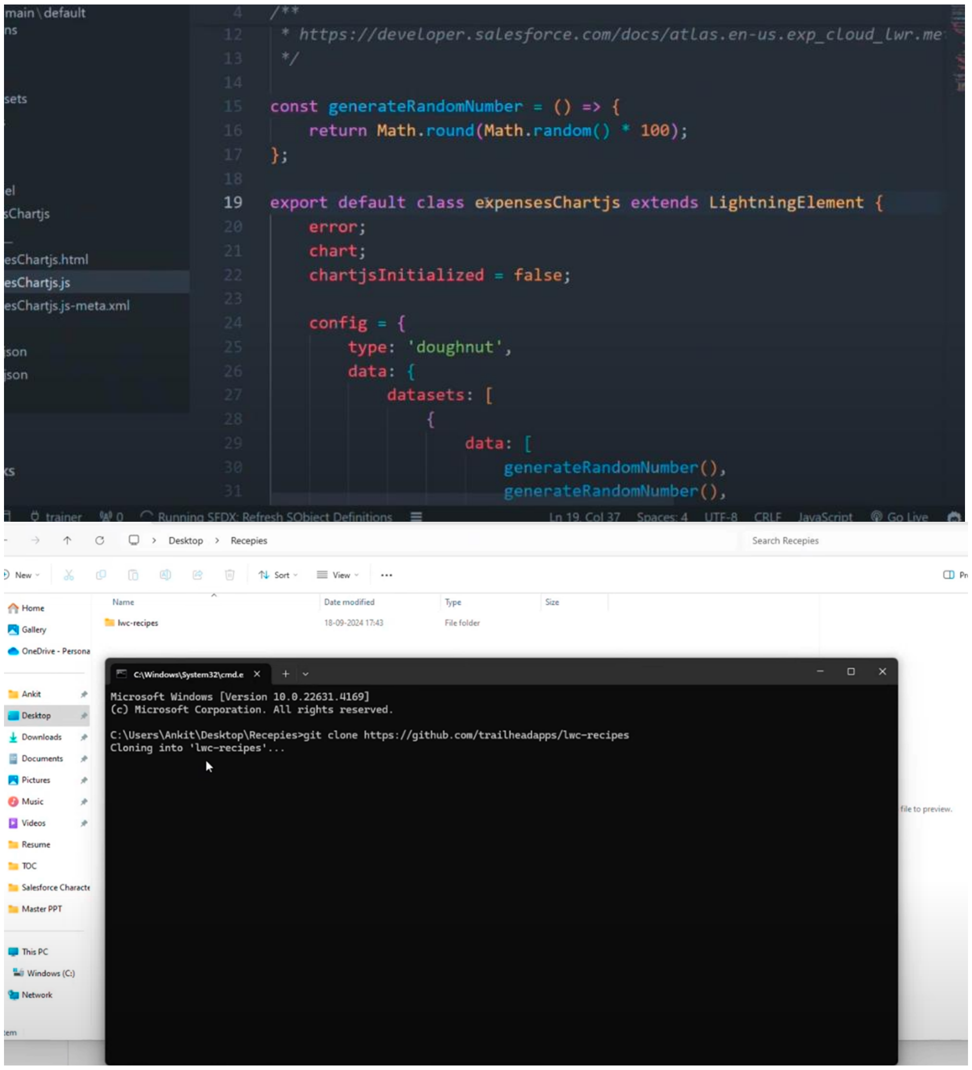
Expense Tracker Project (Phase 5: Apex Programming)

Goal

The goal of this project is to build a **Salesforce-based Expense Tracker** that automates the management and approval of employee expenses. Custom logic using Apex enhances system functionality, enabling automatic record creation, advanced calculations, and dynamic reporting. This reduces manual effort, ensures accuracy, and improves overall expense management efficiency.

Problem Statement

Organizations face challenges in managing and approving employee expenses due to manual processes, leading to delays, errors, and lack of real-time visibility. A Salesforce-based Expense Tracker with custom Apex programming automates record handling, calculations, and reporting, ensuring faster approvals, accurate reimbursements, and better control over organizational spending.

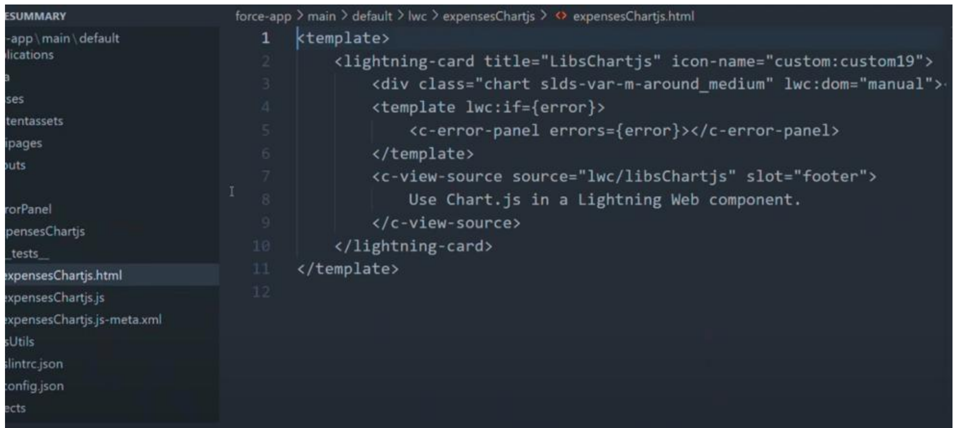


```
/**
 * https://developer.salesforce.com/docs/atlas.en-us.exp_cloud_lwr.me
 */

const generateRandomNumber = () => {
    return Math.round(Math.random() * 100);
};

export default class expensesChartjs extends LightningElement {
    error;
    chart;
    chartjsInitialized = false;

    config = {
        type: 'doughnut',
        data: {
            datasets: [
                {
                    data: [
                        generateRandomNumber(),
                        generateRandomNumber(),
```

```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Ankit\Desktop\Recepies>git clone https://github.com/trailheadapps/lwc-recipes
Cloning into 'lwc-recipes'...
```

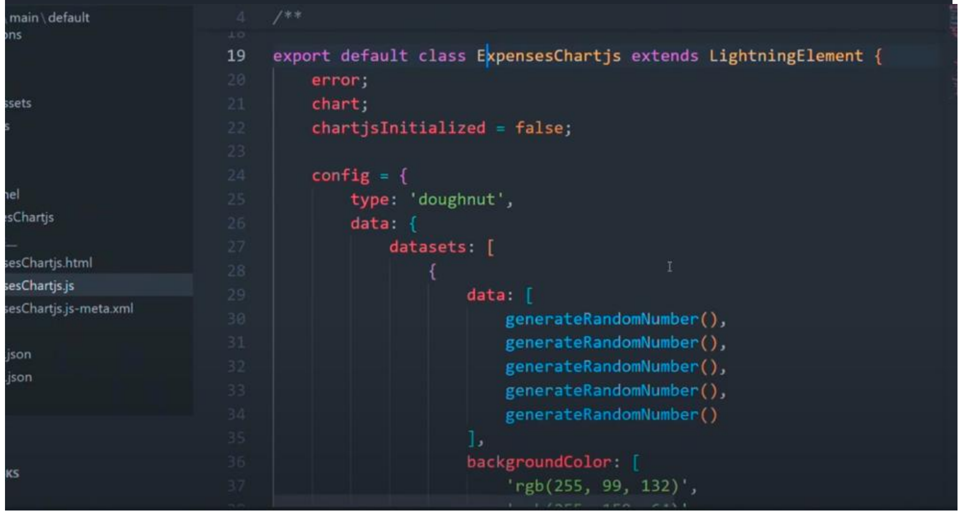
Phase 5: Apex Programming (Developer)

Phase 5 focuses on **Apex Programming** to introduce custom logic and advanced functionality. Custom **Apex Triggers, Classes, and SOQL queries** were developed to automate business processes for the Expense Tracker Project.



SUMMARY
force-app > main > default > lwc > expensesChartjs > expensesChartjs.html
expensesChartjs.html
expensesChartjs.js
expensesChartjs.js-meta.xml

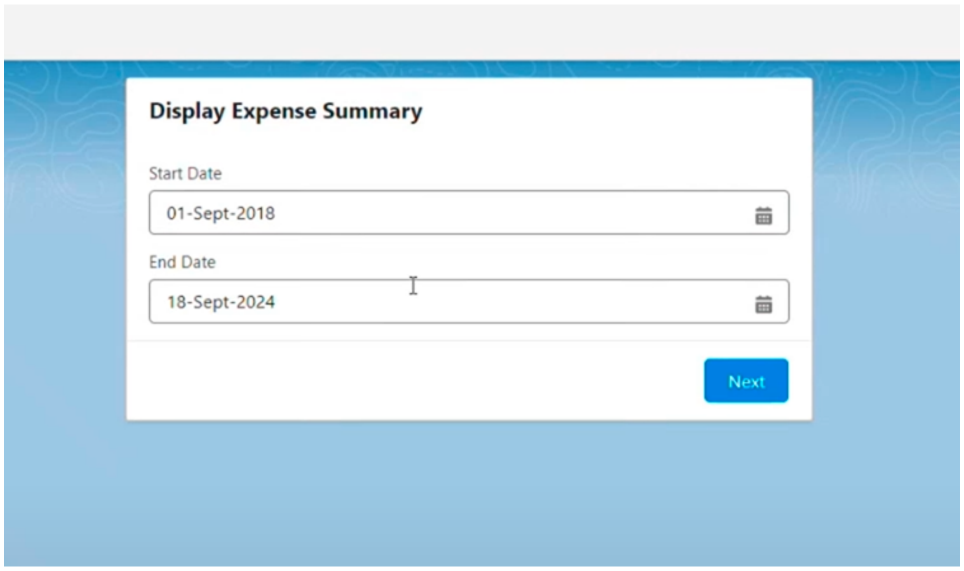
```html
<template>
    <lightning-card title="LibsChartjs" icon-name="custom:custom19">
        <div class="chart slds-var-m-around_medium" lwc:dom="manual">
        <template lwc:if={error}>
            <c-error-panel errors={error}></c-error-panel>
        </template>
        <c-view-source source="lwc/libsChartjs" slot="footer">
            Use Chart.js in a Lightning Web component.
        </c-view-source>
    </lightning-card>
</template>
```

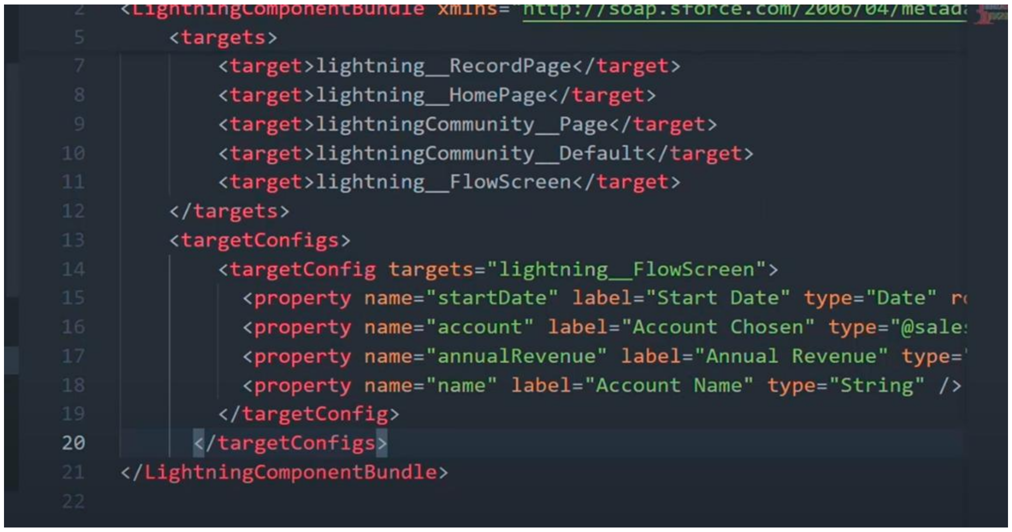


```javascript
/**

export default class ExpensesChartjs extends LightningElement {
    error;
    chart;
    chartjsInitialized = false;

    config = {
        type: 'doughnut',
        data: {
            datasets: [
                {
                    data: [
                        generateRandomNumber(),
                        generateRandomNumber(),
                        generateRandomNumber(),
                        generateRandomNumber(),
                        generateRandomNumber()
                    ],
                    backgroundColor: [
                        'rgb(255, 99, 132)',
```

Apex Triggers

- Developed a trigger to automatically create an **Approval record** whenever a new high-value expense is submitted.
- Ensures that all expenses requiring managerial approval are consistently tracked without manual intervention.

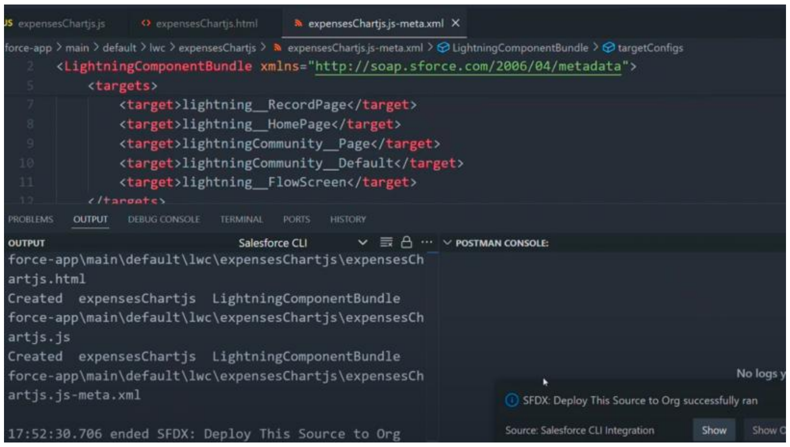




```
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metada
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
        <target>lightningCommunity__Page</target>
        <target>lightningCommunity__Default</target>
        <target>lightning__FlowScreen</target>
    </targets>
    <targetConfigs>
        <targetConfig targets="lightning__FlowScreen">
          <property name="startDate" label="Start Date" type="Date" r
          <property name="account" label="Account Chosen" type="@sale
          <property name="annualRevenue" label="Annual Revenue" type="
          <property name="name" label="Account Name" type="String" />
        </targetConfig>
      </targetConfigs>
</LightningComponentBundle>
```

SOQL Queries

- Designed SOQL queries to fetch **employee expense reports**.
- Generated **total expense summaries** by employee, department, and category.


expensesChartjs.js
expensesChartjs.html
expensesChartjs.js-meta.xml
force-app > main > default > lwc > expensesChartjs > expensesChartjs.js-meta.xml > LightningComponentBundle > targetConfigs
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
HISTORY
OUTPUT
Salesforce CLI
POSTMAN CONSOLE:
No logs y
Show
Show C

```
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__HomePage</target>
        <target>lightningCommunity__Page</target>
        <target>lightningCommunity__Default</target>
        <target>lightning__FlowScreen</target>
    </targets>
```

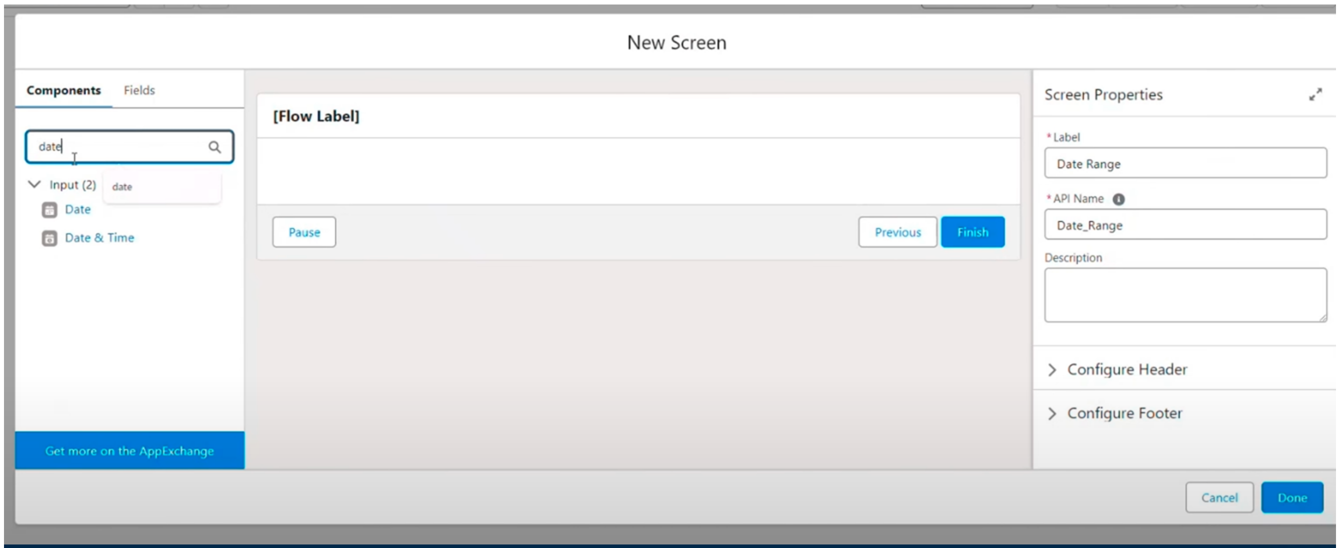
```
force-app\main\default\lwc\expensesChartjs\expensesCh
artjs.html
Created  expensesChartjs  LightningComponentBundle
force-app\main\default\lwc\expensesChartjs\expensesCh
artjs.js
Created  expensesChartjs  LightningComponentBundle
force-app\main\default\lwc\expensesChartjs\expensesCh
artjs.js-meta.xml

17:52:30.706 ended SFDX: Deploy This Source to Org
```

SFDX: Deploy This Source to Org successfully ran

Source: Salesforce CLI Integration

Test Classes

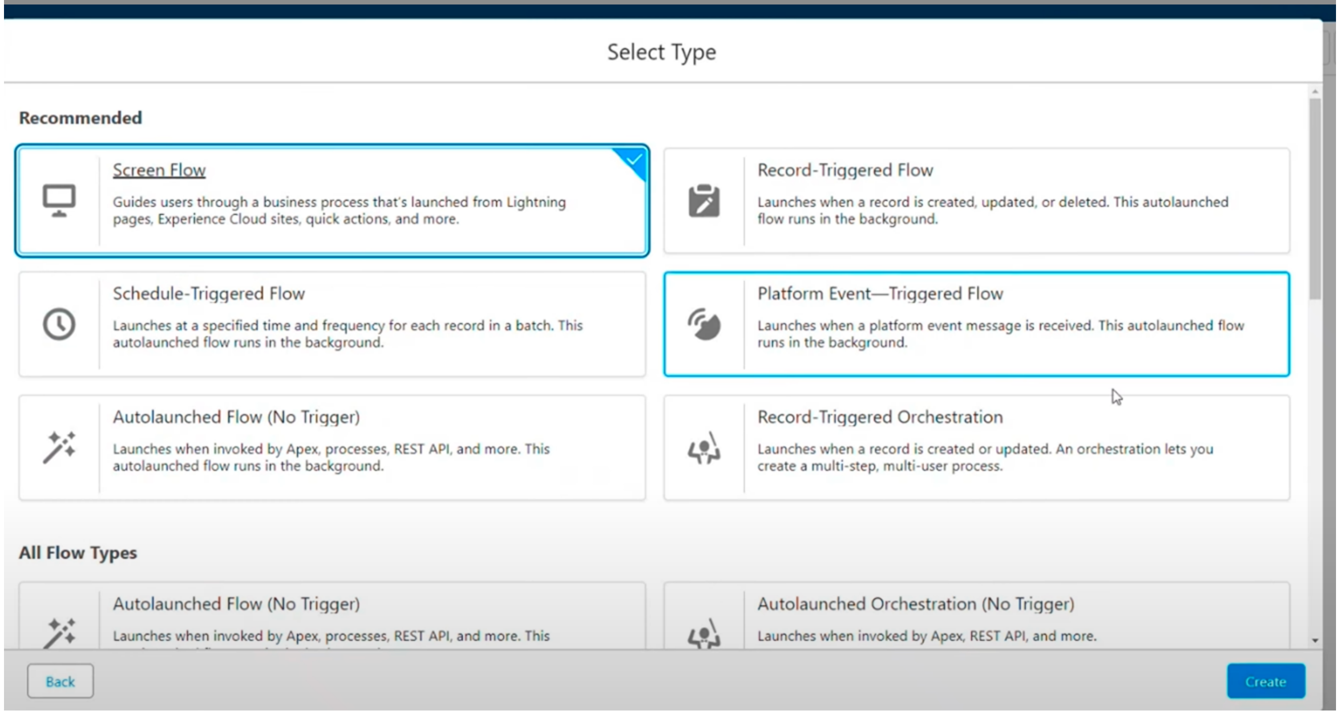
- Added test classes to validate Apex logic and ensure **100% code coverage**.
- Improved system reliability and supported successful deployments.
- Maintained compliance with Salesforce best practices.



*Label: Date Range

*API Name: Date_Range

Description:



Screen Flow
Guides users through a business process that's launched from Lightning pages, Experience Cloud sites, quick actions, and more.

Record-Triggered Flow
Launches when a record is created, updated, or deleted. This autolaunched flow runs in the background.

Schedule-Triggered Flow
Launches at a specified time and frequency for each record in a batch. This autolaunched flow runs in the background.

Platform Event—Triggered Flow
Launches when a platform event message is received. This autolaunched flow runs in the background.

Autolaunched Flow (No Trigger)
Launches when invoked by Apex, processes, REST API, and more. This autolaunched flow runs in the background.

Record-Triggered Orchestration
Launches when a record is created or updated. An orchestration lets you create a multi-step, multi-user process.

Autolaunched Flow (No Trigger)
Launches when invoked by Apex, processes, REST API, and more. This

Autolaunched Orchestration (No Trigger)
Launches when invoked by Apex, REST API, and more.

Conclusion

Phase 5 successfully implemented **Apex Programming** in the Expense Tracker Project. Through triggers, classes, and SOQL queries with robust test coverage, the system achieved advanced automation and custom logic. This ensured **accurate expense tracking, streamlined approvals, and comprehensive reporting**, enhancing the overall functionality and reliability of the Expense Tracker.