

E-Authentication System Using Face Recognition with Spring Boot

Joga Akhil¹, Prof. Kuda Nageswara Rao²

¹PG Scholar (M.Tech, Computer Science and Technology), Department of Computer Science and Systems Engineering, Andhra University College of Engineering (A), Visakhapatnam, Andhra Pradesh, India

²Professor, Department of Computer Science and Systems Engineering, Andhra University College of Engineering (A), Visakhapatnam, Andhra Pradesh, India

Abstract—Password- and token-based authentication remain exposed to credential reuse, phishing and loss, whereas face recognition offers a contactless alternative that needs only a commodity camera. This paper presents AJFace, an e-authentication system built as three independently deployable services: a Java 21 / Spring Boot 3 backend that issues JSON Web Tokens and enforces role-based and object-level access control; a Python / FastAPI AI service that performs detection and recognition with the InsightFace toolkit (RetinaFace detector, ArcFace 512-dimensional embeddings, cosine-similarity search); and a React single-page frontend. The AI service contributes a threshold-anchored similarity-to-confidence rescaling that keeps its decision rule and the backends' rule in agreement, an enrolment-time duplicate-identity check, and an image-quality and heuristic liveness screen that is documented as a placeholder rather than a trained anti-spoofing model. The system is exercised through an enterprise attendance case study in which one face-verification event both authenticates the user and records attendance, supported by leave, payroll, alerting and reporting modules. Twenty-two automated component tests of the AI service pass, and a functional walkthrough of the running prototype confirms the main workflows. No large-scale false-acceptance/false-rejection or latency benchmark was performed; the paper states these limitations explicitly, together with directions for future work.

Index Terms—face recognition, e-authentication, ArcFace, Spring Boot, microservices, JSON Web Token.

I. INTRODUCTION

Authentication mechanisms are conventionally grouped into knowledge factors (passwords, PINs), possession factors (cards, tokens, one-time-password

devices) and inherence, or biometric, factors. Knowledge-based credentials remain dominant, yet they are weakened by reuse across services, susceptibility to phishing and brute-force attack, and the burden that complexity rules place on users. Biometric authentication ties the credential to the person, and face recognition is especially practical because it can be captured contact lessly with the webcam or phone camera already present on most devices.

Modern deep face-recognition models, trained with margin-based losses, map a face to a compact embedding that is far more tolerant of lighting, pose and expression than classical subspace methods. The remaining difficulty is largely one of engineering: recognition is frequently embedded inside a single application so it cannot be reused or scaled on its own; raw similarity scores are hard for administrators to interpret; one person can be enrolled twice under different identities; spoofing with a photograph is rarely addressed; and unmatched attempts often leave no audit trail.

This paper presents AJFace, an e-authentication system in which the user's face replaces the password as the primary credential. To validate the authentication engine under a continuously exercised workload rather than a one-off login demonstration, it is deployed as the recognition core of an enterprise attendance application.

The principal contributions are:

- A three-service architecture that decouples AI inference from business logic behind a narrow REST contract (/enroll, /recognize);

- A threshold-anchored similarity-to-confidence rescaling that keeps the AI service and the backend applying the same accept/reject rule;
- An enrolment-time cross-employee duplicate-identity check together with image-quality gates and a clearly disclosed heuristic liveness screen;
- Layered access control combining JWT authentication, role-based rules and object-level “self-or-admin” ownership checks, including an access-control defect found and corrected during prototype testing; and
- An honest evaluation that separates published benchmark figures for the underlying models from this system's own functional observations.

The remainder of the paper is organised as follows: Section II reviews related work; Section III states the problem; Section IV describes the architecture; Section V details the AI-service algorithms; Section VI covers backend security; Section VII reports testing and results; Section VIII discusses limitations; and Section IX concludes.

II. RELATED WORK

A. From Subspace Methods to Deep Embeddings

Eigenfaces [1] applied principal component analysis to face images, and Fisherfaces [2] replaced it with linear discriminant analysis to better separate identities; both are sensitive to illumination and pose. The Viola–Jones boosted cascade [6] made real-time face detection practical.

Deep metric-learning models such as FaceNet [3], which optimises a triplet loss, and ArcFace [4], which adds an additive angular margin to the softmax loss, produce highly discriminative embeddings; ArcFace reports accuracy above 99.8% on the LFW benchmark [4]. RetinaFace [5] is a single-stage detector that localises faces and five facial landmarks, enabling alignment before recognition. Our system adopts the RetinaFace + ArcFace pairing through the open-source InsightFace toolkit [12].

B. Face-Recognition Attendance Systems

Face-based attendance has been widely explored. Pawaskar and Chavan [9], for example, describe a class-management and attendance system built on classical detection and recognition techniques (Haar cascade and LBPH). Such systems typically target attendance alone. AJFace instead treats recognition as

a reusable authentication service and uses attendance as a case study that continuously exercises it.

C. Anti-Spoofing and Template Protection

Presentation attacks—printed photographs, screen replays and masks—are a central threat to face authentication; Yu *et al.* [10] survey deep-learning countermeasures. Because face embeddings are sensitive and cannot be reissued like a password, template-protection schemes such as cancellable biometrics have been proposed; Krivokuća Hahn and Marcel [11] survey methods for neural-network-based face systems. AJFace implements only a heuristic liveness screen and conventional storage protection, and we identify both as limitations rather than solved problems.

D. Microservices and Token-Based Access

Independently deployable services communicating over lightweight protocols [7] allow a compute-heavy inference component to scale and evolve separately from business logic. Stateless JSON Web Tokens [8] carry identity and role claims between client and server. Broken access control, including object-level authorisation failures, is ranked first in the OWASP Top Ten [13], which motivates the ownership checks described in Section VI.

III. PROBLEM STATEMENT AND OBJECTIVES

Password-based verification is weak in practice, and many existing face-recognition systems use classical, condition-sensitive methods, bundle recognition and business logic into one application, provide no liveness safeguard, and offer no guarantee that a separately deployed recognition component and its consumer agree on what constitutes a match. The problem addressed here is to design and validate an e-authentication system that replaces password verification with real-time deep face recognition, is engineered as an independently scalable service, and is tested under a sustained, realistic workload.

The objectives are: O1 build a face-based authentication engine on ArcFace/RetinaFace; O2 expose it to applications through a stable REST contract; O3 guarantee a single, consistent accept/reject rule across services; O4 prevent duplicate identities and log every attempt, matched or not; O5 enforce role-based and object-level access

control; and O6 validate the engine inside a complete attendance application.

IV. SYSTEM ARCHITECTURE

A. Services and Technology

AJFace comprises three services (Fig. 1 and Table I). The *React frontend* captures a camera frame and renders role-specific portals. The *Spring Boot backend* owns the system of record (MySQL, with a PostgreSQL profile available), authenticates users, applies business rules and calls the AI service through a non-blocking WebClient. The *AI service* is stateless with respect to employee data: it holds only face embeddings, in its own SQLite store, and returns recognition results. A Docker Compose descriptor is provided for single-command deployment; the experiments reported here ran the three services directly on a development machine.

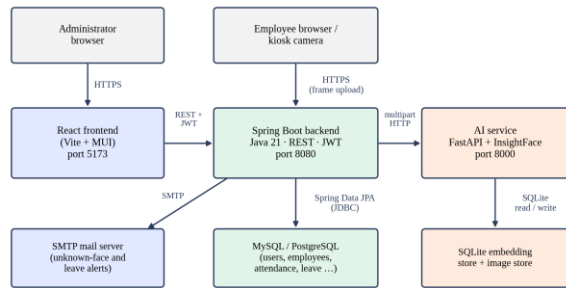


Fig. 1. Three-Service Architecture of AJFace.

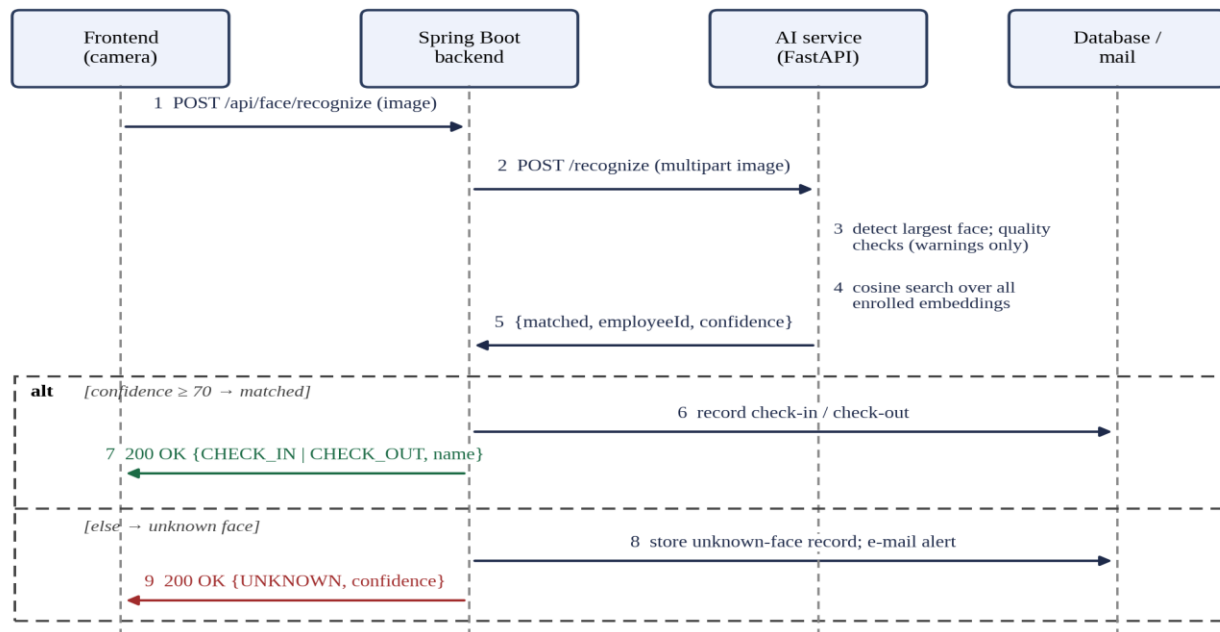


Fig. 2. Sequence of the recognition and attendance flow.

TABLE I Technology stack

Layer	Technology
Frontend	React, Vite, Material UI (port 5173)
Backend	Java 21, Spring Boot 3.3, Spring Security, Spring Data JPA / Hibernate, MySQL 8 (port 8080)
AI service	Python, FastAPI, InsightFace buffalo_l (RetinaFace + ArcFace), FAISS or NumPy, SQLite (port 8000)
Packaging	Docker Compose descriptor; configuration through environment variables

B. Recognition and Attendance Flow

When a face is presented (Fig. 2), the frontend posts the image to the backend, which forwards it as a multipart request to the AI service. The AI service detects the largest face, applies quality checks, searches the embedding store, and returns whether a match was found, the matched employee identifier and a confidence value.

If the confidence meets the accept rule, the backend records a check-in or check-out (or issues tokens, for face login); otherwise, it stores an unknown-face record with the captured image and e-mails administrators. Because one verification both authenticates the person and records attendance, no separate sign-in step is needed.

V. AI-SERVICE ALGORITHMS

A. Detection and Alignment

Images are decoded and passed to the buffalo_l model pack with a 640×640 detection size. RetinaFace returns bounding boxes and five landmarks per face; the landmarks drive a similarity-transform alignment to a canonical pose before recognition. At enrolment, each image must contain exactly one face with a detector score of at least 0.55 and a width of at least 80 pixels; other images are skipped with a warning. At recognition time no such gate is applied and the largest detected face is used.

B. Embedding and Matching

The aligned crop is mapped by ArcFace to a 512-dimensional embedding, which is L2-normalised so that the inner product equals cosine similarity:

$$s(q, e_i) = q \cdot e_i / (\|q\| \|e_i\|) = q \cdot e_i \quad (1)$$

The store performs exact search, using a FAISS IndexFlatIP index [14] when available and a NumPy brute-force dot product otherwise, which is adequate for tens to hundreds of employees. Because an employee may have several embeddings, results are aggregated to the best similarity per employee. A probe is accepted as employee i when $s \geq \tau$, with $\tau = 0.45$ in our configuration.

C. Threshold-Anchored Confidence

Raw cosine similarities for genuine matches cluster in a narrow band above τ , making them unintuitive as a displayed “confidence”. We therefore map similarity onto 0–100 with two linear segments anchored so that $s = \tau$ yields exactly 70:

$$C(s) = 70 \cdot \max(0, s)/\tau, \quad s \leq \tau \quad (2)$$

$$C(s) = 70 + 30 \cdot (s - \tau)/(1 - \tau), \quad s > \tau \quad (3)$$

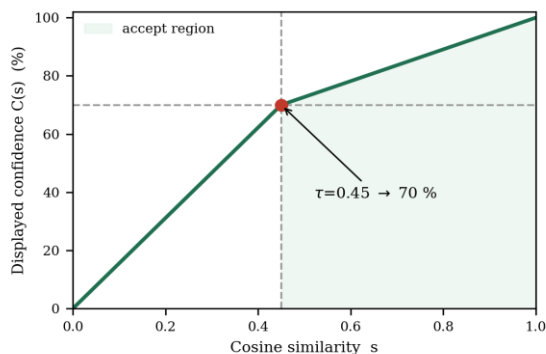


Fig. 3. Threshold-anchored mapping from cosine similarity to displayed confidence.

The mapping is monotonic, so it never changes the ordering of candidates, and the backend independently treats $C \geq 70$ as a match. This is equivalent to $s \geq \tau$, so the two services cannot disagree even though one compares cosine similarity and the other a percentage (Fig. 3, Table II).

TABLE II Similarity-to-confidence mapping (threshold 0.45)

Similarity s	Confidence $C(s)$	Decision
0.225	35.0 %	Reject
0.450	70.0 %	Accept (threshold)
0.725	85.0 %	Accept
1.000	100.0 %	Accept

D. Duplicate-Identity Check

Before an enrolment embedding is stored, it is searched against the embeddings of all *other* employees. If the best such match has $s \geq \tau$, enrolment of that image is rejected with a duplicate-identity warning, preventing one person from holding two records, whether by accident or to create an alias.

E. Quality and Heuristic Liveness Screen

Sharpness is measured as the variance of the Laplacian of the grey-scale face crop; values below 60 are flagged as blurry. A liveness heuristic then combines two statistics: the mean absolute Laplacian response (fine texture energy) and the variance of block-wise standard deviations (local-contrast spread).

Each is scaled to 0–100 and combined as

$$L = 0.6 \cdot T + 0.4 \cdot V \quad (4)$$

where $T = \min(100, 100 \cdot \mu/12)$ is the texture score and $V = \min(100, 100 \cdot \text{Var}(\sigma_b)/400)$ the contrast score, μ being the mean absolute Laplacian response and σ_b the standard deviation of image block b . A score below 35 raises a possible-spoof warning. At enrolment, failing the blur or liveness gate rejects the image; during recognition the checks only produce warnings, so a genuine live attempt is never hard-rejected on quality alone.

The constants are tuning knobs, not derived from a calibration dataset. This heuristic is not a trained anti-spoofing model and can be defeated by a good print or replay; we treat it as an extensible placeholder.

F. Configuration Parameters

All thresholds are externalised as environment-driven settings so that they can be tuned without code changes. Table III lists the values used in this work.

TABLE III Key configuration parameters

Parameter	Value
Model pack / detection size	buffalo_1 / 640×640
Similarity threshold τ	0.45
Confidence at threshold	70 %
Minimum detector score / face width	0.55 / 80 px
Minimum Laplacian variance (blur)	60
Minimum liveness score	35
Access / refresh token lifetime	1 h / 7 days
Late-arrival grace period	10 min

VI. BACKEND SECURITY AND BUSINESS LOGIC

A. Authentication

Passwords are stored as BCrypt hashes. Successful password or face login yields a short-lived access token (one hour) and a refresh token (seven days). Sessions are stateless: a filter validates the bearer token on each request and loads the user from the database, and an explicit entry point returns a JSON 401 for missing or expired tokens so the client can refresh or redirect to login.

B. Layered Access Control

Access is enforced in three layers (Table IV): endpoint rules by role; object-level checks that a non-administrator may read only records linked to their own employee identifier; and the frontend, which hides what a role may not use but is never relied upon for security.

C. An Object-Level Authorisation Defect

While using the prototype with two employee accounts, an employee session displayed another employee's attendance record.

Two causes combined: the shared attendance page requested the company-wide list for every role, and the

per-employee endpoint trusted the employee identifier in the URL.

TABLE IV Access-control summary

Resource	Rule
Authentication endpoints	Public
Employee create/update/delete, departments, face enrolment, unknown-face alerts, reports, dashboard, manual attendance, payroll set-up	Administrator only
Company-wide attendance (today, date, range), correction review	Administrator only
Per-employee attendance, leave and payslip read	Authenticated; caller must be the owner or an administrator

The correction made company-wide queries administrator-only, added an ownership check to per-employee endpoints (the caller's linked employee identifier must equal the requested one, otherwise HTTP 403), and changed the employee view to a personal 30-day history.

The same check had already been applied to the leave and payroll endpoints. We report this defect because object-level authorisation failures are easily introduced when a shared UI is reused across roles.

D. Attendance and Supporting Modules

A check-in after the shift start plus a configurable grace period (default ten minutes) is recorded as late; duplicate check-ins and check-outs on the same day are rejected; check-out computes worked and overtime minutes.

The prototype also provides a leave workflow with e-mail notification, payroll and payslip generation with loss-of-pay calculation, an administrator manual-entry facility, and exportable Excel, CSV and PDF reports (Fig. 4).

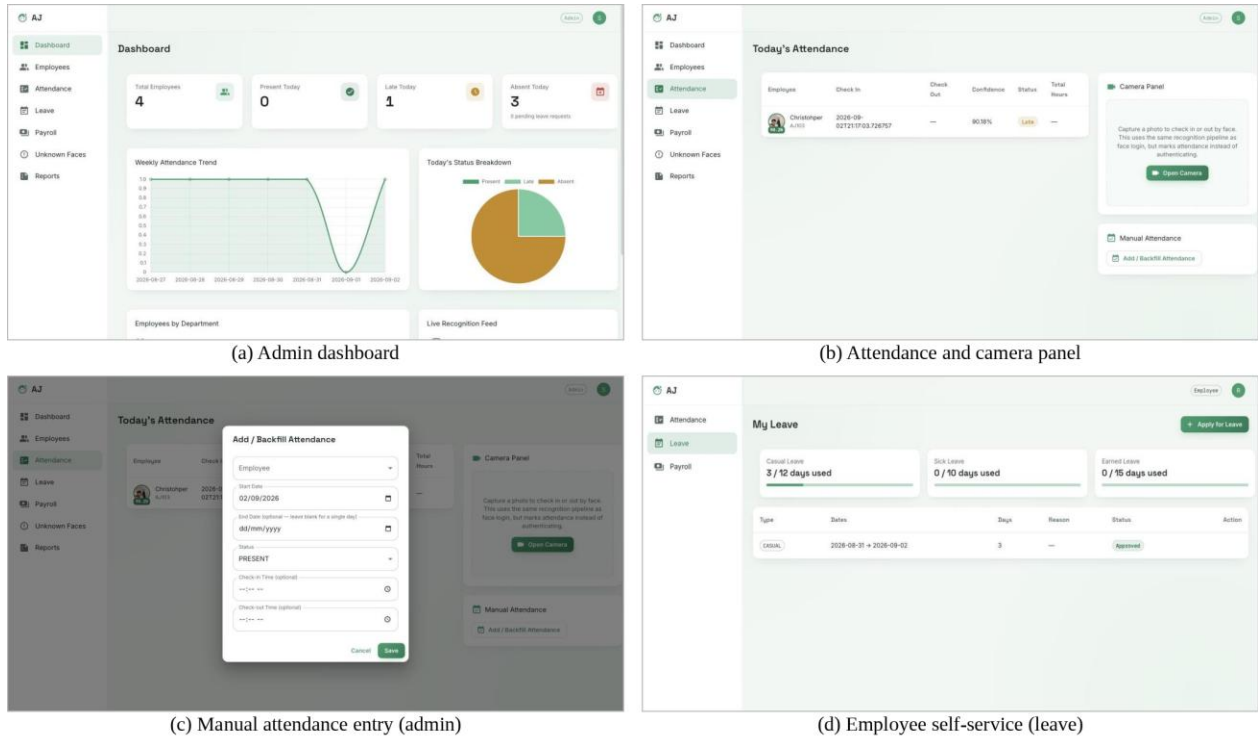


Fig. 4. Prototype screens: (a) dashboard, (b) attendance and camera panel, (c) manual entry, (d) employee self-service.

VII. TESTING AND RESULTS

A. Automated Component Tests

The project-specific logic of the AI service was covered by 22 automated unit tests (Table V), distributed with the source code and run with Python's *unittest* runner. They need neither a camera nor a model download: they use synthetic unit vectors and

synthetic images, and exercise the NumPy search path (FAISS was not installed in the test environment). The pretrained InsightFace detector and recogniser are third-party models and are not re-evaluated here. A central check confirms that the backend rule ($C \geq 70$) and the AI-service rule ($s \geq \tau$) are equivalent over 20,003 similarity values, including values within 10^{-4} of τ . All 22 tests pass.

TABLE V Automated component tests of the AI service

Test group	Tests	Result
Cosine similarity: identical, orthogonal and scaled vectors	3	Pass
Confidence mapping: anchor, end points, tabulated values, monotonicity, rule equivalence	5	Pass
Embedding store: exact, noisy and unrelated probes; per-employee aggregation; duplicate-identity search; own-embedding exclusion; removal; reload from disk	8	Pass
Quality and liveness: blur ordering, enrolment gates, warnings-only recognition mode, score range, empty input	6	Pass
Total	22	Pass

B. Functional Walkthrough of the Prototype

The backend and frontend do not yet have an automated test suite. Their behaviour was instead

checked by exercising the running prototype end to end, as summarised in Table VI. These observations are qualitative and are not a controlled experiment.

TABLE VI Functional walkthrough of the running prototype

Capability	Observed outcome
Face enrolment	Employee list shows each employee as enrolled or not enrolled
Face check-in	Enrolled employee recognised and attendance recorded with a confidence of about 90–91 % and a present or late status (Fig. 4b)
Unknown face	Unmatched face stored as an alert with timestamp and camera location
Role-based portals	Administrators see dashboard, employees, payroll, reports and alerts; employees see only attendance, leave and payroll (Fig. 4a, 4d)
Leave workflow	Employee applies; administrator approves; leave balance and status update (Fig. 4d)
Manual entry	Administrator backfills attendance for a date or range (Fig. 4c)
Payroll	Annual CTC set per employee; monthly payslips generated with loss-of-pay
Reports	Attendance, leave and unknown-face reports exported as Excel, CSV or PDF

C. Recognition Accuracy and Latency

We did not carry out an independent, large-scale evaluation of false-acceptance and false-rejection rates, which requires many identities and many genuine and impostor pairs, nor a formal latency or load test.

The *published* figures for the underlying models (for example ArcFace above 99.8% on LFW [4]) characterise the pretrained models, not this deployment. The confidence values of roughly 90–91% displayed for genuine check-ins in the prototype (Fig. 4b) are illustrative only, because the number of enrolled identities is far too small to establish statistical performance, and we make no such claim.

VIII. DISCUSSION AND LIMITATIONS

The main benefit of the design is separation of concerns: the inference model can be replaced, scaled or moved to a GPU host without changing business logic, and failures of the AI service are reported explicitly (client error, unreachable or timed out) under bounded timeouts rather than as hanging requests. The confidence anchoring removes a class of

integration bug in which two services apply subtly different thresholds. Finally, using one verification event for both authentication and attendance removes redundancy present in conventional login-plus-punch workflows.

The limitations are as follows. (i) Liveness detection is a heuristic, was not evaluated against presentation attacks, and is not expected to resist high-quality print or replay attacks. (ii) Stored embeddings are protected conventionally rather than by cancellable or homomorphic template protection. (iii) No formal FAR/FRR evaluation was performed and the enrolled population was small. (iv) No formal latency or load benchmark was performed, and the backend and frontend have no automated test suite yet. (v) The component tests and the walkthrough were designed by the developers and do not cover extreme lighting, face coverings or the ageing of enrolment photographs. Results should be read with these threats to validity in mind.

IX. CONCLUSION AND FUTURE WORK

This paper presented AJFace, a microservice-based face-recognition e-authentication system built with Spring Boot, FastAPI and InsightFace and validated through an enterprise attendance case study. The design decouples recognition from business logic, keeps the accept/reject rule consistent across services through threshold-anchored confidence, prevents duplicate identities, logs every attempt, and enforces role-based and object-level access control. The 22 automated component tests pass and the prototype walkthrough confirmed the main workflows; robustness to spoofing, recognition accuracy and latency remain unevaluated.

Future work includes (1) adding automated backend and end-to-end tests, and replacing the heuristic with a trained passive-liveness model evaluated on standard presentation-attack benchmarks; (2) adopting biometric template protection for stored embeddings; (3) a formal evaluation reporting FAR, FRR, precision, recall and F1 over a properly constructed dataset; (4) load testing and GPU inference; and (5) multi-factor fallback, for example face plus one-time password, for borderline matches.

DECLARATION ON USE OF GENERATIVE AI

Generative AI tools (Anthropic's Claude) were used to assist with software development, drafting, language editing and formatting of this work. The authors reviewed, verified and edited all content and take full responsibility for the manuscript. No AI tool is listed as an author.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Systems Engineering, Andhra University College of Engineering (A), for the resources and guidance that made this work possible.

REFERENCES

- [1] M. A. Turk and A. P. Pentland, "Eigenfaces for recognition," *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.
- [2] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, Jul. 1997.
- [3] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823.
- [4] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4690–4699.
- [5] J. Deng, J. Guo, E. Ververas, I. Kotsia, and S. Zafeiriou, "RetinaFace: Single-shot multi-level face localisation in the wild," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 5203–5212.
- [6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. I-511–I-518.
- [7] J. Lewis and M. Fowler, "Microservices: A definition of this new architectural term," *martinfowler.com*, 2014. OnlineOnline.

Available:

<https://martinfowler.com/articles/microservices.html>

- [8] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," *IETF RFC 7519*, May 2015.
- [9] S. S. Pawaskar and A. M. Chavan, "Face recognition-based class management and attendance system," in *Proc. IEEE Bombay Section Signature Conference (IBSSC)*, 2020, doi: 10.1109/IBSSC51096.2020.9332212.
- [10] Z. Yu, Y. Qin, X. Li, C. Zhao, Z. Lei, and G. Zhao, "Deep learning for face anti-spoofing: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5609–5631, May 2023, doi: 10.1109/TPAMI.2022.3215850.
- [11] V. Krivokuća Hahn and S. Marcel, "Biometric template protection for neural-network-based face recognition systems: A survey of methods and evaluation techniques," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 639–666, 2023, doi: 10.1109/TIFS.2022.3228494.
- [12] InsightFace, "2D and 3D face analysis project." OnlineOnline. Available: <https://github.com/deepinsight/insightface>
- [13] OWASP Foundation, "OWASP Top Ten." OnlineOnline. Available: <https://owasp.org/www-project-top-ten/>
- [14] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," *arXiv:1702.08734*, 2017.