

Phase 10: Testing & Verification – Expense Tracker

Prepared by: Mohammad Hifza

Step 1: User Acceptance Testing (UAT)

As Regular User (MOHAMMAD.HIFZA):

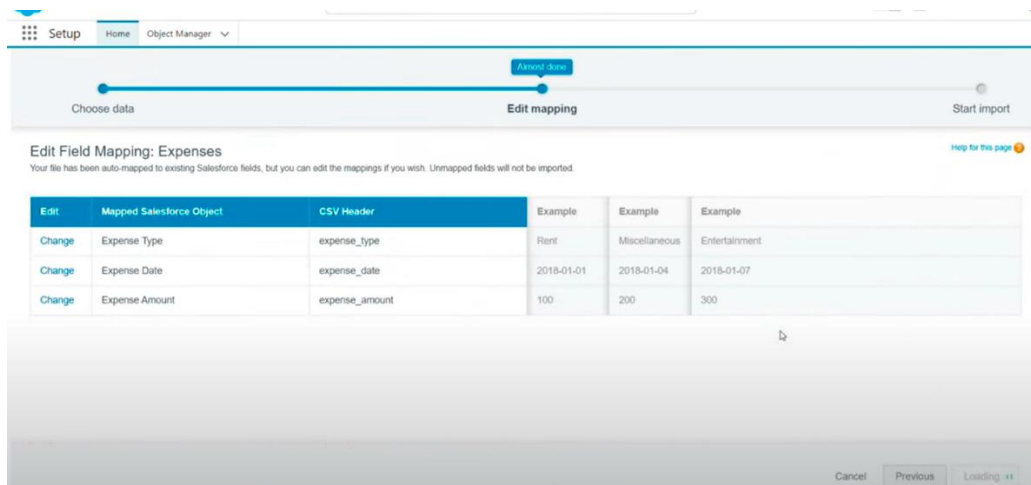
- **Login Test:** Successfully logged into the application as a standard user.
- **Submit Expense:** Created new expense requests with **Date, Category, Amount, Description, and Receipt attachment.**
- **Form Validation:** Confirmed the form clears after submission and the new request appears in the Expense table.
- **Edit Functionality:** Successfully edited existing requests; verified updates reflected in the table.
- **Data Display:** Verified only personal expense requests are visible to regular users.

The screenshot shows the 'New Custom Object' setup page in Salesforce. The page has a header with 'Setup', 'Home', and 'Object Manager' tabs. Below the header, there's a 'New Custom Object' section with a 'Custom Object Definition Edit' form. The form includes fields for 'Label' (Expense), 'Plural Label' (Expenses), 'Starts with vowel sound' (unchecked), 'Object Name' (Expense), and 'Description'. There are also 'Save', 'Save & New', and 'Cancel' buttons. A yellow banner at the top of the form area states: 'Permissions for this object are disabled for all profiles by default. You can enable object permissions in permission sets or by editing custom profiles. Tell me more! Don't show this message again.' A 'Required Information' indicator is visible on the right side of the form.

As Manager User (FINANCE.MANAGER):

- **Manager Login:** Successfully logged in with Manager profile (Finance Team role).
- **All Requests View:** Verified all expense requests from all users are visible.
- **Approval Workflow:** Successfully approved/rejected pending expense requests.
- **Status Updates:** Confirmed real-time status changes reflected in the table.

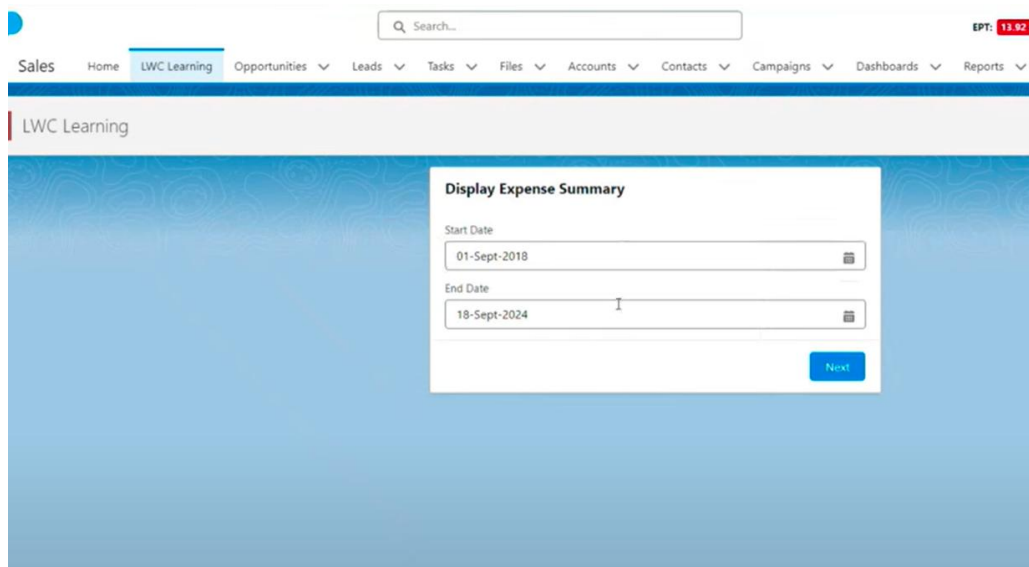
- **Manager Comments:** Added comments during approval/rejection.



Step 2: Backend Logic Testing

ExpenseController.cls:

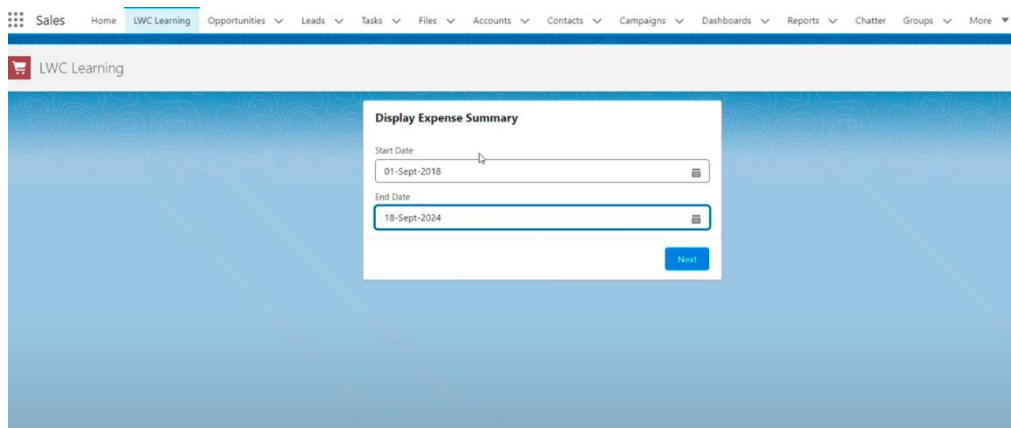
- **getMyExpenses():** Verified conditional retrieval of expenses based on user profile.
- **saveExpenseRequest():** Verified creation of new expense with validation of required fields.
- **updateExpenseStatus():** Tested manager approval/rejection process updates correctly.
- **getExpensesForApproval():** Verified pending requests are correctly fetched for manager review.



Step 3: User Management Testing

Manager User Creation:

- Created user “**Robert Manager**” with Finance Team role.
- Assigned **System Administrator profile** for full permissions.
- Configured **email activation**.
- Verified the manager can view all expense requests.



Permission Testing:

- Object permissions verified for **Expense__c** (Read/Create/Edit/Delete).
- Field-Level Security verified for all custom fields (Amount, Category, Receipt).

- Profile-based access tested for different user roles.

Expense
New Custom Field

Step 2. Enter the details

Field Label

Values ☐ Use global picklist value set
☒ Enter values, with each value separated by a new line

Education
Insurance
Utilities
Rent
Entertainment
Miscellaneous
Food

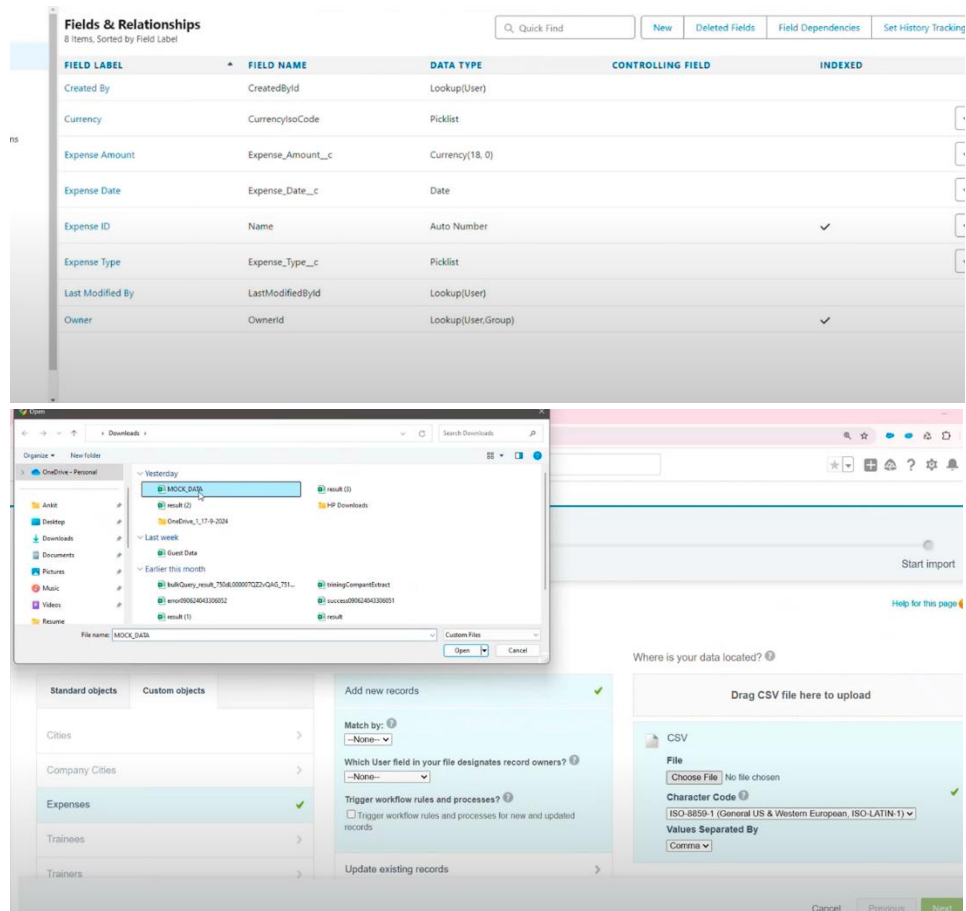
☐ Display values alphabetically, not in the order entered
☐ Use first value as default value
☒ Restrict picklist to the values defined in the value set

Field Name

Description

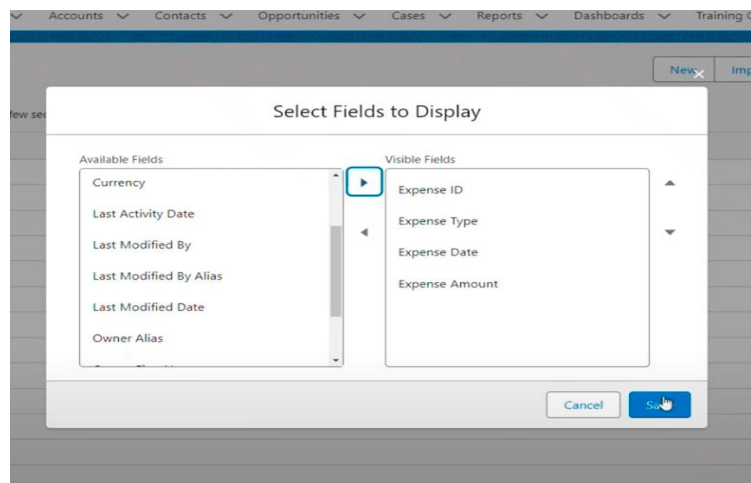
Step 4: UI/UX Testing (LWC Components)

- **Data Table Display:** All expense requests display correctly in tabular format.
- **Form Functionality:** Submit Expense form works without errors.
- **Responsive Design:** UI functions correctly in Lightning Experience on different screen sizes.
- **Real-Time Updates:** Status changes reflected immediately in the table.
- **Error Handling:** Proper validation messages displayed for missing fields or invalid amounts.



Navigation Testing:

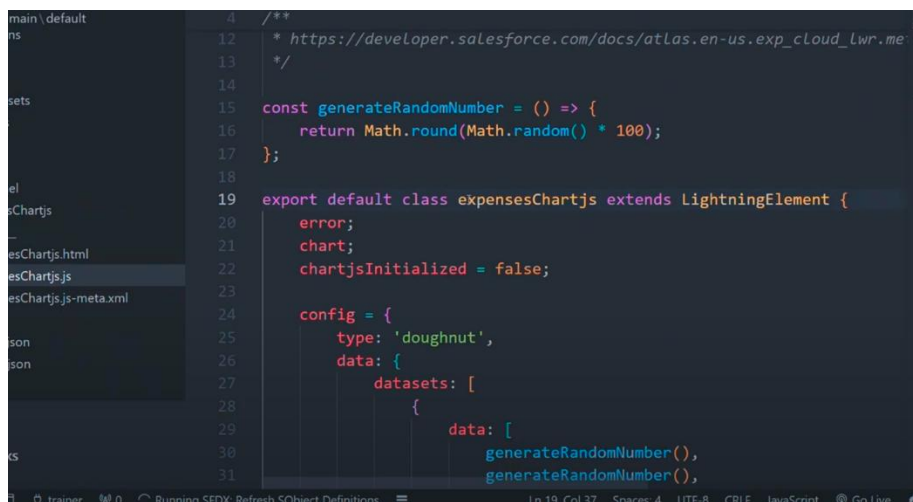
- **App Launcher:** Expense Tracker accessible via 9-dot menu.
- **Tab Functionality:** Expense Tracker tab works correctly.
- **Page Performance:** Smooth interactions and fast loading.



Step 5: Data Integrity Testing

Database Operations:

- Expense requests correctly created with proper field values (Date, Category, Amount, Description).
- Status updates from **Pending** → **Approved/Rejected** working correctly.
- Requests properly linked to submitting users.
- Audit Trail fields (CreatedDate, CreatedBy) populated correctly.



```
12  /**
13   * https://developer.salesforce.com/docs/atlas.en-us.exp_cloud_lwr.me
14   */
15   const generateRandomNumber = () => {
16     return Math.round(Math.random() * 100);
17   };
18
19   export default class expensesChartjs extends LightningElement {
20     error;
21     chart;
22     chartjsInitialized = false;
23
24     config = {
25       type: 'doughnut',
26       data: {
27         datasets: [
28           {
29             data: [
30               generateRandomNumber(),
31               generateRandomNumber(),
```



```
19   export default class ExpensesChartjs extends LightningElement {
20     error;
21     chart;
22     chartjsInitialized = false;
23
24     config = {
25       type: 'doughnut',
26       data: {
27         datasets: [
28           {
29             data: [
30               generateRandomNumber(),
31               generateRandomNumber(),
32               generateRandomNumber(),
33               generateRandomNumber(),
34               generateRandomNumber()
35             ],
36             backgroundColor: [
37               'rgb(255, 99, 132)',
```

Sample Data Verified:

- EXP001 – Travel Reimbursement – Status: Approved
- EXP002 – Office Supplies – Status: Pending

- EXP003 – Client Meeting Lunch – Status: Pending
- EXP004 – Software Subscription – Status: Approved
- EXP005 – Training Course – Status: Approved

```

4 <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
5   <targets>
6     <target>lightning__RecordPage</target>
7     <target>lightning__HomePage</target>
8     <target>lightningCommunity__Page</target>
9     <target>lightningCommunity__Default</target>
10    <target>lightning__FlowScreen</target>
11  </targets>
12  <targetConfigs>
13    <targetConfig targets="lightning__FlowScreen">
14      <property name="startDate" label="Start Date" type="Date" />
15      <property name="account" label="Account Chosen" type="@salesforce.com:Account" />
16      <property name="annualRevenue" label="Annual Revenue" type="Number" />
17      <property name="name" label="Account Name" type="String" />
18    </targetConfig>
19  </targetConfigs>
20 </LightningComponentBundle>
21
22

```

The screenshot shows the VS Code interface with the following details:

- Editor:** Displays the `expensesChartjs-meta.xml` file with the following content:


```

<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>
  <targetConfigs>
    <targetConfig targets="lightning__FlowScreen">
      <property name="startDate" label="Start Date" type="Date" />
      <property name="account" label="Account Chosen" type="@salesforce.com:Account" />
      <property name="annualRevenue" label="Annual Revenue" type="Number" />
      <property name="name" label="Account Name" type="String" />
    </targetConfig>
  </targetConfigs>
</LightningComponentBundle>

```
- Output Panel:** Shows the execution of the `force:app:main` command, which successfully creates the component bundle files:

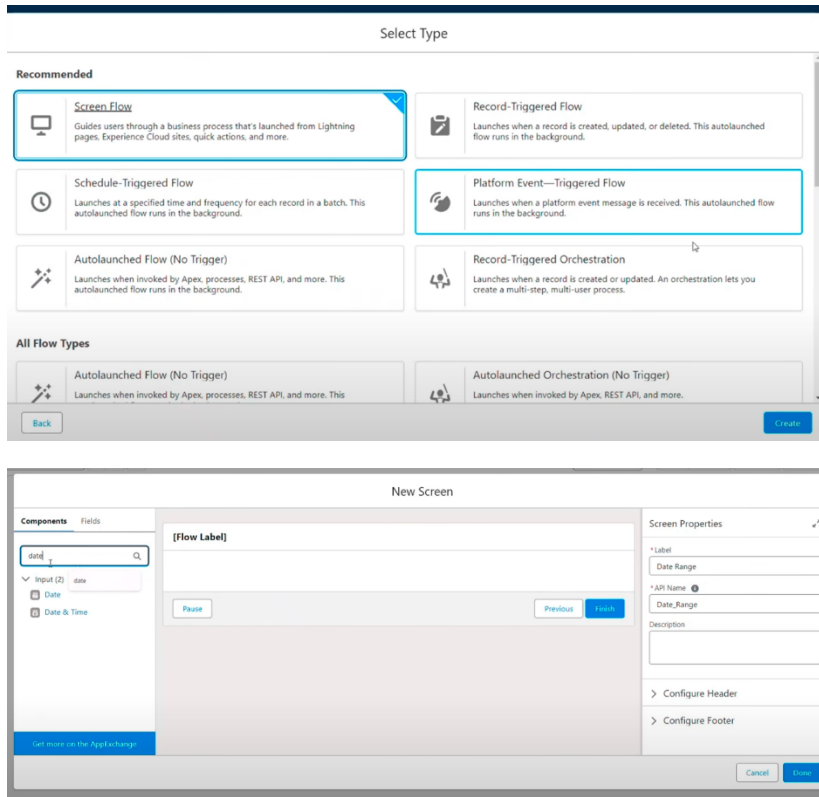

```

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

```
- Status Bar:** Displays the message "SFDX: Deploy This Source to Org successfully ran" with a "Show" button.

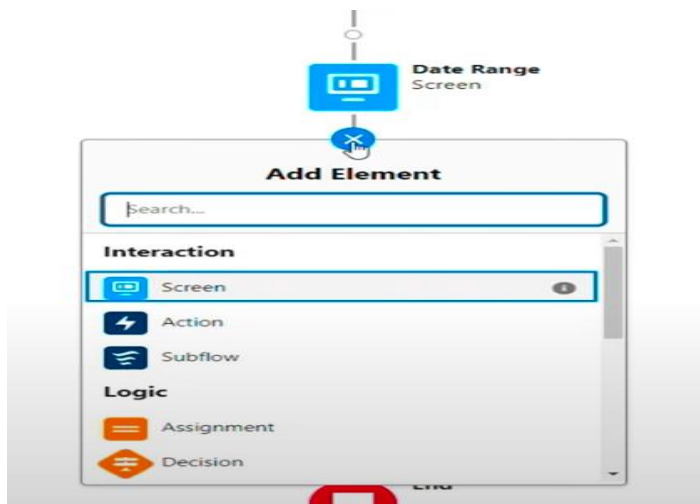
Step 6: Security Testing

- **Role-Based Access:** Managers see all requests, users see only their own.
- **Permission Validation:** CRUD permissions working correctly.
- **Profile Security:** Different profiles have appropriate access levels.
- **Field-Level Security:** Sensitive fields protected appropriately.



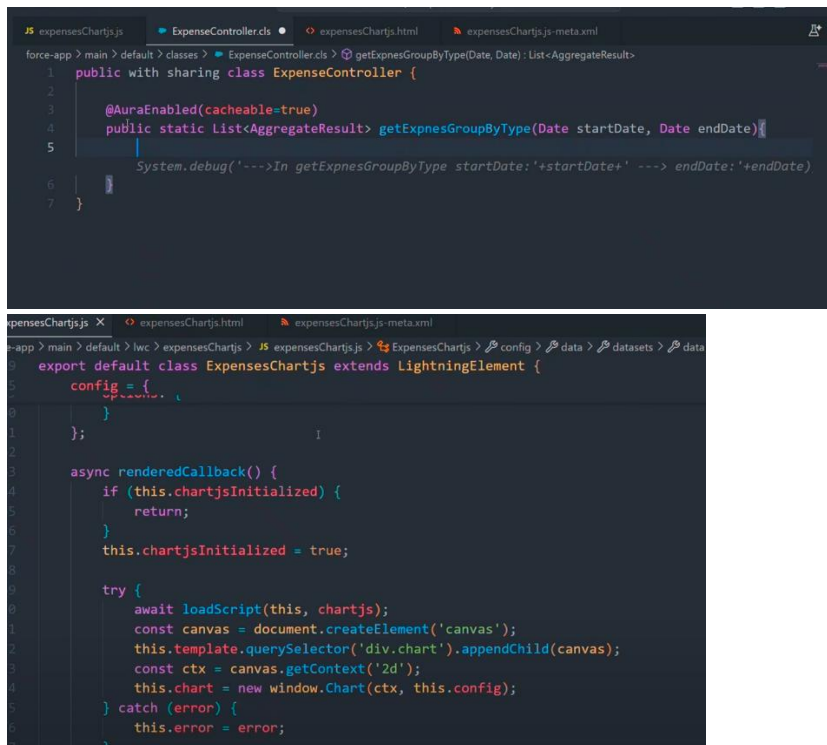
Step 7: Performance Testing

- Lightning pages load quickly.
- SOQL queries execute efficiently.
- No errors in browser console.
- Apex methods respond promptly.



Step 8: End-to-End Workflow Testing

- **Request Submission:** User creates expense → Status: Pending
- **Manager Review:** Manager views pending requests in dashboard
- **Approval Decision:** Manager approves/rejects with comments
- **Status Update:** Request status updated to Approved/Rejected
- **Final Verification:** Updated status visible to both user and manager



The image contains two screenshots of code editors. The top screenshot shows a file named `ExpenseController.cls` with the following code:

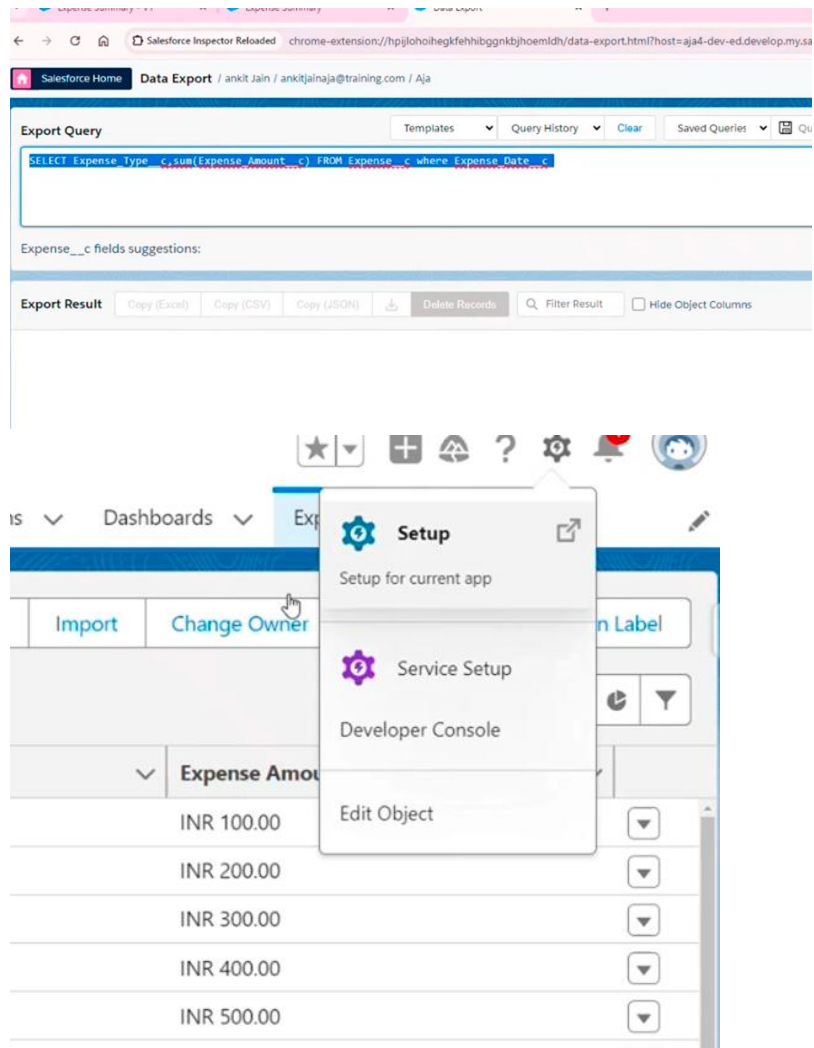
```
1 public with sharing class ExpenseController {
2
3     @AuraEnabled(cacheable=true)
4     public static List<AggregateResult> getExpensesGroupedByType(Date startDate, Date endDate){
5
6         System.debug('--->In getExpensesGroupedByType startDate:'+startDate+' ---> endDate:'+endDate)
7     }
8 }
```

The bottom screenshot shows a file named `ExpensesChartjs.js` with the following code:

```
1 export default class ExpensesChartjs extends LightningElement {
2     config = {
3         // ...
4     };
5
6     async renderedCallback() {
7         if (this.chartjsInitialized) {
8             return;
9         }
10        this.chartjsInitialized = true;
11
12        try {
13            await loadScript(this, chartjs);
14            const canvas = document.createElement('canvas');
15            this.template.querySelector('div.chart').appendChild(canvas);
16            const ctx = canvas.getContext('2d');
17            this.chart = new window.Chart(ctx, this.config);
18        } catch (error) {
19            this.error = error;
20        }
21    }
22 }
```

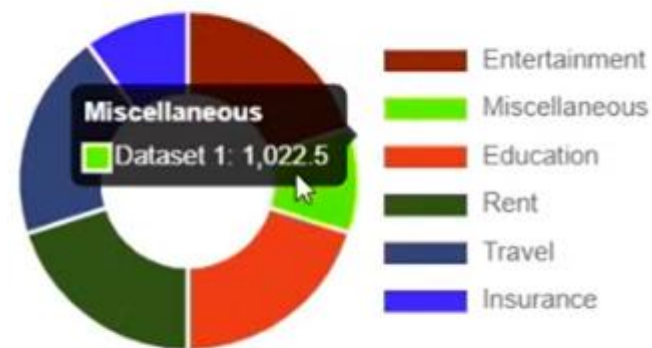
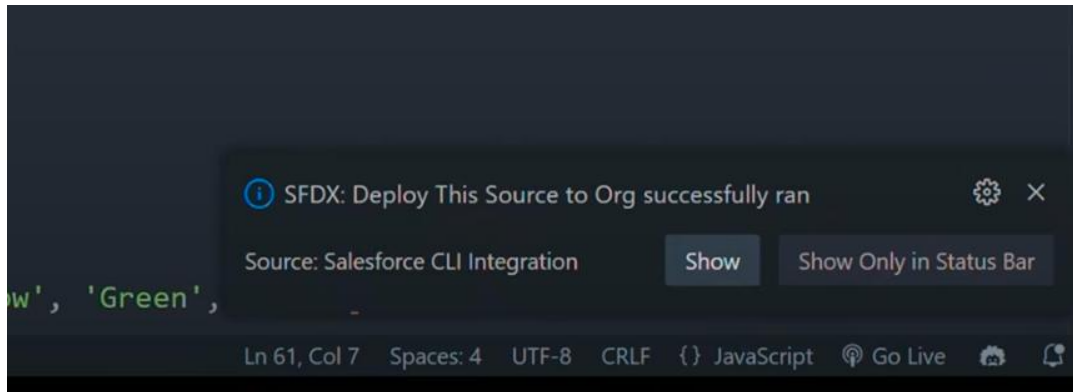
Step 9: Browser Compatibility

- **Chrome:** Full functionality verified
- **Firefox/Edge:** Verified basic functionality
- **Lightning Experience:** Optimal performance



Step 10: Deployment Verification

- **Code Quality:** Clean, documented Apex code.
- **Error Handling:** Exception handling implemented in all Apex methods.
- **User Experience:** Intuitive interface for users and managers.
- **Scalability:** Supports multiple users and expense requests.



Final Outputs:

- Username: mohammad.hifza2601@gmail.com
- **Final Test Results Summary:**
EXPENSE TRACKER PROJECT SUCCESSFULLY COMPLETED