CALCULATING FAMILY EXPENSES USING SERVICE NOW

Team ID : NM2025TMID18260

Team Leader : GOKUL KUMAR L

Team member 1 : DINESHMARAN K

Team member 2 : DIVYESH C

Team member 3 : GOPINATH T

Problem Statement: Unorganized manual expense tracking.

Objective: Automate family expenses in ServiceNow.

Skills: ServiceNow, tables, relationships, business rules, reports.

TASK INITIATION

Milestone 1: Users

Topic Description

1. A custom Family Expenses table is created in ServiceNow to store expense details.

2. Each record captures fields like category, amount, date, and payer.

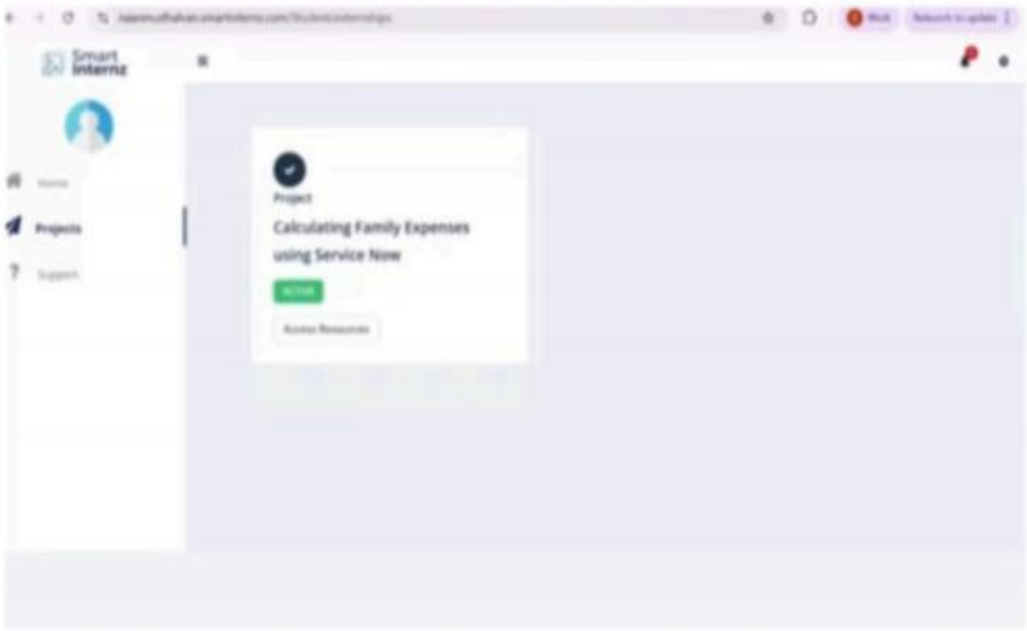
3. Relationships are configured to link expenses with family members.

4. Business rules automate calculations, such as monthly totals.

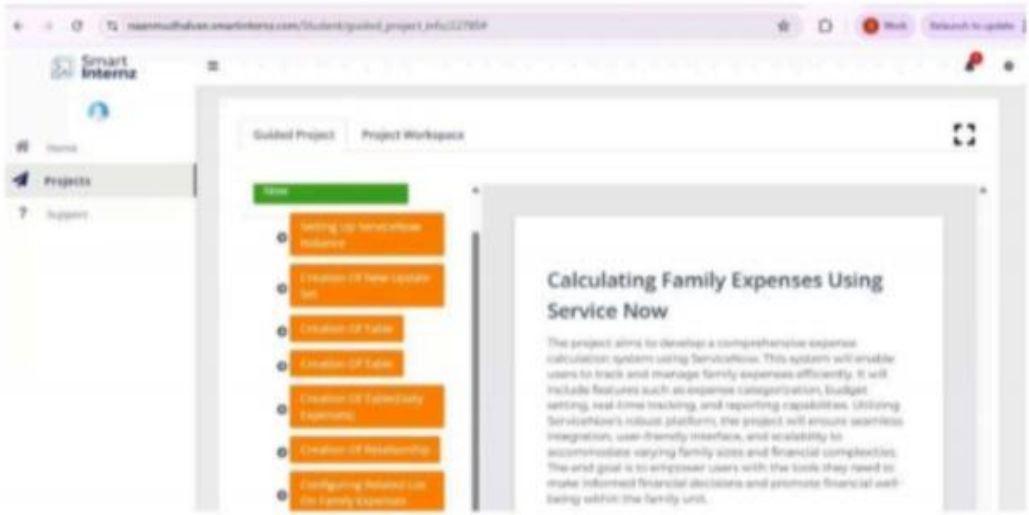
5. Reports and dashboards provide a clear breakdown of spending.

6. This setup ensures organized, transparent, and trackable expense management.

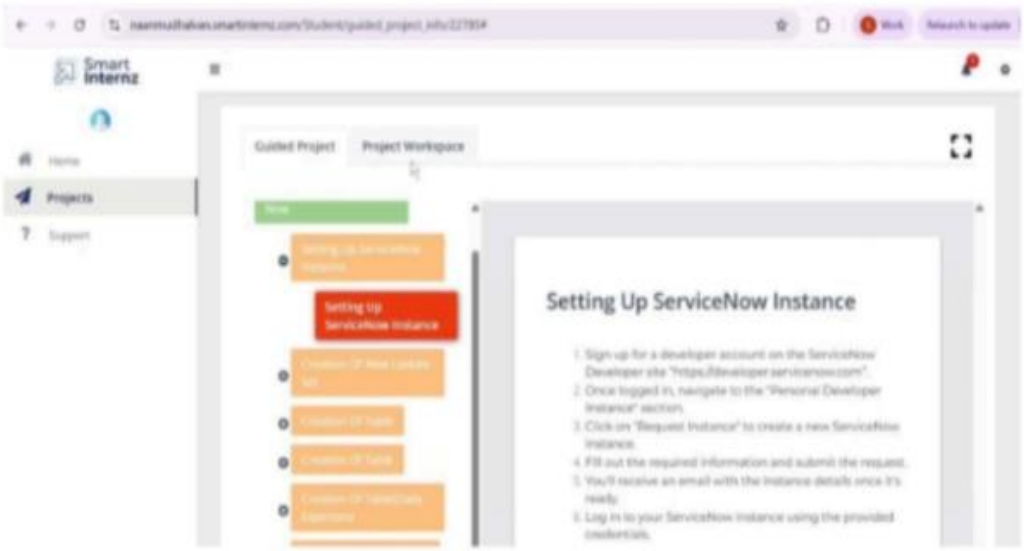
Accessing ServiceNow Instance



Calculating Family Expenses in ServiceNow



Activity 1: Setting Up ServiceNow Instance



SETTING UP SERVICENOW INSTANCE :

Create a ServiceNow Developer Account

- ❖ Go to ServiceNow Developer Portal.
- ❖ Click Sign Up (or Log in if you already have an account).

- ❖ Fill in your details (name, email, etc.).
- ❖ Verify your email to ac vate the account.

Request a Personal Developer Instance (PDI)

- After logging in, go to Manage → Instance.
- Click Request Instance.
- Choose the latest ServiceNow release (ex: Washington DC, Vancouver, etc.).
- Select YES for a new instance (if you don't have one yet).
- Wait a few minutes — you'll get your own ServiceNow subdomain, like:

Access Your Instance

- Once provisioned, you'll see login details:
- Username: admin
- Password: (provided or reset via email).
- Log in to your instance using those credentials.
- Change your password on first login.

Navigate to Update Sets

- Log in to your ServiceNow instance.
- In the Application Navigator (left-hand filter), type:Update Sets
- Go to: System Update Sets → Local Update Sets.

Create a New Update Set

- Click New (top-right).
- Fill in the fields:
- Name → Give a clear name (e.g., Incident Customization Aug 2025).
- Description → (Optional) Explain what's inside (e.g., UI Policy + Business Rule for Incident form).
- State → Keep as In Progress (default).
- Click Submit.

Select the Active Update Set

- After creation, go back to Local Update Sets.
- Open your new update set record.
- At the top-right, click Make Active.

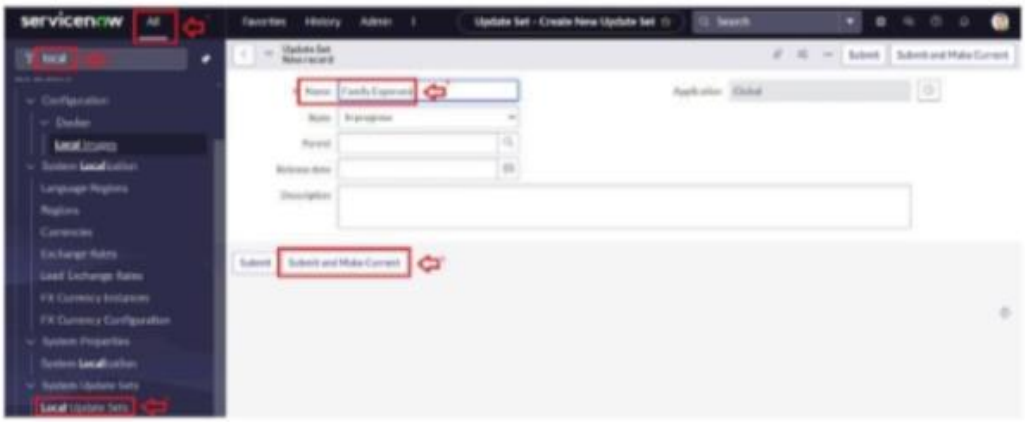
Finalize & Export

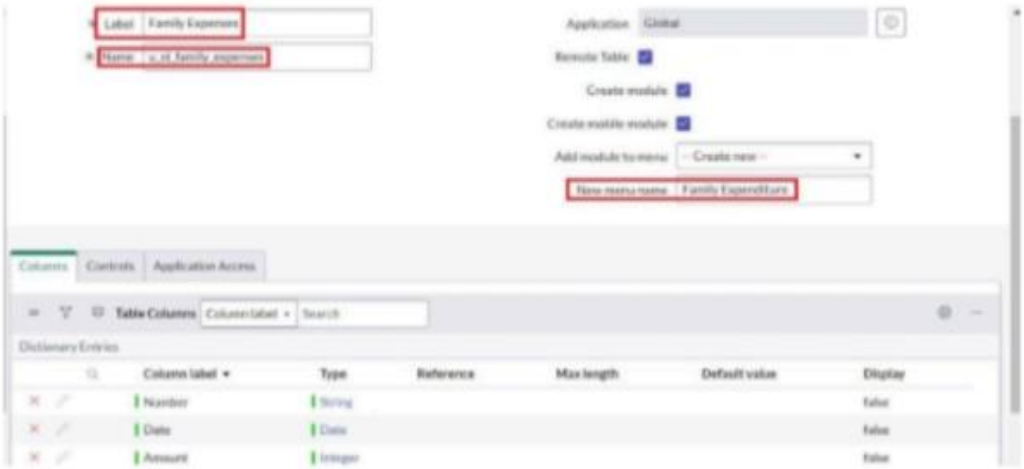
- Once done with your changes, set State → Complete.
- To move it to another instance:
- Go to Retrieved Update Sets in the target instance.
- Import XML of your update set.
- Preview, then Commit.

Activity 2: Creation of Table

Navigate to Table Creation

- Log in to your ServiceNow instance.
- In the Application Navigator, type:
  - Tables
- Go to System Definition → Tables.
- Click New.





Fill in Table Details

- You'll see a Table form:
- Label → Human-readable name (e.g., Student Records).
- Name → Auto-filled (e.g., u_student_records).
- Application → Choose the application scope where this table belongs.
- Extends Table →
- Leave blank for a brand-new table.
- Or extend an existing one (e.g., Task if you want task-like behavior).
- Auto-number → Optionally generate unique IDs (e.g., STU0001, STU0002).
- Add Notes/Attachments → Enable if needed.
- Click Submit.

Add Fields to the Table

- After creation, open your new table record.
- Scroll down to the Columns section.
- Click New to add fields:
- Column Label (e.g., Student Name).
- Column Name auto-fills (e.g., u_student_name).
- Type → Choose field type (String, Integer, Date, Reference, Choice, etc.).
- Length/Attributes → Define as needed.
- Click Submit.

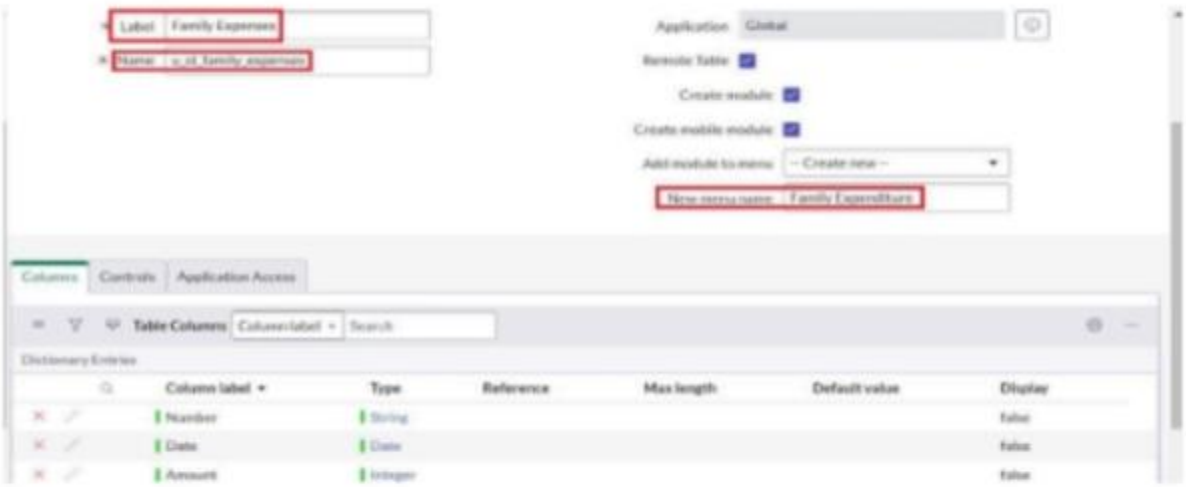
Configure Forms & Lists

❖ Go to Form Layout to arrange fields in your table's form view.
❖ Go to List Layout to choose which fields display in list view.

Test the Table

❖ Type your table's label in the Application Navigator (e.g., Student Records).
❖ Open it → Click New → Create a sample record.
❖ Save → Check if fields, form, and list views look correct.

Activity 3: Creation of Table(Daily Expenses)



Label: Daily Expenses

❖ Name: Auto-fills → u_daily_expenses
❖ Application: Select your custom app (or Global if testing).

- ❖ Extends Table: Leave blank (unless you want to inherit from another table).
- ❖ Optionally check Auto-number → e.g., Prefix EXP with length 6 (EXP000001).
- ❖ Click Submit.

Activity 4: Creation of Relationship

Reference Field (Most Common)

- ❖ Go to your custom table (e.g., Daily Expenses).
- ❖ Add a new field of type Reference.
- ❖ Choose the table you want to relate to (e.g., User table).
- ❖ This creates a one-to-many relationship (one user can have many expenses).

Dictionary Relationship (Parent-Child)

- ❖ Navigate: System Definition → Tables & Columns → Dictionary.
- ❖ Create a new relationship record.
- ❖ Define Parent Table and Child Table.
- ❖ This is used when you want to define a hierarchy (e.g., Department → User → Expense).

Many-to-Many Relationship

- ❖ Use a Many-to-Man (M2M) table.
- ❖ Create a new table with two reference fields:
  - o One referencing Table A (e.g., Projects).
  - o One referencing Table B (e.g., Employees).
- ❖ This allows multiple employees to belong to multiple projects.

Database Views (for Reporting Relationships)

- ❖ Navigate: System Definition → Database Views.
- ❖ Create a new view that combines two or more tables using their relationship fields.
- ❖ Useful for reports and queries, not for direct data entry.

Activity 5: Configuring related list on family expenses

Make Sure Relationship Exists

- Open your Family Expenses table.
- Add a Reference field pointing to the Family table (example: family field, referencing x_app_family).
- This ensures each expense record knows which family it belongs to.

Enable Related List

- Navigate to System UI → Related Lists.
- Click New.

Fill in:

  - Name: Family → Family Expenses
  - Applies to Table: Family (parent table)
  - Query from Table: Family Expenses (child table)
  - Field: the reference field you created (e.g., family)
- Save

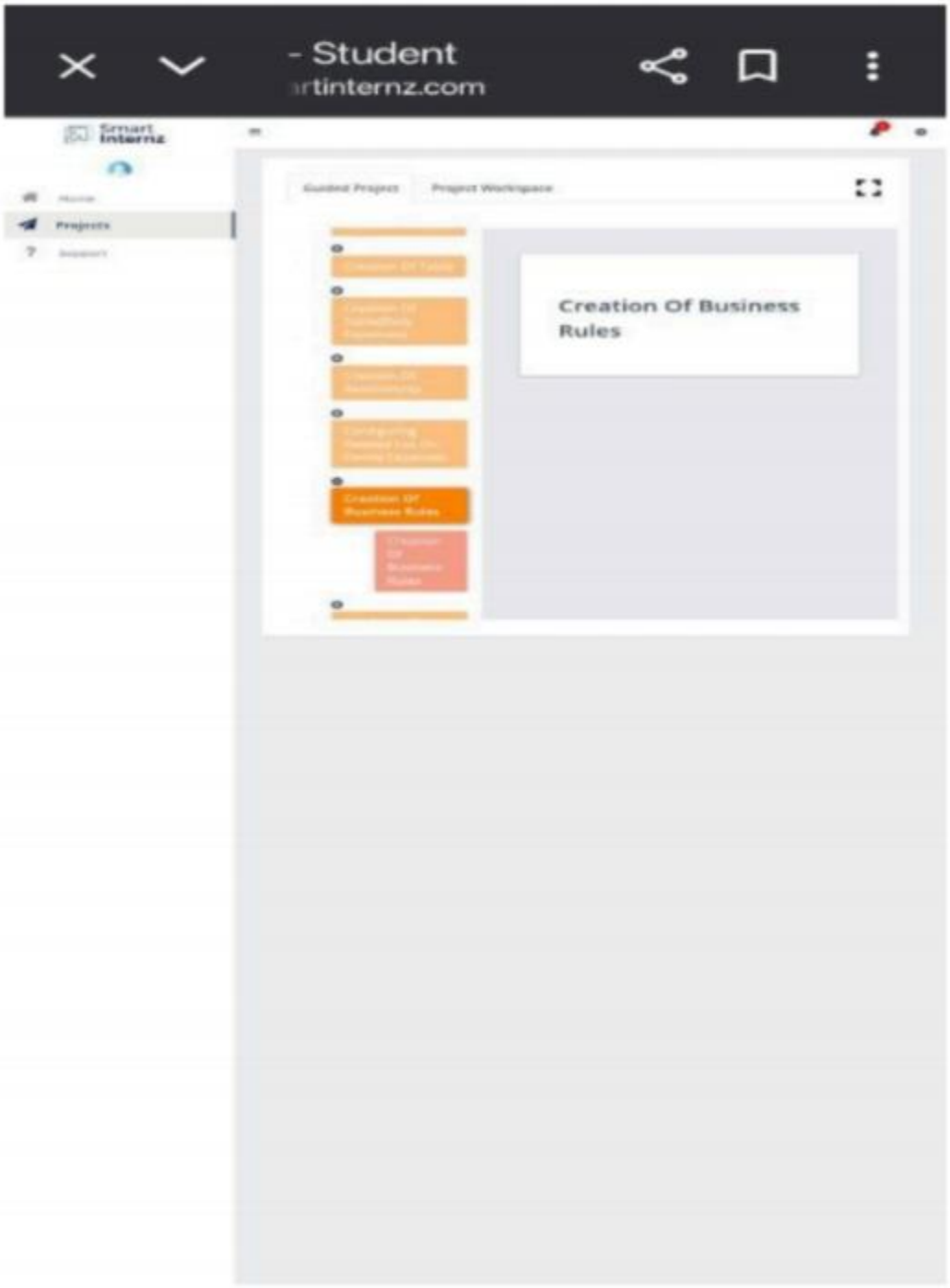
Configure Form Layout

- Open the Family table form.
- Right-click the form header → Configure → Related Lists.
- Add Family Expenses to the available related lists.
- Save.

Test

- Go to any Family record.
- Scroll down → you should see the Family Expenses related list.
- Add new expenses directly from there.

Activity 6: Creation of Business Rules



Navigate to Business Rules

- Go to: System Definition → Business Rules.
- Click New.

4. In when to run Check Insert and Update

Fill Basic Information

- Name: Give a meaningful name (e.g., Set Default Status).
- Table: Select the table it applies to (e.g., Daily Expenses or Family Expenses).
- Active: ✓ (checked).
- Advanced: check this if you want to use scripts.

When to Run

- When: Choose before, after, or async.
- Before → modify data before saving.
- After → run logic after record is saved.
- Async → background processing (performance-friendly).
- Insert / Update / Delete / Query: pick the conditions.

Condition

- Define when it should trigger (example: status is empty).

Script (if needed)

Example: set a default value when an expense is created:

```
(function executeRule(current, previous /null when async/) {
  // If no status is set, default to "Pending"
  if (!current.status) {
    current.status = "Pending";
  }
})(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);
```

Go to the Header and Right click their** click on save

Activity 7: Configure the Relationship

Decide the Relationship Type

- One-to-Many

Example: One Family → Many Expenses

Implemented by adding a Reference field in the child table (Family Expenses) pointing to the parent table (Family).

- Many-to-Many

Example: Many Users → Many Families (if needed).

Implemented by creating a Many-to-Many (M2M) table with two reference fields.

Create Reference Field (One-to-Many)

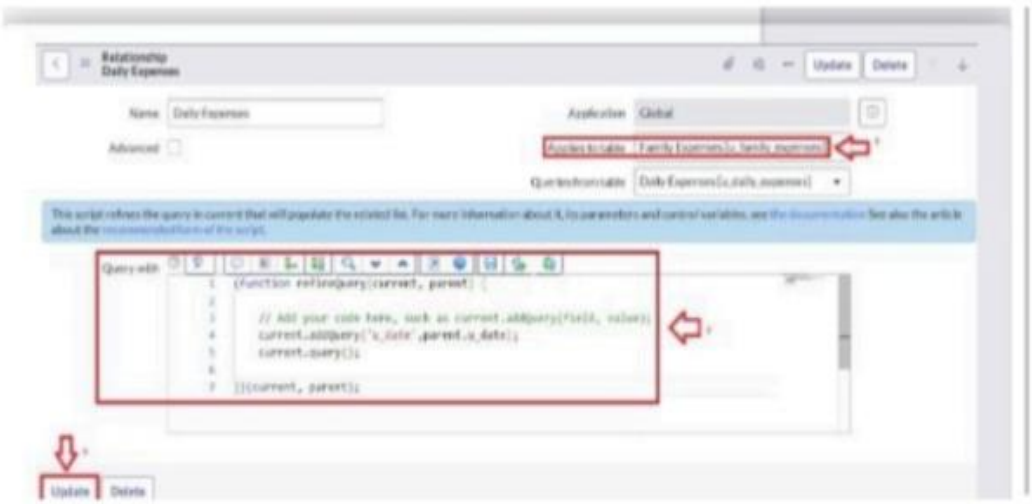
- Navigate: System Definition → Tables & Columns.
- Open your Family Expenses table.
- 3. Add a new field:
  - Type: Reference
  - Reference: Family (parent table)
  - Column Label: Family
  - Column Name: family
- 4. Save.

Configure Related List

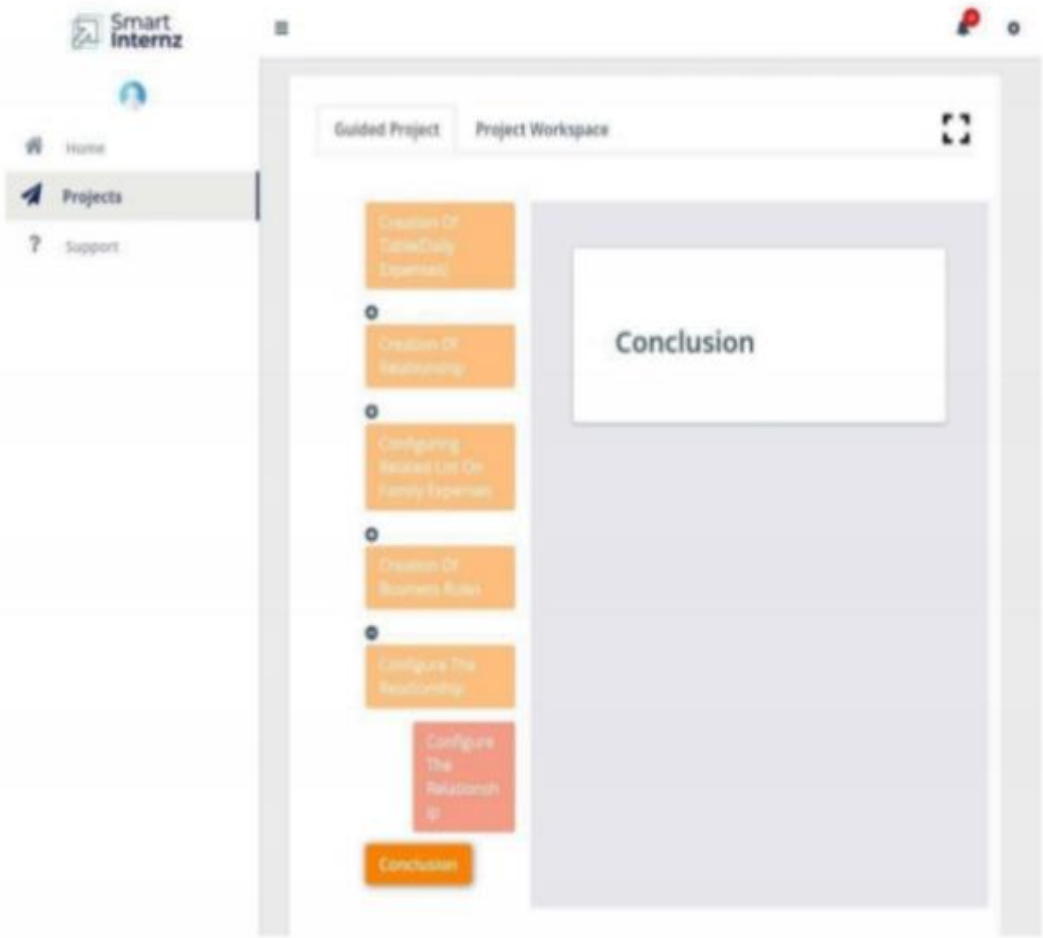
- Open any record in the Family table.
- Right-click the header → Configure → Related Lists.
- Move Family Expenses into the related lists.
- Save.

Many-to-Many Relationship

- Create a new table, e.g., Family_User_Map.
- Add two reference fields
  - family → Family table
  - user → User table
- 3. Add a related list for this table on both Family and User forms.



Conclusion:



Configuring relationships in ServiceNow is essential to link tables and create meaningful data connections. By using reference fields, we can easily establish one-to-many relationships such as Family.

→ Family Expenses, ensuring each expense record is tied to a specific family. For more complex cases, many-to-many relationships can be configured using a mapping table. Once relationships are set up, related lists allow users to view and manage connected records directly from the parent form. Overall, relationships improve data integrity, streamline navigation, and enable more powerful reporting in ServiceNow.