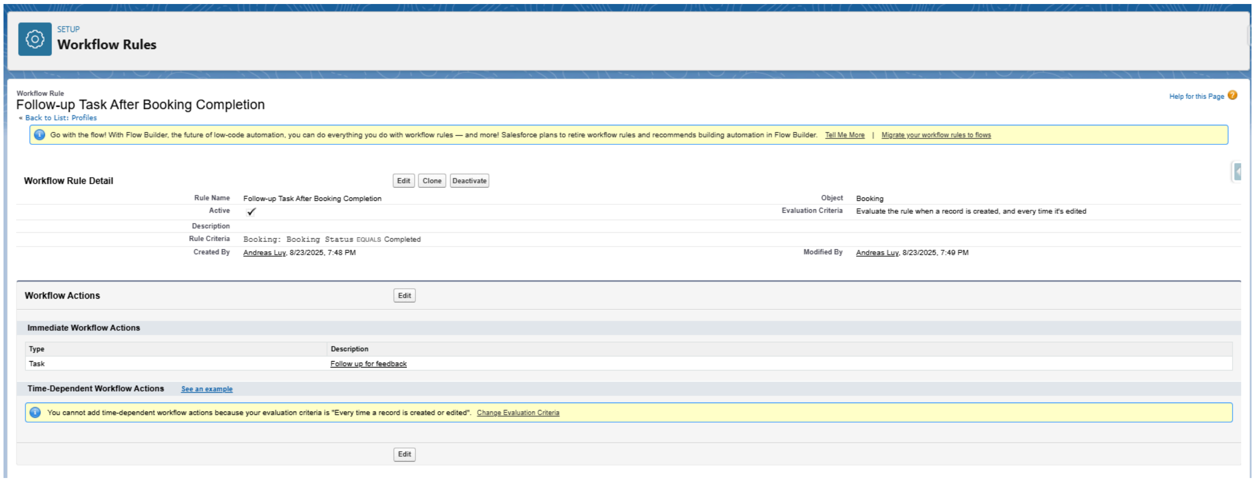


In Salesforce, flows will allow the automation of a business using a visual drag-and-drop interface. They can update records, get records, send notifications, and even guide Users through screens without writing code.

Creating the flows gave me a deeper appreciation of Salesforce's no code automation. I could automate repetitive tasks like record updates and notifications quickly! It actually became one of my favorite features to work on in Salesforce.

Milestone 9: Workflows

Activity 1



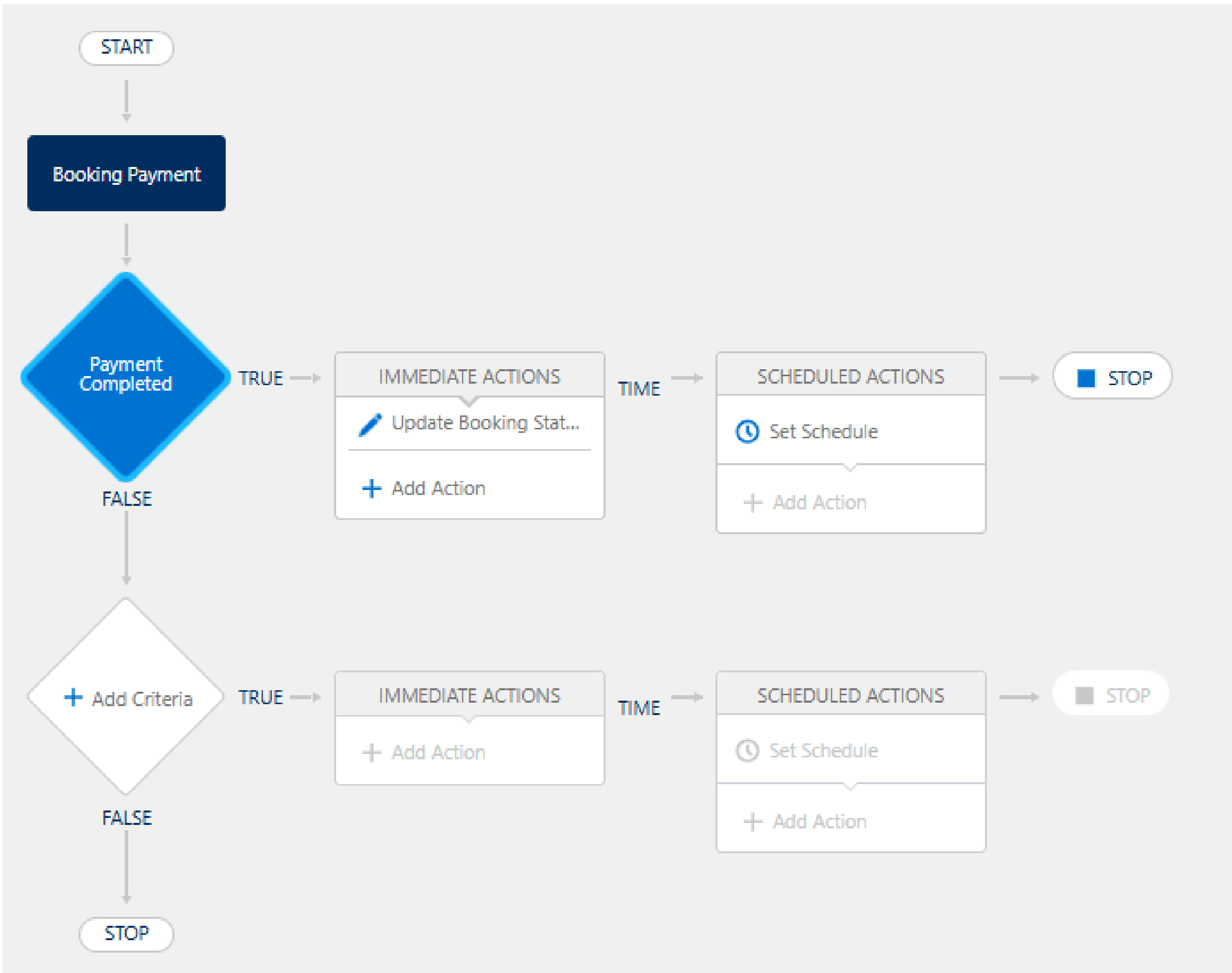
In Salesforce, the workflow automates repetitive tasks by triggering actions when specific conditions are met. They can update fields, send email alerts, and even create tasks which can help streamline the day-to-day processes.

Working with this milestone showed me how simple it is to make rules that could make a huge difference in efficiency. By automating the actions like field updates, I'll be able to reduce the manual effort and ensure that the important tasks are not going to be missed.

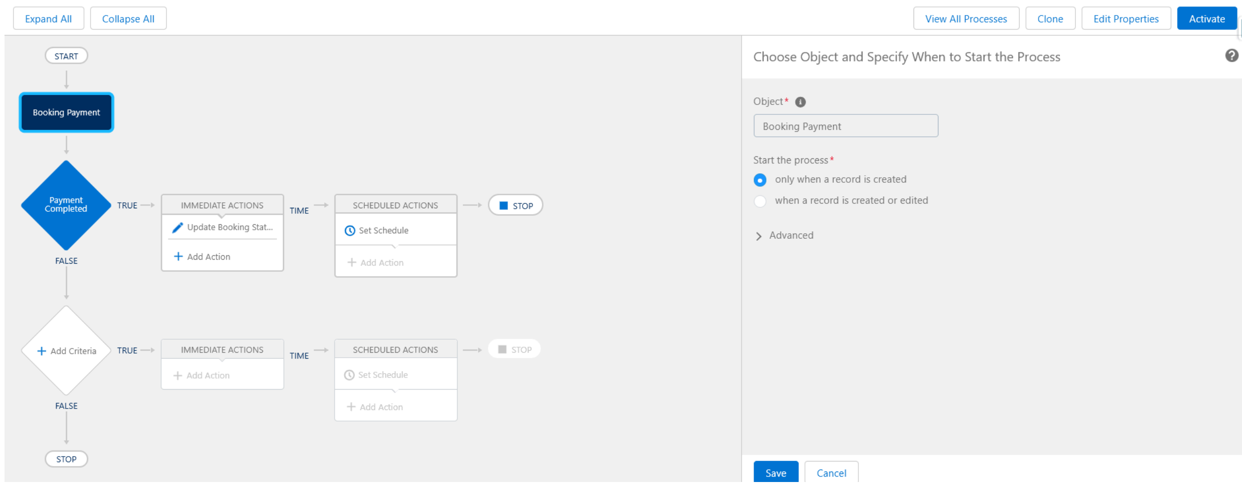
Milestone 10: Process Builder

Activity 1

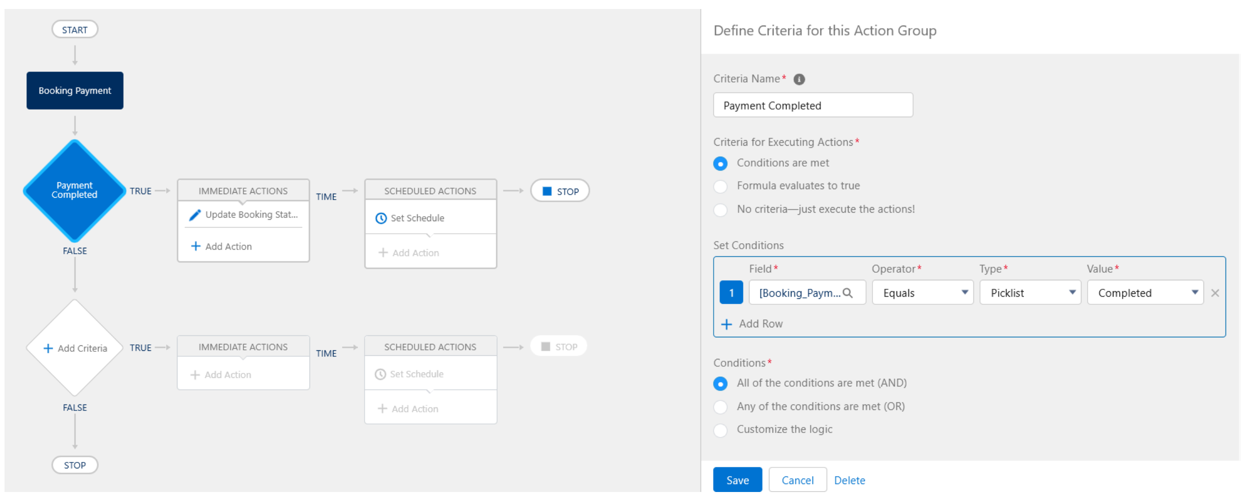
Entire Process



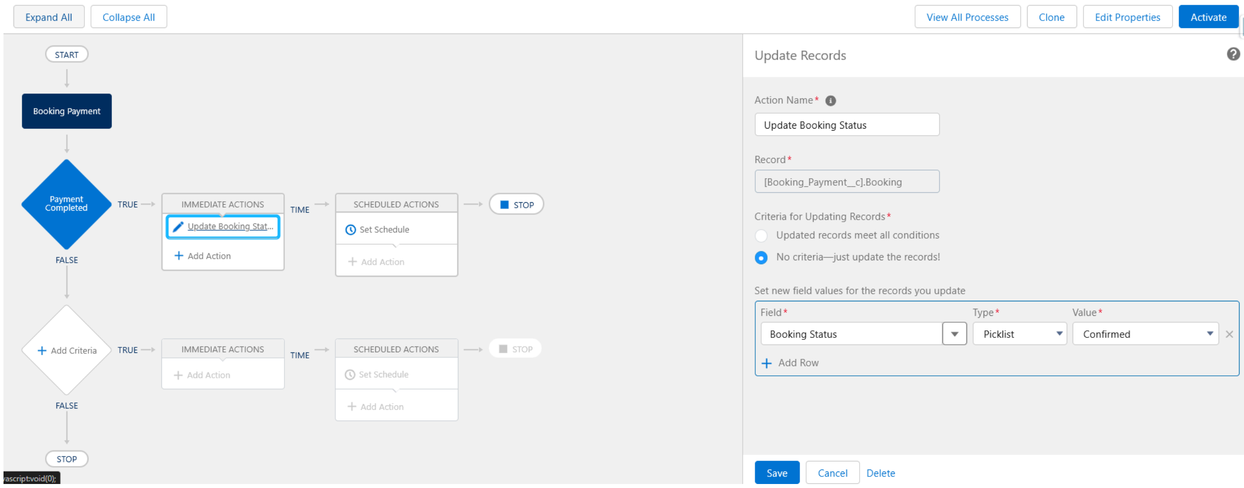
Booking Payment Object



Criteria



Immediate Actions



In Salesforce, the process builder is a point and click automation tool that uses the if/then logic to execute the actions within a single process. Building a process in Process Builder was confusing at first but it helped me gain an insight into how Salesforce automation evolved before flows. Even though Flows are my favorite, working with Process Builder was an important step in seeing how automation has evolved.

Milestone 11: Triggers

Activity 1 : Auto-create Booking Payment Record

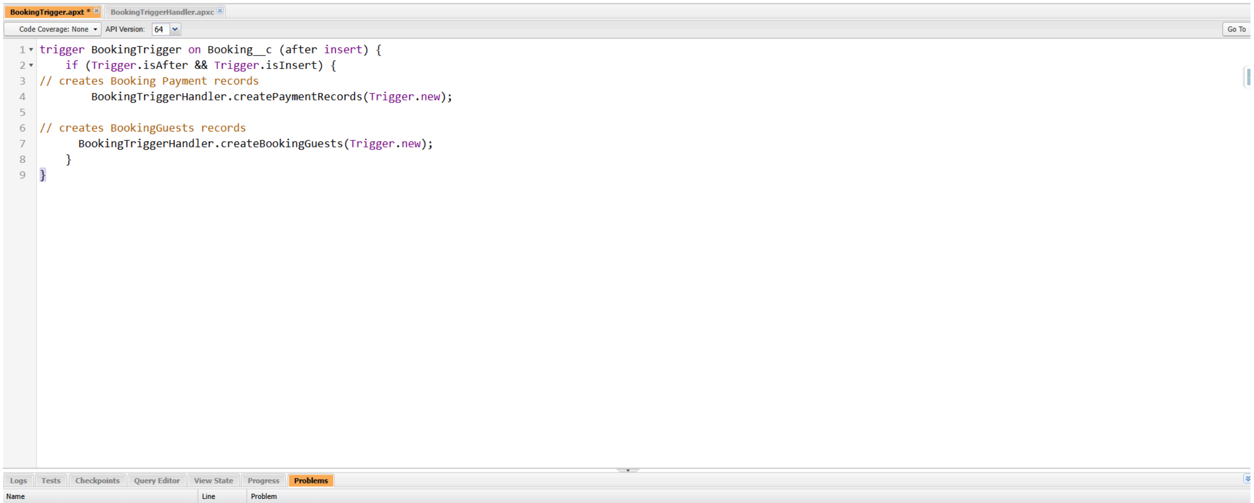
BookingTrigger.apxt | BookingTriggerHandler.apxc

```
trigger BookingTrigger on Booking__c (after insert) {
    if (Trigger.isAfter && Trigger.isInsert) {
// creates Booking Payment records
        BookingTriggerHandler.createPaymentRecords(Trigger.new);
    }
}
```

BookingTrigger.apxt | BookingTriggerHandler.apxc

```
public class BookingTriggerHandler {
    //Booking Payment Record Creation
    public static void createPaymentRecords(List<Booking__c> newBookings) {
        List<Booking_Payment__c> paymentsToInsert = new List<Booking_Payment__c>();
        for (Booking__c booking : newBookings) {
            Booking_Payment__c payment = new Booking_Payment__c();
            payment.Booking__c = booking.Id;
            payment.Payment_Status__c = 'Pending'; // Default status
            paymentsToInsert.add(payment);
        }
        if (!paymentsToInsert.isEmpty()) {
            insert paymentsToInsert;
        }
    }
}
```

Activity 2 : Automate Guest Records Creation Based on Traveller Count



In Salesforce, the Apex triggers are pieces of code that run automatically before or after the database events such as insert, update, or delete. With triggers, the administrators and developers can gain a deeper control over how data is going to behave and interact across objects. Implementing the triggers for the Booking object gave me hands-on experience. It felt like a major step forward because it showed me how to go beyond clicks and use code to design accurate and customized workflows. It’s more technical compared to the earlier milestones, but feels very rewarding when the state in the logs is Success!

Milestone 12: Asynchronous Apex

Activity 1 : Future Asynchronous Apex

BookingConfirmationEmailer.apxc | BookingTrigger.apxt

```
public class BookingConfirmationEmailer {

    @future(callout=false)

    public static void sendBookingConfirmation(Set<Id> bookingIds) {
        List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();
        List<Booking__c> bookings = [SELECT Id, Name, Customer_Email__c,  Total_Billing_Amount__c
                                     FROM Booking__c
                                     WHERE Id IN :bookingIds];

        for (Booking__c booking : bookings) {
            if (String.isNotBlank(booking.Customer_Email__c)) {
                Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
                mail.setToAddresses(new String[] { booking.Customer_Email__c });
                mail.setSubject('Booking Confirmed: ' + booking.Name);
                mail.setPlainTextBody(
                    'Dear Customer,' + '\n\n' +
                    'Your booking has been confirmed. Please find the details below:\n' +
                    'Booking ID: ' + booking.Name + '\n' +
                    'Total Bill Amount Paid: $' + booking.Total_Billing_Amount__c + '\n\n' +
                    'Thank you for booking with us!'
                );
                emails.add(mail);
            }
        }
        if (!emails.isEmpty()) {
            Messaging.sendEmail(emails);
        }
    }
}
```

Logs | Tests | Checkpoints | Query Editor | View State | Progress | Problems

Name	Line	Problem

BookingConfirmationEmailer.apxc | BookingTrigger.apxt

```
trigger BookingTrigger on Booking__c (after insert, after update) {

    if (Trigger.isAfter && Trigger.isInsert) {
        // creates Booking Payment records
        BookingTriggerHandler.createPaymentRecords(Trigger.new);

        // creates BookingGuests records
        BookingTriggerHandler.createBookingGuests(Trigger.new);
    }

    else  if (Trigger.isAfter && Trigger.isUpdate) {
        Set<Id> bookingIdsToSend = new Set<Id>();

        for (Booking__c booking : Trigger.new) {
            Booking__c oldBooking = Trigger.oldMap.get(booking.Id);

            // Check for status change to "Confirmed"
            if (booking.Booking_Status__c == 'Confirmed' && oldBooking.Booking_Status__c != 'Confirmed') {
                bookingIdsToSend.add(booking.Id);
            }
        }

         if (!bookingIdsToSend.isEmpty()) {
            // Call the Future Method
            BookingConfirmationEmailer.sendBookingConfirmation(bookingIdsToSend);
        }
    }
}
```

Logs | Tests | Checkpoints | Query Editor | View State | Progress | Problems

Name	Line	Problem

Activity 2 : Queueable and Schedulable Asynchronous Apex

BookingConfirmationEmailer.apxc | BookingTrigger.apxt | BookingReminderQueueable.apxc | BookingReminderScheduler.apxc

Code Coverage: None | API Version: 64 | Go To

```
public class BookingReminderQueueable implements Queueable {
    List<Booking__c> bookingsToRemind;

    public BookingReminderQueueable(List<Booking__c> bookings) {
        this.bookingsToRemind = bookings;
    }

    public void execute(QueueableContext context) {
        List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();

        for (Booking__c booking : bookingsToRemind) {
            if (String.isNotBlank(booking.Customer_Email__c)) {
                Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
                mail.setToAddresses(new String[] { booking.Customer_Email__c });
                mail.setSubject('Reminder: Your Tour Starts Soon!');
                mail.setPlainTextBody(
                    'Hello,\n\nThis is a friendly reminder that your tour is starting on ' +
                    booking.Travelling_Start_Date__c.format() +
                    '. Please make necessary arrangements.\n\nThank you for choosing us!'
                );
                emails.add(mail);
            }
        }

        if (!emails.isEmpty()) {
            Messaging.sendEmail(emails);
        }
    }
}
```

Logs | Tests | Checkpoints | Query Editor | View State | Progress | Problems

Name	Line	Problem

BookingConfirmationEmailer.apxc | BookingTrigger.apxt | BookingReminderQueueable.apxc | BookingReminderScheduler.apxc

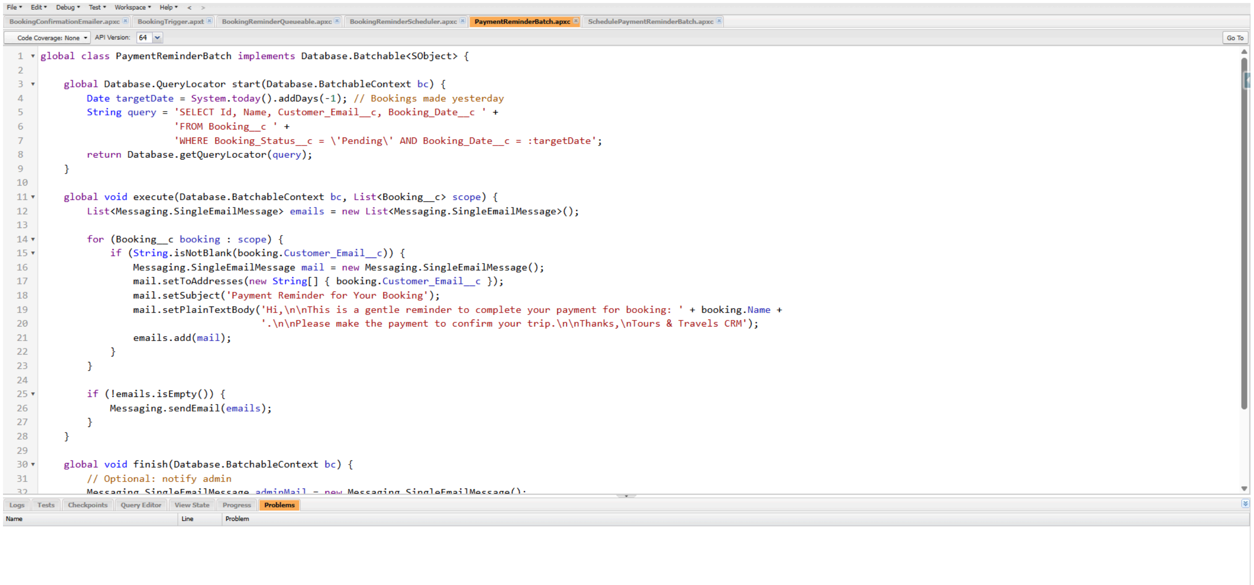
Code Coverage: None | API Version: 64 | Go To

```
public class BookingReminderScheduler implements Schedulable {
    public void execute(SchedulableContext context) {
        Date reminderDate = Date.today().addDays(3);
        List<Booking__c> upcomingBookings = [
            SELECT Id, Travelling_Start_Date__c, Customer_Email__c
            FROM Booking__c
            WHERE Travelling_Start_Date__c = :reminderDate
            AND Booking_Status__c = 'Confirmed'
        ];
        if (!upcomingBookings.isEmpty()) {
            System.enqueueJob(new BookingReminderQueueable(upcomingBookings));
        }
    }
}
```

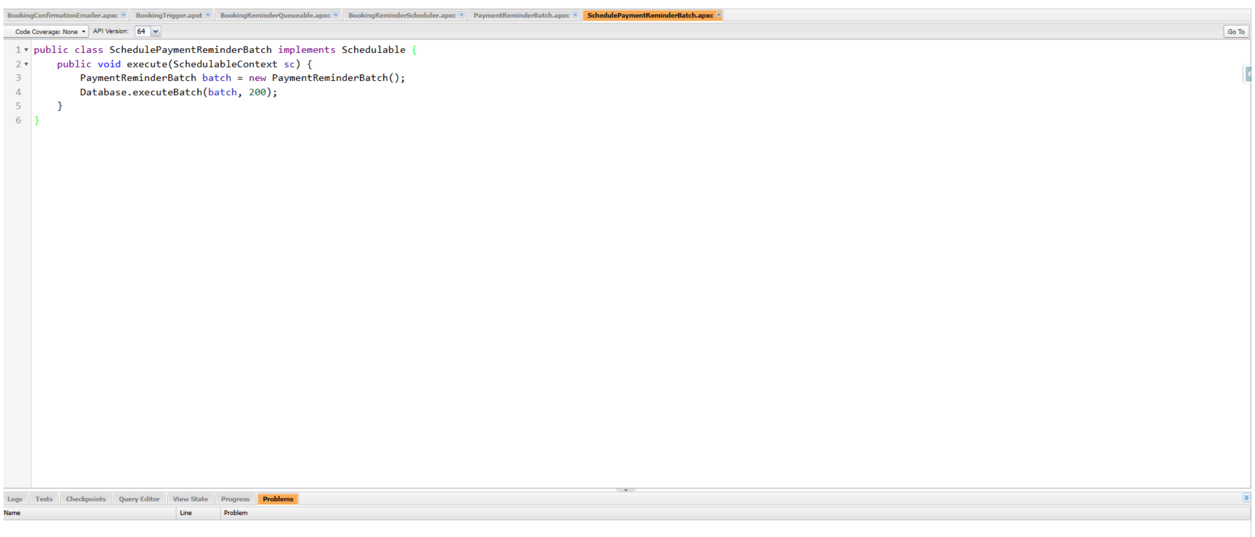
Logs | Tests | Checkpoints | Query Editor | View State | Progress | Problems

Name	Line	Problem

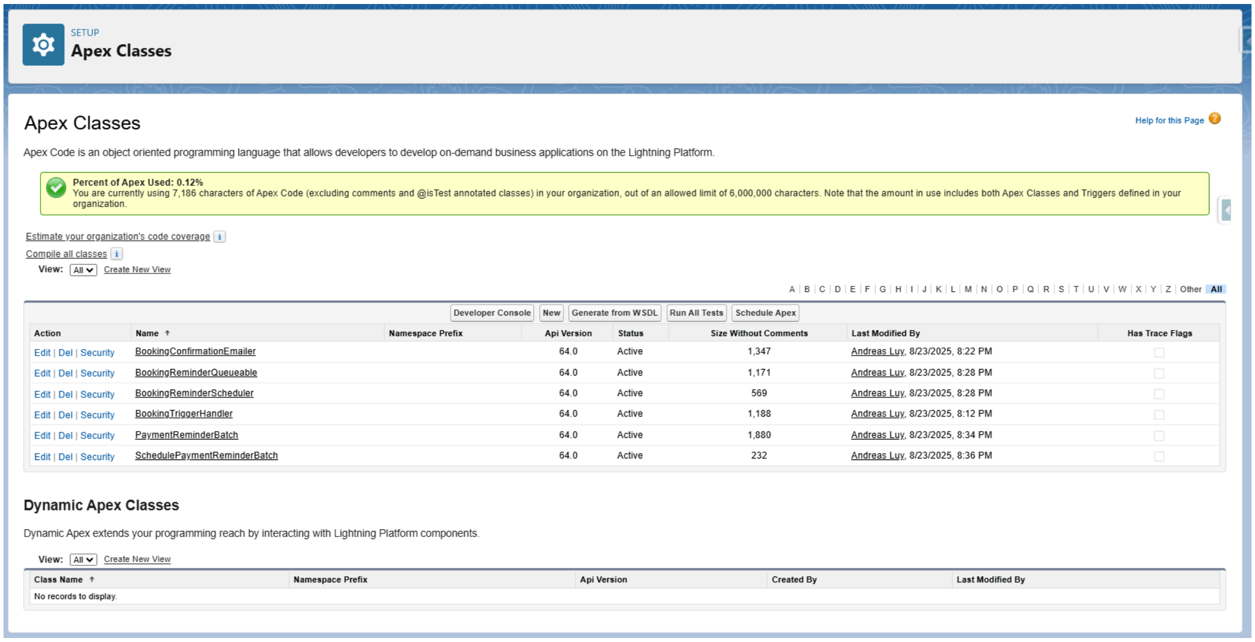
Activity 3 : Batchable and Schedulable Asynchronous Apex



```
global class PaymentReminderBatch implements Database.Batchable<SObject> {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        Date targetDate = System.today().addDays(-1); // Bookings made yesterday
        String query = 'SELECT Id, Name, Customer_Email__c, Booking_Date__c ' +
                       'FROM Booking__c ' +
                       'WHERE Booking_Status__c = \'Pending\' AND Booking_Date__c = :targetDate';
        return Database.getQueryLocator(query);
    }

    global void execute(Database.BatchableContext bc, List<Booking__c> scope) {
        List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();

        for (Booking__c booking : scope) {
            if (String.isNotBlank(booking.Customer_Email__c)) {
                Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
                mail.setToAddresses(new String[] { booking.Customer_Email__c });
                mail.setSubject('Payment Reminder for Your Booking');
                mail.setPlainTextBody('Hi,\n\nThis is a gentle reminder to complete your payment for booking: ' + booking.Name +
                                      '.\n\nPlease make the payment to confirm your trip.\n\nThanks,\nTours & Travels CRM');
                emails.add(mail);
            }
        }

        if (!emails.isEmpty()) {
            Messaging.sendEmail(emails);
        }
    }

    global void finish(Database.BatchableContext bc) {
        // Optional: notify admin
        Messaging.SingleEmailMessage adminMail = new Messaging.SingleEmailMessage();
```



```
public class SchedulePaymentReminderBatch implements Schedulable {
    public void execute(SchedulableContext sc) {
        PaymentReminderBatch batch = new PaymentReminderBatch();
        Database.executeBatch(batch, 200);
    }
}
```



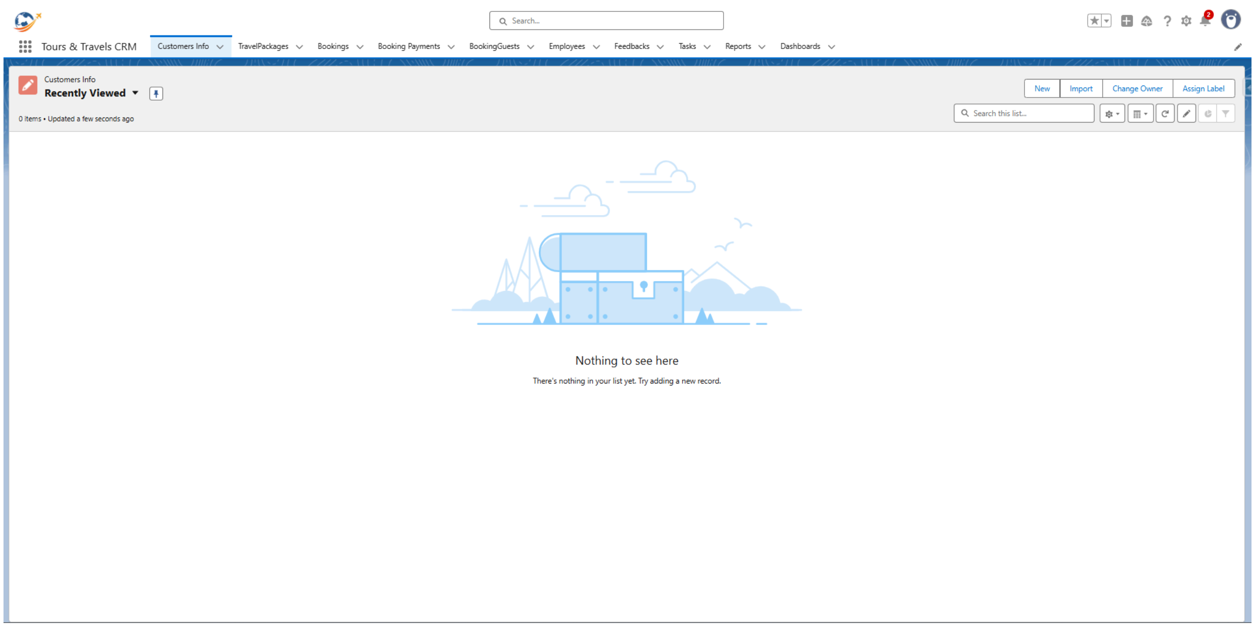
In Salesforce, Asynchronous Apex is designed to handle background tasks that are long-running or resource-intensive.

This milestone was one of the most technical yet rewarding parts of my journey. I implemented asynchronous processes such as automatically sending booking confirmation emails, scheduling tour reminders, and running daily payment reminder batches. It showed me a different way Salesforce can handle complex and large-scale operations efficiently.

Phase 3: UI/UX Development & Customization

Milestone 13: The Lightning App

Activity 1: Create a Lightning App for " Tours & Travels CRM

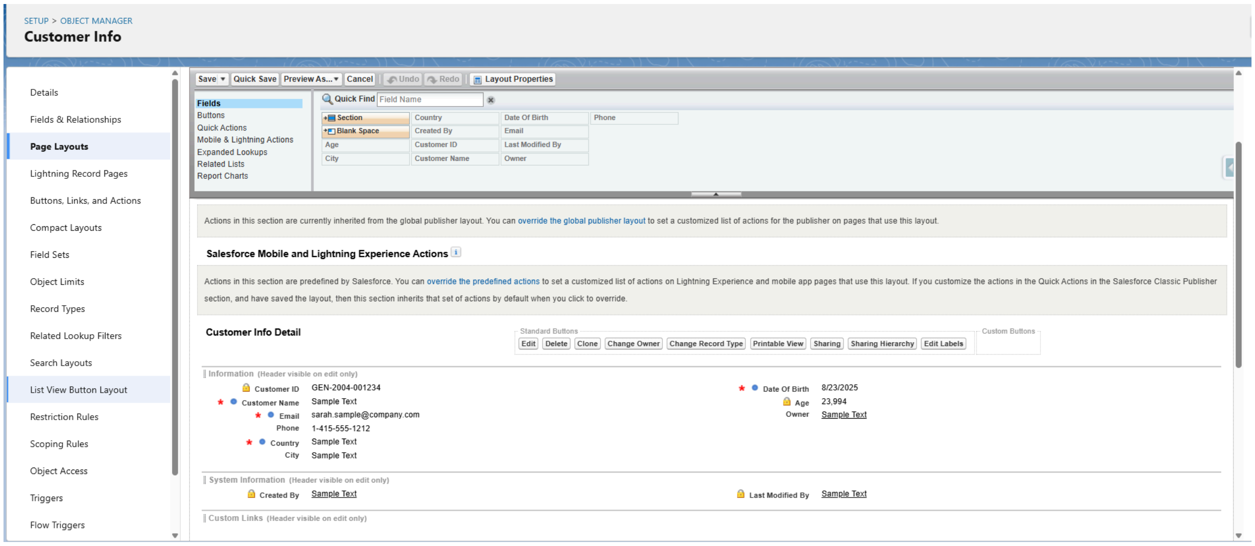


In Salesforce, a Lightning App is a customizable collection of objects, tabs, and utilities. It provides a modern interface tailored to specific roles.

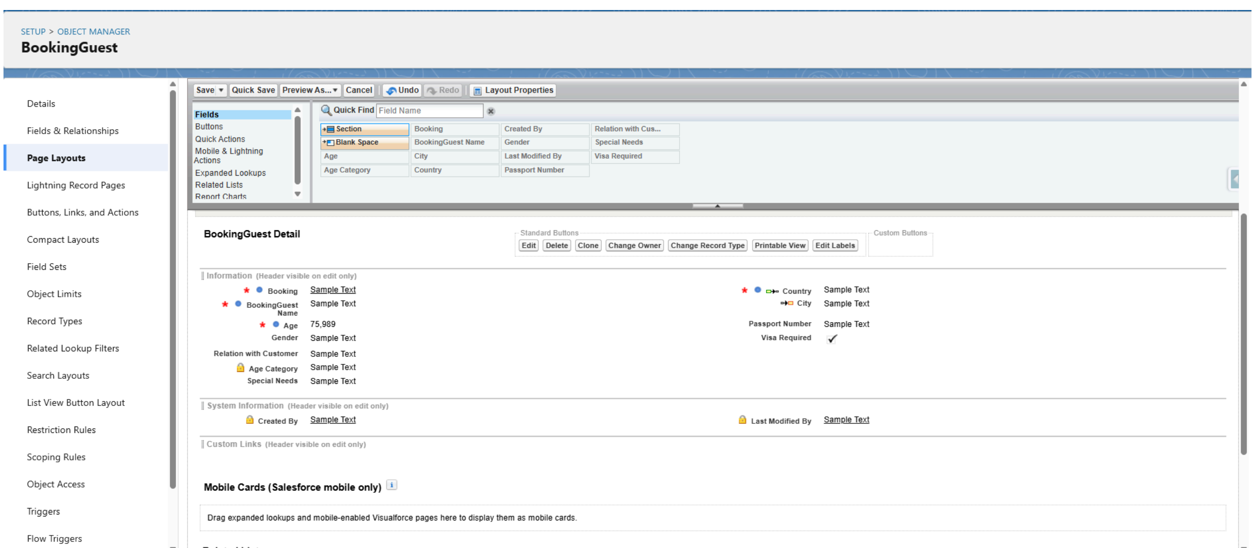
Creating the Tours & Travels CRM felt like bringing all the earlier requirements together! By including the custom objects I have created earlier together in a one unified space, it felt like I was able to design the app by myself. This is like the front door of the project as it gave the application a polished and professional look.

Milestone 14: Editing of Page Layouts

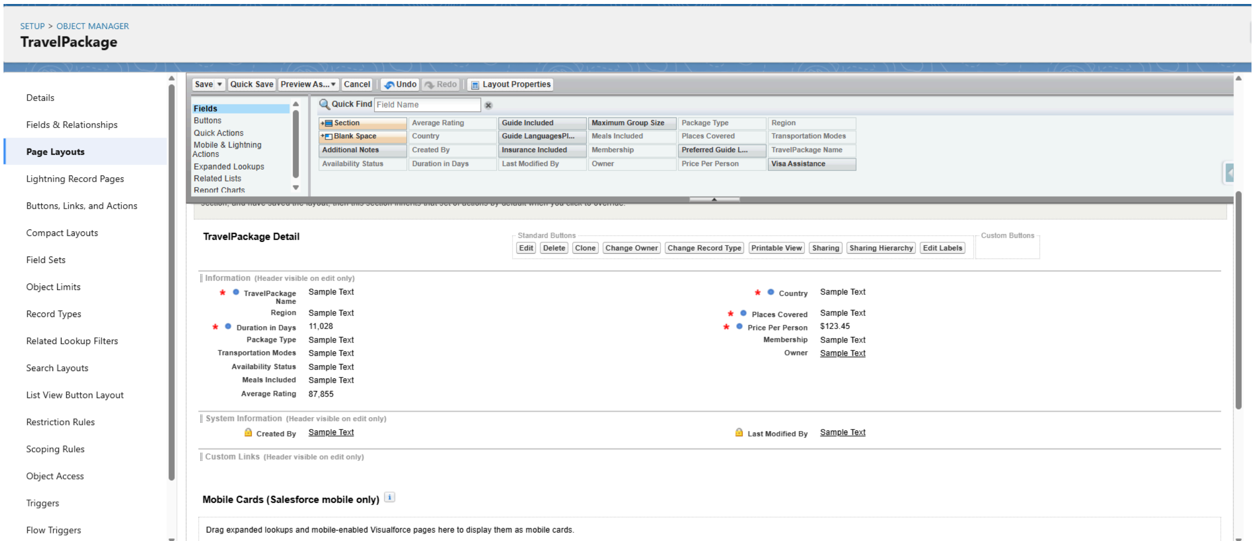
Activity 1: To edit a Page Layout in Customer Info Object



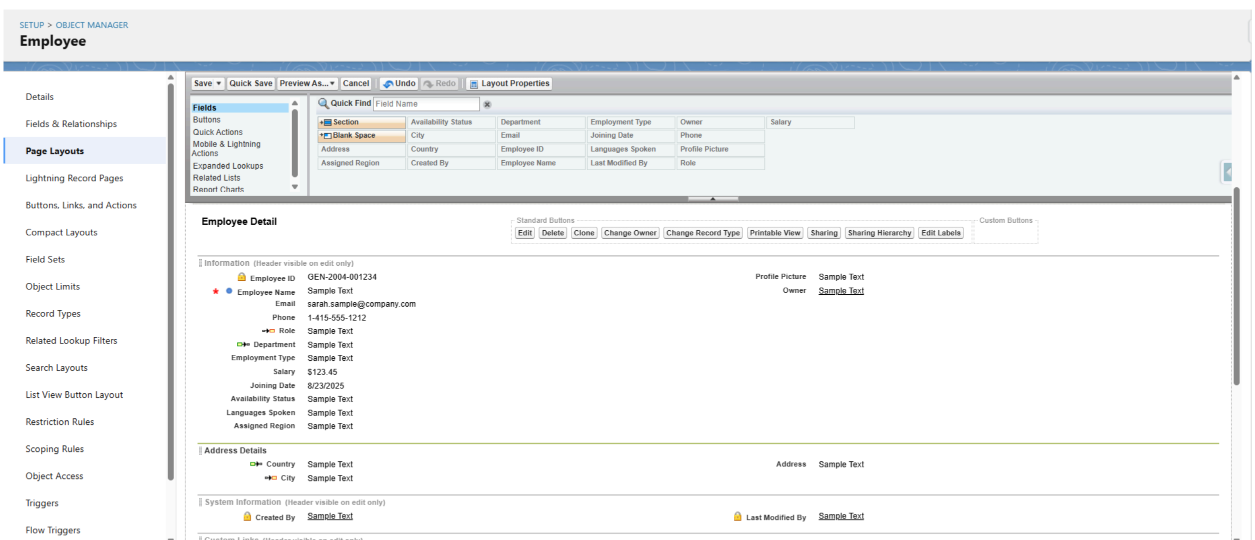
Activity 2: To edit a Page Layout in BookingGuest Object



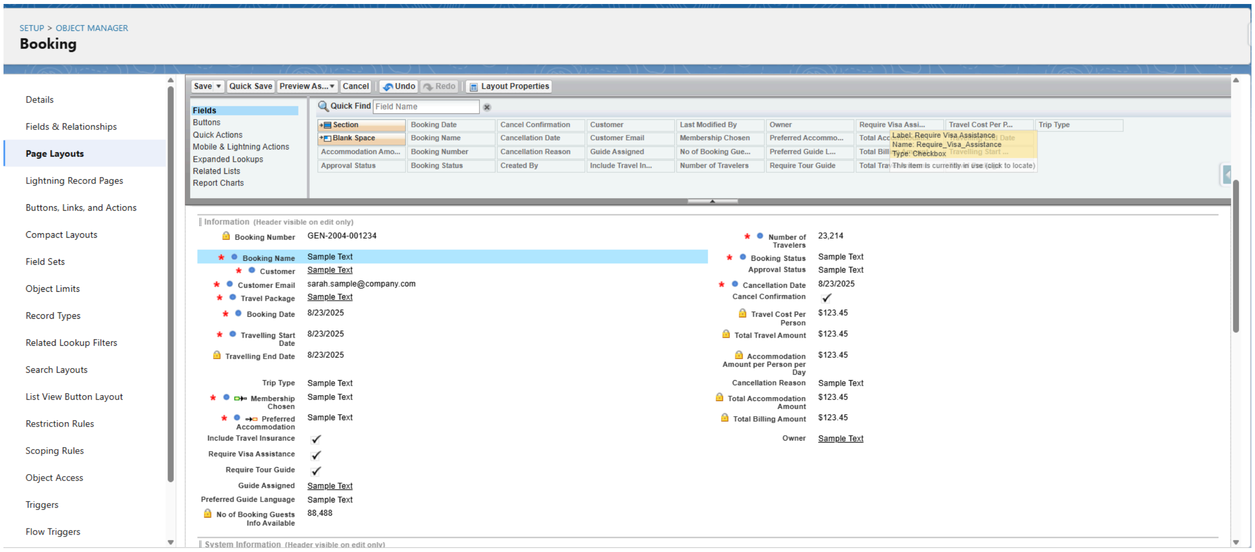
Activity 3: To edit a Page Layout in TravelPackage Object



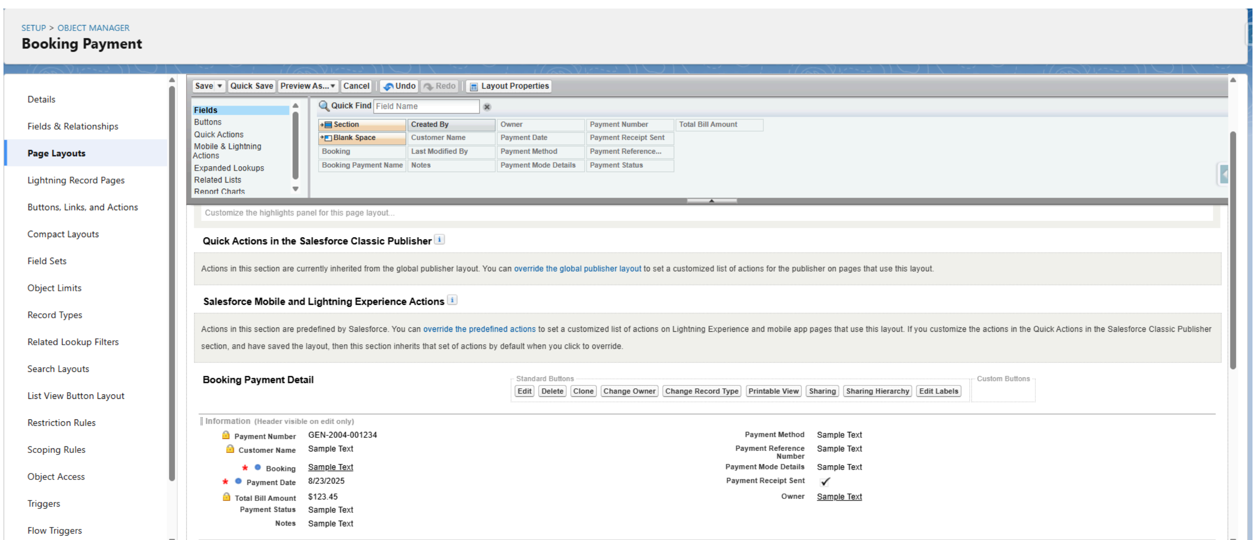
Activity 4: To edit a Page Layout in Employee Object



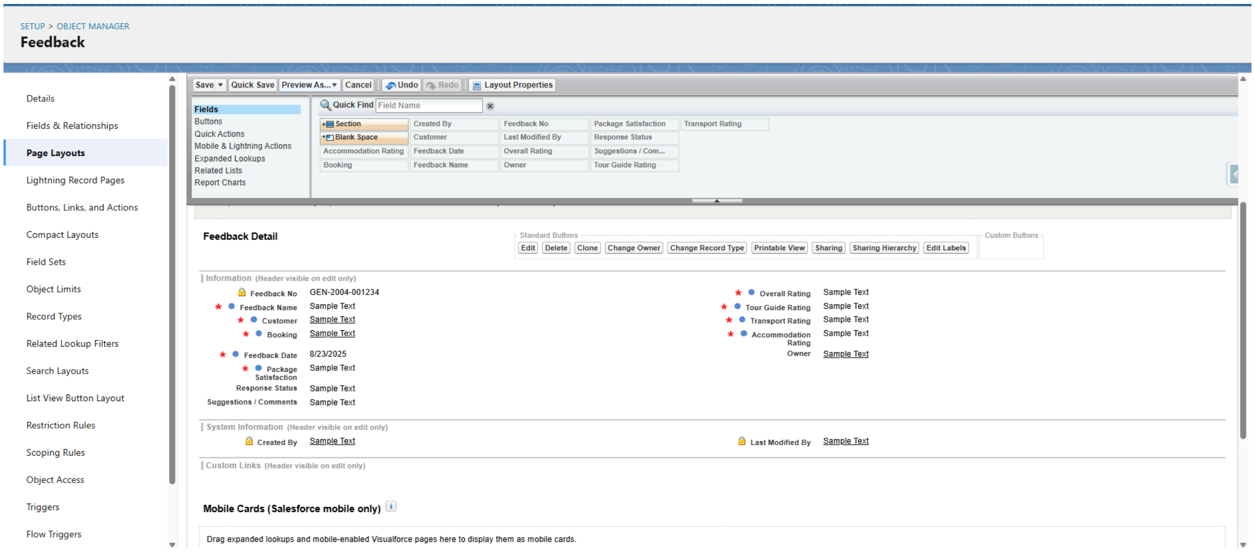
Activity 5: To edit a Page Layout in Booking Object



Activity 6: To edit a Page Layout in Booking Payments Object



Activity 7: To edit a Page Layout in Feedbacks Object

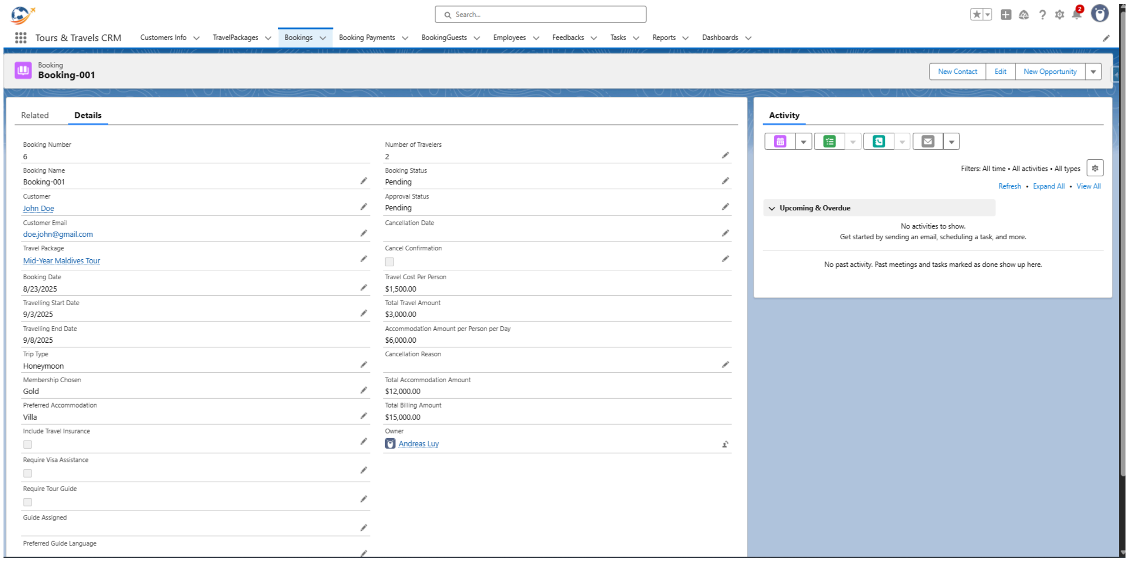


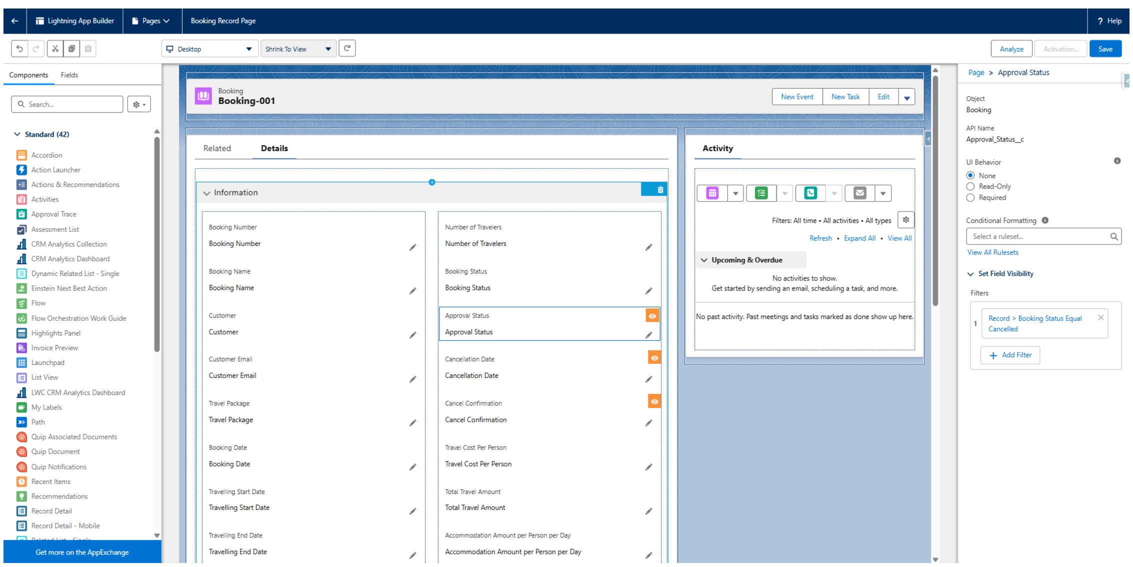
In Salesforce, Page layout can define how the records will appear to the Users by controlling the placement and visibility of fields, section, and other elements.

Upon editing the page layout across different objects like Customer Info, Bookings, and others helped me customize the CRM for a cleaner and Users friendly experience. This milestone surely helped make the application look more professional and intuitive.

Milestone 15: Dynamic Forms

Activity 1: To create a Dynamic Form in Booking Object

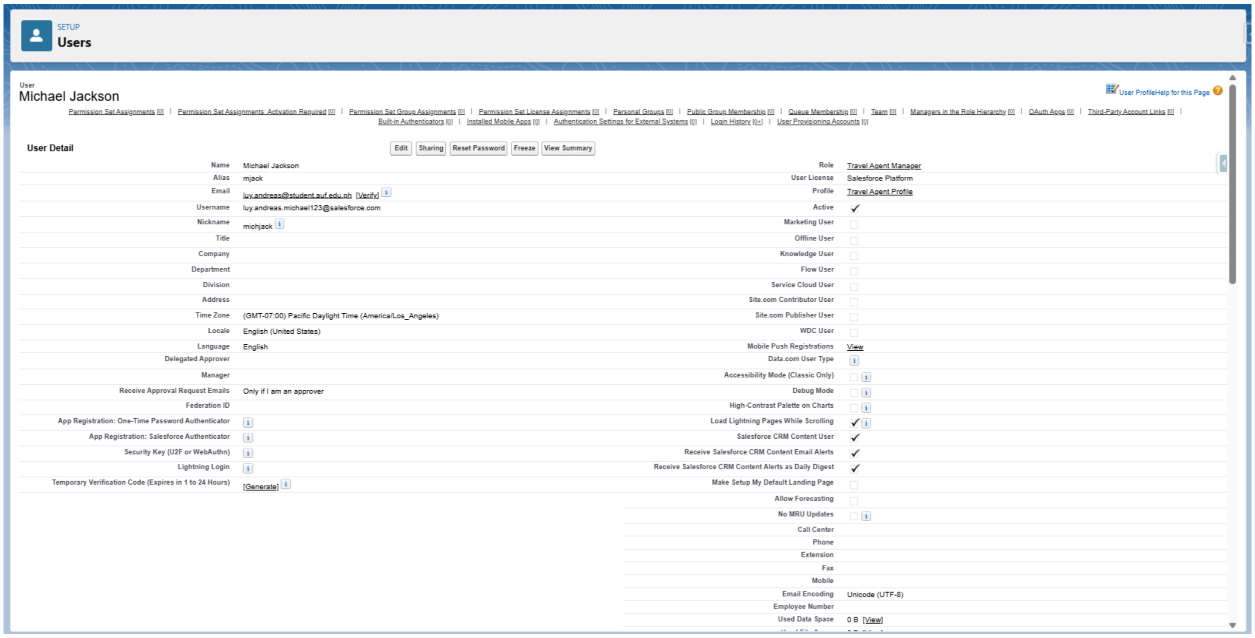




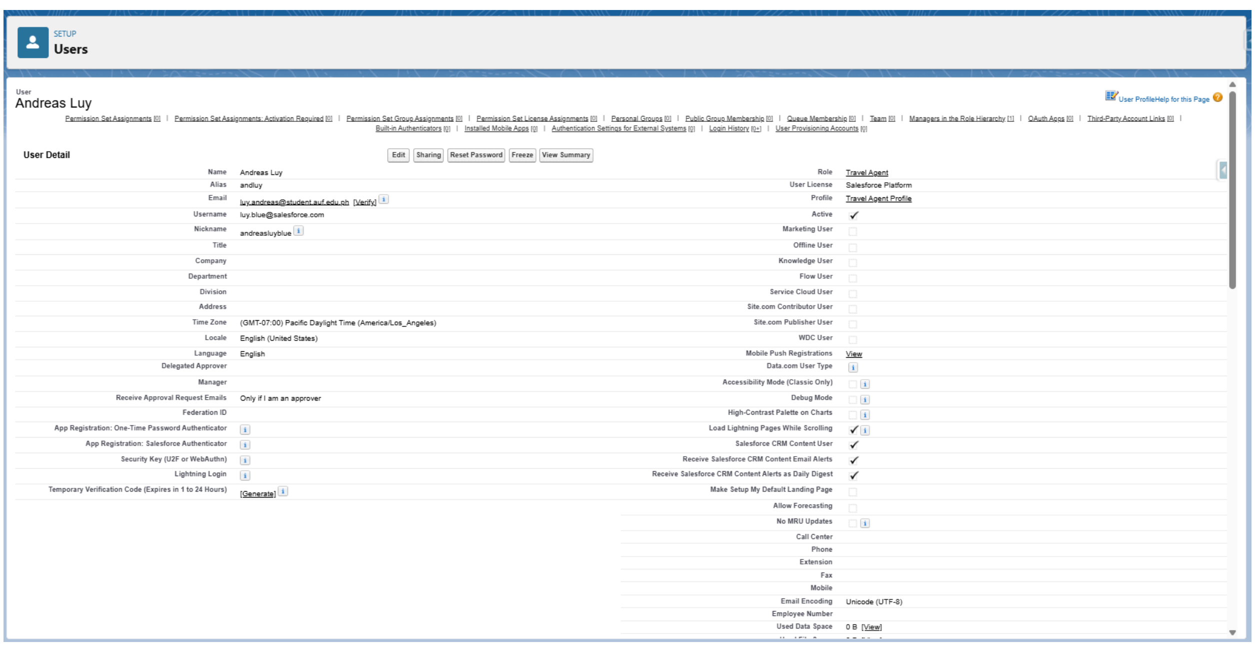
In Salesforce, Dynamic forms will let us configure record pages at the field level within the Lightning App Builder. Unlike the traditional page layouts, it can allow us the conditional visibility of fields and sections based on profile, record values, or logic. Implementing a dynamic form showed me how powerful it can be. It also helped me reduce the clutter on the record page and made the interface more intuitive.

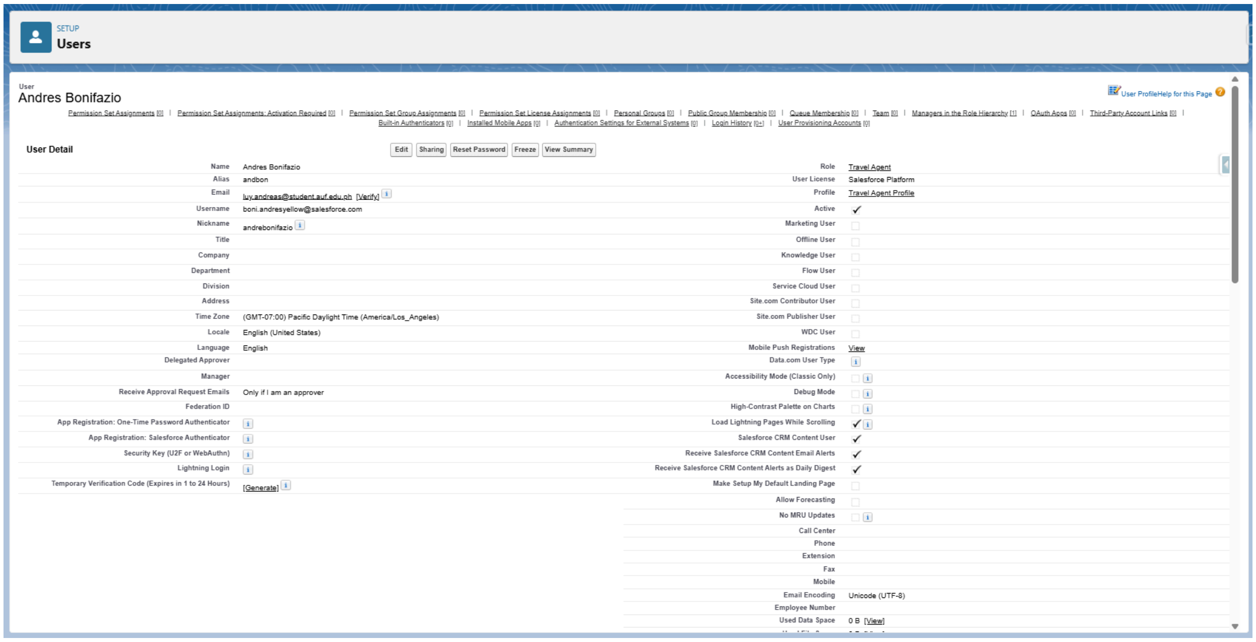
Milestone 16: Users

Activity 1: Create User

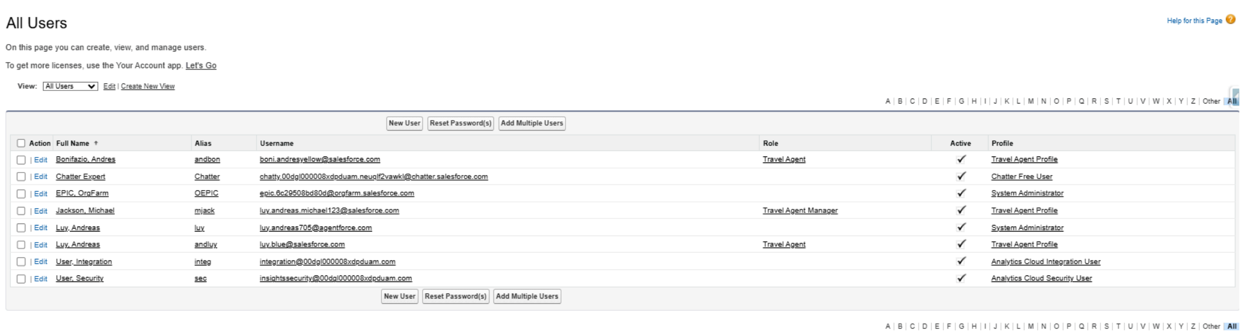


Activity 2: creating other Users





All Users

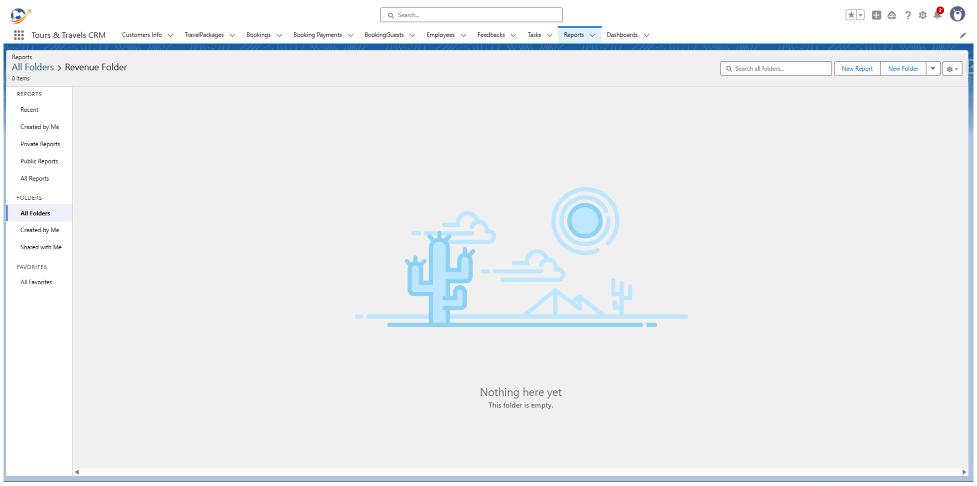


In Salesforce, Users represent the people who can log in and interact with the system. Each Users will be assigned a role, profile, and a license, which will together define their privilege or level of access and permissions.

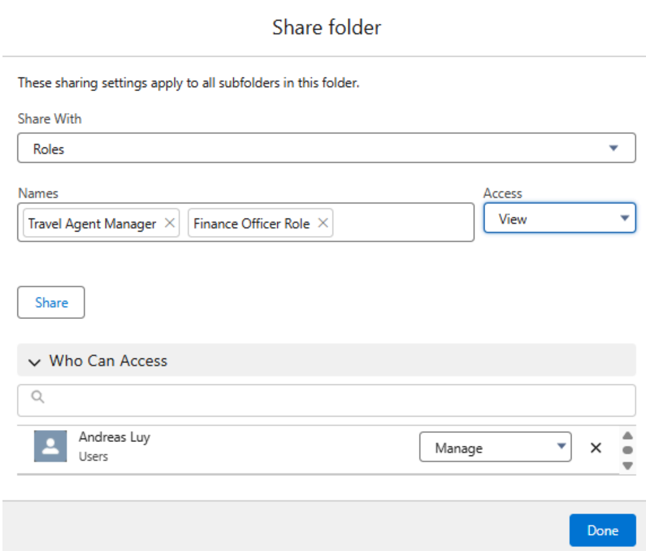
Creating the Users such as travel agents and managers gave the project a more realistic feel. This milestone highlighted how Salesforce can balance flexibility with security, ensuring that the data is available to those who need it.

Milestone 17: Reports

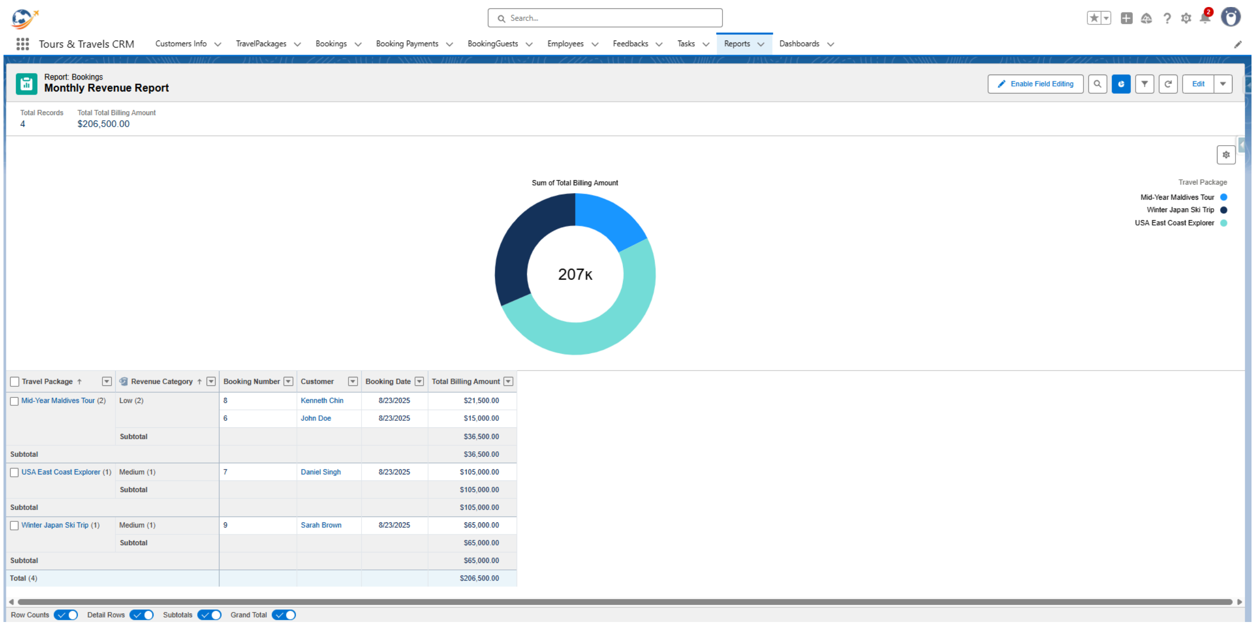
Activity 1: Create a Report folder



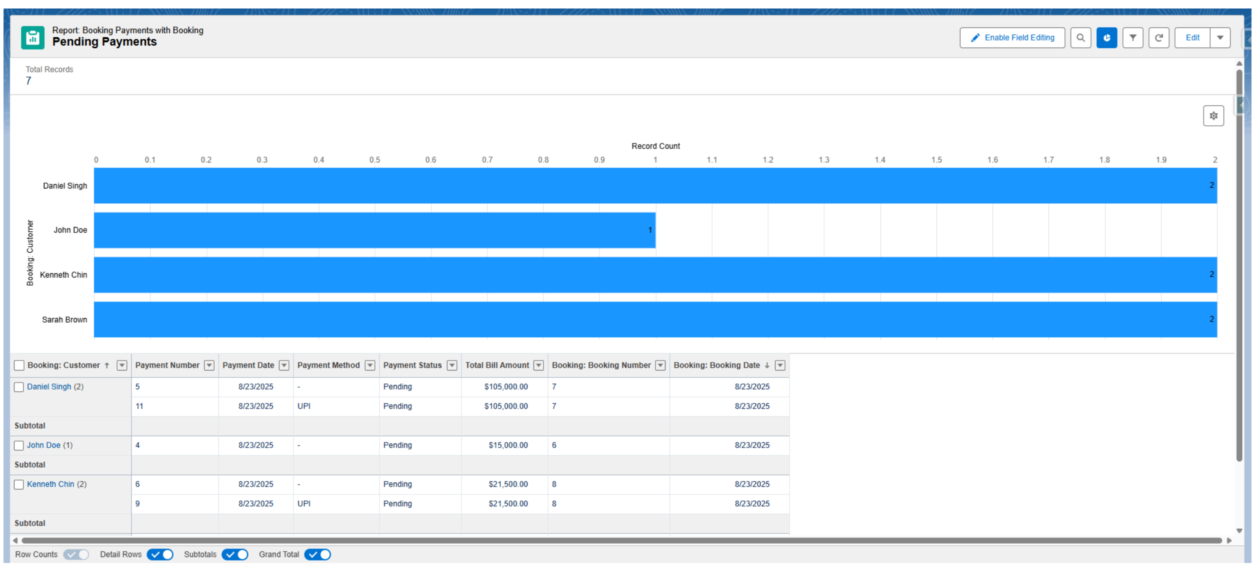
Activity 2: Sharing a Report folder

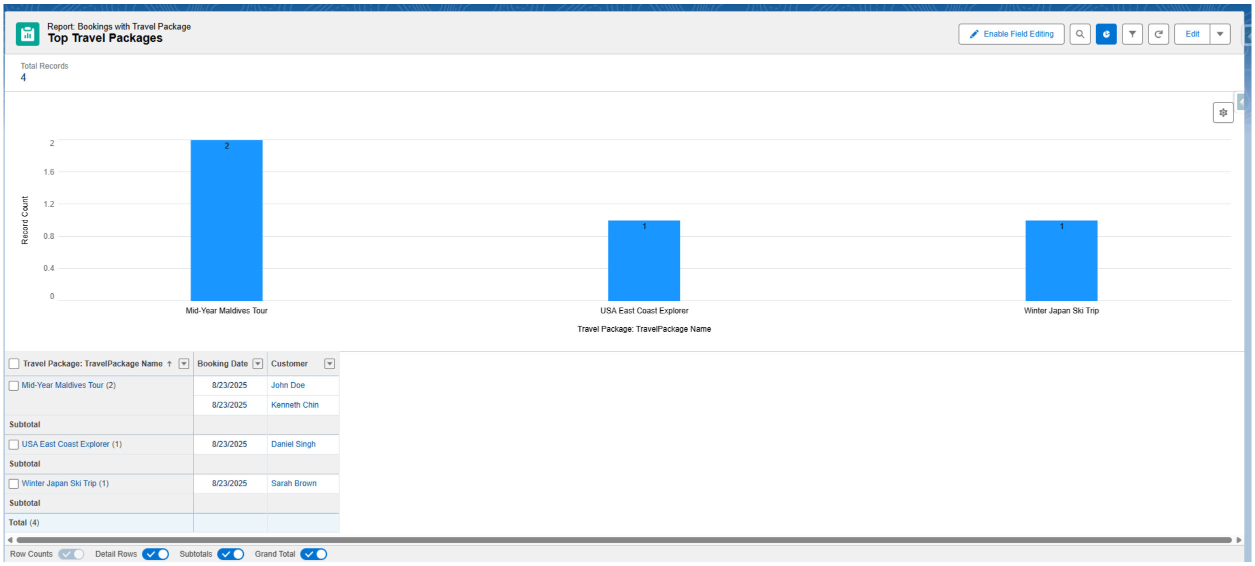


Activity 3: Create Report

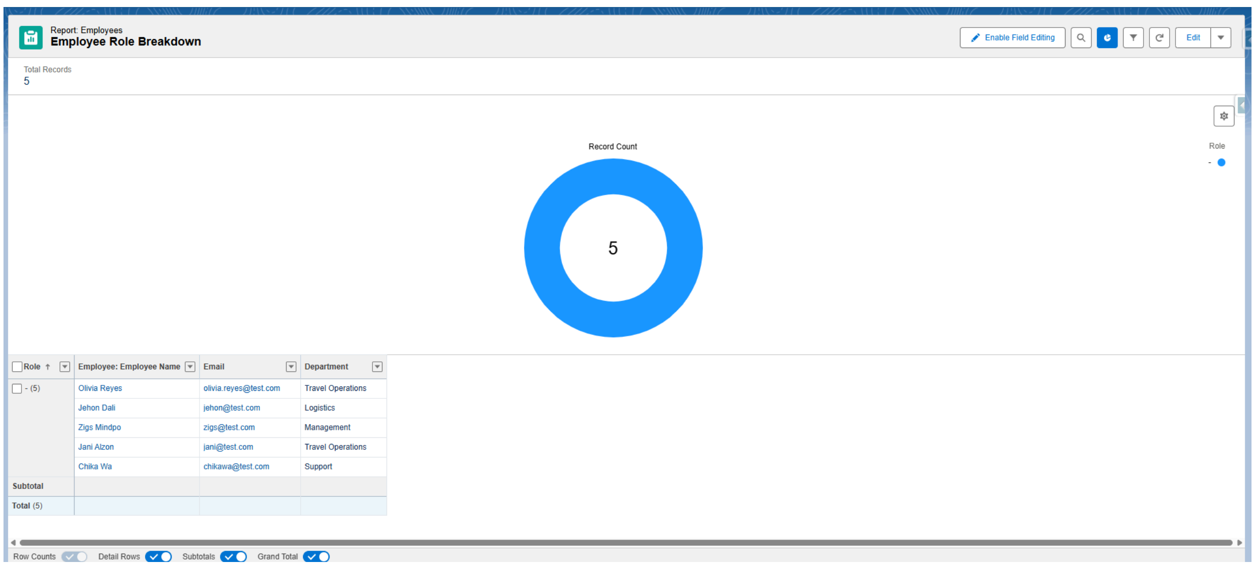


Activity 4: Create Other Reports



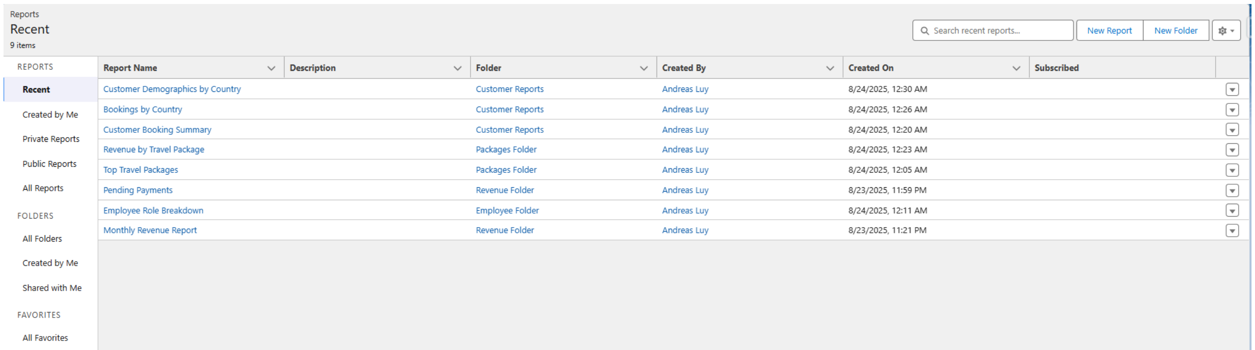


Report: Bookings with Travel Package
Top Travel Packages
Enable Field Editing
Edit
Total Records
4
Record Count
2
1.6
1.2
0.8
0.4
0
2
1
1
Mid-Year Maldives Tour
USA East Coast Explorer
Winter Japan Ski Trip
Travel Package: TravelPackage Name
Travel Package: TravelPackage Name
Booking Date
Customer
Mid-Year Maldives Tour (2)
8/23/2025
John Doe
8/23/2025
Kenneth Chin
Subtotal
USA East Coast Explorer (1)
8/23/2025
Daniel Singh
Subtotal
Winter Japan Ski Trip (1)
8/23/2025
Sarah Brown
Subtotal
Total (4)
Row Counts
Detail Rows
Subtotals
Grand Total



Report: Employees
Employee Role Breakdown
Enable Field Editing
Edit
Total Records
5
Record Count
5
Role
Role
Employee: Employee Name
Email
Department
- (5)
Olivia Reyes
olivia.reyes@test.com
Travel Operations
Jehon Dali
jehon@test.com
Logistics
Zigs Mindpo
zigs@test.com
Management
Jani Alzon
jani@test.com
Travel Operations
Chika Wa
chikawa@test.com
Support
Subtotal
Total (5)
Row Counts
Detail Rows
Subtotals
Grand Total

Reports

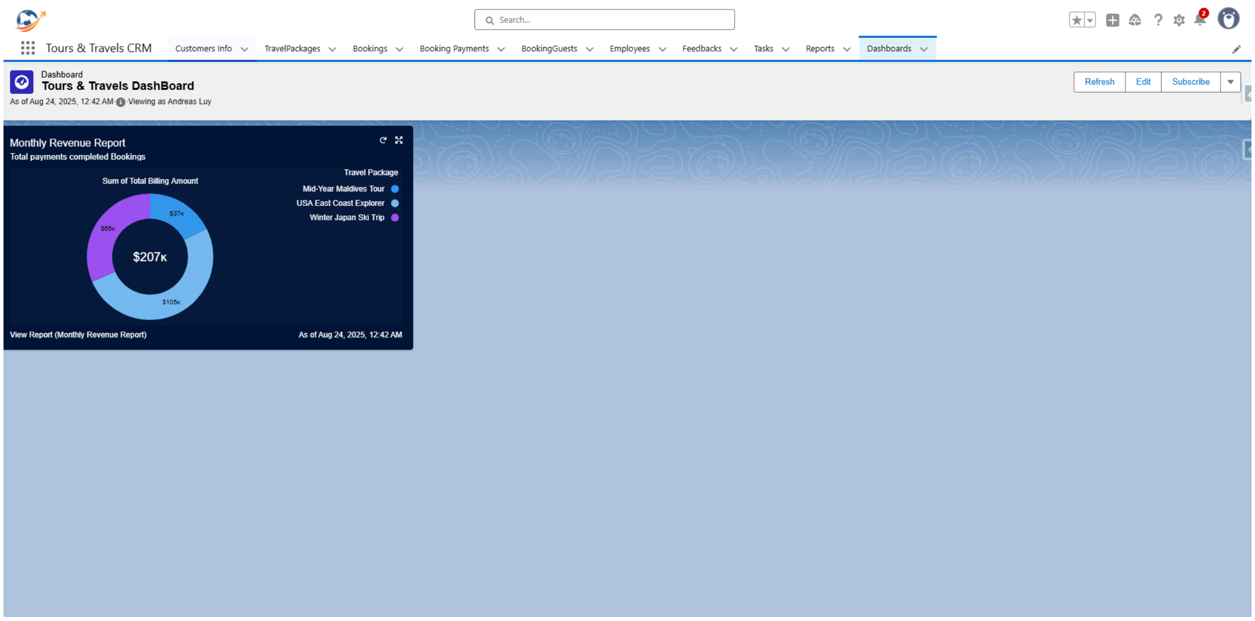


During this Milestone, I created several reports across customers, bookings, payments, employees, and travel packages. Each of the reports are organized into their proper folders. At first, I missed the note about creating records first before creating the reports, which caused delays and blank results until I added some dummy date. I also ran into a few errors when saving a record (in some objects), but I was able to fix them later after rechecking the field requirements.

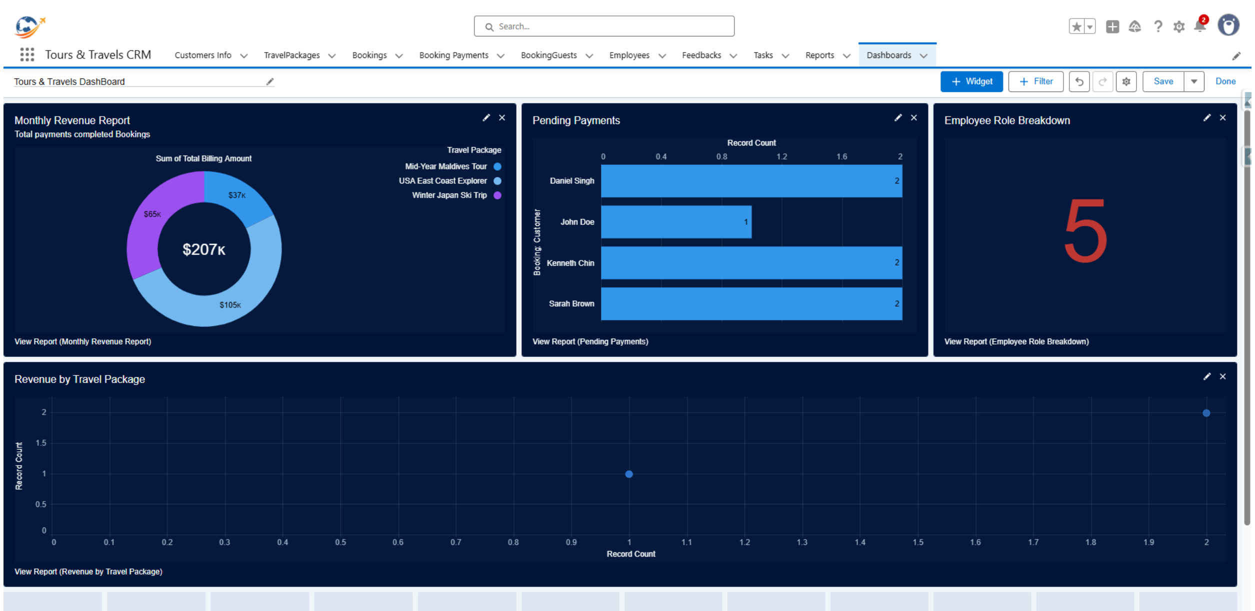
Despite these small challenges, the process gave me hands-on experience with grouping, summarizing, and adding charts in Salesforce's Reports. Overall, this milestone really helped me become familiar with the Salesforce UI and even understand how reports can turn raw data into a useful insight. Now for the next challenge, I think I'll display the reports into the dashboard? Maybe...

Milestone 18: Dashboards

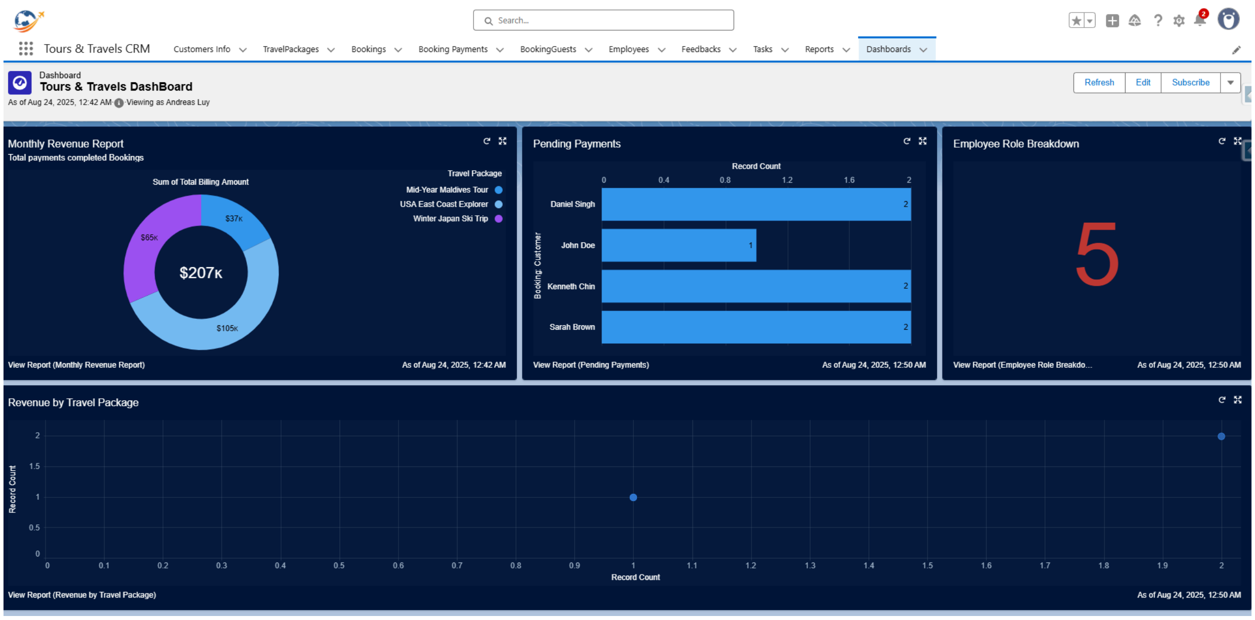
Activity 1: - Create Dashboard



Activity 2: Add other Components in the Dashboard

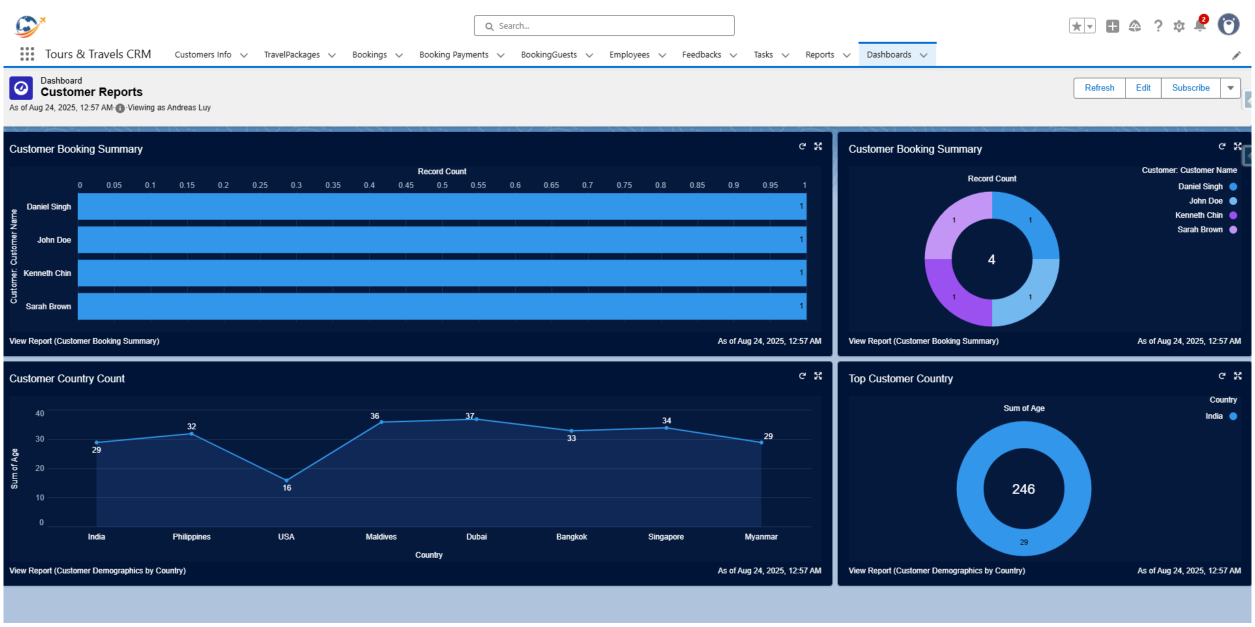


Activity 3: View Dashboard

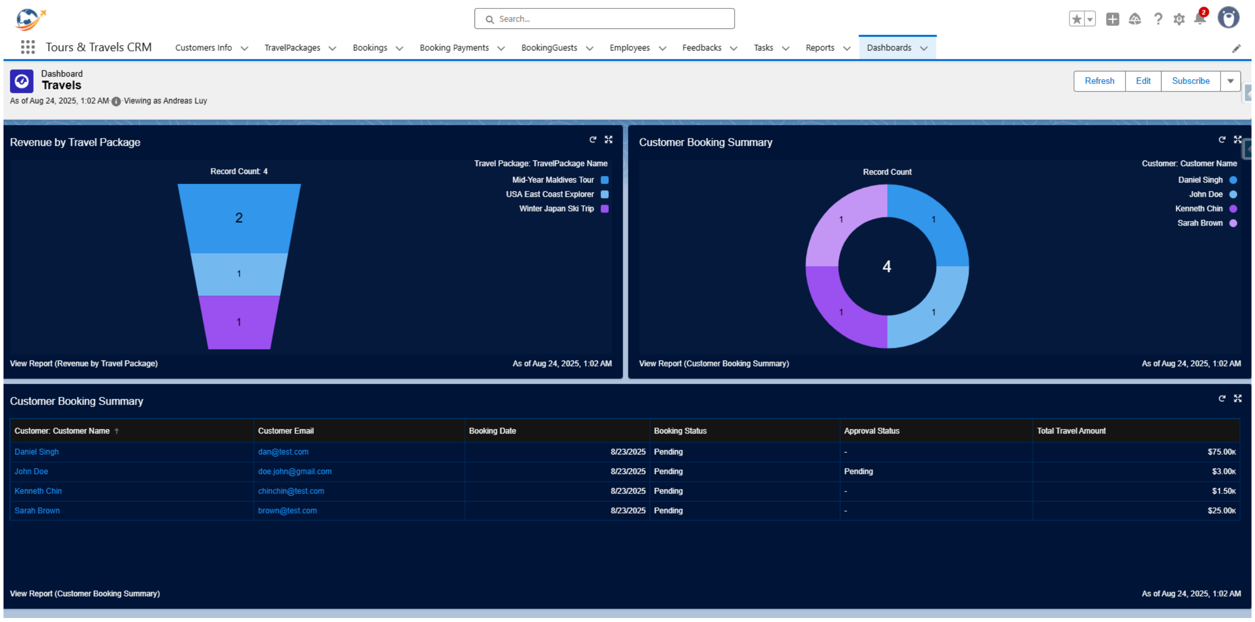


Activity 4: Create another 2 Dashboards

Customer Reports Dashboard



Travels Dashboard



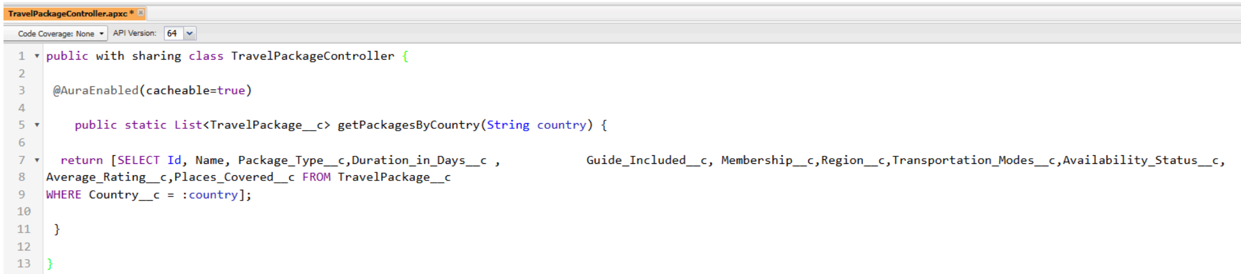
For Milestone 18, I initially struggled with what to put on the dashboard, so I started by asking myself something like "What would a manager need at a glance?" That led me to add widgets for Pending payments, Monthly revenue, Top travel packages, customer booking summary, and an employee role breakdown.

I created the underlying reports (with custom report types where needed) and summarized the data so that it's easier to read. So that when I finally add them to the dashboard, it would be easier and would look organized.

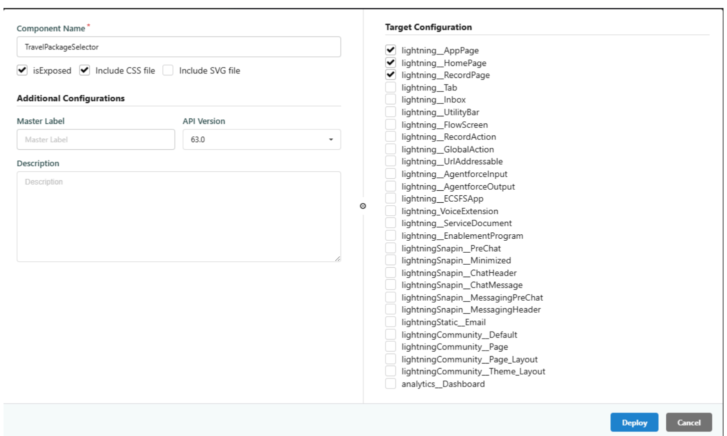
Overall, the dashboard I created gives a clear at-a-glance of what needs to be seen and will keep me up-to-date.

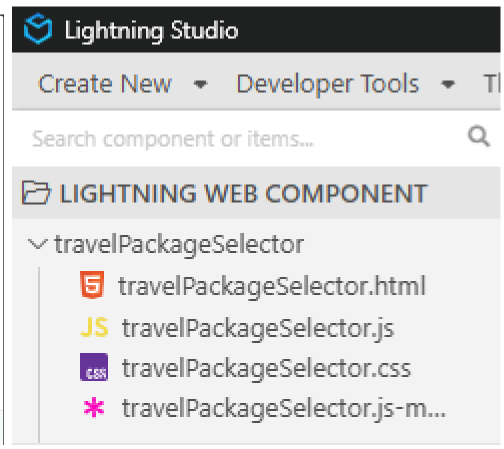
Milestone 19: Lightning Web Component (LWC) Creation

Activity 1: Create Lightning Web Component Creation



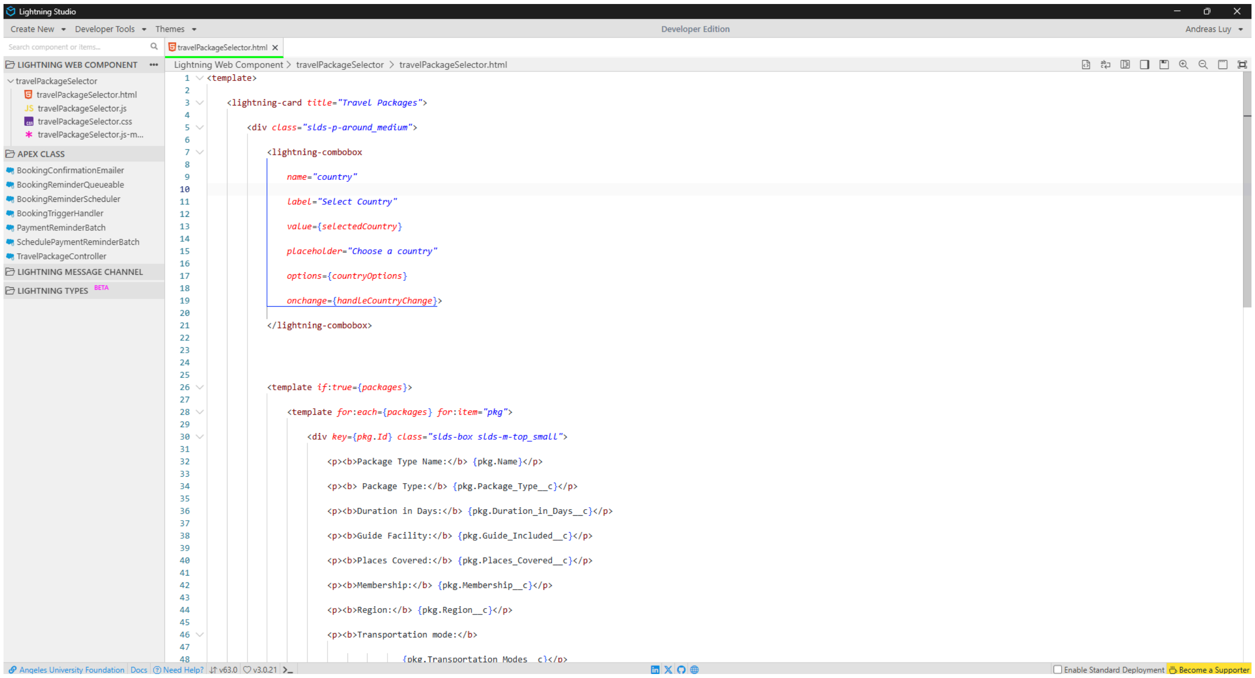

```
public with sharing class TravelPackageController {

 @AuraEnabled(cacheable=true)

    public static List<TravelPackage__c> getPackagesByCountry(String country) {

  return [SELECT Id, Name, Package_Type__c,Duration_in_Days__c ,           Guide_Included__c, Membership__c,Region__c,Transportation_Modes__c,Availability_Status__c,
Average_Rating__c,Places_Covered__c FROM TravelPackage__c
WHERE Country__c = :country];

 }

}
```





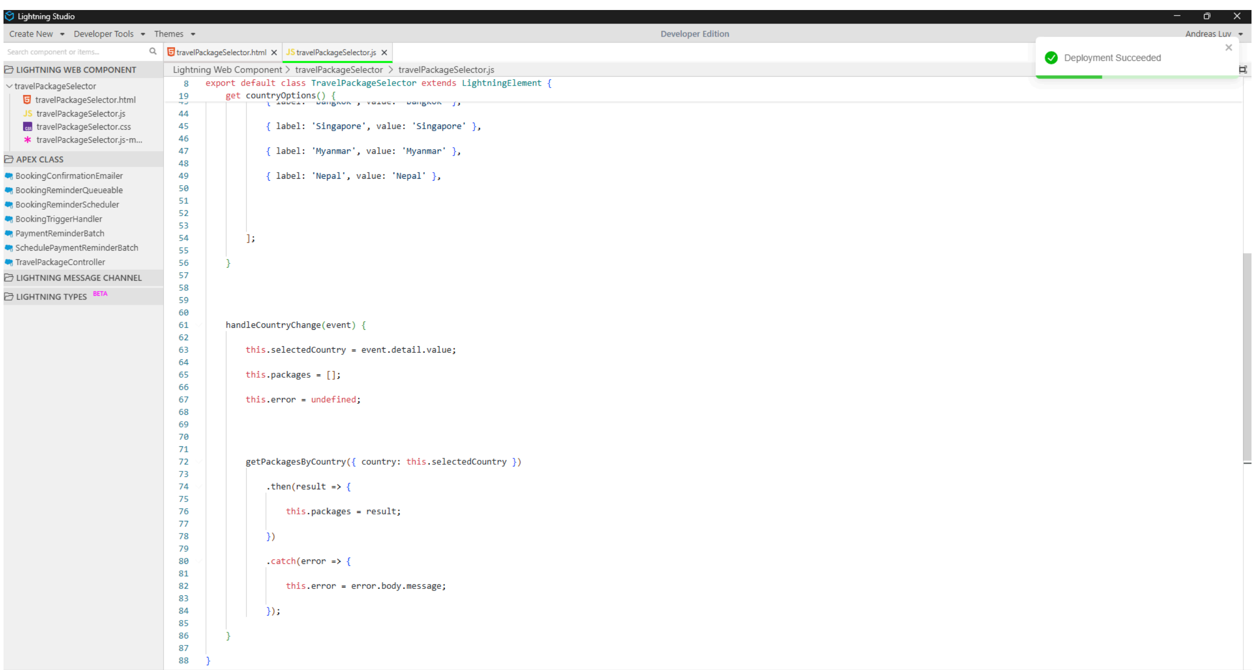
Inserting the source codes:

travelPackageSelector.html



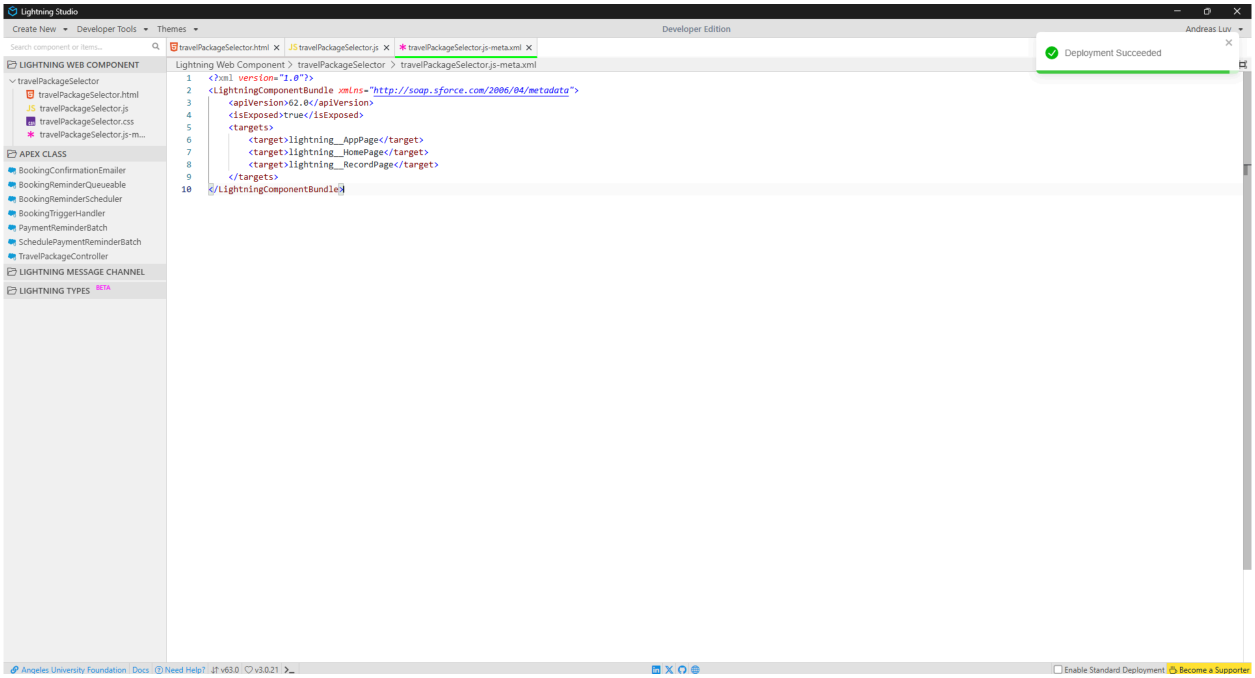
```html
<template>
    <lightning-card title="Travel Packages">
        <div class="slds-p-around_medium">
            <lightning-combobox
                name="country"
                label="Select Country"
                value={selectedCountry}
                placeholder="Choose a country"
                options={countryOptions}
                onchange={handleCountryChange}>
            </lightning-combobox>

            <template if:true={packages}>
                <template for:each={packages} for:item="pkg">
                    <div key={pkg.Id} class="slds-box slds-m-top_small">
                        <p><b>Package Type Name:</b> {pkg.Name}</p>
                        <p><b> Package Type:</b> {pkg.Package_Type__c}</p>
                        <p><b>Duration in Days:</b> {pkg.Duration_in_Days__c}</p>
                        <p><b>Guide Facility:</b> {pkg.Guide_Included__c}</p>
                        <p><b>Places Covered:</b> {pkg.Places_Covered__c}</p>
                        <p><b>Membership:</b> {pkg.Membership__c}</p>
                        <p><b>Region:</b> {pkg.Region__c}</p>
                        <p><b>Transportation mode:</b>
                                {pkg.Transportation_Modes__c}</p>
```

travelPackageSelector.js



```javascript
export default class TravelPackageSelector extends LightningElement {
    get countryOptions() {
                { label: 'Singapore', value: 'Singapore' },
                { label: 'Myanmar', value: 'Myanmar' },
                { label: 'Nepal', value: 'Nepal' },
            ];
    }

    handleCountryChange(event) {
        this.selectedCountry = event.detail.value;
        this.packages = [];
        this.error = undefined;

        getPackagesByCountry({ country: this.selectedCountry })
            .then(result => {
                this.packages = result;
            })
            .catch(error => {
                this.error = error.body.message;
            });
    }
}
```

travelPackageSelector.js-meta.xml

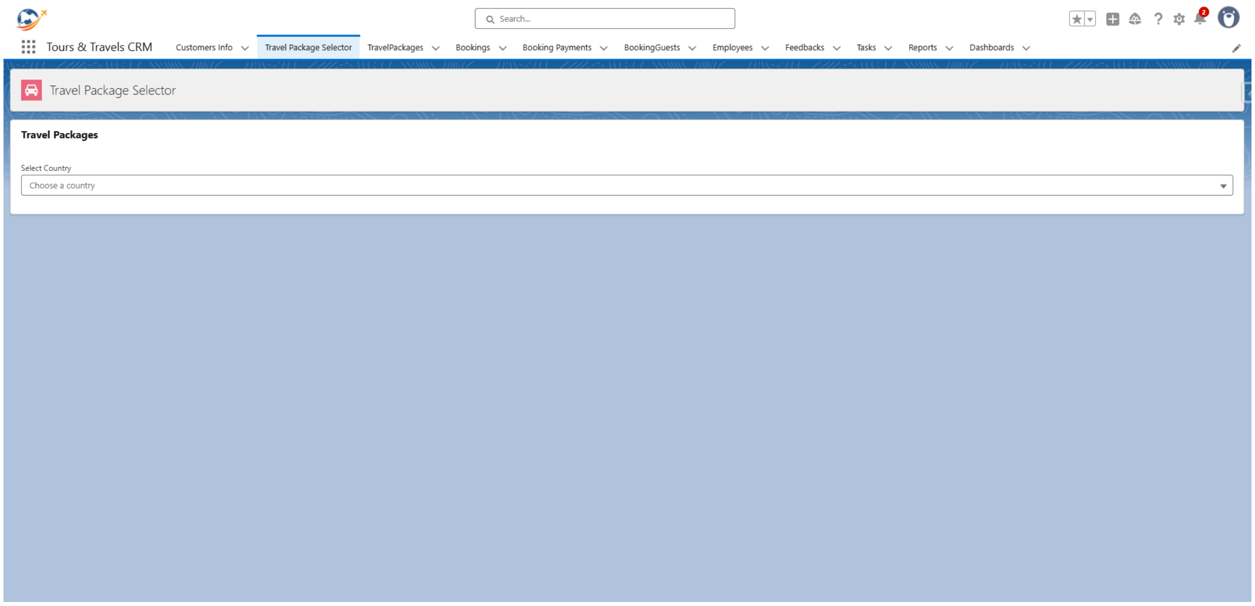


During this Milestone, I built a lightning web component with an Apex Controller in order to display the travel packages dynamically based on the selected country. At first, I found the setup a bit challenging since I haven't completely read the requirements. But after a while, it made me feel comfortable enough to explore by myself. I found it easier to navigate between files by using the extension provided.

This activity helped me understand the flow process much better especially when working with LWCs and even gave me hands-on experience. Overall, it feels like a rewarding milestone especially that I learned a new extension that can help me whenever I need to insert some source codes.

Milestone 20: Lightning App Page Creation

Activity 1: Adding the Lightning Component to the App page



For this Milestone, I was able to set up a Lightning App Page to showcase the Travel Package Selector as a single focused workspace. I created a new page called Travel Package Selector inside the Tours & Travels CRM.

The guide helped me by helping me click the right button and then when I launched it, the component loaded smoothly.

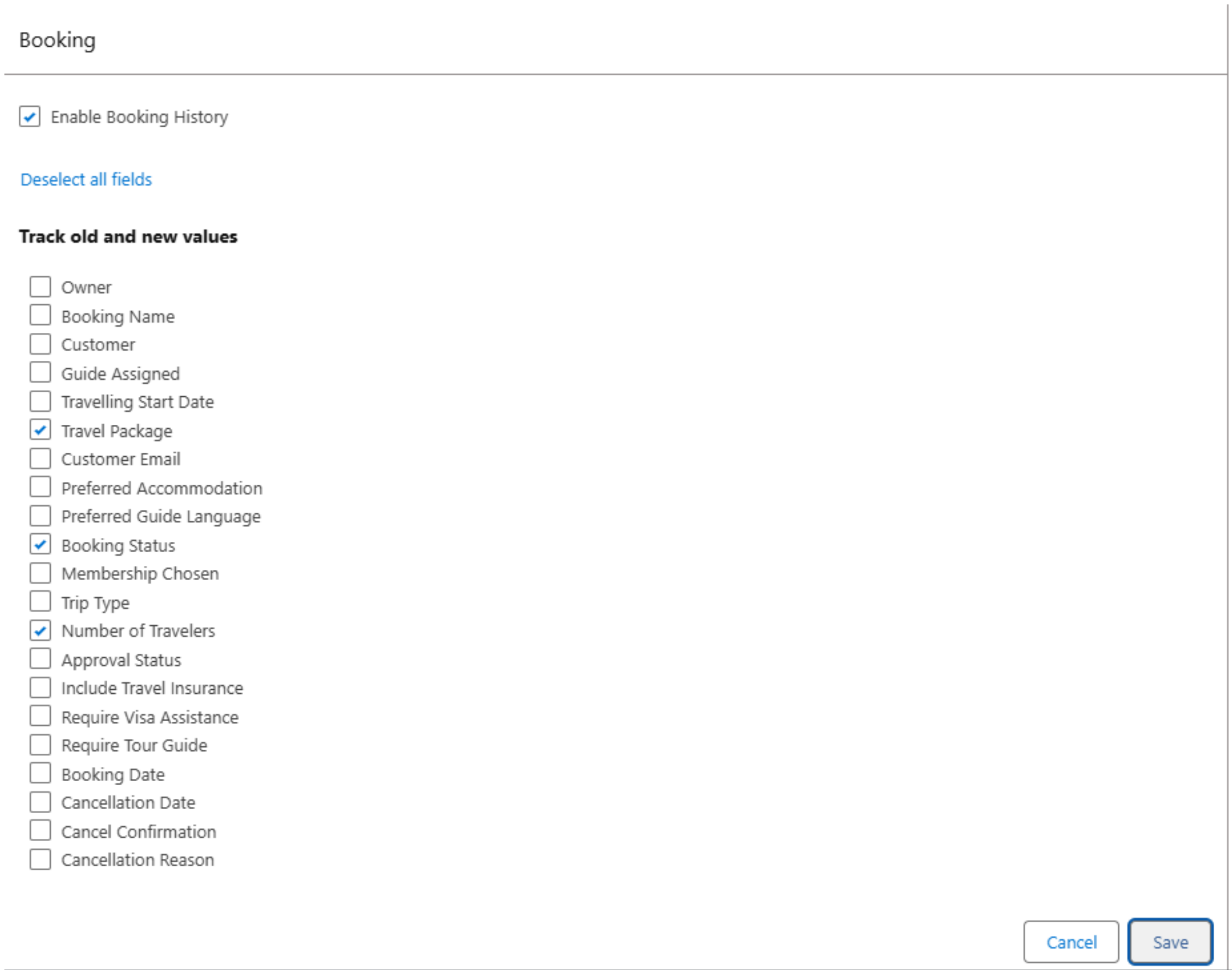
I also visited the newly created tab directly on the Tours & Travels, and it's a good thing that the Select a country option is displaying without any issues.

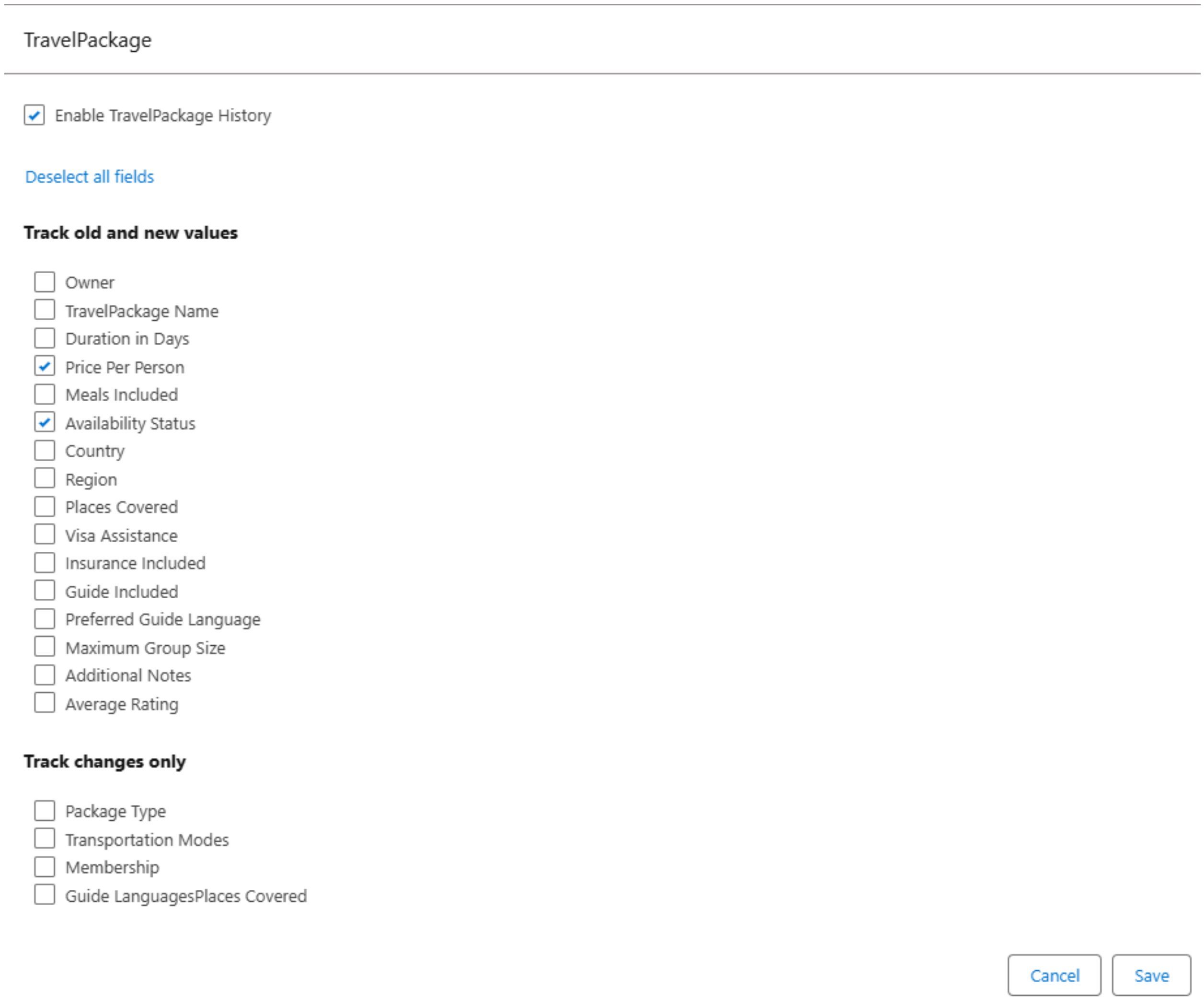
Overall, this was a simple change with a big Users experience impact

Phase 4: Data Migration, Testing & Security

Milestone 21: Field History Tracking

Activity 1



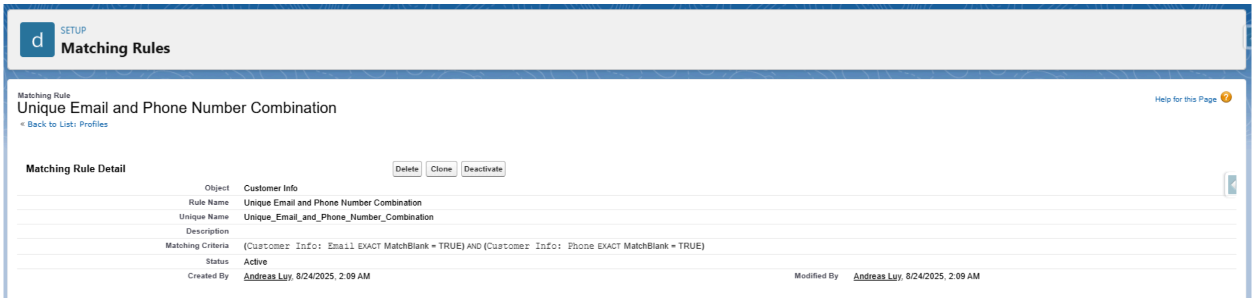


For this milestone, I implemented Field History Tracking. From setup, I was able to find the Field History Tracking and was able to enable the required fields to be tracked for the objects: Booking and Travel Package. This activity helped me understand how I can include an audit trail for my app- Tours & Travels CRM.

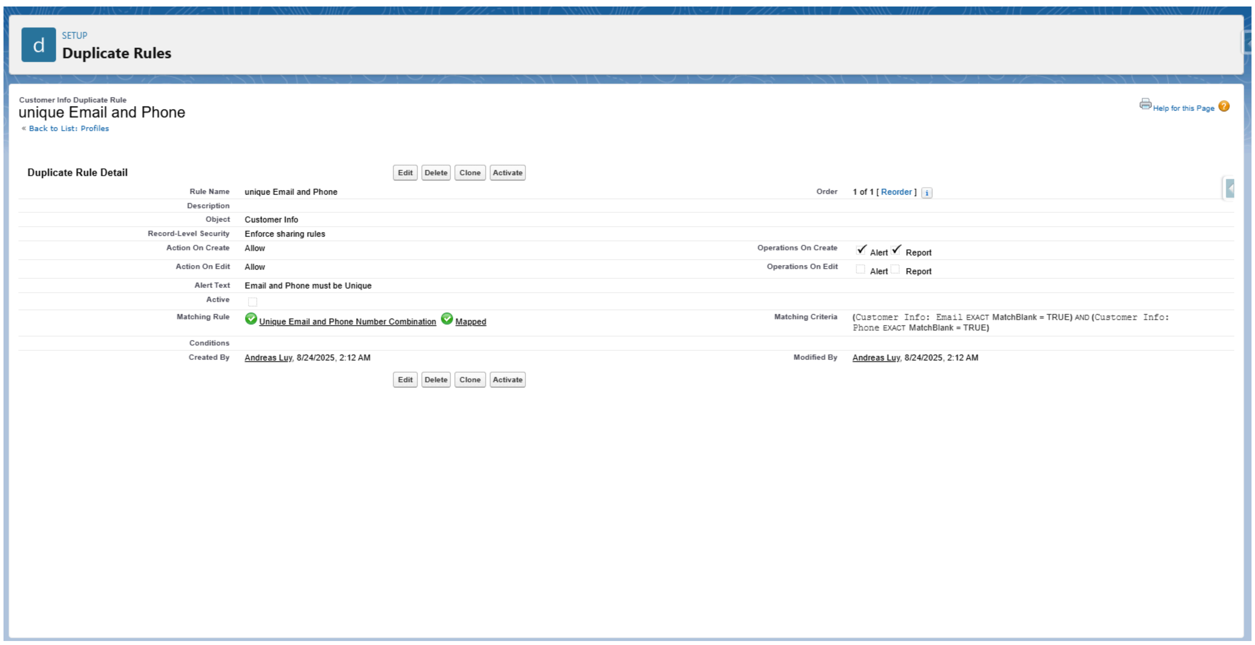
I also learned that tracking begins after it's enabled and supports up to 20 fields per object; however, in a business context we should track only the fields that matter to avoid redundancy.

Milestone 22: Duplicate and Matching Rules

Activity 1: Create a Custom Matching Rule



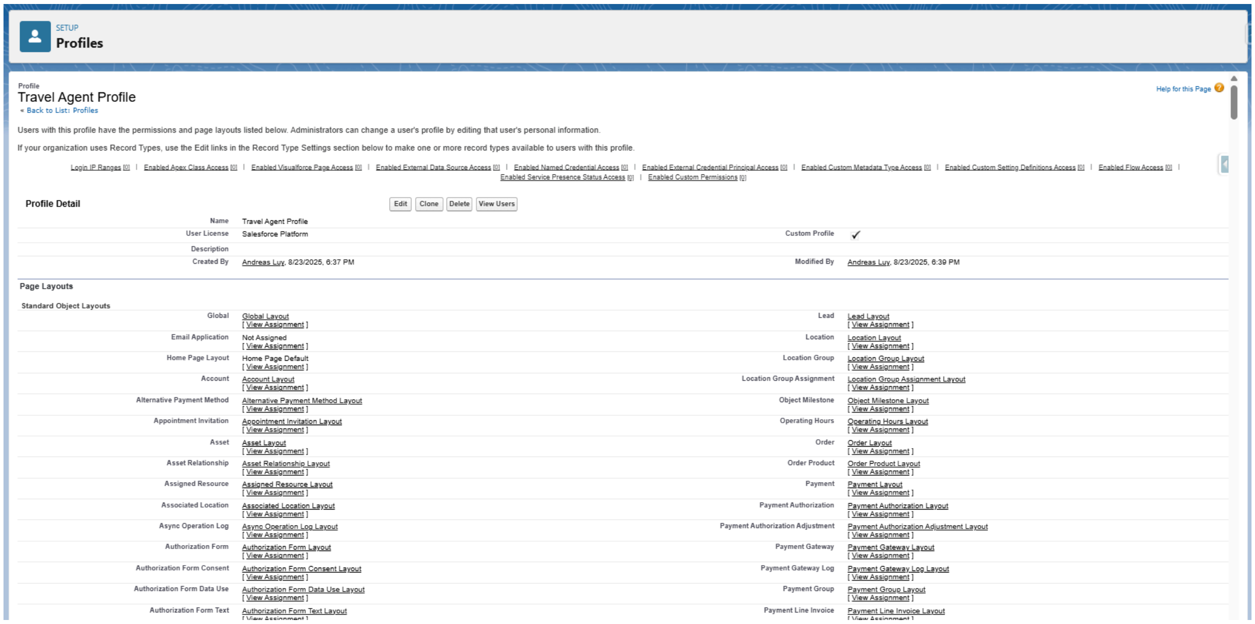
Activity 2: Create a Duplicate Rule



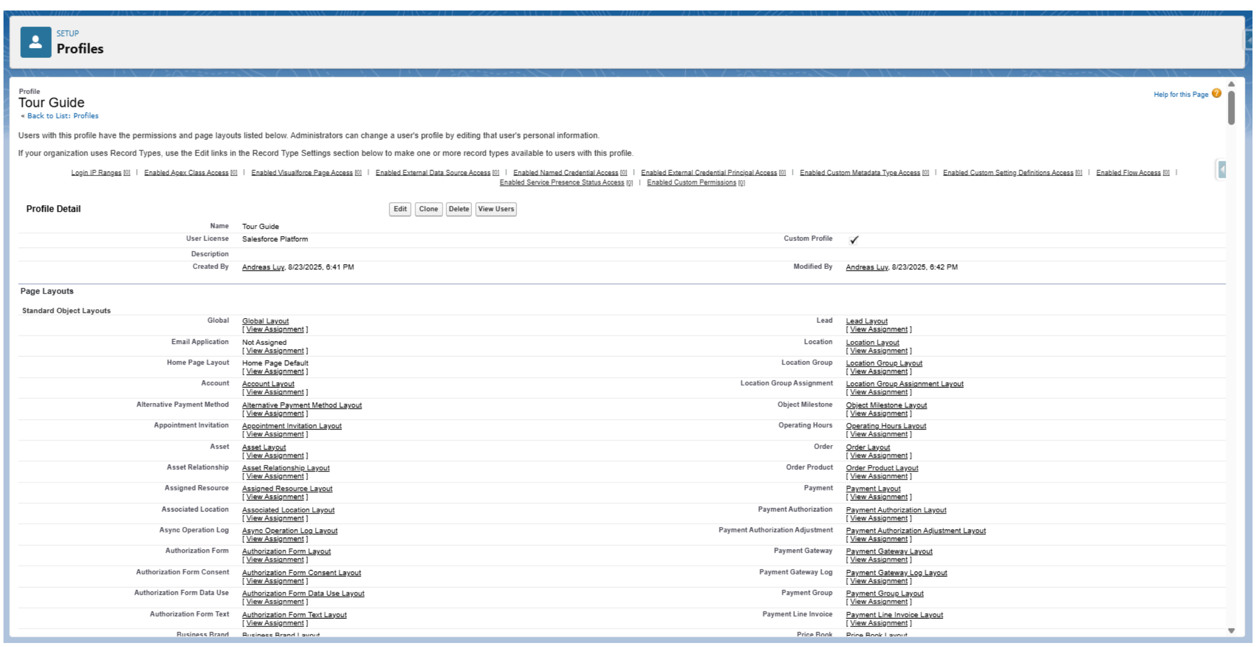
Creating the rules for the Customer Info using email and phone combinations gave me a practical way to ensure the data integrity. It's very important to note how easy it is for the duplicated data to creep in. But with these rules in place, the app can flag it before they became a problem.

Milestone 23: Profiles

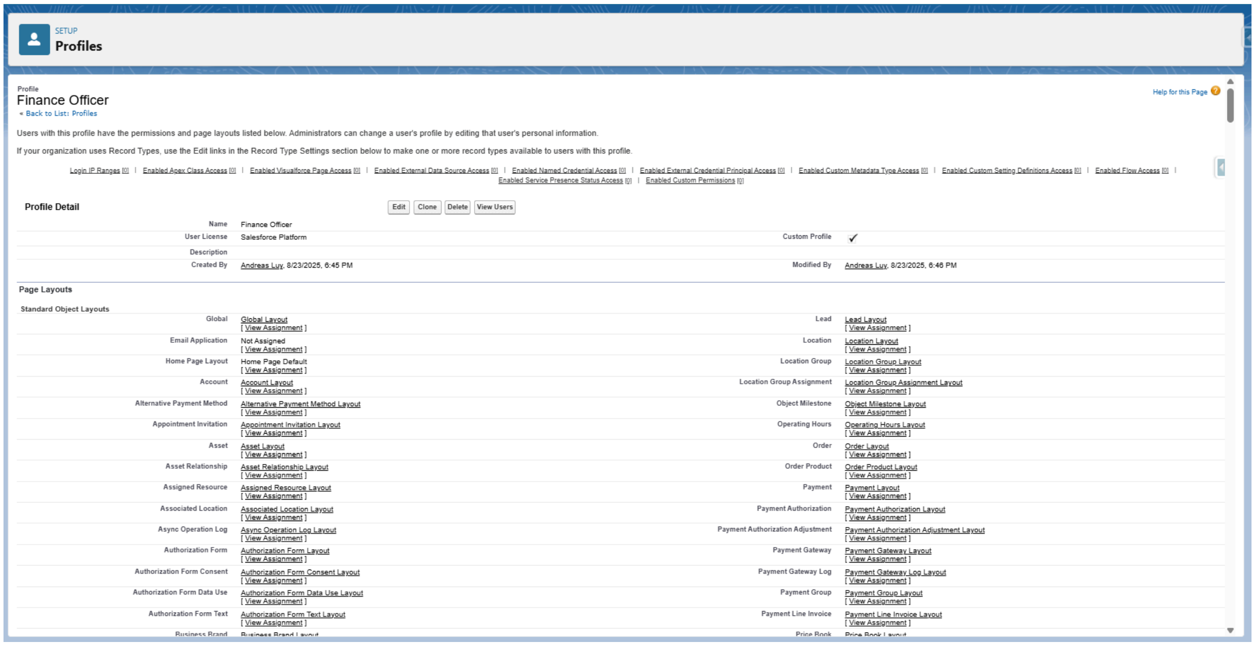
Activity 1: Travel Agent Profile Creation



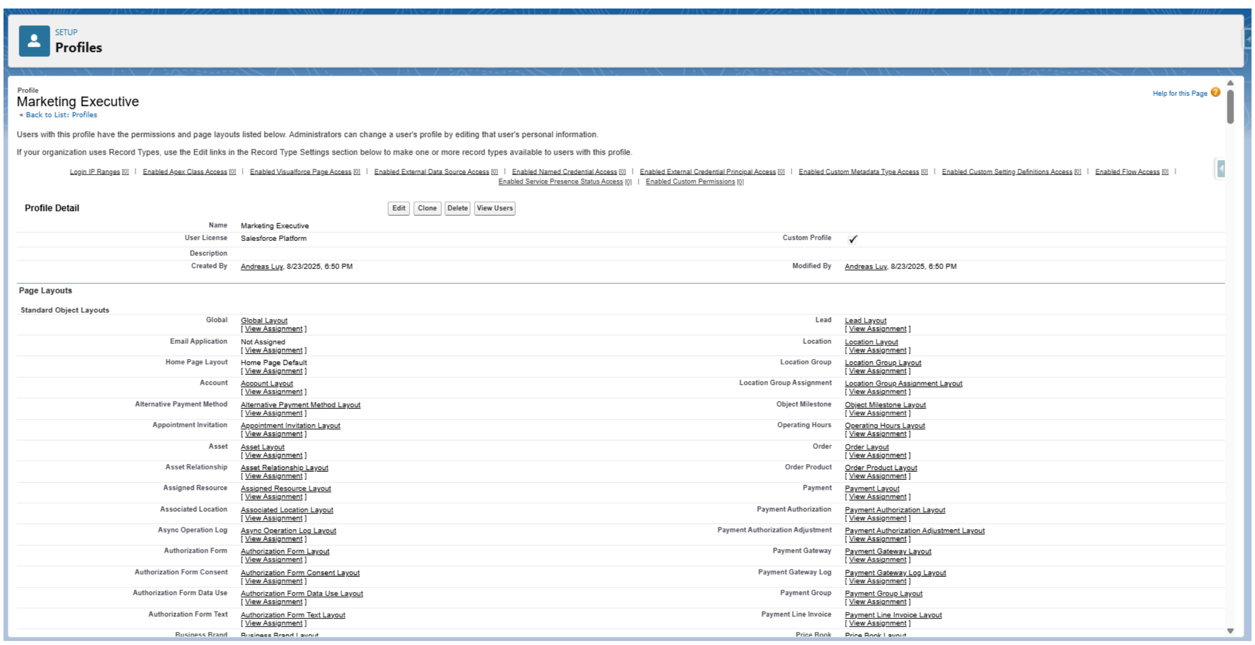
Activity 2: Tour Guide Profile Creation



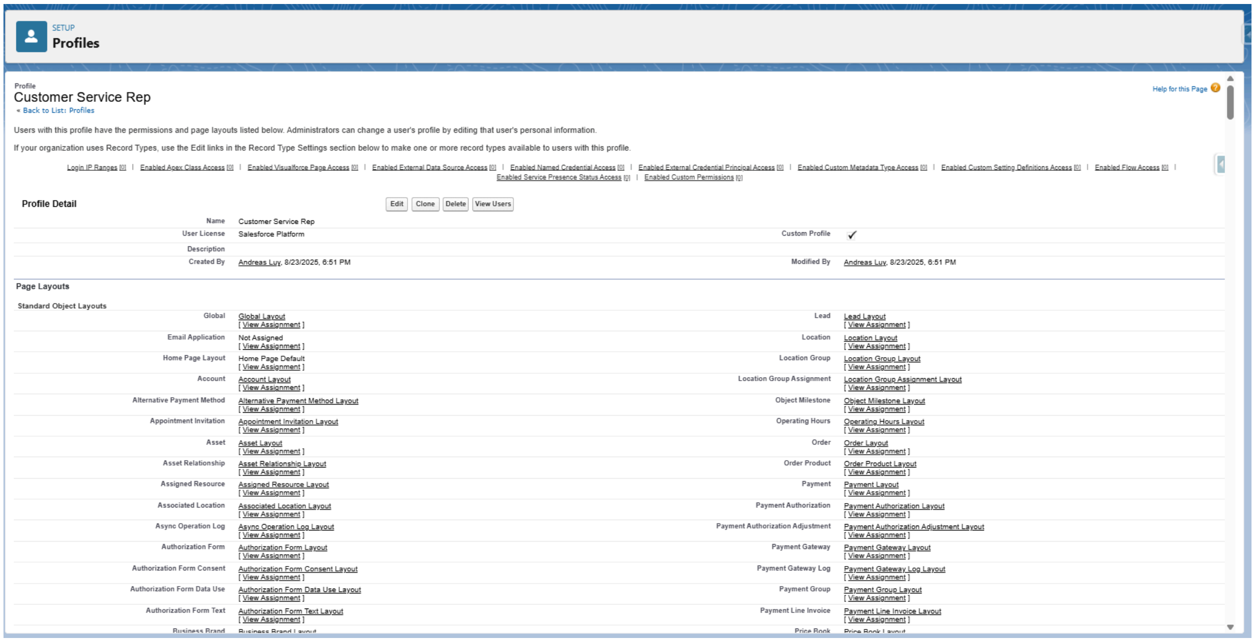
Activity 3: Finance Officer Profile Creation



Activity 4: Marketing Executive Profile Creation



Activity 5: Customer Service Rep Profile Creation

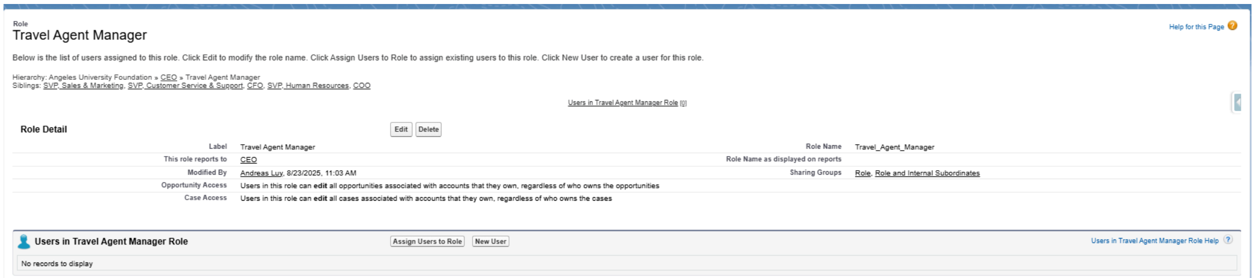


Creating the custom profiles such as the Travel Agent, Tour Guide, Finance Officer, Marketing Executive, and Customer Service Rep helped me design the role-based access for the application I am developing which is the Tours & Travels CRM.

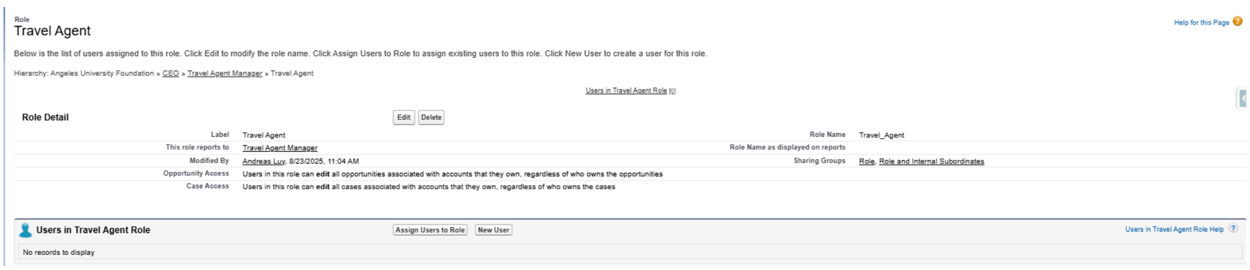
Each profile was given permissions that matched the real-world responsibilities. This milestone helped me understand the importance of security and access control, making the system safe.

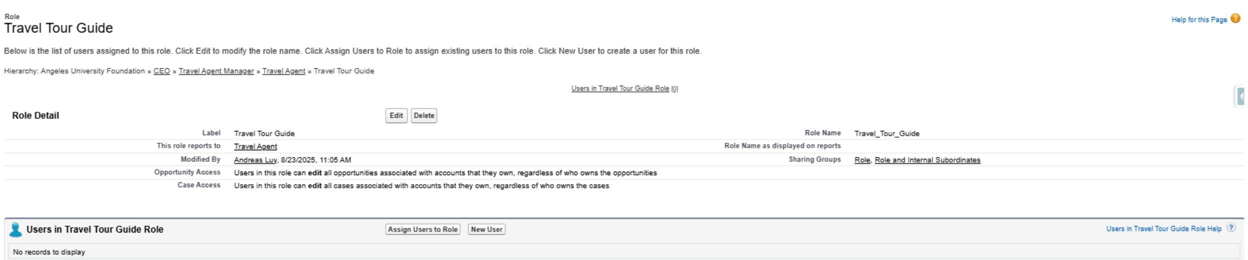
Milestone 24: Roles & Role Hierarchy

Activity 1: Creation of Travel Agent Manager Role

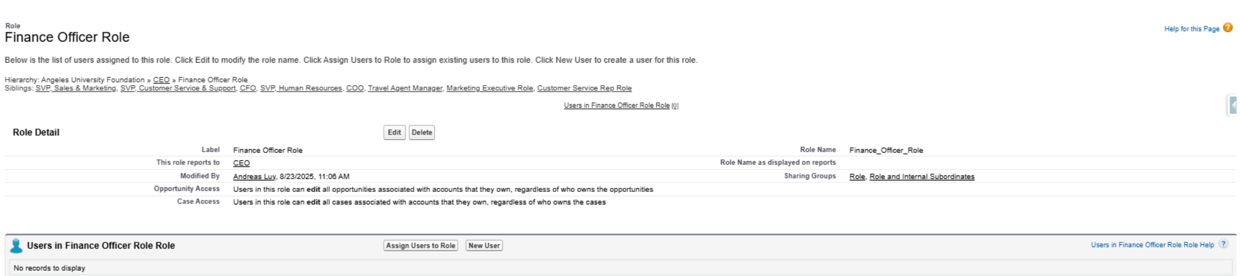


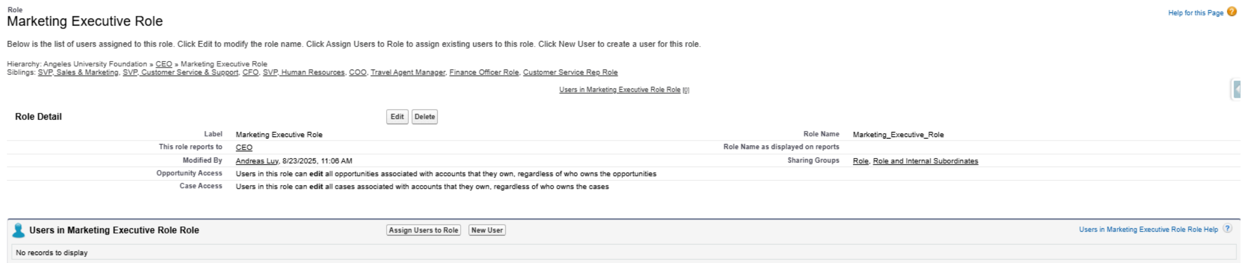
Activity 2: Creating other Roles Under Travel Agent Manager Role

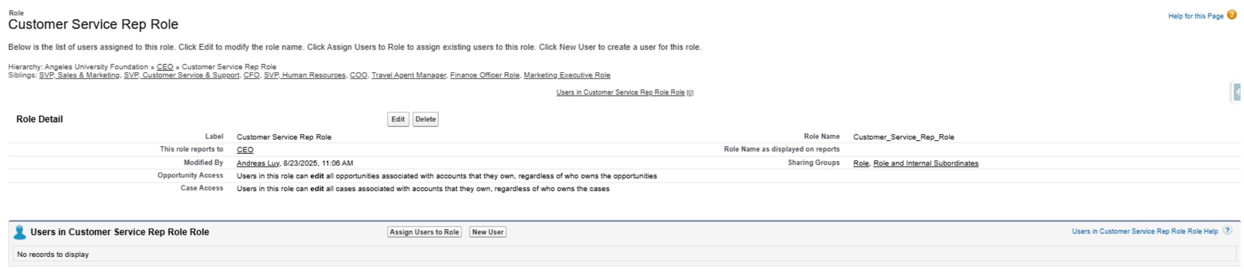




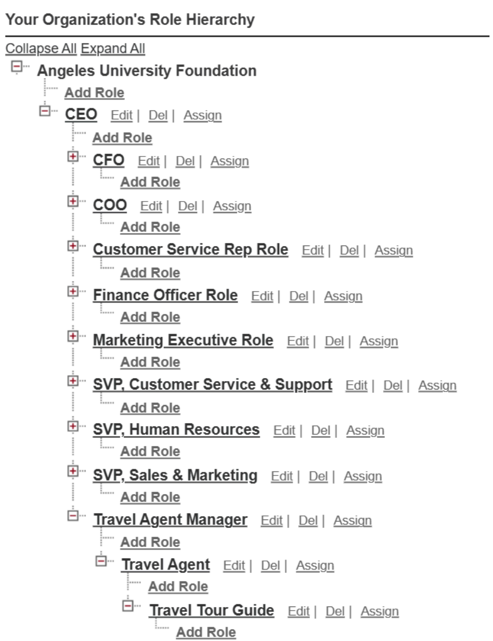
Activity 3: Creating other Three Roles Under CEO Role







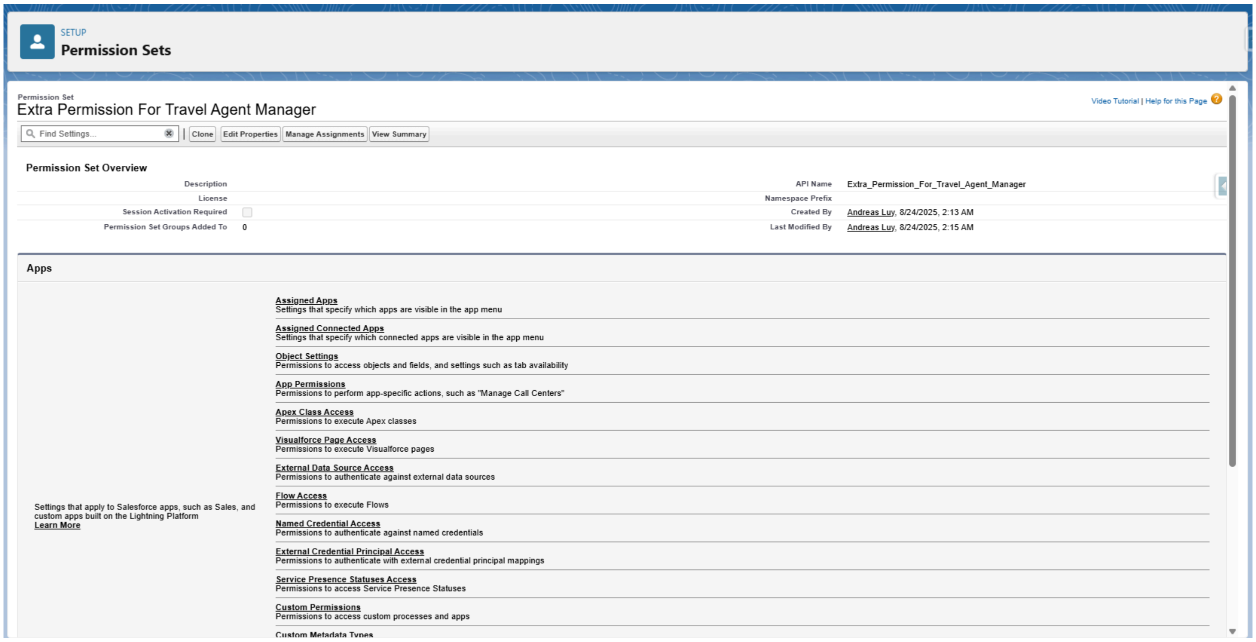
Role Hierarchy

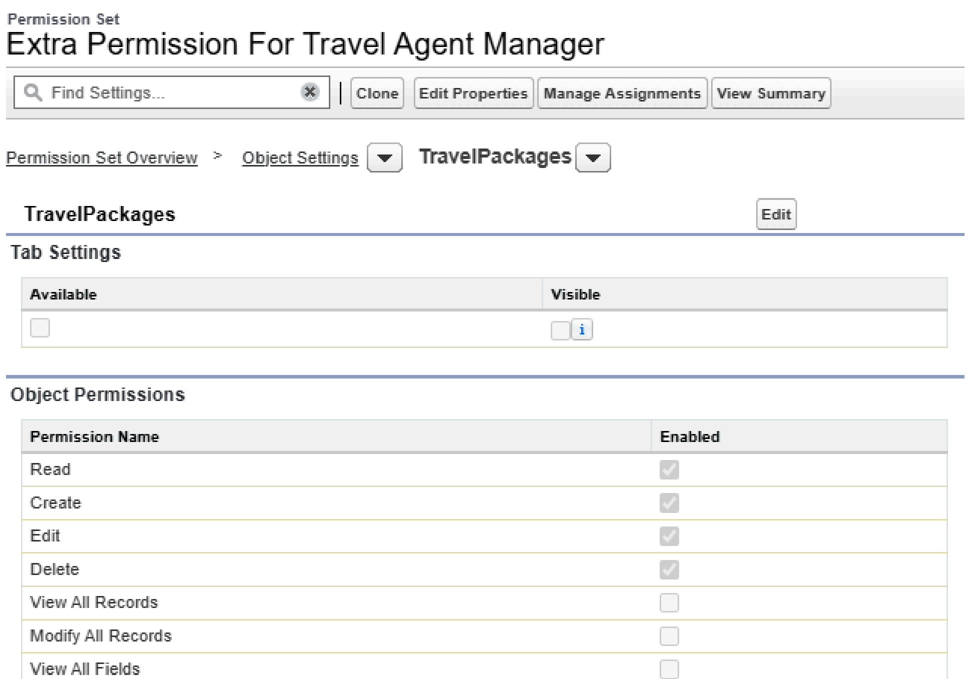


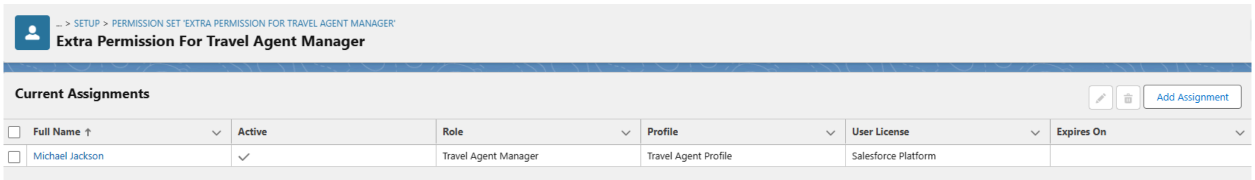
Building the Role Hierarchy for the Tours & Travels CRM made it clear for me how Salesforce supports structured data visibility. This milestone helped me appreciate how each role complements the profiles while the profiles control what the Users can do, and how roles control what data they can see.

Milestone 25: Permission Sets

Activity 1: Creating New Permission Set



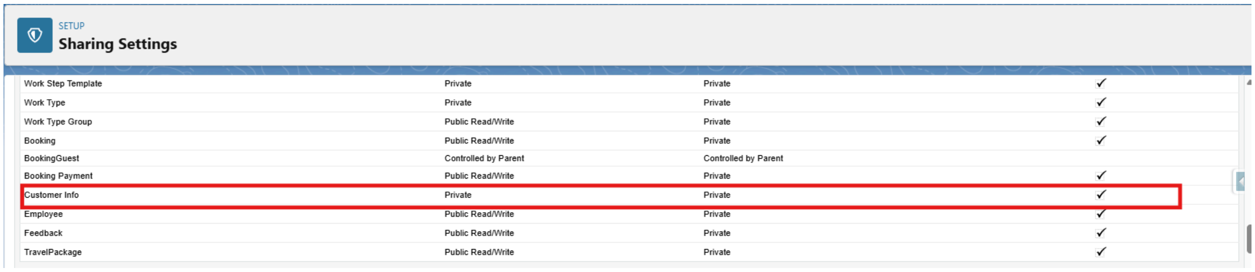




Creating a permission set for the Travel Agent Manger showed me how Salesforce can make access management more adaptable. Instead of creating multiple profiles, I could simply assign an extra permission if needed!

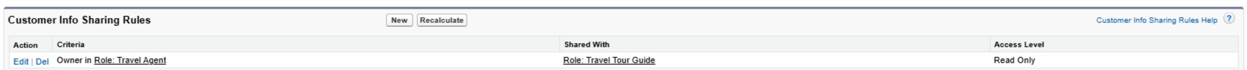
Milestone 26: Sharing Settings

Activity 1: Creating Sharing Rule



SETUP
Sharing Settings

Work Step Template	Private	Private	✓
Work Type	Private	Private	✓
Work Type Group	Public Read/Write	Private	✓
Booking	Public Read/Write	Private	✓
BookingGuest	Controlled by Parent	Controlled by Parent	
Booking Payment	Public Read/Write	Private	✓
Customer Info	Private	Private	✓
Employee	Public Read/Write	Private	✓
Feedback	Public Read/Write	Private	✓
TravelPackage	Public Read/Write	Private	✓



Customer Info Sharing Rules New Recalculate Customer Info Sharing Rules Help

Action	Criteria	Shared With	Access Level
Edit | Del	Owner in Role: Travel Agent	Role: Travel Tour Guide	Read Only

Configuring the sharing rules for the Customer Info helped me see how Salesforce balances the security with usability. By setting the default permissions and access to private and then selectively sharing the customer records with the Tour Guide role, I ensured data privacy while still enabling collaboration. This milestone gave me a better understanding of Salesforce's layered security model.

Milestone 27: Test Classes

Activity 1 : Creation of a Test Class

BookingTriggerTest.apxc

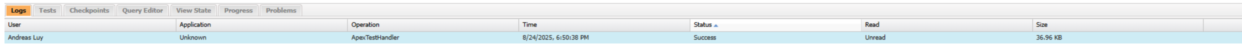
Code Coverage: None API Version: 64 Run Test Go To

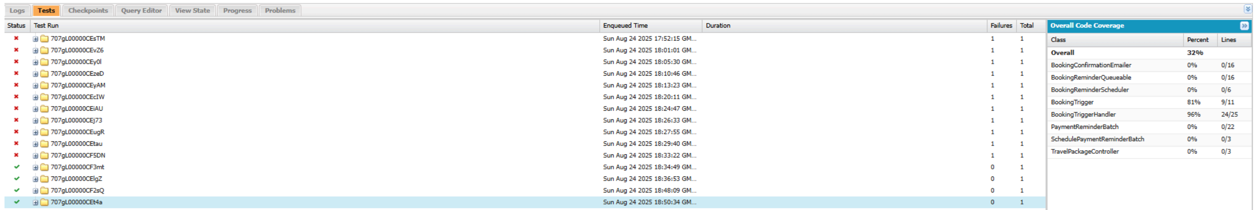
```
@isTest
private class BookingTriggerTest {

    @isTest
    static void testTriggerCreatesPaymentAndGuestsWithUpdatedFields() {

        // 1. Create a sample Customer_Info__c record (required for Booking)
        Customer_Info__c customer = new Customer_Info__c(
            Name = 'Andreas Luy',
            Email__c = 'luy.andreas@student.auf.edu.ph',
            Phone__c = '1234567890',
            Date_Of_Birth__c = Date.newInstance(2004, 7, 27),
            Country__c = 'Philippines'
        );
        insert customer;

        // 2. Create a sample TravelPackage__c record (linked to Booking)
        TravelPackage__c packageRec = new TravelPackage__c(
            Name = 'Taste the great sisig of Angeles',
            Country__c = 'Philippines',
            Price_Per_Person__c = 2000,
            Duration_in_Days__c = 3,
            Places_Covered__c = 'Angeles City'
        );
        insert packageRec;

        // 3. Create a Booking__c record
        //    - This will fire the trigger when inserted
        Booking__c booking = new Booking__c(
            Number_of_Travelers__c = 3,
            Booking_Status__c = 'Pending',
            Travelling_Start_Date__c = Date.today().addDays(10),
            Travel_Package__c = packageRec.Id,
            Membership_Chosen__c = 'Gold',
            Preferred_Accommodation__c = 'Guest House',
            Customer__c = customer.Id,
            Customer_Email__c = 'luy.andreas@student.auf.edu.ph',
            Booking_Date__c = Date.newInstance(2025, 7, 7)
        );
```

Logs, Tests, and Problems

```
        // 4. Run the trigger logic inside a test context
        Test.startTest();
        insert booking; // Trigger fires here -> should create Payment + Guests
        Test.stopTest();

        // 5. Validate Booking_Payment__c was created with correct default values
        List<Booking_Payment__c> payments = [
            SELECT Id, Booking__c, Payment_Status__c
            FROM Booking_Payment__c
            WHERE Booking__c = :booking.Id
        ];
        System.assertEquals(1, payments.size(), 'One payment record should be created.');
        System.assertEquals('Pending', payments[0].Payment_Status__c, 'Default Payment Status should be Pending.');

        // 6. Validate BookingGuest__c records were created (based on traveler count)
        List<BookingGuest__c> guests = [
            SELECT Id, Booking__c, Name
            FROM BookingGuest__c
            WHERE Booking__c = :booking.Id
        ];
        System.assertEquals(3, guests.size(), 'Three BookingGuest records should be created.');
        System.assertEquals('Guest 1', guests[0].Name, 'Guest naming should follow convention.');
    }
}
```

Logs and Tests





Writing the test classes for the Booking Trigger and BookingTriggerHandler gave me a hard time but my key takeaway for this is that in real-world situations, I was able to verify that the booking payment and guest records were being created as intended. This milestone made me understand the importance of testing.

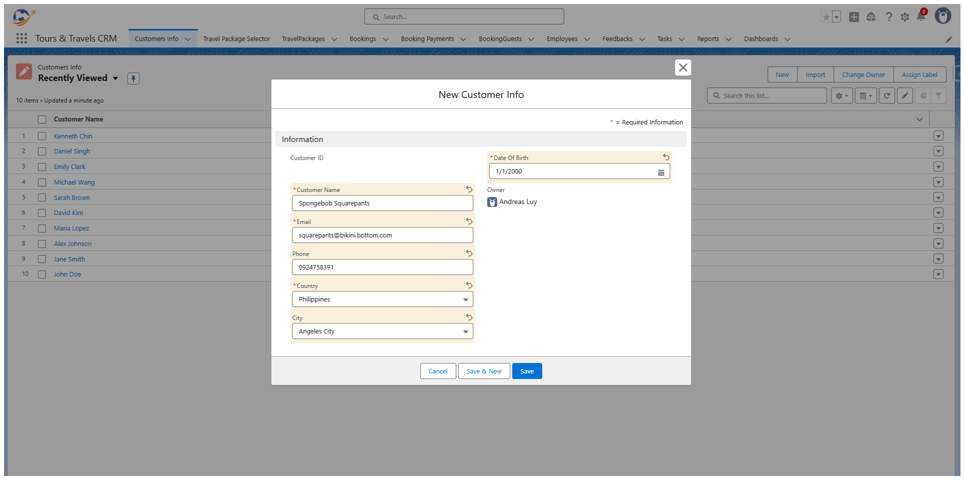
Milestone 28: Preparing Test Cases & Fixing Defects

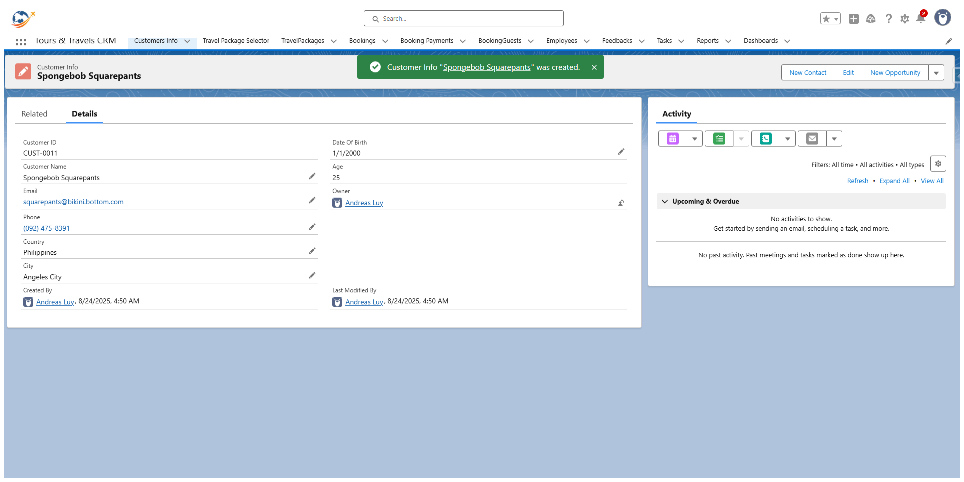
Activity 1: Test Case 1

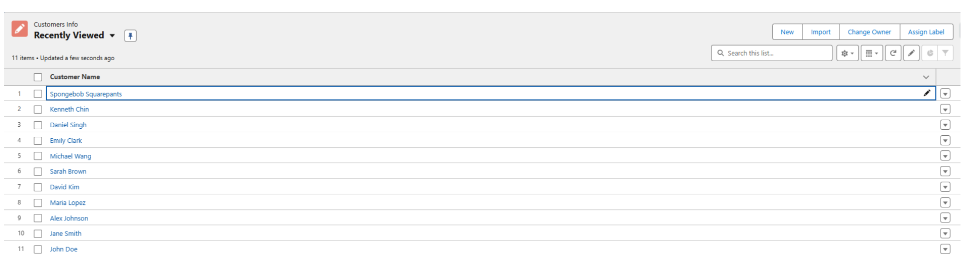
Customer Creation

Test Case	**Verify that a new Customer can be created successfully with all mandatory fields.**
Environment	Includes • OS: Windows 11 • Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** • Navigate to the Customer object page. • Fill in required fields. • Click Save
Expected Result	• The Customer record should be saved successfully. • The record should appear on the Customers List page.
Status	**PASSED**

Test Evidences





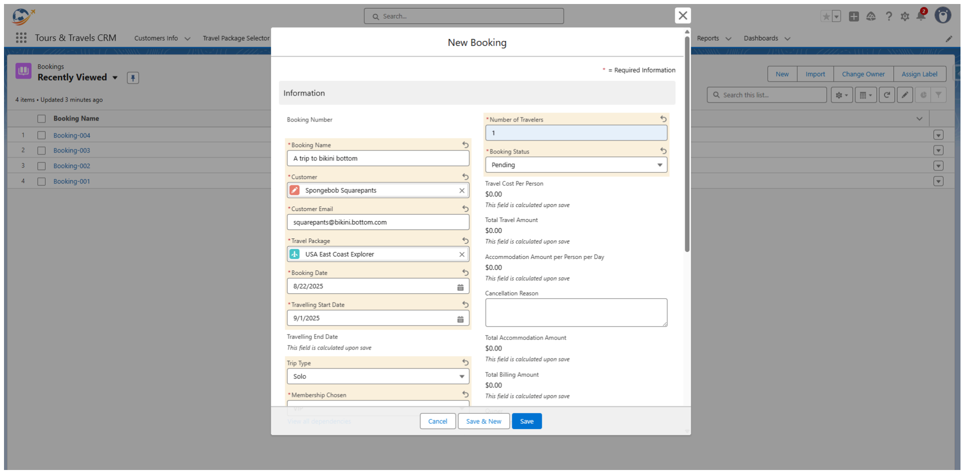


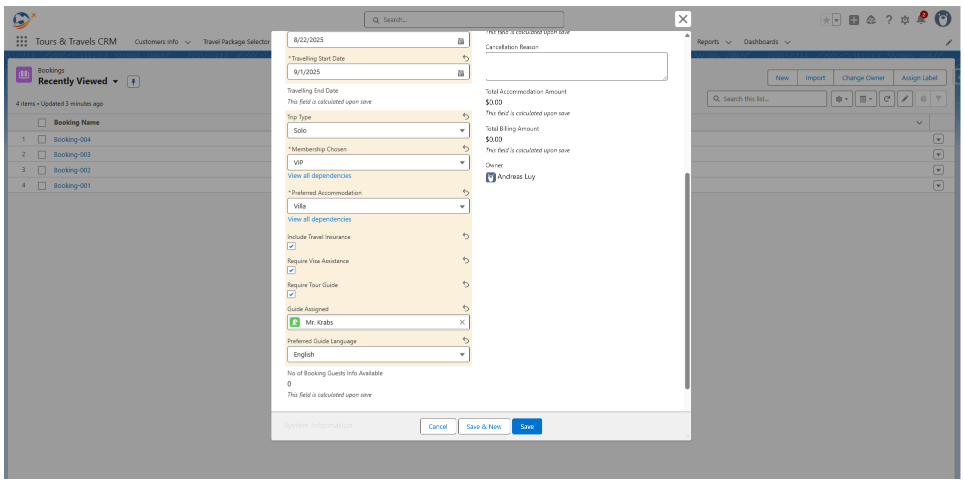
Activity 2: Test Case 2

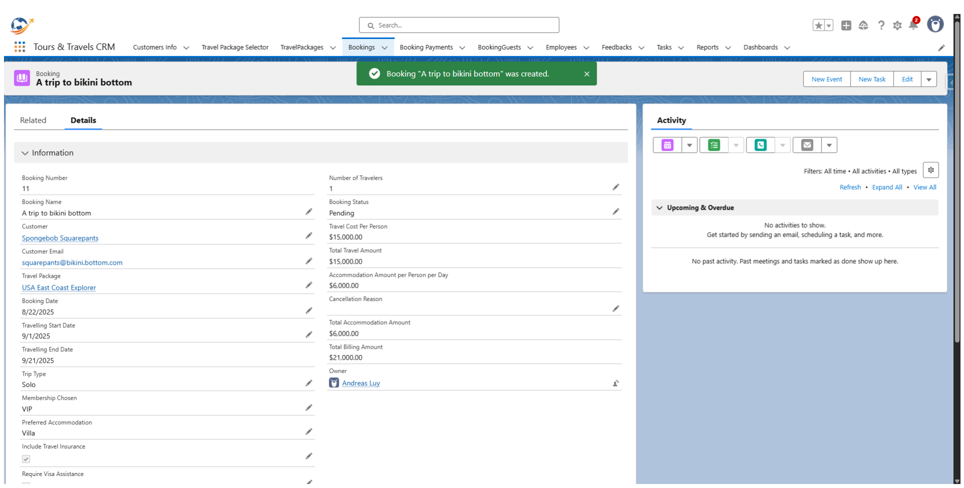
Booking Creation

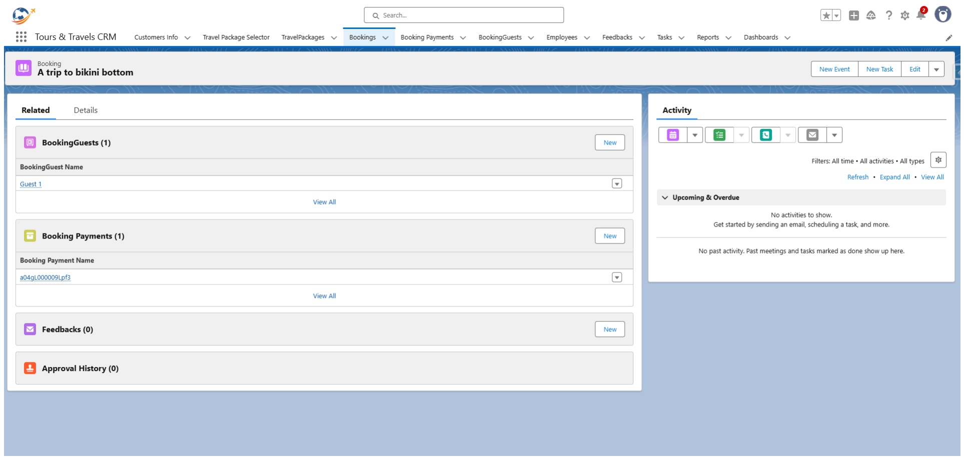
Test Case	**Verify that a new booking can be created successfully with all mandatory fields.**
Environment	Includes • OS: Windows 11 • Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** • Navigate to the Booking object page. • Fill in required fields. • Click Save.
Expected Result	• The Booking record should be saved successfully. • The record should appear on the Booking List page. • A Record related to Booking must be created in the Booking Payments object with Payment Status pending. • BookingGuests records must be created and maintained to exactly match the Number of Travellers value specified in the related Booking record..
Status	**PASSED**

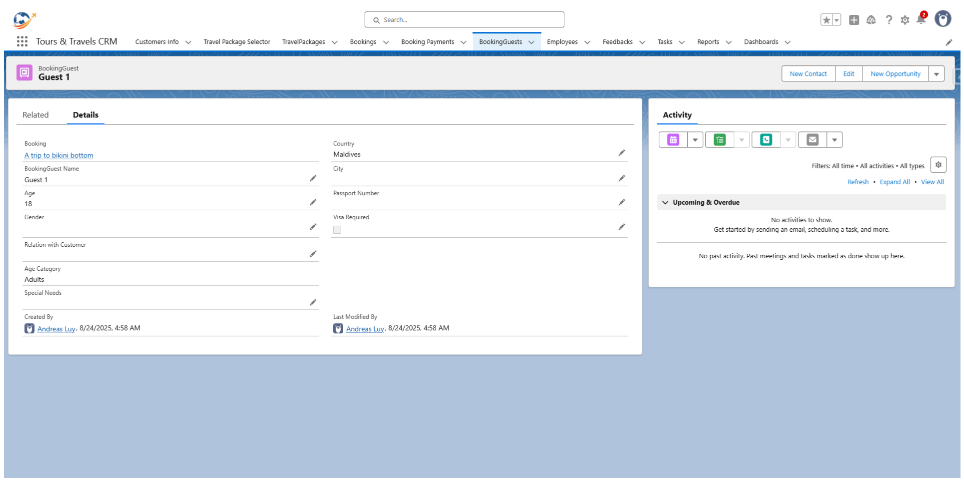
Test Evidences

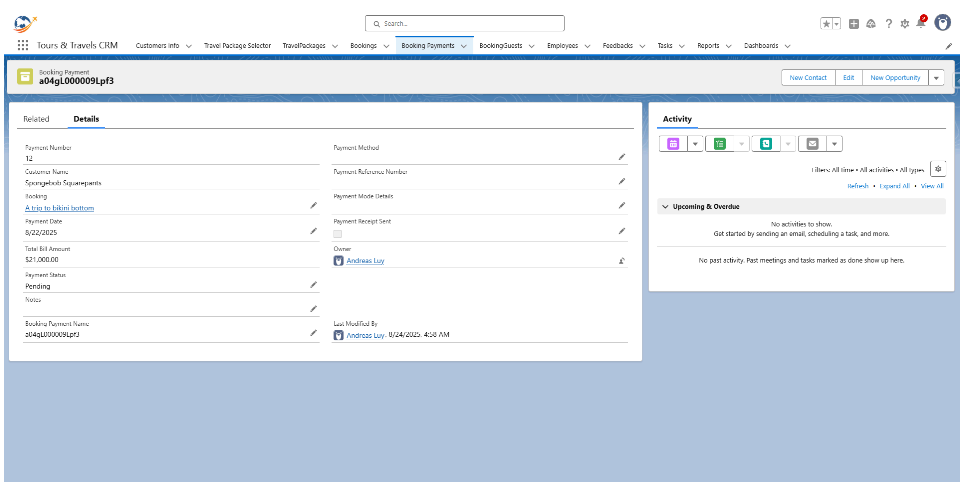







Tours & Travels CRM
Customers Info
Travel Package Selector
TravelPackages
Bookings
Booking Payments
BookingGuests
Employees
Feedbacks
Tasks
Reports
Dashboards
Booking
A trip to bikini bottom
New Event
New Task
Edit
Related
Details
BookingGuests (1)
New
BookingGuest Name
Guest 1
View All
Booking Payments (1)
Booking Payment Name
a04gL000009Lpf3
Feedbacks (0)
Approval History (0)
Activity
Filters: All time • All activities • All types
Refresh • Expand All • View All
Upcoming & Overdue
No activities to show.
Get started by sending an email, scheduling a task, and more.
No past activity. Past meetings and tasks marked as done show up here.


Tours & Travels CRM
BookingGuest
Guest 1
New Contact
Edit
New Opportunity
Related
Details
Booking
A trip to bikini bottom
Country
Maldives
BookingGuest Name
Guest 1
City
Age
18
Passport Number
Gender
Visa Required
Relation with Customer
Age Category
Adults
Special Needs
Created By
Andreas Luy, 8/24/2025, 4:58 AM
Last Modified By
Andreas Luy, 8/24/2025, 4:58 AM
Activity
Upcoming & Overdue
No activities to show.
Get started by sending an email, scheduling a task, and more.
No past activity. Past meetings and tasks marked as done show up here.

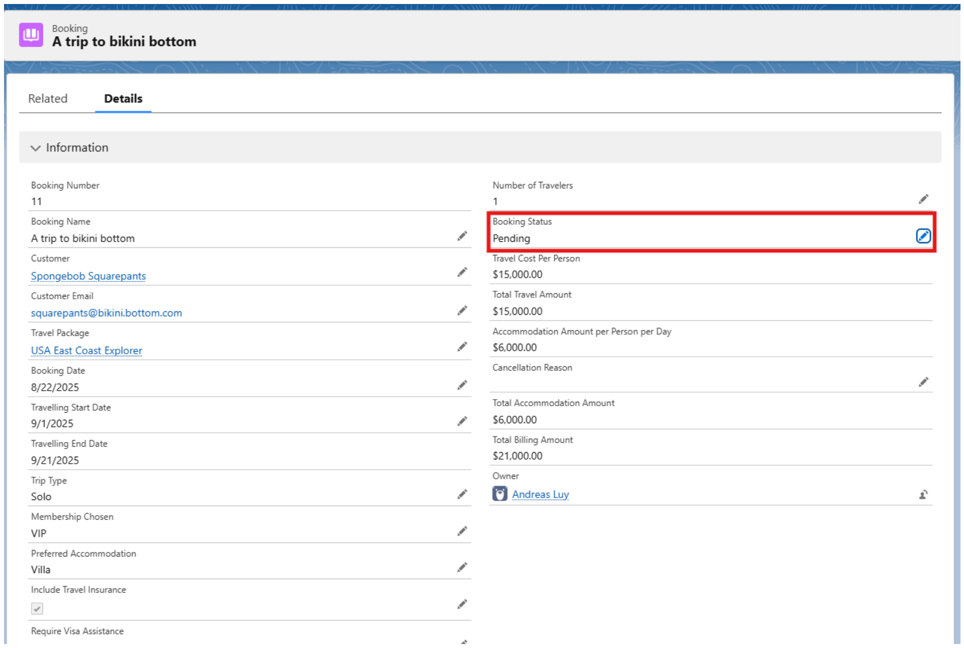

Tours & Travels CRM
Booking Payment
a04gL000009Lpf3
New Contact
Edit
New Opportunity
Related
Details
Payment Number
12
Payment Method
Customer Name
Spongebob Squarepants
Payment Reference Number
Booking
A trip to bikini bottom
Payment Mode Details
Payment Date
8/22/2025
Payment Receipt Sent
Total Bill Amount
$21,000.00
Owner
Andreas Luy
Payment Status
Pending
Notes
Booking Payment Name
a04gL000009Lpf3
Last Modified By
Andreas Luy, 8/24/2025, 4:58 AM
Activity
Upcoming & Overdue
No activities to show.
Get started by sending an email, scheduling a task, and more.
No past activity. Past meetings and tasks marked as done show up here.

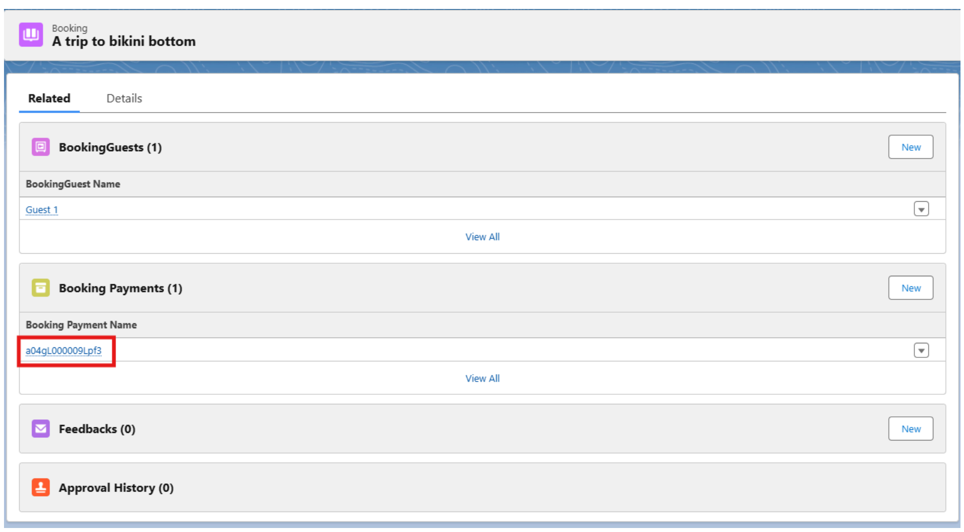
Activity 3: Test Case 3

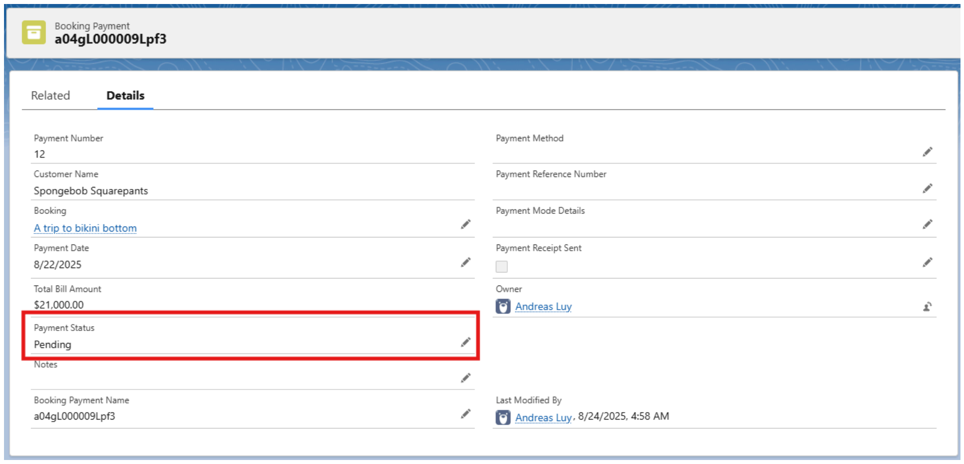
Payment Status Update and Email Notification

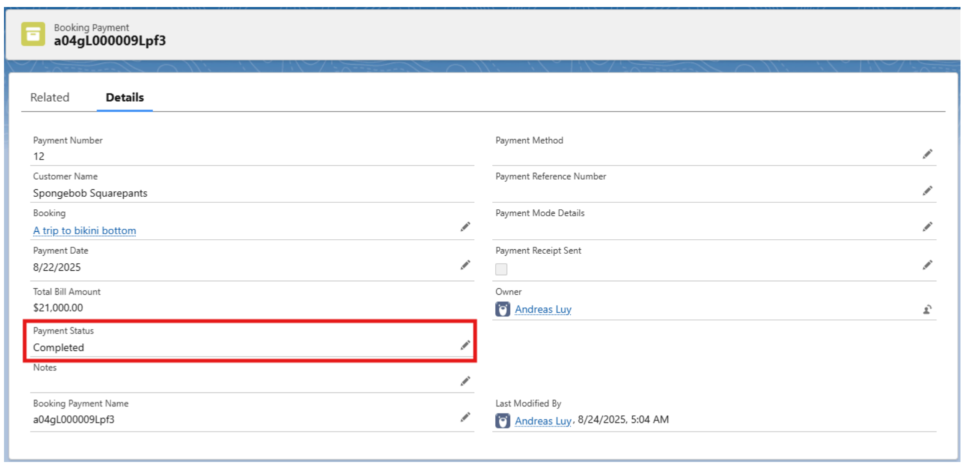
Test Case	**Verify whether the Booking Status is Confirmed in Booking Object when Payment Status field is updated to completed in Booking Payment Object. And also verify whether the customer received the mail regarding Booking confirmation and payment completed**
Environment	Includes ● OS: Windows 11 ● Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** ● Navigate to the Booking Payment object page. ● Update Payment Status field to Completed. ● Click Save
Expected Result	● The Booking Payment record should be Updated successfully. ● The Booking record should be Updated successfully and Booking Status must be Confirmed. ● The customer must receive the mail regarding Booking Confirmation and Payment completion
Status	**PASSED**

Test Evidences

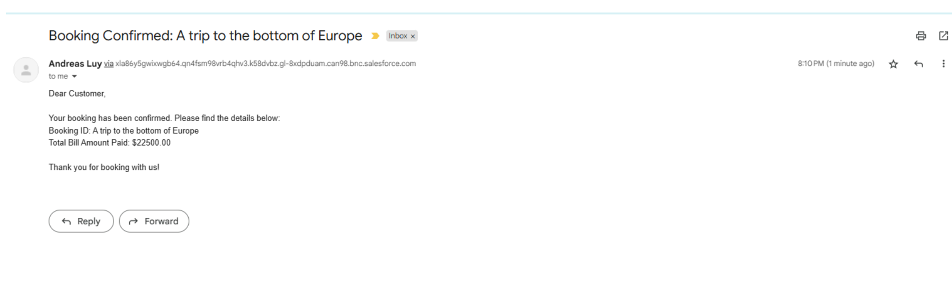




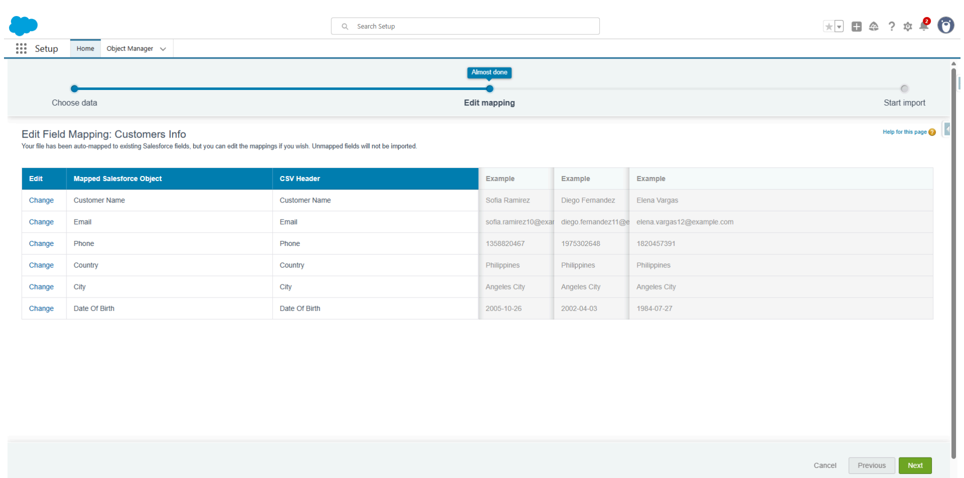


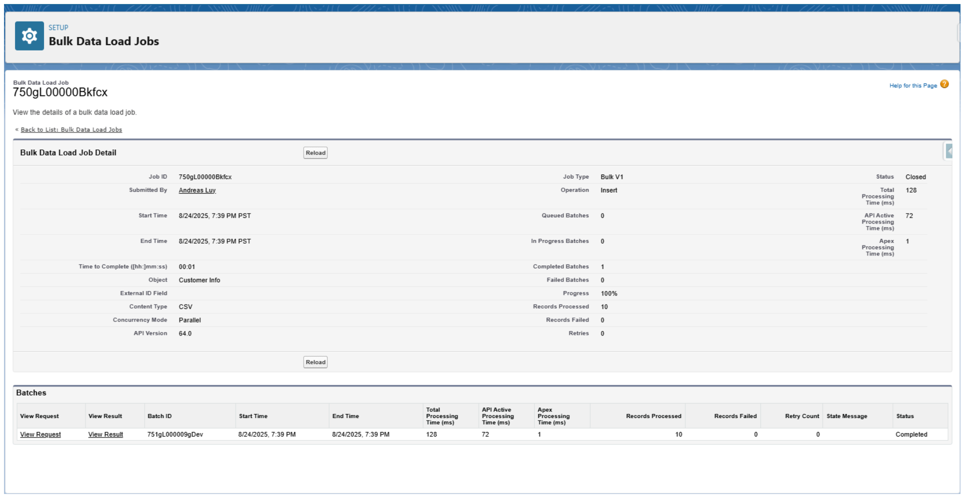


Email Received

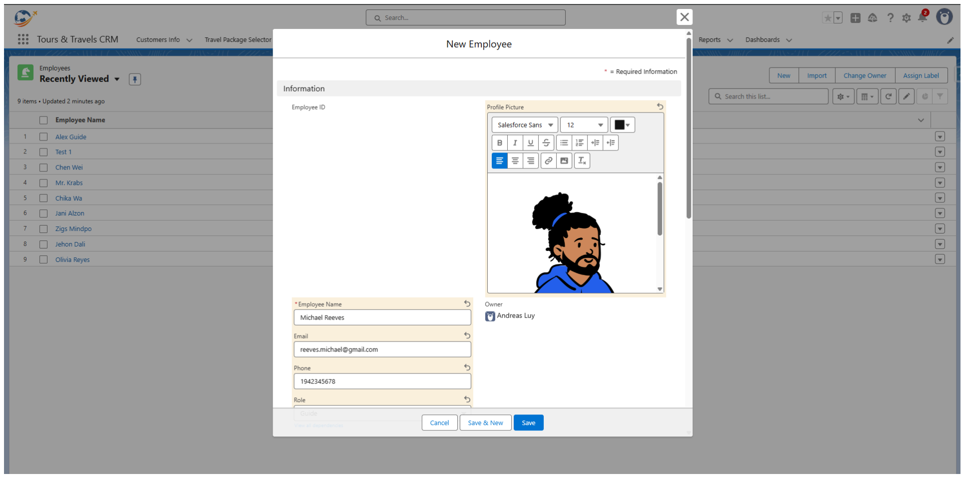


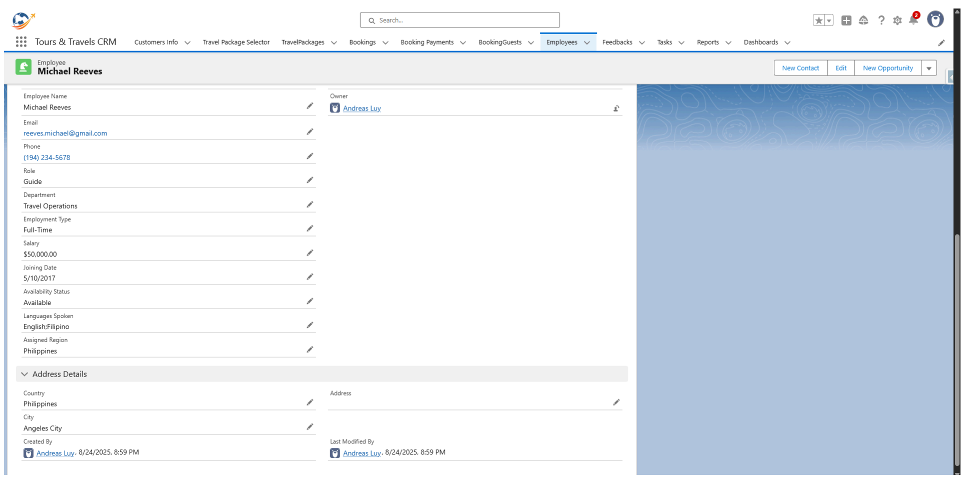
Activity 4: Test Case 4: Other Test Cases

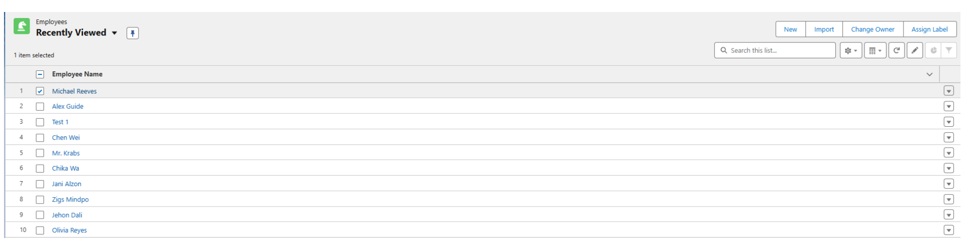
<table>
<tr><td>Test Case</td><td>Import Customers via Data Import Wizard</td></tr>
<tr><td>Environment</td><td>Includes
• OS: Windows 11
• Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)</td></tr>
<tr><td>Steps to Reproduce</td><td>Test Steps:
• Setup → Data Import Wizard
• Choose Object: Customer Info
• Upload CSV
• Map fields
• Start Import → Review results</td></tr>
<tr><td>Expected Result</td><td>• 10 records inserted with no invalid rows</td></tr>
<tr><td>Status</td><td>PASSED</td></tr>
<tr><td>Test Evidences</td><td>
</td></tr>
</table>



Test Case	**Create Employee Record**
Environment	Includes ● OS: Windows 11 ● Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** ● Go to Employees tab → Click New ● Enter details: Name, Role = Guide, Availability Status = Available ● Save
Expected Result	● The employee record is created successfully and appears in the Employees list view.
Status	**PASSED**

<table>
<tr>
<td>Test Evidences</td>
<td>


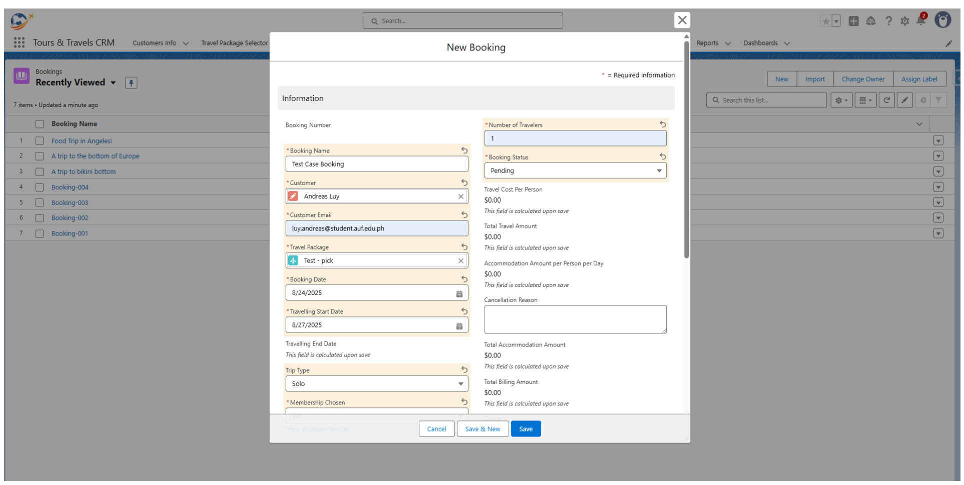


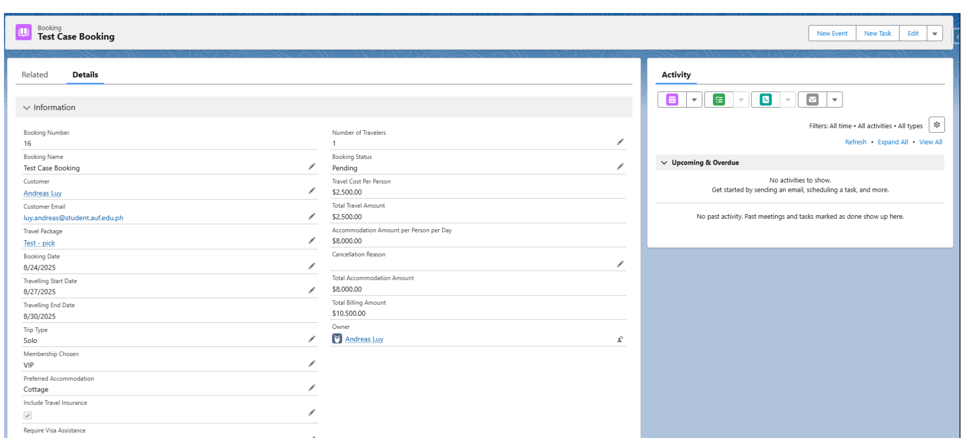


</td>
</tr>
</table>

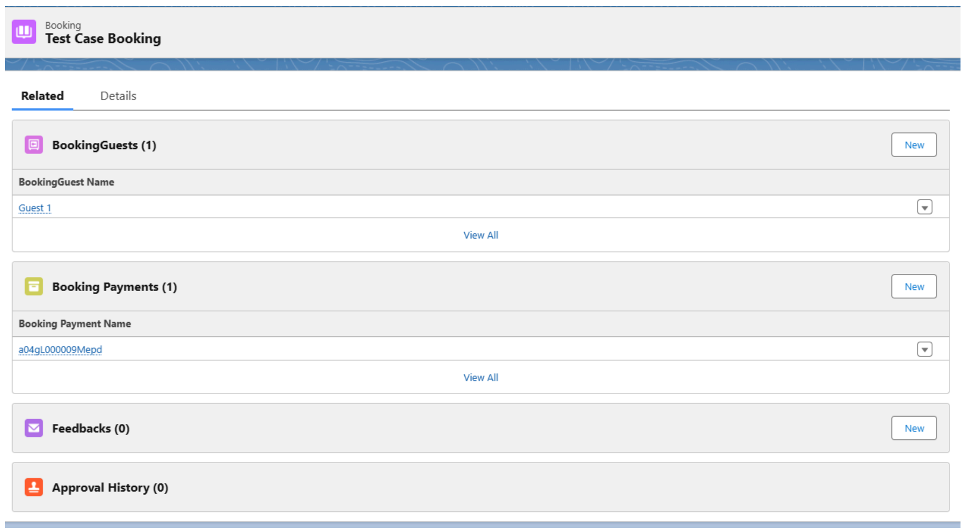
Test Case	**Create Booking & Auto-Generate Related Records**
Environment	Includes ● OS: Windows 11 ● Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)

<table>
<tr><td>Steps to Reproduce</td><td>Test Steps:

Go to Bookings tab → Click New
Select a Customer, Travel Package, Guide Assigned (Employee), enter # of Travelers
Save
</td></tr>
<tr><td>Expected Result</td><td>
Booking record is created.
Related Booking Payment record auto-created with status “Pending”.
Booking Guests auto-created based on # of Travelers.
</td></tr>
<tr><td>Status</td><td>PASSED</td></tr>
<tr><td>Test Evidences</td><td>


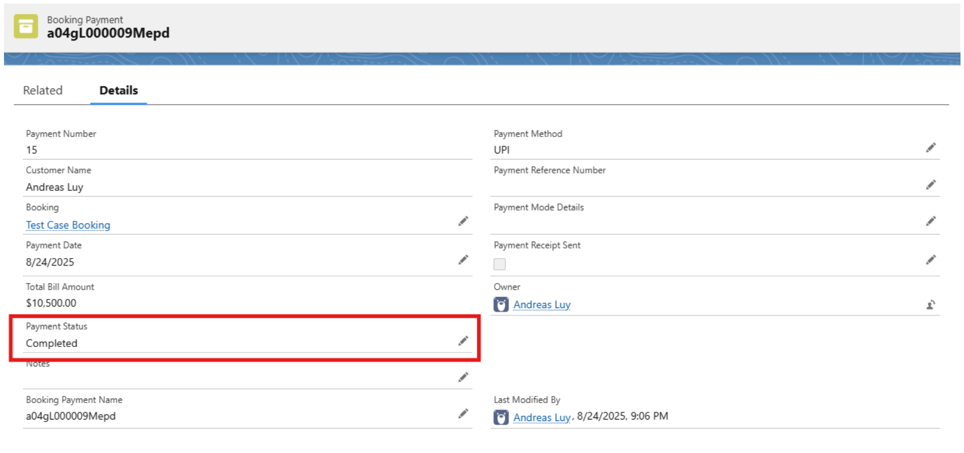


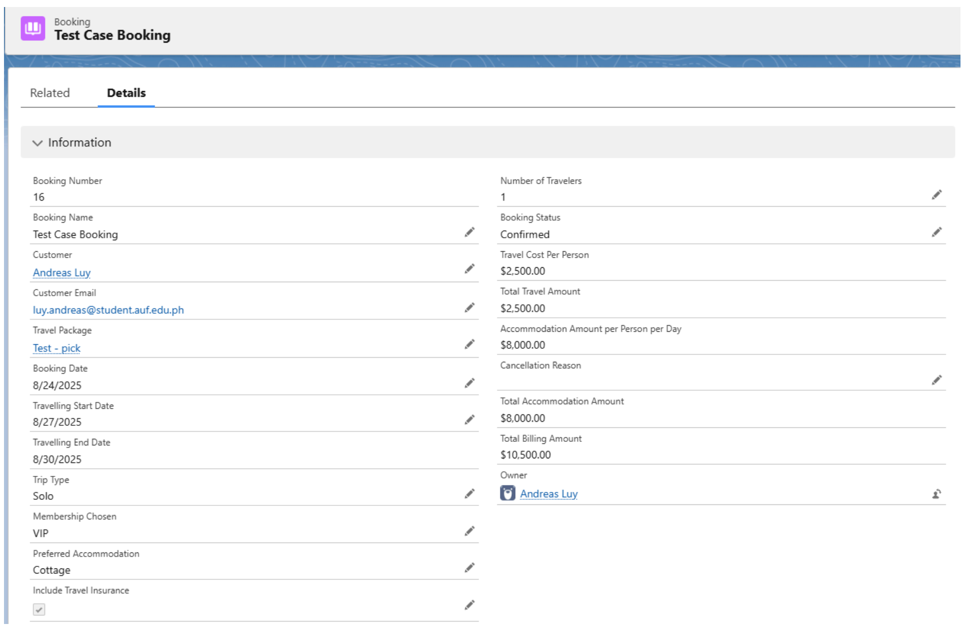
</td></tr>
</table>

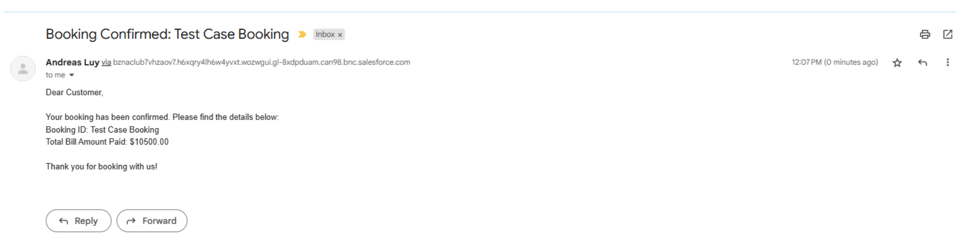
	

Test Case	**Update Payment Status → Email Notification**
Environment	Includes • OS: Windows 11 • Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** • Open a Booking Payment record • Change status from “Pending” → “Completed” • Save
Expected Result	• Payment record updated. • Email notification sent to the Customer’s email address
Status	**PASSED**

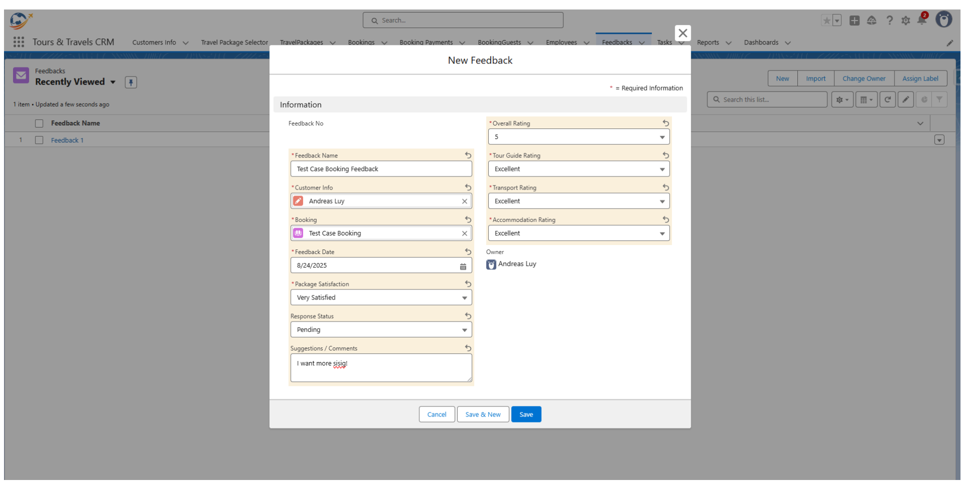
Test Evidences

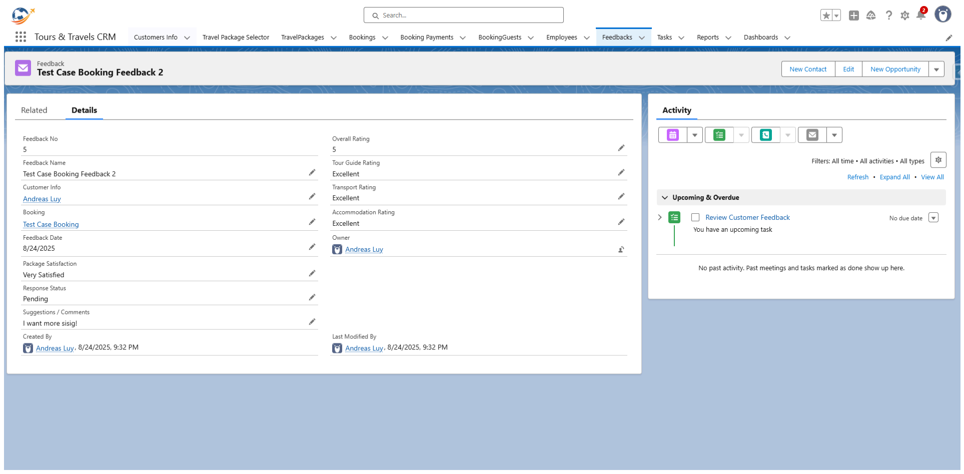




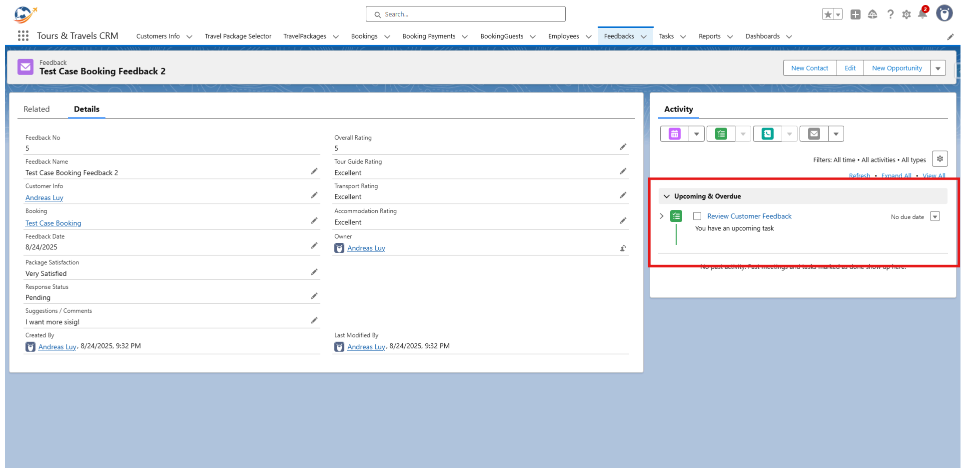
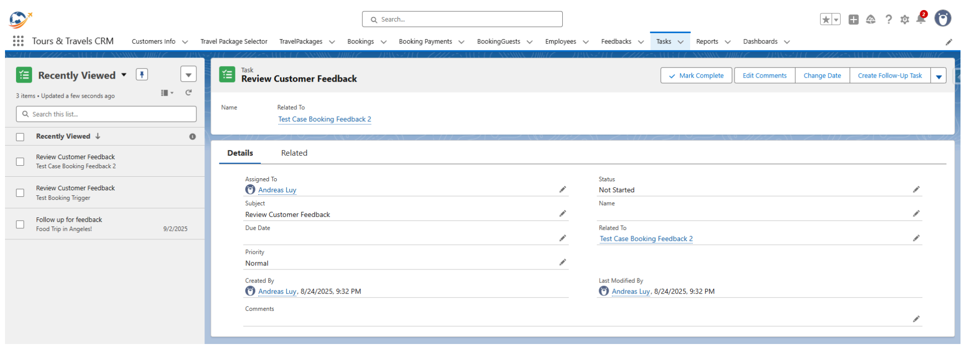


Test Case	**Submit Feedback**

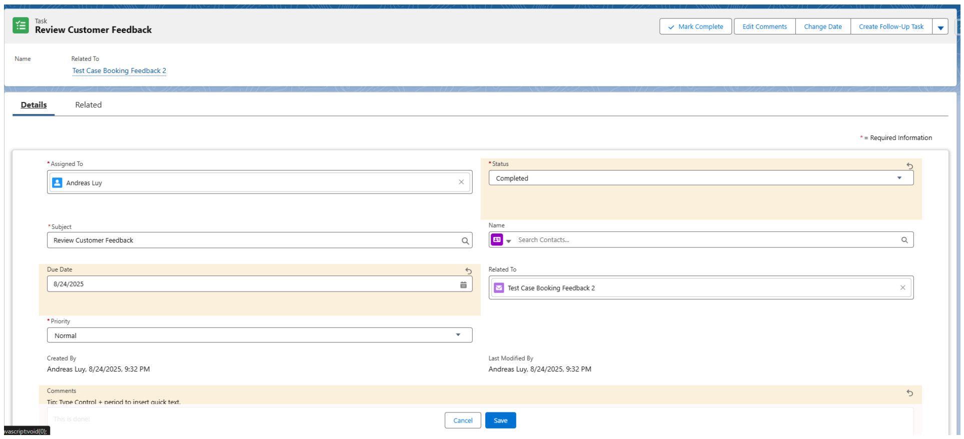
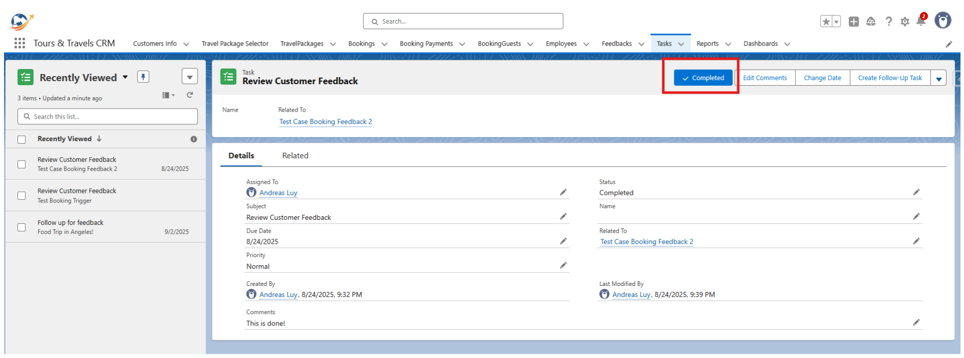
Environment	Includes ● OS: Windows 11 ● Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** ● Go to Feedbacks tab → Click New Feedback ● Select Customer, Booking, enter Feedback details + rating ● Save
Expected Result	● Feedback record created and linked to both Customer Info and Booking.
Status	**PASSED**
Test Evidences	

Test Case	**Auto-Generate Task after new Feedback record**
Environment	Includes • OS: Windows 11 • Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** • Go to Feedbacks tab → Click the recently created Feedback • Verify that there's an upcoming task on the right column
Expected Result	• An upcoming task is created after the feedback a new Feedback record is submitted.
Status	**PASSED**

Test Evidences	 

Test Case	**Verify Task is completed**
Environment	Includes ● OS: Windows 11 ● Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** ● Go to the Tasks tab, click the latest task record. ● Click edit and fill in the necessary details ● Click Save ● Check if the Task is marked as completed

Expected Result	• The upcoming task created will be marked as Completed
Status	**PASSED**
Test Evidences	 

Test Case	**Verify Task is completed**
Environment	Includes • OS: Windows 11 • Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** • Go to the Feedbacks tab

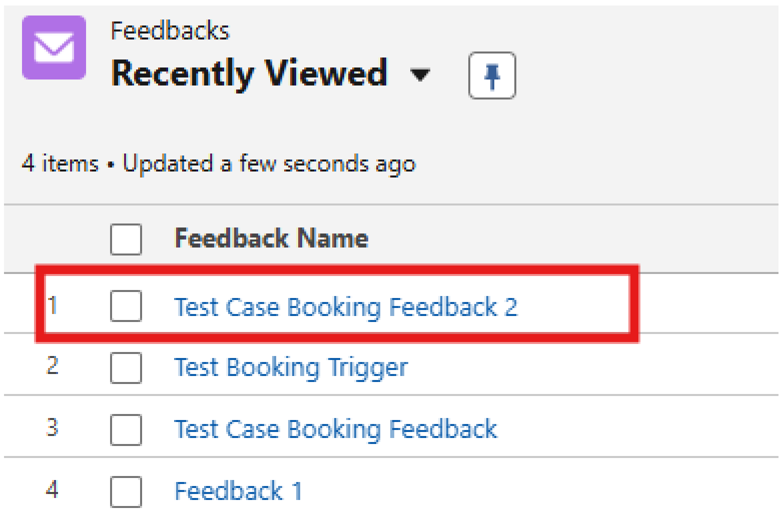
<table>
<tr><td></td><td>

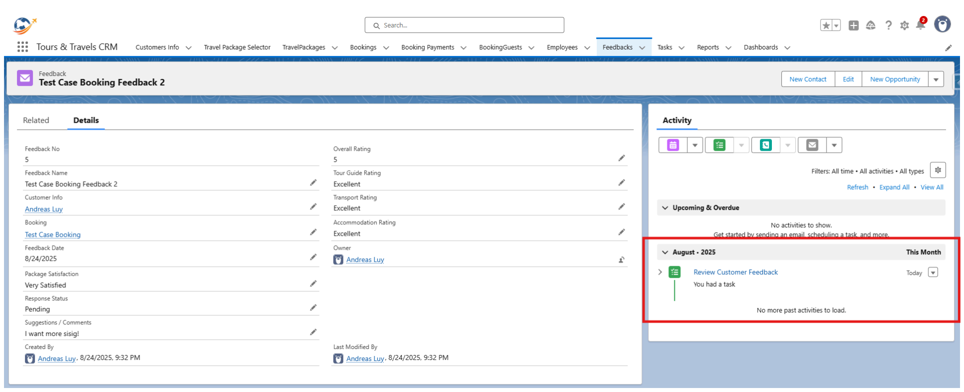
- Click the Feedback record related to it - "Test Case Booking Feedback 2"

</td></tr>
<tr><td>Expected Result</td><td>

- The upcoming task's status is updated and completed.

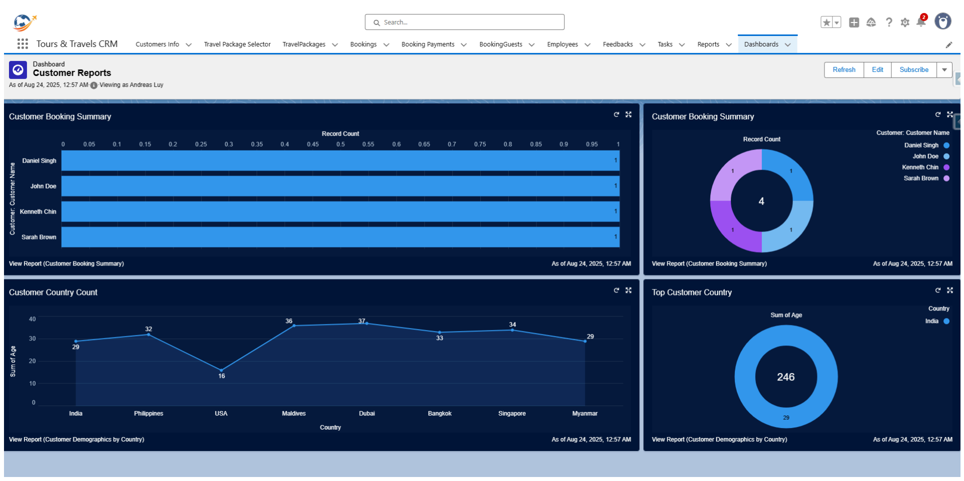
</td></tr>
<tr><td>Status</td><td>PASSED</td></tr>
<tr><td>Test Evidences</td><td>

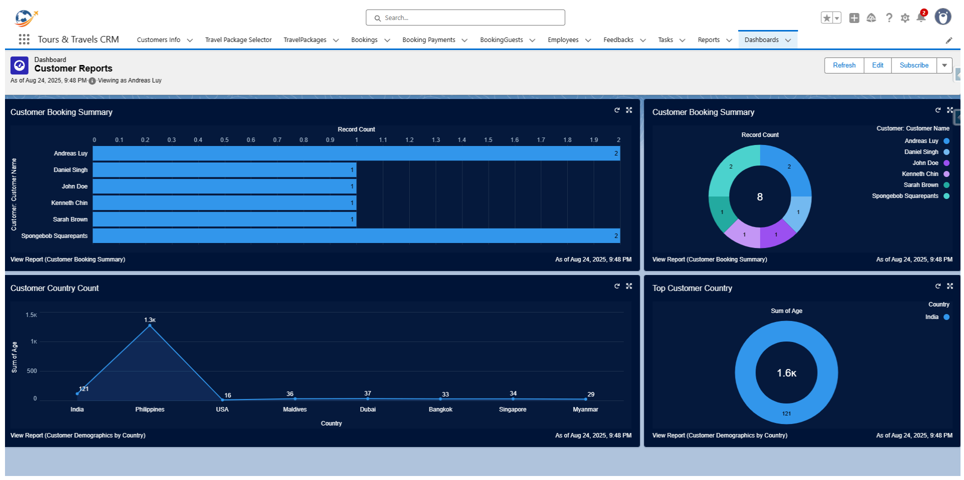




</td></tr>
</table>

Test Case	**Rechecking the Dashboard**
Environment	Includes • OS: Windows 11

	• Browser: Google Chrome Version 139.0.7258.129 (Official Build) (64-bit)
Steps to Reproduce	**Test Steps:** • Go to the Dashboards Tab • Click the created Dashboards from the recent milestones • Refresh the charts to see if the data from the reports are up to date.
Expected Result	• The charts show the up-to-date data from the created reports
Status	**PASSED**
Test Evidences	Before  After Refreshing



For this activity, I created and executed multiple test cases to validate the end-to-end functionality of the Tours & Travels CRM application. The tests covered both record creation across different modules (Customer Info, Travel Packages, Employees, Bookings, Booking Payments, and Feedback) as well as system behaviors such as email notifications, validation rules, automation, and dashboard refreshes.

The results confirmed that key workflows function as expected:

- When a customer makes a booking, the system automatically generates related payment and guest records.
- Updating a payment status to Completed triggers an email confirmation to the customer and updates the booking status.
- Submitting customer feedback links the record to both the booking and the customer, while also auto-generating a follow-up Task for employees.
- Dashboards and reports accurately refresh to display the latest operational data.

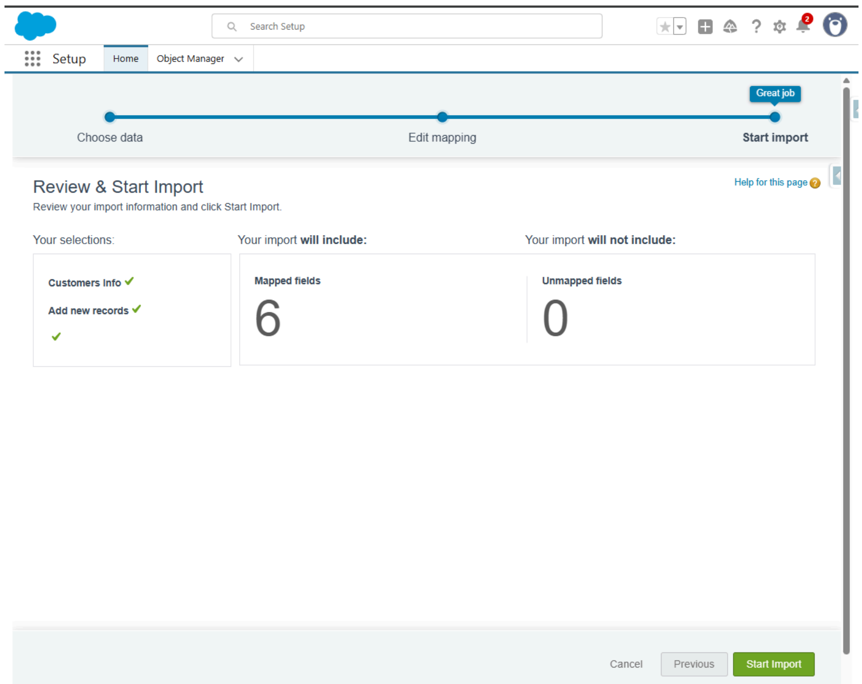
In a real-world scenario, the test cases I covered will ensure that core business processes — from booking to payment to customer feedback — are handled consistently and reliably. This can reduce human error, enforces validation rules, prevents incomplete data entry, ensuring that the employees are notified of follow-up actions through the use of automation.

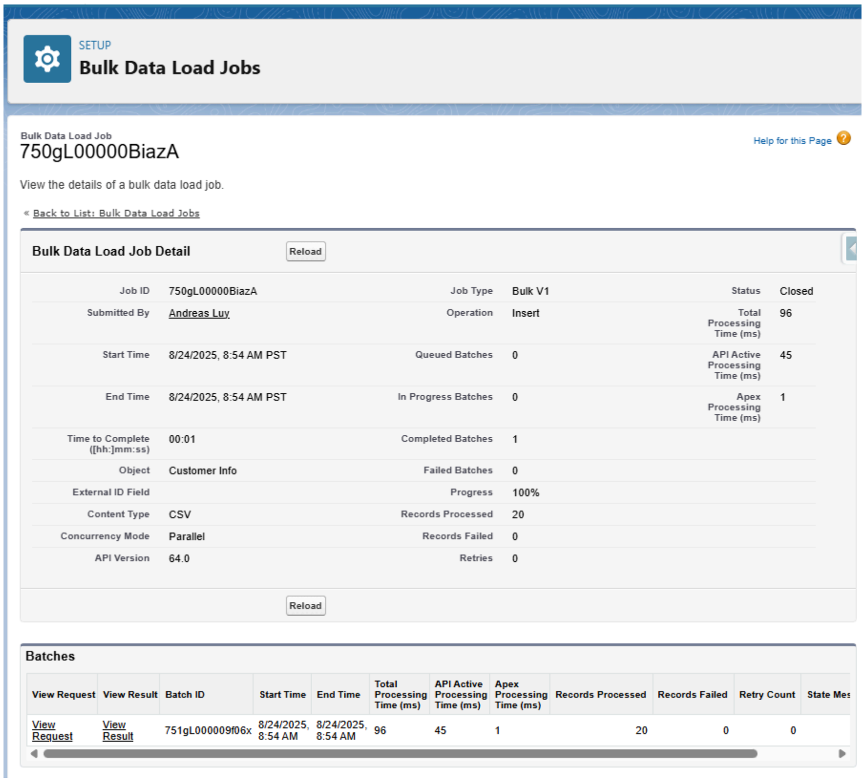
Overall, this milestone emphasized how structured testing can strengthen the system’s reliability, build Users confidence, and support the smooth handling of day-to-day operations like in a travel and tourism business.

Milestone 29: Data Import Wizard

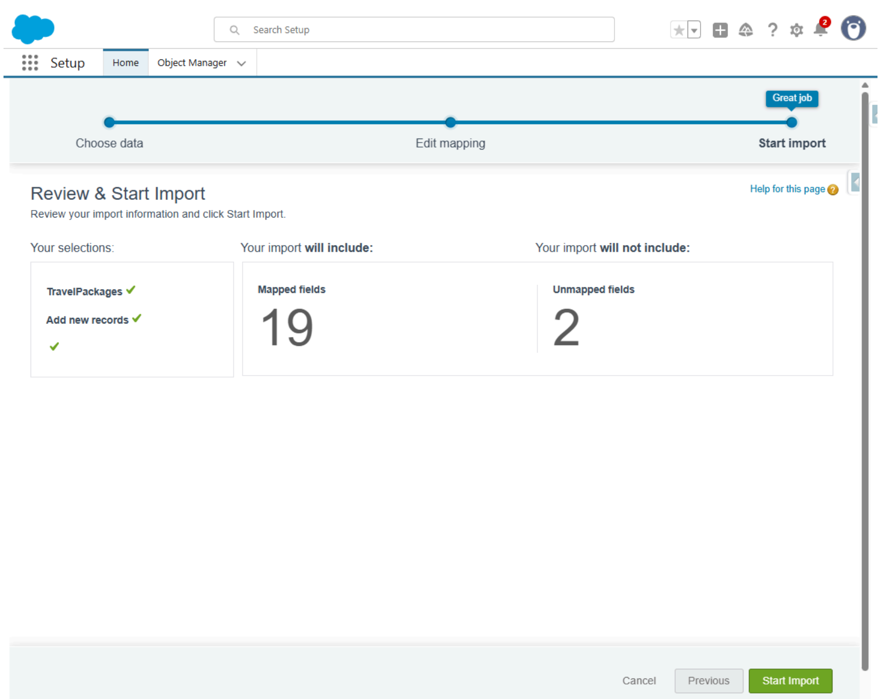
Activity 1: Importing Records into Customer Info, TravelPackage and Employee Objects

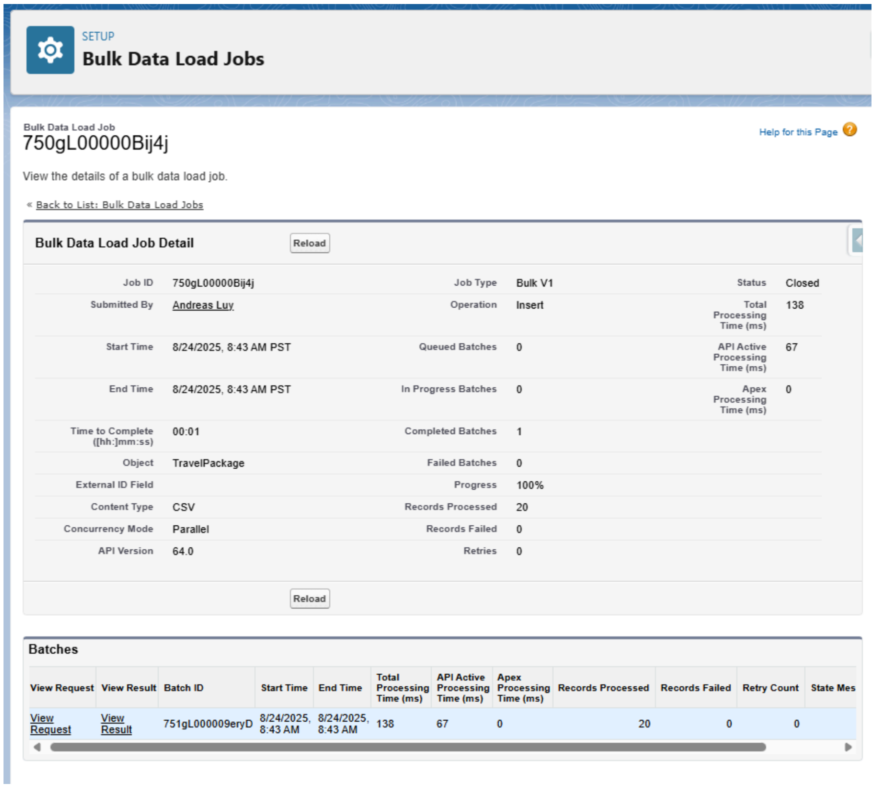
Customer Info



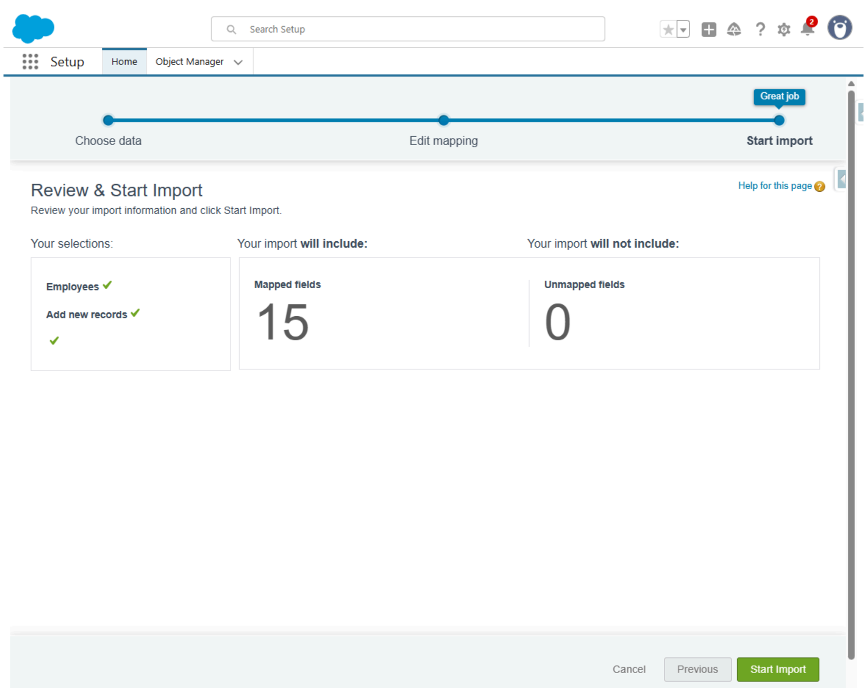


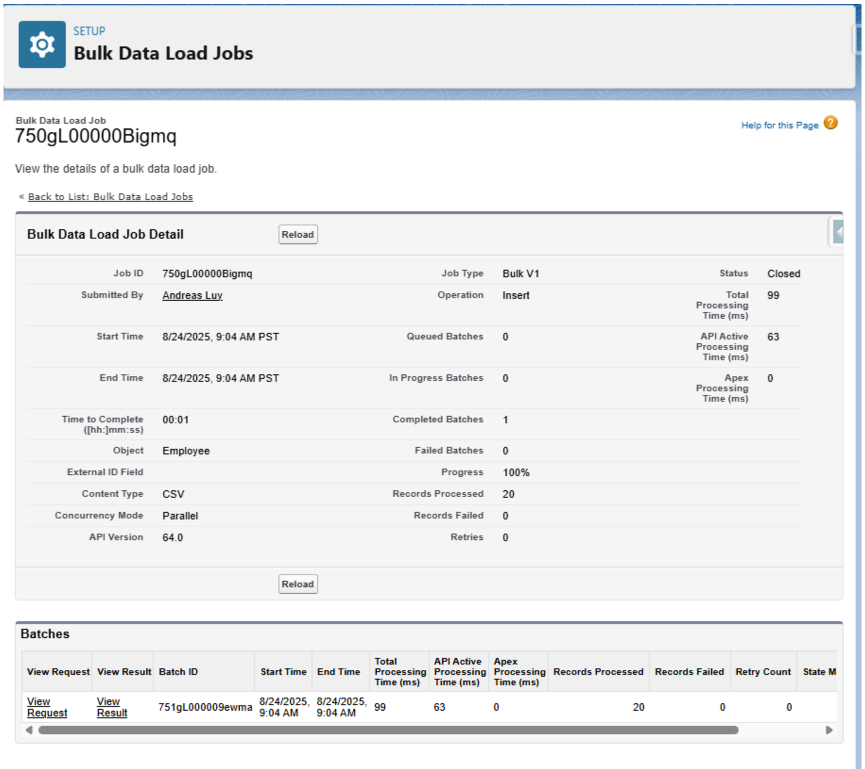
Travel Package




SETUP
Bulk Data Load Jobs
Bulk Data Load Job
750gL00000Bij4j
Help for this Page
View the details of a bulk data load job.
« Back to List: Bulk Data Load Jobs
Bulk Data Load Job Detail
Reload
Job ID 750gL00000Bij4j
Job Type Bulk V1
Status Closed
Submitted By Andreas Luy
Operation Insert
Total Processing Time (ms) 138
Start Time 8/24/2025, 8:43 AM PST
Queued Batches 0
API Active Processing Time (ms) 67
End Time 8/24/2025, 8:43 AM PST
In Progress Batches 0
Apex Processing Time (ms) 0
Time to Complete ([hh:]mm:ss) 00:01
Completed Batches 1
Object TravelPackage
Failed Batches 0
External ID Field
Progress 100%
Content Type CSV
Records Processed 20
Concurrency Mode Parallel
Records Failed 0
API Version 64.0
Retries 0
Reload
Batches
View Request | View Result | Batch ID | Start Time | End Time | Total Processing Time (ms) | API Active Processing Time (ms) | Apex Processing Time (ms) | Records Processed | Records Failed | Retry Count | State Mes
View Request | View Result | 751gL000009eryD | 8/24/2025, 8:43 AM | 8/24/2025, 8:43 AM | 138 | 67 | 0 | 20 | 0 | 0

Employee


Search Setup
Setup
Home
Object Manager
Great job
Choose data
Edit mapping
Start import
Review & Start Import
Help for this page
Review your import information and click Start Import.
Your selections:
Your import will include:
Your import will not include:
Employees
Add new records
Mapped fields
15
Unmapped fields
0
Cancel
Previous
Start Import



The Data Import Wizard is a simple and Users-friendly tool that allows the import of small to medium data volumes and provides field mapping to ensure data accuracy during the import process.

Importing the records for the three objects was one of the most challenging parts of the project. Despite preparing the CSV files carefully, rechecking the fields correctly, I still ran into multiple issues where the data failed to record successfully. This milestone reminded me how important and strict data formatting and accuracy is on Salesforce. It tested my patience, but it was very rewarding once I saw the 20 records successfully inserted.

Phase 5: Deployment, Documentation & Maintenance

Milestone 30: Maintenance, Monitoring & Troubleshooting

UI Validation of Features

All the developed features were thoroughly validated using the Salesforce Users interface. Each tab - including the Customer Info, Employees, Travel Packages, Bookings, Payments, and Feedback were tested to ensure that the forms, required fields, lookups, and related lists functioned as expected. This confirmed that the Users can perform the core operations directly from the User Interface without encountering any navigation issues.

Functional Testing of Scenarios

The critical automation processes such as the Flows, Approval Processes, Workflow Rules, Process Builder actions, and Apex triggers were tested with realistic dummy data (to prevent data privacy concerns) inputs. For example, updating the Booking payment status to Completed will automatically trigger an email notification to the customer and update the related Booking status to Confirmed. These tests have validated that all backend logic were executed as intended.

End-to-End Business Flow Verification

The complete business flow was executed to verify smooth data transitions:

1. Customer creation (via manual entry or Data Import Wizard).
2. Booking creation linked to Customer, Travel Package, and Employee Guide.
3. Booking Payment auto-generation with status Pending.
4. Booking Guests auto-creation based on traveler count.
5. Feedback submission linked to both Booking and Customer.
6. Task auto-generation assigned to the responsible Employee upon Feedback creation.

This confirmed that the Tours & Travels CRM operates as a fully integrated system.

Data Accuracy Checks

Validation focused on the integrity of each relationship between the objects. The Bookings were properly linked to the Customers Info, Travel Packages, and Employees, while the Booking Payment and Booking Guest accurately reflected the details of the parent Booking. The validation rules prevented incomplete or invalid data entry, such as any attempts to save a Booking without specifying the number of travelers

Monitoring Scheduled Jobs and Flows

A review of Scheduled Flows, Apex Jobs, and Queueable Batch processes confirmed that all automated jobs executed successfully without runtime failures. Exception logs were examined to ensure that recurring operations — such as batch updates or scheduled triggers — completed as expected.

Debug Logs Review for Apex & Workflows

Debug logs were generated and analyzed for Apex Triggers and Classes, particularly those handling Booking and Payment logic. The logs confirmed that records were processed without recursion, infinite loops, or governor limit issues. This step also helped identify performance metrics, ensuring optimized code execution.

Error Handling and User Notifications

Both system-generated and custom error messages were tested. Validation errors (e.g., preventing duplicate Customer creation or bookings with zero travelers) were displayed in a clear, Users-friendly format. Additionally, email notifications were verified to alert customers regarding payment completion and booking confirmation. This ensures that Users are guided effectively when issues arise.

Security & Access Testing

Profiles, Permission Sets, and Role Hierarchies were reviewed to ensure compliance with data visibility rules. For instance, Travel Agents had access to Customer and Booking records but not to Finance-only Payment configurations, while Administrators maintained full system access. These checks confirmed that the principle of least privilege was consistently enforced.

Duplicate & Validation Rule Checks

Duplicate Rules were tested by attempting to create Customer Info records with identical email addresses, and the system correctly flagged the duplication. Validation Rules were also validated, such as ensuring that the *Number of Travelers* field must be greater than zero for any Booking. These controls ensure data cleanliness and adherence to business rules.

Report and Dashboard Accuracy

Reports and dashboards were reviewed for accuracy and freshness. For example, the Monthly Revenue Dashboard was refreshed after adding new Bookings and Payments, and the updated data reflected correctly in the charts. This ensures that decision-makers can rely on dashboards for real-time insights.

Dynamic Form Behavior

Dynamic Forms were tested to ensure contextual display of fields. For example, certain fields in the Feedback object only appeared when specific conditions were met (e.g., *Feedback Rating* field displayed only when *Feedback Type = Survey*). This improved the Users experience by reducing clutter and presenting only relevant fields.

Cross-Browser and Device Testing

The Tours & Travels CRM app was tested across different browsers such as Google Chrome, Microsoft Edge, and Brave. The functionality and responsiveness were consistent across different browsers using the Development Tools (F12, CTRL+SHIFT+I, or Right Click and Inspect). This ensured that the Tours & Travels CRM Application was accessible to Users regardless of their preferred device or browser.

Milestone 31: Documentation

Project Demo Video Presentation

Salesforce Capstone Recording and Documentation

Summary & Conclusion

The Tours & Travels CRM capstone project successfully showcased the design development, testing, and the maintenance of a custom Salesforce application tailored for the Travel and Tourism industry. From the initial creation of custom objects to the integration of automation and dashboards, the project covered the entire CRM lifecycle. Each of the phases on the capstone project contributed to a scalable and user-friendly system capable of simulating real world travel agency operations.

The project integrated the core Salesforce features including the use of custom objects, validation rules, approval processes, roles, reports, dashboards, flows, workflows, process builders, and Apex triggers. The automation ensured that the critical processes such as auto-creating Booking Payments, Booking Guests, and Tasks from Feedback were handled seamlessly. Security and data integrity were maintained through Profiles, permission sets, validation rules, and duplicate rules, which ensured a reliable business operation.

Testing was a key focus of the project. Through UI validation, automation functionality checks, end-to-end business flow verification, and dashboard refresh validation, the Tours & Travels CRM App was proven to be consistent and reliable. Evidence from the test cases showed that the system not only worked correctly but also mirrored real-world workflows, such as booking management, payment tracking, customer feedback collection, and employee assignment.

Practices for maintenance and monitoring were also considered, including debug log reviews, scheduled job monitoring, and cross-browser testing, ensuring that the Tours & Travels CRM could be sustained and scaled in a real production environment.

The system successfully simulated a travel agency ecosystem — connecting customers, employees, bookings, packages, payments, feedback, and tasks into one cohesive workflow. The result was a CRM that is not only functional, but also extensible, adaptable, and aligned with the daily operations of a tourism business.

What I learned

The capstone project for Tours & Travels gave me the opportunity to experience building an enterprise-grade Salesforce application from the ground up.

I learned how to:

- Design data models that can capture real-world relationships between customers, bookings, packages, and payments.
- Use declarative tools (Flows, Process Builder, Validation Rules, Dashboards) to automate processes without code.

- Apply programmatic features like Apex triggers and asynchronous jobs to extend functionality.

- Debug and troubleshoot issues using Salesforce's debug logs, test classes, and error handling tools.

- Ensure data accuracy, system reliability, and user accessibility through rigorous testing and security configurations.

- Align technical design with real-world travel agency operations, reducing manual effort and ensuring smooth workflows.

Overall, the Tours & Travels CRM capstone project reflects not only technical knowledge of Salesforce but also the ability to think like a business stakeholder, focusing on efficiency, usability, and scalability. It stands as both a professional portfolio piece and a learning journey into building end-to-end CRM solutions.

Future Enhancements

While the current Tours & Travels CRM successfully automates the processes for Bookings, Payments, and Customer management, there are opportunities for further improvements. Possible enhancements include:

- Integration with Third Party-APIs: Connecting with popular payment gateways (e.g. Maya, Gcash) and travel platforms (e.g. Amadeus) for real-time package availability and seamless transactions.

- Mobile application support: Building a Salesforce Mobile Application extension so customers and agents can manage bookings and payments directly from their mobile devices.

- AI-powered Recommendations: Leveraging Salesforce Einstein to suggest a personalised travel package to customers based on past bookings, preferences, and seasonal trends.

- Multi-Language & Multi-Currency support: Enhancing the CRM's global usability by supporting multiple languages and currencies for international customers.

- Loyalty & Rewards Program: Implement a pointing-based reward system where frequent travelers can earn discounted price or special offers.

- Social Media Integration: Connecting the CRM with popular social media platforms (e.g. Facebook, Instagram) to capture leads from ads and social media inquiries directly into Salesforce.

- Sustainability Tracking: Allowing agencies to showcase eco-friendly travel packages and track carbon footprints for environmentally conscious travelers.

End