**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]

Test Case 8: TachyTherapyShockParametersTest

Test Purpose:

Verify that in the Tachy Therapy page, during the 1-zone test in the VF zone shock parameter, the tilt and width setting buttons cannot be enabled simultaneously.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Validation Logic under Tachy Therapy Shock Parameter setting.

Test Steps:

1. In the tachyDetection page, select 1-zone.

2. Proceed to the Tachy Therapy interface, click on VF Shock1, and then click Delete.

3. Enter the VF Shock1 interface and repeatedly verify the enabled logic by clicking back and forth on the Tilt and Width buttons, randomly selecting values.

Expected Results:

**- The Tilt and Width setting buttons should not be enabled at the same time.**

Test Case 10: TachyTherapyTest2

Test Purpose:

Verify that the Shock Energy in the Tachy Therapy page under Shock Parameter is correctly displayed and returned. Validate the logical relationship between ATP1 and ATP2, and ensure the correct disabling logic of different buttons caused by different ATP types within the ATP settings.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Tachy Therapy 2-zone mode, verification of Shock Energy display. Logic validation between ATPs, and internal logic validation within ATP.

Test Steps:

1. In the tachyDetection page, select 2-zone.

2. Iteratively click each parameter setting under Tachy Detection and set them to valid values.

3. Proceed to the Tachy Therapy interface, click on VF Shock1, FVT ATP1 and 2, and then click Delete on each.

4. Enter both VF Shock1 and VF Shock2 pages separately, randomly select valid values, and confirm to ensure that these parameters do not interfere with our validation for FVT.

5. For FVT's ATP and Shock parameter settings, randomly select values. Verify if Shock Energy units are correctly displayed, if ATP1's type and ATP2's type are inconsistent, and if the ATP type's settings for Burst Decrement and In Burst Decrement button disabling logic are correct, etc.

Expected Results:

- Shock Energy units should be correctly displayed.

- ATP1's type and ATP2's type should be inconsistent.

- The ATP type's settings for Burst Decrement and In Burst Decrement button disabling logic should be correct.

Risks and Dependencies:

- None

Test Case 11: TachyTherapyTest3

Test Purpose:

Verify that the Shock Energy in the Tachy Therapy page under Shock Parameter is correctly displayed and returned. Validate the logical relationship between ATP1 and ATP2, and ensure the correct disabling logic of different buttons caused by different ATP types within the ATP settings.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Tachy Therapy 3-zone mode, verification of Shock Energy display. Logic validation between ATPs, and internal logic validation within ATP.

Test Steps:

1. In the tachyDetection page, select 3-zone.

2. Iteratively click each parameter setting under Tachy Detection and set them to valid values.

3. Proceed to the Tachy Therapy interface, click on VF Shock1, FVT ATP1 and 2, VT ATP1 and 2, and then click Delete on each.

4. Enter both VF Shock1 and VF Shock2 pages separately, randomly select valid values, and confirm to ensure that these parameters do not interfere with our validation for FVT and VT.

5. For FVT and VT's ATP and Shock parameter settings, randomly select values. Verify if Shock Energy units are correctly displayed, if ATP1's type and ATP2's type are inconsistent, and if the ATP type's settings for Burst Decrement and In Burst Decrement button disabling logic are correct, etc.

Expected Results:

- Shock Energy units should be correctly displayed.

- ATP1's type and ATP2's type should be inconsistent.

- The ATP type's settings for Burst Decrement and In Burst Decrement button disabling logic should be correct.

Test Case 12: TachyTherapyPostShockPacing

Test Purpose:

Verify whether the parameters under Post Shock Pacing in the Tachy Therapy page can be set, returned, and displayed correctly. Also, validate if the disabling logic for other buttons is correct when the Pacing Period is set to off within the Post Shock Pacing parameter settings.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Tachy Therapy 1-zone mode, verification logic, and correct display of Post Shock Pacing parameters.

Test Steps:

1. In the tachyDetection page, select 1-zone.

2. Iteratively click each parameter setting under Tachy Detection and set them to valid values.

3. Proceed to the Tachy Therapy interface, enter the Post Shock Pacing interface.

4. Iteratively click each parameter value and validate the effect of Pacing Period on the disabling of other buttons.

5. Click confirm to return to the Tachy Therapy interface. Verify if the Post Shock Pacing parameters have been correctly displayed.

Expected Results:

- The Post Shock Pacing parameters should be displayed correctly.

- The disabling logic for other buttons should be correct when Pacing Period is set to off.

- All parameters should be correctly set, returned, and displayed.

Test Case 13: AlertTest1

Test Purpose:

Verify whether the parameters under the Alerts section can be correctly set and returned.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- All parameters under the Alerts section.

Test Steps:

1. Navigate to the Alerts interface.

2. Sequentially and randomly click on all parameters on this page and set them.

3. Click on the 'program' and 'retrieve' buttons, then verify the completeness and consistency of the parameters.

Expected Results:

- All the parameters under the Alerts section should be set correctly and retrieved with the same values.

- The system should not present any incorrect or unexpected behavior during the process.

Risks and Dependencies:

- It is preferable to test on a virtual device. Real device testing may be affected by possible pop-up interference.

Test Case 14: SafetyCoreTestTest1

Test Purpose:

Verify whether the parameters under the SafetyCore -> Brady section can be correctly set and returned.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- All parameters under the SafetyCore -> Brady section.

Test Steps:

1. Navigate to the SafetyCore -> Brady interface.

2. Sequentially and randomly click on all parameters on this page and set them.

3. Click on the 'program' and 'retrieve' buttons, then verify the completeness and consistency of the parameters.

Expected Results:

- All the parameters under the SafetyCore -> Brady section should be set correctly and retrieved with the same values.

- The system should not present any incorrect or unexpected behavior during the process.

Risks and Dependencies:

- It is preferable to test on a virtual device. Real device testing may be affected by possible pop-up interference.

Test Case 15: SafetyCoreTestTest2

Test Purpose:

Verify whether the parameters under the SafetyCore -> Tachy section can be correctly set and returned.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- All parameters under the SafetyCore -> Tachy section.

Test Steps:

1. Navigate to the SafetyCore -> Tachy interface.

2. Sequentially and randomly click on all parameters on this page and set them.

3. Click on the 'program' and 'retrieve' buttons, then verify the completeness and consistency of the parameters.

Expected Results:

- All the parameters under the SafetyCore -> Tachy section should be set correctly and retrieved with the same values.

- The system should not present any incorrect or unexpected behavior during the process.

Test Case 16, Name: ResetLogTest1

Test Purpose:

To verify if the ResetLog correctly displays an empty log list when all valid data is 0. The goal is to ensure that ResetLog can display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of ResetLog.

Test Steps:

1. Navigate to the ResetLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. The purpose of this step is to ensure that the actual displayed logs are completely consistent with the data returned by the device.

Expected Results:

- If all valid data returned by the device is 0, the log list should be empty.

- If the data returned by the device is not 0, the log list should accurately reflect this data, including log content, quantity, and display order.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case

Test Case 17, Name: ResetLogTest2

Test Purpose:

To verify if the ResetLog correctly displays the log list when all valid data is not 0. The goal is to ensure that ResetLog can display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of ResetLog.

Test Steps:

1. Navigate to the ResetLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. The purpose of this step is to ensure that the actual displayed logs are completely consistent with the data returned by the device, even when the data is not 0.

Expected Results:

- If all valid data returned by the device is not 0, the log list should accurately reflect this data, including log content, quantity, and display order.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or missing entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 18, Name: ResetLogTest3

Test Purpose:

To verify if the ResetLog correctly displays the log list when the deviceId and resetReason are 0, but operationIdLog is not 0. The goal is to ensure that ResetLog can display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of ResetLog.

Test Steps:

1. Navigate to the ResetLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are correctly displayed even when the deviceId and resetReason are 0, but operationIdLog is not 0.

Expected Results:

- All logs should be correctly displayed, reflecting the scenario where deviceId and resetReason are 0, but operationIdLog is not 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or missing entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 19, Name: ResetLogTest4

Test Purpose:

To verify if the ResetLog correctly displays the log list when the deviceId and resetReason are not 0, but operationIdLog is 0. The goal is to ensure that ResetLog can display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of ResetLog.

Test Steps:

1. Navigate to the ResetLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are correctly displayed when the deviceId and resetReason are not 0, but operationIdLog is 0.

Expected Results:

- All logs should be correctly displayed, reflecting the scenario where deviceId and resetReason are not 0, but operationIdLog is 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or missing entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 20, Name: TachyLogTest1

Test Purpose:

To verify if the TachyLog correctly displays the log list as empty when all valid values are 0. The goal is to ensure that TachyLog can display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of TachyLog.

Test Steps:

1. Navigate to the TachyLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are correctly displayed as empty when all valid values are 0.

Expected Results:

- Log list should be empty, reflecting the scenario where all valid values are 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or missing entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 21, Name: TachyLogTest2

Test Purpose:

To verify if the TachyLog correctly displays all logs when all valid values are not 0. The goal is to ensure that TachyLog can display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of TachyLog.

Test Steps:

1. Navigate to the TachyLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are correctly displayed in the correct order and logic when all valid values are not 0.

Expected Results:

- Log list should be in a full state, reflecting the scenario where all valid values are not 0.

- All logs should be displayed in the correct order and logic.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or missing entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 22, Name: TachyLogTest3

Test Purpose:

To verify if the TachyLog correctly displays all logs when recordReason is 0, but recordMode is not 0. The goal is to ensure that TachyLog can display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of TachyLog.

Test Steps:

1. Navigate to the TachyLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are correctly displayed in the correct order and logic when recordReason is 0, but recordMode is not 0.

Expected Results:

- Log list should be in a full state, reflecting the scenario where recordReason is 0, but recordMode is not 0.

- All logs should be displayed in the correct order and logic.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or missing entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 23, Name: TachyLogTest4

Test Purpose:

To verify if the TachyLog correctly displays all logs when recordReason is not 0, but recordMode is 0. The goal is to ensure that TachyLog can display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of TachyLog.

Test Steps:

1. Navigate to the TachyLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are correctly displayed in the correct order and logic when recordReason is not 0, but recordMode is 0.

Expected Results:

- Log list should be in a full state, reflecting the scenario where recordReason is not 0, but recordMode is 0.

- All logs should be displayed in the correct order and logic.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or missing entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case

Test Case 24, Name: FaultLogTest1

Test Purpose:

To verify if the FaultLog correctly displays an empty log list when all effective data is 0. The goal is to ensure that FaultLog can accurately display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of FaultLog.

Test Steps:

1. Navigate to the FaultLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are correctly displayed as empty when all effective data is 0.

Expected Results:

- Log list should be empty, reflecting the scenario where all effective data is 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or missing entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 25, Name: FaultLogTest2

Test Purpose:

To verify if the FaultLog correctly displays the log list when all effective data is not 0. The goal is to ensure that FaultLog can accurately display the corresponding log list based on the device's returned values, including log content, log quantity, display order, etc.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of FaultLog.

Test Steps:

1. Navigate to the FaultLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are correctly displayed when all effective data is not 0.

Expected Results:

- Log list should be full, reflecting the scenario where all effective data is not 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or missing entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 26, Name: FaultLogTest3

Test Purpose:

To verify if the FaultLog displays the log list as empty when the faultId is 0. The test is designed to ensure that the FaultLog can accurately display the corresponding log list, including log content, log quantity, display order, etc., based on the device's returned values.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of FaultLog.

Test Steps:

1. Navigate to the FaultLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are displayed as empty when faultId is 0.

Expected Results:

- Log list should be empty, reflecting the scenario where faultId is 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 27, Name: FaultLogTest4

Test Purpose:

To verify if the FaultLog displays the log list as empty when the faultCount is 0. The test is designed to ensure that the FaultLog can accurately display the corresponding log list, including log content, log quantity, display order, etc., based on the device's returned values.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of FaultLog.

Test Steps:

1. Navigate to the FaultLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are displayed as empty when faultCount is 0.

Expected Results:

- Log list should be empty, reflecting the scenario where faultCount is 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 28, Name: FaultLogTest5

Test Purpose:

To verify if the FaultLog displays the log list normally when the status is 0. This test is aimed at ensuring that the FaultLog can accurately display the corresponding log list, including log content, log quantity, display order, etc., based on the device's returned values.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of FaultLog.

Test Steps:

1. Navigate to the FaultLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are displayed normally when status is 0.

Expected Results:

- Log list should be displayed in normal state, reflecting the scenario where status is 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 29, Name: ChargeLogTest1

Test Purpose:

To verify if the ChargeLog displays the log list as empty when all valid data is 0. This test is aimed at ensuring that the ChargeLog can accurately display the corresponding log list, including log content, log quantity, display order, etc., based on the device's returned values.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of ChargeLog.

Test Steps:

1. Navigate to the ChargeLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are displayed as empty when all valid data is 0.

Expected Results:

- Log list should be empty, reflecting the scenario where all valid data is 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case

Test Case 30, Name: ChargeLogTest2

Test Purpose:

To verify if the ChargeLog displays the log list correctly when all valid data is not 0. This test is aimed at ensuring that the ChargeLog can accurately display the corresponding log list, including log content, log quantity, display order, etc., based on the device's returned values.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of ChargeLog.

Test Steps:

1. Navigate to the ChargeLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are displayed correctly when all valid data is not 0.

Expected Results:

- Log list should display all content, reflecting the scenario where all valid data is not 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 31, Name: ChargeLogTest3

Test Purpose:

To verify if the ChargeLog displays the log list correctly when the Duration is 0. This test is aimed at ensuring that the ChargeLog can accurately display the corresponding log list, including log content, log quantity, display order, etc., based on the device's returned values.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of ChargeLog.

Test Steps:

1. Navigate to the ChargeLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are displayed correctly when the Duration is 0.

Expected Results:

- Log list should display all content, reflecting the scenario where the Duration is 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 32, Name: ChargeLogTest4

Test Purpose:

To verify if the ChargeLog displays the log list correctly when the status is 0. This test is aimed at ensuring that the ChargeLog can accurately display the corresponding log list, including log content, log quantity, display order, etc., based on the device's returned values.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of ChargeLog.

Test Steps:

1. Navigate to the ChargeLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are displayed correctly when the status is 0.

Expected Results:

- Log list should display all content, reflecting the scenario where the status is 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 33, Name: ChargeLogTest5

Test Purpose:

To verify if the ChargeLog displays the log list correctly when the timeStamp is 0. This test is aimed at ensuring that the ChargeLog can accurately display the corresponding log list, including log content, log quantity, display order, etc., based on the device's returned values, especially when the timeStamp is 0.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of ChargeLog.

Test Steps:

1. Navigate to the ChargeLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the logs are displayed correctly when the timeStamp is 0.

Expected Results:

- Log list should display all content, reflecting the scenario where the timeStamp is 0.

- The displayed data must be consistent with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 34, Name: BatteryLogTest1

Test Purpose:

To verify if the BatteryLog displays an empty log list when all valid data are 0. This includes checking whether the BatteryLog can accurately display the corresponding log list based on the device's returned values, ensuring that the log content, log quantity, and display order are correct when all valid data are 0.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of BatteryLog.

Test Steps:

1. Navigate to the BatteryLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the log list is empty when all valid data are 0.

Expected Results:

- Log list should be empty, reflecting the scenario where all valid data are 0.

- The displayed data must align with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 35, Name: BatteryLogTest2

Test Purpose:

To verify if the BatteryLog displays the correct log content when the timestamp is empty, but the status is not empty. This includes checking whether the BatteryLog can accurately display the corresponding log list based on the device's returned values, ensuring that the log content, log quantity, and display order are correct when the timestamp is empty but the status is not.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Display logic of BatteryLog.

Test Steps:

1. Navigate to the BatteryLog page.

2. Repeatedly click the Refresh button.

3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the log content is correctly displayed when the timestamp is empty but the status is not.

Expected Results:

- Log content should be correctly displayed, reflecting the scenario where the timestamp is empty but the status is not.

- The displayed data must align with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

- Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

**Test Case 36, Name: BatteryLogTest3**

Test Purpose: To verify if the BatteryLog displays the correct log content when the timestamp is not empty, but the status is empty. This includes checking whether the BatteryLog can accurately display the corresponding log list based on the device's returned values, ensuring that the log content, log quantity, and display order are correct when the timestamp is not empty but the status is.

Test Environment and Setup:

* BLE Version: 1.0.20
* Firmware Version: 1.0.21
* Software Version: 1.0.21

Test Subject:

* Display logic of BatteryLog.

Test Steps:

1. Navigate to the BatteryLog page.
2. Repeatedly click the Refresh button.
3. Based on the data returned by the device, parse the data locally, and then compare it with the actual displayed data. Specifically, verify that the log content is correctly displayed when the timestamp is not empty but the status is empty.

Expected Results:

* Log content should be correctly displayed, reflecting the scenario where the timestamp is not empty but the status is.
* The displayed data must align with the returned values from the device, and there should be no discrepancies or unexpected entries.

Risks and Dependencies:

* Espresso's data parsing logic heavily depends on the logic in Programmer, so any changes to Programmer logic might affect the accuracy of this test case.

Test Case 37, Name: ManufacturingTest1

Test Purpose:

To verify if the parameters in Manufacturing can be read and written normally in DeBug mode, including the proper functioning of reading, writing, and entering Storage Mode.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters in Manufacturing, including reading and writing functionalities, ensuring that entering Storage Mode works as expected.

Test Steps:

1. Click the gear icon on the main page to go to System Setting.

2. Click on the System Setting section.

3. Enable Manufacturing/Engineering Mode.

4. Go to Debug test pages.

5. Click to enter the Manufacturing page.

6. Write and read the BLE TX Power Level, and perform data verification.

Risks and Dependencies:

- The methods to enter Debug Mode may vary between virtual devices and physical devices, and this must be taken into consideration during testing.

Test Case 38, Name: ManufacturingTest2

Test Purpose:

To verify if the parameters in Manufacturing can be read and written normally in Debug mode, and if entering Storage Mode functions properly. This test also verifies that the Manufacturing Support parameters can be randomly inputted and programmed, and checks whether the data has been written normally.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters under Manufacturing, including reading and writing functionalities, the ability to enter Storage Mode, and the proper writing of Manufacturing Support parameters.

Test Steps:

1. Click the gear icon on the main page to go to System Setting.

2. Click on the System Setting section.

3. Enable Manufacturing/Engineering Mode.

4. Go to Debug test pages.

5. Click to enter the Manufacturing page.

6. Click to enter Storage Mode.

7. For Manufacturing Support parameters, randomly input data and click program.

8. Click to enter the Device Info page, and check if the data has been written normally.

Expected Results:

- Parameters in Manufacturing should be read and written without issues.

- Entry into Storage Mode should be normal.

- Randomly inputted data for Manufacturing Support parameters should be programmed correctly, and the Device Info page should reflect this accurate data writing.

Risks and Dependencies:

- The methods to enter Debug Mode may vary between virtual devices and physical devices, and this must be taken into consideration during testing.

Test Case 39, Name: DeviceCalibrationTest1

Test Purpose:

To verify if the parameters in Device Calibration can be read and written normally in Debug mode. This includes checking whether all parameters on the Sensing page can be clicked and assigned a reasonable value.

Test Environment and Setup:

- BLE Version: 1.0.20 - Firmware Version: 1.0.21 - Software Version: 1.0.21

Test Subject:

- Parameters under Device Calibration, focusing on whether they can be properly read and written.

Test Steps:

1. Click the gear icon on the main page to go to System Setting.

2. Click on the System Setting section.

3. Enable Manufacturing/Engineering Mode.

4. Go to Debug test pages.

5. Click to enter the Device Calibration page.

6. Click to enter Sensing.

7. For all parameters on this page, sequentially click and enter a reasonable value.

Expected Results:

- All parameters under Device Calibration should be accessible, and entering reasonable values for each one should be allowed without issues.

- The application should not crash or present errors when handling these interactions.

- All of the data should be consistent after programming and retrieving.

Risks and Dependencies:

- Values entered must be within a reasonable range, limiting the scope of this test.

- Since this is Debug mode, excessive requirements on data validity are not imposed, potentially leading to cases where improper values might not be identified as issues.

Test Case 40, Name: DeviceCalibrationTest2

Test Purpose:

To verify if the parameters in Device Calibration can be read and written normally in Debug mode. This includes checking whether all parameters on the Pacing page can be clicked and assigned a reasonable value.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters under Device Calibration, focusing on whether they can be properly read and written.

Test Steps:

1. Click the gear icon on the main page to go to System Setting.

2. Click on the System Setting section.

3. Enable Manufacturing/Engineering Mode.

4. Go to Debug test pages.

5. Click to enter the Device Calibration page.

6. Click to enter Pacing.

7. For all parameters on this page, sequentially click and enter a reasonable value.

Expected Results:

- All parameters under Device Calibration should be accessible, and entering reasonable values for each one should be allowed without issues.

- The application should not crash or present errors when handling these interactions.

Risks and Dependencies:

- Values entered must be within a reasonable range, limiting the scope of this test.

- Since this is Debug mode, excessive requirements on data validity are not imposed, potentially leading to cases where improper values might not be identified as issues.

Test Case 41, Name: DeviceCalibrationTest3

Test Purpose:

To verify if the parameters in Device Calibration can be read and written normally in Debug mode, specifically within the ECG Gain page. This includes checking whether all parameters on the ECG Gain page can be clicked and assigned a reasonable value.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters under Device Calibration, focusing on the ECG Gain page and whether they can be properly read and written.

Test Steps:

1. Click the gear icon on the main page to go to System Setting.

2. Click on the System Setting section.

3. Enable Manufacturing/Engineering Mode.

4. Go to Debug test pages.

5. Click to enter the Device Calibration page.

6. Click to enter ECG Gain.

7. For all parameters on this page, sequentially click and enter a reasonable value.

Expected Results:

- All parameters under the ECG Gain section of Device Calibration should be accessible, and entering reasonable values for each one should be allowed without issues.

- The application should not crash or present errors when handling these interactions.

Risks and Dependencies:

- Values entered must be within a reasonable range, limiting the scope of this test.

- Since this is Debug mode, excessive requirements on data validity are not imposed, potentially leading to cases where improper values might not be identified as issues.

Test Case 42, Name: DeviceCalibrationTest4

Test Purpose:

To verify if the parameters in Device Calibration, specifically within the Temperature page, can be read and written normally in Debug mode. This includes checking whether all parameters on the Temperature page can be clicked and assigned a reasonable value.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters under Device Calibration, focusing on the Temperature page and whether they can be properly read and written.

Test Steps:

1. Click the gear icon on the main page to go to System Setting.

2. Click on the System Setting section.

3. Enable Manufacturing/Engineering Mode.

4. Go to Debug test pages.

5. Click to enter the Device Calibration page.

6. Click to enter Temperature.

7. For all parameters on this page, sequentially click and enter a reasonable value.

Expected Results:

- All parameters under the Temperature section of Device Calibration should be accessible, and entering reasonable values for each one should be allowed without issues.

- The application should not crash or present errors when handling these interactions.

Risks and Dependencies:

- Values entered must be within a reasonable range, limiting the scope of this test.

- Since this is Debug mode, excessive requirements on data validity are not imposed, potentially leading to cases where improper values might not be identified as issues.

Test Case 43, Name: DeviceCalibrationTest5

Test Purpose:

To verify if the parameters in Device Calibration, specifically within the HIGH VOLTAGE page, can be read and written normally in Debug mode. This includes checking whether all parameters on the HIGH VOLTAGE page can be clicked and assigned a reasonable value.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters under Device Calibration, focusing on the HIGH VOLTAGE page and whether they can be properly read and written.

Test Steps:

1. Click the gear icon on the main page to go to System Setting.

2. Click on the System Setting section.

3. Enable Manufacturing/Engineering Mode.

4. Go to Debug test pages.

5. Click to enter the Device Calibration page.

6. Click to enter HIGH VOLTAGE.

7. For all parameters on this page, sequentially click and enter a reasonable value.

Expected Results:

- All parameters under the HIGH VOLTAGE section of Device Calibration should be accessible, and entering reasonable values for each one should be allowed without issues.

- The application should not crash or present errors when handling these interactions.

Risks and Dependencies:

- Values entered must be within a reasonable range, limiting the scope of this test.

- Since this is Debug mode, excessive requirements on data validity are not imposed, potentially leading to cases where improper values might not be identified as issues.

Test Case 44, Name: DeviceCalibrationTest6

Test Purpose:

To verify if the parameters in Device Calibration, specifically within the BATTERY page, can be read and written normally in Debug mode. This includes checking whether all parameters on the BATTERY page can be clicked and assigned a reasonable value.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters under Device Calibration, focusing on the BATTERY page and whether they can be properly read and written.

Test Steps:

1. Click the gear icon on the main page to go to System Setting.

2. Click on the System Setting section.

3. Enable Manufacturing/Engineering Mode.

4. Go to Debug test pages.

5. Click to enter the Device Calibration page.

6. Click to enter BATTERY.

7. For all parameters on this page, sequentially click and enter a reasonable value.

Expected Results:

- All parameters under the BATTERY section of Device Calibration should be accessible, and entering reasonable values for each one should be allowed without issues.

- The application should not crash or present errors when handling these interactions.

Risks and Dependencies:

- Values entered must be within a reasonable range, limiting the scope of this test.

- Since this is Debug mode, excessive requirements on data validity are not imposed, potentially leading to cases where improper values might not be identified as issues.

Test Case 45, Name: LeadInfoTest1

Test Purpose:

To verify whether the parameters under Lead Info in Device Info can be normally written to and read from.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters under Lead Info in Device Info, focusing on whether they can be properly read and written.

Test Steps:

1. Click to enter the Device Info page.

2. Write to and then read the parameters to verify data consistency.

Expected Results:

- All parameters under Lead Info should be accessible, and writing to and reading from them should occur without issues.

- The application should not crash or present errors during these interactions.

- The data read should match the data written, ensuring consistency.

Risks and Dependencies:

- None.

Notes:

- Lead Info is vital for obtaining information about the leads connected to a device. Any discrepancies in this section may lead to inaccurate diagnostics and other potential issues.

- The simplicity of this test should be complemented with additional testing to make sure that the system is robust against potential edge cases and unexpected user inputs.

Test Case 46, Name: PatientInformationTest1

Test Purpose:

To verify whether the parameters under Patient Information can be normally written to and read from.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters under Patient Information, focusing on whether they can be properly read and written.

Test Steps:

1. Click to enter the Device Info page.

2. Click to enter Patient Information.

3. Randomly write to and then read the parameters to verify data consistency.

Expected Results:

- All parameters under Patient Information should be accessible, and writing to and reading from them should occur without issues.

- The application should not crash or present errors during these interactions.

- The data read should match the data written, ensuring consistency.

Risks and Dependencies:

- No date validation logic of timestamp yet.

Notes:

- Patient Information is crucial for personalized care and accurate diagnostics. Any discrepancies in this section could lead to errors in patient management.

- Consider adding more detailed test scenarios to cover all possible data fields and formats, especially date-related fields, given the noted lack of date validation logic.

Test Case 47, Name: ClinicianNoteTest1

Test Purpose:

To verify whether the parameters under Clinician Note can be normally written to and read from.

Test Environment and Setup:

- BLE Version: 1.0.20

- Firmware Version: 1.0.21

- Software Version: 1.0.21

Test Subject:

- Parameters under Clinician Note, focusing on whether they can be properly read and written.

Test Steps:

1. Click to enter the Device Info page.

2. Click to enter Clinician Note.

3. Randomly write to and then read the parameters to verify data consistency.

Expected Results:

- All parameters under Clinician Note should be accessible, and writing to and reading from them should occur without issues.

- The application should not crash or present errors during these interactions.

- The data read should match the data written, ensuring consistency.

Risks and Dependencies:

- None.

Notes:

- Clinician notes are vital for maintaining proper patient history and communication between healthcare providers. Ensuring that this information can be accurately recorded and retrieved is essential.

- It may be beneficial to conduct this test with real-world examples of clinician notes to ensure that all potential formats and content are properly supported.