International Journal of Engineering Science Invention Research & Development; Vol. 13, Issue 4, October 2026
www.ijesird.com, E-ISSN: 2349-6185

AUTOMATED SMART GATE PASS AND IDENTITY VERIFICATION SYSTEM FOR SECURE CAMPUS ACCESS

K. Sathwika, N. Sathyasri, S. Navya, Dr. B. Venkateswarlu

Department of Computer Science and Engineering, Geethanjali College of Engineering and Technology, India

23r11a67g1@gcet.edu.in, 23r11a67g6@gcet.edu.in, 23r11a67h9@gcet.edu.in, bvenkat1109@gmail.com

***Abstract*— Manual gate pass procedures in educational institutions rely on handwritten slips and verbal confirmation, which makes them slow, easy to forge and difficult to audit. This paper presents the Automated Smart Gate Pass and Identity Verification System (SmartGPS), a role-based web platform that digitizes the complete gate pass life cycle for secure campus access. A student submits an exit request, the registered parent is notified and confirms it, the Head of Department (HoD) approves or rejects it, and the security guard releases the student only after a live face scan matches the enrolled face. Identity matching uses face embeddings from the VGG-Face convolutional network and cosine similarity with an acceptance threshold of 0.40. The system is built with React and TypeScript, a Node.js and Express REST API, MongoDB, and a Python Flask microservice based on DeepFace and OpenCV. Passwords are hashed with bcrypt, sessions use JSON Web Tokens, and every significant action is written to an audit log. Functional test scenarios, limitations and directions for quantitative biometric evaluation are discussed.**

***Keywords*— Gate Pass Management, Face Recognition, VGG-Face, Cosine Similarity, Campus Security, Role-Based Access Control, Identity Verification**

I. INTRODUCTION

Controlling who leaves a campus, when, and with whose permission is a routine but safety-critical task for colleges and universities. In many institutions a student who must leave early obtains a handwritten gate pass signed by a faculty member or the HoD and shows it to the security guard. This process has several weaknesses: slips can be forged or reused, the guard cannot confirm that the person carrying the slip is the person named on it, parents are rarely informed in real time, and no searchable record is kept.

Biometric verification can close this gap. Face recognition is attractive for campus gates because it is contactless, needs only an inexpensive camera, and does not require students to carry any additional token. Deep convolutional networks such as VGG-Face [2] map a face image to a numerical embedding so that images of the same person lie close together while images of different people lie far apart.

This paper presents SmartGPS. The main contributions are: (i) a four-role workflow that combines parent confirmation, HoD approval and guard verification before a student may exit; (ii) a face verification pipeline based on VGG-Face embeddings and cosine similarity that is applied only to students holding an active approved pass; and (iii) an audit trail that records requests, approvals, successful verifications and denied attempts.The remainder of the paper is organized as follows. Section II reviews related work, Section III describes the architecture, Section IV explains the workflow and the matching method, Section V presents the implementation, Section VI discusses testing and limitations, and Section VII concludes.

II. RELATED WORK

Deep learning has transformed face verification. DeepFace [1] approached human-level accuracy on unconstrained face images by combining explicit alignment with a deep network. Parkhi et al. [2] showed that a very deep network in the style of Simonyan and Zisserman [4], trained on a large face dataset, produces descriptors that can be compared directly for verification; this network is known as VGG-Face. FaceNet [3] instead learns an embedding directly with a triplet loss so that Euclidean distance reflects face similarity.

Before deep networks, face detection commonly relied on boosted cascades of simple features [5], which remain available in the OpenCV library [9]. Open-source frameworks such as the DeepFace library [6] wrap several recognition models, including VGG-Face, behind a simple interface; the biometric service in this work uses that library.

On campuses, face recognition is commonly applied to attendance marking. Gate pass control is different because identity verification must be tied to an explicit authorization: a recognized face is not sufficient unless the person also holds a valid, approved request. SmartGPS therefore couples biometric matching with a multi-party approval workflow.

III. SYSTEM ARCHITECTURE

A. Layered Architecture

SmartGPS follows the four-part architecture shown in Fig. 1. The presentation layer is a single-page application written in React and TypeScript that provides a separate portal for each role. The application layer is a Node.js and Express REST API that exposes authentication, gate pass, department, guard and administration endpoints. The data layer is a MongoDB database accessed through Mongoose. The biometric engine is a Flask microservice that extracts VGG-Face embeddings with DeepFace and OpenCV and offers health, embedding-extraction and embedding-comparison endpoints.

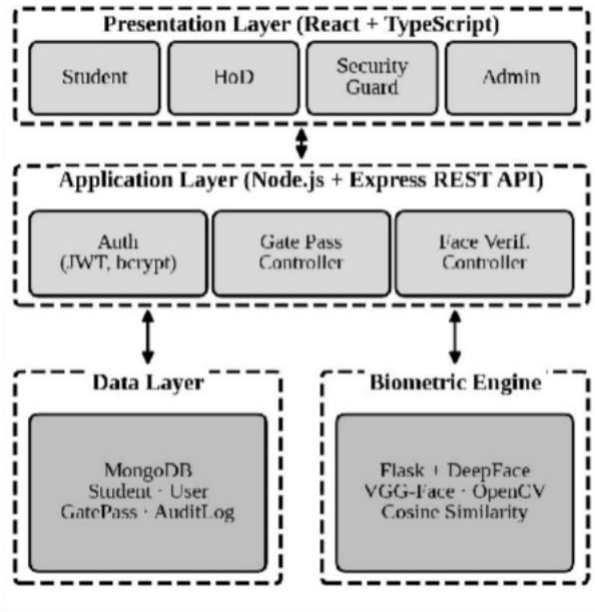


Fig. 1 Layered architecture of SmartGPS

B. Roles and Responsibilities

Table I summarizes the roles. Students are identified by a ten-character alphanumeric roll number, while staff accounts are identified by institutional e-mail style user names.

TABLE I
ROLES AND RESPONSIBILITIES

Role	Responsibilities
Student	Enrolls face image, submits gate pass request, tracks status
Parent	Receives notification and confirms the request to the HoD
HoD	Reviews requests, approves or rejects with remarks, views student history
Guard	Views approved passes, scans face, marks student as left campus
Admin	Enrolls or removes students; views statistics, users and audit logs

C. Data Model

Four collections are used. The Student collection stores the roll number, name, branch, year, section, parent mobile number, face image, facial embedding vector and a hashed password. The User collection stores staff accounts with role, user name and department. The GatePass collection links a student to a requested date, time and reason, together with the parent confirmation flag and timestamp, HoD remarks, status, the similarity score recorded at verification and the exit time. The AuditLog collection stores the action, actor role, actor identifier, student identifier, similarity score, status and timestamp of each event.

IV. METHODOLOGY

A. Gate Pass Workflow

Fig. 2 shows the life cycle of a gate pass. A request moves through the states Pending, Approved or Rejected, and finally Left Campus.

1) Request Submission: The student enters the date, time and reason. The system checks that the roll number exists and that the student has no other pending request, stores the pass in the Pending state and notifies the parent's registered mobile number.

2) Parent Confirmation: The parent confirms the request to the HoD, which sets a confirmation flag and timestamp on the pass.

3) HoD Decision: The HoD reviews the request together with its parent confirmation status and either approves or rejects it with optional remarks.

4) Gate Verification: The guard captures a live image at the gate and the system performs the face verification described in Section IV-B. If it succeeds, the guard marks the student as Left Campus and the exit time is stored. Only a pass in the Approved state can be marked in this way.

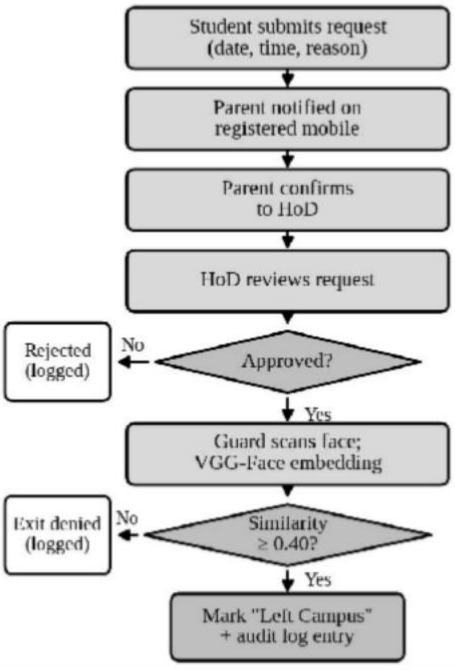


Fig. 2 Gate pass workflow from request to campus exit

B. Face Representation and Matching

At enrollment the student's face image is converted into an embedding vector and stored with the student record. Pixel values are first scaled to the range [0, 1] by dividing by 255. At the gate a second embedding is computed from the live image. Let A be the live embedding and B the stored embedding. Their similarity is the cosine of the angle between them,

$$S(A, B) = (A \cdot B) / (\|A\| \, \|B\|) = \Sigma A_i B_i / (\sqrt{\Sigma A_i^2} \cdot \sqrt{\Sigma B_i^2}) \quad (1)$$

where the sums run over the embedding dimensions. The value lies between −1 and 1, and larger values indicate more similar faces. A match is accepted when

$$S(A, B) \geq T, \quad T = 0.40 \tag{2}$$

Verification is restricted to students who currently hold an approved pass and have not yet left. The live embedding is compared with the stored embedding of each such student, the candidate with the highest similarity is selected, and exit is allowed only if this value reaches T. A scan that finds no approved pass, or whose best similarity is below T, is denied and recorded in the audit log together with the score. The confidence shown to the guard is the similarity expressed as a percentage limited to the range 0 to 100.

C. Security Mechanisms

Passwords are hashed with bcrypt [8] using a cost factor of 10 before storage. Successful logins return a JSON Web Token [7] that expires after one day. Roll numbers must be exactly ten characters and are stored in upper case, a student cannot hold more than one pending request, and request bodies are limited to 15 MB to accommodate image payloads. Every request, approval, parent confirmation, verification and denial generates an audit entry, so a disputed exit can be traced to an actor and a time.

V. Implementation

A. Technology Stack

The technologies used in the prototype are listed in Table II.

TABLE II
Technology Stack

Layer	Technology
Front end	React 19, TypeScript, Vite, Tailwind CSS
Back end	Node.js, Express, Mongoose
Database	MongoDB
Biometrics	Python, Flask, DeepFace (VGG-Face), OpenCV, TensorFlow
Security	bcryptjs, JSON Web Token
Messaging	Twilio SDK (provisioned for SMS)

B. User Portals

The student portal supports registration with a camera-captured face image, request submission and status tracking. The HoD portal lists requests with their parent confirmation status, supports approval and rejection with remarks, shows a student's history, and can identify a student from a face photograph. The security portal shows approved passes, captures the live face, displays the similarity score, the threshold and the verdict, and records the exit. The administrator portal manages student enrollment and removal and displays system statistics, user accounts and audit logs. A seed script creates the initial HoD, guard and administrator accounts.

C. Prototype Scope

The code base has two cooperating halves. The browser application can run on its own: it keeps data in browser storage and uses a lightweight 128-dimensional image descriptor built from regional luminance, color-difference and edge features in place of a deep network, and parent confirmation is simulated by a button in the student portal. This mode is intended for demonstration and interface evaluation. In it, the HoD face lookup uses a lower threshold of 0.35 because it only assists searching and does not release a student. The server-side stack provides the production path with persistent MongoDB storage, REST endpoints and VGG-Face embeddings from the Python microservice.

VI. Testing and Discussion

A. Functional Test Scenarios

Table III lists the principal scenarios that the implemented rules are designed to handle. They cover input validation, the workflow state machine and the verification decision.

TABLE III
Key Functional Test Scenarios

Scenario	Expected outcome
Roll number is not ten characters	Registration rejected
Second request while one is pending	Request refused with message
Delete an approved request	Deletion refused
Scan with no approved pass	Access denied and logged
Best similarity ≥ 0.40	Match accepted, exit allowed
Best similarity < 0.40	Exit denied, mismatch logged
Mark exit for a non-approved pass	Operation refused

B. Limitations and Future Work

The present work has limits. First, no quantitative accuracy evaluation such as false acceptance rate and false rejection rate has been carried out, and the threshold of 0.40 should be calibrated on a representative campus dataset. Second, the system has no liveness detection, so a printed photograph or replayed video could be presented to the camera. Third, the browser demonstration descriptor is not a trained face model and must not be used for real access control. Fourth, token-based authorization should be enforced on all protected endpoints, secrets should be supplied through environment configuration, and default credentials must be changed in deployment. Future work includes liveness detection, SMS or one-time-password based parent confirmation through a gateway, and evaluation under varied lighting and camera conditions.

VII. Conclusions

This paper presented SmartGPS, a role-based system that replaces paper gate passes with a digital workflow involving the student, parent, HoD and security guard, and that verifies identity at the gate using VGG-Face embeddings and cosine similarity. The layered design, audit trail and password and token security make exits traceable and harder to forge. Calibrating the threshold with measured error rates and adding liveness detection are the main steps required before deployment at a live campus gate.

Acknowledgment

The authors thank the Department of Computer Science and Engineering and the management of Geethanjali College of Engineering and Technology for their support.

References

[1] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1701–1708.

[2] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *Proc. British Machine Vision Conf. (BMVC)*, 2015, pp. 41.1–41.12.

[3] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE CVPR*, 2015, pp. 815–823.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2015.

[5] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE CVPR*, 2001, vol. 1, pp. 511–518.

[6] S. I. Serengil and A. Ozpinar, "LightFace: A hybrid deep face recognition framework," in *Proc. Innovations in Intelligent Systems and Applications Conf. (ASYU)*, 2020, pp. 1–5.

[7] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," *RFC 7519*, IETF, May 2015.

[8] N. Provos and D. Mazières, "A future-adaptable password scheme," in *Proc. USENIX Annual Technical Conf.*, 1999, pp. 81–91.

[9] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.