**🧠 SYSTEM FOR BUILDING SYSTEMS (SBS)**

**The Meta-System**

**Purpose:  
SBS — *System for Building Systems* — is a self-replicating automation framework designed to help you define, build, automate, and continuously improve any personal or business system with minimal friction.  
It uses n8n + PostgreSQL as the foundational stack, eliminating reliance on Notion or other proprietary platforms.**

**💡 Core Philosophy**

**Every system follows a consistent 5-step lifecycle:**

1. **Define → Clarify purpose, scope, and KPIs**
2. **Design → Architect workflows, tools, and integrations**
3. **Build → Construct working components**
4. **Automate → Connect triggers, recurring tasks, and alerts**
5. **Review / Iterate → Evaluate, document, and refine over time**

**⚙️ SBS Framework — Applied to Life Areas**

| **Stage** | **Meaning** | **How to Automate / Systematize** |
| --- | --- | --- |
| **🔍 Define** | **Identify purpose, inputs, outputs, success metrics** | **Use Postgres forms or n8n Webhooks to capture key system data** |
| **🧩 Design** | **Select tools, data flows, and roles** | **Auto-generate “System Design Canvas” in Markdown from templates** |
| **🔧 Build** | **Implement actual workflows** | **Use n8n flows to create folders, databases, and scaffolding** |
| **🤖 Automate** | **Link automation and triggers** | **Schedule n8n Cron + Webhooks; connect system events** |
| **🔁 Review** | **Analyze performance and log improvements** | **Auto-prompt via Telegram; log outcomes in Postgres** |

**🧰 CORE SYSTEM STRUCTURE (Postgres + n8n)**

**1. PostgreSQL Tables**

**sql**

**CREATE TABLE systems (**

**id SERIAL PRIMARY KEY,**

**name TEXT,**

**category TEXT,**

**purpose TEXT,**

**inputs TEXT,**

**outputs TEXT,**

**update\_frequency TEXT,**

**current\_stage TEXT DEFAULT 'define',**

**metadata JSONB DEFAULT '{}',**

**created\_at TIMESTAMP DEFAULT NOW()**

**);**

**CREATE TABLE system\_steps (**

**id SERIAL PRIMARY KEY,**

**system\_id INT REFERENCES systems(id),**

**step TEXT, *-- define/design/build/automate/review***

**status TEXT, *-- pending/complete/blocked***

**notes TEXT,**

**metadata JSONB DEFAULT '{}',**

**updated\_at TIMESTAMP DEFAULT NOW()**

**);**

**CREATE TABLE routines (**

**id SERIAL PRIMARY KEY,**

**name TEXT,**

**system\_id INT REFERENCES systems(id),**

**day\_of\_week TEXT,**

**description TEXT,**

**status TEXT DEFAULT 'active',**

**metadata JSONB DEFAULT '{}'**

**);**

**CREATE TABLE system\_templates (**

**id SERIAL PRIMARY KEY,**

**name TEXT,**

**category TEXT,**

**description TEXT,**

**default\_inputs JSONB,**

**default\_outputs JSONB,**

**schema\_ref TEXT**

**);**

**CREATE TABLE system\_logs (**

**id SERIAL PRIMARY KEY,**

**system\_id INT REFERENCES systems(id),**

**event TEXT,**

**details JSONB,**

**created\_at TIMESTAMP DEFAULT NOW()**

**);**

**2. Event-Driven Notifications (Postgres → n8n)**

**sql**

**CREATE OR REPLACE FUNCTION notify\_system\_update()**

**RETURNS trigger AS $$**

**BEGIN**

**PERFORM pg\_notify('system\_update', row\_to\_json(NEW)::text);**

**RETURN NEW;**

**END;**

**$$ LANGUAGE plpgsql;**

**CREATE TRIGGER system\_update\_trigger**

**AFTER INSERT OR UPDATE ON systems**

**FOR EACH ROW**

**EXECUTE FUNCTION notify\_system\_update();**

**This enables real-time orchestration — n8n subscribes to system\_update events and automatically updates workflows, eliminating polling delays.**

**3. n8n Workflows (Key Automations)**

**🔁 System Spawner**

**Trigger: When a new systems record is created (stage = 'define').  
Actions:**

* **Insert 5 lifecycle steps (define → review).**
* **Create default routines based on templates.**
* **Optionally create base folders in Google Drive.**
* **Send notification via Telegram/email.**
* **Log the event in system\_logs.**
* **Advance current\_stage → design.**

**⚙️ System Orchestrator**

**Trigger: When a system\_steps record changes status or stage.  
Actions:**

* **Detect which step is active.**
* **Route to the proper sub-workflow (Define, Design, etc.).**
* **Update systems.current\_stage.**
* **Write progress events to system\_logs.**
* **Optionally notify the user of the next step.**

**🧱 Step Subflows**

* **Define: Generate input form or prefilled values from template.**
* **Design: Create Markdown “System Design Canvas” in Postgres or Drive.**
* **Build: Spawn folders, DB schemas, and integration stubs via APIs.**
* **Automate: Add trigger-based n8n flows, schedules, and check-ins.**
* **Review: Auto-schedule review reminders and log evaluation notes.**

**📅 Routine Engine**

**Trigger: Cron (daily/weekly).  
Actions:**

* **Check routines due for the day.**
* **Send reminders via Telegram or email.**
* **Record completion or feedback.**
* **Optionally trigger maintenance flows.**

**🤖 Telegram / Email Bot**

**Purpose: Provide a conversational interface.  
Functions:**

* **Daily or weekly system check-ins (“It’s Money Monday — ready to review your Budgeting System?”).**
* **Mark routines complete or advance system stages.**
* **Fetch summaries or logs from Postgres.**

**📊 Optional: System Dashboard (No Notion)**

**Implementation options:**

* **Retool: fastest for no-code DB visualization**
* **Supabase Studio: native to Postgres**
* **n8n Webhooks + Custom Frontend: lightweight HTML/JS dashboard**
* **Custom React App: full UI with system triggers and controls**

**Dashboard views:**

* **Systems grouped by category**
* **Lifecycle stage status**
* **Upcoming routine schedule**
* **Last activity logs**
* **Manual “Trigger Automation” buttons**

**💥 Example: Creating a New System**

**New row added to systems:**

**json**

**{**

**"name": "Net Worth Tracker",**

**"category": "Money Monday > Budgeting",**

**"purpose": "Track and visualize net worth automatically",**

**"inputs": "Account balances, assets, liabilities",**

**"outputs": "Net worth graph, monthly change",**

**"update\_frequency": "Monthly"**

**}**

**n8n Reaction Sequence:**

1. **System Spawner inserts default lifecycle steps + routines.**
2. **Folders and logs created automatically.**
3. **Markdown “System Design Canvas” generated and stored.**
4. **Review reminder auto-scheduled in 30 days.**
5. **Telegram bot notifies you:  
   *“System ‘Net Worth Tracker’ created. Ready to define inputs?”***

**🧭 Workflow Architecture (Mermaid Diagram)**

**text**

**flowchart TD**

**A[🧠 User Input<br>(Form / Telegram / API)] -->|Creates new system| B[(💾 PostgreSQL DB)]**

**B -->|New system (stage='define')| C[🔁 n8n Workflow: System Spawner]**

**B -->|Event or update trigger| D[⚙️ n8n Workflow: System Orchestrator]**

**C -->|Insert lifecycle steps + routines| B**

**C -->|Send Telegram/email notification| H[📲 Notification Bot]**

**C -->|Advance stage → design| B**

**D -->|Detect step + route to subflow| E[🧩 Step Handlers]**

**E -->|Define| E1[Define: Purpose + Inputs Form]**

**E -->|Design| E2[Design: Canvas Generation]**

**E -->|Build| E3[Build: APIs + Folders]**

**E -->|Automate| E4[Automate: Flows + Schedules]**

**E -->|Review| E5[Review: Evaluate + Prompt]**

**E -->|Update DB + log| B**

**F[📅 Routine Engine (Cron)] -->|Daily/Weekly check| B**

**F -->|Send check-ins| H**

**H -->|Mark routine complete + push stage| B**

**B --> G[📊 Dashboard (Retool / Supabase / Web UI)]**

**E3 --> I[🌐 External APIs / Integrations]**

**E4 --> I**

**✅ Final Stack Summary**

| **Area** | **Tool** | **Purpose** |
| --- | --- | --- |
| **💾 Core Data Store** | **PostgreSQL** | **Centralized system database** |
| **🔁 Automation Engine** | **n8n** | **Step orchestration, events, and triggers** |
| **🧱 Subflows** | **n8n** | **Lifecycle execution (Define→Review)** |
| **🤖 Bot** | **n8n + Telegram** | **Chat-based interface for updates** |
| **📅 Routine Generator** | **n8n Cron + PG** | **Schedules reviews and routines** |
| **🧩 Templates** | **PostgreSQL** | **Blueprints for new systems** |
| **📊 Dashboard** | **TBD** | **Status monitoring and control panel** |
| **🌐 External Integrations** | **APIs (Drive, Sheets, etc.)** | **Data exchange and file management** |

flowchart TD

%% User & Input Layer

A[🧠 User Input<br>(Form / Telegram / API)] -->|Creates new system| B[(💾 PostgreSQL DB)]

%% Database Backbone

B -->|New system with stage='define'| C[🔁 n8n Workflow: System Spawner]

B -->|Triggers updates / events| D[⚙️ n8n Workflow: System Orchestrator]

%% System Spawner Logic

C -->|Insert 5 lifecycle steps| B

C -->|Create default routines| B

C -->|Send creation notification| H[📲 Notification Bot]

C -->|Advance stage → design| B

%% System Orchestrator Flow

D -->|Check system\_steps.status| E[🧩 Step Subflows]

E -->|Define| E1[Define → Capture Purpose / Inputs]

E -->|Design| E2[Design → Generate Canvas]

E -->|Build| E3[Build → Create Folders / DBs / APIs]

E -->|Automate| E4[Automate → Connect Triggers / Schedules]

E -->|Review| E5[Review → Evaluate / Schedule next cycle]

E -->|Update System Status| B

%% Routine Engine

F[📅 Routine Engine<br>(n8n Scheduled Flow)] -->|Weekly / Daily checks| B

F -->|Send reminders| H

H -->|Mark routine complete or push step| B

%% Dashboard & Integration Layer

B --> G[📊 Dashboard<br>( TBD)]

E3 --> I[🌐 External Integrations<br>(Google Drive / APIs / Webhooks)]

E4 --> I

%% Data flow relationships

style A fill:#ffebc6,stroke:#d4a41c,stroke-width:1px

style B fill:#e2f0d9,stroke:#6aa84f

style C fill:#d9e1f2,stroke:#4f81bd

style D fill:#d9e1f2,stroke:#4f81bd

style E fill:#f2dcdb,stroke:#c0504d

style F fill:#fbe4d5,stroke:#e46c0a

style H fill:#fde9d9,stroke:#c55a11

style G fill:#f4cccc,stroke:#990000

style I fill:#ddebf7,stroke:#2e75b6