



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

1.	Abstract.....	2
2.	Introduction.....	3
3.	Flow of the project.....	4
4.	Software and Hardware Requirements.....	5
5.	Screen shots.....	6
5.1.	Module 1.....	6
5.2.	Module 2.....	20
5.3.	Module 3.....	34
6.	References.....	62
7.	<u>Github Link</u>	63

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

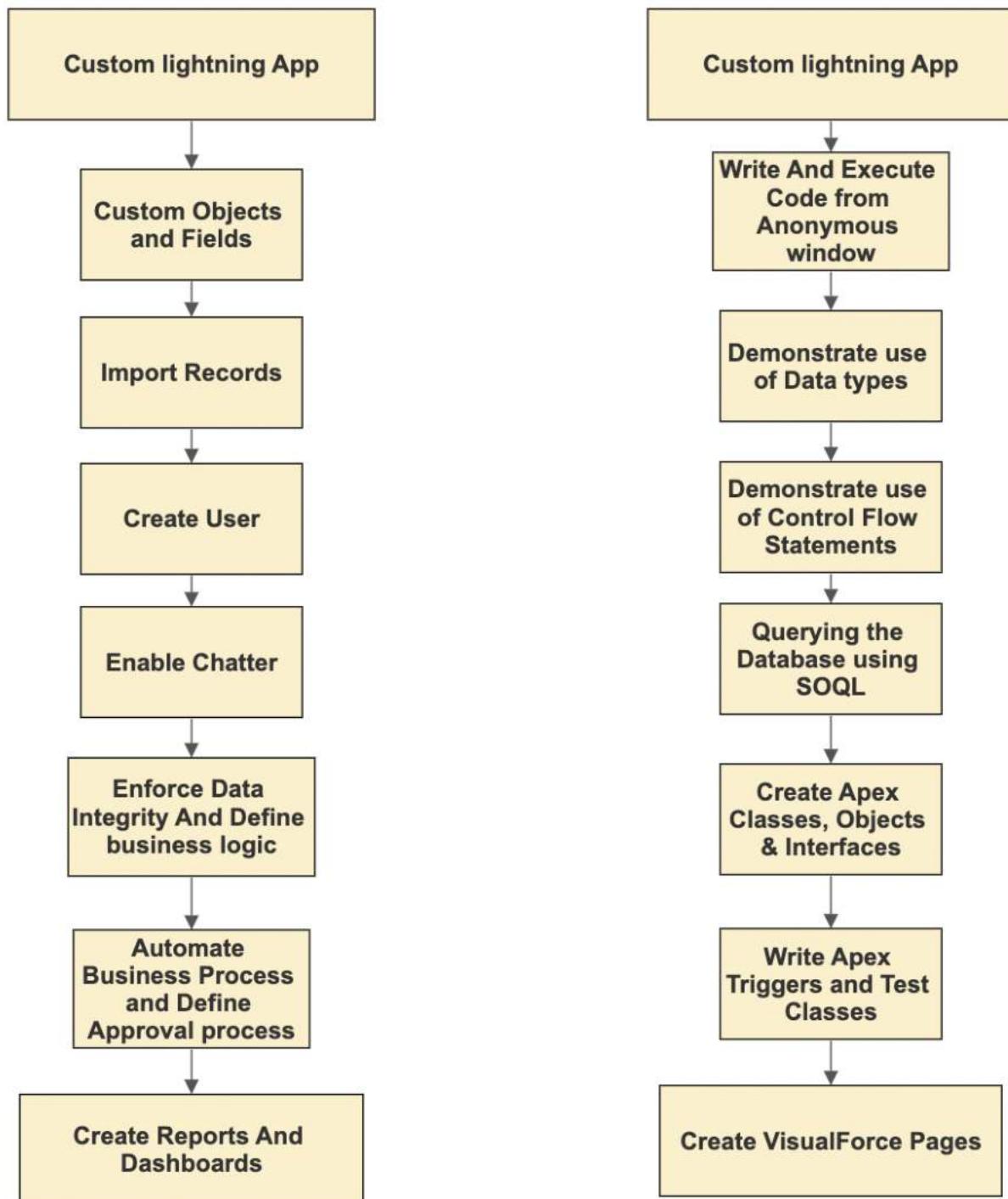


Figure 1: Flowcharts.

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.

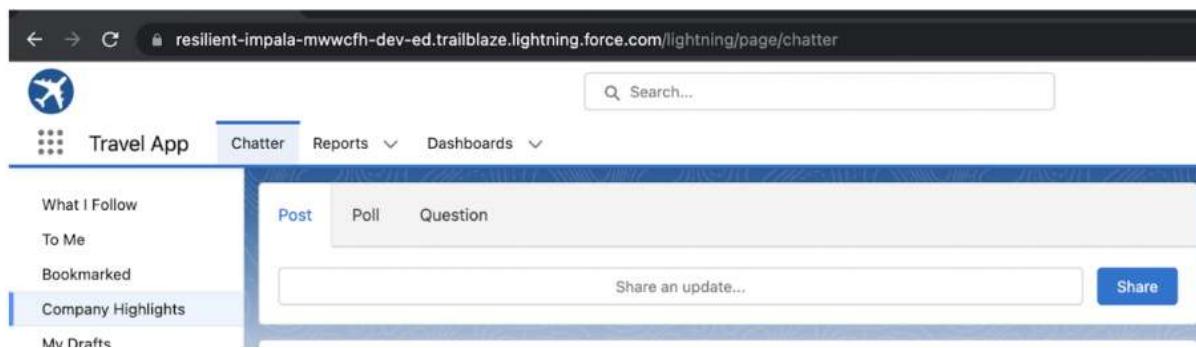


Figure 3

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.

The screenshot shows the Salesforce Object Manager interface. The URL in the browser is resilient-impala-mwwcfh-dev-ed.lightning.force.com/lightning/setup/ObjectManager/01Iw000003WSGs/Details. The page title is "SETUP > OBJECT MANAGER" and the object name is "Department". The left sidebar lists "Fields & Relationships", "Page Layouts", "Lightning Record Pages", "Buttons, Links, and Actions", "Compact Layouts", "Field Sets", and "Object Limits". The main content area has two tabs: "Details" (selected) and "Fields & Relationships". Under "Details", there are sections for "Description" (empty), "API Name" (set to "Department__c"), "Custom" (checkbox checked), "Singular Label" (set to "Department"), "Plural Label" (set to "Departments"), "Enable Reports" (checkbox checked), "Track Activities" (checkbox checked), "Track Field History" (checkbox checked), and "Deployment Status" (set to "Deployed").

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.

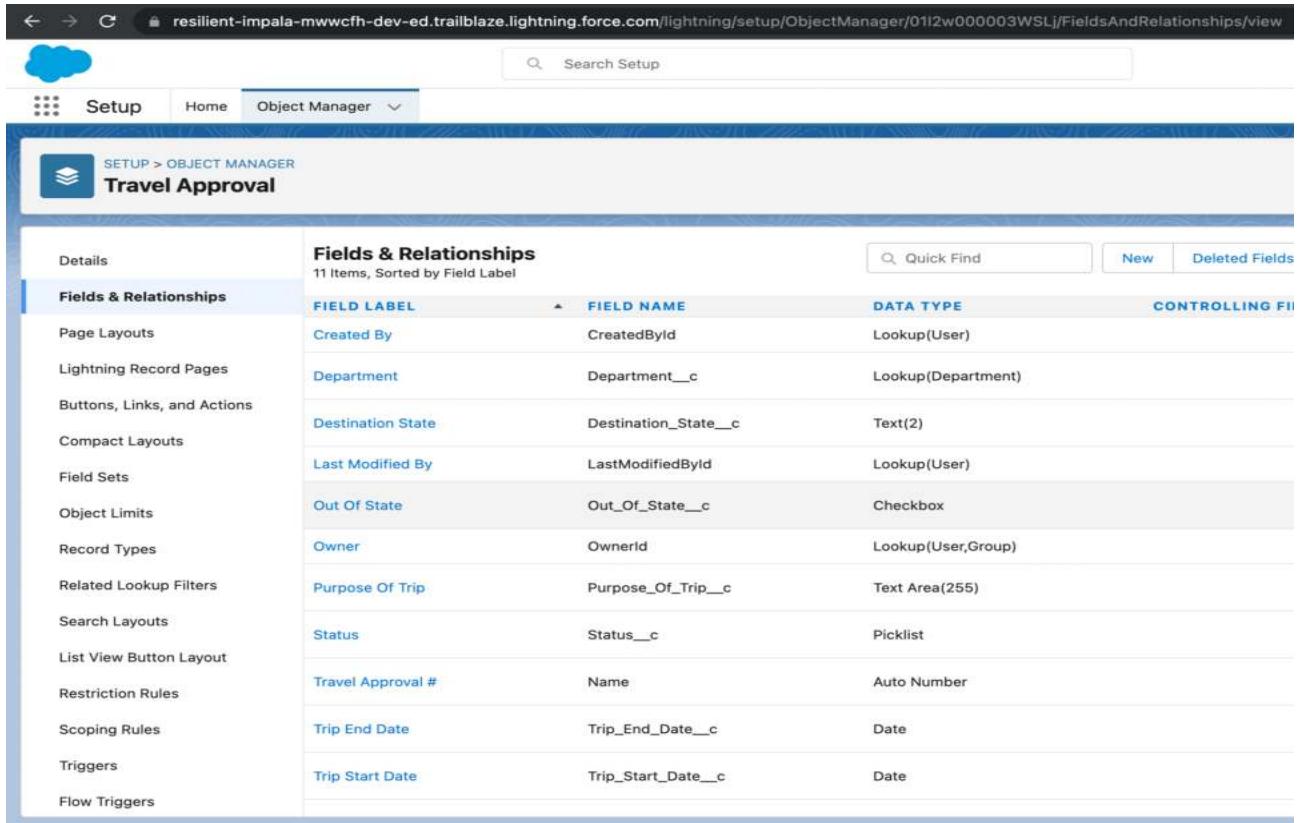
The screenshot shows the Salesforce Object Manager interface for the 'Department' object. The top navigation bar includes 'Setup', 'Home', and 'Object Manager'. The main title is 'Department'. The left sidebar lists various setup categories: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, and Search Layouts. The 'Fields & Relationships' section displays 7 items, sorted by Field Label. A table lists the fields:

	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING
Page Layouts	Created By	CreatedBy	Lookup(User)	
Lightning Record Pages	Department Code	Department_Code__c	Text(10) (Unique Case Sensitive)	
Buttons, Links, and Actions	Department Name	Name	Text(80)	
Compact Layouts	Department Type	Department_Type__c	Picklist	Location
Field Sets	Last Modified By	LastModifiedBy	Lookup(User)	
Object Limits	Location	Location__c	Picklist	
Record Types	Owner	OwnerId	Lookup(User,Group)	

Figure 5

Step 4:

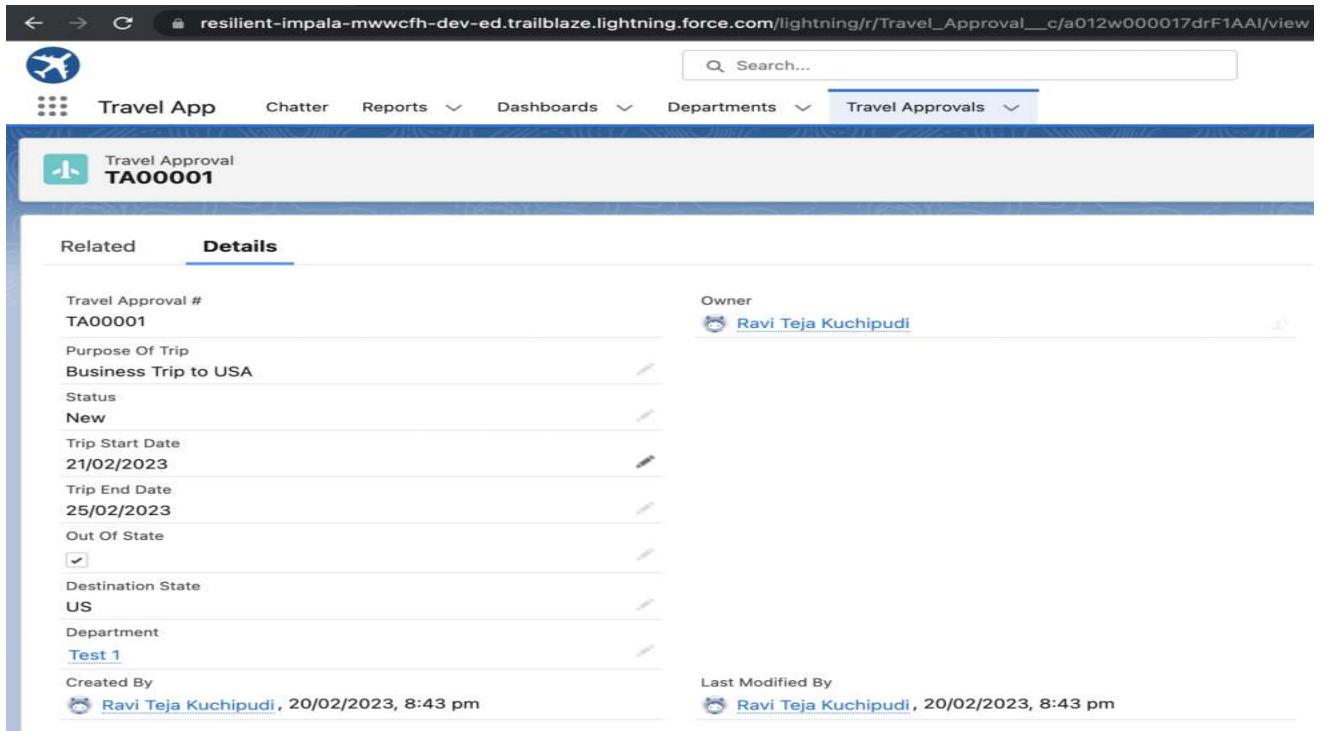
1. Create a Travel Approval Object.
2. Label = “Travel Approval”, Starting Number = 1., Plural = “Travel Approvals”, Object Name = “Travel_Approval”
3. Record Name = “Travel Approval #”, Data Type = “Auto Number”, Display format “TA{00000}”.
4. Select additional options(Allow Reports, Allow Activities, 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.



The screenshot shows the Salesforce Object Manager interface for the 'Travel Approval' object. The left sidebar lists various setup categories like Page Layouts, Lightning Record Pages, Buttons, etc. The main area displays the 'Fields & Relationships' section with 11 items. A table lists the field label, name, data type, and controlling field for each.

Fields & Relationships		11 Items, Sorted by Field Label	Quick Find	New	Deleted Fields
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIEL	
Page Layouts	Created By	CreatedById	Lookup(User)		
Lightning Record Pages	Department	Department__c	Lookup(Department)		
Buttons, Links, and Actions	Destination State	Destination_State__c	Text(2)		
Compact Layouts	Last Modified By	LastModifiedById	Lookup(User)		
Field Sets	Out Of State	Out_Of_State__c	Checkbox		
Object Limits	Owner	OwnerId	Lookup(User,Group)		
Record Types	Purpose Of Trip	Purpose_of_Trip__c	Text Area(255)		
Related Lookup Filters	Status	Status__c	Picklist		
Search Layouts	Travel Approval #	Name	Auto Number		
List View Button Layout	Trip End Date	Trip_End_Date__c	Date		
Restriction Rules	Trip Start Date	Trip_Start_Date__c	Date		
Scoping Rules					
Triggers					
Flow Triggers					

Figure 6



The screenshot shows the Travel App detail page for a travel approval record with ID TA00001. The page includes sections for Related and Details. The Details section contains fields for Travel Approval # (TA00001), Purpose Of Trip (Business Trip to USA), Status (New), Trip Start Date (21/02/2023), Trip End Date (25/02/2023), Out Of State (checked), Destination State (US), Department (Test 1), Created By (Ravi Teja Kuchipudi), and Last Modified By (Ravi Teja Kuchipudi).

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.

The screenshot shows the Salesforce Lightning interface for the 'Travel App'. The top navigation bar includes links for Chatter, Reports, Dashboards, Departments (which is currently selected), and Travel Approvals. A search bar is also present. The main content area displays a list of 'Departments' with 16 items. The list is sorted by 'Department Name' in ascending order. Each item is represented by a checkbox followed by the department name. The departments listed are: Audit Services, Contract Management, Disability Determination Bureau, Division of Aging, Division of Disability and Rehabilitative Services, Division of Family Resources, Division of Finance, Division of Mental Health and Addiction, Human Resources, Legislative Services, Office of Communications and Media, Office of Early Childhood and Out-of-School Learning, Office of General Counsel, Office of Medicaid Policy and Planning, Quality and Compliance Office, and Technology.

Rank	Department Name
1	Audit Services
2	Contract Management
3	Disability Determination Bureau
4	Division of Aging
5	Division of Disability and Rehabilitative Services
6	Division of Family Resources
7	Division of Finance
8	Division of Mental Health and Addiction
9	Human Resources
10	Legislative Services
11	Office of Communications and Media
12	Office of Early Childhood and Out-of-School Learning
13	Office of General Counsel
14	Office of Medicaid Policy and Planning
15	Quality and Compliance Office
16	Technology

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.

The screenshot shows a Salesforce Lightning interface for a Travel Approval record. The URL in the browser is resilient-impala-mwwcfh-dev-ed.lightning.force.com/lightning/r/Travel_Approval__c/a012w000017drF2AAI/view. The top navigation bar includes links for Travel App, Chatter, Reports, Dashboards, Departments, and Travel Approvals. The Travel Approvals link is currently selected. The main content area displays a Travel Approval record with the ID TA00002. The record has the following fields:

- Travel Approval #: TA00002
- Purpose Of Trip: Attend Dreamforce
- Status: Draft
- Trip Start Date: 20/02/2023
- Trip End Date: 24/02/2023
- Out Of State:
- Destination State: CA
- Department: Technology
- Created By: Ravi Teja Kuchipudi, 20/02/2023, 8:53 pm
- Last Modified By: Ravi Teja Kuchipudi, 21/02/2023, 9:44 am

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.

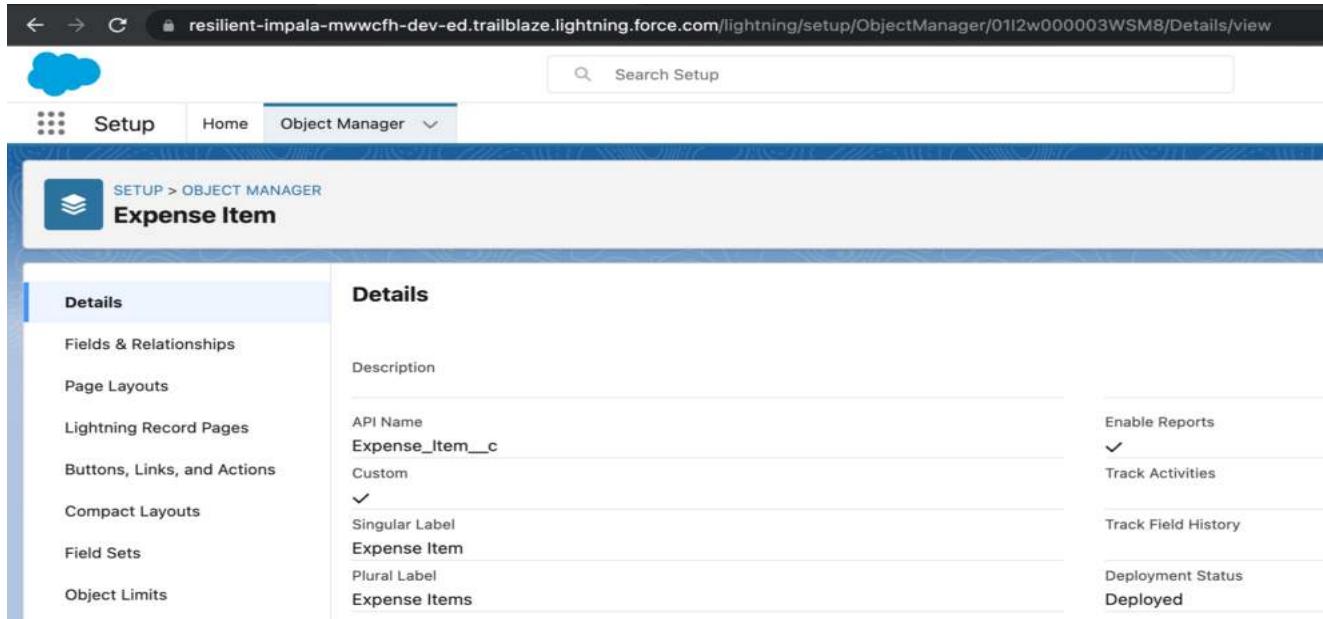


Figure 10

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.

Details		Fields & Relationships			
		6 Items, Sorted by Field Label			
		FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLIN
Page Layouts	Amount	Amount	Amount__c	Currency(16, 2)	
Lightning Record Pages	Created By	Created By	CreatedById	Lookup(User)	
Buttons, Links, and Actions	Expense Item Number	Expense Item Number	Name	Auto Number	
Compact Layouts	Expense Type	Expense Type	Expense_Type__c	Picklist	
Field Sets	Last Modified By	Last Modified By	LastModifiedById	Lookup(User)	
Object Limits	Travel Approval	Travel Approval	Travel_Approval__c	Master-Detail(Travel Approval)	
Record Types					

Figure 11

Step 4:

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

The screenshot shows a Salesforce Lightning page for a Travel Approval record with ID TA00002. The top navigation bar includes links for Travel App, Chatter, Reports, Dashboards, Departments, and Travel Approvals. The main content area displays the Travel Approval details, including a Notes & Attachments section with an 'Upload Files' button, and an Expense Items section showing two items with expense item numbers E - 00001 and E - 00002. A 'View All' link is also present in the Expense Items section.

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.

Eric Executive

User Detail	
Name	Eric Executive
Alias	eexec
Email	ravi.kuchipudi1@wipro.com
Username	ravi.executive@wipro.com
Nickname	User16769520817898622875
Role	CEO
User License	Salesforce
Profile	System Administrator
Active	<input checked="" type="checkbox"/>
Marketing User	<input type="checkbox"/>

Figure 13

Step 6: Add user Eric Executive as your manager

Mailing Address

Street	<input type="text"/>
City	<input type="text"/>
Zip/Postal Code	<input type="text"/>
State/Province	<input type="text"/>
Country	IN

Single Sign On Information

Federation ID	<input type="text"/>
---------------	----------------------

Locale Settings

Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)
Locale	English (India)
Language	English

Approver Settings

Delegated Approver	<input type="text"/>
Manager	Eric Executive
Receive Approval Request Emails	Only if I am an approver

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.

Travel Approvals
Open Out of State Travel Requests

Travel Approval #	Department	Created By	Status	Destinat...	Trip Start Date	Trip End Date
TA00002	Technology	Ravi Teja Kuchipudi	Draft	CA	20/02/2023	24/02/2023

New Import Change Owner Printable View

Search this list...

Filters

Filter by Owner
All travel approvals

Matching all of these filters

Out Of State
equals True

Status
not equal to Approved, Rejected

Add Filter Remove All

Add Filter Logic

Figure 17

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

Travel Approvals
Open Out of State Travel Requests

0 Items - Sorted by Travel Approval # - Filtered by All travel approvals - Status, Out Of State - Updated 2 minutes ago

Select Fields to Display

Available Fields

Visible Fields

Cancel Save

Figure 18

Step 11:

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

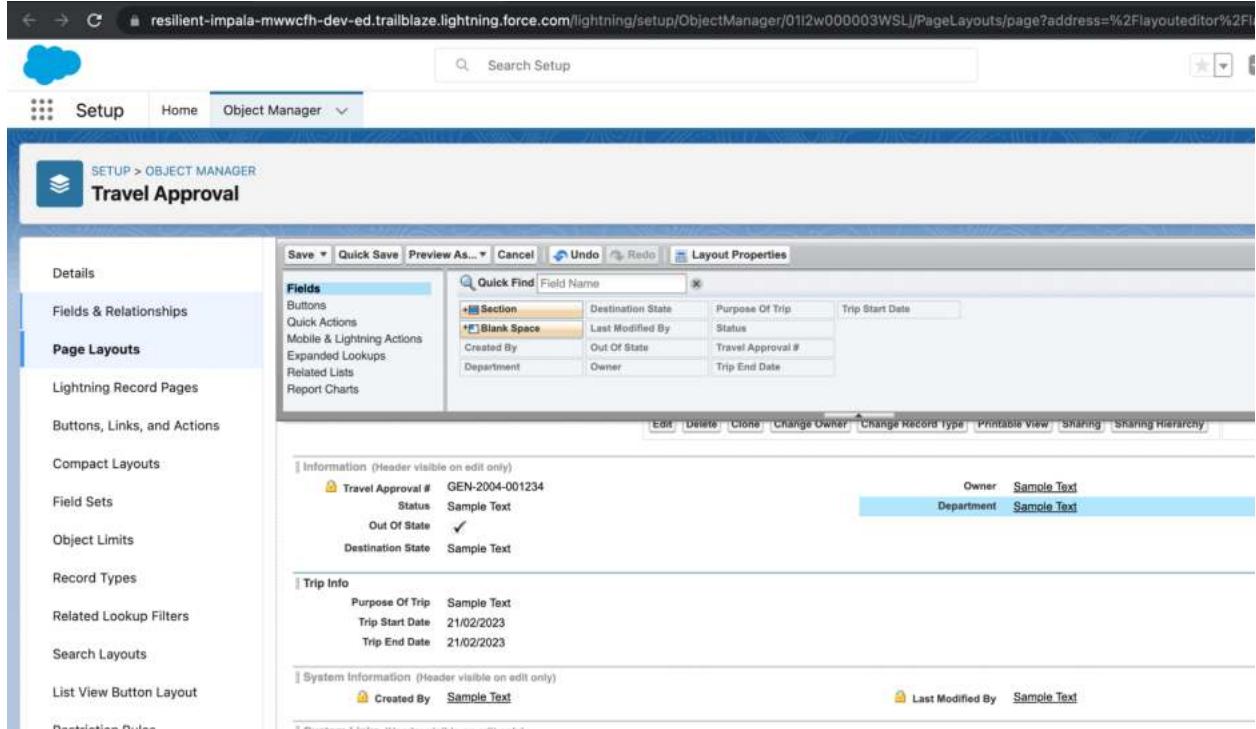


Figure 19

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

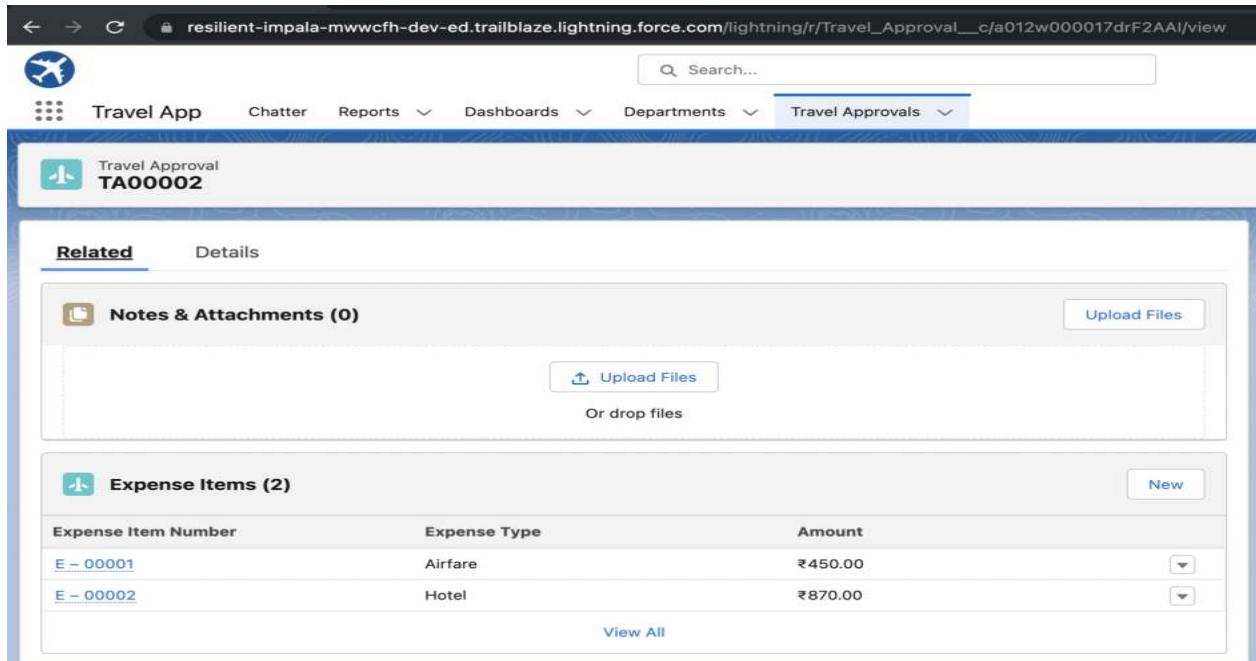


Figure 20

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.

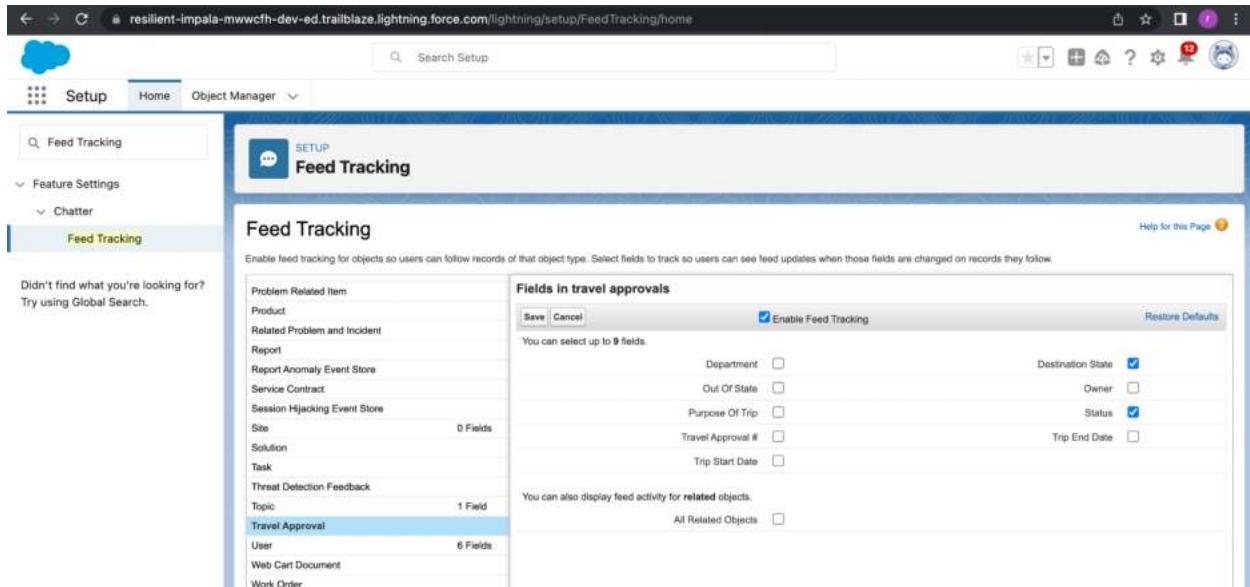


Figure 21

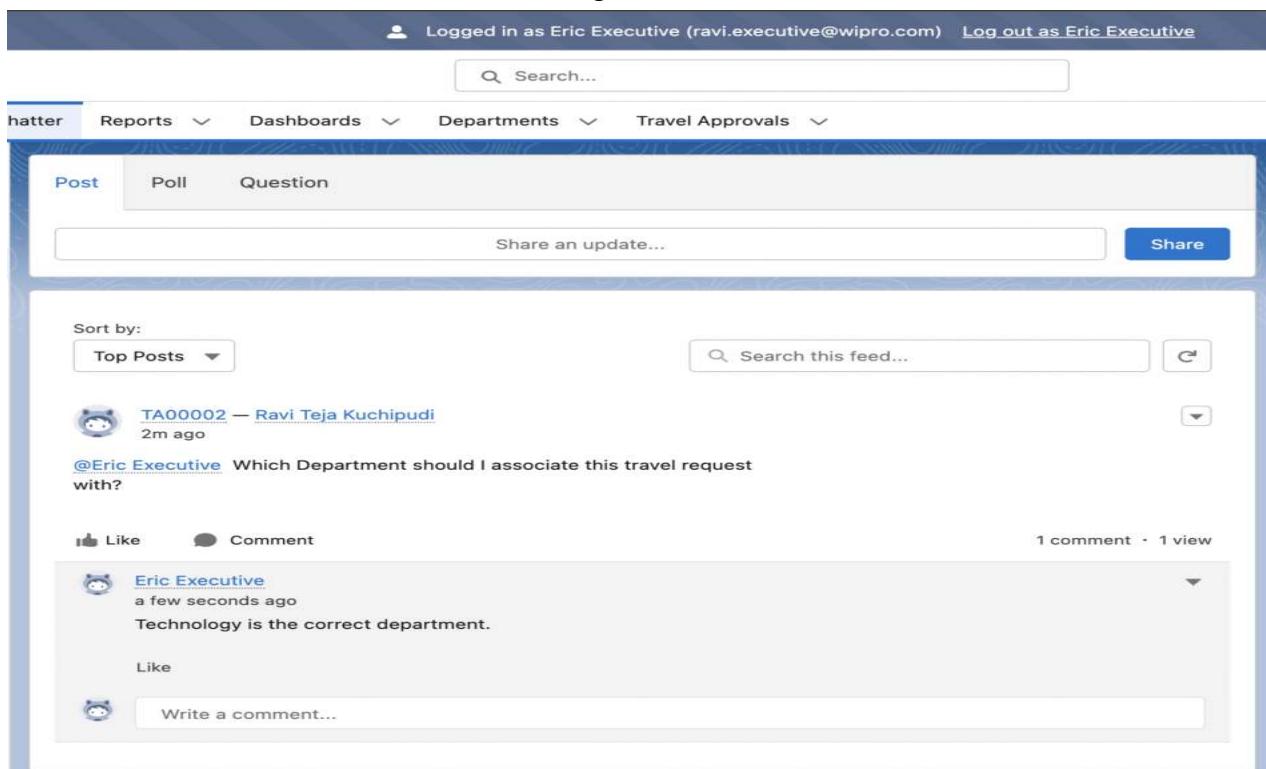


Figure 22

Test The App:

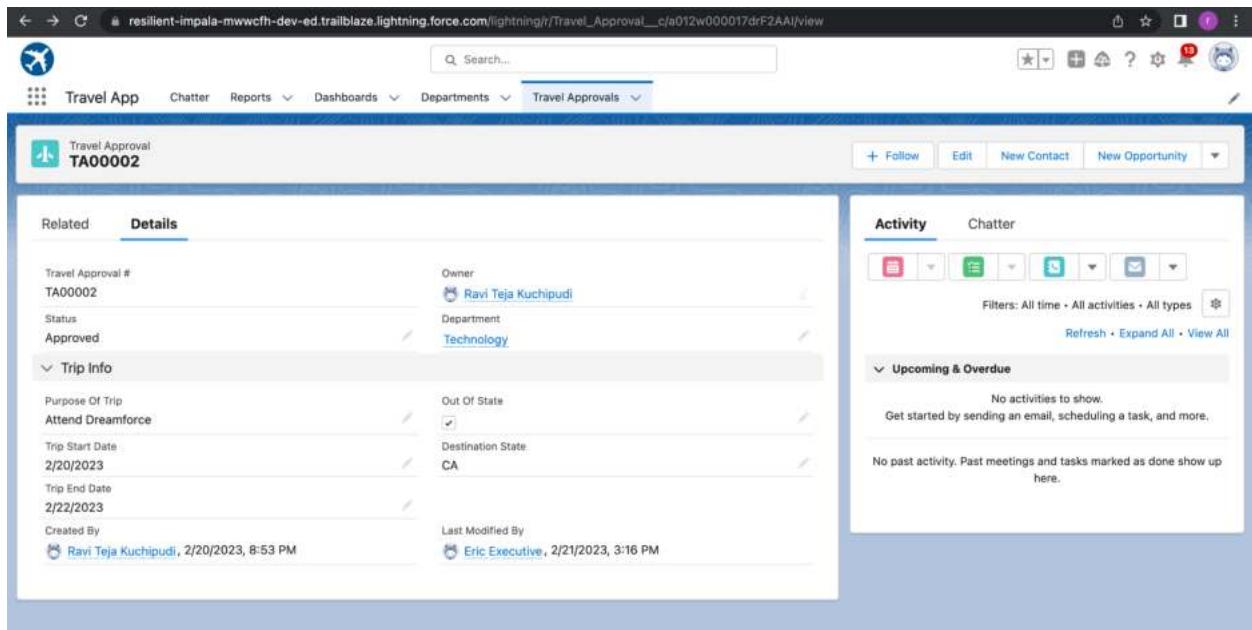


Figure 23

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 (\geq) 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.

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes links for Setup, Home, and Object Manager. The main title is "Travel Approval". On the left, there's a sidebar with options like Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, and Field Sets. The main content area is titled "Travel Approval Validation Rule" and displays the "Validation Rule Detail" section. It shows the rule name "Trip_end_date_after_start_date", the error condition formula "Trip_End_Date__c < Trip_Start_Date__c", and the error message "Trip end date must be greater than or equal to start date". The "Active" checkbox is checked. The "Error Location" is set to "Trip End Date". The "Created By" field shows "Ravi Teja Kuchipudi" with a timestamp of "2/21/2023, 12:18 PM". The "Modified By" field shows "Ravi Teja Kuchipudi" with a timestamp of "2/27/2023, 8:48 AM". There are "Edit" and "Clone" buttons at the bottom of the detail section.

Figure 24

The screenshot shows a 'Trip Info' form in Salesforce. The 'Trip End Date' field contains '10/02/2023'. A validation error message box is displayed, stating: 'We hit a snag.' and 'Review the following fields: • Trip_End_Date'. Below the form, a status bar shows 'Ravi Teja Kuchipudi, 20/02/2023, 8:53 pm' and 'Ravi Teja Kuchipudi, 21/02/2023, 12:38 pm'.

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.

The screenshot shows the Salesforce Object Manager. On the left, a sidebar lists various setup options like Page Layouts, Lightning Record Pages, and Buttons, Links, and Actions. The main area displays the 'Travel Approval' object's custom field 'Total Expenses'. The 'Custom Field Definition Detail' section shows the following details:

Field Information	Object Name
Field Label: Total Expenses	Travel Approval
Field Name: Total_Expenses	
API Name: Total_Expenses__c	
Description	
Help Text	
Data Owner	
Field Usage	
Data Sensitivity Level	
Compliance Categorization	
Created By: Ravi Teja Kuchipudi, 21/02/2023, 12:20 pm	Modified By: Ravi Teja Kuchipudi, 21/02/2023, 12:20 pm

The 'Roll-Up Summary Options' section includes:

- Data Type: Roll-Up Summary
- Summarized Object: Expense Item
- Field to Aggregate: Expense Item: Amount
- Filter Criteria: None
- Summary Type: SUM

Figure 26

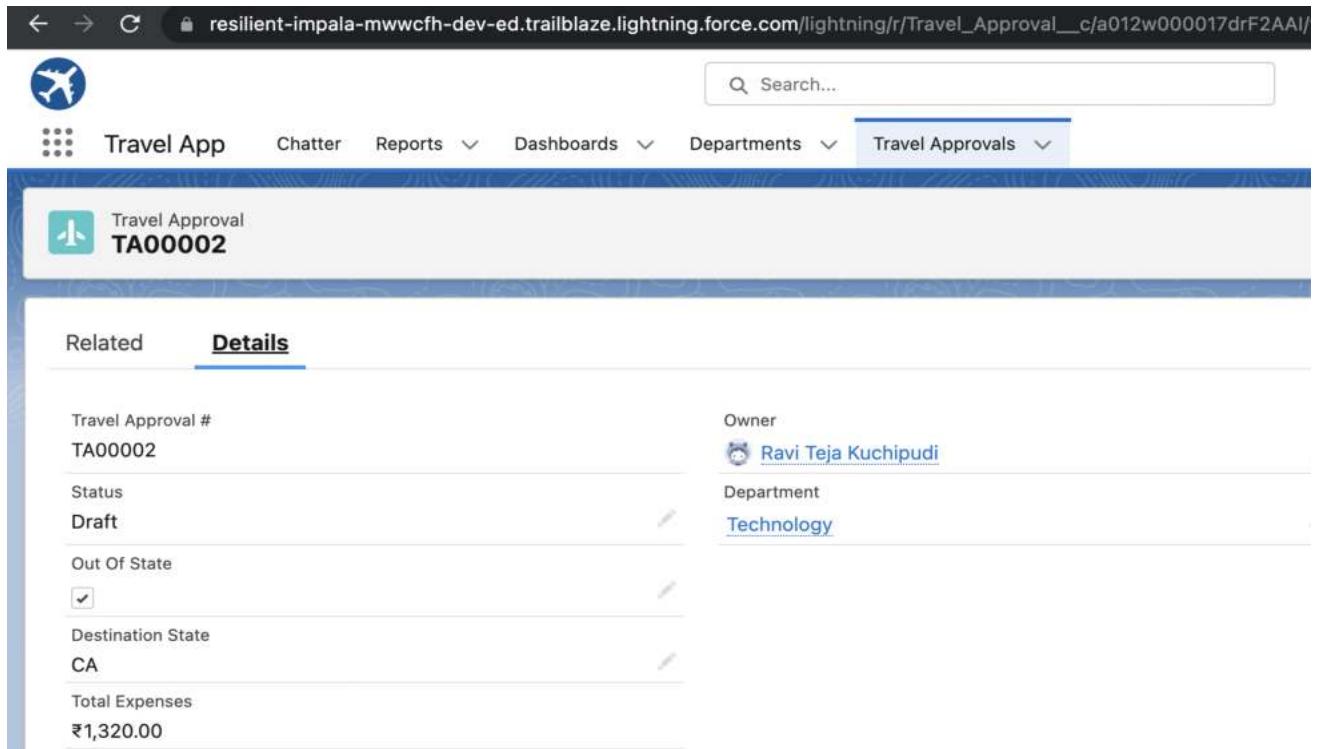


Figure 27

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

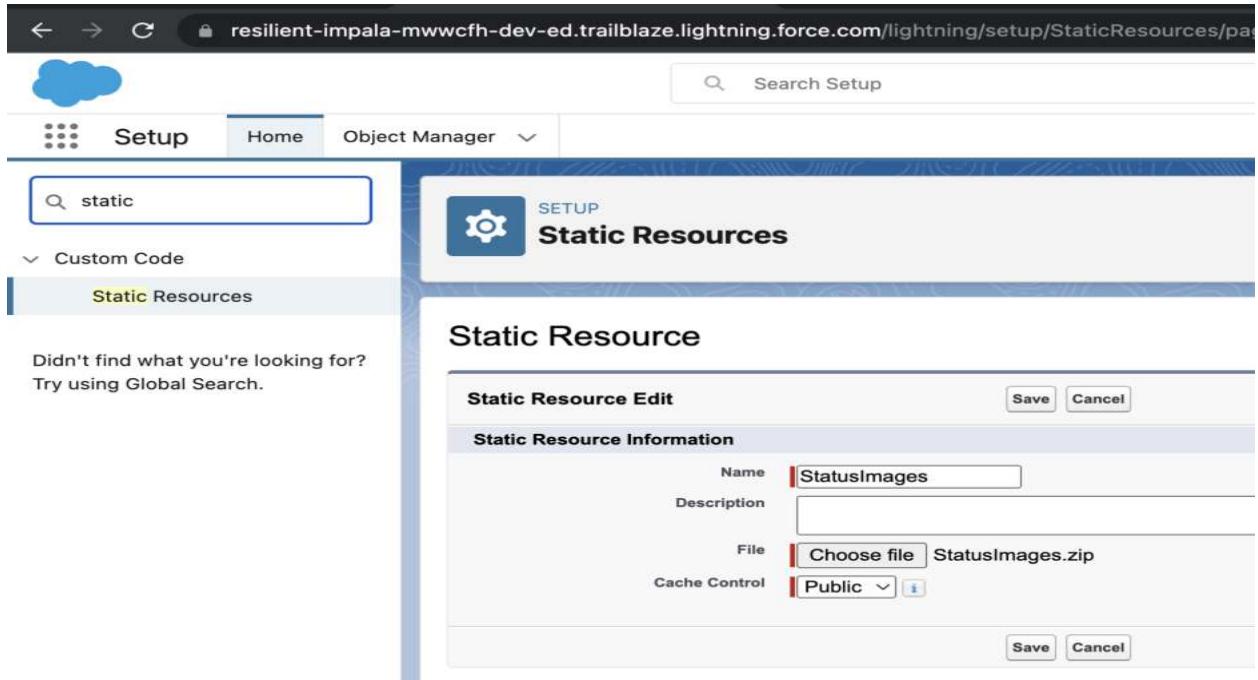


Figure 28

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

← → C resilient-impala-mwwcfh-dev-ed.trailblaze.lightning.force.com/lightning/r/Travel_Approval__c/a012w000017drF2AAI/

Travel App Chatter Reports Dashboards Departments Travel Approvals

Travel Approval
TA00002

Related Details

Travel Approval # TA00002 Owner Ravi Teja Kuchipudi

Status Approved Department Technology

Out Of State Edit Status

Destination State CA

Total Expenses ₹1,320.00

Status Indicator

Figure 30

← → C resilient-impala-mwwcfh-dev-ed.trailblaze.lightning.force.com/lightning/r/Travel_Approval__c/a012w000017drF2AAI/

Travel App Chatter Reports Dashboards Departments Travel Approvals

Travel Approval
TA00002

Related Details

Travel Approval # TA00002 Owner Ravi Teja Kuchipudi

Status Draft Department Technology

Out Of State

Destination State CA

Total Expenses ₹1,320.00

Status Indicator

Figure 31

Step 5:

1. Create a Record – Triggered Flow.

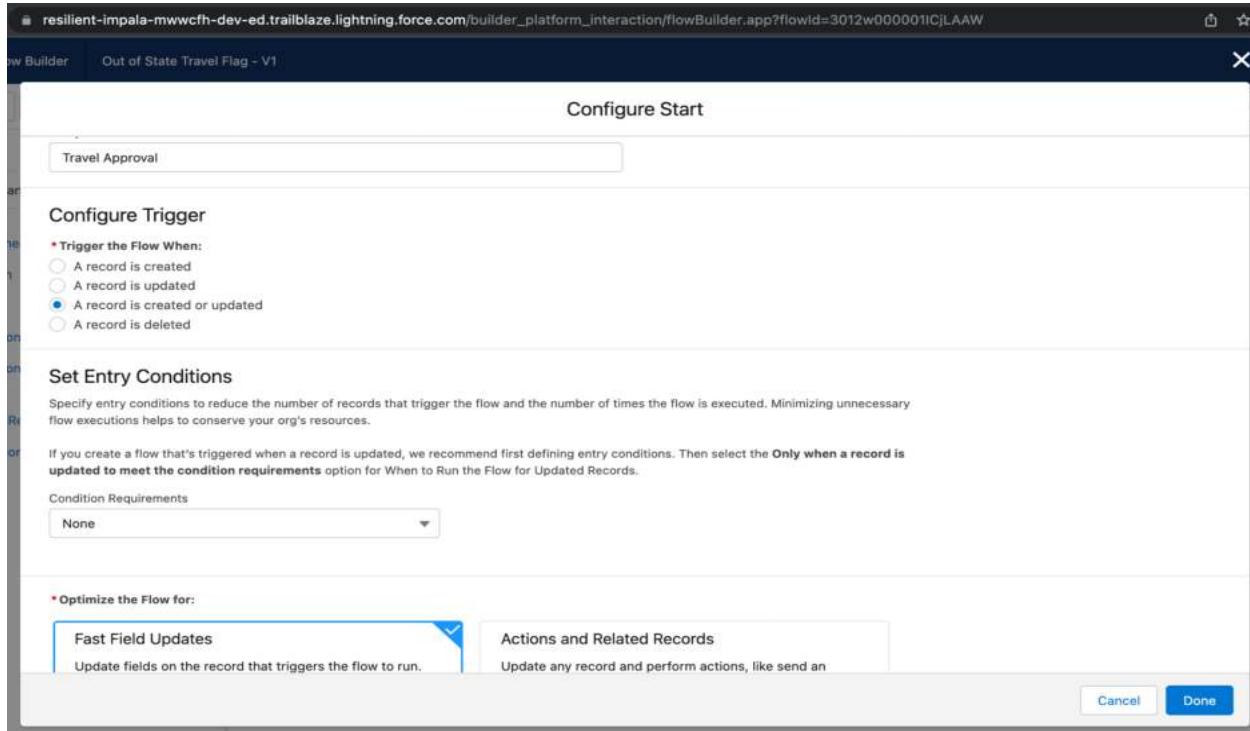


Figure 32

2. Add a Decision Element to the Flow.

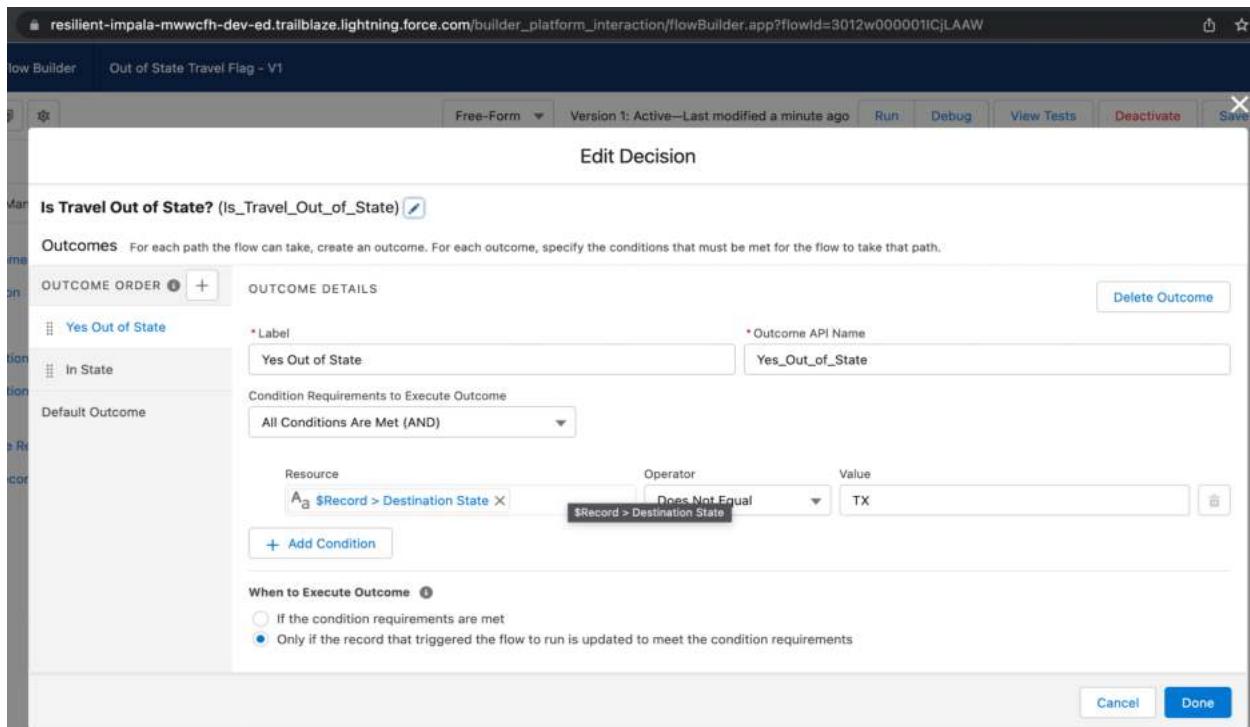


Figure 33

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

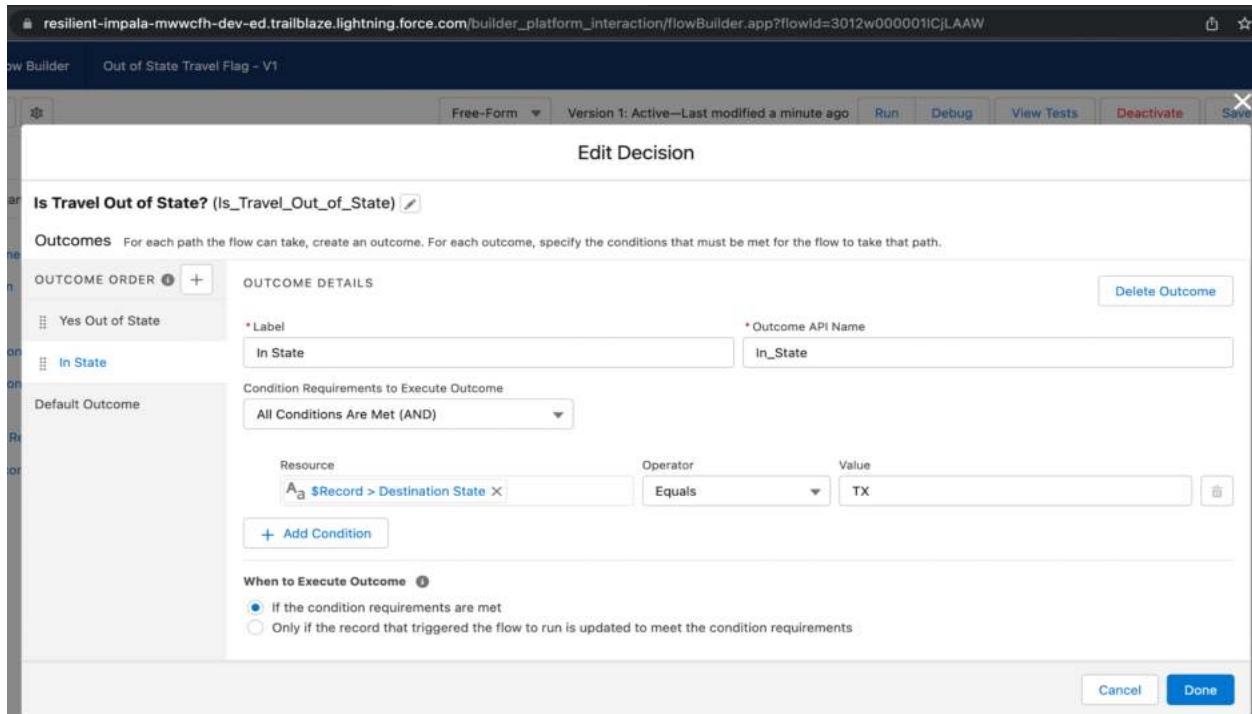


Figure 34

4. Create an Action for the Flow Using Update Records Elements.

5. Value = \$GlobalConstant.True

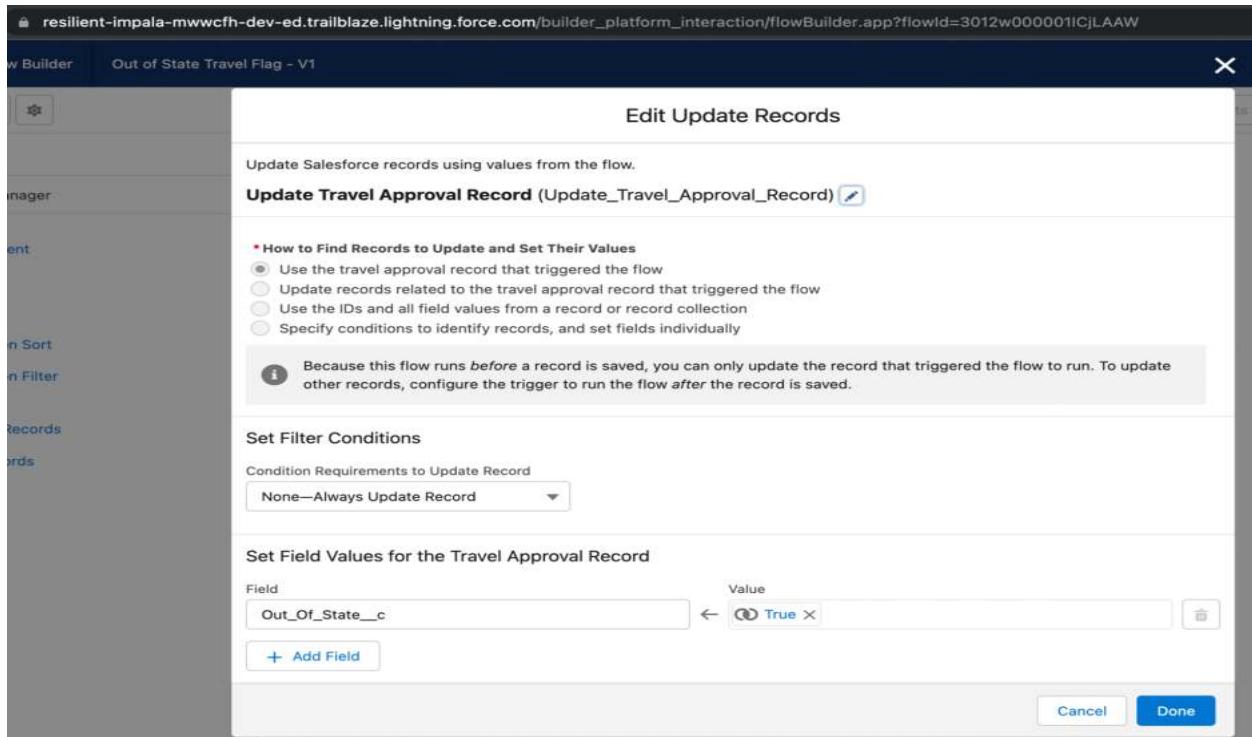


Figure 35

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

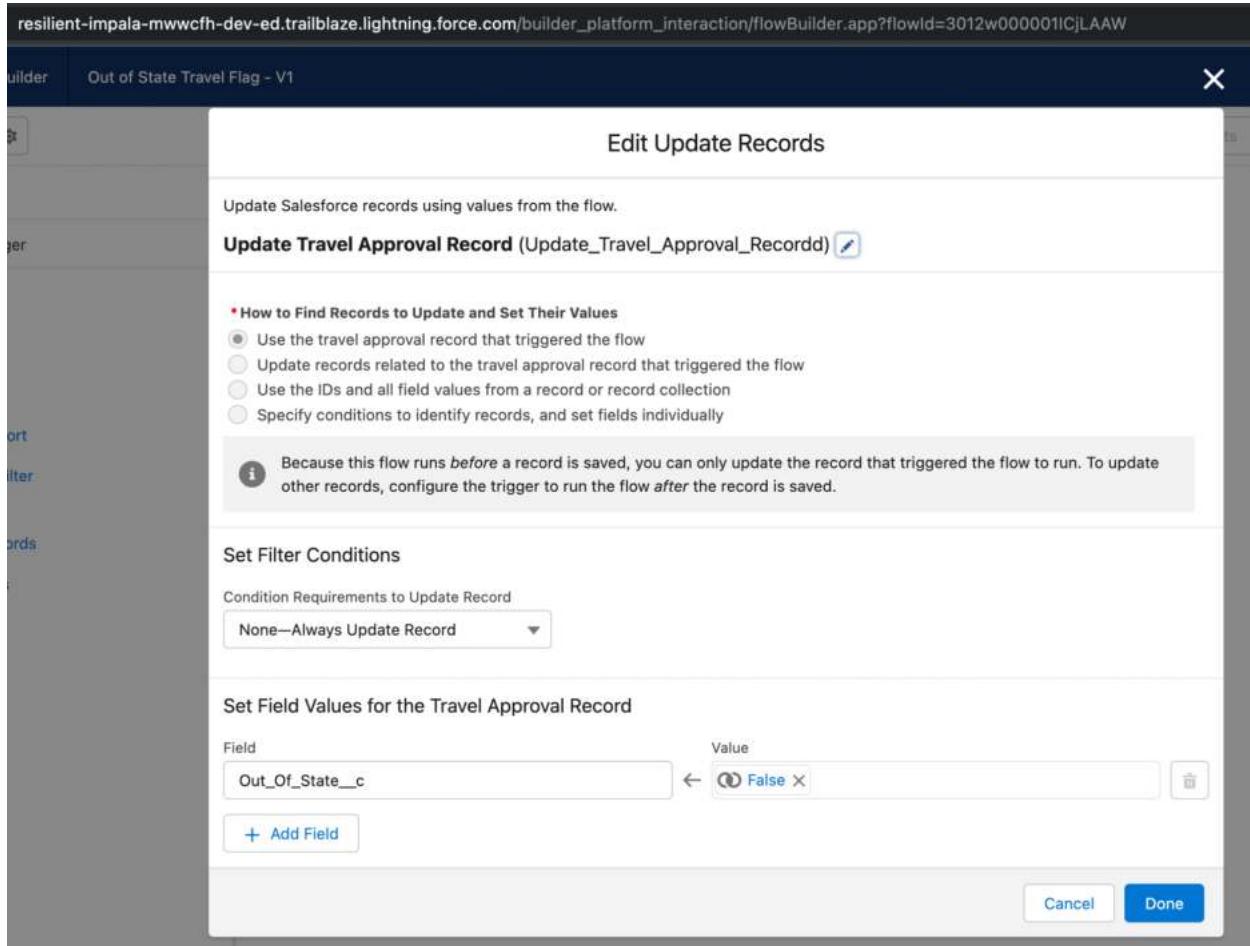


Figure 36

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.

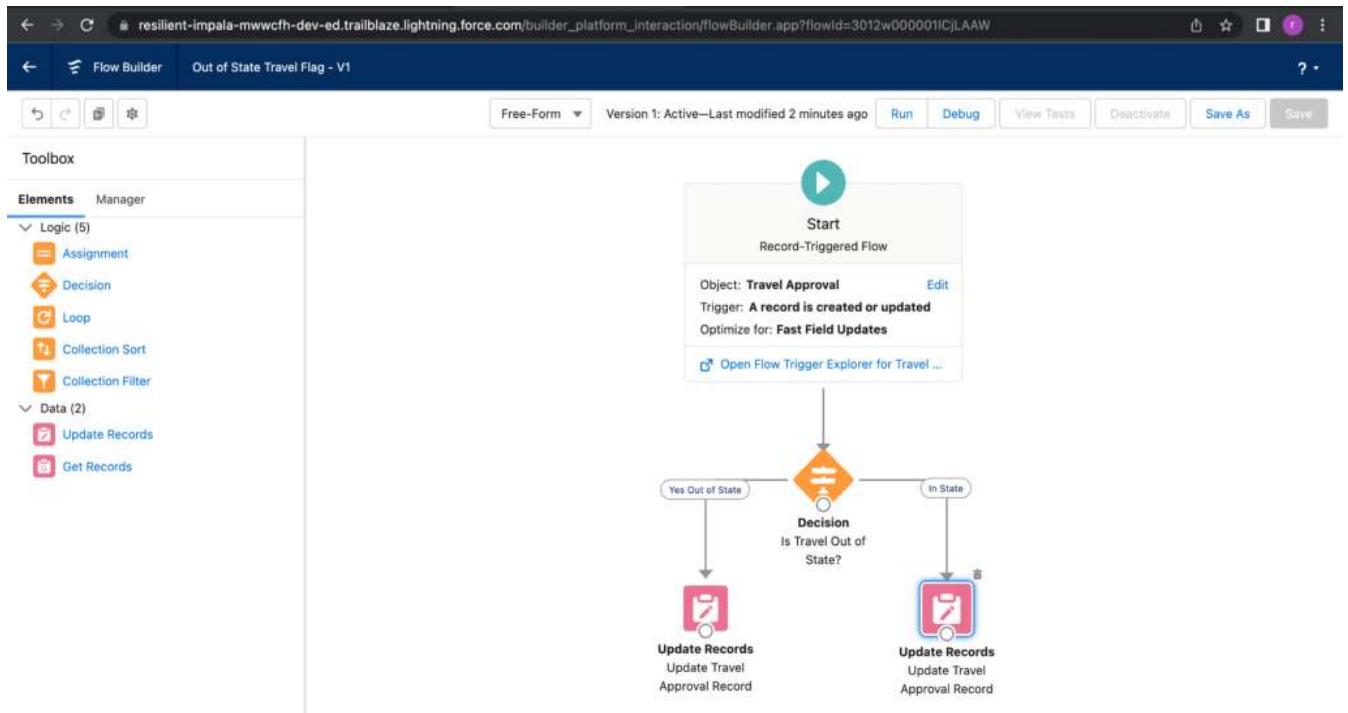


Figure 37

The screenshot shows the "Travel App" interface with a "Travel Approval" record detail view. The record ID is TA00002. The record details include:

- Travel Approval #:** TA00002
- Status:** Draft
- Out Of State:**
- Destination State:** TX
- Owner:** Ravi Teja Kuchipudi
- Department:** Technology

Figure 38

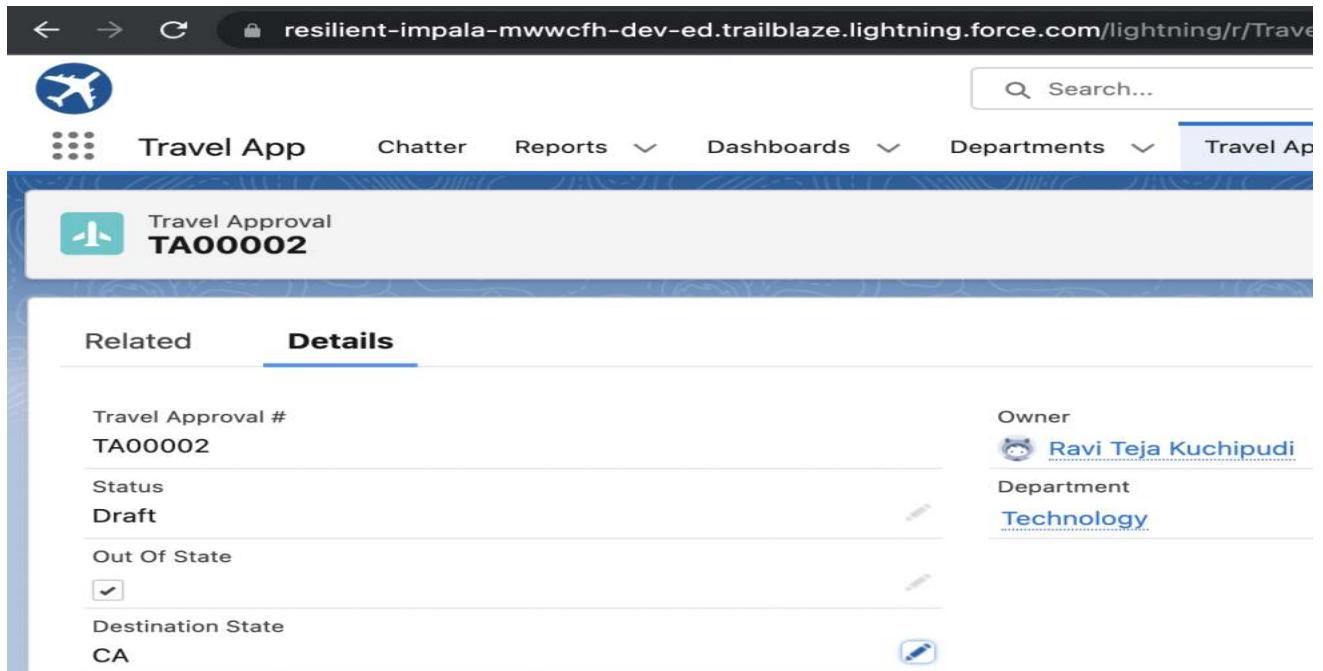


Figure 39

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.

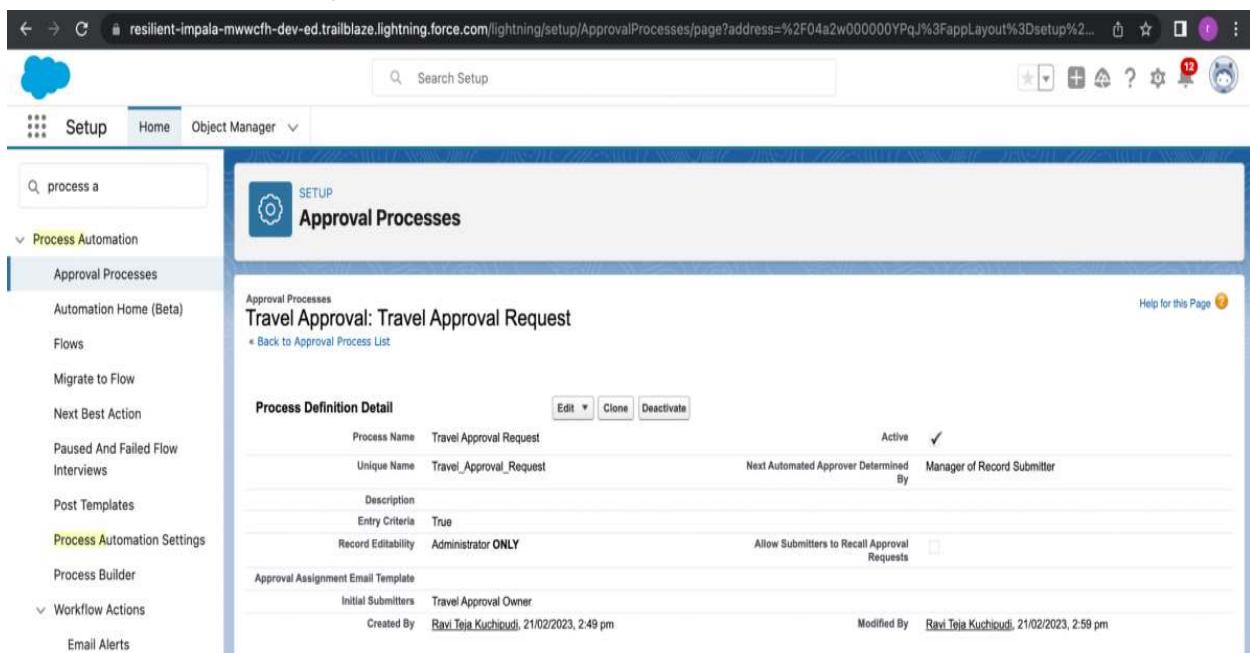


Figure 40

The screenshot shows the Salesforce Setup interface with the following details:

- Initial Submission Actions:**
 - Action Type: Record Lock
 - Description: Lock the record from being edited
- Approval Steps:**

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions Edit	1	Step 1			Manager	Final Rejection
Show Actions Edit	2	Travel_Coordinator_Approval		Travel Approval: Out Of State EQUALS True	User:Eric Executive	Final Rejection
- Final Approval Actions:**
 - Action: Edit
 - Type: Record Lock
 - Description: Lock the record from being edited
 - Action: Edit | Remove
 - Type: Field Update
 - Description: Set Status to Approved
- Final Rejection Actions:**
 - Action: Edit
 - Type: Record Lock
 - Description: Unlock the record for editing
 - Action: Edit | Remove
 - Type: Field Update
 - Description: Set Status to Rejected

Figure 41

The screenshot shows the Travel App interface with the following details:

Approval Requests
Items to Approve

2 items • Sorted by Date Submitted • Filtered by requests assigned to me and my queues • Updated a few seconds ago

Related To	Type	Most Recent Approver	Date Submitted
1 TA00003	Travel Approval	Ravi Teja Kuchipudi	21/02/2023, 3:12 pm
2 TA00002	Travel Approval	Ravi Teja Kuchipudi	21/02/2023, 3:12 pm

Figure 42

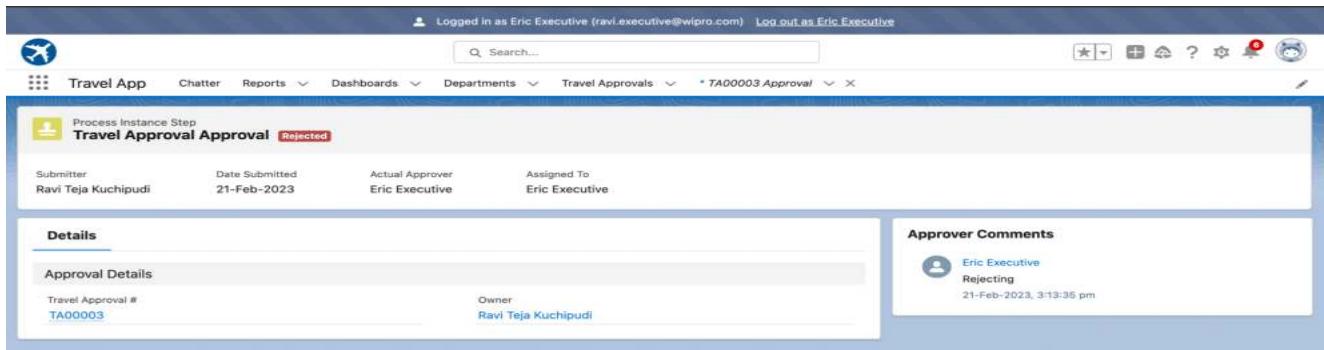


Figure 43

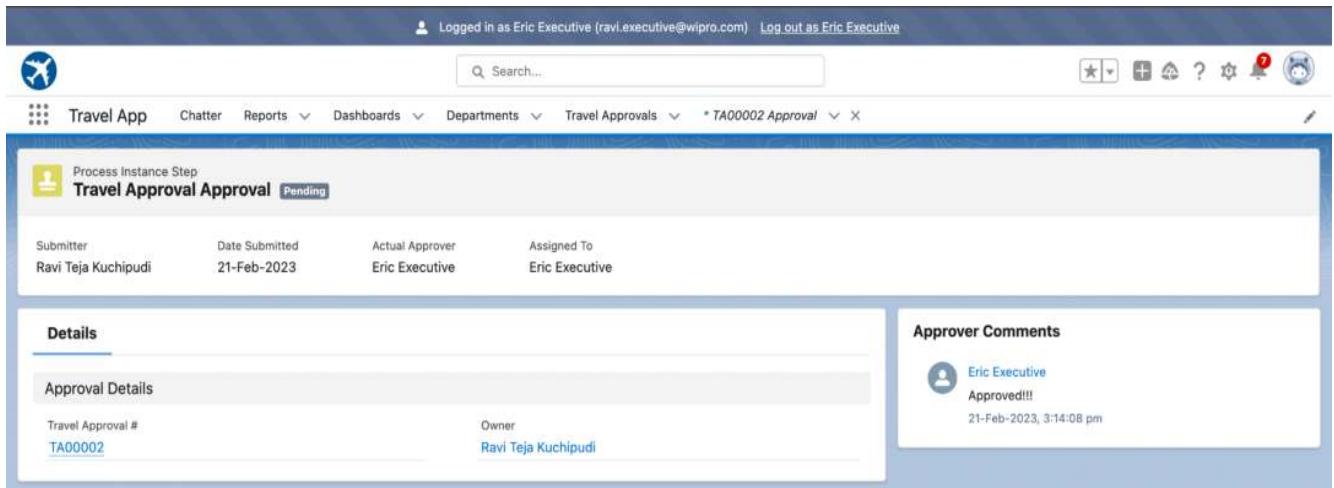


Figure 44

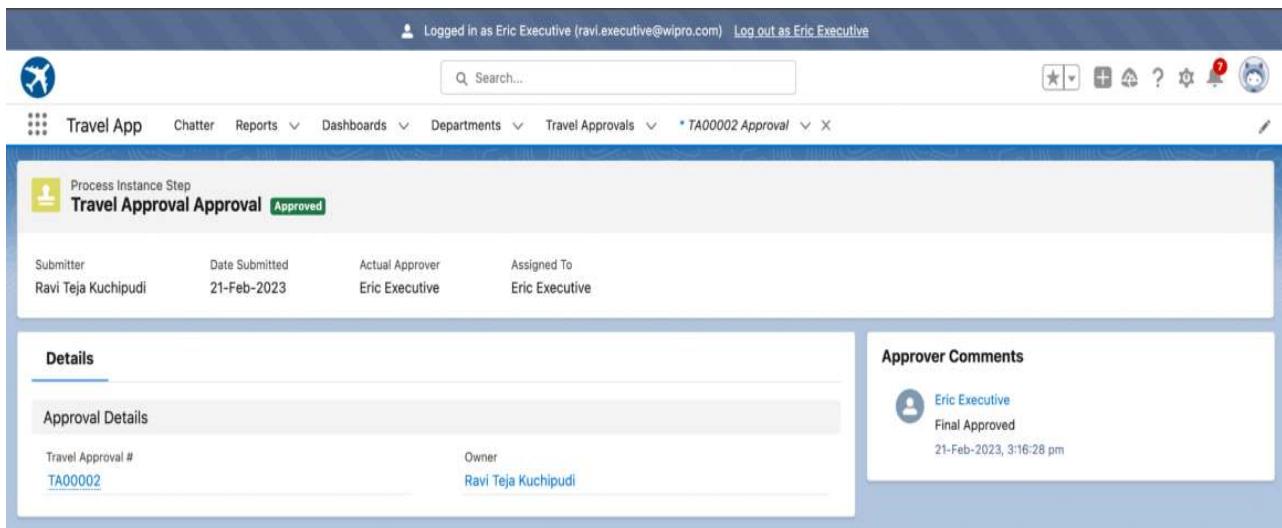
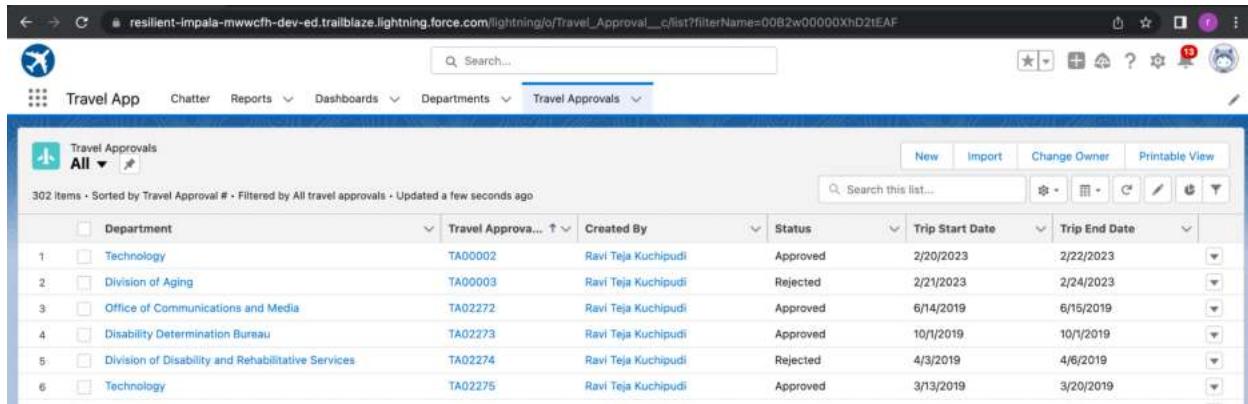


Figure 45

Exercise 2:

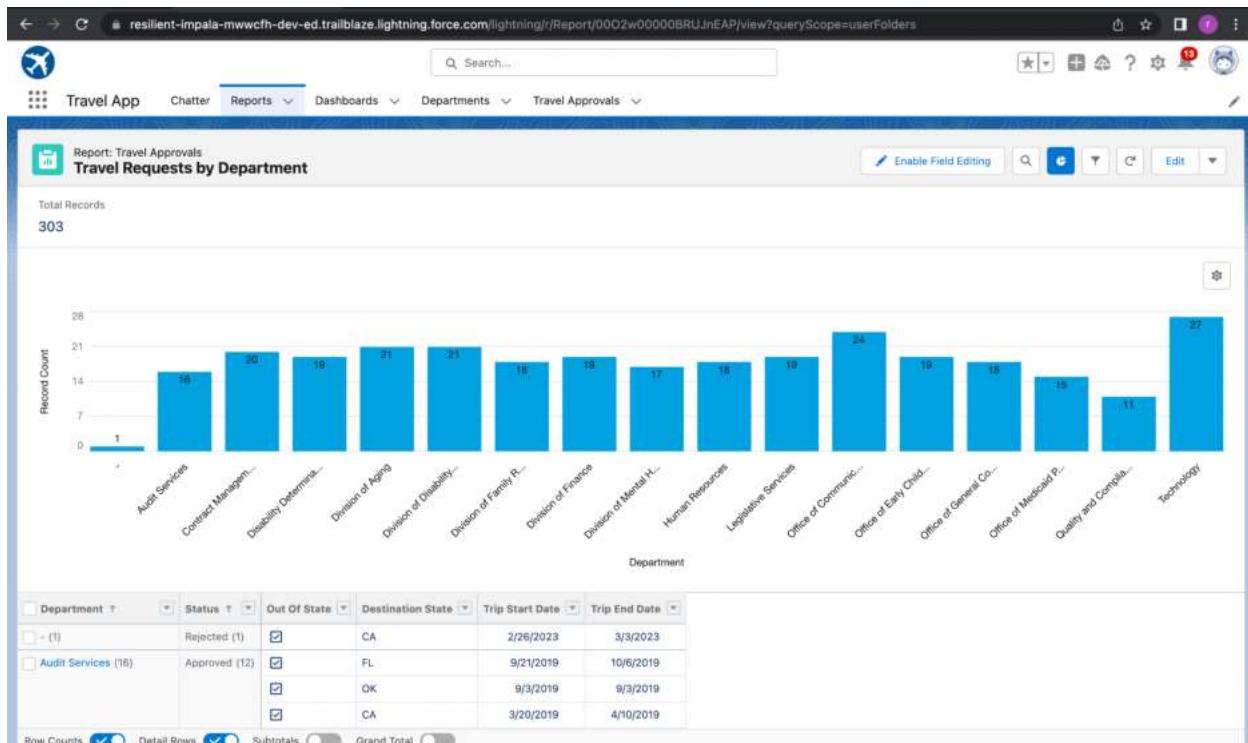
Step 1: Use Data Import Wizard to import Travel Approval records.



The screenshot shows a Salesforce Lightning page titled "Travel Approvals". The URL is https://resilient-impala-mwwcfh-dev-ed.lightning.force.com/lightning/o/Travel_Approval__c/list?filterName=0082w00000XhD2tEAF. The page displays a table with 302 items, sorted by Travel Approval #. The columns include Department, Travel Approval #, Created By, Status, Trip Start Date, and Trip End Date. The data shows various travel approvals from different departments like Technology, Division of Aging, and Office of Communications and Media, with statuses such as Approved and Rejected, and trip dates ranging from 2019 to 2023.

Figure 46

Step 2: Create a Travel Request by Department Report.



The screenshot shows a Salesforce Lightning report titled "Report: Travel Approvals" with the specific report name "Travel Requests by Department". The URL is <https://resilient-impala-mwwcfh-dev-ed.lightning.force.com/lightning/r/Report/00C2w00000GBRUJnEAP/view?queryScope=userFolders>. The report includes a bar chart showing the total record count for each department: Audit Services (1), Contract Management (20), Disability Determination (19), Division of Aging (21), Division of Disability (21), Division of Family R- (18), Division of Finance (19), Division of Mental H- (17), Human Resources (16), Legislative Services (19), Office of Communit- (24), Office of Early Child- (19), Office of General Co- (18), Office of Medicaid P- (16), Quality and Complain- (11), and Technology (27). Below the chart is a detailed table of travel requests for the Audit Services department, showing rows for rejected and approved trips to various states and dates.

Figure 47

Step 3: Create a Travel Request by Month Report.

The screenshot shows a Salesforce Lightning report titled "Travel Requests by Month". The report displays travel approvals for January 2019. The data is organized by department, showing the number of trips, status, destination state, and trip start date. The report includes a subtotal for January and a total for all months.

Trip End Date	Out Of State	Department	Status	Destination State	Trip Start Date
January 2019 (17)	(6)	Contract Management	Approved	TX	1/18/2019
		Quality and Compliance Office	Approved	TX	1/14/2019
		Disability Determination Bureau	Approved	TX	1/12/2019
		Legislative Services	Approved	TX	1/20/2019
		Office of General Counsel	Rejected	TX	1/27/2019
		Division of Aging	Approved	TX	1/22/2019
Subtotal		Contract Management	Approved	FL	1/29/2019
<input checked="" type="checkbox"/> (11)		Contract Management	Approved	GA	1/30/2019
		Office of Communications and Media	Rejected	FL	1/12/2019
		Division of Aging	Approved	OK	1/24/2019
		Division of Aging	Approved	FL	1/3/2019
		Division of Disability and Rehabilitative Services	Approved	OK	1/23/2019
		Division of Disability and Rehabilitative Services	Approved	FL	1/29/2019
		Division of Family Resources	Approved	FL	1/21/2019
		Division of Mental Health and Addiction	Approved	OK	1/3/2019
		Office of Early Childhood and Out-of-School Learning	Rejected	OK	1/4/2019

Figure 48

Step 4: Create a Travel Approvals Dashboard.

The screenshot shows a Salesforce Lightning dashboard titled "Travel Requests Dashboard". The dashboard includes two charts: "Travel Requests by Department" (a bar chart) and "Travel Requests by Month" (a stacked bar chart).

Travel Requests by Department:

Department	Record Count
Audit Services	16
Contract Management	20
Disability Determination B...	19
Division of Aging	21
Division of Disability and ...	21
Division of Family Resour...	18
Division of Finance	19
Division of Mental Health ...	17
Human Resources	18
Legislative Services	19
Office of Communication...	24
Office of Early Childhood ...	19

Travel Requests by Month:

Trip End Date	Out Of State	Record Count
January 2...	false	~18
February 2...	true	~22
March 2019	false	~25
April 2019	true	~30
May 2019	false	~22
June 2019	true	~25
July 2019	false	~22
August 2019	true	~25
September...	false	~22
October 2...	true	~25
November...	false	~22
December...	true	~25
January 2...	false	~22
February 2...	true	~25

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.

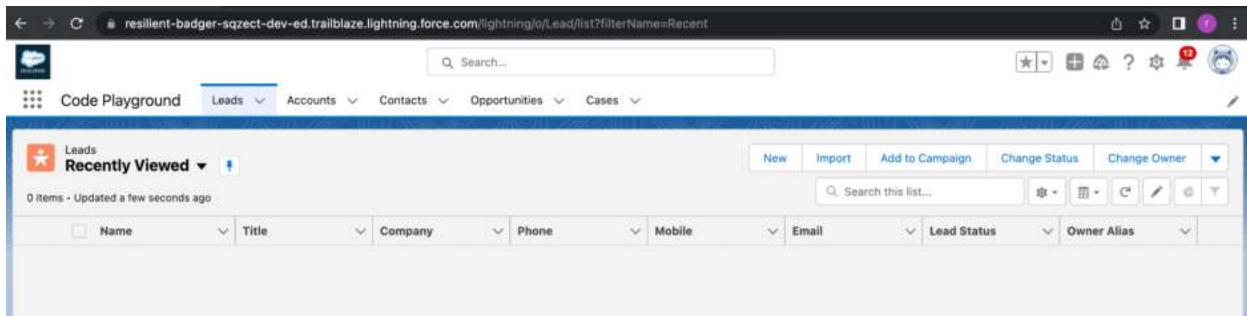


Figure 50

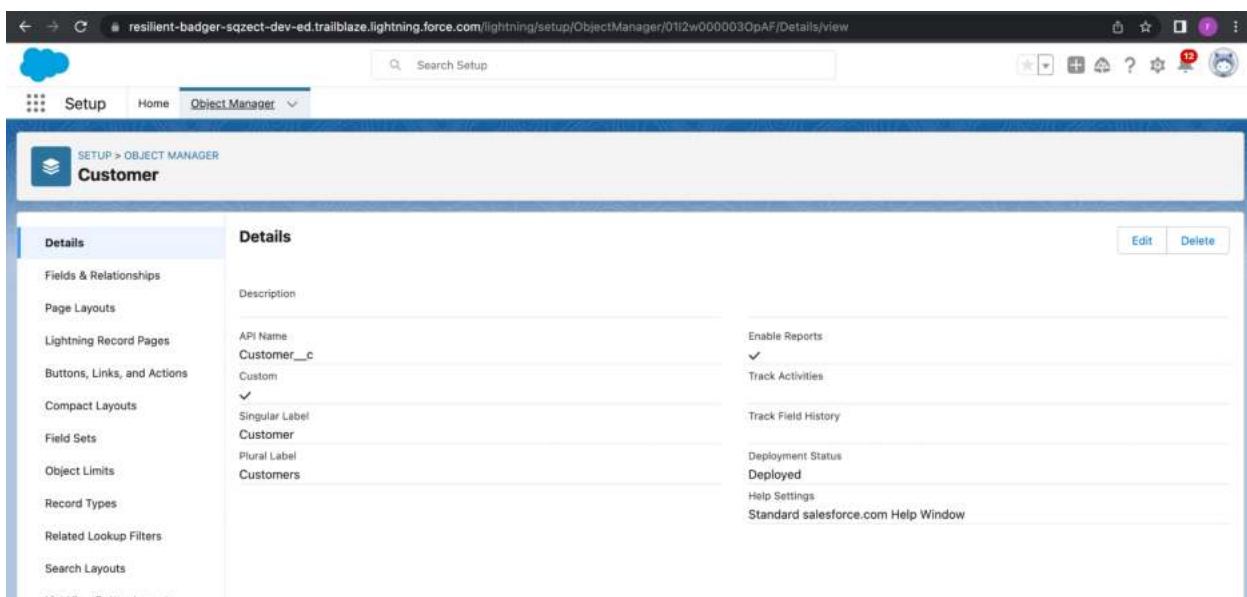


Figure 51

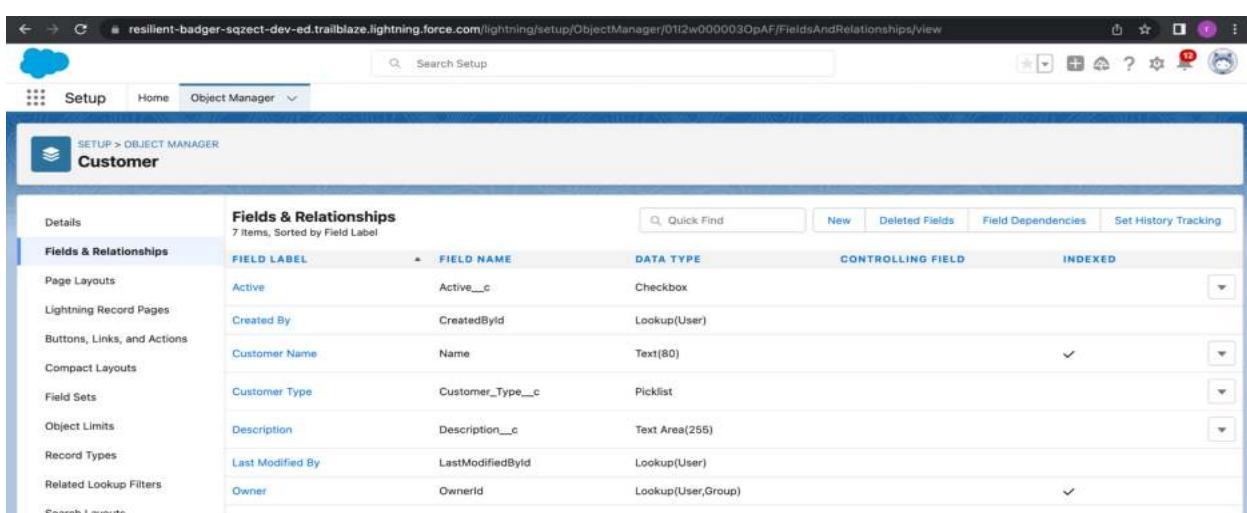


Figure 52

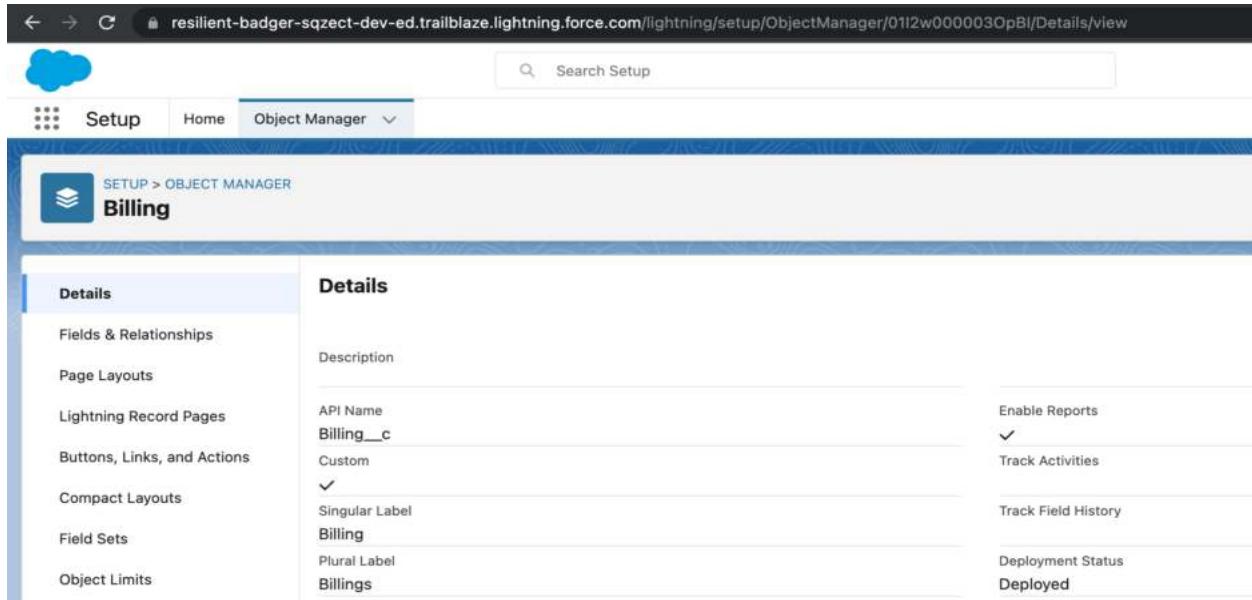


Figure 53

Fields & Relationships					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Amount Paid	Amount_Paid__c	Currency(18, 0)		
Lightning Record Pages	Bill Number	Name	Auto Number		
Buttons, Links, and Actions	Created By	CreatedBy	Lookup(User)		
Compact Layouts	Customer	Customer__c	Master-Detail(Customer)		
Field Sets	Customer Type	Customer_Type__c	Picklist		
Object Limits	Last Modified By	LastModifiedBy	Lookup(User)		
Record Types	Status	Status__c	Picklist		
Related Lookup Filters					
Search Layouts					

Figure 54

	Account Name	Account Site	Billing State/Province	Phone	Type	Account Owner All...
1	Burlington Textiles Corp of America	NC	(336) 222-7000		Customer - Direct	RKuch
2	Dickenson plc	KS	(785) 241-6200		Customer - Channel	RKuch
3	Edge Communications	TX	(512) 757-6000		Customer - Direct	RKuch
4	Express Logistics and Transport	OR	(503) 421-7800		Customer - Channel	RKuch
5	GenePoint	CA	(650) 867-3450		Customer - Channel	RKuch
6	Grand Hotels & Resorts Ltd	IL	(312) 596-1000		Customer - Direct	RKuch
7	Pyramid Construction Inc.		(014) 427-4427		Customer - Channel	RKuch
8	Sample Account for Entitlements					autopro
9	sForce	CA	(415) 901-7000			RKuch
10	United Oil & Gas Corp.	NY	(212) 842-5500		Customer - Direct	RKuch
11	United Oil & Gas, Singapore	Singapore	(650) 450-8810		Customer - Direct	RKuch
12	United Oil & Gas, UK	UK	+44 191 4956203		Customer - Direct	RKuch
13	University of Arizona	AZ	(520) 773-9050		Customer - Direct	RKuch

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.

```

String country = 'India';
System.debug(country.endsWith('ia'));
System.debug(country.endsWith('ab'));

```

Timestamp	Event	Details
18:38:46:003	USER_DEBUG	[2] DEBUG true
18:38:46:003	USER_DEBUG	[3] DEBUG false

Logs		Tests	Checkpoints	Query Editor	View State	Progress	Problems
User	Application	Operation	Time	Status			
Ravi Teja Kuchipudi	Unknown	/services/data/v57.0/tooling/execut...	21/02/2023, 18:38:46	Success			
Ravi Teja Kuchipudi	Unknown	/services/data/v57.0/tooling/execut...	21/02/2023, 18:38:10	Success			

Figure 56

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

The screenshot shows the Salesforce Apex Dev Console interface. At the top, there's a navigation bar with links like File, Edit, Debug, Test, Workspace, Help, and a back/forward button. Below the bar, three log entries are shown: "Log executeAnonymous @21/02/2023, 18:43:17", "Log executeAnonymous @21/02/2023, 18:44:00", and "Log executeAnonymous @21/02/2023, 18:45:47".

Execution Log:

Timestamp	Event	Details
18:45:47:003	USER_DEBUG	[3] DEBUG 2023-02-21 00:00:00
18:45:47:003	USER_DEBUG	[4] DEBUG 2023-03-23 00:00:00

Enter Apex Code:

```

1  Date date1 = Date.today();
2  Date date2 = Date.today().addDays(30);
3  System.debug(date1);
4  System.debug(date2);
5
6
7
8
9
10
11
12
13
14

```

At the bottom of the code editor are buttons for Open Log, Execute, and Execute Highlighted.

Logs Tab:

User	Application	Operation	Time	Status
Ravi Teja Kuchipudi	Unknown	/services/data/v57.0/tooling/execut...	21/02/2023, 18:45:47	Success

Figure 57

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

The screenshot shows the Salesforce Apex Dev Console interface. At the top, there's a navigation bar with links like File, Edit, Debug, Test, Workspace, Help, and a back/forward button. Below the bar, a single log entry is shown: "Log executeAnonymous @21/02/2023, 18:49:47".

Execution Log:

Timestamp	Event	Details
18:49:47:003	USER_DEBUG	[5] DEBUG 10
18:49:47:003	USER_DEBUG	[7] DEBUG 30

Enter Apex Code:

```

1
2
3  String s = '10';
4  Integer i = Integer.valueOf(s);
5  System.debug(i);
6  i += 20;
7  System.debug(i);
8
9
10
11
12
13
14

```

At the bottom of the code editor are buttons for Open Log, Execute, and Execute Highlighted.

Logs Tab:

User	Application	Operation	Time	Status
Ravi Teja Kuchipudi	Unknown	/services/data/v57.0/tooling/execut...	21/02/2023, 18:49:47	Success

Figure 58

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

The screenshot shows the Salesforce Trailblaze Dev Console interface. At the top, the URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. Below the URL is a navigation bar with links for File, Edit, Debug, Test, Workspace, and Help. A tab bar shows two logs: "Log executeAnonymous @21/02/2023, 18:49:47" and "Log executeAnonymous @21/02/2023, 18:51:37". The main area is divided into two panes. The left pane is titled "Enter Apex Code" and contains the following Apex code:

```
1  
2  
3 String country = 'India';  
4 System.debug(country.length());  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14
```

The right pane is titled "Execution Log" and shows the following log entry:

Timestamp	Event	Details
18:51:37:003	USER_DEBUG	[4] DEBUG 5

At the bottom of the code editor, there are three buttons: "Open Log" (checked), "Execute", and "Execute Highlighted". Below the code editor is a toolbar with filters: "This Frame", "Executable", "Debug Only" (checked), "Filter", and "Click here to filter the log". The bottom section is a table titled "Logs" with the following data:

User	Application	Operation	Time	Status
Ravi Teja Kuchipudi	Unknown	/services/data/v57.0/tooling/execut...	21/02/2023, 18:51:37	Success

Figure 59

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

The screenshot shows the Salesforce Execute Anonymous tool interface. At the top, the URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. Below the header, there's a menu bar with File, Edit, Debug, Test, Workspace, Help, and a back/forward button. A toolbar below the menu has buttons for Log, executeAnonymous, and a timestamp (21/02/2023, 19:03:16).

Execution Log

Timestamp	Event	Details
19:03:16:003	USER_DEBUG	[6] DEBUG =====get()=====
19:03:16:003	USER_DEBUG	[7] DEBUG 10
19:03:16:003	USER_DEBUG	[8] DEBUG 20
19:03:16:003	USER_DEBUG	[9] DEBUG 30
19:03:16:003	USER_DEBUG	[10] DEBUG =====set()=====
19:03:16:003	USER_DEBUG	[14] DEBUG (45, 55, 65)
19:03:16:003	USER_DEBUG	[15] DEBUG =====clear()=====
19:03:16:003	USER_DEBUG	[17] DEBUG ()

Enter Apex Code

```

1  List<Integer> integers = new List<Integer>();
2  integers.add(10);
3  integers.add(20);
4  integers.add(30);
5  System.debug('=====get()=====');
6  System.debug(integers.get(0));
7  System.debug(integers.get(1));
8  System.debug(integers.get(2));
9  System.debug('=====set()=====');
10 integers.set(0, 45);
11 integers.set(1, 55);
12 integers.set(2, 65);
13 System.debug(integers);
14 System.debug('=====clear()=====');
15 integers.clear();
16 System.debug(integers);
17

```

Buttons at the bottom right: Open Log, Execute.

Log details at the bottom:

- This Frame
- Executable
- Debug Only
- Filter
- Click here to filter the log

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...	21/02/2023, 19:03:16	Success		

Figure 60

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

The screenshot shows the Salesforce Execute Anonymous tool interface. At the top, the URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. Below the header, there's a menu bar with File, Edit, Debug, Test, Workspace, Help, and a back/forward button. A toolbar below the menu has buttons for Log, executeAnonymous, and a timestamp (21/02/2023, 19:22:03).

Execution Log

Timestamp	Event	Details
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 0
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 1
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 2
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 3
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 4
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 5
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 6
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 7
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 8
19:22:03:003	USER_DEBUG	[5] DEBUG Value of x is: 9

Enter Apex Code

```

1
2  Integer x = 20;
3  while(x > 12){
4    for(Integer x = 0; x < 10; x = x + 1){
5      system.debug('Value of x is: ' + x);
6    }
7    break;
8
9
10
11
12
13
14
15
16
17

```

Buttons at the bottom right: Open Log, Execute, Execut.

Log details at the bottom:

- This Frame
- Executable
- Debug Only
- Filter
- Click here to filter the log

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...	21/02/2023, 19:22:03	Success		

Figure 61

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

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

The screenshot shows the Salesforce Dev Console interface. At the top, the URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. Below the header, there's a toolbar with File, Edit, Debug, Test, Workspace, Help, and navigation buttons. A message bar says "Log executeAnonymous @21/02/2023, 19:25:33". The main area has tabs for Execution Log, Enter Apex Code, and Logs. The Execution Log tab shows a timestamp of 19:25:33:004, an event of USER_DEBUG, and details [5] DEBUG|false. The Enter Apex Code pane contains the following code:

```
1
2
3 Integer myluckyNumber = 15;
4 Integer myunluckyNumber = 7;
5 System.debug(myluckyNumber != myunluckyNumber + 8);
```

Below the code, there are buttons for Open Log, Execute, and Execute High. The Logs tab at the bottom shows a single entry: User Ravi Teja Kuchipudi, Application Unknown, Operation /services/data/v57.0/tooling/execut..., Time 21/02/2023, 19:25:33, Status Success, and Read.

Figure 62

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

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

The screenshot shows the Salesforce Dev Console interface. At the top, the URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. Below the header, there's a toolbar with File, Edit, Debug, Test, Workspace, Help, and navigation buttons. A message bar says "Log executeAnonymous @21/02/2023, 19:27:00". The main area has tabs for Execution Log, Enter Apex Code, and Logs. The Execution Log tab shows a timestamp of 19:27:00:003, an event of USER_DEBUG, and details [6] DEBUG|true. The Enter Apex Code pane contains the following code:

```
1
2 Boolean isTrue = true;
3 Boolean isFalse = false;
4 System.debug(isTrue || isFalse);
```

Below the code, there are buttons for Open Log, Execute, and Execute High. The Logs tab at the bottom shows a single entry: User Ravi Teja Kuchipudi, Application Unknown, Operation /services/data/v57.0/tooling/execut..., Time 21/02/2023, 19:27:00, Status Success, and Read.

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

The screenshot shows the Salesforce Developer Console interface. At the top, there's a menu bar with File, Edit, Debug, Test, Workspace, and Help. Below the menu is a toolbar with Log executeAnonymous @21/02/2023, 19:28:53. The main area is titled 'Execution Log' with tabs for Timestamp, Event, and Details. A timestamp of 19:28:53:003 and event USER_DEBUG [4] DEBUG|true are listed. Below this is a large text area labeled 'Enter Apex Code' containing the following code:

```
1
2 Date today = Date.today();
3 Date tomorrow = Date.today().addDays(1);
4 System.debug(today != tomorrow);
```

At the bottom right of the code editor are 'Open Log' and 'Execute' buttons. Below the code editor is a log viewer with filters for This Frame, Executable, Debug Only, and Filter. It also has a link to Click here to filter the log. The log table has columns for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. One log entry is shown:

User	Application	Operation	Time	Status
Ravi Teja Kuchipudi	Unknown	/services/data/v57.0/tooling/execut...	21/02/2023, 19:28:53	Success

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.

The screenshot shows the Salesforce Apex CS Page interface. At the top, the URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. Below the header is a menu bar with File, Edit, Debug, Test, Workspace, Help, and navigation arrows. A yellow banner at the top says "Log executeAnonymous @21/02/2023, 19:37:11". The main area has tabs for Execution Log, Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Execution Log tab is active, showing a timestamp of 19:37:11:004, an event of USER_DEBUG, and a detail of "[6]||DEBUG|Grade: A". Below this is an "Enter Apex Code" window containing the following Apex code:

```

1 |
2 integer score = 98;
3 if(score == 100)
4     System.debug ('Grade: A+');
5 else if(score >= 90)
6     System.debug ('Grade: A');
7 else if(score >= 80)
8     System.debug ('Grade: B');
9 else
10    System.debug ('Grade: Failed');
11
12
13
14
15
16
17

```

At the bottom of the code editor are checkboxes for "This Frame" and "Executable", and buttons for "Open Log", "Execute", and "Execute Highlighted". The Logs tab is selected, showing a table with one entry:

User	Application	Operation	Time	Status
Ravi Teja Kuchipudi	Unknown	/services/data/v57.0/tooling/execut...	21/02/2023, 19:37:11	Success

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.

```

1 * public class ApexForLoop {
2 *     public static void paidBillings(){
3         /*Adding Billing Records
4          List<Billing__c> billings = new List<Billing__c>();
5
6          Customer__c customer = new Customer__c ();
7          customer.Name = 'Ravi';
8          customer.Customer_Type__c = 'Premium';
9          insert customer;
10
11         for(integer i=0; i<5; i++){
12             Billing__c billing = new Billing__c();
13             billing.Amount_Paid__c = 10000;
14             billing.Customer_Type__c = 'Premium';
15             billing.Status__c = 'Paid';
16             billing.Customer__c = customer.Id;
17             billings.add(billing);
18         }
19         insert billings;
20
21         List<Billing__c> billingRecords = [select Id, Status__c, Name from Billing__c where CreatedDate = TODAY];
22         List<String> ids = new List<String>();
23         for(Billing__c billing: billingRecords){
24             if(billing.Status__c == 'Paid'){
25                 System.debug ('Value of Current Record on which Loop is iterating ' + billing);
26                 ids.add(billing.Name);
27             }
28         }
29         System.debug(ids);
30     }
31 }

```

Figure 66

Timestamp	Event	Details
20:36:03:114	USER_DEBUG	[25][DEBUG]Value of Current Record on which Loop is iterating Billing__c:{Id=a012w000017ZKnTAAW, Status__c=Paid, Name=B - 0019}
20:36:03:114	USER_DEBUG	[25][DEBUG]Value of Current Record on which Loop is iterating Billing__c:{Id=a012w000017ZKnEAAW, Status__c=Paid, Name=B - 0018}
20:36:03:114	USER_DEBUG	[25][DEBUG]Value of Current Record on which Loop is iterating Billing__c:{Id=a012w000017ZNZEEA4, Status__c=Paid, Name=B - 0020}
20:36:03:115	USER_DEBUG	[25][DEBUG]Value of Current Record on which Loop is iterating Billing__c:{Id=a012w000017ZNZF4A4, Status__c=Paid, Name=B - 0021}
20:36:03:115	USER_DEBUG	[25][DEBUG]Value of Current Record on which Loop is iterating Billing__c:{Id=a012w000017ZNZGAA4, Status__c=Paid, Name=B - 0022}
20:36:03:115	USER_DEBUG	[25][DEBUG]Value of Current Record on which Loop is iterating Billing__c:{Id=a012w000017ZNZHAA4, Status__c=Paid, Name=B - 0023}
20:36:03:115	USER_DEBUG	[25][DEBUG]Value of Current Record on which Loop is iterating Billing__c:{Id=a012w000017ZNZIAAA4, Status__c=Paid, Name=B - 0024}
20:36:03:115	USER_DEBUG	[29][DEBUG](B - 0019, B - 0018, B - 0020, B - 0021, B - 0022, B - 0023, B - 0024)

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.

The screenshot shows the Salesforce Dev Console interface. The main area displays the Apex code for the `DiscountClass`. The code defines a class with a final decimal constant `regularDiscount` set to 0.1, and a method `calculateDiscount` that returns the price minus the product of the price and `regularDiscount`. Below the code editor, an `Enter Apex Code` window is open, containing the following anonymous code:

```
1 DiscountClass dis = new DiscountClass();
2 System.debug('finalPrice: '+dis.calculateDiscount(100));
```

At the bottom of the Dev Console, there is a log table showing a single entry from the user Ravi Teja Kuchipudi. The log entry details an operation on the /services/data/v57.0/tooling/execut... endpoint at 21/02/2023, 21:16:26, which was successful and read 4.4 KB.

Figure 68

This screenshot of the Salesforce Dev Console shows the execution log and the anonymous code execution interface. The execution log table at the top shows a single log entry from the user Ravi Teja Kuchipudi at 21:16:26, which was a USER_DEBUG event with the details: [2][DEBUG]finalPrice: 90.0. Below the log is the `Enter Apex Code` window with the same anonymous code as in Figure 68. At the bottom, the log table shows the same successful operation as in 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.

```
1 public interface InterfaceExample {
2     Double percentageDiscountTobeApplied();
3 }
```

Figure 70

```
1 public class PremiumCustomer implements InterfaceExample {
2     public Double percentageDiscountTobeApplied(){
3         return 0.3;
4     }
5 }
```

Figure 71

```
1 public class normalCustomer implements InterfaceExample {
2     public Double percentageDiscountTobeApplied(){
3         return 0.1;
4     }
5 }
```

Figure 72

The screenshot shows the Salesforce Trailhead developer console interface. At the top, the URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. Below the URL, there's a menu bar with File, Edit, Debug, Test, Workspace, Help, and tabs for InterfaceExample.apxc, PremiumCustomer.apxc, normalCustomer.apxc, and Log executeAnonymous @21/02/2023, 21:25:43.

The main area has a title "Execution Log" with columns for Timestamp, Event, and Details. It shows two entries:

Timestamp	Event	Details
21:25:43:009	USER_DEBUG	[2]DEBUG 30.0
21:25:43:019	USER_DEBUG	[5]DEBUG 10.0

Below the log is a modal window titled "Enter Apex Code" containing the following Apex code:

```

1 InterfaceExample ie = new PremiumCustomer();
2 System.debug(ie.percentageDiscountTobeApplied() *100);
3
4 InterfaceExample ie1 = new normalCustomer();
5 System.debug(ie1.percentageDiscountTobeApplied() *100);
6

```

At the bottom of the modal are buttons for Open Log, Execute, and Execute Highlighted. Below the modal is a toolbar with checkboxes for This Frame, Executable, Debug Only, Filter, and a link to Click here to filter the log. The toolbar also includes a dropdown for Time and a Status section.

User	Application	Operation	Time	Status
Ravi Teja Kuchipudi	Unknown	/services/data/v57.0/tooling/execut...	21/02/2023, 21:25:43	Success

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.

← → C 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

```

1  public class DatabaseOperations {
2      public static void DBOps(){
3          //Insert a Customer Record First using simple DML Statement
4          Customer__c customer = new Customer__c ();
5          customer.Name = 'Wipro';
6          customer.Customer_Type__c = 'Premium';
7          insert customer;
8
9          Billing__c billing = new Billing__c ();
10         billing.Status__c = 'Paid';
11         billing.Amount_Paid__c = 5000000;
12         billing.Customer__c = customer.Id;
13
14         List<Billing__c> billings = new List<Billing__c>();
15         billings.add(billing);
16
17         //Insert Billing record using Database methods.
18         Database.SaveResult[ ] srList = Database.insert(billings, false) ;
19
20         for(Database.SaveResult sr : srList ){
21             if(sr.isSuccess())
22                 System.debug(sr);
23             else{
24                 Database.Error[ ] err = sr.getErrors();
25                 for(Database.Error e: err){
26                     System.debug(e.getMessage());
27
28                     System.debug(e.getStatusCode());
29                     System.debug(e.getFields());
30
31                 }
32             }
33         }
34     }

```

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	

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

The screenshot shows the Salesforce Developer Console interface. At the top, the URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. Below the header, there's a menu bar with File, Edit, Debug, Test, Workspace, Help, and navigation buttons. A tab labeled "DatabaseOperations.apxc" is open, with a sub-tab titled "Log executeAnonymous @22/02/2023, 07:39:41". The main area is divided into two sections: "Execution Log" and "Enter Apex Code". The "Execution Log" section has columns for Timestamp, Event, and Details. One entry shows a USER_DEBUG log at 07:39:41:089 with the message "[22]|DEBUG|Database.SaveResult[getErrors=();getId=a012w000017XUjDAAW;isSuccess=true;]". The "Enter Apex Code" section contains a code editor with the following Apex code:

```
1 DatabaseOperations.DBOps();
```

Below the code editor are three buttons: Open Log, Execute, and Execute Highlighted. The "Open Log" button is checked. The bottom part of the interface shows a "Logs" tab selected, displaying a table of logs. The table has columns for User, Application, Operation, Time, and Status. One log entry is shown:

User	Application	Operation	Time	Status
Ravi Teja Kuchipudi	Unknown	/services/data/v57.0/tooling/execut...	22/02/2023, 07:39:41	Success

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”

The screenshot shows a Salesforce developer console window. The URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The tab title is "DatabaseOperations.apxc" and the sub-tab is "Opportunity@10:03 PM". The query executed is:

```
SELECT Id, Amount, StageName, AccountId, Account.Industry, Account.Website FROM Opportunity WHERE Account.Industry = 'Energy' AND Account.AnnualRevenue > 5000
```

The results table has a header "Query Results - Total Rows: 10" and columns: Id, Amount, StageName, AccountId, Account.Industry, and Account.Website. The data rows are as follows:

ID	Amount	StageName	AccountId	Account.Industry	Account.Website
0062w00000K43nAAJ	125000	Negotiation/Review	0012w00001LdWjWAAC	Energy	http://www.ios.com
0062w00000K43nAAJ	270000	Proposal/Price Quote	0012w00001LdWjWAAC	Energy	http://www.ios.com
0062w00000K43nAAJ	120000	Closed Won	0012w00001LdWjWAAC	Energy	http://www.ios.com
0062w00000K43nAAJ	270000	Negotiation/Review	0012w00001LdWjWAAC	Energy	http://www.ios.com
0062w00000K43nAAJ	270000	Closed Won	0012w00001LdWjWAAC	Energy	http://www.ios.com
0062w00000K43nAAJ	915000	Closed Won	0012w00001LdWjWAAC	Energy	http://www.ios.com
0062w00000K43nAAZ	235000	Closed Won	0012w00001LdWjWAAC	Energy	http://www.ios.com
0062w00000K43nBAAZ	440000	Closed Won	0012w00001LdWjWAAC	Energy	http://www.ios.com
0062w00000K43nDAAZ	120000	Closed Won	0012w00001LdWjWAAC	Energy	http://www.ios.com
0062w00000K43nFAAZ	675000	Needs Analysis	0012w00001LdWjWAAC	Energy	http://www.ios.com

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.

The screenshot shows a Salesforce developer console window. The URL is `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The tab title is "CustomerTrigger.apxc" and the sub-tab is "CustomerTrigger.apxc". The code coverage is set to "None" and the API version is "57". The trigger code is:

```
trigger CustomerTrigger on Customer_c (after update) {
    if(Trigger.isAfter && Trigger.isUpdate){
        List<Billing_c> billings = new List<Billing_c>();
        for(Customer_c newC: Trigger.New){
            if( Trigger.OldMap.get(newC.Id).Active_c == false && newC.Active_c == true){
                Billing_c billing = new Billing_c ();
                billing.Amount_Paid_c = 1000000 ;
                billing.Status_c = 'Paid';
                billing.Customer_c = newC.Id;
                billings.add(billing);
            }
        }
        if(billings.size() > 0){
            insert billings;
        }
    }
}
```

Figure 77

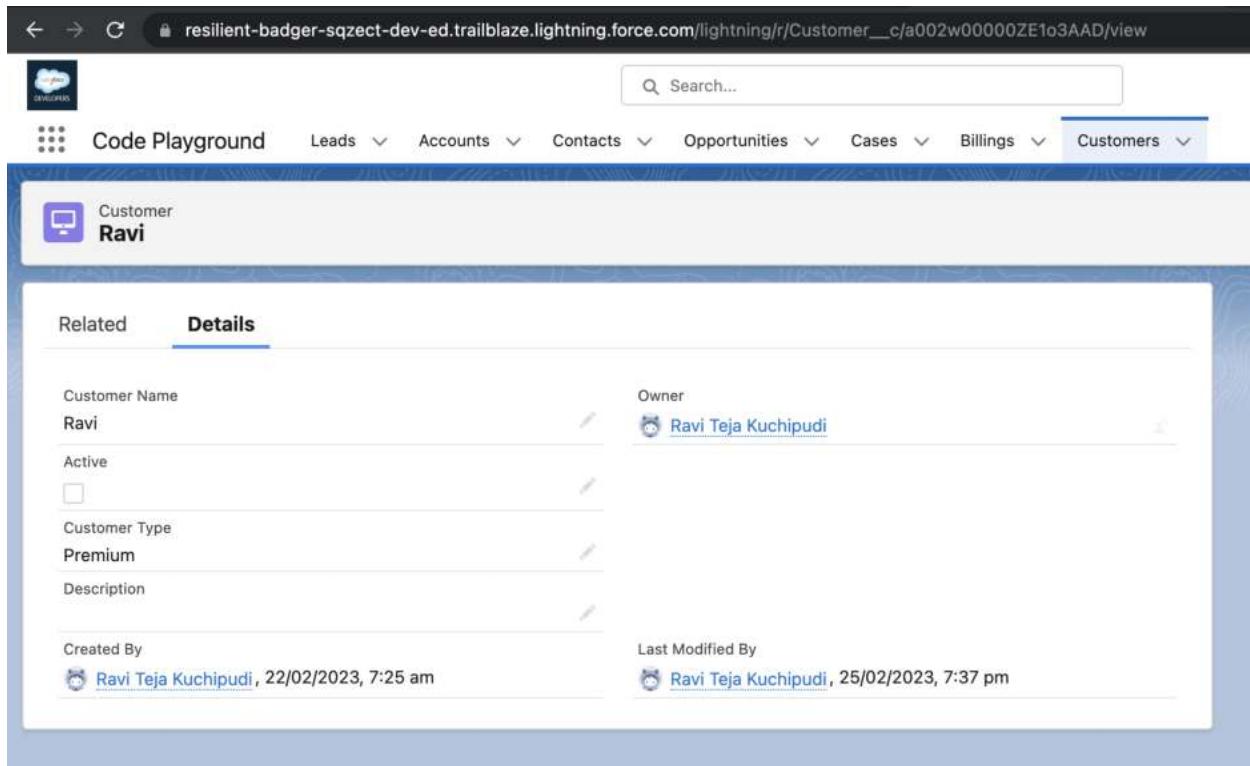


Figure 78

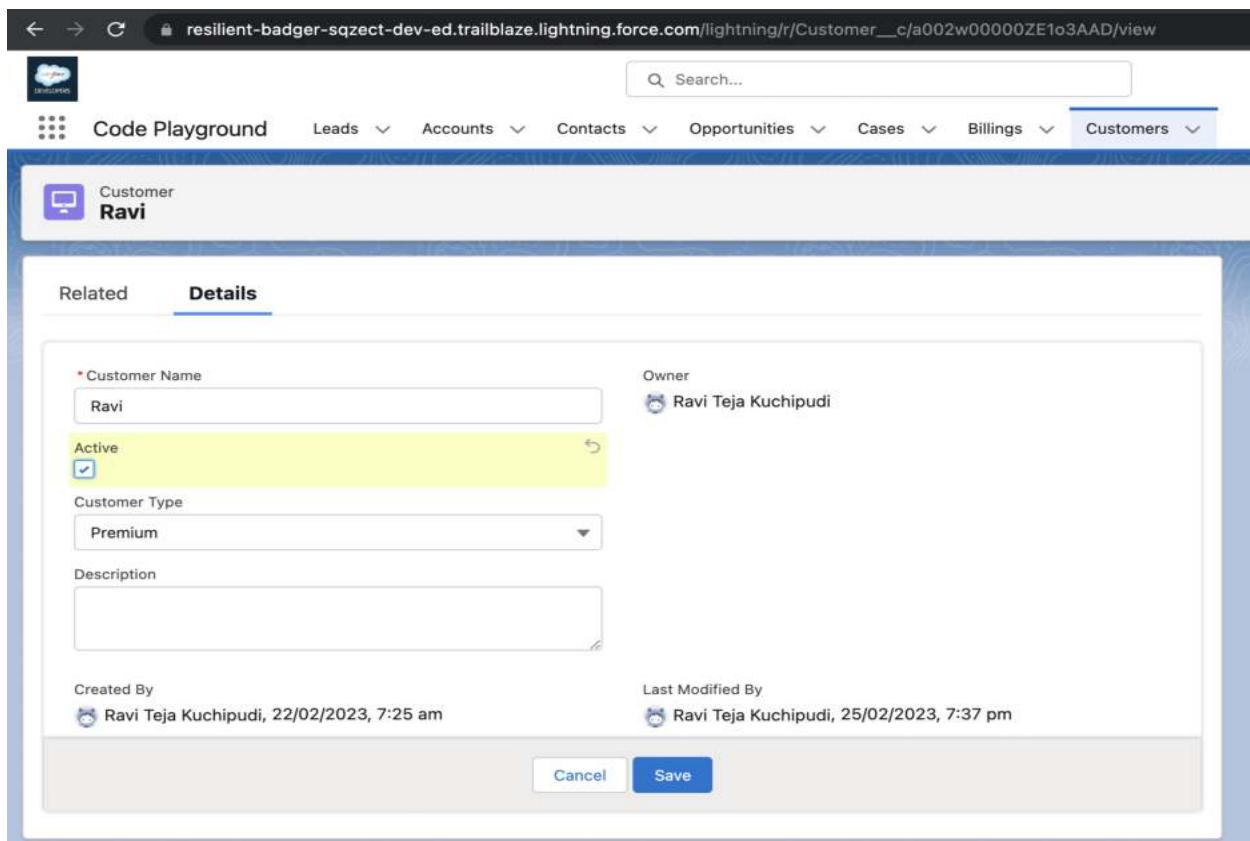


Figure 79

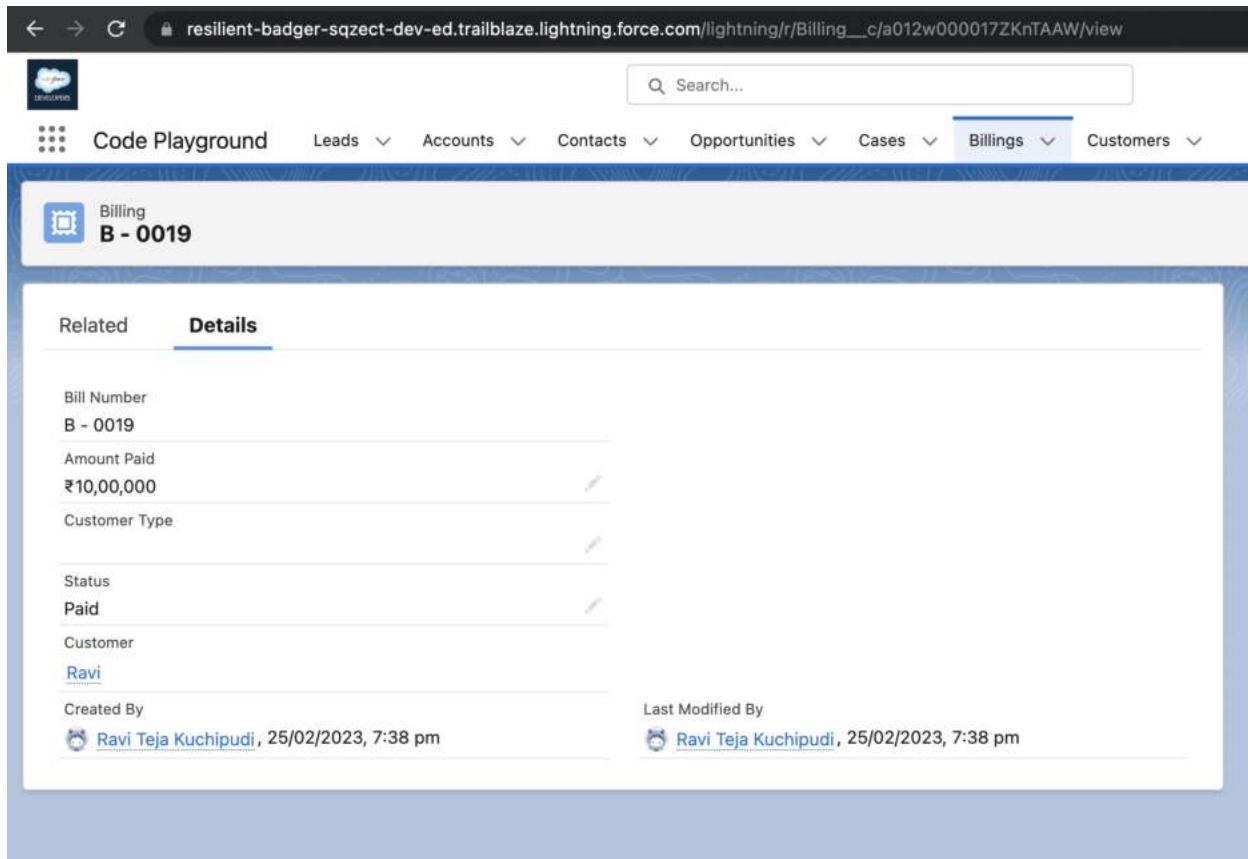


Figure 80

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.

```

1  @isTest
2  * public class CustomerTrigger {
3      @isTest
4      public static void tml(){
5          List<Customer__c> customers = new List<Customer__c>();
6          //Creating Customer Records
7          for(integer i=0; i<10; i++){
8              Customer__c c = new Customer__c ();
9              c.Name = 'Customer '+i;
10             c.Active__c = false;
11             c.Customer_Type__c = 'Premium';
12             customers.add(c);
13         }
14         insert customers;
15         //Getting Each customer and updating its active value
16         List<Customer__c> getCustomers = [select id, Active__c from Customer__c];
17         for(Customer__c c: getCustomers){
18             c.Active__c = true;
19         }
20
21         Test.startTest();
22         update getCustomers;
23         Test.stopTest();
24         //10 customer records are updated so we must see 10 billing reports
25         System.assertEquals(10, [select count() from Billing__c ]);
26     }
27 }

```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total	Overall Code Coverage						
✓	TestRun @ 7:35:32 pm			0	1	<table border="1"> <thead> <tr> <th>Class</th> <th>Percent</th> <th>Lines</th> </tr> </thead> <tbody> <tr> <td>CustomerTrigger</td> <td>100%</td> <td>11/11</td> </tr> </tbody> </table>	Class	Percent	Lines	CustomerTrigger	100%	11/11
Class	Percent	Lines										
CustomerTrigger	100%	11/11										

Figure 81

```

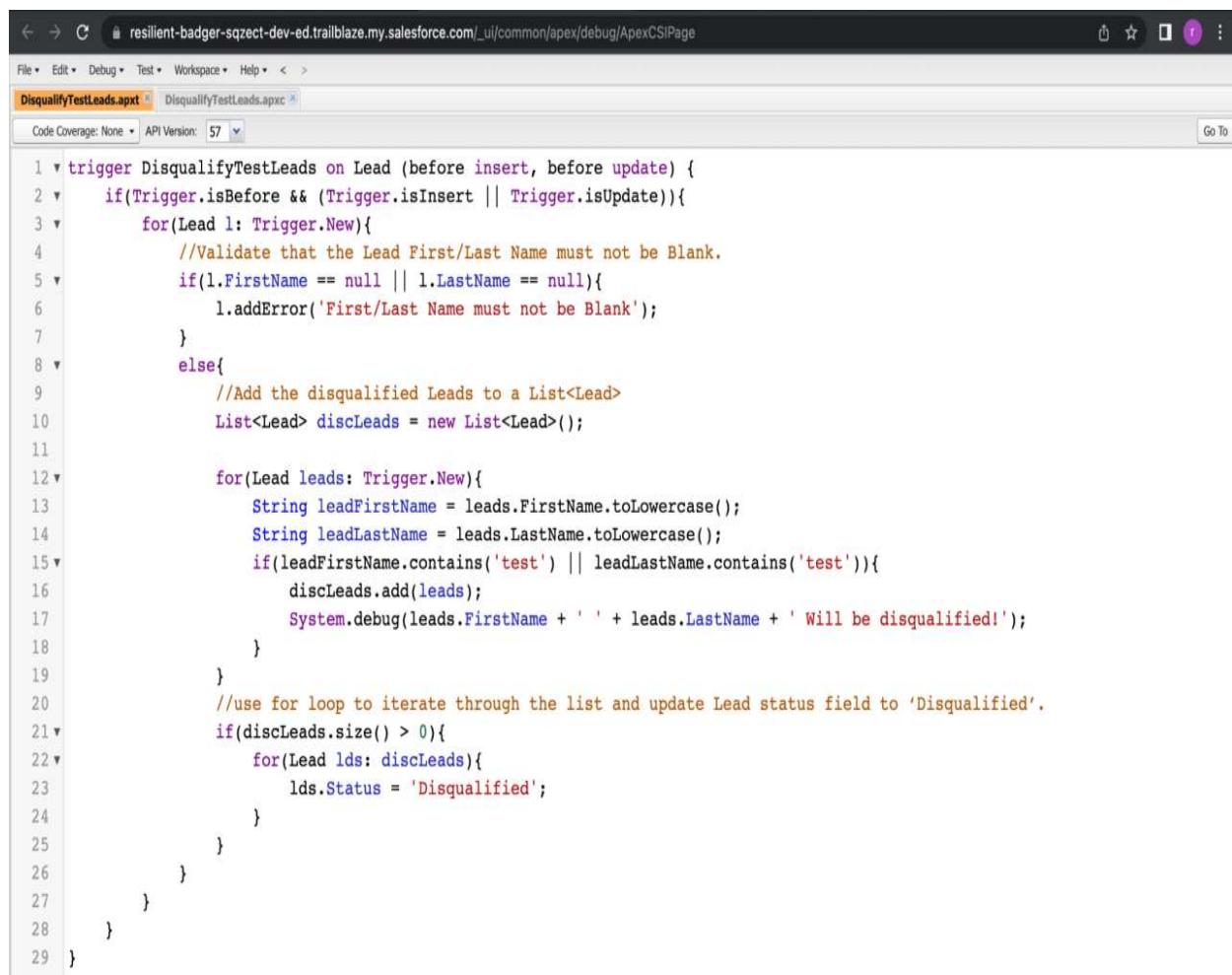
1  trigger CustomerTrigger on Customer__c (after update) {
2      if(Trigger.isAfter && Trigger.isUpdate){
3          List<Billing__c> billings = new List<Billing__c>();
4          for(Customer__c newC: Trigger.New){
5              if( Trigger.OldMap.get(newC.Id).Active__c == false && newC.Active__c == true){
6                  Billing__c billing = new Billing__c ();
7                  billing.Amount_Paid__c = 1000000 ;
8                  billing.Status__c = 'Paid';
9                  billing.Customer__c = newC.Id;
10                 billings.add(billing);
11             }
12         }
13         if(billings.size() > 0){
14             insert billings;
15         }
16     }
17 }

```

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’



The screenshot shows the Salesforce Apex code editor with the following details:

- Title Bar:** Shows the URL `resilient-badger-sqzect-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`.
- File Menu:** File, Edit, Debug, Test, Workspace, Help.
- Trigger Name:** DisqualifyTestLeads.apxc
- API Version:** 57
- Code Coverage:** None
- Code Content:**

```
trigger DisqualifyTestLeads on Lead (before insert, before update) {
    if(Trigger.isBefore && (Trigger.isInsert || Trigger.isUpdate)){
        for(Lead l: Trigger.New){
            //Validate that the Lead First/Last Name must not be Blank.
            if(l.FirstName == null || l.LastName == null){
                l.addError('First/Last Name must not be Blank');
            }
            else{
                //Add the disqualified Leads to a List<Lead>
                List<Lead> discLeads = new List<Lead>();

                for(Lead leads: Trigger.New){
                    String leadFirstName = leads.FirstName.toLowerCase();
                    String leadLastName = leads.LastName.toLowerCase();
                    if(leadFirstName.contains('test') || leadLastName.contains('test')){
                        discLeads.add(leads);
                        System.debug(leads.FirstName + ' ' + leads.LastName + ' Will be disqualified!');
                    }
                }
                //use for loop to iterate through the list and update Lead status field to 'Disqualified'.
                if(discLeads.size() > 0){
                    for(Lead lds: discLeads){
                        lds.Status = 'Disqualified';
                    }
                }
            }
        }
    }
}
```

Figure 83

The screenshot shows a 'New Lead' form in the Salesforce Lightning interface. The 'Name' field is highlighted with a yellow background. An error message 'We hit a snag.' is displayed in a red box above the save buttons, with the sub-error 'First/Last Name must not be Blank'. The 'Lead Status' field is set to 'Open - Not Contacted'.

Field	Value
Name	Teja
Salutation	Mr.
First Name	
Last Name	Teja
Company	RT Group
Title	
Lead Source	--None--
Industry	--None--
Phone	
Mobile	
Fax	
Email	
Website	
Lead Status	Open - Not Contacted

Figure 84

The screenshot shows a 'New Lead' form in the Salesforce Lightning interface. The 'First Name' field is highlighted with a green background, indicating it is valid. The 'Lead Status' field is set to 'Open - Not Contacted'.

Field	Value
Name	tEst
Salutation	--None--
First Name	tEst
Last Name	Teja
Company	RT Group
Title	
Lead Source	--None--
Industry	--None--
Phone	
Mobile	
Fax	
Email	
Website	
Lead Status	Open - Not Contacted

Figure 85

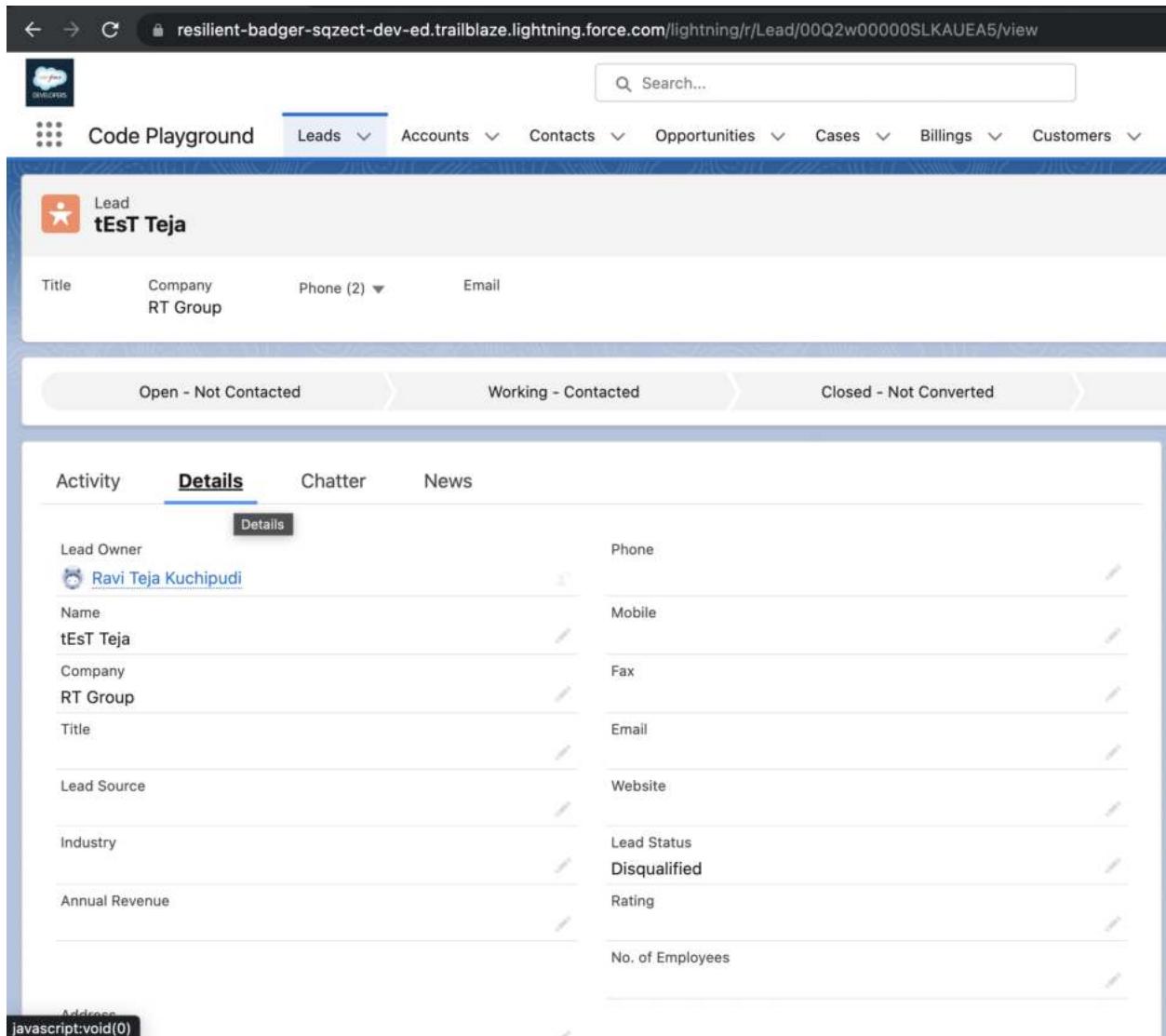


Figure 86

Exercise 15: Write a Test Class for DisqualifyTestLeads.

```

1  @isTest
2  *public class DisqualifyTestLeads {
3      @isTest
4      public static void tm1(){
5          List<Lead> leads = new List<Lead>();
6          for(Integer i=0; i<10; i++){
7              Lead l = new Lead();
8              l.FirstName = 'Test';
9              l.LastName = ' ' + i;
10             l.Company = 'Test Company ' + i;
11             leads.add(l);
12         }
13         Test.startTest();
14         insert leads;
15         Test.stopTest();
16     }
17     System.assertEquals(10, [select count() from Lead where Status = 'Disqualified']);
18 }
19
20 @isTest
21 *public static void tm2(){
22     Lead l = new Lead();
23     l.LastName = 'Teja';
24     l.Company = 'Test Company ';
25
26     Test.startTest();
27     try{
28         insert l;
29     }catch(Exception e){
30         System.debug(e.getMessage());
31     }
32     Test.stopTest();
33 }
34 }
```

Overall Code Coverage

Class	Lines	Percent
DiscountOperations	0/16	0%
DiscountClass	0/4	0%
DisqualifyTestLeads	13/13	100%

Figure 87

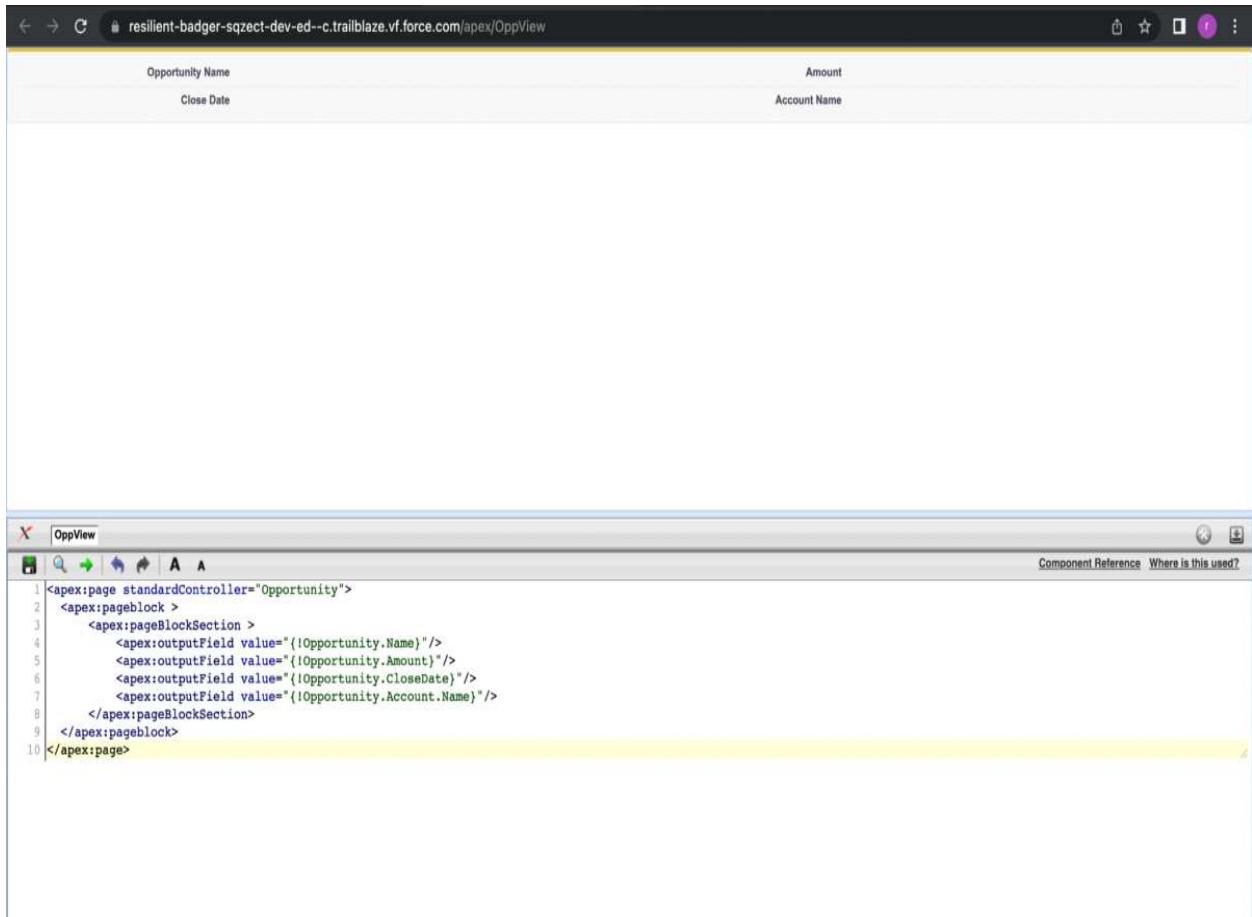
```

1 *trigger DisqualifyTestLeads on Lead (before insert, before update) {
2     if(Trigger.isBefore && (Trigger.isInsert || Trigger.isUpdate)){
3         for(Lead l: Trigger.New){
4             //Validate that the Lead First/Last Name must not be Blank.
5             if(l.FirstName == null || l.LastName == null){
6                 l.addError('First/Last Name must not be Blank');
7             }
8             else{
9                 //Add the disqualified Leads to a List<Lead>
10                List<Lead> discLeads = new List<Lead>();
11
12                for(Lead leads: Trigger.New){
13                    String leadFirstName = leads.FirstName.toLowerCase();
14                    String leadLastName = leads.LastName.toLowerCase();
15                    if(leadFirstName.contains('test') || leadLastName.contains('test')){
16                        discLeads.add(leads);
17                        System.debug(leads.FirstName + ' ' + leads.LastName + ' Will be disqualified!');
18                    }
19                }
20                //use for loop to iterate through the list and update Lead status field to 'Disqualified'.
21                if(discLeads.size() > 0){
22                    for(Lead lds: discLeads){
23                        lds.Status = 'Disqualified';
24                    }
25                }
26            }
27        }
28    }
29 }
```

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.



The screenshot shows the Salesforce Visualforce editor interface. At the top, there's a browser-like header with the URL "resilient-badger-sqzect-dev-ed--c.trailblaze.vf.force.com/apex/OppView". Below the header is the visual representation of the page, which consists of a single row with four columns. The first column is labeled "Opportunity Name", the second "Close Date", the third "Amount", and the fourth "Account Name". The visual representation is a simple table with a light gray background and thin borders. Below the visual representation is the code editor window. The title bar of the code editor says "OppView". The code itself is as follows:

```
1 <apex:page standardController="Opportunity">
2   <apex:pageblock >
3     <apex:pageBlockSection >
4       <apex:outputField value="{!Opportunity.Name}" />
5       <apex:outputField value="{!Opportunity.Amount}" />
6       <apex:outputField value="{!Opportunity.CloseDate}" />
7       <apex:outputField value="{!Opportunity.Account.Name}" />
8     </apex:pageBlockSection>
9   </apex:pageblock>
10 </apex:page>
```

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 HTML list tag
7. Use the apex:outputLink component to link to the respective record detail page

The screenshot shows a browser window and a code editor side-by-side.

Browser View: The browser URL is `resilient-badger-sqzect-dev-ed--c.trailblaze.vf.force.com/apex/AccountList`. The page displays a list of account names as hyperlinks:

- Burlington Textiles Corp of America
- Dickenson plc
- Edge Communications
- Express Logistics and Transport
- GenePoint
- Grand Hotels & Resorts Ltd
- Pyramid Construction Inc.
- Sample Account for Entitlements
- sForce
- United Oil & Gas Corp.
- United Oil & Gas, Singapore
- United Oil & Gas, UK
- University of Arizona

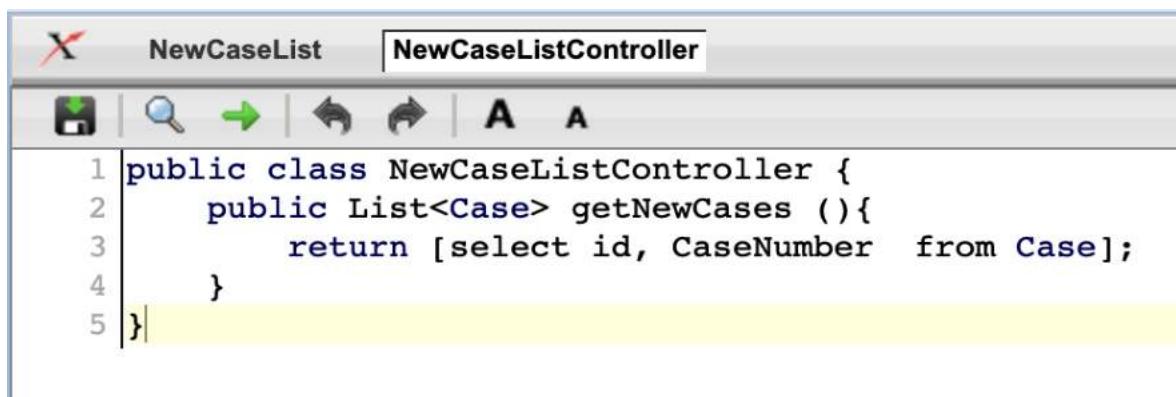
Code Editor: The code editor shows the Visualforce page source code:

```
1<apex:page standardController="Account" recordSetVar="accounts">
2    <apex:pageBlock>
3        <apex:repeat var="a" value="{!accounts}">
4            <li>
5                <apex:outputLink value="/{!a.id}">{!a.Name}</apex:outputLink>
6            </li>
7        </apex:repeat>
8    </apex:pageBlock>
9</apex: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.\



The screenshot shows a code editor window with the title bar "NewCaseList" and the tab "NewCaseListController". The editor has a toolbar with icons for save, search, and navigation. The code area contains the following Apex class:

```
1 public class NewCaseListController {  
2     public List<Case> getNewCases (){  
3         return [select id, CaseNumber from Case];  
4     }  
5 }
```

Figure 91

The figure consists of two vertically stacked screenshots. The top screenshot shows a browser window with the URL `resilient-badger-sqzect-dev-ed--c.trailblaze.vf.force.com/apex/NewCaseList`. The page displays a list of 20 case IDs, each preceded by a bullet point and underlined:

- 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

The bottom screenshot shows the Visualforce page editor for the file `NewCaseList`. The code is displayed in a text editor with syntax highlighting for Apex and Visualforce tags:

```
1 <apex:page controller="NewCaseListController" >
2   <apex:repeat var="case" value="{!newCases}">
3     <li><apex:outputLink value="/{!case.id}">{!case.CaseNumber}</apex:outputLink></li>
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](#)