Requirements Document

for

Hug the Rail IoT

Version 5.0

Prepared by Tristan Kensigner, Grace Mattern, Zachary Schuh, &
Christian Szablewski-Paz

Stevens Institute of Technology

April 6, 2021

Table of Contents

Revision History

Name(s)	Date	Reason For Changes	Version
Christian	2/15	Create document/begin section 1: Intro	1.0
Grace	2/16	Revise section 1: Intro	1.0
Zach, Christian, Tristan	2/16	Organize section 1: Into paragraphs, add notes	1.0
Grace, Christian, Zach, Tristan	2/22	Add section 2: Overview	1.0
Christian	2/23	Rewrite bullet points into paragraphs	1.0
Tristan	2/23	Add section 9: Cost	1.0
Grace	3/1	Revise/organize entire document, add graphics	1.0
Tristan	3/1	Elaborate section 2.4, add graphics	1.0
Grace and Christian	3/2	Add section 3: Requirements	1.1
Grace	3/4	Revise section 3: Requirements	1.1
Grace, Tristan, Christian	3/9	Update section 3: Requirements with CTO of HTR recommended features	1.1
Grace	3/11	Update timeline, slippage system diagram, section 3: Requirements	2.0
Grace	3/16	Incorporate TA 1.0 notes, add section 4: Requirements Modeling, fix formatting	2.1
Christian, Grace, Tristan	3/17	Add use cases in section 4: Requirements Modeling	2.1
Christian	3/20	Work on model index cards	2.2
Christian, Tristan, Grace, Zach	3/21	Finish section 4: Requirements Modeling	3.0

Grace, Christian, Tristan, Zach	3/28	Make changes as directed to section 4: Requirements Modeling	4.0
Tristan, Christian, Grace	4/5	Add section 5: Software Architecture	5.0

1. Introduction

1.1 Problem

Trains are a very common and popular form of transportation that require analysis of data derived from the weather conditions, fuel consumption, and time prediction, among others, in order to ensure safe and successful travel. Current trains retrieve data from servers to receive updates that are needed to make calculations for the Locomotive Control System (LCS). One major setback to this system is that trains need to be connected to servers to keep receiving updates and consequently provide safe travel. This leaves the trains vulnerable when a cellular wifi or internet connection is lost. No new data can be retrieved without this connection and thus the safety of the passengers and operator are jeopardized.

1.2 Purpose

The Internet of things (IoT) Hug the Rail (HTR) project is a system that will allow the LCS to make decisions locally in absence of cellular and wifi connectivity to back offices. For example, it will provide the operator with information regarding if the train is slipping or if there is an obstruction. Furthermore, the operator will be able to make informed decisions regarding the problems and cautions specified by the IoT HTR thereby maintaining safe operation and avoiding potentially dangerous situations. The overall purpose of this product would be to make HTR safer, less costly, and more efficient.

1.3 Team

Our dedicated team of developers are skilled in areas such as problem-solving, programming, and project management and are excited to participate in the project. Despite having little experience with embedded systems, GitLab, and IoT systems, we are driven to design a system that allows trains to efficiently operate without a dependence on a connection. Christian, Zach, and Tristan's main roles will be coders and problem solvers. Coders' responsibilities include coding, debugging and testing. Grace's role will be as a project manager due to her being highly organized and detail-oriented. The project manager's responsibilities include creating a schedule and timeline to execute each phase of the project. We hope to innovate and reimage the future of the locomotive industry. Our team will develop and implement IoT Hug the Rail from February 9th to April 20th of 2021.

1.4 Evolving Current Operations

Current operations on trains receive data from the departure station and cloud. During the travel, updated data is sent from servers in range and the operator utilizes the information to control the LCS. Once at the destination, the train, station, and cloud exchange data. The cloud communicates this information over the larger network. While the LCS has evolved over the years as it now can supply lots of information such as heat of the engine, fuel consumption, and generally offers a failsafe-like "software" to prevent the engine from operating outside of predefined safe limits.

1.5 Approach

The edge devices will capture data from other HTR locomotives and the environment instead of relying on data from servers. Many sensors will be implemented to provide crucial information that has now become unavailable (due to the lack of connection). Data from sensors will be sent to a time sensitive networking router (TSNR) which will act as a bridge between the raw sensor data and IoT HTR. IoT HTR will make any necessary calculations and display data from the sensors. IoT HTR's main goal is to make suggestions via the Display to the operator. The product will provide the capability for the operators to receive statuses and download the latest rules for operation from the Fog/Cloud onto the IoT software. Consequently, a finite state system will be created, and such features will help make the appropriate suggestions to the operator.

1.6 Timeline*

The timeline of this project will closely follow an agile methodology; throughout the development process changes will likely be made to both requirements and capabilities. Within each iteration of development, the team will work to recognize problems early and adapt to any changes while continuing to work efficiently. These include but are not limited to software design, technical requirements, deliverables, and the timeline itself. We find that this model helps provide a high-quality project that is attuned to the needs and desires of the consumers and stakeholders while also reducing the failure to analyze risks.

Week 1: 2/2 -	**Communication phase** (constantly revisited, understand new desires of stakeholders)
	Form teams
	Decide roles and responsibilities
Week 2: 2/9 -	**Planning phase** (revisited to address issues)
	Understand the problem
	Create documentation template
	Start Section 1: Intro
	Research existing systems
Week 3: 2/16 -	Review/revise Section 1: Intro
	Start Section 2: Overview
	Research sensors that can be used to provide information to IoT HTR
	Begin Section 10: Cost
Week 4: 2/23 -	Familiarize team with GitLab/Git
	Upload version 1.0 on GitLab
	Research wheel sensors
	Research front sensors
	Research GPS interaction with wheel sensor
	Research thermometer interaction with wheel sensor
	Understand how sensors capture data from other HTR trains and surrounding environment
Week 5: 3/2 -	**Requirement Analysis** (revisited if old model fails to fit new requirements)
	Start Section 3: Requirements
	Implement sensors that enable the IoT HTR to make suggestions
	Add CTO of HTR features to Section 3
	Research gate status sensors
	Research hardware and OS for IoT HTR
Week 6: 3/9 -	Continue Section 3
	Use analytics engine to process info from the sensors to the LCS
Week 7: 3/16 -	**Requirements Modeling**
	Start Section 4: Use Cases
	Enable operator to download the latest data from Fog/Cloud onto the IoT HTR when connected
Week 8: 3/23 -	Continue Section 4
Week 9: 3/30 -	Continue Section 4
Week 10: 4/1 -	**Software Architecture**

Start Section 5
Week 11: 4/6 - **Coding**
Week 11: 4/13 - **Testing**
Continue coding
Begin testing
Week 11: 4/20 - **Release first iteration**
Copy code to Documentation (Section 6: Code)
Start Section 7: Testing

*Subject to change

2. Overview

2.1 Problem Statement

Current IoT technologies on trains are focused on live data received from the Central Operation Center Servers via WiFi/Cellular networks. This heavy dependence on WiFi/Cellular connectivity introduces a massive weakness regarding safety when a stable connection is lost. Therefore it is imperative that a reliable IoT system based on local data be developed to allow for the continuation of a safe trip. For a train to operate properly external data must be provided. In the case that wifi is unavailable to provide the necessary data, a backup solution must be put in place.

2.2 Data Required to Operate

IoT HTR will need to collect data about the weather conditions (snow, ice, rain, wind speed) as it is essential in recognizing hazardous weather conditions. Additionally, this data provides information regarding safe travel speeds and whether to put down sand or halt. Data regarding moving objects (front or back) is especially important if the obstruction is another train or moving object(s). IoT HTR will alert the operator to slow down, stop, or take another route.The train will also need information about the crossing gate's status (open or closed). If the gates are open, the train will need to slow down or halt because vehicles may be crossing the train crossing. We assume that the train will still have access to the GPS/satellite. The GPS data can coordinate with the wheel sensor data which will provide data on the wheels' spin rate (rpm). In a sense, these two data will act to verify each other since when they contradict each other it should alert the operator that the train is probably slipping.

2.3 Expected Output

The data will be accessible to IoT HTR via the TSNR, from which the operator can read the suggestions from IoT HTR Display and make the appropriate decisions. Specifically, IoT HTR will interface with sensors to collect data, analyze it, and use rule-based logic to issue recommendations or actions to the operator via the IoT HTR Display by using TSNR as a proxy. For example, if the front sensor detects an object is too close, it then will detect if it is moving. This information will be passed to TSNR which will then pass it to IoT HTR. IoT HTR will provide a hazard warning to the operator, who can then slow down or stop the train. Another example is the wheel sensor which will verify its data with the still working GPS. The GPS provides the speed the train is going while the wheel sensor will provide how fast the wheels are rotating (rpm) and thus if one of the data fails to corroborate the other, then a warning will appear on the IoT HTR Display. The thermometer will provide additional information concerning what the likely cause is (i.e. ice, snow, rain, etc.). With both the knowledge that the train is slipping and the likely cause, the operator can slow down or pour sand on the railroad tracks. This same system of sensors can also notify the operator when slipping has stopped.

2.4 Available Technologies

There are a multitude of different sensors available. Thermometers, hygrometers, friction/shear force sensors, anemometers, and tachometers are the main sensors we intend to use to pass weather-related information to the LCS and consequently to the operator. A hygrometer measures the humidity, an anemometer measures wind speeds, and a tachometer measures the rpm of a wheel. Another important sensor is a radar/sonar/ultrasonic-like device that can map out the environment surrounding the train for the purpose of warning/alerting the operator of obstructions. Popular technologies include LED time of flight sensors and

LiDAR sensors, the latter of which are not uncommon on cars and airplanes. A LiDAR sensor continually fires off beams of laser light and then measures how long it takes for the light to return to the sensor. In effect, a LiDAR sensor returns the distance of objects hit by the laser beam and thereby creating a 3D visualization of the environment. With continuous firing, it can also present the movement of objects surrounding the train just as it would notify a driver that a pedestrian, curb, or other vehicle is near. The GPS is a versatile technology that provides information about the position of the train. Not only is it highly reliable, but it also integrates well with the previous technologies. When used in conjunction with weather-based sensors, it can help authenticate that a train is slipping. Additionally, when used in conjunction with obstruction detection sensors, it can provide the appropriate speed to slow down the train.

The sensors for the IoT Hug the Rail Project would be mainly located in three spots. The top of the train will house most of the weather related sensors such as the anemometer, thermometer, and hygrometer. The tachometer and friction sensors will be located near the wheels of the train as these sensors require information from the wheel's rotation and friction. This data will provide the necessary information to IoT HTR about the train's speed, change of speed, and friction force. The front and back of the train will house sensors concerning 3D mapping for obstacle detection. Our IoT system will still have access to the GPS as it does not require a WiFi/cellular connection. The status (open/closed) of cross gates will be handled by object recognition cameras that continuously compare photos taken with a library of reference photos. **Figure 1** provides a general layout of the sensors on HTR trains.

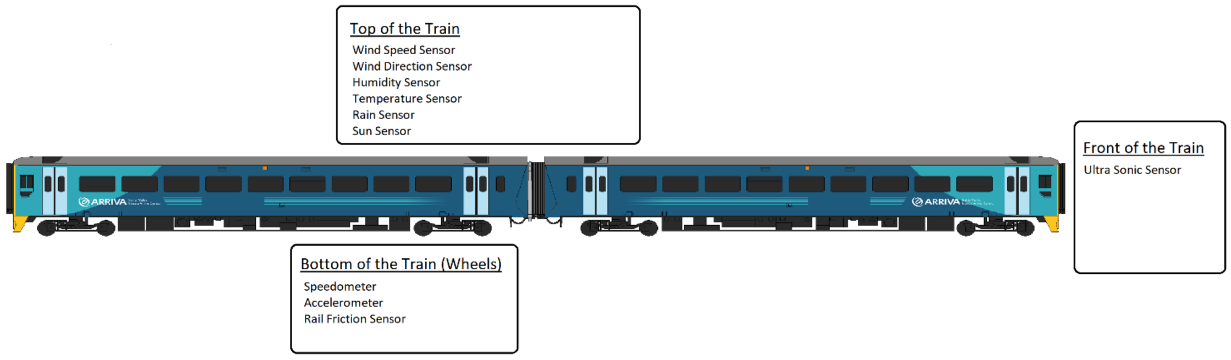


Figure 1

2.5 Systems

The IoT HTR system will collect data retrieved from the various sensors and interpret them accordingly. Results (i.e. weather, speed, obstructions, etc.) will be available via the IoT HTR Display and suggest operational changes to the operator based on its detections. Below are some conceptual architectures for our IoT. **Figure 2** and **Figure 3** communicate the general problem and solutions. **Figure 4** shows how IoT HTR would be used specifically regarding weather based slippage.

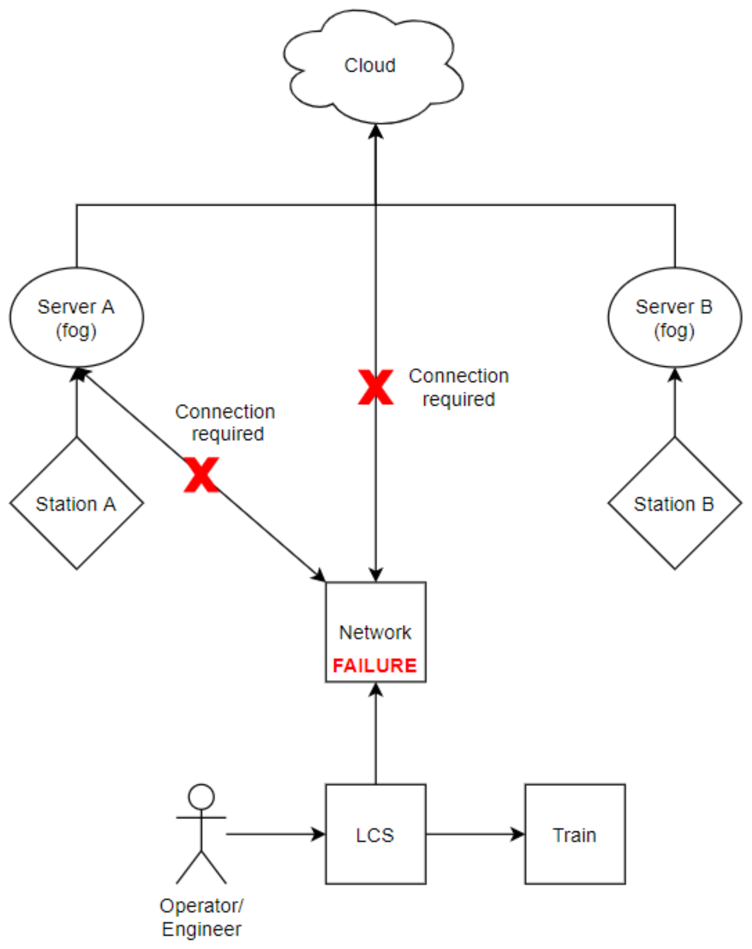


Figure 2: *shows the problems that occur if a connection is lost (represented by the red x's). Supposing the train starts in the range of station A, the LCS is connected to the network which receives data from server A and also the cloud. However when the connection is lost no new data is retrieved from server A or the cloud and the network fails.*

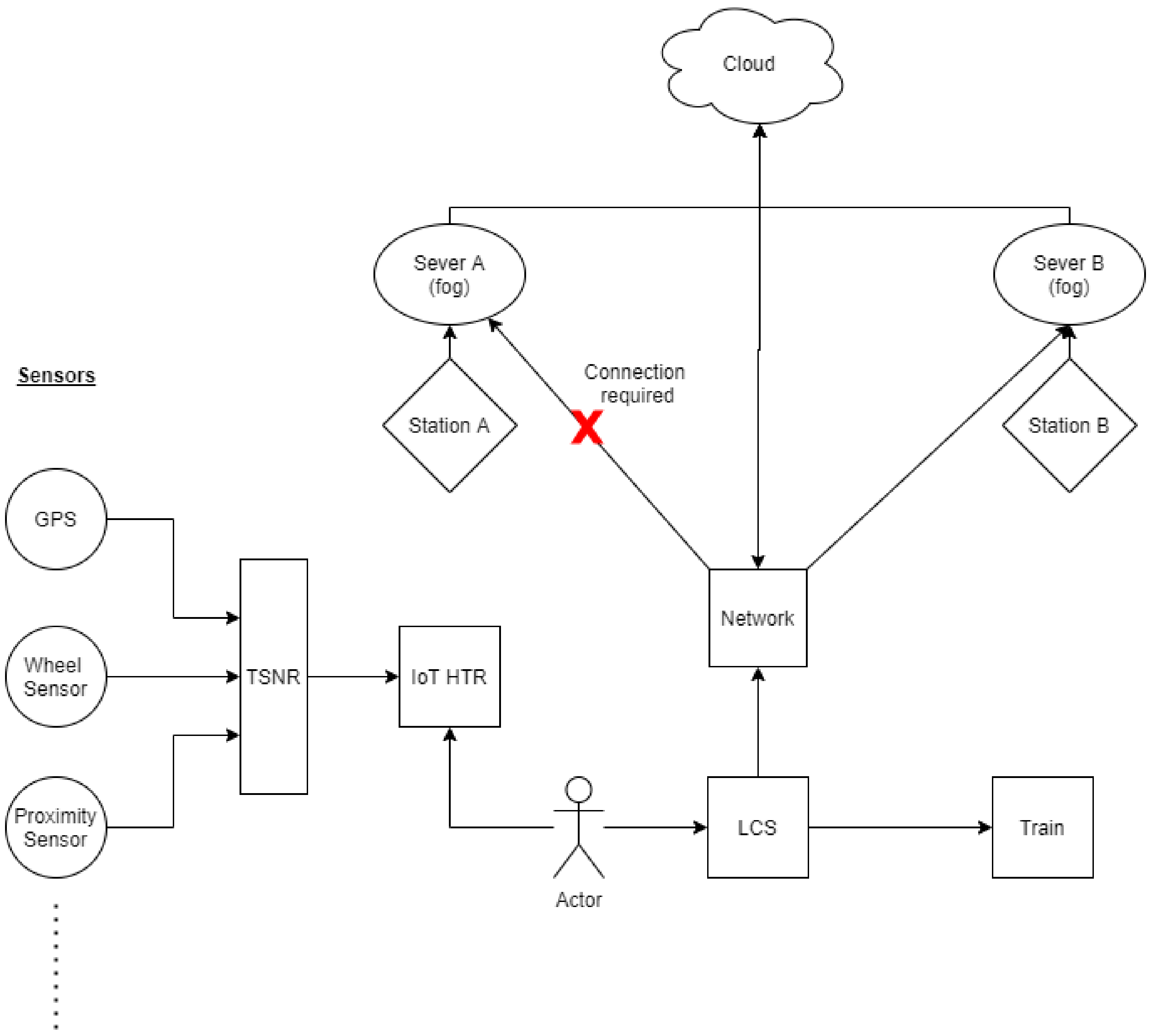


Figure 3: *shows the general solution IoT Hug the Rail Project will provide. Supposing the train is headed to station B without a connection, information from the GPS, proximity sensor, wheel sensor, and other sensors are passed through the TSNR and processed by the IoT HTR. IoT HTR displays the data and any warnings which are read by the operator who can then make the appropriate decisions. Once at station B, a connection is reestablished. The train, server B, and the cloud share data.*

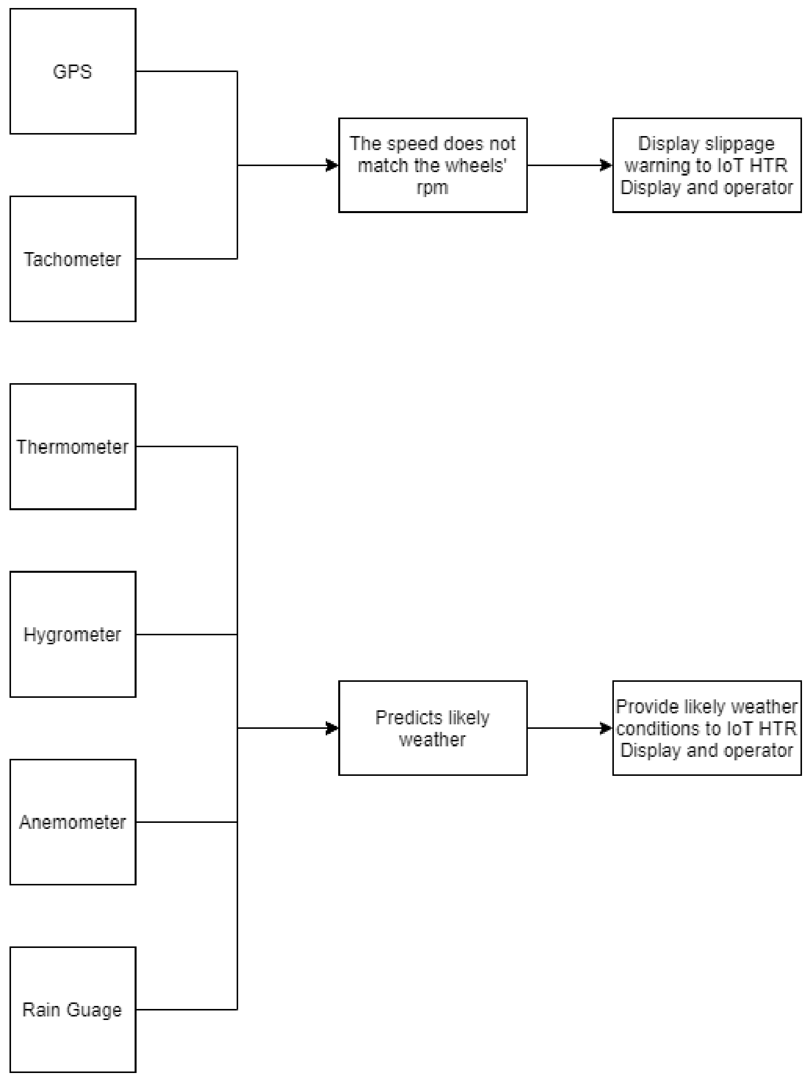

GPS
Tachometer
The speed does not match the wheels' rpm
Display slippage warning to IoT HTR Display and operator
Thermometer
Hygrometer
Anemometer
Rain Guage
Predicts likely weather
Provide likely weather conditions to IoT HTR Display and operator

Figure 4: *Weather sensors can provide information about ice, snow, rain, slipping, storms, winds, etc. For example, the GPS could report the speed of the train but the tachometer would contradict the data by showing that the wheels are rotating faster than the given speed. The thermometer, hygrometer, and anemometer could predict the weather conditions (i.e. snowing, raining, storming, etc.). The following data will be available to the operator on the IoT HTR Display.*

3. Requirements

3.1 Non-Functional Requirements

3.1.1 Security

R-1: IoT HTR shall only be accessed via user ID and password.
R-2: IoT HTR Admin shall have secured (admin ID/Password) access to all sensors and equipment.
R-3: Operators shall only see the operations of the train on the IoT HTR Display .
R-4: Only Admins shall have access to raw data and logs.
R-5: Only Admins shall have access to system configurations.
R-6: Only Admins shall be able to download/update the software.
R-7: Only Admins shall be able to resolve the IoT HTR if it fails.
R-8: IoT HTR network shall be secured by a LoRaWan Protocol.
R-9: IoT HTR shall endode all user-supplied output.
R-10: IoT HTR shall lock the system after 5 failed attempts to login.
R-11: IoT HTR shall unlock after 5 minutes.
R-12: Only Operators and Admins shall have access to capabilities listed in 3.2.

3.1.2 Performance

R-13: IoT HTR sensors shall process an event within 0.5 seconds.
R-14: TSNR shall send sensor data to IoT HTR within 0.5 seconds.
R-15: IoT HTR shall process an event within 0.5 seconds.
R-16: IoT HTR shall process 1,000 events per second without delay.
R-17: IoT HTR shall display processed events within 0.1 seconds.
R-18. The Log will log events every 0.01 seconds.

3.1.3 Reliability

R-19: IoT HTR shall operate with no failures 99.99% of the time.
R-20: IoT HTR shall alert the Operator how fast obstructions are moving with 95% accuracy.
R-21: The precision of distance-based calculations shall be at least 0.000001.
R-22: IoT HTR defect rate shall be less than 1 failure per 1000 hours of standard operation.
R-23: IoT HTR shall activate within 100 ms after the power button is pressed.
R-24: IoT HTR shall deactivate within 100 ms after the power button is pressed.
R-25: IoT HTR shall restart within 100 ms after the power button has been pressed for over 5 seconds.
R-26: IoT HTR shall meet or exceed 99.9% uptime.
R-27: IoT HTR sensor failures shall be identified in 0.01 seconds.
R-28: IoT HTR system failures shall be identified in 0.01 seconds.
R-29: The network of sensors shall be operational at least 99.99% of the time.

3.1.4 Other Non-Functional Requirements

R-30: IoT HTR shall be equipped with a touch screen display.
R-31: IoT HTR shall be equipped with a power button.
R-32: IoT HTR shall be equipped with an auxiliary storage device input.
R-33: IoT HTR shall be able to turn on/off regardless of the status (on/off) of the HTR train.
R-34: Operators and admins shall be able to start and stop the IoT HTR via a power button.

R-35: IoT HTR shall be able to restart when the power button is held down for over 5 seconds.
R-36: The Operator shall be prompted with a login screen after pressing on the power button.
R-37: The Display shall show recommendations from the IoT HTR.
R-38: The Display shall show functionalities as stated by the IoT HTR requirements.
R-39: The Display will display a continue normal operations message if no issues are presented by all the sensors.
R-40: IoT HTR shall send a warning message in the event of hardware or software failure.
R-41: The Log continuously logs data and sends it to a single log file to keep track of all events.
R-42: IoT HTR shall have a clock to attach timestamps to the log file entries.
R-43: IoT HTR shall be able to download data while at a station.
R-44: IoT HTR shall be able to upload operational data when at a station.

3.2 Functional Requirements

3.2.1 Standing Objects

R-45: There shall be four cameras, two in front and two in the back of the train.
R-46: There shall be two LiDAR sensors, one in the front and one in the back of the train.
R-47: The Cameras shall work with LiDAR sensors to take pictures of the environment.
R-48: The LiDAR sensors shall continuously take pictures to map the surroundings.
R-49: The Cameras shall provide obstruction pictures to the TSNR every 0.05 seconds.
R-50: The LiDAR sensors shall provide the distance of obstructions to the TSNR every 0.05 seconds.
R-51: IoT HTR shall receive obstruction data from the TSNR.
R-52: IoT HTR shall determine the position (front or back) of the obstruction to the HTR train.
R-53: IoT HTR shall display a warning message if an obstruction is detected via the Display.
R-54: IoT HTR shall display the distance of standing objects via the Display.
R-55: IoT HTR shall display that the object is moving at 0mph (as it is stationary) via the Display.
R-56: IoT HTR shall display if the standing object is in front or behind the train via the Display.
R-57: IoT HTR shall suggest to the Operator to slow down if there is an object within a 2.0 mile radius of the train via the Display.
R-58: IoT HTR shall suggest to the Operator to brake if there is an object within a 1.0 mile radius of the train via the Display.
R-59: IoT HTR shall suggest to the Operator to speed up when there is no object within a 2.0 mile radius of the train via the Display.

3.2.2 Moving Objects

R-60: There shall be four cameras, two in front and two in the back of the train.
R-61: There shall be two LiDAR sensors, one in the front and one in the back of the train.
R-62: There shall be four motion detection sensors distributed equally along the train's length.
R-63: The Cameras shall work with LiDAR sensors to take pictures of the environment.
R-64: The LiDAR sensors shall continuously take pictures and map the surroundings.
R-65: The Motion detection sensors shall continuously detect infrared energy (heat).
R-66: The Cameras shall provide obstruction pictures to the TSNR every 0.05 seconds
R-67: The LiDAR sensors shall provide the distance of obstructions to the TSNR every 0.05 seconds
R-68: The Motion detection sensors shall provide the speed of obstructions to TSNR every 0.05 seconds.

R-69: IoT HTR shall receive obstruction data from the TSNR.
R-70: IoT HTR shall determine the position (front or back) of the obstruction to the HTR train.
R-71: IoT HTR shall display a warning message if an obstruction is detected via the Display.
R-72: IoT HTR shall display the distance of moving objects via the Display.
R-73: IoT HTR shall display if the moving object is behind or in front of the train via the Display.
R-74: IoT HTR shall display the speed of the moving object via the Display.
R-75: IoT HTR shall suggest to the Operator to slow down if there is an object within a 2.0 mile radius of the train via the Display.
R-76: IoT HTR shall suggest to the Operator to brake if there is an object within a 1.0 mile radius of the train via the Display.
R-77: IoT HTR shall suggest to the Operator to speed up when there is no object within a 2.0 mile radius of the train via the Display.

3.2.3 Gate Crossings

A closed gate status means cars should stop at the railroad crossing as a train is approaching.
An opened gate means that cars can drive over the railroad crossing as no train is approaching.

R-78: There shall be one radar sensor located in the front of the train.
R-79: There shall be four cameras, two in the front and two in the back of the train.
R-80: The Radar sensors shall continuously emit radio waves and checks the change in frequency after omission.
R-81: The Cameras shall continuously take pictures to compare with a set of photos of crossing gates (opened and closed).
R-82: The Radar sensors shall provide the distance of a crossing gate to the TSNR every 0.05 seconds.
R-83: The Cameras shall provide the gate's status to the TSNR every 0.05 seconds.
R-84: IoT HTR shall receive gate crossing data from the TSNR.
R-85: IoT HTR shall display a warning message if the gate is open via the Display.
R-86: IoT HTR shall display the distance of the open gate crossing via the Display.
R-87: IoT HTR shall suggest to the Operator to slow down if the gate is open within a 3.0-mile Radius via the Display.
R-88: IoT HTR shall suggest to the Operator to brake if the gate is open within a 2.0-mile radius via the Display.
R-89: IoT HTR shall suggest to the Operator to speed up/maintain current speed when the gate is closed via the Display.

3.2.4 Wheel Slippage

R-90: IoT HTR is equipped with four tachometers on the four outermost train wheels.
R-91: IoT HTR is equipped with one GPS.
R-92: The Tachometers shall continuously measure the rotation speed of the train wheel in rpm.
R-93: The GPS shall continuously send and receive signals to calculate the distance and position of the train.
R-94: The Tachometers shall provide the wheels' rpm to the TSNR every 0.05 seconds.
R-95: The GPS shall provide the speed of the HTR train to the TSNR every 0.01 seconds.
R-96: IoT HTR shall receive wheel slippage data from the TSNR.
R-97: IoT HTR shall calculate the wheel rpm of the HTR train from the GPS data.
R-98: IoT HTR shall calculate the difference in the wheels' rpm given by the GPS and tachometer.
R-99: IoT HTR shall display a warning message if the train's wheels are slipping via the Display.
R-100: IoT HTR shall suggest to the Operator to slow down if the train is minorly slipping (5mph <

Δspeed < 10mph) via the Display.

R-101: IoT HTR shall suggest to the Operator to brake if the train is slipping severely (Δspeed > 10mph) via the Display.

R-102: IoT HTR shall suggest to the Operator to speed up with caution if slipping stops via the Display.

3.2.5 Weather Conditions

R-103: IoT HTR shall be equipped with one hygrometer.
R-104: IoT HTR shall be equipped with one thermometer.
R-105: IoT HTR shall be equipped with one anemometer.
R-106: IoT HTR shall be equipped with one rain gauge.
R-107: The Hygrometer shall continuously measure the humidity.
R-108: The Thermometer shall continuously measure the temperature.
R-109: The Anemometer shall continuously measure wind speeds.
R-110: The Rain gauge shall continuously measure the amount of rainfall.
R-111: The Hygrometer shall provide humidity data to the TSNR every 0.05 seconds.
R-112: The Thermometer shall provide temperature data to the TSNR every 0.05 seconds.
R-113: The Anemometer shall provide wind speed data to the TSNR every 0.05 seconds.
R-114: The Rain gauge shall provide current amount of rainfall data to the TSNR every 0.05 seconds.
R-#: IoT HTR shall receive weather data from the TSNR.
R-115: IoT HTR shall analyze for weather conditions by comparing it to a library of likely weather given weather data.
R-116: IoT HTR shall display likely weather conditions via the Display.
R-117: IoT HTR shall display a warning message if there are hazardous weather conditions via the Display.
R-118: IoT HTR shall suggest to the Operator to slow down if there is mild weather (rain, storm, snow) via the Display.
R-119: IoT HTR shall suggest to the Operator to slow down if there is ice via the Display.
R-120: IoT HTR shall suggest to the Operator to brake if there is heavy weather (rain, storm, snow) via the Display.
R-121: IoT HTR shall suggest to the Operator to put sand on the rails if ice is present via the Display.
R-122: IoT HTR shall suggest to the Operator to speed up/maintain current speed if nonhazardous weather conditions are present via the Display.

3.3 Hardware & Operating System

R-123: IoT HTR hardware shall be able to support at least 1,000 sensors.
R-124: IoT HTR shall support 5TB of data every day.
R-125: IoT HTR shall be able to import and export information obtained from operating.
R-126: IoT HTR shall be updated yearly with patches in between.

3.3.1 Operating System

R-127: IoT HTR shall be equipped with Microsoft Windows 10 IoT Enterprise
R-128: IoT HTR shall be equipped with Microsoft Windows 10, 64-bit

3.3.2 Processor

R-129: IoT HTR shall be equipped with ARM processor

3.3.3 RAM

R-130: IoT HTR shall be equipped with 2GB of RAM

3.3.4 Graphics Card

R-131: IoT HTR shall be equipped with a GPU that supports DirectX 9

3.3.5 Storage

R-132: IoT HTR shall be equipped with 32GB of storage

3.3.6 Sensors

R-133: IoT HTR shall be equipped with a Tachometer - measures the rpm of a wheel
R-134: IoT HTR shall be equipped with a Hygrometer - measures humidity
R-135: IoT HTR shall be equipped with a Thermometer - measures temperature
R-136: IoT HTR shall be equipped with a Anemometer - measures wind speed
R-137: IoT HTR shall be equipped with a Rain gauge - measures/collects the amount of rain
R-138: IoT HTR shall be equipped with LiDAR sensors - maps surrounding environment
R-139: IoT HTR shall be equipped with Radar sensors - detect is the status of crossing gate
R-140: IoT HTR shall be equipped with Cameras - for gate status and LiDAR
R-141: IoT HTR shall be equipped with a GPS - give global positioning
R-142: IoT HTR shall be equipped with Motion detection sensors - for LiDAR and moving objects

4. Requirements Modeling

4.1 Use Cases

Yellow warning messages mean slow down.
Red warning messages mean stop and apply brakes.

Use case 1:	Initialize IoT HTR
Primary actor:	Operator
Secondary actors:	IoT HTR
Goal in context:	Turn on IoT HTR
Preconditions:	IoT HTR is off
Trigger:	Operator decides to initialize IoT HTR, that is, to press the power button

Scenario:
1. Operator presses the power button
2. Log begins to record events starting at the time of initialization
3. Display turns on
4. IoT HTR displays the login screen
5. Operator types in ID and password
6. IoT HTR displays operations of the HTR train

Exceptions:
1. Incorrect login: Warning message "Incorrect user ID or password" is displayed, Operator re-enters correct ID/password
2. Failed to log in five times: Warning message "Incorrect user ID or password. Wait 5 minutes" is displayed
3. Fails to initialize: Display does not turn on, abort use case. Admin is required to fix the issue(s)

Use case 2:	Initialized IoT HTR with admin privileges
Primary actor:	Admin
Secondary actors:	IoT HTR
Goal in context:	Turn on IoT HTR
Preconditions:	IoT HTR is off
Trigger:	Admin decides to initialize IoT HTR, that is, to press the power button

Scenario:
1. Admin presses the power button
2. Log begins to record events starting at the time of initialization
3. IoT HTR touch screen display turns on
4. IoT HTR displays the login screen
5. Admin types in admin ID and password
6. IoT HTR displays system configurations (software os)

Exceptions:
1. Incorrect login: Warning message "Incorrect user ID or password" is displayed, Admin re-enters correct ID/password
2. Failed to log in five times: Warning message "Incorrect user ID or password. Wait 5 minutes" is displayed
3. Fails to initialize: Display does not turn on, abort use case. Admin is required to fix the issue(s)

Use case 3:	Obstruction detection
Primary actor:	Proximity sensors
Secondary actors:	IoT HTR
Goal in context:	Notify Operator for any encounters of obstructions
Preconditions:	IoT HTR is initialized
Trigger:	Proximity sensors detect that there is an obstruction

Scenario:

1. LiDAR sensors detect obstructions that are within 2 miles and reports it to TSNR
2. Motion detection sensors detect the speed of the obstruction and reports it to TSNR
3. Cameras report pictures of the detected obstructions to TSNR
4. TSNR sends collected data to IoT HTR.
5. IoT HTR determines if the obstruction pictures were taken by front or back Cameras.
6. The Display displays if there is an obstruction.
7. The Display displays the speed of the obstruction.
8. The Display displays the position of the obstruction.
9. The Display displays the distance of the obstruction.
10. The Display displays a yellow warning message when the obstruction's distance is between 1 mile and 2 miles (1 mile < distance < 2 miles).
11. The Display displays a red warning message when the obstruction's distance is less than 1 mile (distance < 1 mile).
12. Log records obstructions.

Use case 4:	Crossing gate status detection
Primary actor:	Crossing gate sensors
Secondary actors:	IoT HTR
Goal in context:	Notify Operator of the gate status
Preconditions:	IoT HTR is initialized
Trigger:	Crossing gate based sensors detect the status of upcoming crossing gates

Scenario:

1. Radar sensors detect closed gates that are within 2 miles and reports it to the TSNR
2. Cameras send pictures of detected closed gates to the TSNR
3. TSNR sends collected data to IoT HTR.
4. The Display displays the status (opened/closed) of the crossing gate
5. The Display displays the distance of the open gate.
6. The Display displays a yellow warning message when the open gate's distance is between 2 miles and 3 miles (2 miles < distance < 3 miles).
7. The Display displays a red warning message when the open gate's distance is less than 2 mile (distance < 2 mile).
8. Log records gate status.

Use case 5:	Wheel slippage detection
Primary actor:	Wheel sensors
Secondary actors:	IoT HTR
Goal in context:	Notify Operator for any encounters of wheel slippage
Preconditions:	IoT HTR is initialized.

Trigger:	Wheel based sensor detects wheel slippage

Scenario:

1. The tachometer detects the rpm of the HTR train wheels and reports it to TSNR.
2. GPS detects the speed of the HTR train and reports it to TSNR
3. TSNR sends collected data to IoT HTR.
4. IoT HTR calculates wheel rpm of the HTR train from GPS data.
5. IoT HTR calculates the difference in wheel rpm given by the tachometer and GPS.
6. The Display will display the difference in the speed of the wheels (wheel slippage)
7. The Display displays a yellow warning message when the difference is between 5mph and 10mph ($5\text{mph} < \Delta\text{speed} < 10\text{mph}$).
8. The Display displays a red warning message when the difference is between greater than 10mph ($\Delta\text{speed} > 10\text{mph}$).
9. The Display will suggest to the conductor to accelerate with caution once the wheel slippage has stopped.
10. Log records slippage

Use case 6:	Hazardous weather conditions detection
Primary actor:	Weather sensors
Secondary actors:	IoT HTR
Goal in context:	Notify Operator of hazardous weather conditions
Preconditions:	IoT HTR is initialized
Trigger:	Weather-based sensors detect hazardous weather conditions.

Scenario:

1. Hygrometer measures humidity and reports it to TSNR
2. Thermometer measures temperature and reports it to TSNR
3. Anemometer measures wind speeds and reports it to TSNR
4. Rain Gauge measures the current amount of rainfall and reports it to TSNR
5. TSNR sends collected data to IoT HTR.
6. The Display displays the presence of mild weather conditions (rain, storm, snow)
7. The Display displays the presence of heavy weather conditions (rain, storm, snow)
8. The Display displays the presence of ice.
9. The Display displays a yellow warning message for mild weather conditions or ice.
10. The Display displays a red warning message for heavy weather conditions.
11. The Display displays a “Put sand on the railroads” message when ice is present.
12. The Display will suggest to the operator to accelerate once non-hazardous weather conditions are present.
13. Log records hazardous weather conditions

Use case 7:	Uninitialize IoT HTR
Primary actor:	Operator
Secondary actors:	IoT HTR
Goal in context:	Turn off IoT HTR
Preconditions:	IoT HTR is on
Trigger:	Operator decides to uninitialize IoT HTR, that is, to press the power button

Scenario:

1. Operator presses the power button
2. IoT HTR displays the “Do you want to turn the power off” message

3. Operator presses “yes”
4. IoT HTR timestamps when IoT HTR was uninitiated into the logs
5. IoT HTR Display turns off

Exceptions:

1. “No” is selected: IoT HTR closes the message and continues to display the operations of the HTR train

---

Use case 8:	Download logs
Primary actor:	Admin
Secondary actors:	IoT HTR
Goal in context:	To check and download operational logs
Preconditions:	There is data in the logs, that is, IoT HTR is been initialized at some point; IoT HTR is initialized
Trigger:	Admin decides to download the operational logs to check data, that is, to connect an auxiliary storage device

Scenario:

1. Admin connects an auxiliary storage device to IoT HTR
2. IoT HTR detects an auxiliary storage device
3. IoT HTR request for admin ID/password
4. IoT HTR requests log(s) from log
5. Log copies the log(s) onto the storage device
6. IoT HTR displays “Log file(s) was successfully copied” message

Exceptions:

1. Incorrect admin login: Warning message “Incorrect user ID or password” is displayed, Operator re-enters
2. Failed admin login five times: Warning message “Incorrect admin ID or password. Wait 5 minutes” is displayed
3. Error copying log file(s) onto the auxiliary storage device: Warning message “Error copying log file(s) onto storage device” is displayed, abort use case

---

Use case 9:	Change configurations
Primary actor:	Admin
Secondary actors:	IoT HTR
Goal in context:	Change the configuration of the HTR train
Preconditions:	HTR train has IoT HTR system installed, IoT HTR is initialized
Trigger:	Admin decides to change the configurations of the HTR train

Scenario:

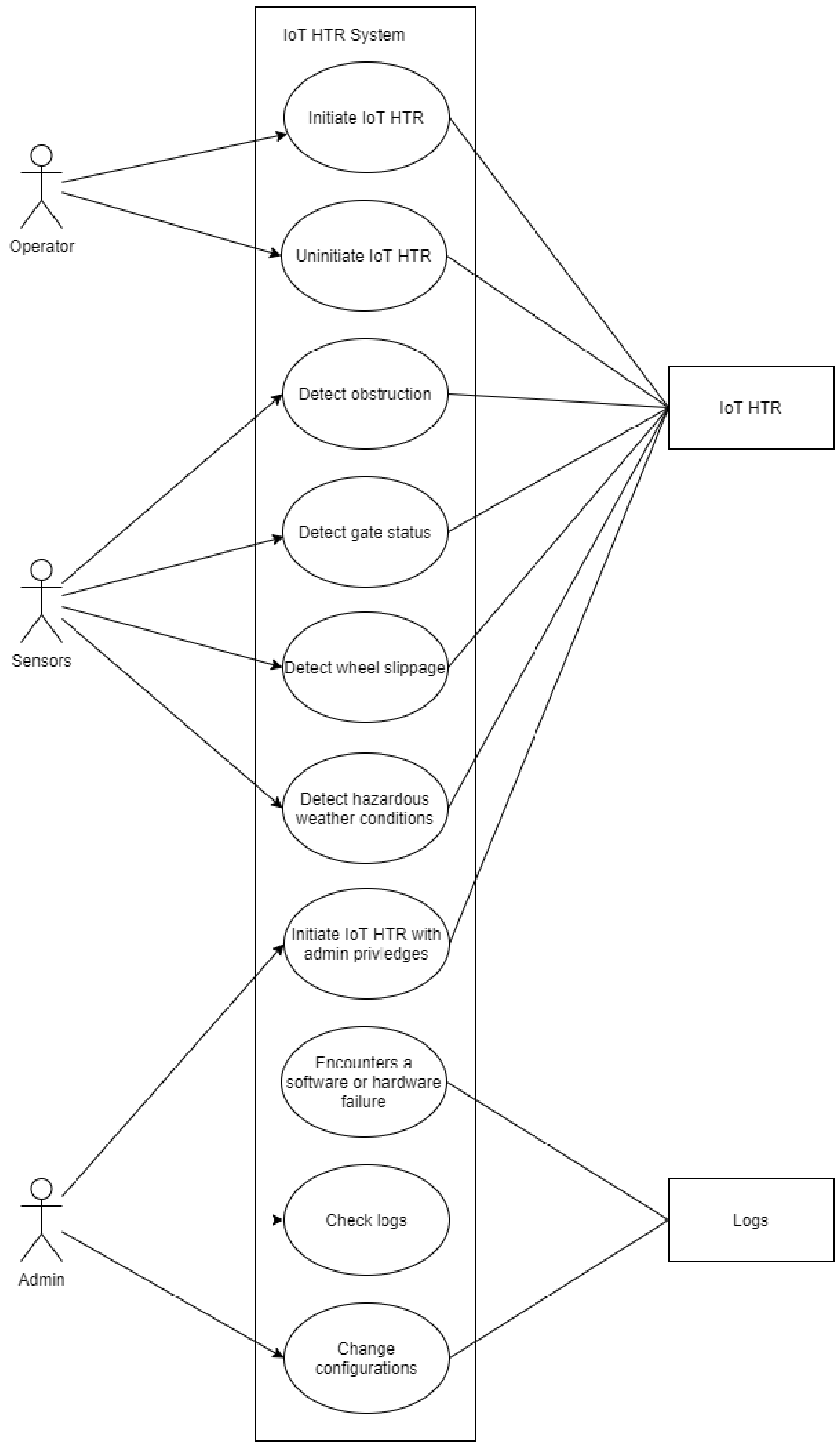
1. Admin supplies correct admin ID/password to IoT HTR
2. IoT HTR prompts Admin if they would like to make changes to configurations
3. Admin selects “yes”
4. Admin makes appropriate changes to IoT HTR
5. Log records changes.
6. Admin confirms changes and updates the system with new changes
7. Admin closes the configuration interface

Exceptions:

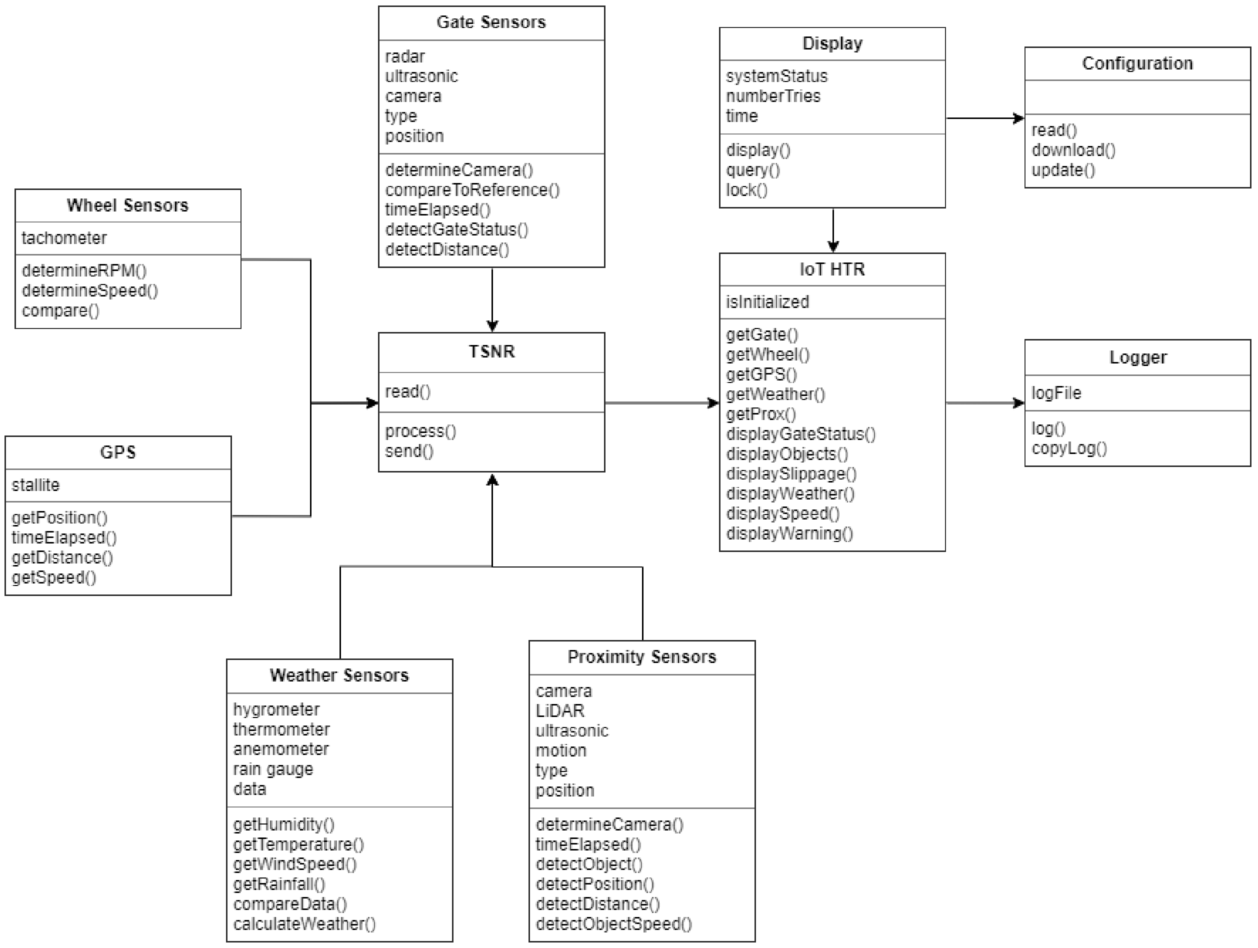
1. Incorrect login: Warning message “Incorrect user ID or password” is displayed, Operator re-enters correct ID/password.

2. Failed to log in five times: Warning message “Incorrect user ID or password. Wait 5 minutes” is displayed.
3. Fail to make configurations: Warning message “Failed to configure” is displayed, abort use case.
4. “No” is selected: IoT HTR closes the configuration interface and continues to display the operations of the HTR train.

4.2 Use Case Diagram



4.3 Class Diagram



4.4 CRC Model Index Cards

Operator	
User who has limited access to IoT HTR	
Responsibility	Collaborator
Initiate IoT HTR	IoT HTR
Uninitiate IoT HTR	

Admin

User who has complete access to IoT HTR	
Responsibility	Collaborator
Check logs	Log
Change configurations	Configurations

IoT HTR	
Takes in all the data from the TSNR, displays it, and determines if the operator needs to be alerted about hazards	
Responsibility	Collaborator
Initiate IoT HTR	
Uninitate IoT HTR	
Continuously log	Log
Display obstructions if present	Proximity sensors via TSNR
Display speed	GPS via TSNR
Display weather	Weather sensors via TSNR
Display slippage if present	Wheel sensors via TSNR
Display gate status when applicable	Gate sensors via TSNR

TSNR	
Takes in all the data from the sensors, processes it, and sends it to IoT HTR	
Responsibility	Collaborator
Send proximity (obstruction) data	Proximity sensors
Send global positioning data	GPS
Send weather data	Weather sensors

Send wheel slippage data	Wheel sensors
Send gate crossing data	Gate sensors

Proximity sensors	
Sensors that can detect objects (stationary and moving) within a 2-mile radius of the train	
Responsibility	Collaborator
Check if there are stationary obstructions	IoT HTR
Check if the stationary object is in the front or back of the HTR train	
Display the distance of stationary objects	
Check if there are moving obstructions	
Check if the moving object is in front or back of the HTR train	
Display the distance of moving objects	
Display the speed of obstructions	
Display warning message for obstructions and their position relative to the HTR train	
Display suggestion	

Wheel sensors	
Sensors that can detect if the train's wheels are slipping	
Responsibility	Collaborator
Determine the rpm of the train's wheels	IoT HTR
Determine the speed based on the wheel's rpm	

Compare wheel sensor's speed with GPS's speed to check for slippage	
Display warning message for slippage	
Display suggestion	

Gate sensors	
Sensors that can detect a crossing gate' status within a 3-mile radius of the train	
Responsibility	Collaborator
Determines if crossing gates are opened/closed	IoT HTR
Determine the distance of a crossing gate	
Display warning message for open crossing gate	
Display suggestion	

Weather sensors	
Sensors that take in data regarding the current weather and its impact on the train	
Responsibility	Collaborator
Determine likely weather conditions	IoT HTR
Display likely weather conditions	
Display warning message for hazardous weather conditions	
Display suggestion	

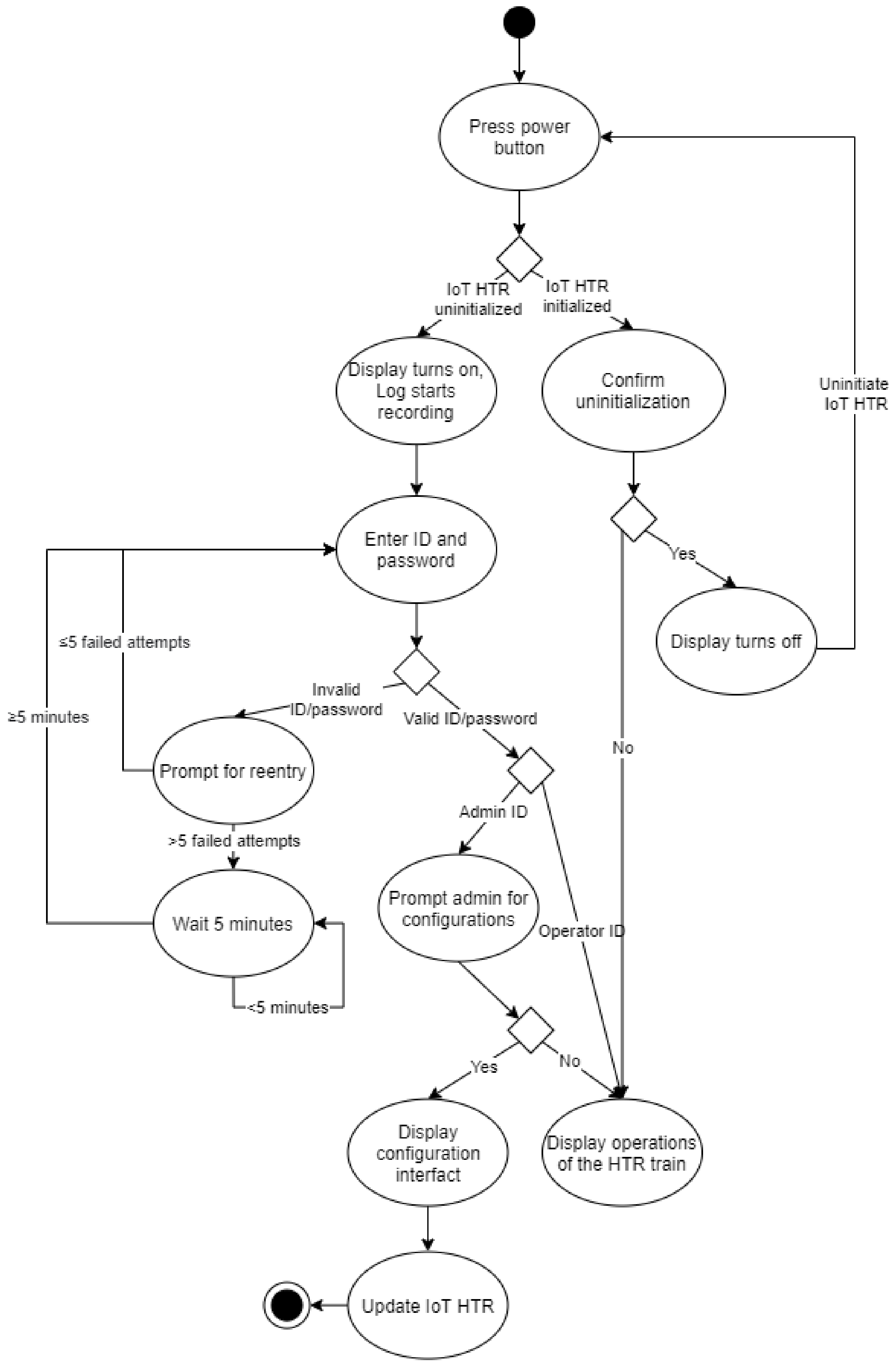
GPS

Gets the global positioning of the HTR train	
Responsibility	Collaborator
Get the speed of the HTR train	IoT HTR

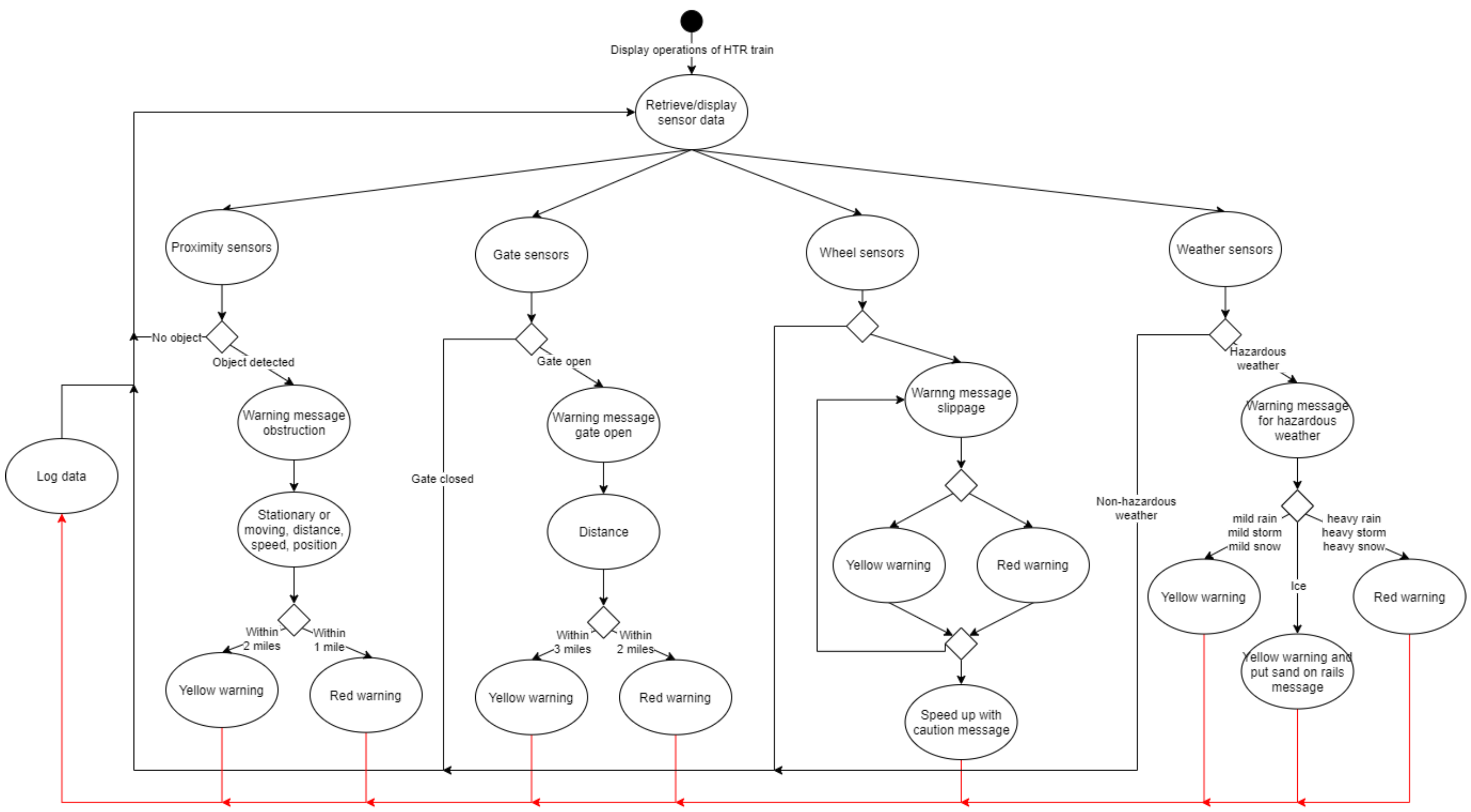
Log	
Logs all events while IoT HTR is on	
Responsibility	Collaborator
Get current time Log events	IoT HTR

Configurations	
User configurations for IoT HTR	
Responsibility	Collaborator
Read admin input to change/update system settings	IoT HTR

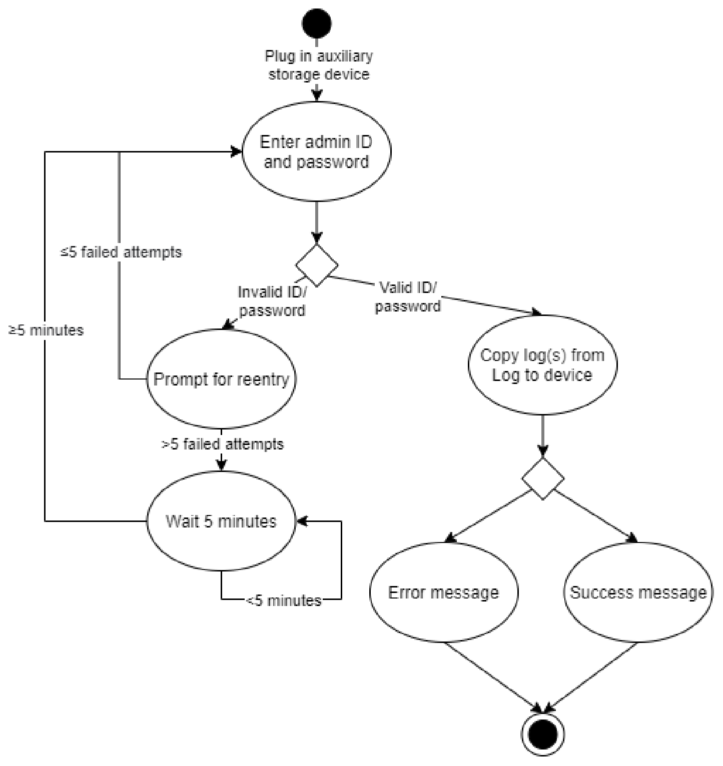
4.5 Activity Diagram



Precondition: IoT HTR has already been initialized

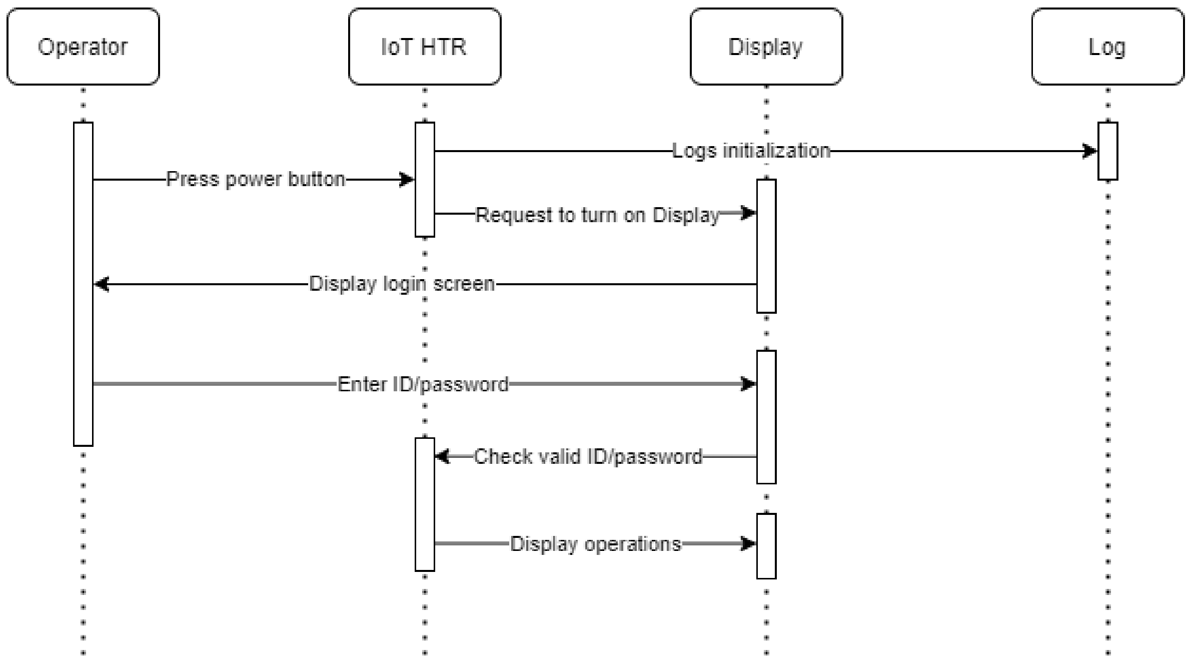


Precondition: IoT HTR has already been initialized

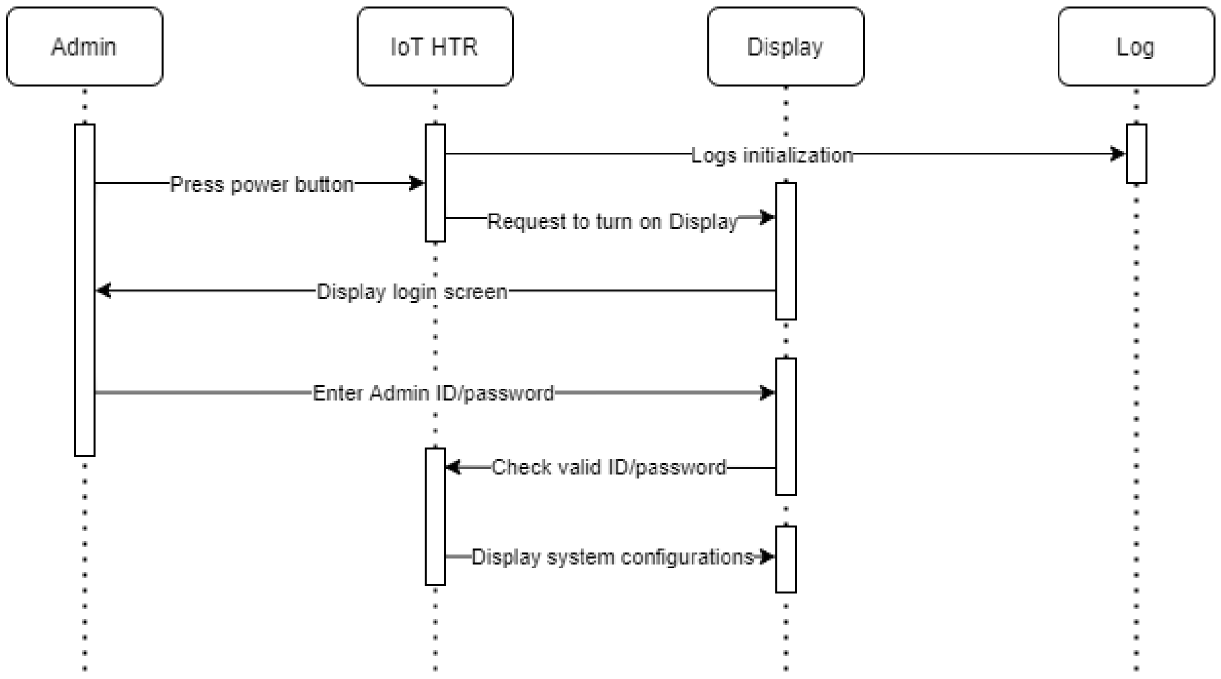


4.6 Sequence Diagrams

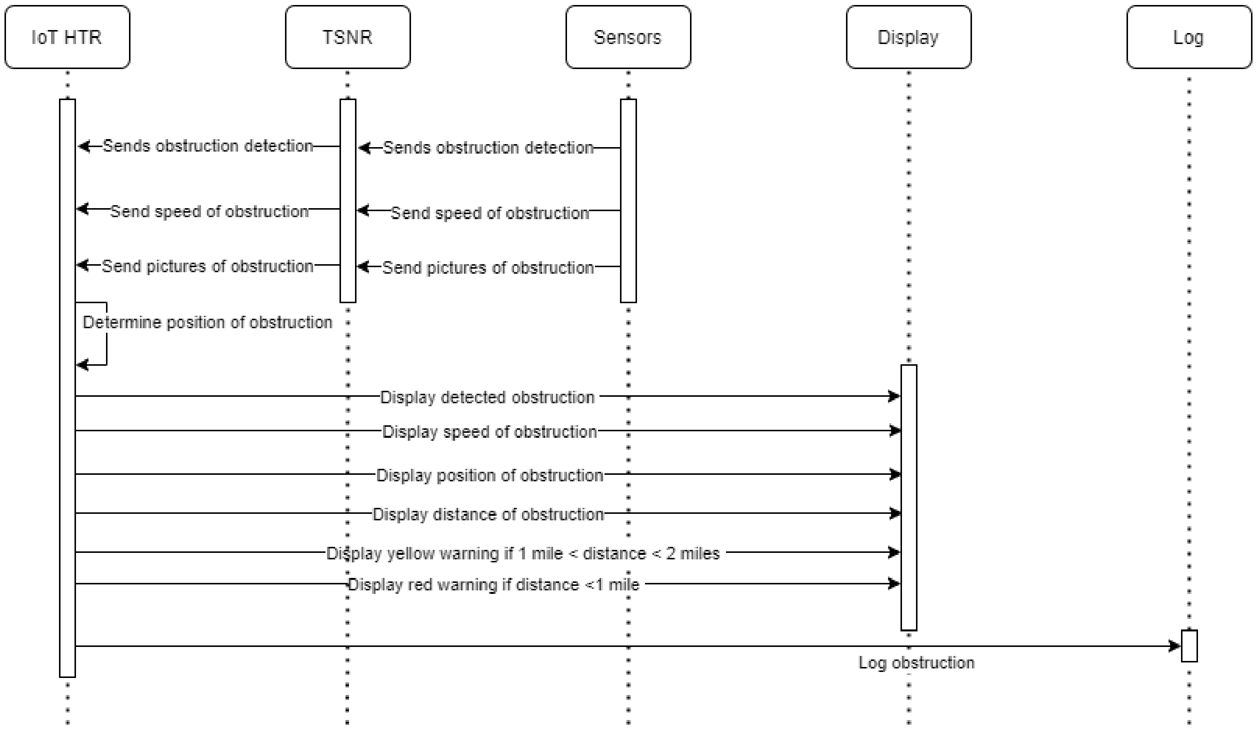
Use case 1:



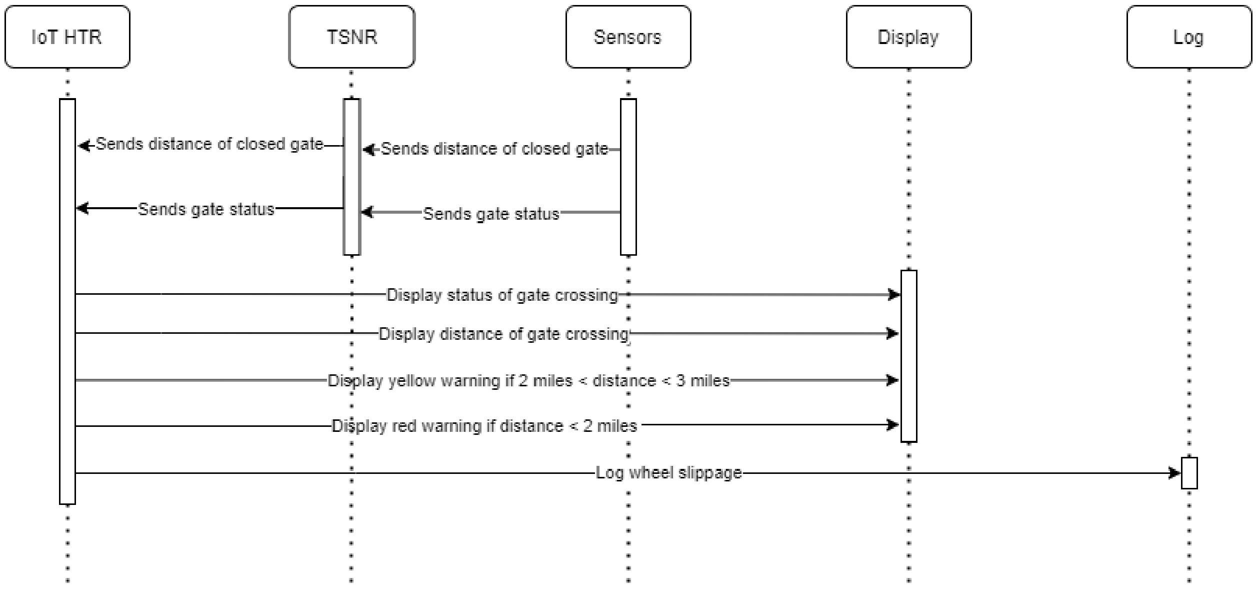
Use case 2:



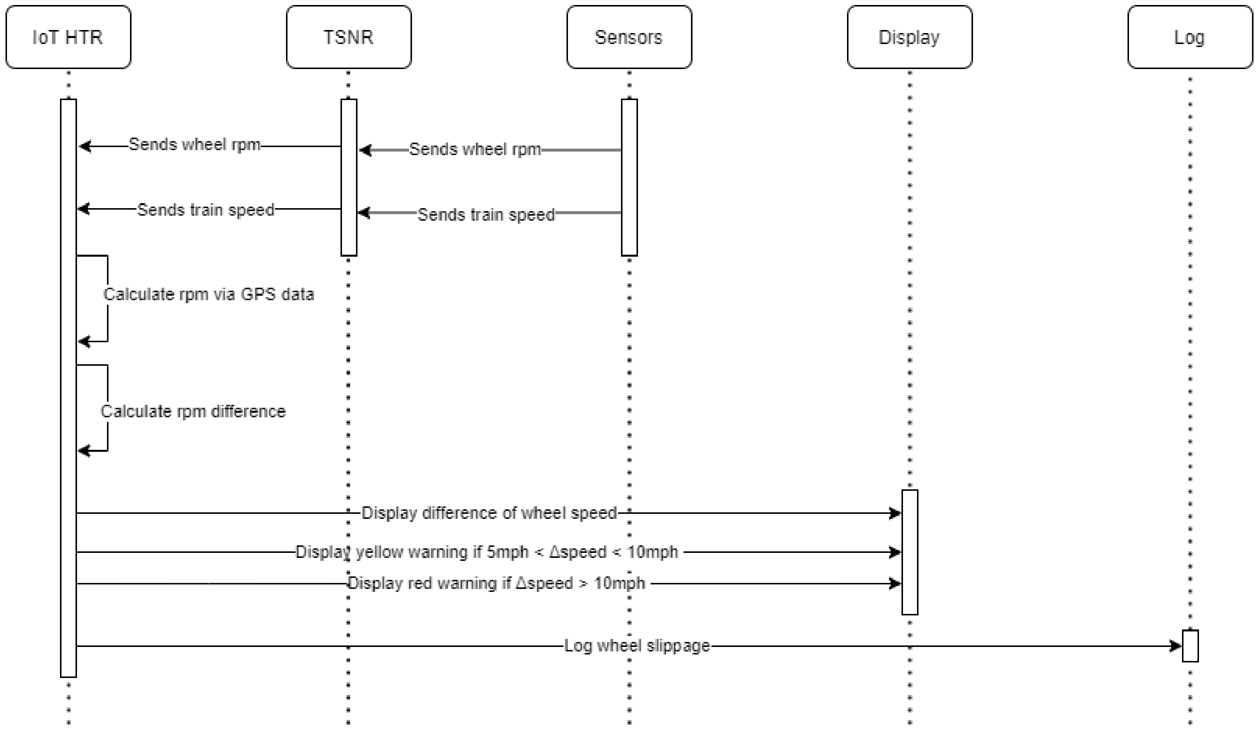
Use case 3:


IoT HTR
TSNR
Sensors
Display
Log
Sends obstruction detection
Sends obstruction detection
Send speed of obstruction
Send speed of obstruction
Send pictures of obstruction
Send pictures of obstruction
Determine position of obstruction
Display detected obstruction
Display speed of obstruction
Display position of obstruction
Display distance of obstruction
Display yellow warning if 1 mile < distance < 2 miles
Display red warning if distance <1 mile
Log obstruction

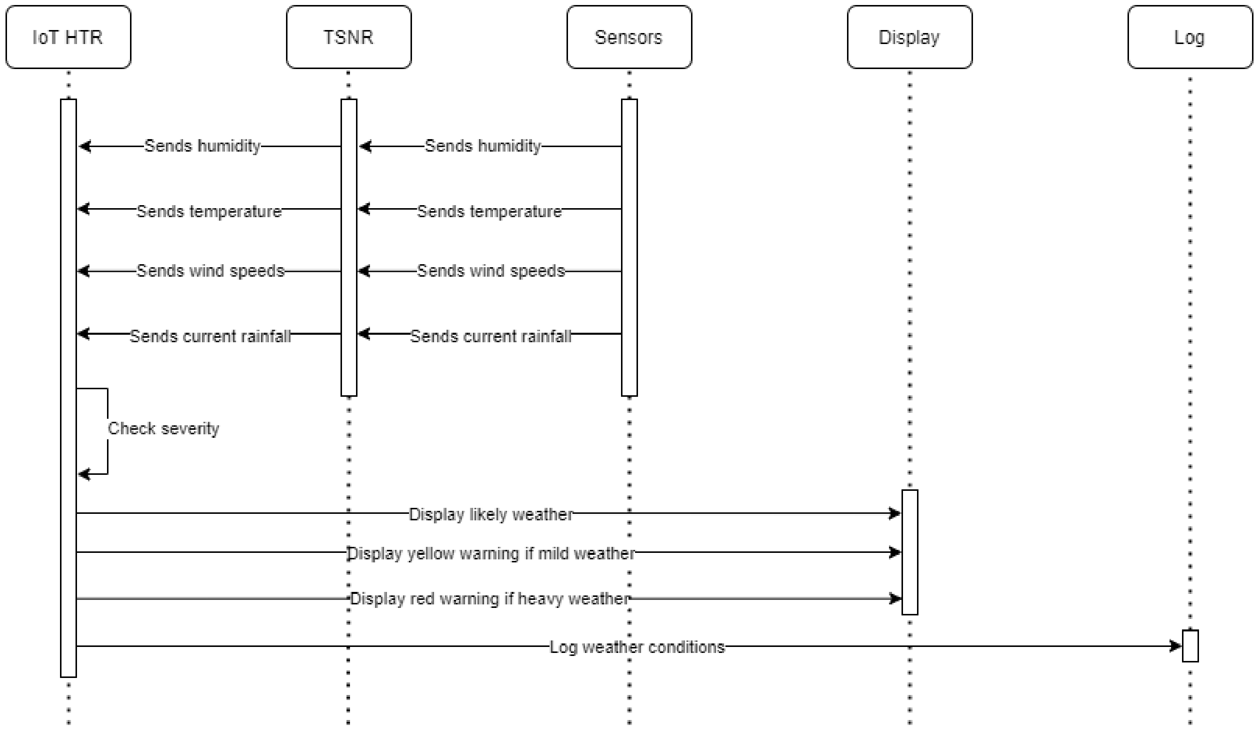
Use case 4:


IoT HTR
TSNR
Sensors
Display
Log
Sends distance of closed gate
Sends distance of closed gate
Sends gate status
Sends gate status
Display status of gate crossing
Display distance of gate crossing
Display yellow warning if 2 miles < distance < 3 miles
Display red warning if distance < 2 miles
Log wheel slippage

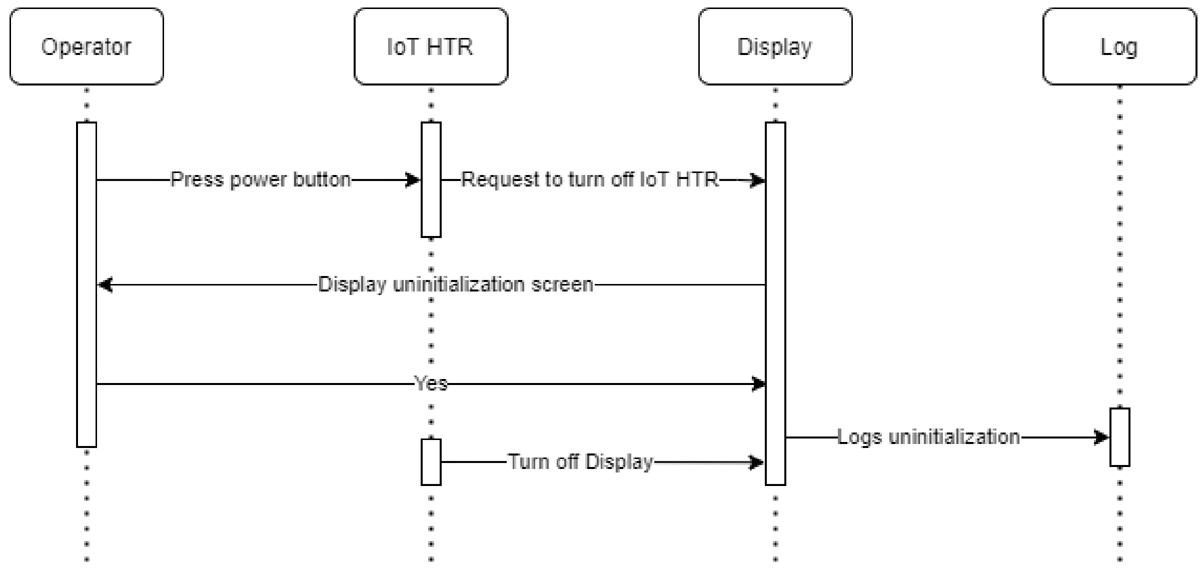
Use case 5:


IoT HTR
TSNR
Sensors
Display
Log
Sends wheel rpm
Sends wheel rpm
Sends train speed
Sends train speed
Calculate rpm via GPS data
Calculate rpm difference
Display difference of wheel speed
Display yellow warning if 5mph < Δspeed < 10mph
Display red warning if Δspeed > 10mph
Log wheel slippage

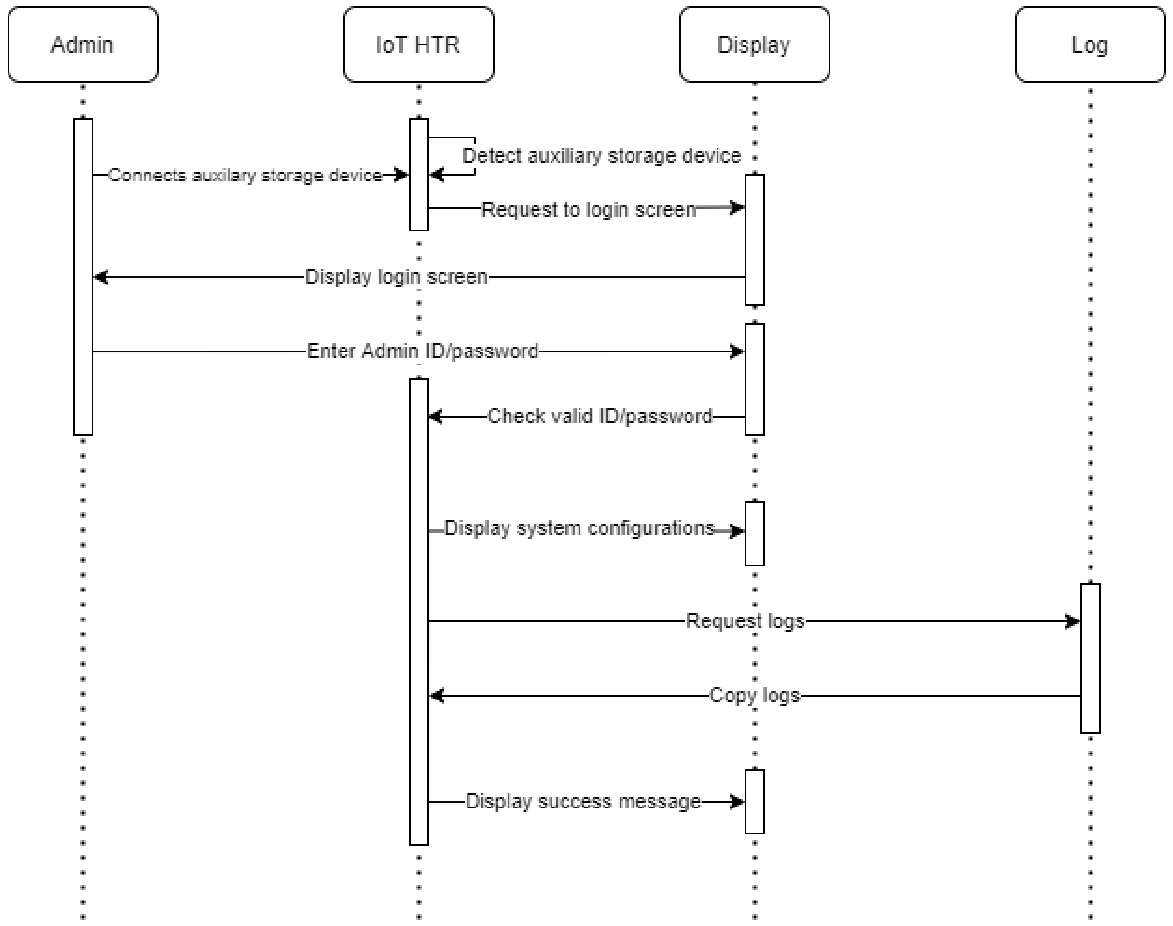
Use case 6:



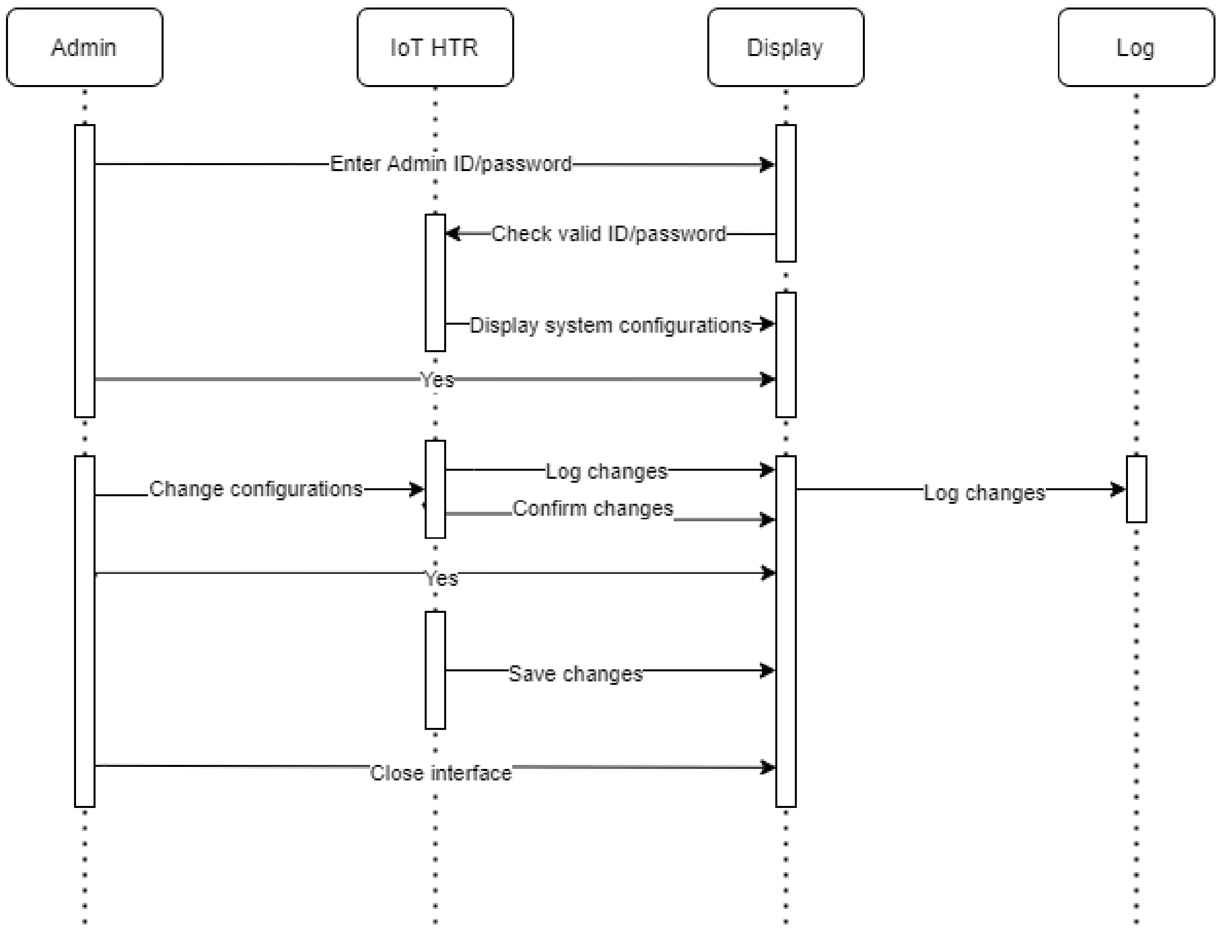
Use case 7:



Use case 8:

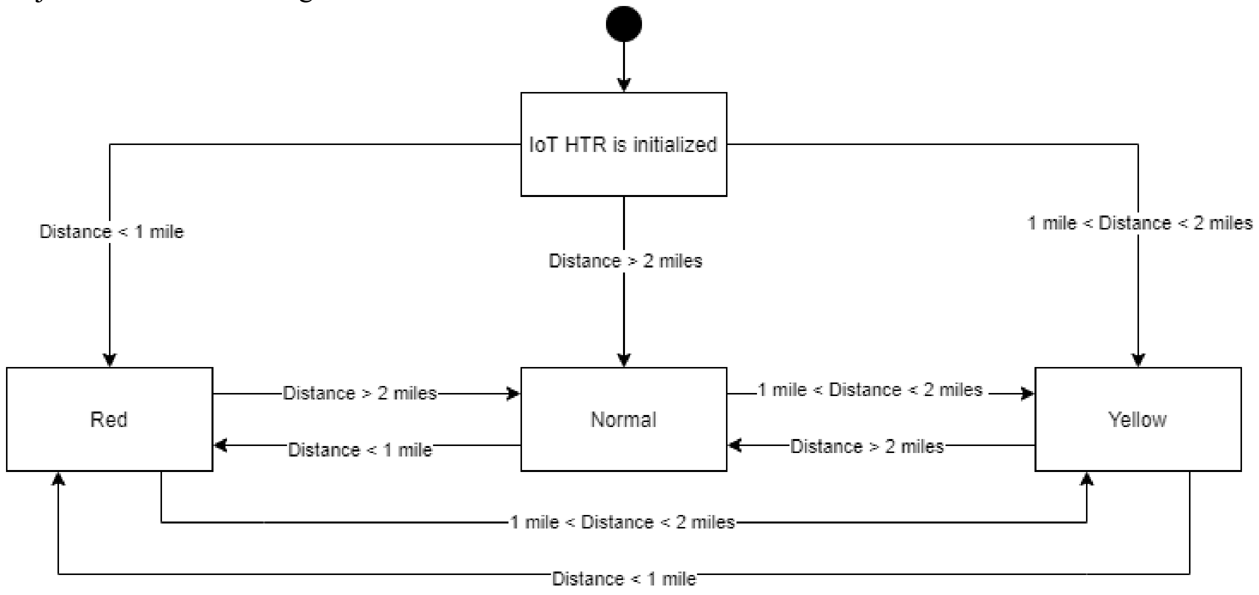


Use case 9:

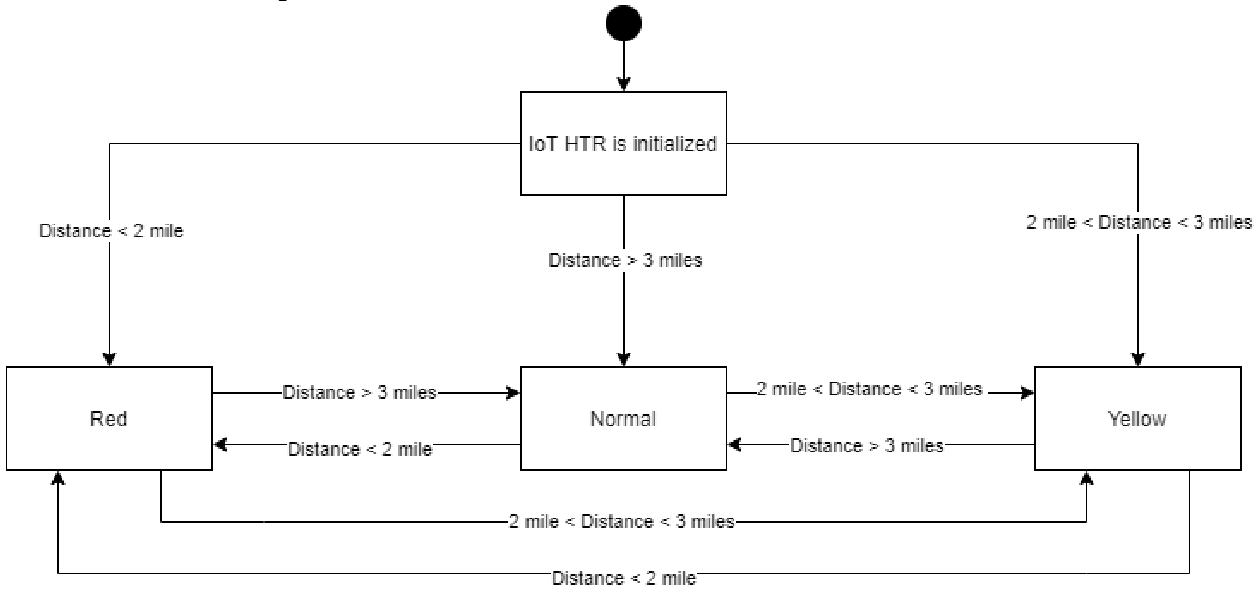

Admin
IoT HTR
Display
Log
Enter Admin ID/password
Check valid ID/password
Display system configurations
Yes
Change configurations
Log changes
Confirm changes
Log changes
Yes
Save changes
Close interface

4.7 State Diagram

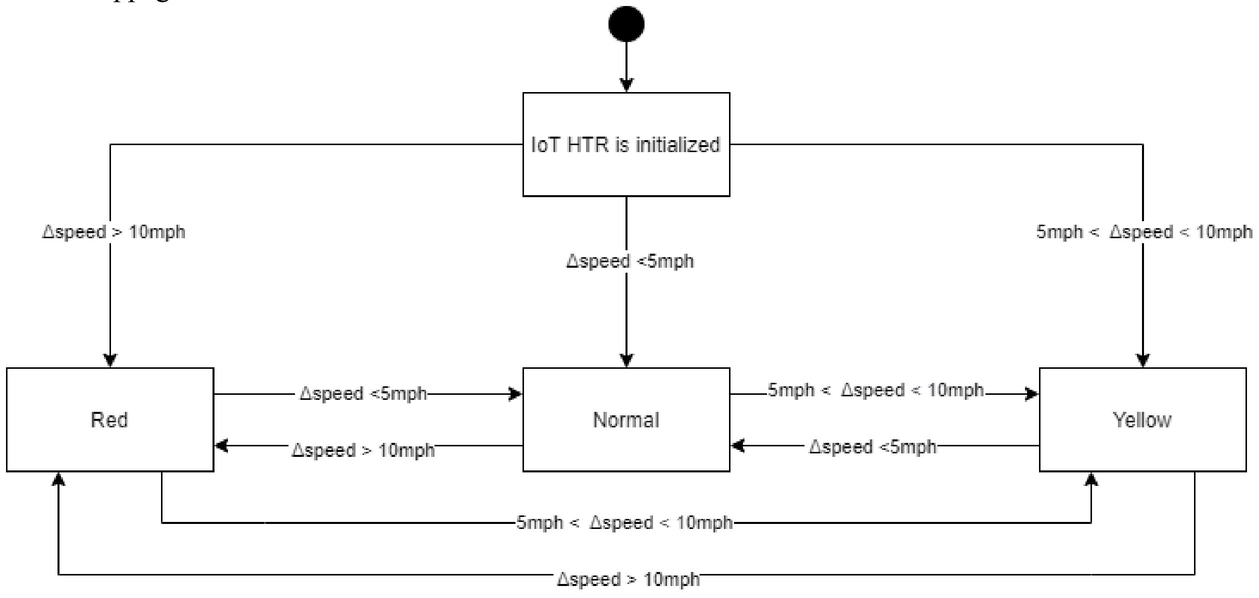
Object detection state diagram:



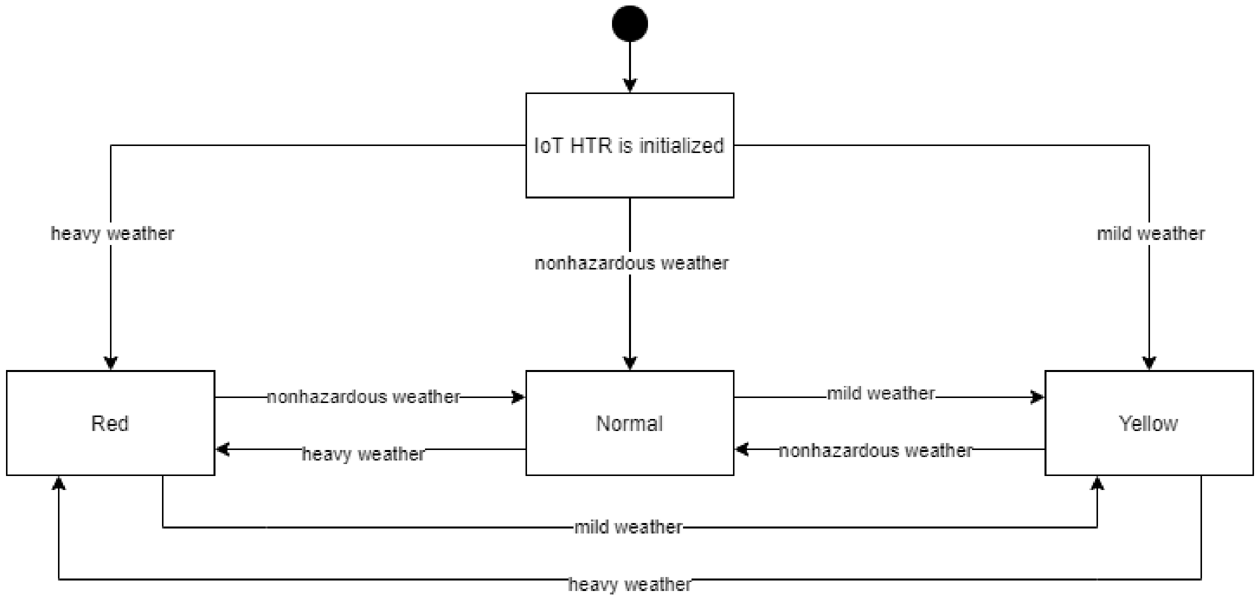
Gate detection state diagram:



Wheel slippage detection:



Hazardous weather detection:



5. Software Architecture

Object Oriented Architecture

Pros

- Less debugging required as the language will support type checking
- Ease of maintainability due to modular design
- Faster development because "nouns" are already stated and reviewed
- Lower cost of development because of object oriented analysis and design beforehand
- More human readable code
- Work can be distributed to classes/modules

Cons

- Object oriented programs typically come at the cost of a larger size
- Object oriented programs typically come at the cost of performance
- Objects may not cohere to procedures resulting in conflicts
- Synchronization of real time between sensors will be hard to implement

Data Centered Architecture

Pros

- Promotes integrability therefore allows existing components to be changed and new client components to be added
- Client components operate independently
- Provides data integrability, backup, and restore features
- Provides scalability and reusability of agents as they do not have direct communication with each other
- Reduces overhead of transient data between software components
- Keep track of data permanently (not needed for IoT, events come and go)

Cons

- It is more vulnerable to failure and data replication or duplication is possible
- High dependency between data structure of data store and its agents
- Changes in data structure highly affect the clients
- Evolution of data is difficult and expensive
- Cost of moving data on network of distributed data

Data Flow Architecture (Pipes and Filters)

Pros

- Provides reusability and simplifies system maintenance
- Clear divisions between any two filters connected by a pipe
- Supports both sequential and parallel execution making the architecture more flexible
- Allows for the use of threads and pipes and "concurrency" by doing multiple actions at once
- Easily modified and low coupling between filters
- The concurrency allows for high throughput (good for excessive data processing)
-

Cons

- Not suitable for dynamic interactions
- Does not provide a way for filters to cooperatively interact to solve a problem
- This architecture does not allow for much dynamic change
- There exists overhead of data transformation between filters

Call Return Architecture

Pros

- This architecture is easily modified
- Accounts for change a lot more than most due to its modular architecture
- Simple to analyze the direction of data flow and control
- Easy to scale

Cons

- Parallel processing is difficult
- Possibly difficult to distribute across machines
- Exceptions are more difficult and awkward to deal with
- Inadequate attention to data structures

Model View Controller Architecture

Pros

- Development of the application becomes fast
- Easy for multiple developers to collaborate and work together
- Easier to update the application
- Easier to debug due to multiple levels properly written in the application
- Easy to test, maintain, and scale

Cons

- Difficult to understand the architecture
- Must have strict rules on methods
- View is dependent on the Controller and Model
- Model oftentimes does much of the work

Layered Architecture

Pros

- Separations of concerns
- Very testable as each layer has less test cases
- Changes in one layer will not affect downstream layers
- Easy to change out layers for other implementations
- Simple to implement
- Easily to change, scale, maintain as it is very flexible

Cons

- Increases cost if there are too many layers
- Performance becomes slower with more layer
- Leaky abstraction can disturb layers

Our team decided to choose an Object Oriented Architecture over the other choices listed above because it was the architecture we were all the most familiar with and we believed that it fit the needs of our project the most. Overall an Object Oriented Architecture would easily be implemented and translated into code because many nouns could be turned into objects. A Data Flow Architecture was considered because it makes sense to view the project as a system with well defined, easily identifiable states/outputes are are the direct result of sequentially transforming a well defined, easily identifiable input. A Call and Return Architectures was also considered because the order of computations in our project is fixed, and components can make no useful progress while awaiting the results of requests to other components. While many of the other architectures support reusability and modifications, we found that they were often held back by poor performance which is unfavorable in a project that is mission critical (data flow, data centered, layered architectures). While call return and model view controller architectures were not held back by their performance, our lack of understanding would slow our progress and/or make it difficult for us to implement the architecture effectively.

6. Code

7. Test Cases

8. Issues

9. Cost

Costs for any project can be daunting and often difficult to calculate because the project expands and evolves over time. We can give a rough cost estimation of this project based on a sample of devices one would buy. These costs are subject to change and there are multiple companies with varying prices that sell these sensors. It also should be noted that a large portion of the project will be software which does not have any external costs.

Sensor	**Purpose**	**Unit Price**
Anemometer	Wind speed	$50-$800
Hygrometer	Humidity	$200
Rain Gauge	Rain	$300
Thermometer	Temperature	$200
Tachometer	Wheel's RPM	$50-$900
LiDAR	3D mapping, detect obstructions	$500-$1,000
Friction/Shear Force	Friction	$6,000-$8,000
System of solid state circuits and pressure sensors	Gate	$1000+
IoT Display and software	UI/UX	
GPS	Position/speed	$100-$1000
Assemble		
Radar	Status of crossing gate	
Cameras	Photos, object recognition	
Motion detection sensors	Detect moving objects	