**Test framework design and strategy**

**Purpose:**

The purpose of this document is to outline the strategy and design of a comprehensive test framework for an e-commerce application. This framework is intended to support efficient, maintainable, and scalable automated testing across functional areas including UI, API and database validation

This document describes the test approach for each applicable test phase, entry and exit criteria’s, test organization structure and test management activities.

**Scope:**

This strategy applies to the full test automation lifecycle and includes:

* Framework Architecture: Defining folder structure, reusable models and configuration management.
* CI/CD Integration: Enabling automated execution of tests and report generation within continuous integration and deployment workflows.
* Test stability: implementing strategies to handle flaky tests and ensure consistent results across environments.
* Environment support: Managing test execution across multiple environments and configurable setup.
* API Test integration: Designing an approach to include API-level validations as part of the overall test automated strategy.
* Database Interaction: Supporting test data setup and cleanup by interfacing with the application’s database.

**Entry criteria**

* Test Cases/Scripts have been written & reviewed by SIT team. Product Owner to review & provide sign-off for the SIT Test Cases ​
* SIT Environment should be configured correctly before testing ​
* Connectivity with all interfaces should be successful ​
* The Code to be tested should be delivered to the SIT Environment ​
* The Unit testing for the functionalities and modules delivered to SIT should have been completed successfully ​
* The test data required for SIT have been downloaded &converted ​
* All Unit test cases should have been executed & the test evidence should be published to the SIT team ​
* Prepare & request Test Data in-line with the Test Environment​

**Exit criteria**

* All test cases and scripts identified for SIT should been executed & results should be published​
* The SIT test results, and evidence should be signed off by the Test Team
* All defects discovered during SIT have been recorded, assessed and appropriate resolution agreed by Teams ​
* No major defects of severity “Critical” or “High “should be outstanding from SIT.​
* Any outstanding defects from SIT and the risk associated, should be signed off by Teams.​

**Architecture overview**

Approach:

* Use a maintainable page object model
* Organized folder structure

Ecommerce project

* Config : environment configs
* Data: Json
* Pages : pages classes
* Tests: test classes
* Utility : helper that will have error handling and screenshot functions
* Screenshots: saved screenshots
* Run\_test : command line entry

**Test stability & Flaky test reduction**

Approach:

* Use explicit waits
* Avoid hard-coded sleeps
* Implement retry logic for unstable steps
* Run tests in headless mode to CL to reduce UI timing issues

**Test environment management**

Address how you would manage test environments

Approach:

* Dynamically load URLs and test data per environment
* Pass environment from CLI: --env staging
* Use a JSON file (test\_data.json) with base URLs per environment

**Database interaction for test data**

Approach:

* Clean up after tests to keep DB consistent
* Setup data before test and validate after
* Use a database connection utility

**CI/CD Integration**

Approach:

* Use Jenkins
* Steps:
* Run tests on push/pull request
* Set fail/pass thresholds
* Generate reports using pytest-html and send out reports via channels

**API testing into your overall test automation efforts**

Approach:

* Can be executed in headless pipelines
* Reusable methods in api