System Documentation: Create Project Workflow

Workflow Name: Create Project Workflow (as inferred from context)
Source File: Main_Chat_Workflow_for_Documentation (1).json
Project Lead: Mike Holland
System Architect: Gemini Pro

1. Overall Goal

This workflow serves as the dedicated backend process for creating a new, simple project. Its sole purpose is to receive a project name from the UI, ensure it doesn't already exist, create all the necessary records and IDs in the database, and return a success response to the UI so a new chat session can begin. It acts as a secure and robust entry point for Scenario 1.

2. Key Components & Architecture

- **Trigger:** A unique n8n **Webhook** (...d0b91f11...) listens for POST requests from the Chat 8 UI.
- **Database:** It interacts exclusively with the project_contexts table in the Postgres database to check for duplicates and create new project records.
- **Logic:** The workflow follows a simple conditional path: if the project already exists, it fails gracefully; if not, it proceeds with creation.

3. Step-by-Step Data Flow

The workflow executes in a clear, linear sequence:

1. **Webhook:** The workflow is triggered when the user submits a new project name from the UI. The payload contains { "projectName": "..." }.
2. **Check if Project Exists:** A Postgres node immediately queries the project_contexts table to see if any record already has the provided projectName.
3. **If (Project Exists?):** An If node checks the result of the previous query.
    - **If True (project exists):** The workflow takes the True branch and immediately triggers the **Respond to Webhook1** node, which sends a specific error message back to the UI, stopping the process.
    - **If False (project is new):** The workflow proceeds down the main False branch to create the project.

4. **Code (Prepare SQL Parameters):** A Code node runs to prepare the data for the database INSERT command. It generates a unique rag_session_id and sessionId and assembles all the required values into an array (sql_params).
5. **Postgres (Create Project Record):** This Postgres node executes the main INSERT query, using the sql_params from the previous step to create the new record in the project_contexts table. Crucially, it uses RETURNING * to output the full, newly created database row, including the permanent xata_id.
6. **Generate Thread ID:** A Code node generates a new, unique chat_session_id (formatted as session_...).
7. **Save Thread ID to Project:** A Postgres node executes an UPDATE query. It takes the new_thread_id and the xata_id from the previous steps and updates the project record to include the new chat session ID.
8. **Prepare Success Response:** A Set node takes the project_id (from the xata_id) and the session_id (from the new_thread_id) and formats them into a clean JSON object.
9. **Respond to Webhook:** The final Respond to Webhook node sends this success object back to the UI, providing it with all the necessary IDs to initialize the new chat session.