Verification and Validation Plan for SmartLock
4TB6 - Mechatronics Capstone

Team #5, Locked & Loaded
Abi Nevo, nevoa
Elsa Bassi, bassie
Steffi Ralph, ralphs1
Abdul Iqbal, iqbala18
Stephen De Jong, dejons1
Anthony Shenouda, shenoa2

April 11, 2023

Table 1: Revision History

Date	**Developer(s)**	**Change**
20-10-22	Steffi	Section 2, 3, 4 & Reflection
01-11-22	Stephen, Elsa, Abi, & Anthony	Section 5 & Reflection
02-11-22	Abdul	Section 4 & Reflection
19-11-22	Steffi	Updates for Consistency
03-03-23	Abi	Revisions to ensure consistency with SRS, TA and peer feedback
06-03-23	Elsa	Further revisions to ensure consistency
05-04-23	Abi	Rev 1 Updates

Contents

List of Tables

1 Symbols, Abbreviations and Acronyms

Refer to Section 1 of the SRS documentation for a full reference section on units, symbols and abbreviations/acronyms.

This document will outline the verification and validation plans we have created to prove that the SmartLock device is a successful product. The tests mentioned in this document will validate our product against the various requirements which we have outlined in the SRS document. At the completion of the plan in this document, we will have the knowledge required to form an iterative design process which improves into a successful device at its completion.

2 General Information

2.1 Summary

The SmartLock can be broken down into two main components, the software (ie. the smartphone application) and the hardware (ie. the physical lock).

The smartphone application, SmartLock, will be a basic UI that our users can interact with to send a signal to unlock/lock the locking mechanism, check in on the battery of the device, and remember where they left their bike when they locked it.

The physical locking device will be used to secure the wheels to the bike frame, which can also be connected to an external mount.

These components will be tested through various methods, to determine if they meet all the requirements outlined in the SRS.

2.2 Objectives

The objective that our project hopes to accomplish is to create a simple-to-use smartphone app that allows users to lock the most important features of their bike(s) with confidence and to allow them to find their bike(s) with ease.

2.3 Relevant Documentation

For more information on the project breakdown, planning or delivery refer to the following documentation: SRS, HA, MG, and MIS.

3 Plan

In this section, various aspects of the verification plan will be illustrated explaining the process from concept to implementation. Beginning with an outline of the Validation team and their roles in guiding the process, this section will cover the verification of our SRS documentation as well as how we plan to design, implement and test our problem and eventually prototype our solution. It also covers any automation tools used as well as a software validation plan.

3.1 Verification and Validation Team

Table 2: Verification and Validation Team Table

VnV Team Members	Role
Group 5 Members	Responsible for quality control on their topic of expertise, keeping the other group members aware of any changes or progress and reflecting that knowledge in the documentation. System tests outlined below will be completed by the full group, together.
Gr 5: Abi Nevo and Steffi Ralph	Unit Testing on Microcontroller
Gr 5: Stephen De Jong and Elsa Bassi	Unit Testing on Mechanical Lock and Lock Frame
Gr 5: Anthony Shenouda	Unit Testing on Mobile App
Dr. Sirouspour	From our supervisor, Dr. Sirouspour, we are looking for technical feedback on components that we might not have considered or fully grasped as well as feedback on any design, materials or documentation considerations
Classmates	Peer review from the perspective of someone who knows what is required of the documentation
Nicholas Annable	Provide specific feedback and grades on the work we have completed so that we can take the appropriate action moving further into the project
Dr. Smith	Provide information and guidance on the project goals/deliverables and documentation requirements

3.2 SRS Verification Plan

Our plan for verification of the SRS includes the following steps:

1. Keep the Document Alive

By keeping the document alive we will be able to continually update any requirements/constraints/objectives that adapt as our project comes to life. This will ensure that all the documentation is accurate.

2. Peer Review

   We will use the peer reviews to verify our work and to look for inconsistencies or aspects that we didn't consider.

3. TA Review

   The TA review will be used to validate what we have done and what needs to be reworked as the project progresses.

4. SRS Checklist

   With every update of the SRS document and progression of the SmartLock we will continue to reference the SRS Checklist to ensure that we continue to meet all the requirements and produce complete documentation.

3.3 Design Verification Plan

Our plan for verification of the Design Plan includes the following steps:

1. Review with Supervisor (Dr. Sirouspour)

   As a novice design team, we will use the meetings with Dr. Sirouspour to review, primarily the hardware components of, our design and understand if we are using the right technical pieces and if our approach can lead to success.

2. Proof of Concept Demo

   The proof of concept demo will be a crucial part of our validation plan. With the intent to be able to demonstrate both a portion of our physical component and a preliminary software UI, the presentation and feedback will provide us with an opportunity to validate our objectives and verify the next steps and problem areas.

3. Testing

   Testing, which will be discussed in detail below, will be used to understand if our requirements can be met given the scope of this

project and help us to determine what needs to change for our final product.

4. MIS & MG Checklists

   The MIS Checklist and the MG Checklist will be used to review the tests that are created and check that they follow the guidelines so that we can properly test the work product. Tests will be reviewed and updated as the project progresses.

3.4 Implementation Verification Plan

Our plan to verify the implementation is outlined below:

1. Functional Requirements Verification

   The verification of our functional requirements will be done by performing the system tests discussed in Tests for Functional Requirements. The success rate of these system tests will help the team to confirm and validate these requirements and therefore will provide the team with good metrics on the core functionality of the product. Note that each requirement references, and is directly mapped to, at least one requirement, showing that these test cases robustly cover the defined requirements in the SRS. Please see the Traceability Table in this document for detailed traceability.

2. Non-Functional Requirements Verification

   The verification of our non-functional requirements will be done by performing the system tests discussed in Tests for Nonfunctional Requirements. The success rate of these system tests will help the team to confirm and validate these requirements and therefore will provide the team with good metrics on the overall quality and class of the product. This will help us in verifying other aspects of the product including the App features, durability and ease of use. Note that each requirement references, and is directly mapped to, at least one requirement, showing that these test cases robustly cover the defined requirements in the SRS. Please see the Traceability Table in this document for detailed traceability.

3. Code Implementation Verification

   In order to verify the implementation of our code base, we will make use of both code walk-throughs and inspections by fellow peers as well as by performing unit tests on our system. Having a peer review done would not only illuminate any shortcomings in our implementation but also help us in identifying more efficient techniques within our code base. By performing unit tests, which are discussed in Unit Test Description, we can verify different modules within our system to see if each is performing as expected and is robust. This will help us maintain best practices while implementing our code base.

3.5 Automated Testing and Verification Tools

The tools used for automated testing and verifying coding standards are outlined below:

1. flutter_lints

   flutter_lints will be used to lint the source code for the mobile application on Flutter.

2. CodeFactor

   CodeFactor will be used for automated code analysis on our repo on Github.

3. Github

   Github will be used to host our codebase including all staging branches and the production branch. All changes will need to be approved and reviewed before merging to production.

4. Git

   Git will be used for version control, resolving merge conflicts and interfacing with different branches.

3.6 Software Validation Plan

The above-mentioned automated tools will be utilized for validating our software and code implementation. Sample user profiles will be created within our smartphone application in order to perform the system tests outlined in detail further in this document. This will allow the team to validate different modules within our software and test their functionality and success rate.

4 System Test Description

4.1 Tests for Functional Requirements

The tests in this section will be used to confirm and validate our functional requirements from the SRS document. Completing these tests will prove the functionality of our product. Note that each requirement references, and is directly mapped to, at least one requirement, showing that these test cases robustly cover the defined requirements in the SRS.

4.1.1 Area of Testing: User Input Related

1. DisengageLock:

   FR1: LockDisengage input must disengage the lock on the bike.

   Control: Automatic

   Initial State: Bike lock is engaged

   Input/ Condition: Signal from App to disengage the lock

   Output/ Results: Pass/fail if the lock is successfully disengaged

How the test will be performed: The bike lock will be engaged. The user will use the App to disengage the lock. If successful, the lock becomes disengaged.

2. LockLocation:

   FR2: Location (coordinates) of user's phone must be able to be saved in the smartphone application as UserPosition.

   Control: Automatic

   Initial State: No location

   Input/ Condition: The user engages the Bike lock using the App

   Output/ Results: Pass/fail if the App places a geotag within 10 metres of the Users current location

   How the test will be performed: The Bike lock will be engaged on the App. The App will be monitored to ensure the geotag is stored and correctly placed on the App.

3. EffectiveLock

   FR3: 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 (which is the average human's arm strength

   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 without breaking.

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

4. EffectiveLockSimulation

   FR3: Effective Bike Lock: The lock is sturdy and cannot be manually opened by the average human once engaged.

   Control: Manual

Initial State: Lock frame and mechanism simulated in CAD.

Input/Condition: Perform strength/durability simulation on lock frame and mechanism in CAD.

Output/Results: A pass/fail if the simulation meets the 200-400 N threshold.

How the test will be performed: CAD simulation

5. Security

   FR4: 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 the test will be performed: A test group will attempt to connect via Bluetooth from their smartphones.

4.1.2 Area of Testing: Bike Input Related

1. LockMount

   FR5: The lock can be mounted to the bike's frame.

   Control: Manual

   Initial State: The lock is in the disengaged state and not mounted to a bike.

   Input/Condition: Users apply force to mount the SmartLock.

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

   How the test will be performed: Users will attempt to mount the SmartLock onto 1-3 bikes each.

4.1.3 Area of Testing: Output Related

1. BatteryPercentage

   FR6: Battery percentage must be shown on the phone app.

   Control: Manual

   Initial State: Phone on with App downloaded and open

   Input/Condition: Observation of the app

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

   How the test will be performed: Apply input condition, observe the expected output.

2. LocationOnApp

   FR7: Location (coordinates) of the bike must be shown on the app as BikePosition.

   Control: Manual

   Initial State: Phone on with App downloaded and open

   Input/Condition: Observation of the app

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

   How the test will be performed: apply input condition, observe the expected output

3. PowerOutput

   FR8: Battery must output enough power to engage the lock.

   Control: Manual

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

   Input/Condition: Supply power to the electromagnet

   Output/Results: Locking mechanism is engaged

   How the test will be performed: Apply input condition, observe the expected output

4.2 Tests for Nonfunctional Requirements

This subset of tests will be used to validate the nonfunctional requirements of our product. Completing these tests will prove various aspects of our product's needs. these aspects include smartphone app features, physical design attributes, accuracy, and the usability of our product. Note that each requirement references, and is directly mapped to, at least one requirement, showing that these test cases robustly cover the defined requirements in the SRS.

4.2.1 Area of Testing: Smart Phone

1. LimitedInstructions

   NFR1: Can reasonably be used without requiring an instruction manual.

   Control: Manual

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

2. AppStorage

   NFR2: App storage under 50 megabytes. A small mobile app should not take up significant space on the user's phone.

   Control: Static

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

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

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

4.2.2 Area of Testing: Physical Lock

1. VisualAppeal

   NFR3: The design must be visually appealing.

   Control: Manual

   Initial State: Locking mechanism assembled

   Input/Condition: Survey users on their opinions of the visual appeal of the device; see Usability Survey Questions.

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

   How the test will be performed: Apply input condition, survey 50 users and observe the expected output.

2. NormalBikeFunction

   NFR4: The lock must not impede normal bike functions.

   Control: Manual

   Initial State: Locking mechanism assembled and mounted on the bike frame

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

   Output/Results: The device does not impede bicycle functioning

   How the test will be performed: apply input condition, observe the expected output

3. Safety

   NFR5: The design must not inflict harm to the user in any way, such as clamping down on a finger, or moving at a force or speed that could cause injury.

   Initial State: Locking mechanism assembled and mounted on the bike frame

   Input/Condition: Attempt to put a finger in crevices where the user may be harmed. Engage/Disengage the lock.

   Output/Results: User is not harmed or pinched.

   How the test will be performed: apply input condition, observe the expected output

4.2.3 Area of Testing: Accuracy

1. BatteryAccuracy

   NFR6: Battery percentage must be calculated accurately within 10%.

   Control: Manual

   Initial State: Full device assembled, battery percentage calculated is 1%.

   Input/Condition: Engage/disengage the lock until the battery dies.

   Output/Results: The number of lock engages possible with 1% battery matches our specification for the number of lock engages possible with 100% battery (multiply the measured # of lock engages possible with 1% battery by 100), within 10% accuracy.

   How the test will be performed: Apply input condition and observe the expected output

4.2.4 Area of Testing: Usability

1. QuickLock

   NFR7: The SmartLock must be quicker to use than a typical keyed or combination bike lock.

   Control: Manual

   Initial State: The SmartLock 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 fastest to slowest.

   How the test will be performed: Users will be timed on locking and unlocking. A group of 3 people will, remove the lock from the transport state, and engage the lock as intended. To lock they will disengage the lock as intended and convert to the transport state. The "transport state" is how the lock is situated/positioned as the user is actively riding their bike. This will be done with 3-5 types of locks including, SmartLock, keyed, and combination.

2. UseForce

   NFR8: Opening and closing lock must require similar force to a typical keyed/combo.

   Control: Manual.

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

3. BatteryLife

   NFR9: Battery must last for greater than 1 month and/or 60 rides before needing to be replaced or charged.

   Control: Manual

   Initial State: The SmartLock 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 SmartLock. A pass if it meets the required number.

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

4. ComponentAccessibility

   NFR10: Batteries and other internal components must be accessible to replace and/or chargeable.

   Control: Manual.

Initial State: The SmartLock is in the Transport state.

Input/Condition: Users will charge/replace batteries as intended. Users are open the SmartLock to access internal components.

Output/Results: Pass if the batteries can be charged/replaced, and internal components can be accessed, as intended.

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

5. NoSpecialTools

   NFR11: 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 twenty 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 the 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. BikeVersatility

   NFR12: 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 bikes 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 each attempt to install the System on three different types of bike, following the procedure outlined in the instructions.

7. AppOS

   NFR17: The App should run on iOS and Android.

   Control: Manual

   Initial State: The System is ready to be installed.

   Input/Condition: Load and open the app on an iOS and Android machine.

   Output/Results: App operates as expected.

   How the test will be performed: Perform the input condition and observe outcome to ensure actual results match expected results.

4.3 Traceability Between Test Cases and Requirements

Table 3: Traceability Table

Test Case	Functional Requirement(s)	Non-Functional Requirement(s)
DisengageLock	FR1	
LockLocation	FR2	
EffectiveLock, EffectiveLockSimulation	FR3	
Security	FR4	
LockMount	FR5	
BatteryPercentage	FR6	
LocationOnApp	FR7	
PowerOutput	FR8	
LimitedInstructions		NFR1
AppStorage		NFR2
VisualAppeal		NFR3
NormalBikeFunction		NFR4
Safety		NFR5
BatteryAccuracy		NFR6
QuickLock		NFR7
UseForce		NFR8
BatteryLike		NFR9
ComponentAccessibility		NFR10
NoSpecialTools		NFR11
BikeVersatility		NFR12
AppOS		NFR17

Note: NFR 13-16 and 18 were deemed out of scope on the SRS, and thus are not present in this Verification and Validation Plan.

5 Unit Test Description

5.1 Physical Locking Mechanism

1. Locking Mechanism

   Setup: Test only the isolated physical locking mechanism–manually move the small locking pin and large pin with your hands.

   Input: Move the small pin and large pin into "locked" and "unlocked" positions (in and out).

   Expected Result: When small pin in "locked" position, large pin cannot be pulled out.

5.2 Microcontroller

1. Arduino Bluetooth Connection

   Unit: setup() in disengageControl.ino

   Setup: Flash Arduino with disengageControl.ino

   Input: Connect to Arduino using the Bluetooth Development App "nRF Connect"

   Expected Result: Prints message to console stating that a central device is connected.

2. Arduino Disengage Signal Output

   Unit: loop() in disengageControl.ino

   Setup: Flash Arduino with disengageControl.ino

   Input: Connect to Arduino using the Bluetooth Development App "nRF Connect", send the correct password.

   Expected Result: Green onboard LED turns on, TRANSISTOR_OUT pin has HIGH voltage.

3. Arduino Incorrect Password Signal Output

   Unit: loop() in disengageControl.ino

   Setup: Flash Arduino with disengageControl.ino

Input: Connect to Arduino using the Bluetooth Development App "nRF Connect", send an incorrect password.

Expected Result: Green onboard LED is off, TRANSISTOR_OUT pin has LOW voltage.

5.3 Circuit

1. Circuit Transistor Test

   Setup: Complete the circuit documented in the System Design document, but substitute the Arduino for a power source, and the solenoid with an LED.

   Input: Connection/disconnection of the power source to the transistor gate.

   Expected Result: When high power signal is transmitted to the gate of the transistor from power source, LED connected to drain of transistor turns on.

2. Circuit Solenoid Test

   Setup: Complete circuit documented in System Design document, but substitute the Arduino for a power source.

   Input: Connection/disconnection of the power source to the transistor gate.

   Expected Result: When high power signal is transmitted to gate of transistor from power source (NOT Arduino), solenoid connected to drain of transistor is enabled.

3. Full Circuit Test

   Setup: Complete circuit documented in System Design document.

   Input: Send correct disengage signal to Arduino from "nFR Connect"

   Expected Result: When high power signal is transmitted to gate of transistor from Arduino, solenoid connected to drain of transistor is enabled.

5.4 Mobile App

1. App Bluetooth Connection

   Unit: device.connect() in device.dart

   Setup: Search for bluetooth devices using the FindDevicesScreen class in main.dart

   Input: Connect to Arduino using the device name "SmartLock" as seen from the FindDevicesScreen

   Expected Result: Prints message to console stating that a central device is connected.

2. App Disengaging Lock

   Unit: e.characteristics[0].write([password]) in device.dart

   Setup: Connect to Arduino using the device.connect() function in device.dart

   Input: Write the password user inputed to Arduino

   Expected Result: If password is correct, lock is disengaged, otherwise lock is engaged

3. App Battery Display

   Unit: getBattery() in main.dart

   Setup: Read the battery information using the readCounter() function in device.dart

   Input: The battery information in the text file 'counter.txt'

   Expected Result: The remaining battery life for the Smartlock displayed as a percentage on the user screen.

4. App Current Location Display

   Unit: getCameraPosition() in main.dart

   Setup: Read the coordinate information using the readLat() and readLog() functions in device.dart

   Input: After the user pushes the current location button, the coordinate information in the text file 'locationLat.txt' and 'locationLog.txt' are collected.

Expected Result: The current location of the user is displayed with coordinates on the main screen.

5.5 Traceability Between Test Cases and Modules

Table 4: Unit Test Traceability Table

Unit Test	**Module**
Locking Mechanism	Locking Mechanism
Arduino Bluetooth Connection	Arduino Bluetooth Communication
Arduino Disengage Signal Output	Hardware Disengage
Arduino Incorrect Password Signal Output	Hardware Disengage
Circuit Transistor Test	Solenoid Actuation
Circuit Solenoid Test	Solenoid Actuation
Full Circuit Test	Solenoid Actuation

6 Appendix A

6.1 Usability Survey Questions

1. Visual Appeal: Survey Question

   Rate this device's visual appeal on a scale of 1-10, with 10 being the most visually appealing, and 1 being the least visually appealing.

6.2 User Testing Questions

1. How long does it take you to open the lock? Is a 15-second window too short? Too long?

2. Is the app intuitive to use? Would you change any features?

7 Appendix B

7.1 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.

   Abi: In order to perform verification, specifically on accuracy, we need to be able to have an expected output to which we can compare our real output. This is especially applicable for the battery life. We need to first be able to calculate an expected battery life to then validate it, which is a skill we will need to acquire.

   Stephen: We will need critical thinking skills to approach our device in such a way that we can complete a set of tests which have been created critically and will prove our product's design and implementation. We will need technical knowledge of how our device works such that these validation tests will be a complete set.

Elsa: The Team must build our collective knowledge in the area of structural analysis of engineering designs. In order to be able to effectively evaluate our designs for efficacy, robustness, longevity and usefulness, the Team will need to acquire or reacquire engineering skills related to how to assess the structural integrity of mechanical devices.

Steffi: A skill we will need to acquire to successfully complete the verification and validation of the project is to be able to be accepting of critique and critical of our choices during the design process. In order to truly be successful we will need to adapt to the information provided to us and not believe that there is only one way to perform our task otherwise we might overlook the best solution.

Abdul: In terms of verifying and validating the lock mechanism, the team will need to consult with someone who is an expert in locks and their security. In addition to that, acquiring some structural knowledge from industry experts would be an asset specifically when verifying the structural integrity of our prototype.

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?

   Abi: One approach to learning this skill would be doing research on the internet, and another might be to ask for the help of an expert. We are not familiar with someone knowledgeable on this topic, and therefore we will be learning how to calculate battery life through internet research.

   Stephen: To acquire critical thinking skills we will need to think outside the box and ask others their thoughts so that we can gain multiple opinions and a different way of thinking. To acquire the technical knowledge about our device, many hours of research as well as talking with industry professionals will be needed.

   Elsa: To acquire or reacquire skills related to the structural analysis of mechanical designs, the Team can review previous coursework in Materials related to stress and strain. We can also take advantage of McMaster resources for physical testing, including online databases and professors who have experience in the subject.

   Steffi: One approach to this is to look at how other designs have

adapted dramatically to feedback and look out for ideas that might influence our design similarly. The method that I believe we will use is to fully discuss the implications of any design changes and how they will play out so we can make educated decisions on how to move forward.

Abdul: Coding in a new language is definitely not easy, however with the right research and open-source documentation, we will be able to implement coding best practices as well as get a better understanding of any libraries that we can use. In terms of testing, taking advantage of any automated tools as well as talking to potential users and identifying their needs and getting feedback would be really insightful during our testing phases. Another approach during our testing phase would be to talk to industry experts and compare quality control standards to our product.