

**FRIDAY AI ASSISTANT**

TEAM MEMBERS:

Soumika Addala(CH.SC.U4CYS23002)

Vikas Samudrala(CH.SC.U4CYS23042)

Sathvik Valivetty(CH.SC.U4CYS23050)

**Abstract**

*The Friday AI Assistant is a voice-activated, multifunctional AI system designed to enhance productivity, particularly for students and professionals. Built in Python, it integrates voice recognition, natural language processing, and external APIs to handle tasks such as study planning, email management, reminders, web searches, and conversational interactions. The project emphasizes hands-free operation, context awareness, and persistent data storage, making it a robust personal assistant. Key innovations include multi-source intelligence for information gathering, automated PDF report generation, and background services for reminders. This report provides a comprehensive overview of the project's architecture, features, technical implementation, and potential applications.*

**Introduction**

**Project Background**

Friday AI Assistant was developed as a comprehensive voice-based AI tool to streamline daily tasks through natural speech interactions. Inspired by virtual assistants like Siri or Alexa but with a focus on academic and productivity features, it combines speech-to-text, text-to-speech, AI-driven responses, and modular components for extensibility. The system runs on a local machine, supporting cross-platform compatibility (Windows, macOS, Linux), and prioritizes privacy by storing data locally in JSON files.

**Objectives**

* Provide hands-free assistance for study planning, email handling, reminders, and information retrieval.
* Maintain conversation context and user preferences across sessions.
* Integrate external APIs for enhanced functionality while ensuring robust error handling and fallbacks.
* Generate professional outputs like PDF study plans to aid users in organizing tasks.

**Scope**

The project includes core modules for speech processing, memory management, and specialized features like study planning. It does not include advanced machine learning training but leverages pre-built APIs like Google Gemini for AI responses. The repository structure (e.g., folders for `ai`, `memory`, `reminders`, `speech`, `study\_planner`, `system`, `utilities`) reflects a modular design, with key files such as `main.py` for initialization and `config.py` for settings.

**System Architecture**

**High-Level Overview**

Friday operates through a voice command pipeline: wake word detection activates the system, followed by intent recognition and routing to specialized modules. Background threads handle persistent tasks like reminder checks. Data persistence is achieved via JSON files (e.g., `contacts.json`, `reminders.json`), and outputs include both speech and text responses.

**Initialization and Startup**

Upon launch via `main.py`, the system initializes components:

- Speech Recognizer and Text-to-Speech (TTS).

- Memory Manager (loads conversation history from `friday\_memory.json`).

- Study Planner, Reminder Manager, and Gemini Client.

- A background thread starts for reminders.

The system greets the user based on time of day and enters a listening loop.

**Command Processing Pipeline**

**1. Wake Word Detection:** Continuously listens for the activation word (e.g., "Friday").

**2.** **Intent Recognition:** Parses lowered and stripped text for keywords (e.g., "write email to" routes to email module).

**3.** **Module Execution:** Routes to specific handlers, with fallbacks to AI (Gemini), Wikipedia, or Google search.

**4.** **Response Generation:** Delivers via TTS and text, saving interactions to memory.

**Data Flow**

* Input: Voice → Speech Recognition → Text Command.
* Processing: Intent Analysis → Module Routing → API/Local Execution.
* Output: Text/Speech Response + Data Storage (JSON).
* Background: Continuous monitoring for reminders.

**Core Modules**

**1. Study Planner**

**Functionality**: Creates optimized study plans from voice commands (e.g., subject and exam date).

**Key Processes**s:

* Topic fetching from Google Search, Gemini AI, or internal generation.
* Priority calculation based on exam date and difficulty.
* Hour allocation across days.
* PDF generation with professional tables (using ReportLab or similar).

**Storage:** Plans saved in `study\_plan.json`.

**Innovations:** Natural language date parsing (e.g., "next week" → calculated date).

**2. Email Management**

* **Functionality**: Composes and sends emails via voice (e.g., recipient extraction, subject/content collection) and clearly reads incoming emails aloud for hands-free access.
* **Key Processes**:
* Contact lookup or voice-based addition/validation, stored in contacts.json.
* SMTP integration for sending.
* Templates for quick composition.
* Email retrieval and parsing using IMAP to fetch incoming emails, converting them to text, and using text-to-speech to read them aloud with clear pronunciation.
* **Innovations**: Persistent contact learning for future efficiency, enhanced by real-time email reading to keep users updated without manual checks.

**3. Memory and Context Management**

* **Functionality:** Stores conversations by date in `friday\_memory.json`.
* **Key Processes:**
* Adds Q&A pairs to history.
* Provides context (last 20 exchanges) to Gemini for responses.
* Commands for listing/clearing history.
* **Innovations**: Enables contextual, personalized dialogues

**4. Reminder System**

* **Functionality**: Sets and triggers reminders via natural language (e.g., "in 30 minutes").
* **Key Processes:**
* Time parsing (relative/absolute) using regex.
* Background loop checks and notifies via TTS.
* **Storage**: In `reminders.json`.
* **Innovations**: Non-blocking operation for seamless multitasking.

**5. Multi-Modal Fallback and Utilities**

* **Functionality**: Handles general queries with layered responses (Gemini → Wikipedia → Google suggestion).
* **Additional Features:** File search by keyword (voice command retrieves matching files), weather/holiday info, music playback, system shutdown.
* **Innovations:** Error recovery with retries and context-aware prompts.

**Key Features**

The following table summarizes the main features categorized by functionality:

* + **Voice Interaction:** Wake word detection, continuous recognition, TTS responses, context-aware retries, file search by keyword.
  + **Study Planner:** Automated plan creation, topic research (multi-source), priority scheduling, PDF reports, progress tracking.
* **Email Management:** Voice composition/sending, contact storage/listing,

templates, reads out messages.

* + **Reminders & Memory:** Natural time parsing, background notifications, conversation history management.
  + **Web & Information:** Google/Wikipedia search, weather/holidays, quick site access (e.g., YouTube, Gmail).
  + **Entertainment:** Music playback, playlist support, browser integration.
  + **AI Conversation:** Gemini integration, context maintenance, personalized responses.
  + **System Utilities:** Shutdown control, configuration management, cross-platform support.
  + **Data Management:** JSON persistence, auto-save, PDF export.
  + **Advanced**: Multi-threading, NLP for intents/entities, error recovery.

**Technical Implementation**

**Programming Language and Dependencies**

* + **Language**: Python (100% as per repository).
  + **Key Libraries**(from `requirements.txt` inference): Speech recognition (e.g., SpeechRecognition), TTS (e.g., pyttsx3), PDF generation (e.g., ReportLab), regex for parsing, threading for backgrounds.
  + **APIs**: Google Gemini, OpenWeatherMap, Google Custom Search, Calendarific, SMTP.

**Innovations and Best Practices**

* **Context Awareness:** Uses flattened history for AI queries.
* **Error Handling:** Retry mechanisms, graceful fallbacks.
* **Security:** Local storage, environment variables for API keys.
* **Extensibility:** Modular folders allow easy additions (e.g., new utilities).

**Challenges and Limitation**

**Dependencies on APIs**: Potential failures (e.g., Gemini errors) are mitigated by fallbacks, but require internet.

**Voice Accuracy:** Relies on speech recognition quality; retries help but may frustrate in noisy environments.

**Scalability:** Local storage limits large-scale use; no cloud sync.

**Customization:** While modular, adding new intents requires code changes.

**Conclusion and Recommendations**

Friday AI Assistant represents a sophisticated integration of AI technologies into a user-friendly voice system, excelling in academic and productivity scenarios. Its voice-first design and persistent learning make it ideal for hands-free environments. Future enhancements could include mobile app integration, advanced NLP models, or cloud backups for multi-device use. Overall, this project demonstrates strong potential for real-world application, with a solid foundation in modular architecture and innovative features.