# CBIIT QA Test Automation Code Review Checklist The purpose of this checklist is to ensure: Code readability: Ensure that code is well-structured, easy to understand, and follows consistent naming conventions. This will make it easier for other testers to read and maintain code.

# Test coverage: Ensure that automation tests cover all relevant scenarios and use cases. Ensure that there are no gaps in the test coverage, and that all critical paths are covered.

# Test data: Ensure that test data used in the automation tests is appropriate, representative, and covers all possible variations. Ensure that the test data is not hardcoded in the tests and that it can be easily modified if needed.

# Error handling: Ensure that automation tests handle errors and exceptions. Ensure that the tests fail with informative error messages.

# Test performance: Ensure that automation tests run efficiently and do not have any performance bottlenecks. Ensure that the tests do not take too long to run.

# Test results: Ensure that automation tests produce accurate and reliable results. Ensure that tests do not produce false positives or false negatives, and that results are consistent across different environments and platforms.

# Code maintainability: Check that automation tests are easy to maintain and update. Ensure that tests are modular and reusable, and that they do not have redundant code.

# Test automation best practices: Ensure that automation tests follow best practices for test automation, such as using page object model, avoiding hardcoding, using descriptive test names, using assertions, etc.

# Before creating a PR, please make sure to have pulled the latest changes from the *master* branch to the branch you are working on and review the following:

## *Naming Convention, File Location, and Classes Location:*

* Ensure proper package name is used.
* Feature belongs under either AnalysisTools.ApplicationName.Features, ServiceNow.ApplicationName.Features, or CustomApps.ApplicationName.Features.
* PageFactory classes belong under PortfolioName.ApplicationName.Pages.
* Steps belong under PortfolioName.ApplicationName.Steps.
* Steps Implementation classes belong under either AnalysisTools.ApplicationName.StepsImplementation, ServiceNow.ApplicationName.StepsImplementation, or CustomApps.ApplicationName.StepsImplementation.
* Utils belong under either AnalysisTools.ApplicationName.Utils, ServiceNow.ApplicationName.Utils, or CustomApps.ApplicationName.Utils.
* Runners belong under either AnalysisTools.AnalysisToolsRunners or ServiceNow.ServiceNowRunners.
* All instances of pages belong in PageInitializer class which can be found under appsCommon package.

*Feature file Practices:*

* Use Declarative testing scenario.
* Tag scenario with User Story ticket number example @SS-3453.
* Tag scenario with Test Case ticket number.
* Tag Scenario with @Smoke for scenarios that fall under smoke test suite.
* Tag scenario with @Regression for scenarios that fall under regression test suite.
* Tag scenario with @Progression only for scenario being worked on.
* Tag scenario you worked on with your NIH user id, example @uddins2.

*PageFactory Practices:*

* @FindBy used for PageFactory.
* Elements in PageFactory have one line block comment above indicating what the element is.
* Every page must have a Non-Parametrized Constructor that initializes the PageFactory.
* Ensure pages class extends CommonUtils.

*Steps and Steps Implementation Practices:*

* Ensure Steps package has hooks class.
* Steps method should be as short as possible. If not use Steps Implementation and call Steps Implementation method in Steps class.
* Ensure both Steps and Steps Implementation class extends PageInitializer.

*Runners Practices:*

* Ensure runners is named RunApplicationNameSmokeTest, RunApplicationNameRegressionTest, and RunApplicationNameProgressionTest.
* Ensure proper tag is used for runners, example tags=@Smoke for RunApplicationNameSmokeTest.
* Use features=”src/test/java/ServiceNow/ApplicationName/Features” or features=”src/test/java/AnalysisTools/ApplicationName/Features”.
* Use glue=”ServiceNow.ApplicationName.Steps” or glue=”AnalysisTools.ApplicationName.Steps”.

*Git and GitHub Practices*

* Prior to committing all changes make sure Jenkins credentials are enabled in localconf.properties file .
* Ensure all changes are committed.
* When committing use message flag and put a message with application name and the change. Example “CEDCD - Added scenarios to user story 1234”. When making a change that is the similar to the previous change use “..” as commit message. Example first commit is “CEDCD - Add wait to class” and second commit “..” .
* Pull changes from master after commit.
* Fix all conflicts.
* Push changes from local to remote branch.
* Create pull request after pushing changes and ensuring changes are on remote branch.
* Notify team members pull request was created and ask for pull request to be reviewed.
* Once pull request is reviewed and accepted, all team members should pull changes.
* If pull request Is rejected, review comments and make necessary changes then repeat the previous points above.