# Vision-Aided Smart Security System


## Abstract

The AI-Based Smart Door Lock System is an innovative integration of artificial intelligence, computer vision, embedded systems, and humancomputer interaction, designed to enhance home and office security through automation and user recognition. This system replaces traditional keys and password-based security methods with biometric facial authentication, controlled via a flexible Python-based GUI and voice interaction modules. Using OpenCV for face detection and recognition, the system identifies individuals in real-time. If the detected face is recognized, the system automatically unlocks the door using a servo motor controlled by an Arduino microcontroller. If the face is unknown, it prompts the administrator through a voice interface for permission to unlock or remember the person.

The administrator can approve access or initiate the learning process by naming the individual either through GUI or voice command. The captured grayscale image is stored in a structured dataset folder and transformed into a matrix representation that is saved in a YAML file for efficient recognition. All control parameters such as servo lock/unlock angles, lock delay, and communication ports are managed through a GUI, and userdefined settings are saved in a JSON file and transferred to the Arduino at runtime. These settings are then stored in the Arduino's EEPROM to preserve state across reboots, ensuring that the door resumes the previous behavior even after power loss.

The system allows full configuration of the lock angles to adapt to various physical lock mechanisms and includes options to manually lock/unlock the door from the GUI. This system is built to function offline, eliminating reliance on cloud or internet connectivity. It delivers a customizable, secure, and interactive experience, combining automation with machine learning, and provides a strong foundation for future expansion through IoT, mobile control, and cloud-based data management.

## Introduction

With the increasing demand for smart home technologies and personalized automation, traditional lock-and-key mechanisms have become outdated due to their vulnerability and inconvenience. Many conventional systems are susceptible to key duplication, unauthorized access, or accidental loss. In contrast, modern security systems are shifting toward biometric authentication and intelligent decision-making to improve safety and user experience. Our project, the AI-Based Smart Door Lock System, is a significant step in this direction, combining facial recognition, voice-assisted interaction, and hardware control using microcontrollers.

This project aims to deliver a practical solution that leverages AI-based facial recognition technology to control door access without the need for physical keys or passwords. A user- friendly graphical interface allows users to configure settings such as servo angles and delay times, while also providing options to train the system with new user faces. It not only offers security but also adds a layer of personalization by allowing voice-based user interaction.
Built using open-source technologies and compatible with standard hardware, this system provides an effective, scalable solution for both home and institutional use cases.

## Objectives

The main objective of the AI-Based Smart Door Lock System is to develop a secure, intelligent, and fully customizable locking system that automates door access based on facial recognition. The system is designed to identify users by comparing their facial features with a pre-trained dataset. When a person is recognized, the door is unlocked via a servo motor mechanism. If the person is unknown, the system interacts with the admin through voice to confirm whether the door should be opened or not. If the admin says "remember," the system proceeds to capture the new user's face, request their name through voice input, and train the model to include the new identity.

In addition to biometric authentication, the system provides a GUI that allows the user to set locking and unlocking servo angles based on the type of lock hardware used. This flexibility ensures that the system can adapt to various mechanical configurations. The delay before re-locking the door after it is opened is also configurable. All these parameters are stored in a configuration file and sent to the

Arduino board, where they are saved to EEPROM, making them persistent across system restarts.

Another key objective is to ensure the system works efficiently without requiring internet connectivity. The entire facial recognition and voice control process operates locally, and the use of YAML for storing training data ensures fast lookups and minimal resource usage. This makes the system suitable for locations where cloud access is not reliable or feasible. Overall, the system aims to blend AI, embedded systems, and real-world usability into a comprehensive, intelligent locking solution.

The objectives of the system are to deliver the following features:

1. **Secure Access via Facial Recognition**: Automatically unlock the door if a recognized face is detected.
2. **Voice-Driven Access for Unknown Users**: Engage the admin through voice prompts when an unrecognized face appears.
3. **Face Registration via Voice**: Allow the admin to name and register new users via voice input.
4. **Configurable Lock Mechanism**: Enable users to adjust servo motor angles and delay timings via GUI.
5. **Persistent Settings Storage**: Save configurations in EEPROM on Arduino to retain functionality after power cycles.
6. **Offline Operation**: Ensure that all processes run locally without internet dependency.
7. **User-Friendly GUI**: Simplify interaction and configuration for nontechnical users.

## System Architecture

The architecture of the AI-Based Smart Door Lock System is modular, combining software intelligence with hardware control. The system primarily consists of four interconnected layers: the **GUI layer**, the **AI and Voice processing layer**, the **Hardware communication layer**, and the **Memory persistence layer**. At the core, a Python-based GUI acts as the control center, where users configure settings such as lock/unlock angles, camera index, COM port, and delay timing. When the system is initiated, it launches a real-time face detection process using OpenCV. If a face is detected and matched with the saved dataset, it proceeds to unlock the door. The GUI also handles the interaction for new user registration and manual override controls.
Simultaneously, the AI module handles face recognition and voice-based interaction. When an unknown person is detected, the system speaks to the admin and listens for a verbal response. If approved or remembered, the system captures the face, prompts for the name, and retrains the dataset. These processes happen in parallel threads to ensure responsiveness.

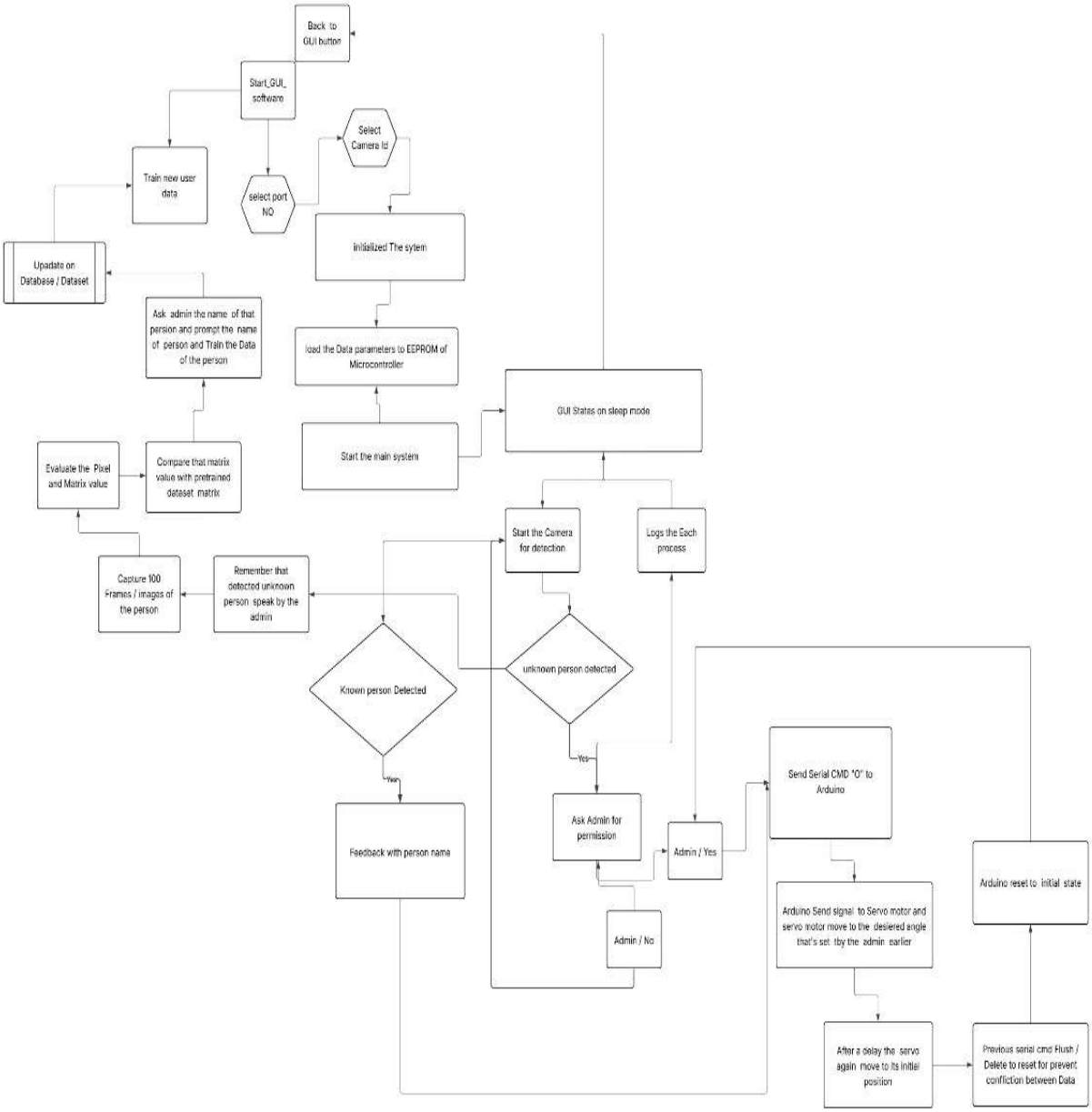
Once an unlock decision is made, a command is sent to the Arduino microcontroller through a serial port. The Arduino, equipped with servo motors, executes the unlock or lock operation based on the received angles. Furthermore, it stores these angles and delay values in EEPROM, allowing the hardware to remember settings across power failures. This layered communication ensures a smooth, fault-tolerant operation, connecting AI with physical actuation.

The following are the layers of the system architecture:

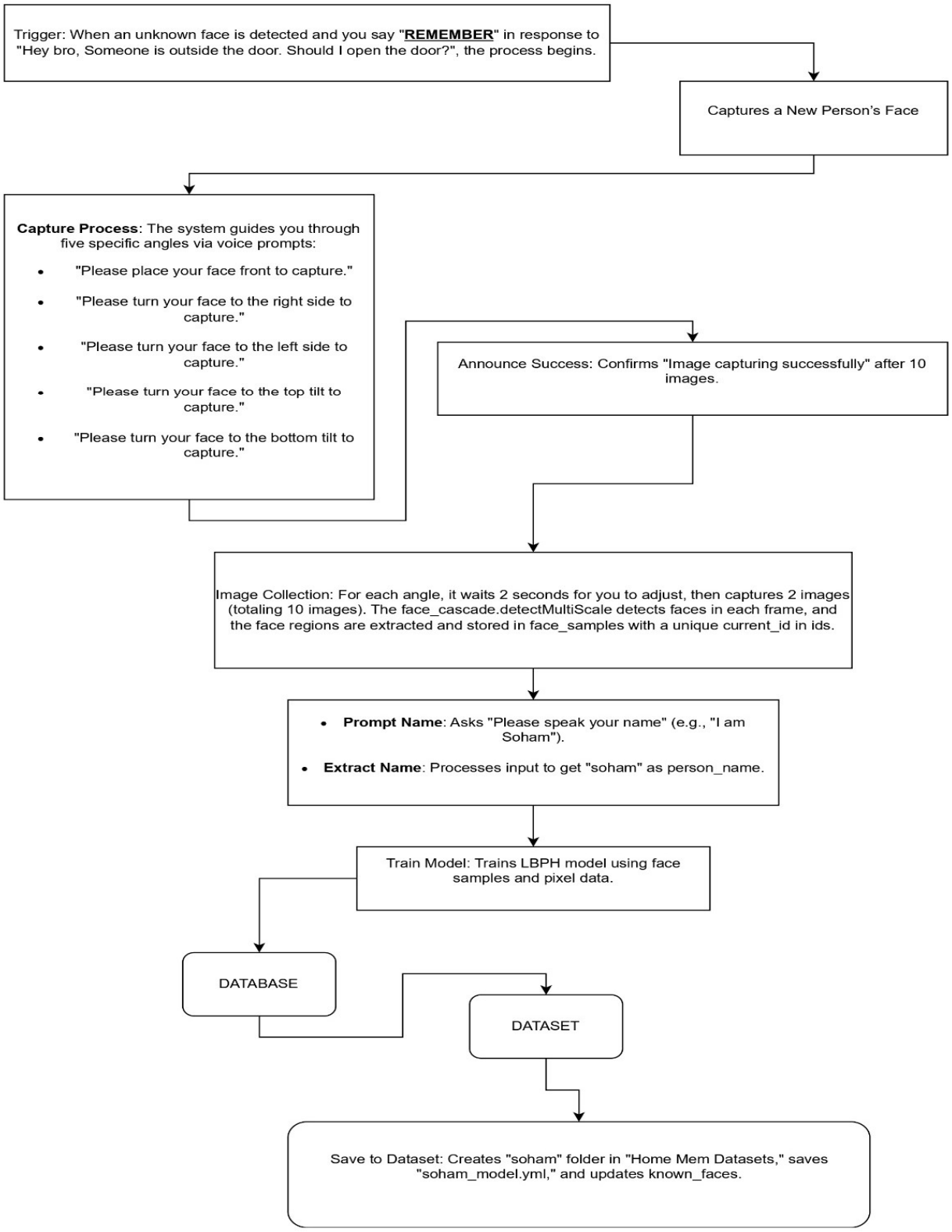
1. **GUI Layer** ○ Developed using **CustomTkinter**. ○ Handles settings configuration (servo angles, COM ports, delay, camera index).
    - Offers manual lock/unlock control and user registration options.
2. **AI & Voice Processing Layer** ○ Performs face detection/recognition using **OpenCV**. ○ Uses **speech_recognition** for voice input and **pyttsx3** for voice output.
    - Supports real-time user identification and registration with parallel threading.
3. **Hardware Communication Layer** ○ Communicates with **Arduino UNO/Nano** via **PySerial**. ○ Controls servo motors for lock/unlock actions using commands like 'O' or angle settings.
    - Transmits configuration data in **JSON** format.
4. **Memory Persistence Layer** ○ Stores facial data in a local dataset and matrix format using **YAML**.
    - Retains hardware configuration in Arduino EEPROM for post-reboot state recovery.

---

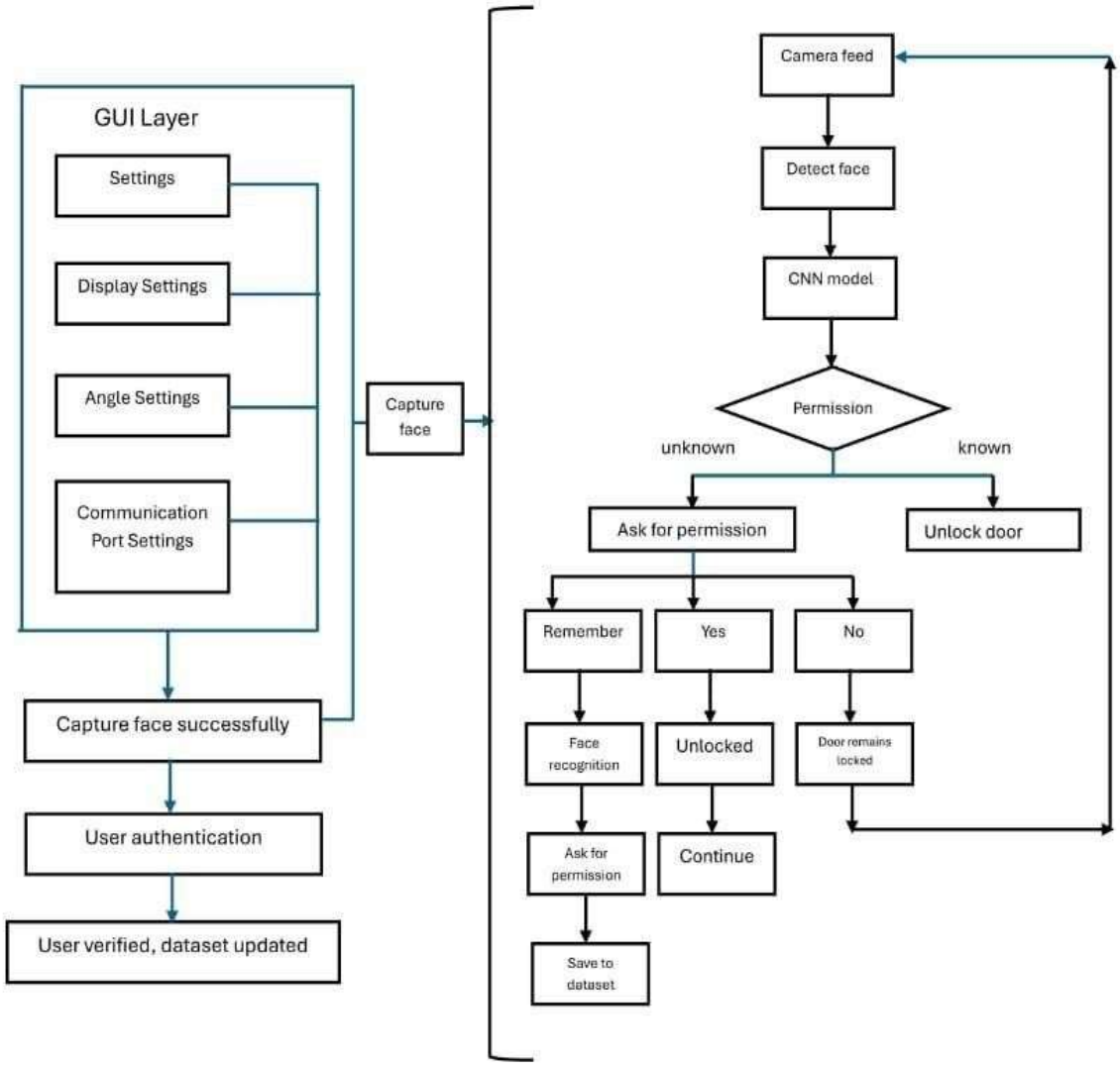
# Complete System workflow

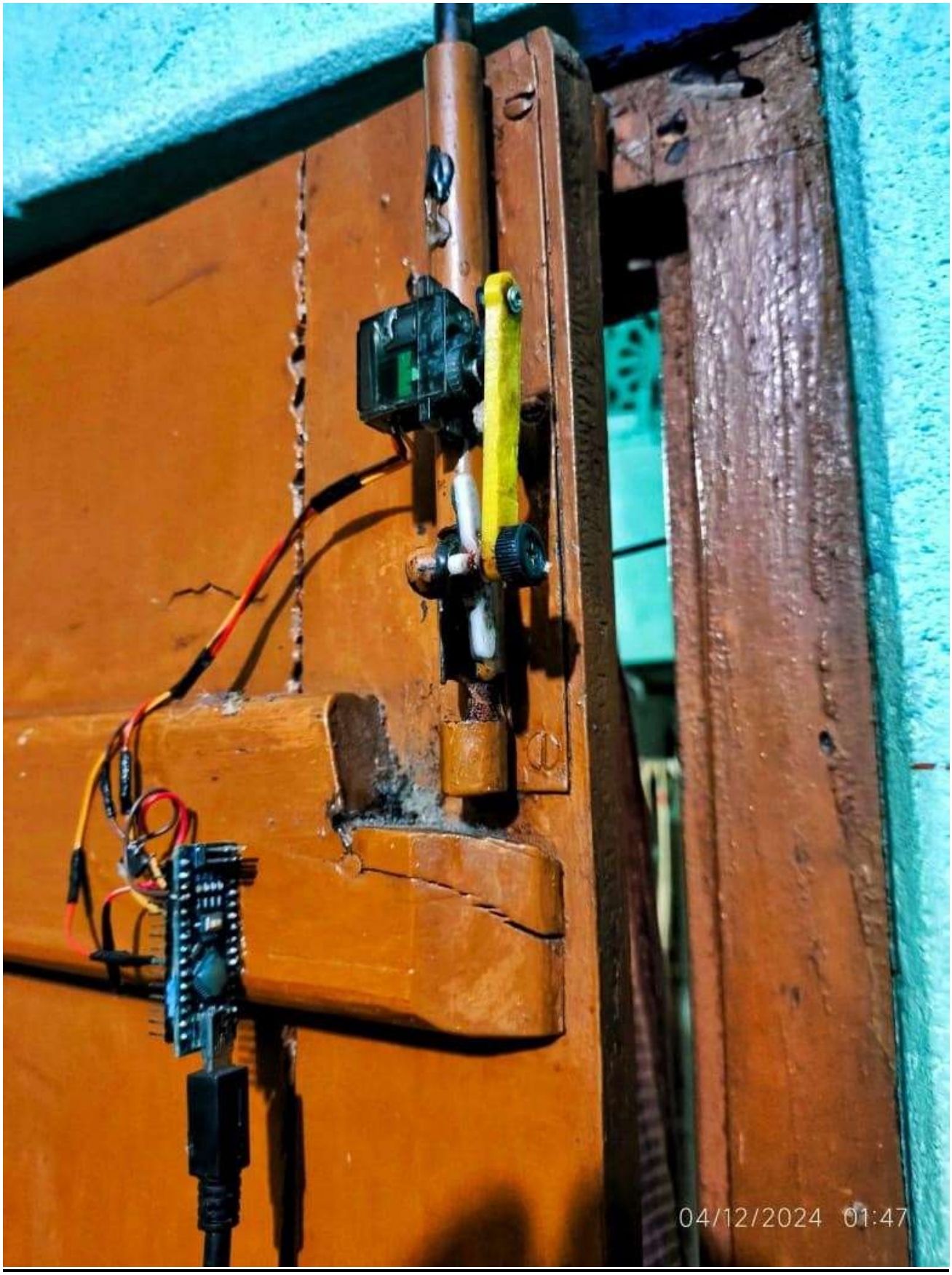


# Data Traning and Gathering  workflow




S


GUI Layer
Settings
Display Settings
Angle Settings
Communication Port Settings
Capture face successfully
User authentication
User verified, dataset updated
Capture face
Camera feed
Detect face
CNN model
Permission
unknown
known
Ask for permission
Unlock door
Remember
Yes
No
Face recognition
Unlocked
Door remains locked
Ask for permission
Continue
Save to dataset


04/12/2024 01:47

**1. Hardware Setup**

The smart door lock system includes the following key components: **a. Arduino Microcontroller:**

- Acts as the central controller of the system.
- Processes serial commands received from the PC via USB.
- Controls the servo motors for lock/unlock actions.
- Compact board with visible USB connector and pin headers.

**b. Servo Motors:**

- Two servo motors are mounted on the door frame.
- Each motor is connected to a mechanical latch system.
- Responsible for rotating the latch to secure (lock) or release (unlock) the door.
- Yellow servo arms visibly indicate connection to the latch.

**c. Wiring and Connectors:**

- Colored wires (red for power, black for ground, yellow for signal) connect Arduino to the servos and power source.
- USB cable used for both power and data communication with the PC.

**d. Mechanical Latch:**

- Mounted on the door edge, connected to the servo arms.
- Engages with the door frame when servos rotate to lock position (e.g., 180°).
- Disengages when servos rotate to unlock position (e.g., 45°).

**e. Door Structure:**

- Setup installed on a wooden door with visible hinges and frame.
- The rough, worn wooden surface demonstrates compatibility with real-world, existing doors.

**f. Compact Arrangement:**

- Components are fixed securely using screws or adhesive mounts.
- Ensures stable operation and minimal movement during locking/unlocking.

**2. Operational Mechanism**

The system operates through coordinated hardware and software interaction:

**a. Locking Process:**

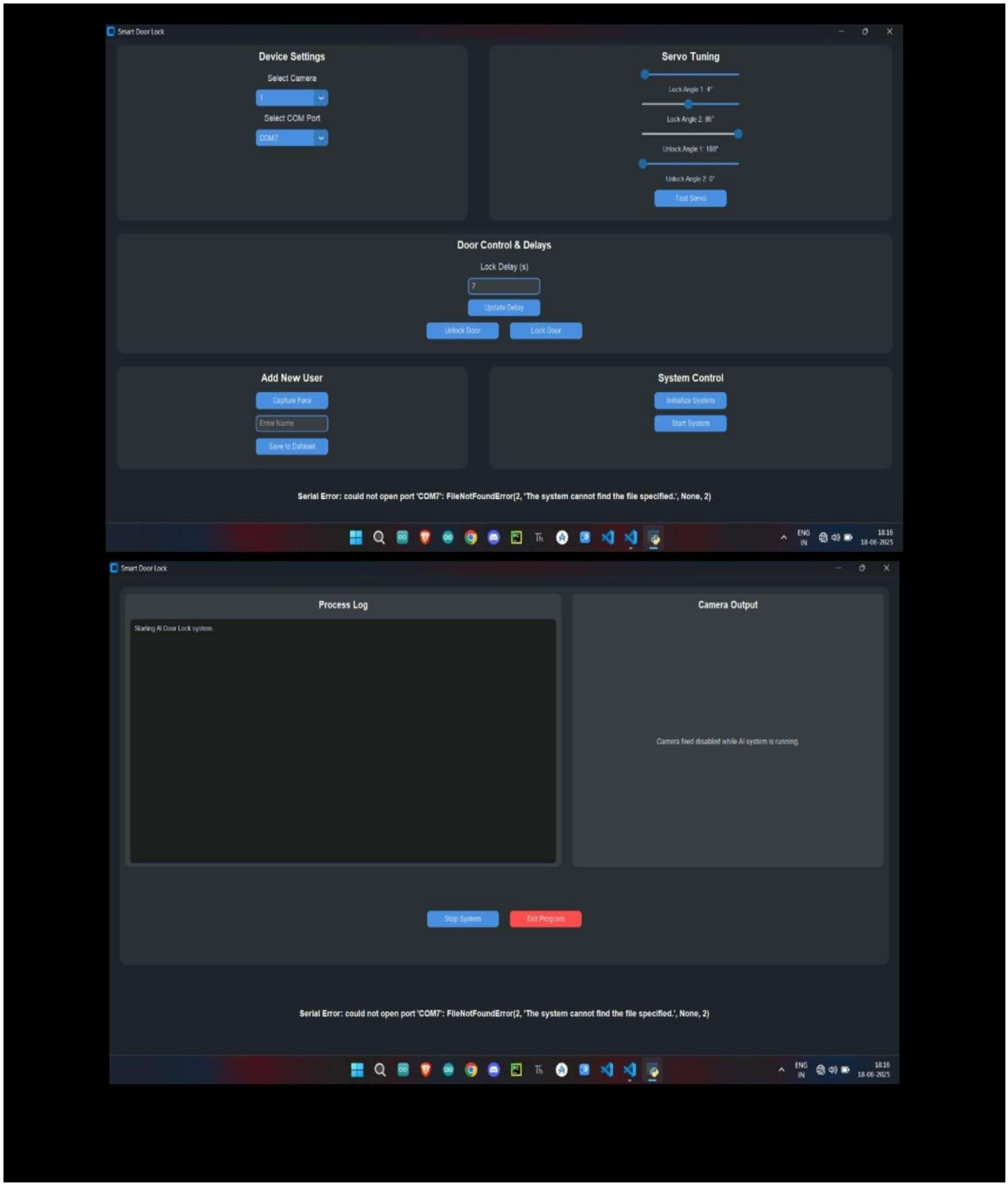
- The PC sends a lock command (e.g., A180,180) via serial communication.
- Arduino interprets the command and rotates both servos to 180°.
- The latch engages, securing the door in the locked position.
- A delay (default 7 seconds, configurable via GUI) allows time for user interaction before automatic locking.

**b. Unlocking Process:**

- The unlock command (e.g., O) is sent from the controlling software.
- Arduino rotates the servos to 45°, moving the latch to the open/unlocked position.
- After the configured delay, the servos return to the locked angle (180°) automatically, re-securing the door.

**c. Manual Override:**

- Users can manually lock or unlock the door through the GUI or web interface.
- Offers added convenience and administrative control without needing physical interaction with the hardware.



## Introduction to the GUI Software

The Graphical User Interface (GUI) for the Smart Door Lock system plays a vital role in bridging the gap between the user and the underlying hardware and AIbased logic. Developed using the CustomTkinter library in Python, the GUI serves as a local control panel, enabling real-time configuration, monitoring, and management of the locking mechanism.

Named "Smart Door Lock", the software ensures a smooth and intuitive interaction experience. It communicates directly with the core script Ai_room_lock.py, as well as the connected Arduino-based hardware via a serial interface. With a clean layout, user-friendly controls, and detailed feedback mechanisms, the GUI is designed for both technical users and everyday individuals who require a secure and manageable access control system.

## 1. Interface Design and Layout

The GUI is crafted with a focus on usability, clarity, and modern aesthetics, employing a dark theme for better contrast and readability in various lighting conditions.

**Key Design Highlights:**

- **Main Window:** ○ Titled "Smart Door Lock". ○ Fixed window size of approximately 1200x800 pixels for optimal component spacing and user interaction.
- **Tabbed Sections:**
  - **The GUI consists of two main sections, accessible via navigation tabs:**
    1. Configuration and Control View – For setup and manual operations.
    2. Feedback View – For system logs and AI camera feed after startup.
- **Color Scheme:**
  - Background: #1f252a (dark gray-black). ○ Frames: #2b3035 (slightly lighter, for clear separation). ○ Text and widgets are styled for contrast and accessibility.
- **Frame-Based Organization:**
  - Each function (like servo tuning, device settings, user addition) is grouped within a rounded frame, improving modularity and clarity.

### 2. Functional Components

The GUI integrates several functional modules, each dedicated to managing a specific part of the system. Below is a detailed breakdown of these components:

**A. Device Settings Frame**

- **Camera Selection:**
  - Dropdown menu for selecting the entry camera index (e.g., "1").
  - Future versions will expand to include multiple cameras:
    - ✦ Entry Camera
    - ✦ QR Scanner Camera
    - ✦ Exit Camera

- **COM Port Selection:** ○ Lists available serial ports (e.g., “COM7”). ○ Connects to the Arduino microcontroller for hardware control.
  - Displays errors such as:

***Serial Error: could not open port 'COM7': FileNotFoundError*** **Suggests that the selected port is either incorrect or unavailable.**

**B. Servo Tuning Frame**

- **Angle Sliders:**
  - Four sliders to adjust the servo motor angles:
    - ✦ Lock Angle 1 and Lock Angle 2 (e.g., 180° for locked position).
    - ✦ Unlock Angle 1 and Unlock Angle 2 (e.g., 45° for open position).
- **Test Servo Button:**
  - Sends the current angle values to the Arduino.
  - Allows users to visually test and fine-tune lock/unlock positions before deploying the system.

**C. Door Control & Delay Configuration Frame** • Lock Delay Entry:

  - Textbox to input the delay time (e.g., "7 seconds") after unlocking before the system auto-locks again.
  - Adjustable using the "Update Delay" button.
- Manual Control Buttons:
  - "Unlock Door" and "Lock Door" buttons for immediate physical control of the door via the GUI.
  - Useful for administrators or manual override situations.

**D. Add New User Frame**

- **Capture Face Button:**
  - Initiates the camera capture process for a new user.
  - Planned update to capture 20 face images to improve training quality.
- **Name Entry Field:** ○ Input for entering the new user's name (e.g., "John").
- **Save to Dataset Button:**
  - Saves the captured face images to the local dataset.
  - Used for training the facial recognition system via train_face.py.

**E. System Control Frame**

- **Initialize System Button:**
  - Prepares all internal variables, ports, and services for operation.
  - Used before launching the main AI lock system.
- **Start System Button:**
  - Executes the core automation script Ai_room_lock.py as a subprocess. ○ Activates facial recognition, voice input, and automated door control.

**F. Feedback Frame (Visible After System Startup)**

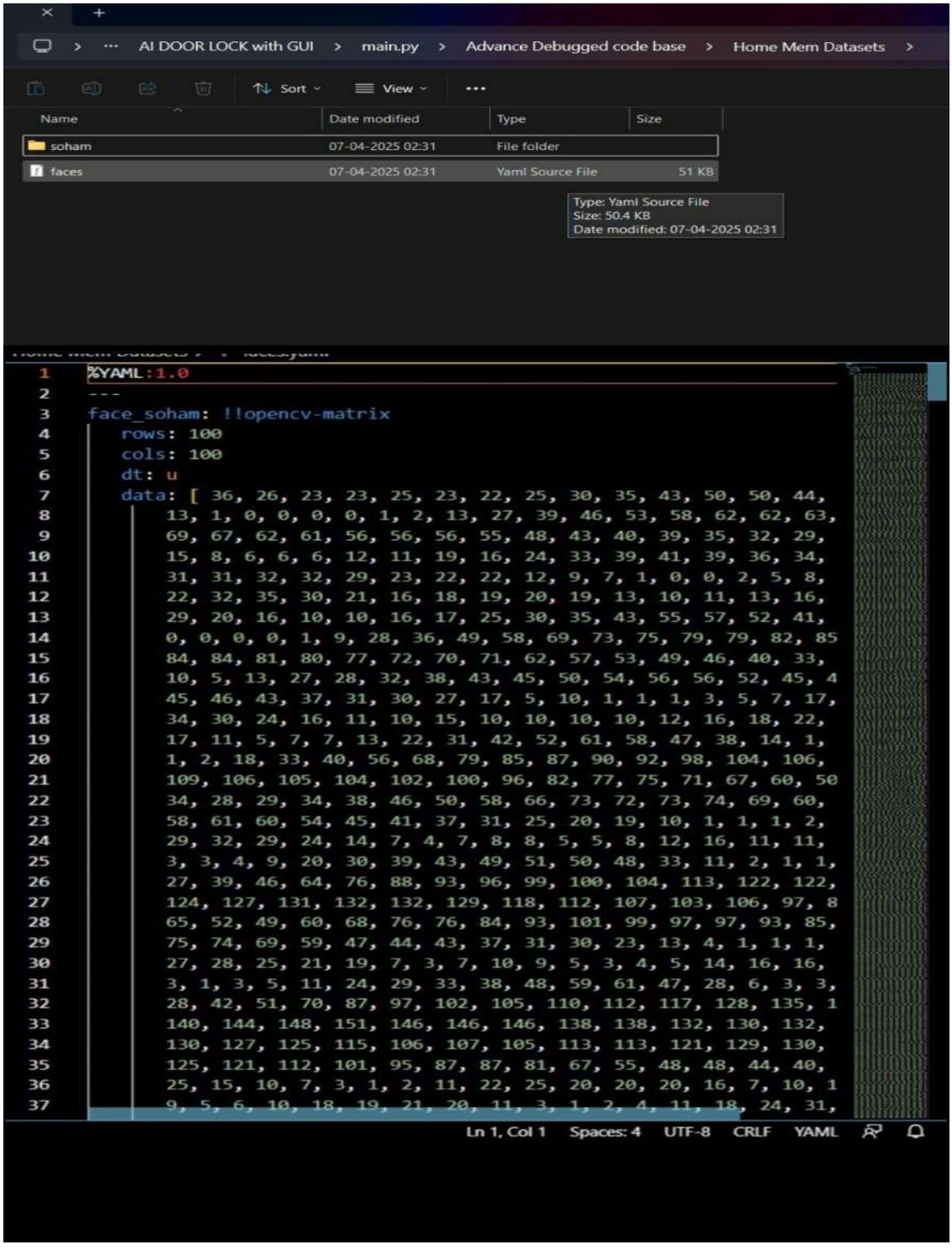
- **Process Log Textbox:**
  - Displays real-time updates from the system:

**e.g., *“Starting AI Door Lock system...”***

- ✦ **Helps in monitoring background processes and debugging.**
- **Camera Output Display:**

- Intended to show the live video feed. ○ Temporarily disabled during AI runtime to avoid conflicts: ***"Camera feed disabled while AI system is running"***

- System State Buttons:
  - "Stop System" – Stops the AI lock subprocess.
  - "Start System Again" – Restarts the subprocess if it was stopped.
  - "Exit Program" – Gracefully exits the application.
    - Highlighted in red to indicate its critical function.



# Introduction to the Face Recognition Dataset

The Smart Door Lock system employs an advanced face recognition mechanism to ensure secure and personalized user access. At the core of this mechanism is the face recognition dataset, which contains digital representations of authorized users' facial features. This dataset is stored in a structured format in a file named faces.yaml, located within the project's dataset directory.
This dataset is critical for the performance and accuracy of the recognition system. It is automatically generated when a new user is added via the GUI, and it's used by the system's machine learning models for identifying and verifying individuals before granting access.

**1. Dataset Structure and Storage**
The dataset is stored in a clearly organized directory structure:
Directory Location:

- Path:
  AI DOORLOCK with GUI > Advance Debugged code base > Home Mem Datasets
- This folder contains:
  - faces.yaml – The main dataset file.
  - User folders (e.g., soham) – Each containing individual face images of the user.

Example File Details:

- File Name: faces.yaml
- File Type: YAML source file
- Size: ~50.4 KB
- Last Modified: July 4, 2025, 02:31 AM User Subdirectories:
- Folders named after each user (e.g., soham) include:
  - Captured or preprocessed face images such as soham_1.jpg, soham_2.jpg, etc.

**2. Data Format in faces.yaml**
The faces.yaml file follows a structured YAML 1.0 format, and is compatible with OpenCV, the computer vision library used in this system.
Structure Overview:

- Root Node: !YAML:1.0
- User Key: face_soham (example for user "soham") Face Data Structure:

Under the key (e.g., face_soham), each user's data is stored as a matrix:
yaml CopyEdit
face_soham:
opencv-matrix:
rows: 100
cols: 100 dt:
u
data: [36, 26, 23, 25, 23, 22, 23, 35, ...] Explanation:

- rows & cols: 100 × 100 pixels (each image is resized to this size)
- dt: u denotes unsigned integer, i.e., pixel values from 0–255
- data: A flattened array of 10,000 grayscale pixel values, representing the face image

This matrix format makes it easy for OpenCV to load, compare, and process face data efficiently during recognition.

**3. Dataset Generation Process**

The dataset is generated through a semi-automated process involving the GUI, OpenCV, and Python scripts (train_face.py).

Step-by-Step Workflow:

**A. Image Capture**

- Triggered by the "Capture Face" button in the GUI.
- Uses the entry camera to:
  - Capture multiple facial images (currently 1, expandable to 20).
  - Detect faces using Haar Cascade Classifier.
  - Crop the Region of Interest (ROI) – the face.

**B. Preprocessing**

- The cropped face images are:
  - Resized to 100x100 pixels.
  - Converted to grayscale.
  - Enhanced using histogram equalization to improve contrast.
- Saved in the user's dedicated folder (e.g., Home Mem Datasets/soham).

**C. Data Aggregation**

- The train_face.py script:
  - Loads the preprocessed images.
  - Flattens each 100x100 image into a 1D array of 10,000 values.
  - Combines and serializes these arrays into faces.yaml for storage.

**D. Training**

- The system initially uses:
  - Template Matching for basic recognition.
- Later upgraded to:
  - DeepFace (a deep learning-based facial recognition library) for more robust and accurate recognition.

**4. Integration with the Face Recognition Algorithm**

During system startup (via Ai_room_lock.py):

- The faces.yaml file is loaded into memory.
- When a user approaches, the system captures their live face image.
- It preprocesses and flattens this image in the same way as the stored data.
- A comparison is made between live and stored images.
- If a match is found, the door is unlocked.

- **This mechanism allows:**
  - Fast and accurate user authentication.
  - Offline operation (no cloud dependency).
  - High security through image preprocessing and structured data matching.

**Conclusion**

The face recognition dataset is a cornerstone of the Smart Door Lock system. It efficiently bridges the gap between real-time face capture and machine learning-based recognition through:

- Structured storage (faces.yaml)
- Efficient preprocessing
- Integration with powerful libraries like OpenCV and DeepFace

By maintaining a clean, scalable, and accurate dataset, the system ensures secure, responsive, and intelligent access control suitable for home, office, or institutional use.

## Technologies Used

The system utilizes a wide range of technologies from various domains of computer science and electronics. **OpenCV**, a powerful computer vision library, is used for face detection and recognition. The system uses Haar cascade classifiers to detect faces and applies template matching for recognizing known individuals. To facilitate interaction, the system incorporates **speech recognition** using the speech_recognition library and **text-to-speech** using pyttsx3, enabling real-time voice input and feedback.

For the graphical interface, the system uses **CustomTkinter**, which provides a modern, stylized version of the classic Tkinter GUI framework. This allows for intuitive control of servo angles, timing, and serial communication settings. On the hardware side, the **Arduino UNO** or **Nano** is used to control servo motors using the **Servo.h** library. The EEPROM library allows persistent storage of user settings on the microcontroller. The communication between Python and Arduino is established via **PySerial**, enabling structured message passing, including JSON-based configuration data.

All facial data is stored locally in structured folders and matrix format using **YAML**, through OpenCV's FileStorage interface. This makes the system lightweight, responsive, and fully offline, eliminating dependencies on cloud services.

Conceptual and Technical Components Used in the Project:

### 1. Python Libraries & Frameworks (Concept & Usage)

- **OpenCV (cv2)**:

Used for capturing video from the camera and detecting faces using Haarcascade classifiers. It also handles image pre-processing like grayscale conversion and histogram equalization for accurate template matching.

- **speech_recognition**:

Captures spoken responses from the admin for decision-making when an unknown face is detected. Converts voice input to text using Google's speech API.

- **pyttsx3**:

Provides text-to-speech functionality to allow the system to verbally communicate with the user or admin, even without an internet connection.

- **CustomTkinter**:

A modern extension of Tkinter used to build the GUI. It supports better theming and interactive widgets for controlling settings like servo angles, COM port, and camera index.

- **pyserial (serial)**:

Used to establish communication between Python and Arduino over the serial port. It transmits configuration data and runtime commands (like unlocking signals).

- **JSON**:

Used to store and transmit configuration data (servo angles, delay times) between Python GUI and Arduino in a structured and readable format.

- **os / time / numpy**:

Used for directory handling (for datasets), managing execution delays (time.sleep), and matrix/image processing.

- **OpenCV FileStorage (YAML format)**:

Saves face training data as image matrices for each user in a structured format, making it easy to retrieve and match during recognition.

. **Microcontroller Core Concepts:**

- **Servo.h Library**:

Controls servo motors connected to Arduino pins. Allows precise rotation to lock/unlock angles.

- **EEPROM.h Library**:

Used to store and retrieve servo positions and other critical settings, ensuring persistent memory across restarts.

- **ArduinoJson.h Library**:

Parses JSON-formatted configuration sent from Python. Allows Arduino to receive multiple parameters in one clean data packet.

- **Serial Communication**:

Arduino uses Serial.read() and Serial.readString() to receive instructions from Python. Configuration data is sent once during setup, and commands like 'O' (Open) are sent during runtime.

- **Conditional Logic in loop()**:

Implements continuous monitoring of serial inputs. Reacts based on the command received, like angle setting or unlock triggers.

- **EEPROM Memory Flow**:
    - During setup(): read servo angles from EEPROM ◦ After unlocking: write updated servo angles to EEPROM

On reboot: Arduino resumes previous state automatically

**Operating System Concepts Applied:**

- **Multithreading (Python)**:

GUI runs in the main thread while camera and voice processes operate in background threads to ensure smooth, non-blocking user interaction.

- **File System Management**:

Dataset images and YAML files are stored, accessed, and updated using the local file system. This involves reading, writing, creating directories, and managing paths dynamically.

- **I/O Device Communication**:

Uses camera (input device) and servo (output device) which interact with the program through drivers and hardware control (serial ports).

- **Inter-process Communication (IPC)**:

While not traditional IPC, the serial communication between Python and Arduino mimics a form of IPC where two systems exchange commands and data in real-time.

- **Persistent Storage**:

EEPROM on Arduino acts like a basic storage unit that remembers state across reboots — a core OS concept similar to saving configuration files.

---

**Structural Design Patterns Followed:**

- **Modular Code Design**:
  - AI_room_lock.py handles core logic ◦ train_face.py handles dataset training ◦ GUI_control.py manages GUI separately
  - Arduino code is neatly segmented into config parsing, command handling, and EEPROM logic
- **Configuration-Driven Execution**:

Uses config.json to allow GUI-modifiable runtime settings without hardcoding values.

- **Data Persistence Strategy**:

Saves user face data both as images and matrices, and hardware settings in EEPROM, preserving both identity and lock behavior.

**Data Handling Techniques:**

- **Grayscale Image Processing**:

All facial data is converted into grayscale and resized to ensure consistency and efficient matching.

- **Template Matching for Face Recognition**:

Instead of deep learning models, efficient OpenCV methods are used to reduce resource consumption.

- **Dynamic Dataset Expansion**:

New users can be added dynamically during runtime via GUI or voice interface without needing to restart or retrain the whole system.

---

## Detailed Working Process

When the system starts, it loads user-defined configuration settings from a JSON file which includes the servo angles, lock delay, COM port, and camera index. The GUI then initializes and waits for the user to either test the servo movement, capture a new face, or start the full AI system.

Upon starting, the camera feed is opened, and face detection is activated. When a face is detected, the system compares it with stored datasets using grayscale matrix matching. If a match is found with confidence above a defined threshold, the name is announced using text- to-speech, and the door is unlocked by sending a command to the Arduino. The Arduino moves the servo motors to the unlock angles and then returns them to the locked state after the specified delay.

If the detected face is unknown, the system prompts the admin using voice interaction: “Someone is outside the door. Should I open the door?” If the admin replies with "yes", it unlocks the door. If "remember" is said, the system captures the unknown face, prompts the admin to speak the name, and then saves the grayscale image and YAML matrix. This updates the system in real time without needing to restart.

All commands to the Arduino, including servo angles and lock duration, are sent via serial port. These parameters are stored inside the EEPROM of the Arduino, ensuring the same configuration is loaded automatically on reboot.

---

**GUI and Hardware Explanation**

The GUI is built using CustomTkinter and serves as the main control dashboard. It has dedicated sections for selecting the camera index and COM port, adjusting servo angles for both lock and unlock positions, and defining lock delay timing. The interface includes sliders for angle control, text entry fields for delay, and buttons to capture face data, initialize the system, and start the AI detection loop.

There is a dedicated frame to show logs and camera feed while the system is running, providing real-time feedback to the user. The GUI also handles all background processes like serial communication, camera availability checks, and configuration saving.

On the hardware side, the Arduino microcontroller is responsible for physically moving the lock using two servo motors. The angle and delay values received from the Python system are parsed and executed. The Arduino sketch also listens for the “O” command, which triggers a predefined unlock-lock sequence using the stored configuration. The servos are powered via pins 6 and 10, and EEPROM is used to ensure the last angles are restored even if the device is reset.

---

**Arduino Communication and EEPROM Functionality**

The Arduino board serves as the actuator component of the system, receiving structured commands from the Python GUI via serial

communication. When the system starts, it sends a JSON string that includes the servo angles and delay values. This configuration is parsed on the Arduino using the ArduinoJson library and applied to the servo motors.

The most critical feature here is the use of **EEPROM memory**, which ensures that user- defined servo angles and behavior are preserved across system restarts or power failures. The Arduino reads from EEPROM during the setup phase and sets the servos to the last saved angles. This memory persistence ensures that even if the system shuts down abruptly, the physical door lock remains consistent in behavior once it reboots.

The Arduino also listens for runtime commands like "A90,45" to manually set angles, and "O" to unlock the door. After executing these actions, it writes the final servo positions back to EEPROM to update the persistent state.

---

**Feasibility**
The proposed system is highly feasible from technical, operational, and economic perspectives**.**

**Technical Feasibility:**
The system is technically sound and practical to implement due to the widespread availability of open-source software libraries and low-cost hardware components. Python, the core programming language used, supports a wide range of libraries such as OpenCV, Tkinter, and PySerial that are not only well-documented but also compatible with most major operating systems including Windows, macOS, and Linux. The integration with Arduino further enhances the technical feasibility, as Arduino boards are inexpensive, reliable, and supported by a large developer community. Moreover, the system has low computational requirements, making it runnable on standard personal computers or even low-power embedded devices.

**Operational Feasibility:**
From an operational standpoint, the system is user-friendly and designed to ensure smooth day-to-day functioning. The graphical user interface (GUI) provides an intuitive experience that requires little to no prior technical expertise. Users can easily interact with the system to perform tasks or monitor outputs. Furthermore, the system is designed to work offline, which ensures uninterrupted functionality regardless of internet connectivity. This offline capability enhances security by reducing the risk of data breaches or external cyber threats. The plug-and-play nature of the setup allows for quick deployment and minimal maintenance.

**Economic Feasibility:**
Economically, the system is very cost-effective. The use of open-source tools eliminates the need for expensive software licenses. Hardware components such as

Arduino boards, sensors, and basic computing units are available at affordable prices in most markets. This makes the solution accessible not only for individual users or students but also for small to medium-sized businesses seeking a budgetfriendly yet efficient system**.**

**Adaptability and Scalability:**
The system is highly adaptable to different environments and use cases. Whether it is for educational purposes, home automation, security, or commercial applications, the architecture of the system allows for easy customization and scaling. Additional modules or features can be integrated with minimal changes to the core structure, ensuring long-term flexibility and usability.

---

**Advantages**
The proposed smart door lock system offers a wide range of advantages over both traditional mechanical locks and existing smart lock solutions. Its design is focused on enhancing security, accessibility, and user convenience, while maintaining low operational costs and minimal technical complexity**.**

**1. Complete Offline Functionality:**
One of the most significant advantages of this system is its ability to operate entirely offline. Unlike many commercial smart locks that rely on cloud services or mobile apps with constant internet connectivity, this system ensures uninterrupted performance even in environments with poor or no internet access. This not only reduces the risk of cyber threats and data leaks but also makes the system suitable for remote or rural locations.

**2. Enhanced Security Through Multi-Modal Authentication:**
The integration of facial recognition and voice control adds layers of security and personalization. Facial recognition provides biometric authentication, reducing the chances of unauthorized access through key duplication or stolen PINs. Voice-based control adds an additional authentication mechanism, ensuring that access is granted only to recognized individuals. The dual-input model significantly strengthens the security framework compared to traditional or single-mode smart locks**.**

**3. User-Friendly and Accessible Interface:**
The system is designed with accessibility in mind. Voice feedback and command functionality make the system especially helpful for elderly users, individuals with disabilities, or those unfamiliar with digital interfaces. It eliminates the need to remember complex passwords or carry physical keys, making daily use more intuitive and efficient.

**4. EEPROM-Based Memory Retention:**
The use of EEPROM (Electrically Erasable Programmable Read-Only Memory) ensures that user data, settings, and access logs are preserved even during power outages or device restarts. This non-volatile memory functionality is critical for maintaining system reliability and avoiding data loss, which is especially important in security-sensitive applications**.**

**5. Modular and Upgradable Design:**
The system is built with a modular architecture, allowing easy upgrades and customizations. For instance, users can integrate additional features such as fingerprint scanning, mobile app connectivity, or RFID access without overhauling the entire system. This adaptability ensures that the system remains relevant and scalable as user needs evolve.

**6. Cost-Effective and Resource-Efficient:**
Compared to commercial smart locks that may require expensive subscriptions or proprietary hardware, this system uses affordable and widely available components like Arduino microcontrollers and open-source libraries. It provides advanced functionality at a fraction of the cost, making it ideal for personal, educational, or small business applications**.**

**7. Environmentally Resilient and Low Maintenance:**
Since the system is not dependent on cloud servers or third-party apps, it is less prone to failures caused by network issues or service outages. This makes it reliable in a variety of environmental conditions. Additionally, the use of robust components ensures low maintenance requirements and long operational life.

---

**Conclusion**
The AI-based smart door locking system presents a modern, intelligent, and highly practical solution to contemporary security challenges. By seamlessly combining advanced technologies such as facial recognition, voice interaction, graphical user interface (GUI) configuration, and embedded hardware control, the system goes beyond conventional locking mechanisms to deliver a sophisticated and responsive security solution.
This project effectively demonstrates the integration of machine learning, computer vision, and embedded systems in a real-world application. Facial recognition ensures biometric-level security, making unauthorized access extremely difficult. Voice interaction not only adds an extra layer of control but also enhances accessibility for users of all ages and abilities. The GUI

provides a simple, interactive platform for system configuration and monitoring, ensuring that the technology remains approachable even for non-technical users.

One of the standout features of this system is its offline capability, which allows it to function independently of internet services. This ensures high reliability and eliminates dependency on cloud-based platforms, reducing vulnerability to remote attacks or network failures. Additionally, the use of EEPROM for memory persistence guarantees that crucial data—such as user profiles, access history, and system settings—remains intact even after power interruptions, thereby ensuring consistent operation and data integrity. The modular and customizable architecture further enhances the system's versatility. Whether deployed in a residential home, office space, school, or other institutional settings, the system can be tailored to meet specific security requirements. Users can expand the system by incorporating additional authentication methods, IoT features, or remote access control as needed. Moreover, this project reflects a strong understanding of cross-disciplinary technology, merging artificial intelligence, software engineering, and hardware interfacing into one cohesive and innovative system. It not only serves as a practical solution but also as a learning model for students, developers, and researchers interested in building secure and intelligent automation systems. In conclusion, the AI-based smart door lock stands as a forward-thinking solution in the domain of smart security. Its blend of innovation, functionality, reliability, and accessibility makes it a compelling option for enhancing safety in modern environments. The successful implementation of this system highlights the potential of intelligent automation in shaping the future of secure, user-centric technologies.

---

## Future Scope

While the current smart door locking system provides a robust and secure solution using facial recognition, voice interaction, and offline capabilities, there are several promising avenues for future development and enhancement. These advancements can significantly improve the system's functionality, accessibility, and scalability, making it suitable not just for personal use, but also for commercial and institutional applications.

---

### 1. 1. IoT Integration for Remote Access

The integration of Internet of Things (IoT) modules such as Wi-Fi (ESP8266/ESP32) or GSM modules can enable real-time remote access and monitoring. Through mobile applications or web dashboards:

- Users can lock/unlock the door remotely from anywhere.
- Receive real-time status updates and alerts.
- Access a central control panel for managing multiple locks across locations.
  This makes the system highly suitable for smart homes, offices, and rental properties where remote management is critical.

---

**2. Advanced Biometric Authentication**

In addition to facial recognition, the system can incorporate other biometric technologies such as:

- Fingerprint scanners for quick, contact-based verification.
- RFID card readers for easy user access and visitor management. These secondary authentication methods provide multi-factor security, reducing the risk of unauthorized access and improving overall reliability.

---

**3. Cloud-Based Data Backup and Analytics**

Storing facial data, user profiles, and access logs on a secure cloud server can:

- Ensure data recovery in case of hardware failure.
- Allow for centralized data synchronization across multiple devices.
- Enable administrators to analyze user behavior, track entry patterns, and generate security reports.
  This is especially beneficial for large buildings, institutions, and businesses with multiple users.

---

**4. Smart Notifications and Alerts**

The addition of a notification system through SMS, email, or push notifications can keep users informed about:

- Door lock/unlock events.
- Failed access attempts or unauthorized activity.
- Low battery or system errors.
  This proactive communication adds a layer of real-time security awareness**.**

---

**5. Intrusion Detection and Alarm System**

To enhance security further, the system can include:

- Motion sensors, door contact sensors, or vibration detectors.
- Trigger alarms or alerts in the event of forced entry attempts or tampering.
  This helps transform the system into a complete smart security solution, not just an access controller.

---

**6. Real-Time User Monitoring and Logs**

Implementing user tracking features can provide:

- Timestamped entry/exit logs for each registered user.
- Insights into user behavior and activity patterns.
- Potential for automated attendance systems in schools or offices.

---

**7. Integration with Home Automation Systems**

Compatibility with smart assistants (like Google Assistant, Alexa, or Siri) can allow:

- Voice-controlled door access.
- Automation routines such as unlocking the door when the user arrives home (geo-fencing).
- Coordination with lights, cameras, or security systems for a fully connected smart home.

---

**8. Enhanced Power Management**

Introducing power backup solutions such as:

- Rechargeable batteries, solar panels, or UPS modules, can ensure uninterrupted operation during power failures, improving the system's reliability in remote or critical setups.

---

**9. Customizable Access Policies**

Developing a role-based access control system could enable:

- Temporary or time-bound access permissions for guests or employees.
- Custom schedules and restrictions for different users.
  This enhances administrative control and is ideal for co-working spaces, institutions, and hospitality services.