

# **LATHAMATHAVAN ENGINEERING COLLEGE**

(Approved by AICTE, New Delhi and Affiliated to Anna University, Chennai) (Recognized under section 2(f) of the UGC Act 1956) M Latha Mathavan Nagar, Kidaripatti (P.O), Alagar Kovil Via, Melur (TK), Madurai-625 301



**Subject Name :      ServiceNow Administrator (NM1051)**  
**(Under Naan Mudhalvan Scheme)**

**Project Title : Calculating Family Expenses using ServiceNow**

**Team ID: NM 2025 TMID07640**

**Team leader :**

**K.ARVINDAN (911022104006)**

**Team members :**

**S.HARAN KUMAR (911022104020)**

**D.KARPAGA VINAYAGAM**

**(911022104027)**

**P.KISHORE(911022104030)**

# Calculating Family Expenses using ServiceNow

---

## 1. Objective

The main objective of this project is to **develop a family expense management system** using the **ServiceNow platform**. The system helps users efficiently track, categorize, and manage family expenses. It aims to promote financial awareness and better budgeting within the household.

---

## 2. Introduction

In today’s fast-paced world, managing family expenses is a critical part of financial stability. Manual tracking often leads to confusion, errors, and lack of insight into spending patterns.

This project leverages **ServiceNow**, a robust cloud-based platform, to create an automated and user-friendly **Family Expense Tracker**.

The application allows users to record expenses, set budgets, categorize spending, and generate real-time reports.

By building this project, we demonstrate how ServiceNow can be extended beyond IT service management to real- world, practical use cases like family budgeting.

---

## 3. Project Scope

- Expense Categorization
  - Budget Setting and Comparison
  - Real-time Expense Tracking
- 

## 5. Tools and Technologies Used

Tool/Technology	Description
ServiceNow	Cloud platform for workflow automation
Update Sets	To capture and move customizations
Tables & Fields	Used to store expense records
Forms & Lists	For user interaction and record display
Reports Module	For data visualization and analytics
Browser	Any modern browser (Chrome/Edge)

## 6. System Requirements

### Hardware Requirements

- Processor: Dual Core or higher
- RAM: Minimum 4 GB
- Internet Connection: Stable broadband

### Software Requirements

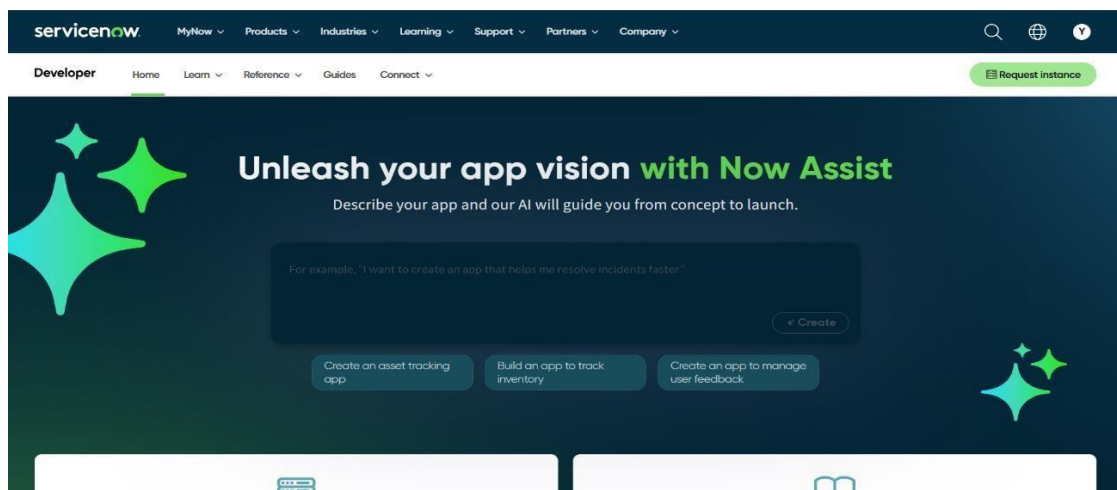
- Web Browser (Google Chrome preferred)
- ServiceNow Developer Instance
- ServiceNow Account (developer.servicenow.com)

---

## 7. Project Implementation Steps

### Step 1: Setting up ServiceNow Instance

1. Visit <https://developer.servicenow.com>
2. Sign up for a free developer account.
3. Log in to your instance to start customizing.



---

### Step 2: Creation of a New Update Set

1. Navigate to **All** → **Local Update Sets** → **New**
2. Enter details:
  - **Name:** Family Expenses
3. Click **Submit and Make Current**  
This ensures all future configurations are captured under the same update set.

ServiceNow Update Set - Create New Update Set

Name: Family Expenses

State: In progress

Parent: [Search]

Release date: [Calendar]

Description: [Text Area]

Application: Global

Submit Submit and Make Current

### Step 3: Creation of Family Expenses Table

1. Navigate to **All** → **Tables** → **New**
2. Enter details:
  - **Label:** Family Expenses
  - **Name:** (Auto-populated)
  - **New menu name:** Family Expenditure
3. Right-click the header and select **Save**.

Label: Family Expenses

Name: list\_family\_expenses

Application: Global

Remote Table: ☒

Create module: ☒

Create mobile module: ☒

### Step 4: Creation of Columns (Fields)

In the **Columns** section, add the following fields:

Column Label	Type	Max Length
Number	String	0
Date	Date	0
Amount	Integer	0
Expense Details	String	800

After adding all fields, **Save** the table.

servicenow All Favorites History Workspaces Admin Table - Family Expenses Search Delete Update Delete All Records

Columns Controls Application Access

Table Columns for text Search 1 to 6 of 6 New

Dictionary Entries	Column label	Type	Reference	Max length	Default value	Display
	Updated by	String	(empty)	40		false
	Created by	String	(empty)	40		false
	Updated	Date/Time	(empty)	40		false
	Sys ID	Sys ID (GUID)	(empty)	32		false
	Created	Date/Time	(empty)	40		false
	Updates	Integer	(empty)	40		false
	Number	String				false
	Date	Date				false
	Amount	Integer				false
	Expense Details	String		900		false
	Insert a new row...					

Delete Update Delete All Records

Related Links

## Step 5: Auto-Number for Family Expenses

Open the **Number** field → **Advanced View** → enable **Use dynamic default** with **Get Next Padded Number** → Update.

Then go to **Number Maintenance** → **New**, set **Table: Family Expenses**, **Prefix: MFE**, and click **Submit**.

## Step 6: Configure Family Expenses Form

Go to **Family Expenses** → **New** → **Configure** → **Form Design**.

Make **Number** read-only, and set **Date** & **Amount** as mandatory → Save.

Family Expenses [u\_family\_expenses] 2 Column

Number Date Amount

Expense Details 1 Column

## Step 7: Create Daily Expenses Table

Navigate to **Tables** → **New**, name it **Daily Expenses**, and link to **Family Expenditure**. Add fields: **Number**, **Date**, **Expense**, **Family Member Name**, **Comments**, then Save.

Label Daily Expenses 1

Name u\_daily\_expenses 1

Extends table

Application Global

Create module

Create mobile module

Add module to menu Family Expenditure 1

## Step 8: Auto-Number for Daily Expenses

Open **Number** field → enable **Get Next Padded Number** → Update.

In **Number Maintenance**, create new with **Table: Daily Expenses**, **Prefix: MDE**, then Submit.

The top screenshot shows the 'Default Value' tab in the Number Maintenance configuration. The 'Use dynamic default' checkbox is checked, and the 'Dynamic default value' is set to 'Get Next Padded Number'. The bottom screenshot shows the 'New record' form for the 'Number' field. The 'Table' is set to 'Daily Expenses', the 'Prefix' is 'MDE', the 'Number' is '1,000', the 'Application' is 'Global', and the 'Number of digits' is '7'. A 'Submit' button is at the bottom.

## Step 9: Configure Daily Expenses Form

Go to **Daily Expenses** → **New** → **Configure** → **Form Design**.

Set **Number** read-only, and **Date & Family Member Name** mandatory → Save.

The screenshot shows the 'Form Design' interface for the 'Daily Expenses' form. The form is divided into two columns. The first column contains the 'Number' field (read-only), the 'Date' field (mandatory), and the 'Comments' field. The second column contains the 'Family Member Name' field (mandatory) and the 'Expense' field (mandatory).

## Step 10: Create Relationship

Open **Relationships** → **New**, set **Name: Daily Expenses**, **Applies to Table: Family Expenses**, and link to **Daily Expenses** → **Save**.

Add **Daily Expenses** as a related list in **Family Expenses** → **Save**.

The screenshot shows the 'Relationship - New Record' form in ServiceNow. The 'Name' field is set to 'Daily Expenses'. The 'Application' is set to 'Global'. The 'Applies to table' dropdown is set to 'Family Expenses [u\_family\_expenses]'. The 'Queries from table' dropdown is set to '-- None --'. Below the form, there is a section for a script that refines the query. The script is as follows:

```
1 (function refineQuery(current, parent) {  
2  
3   // Add your code here, such as current.addQuery(field, value);  
4  
5 })(current, parent);
```

At the bottom of the form, there is a 'Submit' button.

## Step 11: Create Business Rule

Go to **Business Rules** → **New**, name it **Family Expenses BR**, and select **Table: Daily Expenses**. Enable **Advanced**, **Insert**, **Update** and add script to auto-update Family Expenses totals → **Save**.

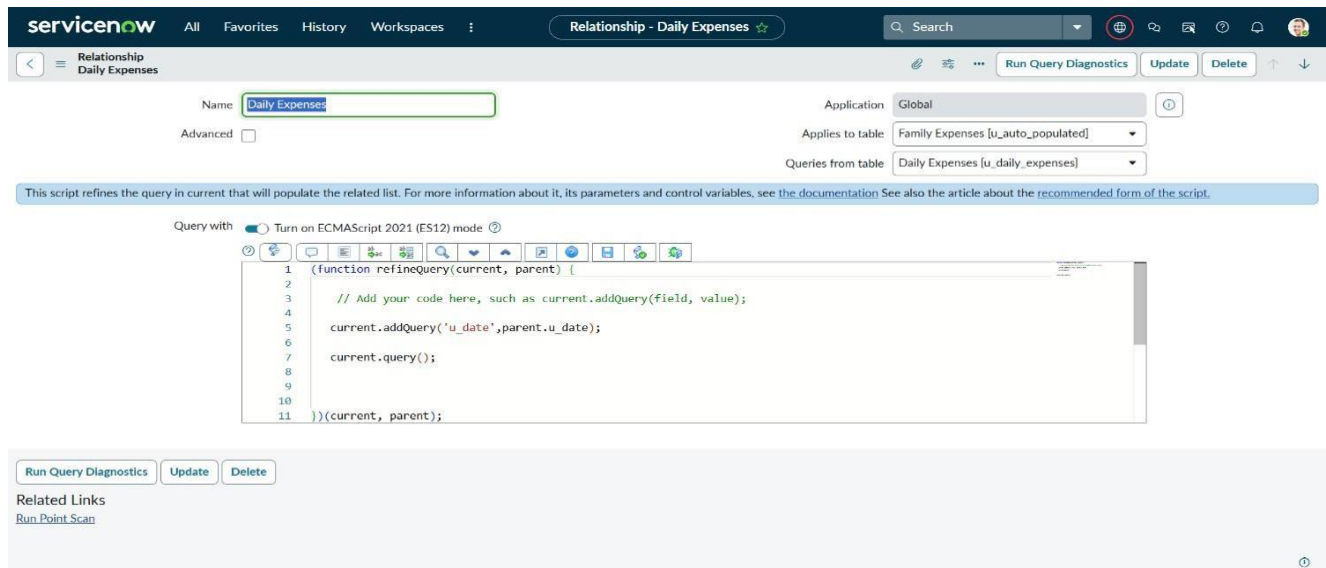
The screenshot shows the 'Business Rule' form in ServiceNow. The 'Name' field is set to 'Daily Expenses'. The 'Application' is set to 'Global'. The 'Applies to table' dropdown is set to 'Family Expenses [u\_family\_expenses]'. The 'Queries from table' dropdown is set to 'Daily Expenses [u\_daily\_expenses]'. Below the form, there is a section for a script that refines the query. The script is as follows:

```
1 (function refineQuery(current, parent) {  
2  
3   // Add your code here, such as current.addQuery(field, value);  
4   current.addQuery('u_date', parent.u_date);  
5   current.query();  
6  
7 })(current, parent);
```

At the bottom of the form, there are 'Update' and 'Delete' buttons. Red arrows point to the 'Applies to table' dropdown, the script editor, and the 'Update' button.

## Step 12: Refine Relationship Query

Open **Daily Expenses Relationship**, set **Applies to: Family Expenses**. Add query script to match **u\_date** fields between tables  
→ Update.



Relationship - Daily Expenses

Name:

Application: Global

Applies to table: Family Expenses [u\_auto\_populated]

Queries from table: Daily Expenses [u\_daily\_expenses]

This script refines the query in current that will populate the related list. For more information about it, its parameters and control variables, see the [documentation](#). See also the article about the [recommended form of the script](#).

Query with: ☒ Turn on ECMAScript 2021 (ES12) mode

```
1 (function refineQuery(current, parent) {
2
3     // Add your code here, such as current.addQuery(field, value);
4
5     current.addQuery('u_date', parent.u_date);
6
7     current.query();
8
9
10
11 })(current, parent);
```

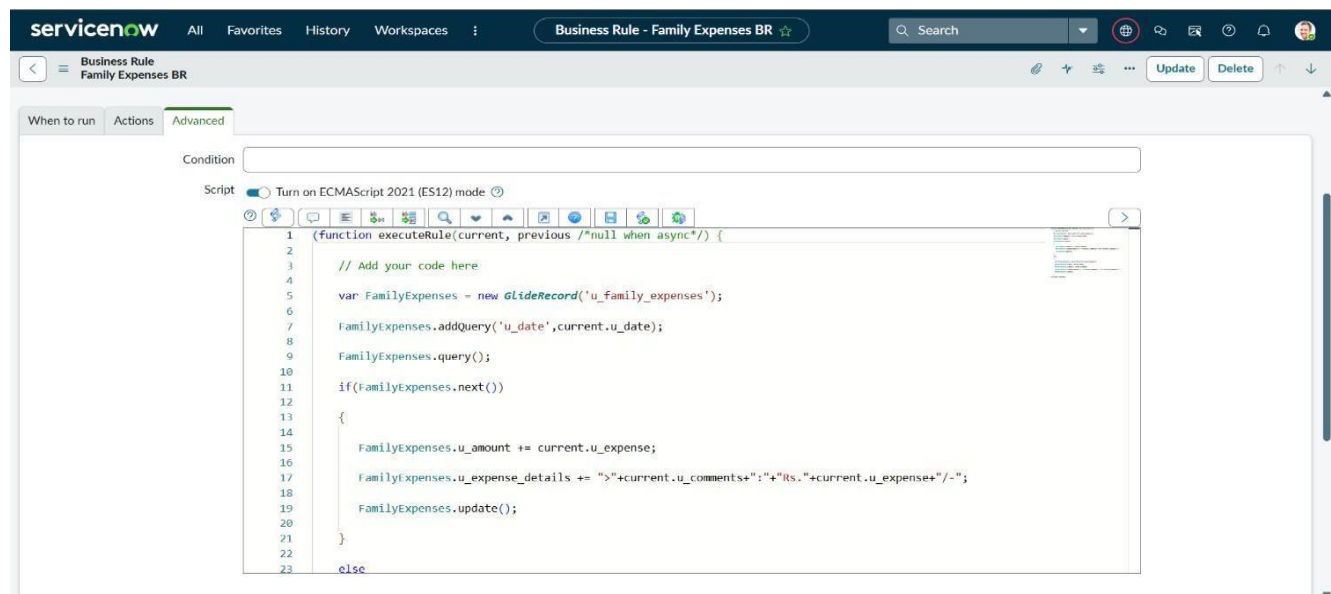
Run Query Diagnostics Update Delete

Related Links

[Run Point Scan](#)

## Step 13: Testing & Output

Add test entries in **Daily Expenses** and verify automatic updates in **Family Expenses**. Check related lists, totals, and generate date-wise or member-wise reports.



Business Rule - Family Expenses BR

When to run: ☒ When a record is created, updated, or deleted

Condition:

Script: ☒ Turn on ECMAScript 2021 (ES12) mode

```
1 (function executeRule(current, previous /*null when async*/) {
2
3     // Add your code here
4
5     var FamilyExpenses = new GlideRecord('u_family_expenses');
6
7     FamilyExpenses.addQuery('u_date', current.u_date);
8
9     FamilyExpenses.query();
10
11     if(FamilyExpenses.next())
12     {
13
14
15         FamilyExpenses.u_amount += current.u_expense;
16
17         FamilyExpenses.u_expense_details += ">" + current.u_comments + ":" + current.u_expense + "-";
18
19         FamilyExpenses.update();
20
21     }
22
23     else
```



## **Conclusion**

A complete ServiceNow-based expense tracker with automated calculations, dynamic relationships, and real-time reporting — demonstrating workflow automation for practical financial management.

