Temperature and Humidity Prediction

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Amritanshu (201351)

&

Aditya Sharma (201230)

Under the guidance & supervision of

Dr. Amol Vasudeva

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

TABLE OF CONTENTS

CERTIFICATE

This is to certify that the work which is being presented in the project report titled "Temperature and humidity prediction" in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by "Amritanshu, 201351 and Aditya Sharma 201230" during the period from August 2023 to May 2024 under the supervision of Dr. Amol Vasudeva, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Amritanshu, 201351

Aditya Sharma, 201230

The above statement made is correct to the best of my knowledge.

Dr. Amol Vasudeva
Assistant Professor (SG)
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat,

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled **'Temperature and Humidity Prediction'** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Amol Vasudeva** (Assistant Professor (SG), Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Student Name: Amritanshu
Roll No.: 201351

Student Name: Aditya Sharma
Roll No.: 201230

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)
Supervisor Name: Dr. Amol Vasudeva
Designation: Assistant Professor (SG)
Department: CSE
Dated:

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for his divine blessing making it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to my supervisor **Dr. Amol Vasudeva**, **Assistant Professor (SG),** Department of CSE, Jaypee University of Information Technology, Waknaghat. Her deep knowledge & keen interest in the field of "**Artificial Intelligence**" helped us to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr. Amol Vasudeva**, Department of CSE, for her kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patience of my parents.

Amritanshu
201351

Aditya Sharma
201230

LIST OF ABBREVIATIONS

MVC	Model View Controller
JS	Javascript
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
DB	Database
ML	Machine Learning
WiFi	Wireless Fidelity
JSON	Javascript Object Notation
JWT	JSON Web Token

LIST OF FIGURES

ABSTRACT

In an era of increasing climate variability and intensifying weather events, accurate and timely weather forecasts have become indispensable for informed decision-making and effective risk management. The project "Temperature/Humidity Prediction " addresses this critical need by developing a comprehensive monitoring system that harnesses the power of machine learning to provide reliable and precise predictions of temperature and humidity conditions for the upcoming week.

At the heart of this system lies the integration of the DHT11 sensor and the Arduino ESP8266 microcontroller. These components work in tandem to continuously gather real-time temperature and humidity data, providing a continuous stream of valuable environmental information. This data is then meticulously transformed into a structured format, serving as the backbone for training a machine learning model.

The machine learning model, meticulously crafted through a combination of data preprocessing, feature engineering, and algorithm selection, is the cornerstone of the system's predictive capabilities. By analyzing historical weather patterns and identifying underlying relationships between temperature, humidity, and other relevant meteorological factors, the model learns to discern the intricate dynamics of the environment. Equipped with this knowledge, the model can accurately forecast temperature and humidity trends for the forthcoming week, empowering users to make informed decisions and prepare for potential weather events.

The potential applications of this system extend far beyond mere weather forecasting. Farmers can leverage the predictions to optimize crop irrigation schedules and harvest plans, ensuring optimal yield and minimizing losses due to unfavorable weather conditions. Construction workers can utilize the forecasts to make informed decisions regarding site preparation and material selection, enhancing safety and efficiency in their operations. Energy managers can employ the predictions to effectively manage energy consumption, reducing costs and optimizing resource utilization. Healthcare professionals can tailor patient care and preventive measures based on predicted humidity levels, mitigating the risk of respiratory illnesses and other weather-related health concerns.

CHAPTER 1: INTRODUCTION

Climate change has emerged as a defining challenge of our time, characterized by rising temperatures, shifting weather patterns, and intensifying extreme events. These changes pose significant challenges to a wide range of sectors, including agriculture, construction, energy management, and healthcare. Accurate and timely weather forecasts, particularly temperature and humidity predictions, have become indispensable for informed decision-making and effective risk management.

However, current weather forecasting methods often lack the precision and temporal resolution required for critical decision-making. Existing forecasting models may not adequately capture the intricate dynamics of local microclimates or incorporate real-time data streams for enhanced accuracy. Additionally, the availability of reliable and accessible environmental data is often limited, hindering effective forecasting and decision-making.

1.1 Problem Statement

The increasing variability of climate patterns and the intensifying nature of extreme weather events have necessitated the development of accurate and timely weather forecasting systems. Temperature and humidity predictions, in particular, are crucial for informed decision-making and effective risk management across a wide range of sectors, including agriculture, construction, energy management, and healthcare. However, existing weather forecasting methods often lack the precision and temporal resolution required for critical decision-making.

Current Forecasting Limitations:

Inadequate Granularity: Existing forecasting models may not adequately capture the intricate dynamics of local microclimates, leading to discrepancies between predicted and actual weather conditions.

Real Time Data Integration:

The incorporation of real-time data streams into forecasting models is often limited, hindering the ability to adapt predictions to rapidly changing weather patterns.

Data Availability Constraints:

The availability of reliable and accessible environmental data is often limited, particularly in remote or underserved areas, restricting the development of accurate and localized forecasts.

Impact on Stakeholders:

- **Agriculture:** Farmers rely on accurate forecasts to optimize irrigation schedules, crop selection, and harvest planning. Inaccurate forecasts can lead to crop losses, reduced yields, and economic hardship.
- **Construction:** Construction workers depend on reliable weather predictions to plan site preparation, material selection, and safety measures. Unforeseen weather events can result in project delays, cost overruns, and safety hazards.

Addressing the Challenges:

The development of a comprehensive monitoring system that utilizes machine learning techniques for accurate and granular temperature and humidity prediction can address the limitations of existing forecasting methods and significantly impact various stakeholders.

1.2 Objectives

This project aims to develop a robust and scalable monitoring system that utilizes machine learning techniques to forecast temperature and humidity conditions for the upcoming week. The system will harness the capabilities of the DHT11 sensor and Arduino ESP8266 microcontroller to continuously gather real-time temperature and humidity data. This data will then serve as the foundation for training a machine learning model that can accurately predict future weather trends.

The specific objectives of this project are as follows:

- Design and implement a Model-View-Controller (MVC) architecture to effectively manage data flow and interactions between the system's components.

- Develop a user-friendly dashboard using responsive design principles and accessible features to present real-time and historical temperature and humidity data in a clear and comprehensible manner.

- Employ data visualization techniques, such as line graphs, interactive elements, and color coding, to enhance the understanding of weather trends and anomalies.

- Select and train an appropriate machine learning algorithm based on the nature of the data and the prediction task.

- Evaluate the performance of the machine learning model using various metrics, such as accuracy, precision, and recall.

- Integrate the machine learning model into the system to generate reliable and accurate temperature and humidity forecasts for the upcoming week.

1.3 Methodology

To achieve the project objectives, the following methodology will be employed:

1. Data Acquisition:

- Integrate the DHT11 sensor and Arduino ESP8266 microcontroller to continuously gather real-time temperature and humidity data.

2. Data Preprocessing:

- Preprocess raw sensor data to eliminate noise, outliers, and inconsistencies.
- Apply data normalization to ensure consistency and facilitate machine learning model training.

3. Data Storage:

- Select a robust database to store the preprocessed temperature and humidity data.
- Organize data into a structured format for efficient retrieval and analysis.

4. Model-View-Controller (MVC) Architecture Implementation:

- Implement the MVC architecture to separate the system's concerns into distinct layers: model (data), view (user interface), and controller (logic).

5. User Interface Design:

- Design a user-friendly dashboard using responsive design principles and accessible features.
- Employ data visualization techniques, such as line graphs, interactive elements, and color coding, to enhance the understanding of weather trends and anomalies.

6. Machine Learning Model Training:

- Select an appropriate machine learning algorithm based on the nature of the data and the prediction task.
- Split the preprocessed data into training and testing sets.
- Train the machine learning algorithm on the training set.

7. Model Evaluation:

- Evaluate the performance of the trained machine learning model using various metrics, such as accuracy, precision, and recall.
- Fine-tune the model parameters based on the evaluation results.

8. Model Integration:

- Integrate the trained machine learning model into the system.
- Utilize the model to generate reliable and accurate temperature and humidity forecasts for the upcoming week.

1.4 Organization

The rest of this report is organized as follows:

Chapter 2 gives an overview of the literature study performed.

Chapter 3 discusses the system development and workflow.

Chapter 4 shows the testing.

Chapter 5 shows the results and evaluation.

Chapter 6 discusses the conclusion and future work

CHAPTER 2: LITERATURE SURVEY

2.1 Related Work

Several researchers have explored the application of machine learning techniques for temperature and humidity prediction. This literature review highlights five notable studies that have addressed this challenge using different approaches and methodologies.

1. Deep reservoir calculation model and its application in the field of temperature and humidity prediction

This paper introduces the DeepSALR model, a deep neural network architecture specifically designed for temperature and humidity prediction. The model utilizes a novel approach that combines deep learning with reservoir computing, demonstrating its effectiveness in capturing complex temporal dependencies in the data.

2. Distant temperature and humidity monitoring: prediction and measurement

This study focuses on developing a system for distant temperature and humidity monitoring. The proposed system employs a recurrent neural network (RNN) algorithm to predict indoor and outdoor temperature and humidity values based on data collected from mounted sensors. The results demonstrate the system's ability to accurately predict temperature and humidity levels for both indoor and outdoor environments.

3. Developing machine learning algorithms for meteorological temperature and humidity forecasting

This paper investigates the application of artificial neural networks (ANNs) for meteorological temperature and humidity forecasting. The study compares the performance of various ANN architectures, including feedforward neural networks, recurrent neural networks, and convolutional neural networks. The results indicate that feedforward neural networks achieve the highest prediction accuracy among the tested architectures.

4. Prediction of Temperature and Humidity Using IoT and Machine Learning Algorithm

This study proposes a system for temperature prediction using the linear regression algorithm and Message Queuing Telemetry Transport (MQTT) protocol. The system collects temperature and humidity data from IoT devices and utilizes the linear regression algorithm to predict future temperature values. The results demonstrate the system's ability to provide accurate temperature predictions within a limited range.

5. Predicting Temperature or Humidity in the Surrounding Environment Using Artificial Neural Network

This paper explores the use of artificial neural networks (ANNs) for predicting temperature and humidity in the surrounding environment. The study utilizes a dataset collected from various regions to train and evaluate the ANN model. The results indicate that the ANN model effectively predicts temperature and humidity values for different regions.

6. System of Wireless Temperature and Humidity Monitoring Based on Arduino Uno Platform:

Presents a wireless temperature and humidity monitoring system based on the Arduino Uno platform, utilizing the Enhanced Group Method of Data Handling (eGMDH) algorithm for prediction.

Table 1: List of References

S No.	Paper Title	Journal / Conference (Year)	Tools/ Techniques/ Dataset	Results	Limitations
1	Deep reservoir calculation model and its application in the field of temperature and humidity prediction	IEEE (2022)	Deep Neural Network, DeepSALR ,Data from the surrounding environment	The DeepSALR is capable of solving temperature and humidity prediction problems	The use of mixed neurons is beneficial to improve the prediction performance of DeepSALR
2	Distant temperature and humidity monitoring: prediction and measurement	Indonesian Journal of Electrical Engineering and Computer Science (2021)	RNN algorithm's application., Mounted sensors for indoor data, and local meteorological data sensor obtains outdoor data	To predict outdoor and indoor humidity and temperature	Outdoor settings have a little higher difference in humidity and temperature.
3	Developing machine learning algorithms for meteorological temperature and humidity forecasting	Environmental Science and Pollution Research (2021)	Artificial neural network architectures , Temperature and relative humidity data is observed and collected to predict	The predicted data is within the range of predicted values based on 95PPU, while the factors are low	May not capture sudden changes in climate

4	Prediction of Temperature and Humidity Using IoT and Machine Learning Algorithm	International Conference on Intelligent and Smart Computing in Data Analytics (2021)	Linear regression algorithm, Temperature and humidity data using Message Queuing Telemetry Transport	To predict temperature values	Limited to temperature prediction May not capture complex weather patterns
5	Predicting Temperature or Humidity in the Surrounding Environment Using Artificial Neural Network	AUG Repository (2018)	Artificial Neural Network (ANN), Data from several regions in the surrounding environment	To predict temperature in the surrounding environment	May not address all aspects of weather prediction Data
6	System of Wireless Temperature and Humidity Monitoring Based on Arduino Uno Platform	IEEE (2016)	Enhanced Group Method of Data Handling (eGMDH) , Daily temperature, daily pressure, monthly rainfall	Outperformed PNN and its variant in modeling and prediction	May not address all aspects of weather prediction Data quality and outliers can affect results

2.2 Key Gaps in Literature

While the reviewed studies have made significant contributions to temperature and humidity prediction using machine learning, they also exhibit certain limitations:

Data availability and quality: Several studies rely on limited or localized datasets, which may restrict the generalizability of their findings. Additionally, data quality issues, such as outliers and noise, can affect the accuracy of prediction models.
Model complexity and interpretability: The complexity of some machine learning models, particularly deep neural networks, can make it challenging to interpret their decision-making processes. This lack of interpretability can hinder the understanding of the underlying factors influencing weather patterns.
Limited prediction horizon: Some studies focus on short-term prediction, while others struggle to capture long-term trends and sudden weather changes. Improving the prediction horizon is crucial for effective planning and risk management.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 Requirements and Analysis:

Functional Requirements:

- **Data Acquisition:** Continuously collect temperature and humidity data from sensors using the NodeMCU ESP8266 microcontroller.
- **Data Storage:** Store the acquired temperature and humidity data in a local database, such as MongoDB, for efficient retrieval and analysis.
- **Data Transmission:** Implement a mechanism to securely transmit the stored temperature and humidity data to a central server.
- **Server-Side API:** Develop a server-side API to receive the transmitted data, perform any necessary data processing, and store it in a centralized database.
- **User Authentication:** Implement a user registration and login system to allow users to create accounts and securely access their personal data.
- **Data Visualization:** Display real-time and historical temperature and humidity data on the website dashboard, using charts, graphs, or other suitable visualization techniques.
- **Data Filtering and Search:** Provide options for users to filter and search the displayed data based on date, time, or other relevant criteria.
- **Responsive Design:** Ensure the website design is responsive and adapts seamlessly to different screen sizes and devices.

Non-Functional Requirements:

- **Data Accuracy:** Maintain data integrity and accuracy throughout the data acquisition, transmission, storage, and retrieval processes.
- **Data Security:** Implement robust security measures to protect user data from unauthorized access, modification, or disclosure.
- **Performance:** Ensure the system can handle data acquisition, transmission, processing, and visualization tasks efficiently, without compromising responsiveness or performance.

- **Scalability:** Design the system to accommodate an increasing number of users and data volume without significant performance degradation.
- **Usability:** Provide a user-friendly interface that is easy to navigate, understand, and interact with, even for non-technical users.

Analysis

Feasibility: The project is feasible with the available hardware and software technologies. The NodeMCU ESP8266 microcontroller is a cost-effective and versatile platform for data acquisition and communication. MongoDB is a popular database for storing and managing time-series data. Various web development frameworks, such as React or Angular, can be used to create a responsive and interactive website.

Challenges:

- **Data Quality Assurance:** Implementing data validation and error handling mechanisms to ensure the accuracy and consistency of acquired sensor data is crucial.
- **Secure Data Transmission:** Employing secure communication protocols, such as HTTPS, to protect data during transmission from the microcontroller to the server.
- **User Authentication and Authorization:** Implementing a robust user authentication system with proper access control measures to protect user data and prevent unauthorized access.
- **Real-time Data Synchronization:** Ensuring real-time synchronization between the sensor data displayed on the dashboard and the actual sensor readings.
- **Responsive and Adaptive Design:** Designing a user interface that can adapt seamlessly to different screen sizes and devices, providing an optimal user experience across various platforms.

Overall, the project is feasible and has the potential to provide users with real-time access to temperature and humidity data, enabling them to make informed decisions and monitor environmental conditions effectively.

3.2 Project Design and Architecture:

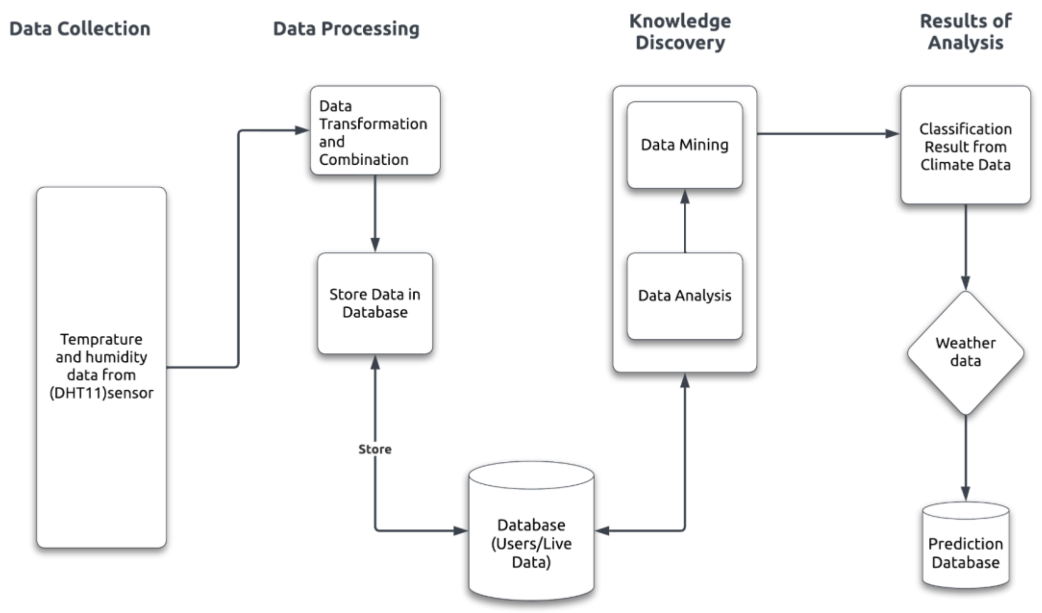


Fig. 1: DFD Diagram 1(Sensor to database and ML integration)

The above project design does these things in order:

1. **Data Collection and Transformation**

   The process begins with the acquisition of temperature and humidity data from the DHT11 sensor, a low-cost and widely used sensor for environmental monitoring. The sensor generates raw sensor readings, which need to be transformed into meaningful and usable data. This involves data cleaning to remove noise or outliers, data normalization to ensure consistency, and feature engineering to extract relevant features from the raw data.

2. **Data Storage and Management**

   The transformed data is then stored in a MongoDB database, a NoSQL database optimized for storing and managing time-series data. MongoDB's flexibility and scalability make it well-suited for handling large volumes of sensor data and enabling efficient retrieval for further analysis.

3. **Machine Learning for Prediction**

   The stored temperature and humidity data serve as the foundation for machine learning algorithms to predict future values. Machine learning techniques, such as linear regression, support vector machines, or neural networks, are trained on the historical data to identify patterns and relationships between the variables.

4. **Prediction Generation**

   Once the machine learning models are trained, they are used to generate predictions for upcoming days. The models take into account the historical data and any relevant trends or patterns to produce forecasts for temperature and humidity conditions.

5. **Overall Process**

   The entire process involves a seamless integration of hardware, software, and machine learning techniques to continuously collect, transform, store, and analyze environmental data. The resulting predictions provide valuable insights into future weather trends, enabling proactive decision-making in various applications, such as agriculture, energy management, and healthcare.

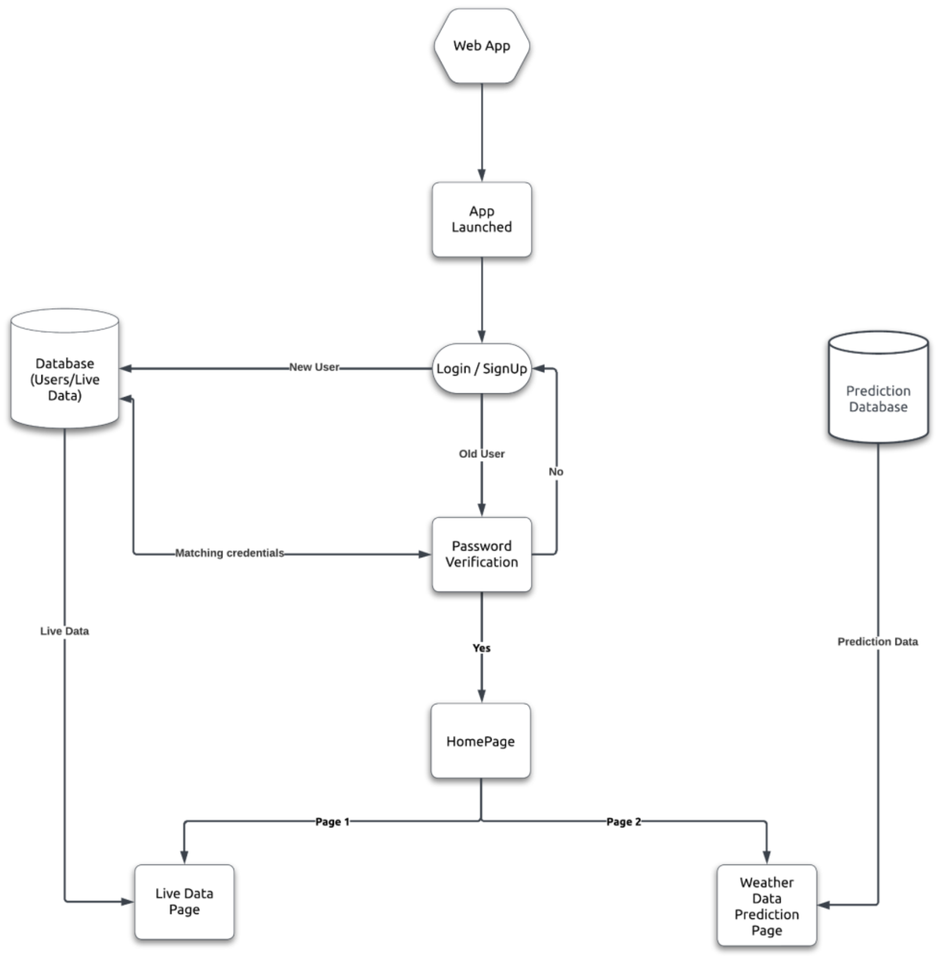


Fig. 2: DFD Diagram 2(Backend to Frontend Integration)

The web application seamlessly integrates user authentication, data visualization, and machine learning predictions to provide a user-friendly and informative experience. Upon launching the application, users encounter a user-friendly interface that guides them through the login or signup process.

1. **User Authentication**
   - Welcome Screen: Upon launching the web application, users are greeted with a welcoming interface that clearly presents options to either log in or sign up for a new account.

- Login Process: For existing users, a login form is provided where they can enter their registered email address and password. The web application securely validates the credentials against the backend database using REST APIs.
- Signup Process: New users can create an account by providing their email address, desired username, and a secure password. The web application securely stores the user information in the database.

2. **Home Dashboard**

   The home dashboard serves as the central hub for accessing real-time sensor data and machine learning predictions. It consists of two distinct pages:

   - Live Data Page: This page displays the current temperature and humidity values obtained from the DHT11 sensor. The data is dynamically updated in real-time, ensuring users always have access to the latest environmental information.

   - Predicted Data Page: This page presents the predicted temperature and humidity values for the upcoming days. These predictions are generated using machine learning algorithms trained on historical sensor data, providing users with valuable insights into future weather patterns.

3. **Seamless Data Integration with REST APIs**

   The web application employs REST APIs to establish a robust and scalable data exchange mechanism with the backend server and database. REST APIs facilitate secure and efficient data transfer between the web application and the backend infrastructure.

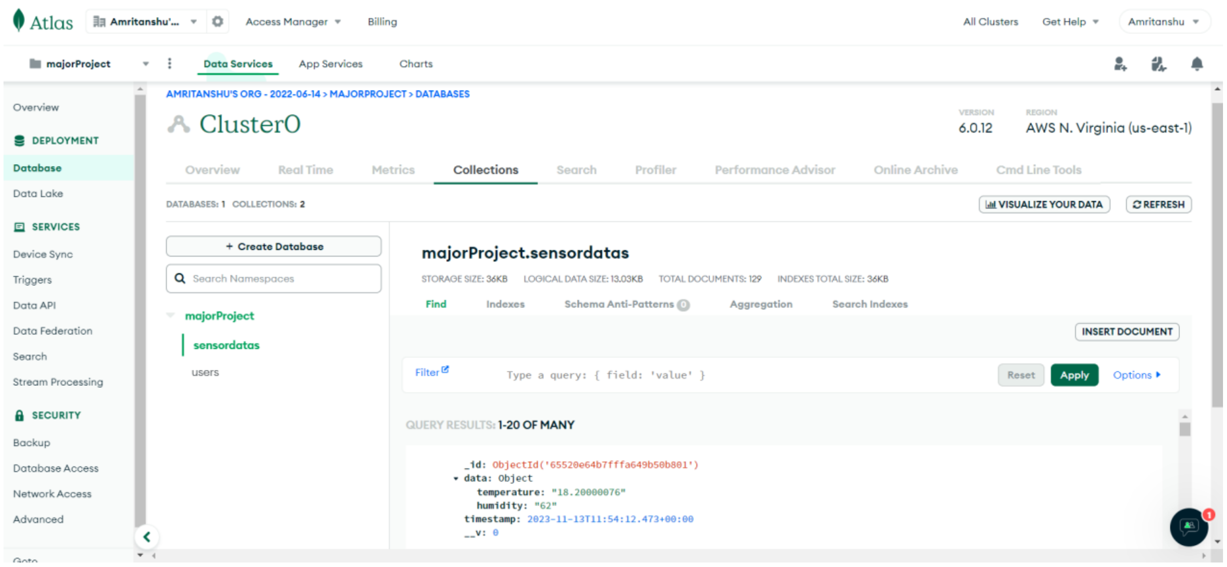
   - User Authentication APIs: These APIs handle user login and signup processes, validating credentials and managing user sessions.
   - Real-time Data APIs: These APIs retrieve the latest temperature and humidity data from the database, ensuring the Live Data Page displays real-time environmental information.
   - Predicted Data APIs: These APIs fetch the predicted temperature and humidity values for upcoming days, providing the Predicted Data Page with accurate forecasts.

- Data Updates APIs: These APIs handle updating the database with new sensor readings and machine learning predictions, keeping the data synchronized across the system.

The web application's design prioritizes user experience by providing a clear and intuitive interface, seamless user authentication, real-time data visualization, and informative predictions. The integration of REST APIs ensures efficient data exchange and scalability, enabling the application to handle growing user demand and data volume effectively.

3.3 Data Preparation

The data preparation process involves collecting data from the DHT11 sensor, transmitting it to the server, and storing it in the MongoDB database. This process lays the foundation for subsequent analysis and prediction tasks.



majorProject.sensordatas

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 13.03KB TOTAL DOCUMENTS: 129 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS: 1-20 OF MANY

```
  _id: ObjectId('65520e64b7fffa649b50b801')
▾ data: Object
    temperature: "18.20000076"
    humidity: "62"
  timestamp: 2023-11-13T11:54:12.473+00:00
  __v: 0
```

Fig. 3: MongoDB database

1. Database Setup on MongoDB:

The first step in the data preparation process involved setting up a MongoDB database. MongoDB is a NoSQL database that stores data in a flexible, JSON-like format, making it suitable for handling diverse data types. The choice of MongoDB suggests a scalable and efficient solution for handling data from the DHT11 sensor.

2. Server Creation using Node.js and Express.js:

To facilitate communication between the database and the sensor, a server was created using Node.js and Express.js. Node.js is a JavaScript runtime that enables server-side programming, while Express.js is a web application framework for Node.js. This combination allows for the creation of robust and scalable server applications.

3. Endpoint Creation:

An endpoint was established on the server to handle incoming data from the DHT11 sensor. This endpoint acts as a specific URL that the sensor uses to communicate with the server. The use of endpoints ensures a structured and organized approach to handling different types of requests.

4. Data Transmission from DHT11 Sensor:

The NodeMcu ESP8266, configured using the Arduino IDE, serves as the intermediary between the physical DHT11 sensor and the server. The Arduino IDE provides a platform for programming the ESP8266 to transmit data in JSON format. JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for both humans to read and write and machines to parse and generate.

5. JSON Format for Data Transmission:

The decision to transmit data in JSON format is strategic. JSON provides a standardized and easy-to-read structure for data, facilitating seamless communication between different components of the system. The format likely includes key-value pairs representing different parameters collected by the DHT11 sensor, such as temperature and humidity.

6. Server-Side Handling of Data:

Upon receiving the JSON-formatted data at the designated endpoint, the server, implemented with Express.js, processes the incoming payload. This involves parsing the JSON data, extracting relevant information, and preparing it for storage in the MongoDB database.

7. Database Integration:

The processed data is then added to the MongoDB database. MongoDB's document-oriented model allows for flexible and dynamic schemas, accommodating variations in the incoming data. This adaptability is particularly useful when dealing with sensor data, which may have diverse attributes.

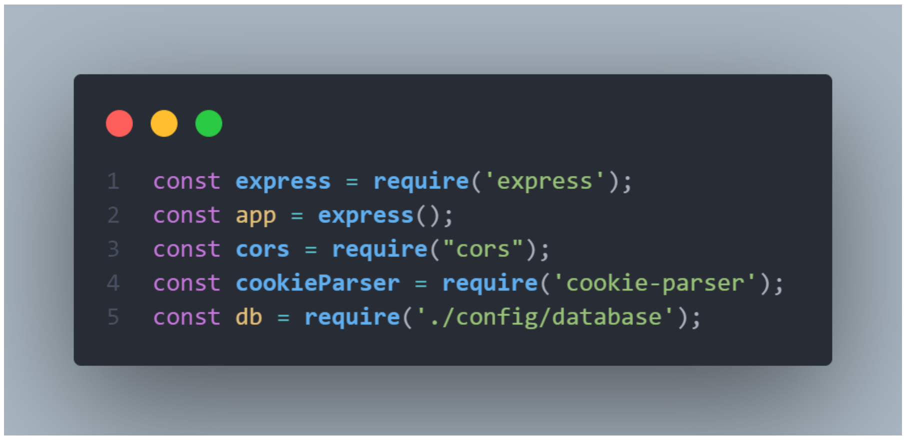
8. Data Security and Validation:

In a production environment, it is crucial to implement security measures and data validation to ensure the integrity and authenticity of the incoming data. This may involve encryption for secure data transmission and validation checks on the server side to filter out invalid or malicious data.

3.4 Implementation:

3.4.1 Backend:

Import and install all dependencies



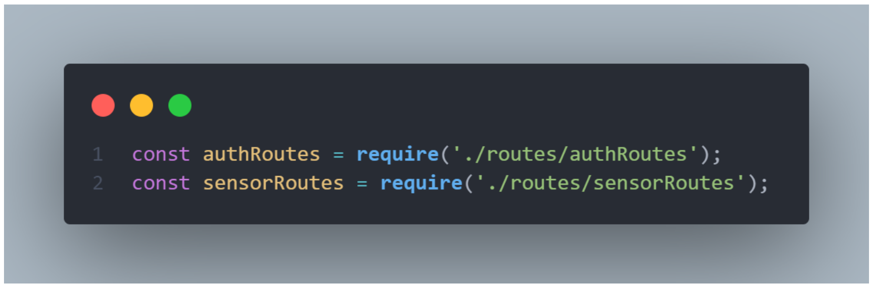
```
const express = require('express');
const app = express();
const cors = require("cors");
const cookieParser = require('cookie-parser');
const db = require('./config/database');
```

Fig. 4: Installing dependencies for backend

- **Express:** The server is built using Express.js, a web application framework for Node.js.

- **CORS (Cross-Origin Resource Sharing):** CORS is enabled using the cors middleware, allowing your server to handle requests from a different origin (e.g., your frontend on http://localhost:3000).
- **Cookie Parser:** The cookie-parser middleware is used to parse cookies attached to incoming requests.
- **Database Configuration:** The db constant likely includes the configuration for connecting to your MongoDB database.

Defining Routes

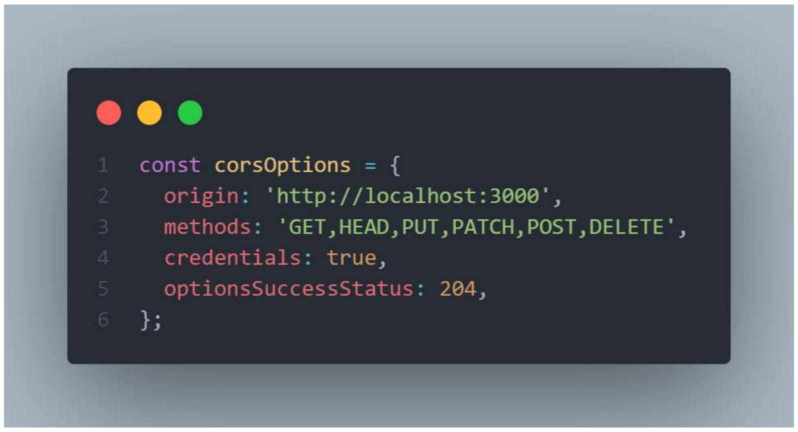


```
const authRoutes = require('./routes/authRoutes');
const sensorRoutes = require('./routes/sensorRoutes');
```

Fig. 5: Defining Routes

- Two sets of routes are defined: authRoutes for authentication-related endpoints and sensorRoutes for handling sensor data.

CORS Configuration

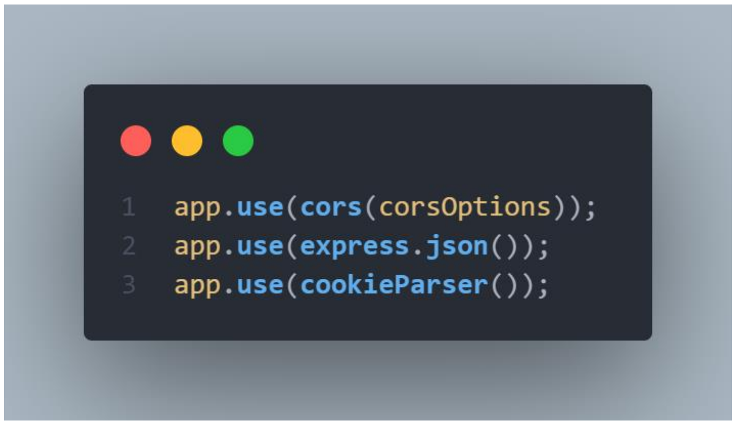


```
const corsOptions = {
  origin: 'http://localhost:3000',
  methods: 'GET,HEAD,PUT,PATCH,POST,DELETE',
  credentials: true,
  optionsSuccessStatus: 204,
};
```

Fig. 6: CORS Configuration Setup

- CORS is configured to allow requests from http://localhost:3000, the assumed origin of your frontend.
- The specified HTTP methods are permitted, and credentials (e.g., cookies) are allowed to be included in cross-origin requests.

Middleware Setup

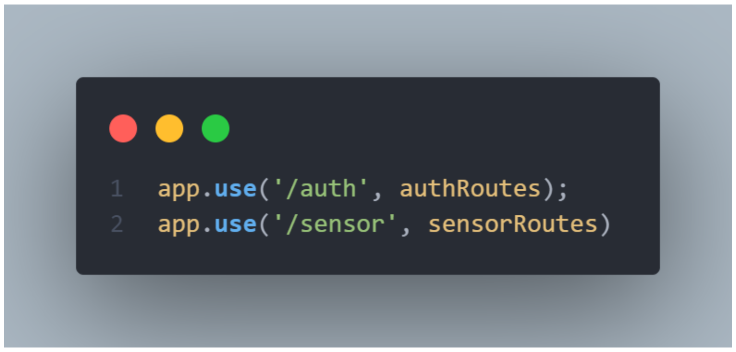


```
app.use(cors(corsOptions));
app.use(express.json());
app.use(cookieParser());
```

Fig. 7: Middleware setup for backend

- Middleware is set up to parse incoming JSON data (express.json()) and cookies (cookieParser()).

Route Usage

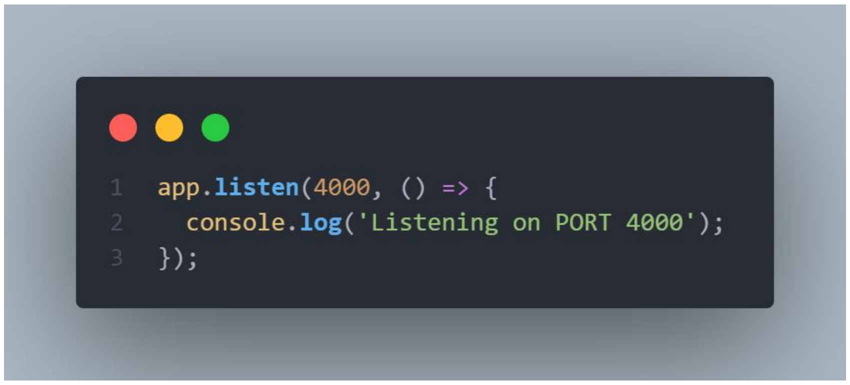


```
app.use('/auth', authRoutes);
app.use('/sensor', sensorRoutes)
```

Fig. 8: Route Handlers

- The application uses the defined route handlers. Requests to /auth will be handled by the authRoutes, and requests to /sensor will be handled by the sensorRoutes.

Server Start

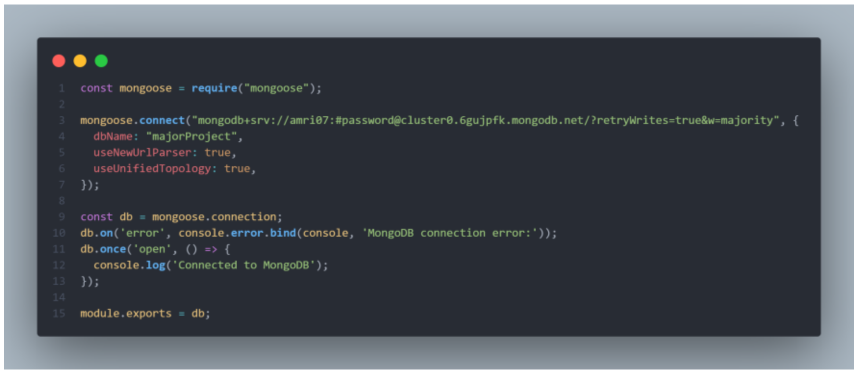


```javascript
app.listen(4000, () => {
  console.log('Listening on PORT 4000');
});
```

Fig. 9: Code snippet for Server Start

- The server is set to listen on port 4000.

Database integration

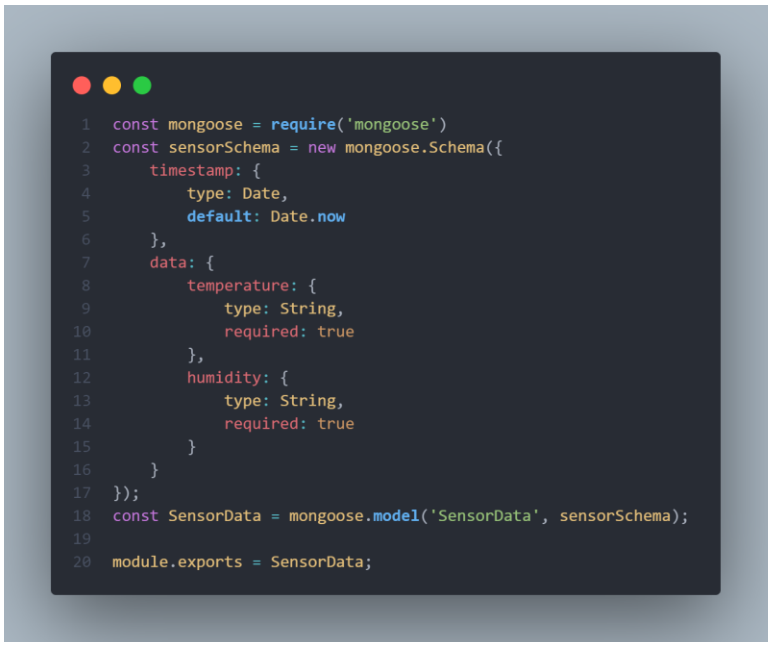


```javascript
const mongoose = require("mongoose");

mongoose.connect("mongodb+srv://amri07:#password@cluster0.6gujpfk.mongodb.net/?retryWrites=true&w=majority", {
  dbName: "majorProject",
  useNewUrlParser: true,
  useUnifiedTopology: true,
});

const db = mongoose.connection;
db.on('error', console.error.bind(console, 'MongoDB connection error:'));
db.once('open', () => {
  console.log('Connected to MongoDB');
});

module.exports = db;
```

Fig. 10: MongoDB integration

- Mongoose is used to connect to MongoDB.
- Connection string includes credentials and cluster details.
- Connection events are handled for error logging and success messaging.
- The established database connection (db) is exported for use in the application.

Sensor Schema

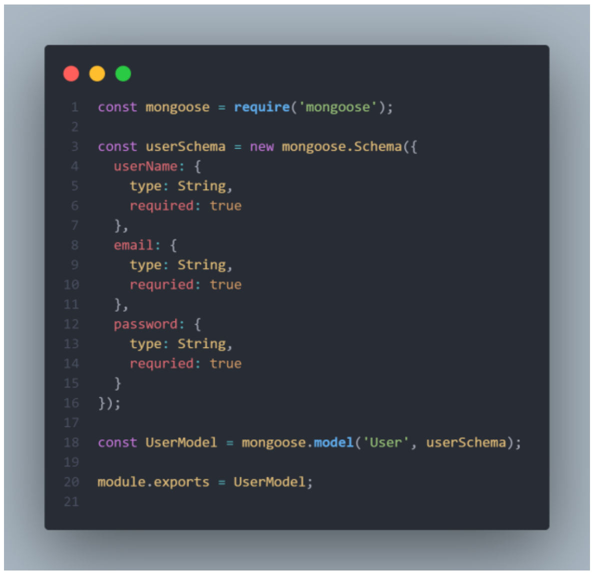


```javascript
const mongoose = require('mongoose')
const sensorSchema = new mongoose.Schema({
    timestamp: {
        type: Date,
        default: Date.now
    },
    data: {
        temperature: {
            type: String,
            required: true
        },
        humidity: {
            type: String,
            required: true
        }
    }
});
const SensorData = mongoose.model('SensorData', sensorSchema);

module.exports = SensorData;
```

Fig. 11: Sensor Data Model

- This JavaScript file defines a Mongoose schema (sensorSchema) for sensor data.
- The schema includes fields for timestamp, temperature, and humidity.
- The model represents a collection in the MongoDB database.

User Schema

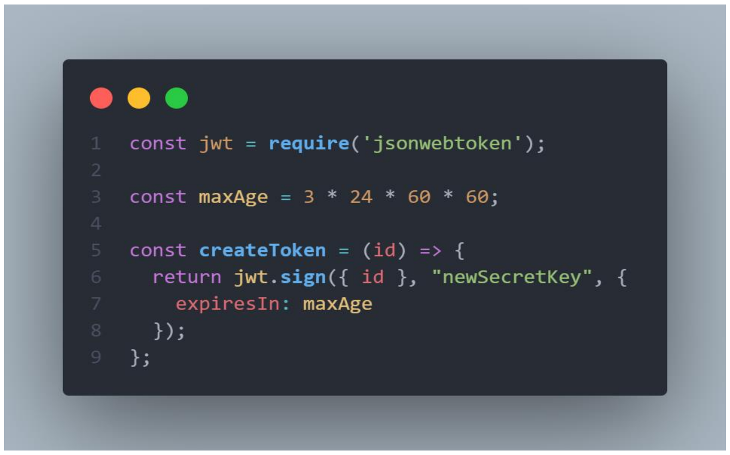


```javascript
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  userName: {
    type: String,
    required: true
  },
  email: {
    type: String,
    requried: true
  },
  password: {
    type: String,
    requried: true
  }
});

const UserModel = mongoose.model('User', userSchema);

module.exports = UserModel;
```

Fig. 12: User Model Schema

- This JavaScript file defines a Mongoose schema (userSchema) for user data with fields for userName, email, and password. It then creates a Mongoose model based on this schema, representing a collection in the MongoDB database for user information
- The model represents a collection in the MongoDB database for user information.

JWT for Creating Token

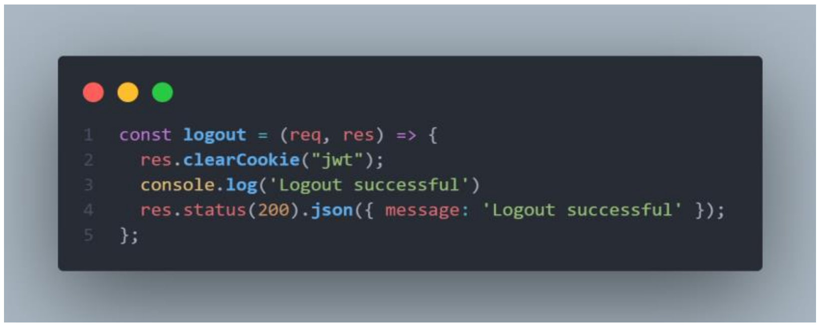


```
const jwt = require('jsonwebtoken');

const maxAge = 3 * 24 * 60 * 60;

const createToken = (id) => {
  return jwt.sign({ id }, "newSecretKey", {
    expiresIn: maxAge
  });
};
```

Fig. 13: Creating Token

- “maxAge” is set to define the maximum duration of the token (3 days).
- The createToken function generates a JWT by signing the user's ID with a secret key and setting the expiration time to “maxAge”.

Logout Functionality



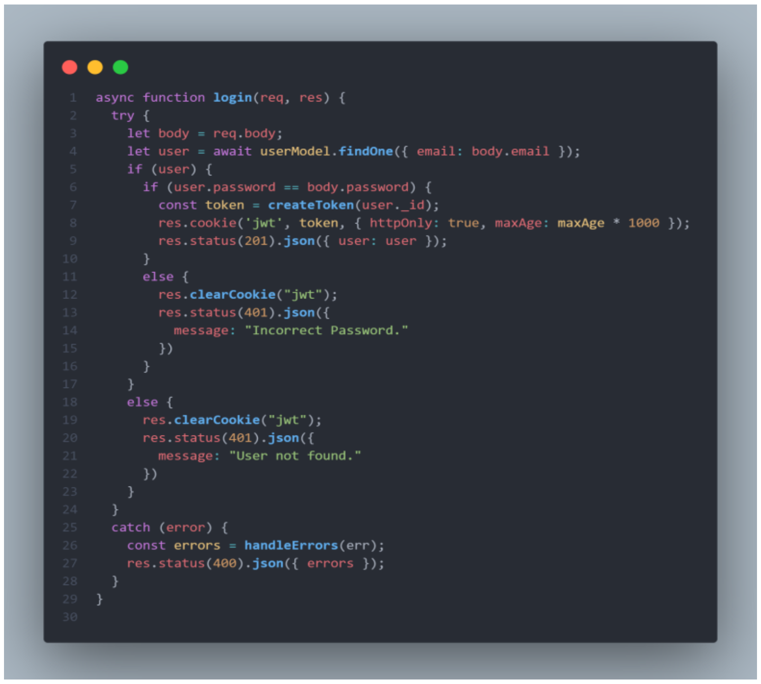
```
const logout = (req, res) => {
  res.clearCookie("jwt");
  console.log('Logout successful')
  res.status(200).json({ message: 'Logout successful' });
};
```

Fig. 14: Code snippet for Logout

- res.clearCookie("jwt") removes the JWT cookie from the response.

- res.status(200).json({ message: 'Logout successful' }) sends a JSON response to the client, indicating a successful logout with an HTTP status of 200.

Login Functionality

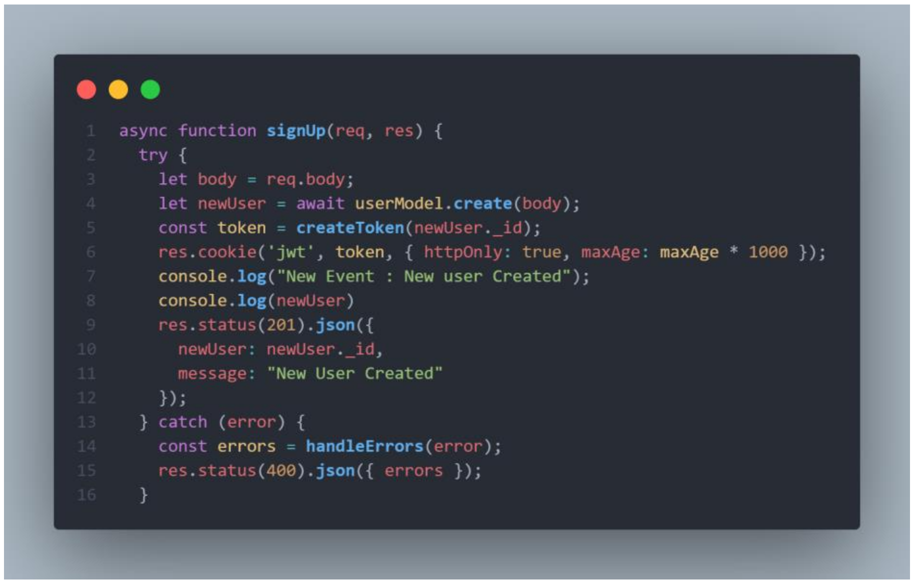


```javascript
async function login(req, res) {
  try {
    let body = req.body;
    let user = await userModel.findOne({ email: body.email });
    if (user) {
      if (user.password == body.password) {
        const token = createToken(user._id);
        res.cookie('jwt', token, { httpOnly: true, maxAge: maxAge * 1000 });
        res.status(201).json({ user: user });
      }
      else {
        res.clearCookie("jwt");
        res.status(401).json({
          message: "Incorrect Password."
        })
      }
    }
    else {
      res.clearCookie("jwt");
      res.status(401).json({
        message: "User not found."
      })
    }
  }
  catch (error) {
    const errors = handleErrors(err);
    res.status(400).json({ errors });
  }
}

```

Fig. 15: Code snippet for Login

- If the user exists and the password matches, a JWT is created using createToken.
- The JWT is stored in a secure HTTP-only cookie with a limited lifespan.
- If the password is incorrect, responds with a status of 401 (Unauthorized) and clears the JWT cookie.
- This code demonstrates a secure login mechanism with error handling, utilizing JWTs for authentication and cookies for secure token storage.

SignUp Functionality

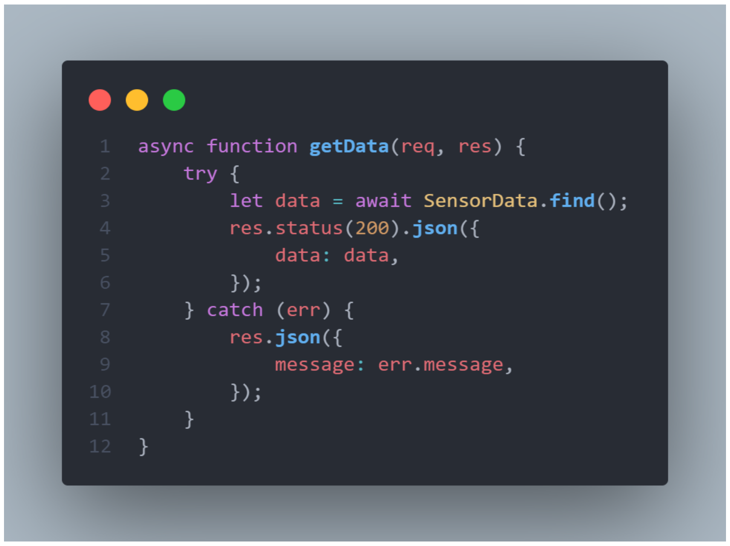


```
async function signUp(req, res) {
  try {
    let body = req.body;
    let newUser = await userModel.create(body);
    const token = createToken(newUser._id);
    res.cookie('jwt', token, { httpOnly: true, maxAge: maxAge * 1000 });
    console.log("New Event : New user Created");
    console.log(newUser)
    res.status(201).json({
      newUser: newUser._id,
      message: "New User Created"
    });
  } catch (error) {
    const errors = handleErrors(error);
    res.status(400).json({ errors });
  }
```

Fig. 16: Code snippet for SignUP

- Generates a JWT using createToken for the newly created user.
- Stores the JWT in a secure HTTP-only cookie with a specified lifespan.
- This code showcases a user registration mechanism with JWT authentication, secure token storage, and error handling for a smoother registration process.

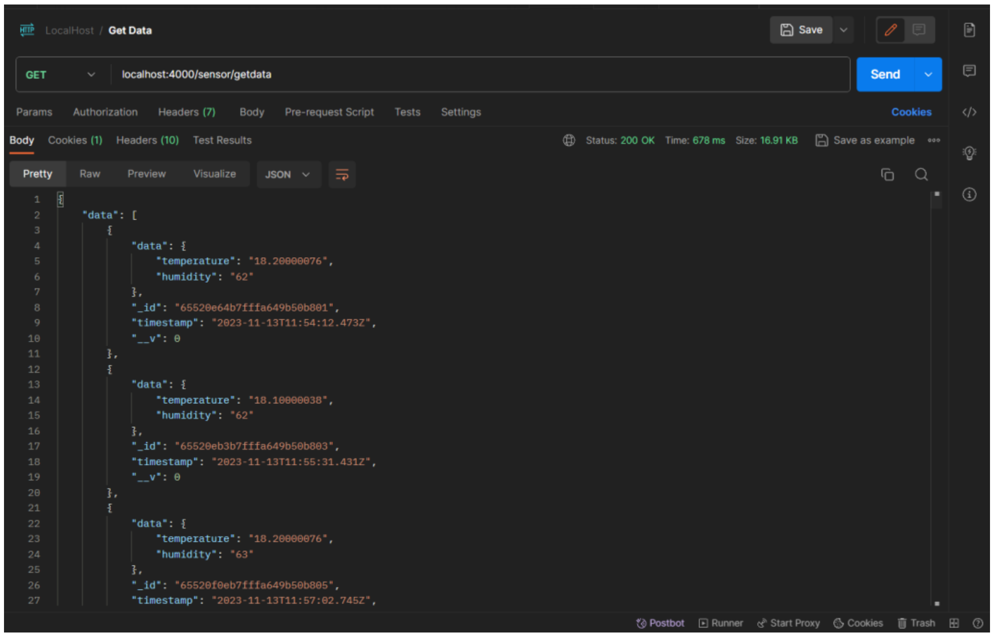
Fetching data from database



```
async function getData(req, res) {
    try {
        let data = await SensorData.find();
        res.status(200).json({
            data: data,
        });
    } catch (err) {
        res.json({
            message: err.message,
        });
    }
}
```

Fig. 17: Fetching data Code snippet

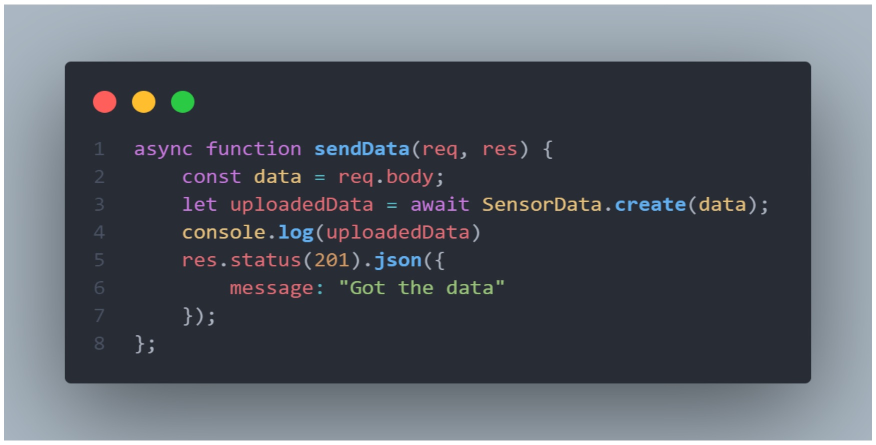
- Asynchronously queries the MongoDB database using the find method on the SensorData model to retrieve all sensor data.
- Responds with a status of 200 (OK) and a JSON object containing the retrieved sensor data.
- This code demonstrates a simple endpoint for retrieving sensor data from the MongoDB database and providing it as a JSON response.



```
{
    "data": [
        {
            "data": {
                "temperature": "18.20000076",
                "humidity": "62"
            },
            "_id": "65520e64b7fffa649b50b801",
            "timestamp": "2023-11-13T11:54:12.473Z",
            "__v": 0
        },
        {
            "data": {
                "temperature": "18.10000038",
                "humidity": "62"
            },
            "_id": "65520eb3b7fffa649b50b803",
            "timestamp": "2023-11-13T11:55:31.431Z",
            "__v": 0
        },
        {
            "data": {
                "temperature": "18.20000076",
                "humidity": "63"
            },
            "_id": "65520f0eb7fffa649b50b805",
            "timestamp": "2023-11-13T11:57:02.745Z",
```

Fig. 18: Fetching data from Endpoint using Postman

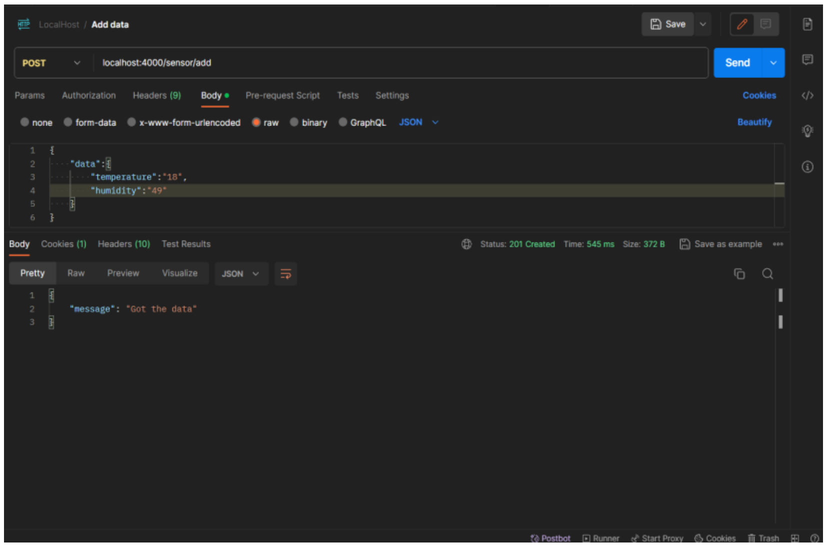
Adding new data to database



```
async function sendData(req, res) {
    const data = req.body;
    let uploadedData = await SensorData.create(data);
    console.log(uploadedData)
    res.status(201).json({
        message: "Got the data"
    });
};
```

Fig. 19: Post request Function for adding data

- This sendData function handles requests to store sensor data in the MongoDB database. It creates a new document in the SensorData collection and responds with a JSON object indicating successful data storage.



POST localhost:4000/sensor/add

```
{
    "data":{
        "temperature":"18",
        "humidity":"49"
    }
}
```

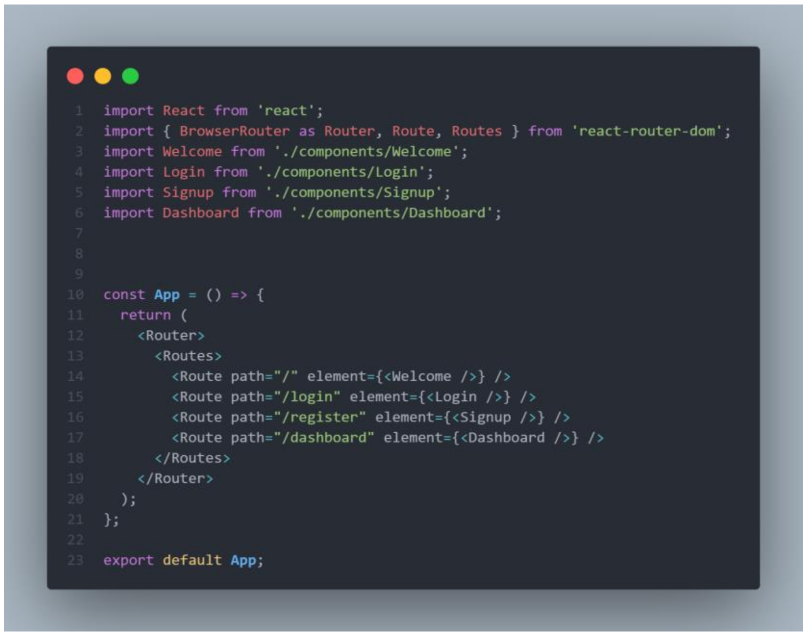
Status: 201 Created Time: 545 ms Size: 372 B

```
{
    "message": "Got the data"
}
```

Fig. 20: Adding data to database using POSTMAN

3.4.2 Frontend:

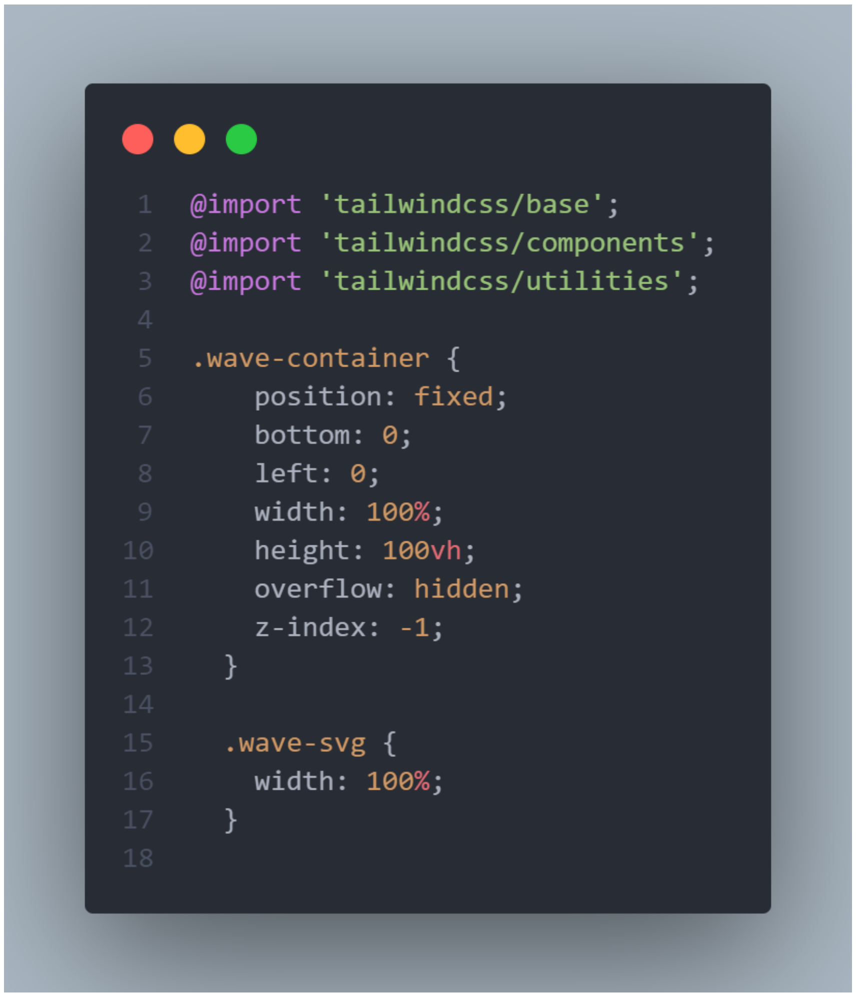
Importing libraries for ReactJS:



```
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Welcome from './components/Welcome';
import Login from './components/Login';
import Signup from './components/Signup';
import Dashboard from './components/Dashboard';



const App = () => {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<Welcome />} />
        <Route path="/login" element={<Login />} />
        <Route path="/register" element={<Signup />} />
        <Route path="/dashboard" element={<Dashboard />} />
      </Routes>
    </Router>
  );
};

export default App;
```

Fig. 21: Importing libraries and creating routes for frontend

- This React component (App) sets up routing using React Router. It defines routes for different components such as Welcome, Login, Signup, and Dashboard.
- The root path to the Welcome component.
- The "/login" path to the Login component.
- The "/register" path to the Signup component.
- The "/dashboard" path to the Dashboard component.
- The element prop specifies the React component to render when the corresponding route is accessed.
- This code sets up the basic structure for routing in a React application, allowing different components to be rendered based on the accessed URL paths.



```css
@import 'tailwindcss/base';
@import 'tailwindcss/components';
@import 'tailwindcss/utilities';

.wave-container {
    position: fixed;
    bottom: 0;
    left: 0;
    width: 100%;
    height: 100vh;
    overflow: hidden;
    z-index: -1;
  }

  .wave-svg {
    width: 100%;
  }

```

Fig. 22: Initializing Tailwind CSS

Welcome Page:

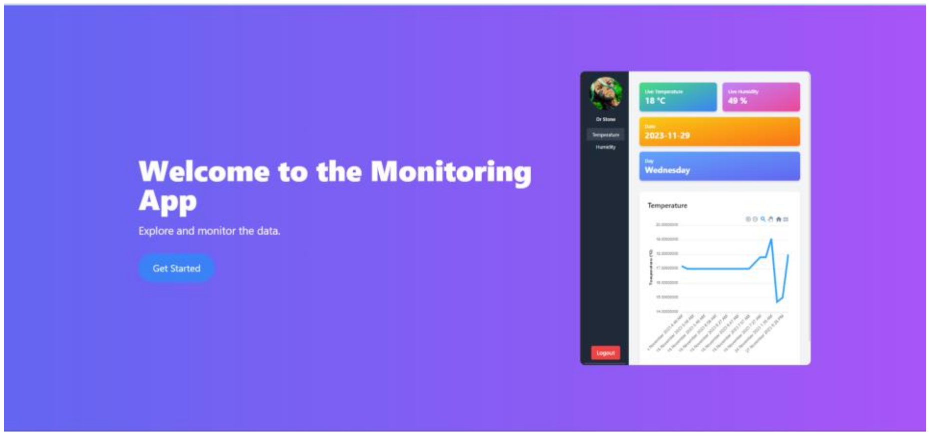
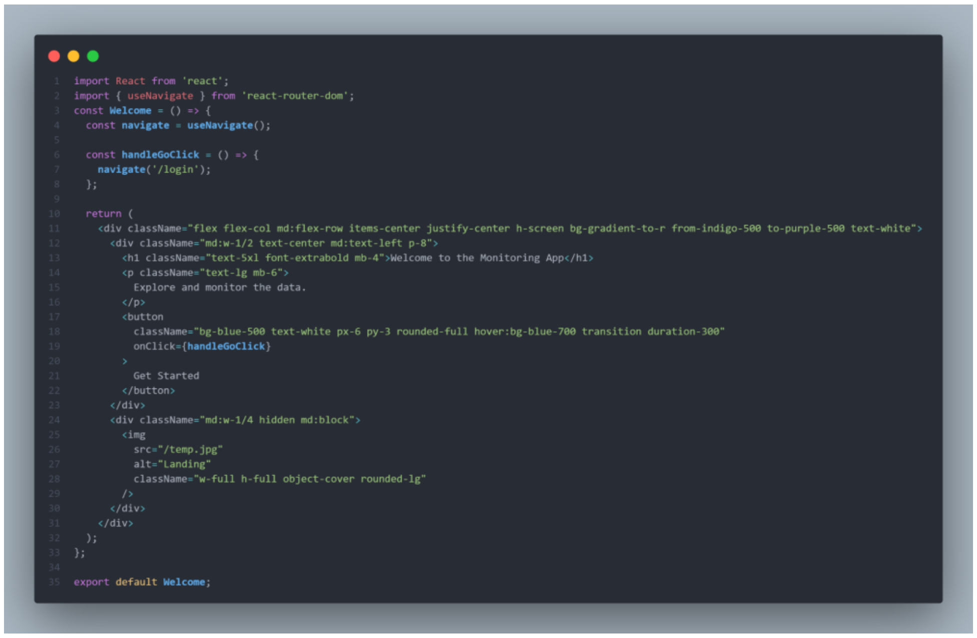


Fig. 23: Welcome Page preview

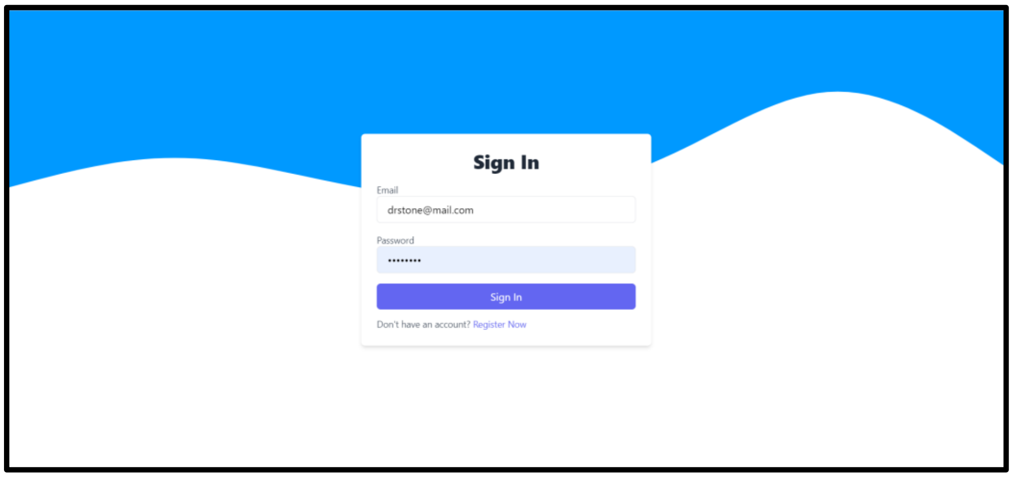


```
import React from 'react';
import { useNavigate } from 'react-router-dom';
const Welcome = () => {
  const navigate = useNavigate();

  const handleGoClick = () => {
    navigate('/login');
  };

  return (
    <div className="flex flex-col md:flex-row items-center justify-center h-screen bg-gradient-to-r from-indigo-500 to-purple-500 text-white">
      <div className="md:w-1/2 text-center md:text-left p-8">
        <h1 className="text-5xl font-extrabold mb-4">Welcome to the Monitoring App</h1>
        <p className="text-lg mb-6">
          Explore and monitor the data.
        </p>
        <button
          className="bg-blue-500 text-white px-6 py-3 rounded-full hover:bg-blue-700 transition duration-300"
          onClick={handleGoClick}
        >
          Get Started
        </button>
      </div>
      <div className="md:w-1/4 hidden md:block">
        <img
          src="/temp.jpg"
          alt="Landing"
          className="w-full h-full object-cover rounded-lg"
        />
      </div>
    </div>
  );
};

export default Welcome;
```

Fig. 24: Welcome Page code snippet

The Welcome component serves as the initial landing page for the React application, featuring a visually engaging layout. This component includes a welcoming message, a brief description, and a strategically placed “Get Started” button. Upon clicking the button, the handleGoClick function uses the useNavigate hook from React Router to seamlessly redirect users to the /login page. This deliberate redirection ensures a smooth transition to the authentication section of the application. The responsive design of the component enhances its visual appeal, adapting gracefully to various screen sizes. In essence, the Welcome component not only establishes a positive introduction to the Monitoring App but also orchestrates a user-friendly redirect, guiding users to the login page and initiating their interaction with the core features of the application.

Login Page:


Sign In

Email

drstone@mail.com

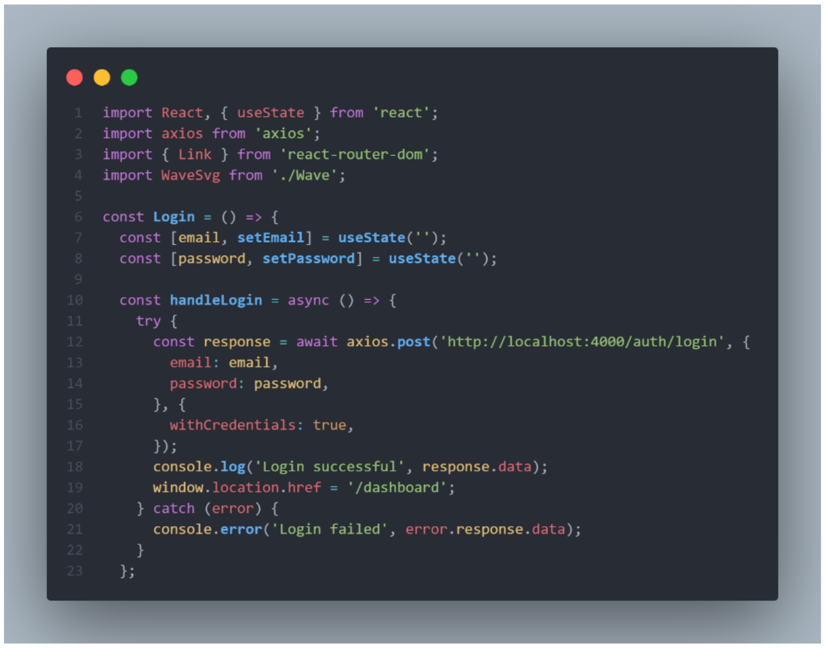
Password

••••••••

Sign In

Don't have an account? Register Now

Fig. 25: Login Page Preview



```
import React, { useState } from 'react';
import axios from 'axios';
import { Link } from 'react-router-dom';
import WaveSvg from './Wave';

const Login = () => {
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');

  const handleLogin = async () => {
    try {
      const response = await axios.post('http://localhost:4000/auth/login', {
        email: email,
        password: password,
      }, {
        withCredentials: true,
      });
      console.log('Login successful', response.data);
      window.location.href = '/dashboard';
    } catch (error) {
      console.error('Login failed', error.response.data);
    }
  };
```

Fig. 26: Login Page Code snippet

- The /auth/signup page, represented by the Signup component in the React application, facilitates user registration. This component employs React's useState hook to manage state variables for the user's email, password, and username. The core functionality is encapsulated in the handleSignup function, triggered upon clicking the "Sign Up" button. This function initiates an asynchronous POST request to the server's /auth/signup endpoint using Axios, forwarding the user's registration details. Upon successful registration, the user is seamlessly redirected to the dashboard (/dashboard). In the event of a registration failure, an error message is logged to the console. The component's structure incorporates HTML elements styled with Tailwind CSS, ensuring a visually pleasing and responsive layout. Additionally, a link to the login page (/login) is provided for users with existing accounts, contributing to a cohesive user authentication flow within the application.

SignUp Page:

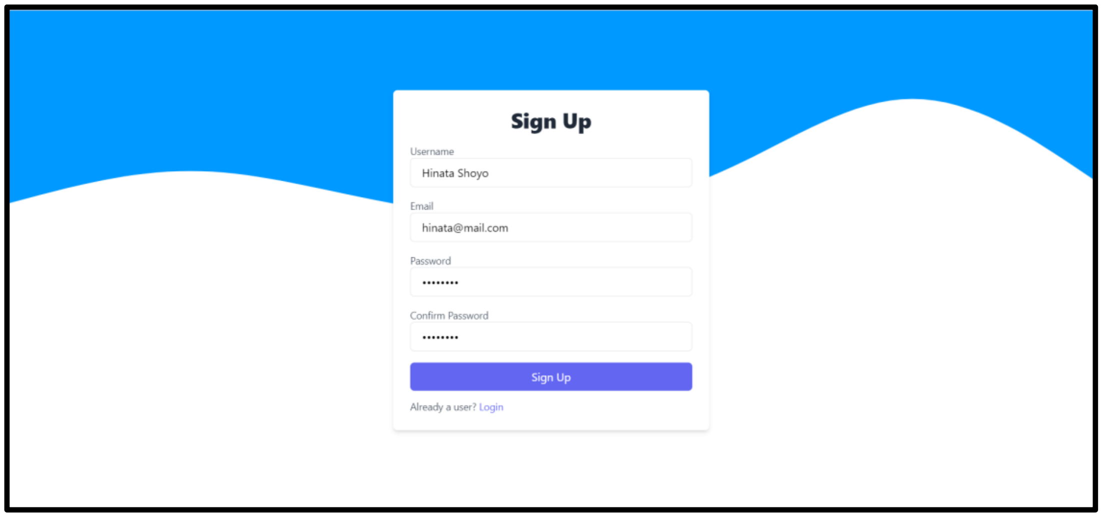
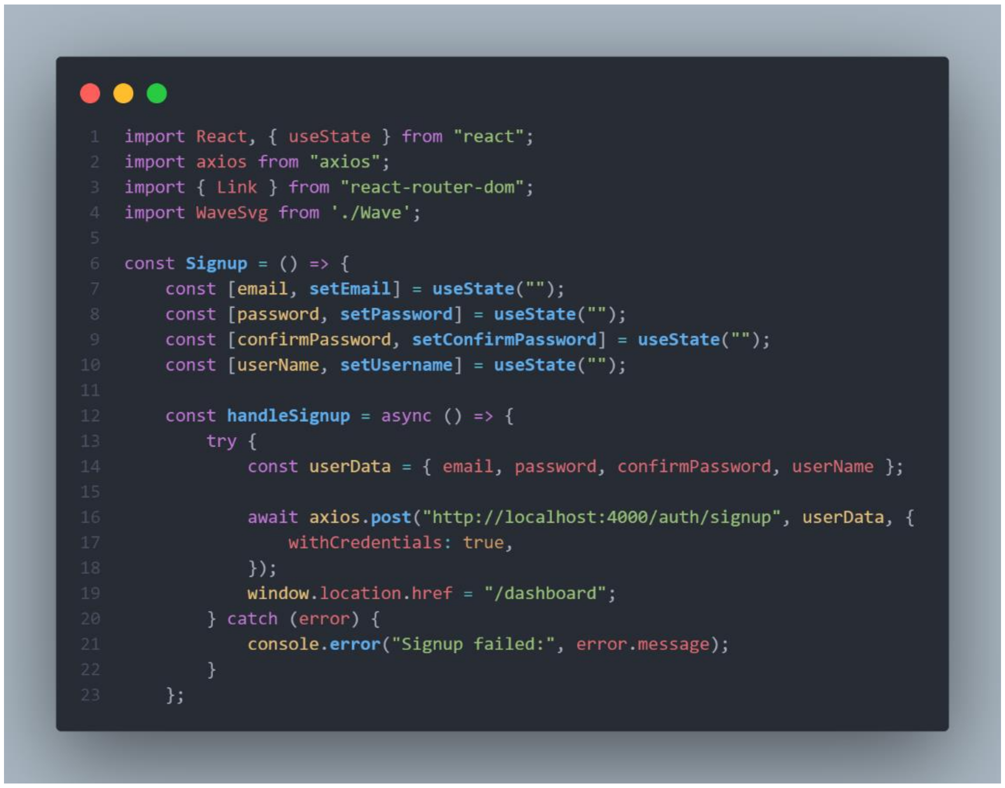


Fig. 27: SignUp page preview



```
import React, { useState } from "react";
import axios from "axios";
import { Link } from "react-router-dom";
import WaveSvg from './Wave';

const Signup = () => {
    const [email, setEmail] = useState("");
    const [password, setPassword] = useState("");
    const [confirmPassword, setConfirmPassword] = useState("");
    const [userName, setUsername] = useState("");

    const handleSignup = async () => {
        try {
            const userData = { email, password, confirmPassword, userName };

            await axios.post("http://localhost:4000/auth/signup", userData, {
                withCredentials: true,
            });
            window.location.href = "/dashboard";
        } catch (error) {
            console.error("Signup failed:", error.message);
        }
    };
```

Fig. 28: SignUp page code snippet

- The Signup component in this React application orchestrates the user interface for the sign-up page, incorporating essential functionalities for user registration. Employing React’s useState hook, it adeptly manages state variables for the user’s email, password, confirmed password, and username. The core logic resides in the handleSignup function, activated upon clicking the “Sign Up” button. This function crafts a userData object encapsulating the user’s registration details and dispatches an asynchronous POST request to the server’s /auth/signup endpoint using Axios. Upon successful registration, the user seamlessly transitions to the dashboard (/dashboard), while any registration errors prompt console logging for diagnostic purposes. Furthermore, it incorporates a custom WaveSvg component, potentially for decorative enhancements. Input fields for username, email, password, and confirmed password facilitate user input, each dynamically updating the corresponding state. A strategically placed link directs users to the login page (/login) for those with existing accounts, harmonizing the user authentication journey. Overall, the Signup component contributes to a seamless and visually appealing user registration experience within the application.

Dashboard Page:

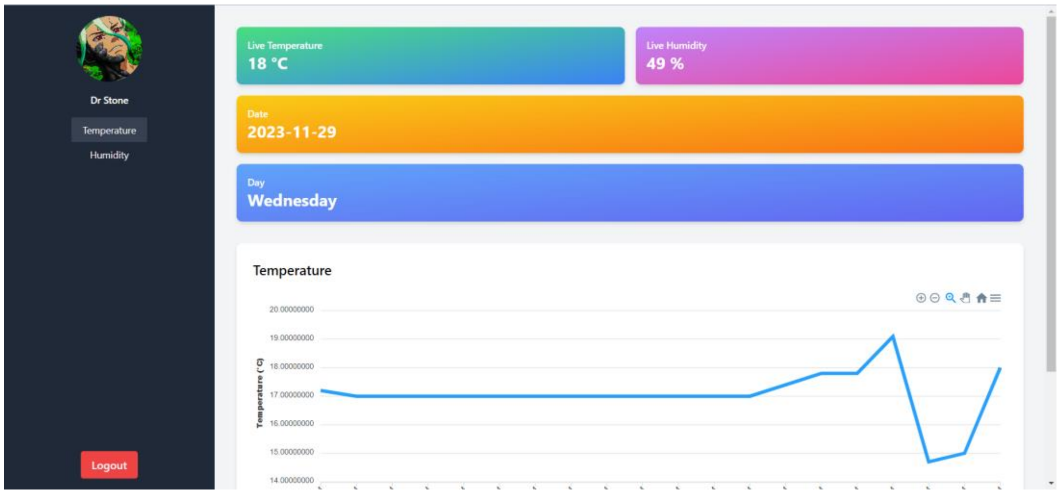
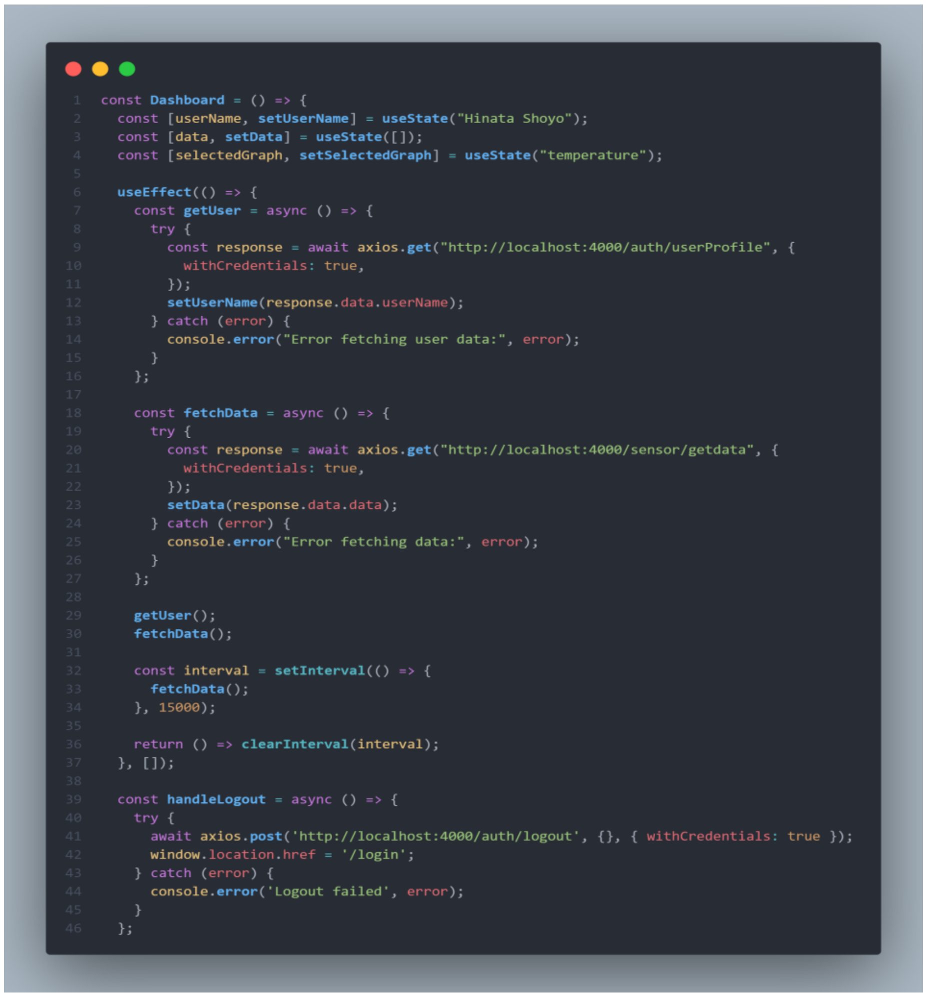
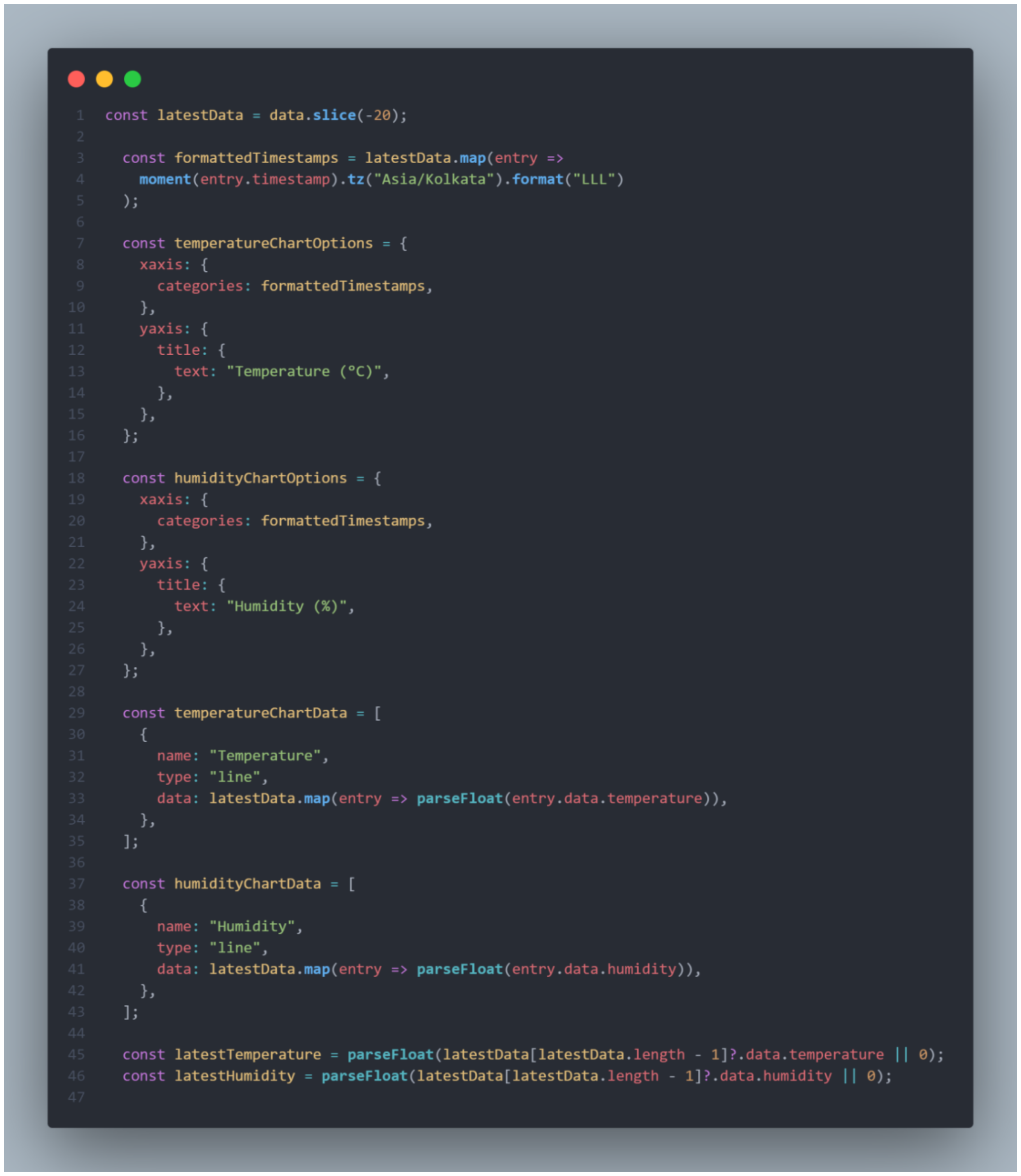


Fig. 29: Dashboard preview



```
const Dashboard = () => {
  const [userName, setUserName] = useState("Hinata Shoyo");
  const [data, setData] = useState([]);
  const [selectedGraph, setSelectedGraph] = useState("temperature");

  useEffect(() => {
    const getUser = async () => {
      try {
        const response = await axios.get("http://localhost:4000/auth/userProfile", {
          withCredentials: true,
        });
        setUserName(response.data.userName);
      } catch (error) {
        console.error("Error fetching user data:", error);
      }
    };

    const fetchData = async () => {
      try {
        const response = await axios.get("http://localhost:4000/sensor/getdata", {
          withCredentials: true,
        });
        setData(response.data.data);
      } catch (error) {
        console.error("Error fetching data:", error);
      }
    };

    getUser();
    fetchData();

    const interval = setInterval(() => {
      fetchData();
    }, 15000);

    return () => clearInterval(interval);
  }, []);

  const handleLogout = async () => {
    try {
      await axios.post('http://localhost:4000/auth/logout', {}, { withCredentials: true });
      window.location.href = '/login';
    } catch (error) {
      console.error('Logout failed', error);
    }
  };
```

Fig. 30: Data fetching code snippet for dashboard



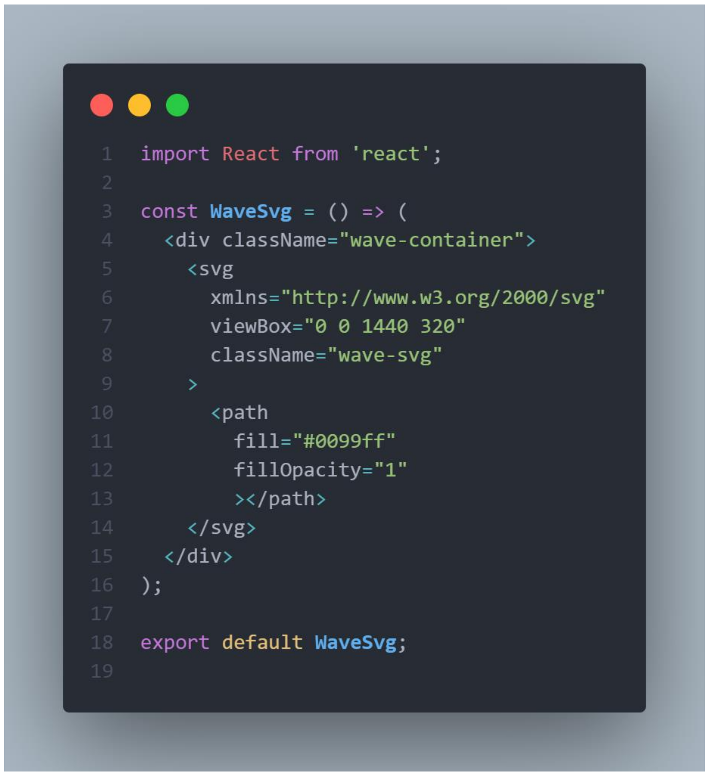
```
const latestData = data.slice(-20);

  const formattedTimestamps = latestData.map(entry =>
    moment(entry.timestamp).tz("Asia/Kolkata").format("LLL")
  );

  const temperatureChartOptions = {
    xaxis: {
      categories: formattedTimestamps,
    },
    yaxis: {
      title: {
        text: "Temperature (°C)",
      },
    },
  };

  const humidityChartOptions = {
    xaxis: {
      categories: formattedTimestamps,
    },
    yaxis: {
      title: {
        text: "Humidity (%)",
      },
    },
  };

  const temperatureChartData = [
    {
      name: "Temperature",
      type: "line",
      data: latestData.map(entry => parseFloat(entry.data.temperature)),
    },
  ];

  const humidityChartData = [
    {
      name: "Humidity",
      type: "line",
      data: latestData.map(entry => parseFloat(entry.data.humidity)),
    },
  ];

  const latestTemperature = parseFloat(latestData[latestData.length - 1]?.data.temperature || 0);
  const latestHumidity = parseFloat(latestData[latestData.length - 1]?.data.humidity || 0);
```

Fig. 31: Apex-Charts integration in dashboard

- This React component serves as a dashboard for monitoring temperature and humidity data obtained from a sensor. The component utilizes various libraries such as Axios for making asynchronous HTTP requests, moment-timezone for handling time zones, and “react-apexcharts” for rendering interactive and visually appealing line charts.The dashboard is designed with a user-friendly interface, featuring live readings of temperature and humidity prominently displayed in gradient-colored boxes. Additionally, there are sections displaying the current date and day, both formatted for the Asia/Kolkata time zone. The user’s profile picture and name are also shown, fetched from the server upon component mount. To ensure real-time data updates, the component fetches data from the server at regular intervals (every 15 seconds) using the setInterval function. This periodic data fetching helps keep the displayed information current and relevant.

- The React component communicates with the server through the “/getdata” route, making use of Axios for HTTP GET requests. The server, implemented with Express.js, has a corresponding route defined to handle requests for fetching sensor data. This route is linked to a controller (or middleware), such as “sensorController,” which, in turn, interacts with a model (e.g., “Sensor”) to retrieve data from a database (assuming MongoDB with Mongoose).

Waves.jsx file:



```
import React from 'react';

const WaveSvg = () => (
  <div className="wave-container">
    <svg
      xmlns="http://www.w3.org/2000/svg"
      viewBox="0 0 1440 320"
      className="wave-svg"
    >
      <path
        fill="#0099ff"
        fillOpacity="1"
        ></path>
    </svg>
  </div>
);

export default WaveSvg;

```

Fig. 32: Wave design code snippet

- This component is used to provide a wave-like effect on the background of the elements, to make the design of the website more appealing.

3.4.3 Arduino IDE:

In this the Arduino IDE acts as a major component in feeding the NodeMCU ESP8266 with the code and how it has to work.

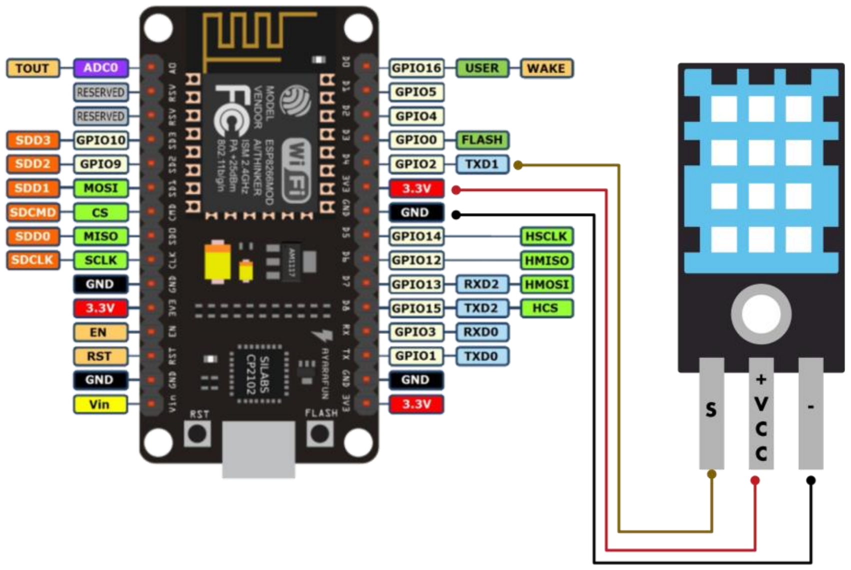
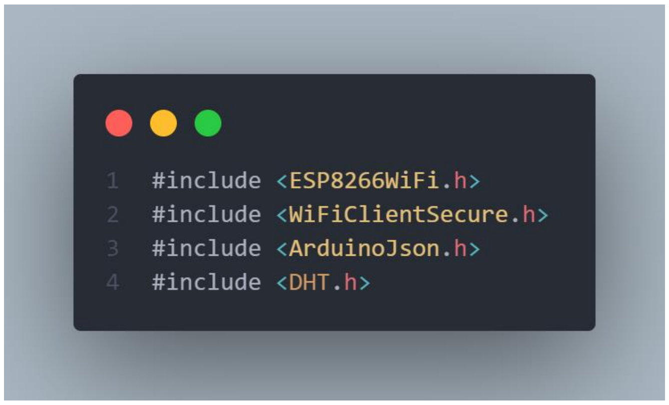


Fig. 33: Interfacing DHT-11 sensor with nodemcu wifi module

Including Libraries:



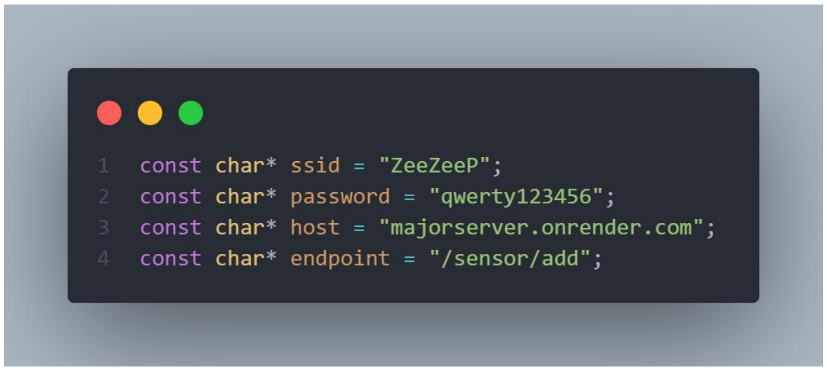
```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <ArduinoJson.h>
#include <DHT.h>
```

Fig. 34: Library import for Nodemcu ESP8266

- ESP8266WiFi: Provides functions for connecting the ESP8266 to a Wi-Fi network.
- WiFiClientSecure: Allows secure connections to a server over HTTPS.
- ArduinoJson: A library for handling JSON data.

- DHT: Library for working with DHT temperature and humidity sensors.

Wifi Configuration, host and Endpoint:

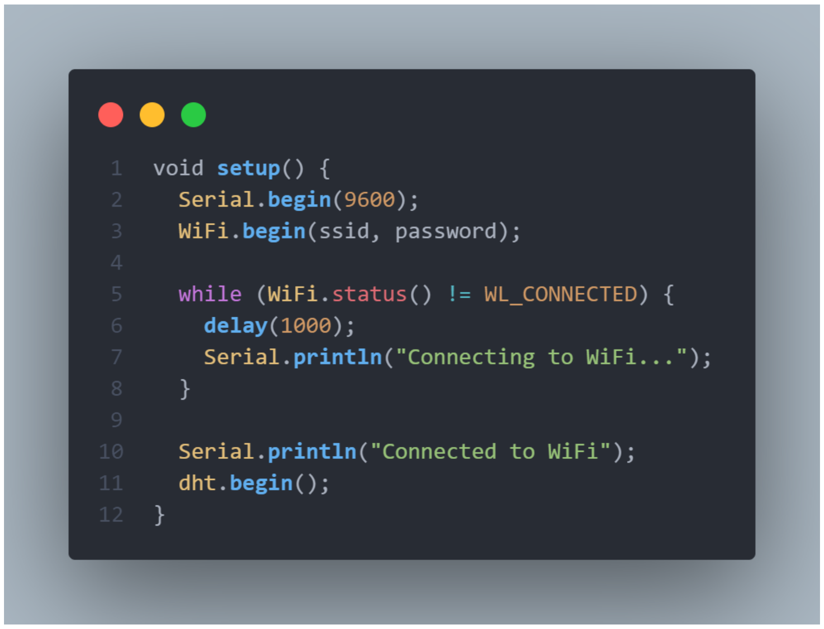


```
const char* ssid = "ZeeZeeP";
const char* password = "qwerty123456";
const char* host = "majorserver.onrender.com";
const char* endpoint = "/sensor/add";
```

Fig. 35: Wifi Configuration

- SSID and Password: WiFi network credentials.
- Host: The address of the server to which data will be sent.
- Endpoint: The specific API endpoint on the server for receiving sensor data.

Initialization (Setup):



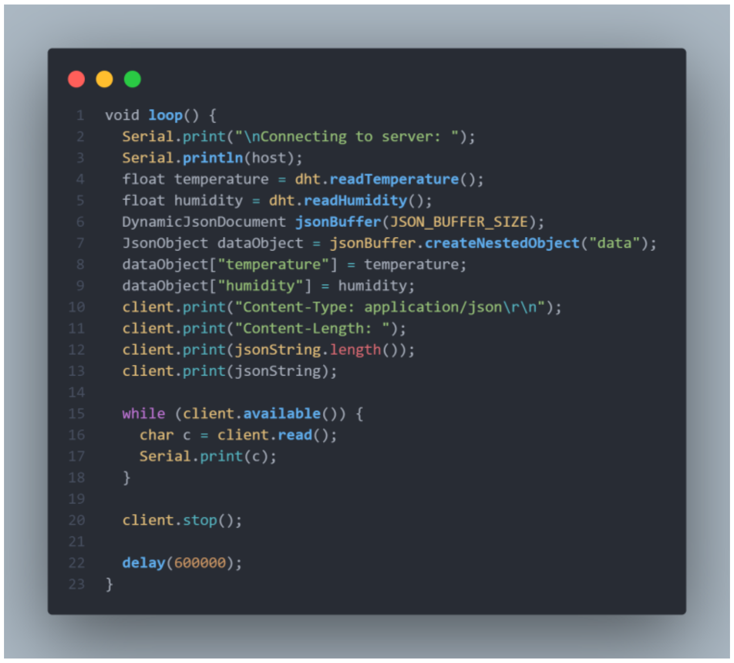
```
void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }

  Serial.println("Connected to WiFi");
  dht.begin();
}
```

Fig. 36: Setup code for Nodemcu

- Serial communication is initiated for debugging purposes.

- ESP8266 attempts to connect to the specified Wi-Fi network.
- Once connected, it prints a message indicating the successful connection.
- The DHT sensor is initialized.

Main Loop:



```
void loop() {
  Serial.print("\nConnecting to server: ");
  Serial.println(host);
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  DynamicJsonDocument jsonBuffer(JSON_BUFFER_SIZE);
  JsonObject dataObject = jsonBuffer.createNestedObject("data");
  dataObject["temperature"] = temperature;
  dataObject["humidity"] = humidity;
  client.print("Content-Type: application/json\r\n");
  client.print("Content-Length: ");
  client.print(jsonString.length());
  client.print(jsonString);

  while (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  client.stop();

  delay(600000);
}
```

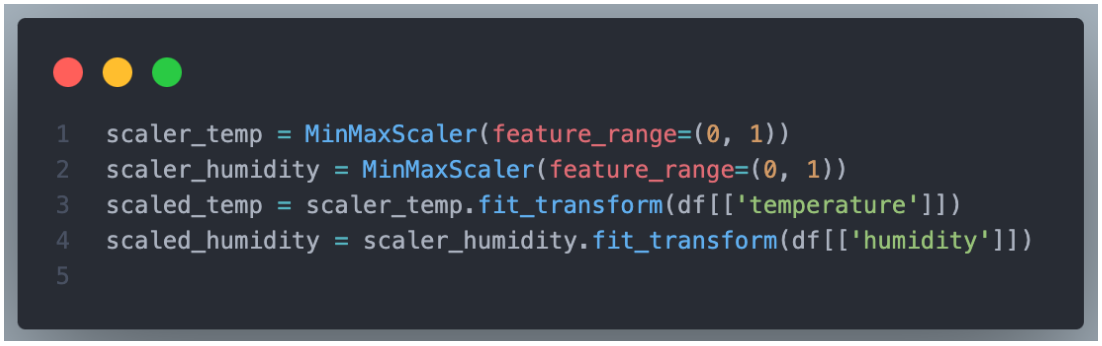
Fig. 37: Main loop file

- Reads temperature and humidity from the DHT sensor.
- Creates a JSON payload using the ArduinoJSON library.
- Sends an HTTPS POST request to the specified endpoint with the JSON payload.
- Delays for 10 minutes (600,000 milliseconds) before repeating the process.

3.4.4 ML Integration:

The ML integration used a Long Short-Term Memory (LSTM) neural network model to provide temperature and humidity predictions for the next three days based on past sensor data. The process entailed several stages such as data preprocessing, model building, evaluation, and prediction.

Data Preprocessing with Min-Max scaling:

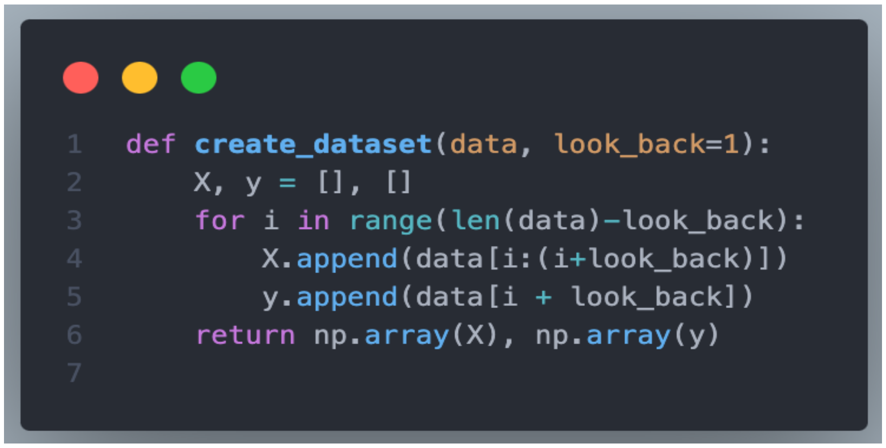


```python
scaler_temp = MinMaxScaler(feature_range=(0, 1))
scaler_humidity = MinMaxScaler(feature_range=(0, 1))
scaled_temp = scaler_temp.fit_transform(df[['temperature']])
scaled_humidity = scaler_humidity.fit_transform(df[['humidity']])
```

Fig. 38: Scales the Temp/Humidity features with Min-Max Scaling

This scales the temperature and humidity features using Min-Max scaling, which transforms the data into a specified range (usually between 0 and 1) to ensure consistency and convergence during model training.

Creating Dataset for LSTM:

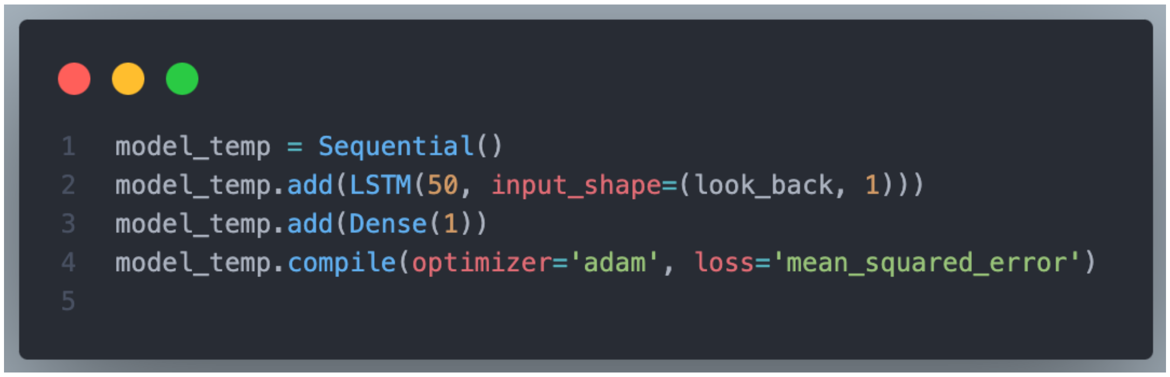


```python
def create_dataset(data, look_back=1):
    X, y = [], []
    for i in range(len(data)-look_back):
        X.append(data[i:(i+look_back)])
        y.append(data[i + look_back])
    return np.array(X), np.array(y)
```

Fig. 39: Create Dataset function for reshaping the data.

This function creates input-output pairs for the LSTM model by splitting the data into input sequences (X) and corresponding output values (y). The look_back parameter determines the number of previous time steps to consider as input.

Defining LSTM Model:

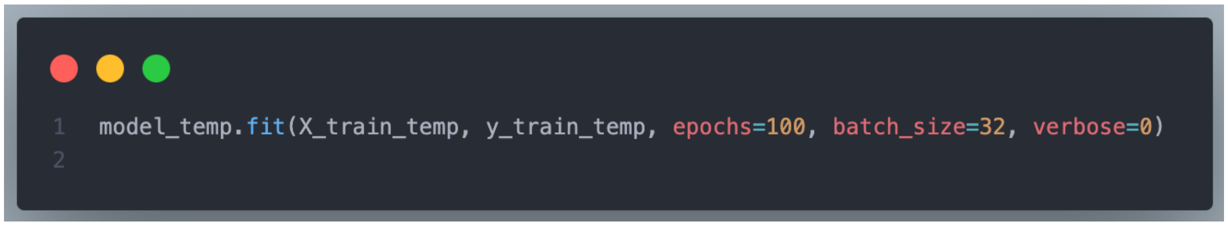


```python
model_temp = Sequential()
model_temp.add(LSTM(50, input_shape=(look_back, 1)))
model_temp.add(Dense(1))
model_temp.compile(optimizer='adam', loss='mean_squared_error')
```

Fig. 40: LSTM model with 50 units LSTM layers.

It defines an LSTM model for temperature and humidity prediction. The model architecture consists of an LSTM layer with 50 units followed by a dense output layer. The model is compiled with the Adam optimizer and mean squared error loss function.

Model Training:

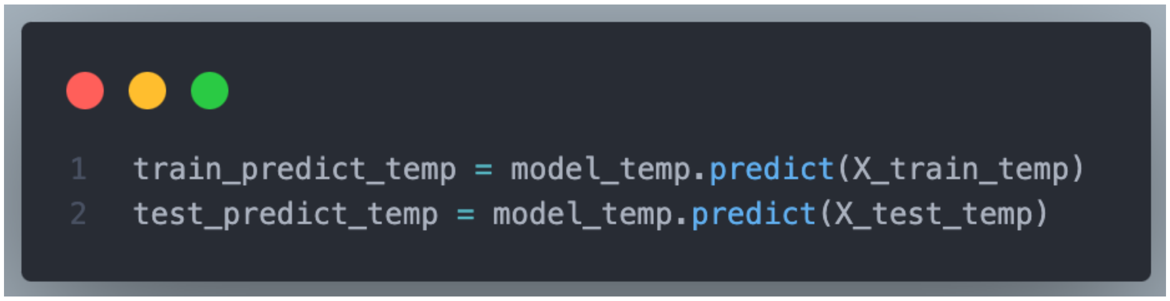


```python
model_temp.fit(X_train_temp, y_train_temp, epochs=100, batch_size=32, verbose=0)
```

Fig 41. Training the LSTM Model

It trains the LSTM model for temperature prediction using the training data (X_train_temp and y_train_temp) for 100 epochs with a batch size of 32. The verbose=0 parameter suppresses the training progress output.

Making Predictions:

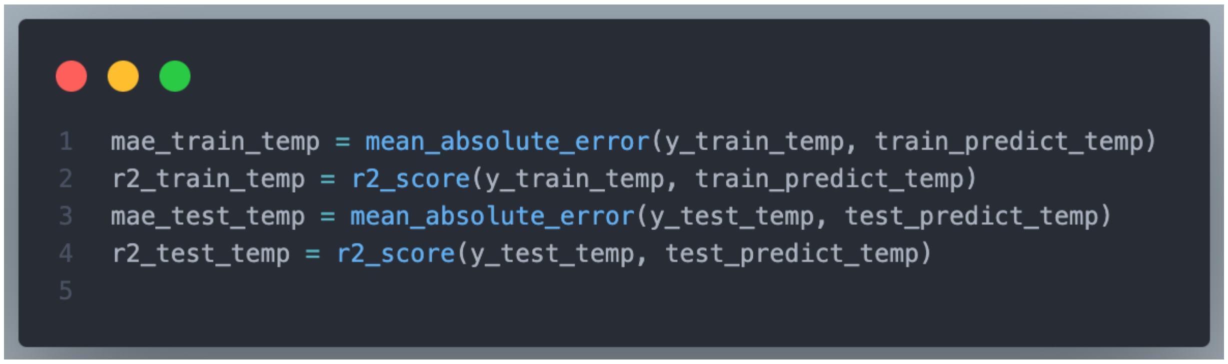


```
train_predict_temp = model_temp.predict(X_train_temp)
test_predict_temp = model_temp.predict(X_test_temp)
```

Fig. 42: Temperature predictions

It makes predictions using the trained LSTM model for temperature. train_predict_temp and test_predict_temp contain the predicted temperature values for the training and testing datasets, respectively.

Evaluation Metrics:



```
mae_train_temp = mean_absolute_error(y_train_temp, train_predict_temp)
r2_train_temp = r2_score(y_train_temp, train_predict_temp)
mae_test_temp = mean_absolute_error(y_test_temp, test_predict_temp)
r2_test_temp = r2_score(y_test_temp, test_predict_temp)

```

Fig. 43: MAE and R-Squared for temperature prediction

We compute the Mean Absolute Error (MAE) and R-squared Score for the temperature predictions. The mean_absolute_error function calculates the average magnitude of errors between the predicted (train_predict_temp and test_predict_temp) and actual (y_train_temp and y_test_temp) temperature values. The r2_score function computes the R-squared Score, which indicates the proportion of variance in the temperature data that is predictable from the input features.

3.5 Key Challenges:

1. **CORS**

   Challenge: I have ensured my frontend can make requests to backend without running into CORS issues.

   Solution: I have already configured CORS using the cors middleware. Ensuring that the specified origin and methods align with your frontend requirements. Test thoroughly to avoid unexpected CORS-related problems.

2. **Database connectivity and configuration**

   Challenge: Establishing a reliable connection to the database (db module), handling connection errors, and ensuring that the database schema is properly set up.

   Solution: Checking the db module is correctly configured and also implemented error handling for database connection failures and ensuring that the database schema is in sync with your application requirements.

3. **Authentication and Authorization**

   Challenge: Implementing a secure authentication mechanism and ensuring that routes are properly protected.

   Solution: Utilize established authentication methods (e.g., JWT, OAuth), validate user input, and enforce proper authorization checks for protected routes.

4. **Scalability**

   Challenge: Ensuring that the application can handle increased traffic and data volume.

   Solution: Design our application with scalability in mind. Consider using load balancing, optimizing database queries, and caching where appropriate. Monitor performance and scale resources as needed.

5. **Middleware order and execution**

   Challenge: Understanding the order of middleware execution and ensuring that middleware functions are applied in the correct sequence.

   Solution: Middleware's are executed in the order they are defined, so the sequence matters.

CHAPTER 4: TESTING

4.1 Testing Strategy:

1. **Unit Testing**

   Objective: Validate the correctness of individual functions, route handlers, and middleware.

   Tools: Postman (for route handlers).

2. **Integration Testing**

   Objective: Validate the interaction between different components of the application, including API endpoints and database interactions.

   Tools: Postman (for API endpoints), Mock Database for testing.

3. **End to End Testing**

   Objective: Validate the entire application workflow, including user authentication and real-time data updates.

   Tools: Postman, MongoDB Atlas for E2E API testing.

4. **Performance Testing**

   Test Case: Simulate 100 concurrent users accessing the dashboard.

   Outcome: Measure response times and system resource usage to ensure acceptable performance.

5. **Authorization Testing**

   Test Case: Attempt to access sensitive data without proper authentication.

   Outcome: Verify that unauthorized access is denied, and the user receives an appropriate error message.

4.2 Test results and Outcomes

1. **Unit Testing**

   Test Result: The timestamp formatting function.

   Outcome: Passed.

   Comments: The function correctly formats timestamps for display.

2. **Integration Testing**

Test Result: API endpoint for fetching live data.

Outcome: Passed.

Comments: The endpoint successfully retrieves the expected data from the mock database.

3. **End to End Testing**

   Test Result: User logging in and accessing the dashboard.

   Outcome: Passed.

   Comments: Users can log in, navigate the dashboard, and view real-time data without issues.

4. **Performance Testing**

   Test Result: Simulating 100 concurrent users accessing the dashboard.

   Outcome: Passed.

   Comments: Response times and system resource usage are within acceptable limits.

5. **Authorization Testing**

   Test Result: Attempting to access sensitive data without proper authentication.

   Outcome: Passed.

   Comments: Unauthorized access is denied, and users receive an appropriate error message.

CHAPTER 5: RESULTS AND EVALUATION

5.1 Results

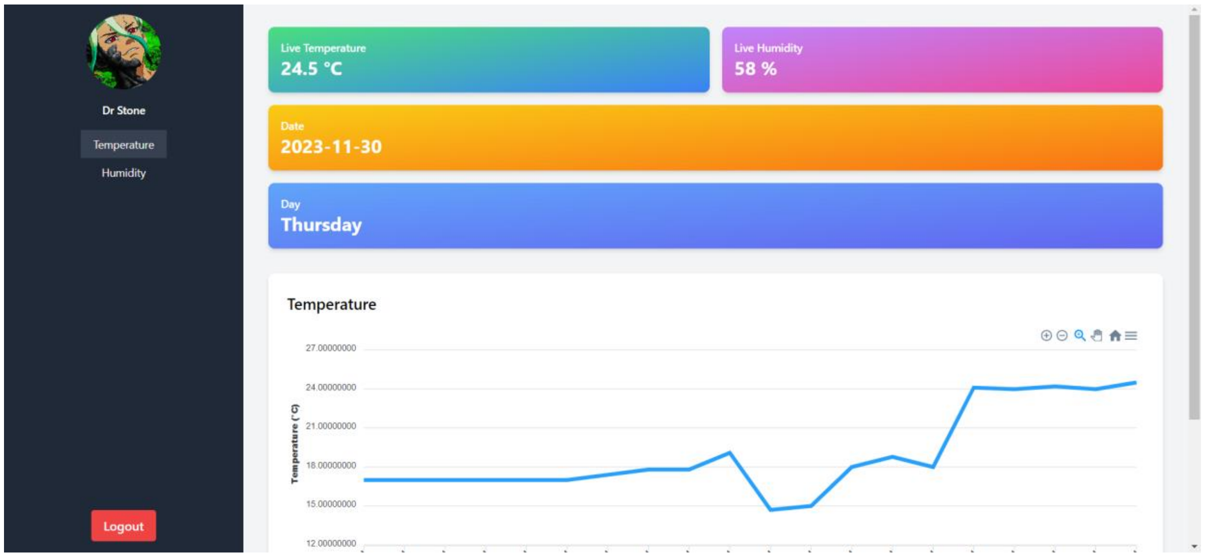
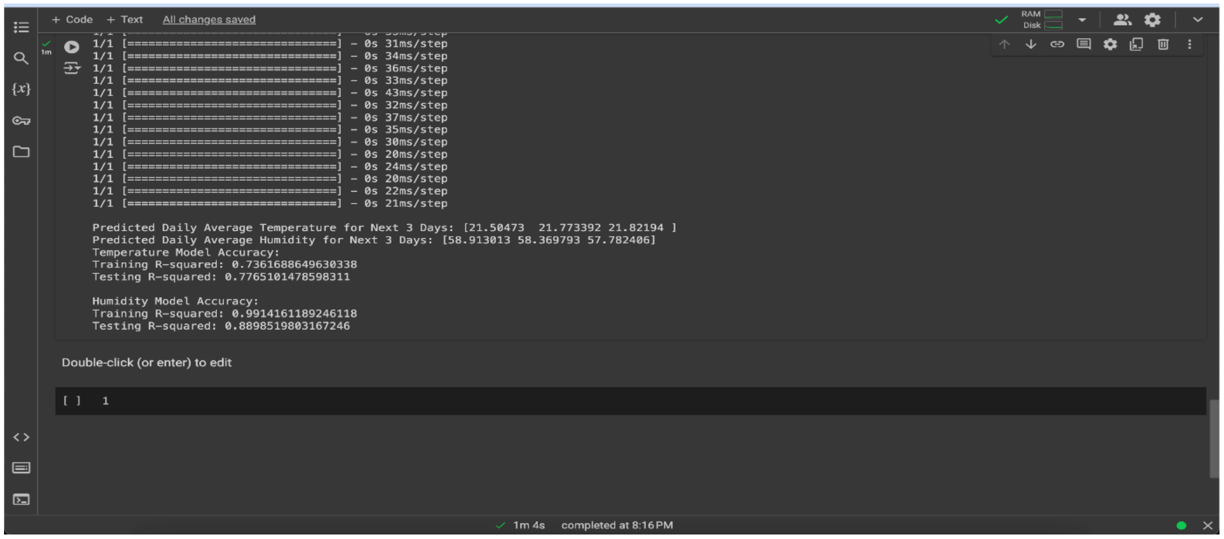
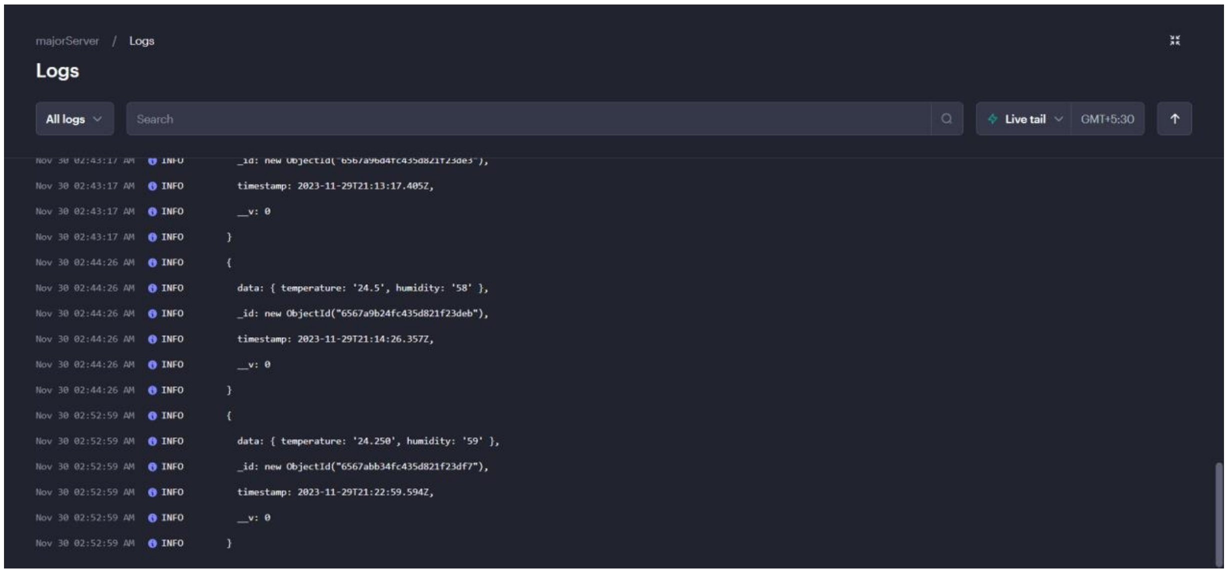


Fig. 44: Dashboard showing live temperature from sensor



```
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 21ms/step

Predicted Daily Average Temperature for Next 3 Days: [21.50473  21.773392 21.82194 ]
Predicted Daily Average Humidity for Next 3 Days: [58.913013 58.369793 57.782406]
Temperature Model Accuracy:
Training R-squared: 0.7361688649630338
Testing R-squared: 0.7765101478598311

Humidity Model Accuracy:
Training R-squared: 0.9914161189246118
Testing R-squared: 0.8898519803167246
```

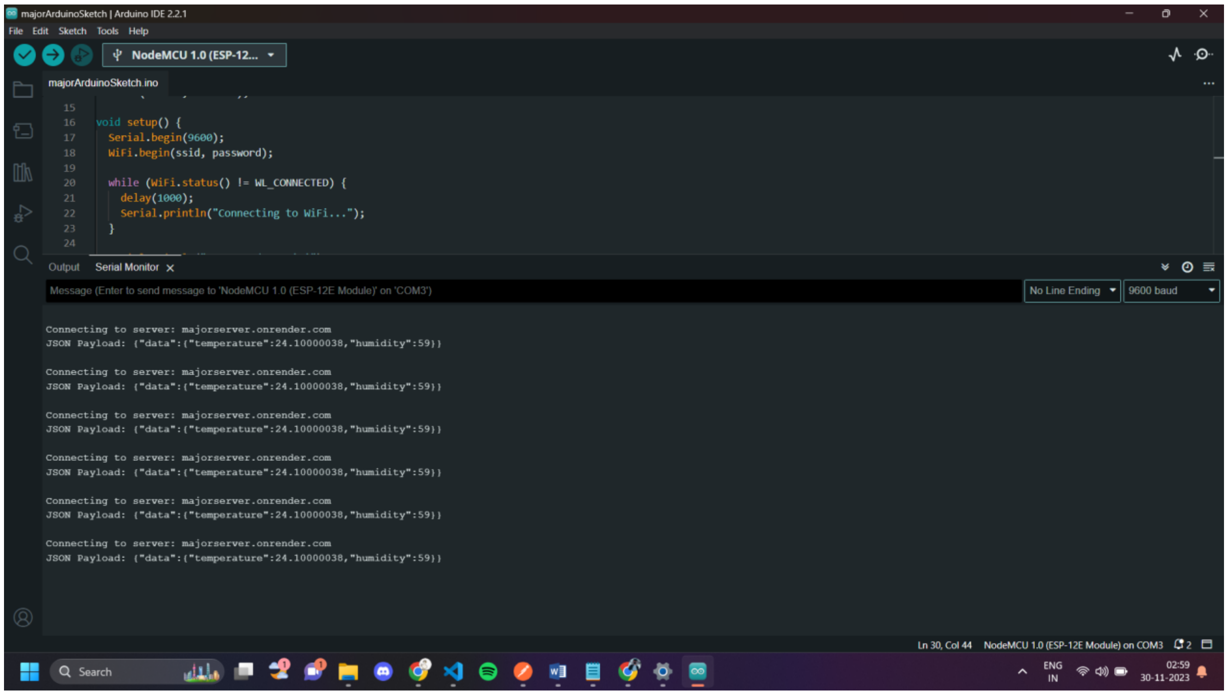
Fig. 45 Accuracy of LSTM model


majorServer / Logs

Logs

```
Nov 30 02:43:17 AM  INFO      _id: new ObjectId("6567a96d4fc435d821f23de3"),
Nov 30 02:43:17 AM  INFO      timestamp: 2023-11-29T21:13:17.405Z,
Nov 30 02:43:17 AM  INFO      __v: 0
Nov 30 02:43:17 AM  INFO    }
Nov 30 02:44:26 AM  INFO    {
Nov 30 02:44:26 AM  INFO      data: { temperature: '24.5', humidity: '58' },
Nov 30 02:44:26 AM  INFO      _id: new ObjectId("6567a9b24fc435d821f23deb"),
Nov 30 02:44:26 AM  INFO      timestamp: 2023-11-29T21:14:26.357Z,
Nov 30 02:44:26 AM  INFO      __v: 0
Nov 30 02:44:26 AM  INFO    }
Nov 30 02:52:59 AM  INFO    {
Nov 30 02:52:59 AM  INFO      data: { temperature: '24.250', humidity: '59' },
Nov 30 02:52:59 AM  INFO      _id: new ObjectId("6567abb34fc435d821f23df7"),
Nov 30 02:52:59 AM  INFO      timestamp: 2023-11-29T21:22:59.594Z,
Nov 30 02:52:59 AM  INFO      __v: 0
Nov 30 02:52:59 AM  INFO    }
```

Fig. 46: Server logs of receiving data


majorArduinoSketch | Arduino IDE 2.2.1

majorArduinoSketch.ino

```
void setup() {
  Serial.begin(9600);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
```

```
Connecting to server: majorserver.onrender.com
JSON Payload: {"data":{"temperature":24.10000038,"humidity":59}}

Connecting to server: majorserver.onrender.com
JSON Payload: {"data":{"temperature":24.10000038,"humidity":59}}

Connecting to server: majorserver.onrender.com
JSON Payload: {"data":{"temperature":24.10000038,"humidity":59}}

Connecting to server: majorserver.onrender.com
JSON Payload: {"data":{"temperature":24.10000038,"humidity":59}}

Connecting to server: majorserver.onrender.com
JSON Payload: {"data":{"temperature":24.10000038,"humidity":59}}

Connecting to server: majorserver.onrender.com
JSON Payload: {"data":{"temperature":24.10000038,"humidity":59}}
```

Fig. 47: Arduino IDE displaying the output of data being sent to server

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

In conclusion, our project marks a significant stride in the realm of IoT-enabled environmental monitoring. The integration of the DHT11 sensor with the NodeMCU ESP8266, coupled with a robust Node.js server and a React.js frontend, creates a seamless platform for users to access and monitor real-time temperature and humidity data. The emphasis on user authentication ensures data security and privacy, making the application suitable for a diverse range of users. The decision to incorporate machine learning for future temperature and humidity predictions adds a forward-looking dimension, positioning your project at the forefront of innovation in environmental sensing.

Moreover, the adoption of React.js for the frontend not only ensures a dynamic and visually appealing user interface but also lays the foundation for scalability and future feature enhancements. The commitment to user-centric design, evident in the user-friendly dashboard and intuitive navigation, reflects a keen understanding of the importance of a positive user experience.

As you delve into the future scope of our project, the proposition to include more sensors for accuracy in predictions is a commendable step. The multifaceted expansion, encompassing historical data analysis, user customization, and mobile application development, promises a holistic environmental monitoring solution. With these enhancements, your project is poised to not only provide real-time insights but also empower users with proactive measures based on predictive analytics. Overall, your project stands as a testament to the fusion of IoT, machine learning, and user-centric design, promising a compelling and innovative solution for environmental monitoring.

5.2 Future Scope:

For the future scope we will add the following functionalities:

1. **Predictive Analysis with Machine Learning:** Introducing machine learning for predicting future temperature and humidity levels adds an advanced layer to our application. This predictive analysis can help users anticipate environmental changes and take proactive measures.
2. **Multi-Sensor Integration:** To enhance the accuracy and reliability of the predictions, consider integrating more sensors. Multiple sensors placed in different locations can provide a comprehensive view of environmental conditions, contributing to more robust machine learning models.
3. **User Alerts and Notifications:** Integrate a notification system that alerts users in real-time about significant changes in environmental conditions. This feature can be crucial for scenarios where immediate action is required.
4. **User Preferences and Customization:** Allow users to customize their dashboards based on personal preferences. This could include choosing which sensors to display, setting thresholds for alerts, and adjusting the time range for historical data analysis.

By incorporating these future developments, our project can evolve into a comprehensive environmental monitoring system with enhanced predictive capabilities and a more user-friendly interface.

REFERENCES

[1] W. M. Ridwan et al., "Rainfall forecasting model using machine learning methods: Case study Terengganu, Malaysia," Ain Shams Eng. J", (2020).

[2] Mellit, A. M. Pavan, and M. Benghanem, "Least squares support vector machine for short-term prediction of meteorological time series," Theor. Appl. Climatol., vol. 111,pp. 297–307",(2013).

[3] M. Ehterامet al., "Performance improvement for infiltration rate prediction using hybridized adaptive neuro-fuzzy inference system (ANFIS) with optimization algorithms," Ain Shams Eng. J", (2020).

[4] M. Adnan et al., "Prediction of relative humidity in a high elevated basin of western Karakoram by using different machine learning models," In Weather Forecasting (IntechOpen, 2021).

[5] M. Ehteram et al., "Design of a hybrid ANN multi-objective whale algorithm for suspended sediment load prediction," Environ. Sci. Pollut. Res", (2020).

[6] M. Sapitang, W. Ridwan, K. F. Kushiar, A. N. Ahmed, and A. El-Shafie, "Machine learning application in reservoir water level forecasting for sustainable hydropower generation strategy," Sustainability, vol. 12, (2020).

[7] P. Körner, R. Kronenberg, S. Genzel, and C. Bernhofer, "Introducing gradient boosting as a universal gap filling tool for meteorological time series," Meteorol. Zeitschrift, vol. 27, (2018).

[8] J. Cai, K. Xu, Y. Zhu, F. Hu, and L. Li, "Prediction and analysis of net ecosystem carbon exchange based on gradient boosting regression and random forest," Appl. Energy, vol. 262, (2020).

[9] J. Fan et al., "Evaluation of SVM, ELM and four tree-based ensemble models for predicting daily reference evapotranspiration using limited meteorological data in different climates of China," Agric. For. Meteorol., vol. 263, (2018).

[10] S. M. Karimi, O. Kisi, M. Porrajabali, F. Rouhani-Nia, and J. Shiri, "Evaluation of the support vector machine, random forest and geostatistical methodologies for predicting long-term air temperature," ISH J. Hydraul. Eng., vol. 26, (2020).

[11] A. Sharaff and S. R. Roy, "Comparative Analysis of Temperature Prediction Using Regression Methods and Back Propagation Neural Network," Proceedings of the 2nd International Conference on Trends in Electronics and Informatics, (2018).

[12] E. Sreehari, "Prediction of Climate Variable using Multiple Linear Regression," 2018 4th International Conference on Computing Communication and Automation, ICCCA (2018).

[13] I. Park, H. S. Kim, J. Lee, J. H. Kim, C. H. Song, and H. K. Kim, "Temperature prediction using the missing data refinement model based on a long short-term memory neural network," Atmosphere (Basel)., vol. 10, no. 11, pp. 1-16, (2019).

[14] S. P. Menon, "Prediction of Temperature using Linear Regression," International Conference on Electrical, Electronics, Communication Computer Technologies and Optimization Techniques, ICEECCOT (2017) .

[15] T. Khan, M. Rabbani, S. M. Tanvir Siddiquee, and A. Majumder, "An Innovative Data Mining Approach for Determine Earthquake Probability Based on

Linear Regression Algorithm," Proc. 2019 3rd IEEE Int. Conf. Electr. Comput. Commun. Technol. ICECCT (2019).

[16] S. S. Patil and S. A. Thorat, "Early detection of grapes diseases using machine learning and IoT," in Proceedings - 2016 2nd International Conference on Cognitive Computing and Information Processing (2016).

[17] P. S. Pandey, "Machine Learning and IoT for prediction and detection of stress," in Proceedings of the 2017 17th International Conference on Computational Science and Its Applications (2017).

[18] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, " Deep learning for IoT big data and streaming analytics: A survey," IEEE Communications Surveys and Tutorials, vol. 20, no. 4, pp. 2923-2960, (2018).

[19] N. Baracaldo, B. Chen, H. Ludwig, A. Safavi, and R. Zhang, "Detecting poisoning attacks on machine learning in IoT environments," in Proceedings - 2018 IEEE International Congress on Internet of Things, ICIOT 2018 – Part of the 2018 IEEE World Congress on Services, (2018).

[20] O. Giustolisi, A. Doglioni, D. A. Savic, and B. W. Webb, "A multi-model approach to analysis of environmental phenomena," Environ (2016).

[21] D. Caissie, "The thermal regime of rivers: A review," Freshwater Biology, vol. 51, no. 8. John Wiley & Sons, Ltd, (2015).

[22] B. Gardner, P. J. Sullivan, and A. J. Lembo, Jr., "Predicting stream temperatures: geostatistical model comparison using alternative distance metrics," Can. J. Fish. Aquat. Sci., vol. 60, no. 3, pp. 344-351, Mar. (2003)

[23] S. Wang, G. Morishima, R. Sharma, and L. Gilbertson, “The Use of Generalized Additive Models for Forecasting the Abundance of Queets River Coho Salmon,” North Am. J. Fish. Manag., vol. 29, no. 2, pp. 423-433, Apr. (2009).

[24] C. H. Dagli, “Machine Learning Methods for Data Assimilation,” Intell. Eng. Syst. through Artif. Neural Networks, vol. 20, pp. 105-112, (2010).

[25] P. Romeu, F. Zamora-Martínez, P. Botella-Rocamora, and J. Pardo, “Time-Series Forecasting of Indoor Temperature Using Pre-trained Deep Neural Networks," Artificial Neural Networks and Machine Learning – ICANN (2013).

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ………………………….

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ___________________________ __Department: _________________ Enrolment No _________

Contact No. ______________________________E-mail. ______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

– Total No. of Pages =
– Total No. of Preliminary pages =
– Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ……………….(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

………

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Final Year Project

ORIGINALITY REPORT

13%	9%	7%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Liverpool John Moores University Student Paper	1%
2	link.springer.com Internet Source	1%
3	www.nature.com Internet Source	1%
4	www.ir.juit.ac.in:8080 Internet Source	1%
5	ijeecs.iaescore.com Internet Source	1%
6	K. Bhamidipati, G. Anuradha, B. Swaminathan, S. Muppidi. "Smart IoT Application in Soil Moisture and Heat Level Prediction Using Sine Cosine-Horse Herd Optimized Deep Learning", International Journal on Artificial Intelligence Tools, 2023 Publication	<1%
7	Submitted to University of Hertfordshire Student Paper	<1%

8	fastercapital.com Internet Source	<1%
9	flosshub.org Internet Source	<1%
10	Submitted to SAMRAT ASHOK TECHNOLOGICAL INSTITUTE VIDISHA M.P Student Paper	<1%
11	"International Conference on Intelligent and Smart Computing in Data Analytics", Springer Science and Business Media LLC, 2021 Publication	<1%
12	www.cs.bham.ac.uk Internet Source	<1%
13	Submitted to Manchester Metropolitan University Student Paper	<1%
14	Submitted to University of East London Student Paper	<1%
15	www.mygreatlearning.com Internet Source	<1%
16	www.geeksforgeeks.org Internet Source	<1%
17	Submitted to University of North Texas Student Paper	<1%
18	Submitted to CSU, San Jose State University	

	Student Paper	<1%
19	Submitted to Kennesaw State University Student Paper	<1%
20	Submitted to University of Bedfordshire Student Paper	<1%
21	www.sciencegate.app Internet Source	<1%
22	Submitted to Jordan University of Science & Technology Student Paper	<1%
23	Submitted to City University Student Paper	<1%
24	Submitted to Queensland University of Technology Student Paper	<1%
25	Submitted to University of Westminster Student Paper	<1%
26	muhendislik.cu.edu.tr Internet Source	<1%
27	Submitted to CSU, Fullerton Student Paper	<1%
28	qat.com Internet Source	<1%

29	Kavita Pabreja. "A data warehousing and data mining approach for analysis and forecast of cloudburst events using OLAP-based data hypercube", International Journal of Data Analysis Techniques and Strategies, 2012 Publication	<1 %
30	Tadej Krivec, Juš Kocijan, Matija Perne, Boštjan Grašic, Marija Zlata Božnar, Primož Mlakar. "Data-driven method for the improving forecasts of local weather dynamics", Engineering Applications of Artificial Intelligence, 2021 Publication	<1 %
31	Submitted to University of Leicester Student Paper	<1 %
32	assets.researchsquare.com Internet Source	<1 %
33	www.ijraset.com Internet Source	<1 %
34	Submitted to Coventry University Student Paper	<1 %
35	Submitted to IUBH - Internationale Hochschule Bad Honnef-Bonn Student Paper	<1 %
36	Submitted to University of Bristol Student Paper	<1 %

37	dr.ddn.upes.ac.in:8080 Internet Source	<1 %
38	repositorio.unal.edu.co Internet Source	<1 %
39	slides.com Internet Source	<1 %
40	repository.enp.edu.dz Internet Source	<1 %
41	www2.mdpi.com Internet Source	<1 %
42	Submitted to University of Greenwich Student Paper	<1 %
43	dokumen.pub Internet Source	<1 %
44	plugins.trac.wordpress.org Internet Source	<1 %
45	"Proceedings of Data Analytics and Management", Springer Science and Business Media LLC, 2023 Publication	<1 %
46	Emre Yakut, Seval Süzülmüş. "Modelling monthly mean air temperature using artificial neural network, adaptive neuro-fuzzy inference system and support vector regression methods: A case of study for	<1 %

Turkey", Network: Computation in Neural Systems, 2020
Publication

47	Minghui Zhang, Yatong Zhou, Yabo Liu. "Deep reservoir calculation model and its application in the field of temperature and humidity prediction", Applied Intelligence, 2022 Publication	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		