29-11-2025

Project RAG | AI Coach for AWS Training

What Problem Are We Solving?

- AWS learning can be overwhelming for beginners
- Hard to find structured explanations, quizzes, or code examples
- No unified tool that can teach, quiz, generate code, and learn new topics from YouTube
- Why not use other AI tools??

What This System Can Do

- Explain any AWS topic
- Run quizzes (MCQs) with scoring + feedback
- Generate AWS code snippets (Python/Boto3)
- Learn new topics automatically from YouTube videos
- Store the knowledge in its own knowledge base (self-learning)

High-Level System Architecture

```
User → Streamlit UI → RAG_Agent
        |              |
        |              → AWS KB VectorStore (Chroma)
        |              → Transcript VectorStore
        ↓
 YouTube Transcript Engine (yt-dlp + Whisper)
```

Key Components

- main.py – Controls logic & routing
- RAG_Agent.py – Core intelligence (teach, quiz, code)
- aws_info.py – Handles KB updates + vectorstore versioning
- NLP.py – Cleans and chunks transcripts
- transcript_extraction.py – Pulls YouTube subtitles / Whisper
- vector_store.py – Handles transcript embeddings
- app.py – The Streamlit user interface

Knowledge Base

- aws_info.py manages all AWS Knowledge Base operations:
- Loads + cleans raw KB text
- Splits into semantic chunks using spaCy
- Builds vectorstore versions (no overwriting)
- Stores meta.json with KB hash → auto-build new version on changes
- Provides retriever to RAG_Agent
- Computes relevance score to see if topic is already known

How RAG Checks Similarity & Answers Questions

- User asks: “What is AWS Glue?”
- RAG pipeline loads latest vectorstore version
- Computes similarity between query embedding and KB chunks
- If relevance score ≤ threshold → topic found → have to fix
- Retrieves top chunks from aws_knowledge_base.txt
- LLM uses ONLY retrieved context to answer
- Prevents hallucinations by restricting to AWS KB content

User Query → Embedding → Chroma Similarity → Top Chunks → LLM Answer

RAG - Brain of the system

- Query normalization (adds AWS prefix, strips quiz words)
- Intent detection (teach / quiz / code)
- Retrieval selection (AWS KB or transcript vectorstore)
- TEACH mode → context-based explanation
- QUIZ mode → LLM-generated MCQs + evaluation
- CODE mode → generate Boto3/Python code with explanation

User Query → Intent → Normalize → Retrieve → LLM Prompt → Output

How the Bot Automatically Chooses Teach / Quiz / Code

<u>Intent Classification Logic:</u>

- If query contains quiz words → QUIZ mode
- If query contains code-related terms → CODE mode
- Otherwise → TEACH mode

<u>Example Detection:</u>

- “quiz on S3” → QUIZ
- “give python code for S3 upload” → CODE
- “explain AWS Glue” → TEACH

<u>Each mode routes to its own chain:</u>

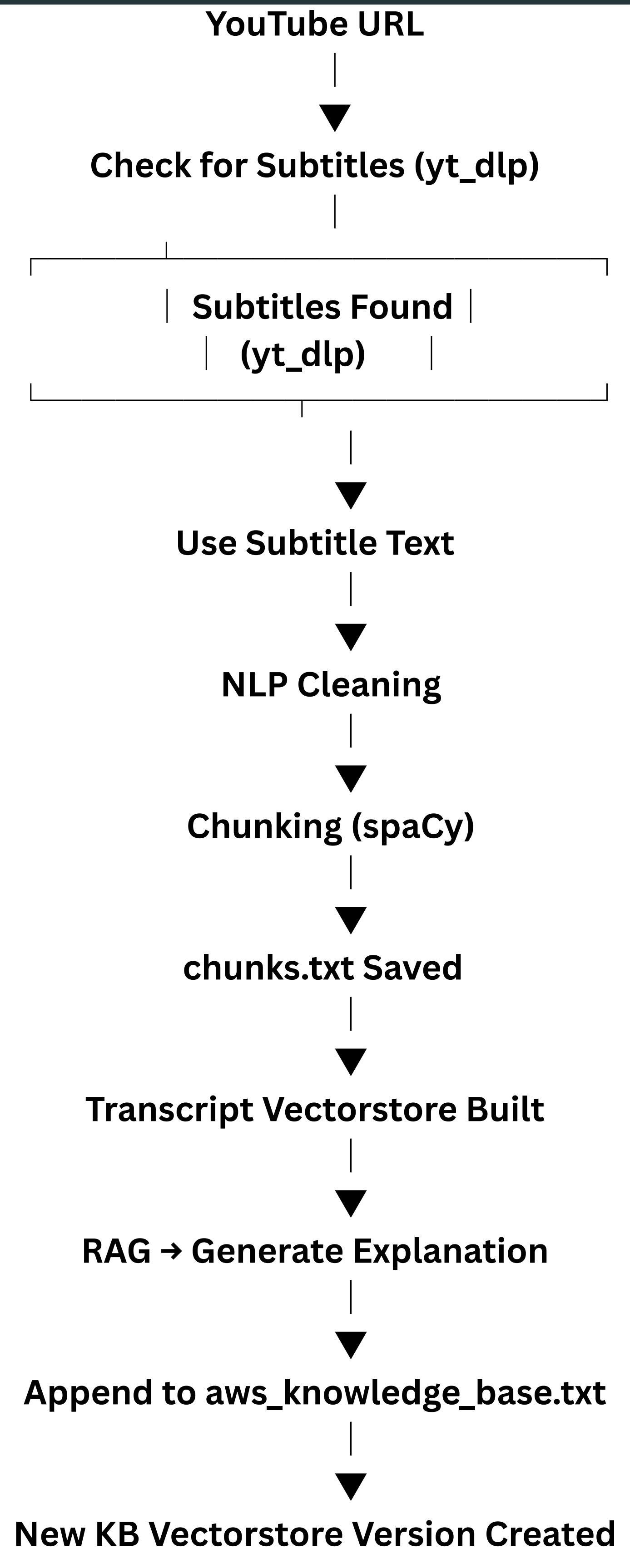
- teach() → context explanation
- generate_quiz() → MCQs
- generate_code_answer() → code + explanation

Agents summary

Step	What It Does	Type of Agent	Description
Step 4A – Retriever	Finds the most relevant transcript chunks	🧠 *Knowledge Agent*	Works like a memory retriever — fetches relevant info for the next step.
Step 4B – RAG Chain (Teacher)	Reads retrieved chunks + explains concepts	👩‍🏫 *Teacher Agent*	Uses GPT (ChatOpenAI) to answer beginner questions clearly, step-by-step.
Step 4C-A – Quiz Maker	Creates MCQs & evaluates answers	🧩 *Quiz Agent*	Generates and grades multiple-choice questions based on transcript context.
Step 4C-B – Code Helper	Generates Python/SQL code + explanations	💻 *Coding Agent*	Uses transcript + LLM reasoning to write short code examples with explanation.
Step 4C-C – Role Router	Decides whether to teach, quiz, or code	🎛️ *Controller / Router Agent*	A meta-agent that routes user queries to the correct specialized sub-agent.

YouTube Transcript Extraction & Learning Workflow

Auto-Switch: yt_dlp → Whisper
System first tries yt_dlp subtitles (fast + accurate).
If subtitles don't exist, it automatically switches to Whisper AI.
Whisper downloads audio → speech-to-text transcription.


YouTube URL
Check for Subtitles (yt_dlp)
Subtitles Found (yt_dlp)
Use Subtitle Text
NLP Cleaning
Chunking (spaCy)
chunks.txt Saved
Transcript Vectorstore Built
RAG → Generate Explanation
Append to aws_knowledge_base.txt
New KB Vectorstore Version Created

Chunk Storage
Saves chunks into chunks.txt
Also saves cleaned version → clean_transcript.txt
Format compatible with Chroma vectorstore.

NLP Cleanup & Chunking
Removes timestamps, noise, brackets, symbols.
Normalizes text → lowercased + clean.
Splits transcript into semantic chunks using spaCy.

Adding as New Knowledge
Transcript chunks → embedded → stored in vectorstore/
RAG Agent uses transcript-based RAG to generate explanation
Explanation is appended to aws_knowledge_base.txt
New KB version is built automatically.

How Hallucinations Are Prevented

1. Strict RAG Enforcement
LLM receives context + instruction:
“If the context does NOT contain the answer, say:
I don’t see that in my available knowledge.”
2. AWS KB Relevance Check
If topic not in vectorstore →
LLM is NOT allowed to answer directly.
Instead:
→ enters YouTube Learning Mode
→ learns real content → stores it → avoids hallucinated answers.
3. Controlled Prompt Templates
- TEACH, QUIZ, CODE prompts prevent open-ended generation
- Format instructions (JSON) reduce randomness

4. Separation of Knowledge Sources
- AWS KB vectorstore → authoritative content
- Transcript vectorstore → used only during learning
- LLM outputs depend ONLY on retrieved chunks

5. Topic Normalization
Ensures ambiguous user input (e.g., “glue”) becomes “AWS Glue” → relevant retrieval instead of guessing.