**Project Documentation: Expense Request Approval Process**

**Name: Shaik Javeed Ahammad**

1. **Introduction**

Managing expenses in an organization is a crucial process. Traditionally, employees submit requests for purchases, which are manually checked against budgets, approved or rejected by managers, and then communicated back. This manual approach is **time-consuming**, **errorprone**, and **inefficient**.

This project implements a **Salesforce Record-Triggered Flow** that automates the **approval/rejection process based on expense category and amount**, and sends an email to the requester using **Send Email action** (without using a template).

**Objectives:**

* + Automate approvals for category-based expenses
  + Notify requesters immediately via email
  + Minimize manual intervention
  + Keep the object and flow simple (Name, Amount, Category, Status)

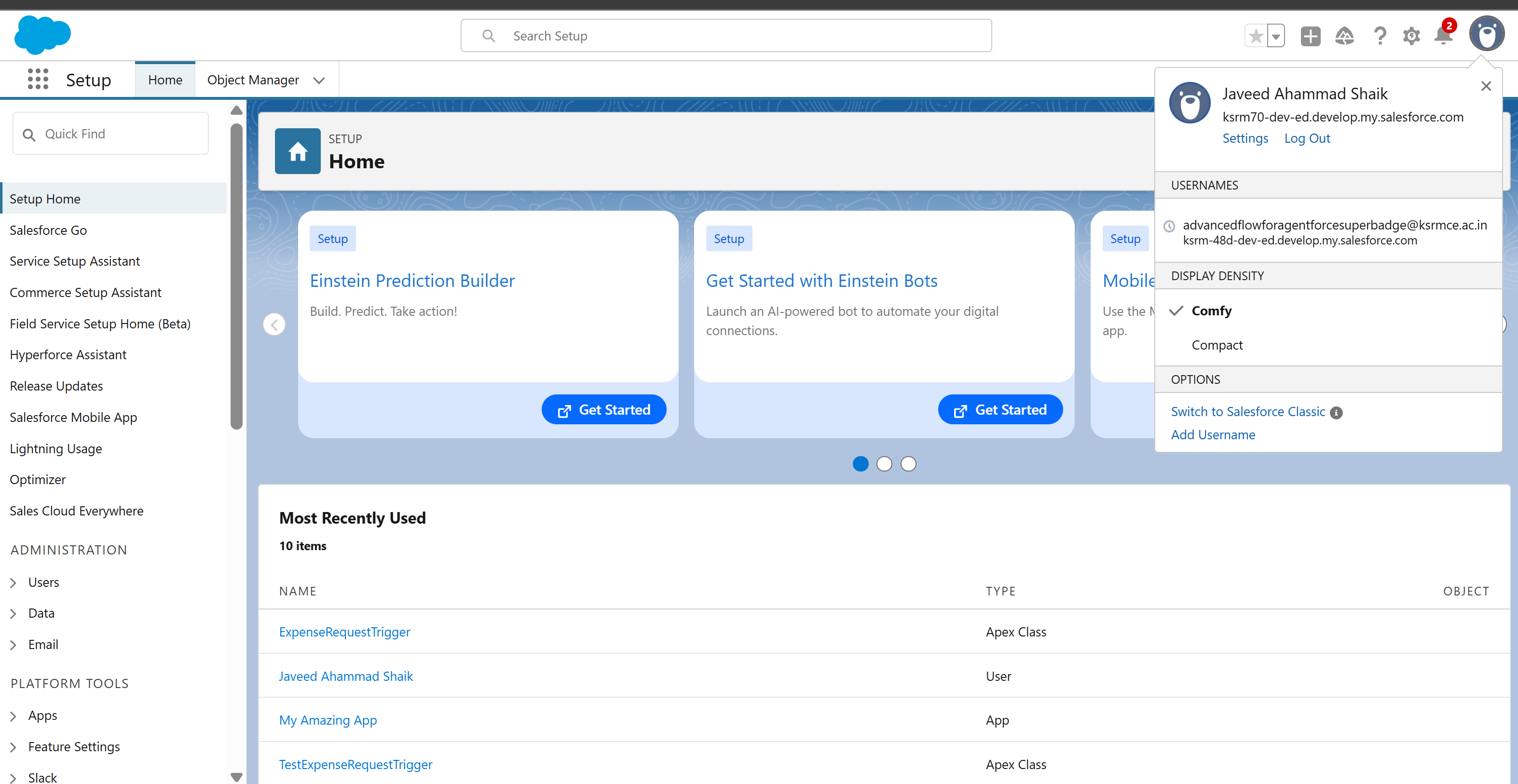
1. **Salesforce Org Setup**

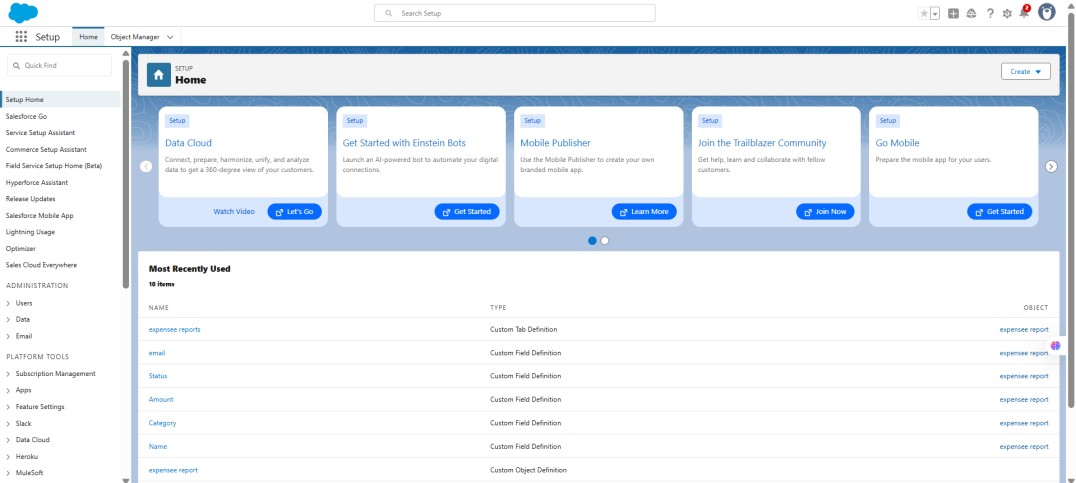
**Step 1: Sign Up / Login**

* + Go to https://trailhead.salesforce.com
  + Login with your credentials, or create a **Developer Edition org** for testing

**Step 2: Navigate to Setup**

* + Click **Gear Icon → Setup**
  + Use **Quick Find → Object Manager** to create custom objects





1. **Create Custom Object: Expense\_Report\_\_c**

**Steps:**

* 1. In **Setup → Object Manager → Create → Custom Object**
  2. Fill details:
     + Label: Expense Report o Plural Label: Expense Reports o Object Name: Expense\_Report\_\_c o Record Name: Name (Text)
  3. Leave optional features unchecked for simplicity
  4. Save object

**Create Fields:**

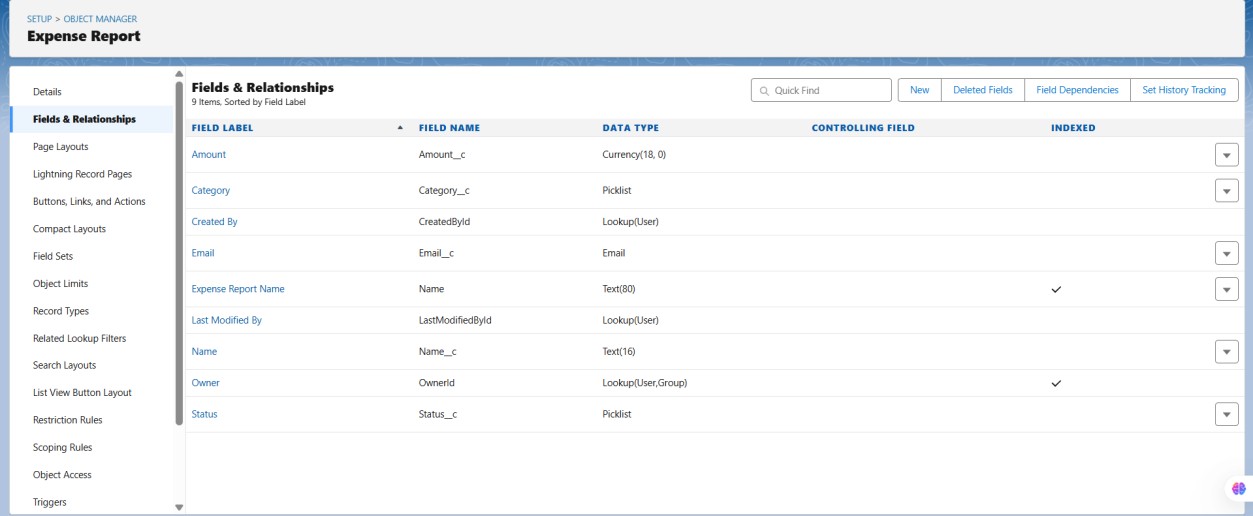
* + **Amount\_\_c**
    - Data Type: Currency o Label: Amount o Length & Decimal: 16, 2
  + **Category**
    - Data Type: Picklist o Values: Veggies, Groceries, Clothing, Savings, Travel, Cosmetics
  + **Status\_\_c** o Data Type: Picklist
    - Values: Submitted, Approved, Not Approved

**4**

**. Build Record**

**-**

**Triggered Flow**



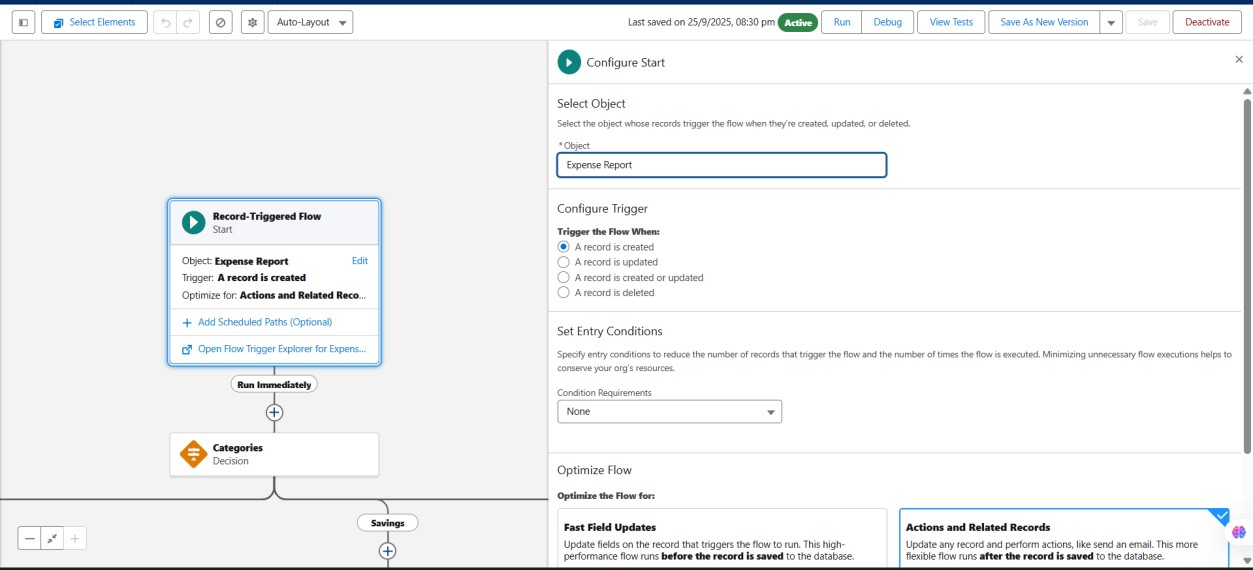
**Step 1: Go to Flow Builder**

* + Setup → Flow → New Flow → **Record-Triggered Flow**
  + Object: Expense\_Report\_\_c
  + Trigger: **When a record is created**
  + Entry Condition: None (all new records trigger flow)
  + Optimize for: Actions and Related Records

•

Click

**Done**



**Step 2: Add Decision Element – Category + Amount**

**Element Name:** Categories

**Purpose:** Directs flow based on Category and Amount

**Outcomes:**

In this Expense Request Approval Flow, each expense category is assigned a maximum allowable amount to determine automatic approval or rejection. For example, “Veggies” has a limit of ₹500, “Groceries” ₹2000, “Clothing” ₹4000, “Savings” ₹8000, “Travel” ₹6000, and “Cosmetics” ₹5000. When a request is submitted, the Flow evaluates both the category and the requested amount: if the amount is within the set limit, the Status field is updated to “Approved” and an approval email is sent to the requester; if the amount exceeds the limit, the Status is set to “Not Approved” and a rejection email is triggered. This structured approach ensures that all requests are handled consistently, minimizes manual intervention, and keeps users informed in real-time about the outcome of their expense requests.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Category** | **Maximum Amount** | **Status if ≤ Max** | **Status if**  **> Max** | **Email Message if Approved** | **Email Message if Rejected** |  |
| Veggies | 500 | Approved | Not  Approved | “Your expense  request of amount {!Amount\_\_c} has been approved under  Veggies.” | “Your expense  request of amount {!Amount\_\_c} exceeds the allowed budget for Veggies.” |
| Groceries | 2000 | Approved | Not  Approved | “Your expense  request of amount {!Amount\_\_c} has been approved under  Groceries.” | “Your expense  request of amount {!Amount\_\_c} exceeds the allowed budget for Groceries.” |
| Clothing | 4000 | Approved | Not  Approved | “Your expense  request of amount {!Amount\_\_c} has been approved under  Clothing.” | “Your expense  request of amount {!Amount\_\_c} exceeds the allowed budget for Clothing.” |
| Savings | 8000 | Approved | Not  Approved | “Your expense  request of amount {!Amount\_\_c} has been approved under  Savings.” | “Your expense  request of amount {!Amount\_\_c} exceeds the allowed budget for Savings.” |
| Travel | 6000 | Approved | Not  Approved | “Your expense  request of amount {!Amount\_\_c} has been approved under  Travel.” | “Your expense  request of amount {!Amount\_\_c} exceeds the allowed budget for Travel.” |
| Cosmetics | 5000 | Approved | Not  Approved | “Your expense  request of amount {!Amount\_\_c} has been approved under Cosmetics.” | “Your expense  request of amount {!Amount\_\_c} exceeds the allowed budget for Cosmetics.” |

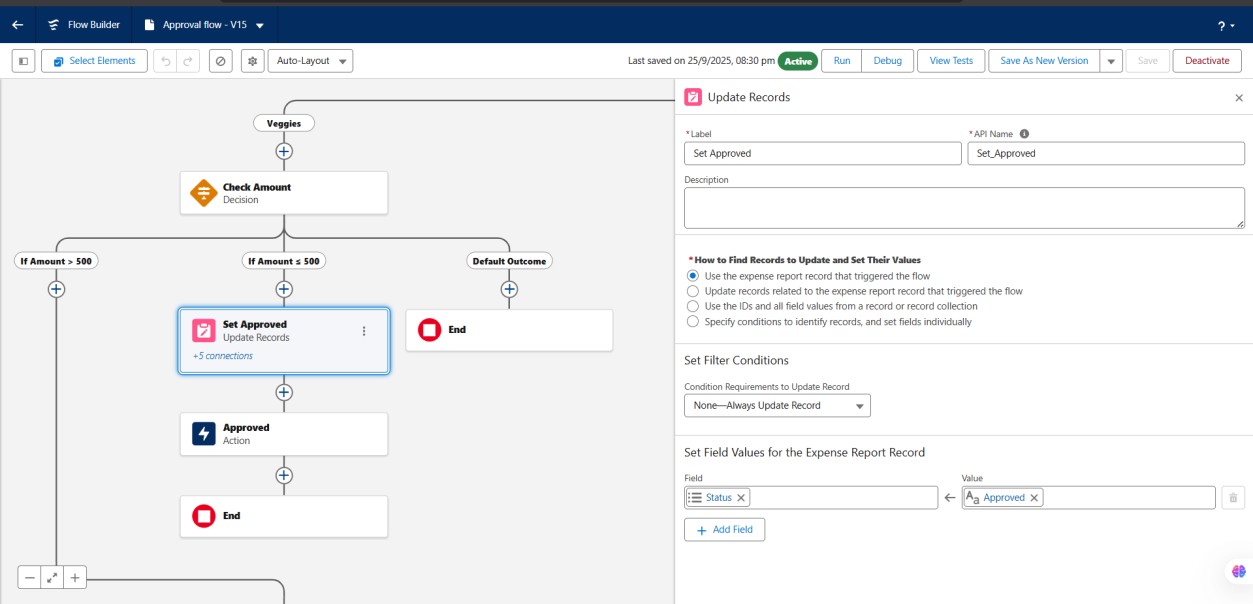
**Notes:**

* Default outcome handles unexpected categories
* Name outcomes descriptively (e.g., “Veggies – Approved”)

**Step 3: Update Records Element**

* Drag **Update Records** onto each branch (Approved/Rejected)
* Update **Status\_\_c**:
  + Approved branch → Status\_\_c = Approved o Rejected branch → Status\_\_c = Not Approved

**Important:** Update occurs **before** sending the email to ensure correct status is included.



**Step 4: Send Email Action (No Template)**

* Drag **Action → Send Email** after Update Records
* Configuration:
  + **Recipient:** User email or a configured email field o **Subject:** Expense Request Status o **Body:** Use HTML in the action itself

**Example for Approval:**

<p>Hello, your expense request of amount {!Amount\_\_c} has been approved under the {!Category} category.</p>

<p>You may proceed with your purchase.</p>

**Example for Rejection:**

<p>Hello, your expense request of amount {!Amount\_\_c} exceeds the allowed budget for the {!Category} category.</p>

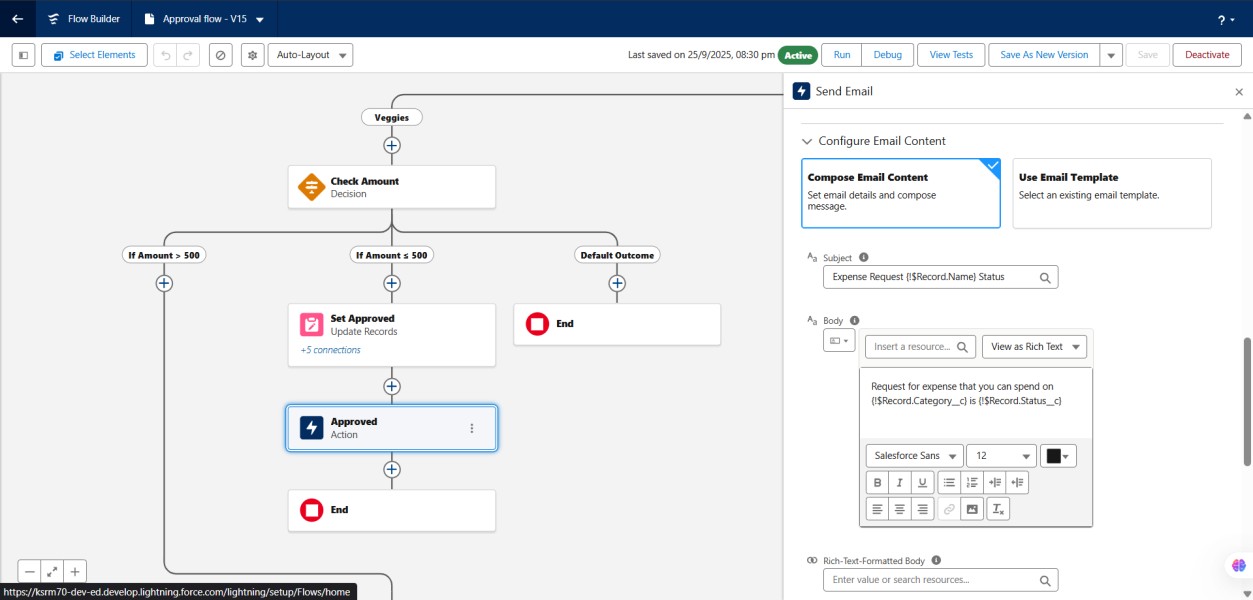
<p>Please review and adjust your request.</p>

**Tips:**

* Use merge fields directly in flow {!Amount\_\_c}, {!Category}
* Email subject can also include merge fields if desired

**5**

**. Testing**



**Step 1: Create Test Records**

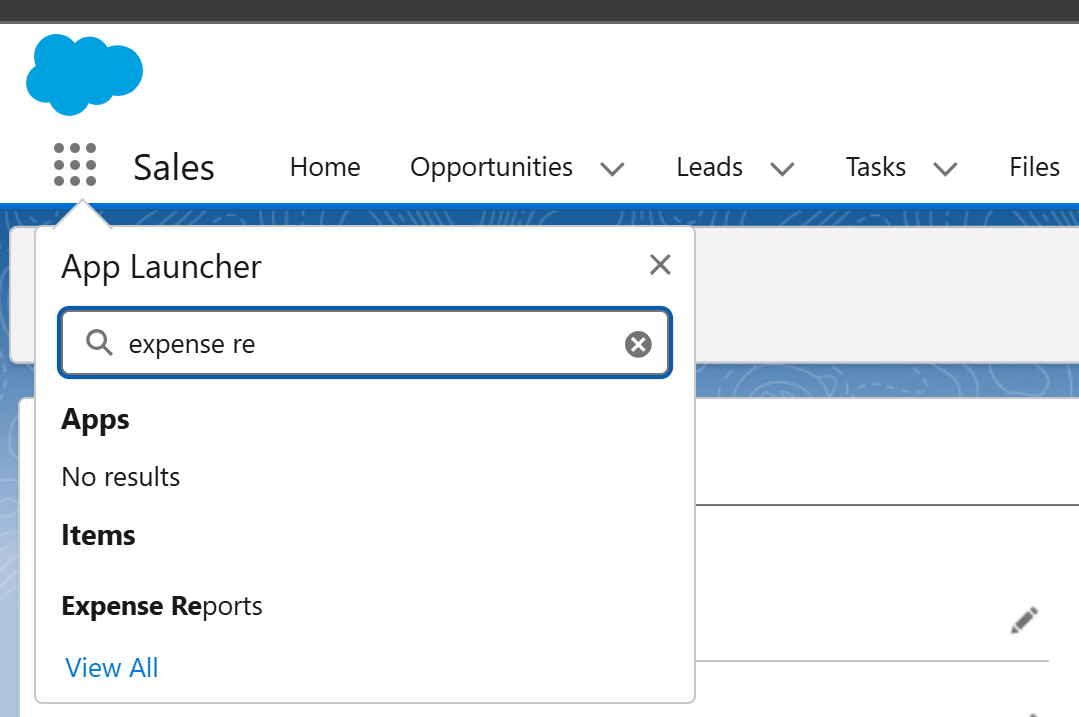
* Go to **Expense Reports → New**
* Enter different combinations of Category and Amount

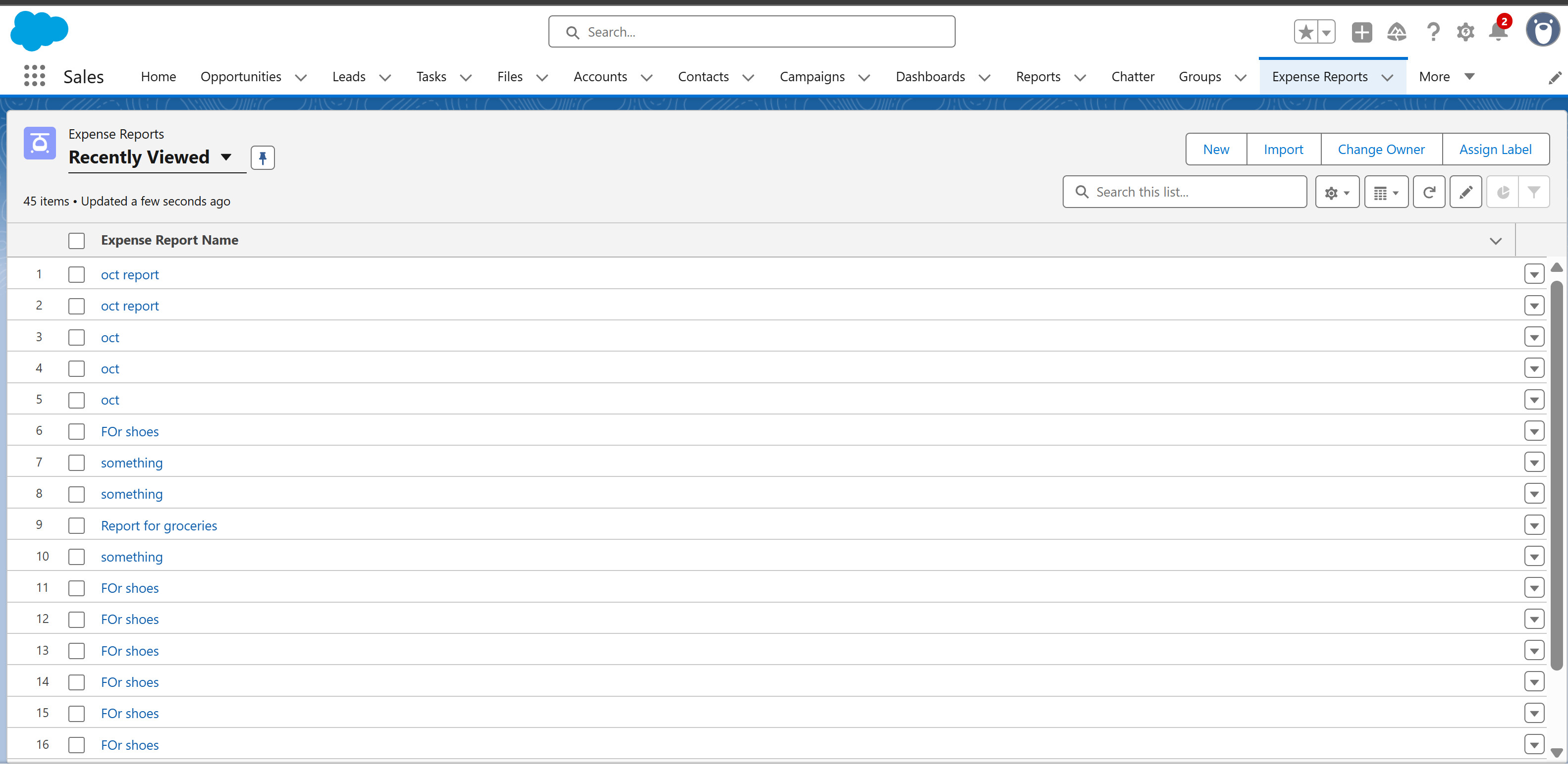
**Step 2: Verify Flow Execution**

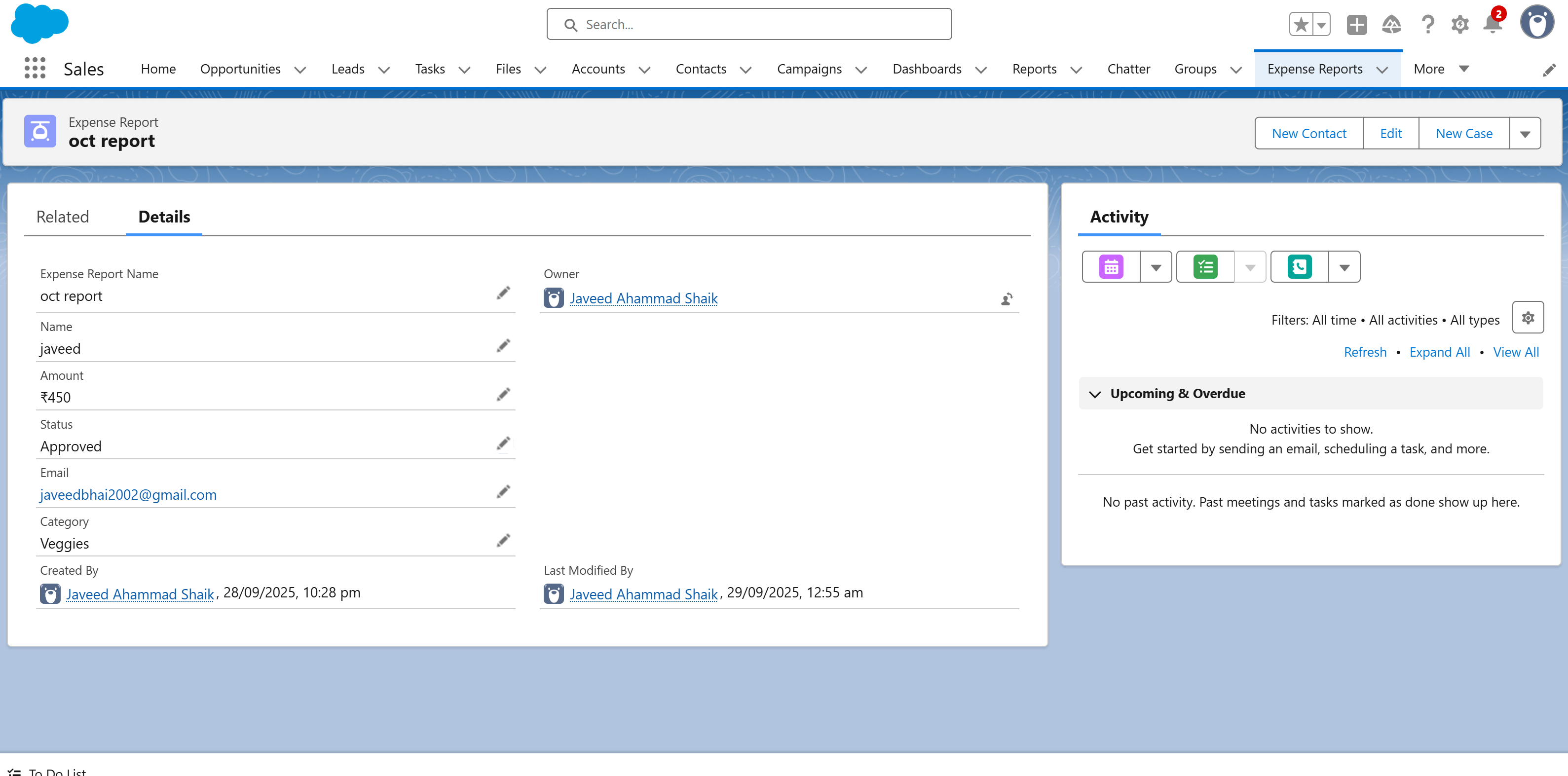
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test**  **Case** | **Category** | **Amount** | **Expected Status** | **Email Output** |
| 1 | Veggies | 450 | Approved | “Amount 450 approved under  Veggies.” |
| 2 | Veggies | 650 | Not Approved | “Amount 650 exceeds allowed budget for Veggies.” |
| 3 | Travel | 6000 | Approved | “Amount 6000 approved under  Travel.” |
| 4 | Travel | 6100 | Not Approved | “Amount 6100 exceeds allowed budget for Travel.” |
| 5 | Clothing | 4000 | Approved | “Amount 4000 approved under  Clothing.” |
| 6 | Clothing | 4500 | Not Approved | “Amount 4500 exceeds allowed budget for Clothing.” |

**Edge Cases:**

* Category blank → Default Outcome → Not Approved
* Amount = exact limit → Approved







1. **Error Handling** 
   * **Status not updating:** Ensure Update Records element runs **before** Send Email
   * **Email not received:** Check user email deliverability and spam folder
   * **Merge fields blank:** Verify API names match {!Amount\_\_c} and {!Category}
   * **Unexpected category:** Default outcome prevents flow failure

* + Fully automated approval/rejection based on Category + Amount
  + Instant email notification without templates
  + Minimal user input (Name, Amount, Category)
  + Scalable for future enhancements
  + Reduces human error and improves efficiency

1. **Future Enhancements** 
   * Add Manager approval for high-value requests
   * Conditional routing for different departments
   * Include attachments (e.g., receipts)
   * Create reporting dashboards
   * Make flow mobile-friendly for submissions

1. **Conclusion**

This **full Category + Amount-based Expense Flow** automates approval/rejection and sends email notifications directly via **Send Email action**.

**Key Highlights:**

* + Simple object: Name, Amount, Category, Status
  + Flexible flow branching for multiple categories
  + Emails dynamically generated based on outcome
  + Error handling for robust execution
  + Scalable, easy to maintain, ready for production