Jaicat System Architecture Map

A high-level view of how voice/text flows through Jaicat and how subsystems interact.

```
flowchart LR
    %% Inputs
    subgraph Inputs
      UText[User Text]
      UVoice[User Voice]
      Wake[Wake Word Listener\nservices/wake_word.py]
    end

    %% Frontend IO
    subgraph IO
      STT[STT (speech_recognition)]
      TTS[TTS (pyttsx3)\nMood-adaptive]
      UI[GUI (Tkinter)\nui/JaicatUI.py]
    end

    %% Core Intelligence
    subgraph Core
      NLU[NLU (facebook/bart-large-mnli)\nconversation/nlu.py]
      DM[Dialogue Manager\nconversation/dialogue_manager.py]
      NLG[NLG / Contextual Responder\nconversation/nlg.py]
      CTRL[Controller\napp/controller.py]
      ROUTER[Command Router\nCore/command_router.py]
    end

    %% Knowledge + Memory
    subgraph Knowledge & Memory
      KB[RAG / KB (FAISS)\nkb/**]
      MEM[User Memory (encrypted JSON)\nservices/memory.py]
      LOGS[Logs & Telemetry]
    end

    %% Services
    subgraph Services
      SP[Spotify]
      WEATH[Weather]
      CAL[Calendar]
      CV[Computer Vision\nYOLOv7/11 + FaceID]
      ONED[OneDrive Sync]
      ML[Local ML (TF-IDF, spaCy, LSTM)\nmachine_learning/**]
    end

    %% LLMs
    subgraph LLMs
      OLL[Ollama (local HTTP)]
      HF[Transformers (online)]
```

```
end

%% Flows
UVoice --> Wake --> STT --> NLU
UText --> NLU
NLU --> DM
DM --> CTRL
CTRL --> ROUTER

%% Router to Services/KB/LLM
ROUTER -->|action| SP
ROUTER -->|action| WEATH
ROUTER -->|action| CAL
ROUTER -->|vision| CV
ROUTER -->|files| ONED
ROUTER -->|analytics| ML

%% Reasoning/Answers
DM -->|query| KB
DM -->|context & answer| NLG
NLG --> OLL
NLG --> HF

%% Output path
NLG --> CTRL
CTRL --> UI
CTRL --> TTS

%% Memory & Logs
CTRL --- MEM
CTRL --- LOGS

%% Feedback loops
UI -->|manual input| NLU
CV -->|events| DM
```

What moves where (quick legend)

- **Inputs → STT/NLU**: Wake word gates the mic; speech goes to STT, then intents/entities via **NLU**.
- **Dialogue Manager**: Decides if this is a direct command (router) or needs knowledge/LLM.
- **Command Router**: Hands off to **Spotify/Weather/Calendar/OneDrive/Computer Vision/Local ML**.
- **Knowledge & LLMs**: KB (FAISS) augments context; **NLG** can call **Ollama** (local) or **Transformers** (online) to craft replies.
- **Output**: Controller pushes responses to **UI** and voice via **TTS**; mood/voice adjusted by sentiment.
- **State**: **Memory** stores user prefs & history; **Logs** capture telemetry and errors.

Typical voice turn (sequence)

1) *“Jaicat…”* → Wake listener opens mic. 2) Audio → **STT** → text → **NLU** (intent + entities). 3) **Dialogue Manager** picks route: command vs Q&A. 4) If command: **Router → Service** (e.g., Weather) → result. 5) If Q&A: **KB (RAG)** for facts → **NLG** → (Ollama/HF) → response. 6) **Controller** formats reply → **UI** + **TTS**. 7) **Memory** updated (prefs, context); **Logs** recorded.

Notes & options

- **Offline-first** path uses STT→NLU→NLG→Ollama; cloud only when needed.
- **Computer Vision** can inject events (e.g., car detected) into **Dialogue Manager** for proactive alerts.
- **Security**: prefer encrypted JSON for user profiles; API keys via env vars.