

# CALCULATING FAMILY EXPENSES

**TEAM ID: NM2025TMID20216**

**TEAM MEMBERS: 3**

**TEAM LEADER: GAYATHRI S**

**MEMBER: MONISHA R**

**MEMBER: ARCHANA K**

**PROJECT STATEMENT:** This Service Now project aims to streamline and improve how users manage family-related financial activities. It will offer features such as expense tracking, category management, budget planning, real-time monitoring, and detailed reporting. Leveraging Service Now's powerful and flexible platform, the project will deliver a user-friendly, scalable solution that fits different family sizes and financial needs. The ultimate goal is to equip users with the tools to make smart financial decisions and enhance overall financial well-being within the family unit.

**Objective :** The " family expensive project " phrase seems to be a misunderstanding, but if referring to Service Now's Strategic Portfolio Management (SPM) , also known as Project Portfolio Management (PPM), its objective is to strategically manage and optimize a portfolio of projects by aligning them with business goals, providing real-time visibility into performance, and improving agility and resource allocation to deliver maximum customer value.

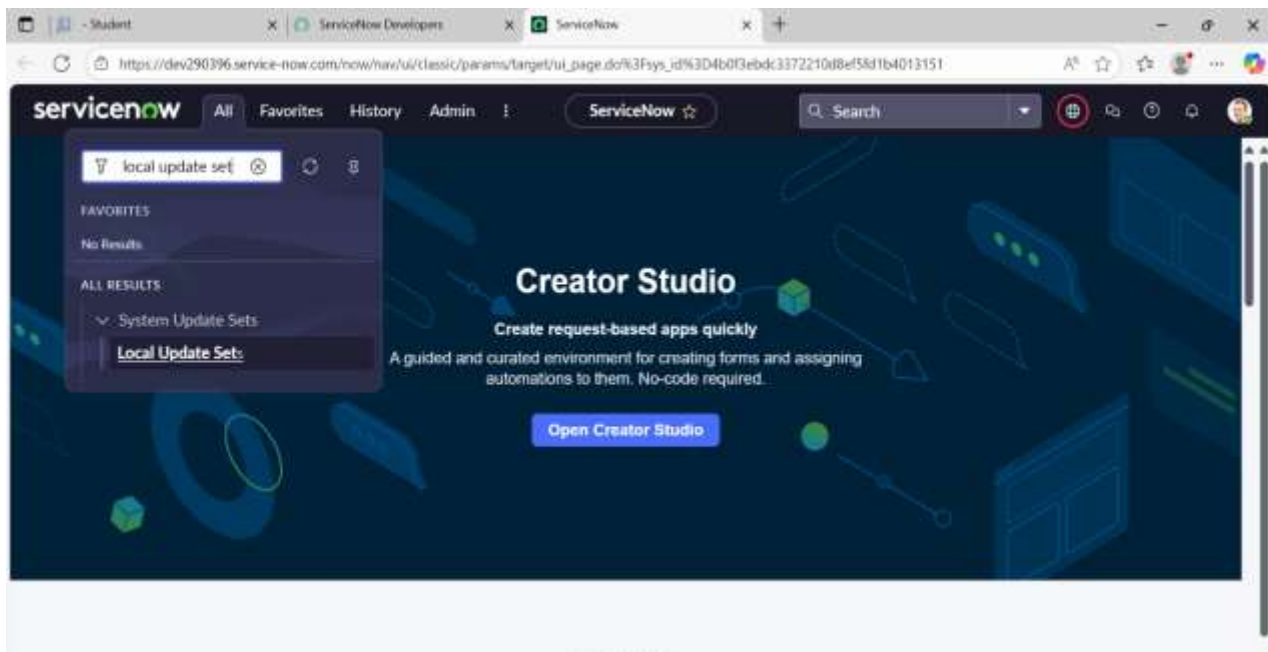
**Skills:** Service now Family expenses IOT Open Hardware platforms, Data Structures

## TASK INITIATION

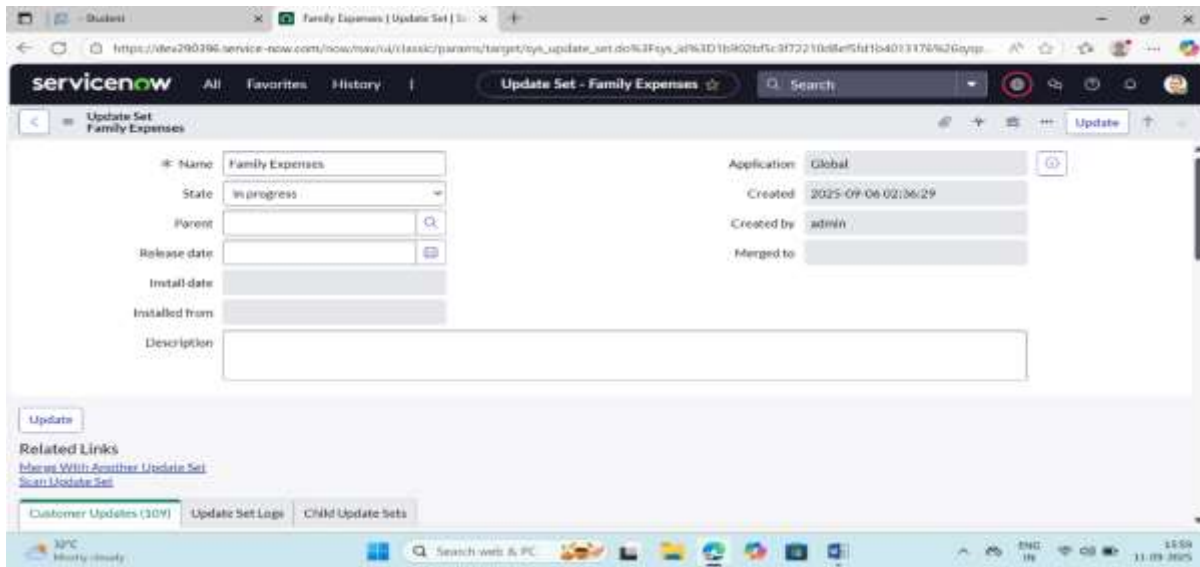
### MILESTONE 1: A NEW UPDATE SET

#### ACTIVITY 1: CREATE NEW LOCAL UPDATE SET

1. Open service now
2. Go to All → search for Local Update set → click on New.



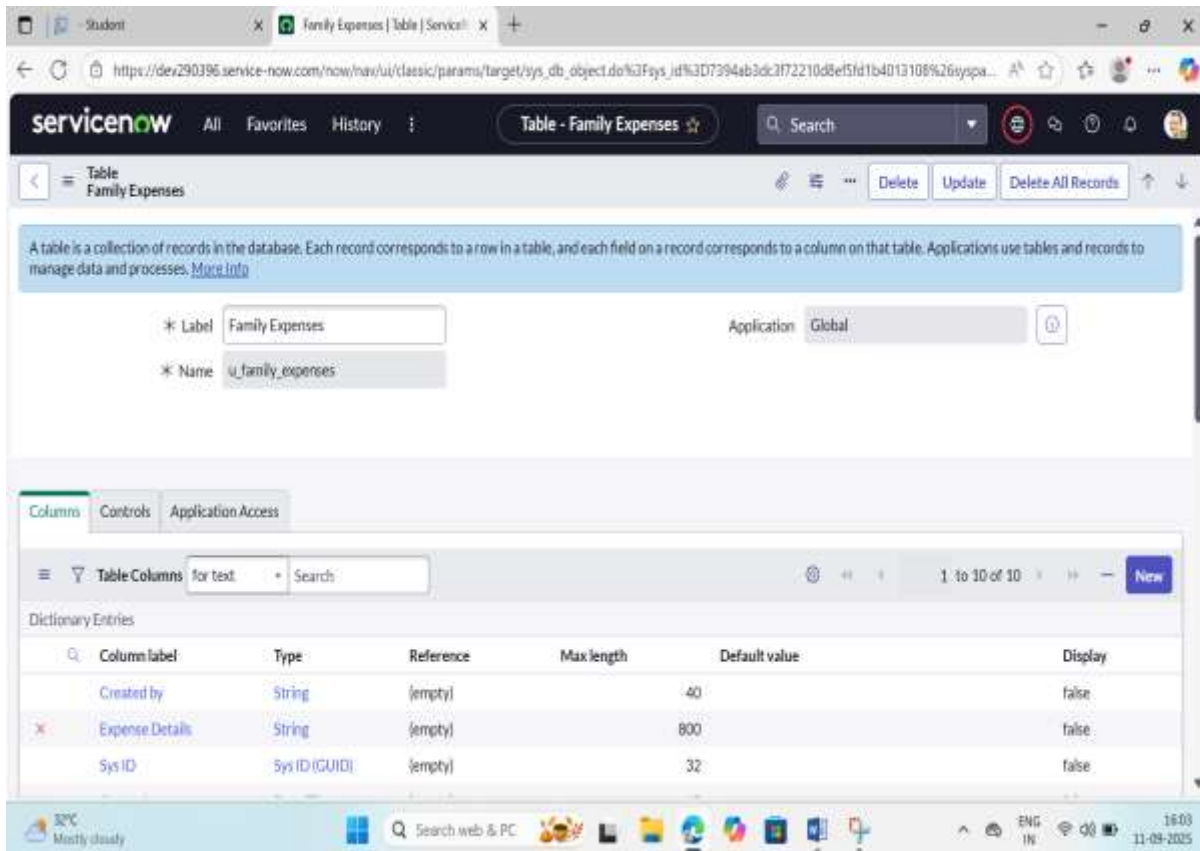
3. Enter the details as:  
Name: Family Expenses
4. Then click on Submit and Make current.



## MILESTONE 2: CREATE TABLE

### ACTIVITY 1: CREATION OF FAMILY EXPENSES TABLE

1. Go to All → search for Tables → click on New.
2. Enter the Details:
  - Label: Family Expenses
  - Name: Auto-Populated
  - New menu name: Family Expenditure



3. Go to the Header and right click there>> click on Save.

## ACTIVITY 2: CREATION OF COLUMNS FIELD

1. Near Columns Double click near insert a new row.
2. Give the details as:  
Column label: Number  
Type: String
3. Double click on insert a new row again
4. Give the details as:  
Column label: Date  
Type: Date
5. Double click on insert a new row again

6. Give the details as:  
 Column label: Amount  
 Type: Integer
7. Double click on insert a new row again
8. Give the details as:  
 Column label: Expense Details  
 Type: String  
 Max length: 800

The screenshot shows a web application interface for a table named 'Family Expenses'. At the top, there are buttons for 'Delete', 'Update', and 'Delete All Records', along with a search bar and a 'New' button. Below this is a table titled 'Dictionary Entries' with the following columns: Column label, Type, Reference, Max length, Default value, and Display. The table contains several rows, with 'Expense Details' highlighted. The 'Expense Details' row has a red 'X' icon in the first column, 'Expense Details' in the 'Column label' column, 'String' in the 'Type' column, '(empty)' in the 'Reference' column, '800' in the 'Max length' column, and 'false' in the 'Display' column. Other rows include 'Created by', 'Sys ID', 'Created', 'Amount', 'Updated by', 'Updates', 'Number', 'Date', and 'Updated'.

	Column label	Type	Reference	Max length	Default value	Display
	Created by	String	(empty)	40		false
X	Expense Details	String	(empty)	800		false
	Sys ID	Sys ID (GUID)	(empty)	32		false
	Created	Date/Time	(empty)	40		false
X	Amount	Integer	(empty)	40		false
	Updated by	String	(empty)	40		false
	Updates	Integer	(empty)	40		false
X	Number	String	(empty)	40	javascript.getNextObj(Number Padded);	false
X	Date	Date	(empty)	40		false
	Updated	Date/Time	(empty)	40		false

9. Go to the Header and right click there>> click on Save.

### ACTIVITY 3: MAKING NUMBER FIELD AN-AUTO NUMBER

1. Double click on the Number Field/Column.
2. Go down and double click on Advanced view
3. In Default Value:  
 Use dynamic default: check the box  
 Dynamic default value: Get Next Padded Number
4. Click on Update.

Reference Specification Choice List Specification Function Definition Dependent Field Calculated Value **Default Value**

Use dynamic default ☒

Dynamic default value

- Go to All → In the filter search for Number Maintenance → select Number Maintenance
- Click on New.
- Enter the below Details:  
Table: Family Expenses  
Prefix: MFE
- Click on Submit.

**servicenow** All Favorites History Workspaces **Numbers**

All > Keywords = MFE

Prefix	Number	Number of digits	Table	Updated
<input type="text" value="Search"/>	<input type="text" value="Search"/>	<input type="text" value="Search"/>	<input type="text" value="Search"/>	<input type="text" value="Search"/>
MFE	1,000	7	Family Expenses	2025-09-06 03:20:23

## ACTIVITY 4: CONFIGURE THE FORM

- Click All → Search for Family Expenses → Open Family Expenses.
- Click on New.
- Go to the Header and right click there → click on Configure → Select Form Design
- Customize or Drag Drop the form as per your requirement.

**Family Expenses [u\_family\_expenses]** 2 Column

Number   Date

Amount

**Expense Details** 1 Column

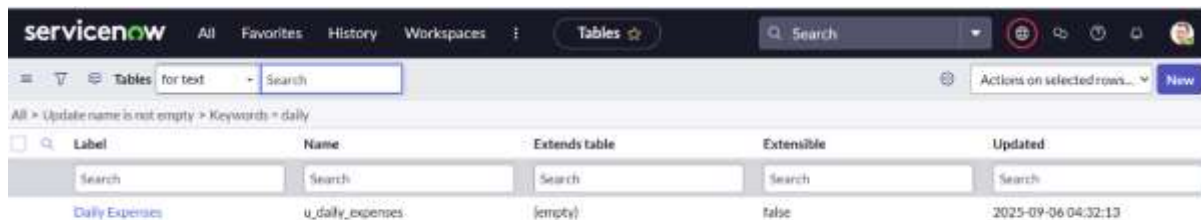
- Make Number Read-Only Field by clicking on the gear icon and checking Read-Only.

6. Make Date, Amount Mandatory Field by clicking on the gear icon and checking Mandatory.
7. Click on Save.

## **MILESTONE 3: DAILY EXPENSES TABLE**

### **ACTIVITY 1: CREATION OF TABLE (DAILY EXPENSES)**

1. Go to All → search for Tables click on New.
2. Enter the Details:  
 Label: Daily Expenses  
 Name: Auto-Populated  
 Add Module to menu: Family Expenditure



The screenshot shows the ServiceNow 'Tables' page. The table 'Daily Expenses' is listed with the following details:

Label	Name	Extends table	Extensible	Updated
Daily Expenses	u_daily_expenses	(empty)	false	2025-09-06 04:32:13

3. Go to the Header and right click there>> click on Save

### **ACTIVITY 2: CREATION OF COLUMNS (FIELD)**

1. Near Columns Double click near insert a new row.
2. Give the details as:  
 Column label: Number  
 Type: String
3. Double click on insert a new row again
4. Give the details as:  
 Column label: Date  
 Type: Date
5. Double click on insert a new row again.
6. Give the details as:  
 Column: label Expense

Type: Integer

7. Double click on insert a new row again.

8. Give the details as:

Column label: Family Member Name

Type: Reference

Max length: 800

9. Double click on insert a new row again.

10. Give the details as:

Column label: Comments

Type: String

Max length: 800

11. Go to the Header and right click there → click on Save.

The screenshot shows the ServiceNow interface for the 'Table Dictionary Entries' of the 'Daily Expenses' table. The table lists various fields with their properties. The 'Number' field is highlighted, showing its configuration as a String with a max length of 40 and a default value of 'javascript:getNextObjNumberPadded()'. The 'Family Member Name' field is also visible, configured as a Reference to the 'User' table with a max length of 32.

Column label	Type	Reference	Max length	Default value	Display
Expense	Integer	{empty}	40		false
Created	Date/Time	{empty}	40		false
Comments	String	{empty}	800		false
Sys ID	Sys ID (GUID)	{empty}	32		false
Updates	Integer	{empty}	40		false
Date	Date	{empty}	40		false
Family Member Name	Reference	User	32		false
Number	String	{empty}	40	javascript:getNextObjNumberPadded()	false
Updated by	String	{empty}	40		false
Updated	Date/Time	{empty}	40		false
Created by	String	{empty}	40		false

### ACTIVITY 3: MAKING A NUMBER FIELD AN AUTO-NUMBER



1. Double click on the Number Field/Column.
2. Go down and double click on Advanced view
3. In Default Value:  
Use dynamic default: check the box  
Dynamic default value: Get Next Padded Number
4. Click on Update.

Reference Specification | Choice List Specification | Function Definition | Dependent Field | Calculated Value | **Default Value**

Use dynamic default ☒

Dynamic default value: Get Next Padded Number

Delete Column | Update

5. Go to All → Search for Number Maintenance → select Number Maintenance
6. Click on New.
7. Enter the below Details:  
Table: Family Expenses  
Prefix: MFE

Prefix	Number	Number of digits	Table	Updated
MFE	1,000	7	Daily Expenses	2025-09-06 04:02:13

8. Click on Submit.

#### **ACTIVITY 4: CONFIGURE THE FORM**

1. Go to All → Search for Daily Expenses → Open Daily Expenses
2. Click on New
3. Go to the Header and right click there → click on Configure → Select Form Design
4. Customize or Drag Drop the form as per your requirement.
5. Make Number Read-Only Field by clicking on the gear icon and checking Read-Only
6. Make Date, Family Member Name Mandatory Field by clicking on the gear icon and checking Mandatory
7. Click on Save.

The screenshot shows a form configuration window for 'Daily Expenses [u\_daily\_expenses]'. The form is divided into two columns. The first column contains three fields: 'Number', 'Date', and 'Comments'. The second column contains two fields: 'Family Member Name' and 'Expense'. Each field has a gear icon next to it, indicating that it can be configured. The 'Number' field is marked as 'Read-Only', and 'Date' and 'Family Member Name' are marked as 'Mandatory'.

#### **MILESTONE 4 : CREATION RELATIONSHIP**

##### **ACTIVITY 1: CREATION RELATIONSHIP BEWTEEN FAMILY EXPENSES AND DAILY EXPENSES TABLE:**

1. Go to All → search for Relationships → Open Relationships.
2. Click on New.
3. Enter the details:  
Name: Daily Expenses

Applies to table: Select Family Expenses

Daily Expenses: Select Daily Expenses

4. Click Save.

## **MILESTONE 5: CONFIGURING RELATED LIST ON FAMILY EXPENSES**

1. Go to All → Search for Family Expenses → Open Family Expenses
2. Click on New
3. Go to the Header and right click there → click on Configure → Select Related Lists
4. Add Daily Expenses to the Selected Area.
5. Click on Save



## **MILESTONE 6: CREATION OF BUSINESS RULE**

### **ACTIVITY 1: CREATION BUSINESS RULE**

1. Go to All → Search for Business Rules.
2. Click on New.
3. Enter the Details:

Name: Family Expenses BR

Table: Select Daily Expenses

Business Rule  
New record

Business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a table is queried. Use business rules to automatically change values in form fields when the specified conditions are met.





Name:  


Table:  

Application:  

Active: ☒

Advanced: ☒ 


#### 4. In when to run Check Insert and Update

**When to run**  **Advanced**

Specify whether the business rule should run on **Insert** or **Update**. Use **Filter Conditions** to specify under which conditions.

When:

Order:

 **Insert** ☒


**Update** ☒

Delete ☐

Query ☐

Filter Conditions:

-- choose field --  -- oper --  -- value --

Role conditions: 

#### 5. In Advance (we write the code): Write the below code >>

```
( function execute Rule (current, previous /*null when async*/) {  
  
  var Family Expenses = new Glide Record ('  
    u_family_expenses '); Family Expense. Add Query ('u_  
    date',current.u _ date);  
  
  Family Expenses.  
  Query (); If (Family  
  Expenses.next())
```

```

{
    Family Expenses .u _amount += current .u _expense;

    Family Expenses .u _expense _details += ">" + current. u comments + ":" + "Rs."
    + current . u _expense + "/- ";

    Family Expenses. update ();
}

else

{

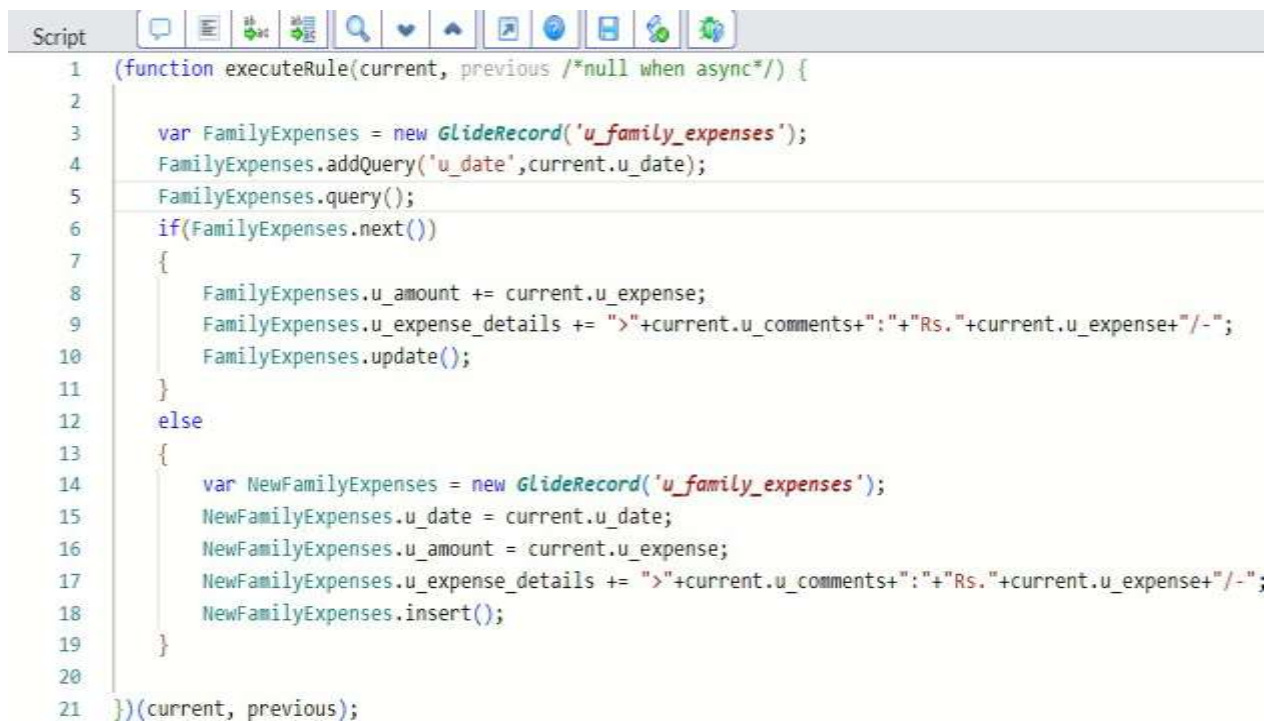
    var New Family Expenses = new Glide Record('u _ family _
    expenses'); New Family Expenses. u _ date = current. u _
    date;

    New Family Expenses. u _ amount = current. u _
    expense; New Family Expenses .u _ expense _
    details += ">" + current.u _comments + ":" + "Rs." + cu
    rrent.u _expense + "/-"; New Family Expenses.
    Insert ();

}

})( current, previous);

```



```

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

```

6. Go to the Header and right click there → click on Save.

## MILESTONE 7: CONFIGURE THE RELATIONSHIP

### ACTIVITY 1: CONFIGURE THE RELATIONSHIP

1. Go to All → Search for Relationships → Open Relationships.
2. In that, open Daily Expenses Relationship.

3. For Applies to table: Select Family Expenses.
4. In Query with: write the below Query.

```
(function refine Query (current, parent) {  
  
  // Add your code here, such as current. add Query (field,  
  value); current. add Query ('u_ date', parent. u_ date);  
  
  current. Query ();  
  
}) (current, parent);
```

5. Click on Update.

The screenshot shows the ServiceNow interface for configuring a relationship. The 'Applies to table' dropdown is set to 'Family Expenses (u\_family\_expenses)'. The 'Query with' section contains the following JavaScript code:

```
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);
```

The 'Update' button is located at the bottom left of the configuration area.

## **CONCLUSION:**

A successful Service Now project shows real business value by aligning with company goals, improving processes, and making the user experience better. This happens through expert planning, clear process mapping, smart use of configuration instead of heavy customization, and open, honest communication. With strong project management and risk control, the implementation delivers clear results, a solid return on investment, and sets the stage for future improvements.