VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

MULTIDISCIPLINARY PROJECT (CO3109)

Smart Home System - *HomeTech*

Advisor: Trương Tuấn Anh, PhD.

Students: Phan Quang Đạo - 2152500 (CS)
Nguyễn Hồ Tiến Đạt - 2152503 (CS)
Nguyễn Thành Đạt - 2152506 (CS)
Phạm Gia Hùng - 2110220 (CE)
Biện Công Thanh - 2053424 (CE)

HO CHI MINH CITY, MAY 2024

Contents

1 Workload

ID	Student ID	Full Name	Workload	Commitment
1	2152500	Phan Quang Đạo	Requirements analysis Database design Back-end development	100%
2	2152503	Nguyễn Hồ Tiến Đạt	Introduction System modeling UI design	100%
3	2152506	Nguyễn Thành Đạt	System modeling Architecture design Front-end development	100%
4	2110220	Phạm Gia Hùng	Devices specification Devices setup and connection Testing	100%
5	2053424	Biện Công Thanh	Devices specification Devices setup and connection Testing	100%

2 Introduction

Traditionally, managing and controlling various aspects of a home, such as lighting, temperature and so on, required manual intervention and separate systems. However, with the emergence of smart home technologies, we have the opportunity to integrate these functionalities into a unified and intelligent system.

In today's world with increasingly advancing technologies, the concept of a smart home has gained significant traction, offering homeowners unparalleled convenience, comfort, and energy efficiency. Many modern houses' owners have been taking advantage of powerful IoT (Internet-of-Thing) devices to control different factors within the home in order to make their lives better and better.

In this project of the Multidisciplinary Project course, our group aims to build a prototype of such a system using wireless connectivity, devices and web-app control. The system can receive the records from the sensors and show them on the app. In addition, users can manually control the devices as well as view the history of sensor recording and statistical figures of factors (temperature, humidity, light intensity) via this app. With the orientation of Software Engineering, we also aim to apply design patterns to implement that system with the knowledge we have learnt.

3 Requirements

3.1 Functional requirements

3.1.1 Homeowners/Users

- **Remote Control:** Users should be able to remotely control various aspects of their home environment, including, temperature, security systems and appliances, through a user-friendly web application or mobile app.
- **Automation:** Users should have the ability to create and customize automation rules based on their preferences and routines, allowing for tasks to be automated for convenience and energy efficiency.
- **Real-time Monitoring:** Users should be able to monitor real-time data from sensors installed throughout their home, such as temperature, humidity, motion detection, and energy consumption.
- **Customization** The system should allow users to personalize settings, schedules, and preferences for different devices and sensors within their smart home ecosystem.

3.1.2 Developers

- **System Development:** Developers should design, develop, and maintain the smart home system, including both front-end and back-end components.
- **Integration:** Developers should ensure seamless integration of various IoT devices and sensors, as well as third-party APIs or services, into the smart home ecosystem.
- **Scalability:** The system should be designed to scale horizontally to accommodate a growing number of users, devices, and data without sacrificing performance.
- **Monitoring and Logging:** Developers should implement monitoring and logging mechanisms to track system performance, detect errors or anomalies, and troubleshoot issues effectively.

3.1.3 Manufacturers of IoT Devices

- **Device Compatibility:** Manufacturers should ensure that their IoT devices and sensors are compatible with the smart home system's communication protocols and standards.
- **Quality Assurance:** Manufacturers should conduct thorough testing and quality assurance processes to ensure the reliability, accuracy, and durability of their products.
- **Document and Support:** Manufacturers should provide comprehensive documentation, manuals, and technical support to assist users and developers in integrating and troubleshooting their devices.

3.1.4 Internet Service Providers (ISPs)

- **Reliable Connectivity:** ISPs should provide reliable and high-speed internet connectivity to ensure seamless communication between smart home devices, sensors, and the central control system.
- **Technical Support:** ISPs should offer technical support and assistance to users experiencing connectivity issues or other network-related problems.

3.2 Nonfunctional requirements

- **Security:** The system should implement robust security measures to protect user data and prevent unauthorized access or tampering.

- **Reliability:** The system should be able to work all day and highly reliable, with minimal downtime and the ability to recover gracefully from failures.
- **Scalability:** The system should be able to accommodate a growing number of users and devices without sacrificing performance or functionality.
- **Usability:** The user interface should be intuitive and user-friendly, allowing users to easily navigate and control their smart home devices after less than 10 minutes of watching demo or reading manual.
- **Performance:** The system should be responsive and capable of handling large volumes of data efficiently, with low latency and high throughput. The period of time between 2 consecutive records of a device must be less than 10 seconds, so that the data is always up-to-date.
- **Compatibility:** The system must be capable of handling a large volume of data from at least 3 different devices.
- **Privacy:** User privacy should be protected, with measures in place to ensure that sensitive data is handled securely and in accordance with applicable regulations.

4 Devices

4.1 LED RGB

In this project, we will use module integrates 4 LED RGB (RGB stands for Red - Green - Blue, the 3 basic colors used to mix many other colors in real life) (Figure 1). This module is a product developed by Adafruit IO company to avoid pin matching errors and help simplify the circuit. The entire LED strip can be expanded to up to 144 lights and each light can control its own color from 3 basic RGB colors. This module will be used as a smart lighting system in the house of our project.

Here is the schematic to connect the module 4 LED RGB to the expansion circuit of Yolo:bit (it will be connected to pin P0) (Figure 2).

In our project, the LED RGB will be turned on and turned off manually using the Dashboard on application or automatically when receiving information from light sensor (this will be discussed later).

Figure 1: *Module 4 LED RGB*

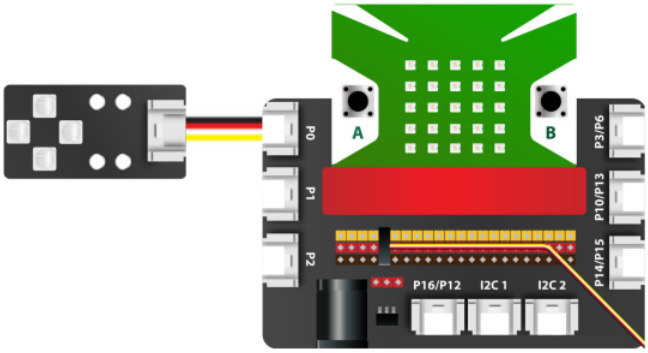

Figure 2: *Connection of module 4 LED RGB to pin P0*

4.2 Temperature - Humid sensor DHT20

DHT20 (Figure 3) is a very popular sensor for civil applications, with the advantage of low price and simple usage. The sensor's measuring range is quite suitable in normal environments without major fluctuations, with humidity in the range of 20 - 90% and temperature of 0 - 50°C. In a smart home application of this project, temperature and humidity information reflects indoor air conditions, which is important information for modern life. The information of the sensor will be sent to 1602 LCD screen to be displayed.

Here is the schematic to connect the DHT20 sensor to the expansion circuit of Yolo:bit (it will be connected to port I2C2) (Figure 4).

Unlike other devices, DHT20 uses a special protocol for control, called I2C (Inter - Integrated Circuit). This is a synchronous serial communication protocol developed by Philips Semiconductors, used to transmit and receive data between electronic devices.

Figure 3: *DHT20 sensor*

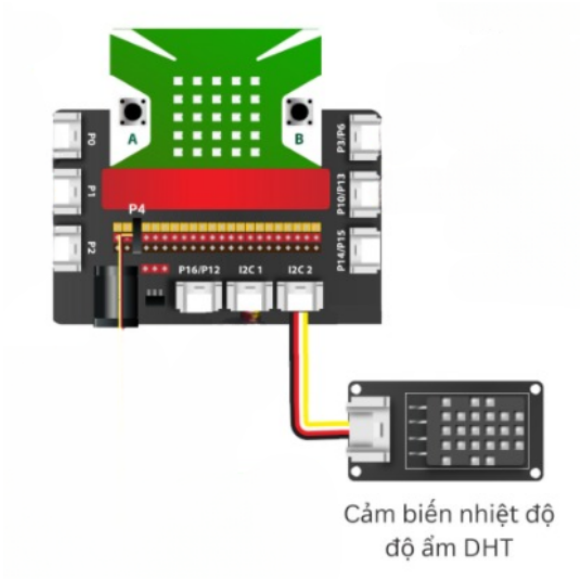

Figure 4: *Connection of DHT20 sensor to port I2C2*

4.3 LCD Screen

In our project, the LCD device we use can display 2 lines with each line having a maximum of 16 characters, so it is also called a 16 x 2 LCD screen (or 1602 LED Screen) (Figure 5). With this LCD Screen, information display will be clearer compared to Yolo:Bit's 25-light screen. However, to control this device, it is necessary to use a lot of connection pins (at least 3 control pins and 4 signal pins). Therefore, to optimize hardware design, an auxiliary circuit called I2C has been designed to accompany the LCD screen.

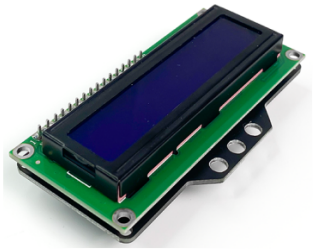

Figure 5: *16 x 2 LCD Screen*

Here is the schematic to connect the 16 x 2 LCD Screen to the expansion circuit of Yolo:bit (it will be connected to port I2C1) (Figure 6).

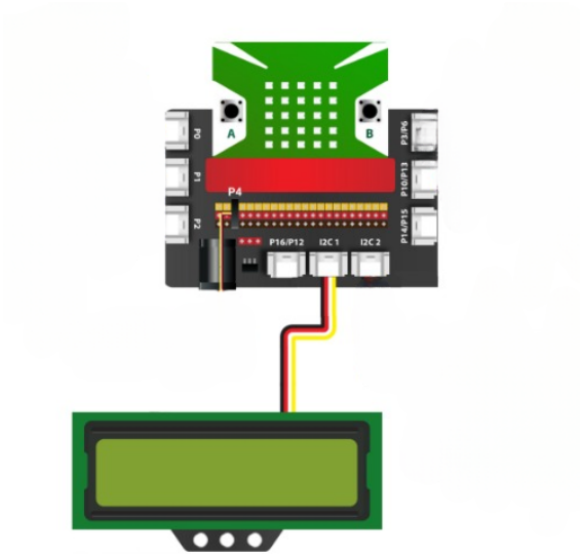

Figure 6: *Connection of 16 x 2 LCD Screen to port I2C1*

4.4 Light sensor

In the context of smart homes, lighting conditions are a useful piece of information to track. One of the important characteristics of the light sensor used in this project (Figure 7) is that the output of the sensor is a voltage. More specifically, it is based on the impedance principle to reflect the parameter to be measured. For light, it's a photoresistor - a device whose resistance changes as the light intensity changes. The information received from the sensor will be sent to

LCD Screen to be displayed. Besides, this information will be used to turn on and turn off the module LED RGB automatically.

Figure 7: *Light sensor*

Here is the schematic to connect the light sensor to the expansion circuit of Yolo:bit (it will be connected to pin P2) (Figure 8).

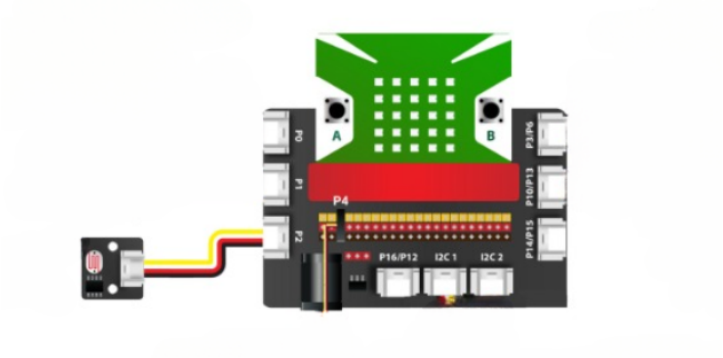

Figure 8: *Connection of light sensor to pin P2*

4.5 Mini Fan

Fans are a necessary device for smart home applications. In this project, we will use a Mini Fan device (Figure 9) to demonstrate controlling a fan motor. Using a motor, we can control the fan speed depending on the purpose of the application. This mini fan device uses a DC motor, also known as a DC (Direct Current) motor. Unlike devices that only need to be turned on and off (such as light bulbs), fans are devices that need to be controlled by speed.

Here is the schematic to connect the Mini Fan to the expansion circuit of Yolo:bit (it will be connected to pin P10) (Figure 10).

Figure 9: *Mini Fan*

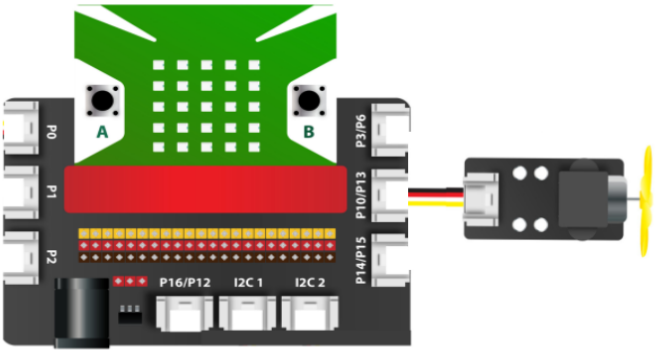

Figure 10: *Connection of mini fan to pin P10*

4.6 IR (Infra-Red) eye

This device (Figure 11) is used to receive the infrared signal from the remote.

4.7 PIR (Passive Infra-Red) sensor

This sensor (Figure 12) is used to detect people, if there is someone nearby, it will send the signal to the Yolo:Bit and the Yolo:Bit will display its LED to inform.

Figure 11: *IR eye*

Figure 12: *PIR sensor*

4.8 Remote and servo

These devices (Figure 13) are used to control the servo motor by entering the password. The signal from the remote is sent to the Yolo:Bit through IR eye, then it will be compared with the default password ("123"). If the password is correct, the servo motor will be activated. We can also change the password when we enter "123123", then press the F button to switch to password changing mode.

5 Use-case specification

The use-case diagram for the whole system is shown in Figure 14 and can be accessed via link for better view.

Below is the detailed specification for each use case (except for the use cases *Sign in*, *Sign up*, *Sign out* as they are too simple).

Figure 13: *Remote and servo*

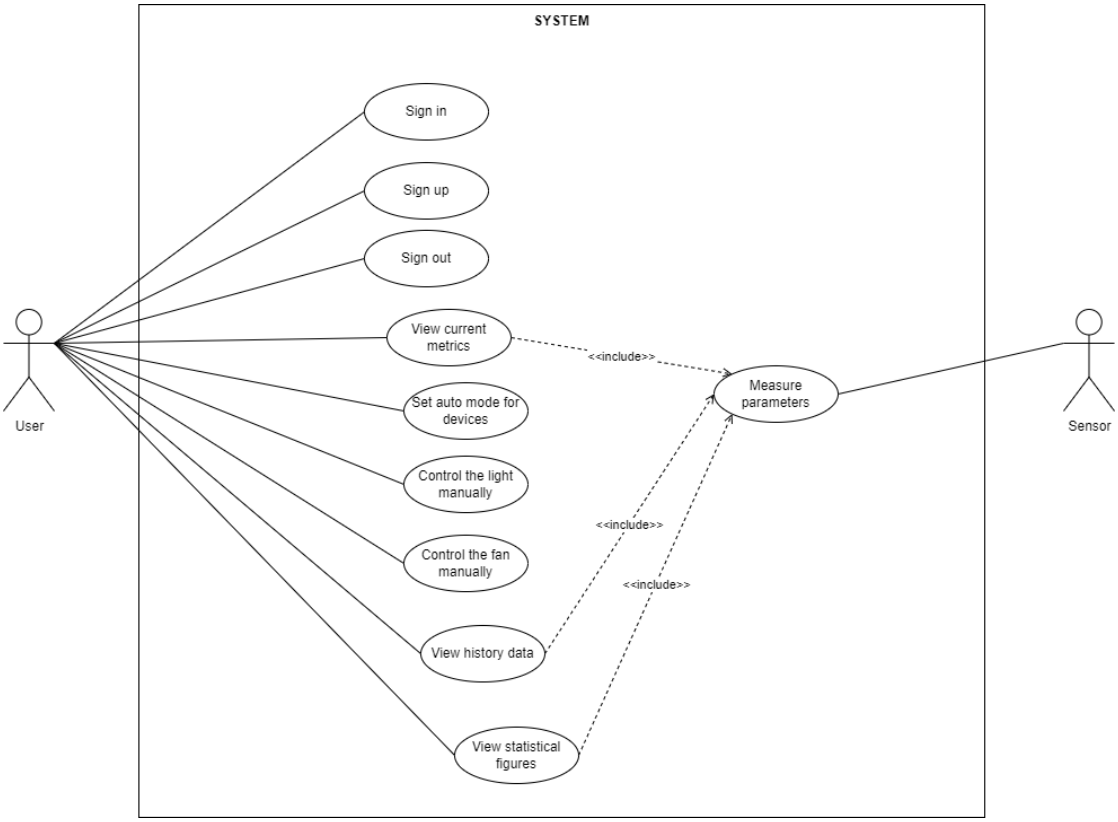

Figure 14: *Use-case diagram for HomeTech system*

Use-case ID	01
Use-case name	Measure parameters
Use-case overview	Sensors measure the parameters (temperature, humidity, light intensity) and store the data into the database.
Actors	Sensors
Precondition	The sensors are connected to the system.
Postcondition	The measured data is stored into the database.
Trigger	None
Normal flow	1. The sensor measures the specific parameter corresponding to its usage. 2. The system receives data from the sensors. 3. The system stores the data into the database.
Alternative flow	None
Exception flow	None

Table 1: *Use-case specification: Measure parameters*

Use-case ID	02
Use-case name	View current metrics
Use-case overview	Users view the current metrics of the house's condition (temperature and humidity).
Actors	Users
Precondition	Users have signed in to the system.
Postcondition	Users can view the current metrics of temperature and humidity.
Trigger	Users click on the *Dashboard* tab.
Normal flow	1. The system fetches the latest data of temperature and humidity from the database. 2. The system show the current temperature and humidity on the dashboard screen. 3. Users click on the specific button (temperature / humidity) to view the metrics clearer.
Alternative flow	None
Exception flow	None

Table 2: *Use-case specification: View current metrics*

Use-case ID	03
Use-case name	Set auto mode for devices
Use-case overview	Users set the automation mode for a device, then it can automatically change its state according to the environment.
Actors	Users
Precondition	Users have signed in to the system.
Postcondition	The device is set to the automation mode.
Trigger	Users click on the *Device control* tab.
Normal flow	1. Users click on the button corresponding to the device they want to set auto mode (fan / light). 2. Users click on the button with the title AUTO MODE to trigger the automation mode.
Alternative flow	None
Exception flow	None

Table 3: *Use-case specification: Set auto mode for devices*

Use-case ID	04
Use-case name	Control the light manually
Use-case overview	Users turn on/off the light and change its color manually.
Actors	Users
Precondition	Users have signed in to the system.
Postcondition	The light works correctly like how it is set by the users.
Trigger	Users click on the *Device control* tab.
Normal flow	1. Users click on the button of the light. 2. Users click on the button with the title SWITCH to turn on the light. 3. Users enter the value of red - green - blue components for the light color. 4. Users click on the button with the title SWITCH to turn off the light.
Alternative flow	*In step 3:* 3b. Users can switch to the Hex input section by clicking the button with that title and enter the hexadecimal code for the light color.
Exception flow	At step 3, if the input of the light color doesn't match the rules (RGB values must be an integer from 0 to 255; hexadecimal code must be in the correct form), a warning appears on the screen to notice the users to input correctly.

Table 4: *Use-case specification: Control the light manually*

Use-case ID	05
Use-case name	Control the fan manually
Use-case overview	Users change the speed of the fan manually.
Actors	Users
Precondition	Users have signed in to the system.
Postcondition	The fan speed is changed corresponding to what users have set.
Trigger	Users click on the *Device control* tab.
Normal flow	1. Users click on the button of the fan. 2. Users change the speed of the fan by clicking the button + (to speed up) or − (to slow down).
Alternative flow	None
Exception flow	None

Table 5: *Use-case specification: Control the fan manually*

Use-case ID	06
Use-case name	View history data
Use-case overview	Users view the history of all recorded data from the sensors or the data in a specific period of time and in a specific type of sensor / measurement.
Actors	Users
Precondition	Users have signed in to the system.
Postcondition	Users can view the history of measurements.
Trigger	Users click on the *History & Statistics* tab.
Normal flow	1. Users click on the *History* button. 2. The system shows all recorded measurement from the database. 3. Users enter the parameters in the *Filter* section to look for the specific data they want.
Alternative flow	None
Exception flow	None

Table 6: *Use-case specification: View history data*

Use-case ID	07
Use-case name	View statistical figures
Use-case overview	Users view the graph and the statistical values (average and peak) of data for a device in a specific day.
Actors	Users
Precondition	Users have signed in to the system.
Postcondition	Users can view the graph of recorded data, the average value and the peak value for a device in a specific day.
Trigger	Users click on the *History & Statistics* tab.
Normal flow	1. Users click on the button corresponding to the measurement they want to view the statistics (temperature / humidity / light intensity). 2. The system shows the line graph of the recorded data with respect to the chosen device, as well as its average and peak in the current day. 3. Users enter the day they want to view the statistics in the *Date* field.
Alternative flow	None
Exception flow	None

Table 7: *Use-case specification: View statistical figures*

6 System modeling

6.1 Activity diagrams

Measure parameters (Figure 15)

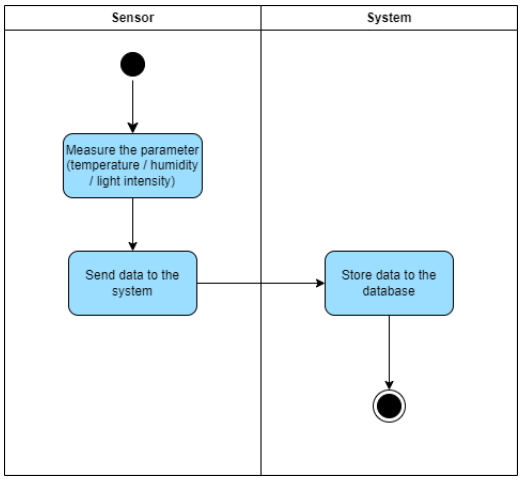

Figure 15: *Activity diagram - Measure parameters*

Each time the sensors measure the environment parameters (temperature / humidity / light intensity), they send the recorded data to the system. And then, the system stores the data into the database.

Link for better view: link.

View current metrics (Figure 16)

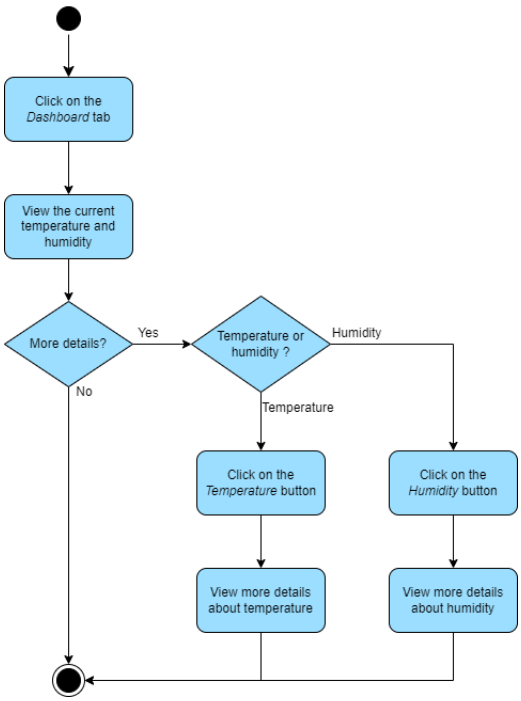

Figure 16: *Activity diagram - View current metrics*

Users navigate to the *Dashboard* tab and view the current temperature and humidity on the screen. To get more details about the metrics, users click on the corresponding button.

Link for better view: link.

Set auto mode for devices (Figure 17)

Users navigate to the *Device control* tab and click on the button corresponding to the device they want to set automation mode. Then users click on the AUTO MODE switch to activate the automation mode of that device.

Link for better view: link.

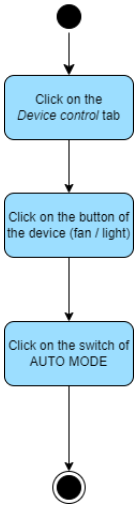

Figure 17: *Activity diagram - Set auto mode for devices*

Control the light manually (Figure 18)

Users navigate to the *Device control* tab and click on the *Light* button. If users want to turn on or turn off the light, they just simply click on the SWITCH. In order to change the light color, they can choose between 2 ways of inputting: RGB values and hexadecimal code. For RGB values, they need to input the values for red - green - blue components, each of which is an integer from 0 to 255; if the input is not in the correct form, a warning appears and they need to input again. For hexadecimal code, they first click on the RGB - Hex switch to change the way of inputting, then input the hexadecimal code for the light color they prefer; if the input is not in the correct form, a warning appears and they need to input again.

Link for better view: link.

Control the fan manually (Figure 19)

Users navigate to the *Device control* tab and click on the *Fan* button. The fan is controlled by its speed so users can change the fan speed here. If they want to speed up the fan, they click on the “+"button; if they want the fan to slow down, they click on the “-"button. They can repeatedly change the speed by redoing the action of clicking the “+"/“-"button.

Link for better view: link.

View history data (Figure 20)

Users navigate to the *History & Statistics* tab and click on the *History* button. The system shows all recorded data in the database to the screen for users to view, including the information

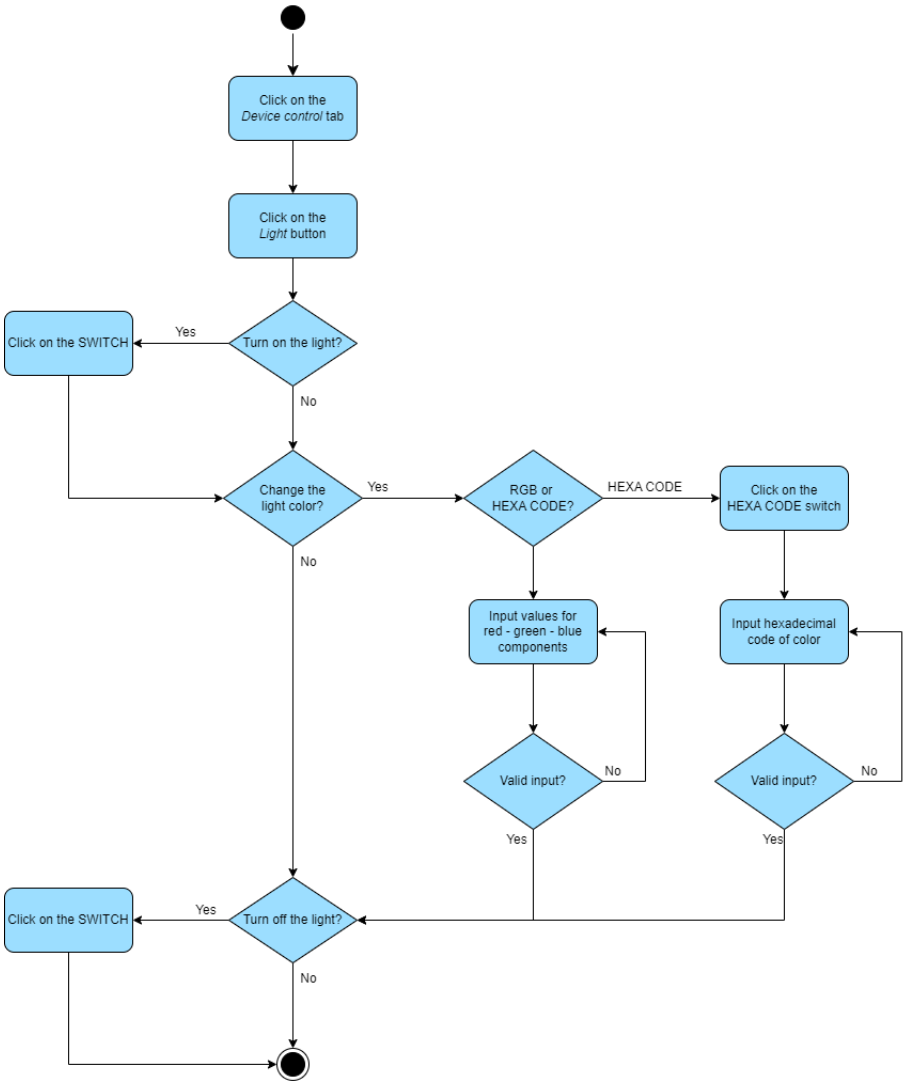

Figure 18: *Activity diagram - Control the light manually*

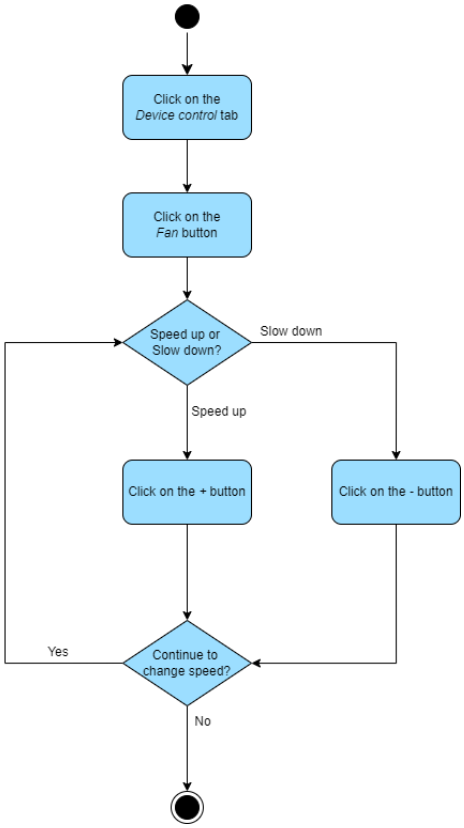

Figure 19: *Activity diagram - Control the fan manually*

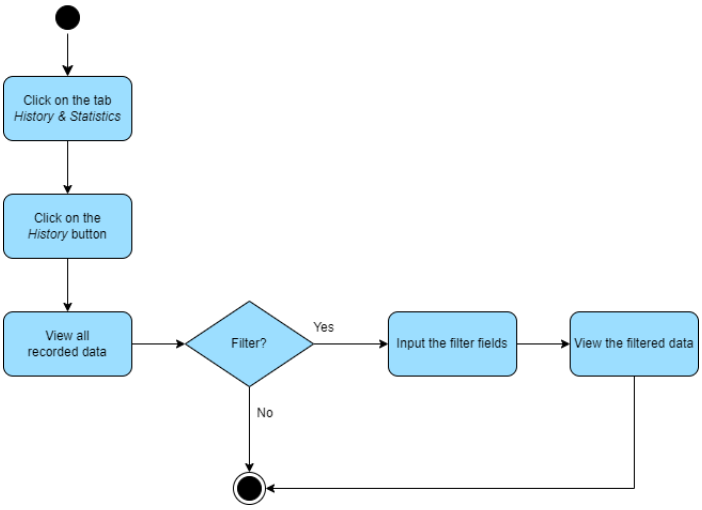

Figure 20: *Activity diagram - View history data*

of date and time, type of sensor, type of measurement and the recorded value. So as to filter the history based on some specific aspects, users need to input the information in the *Filter* field (select the type of sensor, measurement or the period of time the data is recorded) and then the system will show the desired history data for them.

Link for better view: link.

View statistical figures (Figure 21)

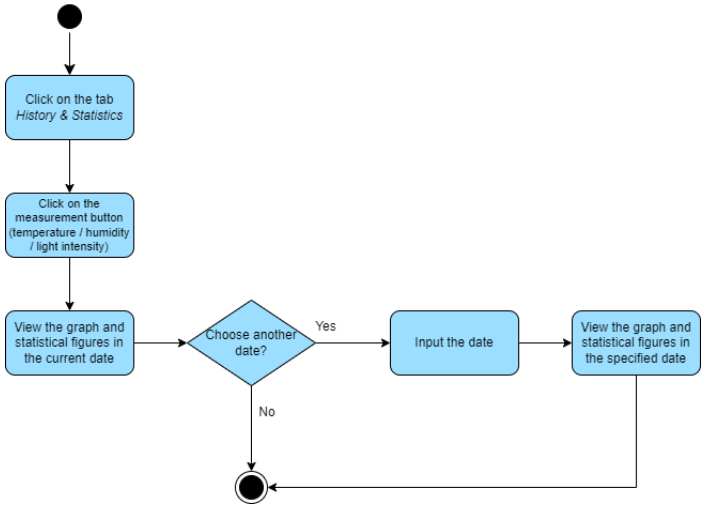

Figure 21: *Activity diagram - View statistical figures*

Users navigate to the *History & Statistics* tab and click on the button corresponding to the factor they want to view the statistical figures (temperature / humidity / light intensity). The system shows a line graph of recorded values as well as their averages and peak value with respect to that factor on the current day. If users want to view the statistical figures on another day, just input the date and the system will show the corresponding line graph, average value and peak value.

Link for better view: link.

6.2 Sequence diagram

The sequence diagram for the whole system is shown in Figure 22. The better view of the diagram can be accessed via link.

For the sensors, each time they measure the environment factors (temperature, humidity, light sensor), they send the recorded data to the system and the system then store the data into the database.

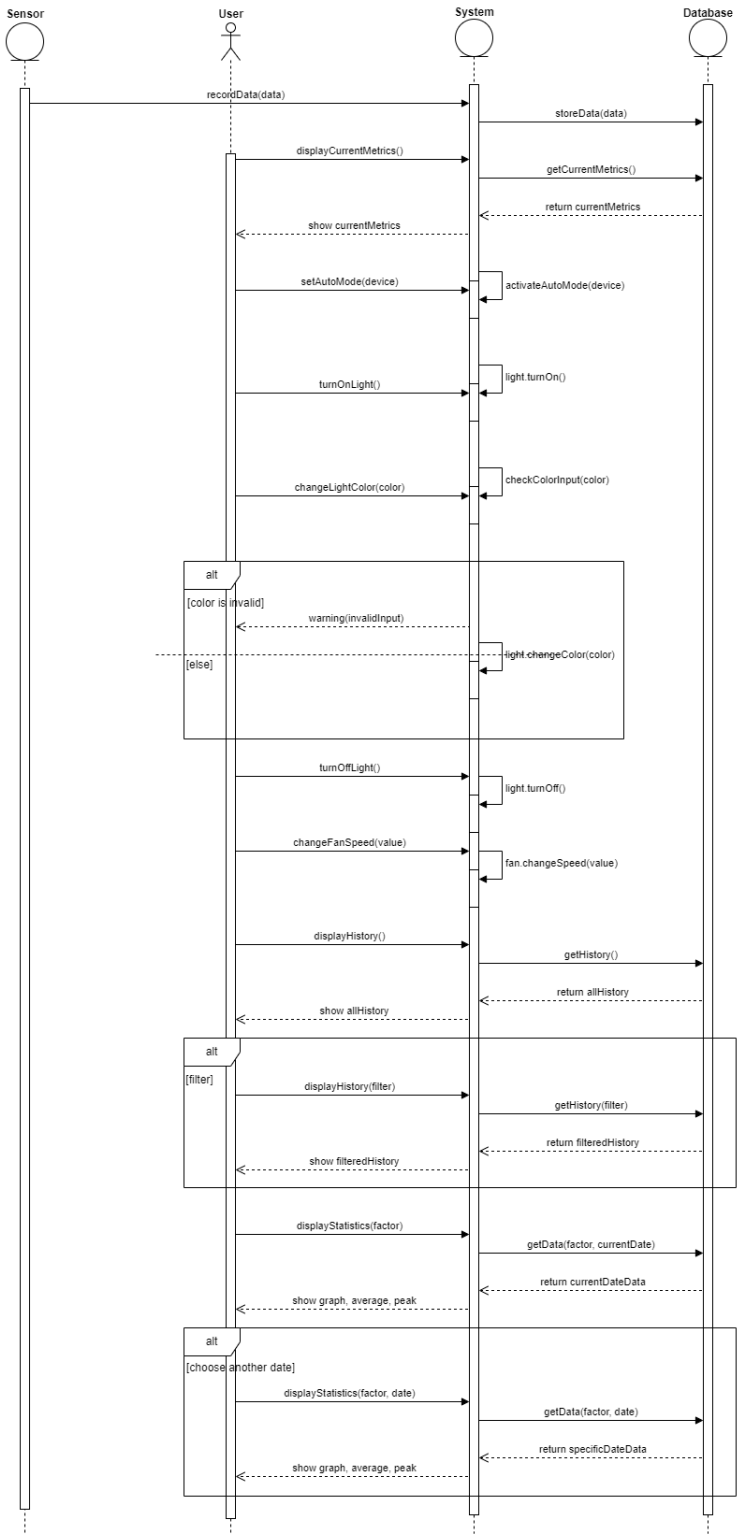

Figure 22: *Sequence diagram*

When users navigate to the *Dashboard* tab, they require the system to display current temperature and humidity. This data is fetched from the database and, after that, shown to the users on the screen.

When users want to set automation mode for a device (light or fan), the system will receive the information of the device and activate automation mode correspondingly.

To turn on/off the light, the corresponding method is called and the system will turn on/off the light. When users want to change the light color, they transfer the input to the system, then the input is checked whether it has the correct form. If the input is valid, the system will update the light color; otherwise, a warning message is sent back to users to remind them to input correctly.

When users want to change the fan speed, the system receives the information of speeding up or slowing down and set the fan speed correspondingly.

For the history viewing, users can view all recorded data that is fetched from the database and shown by the system. Users can also filter the data by transferring the filter fields to the system, then the system fetches the suitable data and shows to the users.

For the statistical figures, after choosing the measurement (temperature / humidity / light intensity), users can view the line graph with the data fetched from the database together with the average value and the peak value in the current date calculated by the system. Users can also change the date by inputting another date and view the corresponding statistical figures.

6.3 Class diagram

The following class diagram of *HomeTech* system (Figure 23) is drawn based on the model of Model-View-Controller (MVC) design pattern and can be accessed via link for better view.

The `HomeTechService` interface plays the role of *Model* with several data classes: `User`, `Sensor`, `Device` (inherited by `Light` and `Fan`), `Record`. It includes some methods for setting and retrieving data:

- `getUserInfo`: retrieve user's information with the username `username`.
- `getDevice`: retrieve the device with ID `deviceId`.
- `setAutoMode`: set or unset the automation mode of the device with ID `deviceId`.
- `switchLight`: set the state (on/off) for the light with ID `lightId`.
- `setLightColor`: set the color `color` for the light with ID `lightId`.
- `setFanSpeed`: adjust the speed of the fan with ID `fanId` by the value `extraSpeed`.

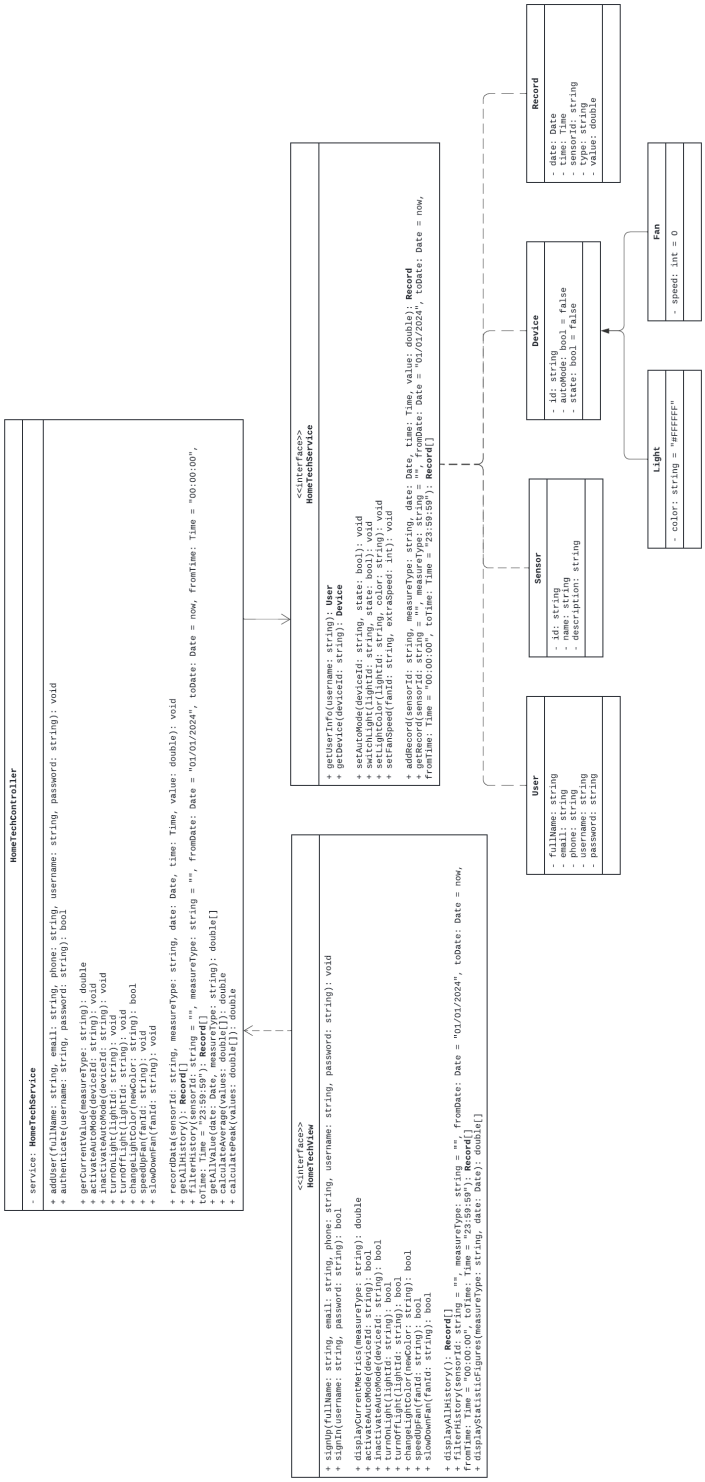

Figure 23: *Class diagram*

- `addRecord`: create a new `Record` with the information: `sensorId`, `measureType`, `date`, `time`, `value`.
- `getRecord`: retrieve all `Record`s that has `sensorId`, `measureType`, between `fromDate` and `toDate`, between `fromTime` and `toTime`. If the `sensorId` or `measureType` is an empty string, that filter field will not be applied.

The `HomeTechView` interface plays the role of *View* with some methods to interact with users:

- `signUp`: register a new user account with the information: `fullName`, `email`, `phone`, `username`, `password`.
- `signIn`: authenticate the user with username `username` and password `password`.
- `displayCurrentMetrics`: get the current value respective to the measurement type `measureType`.
- `activateAutoMode`: activate the automation mode of the device with ID `deviceId` and return whether it is successful.
- `inactivateAutoMode`: inactivate the automation mode of the device with ID `deviceId` and return whether it is successful.
- `turnOnLight`: turn on the light with ID `lightId` and return whether it is successful.
- `turnOffLight`: turn off the light with ID `lightId` and return whether it is successful.
- `changeLightColor`: change the color of the light with ID `lightId` into `color` and return whether it is successful.
- `speedUpFan`: increment the speed of the fan with ID `fanId` and return whether it is successful.
- `slowDownFan`: decrement the speed of the fan with ID `fanId` and return whether it is successful.
- `displayAllHistory`: get all records in the database to display the history.
- `filterHistory`: get all records that has `sensorId`, `measureType`, between `fromDate` and `toDate`, between `fromTime` and `toTime`. If the `sensorId` or `measureType` is an empty string, that filter field will not be applied.
- `displayStatisticFigures`: get all values belonging to the measurement type `measureType` on the date `date` (for sketching the line graph) and calculate their average value and maximum value (peak value).

The `HomeTechController` class plays the role of *Controller*. It has an object of `HomeTechService` and several methods for the `HomeTechView` interface to interact with that object.

- `addUser`: create a new `User` with the information: `fullName`, `email`, `phone`, `username`, `password`.
- `authenticate`: check whether the string `password` is the actual password of the user with username `username`.
- `getCurrentValue`: retrieve the current value respective to the measurement type `measureType`.
- `activateAutoMode`: activate the automation mode of the device with ID `deviceId`.
- `inactivateAutoMode`: inactivate the automation mode of the device with ID `deviceId`.
- `turnOnLight`: turn on the light with ID `lightId`.
- `turnOffLight`: turn off the light with ID `lightId`.
- `changeLightColor`: change the color of the light with ID `lightId` into `color` and return whether it is successful.
- `speedUpFan`: increment the speed of the fan with ID `fanId`.
- `slowDownFan`: decrement the speed of the fan with ID `fanId`.
- `recordData`: create a new Record to store the information: `sensorId`, `measureType`, `date`, `time`, `value`.
- `getAllHistory`: retrieve all records in the database to display the history.
- `filterHistory`: retrieve all records that has `sensorId`, `measureType`, between `fromDate` and `toDate`, between `fromTime` and `toTime`. If the `sensorId` or `measureType` is an empty string, that filter field will not be applied.
- `getAllValue`: retrieve all values recorded with respect to the measurement type `measureType` on the date `date`.
- `calculateAverage`: calculate the average value of all values in a list.
- `calculatePeak`: find the maximum value of all values in a list.

7 UI design - Prototype

The Figma design of *HomeTech* user interface can be accessed via link for better view.

7.1 Sign in and sign up

Figure 24: *Interface - Initial site*

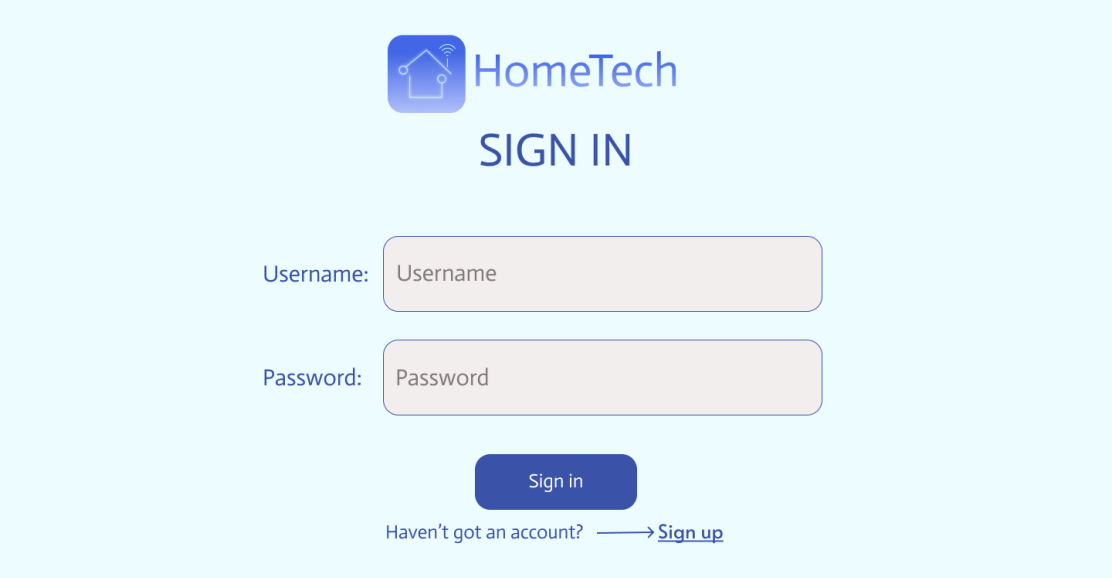

Figure 25: *Interface - Sign in*

The initial site of our web app is the place for sign in or sign up (Figure 24). Here users can choose the option *Sign in* or *Sign up* by clicking on the corresponding button.

With *Sign in* option, the interface is like Figure 25. Users need to input their username and password and then click the button *Sign in* to sign in to our system. If they have not got an account, they can create a new one by navigating to *Sign up* link.

With *Sign up* option, the interface is like Figure 26. Users need to input their personal information, including their full name, email address and phone number. They also need to choose a username and a password for logging in to the system. After completing the input fields, they can click on the button *Sign up* or navigate to the *Sign in* link if they realize they have already had an account of our system.

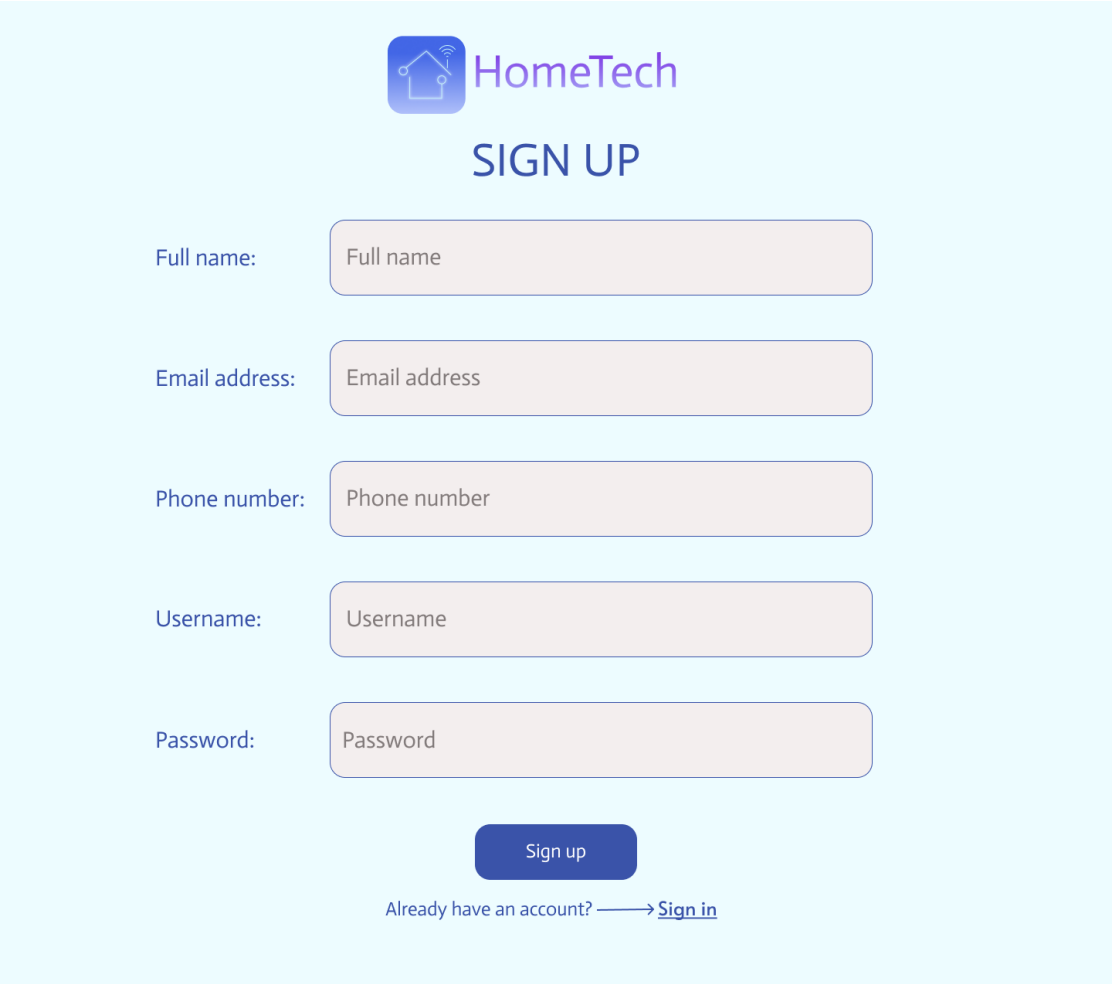

Figure 26: *Interface - Sign up*

7.2 Dashboard

After signing in to our system, the dashboard site of *HomeTech* is displayed (Figure 27). On the left, there is a navigation bar containing the tabs respective to the system's functionalities (*Dashboard*, *Device control*, *History & Statistics*) and *Sign out* option (which will turn the interface back to the initial site). The user's username is also displayed under the *HomeTech* logo. To the right of the screen, the current date and time is shown together with the current temperature and humidity below the picture.

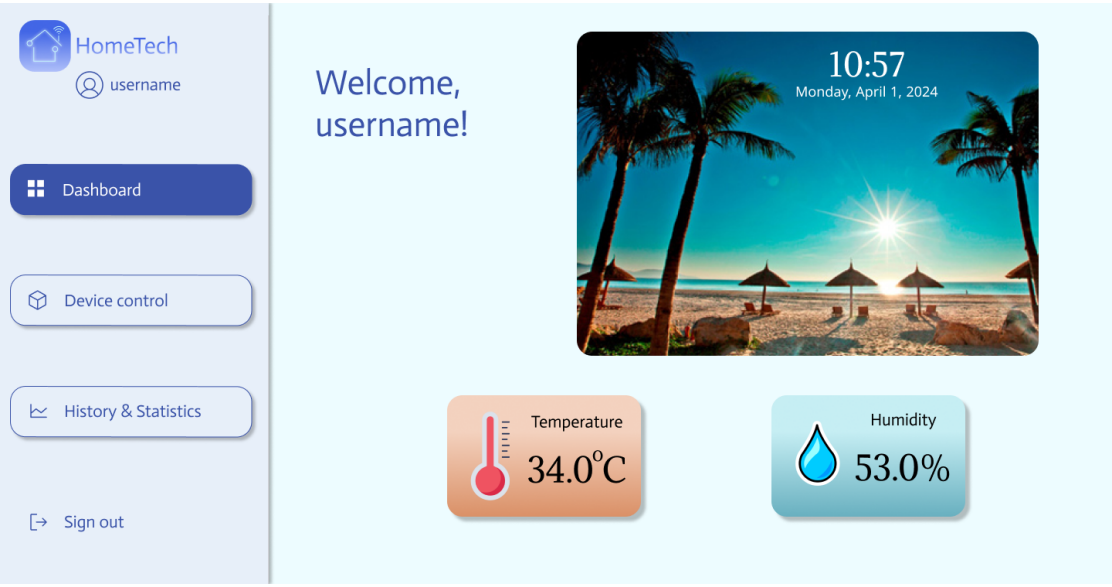

Figure 27: *Interface - Dashboard*

If the users click on the the button *Temperature* or *Humidity*, they can see the details of the corresponding metrics (Figure 28 and Figure 29). Users can come back to the dashboard site by clicking on the arrow on the top-right corner.

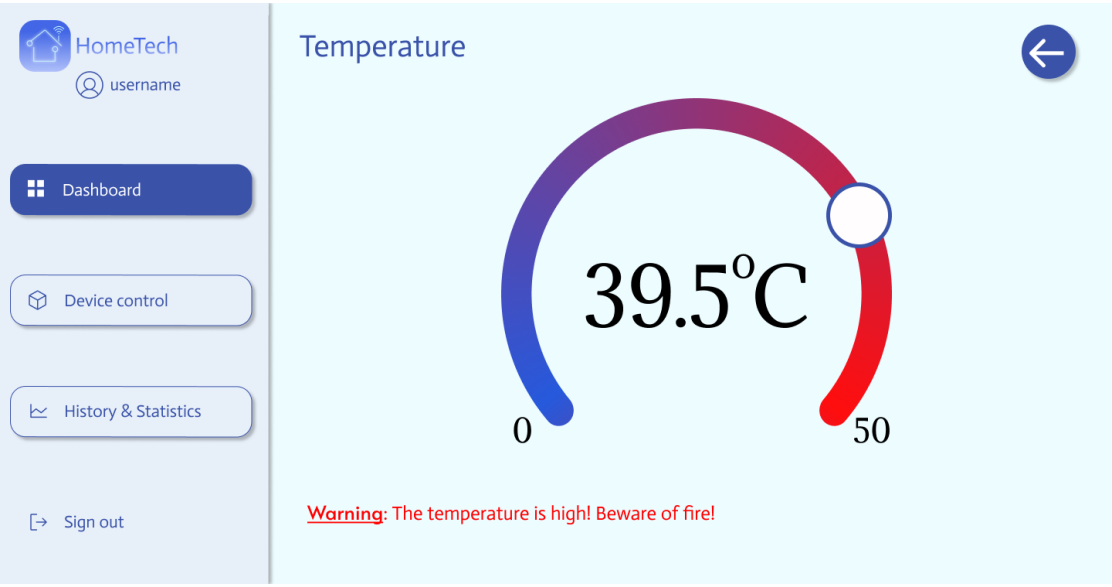

Figure 28: *Interface - Temperature*

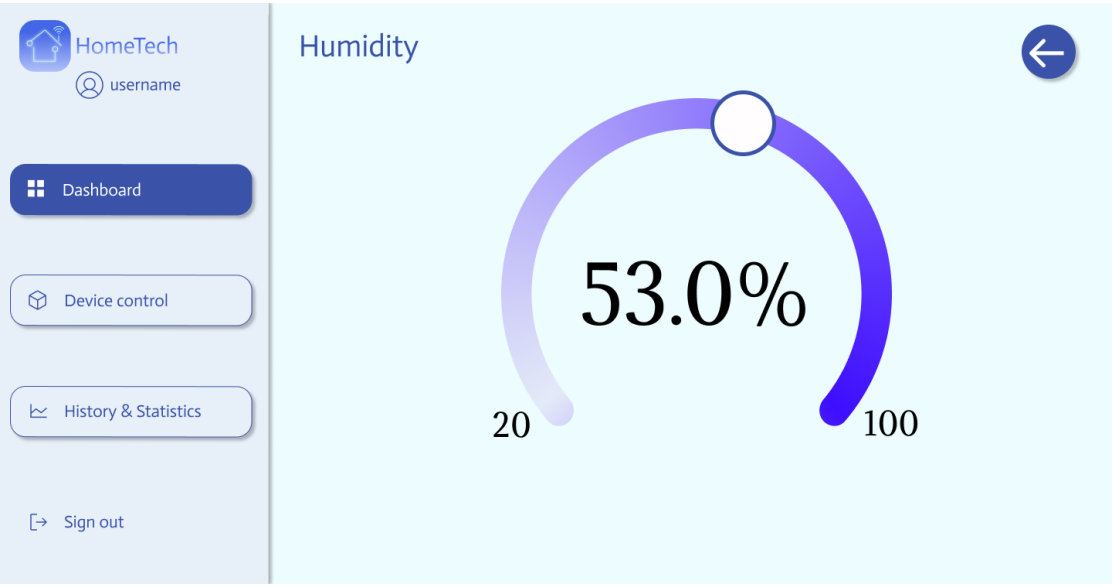

Figure 29: *Interface - Humidity*

7.3 Device control

When navigating to *Device control* tab, the screen will be displayed like Figure 30. In addition to the navigation bar in the dashboard site, users can now click on the *HomeTech* logo to turn back to the dashboard site.

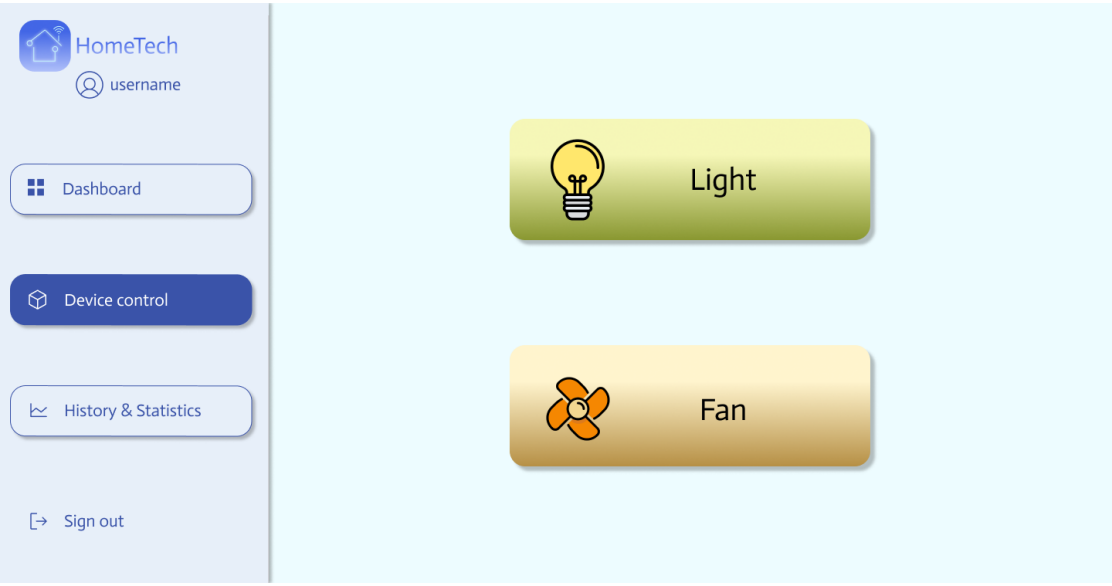

Figure 30: *Interface - Device control*

If users click on the *Light* button, the system will display the current light intensity together with the light control fields (Figure 31). There is a switch for turning on/off the light, a switch for activating or inactivating the light's automation mode and a place for changing the light color. The default format for inputs of light color is the form of the RGB values with 3 integers ranging from 0 to 255 to indicate the red, green, blue components respectively. Users can also switch to input the hexadecimal code of the color they want (Figure 32).

If any input is not in the correct form, a warning will appear and overlay the site to remind the users to input correctly (Figure 33 for RGB, Figure 34 for hexadecimal code). In addition, users can click on the arrow on the top-right corner to go back to the *Device control* site.

If users click on the Fan button, the system will provide some mechanisms to control the fan (Figure 35). Similar to the light control, there is also a switch for users to activate or inactivate the automation mode for the fan and an arrow to turn back to *Device control* site. The fan speed can be changed using the button "+" (speeding up) and "-" (slowing down).

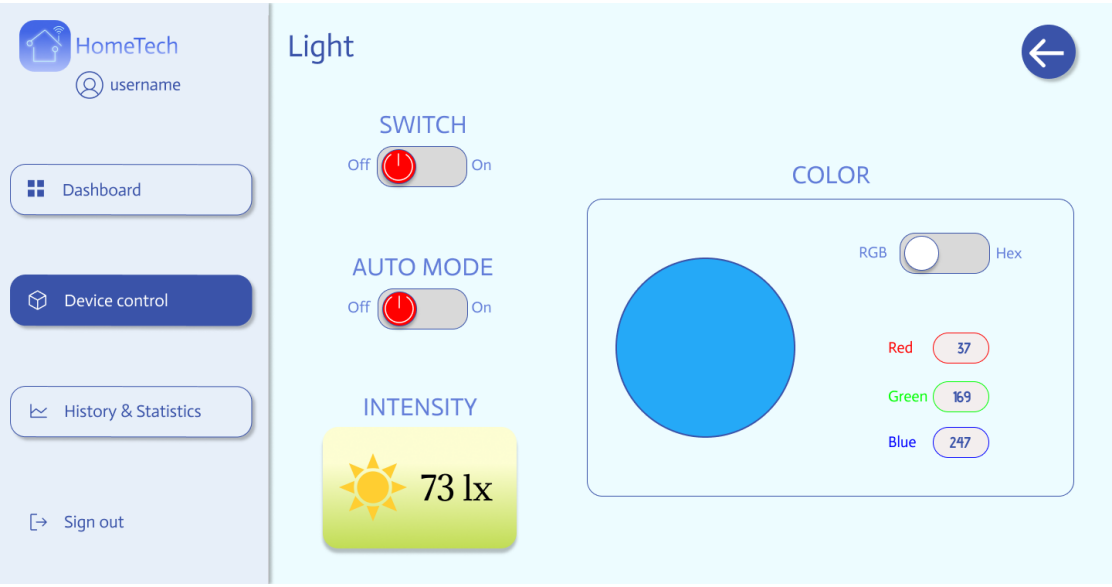

Figure 31: *Interface - Light control (RGB)*

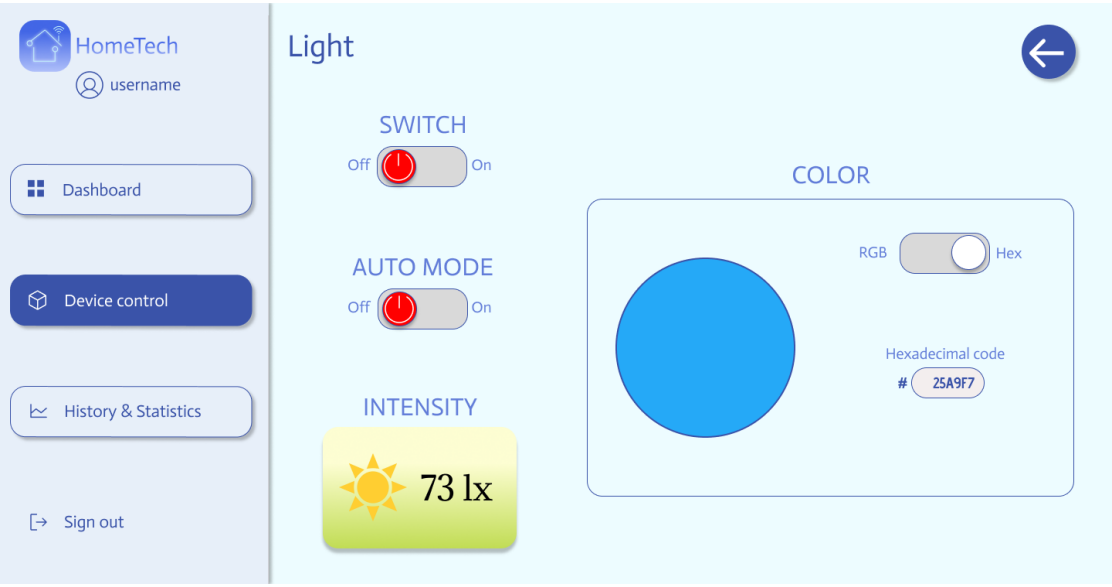

Figure 32: *Interface - Light control (Hex)*

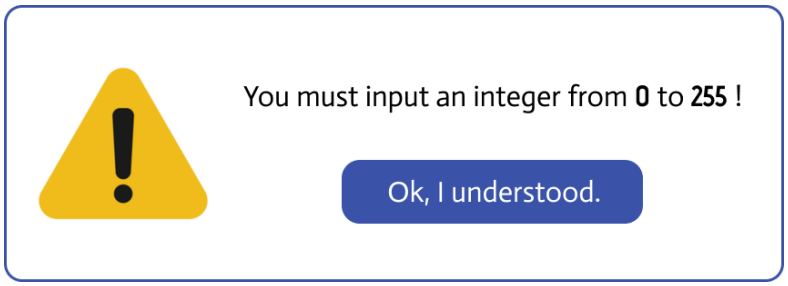

Figure 33: *Interface - Warning (RGB)*

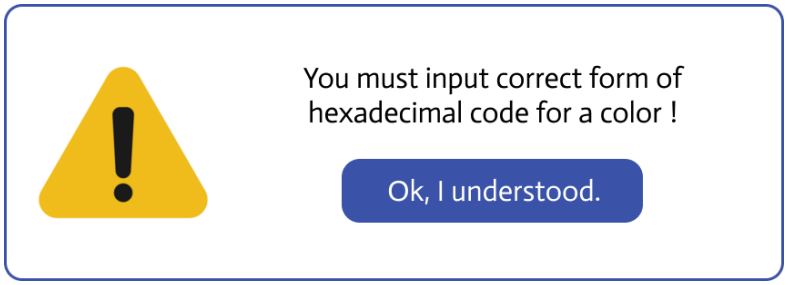

Figure 34: *Interface - Warning (Hex)*

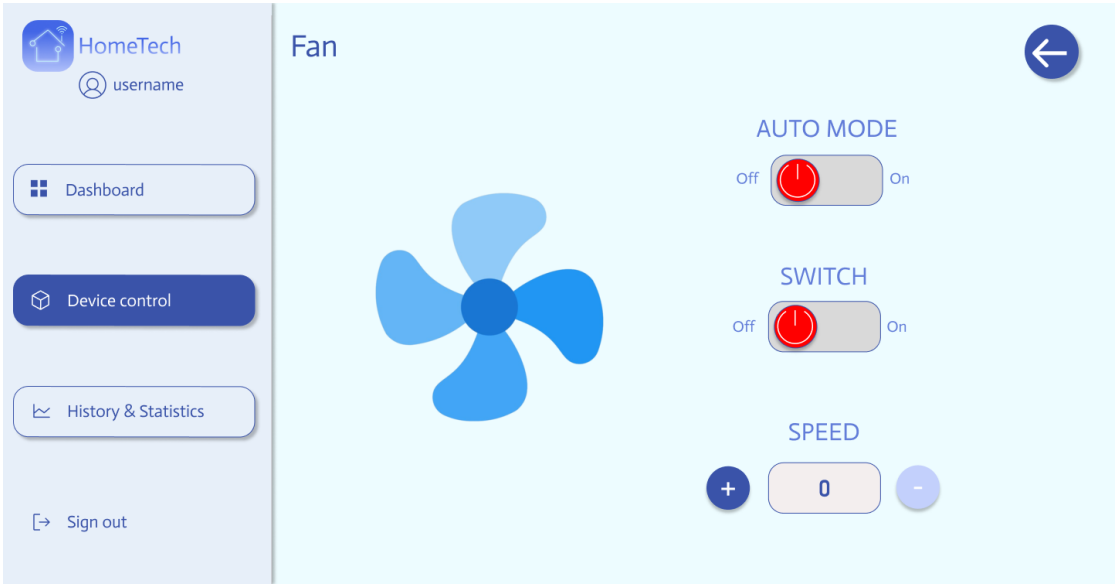

Figure 35: *Interface - Fan control*

7.4 History and statistics

When navigating to the *History & Statistics* tab, the interface will be like Figure 36.

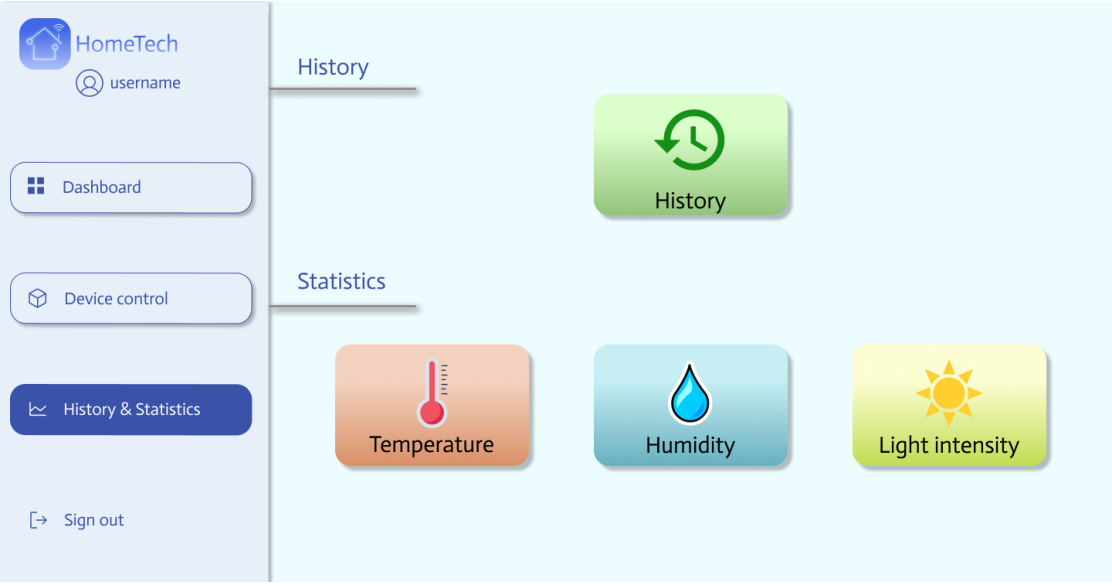

Figure 36: *Interface - History & Statistics*

The *History* section will be displayed if users click on its button (Figure 37). By default, the system will fetch all recorded data in the database and display on the screen. Each row of the displayed data will contain 4 fields: the device measuring the data, the recorded timestamp, the type of measurement and the actual value. Additionally, users can filter the data to get what they want to see by inputting the filter field (which device, from what date/time to what date/time, which measurement) below the table of data. The system will base on these fields to fetch the appropriate data from the database and then display it. The arrow on the top-right corner will help users go back to *History & Statistics* site.

If users want to see the statistical figures of the recorded data, they can click on the corresponding button and see the interface (*Temperature* - Figure 38, *Humidity* - Figure 39 or *Light intensity* - Figure 40). For each measurement type, the system will calculate and show the line graph of recorded data as well as the average value and the peak value recorded on the current date. Users also can specify another date to get the respective figures by inputting that date in the *Date* field. The function of the top-right-corner arrow is the same as the one in the *History* site.

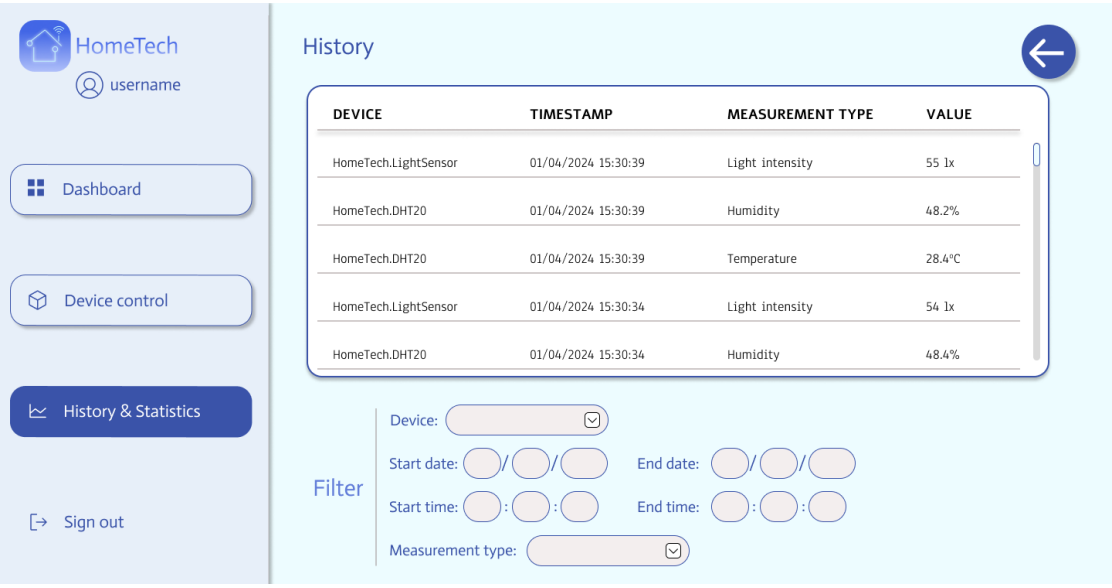

Figure 37: *Interface - History*

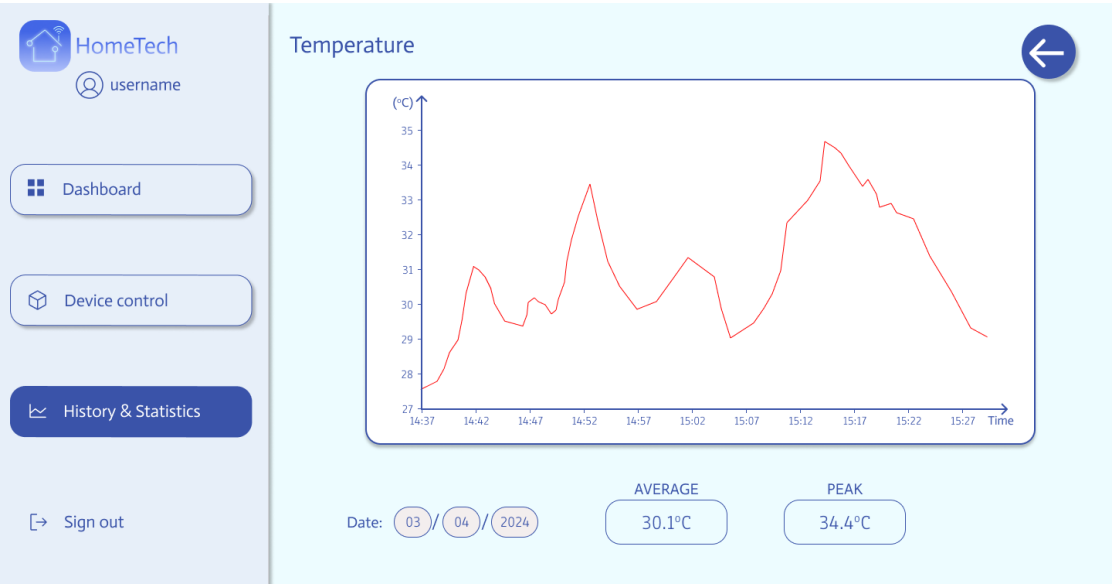

Figure 38: *Interface - Statistical figures for temperature*

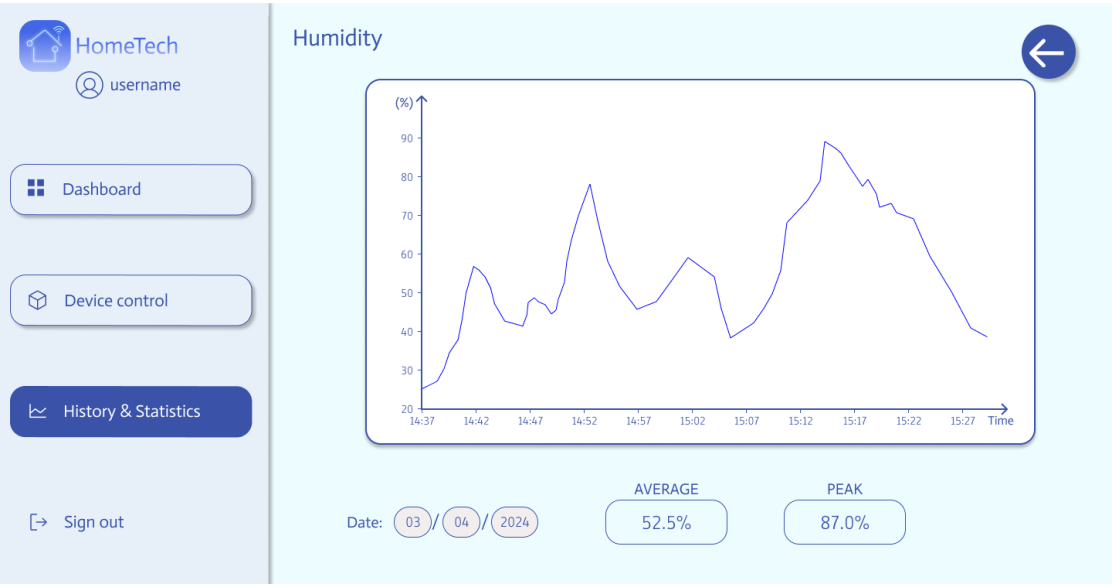

Figure 39: *Interface - Statistical figures for humidity*

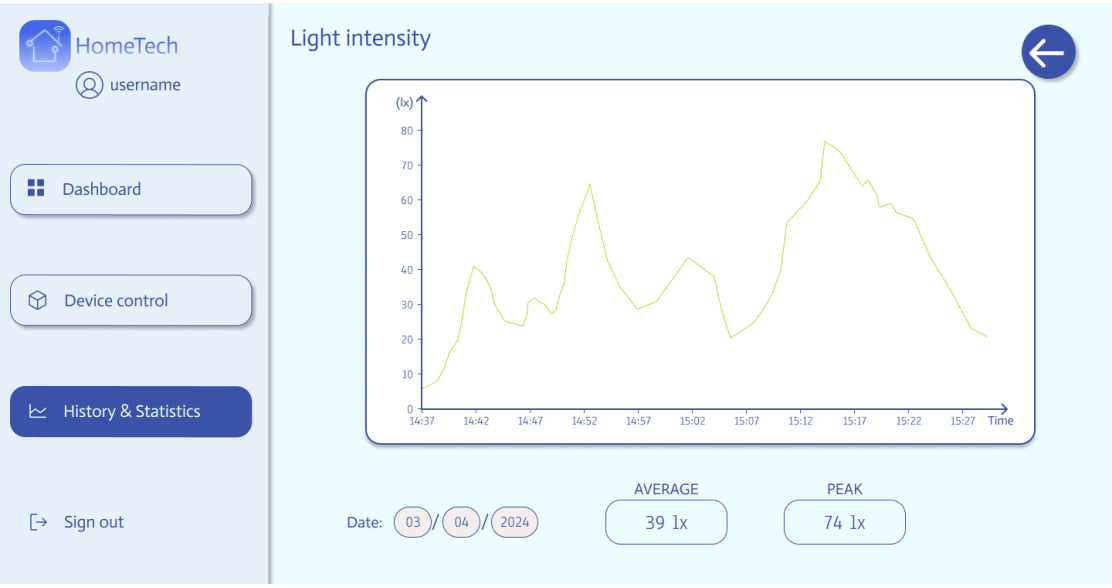

Figure 40: *Interface - Statistical figures for light intensity*

8 Architecture design

The architecture of *HomeTech* is designed based on the layered architecture design (Figure 41). The following diagram can be accessed via link (tab *Architecture diagram*) for better view.

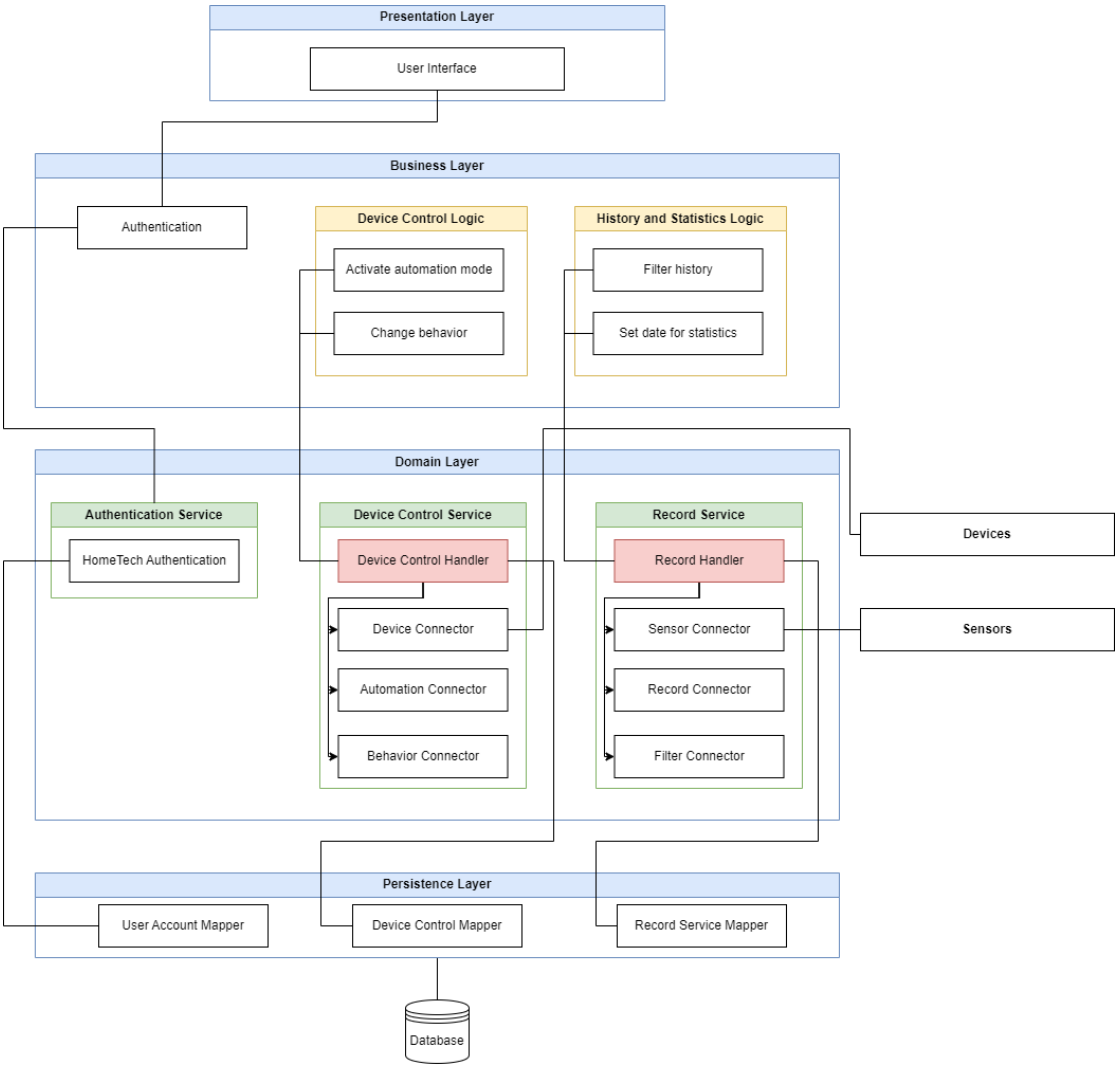

Figure 41: *Layer architecture design*

Presentation layer

User Interface: allows users to interact with *HomeTech* system.

Business layer

- *Authentication*: manages user authentication and user accounts.
- *Device Control Logic*: manage actions relating to device control, including automation mode activation and behavior change (light status, light color, fan speed).
- *History and Statistics Logic*: handles activities relating to recorded data, including filtering history and choosing date to display statistical figures.

Domain layer

The connectors and handlers facilitate the communication between the services and the business logic, ensuring a continuous flow of data and operations.

- *Authentication Service*: facilitates secure user authentication.
- *Device Control Service*: handles actions relating to device control by using *Device Control Handler*, *Device Connector* (which connects to the devices), *Automation Connector* and *Behavior Connector*.
- *Record Service*: manages the activities of measuring, storing and retrieving records with *Record Handler*, *Sensor Connector* (which connects to sensors), *Record Connector* and *Filter Connector*.

Persistence layer / Database

- *User Account Mapper*: connects and maps user information to the database system.
- *Device Control Mapper*: manages data relating to device control.
- *Record Service Mapper*: handles data relating to measurement records.

9 Database design

10 Testing

References

[1] Nguyễn Thiên Ân, Phạm Thanh Danh, Huỳnh Nhữ Hùng, Trần Nhựt Quang, Nguyễn Văn Hạnh, Lê Trọng Nhân, Lê Phương Nam, *Phát triển Gateway IoT bằng Python*, The Dariu Foundation.

[2] *Yolo:Home - AI và IoT cho nhà thông minh*, Ohstem.