SilentScribe – Technical Specification Document

## 📘 Overview

SilentScribe is a fullstack AI-powered narration assistant that takes source code as input (Python scripts or Jupyter notebooks) and generates structured, natural-language voiceover scripts suitable for screencasts, tutorials, or documentation. It aims to support content creators, developers, and educators in simplifying code explanation workflows, now delivered through a responsive web interface.

## 🎯 Project Goals

- Automate narration scripting for educational and documentation purposes.  
- Provide tone-adaptive, context-aware narration that reflects a human teaching style.  
- Build a modern, fullstack web platform for accessible narration generation.  
- Serve as a stepping stone toward multimodal code education (e.g., video, audio).

## 👤 User Personas

🎓 The Educator  
- Needs: Explain code clearly to students with varying skill levels  
- Uses SilentScribe to: Auto-generate walkthroughs for lecture materials  
  
🎥 The Dev YouTuber  
- Needs: Speed up content creation and voiceover writing  
- Uses SilentScribe to: Script tutorials or coding challenges with tone customization  
  
🧪 The Indie Hacker  
- Needs: Communicate code logic to collaborators and investors  
- Uses SilentScribe to: Create voiceover-ready walkthroughs or MVP demos  
  
📚 The Technical Writer  
- Needs: Document complex codebases for manuals or onboarding  
- Uses SilentScribe to: Auto-generate draft text for further editing

## 📋 Functional Requirements

Inputs:  
- Python `.py` scripts  
- Jupyter Notebooks `.ipynb`  
- (Optional Future) Git diffs or file changes  
  
Outputs:  
- Markdown narration script (`.md`)  
- Structured JSON narration metadata  
- (Optional) Audio preview using TTS  
  
Features:  
- Code parsing and segmentation (functions, classes, cells)  
- Narration script generation using LLM (OpenAI GPT-4/4o)  
- Multiple tone presets (e.g., beginner-friendly, expert)  
- Section-wise narration blocks  
- Export to Markdown/JSON  
- TTS voice playback preview (optional)  
- Modern frontend for uploading and viewing results

## 🚧 Edge Cases & Considerations

- Highly abstract code: Lacks comments or clear variable names — narration must infer purpose heuristically.  
- Auto-generated or obfuscated code: Might require fallback to basic description.  
- Very large scripts: Should chunk by logical section to avoid token limits.  
- Dynamic runtime code (e.g., eval, exec): Needs sanitization or summary-only mode.  
- Notebook cells with outputs or plots: Consider adding placeholder notes like “Insert plot explanation here.”

## 🏗️ Architecture Overview

SilentScribe is a fullstack application with modular backend and frontend components.  
  
System Components:  
- Frontend (React + TailwindCSS + Next.js)  
- Backend (FastAPI)  
- Worker (optional)  
  
Workflow Diagram:  
Frontend ➝ Backend ➝ Output Layer (Markdown/JSON, TTS)

## 🧪 Development Stack

Frontend:  
- React + Next.js + TailwindCSS  
  
Backend:  
- FastAPI (Python 3.11+)  
- OpenAI GPT-4/4o API  
- ast, nbformat  
- ElevenLabs / pyttsx3 (optional TTS)  
  
Testing & Tooling:  
- Pytest, React Testing Library  
- GitHub Actions CI

## 🖥️ Fullstack Web App Modules

Frontend:  
- File uploader and tone selector  
- Narration result viewer (with copy/export)  
- Optional TTS playback and download button  
  
Backend:  
- File handling and code parsing route  
- Narration generation route  
- Tone preset endpoint  
- TTS conversion API (optional)

## 🔄 Development Phases

Phase 1 – MVP  
- Frontend file upload + backend processing route  
- Code parsing and narration generation  
- Output markdown displayed in UI  
  
Phase 2 – Tone & TTS  
- Implement tone/style presets  
- Optional: Audio playback using simple TTS  
  
Phase 3 – Polishing  
- Export features (download .md or audio)  
- Multi-file support, syntax highlighting  
- Landing page and documentation

## 📂 Example Directory Structure

SilentScribe/  
├── frontend/ # Next.js app  
│ ├── pages/  
│ ├── components/  
│ └── styles/  
├── backend/ # FastAPI app  
│ ├── main.py  
│ ├── parser.py  
│ ├── narrator.py  
│ ├── formatter.py  
│ └── tts.py  
├── shared/  
│ └── presets.py  
├── tests/  
│ └── test\_backend.py  
├── examples/  
│ └── input\_script.py  
├── README.md  
└── requirements.txt

## 📌 Future Ideas

- IDE extension (e.g., VS Code plugin)  
- Support for multiple programming languages  
- Script-to-video generation (w/ visuals + narration)  
- Personalized voice cloning using user samples  
- Multi-user project dashboard with narration history

## 📞 Contact / Contributions

Interested in contributing? Reach out via GitHub issues or [add your contact info here].