Kyle Agent MVP Planning Document

Project Overview

Kyle is an advanced AI-powered personal assistant designed to function similarly to Jarvis from *Iron Man*. This agent will:

- See the user's screen
- Hear and analyze guitar notes played
- Engage in context-aware conversations with long-term memory
- Provide an intuitive UI to indicate when it is listening and speaking
- Leverage the latest OpenAI models for advanced speech and conversation

Scalability and documentation are key priorities, ensuring the project can grow and improve over time.

Technology Stack & Dependencies

Core AI & Machine Learning

- **OpenAI GPT-4 Turbo or newer** - Conversational AI and reasoning
- **Whisper API** - Speech and audio recognition
- **Vector Database (Weaviate, Pinecone, or ChromaDB)** - Long-term memory storage
- **PyTorch/TensorFlow (Optional)** - If additional ML processing is required

UI & Interaction

- **Electron.js or React** - Frontend for displaying when Kyle is listening and speaking
- **TailwindCSS** - UI styling

- **Framer Motion (Optional)** - Smooth UI animations

Screen & Audio Processing

- **Python OpenCV** - For screen capturing and analysis
- **FFmpeg** - Audio processing and enhancement
- **pydub** - Audio manipulation
- **SpeechRecognition (Python)** - For alternative speech-to-text processing

Backend & Storage

- **Node.js (Express.js) or Python (FastAPI/Flask)** - Backend API to process requests
- **MongoDB/PostgreSQL** - Storing user interaction history
- **Redis** - Caching and quick lookups

Development & Deployment

- **Cursor IDE** - AI-powered coding assistant
- **Docker** - Containerized deployment
- **Git & GitHub/GitLab** - Version control
- **Notion or Docusaurus** - Documentation platform

Project Roadmap

Phase 1: MVP Development

1. **Set Up the Development Environment**
    - Install dependencies and configure API keys
    - Set up a project repository with clear structure
2. **Implement Core Functionalities**
    - Screen capturing and processing
    - Audio listening and transcription (Whisper API)

- GPT-powered chatbot with basic long-term memory
- Simple UI with indicators for listening and responding

3. **Testing & Refinements**
   - Test screen/audio input accuracy
   - Improve response time and optimize API calls
   - UI/UX improvements based on testing feedback
4. **Documentation & Deployment**
   - Write detailed setup and usage documentation
   - Prepare for scalability (containerization, cloud deployment options)

Manual Setup Actions

1. Install Dependencies

- Install **Node.js & npm**
- Install **Python & pip**
- Install required libraries:

```
pip install openai whisper pydub opencv-python numpy torch
npm install express mongoose tailwindcss electron
```

- Install **FFmpeg** (for audio processing)
- Install **Docker** (if containerizing the project)

2. Set Up API Keys

- Generate OpenAI API key (GPT + Whisper)
- Configure environment variables for API keys

3. Configure the Development Environment

- Set up **Cursor IDE**

- Clone the repository from GitHub
- Create a `.env` file with necessary credentials

4. Run the Project Locally

- Start the backend server

```
python server.py  # or use FastAPI
```

- Run the frontend UI

```
npm run dev  # if using React/Electron
```

- Test Kyle's listening and response capabilities

5. Deploy & Optimize

- Deploy backend to a cloud service (AWS, DigitalOcean, or Render)
- Optimize API calls to reduce cost and latency
- Continuously update documentation

Future Considerations

- Implement voice synthesis for more natural responses
- Add multi-user support with authentication
- Improve UI with visual analytics for screen/audio inputs
- Train a custom ML model for enhanced guitar note detection

This document will serve as a reference throughout development. Let me know if any adjustments are needed!