

IoT HTR Test File

Team 9: Topham Hatts

Members: Eleanor Katsman, Alfredo Mendez, Simrun Heir, Murat Ulu

Pledge: "I pledge my honor that I have abided by the Stevens Honor System"

Note:

Our program includes a "Sliders" page, or Controller panel, which acts as a controller to set all of the sensor input values. All tests shown below contain step by step directions on how we tested and how our tests could be recreated. Pictures are included to highlight program behavior during testing.

This "Test File" Acts more as a User Guide to replicate our testing.

Test 1:

→ Description:

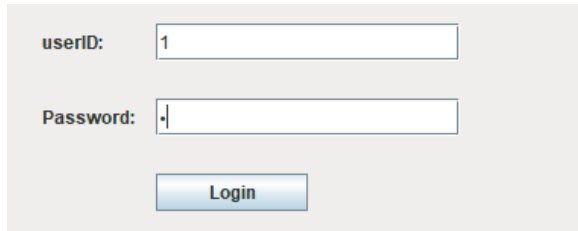
◆ *User attempts to log in to IoT HTR System by entering credentials*

→ Expected Result:

◆ *IoT Display displays the appropriate page layout (Operator View or Administrator View) of IoT HTR System*

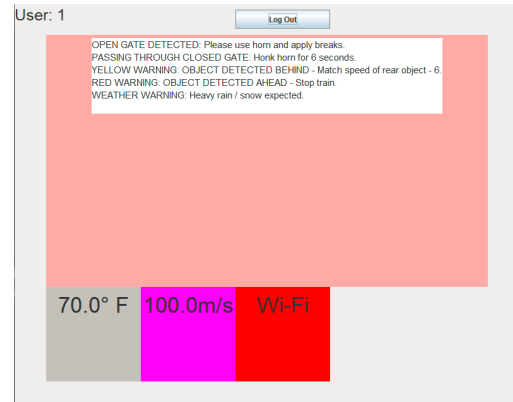
How to test:

- When starting up the program or after logging out of a user account, you will be met with a "Log-In" screen:
 - For "operator" credentials, enter:
 - Username: 1
 - Password: 1
 - **Result:** Program opens up IoT Operator View of our software



A login form with a light gray background. It contains two input fields: 'userID:' with the value '1' and 'Password:' with a single dot. Below the fields is a blue 'Login' button.

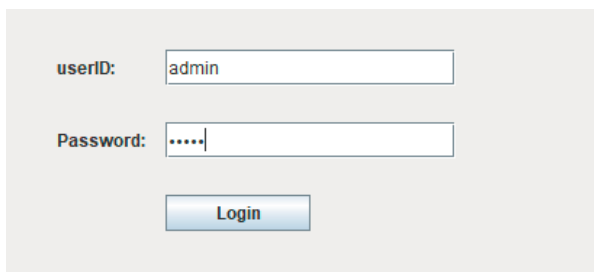
*Operator Credentials entered
(User presses “Login” button after entering
credentials)*



The Operator View interface. At the top left, it says 'User: 1' next to a 'Log Out' button. A large red rectangular area contains a list of warnings: 'OPEN GATE DETECTED: Please use horn and apply breaks.', 'PASSING THROUGH CLOSED GATE: Honk horn for 6 seconds.', 'YELLOW WARNING: OBJECT DETECTED BEHIND - Match speed of rear object - 6.', 'RED WARNING: OBJECT DETECTED AHEAD - Stop train.', and 'WEATHER WARNING: Heavy rain / snow expected.'. At the bottom, there are three colored boxes: a gray box with '70.0° F', a magenta box with '100.0m/s', and a red box with 'Wi-Fi'.

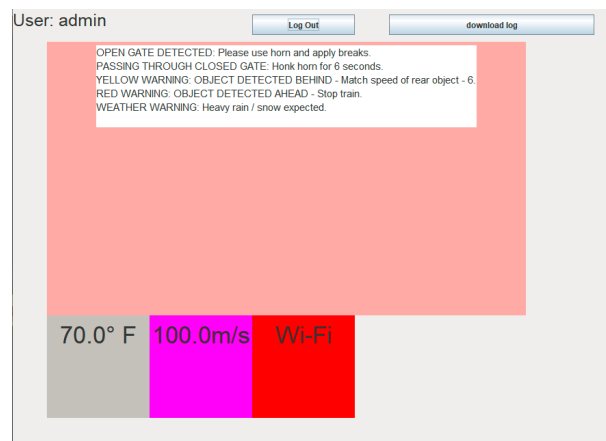
Operator View of software is displayed

- Alternatively:
 - For “administrator” credentials, enter:
 - Username: admin
 - Password: admin
 - **Result:** Program opens up to IoT Administrator View of our software



A login form with a light gray background. It contains two input fields: 'userID:' with the value 'admin' and 'Password:' with six dots. Below the fields is a blue 'Login' button.

*Administrator Credentials entered
(User presses “Login” button after entering
credentials)*



The Administrator View interface. At the top left, it says 'User: admin' next to a 'Log Out' button and a 'download log' button. A large red rectangular area contains the same list of warnings as the Operator View. At the bottom, there are three colored boxes: a gray box with '70.0° F', a magenta box with '100.0m/s', and a red box with 'Wi-Fi'.

*Administrator View of software is displayed
(distinguishable by the “download log”
button in the top right corner)*

Test 2:

→ Description:

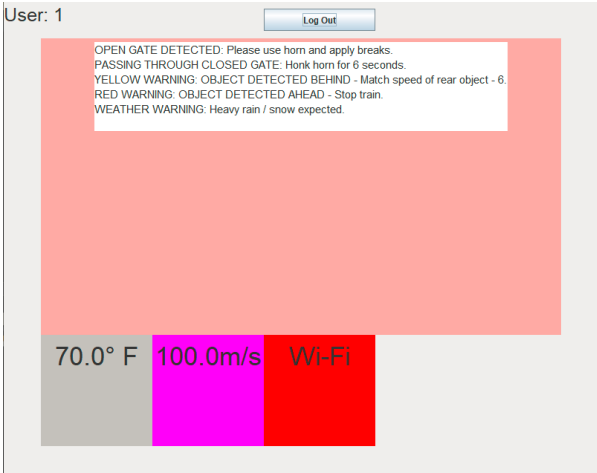
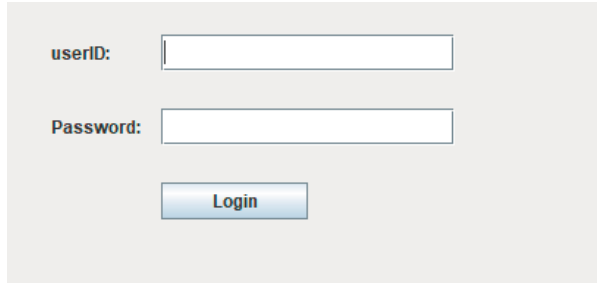
◆ *User attempts to log out of IoT HTR System*

→ Expected Result:

◆ *IoT Display displays the log-in prompt page of IoT HTR System*

How to test:

- While in either the “Operator View” or the “Administrator View” (logged in as user):
 - Press “Log-Out” button
 - **Result:** Program returns user to “Log-In” prompt page

 <p><i>User is in either Operator View or Administrator View, and user presses “Log Out” button</i></p>	 <p><i>Program returns user to “Log-In” prompt page, both input boxes blank and awaiting input</i></p>
---	--

Test 3:

→ Description:

◆ *IoT System processes array of data to get an average sensor reading*

→ Expected Result;

◆ *IoT System produces single float that is the average of an inputted array of floats*

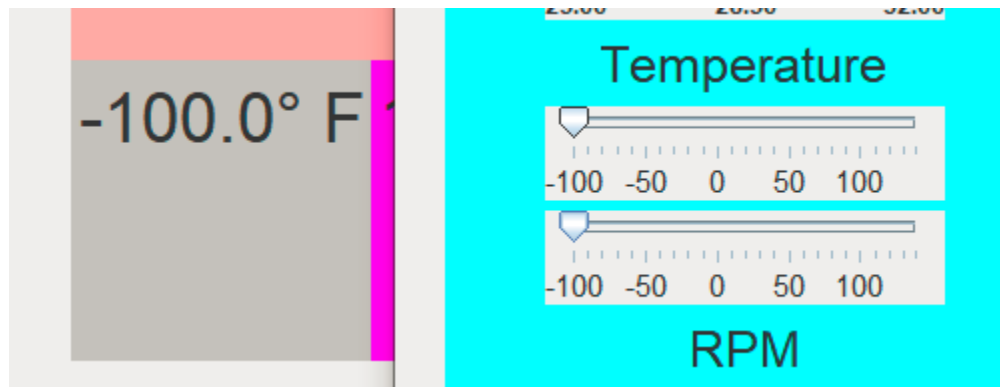
How to test:

- Our software calculates the average of multiple sensor inputs, such as air pressure, temperature, and wheel rotations per minute (rpms)
- **Result:** Average from array of values is calculated to two decimal places

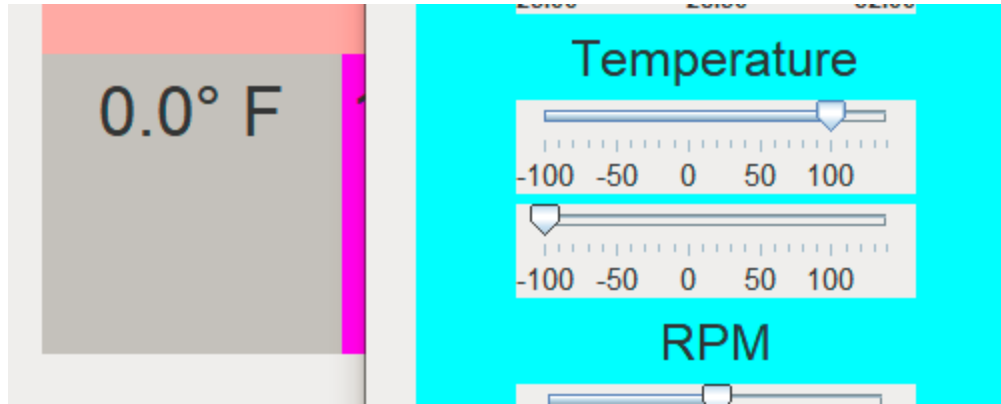
```
//input: double array (calculates average of everything in array)
public double calculateAverage(double [] array) { //input double array
    int sum = 0;
    for(int i = 0; i < array.length; i++) {
        sum += array[i];
    }
    DecimalFormat numberFormat = new DecimalFormat("#.00");
    return Double.parseDouble(numberFormat.format(sum / array.length));
}
```

calculateAverage() function shown above

- We can test this calculateAverage() function by modifying the two “Temperature” sliders in the controller panel, and observing if the Temperature displayed in the User’s view is an average of the two values
 - The temperature displayed is the average value from the two temperature sliders within the Controller panel



Temperature displayed in bottom left corner of User View is constantly updated to match average value of Temperature sliders



With the given temperature values of “-100” and “100”, IoT System calculates the correct average to be 0 degrees Fahrenheit

Test 4:

→ Description:

◆ *IoT System calculates Slip Ratio*

→ Expected Result:

◆ *IoT System produces a percentage value for slip ratio*

How to test:

- The IoT System calculates the slip ratio using the SAE J670 slip ratio equation:

$$(Slip\ Ratio\ \% = (\frac{Angular\ Velocity * Wheel\ Radius}{Speed\ of\ Train} - 1) * 100\%)$$

```
//input: double avgRpm, double speed (speed is gps speed, radius of wheel is assumed to be .46 meters)
public int calculateSlipRatio(double avgRPM, double gpsSpeed) {
    double angularV = avgRPM*6;
    int num = (int)((angularV * .46) / gpsSpeed) -1;
    return num;
}
```

calculateSlipRatio() function shown above

- To test this,
 - Set first RPM slider to 0, and set second RPM slider to 1000.
 - Set the GPS Speed to 200
 - Slip ratio should be “(500*6*.46 / 200) - 1” (equal to 5.0)
 - VALUE IS ROUNDED DOWN to the nearest integer, ensures maximum safety in variable conditions
 - **Result:** IoT System correctly calculates and displays slip ratio



IoT System correctly calculates the slip ratio to be "5.0"

Test 5:

→ Description:

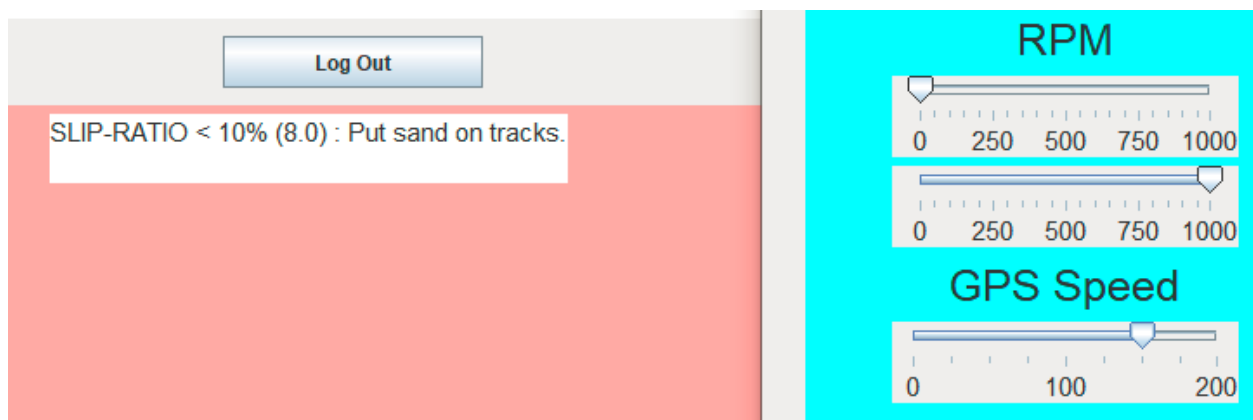
- ◆ *IoT System processes data from TSNR and outputs appropriate warnings for Slip Ratio*

→ Expected Result:

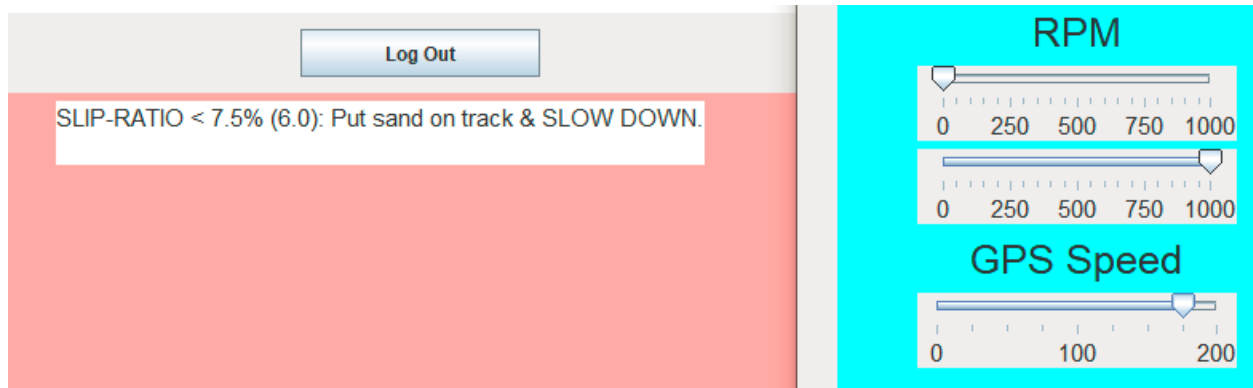
- ◆ *IoT System produces a string that is presented on IoT Display*

How to test:

- The IoT System will generate a string based on the slip ratio and the recommended action for that slip ratio.
 - If the slip ratio is <10% , notify the conductor to put sand on tracks.
 - Set first RPM slider to 0, and set second RPM slider to 1000.
 - Set the GPS Speed to 150
 - **Result:**



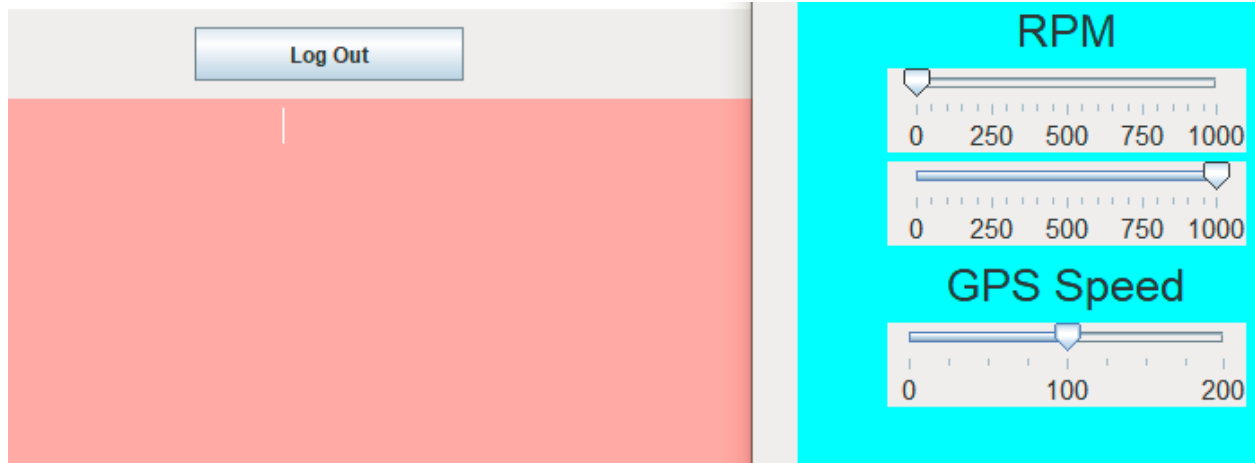
- If the slip ratio is $<7.5\%$, notify the conductor to put sand on tracks and slow down.
 - Set first RPM slider to 0, and set second RPM slider to 1000.
 - Set the GPS Speed to 175
 - **Result:**



- If the slip ratio is $<5\%$, notify the conductor to put sand on tracks and slow down aggressively.
 - Set first RPM slider to 0, and set second RPM slider to 1000.
 - Set the GPS Speed to 200
 - **Result:**



- If the slip ratio is $>10\%$, no warning is shown
 - If RPM average is 500 (as it is in the previous test cases), setting the GPS speed to anything below 125 will result in no Slip-Ratio Warning being displayed
 - **Result:**



Test 6:

→ Description:

- ◆ *IoT System calculates Time until collision when dangerous object on track is detected*

→ Expected Result:

- ◆ *IoT System produces a number that represents Time Until Collision between object on tracks and train*

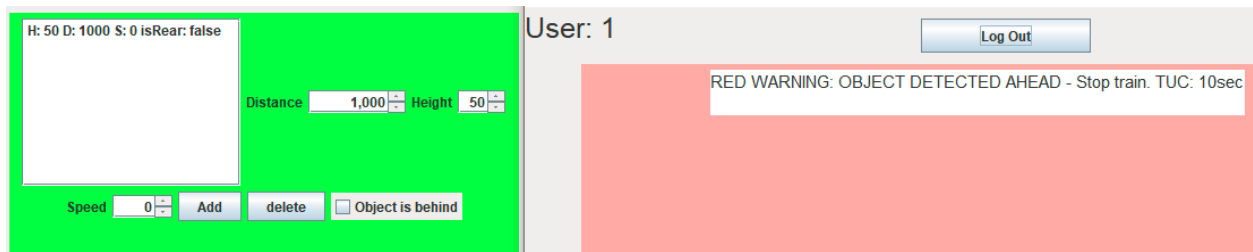
How to test:

```
//input: int trainSpeed, int objSpeed, int objDistance, boolean isRear
public int timeUntilCollision(int trainSpeed, int objSpeed, int objDistance, boolean isRear) {
    if (isRear == false) {
        return (objDistance/(trainSpeed - objSpeed));
    }
    else {
        return (objDistance/(objSpeed - trainSpeed));
    }
}
```

timeUntilCollision() function shown above

- During a “Red Warning”, when an object in front of the train is less than 200 seconds from collision, the Time Until Collision (or TUC) is displayed within the warning message
 - To test this case, create an new Object with the parameters
 - D (distance) : 1000
 - H (height): Anything value above 1.5
 - S (speed): 0
 - IsRear: leave unchecked
 - And set the GPS Speed Slider to 100 m/s (default value)

- **Result:** Warning message should indicate that the Time Until Collision is 10 seconds



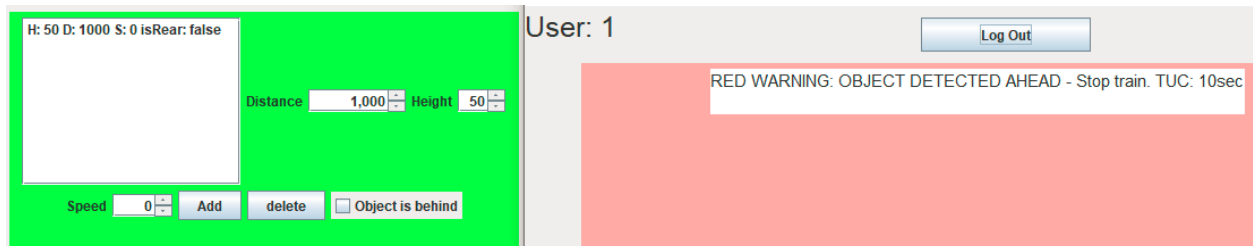
Correct TUC (Time Until Collision) displayed at the end of warning message

Test 7:

- Description:
 - ◆ IoT System processes data from TSNR and outputs appropriate warnings for on track Objects Detected
- Expected Result:
 - ◆ IoT System produces a string that is presented on IoT Display

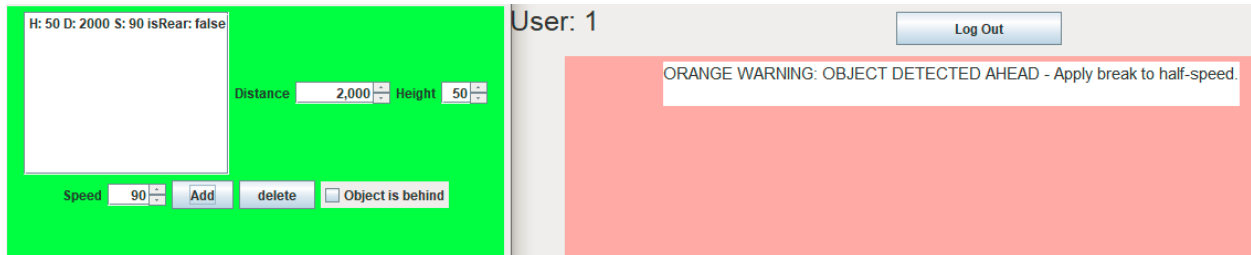
How to test:

- IoT System will display information about the object to the Conductor and will give appropriate warning
 - If the Time Until Collision is less than 200 seconds, the object is bigger than 1.5 feet tall, and the object is in front
 - **Result:** IoT Displays "RED WARNING: OBJECT DETECTED AHEAD - Stop train. TUC: *insert time until collision*"
 - To test this case, create an new Object with the parameters
 - D (distance) : 1000
 - H (height): Anything value above 1.5
 - S (speed): 0
 - IsRear: leave unchecked
 - And set the GPS Speed Slider to 100 m/s (default value)

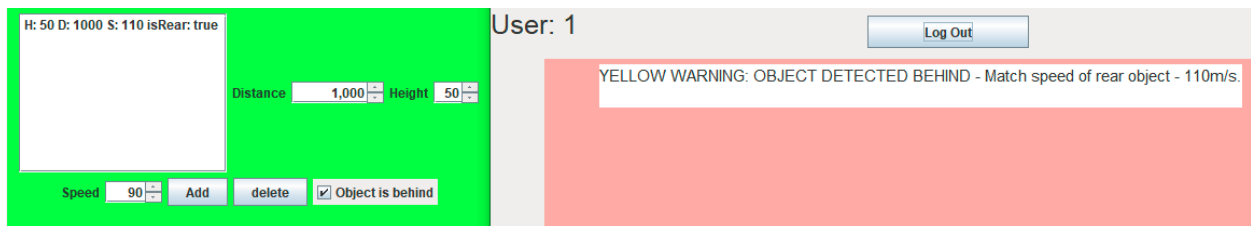


- If the Time Until Collision is less than 260 seconds but greater than or equal to 200 seconds, the object is in front, and the object is bigger than 1.5 feet tall

- **Result:** IoT displays "ORANGE WARNING: OBJECT DETECTED AHEAD - Apply break to half-speed."
- To test this case, create an new Object with the parameters
 - D (distance) : 2000
 - H (height): Anything value above 1.5
 - S (speed): 90
 - IsRear: leave unchecked
- And set the GPS Speed Slider to 100 m/s (default value)



- If the Time Until Collision is less than 260 seconds, the object is behind the train, and the object is bigger than 1.5 feet tall, and the object is going faster approaching the train (going faster than train),
 - **Result:** IoT displays "YELLOW WARNING: OBJECT DETECTED BEHIND - Match speed of rear object - obj.getSpeed()." where obj.getSpeed() is a call to the given objects speed parameter value.
 - To test this case, create an new Object with the parameters
 - D (distance) : 2000
 - H (height): Anything value above 1.5
 - S (speed): 90
 - IsRear: leave unchecked
 - And set the GPS Speed Slider to 100 m/s (default value)



Test 8: ✓

→ Description:

- ◆ *Status of Gate objects and on-track Object objects is gathered*

→ Expected Result:

- ◆ *Gate object and on-track Object object information is accessible for calculation*

How to test:

- All of our detected Objects and Gates are stored in their own respective arrays, shown below

```
protected ArrayList<DetectedObject> objectsDetected;  
protected ArrayList<Gate> gatesDetected;
```

- We can confirm that the status of our Gate objects and on-track Object objects are gathered and stored because the stored Objects and Gates are displayed within the Controller panel (slider panel), and calculations (warning messages) are created using the values they store.
- When testing, any Object or Gate can be deleted or added to their respective arrays

The screenshot shows a Java Swing application with two main panels. The top panel has a green background and contains a text area with two lines of text: "H: 3 D: 4 S: 6 isRear: true" and "H: 500 D: 500 S: 500 isRear: false". To the right of the text area are two sliders labeled "Distance" and "Height", both set to 500. Below the text area are three buttons: "Speed" (set to 500), "Add", and "delete". To the right of these buttons is a checkbox labeled "Object is behind". The bottom panel has a light red background and contains a text area with two lines of text: "is open: true distance: 100" and "is open: false distance: 234". To the right of the text area are two sliders labeled "500" and "Gate is open", both set to 500. Below the text area are two buttons: "Add" and "delete".

Test 9:

→ Description:

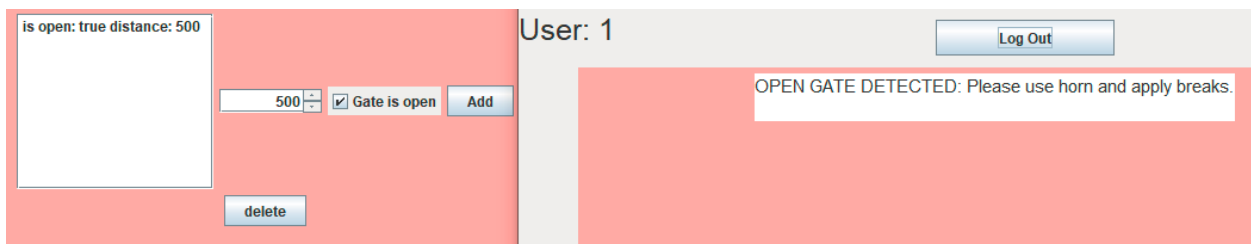
- ◆ *IoT System processes data from TSNR and outputs appropriate warnings for a detected gate*

→ Expected Result:

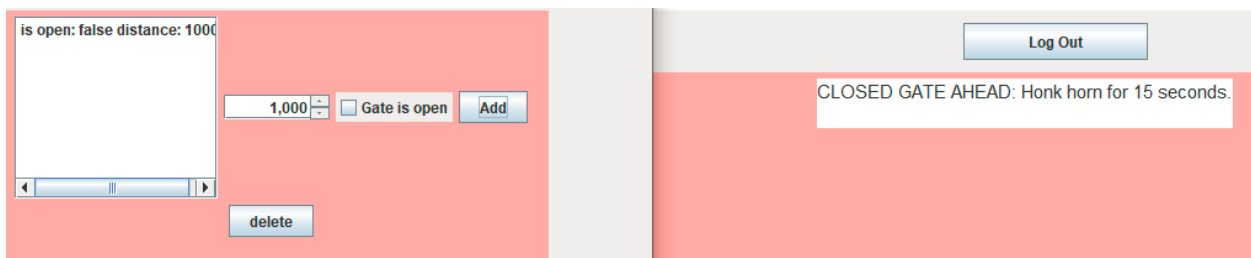
- ◆ *IoT System produces a string that is presented on IoT Display warning section*

How to test:

- If the gate is open,
 - **Result:** Display warning “OPEN GATE DETECTED: Please use horn and apply breaks.”



- If the gate is closed
 - When the train is less than a mile (~1650 meters) but greater than half a mile away (~800 meters) from the gate and gate is closed,
 - **Result:** IoT Display shows warning “HONK HORN FOR 15 SECONDS: Closed gate is 1 mile ahead.”



- When the train is approaching the gate (less than or equal to 400 meters away from gate) and gate is closed,
- **Result:** IoT Display shows warning “HONK HORN FOR 6 SECONDS: Passing through closed gate.”



Test 10:

→ Description:

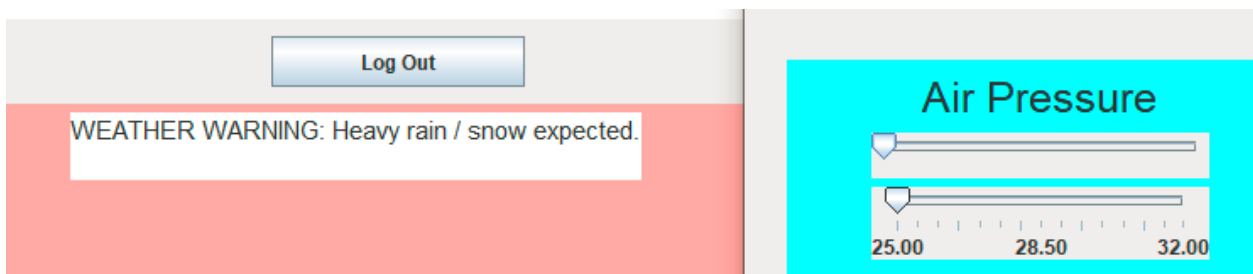
- ◆ *IoT System processes data from TSNR and outputs appropriate warnings for predicted bad weather conditions.*

→ Expected Result:

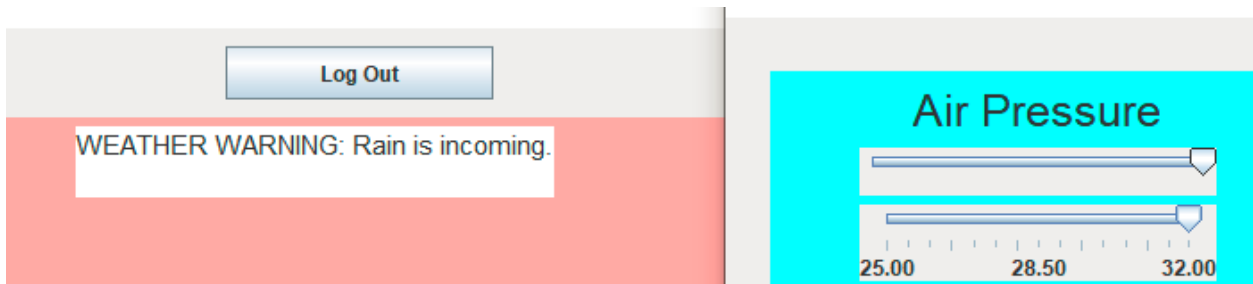
- ◆ *IoT System produces a string that is presented on IoT Display warning section*

How to test:

- If average Hg is 29.8 or below,
- **Result:** IoT Display warns the Conductor that rain storm / snow is incoming.

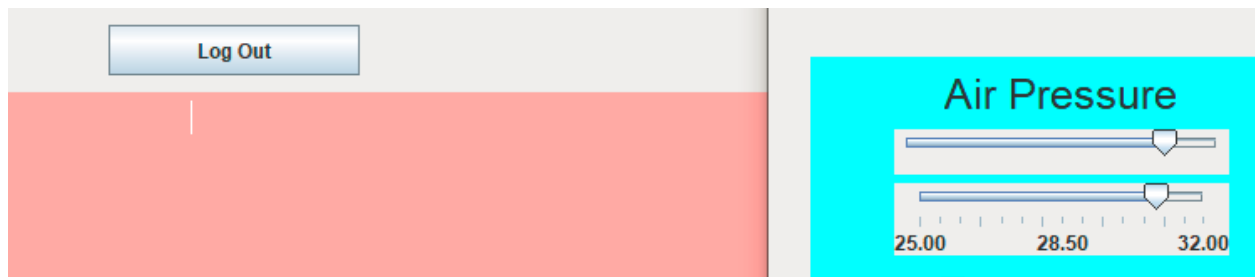


- If average Hg is above or equal to 29.8 and less than 30.2 ,
- **Result:** IoT Display warns that rain is incoming



- If average Hg is above 30.2,

- **Result:** Nothing is displayed as a warning display.

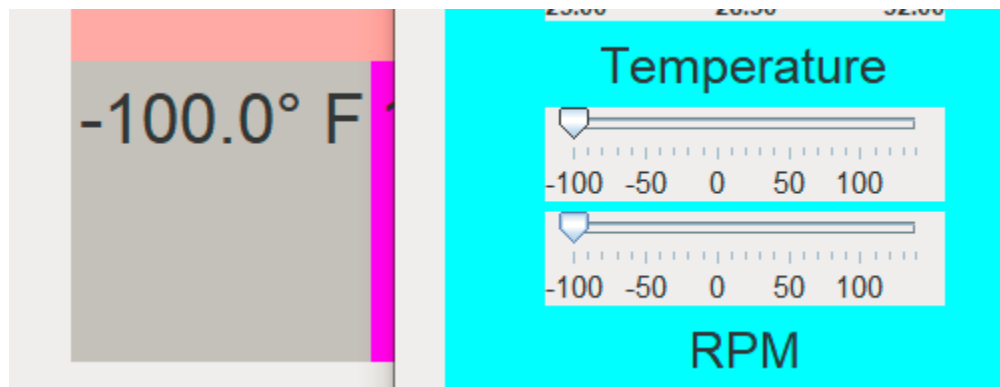


Test 11:

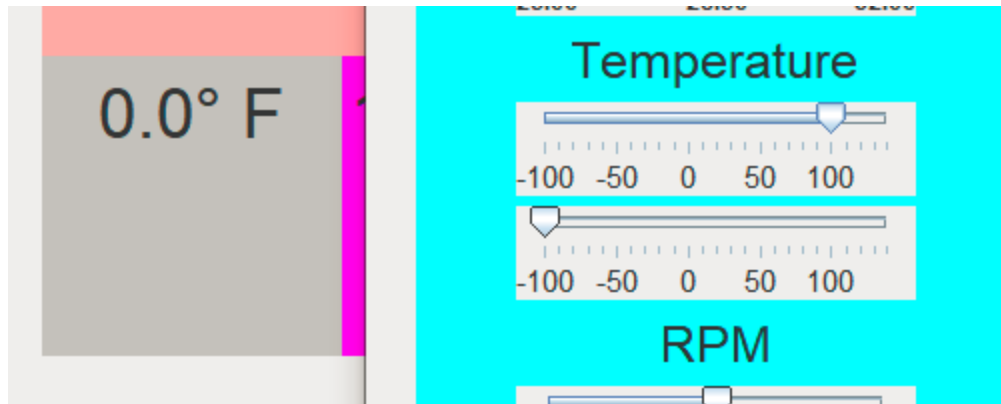
- Description:
 - ◆ *Temperature surrounding the train is being displayed*
- Expected Result:
 - ◆ *The average of the temperature readings is calculated and displayed*

How to test:

- We can test by modifying the two “Temperature” sliders in the controller panel, and observing if the Temperature displayed in the User’s view is an average of the two values
 - Data Flow:
 - TSNR receives readings from sensors
 - IoT System receives readings from TSNR
 - IoT System calculates the average temperature
 - IoT System displays the average temperature



Temperature displayed in bottom left corner of User View is constantly updated to match average value of Temperature sliders



With the given temperature values of “-100” and “100”, IoT System calculates the correct average to be 0 degrees Fahrenheit and displays value on IoT display

Test 12:

→ Description:

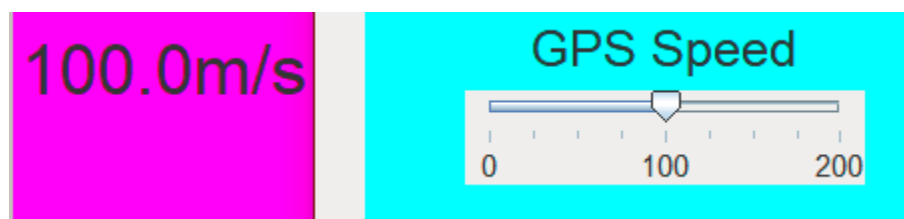
◆ *Speed of the train is being displayed*

→ Expected Result:

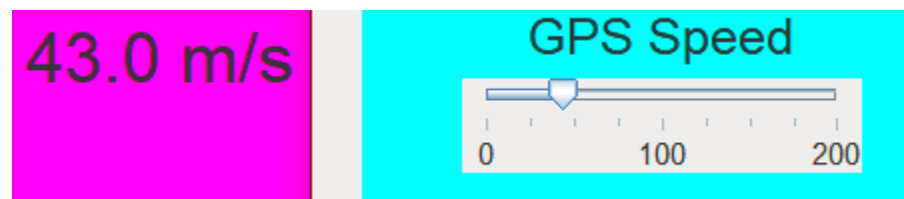
◆ *The current speed of the train is consistently updated*

How to test:

- Change the GPS Speed slider value within the controller panel, and the change should be reflected on Displayed speed portion of the User View Panel
- **Result:** The current speed of the train is consistently updated



GPS Speed is set to 100 as default value



Displayed GPS Speed changes almost instantaneously for ever slider value change, meaning displayed value is constantly updated

Test 13:

→ Description:

◆ *Display shows whether wifi is working or not*

→ Expected Result:

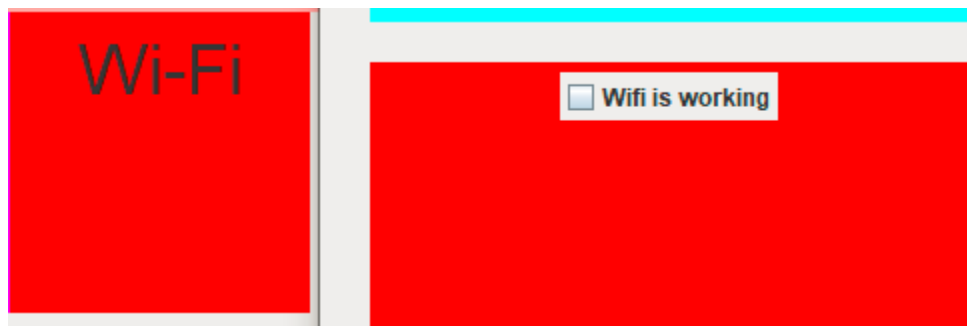
◆ *Display shows wifi panel as green if wifi is working and red if it is not working*

How to test:

- Log into system using Operator Credentials
- Click the “Wifi is working” checkbox on the controller panel
 - Check box is filled:
 - **Result:** “Wi-Fi” square in Operator View Page is colored Green



- Check box is NOT filled:
 - **Result:** “Wi-Fi” square in Operator View Page is colored Red



Test 14:

→ Description:

- ◆ *Determine if password is run through a Sha-256 hashing algorithm when validating log in*

→ Expected Result:

- ◆ *Password is hashed via Sha-256 algorithm*

How to test:

- Our program actually encrypts both the username and password of every user when stored.
- For demonstrational purposes, we have outputted both the plaintext values as well as the hashed values of User credentials into the console of our Java application

Result: Password is hashed via Sha-256 algorithm

```
Un-hashed Username: 1
Un-hashed Password: 1
Hashed Username: 6B86B273FF34FCE19D6B804EFF5A3F5747ADA4EAA22F1D49C01E52DDB7875B4B
Hashed Password: 6B86B273FF34FCE19D6B804EFF5A3F5747ADA4EAA22F1D49C01E52DDB7875B4B
```

Hashed credentials for User account with name and password “1” (output to Java console)

```
Un-hashed Username: admin
Un-hashed Password: admin
Hashed Username: 8C6976E5B54104158DE908BD4DEE15DFB167A9C873FC4B88A81F6F2AB448A918
Hashed Password: 8C6976E5B54104158DE908BD4DEE15DFB167A9C873FC4B88A81F6F2AB448A918
```

Hashed credentials for Administrator account with name and password “admin” (output to Java console)

- You can check that we do not store plaintext credentials by opening our IDandPasswords.java class
 - Usernames and passwords entered by users are hashed, compared against what we have stored, then stored in local variables (for displaying user name on IoT Display) that are cleared when the User logs out

Test 15:

→ Description:

- ◆ *TSNR is currently holding all raw data from sensors*

→ Expected Result:

- ◆ *All data fields within TSNR output non-null values via their get method*

How to test:

- Our “iot_system” Java class doesn’t hold any of the sensor input data, and instead accesses this data by using “get” functions implemented in the “TSNR” class. If the TSNR wasn’t holding the “raw data” (in our programs case, the Controller panel inputs), our program wouldn’t work.
 - Flow of “raw data”:
 - IoT system calls TSNR get method for a specific data field when executing a calculation method
 - IoT system is able to return an output for method called
 - This is done for all methods within our IoT system. For example, If an average value is needed for a function, call calculateAverage(get data from TSNR)
 - It can be assumed that our TSNR is always holding the most recent sensor data values, because our program wouldn’t function otherwise.
-

Test 16:

- Description:
 - ◆ *IoT System is operational within 2 seconds of successful user log-in*
- Expected Result:
 - ◆ *IoT System Display is turned on, and User view is being displayed*

How to test:

- For demonstrational purposes, we output the time of a login attempt (time when login button is pressed) as well as the time when our IoT Display opens up the appropriate User View to the Java console.

```
2021-05-13 20:15:14.381 - login attempt -  
Un-hashed Username: admin  
Un-hashed Password: admin  
Hashed Username: 8C6976E5B5410415BDE908BD4DEE15DFB167A9C873FC4BB8A81F6F2AB448A918  
Hashed Password: 8C6976E5B5410415BDE908BD4DEE15DFB167A9C873FC4BB8A81F6F2AB448A918  
2021-05-13 20:15:14.512 - new login - Current User: admin
```

First and last lines of Java console output are timestamps of login attempts and successful new logins

- We can see here that
 - The login attempt was at 20:15:14.381
 - The login was successful at 20:15:14.512
- The last three numbers represent milliseconds
 - 512ms - 381ms = 131ms

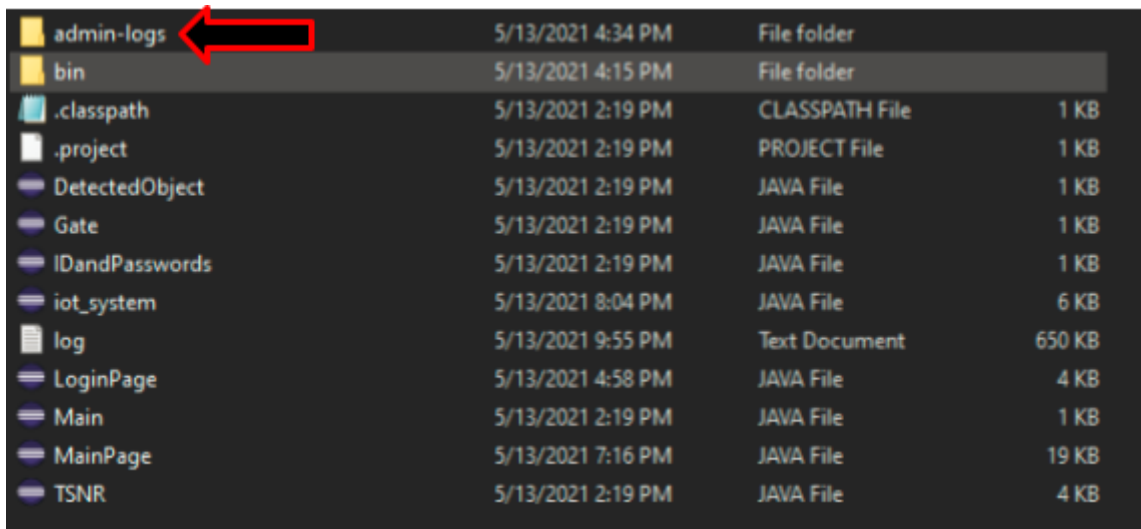
- **Result:** IoT System displays User view in less than 2 seconds of successful user log in
 - Our program is actually operational in less than 200 milliseconds of successful user log-in, far faster than our required 2 seconds

Test 17:

- Description:
 - ◆ Administrator presses the “download log” button
- Expected Result:
 - ◆ Current log file is copied into new timestamped file and new empty log file is created

How to test:

- Log in as an administrator using the username “admin” and the password “admin”
- Press the “download log” button in the top right corner
- The log file is now copied into a folder called “admin-logs” within the top level of the iot file folder



admin-logs	5/13/2021 4:34 PM	File folder	
bin	5/13/2021 4:15 PM	File folder	
.classpath	5/13/2021 2:19 PM	CLASSPATH File	1 KB
.project	5/13/2021 2:19 PM	PROJECT File	1 KB
DetectedObject	5/13/2021 2:19 PM	JAVA File	1 KB
Gate	5/13/2021 2:19 PM	JAVA File	1 KB
IDandPasswords	5/13/2021 2:19 PM	JAVA File	1 KB
iot_system	5/13/2021 8:04 PM	JAVA File	6 KB
log	5/13/2021 9:55 PM	Text Document	650 KB
LoginPage	5/13/2021 4:58 PM	JAVA File	4 KB
Main	5/13/2021 2:19 PM	JAVA File	1 KB
MainPage	5/13/2021 7:16 PM	JAVA File	19 KB
TSNR	5/13/2021 2:19 PM	JAVA File	4 KB

- Within the admin-logs folder the Admin can find the log they requested
- Admin logs are time stamped with format “year|month|day|hour|minute|second”

.gitignore	5/13/2021 2:19 PM	GITIGNORE File	1 KB
admin_log-202105131624.txt	5/13/2021 4:24 PM	Text Document	4 KB
admin_log-202105131628.txt	5/13/2021 4:28 PM	Text Document	3 KB
admin_log-20210513162911.txt	5/13/2021 4:29 PM	Text Document	4 KB
admin_log-20210513163255.txt	5/13/2021 4:32 PM	Text Document	9 KB
admin_log-20210513163419.txt	5/13/2021 4:34 PM	Text Document	4 KB