**IOT ICA**

**Weather Tracker System**

**A logo of a weather forecast

Description automatically generated**

The Weather Lookout website provides tracking of real time temperature and humidity, this data is displayed for the user to view as the temperature and humidity levels change. A Gmail login is required to log in to WeatherLookout.

**By Caitlin Maguire SD3B**

**Table of Contents**

1. **System Architecture**
2. **AWS** 
   1. Custom Domain
   2. HTTPS
   3. Available after submission
3. **Database**
4. **Pubnub** 
   1. SSL/TLS
   2. Communication
5. **Security** 
   1. IOT Device
   2. Access to communication channels
   3. Database
   4. Webserver
   5. Data in transit
6. **System Architecture**
7. **AWS**
   1. **Custom Domain**

I used the website godaddy.com to buy the domain for my project, I came across one called weatherlookout.online which was perfect for my project.

A screenshot of a web page

Description automatically generated

Next I created a hosted zone on my aws instance in route 53 and created multiple records, connecting the instance with my domain using route type A by adding in the IP address.

A screenshot of a computer

Description automatically generated

I then added the name servers into the domain on godaddy.com , which are the same routes I have on AWS.

A screenshot of a computer

Description automatically generated

The website can be access by typing in [www.weatherlookout.online](http://www.weatherlookout.online) or simply weatherlookout.online after adding the route records into route 53.

A screenshot of a computer

Description automatically generatedWebsite using [www.weatherlookout.online](http://www.weatherlookout.online)

A screenshot of a computer

Description automatically generated

Website using weatherlookout.online

* 1. **HTTPS**

After the AWS server and domain were connected the website was using http. The next thing I did was add https access to my weatherlookout.online, to do this I used letsencrypt.org to create a SSL cert for the domain. First I added the inbound security rule for https on the ec2 instance.

A screenshot of a computer

Description automatically generatedI created the FlaskApp.conf and wsgi files by using putty to ssh into the AWS server, and restarted the server. Using the website sslabs.com/ssltest I checked to ensure the website is now secure.

A screenshot of a computer

Description automatically generated

SSL Test

As shown above we can now see that the website weatherlookout.online and [www.weatherlookout.online](http://www.weatherlookout.online) are secure, I also created the SSL for www. Aswell.

* 1. **Available after submission**

I have left the AWS instance running for submission and I have the domain name purchased until December 2025. I left the AWS instance running from the beginning of my project, I found it saved a lot of time as I did not have to change the IP address in my files each time I stopped and started the instance.

1. **Database**

**MongoDB Atlas**

To hold the data about the temperature and humidity sensor I created a cluster on MongoDB Atlas called WeatherLookout.

A screenshot of a computer

Description automatically generated

Within the cluster I created a database called WeatherLookout and a collection called Weather, in the Weather collection there are 4 fields I have created. I added the temperature field, humidity , humidity type which is based on the humidity range and a created\_at timestamp. The timestamp allows me to view at what times each record has been added to the database. The data is added by a function call on the main.html page of my website, the record is added with the current temperature, humidity and associated humidity level. This is then displayed on the tracker.html page of my website.

A screenshot of a computer

Description automatically generated

**Users**

I also created two users within MongoDB Atlas one admin user for myself to use, and a sample user with read only access to the database. The users both have a username and password for database access which can be used in the connection string when connecting to the WeatherLookout database.

A screenshot of a computer

Description automatically generated

**Network Access**

In MongoDB Atlas there is a section called Network Access which allows me to select the IP Addresses who can access my database. Below you can see I have my own IP address and the IP address of the WeatherLookout server on the AWS instance, this allows me to connect when using the server. I also added IP address 0.0.0.0/0 which allows me to connect from any IP address, this ensures I can access the database without adding my IP address in each time.

A screenshot of a computer

Description automatically generated

1. **Pubnub** 
   1. **SSL/TLS**

Pubnub has a built in SSL/TLS protocol this “provides secure communication over the Internet” <https://www.pubnub.com/guides/ssl-tls/> . Pubnub encrypts the connection of the client and server, and ensures the data going between the client and server is confidential and no users can access this data without authorisation. This is a really good feature of Pubnub as it creates less work for us the developers.

* 1. **Communication**

I created the connection between the server and the hardware using Pubnub. To do this I first created a Keyset called WeatherLookout, this developed my publish key, subscribe key and secret key for me on the Pubnub website. Next I created my main.js file and added in the keys into a Pubnub function, within my .client\_secret.json file I added the keys and user id in here for security. The next step was to add the Pubnub code in to the sensors file on my raspberry pi, I installed Pubnub, put in the correct imports and added Pubnub messages at each function. I have Pubnub messages for the PIR motion sensor, the buzzer, red and green LEDS and most importantly the temperature and humidity sensor data. When I run the sensors file on the raspberry pi the sensor data is sent to Pubnub and I can view the data coming in on the Weather Lookout Channel. The data on Pubnub is then accessed by the server to get the temperature and humidity sensor data, to get the two values I set a type in the JavaScript. The temperature and humidity values are taken in via an input box once the submit button is clicked on the main.html page the add function is triggered and the values are added to the database. Each time the add is called the record is created with the sensor values, and a corresponding humidity type field , this is based on the humidity range. On the tracker.html page the user can view the temperature, humidity and humidity level as it changes.

ADD PUBNUB IMAGE

1. **Security** 
   1. **IOT Device**

The main aspect of security on the IOT device is using Pubnub to relay the sensor data from the raspberry pi to the AWS server. In terms of physical security I am using a raspberry pi 400, due to the size of the pi and the breadboard it is difficult to find a box of some sort to cover the hardware. If I had a raspberry pi 4 I would use a small cardboard box or plastic box to keep the hardware protected and covered. For now, I have the raspberry pi sitting on a high desk, the pi is in a bedroom away from any liquids or other things which may effect its physical security. The pi sitting up high, ensures no animals or children can tamper with the hardware, as I have a dog and younger brothers who would love to mess with the hardware.

* 1. **Access to Communication channels**

**5.3 Database**

A large part of database security having a backup running of the database. This can be easy to do using a MySQL database, however I am using a NoSQL database with MongoDB Atlas therefore this was more difficult. The cluster I created for my project is under the free tier section, within the free tier you cannot access the built in backup features of MongoDB Atlas.

I followed the same tutorial as I did for the group project:  <https://medium.com/@andrewskangah/how-to-create-a-middle-man-automatic-backup-for-mongodb-atlas-mo-sandbox-cluster-tier-5e7276d66ee2>.

First of all, I created a local backup of the database on my desktop. I did this by creating a folder called mongobackup, in the terminal I went into this location and pasted in the export connection string from the MongoDB Atlas website. In the connection string I included my username and password for the admin user, as I have full access to the database. I also needed to edit the connection string by including the database name and the specified collection, the collection is Weather.

I first went into MongoDB Atlas and copied the export tools, I put in the username and password of the admin user into the mongoexport and the location of where I would like to store my backup and set the type to JSON.

**Local Database Backup:**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

weather\_data.json

**Backup on Server:**

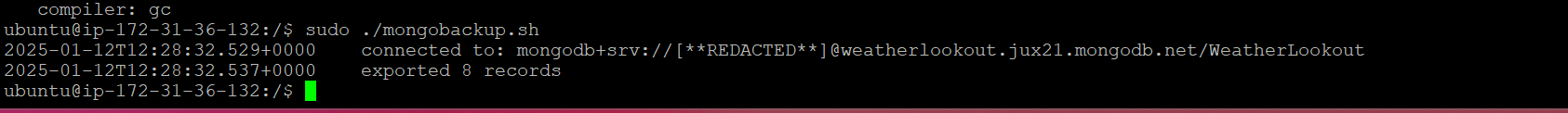
Next I decided to do a backup on the server to create a cron job to back the database up every week. I cannot do this for a local backup as the cron job needs to be done on a server, and the server would not have access to the files on my local desktop.

**Script File:**

A screen shot of a computer

Description automatically generated

I had an error coming up for mongoexport therefore I followed the steps to install the mongo tools from the MongoDB website into the ubuntu server. Next I ran the script file and this is the output:



The server is connected to the database via the mongo export string and has copied over the 8 records from the database.

Next I started working on the cron job to run the script every Friday at 12pm, the database will not have many values in it therefore it makes sense to set the backup every week instead of every day or multiple times per week. For a large database I would set the backup more frequently as I would be dealing with large records of data.

**Cron:**

Using a cron job I have set the script file to run every Friday at 12pm.

A screenshot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

The weather\_data.json has been created in the mongobackup folder successfully. All 8 records in the database have been backed up on the server.

A screen shot of a computer screen

Description automatically generated

* 1. **Webserver**

**Public and Private Keys**

To access the AWS instance I created a key pair, this reduces the need to enter usernames and passwords and makes the login to the instance more secure. In putty I added the .ppk file as the private key and logged in as ubuntu. I did the same for WinSCP and saved the server on putty and WinSCP. By keeping the AWS instance running all of the time it reduces the need for me to change the IP address each time I try to use WinSCP or putty to access my AWS instance.

**A screenshot of a computer

Description automatically generated**

* 1. **Data in transit**

I am reading in my sensor data from a sensors.py file on my Raspberry Pi 400, to log in to the pi a username and password is required. When I use putty or WinSCP to ssh into the pi each time I must enter the IP address, username and password. The Pi is already very secure as no unauthorised user can gain access without the login details.

Pubnub instils security of the sensor data coming from the raspberry pi to the server. In this case I am sending Pubnub messages from the raspberry pi , the messages include whether motion is detected or not, if the green or red LED is on, temperature and humidity.

I have set the Pubnub channel on the debug console as Weather-Lookout-Channel and the app channel is also set in the sensors.py file on the raspberry pi to connect to the channel. When a message is received on Pubnub the server is looking out for a temperature of humidity message type, this will be taken into a hidden input box and added to the database. When the temperature and humidity data has reached the server, no unauthorised users will have access to this data without the private key.

The front end of the project which is hosted on the domain, is secured using an SSL certificate. The website has https therefore users can access my WeatherLookout website securely. The function to get the temperature and humidity in from the server to add to the database is done using a form POST request to send the data securely. The add function is triggered and the weather record containing the temperature and humidity , and the associated humidity level is sent and accessed by the database which creates a new record for the data. The data is accessed again from the database to get the most recent record from the database to display on the front end, the user will be able to view the temperature, humidity and humidity level on the tracker.html page securely. The humidity level is gotten from a humidity range , the levels are low, normal and high. The humidity is compared against the range and based on this a humidity type will be issued, this function is triggered within the server and sent securely to the database.