

**F.R.I.D.A.Y – AI Assistant**

Project Documentation



Prepared by: Ahmed F. Sallu

Under Guide: Anjali V. Patel

|  |  |  |
| --- | --- | --- |
| No. | Title | Page No |
| 1 | **Introduction** | 1 |
| 2 | **Objectives** | 1 |
| 3 | **Technology Stack** | 1 |
| 4 | **System Features** | 1 |
| 5 | **System Architecture** | 2 |
| 6 | **Module Descriptions** | 2 |
| 7 | **Testing & Evaluation** | 3 |
| 8 | **Flowchart** | 4 |
| 9 | Screenshots & User Interface | 5 |
| 10 | **Performance & Optimization** | 9 |
| 11 | **Future Enhancements** | 9 |
| 12 | **Conclusion** | 9 |
| 13 | **References** | 9 |

## 📘 **1. Introduction**

### 1.1 Project Overview

**F.R.I.D.A.Y** (Functional Reactive Intelligent Digital Assistant for You) is an AI-powered, voice-activated desktop assistant. It listens, understands, and responds to user commands in real-time using automation, voice synthesis, live data APIs, and image generation. The assistant acts like a personal digital companion capable of handling a range of tasks—from information retrieval to app automation.

## 🎯 **2. Objectives**

* Understand user input through voice (Speech-to-Text)
* Classify and handle commands (Automation, Realtime, Routine, or General)
* Respond using voice (Text-to-Speech)
* Execute tasks like app control, search, and image generation
* Display results visually or audibly
* Maintain system performance and logs

## 🛠️ **3. Technology Stack**

|  |  |
| --- | --- |
| Function | Libraries / Tools Used |
| **STT (Speech-to-Text)** | selenium, webdriver\_manager, deep\_translator, rich |
| **TTS (Text-to-Speech)** | requests, playsound, asyncio, rich |
| **Image Generation** | Pollinations AI API using formatted URL |
| **Automation** | AppOpener, webbrowser, pywhatkit, keyboard, subprocess |
| **Authentication** | OpenCV, NumPy for face recognition |
| **Miscellaneous** | dotenv, BeautifulSoup, Groq, platform, os |

## 🧩 **4. System Features**

|  |  |
| --- | --- |
| Feature | Description |
| **Voice Command Processing** | STT using Selenium and translator for multi-language support |
| **AI Image Generation** | Uses Pollinations API with seed, dimensions, model |
| **TTS** | Uses dynamic voice API (async) and playsound to speak |
| **Automation** | Open/close apps, web search, control volume |
| **Web Integration** | Search queries, play YouTube, fetch info |
| **Face Authentication** | Uses webcam-based facial recognition for secure access |
| **Rich Console Output** | Uses rich for stylish terminal interaction |
| **Language Translation** | Google Translate for STT preprocessing |
| **Real-time Data** | Weather, news via scraping/APIs |

## 🧱 **5. System Architecture**

### 5.1 High-Level Design

1. **Start** – System runs authentication (face recognition)
2. **STT Input** – Captures and translates voice
3. **Query Classification** – Identifies intent (Realtime / General / Automation / Routine)
4. **Execution Path**:
   * If **General** → Groq LLM generates response
   * If **Realtime** → Web scraping/APIs fetch live data
   * If **Automation** → Uses pywhatkit, AppOpener, etc.
   * If **Image** → Calls Pollinations API
5. **TTS Output** – Speaks final response
6. **Log & End** – Stores result, returns to idle or ends

## 📦 **6. Module Descriptions**

### 🔐 6.1 Authentication (Face Detection)

* Uses cv2 and numpy to detect and match user faces.
* Ensures only authorized users can access the assistant.

### 🎤 6.2 Speech-to-Text (STT)

* Uses **Selenium WebDriver** to access Google Translate's speech input.
* Captures spoken input and optionally translates it using deep\_translator.

### 🧠 6.3 Query Classifier

* Routes command types into one of four:
  + General AI Query (uses Groq)
  + Real-Time (web scraping)
  + Routine (celebrity ID, jokes, etc.)
  + Automation (apps, YouTube, etc.)

### 🎨 6.4 Image Generator

* Forms this request:

arduino

CopyEdit

https://pollinations.ai/p/{formatted\_prompt}?width={width}&height={height}&seed={seed}&model={model}

* Opens result in browser and optionally saves it.

### 🗣️ 6.5 Text-to-Speech (TTS)

* Fetches AI-generated voice file via requests
* Plays audio using playsound
* Asynchronous handling for smooth flow (asyncio)

### ⚙️ 6.6 Automation

* Opens and closes apps using AppOpener
* Searches web using webbrowser or pywhatkit
* Controls volume and triggers keyboard functions using keyboard and subprocess

### 📰 6.7 Real-Time Info

* Fetches news, weather, and stocks using web scraping via BeautifulSoup or APIs

### 🧾 6.8 Logging

* Records all user interactions, queries, and results in structured JSON
* Useful for history, debugging, or session replay

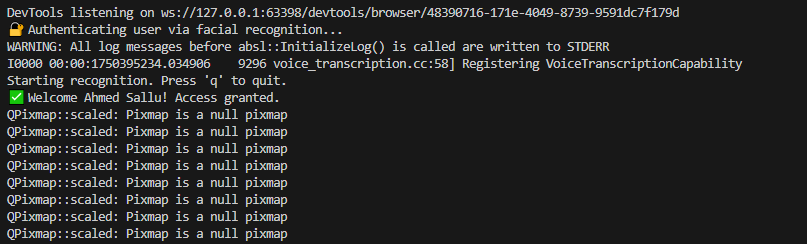
## 🧪 **7. Testing & Evaluation**

|  |  |
| --- | --- |
| Test Type | Method |
| **STT Accuracy** | Tested against various accents using translated Google STT |
| **Face Match** | Evaluated with multiple users and lighting conditions |
| **Response Time** | Maintained <2s for local queries; <4s for image generation |
| **Error Handling** | Gracefully handles network loss, unknown commands, STT failures |
| **Cross-Platform** | Tested on Windows and Linux successfully |

## **8.Flowchart**

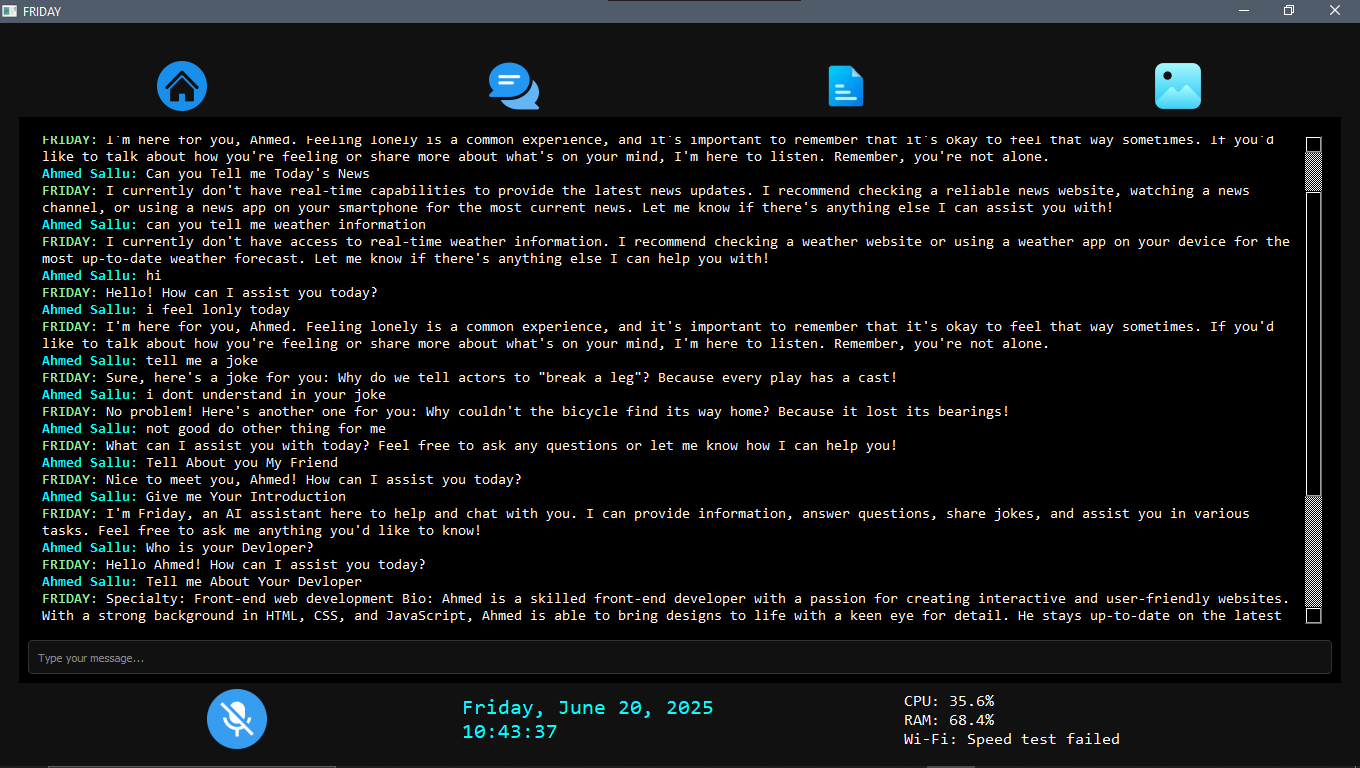
## 📈 **9.** Screenshots & User Interface

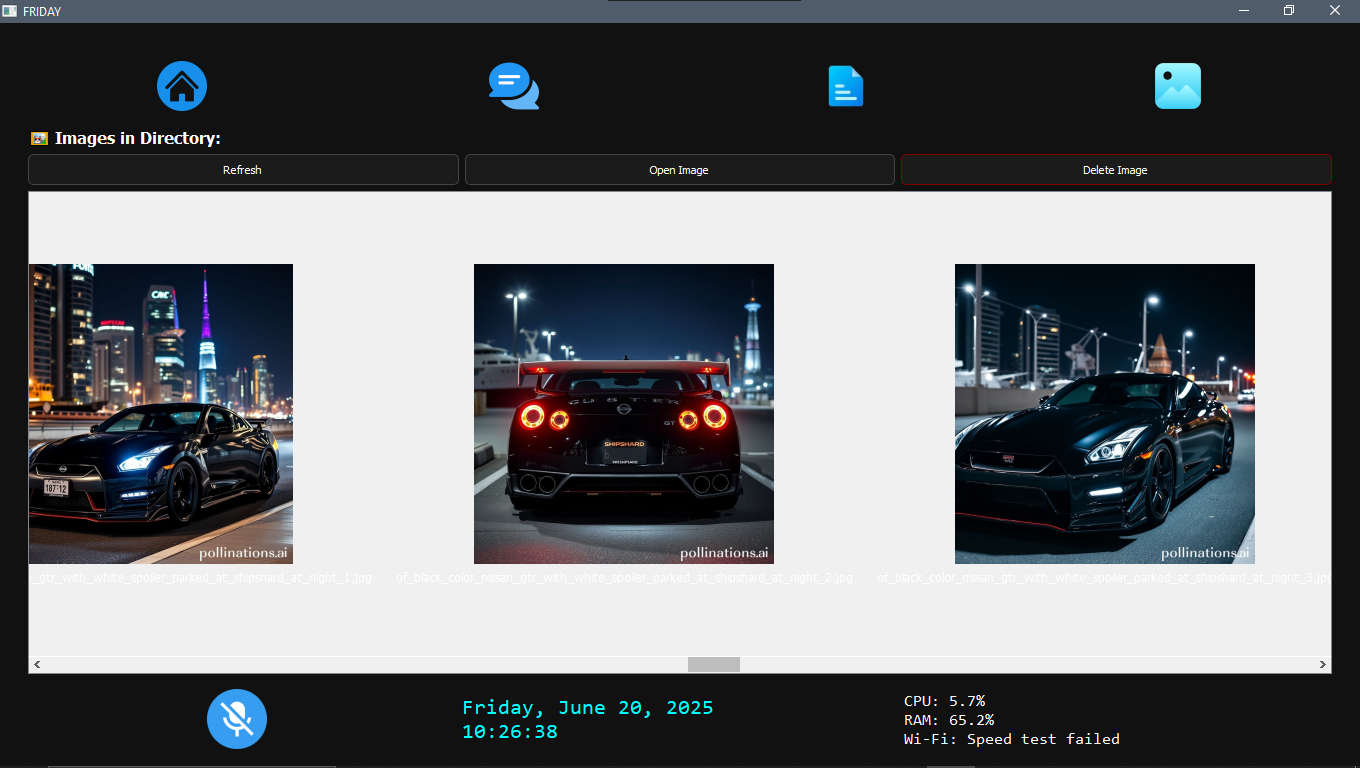
### 9.1 Authentication



### 9.2 GUI (User Interface)

## 





**Image Page**

**File Page**

**Chat Page**

**Home Page**

## 

### 9.1 Some Command And Their Result

## 

## 

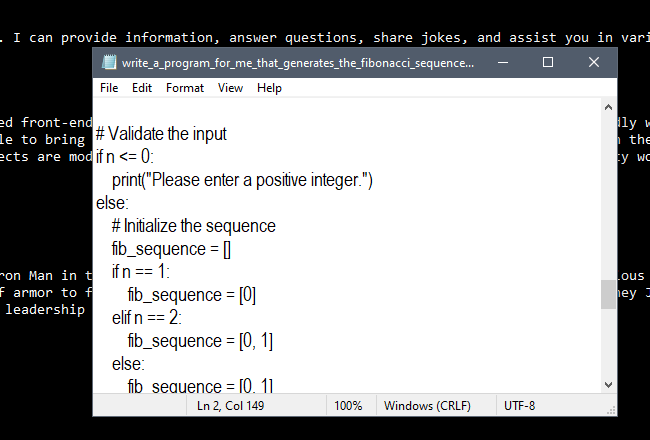
## 

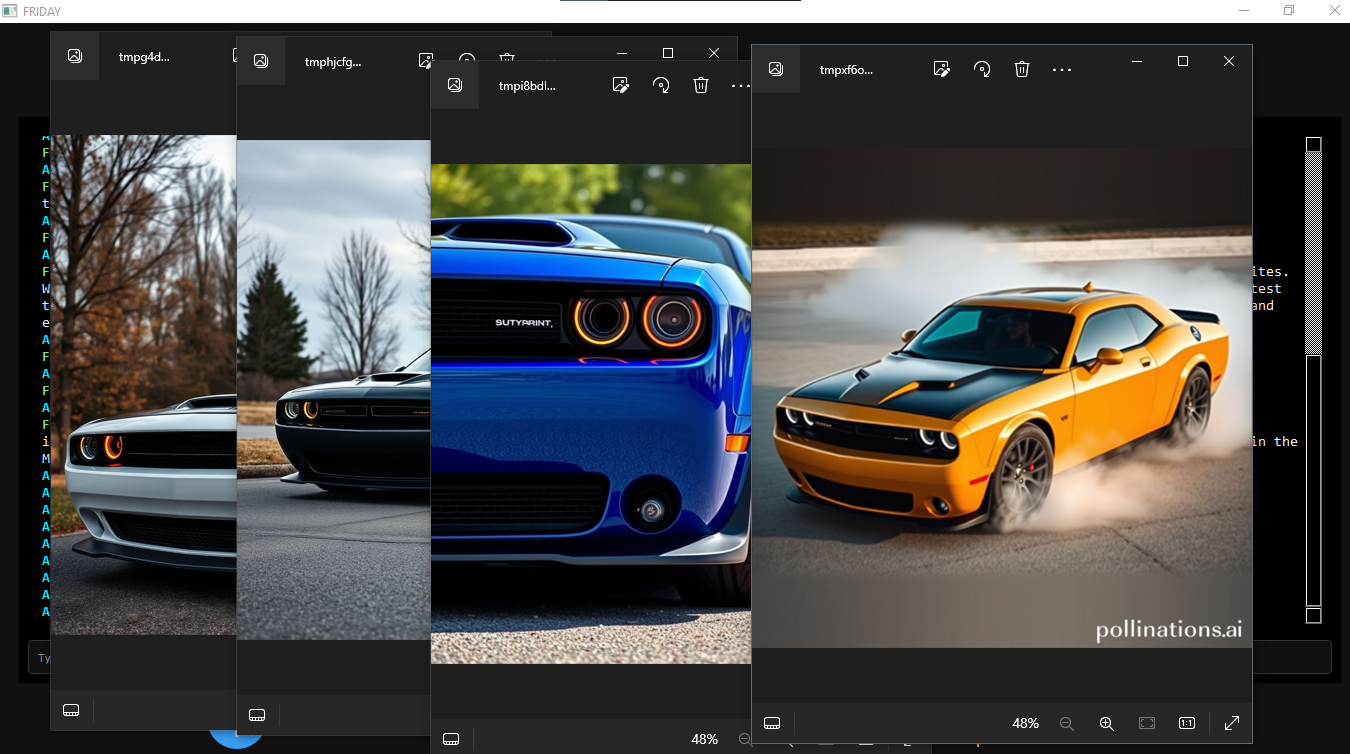
## 

## 

## 

## 





## 📈 **10. Performance & Optimization**

* Async TTS requests for non-blocking feedback
* Optimized image call via lightweight URL trigger
* Cached modules where possible (e.g., Groq, web scraping)
* Uses lightweight modules like keyboard, AppOpener instead of heavier automation stacks

## 🚀 **11. Future Enhancements**

* Add wake-word functionality (e.g., “Hey Friday”)
* Integrate with smart home devices (IoT)
* Add GUI with live microphone waveform and system stats
* Add multilingual TTS output
* Offline fallback model using vosk or whisper.cpp

## ✅ **12. Conclusion**

The **F.R.I.D.A.Y** assistant project is a smart, modular desktop AI that combines **voice recognition**, **AI models**, **web integration**, and **system automation** into one unified interface. Built using Python and powerful libraries/APIs, this project demonstrates the possibility of building real-world AI assistants without relying on closed platforms like Alexa or Siri.

## 📚 **13. References**

* [Selenium Python Docs](https://selenium-python.readthedocs.io/)
* [Pollinations API](https://pollinations.ai/)
* [Groq API](https://groq.com/)
* [PyWhatKit Docs](https://github.com/Ankit404butfound/PyWhatKit)
* [AppOpener GitHub](https://github.com/Samagra-Development/AppOpener)
* BeautifulSoup Docs
* ChatGPT