**Automotive Voice Assistant**

**Overview:**

An intelligent voice-driven assistant system that executes user commands via speech or text, leveraging natural language understanding, web automation, and application control. It enables users to search the web, play media via YouTube/Spotify, open websites/apps, and receive audio feedback using Whisper and gTTS.

**File Structure Summary:**

|  |  |
| --- | --- |
| **File/Folder** | **Purpose** |
| vEngine.py | Core backend with Flask API, intent detection, and command execution |
| intent.py | Handles web automation using Selenium for search, YouTube, Spotify, etc. |
| speechToText.py | Captures audio, converts it to text using Whisper |
| textToSpeech.py | Converts assistant responses to audio using gTTS |
| path.py | Locates .exe paths of Windows apps for execution |
| templates/index.html | UI template for user interaction with voice/text inputs |
| static/styles.css | Custom UI styling |
| static/script.js | Frontend logic to capture user input and display assistant responses |
| data.csv | CSV file with predefined commands for intent matching |

**Backend Modules:**

1. **vEngine.py:**

* **Purpose:** Main Flask app that orchestrates recording, intent parsing, media execution, and voice/text-based interaction.
* **Key Components:**
  + **Flask Routes:**
    - **/text**: Accepts a JSON question, detects intent, executes action.
    - **/start-**recording / /stop-recording: Handles speech input asynchronously.
    - **/:** Renders the HTML frontend.
  + **Intent Parsing:** detect\_intent(question) analyzes text for "play", "open", "search", etc.
  + **Task Execution:** perform\_action(intent, question) dispatches to Spotify, YouTube, Google, or apps based on classified intent.
  + **Command Matching:** Uses SentenceTransformer + cosine\_similarity to match response against data.csv entries and provide a confidence score.
* **Packages:**

|  |  |
| --- | --- |
| **Package Name** | **Purpose** |
| sentence\_transformers | Provides pretrained models like all-MiniLLM-v2 for encoding text |
| cosine\_similarity from sklearn.metrics.pairwise | Measures siilarity between transcribed response and predefined commands |
| speechToText | Contains record\_audio,recognize\_speech and transcribe\_audio functions using Whisper |
| textToSpeech | Conatins speak function to generate and play audio responses |
| flask | Web framework to create APIs and render the HTML interface |
| Request,jsonify,render\_template | Flask submodules used for handling HTTP requests, JSON responses, and HTML rendering |
| pandas | Loads and processes the data.csv file containing predefined commands |
| time | Used for delays (eg, sleep before media playback) |

1. **intent.py:**

* **Purpose:** Executes search and media-based tasks using Selenium.
* **Key Functions:**
  + **search\_google(query):** Opens a browser, performs search, and scrapes concise answers using BeautifulSoup.
  + **play\_youtube(query):** Searches and plays the top YouTube result in an embedded video player.
  + **open\_website(query):** Launches the first detected domain from the query in a browser.
  + **open\_app(app\_keyword, text=None):** Finds and opens local .exe using find\_exe\_path, optionally types in a message using pyautogui.
  + **play\_in\_spotify(track\_name):** Opens Spotify Web, searches for the track, and triggers playback via UI automation.
* **Packages:**

|  |  |
| --- | --- |
| **Package Name** | **Purpose** |
| selenium | Core library for web automation and browser interaction |
| webdriver,Service | Launches ChromeDriver to control the browser |
| By,Keys | For locationg elements and simulating keyboard input |
| WebDriverWait,EC | Implements wait strategies to ensure elements are interactable |
| BeautifulSoup | Parses HTML responses from search results for relevant answers |
| subprocess | Lanches local applications using .exe paths |
| pyautogui | Simulates keyboard input to type text inside open applications |
| pygetwindow | Brings the target application window to the foreground for typing |
| os | Access local environment paths like AppData |

1. **SpeechToText.py**

* **Purpose:** Records and transcribes voice input using Whisper.
* **Key Functions:**
  + **record\_audio():** Captures 10 seconds of 16kHz mono audio, saves as .wav.
  + **recognize\_speech():** Waits for recording flag, transcribes the temporary file.
  + **transcribe\_audio(path):** Converts a specific audio file to text.
* **Packages:**

|  |  |
| --- | --- |
| **Package Name** | **Purpose** |
| sounddevice | Sounddevice real-time audio from the microphone and streams it into a NumPy array |
| numpy | Handles numeric array transformations, especially to convert recorded audio into bytes |
| wave | Saves raw PCM data as a .wav audio file |
| whisper | Transcribes .wav audio to text using OpenAi’s pretrained Whisper model |

**4. textToSpeech.py:**

* **Purpose:** Generates spoken responses using Google Text-to-Speech (gTTS) and plays them.
* **Highlights:**
  + Uses gTTS to generate an MP3
  + Plays via pygame
  + Converts to .wav using pydub for further processing
  + Returns final .wav file path for use with similarity checks
* **Packages:**

|  |  |
| --- | --- |
| **Package Name** | **Purpose** |
| gtts | Generates spoken audio from text using Google Text-to-Speech API |
| pygame | Initializes audio mixer and plays back the generated .mp3 |
| pydub | Converts .mp3 files to .wav format for compatibility with Whisper or similarity scoring |
| os | File operations |
| time | Pauses to prevent file lock issues before file removal |
| io | Creates in-memory byte streams for format conversion |

**5. path.py:**

* **Purpose:** Finds executable paths for launching desktop apps.
* **How It Works:**
  + Attempts where <app> in shell
  + If failed, recursively walks C:/Program Files and AppData
  + Returns full .exe path if found
* **Packages:**

|  |  |
| --- | --- |
| **Package Name** | **Purpose** |
| subprocess | Executes system-level commands to locate executables via shell |
| os | Access and traverses file system directories like Program Files, AppData, etc. And builds absolute paths with oc.path.join |

**Frontend Overview:**

1. **Index.html:**

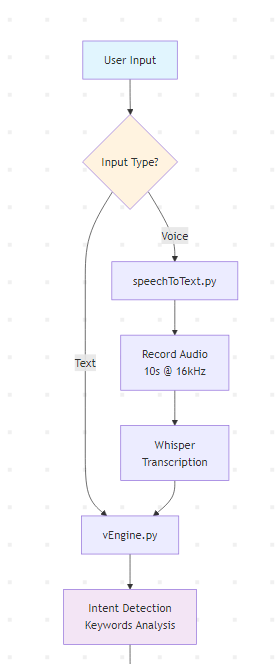
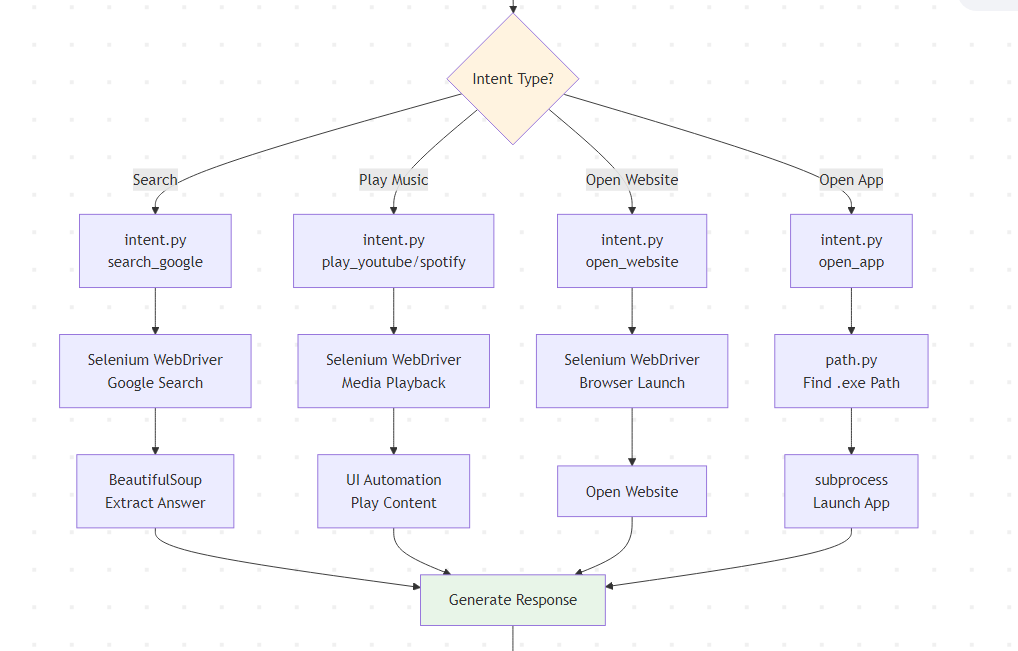
* Bootstrap-based interface with:
  + Title & branding
  + Assistant output panel
  + **Buttons:** Record, Stop, and Text Send

**2. style.css:**

* **Defines:**
  + Layout centering and spacing
  + Button and input field styling
  + Assistant container with card-like aesthetic

**3. script.js:**

* Handles all frontend interactivity:
  + **record-btn:** Starts voice recording
  + **stop-btn:** Stops recording, fetches transcribed intent and assistant reply
  + **send-btn:** Sends text query
* Dynamically updates the #assistant-output section with:
  + Input/response text
  + Match score
  + Colored feedback (green if matched, red if not)

****

