Jarvis AI Assistant – Starter Documentation

This document serves as a comprehensive guide to understanding and initiating development of the Jarvis AI Assistant project from scratch. It is crafted to be AI-friendly, making it easy for any intelligent system or developer to understand the project architecture, functionalities, and prompts required.

Recommended Python Version

Use Python 3.10 or Python 3.11 for best compatibility. Avoid 3.13 due to several package issues with GUI and AI tools.

Folder Structure

The main project folder is located at: D:/Jarvis_Part_3/ This folder should contain: - intro_launcher.py (for video intro) - agent.py (main entry point) - assets/ (folder for video, audio, and UI assets) - modules/ (contains modular scripts for face UI, animation, etc.) - env/ (Python virtual environment)

Core Features Overview

- ■ **Intro Launcher**: Plays a short animated video on startup using `cv2` or `tkinter` alternative. Audio sync is included.
- ■ **Face UI**: Launches a Tkinter-based face UI for Jarvis. Responsive layout and interactive design.
- ■ **TTS (Text-to-Speech)**: Converts Jarvis's responses to voice using pyttsx3 or gTTS. Controlled start/stop function.
- ■ **Mouth Animation**: Visual mouth animation using waveform bars that animate based on TTS audio—not mic input.
- ■ **Webcam Integration**: Future extension: supports webcam feed for visual interaction.
- ■■ **Modular Architecture**: Each major feature is a separate module for easier maintenance and upgrades.
- ■ **AI Control Prompts**: Each feature is invoked with structured prompts, making it easy for AIs to manage logic flow.
- ■ **PyInstaller Ready**: Project structured to be packaged into an EXE. Use `pyinstaller agent.py --onefile`.

Integrated Prompt System

Each module supports a clearly defined function and usage prompt. Example: - From `agent.py`, launch the intro: `from intro_launcher import play_intro` - Start face UI: `from face_ui import launch_ui` - Trigger mouth animation: `mouth_animator.start()` after TTS starts All animations and UI features should be controlled using .start() / .stop() methods to maintain state integrity.

Deployment & Packaging

1. Create virtual environment inside the main project folder. 2. Install dependencies via requirements.txt. 3. Test modules independently: intro > UI > TTS > Animation. 4. Run: `pyinstaller agent.py --onefile --noconsole --icon=icon.ico` 5. For Android GUI: use Kivy or Flutter alternative. Currently desktop-focused.

Final Note

This documentation is meant to guide any AI or developer to understand the foundation of the Jarvis AI project. It is modular, expandable, and ready for intelligent automation. Feel free to ask any query regarding the design or logic.