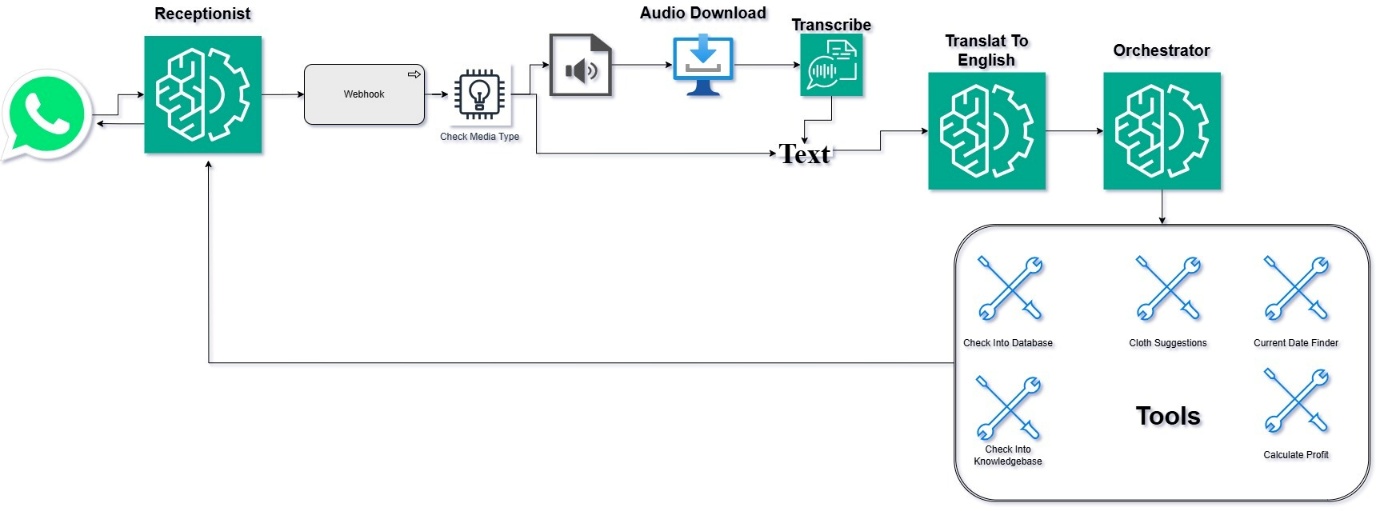
**AI Receptionist System – Technical Documentation**

****

**1. Overview**

This system is a Flask-based backend that integrates with Twilio to handle WhatsApp messages (both text and voice). It intelligently interprets user queries using Groq and responds with relevant information using various internal tools and services. Voice messages are transcribed using OpenAI Whisper. The system can interact with a database, search documents via Pinecone, and respond with tailored messages based on the context.

**2. Flask Setup**

The Flask application includes the following routes:

* /webhook: Main endpoint connected to Twilio for incoming WhatsApp messages.
* /logs: Endpoint to view logs for debugging and monitoring.
* /chat: Optional route for testing chat-based interactions.
* /: Root route, can be used for health checks or welcome message.

**3. Twilio Integration**

Twilio is configured to send incoming WhatsApp messages to the /webhook route.

**Message Handling Logic**

1. **Determine message type:**
   * If text: extract the message body.
   * If voice:
     + Get the media URL from Twilio.
     + Download and convert to WAV format.
     + Transcribe to text using the Whisper API.
2. **Language Detection and Translation:**
   * Use translate\_to\_english(text) if input is not in English.
3. **Orchestration:**
   * Forward translated text to the orchestrator() method of the AIReceptionist class.

**4. AIReceptionist Class Flow**

The orchestrator coordinates the flow of logic and tool execution:

1. Input (text or transcribed voice) is passed to orchestrator(user\_input).
2. Input is translated to English if needed using translate\_to\_english().
3. Tools are identified based on the query using Groq API.
4. Matching tools are executed:
   * check\_in\_db()
   * check\_in\_document()
   * suggest\_clothing\_combination()
   * calculate\_profit()
   * get\_current\_date()
5. The outputs from each tool are combined into a final response.
6. Response is returned to the user over WhatsApp via Twilio.

**5. Tool Functions**

Each tool handles a specific task:

* translate\_to\_english(text): Detect and translate non-English input.
* extract\_parameters(text): Extract relevant parameters like color, material, or quality.
* check\_in\_db(user\_input): Query the database for product rates based on parameters.
* check\_in\_document(user\_input): Query Pinecone vector index for company/document-related questions.
* query\_llm\_with\_chunks(response, groq\_api, user\_input): Use LLM on top Pinecone results.
* suggest\_clothing\_combination(user\_input): Recommend clothing combinations based on text.
* get\_current\_date(user\_input): Return today’s date in a readable format.
* calculate\_profit(user\_input): Compute profit based on cost and selling price in input.

**6. Pinecone Integration**

* Initialize Pinecone index if needed.
* Store vectorized chunks from documents.
* On user query, search top relevant vectors.
* Return top matching results for document-based responses.

**7. Product Database Integration**

* Uses PyMySQL or similar library to connect to the product database.
* Queries based on parameters like color, material, quality.
* Returns rate information in a clean table format.

**8. Voice Transcription with Whisper**

* Voice messages received via Twilio are downloaded.
* Audio is converted to .wav format if needed.
* Audio is transcribed using OpenAI Whisper API.
* Transcription output is used as standard input text.

**9. Logging**

* Logging is handled using Python’s logging module.
* logger.info is used for normal operations.
* logger.error is used for errors and exceptions.
* Logs are viewable at the /logs endpoint.

**10. Flow Summary**

1. User sends WhatsApp message (text or voice).
2. Twilio forwards the message to /webhook.
3. Flask app determines if voice or text.
4. Voice is transcribed if needed.
5. Input is translated to English.
6. Orchestrator detects intent and runs appropriate tools.
7. Tool outputs are combined into a response.
8. Response is sent back to the user via WhatsApp.