AI-Driven Performance Testing Framework

Architecture, Flow & Automation Assets

Executive Summary

This document describes an **AI-driven performance testing framework** that automates the complete journey from **real user interaction** to **protocol-level load testing**. The framework integrates **Claude AI**, **Playwright**, **Apache JMeter**, and the **Model Context Protocol (MCP)** to eliminate manual scripting and enable intelligent orchestration, execution, and analysis.

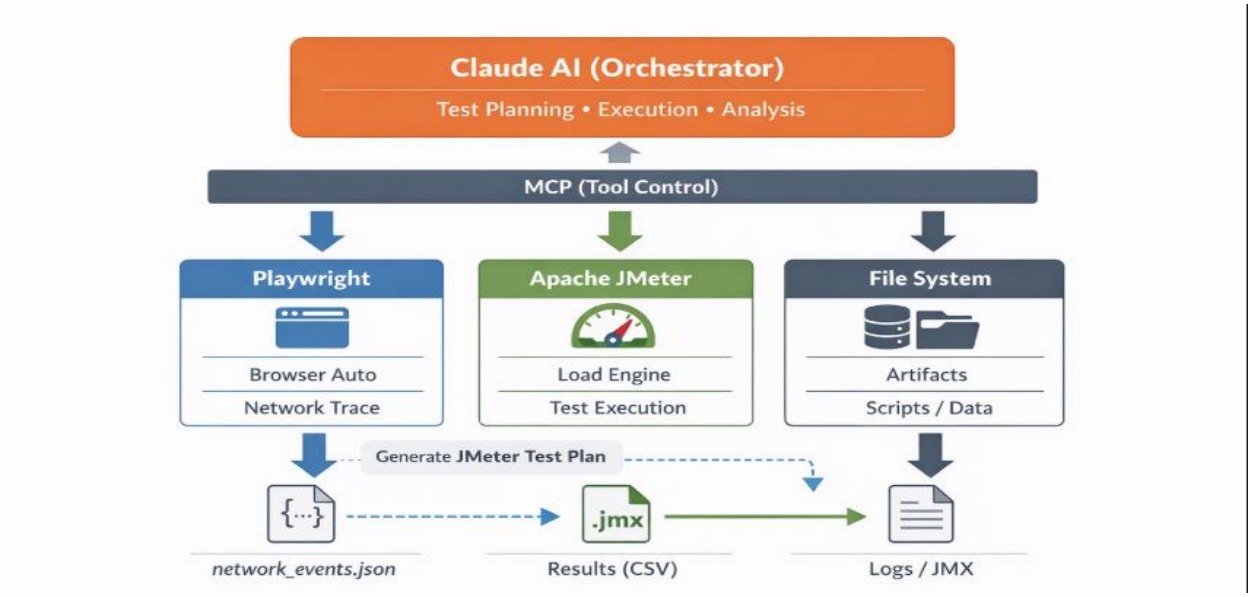
The attached diagrams and automation script demonstrate how **browser network traffic** is transformed into **JMeter (.jmx) test plans** and executed at scale.

1. High-Level Architecture (1.1)

The architecture positions **Claude AI** as the central orchestrator, controlling all tools through MCP while keeping execution engines decoupled and scalable.

Included Asset: Architectural Diagram (Claude AI – Playwright – JMeter – File System)

Key Responsibilities: - **Claude AI** – Test planning, orchestration, execution control, and result analysis - **MCP (Model Context Protocol)** – Secure AI-to-tool communication layer - **Playwright** – Browser automation and network traffic capture - **Apache JMeter** – Protocol-level load test execution - **File System** – Persistent storage for artifacts (JSON, JMX, CSV, logs)



2. End-to-End Execution Flow

This flow illustrates how real user behavior becomes a reusable performance test without manual intervention.

Included Asset: Flow Diagram (User Journey → Network Capture → AI Interpretation → JMeter Execution → AI Analysis)

Flow Steps:

1. **User Journey** – Real application interaction via browser
2. **Network Capture** – HTTP/S traffic recorded by Playwright
3. **AI Interpretation** – Claude AI analyzes traffic and generates test logic
4. **JMeter Execution** – Load test executed at protocol level
5. **AI Analysis** – Intelligent interpretation of performance metrics

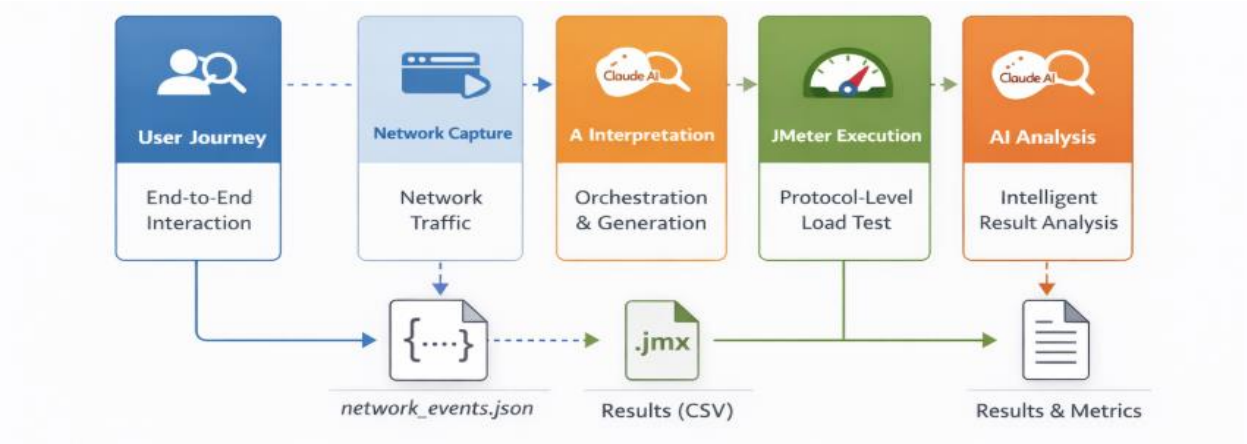
3. Network Events to JMX Generation Flow

The framework uses `network_events.json` as the **single source of truth** for load test creation.

Transformation Pipeline:

```
Playwright Network Capture
        ↓
network_events.json
        ↓
AI-driven request grouping & parameterization
        ↓
JMeter HTTP Samplers & Thread Groups
        ↓
Generated .jmx Test Plan
```

This approach ensures: - High-fidelity request sequencing - Accurate headers and payloads - Easy correlation and parameterization



4. Playwright Network Capture Script

The following script captures browser-level network traffic and stores it in a structured JSON format.

```
const { chromium } = require('playwright');
const fs = require('fs');

(async () => {
  const browser = await chromium.launch({ headless: true });
  const context = await browser.newContext();
  const page = await context.newPage();

  const networkEvents = [];

  page.on('request', request => {
    networkEvents.push({
      type: 'request',
      url: request.url(),
      method: request.method(),
      headers: request.headers(),
      postData: request.postData()
    });
  });

  page.on('response', async response => {
    networkEvents.push({
      type: 'response',
      url: response.url(),
      status: response.status(),
      headers: response.headers()
    });
  });

  // Sample application flow
  await page.goto('https://blazedemo.com/');
  await page.selectOption('select[name="fromPort"]', { index: 1 });
  await page.selectOption('select[name="toPort"]', { index: 2 });
  await page.click('input[type="submit"]');

  await browser.close();

  fs.writeFileSync(
    'network_events.json',
    JSON.stringify(networkEvents, null, 2)
  );
})();
```

5. Generated Artifacts

Artifact	Purpose
`network_events.json`	Captured browser traffic
`.jmx`	Auto-generated JMeter test plan
`.csv`	Performance metrics
Logs	Execution & audit trail

---

6. Benefits of This Approach

- Zero manual JMeter scripting
- Real-user fidelity in load tests
- AI-driven orchestration and analysis
- CI/CD-ready and enterprise friendly

---

7. Conclusion

This framework demonstrates a modern, AI-first approach to performance testing—transforming how load tests are designed, executed, and analyzed by directly leveraging real user behavior and intelligent orchestration.
