An ESP32-CAM Face Recognition Prototype with Course Enrolment and Blink Liveness Checks to Address Proxy Attendance in University Lectures

Luqman Hakim Mohd Nazir, Muhamad Sani Mustafa*, Zainuddin Mat Isa & Zulkifli Husin

Faculty of Electronic Engineering & Technology, Universiti Malaysia Perlis, Arau, Perlis, Malaysia

***Corresponding Author**

DOI: https://doi.org/10.47772/IJRISS.2026.101000421

Received: 18 October 2026; Accepted: 23 October 2026; Published: 04 November 2026

ABSTRACT

Class attendance is closely linked to students' academic performance, and many universities use attendance records to decide examination eligibility. Manual sign-in sheets, shared QR codes and online forms, however, still allow students to mark attendance for absent classmates. Face recognition confirms who is present, but a plain recogniser can be fooled by a photograph, and recognition alone does not show that the student belongs to the course session. This study aimed to design and functionally validate EZAttendance, a lecture attendance system that records a student as present only after three checks are passed in sequence. An ESP32-CAM module streams video over Wi-Fi to a Python server, which detects faces with a histogram of oriented gradients (HOG) detector and compares 128-dimensional face encodings with those of registered students using a Euclidean distance tolerance of 0.55. The identified student is then checked against the course enrolment list in Cloud Firestore, and liveness is confirmed by an eye blink, detected when the eye aspect ratio falls below 0.19 and returns above 0.26 within 3.0 s. An Android application lets lecturers create courses, run sessions, generate reports and send absence reminders, while students enrol their faces and view their own records. In functional tests with two registered student volunteers, the system marked a blinking enrolled student as present, refused a registered student who was not enrolled in the course, and labelled a photograph shown on a phone screen as fake without writing any record. Attendance updates appeared in the mobile application through Firestore synchronisation. These results show that the three checks can be combined on an ESP32-CAM and a laptop in a small functional test, but they do not yet show how reliable the system is in a real classroom. The study sets out the quantitative evaluation needed before wider deployment.

Keywords: proxy attendance, face recognition, liveness detection, higher education, Internet of Things

INTRODUCTION

Lecture attendance is more than an administrative routine. A meta-analysis of 69 studies found that class attendance correlates with class grades at $\rho = .44$, making it a stronger predictor of college grades than admission test scores or study habits (Credé et al., 2010). Sensor-based tracking of nearly 1,000 undergraduates later showed the same link between attendance and performance (Kassarnig et al., 2017). Some universities compute examination eligibility directly from attendance records (Shamila, 2025), which makes the honesty of the attendance record a matter of fairness as well as record-keeping.

The common methods do not protect that record well. Roll-call and sign-in sheets take class time and are easy to sign on behalf of an absent friend. QR-code systems speed up the process (Masalha & Hirzallah, 2014; Shamila, 2025), but a static code can be photographed and forwarded, and an online form link can be shared in a group chat. Later QR designs bind the scan to a registered device, an IMEI number, a geofence or a GPS

position (Nwabuwe et al., 2023; Kanapathy et al., 2026). These checks confirm where a phone is, not who is holding it; a student can still hand the phone to a classmate.

Face recognition answers the question of who is present, and it has been used for classroom attendance for more than a decade (Chintalapati & Raghunadh, 2013). Lower-cost versions use the ESP32-CAM module as a networked camera (Das & Halder, 2024), while the recognition itself usually runs in Python with OpenCV or dlib on a nearby computer (Makigod et al., 2024; Gaurav, 2026). Two weaknesses remain. First, a plain 2D recogniser can be fooled by a printed photograph or a face shown on a phone screen, a presentation attack that is well documented in the biometrics literature (Ramachandra & Busch, 2017; Yu et al., 2023). Second, recognising a registered face does not show that the person belongs in that particular class session.

Recent attendance systems have added liveness detection to deal with the first weakness. Surantha and Sugijakko (2024) ran a MobileNetV2 liveness model on a Raspberry Pi; Potdar et al. (2022) combined blink detection with a CNN classifier; and Hosen et al. (2023) used difference-of-Gaussian filtering. Several designs use the eye aspect ratio (EAR) of Soukupová and Čech (2016) on dlib face encodings, either with an RFID card on an ESP32-CAM (Sapna, 2025) or with a webcam and a desktop computer (Deephika et al., 2026; Gaurav, 2026). The second weakness has been addressed by binding face recognition to the course. Sunaryono et al. (2021) required students to capture a course QR code together with their face image, Abdylhalyk et al. (2026) added QR-based session control and geolocation, and Muzammil et al. (2026) validated attendance against the timetable. These designs run on students' phones, web cameras or desktop software. Few systems built around an ESP32-CAM camera node check course enrolment before liveness, and few describe how the result feeds the lecturer's follow-up of absent students.

This paper reports the design and functional validation of EZAttendance, a lecture attendance system built on an ESP32-CAM, a Python recognition server, Google Firebase and an Android application. It applies a three-check rule in a fixed order: a student is marked present only when (i) the live face matches a registered encoding, (ii) the matched student is enrolled in the active course, and (iii) a natural blink confirms that a live person is in front of the camera. The contribution lies in implementing this rule with an ESP32-CAM camera node and linking it to a two-sided mobile workflow, in which lecturers run sessions, view cumulative reports and send absence reminders, and students see their own records. The remainder of the paper describes the methodology, presents and discusses the test results, and closes with the conclusions and the evaluation work still required.

METHODOLOGY

The study followed a design-and-development approach with functional validation. Three objectives guided the work: to identify students by face, to block proxy attendance, and to store and share attendance records in real time. The subsections below describe the system architecture, the operating workflow, the three verification checks in the order they run, the data security arrangements and the test procedure.

System architecture

EZAttendance has four parts (Figure 1). An ESP32-CAM module with an OV2640 2-MP sensor captures video and is intended to be placed at the lecture hall entrance. It sits on an ESP32-CAM-MB programmer board, is powered at 5 V through a micro-USB cable from the lecturer's laptop, and is housed in a 3D-printed case. Its firmware, written in the Arduino IDE from the CameraWebServer example, joins the local Wi-Fi network and serves a Motion-JPEG video stream over HTTP using the esp_http_server library.

Because the ESP32-CAM lacks the memory and processing power for face recognition, all recognition runs on a Python server on the lecturer's laptop, using OpenCV, NumPy, dlib (King, 2009) and the firebase-admin SDK. Google Firebase provides authentication, the Cloud Firestore database and Cloud Storage. An Android application written in Java (Android Studio) serves students and lecturers, and a desktop launcher (EZAttendance Software) starts the Python scripts when the lecturer starts a class session from the application.

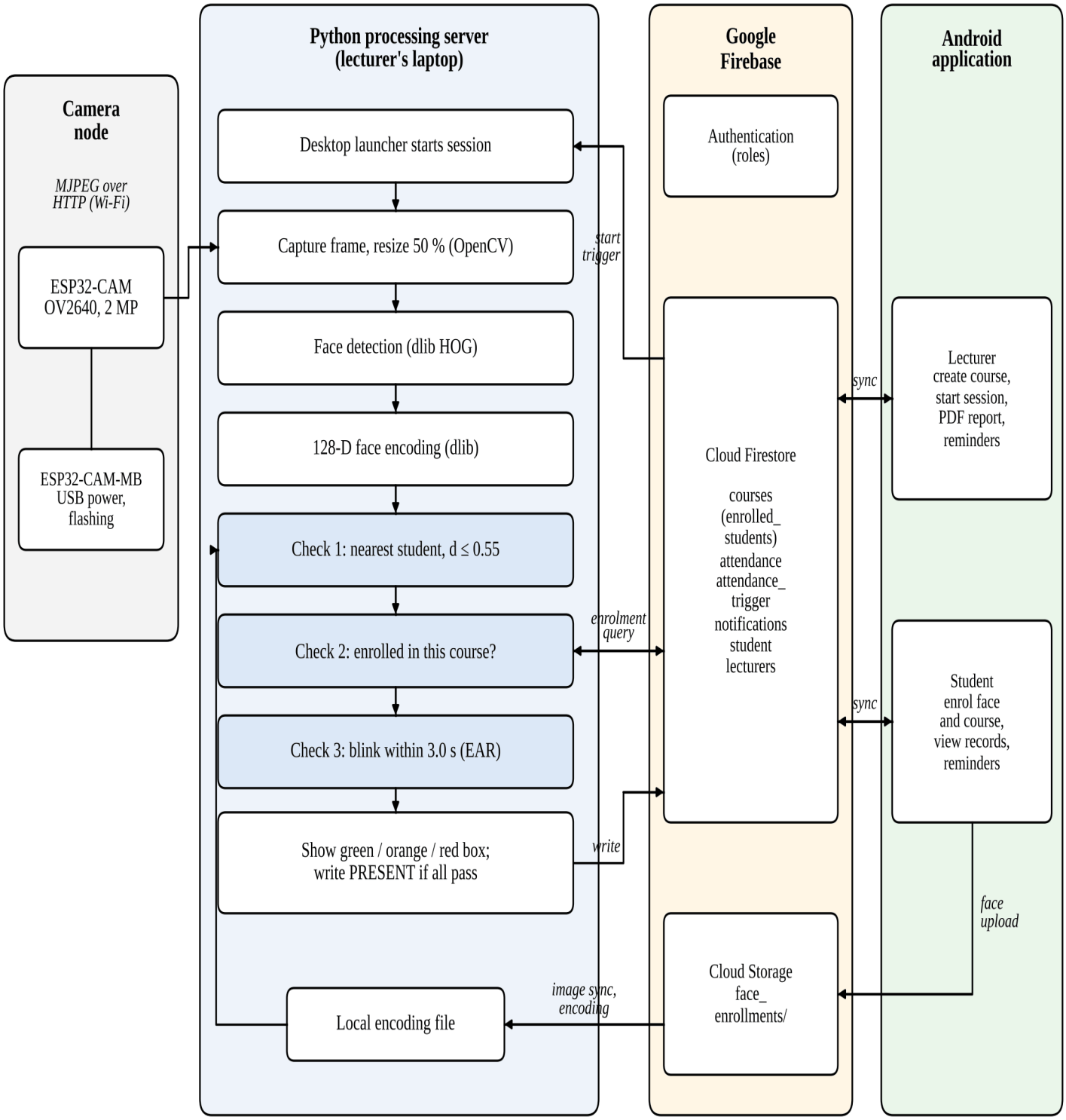


Figure 1. Block diagram of EZAttendance: the ESP32-CAM camera node, the Python processing server with the three checks, Google Firebase services and the Android application

Operating workflow

The system runs in three phases. In the pre-class phase, a lecturer registers and creates a course with its code, name, day, venue and start and end times. A student registers and captures a face image through the application, which checks that the face is clear before upload, and then enrols in the course. The image is stored in Cloud Storage under the folder face_enrollments/ and named by the student's matric number. In the live-session phase, the lecturer selects the teaching week (1 to 14) and starts the session, which launches the camera stream and the recognition server; this phase is shown in Figure 2. In the post-class phase, the lecturer reviews weekly and cumulative attendance, exports a PDF report and sends reminders to absent students, which appear on each student's notification board.

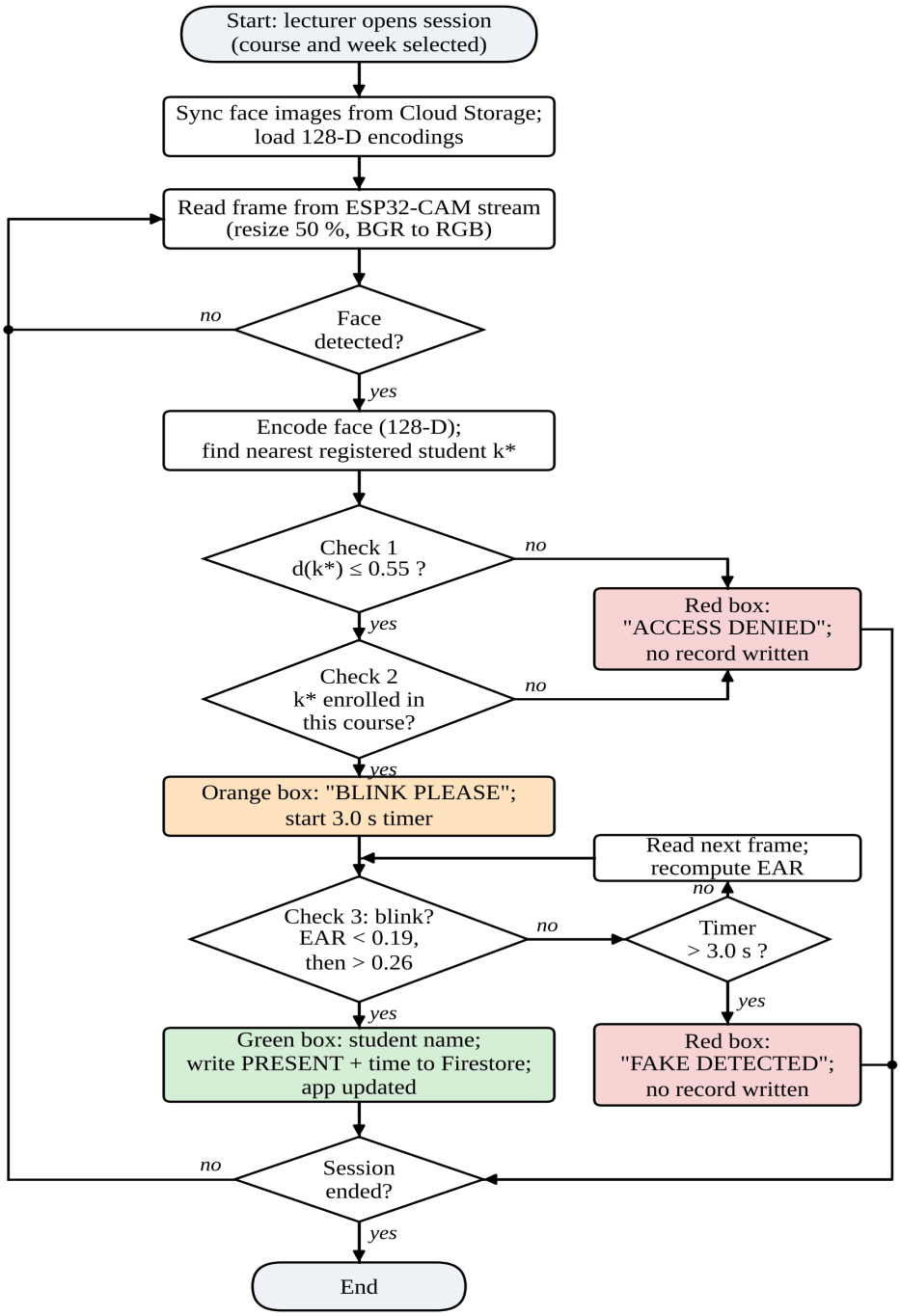


Figure 2. Live-session flowchart showing the three checks in order: face match, course enrolment and blink-based liveness

Check 1: face encoding and identification

At session start, a synchronisation script downloads the registered face images from Cloud Storage, converts the raw bytes into an 8-bit unsigned integer array, decodes it with OpenCV and converts it to RGB. Each face is encoded as a 128-dimensional vector with the face recognition model distributed with the dlib toolkit (King, 2009), a network derived from ResNet-34 (He et al., 2016). The encodings are saved to a local encoding file and loaded into memory for the session, and the downloaded images are kept as local copies on the laptop.

During the session, each frame from the stream is resized to 50 % of its original size to cut computation and converted from BGR to RGB. Faces are located with dlib's HOG detector (Dalal & Triggs, 2005) and encoded in the same way. The live encoding f is compared with every registered encoding f_k by Euclidean distance, as in Eq. (1).

$$d_k = \|\mathbf{f} - \mathbf{f}_k\|_2, \qquad k^* = \arg\min_k d_k \tag{1}$$

The closest student k∗ is accepted as a match only if d(k∗) ≤ τ, with the tolerance τ set to 0.55. A face with no registered encoding within this tolerance is refused and shown "ACCESS DENIED" in a red box. The tolerance was set empirically during development. It is stricter than the distance of 0.6 at which the dlib model reaches 99.38% accuracy on the Labeled Faces in the Wild benchmark (King, n.d.), so a false match is less likely at the cost of more false rejections. It was not calibrated on recorded data from this study.

Check 2: course enrolment

A matched student is next checked against the enrolment list of the active course, held in the enrolled_students sub-collection of that course in Cloud Firestore. A registered student who is not enrolled in the course is refused at this point with a red denial label, and the attendance decision does not proceed to the liveness check.

Check 3: blink-based liveness

A photograph or screen image of an enrolled student can pass the first two checks, so the third check requires a natural blink. dlib's 68-point landmark predictor (Kazemi & Sullivan, 2014) locates six points p_1 to p_6 around each eye, and the eye aspect ratio is computed as in Eq. (2) (Soukupová & Čech, 2016).

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\,\|p_1 - p_4\|} \tag{2}$$

When an enrolled student is identified, the system draws an orange box with the prompt "BLINK PLEASE" and starts a 3.0 s timer; EAR is recomputed on each new frame while the timer runs. A blink is accepted when EAR drops below 0.19 (eyes closed) and then rises above 0.26 (eyes open); the gap between the two thresholds is intended to stop small fluctuations from being counted as blinks. Both values were chosen empirically while testing with the volunteers. They lie close to the threshold of 0.18 reported as optimal for a dashboard-camera dataset (Dewi, Chen, Chang, et al., 2022), but the best threshold differs between datasets and individuals (Dewi, Chen, Jiang, & Yu, 2022), so the values used here are working settings rather than calibrated ones. If no blink is seen before the timer expires, the face is classified as a presentation attack and shown "FAKE DETECTED" in red. Only when all three checks pass does the system draw a green box with the student's name and write the status PRESENT, with a timestamp, to Firestore, from where it is synchronised to the mobile application. Table 1 lists the settings used.

Table 1. System settings and tools

Item	Setting / tool
Camera module	ESP32-CAM (ESP32-S, OV2640 2 MP) on ESP32-CAM-MB (CH340 USB-to-serial)
Power	5 V via micro-USB from the lecturer's laptop
Video stream	Motion-JPEG over HTTP on local Wi-Fi (esp_http_server)
Firmware	Arduino IDE, ESP32 board package (Espressif), CameraWebServer example
Processing server	Python 3.12 with OpenCV, NumPy, dlib and firebase-admin on a Windows laptop
Frame pre-processing	Resize to 50 %; BGR to RGB conversion
Face detector	dlib HOG detector
Face encoder	dlib ResNet-34-derived network, 128-D output
Check 1: match rule	Nearest registered student by Euclidean distance; tolerance 0.55
Check 2: enrolment	enrolled_students sub-collection of the active course (Cloud Firestore)

Check 3: liveness	dlib 68-point landmarks; EAR < 0.19 then EAR > 0.26 within 3.0 s
Cloud services	Firebase Authentication, Cloud Firestore, Cloud Storage
Mobile application	Android (Java, Android Studio)
Semester structure	14 teaching weeks per course

Data access and security

Firebase Authentication handles log-in for both roles. Firestore security rules check the identity of every read and write request: users may update only their own profile (request.auth.uid == userId), lecturers may create courses, and students have read-only access to course details. Face images are kept in Cloud Storage, and each student document stores a reference to the image (imageUrl) rather than the image itself. During a session, the downloaded images and the local encoding file sit on the lecturer's laptop; the project record describes no separate encryption for these local files.

Test procedure

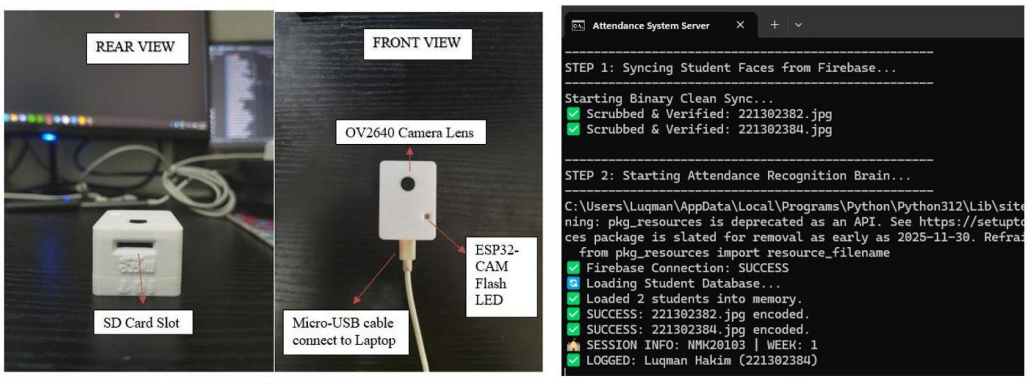
Two students volunteered for the tests. Both registered in the system and enrolled their faces, and one of them was the first author. The two students have similar names and were registered as "luqman" (221302382) and "Luqman Hakim" (221302384), which explains the two name forms in Figures 3 to 5. Four scenarios were run: (1) an enrolled student who blinks; (2) an enrolled student's photograph displayed on a phone screen; (3) a registered student who is not enrolled in the active course; and (4) the end-to-end flow from registration and course enrolment to report generation and absence reminders in the application. The outcome of each scenario was judged by the colour and message of the on-screen box, by whether a record was written to Firestore, and by whether the change appeared in the mobile application. The tests were functional: test location, lighting and camera distance were not recorded, and neither the number of repeated trials nor timing measurements were logged.

RESULTS AND DISCUSSION

In each of the four demonstrated scenarios, the prototype behaved as designed: attendance was written only for an enrolled student who blinked, while a phone-screen photograph and a registered but non-enrolled student were both refused. The subsections below report the session start-up, the classification outcomes and the application functions, and then place the results against related work.

Session start-up and streaming

The assembled camera unit is shown in Figure 3(a). When the lecturer started a session, the server console (Figure 3(b)) showed that the face images of the two registered students had been downloaded from Cloud Storage and encoded ("Loaded 2 students into memory"), that the course and week had been loaded ("SESSION INFO: NMK20103 | WEEK: 1"), and that attendance had been logged for one of them ("LOGGED: Luqman Hakim (221302384)"). The ESP32-CAM stream then opened on the lecturer's screen.



(a) (b)

Figure 3. (a) ESP32-CAM camera unit in its 3D-printed case; (b) server console after session start-up

Classification outcomes

Table 2 lists the four states the system produced, and Figure 4 shows them on the lecturer's screen. When an enrolled student stood in front of the camera, Checks 1 and 2 passed and attendance was held under an orange "BLINK PLEASE" box (Figure 4(a)). When a photograph of an enrolled student was shown on a phone screen, the face also passed Checks 1 and 2 and was held in the orange state (Figure 4(b)); because the photograph could not blink, the 3.0 s timer expired and the box turned red with "FAKE DETECTED" (Figure 4(c)). No record reached Firestore. When the enrolled student blinked, the blink rule was met and the box turned green with the student's name (Figure 4(d)). In the same frame, a registered student who was not enrolled in the course was refused at Check 2 with a red "DENIED: …" label, cut off at the frame edge, while the enrolled student was accepted (Figure 4(e)). Eye landmarks are also drawn on the refused face because the landmark overlay is applied to every detected face; the attendance decision itself follows the order of Checks 1 to 3.

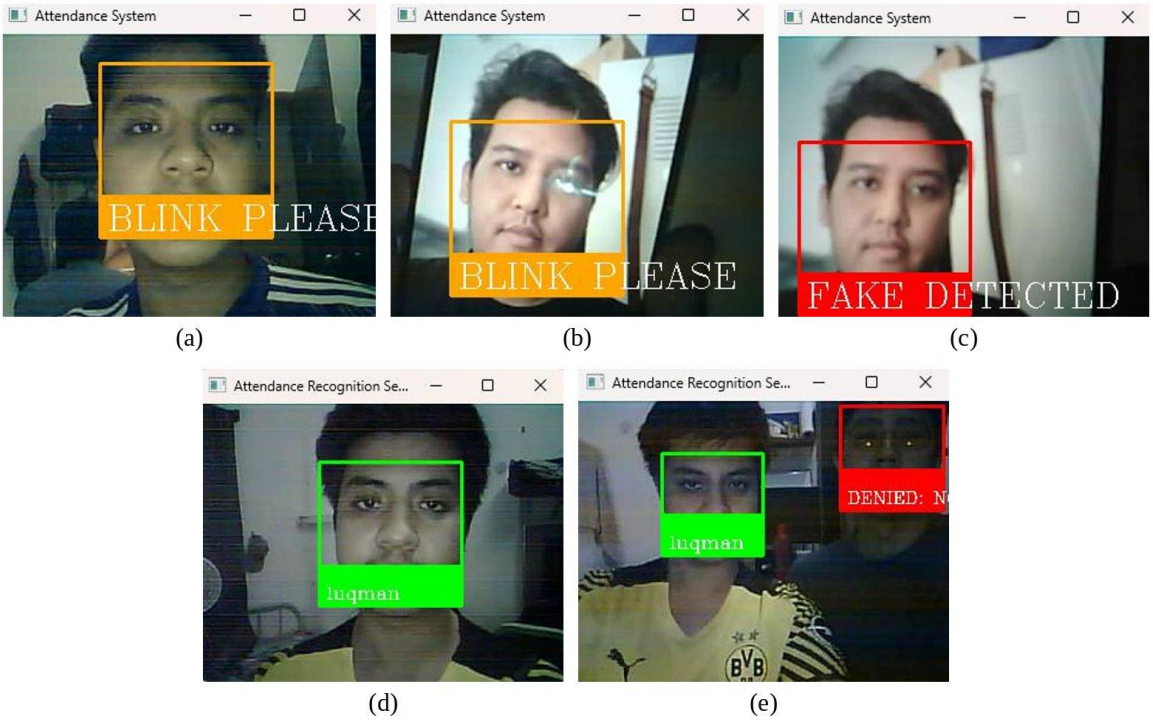


Figure 4. Lecturer-side display: (a) liveness check; (b) phone-screen photograph held for liveness; (c) photograph rejected as fake; (d) enrolled student accepted; (e) enrolled student accepted and non-enrolled student denied

Table 2. Observed classification outcomes

Scenario	Box colour	Prompt	Condition	Outcome
Liveness check	Orange	BLINK PLEASE	Checks 1 and 2 passed; no blink yet	Attendance held
Presentation attack	Red	FAKE DETECTED	Photograph on phone screen; no blink within 3.0 s	Refused; nothing written
Not enrolled	Red	DENIED: …	Registered face; ID not in course enrolment list (Check 2)	Refused; no PRESENT record by design
Valid attendance	Green	Student name	Enrolled student blinks within 3.0 s	PRESENT written to Firestore and shown in app

These outcomes come from the system's own display and, for the spoof case, from the absence of a Firestore record. Only two registered faces were used and no repeated-trial statistics were recorded, so the table shows that each decision path works, not how often it works.

Application and database functions

In the Android application, students registered, enrolled their faces, enrolled in courses, viewed daily and weekly timetables and checked a 14-week attendance record for each course. When a session starts, the lecturer's screen shows the current week, a counter of students present and a live feedback panel that lists each student as they are verified; Figure 5(a) shows this screen at the start of a Week 2 session, before any student had been detected. After the session, weekly records listed each attendee with the scan time (Figure 5(b)). The cumulative view gave each student's attendance from Week 1 to the selected week; for example, one student attended Week 1 but not Week 2, so the view showed 100 % at Week 1 and 50 % at Week 2 (Figure 5(c)). The lecturer could export this summary as a PDF report. For a week with no absences, the reminder screen returned the message "All students attended this week!" (Figure 5(d)); for a week with absences, it listed the absent students with a REMIND button, and the reminder appeared on the student's notification board; Figure 5(e) shows such a reminder for an absence in Week 3. In the Firebase console, records were stored in collections including attendance, attendance_trigger, courses (with an enrolled_students sub-collection), lecturers, notifications and student, and face images were held in face_enrollments/ named by matric number.

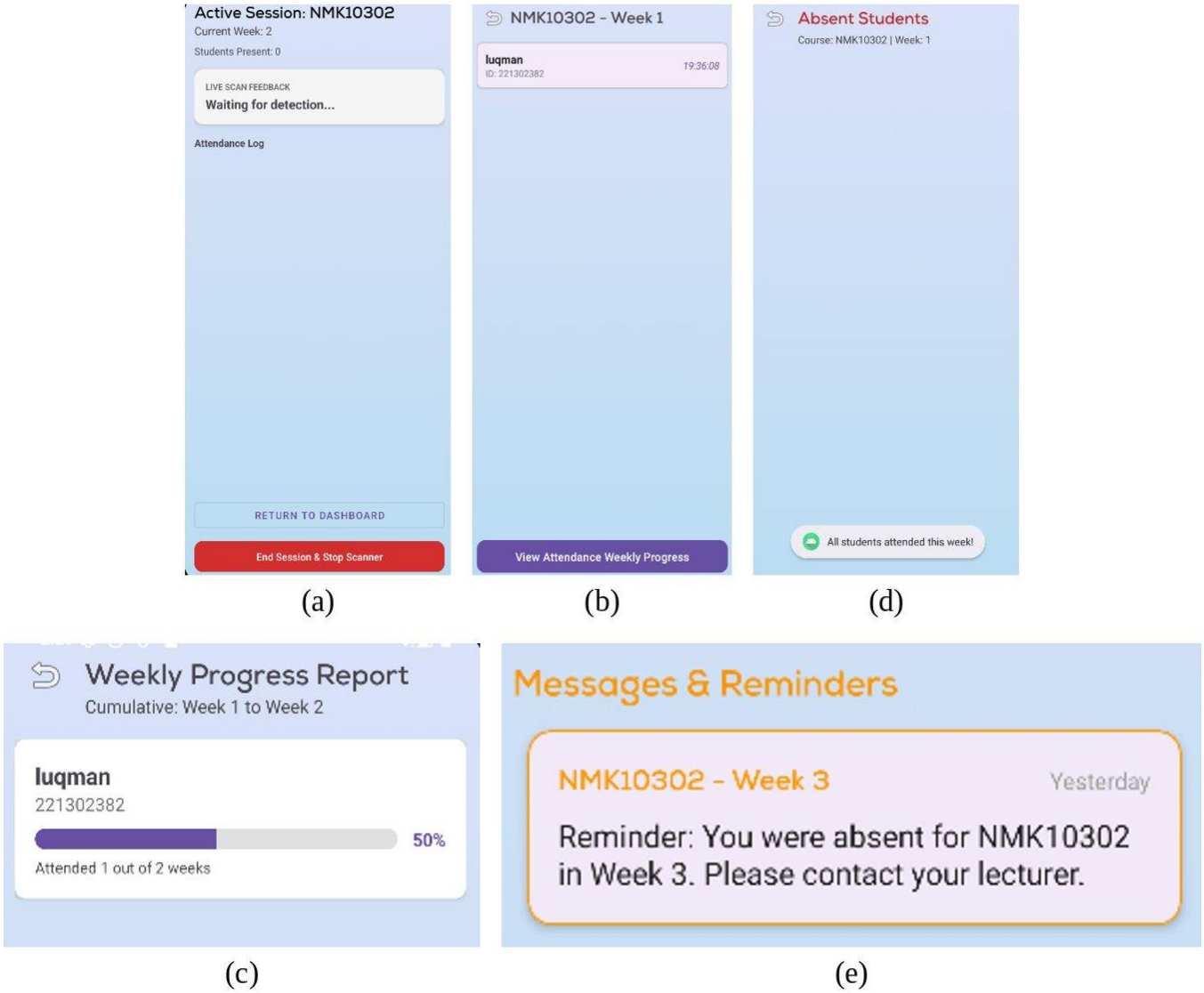


Figure 5. Mobile application screens: (a) session start, before detection; (b) weekly attendance detail; (c) cumulative progress; (d) week with no absences; (e) student notification board with an absence reminder

Comparison with related work

Table 3 compares EZAttendance with recent attendance systems that target proxy attendance.

Table 3. Comparison with related attendance systems

Study	Identity check	Liveness check	Course / session check	Platform	Reported evaluation
Sunaryono et al. (2021)	Face image (LDA) sent from student phone	None	Course QR code displayed in class	Android app + server	97.29 % recognition accuracy
Kanapathy et al. (2026)	QR scan on student phone	None	Dynamic QR and GPS within classroom	Mobile app, cloud	Reduced proxy attendance reported
Hosen et al. (2023)	Haar + LBPH	Difference-of-Gaussian	Not reported	Camera + PC software	87 % recognition, 15 % false positive
Surantha & Sugijakko (2024)	Lightweight face model	MobileNetV2	Not reported	Raspberry Pi	96 % live; 70–83.7 % across spoof levels; < 0.6 s
Sapna (2025)	RFID card + ResNet-34 encodings	EAR blink over frame burst	Not reported	ESP32 + ESP32-CAM, server	0 % FAR; 95 % static photos rejected
Abdylhalyk et al. (2026)	SCRFD + ArcFace	Liveness-related checks	QR session control and geolocation	Web application	72 students; 94.2 % accuracy; formal spoof tests pending
Deephika et al. (2026)	dlib 128-D + neural classifier	EAR blink	Not reported	Webcam + PC	Not reported
Muzammil et al. (2026)	face_recognition library	None	Timetable validation	Python, Streamlit, web camera	No accuracy measured (stated)
Gaurav (2026)	face_recognition 128-D	EAR blink	Duplicate-entry lock only	Python, Streamlit, PC	50 subjects; 95.8 % accuracy; 40 of 40 photo spoofs blocked; 0.8 s
This work	dlib 128-D, distance ≤ 0.55	EAR blink, 3.0 s timeout	Firestore course enrolment list	ESP32-CAM, laptop, Firebase, Android	Functional validation of all decision paths

Two points stand out. First, the QR-based systems verify a phone rather than a person; EZAttendance verifies the face itself, so lending a phone or forwarding a link is not enough on its own, although a replayed video could still defeat the blink test (see Limitations). Second, the closest designs are Sapna (2025), Deephika et al. (2026) and Gaurav (2026), which also pair dlib-style 128-D encodings with an EAR blink test. Course binding itself is not new (Sunaryono et al., 2021; Muzammil et al., 2026). EZAttendance differs in using the ESP32-CAM as a fixed networked camera rather than the student's own phone, in checking the matched student against the course enrolment list before liveness is tested, and in returning results to a two-sided mobile workflow with reports and reminders. Its clear weakness against these studies is the lack of measured rates: Gaurav (2026), for example, reports recognition accuracy over 50 subjects and the rejection of 40 photo attacks, whereas the present tests show only that each decision path works.

What the results add

The enrolment check covers a case that recognition alone misses: a genuine, live, registered student who is in the wrong class session. In the demonstrated case, the system refused such a student at Check 2, which keeps the record tied to the course rather than only to a registered identity. For lecturers, the cumulative view and one-click reminders are intended to turn attendance from a sheet to be collected into information that prompts

early follow-up with students whose attendance is falling; whether lecturers use them in this way remains to be evaluated.

LIMITATIONS

The evaluation was functional only, with two registered students and no record of the number of trials, test location, lighting or camera distance, so recognition rate, false acceptance and false rejection rates, spoof-rejection rate and latency remain unmeasured. The results therefore show that each decision path of the prototype works; they do not establish its reliability in a classroom or its readiness for deployment. A blink test can be beaten by replaying a video of the student, a known weakness of motion-based liveness cues (Ramachandra & Busch, 2017); for example, a short clip of an enrolled student blinking, played on a phone, would be expected to pass Check 3, and this attack was not tested. The tolerance and EAR thresholds were not tuned on recorded data, although studies show that the best EAR threshold varies with the dataset (Dewi, Chen, Jiang, & Yu, 2022). The developer noted that recognition needs a well-lit area and a student who pauses in front of the camera, but illumination, camera distance, student movement and the make-up of the student group were not varied, so performance under these conditions is unknown. The ESP32-CAM may also overheat during long continuous streams, causing frame drops or resets, although this was not observed in the short tests. The system also depends on stable Wi-Fi and on the lecturer's laptop, and the application currently runs only on Android.

A follow-up evaluation should report, for a defined group of students over repeated sessions and under different lighting, distances and movement: the false acceptance and false rejection rates of Check 1 over a range of tolerances; the attack presentation classification error rate (APCER) and bona fide presentation classification error rate (BPCER) of Check 3 for printed photographs, screen images and replayed videos, following ISO/IEC 30107-3 (International Organization for Standardization, 2023); and the time from a student's arrival in front of the camera to the PRESENT record. Recorded EAR traces from these sessions would also allow the two thresholds to be calibrated. To resist replay, Check 3 could be replaced by a randomised challenge-response, such as asking the student to blink twice or turn the head towards a side chosen by the system, an approach shown to reject photograph and video attacks in earlier work (Singh et al., 2014).

Implications for practice and policy

Facial images are personal data under Malaysia's Personal Data Protection Act 2010, and their use in classrooms raises questions of consent, purpose and surveillance. In the prototype, face images are stored in Google Firebase Cloud Storage, a third-party cloud service, and local copies are kept on the lecturer's laptop during sessions. Student records hold an image reference rather than the image, security rules restrict each user to their own profile, and the recognition step itself runs on the laptop rather than through an external recognition service.

The tested system does not yet include a consent step within the application, a retention period for face images and attendance logs, or a tested manual route for marking attendance; the project record states that lecturers can modify attendance logs, but this function was not part of the tests. These are therefore requirements for any deployment rather than features of the prototype. Four points follow for institutions that consider such a system. First, consent: students should give written, informed consent at face enrolment, stating the purpose, the storage locations and who can access the images, and should be able to withdraw it. Second, alternative verification: students who decline face enrolment, and students wrongly refused by the system, need a manual route, such as verification by the lecturer, with the reason recorded. Third, retention: retention periods should be set for face images, encodings and attendance logs, with images and encodings removed from cloud storage and laptops when a course ends or a student withdraws consent. Fourth, protection: local copies on the laptop should be encrypted or deleted at the end of each session, and access to cloud storage limited to the recognition server and the image owner. Because each record is tied to course enrolment, attendance rules for examination eligibility can then be applied per course with an audit trail.

CONCLUSION

This study designed and functionally validated EZAttendance, a lecture attendance system that records a student as present only after a face match, a course-enrolment check and a blink-based liveness check, applied in that order. Built from an ESP32-CAM, a Python recognition server, Firebase and an Android application, and tested with two registered students, the prototype wrote attendance for an enrolled student who blinked and refused both a phone-screen photograph and a registered student not enrolled in the course. Results were reflected in the mobile application for both students and lecturers, together with cumulative reports and absence reminders.

The main contribution is an ESP32-CAM-based implementation of identity, enrolment and liveness checks, applied in that order and linked to the lecturer's follow-up of absent students. Its main limitation is that performance was not measured, so the results should not be read as evidence of classroom reliability. Future work should (i) test the system with a larger, defined group of students over repeated trials and report recognition, false acceptance, false rejection and spoof-rejection rates and latency; (ii) add a challenge-response or depth-based cue to resist video replay; (iii) tune the EAR thresholds on recorded data; (iv) move recognition to an edge device such as a Raspberry Pi 4 or NVIDIA Jetson Nano to remove the laptop; (v) add infrared illumination for dim rooms; and (vi) survey lecturers and students on acceptance and privacy concerns.

ETHICS STATEMENT

Participants were two student volunteers, one of whom is the first author. Both registered themselves in the system and agreed to the capture of their facial images for testing and to the use of these images in this paper.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

DATA AVAILABILITY

The data are not publicly available because they contain students' facial images. Non-identifying data and the source code are available from the corresponding author on reasonable request.

ACKNOWLEDGEMENTS

The authors thank the Faculty of Electronic Engineering & Technology, Universiti Malaysia Perlis, for the facilities used in this work.

REFERENCES

1. Abdylhalyk, D., Zhuma, S., Aubakir, Y., & Kydyrbekova, A. (2026). An AI-based face recognition approach for student attendance: A prototype system. In *2026 International Conference on Cybersecurity, Digital Forensics, and AI Applications (ICCSDFAI)* (pp. 145–150). IEEE. https://doi.org/10.1109/ICCSDFAI70505.2026.11648057
2. Chintalapati, S., & Raghunadh, M. V. (2013). Automated attendance management system based on face recognition algorithms. In *2013 IEEE International Conference on Computational Intelligence and Computing Research* (pp. 1–5). IEEE. https://doi.org/10.1109/ICCIC.2013.6724266
3. Credé, M., Roch, S. G., & Kieszczynka, U. M. (2010). Class attendance in college: A meta-analytic review of the relationship of class attendance with grades and student characteristics. *Review of Educational Research, 80*(2), 272–295. https://doi.org/10.3102/0034654310362998
4. Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Vol. 1, pp. 886–893). IEEE. https://doi.org/10.1109/CVPR.2005.177

5. Das, B., & Halder, K. (2024). Face recognition using ESP32-Cam for real-time tracking and monitoring. In *2024 International Conference on Advances in Computing, Communication, Electrical, and Smart Systems (iCACCESS)* (pp. 1–6). IEEE. https://doi.org/10.1109/iCACCESS61735.2024.10499606
6. Deephika, R., Kaviya, S., & Gowri Vidhya, N. (2026). Blink verified face recognition attendance system using Dlib embeddings and deep learning techniques. In *2026 International Conference on Electronic Systems and Intelligent Computing (ICESIC)* (pp. 598–602). IEEE. https://doi.org/10.1109/ICESIC67389.2026.11496542
7. Dewi, C., Chen, R.-C., Chang, C.-W., Wu, S.-H., Jiang, X., & Yu, H. (2022). Eye aspect ratio for real-time drowsiness detection to improve driver safety. *Electronics, 11*(19), 3183. https://doi.org/10.3390/electronics11193183
8. Dewi, C., Chen, R.-C., Jiang, X., & Yu, H. (2022). Adjusting eye aspect ratio for strong eye blink detection based on facial landmarks. *PeerJ Computer Science, 8*, e943. https://doi.org/10.7717/peerj-cs.943
9. Gaurav, K. (2026). An intelligent attendance monitoring system using AI and computer vision. *International Journal for Research in Applied Science and Engineering Technology*. https://doi.org/10.22214/ijraset.2026.80132
10. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). IEEE. https://doi.org/10.1109/CVPR.2016.90
11. Hosen, M. A., Moz, S. H., Khalid, M., Kabir, S. S., & Galib, S. (2023). Face recognition-based attendance system with anti-spoofing, system alert, and email automation. *Radioelectronic and Computer Systems*, (2). https://doi.org/10.32620/reks.2023.2.10
12. International Organization for Standardization. (2023). *ISO/IEC 30107-3:2023 Information technology — Biometric presentation attack detection — Part 3: Testing and reporting*. ISO.
13. Kanapathy, P., Mustafa, N., Yusop, N., Ibrahim, Y., Moketar, N. A., & Hakimi, H. (2026). Enhancing attendance authenticity in higher education using a QR code and GPS-based smart attendance system. *International Journal of Research and Innovation in Social Science, 10*(1), 7855–7867. https://doi.org/10.47772/IJRISS.2026.10100611
14. Kassarnig, V., Bjerre-Nielsen, A., Mones, E., Lehmann, S., & Lassen, D. D. (2017). Class attendance, peer similarity, and academic performance in a large field study. *PLoS ONE, 12*(11), e0187078. https://doi.org/10.1371/journal.pone.0187078
15. Kazemi, V., & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1867–1874). IEEE. https://doi.org/10.1109/CVPR.2014.241
16. King, D. E. (2009). Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research, 10*, 1755–1758. https://www.jmlr.org/papers/v10/king09a.html
17. King, D. E. (n.d.). *face_recognition.py* [Example program]. dlib C++ Library. Retrieved October 23, 2026, from http://dlib.net/face_recognition.py.html
18. Makigod, G. M. G., Arboleda, E. R., Sebastian, J. J. B., & Ongquit, L. A. T. (2024). Design and development of IoT-based automated attendance system using facial recognition. *ITEGAM Journal of Engineering and Technology for Industrial Applications, 10*(48), 135–143. https://doi.org/10.5935/jetia.v10i48.976
19. Masalha, F., & Hirzallah, N. (2014). A students attendance system using QR code. *International Journal of Advanced Computer Science and Applications, 5*(3). https://doi.org/10.14569/IJACSA.2014.050310
20. Muzammil, M., Khan, M. A., Yunus B, M., Raza, M. N., Wakde, B., & Komal V, S. (2026). AI smart attendance system using face recognition. *Iconic Research and Engineering Journals, 10*(4). https://doi.org/10.64388/IREV10I4-1723595
21. Nwabuwe, A., Sanghera, B., Alade, T., & Olajide, F. (2023). Fraud mitigation in attendance monitoring systems using dynamic QR code, geofencing and IMEI technologies. *International Journal of Advanced Computer Science and Applications, 14*(4), 938–945. https://doi.org/10.14569/IJACSA.2023.01404104
22. Personal Data Protection Act 2010 (Act 709). (2010). *Laws of Malaysia*.

23. Potdar, A., Barbhaya, P., & Nagpure, S. (2022). Face recognition for attendance system using CNN based liveliness detection. In *2022 International Conference on Advances in Computing, Communication and Materials (ICACCM)* (pp. 1–6). IEEE. https://doi.org/10.1109/ICACCM56405.2022.10009024
24. Ramachandra, R., & Busch, C. (2017). Presentation attack detection methods for face recognition systems: A comprehensive survey. *ACM Computing Surveys, 50*(1), Article 8. https://doi.org/10.1145/3038924
25. Sapna, M. K. (2025). FaceRF: A secure, dual-factor automated attendance system with server-side liveness detection. *International Research Journal of Computer Science, 12*(10). https://doi.org/10.26562/irjcs.2025.v1210.07
26. Shamila, L. M. A. (2025). QR code based attendance management system. *International Journal of Research and Innovation in Social Science, 9*(4). https://doi.org/10.47772/IJRISS.2025.90400372
27. Singh, A. K., Joshi, P., & Nandi, G. C. (2014). Face recognition with liveness detection using eye and mouth movement. In *2014 International Conference on Signal Propagation and Computer Technology (ICSPCT)* (pp. 592–597). IEEE. https://doi.org/10.1109/ICSPCT.2014.6884911
28. Soukupová, T., & Čech, J. (2016). Real-time eye blink detection using facial landmarks. In *21st Computer Vision Winter Workshop*. Rimske Toplice, Slovenia.
29. Sunaryono, D., Siswantoro, J., & Anggoro, R. (2021). An android based course attendance system using face recognition. *Journal of King Saud University – Computer and Information Sciences, 33*(3), 304–312. https://doi.org/10.1016/j.jksuci.2019.01.006
30. Surantha, N., & Sugijakko, B. (2024). Lightweight face recognition-based portable attendance system with liveness detection. *Internet of Things, 25*, 101089. https://doi.org/10.1016/j.iot.2024.101089
31. Yu, Z., Qin, Y., Li, X., Zhao, C., Lei, Z., & Zhao, G. (2023). Deep learning for face anti-spoofing: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 45*(5), 5609–5631. https://doi.org/10.1109/TPAMI.2022.3215850