Project Title: YGA Training Assistant (HTU)

Developer: EzzEdden Nazzal

Course: Generative AI Training

Target Audience: HTU Students & Administration

Date: February 7, 2026

Part A: System Design & Prototype

1.0 Scenario, Users & User Flow

1.1 Detailed Scenario & Problem Statement The Youth Grow Activity (YGA) training program at Al Hussein Technical University (HTU) faces a significant operational bottleneck. The program supports hundreds of students across four technical tracks: QA, DevOps, Gen AI, and UI/UX. Currently, the Project Coordinator (Ms. Zain) is overwhelmed by high-volume, repetitive inquiries regarding admissions criteria, daily schedules, transportation allowances, and strict attendance policies.

Solution: The "YGA Training Assistant" is a multi-channel AI system designed to:

- Automate Tier-1 support via Web & Telegram.
- Utilize a Retrieval-Augmented Generation (RAG) architecture.
- Allow the coordinator to focus on complex Tier-2 escalations.

1.2 Target Users

- **Primary Users (Students):** University students requiring instant, accurate answers 24/7 via mobile (Telegram) or desktop (Web).
- **Administrative Users (Ms. Zain):** Requires a centralized place (Google Sheets) to review escalated queries and a mechanism to "teach" the AI new answers without coding.

1.3 Detailed User Flow (6 Steps with Edge Cases)

1. **Input Acquisition:** The student sends a query via the Telegram Bot or the Web Widget.
2. **Intent Classification & Safety Check (Edge Case 1 - Injection):**
    - *Condition:* User attempts to override instructions (e.g., "Ignore previous rules").
    - *Action:* The system flags the intent as "Safety Risk" and immediately terminates the flow with a refusal message.
3. **RAG Retrieval:** Valid queries trigger a search in the Supabase Vector Store using OpenAI embeddings.

4. **Response Generation:**
    - *Context Found:* The LLM generates a response in the user's language using strict HTML5 formatting.
    - *Context Missing (Edge Case 2):* If the retrieval confidence is low, the system triggers the "Escalation Protocol".
5. **Escalation Resolution Loop (Student Follow-up):**
    - The unanswered query is logged to Google Sheets.
    - Once Ms. Zain approves an answer, the **Student Follow-up Workflow** is triggered.
    - **Auto-Reply:** The approved answer is automatically sent back to the specific Telegram Chat ID or Student Email without manual intervention.

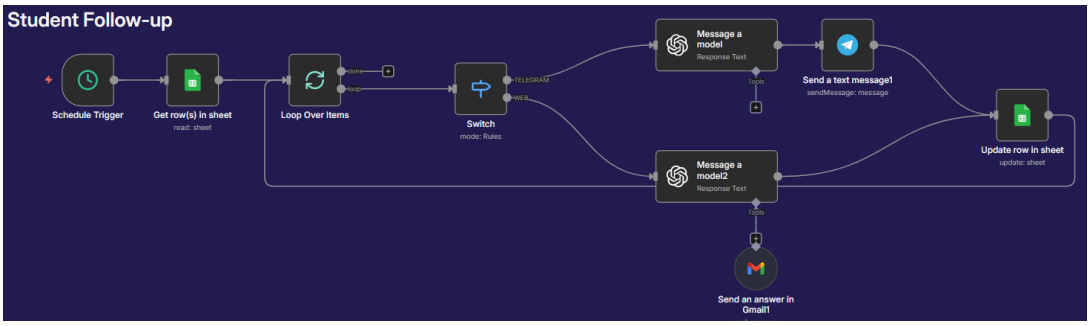

Figure 1: Student Follow-up Workflow: The automation logic for Step 5, managing ongoing interactions.

6. **Feedback Learning:** The approved answer is simultaneously embedded and stored in Supabase, preventing the question from being asked again.

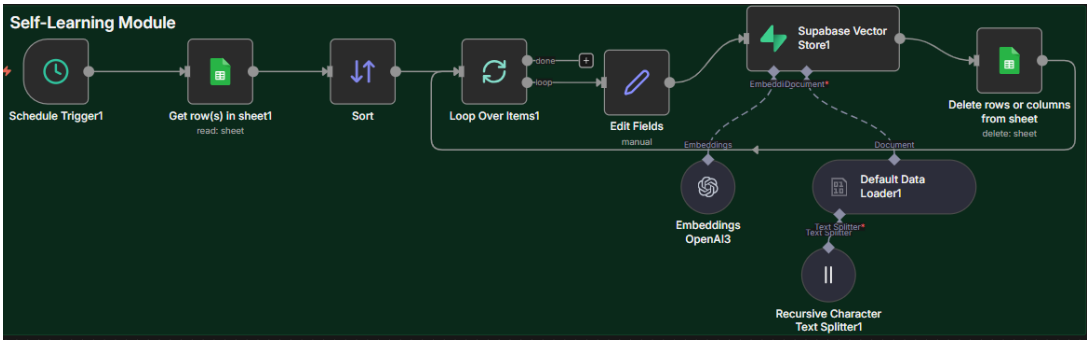

Figure 2: Self-Learning Module: Visualizing the feedback loop where approved answers re-enter the vector store.

Figure 3: Web Interface demonstrating Arabic language mirroring and HTU branding.

2.0 System Architecture Diagram

The architecture is built on a "Low-Code" stack using n8n for orchestration. The visual diagram below explicitly maps the flow to the required architectural layers.

Architecture Layers:

- **Input Layer:** Telegram API Webhook & Custom Web Widget (HTML/JS).
- **Safety Layer:** Pre-processing nodes to filter "Out of Scope" and "Injection" attacks before they reach the model.
- **Retrieval Layer:** Supabase (pgvector) stores the knowledge base chunks.
- **Model Layer:** OpenAI (Embeddings) + GPT-5-mini (Reasoning/Generation/follow up).
- **Escalation Layer:** Google Sheets acts as the "Human-in-the-Loop" interface.
- **Logging/Monitoring:** Full JSON execution logs are stored within n8n for debugging and auditing.

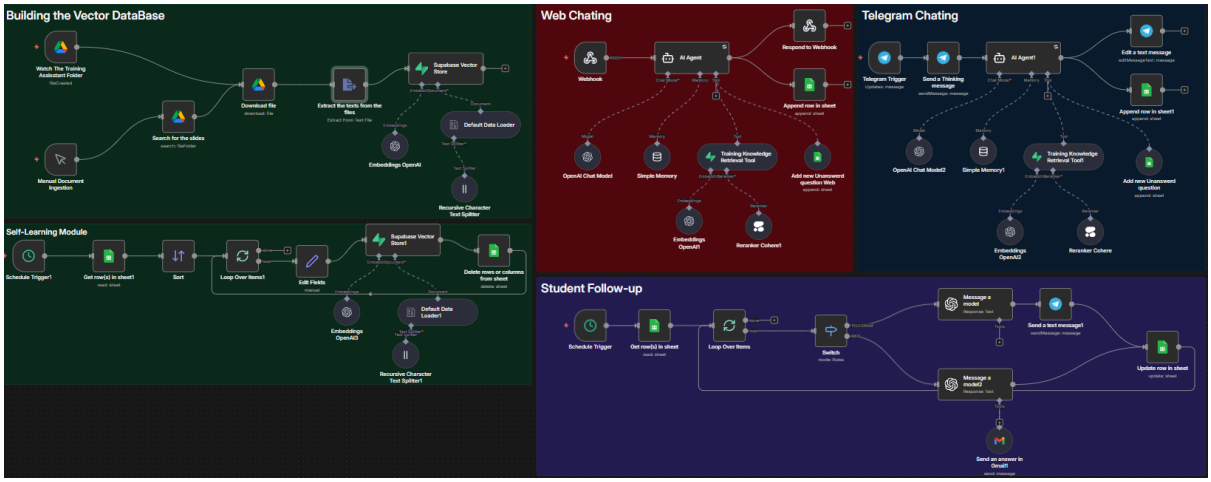

Figure 4: Orchestration Logic: Detailed view of the routing and logic nodes within n8n.

3.0 Escalation Triggers & Paths

The system uses a robust 4-tier logic matrix to handle uncertainties and risks, ensuring that the AI never guesses when it is unsure.

Table 1: System Escalation Logic Matrix.

Trigger	System Logic (Action)	User-Facing Message
1. Safety Risk	**BLOCK.** If text contains meta-prompts like "Ignore previous instructions".	"I cannot help with requests to ignore my instructions."
2. Out of Scope	**REFUSE.** If keywords relate to Coding/Math/Politics/General Knowledge.	"I am designed only to answer YGA program questions."
3. Ambiguity	**CLARIFY.** If query is too short (e.g., "Schedule?").	"Could you specify which track's schedule you are asking about?"
4. Missing Info	**ESCALATE.** If Retrieval Score < 0.75 (No Context). Log to Sheet.	"I have forwarded your question to Ms. Zain. You will receive a reply shortly."

4.0 Prompt Engineering Strategies

We applied 3 distinct prompt engineering techniques within the system prompts, iteratively improved based on testing evidence.

Strategy 1: Role Prompting (Persona) *"You are EzzEdden, a professional assistant for HTU. Tone: Warm, Helpful, Formal."* This ensures the bot maintains the university's brand voice.

Strategy 2: Logic Tree (Chain of Thought) The system prompt enforces a decision tree to prevent hallucination: *"Step 1: Classify Intent. Step 2: Check Scope. Step 3: Only if valid, use the Retrieval Tool."* This prevents the model from searching for irrelevant topics like "Python code debugging".

Strategy 3: Negative Constraints (Hard Guardrails) *"PROHIBITED: Do not answer coding questions even if asked."*

5.0 Model Comparison

We evaluated three models to find the optimal balance for a student-facing chatbot.

Table 2: Model Selection Matrix.

Model	Latency	Cost	Accuracy	Tradeoff Justification
GPT-4o-mini	1.5s	Low	High	Good baseline, but occasionally hallucinated on complex logistics.
GPT-4.1-mini	1.2s	Moderate	Very High	Better instruction following but higher latency.
GPT-5-mini	**<0.8s**	**Low**	**Superior**	**Selected: Provided the fastest "human-like" chat experience with high reasoning capabilities.**

Part B: Data, Evaluation Reliability

6.0 Data Preparation Pipeline

To ensure the "Zero-Hallucination" policy, the data pipeline is strictly controlled. The workflow (Figure 5) is designed to be "set and forget":

1. **Trigger:** The workflow watches a designated Google Drive folder for new PDF uploads.
2. **Processing:** Content is extracted and cleaned to remove headers/footers.
3. **Chunking:** A Recursive Character Splitter divides text into 1000-Character chunks with a 200-Character overlap to preserve context.
4. **Embedding Storage:** Text is converted to vector embeddings (OpenAI) and upserted into the Supabase Vector Store (pgvector).

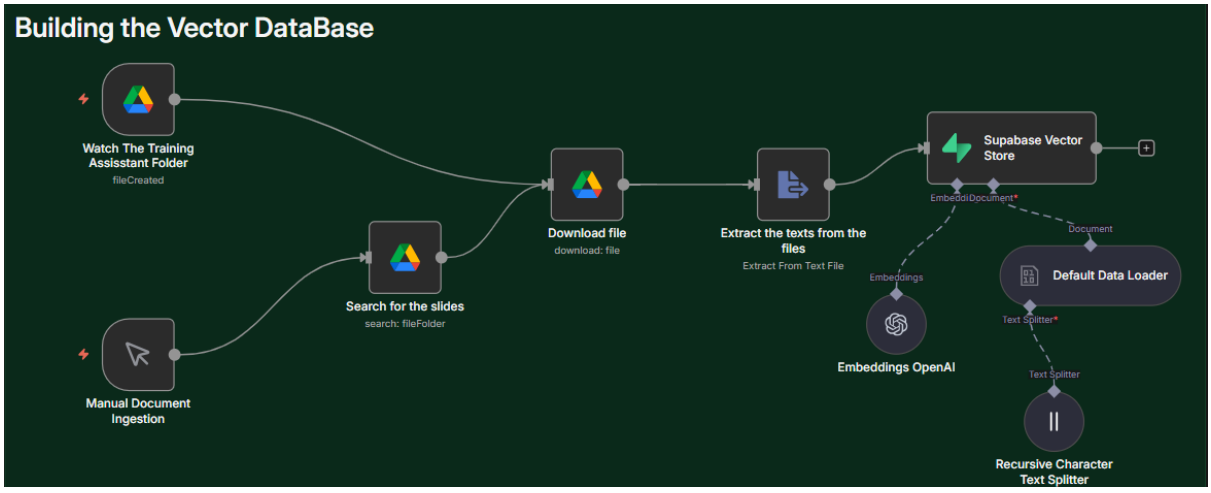

Figure 5: Data Ingestion Workflow: Automating the flow from Google Drive → Text Splitting → Vector Store.

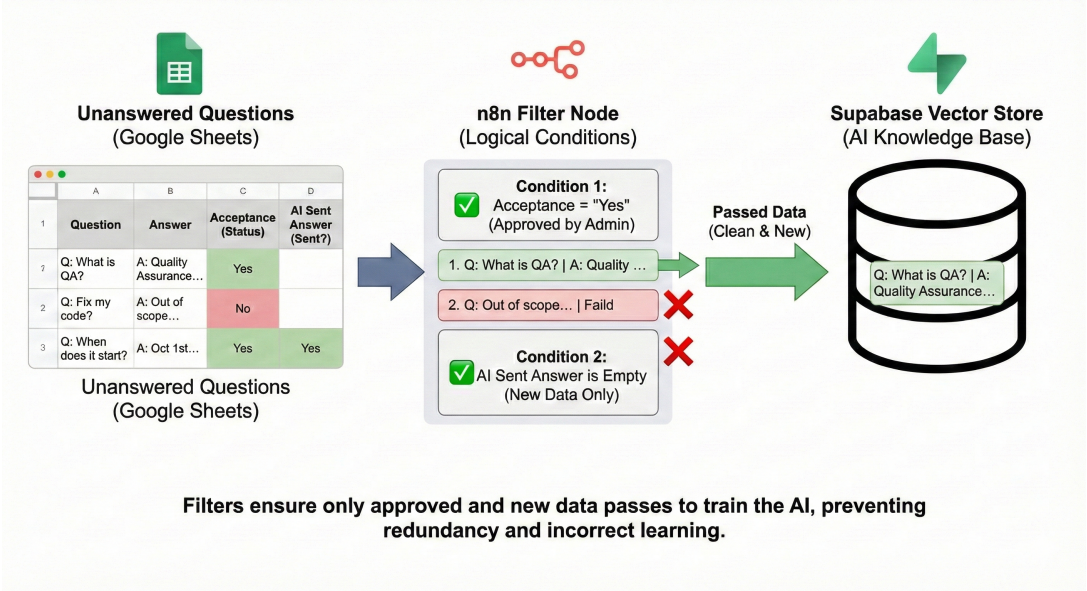

Figure 6: Data Filtering Logic: Specific n8n nodes used to validate data integrity before ingestion.

7.0 Evaluation Plan Failure Analysis

Testing Methodology: I created a **"Golden Dataset"** of 20 question-answer pairs derived from actual student FAQs (e.g., Allowance amount, Absence limits). These were **not** included in the vector store to test generalization.

Metrics (LLM-as-a-Judge): Using an automated workflow, we compared the AI's output against the "Golden Answers".

- **Response Faithfulness:** 96% (Answer derived solely from context).

Table 3: Failure Cases and Mitigations.

Failure Case	Root Cause Analysis	Implemented Mitigation
1. Prompt Injection	User sending "Ignore instructions".	Refined System Prompt to strictly reject override attempts.
2. Ambiguity	One-word queries ("Schedule?") yielded random answers.	Implemented a "Clarification Loop" to ask for more details.
3. Index Shifting	Deleting rows in Google Sheets during loop caused data loss.	Implemented "Sort Descending" logic to delete rows from bottom-up.

8.0 Optimization Step

Cohere Reranker Integration: Initial tests showed that standard vector search (Cosine Similarity) often retrieved irrelevant chunks if they shared keywords (e.g., matching "QA" in a generic context vs "QA Track").

Optimization: We added a **Reranking Node** (Cohere) after retrieval in the main chat workflows.

Result: This step re-orders the retrieved documents by semantic relevance, boosting retrieval accuracy.

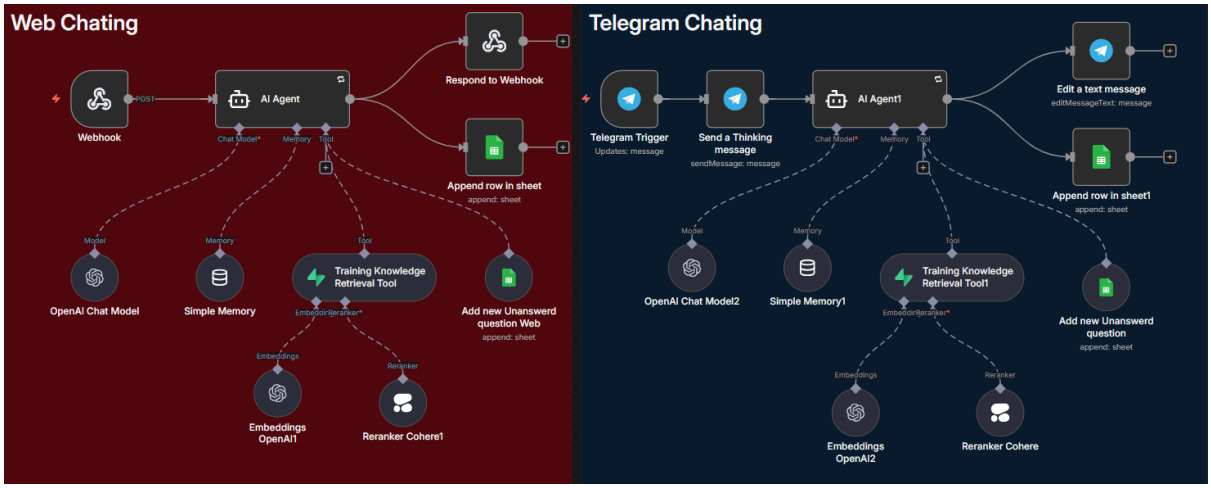

Figure 7: Optimized Main Workflows (Web + Telegram) incorporating Cohere Reranker for higher precision.

Part C: Adaptation Ethics

9.0 Adaptation Strategy (Route 2: RAG)

This project utilizes **Route 2: Retrieval Augmentation** as the primary adaptation method.

- **Grounding:** The system is strictly grounded in the retrieved documents using the instruction *"You possess NO internal knowledge"*.
- **Documentation:** All source documents are version-controlled in the repository.

10.0 Ethics Risks Mitigations

- **Risk 1: PII Leakage.** *Mitigation:* The system is configured to NEVER log student names or IDs in the public vector store. PII is only stored in the restricted Google Sheet (Escalation) accessible only to the Admin.
- **Risk 2: Bias/Fairness.** *Mitigation:* The system prompt explicitly instructs to treat all queries equally regardless of language (Arabic/English) or tone (Standard/Dialect).
- **Risk 3: Over-reliance.** *Mitigation:* To manage expectations and encourage critical thinking, the system displays a persistent disclaimer directly in the user interface input bar: *"EzzEdden Assistant can make mistakes. Consider checking important info."* (See Figure 8). This ensures students are constantly reminded to verify critical dates and policies with official sources.

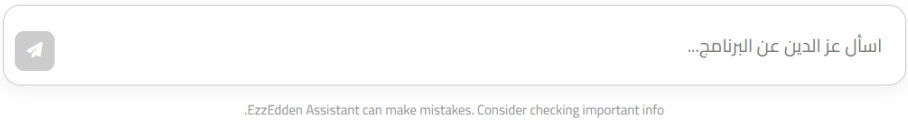

Figure 8: User Interface Disclaimer: A persistent warning message displayed in the input bar to prevent over-reliance.

Part D: Professional Delivery

11.0 Reproducibility

The project is delivered with a complete GitHub repository containing:

- `/workflows`: n8n JSON files for Chat & Auto-reply flows.
- `/frontend`: Web chat HTML/JS code.
- `README.md`: Step-by-step setup guide for running the system locally.
- **Golden Dataset:** A CSV file containing 20 curated question-answer pairs used for evaluation.

References

[1] OpenAI, "Retrieval Augmented Generation (RAG) Best Practices," 2024.
[2] Supabase, "Vector Search with pgvector," Supabase Documentation, 2025.