# **Quality Health Check & Testing Strategy Template**

# Application:

# Date:

## Why?

The Engineering strategy targets; independent testing & delivery, of high quality software, delivered in a timely fashion, and in a cost effective manner.

## What?

This document serves three aims:

* a template for documenting the PODs approach to testing and quality
* leading the POD into following best quality practices
* identifying gaps between current approach and best practice generating tech debt stories

There is a particular focus on Agile methodology and a strong reliance on test automation.

## Who?

It should be completed by the QA and should be agreed by the whole team.

## Where?

The strategy exists for each Application Under Test(AUT) and should be stored in the respective project code repo.

## When?

* Should be created asap
* Revised when changes in the architecture take place
* Revised when changes in strategy are implemented
* Ad-hoc reviews

# 

# Principles & Processes

|  |  |  |  |
| --- | --- | --- | --- |
| **Area** | **HML** | **Current Strategy** | **Further actions to goal** |
| What application does this strategy cover with the view of supporting independent progress?   * Standalone f-e * Standalone microservice / API * Traditional web app (f-e plus b-e) * Batch / scheduled script * Lambdas / Functions | H |  |  |
| A process for passing domain knowledge (why it’s here and where it fits into architecture) to new POD members. | L |  |  |
| Is acceptance criteria expressed in Gherkin in automation | M |  |  |
| M(etric)1. Code coverage thresholds enforced | M |  |  |
| M2, CI regression test time in CI, target (<5mins) enforced | H |  |  |
| M3. Analyse metrics for bugs escaped into exploratory testing | L |  |  |
| M4. Analyse metrics for bugs escaped into production | M |  |  |
| M5. Analyse metrics Cycle time | H |  |  |
| M6. Analyse metrics Mean Time To Recovery | L |  |  |
| Following the agreed ATDD flow | H |  |  |
| Devs build the all automated tests | M |  |  |
| BDD automation tool used (e.g. Spock, Cucumber) | L |  |  |
| Following the agreed CI Stages & Flow with a well understood branching strategy | M |  |  |
| Test and feature code stored in the same GIT branch | H |  |  |
| Functional tests decoupled from third parties (Independent Progress) with (swagger enabled) stubs | H |  |  |
| Dedicated environments for:   1. Functional CI Automation (stubbed) 2. System CI Automation (non stubbed) 3. System Exploratory (non stubbed) | M |  |  |
| Are you testing against correct version of dependencies | M |  |  |
| An agreed Definition of Ready/Done | H |  |  |
| For APIs, a detailed Swagger/OpenAPI approved by architect / Tech Lead | M |  |  |
| For UI test automation, following Page Object Model | M |  |  |
| Following the agreed Issues Flow | M |  |  |
| Testing GDPR compliance / personal data | H |  |  |
| Where are manual-regression-only (please minimise this) test documented, eg. end-to-end, visual checks | M |  |  |
| Disaster Recovery approach | M |  |  |
| Has strategy been published (GIT) and agreed with POD | M |  |  |
| What is the notification process of build failures to encourage fast remedy | L |  |  |
| Who reviews the code | M |  |  |

# 

# Tests Automation Layers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Area** | **HML** | **Current Strategy** | **Tooling** | **Further actions to goal** |
| Unit | M |  |  |  |
| Code coverage (Unit and functional) | M |  |  |  |
| Static analysis (via CI):   * Code quality / standards * Security exposures * 3rd party dependency validation * Mutation testing | H |  |  |  |
| Functional test (stubbed):   * Component/endpoint behaviour * Journey (state across components/endpoints) * Third party invocation validation * Swagger consistency (API only) | H |  |  |  |
| Dependency integration (you and 3rd parties) | M |  |  |  |
| Downstream system test (if loose contract) | M |  |  |  |
| Consumer integration (3rd parties and you) | M |  |  |  |
| Performance:   * Load & concurrency under CI * Stress / infrastructure resilience * Soak   Levering corporate approach | M |  |  |  |
| Dynamic security / penetration testing | M |  |  |  |
| Browser/OS/device compatibility  and deciding which browsers | M |  |  |  |
| Exploratory being done and by whom? | H |  |  |  |
| Accessibility (f-e) | L |  |  |  |
| Infrastructure resilience testing | L |  |  |  |
| Fallback and recovery testing | M |  |  |  |

# 

# Production Analysis / DevOps

|  |  |  |  |
| --- | --- | --- | --- |
| **Area** | **HML** | **Chosen Strategy** | **Further actions to goal** |
| How will you ensure that the fully tested version of the software is the one that gets released to Prod.  eg. GIT version displayed in the app | H |  |  |
| Is there a documented release process | M |  |  |
| Do you hold a pre-release go/no-go meeting prior to releasing. | H |  |  |
| Can you test in production to minimise risk (scaled release, A/B switches, limit by IP, feature flags etc) | L |  |  |
| Will there be a post-release warranty period, where the POD work closely with Tier1 | L |  |  |
| How will you monitor/alert for issues | M |  |  |
| How will you investigate issues, eg. access to prod logs, databases | H |  |  |
| What is the process for testing ‘hot fixes’ and re-integrating them into the normal flow/branching strategy. | M |  |  |
| How to solicit Prod feedback - to ensure that escaped bugs are analysed and the SDLC reviewed in a timely fashion | L |  |  |
| Is a root cause analysis meeting held following major issues | M |  |  |
| Is there a job that monitors the live app key behaviors? | M |  |  |