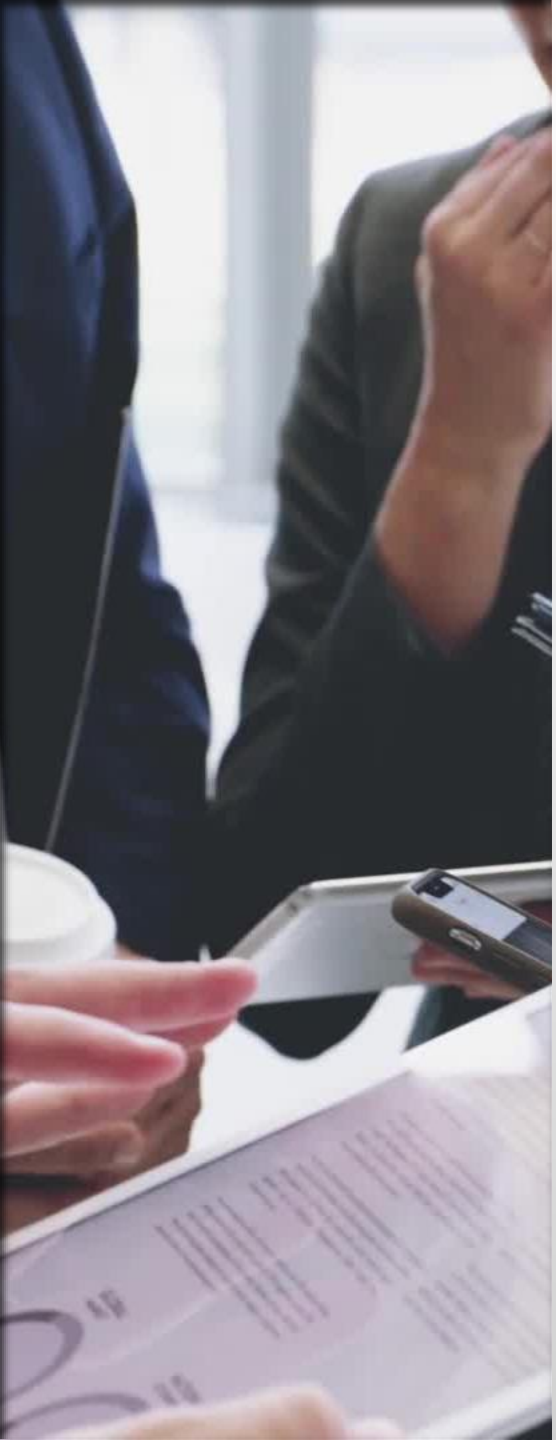
Agentic AI Test Automation Framework

Workflow | Strategy | Tools | Value Proposition

create by Ratna Vamsi



- Manual testing and coverage tracking is time-consuming, error-prone, and hard to scale.
- No streamlined way to convert user stories to automated test cases.
- Difficulty in maintaining test coverage across large, evolving codebases.
- Lack of intelligent feedback loops for flaky test identification or improvement.

Business Problem

Objective of the Project

🔷 Automate test generation using AI for both frontend and backend.

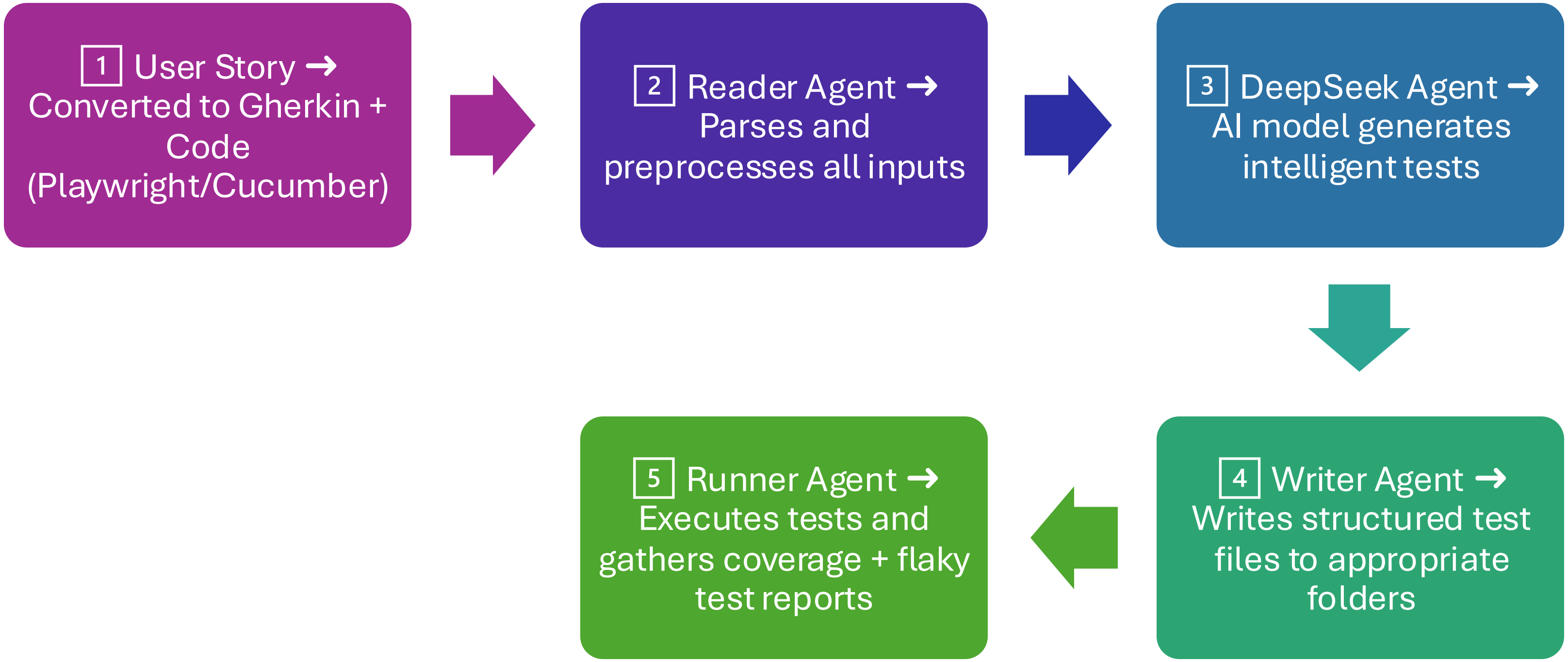
🔷 Convert user stories directly into Gherkin format and generate Playwright/Cucumber tests.

🔷 Measure and improve code coverage automatically.

🔷 Integrate expert reasoning through agent workflows.

Inputs Required

- 📂 Python Codebase – Functional logic for backend/frontend
- 📜 Feature Requirements – Clearly defined user goals/use cases
- ✅ Test Cases – Existing or newly generated tests
- 📊 Code Coverage Report – To track untested parts
- 👥 Team Inputs – Summarized requirements/stories
- 🎯 Vision – Direction from leadership on agent-first design



1 User Story → Converted to Gherkin + Code (Playwright/Cucumber)

2 Reader Agent → Parses and preprocesses all inputs

3 DeepSeek Agent → AI model generates intelligent tests

4 Writer Agent → Writes structured test files to appropriate folders

5 Runner Agent → Executes tests and gathers coverage + flaky test reports

◆ CrewAI – Agentic framework to orchestrate multiple agents

◆ Ollama – Lightweight LLM runner (local or API)

◆ DeepSeek Coder – AI model to generate intelligent test code

◆ Playwright – Frontend testing

◆ Cucumber – Behavior-driven backend testing

◆ Coverage.py – For measuring code coverage

Why Use Agentic Framework?

- ✅ Modular & extensible: Easy to plug in tools/models
- ✅ Human-like workflows: Mirrors real QA/DevOps process
- ✅ Reusability: Agents can be repurposed across projects
- ✅ Automation at scale: Enables test creation without manual effort

Why Not Traditional QA/Testing?

- ❌ High maintenance overhead
- ❌ Delays in test creation from feature changes
- ❌ Manual effort to track flaky or low-coverage tests
- ❌ Inconsistent documentation-to-test translation

Bonus Add-ons (Advanced)

🔷 Self-healing Tests – Agent rewrites failed test logic automatically

🔷 Integration with CI/CD pipelines

🔷 Auto-comment PRs with uncovered logic

🔷 Agentic PR reviewers for quality gating

🔷 Slack/Teams notifications with test insights

Expected Output & Benefits

📁 GHERKIN + PLAYWRIGHT/CUCUMBER TEST FILES

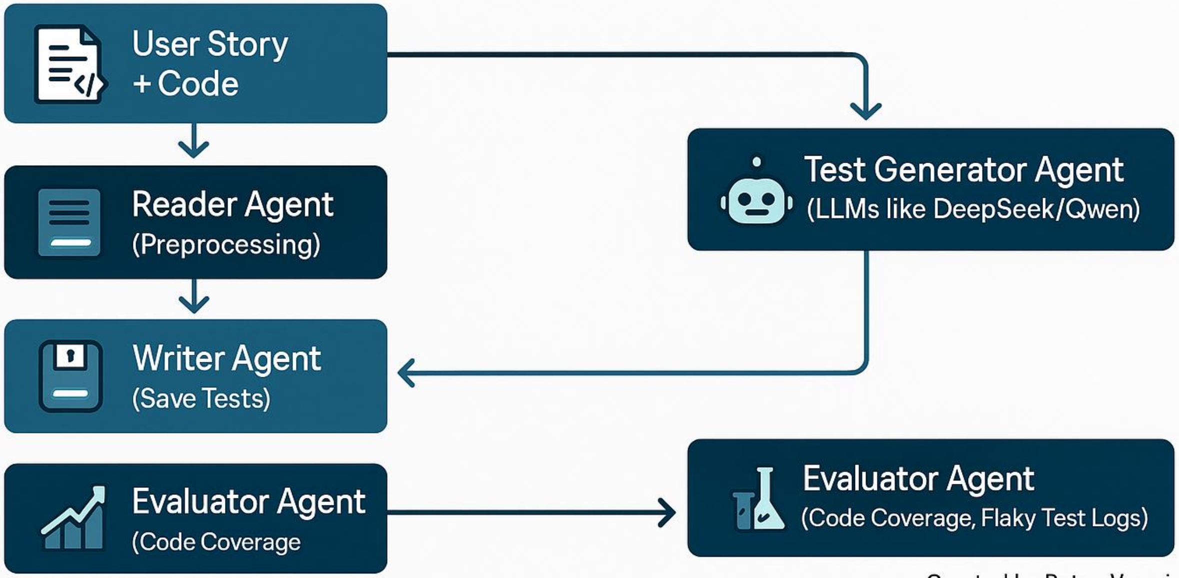
📊 CODE COVERAGE REPORT WITH TEST QUALITY SCORE

🧠 HUMAN-LIKE REASONING IN TEST DESIGN

🚀 FASTER RELEASE CYCLES AND REDUCED QA EFFORT

💡 CONTINUOUS IMPROVEMENT THROUGH AGENTIC FEEDBACK LOOPS

Agentic AI Test Automation Workflow



User Story + Code

Reader Agent (Preprocessing)

Writer Agent (Save Tests)

Evaluator Agent (Code Coverage

Test Generator Agent (LLMs like DeepSeek/Qwen)

Evaluator Agent (Code Coverage, Flaky Test Logs)

Created by Ratna Vamsi