Project Title: 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

November 1, 2022

1 Revision History

Date	Version	Notes
Date 1	1.0	Notes
Date 2	1.1	Notes

Contents

List of Tables

[Remove this section if it isn't needed —SS]

List of Figures

[Remove this section if it isn't needed —SS]

2 Symbols, Abbreviations and Acronyms

Symbol	Description
DT	Duration Test
MTT	Minor Tracking Test
PT	Prompt Test
TT	Threshold Test
DST	Data Storage Test
DXT	Data Extraction Test
HC#	Hardware Constraint #

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

[symbols, abbreviations or acronyms — you can simply reference the SRS (**?**) tables, if appropriate —SS]

[Remove this section if it isn't needed —SS]

This document ... [provide an introductory blurb and roadmap of the Verification and Validation plan —SS]

3 General Information

3.1 Summary

[Say what software is being tested. Give its name and a brief overview of its general functions. —SS]

3.2 Objectives

[State what is intended to be accomplished. The objective will be around the qualities that are most important for your project. You might have something like: "build confidence in the software correctness," "demonstrate adequate usability." etc. You won't list all of the qualities, just those that are most important. —SS]

3.3 Relevant Documentation

[Reference relevant documentation. This will definitely include your SRS and your other project documents (design documents, like MG, MIS, etc). You can include these even before they are written, since by the time the project is done, they will be written. —SS]

?

4 Plan

[Introduce this section. You can provide a roadmap of the sections to come. —SS]

4.1 Verification and Validation Team

[Your teammates. Maybe your supervisor. You shoud do more than list names. You should say what each person's role is for the project's verification. A table is a good way to summarize this information. —SS]

4.2 SRS Verification Plan

[List any approaches you intend to use for SRS verification. This may include ad hoc feedback from reviewers, like your classmates, or you may plan for something more rigorous/systematic. —SS]

[Maybe create an SRS checklist? —SS]

4.3 Design Verification Plan

[Plans for design verification —SS]

[The review will include reviews by your classmates —SS]

[Create a checklists? —SS]

4.4 Verification and Validation Plan Verification Plan

[The verification and validation plan is an artifact that should also be verified. —SS]

[The review will include reviews by your classmates —SS]

[Create a checklists? —SS]

4.5 Implementation Verification Plan

[You should at least point to the tests listed in this document and the unit testing plan. —SS]

[In this section you would also give any details of any plans for static verification of the implementation. Potential techniques include code walkthroughs, code inspection, static analyzers, etc. —SS]

4.6 Automated Testing and Verification Tools

[What tools are you using for automated testing. Likely a unit testing framework and maybe a profiling tool, like ValGrind. Other possible tools include a static analyzer, make, continuous integration tools, test coverage tools, etc. Explain your plans for summarizing code coverage metrics. Linters are another important class of tools. For the programming language you select, you should look at the available linters. There may also be tools that verify that coding standards have been respected, like flake9 for Python. —SS]

[If you have already done this in the development plan, you can point to that document. —SS]

[The details of this section will likely evolve as you get closer to the implementation. —SS]

4.7 Software Validation Plan

[If there is any external data that can be used for validation, you should point to it here. If there are no plans for validation, you should state that here. —SS]

[You might want to use review sessions with the stakeholder to check that the requirements document captures the right requirements. Maybe task based inspection? —SS]

[This section might reference back to the SRS verification section. —SS]

5 System Test Description

5.1 Tests for Functional Requirements

[Subsets of the tests may be in related, so this section is divided into different areas. If there are no identifiable subsets for the tests, this level of document structure can be removed. —SS]

[Include a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. —SS]

[It would be nice to have a blurb here to explain why the subsections below cover the requirements. References to the SRS would be good here. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

5.1.1 Device should stay on during the monitoring period

The following tests will check the 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, dates from the past or too far into the future beyond what the battery life can sustain (HC2).

Duration Test

1. **DT_1: Regular Inputs**

   Control: Manual

   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 the Monitoring Period and ensuring that the device maintains power throughout this period.

2. **DT_2: Invalid Inputs**

   Control: Manual

   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

   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

   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 are 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

   Initial State: Device is worn with all systems working.

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

   Output: Status returned by the Sensor Array is no error (BED_ERR_NONE).

   How test will be performed: The test is performed by strapping the device onto a test volunteer who will perform the tracked activities at a normal/regular pace. This is done to ensure that the device can work under normal scenarios.

2. **MTT_2: Slow Movement**

   Control: Manual

   Initial State: Device is worn with all systems working.

Input: Wearer performs activities at a very slow pace.

Output: Status returned by the Sensor Array is no error (BED_ERR_NONE).

How test will be performed: The test is performed by strapping the device onto a test volunteer who will perform the tracked activities at a very slow pace. This is done to ensure that the device can work under scenarios in which users have limited mobility.

5.1.3 Device prompts user when activity is detected

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

Prompt Tests

1. **PT_1: Test Prompt**

   Control: Manual

   Initial State: Device has just been reconfigured.

   Input: Device is turned on for the first time.

   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, the user should receive a test prompt that will confirm that the prompting system is working correctly.

2. **PT_2: Activity Prompt**

   Control: Manual

   Initial State: Device is in the idle state.

   Input: Activity has been detected.

   Output: Specific activity prompt is displayed.

   How test will be performed: The test is done by having a test volunteer perform one of the activities that are registered. This should result in a prompt for the volunteer that is generated based on the specific activity performed.

5.1.4 Customizable Thresholds

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

Threshold Tests

1. **TT_1: Regular Inputs**

   Control: Manual

   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.

2. **TT_2: Below lower limit**

   Control: Manual

   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

   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

   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

   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

   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

   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 is performed by performing a registered activity and ensuring that the prompt generated is answered and its result is stored in the internal storage buffer.

3. **DST_3: Sensor Storage**

   Control: Manual

   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 is performed by performing a registered activity 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 can be extracted and presented in a graphical manner deemed acceptable for the purpose of EMA analysis.

1. **DXT_1: Extracting Regular Data**

Control: Manual

Initial State: Device is connected to the device manager.

Input: 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

Initial State: Device is connected to the device manager.

Input: 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 No Data**

Control: Manual

Initial State: Device is connected to the device manager.

Input: 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

[The nonfunctional requirements for accuracy will likely just reference the appropriate functional tests from above. The test cases should mention reporting the relative error for these tests. Not all projects will necessarily have nonfunctional requirements related to accuracy —SS]

[Tests related to usability could include conducting a usability test and survey. The survey will be in the Appendix. —SS]

[Static tests, review, inspections, and walkthroughs, will not follow the format for the tests given below. —SS]

5.2.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

5.2.2 Area of Testing2

...

5.3 Traceability Between Test Cases and Requirements

[Provide a table that shows which test cases are supporting which requirements. —SS]

6 Unit Test Description

[Reference your MIS (detailed design document) and explain your overall philosophy for test case selection. —SS] [This section should not be filled in until after the MIS (detailed design document) has been completed. —SS]

6.1 Unit Testing Scope

[What modules are outside of the scope. If there are modules that are developed by someone else, then you would say here if you aren't planning on verifying them. There may also be modules that are part of your software, but have a lower priority for verification than others. If this is the case, explain your rationale for the ranking of module importance. —SS]

6.2 Tests for Functional Requirements

[Most of the verification will be through automated unit testing. If appropriate specific modules can be verified by a non-testing based technique. That can also be documented in this section. —SS]

6.2.1 Module 1

[Include a blurb here to explain why the subsections below cover the module. References to the MIS would be good. You will want tests from a black box perspective and from a white box perspective. Explain to the reader how the tests were selected. —SS]

1. test-id1

Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

Initial State:

Input:

Output: [The expected result for the given inputs —SS]

Test Case Derivation: [Justify the expected value given in the Output field —SS]

How test will be performed:

2. test-id2

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input:

   Output: [The expected result for the given inputs —SS]

   Test Case Derivation: [Justify the expected value given in the Output field —SS]

   How test will be performed:

3. ...

6.2.2 Module 2

...

6.3 Tests for Nonfunctional Requirements

[If there is a module that needs to be independently assessed for performance, those test cases can go here. In some projects, planning for nonfunctional tests of units will not be that relevant. —SS]

[These tests may involve collecting performance data from previously mentioned functional tests. —SS]

6.3.1 Module ?

1. test-id1

   Type: [Functional, Dynamic, Manual, Automatic, Static etc. Most will be automatic —SS]

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

6.3.2 Module ?

...

6.4 Traceability Between Test Cases and Modules

[Provide evidence that all of the modules have been considered. —SS]

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

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

7.2 Usability Survey Questions?

[This is a section that would be appropriate for some projects. —SS]

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

Appendix — Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete the verification and validation of your project? Examples of possible knowledge and skills include dynamic testing knowledge, static testing knowledge, specific tool usage etc. You should look to identify at least one item for each team member.

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?