

**МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ**  
**МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ**  
**МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ**

Нарантуяагийн Дөлгөөн

**Зоорины халаалт, агааржуулалт,  
чийгшүүлэлтийн автомат систем хөгжүүлэлт**  
**(Developing automated system for cellar)**

Мэдээллийн технологи (D061303)  
Бакалаврын судалгааны ажил

Улаанбаатар

2025 оны 04 сар

**МОНГОЛ УЛСЫН ИХ СУРГУУЛЬ**  
**МЭДЭЭЛЛИЙН ТЕХНОЛОГИ, ЭЛЕКТРОНИКИЙН СУРГУУЛЬ**  
**МЭДЭЭЛЭЛ, КОМПЬЮТЕРЫН УХААНЫ ТЭНХИМ**

**Зоорины халаалт, агааржуулалт, чийгшүүлэлтийн автомат  
систем хөгжүүлэлт**  
**(Developing automated system for cellar)**

Мэдээллийн технологи (D061303)  
Бакалаврын судалгааны ажил

Удирдагч: \_\_\_\_\_ Дэд профессор Э.Нямлхагва

Хамтран удирдагч: \_\_\_\_\_

Гүйцэтгэсэн: \_\_\_\_\_ Н.Дөлгөөн (19B1NUM0011)

Улаанбаатар

2025 оны 04 сар

# Зохиогчийн баталгаа

Миний бие Нарантуяагийн Дөлгөөн ”Зоорины халаалт, агааржуулалт, чийгшүүлэлтийн автомат систем хөгжүүлэлт” сэдэвтэй судалгааны ажлыг гүйцэтгэсэн болохыг зарлаж дараах зүйлсийг баталж байна:

- Ажил нь бүхэлдээ эсвэл ихэнхдээ Монгол Улсын Их Сургуулийн зэрэг горилохоор дэвшүүлсэн болно.
- Энэ ажлын аль нэг хэсгийг эсвэл бүхлээр нь ямар нэг их, дээд сургуулийн зэрэг горилохоор оруулж байгаагүй.
- Бусдын хийсэн ажлаас хуулбарлаагүй, ашигласан бол эшлэл, зүүлт хийсэн.
- Ажлыг би өөрөө (хамтарч) хийсэн ба миний хийсэн ажил, үзүүлсэн дэмжлэгийг дипломын ажилд тодорхой тусгасан.
- Ажилд тусалсан бүх эх сурвалжид талархаж байна.

Гарын үсэг: \_\_\_\_\_

Огноо: \_\_\_\_\_

## ГАРЧИГ

УДИРТГАЛ.....	1
1. СИСТЕМИЙН ТАНИЛЦУУЛГА .....	2
1.1 Оршил.....	2
1.2 Үндэслэл .....	2
1.3 Зорилго.....	3
1.4 Зорилт.....	3
2. СИСТЕМИЙН СУДАЛГАА .....	4
2.1 Технологийн судалгаа.....	4
2.2 Ижил төстэй системийн судалгаа .....	7
2.3 Бүлгийн дүгнэлт .....	10
3. СИСТЕМИЙН ШИНЖИЛГЭЭ.....	12
3.1 Төхөөрөмжийн шаардлага.....	12
3.2 Функциональ шаардлага .....	12
3.3 Функциональ бус шаардлага.....	13
3.4 Use-Case диаграмм .....	14
3.5 Өгөгдлийн урсгалын диаграмм .....	15
3.6 Entity-Relation диаграмм .....	18
3.7 Бүлгийн дүгнэлт .....	19
4. СИСТЕМИЙН ЗОХИОМЖ .....	20
4.1 Дэлгэцийн зохиомж .....	20
4.2 Техник хангамжийн зохиомж.....	21
5. ХЭРЭГЖҮҮЛЭЛТ .....	23

ГАРЧИГ

ГАРЧИГ

ДҮГНЭЛТ .....	25
НОМ ЗҮЙ .....	26
ХАВСРАЛТ .....	27
А. КОДЫН ХЭРЭГЖҮҮЛЭЛТ .....	28

## ЗУРГИЙН ЖАГСААЛТ

3.1	Use Case диаграмм .....	15
3.2	0-р түвшний ӨУД Зоорины автоматжуулалтын систем .....	16
3.3	1-р түвшний ӨУД Хэрэглэгчийн удирдлага .....	17
3.4	1-р түвшний ӨУД Хэрэглэгчийн тохиргоо ба хяналт .....	17
3.5	1-р түвшний ӨУД Мэдрэгчийн өгөгдөл боловсруулалт.....	18
3.6	Зоорины автоматжуулалтын системийн ERD .....	19
4.1	”Нүүр” хуудас.....	20
4.2	”Удирдлагын самбар” хуудас .....	21
4.3	ESP32 нэгдсэн хэлхээний хөлүүдийн байрлалыг харуулсан диаграмм ..	22
5.1	ESP32-н угсарсан байдал .....	23
5.2	Өгөгдөл болон релэйн мэдээллийг харуулсан байдал.....	24

## ХҮСНЭГТИЙН ЖАГСААЛТ

2.1	Ижил төстэй системүүдийн харьцуулалт .....	10
3.1	Ногооны хадгалах тохиромжтой нөхцөл .....	12

# Кодын жагсаалт

A.1	ESP32-т зориулан бичсэн код . . . . .	28
A.2	MqttConfig.java тохиргоо . . . . .	32
A.3	webconfig.java CORS тохиргоо . . . . .	33
A.4	SensorData.java мэдрэгчийн мэдээллийг хадгалах объектын ашиглах класс . .	33
A.5	SensorDataController.java мэдрэгчүүдийн мэдээллийг авахад ашиглах endpoint-ууд . . . . .	34
A.6	MqttListenerService.java релей болон мэдрэгчийн мэдээллийг ESP32-с авах . .	34
A.7	SensorDataRepository.java өгөгдлийн сангаас өгөгдөл авах . . . . .	36
A.8	api.js backend-с өгөгдөл татахад ашиглагдана. . . . .	36
A.9	Home.vue . . . . .	37
A.10	Dashboard.vue ESP32-н мэдээллийг харуулах view. . . . .	38



## УДИРТГАЛ

Орчин үеийн хөдөө аж ахуй болон хүнсний хадгалалтын салбарт автоматжуулалтын хэрэгцээ өсөн нэмэгдэж байгаа бөгөөд зоорины орчны хяналт, удирдлагын систем нь бүтээгдэхүүний чанарыг хадгалах, эрчим хүчний хэрэглээг оновчлох, гар ажиллагаанаас үүдэх алдааг багасгах чухал ач холбогдолтой болж байна. Гэвч уламжлалт аргаар зоорийг хянах, тохируулах нь цаг хугацаа их шаарддаг, тогтвортой бус, алдаатай байх магадлал өндөртэй тул автоматжуулсан, ухаалаг систем хөгжүүлэх хэрэгцээ гарч байна. Энэхүү судалгааны ажлын зорилго нь зоорины дотоод орчныг (температур, чийгшил, агаарын солилцоо) бодит цагийн горимд хянах, шаардлагатай үед автоматаар тохируулах боломжтой микроконтроллер болон мэдрэгчүүд дээр суурилсан ухаалаг системийг хөгжүүлэх явдал юм. Систем нь веб интерфэйсээр дамжуулан хэрэглэгчдэд мэдээлэл өгөх, алсын зайнаас удирдах боломжийг бүрдүүлнэ. Судалгааны явцад зоорины орчны стандарт шаардлагыг судалж, тохиромжтой мэдрэгч, хяналтын төхөөрөмжүүдийг сонгон авч, програмчлагдсан автомат ажиллагааг хөгжүүлэх бөгөөд үр дүнд нь зоорины хадгалалтын нөхцөлийг сайжруулах, бүтээгдэхүүний чанар алдагдахаас сэргийлэх, эрчим хүчний үр ашгийг дээшлүүлэх, хэрэглэгчийн оролцоог багасгах боломж бүрдэнэ.

# 1. СИСТЕМИЙН ТАНИЛЦУУЛГА

## 1.1 Оршил

Орчин үед технологийн хөгжлийн дэвшил нь хөдөө аж ахуй, хүнсний үйлдвэрлэл болон агуулахын системд ихээхэн нөлөө үзүүлж байна. Зоорины орчны тогтвортой байдлыг хадгалах нь хадгалалтын хугацааг уртасгаж, бүтээгдэхүүний чанарыг сайжруулах чухал хүчин зүйлсийн нэг юм. Гэвч уламжлалт зоорины хяналтын систем нь ихэвчлэн гар ажиллагаанд тулгуурласан, алдаатай, цаг хугацаа шаардсан байдгаас үр ашиг багатай хэвээр байна.

## 1.2 Үндэслэл

Хүнсний бүтээгдэхүүний хадгалалт нь температур, чийгшил, агаарын урсгал зэргээс шууд хамааралтай байдаг. Зоорины орчны өөрчлөлтийг бодит цаг хугацаанд хянах, шаардлагатай үед автоматаар тохируулах боломжтой систем байхгүй тохиолдолд дараах хүндрэлүүд үүсдэг:

- Цахилгаан эрчим хүчийг үр ашиггүй зарцуулах
- Хэт халуун эсвэл хэт хүйтэн нөхцөл үүсгэснээр хадгалагдаж буй бүтээгдэхүүний чанар алдагдах.
- Чийгшлийн тохиргоо буруу байснаас хөгц, мөөгөнцөр үүсэх.
- Чийгшүүлэгч, халаагчийг шаардлагатай үед асааж, унтраахаа мартсанаас болж орчны тохиргоо алдагдах.
- Байнгын хараа хяналт шаардлагатайгаас үүдэн хувь хүн, цаг хугацаа алдах

### **1.3 Зорилго**

Энэхүү судалгааны ажлын гол зорилго нь зоорины орчныг хянах, удирдах автоматжуулсан системийг хөгжүүлэх явдал юм. Уг систем нь зоорины дотоод орчны температур, чийгшил, агаарын чанарыг бодит хугацаанд хэмжиж, шаардлагатай үед халаагч болон агааржуулагчийг автоматаар удирдах боломжтой байна. Мөн хэрэглэгч веб интерфэйсээр дамжуулан мэдээлэл авах, системийн ажиллагааг хянах, удирдах боломжтой байх бөгөөд ингэснээр зоорины дотоод орчныг тохиромжтой нөхцөлд байлгах, бүтээгдэхүүний хадгалах хугацааг уртасгах боломжийг бүрдүүлнэ.

### **1.4 Зорилт**

Зорилгод хүрэхийн тулд дараах зорилтуудыг хэрэгжүүлнэ:

1. Зоорины орчны хяналтын шаардлага, стандартуудыг судлах
2. Мэдрэгчүүд болон микроконтроллер ашиглан өгөгдөл цуглуулах,
3. Цуглуулсан өгөгдлийг өгөгдлийн санд хадгалах
4. Өгөгдлийг веб интерфейст дамжуулах, харуулах
5. Веб интерфэйс болон удирдлагын системийг хөгжүүлэх
6. Өгөгдлийг боловсруулах алгоритмыг хөгжүүлэх
7. Системийн туршилт, алдааг олж засварлах

## 2. СИСТЕМИЙН СУДАЛГАА

Энэхүү судалгааны хэсэгт ESP32 микроконтроллер, DHT22 мэдрэгч, MQTT, LED гэрэл, болон программ хангамжийн технологиуд болох Spring Boot, Vue.js, MySQL-ийн онцлог, хэрэглээ, сонгох болсон шалтгааныг тайлбарлана. Зоорины орчны автоматжуулалт, хяналтын системийг хөгжүүлэхийн тулд эдгээр технологиудыг сонгосон бөгөөд тэдгээрийн ажиллагаа, давуу талуудыг нарийвчлан судалсан болно. Энэхүү системийг хийхийн тулд аль болох хямд өртгөөр үр бүтээл өндөртэйгөөр байхаар сонгосон.

### 2.1 Технологийн судалгаа

Системийн хөгжүүлэлтэд ашиглах технологиудын судалгаа болон сонголтын үндэслэлийг дэлгэрэнгүй тайлбарлана.

#### 2.1.1 ESP32

ESP32 нь Wi-Fi болон Bluetooth холболттой, өндөр хүчин чадалтай 32-bit микроконтроллер юм. Энэ нь бага эрчим хүч зарцуулдаг, олон төрлийн мэдрэгч болон холболтын боломжийг дэмждэгээрээ давуу талтай. IoT болон автоматжуулалтын шийдлүүдэд өргөн хэрэглэгддэг.

ESP32-г сонгосон шалтгаан:

- Wi-Fi болон Bluetooth-ийг дэмждэг тул дата дамжуулах уян хатан сонголттой.
- Олон төрлийн GPIO (Оролт, Гаралтын Порт)-той тул мэдрэгч, релей гэх мэт нэмэлт төхөөрөмжүүдийг хялбархан холбох боломжтой.
- Дотоод санах ой (RAM: 512KB хүртэл, Flash: 4MB+ хүртэл) ихтэй тул илүү боловсронгуй код, алгоритм хэрэгжүүлэх боломжтой.
- Хямд үнэ, Arduino болон Raspberry Pi-тай харьцуулахад илүү хэмнэлттэй.

- Бага эрчим хүчний горимтой (Deep Sleep Mode: 10μA хүртэл бага хэрэглээ) учир зооринд удаан ажиллахад тохиромжтой.

### **2.1.2 DHT22 (Температур, Чийгшил Мэдрэгч)**

DHT22 нь хүнсний ногооны хадгалалтад шаардлагатай чийгшил болон температурыг тогтмол хянах зориулалттай мэдрэгч юм. Өмнөх загвар болох DHT11 нь илүү хямд боловч нарийвчлал муутай учир DHT22 мэдрэгчийг сонгосон.

DHT22 мэдрэгчийн давуу талууд:

- Температур хэмжих нарийвчлал:  $\pm 0.5^{\circ}\text{C}$
- Чийгшил хэмжих нарийвчлал:  $\pm 2-5\%$
- Температур хэмжих хязгаар:  $-40^{\circ}\text{C}$ -аас  $+80^{\circ}\text{C}$  хүртэл
- Чийгшил хэмжих хязгаар: 0% - 100% RH
- Өндөр нарийвчлалтай аналог-дигитал хувиргагчтай (ADC) тул өгөгдлийг тогтвортой гаргаж өгдөг.
- 3.3V болон 5V тэжээлтэй ажиллах боломжтой, ESP32-тэй шууд холбогдож ажиллах чадвартай.
- Бэлэн Arduino болон ESP32-ийн программчлалын сангуудтай, хөгжүүлэлт хийхэд хялбар.
- 1-wire интерфейс ашигладаг тул код болон холболт энгийн.

### **2.1.3 MQTT**

MQTT нь Message Queuing Telemetry Transport гэсэн үгийн товчлол бөгөөд бага зурвасын өргөнтэй, сүлжээнд зориулагдсан хөнгөн, нийтлэх, захиалах үндсэн дээр суурилсан мессежийн протокол юм. Энэ нь IOT-д өргөн хэрэглэгддэг.

- Real-time ажилладаг, Publish/Subscribe механизмаар төхөөрөмжүүдийн хооронд өгөгдлийг real-time горимд хурдан солилцоно.
- Хөнгөн жинтэй, бага нөөц зарцуулдаг тул Сүлжээний хурд багатай, CPU, RAM хязгаарлагдмал IoT төхөөрөмжүүдэд тохиромжтой.
- Холболт тасарсан үед буцаан илгээх боломжтой (Last Will and Retain Messages) – Хэрэглэгчийн холболт тасарсан тохиолдолд сервер автоматаар ”сүүлчийн хүсэл”-ийг (Last Will message) илгээж, сүлжээний алдаа эсвэл төхөөрөмжийн унтралт зэргийг илүү найдвартай хянах боломжтой.

#### **2.1.4 LED гэрлүүд**

Системд ашиглагдах LED гэрлүүд нь жинхэнэ төхөөрөмжүүдийг төлөөлөх туршилтын элементүүд юм. Үүнд:

- Улаан LED – халаагуур ажиллаж буйг илтгэнэ.
- Ногоон LED – агааржуулалтын хавхлаг ажиллаж буйг илтгэнэ.
- Жинхэнэ хувилбарт эдгээр LED-үүдийг халаагч, хөргүүр, агааржуулалтын хавхлагууд, сэнс гэх мэт тоног төхөөрөмжөөр орлуулах боломжтой.

#### **2.1.5 Программ хангамж**

Системийн программ хангамжийн хөгжүүлэлтэд дараах технологиудыг ашиглана.

##### **Backend (Серверийн хэсэг)**

- Spring Boot:
  - Java дээр суурилсан, орчин үеийн веб үйлчилгээ хөгжүүлэхэд зориулсан framework.
  - Модульчлагдсан бүтэцтэй тул хөгжүүлэлт, өргөтгөл хийхэд хялбар.

- MSSQL өгөгдлийн сантай холбогдож, мэдээлэл боловсруулах, хадгалах үүрэгтэй.
- REST API ашиглан ESP32 болон веб интерфейстэй холбогдоно.

## **Frontend (Хэрэглэгчийн веб интерфейс)**

- Vue.js:
  - Хэрэглэгчийн веб интерфейсийг хөгжүүлэхэд ашиглана.
  - SPA (Single Page Application) загвар тул хурдан ажиллана.
  - Компонент суурьтай учир дахин ашиглагдах UI элементүүд үүсгэх боломжтой.
  - Real-time дата харуулах боломжтой.

## **Өгөгдлийн сан**

- MySQL
  - Өндөр найдвартай, баталгаатай өгөгдлийн сан учир их хэмжээний мэдээлэл боловсруулахад тохиромжтой.
  - ACID(Atomicity, Consistency, Isolation, and Durability) шинж чанар хангагдсан тул мэдээллийн алдагдлаас хамгаална.
  - Үнэгүй ашиглах боломжтой бөгөөд олон нийтийн дэмжлэг сайн.
  - SQL хэл дээр суурилсан тул сурахад ойлгомжтой, түгээмэл хэрэглэгддэг.

## **2.2 Ижил төстэй системийн судалгаа**

Зоорины орчны хяналт, автоматжуулалтын чиглэлээр олон төрлийн системүүд хөгжүүлэгдсэн байдаг. Энэ хэсэгт бид дараах ижил төрлийн хоёр системийг авч үзэн, тэдгээрийн онцлог болон өөрийн хөгжүүлсэн системтэй харьцуулалт хийнэ.

### **2.2.1 *Inkbird ITH-20R Temperature and Humidity Monitor***

#### **Давуу талууд:**

- Хос мэдрэгчтэй (indoor/outdoor) хэмжилт хийдэг.
- Өгөгдөл хадгалалт, дохиоллын системтэй.
- Утасгүй (wireless) технологи ашиглан өгөгдөл дамжуулдаг.
- Ашиглахад хялбар, хэрэглэгчдэд зориулсан UI-тэй.

#### **Сул талууд:**

- Бодит цагийн интернэт хандалтгүй.
- Автомат удирдлагын гаралт буюу релейн хяналт байхгүй.
- Тохиргоо, хянах боломж зөвхөн төхөөрөмж дээр хязгаарлагддаг.

### **2.2.2 *Sonoff TH16 + Temperature/Humidity Sensor***

#### **Давуу талууд:**

- Ухаалаг релей бүхий утасгүй удирдлагын төхөөрөмж.
- WiFi холболтоор дамжуулан eWeLink апп-аар удирдах боломжтой.
- Температур болон чийгшлийн мэдрэгч холбож, автоматаар релейг асаах/унтраах боломжтой.
- Cloud болон мобайл аппаар хянах боломжтой.

#### **Сул талууд:**

- Хаалттай систем тул өөрчлөлт, өргөтгөл хийхэд хязгаарлагдмал.
- Бодит цагийн өгөгдлийн сан, дүн шинжилгээ хийх боломжгүй.
- Системийн уян хатан байдал сул, зөвхөн өгөгдсөн хэрэглээнд зориулагдсан.



### **2.2.3 Өөрийн хөгжүүлсэн системийн харьцуулалт**

#### **Давуу талууд:**

- Бодит цагийн температур, чийгшлийн хяналт болон 4 релейн удирдлага.
- Автомат горим ба хэрэглэгчийн гараар удирдах боломжтой UI бүхий вэб систем.
- Өгөгдлийг өгөгдлийн санд хадгалж, дүн шинжилгээ хийх боломжтой.
- MQTT ашиглан хөнгөн жинтэй, хурдан холболттой өгөгдөл дамжуулалттай.
- Config тохиргоог backend-ээс real-time шинэчилдэг.
- Уян хатан архитектуртай тул цаашид IoT төхөөрөмж нэмэхэд амархан.

#### **Сул талууд:**

- Интерфэйс болон хэрэглэгчийн тохиргоо UI-г сайжруулах шаардлагатай.
- Сүлжээ тасарсан тохиолдолд найдвартай байдлыг хангах нэмэлт шийдэл хэрэгтэй.
- Мэдээллийн аюулгүй байдал (authentication, encryption) бүрэн хийгдээгүй.
- Мобайл дэмжлэг сул (responsive дизайн эсвэл апп шаардлагатай).

Үзүүлэлт	Inkbird ITH-20R	Sonoff TH16	Өөрийн систем
Температур, чийгшил хэмжих	Тийм	Тийм	Тийм
Релей удирдлага	Үгүй	Тийм	Тийм
Интернэтээр хянах	Үгүй	Тийм	Тийм
Өгөгдөл хадгалах	Хязгаарлагдмал	Үгүй	Тийм
Real-time өгөгдөл дамжуулалт	Үгүй	Тийм	Тийм
Хэрэглэгчийн UI	Хялбар	App-тай	Web
Систем өргөтгөх боломж	Үгүй	Хязгаарлагдмал	Тийм

Хүснэгт 2.1: Ижил төстэй системүүдийн харьцуулалт

## 2.3 Бүлгийн дүгнэлт

Судалгаанаас харахад ESP32 нь бага эрчим хүч зарцуулалт, утасгүй холболт, олон төрлийн мэдрэгчтэй нийцэх чадвар зэргээрээ хамгийн тохиромжтой сонголт болсон. DHT22 мэдрэгч нь зоорины чийг, температурыг өндөр нарийвчлалтай хэмжих боломжтой тул сонгогдсон. LED гэрлүүд нь системийн ажиллагааг төлөөлөхөд ашиглагдана. Программ хангамжийн хувьд, Spring Boot нь серверийн хэсэгт, Vue.js хэрэглэгчийн интерфейст, MySQL өгөгдөл хадгалах, хэрэглэгчийн бүртгэл хийхэд тус тус ашиглагдана. Эдгээр технологиудыг хослуулснаар найдвартай, хэмнэлттэй, хялбар удирдлагатай автоматжуулалтын систем хөгжүүлэх боломжтой гэж үзсэн. Үүнтэй зэрэгцээд ижил төстэй системүүдийн судалгаагаар Inkbird ITH-20R болон Sonoff TH16 зэрэг төхөөрөмжүүдийг авч үзсэн. Inkbird ITH-20R нь хос мэдрэгчтэй, өгөгдөл хадгалах боломжтой, утасгүй холболтоор өгөгдөл дамжуулдаг ч автомат удирдлага болон real-time хянах боломжгүй гэдэг сул талтай. Харин Sonoff TH16 нь WiFi холболтоор утасгүй

удирдлагатай бөгөөд температур, чийгшлийг автоматжуулан хянах боломжтой ч системийн уян хатан байдал болон өргөтгөх чадвар нь хязгаарлагдмал. Бидний хөгжүүлсэн систем нь эдгээрээс давуу талтай бөгөөд бодит цагийн өгөгдөл дамжуулалт, релей удирдлага, хэрэглэгчийн интерфэйсийн хяналт, болон системийн өргөтгөх боломжтой байдлаараа онцлог юм.

## 3. СИСТЕМИЙН ШИНЖИЛГЭЭ

Энэхүү хэсэгт зоорины автоматжуулалтын системийн шаардлагуудыг тодорхойлох зорилготой.

Энэ нь системийн үндсэн үйл ажиллагаа, хэрэглэгчийн хүсэлт, хүлээгдэж буй үр дүнг тусгана.

### 3.1 Төхөөрөмжийн шаардлага

Системийн ажиллах горимыг тохируулахдаа Монгол улсын хүнсний ногоо хадгалах стандарт, нөхцөлүүдийг харгалзан дараах үзүүлэлтүүдийг баримтална:

Ногооны төрөл	Температур (°C)	Чийгшил (%)
Төмс	4 - 15	80 - 95
Лууван, манжин	0 - 3	90 - 95
Байцаа	0 - 3	90 - 95
Сонгино, сармис	10 - 15	50 - 70
Улаан лооль	7 - 21	80 - 95
Өргөст хэмх	10 - 14	80 - 95
Шош, вандуй	7 - 10	80 - 95

Хүснэгт 3.1: Ногооны хадгалах тохиромжтой нөхцөл

### 3.2 Функциональ шаардлага

1. Систем нь температур, чийгшлийн өгөгдлийг бодит цаг хугацаанд хянах ёстой.
2. Систем нь ESP32-оос өгөгдөл хүлээн авч, өгөгдлийн санд дамжуулдаг байх ёстой.
3. Систем нь өгөгдлийн санг ашиглаж, өгөгдөл хадгалдаг байх ёстой.
4. Систем нь температур, чийгшлийн утгууд тохирсон хязгаараас хэтэрсэн үед автоматаар халаагч эсвэл сэнсийг асаах ёстой.

5. Систем нь зоорины орчны тохиргоог хэрэглэгчийн хүссэнээр өөрчлөх боломжтой байх ёстой.
6. Систем нь хэрэглэгчид хадгалалтын горим тохируулах, урьдчилан сэргийлэх сэрэмжлүүлэг хүлээн авах боломжийг олгох ёстой.
7. Систем нь лог хөтлөлт хийх ба хэрэглэгч өөрийн түүхэн өгөгдлийг харах боломжтой байх ёстой.
8. Систем нь сүлжээ тасарсан тохиолдолд хэрэглэгчид мэдэгдэх ёстой.
9. Систем нь гар утас болон компьютероос веб интерфэйсээр дамжуулан удирдах боломжтой байх ёстой.
10. Систем нь ашиглалтын явцад хэрэглэгчдэд мэдэгдэл илгээх боломжтой байх ёстой (температур хэтэрсэн, чийгшил тохиромжгүй гэх мэт).

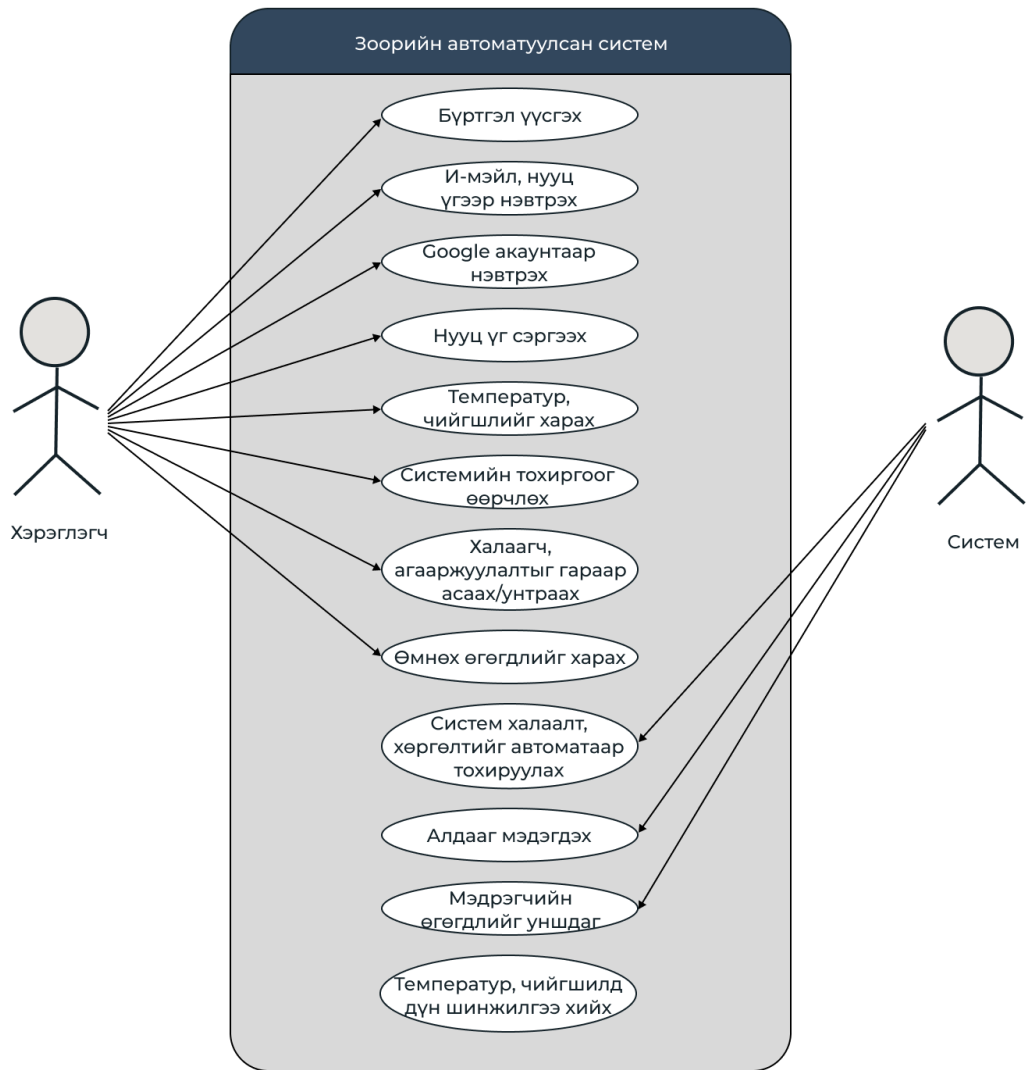
### **3.3 Функциональ бус шаардлага**

1. Систем нь 24/7 тасралтгүй ажиллах боломжтой байх ёстой.
2. Систем нь 5 секунд тутамд шинэ өгөгдлийг авах хүчин чадалтай байх ёстой.
3. Систем нь 1 секундээс бага хугацаанд хэрэглэгчийн хүсэлтэд хариу өгөх ёстой.
4. Систем нь өгөгдлийн сантай найдвартай ажиллах, өгөгдлийн бүрэн бүтэн байдлыг хангах ёстой.
5. Системийн веб интерфэйс нь хурдан бөгөөд хэрэглэгчид ойлгомжтой байх ёстой.
6. Систем нь шалгаж баталгаажсан хэрэглэгчдэд л хандах боломж олгох, аюулгүй байдлыг хангах ёстой.
7. Систем нь нэмэлт тоног төхөөрөмж шаардалгүйгээр интернэт хөтөч дээр ажиллах ёстой.

8. Систем нь монгол хэл дээр ажиллах боломжтой байх ёстой.
9. Системийн ихэнх тохиргоог хэрэглэгч өөрийн хэрэгцээнд тохируулан өөрчлөх боломжтой байх ёстой.
10. Систем нь хэрэглэгчийн хувийн мэдээллийг шифрлэх, зөвшөөрөлгүй хандалтаас хамгаалах ёстой.
11. Систем нь бүх өгөгдлийг нууцлалтайгаар дамжуулах (HTTPS, TLS 1.2 ба түүнээс дээш) шаардлагатай.

### **3.4 Use-Case диаграмм**

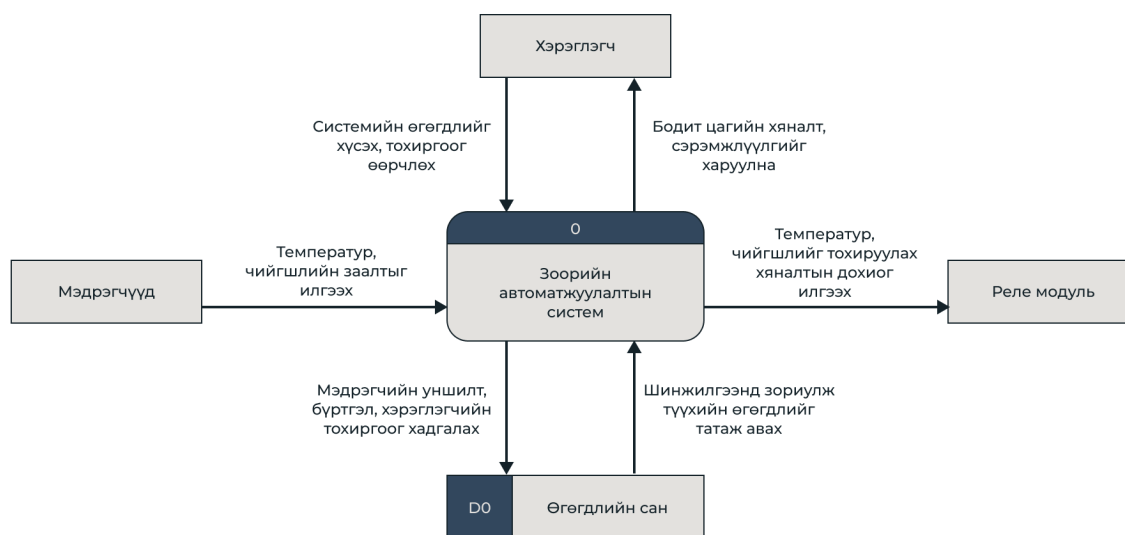
Энэхүү диаграмм нь хэрэглэгч болон систем хоорондын харилцааг харуулсан бөгөөд үндсэн функцүүд болон автоматжуулалтын үйл явцыг харуулсан болно. Энэ нь жүжигчдийн үүрэг, тэдгээрийн системийн янз бүрийн функцтэй харилцах харилцааг харуулна.



Зураг 3.1: Use Case диаграмм

### 3.5 Өгөгдлийн урсгалын диаграмм

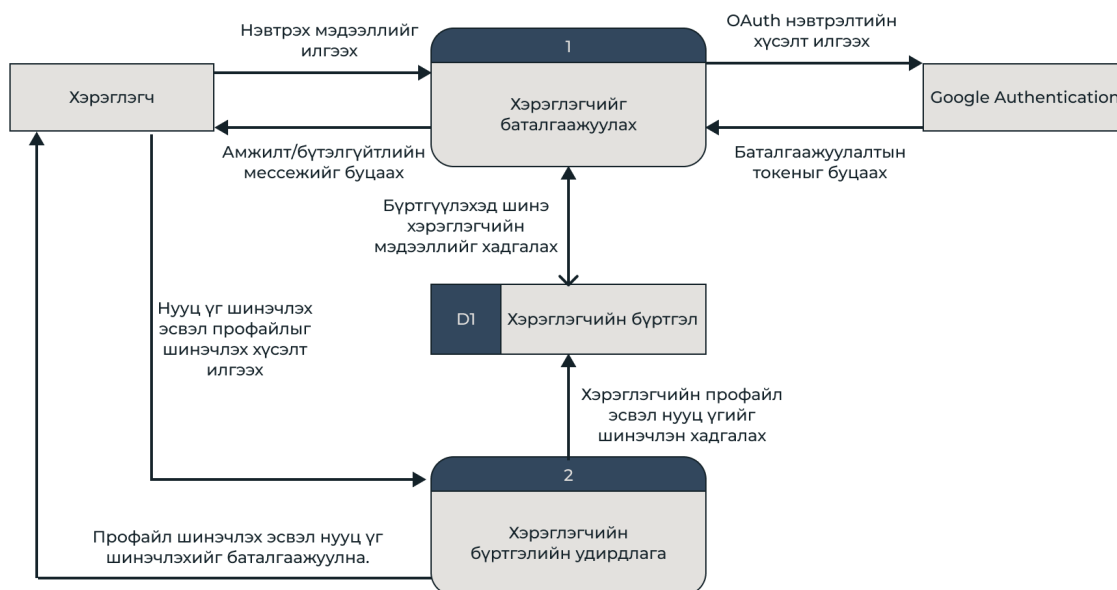
Энэхүү диаграмм нь бүхэл системийн өндөр түвшний тоймыг гаргаж, хэрэглэгчид түүнтэй хэрхэн харилцаж, гадаад нэгж, процесс, хадгалалтын хооронд өгөгдөл хэрхэн урсаж байгааг харуулна.



Зураг 3.2: 0-р түвшний ӨУД Зоорины автоматжуулалтын систем

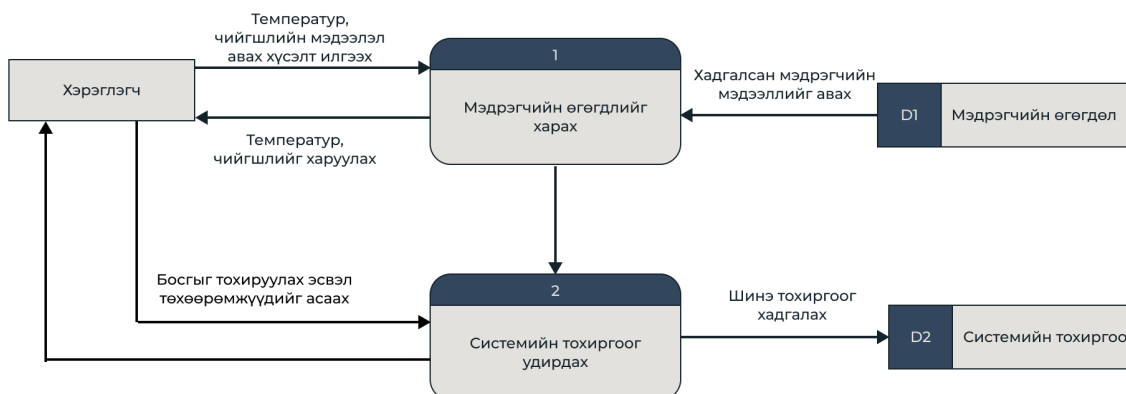
Энэ диаграммд хэрэглэгчид хэрхэн бүртгүүлэх, нэвтэрч орох, бүртгэлээ удирдах, тухайлбал нэвтрэлт танилт, профайлын шинэчлэлтүүдийг дэлгэрэнгүй тайлбарлана.





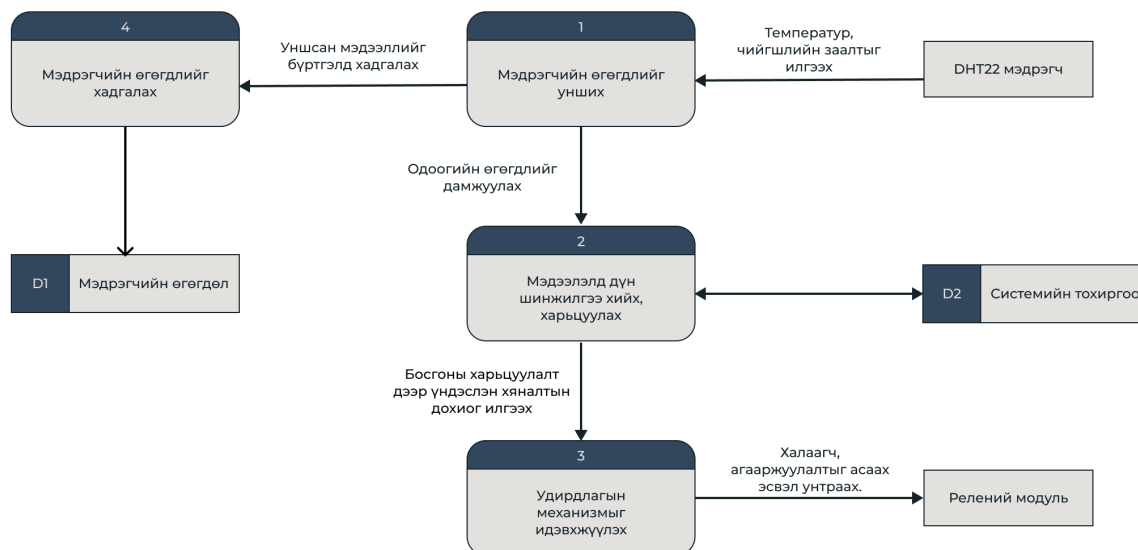
Зураг 3.3: 1-р түвшний ӨУД Хэрэглэгчийн удирдлага

Энэ диаграмм нь хэрэглэгчид системийн интерфейсээр дамжуулан зоорины нөхцөлийг (жишээлбэл, халаалт эсвэл агааржуулалтыг асаах/унтраах) гараар хэрхэн удирдаж байгааг харуулж байна.



Зураг 3.4: 1-р түвшний ӨУД Хэрэглэгчийн тохиргоо ба хяналт

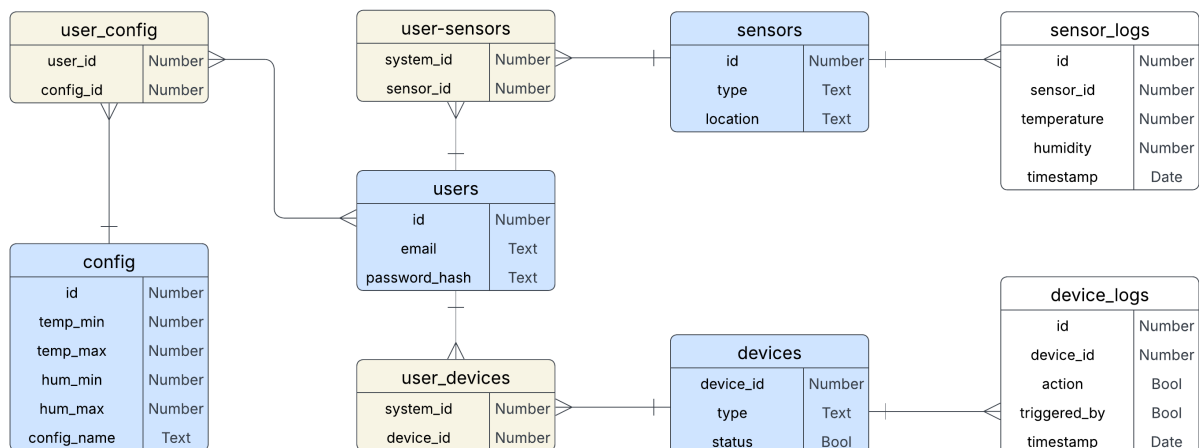
Энэхүү диаграмм нь систем нь мэдрэгчээс өгөгдлийг хэрхэн цуглуулж, боловсруулж, автомат удирдлагад зориулж хадгалдаг болохыг харуулж байна.



Зураг 3.5: 1-р түвшний ӨУД Мэдрэгчийн өгөгдөл боловсруулалт

### 3.6 Entity-Relation диаграмм

Энэхүү Entity-Relationship Diagram (ERD) нь зоорины орчны хяналт, удирдлагын автоматжуулсан системийн өгөгдлийн сангийн бүтэц, түүний бүрэлдэхүүн хэсгүүдийн хоорондын харилцааг дүрсэлнэ.



Зураг 3.6: Зоорины автоматжуулалтын системийн ERD

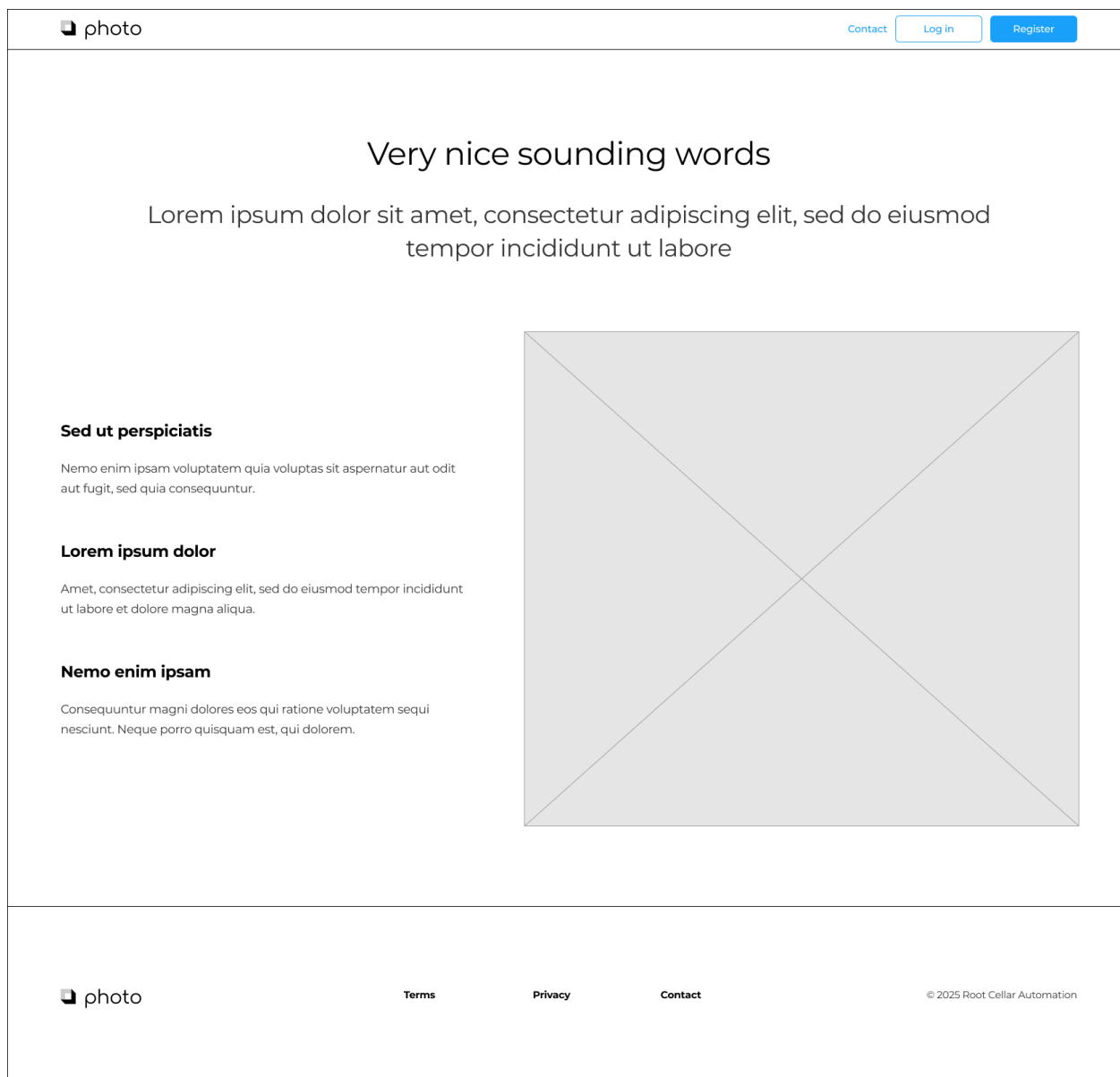
### 3.7 Бүлгийн дүгнэлт

Энэхүү бүлэгт системийн бүтэц, үйл ажиллагааны шаардлагууд болон өгөгдлийн урсгал, харилцааг тодорхойлж, цаашдын хөгжүүлэлтийн суурь судалгааг бүрэн гүйцэтгэлээ. Энэ нь системийн хөгжүүлэлтийн дараагийн шат болох архитектурын төлөвлөлт, хэрэгжилтийн ажилд үндэс суурь болж өгнө.

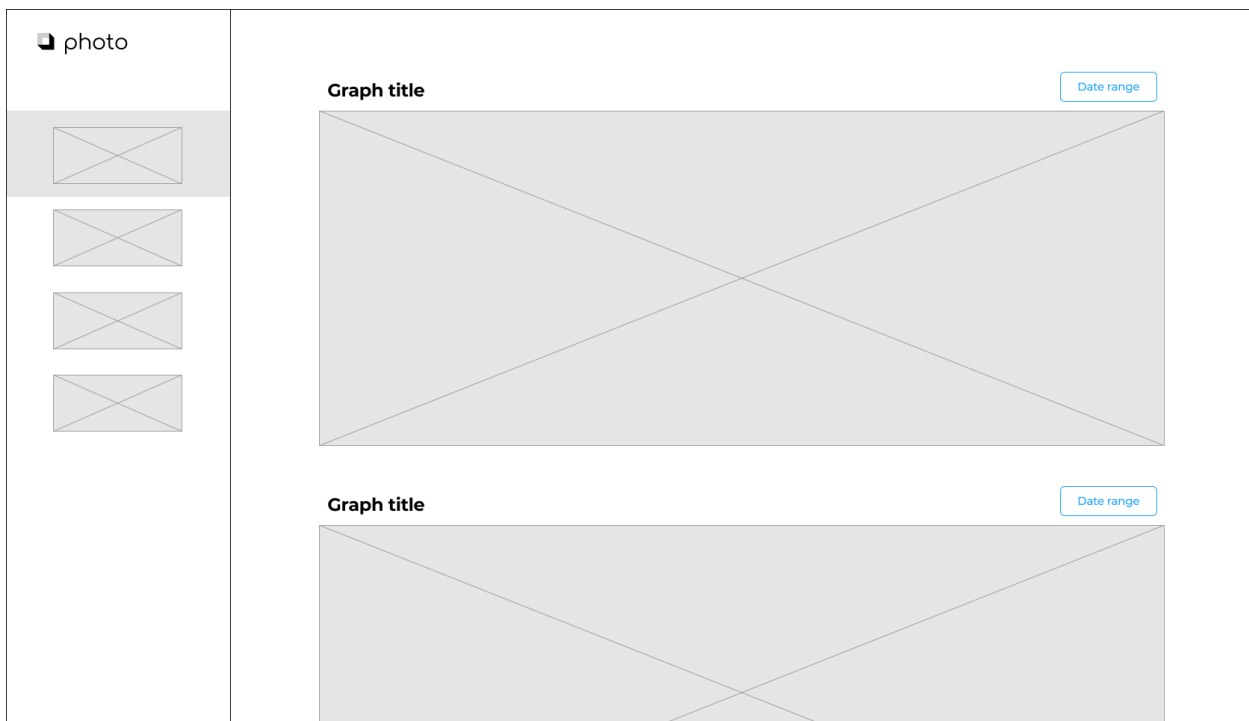
# 4. СИСТЕМИЙН ЗОХИОМЖ

## 4.1 Дэлгэцийн зохиомж

Энэхүү хэсэгт дэлгэцийн зохиомжийн хар зургийг үзүүлнэ.



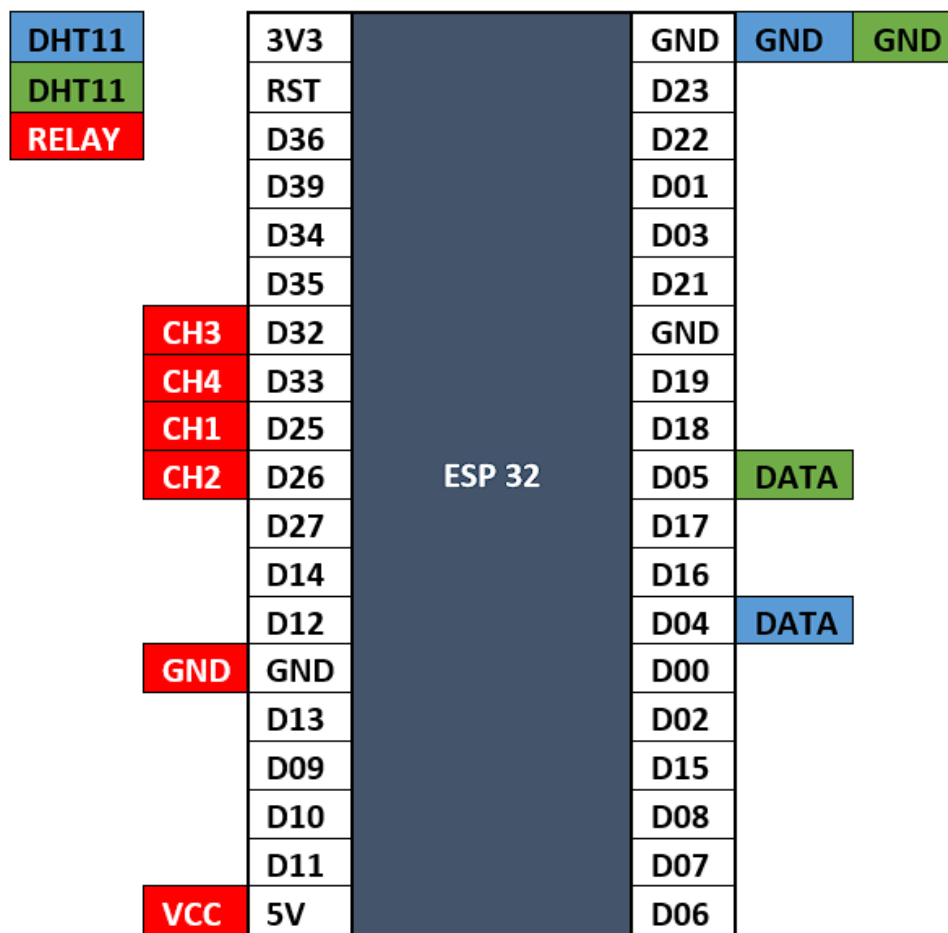
Зураг 4.1: "Нүүр" хуудас



Зураг 4.2: "Удирдлагын самбар" хуудас

## 4.2 Техник хангамжийн зохиомж

Доорх диаграмм нь esp32-н хөлүүд дээрх бусад модулиудын аль хөл холбогдохыг харуулсан диаграмм юм. Үүнд 2 чийгшил мэдрэгч DHT11 мэдрэгч, 4 сувагт 1 релей модуль багтсан. Үүн дээр цаашид 2 ширхэг DHT22 мэдрэгч, халаагч, хөргүүрийг төлөөлүүлэх LED гэрлүүд нэмэгдэнэ.

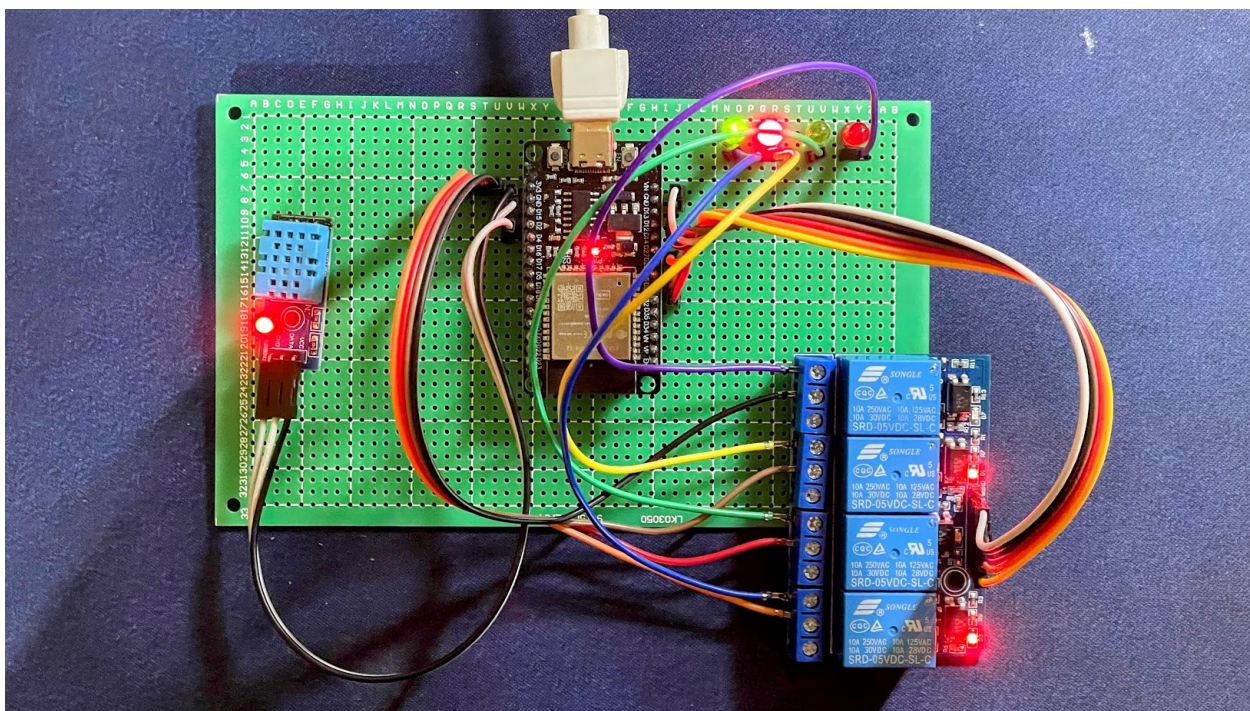


Зураг 4.3: ESP32 нэгдсэн хэлхээний хөлүүдийн байрлалыг харуулсан диаграмм

## 5. ХЭРЭГЖҮҮЛЭЛТ

Техник хангамжийн зохиомжид үзүүлснийг бодитоор угсран бэлдсэн байгаа. Эдгээр релей нь хөргүүр, халаагуур, агаар чийгшүүлэгч, агааржуулагч зэрэг төхөөрөмжүүдийн унтраалга болон ажиллах юм.

ESP32 нь өгөгдөл өөрчлөгдөхөд үүнийгээ MQTT протокол ашиглан MQTT Broker-рүү илгээдэг.



Зураг 5.1: ESP32-н угсарсан байдал

MQTT протокол ашиглан ESP32-н илгээж буй мэлээллийг MQTT broker-с авч backend-с өгөгдлийн сан руу хадгалж байгаа.

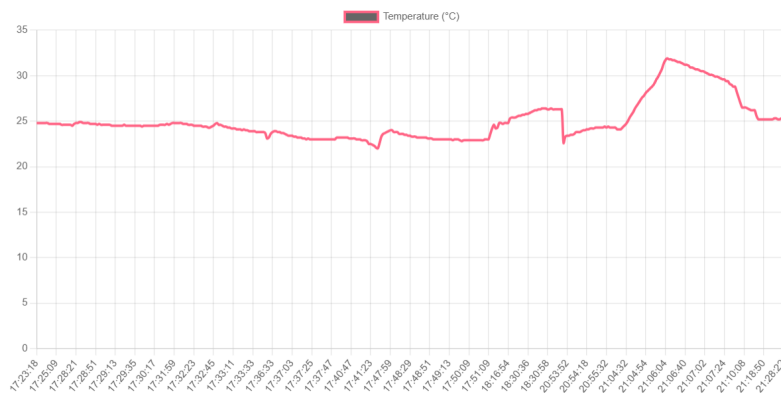
Frontend секунд тутам Backend-с өгөгдлийг авч харагдцыг шинэчилнэ. Ингэснээр бодит цагийн мэдээлэл авна.

## Cellar Dashboard

### Temperature



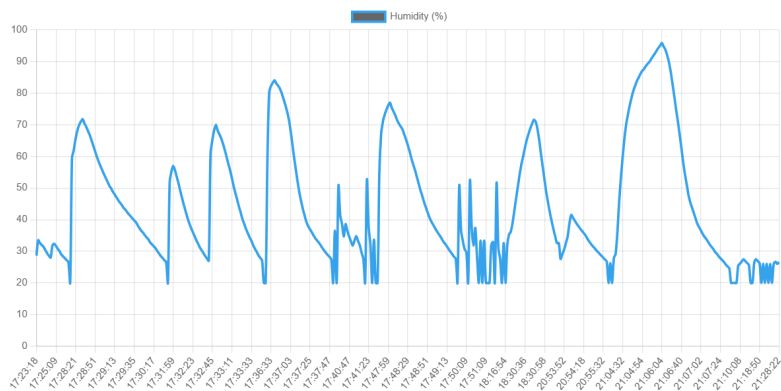
25.3 °C



### Humidity



26.6 %



Heater



OFF

Cooler



ON

Humidifier



ON

Ventilation



OFF

Зураг 5.2: Өгөгдөл болон релэйн мэдээллийг харуулсан байдал

Цагираг графикаар одоогийн байгаа дулаан болон чийгшлийн мэдээллийг, зураасан графикар нийт дулаан болон чийгшлийн өөрчлөлтийг дүрслэн харуулсан. Доор байх 4 гэрэл нь халаагуур, хөргүүр, чийгшүүлэгч, агааржуулагчийг тус тус төлөөлөнө. Ногоон байвал ассан, улаан байвал унтарсан гэдгийг илтгэнэ.



# Дүгнэлт

Энэхүү системийн судалгаа, шинжилгээ, зохиомж болон хэрэгжүүлэлтийн явцад үндсэн зорилго болох зоорины автоматжуулалтын системийг хөгжүүлэхэд шаардлагатай техник, программ хангамжийн шийдлүүдийг тодорхойлон, туршилтын загварыг боловсрууллаа.

Системийн үндсэн хэсгүүд нь:

- Температур, чийгшил хэмжих DHT11 мэдрэгч
- Дулааны хяналт, агааржуулалтын удирдлага хийх 4 сувгийн релей
- Хяналтын дохио үзүүлэх LED гэрлүүд
- ESP32 микроконтроллер ашиглан мэдээллийг боловсруулах, хянах
- Технологийн судалгааны үр дүнд ESP32, MySQL, Spring Boot, Vue.js ашиглах боломжийг судалж, хамгийн тохиромжтой шийдлүүдийг сонгов. Системийн шинжилгээний үе шатанд төхөөрөмж болон программ хангамжийн шаардлагыг тодорхойлж, Use-Case, ERD болон өгөгдлийн урсгалын диаграммуудыг гаргав.

Хэрэгжүүлэлтийн явцад:

- MQTT topic-ууд ашиглаж эхэлсэн
  - esp32/data – мэдрэгчийн мэдээлэл
  - esp32/config – серверээс ирсэн тохиргоо
  - esp32/relay/status – одоогийн релейний төлөв
- ESP32 дээр дараах өөрчлөлтүүдийг хийсэн
  - WiFi болон MQTT брокерт холбогдоно.

- DHT11 мэдрэгчээс температурыг болон чийгшлийг уншина.
  - minTemp, maxTemp, minHum, maxHum утгууд дээр үндэслэн хяналтын логик ажиллуулна.
  - MQTT-ээр ирж буй тохиргооны мэдээллийг (esp32/config) хүлээн авч, утгуудыг шинэчилнэ.
  - Релейн төлвийг хадгалж, зөвхөн өөрчлөгдсөн үед нь MQTT-д илгээнэ.
  - Температур, чийгшлийн утга өөрчлөгдөх үед л мэдрэгчийн мэдээллийг илгээнэ.
- Backend дээр
    - esp32/data, esp32/relay/status сэдвүүдэд subscribe хийнэ.
    - JSON мэдээллийг задлаж, мэдрэгч болон релейний мэдээллийг өгөгдлийн санд хадгална.
    - SensorData модель, репозитор, үйлчилгээ бичигдсэн.
    - RelayStatus модель, репозитор, үйлчилгээ бичигдсэн.
    - Мэдрэгч болон релейний өөрчлөлтүүдийг орж ирсэн цагтай нь бүртгэнэ.
    - MQTT-ийн JSON өгөгдлийг JPA энтитид хөрвүүлнэ.
  - Frontend дээр
    - Хамгийн сүүлийн мэдрэгчийн мэдээллийг графикаар харуулна.
    - Релейний төлөвийг бодит цагийн байдлаар харуулна.

Дүгнэж хэлэхэд, системийн үндсэн хэсгүүд амжилттай хөгжүүлэгдэж, цаашдын өргөтгөл, интеграцид бэлэн болсон. Үлдсэн ажилд хэрэглэгчээс тохиргооны мэдээлэл авах view үүсгэх, хэрэглэгчийг бүртгэх хэсэгийг хөгжүүлэх, системд гэмтэл гарах, хэт халалт хөрөлт зэрэгт хэрэглэгчид мэдэгдэл явуулдаг болох шаардлагатай.

# Bibliography

- [1] Espressif Systems. (2020). *ESP32 Technical Reference Manual*. <https://www.espressif.com/en/products/socs/esp32/resources>
- [2] Aosong Electronics. (2018). *DHT22 Temperature & Humidity Sensor Datasheet*. <https://cdn.sparkfun.com/assets/f/7/d/9/c/DHT22.pdf>
- [3] Spring.io. (2024). *Spring Boot Documentation*. <https://docs.spring.io/spring-boot/docs/current/reference/html/>
- [4] Vue.js. (2024). *Vue.js Documentation*. <https://vuejs.org/guide/introduction.html>
- [5] MySQL. (2024). *MySQL Documentation*. <https://dev.mysql.com/doc/>
- [6] Inkbird. (n.d.). *ITH-20R Wireless Thermometer and Hygrometer*. <https://www.ink-bird.com/>
- [7] Sonoff. (n.d.). *TH16 Smart Switch*. <https://sonoff.tech/product/smart-home/smart-switches/th16/>
- [8] GitHub. (n.d.). *Cellar Control System by Dulguun*. <https://github.com/Dulguun06/cellar-control-system>

# A. КОДЫН ХЭРЭГЖҮҮЛЭЛТ

```
1 #include <WiFi.h>
2 #include <PubSubClient.h>
3 #include <DHT.h>
4 #include <ArduinoJson.h>
5 #include <WiFiClientSecure.h>
6
7 // WiFi credentials
8 const char *ssid = "XXXXXXXXXX";
9 const char *password = "XXXXXXX";
10
11 // MQTT broker
12 const char *mqttServer = "XXX.XXX.XXX.XXX";
13 const int mqttPort = 1883;
14
15 const char *mqtt_user = "XXXX";
16 const char *mqtt_pass = "XXXXXX";
17
18 const char *root_ca = R"EOF(
19 -----BEGIN CERTIFICATE-----
20 MIIETTCCAzWgAwIBAgIU5I6cDJLEz3r0VuIgKhH5QzBXlmoWdQYJKoZIhvcNAQEL
21 BQAwgbUxCzAJBgNVBAYTAk10MRQwEgYDVQQIDAtVbGFhbmJhYXRhcjENMAsGA1UE
22 BwwEY2l0eTEoTEoMcyGA1UECgwFTmF0aW9uYWwgVW5pdWVyc2l0eSBvZiBNb25nb2xp
23 YTEbMBkGA1UECwwSRW5naW5lZXJpbmcgU2Nob29sMRAwDgYDVQQDDAdEdWxndXVu
24 MSgwJgYJKoZIhvcNAQkBFhkdWxndXVumF0Ym9sZDFAZ21haWwuY29tMB4XDTI1
25 MDUwNTEzMTIOM1oXDTI2MDUwNTEzMTIOM1owgbUxCzAJBgNVBAYTAk10MRQwEgYD
26 VQQIDA|.....
27 .....CLEARED THIS PART BECAUSE OF SECURITY REASONS.....
28 .....|sxvBn11vH
29 QxbGquZiUcZCTl1hoQkw7uJp40BUv65MS4wnbxbrG0jcgozAS640F9nt6XjPD101
30 e1r0/zLT336Gto5xZmpK1ybix6tCpGtBtPbgV+x6bDCkDgJ9QcI+aFAGFM30MhSs
31 CXcpIA9C1PeR4cIn9jflsOK7gl4/odyyiUKsxWhomwGXLTOAypk6ITN7h0eII8lR
32 8dQsASq9wjN2CyaimCsDSLzfMvKf6KFPaeHvTkEN3g6jSP/6x2oKqCuQd8ufl/5f
33 3croBvAsKlh1JIwpoCMikI81cbVWzvyK1ydRGY5lc2y6VynDd79enHb+t4shBcPG
34 aw==
35 -----END CERTIFICATE-----
36 )EOF";
37
38 // MQTT topics
39 const char *dataTopic = "esp32/data";
40 const char *configTopic = "esp32/config";
41 const char *relayStatusTopic = "esp32/relay/status";
42
43 // DHT settings
44 #define DHTPIN 4
45 #define DHTTYPE DHT11
46
47 // Relay pins
48 #define HEATER 25
```

```

49 #define COOLER 26
50 #define HUMIDIFIER 32
51 #define VENTILATION 33
52
53 WiFiClientSecure espClient;
54 PubSubClient client(espClient);
55 DHT dht(DHTPIN, DHTTYPE);
56
57 // Default thresholds (can be updated by MQTT config)
58 float minTemp = 4.0, maxTemp = 15.0;
59 float minHum = 80.0, maxHum = 95.0;
60
61 unsigned long lastSendTime = 0;
62 const unsigned long sendInterval = 1000;
63
64 // Store previous relay states
65 bool lastHeater = false;
66 bool lastCooler = false;
67 bool lastHumidifier = false;
68 bool lastVentilation = false;
69
70 //Sensor Data
71 float lastTemp = NAN;
72 float lastHum = NAN;
73 const float changeThreshold = 0.5; // Adjust based on how sensitive
    you want it
74
75 void setup() {
76     Serial.begin(115200);
77     dht.begin();
78
79     pinMode(HEATER, OUTPUT);
80     pinMode(COOLER, OUTPUT);
81     pinMode(HUMIDIFIER, OUTPUT);
82     pinMode(VENTILATION, OUTPUT);
83
84     // Turn everything off initially
85     digitalWrite(HEATER, LOW);
86     digitalWrite(COOLER, LOW);
87     digitalWrite(HUMIDIFIER, LOW);
88     digitalWrite(VENTILATION, LOW);
89
90     connectToWiFi();
91
92     configTime(0, 0, "pool.ntp.org"); // Set time via NTP
93     while (time(nullptr) < 8 * 3600 * 2) {
94         delay(500);
95         Serial.print(".");
96     }
97     Serial.println("Time synchronized.");
98
99     espClient.setCACert(root_ca);

```

```

100 client.setServer(mqttServer, mqttPort);
101 client.setCallback([](char *topic, byte *payload, unsigned int length
    ) {
102     if (String(topic) == configTopic) {
103         payload[length] = '\0';
104         handleConfigUpdate((char *)payload);
105     }
106 });
107
108 connectToMQTT();
109 client.subscribe(configTopic);
110 }
111
112 void loop() {
113     if (!client.connected()) connectToMQTT();
114     client.loop();
115
116     unsigned long now = millis();
117     if (now - lastSendTime >= sendInterval) {
118         sendSensorData();
119         lastSendTime = now;
120     }
121 }
122
123 void connectToWiFi() {
124     Serial.print("Connecting to WiFi...");
125     WiFi.begin(ssid, password);
126     while (WiFi.status() != WL_CONNECTED) {
127         delay(500);
128         Serial.print(".");
129     }
130     Serial.println(" connected!");
131 }
132
133 void connectToMQTT() {
134     while (!client.connected()) {
135         Serial.print("Connecting to MQTT...");
136         if (client.connect("ESP32Client", mqtt_user, mqtt_pass)) {
137             Serial.println(" connected!");
138             client.subscribe(configTopic);
139         } else {
140             Serial.print(" failed, rc=");
141             Serial.println(client.state());
142             delay(5000);
143         }
144     }
145 }
146
147 void handleConfigUpdate(char *payload) {
148     StaticJsonDocument<256> doc;
149     DeserializationError error = deserializeJson(doc, payload);
150     if (error) {

```

```

151     Serial.println("Failed to parse config JSON");
152     return;
153 }
154
155 minTemp = doc["minTemp"] | minTemp;
156 maxTemp = doc["maxTemp"] | maxTemp;
157 minHum = doc["minHum"] | minHum;
158 maxHum = doc["maxHum"] | maxHum;
159
160 Serial.println("Updated config:");
161 Serial.printf("Temp: %.2f - %.2f | Hum: %.2f - %.2f\n", minTemp,
    maxTemp, minHum, maxHum);
162 }
163
164 void applyControl(float temp, float hum) {
165     bool isCold = temp < minTemp;
166     bool isHot = temp > maxTemp;
167     bool isHumid = hum > maxHum;
168     bool isDry = hum < minHum;
169
170     bool heater = isCold && !isHumid;
171     bool cooler = isHot;
172     bool humidifier = isDry && !isHot;
173     bool ventilation = isHumid || (isCold && isHumid);
174
175     digitalWrite(HEATER, heater ? HIGH : LOW);
176     digitalWrite(COOLER, cooler ? HIGH : LOW);
177     digitalWrite(HUMIDIFIER, humidifier ? HIGH : LOW);
178     digitalWrite(VENTILATION, ventilation ? HIGH : LOW);
179
180     // Send MQTT only if any status has changed
181     if (heater != lastHeater || cooler != lastCooler || humidifier !=
        lastHumidifier || ventilation != lastVentilation) {
182         sendRelayStatus(heater, cooler, humidifier, ventilation);
183         lastHeater = heater;
184         lastCooler = cooler;
185         lastHumidifier = humidifier;
186         lastVentilation = ventilation;
187     }
188 }
189
190 void sendSensorData() {
191     float temp = dht.readTemperature();
192     float hum = dht.readHumidity();
193
194     Serial.printf("Sensor: %.2f°C, %.2f%%\n", temp, hum);
195
196     applyControl(temp, hum);
197
198     // Only send if temp or hum changed more than the threshold
199     if (isnan(lastTemp) || isnan(lastHum) || abs(temp - lastTemp) >
        changeThreshold || abs(hum - lastHum) > changeThreshold) {

```

```

200
201     StaticJsonDocument<256> doc;
202     JsonObject sensor = doc.createNestedObject("sensor");
203     sensor["temp"] = temp;
204     sensor["hum"] = hum;
205
206     char buffer[256];
207     size_t len = serializeJson(doc, buffer);
208     client.publish(dataTopic, buffer, len);
209
210     Serial.println("Sensor data changed → sent to MQTT.");
211
212     // Update previous values
213     lastTemp = temp;
214     lastHum = hum;
215 } else {
216     Serial.println("No change in sensor data → MQTT not sent.");
217 }
218 }
219
220
221 void sendRelayStatus(bool heater, bool cooler, bool humidifier, bool
    ventilation) {
222     StaticJsonDocument<128> doc;
223     doc["heater"] = heater;
224     doc["cooler"] = cooler;
225     doc["humidifier"] = humidifier;
226     doc["ventilation"] = ventilation;
227
228     char buffer[128];
229     size_t len = serializeJson(doc, buffer);
230     client.publish(relayStatusTopic, buffer, len);
231
232     Serial.println("Relay status updated:");
233     Serial.println(buffer);
234 }

```

Код А.1: ESP32-т зориулан бичсэн код

```

1 package num.edu.cellar.config;
2
3 import org.eclipse.paho.client.mqttv3.MqttClient;
4 import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
5 import org.eclipse.paho.client.mqttv3.MqttException;
6 import org.springframework.context.annotation.Bean;
7 import org.springframework.context.annotation.Configuration;
8
9 @Configuration
10 public class MqttConfig {
11
12     @Bean
13     public MqttClient mqttClient() throws MqttException {

```



```

14     String broker = "tcp://xxx.xxx.x.x:xxxx"; //MQTT Broker - ip
15     String clientId = "SpringBootClient";
16     MqttClient client = new MqttClient(broker, clientId);
17     MqttConnectOptions options = new MqttConnectOptions();
18     options.setAutomaticReconnect(true);
19     options.setCleanSession(true);
20
21     client.connect(options);
22     return client;
23 }
24 }

```

Код A.2: MqttConfig.java тохиргоо

```

1 package num.edu.cellar.config;
2
3 import org.springframework.context.annotation.Configuration;
4 import org.springframework.web.servlet.config.annotation.*;
5
6 @Configuration
7 @EnableWebMvc
8 public class WebConfig implements WebMvcConfigurer {
9
10     @Override
11     public void addCorsMappings(CorsRegistry registry) {
12         registry.addMapping("/api/**")
13             .allowedOrigins("*")
14             .allowedMethods("GET", "POST", "PUT", "DELETE") //
15                 Allowed HTTP methods
16             .allowedHeaders("*") // Allowed request headers
17             .allowCredentials(false)
18             .maxAge(3600);
19     }
20 }

```

Код A.3: webconfig.java CORS тохиргоо

```

1 package num.edu.cellar.model;
2
3 import jakarta.persistence.*;
4 import lombok.*;
5
6 import java.time.LocalDateTime;
7 import java.util.Date;
8
9 @Entity
10 @Data
11 @Builder
12 @NoArgsConstructor
13 @AllArgsConstructor
14 public class SensorData {
15
16     @Id

```

```

17     @GeneratedValue(strategy = GenerationType.IDENTITY)
18     private Long id;
19     private float temperature;
20     private float humidity;
21     private Date timestamp;
22 }

```

Код А.4: SensorData.java мэдрэгчийн мэдээллийг хадгалах объектын ашиглах класс

```

1 package num.edu.cellar.controller;
2
3 import lombok.RequiredArgsConstructor;
4 import num.edu.cellar.model.SensorData;
5 import num.edu.cellar.repository.SensorDataRepository;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RestController;
9
10 import java.util.List;
11
12 @RestController
13 @RequestMapping("/api/sensor")
14 @RequiredArgsConstructor
15 public class SensorDataController {
16
17     private final SensorDataRepository repository;
18
19     @GetMapping("/all")
20     public List<SensorData> getAllData() {
21         return repository.findAll();
22     }
23
24     @GetMapping("/latest")
25     public SensorData getLatestData() {
26         return repository.findTopByOrderByTimestampDesc();
27     }
28 }

```

Код А.5: SensorDataController.java мэдрэгчүүдийн мэдээллийг авахад ашиглах endpoint-ууд

```

1 package num.edu.cellar.service;
2
3 import com.fasterxml.jackson.databind.JsonNode;
4 import com.fasterxml.jackson.databind.ObjectMapper;
5 import jakarta.annotation.PostConstruct;
6 import lombok.RequiredArgsConstructor;
7 import num.edu.cellar.model.RelayStatus;
8 import num.edu.cellar.model.SensorData;
9 import num.edu.cellar.repository.RelayStatusRepository;
10 import num.edu.cellar.repository.SensorDataRepository;
11 import org.eclipse.paho.client.mqttv3.*;
12 import org.springframework.stereotype.Service;
13

```

```

14 import java.sql.Date;
15
16 @Service
17 @RequiredArgsConstructor
18 public class MqttListenerService {
19
20     private final MqttClient mqttClient;
21     private final SensorDataRepository repository;
22     private final RelayStatusRepository relayStatusRepository;
23     private final ObjectMapper objectMapper = new ObjectMapper();
24
25     @PostConstruct
26     public void subscribeToTopic() throws MqttException {
27         mqttClient.subscribe("esp32/data", this::handleSensorData);
28         mqttClient.subscribe("esp32/relay/status", this::
                handleRelayStatus);
29     }
30
31     private void handleSensorData(String topic, MqttMessage message) {
32         {
33             String payload = new String(message.getPayload());
34             System.out.println("Received: " + payload);
35
36             try {
37                 JsonNode root = objectMapper.readTree(payload);
38                 JsonNode sensor = root.get("sensor");
39
40                 if (sensor != null && sensor.has("temp") && sensor.has(
41                     "hum")) {
42                     SensorData data = SensorData.builder()
43                         .temperature((float) sensor.get("temp").
44                             asDouble())
45                         .humidity((float) sensor.get("hum").
46                             asDouble())
47                         .timestamp(new Date(System.
48                             currentTimeMillis()))
49                         .build();
50
51                     repository.save(data);
52                     System.out.println("Sensor data saved to DB.");
53                 } else {
54                     System.err.println("Invalid sensor data format.");
55                 }
56             } catch (Exception e) {
57                 System.err.println("Failed to parse or save: " + e.
58                     getMessage());
59             }
60         }
61     }
62
63     private void handleRelayStatus(String topic, MqttMessage message) {
64         String payload = new String(message.getPayload());

```

```

60     System.out.println("Relay Status Received: " + payload);
61
62     try {
63         JsonNode root = objectMapper.readTree(payload);
64
65         RelayStatus status = RelayStatus.builder()
66             .heater(root.get("heater").asBoolean())
67             .cooler(root.get("cooler").asBoolean())
68             .humidifier(root.get("humidifier").asBoolean())
69             .ventilation(root.get("ventilation").asBoolean())
70             .timestamp(new Date(System.currentTimeMillis()))
71             .build();
72         relayStatusRepository.save(status);
73         System.out.println("Relay status saved.");
74     } catch (Exception e) {
75         System.err.println("Failed to parse relay status: " + e.
76             getMessage());
77     }
78 }

```

Код A.6: MqttListenerService.java релей болон мэдрэгчийн мэдээллийг ESP32-с авах

```

1 package num.edu.cellar.repository;
2
3 import num.edu.cellar.model.SensorData;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 import java.util.List;
7
8 public interface SensorDataRepository extends JpaRepository<SensorData,
9     Long> {
10
11     SensorData findTopByOrderByTimestampDesc();
12 }

```

Код A.7: SensorDataRepository.java өгөгдлийн сангаас өгөгдөл авах

Frontend-н кодууд

```

1 import axios from 'axios'
2
3 const API_URL = 'http://localhost:8080/api'
4
5 export const getLatestSensorData = () => {
6     return axios.get(`${API_URL}/sensor/latest`)
7 }
8 export const getAllSensorData = ()=>{
9     return axios.get(`${API_URL}/sensor/all`)
10 }
11 export const getRelayStatus = () => {
12     return axios.get(`${API_URL}/relay/latest`)
13 }

```

Код A.8: api.js backend-с өгөгдөл татахад ашиглагдана.

Home.vue нь vue component-уудыг нэгтгэх гол суурь нь юм. Эндээс Navbar ашиглан бусад хуудсуудыг өөр дээрээ харуулах юм.

```
1 <template>
2   <div>
3     <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
4       <div class="container-fluid">
5         <a class="navbar-brand" href="#">Cellar Control</a>
6         <button
7           class="navbar-toggler"
8           type="button"
9           data-bs-toggle="collapse"
10          data-bs-target="#navbarNav"
11          aria-controls="navbarNav"
12          aria-expanded="false"
13          aria-label="Toggle navigation"
14        >
15          <span class="navbar-toggler-icon"></span>
16        </button>
17        <div class="collapse navbar-collapse" id="navbarNav">
18          <ul class="navbar-nav ms-3">
19            <li class="nav-item">
20              <a class="nav-link" :class="{ active: activeTab === '
21                realtime' }" href="#" @click="activeTab = 'realtime'">
22                Dashboard</a>
23            </li>
24            <li class="nav-item">
25              <a class="nav-link" :class="{ active: activeTab === '
26                config' }" href="#" @click="activeTab = 'config'">
27                Configurations</a>
28            </li>
29            <li class="nav-item">
30              <a class="nav-link" :class="{ active: activeTab === '
31                profile' }" href="#" @click="activeTab = 'profile'">
32                User Profile</a>
33            </li>
34          </ul>
35        </div>
36      </div>
37    </nav>
38
39    <!-- Main Content Section -->
40    <div class="container mt-4">
41      <Dashboard v-if="activeTab === 'realtime'" />
42      <Configs v-if="activeTab === 'config'" />
43      <Profile v-if="activeTab === 'profile'" />
44    </div>
45  </div>
46</template>
47
48<script>
```

```

44 import Dashboard from './Dashboard.vue';
45 import Configs from './Configs.vue';
46 import Profile from './Profile.vue';
47
48 export default {
49   components: {
50     Dashboard,
51     Configs,
52     Profile,
53   },
54   data() {
55     return {
56       activeTab: 'realtime',
57     };
58   },
59 };
60 </script>

```

Код А.9: Home.vue

Dashboar.vue нь өөртөө DonutChart, SensorGraph, RelayStatus гэсэн 3 хэсгээс бүтэх ба эдгээрийг нэгтгэн харуулах үүрэгтэй

```

1  <div class="container py-4">
2    <h1 class="text-center mb-5 display-5 fw-bold">Cellar Dashboard</h1>
3
4    <div class="row mb-3">
5      <div class="col-md-3 mb-3 mb-md-0">
6        <DonutChart />
7      </div>
8      <div class="col-md-9">
9        <SensorGraph />
10     </div>
11   </div>
12
13   <div class="row">
14     <div class="col">
15       <RelayStatus />
16     </div>
17   </div>
18 </div>
19 </template>
20
21 <script setup>
22 import DonutChart from "@components/DonutChart.vue";
23 import SensorGraph from "@components/SensorGraph.vue";
24 import RelayStatus from "@components/RelayStatus.vue";
25 </script>

```

Код А.10: Dashboard.vue ESP32-н мэдээллийг харуулах view.

```

1 <template>
2   <div class="flex flex-col items-center row-gap-5">

```

```

3      <!-- Temperature Donut Chart -->
4      <div class="flex flex-col items-center">
5          <h2 class="text-xl font-semibold mb-2">Temperature</h2>
6          <div class="w-48 h-48">
7              <canvas ref="tempChartRef"></canvas>
8          </div>
9          <p class="text-lg font-semibold mt-2">{{ sensorData.temperature
              }} °C</p>
10     </div>
11
12     <!-- Humidity Donut Chart -->
13     <div class="flex flex-col items-center" id="hum">
14         <h2 class="text-xl font-semibold mb-2">Humidity</h2>
15         <div class="w-48 h-48">
16             <canvas ref="humidityChartRef"></canvas>
17         </div>
18         <p class="text-lg font-semibold mt-2">{{ sensorData.humidity }}
              %</p>
19     </div>
20 </div>
21 </template>
22
23 <script setup>
24 import { ref, onMounted, onBeforeUnmount } from "vue";
25 import { Chart } from "chart.js/auto";
26 import { getLatestSensorData } from "@service/api"; // Make sure to
    update this path according to your project structure
27
28 const tempChartRef = ref(null);
29 const humidityChartRef = ref(null);
30 const sensorData = ref({
31     temperature: 0, // Default value before data is fetched
32     humidity: 0     // Default value before data is fetched
33 });
34 let chartInstanceTemp = null;
35 let chartInstanceHumidity = null;
36
37 // Function to create the donut chart
38 const createDonutChart = (refElement, data, label, color) => {
39     return new Chart(refElement, {
40         type: 'doughnut',
41         data: {
42             labels: [label],
43             datasets: [{
44                 data: [data, 100 - data],
45                 backgroundColor: [color, '#e0e0e0'],
46                 borderWidth: 0,
47             }]
48         },
49         options: {
50             responsive: true,
51             plugins: {

```

```

52     tooltip: {
53         enabled: false
54     },
55     legend: {
56         display: false
57     },
58     datalabels: {
59         display: true,
60         color: '#333',
61         font: {
62             weight: 'bold',
63             size: 18,
64         },
65         formatter: (value) => value + '%', // Display percentage in
            the center
66     }
67 }
68 }
69 });
70 };
71
72 // Fetch latest sensor data
73 const fetchSensorData = async () => {
74     try {
75         const res = await getLatestSensorData(1); // 1 is the sensorId (use
            the appropriate ID)
76         sensorData.value = res.data; // Assuming the response data contains
            temperature and humidity
77         // Update the charts with new data
78         if (chartInstanceTemp) {
79             chartInstanceTemp.data.datasets[0].data = [sensorData.value.
                temperature, 100 - sensorData.value.temperature];
80             chartInstanceTemp.update();
81         }
82         if (chartInstanceHumidity) {
83             chartInstanceHumidity.data.datasets[0].data = [sensorData.value.
                humidity, 100 - sensorData.value.humidity];
84             chartInstanceHumidity.update();
85         }
86     } catch (error) {
87         console.error("Failed to fetch sensor data:", error);
88     }
89 };
90
91 // Set interval to fetch sensor data every second
92 let intervalId = null;
93
94 onMounted(() => {
95     // Create initial charts
96     chartInstanceTemp = createDonutChart(tempChartRef.value, sensorData.
        value.temperature, 'Temp', 'rgba(255, 99, 132, 1)');
97     chartInstanceHumidity = createDonutChart(humidityChartRef.value,

```



```

    sensorData.value.humidity, 'Humidity', 'rgba(54, 162, 235, 1)');
98
99 // Start fetching sensor data every second
100 intervalId = setInterval(fetchSensorData, 1000);
101 });
102
103 onBeforeUnmount(() => {
104 // Clear interval when component is unmounted
105 clearInterval(intervalId);
106
107 // Destroy charts when unmounted
108 if (chartInstanceTemp) chartInstanceTemp.destroy();
109 if (chartInstanceHumidity) chartInstanceHumidity.destroy();
110 });
111 </script>
112
113 <style scoped>
114 canvas {
115   max-width: 100%;
116 }
117 #hum{
118   margin-top: 50%;
119 }
120 </style>

```