

Stevens Institute of Technology

TRAINX **ARCHITECTS**

TrainX Architects
IoT Hug the Rail
v.3.1

Bonnie Nguyen, Roma Razdan, Matthew Viafora, and Michael Zylka
Team 3 Section D
Software Development Process
Professor Peyrovian

Table of Contents

Section 1:	2
Introduction:	2
Timeline:	2
Model:	3
Section 2:	4
Problem Statement:	4
System Overview	4
Data Required	5
Technology Required	5
Section 3:	6
Requirements	6
3.1 Non-Functional Requirements	6
3.2 Functional Requirements	7
3.3 Hardware & Operating System	8
Section 4:	9
Requirements Analysis Modeling	9
4.1 Use Cases	9
Use Case 6: IoT Engine Suggests Change of Speed to Accommodate for Gate Closing/Opening Times	11
Use Case 8: IoT is Deactivated	12
4.2 Use Case Diagram	13
4.3 Class-Based Modeling	14
4.4 CRC Modeling/Cards	15
4.5 Activity Diagrams	16
4.6 Sequence Diagrams	17
4.7 State Diagram	18

Section 1:

Introduction:

Using the internet of things, our group will devise a method to allow decisions to be made locally in absence or failure of cellular and wifi. This will be implemented into the Train System (Hug the Rail) as a means of making it safer, less costly, and more efficient.

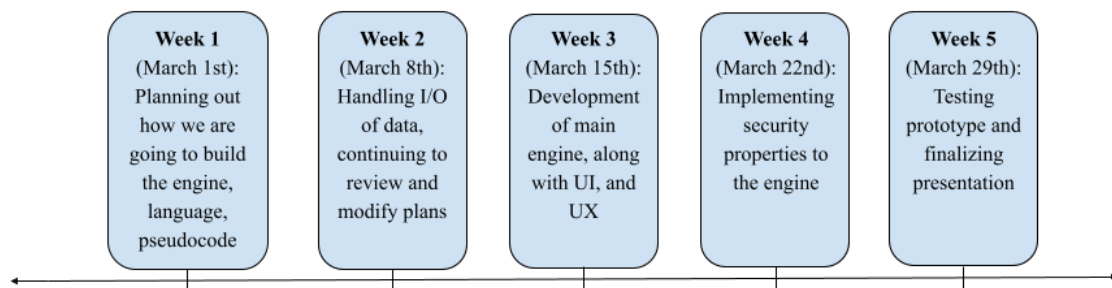
Our team consists of several smart, problem-solving, communicative individuals who strive to create an innovative design. Mike is a creative individual who will take the proper initiative to learn about the best possible solution for our project. Matt is a member who is very detail oriented and will ensure that the work we produce is truly working to create a safer, less costly, and more efficient solution. Bonnie is someone who is very task oriented and will help the overall flow of the project move smoothly, and Roma is someone who is people-oriented and will ensure that this model will be user-friendly and meet all the stakeholders' criteria.

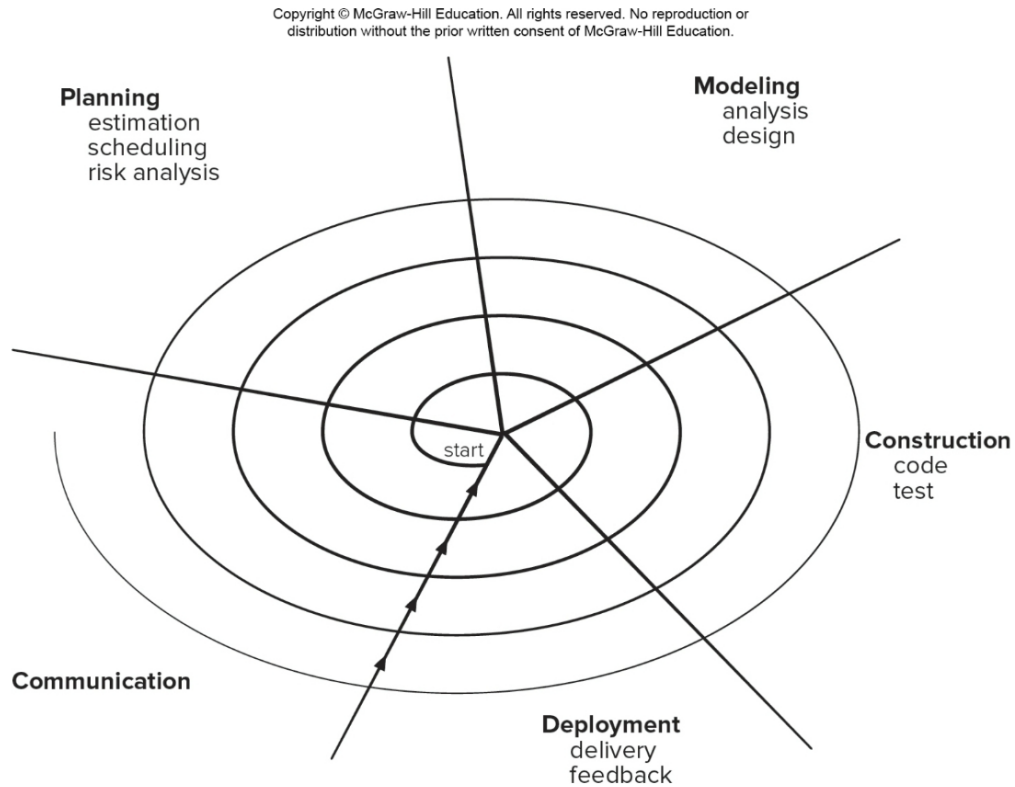
Together this team, otherwise known as TrainX Architects, will work together to revolutionize the future of railroads as we know it today. We are looking to innovate the current solution, as well as looking for new ways to expedite the user's experience while maintaining a connection to our design.

Our personal perspective on this project is to produce a system (engine) that informs the user of the train about current conditions, then to have an interface that allows the conductor to manipulate the train speeds according to that information. While this is rather general, we will work to pose a more detailed solution to our problem as we move through the planning process that this project will entail.

We see ourselves working to continue planning, then soon to begin modeling, construction, and deployment. After that we will reevaluate and communicate with our stakeholders and our group to ensure we are meeting their expectations. We have yet to create deliverables where we set due dates for each aspect of our project but we look forward to making more progress in reaching our goals.

Timeline:



Model:**(Spiral Process Model)**

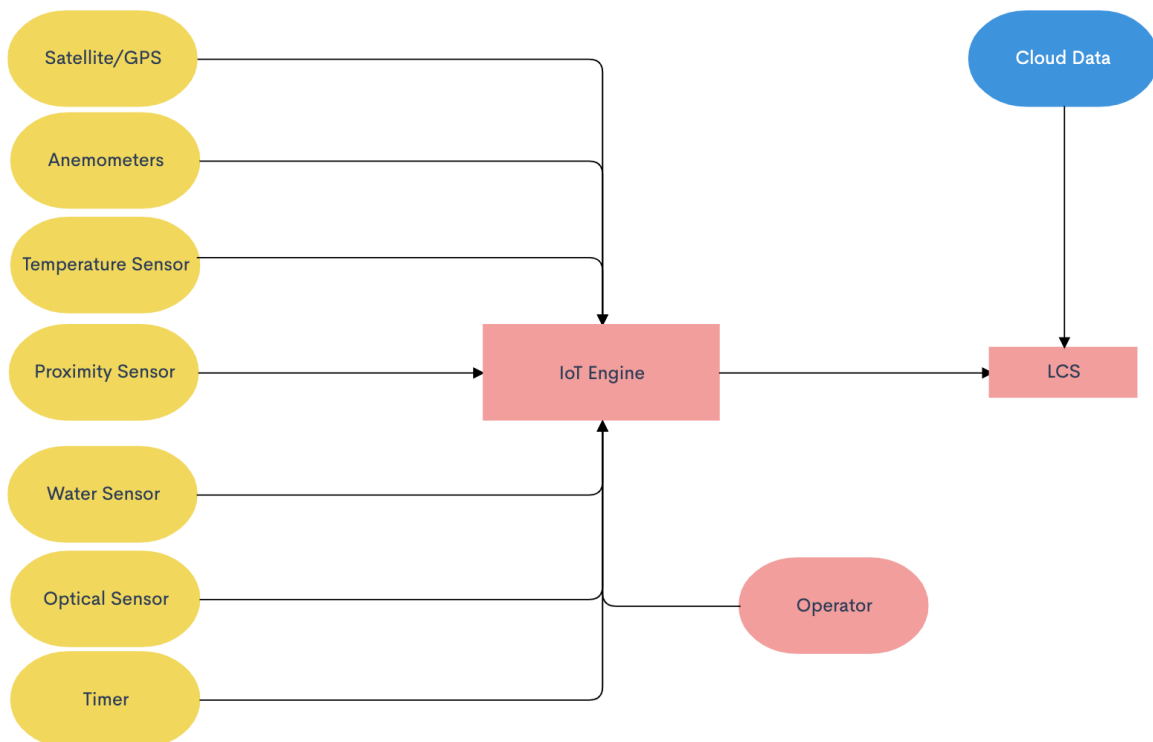
Pros	Cons
<ul style="list-style-type: none"> • Continuous customer involvement • Development risks are managed • Suitable for large, complex projects • Allows to change • Works well for extensible products 	<ul style="list-style-type: none"> • Risk analysis failures can doom the project • Project may be hard to manage • Requires an expert development team

Section 2:

Problem Statement:

The train operation depends on live data received from the Central Operation Center Servers via WiFi/Cellular network. We need to be able to operate the train when we lose WiFi/Cellular network. We need to develop a system to allow us to continue the trip safely based on local data that we can collect. We assume GPS data is available.

System Overview



- Using installed sensors and computers outlined in the flowchart above, the IoT engine will take in different inputs from the data collected by the external sensors.
- The operator also has input into the IoT engine as they have power to adjust any part of the IoT engine including the speed of the train. The operator may have access to information that is not collected by the external sensors via radio, visual, etc.
- The IoT engine utilizes the data collected from these different sensors and/or if the operator influences anything and dynamically adjusts the throttle to brake or speed up appropriately.

- The LCS also has access to cloud data previously collected before the train lost connection, any previous data of value including arrival and departure times can be used until connection is regained.

Data Required

- Weather condition (such as snow, ice, rain, and wind)
- Hazards on the track both front and back
- Gate opening and closing times
- Arrival and departure times

Technology Required

- Temperature
 - Temperature sensor will measure the temperature outside and work with the optical sensor to determine an ideal speed for the train while not connected.
- Optical sensor
 - Ensure that the train does not start “slipping” on tracks and the locomotive is kept in control the entire time.
- Proximity sensor
 - In order to detect hazards on the railway both in front and in back of the train, there will be proximity sensors placed on the train that will work with IoT in order to determine if a change in the locomotive speed should occur.
- Water Sensor
 - Water sensors should be placed in strategic locations on the train in order to properly monitor weather conditions outside. Depending on the severity of the weather conditions, the water sensor should interface with the IoT to properly determine the ideal travel speed for the train.
- Timer
 - In order to properly arrive at gates on time and be prepared to stop in case of a scheduled downed gate or terminal, all arrival and departure times should be downloaded before losing connection. The IoT will work with this pre-downloaded data in order to determine the proper speed to arrive on time and not too early or too late.
- Anemometers
 - In case of harsh weather conditions, the locomotive should be prepared to travel with caution. An anemometer to measure wind speeds should be programmed to collaborate with the IoT architecture to determine if the train should travel at a cautious speed.
- Satellite Technology/GPS
 - Rely on GPS technology to determine the current position and direction of the train. Data can be drawn from the GPS to ensure the train stays on its intended route and not conflict with other trains in the area.

- GPS also provides data on the speed of the train to interface with the other sensors and IoT in order to accommodate for outside conditions.

Section 3:

Requirements

3.1 Non-Functional Requirements

3.1.1: IoT HTR shall only be accessed via Operator ID and Password.

- We have to make sure that our information can only be assessed by a secure ID and password to ensure that we are to have complete control over our project.

3.1.2: IoT HTR Admin shall have secured (Admin ID/Pwd) to all sensors and equipment.

- Additionally, we have to ensure that our ID and password also are incorporated into the sensors and equipment to ensure the same security for our devices. This will allow our whole system to work without a fail rate.

3.1.3: IoT HTR Network shall be secured by LoRaWan protocol

- We will use a lower power wide area connection which will be wireless which provides single hop links between our server and the engine allowing for secure connections.

3.1.4: IoT HTR shall process an event within 0.5 seconds.

- Between 0.1 second and 1 second is a comfortable range in system response time where the user feels as though the system is responding without interrupting flow of thought, so in order to maintain the usability of the product, the sensors as well as the system itself must provide 0.5 seconds or less of processing time.

3.1.5: IoT HTR shall process 1000 events per second without degrading service.

- The scalability of the system must still process an event within 0.5 even when processing 1000 events/second, including responding to the user and working simultaneously with other processes in the system over a WiFi/LTE connection.

3.1.6: IoT HTR shall operate with no failure 99.99% of the time.

- In order to ensure the safety of the operators and reliability of the IoT engine, extensive testing will be required until there are little to no failures whatsoever.
- The reliability of the IoT engine is mission critical to the safety and security of TrainX, its customers and the environment traveled.

3.2 Functional Requirements

3.2.1 Detect standing objects on the path of the train with distance, speed and suggestion to Conductor on braking or decreasing/increasing speed

- Proximity sensors shall be installed on both the front and rear of the train
- Proximity sensors report constant updates on standing hazards on the track to the IoT Engine
- IoT Engine makes needed calculations to recommend amount of braking required to possibly make an emergency stop

3.2.2 Detect moving objects ahead or behind and their speed, distance, with suggestion of brake, increase/decrease speed

- Proximity sensors report speed of moving objects ahead or behind train to IoT Engine
- IoT Engine makes needed calculations to determine whether and how much to increase or decrease speed

3.2.3 Detect gate crossing open/closed, distance, speed and suggestion of braking, speed increase /decrease

- IoT Engine shall utilize local timers, data, and GPS speed to determine appropriate speed to arrive at gate crossing at scheduled time

3.2.4 Detect wheel slippage, using GPS and wheel RPM, suggest braking or increase/decrease speed

- Optical sensor and satellite GPS will send data to IoT Engine regarding to train speed and RPM to calculate whether the train is slipping on the tracks
- IoT Engine will suggest braking or increasing speed based on data from sensors

3.2.5 Detect severe weather conditions, suggest braking or increase/decrease speed

- Water sensor, Anemometer and Temperature sensor will send data to IoT Engine every 5 seconds relating to rainfall, wind speeds, and temperature
- IoT Engine will make appreciate calculations to determine if severe weather conditions warrants braking, increasing/decreasing speed

3.2.4 Display Requirements

- The IoT HTR Engine will be able to display the data in its own way they receive to the train operator in an interface.

- The operator and admin will have a terminal based display and interface that will allow them to use the engine.

3.2.5 Operator, and Admin privileges

- Operator: Has access to the polished result of sensor data.
 - Operator will access display via User ID and password
- Admin: Has access to the data received by all the sensors, can enable or disable sensors, create or delete other accounts, and change account privileges
 - Admin will access IoT HTR via Admin ID and password
 - Admin has access to log data, software update and configuration, etc.

3.3 Hardware & Operating System

3.3.1: IoT Hardware shall be able to support 10,000 sensors

- The compatibility of the hardware must ensure that all the sensors of the system can co-exist and function/interact properly within the same environment.

3.3.2: The Engine will be able to support up to 5TB of data a day.

- The engine will be able to transport up to 5TB of data a day between all of the sensors and throughout the Network.

Section 4:

Requirements Analysis Modeling

4.1 Use Cases

Use Case 1: User Logs Into IoT Engine

Primary Actor: Operator

Goal in Context: To log into and Initialize the IoT Engine so that if the locomotive were to lose connection to WiFi/Cellular, the IoT Engine will be activated.

Preconditions: Locomotive is running.

Trigger: The operator initializes the IoT Engine.

1. Operator requests login of IoT Engine.
2. The IoT Engine activates operator display with sensor statuses.
3. The IoT Engine connects with all sensors, trains and displays statuses.
4. The IoT Engine tests sensors to ensure all are in working order.

Exceptions:

1. Failed to initialize: Admin is required to troubleshoot IoT Engine

Use Case 2: IoT Engine is Activated

Primary Actor: Operator

Goal in Context: To activate the IoT Engine when the locomotive loses connection to the WiFi/Cellular.

Preconditions: Locomotive is moving and IoT Engine is Initialized.

Trigger: The locomotive loses connection to the WiFi/Cellular.

1. The Locomotive loses connection to the WiFi/Cellular.
2. The IoT Engine is activated from initialization.
3. The sensors that are connected to the IoT Engine are activated
4. The IoT Engine immediately begins to perform calculations to determine further action
5. Data from sensors is continuously sent to IoT Engine to monitor conditions
6. Log is initialized and starts logging to display.

Use Case 3: IoT Engine Suggests Immediate Braking Due to Standing Hazard

Primary Actor: Proximity Sensors

Goal in Context: To apply brakes immediately when a standing hazard is detected on the tracks

Preconditions: Locomotive is moving and IoT Engine is activated.

Trigger: Proximity sensors detect standing hazard on track.

1. The proximity sensor detects hazards on track.
2. Standing Hazard distance data is reported to IoT Engine.
3. The IoT Engine makes needed calculations to determine how hard to safely brake.
4. The IoT Engine displays braking recommendations to operators.
5. Operator initializes braking according to IoT Engine recommendation.
6. Immediate braking recommendation is logged to log and display.

Exceptions:

1. Standing hazard is removed from tracks and locomotive can continue as normal.

Use Case 4: IoT Engine Suggests Decreasing/Increasing Speed Due to Moving Hazard

Primary Actor: Proximity Sensors

Goal in Context: To reduce speed of the train due to moving hazards on track.

Preconditions: Locomotive is moving and IoT Engine is activated.

Trigger: Proximity sensors detect moving hazard on track.

1. The proximity sensor detects moving hazards on track.
2. Hazard distance and speed data is reported to IoT Engine.
3. Speed of train is reported to IoT Engine from GPS
4. The IoT Engine makes needed calculations to determine what speed should be set to in order to ensure a safe distance from moving hazards.
5. The IoT Engine displays speed recommendations to operators.
6. Operator adjusts speed according to IoT Engine recommendation.
7. Decrease/Increase speed recommendation is logged to log and display.

Exceptions:

1. Moving hazard speed becomes negligible to locomotive's and locomotive can continue as normal.

Use Case 5: IoT Engine Suggests Change of Speed to Accommodate for Gate Closing/Opening Times

Primary Actor: GPS Speed, Timer

Goal in Context: To adjust speed of the train so that the train safely proceeds in case of wheel slippage.

Preconditions: Locomotive is moving and IoT Engine is activated.

Trigger: IoT Engine detects timing error with regard to gate closing/opening times.

1. The downloaded gate closing/opening times, timer data, GPS speed should all be reported to IoT Engine.
2. IoT Engine continuously monitors arrival time based on GPS speed, timer data, distance to destination.
3. IoT Engine calculates a discrepancy in arrival time based on sensor data.
4. IoT Engine recommends a speed on display.
5. Operator adjusts speed of train to match IoT Engine recommendation.
6. Change of speed is logged to log and display.

Exceptions:

1. Schedules are adjusted and locomotive can continue as normal.

Use Case 6: IoT Engine Suggests Change of Speed to Accommodate for Wheel Slippage.

Primary Actor: Optical Sensor

Goal in Context: To adjust speed of the train so that it arrives at the gate at an appropriate time.

Preconditions: Locomotive is moving and IoT Engine is activated.

Trigger: IoT Engine detects slippage.

1. GPS speed data, optical sensor data will continuously report data to IoT Engine.
2. IoT Engine continuously makes calculations to monitor slippage.
3. IoT Engine detects slippage severity and makes speed recommendations based on sensor data.
4. Speed recommendation is reported on display.
5. Operator adjusts speed of train to match IoT Engine recommendation.
6. Change of speed is logged to log and display.

Use Case 7: IoT Engine Suggests Change of Speed to Accommodate for Weather Severity

Primary Actor: Weather Sensor

Goal in Context: To adjust speed of the train so that it safely proceeds through treacherous weather conditions.

Preconditions: Locomotive is moving and IoT Engine is activated.

Trigger: IoT Engine detects severe weather.

1. Water sensor, Anemometer and Temperature sensor will send data to the IoT Engine continuously.
2. IoT Engine continuously determines “severe weather”.
3. IoT Engine detects “severe weather” and makes suggested calculations for recommended speed of train.
4. Speed recommendations are displayed to the operator.
5. Operator adjusts trains speeds to match recommendations from IoT Engine.
6. Change of speed is logged to log and display.

Use Case 8: IoT is Deactivated

Primary Actor: Operator

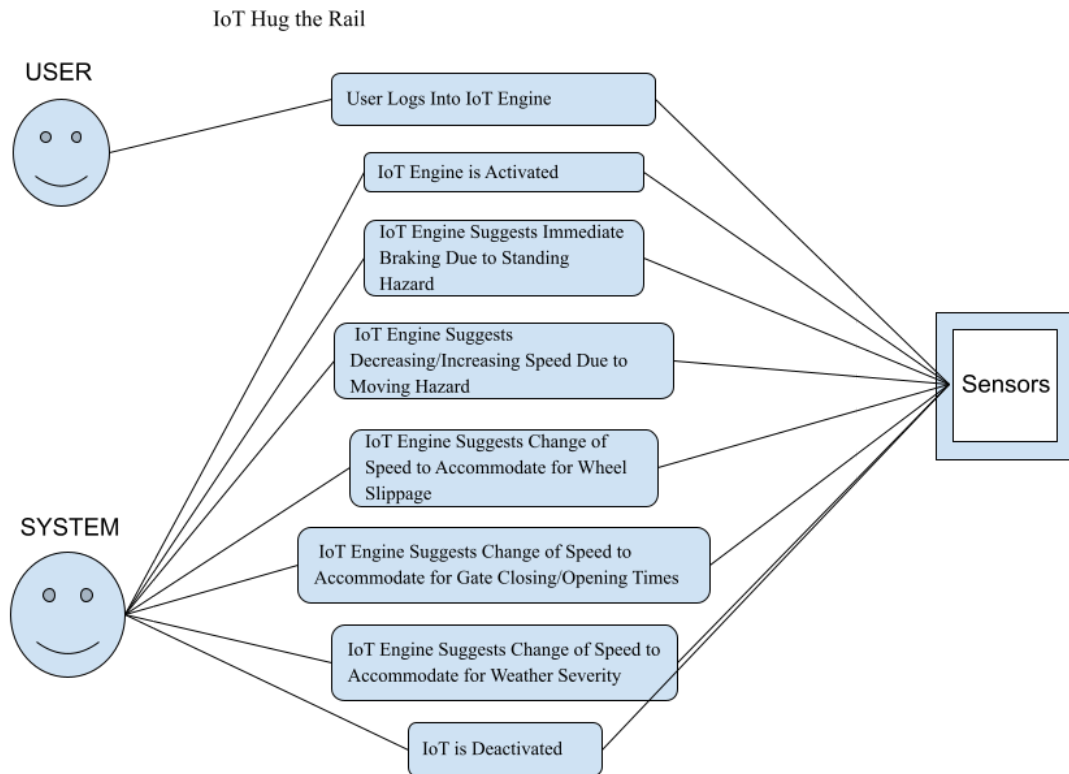
Goal in Context: To deactivate IoT Engine once cloud connection is returned.

Preconditions: Locomotive is moving and IoT Engine is activated.

Trigger: Locomotive reconnections to WiFi/Cellular.

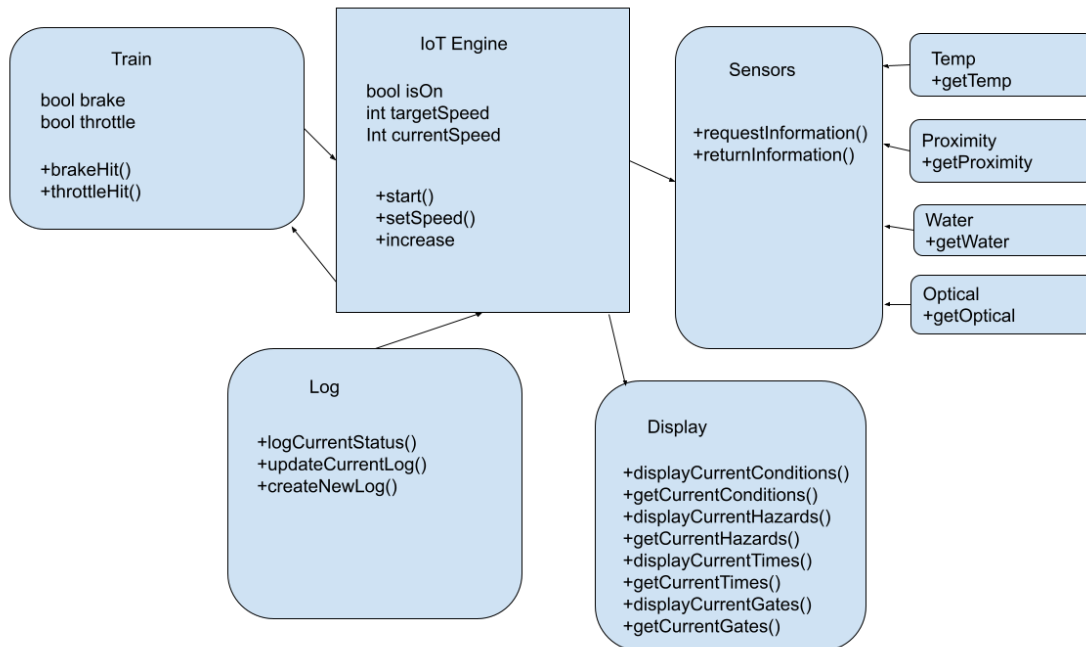
1. Locomotive reconnects to WiFi/Cellular.
2. Deactivation of the IoT Engine is displayed to the operator.
3. Deactivation is logged to log and display.
4. Log is saved into a database for records.

4.2 Use Case Diagram



For the Case Diagram we decided to take our eight main cases and then establish whether they were in relation to the User or the System which in our case is the Internet of Things Engine. For seven out of our eight cases we noticed the sensors would be the primary accessed by the Engine and not the user which makes sense because we want to make the life of the user easier so really the only thing they should handle is logging into the Engine. After that our Engine will take over all the tasks. Additionally for security reasons we feel we want to handle all the data.

4.3 Class-Based Modeling



For purposes of organization, we have divided up the key components of our project into five main classes. The first will focus on the brake and throttle of the Train which will access the IoT Engine. The IoT Engine will contain the speed including the current speed and target speed in order to ensure that the train does not surpass the speed it can handle. Additionally we will be using our Sensor class to keep track of our data where two main methods will take place - the `requestInformation` method and the `returnInformation` method. These two methods will seamlessly access the subclasses such as Temp, Proximity, Water, and Optical. These are the key components we felt would help assess the capabilities of our Train. Additionally we then needed a class to display all this information. Lastly, we wanted to keep a log of all of our data, specifically the status of the train at all times.

4.4 CRC Modeling/Cards

IoT Engine	
<ul style="list-style-type: none"> • Regulates Sensor data • Regulates Log data • Regulates Train controls • Regulates data to display 	<ul style="list-style-type: none"> • Sensors • Log • Train • Display

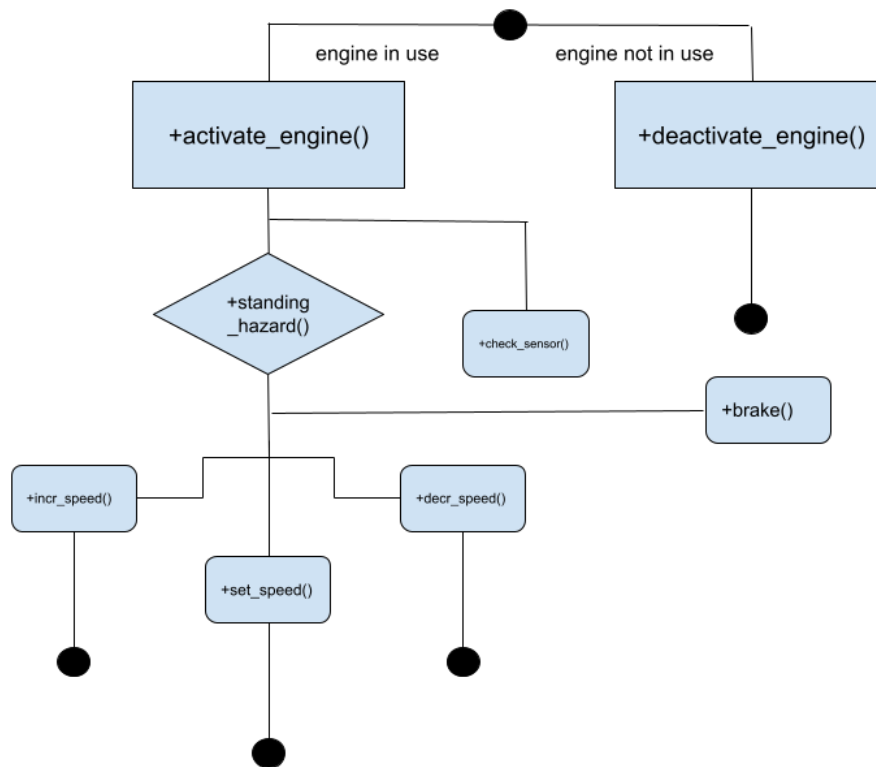
Sensors	
<ul style="list-style-type: none"> • Returns Information about a specific sensor • Requests Information from a specific sensor 	<ul style="list-style-type: none"> • IoT Engine

Log	
<ul style="list-style-type: none"> • Updates the Current log with information • Creates a new Log with information • Logs the current status 	<ul style="list-style-type: none"> • IoT Engine

Train	
<ul style="list-style-type: none"> • Controls the train's functionality • Allows the brakes to be activated • Allows the throttle to be activated 	<ul style="list-style-type: none"> • IoT Engine

Display	
<ul style="list-style-type: none"> • Displays current Conditions • Gets current Conditions • Displays Hazards • Get necessary Hazard data • Display Current Gates • Get the Current dates 	<ul style="list-style-type: none"> • IoT Engine

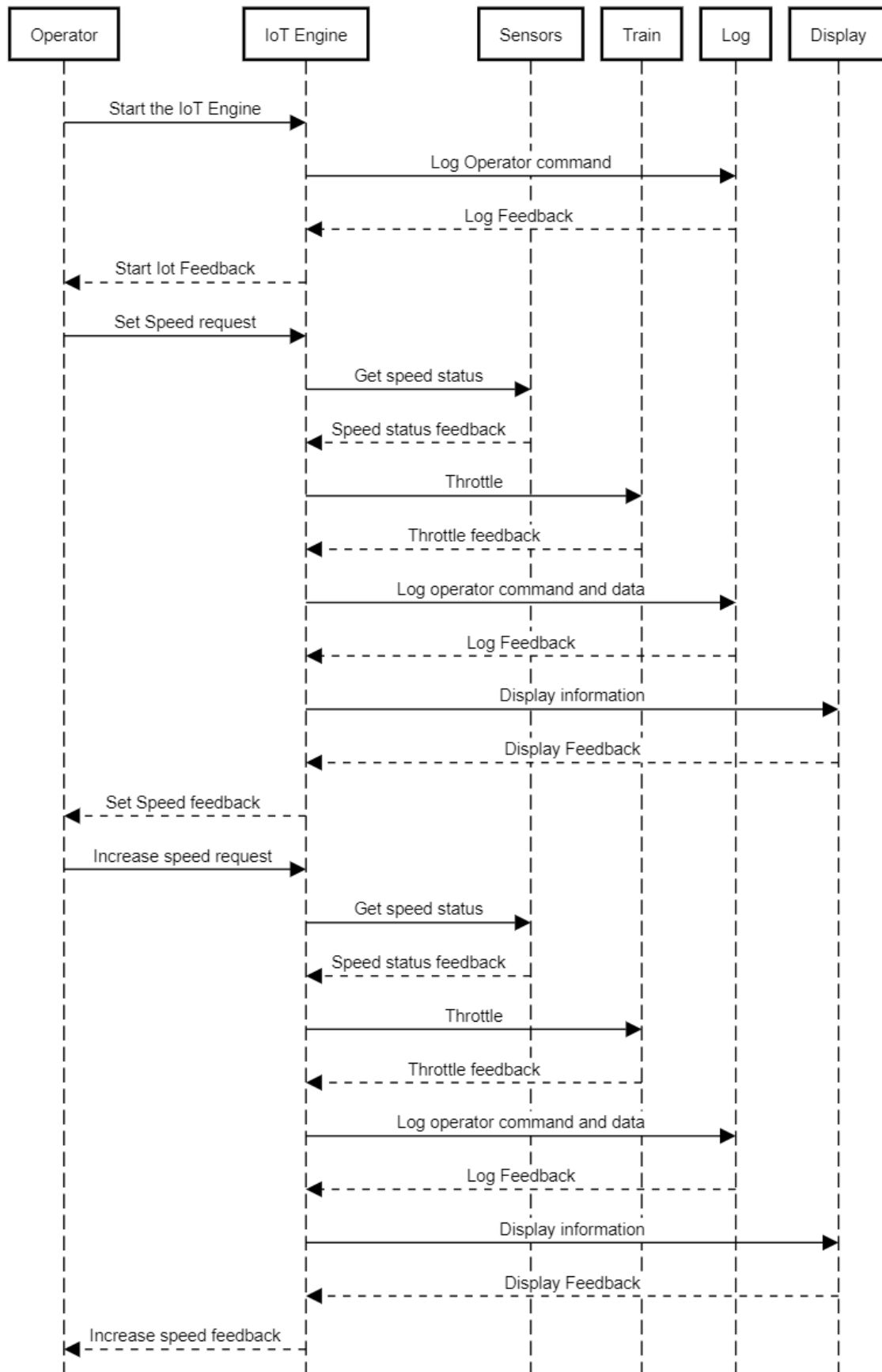
4.5 Activity Diagrams



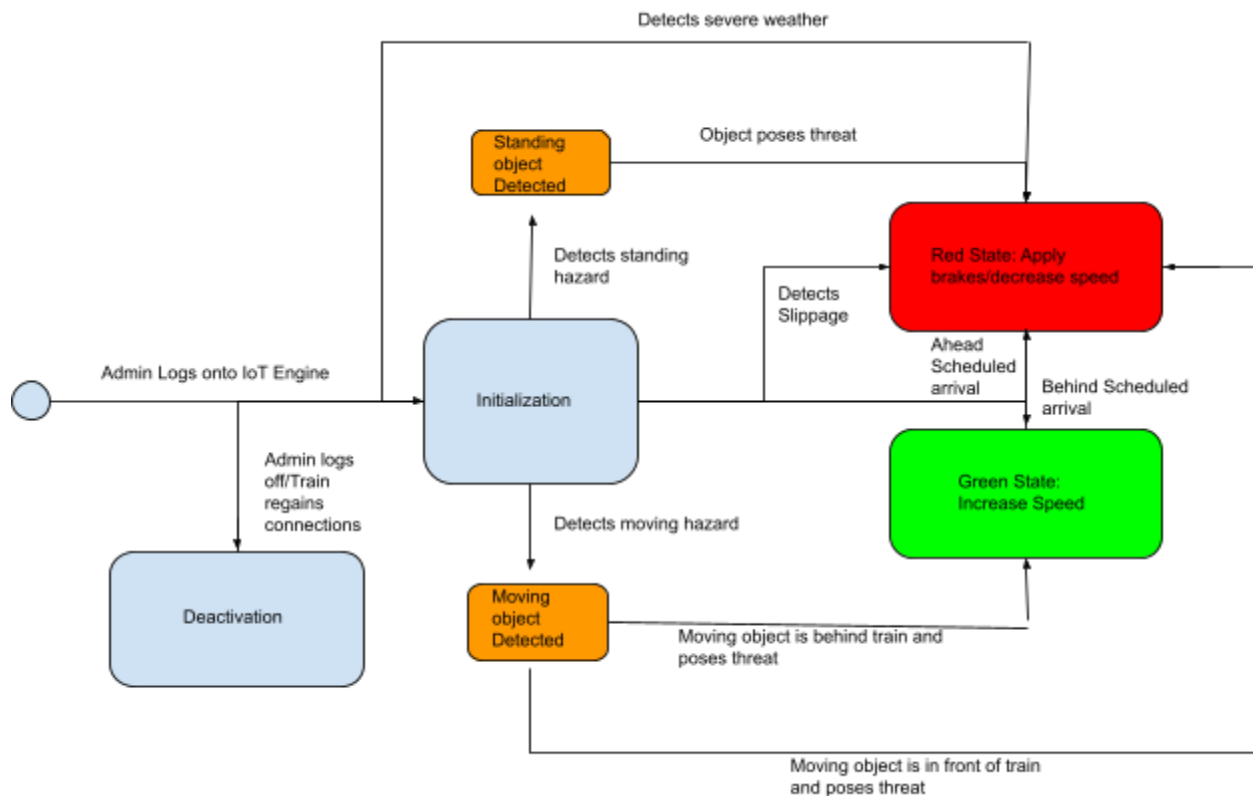
This activity diagram depicts the general skeleton of the behavior of the IoT Engine when it is in use versus when it is not in use. Following the description for the usage of the engine, the engine can possibly be activated when it loses connection to WiFi. In that case, it would have to abide by thought-out steps to ensure that the system can still run smoothly even if internet connection is not optimal. In this case, sensors will need to be in place and checked even without connection, and the diagram illustrates that that is the first priority once the engine is activated. The feedback from this sensor check will then lead to requests to check speed and thus assist the engine in making suggestions to increment or decrement speed accordingly, and setting speed based on the conditions derived from the sensor check. This check to assess whether change in speed is necessary is crucial to most of the potential cases in the operation of the engine/locomotive. The last case illustrated is the engine being deactivated, and this will be displayed once it reconnects to Wifi or the internet.

4.6 Sequence Diagrams

IOT HTR Sequence Diagram



4.7 State Diagram



For our state diagram we have modeled our process by configuring the IoT Engine around one main “initialization” stage. Once the Admin logs onto the IoT Engine, the IoT Engine will remain in the initialization stage. From here all functional capabilities of the IoT Engine will be utilized. If the IoT Engine receives data from the sensors that there is a standing hazard it moves up to the standing object state and then if the object poses a threat then the IoT Engine will move into the “Apply brakes/decrease speed” state. Similarly, if the IoT Engine detects a moving hazard it moves to the “moving object” state where it will determine whether to move to the “Increase speed” or “Apply brakes/decrease speed” state. From the Initialization state the IoT Engine will also determine whether to move to the “Apply brakes/decrease speed” state or “Increase speed” state if it detects things like “Severe weather”, “Slippage”, “Behind schedule”, and “Ahead of schedule”. Once the Admin logs off of the IoT Engine, the state will be moved to “Deactivation”.