Agentathon 2026

Education and Knowledge Management

AnduPandu Society

Amitesh Jammula | C Kaustubh | Abhinav Ballal

PES University

PROBLEM STATEMENT

CodeRonin

Don't just learn to code. Learn to debug under fire.

The only coding platform that fights back.

The Issue: Coding tutorials (Codecademy, Udemy) only teach the "Happy Path", where everything works perfectly.

The Reality: 50-80% of a developer's time is spent on the "Sad Path", debugging weird edge cases, version conflicts, and silent failures, yet almost no platforms train this skill deliberately.

The Consequence: Junior devs freeze when things break because they've <u>never practiced resilience</u>. They rely on ChatGPT to fix errors, outsourcing their critical thinking.

EXISTING SOLUTIONS

LeetCode/Codeforces	Teaches what to write, not what breaks
Tutorial Platforms	Assume perfect inputs & perfect environments
Copilot/Cursor/GPTs	Fixes bugs for you, not with you

CodeMafia: Entropy: Adds beneficial friction. It acts like a Senior Engineer forcing you to fix your own mess.

CodeMafia adds productive struggle.

Encode.ai

OUR APPROACH

PES UNIVERSITY

Core Concept - Single Player Gamified Debugging.

The Mechanic:

User selects a track from an available list of options (e.g., "NumPy Mastery").

User writes code to pass a test case (e.g., "Normalize this Matrix").

The Agentic Twist: As you code, the "Saboteur" Agent activates.

MCP Integration:

The Agent isn't guessing. It uses an MCP Tool (consult_docs) to read a knowledge base of specific library errors.

Example: If you use numpy.dot, the Agent reads the docs via MCP, learns about "Dimension Mismatch," and subtly changes your array shape to trigger that exact error.

Dynamic Difficulty Adjustment (DDA)

Syntax Goblin (Beginner) **-** Teaches: Attention to detail

Logic Gremlin (Intermediate) - Teaches: Understanding error traces.

Semantic Impostor (Advanced) - *Teaches: Deep library comprehension*.

OUR APPROACH

The Player Loop

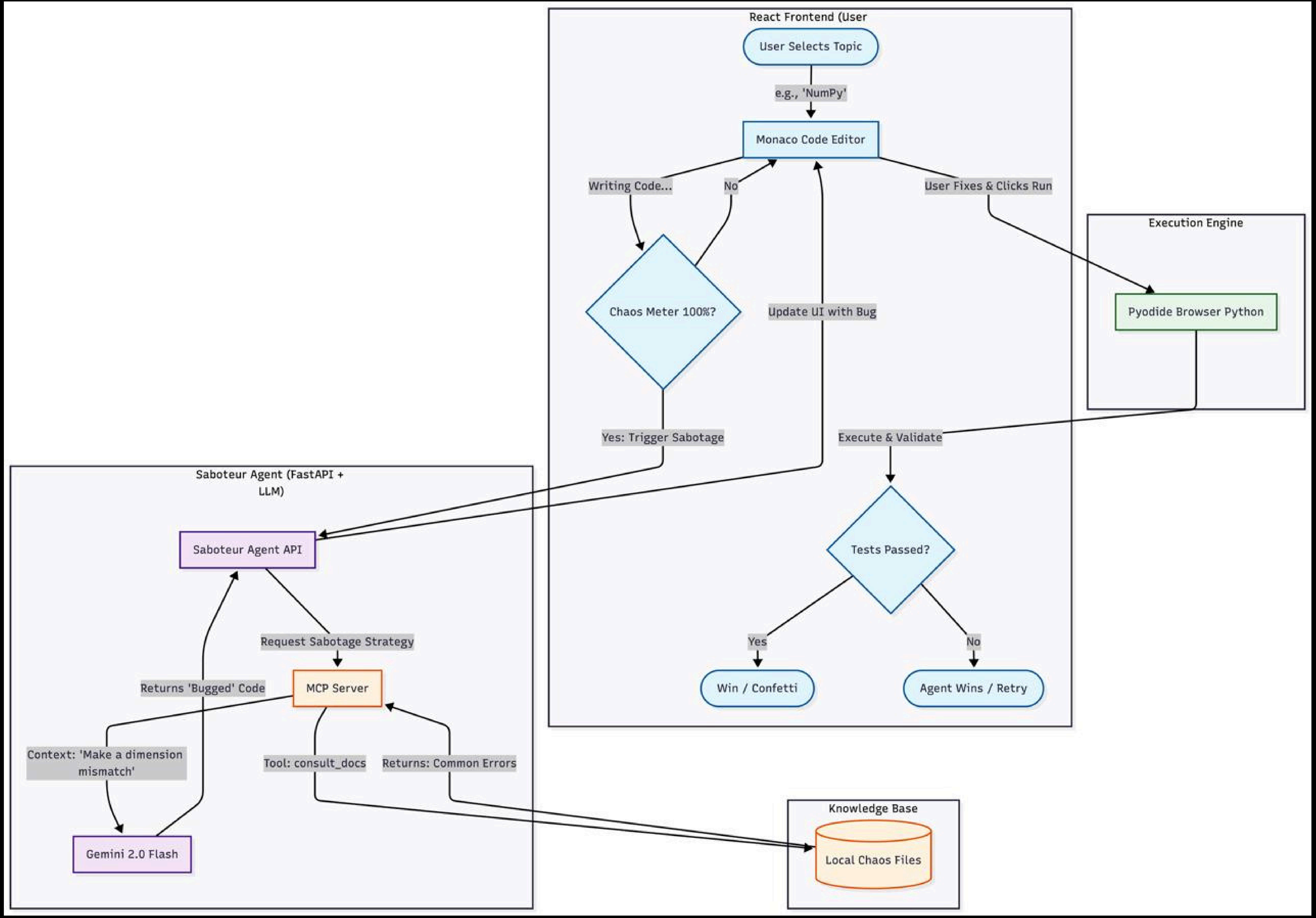
1. Read task
2. Write code
3. Run tests
4. Fix failures

The Agent Loop

1. Observe code
2. Detect library usage
3. Query MCP docs
4. Inject plausible real-world bug

"The Agent behaves like a malicious senior engineer who knows the docs better than you."

DATA FLOW DIAGRAM



Encode.ai

TECH STACK

PES UNIVERSITY

- Frontend: React, Vite, Framer Motion (Glitch effects), Monaco Editor.
- Execution Environment: Pyodide (Runs Python in the browser and is fast and secure).
- Agent Backend: Python (FastAPI).
- AI Logic: Gemini 2.0 Flash (Low Latency).
- Agent Protocol: MCP (Model Context Protocol) connecting the LLM to our Documentation Knowledge Base.

FEASIBILITY

- Client-Side Execution: Pyodide runs Python directly in the browser to eliminate the need for complex server sandboxing.
- Optimized MCP Strategy: The Agent queries curated local Chaos Files instead of live web scraping to ensure zero latency during the demo.
- Prompt-Based Logic: Dynamic Difficulty is implemented by swapping system prompts in real time rather than training resource-heavy models.
- Responsive UX: The Chaos Meter animation masks LLM generation time to keep the gameplay feeling instant and fluid.
- Secure Architecture: Executing code on the client side ensures zero security risk to our backend infrastructure.

RESEARCH AND REFERENCES

https://modelcontextprotocol.io/docs/getting-started/intro

https://pyodide.org/en/stable/

https://github.com/microsoft/monaco-editor

Encode.ai

Agentathon 2026

PES UNIVERSITY

Thank You