2025

Spok

Spok is a personal, voice-first AI assistant designed to operate entirely offline, ensuring privacy and local control

MADE BY MOEEZ MUFTI

Spok: Voice-First, Locally-Run AI Assistant

Abstract

Spok is a personal, voice-first AI assistant designed to operate entirely offline, ensuring privacy and local control. The project integrates voice recognition, natural language processing, task automation, and AI-driven intelligence to create a robust assistant capable of managing reminders, messages, and information retrieval without relying on cloud services.

Table of Contents

Introduction

The modern technological landscape is rapidly evolving, with artificial intelligence, machine learning, and automation becoming integral to both personal and professional domains. Among these innovations, voice-first AI assistants have emerged as powerful tools for enhancing productivity, enabling seamless interaction with technology, and providing personalized support. The "Spok" project aims to develop a fully offline, locally-run AI assistant that integrates advanced language models, automation workflows, and deep AI capabilities while prioritizing privacy and user control.

This project serves as both a practical implementation of AI and a comprehensive test of software engineering, system integration, and programming skills. By combining Linux-based systems, Python automation, and deep learning models, Spok will provide an interactive and intelligent assistant capable of handling tasks such as reminders, messaging, information retrieval, and other user-defined functionalities. While Python will serve as the primary language for Spok's development, C++ will be explored alongside as an optional skill-building exercise to enhance system-level programming knowledge.

The purpose of this initiative is not only to create a functional prototype but also to explore the challenges of integrating multiple technologies into a cohesive system. It provides an opportunity to gain hands-on experience in Linux environment management, offline AI deployment, and real-world software design, ultimately bridging the gap between theoretical knowledge and applied engineering.

Note: the requirements and scope of this project is subject to change as the project fleshes out.

Project Description

Core Components:

1. **Voice Input/Output:** Offline speech-to-text (STT) and text-to-speech (TTS) modules.
2. **LLM Integration:** Lightweight local LLM for natural language understanding and response generation.
3. **Python Automation:** Automates reminders, file management, notifications, and workflows.
4. **Messaging Integration:** Offline-ready handling of WhatsApp, email, and other APIs.
5. **Modular Architecture:** Expandable system enabling future IoT or Raspberry Pi integration.

Project Objectives:

- Develop a fully offline, voice-first AI assistant.
- Implement a reliable local LLM for natural language processing.

- Integrate task automation and reminder functionalities.
- Enable messaging services with offline-first architecture.
- Create a modular system for future expansion and customization.

Methodology / Approach

Step-by-Step Development Plan:

1. **Requirement Analysis:** Define key functionalities of Spok, including voice recognition, task automation, messaging, reminders, and offline AI capabilities.
    - **SysML Diagram:** *Requirements Diagram* to capture functional and non-functional requirements.
2. **System Design:** Create modular architecture for Spok, detailing each component, data flow, and integration points.
    - **SysML Diagram:** *Block Definition Diagram (BDD)* to represent system components and their relationships.
    - *Internal Block Diagram (IBD)* for detailed interaction between modules (voice module ↔ LLM ↔ automation scripts).
3. **Environment Setup:** Install and configure Linux environment, Python libraries, C++ IDE, and AI/LLM frameworks.
4. **Module Development:**
    - **SysML Diagram:** *Sequence Diagram* to show the interactions between modules during typical use cases.
    - **Voice Input/Output:** Implement offline speech-to-text and text-to-speech modules.
    - **Task Automation:** Develop Python scripts for automation routines.
    - **LLM Integration:** Embed pre-trained local LLM for natural language understanding and responses.
    - **Messaging/Reminders:** Integrate APIs or local modules for notifications and messaging.
5. **Module Integration:** Combine all modules into a cohesive system, ensuring smooth data flow and synchronization.
6. **Testing:** Conduct unit testing for individual modules, followed by integration testing and full system functionality checks.
7. **Iteration and Optimization:** Refine performance, reduce latency, optimize resource usage, and improve accuracy of AI responses.

Technology Stack:

- **Operating System:** Linux (CachyOS or Arch Linux).
- **Programming Languages:** Python (automation, AI integration), C++ (performance-critical modules).
- **Frameworks & Libraries:** PyTorch, TensorFlow, Hugging Face Transformers (for LLM), SpeechRecognition, gTTS or Coqui TTS, and other relevant AI libraries.

- **Version Control:** Git/GitHub for code management and collaboration.

Development Methodology:

- **Agile & Iterative:** Build Spok incrementally, delivering functional modules weekly, allowing adjustments based on testing outcomes and user requirements.

Testing Approach:

- **Unit Testing:** Validate each module independently to ensure correct functionality.
- **Integration Testing:** Confirm modules work together seamlessly.
- **System Testing:** Evaluate complete system performance, responsiveness, and reliability.
- **Edge Cases & Stress Testing:** Test with varied commands, inputs, and workloads to ensure robustness.

6-Week Roadmap (42 Days)

Legend:

- **Linux:** Command-line mastery, package management, shell scripting
- **Python:** Automation scripts, workflow integration
- **C++:** LLM module integration, performance-critical code
- **LLM/AI:** Offline model research, setup, and fine-tuning
- **Spok Core:** Voice input/output, reminders, messaging

Week 1: Environment Setup & Linux Mastery

Day	Hours	Focus
1	2	Linux setup, CLI basics, directories, permissions
2	2	Package management, installing Python, pip, VSCode
3	2	Shell scripting basics, cron jobs, automation setup
4	2	Git setup, repository management, version control
5	2	Python basics review, script execution, debugging
6	2	Python file operations, reading/writing data
7	2	Review Linux + Python setup, create first mini automation script

Week 2: Voice Input/Output & LLM Basics

Day	Hours	Focus
8	2	Install offline STT (e.g., Vosk, Whisper local)

Day	Hours	Focus
9	2	Install offline TTS (e.g., Coqui TTS, pyttsx3)
10	2	Test STT+TTS pipeline in Python
11	2	Intro to lightweight LLMs (e.g., GPT4All, llama.cpp)
12	2	Integrate LLM into Python script, basic Q&A
13	2	Combine STT+LLM+TTS into a small prototype
14	2	Debugging, logging, test voice assistant responses

Week 3: Task Automation & Local Data Management

Day	Hours	Focus
15	2	Automate reminders in Python (local SQLite or JSON)
16	2	Automate file management tasks (rename, move, organize)
17	2	Automate notifications (desktop alerts, TTS reminders)
18	2	Integrate Python scripts with voice commands
19	2	Local data storage, secure handling of tasks/messages
20	2	Test full automation workflow end-to-end
21	2	Refactor and modularize automation scripts

Week 4: Deep AI Capabilities & Enhancements

Day	Hours	Focus
22	2	Enhance LLM responses with context memory
23	2	Add conversational memory for multi-turn dialogs
24	2	Integrate Python logic for intelligent recommendations
25	2	Build a local knowledge base for Spok
26	2	Implement fallback logic for unclear commands
27	2	Test AI reasoning and response quality
28	2	Optimize LLM performance, reduce latency

Week 5: Messaging & Workflow Expansion

Day	Hours	Focus
29	2	Offline-ready messaging setup (WhatsApp/email API mock)
30	2	Automate sending reminders/messages via Python
31	2	Voice commands to trigger messaging
32	2	Integrate Spok core with workflow engine

Day	Hours	Focus
33	2	Test full messaging + automation pipeline
34	2	Debugging, optimize modular code
35	2	Prepare scripts for deployment on Linux/Raspberry Pi

Week 6: Testing, Optimization & Final Deployment

Day	Hours	Focus
36	2	End-to-end testing of Spok features
37	2	Debug memory, resource, and voice processing
38	2	Optimize Python, LLM, and automation performance
39	2	Prepare user guide & documentation
40	2	Prepare demo video & showcase scenarios
41	2	Code refactoring, modularity check, final QA
42	2	Final deployment on local machine or Raspberry Pi

***Note**: The schedule is tentative and may be adjusted due to the unpredictable nature of software development and project management. A finalized timetable will be provided upon completion of the project planning phase.*

Resources and Tools

Hardware:

- Development PC (Linux-compatible)
- Optional: Raspberry Pi for portable deployment or hardware integration

Software:

- Linux Distro: CachyOS, Arch Linux, or Fedora
- IDEs: Visual Studio Code, PyCharm (Python), CLion or VS Code (C++)
- AI & LLM Libraries: PyTorch, TensorFlow, Hugging Face Transformers, Coqui TTS
- Python Libraries: SpeechRecognition, pyttsx3, NumPy, pandas (for automation tasks)

References:

- GitHub repositories for offline LLMs, speech recognition, and Python automation projects
- Research papers on offline AI assistants and lightweight LLM deployment
- Online courses/tutorials on Linux, Python automation, C++, and AI/ML integration

Deliverables

- Fully functional Spok AI assistant running offline.
- Python automation scripts for reminders, messaging, and file management.
- Modular, documented codebase ready for AI enhancements or IoT integration.
- Thorough documentation which would include SysML diagrams, well-documented code, version control and regular updation through GitHub.
- User guide with setup, usage, and troubleshooting instructions.