CodeWhisperer Local — Your Private, Offline AI Coding Copilot

> *"Run. Break. Fix. Learn. All on your machine. Zero cloud. Zero APIs. Zero data leaks."*

Project Title

CodeWhisperer Local
An intelligent, privacy-first, desktop-native coding assistant powered by Ollama (Mistral) — designed for developers who value control, speed, and security.

Core Vision

A sleek, local-first AI coding environment where you:

- **Write code** in a beautiful editor
- **Run it locally** — errors caught in a secure sandbox
- **Get AI-powered fixes** — not just syntax, but logic, environment, and dependency errors
- **Approve every fix** — full user control, no auto-overwrites
- **Explain code in 2 lines** — instant plain-English summaries
- **Refactor on demand** — cleaner, more efficient, Pythonic code

All 100% offline. All private. All yours.

Features (Small to Big — Nothing Missed)

1. Beautiful Local Code Editor

- **Monaco-based editor** (same as VS Code) with syntax highlighting, bracket matching, line numbers
- **Clean, distraction-free UI** — no clutter, no ads, no bloat
- **Persistent local storage** — your code stays on your machine

2. "Run" Button — Local Execution with Safety

- Runs your code in a **restricted, sandboxed environment** (no `exec()`, no global scope)
- Auto-captures **full runtime output, errors, and tracebacks**
- **Timeout protection** — kills runaway code after 10 seconds
- **Environment isolation** — runs in a clean Python subprocess to avoid polluting your system

3. AI-Powered Auto-Fix (The Heart of the Project)

➢ Error Detection & Context Capture

- Catches **syntax errors, runtime exceptions, logic bugs, and environment errors** (e.g., `ModuleNotFoundError`)
- Captures **full error type, message, traceback, and surrounding code context**
- Detects **missing packages** (`pip install pandas`) and suggests install commands
- Understands **environment context** — virtual env? Conda? Global? — and tailors fixes accordingly

➢ Smart Fix Generation via Ollama (Mistral)

- Sends structured error + code to **locally running Mistral via Ollama**
- Returns **minimal, precise fix** — no essays, no fluff
- Uses **specialized prompts** for different error types (syntax, runtime, import, logic)

➢ User-Controlled Fix Workflow

- **Never auto-applies fixes** — always shows suggestion first
- **"Apply & Run" button** — applies fix + auto-re-runs code
- **"Try Another Fix" button** — sends current code + "This fix failed. Try a different approach." → gets new fix
- **"Copy to Clipboard" button** — for manual pasting
- **Max 3 attempts** — prevents infinite loops. After 3 tries, shows final suggestion + manual help tips

➢ Environment & Dependency Intelligence

- If `import pandas` fails → suggests `pip install pandas`
- If `pip install` fails → suggests `pip install --upgrade pip` or `conda install pandas`
- Adds **pro tips**: "Activate your virtual env first" with OS-specific commands
- Never runs `pip install` without explicit user consent — **safety first**

4. "Explain" Button — 2-Line Code Summary

- Click → AI generates a **crisp, 2-line plain-English summary** of what the code *does* (not how)
- Perfect for legacy code, team projects, or your own forgotten scripts
- **Example:**
  Input: def calculate_discount(price, discount_percent): return price * (1 - discount_percent / 100)
  Output: Applies a percentage discount to a given price. Returns the final amount after discount.

5. "Refactor" Button — Code Beautification

- Click → AI returns a **cleaner, more efficient, Pythonic version** of your code
- Preserves functionality — no behavior change
- **Example:**
  Input: result = [ ]; for i in range(len(items)): if items[i] > 0: result.append(items[i] * 2)
  Output: result = [item * 2 for item in items if item > 0]

6. Output & AI Panel — Clear, Organized Feedback

- **Output Panel**: Shows runtime results (green) or errors (red) — always visible
- **AI Fix Panel**: Appears only on error — clearly labeled "AI SUGGESTED FIX (Attempt X)"
- Shows **fix text + action buttons** — never mixes AI suggestions with your original code
- Includes **"Why This Fix?" tooltip** — hover for context (e.g., "This package is required for dataframes")

7. Safety & Control Features

- **No auto-execution** of fixes or `pip install` commands — user must click
- **No data leaves your machine** — all processing local
- **No cloud APIs** — zero cost, zero latency, zero privacy risk
- **Sandboxed execution** — protects your system from malicious or broken code

8. Extensibility & Future-Proofing

- **Modular design** — easy to add new features:
  - "Optimize Performance" button
  - "Add Type Hints" button
  - "Generate Unit Tests" button
- **Configurable** max attempts, timeout, and sandbox settings
- **Ready for multi-language support** (JS, SQL) — just swap the LLM prompt

---

Technology Stack

AI & Backend

- **LLM**: **Ollama (Mistral)** — runs locally, zero API cost, full privacy
- **Backend Framework**: **FastAPI** — lightweight, async, perfect for local tools
- **Sandbox**: Python `subprocess` + restricted environment — safe code execution
- **Error Parsing**: Custom Python logic to extract error type, message, traceback, and context

Frontend

- **Editor**: **Monaco Editor** — professional-grade, syntax-highlighted code editor
- **UI Framework**: Vanilla **HTML/CSS/JavaScript** — no React/Vue bloat, fast and lightweight
- **Styling**: Clean, modern CSS — responsive, accessible, beautiful

Local Infrastructure

- **Database**: None needed — all state stored in browser `localStorage` or temporary files
- **Package Management**: Integrates with your local `pip` / `conda` — no bundled dependencies
- **Deployment**: Runs 100% on localhost — no server required. Demo with `ngrok` if needed

---

Advantages (Why This Beats Everything Else)

For Developers

- **Privacy**: Your code never leaves your machine — no cloud, no APIs, no data leaks
- **Speed**: Local LLM + local execution = near-instant fixes (no network latency)
- **Control**: You approve every fix — no black-box auto-overwrites
- **Learning**: "Explain" and "Refactor" teach you better coding practices
- **Cost**: Zero API fees — Mistral runs free on your GPU/CPU

For Recruiters & Judges

- **Depth**: Shows mastery of local LLM orchestration, sandboxing, prompt engineering, and UI/UX
- **Originality**: No one in the dataset is building a user-approved, locally-run, agentic code debugger
- **Polish**: Beautiful frontend + professional UX — not a Streamlit/Gradio toy
- **Relevance**: Solves daily frustrations for every developer — not an academic exercise

For the Future

- **On-Device AI Trend**: Matches Apple Intelligence, Microsoft Copilot+ PCs — local, private, powerful
- **Extensible**: Easy to add new features, languages, or integrations
- **Deployable**: Package as a desktop app (Electron) or browser extension

---

Final Note

This isn't just another "AI code assistant."
This is **the future of intelligent, private, local development tools** — built by you, for you, on your machine.

No compromises. No cloud. No kidding.