
# System Verification and Validation Plan

Cyclops Ride Assist: Real-time bicycle crash detection and blindspot monitoring.

**Team 9**
Aaron Li (lia79)
Amos Cheung (cheuny2)
Amos Yu (yua25)
Brian Le (leb7)
Manny Lemos (lemosm1)

# Table Of Contents


- 1. Revision History
- 2. Symbols, Abbreviations, and Acronyms
- 3. General Information
  - 3.1. Purpose
  - 3.2. Scope
  - 3.3. Relevant Documentation
- 4. Plan
  - 4.1. Verification and Validation Team
  - 4.2. SRS Verification Plan
  - 4.3. Design Verification Plan
  - 4.4. Verification and Validation Plan Verification Plan
  - 4.5. Implementation Verification Plan
  - 4.6. Automated Testing and Verification Tools
  - 4.7. Software Validation Plan
- 5. System Test Description
  - 5.1. Tests for Functional Requirements
    - 5.1.1. Vehicle Detection Test
    - 5.1.2. Crash Detection Test
    - 5.1.3. Video Logging Test
    - 5.1.4. Full SD Test
    - 5.1.5. Resume Detection Test
    - 5.1.6. Front Light On Test
    - 5.1.7. Front Light Off Test
    - 5.1.8. Power On System Test
    - 5.1.9. Power Off System Test
  - 5.2. Tests for Nonfunctional Requirements
    - 5.2.1. Appearance and Style Test
    - 5.2.2. Hardware Ease of Use Test
    - 5.2.3. Software Ease of Use Test
    - 5.2.4. Learning Test
    - 5.2.5. Understandability Test
    - 5.2.6. Accessibility Test
    - 5.2.7. Power Response Speed Test
    - 5.2.8. Video Upload Speed and Latency Test
    - 5.2.9. Crash Detection Speed and Latency Test
    - 5.2.10. Car Detection Speed and Latency Test
    - 5.2.11. External Safety-Critical Test
    - 5.2.12. CRA Emission Safety-Critical Test
    - 5.2.13. CRA Decimal Precision Test
    - 5.2.14. CRA Accelerometer Accuracy Test
    - 5.2.15. CRA Camera Time Accuracy Test
    - 5.2.16. Reliability and Availability Test
    - 5.2.17. Robustness and Fault-Tolerance Test
    - 5.2.18. Max Storage Capacity Test

        - 5.2.19. Low Battery Capacity Test
        - 5.2.20. Scalability and Extensibility Test
        - 5.2.21. Longevity Test
        - 5.2.22. Physical Environment Test
        - 5.2.23. Interfacing with Adjacent Systems Test
        - 5.2.24. Productization and Release Test
        - 5.2.25. Interfacing with Adjacent Systems Test
        - 5.2.26. Access Security Test
        - 5.2.27. Privacy Test
        - 5.2.28. Cultural Test
        - 5.2.29. Compliance Test
        - 5.2.30. Standards Test
- 6. Unit Tests
    - 6.1. Unit Testing Scope
    - 6.2. Test for Functional Requirements
    - 6.3. Test for Nonfunctional Requirements
    - 6.4. Traceability Between Test Cases and Modules
- 7. Appendix
    - 7.1. Symbolic Parameters
    - 7.2. Usability Survey Questions for Non Functional Tests
- 8. Reflection


## List of Tables


- Table 1.1: Revision History


## List of Figures

## 1. Revision History

Table 1.1: Revision History

| Date | Developer(s) | Change |
|---|---|---|
| October 25, 2022 | Aaron Li, Amos Cheung, Amos Yu, Brian Le, Manny Lemos | Document created |

## 2. Symbols, Abbreviations, and Acronyms

| Name | Description |
|---|---|
| CRA | Abbreviation of Cyclops Ride Assist. |
| Cyclist | A person who operates a bicycle as a means of transportation. |
| User | A person who will operate the final product. See Cyclist. |
| Client | See user. |
| SRS | Software Requirements Specification |

# 3. General Information

## 3.1. Purpose

The purpose of this document is to describe the testing, validation, and verification processes which will be carried out on the CRA system. Testing instills confidence that software and hardware function as expected independently, and interface correctly.

## 3.2. Scope

This validation and verification document lays the foundation for testing the CRA system. The requirements for correct system functionality presented in the SRS document will be extended upon to include specific, quantifiable metrics. Success in the testing described will verify that CRA has met these requirements in a measurable, meaningful way.

Specifically, this document will address testing pertaining to both the software and hardware of CRA. Out of scope testing would include testing for requirements not covered in the SRS document.

## 3.3. Relevant Documentation

SRS Document

# 4. Plan

## 4.1. Verification and Validation Team

There will be two distinct subgroups of the verification and validation team.

Group one will be responsible for any hardware testing. This includes testing of the CRA housing, mount, circuitry, sensors, and video feeds. This group will be composed of Amos Cheung, Aaron Li, and Manny Lemos.

Group two will be responsible for software testing. This includes testing of the CRA crash detection, video logging, and vehicle blindspot detection. This group will be composed of Brian Le and Amos Yu.

## 4.2. SRS Verification Plan

The requirements set about in the SRS document will be directly linked to test cases using the Traceability Between Test Cases and Requirements table found in section 5.3.

The SRS document has been verified as correct and complete using a number of methods. The Cyclops team has discussed the continued validity of this document and its requirements in team meetings. Further, peer reviews have been conducted on the document, concerns were raised, and issues were resolved.

## 4.3. Design Verification Plan

Our design verification plan to fulfill modular testing requirements is composed of two distinct sections; software unit testing, and modular hardware testing.

Software unit testing will be performed using pytest. As new modules of code are developed, or existing modules are updated and features are changed, thorough black box and white box testing will be conducted

to verify correct functionality.

Modular hardware testing will occur as discrete components of the hardware are completed. This means that components such as the bicycle mount, will be tested independently of the housing to verify that the component functions as expected. This will act as a baseline for assumed hardware functionality, and will enable the CRA team to identify sources of hardware failure more efficiently.

### 4.4. Verification and Validation Plan Verification Plan

This document will be extensively reviewed by group members as project development progresses. The continued validity of this document will be maintained, and any missing information will be rectified. Moreover, peer reviews will be conducted. If any issues are presented, they will be addressed and resolved.

### 4.5. Implementation Verification Plan

In order to successfully implement the CRA verification plan, a series of static and dynamic testing techniques will be used. At a high level, requirements specifications set about in the SRS document will be linked with the test cases laid out in the VnV plan and design documents. This framework of documents will then be subject to static testing. Specifically, this process will involve reviewing the aforementioned documents to provide a detailed scope of exactly what CRA aims to achieve as compared to what the testing of requirements will validate. Static testing will rigorously verify that the requirements for CRA are fulfilled by the code if testing is successful.

Reaching beyond documentation, the software code will be subject to static testing. Manual reviews of code will be conducted by team members of CRA. Specifically, a team member who did not write a module under review will be responsible for the review. This structure aims to promote writing understandable well commented code which can be reviewed by a relatively unbiased third party. Further, system-wide static testing will also be used in the form of Pylint. This automated analysis tool will enforce coding standards and enable us to identify potential sources of issues.

The functionality of the systems hardware and software will be tested dynamically as a complete product. This testing will take the form of manual testing. The CRA team will mount the system to a bicycle and simulate a variety of behaviours with expected outcomes. The true outcomes of these cases, along with other characteristics of the system’s response such as reaction time, will be analyzed and compared with the expected results.

### 4.6. Automated Testing and Verification Tools

All automated testing of software will be conducted using Pytest. All automated code analysis will be conducted using Pylint.

### 4.7. Software Validation Plan

To validate that the software fulfills all of the correct requirements, we will continually amend and improve upon the testing methodologies used, and the components tested. The aim of this endeavor is to ensure that testing is directly aligned with any updates to the SRS document and any other documents which are directly tied to software requirements.

## 5. System Test Description

## 5.1. Tests for Functional Requirements

### 5.1.1. Vehicle Detection Test

| **CFRST1** | |
|---|---|
| Control | Manual (Dynamic) |
| Initial State | Application is running with CRA mounted on moving bike |
| Input | Read video input |
| Output | LED blindspot indicator |
| Test Case Derivation | LED's should light up and be reflective of whether a car is located in the riders blind spot |
| How will test be performed | This test will be done dynamically in a real world environment where the bike will be moving at a constant speed with the presense or absence of a vehhicle in the blind spot |
| Requirements Referenced | CFR1, CFR2, CFR6, CFR10, CFR12 |

### 5.1.2. Crash Detection Test

| **CFRST3** | |
|---|---|
| Control | Manual (Dynamic) |
| Initial State | Accelerometer input with CRA mounted to moving bike |
| Input | Accelerometer electrical input caused by bike movement |
| Output | Video log/clip |
| Test Case Derivation | Cyclops should detech when a crash has occured when the acceleration of the rider has passed a certian threshold. This then prompts the CRA to perform video logging task inresponse to the crash |
| How will test be performed | The test will be done dynamically in a real world environment in which the accelerometer unit will be mounted onto the bike and updates values as the bike experiences a large change in its acceleration (a crash). The SD card will then be verified if a clip has been logged |
| Requirements Referenced | CFR3, CFR4, CFR5, CFR7 |

### 5.1.3. Video Logging Test

| **CFRST4** | |
|---|---|
| Control | Manual (Dynamic) |
| Initial State | Application is running with CRA mounted onto a bike |

| **CFRST4** | |
|---|---|
| Input | Crash detection system, front and reach video input |
| Output | Video log to be stored on a local SD |
| Test Case Derivation | CRA will log a snipped(last buffer_time_minutes minutes_ of the vfront feed when the user has crashed) |
| How will test be performed | The test will be done manually in a real world environment in which a test bike with the unit mounted will go through a simulated crash to trigger the crash procedure |
| Requirements Referenced | CFR7, CFR8 |

#### 5.1.4. Full SD Test

| **CFRST5** | |
|---|---|
| Control | Manual (Static) |
| Initial State | Application should be running, SD card should be full or near full capacity |
| Input | SD card storage/space remaining |
| Output | SD_storage_full LED whether the SD card is too full for video logging |
| Test Case Derivation | CRA shall have a seperate LED to signify that the current card is full. The CRA should only be able to run when the SD card has ample storage space |
| How will test be performed | The system shall be turned on when a full/near full SD card in which the tester would then look to confirm that the LED prompts to empty out SD card |
| Requirements Referenced | CFR9, CFR10 |

#### 5.1.5. Resume Detection Test

| **CFRST6** | |
|---|---|
| Control | Manual (Static) |
| Initial State | CRA is mounted on a bike with the application is running and in its crashed state |
| Input | N/A |
| Output | N/A |
| Test Case Derivation | CRA should be expected to resume its vehicle and crash detection after a previous crash just been detected and logged |
| How will test be performed | Monitor the system after a crash state. The system is looked at to see if it continues to take in front and rear video feed and perform its vehicle and crash detection |

| **CFRST6** | |
|---|---|
| Requirements Referenced | CFR12, CFR13 |

#### 5.1.6. Front Light On Test

| **CFRST7** | |
|---|---|
| Control | Manual (Dynamic) |
| Initial State | CRA is mounted on bike and light is set to off |
| Input | Flash light button |
| Output | Front light emitted |
| Test Case Derivation | The front light should be turned on when the user prompts it |
| How will test be performed | The test will be done dynamically in a real world environment in which the front light will be toggled from off ot on. The test will be looking for whether the light gets emitted |
| Requirements Referenced | CFR14 |

#### 5.1.7. Front Light Off Test

| **CFRST8** | |
|---|---|
| Control | Manual (Dynamic) |
| Initial State | CRA is mounted on bike and light is set to on |
| Input | Flash light button |
| Output | Front light turned off |
| Test Case Derivation | The front light should be turned off when the user prompts it |
| How will test be performed | The test will be done dynamically in a real world environment in which the front light will be toggled from on to off. This test will be looking for whether the light gets cut |
| Requirements Referenced | CFR14 |

#### 5.1.8. Power On System Test

| **CFRST9** | |
|---|---|
| Control | Manual (Dynamic) |
| Initial State | CRA is mounted on a bike and the system/isPoweredLED is off |

| CFRST9 | |
|---|---|
| Input | Power button used to power on/off the system |
| Output | LED used to represent if the system is powered on |
| Test Case Derivation | LED's should light up and be show that the system/application has power and is running |
| How will test be performed | This test will be done dynamically in a real world environment where the power button is physically pressed to turn on CRA |
| Requirements Referenced | CFR11 |

#### 5.1.9. Power Off System Test

| CFRST10 | |
|---|---|
| Control | Manual (Dynamic) |
| Initial State | CRA is mounted on a bike and system/isPOweredLED is on |
| Input | Power button used to power on/off the system |
| Output | isPoweredLED used to represent if the system is powered on |
| Test Case Derivation | LED's should turn off to reflect that the system/application has been powered down |
| How will test be performed | This test will be done dynamically in a real world environment where the pwoer button is physicall pressed to turn off CRA |
| Requirements Referenced | CFR11 |

## 5.2. Tests for Nonfunctional Requirements

#### 5.2.1. Appearance and Style Test

| CNFRST1 | |
|---|---|
| Control | Manual, Static |
| Initial State | CRA has been created. |
| Input | The user will view the CRA. |
| Output | CRA and associated software application is considered appealing, fitted, safe, and unintrusive. |
| Test Case Derivation | Stakeholders and developers will view the CRA and software application and give a response using a Usability Survey. |

| **CNFRST1** | |
|---|---|
| How will test be performed | A visual inspection test will be given to identify whether or not the CRA and software meets our requirements (as outlined below). A Usability Survey with a score of 10 with a needed average score of 80% will be required to satisfy the test. The user will also be asked if they would change the design or if they would want to add their own customizations. |
| Requirements Referenced | CNFR1, CNFR 2, CNFR3, CNFR4, CNFR9, CNFR12, CNFR47 |

### 5.2.2. Hardware Ease of Use Test

| **CNFRST2** | |
|---|---|
| Control | Manual, Static |
| Initial State | CRA is currently not mounted on the bicycle. |
| Input | The user will view and handle the CRA's mechanical system. |
| Output | Placing the CRA onto the bicycle with minimal effort. |
| Test Case Derivation | The CRA should be able to be mounted onto the bicycle as it will be constructed in a way that reduces the amount of potential human error. The CRA will also allow the user to place the hardware system wherever they please (excluding the cameras) to increase customization. |
| How will test be performed | The CRA will be given to various users and stakeholders. The function of the CRA will be stated and we will allow the user to place the CRA onto the bicycle. The developers and testers will run twenty recorded timed tests and the average will be taken. |
| Requirements Referenced | CNFR5, CNFR7, CNFR8, CNFR9, CNFR14 |

### 5.2.3. Software Ease of Use Test

| **CNFRST3** | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA software application is closed. |
| Input | The user will press the System On button. |
| Output | The CRA software application should open and be ready for use after the system loads. |
| Test Case Derivation | The CRA software application will need to open right after loading the system to ensure that the user will be able to use the CRA as soon as possible. |
| How will test be performed | The developers will test at different time intervals and button presses to ensure that the system is working as intended. Meetings will be used to address concerns. The testers will ask the users for feedback using a survey. |

| CNFRST3 | |
|---|---|
| Requirements Referenced | CNFR6, CNFR7, CNFR8 |

### 5.2.4. Learning Test

| CNFRST4 | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA will be powered off and mounted onto the bicycle properly. |
| Input | The user will have an instruction manual with all added functionalities of the CRA including startup, shutdown etc. |
| Output | The user will be able to operate the CRA to its full extent. |
| Test Case Derivation | The CRA must be easy to use for any user and can be easily re-created. |
| How will test be performed | The instruction manual will be given to various stakeholders and people to try to use the CRA. A Usability Survey will be used to gauge the effectiveness on a scale of 10 with a passing grade of 80%. |
| Requirements Referenced | CNFR10, CNFR11, CNFR41 |

### 5.2.5. Understandability Test

| CNFRST5 | |
|---|---|
| Control | Manual, Static |
| Initial State | The instruction manual will be provided and software application of the CRA will be turned on. |
| Input | A user will view the information on the instruction manual and software application. |
| Output | The user will be able to understand the language, words, symbols. |
| Test Case Derivation | The manual and system will be designed in a way that uses non-technical language and is easily interpreted by a regular user with zero prior experience. |
| How will test be performed | The instruction manual and view of the software system will be given to various stakeholders and people to try to understand the functions of the CRA. A Usability Survey will be used to gauge the effectiveness on a scale of 10 with a passing grade of 80%. |
| Requirements Referenced | CNFR5, CNFR6, CNFR7, CNFR8, CNFR41 |

### 5.2.6. Accessibility Test

| **CNFRST6** | |
|---|---|
| Control | Manual, Dynamic |
| Initial State | CRA will be mounted onto the bicycle and powered on. |
| Input | A vehicle is detected. |
| Output | An audiovisual cue to let the user know of the vehicle. |
| Test Case Derivation | Similar to CFRST1, CRA will be able to output an LED light to alert the user. We will run this test multiple times. |
| How will test be performed | The test will be the same as CFRST1. The test will be run twenty different times. |
| Requirements Referenced | CNFR13 |

### 5.2.7. Power Response Speed Test

| **CNFRST7** | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA will be mounted onto the bicycle with the power button off. |
| Input | The power button will be pressed. |
| Output | The CRA will turn on within RESPONSE_TIME_MILLISECONDS as outlined in our SRS. |
| Test Case Derivation | The lower the value, the faster a user will be able to use the CRA quickly and be on their way which will improve satisfaction for the user. |
| How will test be performed | The system will be powered on and the amount of time to start the system will be measured. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR15 |

### 5.2.8. Video Upload Speed and Latency Test

| **CNFRST8** | |
|---|---|
| Control | Manual, Dynamic |
| Initial State | The CRA system power will be on. |
| Input | The CRA will create a video. |
| Output | The video will be uploaded to the external storage device. |
| Test Case Derivation | We want to see that the video is uploaded to the external storage device with a max time of UPLOAD_TIME_SECONDS as defined in the SRS. |

| CNFRST8 | |
|---|---|
| How will test be performed | The time between creating and uploading a video will be timed using a stopwatch. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR16 |

### 5.2.9. Crash Detection Speed and Latency Test

| CNFRST9 | |
|---|---|
| Control | Manual, Dynamic |
| Initial State | The CRA system power will be on and will be mounted onto the bicycle. |
| Input | The acceleration of the bicycle drops quickly as seen in CFRST3. |
| Output | The CRA will keep the recording of the video before and after the collision as outlined in CNFR17 and this will be determined within 1s. |
| Test Case Derivation | The CRA should be able to detect if there is a crash within 1s to allow the user to have the needed footage. |
| How will test be performed | A crash will be simulated to test and a stopwatch will be used to measure the time taken. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR17 |

### 5.2.10. Car Detection Speed and Latency Test

| CNFRST10 | |
|---|---|
| Control | Manual, Dynamic, Functional |
| Initial State | The CRA software detection will be loaded and running. |
| Input | Cars and other objects will be placed in the view of the camera. |
| Output | The software will correctly detect vehicles vs other objects. |
| Test Case Derivation | We need to make sure that the blindspot detection is for vehicles and not stationary objects such as a fire hydrant or tree. |
| How will test be performed | We will be running the module by itself and implementing it into the system to run further tests. A score of 80% on our learned software will suffice. The score is displayed when we run the test. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR18 |

### 5.2.11. External Safety-Critical Test

| **CNFRST11** | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA has been created. |
| Input | The CRA will be placed on the vehicle. |
| Output | The CRA will not cause any external damage to the bicycle and the wires will not protrude into the bicycle workings. |
| Test Case Derivation | We will make sure that there is no external damage to the bicycle and that the wires will not affect or become tangled to ensure safety. |
| How will test be performed | Users will place the CRA onto their own bicycles. Once placed on, the developers and testers will verify that the CRA is not hindering any of the bicycle's functions. We will ask for feedback from the user on if it hinders their riding as well. |
| Requirements Referenced | CNFR21, CNFR23 |

### 5.2.12. CRA Emission Safety-Critical Test

| **CNFRST12** | |
|---|---|
| Control | Manual, Dynamic, Functional |
| Initial State | The CRA has been powered on. |
| Input | The CRA is in use by a user. |
| Output | The CRA will not emit any emissions to the atmosphere. |
| Test Case Derivation | This is to ensure that the CRA is emission-friendly and environmentally safe. Chemicals harmful to the environment will not be used. |
| How will test be performed | The CRA will be run twenty different times and will be checked for any sounds that could be a leak and checked for any possible burning. Any signs of degradation will be noted and presented in a meeting. |
| Requirements Referenced | CNFR22 |

### 5.2.13. CRA Decimal Precision Test

| **CNFRST13** | |
|---|---|
| Control | Manual, Dynamic, Functional |
| Initial State | The CRA is powered on. |
| Input | A component of the CRA will be run that includes measurements. |
| Output | The component will have all decimal places rounded to three (3) decimal places. |

| **CNFRST13** | |
|---|---|
| Test Case Derivation | This is to ensure that all our components will have the same number of decimals which will allow for relative precision with each other. |
| How will test be performed | Different components will have tests run in parallel including the speed, time (milliseconds), etc. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR24 |

### 5.2.14. CRA Accelerometer Accuracy Test

| **CNFRST14** | |
|---|---|
| Control | Manual, Dynamic, Functional |
| Initial State | The CRA will be present and mounted on the bicycle. |
| Input | The accelerometer will be given force so that αx != 0 and/or αy != 0 and/or αz != 0 and CRA will have a velocity |
| Output | The accelerometer will generate values and the speed reading will be accurate within 1km/h. |
| Test Case Derivation | We need to be able to determine the speed of the CRA and other vehicles. |
| How will test be performed | The CRA will be run at a certain speed with another external speedometer and accelerometer to determine accuracy. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR25 |

### 5.2.15. CRA Camera Time Accuracy Test

| **CNFRST15** | |
|---|---|
| Control | Manual, Dynamic, Functional |
| Initial State | The camera and the CRA are set up correctly. |
| Input | The acceleration of the CRA stops abruptly and becomes αx = 0 and/or αy = 0 and/or αz = 0 |
| Output | The camera should retrieve the video information of BUFFER_TIME_MINUTES within 1 second. |
| Test Case Derivation | We need to be sure to save all the footage that is required for the user. |

| CNFRST15 | |
|---|---|
| How will test be performed | A crash will be simulated and the delay between the “crash” and output video will be noted. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR26 |

#### 5.2.16. Reliability and Availability Test

| CNFRST16 | |
|---|---|
| Control | Manual, Dynamic, Functional |
| Initial State | The SD card is either empty or has footage located on the device and is loaded onto the CRA. |
| Input | A video is uploaded to the SD card. |
| Output | The CRA will allow for another video to be uploaded immediately. |
| Test Case Derivation | This is to make sure that CRA can continuously run for the user in the event of more than one accident. |
| How will test be performed | A video will be uploaded to the CRA and immediately after, the CRA will be stopped once again. A video will be created once again. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR27, CNFR28 |

#### 5.2.17. Robustness and Fault-Tolerance Test

| CNFRST10 | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA has been created and is powered on or off. |
| Input | The CRA is accidentally dropped from the bicycle. |
| Output | The CRA will continue working as intended. |
| Test Case Derivation | The CRA must be built to withstand accidents anywhere and at anytime (under any realistic condition or weather) |
| How will test be performed | A drop test will be performed along with an air pressure test to determine if the CRA is sealed properly. Furthermore, any possible water damage from rain, puddles, etc. will be tested. The developers and testers will run twenty recorded tests and the average will be taken. |

| CNFRST10 | |
|---|---|
| Requirements Referenced | CNFR29 |

### 5.2.18. Max Storage Capacity Test

| CNFRST18 | |
|---|---|
| Control | Manual, Dynamic |
| Initial State | The CRA will be powered on. |
| Input | The CRA will run as usual with a near-full (90%) SD card. |
| Output | The CRA will alert the user with a notification to let them know that the system is full. |
| Test Case Derivation | The CRA must have enough capacity to continue storing footage of any incidents. |
| How will test be performed | The SD card will be filled with videos or pictures. The CRA will start and an alert will show on the software application. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR19, CNFR30 |

### 5.2.19. Low Battery Capacity Test

| CNFRST19 | |
|---|---|
| Control | Manual, Dynamic |
| Initial State | The CRA will be powered on. |
| Input | The CRA will run as usual with a near-empty (20%) battery. |
| Output | The CRA will alert the user with a notification to let them know that the battery is running out. |
| Test Case Derivation | The CRA must have enough battery capacity to keep the whole system running. |
| How will test be performed | The battery will be depleted to 20%. Multiple batteries and types of batteries will be used.. The battery will be placed into the console and an alert will be awaited upon. The developers and testers will run twenty recorded tests and the average will be taken. |
| Requirements Referenced | CNFR20 |

### 5.2.20. Scalability and Extensibility Test

| CNFRST20 | |
|---|---|

| CNFRST20 | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA will be created. |
| Input | The CRA has all components connected. |
| Output | The CRA will run as expected with extra storage and software for additional and extended components. |
| Test Case Derivation | The system needs to be developed with the future in mind to ensure that the CRA can be used by clients further down the road. |
| How will test be performed | Tests will include a Usability Survey to see how users would want to expand the CRA or if they have concerns with their own bicycles. |
| Requirements Referenced | CNFR31, CNFR32 |

#### 5.2.21. Longevity Test

| CNFRST21 | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA will be created. |
| Input | The CRA has all components connected. |
| Output | The CRA will run as expected for a certain period of time before maintenance or cleaning is required. |
| Test Case Derivation | This will ensure that the product is able to last for at least a year before requiring maintenance. |
| How will test be performed | Constant use will allow the development team to see when the expected maintenance will be. This will be a continuous test. The product will also be distributed to users and feedback will be gathered through a survey where it will be discussed in a meeting. |
| Requirements Referenced | CNFR33 |

#### 5.2.22. Physical Environment Test

| CNFRST22 | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA is present and mounted on the bicycle |
| Input | The bicycle is outdoors on the road. |
| Output | The CRA will work as intended with all desired functionalities. |

| CNFRST22 | |
|---|---|
| Test Case Derivation | Ensure the CRA is workable outside and not just inside for testing. |
| How will test be performed | The developers will use the CRA outside to ensure that it is working as intended. The developers and testers will run twenty recorded tests. |
| Requirements Referenced | CNFR34, CNFR35 |

#### 5.2.23. Interfacing with Adjacent Systems Test

| CNFRST23 | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA will be created. |
| Input | The CRA will have all components connected and will be connected to an external system like a PC. |
| Output | The CRA will work as intended. |
| Test Case Derivation | The CRA will work with all types of PCs, OS, software to ensure all can use the system. |
| How will test be performed | The CRA will be connected to various computer systems, adapters, PCs, laptops from different users. A pass would be if the PC is able to read the external device. The developers and testers will run twenty recorded tests. |
| Requirements Referenced | CNFR36, CNFR42 |

#### 5.2.24. Productization and Release Test

| CNFRST24 | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA software and items list will be created. |
| Input | The CRA software and items listed on Github as a public repository. |
| Output | The software and hardware can be easily found and purchased if a user would like to develop it further. |
| Test Case Derivation | The CRA is a publicly available open-source system. |
| How will test be performed | Check using different Github accounts across different browsers to ensure that the repository is public and all required information is present. |

| CNFRST24 | |
|---|---|
| Requirements Referenced | CNFR37, CNFR38 |

### 5.2.25. Interfacing with Adjacent Systems Test

| CNFRST25 | |
|---|---|
| Control | Manual, Dynamic, Functional |
| Initial State | The CRA will be powered on. |
| Input | The CRA will run as usual. |
| Output | The CRA will output errors to a crash log file. |
| Test Case Derivation | The CRA will tell the developers through crash logs how the software or hardware failed so that the developers can easily diagnose and fix the issue. |
| How will test be performed | Small errors will be recreated (unplugged, missing hardware, software bug, etc). Multiple iterations of these issues will be run and the crash logs will be checked each time. |
| Requirements Referenced | CNFR39, CNFR40 |

### 5.2.26. Access Security Test

| CNFRST26 | |
|---|---|
| Control | Manual, Static |
| Initial State | N/A |
| Input | The CRA’s external storage device card is connected to the user’s PC. |
| Output | The PC will allow the user to view the files and videos required. |
| Test Case Derivation | This is to ensure that the CRA does not encrypt or corrupt files when uploading. |
| How will test be performed | The SD card will be placed into different PCs, laptops and tested to see if it is readable. The developers and testers will run twenty recorded tests. |
| Requirements Referenced | CNFR43 |

### 5.2.27. Privacy Test

| CNFRST27 | |
|---|---|
| Control | Manual, Dynamic |
| Initial State | CRA is powered on. |

| **CNFRST27** | |
|---|---|
| Input | The video will be placed onto the SD card through the Raspberry Pi |
| Output | The video will only be stored locally on the SD card |
| Test Case Derivation | This is to ensure that the Raspberry Pi does not upload to the Internet or the cloud. |
| How will test be performed | A video will be uploaded onto the SD card through the Raspberry Pi. Using an external adapter and PC, the video will be viewed locally. |
| Requirements Referenced | CNFR44, CNFR45 |

### 5.2.28. Cultural Test

| **CNFRST28** | |
|---|---|
| Control | Manual, Static |
| Initial State | The CRA will be on. An instruction manual will be present. |
| Input | The software and instruction manual will be opened. |
| Output | The software and instruction manual are in English (Canadian) |
| Test Case Derivation | The developers will create in English so it should not be changed to another language. |
| How will test be performed | The CRA will be powered on and the manual will be opened by developers. |
| Requirements Referenced | CNFR46 |

### 5.2.29. Compliance Test

| **CNFRST29** | |
|---|---|
| Control | Manual, Static |
| Initial State | N/A |
| Input | The CRA will be created. |
| Output | The CRA will follow as closely to the average bicycle standard for weight and size restrictions. |
| Test Case Derivation | CRA will adhere to all types of bicycles. |
| How will test be performed | Various types of bicycles across different brands will be observed to check for possible restrictions. A survey will be done with a group of twenty users with varying types of bicycles. |

| **CNFRST29** | |
|---|---|
| Requirements Referenced | CNFR48 |

#### 5.2.30. Standards Test

| **CNFRST30** | |
|---|---|
| Control | Manual, Static |
| Initial State | N/A |
| Input | The CRA will be created. |
| Output | The CRA will comply with all of the product quality standards for automotive products. |
| Test Case Derivation | We need to ensure that our product is up to par with the products on the market currently. |
| How will test be performed | Documentation will be reviewed on quality for automotive products and compare it to the CRA. Any potential signs of a breach of a standard will be discussed in a team meeting. |
| Requirements Referenced | CNFR49 |

# 6. Unit Tests

## 6.1. Unit Testing Scope

## 6.2. Test for Functional Requirements

## 6.3. Test for Nonfunctional Requirements

## 6.4. Traceability Between Test Cases and Modules

# 7. Appendix

## 7.1. Symbolic Parameters

## 7.2. Usability Survey Questions for Non Functional Tests

| **CNFRST Reference** | **Question** |
|---|---|
| CNFRST1 | Is the CRA designed in a way that is appealing? |
| CNFRST1 | Is the CRA designed in a way that is non-intrusive? |
| CNFRST1 | What would you add to the CRA if you were the designer? |
| CNFRST3 | Is the CRA software designed in a way that is appealing? |

| CNFRST Reference | Question |
|---|---|
| CNFRST3 | How would you rate the CRA's software GUI and application? |
| CNFRST3 | Was the Power ON system button easy to use? |
| CNFRST4 | Do you think you understand the CRA as a whole more now with the manual? |
| CNFRST5 | Is the document you were presented non-technical? |
| CNFRST4, CNFRST5 | How would you rate the understandability of the instruction manual provided? |
| CNFRST11 | How would you rate the overall feelability of the CRA to your bicycle? |
| CNFRST11 | Does the CRA hinder your ability to ride your bicycle? If so, in what way? |
| CNFRST21 | Is there any way that you would like to expand the CRA? |
| CNFRST21 | Do you have any concerns with the CRA on your own personal bicycle? |
| CNFRST29 | What is your brand and model of bicycle and if you know, what are the size and weight restrictions? |

# 8. Reflection

In preparation for the validation process of testing and system verification, a number of skills must be acquired. A greater depth of knowledge in these fields allows the CRA team to accurately substantiate the requirements set about in the SRS document via the testing methods identified in the VnV Plan.

As discussed in section 4.3. Design Verification Plan and section 4.6. Automated Testing and Verification Tools, Pytest will be used for automated unit testing of the software code developed for CRA. As such, it is expected that CRA team members will be responsible for educating themselves on how to efficiently and effectively use the tools provided by Pytest. There are a number of great resources available online that will aid our team by providing specific information regarding problems we might encounter. Some of these sites include stackoverflow.com and geeksforgeeks.org. However, as a general introduction it is advised that CRA team members review the pytest documentation and use the Linkedin learning course on pytest. As this project is a complete product, combining both physical and software systems, the verification and validation process will need to stretch beyond automated software test suites. Specifically, the team will need to develop methods for testing the mechanical and electrical domains of the project, which are mainly listed as Tests For Nonfunctional Requirements in Section 5.2. These tests will likely be manually carried out, as creating an automated mechanical/electrical testing structure will surely fall beyond the scope of this project. To begin, the team will collectively need the knowledge and skills to properly diagnose electrical circuits and components by hand. To name a few, team members will need to know how to follow a schematic/wiring diagram, solder, and use a multimeter. As per the scope of this project, only one prototype will be built and shared amongst the five team members. Thus, each member will need to be able to apply these skills should any electrical problems arise while the prototype is in their possession. In the same way, the team will collectively need the knowledge and skills to validate the mechanical aspects of the system. Validation will require the team to manually and properly set up, execute, and diagnose physical tests (for example, impact tests and repetitive strain tests). Validation will also involve testing of the mechanical-electrical interface (for example, testing of the mounting of the circuit board/peripherals). Another core component that CRA will be

highly dependent on to be successful is how well the team can validate components and various systems detailed in Section 5.1. The team will have to be able to pick up an understanding on how to test and validate modules that communicate with modern libraries and interdisciplinary scientific fields. Knowledge on how to validate these components are well beyond the scope of the course so it will be one of the key areas that CRA shall focus on. Not only must the CRA team members learn how to develop with these libraries, but they must come full circle and be able to understand how these tools are expected to perform and transition into a real world environment. Accurate and consistent validation will be a result of the team understanding how expected outcomes are formalized and why each validation process should be considered as industry standard. Our last component to address is seen in Section 5.2 and 7 of the Tests For Nonfunctional Requirements and Usability Surveys. Many different testing tools are required to ensure that the CRA will be fully operable and with as little error as possible. However, one major point of emphasis is our user feedback and Usability Surveys. Many of our tests will be using feedback from stakeholders so it will be necessary for the CRA team to document these well. In order to track these responses, we will use a Google Form for submissions and a Google Sheets. All members will be expected to use the responses given to further build upon the CRA system. We will gather the knowledge and skills required to use these applications.Skills that are required for this section but not explicitly stated is our need for soft skills. Being able to communicate our questions effectively in words, analyze and diagnose the responses of our users, and create thoughtful unit tests based on their feedback is of utmost importance. All of our team looks forward to continuing to work with our stakeholders and their critique to continue to develop our system and our tests. In summary, we can see that learning how to test effectively is very important to the CRA. Through continuously reading through and improving our VnV document, developing our software and hardware testing skills through the use of various resources, creating new system and unit tests, we will be able to effectively debug and rid our system of any potential errors.