Chillax.AI

Stop Stressing, Start Understanding

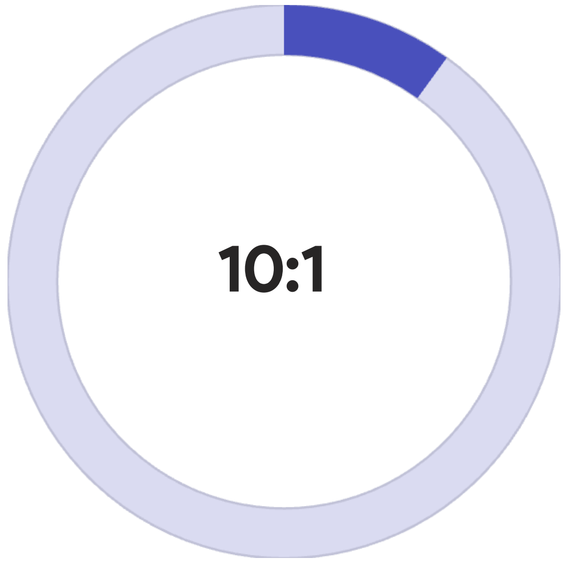
Your offline, AI-powered code companion for untangling legacy systems

Team: **"Pro"crastinators**

ET Gen AI Hackathon — Open Novel Innovation Category | Built for Hackathons 🚀

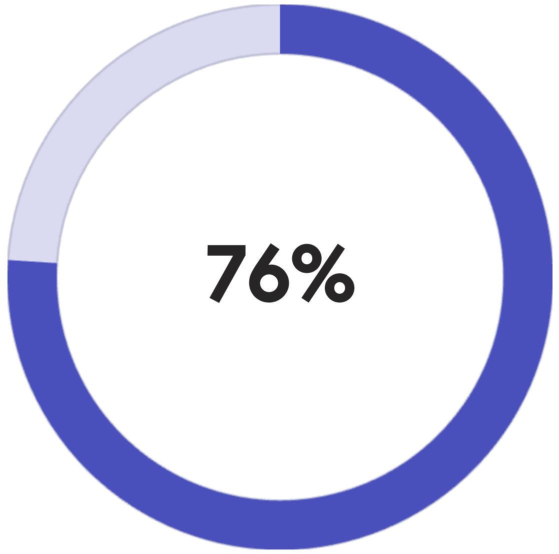
The Problem

Developers are drowning in code they didn't write.



Reading vs Writing Code Ratio
Source: Robert C. Martin, Clean Code

Time for First 3 Meaningful PRs
54% of engineering leaders confirm this (Source: Cortex.io, 2024)



Developers Using Cloud AI Tools
That send proprietary code to external servers (Source: Sourcegraph, 2024)

The Gap

- Existing AI tools like GitHub Copilot and ChatGPT require sending proprietary code to external servers — unacceptable for enterprises in finance, healthcare, and defense.
- 44% of organizations experienced a cloud data breach in 2024, costing an average of $4.88M per breach — Source: IBM/Forbes, 2024

Tools exist to help developers write faster, but nothing helps them **understand existing code at a system level**.

Our Solution

A standalone desktop IDE combining visual mapping, offline AI, and execution visualization.

Interactive Code Map

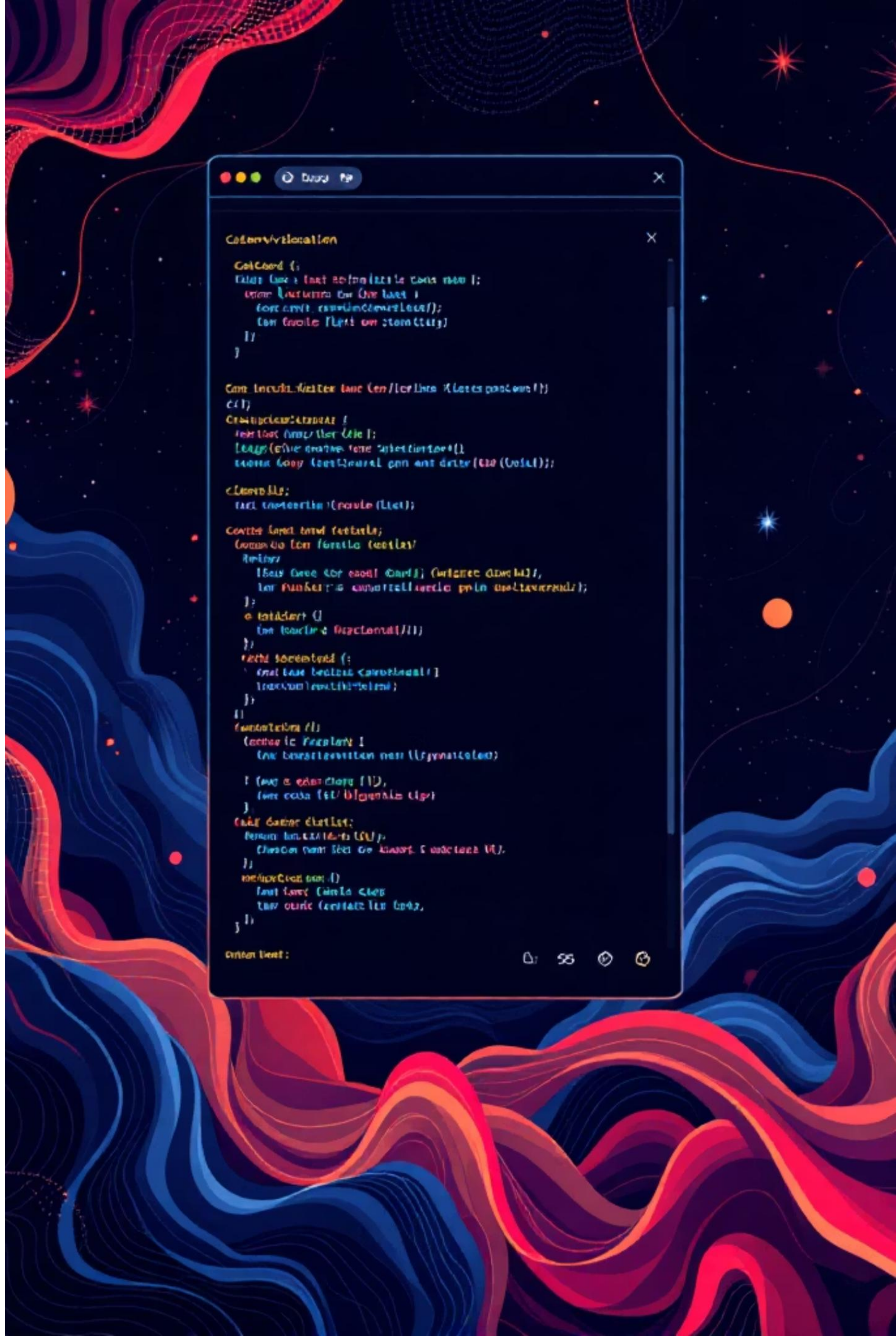
Visual graph showing all modules, functions, classes, and their relationships

Offline AI Assistant

LLaMA 3 running locally explains code in plain English with full context

Execution Visualizer

Animated walkthrough showing code flow with playback controls



How It Works

Visual Graph Theory + GenAI

Unlike chatbots that just output text, **Chillax.AI combines visual mapping with AI comprehension** — we show code execution visually.

Load Codebase
Point to any Python project folder

Analyze Structure
AST parser builds a Knowledge Graph of all functions, classes, and call chains in seconds

Ask Questions
Local LLM answers with full architectural context — no internet needed

Understand Visually
Interactive code map + animated execution flow — not just text

Privacy-First Architecture

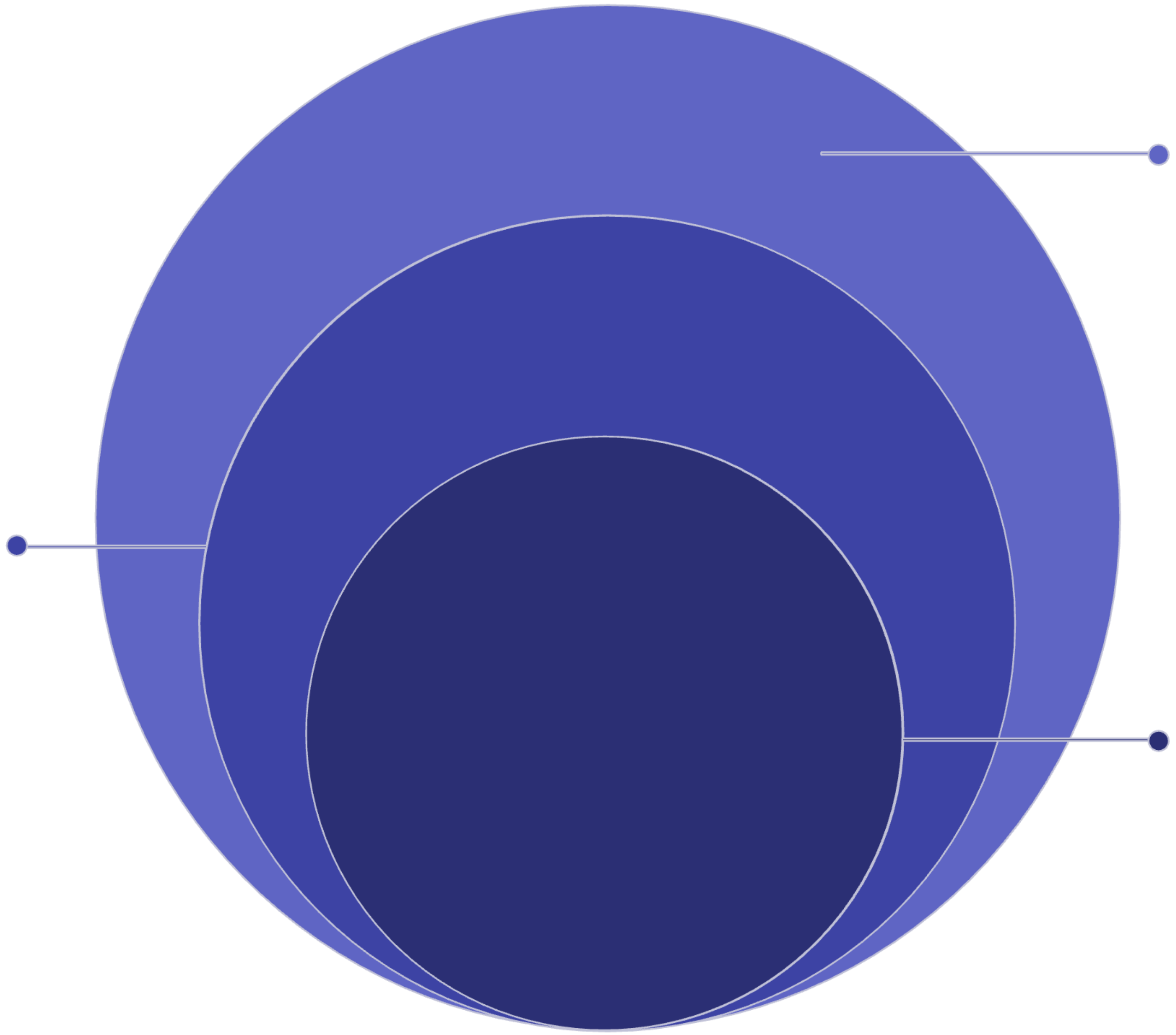
All code analysis and AI processing happens **100% offline** on the developer's local machine.

Zero data leaves the machine. No cloud dependencies. No proprietary code exposed.

AI-using developers reach their 10th PR in 49 days vs 91 days without AI (Source: DX, 2025)

Technical Architecture

Three-Layer Neuro-Symbolic Approach



GenAI + Viz

Generate explanations and interactive visualizations

Knowledge Graph

Represent relations and feed graph to GenAI

Symbolic Analysis

Parse code into AST and extract semantics

Tech Stack: Electron, React 18, Vite, Monaco Editor, FastAPI, Python AST, Ollama, React Flow, Custom dark theme CSS

Validation Strategy

Three complementary approaches to prove effectiveness.

1 Accuracy Testing

Compare AI-generated explanations against ground-truth AST dependency graphs. **Target: 90%+ correctness match with AST ground truth**

2 Performance Benchmarking

Measure inference latency. **Target: sub-5-second responses on standard 8GB RAM laptops**

3 User Study

Simulated onboarding tasks measuring time-to-discovery with vs. without Chillax.AI. **Target: 50% faster onboarding — aligned with DX 2025 study showing AI devs reach 10th PR in 49 days vs 91 days**

Data Sources

- Input: User's own local source code (never uploaded anywhere)
- AI Models: Open-source LLMs via Ollama (LLaMA 3, CodeLlama, Mistral)
- Demo: Bundled sample e-commerce Python project — entirely self-contained, no internet required

Target Users & Use Cases

Privacy-Sensitive Industries

Banking, healthcare, and defense teams working with proprietary, classified, or regulated codebases that cannot leave local machines. (44% of orgs had a cloud breach in 2024 — IBM/Forbes)

Legacy System Maintainers

Engineers supporting decade-old codebases that lack documentation but power critical business operations. (54% of leaders say onboarding takes 1–3 months — Cortex.io, 2024)

Computer Science Educators

Professors and TAs using visualization to teach students how complex systems actually work. (10:1 reading-to-writing ratio makes code comprehension the #1 skill gap)

Use Case Scenario

A developer joins a team with a massive, undocumented 10-year-old Python codebase. They open **Chillax.AI**, visualize the architecture as an interactive graph, and ask "How does the payment module work?" — **all without data leaving their machine**.

Why Chillax.AI?

Feature	GitHub Copilot	ChatGPT	Chillax.AI
Works Offline	❌	❌	✅ 100% Local
Code Visualization	❌	❌	✅ Interactive Graph
Execution Animation	❌	❌	✅ Step-by-Step
Data Privacy	❌ Cloud	❌ Cloud	✅ Zero data leaves machine
Neuro-Symbolic AI	❌	❌	✅ AST + LLM Hybrid

Neuro-Symbolic Innovation
Combine deterministic AST parsing with probabilistic LLMs to reduce hallucinations and increase accuracy.

Visual Execution Flow
Go beyond text explanations with animated code execution visualization showing exactly how systems work.

Truly Offline by Design
Privacy and security aren't features — they're fundamental to our architecture. No cloud dependencies.

Language-Agnostic Future
Python-first, but architecture adapts to JavaScript, Java, C++, Go, and more. Universal legacy platform.

Novelty: Visual Graph Theory + GenAI

This hybrid approach allows us to show code execution visually within a unified interface where developers simultaneously explore architecture graphs, engage the AI assistant, and watch animated execution traces — all on their local machine.

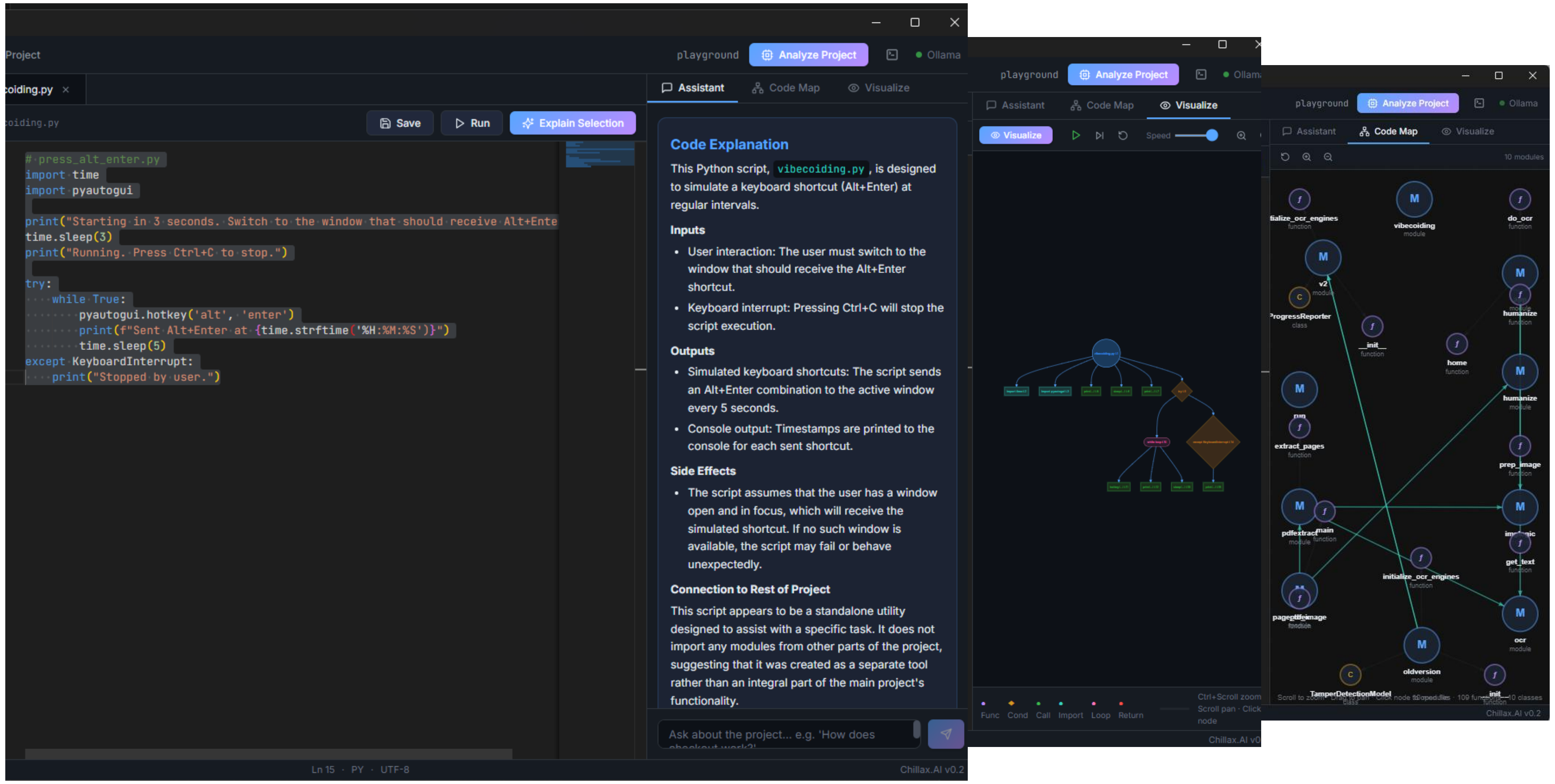
GitHub: github.com/Sansyuh06/Chillax.AI-AI-Based-IDE | Team: **"Pro"crastinators**

Take the Stress Out of Legacy Code

Chillax.AI

Built for the ET Gen AI Hackathon
Open Innovation Category

View on GitHub



```
# press_alt_enter.py
import time
import pyautogui

print("Starting in 3 seconds. Switch to the window that should receive Alt+Ente
time.sleep(3)
print("Running. Press Ctrl+C to stop.")

try:
    while True:
        pyautogui.hotkey('alt', 'enter')
        print(f"Sent Alt+Enter at {time.strftime('%H:%M:%S')}")
        time.sleep(5)
except KeyboardInterrupt:
    print("Stopped by user.")
```

Code Explanation

This Python script, `vibecoiding.py`, is designed to simulate a keyboard shortcut (Alt+Enter) at regular intervals.

Inputs

- User interaction: The user must switch to the window that should receive the Alt+Enter shortcut.
- Keyboard interrupt: Pressing Ctrl+C will stop the script execution.

Outputs

- Simulated keyboard shortcuts: The script sends an Alt+Enter combination to the active window every 5 seconds.
- Console output: Timestamps are printed to the console for each sent shortcut.

Side Effects

- The script assumes that the user has a window open and in focus, which will receive the simulated shortcut. If no such window is available, the script may fail or behave unexpectedly.

Connection to Rest of Project

This script appears to be a standalone utility designed to assist with a specific task. It does not import any modules from other parts of the project, suggesting that it was created as a separate tool rather than an integral part of the main project's functionality.

Ask about the project... e.g. 'How does

Built with ❤️ for ET Gen AI Hackathon 🚀

Team: **"Pro"crastinators** | Contact: santhnu006@outlook.com