

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: NM2025TMID07640

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

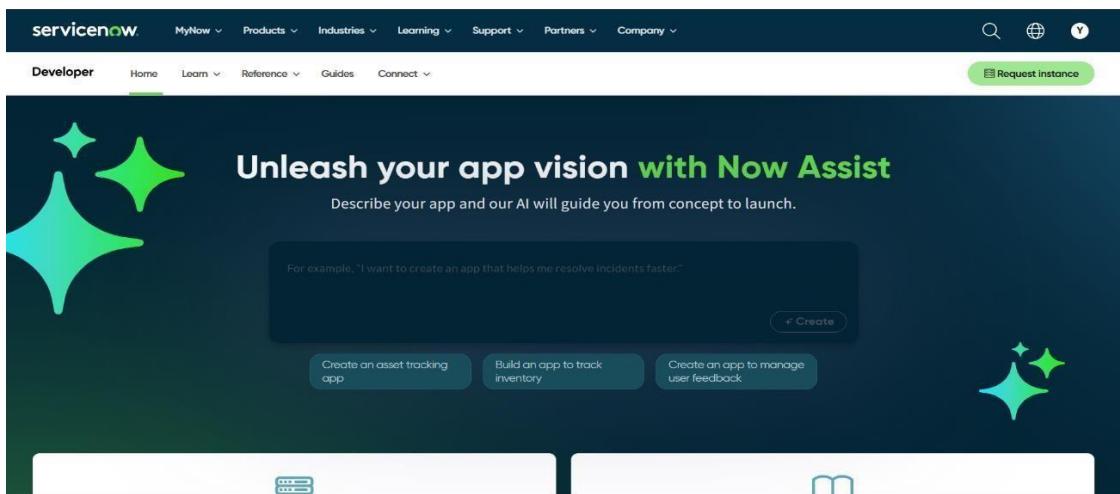
Software Requirements

- 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.

The screenshot shows the ServiceNow interface for creating a new update set. At the top, there are tabs for 'All', 'Favorites', 'History', and 'Workspaces'. The title bar says 'Update Set - Create New Update Set'. Below the title bar, there are several input fields: 'Name' (Family Expenses), 'State' (In progress), 'Parent' (with a search icon), 'Release date' (with a calendar icon), and a large 'Description' text area. At the bottom of the form are two buttons: 'Submit' and 'Submit and Make Current'. The application is set to 'Global'.

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**.

The screenshot shows the ServiceNow Table Definition screen. It displays a table with two rows. The first row has a 'Label' field containing 'Family Expenses' and a 'Name' field containing 'ui_st_family_expenches', both of which are highlighted with red boxes. The second row contains the text 'Remote Table' followed by a checked checkbox. To the right of the table, there are three checkboxes: 'Create module' (checked), 'Create mobile module' (checked), and a dropdown menu labeled 'Create view'.

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.

The screenshot shows the 'Table - Family Expenses' configuration page in ServiceNow. It displays a list of columns with their properties:

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		800		false

Buttons at the bottom include 'Delete', 'Update', and 'Delete All Records'.

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.

The screenshot shows the 'Family Expenses [u_family_expenses]' form configuration. It displays three sections:

- Section 1:** Contains fields for **Number**, **Date**, and **Amount**. The **Number** field is set to **Read Only**.
- Section 2:** Contains a single field labeled **Expense Details**.
- Section 3:** A summary section showing the total amount.

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.

The screenshot shows the 'Create module' configuration page for the 'Daily Expenses' table. It includes the following settings:

- Label:** Daily Expenses
- Name:** u.daily.expenses
- Extends table:** Family Expenditure
- Application Global:** Create module (checked), Create mobile module (checked), Application Menu (checked)
- Add module to menu:** Family Expenditure

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 screenshot shows the Oracle Forms Number Maintenance window. At the top, there are tabs for 'Choice List Specification', 'Calculated Value', and 'Default Value'. The 'Default Value' tab is selected and highlighted with a red box and a red arrow pointing to it. Below this, a note says 'The Default value specifies what value the field has when first displayed.' Under 'Default Value', there is a checkbox 'Use dynamic default' which is checked, and a dropdown menu showing 'Get Next Padded Number' with a red box and a red arrow pointing to it. At the bottom of this section are 'Delete' and 'Update' buttons, both with red arrows pointing to them.

Below this, the main configuration area has a title 'Number New record'. It includes fields for 'Table' (set to 'Daily Expenses'), 'Prefix' (set to 'MDE'), 'Number' (a numeric field set to 1.000), 'Application' (set to 'Global'), 'Number of digits' (set to 7), and a 'Submit' button at the bottom left.

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 Oracle Forms Daily Expenses Form Design window. The form has two sections. The top section is titled '# Daily Expenses [x_daily_expenses]' and contains fields for 'Number' (read-only, indicated by a lock icon), 'Date', 'Family Member Name', and 'Expense'. The bottom section is titled '# Comments' and contains a single text input field. Both sections have a '2 Column' layout option at the top right.

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 ServiceNow interface for creating a new relationship. The top navigation bar includes 'servicenow', 'All', 'Favorites', 'History', 'Workspaces', and a search bar. The main title is 'Relationship - New Record'. The 'Name' field is set to 'Daily Expenses'. The 'Applies to table' dropdown is set to 'Family Expenses [u_family_expenses]'. The 'Queries from table' dropdown is set to 'None'. A script editor window contains the following code:

```
Query with (function refineQuery(current, parent) {  
    // Add your code here, such as current.addQuery(field, value);  
})(current, parent);
```

At the bottom right, 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 ServiceNow interface for creating a new business rule. The top navigation bar includes 'servicenow', 'All', 'Favorites', 'History', 'Workspaces', and a search bar. The main title is 'Relationship - Daily Expenses'. The 'Name' field is set to 'Daily Expenses'. The 'Applies to table' dropdown is set to 'Family Expenses [u_family_expenses]' (highlighted with a red arrow). The 'Queries from table' dropdown is set to 'Daily Expenses [u_daily_expenses]'. A script editor window contains the following code, which is highlighted with a red box:

```
Query with (function refineQuery(current, parent) {  
    // Add your code here, such as current.addQuery(field, value);  
    current.addQuery('u_date',parent.u_date);  
    current.query();  
})(current, parent);
```

At the bottom left, there are 'Update' and 'Delete' buttons, both of which have red arrows pointing to them.

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.

The screenshot shows the ServiceNow interface for a relationship named "Daily Expenses". The "Name" field is set to "Daily Expenses". Under "Applies to table", "Family Expenses [u_auto_populated]" is selected. Under "Queries from table", "Daily Expenses [u_daily_expenses]" is selected. The "Query with" section contains the following ECMAScript code:

```
Turn on ECMAScript 2021 (ES12) mode ⓘ
1 (function refinequery(current, parent) {
2   // Add your code here, such as current.addQuery(field, value);
3   current.addQuery('u_date',parent.u_date);
4   current.query();
5 }
6 })(current, parent);
```

Below the code, there are buttons for "Run Query Diagnostics", "Update", and "Delete". A note at the bottom states: "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."

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.

The screenshot shows the ServiceNow interface for a business rule named "Business Rule - Family Expenses BR". The "When to run" tab is selected. The "Actions" tab is also visible. The "Advanced" tab is selected. The "Condition" field is empty. The "Script" field contains the following ECMAScript code:

```
Turn on ECMAScript 2021 (ES12) mode ⓘ
1 (function executeRule(current, previous /*null when async*/) {
2   // Add your code here
3
4   var FamilyExpenses = new GlideRecord('u_family_expenses');
5
6   FamilyExpenses.addQuery('u_date',current.u_date);
7
8   FamilyExpenses.query();
9
10  if(FamilyExpenses.next())
11  {
12
13    FamilyExpenses.u_amount += current.u_expense;
14
15    FamilyExpenses.u_expense_details += ">" + current.u_comments + ":" + rs + current.u_expense + "-";
16
17    FamilyExpenses.update();
18
19  }
20
21
22  else
23}
```

Conclusion

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

