Manny Castillo

Senior Software Engineer in Test
Backend & API Test Automation · Data Pipelines · Python · Distributed Systems · CI/CD
Manuel.Franklin.Castillo@gmail.com · linkedin.com/in/manuelfcastillo

SUMMARY

Senior SDET with 14+ years testing backend services, data pipelines, and large-scale distributed systems. API and integration automation in Python, Airflow pipeline validation with nightly runs, and performance work against the services underneath — plus the web and mobile E2E layer on top of it in Playwright and Maestro. Builds the test frameworks and internal tooling other engineers depend on, across enterprise SaaS, gaming infrastructure, storage, and cybersecurity.

SKILLS

Languages: Python, TypeScript, JavaScript, Go, Bash
Backend & API Testing: pytest, API Testing, FastAPI, REST, GraphQL, Integration Testing, Microservices Testing, Postman, Test Plans & Strategy
Data & Pipelines: Airflow, Google Cloud Composer, Data Pipeline Validation, Nightly Regression Runs, Kafka, PostgreSQL, AlloyDB, Elasticsearch
ML & LLM Validation: ML/LLM Output Validation, MLOps Support, Non-deterministic Testing, RAG & Retrieval Grounding, Vector Search (kNN), Self-hosted LLMs (Ollama), LLM-Assisted Test Engineering
Cloud & Infrastructure: AWS (EC2, S3, Lambda, boto3), Google Cloud, Kubernetes, Docker, Linux, Distributed Systems
Performance & Reliability: JMeter, Locust, Grafana, Prometheus, Benchmarking, Metrics Analysis, Automated Failure Triage
CI/CD & DevOps: GitLab CI/CD, GitHub Actions, Jenkins, Git, Secure Workflows, Distributed Test Execution
Web & Mobile UI Testing: Playwright (TypeScript), Maestro (iOS), XCUITest, Selenium, End-to-End (E2E) Testing, Mobile E2E Testing, UI Smoke Testing, Cross-Browser Testing, Regression Testing
Web Technologies: HTML, CSS, JavaScript, REST, GraphQL, OAuth2
Internal Tooling: Chrome Extensions, Three.js, D3.js, Data Visualization, Developer Experience
Hardware & Systems Validation: Enterprise Storage (SLIC, DAE, DPE, SP), Environmental Dwell Testing, VMware ESXi, Debugging, Root Cause Analysis, Reliability Testing

EXPERIENCE

Lead SDET, Medical Team — Backend, Pipeline & UI Automation — Sorcero Inc.

Remote · 2020 — 2026

- Owned test strategy and end-to-end automation for the Medical Team, and mentored engineers across teams.
- Automation reaching past the browser into the REST and GraphQL services, so failures are attributed to the layer that actually broke.
- Tests asserting that Airflow DAGs running in Google Cloud Composer transform and land the data the product depends on — not just that the pipeline ran, but that what came out the other side was correct.
- Scheduled nightly validation against the data pipelines, so breakages introduced during the day are caught overnight and triaged in the morning instead of being discovered in the product.
- Owned validation of model and LLM outputs for the MLOps team — non-deterministic results need a different testing approach than an assert on a fixed value. Also integrated LLM-assisted tooling into day-to-day test engineering.
- Moved suite execution onto Kubernetes so runs scale horizontally instead of serialising on one machine.

- Led the migration to GitLab CI with secure workflows, and wired the suites to execute as part of deployment — on a dedicated dev/QA environment during development, then again on each promotion to staging and to production. The suites gated those promotions in practice: a release waited on a clean run, enforced as a process step rather than as an automated block in the pipeline.
- JMeter and Locust tooling in Python identifying front-end and back-end bottlenecks, feeding directly into optimization work.
- Dashboards plus automated triage workflows, so a red run arrives with a first guess at why rather than a wall of logs.
- End-to-end and UI smoke coverage in TypeScript across two major customer-facing platforms, built alongside the QA team rather than in isolation, replacing a manual regression pass that consumed hours every release cycle.
- Extended automated coverage to the native mobile app, writing E2E flows for iOS in Maestro so mobile regressions surface in the same cycle as web.
- The full smoke suite runs against core platform functionality on each release, covering the paths a user hits first.

Software Engineer in Test — Sony PlayStation
Austin, TX · 2015 — 2020

- Built and enhanced Daruma with reusable helper libraries and AWS tooling; it became shared infrastructure for automated testing beyond the immediate team.
- Wrote pytest-based tests and the test infrastructure around them for microservices spanning the PlayStation Now service platform, including datacenter streaming validation.
- Boto3 automation to provision and validate live game streams against new European datacenters ahead of launch.
- Led QA initiatives supporting SRE through large-scale datacenter migration testing for the expansion of PlayStation Now infrastructure, validating capacity ahead of launch to 700,000+ concurrent users.
- Designed the backend architecture for an in-house dashboard test runner — the system QA teams used to launch suites and read results.
- Extended Prometheus exporters in Go so that quality and reliability signals were measurable rather than anecdotal.

Software QA Engineer (Infrastructure Validation) — Symantec
Boston, MA · 2014 — 2015

- Test plans and Python automation cover enterprise security products.
- Configured and managed the virtualization layer the appliance testing depended on.
- Found and helped close vulnerabilities, improving product resilience.
- Localization testing passes across supported locales.

Software Engineer — EMC
Hopkinton, MA · 2011 — 2013

- Hardware validation and regression testing across SLICs (I/O modules), DAEs (disk array enclosures), DPEs (disk processor enclosures) and storage processors, driven by Python automation frameworks.
- Environmental and thermal dwell testing on enterprise storage enclosures — validating hardware behaviour across temperature extremes, not just software behaviour.
- Modified Jenkins plugins in Python to make the build system support the validation workflows the lab needed.
- Linux builds managed and reproducible.

PROJECTS

Ask the Library — self-hosted AI reading platform over 50,000 books

Personal project · Python, FastAPI, Elasticsearch, kNN Vector Search, RAG, Ollama, React, TypeScript, Playwright, Docker, Stable Diffusion

- Eleven specs covering browse → search → detail → reader, made deterministic by intercepting at the network layer and serving fixtures. No infrastructure required, so it never flakes on a cold GPU or a slow index.
- A second layer pointed at real infrastructure — vector search, LLM generation, the media pipeline — with environment-switchable targets for post-deploy verification. It caught GPU contention starving the search embeddings, which the offline suite by design could not.
- Elasticsearch kNN over locally-computed embeddings — the retrieval layer the answers are grounded in.
- Streaming, citation-grounded Q&A served by Ollama across 14B/3B/1B tiers on a two-node home lab. Nothing is sent to a third-party API.
- One of several infrastructure faults diagnosed end to end, alongside a CUDA/cuDNN conflict silently breaking GPU inference and request starvation from single-GPU contention.
- Public-domain acquisition from the Internet Archive filtered by OCR word-ratio heuristics, perceptual-hash detection of scanner boilerplate, and an LLM sniff test — with on-demand AI cleanup of the OCR text that survives.
- Covers scored by Laplacian variance and a vision model, then regenerated through Stable Diffusion on a local GPU — with a before/after debug page that recomputes the scores client-side so every automated keep-or-replace decision can be audited. It exposed two classifier blind spots that became fixes.
- A study-focused reader with themes, a two-page spread, a chapter-grounded chat panel and a vocabulary builder; plus a discovery UI with semantic search, token auth and saved lists.

Fare — a rideshare app whose economics are actually fair

Personal project · Swift, iOS, TypeScript, Firebase, Vercel, Dispatch

- Native iOS driver app built in Swift.
- A dispatcher-facing web app alongside the driver app, so rides can be assigned and tracked from a desk.
- A middleware service in TypeScript rather than letting clients talk to Firebase directly — the seam where pricing and dispatch rules live.
- Listed as in-progress on purpose. It is a real build with real code, not a shipped product.

Tiếng Việt — a Vietnamese learning app that takes dialect seriously

Personal project · Swift, SwiftUI, Speech Framework, XCTest, XCUITest, iOS

- North, Central (Huế) and South, each voiced by a different native speaker, with a dialect selector wired through the app so a learner can choose the accent they actually need.
- Speaking practice captures the learner saying a word and checks it against on-device speech recognition — feedback on production, not just recognition.
- Game logic lives in its own view model, which is what makes it unit-testable rather than tangled into the view.
- Practice covers words, phrases and full sentences.
- Bundled offline, so lookups work without a network round trip.
- The game view model has unit tests and the app has UI automation — the same discipline applied to a side project as to production work.

Sorceror — Chrome extension for regression testing

Internal tool · Sorcero · Chrome Extension, JavaScript, REST, Caching, Internal Tooling

- Removed the manual token-fetching step that every quality engineer paid on every session.
- Injected an overlay onto the product page exposing what testers actually needed to see — record and article counts, stakeholder counts — none of which the product surfaced on its own.
- Surfaced the ontology backing a project alongside it, so testers could see the structure their test data was shaped by.

- Sorting, management and basic CRUD over the test projects themselves — the setup work happened in the same place as the testing.
- Environment-aware, so the same extension served dev, QA, staging and production without reconfiguration.
- Caching kept the overlay fast enough that using it never cost more time than it saved.

Tesseract — a visual Playwright runner for disaster recovery

Internal tool · Sorcero · Three.js, D3.js, Playwright, TypeScript, Microservices, Disaster Recovery

- Built with Three.js and D3.js so the running system could be seen rather than inferred from logs.
- Bound the visual topology to the actual Playwright repo, so coverage of a given service was a thing you could look at.
- Selecting services selected their specs — orchestration by architecture rather than by remembered file paths.
- During DR exercises, two regions could be run and read against each other directly, which is the question a DR drill is actually asking.
- Persisted the output of each run so drills accumulated into a record that could be compared over time.

EDUCATION

B.S. Computer Science, University of Massachusetts, Lowell, MA — 2012
Study Abroad, Computer Science, American University of Sharjah, Sharjah, U.A.E. — 2008