

**GENIX**

**Test Automation Architecture & Design Document**

Contents

[1. Terminology 7](#_Toc64622780)

[2. Introduction 8](#_Toc64622781)

[3. Scope 8](#_Toc64622782)

[3.1 In Scope 8](#_Toc64622783)

[3.2 Out of Scope 11](#_Toc64622784)

[4. Use Cases 12](#_Toc64622785)

[4.1 Use Case – 01 12](#_Toc64622786)

[4.2 Use Case – 02 13](#_Toc64622787)

[4.3 Use Case – 03 14](#_Toc64622788)

[4.4 Use Case – 04 15](#_Toc64622789)

[4.5 Use Case – 05 16](#_Toc64622790)

[4.6 Use Case – 06 17](#_Toc64622791)

[4.7 Use Case – 07 17](#_Toc64622792)

[4.8 Use Case – 08 18](#_Toc64622793)

[4.9 Use Case – 09 19](#_Toc64622794)

[4.10 Use Case – 10 20](#_Toc64622795)

[4.11 Use Case – 11 20](#_Toc64622796)

[4.12 Use Case – 12 21](#_Toc64622797)

[4.13 Use Case – 13 22](#_Toc64622798)

[4.14 Use Case – 14 23](#_Toc64622799)

[4.15 Use Case – 15 23](#_Toc64622800)

[4.16 Use Case – 16 24](#_Toc64622801)

[4.17 Use Case – 17 25](#_Toc64622802)

[4.18 Use Case – 18 26](#_Toc64622803)

[4.19 Use Case – 19 26](#_Toc64622804)

[4.20 Use Case – 20 27](#_Toc64622805)

[4.21 Use Case – 21 28](#_Toc64622806)

[4.22 Use Case – 22 28](#_Toc64622807)

[4.23 Use Case – 23 28](#_Toc64622808)

[4.24 Use Case – 24 29](#_Toc64622809)

[4.25 Use Case – 25 29](#_Toc64622810)

[4.26 Use Case – 26 30](#_Toc64622811)

[4.27 Use Case – 27 30](#_Toc64622812)

[4.28 Use Case – 28 30](#_Toc64622813)

[4.29 Use Case – 29 31](#_Toc64622814)

[4.30 Use Case – 30 31](#_Toc64622815)

[4.31 Use Case – 31 31](#_Toc64622816)

[4.32 Use Case – 32 32](#_Toc64622817)

[4.33 Use Case – 33 32](#_Toc64622818)

[4.34 Use Case – 34 32](#_Toc64622819)

[4.35 Use Case – 35 33](#_Toc64622820)

[4.36 Use Case – 36 33](#_Toc64622821)

[4.37 Use Case – 37 33](#_Toc64622822)

[4.38 Use Case – 38 34](#_Toc64622823)

[4.39 Use Case – 39 34](#_Toc64622824)

[4.40 Use Case – 40 34](#_Toc64622825)

[4.41 Use Case – 41 35](#_Toc64622826)

[4.42 Use Case – 42 35](#_Toc64622827)

[4.43 Use Case – 43 35](#_Toc64622828)

[4.44 Use Case – 44 35](#_Toc64622829)

[5. Data Design 36](#_Toc64622830)

[5.1 Data Structure 36](#_Toc64622831)

[5.2 Database 36](#_Toc64622832)

[5.3 Configuration 37](#_Toc64622833)

[6. Automation Architecture / Component Overview 37](#_Toc64622834)

[6.1 High Level Automation Architecture 37](#_Toc64622835)

[6.2 Component Overview 38](#_Toc64622836)

[6.2.1 Run Time View of Framework 38](#_Toc64622837)

[6.2.2 Configuration Module 39](#_Toc64622838)

[6.2.3 Read Module 44](#_Toc64622839)

[6.2.4 Prepare Module 45](#_Toc64622840)

[6.2.4.1 Object Locator Strategy 45](#_Toc64622841)

[6.2.4.2 Page Object Classes 46](#_Toc64622842)

[6.2.4.2.1 Common Reusable files 46](#_Toc64622843)

[6.2.4.2.2 Sign In page 46](#_Toc64622844)

[6.2.4.2.3 Home Page 46](#_Toc64622845)

[6.2.4.2.4 Application Page 47](#_Toc64622846)

[6.2.4.2.5 Digital Apps Center 47](#_Toc64622847)

[6.2.4.2.6 System Anomaly Detection 47](#_Toc64622848)

[6.2.4.2.6.1 System Configuration 47](#_Toc64622849)

[6.2.4.2.6.2 Anomaly Detection 48](#_Toc64622850)

[6.2.4.2.7 Opportunity Loss Manager 48](#_Toc64622851)

[6.2.4.2.7.1 Loss Configuration 48](#_Toc64622852)

[6.2.4.2.7.2 Loss Management 48](#_Toc64622853)

[6.2.4.2.8 Pipeline Integrity 48](#_Toc64622854)

[6.2.4.2.9 Machine Performance Analysis 48](#_Toc64622855)

[6.2.4.2.10 Major Accident Hazard Analysis 48](#_Toc64622856)

[6.2.4.2.11 ABC Analysis 48](#_Toc64622857)

[6.2.4.2.12 Condition Based Maintenance 48](#_Toc64622858)

[6.2.4.2.13 Asset Life Assessment 48](#_Toc64622859)

[6.2.4.2.14 Terminal Performance 48](#_Toc64622860)

[6.2.4.2.15 Contextual Fusion Hub 48](#_Toc64622861)

[6.2.4.2.15.1 Establish Connection 48](#_Toc64622862)

[6.2.4.2.15.2 Metadata Mapping 48](#_Toc64622863)

[6.2.4.2.15.3 Data Viewer 48](#_Toc64622864)

[6.2.4.2.16 System Twin Integrity Hub 48](#_Toc64622865)

[6.2.4.2.16.1 Asset Information 48](#_Toc64622866)

[6.2.4.2.16.2 Validation 48](#_Toc64622867)

[6.2.4.2.16.3 Master Data Consolidation 48](#_Toc64622868)

[6.2.4.2.16.4 System Hierarchy Transformation 48](#_Toc64622869)

[6.2.4.2.17 Knowledge Services Hub 49](#_Toc64622870)

[6.2.4.2.17.1 Business Service 49](#_Toc64622871)

[6.2.4.2.17.2 Cognitive Storage layer 49](#_Toc64622872)

[6.2.4.2.18 Model Fabric 49](#_Toc64622873)

[6.2.4.2.18.1 Data Exploration 49](#_Toc64622874)

[6.2.4.2.18.2 Model Registry 49](#_Toc64622875)

[6.2.4.2.18.3 Deployment 49](#_Toc64622876)

[6.2.4.2.19 Analytics App Studio 49](#_Toc64622877)

[6.2.4.2.19.1 KPI Designer 49](#_Toc64622878)

[6.2.4.2.19.2 Dashboard Designer 49](#_Toc64622879)

[6.2.4.2.19.3 Workflow Engine 49](#_Toc64622880)

[6.2.4.2.19.4 Collaboration Engine 49](#_Toc64622881)

[6.2.4.2.19.5 Manage User Data 49](#_Toc64622882)

[6.2.4.2.20 Industry Insights 49](#_Toc64622883)

[6.2.4.2.20.1 Energy 49](#_Toc64622884)

[6.2.4.2.20.2 Process 49](#_Toc64622885)

[6.2.4.2.20.3 Marine & Ports 49](#_Toc64622886)

[6.2.4.2.20.4 Measurement & Analytics 49](#_Toc64622887)

[6.2.4.2.21 Asset Twin Viewer 49](#_Toc64622888)

[6.2.4.2.21.1 Plant Hierarchy 49](#_Toc64622889)

[6.2.4.2.21.2 Maintenance 49](#_Toc64622890)

[6.2.4.2.21.3 Inspection 49](#_Toc64622891)

[6.2.4.2.21.4 Simulation 49](#_Toc64622892)

[6.2.4.2.22 Platform Administration 50](#_Toc64622893)

[6.2.4.2.22.1 Roles & User Management 50](#_Toc64622894)

[6.2.4.2.22.2 Application Provisioning 50](#_Toc64622895)

[6.2.5 Execute Module 50](#_Toc64622896)

[6.2.6 Report Module 51](#_Toc64622897)

[6.2.7 Class Diagram 52](#_Toc64622898)

[7. Deployment Overview 54](#_Toc64622899)

[7.1 Deployment Architecture 54](#_Toc64622900)

[7.2 CI/CD Integration 54](#_Toc64622901)

[7.2.1 Windows Batch file 55](#_Toc64622902)

[Open Jenkins using http://localhost:8080/ 55](#_Toc64622903)

[Click on New Item 55](#_Toc64622904)

[Enter Job Name and Choose Free Style Project 55](#_Toc64622905)

[Choose advanced options under the General section 56](#_Toc64622906)

[Enter the folder where your protractor conf file is present 56](#_Toc64622907)

[Now Add a Build Step and Choose Execute Windows Batch Command 56](#_Toc64622908)

[Provide the command to run the protractor, protractor conf.js. If user have not configured the workspace path, then user need to write the command to navigate to the workspace and then protractor run command like below 56](#_Toc64622909)

[Save the configuration and Click Build Now Link 56](#_Toc64622910)

[Now user will be able to see the execution, under console inside the instance number 56](#_Toc64622911)

[7.3 Command line Execution 58](#_Toc64622912)

[8. Non-Functional Testing 58](#_Toc64622913)

[9. Risks & Constraints 58](#_Toc64622914)

[10. Approval 59](#_Toc64622915)

[11. References 61](#_Toc64622916)

# Terminology

|  |  |
| --- | --- |
| SDET | Software Development Engineer in Test |
| BST | Build Stability Tests |
| AC | Acceptance Criteria |
| NFR | Non-Functional Requirements |
| TDM | Test data Management |
| BDD | Behavior Driven Development |
| UTC | Unit Test Cases |
| UAT | User Acceptance Test |
| ADA | Americans with Disabilities Act |
| MFE | Micro Frontends |
| DAST | Dynamic Application Security Testing |
| UX/CX | User Experience, Customer Experience |
| PWD | Person(s) With disabilities (or disability) |
| WCAG | Web Content Accessibility Guidelines |
| RCA | Root Cause Analysis |
| CI/CD | Continuous integration/Continuous Deployment |
| APM | Application Performance Monitoring |
| DXL | Digital Experience Layer |
| SAD | System Anomaly Detection |
| OLM | Opportunity Loss Manager |
| CFH | Contextual Fusion Hub |
| STIH | System Twin Integrity Hub |
| KSH | Knowledge Services Hub |
| KPI | Knowledge Process Index |
| PA | Platform Administration |

# Introduction

The objective of this document is to provide the details of different components of the Test Automation framework.

To enable end user to understand and use the framework, this document details the way the framework supports the development, maintenance and execution of the automation scripts. In addition to this, the document provides necessary details related to configuration of the framework as well as the procedure to create and use external data sources for enabling creation of automation of automated test scripts using the data driven approaches. The document describes how additional dimensions of testing (performance, accessibility) can be layered on top of this basic framework

# Scope

# In Scope

|  |  |  |
| --- | --- | --- |
| **Version** | **Module** | **Functionality** |
| Genix Build  0.1.x | Contextual Fusion Hub | * Connectivity and mapping configuration to data source systems * Data Integration with source systems - Edge Systems, IT Systems, Geospatial Systems * Data file compression and transfer to cloud for bulk data * Transformation components to process and contextualize data * System Twin Integrity Hub * Rules Configuration * Data source mapping & transformation * Data Consistency Check   Dashboards 7 Reports |
| Cognitive  Analytics Hub | * Data model – Core, Reference and Master data entities, Value Pillars & Drivers   Data lake with raw store, curated store |
| Cognitive  Analytics Apps | * Digital Value Apps Centre * Chart Library * Opportunity Loss manager   System Anomaly Detection |
| Knowledge Services Hub | * Business Services   Custom Services |
| Platform  Features | * Authentication, Authorization * Role based access control   Audit and Logging |
| **Responsibilities**   * Create and Execute Test Strategy Document, Test Plans –Integration, System. The tests will include but not limited to, Functional, Technical, Security, UI/UX, Compatibility test scenarios and cases. These need to be prepared based on the defined requirements and should be approved prior to execution * Test Cases will need to include but not limited to message reliability, data integrity, data routing, interfacing with enterprise (e.g. SAP, IBM Maximo, etc.) and edge systems (Historian, Edge Devices/ Systems, etc.) and ingestion of data in real time and batch (e.g.ETL /pipeline) * Create Test Automation frameworks and Run Automated Test Scripts * Functional tests of Data pipeline Automation * Performance tests of Data pipeline Automation * Create and execute tests using open source Test Automation frameworks for UI, Web Services and performance tests (Example: Jmeter, Selenium, TestNG, REST Assured) * Maintain Test Data used during testing * Maintain Defect Logs * Create Text Execution and completion report | | |

|  |  |  |
| --- | --- | --- |
| **Version** | **Module** | **Functionality** |
| Genix Build  0.2.x | Contextual Fusion Hub | * Connectivity and data ingestion from additional source systems / modules   Components for non-Azure environment |
| Cognitive Analytics Hub | * • Incremental development of data models for Value Pillars and Value Drivers |
| Cognitive Analytics Apps | * Dashboard and KPI Designer * Additional charts within Chart Library * Predictive Maintenance Heat Transfer Equipment |
| Business Value Applications | * Opportunity loss manager * System anomaly detection * Application App Studio * BVA Dashboards |
| Industry Insights applications | * Industry specific apps * Dashboard manager * Industry specific Dashboards |
| Model Fabric | * Data Exploration & Treatment * Sandbox * Model Training |
| Knowledge Services Hub | * User Experience and Interface * Services Registry |
| Platform Features | * Alerts and Notifications * LCM |
| **Responsibilities**   * All activities to be performed for Azure and non-Azure build of GENIX * Regression Test of GENIX Build 0.1.x build * Create and Execute Test Strategy Document, Test Plans for incremental functionality –Integration, System. The tests will include but not limited to, Functional, Technical, Security, UI/UX, Compatibility test scenarios and cases. These need to be prepared based on the defined requirements and should be approved prior to execution * Test Cases will need to include but not limited to message reliability, data integrity, data routing, interfacing with enterprise (e.g. SAP, IBM Maximo, etc.) and edge systems (Historian, Edge Devices/ Systems, etc.) and ingestion of data in real time and batch (e.g. ETL /pipeline) * Create/ Update Test Automation frameworks and Create, Run Automated Test Scripts * Functional tests of Data pipeline Automation * Performance tests of Data pipeline Automation * Create and execute tests using open source Test Automation frameworks for UI, Web Services and performance tests (Example: Jmeter, Selenium, TestNG, REST Assured) * Integrate & Execute test automation through CI/CD pipeline * Maintain Test Data used during testing * Maintain Defect Logs * Create Text Execution and completion report | | |

|  |  |  |
| --- | --- | --- |
| **Version** | **Module** | **Functionality** |
| Genix Release 1.0 | Entire Platform | • Hardening of existing functionality |
| * **Responsibilities** * Regression Test of all test’s cases until AIAAS Build 0.2x * All activities to be performed for Azure build of AIAAS * Execute Test Strategy Document, Test Plans for incremental functionality –Integration, System. The tests will include but not limited to, Functional, Technical, Security, UI/UX, Compatibility test scenarios and cases. These need to be prepared based on the defined requirements and should be approved prior to execution * Test Cases will need to include but not limited to message reliability, data integrity, data routing, interfacing with enterprise (e.g. SAP, IBM Maximo, etc.) and edge systems (Historian, Edge Devices/ Systems, etc.) and ingestion of data in real time and batch (e.g. ETL /pipeline) * Create/ Update Test UI Automation frameworks and Create, Run Automated Test Scripts * Functional tests of Data pipeline Automation * Performance tests of Data pipeline Automation * Create and execute tests using open source Test Automation frameworks for GENIX business value apps and Industry Apps (Example: Jmeter, Selenium, TestNG, REST Assured) * Integrate & Execute test automation through CI/CD pipeline * Maintain Test Data used during testing * Maintain Defect Logs * Create Text Execution and completion report | | |

The scope includes—

* Understanding of the Requirements
* Detail design of Test plans, Develop Test Strategy, Test cases, Defining Entry/Exit criteria, Requirement Traceability Matrix and manage them through Azure DevOps
* Integration Testing, Data pipeline testing, Checking intermittent output, System Testing, Performance Testing, Security Testing, Functionality Testing, Black Box Testing, API testing
* Automation of Testing activities

a. UI automation of Business Value Applications and Industry Application

* Incorporating testing within CI/CD pipelines
* Test coverage is for Azure Cloud environment and deployments
* 65% Execution of automated regression test suite for On-Premise deployments with same functional tests as that of Azure Cloud
* 65% API, ETL & Performance automation updates and maintenance. Script creation for new features. Weekly scheduled runs for the build regression.
* Test deliverables including but not limited to Test Results documentation, Defect Report, Installation/ Test procedures guidelines, Release notes
* Test plans reviews, Testing coverage reviews shall be reviewed by senior resource from Supplier who was already interviewed by ABBGISPL.
* Supplier shall provide the details of resources involved in the project. Any resource shuffling shall be strictly avoided. Any such resources shuffling all be informed to and approved by ABBGISPL.
* Completion of each deliverable shall be considered only after ABBGISPL acceptance is done.

# Out of Scope

1. Testing for the following services is not considered based on the inputs from ABB technical team—
   1. Device Management
   2. Device Registration & provisioning
   3. Device authentication
2. Testing of data backup & archival storage
3. Development of device simulator
4. Unit testing is not part of the scope. Referring to sec. 4.4, White Box testing activities are not considered part of the Scope
5. Testing of Non-Azure cloud build of AIAAS (for example AWS)

# Use Cases

# Use Case – 01

|  |  |
| --- | --- |
| Requirement ID | 5396 |
| Requirement | Configurable Test environment |
| User Story | As Ari, I should be able to configure UI automation to point to my development environment so that I can run the tests against it to catch potential regressions. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation for Configurable Test Environment, where configuration json/excel file checks on which environment execution should happen, required environment details should read and configure the same.   * If Environment name provided E.g.: UAT, Pilot, SIT, corresponding property values (like   + Browser   + Operating System   + Language, Source URL   + Test Suite Location details   + Test Report location details   + Test log location details   + DB connection details   ) from configuration json/excel file and provide the same to   * + The Read module when Test Suite and Test Case are getting selected   + Report & Logging Module when report and log generation is getting created   + Prepare module to create DB connection. * If No values provided as environment, UAT taken as default and the values for the same into consideration. * If the respective values provided for the properties as part command line or Jenkins task execution, the current provided values take precedence over the default values |
| Comments | * It should be possible to configure the automation system to test any environment. * Examples, UAT, Nightly/Weekly, Pilot * DB configuration. * Configuration name should be all small case and hyphen should be used to separate words. Ex. client-secret. * No more than one configuration file should be used by the framework. * The configuration should be in JSON format with well-defined schema. * Related configuration should be collocated. |
| Component Mapping | Configuration Module – Config\_Properties.json provides the list of environments available to select from.  Execution parameter at run time decides, on which Environment Test Suite to be executed. Based on that it will get the required properties from properties json file and prepared environment. |

# Use Case – 02

|  |  |
| --- | --- |
| Requirement ID | 5958 |
| Requirement | Execute from command line |
| User Story | As Ari (or Shaili), I should be able run tests/test suites from command line. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to execute the automation scripts from command line with required custom defined parameters   * The expected parameters from command line are * help * environment (Default – UAT) * browser (Default – Chrome) * operating-system (Default – Windows) * language (Default – English) * source-URL (Default – Current URL) * testsuite-file-location (Default – Test data folder in project) * test-report-file-location (Default – reports folder in project) * test-log-location (Default – logs folder in project) * bd-connection-details (Default – UAT environment db connection details) * assertion-required (Default – yes) * component (Default – yes) * we will add few more based on requirement. * If case of invalid parameter name provided in command line - help command should be executed and return help about command line parameters * All these options forwarded to configuration module, based on that all selections like URL, OS, browser, language, test suite location, test report location, log location, assertion configuration will be executed as defined. |
| Comments | ex:   * ant -f test.xml clean build all * ant testsuite.xml clean build all |
| Component Mapping | Command line Execution - User can able to execute test suite from command line using different parameters like config file, properties file and other required parameters for execution |

# Use Case – 03

|  |  |
| --- | --- |
| Requirement ID | 4407 |
| Requirement | Override configuration on command line |
| User Story | As Ari (or Shaili), I should be able to override all configuration from the command line. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to override the default parameter value, with the value specified in command line for parameter.   * The expected parameters from command line are * help * environment (Default – UAT) * browser (Default – Chrome) * operating-system (Default – Windows) * language (Default – English) * source-URL (Default – Current URL) * test-suite-file-location (Default – Test data folder in project) * test-report-file-location (Default – reports folder in project) * test-log-location (Default – logs folder in project) * bd-connection-details (Default – UAT environment db connection details) * assertion-required (Default – yes) * component (Default – yes) * we will add few more based on requirement. * If case of invalid parameter name provided in command line - help command should be executed and return help about command line parameters * All these options forwarded to configuration module, based on that all selections like URL, OS, browser, language, test suite location, test report location, log location, assertion configuration will be executed as defined. * When there is value parameter provided for any of the allowed, that value take precedence over the default value * When parameter not provided for the configuration, it will take the default value as configuration value from configuration json or excel file |
| Comments | * Configuration specified on the command line should override the configuration specified in the configuration file. * Those configurations not specified on the command line should use the ones specified in the configuration file. |
| Component Mapping | Command line Execution – If no parameters defined in command line execution, test runner should take default config and properties files. If parameter and value pair specified in command line execution, the specified parameter and value pair should take precedence over the default set of values. |

# Use Case – 04

|  |  |
| --- | --- |
| Requirement ID | 6877 |
| Requirement | Author test case without coding |
| User Story | As Shaili, I should be able to define UI steps in an Excel sheet so that I can refer them when I assemble a test scenario. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation or test steps, which can be referred to build test case or test scenario without changing the automation script.   * Test cases should be defined / configured through test case XL sheet   + Test Case ID   + Suite   + Test Step ID   + Expected value   + Parameters (Param1, param2...) * Each test case should be configured as combination two or more test steps * Each test steps configured as combination of   + Test Step ID   + Test Step Name   + Page Object   + Operation   + Dependency   + Parameters (Param1, param2...) * Look up sheet which is repository of all page objects * All this test case construction happens in prepare module based on the same, it will forward to execute module * If there is any failure or misconstruction happening while defining test case, test case result should return as exception saying that problem in test case construction. (if the selected lookup values in not available in implementation). |
| Comments | * Excel spreadsheet may be used to specify the tests. * Fragment of UI interaction should be modelled as a 'step'. Test cases may have multiple 'steps'. * UI Step should be separated out from test case scenario. Ex., test case definition is TestCase1: result = parameter1 + parameter2). Test case instance is TestCase1(4, 5) or TestCase1(6.3, 9.1). * Steps may have an operation, take parameters to execute, have a specific return value. |
| Component Mapping | Testcase XL sheet and Test step - User should be able to write/ define new test cases in xl sheet by providing required test case and test step specification without altering automation code. |

# Use Case – 05

|  |  |
| --- | --- |
| Requirement ID | 9121 |
| Requirement | Author test case without coding |
| User Story | As Shaili (or Ari), I should be able to automate a scenario using steps so that I can easily assemble the steps that make up a UI flow. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide ease of usage of framework so that end user can design test case/ scenario by using defined steps without modifying test automation script.   * Test cases should be defined / configured through test case XL sheet   + Test Case ID   + Suite   + Test Step ID   + Expected value   + Parameters (Param1, param2...) * Each test case should be configured as combination two or more test steps * Each test steps configured as combination of   + Test Step ID   + Test Step Name   + Page Object   + Operation   + Dependency   + Parameters (Param1, param2...) * Look up sheet which is repository of all page objects * All this test case construction happens in prepare module based on the same, it will forward to execute module * If there is any failure or misconstruction happening while defining test case, test case result should return as exception saying that problem in test case construction. (if the selected lookup values in not available in implementation). |
| Comments | * Excel spreadsheet may be used to specify a Scenario. * A scenario models a test case in QE’s Test Management System. * A scenario may refer one or several steps in the sequence specified in the corresponding test case. * The steps may specify parameters associated with them and as defined by the corresponding test case. |
| Component Mapping | Testcase XL sheet and Test step - User should be able to write/ define new test cases in xl sheet by providing required test case and test step specification without altering automation code. |

# Use Case – 06

|  |  |
| --- | --- |
| Requirement ID | 4473 |
| Requirement | Specify test suite against a test scenario |
| User Story | As Ari (or Shaili), I should be able to specify a suite name against a scenario so that I can determine the frequency of its run. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to write a new scenario against a specific suite name. In case of invalid suite name execution engine should alert with exception.   * Test cases should be defined / configured through test case XL sheet   + Test Case ID   + Suite   + Test Step ID   + Expected value   + Parameters (Param1, param2...) * If there is any failure or misconstruction happening while defining test case, test case result should return as exception saying that problem in test case construction. (if the selected lookup values in not available in implementation). * If the ‘Suite Name’ is not present in Execution precedence sheet it should alert with invalid suite name failing the test case. |
| Comments | * Scenario should be directly associated with a suite where it should run in accordance with the corresponding test case in Test Management System. * Invalid suite names (ones that are not configured) should be flagged as a failure against the scenario. |
| Component Mapping | TBD |

# Use Case – 07

|  |  |
| --- | --- |
| Requirement ID | 1407 |
| Requirement | Execute scenarios specified in multiple files |
| User Story | As Ari (or Shaili), I should be able to execute test scenarios specified in multiple Excel files so that I can use a “master” file for (predefined) steps and another to test a scenario |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to create a master of file of test steps. Which eventually used to create test scenario based on requirement and execute the same (Bug Fixes, new scenario creation) |
| Comments | * It can be envisioned that QE team (primarily) and Engineering teams will co-create a “master” file for test steps. This file may be used to create as many scenarios as needed. * If multiple files are specified, the test scenarios should be merged. |
| Component Mapping | TBD |

# Use Case – 08

|  |  |
| --- | --- |
| Requirement ID | 3218 |
| Requirement | Specify Locator Strategy |
| User Story | As Shaili, I should be able to specify multiple strategy to locate an element so that I can achieve higher reliability of test runs. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide define and implement object locator strategy, where each object can be identified with one of the specified options.  The object locator strategy should be defined in following order of precedence   * As priority trying to find unique object by angular model properties - binding, model, options, repeater required to locate the angular attributes, ng-bind, ng-model, ng-options, ng-repeat. * To identify with Angular class properties which lead unique identification object * To identify element with Tag properties which can identify unique property - text/partial text locators such as buttonText and partialButtonText * To identify with CSS selectors, if defined any - elements using CSS selectors containing text using cssContainingText locator. * As least priority, to identify element with configured Xpath for that object * Execute module will take care of identifying object with one of the available locator strategy options to identify object and apply action on the same. * If object is not identified with any of the locator option, there is retry mechanism once. * In case of retry mechanism also fails identify object, execute process throws an exception, which causes testcase fail, with provided valid reason in report and log |
| Comments | Framework should capable of identifying the locators dynamically.   * Find an element using a css selector. by.css('.myclass') * Find an element with the given id. by.id('myid') * Find an element using an input name selector. [by.name](http://by.name/) ('field\_name') * Find an element with a certain ng-model. by.model('name') * Find an element bound to the given variable. by.binding('bindingname') |
| Component Mapping | Prepare Module – There should strategy defined to locate object on the page. It checks for Angular binding, Angular Class, CSS properties, Tag properties and Xpath in sequence.  Angular binding as the priority and Xpath as last priority to consider locating object.  Locator strategy defined in the respective POM files of the object. |

# Use Case – 09

|  |  |
| --- | --- |
| Requirement ID | 4008 |
| Requirement | Skip common steps when necessary |
| User Story | As Shaili (or Ari), I should be able to skip common & repetitive prerequisite steps so that I can avoid executing them to reduce test time |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide configuration / implementation where the prerequisite steps execute once /multiple times based on the defined requirement.   * By default – the dependent test case to be executed once per session * If forced execution set to ‘yes’ – the dependent test case to be called as many times it got set to call * If ‘To be executed value’ - ‘No’ and it is part of dependent test case, it should be executed * To be executed ‘Always’ – it should be executed always * To be executed ‘Never’ – it shouldn’t be executed at any point of call * Configure module will read the test suite configuration and dependency among the test cases and execution precedence and forward that to prepare module * Prepare module will create the execution flow based on the configuration come and sent it to execute module |
| Comments | Example of a prerequisite step is logging in. This can be avoided for subsequent tests. |
| Component Mapping | Configuration Module, Prepare Module – Configuration module for test case defines the pre-requisite/Dependency test cases and its execution priority.  If the test case is pre-requisite/dependency test case for any other test case, it will be executed only once in that execution run session. If the configurable parameter ‘Forced Execution’ Set to ‘Yes’ for that it will be executed n no. of times it gets called.  Log should capture the skipped test cases information for analysis. |

# Use Case – 10

|  |  |
| --- | --- |
| Requirement ID | 1711 |
| Requirement | Execute component level regression |
| User Story | As Ari (or Shaili), I should be able to run tests pertinent to the component I’m working on so that I can quickly verify if I’m not causing a potential regression before committing my code. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation where user can configure and execute component/module level test suite/test cases.   * Test cases are maintained separate excel sheets in test suite configuration excel * When –component “SAD” is provided in command line or Jenkins job, process will take test case from respective component sheet. * It will execute all the test cases in component excel sheet except for the ‘Never’ executed test cases * If component name given wrong, process will generate exception with valid reason in report and log file. * If there is no value given, only --component given with empty value, it will take sanity as default |
| Comments | * A list of components that can tested independently of each other should be displayed on the command line without further markup. * The user should be able to invoke the test run from the command line for the specified component alone. |
| Component Mapping | Configuration Module, Prepare Module - Based on the parameters provided at execution time, Configuration module selects the test cases to be executed based on the component against which the test case gets mapped. In this way component level test case execution achieved. |

# Use Case – 11

|  |  |
| --- | --- |
| Requirement ID | 4812 |
| Requirement | Select test cases to run |
| User Story | As Shaili, I should be able to skip automated tests from executing so that I may decide to skip (known) unstable tests |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide configuration & implementation for not to execute selected test cases from current set of execution   * Test cases are maintained separate excel sheets in test suite configuration excel * By default – the dependent test case to be executed once per session * If forced execution set to ‘yes’ – the dependent test case to be called as many times it got set to call * If ‘To be executed value’ - ‘No’ and it is part of dependent test case, it should be executed * *To be executed ‘Always’ – it should be executed always* * *To be executed ‘Never’ – it shouldn’t be executed at any point of call* * Configure module will read the test suite configuration and dependency among the test cases and execution precedence and forward that to prepare module * Prepare module will create the execution flow based on the configuration come and sent it to execute module |
| Comments | * Test scenarios may be disabled from executing from within the Excel file using ‘disabled’ reserved suite name. |
| Component Mapping | Configuration Module, Prepare Module – when the test cases to be excluded from execution at any case, ‘Execution Type’ should be set as ‘Disabled’. |

# Use Case – 12

|  |  |
| --- | --- |
| Requirement ID | 9632 |
| Requirement | Test reporting |
| User Story | As Ari (or Shaili), I should be able to view a report so that I can easily determine the result of UI test run and possibly reasons for any failures. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should be able to provide implementation, where granular level of reporting details provided to get the failure reasons and to do execution analysis. Reporting engine should be in plug and play for new kind of reports with minimal code enhancement.   * Report option should be pluggable option * Implementation should be available, to configure and select one of the reporting options available (jasmine, Extent, TestNG…) * Based on the reporting option provided in the command line or Jenkins job, that specific report should be generated * If the no reporting option provided, default option should be generated as Jasmine report * If unavailable report option provided, executor should quit with proper error message to user / should generate jasmine report * ReportingManager package should be abstract class, where only method name definitions available. * For each option it should be and extendable class for ReportingManager, where exact implementation to that option available. * When new option included, just one more extend class with that report option functionality available without changing the base class and other option classes. |
| Comments | * Report generation using Extent framework * Integration of Extent framework should give results of each test case status(pass/fail), screenshot of failed testcase, detailed error log * It should be possible to switch the reporting engine at will. |
| Component Mapping | Report Module – Will provide the results of executed test suite – Test case – Test Step details.  Provides the screen shot of failed scenario and error log pertaining to that specific failure |

# Use Case – 13

|  |  |
| --- | --- |
| Requirement ID | 9078 |
| Requirement | Capture screen shot and video recording in case of failure scenarios |
| User Story | As Ari (or Shaili), I should be able to view a screenshot/video of the browser at the time of failure to quickly determine the possible reasons for failure. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide screen shot to analyze the reason for failure.  As an Automation Engineer, I should provide video recording in case of failed test cases to determine the failure reason.   * This is part of Reporting Manager implementation * There is option configuration, where the screen shots required * By default, Reporting manager captures screen shot for each test step * Based on the screen shot capture configuration, it will enable or disable for screenshot capture * If no option provides from command line/Jenkins job, it will all screen shot as by default   Video recording – TBC |
| Comments | The videos and screenshots should be in line with the issue/failure. |
| Component Mapping | Report Module – Will provide the results of executed test suite – Test case – Test Step details.  Provides the screen shot of failed scenario and error log pertaining to that specific failure |

# Use Case – 14

|  |  |
| --- | --- |
| Requirement ID | 9135 |
| Requirement | Support multiple browsers & multiple OS |
| User Story | As Ari (or Shaili), I should be able to run automation tests against multiple browsers so that I can determine the browser compatibility and take corrective action if required.  As Ari (or Shaili), I should be able run automation from any operations system so that I can determine the browser compatibility and take corrective action if required. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to verify automation execution different supported Browsers  As an Automation Engineer, I should provide implementation to verify automation pack execution on different operating system.   * There will be option in configuration for browser and OS selection * If No browser option provided, Chrome will be taken as default browser option * If No OS option provided, Windows will be taken as default browser option * From command line, Jenkins job, user should be provided with option –browser, --operating-system to provide values. * Comma separated values allowed when more than one browser option, operating system to be provided. |
| Comments | **Browser support:**   * Chrome * Edge * Firefox * Safari   **OS support:**   * Windows * Linux * Mac |
| Component Mapping | Configuration Module, Prepare Module, Execute Module – in Configuration.json or Test configuration file, there is provision to select one or more browsers and one or more OS on which test run to perform |

# Use Case – 15

|  |  |
| --- | --- |
| Requirement ID | 5793 |
| Requirement | Verify against automated API response |
| User Story | As Ari (or Shaili), I should be able to verify the results on the UI with the corresponding backend API so that the data validation is precise |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to provide validation of data base through KSH API Call, wherever applicable.  As an Automation Engineer, I should provide implementation to provide validation through customized API call for database validation. |
| Comments | After obtaining the results from the UI, it is required to verify this with the corresponding API from API automation suite.   * One level of validation, will be comparing the UI value with the API Response value for the same operation * Most of the times, KSH module application services will be used to obtain the values from backend * If KSH API is not available, try to call the trace flow from the engineering side and call the same Rest API * In case of unavailability of Point no.2 & 3, need to build and publish rest API to get the validation done * Database access cannot be done directly due to security reasons, so all required queries should be encapsulated in Rest API and get published. * In case of assertion failure on comparison of UI value with API response, the failure will be forwarded to the report module and test case failure resulted based on the scenario. |
| Component Mapping | Execute Module – For most of the event action performed from UI, there is corresponding KSH call defined.  Calling through KSH module will be applicable once and validate against the UI displayed value  In case of no KSH call available, with the help of SQL manager, corresponding results queried from database and validate against UI displayed value. |

# Use Case – 16

|  |  |
| --- | --- |
| Requirement ID | 5830 |
| Requirement | Verify against fixed value (Expected Result) |
| User Story | As Ari (or Shaili), I should be able to verity the results on the UI with a fixed value so that the data validation is precise |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to provide validation/assertion against the expected value provided through excel sheet.  After obtaining the results from the excel sheet value, it is required to verify this with the corresponding API from API automation suite.   * One level of validation, will be comparing the expected value provided through excel cell with the API Response value for the same operation * Most of the times, KSH module application services will be used to obtain the values from backend * If KSH API is not available, try to call the trace flow from the engineering side and call the same Rest API * In case of unavailability of Point no.2 & 3, need to build and publish rest API to get the validation done * Database access cannot be done directly due to security reasons, so all required queries should be encapsulated in Rest API and get published. * In case of assertion failure on comparison of value from the excel sheet with API response, the failure will be forwarded to the report module and test case failure resulted based on the scenario. |
| Comments | After obtaining the results from the UI, it is required to verify this with the corresponding value specified against the step in the Excel file. |
| Component Mapping | Execute Module – For the fixed value assert, the value displayed on UI or the value retrieved from API call against the value present in the excel sheet for the test step. |

# Use Case – 17

|  |  |
| --- | --- |
| Requirement ID | 2590 |
| Requirement | Execute verification in parallel |
| User Story | As Ari (or Shaili), I should be able to execute tests quickly so that I can proceed with further steps in the process. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to provide the validation based on configuration parameter (Always, Never, Sometimes).  As an Automation Engineer, I should provide implementation to do the validation in a separate thread which is independent of execution process running in parallel.   * The assert process/validation process implemented in separate thread. * All validation push to different queue * Once validationAction push to queue, execution process proceeds to next set of action in sequence * Assertion Queue actions will execute parallel to the regular execute process * In case of any assertion fails, failure will be resulted in report module |
| Comments | * Verification of the value on the UI with the corresponding value in Excel sheet/Backend (using API call) should be done asynchronously. * The verification should be done in a separate thread as the test thread. |
| Component Mapping |  |

# Use Case – 18

|  |  |
| --- | --- |
| Requirement ID | 1184 |
| Requirement | Verify against API response |
| User Story | As Ari (or Shaili), I should be able to verify the results on the UI with the corresponding backend API so that the data validation is precise |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to provide validation of data base through KSH API Call, wherever applicable.  As an Automation Engineer, I should provide implementation to provide validation through customized API call for database validation   * One level of validation, will be comparing the UI value with the API Response value for the same operation * Most of the times, KSH module application services will be used to obtain the values from backend * If KSH API is not available, try to call the trace flow from the engineering side and call the same Rest API * In case of unavailability of Point no.2 & 3, need to build and publish rest API to get the validation done * Database access cannot be done directly due to security reasons, so all required queries should be encapsulated in Rest API and get published. * In case of assertion failure on comparison of UI value with API response, the failure will be forwarded to the report module and test case failure resulted based on the scenario. |
| Comments | * After obtaining the results from the UI, it is required to verify this with the corresponding product API. * Note that this should be an exception as all the APIs should be automated in the API Automation Suite. The first choice is to invoke the corresponding API from API Automation Suite. |
| Component Mapping | Execute Module – For most of the event action performed from UI, there is corresponding KSH call defined.  Calling through KSH module will be applicable once and validate against the UI displayed value  Existing API automation need to check how we can have one row of data, as API automation suite required pre-requisite configuration |

# Use Case – 19

|  |  |
| --- | --- |
| Requirement ID | 9788 |
| Requirement | Verify against DB |
| User Story | As Shaili, I should be able to verify the results on the UI with the corresponding DB data so that the data validation is precise |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to provide validation through customized API call for database validation.   * One level of validation, will be comparing the UI value with the Database value retrieved through API call * Most of the times, KSH module application services will be used to obtain the values from backend * If KSH API is not available, try to call the trace flow from the engineering side and call the same Rest API * In case of unavailability of Point no.2 & 3, need to build and publish rest API to get the validation done * Database access cannot be done directly due to security reasons, so all required queries should be encapsulated in Rest API and get published. * In case of assertion failure on comparison of UI value with API response, the failure will be forwarded to the report module and test case failure resulted based on the scenario. |
| Comments | * Always use KSH API to make the call to the DB. * All non-KSH calls to DB should be through a REST API. |
| Component Mapping | Execute Module – For most of the event action performed from UI, there is corresponding KSH call defined.  Calling through KSH module will be applicable once and validate against the UI displayed value  In case of no KSH call available, with the help of SQL manager, corresponding results queried from database and validate against UI displayed value. |

# Use Case – 20

|  |  |
| --- | --- |
| Requirement ID | 5567 |
| Requirement | Identify flaky test |
| User Story | As Ari, I should be able correlate product log and test log to identify an unstable test, so that I can speed up RCA. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to provide the flaky test information by using the execution trend and provide engineering/product logs along with the required artefacts which provide for failed test case.   * There will be implementation mechanism to provide identify most failure by comparing the previous execution and test logs * For every failure we need capture product log files (Engineering team log files provided in development server) * Require access and authentication to connect to product server logs * In addition to test automation logs, screen shots and other required artefacts, should provide logs accessed from product server * It will be better used for developer, to get the exact failure reasons. |
| Comments | * The product log at the time of failure, the test log, screenshot/video should be juxtaposed in the report so that engineers can rapidly complete RCA |
| Component Mapping | Report Module: Reporting module provides the execution trend to identify most failure test across execution and provide engineering / product logs along with the required artefacts which provide for failed test case. |

# Use Case – 21

|  |  |
| --- | --- |
| Requirement ID | 9608 |
| Requirement | Report execution metrics |
| User Story | As Shaili (or Ari), I should be able to obtain metrics on execution time of the overall test run, individual test so that I can determine if the application is working as expected or sluggish. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to provide execution trend in terms of response time, average of previous response times and validate with the current response times and detailed about which test cases has variance in response times   * There should be implementation to provide response time for each action, test step, test case and test suite * Report should detail about current response time and previous run average response times at each level (action, test step, test case and test suite) * The above metrics will be used easily identify execution trend and performance improvement or not for the current execution with over all response execution trend. |
| Comments |  |
| Component Mapping | Report Module – Reporting module provides the execution time of each test case and overall test’s suite execution for all test execution performed until the expected point, so that it will be used for application performance analysis. |

# Use Case – 22

|  |  |
| --- | --- |
| Requirement ID | 1633 |
| Requirement | Name test suite |
| User Story | As Shaili, I should be able to create a test suite so that I can test varied number of scenarios for varied testing depth needs. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide configuration and implementation to name test suite against defined number of scenarios.  Reserved test suite name always executable, Disabled test suite name never executable.   * Test cases are maintained separate excel sheets in test suite configuration excel * By default – the dependent test case to be executed once per session * If forced execution set to ‘yes’ – the dependent test case to be called as many times it got set to call * If ‘To be executed value’ - ‘No’ and it is part of dependent test case, it should be executed * *To be executed ‘Always’ – it should be executed always* * *To be executed ‘Never’ – it shouldn’t be executed at any point of call* * Configure module will read the test suite configuration and dependency among the test cases and execution precedence and forward that to prepare module * Prepare module will create the execution flow based on the configuration come and sent it to execute module |
| Comments | * Names of test suites should be application configuration. * The test suite names may be specified against a scenario to determine its run frequency. * A “reserved” suite name called always and disabled should be implicitly available. |
| Component Mapping | Configuration Module – Configuration for test suite names configured against test script/test case name to determine on which instance the test script to be executed and how many times test case is executed. |

# Use Case – 23

|  |  |
| --- | --- |
| Requirement ID | 2667 |
| Requirement | Specify test suite order of precedence |
| User Story | As Shaili, I should be able to specify a precedence order of named test suites so that I can run tests in a suite and all the tests in the “lower” suites. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to select test cases based on suite execution at present. Execution precedence should be defined, and test case selection should be based on that defined precedence.   * There will be implementation to select the test cases based on their suite configuration * There will execution precedence sheet which defines order of test cases to select * If order is like NightlyRun -> ShortRun -> LongRun -> RegressionTest -> SystemTest   + If NightlyRun selected, only test cases named against Nightly run test cases to be selected   + If ShortRun Selected, testcases named against ShortRun and level below i.e., test cases named against NightlyRun testcases also should get selected   + If Long Run Selected, testcases named against Longrun should be selected along with below levels NightlyRun and Short Run. The same logic should be applicable for all defined test execution levels. * If the testcase named against ‘To Be Executed’ set as ‘Always’, the test case should be selected in all test suite execution, irrespective current test case execution suite selection * If the testcase named against ‘To Be Executed’ set as ‘Never’, it should not be selected in any of the test execution. * ‘Always’ and Never’ reserved keywords should have more precedence over all other configuration/selection. |
| Comments | * It’s a general practice to run test scenarios in “shorter” test suites when “larger” test suites are executed. Ex. When “Large Regression Suite” is run, the test scenarios in “Short Regression Suite”, “Sanity”, “Minimum Acceptance Test Suite” are also run. That is, LRG ⊃ SRG ⊃ Sanity ⊃ MAT. |
| Component Mapping | Configuration Module – Configuration for test suite names configured against test script/test case name to determine in which named test suite to execute and precedence defines which order group test suite must choose test cases. |

# Use Case – 24

|  |  |
| --- | --- |
| Requirement ID | 7204 |
| Requirement | Run a named test suite |
| User Story | As Shaili, I should be able to run sanity, short/large regression scenarios so that I can execute the most pertinent suites given the need. As Ari, I should be able to run component-level, short regression tests so that I can verify if the feature/fix I'm about to commit doesn't cause regression |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to select test cases based on suite execution at present. Execution precedence should be defined, and test case selection should be based on that defined precedence.  Reserved test suite name always executable, Disabled test suite name never executable.   * There will be implementation to select the test cases based on their suite configuration * There will execution precedence sheet which defines order of test cases to select * If order is like NightlyRun -> ShortRun -> LongRun -> RegressionTest -> SystemTest   + If NightlyRun selected, only test cases named against Nightly run test cases to be selected   + If ShortRun Selected, testcases named against ShortRun and level below i.e., test cases named against NightlyRun testcases also should get selected   + If Long Run Selected, testcases named against Longrun should be selected along with below levels NightlyRun and Short Run. The same logic should be applicable for all defined test execution levels. * If the testcase named against ‘To Be Executed’ set as ‘Always’, the test case should be selected in all test suite execution, irrespective current test case execution suite selection * If the testcase named against ‘To Be Executed’ set as ‘Never’, it should not be selected in any of the test execution. * ‘Always’ and Never’ reserved keywords should have more precedence over all other configuration/selection. * There we will be able to create new suite name in ‘Execution Precedence’ and map the same against the Testcase * When we mention new newly defined suite name to execute, it should select the defined testcases to execute. |
| Comments | * The test suite name may be specified from the command line. * Invalid test suite name should fail the entire run.   All test scenarios marked always must always execute. |
| Component Mapping | Configuration Module – Configuration for test suite names configured against test script/test case name to determine in which named test suite to execute and precedence defines which order group test suite must choose test cases. |

# Use Case – 25

|  |  |
| --- | --- |
| Requirement ID | 8716 |
| Requirement | Specify Excel sheet from command line |
| User Story | As Ari, I should be able to create a custom Excel sheet for a test case subset/new tests so that I can focus on the new test cases I'm authoring/debugging. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide the implementation to execute test cases from command line and options and value pair provided at command line should take over precedence. |
| Comments | As an Automation Engineer, I should provide implementation to override the default parameter value, with the value specified in command line for parameter.   * The expected parameters from command line are * help * environment (Default – UAT) * browser (Default – Chrome) * operating-system (Default – Windows) * language (Default – English) * source-URL (Default – Current URL) * test-suite-file-location (Default – Test data folder in project) * test-report-file-location (Default – reports folder in project) * test-log-location (Default – logs folder in project) * bd-connection-details (Default – UAT environment db connection details) * assertion-required (Default – yes) * component (Default – yes) * we will add few more based on requirement. * When you provide test-suite-file-location parameter along with value in command line execution, it should take precedence over the default value * When no value provided only parameter provided, it should throw an exception with proper error message in report. |
| Component Mapping | Command line Execution - User can able to execute desired subset of text cases or new test cases from command line using different parameter -Testcase Sheet |

# Use Case – 26

|  |  |
| --- | --- |
| Requirement ID | 6307 |
| Requirement | Run Jenkins job with the latest code base from Git |
| User Story | As Ari (or Shaili), I should be able to run tests/test suite from Jenkins. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide GIT and Jenkins configuration. Jenkins job to fetch the latest code from GIT repository and execute the test automation suite.   * Should be able to check the code in git branch * Should be able create a Jenkins job which will pull the latest baseline code from git branch * Should be able create a Jenkins job which will execute the latest code with default or provided values. |
| Comments |  |
| Component Mapping | Jenkins Execution – Jenkins job created to execute get latest codebase from git and execute the configured test suite pack configured. |

# Use Case – 27

|  |  |
| --- | --- |
| Requirement ID | 1588 |
| Requirement | Preserve test results and logs |
| User Story | As Ari (or Shaili), I should be able to preserve test results, logs, screenshots/videos so that I can conclude RCA after a few days. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide configuration in Jenkins to store each execution result separately.   * Should be able to save the reports and logs in the specified location * Should be able to get the previous reports trend in the current execution * There will be implementation mechanism to provide identify most failure by comparing the previous execution and test logs * For every failure we need capture product log files (Engineering team log files provided in development server) * Require access and authentication to connect to product server logs * In addition to test automation logs, screen shots and other required artefacts, should provide logs accessed from product server * It will be better used for developer, to get the exact failure reasons. |
| Comments | Jenkins instance should be configured to store results of each execution results separately. |
| Component Mapping | Jenkins Execution – Jenkins configuration mapped properly to execute and store results at desired configurable location |

# Use Case – 28

|  |  |
| --- | --- |
| Requirement ID | 6124 |
| Requirement | Email test run report |
| User Story | As Shaili, I should get an email of a test run so that I can determine the health of the automation test suite. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to generate emailable reports with predefined format to different stake holders   * There should be implementation to provide way of communication to save in specified folder and send email to desired recipients * Recipients also should be configured based on suite type * Email body content should be mapped with suite type |
| Comments | Summary |
| Component Mapping | Jenkins Execution – Jenkins configuration mapped properly to execute and send email results in defined format to required stake holders |

# Use Case – 29

|  |  |
| --- | --- |
| Requirement ID | 5315 |
| Requirement | Repeat test scenarios |
| User Story | As Ari (or Shaili), I should have the option to provide the testcase execution |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation and configuration to execute the test cases repeatedly   * There shall be implementation to be provided repeated test case execution * Test data must be provided for each execution iteration |
| Comments |  |
| Component Mapping | TBD |

# Use Case – 30

|  |  |
| --- | --- |
| Requirement ID | 3247 |
| Requirement | Invoke a scenario multiple time |
| User Story | As Shaili (or Ari), I should be able to run a test scenario multiple times with different values for each run so that I can write clean test scenarios in Excel without having to repeat it as many times. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to write test cases in excel file with configurable value as regular expressions.   * There shall be implementation to be provided invoke test case multiple times * There should be provision to allow test data based on the regular expressions. |
| Comments | * It should be possible to specify multiple values for an operation and pass those values to subsequent steps for verification (or correct functioning). Ex. Step1(a##1, b##2, c##3); Step2(a##1, b##2); Step3(c##3); * The keyword ## indicates a 4-digit random number and a number next to it indicates it ordinal. |
| Component Mapping | Prepare Module- There is dynamic generation of test data to corelate between test case ordinal execution and test to be mapped. |

# Use Case – 31

|  |  |
| --- | --- |
| Requirement ID | 7488 |
| Requirement | Invoke a test scenario with REST API |
| User Story | As Ari (or Shaili), I should be able to invoke any test scenario using a REST API so that I can use it for other tests such as performance testing. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to expose each test case as REST API call   * These should be implementation to publish each Test Scenario as API * The Test data input will be provided as body for the rest API call * The report file location will be output of the API call. * There should be configuration to publish list of services with input and output parameters. |
| Comments | It should be possible to invoke all automated test case scenarios with a REST API. |
| Component Mapping | Execution Module – There is a REST API provided to individual test case and at the whole suite level, so that other components can access this test case/suite with Rest API call. |

# Use Case – 32

|  |  |
| --- | --- |
| Requirement ID | 5503 |
| Requirement | Specify the location of reports |
| User Story | As Ari (or Shaili),i should be able to configure the reports location and its accessible from anywhere |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to provide parameters from command line. If local value provided on command line, it should take precedence over the default value.   * The expected parameters from command line are * help * environment (Default – UAT) * browser (Default – Chrome) * operating-system (Default – Windows) * language (Default – English) * source-URL (Default – Current URL) * test-suite-file-location (Default – Test data folder in project) * test-report-file-location (Default – reports folder in project) * test-log-location (Default – logs folder in project) * bd-connection-details (Default – UAT environment db connection details) * assertion-required (Default – yes) * component (Default – yes) * we will add few more based on requirement. * When you provide test-report-file-location parameter along with value in command line execution, it should take precedence over the default value * When no value provided only parameter provided, it should throw an exception with proper error message in report. |
| Comments | For command line execution, it should take as a parameter. For other executions, reports location should be in a configuration file as a parameter. |
| Component Mapping | Configure Module, Report Module - There is configuration value to specify the location of reports. Reporting module takes that path from configuration and prepares the report at specified location.  If the custom report path defined on Command line, execution will take the local specified value. |

# Use Case – 33

|  |  |
| --- | --- |
| Requirement ID | 2726 |
| Requirement | Log all actions and result for easier analysis |
| User Story | As Ari (or Shaili), I should be able to get the log trace of the full test execution |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation for logging mechanism for all events occurring during the execution. The logging should be like pluggable interface   * Logging option should be pluggable option * Implementation should be available, to configure and select one of the reporting options available (Jasmine, Log4J…) * Based on the logging option provided in the command line or Jenkins job, that specific log should be generated * If the no logging option provided, default option should be generated as log4j report * If unavailable logging option provided, executor should quit with proper error message to user / should generate jasmine report * LoggingManager package should be abstract class, where only method name definitions available. * For each option it should be and extendable class for LoggingManager, where exact implementation to that option available. * When new option included, just one more extend class with that report option functionality available without changing the base class and other option classes. |
| Comments | * Integration of Log4J logging in Framework for ease of automated execution analysis * Log file should contain the pass/fail of the testcases and should capture errors/exceptions in detail. It should also contain application logs. |
| Component Mapping | Report Module – Reports and Logs gets generated in case of both pass and failure scenarios. Execution captures the test step and case specific error logs in case of exceptions, provides detailed level logs and reports. |

# Use Case – 34

|  |  |
| --- | --- |
| Requirement ID | 3185 |
| Requirement | Navigate across screens of same as well as multiple applications |
| User Story | As Ari (or Shaili), i should be able to navigate across screens of same as well as multiple applications during the test execution |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should be able to provide implementation to move across different pages in application as well as among different applications. |
| Comments |  |
| Component Mapping | Prepare Module – User provided with options / steps to move across the pages in application. Able to move across different application url’s |

# Use Case – 35

|  |  |
| --- | --- |
| Requirement ID | 1094 |
| Requirement | Test product with various languages |
| User Story | As Ari (or Shaili), I should be able to run test automation by setting up any language the application is translated to so that I can ensure the application works well with non-default (English) language. |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation which will provide validation for different languages with minimal configuration changes.   * The expected parameters from command line are * help * environment (Default – UAT) * browser (Default – Chrome) * operating-system (Default – Windows) * language (Default – English) * source-URL (Default – Current URL) * test-suite-file-location (Default – Test data folder in project) * test-report-file-location (Default – reports folder in project) * test-log-location (Default – logs folder in project) * bd-connection-details (Default – UAT environment db connection details) * assertion-required (Default – yes) * component (Default – yes) * we will add few more based on requirement. * When you provide language parameter along with value in command line execution, it should take precedence over the default value * When no value provided only parameter provided, it should throw an exception with proper error message in report. * If comma separated value provided execution should perform for all provided values as well as report and log generation for each language separately |
| Comments | Framework should be capable of supporting defined languages |
| Component Mapping | Configure Module, Execute Module - localization support required without changing the code.  Automation should support validating the different languages. |

# Use Case – 36

|  |  |
| --- | --- |
| Requirement ID | 7541 |
| Requirement | Compare between baseline and actual result |
| User Story | As Ari (or Shaili), I should get Test results with detailed information about Expected and Actual result.   * After obtaining the results from the UI, Levels of validation   + will be comparing the UI value with the API Response value for the same operation   + will be comparing the expected value provided through excel cell with the API Response value for the same operation   + will be comparing the UI value with the Database value retrieved through API call * Most of the times, KSH module application services will be used to obtain the values from backend * If KSH API is not available, try to call the trace flow from the engineering side and call the same Rest API * In case of unavailability of Point no.2 & 3, need to build and publish rest API to get the validation done * Database access cannot be done directly due to security reasons, so all required queries should be encapsulated in Rest API and get published. * In case of assertion failure on comparison of UI value with API response, the failure will be forwarded to the report module and test case failure resulted based on the scenario |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation which report the granular information on comparison between actual and expected result. |
| Comments |  |
| Component Mapping | Execute Module – Execution process validates the actual values which against the expected value mentioned in excel sheet. |

# Use Case – 37

|  |  |
| --- | --- |
| Requirement ID | 4028 |
| Requirement | Test data integrity (Data comparison between graphs and DB) |
| User Story |  |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to validate the data populated against the graphs with DB values. |
| Comments |  |
| Component Mapping |  |

# Use Case – 38

|  |  |
| --- | --- |
| Requirement ID | 8122 |
| Requirement | Obtain baseline code coverage and document enhancements |
| User Story |  |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide configuration which provides code coverage analysis. |
| Comments |  |
| Component Mapping | Report Module – Third party tool integration of code coverage tool provides the code coverage analysis. |

# Use Case – 39

|  |  |
| --- | --- |
| Requirement ID | 6770 |
| Requirement | Integrate with TESTNG logging framework |
| User Story |  |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should be able to provide implementation for pluggable logging Module, so that whichever logging requirement that implementation can be available with minimal amendments.   * Logging option should be pluggable option * Implementation should be available, to configure and select one of the reporting options available (Jasmine, Log4J…) * Based on the logging option provided in the command line or Jenkins job, that specific log should be generated * If the no logging option provided, default option should be generated as log4j report * If unavailable logging option provided, executor should quit with proper error message to user / should generate jasmine report * LoggingManager package should be abstract class, where only method name definitions available. * For each option it should be and extendable class for LoggingManager, where exact implementation to that option available. * When new option included, just one more extend class with that report option functionality available without changing the base class and other option classes. |
| Comments | for ease of automated execution analysis |
| Component Mapping | Report Module – Reports and Logs gets generated in case of both pass and failure scenarios. Execution captures the test step and case specific error logs in case of exceptions, provides detailed level logs and reports. |

# Use Case – 40

|  |  |
| --- | --- |
| Requirement ID | 2185 |
| Requirement | Execute tests in parallel |
| User Story |  |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation or configuration to execute test cases in parallel on same or different browsers. |
| Comments | grouping of tests, tests prioritization, data parameterization, logs, results |
| Component Mapping |  |

# Use Case – 41

|  |  |
| --- | --- |
| Requirement ID | 6082 |
| Requirement | Capture responses in separate file for ease of analysis during post execution |
| User Story |  |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to gather all responses in independent file for each test case, which would be eventually helpful for easy analysis. |
| Comments |  |
| Component Mapping | Report Module – Reports and Logs gets generated in case of both pass and failure scenarios. Execution captures the test step and case specific error logs in case of exceptions, provides detailed level logs and reports. |

# Use Case – 42

|  |  |
| --- | --- |
| Requirement ID | 4859 |
| Requirement | Framework should be well integrated with functional and non-functional tests including Performance, Security, Accessibility (WCAG 2.1) |
| User Story |  |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to integrate with non-functional execution activities. |
| Comments |  |
| Component Mapping | Integration with other Test Activities - |

# Use Case – 43

|  |  |
| --- | --- |
| Requirement ID | 9400 |
| Requirement | Specify dependencies between tests |
| User Story | As Shaili (or Ari), I should be able to specify dependencies between automated test cases so that I'm assured the prerequisite tests are executed before the "final" test |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to take care of predefined steps / dependent test cases which should execute prior to current test case execution. |
| Comments | A test may have a few steps that must be executed prior to it. |
| Component Mapping | Configuration Module, Prepare Module – There is test case dependency provided with Test Case dependency and Forced execution parameters defined in Test Suite Configuration |

# Use Case – 44

|  |  |
| --- | --- |
| Requirement ID | 5959 |
| Requirement | Execute from command line with help option |
| User Story | As Ari (or Shaili), I should be able run tests/test suites from command line and –help should provide help for command line |
| DevOps User Story ID |  |
| Acceptance Criteria | As an Automation Engineer, I should provide implementation to execute the test cases from command line.  --help parameter provides the help of all options available to execute at command line |
| Comments | ex:  npm –help, should provide help for command line parameters |
| Component Mapping | Command line Execution - User can able to execute test suite from command line using different parameters like config file, properties file and other required parameters for execution |

# Data Design

# Data Structure

All Data structures used for Test Suite, Test case and Test data detailed in respective places.

Attached Here for Reference:







# Database

Configuration Module details about the which database to connect based on the execution environment using

SQL Manager Type Script – CreateConnection, ExecuteQuery, CloseConnection functions

Sample file attached here for reference



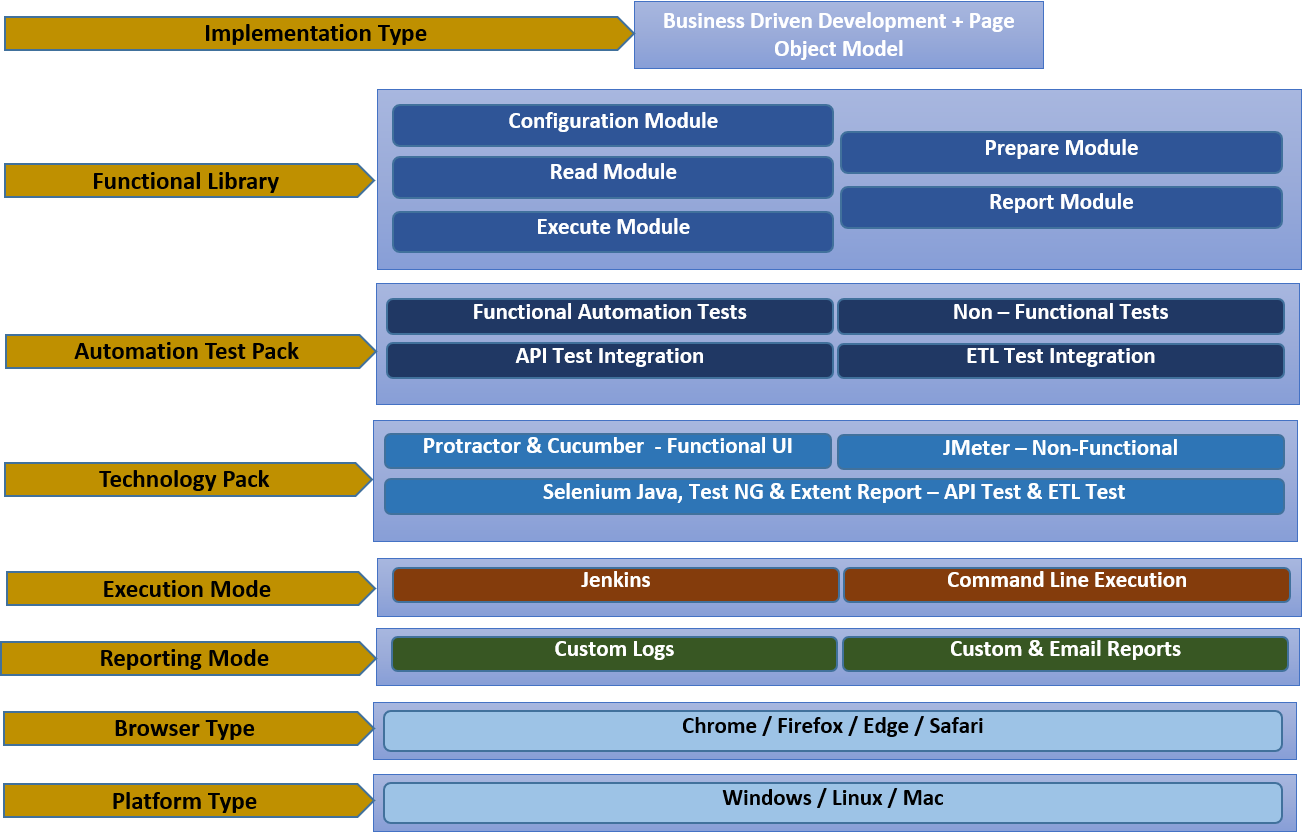


# Configuration

All the configuration details available in above two sections attachments.

# Automation Architecture / Component Overview

# High Level Automation Architecture



The above diagram depicts the High-Level architecture of Test Automation activity carried out during the design, implementation, execution & delivery phases.

Implementation Type:

This section is detailing about how the Business-Driven Development and Page Object Model integration to drive to achieve automation requirements provided.

Functional Library:

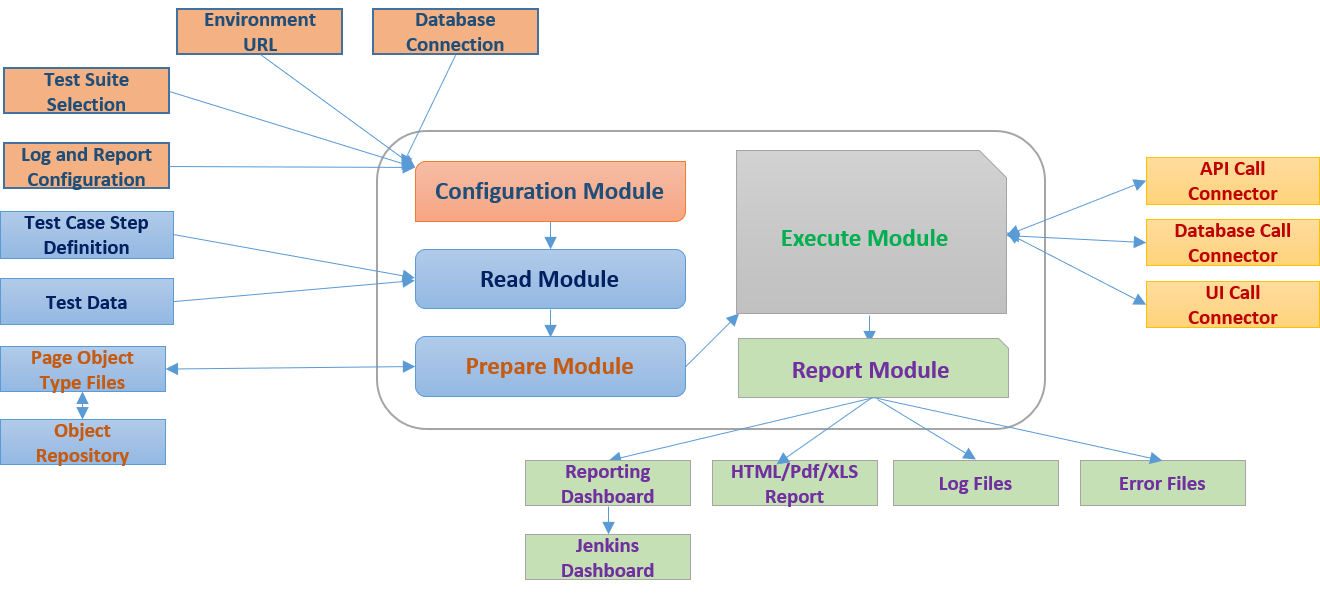
This module details about

1. Configuration module roles & responsibilities
2. Where the Integration module take control
3. What is the functionality evolved in Read module
4. What are characteristics of Prepare module
5. How the execute module works

The way of reporting and Logging implemented

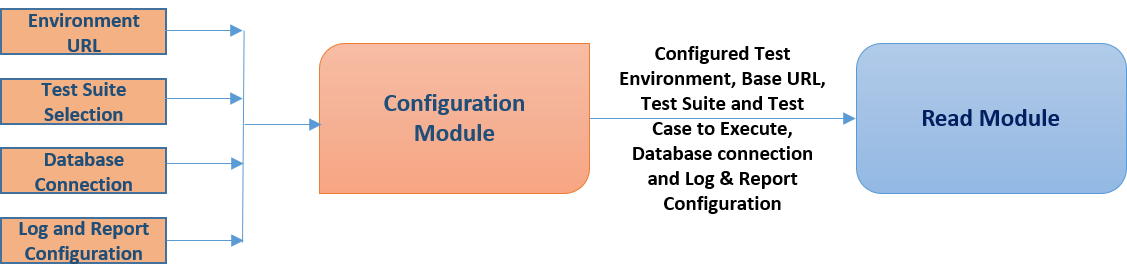
# Component Overview

# Run Time View of Framework



# Configuration Module

* This Module is to configure environment
* Prepare data base connection and establish connection
* Provide the access URL’s based on selected environment
* Test Suite Configuration – which suite to select for execution
* Test Module Configuration – which module for execution
* Test Case Configuration – which test cases to execute



Config\_Properties.json: Property file for UI Automation Framework

Environment manager class: This will read properties from config. properties and manage it

Test Execution Configuration class: This will set which test cases to execute based on test execution configuration

Data Base Configuration class: this will set the Database connection based on the environment selection and provides the connection

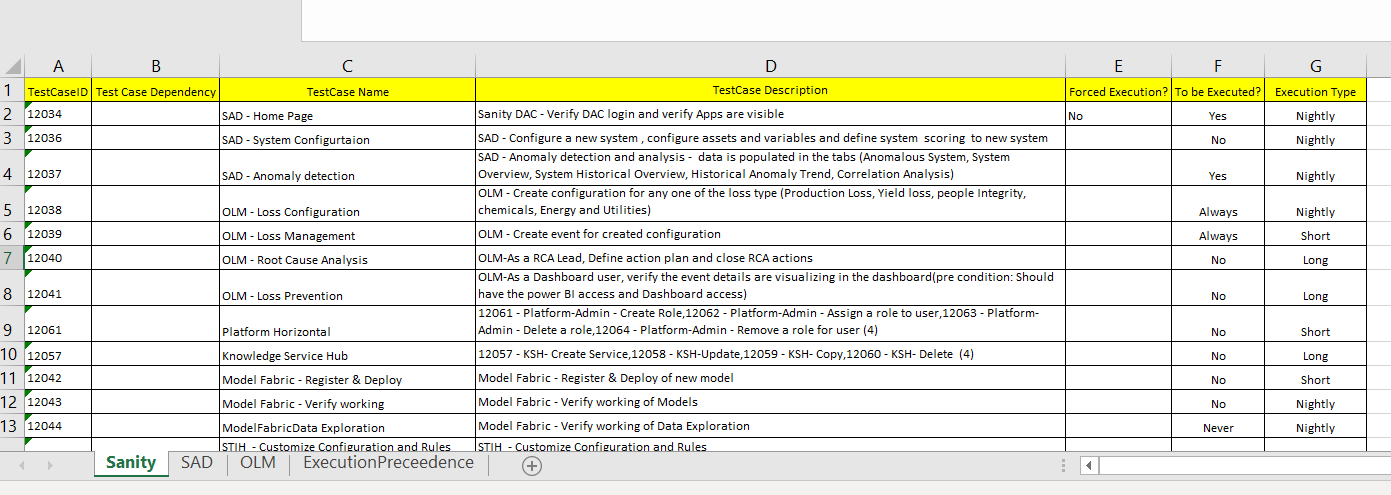
**Sample configuration file looks like:**



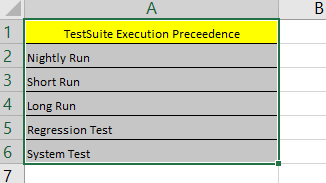
Based on Environment the associated properties will be mapped to build required environment for execution.



**Test Case Suite Configuration:**



**Test Case Suite Execution Precedence:**

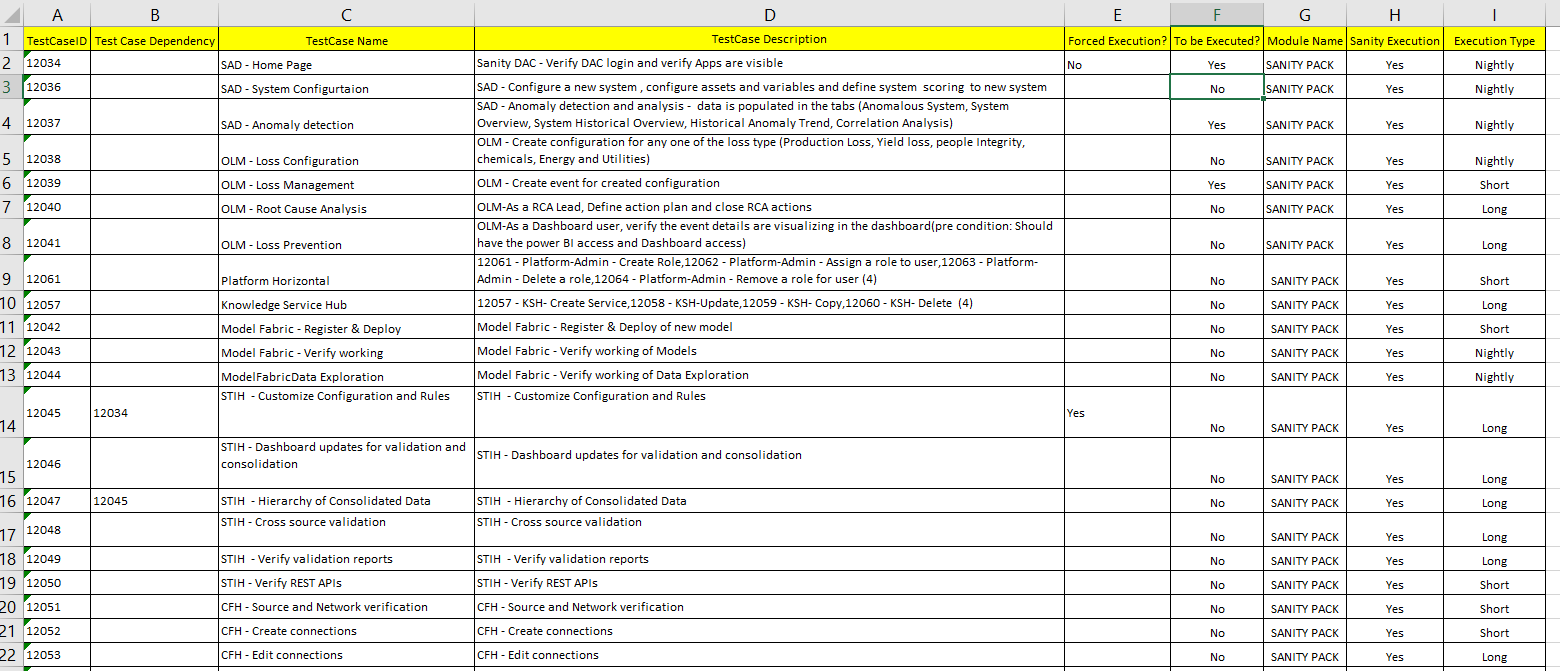


User should be able to provide Configuration which Test cases come under Sanity Pack Execution

There should be provision to select what kind of execution user want to run:

1. Sanity Pack
2. Module Wise
3. Nightly Run
4. Short Run
5. Long Run
6. Regression Test
7. System Test

**Test Case Dependencies:**



Dependency of the test case will be capitalized from ‘Test Case ID’, ‘Test Case Dependency’ , ‘Forced Execution’ and ‘To be Executed’ (As shown in picture above):

**Testcase Dependency:**

If one test case is prerequisite for other test cases to be executed that prerequisite/dependency test case ID should be mention in ‘Test Case Dependencies’ column.

Ex: TC 12047 execution is dependent on TC 12045 Testcase execution – First automation process will execute 12045 then 12047 will get executed

TC 12045 execution in dependent on TC 12034 Test case execution – First automation process will execute 12034 then 12045 will get executed.

Overall execution process will be 12034 -> 12045 -> 12047

**Forced Execution:**

As default each test case executability will be once in execution session.

But in case of dependencies,

* If the ‘Forced Execution’ set to ‘Yes’ for a test case to be executed again, no. of times it gets called as dependency
* If the ‘Forced Execution’ set to ‘No’ for a test case to be executed once for session, irrespective of the test case called no. of times as dependency.

Ex: 12045 will be executed no. of times it gets called in the execution session

12034 will be executed Only Once, though it gets called no. of times in execution session

**To Be Executed:**

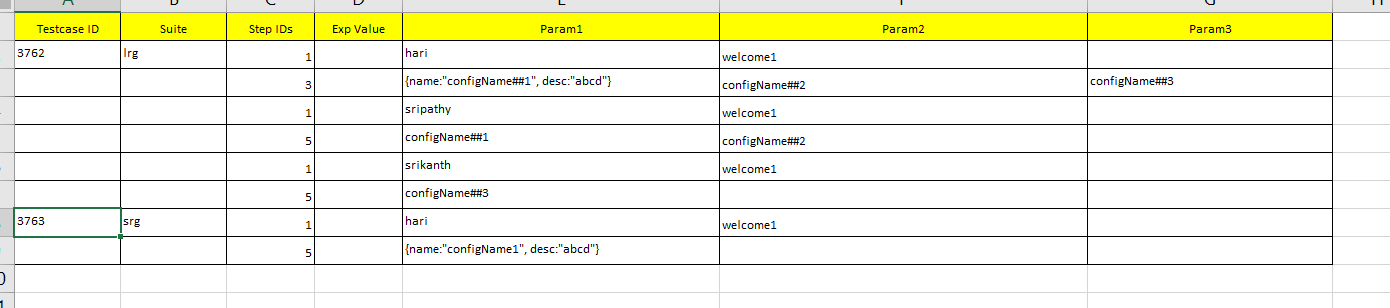
Test case to be executed or not will be dependent on ‘To be Executed’ Column.

* + If the ‘To be Executed’ set to ‘Yes’, the test case will be executed in normal flow of execution
  + If the ‘To be Executed’ set to ‘No’, the test case will not be executed in normal flow of execution
  + In case of dependencies, irrespective ‘To be Executed’ value, the test case will be executed if it is executed as prerequisite/dependency

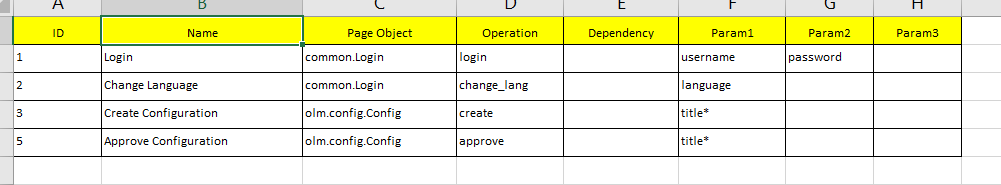


**Test Case Organization:**

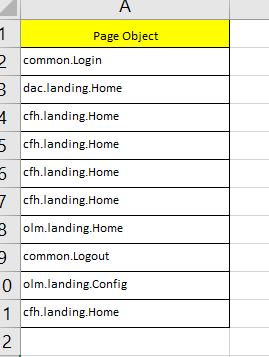
Test Case Setup:



Test Step Setup:



Page Object Lookup:

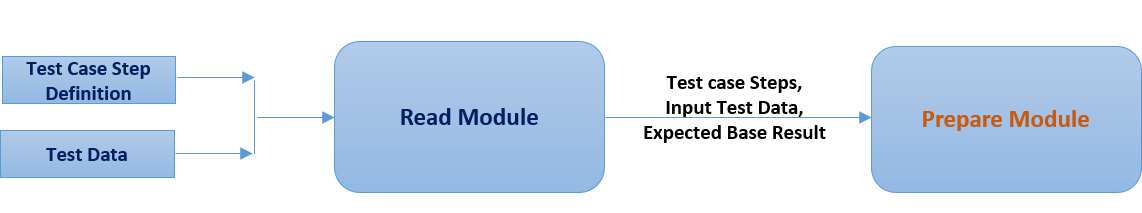




# Read Module

This module maintains

* the input file regarding configuration details, where user will provide input data such as Environment URL’s, Database properties and provide the same to Configuration module.
* the input file of test data with respect to test cases of Configuration module and sends input and output test data



Excel Reader class: This class is used to read the input excel data.xls

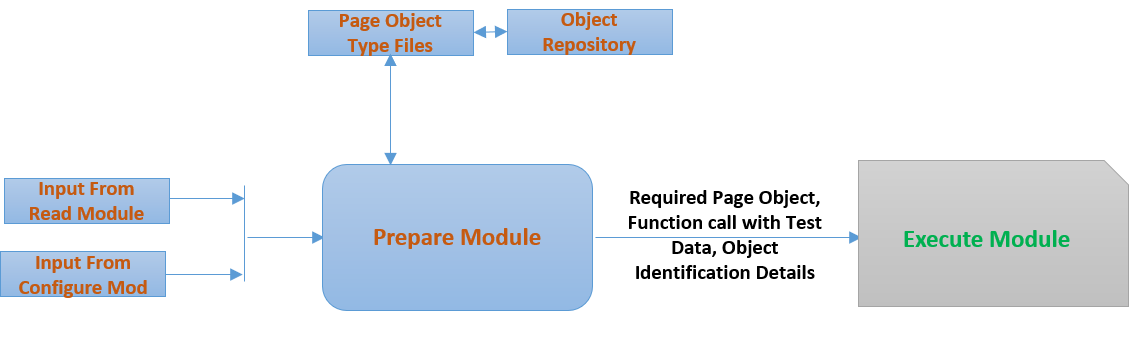
Base class: This class is used to store the excel input data details

Test Case Reader class: Based on the type of execution, this class should collect the test cases to be executed and it should collect the required input data to execute the particular test case.

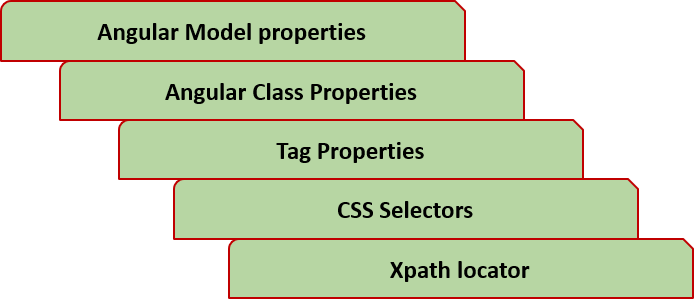
# Prepare Module

Config.properties: Property file for Test case execution based on direct UI call or API call/ETL call in parallel to UI Test and provides the necessary artefacts to the process.

Test case prepare class: This will set the input file on required environment, with required BD Pipeline and Set the expected output



# Object Locator Strategy



Object Identification Strategy Defined as

* As priority trying to find unique object by angular model properties - binding, model, options, repeater required to locate the angular attributes, ng-bind, ng-model, ng-options, ng-repeat.
* To identify with Angular class properties which lead unique identification object
* To identify element with Tag properties which can identify unique property - text/partial text locators such as buttonText and partialButtonText
* To identify with CSS selectors, if defined any - elements using CSS selectors containing text using cssContainingText locator.
* As least priority, to identify element with configured Xpath for that object

This page object strategy for each object get defined in the respective page object file of the object being identified.

# Page Object Classes

# Common Reusable files

ExcelManager

SQLManager

EnvironmentManager

TestSuiteReader

TestCaseReader

Reporter

Logger

ReusableManager

UtilityManager

# Sign In page

Elements/Objects:

txtbox\_SignInPage\_SignIn

btn\_SignInPage\_Next

Methods:

Navigate – To open browser with configured URL based on the environment

LoginDAC – To Login to application with registered mail ID

# Home Page

Element/Objects:

btn\_HomePage\_AllApplicationsIcon

btn\_HomePage\_DigitalAppsIcon

btn\_HomePage\_ContextualFusionHubIcon

btn\_HomePage\_SystemTwinIntegrityHubIcon

btn\_HomePage\_KnowledgeServiceHubIcon

btn\_HomePage\_ModelFabricIcon

btn\_HomePage\_AnalyticsAppsStudioIcon

btn\_HomePage\_IndustryInsightIcon

btn\_HomePage\_AssetTwinViewerIcon

btn\_HomePage\_PlatformAdministrationIcon

Methods:

VerifyHomePageElements – to check all the application elements available on Application Home Page.

MoveToApplication – to move to the desired application, from the available applications by clicking on application icon.

# Application Page

NA

# Digital Apps Center

NA

# System Anomaly Detection

Element/Objects:

Ele\_SADHomePage\_SystemConfigurtaion

Ele\_SADHomePage\_AnomalyDetection

Methods:

SADMoveToTab - To move the required option from System Configurtaion or

Anomaly Detection.

# System Configuration

Element/Objects:

btn\_SystemConfigurationPage\_NewSystem

input\_SystemConfigurationPage\_Search

btn\_SystemConfigurationPage\_CloseSearch

btn\_SystemConfigurationPage\_searchSearch

txtbox\_SystemConfigurationPage\_SystemName

txtbox\_SystemConfigurationPage\_Description

select\_SystemConfigurationPage\_Class

btn\_SystemConfigurationPage\_SerachLocation

select\_SystemConfigurationPage\_Country

select\_SystemConfigurationPage\_Region

select\_SystemConfigurationPage\_Plant

select\_SystemConfigurationPage\_Area

select\_SystemConfigurationPage\_Unit

btn\_SystemConfigurationPage\_Cancel

btn\_SystemConfigurationPage\_Apply

btn\_SystemConfigurationPage\_Save

btn\_SystemConfigurationPage\_Submit

select\_SystemConfigurationPage\_Model

btn\_SystemConfigurationPage\_Upload

# Anomaly Detection

# Opportunity Loss Manager

# Loss Configuration

# Loss Management

# Pipeline Integrity

# Machine Performance Analysis

# Major Accident Hazard Analysis

# ABC Analysis

# Condition Based Maintenance

# Asset Life Assessment

# Terminal Performance

# Contextual Fusion Hub

# Establish Connection

# Metadata Mapping

# Data Viewer

# System Twin Integrity Hub

# Asset Information

# Validation

# Master Data Consolidation

# System Hierarchy Transformation

# Knowledge Services Hub

# Business Service

# Cognitive Storage layer

# Model Fabric

# Data Exploration

# Model Registry

# Deployment

# Analytics App Studio

# KPI Designer

# Dashboard Designer

# Workflow Engine

# Collaboration Engine

# Manage User Data

# Industry Insights

# Energy

# Process

# Marine & Ports

# Measurement & Analytics

# Asset Twin Viewer

# Plant Hierarchy

# Maintenance

# Inspection

# Simulation

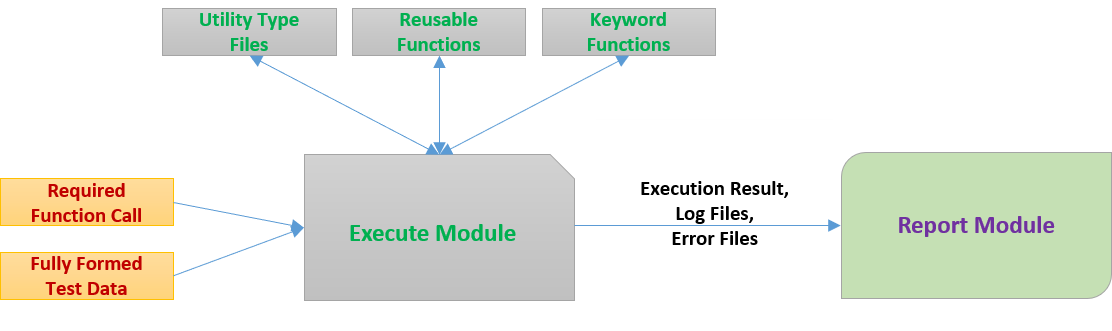
# Platform Administration

# Roles & User Management

# Application Provisioning

# Execute Module

This module is to execute the test cases prepared from Prepare Module.



UI utility classes:

This is base class which implements all the UI common and executing methods, The Methods available in UI Utility Class:

Select\_List\_Option\_Value

Select\_Checkbox

DeSelect\_Checkbox

Select\_ListElement

mouseOver

Drag\_And\_Drop

Enter\_TextBox\_Value

Click\_Element

GetTxt\_Element

Upload\_File

Verify\_Element\_Exist

ValidateData

VaidateTableRowData

ValidateTableCellData

MoveToPage

Database utility class:

Using this class, we will execute the SQL queries and get the DB records, The Methods available are:

dbCreateConnection

getDBTableEmptyStatus

getDBTableResult

executeQuery

dbCloseConnection

Text file comparator:

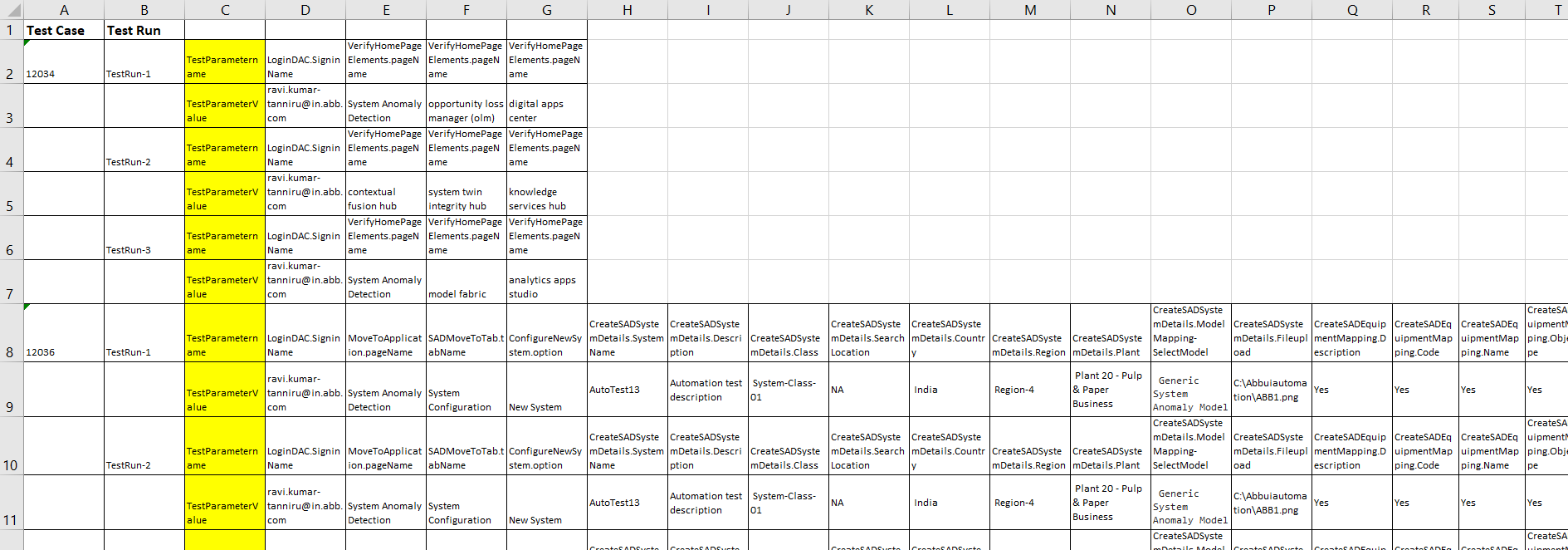
Using text file comparator, we will compare both Expected vs actual files data line by line.

File upload utility:

Using file upload utility, we can upload source data (test data) in to CDL Blob storage using App Studio File upload utility API.

Dynamic Result validation: for every UI validation call, there will be respective API call (from KSH module or API call to be build) which will dynamically retrieves/generates the expected result and validate actual result on the UI as in separate thread.

**Multiple Test Run Execution**



# 

Each Test Case – Test Data will be configured across multiple Test Run’s

Based on the Test Run execution condition particular test run data will be mapped and executed.



# Report Module

Jasmine Reporting Type script file

Using this class, we will generate the HTML report after the test execution completes.

User provide with following artifacts from Reporting Module

* HTML/PDF/XLS and emailable reports
* Log Files
* Error Logs
* Reporting Dashboards – Jenkins

Jasmine Execution Dashboard provide below details:

* Test Suite execution over all summary
* Test case execution detailed details
* Test Execution time metrics (Test case and Test suite level)
* Overall Test execution summary (All test suite execution history)
* Execution trend analysis (detailed analysis of execution coverage, Pass and Fail summary, execution time metrics)

We should provide different kind of reports (Jasmine, TestNG, Extent) – based on feasibility and applicability specific report selection happen though configuration.

Report module designed as pluggable component, can add new format report easily with minimal amendments.

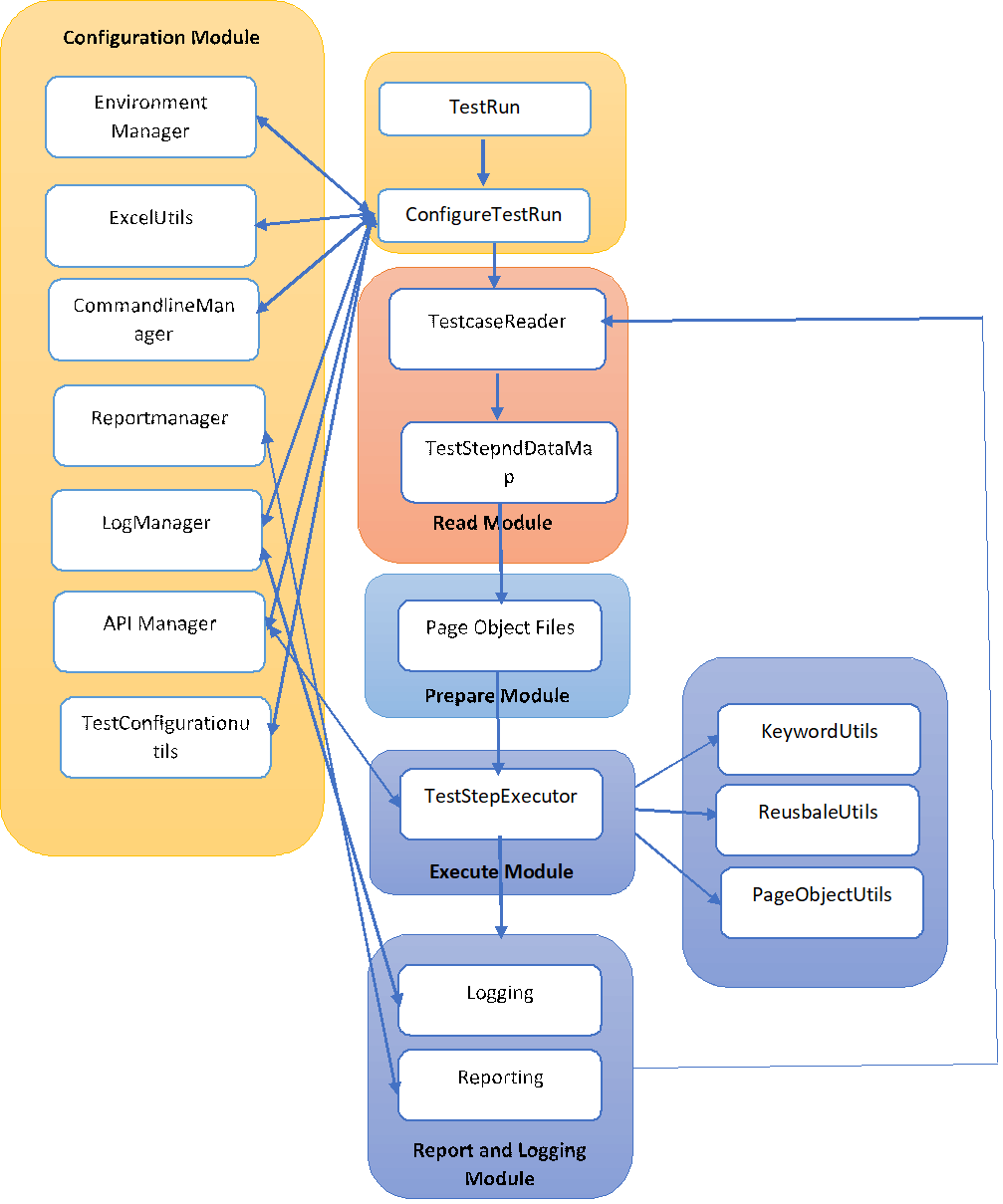
Logging strategy:

* Each action should be logged
* There should be log management for overall test suite and test case level execution
* Post execution completion of test case, test case level log should be saved.
* Post execution completion of test suite, test suite level log should be saved.
* In case of error/exception – specific exception or error log should be attached/referenced to that step for easy analysis.
* Screen shot should be attached in case of error / failure.

We should provide different kind of logging strategy (log4j, TestNG) – based on feasibility and applicability specific report selection happen though configuration.

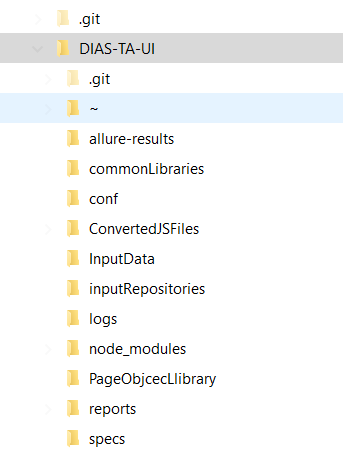
Logging module designed as pluggable component, can add new format logging strategy easily with minimal amendments.

# Class Diagram



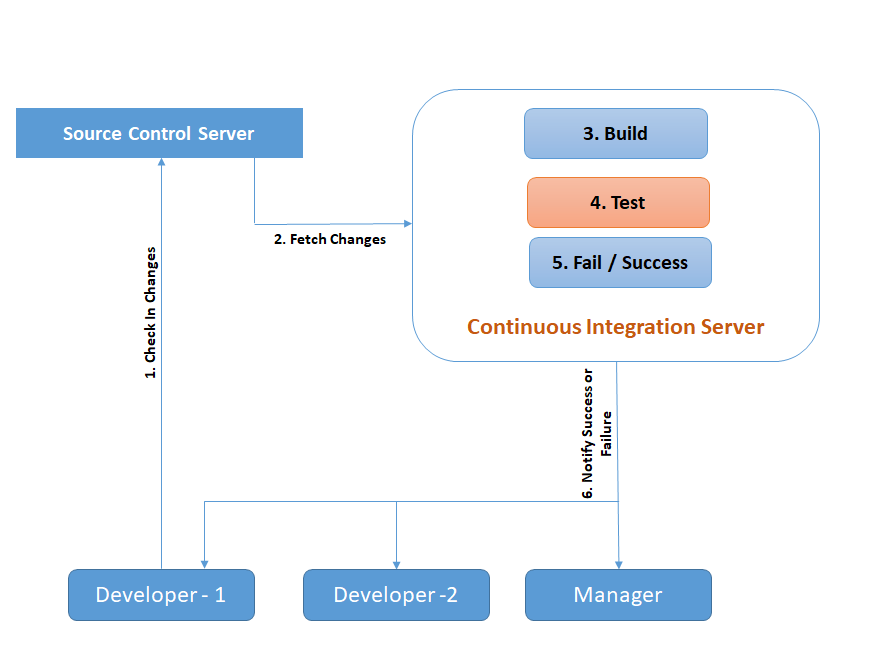
# 

# Folder Structure



# Deployment Overview

# Deployment Architecture



# CI/CD Integration

GIT Code location – DIAS-TA-UI

GIT UTL - <TBD>

Frequently used commands in GIT

1. Command to clone “git clone <repo url>” (Check-out)
2. Command to add all file from project to git “git add -A”
3. Command to commit from workspace to git “git commit -m “comment for commit”
4. Command to push from workspace to GitHub “git push” (Check-in)

Jenkin Jobs Management

Procedure to start Jenkins:

1. Open command prompt and go to the path where the Jenkins. War file is present (Example: In VM 51.137.38.156 the path would be “cd C:\Users\qeadmin\Documents\”)
2. Run the command “java -jar jenkins.war”
3. Launch any Browser and go to <http://localhost:8080/> (Port 8080 is where the Jenkins run)

Procedure to run Job in Jenkins:

1. Select the Job (UIAutomation\_GIT) in Jenkins Dashboard -> Select “Build with Parameters”
2. Select the XML file for the execution -> Select “Build” and the execution begins

Jenkins XML files used:

The command used for execution in Jenkins: “clean test -DsuiteXMLFile=$XML\_file”

where XML\_file can be for all test suites or a specific test suite

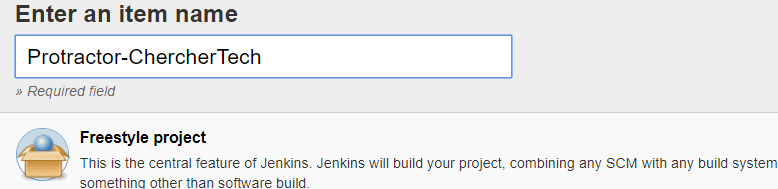
# Windows Batch file

# Open Jenkins using http://localhost:8080/

# Click on New Item



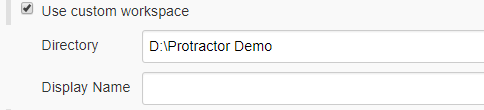
# Enter Job Name and Choose Free Style Project



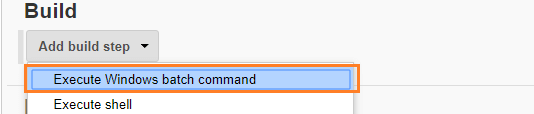
# Choose advanced options under the General section

general-sectionjenkins-custom-workspace

# Enter the folder where your protractor conf file is present



# Now Add a Build Step and Choose Execute Windows Batch Command



# Provide the command to run the protractor, protractor conf.js. If user have not configured the workspace path, then user need to write the command to navigate to the workspace and then protractor run command like below

d: # change the driver to d

cd D:Protractor Demo

# navigate to the folder where the conf file is present.

protractor conf.js

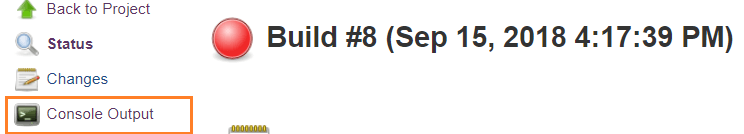
# Save the configuration and Click Build Now Link



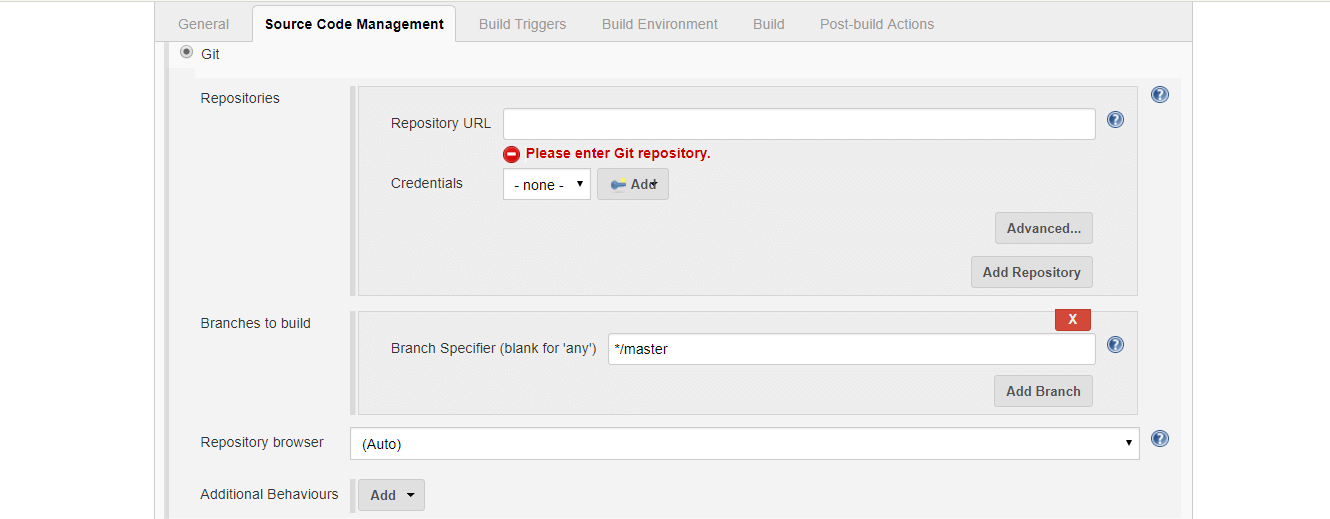
# Now user will be able to see the execution, under console inside the instance number

Fetch Code from Repository:

Not all the times, user will have to run in the local machine because you got to execute the automation in node/slave machines.  
  
In such cases, we have to fetch the code from the repository and execute if user code Javascript then user can run directly, but if it is Typescript then user need to compile the code(tsc) and then run it.



Under Source Management, choose the required tool and then provide the git URL and then the branch to clone.



User might need to change the build steps because this time, you should install the npm packages and then compile the code and then execute.

d: # change the driver to d

cd D:Protractor Demo # navigate to the folder where the conf file is present.

npm install protractor

#please do install required, I have limited with protractor to show you

tsc

protractor conf.js

# Command line Execution

npm run e2e -- --base-url=http://testurl:8080 --suite=suite\_name\_defined\_in\_config --capabilities.browserName=browser\_Name

where npm package.json :

"e2e": "protractor tests/protractor-conf.js",

and config file contains :

suites: {

BVT: 'e2e/TestSuites/\_BVT/\*.js',

Full: 'e2e/TestSuites/Full/\*\*/\*.js',

},

capabilities: {

'browserName': 'chrome'

},

baseUrl: 'http://localhost:8080/',

Parameters available on command line are

--help – to provide help of all command line parameters usage

--report-location

--testsuite-file

--testcase-file

--environment

--config-file

--browser

--report

--log-location

--log

# Non-Functional Testing

# Risks, Assumptions and Dependencies

# Assumptions

# Technical Assumptions

1. *This Design documented in view of framework design irrespective of implementation technology being considered, so based on the implementation technology some of the use cases not at all possible to implement, In that case those test case acceptance criteria need to changed with valid reasons of implementation challenges.*
2. Scope is limited to testing following features on the Edge platform:
   1. Connectivity and mapping configuration to data source systems.
   2. Data Integration with source systems - Edge Systems, IT Systems, Geospatial Systems.
   3. Data file compression and transfer to cloud for bulk data.
3. Following data transfer needs to be tested for Data movement to Edge and cloud platform:
   1. RPC
   2. AMQP
   3. Message broker based on MQTT, RabbitMQ, Mosquitto & IoTHub
   4. JDBC/ODBC
4. Scope is limited to testing following Data storages:
   1. Cloud BLOB Storage
   2. Edge local storage
   3. Cosmos DB
5. Big data volume testing for Genix platform will be performed
6. The personas to be included in the scope of testing is 3 (Admin, Business IT User, Developer, and Business User)
7. Approximately 6 Batch data sources needs to be tested as part of the data transformation testing.
8. 6 to 8 models (both model objects and Rest APIs) needs to be tested as part of the scope.
9. Number of business value Apps to be verified is 5 and number of screens to be tested/verified is 30

# Infrastructure Assumptions

1. Assuming all these activities are carried out in the ABBGISPL - Bengaluru BTP premises due to hardware dependency constraints, on some occasions, based on the need, the work will be done at the HCL premises.
2. ABBGISPL to provide devices to HCL which must be returned after the project or on demand by ABBGISPL.

# Risks

Risk factors will be tracked continuously, and timely action will be taken to mitigate the risk. HCL and ABBGISPL will jointly work towards the mitigation with appropriate risk mitigation plans and contingency plans.

|  |  |  |
| --- | --- | --- |
| **Risks** | **Severity/ Impact** | **Mitigation** |
| Non-availability of HCL resources | Delay in Deliverables | Support will be provided over & above the agreed timelines as per the sprint plan under this SOW without additional cost to ABBGISPL. (The milestones will be pro-rated accordingly)  Any planned leaves by an engineer will be duly discussed and agreed with ABBGISPL. VENDOR engineers are entitled to one day of planned leave per month, on an annual cumulative basis. |
| Non-Availability of User Stories | Unutilized bandwidth | ABBGISPL will provide alternate user stories after mutual agreement else ABBGISPL to consider the unutilized bandwidth as milestone deliveries (equivalent User stories) and pay for the proportionate period. |
| Change in scope | Schedule, effort and budget overruns. Impact to Quality. | Well defined change management process in place will be triggered.  ABBGISPL will provide the approvals for change in scope. |
| Cancel or suspension of the project | Project stop | The project is sponsored by stakeholders and ABB gate model are being executed to ensure successful continuation of the project, until release. In the event the project is decided to be stopped by the sponsored stake holders, then the project termination information will be communicated to all dependent and service providers, two month in advance and project pre-closure notification will be triggered by concerned parties from ABB SCM team or Project Manager as applicable. |

# Dependencies

|  |  |  |
| --- | --- | --- |
| **ID** | **Dependency** | **Timeline** |
| DEP#1 | Availability of QA environment infrastructure | T0 |
| DEP#2 | Availability of required stakeholders for functional and technical walkthroughs | T0 |
| DEP#3 | Availability of test data for cloud platform/on-premise testing | T0 + 2 weeks |
| DEP#4 | Availability of documentation of all APIs, Models, Screens, Business APPs, access privileges to project repositories. | T0 |
| DEP#5 | Access to test environment for HCL members | T0 + 2 weeks |
| DEP#6 | Availability of Tools and Software licenses to be used for testing. | T0 + 2 weeks |
| DEP#7 | Dependency on ABB’s development team for Sanity check before code release to QA environment | Continuous |
| DEP#8 | Availability and access to Edge platform | T0 + 2 weeks |
| DEP#9 | Admin rights to change Platform services for performance testing | T0 + 2 weeks |
| DEP#10 | On schedule build release from ABB development team as per defined timelines | Continuous |
| DEP#11 | Device Simulator capable of generating large data for load and performance testing | T0 + 2 weeks |
| DEP#12 | System testing is dependent on timely release from development team | T0 + 20 weeks |
| DEP#13 | All the 5 business value applications will be released to VENDOR after sanity testing within 14 weeks from project start in following sequence: | 1. Business value App 1 – T0 + 4 weeks 2. Business value App 2 – T0 + 4 weeks 3. Business value App 3 – T0 + 8 weeks 4. Business value App 4 – T0 + 12 weeks 5. Business value App 5 – T0 + 14 weeks |

# Approval

**Version History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revision No | Date | Prepared / Modified by | Reviewed by | Significant Changes |
| 0.1 | 01-Feb-2021 | Ravi K Tanniru | HCL CoE | Refer Below Row |
| Section 3 - In scope and OOS should be mentioned here  Section 5 - 5.1 Here u should mention all the models maintained viz Excel templates  5.2 If we are doing any DB querying and query creation for DB validations same should be documented here  How are we connecting to DB ?  5.3 All the configuration info used, what is the values and toggles used. Details of all of them  Section 6 - 6.2.2 Is this the only properties required?  What about the browser detail,  Are we taking care of multi browser execution?  Why we are taking care of the DB details?  What is that we are using the DB for?  Section 6 - 6.2.2 Test Suite Configuration  Does a TC have multiple Dependency?  1) If yes same should be addressed  2) How about which feature is mapped to the TC Is that not relevant?  3) Also Execution type can be multi valued Means TC can run either nightly or as a smoke also  4) Did we look at a rerun strategy Many times after the first run,  testers want to rerun failed ones again later or one might want to re-run in the same run also  5) Also where we are taking care of the browser details  6) If I want to run a TC on 3 different browsers parallely how can I do it?  7) What are the different Module names we can have? Pls document all  8) To be Executed is good to have but how to run all from the suite level.  Meaning how to provide “All” with out TC level yes and no  Section 6 - 6.2.4.1 Page Object Repository  Better to have a column for Page also  So Excel object repo is what we will go ahead  there may be issue with Linux?  Look at some other format or DB  Section 6 - 6.2.4.2 Page Object Classes --- This has to iclude  Where are all the reusable keywords put?  Identify common functions  Also there is API validations also so how is this addressed?  Have common class or location for all the common business functions to be used in the TCs  Section 6 - 6.2.4.2.2  Maintain some structure for naming of the variables  a) btnHomePageAllApplncationIcons  ex control short cut+Page+Control name  Section 6 - 6.2.6  Identify different reports that will be generated  Identify any dashboards that framework will create  Section 6 - 6.2.6  Document what is the logging strategy  Section 7 - 7.2  Create a CI CD strategy and build strategy and how this will fit with the overall Dev pipelines etc | | | | |
| 0.2 | 03-Feb-2021 | Ravi K Tanniru | Srikanth Manikanti |  |
| Section 4 - Use Cases  Please add Acceptance Criteria for all the use cases.  4.2 As discussed, add new 5959 requirement for command line help, please add the same.  4.3 please mention which is default if no parameters provided. like default is sanity suite, env,..  4.28 if possible, have email reports to local run as well, since results will override in local so that we can have the logs to compare in future.  4.32 command line help should provide the details how to specify in command  4.39 identify which logging framework best fit for our framework and update. | | | | |
|  |  |  |  |  |
|  |  |  |  |  |

# References

<https://dev.azure.com/abbiatyiap/IA%20Digital%20-%20DIAS%20Platform/_wiki/wikis/Genix/655/Requirements>

Git commands are available at – <http://guides.beanstalkapp.com/version-control/common-git-commands.html>

Custom command - <https://mundrisoft.com/tech-bytes/protractor-script-execution-using-custom-command/>

Rest API - <https://medium.com/swlh/how-to-rest-api-a-tale-of-node-js-express-and-typescript-77bc598b280c>

Jenkins - <https://chercher.tech/protractor/jenkins-protractor>