|  |
| --- |
| **Test Strategy**  **SEAT\_RFQ\_HMI\_FPK\_Test\_V01** |

**Quest Global - HMI Experience**

At Quest Global we have more than a decade of experience working closely with a Global OEM in their evolution from a traditional automotive player with discrete HMI, to a fully digital cockpit that is on the path of transforming into a Software Defined Vehicle.

This close collaboration involves all elements of the entire HMI process, from supporting their UX Research team in rapid prototyping and evaluating software and hardware for suitability for the HMI Requirements, all the way through to developing UI/UX designs and assets for a given HMI, to developing re-usable models and state charts for the HMI transitions, and on to building RTOS and Android Applications, and Services. This support also includes integration and validation testing, along with code deployment at scale across multiple display sizes, shapes, themes, brands and display configurations.

Our journey in HMI began back in 2010 when we started consulting with Microsoft’s Connected Car Group in Redmond, USA. We were focused on making HMI that was effective and usable, creating prototypes and user testing them on a simulator. Then we were brought into this Global multi-brand OEM to help them revamp their HMI design to development processes. We combined our expertise in human factors with our technical depth in embedded systems and engineered a process that allowed them to go from design to prototype on target hardware within a week. This allowed them to usability test new concepts much more rapidly, and at a much lower cost, than ever before, greatly increasing both the quality and the usability of their interfaces.

From there we expanded into their digital cluster displays and responsive interfaces as our team scaled rapidly to over 1000-members across multiple functions and disciplines including design, development and QA. We follow Agile processes and have developed a fully automated CI/CD pipeline that handles multiple release-lines over multiple production programs across 12 worldwide brands, targeting over 70 display sizes. We also support integration testing with other vehicle systems including functional integration testing, interface testing, data flow testing, integration with external systems, and performance testing. In addition, we support in-vehicle testing for safety, performance, durability, and regulatory compliance before final release to production lines.

Our strong experience on Android has delivered over 30 different Android Apps, along with all the Realtime HMI apps for the digital Instrument Cluster, Fail Safe Monitor (ASIL-B) and the Camera. This work has also included Auto Park Assist, Climate Control and Soft switches.

Today over 50 million vehicles on the road run the HMI software we developed in close coordination with this Global OEM. These vehicles are available for sale around the world and the interfaces are localized into 36 different languages.

We firmly believe that this extensive experience in delivering a scalable HMI to a Global OEM for all of their brands is in line with the scope of this RFQ and we look forward to serving CARIADs current and future HMI needs.

Contents

[1. HMI FPK Test 4](#_Toc181808603)

[2. Scope and Objective 4](#_Toc181808604)

[3. Testing Approach/Workflow 5](#_Toc181808605)

[4. Testing Activities 6](#_Toc181808606)

[4.1. Approach for Manual Testing 6](#_Toc181808607)

[4.1.1. Quick Test (Sanity Test) 7](#_Toc181808608)

[4.1.2. FRP Test (Functional Test) 7](#_Toc181808609)

[4.1.2.1 Alert and Warning System Testing 8](#_Toc181808610)

[4.1.2.5 Performance and Responsiveness Testing 8](#_Toc181808611)

[4.1.3. Regression Test 9](#_Toc181808612)

[4.1.4. Language Test 10](#_Toc181808613)

[4.1.5. Free Testing (Ad-hoc) 10](#_Toc181808614)

[4.2. Approach for Test Automation 11](#_Toc181808615)

[4.2.1. Strategy for Automating UI/Layout Validation 11](#_Toc181808616)

[4.2.2. Strategy for Automating Functionality Validation 12](#_Toc181808617)

[4.2.3. Test Automation Tools and Framework 12](#_Toc181808618)

[4.2.4. Test Automated Deployment 13](#_Toc181808619)

[5. Test Environment 15](#_Toc181808620)

[6. Vehicle Testing & Management 15](#_Toc181808621)

[6.1. Vehicle Test Category 16](#_Toc181808622)

[7. Test Execution Strategy 16](#_Toc181808623)

[7.1. Entry Criteria: 16](#_Toc181808624)

[7.2. Exit Criteria: 16](#_Toc181808625)

[7.3. Suspension Criteria 17](#_Toc181808626)

[7.4. Test Cycles 17](#_Toc181808627)

[7.5. Severity Levels of Bugs 18](#_Toc181808628)

[8. Defect Management 18](#_Toc181808629)

[9. Constraints and Risks: 19](#_Toc181808630)

[10. Documentation and Metrics: 20](#_Toc181808631)

[10.1. Test Cycle - Artefacts 20](#_Toc181808632)

[10.2. Test Plan and Test Case Templates 20](#_Toc181808633)

[10.3. Process Defect Management 21](#_Toc181808634)

[10.4. Release Notes 21](#_Toc181808635)

[10.5. Metrics 22](#_Toc181808636)

[11. Test Reporting and Management 23](#_Toc181808637)

# HMI FPK Test

HMI Testing is a vital stage that every build triggered by a code change goes through. These builds must successfully pass all the levels of testing such as Sanity, Regulatory (Legal), Certification (Android Auto/CarPlay), Functional, Performance Testing on all the applicable Hardware Variants, Displays (Sizes/Shapes/Resolutions), and Display Configuration.

To support this work, we have built the required Automated Devops Pipelines and Test Automation frameworks. We have also used various test automation tools and developed a software Test Harness to execute this huge volume of work.

# Scope and Objective

Objective: To ensure the delivery of high-quality Work Package and Product that fulfils all the requirements of the customer, by infusing quality in all the areas of the development life cycle and by minimizing the manual effort through automation.

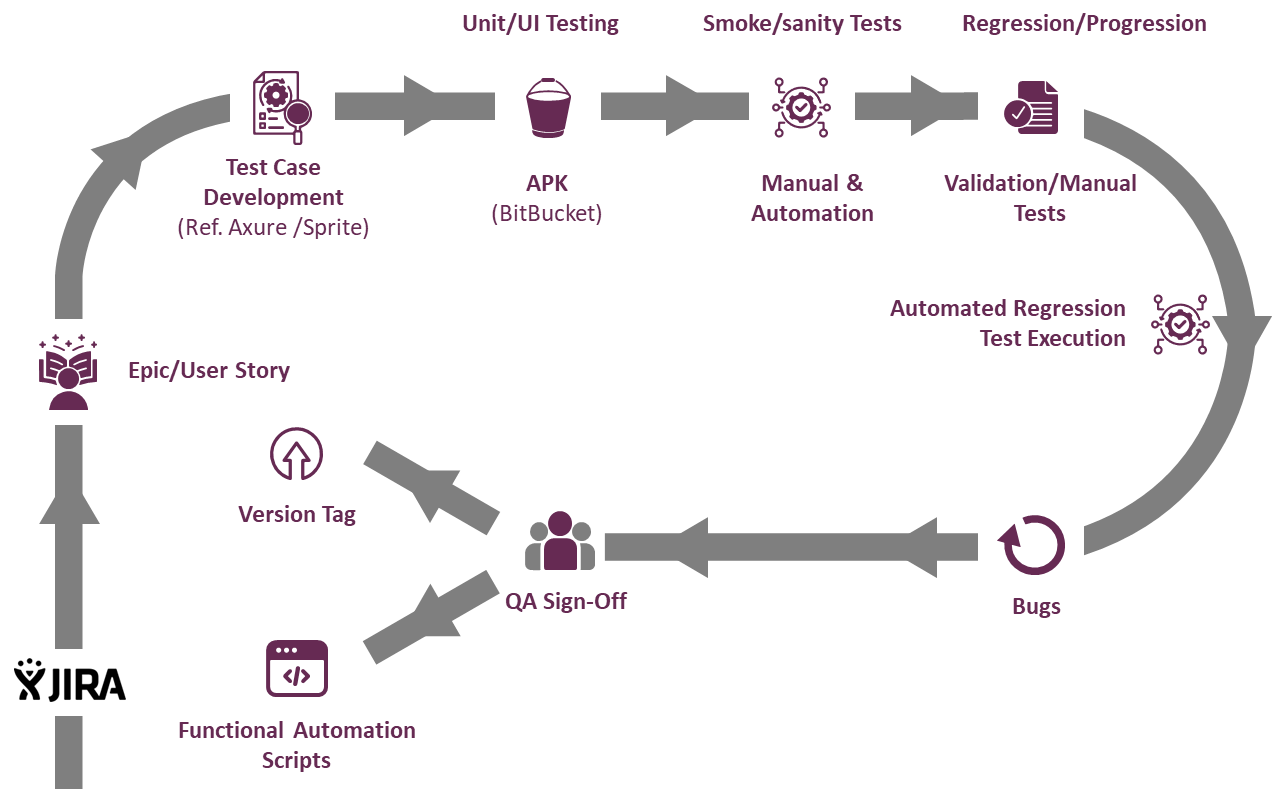
Manual Testing scope: Testing the ongoing Work Package releases for the modules mentioned in the document.

Automation Scope: The automation of UI and Functionality validation.

Variant Scope: Screens size variation.

# Testing Approach/Workflow

The following is the Testing workflow covering both Manual and Automation testing.



The above testing workflow explains various stages of software and system testing, with a focus on automation and continuous integration.

* **Epic/User Story Creation (JIRA)**:
* Requirements are documented/linked in JIRA as epics or user stories.
* Testing Scopes are well defined including the Test environment
* **Test Case Development**:
* Test cases are developed based on requirements.
* Test case will be stored in Test Rail & it must be linked to user story after review closure to maintain the traceability.
* **Functional Automation Scripts**:
* Automated scripts are created for core functionality testing.
* Allows repeated testing and faster execution of common test cases.
* **Unit/UI Testing:**
* verify individual components and are done at developer level test as apart of Unit Testing
* The application package (APK) is uploaded to BitBucket.
* Version control and distribution for consistent builds.
* **Manual & Automation Testing**:
* Smoke /Sanity will be done at first to confirm the build stability
* Ensures functionality aligns with requirements.
* **Regression/Progression Testing**:
* Checks for new issues introduced by changes (regression).
* Validates new features work as intended (progression).
* **Automated Regression Test Execution**:
* Following the completion of test case development, functional automation scripts are initiated. And when the Build is available, the same will be carried out.
* The automated regression test suite can also be triggered as part of Continuous Integration.
* Detects potential issues early in the development cycle.
* To check the issue Reproduced rate, multiple cycles can be run with automation for a certain Test case
* **Bugs**:
* Identified issues are logged, prioritized, and addressed by the Dev team.
* Bug fixes are integrated into subsequent builds.
* **QA Sign-Off**:
* Final approval from the QA Manager after all tests pass.
* Indicates the build is ready for release or further vehicle testing.

# Testing Activities

## Approach for Manual Testing

If a task is given to the test team, it will be closed after relevant work has been finished in accordance with the requirements and scope of the story.

Close the user story once all the Tasks are completed and the bugs mapped to the test cases are moved into a logical end state (Completed/Deferred/Rejected etc.)

The following are the Workflows –

A diagram of a workflow

Description automatically generated

The following testing activities are performed to validate the Software Quality.

### Quick Test (Sanity Test)

* Conducted to confirm the software build's stability
* Verify that the HMI display initializes correctly , all relevant CAN messages are present on the CAN bus.
* Check basic functionalities for Safety related tell tales, gauges, alerts, info pages, animations, and HUD for proper operation.
* Performed on daily builds and official builds using a bench setup or simulator.

### FRP Test (Functional Test)

* Functional validation is performed to validate each HMI requirement is correctly implemented in the software & system performs as expected from an end user prospect.

The following activities are performed as a part FRP testing.

##### 4.1.2.1 Alert and Warning System Testing**:** Validate that all alerts (e.g., Vehicle overspeed, seatbelt warning) behave precisely according to HMI requirements.

* + - Confirm that alerts display with correct timing, duration, and priority based on the urgency of the issue.
    - Test that alerts can be dismissed or acknowledged by the user when applicable and that high-priority alerts persist until resolved.
    - Ensure that alert icons, animations, and sounds match the specifications for visual

##### **4.1.2.2 Usability Testing:** To Ensure HMI elements are easy to interact with & clear visibility.

* + - Size & Position of the UI related to HMI
    - Touch functionalities to validate the different views and different gauges
    - Placements of any UI related

##### **4.1.2.3 Behavior of States and Modes:** To Ensure HMI behaviors are aligned with different vehicle states or driving modes

* + Validate the visuals & Gauges are displayed for different driving mode (Tour, Sport, Offroad etc.)
  + Validate the behavior of Gauges in different power mode (Off, Accessory, Run, Crank, Propulsion state)
  + To confirm HMI switches to night vision mode & revert to day light
  + To Validate the accurate display changes upon changing the Gear states (Park, Reverse, Neutral, Forward)

##### **4.1.2.4 Requirement Validation Testing:** To Verify that each specific HMI requirement is implemented correctly.

* + Confirm that all gauges (e.g., speedometer, fuel gauge, PRND Gauges) display accurate and real-time data, as specified in the requirements.
  + Ensure correct behavior for tell tales like headlights, turn signals, and hazard lights, appearing only under the right conditions.
  + Test customization features (e.g., theme selection, gauge style changes) to confirm that user preferences are saved and persist after restarts.
  + Ensure the correct behavior of Info-pages (Odometer, Trip Info, date & Time, energy efficiency)

##### 4.1.2.5 Performance and Responsiveness Testing**:** To Ensure the HMI performs efficiently, with quick response times for critical interactions and displays.

* + Measure the speed of gauge updates and alert displays to confirm that they respond to real-time data changes without delays.
  + Evaluate loading times for startup screens and welcome animations to confirm they meet requirements. (for ex: Screen boot up time is 9s, Welcome animation should display for 30s if there are no interruption)
  + Test the responsiveness of the HMI to driver inputs, such as steering wheel buttons or touch inputs without delay.

##### 4.1.2.6 **Regulatory Testing:**

As per the FMVSS (Federal Motor Vehicle Safety Standards) & CMVSS (Canada Motor Vehicle Safety Standards), the alerts should display within defined time frame.

For ex:

1. Collision Alert should display for 10s only
2. Speed Limit Set Alert should be there for 3s
3. Speed Brake warnings should be displayed with higher priority & clarity
4. Headlights and turn signal must be indicated by specific icons. And blue color icon should be for High beams & green for turn signals

### Regression Test

The objective of regression test is to ensure newly implemented functionalities or change in existing functions are not disrupting the existing functions.

Regression tests need to make sure whole functional requirements are validated with 100% of coverage with the requirement.

These are the common Test practices to have in the regression suite.

**Startup and Initialization:**

* Validate startup screens and welcome animations load exactly as per design, with no deviations in timing.
* Confirm that the HMI reaches the default display mode without unexpected changes or flickers.

**Gauges, Telltale and Views:**

* Verify that all gauges (e.g., speedometer, tachometer, fuel gauge) maintain their expected look and values.
* Check for correct behavior and styling of tell tales, including alignment, colors, and icons.
* Verify all the views, for ex: Gauge View, Map view, Driver Assist View, Clean view etc.

**Alert Displays:**

* Ensure that all Alerts (e.g., Driver drowsy, seatbelt) display exactly as per the reference, including position, color, and size.
* Confirm that high-priority alerts take the priority and there should not be any overlap
* To validate the low priority alerts are in the queue & should be displayed after disappearing the high priority alerts

**Mode Transitions and Environmental States:**

* Test transitions between day/night modes to confirm consistent colors, brightness, and theme changes.
* Simulate various driving modes (tour, sport) and confirm that all elements (gauge colors, background themes) display as defined in the requirements.

**Interactive Features:**

* Confirm that the HMI responds correctly to user inputs, such as adjusting settings or navigating menus.
* Validate that all selections and interactions reflect correctly on the display, with no lag, misalignments, or unexpected changes.

**RHMI Features:**

Confirm that the android auto and Car play features are working as per the requirement (for ex: Incoming calls, now playing, navigations, Bluetooth)

### Language Test

To ensure all text & labels displayed correctly in each supported language.

Followings are common practices to have in the test cases as a part of language testing

* Verify that all texts for alerts switch accurately to the selected language.
* Test for proper alignment and spacing of labels to ensure they fit within designated areas, regardless of language
* Check for any truncation or overflow of text in languages
* To verify the units and formatting based on locales of the selected language (ex: KMPH for Europe, MPH for US)
* To Confirm Date format, align with local conventions (e.g. MM/DD/YYYY for US, DD/MM/YYYY for Europe)
* To Verify that once a language is set, it persists after the vehicle is restarted or HMI is rebooted.
* RTL (Right to Left) Testing

### Free Testing (Ad-hoc)

Following tests are part of exploratory testing

* **Switching between pages**: To verify the switching between different info-pages rapidly to observe any elements from previous screen linger on the next screen
* **Rapid acceleration and Deceleration**: To verify speedometer & Tachometer update in real time without any lag
* **Extreme Values of Gauges**: Validate the minimum and maximum value for different temperatures, battery levels to make sure there is no unusual behavior at these limits
* **Gauge Fluctuations**: Simulate different gauges (speedometer, Fuel level) rapidly to observe if the gauges update smoothly.
* **Simulation of multiple Alerts**: Trigger multiple critical alerts simultaneously and observe the alert display priority ensuring high priority alerts are taken the top priority .
* **Dismissible alerts**: To verify alerts get disappeared after dismissing them in mid-way
* **Language and Localization**: Quickly switch between different languages and observer no truncations or alignment issue.
* **Unsupported Language**: To verify the unsupported language & check if the HMI shows the default language (English)
* **Driving Mode Transition**: Change the driving mode rapidly & observe HMI for smooth transition in layout & Gauge updates.

## Approach for Test Automation

### Strategy for Automating UI/Layout Validation

The following are the steps followed for the automation of UI Validation –

1. Identifying the automation candidates
2. Identifying the suitable tools and framework
3. Identifying the resources with suitable skill set
4. Training on tools /technology as needed
5. Development/Testing/Debugging of the automation test
6. Automation Test is added to the automation test repository

The following is the list of features that are considered for UI/layout validations –

1. The number of objects/element (assets)
2. The shape of objects/element regardless of screen size/resolution
3. The position of the elements
4. The colour of the objects/elements (before and after the events)
5. Overlapping of the elements
6. The clarify of the font and its readability
7. The text and the colours of the error/warning messages
8. The clarity of the images
9. Spellings
10. Working of the scrollbars according to the size of the page
11. Disabled/greyed-out fields
12. Colour of the hyperlink (if any)
13. Testing the change in status of the Property (of an element/widget) based on the action (click, select, check etc.,) performed on it
14. Testing if the intended purpose of the individual element is met for the elements such as buttons, links, menus, dropdown lists, checkboxes, radio buttons etc.,

### Strategy for Automating Functionality Validation

The following are the steps followed in the automation approach for Functionality Testing –

1. Prepare the test cases for the entire module based on the Specification documents
2. Identify the automation candidates from among the total test cases
3. Develop automation scripts for the identified automation cases
4. Identify the resources with suitable skill set
5. Training on tools/technology as needed
6. Development/Testing/Debugging of the automation test
7. Automation Test is added to the automation test repository.

We have about the following amount of test cases, which are based on our expertise with a global OEM client.

|  |  |
| --- | --- |
| Module | Total no. of Test Cases Identified (Approx) |
| Gauges | 1200 |
| Info Pages | 1100 |
| Core | 700 |
| RHMI | 250 |
| Tell Tales | 2500 |
| Alerts | 25000 |
| Hud | 500 |

### Test Automation Tools and Framework

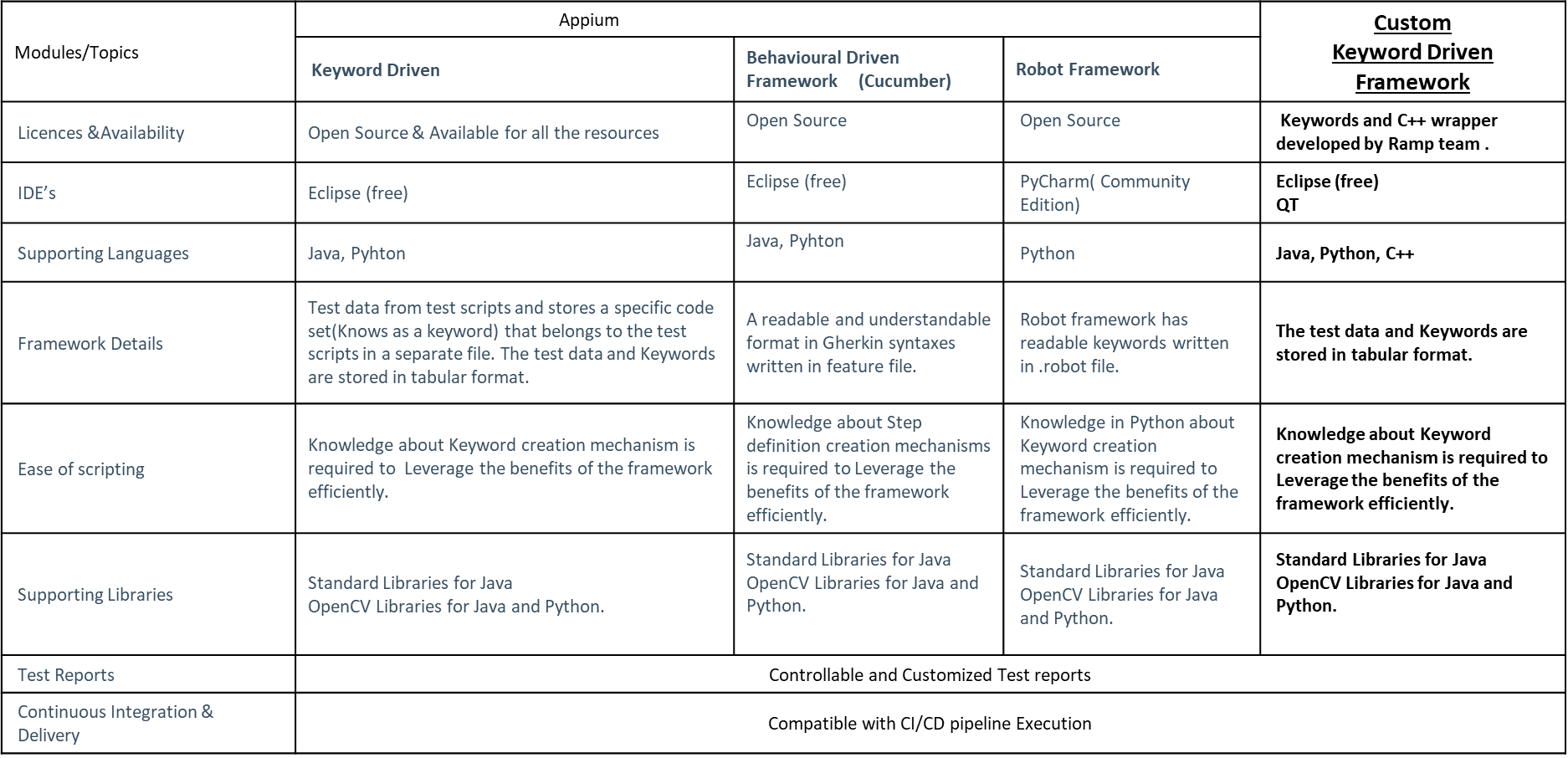
The test automation framework contains the following tools/technology - Cucumber/BDD, Espresso APIs, Java, Android Studio, Gradle, VS Code, Junit for Android Platform. Python, Robot, Custom Keyword driven Frameworks for Cluster / QNX Platform.

Following Tools & Frameworks are being used to do the Test Automation.

*Tools : Selenium, Sikuli, Appium, API Testing, TestNG , Extent Reports, JMeter,* VS Code

Test Frameworks: *Keyword Driven, Hybrid, Page Object Model, Robot*

* *Keyword Driven framework, where we have developed keywords (functions) to take actions on the widgets, were Appium server been used to interact with test devices or h/w.*
* *Behavioural Driven Framework (Cucumber), the test cases steps been written in plain English, to act on widgets keyword (functions) been used, again Appium server been used to interact with test devices or h/w.*
* *CAT Robot Framework, it is key word driven framework with python to automate the Cluster Display as well as Android applications with screenshot validations.*

**

### Test Automated Deployment

#### **4.2.4.1 Objective and Scope**

* **Objective**: Automate deployment and testing on automotive clusters, RTOS, and HMI.
* **Scope**: This automated deployment and testing setup is intended for the automotive cluster and RTOS model, specifically for the regression test suite. Testing includes.
  + **Functional Testing**: Conducted through screen comparison and widget validation.
  + **Communication Protocols**: CAN and LIN protocols are used to send signals via the GM Vehicle Simulator.

#### **4.2.4.2 Environment Setup**

* Refer section 5

#### **4.2.4.3 Jenkins Configurations**

* **Node Creation**: Start by creating a node for the systems.
* **Job Creation**: Set up a job using a freestyle pipeline in Jenkins.
* **Job Configuration**: Configure the job and schedule it is using the task scheduler.
* Scripting:
  + Use Groovy scripts in Jenkins to define and automate tasks.
  + The scripts will leverage the task scheduler to execute multiple stages in the process.

#### **4.2.4.4 Build & Deployment Process**

* A Jenkins job will trigger a task scheduler that manages all related stages
  + Stage One
* Pull the build from the repository.
* Launch the VIP application.
* Rename required files (e.g., Contains.txt).
* Complete the VIP process, concluding Stage One.
* Stage Two
  + Execute QFil tasks.
  + Upon completion, proceed to Stage Three.
* Stage Three
  + Perform tasks using Tera Term.
  + After Tera Term tasks are complete, move to Stage Four.
* Stage Four
* Launch automation scripts to execute test scenarios.
* Once all scenarios are completed, Jenkins will compile and display results on the current job's dashboard.
* **Additional Features:**
  + **Status Monitoring with Light Bulb Indicators**
  + **Red**: Indicates a stage failure.
  + **Amber**: Shows a stage in progress.
  + **Green**: Confirms all stages have passed successfully.

#### **4.2.4.5 Test Execution on Hardware**

* **Functional Testing**:
  + **UI Tests**: Validate user interface elements and interactions.
  + **Signal/CAN Message Validations**: Ensure accuracy and integrity of CAN messages and signal processing.
* **Stage Five**:
  + Automation scripts are launched to execute all test scenarios.
  + After completing the tests, Jenkins compiles the results and displays them on the current job's dashboard.

#### **4.2.4.6 Results Collection and Reporting**

* **Result Reporting on Jenkins Dashboard**:
  + To view test results on the Jenkins dashboard, install the **Robot Framework plugin**.
  + At the end of Stage Four, after all test cases are completed, the plugin will display the pass/fail status of each test case.
* **Result Notifications**:
  + All test results are displayed in HTML format on Jenkins.
  + Results are also sent to the appropriate email IDs for notifications.

#### **4.2.4.7 Error Handling and Notifications**

* Stage Failure Indicator
  + If a stage fails, the light bulb turns red, signalling that a failure has occurred.
  + This provides immediate visibility, allowing team members to investigate the issue.

#### **4.2.4.8 Maintenance and Troubleshooting**

* **Live Logs and Error Monitoring**:
  + All test case logs are available in real-time on the Jenkins console.
  + In case of any human or system errors, they can be monitored directly from Jenkins and addressed promptly.

# Test Environment

* **Host Machine Software**:
  + Tera Term
  + QNX
  + Emulator
  + Flashing Tools
  + QFil
  + VIP
  + Cancase
* Hardware
  + Physical Display
  + ECU (Electronic Control Unit)
  + VCU (Virtual Cockpit Unit)
  + Debugger
  + Harness and display cables
  + Display
* CI/CD Setup
  + Jenkins setup for automated deployment and testing

# Vehicle Testing & Management

The following activities need to be conducted before being released to the customer.

## Vehicle Testing Phases

Objective is to ensure cluster system performs accurately in real time including responsiveness, and durability of the cluster over time and usage.

### HMI Functional Testing

To validate all the cluster functions, operate as per the requirement & providing the accurate & timely information in the display

For ex:

* Turn Signal Indicator test for both left & right side
* Set the Cruise to 120 km/h & verify the set speed is showing correctly on cluster
* Lane keep/Centering Assist warning test
* To Verify all cluster display elements such as Temp, RPM, Fuel Level , Alerts etc.

### HMI Integration Testing (Cluster + Infotainment)

To verify Cross-device testing of HMI concepts in the vehicle related to controls with display connection (head unit, instrument cluster, head-up display, passenger display).

For Ex:

* Information displayed on the cluster and HUD (if equipped) should synchronized with no delay
* Time, Language should be in sync between CSM & IPC display
* Info pages should show correctly among CSM & IPC Screen
* Ambient temp must be same both in IPC & CSM
* Ensure Volume Control, Navigation data, ADA warnings are in Synchronized across cluster & infotainment system

### HMI Performance Test

To verify Responsiveness and robustness of the system on Vehicle

For example:

* Seat Belt warning should get activated within 1s if seat belt buckled
* cluster should display real-time trip information, such as average speed, fuel consumption, and elapsed time
* Real-time updates on battery health status and charging percentage should be displayed, with charging updates every 5 seconds
* Drive mode changes (ex: Sport to Eco etc) should reflect in cluster
* To validate the system stability during multiple simultaneous alerts.
* To validate the system responsiveness upon updation of real time elements (e.g. RPM, Odometer data) and alerts (e.g. Lane Departure)

### Mileage and Durability Testing

Verify the long-term reliability of the cluster ensuring it displays correct data with extensive usage & under different environment conditions.

* Monitor Navigations & indicators during long term use
* Verify the behavior of Alerts & screen readability
* Cover at least 18000 KM with C-Sample VCU under winter test & hot trip test

## Reporting and Issue Tracking

Vehicle Test report should contain the following.

* List of Art tested in the specific release
* Test Results
* Result/Status of each Art
* Impact on the functionality in case any defect found
* JIRA Ticket information for both Art, Testing
* Summarize findings, any unresolved issues, final recommendations
* Creation and maintenance of test catalogue, validation materials
* Maintenance of Test Racks
* Planning the Test Drives
* Participation in Vehicle management meetings
* Update the Project status, VV Report, risks.

For more details, please Refer section 11.

## Tools and Resources

The following are the Tool & resources details for Vehicle testing.

**Diagnostic and Analysis Tools**: Data Loggers, Vector Canoe, Diagnostic Software (Scan tool), Tera Term

**Vehicle Equipment**: Test Vehicles with Latest Cluster & Infotainment Software

**Resources**: Test Engineers, test Vehicle, Trained drivers for mileage test,

# Test Execution Strategy

## Entry Criteria:

The following Entry Criteria refer to the desirable conditions to initiate test execution.

* User Stories are available in JIRA
* Develop Test Plan, test conditions, test cases, expected results and execution scripts
* Availability of Test Environment – JIRA; Hardware and Tabs
* The Test Environment can be Windows Emulator, Hardware Bench Setup, Vehicle

## Exit Criteria:

Below are the exit criteria that need to be met in order proceed with the implementation.

* + 100% Test Scripts executed
  + 95% pass rate of Test Scripts
  + No open Critical and High severity defects
  + 95% of medium severity defects have been closed
  + All remaining defects are either cancelled or documented as User Stories in the backlog
  + All expected and actual results are captured and documented well in JIRA
  + All test metrics are collected
  + All defects are logged

## Suspension Criteria

Suspension criteria specify the criteria to be used to suspend all or a portion of the testing activities while resumption criteria specify when testing can resume after it has been suspended.

|  |  |
| --- | --- |
| Suspension | Resumption |
| When we receive the release kit without updated files, updated release document and unit test case document. | When the release kit is provided with proper files, documents and unit test cases |
| The build contains many serious defects, which seriously prevent or limit testing to proceed further (ex: When User is not able to login the application) | All the fixes are successfully implemented, and the Testing Team is notified to continue testing. |
| Unavailability of external dependent systems during execution | When the external dependent systems become available again |
| Unavailability of dependent modules | When the dependent modules are ready |
| Due to environmental problems (Network Unavailability, Software Unavailability) | When the environmental issues are resolved |
| Due to valid customer changes | After all the customer changes are freezes and changed software is released to testing |
| Assigned test resources are not available when needed by the test team | After the test resource is available |
| When the release is given without increased version numbers | When the release is given with increased version numbers |

Note: Entry, Exit and Suspension criteria are flexible benchmarks. If they are not met, the test team will assess the risk, identify mitigation actions and provide a recommendation. All this is input to the project manager for a final “Go-No Go” decision.

## Test Cycles

* There will be two cycles for functional testing. Each cycle will execute all the scripts.
* The objective of the first cycle is to identify any blocking, critical defects, and most of the high defects.
* The objective of the second cycle is to identify remaining high and medium defects and to remove the workarounds (if any) from the first cycle.

## Severity Levels of Bugs

Defects found during the Testing will be categorized according to following categories:

|  |  |
| --- | --- |
| **Severity** | **Impact** |
| Highest/Critical | 1. This bug is critical enough to crash the system, cause file corruption, or cause potential data loss 2. It causes an abnormal return to the operating system (crash or a system failure message appears).  * It causes the application to hang and requires re-booting the system. |
| High | 1. It causes a lack of vital program functionality with workaround. |
| Medium | * This Bug will degrade the quality of the System. However, there is an intelligent workaround for achieving the desired functionality - for example through another screen. * This bug prevents other areas of the product from being tested. However other areas can be independently tested. |
| Low | * There is an insufficient or unclear error message, which has minimum impact on product use. |
| None | * There is an insufficient or unclear error message that has no impact on product use. |

# Defect Management

Error/Defect management includes the following activities.

* Preliminary analysis of Defects & raising the Ticket in JIRA System
* Verifying the error reproducibility rate & provide the log for the same.
* Synchronization of error management system between JIRA, Test Rail
* Creation of Error Statistics & Reports
* Tracking of the Defect, including Due Dates, fix version etc.

The following chart explains the defect management process.

A diagram of a bug workflow

Description automatically generated

* **Assigned Ticket Management**: JIRA tickets assigned to the testing team are managed promptly, with a maximum resolution time of **3 days** for effective issue handling.
* **Resolved Ticket Verification**: Verification of tickets marked as “TO BE TESTED” is conducted within **3 days** to confirm that fixes are functional and do not introduce regressions.
* **Critical Issue Analysis**: Any critical issues are analyzed within **24 hours** to provide a quick resolution pathway and minimize impact on subsequent testing cycles.

# Constraints and Risks:

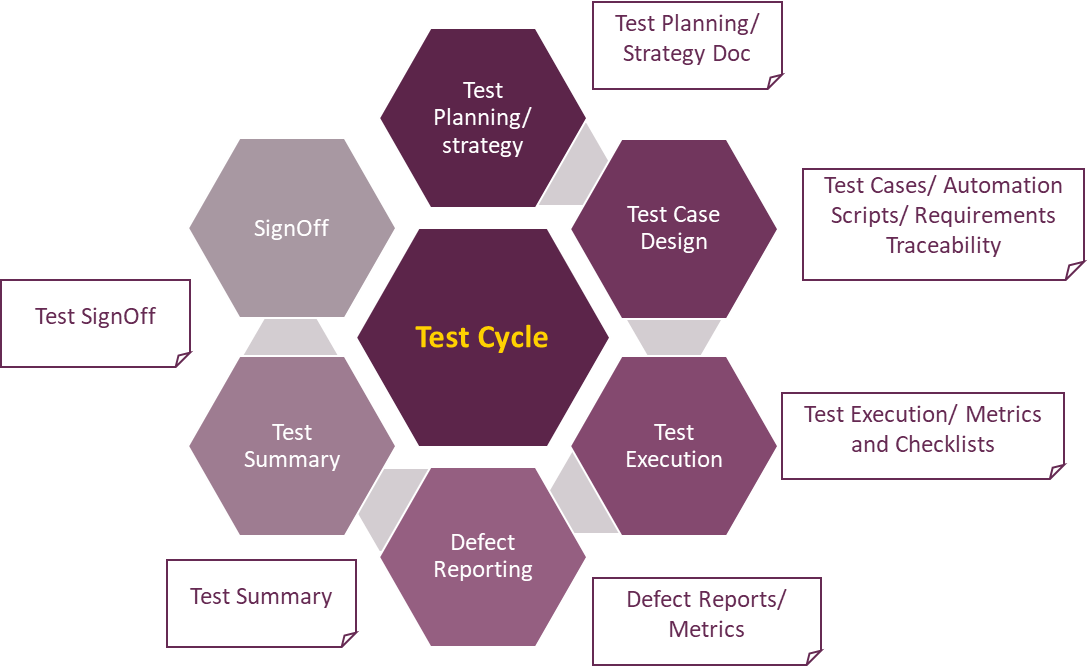
System Integration Testing for all the modules is not being performed.

The scarcity of the following resources would have adverse impact on Testing –

* Hardware
* Required System configurations.
* Tabs/Devices
* Technically Skilled testers

# Documentation and Metrics:

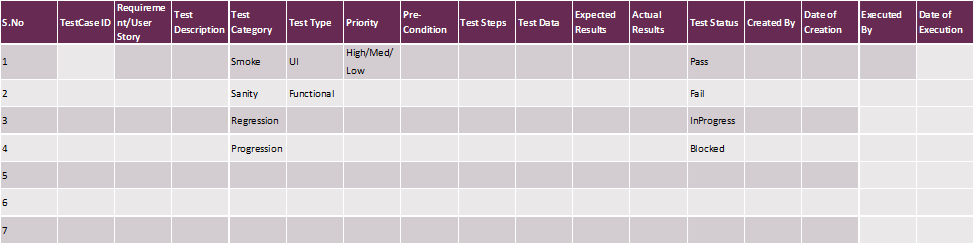
## Test Cycle - Artefacts



## Test Plan and Test Case Templates

Test Plan Template - 

The following is the template for developing Test Cases

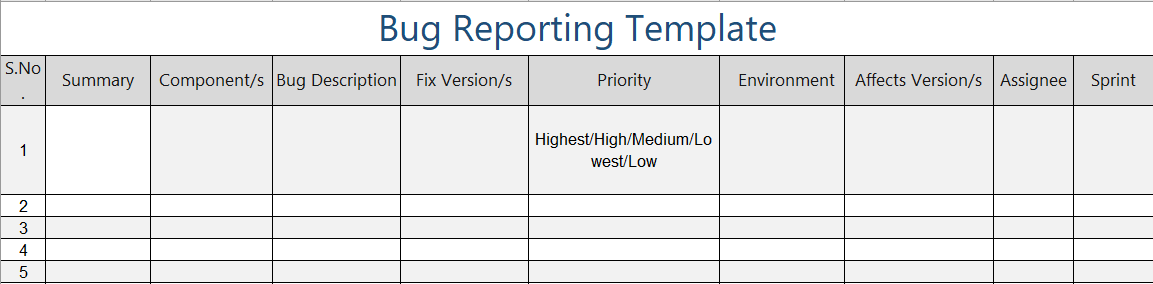


Identification of Test Case Category -

|  |  |
| --- | --- |
| **Test Case Category** | **Identification Criteria** |
| Smoke | Critical functionality |
| Sanity | Core features of new functionality |
| Regression Testing | The impacted on the existing functionality due to development of new features or due to the bug fixes |
| Performance testing | Responsiveness and robustness of the system |
| Stress Testing | Ensure all apps robust and maintaining performance under extreme conditions |
| Load Testing | Key user scenarios, system behaviour under expected load conditions in a vehicle |
| Progression Testing | The detailed functionality of the features that are identified to be released in a sprint. |

## Process Defect Management

The following is the bug reporting template.



## Release Notes

Project release notes:

This release notes provides a comprehensive summary of the changes implemented, ensuring that the entire project team, including management, testers, developers, and designers, is aligned and informed. Additionally, it demonstrates how the development has influenced user interactions with the system, helping clients and end users understand the impact of these changes.

Release Notes should contain the following information:

A Build Identifier: A build identifier helps a person keep track of what changes occurred in which release of the product.

Installation Instructions: Instructions for installation can be in the release notes if they are short and need to be done for the version that is being released.

Advice for Testers: An advice of caution that allows the testers to improve their testing efforts. For example, the change made to feature ‘A’ could have a ripple effect through other features.

New Functionality: Information on new features or functionality that is added in the current release.

Removed Functionality: Detail the features or functionality that is taken out from this release.

Implementation Notes: Explains any features that are implemented in a way that differs from the requirements.

Fixes: Discuss any errors or bugs that have been fixed in this release and their ticket number.

Outstanding Issues: Include features that are not fully implemented or that are released with known errors. Document workarounds for the outstanding issues as applicable.

## Metrics

The following are some of the metrics that can be captured for each sprint:

* Test Execution Progress (Planned vs. Actual Test Case Execution)
* Module wise defect trend (no. of Critical, High, Medium, Low defects)
* Defects Opened vs. Defects Resolved

Total Number of Defects

* Defect Density = --------------------------------- \* 100

Total Number of Test Cases

For a quality product Defect Density should be aimed less than or equal to 1%

Number of defects Resolved

* Test Efficiency = ---------------------------------------- \* 100

Total no. of Defects Submitted

Total no. of Defects opened in Test Cycle

* Test Effectiveness = ------------------------------------------------------- \* 100

Total Defects Count (Defects Opened in Test Cycle +

Defects opened after Release)

# Test Reporting and Management

The following information needs to be reported to SEAT & VW Group .

* For each software release, the HMI Validation report shall be shared at the end of validation phase
* Any Critical error with high severity shall be presented
* Regular synchronization sessions will be conducted with Testing, Integration team to ensure consistent alignment on requirement, issue resolution & testing priorities
* All testing process, tools and methodologies are documented to maintain transparency, facilitate knowledge sharing

|  |  |  |
| --- | --- | --- |
| **Report** | **Description** | **Frequency** |
| Test preparation | Report the status of Test case authoring – In-progress/Complete | Weekly / Daily (optional) |
| Daily Execution Status | To report on Pass, Fail, Total Bugs, Highlight Showstopper/Critical Defects | Daily |
| Weekly Status Report (or) Sync-up Call | Weekly Status Report to be shared with all the stakeholders to inform the overall Testing status, impediments, to do follow-ups, to discuss process changes etc. | Weekly |