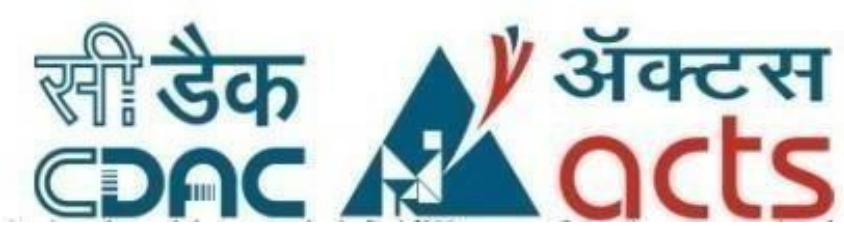
Project Report

On

IOT-Based Containerized Temperature Monitoring System



Submitted
In partial
fulfillment
For the award of the Degree of

PG-Diploma in Internet of Things

(C-DAC, ACTS (Pune))

Guided By:

- Mr. Shubham Srivastava

Submitted By:

Ayush Patwa (240340126003)
Prasad Kadam (240340126006)
Kaushal Kumar Maurya (240340126007)
Prafulkumar Bhoi (240340126010)

Centre for Development of Advanced Computing

(C-DAC, ACTS Pune- 411008)

Acknowledgement

This is to acknowledge our indebtedness to our Project Guide, Mr. Shubham Srivastava, C-DAC ACTS, Pune for his constant guidance and helpful suggestion for preparing this project on IOT-Based Containerized Temperature Monitoring System. We express our deep gratitude towards his for inspiration, personal involvement, constructive criticism that he provided us along with technical guidance during the course of this project.

We take this opportunity to thank Mr. Gaur Sunder (Associate Director, Head of Department, CDAC) for providing us such a great infrastructure and environment for our overall development.

We express sincere thanks to Ms. Namrata Ailawar (Joint Director, Process Owner, CDAC) for their kind cooperation and extendible support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude towards Ms. Risha P.R (Associate Director, Program Head, CDAC) and Ms. Pratiksha Harshe (Course Coordinator, PG-DIoT) for their valuable guidance and constant support throughout this work and help to pursue additional studies.

Also, our warm thanks to C-DAC ACTS Pune, which provided us this opportunity to carry out, this prestigious Project and enhance our learning in various technical fields.

Ayush Patwa (240340126003)
Prasad Kadam (240340126006)
Kaushal Kumar Maurya (240340126007)
Prafulkumar Bhoi (240340126010)

ABSTRACT

The "Smart Containerized Temperature and Humidity Monitoring System" is a comprehensive solution designed to continuously monitor environmental conditions within containerized spaces, particularly focusing on temperature and humidity levels. This system leverages the computational power of a Raspberry Pi, which acts as the central gateway, in conjunction with a DHT sensor to accurately measure the ambient temperature and humidity inside the container. The data collected by the sensor is processed in real-time using a backend developed with Python and the Flask framework. The processed data is then stored in a local MySQL database, ensuring that all environmental readings are securely logged and can be retrieved when necessary.

The system is equipped with a robust and intuitive web-based interface built using HTML, CSS, and JavaScript, allowing users to log in securely, view live data, and access historical records. The interface provides real-time graphical visualizations of the temperature and humidity data, enabling users to monitor environmental conditions at a glance. In addition, users can retrieve historical data by specifying parameters such as date and time, making it easy to analyze past trends.

To ensure the system's reliability and scalability, data is also backed up periodically to an AWS S3 bucket, where it is stored securely in the cloud. This redundancy ensures that even if the local storage is compromised, the data remains accessible and intact. The system also integrates AWS Simple Email Service (SES) to send automated email alerts when the temperature or humidity exceeds user-defined thresholds. This alert mechanism ensures that users are promptly notified of any potentially dangerous conditions, allowing them to take immediate corrective action.

The system is globally accessible through a web application hosted on PythonAnywhere, allowing users to monitor their container’s environment from anywhere. This real-time monitoring, secure data storage, and alert functionality make it a versatile solution for various applications, from industrial containers to shipping units. The project effectively integrates IoT technology, cloud services, and web development to deliver a reliable and user-friendly environmental monitoring system.

Table of Contents

Table Of Figures

Table Of Images

Chapter 1

1.1 Introduction

In today's rapidly evolving world, the need for efficient and reliable environmental monitoring systems has become increasingly critical, especially in industries where maintaining specific conditions is essential for the safety and quality of stored items. Containers used for industrial, logistical, and storage purposes often house goods that are sensitive to temperature fluctuations. Ensuring that these containers maintain the appropriate temperature is vital to prevent spoilage, degradation, or damage to the contents. Recognizing this necessity, the "Smart Containerized Temperature and Humidity Monitoring System" was developed to provide a robust solution for real-time monitoring of temperature within these containers.

This project centers around the Raspberry Pi, a versatile and powerful microcontroller that serves as the central gateway for data collection and processing. The system uses a DHT sensor, known for its reliability and accuracy, to continuously measure the temperature inside the container. The data collected from the sensor is processed using Python within the Flask framework, which forms the backbone of the system's backend operations. This data is then securely stored in a MySQL database, providing a local repository where all temperature readings are logged for future reference and analysis.

The system is equipped with a user-friendly web interface developed using HTML, CSS, and JavaScript. This interface allows users to securely log in, view real-time temperature readings, and access historical data. The data is presented in clear graphical formats, making it easy for users to monitor the container's conditions at a glance. Additionally, users can specify date and time parameters to retrieve historical data, facilitating a detailed analysis of temperature trends over time.

To ensure the system's reliability and data accessibility, cloud-based services such as AWS S3 are integrated for data storage, while AWS SES is used for sending alert notifications. Temperature data is periodically backed up to the cloud, ensuring that it remains secure and accessible even in the event of local storage failure. The system is also designed to send automated email alerts via AWS SES whenever the temperature exceeds predefined thresholds, enabling users to respond swiftly to any potential issues.

One of the key features of this system is its global accessibility. The web application is hosted on PythonAnywhere, allowing users to monitor and track the temperature inside their containers from anywhere in the world. This makes the system particularly useful for a wide range of applications, from industrial containers to shipping and storage units, where maintaining consistent temperature conditions is crucial.

The "Smart Containerized Temperature and Humidity Monitoring System" exemplifies the effective integration of Internet of Things (IoT) technology with cloud services and web development. By focusing on temperature monitoring, the system offers a comprehensive and scalable solution for ensuring that container environments remain within safe limits, thereby protecting the integrity of the stored contents.

1.1 Objective

The primary objective of the "Smart Containerized Temperature and Humidity Monitoring System" is to develop an integrated solution that effectively monitors and manages environmental conditions within containerized spaces. This objective is achieved through the following specific goals:

Real-Time Monitoring: To utilize a Raspberry Pi and DHT sensor for accurate

real-time measurement of temperature and humidity within the container, ensuring continuous environmental oversight.

Data Management: To design and implement a backend system using Python and Flask for processing and storing environmental data in a local MySQL database, ensuring secure and reliable data logging.

User Interface: To create an intuitive web-based interface using HTML, CSS, and JavaScript that allows users to log in securely, view live data, access historical records, and analyze trends through graphical visualizations.

Data Backup and Security: To incorporate periodic data backup to AWS S3 for cloud storage, providing redundancy and ensuring data integrity and accessibility even in the event of local storage failure.

Alert Mechanism: To integrate AWS Simple Email Service (SES) for automated email notifications, alerting users when temperature or humidity levels exceed predefined thresholds and enabling prompt corrective actions.

Global Accessibility: To deploy the system on PythonAnywhere, enabling users to access and manage their container's environmental conditions from anywhere in the world, thereby enhancing operational flexibility.

1.2 Project Model

This project is developed using the "Spiral Model," an iterative development model introduced by Barry Boehm in 1988. The Spiral Model is particularly effective for projects that require continuous refinement and risk management, making it ideal for the development of a system as complex and critical as the "Smart Containerized Temperature Monitoring System." Each phase of the Spiral Model begins with a design goal and ends with a client review, ensuring that the project is aligned with user needs and expectations at every step.

Steps of the Spiral Model as Applied to This Project:

1. **Requirements Definition**:
   The system requirements for the temperature monitoring system were defined in detail, taking into account the needs of the users and the operational constraints of the containerized environment. This phase involved extensive interviews with stakeholders, including potential users and experts in

environmental monitoring.

2. Preliminary Design:

A preliminary design of the system was created, focusing on the core functionality of temperature monitoring using a DHT sensor and data storage on a Raspberry Pi. The design also included the initial framework for the web interface and cloud integration.

3. First Prototype Construction:

A first prototype was constructed based on the preliminary design. This prototype included basic functionality such as temperature data collection, local storage in a MySQL database, and a simple user interface to display real-time data.

4. Second Prototype Development:

The second prototype was developed following a four-step procedure:

1. Evaluation of the First Prototype:
   The first prototype was evaluated to identify its strengths, weaknesses, and potential risks, such as data accuracy, system reliability, and user interface usability.
2. Refinement of Requirements:
   Based on the evaluation, the requirements were refined to address identified issues enhance system performance, including improving data visualization and integrating AWS services for alerts and cloud storage.
3. Planning and Designing the Second Prototype:
   A detailed plan was developed for the second prototype, incorporating the refined requirements and addressing the weaknesses of the first prototype.
4. Construction and Testing of the Second Prototype:
   The second prototype was constructed and rigorously tested to ensure it met the updated requirements. This version included a more robust web interface, enhanced data storage capabilities, and the integration of AWS services for alerts and backup.

5. Risk Evaluation:

At each iteration, the system was evaluated for risks, including potential cost overruns, data security concerns, and operational reliability. Based on this evaluation, the project could have been adjusted or even aborted if the risks were deemed too great, though in this case, the risks were managed

effectively.

6. Iterative Refinement:

The preceding steps were repeated, with each iteration bringing the prototype closer to the final product. This iterative process ensured that the system was continually refined and improved, incorporating user feedback and addressing any newly identified risks or issues.

7. Final System Construction:

Once the refined prototype met all requirements and passed all tests, the final system was constructed. This system included all the desired features, such as real-time temperature monitoring, secure data storage, cloud integration, and global accessibility.

8. System Evaluation and Maintenance:

The final system underwent thorough evaluation and testing to ensure it met the desired performance standards. Routine maintenance procedures were established to prevent failures and minimize downtime, ensuring the system's long-term reliability and effectiveness.

Fig 1: Spiral Model

Chapter 2

LITERATURE REVIEW

The integration of sensor technologies with web-based platforms for environmental monitoring has gained significant traction in recent years. This literature review examines the methodologies and technologies employed in the development of a Smart Containerized Temperature and Humidity Monitoring System, which utilizes a DHT sensor, Raspberry Pi, Python Flask, MySQL, HTML/CSS/JavaScript, AWS SES, AWS S3, and PythonAnywhere. The review explores the core components and methodologies relevant to this project, providing insights into their application and effectiveness.

Sensor Technologies

Temperature and humidity sensors, such as the DHT series, are widely used in environmental monitoring due to their affordability and reliability. The DHT11 and DHT22 sensors, commonly used in similar projects, are known for their ease of use and reasonable accuracy. This sensors provide digital signals that can be easily read by microcontrollers, making them suitable for real-time monitoring applications.

Raspberry Pi as a Gateway

The Raspberry Pi, a versatile single-board computer, is frequently used as a gateway device in IoT (Internet of Things) applications. Its popularity stems from its low cost, ease of programming, and wide support for various interfaces. Studies such as those by highlight the Raspberry Pi's role in bridging the gap between sensor data acquisition and web-based data management. It is an effective platform for collecting data from sensors and processing it before

transmitting it to a web server.

Data Storage and Management

Data storage in relational databases, like MySQL, is a common practice for managing structured data. MySQL's robustness and scalability make it a preferred choice for local storage solutions in many IoT projects. The use of MySQL in the proposed system allows for efficient data retrieval and manipulation, supporting functionalities such as historical data querying and real-time data updates.

Web Technologies for Visualization

HTML, CSS, and JavaScript are fundamental technologies for web development. These technologies are used to create user interfaces that display real-time and historical data. JavaScript libraries such as Chart.js or D3.js are often employed to visualize data in graphical formats. These libraries enable dynamic and interactive data visualization, which is crucial for user engagement and effective data interpretation.

Email Notifications and Alerts

AWS Simple Email Service (SES) is a scalable email service that integrates well with various applications for sending notifications and alerts. AWS SES provides reliable email delivery with features for managing email quotas and handling bounced emails. In the context of this project, SES is utilized to send alerts when temperature or humidity levels exceed predefined thresholds, ensuring timely notifications to users.

Cloud Storage and Hosting

Storing data in cloud storage solutions, such as AWS S3, offers scalability and durability. AWS S3 allows for the storage of large volumes of data with high availability. In this project, S3 is used to archive data at regular intervals,

ensuring that historical data is preserved and accessible for long-term analysis. Additionally, hosting the web application on platforms like PythonAnywhere provides a global reach, allowing users to access the monitoring system from anywhere.

The integration of these technologies and methodologies creates a comprehensive monitoring system capable of real-time data acquisition, visualization, and management. The use of Raspberry Pi and DHT sensors for data collection, coupled with MySQL for local data storage and Python Flask for web-based interactions, provides a robust framework for environmental monitoring. The incorporation of AWS SES for alerts and AWS S3 for data archiving further enhances the system's functionality and reliability.

Chapter 3

Methodology

The methodology for the Smart Containerized Temperature and Humidity Monitoring System involves several key steps, from data acquisition to web-based visualization and alerting. This section outlines the process in detail, covering sensor integration, data management, user interface development, and cloud services.

1. Sensor Integration

- Sensor Selection: The DHT22 sensor was chosen for its accuracy and reliability in measuring temperature and humidity.
- Hardware Setup: The DHT22 sensor is connected to a Raspberry Pi via GPIO pins. The sensor's data pins are connected to the appropriate GPIO pins on the Raspberry Pi, which reads the digital signals output by the sensor.
- Data Reading: Python libraries such as Adafruit_DHT are used to interface with the sensor and read temperature and humidity values. A Python script running on the Raspberry Pi periodically fetches these readings.

2. Data Storage

- Database Design: A MySQL database is set up on the Raspberry Pi to store the sensor data. The database schema includes tables for storing temperature, humidity, timestamps, and user details.
- Data Insertion: Python scripts running on the Raspberry Pi insert the collected data into the MySQL database at regular intervals. The script also ensures that the data is formatted correctly and stored with accurate

timestamps.

3. Web Development

- Backend Development: Python Flask is used to develop the backend server that interacts with the MySQL database. Flask routes handle data retrieval requests, user authentication, and data submission.
- Frontend Development: The user interface is created using HTML, CSS, and JavaScript. JavaScript libraries such as Chart.js are utilized to display real-time temperature and humidity data in graphical formats.
- User Authentication: A login page is implemented to ensure that only authorized users can access the data. User-specific databases are managed to store and retrieve data according to user credentials.

4. Data Visualization

- Real-Time Display: The web application provides real-time graphical representation of the temperature and humidity data. JavaScript dynamically updates the charts as new data arrives.
- Historical Data: Users can view historical data by selecting specific time periods. Queries to the MySQL database fetch and display historical records in graphical format.

5. Alerting System

- Threshold Setup: Users can define thresholds for temperature and humidity levels. When these thresholds are exceeded, the system triggers an alert.
- Email Notifications: AWS SES is configured to send email notifications when the sensor readings exceed the predefined thresholds. The Flask backend integrates with AWS SES to handle email dispatch.

6. Cloud Storage and Backup

- Data Archiving: To ensure long-term data preservation, data is uploaded to AWS S3 buckets at regular intervals (every hour). This process is automated using Python scripts that interact with the AWS S3 API.
- Global Hosting: The web application is hosted on PythonAnywhere, providing global access and scalability. This ensures that users can access the monitoring system from anywhere with an internet connection.

Chapter 4

Implementation

4. Requirements: -

4.1. Hardware Requirements

1) Raspberry Pi:

Model: Raspberry Pi 3 Model B

Accessories: MicroSD card (at least 16GB), power supply, and case

Connectivity: Internet connection (via Wi-Fi or Ethernet)

2) Sensor:

Type: DHT22 (for accurate temperature and humidity measurement)

Wiring: Required cables to connect the sensor to the Raspberry Pi GPIO pins

3) Additional Hardware:

PC/Laptop: For development and configuration tasks

Monitor, Keyboard, and Mouse: For initial setup and debugging of the Raspberry Pi

2. Software Requirements

1) Operating System:

Raspberry Pi OS: The latest version of Raspberry Pi OS (formerly Raspbian) installed on the microSD card

2) Programming Languages and Libraries:

Python: Version 3

3) Libraries:

- Adafruit_DHT for sensor data acquisition
- Flask for backend development
- MySQL Connector for database interaction
- boto3 for interacting with AWS services
- Werkzeug

4) Web Technologies:

Frontend: HTML5, CSS3, JavaScript

JavaScript Libraries: Chart.js and Canvas

5)Database:

MySQL: Installed and configured on the Raspberry Pi
Cloud Services:

AWS SES: For sending email notifications
AWS S3: For storing historical data
Hosting:

PythonAnywhere: For deploying and hosting the web application globally
3. Network Requirements
Internet Connectivity:

Raspberry Pi: Reliable internet connection for data transmission, cloud service interaction, and web hosting
PC/Laptop: Internet access for development and testing
Access and Security:

Database: Secure access to the MySQL database (local and remote as needed)
AWS Services: Proper configuration of AWS SES and S3 with appropriate access controls
4. User Requirements
Authentication:

User Accounts: A login system for users to access their specific data and settings
Alerts:

Email Alerts: Users need to configure email settings to receive notifications about threshold breaches

4.1 Working: -

1. Data Acquisition

Sensor Measurement:

DHT Sensor: The DHT sensor is installed inside the container. This sensor measures the ambient temperature and humidity. It provides data at regular intervals based on its sampling rate.

Data Collection:

Raspberry Pi Interface: The DHT sensor is connected to the Raspberry Pi via GPIO (General Purpose Input/Output) pins. The Raspberry Pi reads the data from the sensor using Python scripts. These scripts interface with the sensor's API to obtain temperature and humidity values.

2. Data Processing

Data Processing on Raspberry Pi:

Data Acquisition Script: A Python script running on the Raspberry Pi continuously collects data from the DHT sensor. This script ensures that data is read accurately and handles any sensor communication issues.

Data Formatting: The raw data is processed and formatted into a structure suitable for database insertion, such as a JSON or a CSV format.

Database Insertion:

MySQL Database: The formatted data is then inserted into a local MySQL database. The database schema includes tables for storing real-time data, historical data, and user information.

Data Storage: Data is inserted into the database with a timestamp, allowing for accurate tracking of temperature and humidity over time.

3. Data Display

Web Interface Access:

User Access: Users access the web application via a web browser. The application is hosted globally on PythonAnywhere, ensuring that users can access it from anywhere.

Live Data Retrieval:

Backend API: The web application uses a Flask-based API to query the MySQL database for the latest temperature and humidity data.

Data Retrieval: The API retrieves the most recent data entries and sends them to the front end of the application.

Graphical Visualization:

Front-End Interface: The web application, developed with HTML, CSS, and JavaScript, processes the data received from the backend. It uses JavaScript libraries like Chart.js or D3.js to create interactive graphs and charts.

Live Data Display: The data is displayed in real-time on the user's dashboard, providing a visual representation of current environmental

conditions.

4. Historical Data Access

User Interaction:

User Requests: Users can interact with the web interface to specify the range of historical data they wish to view, such as selecting a date range or specific time periods.

Data Querying:

Backend Queries: The Flask API handles requests for historical data. It queries the MySQL database based on user input and retrieves the relevant historical data entries.

Data Display:

Graphical Representation: Historical data is visualized in graphs and charts, similar to live data. This allows users to analyze trends, compare past and present conditions, and make informed decisions based on historical insights.

5. Email Alerts

Threshold Monitoring:

Alert Configuration: AWS SES (Simple Email Service) is configured to monitor temperature and humidity data for predefined thresholds (e.g., maximum and minimum limits).

Alert Triggering:

Threshold Detection: If the system detects that the data exceeds or falls below these thresholds, it triggers an alert.

Email Composition: AWS SES composes an email alert that includes information about the threshold breach and relevant data.

Email Delivery:

Notification Dispatch: The email is sent to the user's registered email address, providing details about the condition that triggered the alert and any recommended actions.

6. Data Archiving

Hourly Uploads:

Data Aggregation: At the end of each hour, the collected data is aggregated into a batch.

Upload to AWS S3: The aggregated data is uploaded to an AWS S3 bucket, a cloud storage service. This batch includes the timestamped data for that hour.

Archival Management:

Storage Management: AWS S3 ensures the durability and availability of the data, handling replication and backup automatically to prevent data loss.

7. Global Access

Web Hosting:

PythonAnywhere Deployment: The web application is hosted on PythonAnywhere, which provides a platform for deploying and running Python-based web applications globally.

User Access:

Global Reach: Users can log in to the web application from any location with internet access. The hosted application provides a consistent user experience regardless of geographical location, allowing users to monitor and manage their container's environmental conditions from anywhere in the world.

8. HTTP (Hypertext Transfer Protocol)

The Hypertext Transfer Protocol (HTTP) is the foundation of the World Wide Web, and is used to load webpages using hypertext links. HTTP is an application layer protocol designed to transfer information between networked devices and runs on topof other layers of the network protocol stack.

HTTP is a protocol for fetching resources such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.

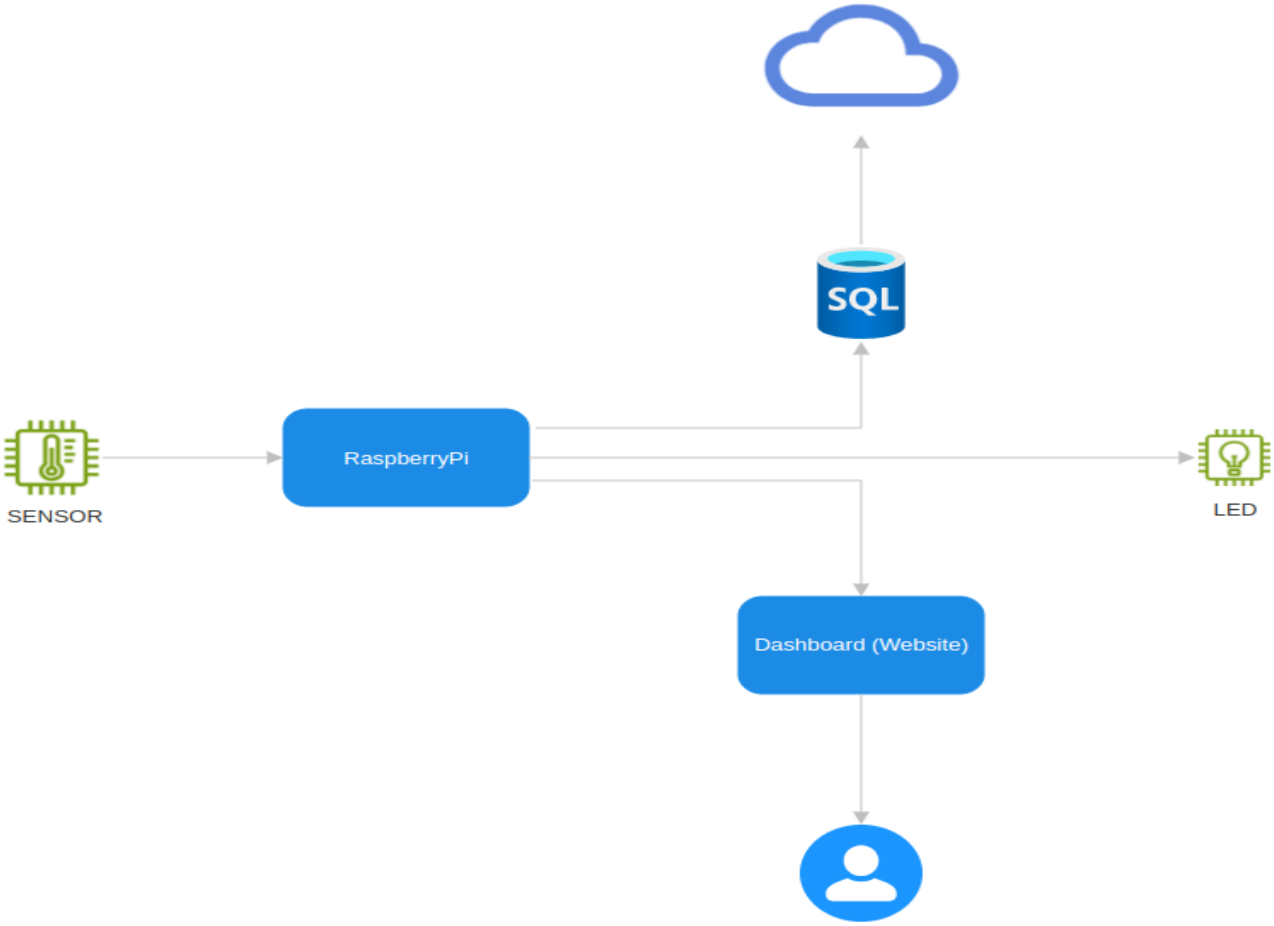


Fig 1: Block Diagram

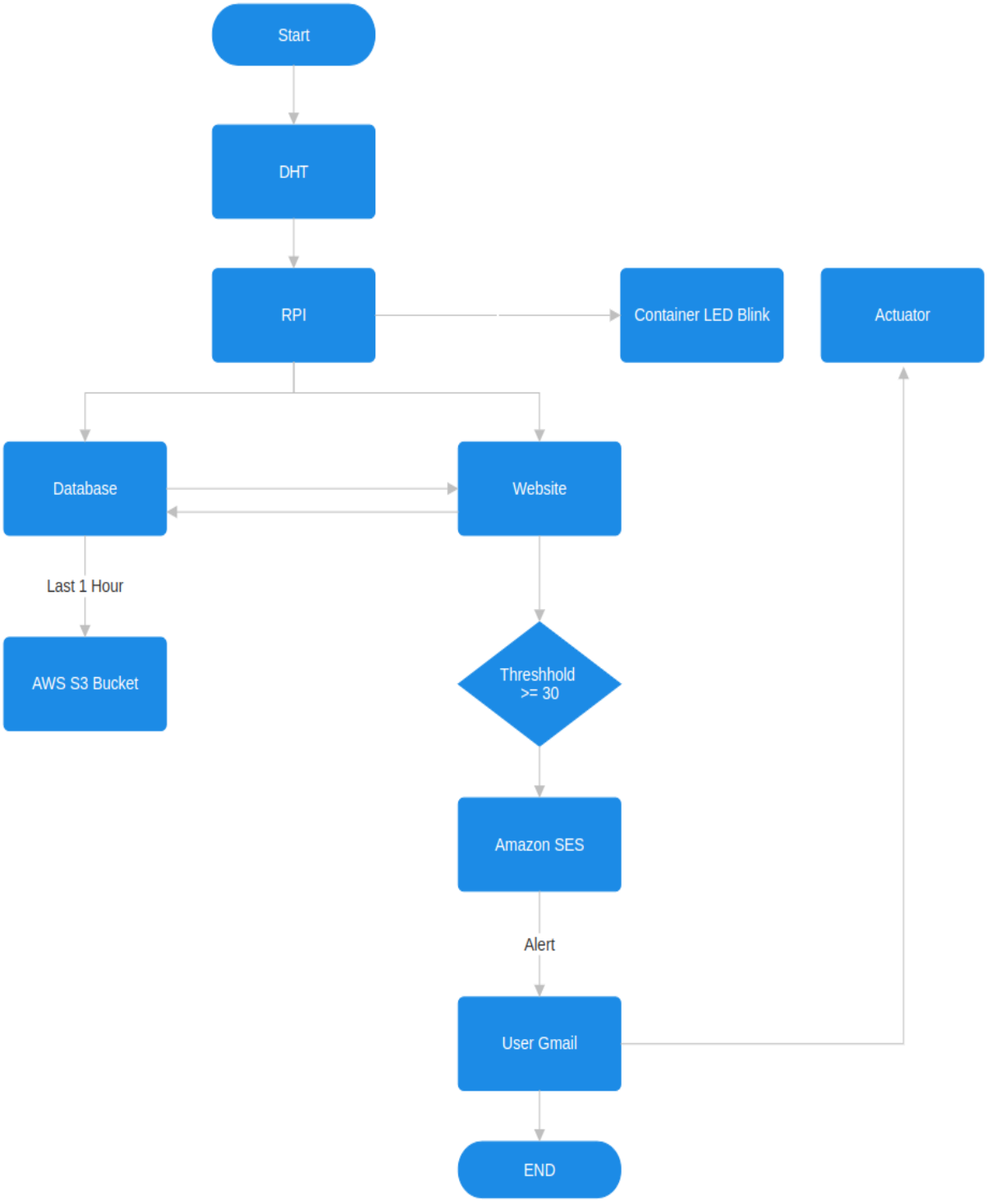


Fig 2: Flow Chart

a. Components: -

4.5.1 ESP-32

Raspberry Pi:

Description: The Raspberry Pi is a small, affordable, and versatile single-board computer developed by the Raspberry Pi Foundation. For this project, a Raspberry Pi 3 or 4 model is typically used due to its enhanced processing power and connectivity options.

Specifications:

Processor: Quad-core ARM Cortex-A72 CPU (in Raspberry Pi 4) or similar.

Memory: 2GB, 4GB, or 8GB RAM options (depending on the model).

Connectivity: Built-in Wi-Fi, Bluetooth, and Ethernet ports for network connectivity.

GPIO Pins: General Purpose Input/Output (GPIO) pins for interfacing with external sensors and devices.

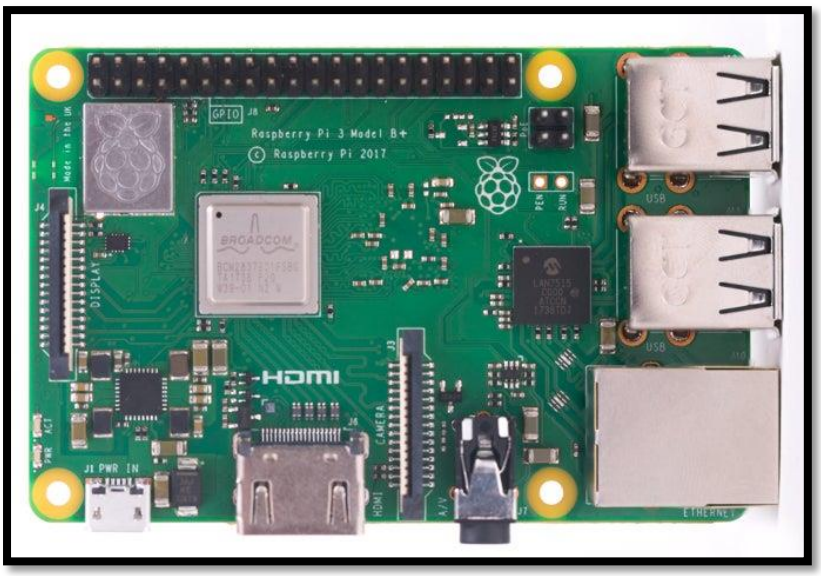
Role in the Project:

Data Collection: Interfaces with the DHT sensor through its GPIO pins to collect temperature and humidity readings.

Data Processing: Runs Python scripts to process sensor data and handle logic for storing and retrieving data from the MySQL database.

Web Hosting: Hosts the Flask-based web application that provides users with access to real-time and historical data via a web browser.

Data Management: Manages data upload to AWS S3 and sends alerts using AWS SES.



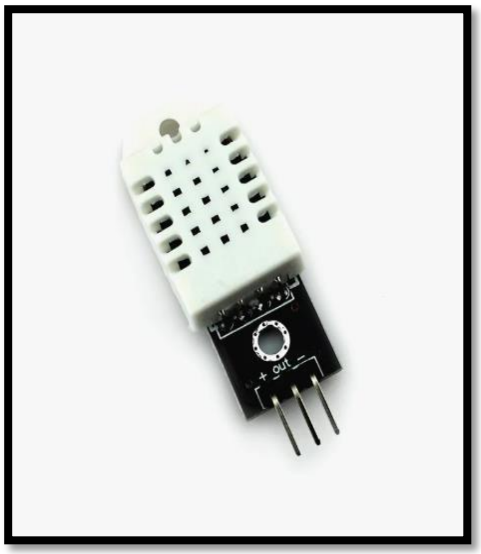
IMG 1: RaspberryPi

4.5.2 DHT 22

DHT22 is a popular temperature and humidity based digital sensor. Thisis the upgraded version of the DHT11 temperature and humidity sensor. The sensor uses a capacitive humidity sensor and a thermistor-based tempera-ture sensor to measure the ambient humidity and temperature. The humidi-ty sensing ranges from 0% to 100% with ±1% accuracy and the temperature sensing ranges from -40 degrees to the 80 degrees Celsius with ±0.5°C ac- curacy. The sampling time of this sensor is 2 seconds almost.

This Temperature and Humidity Sensor uses digital pins to communicate with the microcontroller unit and does not have any kind of analogy pins. The module also has the inbuilt pull-up resistor and additional filter capaci-tor to support the DHT22 sensor. Thus, the module is available in ready to gomode and can be directly connected with the microcontroller unit without using any kind of additional components.

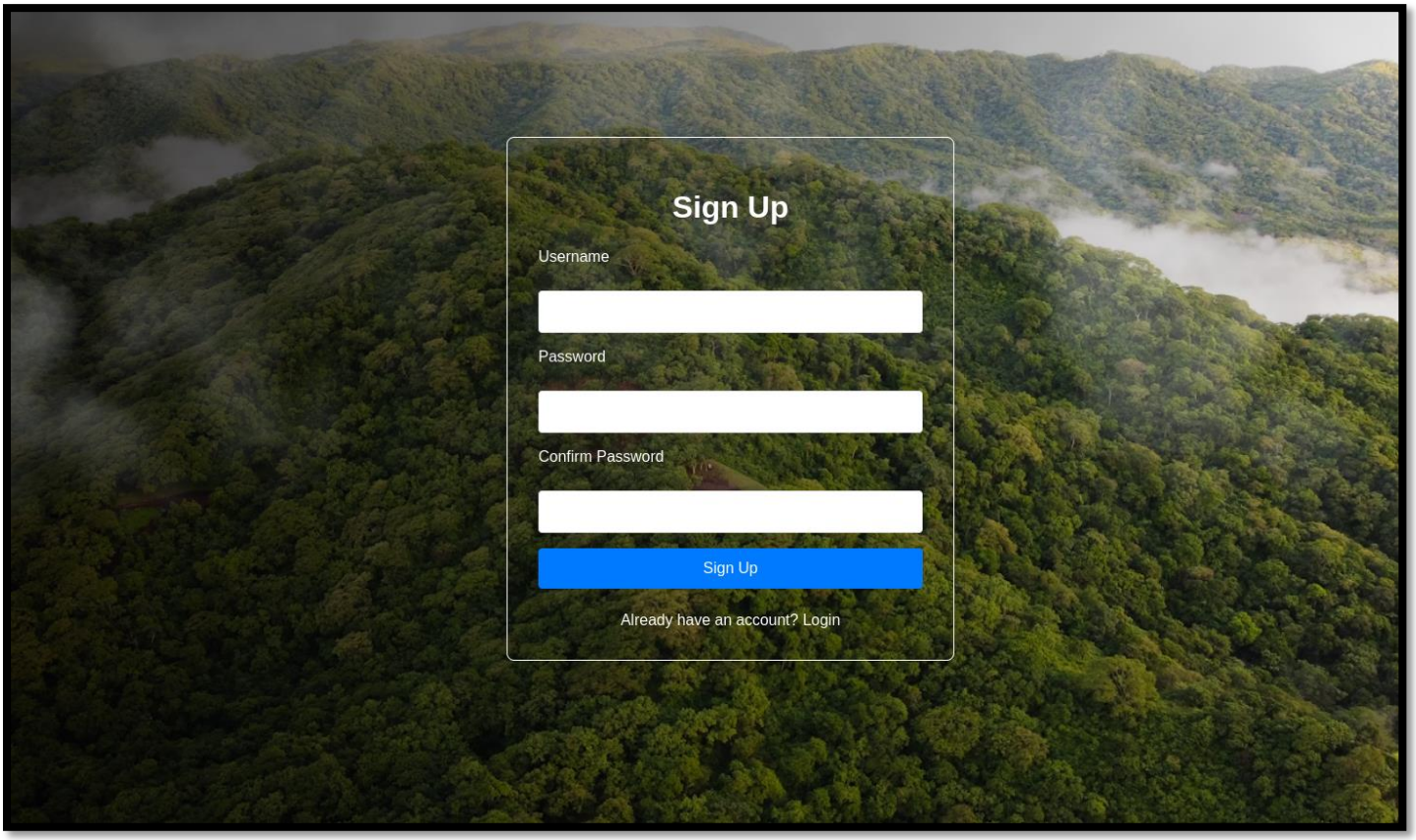
1. Operating Voltage: 3.5V to 5.5V
2. Digital I/O
3. 5 Hz sampling rate (Once every 2 Seconds)
4. Measuring current: 0.3mA
5. Low power consumption
6. Temperature range: -40 to 80 degree C
7. Humidity range: 0% to 100%
8. Accuracy: ±0.5°C and ±1%
9. Dimension: 5x 4 x 2 cm

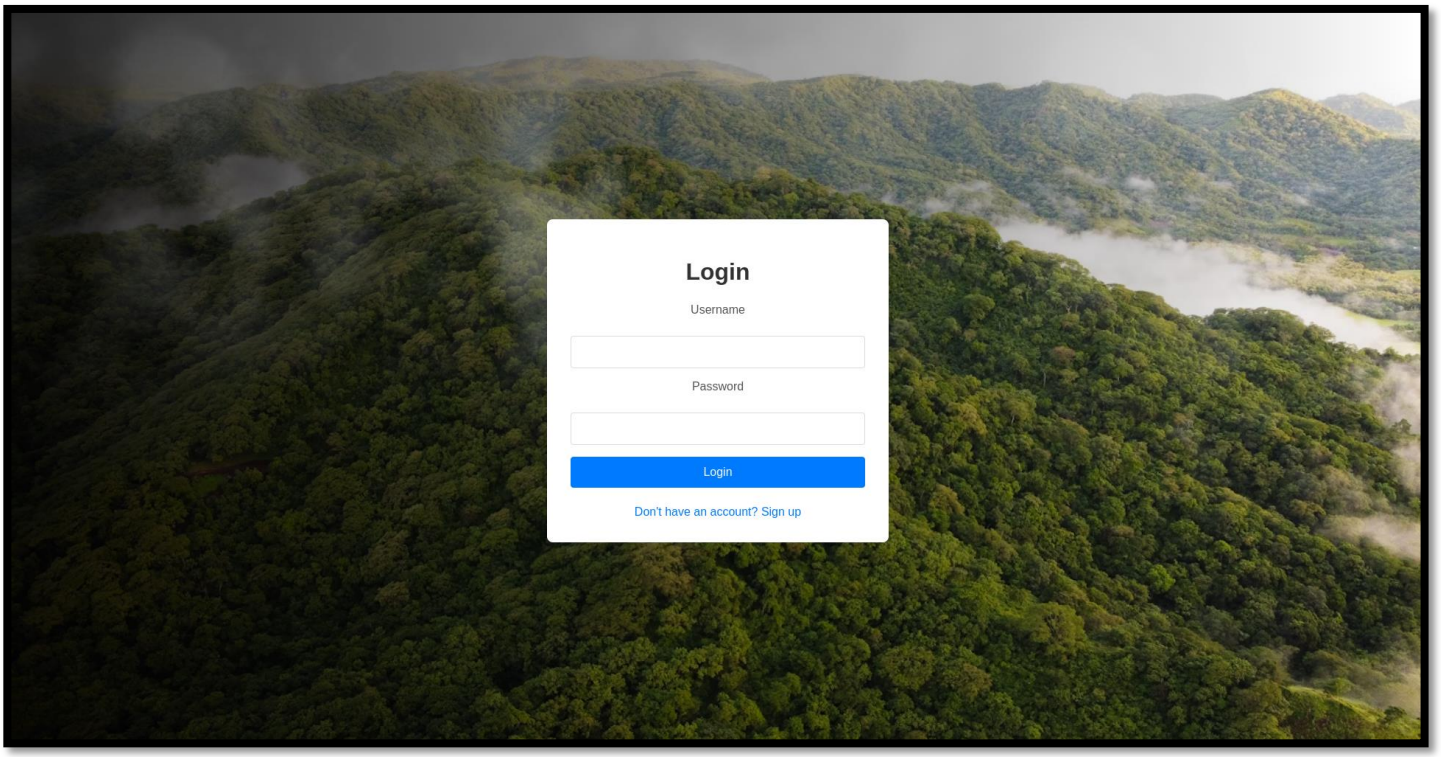


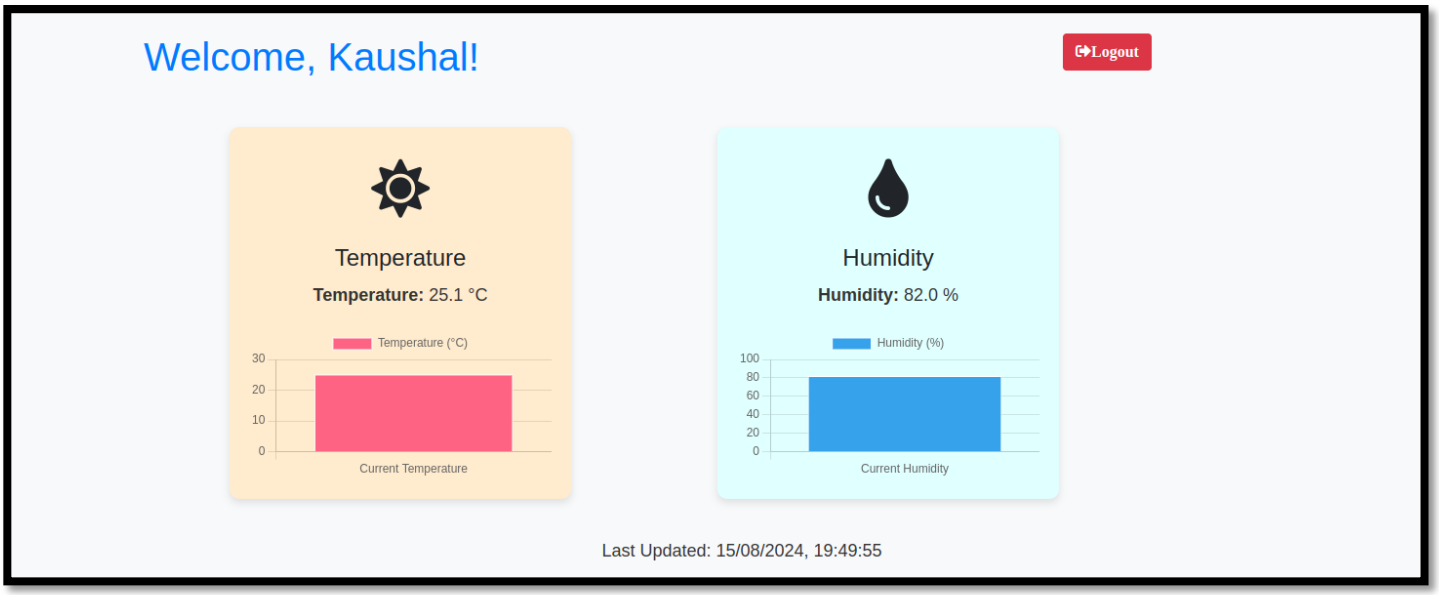
IMG 2: DHT 22

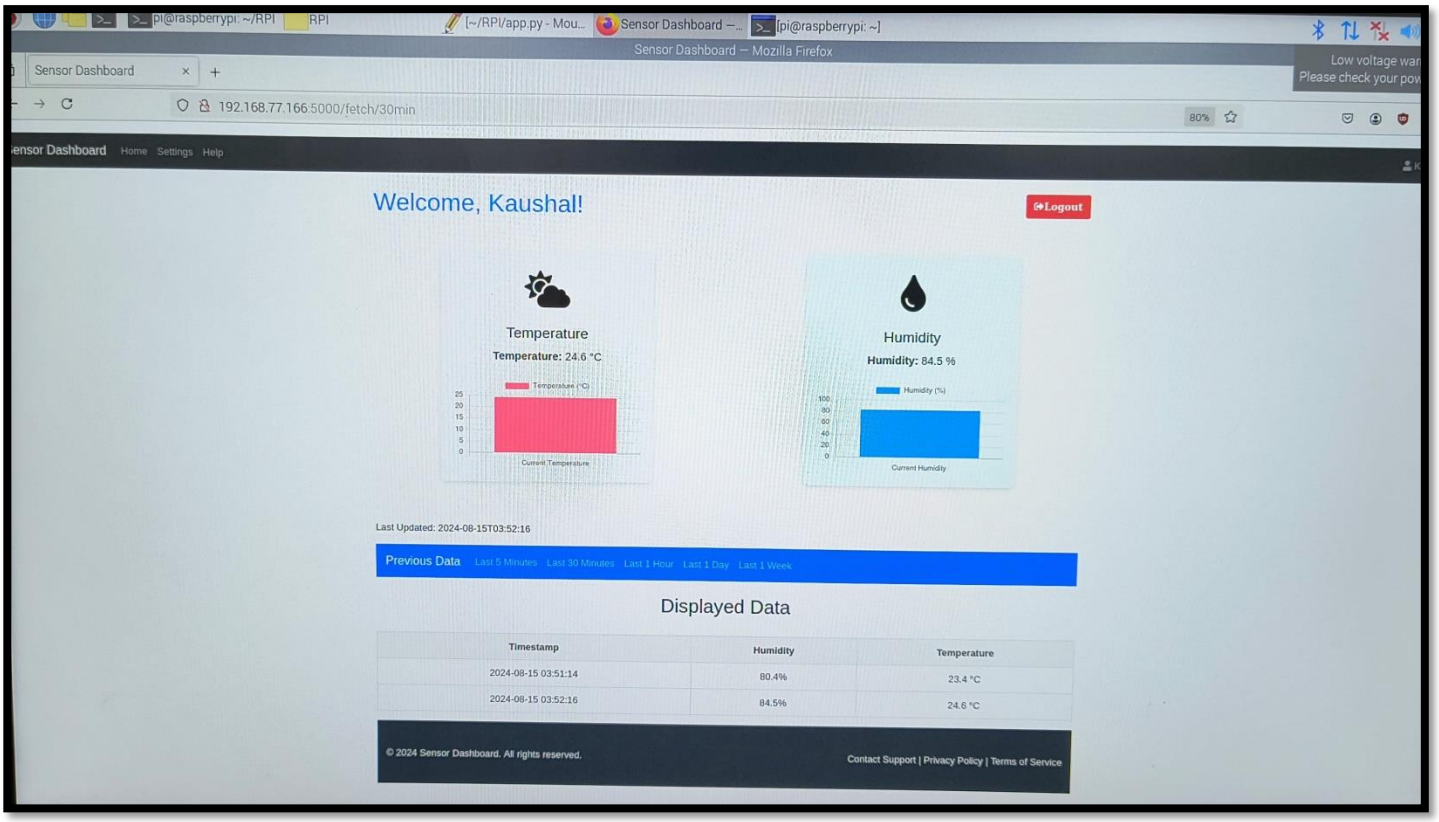
Chapter 5

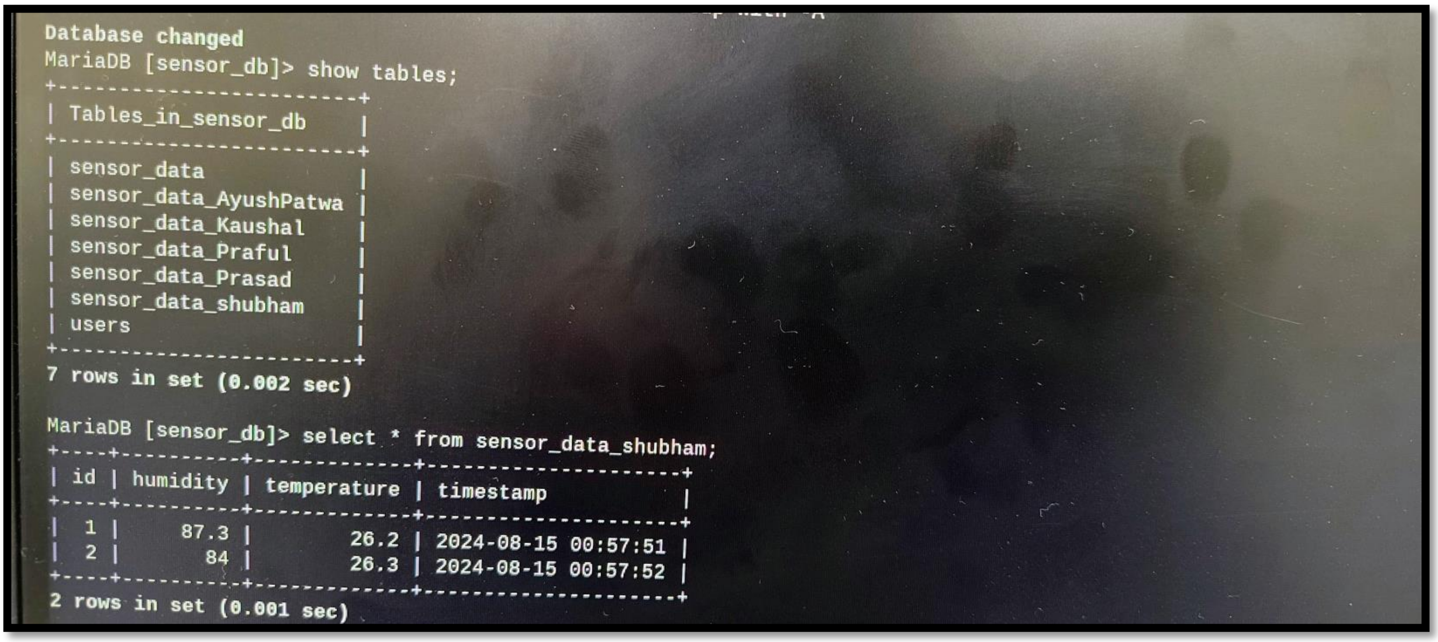
Results






Welcome, Kaushal!
Logout
Temperature
Temperature: 25.1 °C
Temperature (°C)
30
20
10
0
Current Temperature
Humidity
Humidity: 82.0 %
Humidity (%)
100
80
60
40
20
0
Current Humidity
Last Updated: 15/08/2024, 19:49:55


Sensor Dashboard — Mozilla Firefox
192.168.77.166:5000/fetch/30min
Sensor Dashboard
Home
Settings
Help
Welcome, Kaushal!
Logout
Temperature
Temperature: 24.6 °C
Humidity
Humidity: 84.5 %
Last Updated: 2024-08-15T03:52:16
Previous Data
Last 5 Minutes
Last 30 Minutes
Last 1 Hour
Last 1 Day
Last 1 Week
Displayed Data
Timestamp
Humidity
Temperature
2024-08-15 03:51:14
80.4%
23.4 °C
2024-08-15 03:52:16
84.5%
24.6 °C
© 2024 Sensor Dashboard. All rights reserved.
Contact Support | Privacy Policy | Terms of Service

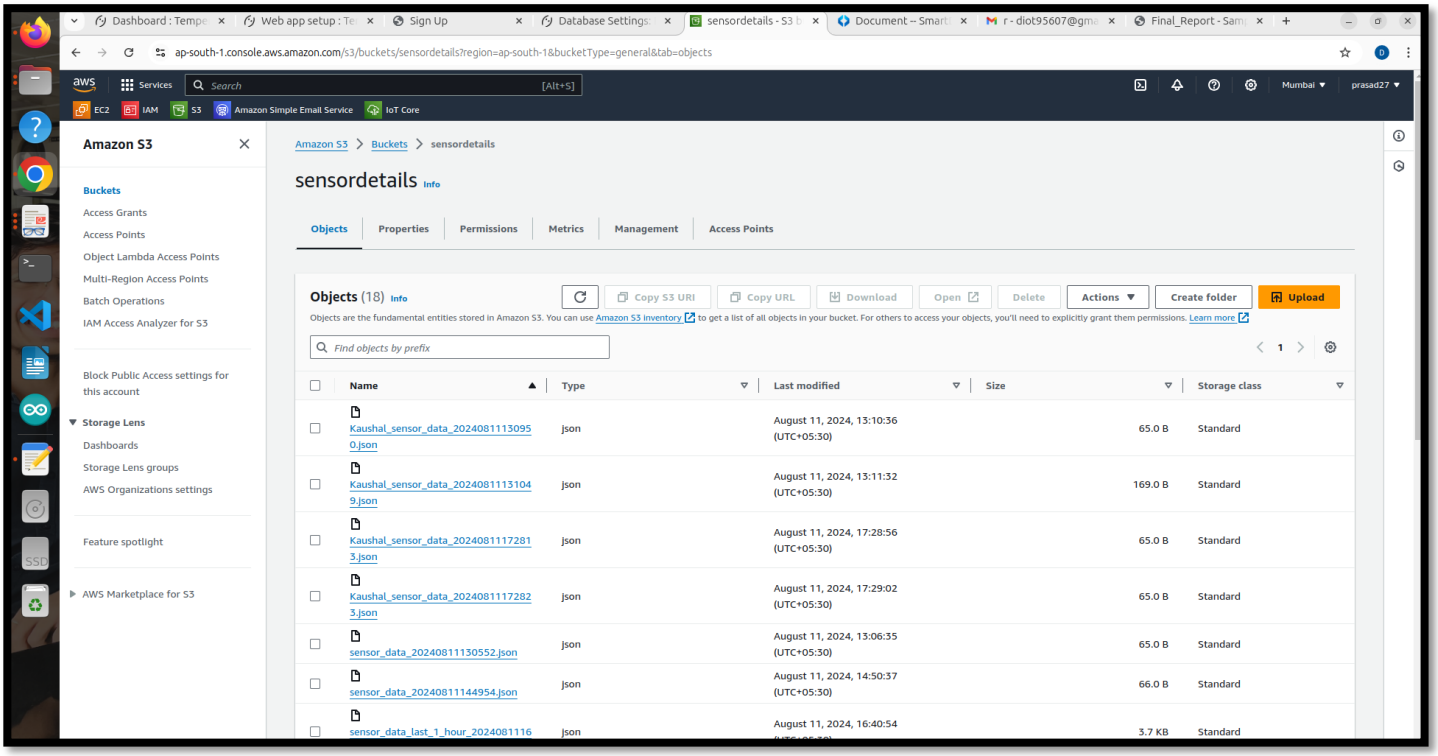


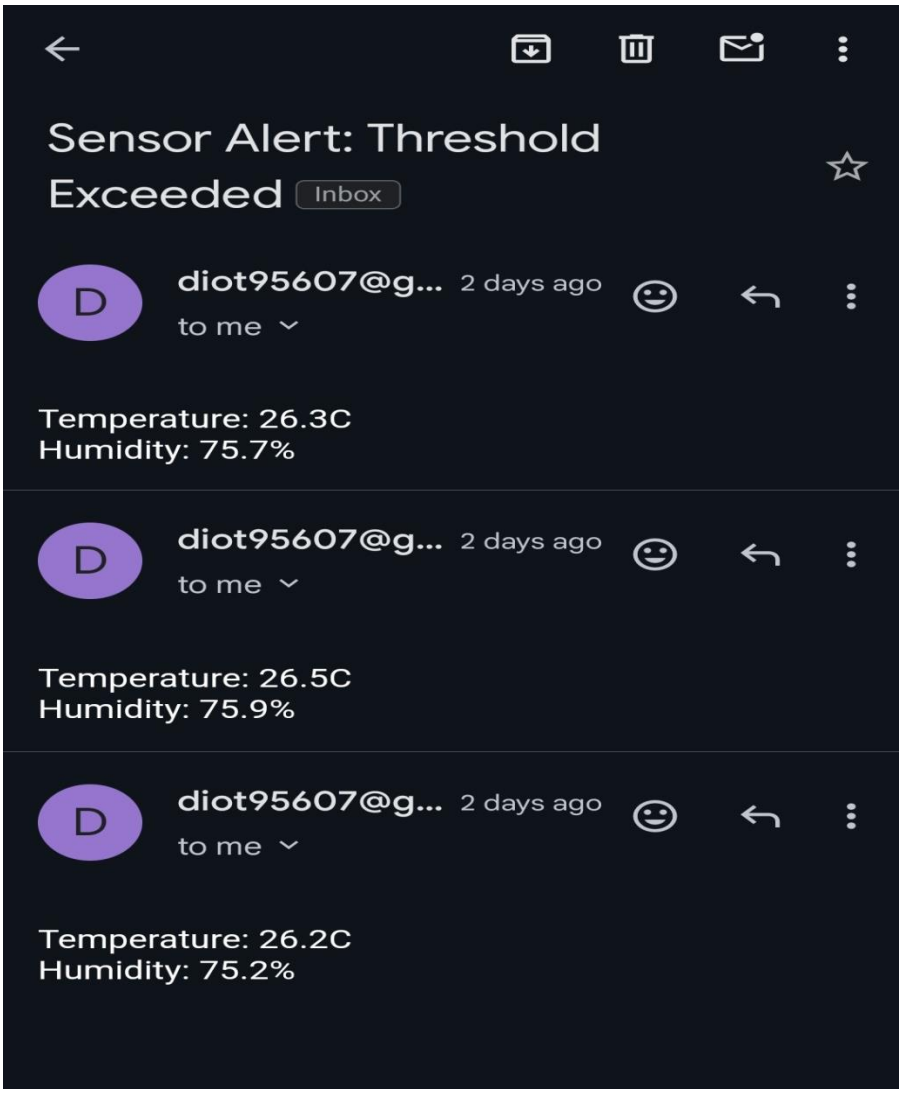
```
Database changed
MariaDB [sensor_db]> show tables;
+-------------------------+
| Tables_in_sensor_db     |
+-------------------------+
| sensor_data             |
| sensor_data_AyushPatwa  |
| sensor_data_Kaushal     |
| sensor_data_Praful      |
| sensor_data_Prasad      |
| sensor_data_shubham     |
| users                   |
+-------------------------+
7 rows in set (0.002 sec)

MariaDB [sensor_db]> select * from sensor_data_shubham;
+----+----------+-------------+---------------------+
| id | humidity | temperature | timestamp           |
+----+----------+-------------+---------------------+
|  1 |     87.3 |        26.2 | 2024-08-15 00:57:51 |
|  2 |       84 |        26.3 | 2024-08-15 00:57:52 |
+----+----------+-------------+---------------------+
2 rows in set (0.001 sec)
```

```
MariaDB [sensor_db]> select * from users;
+----+------------+------------------------------------------------------------------------------------------------------+---------------------+
| id | username   | password                                                                                             | created_at          |
+----+------------+------------------------------------------------------------------------------------------------------+---------------------+
|  1 | Kaushal    | pbkdf2:sha256:600000$Ysy2URxrExbayssK$030c3c697f30626a168b46b8837f0686ed3c108d39bd35f35498dc2662f2a797 | 2024-08-05 18:41:05 |
|  2 | Praful     | pbkdf2:sha256:600000$vdi93EDw3v1clID9$7bcfd0ecbe86f3f29bcfbdff51a54291eccc4c02997a73cb6942f48f1f43d714 | 2024-08-06 00:17:26 |
|  3 | Prasad     | pbkdf2:sha256:600000$ad5JZW4Z7JX3eRDi$0f9a84eeeab37aee6071dec7d33c1cd828e8bbd834417d8f6eb4f6ccc35014d6 | 2024-08-06 05:48:06 |
|  4 | AyushPatwa | pbkdf2:sha256:600000$eLSe6U84kL5cGHLY$460381cad23be15f2407038ad1d0ebcf139b0a31df3d5ea84443c5f1ae38ab2e | 2024-08-12 12:10:56 |
|  5 | shubham    | pbkdf2:sha256:600000$n0C51Te7Cr0AxIzQ$31795329170931598b31485aa51282f111e82c88121c84f4d7bf0b8d485e5d40 | 2024-08-15 00:57:33 |
+----+------------+------------------------------------------------------------------------------------------------------+---------------------+
5 rows in set (0.002 sec)

MariaDB [sensor_db]>
```


Amazon S3
Buckets
Access Grants
Access Points
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
Feature spotlight
AWS Marketplace for S3
Amazon S3 > Buckets > sensordetails
sensordetails Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (18) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Find objects by prefix
Name
Type
Last modified
Size
Storage class
Kaushal_sensor_data_20240811130950.json
Kaushal_sensor_data_20240811131049.json
Kaushal_sensor_data_20240811172813.json
Kaushal_sensor_data_20240811172823.json
sensor_data_20240811130552.json
sensor_data_20240811144954.json
sensor_data_last_1_hour_2024081116
json
August 11, 2024, 13:10:36 (UTC+05:30)
August 11, 2024, 13:11:32 (UTC+05:30)
August 11, 2024, 17:28:56 (UTC+05:30)
August 11, 2024, 17:29:02 (UTC+05:30)
August 11, 2024, 13:06:35 (UTC+05:30)
August 11, 2024, 14:50:37 (UTC+05:30)
August 11, 2024, 16:40:54
65.0 B
169.0 B
66.0 B
3.7 KB
Standard


Sensor Alert: Threshold Exceeded
Inbox
diot95607@g... 2 days ago
to me
Temperature: 26.3C
Humidity: 75.7%
diot95607@g... 2 days ago
to me
Temperature: 26.5C
Humidity: 75.9%
diot95607@g... 2 days ago
to me
Temperature: 26.2C
Humidity: 75.2%

Chapter 6

6.1 Conclusion

The Smart Containerized Temperature and Humidity Monitoring System effectively combines hardware and software to provide accurate, real-time environmental monitoring. Using a Raspberry Pi and DHT sensor, the system collects and processes temperature and humidity data, which is then displayed through an intuitive web interface. Key features include live data visualization, historical data access, email alerts via AWS SES, and secure data archiving on AWS S3. Hosted on PythonAnywhere, the application offers global accessibility, ensuring users can monitor conditions from anywhere. Overall, the project meets its objectives and provides a solid foundation for future enhancements.

6.2 Future Enhancement:

To advance the Smart Containerized Temperature and Humidity Monitoring System, several enhancements could be considered. Implementing advanced data analytics and machine learning algorithms could enable predictive insights, allowing for proactive adjustments to maintain optimal conditions. Developing a mobile application for iOS and Android would offer users real-time monitoring and alerts on their smartphones, enhancing accessibility and convenience. Integrating additional sensors for parameters such as air quality, light intensity, or pressure would provide a more comprehensive view of the container environment. Improving the web interface with interactive dashboards and sophisticated data visualization tools would enhance user experience and data analysis capabilities. Furthermore, incorporating role-based access control would enable personalized user management, while integration with IoT platforms could facilitate broader data connectivity and automation. Lastly, adding multi-language support would make the system more accessible to a global audience, ensuring that users from various regions can effectively utilize the system.

Chapter

References

[1] Raspberry Pi Foundation. (2023). Raspberry Pi Documentation. Retrieved from https://www.raspberrypi.org/documentation.

[2] Adafruit. (2023). DHT11 & DHT22 Sensors. Retrieved from https://learn.adafruit.com/dht/overview.

[3] Flask Documentation. (2024). Flask Web Framework. Retrieved from https://flask.palletsprojects.com/
Official documentation for Flask, including installation, tutorials, and API references..

[4] MySQL Documentation. (2024). MySQL Reference Manual. Retrieved from https://dev.mysql.com/doc/

Comprehensive guide to MySQL database management and operations.

[5] AWS SES Documentation

Amazon Web Services. (2024). Amazon Simple Email Service (SES) Documentation. Retrieved from https://docs.aws.amazon.com/ses/

Information on setting up and using AWS SES for email notifications.

[6] AWS S3 Documentation

Amazon Web Services. (2024). Amazon S3 Documentation. Retrieved from https://docs.aws.amazon.com/s3/

Details on using Amazon S3 for cloud storage, including configuration and best practices.

[7] PythonAnywhere Documentation

PythonAnywhere. (2024). PythonAnywhere Documentation. Retrieved from https://help.pythonanywhere.com/ Guide to hosting Python applications on PythonAnywhere, including setup and deployment instructions.