ECE445
SENIOR DESIGN LABORATORY

AI Facial Recognition for Automated Room Access

Design Document

Team #1

Chaohua Yao [chaohua4]
Jianchong Chen [jc131]
Haowen Lin [haowenl3]
Zitong Qu [zitongq2]

Advisor: Hua Chen

April 2, 2025

Contents

1. Introduction

1.1 Problem Statement

In campus buildings, temporary access cards are still commonly distributed through manual front-desk procedures. This workflow creates long queues during peak hours, increases repetitive workload for staff, and leads to a unsatisfied user experience. Existing automation solutions are often based on industrial card dispensers and high-performance embedded platforms, which are expensive and difficult to maintain for small laboratories or student projects. This project targets a lightweight and deployable solution for low-cost scenarios. The design goal is to achieve a complete "request–verify–dispense–feedback" loop with affordable open-source hardware, while preserving basic security constraints through face verification and permission checks. Moreover, our project can be easily used through voice control without extra actions including manually inputting personal information and we also embed LLM model [1] in our project which can server as a small AI assistant to chat with you, facilitating students using.

1.2 Solution Overview & Visual Aid

The proposed system is a low-cost AI-assisted temporary card distribution terminal driven by a Python finite state machine. The architecture contains seven core components: mainboard, microphone, large language model interface, camera, speaker, mechanical ejection unit, and screen. Instead of relying on expensive industrial equipment, the system adopts an Raspberry Pi controller, a USB camera and microphone, a servo-based card pushing mechanism, and lightweight software modules (Vosk, OpenCV/face_recognition, Pygame, and cloud LLM API). Operationally, the terminal starts in an idle state, listens for voice input, and uses speech-to-text plus intent analysis to determine whether card retrieval is requested. If required, the system captures a face image for identity matching. On successful verification, the servo mechanism ejects one card; otherwise, dispensing is rejected. Finally, the UI and audio modules provide immediate visual and spoken feedback, and the state machine resets to idle for the next user. Moreover, if the system judges that it does not receive the signal for ejecting the card, it will switch to the chat mode so you can talk with our AI model.

1.3 High-Level Requirements List

- **Interaction Reliability:** The speech interface shall correctly classify card-request intent and non-card conversation under typical indoor noise conditions.
- **Verification and Access Control:** Card dispensing shall be triggered only after successful face verification and valid permission logic.
- **End-to-End Responsiveness:** The complete workflow from user request to feedback shall finish within a practical real-time interaction window for front-desk use.

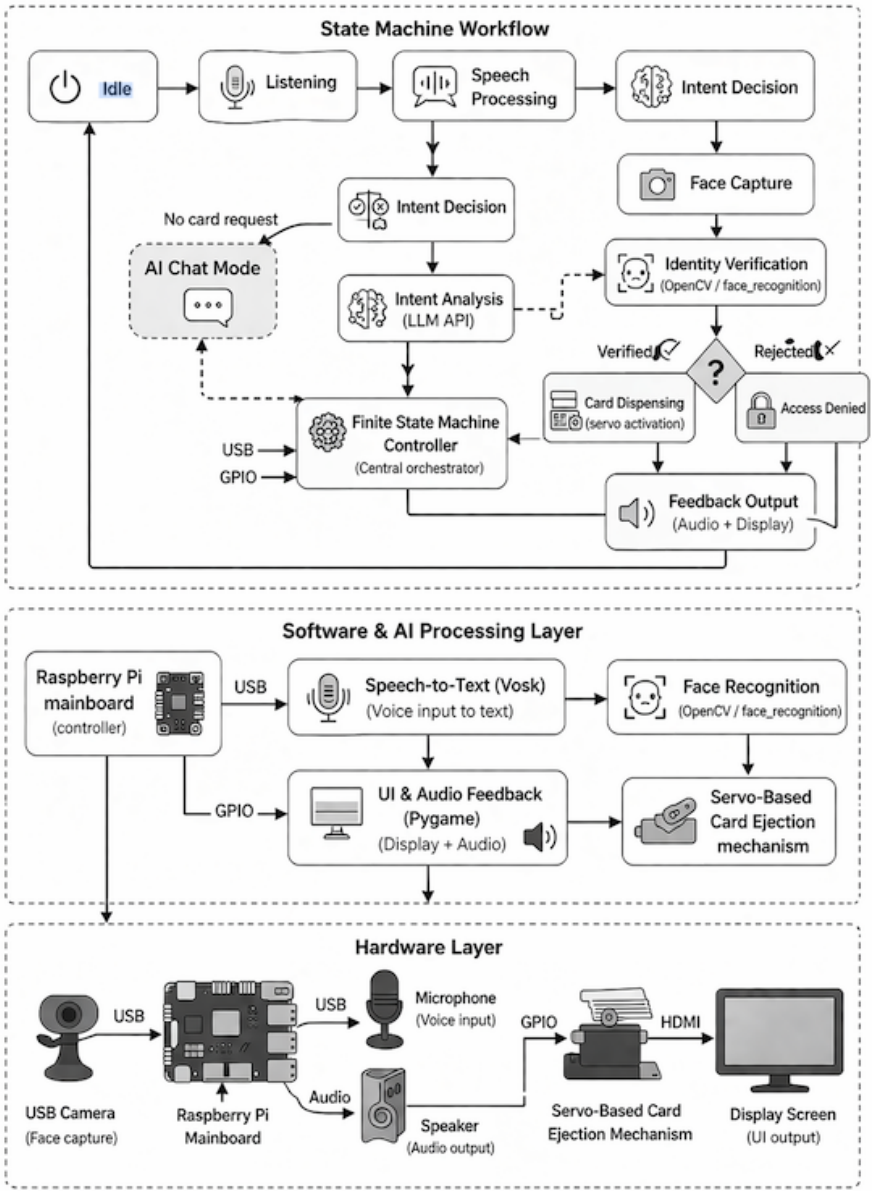

Figure 1: Visual Aid of the whole system

- **Safety and Robustness:** The mechanical subsystem shall dispense one card per authorized cycle, prevent repeated unintended ejection, and return to a safe idle state after each interaction.

2. Design

2.1 Block Diagram

The system consists of three main subsystems: the Human-Computer Interaction System, the Identity Verification System, and the Mechanical Control System.

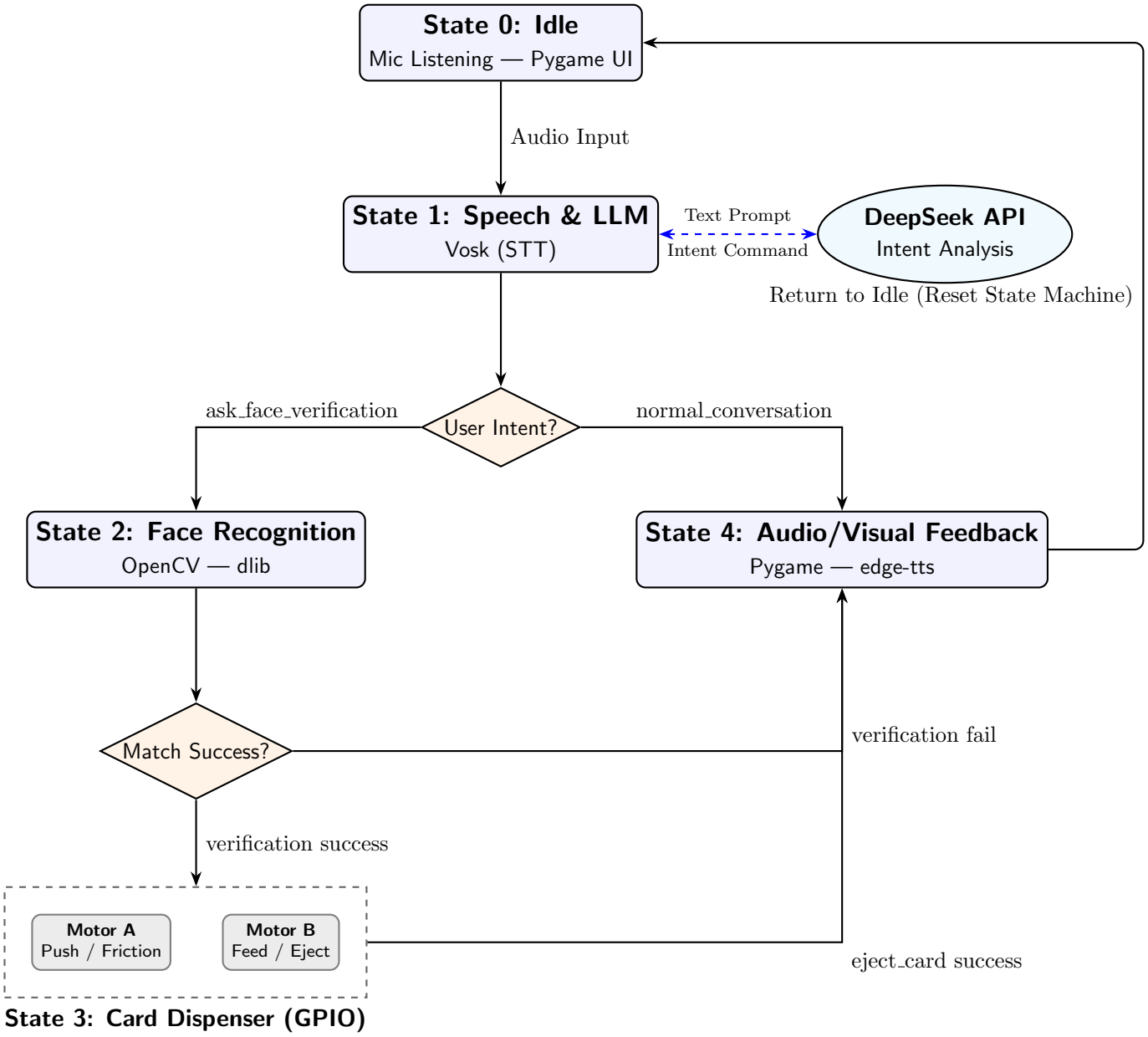

Figure 2: State Machine and Block Diagram for AI Automated Room Access

2.2 Speech Question-Answer System

The Speech Question-Answer System is responsible for user voice interaction in the intelligent front-desk terminal. It receives the user's spoken request, converts the speech signal into text, interprets the request under the current system state, and generates both a spoken reply and a structured response label for downstream modules. This subsystem does not directly authorize card issuance. Instead, it determines whether the request should trigger the next step of the workflow, such as face verification, normal conversation handling, or a repeated-input prompt.

This subsystem is state-aware rather than an unrestricted chatbot. The recognized user utterance is processed together with the current system state, so the language model can only produce outputs that are valid for the current operating condition. In the `idle` state, for example, a valid temporary-card request produces the label `ask_face_verification`, while unrelated or uncertain input produces either a normal conversational response or a repeat-prompt response. The subsystem consists of three functional parts: audio capture and speech recognition, state-aware dialogue management, and response generation.

2.2.1 Audio Capture and Speech Recognition

The audio front end captures short spoken English commands from a user standing in front of the terminal. Since the expected interaction is based on command-level utterances such as "I want to get a card," the subsystem is optimized for short requests rather than long-form dialogue. In this project, speech recognition is implemented using Vosk, which performs local ASR inference and outputs recognized text to the dialogue manager. If the recognition result is incomplete or has low confidence, the subsystem requests the user to speak again instead of forcing an uncertain state transition.

Table 1: Requirements and verification for Audio Capture and Speech Recognition

Requirements	**Verification**
1. The subsystem shall correctly recognize key English command phrases with at least 95% accuracy for utterances lasting 1–5 seconds in an indoor environment.	1. Prepare a test set of 100 utterances from at least 5 speakers, including valid commands, unrelated conversation, and incomplete requests. A trial is counted as correct if the Vosk transcription preserves the keywords required for downstream intent classification. The measured command-recognition accuracy shall be at least 95%.
2. The end-to-end latency from end-of-speech detection to ASR text output shall be no more than 1.0 second in at least 95% of trials.	2. Record timestamps for end-of-speech detection and Vosk output generation for 100 trials. At least 95 of the 100 trials shall complete within 1.0 second.

2.2.2 State-Aware Dialogue Management

After Vosk transcription is completed, the dialogue manager packages the recognized text together with the current system state and sends them to the large language model. The model is not used as a general-purpose chatbot. Instead, it acts as a constrained classifier and response generator. Its outputs are restricted to a predefined set of valid response labels, such as `normal_conversation`, `request_speak_again`, and `ask_face_verification`. This design prevents illegal state transitions and reduces the probability of unsafe or inconsistent behavior. The dialogue manager therefore serves as the core logic block that maps user speech and system context into deterministic control outputs.

Table 2: Requirements and verification for State-Aware Dialogue Management

Requirements	**Verification**
1. The subsystem shall output only labels from a predefined valid set and shall produce zero illegal state transitions for all tested state-input combinations.	1. Build a scripted test suite covering all supported states and representative utterances. Compare each generated label against the allowed output set for that state. Zero illegal labels or illegal transitions are permitted in 200 test cases.
2. For the three predefined intent classes `normal_conversation`, `request_speak_again`, and `ask_face_verification`, the subsystem shall achieve at least 95% classification accuracy on a labeled test set.	2. Create a labeled evaluation set of 100 state-utterance pairs. Run the full subsystem and compare the generated label against ground truth. The classification accuracy shall be at least 95%.

2.2.3 Response Generation and Output Formatting

The output stage generates two synchronized results. The first is a short natural-language reply for speaker playback, which informs the user of the next required action. The second is a structured response label for the control subsystem. For example, when the user successfully requests a temporary card in the `idle` state, the subsystem may generate the spoken reply "Please look at the camera for verification" together with the label `ask_face_verification`. By separating the spoken reply from the structured label, the subsystem allows future revisions of the dialogue wording without modifying the low-level interface with other subsystems.

To support integration with the control subsystem, the structured output of the Speech Question-Answer System follows a fixed interface format:

$$O = \{S_{curr},\ U,\ L,\ R\} \tag{1}$$

where S_{curr} is the current state, U is the recognized user utterance, L is the generated response label, and R is the spoken reply text. In implementation, the control subsystem

uses L as the machine-readable signal for state progression, while the audio subsystem uses R for speaker playback.

Table 3: Requirements and verification for Response Generation and Output Formatting

Requirements	**Verification**
1. When Vosk confidence is below threshold, or when the recognized utterance is ambiguous, the subsystem shall return the label `request_speak_again` within 1.0 second and shall keep the workflow in the current safe state.	1. Inject 30 noisy, truncated, or intentionally ambiguous utterances into the subsystem. In every case, the subsystem shall generate `request_speak_again` within 1.0 second and shall not trigger the next control action.
2. The subsystem shall generate both a valid spoken reply and a valid control label for every successful inference.	2. Run 100 consecutive trials and inspect the returned output object. Every output shall contain both a non-empty reply string and a valid response label. Missing fields are not allowed.

This subsystem improves safety and modularity by ensuring that the large language model does not directly issue physical actuation commands such as card ejection. Instead, it only generates validated conversational outputs and structured response labels for the next stage of the workflow. As a result, the speech interface remains flexible for natural interaction while the overall system maintains deterministic control over critical actions.

2.3 Identity Verification System

This system is designed to ensure that only authorized users can access the card-dispensing function.
It includes face enrollment, face recognition, and permission management. In the enrollment stage, an administrator records a user's facial information and stores the corresponding identity data in a local database. During operation, when a user requests to take a card, the camera captures the user's face and the system compares it with the enrolled face database. If the facial features match a registered user and the user has permission to access the card, the verification is considered successful. Otherwise, the request is rejected. In addition to recognition, this subsystem also manages user permissions, such as whether a user is allowed to receive a card, whether they have already collected one, and whether their information should be updated or removed. This system is important because it adds security and prevents unauthorized card collection.

2.3.1 Facial Recognition Subsystem Description

Algorithm and Data Pipeline This module processes the raw video stream captured by the camera. First, a face detection algorithm isolates the facial region from the background

to optimize computational efficiency. Next, the system extracts facial features and converts them into high-dimensional feature vectors (embeddings). To authenticate the user, the system calculates the cosine similarity between the captured feature vector (v_c) and the registered vectors (v_d) stored in the database:

$$\text{Similarity} = \frac{v_c \cdot v_d}{\|v_c\|\|v_d\|}$$

If the similarity score exceeds a predefined threshold, the authentication is deemed successful, and the system retrieves the corresponding unique User ID.

Software Interface and State Management Unlike a standalone mobile application, the software interface for the facial recognition subsystem is integrated into a global state machine UI driven by Pygame. The subsystem operates specifically within "State 2" of the system architecture. Initially, the system remains in a microphone-listening idle mode (State 0). Once State 2 is invoked, the interface utilizes OpenCV and the dlib library to capture real-time camera frames and perform facial feature extraction. To maintain optimal system responsiveness, the computationally intensive vision tasks are designed to be non-blocking relative to the Pygame event loop, ensuring the user interface does not freeze during the matching process.
Within State 2 (Facial Recognition), the system establishes a connection with the underlying camera hardware via OpenCV's `VideoCapture` API. To optimize resource consumption and ensure user privacy, the camera remains physically inactive during idle states. Hardware wake-up and video stream initialization are triggered exclusively when the application layer receives the `ask_face_verification` intent generated by the LLM module.
Once initialized, the camera continuously pushes image data into memory at a predefined frame rate. Rather than executing a single discrete "photo capture," the system relies on dlib's face detection algorithm to continuously scan the incoming buffer frames. A critical frame is automatically extracted for feature matching only when a stable facial bounding box, meeting predefined quality thresholds, is detected. Regarding data rendering and UI feedback, the raw BGR image matrices (NumPy arrays) fetched by OpenCV are dynamically converted into RGB color space within memory. These arrays are subsequently transformed into `Pygame Surface` objects.

Interaction with Other Subsystems This subsystem interacts with the LLM subsystem using a decoupled communication structure based on a standardized JSON protocol. Upon successful authentication, the application transmits a trigger packet containing the `User_ID` and `Auth_Status` via network protocols (e.g., HTTP or WebSocket) to the LLM module. Receiving this signal "wakes up" the LLM, allowing it to retrieve user-specific data and initiate the voice interaction. During the interaction and dispensing phases, the facial recognition App enters a "locked/waiting" state. It will only reset to the "Idle" state after the Mechanical subsystem completes the card ejection and broadcasts a "task finished" signal.

2.3.2 Requirements for Facial Recognition Subsystem

To ensure the facial recognition subsystem functions reliably within the overarching state machine and meets the necessary security standards for the automated card dispenser, the system design is governed by the following quantitative engineering requirements and tolerances. These parameters evaluate the integrated software pipeline rather than off-the-shelf hardware specifications.

(i) Processing Latency and State Transition Speed

- *Verification Latency:* The total time elapsed from the initiation of State 2 (activating the video stream) to generating a conclusive branching signal (Verification Success or Fail) must be 4 ± 2 s.
- *Non-blocking Constraint:* To prevent UI freezing in the Pygame environment, the dlib facial feature extraction must run asynchronously. The algorithm must not block the main UI event loop for more than 33 ms per cycle, ensuring the interface maintains a minimum refresh rate of 30 ± 2 FPS.

(ii) Recognition Accuracy and Security Metrics

- *Error Rates:* The system's False Acceptance Rate (FAR)—the probability of incorrectly granting access to an unregistered face—must be controlled at $\leq 0.1\%$. The False Rejection Rate (FRR)—the probability of failing to recognize a registered user—must be $\leq 5.0\%$ under standard laboratory conditions.
- *Similarity Threshold:* The authentication logic requires the cosine similarity between the real-time captured face embedding and the stored database embedding to satisfy the following condition to trigger State 3:

$$\text{Similarity Score} \geq 0.8 \pm 0.02$$

(iii) Environmental and Operational Tolerances

- *Operational Distance:* The face detection algorithm must successfully extract bounding boxes and feature vectors when the user's face is positioned at a distance of 45 ± 15 cm from the camera lens.
- *Illumination Range:* The visual processing pipeline must maintain the aforementioned accuracy metrics across varying lighting environments, specifically operating reliably within an ambient illuminance range of low (dim indoor lighting) to high (bright fluorescent lighting).

2.3.3 Verification tests for Facial Recognition Subsystem

To empirically validate the subsystem requirements, the following test procedures and success criteria have been established. Each test is designed to quantitatively verify the performance of the integrated software pipeline.

Test 1: Latency and Non-blocking Verification

- **Procedure:** A Python test script will be utilized to inject the `ask_face_verification` intent into the state machine 50 consecutive times. The system will log timestamps at the exact moment of intent reception and at the moment the branching signal (Success/Fail) is emitted. Concurrently, a timing hook will be added to the Pygame event loop to record the execution time of each frame during State 2.

- **Success Criterion:** The average total latency across the 50 trials must fall strictly within the 600 ms to 1000 ms range. Additionally, 100% of the recorded Pygame event loop iterations during the testing phase must execute in under 33 ms, proving zero UI blockage.

Test 2: Recognition Accuracy and Security Metric Test

- **Procedure:** A controlled physical test will be conducted involving a cohort of 5 registered users and 5 unregistered individuals. Under standard laboratory lighting, each participant will trigger the facial recognition sequence 20 times. The backend will log the calculated cosine similarity score for every attempt.

- **Success Criterion:** For unregistered users, 0 out of the 100 attempts shall yield a similarity score ≥ 0.8 (verifying the FAR requirement). For registered users, no more than 5 out of the 100 attempts shall result in a similarity score < 0.8 (verifying the requirement of $\leq 5.0\%$).

Test 3: Environmental and Operational Tolerance Test

- **Procedure:** Utilizing a measuring tape and a digital lux meter, a registered user will be positioned at the boundary operational distances (30 cm and 60 cm). Subsequently, ambient lighting will be artificially adjusted to the conditions of dim and bright. The user will perform 10 authentication attempts under each boundary condition.

- **Success Criterion:** Across all distance and illumination boundary conditions, the dlib algorithm must successfully extract the facial bounding box, and the system must return a similarity score ≥ 0.85, successfully transitioning the system to State 3.

2.3.4 Flowchart for Facial Recognition System

See figure.3

2.4 Mechanical Control System

This system is responsible for the physical card-dispensing action. After the user passes identity verification, the control module sends a signal to the motor driver, which then activates the motor or servo to move the dispensing mechanism.
The mechanism pushes or releases one card from the storage slot so that the user can collect it. This subsystem includes the motor, driver board, power supply, and the mechanical structure used to hold and dispense the cards. It must also include basic safety measures, such as limiting motor motion, preventing repeated dispensing, and stopping the motor if a

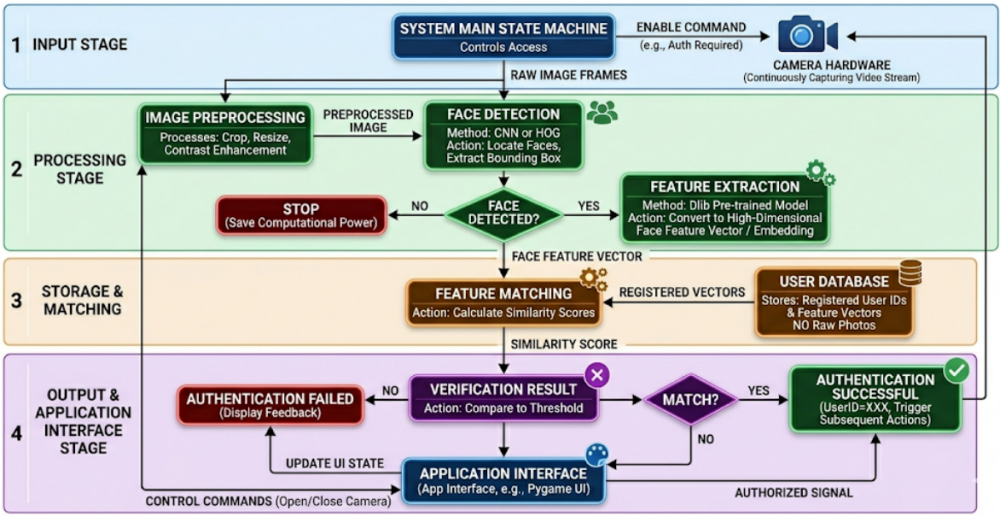

Figure 3: Flowchart for Facial Recognition System

jam or abnormal condition occurs. In this way, the mechanical control system converts the digital decision of the software into an actual physical output. It is the final execution part of the project and directly determines whether the device can operate reliably in practice.

2.4.1 Control Subsystem

This subsystem is responsible for receiving the verified command from the software and converting it into control signals for the actuator. After the user passes identity verification, the control module sends a signal to the motor driver, which then activates the motor or servo according to the predefined dispensing logic. This subsystem also includes basic safety control functions, such as limiting motor motion, preventing repeated dispensing commands, and stopping the motor if a jam or abnormal condition is detected. In this way, the mechanical control subsystem serves as the interface between the software decision layer and the physical actuation layer.

Requirement

- The Mechanical Control Subsystem shall receive the verified dispensing command and generate the appropriate control signal to drive the motor or servo, while preventing repeated or unsafe actuation.

Verification

- Send predefined verified dispensing commands to the subsystem and verify that the correct motor control signal is generated, and that repeated or abnormal commands do not result in unintended actuation.

2.4.2 Card Dispensing Mechanism Subsystem

This subsystem is responsible for the physical implementation of card storage and dispensing. It includes the motor-driven dispensing structure, the card storage slot, the guiding

mechanism, and other structural components that ensure one card can be reliably pushed out or released for user collection. Its design determines whether the cards can be dispensed smoothly, accurately, and without jamming. Therefore, this subsystem is the physical execution part of the project and directly affects the reliability and practicality of the entire device.

Requirement

- The Card Dispensing Mechanism Subsystem shall physically dispense one card reliably from the storage slot for user collection without jamming.

Verification

- Perform repeated dispensing tests and verify that the mechanism releases exactly one card each time and operates smoothly without card jams or misfeeds.

2.5 Tolerance Analysis

For this project, tolerance is defined around three critical interfaces: face-verification reliability, card-dispensing repeatability, and end-to-end response time. Unlike high-cost industrial dispensers, the proposed system uses low-cost camera and servo components, so software thresholds are used to absorb hardware variation while still preventing unsafe behavior.For identity verification, the decision is based on embedding distance between a live face vector and enrolled templates. Let

$$d = \min_i \|f_{live} - f_i\|_2 \tag{2}$$

where f_{live} is the captured feature vector and f_i is the i-th enrolled vector. Verification succeeds only when

$$d \leq \tau_f \tag{3}$$

where τ_f is the face-match tolerance. A smaller τ_f reduces false acceptance but may increase false rejection; therefore τ_f is tuned experimentally under indoor lighting conditions.
For the mechanical subsystem, the effective card displacement is modeled by

$$x = k_s \omega t \tag{4}$$

where ω is servo speed, t is drive time, and k_s maps shaft rotation to linear card motion. To ensure single-card dispensing, the command window is constrained by

$$x_{min} \leq x \leq x_{max} \tag{5}$$

where x_{min} is the minimum displacement required to eject one card and x_{max} is the upper limit before risking multi-card feed.

For interactive performance, the total response latency is bounded as

$$T_{total} = T_{asr} + T_{llm} + T_{face} + T_{motor} + T_{tts} \tag{6}$$

and the system target is to keep T_{total} within a practical front-desk waiting time. If any stage exceeds timeout, the state machine returns to `idle` and requests retry, which maintains robustness under network jitter and sensor noise.

3. Cost

Refer to Table 4.

Table 4: Project Component Cost Estimates

Component	**Price Range (CNY)**
Raspberry Pi 5 (2GB)	500 – 650
Camera Module 3	180 – 260
64GB Memory Card	30 – 50
5V/5A Power Supply	60 – 100
USB Microphone	40 – 100
Small Speaker	30 – 80
Servo	20 – 60
PCA9685 Driver Board	20 – 40
Buttons / Wires / Breadboard / Small Accessories	50 – 120
Total Estimated Cost	**930 – 1,460**

4. Schedule

See table.5

5. Ethics and Safety

This project integrates voice interaction, cloud-based conversational AI, face recognition, and motor control to create an intelligent card-dispensing system. Because the system involves both personal data and physical actuation, ethical and safety considerations are essential. First, face recognition data must be collected and stored responsibly. Users should be informed before enrollment, and facial data should only be used for identity verification related to card access. Access to the face database should be restricted to authorized administrators, and unnecessary personal information should not be collected. In addition, the system should provide a way to update or delete enrolled face data when needed.

Second, privacy and data security must be carefully addressed. Since the chatbot function relies on an online large language model, voice or text input may be transmitted to a cloud service. Sensitive information should therefore be minimized, and secure network communication should be used whenever possible. The conversational model should not directly control card dispensing. Instead, it should only identify user intent, while the final decision to dispense a card must depend on explicit face verification and permission checks. This reduces the risk of accidental or malicious triggering.

Third, physical safety must also be considered. Because the system controls a motorized card output mechanism, protective measures should be included to prevent jamming, unintended motion, or injury. The motor should operate only within limited motion ranges,

Week	Chaohua Yao	Haowen Lin	Zitong Qu	Jianchong Chen
4/6	Finalize card dispenser mechanism and component list	Finalize speech state machine and LLM label set	Finalize face recognition pipeline and camera flow	Finalize student ID matching and permission logic
4/13	Build first card ejection prototype	Build Vosk + cloud LLM prototype	Implement face detection and embedding extraction	Implement local user database and ID lookup module
4/20	Test single-card ejection reliability and revise structure	Integrate ASR output with LLM intent classification	Test face matching accuracy with enrolled users	Integrate student ID verification with face recognition
4/27	Integrate motor control with main controller for dispensing	Add speaker playback and fallback responses	Optimize verification latency and camera capture stability	Connect permission logic to global state machine
5/4	Integrate physical subsystem with overall workflow	Joint debug speech with identity subsystem	Joint debug face recognition with speech subsystem	Joint debug identity subsystem with mechanical subsystem
5/11	Verify dispensing reliability and return-to-idle behavior	Verify speech accuracy, latency, and consistency	Verify recognition threshold, FAR, and FRR	Verify access logic and exception handling
5/18	Final presentation preparation, full-system demonstration, and final document polishing			

Table 5: Project schedule and division of labor

and an emergency stop or software timeout should be implemented in case of abnormal behavior. Overall, the system should follow the principles of user consent, minimum necessary data collection, secure data handling, and fail-safe hardware control to ensure that it is both ethically responsible and safe to use.

6. References

References

[1] OpenAI and Josh Achiam et al. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.