**IEEE 829: Software & System**

**Test Documentation**

[1. Introduction 2](#_2zs24fxw5vuw)

[1.1. Purpose 2](#_psjdjs2h99x5)

[1.2. Scope 2](#_54391oeecktz)

[1.3. Definitions, Acronyms, & Abbreviations 2](#_dm134xwjf0ss)

[2. Test Items 3](#_ncfbadyrqxyo)

[3. Test Deliverables 3](#_b24ozhenr8j8)

[4. Responsibilities 4](#_yqr9b06oxjer)

[5. Features To Be Tested 4](#_gp7sqkkz9kaa)

[5.1. General Use-Cases 4](#_i7ly33i1f4cg)

[5.2. Features Per-Module 7](#_2m1nur78w6ew)

[5.2.1. CTC 7](#_5ddx16jt4o6d)

[5.2.2. Wayside (Software) 11](#_bu9653qc81c0)

[5.2.3. Wayside (Hardware) 14](#_w985z0la15fz)

[5.2.4. Track Model 18](#_2nduy8m67y7k)

[5.2.5. Train Model 22](#_boibky5q6mal)

[5.2.6. Train Controller (Software/Hardware) 25](#_q9ykhqyf0bv5)

# **Introduction**

## **Purpose**

This test plan for the Train Control System will specify the parts of the system to be tested, including responsibilities and deliverables. The objective is to ensure that every module of the system performs as specified, thereby affirming the system’s reliability, operational safety, and efficiency.

## **Scope**

The scope of this test document aims to clearly cover the testing of the Train Control System (TCS). Key areas to be tested include the Centralized Traffic Control (CTC), Wayside software and hardware components, the Track Model, Train Model, and both software and hardware aspects of the Train Controller. As such, this document outlines the parameters for each testing phase, including specific features, modules, and system integrations. The test environment will primarily be a Windows 10 platform to ensure compatibility and performance standards are met.

The main goal of this document is to affirm system reliability, safety, and efficiency through rigorous testing protocols, thereby certifying the system's readiness for live operational settings.

## **Definitions, Acronyms, & Abbreviations**

| **Term** | **Definition** |
| --- | --- |
| Authority | Distance in meters a train has the clearance to travel |
| CSV file | Comma Separated Value File |
| CTC | Centralized Traffic Control |
| Dispatcher | User who specifies when and where trains are supposed to travel in the system |
| GUI | Graphical User Interface |
| KI | Integral Gain |
| KP | Proportional Gain |
| Murphy | Entity/User of the system that causes failures by breaking the track and train |
| PLC | Programmable Logic Controllers |
| Programmer | User who creates and uploads the PLC files to the wayside controllers |
| PyQt5 | A python library for creating GUIs utilizing the Qt Company’s GUI toolkit |
| Route | The specific path a train takes as it travels along the track |
| SDD | Software/Hardware Design Description |
| SRS | Software Requirements Specifications |
| System Architecture | Encompassing term for how our TCS is divided into modules, and the communications between those modules and the users of the system. |
| TCS | Train Control System |
| Simulation Timing | How the system is synchronized in our simulation while simulating real world conditions of delay for certain events. |
| Train Controller | Vital component that is responsible for the behavior of each train |
| Train Engineer | User who determines and provides KI and KP for the Train Controller |
| UI | User Interface |
| User | A person who will interact with the system |
| Vital | Responsible for the safety of the system |

# **Test Items**

Each module of the system will be tested, including CTC, Wayside (Software), Wayside (Hardware), Track Model, Train Model, Train Controller (Software), and Train Controller (Hardware). These systems will be tested individually for functionality according to the Software/Hardware Design Description for the Autonomous Train Control System. The integrated system will then be tested. The system should be tested on a Windows 10 machine.

# **Test Deliverables**

Here, we will list the tools we used to perform our testing.

1. Test Plan: Here, different functions are tested and documented to ensure each component of each subsystem is working.
2. Test Cases: Various specific cases were tested to ensure typical user actions would be able to work without issue.
3. Unit Tests: Unit tests were created for each module to verify full functionality.

# **Responsibilities**

For testing, each module is in charge of thoroughly testing before integrating. Thus, testing assignments are as such…

CTC - Abby Magistro

Wayside Software - Alex Smith

Wayside Hardware - Lilly Jones

Track Model - Anh Nguyen

Train Model - Tanvi Verma

Train Controller Software - Lauren Gilfillan

Train Controller Hardware - Chad Collina

# **Features To Be Tested**

## **General Use-Cases**

Major features of the train system, as outlined in IEEE 1060: Software/Hardware Design Description, were tested to ensure system-wide functionality. Functions tested are included below.

| **Feature** | **Description** | **Risk** | **Testing Status** |
| --- | --- | --- | --- |
| Dispatch a Train (Manual) | Dispatcher enters the destination and arrival time for a single train. Train departs at the calculated time. | High | Tested |
| Upload a Schedule | Dispatcher uploads a train schedule in csv format, and the information should populate the schedule and run on the clock. | High | Tested |
| Display Block Occupancy Information | Block occupancy and train locations displayed on the CTC UI. | Low | Tested |
| Close a Block for Maintenance | The dispatcher can close a block in maintenance mode. | Medium | Tested |
| Set a Switch Position in Maintenance | The dispatcher can set switch positions in maintenance mode. | Medium | Tested |
| Display Throughput Information | Based on ticket sale data, throughput information from both tracks should be displayed. | Low | Not Tested |
| Serial Communication | Serial communication is enabled for hardware modules. | High | Tested |
| Uploading PLC File | Wayside programmer is able to create and upload a PLC file to enable automatic operation. | High | Tested |
| Wayside Logic (Automatic Mode) | Switches, lights, and crossing positions automatically change in accordance with PLC file contents and current track occupancies. | High | Tested |
| Wayside Logic (Manual Mode) | Switches, lights, and crossing positions change as indicated by user input. | Medium | Tested |
| Train Model Emergency Brake | Train should come at a fast but paced stop following Newtonian Laws when the emergency brake is enabled. | High | Tested |
| Train Accelerates and Decelerates | Train Speeding up and Slowing down should reflect across all UI’s and modules through Block Occupancies and Decrementing Authority | High | Tested |
| Train Stops Upon Request or Emergency | Train will toggle either its service or emergency brake at the appropriate request of other modules | High | Tested |
| Uploading a Track Layout File | The Track Builder is able to create and upload a Track Layout file (CSV or XLSL) for both red and green lines and the block data and the stations should reflect that on the UI. | Medium | Tested |
| Set Environmental Temperature | Track Model will toggle the environmental temperature and the track heaters should reflect that based on the threshold of 35.06 degrees Farenheits. | Low | Tested |
| Display Block Occupancy Information | Block occupancy and train locations displayed on the Track Model UI. | High | Tested |
| Track Model sending block occupancies based on Murphy’s failures. | Murphy can set block failures, and the UI should reflect this. Additionally, the block should be sent to Wayside Controllers as block occupancies. | Medium | Tested |
| Take in Commanded Speed, Authority, Track Lights and Switch and Crossing Status | The Track Model must be able to receive commanded speed and authority and send that information to the Train Model. Additionally, it must also be able to take in the status of switches, lights, and crossings and implement the track based on this information. | High | Tested |
| Sending Block Occupancy Information | The Track Model must be able to send block occupancy to Wayside Controllers when there is a train presence, as well as any block failures. | High | Tested |

## **Features Per-Module**

### CTC

| **Test: Dispatch a Train (Manual)** | |
| --- | --- |
| Description | For this test, we will ensure that a train can be manually dispatched by the user to any station in the system. Will be tested with multiple stations and dispatch times to ensure timing warnings are appearing when appropriate. |
| Features Tested | * Departure time calculation * Departure Station |
| Inputs | * Manual Mode selected * Line Color * Train ID * Destination * Arrival Time |
| Expected Outputs | * Schedule information for the train will populate the schedule accurately. * Train will dispatch at the listed departure time * Warning windows will pop up and allow the user to cancel their action if they choose to |
| **Test Results** | |
| Tester | Abby Magistro |
| Outcome | Trains successfully added to schedule. Warning pop up when timing conflicts. |

| **Test: Upload a Schedule** | |
| --- | --- |
| Description | The user will be able to upload a .csv file holding the schedule. Schedule should populate each entry with the correct information or warn the user if there are conflicts. |
| Features Tested | * Schedule Parsing * Departure time calculation * Departure Station |
| Inputs | * Schedule File: Formatted with Line, TrainID, Destination, and ArrivalTime columns. |
| Expected Outputs | * Schedule information for each entry will populate the schedule accurately. * All trains will dispatch at the listed departure time * Warning windows will pop up and allow the user to cancel their action if the file is incorrectly formatted |
| **Test Results** | |
| Tester | Abby Magistro |
| Outcome | Schedule was parsed and appeared on the schedule correctly. |

| **Test: Display Block Occupancy Information** | |
| --- | --- |
| Description | The CTC UI should display all current block occupancies, and indicate which occupancies correspond to each train. |
| Features Tested | * Occupied Blocks received from all waysides on both lines * Matching occupancies to the current trains on the track |
| Inputs | * Block occupancies from wayside * Current trains in the system (Stored internally) |
| Expected Outputs | * Correct block occupancies displayed with the correct Train ID * If no train should be in the area, an X will appear to indicate a likely failure |
| **Test Results** | |
| Tester | Abby Magistro |
| Outcome | Passes (on green line) |

| **Test: Close a Block for Maintenance** | |
| --- | --- |
| Description | In maintenance mode, the user should be able to select any block from both lines and indicate to the waysides that the block should be closed for maintenance. Trains should not be allowed to travel onto closed blocks. |
| Features Tested | * Closed blocks communicated to wayside * Trains do not travel onto closed blocks |
| Inputs | * Block selection to close from the user of the CTC UI |
| Expected Outputs | * Block closure appears on CTC UI * Block is shown as occupied on CTC UI, acknowledging that wayside is treating it as closed * All trains should stop before the closed block |
| **Test Results** | |
| Tester | Abby Magistro |
| Outcome | Passed |

| **Test: Set a Switch Position in Maintenance** | |
| --- | --- |
| Description | In maintenance mode, the user should be able to select any switch from both lines and indicate to the waysides that the block what position that switch should be in. Trains should follow the correct path if safe, or stop before the switch if unsafe. Switch’s position should not change until the switch is taken out of maintenance. |
| Features Tested | * Switch positions in maintenance communicated to wayside * Switch positions communicated to wayside and track model UIs * Trains exhibit the correct and safe behavior when switches are under maintenance. |
| Inputs | * Switch selection and position from the user of the CTC UI |
| Expected Outputs | * Switch positions appear on the CTC UI * Switch positions reflected correctly on wayside and track model UIs * Trains follow the switch position as set, or stop if unsafe. |
| **Test Results** | |
| Tester | Abby Magistro |
| Outcome | Passed |

| **Test: Display Throughput Information** | |
| --- | --- |
| Description | As tickets are sold and trains arrive at stations, ticket sales will be generated by track model and sent to the CTC. This information should be displayed on the CTC UI throughput graph. |
| Features Tested | * Communication between track model and CTC at the correct time * Ticket sales to throughput calculation is correct and displays correctly on the CTC UI |
| Inputs | * Ticket sales for track model, generated as trains stop at stations |
| Expected Outputs | * CTC graph reflects the correct information |
| **Test Results** | |
| Tester | Abby Magistro |
| Outcome | Not Tested, Track Model didn’t implement sending ticket sales. |

### Wayside (Software)

| **Test: Uploading PLC File** | |
| --- | --- |
| Description | For this test, we will make sure that a PLC file can be successfully uploaded and that the track changes (switches, lights, etc.) are reflected correctly within the system when the wayside module is in automatic mode. If lights are red, authority at those blocks are set to 0. |
| Features Tested | * Authority * Automatic mode logic * PLC attribute changes (switches, lights, crossings) * Communication between Wayside and Track Model modules |
| Inputs | * Upload PLC File * Dispatch a train (via CTC) to Whited station |
| Expected Outputs | * Trains will always go left on the track, meaning switches should follow suit. |
| **Test Results** | |
| Tester | Alex Smith |
| Outcome | In automatic mode, PLC-related track attributes (switches, lights, and crossing positions) were successfully calculated. |

| **Test: Crossing Positions** | |
| --- | --- |
| Description | This test focused on crossing positions being correctly calculated. This calculation occurs in automatic mode, in accordance with the PLC file contents. |
| Features Tested | * Crossing positions * Communication between Wayside and Track Model modules |
| Inputs | * Upload PLC File * Dispatch a train (via CTC) to Whited station |
| Expected Outputs | * When a train enters any block in a section with a crossing, it will go down. Otherwise, it will be up.. |
| **Test Results** | |
| Tester | Alex Smith |
| Outcome | Crossing position was correctly calculated according to current track occupancies. |

| **Test: Uploading Incorrect PLC File** | |
| --- | --- |
| Description | In this test, a PLC file is uploaded that will not calculate the correct switch/light positions, and may result in unsafe track behavior. |
| Features Tested | * Authority * Wayside vitality * Communication between Wayside and Track Model modules |
| Inputs | * Upload incorrect PLC File * Dispatch a train to any station on Green Line |
| Expected Outputs | * Anticipated switch positions are not able to automatically be changed. |
| **Test Results** | |
| Tester | Alex Smith |
| Outcome | The test was successful. |

| **Test: Manual Mode Logic** | |
| --- | --- |
| Description | In this test, the wayside module is set in manual operation. Switches, lights, and crossing positions are dependent solely on user input. |
| Features Tested | * Manual mode logic * Communication between Wayside and Track Model modules |
| Inputs | * Switch, light, and crossing states |
| Expected Outputs | * If the user of the wayside module inputs a specific switch position using the Wayside (SW) UI, this change should be reflected on the Track Model UI. |
| **Test Results** | |
| Tester | Alex Smith |
| Outcome | The test was successful, and the appropriate changes were reflected on the Track Model UI. |

### Wayside (Hardware)

| **Test: Serial Communication (Raspberry Pi 4)** | |
| --- | --- |
| Description | For this test, we will ensure that the Raspberry Pi 4 is able to receive data from the main computer in order to conduct vital calculations. Here, a train is dispatched to Whited station, and section occupancies are sent from the main device to RPi. |
| Features Tested | * Serial communication * Automatic mode logic * PLC attribute changes (switches, lights, crossings) * Communication between CTC, Wayside and Track Model modules |
| Inputs | * Upload PLC file * Dispatch a train (via CTC) to Whited station |
| Expected Outputs | * Raspberry Pi will receive a list of occupied block sections from the main device, and will send a list indicating switch, light, and crossing positions |
| **Test Results** | |
| Tester | Lillian Jones |
| Outcome | Serial communication was successfully completed. |

| **Test: Uploading PLC File** | |
| --- | --- |
| Description | For this test, we will make sure that a PLC file can be successfully uploaded to the Raspberry Pi using a simple UI, and that the track changes (switches, lights, etc.) are reflected correctly within the system when the wayside module is in automatic mode. If lights are red, authority at those blocks are set to 0. |
| Features Tested | * Authority * Automatic mode logic * PLC attribute changes (switches, lights, crossings) * Communication between Wayside and Track Model modules |
| Inputs | * Upload PLC File * Dispatch a train (via CTC) to Whited station |
| Expected Outputs | * Expected outputs will vary depending on track occupancies. For example, if only block A1 is occupied:   + Switch Positions:     - D13 = RIGHT (A1)     - F28 = LEFT (G29)   + Light Positions:     - A1 = GREEN     - C12 = GREEN     - G28 = GREEN     - Z150 = RED     - Crossing Positions: ALL UP |
| **Test Results** | |
| Tester | Lillian Jones |
| Outcome | In automatic mode, PLC-related track attributes (switches, lights, and crossing positions) were successfully calculated. |

| **Test: Crossing Positions** | |
| --- | --- |
| Description | This test focused on crossing positions being correctly calculated. This calculation occurs in automatic mode, in accordance with the PLC file contents. Here, the crossing at block E19 was examined. |
| Features Tested | * Crossing positions * Communication between Wayside and Track Model modules |
| Inputs | * Upload PLC File * Dispatch a train (via CTC) to Whited station |
| Expected Outputs | * When a train enters any block in section E, the crossing at E19 should be DOWN in order to prevent vehicles from crossing the track. After leaving section E, the crossing should return to the UP position. |
| **Test Results** | |
| Tester | Lillian Jones |
| Outcome | Crossing position was correctly calculated according to current track occupancies. |

| **Test: Uploading Incorrect PLC File** | |
| --- | --- |
| Description | In this test, a PLC file is uploaded that will not calculate the correct switch/light positions, and may result in unsafe track behavior. |
| Features Tested | * Authority * Wayside vitality * Communication between Wayside and Track Model modules |
| Inputs | * Upload incorrect PLC File * Dispatch a train to any station on Green Line |
| Expected Outputs | * Anticipated switch positions are calculated on both software and hardware. The hardware calculations will not be consistent with those on software. An error is detected, and the authority of all trains currently on the track is set to 0. |
| **Test Results** | |
| Tester | Lillian Jones |
| Outcome | The test was successful, and 0 authority was sent to all trains. |

| **Test: Manual Mode Logic** | |
| --- | --- |
| Description | In this test, the wayside module is set in manual operation. Switches, lights, and crossing positions are dependent solely on user input. |
| Features Tested | * Manual mode logic * Communication between Wayside and Track Model modules |
| Inputs | * Switch, light, and crossing states |
| Expected Outputs | * If the user of the wayside module inputs a specific switch position using the Wayside (HW) UI, this change should be reflected on the Track Model UI. |
| **Test Results** | |
| Tester | Lillian Jones |
| Outcome | The test was successful, and the appropriate changes were reflected on the Track Model UI. |

### Track Model

| **Test: Toggle Failures** | |
| --- | --- |
| Description | Murphy will initiate a failure whenever they need to. This initiating scenario includes broken rails, track circuit failures, and power failures. |
| Features Tested | * Toggle broken rail button and see resulting block(s) on UI * Toggle track circuit failure and see resulting block(s) on UI * Toggle power failure button and see resulting block(s) on UI |
| Inputs | * Button clicked signal |
| Expected Outputs | * Resulting block(s) changed color on UI * Resulting block(s) sent to Wayside Controller as a block occupancy. (Shown on their UI) * Should disappear when the buttons toggle again |
| **Test Results** | |
| Tester | Anh Nguyen |
| Outcome | The changes were successful and the appropriate changes were reflected on the track model, and Wayside Controller UI. |

| **Test: Uploading Track Layout** | |
| --- | --- |
| Description | The Track Builder is responsible for updating track layout by uploading the track layout data stored in CSV or Excel files. When they do so track layout data will update according to the file data. |
| Features Tested | * Upload track layout and UI reflecting changes based on the uploaded file. |
| Inputs | * Button clicked signal and uploading a file (CSV or XLSL) |
| Expected Outputs | * After uploading the file, by clicking on the blocks, the changes will be reflected on the UI in the block and station data. |
| **Test Results** | |
| Tester | Anh Nguyen |
| Outcome | The changes were successful and the appropriate changes were reflected on the track model UI. |

| **Test: Take in Commanded Speed, Authority, Track Lights and Switch and Crossing Status** | |
| --- | --- |
| Description | The Wayside Controller must send commanded speed, authority, and lights and switch status so that the information can be passed onto the train model and eventually be applied to the physical train. |
| Features Tested | * Successfully send commanded speed and authority information to the train model. * Display track lights, switches, and crossing status on the UI for each block. |
| Inputs | * PyQt signals from Wayside Controller software and hardware. |
| Expected Outputs | * The commanded speed and authority are successfully sent to the train model so that it can be relayed to the train controller. * Lights, switches, and crossing status are displayed correctly on the UI. |
| **Test Results** | |
| Tester | Anh Nguyen |
| Outcome | The changes were successful, and the appropriate changes were reflected in the track model UI. The commanded speed and authority were successfully sent to the train model. |

| **Test: Sending Ticket Sales** | |
| --- | --- |
| Description | Everytime the train stops at the station, the track model sends the CTC and Train Model the ticket sales. |
| Features Tested | * Successfully generate and send ticket sales information to the Train Model and CTC |
| Inputs | * PyQt signals to CTC and Train Model |
| Expected Outputs | * Upon receiving the ticket sales data, the CTC Operator reviews the information and makes strategic decisions to adjust train schedules. correctly. * The Train Model will use that information to calculate people disembarking and send that information back to the track model. |
| **Test Results** | |
| Tester | Anh Nguyen |
| Outcome | The changes were not able to be resolved. |

| **Test: Sending Presence on Block to Wayside Controller** | |
| --- | --- |
| Description | Track Model will send block occupancy information to Wayside Controller. |
| Features Tested | * Track model will use actual velocity from train model and authority along with block length from Excel parser to calculate block occupancy to be sent to Wayside Controller. |
| Inputs | * PyQt signals to Wayside Controllers |
| Expected Outputs | * Upon receiving the occupancy information, the Wayside controller will send information to CTC and update its system of the status of the train’s location. |
| **Test Results** | |
| Tester | Anh Nguyen |
| Outcome | The changes were successful, and the appropriate changes were reflected in the track model and Wayside Controller UI. |

| **Test: Setting Environmental Temperature** | |
| --- | --- |
| Description | Track Model can set environmental temperature which toggle track heaters on if the temperature is below 35.06 degrees Fahrenheit. If the environmental temperature is above 35.06 degrees Fahrenheit, then track heaters toggles off. |
| Features Tested | * Track model will set environmental temperature. |
| Inputs | * Numbers larger than 35.06. * Numbers less than 35.06. |
| Expected Outputs | * Upon receiving environmental temperature, track model UI will set track heaters on or off based on the threshold of 35.06 degrees Fahrenheit. |
| **Test Results** | |
| Tester | Anh Nguyen |
| Outcome | The changes were successful, and the appropriate changes were reflected in the track mode UI. |

### Train Model

| **Test: Emergency Brake Stop** | |
| --- | --- |
| Description | When a user clicks the Emergency brake button, the train should gradually come to a halt while power drops to 0. |
| Features Tested | * Emergency brake UI button * Emergency brake PyQt signal between Train Model and Train Controller |
| Inputs | * Button clicked signal |
| Expected Outputs | * Emergency Brake button grayed out and cannot be toggled back. * Power input displays 0. * Acceleration and Current velocity gradually drop until the values reach 0. |
| **Test Results** | |
| Tester | Tanvi Verma |
| Outcome | The changes were successful and the appropriate changes were reflected on the train model and train controller UI. |

| **Test: Murphy Failure Signals** | |
| --- | --- |
| Description | When a user enables any of the 3 Murphy failure signals, the changes should reflect on the Train Model UI and should be sent to the Train Controller. |
| Features Tested | * Failure signal enable and disable button * PyQt signal for each signal. |
| Inputs | * Enable/Disable UI button. |
| Expected Outputs | * Depending on the button clicked, the enable or disable button should change color and reflect the selection state. * If the enable button is clicked, the disable button should get unselected for the particular failure. * No constraints should exist while selecting different enable/disable buttons for different failures. |
| **Test Results** | |
| Tester | Tanvi Verma |
| Outcome | The changes were successful and the appropriate changes were reflected on the train model and train controller UI. |

| **Test: Grade Functionality** | |
| --- | --- |
| Description | When a train enters a block with grade, the acceleration value should reflect the change in grade and the current velocity should change accordingly. |
| Features Tested | * Grade PyQt signal from track model. * Force, acceleration and current speed calculations. |
| Inputs | * PyQt signal from Track Model as blocks change. |
| Expected Outputs | * Acceleration starts to drop slightly as grade increases and vice versa as grade decreases. * Current velocity drops or increases depending on the current acceleration. |
| **Test Results** | |
| Tester | Tanvi Verma |
| Outcome | The changes were successful and the appropriate changes were reflected on the train model UI. |

| **Test: Cabin Temperature changing as live data** | |
| --- | --- |
| Description | A running train should display a reasonable temperature in Fahrenheit and display slight changes throughout the journey to indicate live data. |
| Features Tested | * Global timer. |
| Inputs | * PyQt signal from the main simulation as an instance of the train model is created. |
| Expected Outputs | * Cabin temperature displaying temperatures in the range of 65-75 F and changing every minute on the train model UI. |
| **Test Results** | |
| Tester | Tanvi Verma |
| Outcome | The changes were successful and the appropriate changes were reflected on the train model UI. |

| **Test: Passenger Count and mass changing at every station.** | |
| --- | --- |
| Description | A running train should display passengers disembarking and boarding at all stations throughout the journey except the yard. |
| Features Tested | * Passengers onboarding and offboarding signal between the track and train model. * Mass calculations for train model. |
| Inputs | * PyQt signal between track model and train model. |
| Expected Outputs | * At every station, the mass value under live statistics of the train should update and the passenger count should change. |
| **Test Results** | |
| Tester | Tanvi Verma |
| Outcome | The changes were successful and the appropriate changes were reflected on the train model UI. |

### Train Controller (Software/Hardware)

| **Test: Emergency Brake** | |
| --- | --- |
| Description | When the emergency brake is pressed, either by a passenger or the driver, the Train will slow down and come to a complete stop |
| Features Tested | * E Brake Button on UI * Vital Safety Logic |
| Inputs | * E Brake Signal |
| Expected Outputs | * Train slows down when EBrake is applied * Train cannot speed up when the Ebrake is applied * All other inputs are disabled until Ebrake is disabled |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Brake Failure** | |
| --- | --- |
| Description | When a Brake Failure Signal is passed to the Train, the train should take appropriate safe actions and apply the emergency brake |
| Features Tested | * Vital Safety Logic |
| Inputs | * Brake Failure Signal |
| Expected Outputs | * Emergency Brake is toggled * Accelerator and Service Brake are disabled * Brake Failure Indicator Lights Up |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Signal Pickup Failure** | |
| --- | --- |
| Description | When a Brake Failure Signal is passed to the Train, the train should take appropriate safe actions and apply the emergency brake |
| Features Tested | * Vital Safety Logic |
| Inputs | * Signal Failure Signal |
| Expected Outputs | * Emergency Brake is toggled * Accelerator and Service Brake are disabled * Signal Failure Indicator Lights Up |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Power Failure** | |
| --- | --- |
| Description | When a Power Failure Signal is passed to the Train, the train should take appropriate safe actions and apply the emergency brake |
| Features Tested | * Vital Safety Logic |
| Inputs | * Power Failure Signal |
| Expected Outputs | * Emergency Brake is toggled * Accelerator and Service Brake are disabled * Power Failure Indicator Lights Up |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Manual Change KI and KP** | |
| --- | --- |
| Description | A Train Engineer Should be able to modify KI and KP while the Train is moving and observe how it affect the power |
| Features Tested | * Power Output * KI Input Scroll Box * Kp Input Scroll Box |
| Inputs | * KI * KP |
| Expected Outputs | * As Kp rises, the Power should proportionally become larger * As Ki rises, the Power should increase rate at which it rises |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Power (SW)** | |
| --- | --- |
| Description | Given an Acceleration Slider Input and Commanded Speed, Power should continue to rise as the Train Speeds up, it should be zero when the Train is deccelerating |
| Features Tested | * Power Equation * Acceleration Slider |
| Inputs | * Acceleration Slider (0 -100) |
| Expected Outputs | * As Acceleration Slider becomes larger, power should become larger * Over time, before reaching peak speed, Power should be growing |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Power (SW)** | |
| --- | --- |
| Description | Given an Acceleration Slider Input and Commanded Speed, Power should continue to rise as the Train Speeds up, it should be zero when the Train is deccelerating |
| Features Tested | * Power Equation * Acceleration Slider |
| Inputs | * Acceleration Slider (0 -100) |
| Expected Outputs | * As Acceleration Slider becomes larger, power should become larger * Over time, before reaching peak speed, Power should be growing |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Doors (Manual Mode)** | |
| --- | --- |
| Description | When in manual mode, the driver should be able to toggle which doors are open only when the train is at a complete stop |
| Features Tested | * Open / Close Buttons * Vital Safety Logic |
| Inputs | * Toggle Open / Close Button |
| Expected Outputs | * Only if Current Speed is 0 , should the doors button be allowed to toggle |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Interior Lights (Manual Mode)** | |
| --- | --- |
| Description | When in manual mode, the driver should be able to toggle which mode the interior lights are set at |
| Features Tested | * Interior Light Slide Bar |
| Inputs | * Click on Slide Bar |
| Expected Outputs | * As Interior Light Bar is changed, the appropriate value should be outputted to Train Model reflecting the state of lights |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Exterior Lights Toggled Underground (Manual Mode)** | |
| --- | --- |
| Description | When the Train enters underground part of the track, the exterior lights will automatically toggle on |
| Features Tested | * Non Vital Logic |
| Inputs | * Block Changed * Beacon Information |
| Expected Outputs | * When the train enters an underground portion of the track, the Non Vital logic will emit a 1 to Train model via the Exterior Light S |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Doors (Automatic Mode)** | |
| --- | --- |
| Description | When the Train arrives at the station, the doors will automatically open based on the side of the track the station is |
| Features Tested | * Non Vital Logic * Beacon Information |
| Inputs | * Authority * Beacon Information |
| Expected Outputs | * Train Controller will emit proper values based on door side to the Train Model |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Interior Lights (Automatic Mode)** | |
| --- | --- |
| Description | As a Train arrives at a station, it will turn interior lights on |
| Features Tested | * Non Vital Control Logic * Beacon Information |
| Inputs | * Block Changed |
| Expected Outputs | * At arrival at station, Train controller will output 1 via the interior light signal, and while leaving the station will output 0 via |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Temperature Control (Automatic Mode)** | |
| --- | --- |
| Description | Automatic Controller will monitor temperature and set it to 69 degrees at all time |
| Features Tested | * Cabin Temperature Set * Non Vital Logic |
| Inputs | * Cabin Temperature |
| Expected Outputs | * As Train moves, the controller will increase and decrease the temperature based on the current cabin temperature |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | Failed |

| **Test: Announcements (Automatic Mode)** | |
| --- | --- |
| Description | As the Train moves along the track, it will announce the next station 1 block before it arrives there |
| Features Tested | * Announcements * Beacon Information * Non Vital Logic |
| Inputs | * Block Polarity |
| Expected Outputs | * As a train approaches a station, it will output announcement 1 block before and at the station |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Beacon Information Display** | |
| --- | --- |
| Description | As the Train Moves along beacons on the Track, the Next Station, Speed Limit and Annoucments fields will propagate with information |
| Features Tested | * Line\_Dictionary * Beach Pickup Signal * Announcements * Speed Limit Display |
| Inputs | * Beacon Information (128 bit value) |
| Expected Outputs | * Values will update accordingly based on location on Track |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Boolean Authority** | |
| --- | --- |
| Description | When receiving a boolean authority, the trains will break until boolean authority is removed |
| Features Tested | * Vital Safety Logic |
| Inputs | * Boolean Authority * Current Speed |
| Expected Outputs | * When the boolean authority signal is pulled high, the train will apply the brakes until the signal is pulled low |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Speed Control - Driver attempts to go over speed limit (Manual Mode)** | |
| --- | --- |
| Description | A driver may try to increase the speed of the train in manual mode that may lead to the train exceeding the speed limit, creating a dangerous situation |
| Features Tested | * Vital Safety Logic |
| Inputs | * Acceleration * Current Speed * Speed Limit * Commanded Speed |
| Expected Outputs | * If the driver attempts to go over the speed limit, the brake will automatically be toggled until the Train is back under it speed limit |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Speed Control - Automatic Mode Drivers at Commanded Speed** | |
| --- | --- |
| Description | Automatic Control Logic will not allow the system to drive above or below the commanded speed and the speed limit |
| Features Tested | * Vital Safety Logic |
| Inputs | * Speed Limit, Current Speed, Commanded Speed |
| Expected Outputs | * Provided both a commanded speed and speed limit, the Train will not exceed or remain significantly below either value |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Decrementing Authority** | |
| --- | --- |
| Description | Train will properly calculate how to stop so that it is at a full stop before it reaches zero authority |
| Features Tested | * Authority Monitor |
| Inputs | * Authority * Current Speed |
| Expected Outputs | * As the Train moves, the Authority will properly decrement based on current speed |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

| **Test: Stopping at Authority (Automatic Mode)** | |
| --- | --- |
| Description | Train will properly calculate how to stop so that it is at a full stop before it reaches zero authority |
| Features Tested | * Vital Safety Logic |
| Inputs | * Authority * Current Speed |
| Expected Outputs | * As Train Approached the end of its Authority, the logic will appropriately toggles the service brake in order to stop at Authority is less than zero |
| **Test Results** | |
| Tester | Lauren Gilfillan |
| Outcome | No Bugs Found |

### 