**Introduction to Espresso**

Espresso is a testing framework for Android that makes it easy to write reliable user interface tests directly within Android Studio. It is part of the Android Testing Support Library and provides APIs to simulate user interactions and write functional UI tests. Espresso automatically synchronizes test actions with the user interface of the application, ensuring that the test environment reflects the current state of the app.

**Espresso Testing Methods and Principles**

Testing Methods:

1. Write Concise and Effective Tests: Espresso allows developers to write concise and straightforward tests, focusing on what you want to test rather than how to test it.

2. Use of ViewMatchers, ViewActions, and ViewAssertions: Espresso works with these three main components to match views, perform actions on those views, and verify the views' state.

3. Synchronization: Espresso provides automatic synchronization between the test and the application UI, waiting for all background tasks to finish before executing the next test instruction.

**Testing Principles:**

1. Deterministic: Tests written using Espresso are deterministic, meaning they should produce the same results every time.

2. Flakiness Minimization: By automatically synchronizing with the app's UI thread, Espresso minimizes flakiness, making the tests more reliable.

3. Scalability: Espresso tests can be run on real devices or emulators and are easily scalable, allowing for efficient execution of a large test suite.

**Additional Considerations for Espresso Testing Protocol**

- Integration with Other Tools: Espresso can be integrated with tools like JUnit for unit testing and supports continuous integration systems.

- Thorough Documentation and Community Support: There is a vast array of documentation available for Espresso, and an active community of developers provides support and best practices.

- Consider Target Devices and Versions: When writing Espresso tests, considering various screen sizes, orientations, and Android versions ensures that the application behaves as expected across different scenarios.

- Focus on User Experience: Espresso tests should ideally mimic user interactions and behavior, providing a valuable feedback loop for development, ensuring that the app's user experience is smooth and intuitive.

**Test Cases**

**Test Case 1: BradyParameter Test1**

**Purpose and Scope:**

This test is aimed to validate the program and retrieve button functioning in the Brady page on an Android programmer, and also check the BradyMode behavior. It ensures accurate data transmission by checking that the retrieved values remain consistent after programming, and verifies the disabling logic of 4 buttons under VVIR Mode.

**Reference Documents:**

- Android Programmer Specification

- BLE Version Requirements: 1.0.21

**Preconditions:**

- Navigate to the Tachy Detection page, select 1-zone mode, and set the VF rate to 205 to ensure no maxSensorRate overlap with Tachy during the Brady test.

**Test Environment and Setup:**

- Android programmer

- Version 1.0.21

**Test Object:**

- Radiogroup elements on the Brady page

- Program and retrieve buttons

- BradyMode and corresponding 4 button options

**Test Steps:**

1. Click each Radiogroup on the Brady page.

2. Randomly select a clickable value within each Radiogroup.

3. Check BradyMode, if not VVIR, verify that 4 button options are disabled.

4. Click the program button.

5. Click the retrieve button to verify that the values on the page remain consistent after retrieval.

**Expected Output:**

- After retrieval, all values should be consistent with what was programmed.

- If Mode is not VVIR, 4 button options under VVIR should be disabled.

- If any inconsistency or wrong disabling logic is found, Espresso will return an Assertion Failure, and the test will be marked as failed, moving to the next test.

**Risks and Dependencies:**

- Potential inconsistency between program and retrieve actions

- Incorrect disabling logic for the 4 button options under VVIR Mode

**Test Case 2: BradyParameter Test2**

**Purpose and Scope:**

This test is designed to validate the program and retrieve button functioning on the Brady page of an Android programmer. It checks the BradyMode behavior, the disabling logic of 4 buttons under VVIR Mode, and the pop-up and button disabling logic during an overlap situation between Brady and Tachy.

**Reference Documents:**

- Android Programmer Specification

- BLE Version Requirements: 1.0.21

**Preconditions:**

- Navigate to the Tachy Detection page, select 1-zone mode, and set the VF rate to 120. This is intentionally set to test the overlap situation between Brady and Tachy and validate the corresponding pop-up and button disabling logic.

**Test Environment and Setup:**

- Android programmer

- Version 1.0.21

**Test Object:**

- Radiogroup elements on the Brady page

- Program and retrieve buttons

- BradyMode and corresponding 4 button options

- Overlap situation between Brady and Tachy

**Test Steps:**

1. Click each Radiogroup on the Brady page.

2. Randomly select a clickable value within each Radiogroup.

3. Check BradyMode, if not VVIR, verify that 4 button options are disabled.

4. Click the program button.

5. Click the retrieve button to verify that the values on the page remain consistent after retrieval.

6. In case of overlap between Brady and Tachy, verify the pop-up warning and button disabling logic.

**Expected Output:**

- After retrieval, all values should be consistent with what was programmed.

- If Mode is not VVIR, 4 button options under VVIR should be disabled.

- Correct pop-up warning and button disabling logic during overlap between Brady and Tachy.

- If any inconsistency or incorrect logic is found, Espresso will return an Assertion Failure, and the test will be marked as failed.

**Risks and Dependencies:**

- Potential inconsistency between program and retrieve actions

- Incorrect disabling logic for the 4 button options under VVIR Mode

- Incorrect pop-up or button disabling logic during overlap situation

**Review and Approval:**

- [To be completed by relevant stakeholders]

**Version Control:**

- SourceTree, version tracking as per organizational guidelines

**Attachments and Appendices:**

- [Any additional diagrams, charts, or visual aids]

**Test Case 3: BradyParameter Test3**

**Purpose and Scope:**

This test focuses on the interaction between LowerRateLimit and MaxSensorRate within the BradyMode set to VVIR on an Android programmer. It is designed to validate the button disabling and pop-up warning logic when LowerRateLimit is greater than or equal to MaxSensorRate. The test also verifies that the retrieved values remain consistent after programming.

**Reference Documents:**

- Android Programmer Specification

- Version Requirements:

- Android Programmer: 1.0.21

- Firmware: 1.0.21

- BLE: 1.0.20

**Preconditions:**

- Navigate to the Tachy Detection page, select 1-zone mode, and set the VF rate to 205. VF rate overlap is not considered in this test case.

**Test Environment and Setup:**

- Android programmer

- Version: 1.0.21

**Test Object:**

- BradyMode (set to VVIR)

- LowerRateLimit and MaxSensorRate selection

- Program and retrieve buttons

- Button disabling and pop-up warning logic

**Test Steps:**

1. Set BradyMode to VVIR.

2. Randomly select available options for LowerRateLimit and MaxSensorRate.

3. Check if the button is disabled and the pop-up warning logic when LowerRateLimit is greater than or equal to MaxSensorRate.

4. Click the program button.

5. Click the retrieve button to verify that the values on the page remain consistent after retrieval.

**Expected Output:**

- Proper button disabling and pop-up warning when LowerRateLimit is greater than or equal to MaxSensorRate.

- After retrieval, all values should be consistent with what was programmed.

- If any incorrect logic or inconsistency is found, the test will be marked as failed.

**Risks and Dependencies:**

- Incorrect interaction between LowerRateLimit and MaxSensorRate

- Potential inconsistency between program and retrieve actions

Test Case 4: BradyParameter Test 4

**Purpose and Scope:**

This test focuses on the complex interactions between Tachy Detection, LowerRateLimit, and MaxSensorRate within the BradyMode set to VVIR on an Android programmer. The main goal of this test is to validate the disabling logic of buttons and the warning alerts considering all three parameters together. Additionally, it ensures the retrieved values remain consistent after programming.

**Reference Documents:**

- Android Programmer Specification

- Version Requirements:

- Android Programmer: 1.0.21

- Firmware: 1.0.21

- BLE: 1.0.20

**Preconditions:**

- Navigate to the Tachy Detection page, select 1-zone mode, and set the VF rate to 120. This test case will take into account the collective effects of Tachy Detection, LowerRateLimit, and MaxSensorRate.

**Test Environment and Setup:**

- Android programmer

- Version: 1.0.21

**Test Object:**

- BradyMode (set to VVIR)

- LowerRateLimit, MaxSensorRate, and Tachy Detection selection

- Program and retrieve buttons

- Button disabling and pop-up warning logic

**Test Steps:**

1. Set BradyMode to VVIR.

2. Randomly select available options for LowerRateLimit and MaxSensorRate.

3. Check the interplay between Tachy Detection, LowerRateLimit, and MaxSensorRate to test button disabling and warning alerts.

4. Click the program button.

5. Click the retrieve button to verify that the values on the page remain consistent after retrieval.

**Expected Output:**

- Proper button disabling and pop-up warnings considering the collective effects of Tachy Detection, LowerRateLimit, and MaxSensorRate.

- After retrieval, all values should be consistent with what was programmed.

- If any incorrect logic or inconsistency is found, the test will be marked as failed.

**Risks and Dependencies:**

- Incorrect interaction between Tachy Detection, LowerRateLimit, and MaxSensorRate.

- Potential inconsistency between program and retrieve actions.

**Review and Approval:**

- [To be completed by relevant stakeholders]

**Version Control:**

- SourceTree, version tracking as per organizational guidelines

- Firmware version: 1.0.21

- BLE version: 1.0.20

**Attachments and Appendices:**

- [Any additional diagrams, charts, or visual aids]

Test Case 5: TachyDetectionTest1

**Purpose and Scope:**

The test aims to verify the Input/Output functionality of all buttons in the 1-zone mode within the Tachy Detection page. It ensures proper interactions, programming, and retrieval for each button and value.

Reference Documents:

- None

Test Environment and Setup:

- BLE: 1.0.20

- Firmware: 1.0.21

- Software: 1.0.21

Preconditions:

- Automatic Template Acquisition must be in the "off" state.

Test Object:

- Tachy Detection interface, including all parameter settings.

- Program and retrieve buttons within the 1-zone mode.

Test Steps:

1. Navigate to the Tachy Detection page.

2. Select 1-zone mode.

3. Individually access each button within the mode, randomly selecting a value.

4. Click the program button to apply the settings.

5. Click the retrieve button to validate the programmed data.

6. Verify that all values and settings are consistent with what was programmed and retrieved.

Expected Output:

- All buttons in the 1-zone mode on the Tachy Detection page should respond appropriately.

- Programmed values should be exactly retrieved, and any inconsistencies will result in a failed test.

Risks and Dependencies:

- The functionality of the program and retrieve buttons.

- The correctness of the data transmission process.

- Interaction between different settings and values.

Review and Approval:

- [To be completed by relevant stakeholders]

Version Control:

- SourceTree, version tracking as per organizational guidelines

- BLE version: 1.0.20

- Firmware version: 1.0.21

- Software version: 1.0.21

Attachments and Appendices:

- [Any additional diagrams, charts, or visual aids]

Test Case 6: TachyDetectionTest2

Purpose and Scope:

The test aims to verify the Input/Output functionality of all buttons in the 2-zone mode within the Tachy Detection page. The test will also validate the logical relationship between the VF parameters and FVT parameters to ensure consistency and correctness.

Reference Documents:

- None

Test Environment and Setup:

- BLE: 1.0.20

- Firmware: 1.0.21

- Software: 1.0.21

Preconditions:

- Automatic Template Acquisition must be in the "off" state.

Test Object:

- Tachy Detection interface, including all parameter settings in 2-zone mode.

- Program and retrieve buttons within the 2-zone mode.

Test Steps:

1. Navigate to the Tachy Detection page.

2. Select 2-zone mode.

3. Individually and randomly access each parameter setting, randomly selecting and confirming a value.

4. Evaluate logical judgments among all parameters; check for any red bubbles indicating parameter conflicts.

5. Verify the program button is disabled when conflicts are detected.

6. If no conflicts are present, click the program button to apply the settings, followed by the retrieve button to validate the data integrity.

7. Continue from Step 3.

Expected Output:

- All buttons in the 2-zone mode on the Tachy Detection page should respond appropriately.

- Programmed values should be exactly retrieved, and any inconsistencies or conflicts should result in the program button being disabled and a failed test.

Risks and Dependencies:

- The functionality of the program and retrieve buttons.

- The logical relationship between VF parameters and FVT parameters.

- Interaction between different settings and values, especially conflicts detection.

Review and Approval:

- [To be completed by relevant stakeholders]

Version Control:

- SourceTree, version tracking as per organizational guidelines

- BLE version: 1.0.20

- Firmware version: 1.0.21

- Software version: 1.0.21

Attachments and Appendices:

- [Any additional diagrams, charts, or visual aids]

Test Case 7: TachyDetectionTest3

Purpose and Scope:

The test is designed to validate the I/O functionality of buttons within the Tachy Detection page's 3-zone mode and to verify the logical relationship among VF, FVT, and VT parameters. The test ensures that the parameters' interactions are in line with expected behavior.

Reference Documents:

- None

Test Environment and Setup:

- BLE: 1.0.20

- Firmware: 1.0.21

- Software: 1.0.21

Preconditions:

- Automatic Template Acquisition must be in the "off" state.

Test Object:

- Tachy Detection interface, including all parameter settings in 3-zone mode.

- Program and retrieve buttons within the 3-zone mode.

Test Steps:

1. Navigate to the Tachy Detection page.

2. Select 3-zone mode.

3. For each parameter setting, randomly choose a button, select an available option, and confirm.

4. Once all parameters are selected randomly, evaluate logical judgments; if a red bubble appears on the current page, it indicates parameter conflicts.

5. If there are conflicts, the program button should be disabled. If no conflicts, program the selected values, and then retrieve to verify data integrity.

6. Repeat from step 3.

Expected Output:

- The buttons in 3-zone mode on the Tachy Detection page should function correctly.

- The logical relationships between VF, FVT, and VT parameters should be appropriately enforced.

- Any inconsistencies or conflicts between parameters should result in the program button being disabled, and the test case should be marked as failed.

Risks and Dependencies:

- Correct enforcement of logical relationships between VF, FVT, and VT parameters.

- Dependency on the correct functioning of the program and retrieve buttons.

- Interaction between different settings and values, especially conflict detection.

Review and Approval:

- [To be completed by relevant stakeholders]

Version Control:

- SourceTree, version tracking as per organizational guidelines

- BLE version: 1.0.20

- Firmware version: 1.0.21

- Software version: 1.0.21

Attachments and Appendices:

- [Any additional diagrams, charts, or visual aids]