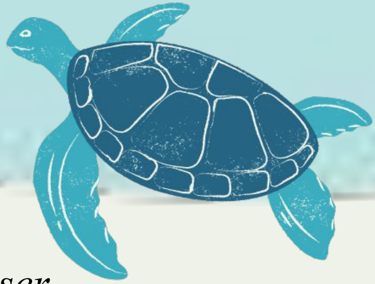
# Web Application for Aqualab Sensor Monitoring and Analysis - Milestone 1

Ruth Garcia, Haley Hamilton, Greg Thompson

# Milestone 1 Overview:



**Compare and select technical tools for:**

*-communicating with sensors, displaying the data, data analysis tools, user interface, recording data, and accessing recorded data*

**Provide small ("hello world") demo(s) to evaluate the tools for:**

-*communicating with sensors, displaying the data, data analysis tools, user interface, recording data and uploading to cloud, accessing recorded data*

**Resolve technical challenges:**

-*Connecting to different sensors via different APIs/connections and libraries, Collecting data and displaying it accurately in real time, Hosting a server for 24/7 access that is accessible anywhere, Displaying/plotting data over time in an easy to read graph*

**Compare and select collaboration tools for software development, documents/ presentations, communication, task calendar**

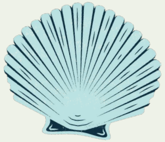
**Create Requirement Document, Design Document, Test Plan**

# Collaboration Tools

- Code Development and Code Collaboration:
  - Github
  - Visual Studio Code/personal IDE
- Task Management and Task Calendar:
  - Jira

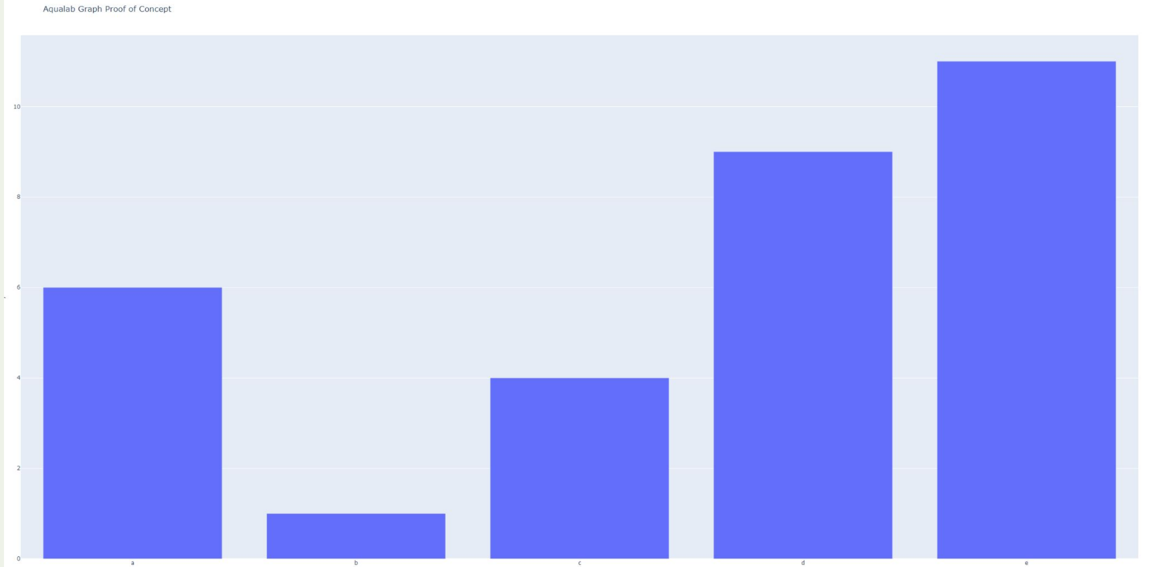
# Technical Tools

- Communicating with sensors:
  - Water Quality Sensor - Manta+40 sensor, RS232-USB connection, pyserial library
  - Air Quality and Pressure Sensor - Vernier sensors, Arduino Interface Shield and Arduino hardware, USB connection, pyserial library and Arduino code
- Displaying the data: Plotly library for Python.
- Data analysis tools: Pandas library for Python.
- User interface: React/JavaScript.
- Recording and Accessing data: MongoDB
- General Framework: MongoDB Database + Flask Backend + React Frontend



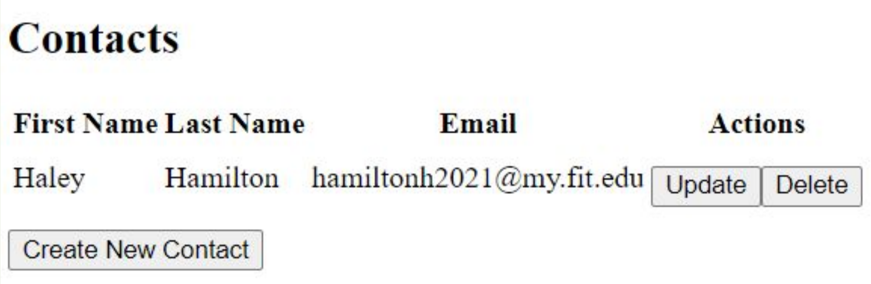
# Demos:

- **Displaying data with Plotly graph proof of concept:**



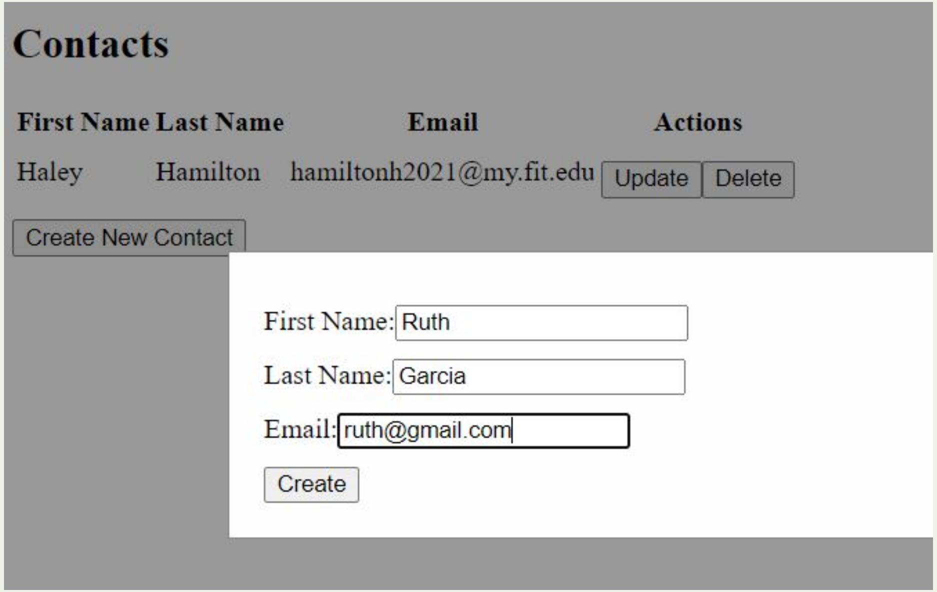
# Demos:

- **Framework proof of concept (basic CRUD operations):**



## Contacts

| First Name | Last Name | Email | Actions |
|---|---|---|---|
| Haley | Hamilton | hamiltonh2021@my.fit.edu | Update Delete |



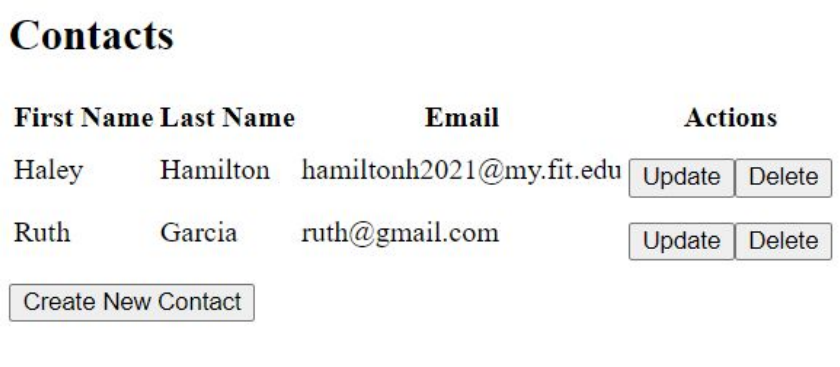
## Contacts

| First Name | Last Name | Email | Actions |
|---|---|---|---|
| Haley | Hamilton | hamiltonh2021@my.fit.edu | Update Delete |

First Name: Ruth

Last Name: Garcia

Email: ruth@gmail.com

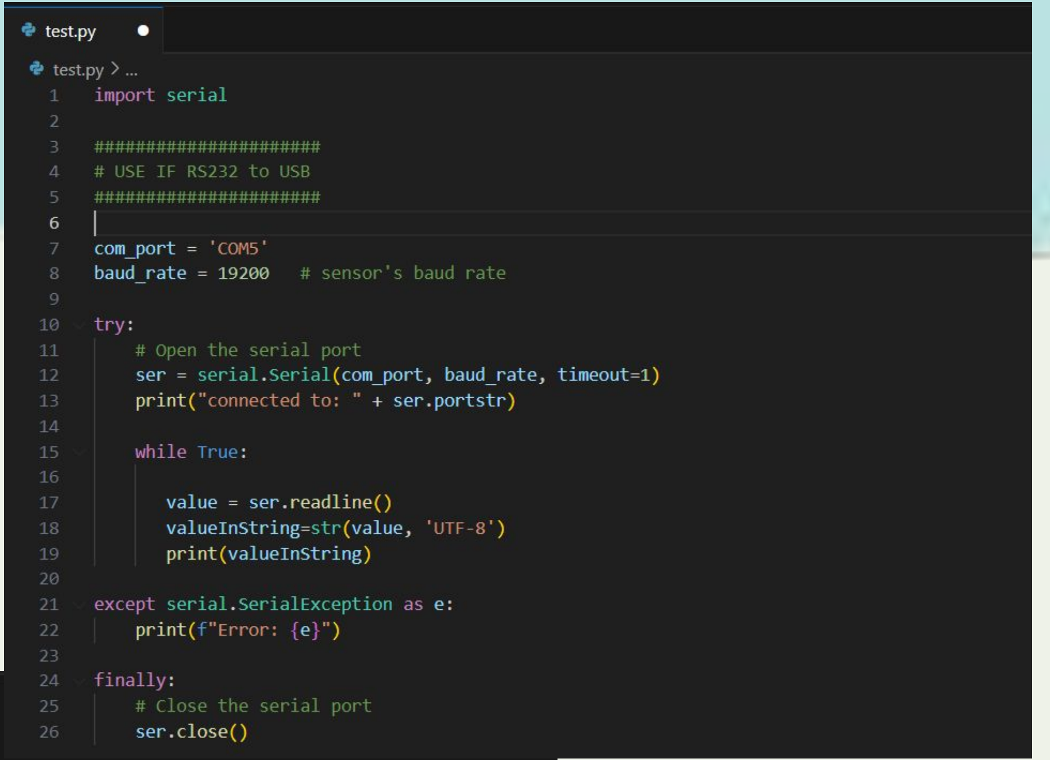


## Contacts

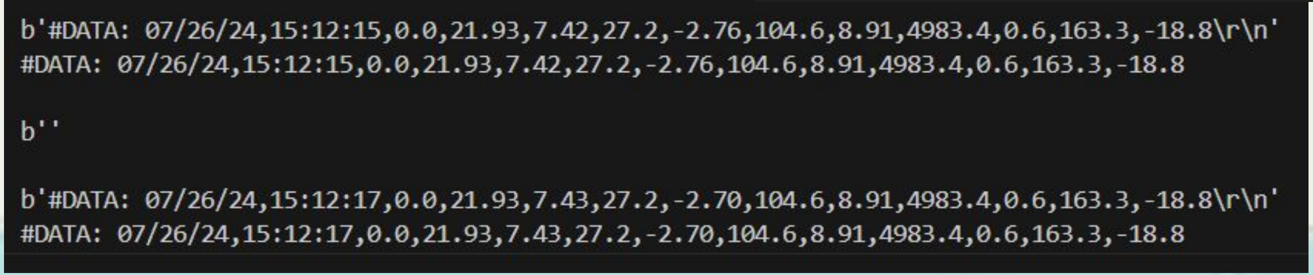
| First Name | Last Name | Email | Actions |
|---|---|---|---|
| Haley | Hamilton | hamiltonh2021@my.fit.edu | Update Delete |
| Ruth | Garcia | ruth@gmail.com | Update Delete |

# Demos:

- **Communicating with sensors proof of concept:**



```python
import serial

######################
# USE IF RS232 to USB
######################

com_port = 'COM5'
baud_rate = 19200   # sensor's baud rate

try:
    # Open the serial port
    ser = serial.Serial(com_port, baud_rate, timeout=1)
    print("connected to: " + ser.portstr)

    while True:

        value = ser.readline()
        valueInString=str(value, 'UTF-8')
        print(valueInString)

except serial.SerialException as e:
    print(f"Error: {e}")

finally:
    # Close the serial port
    ser.close()
```



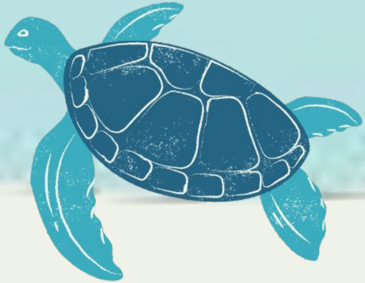
```
b'#DATA: 07/26/24,15:12:15,0.0,21.93,7.42,27.2,-2.76,104.6,8.91,4983.4,0.6,163.3,-18.8\r\n'
#DATA: 07/26/24,15:12:15,0.0,21.93,7.42,27.2,-2.76,104.6,8.91,4983.4,0.6,163.3,-18.8

b''

b'#DATA: 07/26/24,15:12:17,0.0,21.93,7.43,27.2,-2.70,104.6,8.91,4983.4,0.6,163.3,-18.8\r\n'
#DATA: 07/26/24,15:12:17,0.0,21.93,7.43,27.2,-2.70,104.6,8.91,4983.4,0.6,163.3,-18.8
```

# Requirements

# External Interfaces:

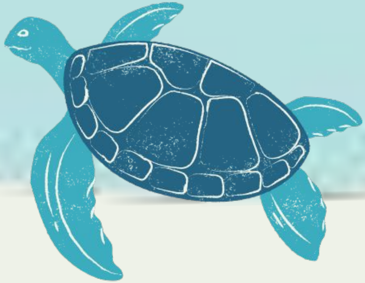
- User Interfaces: User interacts with software via a different screens/pages of the web app (click through screens, click buttons, and submit input)
- Hardware Interfaces: Interfaces with the sensors (water quality, air quality, and pressure) using wired connections (RS232-USB, Arduino/Arduino Interface Shield)
- Software Interfaces:
  - Interfaces with APIs/ libraries for sensors (e.g. pyserial).
  - Interfaces with database (MongoDB), backend (Flask),  and frontend (React)
- Communications Interfaces:
  - HTTP/HTTPS protocol for secure web application communication
  - Communicate with users via phone/email push notification

# Functional Requirements:



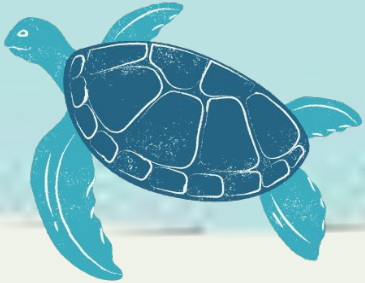
- Sensor Connections:
  - REQ1-3: The system shall utilize the necessary physical hardware as well as libraries or API's to connect with and read from the sensors
  - REQ4-5: The system shall allow Admin users to input connection information about the sensors so the system can connect to them and configure the number/type of sensors.
- Monitoring Current/Recent Sensor Data:
  - REQ-6: The system shall display the current and recent measurements read from the sensors.
  - REQ7-8: The system shall allow Admin users to enter desired ranges/values for each sensor and alert users if the sensor data does not fall within the specified range/value via an on screen alert and a push notification.

# Functional Requirements:



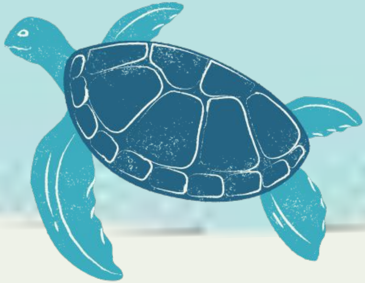
- **Analysis of Past Measurements:**
  - REQ9-11: The system shall record past measurements for the sensors to a database, plot all recorded data in a graph, and receive user input to filter through data
  - REQ-12: The system shall use recorded data to calculate and display relationships between sensor data as requested by the client.
  - REQ-13: The system shall allow users to export collected measurements (filtered or unfiltered) into a CSV file that can be downloaded to their computer.
  - REQ-14: The system shall allow the Admin user to change the frequency of when data is recorded to the database.

# Functional Requirements:

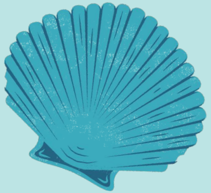


- Mitigate Disk Overflow Risk:
  - REQ15/16: The system shall display how much local disk storage is currently being taken up and alert the user when its getting full.
  - REQ17-18: The system shall have a default backup method where data is uploaded to a cloud, allow Admin change the data cloud backup settings and move/delete recorded data.
- User Authentication and Security:
  - REQ20, 23: The system shall allow users to log in and specify receiving email or text notifications.
  - REQ-21: The system shall allow Admin users to create a new user.
  - REQ-22: The system shall have three different role types, Admin, Operator, and Observer, each with different levels of user privileges and access.
  - REQ-24: The system shall log user login/logout activity.

# Non-Functional Requirements:

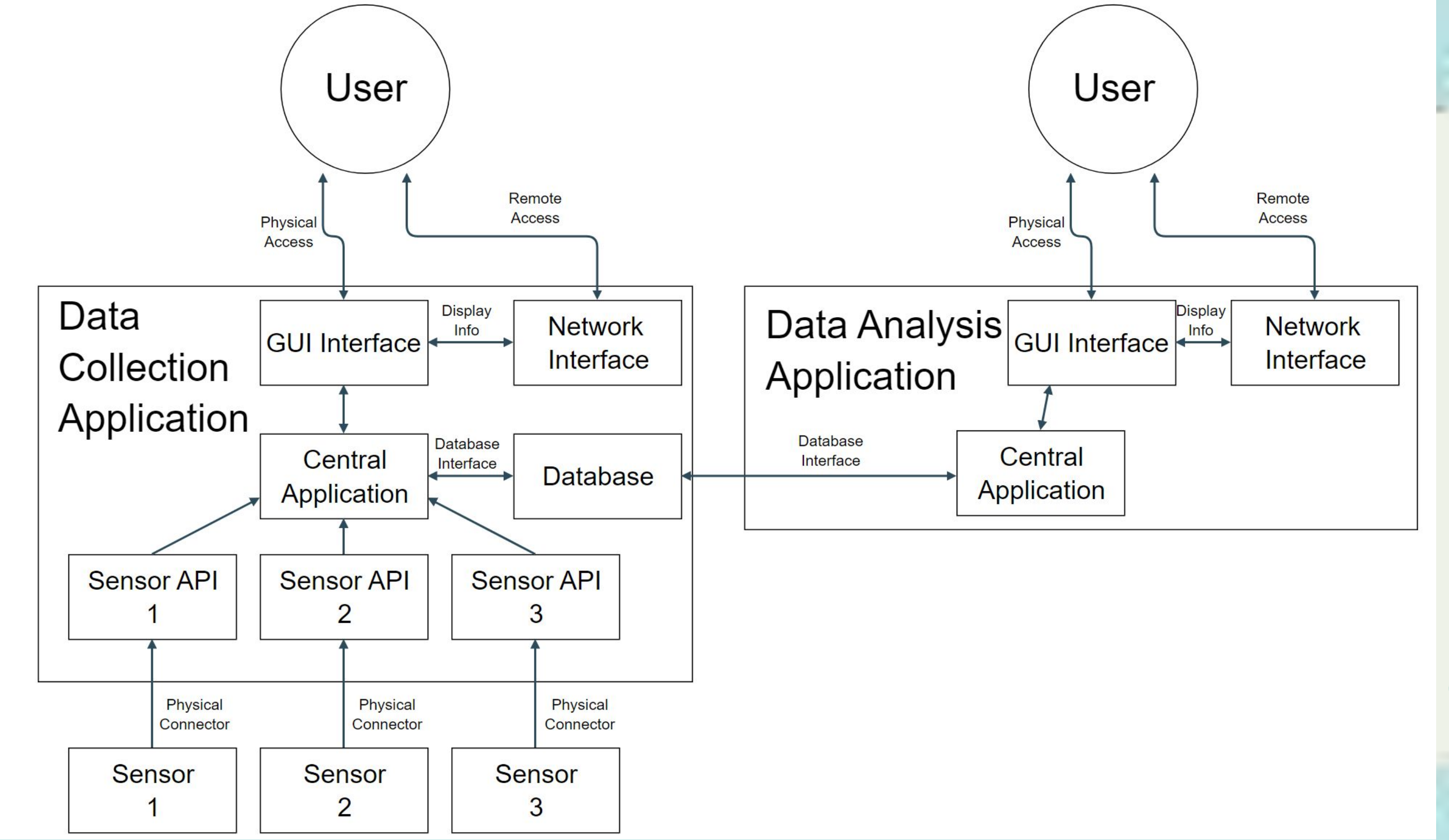


- Performance Requirements - ensure optimal user experience and efficiency
    - display data from the sensors soon after reading data
    - respond to user requests and data inputs quickly
- Safety Requirements - imperative to exercise caution around the computer, wires, and other equipment. (Aqualab = large tanks of water)
- Security Requirements - important for only registered users to have access to system, they are able to access only the features associated with their user role.
- Software Quality Attributes -
    - user-friendly user interface that is easy to navigate and intuitive.
    - system shall be scalable, reliable, and robust



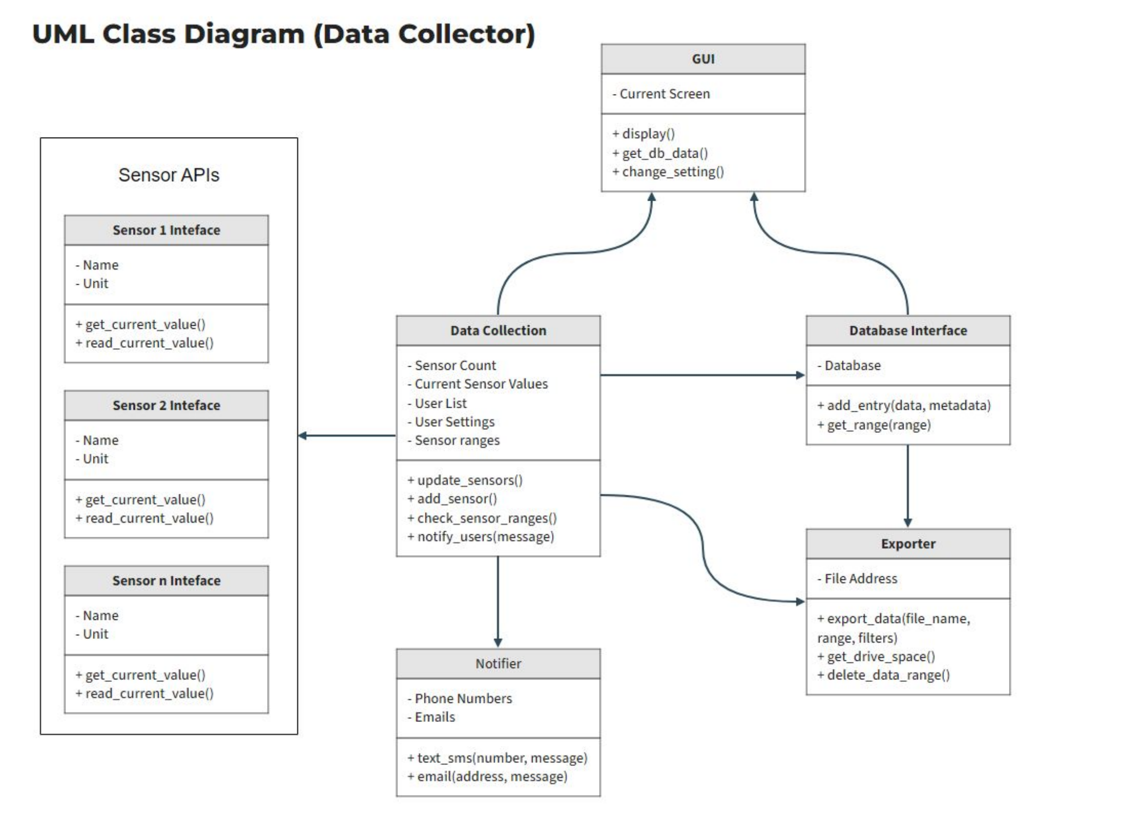
# Design

# Diagrams:

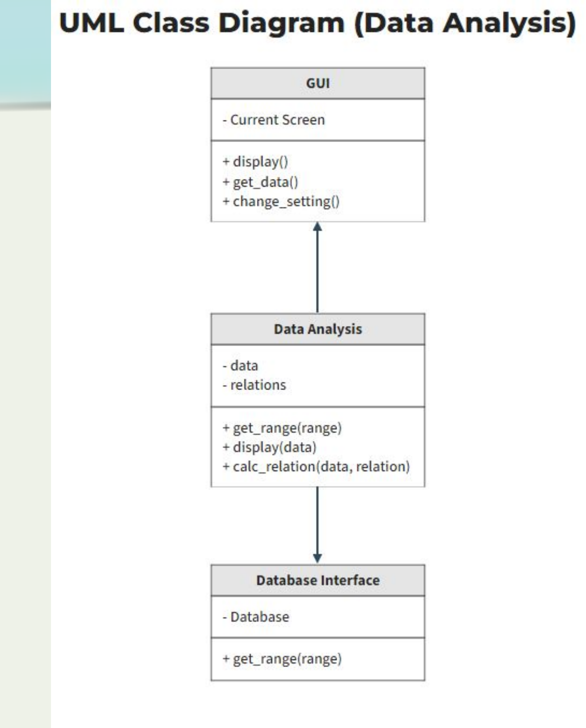


User
Physical Access
Remote Access
Data Collection Application
GUI Interface
Display Info
Network Interface
Central Application
Database Interface
Database
Sensor API 1
Sensor API 2
Sensor API 3
Physical Connector
Physical Connector
Physical Connector
Sensor 1
Sensor 2
Sensor 3
Database Interface
User
Physical Access
Remote Access
Data Analysis Application
GUI Interface
Display Info
Network Interface
Central Application

# Diagrams:

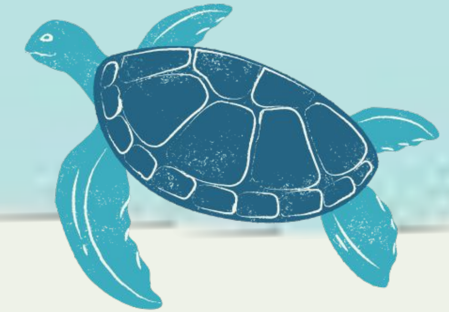


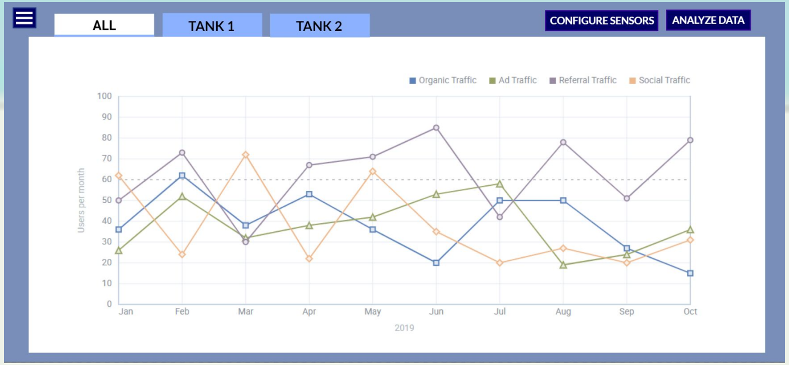
UML Class Diagram (Data Collector)

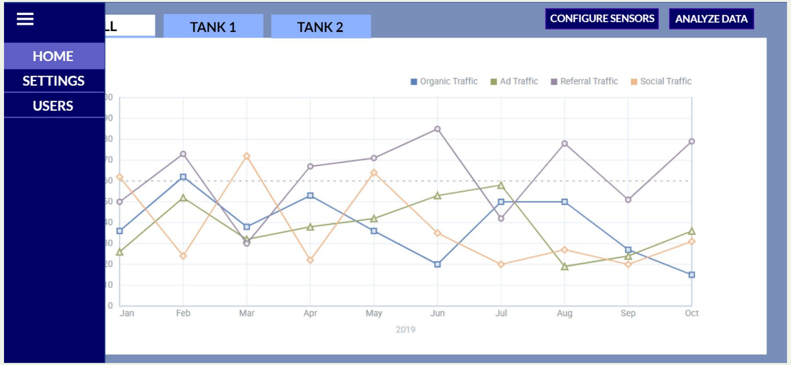


UML Class Diagram (Data Analysis)

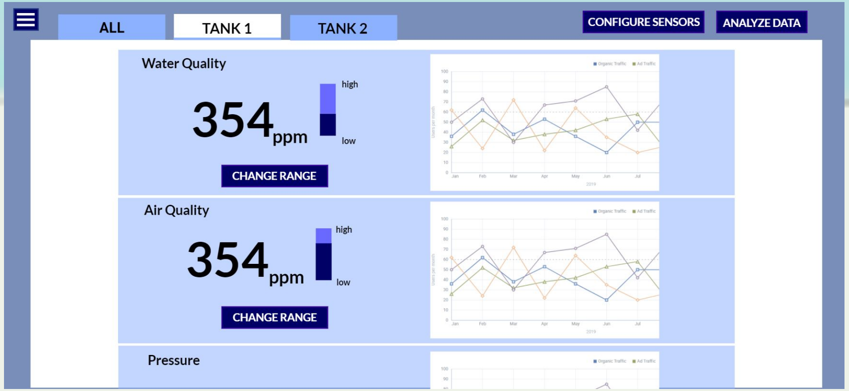
# UI Mockups: Home Page

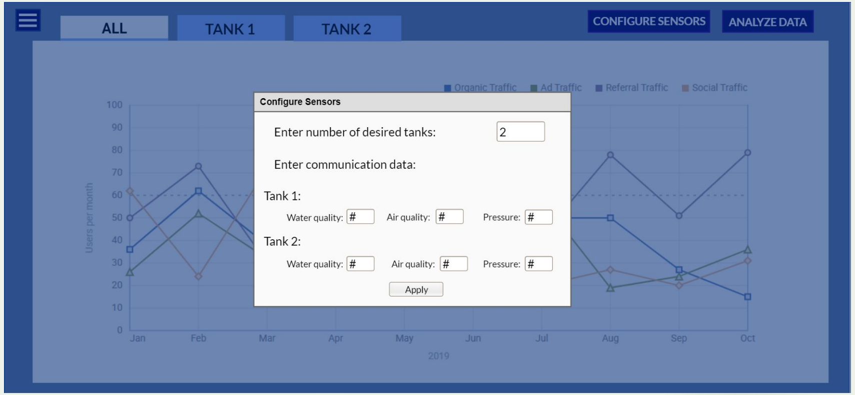






# UI Mockups: Sensor View





Configure Sensors

Enter number of desired tanks: 2

Enter communication data:

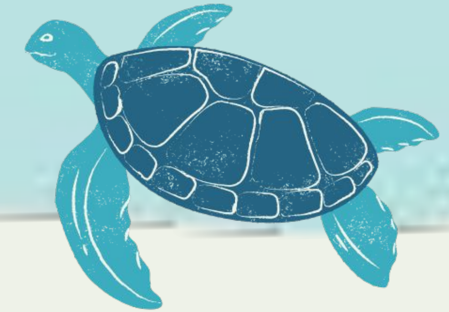
Tank 1:

Water quality: # Air quality: # Pressure: #

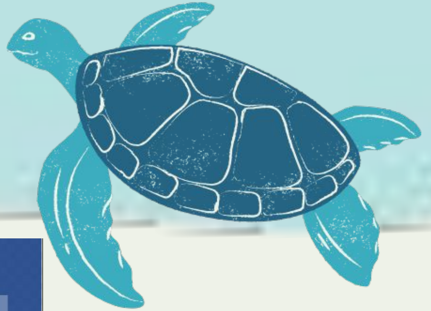
Tank 2:

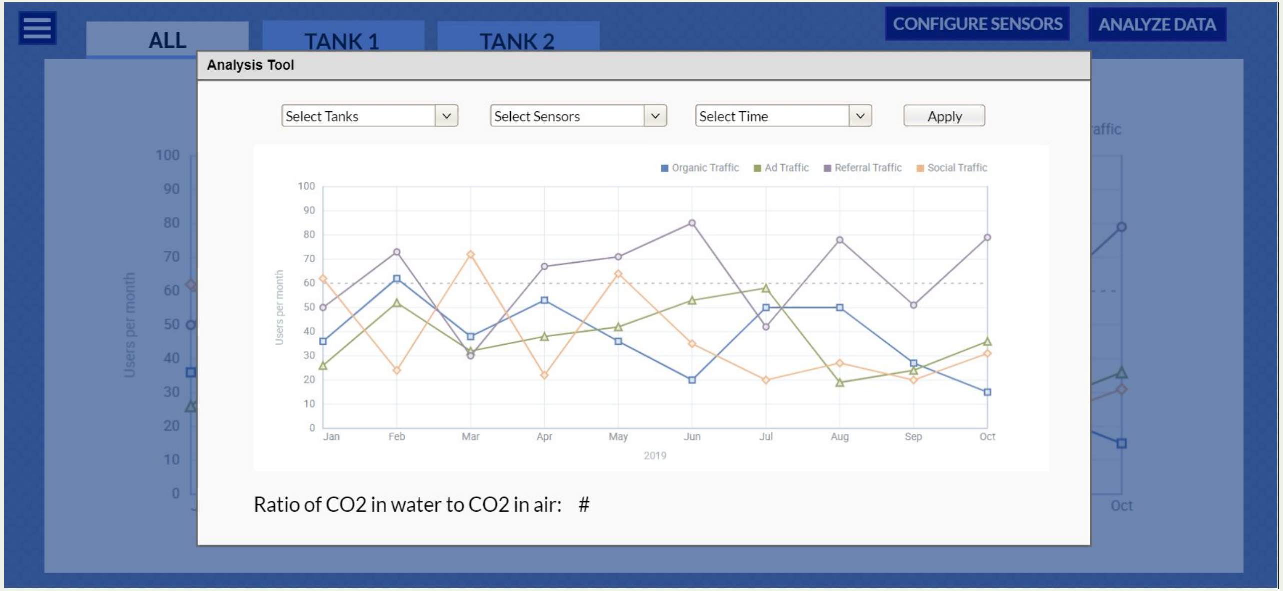
Water quality: # Air quality: # Pressure: #

Apply



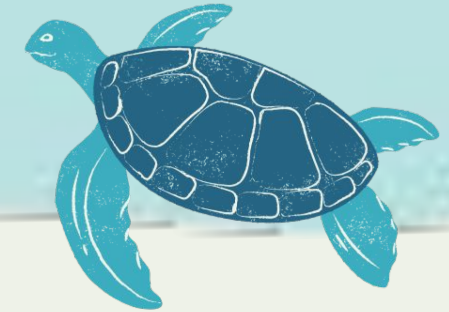
# UI Mockups: Analysis Tool

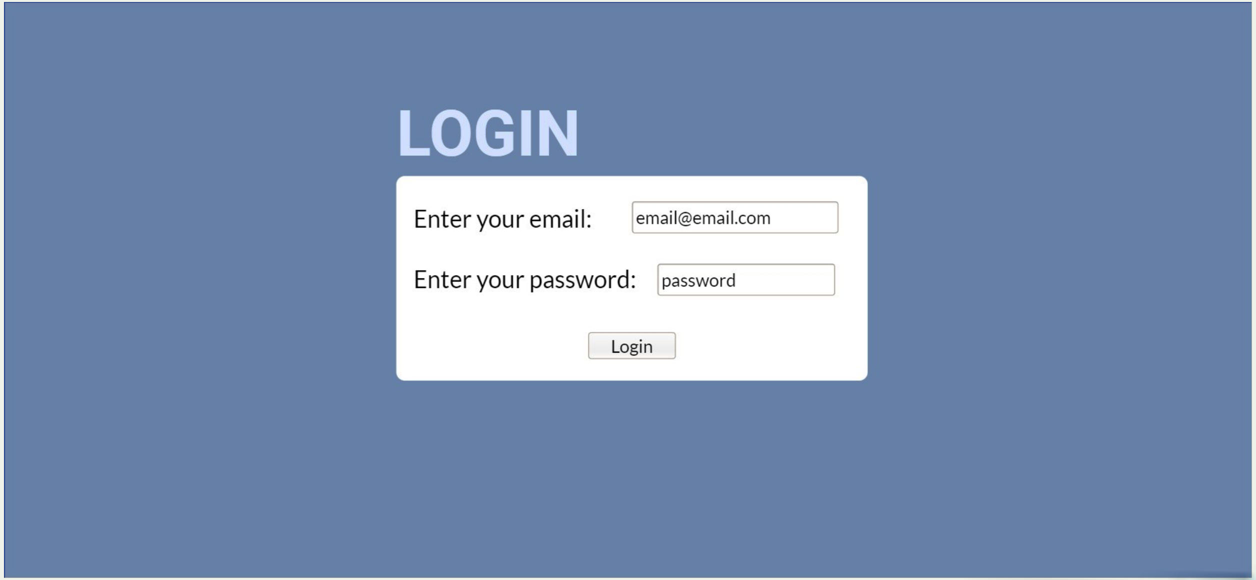




Ratio of CO2 in water to CO2 in air: #

# UI Mockups: Login Page



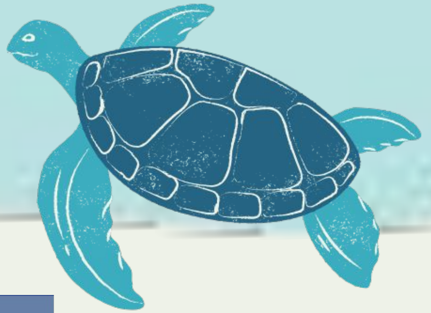


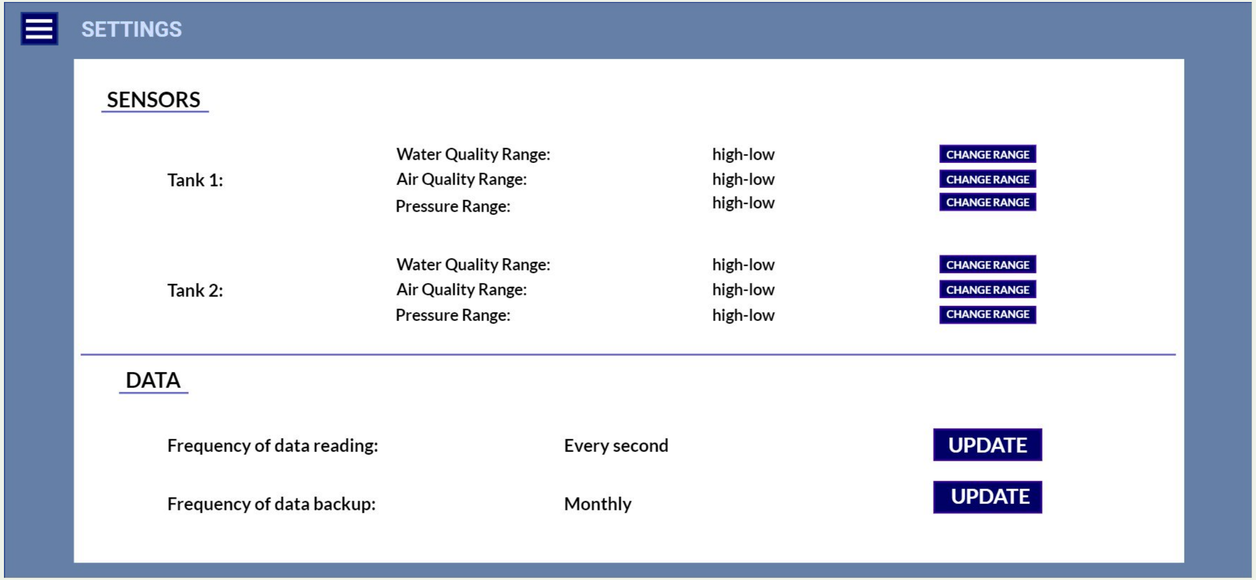
LOGIN

Enter your email: email@email.com

Enter your password: password

# UI Mockups: Settings





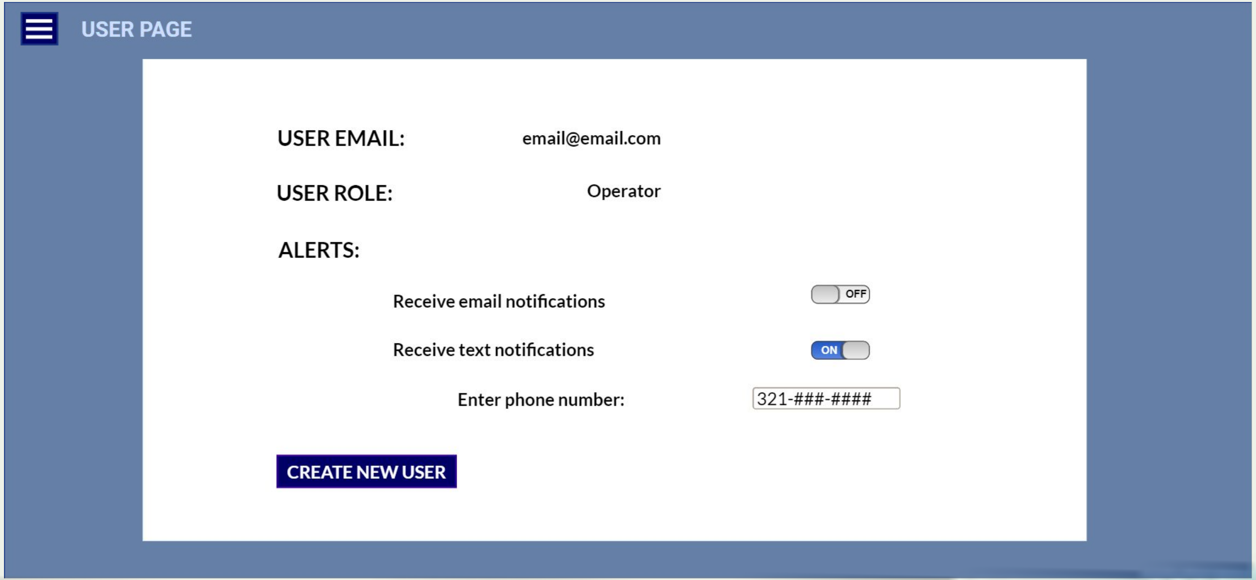
SENSORS

| | | | |
|---|---|---|---|
| Tank 1: | Water Quality Range: | high-low | CHANGE RANGE |
| | Air Quality Range: | high-low | CHANGE RANGE |
| | Pressure Range: | high-low | CHANGE RANGE |
| Tank 2: | Water Quality Range: | high-low | CHANGE RANGE |
| | Air Quality Range: | high-low | CHANGE RANGE |
| | Pressure Range: | high-low | CHANGE RANGE |

DATA

| | | |
|---|---|---|
| Frequency of data reading: | Every second | UPDATE |
| Frequency of data backup: | Monthly | UPDATE |

# UI Mockups: User Options



USER PAGE

USER EMAIL: email@email.com

USER ROLE: Operator

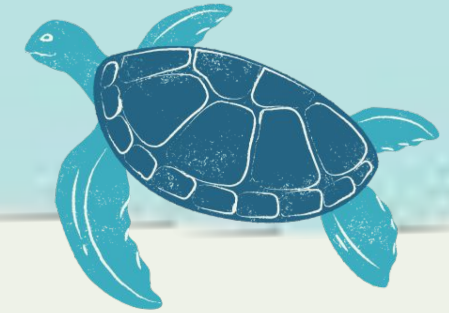
ALERTS:

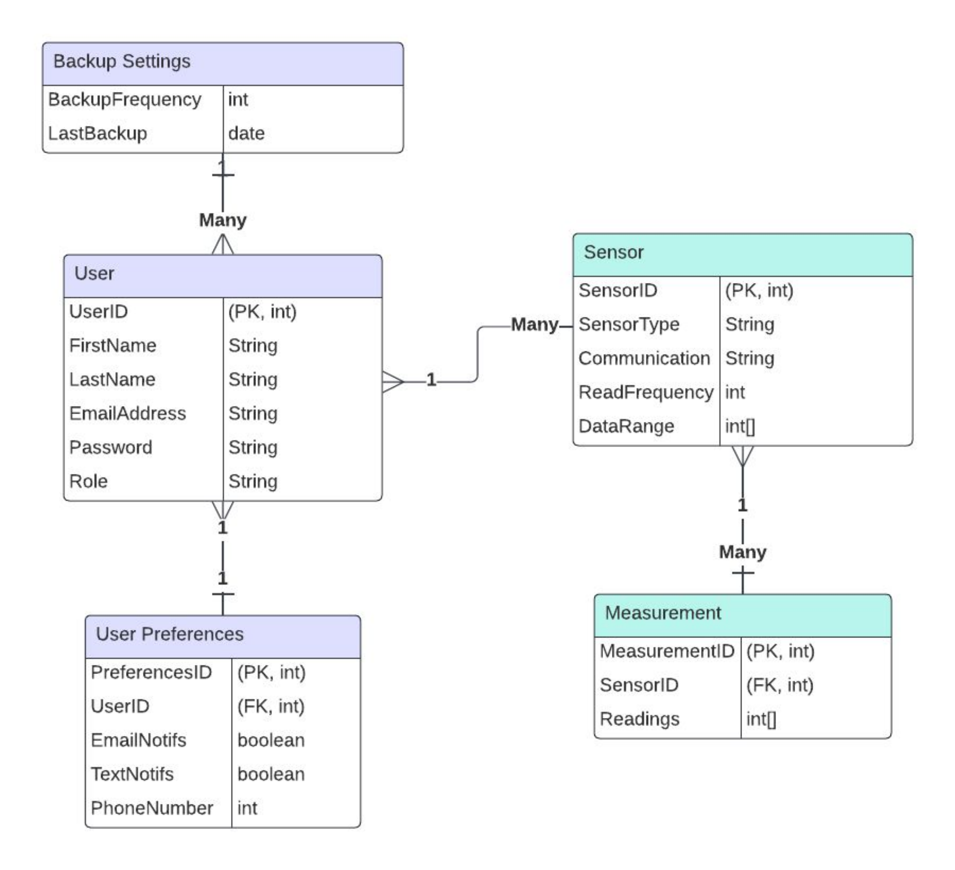
Receive email notifications: OFF

Receive text notifications: ON

Enter phone number: 321-###-####

# Database Design





**Backup Settings**

| | |
|---|---|
| BackupFrequency | int |
| LastBackup | date |

**User**

| | |
|---|---|
| UserID | (PK, int) |
| FirstName | String |
| LastName | String |
| EmailAddress | String |
| Password | String |
| Role | String |

**User Preferences**

| | |
|---|---|
| PreferencesID | (PK, int) |
| UserID | (FK, int) |
| EmailNotifs | boolean |
| TextNotifs | boolean |
| PhoneNumber | int |

**Sensor**

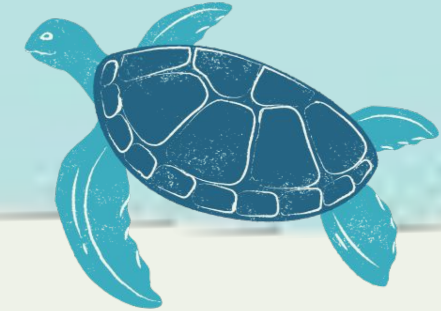
| | |
|---|---|
| SensorID | (PK, int) |
| SensorType | String |
| Communication | String |
| ReadFrequency | int |
| DataRange | int[] |

**Measurement**

| | |
|---|---|
| MeasurementID | (PK, int) |
| SensorID | (FK, int) |
| Readings | int[] |

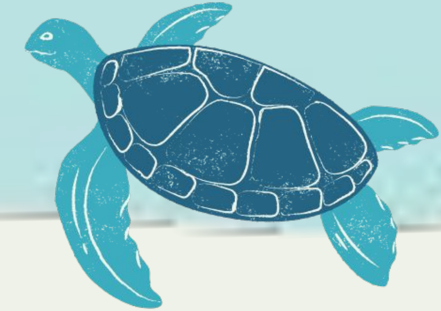
# Testing

# Test Levels



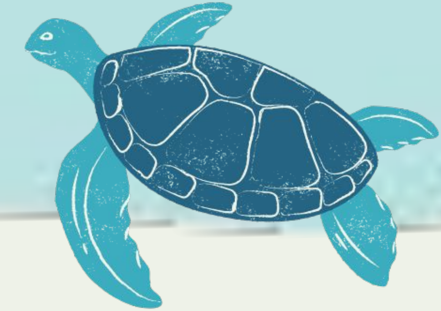
- Testing Levels:
  - Unit Testing: verifies individual components and modules
  - Integration Testing: Verifies interactions between modules
  - System testing: Verifies the system as a whole
  - Acceptance testing: Validates the system meets client expectations

# Testing Methods



- Testing Methods:
  - Manual Testing: necessary to validate UI, interaction, and user experience
  - Automated Testing: For frequent, repeatable test cases
- Types of Testing:
  - Functional testing: Will verify that all features work as required
  - Performance testing: Will evaluate system performance under load
  - Data Integrity Testing: Will ensure the accuracy and completeness of data collection and analysis.

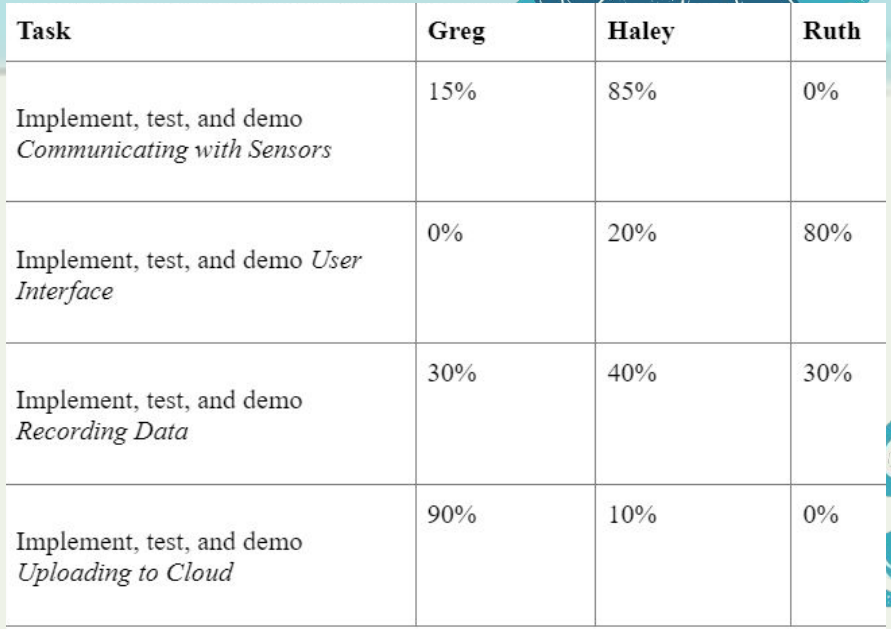
# Test Items



- User Management (User creation, Role assignment, and Permissions)
  - Admin, operator, and observer roles
- Sensor Connectivity
- Monitoring and Display of Sensor Data
- Analysis of Past Data
- Disk Overflow Mitigation
- User Input and Alert Management
- User Action Logging

# Milestone 2:

- Implement, test, and demo Communicating with Sensors
- Implement, test, and demo User Interface
- Implement, test, and demo Recording Data
- Implement, test, and dem Uploading to Cloud



| **Task** | **Greg** | **Haley** | **Ruth** |
|---|---|---|---|
| Implement, test, and demo *Communicating with Sensors* | 15% | 85% | 0% |
| Implement, test, and demo *User Interface* | 0% | 20% | 80% |
| Implement, test, and demo *Recording Data* | 30% | 40% | 30% |
| Implement, test, and demo *Uploading to Cloud* | 90% | 10% | 0% |

# Questions?