System Architecture Overview: The "Bob" AI Memory Project

Project Lead: Mike Holland
System Architect: Gemini Pro
Version: 2.1 (Post-Refactor, Persona-Enabled)

1. Executive Summary

The "Bob" AI Memory Project is a sophisticated, multi-component application designed to overcome the inherent limitations of standard Large Language Model (LLM) interactions, specifically the problem of finite context windows and the lack of persistent memory. It achieves this by integrating a stateful web interface with a suite of powerful backend data processing workflows, creating a robust system capable of both short-term conversational recall and long-term, project-specific knowledge retention. The architecture is built on the principles of Retrieval-Augmented Generation (RAG), providing the AI with a dynamic, relevant, and persistent memory store that evolves with user interaction.

2. The Core Problem & The "Bob" Solution

Standard chatbots suffer from "amnesia"; they forget the details of a conversation as soon as it exceeds their context window. The "Bob" project solves this by creating two distinct types of memory:

- **Short-Term Memory (conversation_history):** A sliding window of the most recent conversational turns, allowing the AI to maintain natural, flowing dialogue.
- **Long-Term Memory (rag_store):** A permanent, vectorized knowledge base. Users can explicitly "commit" key facts from a conversation to this store. This knowledge is then automatically retrieved and provided to the AI as context in future conversations within the same project, giving the AI a true, persistent memory.

3. System Components

The project is comprised of three primary, interconnected components:

a. The Chat 8 UI (The "Cockpit")

A comprehensive, single-page web application that serves as the user's complete control center. It is responsible for:

- **Project Lifecycle Management:** Creating new projects, either as empty sandboxes (Scenario 1) or by bootstrapping knowledge from external JSON files (Scenario 2).
- **Session Management:** Loading and resuming existing projects, ensuring a persistent user experience across browser sessions.

- **State Orchestration:** Managing all critical identifiers (project_id, chat_session_id, rag_session_id) and ensuring they are correctly passed to the backend, governed by a strict **Master Data Contract**.
- **User Interaction:** Providing the core chat interface and triggering all backend processes via the "Send" and "Commit to Memory" buttons.

b. The n8n Workflows (The "Engine Room")

A suite of five specialized, serverless workflows that handle all backend logic and data processing. Each workflow is a dedicated microservice with a single responsibility:

1. **Setup Workflow - Project Launcher:** The system's "receptionist," responsible for listing existing projects and handling the initial phase of new project creation.
2. **Create Project Workflow:** The "registrar," which creates the permanent database records for new, simple projects.
3. **RAG Indexing Workflow:** The "librarian," a powerful data pipeline that ingests external JSON files, chunks the text, generates vector embeddings, and populates the long-term memory store.
4. **System: Context Distillation & Indexing:** The "Commit to Memory" engine. This workflow takes recent conversation history, uses an AI to distill it into atomic facts, and saves those facts to the long-term memory store.
5. **Main Chat Workflow:** The "brain" of the operation. This is the primary RAG workflow that receives every user message, orchestrates the parallel retrieval of both short-term and long-term memory, assembles the final, context-rich prompt, and generates the AI's response.

c. The Postgres/pgvector Database (The "Filing Cabinet")

The persistent data layer of the system, comprised of several key tables:

- **project_contexts:** The master index, linking human-readable project names to the system's machine-readable IDs.
- **conversation_history:** The complete, raw transcript of all conversations, serving as the source for short-term memory.
- **rag_store:** The vectorized long-term memory. This is the core of the RAG system, where distilled facts and their embeddings are stored for high-speed similarity searches.

- **memory_commit_log:** A "high-water mark" table that ensures the "Commit to Memory" process is efficient and never processes the same message twice.
- **prompts & project_prompts:** The advanced Prompt Management System, allowing different projects to be assigned unique AI personalities and rule sets.

4. Conclusion

The "Bob" project is a testament to the power of a well-architected, multi-component system. Through a rigorous process of design, implementation, and "battle-hardening" debugging, it has evolved from a simple concept into a robust, feature-rich, and stable platform. It successfully solves the core problem of AI amnesia and provides a powerful foundation for future development and expansion. The system is online and fully operational.