

Weather Monitoring System

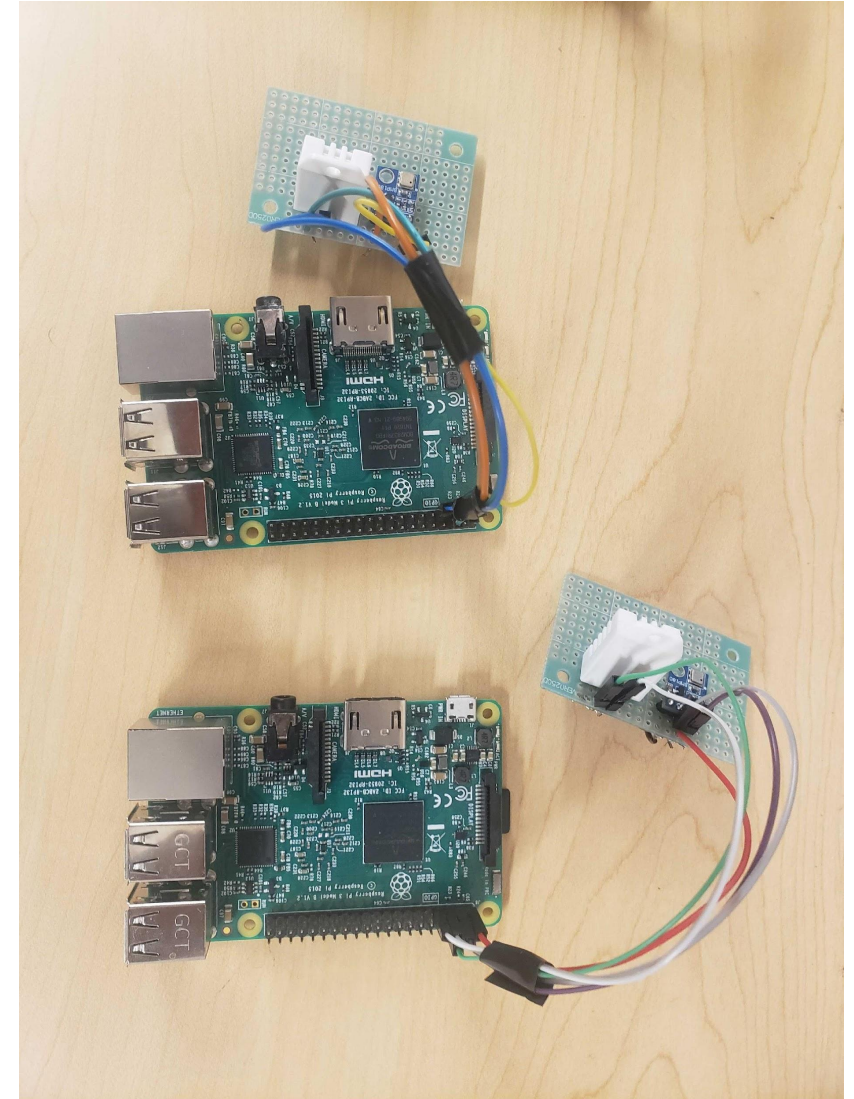
– Project 6 Update

Sagar Pathare and Vimal Kakaraparthi

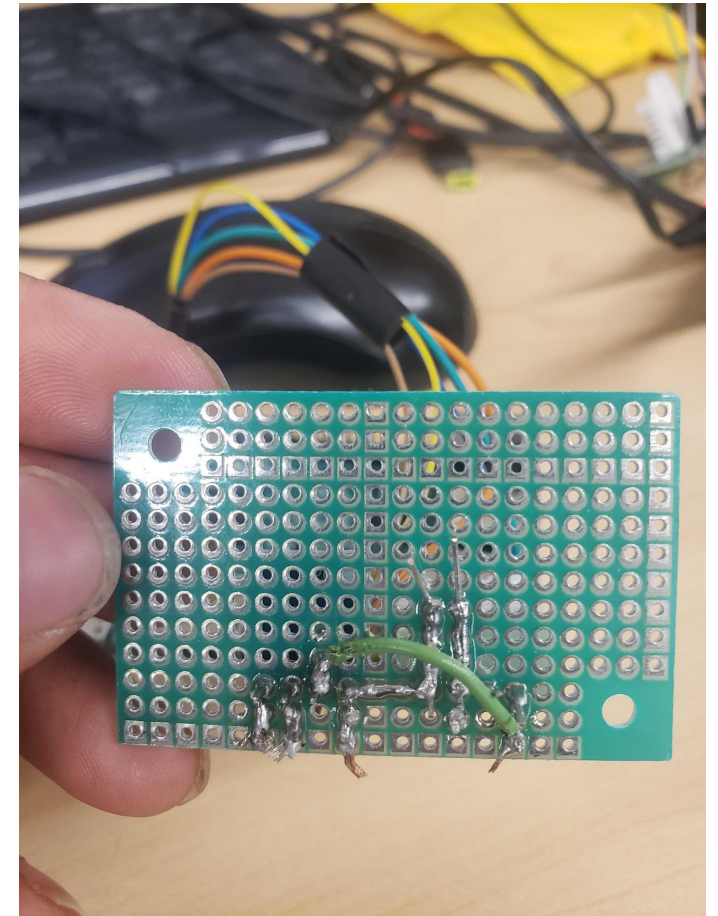
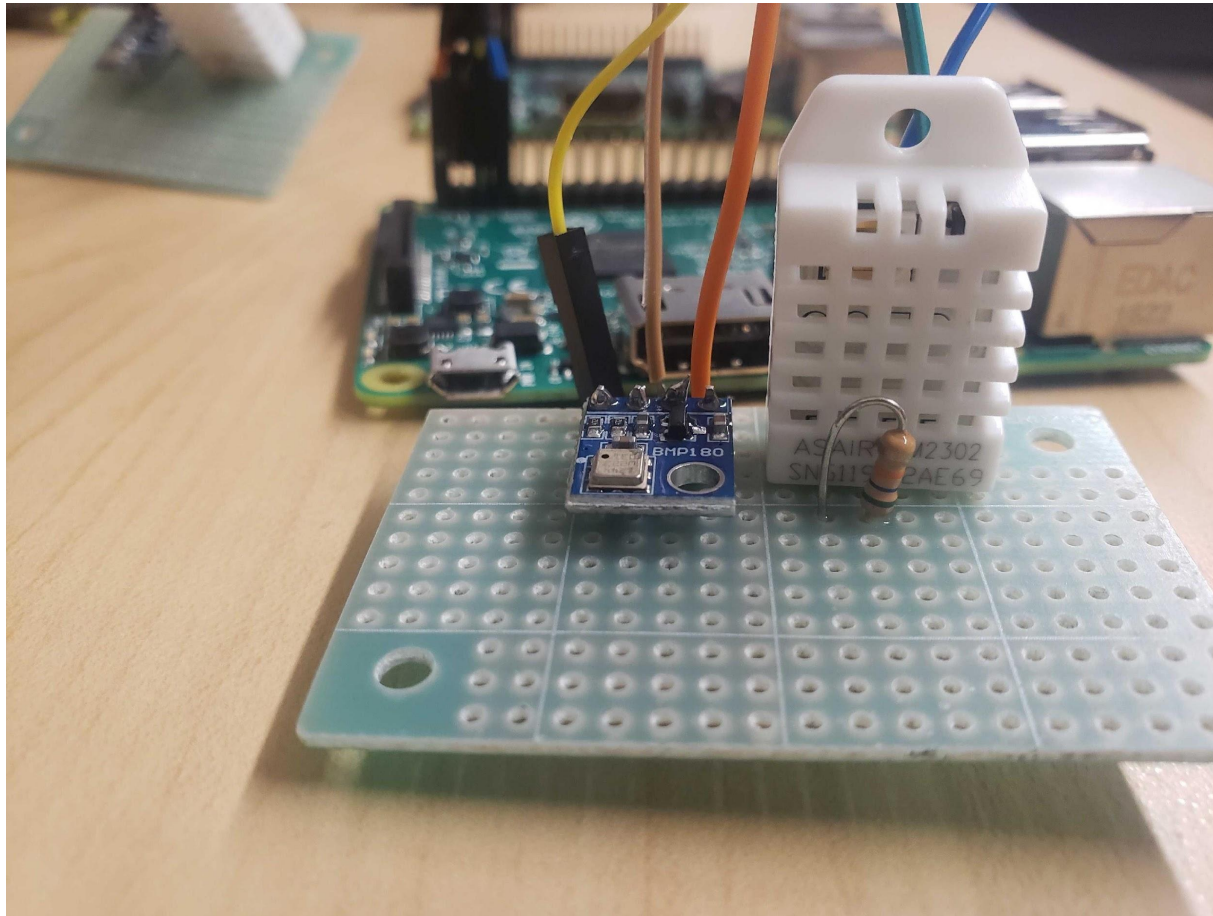
Status Summary

Work Done (Hardware)-

- Vimal implemented the hardware which included setting up the Raspberry Pis, configuring and testing the sensors (BMP 180, DHT 22), soldering the sensors onto a compact breadboard and making the weather stations report the necessary data.
- We are implementing four weather stations. All of them are up and running with a basic code that prints the sensors' output (temperature, humidity, pressure, altitude) to the console.

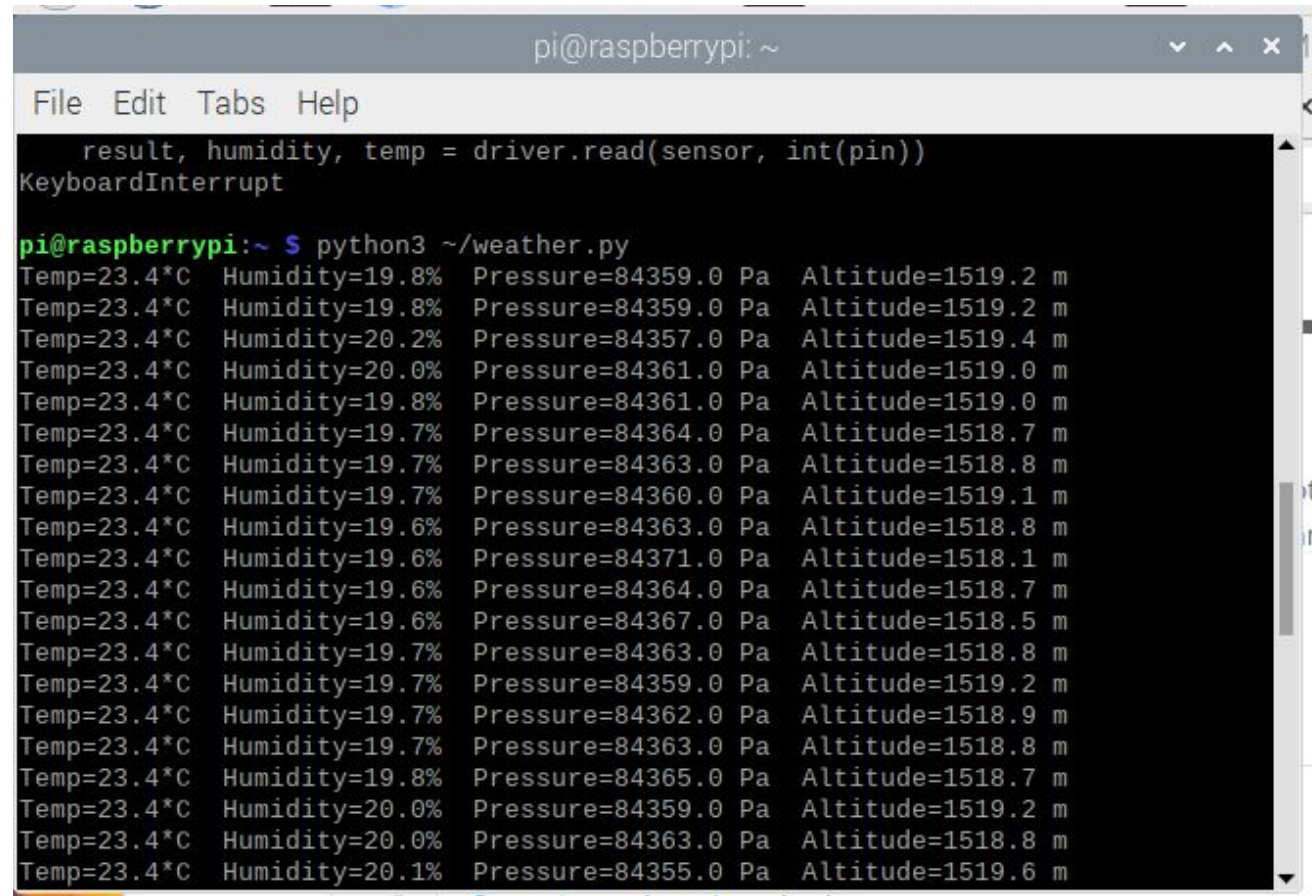


Work Done (Hardware)-



Sensors compactly soldered to a breadboard

Work Done (Hardware)-



The image shows a terminal window titled 'pi@raspberrypi: ~'. The window has a menu bar with 'File', 'Edit', 'Tabs', and 'Help'. The terminal content shows a Python script being executed, which reads sensor data and prints it to the console. The output consists of 20 lines of data, each containing temperature, humidity, pressure, and altitude.

```
pi@raspberrypi: ~  
File Edit Tabs Help  
    result, humidity, temp = driver.read(sensor, int(pin))  
KeyboardInterrupt  
pi@raspberrypi:~ $ python3 ~/weather.py  
Temp=23.4°C Humidity=19.8% Pressure=84359.0 Pa Altitude=1519.2 m  
Temp=23.4°C Humidity=19.8% Pressure=84359.0 Pa Altitude=1519.2 m  
Temp=23.4°C Humidity=20.2% Pressure=84357.0 Pa Altitude=1519.4 m  
Temp=23.4°C Humidity=20.0% Pressure=84361.0 Pa Altitude=1519.0 m  
Temp=23.4°C Humidity=19.8% Pressure=84361.0 Pa Altitude=1519.0 m  
Temp=23.4°C Humidity=19.7% Pressure=84364.0 Pa Altitude=1518.7 m  
Temp=23.4°C Humidity=19.7% Pressure=84363.0 Pa Altitude=1518.8 m  
Temp=23.4°C Humidity=19.7% Pressure=84360.0 Pa Altitude=1519.1 m  
Temp=23.4°C Humidity=19.6% Pressure=84363.0 Pa Altitude=1518.8 m  
Temp=23.4°C Humidity=19.6% Pressure=84371.0 Pa Altitude=1518.1 m  
Temp=23.4°C Humidity=19.6% Pressure=84364.0 Pa Altitude=1518.7 m  
Temp=23.4°C Humidity=19.6% Pressure=84367.0 Pa Altitude=1518.5 m  
Temp=23.4°C Humidity=19.7% Pressure=84363.0 Pa Altitude=1518.8 m  
Temp=23.4°C Humidity=19.7% Pressure=84359.0 Pa Altitude=1519.2 m  
Temp=23.4°C Humidity=19.7% Pressure=84362.0 Pa Altitude=1518.9 m  
Temp=23.4°C Humidity=19.7% Pressure=84363.0 Pa Altitude=1518.8 m  
Temp=23.4°C Humidity=19.8% Pressure=84365.0 Pa Altitude=1518.7 m  
Temp=23.4°C Humidity=20.0% Pressure=84359.0 Pa Altitude=1519.2 m  
Temp=23.4°C Humidity=20.0% Pressure=84363.0 Pa Altitude=1518.8 m  
Temp=23.4°C Humidity=20.1% Pressure=84355.0 Pa Altitude=1519.6 m
```

Raspberry Pi Console output

Work Done (Software)-

- Sagar worked on the software part of the project. He set up a node backend web server connected to a PostgreSQL database with an MVC pattern. He also deployed it on Heroku and set up the auto-deployment pipeline.
- He implemented the following pages, including the frontend UI and the backend logic- registration, login, dashboard, update profile, update preferences.
- He also implemented an API that the weather station will call to update the weather values in the database. Here's the link for the project- <https://ooad-sv-wms.herokuapp.com/>

Work Done (Software)-

Weather Monitoring System

Login Register

Login

Email

Password

Submit

[I don't have an account](#)

Login page

Work Done (Software)-

Weather Monitoring System

Login Register

Register

First name

Sampath

Last name

Kumar

Email

sampath.kumar@me.com

Password

.....

Confirm password

.....|

Submit

User registration page

Work Done (Software)-

Weather Monitoring System

[Dashboard](#) [Preferences](#) [Profile](#)

Logged in as Vimal [Logout](#)

Preferences

Preferences updated successfully!

Weather Station Subscriptions:

☒ Weather Station 1

☐ Weather Station 2

☐ Weather Station 3

☒ Weather Station 4

Notifications:

☒ Receive interval notifications

Time interval

4 hours

☐ Receive alarm notifications

Minimum Temperature

0.75

Maximum Temperature

1.75

Minimum Pressure

0.75

Maximum Pressure

1.75

Minimum Humidity

0.75

Maximum Humidity

1.75

Minimum Altitude

0.75

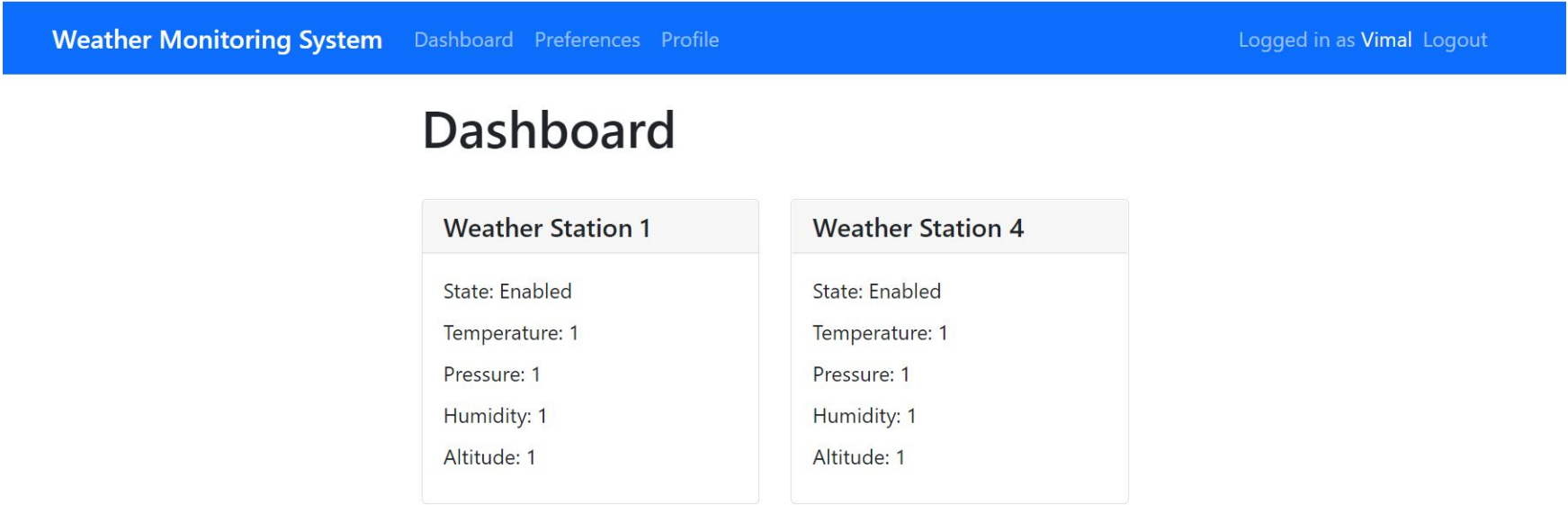
Maximum Altitude

1.75

Submit

User Preferences page

Work Done (Software)-



User Dashboard

Changes or Issues Encountered

- Originally, we were going to have a single registration page where the user enters all the registration details and the preferences data.
- But we realized that if the registration details have an error like invalid email or password, then the preferences data entered by the user will be a waste of time as they would have to fill it all over again.
- So, we simplified the registration page only to include the registration details. Once the user is registered, after logging in, if they have not set the preferences, they are automatically taken to the preferences page first instead of the dashboard.

(Recap- Patterns from Project 5)

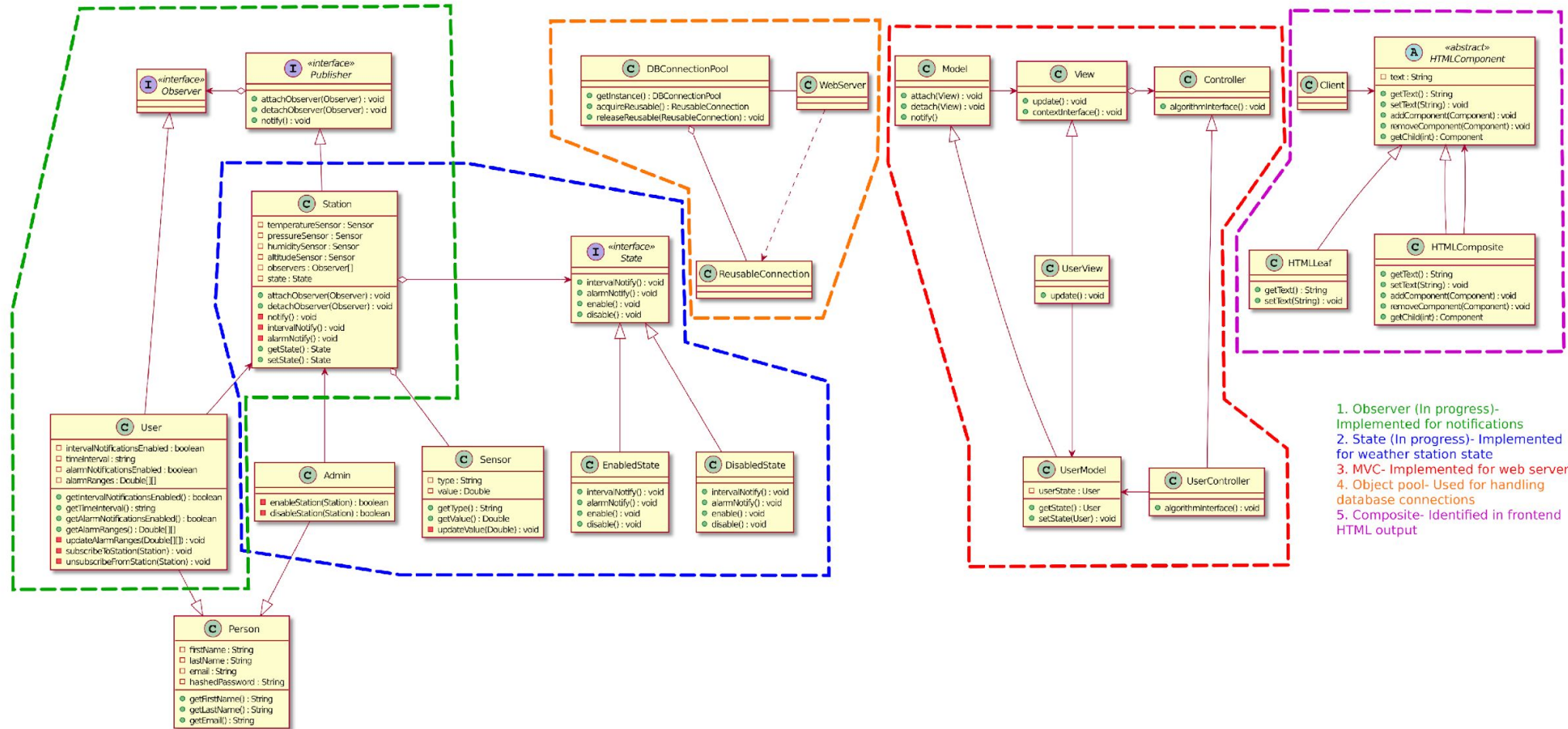
- Observer: The User class implements the Observer interface. The Station class implements the Publisher interface and stores references to the Observer interface and implements attachObserver, detachObserver and notify methods. The station class has two other notify methods named intervalNotify and alarmNotify, which internally call the same notify method.
- State: The Station class stores a reference to the State interface which stores its state. Concrete classes EnabledState and DisabledState implement the State interface and override the intervalNotify, alarmNotify, enable and disable methods.
- Object Pool: We are using Node.js for the web server and PostgreSQL for the database. The WebServer will get an instance of the DBConnectionPool using the getInstance method. It will get an instance of ReusableConnection using the acquireReusable method and later release it using the releaseReusable method.
- MVC: We are using Node.js for the web server along with the Express.js framework. We're applying the MVC pattern where user interacts with the View, Controller changes the Model state and renders the View and Model updates the View. For example, a webpage for displaying user information will involve UserModel to store the user information, UserView to display it and UserController to update the state and render the UserView.
- Composite: We are using HTML to display the web UI. The DOM tree in HTML is a good example of the Composite pattern. Each element in the DOM tree implements the HTMLComponent interface. An element can either be an HTMLComposite, which further contains more elements, or it could be an HTMLLeaf, the leaf of that tree node. Both HTMLLeaf and HTMLComposite implement the HTMLComponent interface.

Patterns

- So far, for the backend web server, we have implemented the MVC pattern. It has really helped us in 'Separation of Concerns'- when any changes are made to any one of the model/view/controller, the other two remain agnostic of the change. For example, any changes to the database design are only reflected in the models and the view and controller are agnostic of those changes.
- We have used the Object pool for handling the database connections and identified the Composite pattern in the frontend HTML output.
- We are currently working on the Observer pattern and the State pattern- the classes have been created with the variables, but the methods are yet to be implemented.

Class Diagram

Implementation progress- annotated version of class diagram from Project 5 (please find the full size image in the repo)



Plan for Next Iteration (Hardware)-

- Make the Raspberry Pi weather stations report to the webserver
- Add more weather stations if possible
- Make an enclosure to contain the weather station if possible

Plan for Next Iteration (Software)-

We need to implement the following functionalities:

- Role-based different UI for user and admin
- Allowing admin to update the state of the weather station
- Setting up the email notification service
- Setting up the cron job for sending the interval notifications to users
- Sending the alarm notifications to users

If we finish all these functionalities before time, we plan to extend the scope of the original project with the following additional functionalities-

- Showing a graph of different weather values
- Sending the notifications about legally critical weather values to admin so that they could inform the local authorities