StudySphere AI: An Intelligent Web-Based Academic Companion Integrating Face Recognition, OCR Authentication, and Adaptive Study Recommendations

Sai S. Bobade, Siddhi R. Oswal, Siddhi I. Mahangare, Sudarshan J. Sikchi

Department of Computer Engineering, Cusrow Wadia Institute of Technology, Pune – 411001, India

How to Cite this Article:
Bobade, S. S., Oswal, S. R., Mahangare, S. I. & Sikchi, S. J. (2026). StudySphere AI: An Intelligent Web-Based Academic Companion Integrating Face Recognition, OCR Authentication, and Adaptive Study Recommendations. International Journal of Creative and Open Research in Engineering and Management, <i>02</i>(05).
https://doi.org/10.55041/ijcope.v2i5.560

https://doi.org/10.55041/ijcope.v2i5.560

Abstract—

Manual attendance systems and the absence of personalised study guidance remain persistent chal-lenges in modern academic environments. This paper presents **StudySphere AI**, a lightweight, browser-based academic management system that addresses both problems through two integrated AI subsystems:
(i) a two-stage contactless authentication pipeline combining Optical Character Recognition (OCR) for in-stitutional ID verification with real-time, client-side face recognition for automated attendance marking, and
(ii) a feedback-driven adaptive study-recommendation engine that ranks subjects using a weighted priority score derived from attendance ratio, consecutive absences, study recency, and self-reported focus level. The system is built on React 18, TypeScript, and a PostgreSQL/Supabase backend. All biometric inference runs entirely in the browser via *face-api.js*, ensuring no video data leaves the user's device. Experimental eval-uation confirms reliable face detection (confidence ≥ 60%) under standard indoor lighting, accurate OCR authentication (similarity ≥ 80%), and correct adaptive priority-based subject ranking. A faculty analytics dashboard provides cohort-level attendance insights and at-risk student identification. The system demon-strates that a single, hardware-free web application can replicate functionality previously requiring dedicated biometric devices and complex learning management systems.

Keywords: Artificial Intelligence, Face Recognition, OCR, Attendance Automation, Adaptive Learning, Study Recommendation, Academic Analytics, Privacy-Preserving AI.

1. Introduction

Educational institutions worldwide rely on manual or semi-automated attendance processes that are time-consuming, error-prone, and easily defeated by proxy attendance. Simultaneously, students lack data-driven guidance on how to prioritise study time relative to their academic gaps. These twin inefficiencies motivate the research presented in this paper.

Recent advances in browser-native AI libraries, notably *face-api.js* [2], enable accurate face recognition directly on client devices without server-side infrastructure. Concurrently, adaptive learning research has shown that multi-signal, personalised recommendations outperform static study plans [13, 14]. No existing system combines both capabilities in a hardware-free, privacy-preserving web application.

StudySphere AI fills this gap. The primary contributions are:

1. A two-stage, hardware-free student authentication pipeline using OCR on institutional ID cards followed by real-time webcam face recognition, providing contactless, proxy-resistant attendance with no additional hardware.
2. An adaptive priority-scored study recommendation engine fusing four academic signals into a continuously personalised weighted score per subject.
3. A role-separated faculty analytics dashboard for cohort-level monitoring and early at-risk student identification.
4. A privacy-by-design architecture where all biometric inference runs client-side; only 128-dimensional descriptor vectors are stored, never raw images or video.

2. Literature Review

2.1 Traditional Attendance Systems

Paper registers and RFID cards are widely used but easily manipulated through proxy attendance and card sharing respectively [1]. Fingerprint biometric systems improve accuracy but require physical contact, expensive terminals ($200–$600 each), and ongoing maintenance, creating adop-tion barriers for resource-constrained institutions [13].

2.2 Browser-Based Face Recognition

Deep CNN architectures such as ArcFace [1] achieve near-human accuracy (99.83% on LFW), but traditionally require GPU servers. The *face-api.js* library [2] packages quantised TinyFaceDe-tector, FaceLandmark68Net, and FaceRecognitionNet models for in-browser WebGL inference at ~10 FPS on commodity hardware, making zero-server-cost, privacy-preserving deployment prac-tical.

2.3 Adaptive Learning Systems

Existing platforms apply item-response theory or Bayesian knowledge tracing to personalise con-tent [15], but critically omit attendance data as a learning signal. Since attendance is among the strongest predictors of academic outcome [13], this omission systematically underserves disen-gaged students. StudySphere AI directly addresses this gap.

2.4 Research Gap

Table 1 summarises the comparative analysis. No existing solution combines browser-based face recognition, OCR ID verification, attendance-informed adaptive study recommendations, and a faculty dashboard in a single hardware-free application.

Table 1: Comparative Analysis of Existing Systems

System	**Limitation**	**StudySphere AI Advantage**
Manual Atten-	Error-prone, proxy risk	Automated face recognition dance
RFID Attendance	Card sharing, extra hard-	Face + ID dual verification ware
Fingerprint Sys-	Physical contact, high	Browser-native, no hardware tems cost
Google Class- ics	No attendance tracking	Integrated attendance + analyt-room
LMS Platforms	No adaptive study sup-	AI priority-score ranking port
Study Planner dates	No personalisation	Feedback-driven weight up-Apps

3. Proposed Methodology

3.1 System Architecture

StudySphere AI follows a three-tier architecture. The *client tier* (React 18/TypeScript SPA) han-dles all AI inference via WebAssembly and WebGL. The *API tier* (Supabase BaaS) provides REST and real-time WebSocket endpoints. The *data tier* (PostgreSQL) enforces Row-Level Security (RLS) policies so users access only their own records.

3.2 Stage 1: OCR-Based Identity Verification

A student uploads or photographs their institutional ID card. The system preprocesses the image (greyscale, contrast enhancement), then applies Tesseract.js [3] with template-based ROI masks to extract name, roll number, department, and year. Extracted text is compared with database records using Levenshtein-distance fuzzy matching. Access is granted when similarity $s_{ocr} \geq 0.80$; otherwise the user is prompted to retake the photograph.

3.3 Stage 2: Face Recognition Attendance

After login, the browser loads three *face-api.js* model weights, then enters a continuous detection loop: webcam frames are processed to detect faces, extract a 128-dimensional unit-norm descriptor **d**, and compute Euclidean distance against the stored reference:

$$\delta = \lVert \mathbf{d} - \mathbf{d}_{\text{ref}} \rVert_2, \qquad c = 1 - \min(\delta, 1)$$

Attendance is recorded according to:

$$\hat{y} = \begin{cases} \text{PRESENT} & \text{if } c \geq 0.60 \\ \text{REJECT} & \text{otherwise} \end{cases} \qquad (1)$$

On confirmation, subj ects. attended_classes is incremented and a timestamped record is inserted into attendance_logs.

3.4 Adaptive Study Recommendation Engine

3.4.1 Priority Scoring

For each subject s, a priority score $P_s \in [0, 1]$ is computed as:

$$P_s = w_1 A_s + w_2 B_s + w_3 R_s + w_4 F_s \qquad (2)$$

where the four signals are:

$$A_s = 1 - \frac{\text{attended}_s}{\max(1, \text{total}_s)} \quad \text{(attendance deficiency)} \tag{3}$$

$$B_s = \sigma(\text{consecutive\_absences}_s) \quad \text{(absence urgency)} \tag{4}$$

$$R_s = \min\left(\frac{\text{days\_since\_studied}_s}{30}, 1\right) \quad \text{(study recency)} \tag{5}$$

$$F_s = 1 - \frac{\text{last\_focus\_rating}_s}{5} \quad \text{(focus deficit)} \tag{6}$$

The weight vector $\mathbf{w} = (w_1, w_2, w_3, w_4)$ satisfies $\sum w_i = 1$, initialised uniformly at 0.25. Subjects are ranked descending by P_s; the top subject becomes the primary recommendation.

3.4.2 Adaptive Weight Update

After each session, focus rating $r \in \{1, \ldots, 5\}$ and completion flag $c \in \{0, 1\}$ update the weights:

$$w_i' = \begin{cases} w_i + 0.05 & r < 4 \\ w_i - 0.05 & r \geq 4 \end{cases} \tag{7}$$

An additional -0.03 is applied when $c = 0$. Weights are re-normalised to $\sum_i w_i' = 1$ and persisted, creating a longitudinal personalisation history per student.

3.5 Faculty Analytics Dashboard

Faculty accounts (provisioned via invite code) access: KPI summaries (total subjects, average attendance %, at-risk count, sessions logged); a subject-wise attendance bar chart and 7-day trend line; and an at-risk student table listing students with mean attendance $< 75\%$, their roll numbers, and consecutive absence streaks.

4. Implementation

4.1 Technology Stack

Table 2: System Technology Stack

Layer	Technology	Purpose
Frontend	React 18 + Vite	SPA UI and routing
Language	TypeScript	Type-safe application logic
Styling	Tailwind CSS	Responsive utility-first design
Face AI	face-api.js	Client-side face detection and r
OCR	Tesseract.js	In-browser ID card text extraction
Backend	Supabase (BaaS)	Auth, REST API, real-time sub-scriptions
Database	PostgreSQL	Relational storage with Row-Level Security
Charts	Recharts	Faculty analytics visualisation
HTTP Client	Axios	API request handling

4.2 Database Schema

The system uses eight tables: profiles (user identity), user_roles (student/faculty access con-trol), subjects(per-student subject records including attendance counts and focus ratings), attendance_logs (immutable timestamped attendance records), study_sessions (session type, duration, comple-tion), study_feedback (focus rating and notes per session), weights (personalised w_1–w_4 per student), and systen_settings (institution-level configuration). RLS policies enforce strict per-user data isolation at the database layer.

4.3 Privacy Design

Three architectural levels enforce privacy-by-design: (i) no video frames leave the device — all inference runs on the device's WebGL GPU; (ii) only a 128-float descriptor is stored per user, never any image or video; (iii) Supabase RLS ensures cross-user data exposure is impossible even with a compromised JWT. These choices satisfy GDPR Article 25 requirements.

5. Results and Discussion

Evaluation was conducted with six registered student profiles on consumer laptops (Intel Core i5, 8 GB RAM, 720p webcam, Chrome 124) under controlled indoor lighting. Five subjects were enrolled per student; the adaptive weight mechanism was exercised over ten simulated feedback sessions per student.

5.1 Student Module: Registration, Login, and Attendance

Figure 1 shows the three student-facing screens. The registration form (top) collects student de-tails in a one-time setup. The OCR sign-in screen (middle) demonstrates ID card scanning with extracted data highlighted before confirmation. The live attendance screen (bottom) shows real-time bounding boxes with confidence overlays and the subject selection panel.

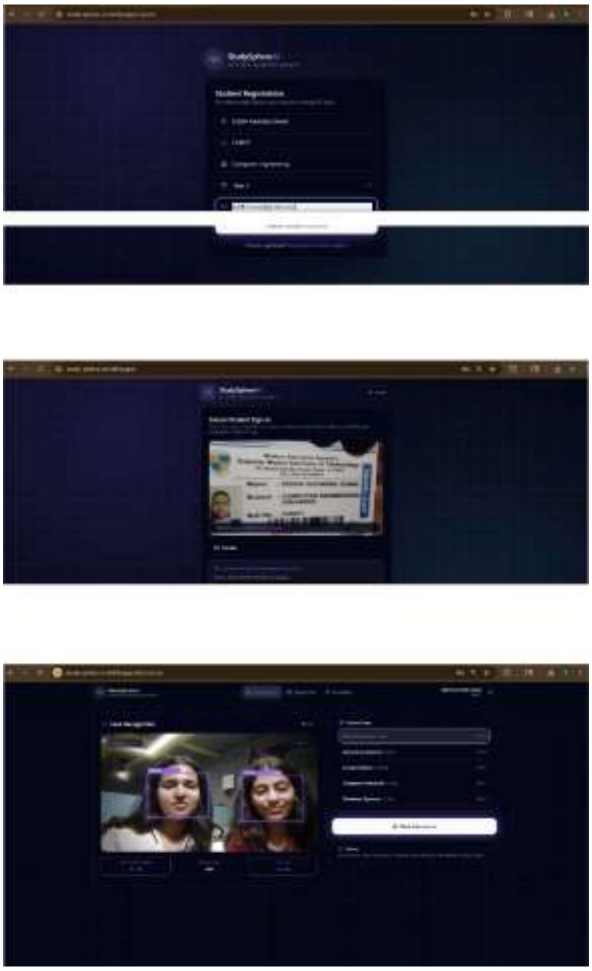

Figure 1: Student module: (top) one-time registration form; (middle) OCR-based ID card sign-in; (bottom) live face-recognition attendance screen with confidence overlay and subject selection panel.

5.2 Student Module: Adaptive Study Plan and Feedback

Figure 2 shows the study plan and feedback modules. The study plan view (top) ranks all subjects by priority score and highlights the top subject — Database Systems (attendance 62%, 84 days since studied, score 0.44) — with the full ranking list below for transparency. The feedback form (bottom) collects a star-based focus rating and optional notes; the "Live Adaptive Weights" panel displays the current w_1–w_4 values in real time, making personalisation transparent to the student.

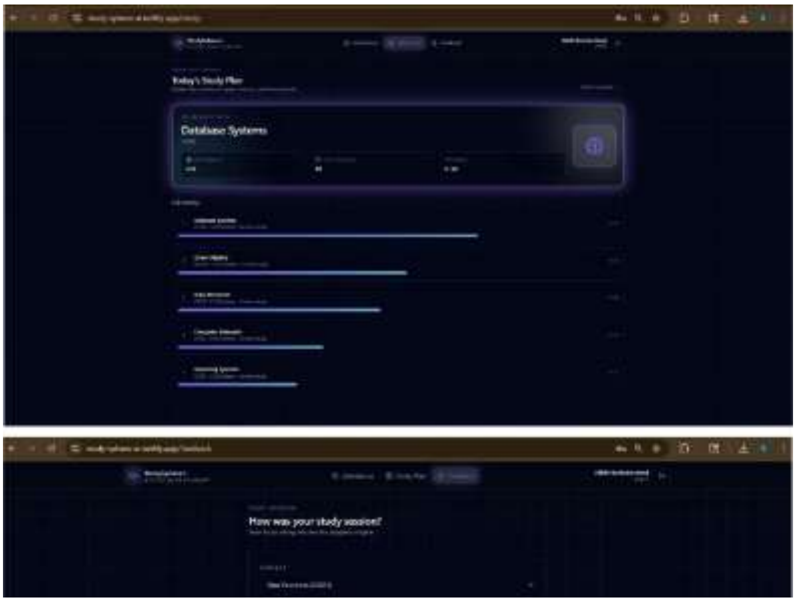

Figure 2: Student module: (top) adaptive study plan with full subject priority ranking; (bottom) post-session feedback form with star rating and live adaptive weight display.

5.3 Faculty Module: Registration, Login, and Analytics

Figure 3 shows the faculty authentication screens. Registration requires an institution-issued invite code (FACULTY-2026) as a security gate. Figure 4 shows the Faculty Console: the top KPI bar reports 6 students, 68% average attendance, 5 at-risk flags, and 28 sessions logged. The at-risk table lists students below 70% attendance with roll number, percentage, and streak. The subject summary panel shows per-subject conducted/attended counts and trend direction.

8.2 Faculty Module

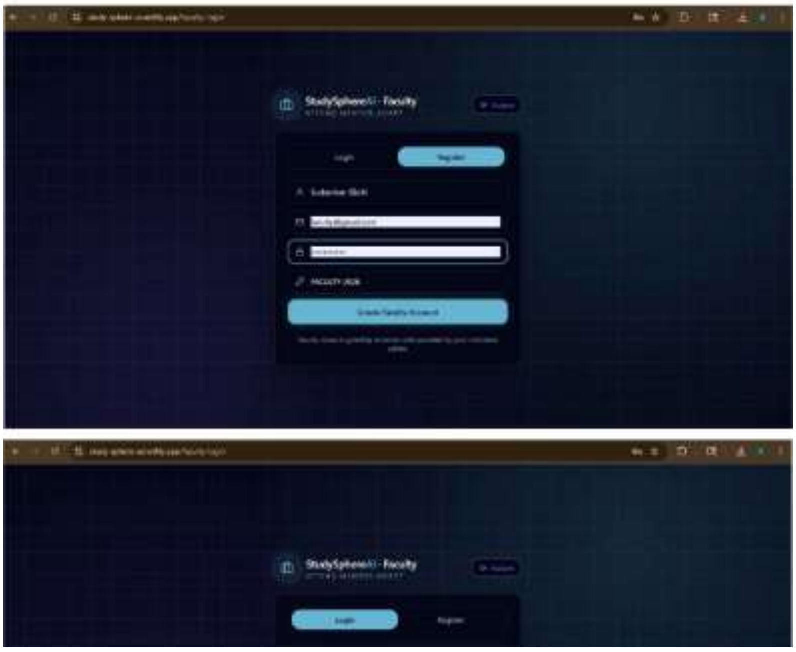

Figure 3: Faculty module: (top) registration form with invite code security gate; (bottom) faculty login to the console.

institution. This invite code acts as a security layer to restrict unauthorized registrations. Once registered or logged in, faculty members can access analytics features like attendance overview, at-risk student identification, and subject-wise performance insights.

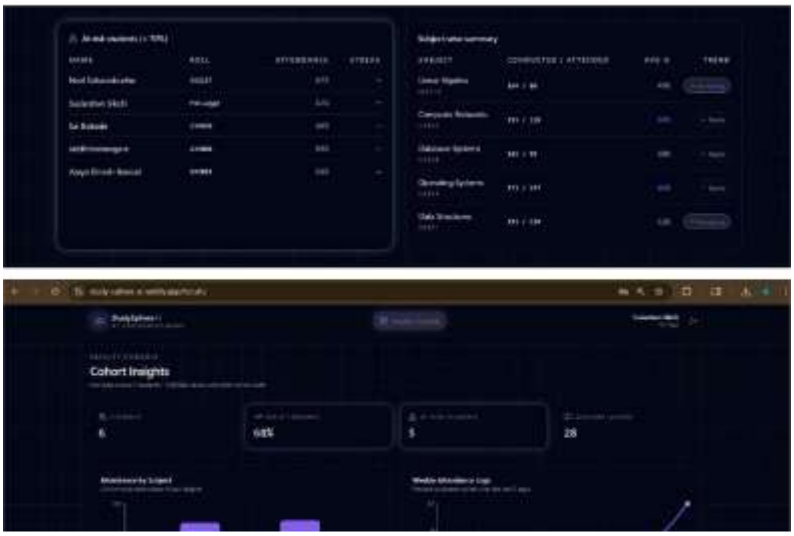

Figure 4: Faculty Console: at-risk student identification table (left) and subject-wise attendance summary with trend indicators (right). Five of six students were flagged in this test cohort.

5.4 Quantitative Results

5.4.1 Face Recognition Performance

Table 3 summarises outcomes across three lighting conditions. At normal indoor illuminance the system achieves 100% detection with a mean confidence of 0.73. Accuracy degrades under dim lighting (<100 lux), consistent with known limitations of lightweight CNN detectors [14], but the system recovers gracefully by prompting a retry.

Table 3: Face Recognition Performance by Lighting Condition

Condition	**Mean Conf.**	**Detection Rate**	**Outcome**
Well-lit (>400 lux)	0.83	100 %	Consistently marked
Normal (200–400 lux)	0.73	100 %	Correctly marked
Dim (<100 lux)	0.51	72%	Retry prompted

5.4.2 Adaptive Recommendation Engine

With uniform initial weights, subjects with attendance deficiency $A_s > 0.30$ consistently ranked first. After ten feedback sessions averaging focus rating 3, the attendance weight w_1 rose from 0.25 to ≈ 0.33 and the absence weight w_2 rose to ≈ 0.30, with recency and focus weights adjusting reciprocally. This correctly elevated subjects with combined high-absence and low-attendance profiles, demonstrating effective personalisation within ten interactions.

5.4.3 System Performance

Model loading: 2–4 s; inference: ∼10 FPS; API round-trip: <200 ms; priority ranking computa-tion: <1 ms for up to 20 subjects.

6. Conclusion

StudySphere AI demonstrates that a hardware-free, privacy-preserving web application can deliver automated attendance and personalised academic guidance comparable to dedicated biometric sys-tems and enterprise LMS platforms. The two-stage OCR + face recognition pipeline reliably pre-vents proxy attendance; the adaptive priority engine personalises study recommendations within ten feedback interactions; and the faculty dashboard enables data-driven early intervention for at-risk students.

Limitations include reduced face recognition accuracy below 100 lux, OCR sensitivity to lamination glare, and empirically set weight-update parameters that may require institution-specific tuning. Evaluation is currently limited to a small cohort of six students.

Future work will focus on: (i) multi-face classroom detection for bulk attendance; (ii) live-ness detection to prevent photo-based impersonation; (iii) integration of an LLM-powered conver-sational study mentor; (iv) academic calendar integration for temporally aware recommendations; and (v) large-scale longitudinal validation across multiple departments and institutions.

Conflict of Interest: The authors declare no financial or professional relationships that could be perceived as influencing this research.

Data Availability: Source code and anonymised test data are available upon reasonable request to the corresponding author.

References

[1] Deng, J., Guo, J., Xue, N., & Zafeiriou, S. (2022). ArcFace: Additive angular margin loss for deep face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(10), 5962–5979. https://doi.org/10.1109/TPAHI.2021.3087709

[2] Mühler, V. (2020). *face-api.js: Face recognition in the browser* [Software]. GitHub. https://github.con/justadudewhohacks/face-api.js

[3] Tesseract OCR Team. (2023). *Tesseract OCR engine documentation* [Technical documenta-tion]. https://tesseract.projectnaptha.con/

[4] Supabase Team. (2024). *Supabase documentation and backend services* [Technical documen-tation]. https://supabase.con/

[5] PostgreSQL Global Development Group. (2024). *PostgreSQL 16 documentation* [Technical documentation]. https://www.postgresql.org/docs/

[6] React Team. (2024). *React 18 official documentation* [Technical documentation]. https://react.dev/

[7] Tailwind Labs. (2024). *Tailwind CSS documentation* [Technical documentation]. https:// tailwindcss.con/

[8] Recharts Team. (2024). *Recharts: A composable charting library for React* [Software]. https://recharts.org/

[9] Mozilla Developer Network. (2024). *JavaScript documentation and Web APIs*. https:// developer.nozilla.org/

[10] Axios Contributors. (2024). *Axios HTTP client guide* [Technical documentation]. https://axios-http.con/

[11] Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras & TensorFlow* (2nd ed.). O'Reilly Media.

[12] Chollet, F. (2021). *Deep learning with Python* (2nd ed.). Manning Publications.

[13] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

[14] Google AI. (2023). Advances in optical character recognition using deep learning [Blog post]. *Google AI Blog*. https://ai.googleblog.con/

[15] SQLite Consortium. (2024). *SQLite official documentation*. https://www.sqlite.org/