

Project: Calculating Family Expenses Using ServiceNow

Objective:

Develop a comprehensive system on ServiceNow to track, categorize, and manage family expenses efficiently, enabling informed financial decisions. The project aims to develop a comprehensive expense calculation system using ServiceNow. This system will enable users to track and manage family expenses efficiently. It will include features such as expense categorization, budget setting, real-time tracking, and reporting capabilities. Utilizing ServiceNow's robust platform, the project will ensure seamless integration, user-friendly interface, and scalability to accommodate varying family sizes and financial complexities. The end goal is to empower users with the tools they need to make informed financial decisions and promote financial well-being within the family unit.

Step 1: Setting up Your ServiceNow Instance

1. Sign Up:

- Go to [ServiceNow Developer Site](#) and sign up for a developer account.

2. Request Personal Developer Instance (PDI):

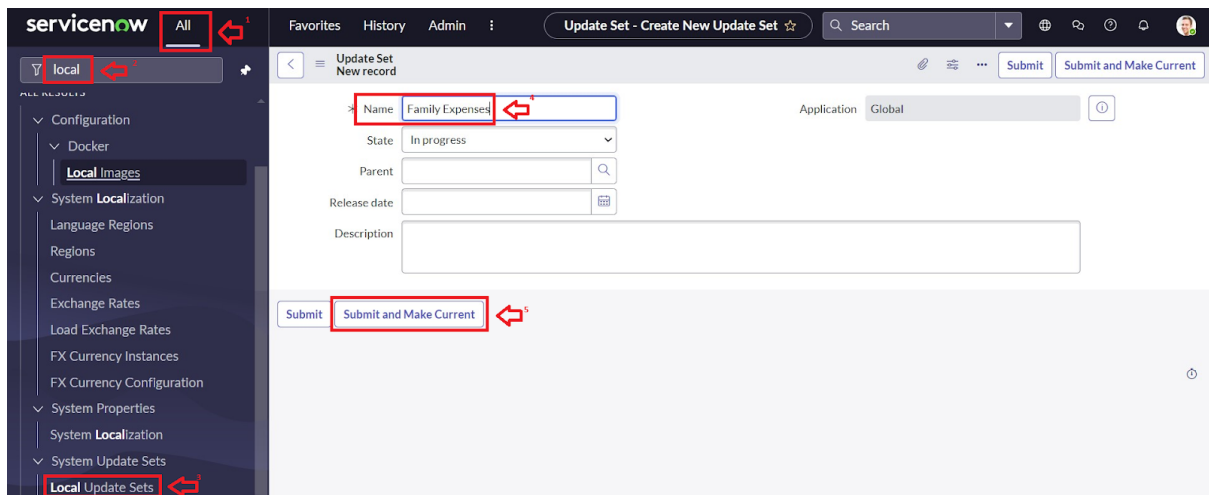
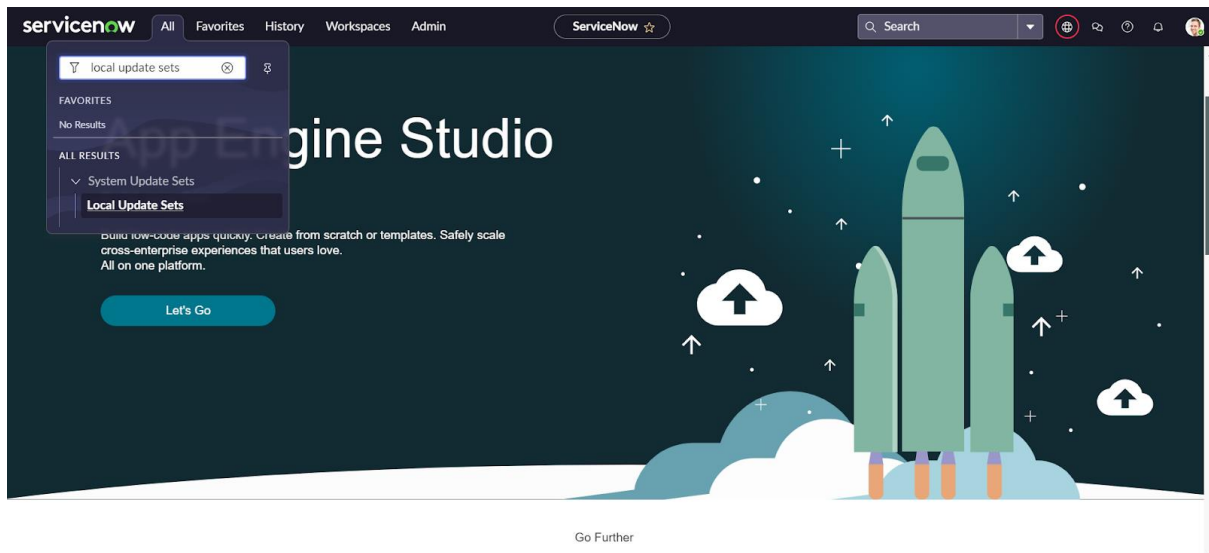
- After logging in, navigate to **Personal Developer Instance** section.
- Click **Request Instance** and fill out the form.
- Wait for the confirmation email with your instance details.

3. Log In:

- Use the credentials provided to log into your personal ServiceNow instance.
-

Step 2: Create a New Update Set

1. Navigate to **All > Local Update Sets**.
2. Click **New** and create an update set with the following details:
 - **Name:** Family Expenses
3. Submit and make this update set the current active update set.



Step 3: Create Family Expenses Table

1. Navigate to **All > Tables**, then click **New** to create a new table.
2. Fill in the details:
 - **Label:** Family Expenses
 - **Name:** Auto-populated (will default)
 - **New menu name:** Family Expenditure
3. Save the table.

* Label
 * Name

Application ⓘ

Remote Table ☒

Create module ☒

Create mobile module ☒

Add module to menu

New menu name

Columns Controls Application Access

Table Columns Search

Dictionary Entries

	Column label	Type	Reference	Max length	Default value	Display
<input type="checkbox"/> <input type="checkbox"/>	Number	String				false
<input type="checkbox"/> <input type="checkbox"/>	Date	Date				false
<input type="checkbox"/> <input type="checkbox"/>	Amount	Integer				false

Step 4: Add Columns (Fields) to Family Expenses Table

Add the following fields:

- **Number** (String)
- **Date** (Date)
- **Amount** (Integer)
- **Expense Details** (String, max length 800)

Save your changes.

Columns Controls Application Access

Table Columns Search

Dictionary Entries

	Column label	Type	Reference	Max length	Default value	Display
<input type="checkbox"/> <input type="checkbox"/>	Number	String				false
<input type="checkbox"/> <input type="checkbox"/>	Date	Date				false
<input type="checkbox"/> <input type="checkbox"/>	Amount	Integer				false
<input type="checkbox"/> <input type="checkbox"/>	Expense Details	String		800		false

Insert a new row...

Step 5: Make Number Field Auto-Generated

1. Open the **Number** field record.
2. Scroll to **Advanced View**.
3. Check **Use dynamic default** and set **Dynamic default value** to **Get Next Padded Number**.
4. Update the field.

Choice List Specification | Calculated Value | **Default Value**

The **Default value** specifies what value the field has when first displayed.

Use dynamic default ☒

Dynamic default value: Get Next Padded Number

Delete Column | **Update**

5. Navigate to **Number Maintenance**:

- Create a new record with:
 - **Table:** Family Expenses
 - **Prefix:** MFE
- Submit.

< ≡ Number MFE

Table: Family Expenses

Prefix: MFE

* Number: 1,000

Application: Global

Number of digits: 7

Update Delete

Step 6: Configure the Family Expenses Form

1. Go to **All > Family Expenses**.
2. Click **New** to open the form.
3. Right-click on the form header, select **Configure > Form Design**.

4. Customize the form layout as needed (drag & drop fields).
5. Set:
 - **Number** field as **Read-Only**.
 - **Date** and **Amount** fields as **Mandatory**.
6. Save the form.

Family Expenses [u_family_expenses] 2 Column

Number Date Amount

Expense Details 1 Column

Step 7: Create Daily Expenses Table

1. Navigate to **All > Tables** and click **New**.
2. Enter details:
 - **Label:** Daily Expenses
 - **Name:** Auto-populated
 - Add module to **Family Expenditure** menu.
3. Save.

* Label Daily Expenses 1

* Name u_daily_expenses 2

Extends table

Application Global

Create module ☒

Create mobile module ☒

Add module to menu Family Expenditure 3

Step 8: Add Columns to Daily Expenses Table

Add the following fields:

- **Number** (String)
- **Date** (Date)
- **Expense** (Integer)
- **Family Member Name** (Reference to Family Expenses table)
- **Comments** (String, max length 800)

Save the changes.

Step 9: Make Daily Expenses Number Field Auto-Generated

Repeat the steps from Step 5 for the **Daily Expenses** table:

- Dynamic default: **Get Next Padded Number**
- Create Number Maintenance record with prefix as **MFE** for the Daily Expenses table.

Choice List Specification | Calculated Value | **Default Value** ¹

The **Default value** specifies what value the field has when first displayed.

Use dynamic default ☒ ²

Dynamic default value: **Get Next Padded Number** ³

Delete Column | **Update** ⁴

Step 10: Configure Daily Expenses Form

1. Go to **All > Daily Expenses**, click **New**.
2. Configure the form design:
 - Set **Number** as **Read-Only**.
 - Make **Date** and **Family Member Name** mandatory.
3. Save the form.

Step 11: Create Relationship Between Tables

1. Go to **All > Relationships**, click **New**.
2. Define relationship:
 - **Name:** Daily Expenses
 - **Applies to table:** Family Expenses
 - **Related Table:** Daily Expenses
3. Save.

Step 12: Add Related List to Family Expenses Form

1. Open the **Family Expenses** table form.
2. Right-click on the header, select **Configure > Related Lists**.
3. Add **Daily Expenses** as a related list.
4. Save.



Step 13: Create Business Rule for Expense Aggregation

1. Navigate to **Business Rules**, click **New**.
2. Enter details:
 - **Name:** Family Expenses BR
 - **Table:** Daily Expenses
 - Check **Advanced**

Business Rule
New record

ss 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 Family Expenses BR

Table Daily Expenses [u.daily_expenses]

Application Global

Active ☒

Advanced ☒

- When to run: **Insert and Update**

When to run

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

When before

Order 100

Insert ☒

Update ☒

Delete ☐

Query ☐

Filter Conditions Add Filter Condition Add "OR" Clause

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

Role conditions

3. Paste this code in the **Advanced** script section:

```

Script
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);

```

```

(function executeRule(current, previous /*null when async*/) {
    var FamilyExpenses = new GlideRecord('u_family_expenses');
    FamilyExpenses.addQuery('u_date', current.u_date);
    FamilyExpenses.query();
    if (FamilyExpenses.next()) {
        FamilyExpenses.u_amount += current.u_expense;
        FamilyExpenses.u_expense_details += ">" + current.u_comments + ": Rs." +
current.u_expense + "/-";
        FamilyExpenses.update();
    } else {
        var NewFamilyExpenses = new GlideRecord('u_family_expenses');
        NewFamilyExpenses.u_date = current.u_date;
        NewFamilyExpenses.u_amount = current.u_expense;
        NewFamilyExpenses.u_expense_details = ">" + current.u_comments + ": Rs." +
current.u_expense + "/-";
        NewFamilyExpenses.insert();
    }
})(current, previous);

```

4. Save the Business Rule.

Step 14: Refine Relationship Query

1. Go to **Relationships**, open **Daily Expenses Relationship**.
2. Set **Applies to table**: Family Expenses.
3. In **Query with** script, add:

```
(function refineQuery(current, parent) {
  current.addQuery('u_date', parent.u_date);
  current.query();
})(current, parent);
```

4. Update.

Relationship
Daily Expenses

Name:

Advanced: ☐

Application:

Applies to table: 1

Queries from table:

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

```

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);

```

2

3

Conclusion

By implementing this project, your family can:

- Submit and track daily expenses.
- Aggregate expenses on a daily basis.
- View summarized family expense data.
- Use a centralized and easy-to-use system.
- Make better financial decisions with real-time data.

This system leverages ServiceNow's platform for scalability, user-friendly interface, and automation.