Project documentation: Calculating the family expenses using ServiceNow

Team Id: NM2025TMID17845

Team Members:

Team Leader: VENKAT BOSE B

Team Member 1: THIRAVIYAN P

Team Member 2: YUVARAJ KUMAR V

Team Member 3: VENKATESH M

Problem Statement:

Manul tracking of family expenses is inefficient and inaccurate, making budget management difficult.

Objective:

To build a ServiceNow solution to automate expense tracking, categorization, and reporting for better financial planning.

Skills:

ServiceNow, Data Structures, Database Management.

Introduction

This guide explains how to build a simple custom app in ServiceNow to track and calculate family expenses. The app helps manage household finances by categorizing daily spending and summarizing totals. Key steps include creating tables, setting relationships, and adding business logic.

2. Project Setup

This section details the initial steps taken to prepare the ServiceNow instance for development.

Step 2.1: Sign Up & Create Developer Instance

Create a personal developer instance (PDI) from the ServiceNow Developer Program.

This provides a safe environment for testing and development.

1. Step 2.2: Create Update Set

In Filter Navigator, search Update Sets → click New.

2. Fill details:

Name: Family Expenses App

Description: Customizations for family expenses

3. Click Submit and Make Current to activate.

3. Creating Custom Tables

We created two custom tables to store the financial data: one for summarizing family expenses
and another for logging individual daily expenditures.

Step 3.1: Creation of the Family Expenses Table

This table serves as the primary record for tracking total monthly or weekly expenses.

1. In the Filter Navigator, type Tables and click on Tables under System Definition.

2. Click New to create a new table.

3. Enter the following details:

- **Label:** Family Expenses
- **Name:** x_<your scope>_ family expenses (This is auto-populated)
- Add module to menu: Family Expenditure

4. Click Submit to create the tab

Here's a visual of the process for creating the Family Expenses Table:

This table

includes fields for different expense categories, such as Food, Rent, and Travel, allowing for a
detailed breakdown of costs.

Step 3.2: Creation of the Daily Expenses Table

This table stores a list of all individual expenses incurred on a day-to-day basis.

1. In the Filter Navigator, type Tables and click on Tables under **System Definition`.

2. Click New to create a new table.

3. Enter the following details:

○ Label: Daily Expenses

○ Name: x_<your_ scope>_daily_ expenses (This is auto-populated)

○ Add module to menu: Daily Expenses

4. Click Submit to create the table.

Here's how to create the Daily Expenses Table:

Here's how to create the Daily Expenses Table:

Here's how to create the Daily Expenses Table:

Here's how to create the Daily Expenses Table:

This table includes fields for amount, category, and short description, providing the granular

data needed to calculate the totals.

4. Establishing Relationships

To link the two tables and ensure data integrity, we created a relationship between Daily
Expenses and Family Expenses. This relationship allows us to view all related daily expense
records directly from a single Family Expenses record.

Step 4.1: Creating the Relationship

1. In the Filter Navigator, type Relationships and click on Relationships under System
Definition.

2. Click New.

3. Configure the relationship as follows: ○ Name: Daily Expenses

○ Applies to table: Family Expenses

○ Queries from table: Daily Expenses

○ Query with: current. Add Query ('family _expense', parent. sys_ id); (This ensures that

only records related to the parent Family Expenses record are displayed).

4. Click Submit to save the relationship.

Here's an image depicting the creation of relationship:

Step 4.2: Configuring the Related List

To make the relationship visible on the Family Expenses form, we configured a related list.

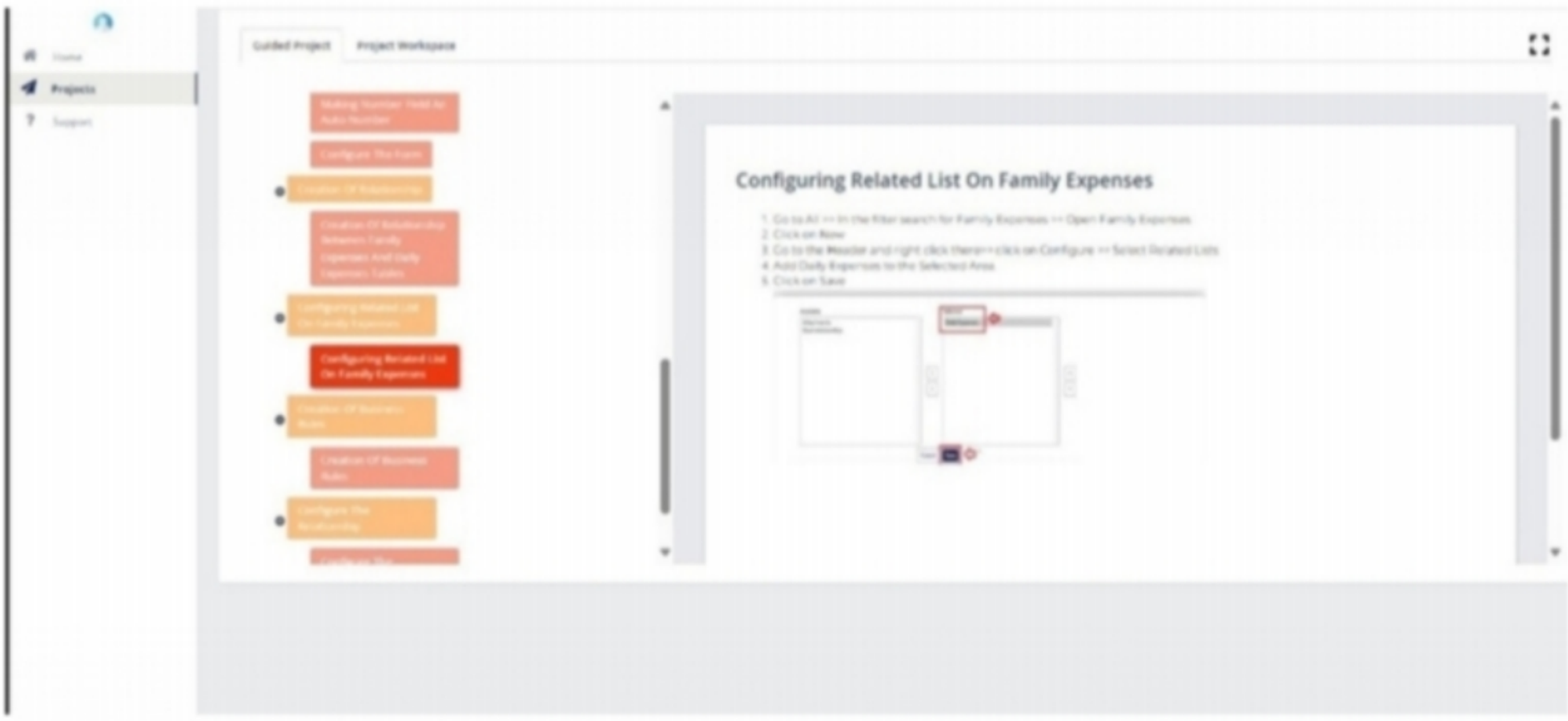
1. Navigate to a Family Expenses record.

2. Right-click the form header and select Configure > Related Lists.

3. Move Daily Expenses -> Family Expenses from the Available slush bucket to the
Selected slushbucket.

4. Click Save.

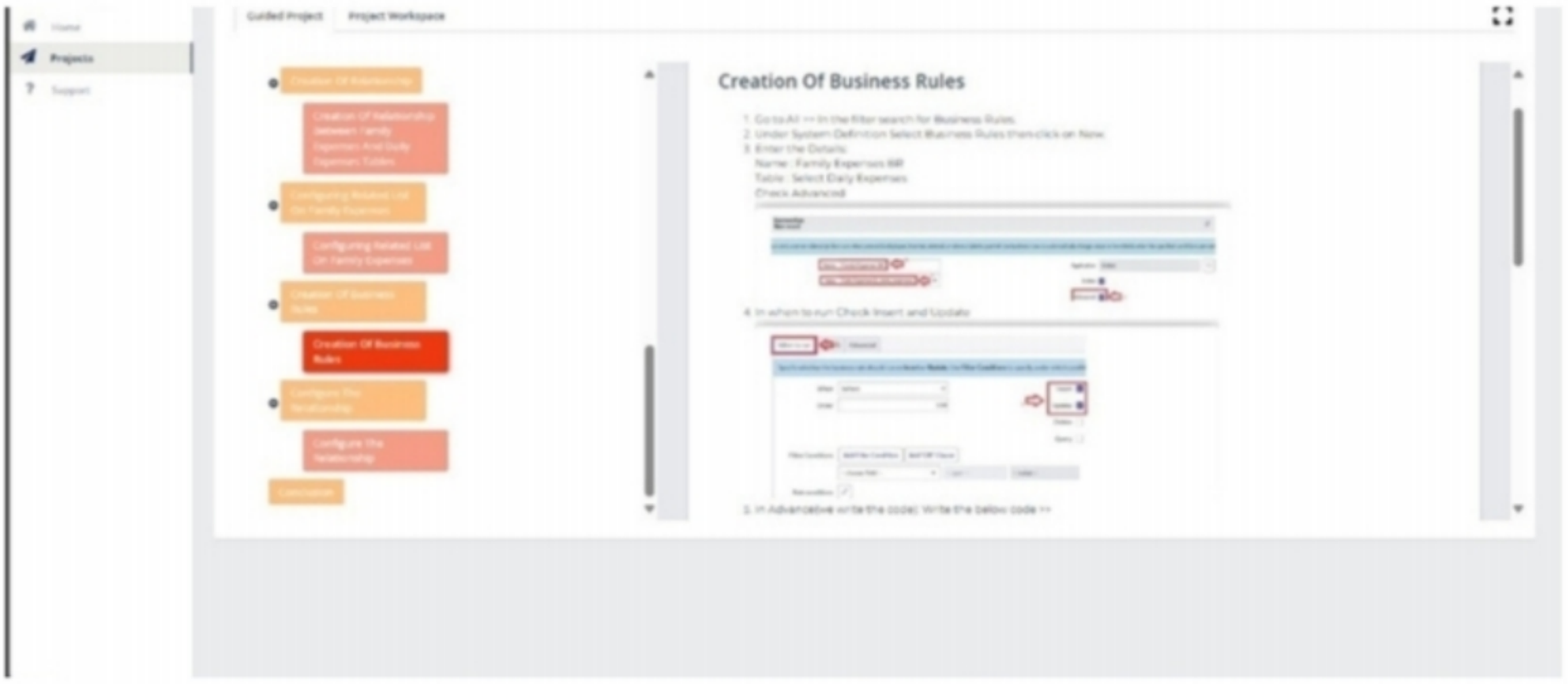
This action adds a related list tab at the bottom of the Family Expenses form, allowing users to
see and manage all daily expenses associated with that particular record. Here's an image
showing how to configure the Related List.



Implementing Business Rules:

Business rules are server-side scripts that run when a record is inserted, updated, or deleted.
We used a business rule to automatically calculate and update the total expenses in the Family
Expenses table whenever a new Daily Expenses record is created.

5.1: Creating the business Rule



5.2: Adding the Script

1. In the Filter Navigator, type Business Rules and click on Business Rules under System Definition.

2. Click New.

3. Enter the following details:

○ Name: Update Family Expenses

○ Table: Daily Expenses

○ When to run: Select after for the Insert and Update operations.

○ Filter Condition: None, as it applies to all records.

○ Advanced: Check this box to enable the scripting section. Here's

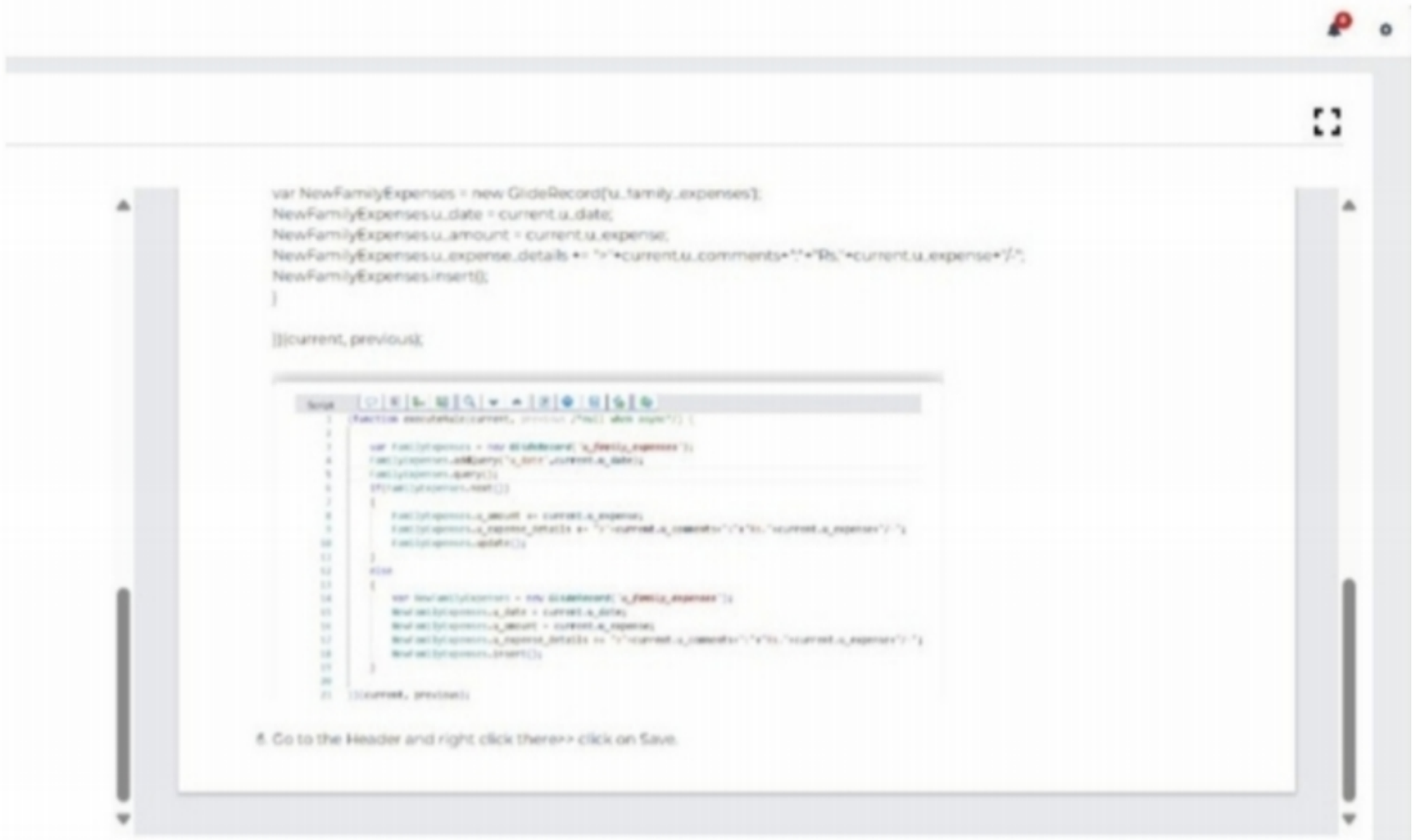
an image showing the creation of the business rule:

The following script was placed in the Advanced tab to execute the calculation logic.

```
(function executeRule (current,previous /nullwhen async/){
 var family Expenses = new
Glide Record ('x_<your _scope>_family_ expenses');
Family Expenses. Get (current. family_ expense);
if (family Expenses. Is Valid Record ())
{ // Increment the total expense
Amount
family Expenses. Set Value ('total_ amount',
```

```
family Expenses. Get Value ('total_ amount') + current. amount);
// Update the category fields based on the daily expense
category var category =
current. cate gory. Get Display Value (); if (category ==
'Food') { family Expenses. Set Value ('food',
Family Expenses. get Value('food') + current. amount); }
else if (category == 'Rent') {
family Expenses. Set Value ('rent', family Expenses. Get Value('rent')
+ current. Amount ); } else if (category == 'Travel') {
Family Expenses. Set Value ('travel',
Family Expenses. Get Value('travel') + current. amount);
 }
 Family Expenses. update();
 }
})(current, previous);
```

Here is an image showing the script for the business rule:



performs the following actions:

1. It retrieves the Family Expenses record associated with the current Daily Expenses record.
2. It adds the amount of the daily expense to the total_ amount of the family expense record.
3. It then updates the specific category field (e.g., food, rent, travel) based on the category
selected for the daily expense.
4. Finally, it saves the changes to the Family Expenses record, ensuring the totals are always up-to-date.

6.Conclusion: By completing these steps, we developed a straightforward yet effectiv application on
the ServiceNow platform. The solution streamlines family expense tracking through the use of custom tables, defined relationships, and an automated business rule that ensures real-time updates. This project highlights how ServiceNow’s low-code environment can

be utilized to quickly design and deploy business-ready applications with both efficiency and flexibility.