**🧩 What Is Solution Architecture?**

A **solution architecture** shows how the **technical building blocks** of your application will solve the problem — in this case, **tracking and managing family expenses** using ServiceNow. It includes system components, data flow, access roles, and integration points.

**✅ 1. Key Layers of the Architecture**

Organize your solution into logical layers:

| **Layer** | **Components** |
| --- | --- |
| **Presentation (UI)** | Service Portal, Now Mobile App, Widgets |
| **Application Logic** | Flow Designer, Business Rules, Script Includes, Approvals |
| **Data Layer** | Custom Tables (Expenses, Budget), sys\_user, GlideRecord APIs |
| **Automation** | Notifications, Scheduled Jobs, Threshold Alerts |
| **Security Layer** | ACLs, Roles, Data Policies, User Permissions |
| **Integration (optional)** | REST APIs, IntegrationHub (e.g., Google Sheets, external bank feeds) |

**🏗 2. Solution Architecture Diagram (Textual Representation)**

pgsql

CopyEdit

+------------------------------+

| End Users |

|------------------------------|

| - Family Admin (Parent) |

| - Family Member (Spouse) |

+-------------+----------------

|

v

+------------------------------+

| Presentation Layer |

|------------------------------|

| - Service Portal (Web UI) |

| - Now Mobile App |

+-------------+----------------

|

v

+------------------------------+

| Application Logic Layer |

|------------------------------|

| - Expense Submission Form |

| - Budget Setup |

| - Approval Workflow (if any)|

| - Flow Designer (alerts) |

+-------------+----------------

|

v

+------------------------------+

| Data Layer |

|------------------------------|

| - Expense Table |

| - Budget Table |

| - User Table (sys\_user) |

+-------------+----------------

|

v

+------------------------------+

| Automation Layer |

|------------------------------|

| - Notifications (email/SMS)|

| - Scheduled Jobs |

| - Threshold Alert Logic |

+-------------+----------------

|

v

+------------------------------+

| Security Layer |

|------------------------------|

| - ACLs for Tables |

| - Roles: Admin, Member |

| - Audit Logs |

+-------------+----------------

|

v

+------------------------------+

| (Optional) Integration Layer|

|------------------------------|

| - REST APIs |

| - External Budget Apps |

+------------------------------+

**🔑 3. Component Details**

**🧾 Custom Tables (Data Layer)**

| **Table Name** | **Fields** |
| --- | --- |
| x\_expense\_entry | date, amount, category, submitted\_by, description |
| x\_budget\_setup | category, limit, start\_date, end\_date, created\_by |

**⚙️ Core Logic Components**

| **Component** | **Role** |
| --- | --- |
| **Flow Designer** | Triggers alerts when expenses hit budget threshold |
| **Script Includes** | Handle calculations (e.g., total by category) |
| **Business Rules** | Data validation before submission |
| **Approval Workflow** (optional) | Auto-approval if amount > threshold |

**🔐 Security Design**

| **Role** | **Access Level** |
| --- | --- |
| Family Admin | Full CRUD access, report visibility |
| Family Member | Create/view own expenses only |
| ACLs | Limit access per role and table |

**📱 UI/UX Design Tools**

* **Service Portal Pages**:
  + Expense Submission Page
  + Budget Dashboard
* **Widgets**: Use AngularJS-based widgets or GlideForm UIs
* **Mobile Views**: Designed with Now Mobile Studio for on-the-go entry

**🎯 4. Architecture Goals**

| **Goal** | **Strategy** |
| --- | --- |
| Simplicity for family users | Mobile/portal UI, minimal fields |
| Real-time visibility | Dashboards, automated updates |
| Flexibility | Role-based UI, customizable categories |
| Scalability | Add savings goals, integrate with banks |
| Maintainability | Low-code using Flow Designer |

**🖼️ Optional: Visual Diagram Tools**

To create the **architecture diagram visually**, use:

* [Draw.io](https://draw.io) (free and intuitive)
* Lucidchart or Miro (collaborative options)
* Figma (for UI + logic diagrams)

Or, I can generate a visual architecture diagram for you — just ask!