Test Strategy for Cloud Migration of AI-Supported Case Processing Platform

Natalia Pawlak – Senior QA Lead

Context & Business Goals

Context:

Existing Angular web app for AI-supported, automated case processing in hospitals

Domain is safety-critical & regulated, requires medical expertise

Goal:

Migrate to cloud (AWS) with reliable, secure, multilingual solution

Testing must protect patients, clinicians, and data while enabling fast delivery

Quality Risks & Test Objectives

Risks	Objectives
Incorrect AI or business logic → wrong clinical decisions	Validate functional correctness of workflows & AI guard-rails
Performance issues in peak hospital hours	Ensure non-functional quality: performance, availability, security
Data privacy & regulatory breaches (GDPR, HIPAA-like constraints)	Provide traceability from requirements → test cases → results
Multilingual issues: mistranslations, layout breaks, locale bugs	Enable continuous, automated regression in CI/CD

Test Scope & Levels

Unit tests – Angular components, services, pipes, validators

Component & integration tests – module interactions, API integration

End-to-end (E2E) – user journeys with Playwright (TypeScript)

API & contract tests – ReadyAPI for REST/GraphQL & AI endpoints

Accessibility tests

Performance/load (ReadyAPI / JMeter)

Security (OWASP checks, AWS security tooling)

Environment validation – infrastructure as code, Docker images, AWS deployment smoke tests

Overall Test Approach

Risk-based, focusing on safety-critical flows:

Case creation, triage, AI recommendation, clinician override, audit trail

Shift-left

QA participates in refinement, creates acceptance criteria & Gherkin scenarios

Unit & API tests added during development, not after

Test data strategy

Synthetic & anonymised clinical data only

Data sets per environment & per language (DE, EN, etc.)

Multilingual testing

Locale-specific tests with Playwright: content, formatting, RTL/LTR if needed

Automation Strategy (UI & API)

UI Automation → Playwright + TypeScript

Page Object Model for key areas (Login, Case List, Case Detail, Admin)

Stable selectors with data-testid attributes

Cross-browser (Chromium, WebKit, Firefox) & responsive views

Smoke suite (fast) + regression suite (broader)

API Automation – ReadyAPI

Contract tests against OpenAPI spec

Scenario tests for AI endpoints with “safe defaults” & guard-rails

Performance tests for high-volume case creation/search

Source-controlled in GitHub, executed via GitHub Actions and in Docker

CI/CD & Continuous Testing

GitHub + GitHub Actions for CI/CD

On Pull Request

Install dependencies, run linters & unit tests

Run Playwright smoke and ReadyAPI smoke

Publish reports & screenshots, comment status on PR

On main branch / nightly

Full regression: Playwright regression + ReadyAPI suite

Build Docker image of Angular app, push to Nexus / AWS ECR

Deploy to AWS (e.g., ECS/EKS) test environment

Run post-deploy smoke tests

Continuous feedback

Results sync to Zephyr for traceability

Dashboards for test pass rate, coverage, defect trends

Toolchain Mapping

AWS – hosting app & services (ECS/EKS, RDS, S3, CloudWatch, Secrets Manager)

Docker – container images for app & test runners (Playwright/ReadyAPI)

Nexus – artifact repository for Docker images & versioned test artifacts

GitHub & GitHub Actions – source control & CI/CD orchestration

Playwright (TypeScript) – UI/E2E automation framework

ReadyAPI – API functional & performance testing (Postman maybe should be taken into consideration)

GitHub Copilot with MCP Server – assist in generating tests & boilerplate (code reviewed by QA/Dev)

PactFlow - contract testing

Zephyr (Test Management) – store test cases, link to user stories & automated test results

Environments & Data

Environments

DEV: fast feedback, feature branches, less strict data

QA/STAGE: stable, close to prod, automated regression

PROD: monitored with synthetic probes for key journeys

Data

Seed data via scripts/fixtures in Docker containers

Locale-specific data sets

Sensitive configs in AWS Secrets Manager (not in repos)

Reporting & Metrics

Metrics tracked

Test coverage by risk area

Automated vs manual execution ratio

Defect density & escape rate

Mean time to detect (MTTD) & fix (MTTR)

Requirements → Test Cases → Test Executions mapping

CI results posted via API or plugin

Dashboards for management & auditors

Sample Automated UI Test

Show high-level flow of the mock-up test:

Login as clinician

Open a patient case

Verify AI suggestion displayed & clinician override possible

Confirm audit trail entry created

https://github.com/npawlak/solventum

Risks & Mitigations

AI behaviour changes with new data → strong regression + golden datasets

Complex clinical workflows → close collaboration with medical SMEs

Cloud infra misconfigurations → IaC testing, security scanning

Time-to-market pressure → risk-based prioritisation & automation first

Roadmap / Next Steps

Phase 1: Stabilise core flows & smoke suite in CI

Phase 2: Expand regression, multilingual & performance testing

Phase 3: Increase observability & shift-right testing in production

Continual improvement with feedback from clinicians & operations