A.R.J COLLEGE OF ENGINEERING& TECHNOLOGY

EDAIYARNATHAM, MANNARGUDI-614 001 (APPROVED BY AICTE, NEW DELHI & AFFILIATED FOR ANNA UIVERSITY) (AN ISO 9001:2000 CERTIFIED INSITITUTION)

NM1051—SERVICENOW ADMINISTRATOR

DEPARTMENT OF COMPUTER SCIENCE

PROJECT TITLE: CALCULATING FAMILY EXPENSES USING SERVICE NOW

TEAM ID : NM2025TMID06941

SUBMITTED BY

1. **Team Leader** : Sivasankari S
2. **Team member :** Kalidhasan S
3. **Team member** : Ranjith M
4. **Team member :** Shriwadhsan C

Calculating Family Expenses using Service Now

ABSTRACT:

The project **“Calculating Family Expenses using ServiceNow”** aims to automate the process of recording, managing, and calculating family expenses efficiently. Traditionally, families use manual methods such as notebooks or spreadsheets to track expenses, which can be time-consuming and prone to human errors.

This project uses the **ServiceNow platform** to simplify this process by creating two connected tables — **Family Expenses** and **Daily Expenses** — and linking them with a **Business Rule** that automatically updates total amounts whenever a new daily expense is added. The system ensures data consistency, accuracy, and quick access to information through a user-friendly interface. The main goal of the project is to provide a reliable, low-code solution for effective financial tracking within a family environment, reducing manual work and improving decision-making.

TABLE OF CONTENTS:

1. INTRODUCTION

Managing family expenses is a vital part of maintaining financial stability. In many cases, daily spending is tracked manually, which often leads to calculation errors and difficulties in maintaining records. The project **"Calculating Family Expenses using ServiceNow"** provides an automated way to manage this process using ServiceNow's low-code environment.

By creating two interconnected tables — *Family Expenses* and *Daily Expenses* — users can easily record their daily transactions and automatically calculate total spending. The use of **Business Rules** ensures that updates happen instantly, maintaining accuracy and efficiency.

This system can be customized to handle any number of family members or spending categories. It offers a modern approach to expense management by combining automation, simplicity, and scalability. Ultimately, the project demonstrates how ServiceNow can be used effectively beyond IT workflows, extending into personal and household financial management.

2.1 OVERVIEW OF FAMILY EXPENSE MANAGEMENT

Managing household expenses manually can be time-consuming and error-prone. This project provides a digital method to manage financial data effectively.

2.2 NEED FOR AUTOMATION

Automation reduces human errors and improves efficiency. Using ServiceNow, we can handle expenses faster with accurate data updates.

2.3 ROLE OF SERVICENOW IN DATA MANAGEMENT

ServiceNow acts as a low-code platform that helps in creating custom applications, managing data tables, and automating business processes easily.

3. EXISTING SYSTEM

3.1 MANUAL EXPENSE MANAGEMENT

In the current system, family members record expenses in notebooks or spreadsheets manually.

3.2 CHALLENGES IN CURRENT SYSTEM

- Time-consuming calculations
- Errors in totaling expenses
- No data relationship between daily and family records

3.3 LIMITATIONS IDENTIFIED

The existing system lacks automation, data linkage, and accuracy in tracking overall spending.

4. PROPOSED SYSTEM

4.1 OBJECTIVES OF THE PROPOSED SYSTEM

- Automate expense calculation
- Link family and daily expenses dynamically
- Store all records in one platform

4.2 KEY FEATURES

- Family and Daily Expenses tables
- Reference fields for linking records
- Automatic total calculation using scripts

4.3 SCOPE OF IMPLEMENTATION

The project can be implemented for small families or even extended for organizations to manage expense tracking.

5. SYSTEM ANALYSIS

5.1 FUNCTIONAL REQUIREMENTS

- Create and manage Family Expenses
- Add daily records
- Automatically calculate totals

5.2 NON-FUNCTIONAL REQUIREMENTS

- Easy to use interface
- Reliable and secure data storage
- Fast query response

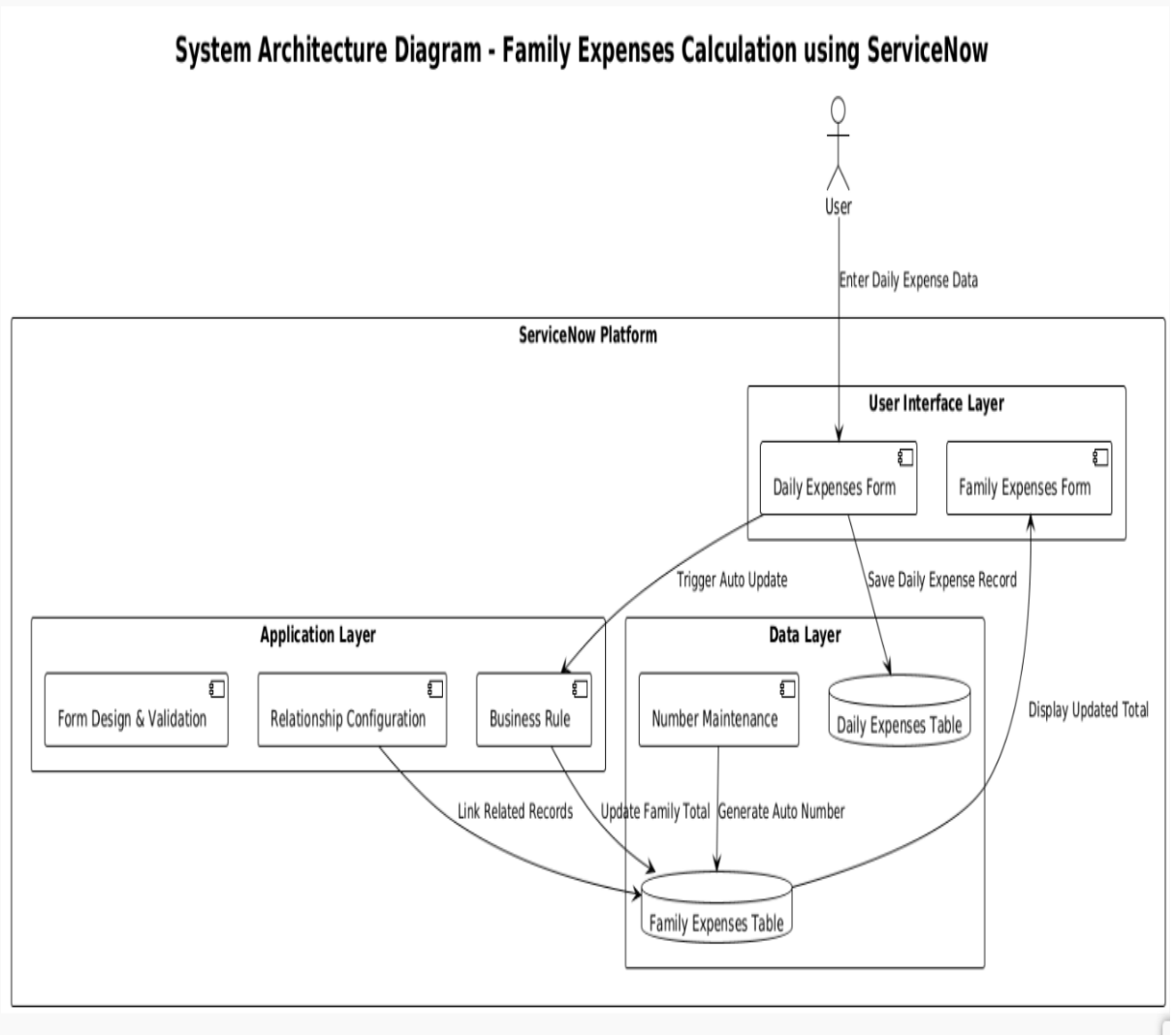
5.3 SYSTEM ARCHITECTURE

The architecture includes:

1. User Interface (Forms & Lists)
2. ServiceNow Database Tables
3. Business Rules and Client Scripts for automation

6. SYSTEM DESIGN

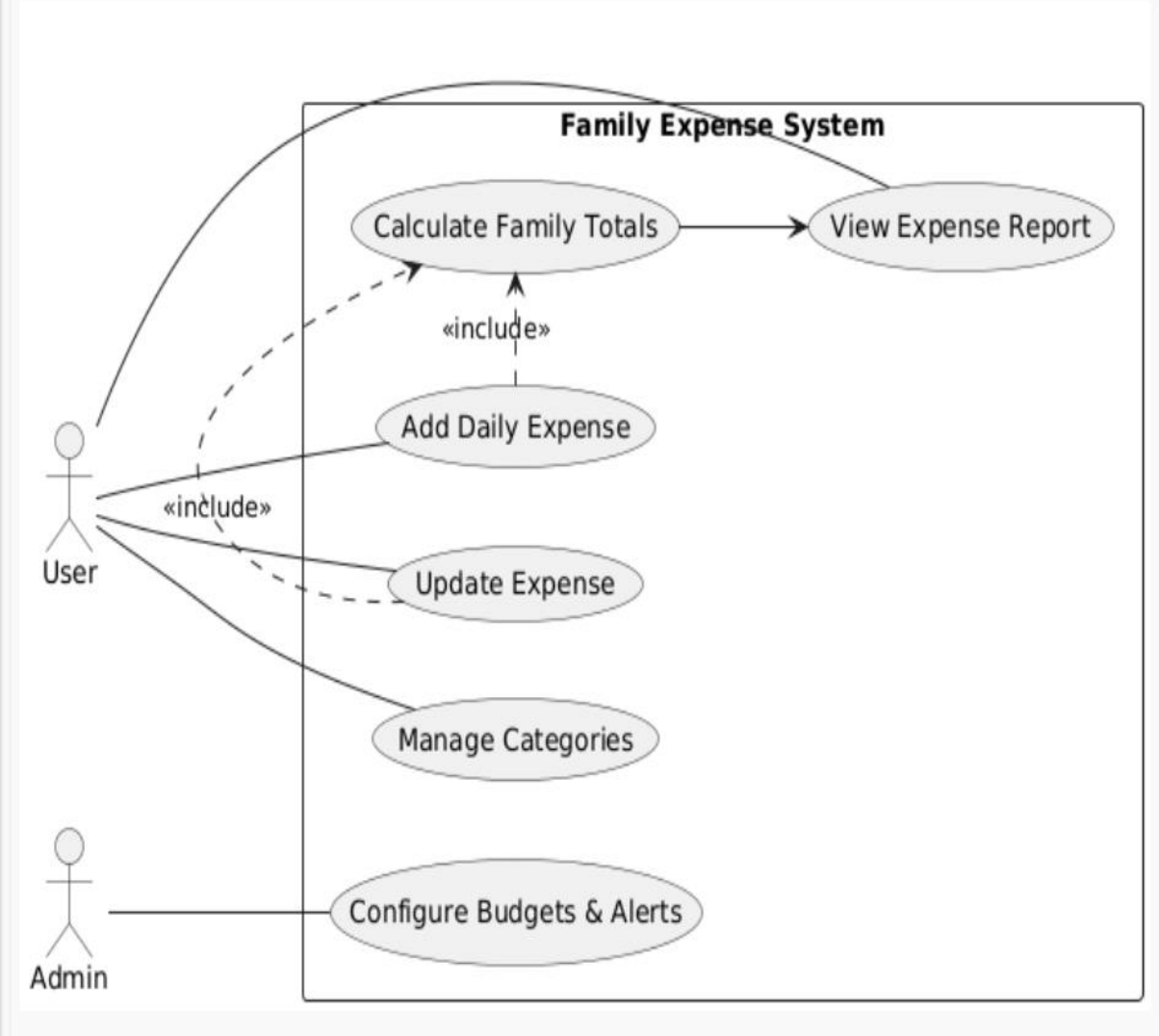
6.1 SYSTEM ARCHITECTURE DIAGRAM



6.2 USE CASE DIAGRAM

Actors: User (Family Member), System (ServiceNow)
Use Cases:

- Add family details
- Add daily expenses
- View related expenses
- Calculate total amount
- Generate report

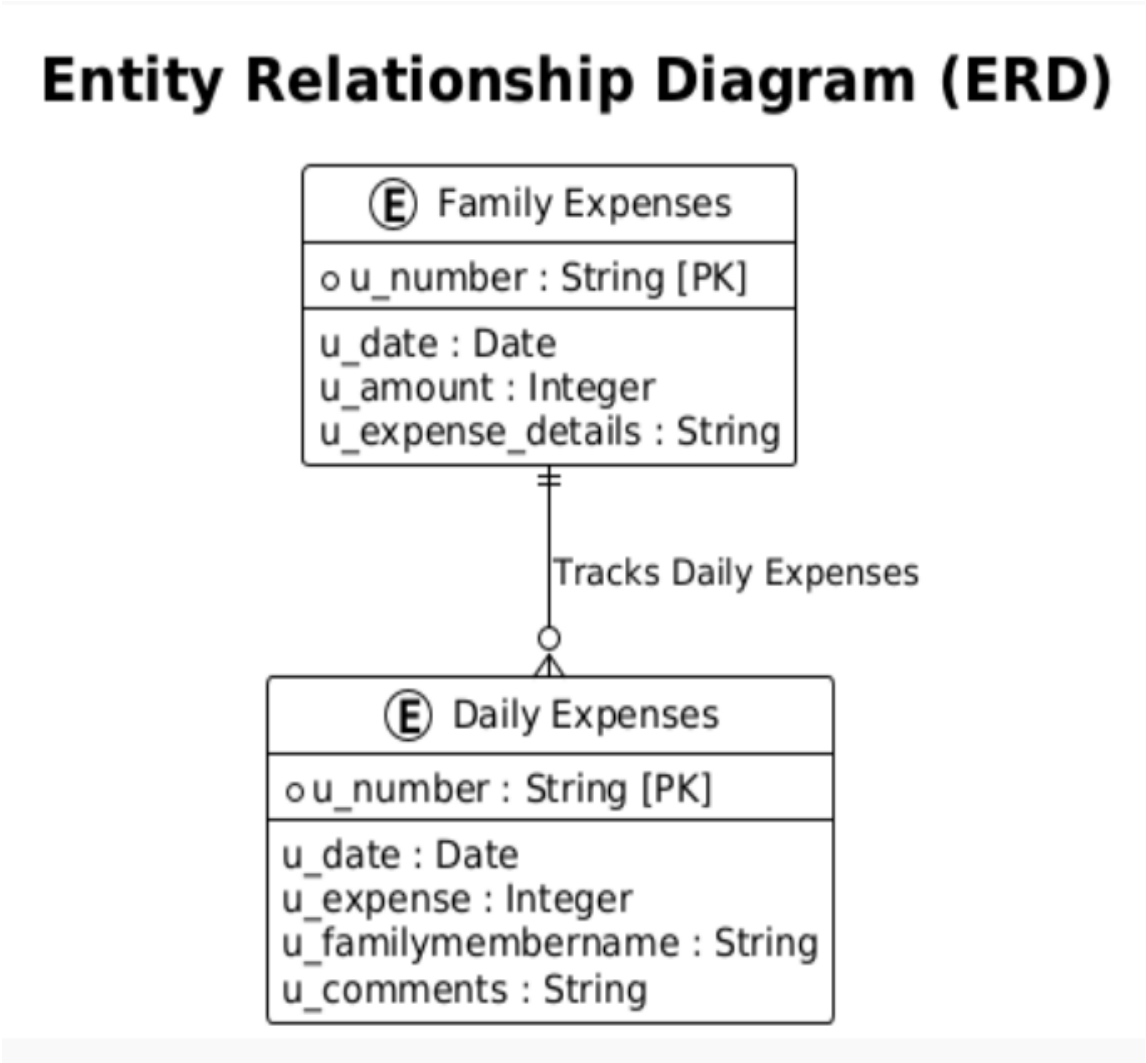


6.3 ENTITY RELATIONSHIP DIAGRAM (ERD)

ENTITIES:

- Family_Expenses (u_familyname, u_total, u_date)
- Daily_Expenses (u_item, u_amount, u_date, u_familyname)

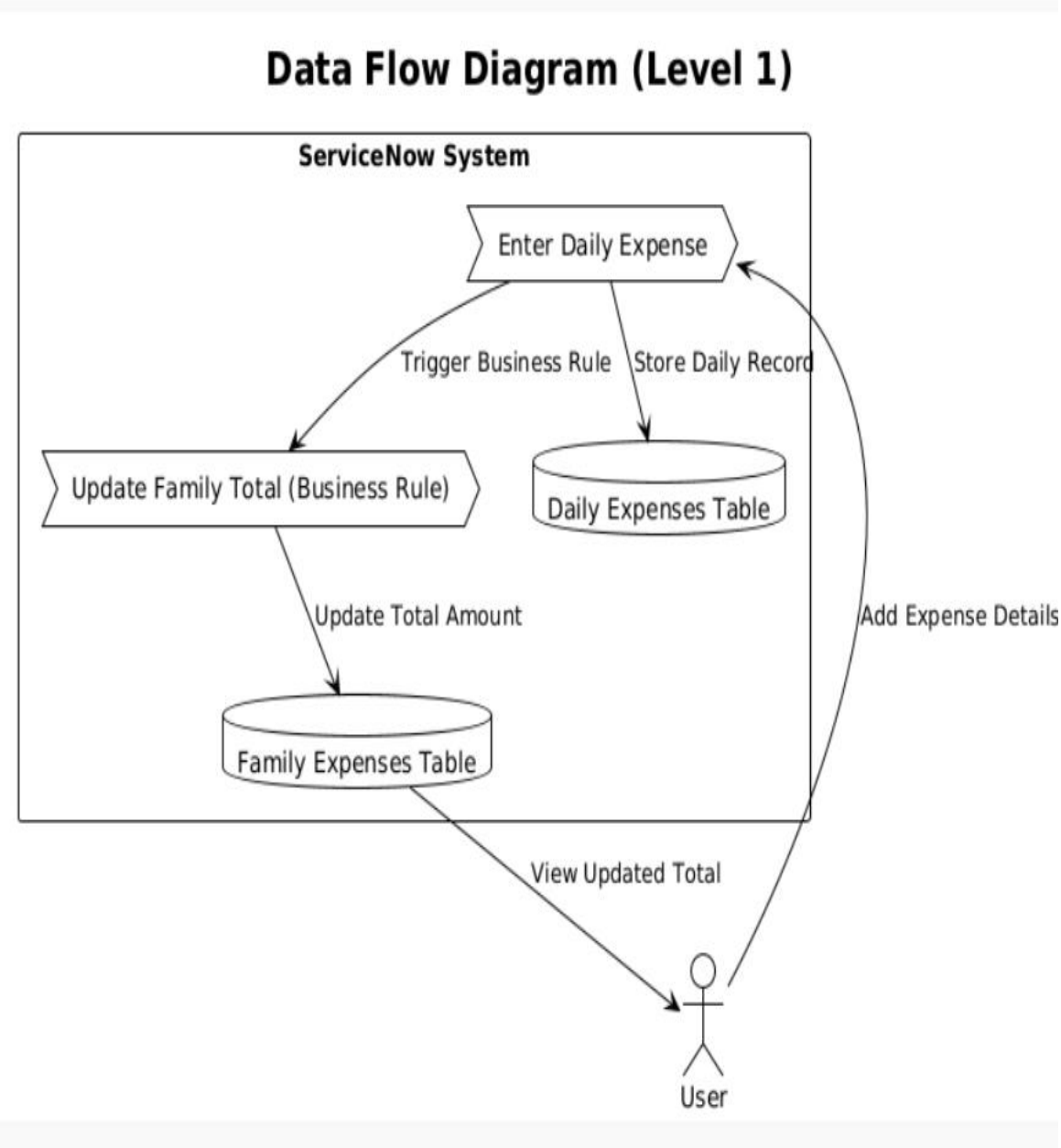
Relationship: Family_Expenses → Daily_Expenses (One-to-Many)



6.4 DATA FLOW DIAGRAM (DFD)

Level 0:
User → ServiceNow → Database → Output (Expense Summary)

Level 1:
Add Expense → Validate Data → Store in Daily_Expenses → Update Family_Expenses total



7. MODULE DESCRIPTION

7.1 FAMILY EXPENSES MODULE

Stores family details and overall spending totals.

7.2 DAILY EXPENSES MODULE

Stores individual daily spending items.

7.3 RELATIONSHIP CONFIGURATION MODULE

Connects both tables using a reference field and query.

8. METHODOLOGY

8.1 CREATION OF UPDATE SET

Created a new update set named *Family Expenses.*

8.2 CREATION OF TABLES

Two tables created: *Family Expenses* and *Daily Expenses.*

8.3 TABLE RELATIONSHIP SETUP

Configured relationship linking *Family Expenses* with *Daily Expenses.*

8.4 RELATED LIST CONFIGURATION

Added *Daily Expenses* to the *Family Expenses* related list.

8.5 REFERENCE FIELD CREATION

Created reference type field `u_familyname` in the *Daily Expenses* table.

8.6 QUERY CONFIGURATION

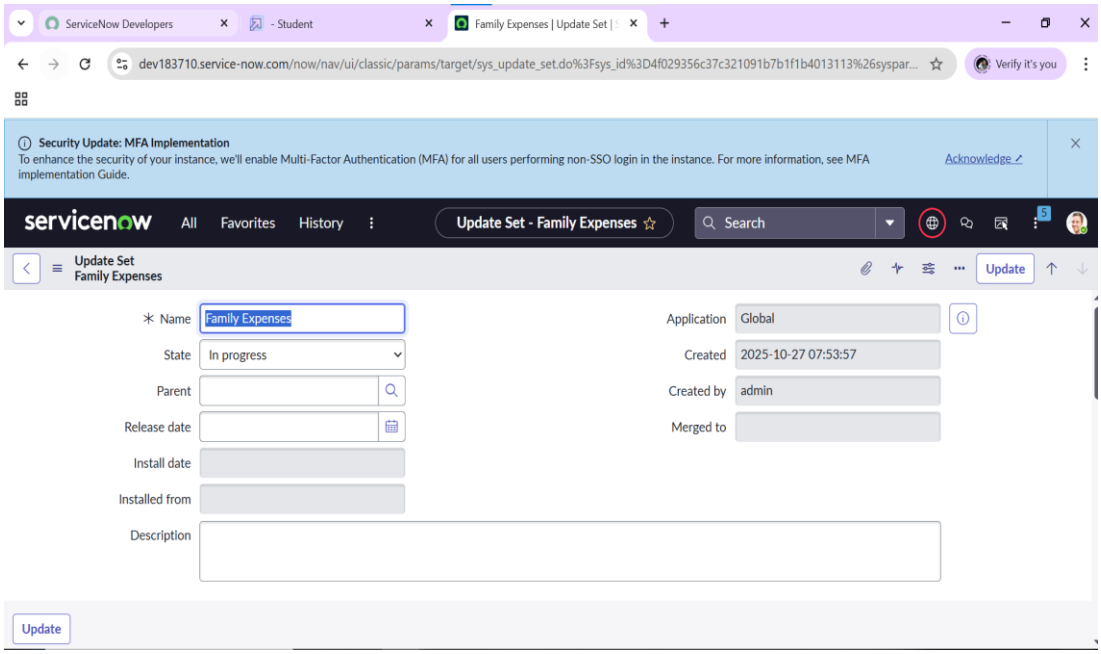
Configured relationship query:

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

IMPLEMENTATION:

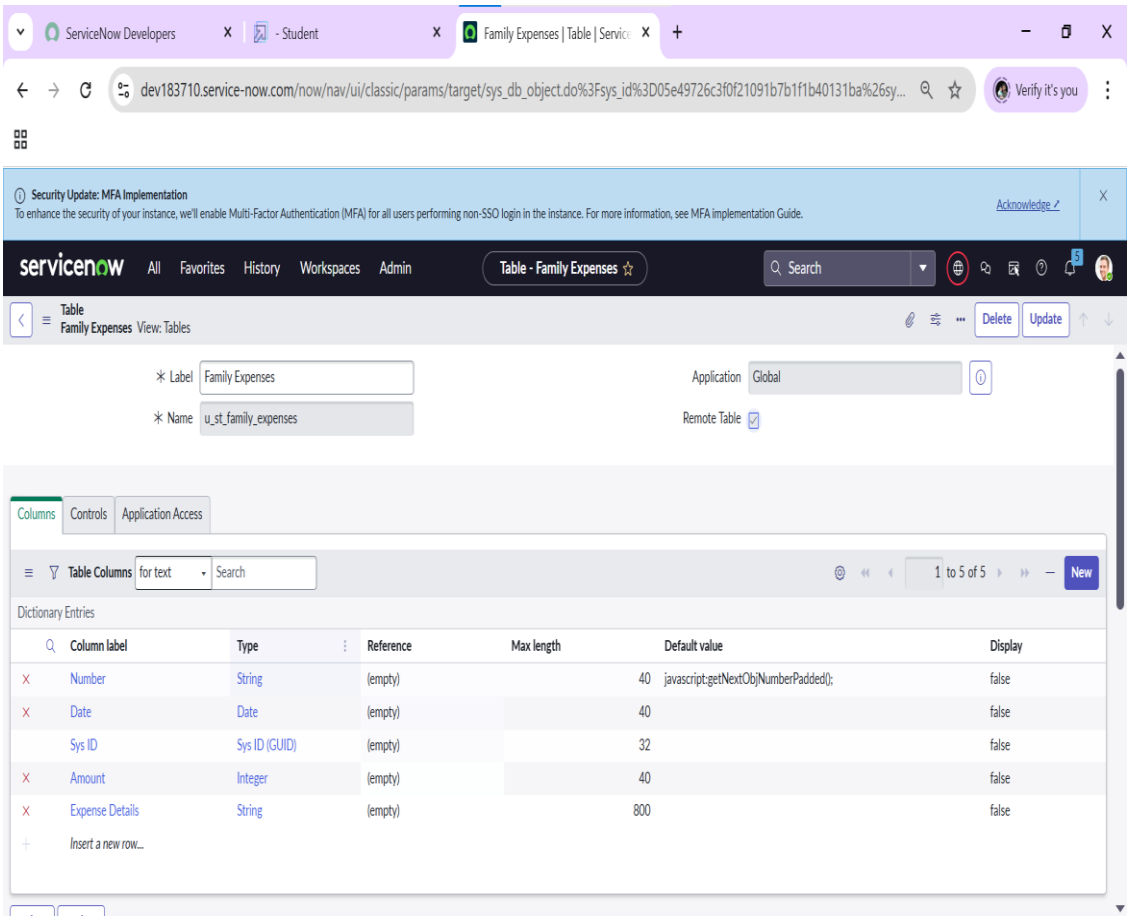
STEP 1 – CREATION OF UPDATE SET

1. Go to **All → Local Update Set → New**.
2. Enter:
    - **Name:** Family Expenses
3. Click **Submit** and **Make Current**.



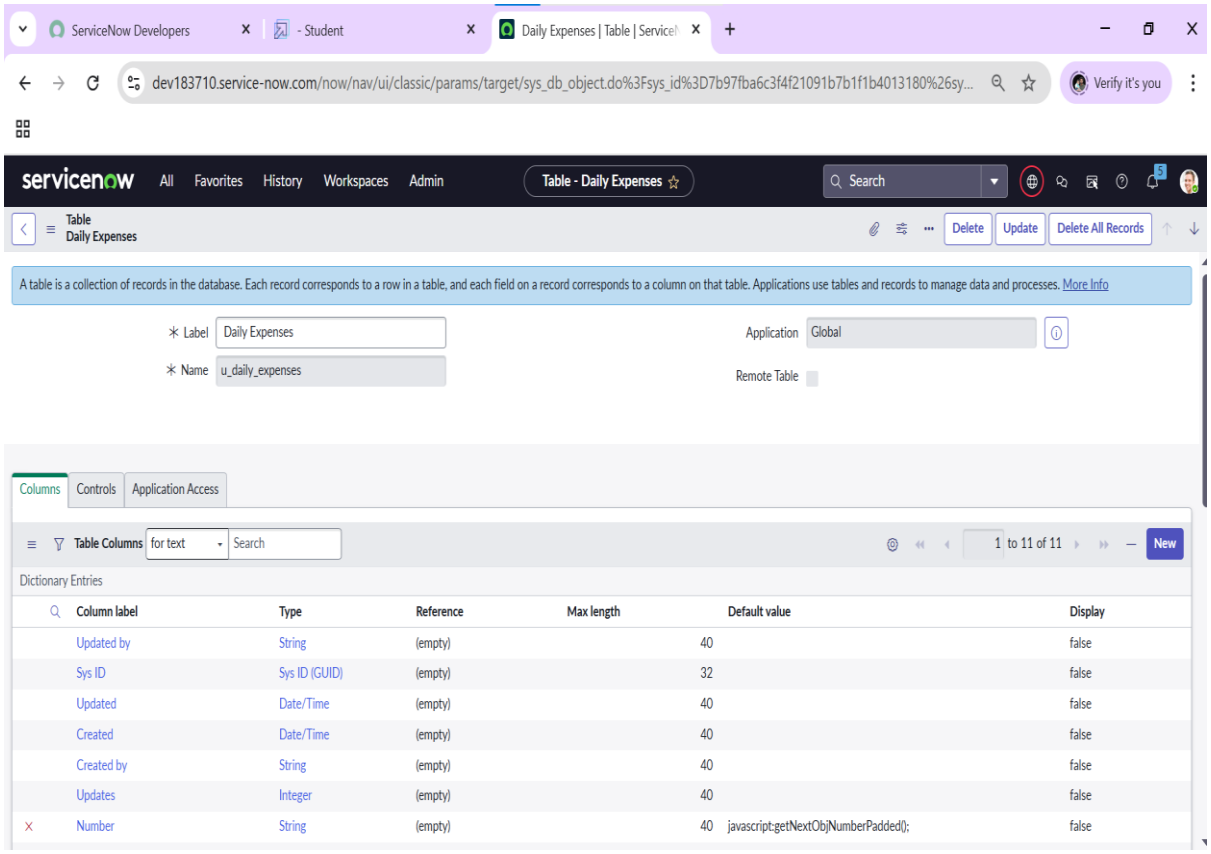
STEP 2 – CREATION OF FAMILY EXPENSES TABLE

1. Navigate to **All → Tables → New**.
2. Enter details:
    - **Label:** Family Expenses
    - **Menu Name:** Family Expenditure
3. Save the table.
4. Add columns:
    - Number (String)
    - Date (Date)
    - Amount (Integer)
    - Expense Details (String, Max length: 800)
5. Configure Auto-Number with prefix **MFE**.
6. In Form Design, make **Number** read-only and **Date**, **Amount** mandatory.



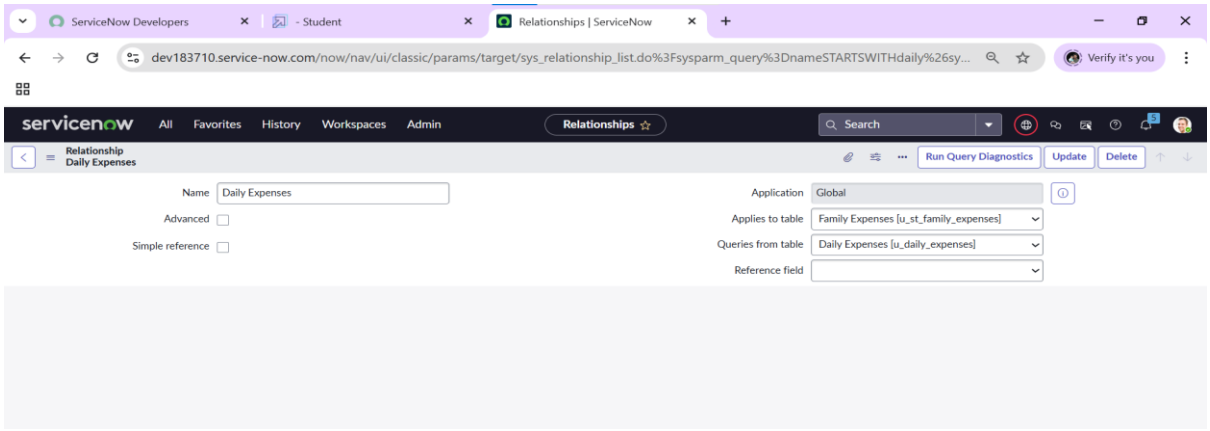
STEP 3 – CREATION OF DAILY EXPENSES TABLE

1. Navigate to **All → Tables → New**.
2. Enter details:
    - **Label:** Daily Expenses
    - **Menu:** Family Expenditure
3. Save the table.
4. Add columns:
    - Number (String)
    - Date (Date)
    - Expense (Integer)
    - Family Member Name (Reference)
    - Comments (String, Max length: 800)
5. Configure Auto-Number with prefix **MFE**.
6. In Form Design, make **Number** read-only and **Date**, **Family Member Name** mandatory.



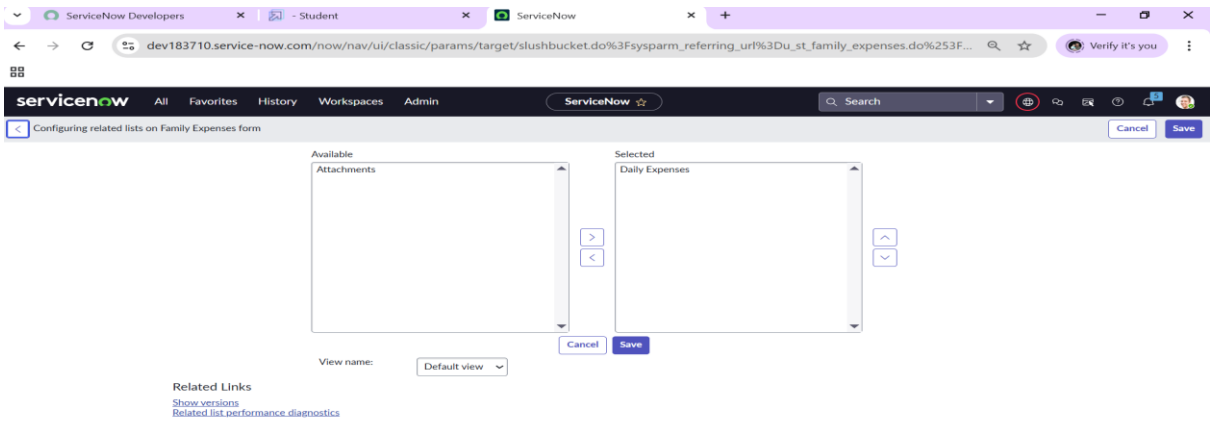
STEP 4 – CREATION OF RELATIONSHIP

1. Navigate to **All → Relationships → New**.
2. Enter details:
   - **Name:** Daily Expenses
   - **Applies to Table:** Family Expenses
   - **Daily Expenses:** Daily Expenses
3. Click **Save**.



STEP5 – Configuring Related List

1. Open **Family Expenses → New**.
2. Right-click header → **Configure → Related Lists**.
3. Add **Daily Expenses** to Selected Area.
4. Click **Save**.



STEP 6 – CREATION OF BUSINESS RULE

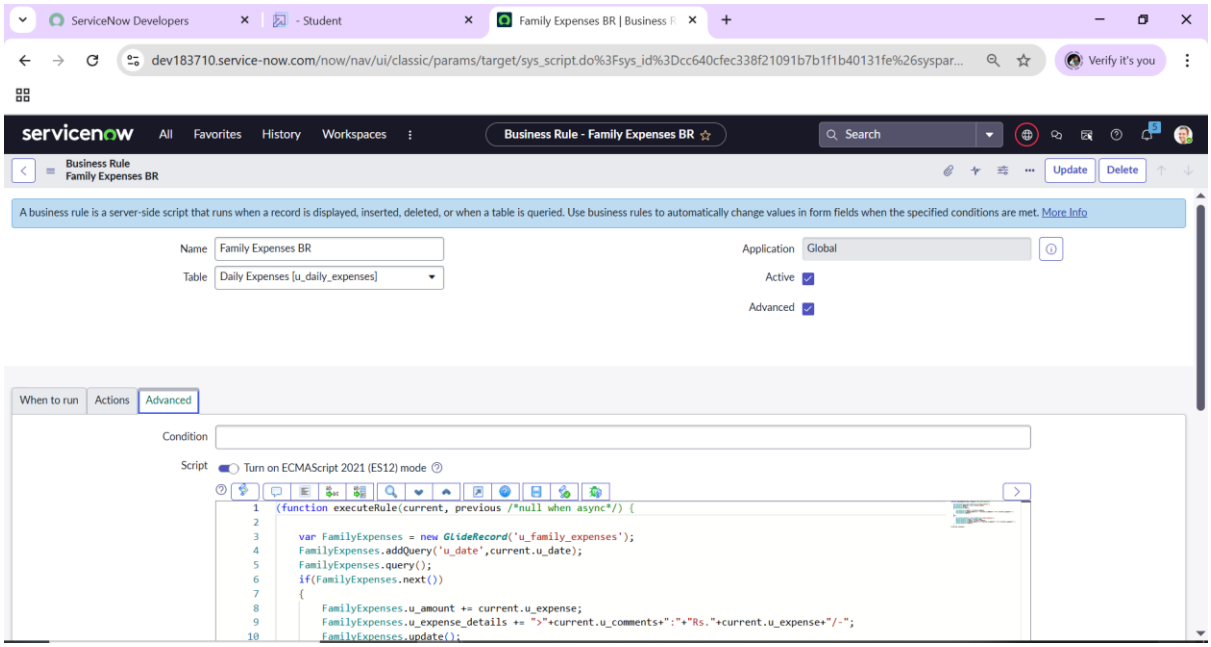
1. Navigate to **All → System Definition → Business Rules → New**.
2. Enter details:
    - **Name:** Family Expenses BR
    - **Table:** Daily Expenses
    - Check **Advanced**, and **Insert**, **Update**.
3. Add the following script:

```
(function executeRule(current, previous)
{
    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);
```

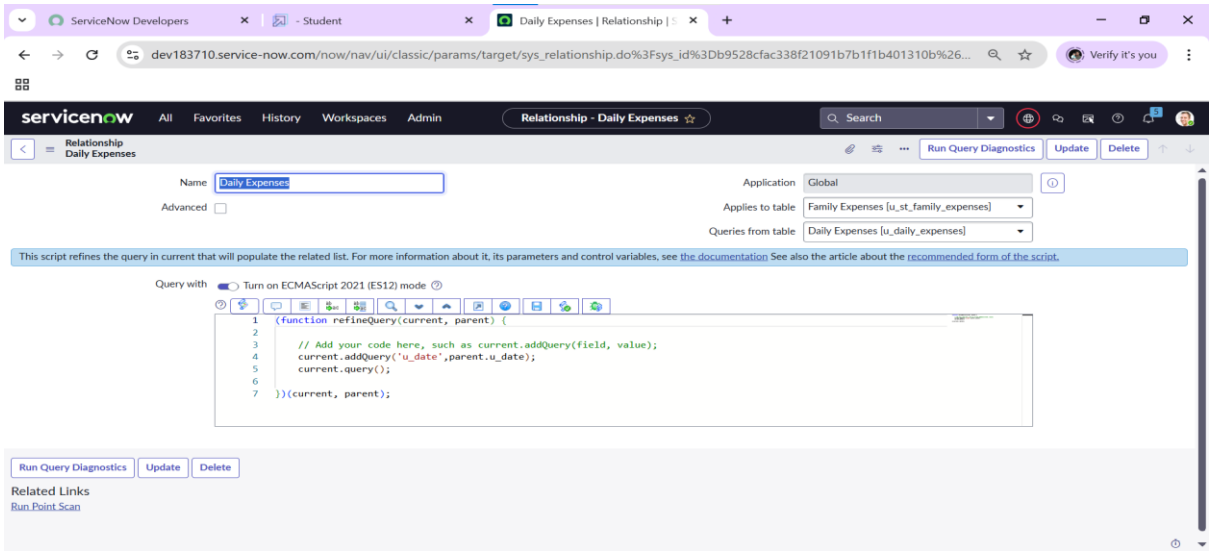


STEP 7 – CONFIGURE THE RELATIONSHIP

1. Navigate to **All → Relationships → Open Daily Expenses Relationship**.
2. Set **Applies to Table:** Family Expenses.
3. In “Query with” section, add:

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

4. Click **Update**.



9. RESULTS

Successfully created a dynamic and automated expense tracking system.
The Family and Daily Expenses tables are linked, and data flows correctly between them.

10. ADVANTAGES

- Simple and user-friendly
- Reduces manual calculation
- Accurate and fast data updates
- Automatic linkage between tables

11. DISADVANTAGE

- Works only inside ServiceNow
- Needs internet access
- Limited visual reporting features

12. FUTURE ENHANCEMENTS

- Add data visualization using charts
- Enable mobile access
- Include monthly summary reports
- Integrate export to Excel feature

13. CONCLUSION

The Family Expenses project is successfully developed using ServiceNow.
It automates the expense calculation process, saves time, and provides accurate results.
This project shows how ServiceNow can be effectively used for simple data management applications.

14. REFERENCES

- ServiceNow Documentation Portal – https://docs.servicenow.com
- ServiceNow Developer Online Learning – https://developer.servicenow.com
- Educational Institution Workflow Automation Research Papers, IEEE Access Journals
- ServiceNow Admin & Fundamentals Training Material
- Official Tutorials and System Configuration Guidelines from ServiceNow Community