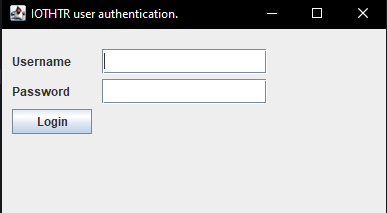
**USER GUIDE**

IOTHTR Engine

Bonnie Nguyen, Roma Razdan, Matthew Viafora, and Michael Zylka

**Introduction and First Start Up:**

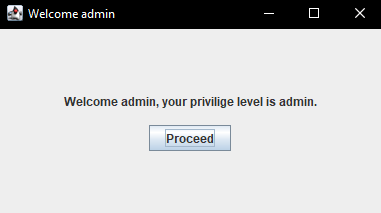
When first starting the IoTEngine, the only user in the system will be the admin. The username is **admin** and the password is **admin.** The admin password can be changed in account configurations and more users can be created with operator privileges in the account configurations as well.



**Username:** admin

**Password:** admin

There will be a greeting for the IoTEngine as shown below. This just displays the name of the user that was logged in and their privilege level. The two privilege levels are **operator** and **admin.** Click proceed to start up the engine.

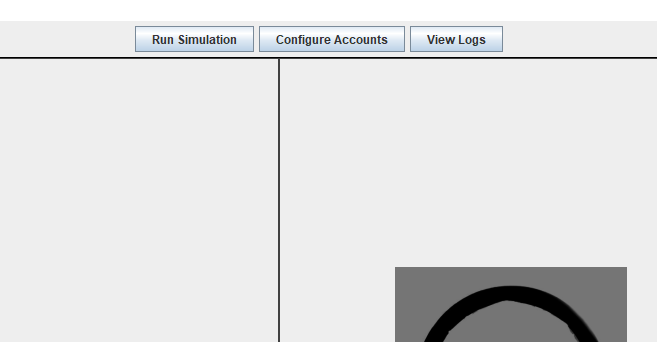
****

The IoTEngine will start up in full screen but it is completely responsive so feel free to drag the window around to fit your workspace better.

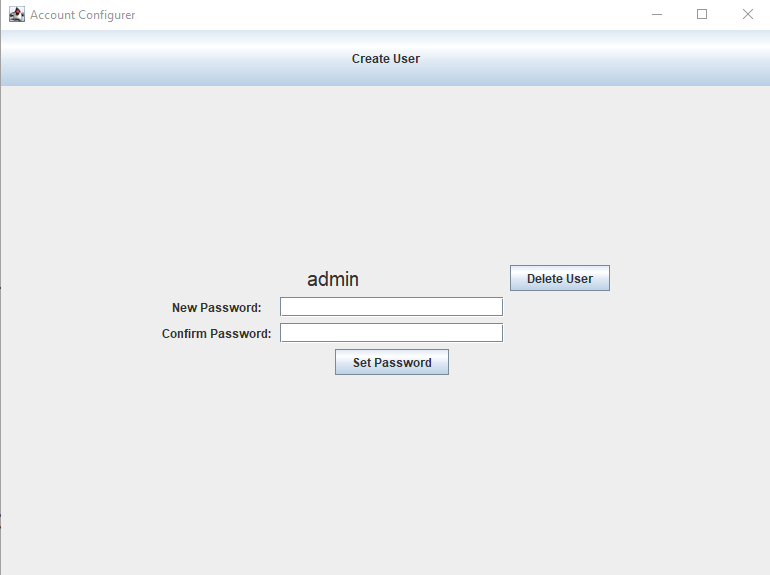
**Note:** On first startup the only user in the engine is the admin. To create a user with operator privileges read **Configure Accounts.** However, if you are running the IoTEngine on a personal computer and not a train, the operator cannot do much as there are no physical sensors to receive input. The admin however has a simulator where you can input specific data to test the system. Check out **Run Simulation** for more.

**Configure Accounts:**

As an admin, you have the feature to create, delete, or change the password of other users. To do so, navigate to the top of IoTEngine and click on **Configure Accounts**.

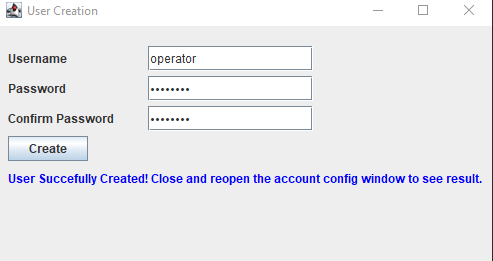


A window will pop up with the ability to change a user’s password, delete a user, or create a new user.



**Create User:**

To create a new user, click on the button at the top that says **Create User** and a new window will come up. Here it will ask for a username and password of the new user. Have the user type in their information and click **Create** when done.

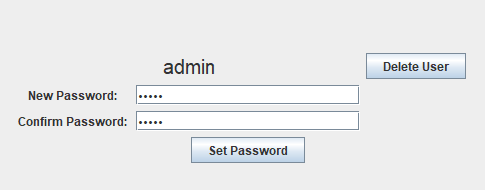


**Note:** If the passwords do not match or if there was a problem creating the user, there will be an error in red stating the information. Just have the user retype the password and the confirm password.

As you see in the figure above, the user was successfully created. To see the changes in the account configurer, you will need to close the window and re-open the window by clicking **Configure Accounts.**

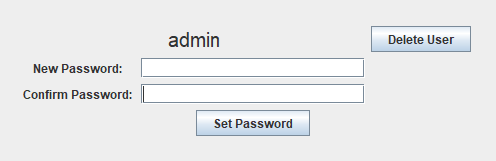
**Change Passwords:**

To Change a users password simply type in the user’s new password and have them retype it to make sure it was typed in correctly. Then hit **Set Password** to overwrite the previous password.

****

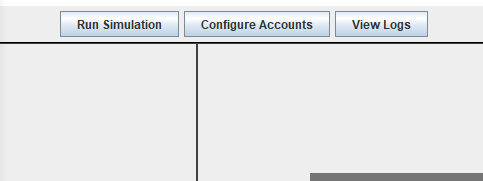
**Delete User:**

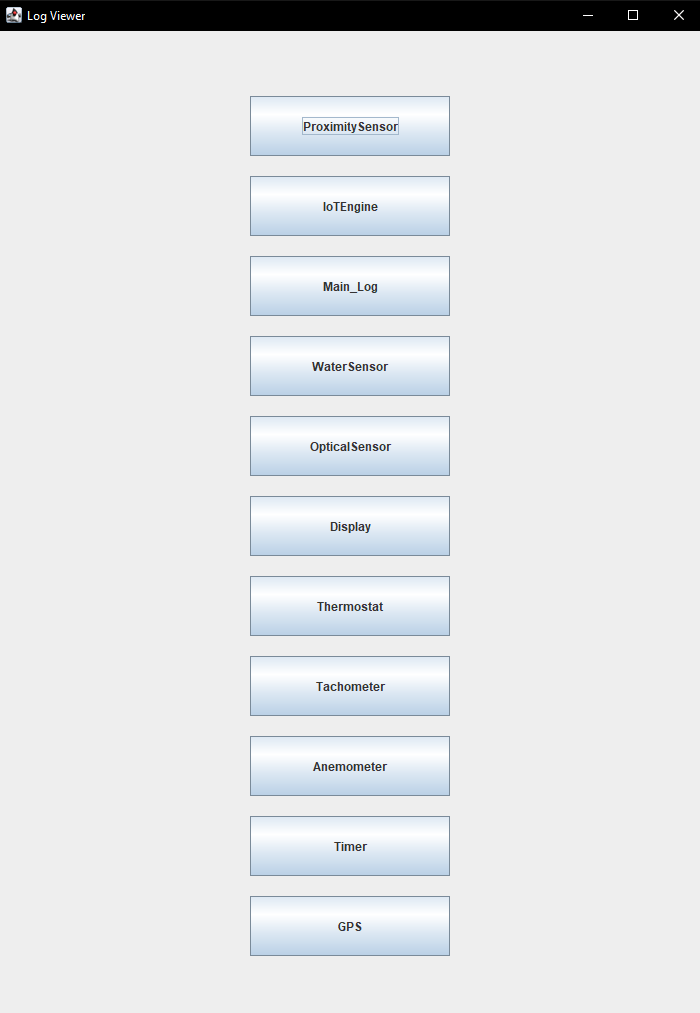
To delete the user simply click the **Delete User** button next to the user you want to delete.



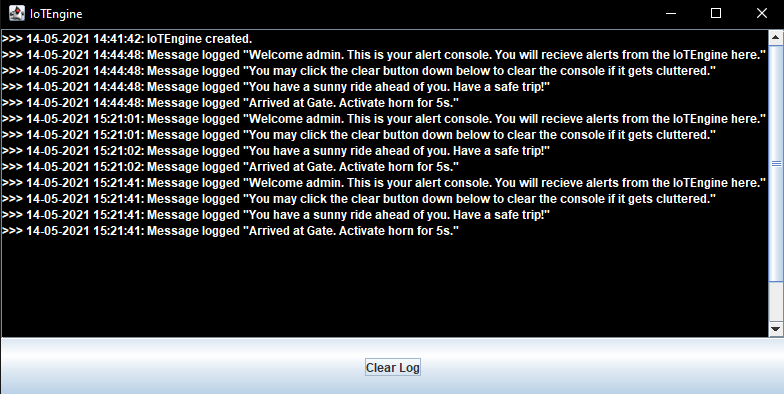
**View Logs:**

To view the logs, click **View Logs** at the top of the IoTEngine and a window will pop up with buttons that will let you see specific logs about that Sensor.





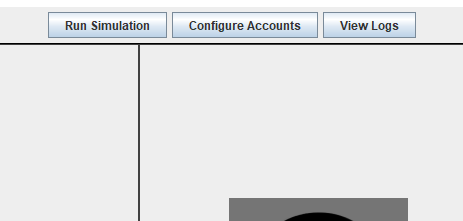
Clicking on a button like IoTEngine for example will open a window for that Log.



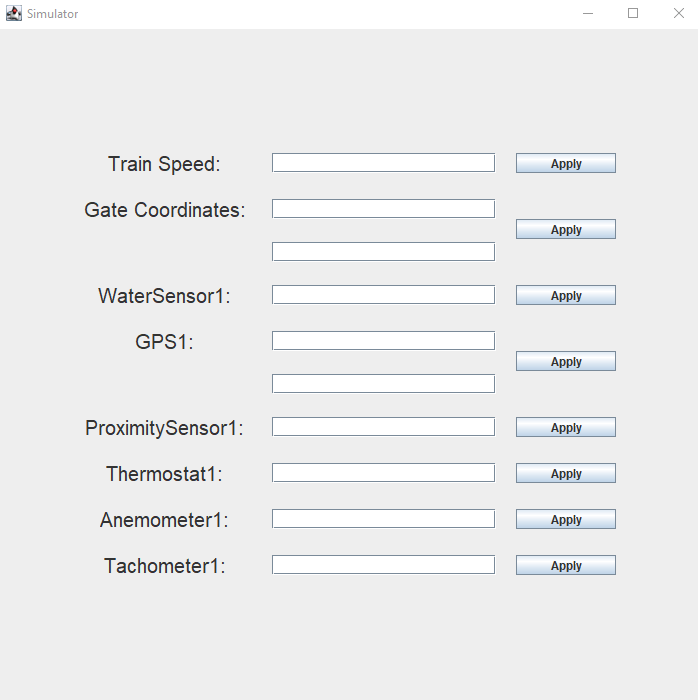
There is also a clear log button at the bottom that will allow the admin to wipe the log completely.

**Run Simulation:**

As an admin, you have the ability to run a simulation to test the entire system. To start up the simulation simply go to the top of the screen and click **Run Simulation.**

****

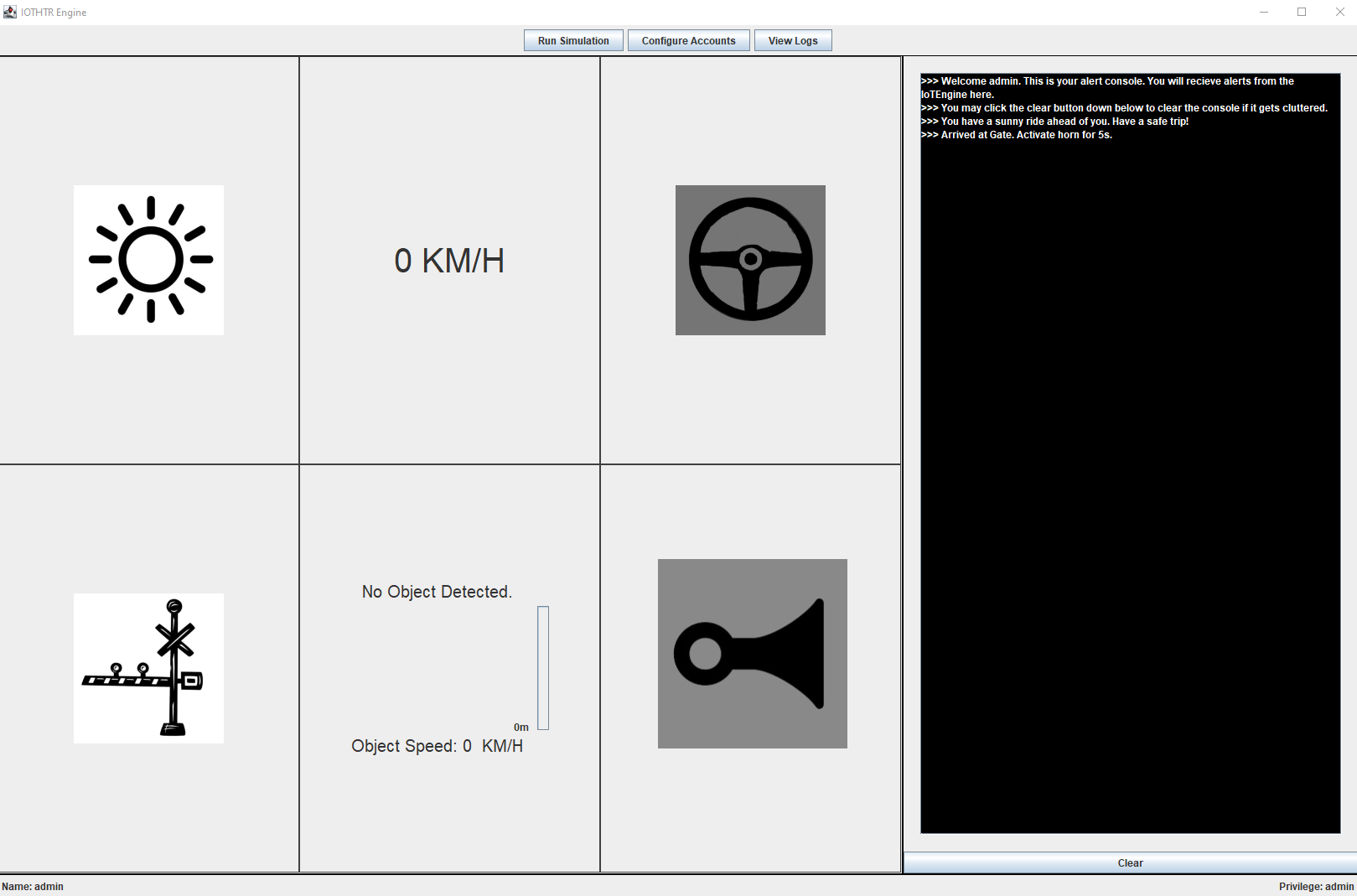
Here you will be able to manually enter data to each sensor and it will be run by the IoTEngine and displayed in the IOTHTR Engine.



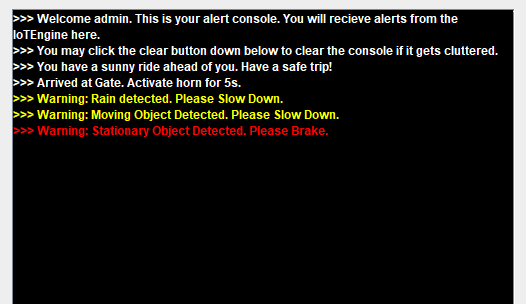
Once the data is inputted into the corresponding text box, just click the apply button and you will see the data affect the IOTHTR Engine window.

**General Usage:**

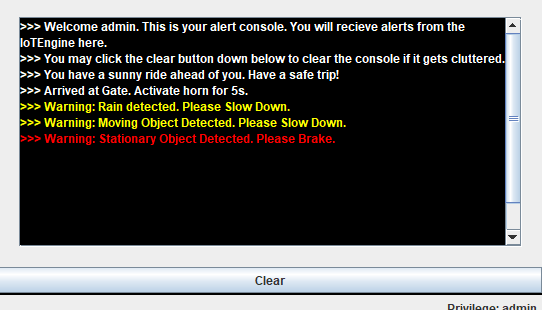
The IOTHTR Engine window is the main window where all of the data can be viewed by the operators and the admins.



The 6 panels in the middle of the screen is where the operator will see general information like the weather conditions, the speed of the train, the distance of objects, etc. On the right of the screen is the operator’s console. This console will display information and recommendations to the operator. The operator will be able to react to the messages and react accordingly.



At the bottom of the operator’s console, there is a **Clear** button that will allow the operator to fully clear the screen in case it is too cluttered.



**Testing:**

The following are test cases that can be run by the admin in the simulation window to make sure the features and the sensors work correctly.

**Validation Testing**

**Test Case 1 Description:** IoT Engine suggests immediate braking due to standing hazard.

**Test Sim Input:** Set the train speed to 0. Set the proximity sensor to 450m and then hit apply.

Then set the proximity sensor to 450m again and hit apply again.

**Conditions:** Proximity sensor returns: (distance: 450m, speed: 0)

**IoT Actions:** Display warning (decrease speed), log standing hazard

**Test Case 2 Description:** IoT Engine suggests immediate braking due to moving.

**Test Sim Input:** Set the train speed to 0. Set the proximity sensor to 749m and then hit apply.

Then set the proximity sensor to 750m again and hit apply again.

**Conditions:** Proximity sensor returns: (distance: 750m, speed: 14)

**IoT Actions:** Display warning (decrease speed), log moving hazard

**Test Case 3 Description:** IoT Engine suggests breaking due to wheel slippage.

**Test Sim Input:** Set the RPM to 38. Set the first GPS coords to **40.74729510704591,**

**-74.02790421415445.** Set the secondGPS coords to **40.74728901096995,**

**-74.0278954969635.**

**Conditions:** Tachometer returns: (RPM: 38) GPS returns: (**point1:** 40.74729510704591, -74.02790421415445; **point2:**40.74728901096995, -74.0278954969635)

**IoT Action:** Display warning (decrease speed), log wheel slippage

**Test Case 4 Description:** IoT Engine does not suggest breaking due to wheel slippage.

**Test Sim Input:** Set the RPM to 40. Set the first GPS coords to **40.74729510704591,**

**-74.02790421415445.** Set the secondGPS coords to **40.74728901096995,**

**-74.0278954969635.**

**Conditions:** Tachometer returns: (RPM: 40) GPS returns: (**point1:** 40.74729510704591, -74.02790421415445; **point2:**40.74728901096995, -74.0278954969635)

**IoT Action:** Turn off slip icon

**Test Case 4 Description:** IoT Engine suggests change of speed to accommodate for weather severity.

**Test Sim Input:** Set the water sensor to 1;

**Conditions:** Water sensor returns: (Rain: True)

**IoT Action:** Display warning (decrease speed), log weather (rain)

**Test Case 5 Description:** IoT Engine suggests change of speed to accommodate for weather severity.

**Test Sim Input:** Set the anemometer to 55;

**Conditions:** Anemometer returns (Wind speed: 55km/h)

**IoT Action:** Display warning (decrease speed), log weather (wind)

**Test Case 6 Description:** IoT Engine suggests change of speed to accommodate for weather severity.

**Test Sim Input:** Set the Temperature to -4 and water sensor to 1 ;

**Conditions:** Thermometer returns: (Temperature: -4 degrees fahrenheit)

**IoT Action:** Display warning (decrease speed), log weather (temperature)

**Test Case 7 Description:** IoT Engine displays horn blowing

**Test Sim Input:** set the GPS coords to **40.75024763739186, -74.04664947835427.** Set the gate

coords to **40.74722320391527, -74.02793917030857.**

**Conditions:** Train GPS coordinates: (40.75024763739186, -74.04664947835427), Gate GPS coordinates: (40.74722320391527, -74.02793917030857)

**IoT Action:** Display Horn for 15 seconds, log 1 mile to gate

**Test Case 7 Description:** IoT Engine displays horn blowing

**Test Sim Input:** set the GPS coords to **40.744748, -74.028704.** Set the gate

coords to **40.744748, -74.028704.**

**Conditions:** Train GPS coordinates: (40.744748, -74.028704), Gate GPS coordinates: (40.744748, -74.028704)

**IoT Action:** Display Horn for 5 seconds, log gate crossing

#### **Scenario-Based Testing**

**Test Case 1:** User logs in

**Description:** The user logs into the IoT Engine via GUI after being granted access and permissions.

**Action:** User initiates Logon(name,pass).

**Expected:** Access granted if valid user name and password is presented.

**Exception:** Access denied if invalid user name and password is presented.

**Test Case 2:** User activates IoTEnigne.

**Description:** The user activates IoT Engine via GUI after successfully logging in.

**Action:** User initiates activateIoTEngine(verify).

**Expected:** Success

**Exception:** Failure if an error occurs

**Test Case 4:** Admin creates a new user.

**Description:** The admin creates a new user via GUI that will have access to user tools in the IoT Engine.

**Action:** User initiates newUser(AdminAccess,name,pass)

**Expected:** User created successfully

**Exception:**

* User not created if error occurs
* User not created if user name or password contains invalid characters

**Test Case 6:** User commits manual log entry to log.

**Description:** The user enters a manual log entry into the log via GUI.

**Action:** User initiates logEntry(manEntry)

**Expected:** Entry successfully committed.

**Exception:** N/A