COM S 319

Construction of User Interfaces

Final Project Documentation for Team 47

Prepared for

Dr. Abraham Aldaco

Prepared by

Randy Nguyen

rvnguyen@iastate.edu

Matthew Duncan

mlduncan@iastate.edu

May 10th, 2023

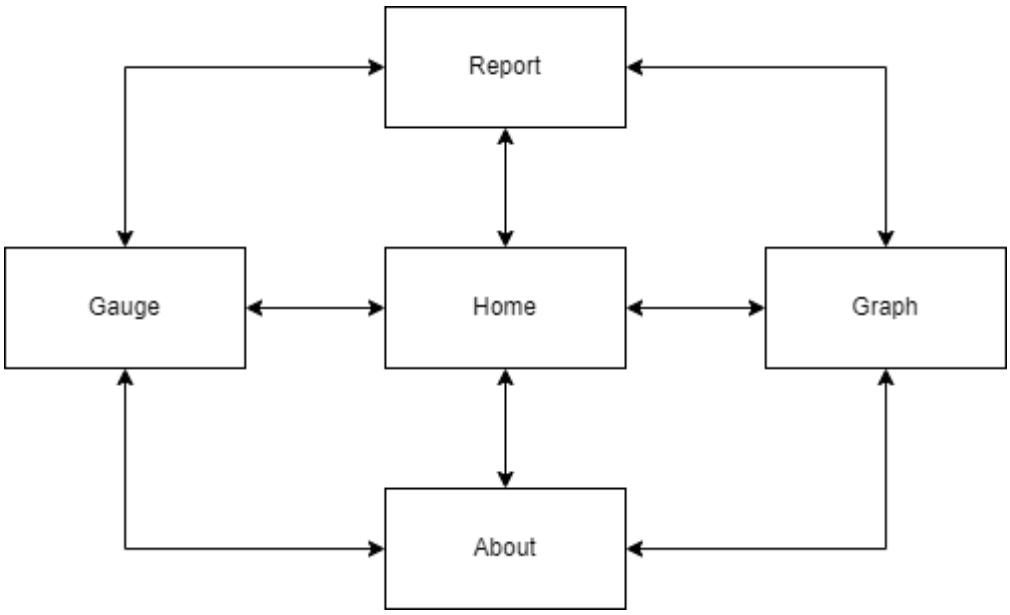
Table of Contents

Description

We have had a previous interest in Raspberry Pis and we thought this would be a great opportunity to learn more about it while learning what this course is originally about: the construction of user interfaces. Creating a website using the data collected directly by us feels more rewarding than using data from outside sources.

With that in mind, our final project is a web application that displays the temperature and humidity of a room. We utilized a Raspberry Pi and a DHT11 to collect the data, and used React, HTML, CSS, and JavaScript/Express, and MongoDB to create the web application. There are many ways to view this information within our web application such as simply viewing the numbers, analyzing through a graph, or looking at a gauge. There is also a feature to save current temperatures with timestamps, so the user can keep information about previous times. They also have the ability to add a note to the record. With the multitude of ways to view the information in an interesting way, we believe we have created a great experience for the user.

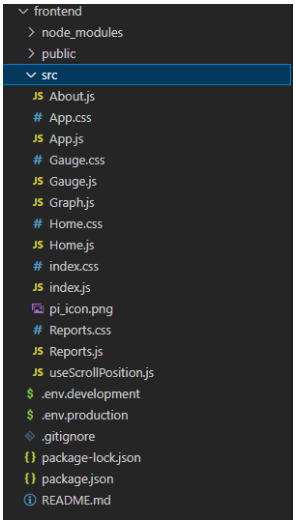
We have a simple web application with only 5 different views. We believe that when trying to obtain information about temperature and humidity complicated views and flows will hinder the user experience. All of these views will be accessible to each other by the navbar.


Report
Gauge
Home
Graph
About

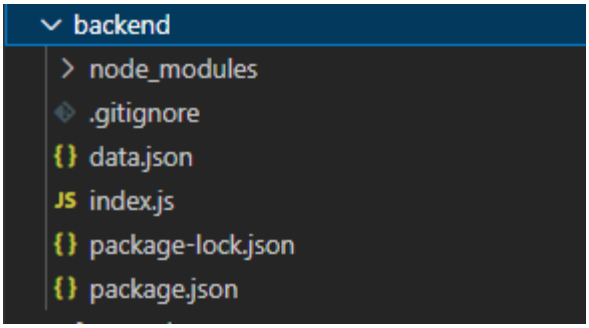
File and Directory Architecture

Our project is split up into two different folders. The backend and the frontend.

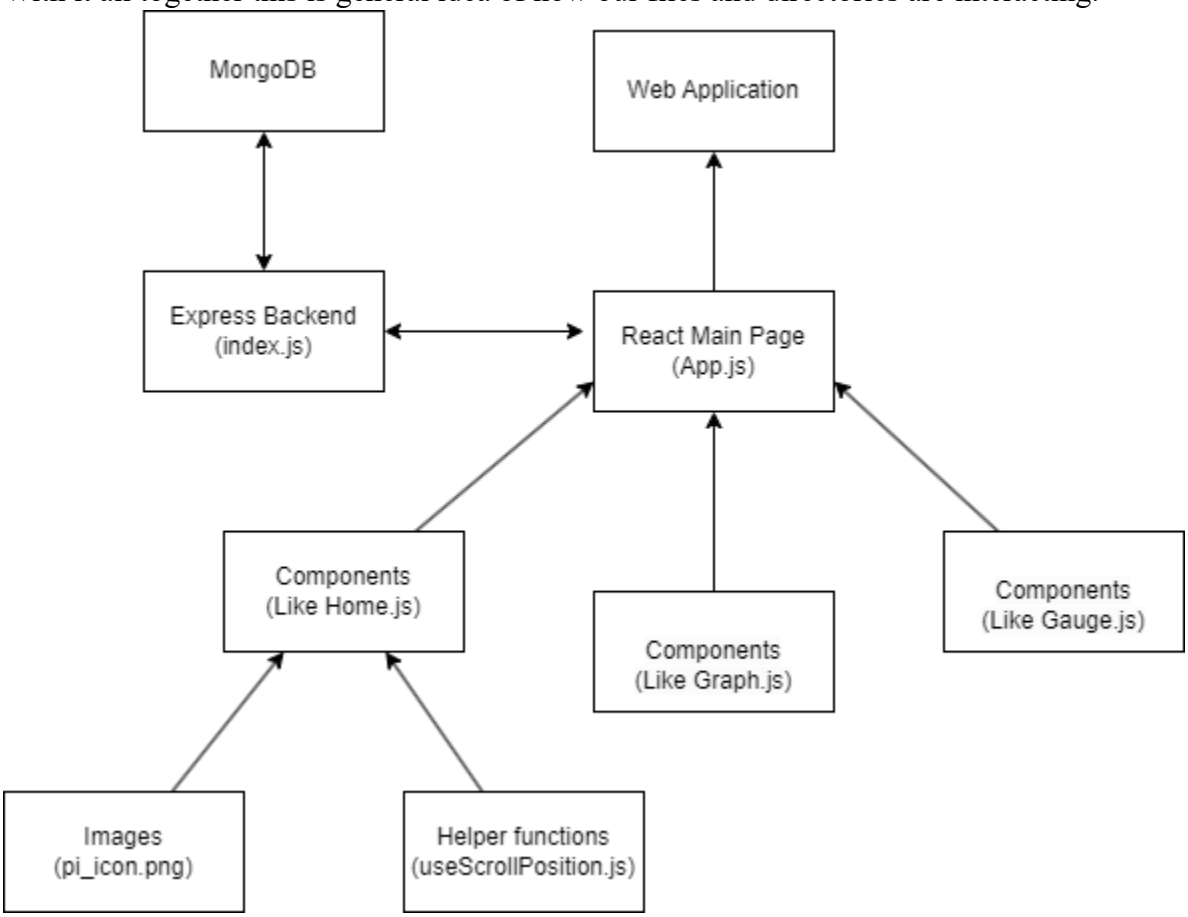
The frontend is a React Project that utilizes multiple components for our single view project.



The backend is a simple express project that holds all the endpoints needed for the frontend and the MongoDB.

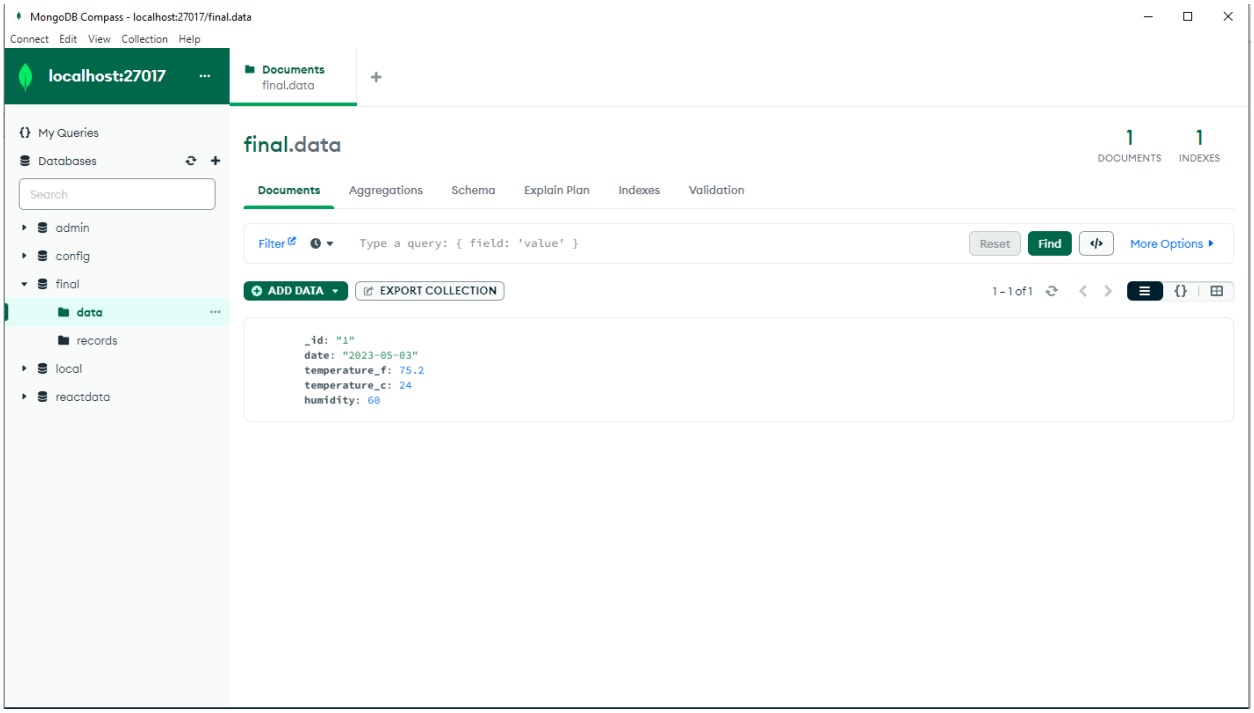


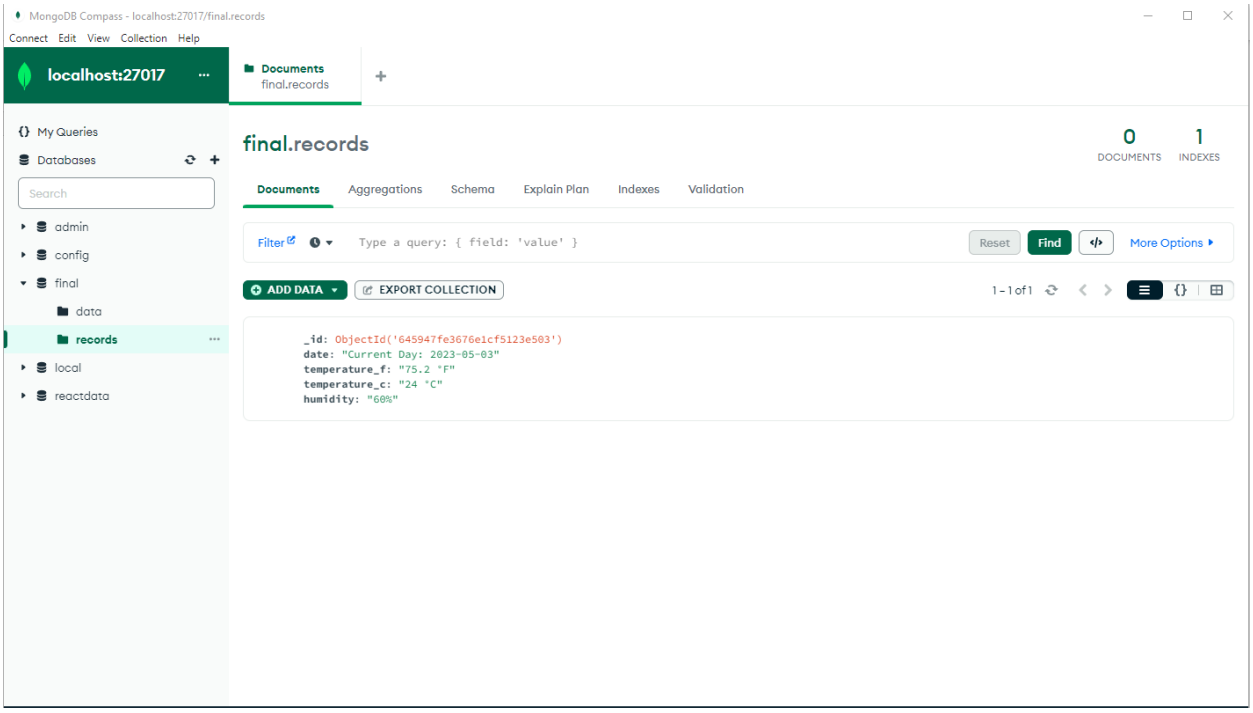
With it all together this is general idea of how our files and directories are interacting.



Server Architecture

The server application used in our project is MongoDB. It's a simple and great application to hold information about temperature and humidity for our project. We have a simple database "final" with 2 collections "data" and "records". The data collection holds one json object which is the current temperature and humidity which would be consistently updated by the Raspberry Pi. The records collection holds the current saved information that the user saved.

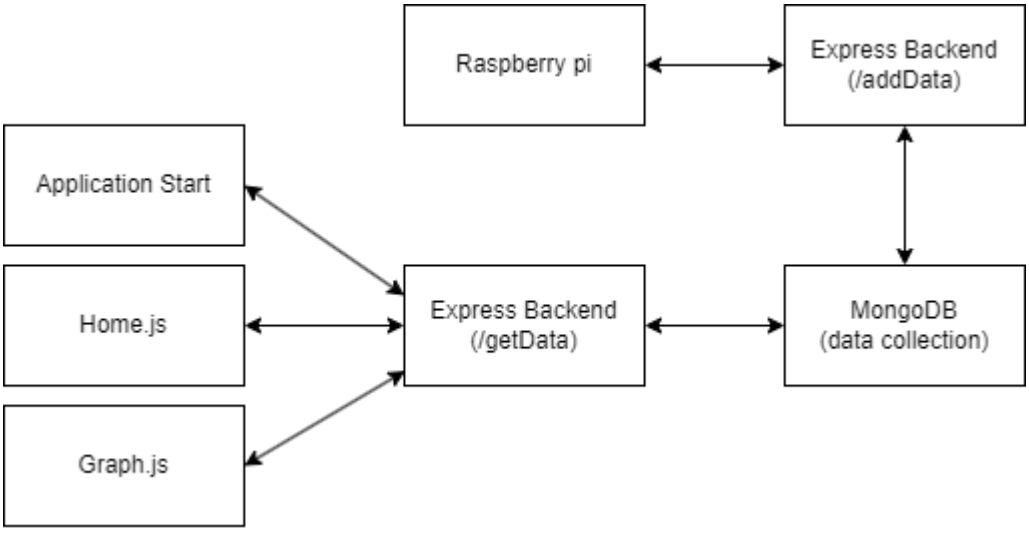



MongoDB Compass - localhost:27017/final.records
Connect Edit View Collection Help
localhost:27017
Documents
final.records
My Queries
Databases
Search
admin
config
final
data
records
local
reactdata
final.records
0
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
Filter
Type a query: { field: 'value' }
Reset
Find
More Options
ADD DATA
EXPORT COLLECTION
1 - 1 of 1
_id: ObjectId('645947fe3676e1cf5123e503')
date: "Current Day: 2023-05-03"
temperature_f: "75.2 °F"
temperature_c: "24 °C"
humidity: "60%"

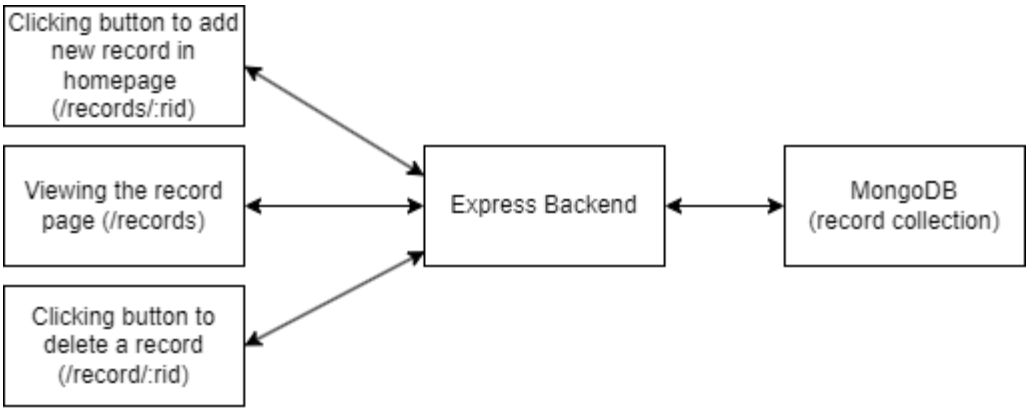
Logical Architecture

There are many ways that the frontend and raspberry pi interacts with the backend to change information on the database.

At the start the application sends a get request (/getData) to the data collection to get the current temperature and humidity. The raspberry pi would send a post request (/addData) to update the current temperature and humidity.

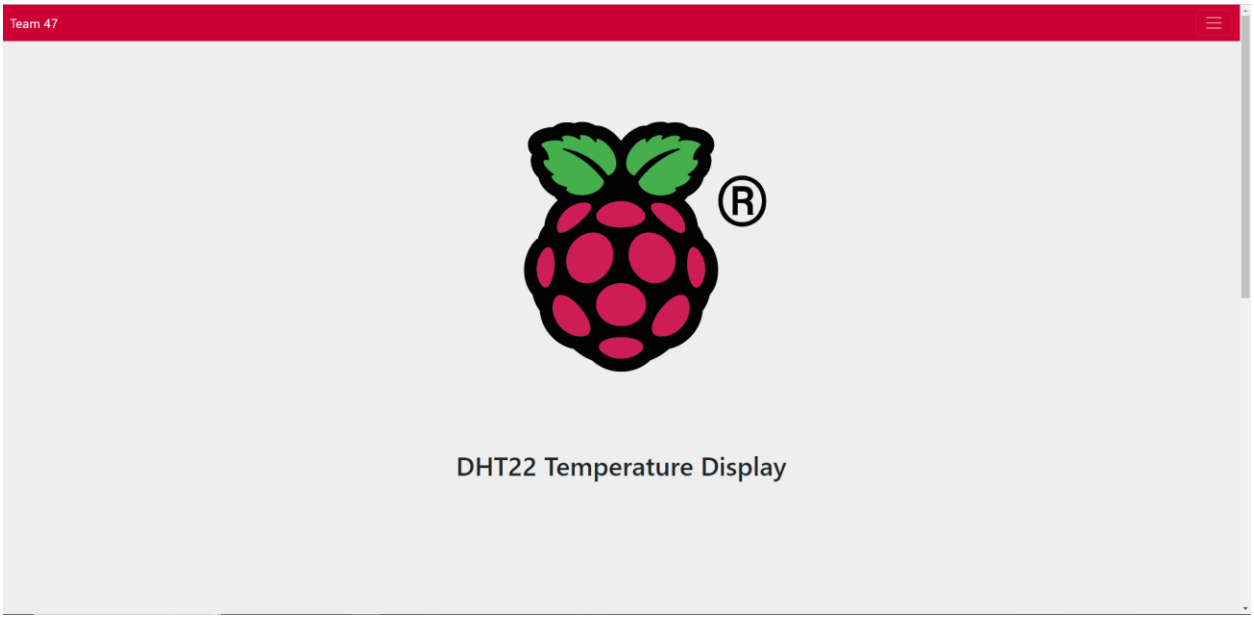


In the frontend the user can also interact with the backend/database by using the reports. In the home page the user can add a new report to the report. Within the reports view the user can update and delete records. This flow sends multiple types of request to the backend such as get (/records), post (/record/), put (/records/:rid), delete (/record/:rid).

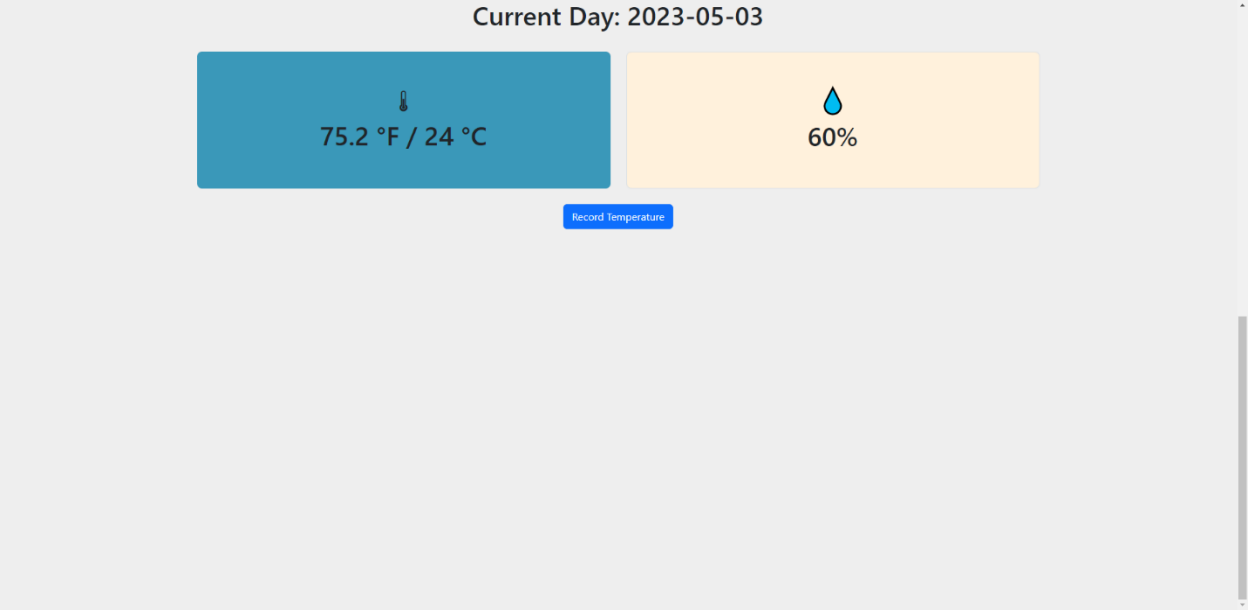

Clicking button to add new record in homepage (/records/:rid)
Viewing the record page (/records)
Clicking button to delete a record (/record/:rid)
Express Backend
MongoDB (record collection)

Views

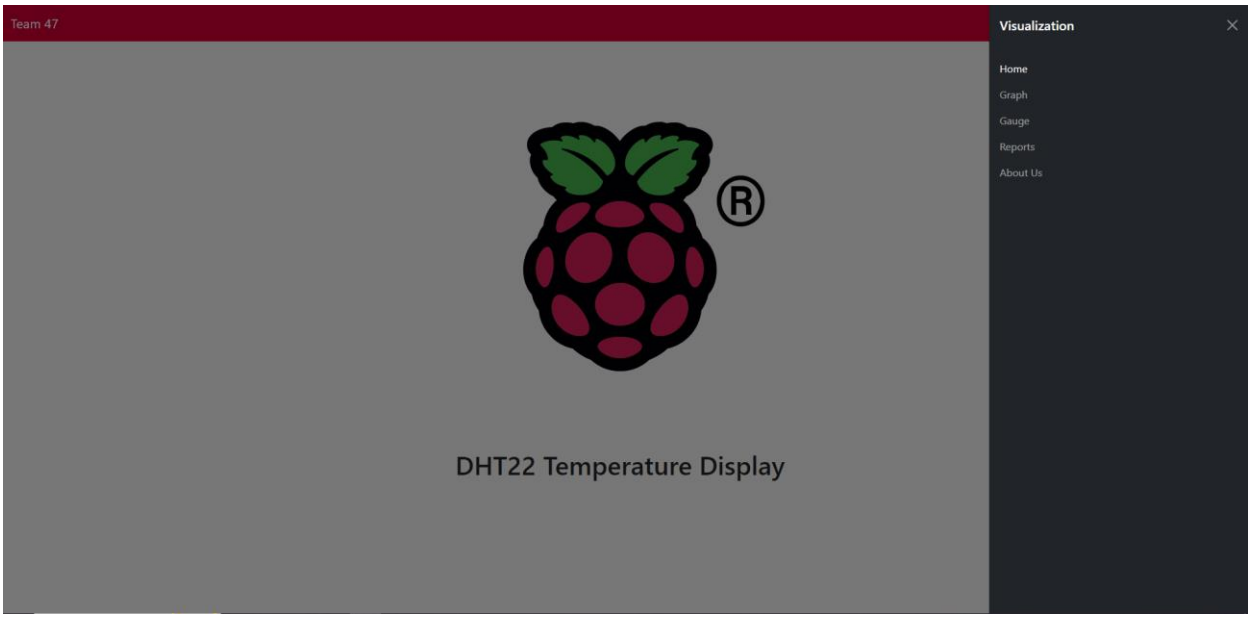
The homepage will greet the user with a title screen and on the same page the user will be able to access the temperature and humidity such as after or scrolling down the webpage. As stated before, when trying to obtain temperature and humidity it should be as seamless as possible, so adding it to the homepage will allow the user to quickly access that information, and if they would like to visualize the data in any other way such as a graph or a gauge, they can use the navbar to access those views.



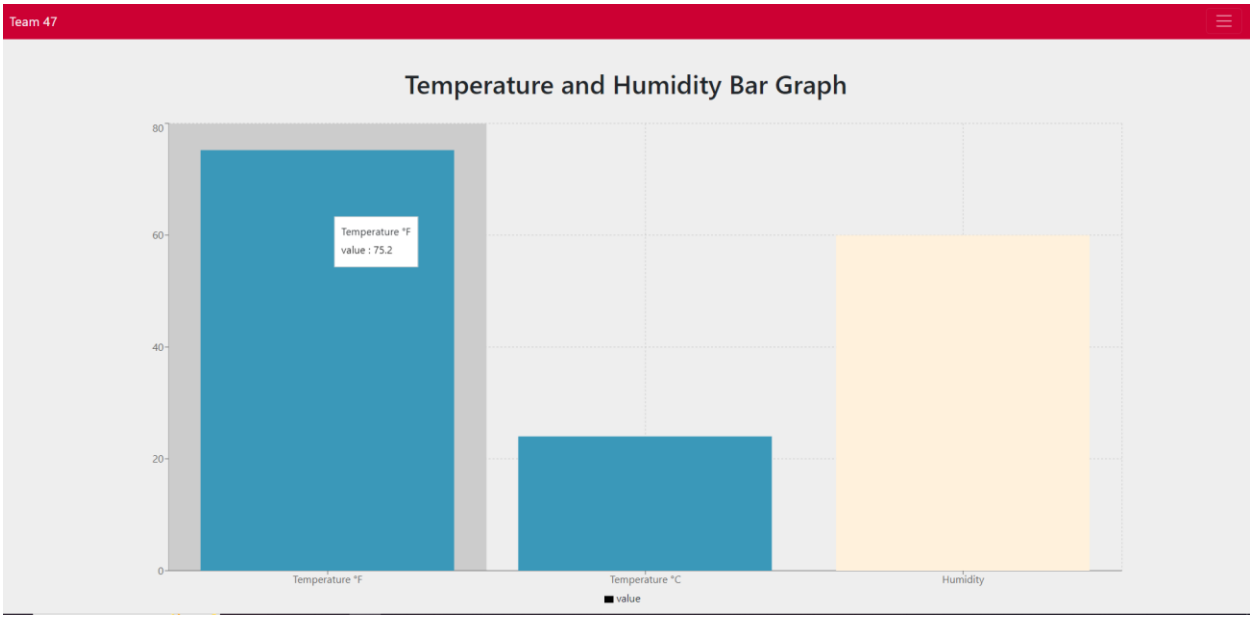
This is the home view that the user first sees when they open the web application. They can scroll down to view the temperature and humidity.



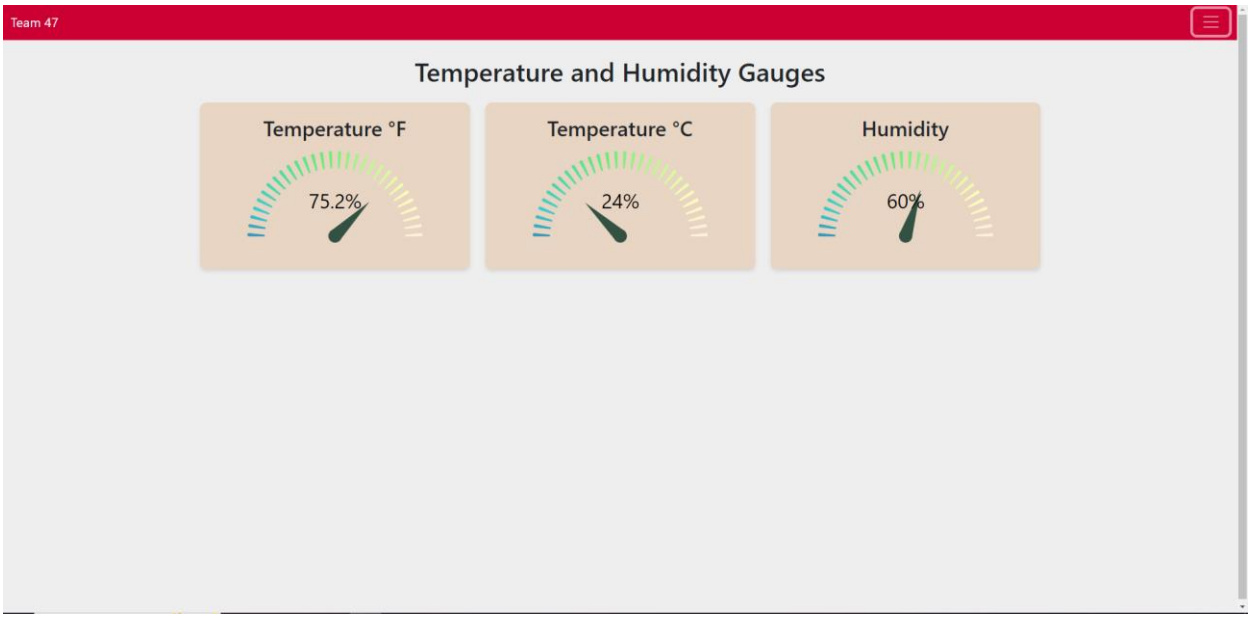
The user can record the current information and it will be displayed in Reports view.



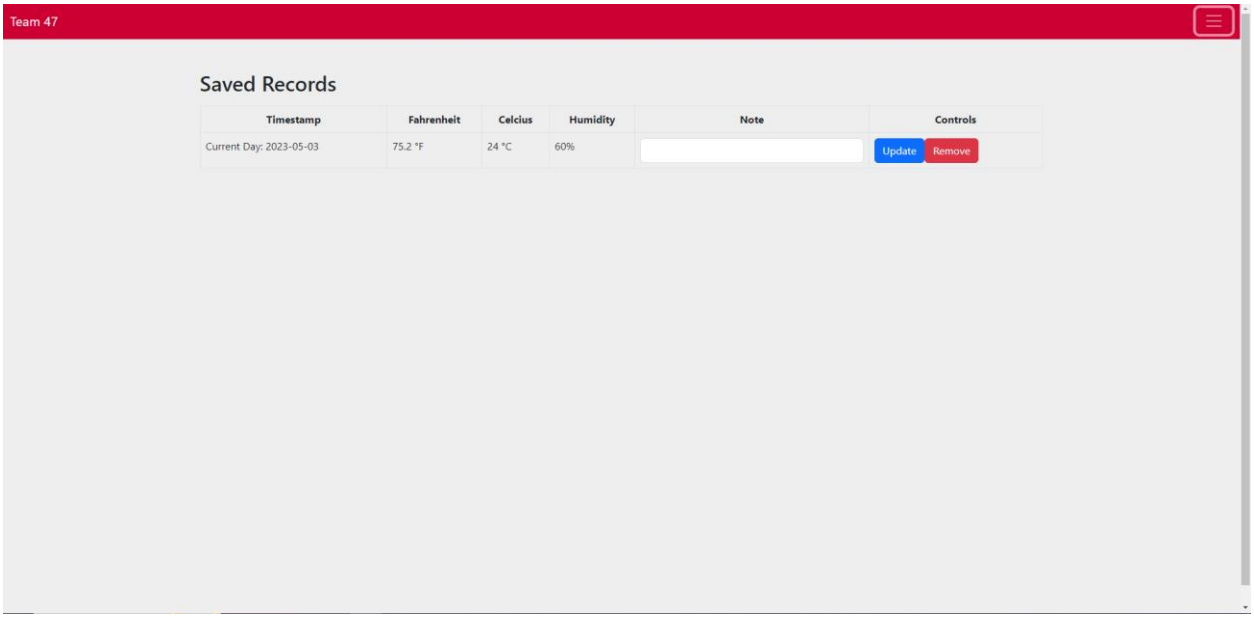
The user can access the navbar at any time and change views.



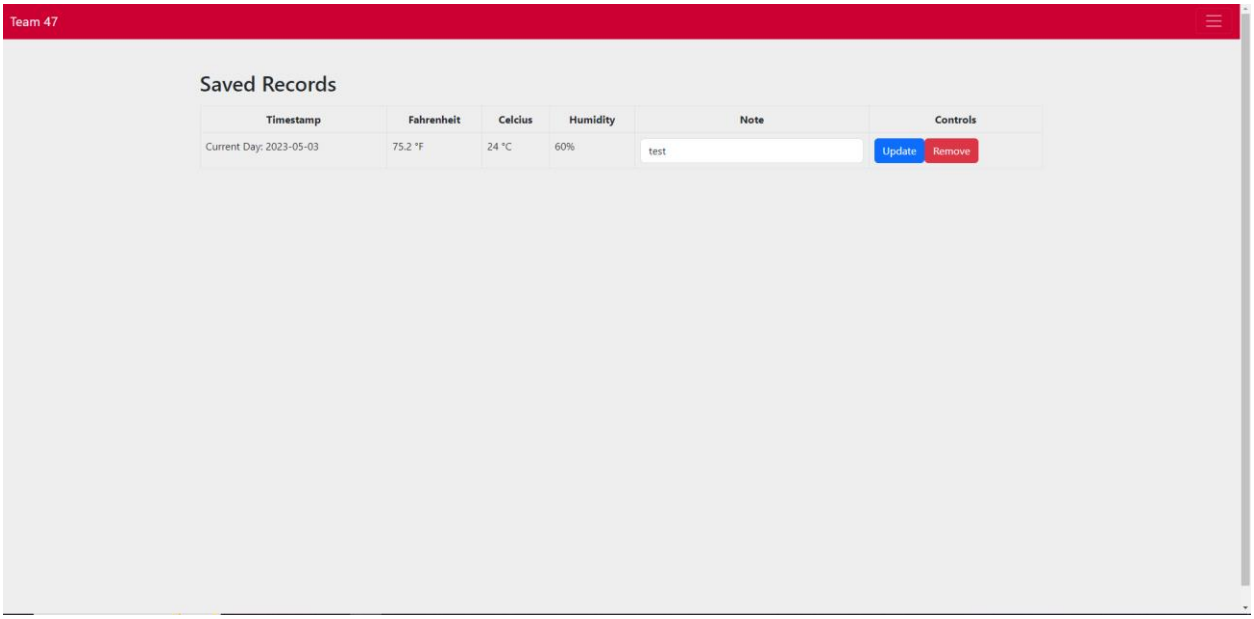
This is the graph view where the user can view the information in a graph format. Hovering over each bar graph will display the actual value.



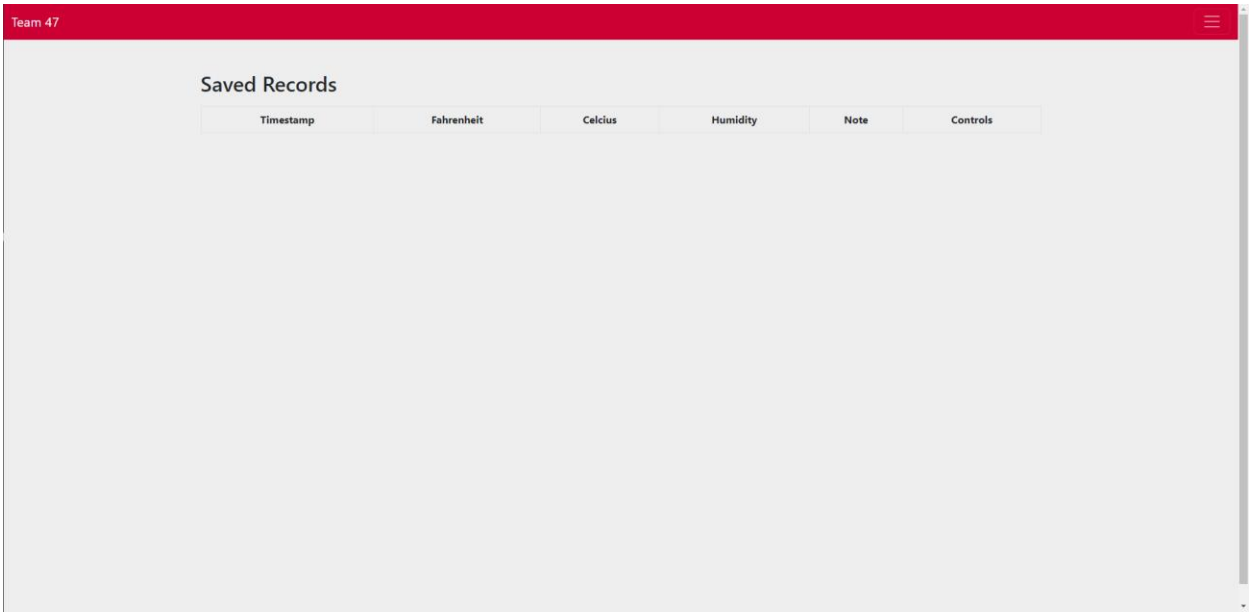
This is the gauge view where the user can view the information in a gauge format. There is an animation that plays when viewing these gauges.



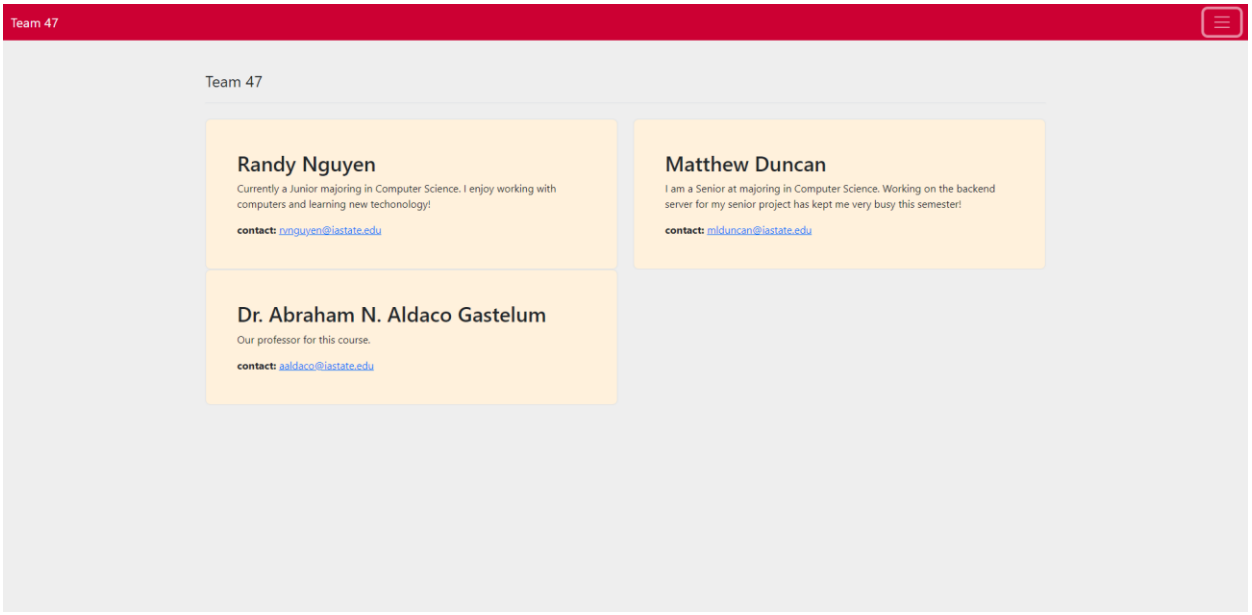
This is the reports view where the user can view the saved information they have. In the homepage if the user clicks the button to save the record, it will be viewable here.



The user can add a note to the record by typing it in the note section and then clicking update.



The user can finally remove a record by click the remove button.



Finally, we have a simple view of our information.

Installation

Installation is simple as it utilizes many of the tools, we have learned in class so far.

To run the backend use the command:

node index

If there are errors of missing packages, you may need to npm install a couple packages. The required packages for the backend are:

npm install express

npm install cors

npm install body-parser

npm install mongodb

You should see this message if its running correctly:

“`App listening at http://localhost:8081`”

To run the frontend use the command:

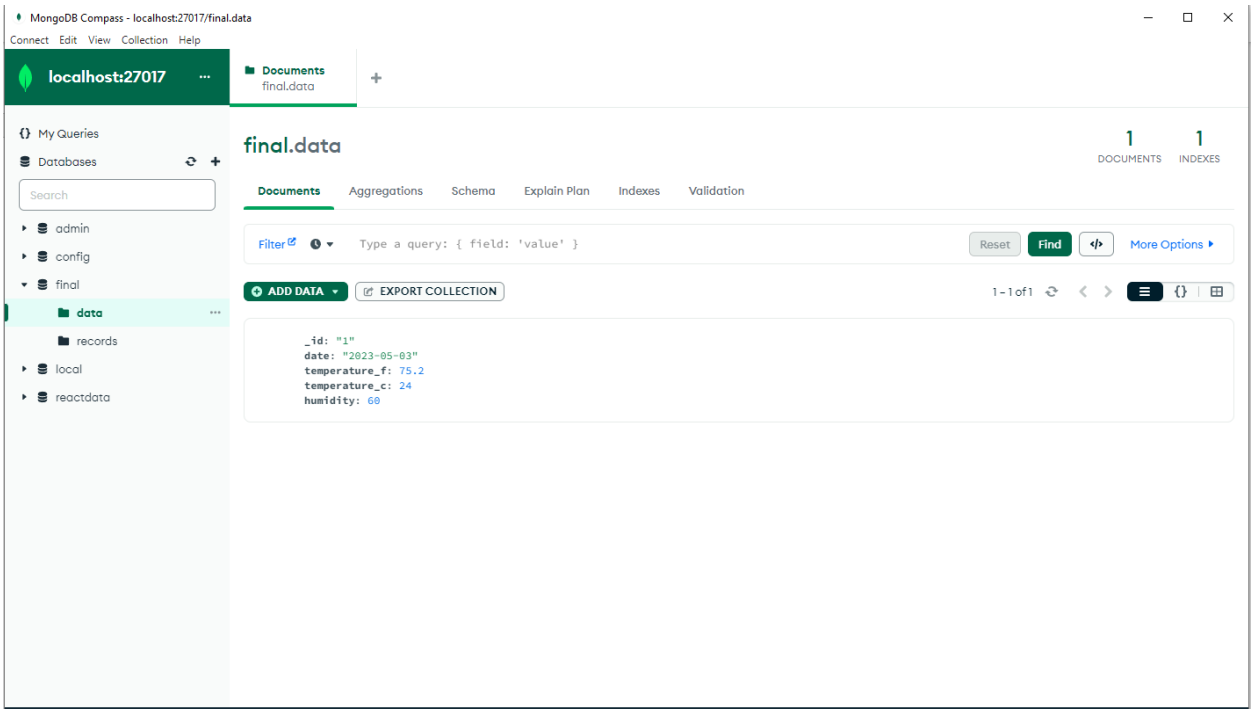
npm start

If there are errors of missing packages, you may need to npm install a couple packages. The required packages for the backend are:

npm install react-scroll

A MongoDB client is required to run the web application.

Create a new database called final and have 2 collections: data, and records. In the data collection, if you want to quickly test the web application without the raspberry pi, you can grab an example input from data.json in the backend project. Just manually place that input in mongodb. It should look something like this.



The records collection needs nothing to be done as it will change when the user interacts with the web application.