Phase 6: User Interface Development

PROJECT TITLE:-

Expense On a Page: An expense approval & insight system.
Industry: Finance / Corporate Expense Management.
Target User: Employees, Managers, and Finance Teams.

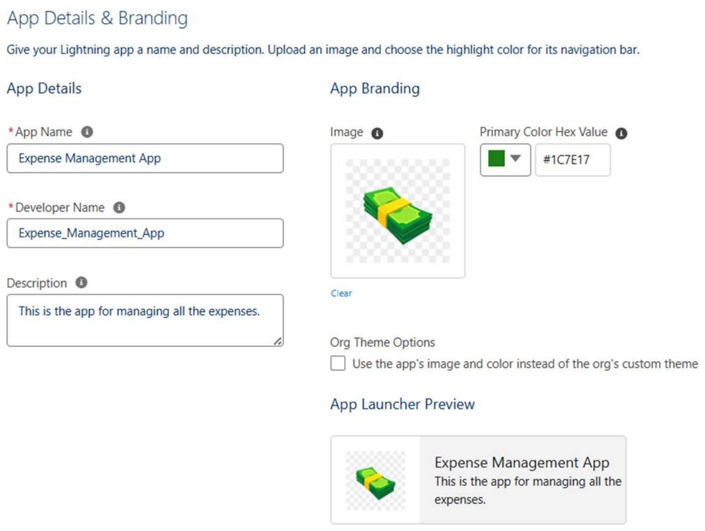
LIGHTNING APP BUILDER:-

Building an App for *the Expense Management* accessible *by the Manager, Employee & Finance Team* so they can have a *single place to work* with all of the object *Expense and Expense_Line Object*.

Creating a New Lightning App:

App Name: Expense Management App
Developer Name: Expense_Management_App
Description: This is the app for managing all the expenses.
Image:

Colour: #1C7E17
Letting the App Option as Default
Letting utility Item to be default for now as well
Navigation Items: Adding Expense, Expense_Lines, Reports
User Profile: System Administrator, Standard Platform User, Standard User



App Details & Branding
Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

*App Name: Expense Management App
*Developer Name: Expense_Management_App
Description: This is the app for managing all the expenses.

App Branding

Image
Clear
Primary Color Hex Value: 🟩 #1C7E17

Org Theme Options
☐ Use the app's image and color instead of the org's custom theme

App Launcher Preview
Expense Management App
This is the app for managing all the expenses.

RECORD PAGES:-

Creating a Custom Record Page for the Expense object so that *it would be easily accessible and will be according the Stakeholder demand*.

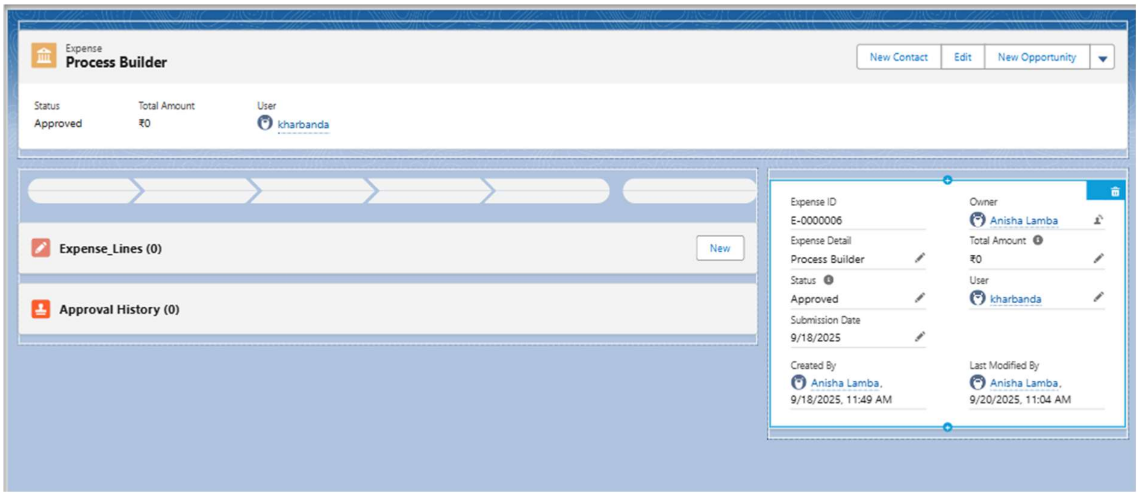
Creating a custom Record Page:

Creating a new Lightning page: Record Page
Label: Expense Record Page
Object: Expense
Choose Page Template: Header and Right Sidebar

Header: Adding Highlight Panel at the top of the page
Left-Side Bar: Adding a Related Lists at the left hand of the Page
Right-Side Bar: Adding Records Detail at the

Path: Adding path at the top of the left-side bar
- Set Up Path
- Path Name- Expense Status
- API Reference Name- Expense_Status
- Object- Expense
- Record Type- Master
- Picklist- Status
- Activate Your path- Enabled

Saving and Activating it:



Expense
Process Builder

Status: Approved
Total Amount: ₹0
User: kharbanda

Expense_Lines (0)

Approval History (0)

Expense ID: E-0000006
Owner: Anisha Lamba
Expense Detail: Process Builder
Total Amount: ₹0
Status: Approved
User: kharbanda
Submission Date: 9/18/2025
Created By: Anisha Lamba, 9/18/2025, 11:49 AM
Last Modified By: Anisha Lamba, 9/20/2025, 11:04 AM

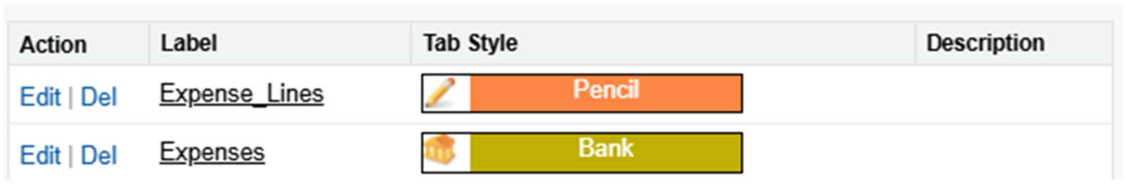
Adding the custom Components and dynamic fields later in the phase.

<u>TABS:-</u>

The Tabs for Custom Object Expense and Expense_Line were created while making the Object and have added them to the app navigation.

Expense: Pencil

Expense_Line: Bank

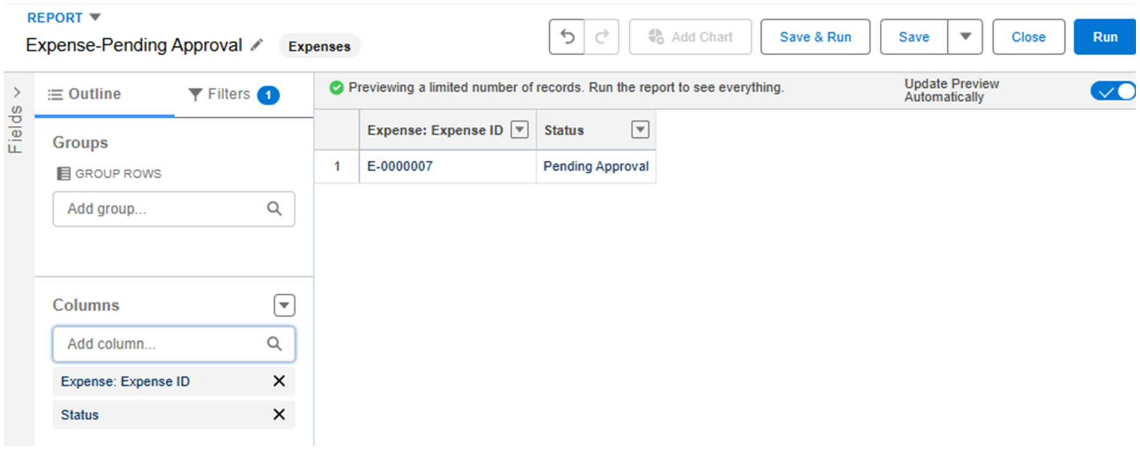


Action	Label	Tab Style	Description
Edit | Del	Expense_Lines	Pencil	
Edit | Del	Expenses	Bank	

<u>HOME PAGE LAYOUTS:</u>

In order to make a Home Page Layout we have to do some Pre-work so let's do that.

1. ***Creating reports-***
   - Creating a report with Pending Approvals of all Expense (For Managers):
     Report Object: Expense.
     Filter: Show Me- All expenses
     Submission Date- All Time
     Status- equals Pending Approval
     Columns: Status
     Report Name: Expense-Pending Approval
     Report Unique Name: ExpensePending_Approval
     Report Description: A report that will be showing all of the expenses with Pending Approval
     Folder: New Folder
     Folder Name- Finance Folder
     Folder Unique Name- FinanceFolder

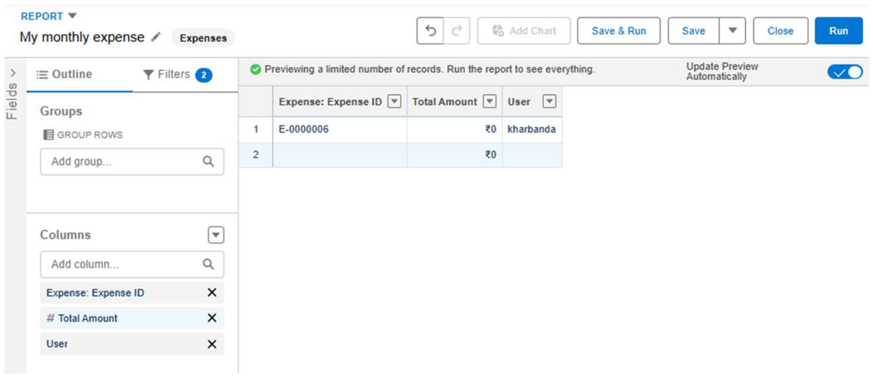


Previewing a limited number of records. Run the report to see everything.

	Expense: Expense ID	Status
1	E-0000007	Pending Approval

Columns: Expense: Expense ID, Status

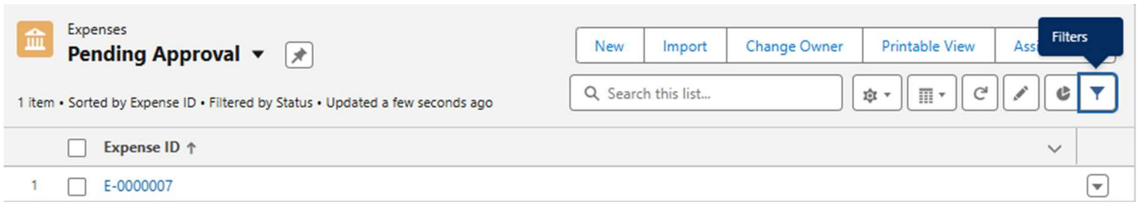
- Creating a report for the employee monthly spending (For Employee):
  Report Object: Expense.
  Filter: Show Me- All expenses
  Submission Date- This Month
  User- equals kharbanda
  Columns: Adding Total Amount
  Report Name: My monthly expense
  Report Unique Name: My_monthly_expense
  Report Description: A report that will be showing all of the expenses of the current month
  Folder: Finance Folder



Previewing a limited number of records. Run the report to see everything.

	Expense: Expense ID	Total Amount	User
1	E-0000006	₹0	kharbanda
2		₹0	

2. ***Creating list views on Expense Object-***
   - *Pending Approval:*
     List Name: Pending Approval
     List API Name: Pending_Approval
     Who sees this list view: Share list view with groups of user: Manager
     Filter: Fields- Status
     Operator- Equals
     Value- Pending Approval

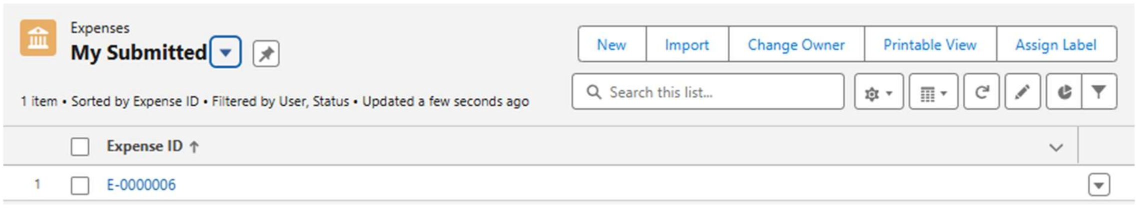


1 item • Sorted by Expense ID • Filtered by Status • Updated a few seconds ago

	Expense ID
1	E-0000007

   - *My Submitted:*
     List Name: Pending Approval
     List API Name: Pending_Approval
     Who sees this list view: All user can
     Filter: Fields- User
     Operator- Equals
     Value- Kharbanda

Fields- Status
Operator- contains
Value- Submitted, Approved, Rejected, Pending Approval



1 item • Sorted by Expense ID • Filtered by User, Status • Updated a few seconds ago

	Expense ID
1	E-0000006

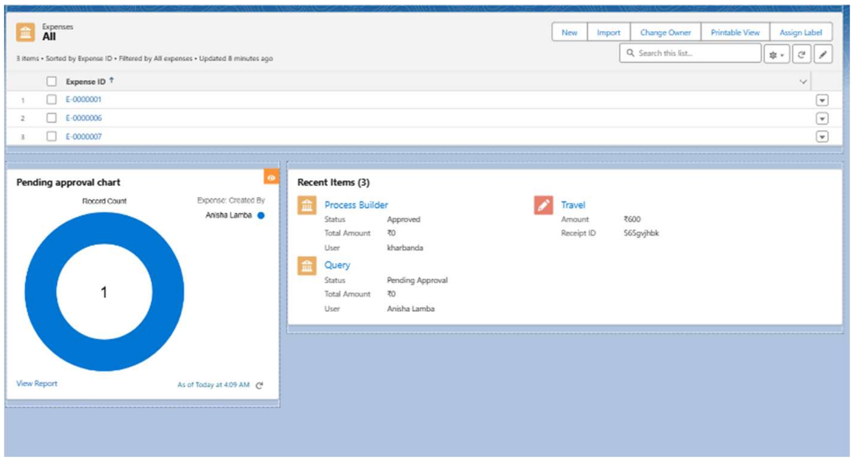
Making the Home Page Custom Layout now
Creating a new Lightning page: Home Page
Label: Expense Manager Page
Choose Page Template: Home page header two columns left side bar

Report Chart: Label- Pending approval Chart
Report- Expense-Pending Approval
Filter By- User> Role> Name Equal Manager

List View: Object- Expense
Filter- Pending Approval
Number of Records to Display- 10
Enable Inline edit

Recent Items: Label- Recent Items
Objects- Expense, Expense_Line
Number of Records to Display- 3

Saving and Activating the custom Home Page only for the Expense Manager App



3 items • Sorted by Expense ID • Filtered by All expenses • Updated 8 minutes ago

	Expense ID
1	E-0000001
2	E-0000006
3	E-0000007

Process Builder
Status Approved
Total Amount ₹0
User kharbanda

Query
Status Pending Approval
Total Amount ₹0
User Anisha Lamba

Travel
Amount ₹600
Receipt ID 565gvjhbk

Adding the custom component on the field later on the phase

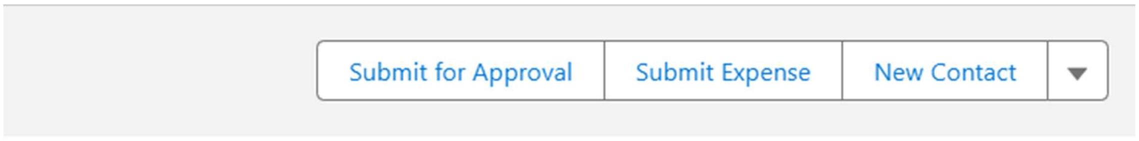
<u>UTILITY BAR:-</u>

1. Making a Quick Utility Bar to *Submit Expense for Approval-*

   Object Manager-> Expense__c-> Buttons, Links, and Actions-> New Action
   Action Type: Update Record
   Label: Submit Expense
   Name: Submit_Expense
   Description: Will be used for submitting the record for approval
   Success Message: Your record has been submitted
   Save

   Predefined Field Value:
   Field Name: Status
   Specific Value: Submitted
   Save

   Adding it to the page layout



<u>LWC (LIGHTNING WEB COMPONENTS):-</u>

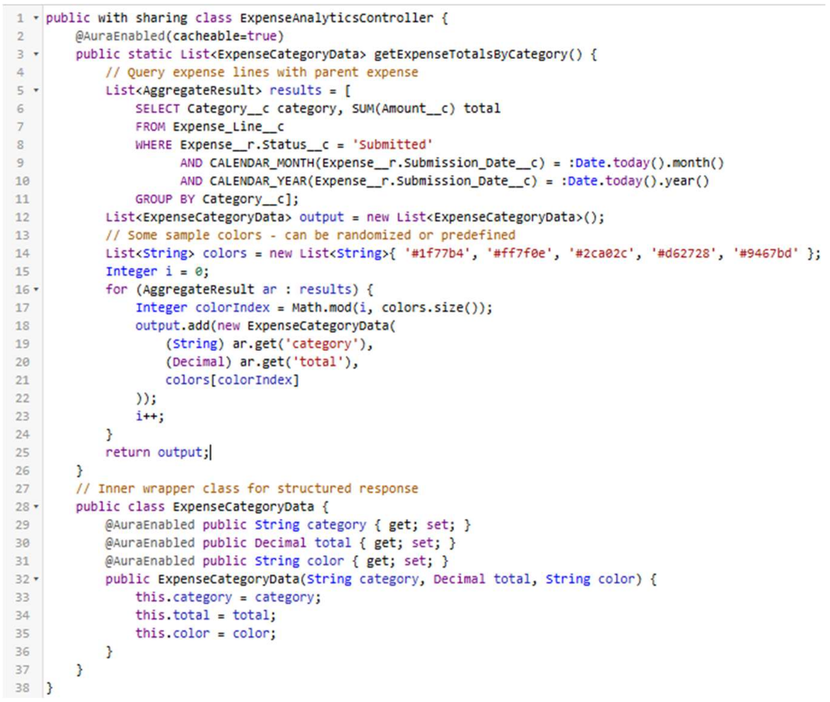
Making a custom Lightning Web Component for Employee and Manager to see the monthly expense on the basis of category
The object we used are : Expesne__c and Expense_line__c

Making an Apex Controller class:

Code:

```
public with sharing class ExpenseAnalyticsController {
   @AuraEnabled(cacheable=true)

 public static List<ExpenseCategoryData> getExpenseTotalsByCategory() {
List<AggregateResult> results
          SELECT Category__c category, SUM(Amount__c) total
          FROM Expense_Line__c
          WHERE Expense__r.Status__c = 'Submitted'
```

```
        AND CALENDAR_MONTH(Expense__r.Submission_Date__c) = :Date.today().month()
        AND CALENDAR_YEAR(Expense__r.Submission_Date__c) = :Date.today().year()

 GROUP BY Category__c];
List<ExpenseCategoryData> output = new List<ExpenseCategoryData>();
List<String> colors = new List<String>{ '#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd'};
integer i = 0;
for (AggregateResult ar : results) {
          Integer colorIndex = Math.mod(i, colors.size());
          output.add(new ExpenseCategoryData(
          (String) ar.get('category'),
          (Decimal) ar.get('total'),
          colors[colorIndex]));
        i++;}
    return output;}
// Inner wrapper class for structured response
     public class ExpenseCategoryData {
    @AuraEnabled public String category { get; set; }
    @AuraEnabled public Decimal total { get; set; }
    @AuraEnabled public String color { get; set; }
public ExpenseCategoryData(String category, Decimal total, String color) {
        this.category = category;
        this.total = total;
        this.color = color;
        }
    }
}
```

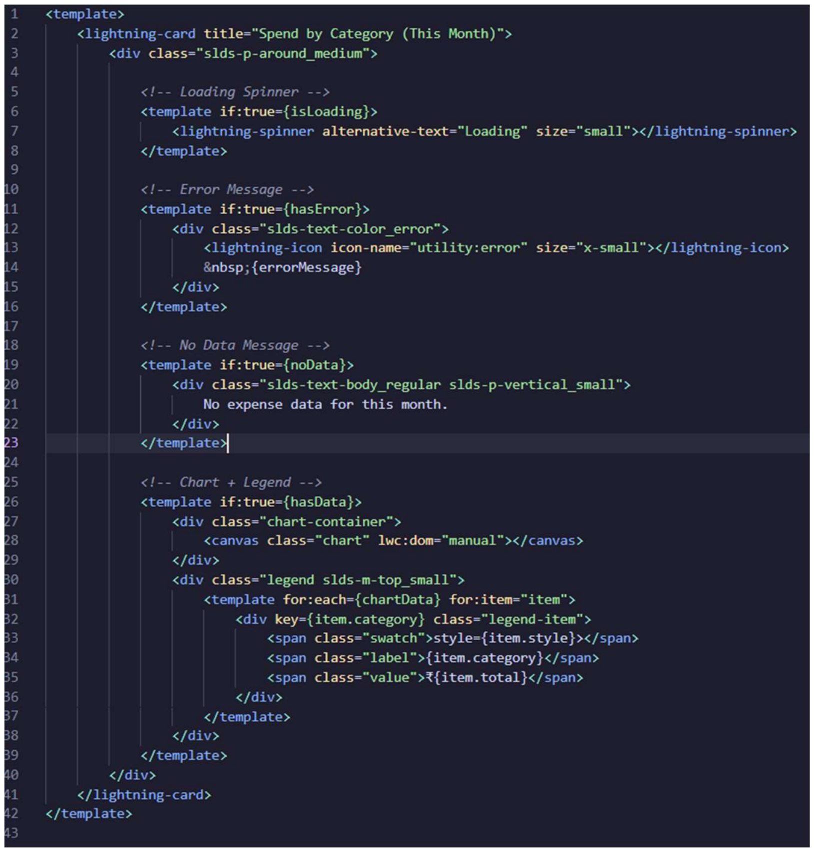


```
public with sharing class ExpenseAnalyticsController {
    @AuraEnabled(cacheable=true)
    public static List<ExpenseCategoryData> getExpenseTotalsByCategory() {
        // Query expense lines with parent expense
        List<AggregateResult> results = [
            SELECT Category__c category, SUM(Amount__c) total
            FROM Expense_Line__c
            WHERE Expense__r.Status__c = 'Submitted'
                  AND CALENDAR_MONTH(Expense__r.Submission_Date__c) = :Date.today().month()
                  AND CALENDAR_YEAR(Expense__r.Submission_Date__c) = :Date.today().year()
            GROUP BY Category__c];
        List<ExpenseCategoryData> output = new List<ExpenseCategoryData>();
        // Some sample colors - can be randomized or predefined
        List<String> colors = new List<String>{ '#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd' };
        Integer i = 0;
        for (AggregateResult ar : results) {
            Integer colorIndex = Math.mod(i, colors.size());
            output.add(new ExpenseCategoryData(
                (String) ar.get('category'),
                (Decimal) ar.get('total'),
                colors[colorIndex]
            ));
            i++;
        }
        return output;
    }
    // Inner wrapper class for structured response
    public class ExpenseCategoryData {
        @AuraEnabled public String category { get; set; }
        @AuraEnabled public Decimal total { get; set; }
        @AuraEnabled public String color { get; set; }
        public ExpenseCategoryData(String category, Decimal total, String color) {
            this.category = category;
            this.total = total;
            this.color = color;
        }
    }
}
```

LWC Component

HTML Code:

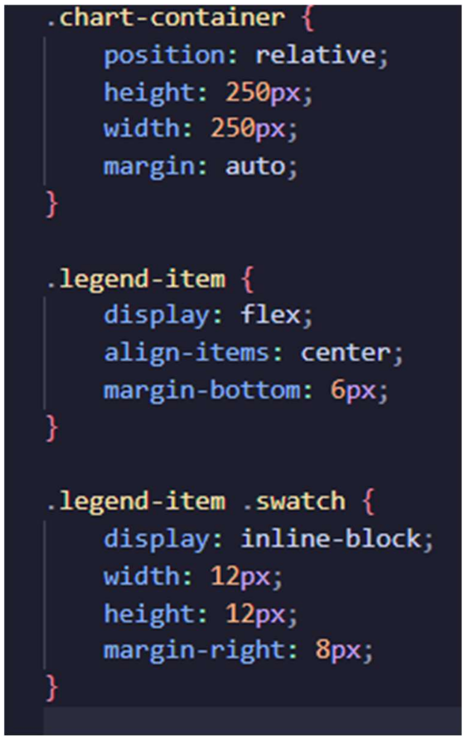
```
<template>
    <lightning-card title="Spend by Category (This Month)">
     <div class="slds-p-around_medium">
       <!-- Loading Spinner -->
            <template if:true={isLoading}>
            <lightning-spinner alternative-text="Loading" size="small"></lightning-spinner>
            </template>
       <!-- Error Message -->
            <template if:true={hasError}>
            <div class="slds-text-color_error">
            <lightning-icon icon-name="utility:error" size="x-small"></lightning-icon>
            &nbsp;{errorMessage}
            </div>
           </template>
     <!-- No Data Message -->
          <template if:true={noData}>
          <div class="slds-text-body_regular slds-p-vertical_small">
                      No expense data for this month.
          </div>
         </template>
   <!-- Chart + Legend -->
         <template if:true={hasData}>
         <div class="chart-container">
        <canvas class="chart" lwc:dom="manual"></canvas>
         </div>
             <div class="legend slds-m-top_small">
             <template for:each={chartData} for:item="item">
             <div key={item.category} class="legend-item">
            <span class="swatch">style={item.style}></span>
            <span class="label">{item.category}</span>
           <span class="value">₹{item.total}</span>
            </div>
           </template>
           </div>
          </template>
          </div>
         </lightning-card>
        </template>
```



```html
<template>
    <lightning-card title="Spend by Category (This Month)">
        <div class="slds-p-around_medium">

            <!-- Loading Spinner -->
            <template if:true={isLoading}>
                <lightning-spinner alternative-text="Loading" size="small"></lightning-spinner>
            </template>

            <!-- Error Message -->
            <template if:true={hasError}>
                <div class="slds-text-color_error">
                    <lightning-icon icon-name="utility:error" size="x-small"></lightning-icon>
                    &nbsp;{errorMessage}
                </div>
            </template>

            <!-- No Data Message -->
            <template if:true={noData}>
                <div class="slds-text-body_regular slds-p-vertical_small">
                    No expense data for this month.
                </div>
            </template>

            <!-- Chart + Legend -->
            <template if:true={hasData}>
                <div class="chart-container">
                    <canvas class="chart" lwc:dom="manual"></canvas>
                </div>
                <div class="legend slds-m-top_small">
                    <template for:each={chartData} for:item="item">
                        <div key={item.category} class="legend-item">
                            <span class="swatch">style={item.style}></span>
                            <span class="label">{item.category}</span>
                            <span class="value">₹{item.total}</span>
                        </div>
                    </template>
                </div>
            </template>
        </div>
    </lightning-card>
</template>
```

CSS Code:

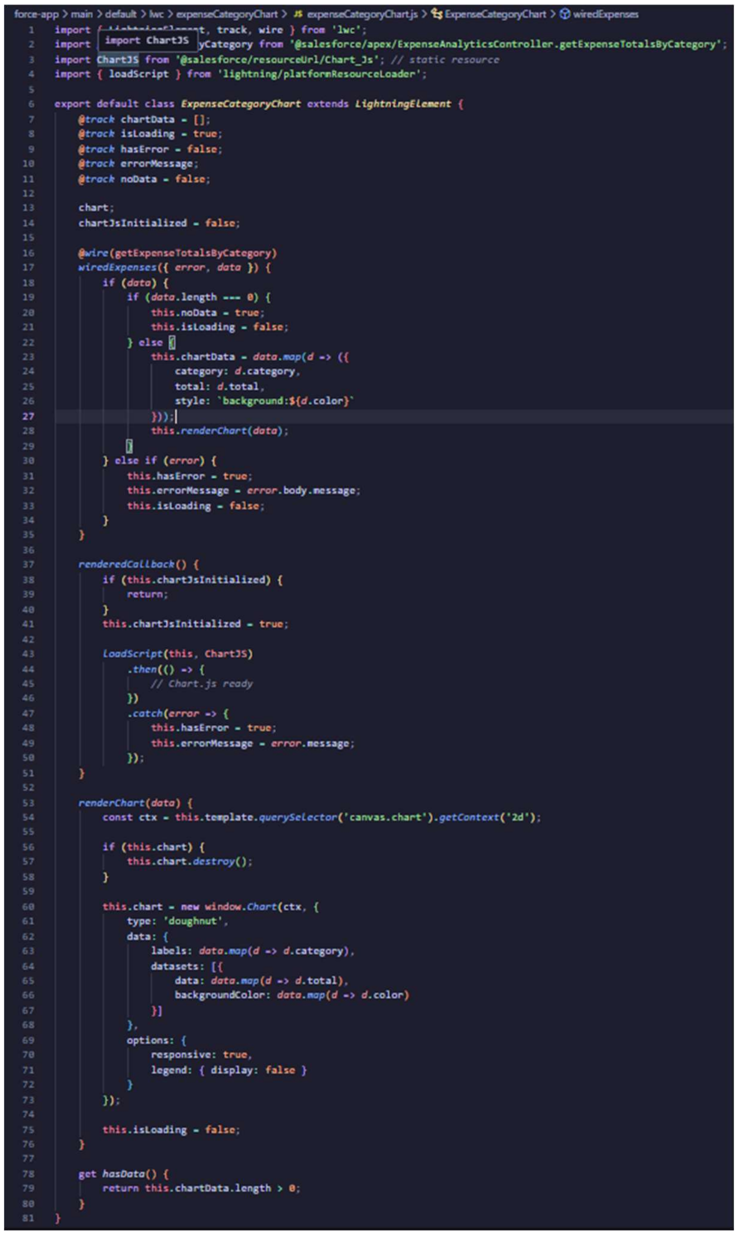
```css
.chart-container {
   position: relative;
   height: 250px;
   width: 250px;
   margin: auto;
}

.legend-item {
   display: flex;
   align-items: center;
   margin-bottom: 6px;
}

.legend-item .swatch {
   display: inline-block;
   width: 12px;
   height: 12px;
   margin-right: 8px;
}
```



```css
.chart-container {
    position: relative;
    height: 250px;
    width: 250px;
    margin: auto;
}

.legend-item {
    display: flex;
    align-items: center;
    margin-bottom: 6px;
}

.legend-item .swatch {
    display: inline-block;
    width: 12px;
    height: 12px;
    margin-right: 8px;
}
```

JavaScript Code:

```
import { LightningElement, track, wire } from 'lwc';
import getExpenseTotalsByCategory from
'@salesforce/apex/ExpenseAnalyticsController.getExpenseTotalsByCategory';
import ChartJS from '@salesforce/resourceUrl/Chart_Js'; // static resource
import { loadScript } from 'lightning/platformResourceLoader';
export default class ExpenseCategoryChart extends LightningElement {
  @track chartData = [];
  @track isLoading = true;
  @track hasError = false;
  @track errorMessage;
  @track noData = false;
  chart;
  chartJsInitialized = false;
 @wire(getExpenseTotalsByCategory)
  wiredExpenses({ error, data }) {
     if (data) {
       if (data.length === 0) {
         this.noData = true;
         this.isLoading = false;
        } else {
          this.chartData = data.map(d => ({
           category: d.category,
           total: d.total,
           style: `background:${d.color}`}));
        this.renderChart(data); }
} else if (error) {
       this.hasError = true;
       this.errorMessage =
error.body.message;
       this.isLoading = false;
     }
}
 renderedCallback() {
     if (this.chartJsInitialized) {
     return; }
  this.chartJsInitialized = true;
   loadScript(this, ChartJS)
      .then(() => {})
        .catch(error => {
         this.hasError = true;
         this.errorMessage = error.message; });
renderChart(data) {
      const ctx = this.template.querySelector('canvas.chart').getContext('2d');
```



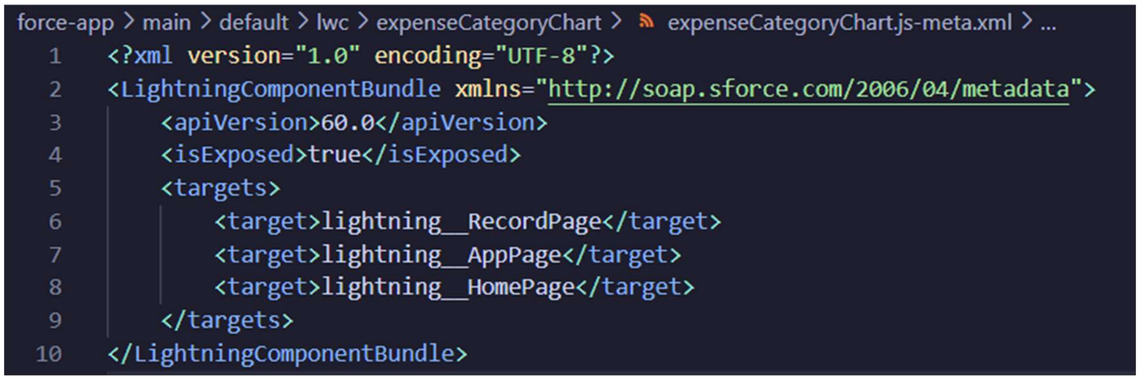
force-app > main > default > lwc > expenseCategoryChart > expenseCategoryChart.js > ExpenseCategoryChart > wiredExpenses

```
import { LightningElement, track, wire } from 'lwc';
import  import ChartJS  yCategory from '@salesforce/apex/ExpenseAnalyticsController.getExpenseTotalsByCategory';
import ChartJS from '@salesforce/resourceUrl/Chart_Js'; // static resource
import { loadScript } from 'lightning/platformResourceLoader';

export default class ExpenseCategoryChart extends LightningElement {
    @track chartData = [];
    @track isLoading = true;
    @track hasError = false;
    @track errorMessage;
    @track noData = false;

    chart;
    chartJsInitialized = false;

    @wire(getExpenseTotalsByCategory)
    wiredExpenses({ error, data }) {
        if (data) {
            if (data.length === 0) {
                this.noData = true;
                this.isLoading = false;
            } else {
                this.chartData = data.map(d => ({
                    category: d.category,
                    total: d.total,
                    style: `background:${d.color}`
                }));
                this.renderChart(data);
            }
        } else if (error) {
            this.hasError = true;
            this.errorMessage = error.body.message;
            this.isLoading = false;
        }
    }

    renderedCallback() {
        if (this.chartJsInitialized) {
            return;
        }
        this.chartJsInitialized = true;

        loadScript(this, ChartJS)
            .then(() => {
                // Chart.js ready
            })
            .catch(error => {
                this.hasError = true;
                this.errorMessage = error.message;
            });
    }

    renderChart(data) {
        const ctx = this.template.querySelector('canvas.chart').getContext('2d');

        if (this.chart) {
            this.chart.destroy();
        }

        this.chart = new window.Chart(ctx, {
            type: 'doughnut',
            data: {
                labels: data.map(d => d.category),
                datasets: [{
                    data: data.map(d => d.total),
                    backgroundColor: data.map(d => d.color)
                }]
            },
            options: {
                responsive: true,
                legend: { display: false }
            }
        });

        this.isLoading = false;
    }

    get hasData() {
        return this.chartData.length > 0;
    }
}
```

```
            if (this.chart) {
this.chart.destroy();}
      this.chart = new window.Chart(ctx, {
           type: 'doughnut',
              data: {
                 labels: data.map(d => d.category),
                 datasets: [{
                 data: data.map(d => d.total),
                 backgroundColor: data.map(d => d.color) }]
                },
          options: {
              responsive: true,
              legend: { display: false }
           }
          });
this.isLoading = false;
     }
    get hasData() {
    return this.chartData.length > 0;
    }
}
```

Meta Xml File:

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>60.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
</targets>
</LightningComponentBundle>
```



force-app > main > default > lwc > expenseCategoryChart > expenseCategoryChart.js-meta.xml > ...

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>60.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

The final Lightning Web Component:

Spend by Category (This Month)

No expense data for this month.

APEX WITH LWC:-

Using Apex in controller in the above code to fetch the data dynamically.

EVENTS IN LWC:-

Using it to connect two component so that they can communicate with each other.

Eg: this.dispatchEvent(new CustomEvent('refresh'));

WIRE ADAPTER:-

Importing classes from salesforce apex and connecting them with the LWC.

Eg:

import getExpenseTotalsByCategory
from'@salesforce/apex/ExpenseAnalyticsController.getExpenseTotalsByCategory';

import ChartJS from '@salesforce/resourceUrl/Chart_Js'; *// static resource*

IMPERATIVE APEX CALLS:-

We do not have to use this feature in this App.

NAVIGATION SERVICE:-

Used in the above code.