Test-Genie – Automated Test Writer

Problem:

Writing unit tests takes a lot of time, and most developers don't enjoy doing it. Tests break after refactors, edge cases get missed, and small changes in the codebase can create long debugging sessions. Tools that try to generate tests automatically usually give you one batch of tests and stop there. If something fails, you're back to fixing everything by hand.

There's also the issue of safety. Running auto-generated tests directly on your machine can be risky since they might write to the wrong files, install random packages, or behave unpredictably. Because of all this, teams often end up with weaker testing habits, lower coverage, and slower delivery.

Solution:

Test-Genie is designed to take the repetitive work out of testing. It automatically writes tests, runs them, checks the failures, and fixes whatever breaks — repeating the cycle until everything passes. It uses LangGraph to guide the correction steps and runs all tests inside a controlled Docker environment so nothing harmful touches the real machine.

Test-Genie can:

- Generate tests for files.
- Re-write failing tests based on error output
- Improve coverage by spotting untested areas
- Use comments, type hints, and existing patterns to write more realistic tests
- Produce clear summaries: what was generated, what failed, and what was fixed

Developers can use it from the command line

Technical Challenges:

1. Running tests safely
The tool needs a secure environment strong enough to run any code it generates, without letting that code escape the container, write to random directories, or make unauthorized network calls.

2. Designing the self-fixing loop
The agent has to interpret Node error messages, understand what the failure means, and decide whether the test is wrong or the code is wrong. Only the failing part of the test should be rewritten.

3. Matching the user's environment
For tests to run properly, the sandbox has to mimic the user's setup:
– same runtime versions
– same dependencies

4. Producing high-value tests
Generated tests should be stable, predictable, and cover useful cases.

5. Keeping performance reasonable
Running tests again and again takes time, especially on large projects. The agent needs ways to manage containers efficiently, and avoid long loops.

6. Knowing when to stop
Sometimes the code is broken, not the test. Other times the error can't be resolved automatically. The agent needs guardrails so it doesn't rewrite tests forever.

Tech Stack: Node js, langchain, express, docker, llm provider

Test-Genie – Automated Test Writer

Problem:

Writing unit tests takes a lot of time, and most developers don't enjoy doing it. Tests break after refactors, edge cases get missed, and small changes in the codebase can create long debugging sessions. Tools that try to generate tests automatically usually give you one batch of tests and stop there. If something fails, you're back to fixing everything by hand.

There’s also the issue of safety. Running auto-generated tests directly on your machine can be risky since they might write to the wrong files, install random packages, or behave unpredictably. Because of all this, teams often end up with weaker testing habits, lower coverage, and slower delivery.

Solution:

Test-Genie is designed to take the repetitive work out of testing. It automatically writes tests, runs them, checks the failures, and fixes whatever breaks — repeating the cycle until everything passes. It uses LangGraph to guide the correction steps and runs all tests inside a controlled Docker environment so nothing harmful touches the real machine.

Test-Genie can:

- Generate tests for files.
- Re-write failing tests based on error output
- Improve coverage by spotting untested areas
- Use comments, type hints, and existing patterns to write more realistic tests
- Produce clear summaries: what was generated, what failed, and what was fixed

Developers can use it from the command line

Technical Challenges:

1. Running tests safely

The tool needs a secure environment strong enough to run any code it generates, without

letting that code escape the container, write to random directories, or make unauthorized network calls.

2. Designing the self-fixing loop

The agent has to interpret Node error messages, understand what the failure means, and decide whether the test is wrong or the code is wrong. Only the failing part of the test should be rewritten.

3. Matching the user's environment

For tests to run properly, the sandbox has to mimic the user's setup:
– same runtime versions
– same dependencies

4. Producing high-value tests

Generated tests should be stable, predictable, and cover useful cases.

5. Keeping performance reasonable

Running tests again and again takes time, especially on large projects. The agent needs ways to manage containers efficiently, and avoid long loops.

6. Knowing when to stop

Sometimes the code is broken, not the test. Other times the error can't be resolved automatically. The agent needs guardrails so it doesn't rewrite tests forever.

Tech Stack: Node js, langchain, express, docker, llm provider

Test-Genie – Automated Test Writer

Problem:

Writing unit tests takes a lot of time, and most developers don't enjoy doing it. Tests break after refactors, edge cases get missed, and small changes in the codebase can create long debugging sessions. Tools that try to generate tests automatically usually give you one batch of tests and stop there. If something fails, you're back to fixing everything by hand.

There's also the issue of safety. Running auto-generated tests directly on your machine can be risky since they might write to the wrong files, install random packages, or behave unpredictably. Because of all this, teams often end up with weaker testing habits, lower coverage, and slower delivery.

Solution:

Test-Genie is designed to take the repetitive work out of testing. It automatically writes tests, runs them, checks the failures, and fixes whatever breaks — repeating the cycle until everything passes. It uses LangGraph to guide the correction steps and runs all tests inside a controlled Docker environment so nothing harmful touches the real machine.

Test-Genie can:

- Generate tests for files.
- Re-write failing tests based on error output
- Improve coverage by spotting untested areas
- Use comments, type hints, and existing patterns to write more realistic tests
- Produce clear summaries: what was generated, what failed, and what was fixed

Developers can use it from the command line

Technical Challenges:

1. Running tests safely
The tool needs a secure environment strong enough to run any code it generates, without letting that code escape the container, write to random directories, or make unauthorized network calls.

2. Designing the self-fixing loop
The agent has to interpret Node error messages, understand what the failure means, and decide whether the test is wrong or the code is wrong. Only the failing part of the test should be rewritten.