Capstone Project Report
On
Travel Application
And
Code Playground Application

Submitted by: Ravi Teja Kuchipudi

Name: Ravi Teja Kuchipudi
Batch Wipro Salesforce B13 (WIP-SF-13)
LMS Id: MGSA_519
Date: 28/02/2023
GitHub Link: Github Link - Ravi Teja Kuchipudi

Table of contents

Abstract

Salesforce is a comprehensive and flexible platform that enables businesses to manage their customer relationships, streamline their sales and marketing processes, and promote business growth.

There are several ways to adjust the user interface in your Salesforce org to meet your needs. Many customizations can be made through configuration menus and drag-and-drop page layout editors, all with no coding required. This project contains three modules. First module focuses on the non-coding approach to modifying the user interface, as you customize an app that tracks company travel.

Within the Salesforce Platform there are several features and tools that let you streamline and automate your business processes. As you discover in this project, the Salesforce Lightning Platform makes business process automation a snap and doesn't require any code. This Second module focuses on the Automation and Analytics Tools to enhance your custom App development.

Apex enables developers to build complex business processes, customize user interfaces, and integrations with third-party systems, when declarative tools aren't up to the task. This last module focuses on the coding approach to extend the functional capabilities of your custom App.

Introduction

Salesforce is a cloud-based Customer Relationship Management (CRM) platform that helps businesses manage and streamline their customer interactions. The platform offers a suite of tools that enable organizations to effectively manage their sales, marketing, customer service, and analytics functions all in one place.

With Salesforce, businesses can manage their entire customer lifecycle, from lead generation to customer retention, all while providing their employees with real-time access to customer data and insights. The platform offers a wide range of integrations with other software solutions, allowing businesses to easily connect Salesforce with their existing systems and workflows.

Figure 1: Salesforce Cloud Services (*Source*)

Talking about Travel Application and Code playground Application, Travel Application focuses on the non-coding approach to modifying the user interface, as you customize an app that tracks company travel and also focuses on the Automation and Analytics Tools to enhance your custom App development. By the end of the Travel Application , we will have a working prototype of the new travel approval application and also a working custom Travel Approval App with added Business logics and analytics to improve the user experience. Code Playground Application focuses on the coding approach to extend the functional capabilities of your custom App.

Flow of the Project

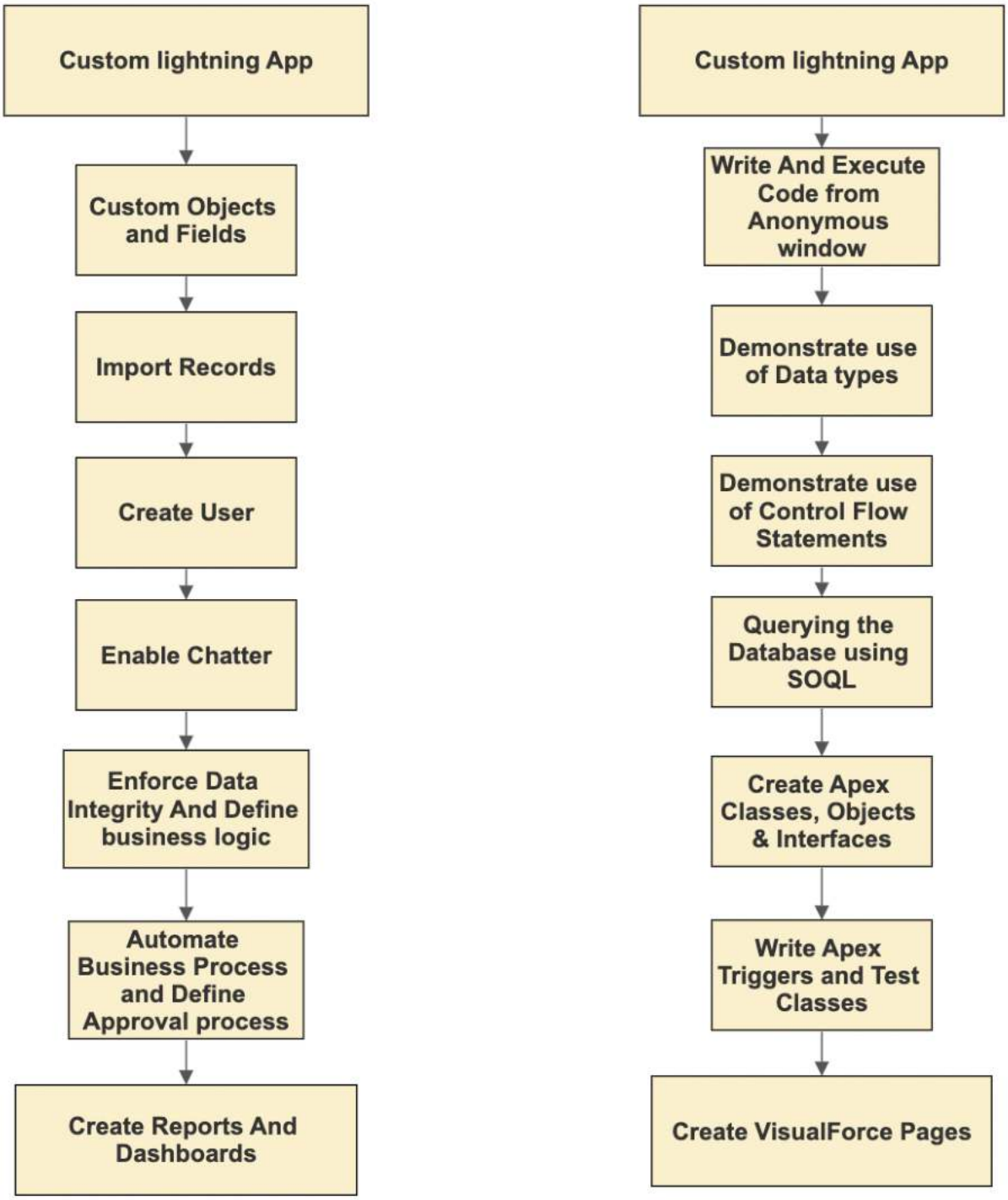

Figure 1: Flowcharts.

Software and Hardware Requirements

Software Requirements:

1. **Web browser:** Latest version of Google Chrome, Mozilla Firefox, Apple Safari or Microsoft Edge
2. **Operating System:** Windows 10, Mac OS X, or any Linux distribution that supports the latest versions of Firefox or Chrome
3. **Internet Connection:** 1 Mbps or faster

Hardware Requirements:

1. **Processor:** Minimum 2 GHz
2. **RAM:** Minimum 4 GB
3. **Storage Space:** Minimum 20 GB

Screen Shots

Module 1

Business Requirements:

- Set up a Lightning app to streamline the travel approval process.
- You need to create a custom application that meets these requirements: Build a Lightning app, add tabs, and customize page layouts.
- Create custom objects and fields for the app.
- Define relationships between objects.
- Import data.
- Create Travel Approvals, Travel Expenses & Users.
- Customize List views, Page Layouts, Related Lists.
- Setup Approval settings for User.
- Enable Chatter on the Travel Approval Object.

Exercise 1:

Step 1:

1. Create a new custom lightning App, name: Travel App.
2. Upload Travel.png as App image, Keep “Org Theme options” checked.
3. Navigation Style = ‘Standard Navigation’, leave rest as it is.
4. Select and add Chatter, Reports & Dashboards to the Navigation items.
5. User Profile = “System Administrator”.
6. Save & Finish.

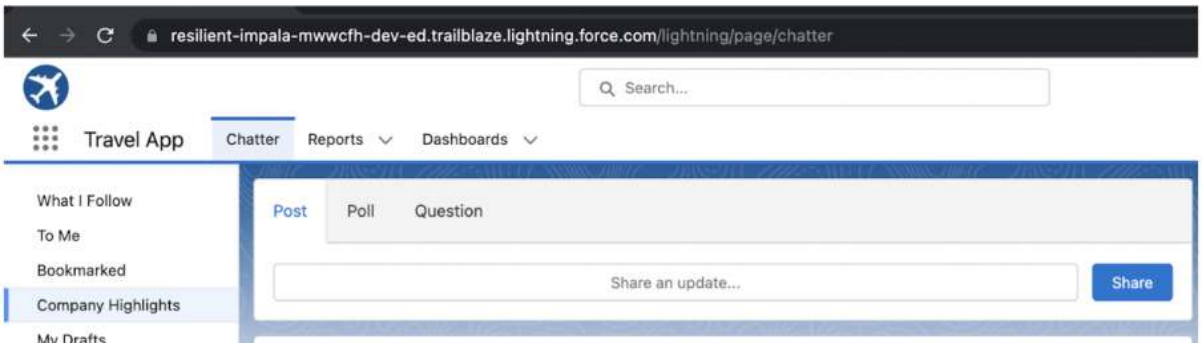

Figure 3

Step 2:

1. Create a Department custom object.
2. Label = “Department” ,Plural = “Departments” ,Data Type = “Text”
3. Select “Allow Reports”, “Allow Search”. Keep other options as it is.
4. Use the custom tab of your choice and Include it only for the Travel App.
5. Tab visibility = As it is, Save.

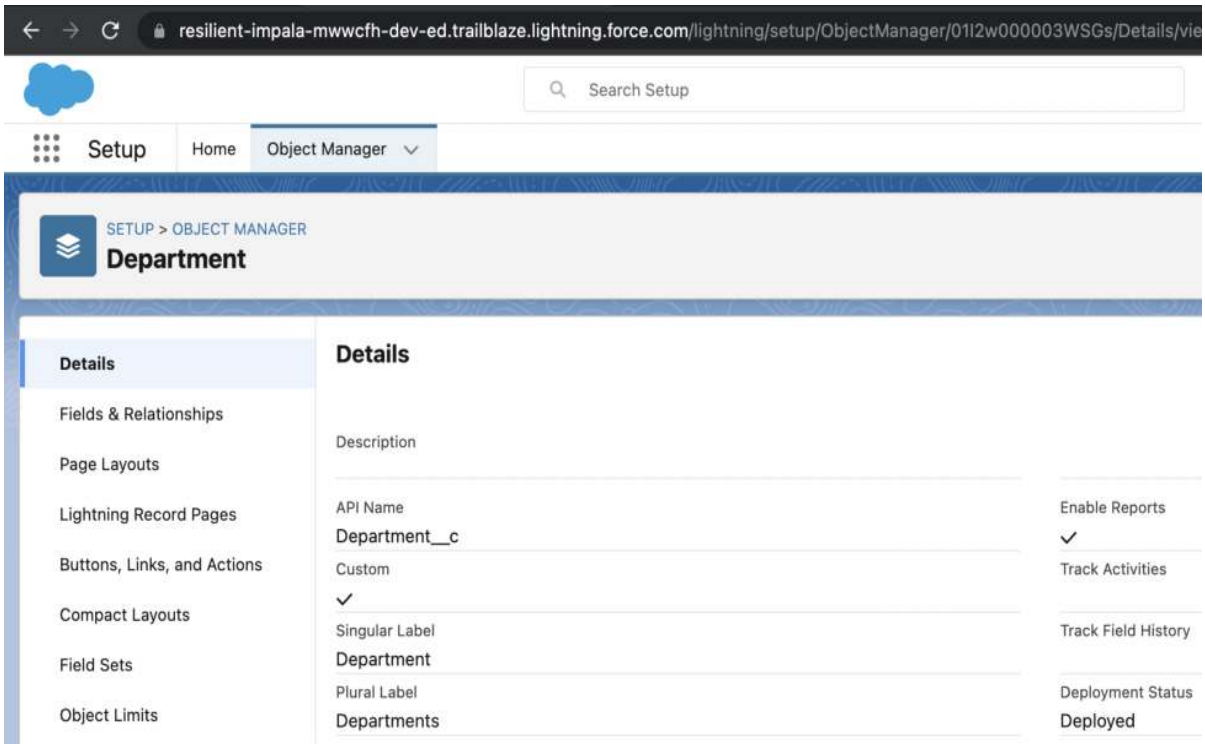

Figure 4

Step 3:

1. Department Code, Text, Length = 10, Required, Select Unique & Case sensitive.
2. Location, Picklist, Value: Kolkata, Delhi.
3. Department Type, Picklist, Values: Banking, Finance, Education, Energy, IT.
4. Create Field Dependency, Controlling field = Location, Dependent field = Department Type.

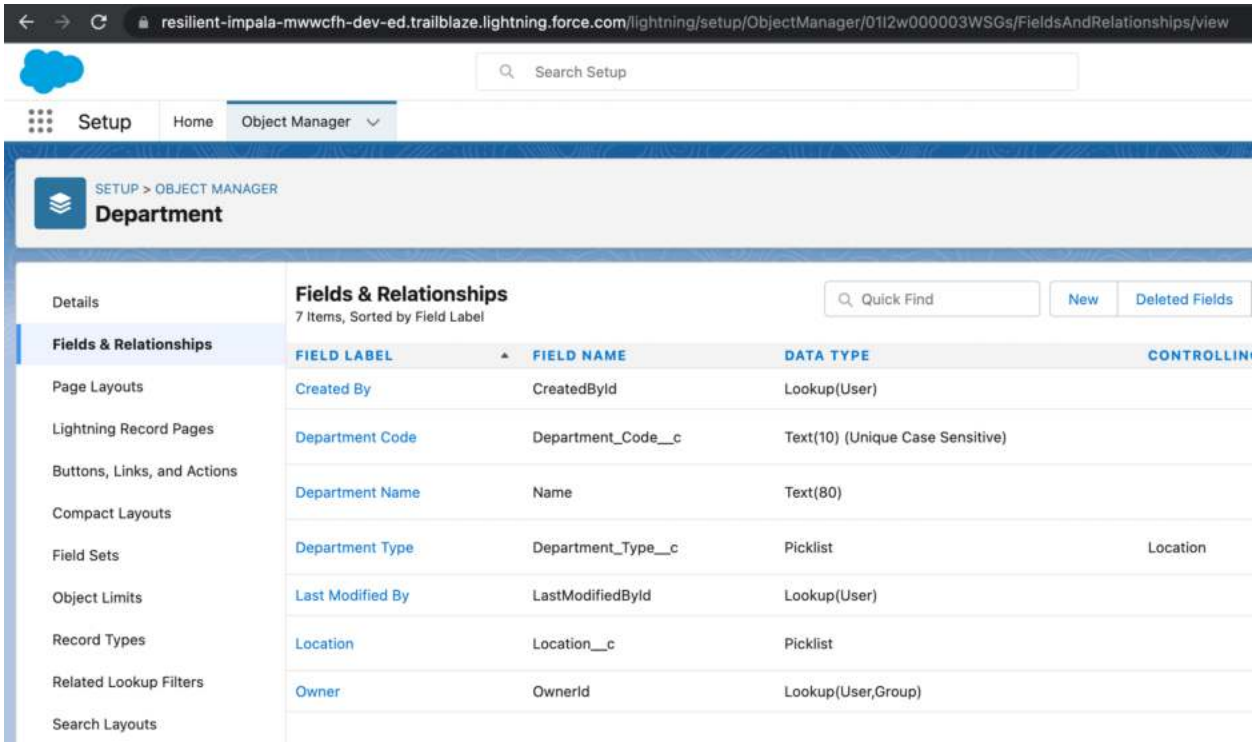

Figure 5

Step 4:

1. Create a Travel Approval Object.
2. Label = “Travel Approval”, Starting Number = 1., Plural = “Travel Approvals”, Object Name = “Travel_Approval”
3. Record Name = “Travel Approval #, Data Type = “Auto Number”, Display format “TA{00000}”.
4. Select additional options(Allow Reports, Allow Activities, Track Field History, Allow Search, Add Notes & Attachments.)
5. Create a custom Tab of your choice, Add the Tab only to the Travel App.
6. Create Custom fields:
    a. Purpose Of Trip, Text Area.
    b. Status, Picklist, Values = New, Submitted, Pending Approval, Approved, Rejected, Draft.
    c. Trip Start Date, Date.
    d. Trip End Date, Date.
    e. Out Of State, Checkbox.
    f. Destination State, Text, Length = 2.
    g. Department, Lookup, Related To = Department custom object.

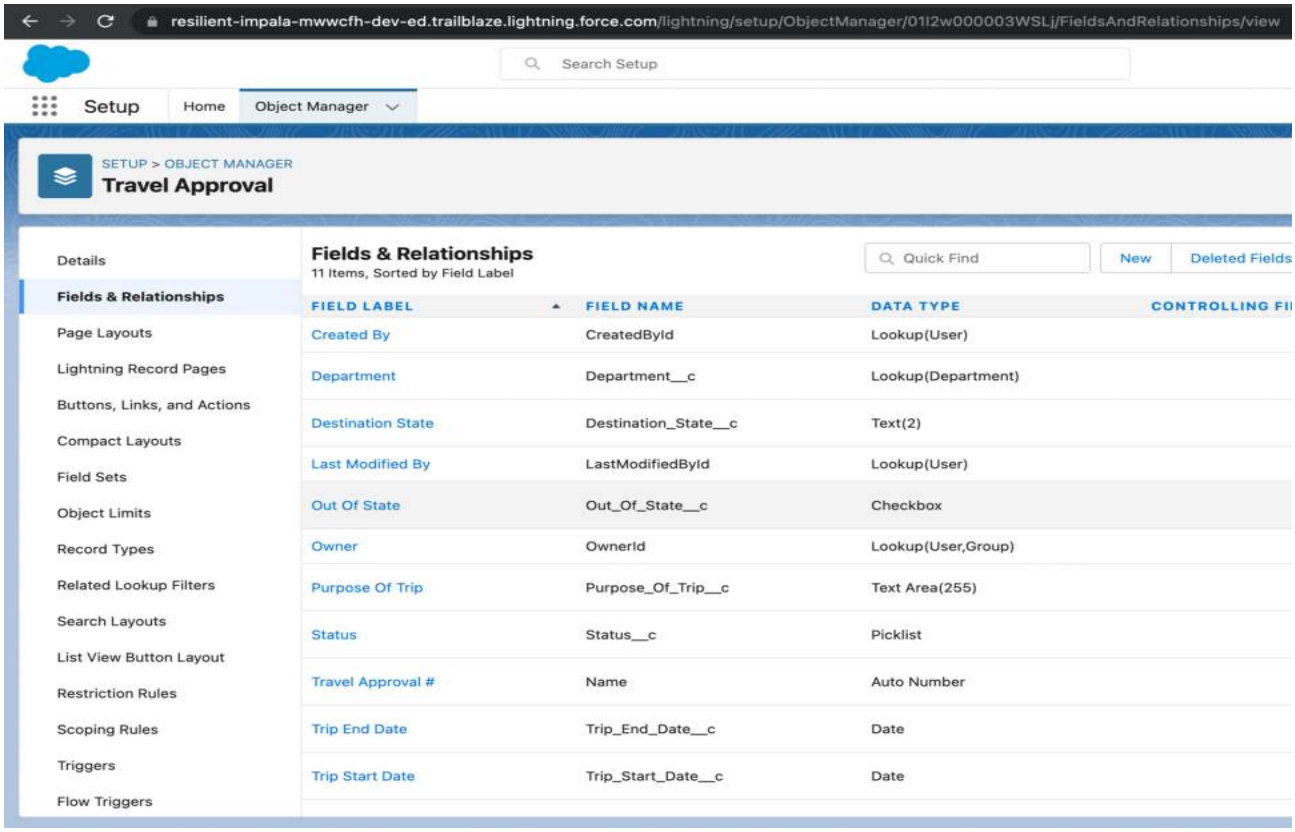

Figure 6

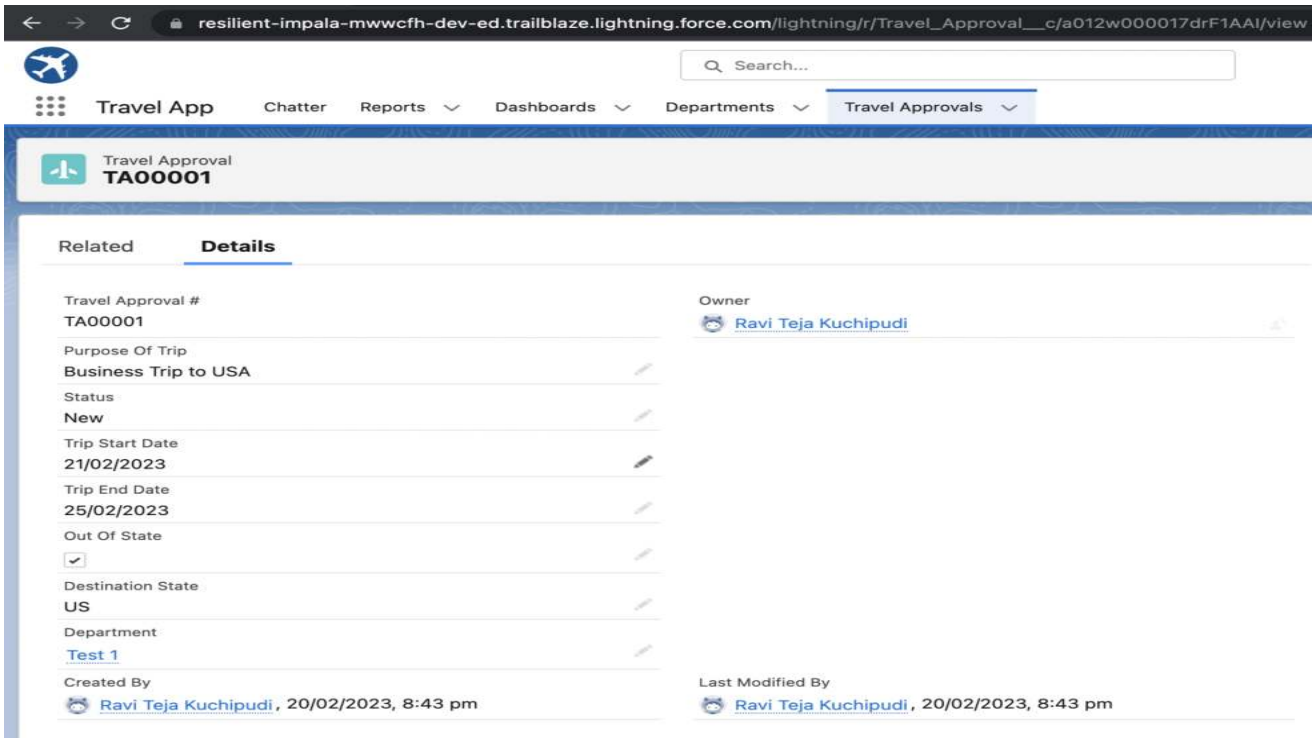

Figure 7

Step 5:

1. Import Departments
2. Download The Department.CSV File
3. Open “Data Import Wizard”, Click “Launch Wizard”
4. Select “Departments” Custom Object
5. Add New Records Upload the file by clicking the CSV icon
6. Map the Fields, click Next
7. Start Start Import
8. Make sure all the records were inserted.

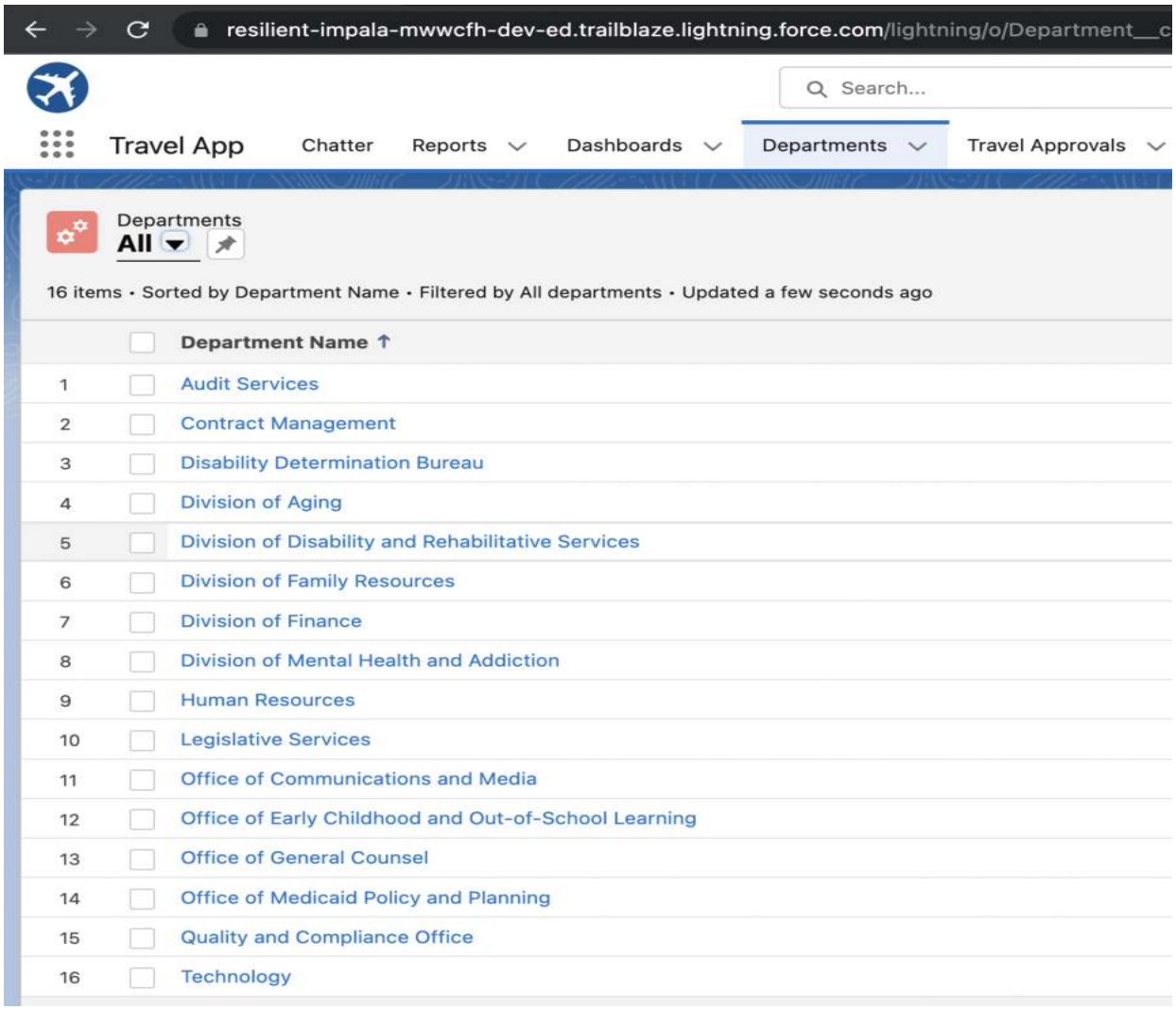

Figure 8

Exercise 2:

Step 1:

1. Create a Travel Approval record.
2. Purpose of Trip = “Attend Dreamforce”, Status = “Draft”, Trip Start Date/End Date = “Pick any Date”, Out of state = True, Destination State = CA, Department = Technology. Save.

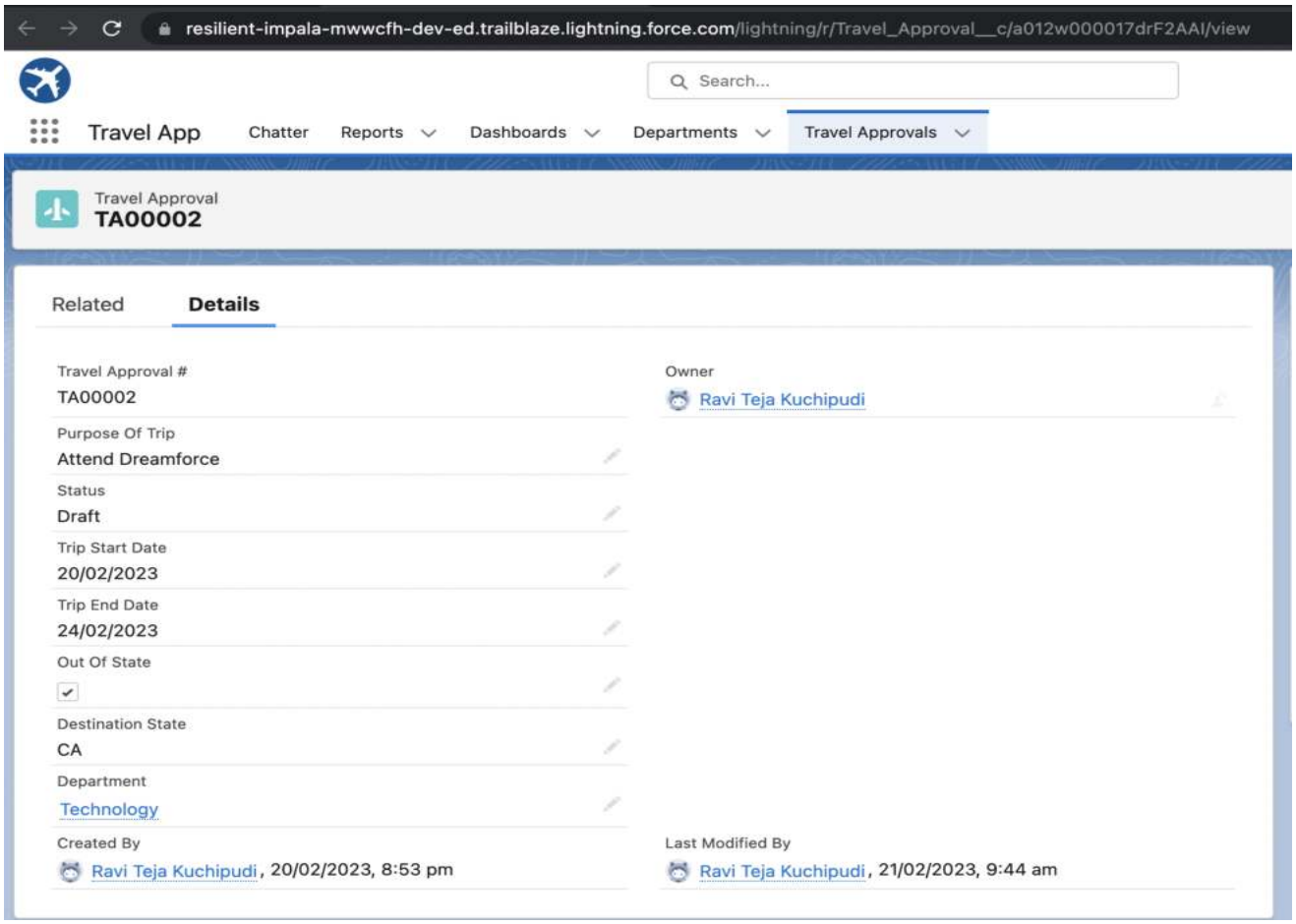

Figure 9

Step 2:

1. Create an Expense Item Object, Label = “Expense Item”, Plural = “Expense Items”, Starts with vowel sound = Checked,
2. Record Name = Expense Item Number, Data Type = Auto Number, Display Format = E – {00000}, Starting Number = 1
3. Allow Reports, Do not create Custom Tab, Save.

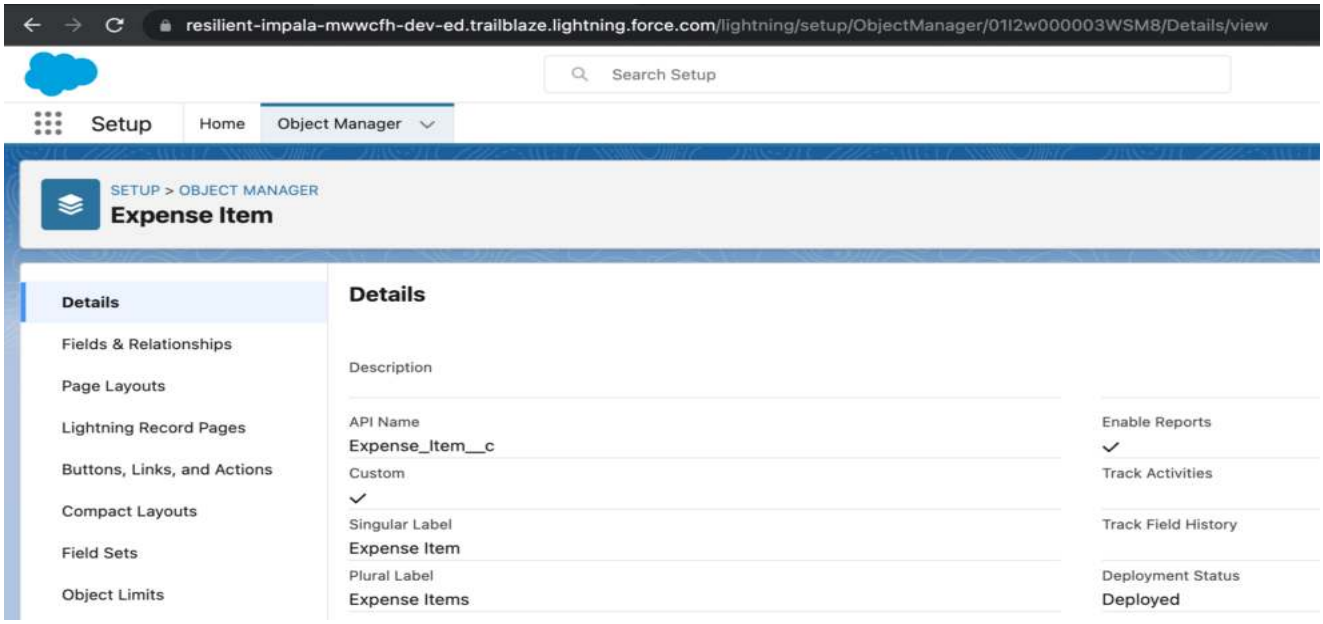

Figure 10

Step 3:

1. Create the following custom fields:
    a. Amount, Length = 16, Type = Currency, Decimal = 2, Required = True.
    b. Expense Type, Type = Picklist, Values = Airfare, Hotel, Rental Car, Meals, Other, Required = True.
    c. Travel Approval, Type = Master-Detail,Related To – Travel Approval.

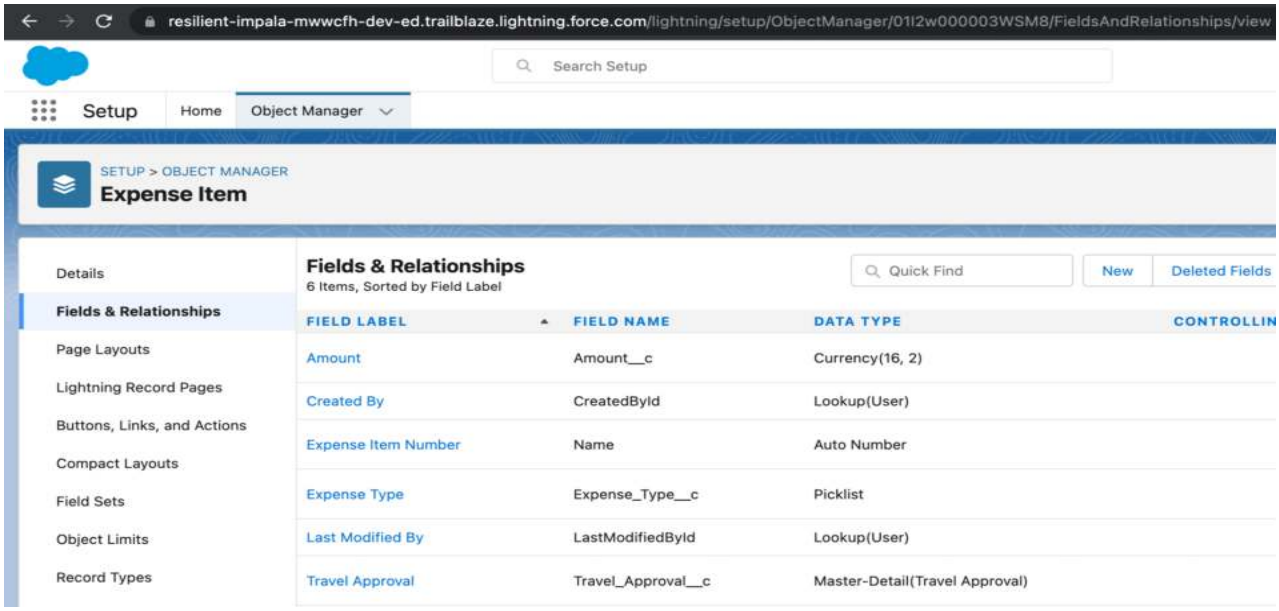

Figure 11

Step 4:

1. Create Expense Items.
    a. Amount = 870, Expense Type = “Hotel”, Save.
    b. Amount = “450”, Expense Type = “Airfare”, Save.

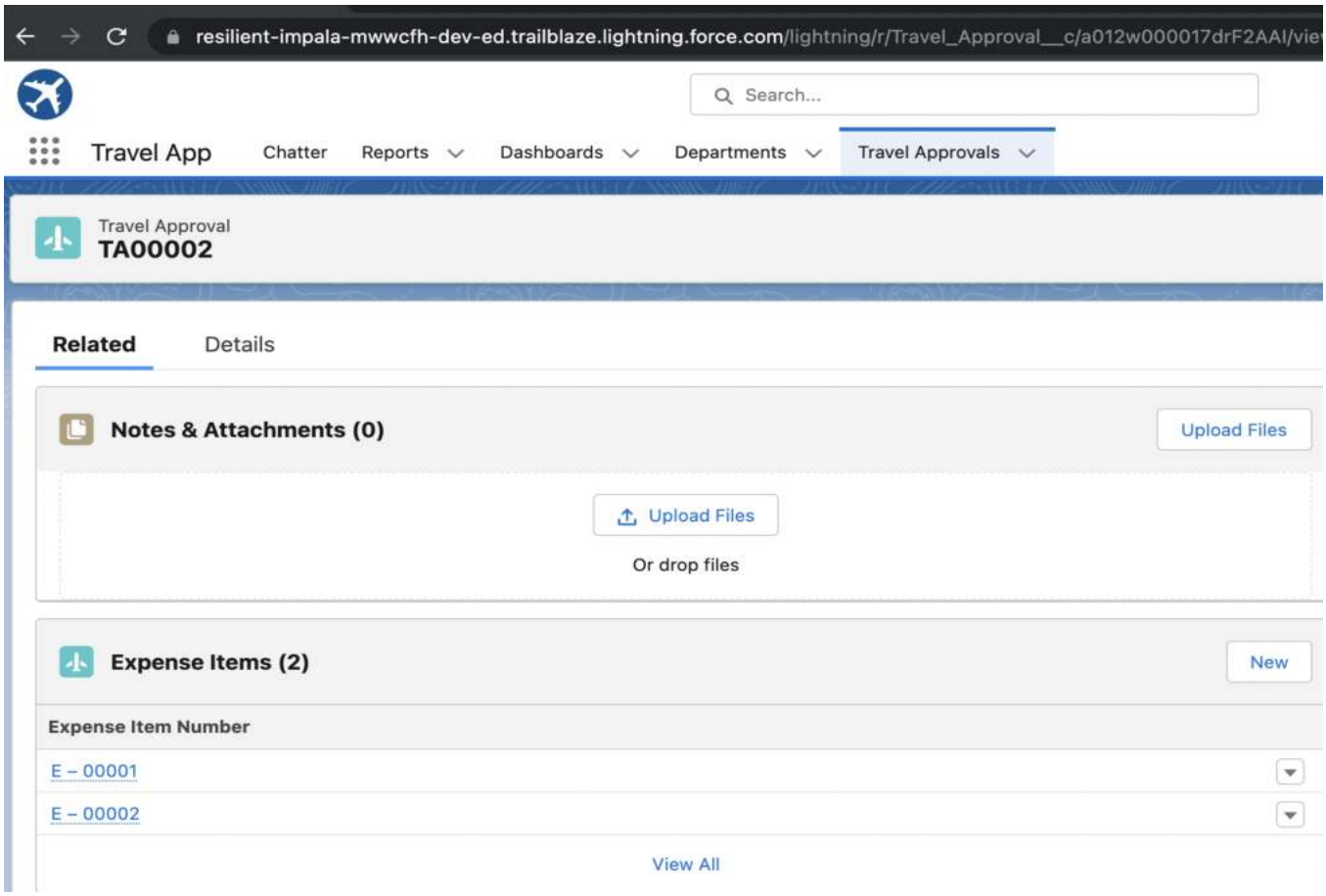

Figure 12

Step 5:

1. Create a User.
2. First Name = “Eric”, Last Name = “Executive”, Email = “Use your own email”, Username Name = “Choose a Unique username”
3. Role = “CEO”
4. License = Salesforce.
5. Profile = System Administrator.
6. Save.

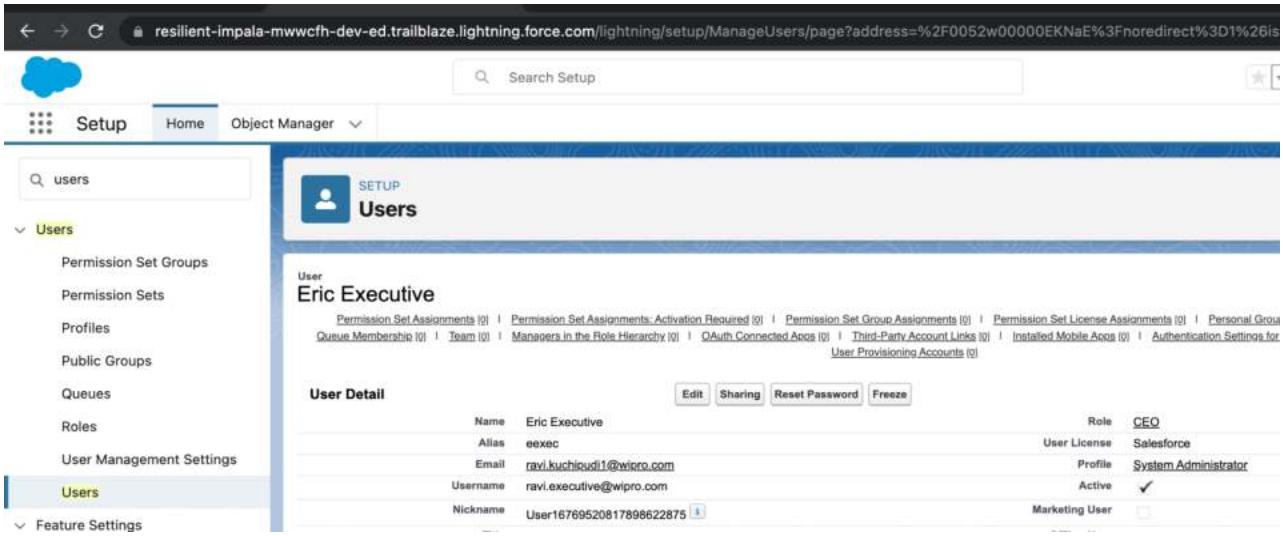

Figure 13

Step 6: Add user Eric Executive as your manager

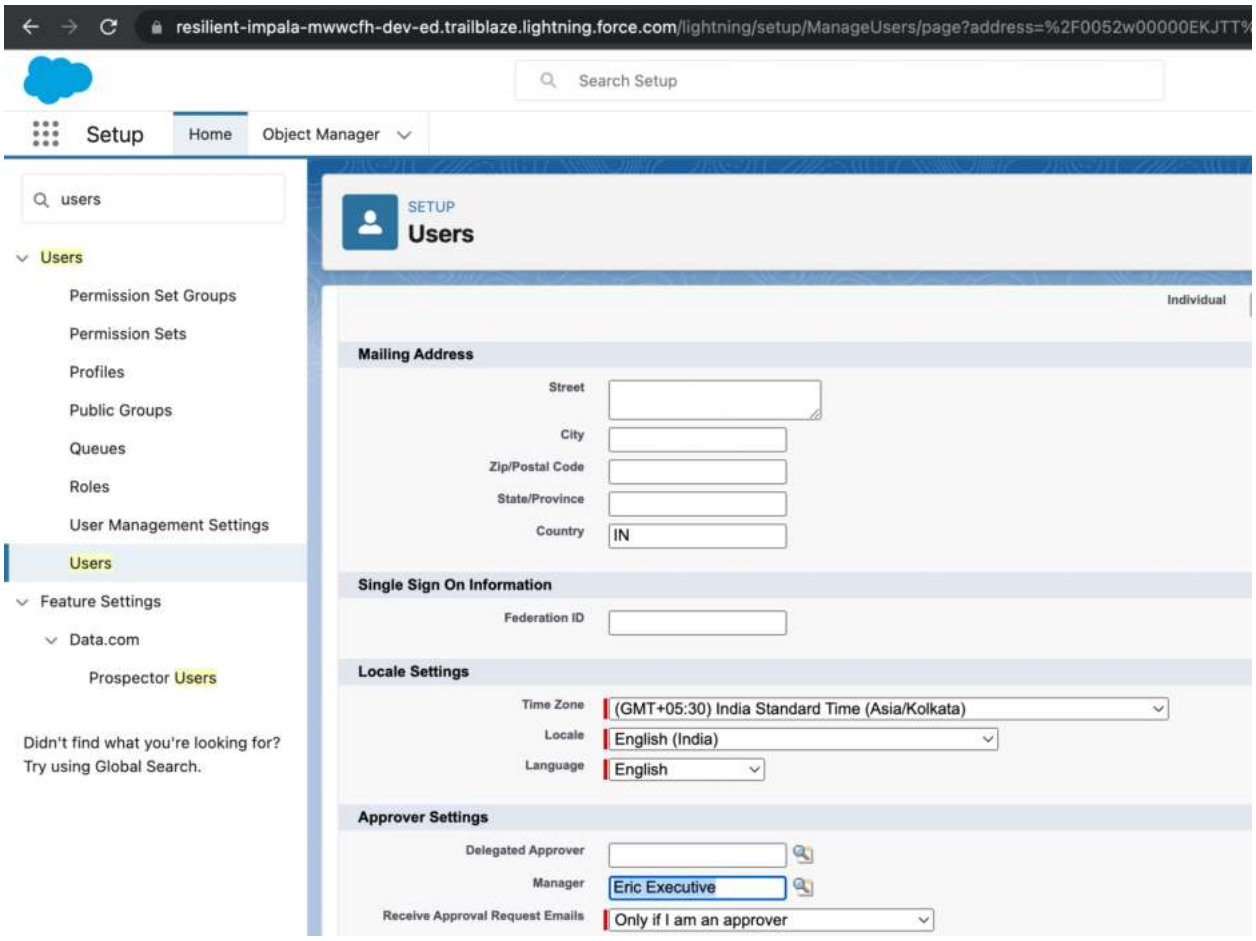

Figure 14

Step 7: Customize the Travel Approval Default search layout.

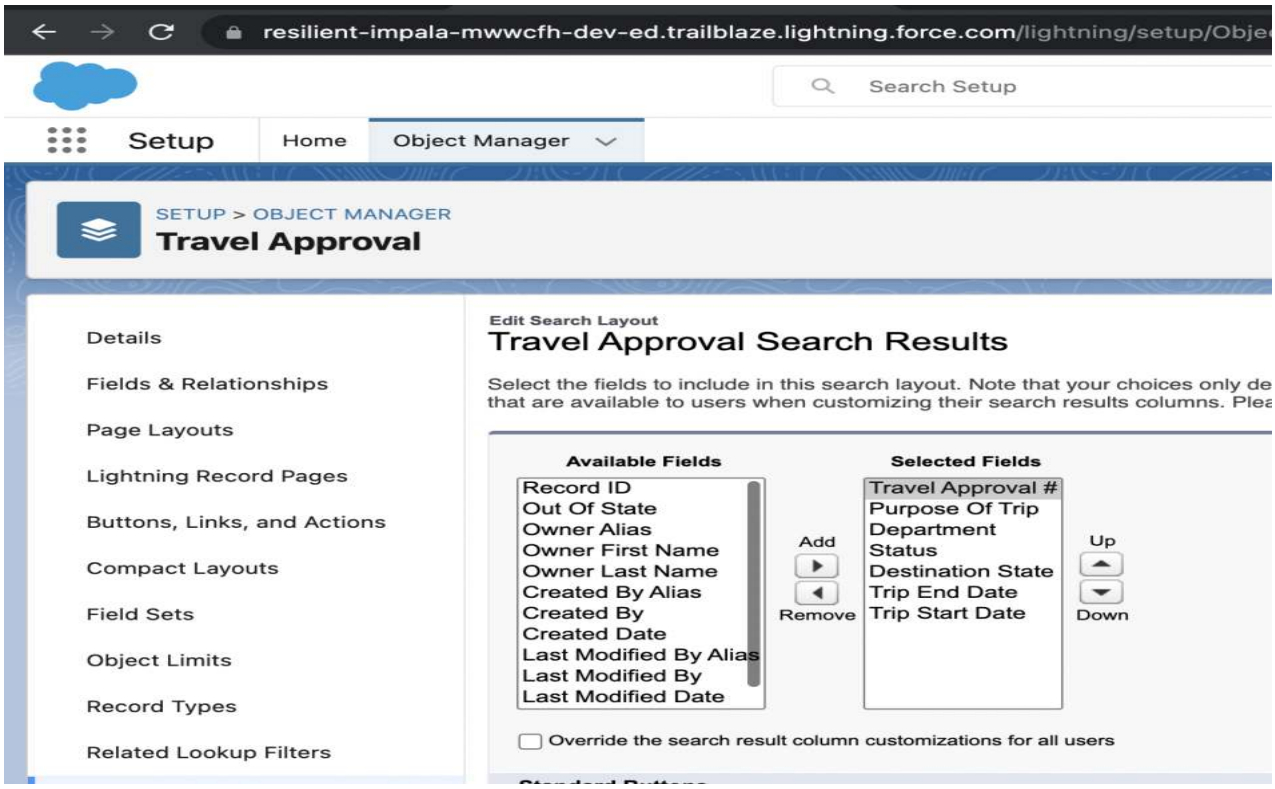

Figure 15

Step 8: Select fields to display in the Travel Approval "All" List view.

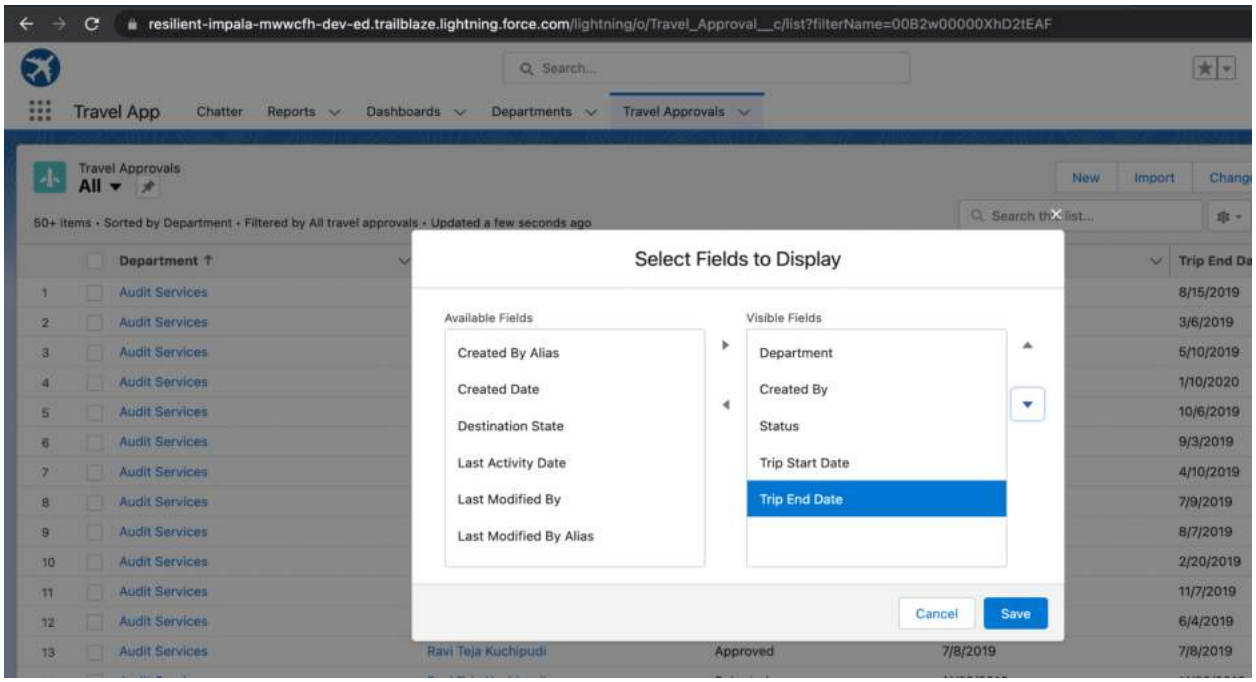

Figure 16

Step 9:

1. Create Travel approval custom List View “Open Out of State Travel Requests” .
2. All users should be able to see this list view.

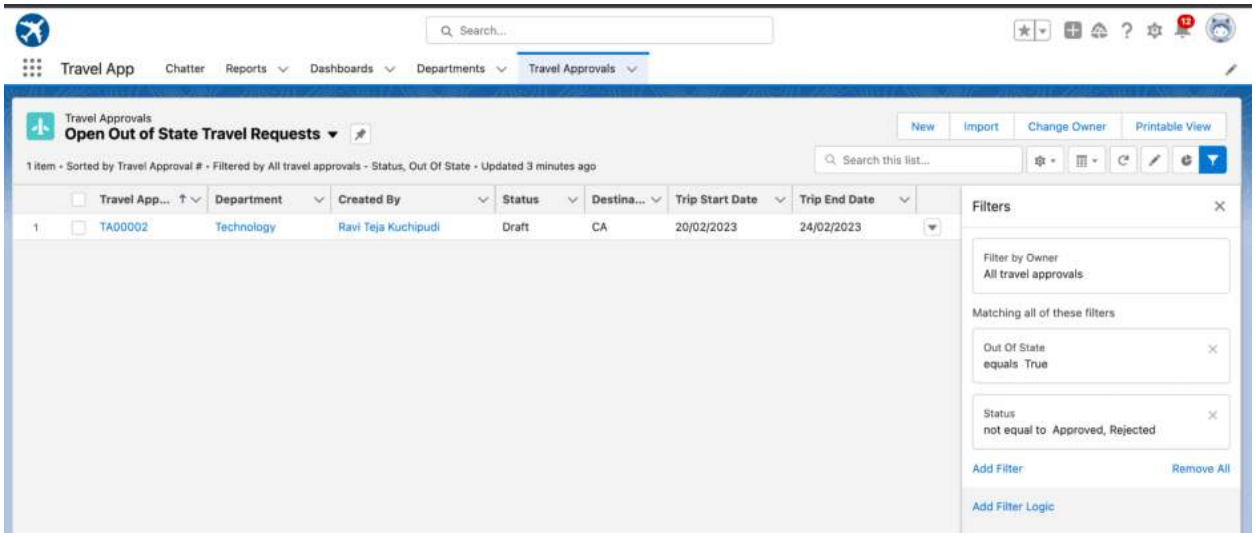

Figure 17

Step 10: Select fields to display in the Travel Approval “Open Out of State Travel Requests” List view.

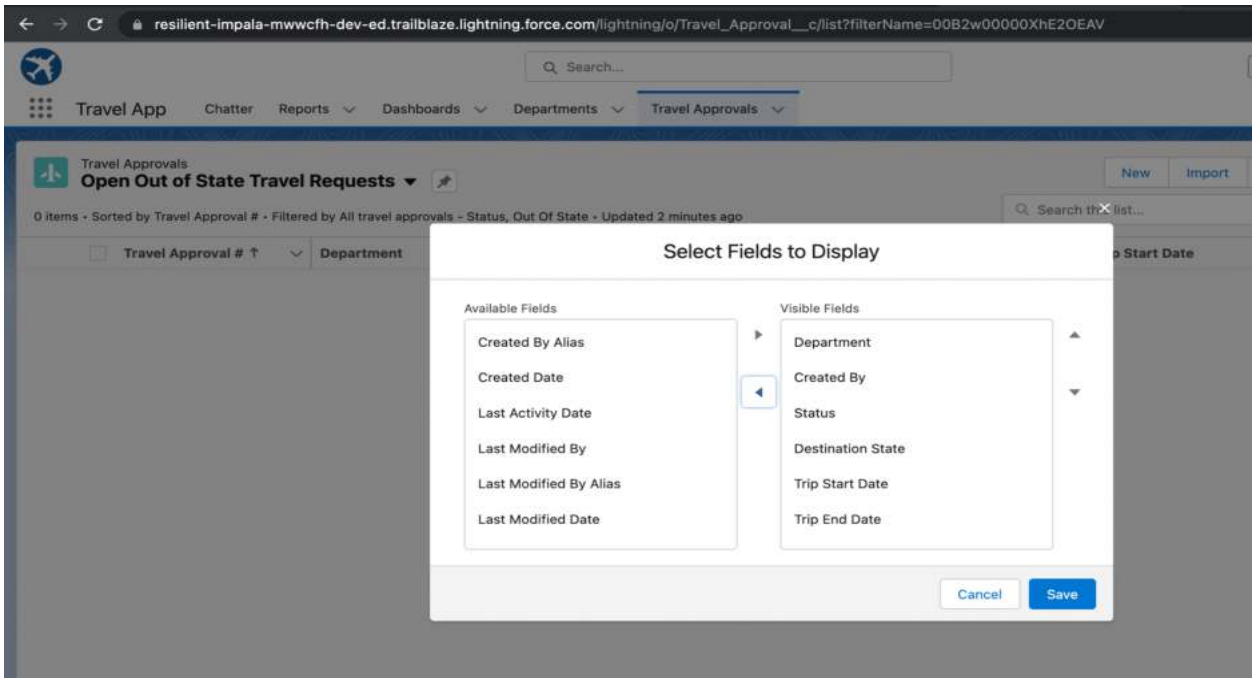

Figure 18

Step 11:

1. Add a section to the page layout called “Trip Info” and add the fields.

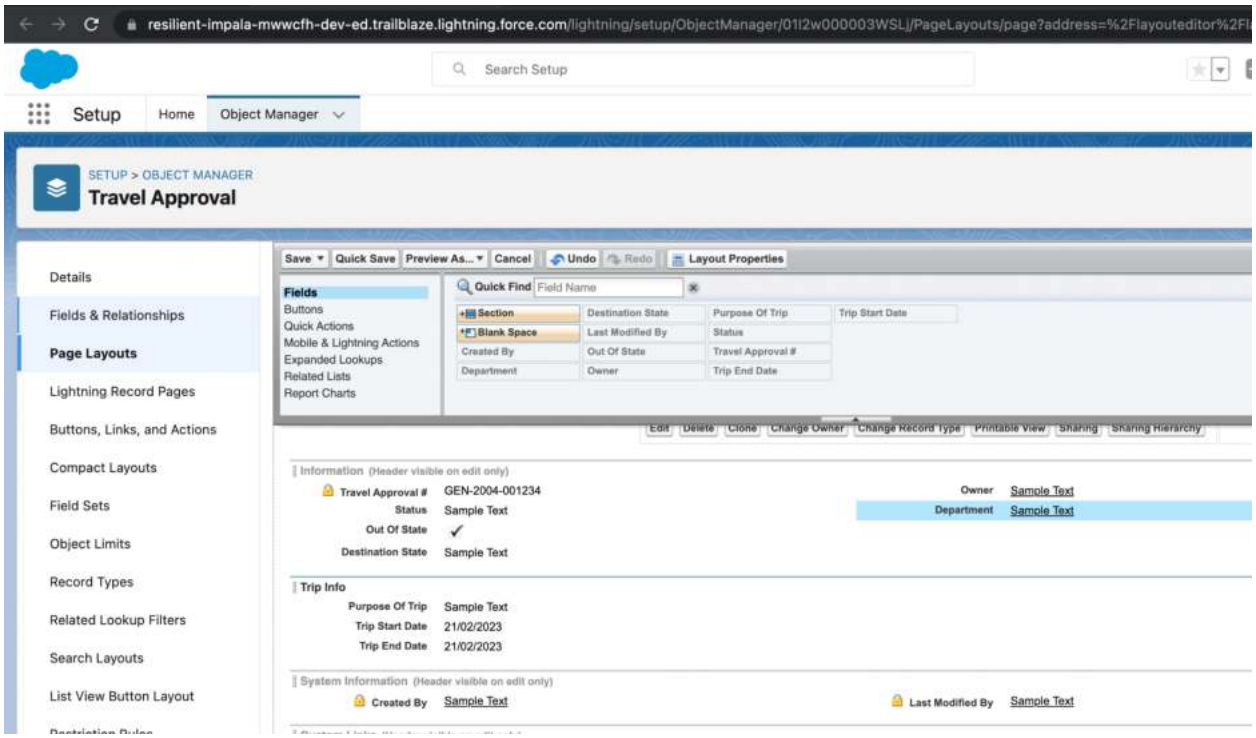

Figure 19

Step 12: Customize the Expense Item Related List under the Travel Approval page

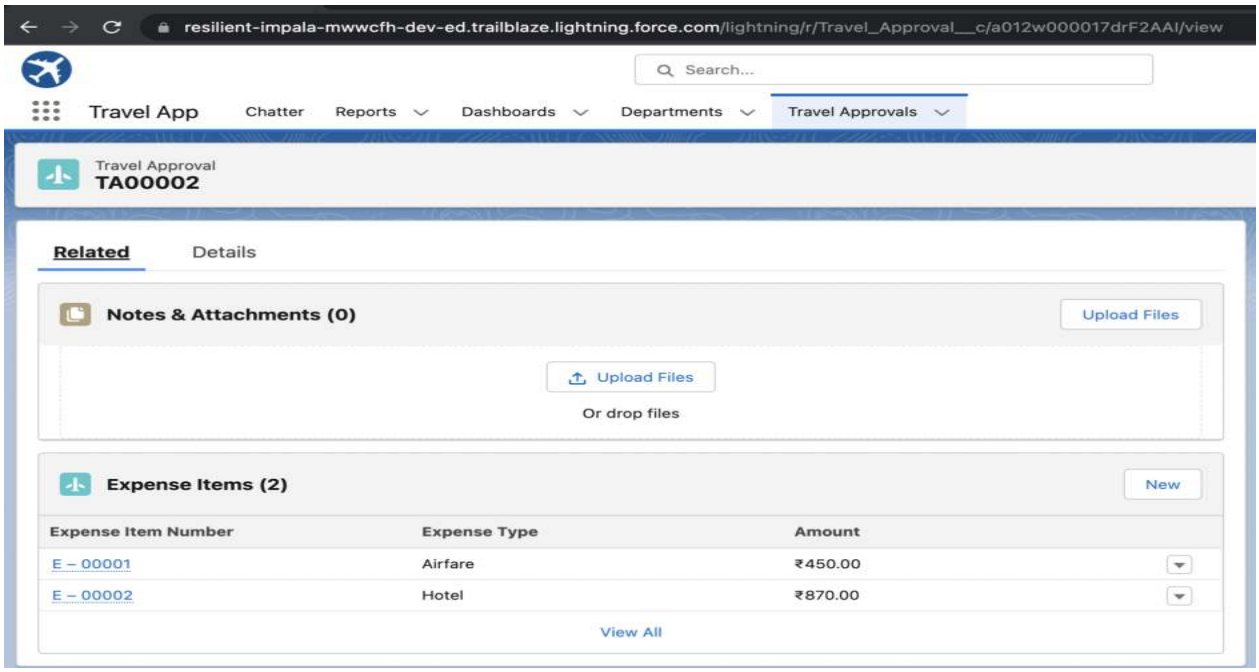

Figure 20

Step 13:

1. Enable Chatter and "Feed Tracking" for Travel Approval Object.
2. Select these 2 fields: Destination State Status Save.
3. Open a Travel Approval record. Click on Chatter Tab. Share a post
4. Login in as Eric and reply to the email.

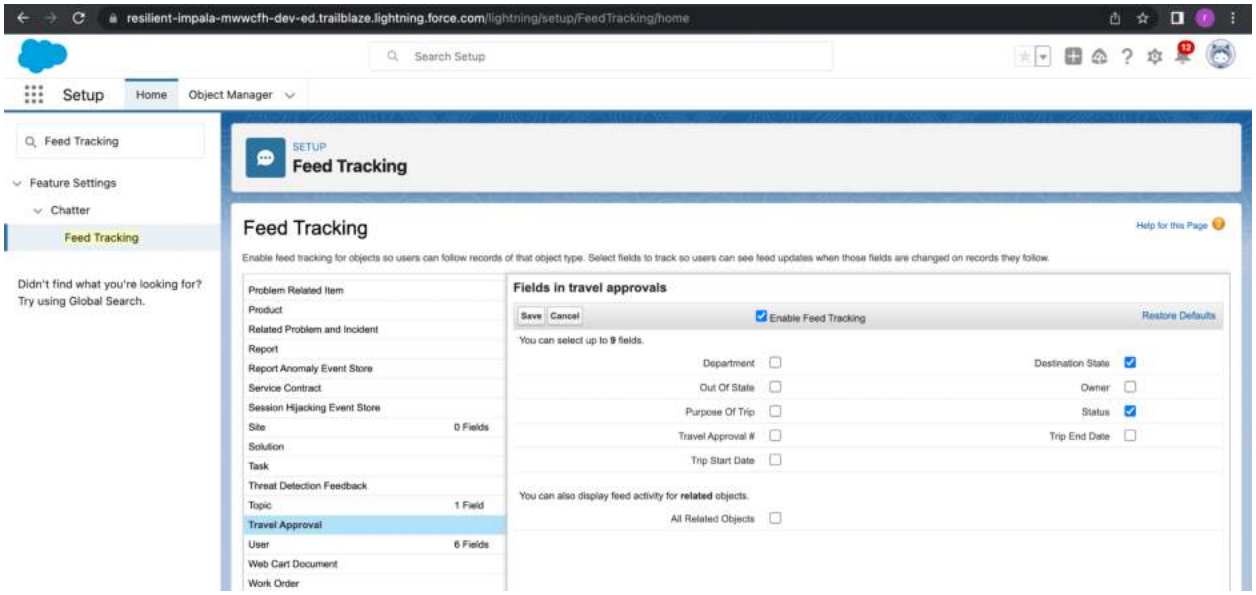

Figure 21

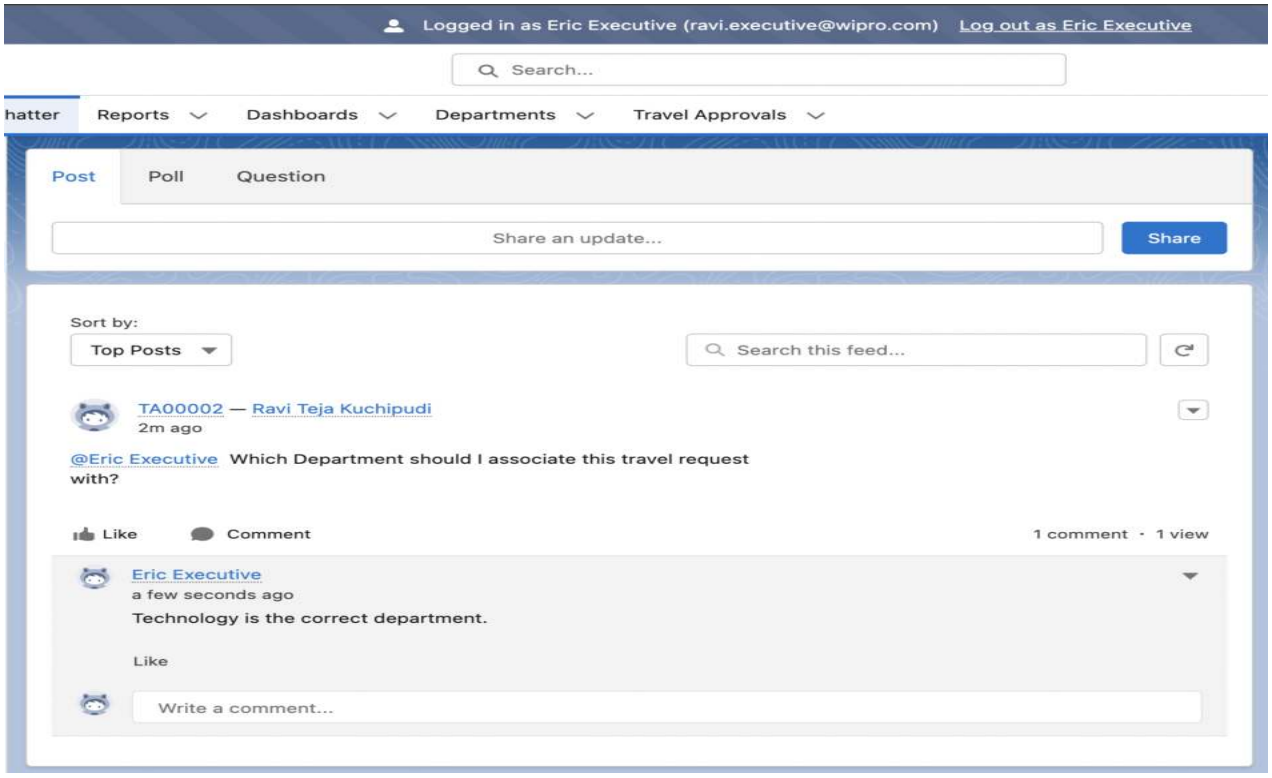

Figure 22

Test The App:

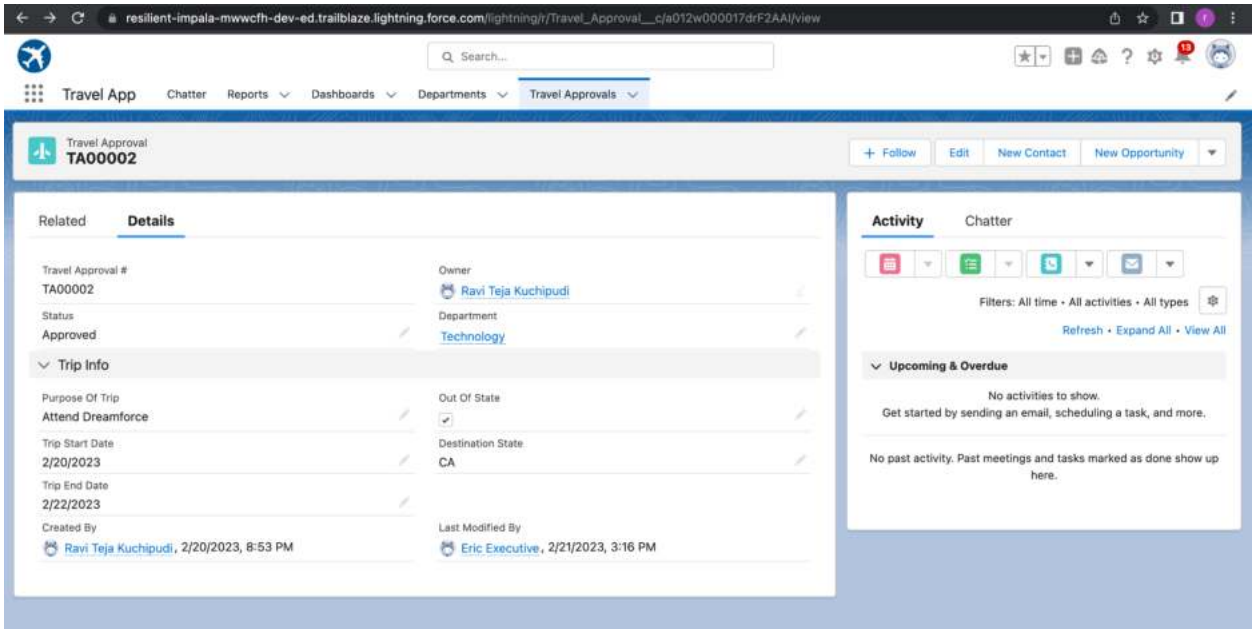

Figure 23

Module 2

Business Requirements:

- Add Business logic & Analytics to the Travel Approval App.
- You will be adding Business logics, Reports and Dashboards to:
- Enforce data integrity with validation rules.
- Define business logic using roll-up summary and formula fields
- Automate business processes with Flows.
- Define Approval Process for Travel Approval App.
- Create Reports to analyze Travel Approvals.
- Build Dashboards for Travel Approval Management.

Exercise 1:

Step 1:

1. Create Validation Rule.
2. Trip end date must always be greater than (>=) the trip start date.
3. Name: Trip end date after start date.
4. Make sure to keep “Active” selected/checked.
5. Error Message: Trip end date must be greater than or equal to start date.
6. Error Location: Select Field and pick Trip End Date as the location.

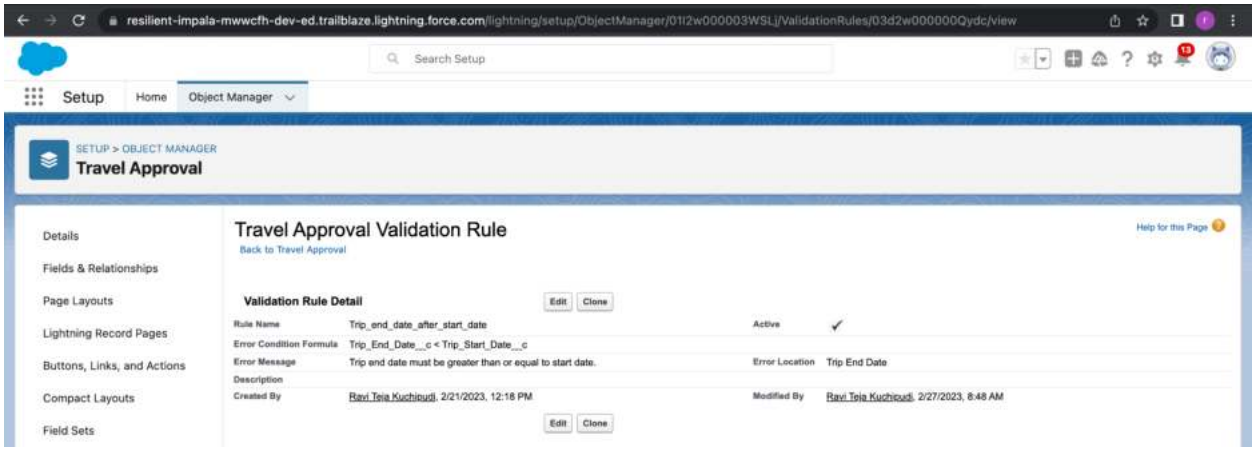

Figure 24

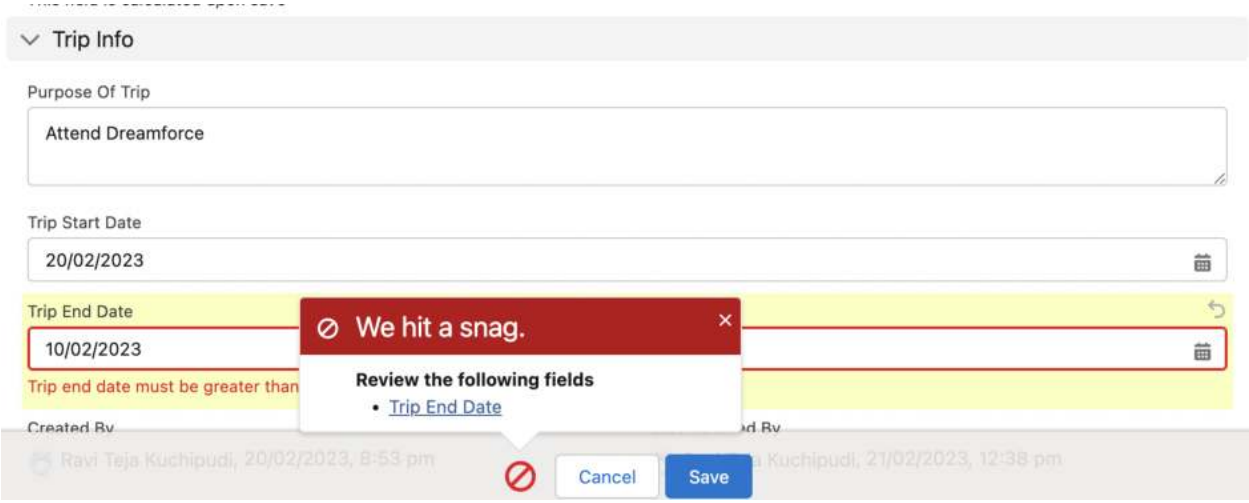

Figure 25

Step 2:

1. Create a Roll-Up Summary Field on Travel Approval object.
2. Business Logic: Automatically sum up the total amount of expenses from the related Expense Item.
3. Field Label = Total Expenses
4. Field Name = Total_Expenses
5. Roll-Up Type: SUM
6. Field to Aggregate: Amount
7. Filter Criteria: All records should be included in the calculation and Save.

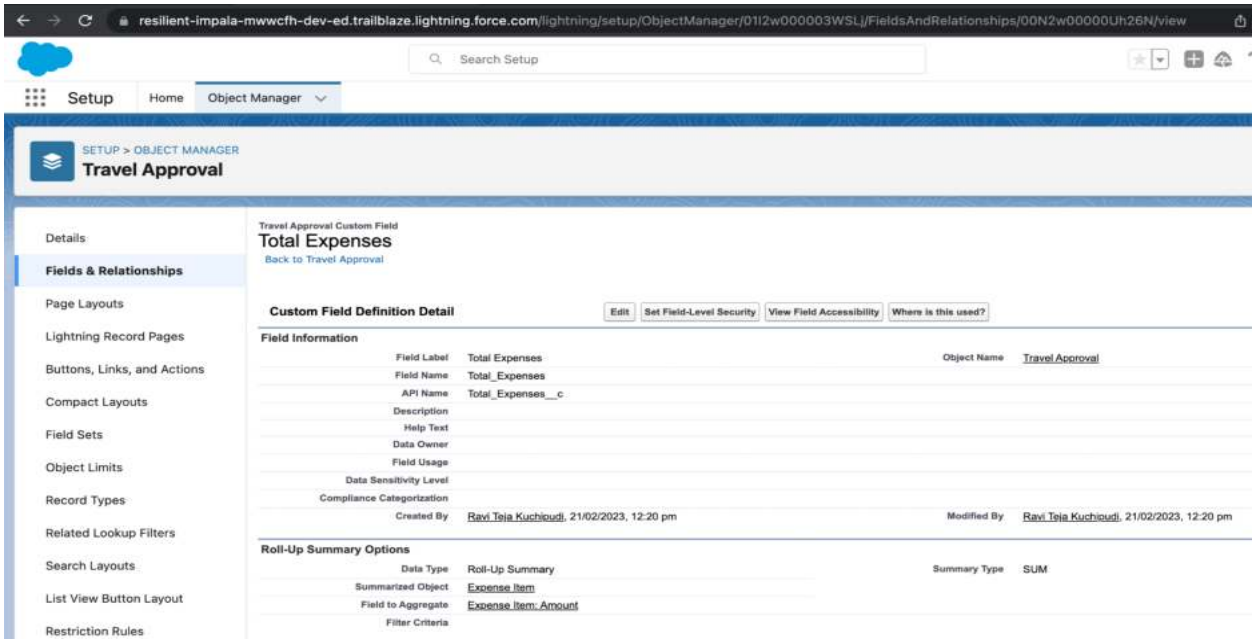

Figure 26

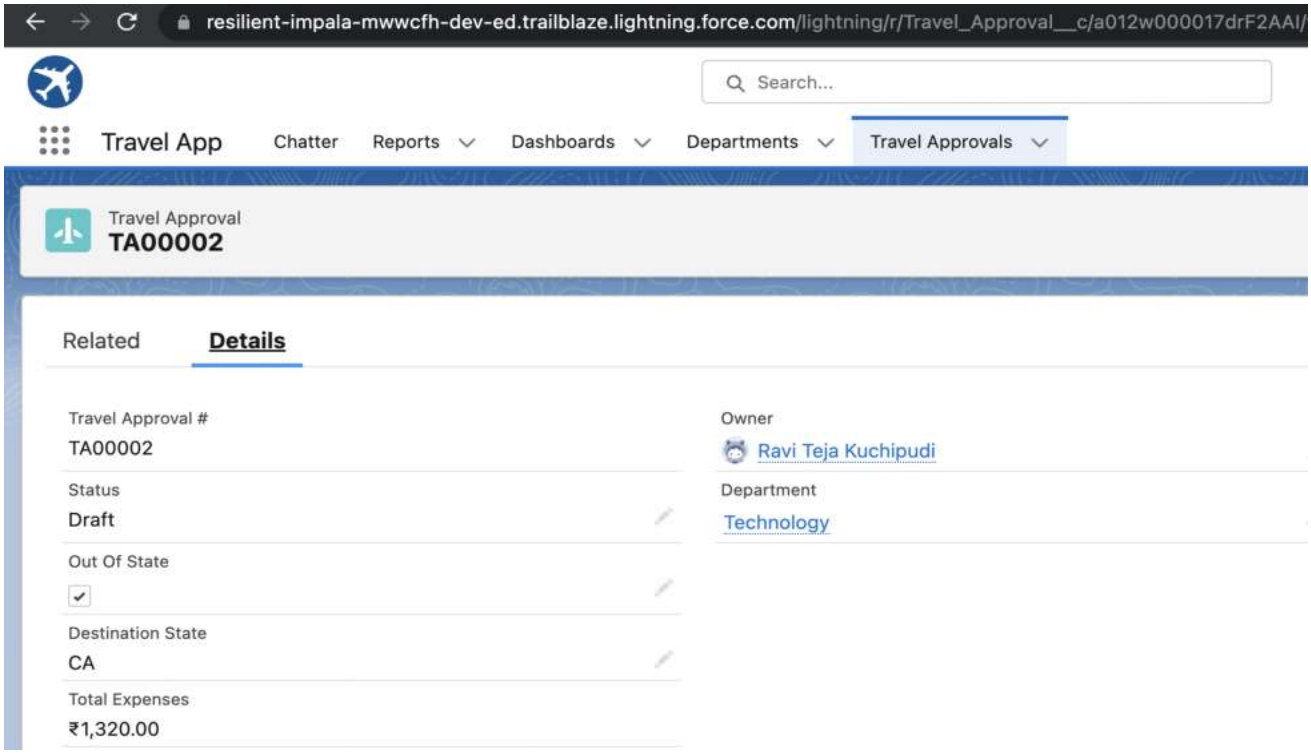

Figure 27

Step 3:

1. Create Formula Fields.
2. Business Logic: Create a field that shows a visual indicator based on the value of the Status field.
3. Cache-Control = Public.
4. File = StatusImages.zip [upload it from the Project Folder]
5. Name = StatusImages
6. Setup | Custom Code | Static Resource | New
7. Save

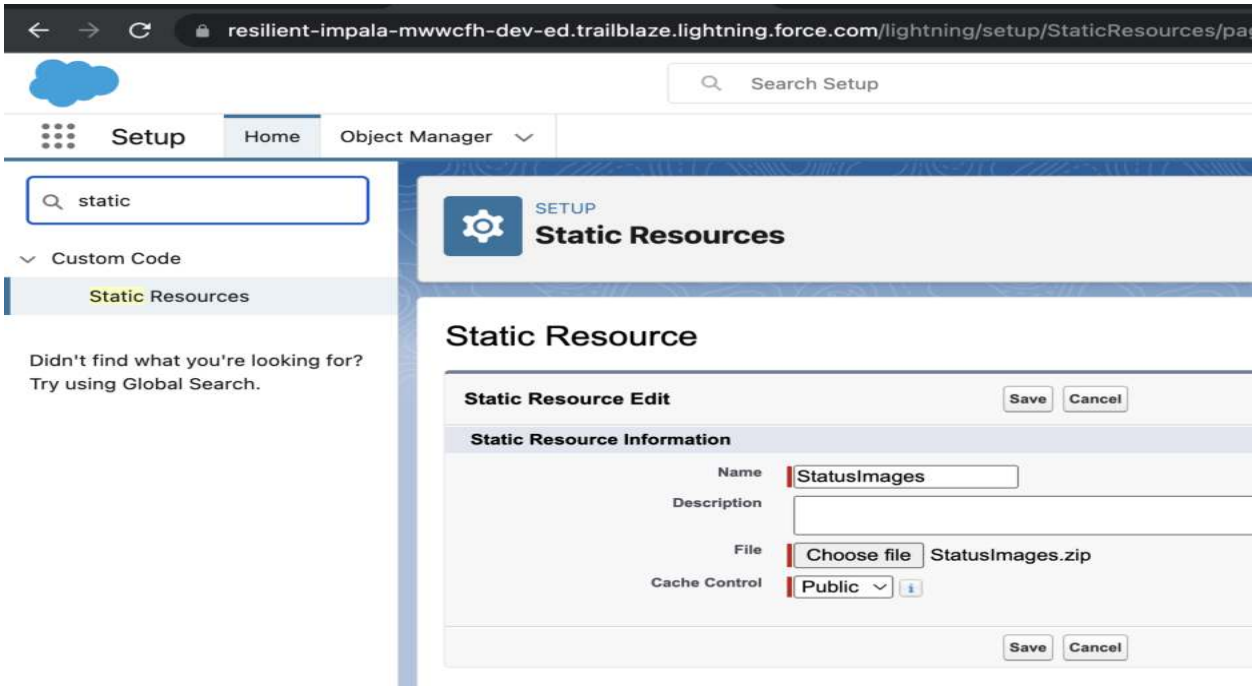

Figure 28

Step 4:

1. Create a Formula field on the Travel Approval object to show an image based on the Status field.
2. Field Label: Status Indicator, Formula Return Type = Text

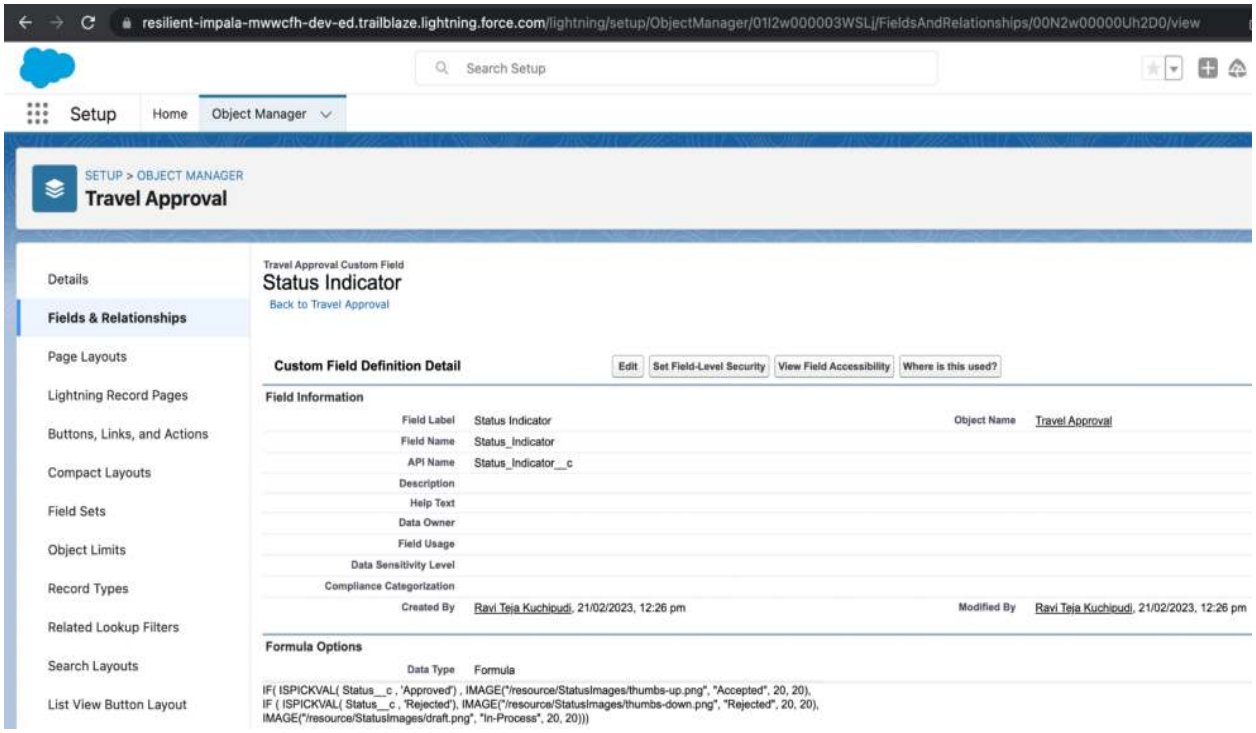

Figure 29

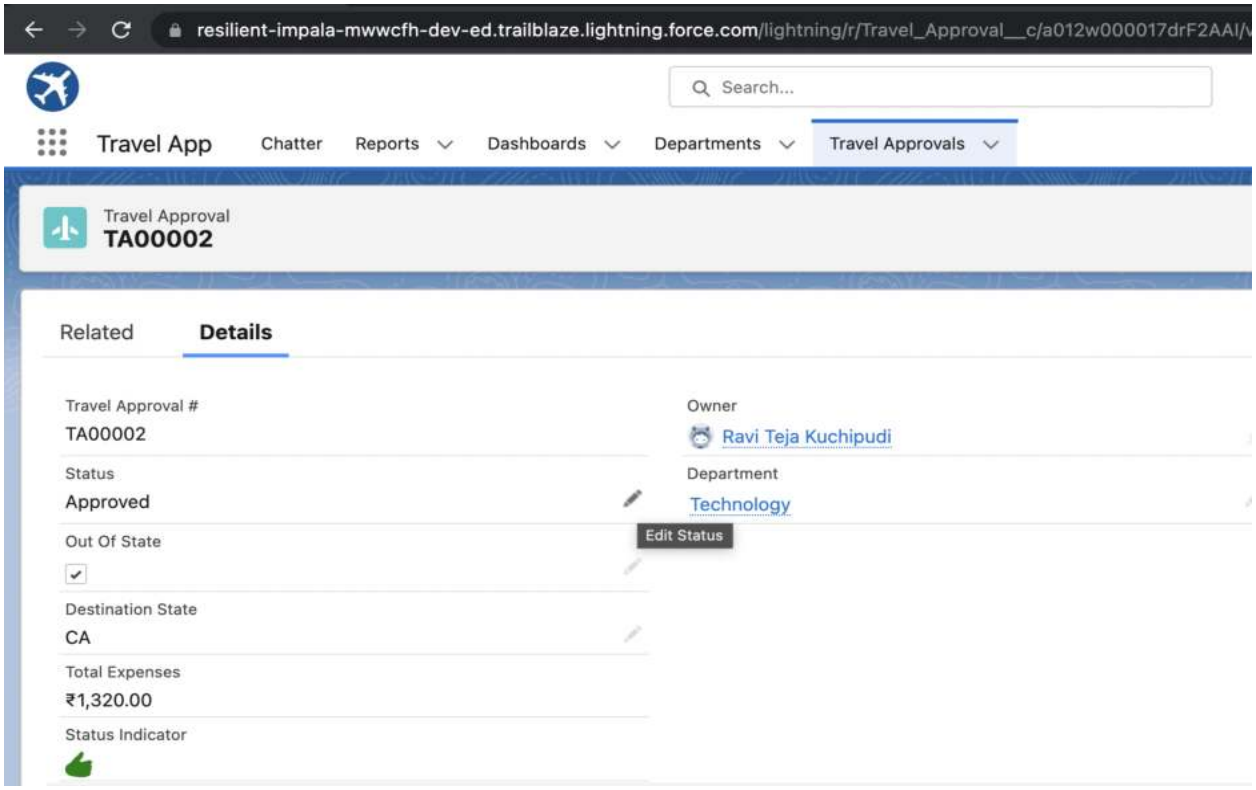

Figure 30

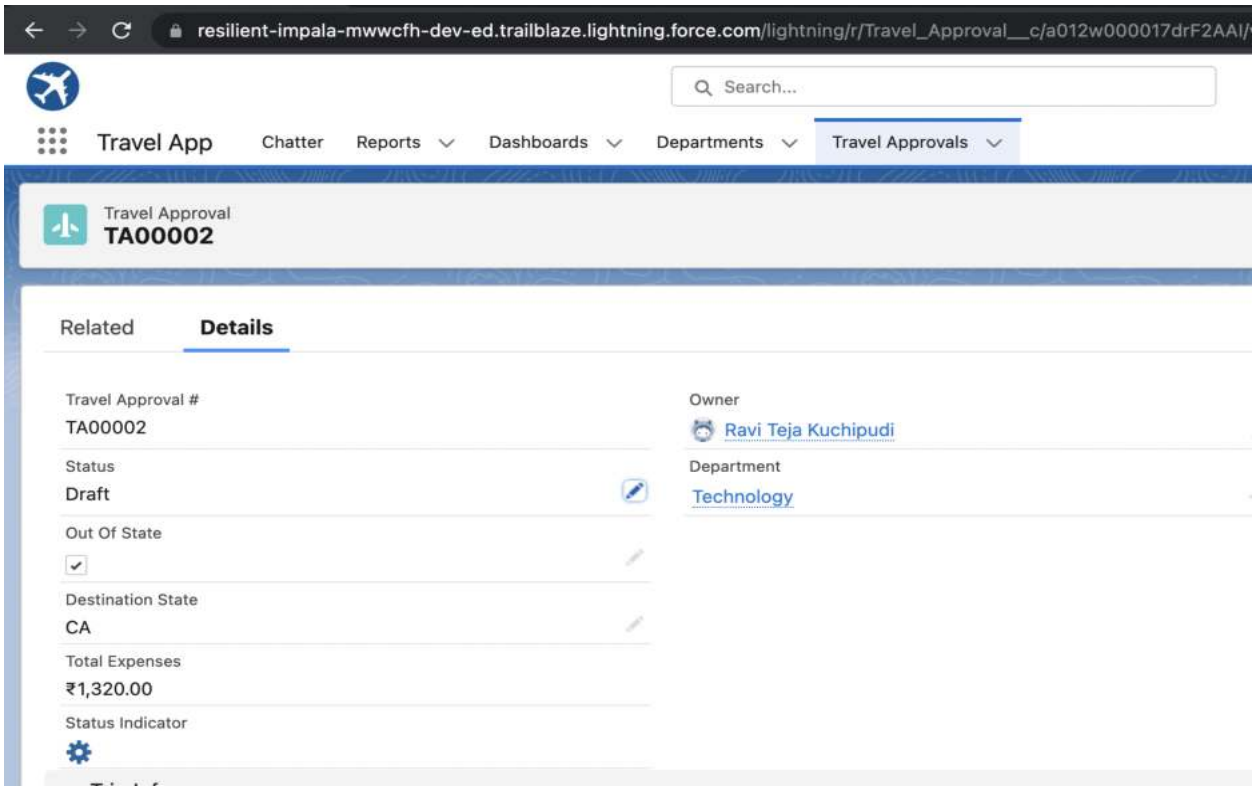

Figure 31

Step 5:

1. Create a Record – Triggered Flow.

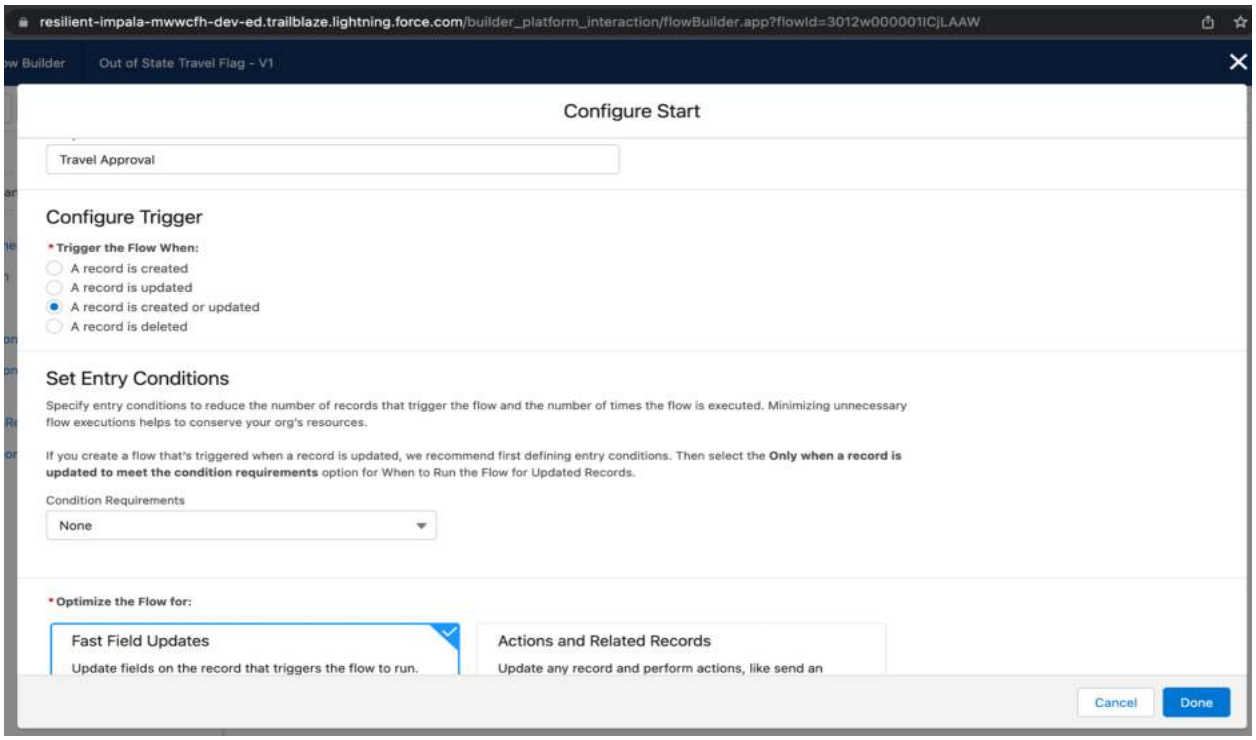

Figure 32

2. Add a Decision Element to the Flow.

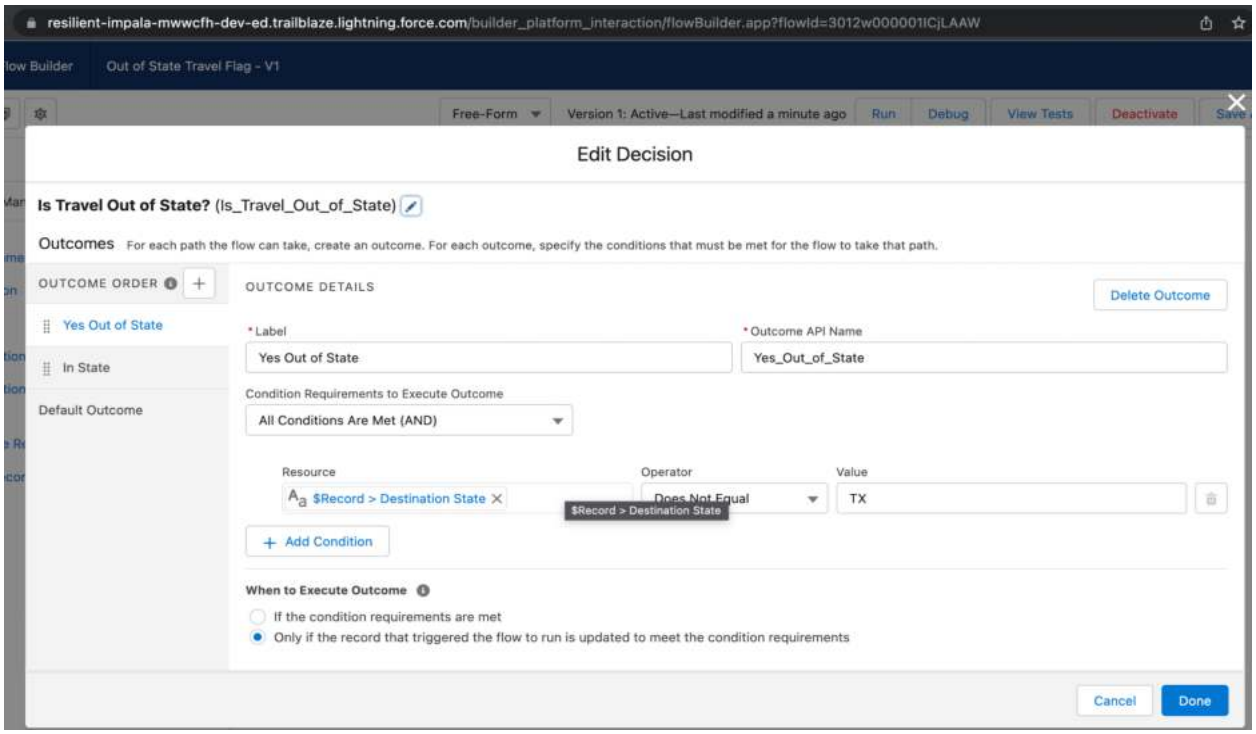

Figure 33

3. Next to Outcome Order click the + button to add another outcome.

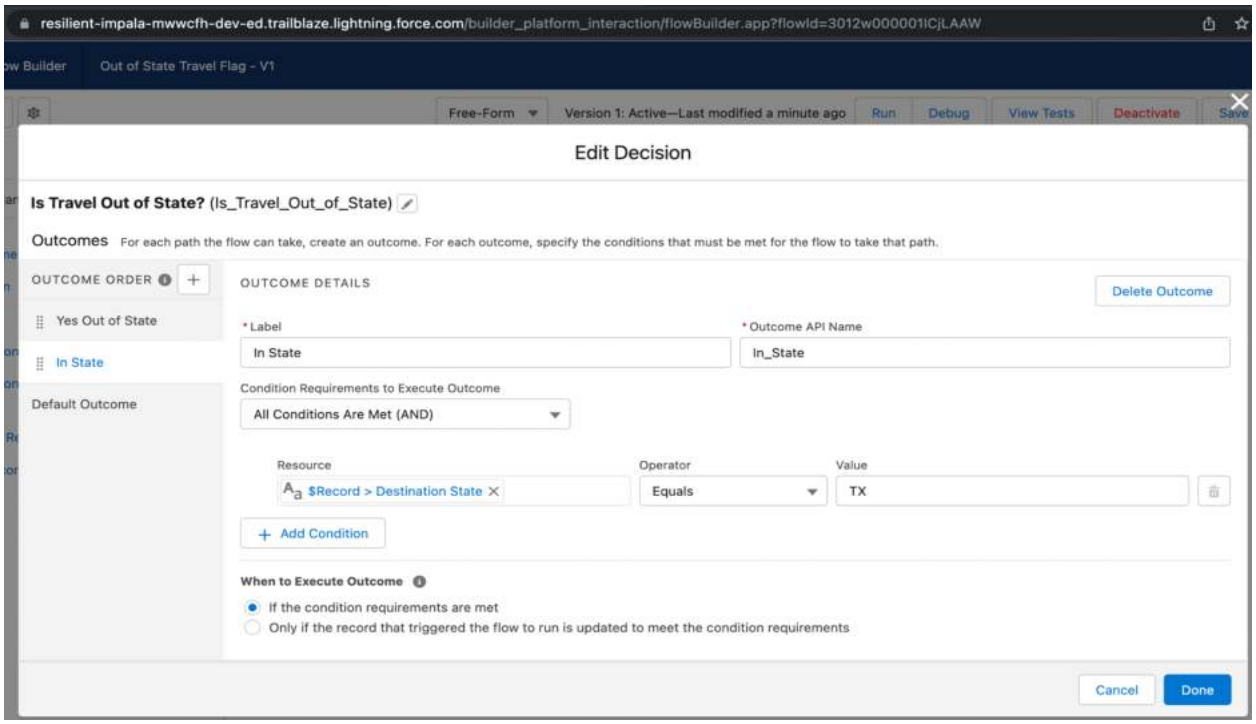

Figure 34

4. Create an Action for the Flow Using Update Records Elements.
5. Value = $GlobalConstant.True

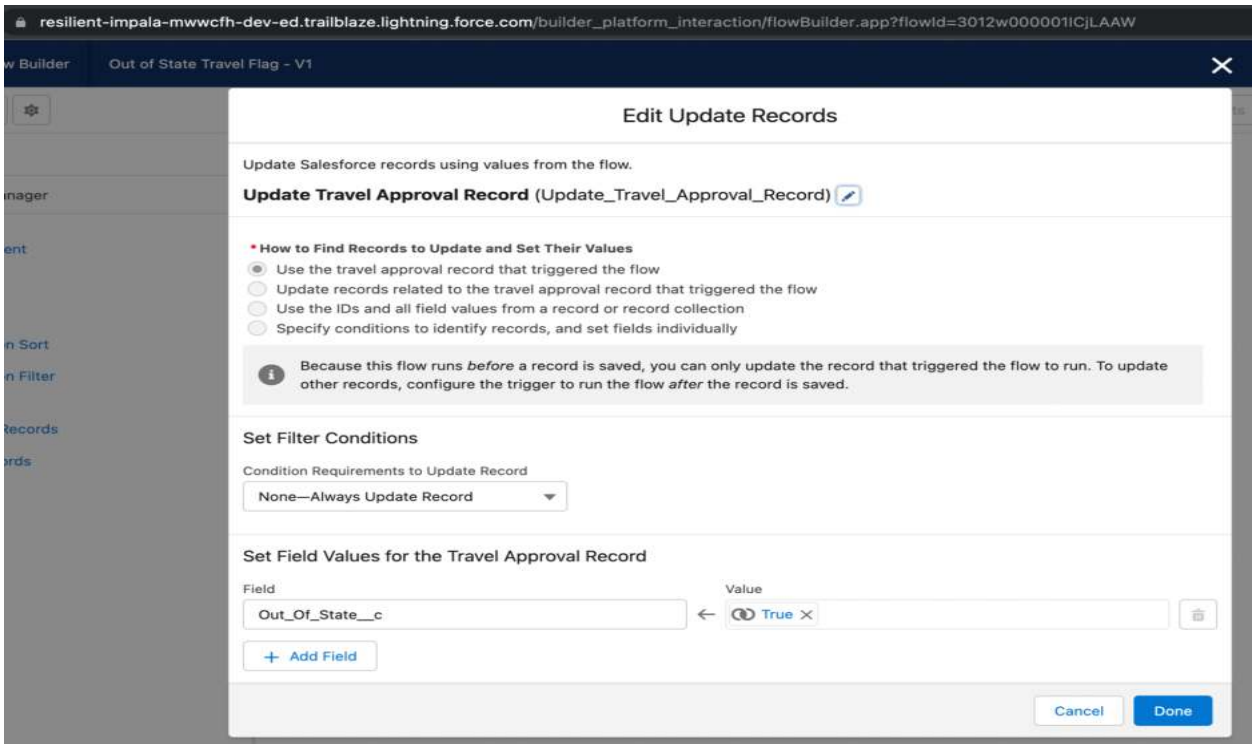

Figure 35

6. Create an Action for the Flow Using Update Records Elements.
7. Value = $GlobalConstant.False

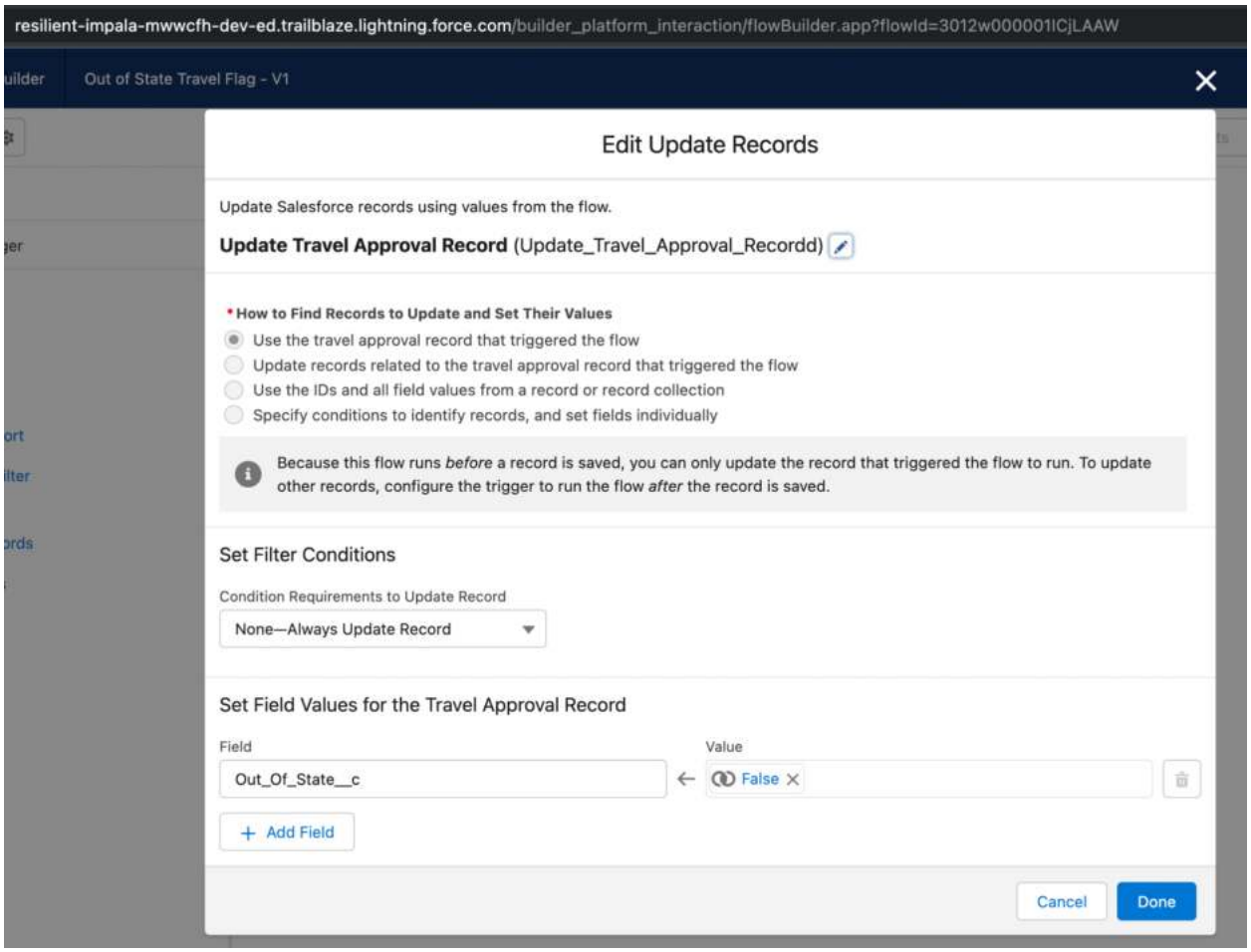

Figure 36

8. Make sure to save and activate the flow.
9. Flow Label = Out of State Travel Flag
10. How to find Records = Use the travel approval record that triggered the flow
11. Click Save.
12. Click Activate.

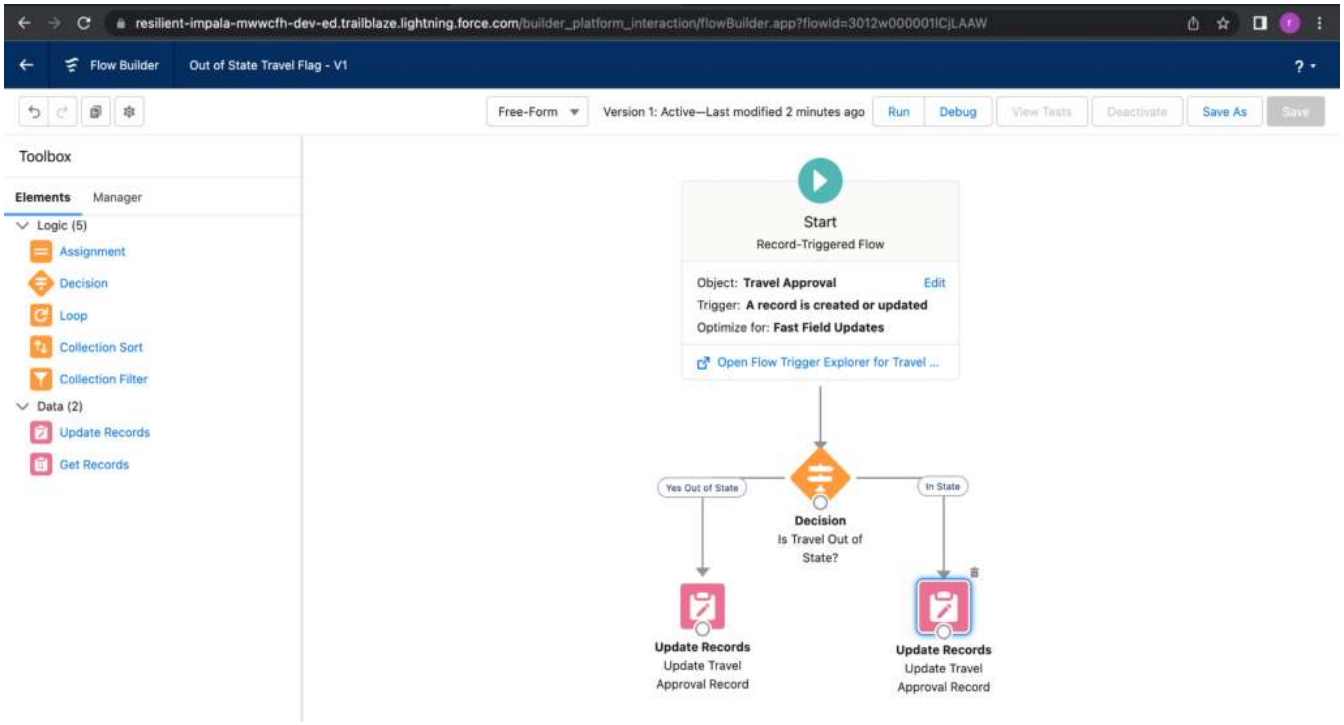

Figure 37

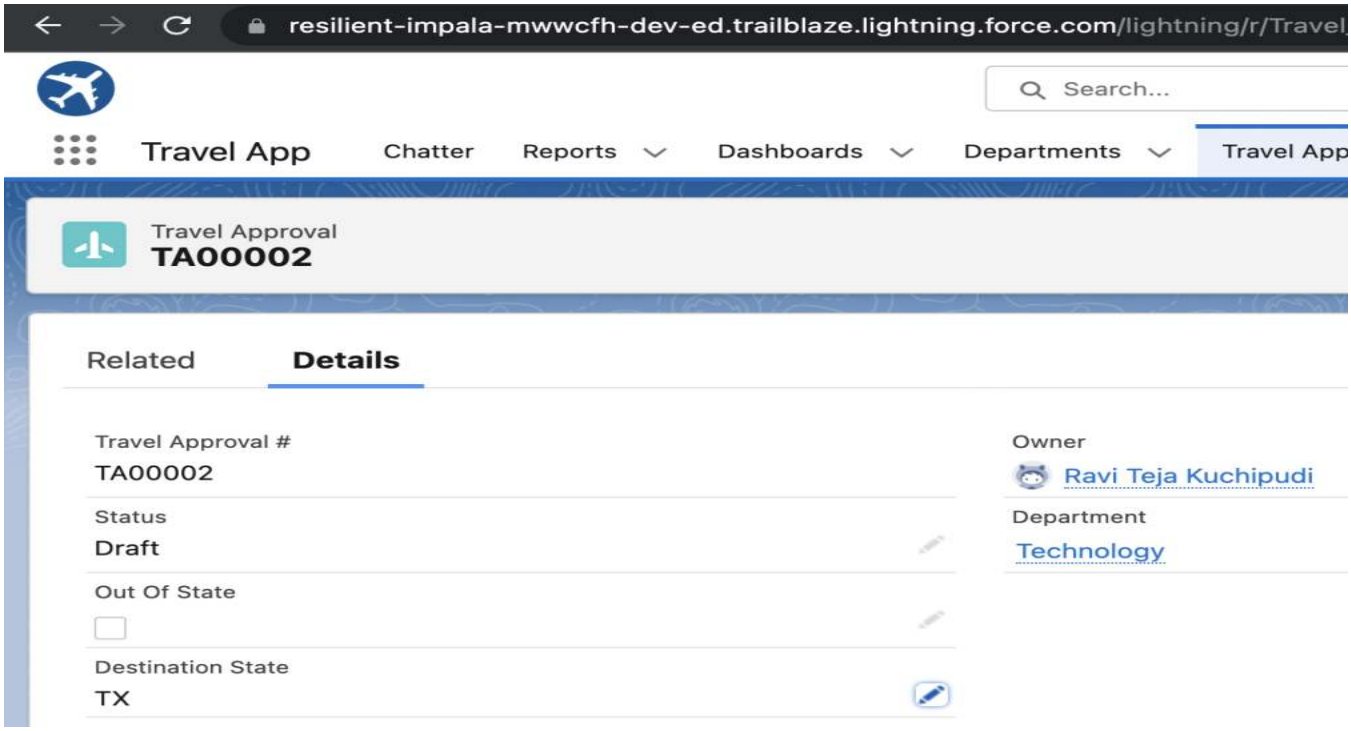

Figure 38

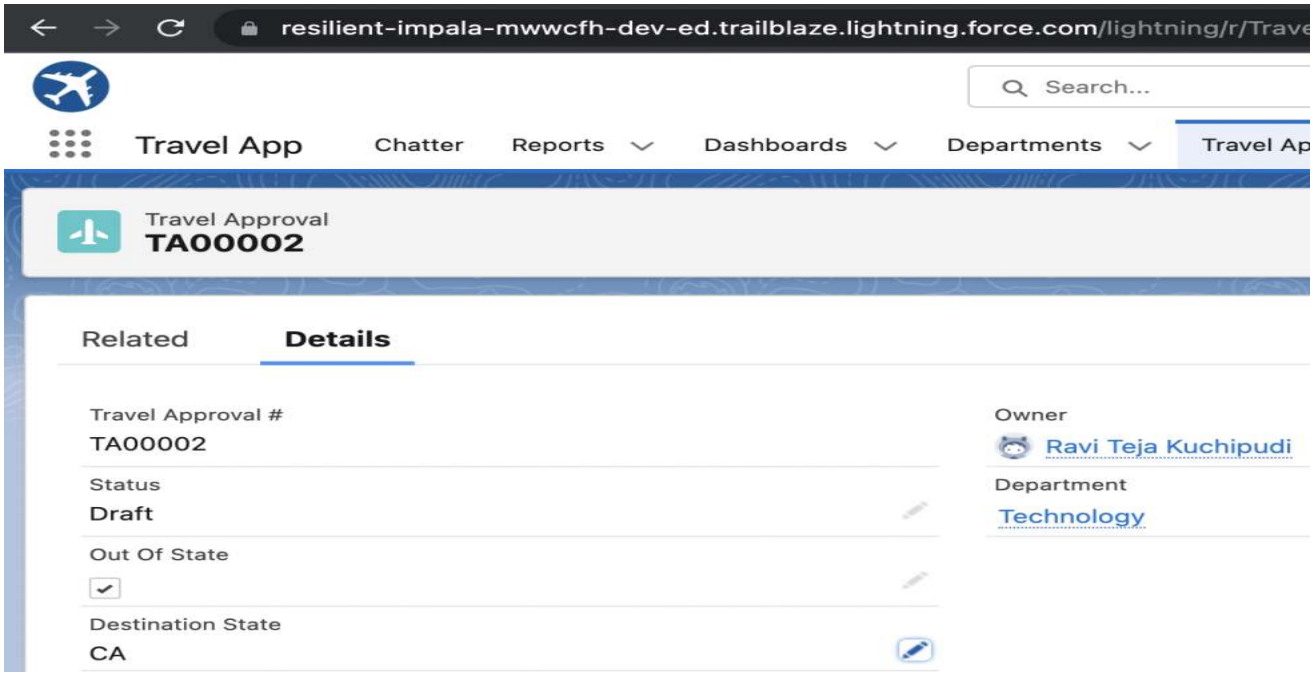

Figure 39

Step 6:

1. Create an Approval Process to send Travel approvals to the Manager or Travel coordinator. Create an Approval Step for Out-of-State Travel.
2. Create Final Approval action.
3. Create Final Rejection action.

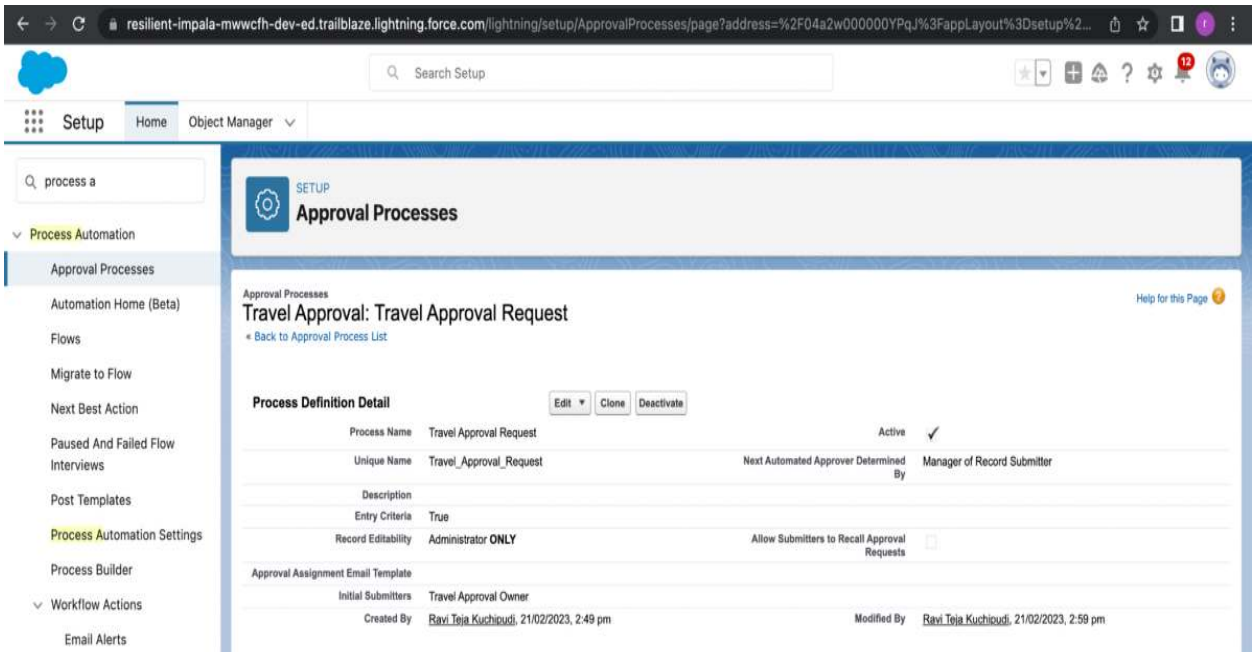

Figure 40

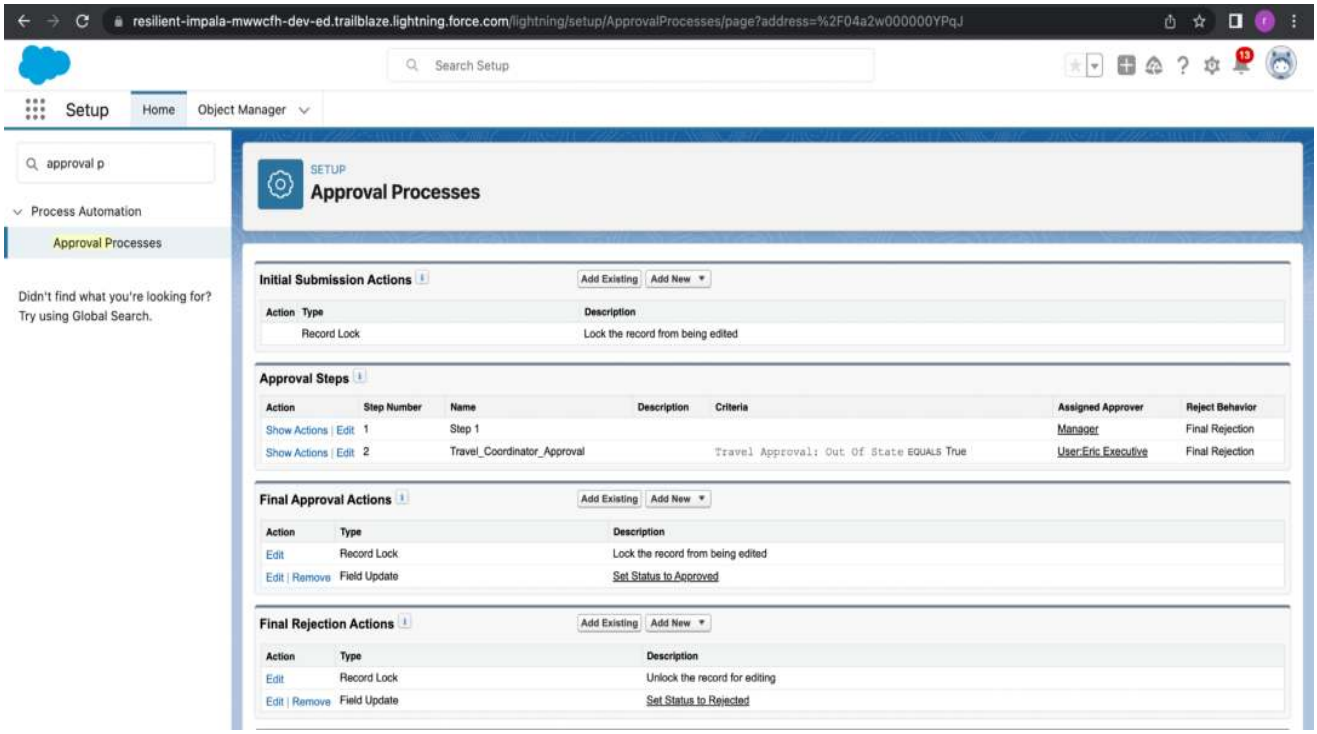

Figure 41

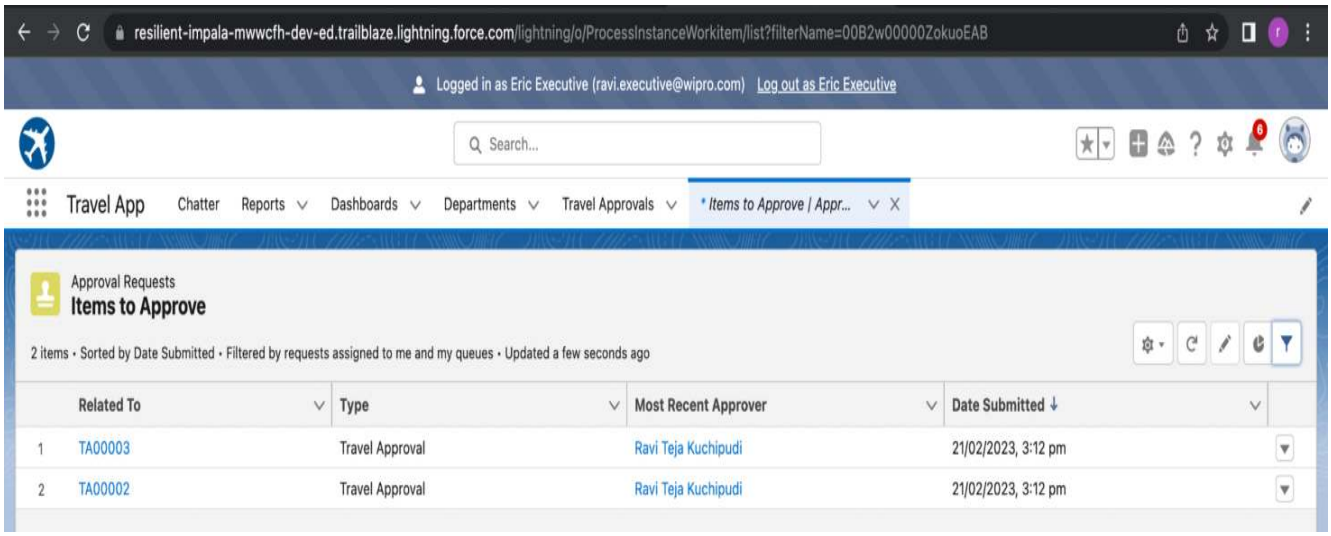

Figure 42

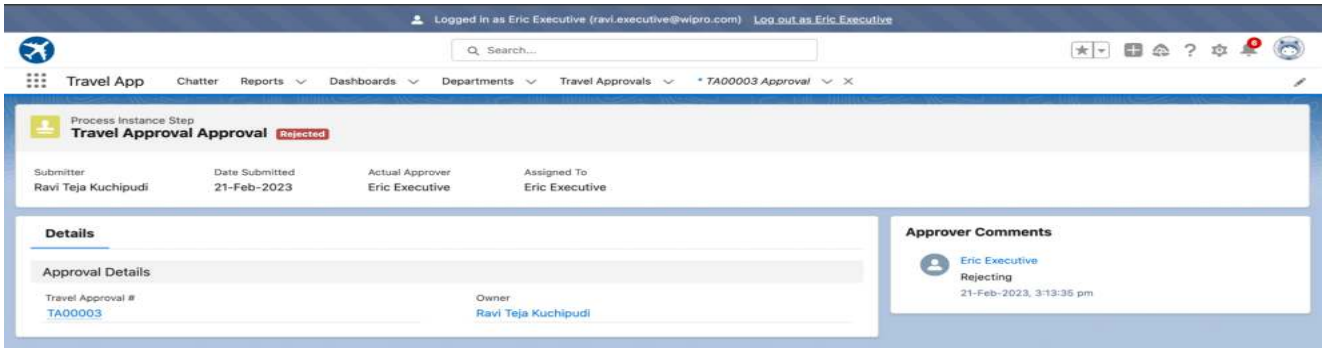

Figure 43

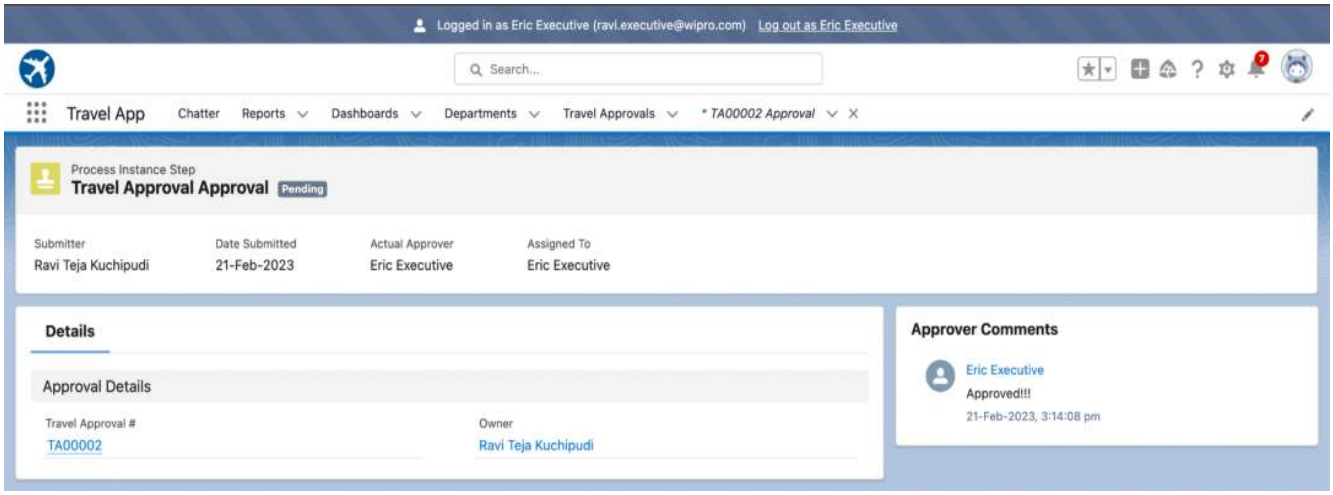

Figure 44

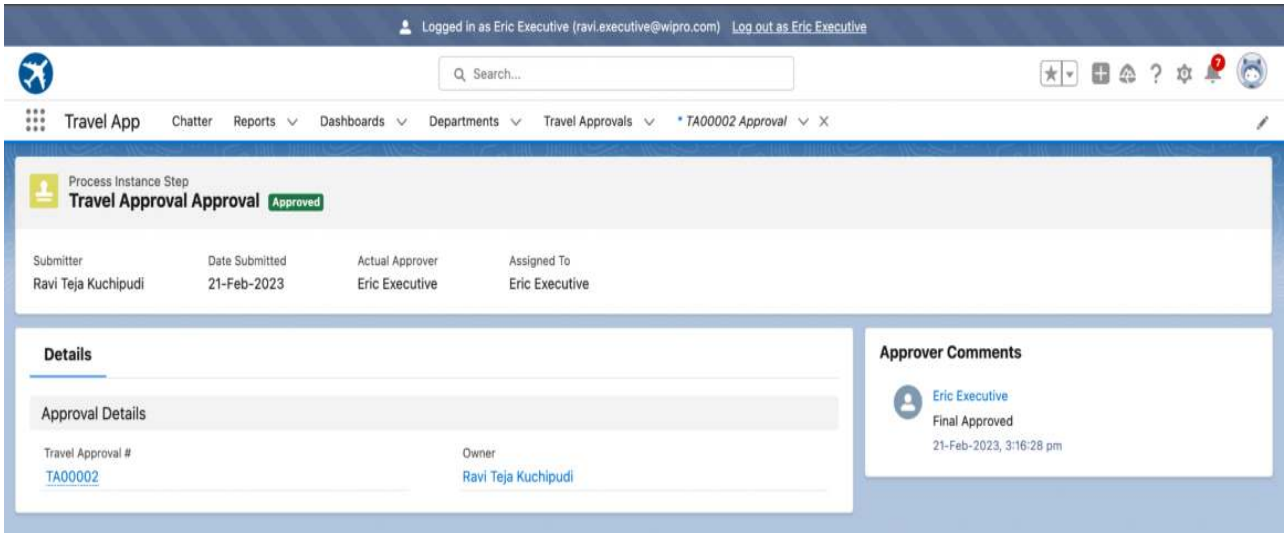

Figure 45

Exercise 2:

Step 1: Use Data Import Wizard to import Travel Approval records.

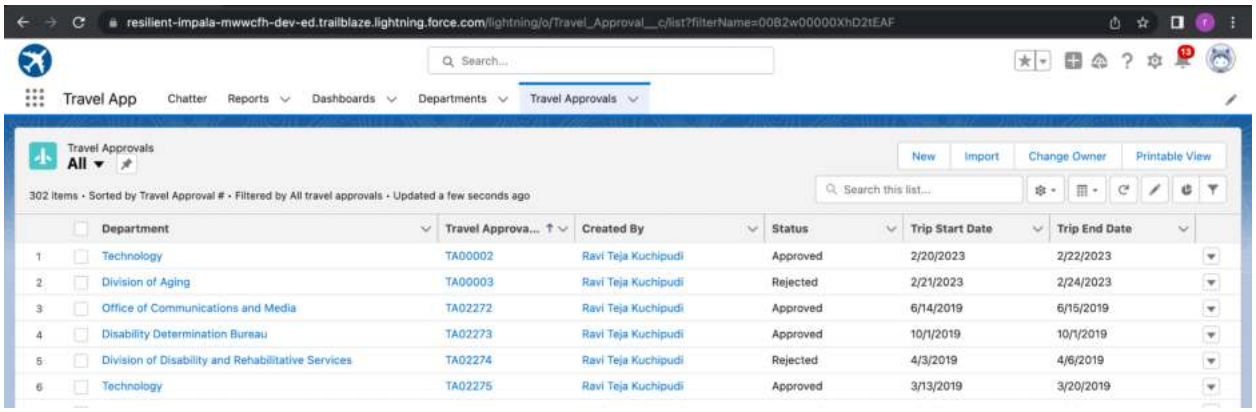

Figure 46

Step 2: Create a Travel Request by Department Report.

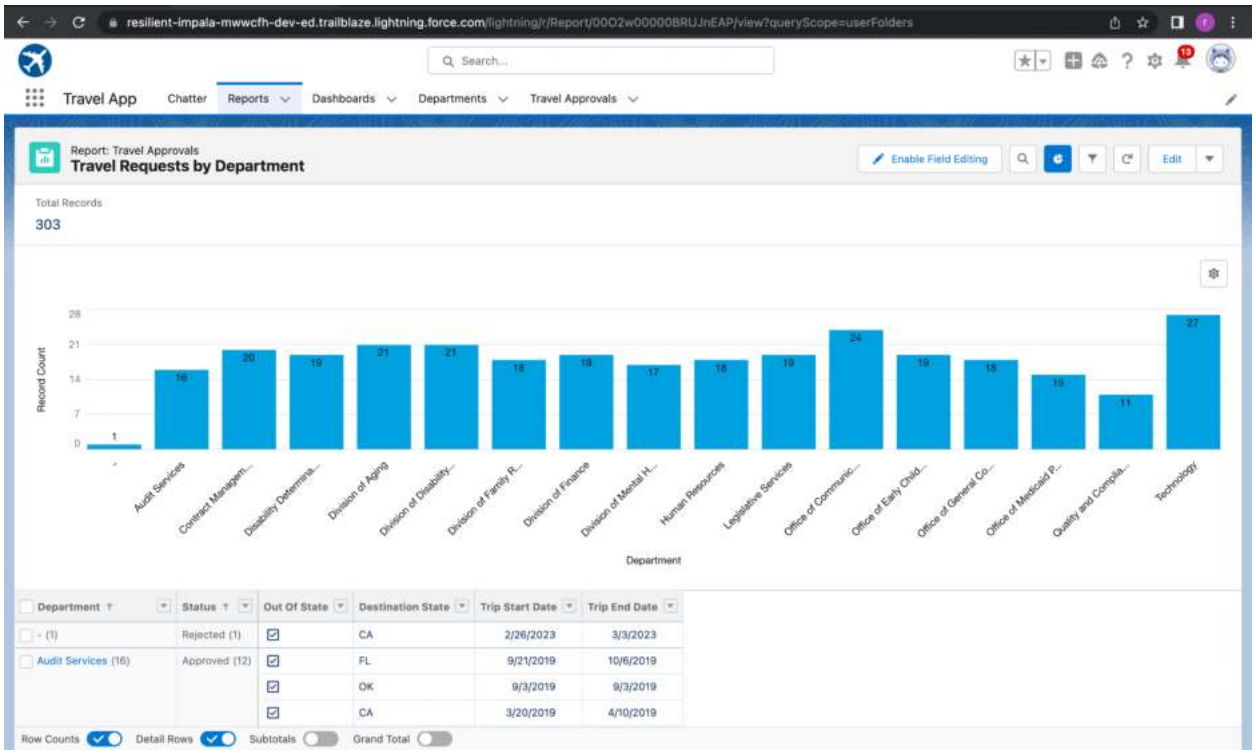

Figure 47

Step 3: Create a Travel Request by Month Report.

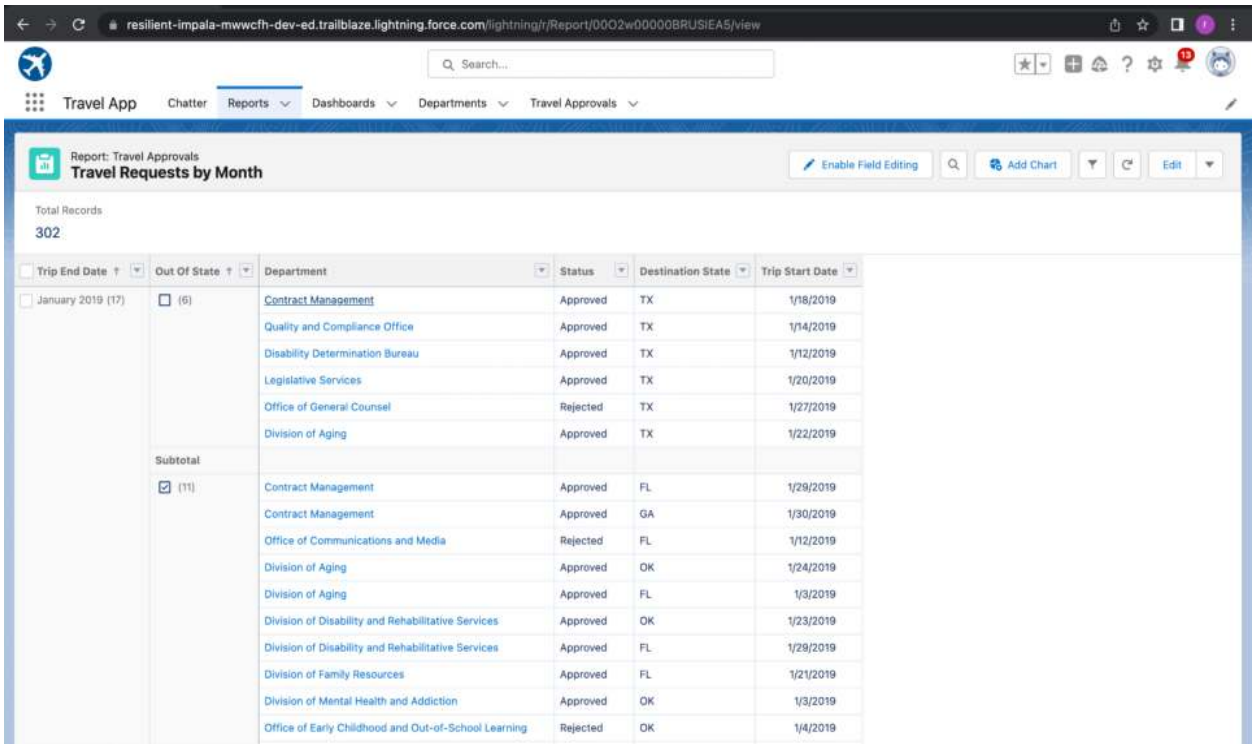

Figure 48

Step 4: Create a Travel Approvals Dashboard.

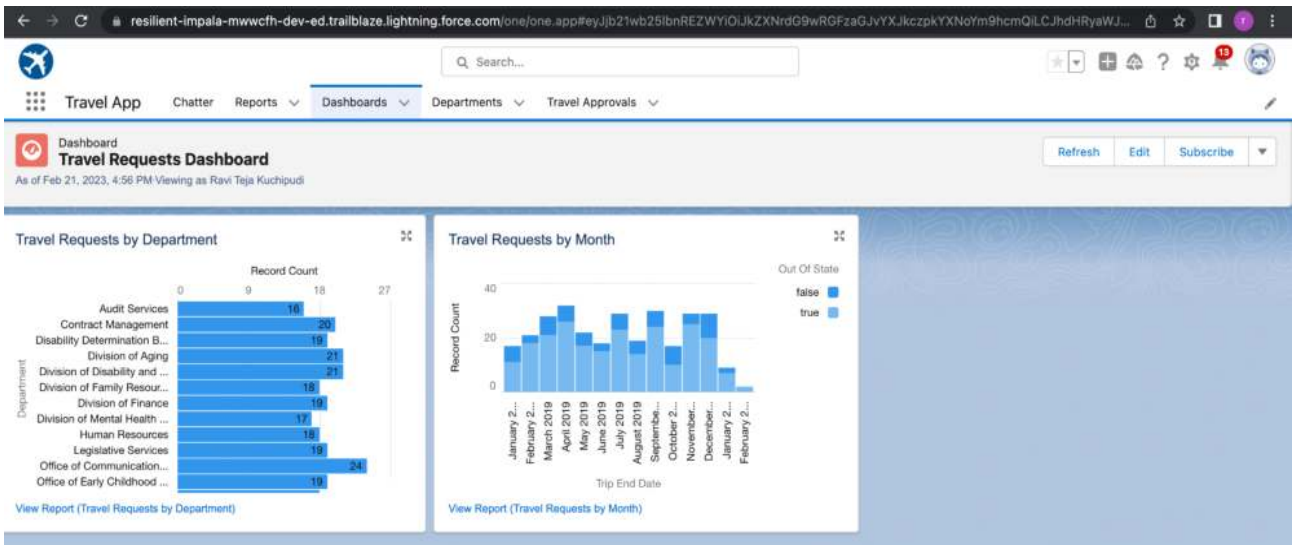

Figure 49

Module 3

Business Requirements:

- Demonstrate your Apex programming skills to:
- Write and execute codes from the Anonymous window.
- Demonstrate the use of Data types & Variables.
- Demonstrate the use of Control Flow Statement.
- Define relationships between objects. Data Manipulation using DML.
- Querying the Database using SOQL.
- Create Apex Classes, Objects & Interfaces
- Write Apex Triggers
- Write Test Classes.
- Create VisualForce Pages.

Exercise 1:

1. Create a new custom lightning App, name: Code Playground.
2. Create a Customer Custom Object
3. Create a Custom fields for Customer Object
    a. Label = Active, Checkbox, Save.
    b. Label = Customer Type, Picklist, Values: Premium, Standard.
    c. Label = Description, Text Area, Save.
    d. Label = Billing, Master-Detail, Related To – Customer custom object.
4. Create a Billing Custom Object
5. Create Custom fields for Billing Objects.
    a. Label = Amount Paid, Currency, Save.
    b. Label = Customer Type, Picklist, Values: Premium, Standard.
    c. Label = Status, Picklist, Values: Paid, Unpaid.

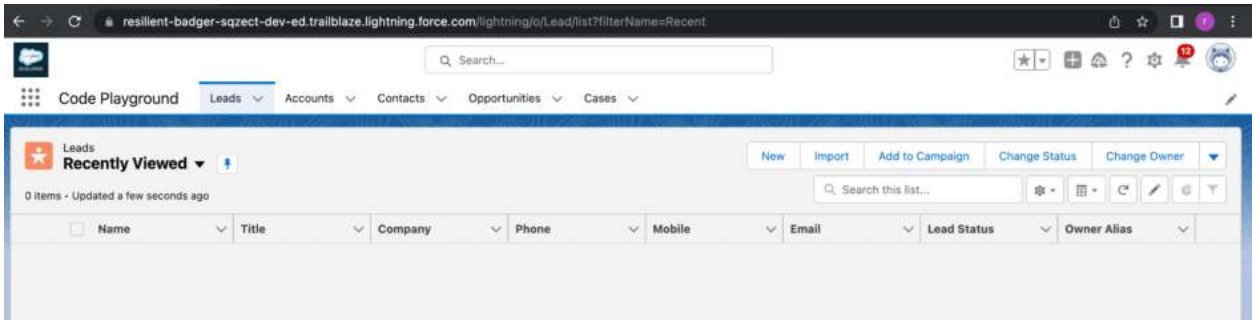

Figure 50

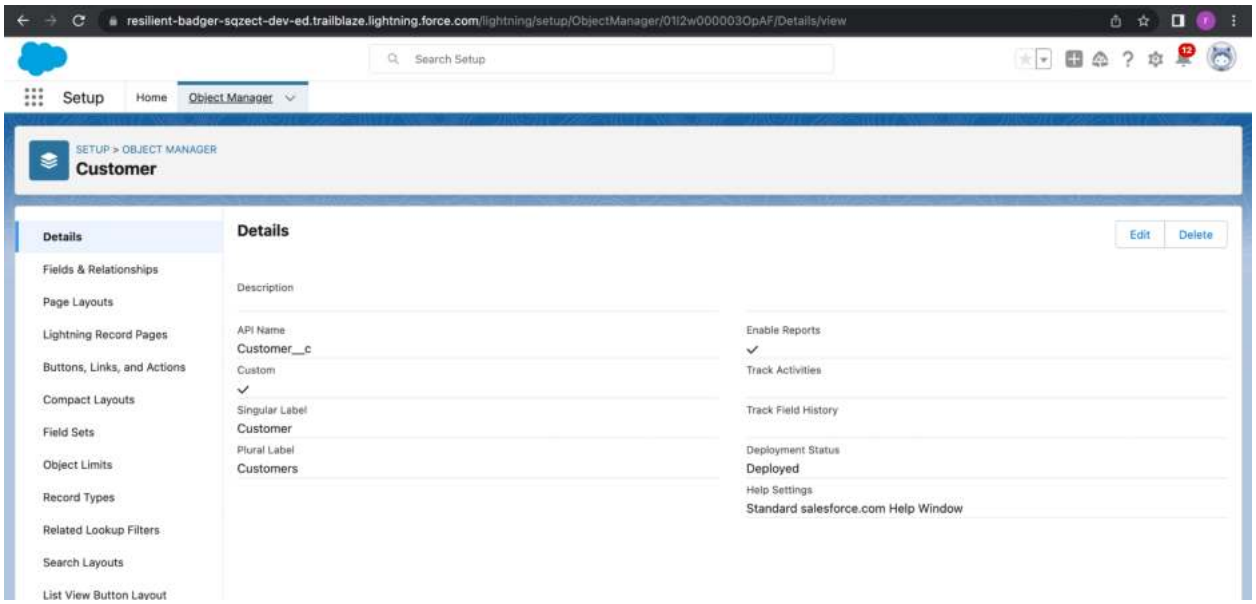

Figure 51

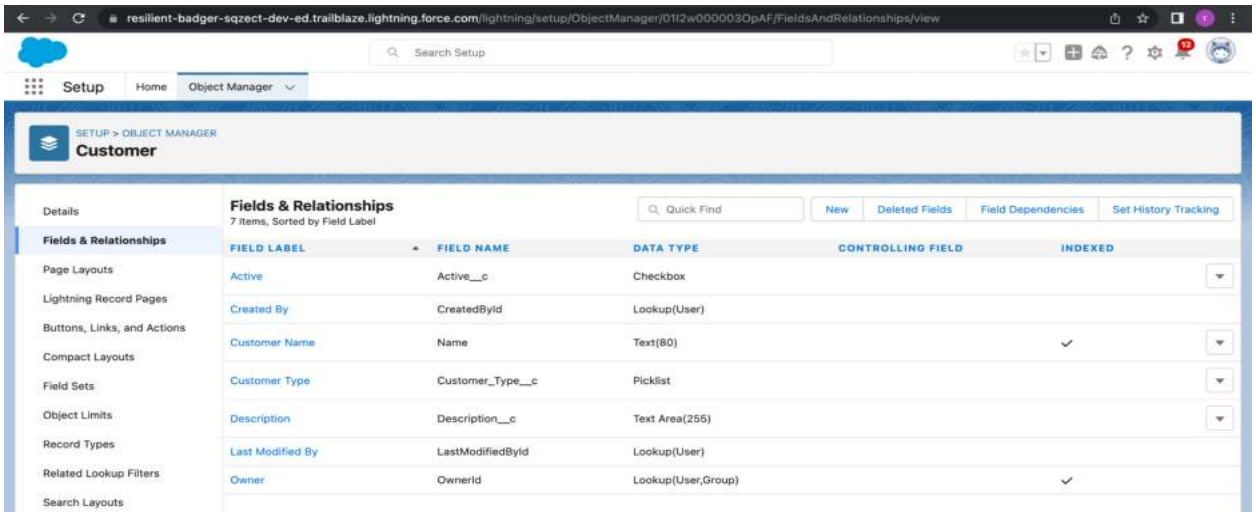

Figure 52

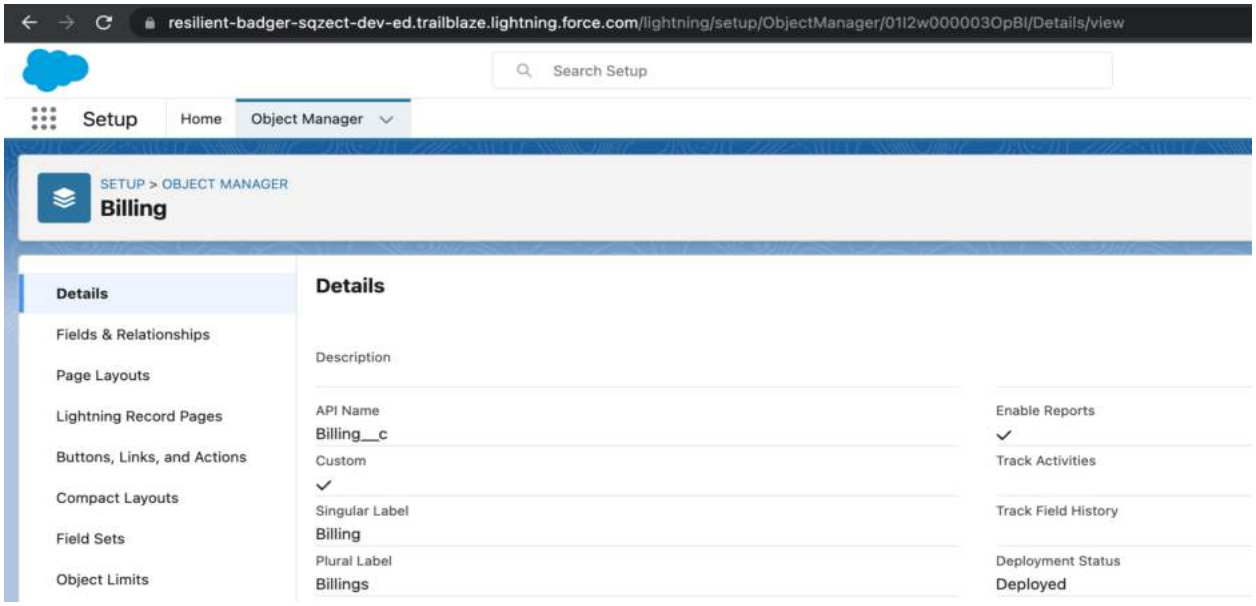

Figure 53

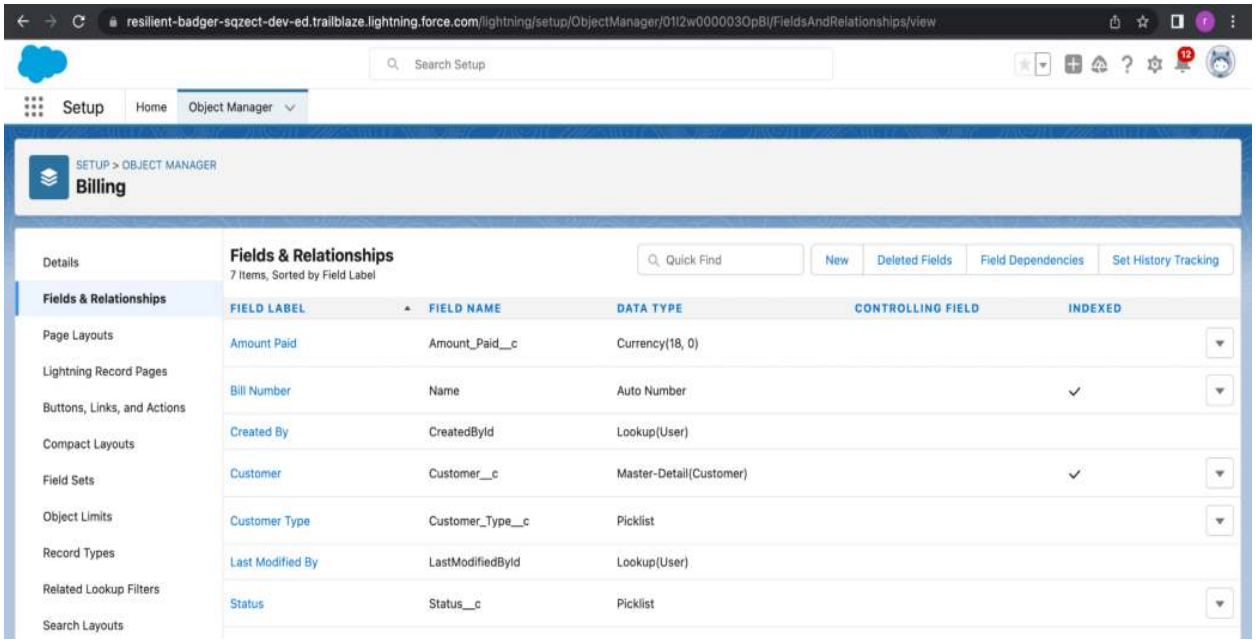

Figure 54

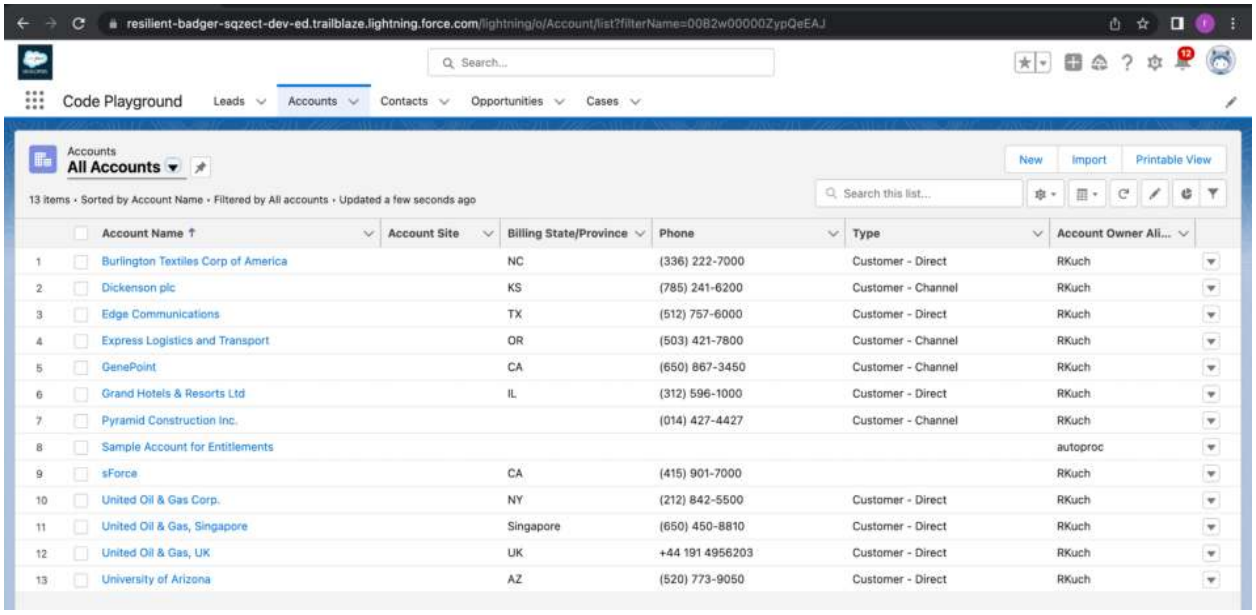

Figure 55

Exercise 2: Use Execute Anonymous to define and execute the following code:

1. Define a String Variable & use string method ‘endsWith’ to display the output.

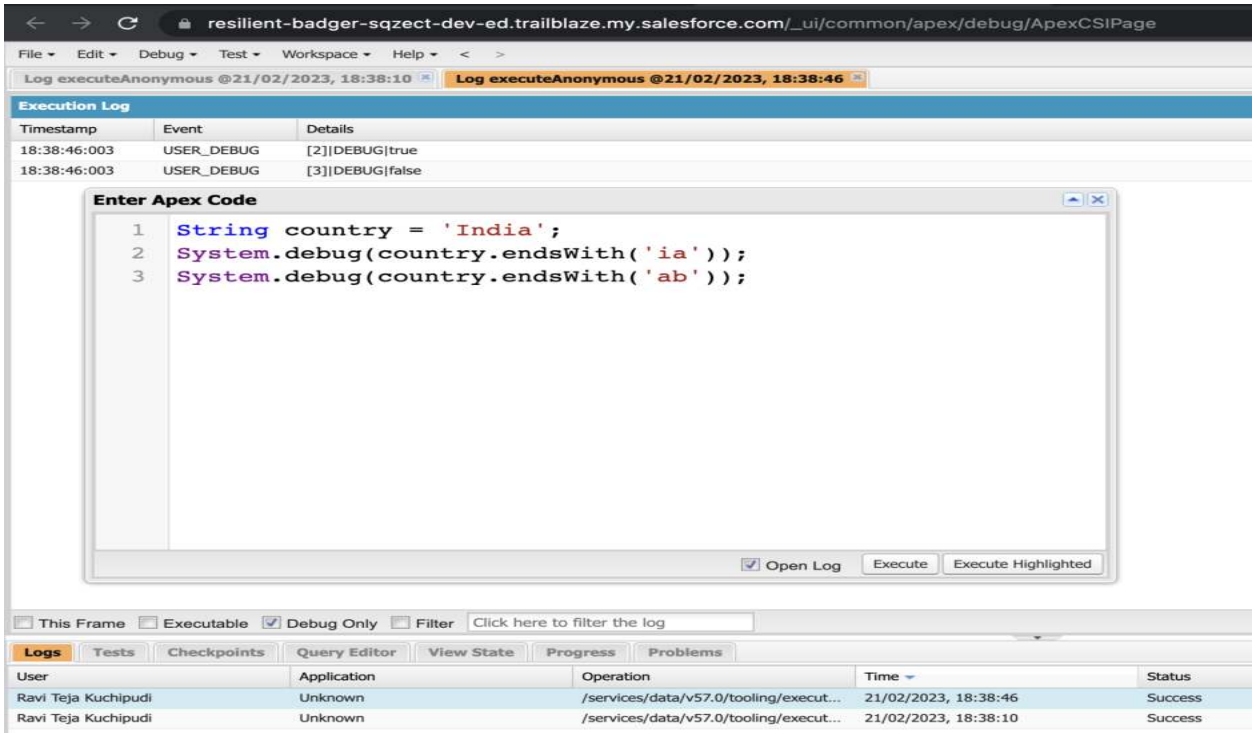

Figure 56

2. Define 2 Date type variables, use Date method today() & addDays(30) to display the output.

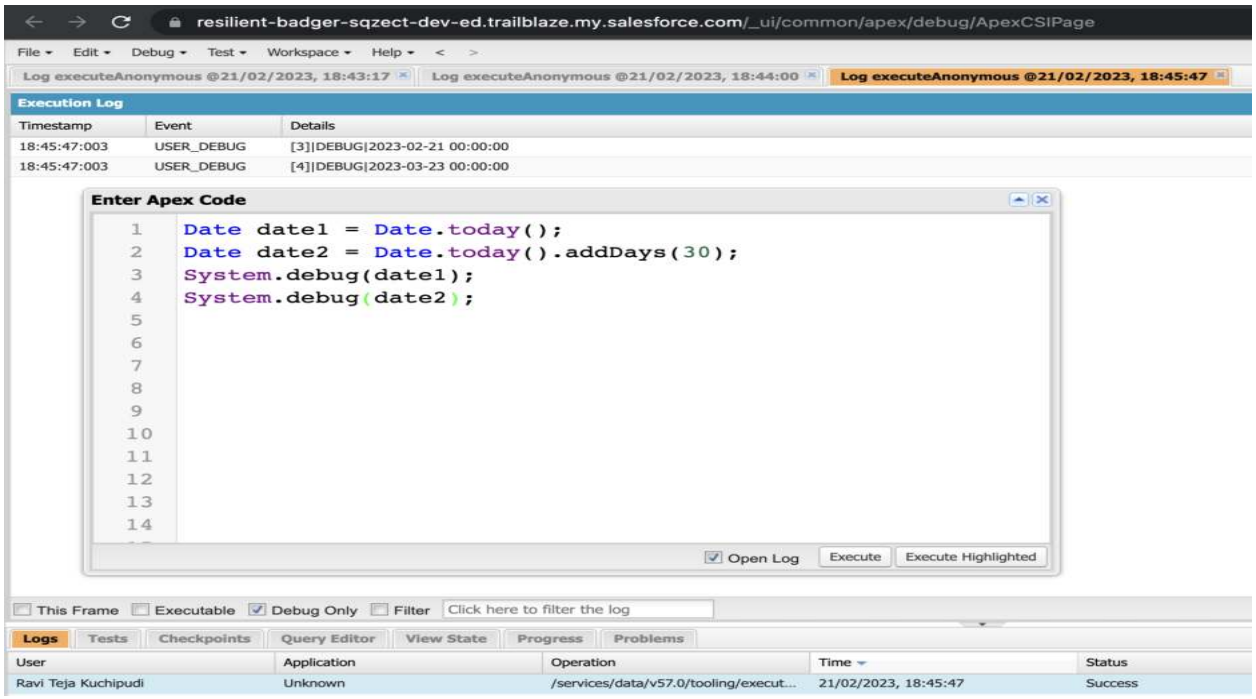

Figure 57

3. Display the output of an Integer variable from string '10' and add 20 to it.

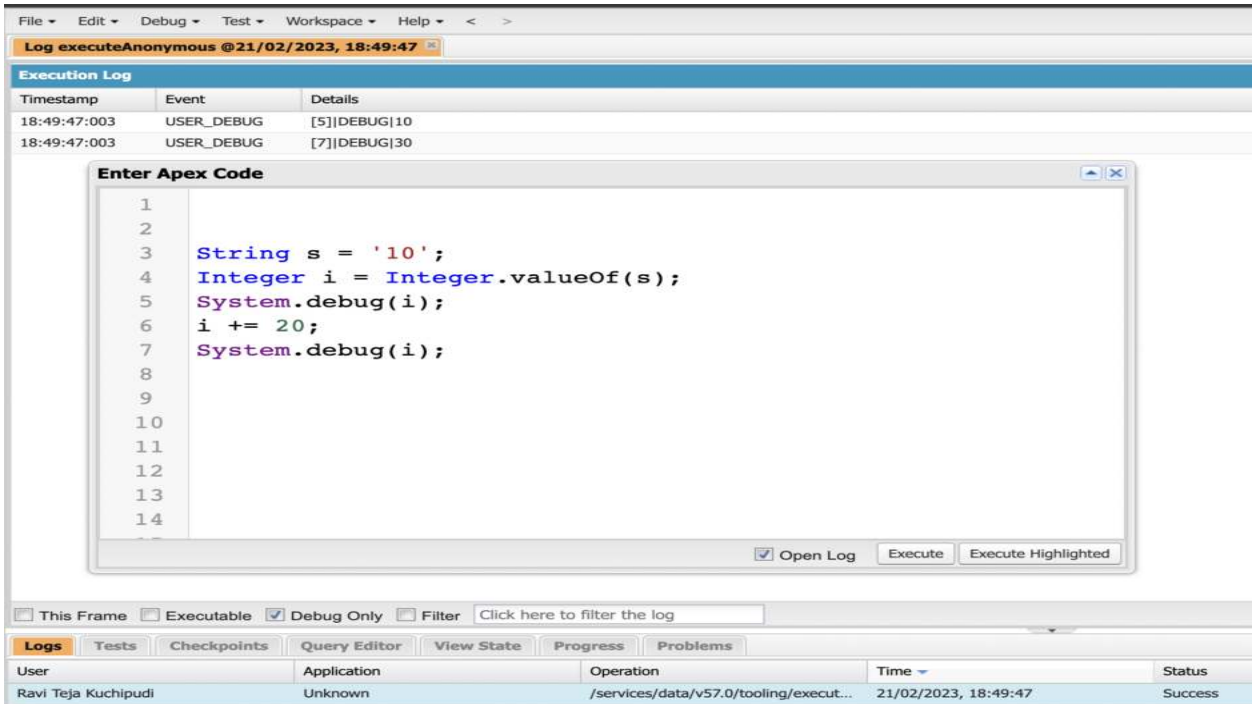

Figure 58

4. Define a String Variable & use string method length() to display the output.

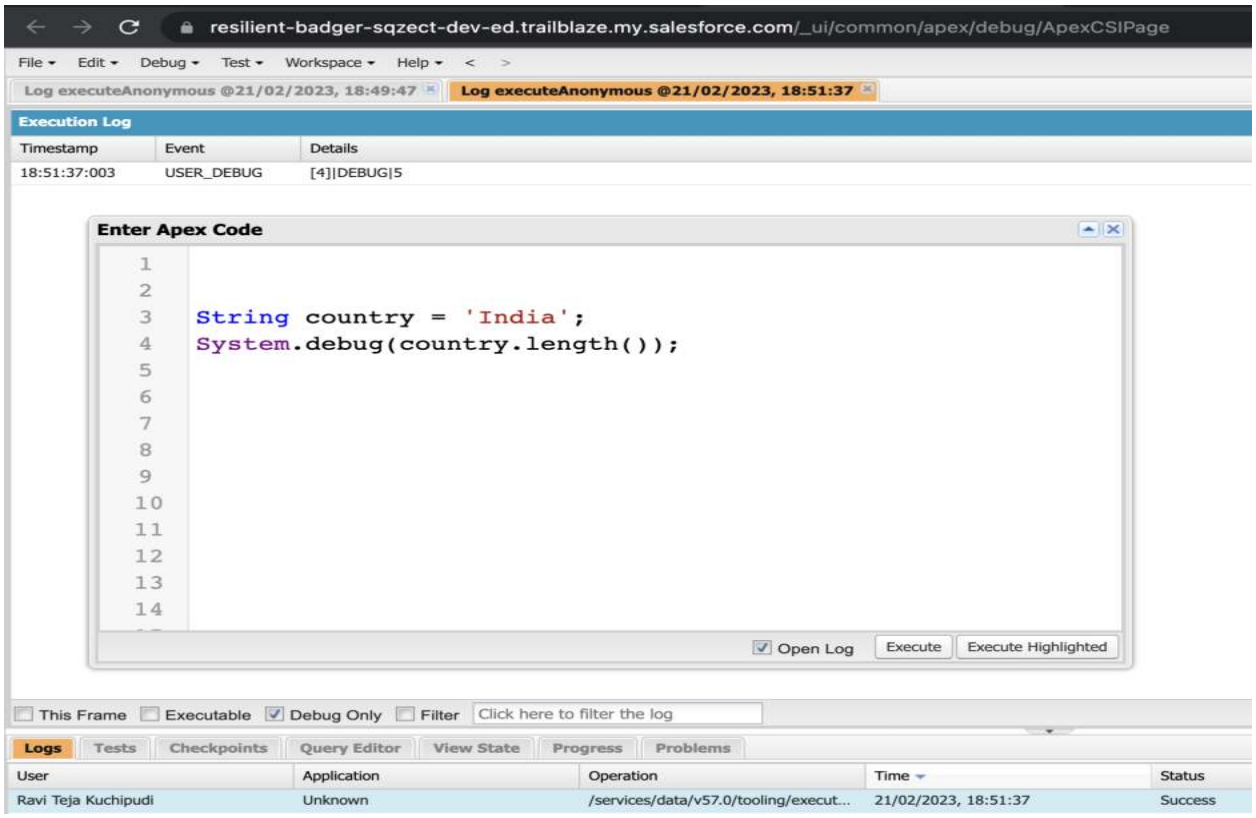

Figure 59

5. Define a List of integer and display the output using add(), get(), set(), clear(), methods

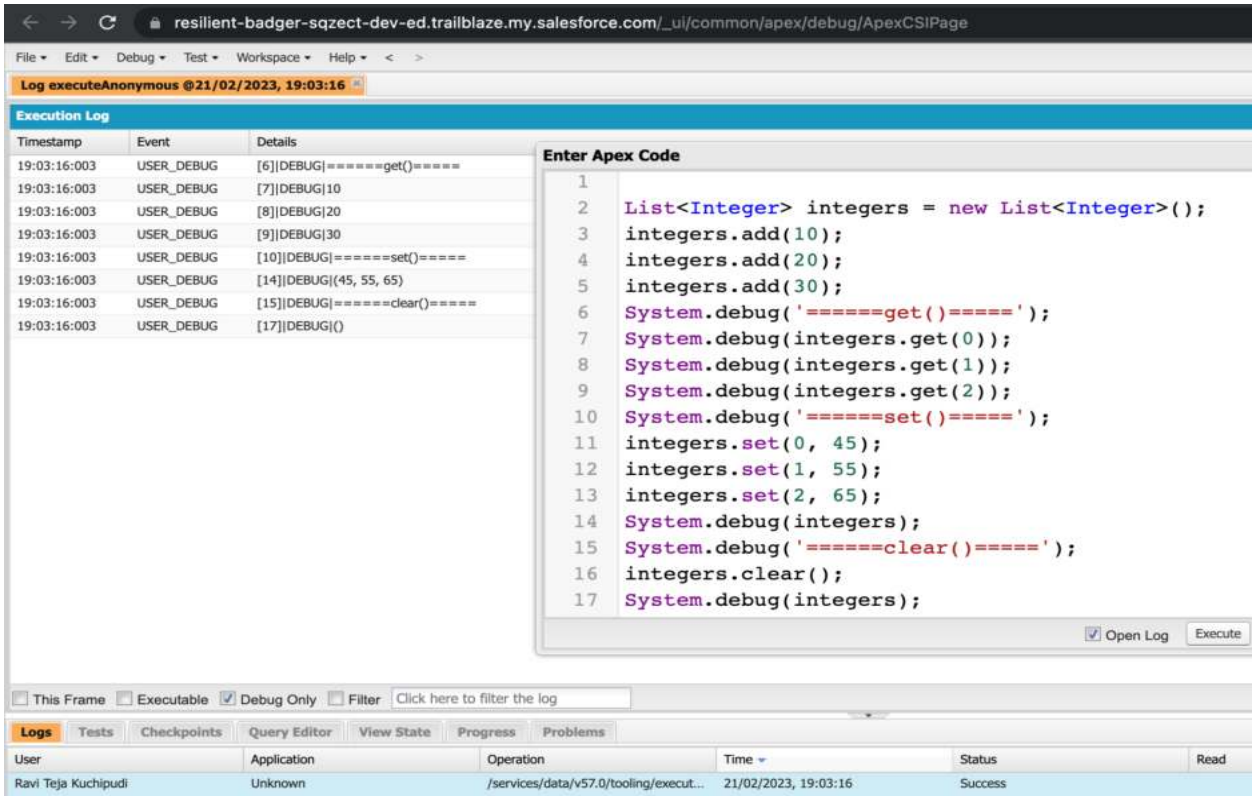

Figure 60

6. Use Execute Anonymous to define and execute the following code to display the value of x = 0 to 9.

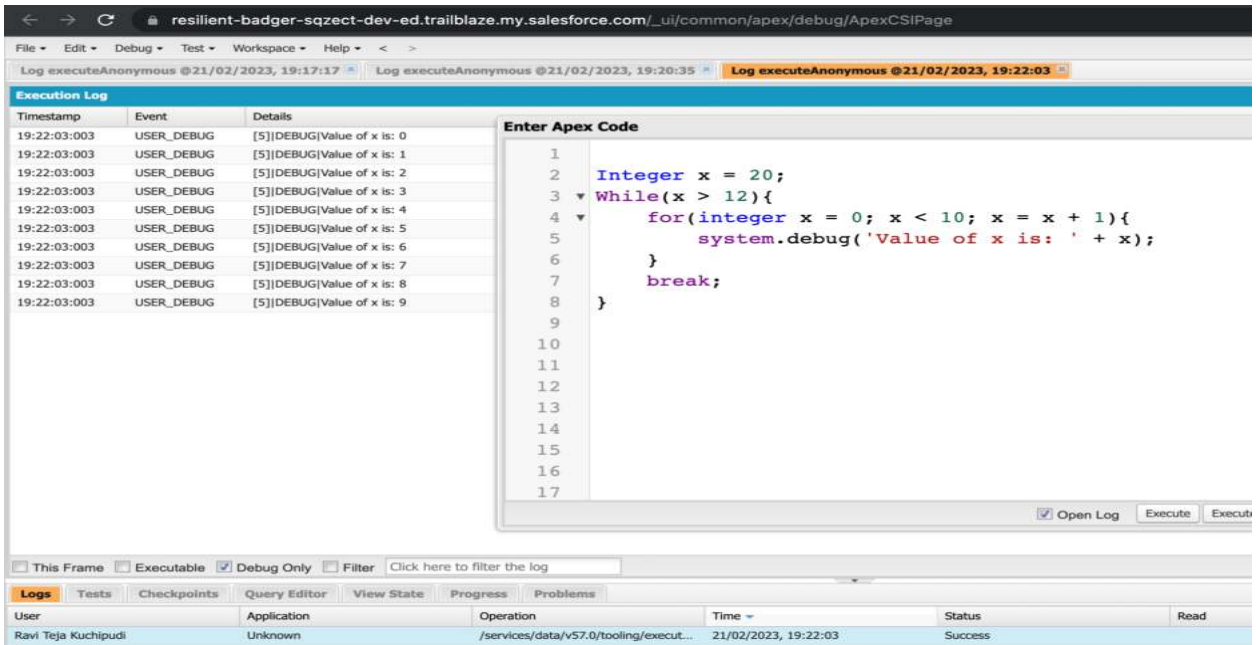

Figure 61

Exercise 3: Answer the following in True Or False: **(False)**

1. Integer myluckyNumber = 15; Integer myunlucklyNumber = 7;
2. myluckyNumber != myunlucklyNumber + 8.

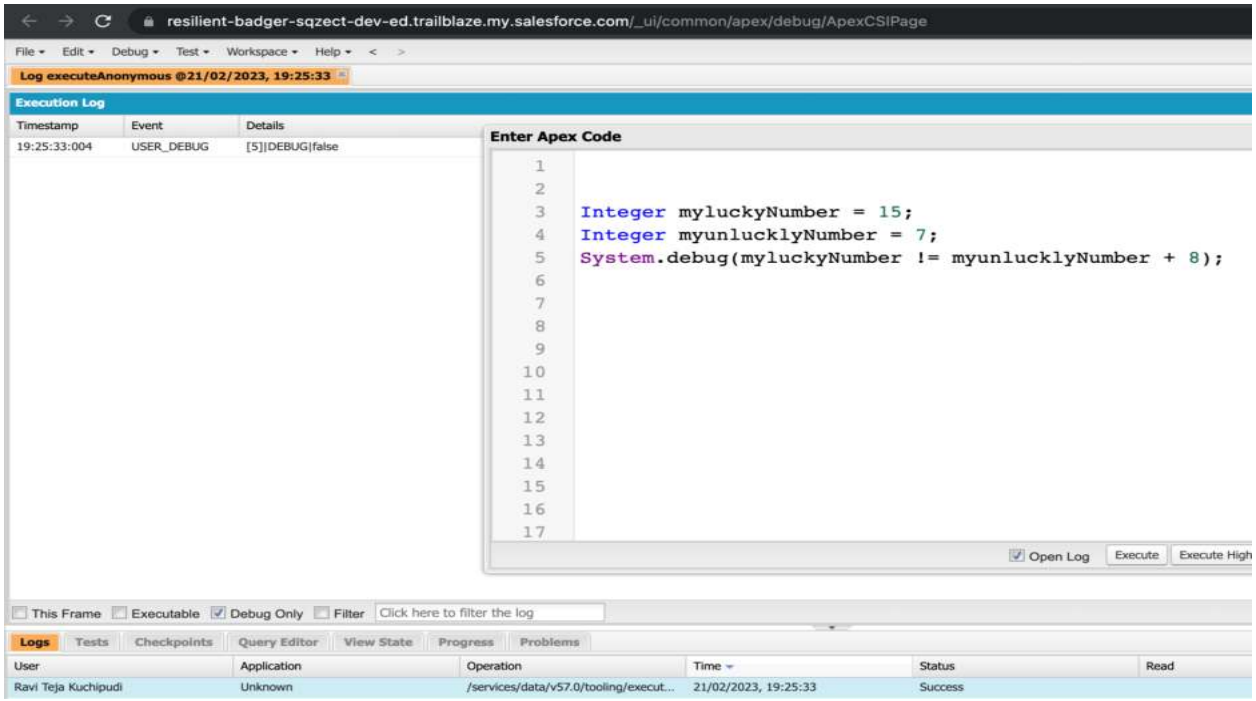

Figure 62

Exercise 4: Answer the following in True Or False: **(True)**

1. Boolean isTrue = True; Boolean isFalse = false; isTrue || isFalse

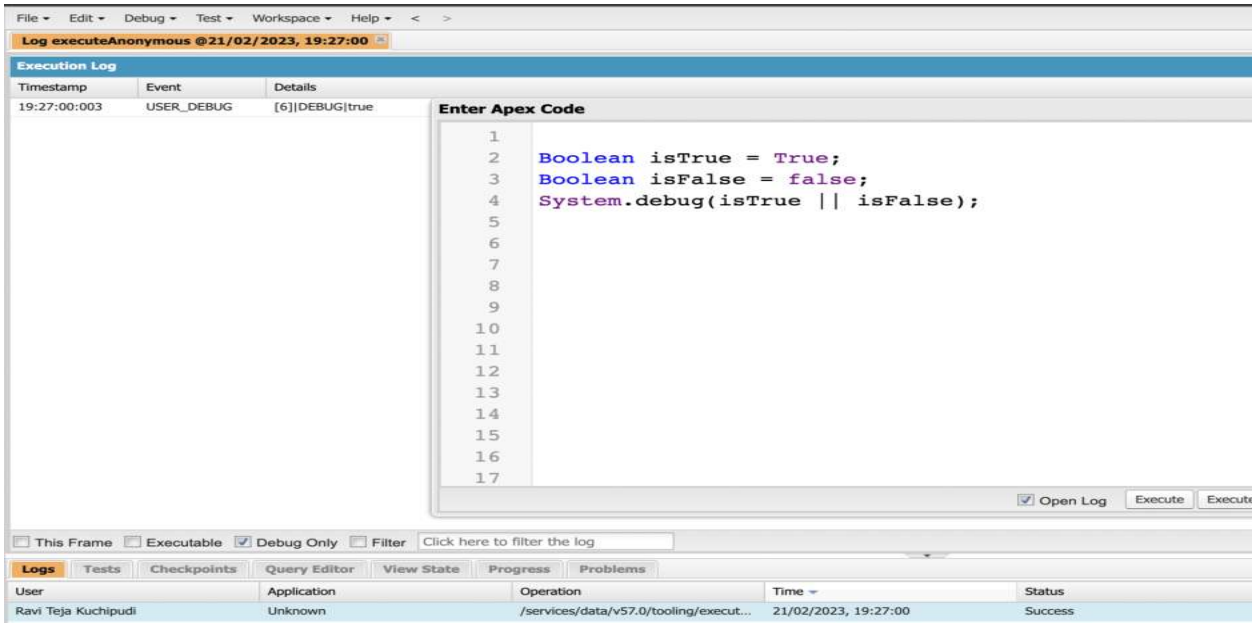

Figure 63

Exercise 5: Answer the following in True Or False: **(True)**

1. Date today = Date.today();
2. Date tomorrow = Date.today().addDays(1);
3. today != tomorrow

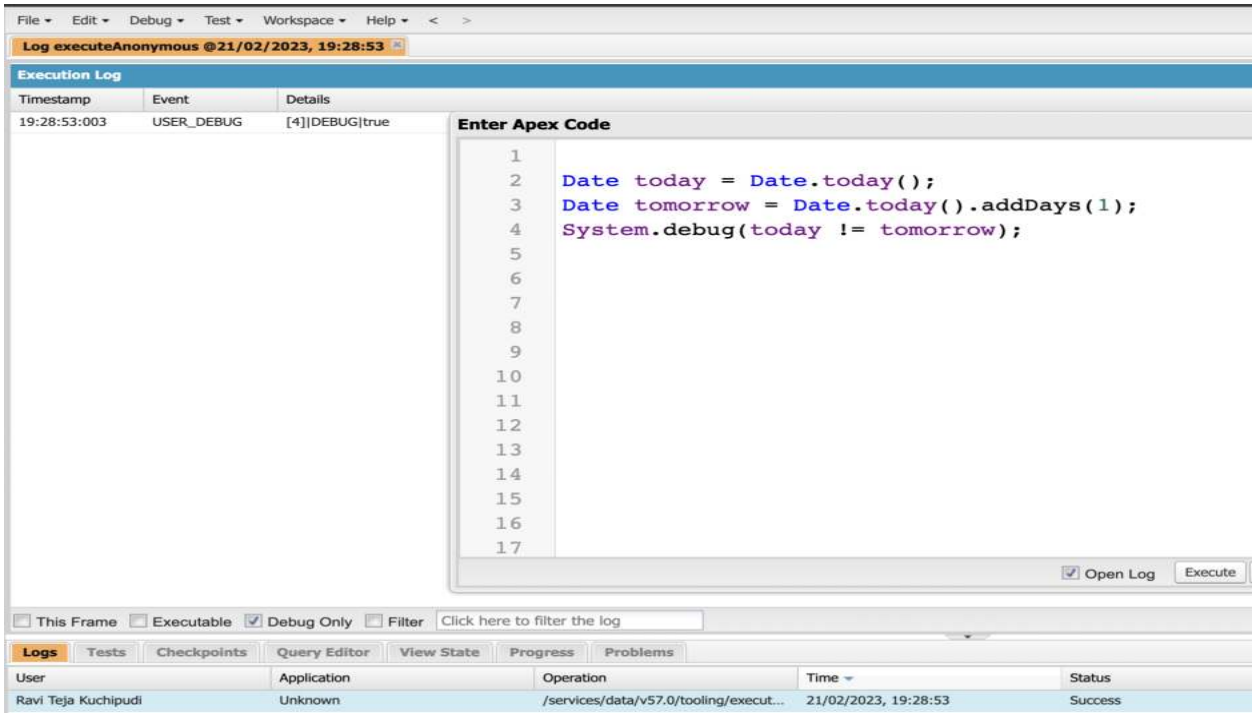

Figure 64

Exercise 6: Write a program and execute to demo the use of "If..else if...else"

1. Define an Integer variable called Score.
2. System.debug ('Grade: A+'), if Score is equal to 100.
3. System.debug ('Grade: A'), if Score is greater than or equals to 90.
4. System.debug ('Grade: B'), if Score is greater than or equals to 80.
5. System.debug ('Grade: Failed'), define as final else.

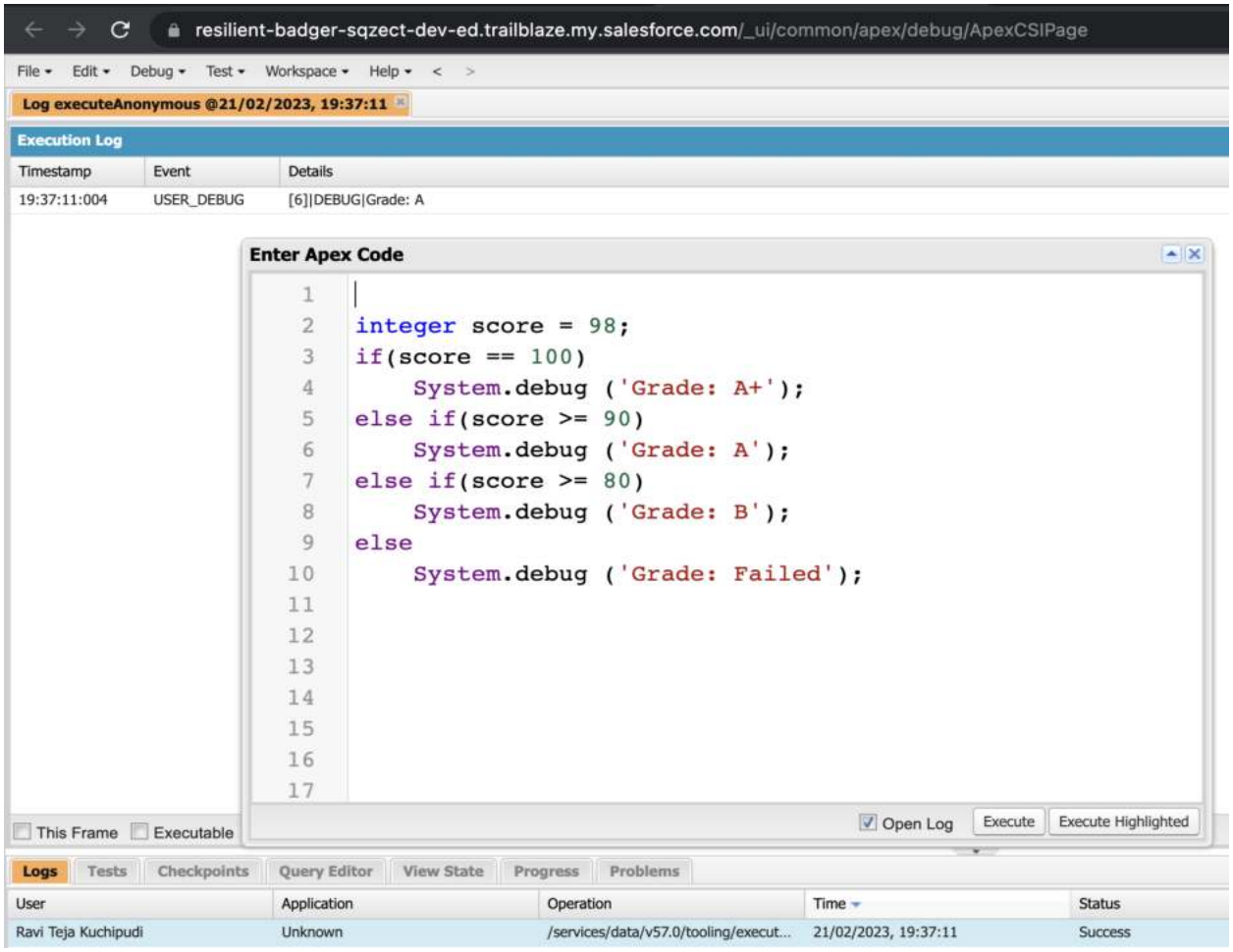

Figure 65

Exercise 7: Write a program to execute and demo the use of "Apex – for Loop"

1. Initialize the Billing object list to store the Billing Records created today, use SOQL.
2. Use for Loop to iterate over the list and check if the Billings status = Paid.
3. System.debug ('Value of Current Record in the Loop'+ Billing List);
4. if Status is paid then add the Billings into a List of String.
5. System.debug('Value of BillingList '+BillingList);
6. Note: Create at least 2 Billing records with Status = Paid for this exercise.

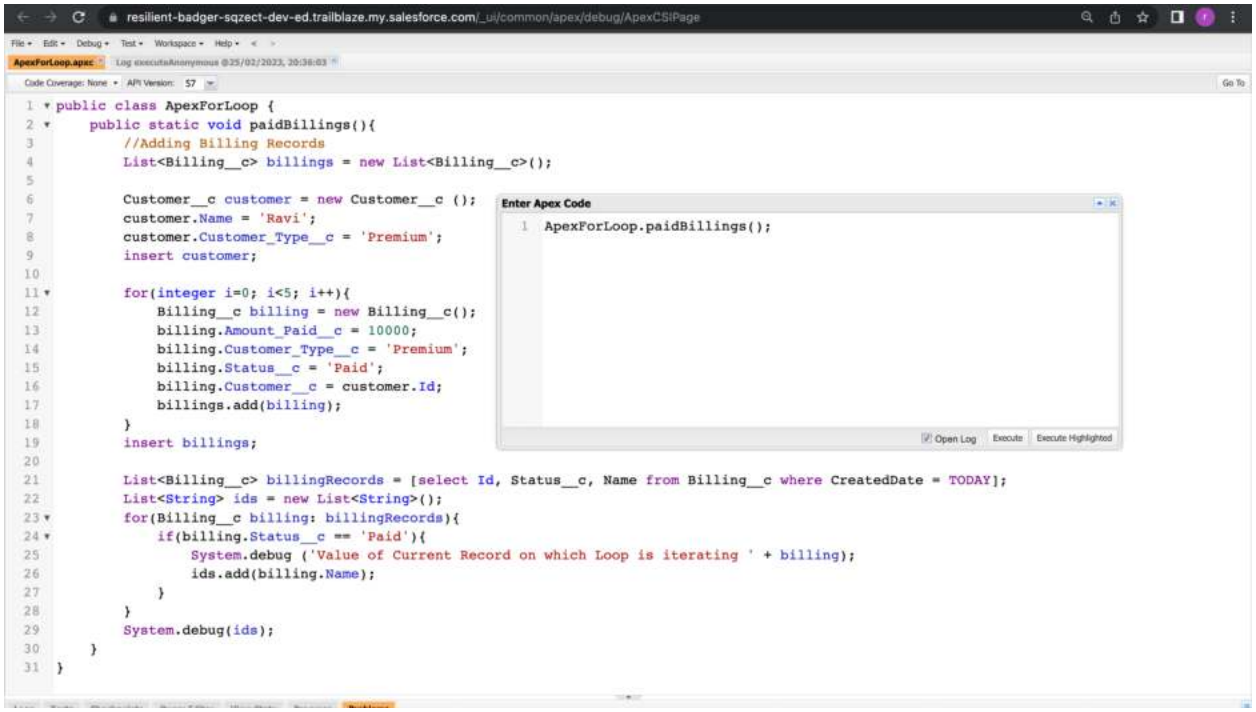

Figure 66

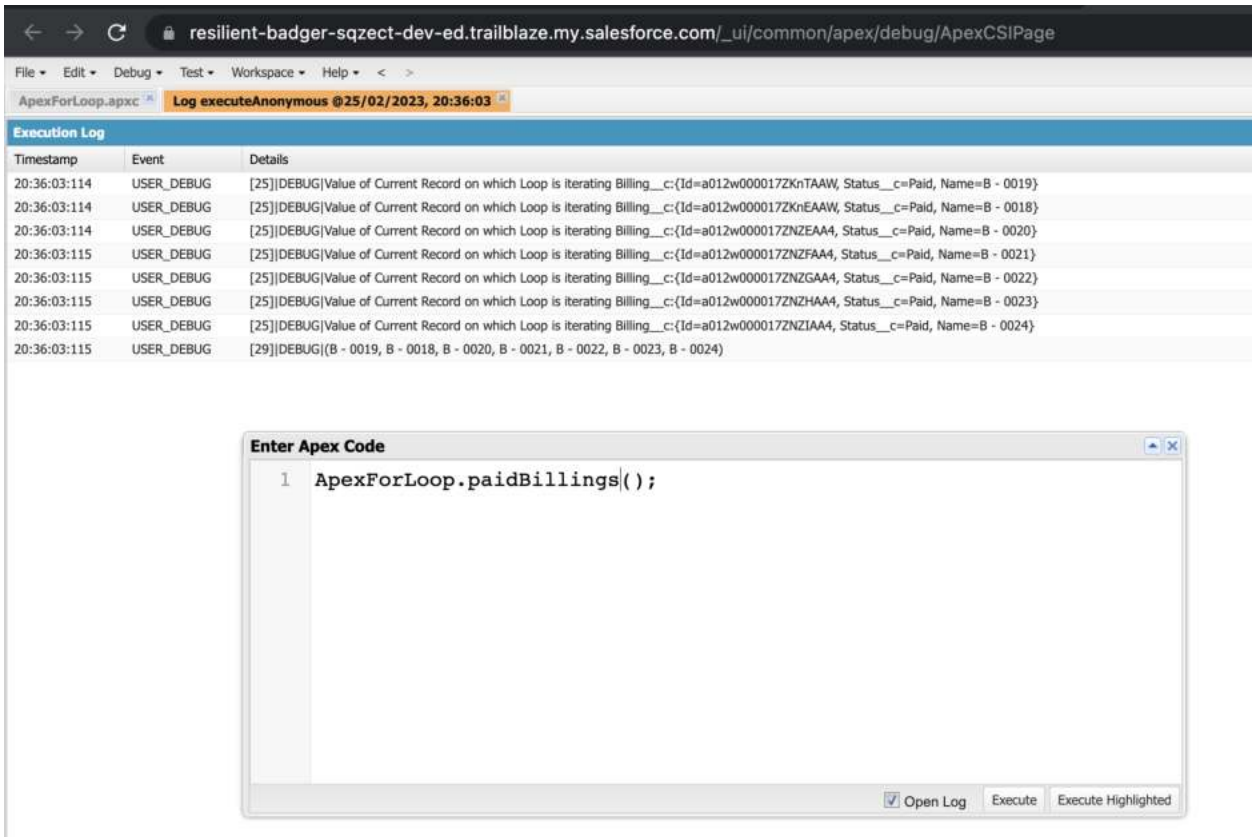

Figure 67

Exercise 8: Write a Class to demo the use of Constants in Apex

1. Class Name = DiscountClass.
2. Define a Final Decimal variable called “regularDiscount”.
3. Define a Decimal variable called “finalPrice”.
4. Define a Method called “calculateDiscount(Integer price)”, takes a Decimal data type.
5. finalPrice = price - price*regularDiscount;
6. Don’t forget to return a Decimal data type.

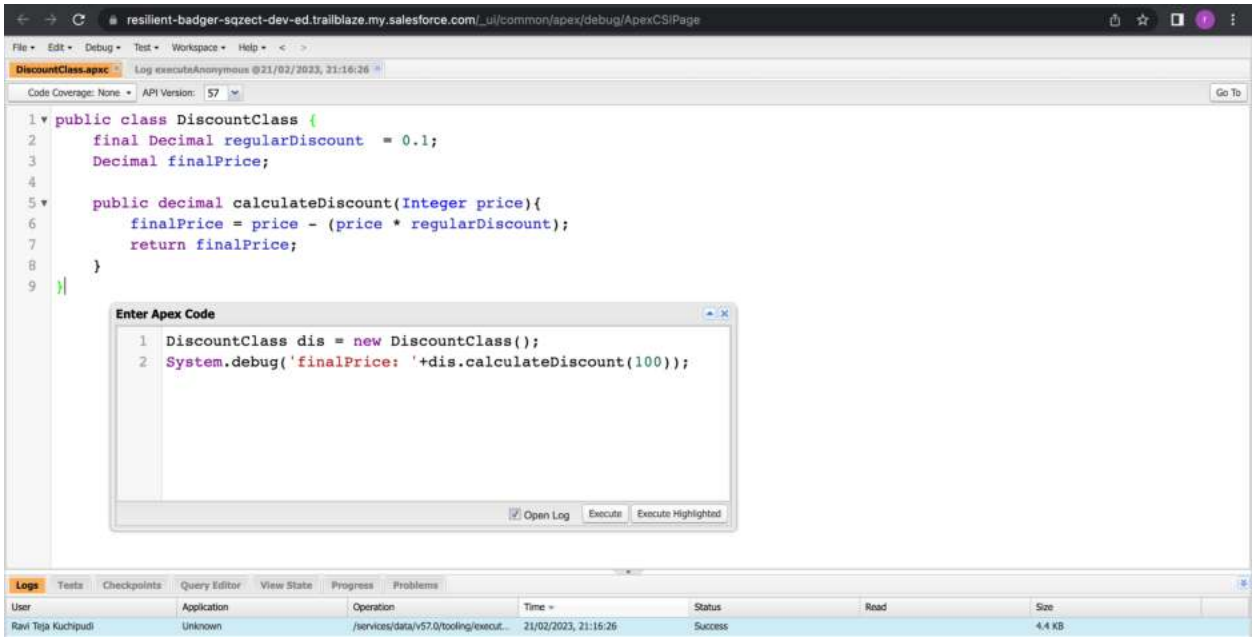

Figure 68

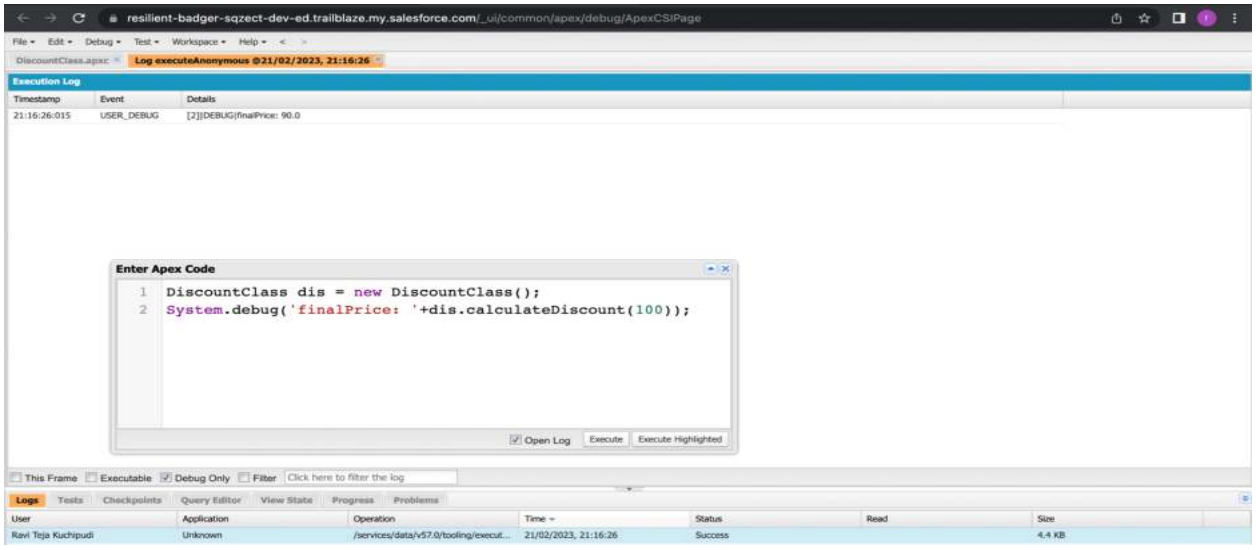

Figure 69

Exercise 9: Write a Class to demo the use of Interface in Apex

1. Interface Class Name = InterfaceExample. Method Signature = Double percentageDiscountTobeApplied();
2. Create an Apex Class to implement the Interface Class, Name = PremiumCustomer . Call the interface method to return 30% Discount.
3. Create another Apex Class to implement the Interface Class, Name = normalCustomer Call the interface method to return 10% Discount.

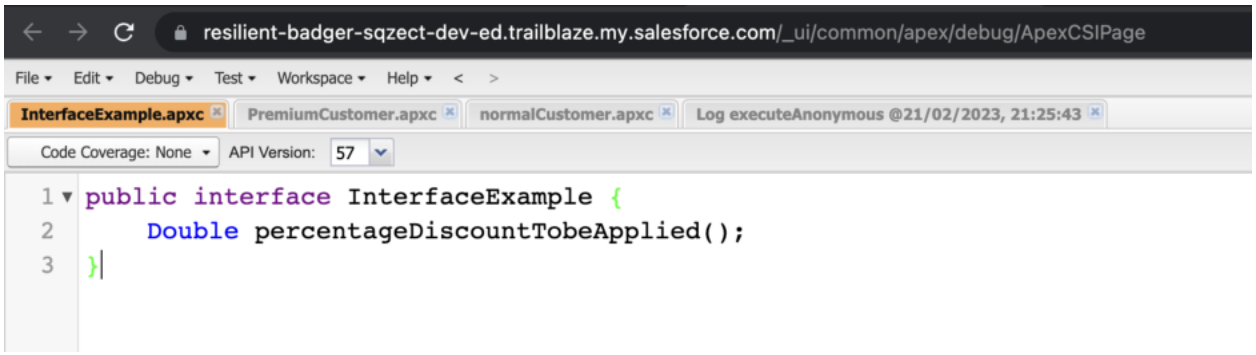

Figure 70

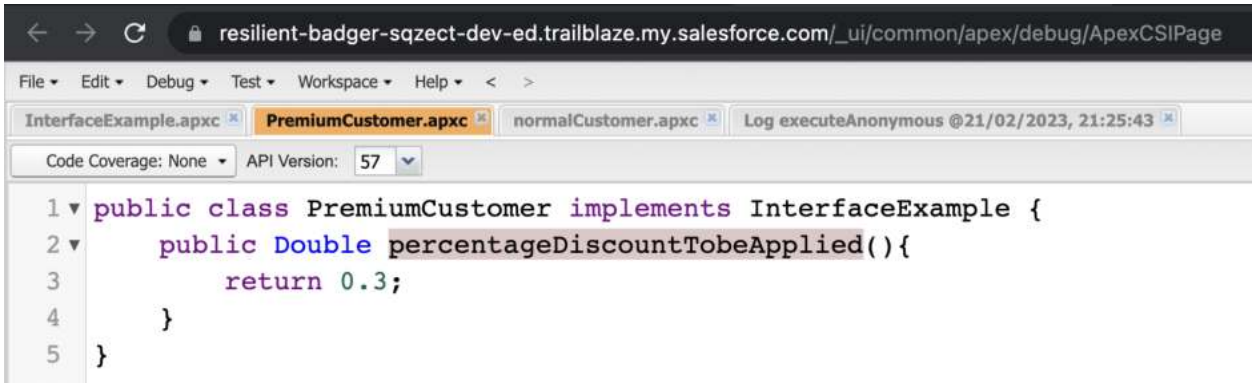

Figure 71

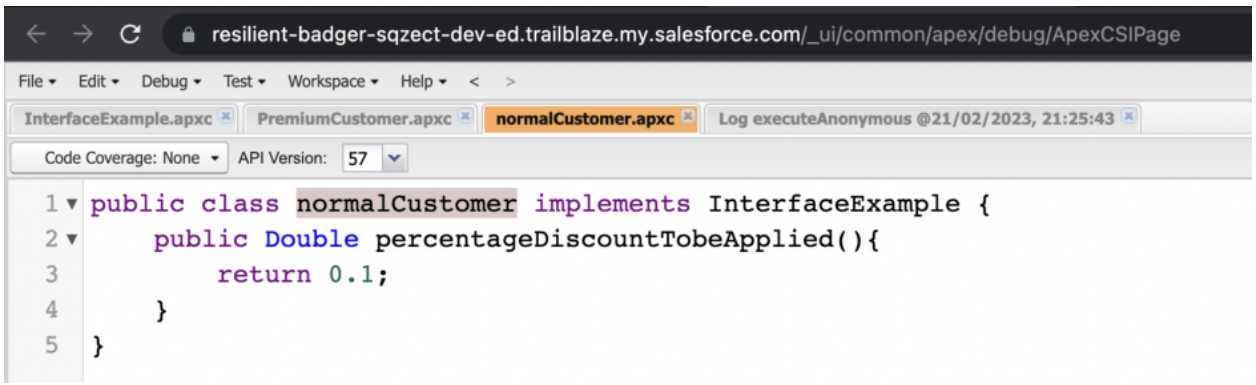

Figure 72

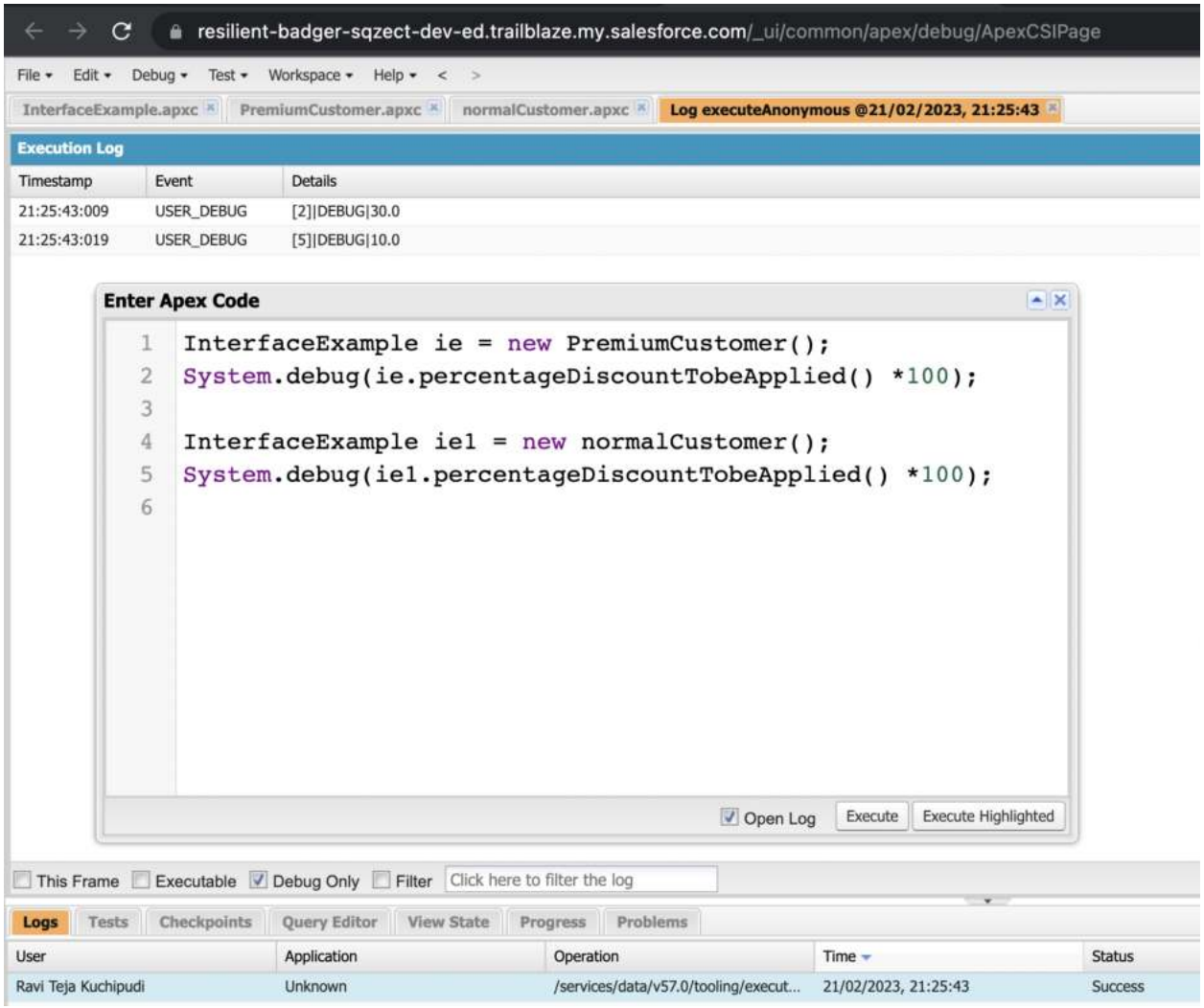

Figure 73

Exercise 10: Demo on DML Insert Operation Using Database methods

1. Insert a Customer Record First using simple DML Statement.
2. Customer Name = “Wipro”, Type = “Premium”.
3. Insert Billing record using Database methods.
4. Billing Status = Paid, Amount Paid = 5000000, link Billing record to Customer using the customer ID.
5. Use a List<Billing c> to perform the Database insert method & store the result in srList.
6. Iterate through the Success/Error result, show inserted record ID, or the error message with fields.

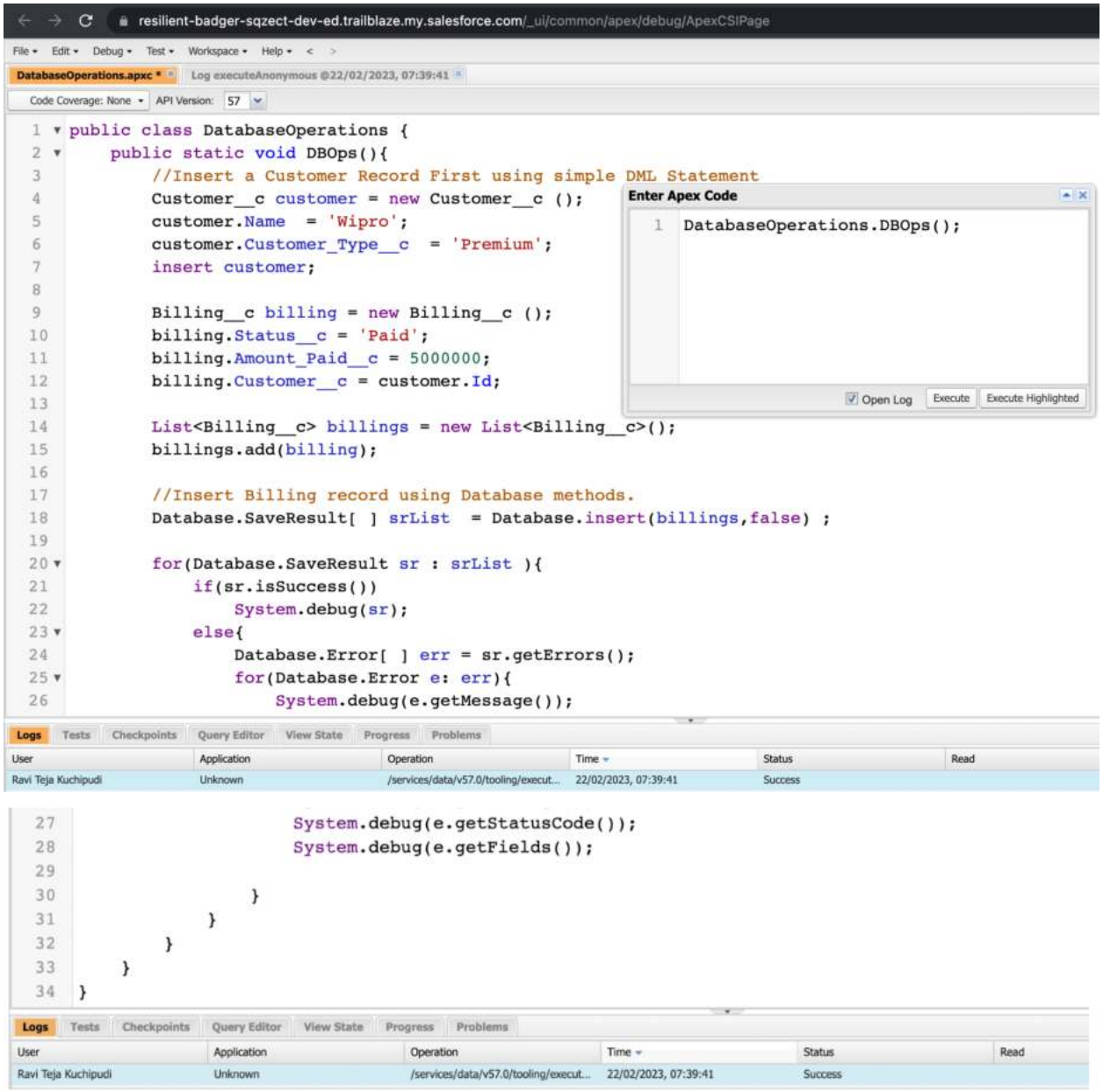
resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
DatabaseOperations.apxc *
Log executeAnonymous @22/02/2023, 07:39:41
Code Coverage: None
API Version: 57
public class DatabaseOperations {
public static void DBOps(){
//Insert a Customer Record First using simple DML Statement
Customer__c customer = new Customer__c ();
customer.Name = 'Wipro';
customer.Customer_Type__c = 'Premium';
insert customer;
Billing__c billing = new Billing__c ();
billing.Status__c = 'Paid';
billing.Amount_Paid__c = 5000000;
billing.Customer__c = customer.Id;
List<Billing__c> billings = new List<Billing__c>();
billings.add(billing);
//Insert Billing record using Database methods.
Database.SaveResult[ ] srList = Database.insert(billings,false) ;
for(Database.SaveResult sr : srList){
if(sr.isSuccess())
System.debug(sr);
else{
Database.Error[ ] err = sr.getErrors();
for(Database.Error e: err){
System.debug(e.getMessage());
Enter Apex Code
DatabaseOperations.DBOps();
Open Log
Execute
Execute Highlighted
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
User
Application
Operation
Time
Status
Read
Ravi Teja Kuchipudi
Unknown
/services/data/v57.0/tooling/execut...
22/02/2023, 07:39:41
Success
System.debug(e.getStatusCode());
System.debug(e.getFields());
}
}
}
}
}
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
User
Application
Operation
Time
Status
Read
Ravi Teja Kuchipudi
Unknown
/services/data/v57.0/tooling/execut...
22/02/2023, 07:39:41
Success

Figure 74

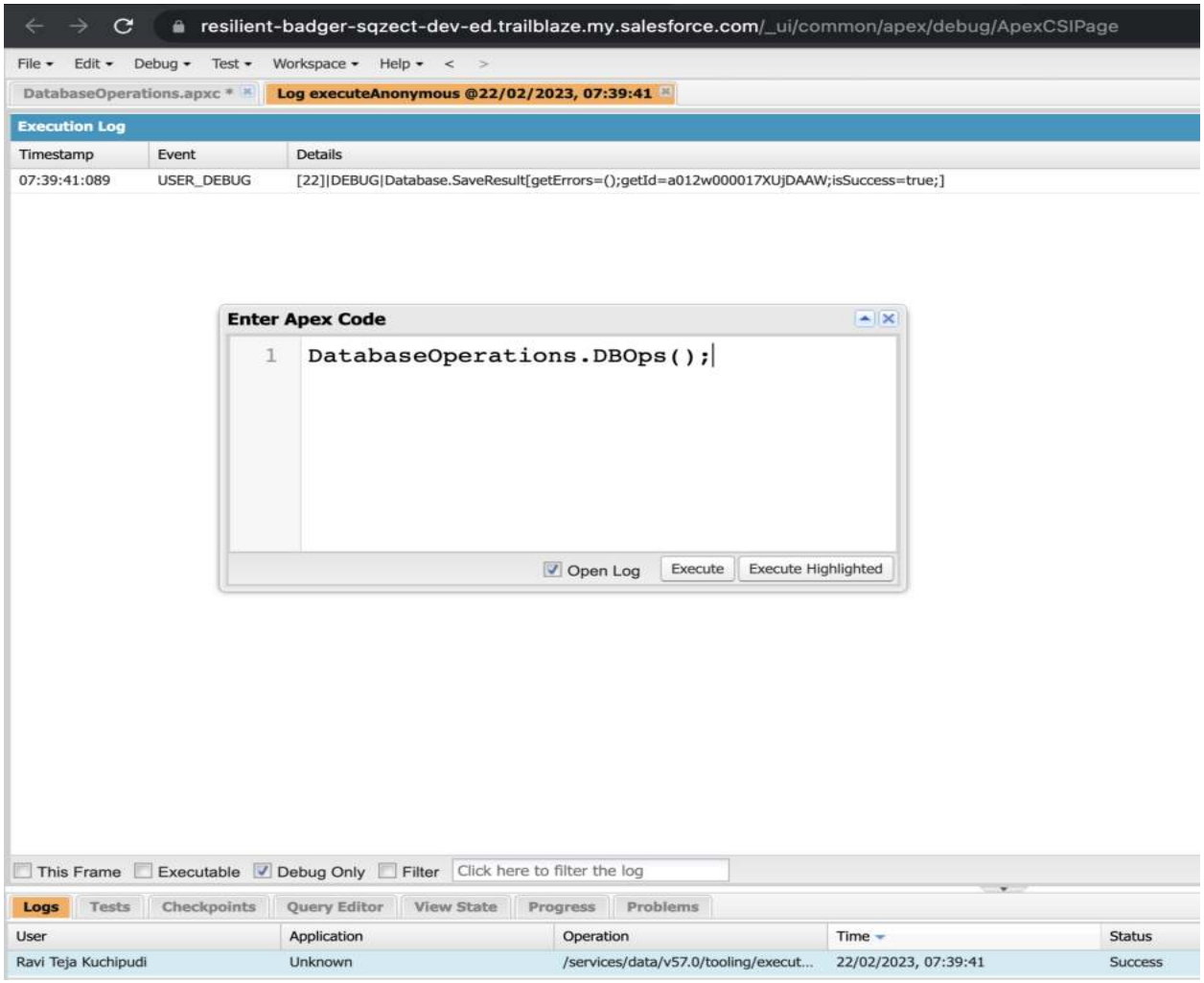

Figure 75

Exercise 11: Write and execute SOQL queries from Developer Console

1. Display ID, Amount, Stage, Account Name, Account Industry, Account Website From Opportunity
2. Add a Where Clause “Account Industry = Energy”
3. AND “Account Annual Revenue > 5000”

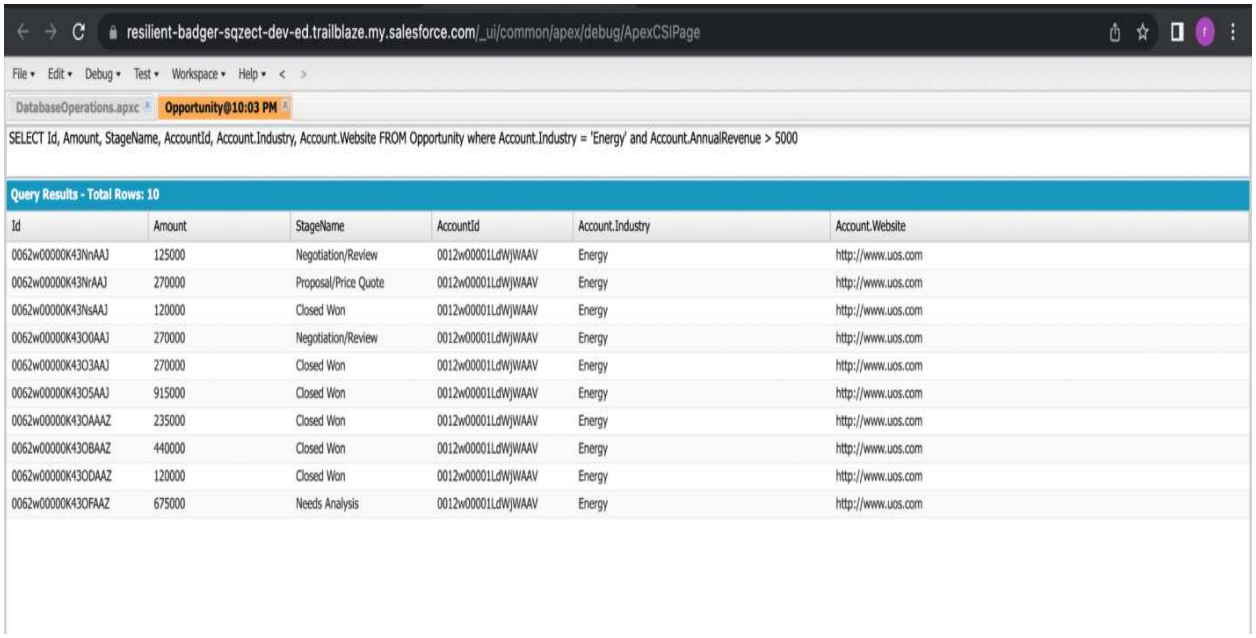

Figure 76

Exercise 12: Write an Apex Trigger, Name = CustomerTrigger.

1. Create a Billing Record when on Customer Object the 'Status' field changes to Active from Inactive.
2. Billing “Status” must be: Paid, Amount Paid = 1000000.
3. Perform the DML on a List.

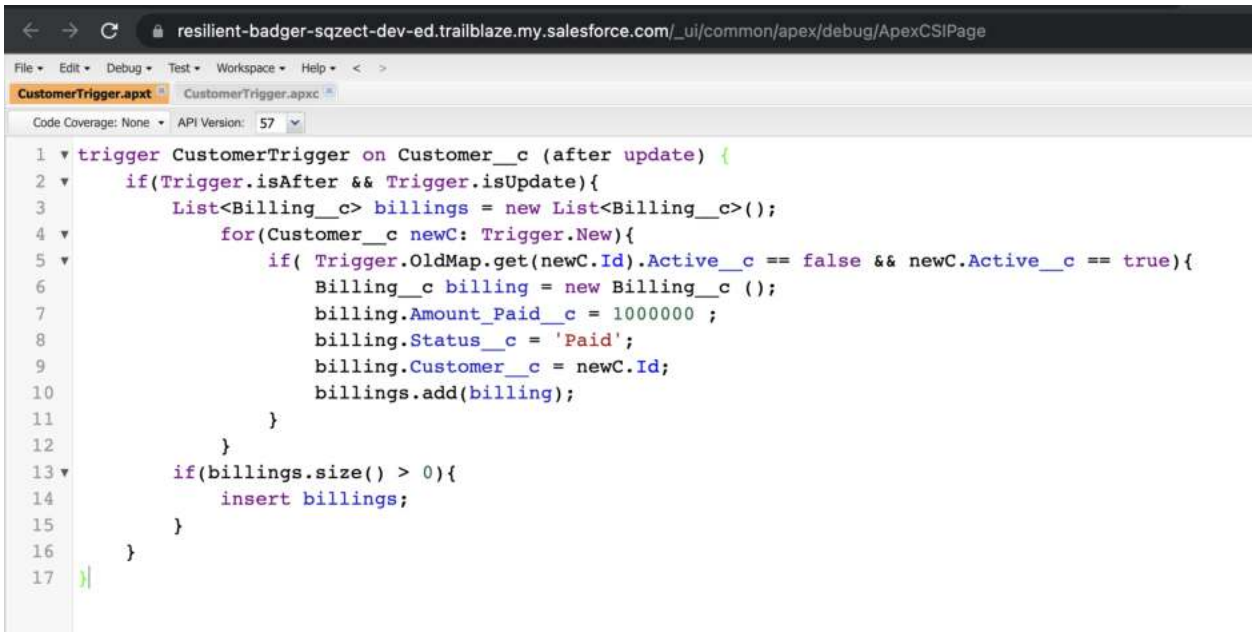

Figure 77

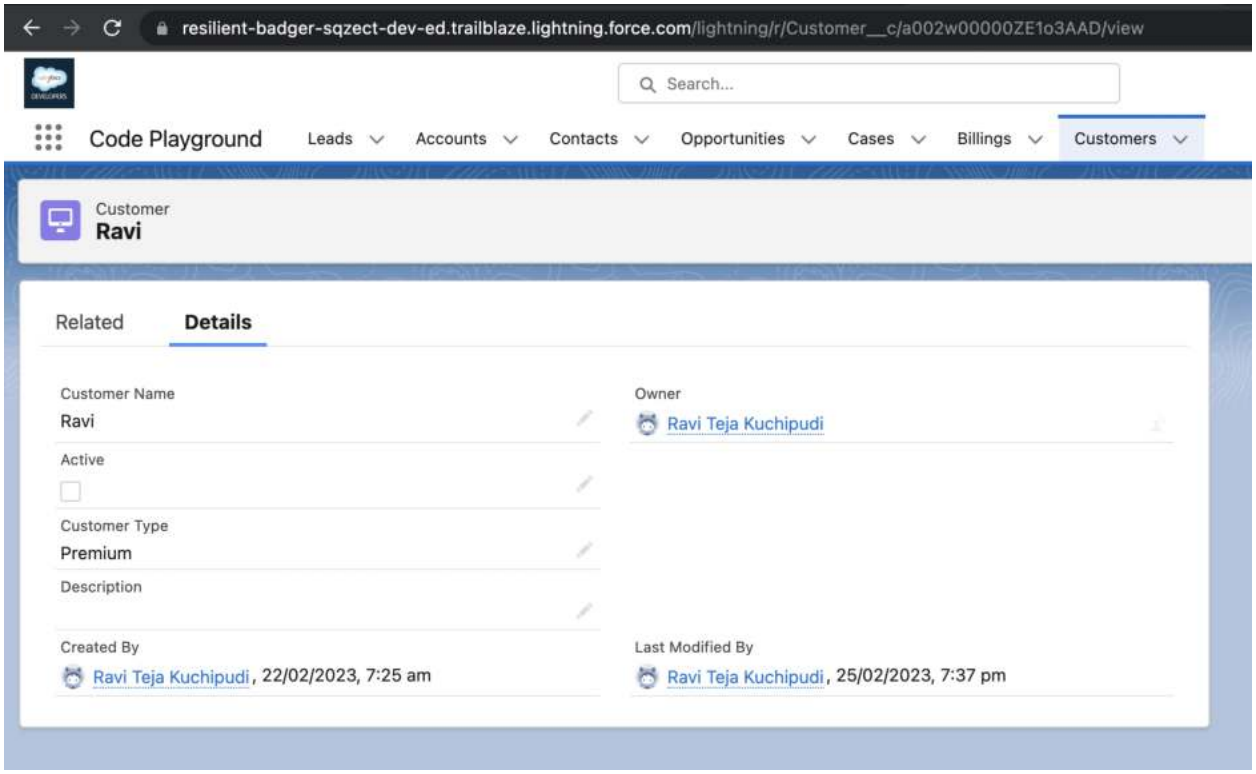

Figure 78

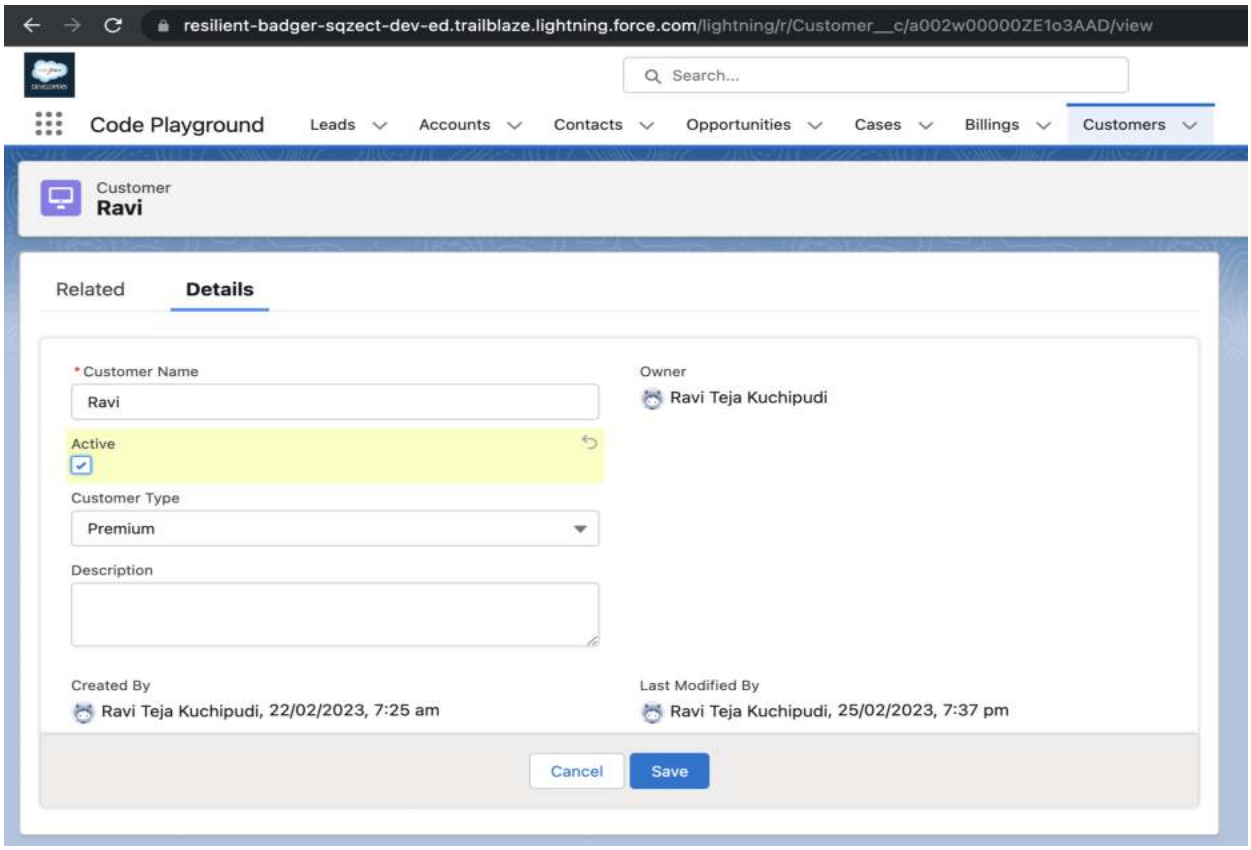

Figure 79

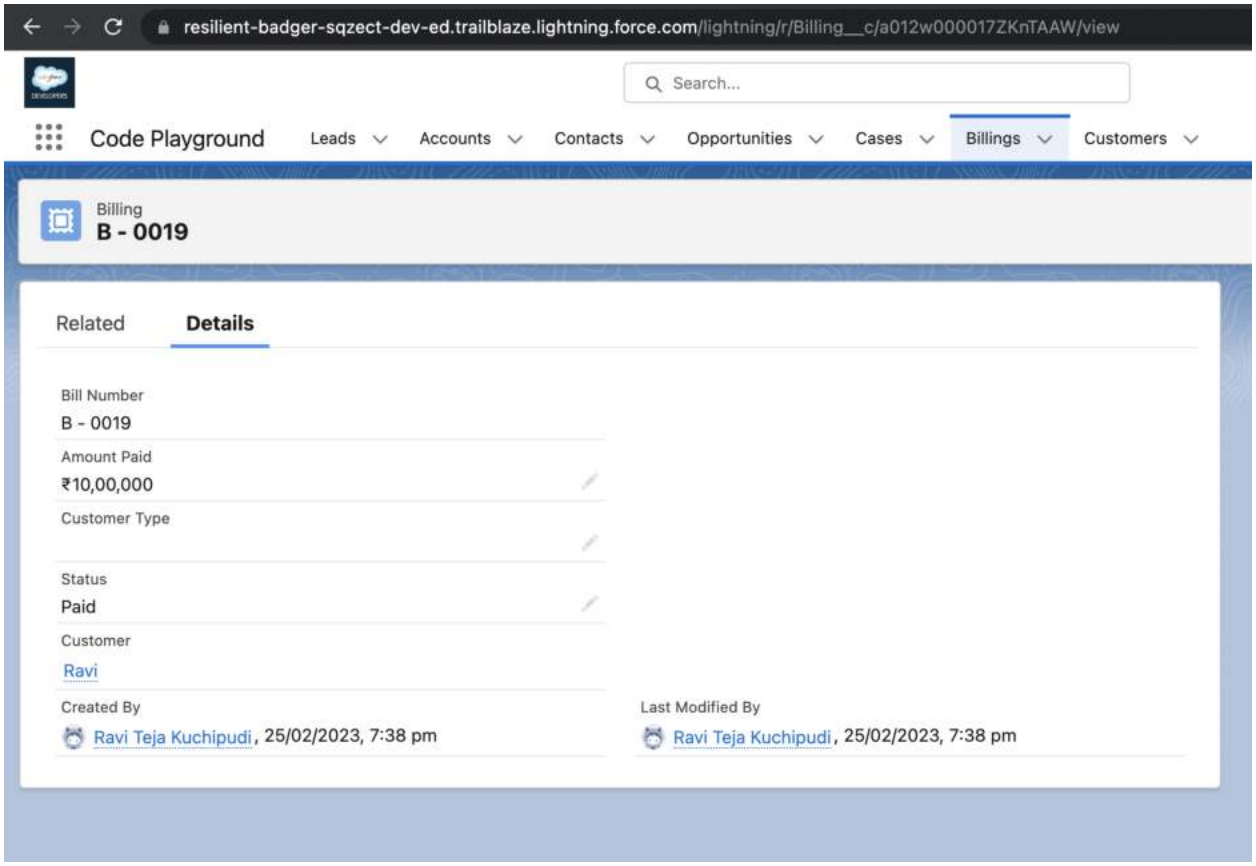

Figure 80

Exercise 13: Write a Test Class for CustomerTrigger.

1. Create the Data for the Customer custom Object
2. Use Test.startTest() & Test.stopTest().
3. Use assert statements to validate the output, Test Class must have full code coverage.

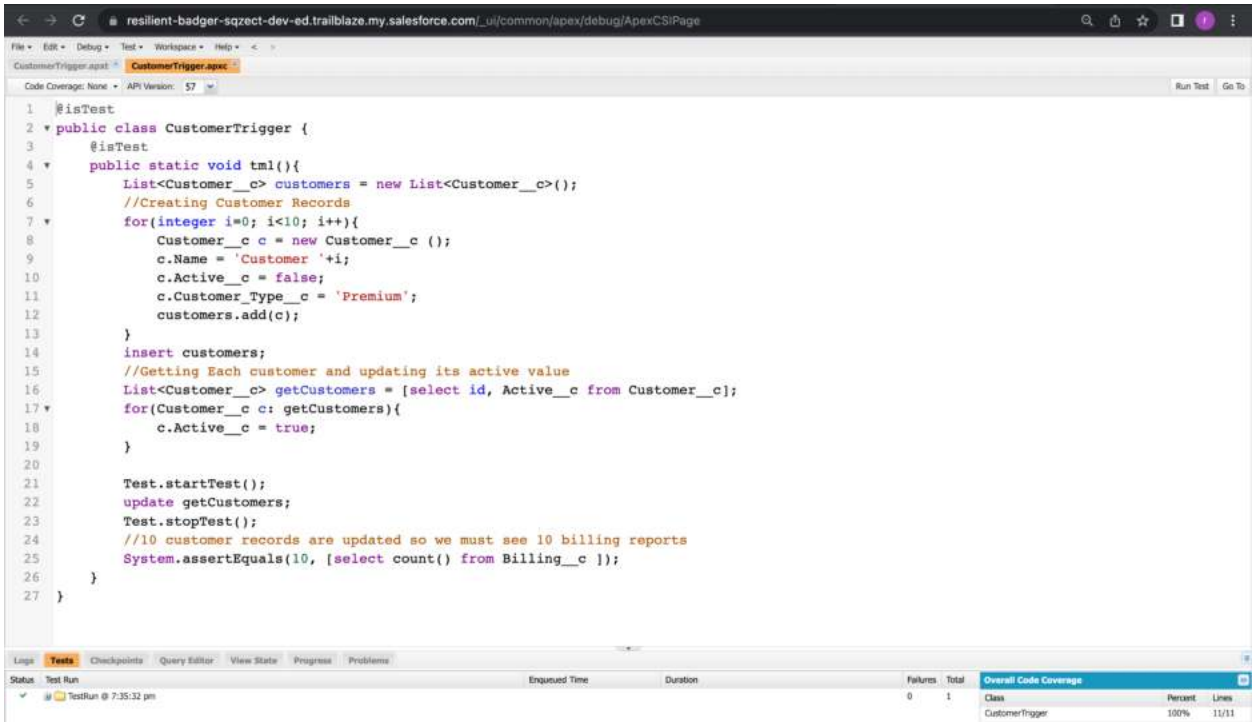

Figure 81

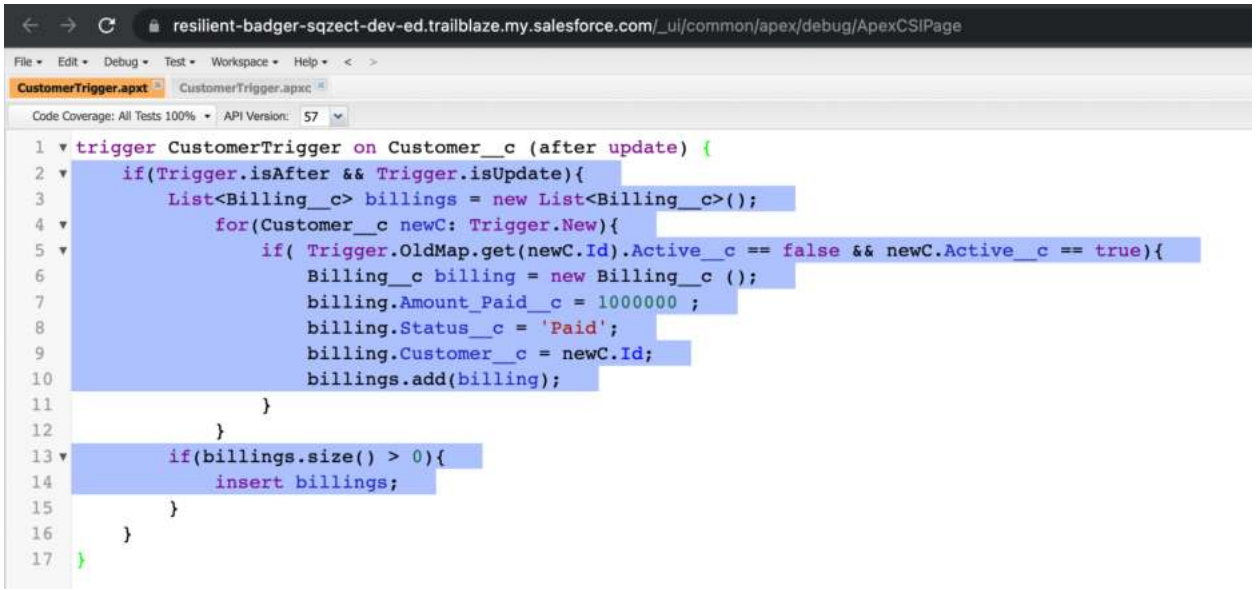

Figure 82

Exercise 14: Write an Apex Trigger, Name = DisqualifyTestLeads.

1. Define logic to find Leads with ‘Test’ in the Name, Ignore Case. Validate that the Lead First/Last Name must not be Blank.
2. IF ‘Test Lead’ found, system.debug(Lead First Name + ‘ ’ + Lead Last Name + ‘Will be disqualified!’);
3. Add the disqualified Leads to a List<Lead>, use for loop to iterate through the list and update Lead status field to ‘Disqualified

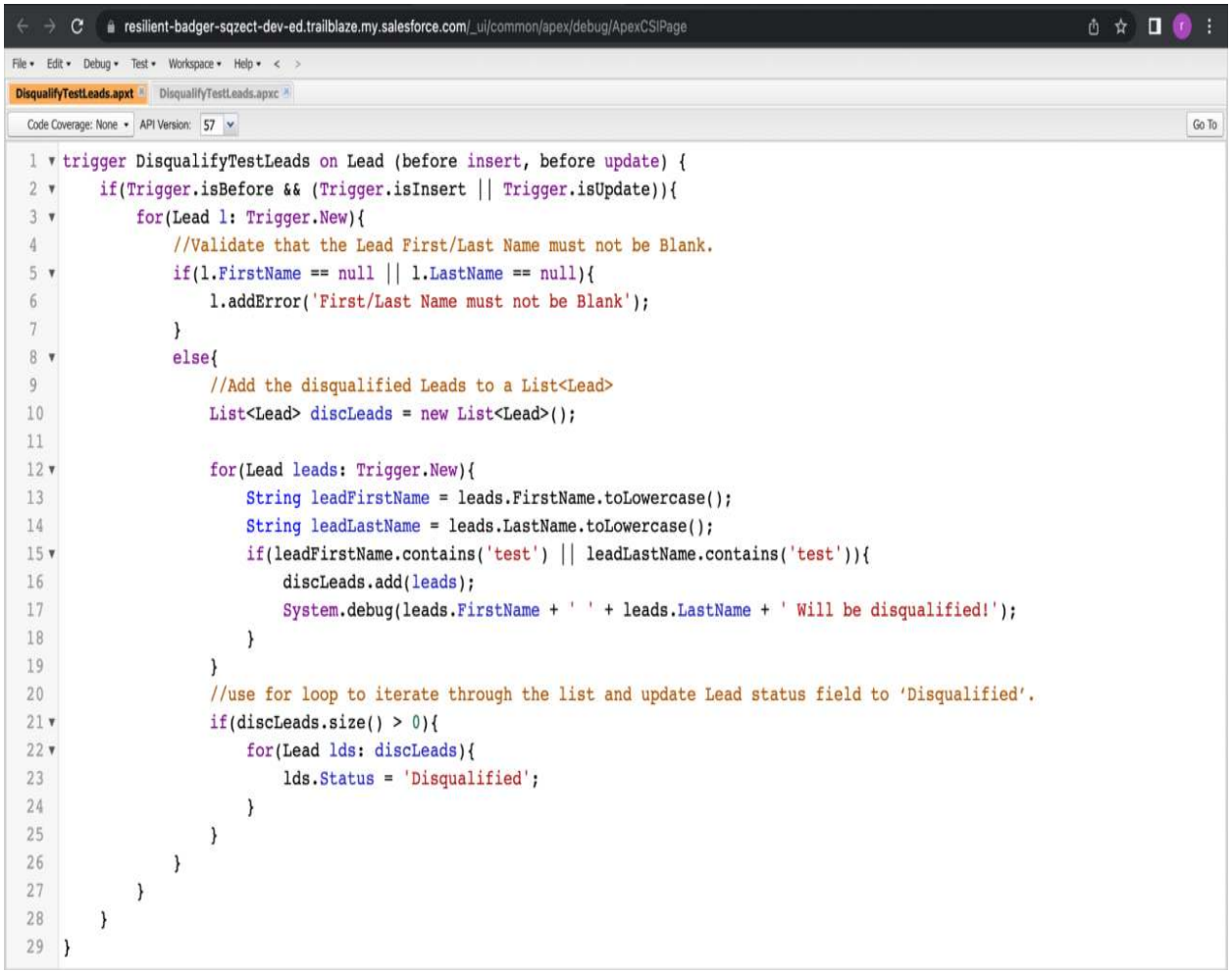

Figure 83

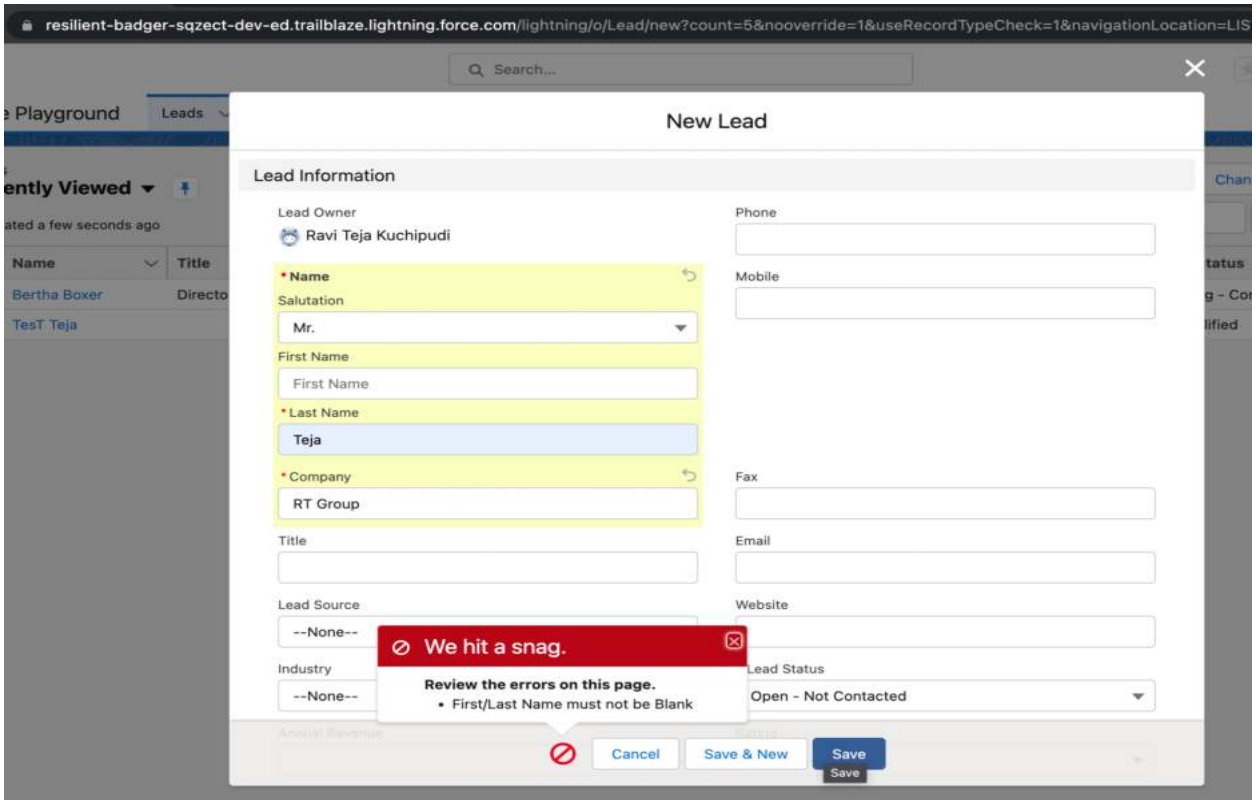

Figure 84

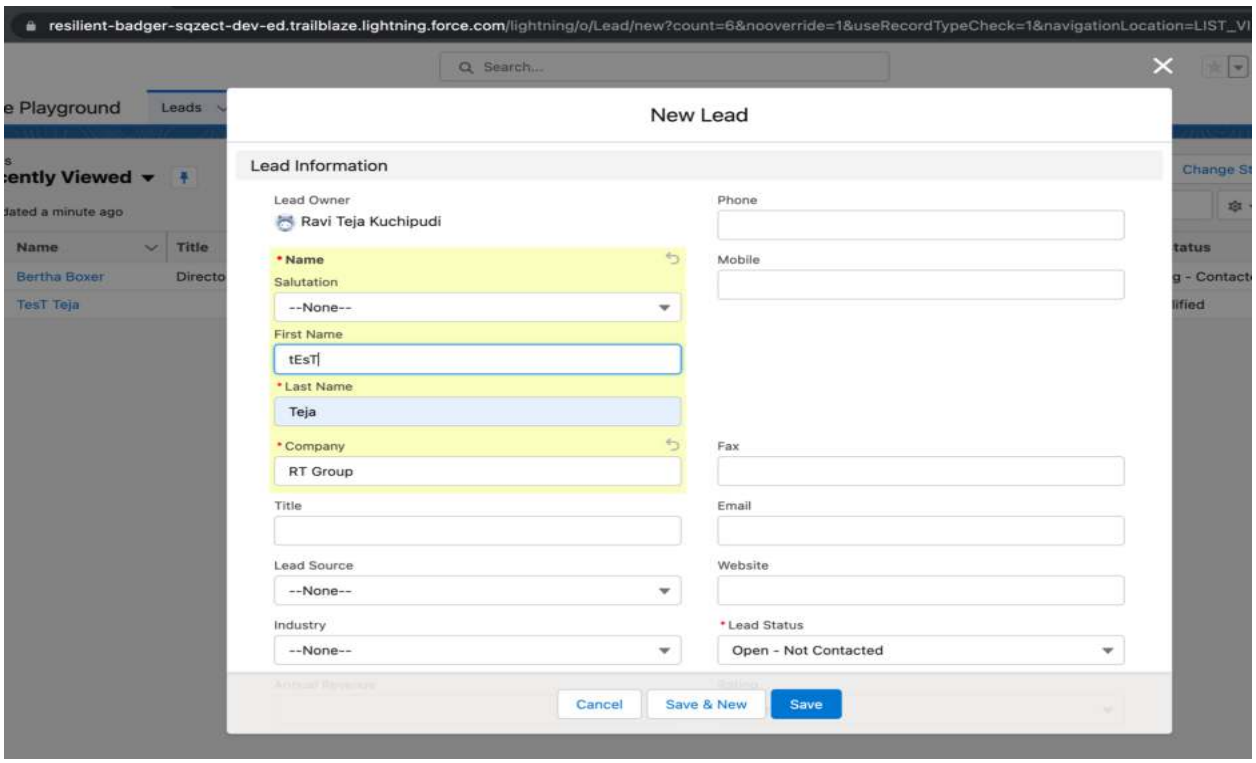

Figure 85

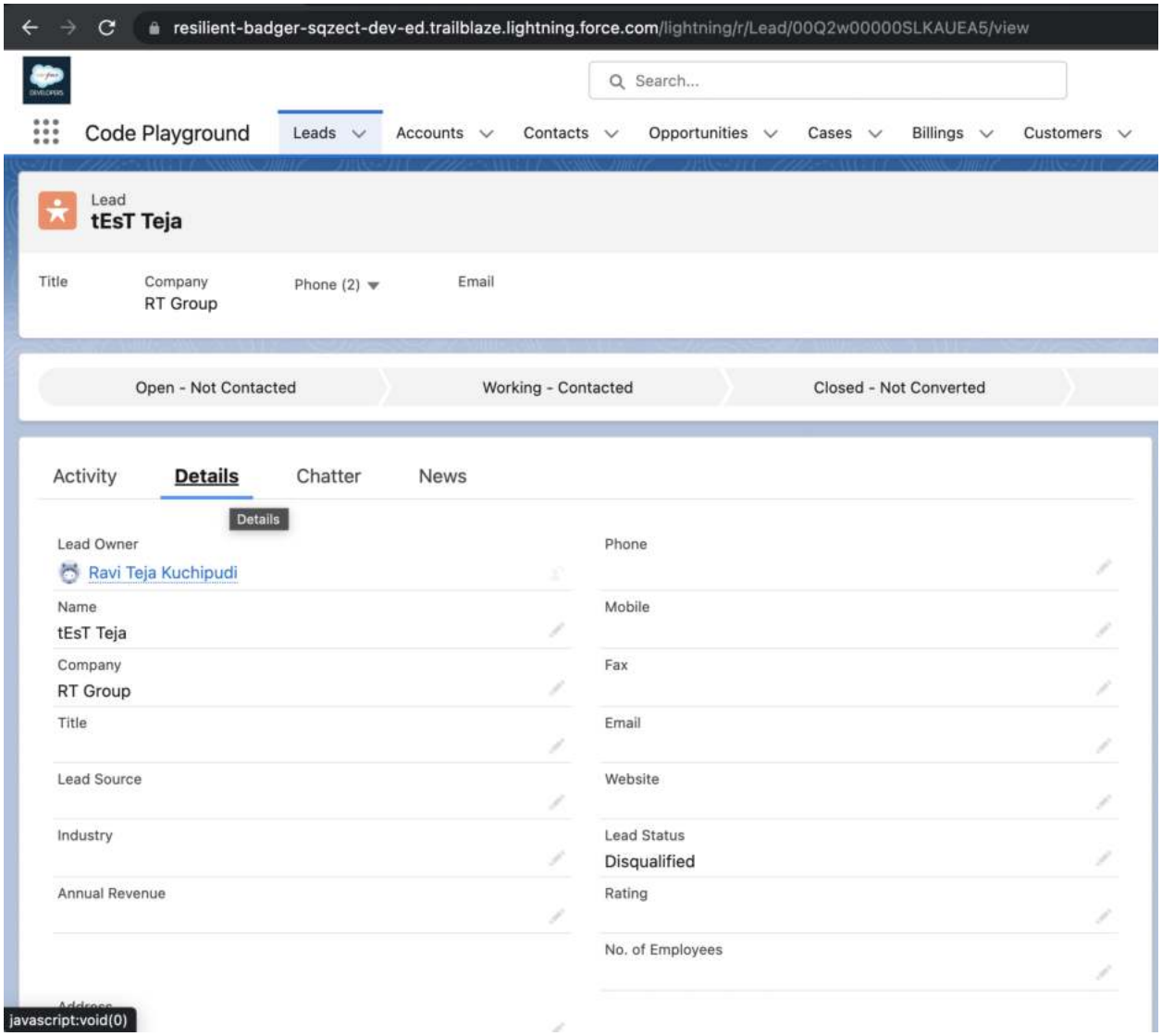

Figure 86

Exercise 15: Write a Test Class for DisqualifyTestLeads.

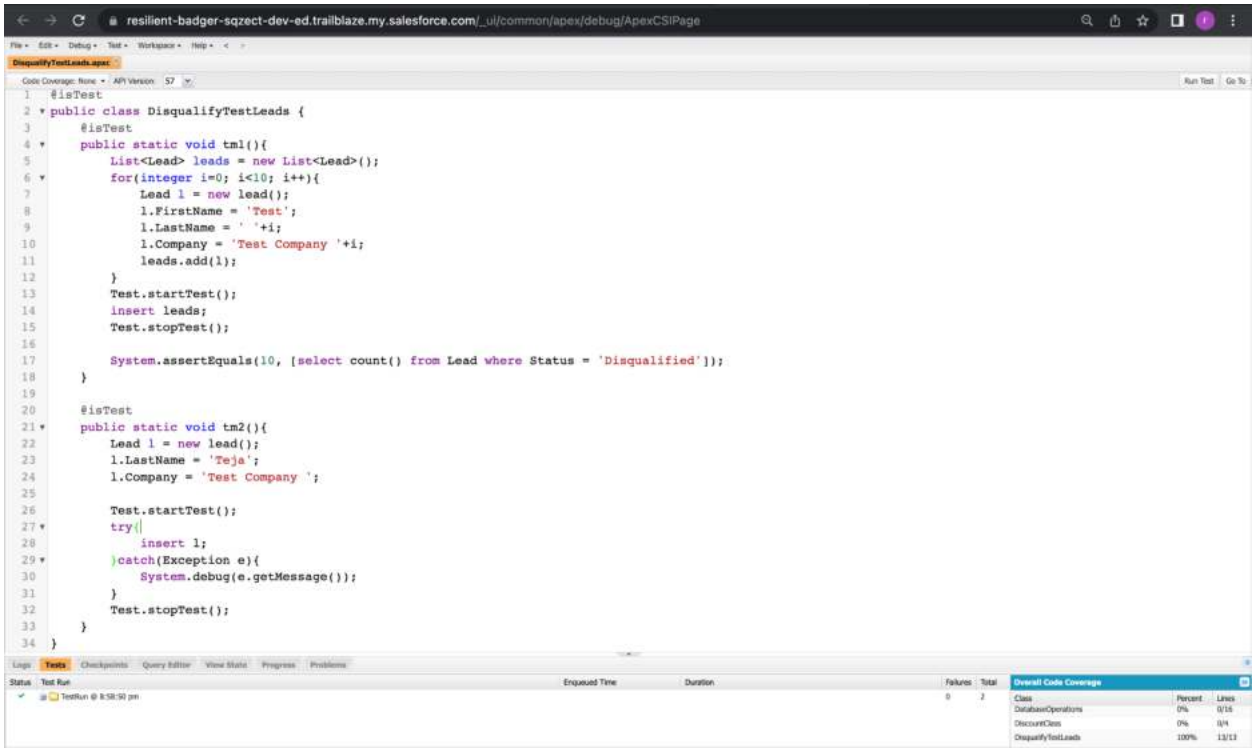

Figure 87

Figure 88

Exercise 16:

1. Create a Visualforce page which displays Opportunity fields as output fields.
2. The page must be named OppView, It must reference the Opportunity standard controller.
3. Define 4 apex:outputField components: Opportunity Name, Amount, Close Date, Opportunity Account Name.

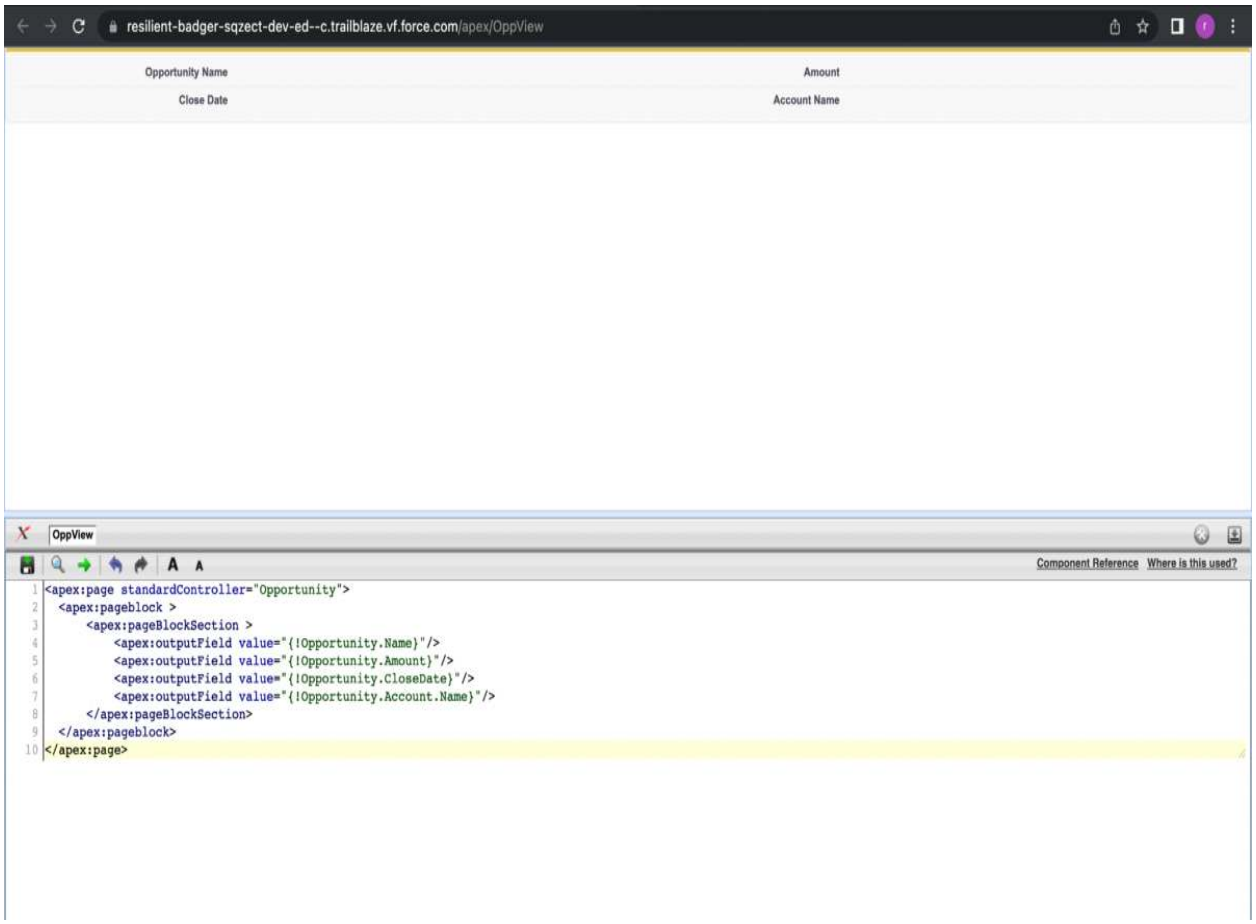

Figure 89

Exercise 17:

1. Create a Visualforce page which shows a list of Accounts linked to their record pages
2. The page must be named AccountList
3. It must reference the Account standard controller.
4. It must have a recordSetVar attribute equal to accounts
5. It must have a Visualforce apex:repeat component, with the following:
6. Use the var attribute set to “a”, Use the <li> HTML list tag
7. Use the apex:outputLink component to link to the respective record detail page

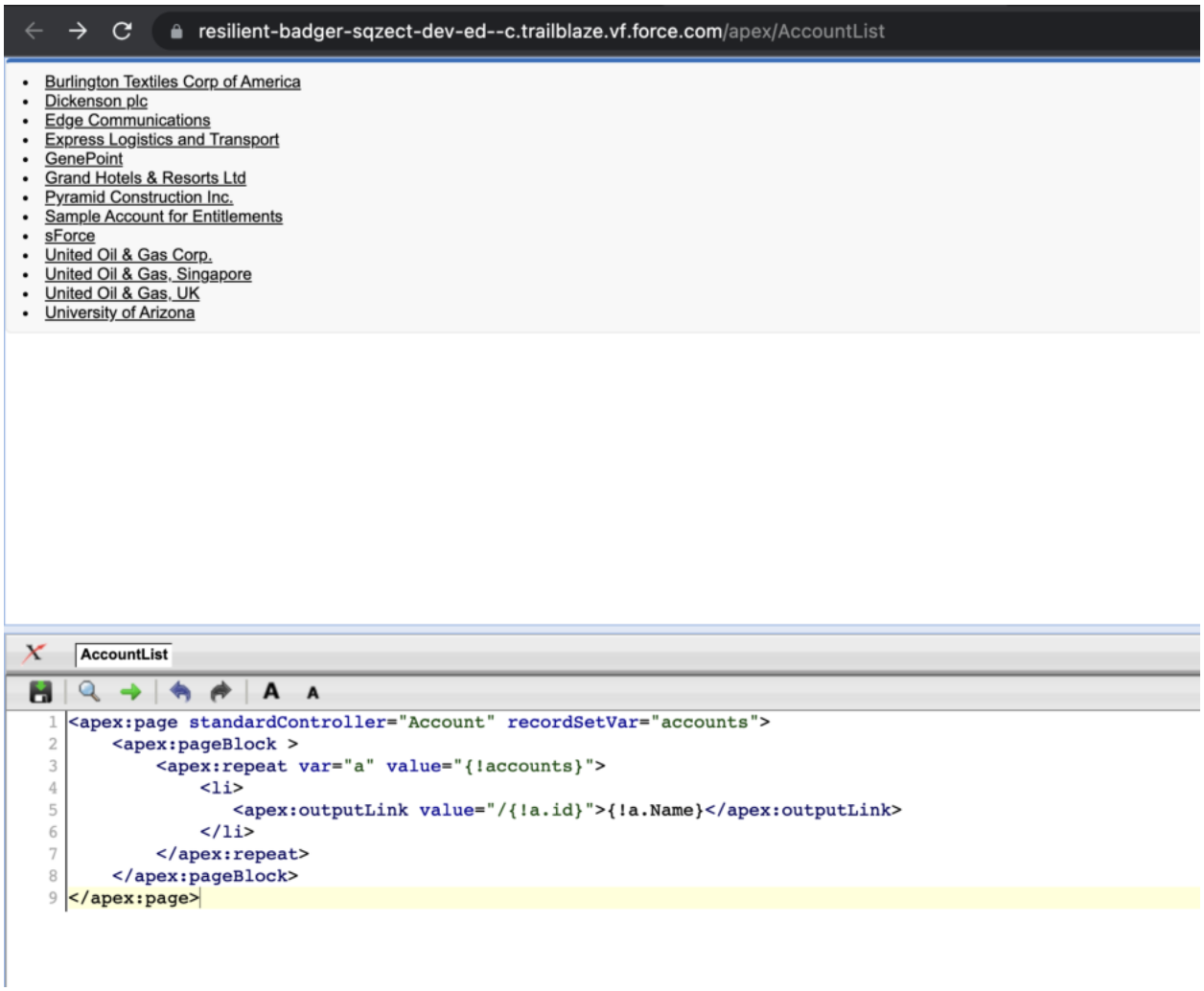

Figure 90

Exercise 18:

1. Create a Visualforce page that uses a custom controller to display a list of cases with the status of 'New'. The page must be named NewCaseList.
2. The custom controller Apex class must be named NewCaseListController and include the following:
3. A publicly scoped method named getNewCases | Use the return type of List<Case>
4. Return a list of case records that includes the ID and CaseNumber fields..
5. Filter the results returned to only have a status of New.
6. The NewCaseList Visualforce page must use an apex:repeat component, which is: Bound to newCases, Refers to the var attribute as case, bind an apex:outputLink component to the ID of the case.\

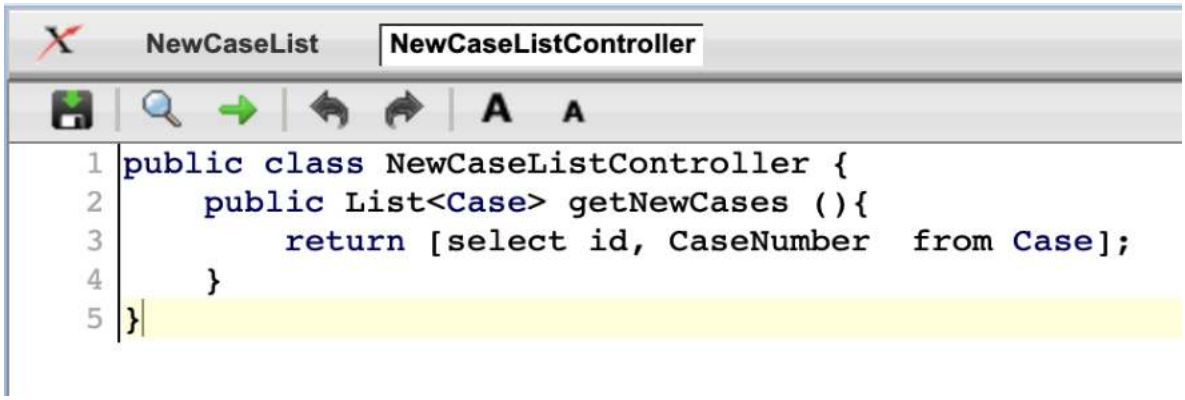

Figure 91

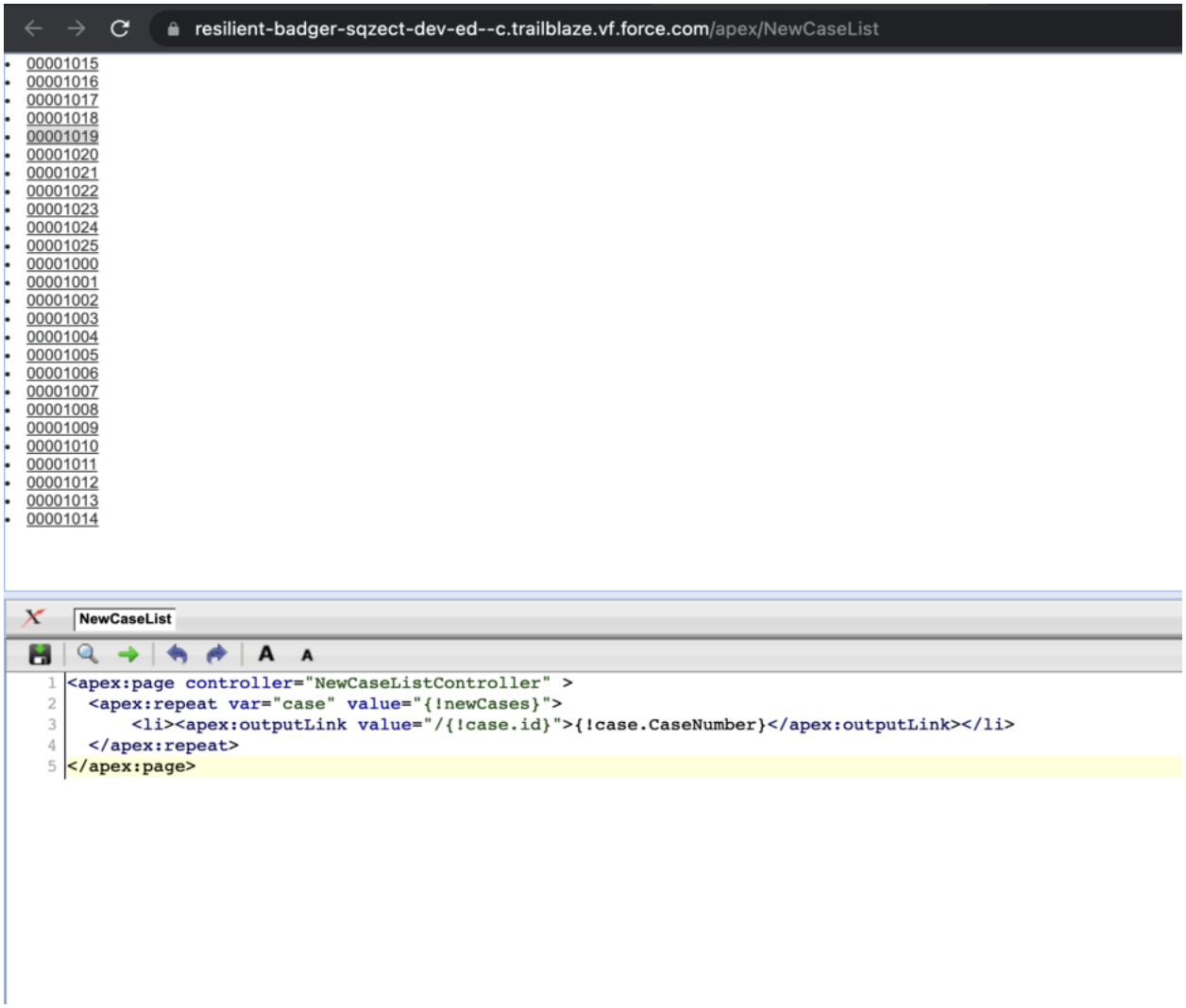
resilient-badger-sqzect-dev-ed--c.trailblaze.vf.force.com/apex/NewCaseList
00001015
00001016
00001017
00001018
00001019
00001020
00001021
00001022
00001023
00001024
00001025
00001000
00001001
00001002
00001003
00001004
00001005
00001006
00001007
00001008
00001009
00001010
00001011
00001012
00001013
00001014
NewCaseList
1 <apex:page controller="NewCaseListController" >
2 <apex:repeat var="case" value="{!newCases}">
3 <apex:outputLink value="/{!case.id}">{!case.CaseNumber}</apex:outputLink>
4 </apex:repeat>
5 </apex:page>

Figure 92

References

1. Manage sales - Salesforce IN
2. Salesforce - ADX201 Administrative Essentials for New Admins in Lightning Experience (SFADX201) (qa.com)
3. https://www.javatpoint.com
4. Understand the Salesforce Architecture Unit | Salesforce Trailhead

Github Link

Github Link - Ravi Teja Kuchipudi