Web Application for Aqualab Sensor Monitoring and Analysis

System Design Document

Prepared by:	Gregory Thompson - gthompson2022@my.fit.edu Haley Hamilton - hamiltonh2021@my.fit.edu Ruth Garcia - ruth2021@my.fit.edu
Project Advisor:	Dr. Slhoub - kslhoub@fit.edu
Project Client:	Dr.Turingan

Version:	0.2
Date Created:	09/22/2024

Table of contents

1. Introduction

1.1 Purpose

The purpose of this project is to develop a system that reads and displays data for use in Dr. Turingan's lab research. The product allows him and his lab team to connect their laboratory sensors/apparatuses to a system and view the data remotely and in real time as well as give alerts when measurements are outside of desired ranges. This document aims to clearly and concretely describe the system design of this project.

1.2 Scope

The scope of this product is to develop a web application that will connect to and read in data from sensors and display it to allow the client to view the data and analyze it. This capability gives the client a centralized system that automatically reads and records data from different sensors, alerts them when sensor measurements are not in the expected range, and gives them a versatile tool to compare and analyze the data. By implementing this system, the client will streamline their research, making the process more efficient and reducing the likelihood of errors.

1.3 Objective

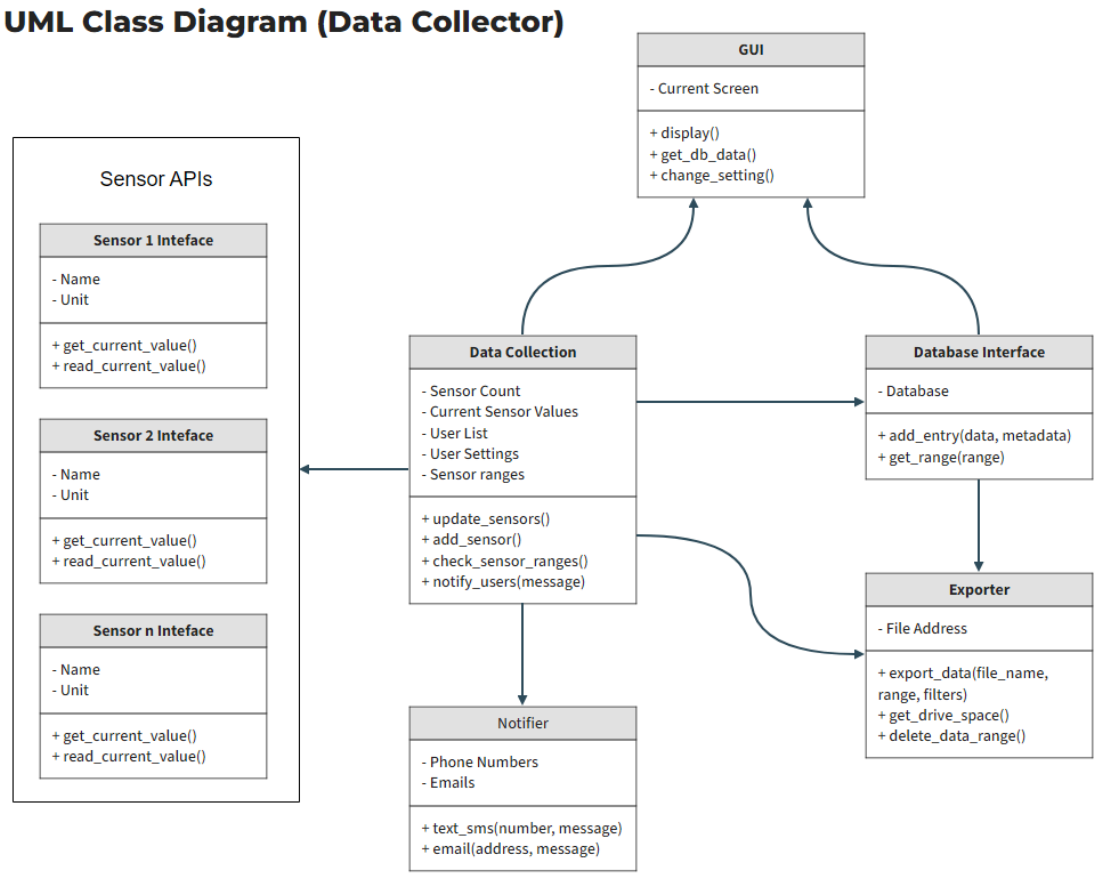
The code will operate in two distinct programs. The data collection program will be able to simultaneously monitor and record the data from any number (under 255) of sensors being used by Dr. Turingan and his lab team (henceforth the lab team). The lab team will be able to view the current sensor values, access and analyze previous sensor values, and be notified via email and sms if sensor values exceed acceptable ranges. The lab team will be able to access the data and perform specific tasks remotely using a JavaScript web server. Data analysis will be done using a separate data analysis program designed to lab team specifications.

1.4 Key Interactions

Sensors will connect to the central computer through a variety of sensor-specific methods including but not limited to USB and Ethernet. The data from these sensors will be stored in a database on the central computer's drive. The software will use libraries to enable the sending of sms and email messages to the lab team as needed. All data, accessed locally or remotely, shall be displayed on a graphical user interface.

2. System Architecture

2.1 UML Class Diagram (Data Collector and Display)



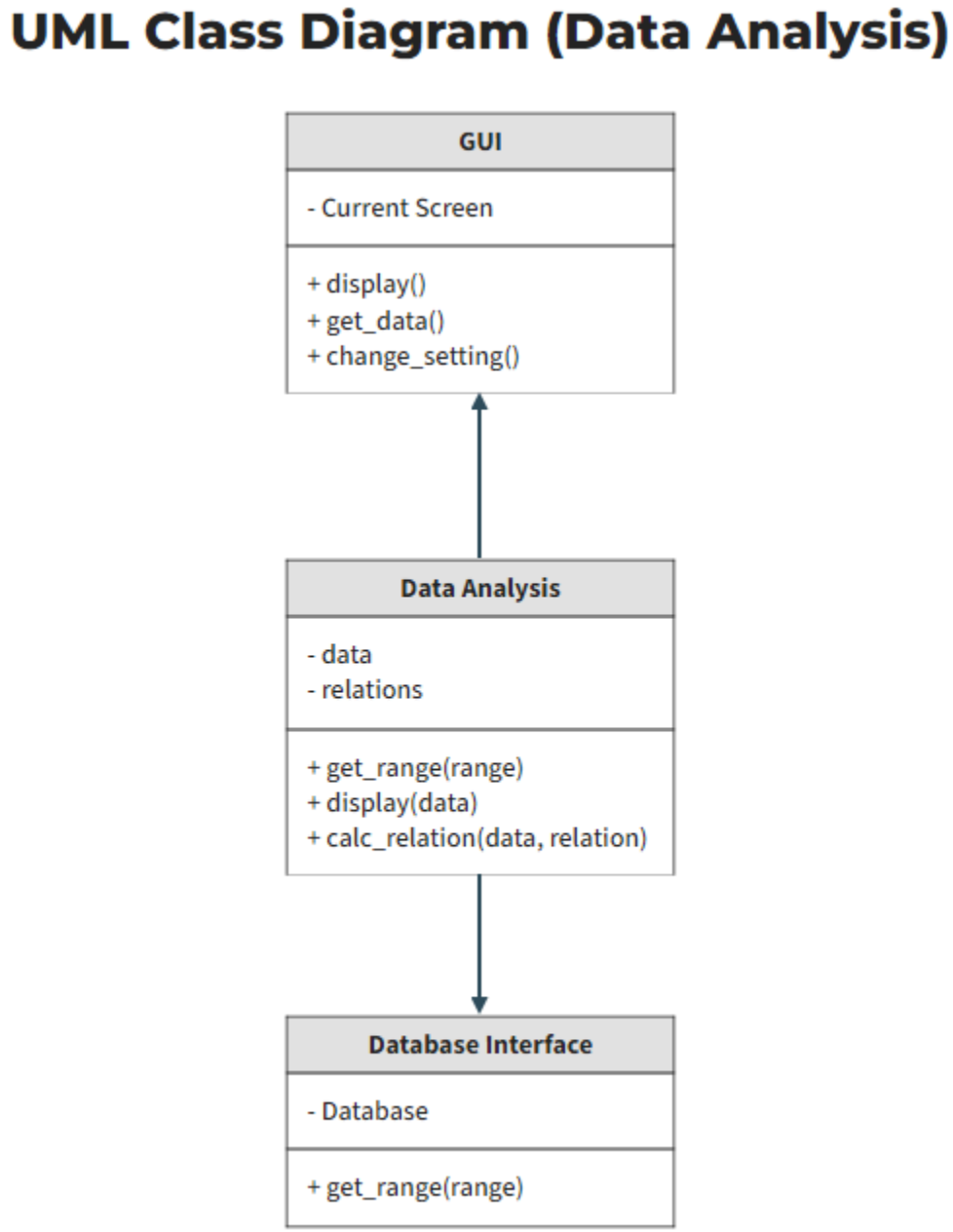
Class Details:

- **Data Collection:** This is the collection application's entry point. The settings, user data, and other core features will operate from this class. It is also responsible for verifying that data from sensor APIs is in acceptable range. This class will create threads for the sensor APIs to run on, collect data from the sensor APIs, and send that data to the database interface. If the data is out of acceptable range, this class will call the notifier class to handle SMS or email as necessary.
- **Sensor APIs:** These classes will be tailored to specific sensor types and communicate their data to the data collection class. These will each monitor sensor data on their own thread to allow near-simultaneous execution. The data collection class will call the get_current_value method to receive the most recent reading.
- **Notifier:** The notifier class' role is to send SMS and email messages to users. This will happen once during setup to confirm information is entered correctly, and anytime the data collection class decides data is out of range. The Sinch API will be used to send SMS and the Python SMTP and Email libraries to send emails.
- **GUI:** The GUI class is responsible for managing the graphical user interface. It will display data gathered from both the data collection class (such as settings or

out-of-range notifications) and the database interface class (Past sensor readings). The GUI will transmit user inputs to both of those classes as necessary. The Plotly and Dash libraries will be used to display graphs and UI elements respectively.

- **Database Interface:** The database interface shall maintain the database, adding, removing, or modifying data as determined by the other classes.
- **Exporter:** The exporter's role is to allow the user to transform their data into a CSV file. This is to make it easier for the data to be used by other common analysis programs such as Excel. The Exporter class will also allow data to be deleted from the database after it has been exported to support drive space management.

2.2 UML Class Diagram (Data Analysis)

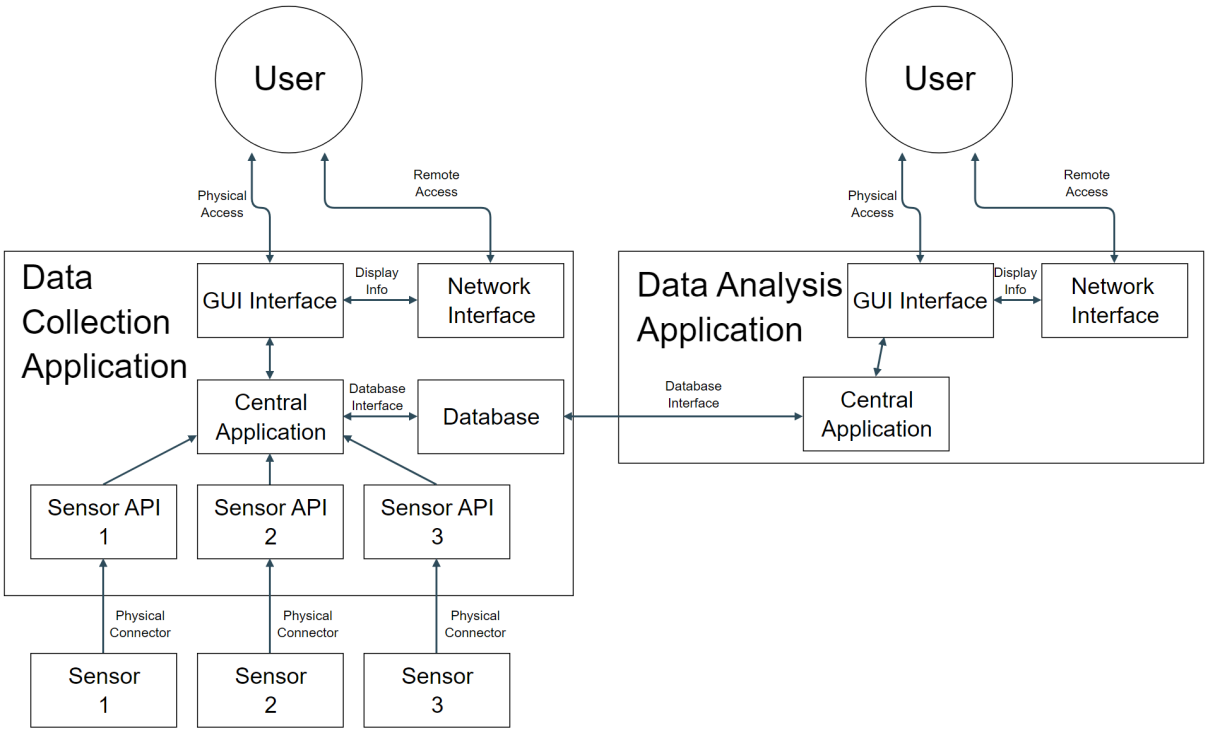


Class Details:

- **Data Analysis:** This is the analysis application's entry point. This class will store important relations as determined by the lab team as well as support the addition of new relations anytime during operation. Data necessary for the calculating of these relations will be collected from the database interface. This class will then calculate the relations and pass the necessary results to the GUI class for display to the user.

- **GUI:** This application's GUI class will be responsible for displaying important relations to the user. The user will be able to select known relationships or create new ones, which will then be displayed on the analysis user interface.
- **Database Interface:** This application's database interface will only support the retrieval of data, not removal or modification. Queries will be made by the data analysis class and translated by this interface. Data will then be retrieved from the database and passed back to data analysis.

2.3 High-Level Architecture



Sensors connect to the data collection application through sensor APIs. Data is stored and retrieved from the database through an API. Necessary information passes to the GUI which displays on the host device or remotely. The data analysis application reads from the same database, processes analysis internally, and displays similarly to collection.

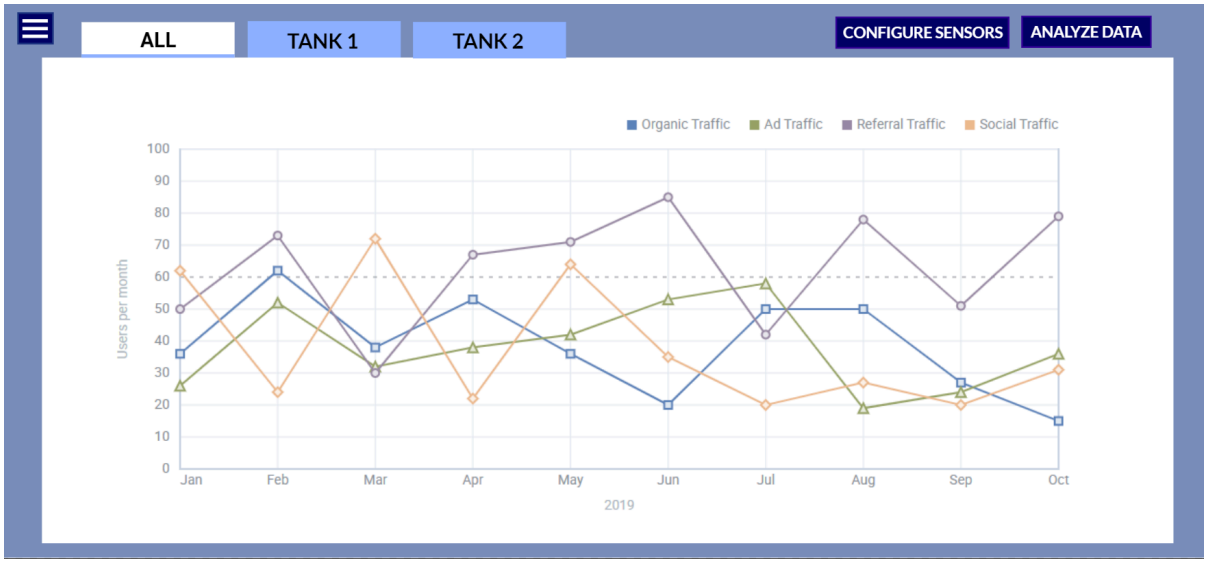
3. User Interface Design

3.1 Details

The user interface for the data collection will be organized by a hamburger menu. On initial startup, it will show a setup screen for connecting to new sensors. Once the experiment is set up, the home screen will show a general graph of all sensor data. Tabs in the menu will include a tab for viewing specific sensor data, a tab for exporting data in specific ranges, and a settings tab. The specific sensor data tab will display an initially blank graph with buttons to add and modify specific data from sensors. The Export tab will display various options for what data the user wants to be exported. The settings tab will display a list of settings determined by user permissions. These settings include user management, contact preferences, and drive space management settings.

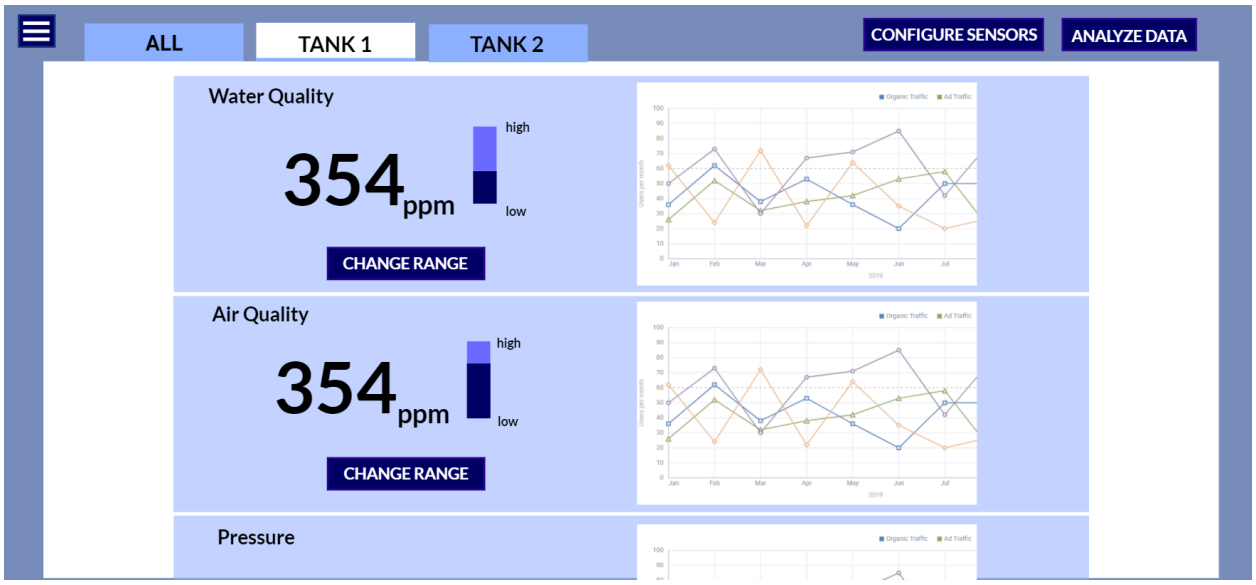
3.2 Home Page

3.2.1: Sample homepage, displays all data and tabs for specific data.

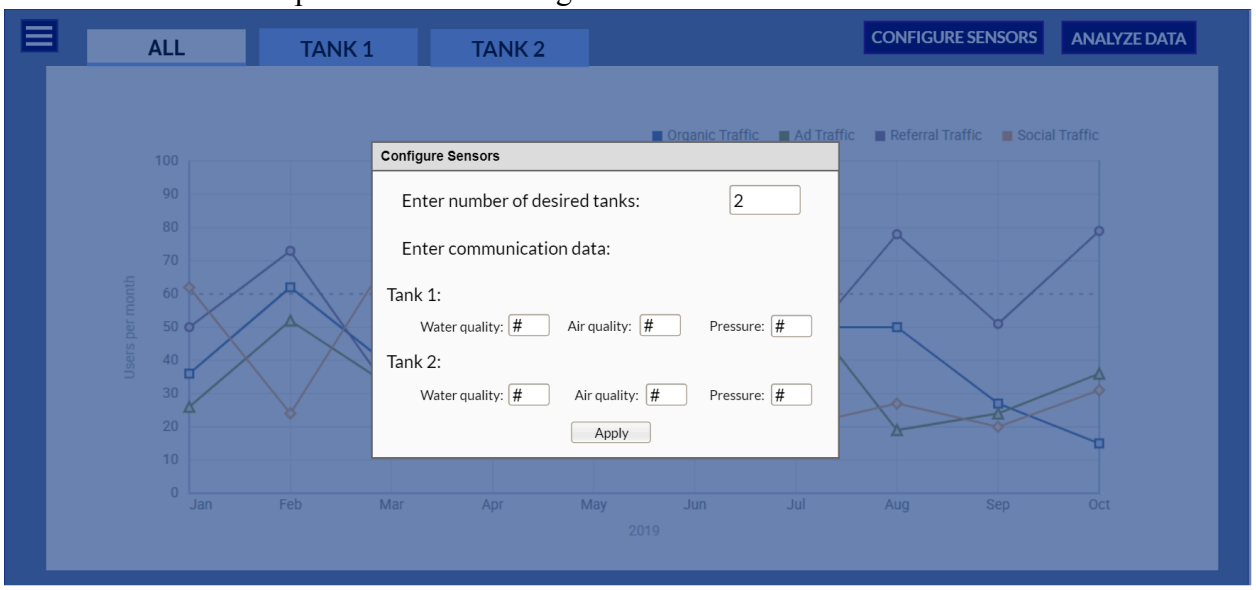


3.2.2: Homepage (background), with hamburger menu extended on the left.

3.2.3: An individual view of an experiment tank with each sensor shown in detail.



3.2.4: A mockup of the sensor configuration GUI.

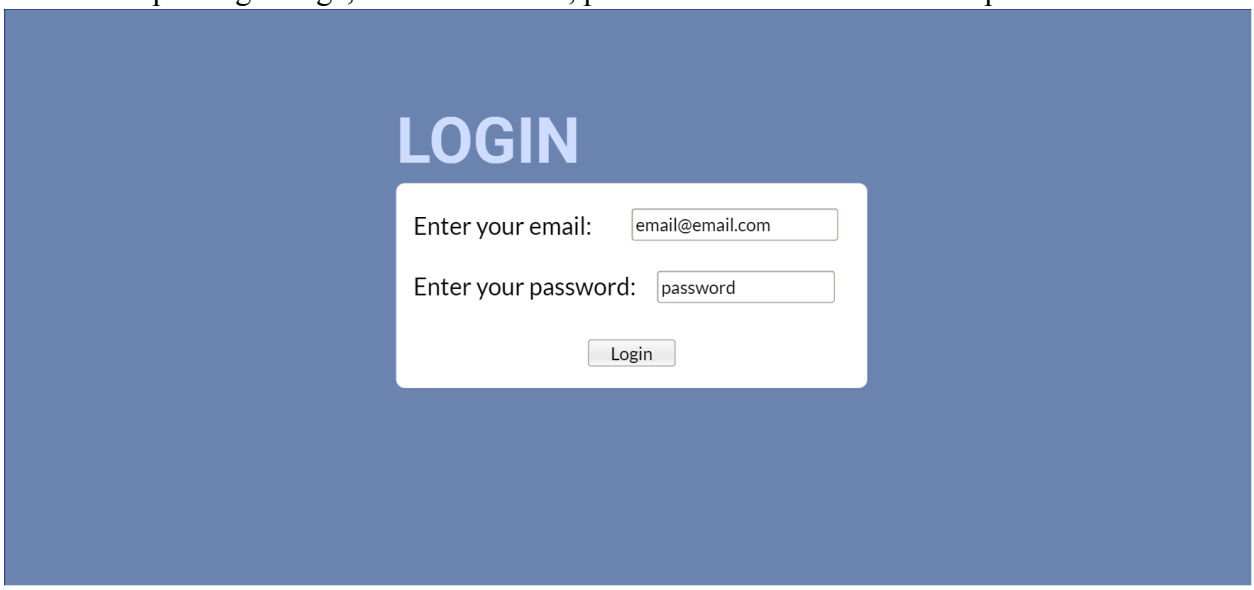


3.3 Analysis Tool

Sample analysis tool, shows potential comparison techniques for data.

3.4 Login Page

Sample Login Page, Email as user id, password will be dotted in complete version..

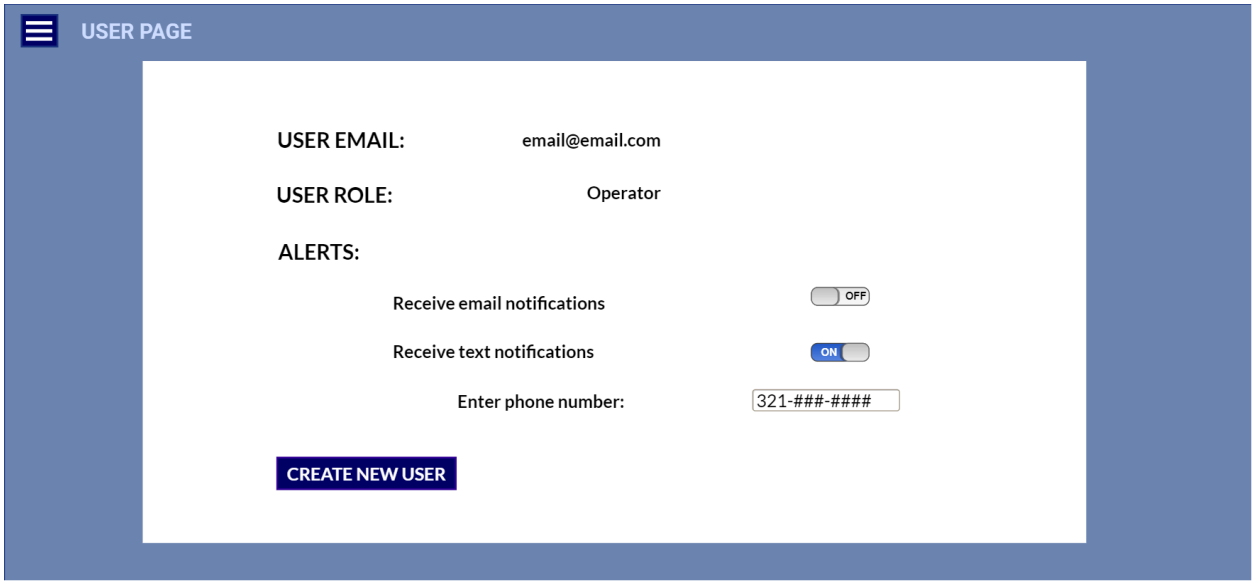


3.5 Settings

Sample Settings page, ability to select specific tank settings as well as full system settings.

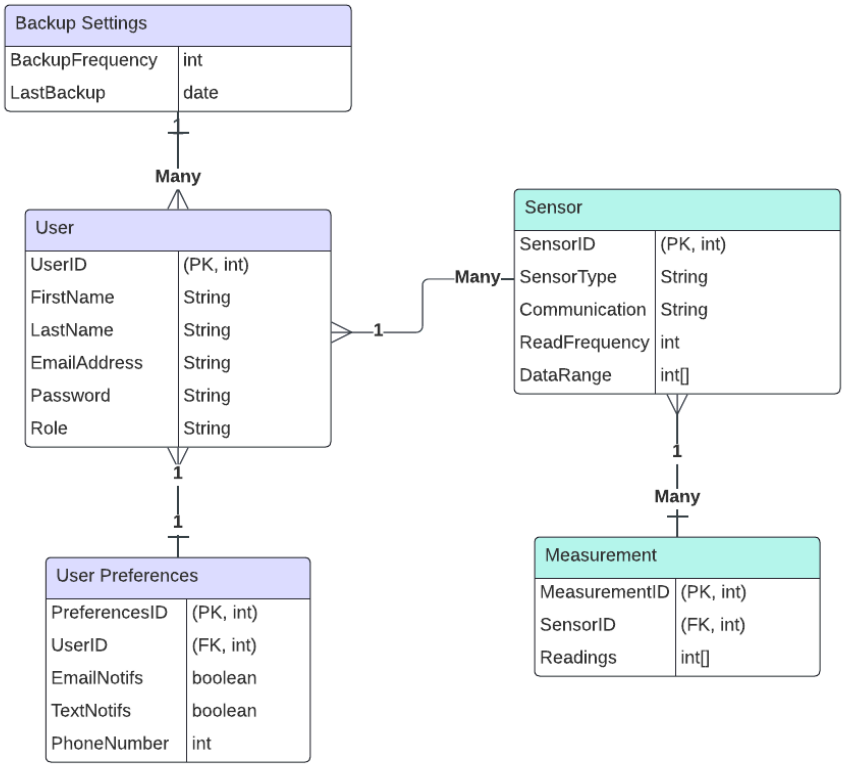
3.6 User Page

Admin user creation page, Allows administrator to create and set roles for a new user. All users will be made directly by the admin.



4. Database Design

4.1 Entity Relationship Diagram



4.2 User Collection

4.2.1 User Records

- UserID (Primary Key): A unique identifier for each user.
- FirstName: The user's first name.
- LastName: The user's last name.
- EmailAddress: The user's Florida Tech email address.
- Password: The user's password.
- Role: The user's role.

4.2.2 User Description

This collection is used to store all the user's information and differentiate between the users. The email is used as the username and with the password is used to validate users and allow them access to the application. The role specifies the role of the user and what application features they are allowed to access.

4.3 User Preferences Collection

4.3.1 User Preferences Records

- PreferencesID (Primary Key): A unique identifier for each user's preferences.
- UserID (Foreign Key): Linked to the User Collection.
- EmailNotifs: Whether the user wants email notifications or not.
- TextNotifs: Whether the user wants text notifications or not.
- PhoneNumber: The user's phone number (if they want text notifications).

4.3.2 User Preferences Description

This collection is used to store the user's notification preferences and whether they would like to receive email or/and text notifications, and if they would like to receive text notifications, the system inputs their phone number.

4.4 Backup Settings Collection

4.4.1 Backup Settings Records

- BackupFrequency: The frequency of which data is backed up to the cloud.
- LastBackup: The date of the last time data was backed up to the cloud.

4.4.2 Backup Settings Description

This collection is used to store data about the whole system, specifically how often the system is set to back up data to the cloud and the last date the system performed a cloud backup.

4.5 Sensor Collection

4.5.1 Sensor Records

- SensorID (Primary Key): A unique identifier for each sensor.
- SensorType: The type of sensor: water quality, air quality, or pressure.
- Communication: The communication data needed for the sensor to connect to the sensor.
- ReadFrequency: The frequency of which data is read from the sensor.
- DataRange: The desired range/value of the sensor measurements.

4.5.2 Sensor Description

This collection stores data about the sensors in use, including their unique identifier, the type of sensor, and the data needed to communicate with them (e.g. a COM port number). This field also includes user customizable fields such as the frequency at which data is read from the sensor and the desired range or value of the measurements from the sensor.

4.6 Measurements Collection

4.6.1 Measurements Records

- MeasurementID (Primary Key): A unique identifier for each measurement.
- SensorID (Foreign Key): Linked to the Sensor Collection.
- Readings: The measurements and readings from the sensor.

4.6.2 Measurements Description

This collection stores all the measurements from the sensor. It includes the unique identifier for each measurement, the unique identifier of the sensor it is reading from, and all the data.