Building an Automated AI Code Generation and Validation Agent

Project Development Summary

Abstract

This project presents a proof-of-concept autonomous coding assistant designed to generate code, write tests, run them in a sandbox environment, and iteratively try to fix errors based on failure feedback. This prototype is not intended to guarantee fully functional or production-ready code, but rather to demonstrate the workflow and feasibility of An automated development and refinement process for code.

Large language models power the system for software generation and automated testing. mutation testing, and basic security checks. However, due to the early-stage nature of this prototype, its output may be incomplete or incorrect, and it does not currently ensure successful Resolution of all failures: the final code quality largely depends on the selected model and the Prompt design, and the underlying orchestration logic.

This work initiates the exploration into software automation driven by AI. Significant improvements are still needed, such as better model selection strategies, improved error- Handling logic, reinforcement learning, and more advanced agent architecture. However, The prototype successfully realized an end-to-end concept and demonstrated foundational capability and intent for future development in this field.

1 Introduction

This rapid movement in the development of AI-powered tools presents new opportunities to increase Efficiency in software engineering. This project entails the development and implementation of an AI coding agent, which is capable of of generating, testing, validating, and improving software code on its own. The aim was a prototype that will demonstrate:

- Automated code and unit test generation using a local Large Language Model (LLM)
- Execution within a secure sandboxed environment
- Iterative refinement and automated debugging
- Security scanning and mutation testing
- Web-based graphical interface for non-technical users

This prototype follows research directions in AI-assisted software engineering and serves as a practical basis for further investigation into the context of autonomous multi-agent coding systems.

2 System Design Overview

The system consists of the following components:

- **Local LLM Backend:** leveraging models such as Qwen and Phi via Ollama
- **Agent:** controls test generation, code generation, evaluation, and refinement
- **Sandbox:** Docker-based Python execution environment
- **Validation modules:** Pytest, Bandit (security), pip-audit (dependency audit), Mutmut (mutation testing)
- **User Interface:** Gradio web application with real-time logging and model selection

3 Development Process

The following sequential development stages were executed:

3.1 Environment Setup

A local development environment was configured including:

- Python and virtual environments
- Docker for isolated sandbox execution
- Ollama for local model inference

3.2 Core Agent Implementation

A Python Agent script was developed to:

1. Generate tests from natural-language task descriptions
2. Generate code that satisfies the test suite
3. Execute tests inside a container
4. If failures occur, refine code iteratively via the model

3.3 Security and Reliability Enhancements

The agent was extended to perform:

- *Security scanning* using Bandit
- *Dependency audit* using pip-audit
- *Mutation testing* using Mutmut to evaluate test quality

3.4 User Interface Development

A user-friendly web UI was created using Gradio:

- Model selection dropdown
- Natural language task input
- Live execution logs
- Display of generated solution code

3.5 User Deployment Setup

Cross-platform installation scripts were created for:

- Windows (`.bat`)
- macOS (`.sh`)
- Linux (`.sh`)

These scripts automate installing Python, Docker, Ollama, pulling models, and launching the UI, enabling non-technical users to run the system locally.

4 Conclusion

This project was a very simple proof of concept, meant to showcase the idea of an autonomous coding assistant, and to show our interest and capability in this area. The prototype intended tionally focuses on core functionality rather than scale or sophistication. Despite its simplicity, It effectively automates code generation, testing, refinement, and basic security checks, while providing a user-friendly interface. This provides the ground for exploring more advanced agentic development systems, improved model orchestration, and wider real-world applications in the future. work.