Kyle Agent - Development Task List

Phase 1: Project Setup & Environment Configuration

✅ 1. Install Required Software

- Install **Node.js & npm**
- Install **Python & pip**
- Install **Git** and set up a **GitHub/GitLab repository**
- Install **Cursor IDE** for AI-powered coding
- Install **Docker** (if using containerization)

✅ 2. Install Dependencies

- Install Python libraries:

```
sh
CopyEdit
pip install openai whisper pydub opencv-python numpy torch
```

- Install Node.js libraries:

```
sh
CopyEdit
npm install express mongoose tailwindcss electron
```

- Install **FFmpeg** for audio processing

✅ 3. Set Up API Keys

- Generate **OpenAI API Key** (for GPT + Whisper)
- Store API keys securely in a `.env` file

✅ 4. Configure the Project Repository

- Clone the project from GitHub
- Set up a clear folder structure:

```
bash
CopyEdit
/kyle-agent
  ├── backend/ (Python or Node.js)
  ├── frontend/ (React/Electron)
  ├── docs/ (Documentation)
  ├── .env
  ├── README.md
```

- Add initial documentation in **Notion or Docusaurus**

Phase 2: Backend Development (Core AI & Processing)

✅ 5. Implement AI Processing (Speech & Screen Recognition)

- Set up **Whisper API** for real-time audio transcription
- Implement **OpenCV** for screen analysis
- Test audio and screen input

✅ 6. Develop Long-Term Memory System

- Choose a vector database: **Weaviate, Pinecone, or ChromaDB**
- Store and retrieve conversation history efficiently

✅ 7. Create the Backend API

- Use **FastAPI** (Python) or **Express.js** (Node.js)
- Expose API endpoints for:
  - Listening to audio (`/listen`)

- Processing screen input (`/screen`)
- Responding with AI (`/respond`)

✅ **8. Implement Logging & Debugging**

- Add error handling & logging
- Test responses and refine accuracy

Phase 3: UI Development (Listening & Speaking Interface)

✅ **9. Design & Develop the UI**

- Use **React.js with TailwindCSS**
- Implement **Electron.js** for a desktop app
- Add listening & speaking indicators

✅ **10. Integrate Backend with Frontend**

- Connect API endpoints for real-time interaction
- Display Kyle's status (Listening, Processing, Speaking)

✅ **11. UI Testing & Refinements**

- Test UI responsiveness & usability
- Optimize performance

Phase 4: Testing & Optimization

✅ **12. Run Local Tests**

- Test Kyle’s **audio recognition**
- Verify **screen capture functionality**
- Ensure **long-term memory retrieval works correctly**

✅ **13. Optimize API Calls & Performance**

- Reduce unnecessary API calls to **cut costs**

- Implement caching with **Redis** for faster responses

✅ **14. Bug Fixing & Debugging**

- Identify & fix errors
- Improve response speed

✅ **15. Write Complete Documentation**

- Update **installation/setup guides**
- Document **API endpoints** and **UI functionality**

Phase 5: Deployment & Scalability

✅ **16. Set Up Cloud Deployment**

- Choose hosting: **AWS, DigitalOcean, or Render**
- Deploy the backend server
- Deploy frontend as an **Electron app**

✅ **17. Optimize for Scalability**

- Enable **containerization** with Docker
- Set up **automatic logging & monitoring**

✅ **18. Release MVP & Gather Feedback**

- Test real-world usage
- Collect feedback & prioritize improvements

Future Enhancements (Post-MVP)

➡️ **19. Implement Advanced Features**

- Add **voice synthesis** for speaking responses
- Improve UI with **analytics & customization**
- Enhance **guitar note detection & feedback**

Tracking Progress

✅ Completed tasks

➡️ Next steps

Let me know if you want anything adjusted! 🚀