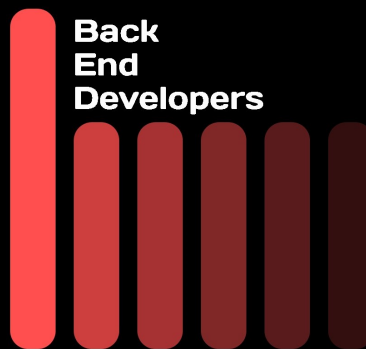


# EMA Tracker for Elderly LSS Patients: System Verification and Validation Plan for Mechatronics Engineering



Team #1, Back End Developers

Jessica Bae

Oliver Foote

Jonathan Hai

Anish Rangarajan

Nish Shah

Labeeb Zaker

April 4, 2023

## 1 Revision History

Date	Version	Notes
2022-11-02	1.0	Initial Documentation
2023-03-15	2.0	Minor improvements and proof reading for revision 1
2023-03-17	2.1	Incorporated TA feedback from revision 0 evaluation: Removed HST4 and added traceability for hazard analysis requirements
2023-04-03	2.2	Added more unit testing cases
2023-04-04	2.3	Added logo and added style to the document

# Contents

<b>1</b>	<b>Revision History</b>	<b>i</b>
<b>2</b>	<b>Symbols, Abbreviations and Acronyms</b>	<b>iv</b>
<b>3</b>	<b>General Information</b>	<b>1</b>
3.1	Summary . . . . .	1
3.2	Objectives . . . . .	2
3.3	Relevant Documentation . . . . .	2
<b>4</b>	<b>Plan</b>	<b>2</b>
4.1	Verification and Validation Team . . . . .	3
4.2	SRS Verification Plan . . . . .	4
4.3	Design Verification Plan . . . . .	5
4.4	Verification and Validation Plan Verification Plan . . . . .	6
4.5	Implementation Verification Plan . . . . .	6
4.6	Automated Testing and Verification Tools . . . . .	7
4.7	Software Validation Plan . . . . .	7
<b>5</b>	<b>System Test Description</b>	<b>8</b>
5.1	Tests for Functional Requirements . . . . .	8
5.1.1	Duration Test . . . . .	8
5.1.2	Device should track Minor Movements . . . . .	9
5.1.3	Prompt Tests . . . . .	10
5.1.4	Customizable Thresholds . . . . .	11
5.1.5	Data Storage . . . . .	13
5.1.6	Data Extraction . . . . .	14
5.2	Tests for Nonfunctional Requirements . . . . .	15
5.2.1	Hardware Safety . . . . .	15
5.2.2	Reusability . . . . .	16
5.2.3	Accuracy . . . . .	16
5.2.4	Usability . . . . .	17
5.2.5	Performance . . . . .	19
5.2.6	Data Security Tests . . . . .	20
5.3	Traceability Between Test Cases and Requirements . . . . .	21
<b>6</b>	<b>Unit Test Description</b>	<b>23</b>
6.1	Unit Testing Scope . . . . .	23
6.2	Unit Testing Plan . . . . .	23
6.3	Tests for Functional Requirements . . . . .	26
6.4	Tests for Nonfunctional Requirements . . . . .	26
6.5	Traceability Between Unit Test Cases and Requirements . . . . .	26

## List of Tables

1	Table of Symbols . . . . .	iv
2	Team roles for verification and validation. . . . .	3
3	SRS verification checklist and task based inspection . . . . .	4
4	Plan for verification of the design of the system. . . . .	5
5	Requirements Traceability Matrix Part 1 . . . . .	21
6	Requirements Traceability Matrix Part 2 . . . . .	21
7	Requirements Traceability Matrix Part 3 . . . . .	22
8	Hazard Analysis Traceability Matrix Part 1 . . . . .	22
9	Hazard Analysis Traceability Matrix Part 2 . . . . .	22
10	Unit Test Traceability Matrix Pt 1 . . . . .	26
11	Unit Test Traceability Matrix Pt 2 . . . . .	27
12	Table of Errors . . . . .	29

## 2 Symbols, Abbreviations and Acronyms

Symbol	Description
DT	Duration Test
MTT	Minor Tracking Test
PRT	Prompt Test
TT	Threshold Test
DST	Data Storage Test
DXT	Data Extraction Test
HC#	Hardware Constraint #
SAST	Static Application Security Testing
DAST	Dynamic Application Security Testing
UT	Usability Test
PT	Performance Test
HST	Hardware Safety Test
RT	Re-usability Test
DSQT	Data Security Test

Table 1: Table of Symbols

## 3 General Information

### 3.1 Summary

Researchers at the School of Rehabilitation Sciences (SReS) at McMaster University are interested in performing Ecological Momentary Assessment (EMA) for victims of spinal disorders and back pain. EMA aims to study the thoughts, experiences, and behaviours of a participant's daily life by repeatedly collecting data in an individual's normal environment, at or close to the time they carry out that behaviour.

The type of EMA that the SReS is interested in is focused on analyzing the daily activities and symptoms of mostly-older adults with mobility and spinal issues. They wish to track that participant's walking activity as they go about their daily life, along with prompting participants with questions when relevant events occur. The answers to event-based prompts will be combined with activity monitoring data to form a better picture about the experience this participant has with their spinal and mobility issues.

The EMAnator will perform EMA analysis in a manner which even older participants can use, integrated into one package which gathers relevant data about a participant's activities and allows them to easily report what is currently going on and how they feel. They also are looking for a way to access EMA data in various ways. This includes graphical representations of the data which are meaningful to researchers, along with the raw data itself. This data could be activity data, symptoms reporting data, both types of data collated together, and so on.

This document is intended to describe the plan for verification and validation of the device. Here, verification and validation are both technical terms with very specific meanings.

Verification involves checking whether or not the specifications which were described in the planning phase of the project are implemented correctly by the final system designed. Validation involves checking whether or not the product actually fulfills the needs of the end user of the device. In the words of Barry Boehm, verification asks, "Are we building the product right?" and validation asks, "Are we building the right product?" [Pham \(1999\)](#).

## 3.2 Objectives

The objectives aimed to be accomplished by this verification and validation are to:

- Validate if user requirements truly represent the goals of the stakeholders of the EMAnator.
- Validate if the input provided by the operators of the device meet the established rules and constraints.
- Verify if the high-level design of the device correctly fulfills the specifications of the functional and non-functional requirements.
- Verify if the components of the device (e.g. source code, database, physical construction, user interface, etc.) fulfill the specifications of the design.
- Discover any faults, failures, or malfunctions in the device.

## 3.3 Relevant Documentation

Please refer to the following documentation for reference and more information:

- Development Plan: [Developers \(2022a\)](#)
- Problem Statement & Goals: [Developers \(2022c\)](#)
- SRS: [Developers \(2022e\)](#)
- Hazard Analysis: [Developers \(2022b\)](#)
- VnV Plan: [Developers \(2022g\)](#)
- Reflection: [Developers \(2022d\)](#)
- User Guide: [Developers \(2022f\)](#)

## 4 Plan

The Verification and Validation plan will be done on both the hardware and software side individually as well as on the system integration side. The researchers, stakeholders and the Back End Developers will have different roles and techniques to test the device's functionality and usability as described below.

## 4.1 Verification and Validation Team

The following table defines the different roles for the stakeholders and the Back End Developers for verification and validation.

Table 2: Team roles for verification and validation.

Name	Role	Description
Dr Luciano Macedo	End user	The stakeholder. She and her team of researchers will verify all the user requirements and functionality according to each sprint.
Jonathan Hai	Subsystem tester	There will be a set of test cases pre-defined for every functional requirement. Jonathan will be testing the specific functions against the test cases using unit testing, integration testing and boundary condition tests as well as SRS verification.
Jessica Bae	Black Box tester	Jessica will test the specific functions of the device without knowing how the internal workflow of said device is structured. This will include stress testing for I/O for all subsystems and boundary condition testing. Valgrind will also be used for code profiling and debugging memory leaks.
Labeeb Zaker	Remote codebase manager	Labeeb will ensure the codebase on GitHub has no flaws, maintain CI/CD, and review every commit added to ensure it meets the requirements without any errors.
Nish Shah	Automated tester	Nish will monitor and maintain the automated testing scripts to run certain functions against test cases that are repetitive to continuously check if the system's code functions as intended. This will be done primarily using Python code coverage tools and unit testing frameworks for code development (unittest, Coverage.py).
Anish Rangarajan	Hardware & Automated tester	Anish will verify and test embedded hardware designs in C using Cunit and bullseye coverage for unit testing framework and code coverage.
Oliver Foote	White Box & Database testing	Oliver will test the internal workflow for valid outputs. Database validation will be performed using Orion for stress testing of I/O and database functionality.



## 4.2 SRS Verification Plan

The SRS will be verified according to a checklist. It will also follow a rigorous task based inspection listed in the table below. This is to make sure all the requirements and goals receive several iterations of verification and validation.

Table 3: SRS verification checklist and task based inspection

Section	Description	Approach of Feedback
Overall	<p>The verification will be done based on an ad hoc approach through the checklist below:</p> <ul style="list-style-type: none"><li><input type="checkbox"/> System constraints for hardware and software are valid.</li><li><input type="checkbox"/> Monitored and controlled variables are consistent.</li><li><input type="checkbox"/> Comparison to existing solutions is unambiguous.</li><li><input type="checkbox"/> Project Goals are relevant to description of device.</li></ul>	Will be done by Stakeholder Dr. Luciana Macedo, supervisor, classmates and the Back End Developers.
4.1-4.3	The verification for this section will be performed by checking if system constraints and user characteristics are provided in detail without any ambiguity.	Will be done by team member Jonathan Hai.
5	Terminology and the use of monitored and controlled variables must be verified to be consistent and correct. Assumptions are clearly mentioned and provided along with goals of the project.	Will be done by team member Nish Shah.
7	Functional and non-functional requirements must be verified. In addition, the requirements must be checked to see if they are classified correctly. Checking required behaviour as well.	Will be done by team member Anish Rangarajan.
8-12	Matrices and graphs must be verified according to requirements. In addition, verification must be performed to see if normal operation covers the required goals of the device.	Will be done by team member Labeeb Zaker.
14-16	The FSM, Legal factors and Phase in Plan must be verified.	Will be done by team member Jessica Bae.

### 4.3 Design Verification Plan

Table 4: Plan for verification of the design of the system.

Test Categories	Description	Approach
Response Feedback Test	This test will test the majority of the sensors of the system. Sensor input will be pushed beyond the established boundaries in different testing scenarios, with feedback being observed for erroneous or strange readings.	Performed by Anish Rangarajan and stakeholders
User Interface Test	Different elements of the user interface will be tested by manipulating said elements in all possible scenarios. In addition, characteristics such as screen size, brightness, and visibility will be tested in different visual environments to determine whether or not they are visible in the correct ranges of use.	Performed by Nish Shah
Durability Test	Each sensor will be tested for failure independence. Should one sensor fail, it must not impact the functionality of any other sensors of the system. To test this, individual sensors will be rendered non-functional, and other sensor values will be checked for erroneous readings.	Performed by Jonathan Hai
Database Scalability Test	This system's database will be tested for volume overload issues. Postman (a request tool) will be used to feed as many requests as possible to the database. The database will then be monitored for faults and errors.	Performed by Labeeb Zaker
Hardware Communication Test	Valgrind will be used at every interaction between hardware components to determine whether or not memory has been allocated properly. An execution tracker will also be used to ensure the workflow is functioning as expected.	Performed by Jessica Bae

## 4.4 Verification and Validation Plan Verification Plan

Since the VnV plan is focused on validating the functionalities of the device through verification of different test cases, the only way to verify this document is to check for all such scenarios and their expected results.

The following will be done to verify the VnV plan:

- ❑ Each task will be verified as per [4.1](#) throughout the development phase. Verifiers must ensure that tests do not fail at any of the expected scenarios and works as expected from user's point of view.
- ❑ Each section of the SRS will be verified as per [4.2](#). Each variable needs to be controlled accordingly and specific scenarios need to be tested to ensure functionality without errors.
- ❑ Each component of the design must be verified as per [4.3](#) as it is being developed, so that unit testing can be performed and evaluated before system testing.
- ❑ Code-review meetings must be attended by all code evaluators. If an evaluator fails to join the review, the review will be rescheduled. Tests will be performed against all possible scenarios to ensure all cases work as intended.
- ❑ All test cases for each functional requirement as per [5.1](#) must be verified accordingly by ensuring that the outputs are expected and within bounds, and that variables are being properly controlled and monitored.

## 4.5 Implementation Verification Plan

The Implementation Verification Plan will utilize static and dynamic code analysis as the debugging methods to find bugs early in development. The checklist below can be used in development to follow the implementation verification plan.

The following methods will be used for static code analysis:

- Bi-weekly code review with sub-teams (hardware, software, frontend/backend) for code walk through.
- Weekly code review with entire team for system-level software design.
- A SAST tool will be used to scan source code, binary and byte code to reveal vulnerabilities.
- Static analysis tools such as Pylint and lint described in [4.6](#) will be used to find bugs in code.

Dynamic code testing will help identify exploitable vulnerabilities. The following methods will be used for dynamic code analysis:

- A DAST tool will be used as a black-box tester. It will input malicious SQL queries, long input strings and invalid data to exploit assumptions made by the developers.
- Modular code will be run separately against a variety of inputs to find bugs in code.
- Open source tools will be used based on the programming language (CrossHair for Python, CHAP for C) to provide a comprehensive view of performance and security of the device.

## 4.6 Automated Testing and Verification Tools

Automated testing and verification tools will be selected based on the different programming languages and tools used in the development of the device. As covered in the development plan [Developers \(2022a\)](#) the following unit testing frameworks, profiling tools and linters will be used for verification of code:

- Python:
  - unittest will be used as the Unit testing framework for Python to cover test automation, aggregation of tests (integration testing) and independence of tests from the reporting framework.
  - Coverage.py will be used to measure code coverage and to gauge the effectiveness of tests performed.
  - Pylint will be used as the linter/static code analyser which will check for errors and enforce a coding standard.
- C:
  - CUnit will be used as a unit testing framework for embedded system development.
  - Bullseye coverage will be used for C code coverage analysis.
  - lint will be used as the linter/static code analyser for development in C.
- Valgrind will be used for Python (pytest-valgrind) and C as a debugging tool for code profiling.
- Orion will be used for SQL/Database testing. It will be used to stress test I/O coverage and to make sure the database functions as expected.

## 4.7 Software Validation Plan

There will be no plan for Software validation since there is no external data used and all the software-related validations are covered in section [4.2](#).

## 5 System Test Description

### 5.1 Tests for Functional Requirements

#### 5.1.1 Duration Test

The following tests will check whether the device will maintain an "ON" state throughout the duration of the monitoring period. The primary tests will involve different monitoring periods with valid inputs, invalid inputs, past dates, or dates too far into the future beyond what the battery life can sustain (HC2).

##### 1. DT\_1: Regular Inputs

Control: Manual, Dynamic

Initial State: Device waits for the monitoring period to be set up in the configuration of the device.

Input: Monitoring Period ["Date","Time"] : ["03-11-2022", "05:30:PM"].

Output: Device turns off after monitoring period.

How test will be performed: The test is performed by passing in a valid monitoring period and ensuring that the device maintains power throughout this period.

##### 2. DT\_2: Invalid Inputs

Control: Manual, Dynamic

Initial State: Device waits for the monitoring period to be set up in the configuration of the device.

Input: Monitoring period ["Date","Time"] : ["3rd November 2022", "Five Thirty PM"].

Output: Device returns an error code to the error handler and asserts the Invalid Data Error (BED\_ERR\_INVALID\_DATA).

How test will be performed: The test is performed by passing an invalid input and ensuring the appropriate error code is returned.

### 3. DT\_3: Earlier Date

Control: Manual, Dynamic

Initial State: Device waits for the monitoring period to be set up in the configuration of the device.

Input: Monitoring period ["Date","Time"] : ["01-1-1999", "05:30:PM"].

Output: Device returns an error code to the error handler and asserts the Invalid Data Error (BED\_ERR\_INVALID\_DATA).

How test will be performed: The test is performed by passing an old date (prior to current date).

### 4. DT\_4: Date Beyond Capabilities

Control: Manual, Dynamic

Initial State: Device waits for the monitoring period to be set up in the configuration of the device.

Input: Monitoring period ["Date","Time"] : ["9-12-2300", "05:30:PM"].

Output: Device returns an error code to the error handler and asserts the Invalid Data Error (BED\_ERR\_INVALID\_DATA).

How test will be performed: The test is performed by passing a date greater than what the battery life of the device can support.

## 5.1.2 Device should track Minor Movements

The following tests will be run to ensure that the device is able to track activities to a resolution deemed sufficient for general activity tracking. Tests will involve varying the rate at which the device is moved, rotated, oriented, etc. and checking if the status of the sensors is valid.

### Minor Tracking Test

#### 1. MTT\_1: Regular Movement

Control: Manual, Dynamic

Initial State: Device is worn with all systems working.

Input: Tester performs activities at a regular/normal pace.

Output: Status returned by the sensor array is no error (BED\_ERR\_NONE) and no prompt should be generated (stay in idle-mode).

How test will be performed: The test is performed by strapping the device onto a tester who will perform the tracked activities between 60 - 100 BPM heartrate and

walking pace, the tester should not have to feel any level of excessive effort for this level of activity. This is done to ensure that the device is functional under "normal" scenarios and is able to differentiate it from "no activity". If tester's heart rate is not within the specified limit, the particular set of data is to be disregarded.

## **2. MTT\_2: Slow Movement**

Control: Manual, Dynamic

Initial State: Device is worn with all systems working.

Input: Tester performs activities at a very slow pace.

Output: Status returned by the sensor array is no error (BED\_ERR\_NONE) and all sensor data are correctly stored within microSD.

How test will be performed: The test is performed by strapping the device onto a tester who will walk at a speed that would exceed 12 steps per 1 minute. Then microSD is inserted into a PC to examine its contents. This is done to ensure that the device can recognize and differentiate slow movements from regular paced movements and capture corresponding sensor data.

### **5.1.3 Prompt Tests**

The following tests will be done to ensure that the proper information is prompted to the user when activities are detected. Tests involve generating the test prompt, and generating prompts for different activities.

#### **1. PRT\_1: Test Prompt**

Control: Manual, Dynamic

Initial State: Device has just been reconfigured.

Input: Device is powered on.

Output: Test Prompt is displayed.

How test will be performed: The test is performed by turning on the device for the first time after reconfiguration. Upon turning on the device, tester should receive a test prompt that to confirm that the prompting system is working correctly. This prompt should not be dependent of any user activities.

Test Prompt: Is this Device on?

Possible Answers: Yes or No

#### **2. PRT\_2: Activity Prompt**

Control: Manual, Dynamic

Initial State: Device is in the idle state.

Input: Walk for 5 seconds and then limp for 5 seconds.

Output: Prompt is generated to ask a question to the tester.

How test will be performed: The test is done by having a tester imitate limping activity. The walking portion of the activity should be performed under the same criteria as MTT\_1.

Prompt: Are you in pain?

Response choices: Yes, No

#### **5.1.4 Customizable Thresholds**

The following tests will be done to ensure that the proper information is prompted to the user when activities are detected. Tests involve generating the test prompt, and generating prompts for different activities.

##### **1. TT\_1: Regular Inputs**

Control: Manual, Dynamic

Initial State: Device is in Configuration mode.

Input: Regular values for thresholds within limits.

Output: Config File generated successfully.

How test will be performed: The test is performed by setting the device to configuration mode and then setting valid values for the thresholds.

eg:

Speed Threshold:

Limits:  $0\text{m/s} < \text{Threshold} < 5\text{m/s}$

##### **2. TT\_2: Below Lower Limit**

Control: Manual, Dynamic

Initial State: Device is in Configuration mode.

Input: Values for thresholds below lower limits.

Output: Config File generates a BED\_ERR\_OUT\_OF\_BOUNDS.

How test will be performed: The test is performed by setting the device to configuration mode and then setting values for the thresholds above allowable limits.



### **3. TT\_3: Above Upper limit**

Control: Manual, Dynamic

Initial State: Device is in Configuration mode.

Input: Values for thresholds above upper limits.

Output: Config File generates a BED\_ERR\_OUT\_OF\_BOUNDS.

How test will be performed: The test is performed by setting the device to configuration mode and then setting values for the thresholds below allowable limits.

### **4. TT\_4: Invalid Value**

Control: Manual, Dynamic

Initial State: Device is in Configuration mode.

Input: Invalid values for thresholds.

Output: Config File generates a BED\_ERR\_INVALID\_DATA.

How test will be performed: The test is performed by setting the device to configuration mode and then setting invalid values for the thresholds.

### **5. TT\_5: No Value**

Control: Manual, Dynamic

Initial State: Device is in Configuration mode.

Input: No values for thresholds.

Output: Config File generates a BED\_ERR\_INVALID\_DATA.

How test will be performed: The test is performed by setting the device to configuration mode and then setting no values for the thresholds.

### 5.1.5 Data Storage

The following tests will be done to ensure that data is stored when appropriate. Tests include checking when storage buffer is full, when prompts are generated and when sensor data needs to be logged.

#### 1. DST\_1: Storage Buffer Full

Control: Manual, Dynamic

Initial State: Device is in an idle state.

Input: Activity detected causing prompt to be generated (Internal storage is full).

Output: Data Storage system generates a BED\_ERR\_MEMORY\_FULL

How test will be performed: The test is performed by first loading the internal memory buffer with garbage values so that it is nearly/completely full. Then a registered activity is triggered generating a prompt. Once this is answered, the system will not have enough memory to store the new values thus resulting in an error.

#### 2. DST\_2: Prompt Generated

Control: Manual, Dynamic

Initial State: Device is in an idle state.

Input: Activity detected causing prompt to be generated.

Output: Prompt Response is saved into the internal memory.

How test will be performed: The test performs registered activities and ensuring that the prompt generated is answered and its result is stored in the internal storage buffer.

eg: Registered Activity: Participant slows down or comes to a stop for an extended period of time

Prompt Generated:

Are you in pain?

Possible Answers: Yes or No

Prompt Generated:

Do you need assistance?

Possible Answers: Yes or No

### 3. **DST\_3: Sensor Storage**

Control: Manual, Dynamic

Initial State: Device is in an idle state.

Input: Activity detected causing prompt to be generated.

Output: Specific sensor values that triggered a prompt are stored in the internal memory.

How test will be performed: The test performs registered activities and ensuring that the sensor values that caused the prompt are stored in the internal storage buffer.

#### 5.1.6 **Data Extraction**

The following tests will be done to ensure that data can be extracted and presented in a graphical manner deemed acceptable for the purpose of EMA analysis.

##### 1. **DXT\_1: Extracting Data**

Control: Manual, Dynamic

Initial State: Device is connected to the device manager.

Input: A command that tells the device manager to extract all data from the internal memory.

Output: Extracted data is sent to the Host Software where it is converted to a presentable form.

How test will be performed: The test is performed by first running the device as intended and waiting for a small monitoring period to finish. After this the device is connected to the host software with the help of the device manager driver. Once connected, the user can start the extraction process and should be able to see all the relevant data in a presentable manner.

##### 2. **DXT\_2: Extracting No Data**

Control: Manual, Dynamic

Initial State: Device is connected to the device manager.

Input: A command that tells the device manager to extract all data from the internal memory.

Output: Device Manager returns a BED\_ERR\_EMPTY\_DATA error

How test will be performed: The test is performed by first deleting all the contents of the internal memory prior to connection with the host software. Once connected and the extraction process begins, the system should return an error due to no data being present to extract.

### 3. **DXT\_3: Extracting Corrupted Data**

Control: Manual, Dynamic

Initial State: Device is connected to the device manager.

Input: A command that tells the device manager to extract all data from the internal memory.

Output: Device Manager returns a `BED_ERR_INVALID_DATA` error

How test will be performed: The test is performed by first deleting all the contents of the internal memory and filling it with garbage values prior to connection with the host software. Once connected and the extraction process begins, the system should return an error due to corrupted data being present to extract.

## 5.2 **Tests for Nonfunctional Requirements**

### 5.2.1 **Hardware Safety**

These tests are designed to test the safety of the device concerning electrical and hardware components. It is important that the users of the device are kept safe, that any systems in place to protect them from electrical shocks are functioning properly, and that the indicator LED functions as expected under certain conditions.

#### 1. **HST\_1: High Voltage**

Control: Manual, Dynamic

Initial State: Device begins in "off" mode.

Input: When the device is turned on a voltage over expected voltage is applied.

Output: Device detects high voltage and indicates that voltage is high using an indicator LED. The breaker or fuse then trips or the voltage is sent to ground.

Test Case Derivation: N/A

How test will be performed: Use a power supply to oversupply the system with voltage.

#### 2. **HST\_2: Normal Voltage**

Control: Manual, Dynamic

Initial State: Device begins in "off" mode.

Input: When device is turned on a normal voltage is applied.

Output: Device powers on, no issues are detected.

Test Case Derivation: N/A

How test will be performed: The power supply will be used to provide the expected voltage to the device.

### 3. **HST\_3: Low Voltage**

Control: Manual, Dynamic

Initial State: Device begins in "off" mode.

Input: When device is turned on a voltage below the expected amount is applied.

Output: If device powers on, indicator LED indicates that the voltage is lower than expected, device goes into power saving mode.

Test Case Derivation:

How test will be performed: Use power supply to provide a lower than expected voltage to the device.

## 5.2.2 **Reusability**

These tests will evaluate the re-usability of various hardware systems of the device. The device should be reusable by multiple participants to prevent electronic waste and reduce cost for the researchers.

### 1. **RT\_1: Battery Charge/Discharge**

Control: Manual, Dynamic

Initial State: Battery is fully discharged.

Input: Connect battery to the charging cable.

Output: Battery is fully charged without issue.

Test Case Derivation: N/A

How test will be performed: Battery component will be isolated from device and fully charged and discharged 5 times.

### 2. **RT\_2: Battery Charge Protection**

Control: Manual, Dynamic

Initial State: Battery is fully charged.

Input: Connect battery to the charging cable.

Output: Battery will not overcharge.

Test Case Derivation: N/A

How test will be performed: Battery component will be isolated from device and fully charged. The battery will then plugged in to the charger for 5 minutes.

## 5.2.3 **Accuracy**

The following set of tests will ensure that the device can recognize its state and perform its main functionality in accurate manner. The following functional requirements test cases are applicable for accuracy:

- MTT\_1: [Regular Movement](#)
- MTT\_2: [Slow Movement](#)
- PRT\_1: [Test Prompt](#)
- PRT\_2: [Activity Prompt](#)

#### 5.2.4 Usability

The following set of tests will ensure that the device is usable by target user population and cause no problems in terms of maintenance and user interactions.

##### 1. UT\_1: Device UI

Type: Manual, Static

Initial State: The device is turned on and a question prompt is generated on screen.

Input/Condition: Testers are asked to answer the questionnaire set without assistance in less than 1 minute.

Output/Result: All testers are able to answer their prompt successfully.

How test will be performed: A test group of people with age of 40 or older will be asked to receive a powered-on EMAnator and asked to respond to a set of 3 questions in 1 minute. These questions will randomly be selected from the following 10 questions.

- Are you a dog person or a cat person?
- Are you currently inside of a building?
- Are you currently a student?
- Did you make your bed this morning?
- Do you feel tired right now?
- Do you prefer coffee or tea?
- Do you have a G driver's license?
- Do you feel sleepy right now?
- Which is better: Summer vs Winter
- Is it daytime right now?

##### 2. UT\_2: Comfortable Device

Type: Manual, Static

Initial State: The device is turned on and left on idle mode.

Input: Testers are asked to wear the device for 24 hours.

Output: Testers fill out a survey form regarding the comfort of the device on their body.

How test will be performed: At the end of their 1 day cycle, testers will be asked to fill out an online form indicating how comfortable they felt the device was regards to weight, shape, stability, etc. The following questions will be asked.

- Did the device every fall off? If so, please record the following for each case: What you were doing each time? When did the incident happen?
- How do you feel regarding the weight of the device? Was it too heavy or too light?
- How do you feel regarding the texture of the device? Did you find it uncomfortable in any way?

### **3. UT\_3: Reusability**

Type: Manual, Static

Initial State: The device is cleaned using isopropyl alcohol wipes.

Input: Testers visually inspect how clean the device is.

Output: The device is deemed clean and the device is not harmed.

How test will be performed: 1 member of Back End Developers will clean the device and 5 other members will inspect and judge if the device is clean enough to be reused. Among the 5 inspecting members, 1 member will then turn on the device to make sure that there was no harm done to the device while cleaning it.

### **4. UT\_4: Host Software UI**

Type: Manual, Static

Initial State: Main page displayed after login page.

Input: Testers try to filter out database record of anyone 70 years old and write down the first names.

Output: John, Marry, Bill

How test will be performed: The software application is presented with its main page. Testers are required to navigate through the software to use the dataview filtering functionality and filter out data as requested by the test. They are then asked to write down their responses on a piece of paper. Later, the papers are collected and compared for validation.

### 5.2.5 Performance

The following set of questions will ensure that the device is capable of meeting user expectations.

#### 1. **PT\_1: Data Transfer Time**

Type: Automatic, Dynamic

Initial State: The device has collected a complete set of data.

Input: The device is connected to the host software and connection is established for data transfer.

Output: Data transfer to finish within the specified parameter, TRASFER\_TIME.

How test will be performed: 2 group members of the Back End Developers will each perform data transfer tests 3 times and record the total transfer time for each execution. All 6 tests should result in execution time less than or equal to TRANSFER\_TIME.

#### 2. **PT\_2: Battery Life**

Type: Automatic, Static

Initial State: The device is turned on and left on idle mode.

Input: Testers wear the device for standard monitoring period.

Output: The total amount of time before the device runs out of battery is greater or equal to the parameter, BATTERY\_LIFE.

How test will be performed: 2 group members of the Back End Developers each wear the device for standard monitoring period under their normal lives. At the end of each calendar day, they are asked to record their response to a simple question: "Does the device still have battery life left? If so, how much percentage?".



### 5.2.6 Data Security Tests

These tests are designed to test the security of the system and the accessibility of important medical records. Ensuring that only certain people are able to access these records is a vital part of the design, and ensuring that records cannot be modified is also important for research data integrity.

#### 1. DSQT\_1: Records Safety Test

Control: Manual, Dynamic

Initial State: Device is powered on.

Input: Data transfer cable is plugged in from device to a generic PC, a tester tries to access the database and modify the data within.

Output: Database inaccessible and requests an administrative security key. The tester should be unable to access or modify the data.

Test Case Derivation: N/A

How test will be performed: Device will be given to a tester with technical know-how who does not have access to the administrator security key. They will attempt to access the database.

#### 2. DSQT\_2: Data Storage Selection Test

Control: Manual, Dynamic

Initial State: Device is powered on.

Input: Less than 3 seconds of limping from tester.

Output: Corresponding sensor data is not included in sdcard data.

Test Case Derivation: N/A

How test will be performed: One member of Back End Developers wears the device and pretends to "limp" for 3 seconds, to create garbage sensor data. Then microSD card is inserted in to a PC to examine the content.

### 5.3 Traceability Between Test Cases and Requirements

	DT1	DT2	DT3	DT4	MTT1	MTT2	PRT1	PRT2	TT1
<b>R1</b>	X	X	X	X					
<b>R2</b>					X	X			
<b>R3</b>							X	X	
<b>R4</b>									X
<b>R5</b>									
<b>R6</b>									
<b>NFR1</b>					X	X	X	X	
<b>NFR12</b>							X		

Table 5: Requirements Traceability Matrix Part 1

	TT2	TT3	TT4	TT5	DST1	DST2	DST3	DXT1	DXT2	DXT3
<b>R1</b>										
<b>R2</b>										
<b>R3</b>										
<b>R4</b>	X	X	X	X						
<b>R5</b>					X	X	X			
<b>R6</b>								X	X	X

Table 6: Requirements Traceability Matrix Part 2

	HST1	HST2	HST3	RT1	RT2	UT1	UT2	UT3	UT4	PT1	PT2	DSQT1
<b>NFR2</b>						X	X	X	X			
<b>NFR3</b>											X	
<b>NFR4</b>							X					
<b>NFR5</b>								X				
<b>NFR6</b>	X	X	X									
<b>NFR7</b>				X								
<b>NFR8</b>												X
<b>NFR9</b>						X			X			
<b>NFR10</b>						X			X			
<b>NFR11</b>	X	X	X	X								
<b>NFR13</b>										X		
<b>NFR14</b>												X

Table 7: Requirements Traceability Matrix Part 3

	HST1	HST2	HST3	PRT1	RT2	TT1	TT2	TT3	TT4	TT5
<b>HR1</b>	X	X	X							
<b>HR2</b>					X					
<b>SR1</b>				X						
<b>SR2</b>						X	X	X	X	X
<b>DSR1</b>										
<b>DSR2</b>										
<b>DSR3</b>										

Table 8: Hazard Analysis Traceability Matrix Part 1

	DST3	DSQT1	DSQT2
<b>HR1</b>			
<b>HR2</b>			
<b>SR1</b>			
<b>SR2</b>			
<b>DSR1</b>	X		
<b>DSR2</b>		X	
<b>DSR3</b>			X

Table 9: Hazard Analysis Traceability Matrix Part 2

## 6 Unit Test Description

Unit testing is a form of black box testing that examines specific functions of the code and their inputs and outputs. It is concerned about the "what" behaviour of the function as opposed to the behind-the-scene stories of "how" it is implemented.

### 6.1 Unit Testing Scope

Every single one of the functions were included for unit testing, nothing was left out for both hardware and software codes. In most cases, outputs require human interpretation. Thus, instead of using a test code, all programming-related outputs are to be read off of the serial communication monitor whenever applicable.

The [MIS](#) lists out all the modules and their use, syntax, and semantics.

### 6.2 Unit Testing Plan

The following tables list all the unit testing cases and lays out the template for the [Verification and Validation Report](#).

No.	Name	Ref.	Action	Expected Output
1	bed_HR_setup	<a href="#">R2</a>	Power on the device.	"PulseSensor object created", LED flash per each heartbeat.
2	bed_HR_detect	<a href="#">R2</a>	Walk for 10 seconds while wearing the device, then stop.	"HeartBeat Detected. BPM: 70".
3	bed_HR_detect	<a href="#">R2</a>	Power on, and shake the watch without anyone wearing it.	"HeartBeat Not Detected."
4	bed_MPU_setup	<a href="#">R2</a>	Power on the device.	"Calibrating MPU in 4 seconds! Hold it still! avg X acc: 20, , avg Y acc: 20, avg Z acc: 20"
5	bed_MPU_detect	<a href="#">R2</a>	Limp	"limp count + limp count + limp count + limp count + limp count + limping. Stepcount: 5"
6	bed_MPU_detect	<a href="#">R2</a>	Walk	"Stepcount: 13"
7	bed_MPU_detect	<a href="#">R2</a>	Stay still	"Stepcount: 0"
8	bed_init_display	<a href="#">NFR2</a>	No action	Black screen on

No.	Name	Ref.	Action	Expected Output
9	bed_scroll_test	NFR2	Scroll through 2 bezels in counter clock-wise direction.	Selection goes up.
10	bed_scroll_test	NFR2	Scroll through 2 non-adjacent bezels in counter clock-wise direction with 2 seconds of delay inbetween.	Selection goes up.
11	bed_scroll_test	NFR2	Scroll through 2 bezels in counter clock-wise direction.	Selection goes up.
12	bed_splash_screen	NFR2	Power on the device.	"Back End Developers"
13	bed_display_one_line	NFR2	Select a response for generated prompt	Screen clears prompt list and only displays current date and time.
14	OpenFile	R6	Press button and choose the "data.txt" file.	Contents of "data.txt" is displayed on UI.
15	bed_display_prompt	R3	Move around for 5 seconds and then stop for 5 seconds.	"Are you in pain?", Yes, No
16	bed_format_prompt	R3	Limp with the device to activate prompt generation.	Prompt text color is white while background is black.
17	OpenSerial	R6, NFR14	Connect device to computer via USB port.	UI displays green dot.
18	bed_init_rtc	R6, NFR1	Power on the device.	"March 5th, 16:21".
19	bed_display_info	NFR2	Power on the device.	"March 5th, 16:23".
20	bed_set_explicit_date_time	R6, NFR1	Disconnect power source, reconnect, and power on the device.	"March 5th, 16:46".
21	bed_touch_detect	NFR2	Touch 2 bezels in clockwise direction.	1
22	__init__	N/A	Run the host software.	Login page appears.
23	saveAsTxt	R6	Press Save as button and choose a directory to save file in.	Config.txt file created and saved.

No.	Name	Ref.	Action	Expected Output
24	BackToMain	NFR2	Press the main button.	Main screen appears and current screen closes.
25	InsertDB	R6, NFR14	Press save button.	Data appears on database table.
26	LoadDb	R6, NFR14	Press load button.	Data appears on software UI table.
27	dataHead	R6, NFR14	Press load button.	Data appears on software UI table.
28	search	R6, NFR14	Filter-search participant first name of "John"	All participant record with first name "John" appears.
29	OpenGraph	R7	Press graph button.	Empty graph chart appears.
30	OpenHeart	R7	Press heart rate button.	Only heart rate graph appears.
31	OpenSteps	R7	Press steps button.	Only number-of-steps graph appears.
32	OpenActivity	R7	Press activity button	Only activity graph appears.
33	plot	R7	Press steps button.	Only number-of-steps graph appears.
34	loginAttempt	NFR8, NFR14	ID: "admin", PW: "capstone"	Main screen appears.
35	loginAttempt	NFR8, NFR14	ID: "wr1ong", PW: "creden5tials"	"Wrong credentials!"
36	showRecord Window	NFR2	Press records button.	Records window appears and main window closes.
37	showCreate Records	NFR2	Press create records button.	Create Records window appears and main window closes.
38	showConfig View	NFR2	Press configuration button.	Configuration window appears and main window closes.
39	showDataView	NFR2	Press data view button.	Data view window appears and main window closes.

### 6.3 Tests for Functional Requirements

Section 6.2 outlines the relevant functional requirements per each module. The corresponding traceability matrix is included in section 6.5.

### 6.4 Tests for Nonfunctional Requirements

Section 6.2 outlines the relevant functional requirements per each module. The corresponding traceability matrix is included in section 6.5.

### 6.5 Traceability Between Unit Test Cases and Requirements

	R2	R3	R6	R7	NFR1	NFR2	NFR8	NFR14
1	X							
2	X							
3	X							
4	X							
5	X							
6	X							
7	X							
8						X		
9						X		
10						X		
11						X		
12						X		
13						X		
14			X					
15		X						

Table 10: Unit Test Traceability Matrix Pt 1

	R2	R3	R6	R7	NFR1	NFR2	NFR8	NFR14
16		X						
17			X					X
18			X		X			
19						X		
20			X		X			
21						X		
22								
23			X					
24						X		
25			X					X
26			X					X
27			X					X
28			X					X
29				X				
30				X				
31				X				
32				X				
33				X				
34							X	X
35							X	X
36						X		
37						X		
38						X		
39						X		

Table 11: Unit Test Traceability Matrix Pt 2



## References

- Back End Developers. Development plan. [https://github.com/zakerl/Capstone\\_Project/blob/main/docs/DevelopmentPlan/DevelopmentPlan.pdf](https://github.com/zakerl/Capstone_Project/blob/main/docs/DevelopmentPlan/DevelopmentPlan.pdf), 2022a.
- Back End Developers. Hazard analysis. [https://github.com/zakerl/Capstone\\_Project/blob/main/docs/HazardAnalysis/HazardAnalysis.pdf](https://github.com/zakerl/Capstone_Project/blob/main/docs/HazardAnalysis/HazardAnalysis.pdf), 2022b.
- Back End Developers. Problem statement & goals. [https://github.com/zakerl/Capstone\\_Project/blob/main/docs/ProblemStatementAndGoals/Team1\\_ProblemStatement%20%26%20Goals.pdf](https://github.com/zakerl/Capstone_Project/blob/main/docs/ProblemStatementAndGoals/Team1_ProblemStatement%20%26%20Goals.pdf), 2022c.
- Back End Developers. Reflection. [https://github.com/zakerl/Capstone\\_Project/blob/main/docs/Reflection/Reflection.pdf](https://github.com/zakerl/Capstone_Project/blob/main/docs/Reflection/Reflection.pdf), 2022d.
- Back End Developers. System requirements specification. [https://github.com/zakerl/Capstone\\_Project/blob/main/docs/SRS/SRS.pdf](https://github.com/zakerl/Capstone_Project/blob/main/docs/SRS/SRS.pdf), 2022e.
- Back End Developers. User guide. [https://github.com/zakerl/Capstone\\_Project/blob/main/docs/UserGuide/UserGuide.pdf](https://github.com/zakerl/Capstone_Project/blob/main/docs/UserGuide/UserGuide.pdf), 2022f.
- Back End Developers. Vnv plan. [https://github.com/zakerl/Capstone\\_Project/blob/main/docs/VnVPlan/VnVPlan.pdf](https://github.com/zakerl/Capstone_Project/blob/main/docs/VnVPlan/VnVPlan.pdf), 2022g.
- Hoang Pham. page 567–567. John Wiley & Sons, Inc, 1999.

## Appendix — Symbolic Parameters

The definition of the test cases will call for `SYMBOLIC_CONSTANTS`. Their values are defined in this section for easy maintenance.

`TRANSFER_TIME`: Maximum acceptable amount of time to transfer all data to the Host Software.

`BATTERY_LIFE`: Represents the minimum battery life requirement for the device.

Error	Description
<code>BED_ERR_NONE</code>	Represents no errors
<code>BED_ERR_INVALID_DATA</code>	Represents invalid data being used/stored
<code>BED_ERR_INVALID_DATA_SIZE</code>	Represents insufficient size for data storage
<code>BED_ERR_OUT_OF_BOUNDS</code>	Represents value of data past allowable limits
<code>BED_ERR_MEMORY_FULL</code>	Represents full internal memory buffer

Table 12: Table of Errors

## Appendix — Reflection

1. *What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.*

Approximately a third of this project's timeline has passed at this point. So far, team members have spent the vast majority of their time generating documentation for this project. In the process of doing so, the team has garnered a good sense of the learning styles, working styles, and habits of each individual team member.

Now that the planning phase has largely passed, now is the right moment to utilize this newfound knowledge about the team to maximum effect. Team #1 is lucky; each of its team members is deeply committed to achieving excellence in this project. As a result, they are willing to use knowledge about the strengths, weaknesses, and differences of each member in the aim of working as efficiently and effectively as possible. Synergy has become the greatest strength for Team #1, on top of the large collection of diverse skills held by Team #1's individual members.

Naturally, identifying the advantages of collaboration in Team #1 also sheds light on potential blind spots. For this specific team, these blind spots mostly rest within the "soft skills", as team members already are quite proficient with the tools and hard skills necessary to bring this project to fruition.

These soft skills are:

- **Jessica:** Critical Thinking
- **Oliver:** Engagement During Meetings
- **Jonathan:** Time Management
- **Anish:** Open-mindedness
- **Nish:** Leadership
- **Labeeb:** Communication

Developing these skills will be essential to the success of the project. It is the responsibility of each individual team member to work on their skills, but it is also necessary for the rest of the team to support them appropriately and make considerations to help fill any holes left behind.

2. *For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of*

*the identified approaches, which will each team member pursue, and why did they make this choice?*

- **Critical Thinking:**

- (a) *For Jessica:* Consider the "why" for each important decision made in the project.
- (b) *For the Team:* Host design review sessions regularly to look back and determine whether or not the project is headed in the right direction.

- **Engagement During Meetings:**

- (a) *For Oliver:* Chair a third of all meetings going forward.
- (b) *For the Team:* Assign a portion of every meeting to questions and feedback, without any other tasks preempting this section.

- **Time Management:**

- (a) *For Jonathan:* Set up automated reminders and schedules for capstone related tasks.
- (b) *For the Team:* Parcel work into sections in which the workload required is easily understood and planned for.

- **Open-mindedness:**

- (a) *For Anish:* Attach equivalent value to conflicting ideas, and weigh objective pros and cons.
- (b) *For the Team:* Actively promote "devil's advocate" mindsets when making important decisions.

- **Leadership:**

- (a) *For Nish:* Ensure input at least once into every major decision made by the team.
- (b) *For the Team:* Whenever making an important decision, ask each individual member for their opinion.

- **Communication:**

- (a) *For Labeeb:* Raise every concern that comes to mind. Regardless of size or importance.
- (b) *For the Team:* Create a section on the Trello board specifically for non-objective related concerns.