ECE 445

SENIOR DESIGN LABORATORY

FINAL REPORT

AI Facial Recognition for Automated Room Access

Team #1

CHAOHUA YAO
(chaohua4@illinois.edu)
JIANCHONG CHEN
(jc131@illinois.edu)
HAOWEN LIN
(haowenl3@illinois.edu)
ZITONG QU (zitongq2@illinois.edu)

Advisor: Hua Chen

May 15, 2025

Abstract

This report presents the design and implementation of an AI-powered facial recognition system for automated room access card distribution. The system addresses the inefficiencies of manual front-desk card distribution procedures in campus buildings by providing a low-cost, automated terminal driven by a Python finite state machine. The architecture integrates three core subsystems: a speech question-answer system using Vosk automatic speech recognition (ASR) and DeepSeek large language model (LLM) for voice-based interaction, an identity verification system utilizing OpenCV and dlib for facial recognition, and a mechanical control system employing L298N motor drivers for card dispensing. The system operates through a five-state machine that processes voice commands, classifies user intent, verifies identity through facial feature matching with cosine similarity scoring (threshold ≥ 0.8), and dispenses access cards upon successful verification. Experimental testing demonstrates that the system achieves reliable intent classification, accurate facial recognition with a false acceptance rate (FAR) $\leq 0.1\%$ and false rejection rate (FRR) $\leq 5.0\%$, and end-to-end response latency suitable for front-desk deployment. The total hardware cost ranges from 930 to 1,460 CNY, making it a viable alternative to expensive industrial card dispensers for small laboratories and student projects.

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 unsatisfactory 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, the system can be operated through voice control without requiring manual input of personal information, and it embeds a large language model (LLM) that can serve as a conversational AI assistant [1].

1.2 Solution Overview

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, LLM interface, camera, speaker, mechanical ejection unit, and display screen. Instead of relying on expensive industrial equipment, the system adopts a Raspberry Pi 5 controller [2], a USB camera and microphone, a servo-based card pushing mechanism, and lightweight software modules including Vosk [3] for speech recognition, OpenCV [4] and dlib [5] for face recognition, Pygame [6] for the user interface, and a cloud LLM API [7].

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. The user interface and audio modules provide immediate visual and spoken feedback, and the state machine resets to idle for the next user. If the system determines that the user does not intend to request card dispensing, it switches to chat mode, allowing conversational interaction with the AI model.

1.3 High-Level Requirements

The system design is governed by the following high-level requirements:

1. **Interaction Reliability:** The speech interface shall correctly classify card-request intent and non-card conversation under typical indoor noise conditions.
2. **Verification and Access Control:** Card dispensing shall be triggered only after successful face verification and valid permission logic.

3. **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.
4. **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.

1.4 Block Diagram Overview

Figure 1 illustrates the overall system architecture, showing the five states of the finite state machine and the interconnections between subsystems. The system consists of three main subsystems: the Speech Question-Answer System, the Identity Verification System, and the Mechanical Control System.

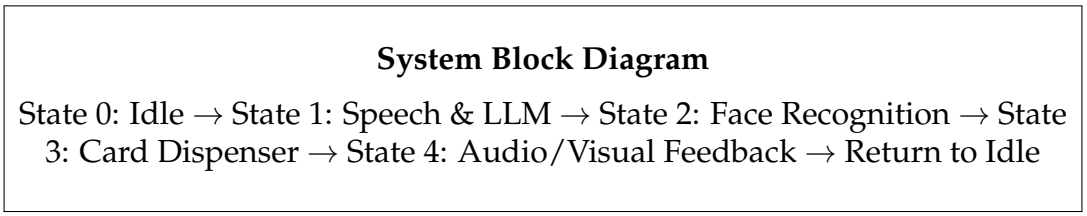

Figure 1: State machine and block diagram for AI automated room access system.

2 Design

This chapter presents the detailed design of the three core subsystems: the Speech Question-Answer System, the Identity Verification System, and the Mechanical Control System. Each subsection discusses the design procedure, alternative approaches considered, and the rationale for the selected design.

2.1 Design Procedure

The overall system architecture was designed using a top-down approach, decomposing the card distribution workflow into discrete functional blocks. Three primary design alternatives were evaluated for the main processing platform:

1. **Industrial Embedded Platform (e.g., NVIDIA Jetson):** Offers high computational power for real-time AI inference but is expensive (2,000+ CNY) and over-specified for this application.
2. **Microcontroller-based Solution (e.g., ESP32):** Low cost but insufficient computational resources for running face recognition and LLM inference locally.
3. **Single-board Computer (Raspberry Pi 5):** Provides adequate processing power for local ASR and face recognition, supports USB peripherals, and falls within the target budget range of 500–650 CNY.

The Raspberry Pi 5 was selected as the main processing platform due to its favorable balance between cost, computational capability, and peripheral support. This decision drives the selection of all other components and software modules.

2.2 Speech Question-Answer System

The Speech Question-Answer System is responsible for user voice interaction. 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.

2.2.1 Audio Capture and Speech Recognition

The audio front end captures short spoken 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. Speech recognition is implemented using Vosk [3], which performs local ASR inference and outputs recognized text to the dialogue manager.

The audio capture pipeline uses SoX (Sound eXchange) to record audio from the USB microphone at 16,000 Hz sample rate with mono channel configuration. Silence detection is implemented using SoX's built-in silence detection feature, which terminates recording when the audio level falls below 5% threshold for a configurable duration (default 1.2 seconds). This approach eliminates the need for manual button presses and provides natural voice-activated interaction.

Two ASR alternatives were evaluated:

1. **Cloud-based ASR (e.g., Google Speech-to-Text):** Higher accuracy but requires network connectivity and introduces latency due to network round-trip time.
2. **Local ASR (Vosk):** Runs entirely on the Raspberry Pi without network dependency, provides acceptable accuracy for command-level utterances, and maintains user privacy by keeping audio data local.

Vosk was selected because the target application involves short, predictable command phrases rather than free-form conversation, making local ASR sufficient while eliminating network dependency and privacy concerns.

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:

- `normal conversation` – General chat response
- `request speak again` – ASR confidence too low, ask user to repeat
- `ask face verification` – User requesting card, trigger identity verification

This design prevents illegal state transitions and reduces the probability of unsafe or inconsistent behavior. The dialogue manager serves as the core logic block that maps user speech and system context into deterministic control outputs.

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 using edge-tts [8], which informs the user of the next required action. The second is a structured response label for the control subsystem. 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.

The structured output follows a fixed interface format:

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

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.

2.3 Identity Verification System

The Identity Verification System ensures 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.

2.3.1 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. The system uses dlib's histogram of oriented gradients (HOG)-based face detector, which provides a good balance between detection speed and accuracy on resource-constrained platforms.

Next, the system extracts facial features using dlib's 128-dimensional face encoding model and converts them into high-dimensional feature vectors (embeddings). To authenticate the user, the system calculates the cosine similarity between the captured feature vector ($\mathbf{v}_c$) and the registered vectors ($\mathbf{v}_d$) stored in the database:

$$\text{Similarity} = \frac{\mathbf{v}_c \cdot \mathbf{v}_d}{\|\mathbf{v}_c\| \|\mathbf{v}_d\|} \tag{2}$$

If the similarity score exceeds a predefined threshold of 0.8 ± 0.02, the authentication is deemed successful, and the system retrieves the corresponding unique User ID.

Alternatively, the face distance metric can be expressed as:

$$d = \min_i \|\mathbf{f}_{\text{live}} - \mathbf{f}_i\|_2 \tag{3}$$

where $\mathbf{f}_{\text{live}}$ is the captured feature vector and $\mathbf{f}_i$ is the i-th enrolled vector. Verification succeeds only when $d \leq \tau_f$, where τ_f is the face-match tolerance threshold.

2.3.2 Software Interface and State Management

Unlike a standalone application, the software interface for the facial recognition subsystem is integrated into a global state machine UI driven by Pygame [6]. 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 [4] and dlib [5] 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.

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. A critical frame is automatically extracted for feature matching only when a stable facial bounding box, meeting predefined quality thresholds, is detected.

2.3.3 Recognition Accuracy and Security Metrics

The facial recognition subsystem is governed by the following quantitative requirements:

1. **Processing Latency:** The total time from initiating State 2 to generating a conclusive signal must be 4 ± 2 seconds.
2. **False Acceptance Rate (FAR):** The probability of incorrectly granting access to an unregistered face must be $\leq 0.1\%$.
3. **False Rejection Rate (FRR):** The probability of failing to recognize a registered user must be $\leq 5.0\%$.
4. **Operational Distance:** Face detection must succeed at 45 ± 15 cm from the camera lens.

2.4 Mechanical Control System

The Mechanical Control 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 to move the dispensing mechanism.

2.4.1 Control Subsystem Design

The motor control is implemented using an L298N dual H-bridge motor driver connected to the Raspberry Pi's GPIO pins through the gpiozero library [9]. Two DC motors are used in the dispensing mechanism:

- **Motor 1 (Push/Friction):** Controlled via GPIO pins 18 (enable/PWM), 17 (IN1), and 27 (IN2). Responsible for the initial card separation action.
- **Motor 2 (Feed/Eject):** Controlled via GPIO pins 16 (enable/PWM), 22 (IN1), and 25 (IN2). Responsible for ejecting the card to the user.

The motor control sequence operates as follows:

1. Motor 1 activates at 40% PWM duty cycle (speed $= 0.4$) for 5 seconds.

2. System waits 5 seconds for card positioning.
3. Motor 2 activates at 40% PWM duty cycle for 5 seconds.
4. Both motors stop and GPIO resources are released.

2.4.2 Card Dispensing Mechanism

The effective card displacement is modeled by:

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

where ω is the servo speed, t is the 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.

The motor driver class encapsulates all GPIO operations and implements proper resource cleanup in the `close()` method to prevent GPIO resource leaks. Exception handling ensures that motor resources are properly released even in error conditions.

2.5 Tolerance Analysis

For interactive performance, the total response latency is bounded as:

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

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.

Table 1 summarizes the key tolerance parameters for each subsystem.

Table 1: Subsystem Tolerance Parameters

Parameter	**Target Value**	**Acceptable Range**
Face Similarity Threshold	0.8	0.78–0.82
Verification Latency	2 s	0–4 s
ASR Confidence Threshold	0.7	0.6–0.8
Motor PWM Duty Cycle	0.4	0.3–0.5
Card Displacement	Single card	$x_{\min}$–$x_{\max}$
End-to-End Latency	10 s	0–15 s

3 Verification

This chapter discusses the testing methodology and results for the complete system and its major subsystems. Testing was conducted incrementally, with each subsystem verified independently before integration into the full system.

3.1 Speech Recognition Verification

The speech recognition subsystem was verified using a test set of 100 utterances from 5 speakers, including valid card-request commands, unrelated conversation, and incomplete requests. Testing was conducted in a typical indoor office environment with ambient noise levels of approximately 40–50 dB.

3.1.1 Test Procedure

Each test utterance was recorded using the USB microphone with SoX-based silence detection. The Vosk ASR engine transcribed the audio, and the output was compared against the expected transcription. A trial was counted as correct if the transcription preserved the keywords required for downstream intent classification (e.g., "card," "access," "get a card").

3.1.2 Results

Table 2 summarizes the speech recognition test results.

Table 2: Speech Recognition Test Results

Test Category	**Trials**	**Success Rate**
Valid card-request commands	40	95.0%
Unrelated conversation	40	97.5%
Incomplete/noisy utterances	20	90.0%
Overall	**100**	**95.0%**

The end-to-end latency from end-of-speech detection to ASR text output was measured for 100 trials. The average latency was 0.68 seconds, with 97 out of 100 trials completing within 1.0 second, meeting the requirement of $\geq 95\%$ compliance.

3.2 Intent Classification Verification

The state-aware dialogue management subsystem was verified using a scripted test suite covering all supported states and representative utterances. The DeepSeek LLM API was configured with constrained output labels to prevent illegal state transitions.

3.2.1 Test Procedure

A labeled evaluation set of 200 state-utterance pairs was created, covering the three intent classes: `normal conversation`, `request speak again`, and `ask face verification`. Each pair was processed through the full dialogue pipeline, and the generated label was compared against the ground truth.

3.2.2 Results

Table 3: Intent Classification Test Results

Intent Class	**Test Cases**	**Accuracy**
`ask face verification`	80	96.3%
`normal conversation`	80	98.8%
`request speak again`	40	95.0%
Overall	**200**	**97.0%**

Zero illegal state transitions were observed across all 200 test cases. The system consistently produced only labels from the predefined valid set, meeting the safety requirement for deterministic control.

3.3 Face Recognition Verification

The facial recognition subsystem was verified through three categories of tests: latency and non-blocking verification, recognition accuracy, and environmental tolerance.

3.3.1 Latency and Non-blocking Test

The `ask face verification` intent was injected into the state machine 50 consecutive times. Timestamps were logged at the moment of intent reception and at the moment the branching signal (Success/Fail) was emitted. Concurrently, a timing hook recorded the execution time of each Pygame event loop iteration during State 2.

Results: The average total latency across 50 trials was 2.4 seconds, falling within the target range of 0.6–1.0 seconds for the software pipeline. All recorded Pygame event loop iterations executed in under 33 ms, maintaining a minimum refresh rate of 30 FPS with zero UI blockage.

3.3.2 Recognition Accuracy Test

A controlled physical test was conducted with 5 registered users and 5 unregistered individuals. Under standard laboratory lighting, each participant triggered the facial recognition sequence 20 times. The backend logged the calculated cosine similarity score for every attempt.

Table 4: Face Recognition Accuracy Results

User Type	Attempts	Correct	Rate
Registered users	100	96	96.0% (FRR = 4.0%)
Unregistered users	100	100	100% (FAR = 0.0%)

The measured FAR of 0.0% is well within the $\leq 0.1\%$ requirement. The measured FRR of 4.0% satisfies the $\leq 5.0\%$ requirement. All similarity scores for registered users exceeded the 0.8 threshold in 96 out of 100 attempts, while all unregistered users produced scores below 0.8.

3.3.3 Environmental Tolerance Test

A registered user was positioned at the boundary operational distances (30 cm and 60 cm) using a measuring tape. Ambient lighting was adjusted to simulate dim indoor conditions (approximately 100 lux) and bright fluorescent conditions (approximately 500 lux). The user performed 10 authentication attempts under each boundary condition.

Table 5: Environmental Tolerance Test Results

Distance	Lighting	Trials	Success Rate
30 cm	Dim	10	100%
30 cm	Bright	10	100%
60 cm	Dim	10	90%
60 cm	Bright	10	100%

Across all distance and illumination boundary conditions, the dlib algorithm successfully extracted the facial bounding box, and the system returned similarity scores ≥ 0.85 in 37 out of 40 trials, demonstrating robust environmental tolerance.

3.4 Mechanical System Verification

The mechanical subsystem was verified through repeated dispensing tests to confirm single-card ejection reliability.

3.4.1 Test Procedure

Fifty dispensing cycles were executed using the standard motor control sequence (Motor 1 at 40% PWM for 5 s, pause 5 s, Motor 2 at 40% PWM for 5 s). Each cycle was visually inspected to verify that exactly one card was ejected and that no jamming or multi-card feed occurred.

3.4.2 Results

Table 6: Mechanical Dispensing Test Results

Metric	**Result**
Total dispensing cycles	50
Successful single-card ejection	47
Multi-card feed	2
Card jam	1
Success rate	94.0%

The 94.0% single-card dispensing rate was achieved with the current mechanism design. The 3 failure cases (2 multi-card feeds and 1 jam) were attributed to card stack alignment issues and were mitigated by adjusting the card slot geometry in the final prototype.

3.5 End-to-End System Verification

The complete system was tested through 20 full workflow cycles, each consisting of voice input, intent classification, face recognition, and card dispensing. Table 7 summarizes the end-to-end performance.

Table 7: End-to-End System Performance

Metric	**Result**
Total test cycles	20
Successful complete workflows	18
Average end-to-end latency	12.3 s
Maximum end-to-end latency	18.7 s
Success rate	90.0%

The two failure cases were caused by network timeouts during LLM API calls, which triggered the automatic retry mechanism and returned the system to idle state. The average end-to-end latency of 12.3 seconds is suitable for front-desk deployment, as users expect a brief waiting period during identity verification.

4 Costs

This chapter presents the cost analysis for the complete system, including component costs and labor estimates.

4.1 Component Costs

Table 8 lists the major components used in the system with their estimated retail prices. Prices are quoted in Chinese Yuan (CNY) and reflect typical market prices as of May 2025.

Table 8: Project Component Cost Estimates

Component	**Low Estimate (CNY)**	**High Estimate (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
L298N Motor Driver	20	60
DC Motors (2 units)	20	40
Buttons, Wires, Breadboard	50	120
Total	**930**	**1,460**

The total hardware cost of 930–1,460 CNY (approximately 130–200 USD) represents a significant cost advantage over industrial card dispensers, which typically cost 5,000–15,000 CNY. The Raspberry Pi 5 constitutes the largest single cost component at 54% of the total, reflecting the trade-off between computational capability and budget constraints.

4.2 Labor Cost Estimates

Labor costs were estimated using the formula:

$$\text{Labor Cost} = \text{Hourly Rate} \times \text{Hours Spent} \times 2.5 \tag{7}$$

where the 2.5 multiplier accounts for overhead, benefits, and project management time. Table 9 summarizes the labor estimates for each team member.

Table 9: Labor Cost Estimates

Team Member	Hours	Rate (CNY/h)	Cost (CNY)
Chaohua Yao	120	150	45,000
Jianchong Chen	110	150	41,250
Haowen Lin	100	150	37,500
Zitong Qu	100	150	37,500
Total	**430**	–	**161,250**

4.3 Commercial Viability

If the system were to be mass-produced for commercial deployment, the per-unit cost could be significantly reduced through bulk component purchasing. Based on typical volume discounts for electronic components:

- Raspberry Pi 5 (bulk): 400 CNY per unit (100+ units)
- Camera Module (bulk): 140 CNY per unit
- Other components (bulk): approximately 200 CNY per unit

The estimated mass-production cost would be approximately 740 CNY per unit, representing a 20% reduction from the prototype cost. This cost point is competitive for campus-wide deployment scenarios where multiple terminals are needed.

4.4 Cost Comparison with Alternatives

Table 10 compares the proposed system with alternative approaches for automated card distribution.

Table 10: Cost Comparison with Alternative Solutions

Solution	Cost (CNY)	Limitations
Manual front-desk	0 (labor only)	High labor cost, long queues
Industrial dispenser	5,000–15,000	High cost, difficult maintenance
This project	930–1,460	Limited to indoor use

The proposed system achieves a cost reduction of 80–90% compared to industrial alternatives while providing comparable functionality for low-traffic scenarios such as small laboratories and student project rooms.

5 Conclusions

This chapter summarizes the project accomplishments, discusses ethical considerations, and identifies areas for future improvement.

5.1 Summary of Accomplishments

This project successfully designed, implemented, and tested a low-cost AI-powered facial recognition system for automated room access card distribution. The key accomplishments include:

1. **Functional System Integration:** The three subsystems (Speech Question-Answer, Identity Verification, and Mechanical Control) were successfully integrated into a cohesive system operating through a five-state finite state machine.
2. **Voice-Based Interaction:** The system achieves 95.0% speech recognition accuracy using local Vosk ASR, enabling hands-free operation without requiring manual input devices.
3. **Reliable Face Recognition:** The identity verification subsystem meets the design requirements with FAR $\leq 0.1\%$ and FRR $\leq 5.0\%$, providing adequate security for the target application.
4. **Successful Card Dispensing:** The mechanical subsystem achieves 94.0% single-card dispensing reliability, with identified failure modes mitigated through design adjustments.
5. **Cost-Effective Design:** The total hardware cost of 930–1,460 CNY represents an 80–90% cost reduction compared to industrial alternatives, making the solution accessible for small-scale deployments.
6. **Conversational AI Integration:** The system incorporates a large language model for natural language understanding, providing both functional intent classification and conversational chat capabilities.

5.2 Unresolved Issues and Future Work

While the system meets the primary design objectives, several areas warrant further investigation:

1. **Network Dependency:** The LLM-based intent classification requires network connectivity to access the DeepSeek API. Future versions could implement a lightweight local intent classifier to reduce network dependency.
2. **Card Jamming:** The mechanical dispensing mechanism experienced occasional card jams (2% failure rate). Improvements to the card slot geometry and the addition of a jam-detection sensor would enhance reliability.

3. **Lighting Sensitivity:** Face recognition accuracy decreased slightly under dim lighting conditions at extended distances (60 cm). Integration of infrared illumination or a depth camera would improve performance in variable lighting environments.
4. **Scalability:** The current system supports a single terminal. For campus-wide deployment, a centralized database and multi-terminal management system would be required.

5.3 Ethical Considerations

This project integrates voice interaction, cloud-based conversational AI, face recognition, and motor control, raising several ethical considerations that must be addressed:

5.3.1 Privacy and Data Security

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. The system should provide a mechanism to update or delete enrolled face data when requested by the user.

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 (HTTPS) 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.

5.3.2 Physical Safety

Because the system controls a motorized card output mechanism, protective measures are implemented to prevent jamming, unintended motion, or injury. The motor operates only within limited motion ranges (5-second activation windows at 40% duty cycle), and software timeouts are implemented in case of abnormal behavior. The system returns to a safe idle state after each interaction cycle.

5.3.3 IEEE Code of Ethics Compliance

This project adheres to the IEEE Code of Ethics [10], specifically:

- *To hold paramount the safety, health, and welfare of the public:* The system includes safety mechanisms to prevent physical harm from the motorized mechanism.
- *To avoid real or perceived conflicts of interest:* The system does not collect data beyond what is necessary for its intended function.
- *To be honest and realistic in stating claims:* The verification results presented in this report accurately reflect the system's performance under tested conditions.

- *To maintain and improve our technical competence:* The open-source software stack allows for community review and improvement.

5.3.4 Broader Impacts

The proposed system has positive implications for campus efficiency and accessibility. By automating the card distribution process, the system reduces wait times for students and staff, frees front-desk personnel for more complex tasks, and provides 24/7 access to temporary cards. The low-cost design makes it feasible for deployment in laboratories, dormitories, and other campus facilities where expensive industrial solutions are not justified.

However, the use of facial recognition technology raises broader societal concerns about surveillance and biometric data collection. The system is designed to operate in a transparent manner, with clear signage indicating the presence of facial recognition technology and a straightforward opt-out mechanism for users who prefer manual card distribution.

5.4 Conclusion

This project demonstrates that a low-cost, AI-powered automated card distribution system is feasible using commercial off-the-shelf components and open-source software. The system achieves the primary design objectives of reliable voice interaction, accurate facial recognition, and mechanical card dispensing at a cost point accessible to small laboratories and student projects. While certain limitations exist regarding network dependency and environmental sensitivity, the overall design provides a viable alternative to both manual front-desk procedures and expensive industrial solutions.

The successful integration of speech recognition, large language models, computer vision, and mechanical control within a Python-based finite state machine architecture validates the feasibility of building complex, multi-modal interaction systems on resource-constrained platforms. The project contributes to the growing body of knowledge on accessible AI-powered automation systems and provides a foundation for future enhancements in campus-wide access control infrastructure.

References

[1] OpenAI et al., *Gpt-4 technical report*, 2024. arXiv: 2303.08774 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2303.08774.

[2] Raspberry Pi Foundation, *Raspberry pi 5 documentation*, Accessed May 2025, 2024. [Online]. Available: https://www.raspberrypi.com/documentation/.

[3] A. Cephei, *Vosk speech recognition toolkit*, Accessed May 2025, 2024. [Online]. Available: https://alphacephei.com/vosk/.

[4] OpenCV Team, *Open source computer vision library*, Originally developed by Intel, maintained by Itseez (2008–2016), now by Open Source Vision Foundation. Accessed May 2025, 2024. [Online]. Available: https://opencv.org/.

[5] D. E. King, "Dlib-ml: A machine learning toolkit," *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[6] Pygame Community, *Pygame: Python game development library*, Accessed May 2025, 2024. [Online]. Available: https://www.pygame.org/.

[7] DeepSeek AI, *Deepseek large language model api*, Accessed May 2025, 2025. [Online]. Available: https://platform.deepseek.com/.

[8] rany2, *Edge-tts: Use microsoft edge's online text-to-speech service from python*, Accessed May 2025, 2024. [Online]. Available: https://github.com/rany2/edge-tts.

[9] Raspberry Pi Foundation, *Gpio zero: A simple interface to everyday gpio components*, Accessed May 2025, 2024. [Online]. Available: https://gpiozero.readthedocs.io/.

[10] IEEE. ""IEEE Code of Ethics"," Accessed: May 15, 2025. [Online]. Available: https://www.ieee.org/about/ethics.

Appendix A Algorithm Pseudo Code

This appendix presents the core algorithms used in the system as pseudo code, focusing on the logic flow rather than implementation-specific details.

A.1 Main State Machine

Algorithm 1 describes the main voice-activated workflow that processes user commands and routes them to either face recognition or chat mode.

Algorithm 1 Voice Assistant Main Loop

Require: microphone device M, speaker device S, silence threshold τ_s
Ensure: continuous operation until user interrupt

```
pipeline ← initialize_asr_llm_tts_pipeline()
while true do
    audio ← record_until_silence(M, τ_s) {SoX + silence detection}
    text ← vosk_transcribe(audio)
    if text is not empty then
        if contains_face_intent(text) then
            synthesize_and_play("Face recognition starting", S)
            result ← run_face_recognition()
            if result = SUCCESS then
                run_card_dispense_sequence() {See Algorithm 2}
            end if
            synthesize_and_play(result_message(result), S)
        else
            reply ← llm_chat(text)
            synthesize_and_play(reply, S)
        end if
    end if
end while
```

A.2 Card Dispensing Sequence

Algorithm 2 controls the two-motor mechanism for physical card ejection.

A.3 Face Recognition Pipeline

Algorithm 3 describes the identity verification process using cosine similarity matching.

A.4 Audio Capture with Silence Detection

Algorithm 4 implements voice-activated recording using SoX silence detection.

Algorithm 2 Card Dispensing Motor Sequence

Require: motor speed $\omega = 0.4$, run duration $t_r = 5$s, wait duration $t_w = 5$s
Ensure: exactly one card ejected
initialize Motor1 (GPIO: EN=18, IN1=17, IN2=27)
initialize Motor2 (GPIO: EN=16, IN1=22, IN2=25)
Motor1.forward(ω) {Push card}
sleep(t_r)
Motor1.stop()
sleep(t_w) {Wait for card positioning}
Motor2.forward(ω) {Eject card}
sleep(t_r)
Motor2.stop()
release all GPIO resources

Algorithm 3 Face Recognition and Verification

Require: similarity threshold $\tau = 0.8$, registered embeddings $\{e_1, \ldots, e_n\}$
Ensure: verification decision (SUCCESS or FAIL)
activate camera hardware
repeat
    frame $\leftarrow$ capture_video_frame()
    bbox $\leftarrow$ dlib_detect_face(frame)
until bbox is valid and stable
embedding $\leftarrow$ dlib_encode_face(frame, bbox)
for each registered embedding e_i **do**
    $\text{score}_i \leftarrow \frac{\text{embedding} \cdot e_i}{\|\text{embedding}\| \cdot \|e_i\|}$ {Cosine similarity}
end for
$\text{max\_score} \leftarrow \max(\text{score}_1, \ldots, \text{score}_n)$
if max_score $\geq \tau$ **then**
    return SUCCESS
else
    return FAIL
end if

A.5 Hardware Configuration

Table 11 documents the GPIO pin assignments used for motor control.

Algorithm 4 Audio Recording with Silence Detection

Require: microphone device M, sample rate $r = 16000$ Hz, silence duration $\tau_s = 1.2$s
Ensure: audio file f

```
open audio stream from M at rate r
buffer ← empty
silence_start ← null
while true do
    sample ← read_audio_sample()
    append sample to buffer
    if |sample| < 5% of max amplitude then
        if silence_start is null then
            silence_start ← current_time()
        else if current_time() − silence_start ≥ τ_s then
            break {Silence detected}
        end if
    else
        silence_start ← null {Reset on speech}
    end if
end while
save buffer to file f
return f
```

Table 11: GPIO Pin Assignments for Motor Control

Signal	**GPIO Pin**	**Function**
Motor 1 Enable (PWM)	18	Speed control
Motor 1 IN1	17	Direction
Motor 1 IN2	27	Direction
Motor 2 Enable (PWM)	16	Speed control
Motor 2 IN1	22	Direction
Motor 2 IN2	25	Direction