Project Title: System Verification and Validation Plan for 4TB6 - Mechatronics Capstone Smart Bike Lock

Abi Nevo
Elsa Bassi
Steffi Ralph
Abdul Iqbal
Stephen De Jong
Anthony Shenouda

November 1, 2022

1 Revision History

Date	Version	Name	Notes
20-10-22	1.0	Steffi	Section 2

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
SRS	Software Requirements Specification
FR	Functional Requirements
NFR	Nonfunctional Requirements
LC	Likely Changes
ULC	Unlikely Changes
SC	System Contraints
A	Assumptions
MV	Monitored Variables

REFERENCE TO SRS FOR OTHER TABLES + ADD CAPTION

[symbols, abbreviations or acronyms – you can simply reference the SRS (Author, 2019) tables, if appropriate —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 (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]

Author (2019)

4 Plan

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

4.1 Verification and Validation Team

[You, your classmates and the course instructor. Maybe your supervisor. You shoud do more than list names. You should say what each person's role is for the project. 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 just be ad hoc feedback from reviewers, like your classmates, or you may have something more rigorous/systematic in mind.. —SS]

[Remember you have an SRS checklist —SS]

4.3 Design Verification Plan

[Plans for design verification —SS]

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

[Remember you have MG and MIS checklists —SS]

4.4 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]

4.5 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]

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

4.6 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]

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. —SS]

5.1.1 Area of Testing1

[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. If a section covers tests for input constraints, you should reference the data constraints table in the SRS. —SS]

Title for Test

1. test-id1

   Control: Manual versus Automatic

   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

   Control: Manual versus Automatic

   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:

5.1.2 User Input Related

1. test-id1: FR8: Effective Bike Lock: The lock is sturdy and cannot be manually opened by the average human once engaged.

   Control: Manual

   Initial State: Lock is engaged, and bike is locked.

   Input/Condition: A prying, pulling, kicking, etc. force between 200-400N.

   Output/Results: A pass/fail as well as a score from 1-4 for the following cases; a fail if the lock disengages and breaks, a fail if the lock disengages, a pass if the lock stays engaged but breaks, and a pass if the lock system can stay engaged with out breaking.

   How test will be performed: A test group of 2-3+ adults, completing 50+ trials, each giving the required input forces.

2. test-id1: FR9: Lock must only be engaged/disengaged by the intended user(s).

   Control: Manual

   Initial State: Lock is engaged, and bike is locked.

   Input/Condition: Iphone Bluetooth signal.

   Output/Results: A fail if they are able to connect to the smart bike lock and disengage it and a pass if they can't connect and disengage the lock.

   How test will be performed: A test group will attempt to connect via Bluetooth from their smartphones.

5.1.3 Bike Input Related

1. test-id1: FR10: The lock can be mounted to the bike's frame.

   Control: Manual

   Initial State: Lock is in disengaged state and not mounted to a bike.

   Input/Condition: Users force to mount the SmartLock.

   Output/Results: A pass/fail for each bike if the lock is/isn't able to mount in to the bike as intended. A score from 0-the number of bikes tested will also be given.

   How test will be performed: Users will attempt to mount the Smart-Lock onto 1-3 bikes each from 3-5 categories of bikes including (adult standard/hybrid, road, mountain, kids standard/hybrid).

5.1.4 Output Related

1. FR11: Battery percentage must be shown on the phone app.
   Type: Manual

   Initial State: Phone on with App downloaded and open

   Input/Condition: observation of app

   Output/Results: user must be able to view the battery percentage of the lock

2. FR12: Location (coordinates) of bike must be shown on the app as BikePosition.
   Type: Manual

   Initial State: Phone on with App downloaded and open

   Input/Condition: observation of app

   Output/Results: user must be able to view the saved coordinates

3. FR13: Battery must output enough power to engage the lock.
   Type: Manual

   Initial State: Locking mechanism disengaged, electro magnet has ability to get power from battery (functional circuit)

   Input/Condition: supply power to electromagnet

Output/Results: lock mechanism is engaged

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. —SS]

[Tests related to usability could include conducting a usability test and survey. —SS]

5.2.1 Area of Testing1: Smart Phone

Title for Test:

1. test-id1: NFR1: Can be used by people who speak any native language.

   Type: Manual

   Initial State: Lock is engaged and phone is on with App downloaded and open.

   Input/Condition: Non-English-speaking test user has no prior knowledge of how to use the System.

   Output/Results: The lock is successfully engaged and the bike is securely locked by test user.

   How test will be performed: User attempts to disengage the mechanism through the App, attach bike to external frame and re-engage mechanism.

2. test-id2: NFR2: Can reasonably be used without requiring an instruction manual.

   Type: Manual

   Initial State: Lock is engaged and phone is on with App downloaded and open.

   Input/Condition: A group of test users with no prior knowledge of how to use the System.

Output/Results: The lock is successfully engaged, and the bike is securely locked by all test users.

How test will be performed: Users are told to disengage the mechanism through the App, attach the bike to an external frame and re-engage the mechanism without any instruction. The time required for full usage per test is measured using a stopwatch and must below ten minutes. Learning period is subsequently calculated and plotted.

3. test-id3: 3. NFR3: App storage under 50 megabytes. A small mobile app should not take significant space on the user's phone.

   Type: Static

   Initial State: App development IDE is open and up to date.

   Output/Results: App storage is less than 50 megabytes.

   How test will be performed: App storage displayed on the IDE is kept below the upper threshold of 50 megabytes.

5.2.2 Area of Testing: Physical Lock

1. NFR4: The design must be visually appealing.
   Type: Manual

   Initial State: Locking mechanism assembled

   Input/Condition: Survey users on their opinions of the visual appeal of the device

   Output/Results: Visual appeal is rated 7 or higher (on a scale of 1-10)

2. NFR5: The lock must not impede normal bike functions.
   Type: Manual

   Initial State: Locking mechanism assembled and mounted on bike frame

   Input/Condition: Normal bike operation (ride forward, turn right, turn left, ride downhill, ride uphill)

   Output/Results: Device does not impede bicycle functioning

3. NFR6: The lock must be waterproofed to withstand normal rainfall. And NFR7: The lock must be waterproofed to withstand normal splashing while riding.

   Test 1: empty housing (no electronics inside)

   Type: Manual

   Initial State: housing assembled

   Input/Condition: simulate rain: hold housing under shower for 5 minutes, slowly rotating

   Output/Results: inside of housing is completely dry

   Test 2: full mechanism

   Type: Manual

   Initial State: Locking mechanism, electronics, and housing assembled, microcontroller on

   Input/Condition: simulate rain: hold housing under shower for 5 minutes, slowly rotating

   Output/Results: device remains functional and operational

4. NFR8: Accuracy of bike lock status must be above 95%.
   Type: Manual

   Initial State: Locking mechanism, electronics, and housing assembled

   operational and within range

   Input/Condition: Engage/disengage lock x100

   Output/Results: Ensure engage/disengage status matches current physical state

5.2.3 Usability

Title for Test:

1. test-id1: NFR11: iPhone app locking must be quicker to use than a typical keyed/combo bike lock.

   Control: Manual

Initial State: The Smart lock is in the transport state with the test user standing beside the bike at a bike rack.

Input/Condition: User engaging and disengaging the SmartLock.

Output/Results: Scores from 1-10 will be given for weighted time from fasted to slowest.

How test will be performed: Users will be timed on locking and unlocking. As a group of 3 people will, remove lock from the transport state, and engage the lock as intended. To lock they will disengage the lock as intended and convert to transport state. This will be done with 3-5 types of locks including, SmartLock, keyed, and combination.

2. test-id1: NFR12: Opening and closing lock must require similar force to a typical keyed/combo.

   Control: Manual

   Initial State: The Smart Lock will be in transport state with the test user standing beside the bike at a bike rack.

   Input/Condition: Users Engaging and Disengaging the SmartLock.

   Output/Results: Scores from 1-5 will be given from most to least amount of relative force required.

   How test will be performed: Users will measure force while locking and unlocking. As a group of 3 people will, remove lock from the transport state, and engage the lock as intended. To lock they will disengage the lock as intended and convert to transport state. This will be done with 3-5 types of locks including, SmartLock, keyed, and combination.

3. test-id1: NFR13: Battery must last for greater than 1 month and/or 60 rides before needing to be replaced or charged.

   Control: Manual

   Initial State: The Smart Lock is fully charged

   Input/Condition: Users Engaging, Locating, and Disengaging the SmartLock.

   Output/Results: The amount of time and quantity of lock/unlocks of the Smart Lock. A pass if it meets the required number.

How test will be performed: A group of users will take turns to bike to a new bike lock, lock the bike, mark the bikes location in the app, unlock, and then repeat. This will be done until the battery dies.

4. test-id1: NFR14: Batteries must be accessible to replace or chargeable.

   Control: Manual

   Initial State: The Smart Lock is in the Transport state.

   Input/Condition: Users will charge/replace batteries as intended.

   Output/Results: Pass if the batteries can be charged/replaced as intended.

   How test will be performed: Users will charge/replace batteries as intended.

5. test-id1: NFR15: The lock must be easily mounted on the bike frame. It does not require special tools, (i.e., those not found in a typical toolbox, such as power tools), to be installed and does not take more than twnety minutes to install.

   Control: Manual

   Initial State: The System is ready to be installed.

   Input/Condition: A group of test users with an average understanding of how to use typical tools. They have access to those typical tools.

   Output/Results: The System is successfully mounted on the bike frame without special tools for each test user.

   How test will be performed: Each user attempts to install the System, following the procedure outlined in the instructions. This procedure will be developed following the completion of the first stage of prototyping.

6. test-id2: NFR16: The lock can be used for many different models of mountain, city, and road bikes.

   Control: Manual

   Initial State: The System is ready to be installed.

Input/Condition: A group of test users with an average understanding of how to use typical tools. They have access to those typical tools. Three different models of bike are tested; one for each type (mountain, city and road).

Output/Results: The System can be mounted on all three bike models successfully by each test user.

How the test will be performed: Three users attempt to install the System, following the procedure outlined in the instructions.

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 and explain your overall philosophy for test case selection. —SS] [This section should not be filled in until after the MIS 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]

References

Author Author. System requirements specification. https://github.com/..., 2019.

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:

1.

2.