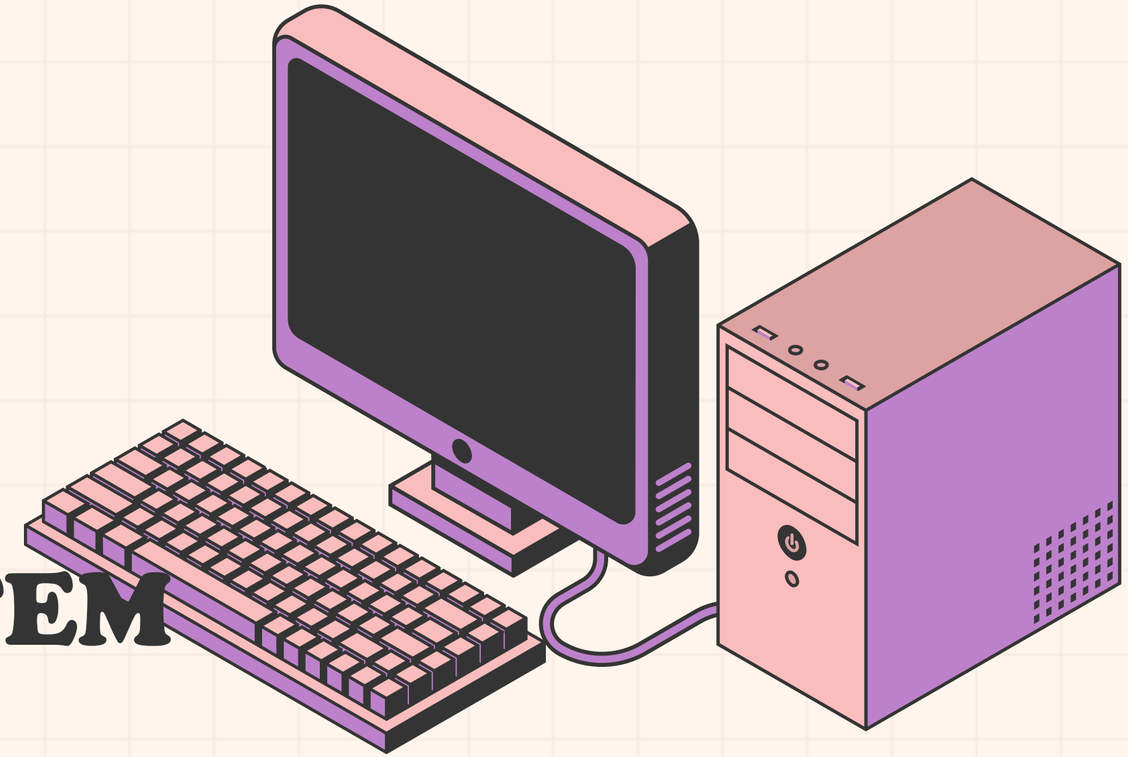
SMART HOME INTEGRATED SECURITY MANAGEMENT SYSTEM



The Smart Home Integrated Security Management System provides facial recognition-based door access control, video recording, and a touch display for weather, memos, and visitor logs.
Users can also monitor access history and control the door remotely through a web dashboard.

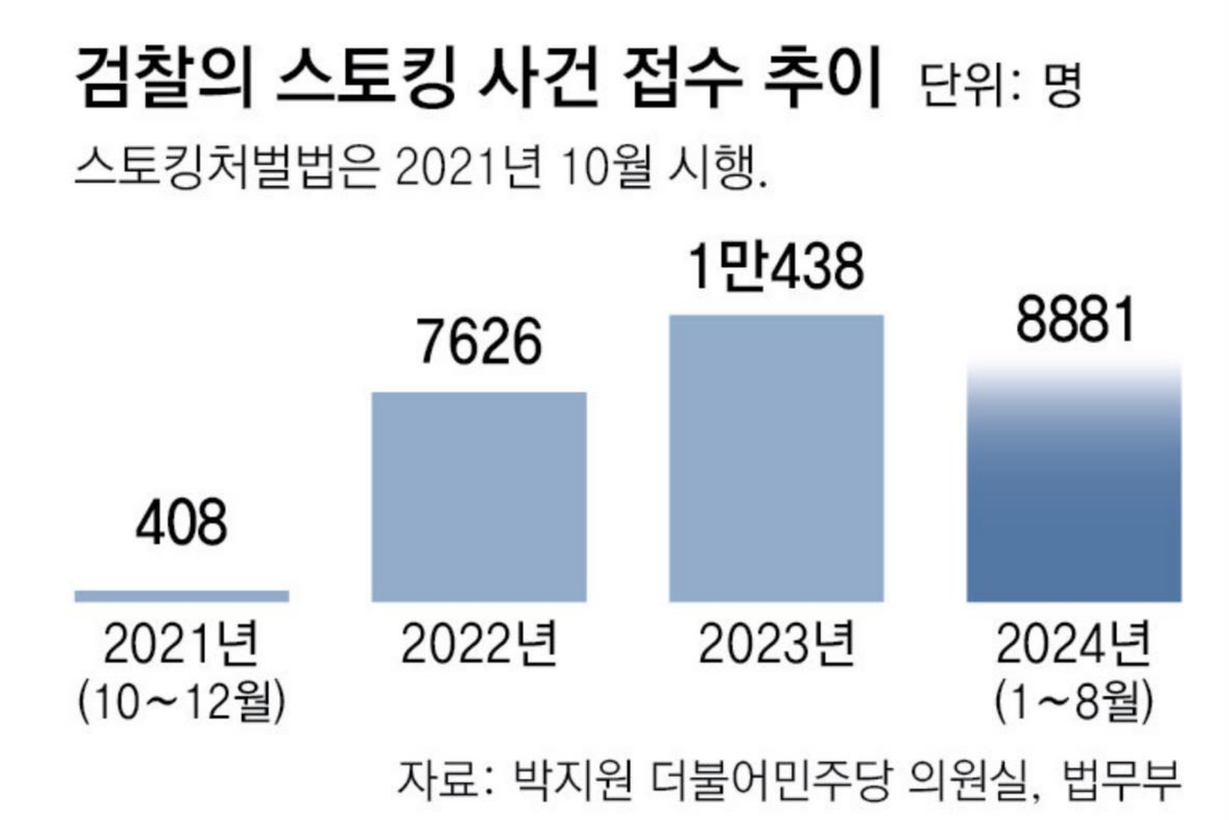
DESIGN BACKGROUND AND MOTIVATION

LG U+, 우리집지킴이 도어캠 출시…택배·화재·도난 보험 제공

(사진=LG유플러스)

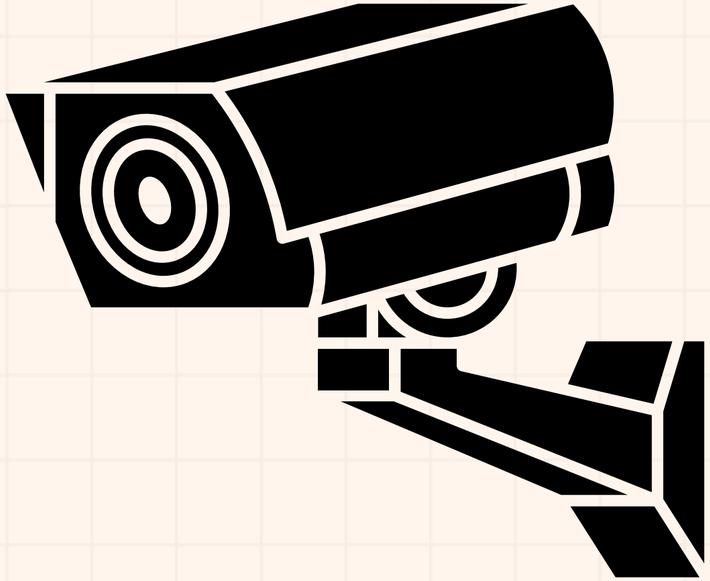
등록 2025-05-30 오전 9:43:27
수정 2025-05-30 오전 9:43:27

DESIGN BACKGROUND AND MOTIVATION II

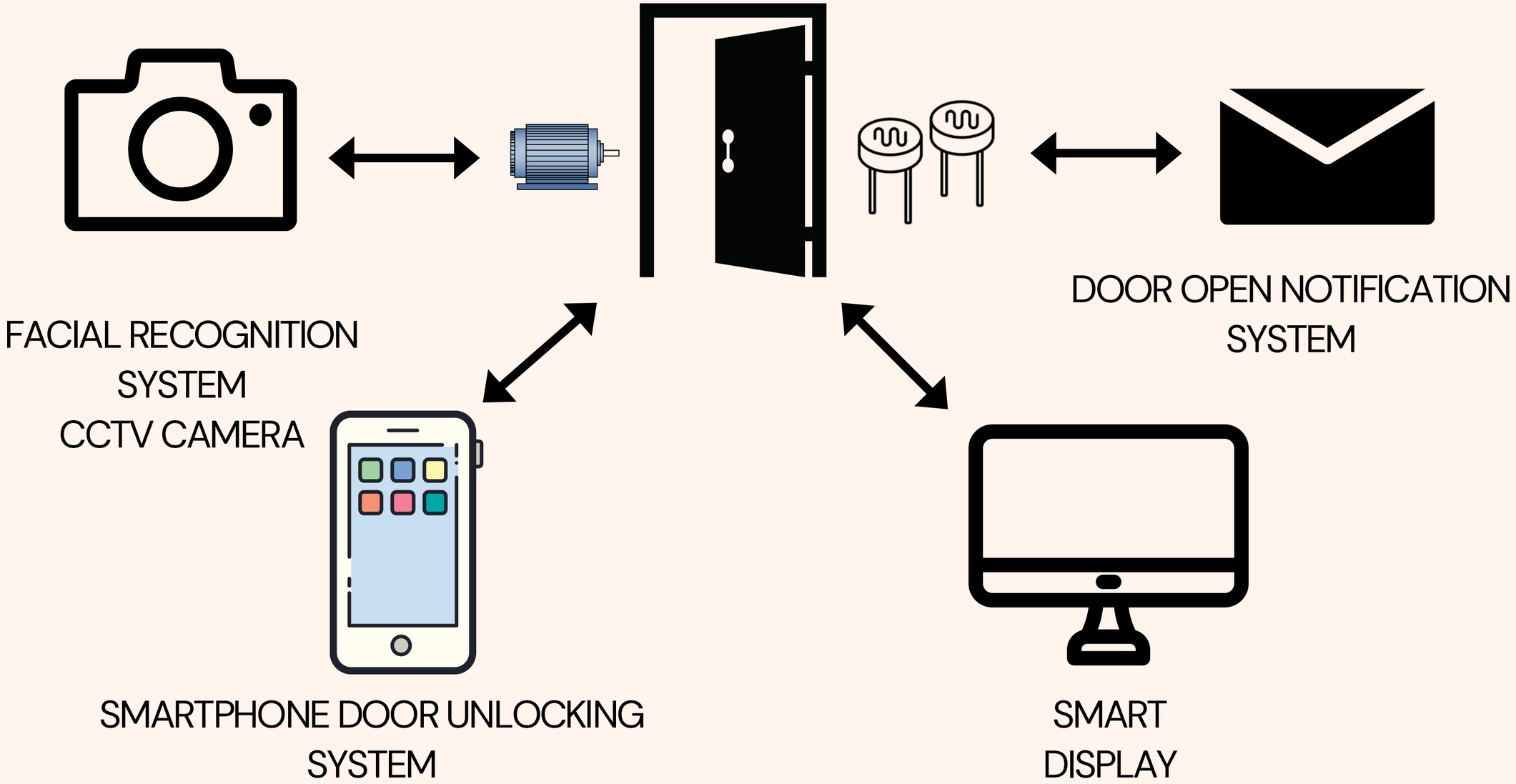


DESIGN GOAL

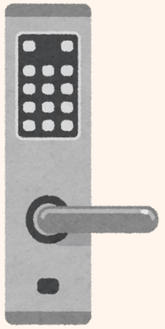
smart door lock system :
Notifies you when the door is opened and records the time it was opened
Allows you to remotely unlock the door for others without exposing your password
Features advanced facial recognition for secure access
Provides useful information such as today's weather and to-do lists through a CCTV monitor



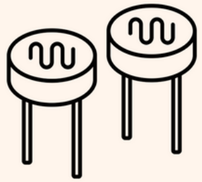
SMART HOME
INTEGRATED SECURITY MANAGEMENT SYSTEM



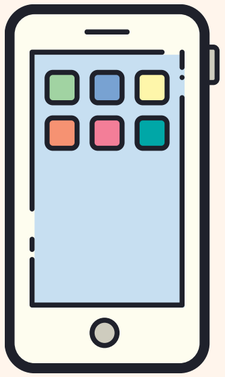
HOW IT WORKS



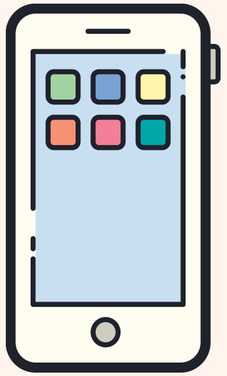
IF OPENED MANUALLY



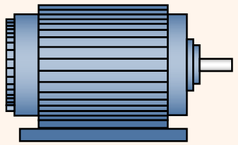
CDS CELL DETECT



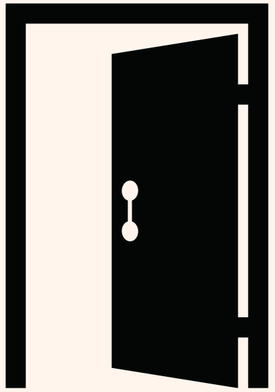
VIEWABLE ON SMARTPHONE



SMARTPHONE ACCESSIBLE

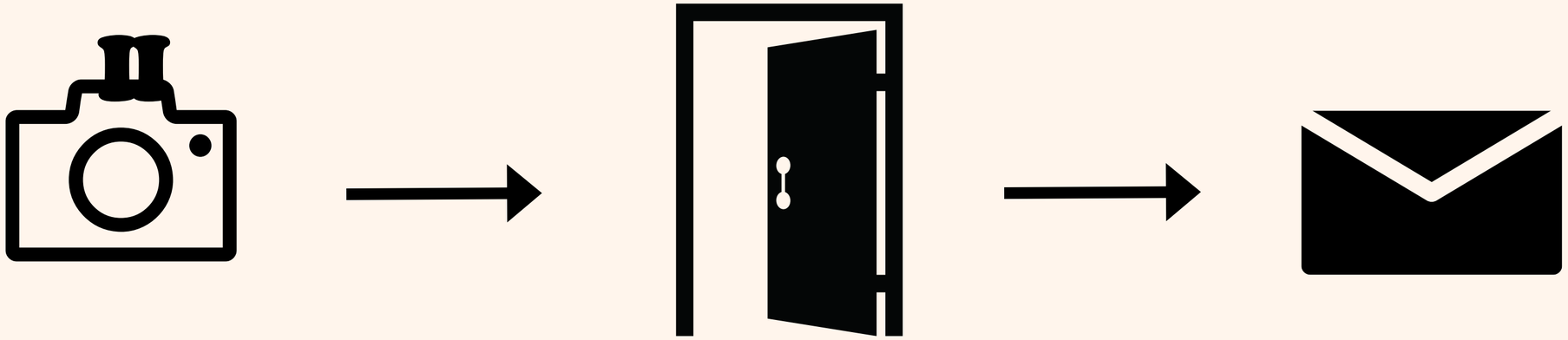


SERVO MOTOR



UNLOCK WITH SMARTPHONE

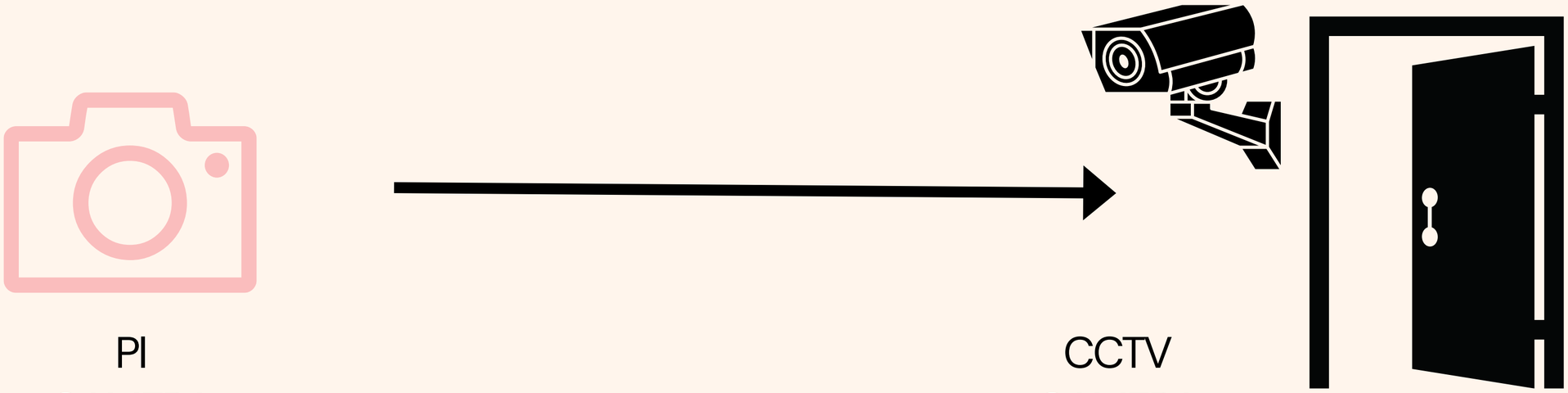
HOW IT WORKS



FACIAL RECOGNITION SYSTEM

UNLOCK

DOOR OPEN NOTIFICATION SYSTEM



PI CAMERA

CCTV CAMERA

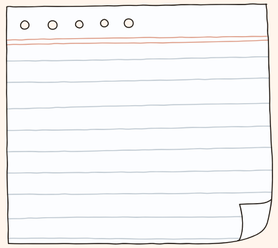
HOW IT WORKS III

CCTV MONITORING

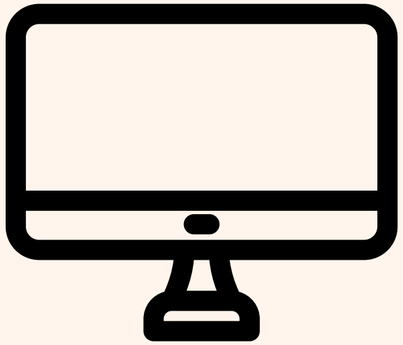
FACE REGISTRATION AND EDITING SYSTEM

DIGITAL CANVAS

USING THE WEATHER AGENCY API



NOTEPAD SYSTEM

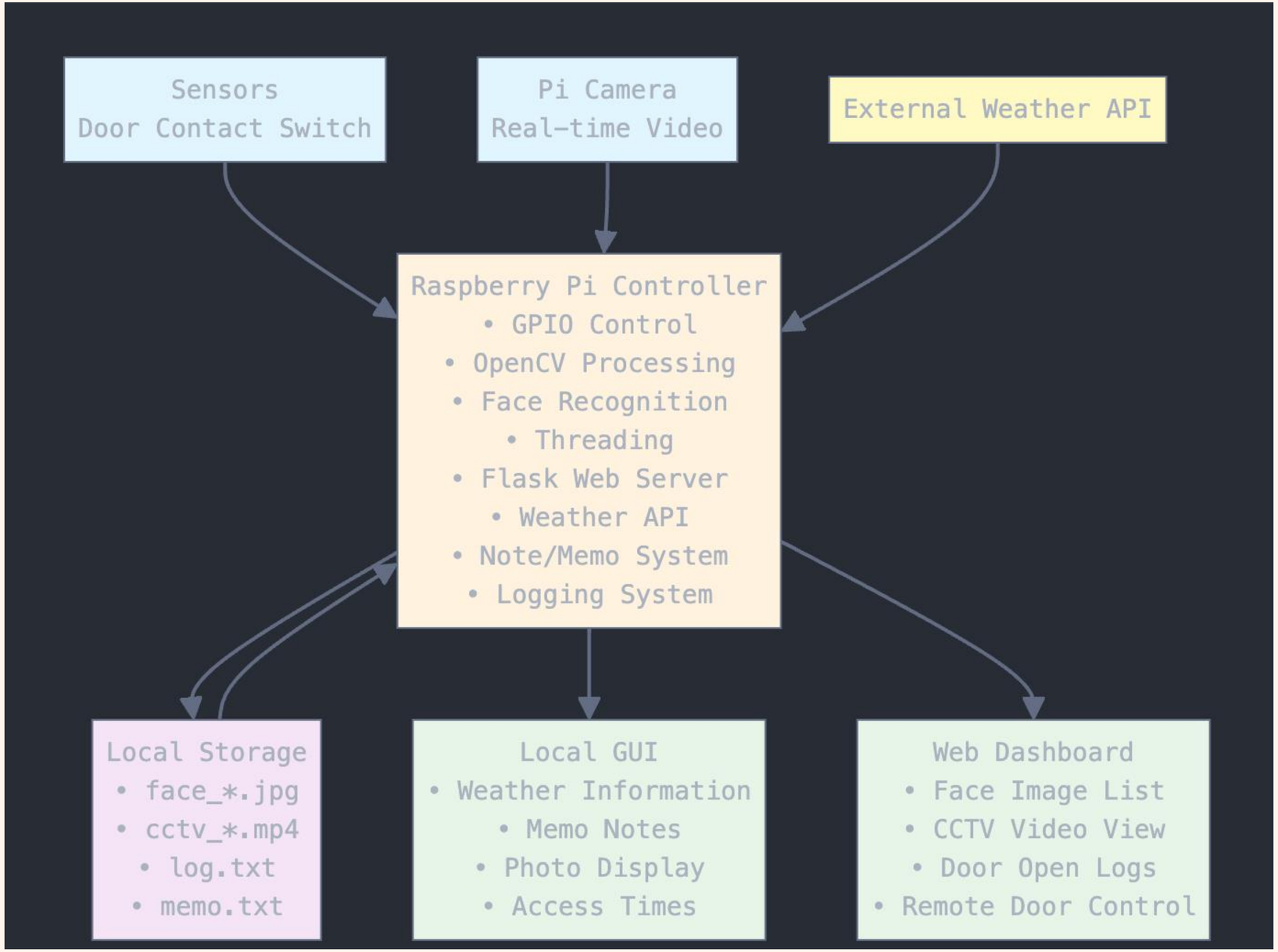


SMART DISPLAY

TECHNOLOGIES USED IN THE PROJECT

Library / Technology	Description	Icon / Logo Search Keywords
Python	Main programming language of the project	`Python logo`, `Python icon`
RPi.GPIO	Controls Raspberry Pi GPIO pins	`Raspberry Pi GPIO`, `Raspberry Pi logo`
Picamera2	Controls Raspberry Pi camera module	`Raspberry Pi Camera`, `Picamera2`
OpenCV (`cv2`)	Image processing and face detection	`OpenCV logo`, `Computer Vision icon`
face_recognition	Performs facial recognition	`Face Recognition icon`, `AI face detection`
tkinter	GUI toolkit for desktop applications	`Tkinter`, `Desktop GUI icon`
tkcalendar	Calendar widget for date selection	`Calendar icon`, `Tkinter calendar`
Pillow (PIL)	Image display and manipulation	`Pillow Python`, `Image Processing icon`
Flask	Web server framework	`Flask logo`, `Flask Python icon`
threading	Enables multithreading and async tasks	`Thread icon`, `Concurrency icon`
smtplib / email	Sends email notifications	`Email icon`, `SMTP icon`
requests / json	External API requests and JSON handling	`HTTP request icon`, `JSON icon`, `API icon`
spidev	SPI communication with external sensors	`SPI icon`, `Embedded system icon`
datetime / time	Handles timestamps and time-based logs	`Clock icon`, `Time icon`, `Calendar icon`
os	File path and directory management	`OS icon`, `Folder icon`

SYSTEM ARCHITECTURE DIAGRAM



SATISFACTION OF PROJECT REQUIREMENTS

✅ Implemented Features

🔄 End-to-End Service

Fully automated flow from input to processing, storage, notification, and control

Python | GPIO | OpenCV | Threading

📡 Wireless Communication

Stable data transmission using **Wi-Fi** connectivity

🖥️ Python GUI

User-friendly interface via **Tkinter**

Tkinter | GUI Design

⚙️ Shell Programming

Automated program execution and system startup management

Bash Scripts | System Automation

🌐 Web Server

Flask-based web server for viewing logs, face data, and CCTV footage through browser interface

Flask | Web Dashboard | REST API

SATISFACTION OF PROJECT REQUIREMENTS II

❌ Not Implemented

🗄️ SQLite Database

Not used. Instead used **local file storage** (text logs, images, videos)

💡 *Provided simplicity and minimized dependencies while meeting all requirements*

📱 Node.js Integration

Not implemented. Instead developed responsive **Flask web dashboard**

💡 *Web dashboard accessible from mobile browsers = No separate mobile app needed*

FINAL FEATURE DIFFERENCE TABLE

#	Proposed Feature	Status	Implementation Notes
1	**Facial recognition-based automatic door access system**	✅	Using `OpenCV` + `face_recognition` with Pi Camera real-time detection
2	**Real-time CCTV recording and server-side storage**	✅	OpenCV video capture + local file storage as `cctv_YYYYMMDD_HHMMSS.mp4`
3	**Remote control and smartphone app integration**	❌	`Node.js → Flask web dashboard` (responsive, mobile-accessible)
4	**Display weather info, calendar, and to-do list**	✅	Weather + photo viewer + memo GUI with `tkinter` (no calendar)
5	**Raspberry Pi 5 for system control**	✅	Central processing unit for all system operations
6	**Pi Camera Module**	✅	Using `picamera2` with OpenCV integration
7	**Door lock module**	✅	GPIO control via `setServoPos()` function
8	**Doorbell switch**	✅	GPIO input with event-triggered notifications
9	**DHT sensor for temperature/humidity**	🔄	`DHT sensor → KMA Weather API` using `requests`

FINAL FEATURE DIFFERENCE TABLE II

#	Proposed Feature	Status	Implementation Notes
10	**Display screen for user interface**	✅	`tkinter` GUI with weather, memos, photos, face info
11	**Python development**	✅	Used throughout entire system
12	**Flask web framework**	✅	Web server for logs, face list, video viewing
13	**OpenCV for face recognition**	✅	Used with `face_recognition` library
14	**SQLite database management**	❌	`SQLite → Local file storage` (images, logs, memos)
15	**SMTP email alerts**	✅	Door-open notifications via email
16	**Shell/C programming for auto-start**	✅	`a.sh` script launches `app.py` + `cvex.py`
17	**Wi-Fi wireless communication**	✅	Browser-based monitoring and control

APP.PY

```python
from flask import Flask, render_template, request
import time
import os
from datetime import datetime
from common import setServoPos, send_email_notification  # ✅ GPIO, 이메일 기능을 common에서 가져옴

app = Flask(__name__)

# 경로 설정
BASE = "/home/moomininmoon/다운로드/proj"
FACE = os.path.join(BASE, "faces")
CCTV = os.path.join(BASE, "cctv")
LOG = os.path.join(BASE, "log.txt")

# 로그 저장 함수
def log_door_open():
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(LOG, "a") as f:
        f.write(f"{now} 문 열림\n")

# 메인 페이지
@app.route("/")
def home():
    face_files = sorted(os.listdir(os.path.join("static", "faces"))) if os.path.exists(FACE) else []
    cctv_files = sorted(os.listdir(os.path.join("static", "cctv")), reverse=True) if os.path.exists(CCTV) else []

    try:
        with open(LOG, "r") as f:
            logs = f.read()
    except FileNotFoundError:
        logs = "로그 없음"

    return render_template("index.html",
                           face_files=face_files,
                           cctv_files=cctv_files,
                           logs=logs)

# 문 열기
@app.route("/unlock", methods=["POST"])
def unlock():
    setServoPos(90)
    time.sleep(1)
    setServoPos(0)
    log_door_open()
    send_email_notification()
    return "문이 열렸습니다."

# 대시보드
@app.route("/dashboard")
def dashboard():
    face_files = sorted(os.listdir(os.path.join("static", "faces"))) if os.path.exists(FACE) else []
    cctv_files = sorted(os.listdir(os.path.join("static", "cctv")), reverse=True) if os.path.exists(CCTV) else []

    try:
        with open(LOG, "r") as f:
            logs = f.read()
    except FileNotFoundError:
        logs = "로그 없음"

    return render_template("dashboard.html",
                           face_files=face_files,
                           cctv_files=cctv_files,
                           logs=logs)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

COMMON.PY

```python
# 이메일 설정
EMAIL_USER = "daeerene@gmail.com"
EMAIL_PASS = "ytji ebis trng vore"
EMAIL_TO = "qkdansgur2@naver.com"

# GPIO 설정
SERVO_PIN = 12
_initialized = False
_servo = None

def init_gpio():
    global _initialized, _servo
    if _initialized:
        return
    try:
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(SERVO_PIN, GPIO.OUT)
        _servo = GPIO.PWM(SERVO_PIN, 50)
        _servo.start(0)
        _initialized = True
        print("[GPIO] 초기화 완료")
    except Exception as e:
        print(f"[GPIO] 초기화 실패: {e}")

def setServoPos(degree):
    global _servo
    if not _initialized:
        init_gpio()
    if _servo is None:
        print("[GPIO] 서보가 초기화되지 않았습니다.")
        return

    if degree > 180:
        degree = 180
    try:
        duty = 3 + (degree * (12 - 3) / 180.0)
        _servo.ChangeDutyCycle(duty)
        time.sleep(0.3)
        _servo.ChangeDutyCycle(0)
    except Exception as e:
        print(f"[GPIO] 서보 제어 오류: {e}")

def send_email_notification():
    now = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    msg = MIMEText(f"[스마트도어락] 문이 열렸습니다.\n시간: {now}")
    msg['Subject'] = '[ 도어락 알림'
    msg['From'] = EMAIL_USER
    msg['To'] = EMAIL_TO

    try:
        server = smtplib.SMTP_SSL('smtp.gmail.com', 465)
        server.login(EMAIL_USER, EMAIL_PASS)
        server.sendmail(EMAIL_USER, EMAIL_TO, msg.as_string())
        server.quit()
        print("[이메일] 전송 성공")
    except Exception as e:
        print("[이메일] 전송 실패:", e)

def cleanup_gpio():
    try:
        if _initialized:
            GPIO.cleanup()
            print("[GPIO] cleanup 완료")
    except Exception as e:
        print(f"[GPIO] cleanup 오류: {e}")
```

CVEX.PY I

```python
import tkinter as tk
from tkinter import messagebox
from PIL import Image, ImageTk
from picamera2 import Picamera2
import cv2
import os
import face_recognition
import time
import datetime
from tkcalendar import Calendar
import datetime
import requests
import json
import RPi.GPIO as GPIO
import spidev
import threading
from common import setServoPos, send_email_notification


spi = spidev.SpiDev()
spi.open(0, 0)
spi.max_speed_hz = 1000000
spi.bits_per_word = 8



photo_dir = "/home/moomininmoon/다운로드/proj/photos"
SAVE_DIR = "faces"
CCTV_DIR = "cctv"
os.makedirs(CCTV_DIR, exist_ok=True)
os.makedirs(SAVE_DIR, exist_ok=True)
face_cascade = cv2.CascadeClassifier('/home/moomininmoon/opencv_haarcascades/haarcascade_frontalface_default.xml')

picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration(main={"format": "RGB888", "size": (640, 480)}))
picam2.start()
#조도 센서 조건 만족 함수를 추가르르르르르르르르를 해보자
def measure(channel):
    cmd = 0x40 | 0x20 | (0x10 if channel == 1 else 0x00) | 0x08
    r = spi.xfer2([cmd, 0xFF])
    val = ((r[0] & 0x03) << 8) | r[1]
    return (val * 3.3) / 1023


# 얼굴 인식용
def load_known_faces():
    encodings, names = [], []
    for f in os.listdir(SAVE_DIR):
        if f.endswith(".jpg"):
            img = face_recognition.load_image_file(os.path.join(SAVE_DIR, f))
            enc = face_recognition.face_encodings(img)
            if enc:
                encodings.append(enc[0])
                names.append(f.replace("face_", "").replace(".jpg", ""))
    return encodings, names


# 등록 창
def open_register_window():
    win = tk.Toplevel(root)
    win.title("얼굴 등록")
    win.geometry("500x550")

    # 상단 입력 필드
    tk.Label(win, text="이름").pack()
    name_entry = tk.Entry(win)
    name_entry.pack()

    tk.Label(win, text="전화번호").pack()
    phone_entry = tk.Entry(win)
    phone_entry.pack()
```

Navigate to Related Items

CVEX.PY

II

```python
def open_register_window():
    frame_container = tk.Frame(win)
    frame_container.pack(expand=True, fill="both")

    video_label = tk.Label(frame_container)
    video_label.grid(row=0, column=0, sticky="nsew")

    def save_face():
        name = name_entry.get().strip()
        phone = phone_entry.get().strip()
        if not name or not phone:
            messagebox.showerror("오류", "이름과 전화번호를 입력하세요.")
            return

        frm = picam2.capture_array()
        gray = cv2.cvtColor(frm, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)
        if len(faces) == 0:
            messagebox.showerror("실패", "얼굴을 감지하지 못했습니다.")
            return

        x, y, w, h = faces[0]
        face_img = frm[y:y+h, x:x+w]
        face_img = cv2.resize(face_img, (200, 200))
        filename = f"{SAVE_DIR}/face_{name}_{phone}.jpg"
        cv2.imwrite(filename, face_img)
        messagebox.showinfo("성공", f"{filename}에 얼굴이 저장되었습니다.")

    save_button = tk.Button(frame_container, text="얼굴 저장", command=save_face, bg="blue", fg="white")
    save_button.grid(row=1, column=0, pady=10)

    frame_container.grid_rowconfigure(0, weight=1)
    frame_container.grid_rowconfigure(1, weight=0)
    frame_container.grid_columnconfigure(0, weight=1)

    def update_video():
        frame = picam2.capture_array()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        # 얼굴 주변에 사각형 그리기
        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = ImageTk.PhotoImage(Image.fromarray(rgb))
        video_label.configure(image=img)
        video_label.image = img
        win.after(30, update_video)


    update_video()
# 얼굴인식? 뭐 이런  및 감지로 로그 만들기
def unlock_door(source="얼굴인식"):
    try:
        # 서보 작동
        setServoPos(90)
        time.sleep(1)
        setServoPos(0)
        #이메일 기록을 합니다
        send_email_notification()
        # 로그 기록
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
        log_path = "/home/moomininmoon/다운로드/proj/log.txt"
        with open(log_path, "a", encoding="utf-8") as f:
            f.write(f"{timestamp} 문 열림 ({source})\n")
        print(f"[문 열림] {timestamp} ({source})")

    except Exception as e:
        print(f"[문 열림 실패] {e}")
```

CVEX.PY III

```python
def open_unlock_window():
    global unlock_window

    if unlock_window is not None and unlock_window.winfo_exists():
        unlock_window.lift()  # 이미 창이 열려있다면 최상단으로
        return

    unlock_window = tk.Toplevel(root)
    unlock_window.title("문 열기")
    unlock_window.geometry("500x500")

    frame_container = tk.Frame(unlock_window)
    frame_container.pack(expand=True, fill="both")

    video_label = tk.Label(frame_container)
    video_label.grid(row=0, column=0, sticky="nsew")

    def authenticate():
        try:
            frm = picam2.capture_array()
            rgb = cv2.cvtColor(frm, cv2.COLOR_BGR2RGB)
            encodings, names = load_known_faces()

            face_encs = face_recognition.face_encodings(rgb)
            for enc in face_encs:
                matches = face_recognition.compare_faces(encodings, enc, tolerance=0.5)
                if True in matches:
                    name = names[matches.index(True)]

                    # 로그 저장
                    try:
                        log_path = "/home/moomininmoon/다운로드/proj/log.txt"
                        with open(log_path, "a", encoding="utf-8") as f:
                            timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
                            f.write(f"{timestamp} 문 열림 (인증: {name})\n")
                            f.flush()
                    except Exception as e:
                        print(f"[로그 실패] {e}")

                    def close_after_ok():
                        messagebox.showinfo("인증 성공", f"{name} 님 인증됨\n문 문이 열렸습니다!")
                        setServoPos(90)
                        time.sleep(1)
                        setServoPos(0)
                        send_email_notification()
                        unlock_window.destroy()

                    root.after(100, close_after_ok)
                    return

            messagebox.showwarning("인증 실패", "일치하는 얼굴이 없습니다.")
        except Exception as e:
            print(f"[인증 오류] {e}")

    # 버튼
    button = tk.Button(frame_container, text="문 열기", command=authenticate, bg="green", fg="white")
    button.grid(row=1, column=0, pady=10)

    frame_container.grid_rowconfigure(0, weight=1)
    frame_container.grid_rowconfigure(1, weight=0)
    frame_container.grid_columnconfigure(0, weight=1)

    def update_video():
        frame = picam2.capture_array()
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

CVEX.PY IV

```python
        if unlock_window and unlock_window.winfo_exists():
            unlock_window.after(30, update_video)

    update_video()


def open_registered_list_window():
    win = tk.Toplevel(root)
    win.title("등록된 사용자 목록")
    win.geometry("600x600")

    canvas = tk.Canvas(win)
    scrollbar = tk.Scrollbar(win, orient="vertical", command=canvas.yview)
    scrollable_frame = tk.Frame(canvas)

    scrollable_frame.bind(
        "<Configure>",
        lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
    )

    canvas.create_window((0, 0), window=scrollable_frame, anchor="nw")
    canvas.configure(yscrollcommand=scrollbar.set)

    canvas.pack(side="left", fill="both", expand=True)
    scrollbar.pack(side="right", fill="y")

    # 내부 함수: 수정
    def edit_user(file, old_name, old_phone):
        def apply_changes():
            new_name = name_entry.get().strip()
            new_phone = phone_entry.get().strip()
            if not new_name or not new_phone:
                messagebox.showerror("오류", "이름과 전화번호를 입력하세요.")
                return
            new_filename = f"face_{new_name}_{new_phone}.jpg"
            os.rename(os.path.join(SAVE_DIR, file), os.path.join(SAVE_DIR, new_filename))
            messagebox.showinfo("수정 완료", "사용자 정보가 수정되었습니다.")
            top.destroy()
            win.destroy()
            open_registered_list_window()

        top = tk.Toplevel(win)
        top.title("사용자 정보 수정")
        tk.Label(top, text="이름").pack()
        name_entry = tk.Entry(top)
        name_entry.insert(0, old_name)
        name_entry.pack()

        tk.Label(top, text="전화번호").pack()
        phone_entry = tk.Entry(top)
        phone_entry.insert(0, old_phone)
        phone_entry.pack()

        tk.Button(top, text="변경 적용", command=apply_changes).pack(pady=10)

    # 내부 함수: 삭제
    def delete_user(file):
        confirm = messagebox.askyesno("삭제 확인", f"{file} 사용자를 삭제하시겠습니까?")
        if confirm:
            os.remove(os.path.join(SAVE_DIR, file))
            messagebox.showinfo("삭제 완료", "사용자가 삭제되었습니다.")
            win.destroy()
            open_registered_list_window()

    # 헤더
    headers = ["이름", "전화번호", "사진", "등록일시", "", ""]
    for col, text in enumerate(headers):
        tk.Label(scrollable_frame, text=text, font=("Arial", 12, "bold")).grid(row=0, column=col, padx=10, pady=5)
```

CVEX.P
YV

```python
    for i, file in enumerate(os.listdir(SAVE_DIR)):
        if file.endswith(".jpg") and file.startswith("face_"):
            parts = file.replace("face_", "").replace(".jpg", "").split("_")
            if len(parts) >= 2:
                name, phone = parts[0], "_".join(parts[1:])
                img_path = os.path.join(SAVE_DIR, file)
                dt_str = datetime.datetime.fromtimestamp(os.path.getctime(img_path)).strftime("%Y-%m-%d %H:%M:%S")

                img = Image.open(img_path).resize((100, 100))
                img_thumb = ImageTk.PhotoImage(img)

                tk.Label(scrollable_frame, text=name).grid(row=i+1, column=0, padx=5, pady=5)
                tk.Label(scrollable_frame, text=phone).grid(row=i+1, column=1, padx=5, pady=5)
                tk.Label(scrollable_frame, image=img_thumb).grid(row=i+1, column=2, padx=5, pady=5)
                tk.Label(scrollable_frame, text=dt_str).grid(row=i+1, column=3, padx=5, pady=5)

                tk.Button(scrollable_frame, text="수정", command=lambda f=file, n=name, p=phone: edit_user(f, n, p)).grid(row=i+1, column=4, padx=5)
                tk.Button(scrollable_frame, text="삭제", command=lambda f=file: delete_user(f)).grid(row=i+1, column=5, padx=5)

                scrollable_frame.image = getattr(scrollable_frame, "image", []) + [img_thumb]

#조도 센서 값 작동 우우우

def monitor_light():
    t0 = None
    while True:
        v = measure(1)
        if v >= 2.3:
            if t0 is None:
                t0 = time.time()
            elif time.time() - t0 >= 10:
                print("🔆 밝기 유지됨 → 로그 기록 시도")
                try:
                    log_path = "/home/moomininmoon/다운로드/proj/log.txt"
                    with open(log_path, "a", encoding="utf-8") as f:
                        timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
                        f.write(f"{timestamp} 문 열림 (번호 키 )\n")
                        f.flush()
                    print(f"[✅ 로그 기록 완료] {timestamp}")
                except Exception as e:
                    print(f"[❌ 로그 기록 실패] {e}")
                t0 = None
        else:
            t0 = None
        time.sleep(0.5)


threading.Thread(target=monitor_light, daemon=True).start()


#cctv 작동
is_recording = False
video_writer = None
cctv_filename = ""
def start_cctv_recording():
    global is_recording, video_writer, cctv_filename
    if is_recording:
        messagebox.showinfo("CCTV", "이미 녹화 중입니다.")
        return

    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    cctv_filename = f"cctv_{time.strftime('%Y%m%d_%H%M%S')}.mp4"
    path = os.path.join(CCTV_DIR, cctv_filename)
    video_writer = cv2.VideoWriter(path, fourcc, 20.0, (640, 480))
    is_recording = True
    messagebox.showinfo("CCTV", "녹화 시작됨")
    record_loop()
```

CVEX.PY
VI

```python
def record_loop():
    if is_recording:
        frame = picam2.capture_array()
        video_writer.write(frame)
        root.after(50, record_loop)

def stop_cctv_recording():
    global is_recording, video_writer
    if is_recording:
        is_recording = False
        video_writer.release()
        messagebox.showinfo("CCTV", "녹화 종료됨")
    else:
        messagebox.showwarning("CCTV", "현재 녹화 중이 아닙니다.")
def open_cctv_manage_window():
    win = tk.Toplevel(root)
    win.title("CCTV 파일 관리")
    win.geometry("600x500")

    canvas = tk.Canvas(win)
    scrollbar = tk.Scrollbar(win, orient="vertical", command=canvas.yview)
    scroll_frame = tk.Frame(canvas)

    scroll_frame.bind(
        "<Configure>",
        lambda e: canvas.configure(scrollregion=canvas.bbox("all"))
    )

    canvas.create_window((0, 0), window=scroll_frame, anchor="nw")
    canvas.configure(yscrollcommand=scrollbar.set)

    canvas.pack(side="left", fill="both", expand=True)
    scrollbar.pack(side="right", fill="y")

    headers = ["파일명", "촬영 시간", "", "", ""]
    for col, text in enumerate(headers):
        tk.Label(scroll_frame, text=text, font=("Arial", 12, "bold")).grid(row=0, column=col, padx=10, pady=5)

    def delete_video(file):
        confirm = messagebox.askyesno("삭제 확인", f"{file} 을 삭제하시겠습니까?")
        if confirm:
            os.remove(os.path.join(CCTV_DIR, file))
            win.destroy()
            open_cctv_manage_window()

    def rename_video(file):
        def apply_rename():
            new_name = entry.get().strip()
            if not new_name.endswith(".mp4"):
                new_name += ".mp4"
            os.rename(os.path.join(CCTV_DIR, file), os.path.join(CCTV_DIR, new_name))
            messagebox.showinfo("완료", "이름이 변경되었습니다.")
            top.destroy()
            win.destroy()
            open_cctv_manage_window()

        top = tk.Toplevel(win)
        top.title("이름 변경")
        entry = tk.Entry(top)
        entry.insert(0, file)
        entry.pack()
        tk.Button(top, text="변경", command=apply_rename).pack()

    for i, file in enumerate(os.listdir(CCTV_DIR)):
        if file.endswith(".mp4"):
            full_path = os.path.join(CCTV_DIR, file)
            ctime = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime(os.path.getctime(full_path)))

            tk.Label(scroll_frame, text=file).grid(row=i+1, column=0, padx=5)
            tk.Label(scroll_frame, text=ctime).grid(row=i+1, column=1, padx=5)
            tk.Button(scroll_frame, text="재생", command=lambda f=file: os.system(f"xdg-open '{os.path.join(CCTV_DIR, f)}'")).grid(row=i+1, column=2, padx=5)
            tk.Button(scroll_frame, text="삭제", command=lambda f=file: delete_video(f)).grid(row=i+1, column=3, padx=5)
```

CVEX.PY VII

```python
            tk.Button(scroll_frame, text="삭제", command=lambda f=file: delete_video(f)).grid(row=i+1, column=3, padx=5)
            tk.Button(scroll_frame, text="이름 변경", command=lambda f=file: rename_video(f)).grid(row=i+1, column=4, padx=5)

# 🕒날짜/시간 갱신 함수
def update_datetime():
    now = datetime.datetime.now()
    date_str = now.strftime("%Y-%m-%d")
    time_str = now.strftime("%H:%M:%S")
    weather = get_weather()
    datetime_label.config(text=f"{date_str} {time_str}\n{weather}")
    root.after(60000, update_datetime)  # 1분마다 날씨 갱신

def get_weather(city="Sejong"):
    try:
        def get_latest_base_time():
            now = datetime.datetime.now()
            hour = now.hour
            minute = now.minute

            if minute < 45:
                hour -= 1
                if hour < 0:
                    hour = 23
                    now -= datetime.timedelta(days=1)

            return now.strftime("%Y%m%d"), f"{hour:02d}00"

        base_date, base_time = get_latest_base_time()

        nx = 66  # 세종시 격자
        ny = 103
        SERVICE_KEY = "25j2JTrwGCqvjzBs5amX0JPAucXLuLeLwtOmhwYiAWmbXSm3JUv5Ql87KhpCs08I8jExLdYpPexnUigmZEKmfQ%3D%3D"

        url = (
            f"http://apis.data.go.kr/1360000/VilageFcstInfoService_2.0/getUltraSrtNcst?"
            f"serviceKey={SERVICE_KEY}&numOfRows=10&pageNo=1&dataType=JSON"
            f"&base_date={base_date}&base_time={base_time}&nx={nx}&ny={ny}"
        )

        response = requests.get(url)
        data = response.json()

        if 'response' not in data or 'body' not in data['response']:
            print("❌ 기상청 API 오류: 'body' 없음")
            print("[DEBUG] 전체 응답:")
            print(json.dumps(data, indent=2, ensure_ascii=False))
            return "날씨 정보 없음"

        items = data['response']['body']['items']['item']
        result = {item['category']: item['obsrValue'] for item in items}

        temp = result.get('T1H', 'N/A')
        reh = result.get('REH', 'N/A')
        pty = result.get('PTY', '0')

        if pty == '0':
            rain_status = "☀️ 비 없음"
        elif pty == '1':
            rain_status = "🌧️ 비 오는 중"
        elif pty == '2':
            rain_status = "🌨️ 비/눈"
        elif pty == '3':
            rain_status = "❄️ 눈"
        elif pty == '4':
            rain_status = "🌦️ 소나기"
        else:
            rain_status = "🌈 알 수 없음"
```

CVEX.PY VIII

```python
    except Exception as e:
        print("기상청 API 오류:", e)
        return "날씨 정보 없음"

#메모장 관련 함수

MEMO_FILE = "memo.txt"

def save_memo():
    with open(MEMO_FILE, "w", encoding="utf-8") as f:
        f.write(memo_text.get("1.0", tk.END))
    messagebox.showinfo("메모장", "메모가 저장되었습니다.")

def load_memo():
    if os.path.exists(MEMO_FILE):
        with open(MEMO_FILE, "r", encoding="utf-8") as f:
            memo_text.delete("1.0", tk.END)
            memo_text.insert(tk.END, f.read())

# 사진첩 함수
photo_files = sorted([f for f in os.listdir(photo_dir) if f.lower().endswith(('.jpg', '.jpeg', '.png'))])
current_index = 0
photo_display_img = None

def show_current_photo():
    global photo_display_img
    if not photo_files:
        return
    path = os.path.join(photo_dir, photo_files[current_index])
    img = Image.open(path).resize((230, 170))
    photo_display_img = ImageTk.PhotoImage(img)
    photo_canvas.delete("all")
    photo_canvas.create_image(0, 0, anchor="nw", image=photo_display_img)

def prev_photo():
    global current_index
    if photo_files:
        current_index = (current_index - 1) % len(photo_files)
        show_current_photo()

def next_photo():
    global current_index
    if photo_files:
        current_index = (current_index + 1) % len(photo_files)
        show_current_photo()
#로그 확인용 함수 추가
def open_log_view_window():
    win = tk.Toplevel(root)
    win.title("문 열림 로그")
    win.geometry("600x400")

    text_widget = tk.Text(win, wrap="none")
    text_widget.pack(expand=True, fill="both")

    log_path = "/home/moomininmoon/다운로드/proj/log.txt"
    if os.path.exists(log_path):
        with open(log_path, "r", encoding="utf-8") as f:
            logs = f.read()
            text_widget.insert("1.0", logs)
    else:
        text_widget.insert("1.0", "❌ 로그 파일이 존재하지 않습니다.")


# 메인 창 설정
root = tk.Tk()
root.configure(bg="white")
root.title("스마트 도어락 시스템")
root.geometry("900x600")
```

CVEX.PY
IX

```python
memo_frame = tk.Frame(root, bg="white", bd=2, relief="groove")
memo_frame.place(x=20, y=20, width=250, height=330)

tk.Label(memo_frame, text="메모장", bg="white", font=("Arial", 12, "bold")).pack()
memo_text = tk.Text(memo_frame, height=12, width=30)
memo_text.pack()

# 사진첩 프레임 생성 (이 코드는 root 이후에 위치해야 함)
photo_frame = tk.Frame(root, bg="white")
photo_frame.pack(pady=20)

photo_canvas = tk.Canvas(photo_frame, width=230, height=170)
photo_canvas.pack()

nav_frame = tk.Frame(photo_frame, bg="white")
nav_frame.pack(pady=3)

tk.Button(nav_frame, text="◀ 이전", command=prev_photo, width=8).pack(side="left", padx=5)
tk.Button(nav_frame, text="다음 ▶", command=next_photo, width=8).pack(side="right", padx=5)

show_current_photo()

# 저장 버튼 추가
tk.Button(memo_frame, text="메모 저장", command=save_memo).pack(pady=5)

# 앱 시작 시 자동 로딩
load_memo()


# 🕒 날짜/시간 (오른쪽 상단)
info_frame = tk.Frame(root, bg="white")
info_frame.place(x=620, y=20, width=250, height=80)  # 조금 키워서 날씨 보이게
datetime_label = tk.Label(info_frame, text="", font=("Arial", 12), bg="white", justify="right")
datetime_label.pack(anchor="ne")
update_datetime()

# 📅 달력 (오른쪽 하단)
calendar_frame = tk.Frame(root, bg="white")
calendar_frame.place(x=620, y=100, width=250, height=250)
cal = Calendar(calendar_frame, selectmode='day',
               year=datetime.datetime.now().year,
               month=datetime.datetime.now().month,
               day=datetime.datetime.now().day)
cal.pack()

# 중앙 하단 타이틀
tk.Label(root, text="오늘도 화이팅...!",
         font=("Arial", 20, "bold"), bg="white", fg="black").place(relx=0.5, rely=0.92, anchor="center")

# 메뉴 설정 (유지)
menubar = tk.Menu(root)
action_menu = tk.Menu(menubar, tearoff=0)
action_menu.add_command(label="얼굴 등록", command=open_register_window)
action_menu.add_command(label="문 열기", command=open_unlock_window)
action_menu.add_command(label="등록된 사용자 보기", command=open_registered_list_window)
action_menu.add_command(label="문열림 로그 확인", command=open_log_view_window)

cctv_menu = tk.Menu(menubar, tearoff=0)
cctv_menu.add_command(label="작동", command=start_cctv_recording)
cctv_menu.add_command(label="정지", command=stop_cctv_recording)
cctv_menu.add_command(label="관리", command=open_cctv_manage_window)

menubar.add_cascade(label="메뉴", menu=action_menu)
menubar.add_cascade(label="CCTV 작동 및 관리", menu=cctv_menu)
root.config(menu=menubar)
root.mainloop()
```

moomininmoon@raspberrypi: ~/다운로드/proj

파일(F) 편집(E) 탭(T) 도움말(H)

```
moomininmoon@raspberrypi:~/다운로드/proj $ sh sys.sh
```

삭제 이름 변경
삭제 이름 변경
삭제 이름 변경

4:44

🏠 스마트 도어락 대시보드

🔐 도어락 제어

🔓 문 열기

👥 저장된 얼굴 목록

이름: moon

전화번호: 12345678

📹 CCTV 영상 목록

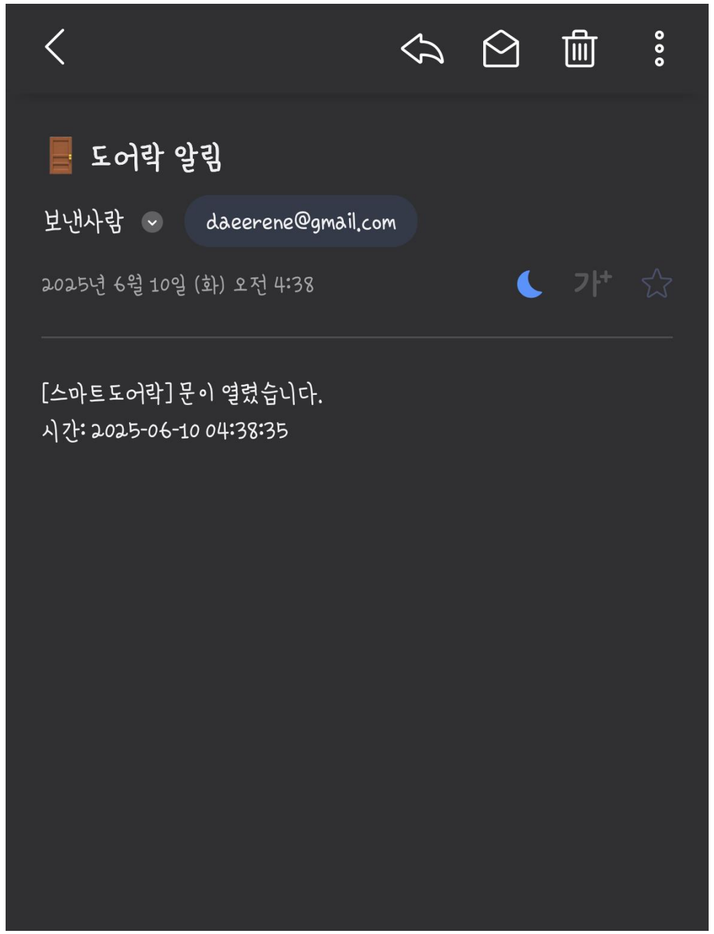
날짜: 2025년 06월 10일 04시 39분 19초

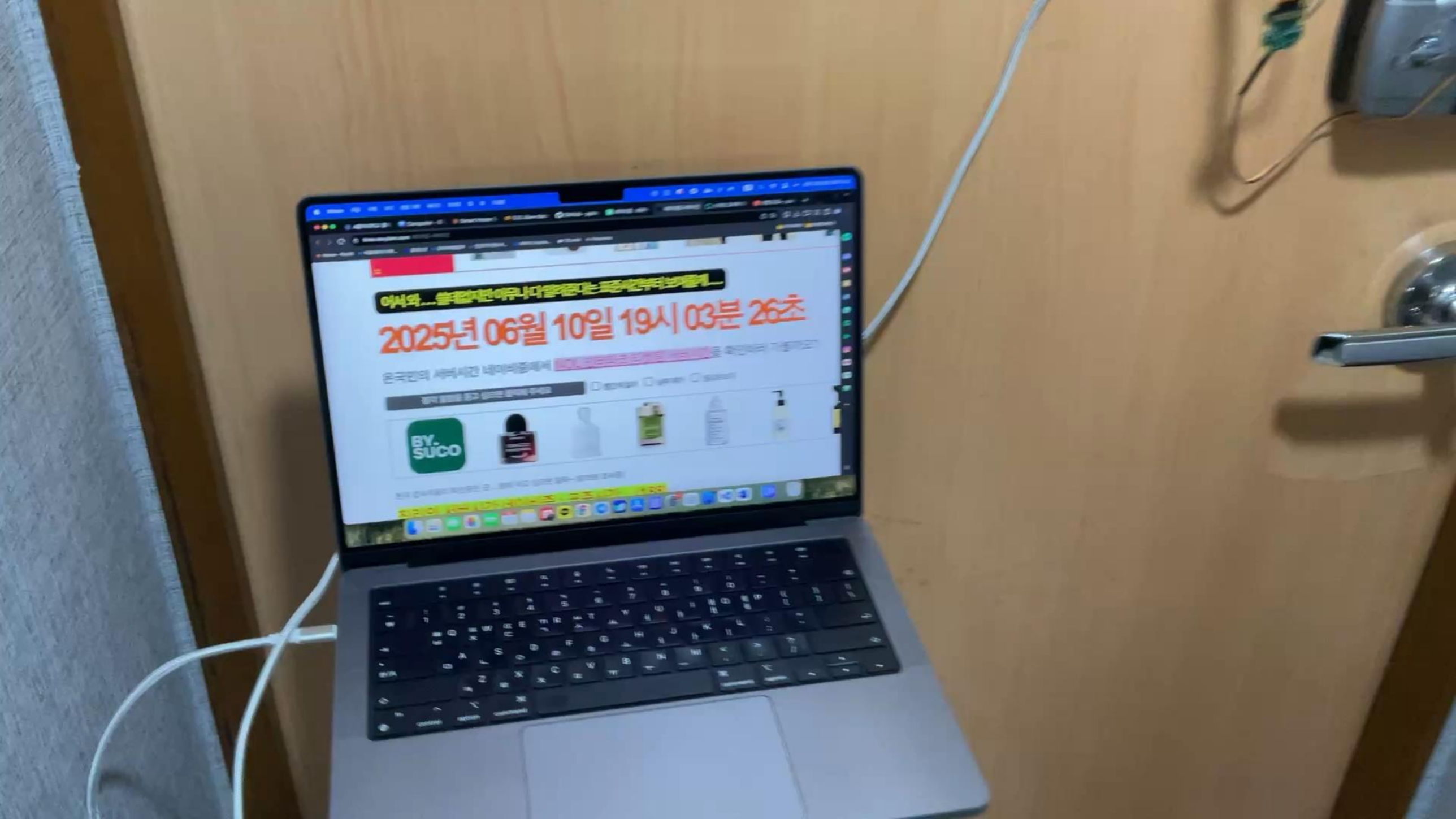
날짜: 2025년 06월 10일 04시 31분 39초

날짜: 2025년 05월 31일 22시 39분 54초

📜 문 열림 로그

192.168.0.3


도어락 알림
보낸사람
daeerene@gmail.com
2025년 6월 10일 (화) 오전 4:38
[스마트도어락] 문이 열렸습니다.
시간: 2025-06-10 04:38:35


2025년 06월 10일 19시 03분 26초
BY.
SUCO

CONCLUSION

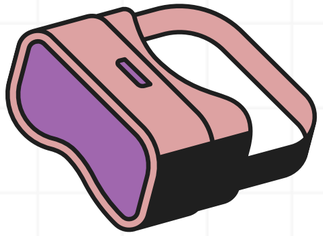
HARDWARE COMPATIBILITY

PERFORMANCE LIMITATIONS

SOFTWARE INTEGRATION

B889031
방문혁

HOME SERVICE ABOUT US CONTACT US



THANK YOU