



International Journal of Science, Architecture, Technology and Environment

UPStudy: AI-Powered Academic Management & Learning Platform

Prof. Swati Kosankar^{1*}, Priyanshu Ambhore², Princy Tabhane³, Yash Tamhankar⁴, Vedant Birgade⁵, Kunal Jiwtode⁶, Kunali Pathrabe⁷

^{1,2,3,4,5,6,7}Department of Computer Science and Engineering, G H Rasoni University, Amravati, India

*Corresponding author, swati.kosankar@ghrua.edu.in

DOI: <https://doi.org/10.63680/ijstate102649.43>

Abstract

UPStudy is an integrated academic management and learning platform designed for higher education institutions. It combines attendance processing, assignment tracking, learning resources, AI-assisted study support, quizzes, academic analytics, course learning, a student learning marketplace, and notifications in a single web application. The platform emphasizes early intervention by identifying low-attendance or academic conditions that require attention and presenting timely alerts. Faculty can upload attendance data through CSV, image, or PDF workflows, review OCR-derived records, manage assignments and resources, and monitor academic indicators. Students can access resources, use AI assistance, attempt quizzes, track learning progress, and participate in controlled course or marketplace workflows. The implementation follows a local-first design using React, Vite, Node.js, local JSON/file storage, Tesseract.js, PDF.js, Recharts, and optional local Ollama with an offline fallback. Controlled evaluation covered functional behavior, analytics calculations, security- boundary probes, and local performance. The design prioritizes simplicity, explainability, low-cost operation, and reproducibility in a college environment.

Keywords: Academic Management, OCR Attendance, Early Intervention, AI Learning Assistant, Quiz Generation, Learning Marketplace, Academic Analytics

I. INTRODUCTION

Higher education generates a continuous stream of attendance records, assignment submissions, learning resources, assessments, progress data, and student communication. These activities are often handled through separate tools or manual processes, which can delay updates and make academic follow-up difficult. The project material identifies manual attendance, inefficient assignment tracking, scattered resources, limited AI support, and weak student-faculty communication as important problems addressed by UPStudy.

UPStudy is designed as a single academic ecosystem rather than only a dashboard. It connects academic records with learning and intervention: attendance and assignment information can be processed, academic indicators can be derived, and alerts can be generated when a defined condition is met. On the learning side, students and faculty can share educational material, while students can use AI assistance, practice quizzes,

courses, and progress tracking. The overall objective is to reduce repetitive faculty work while improving student visibility and timely academic action. [11]

II. RELATED WORK

Existing educational platforms such as Google Classroom, Moodle, Vedantu, and Brainly demonstrate the value of centralized resources, assessment, communication, and student engagement. Research on AI and learning analytics highlights the potential for data-driven academic decisions and personalized intervention while also emphasizing accuracy and data-security concerns. Work integrating attendance monitoring with formative assessment supports the value of connecting academic monitoring with learning feedback. Ray Smith's Tesseract work provides a foundation for open-source OCR-based document text extraction, while recent studies on automated attendance and AI educational assistants are directly relevant to UPStudy's OCR and AI modules. These works motivate an integrated approach that combines academic management, OCR processing, explainable analytics, and AI-assisted learning in one platform. [1]-[5]

III. PROBLEM STATEMENT

The problem addressed by UPStudy is the fragmentation of academic management and learning support in higher education. Attendance may be maintained manually, assignment status may require repeated checking, learning material may be distributed through multiple channels, and students may not receive timely information when they fall below academic expectations. Useful material created by students and faculty may also remain distributed across disconnected platforms. UPStudy addresses this gap by providing one platform in which academic data is collected and processed, students receive condition-based notifications, AI supports learning activities, and educational content is shared through structured course and marketplace workflows.

IV. OBJECTIVES

- Automate attendance processing using OCR and reduce repetitive manual entry.
- Detect low-attendance conditions using a configurable threshold and generate timely alerts.
- Digitize assignment creation, submission tracking, pending-work identification, and reminders.
- Provide a centralized repository for notes, PDFs, recorded lessons, previous-year questions, and study resources. Provide AI-assisted concept explanation, coding help, exam preparation, summarization, and practice-quiz generation.
- Provide explainable analytics for attendance, assignments, quiz performance, and course progress.
- Allow students and faculty to create and share courses and educational resources, with enrollment, progress tracking, and certificates.
- Improve student-faculty communication through in-app alerts and configurable notification adapters.

V. PROPOSED SYSTEM

UPStudy is a role-based web application for Student, Faculty, and Admin users. The Student dashboard provides attendance, assignments, notifications, AI support, quizzes, courses, and progress. Faculty tools support attendance upload and review, assignment management, resource sharing, course management, and academic analytics. Admin functions provide user oversight, moderation, content control, and system-level summaries. The platform also includes a learning marketplace and crash-course workflow in which creators can publish structured learning material and learners can browse, preview, enroll, or use a controlled demo purchase/unlock flow.

The core technology stack is React 19 with Vite 8 and React Router 7 on the client side, a Node.js built-in HTTP server/API layer, local JSON and file storage, Tesseract.js and PDF.js for document processing, optional local Ollama with an offline deterministic fallback for AI features, Recharts for analytics, and Lucide React for interface icons. Email is designed through a configurable adapter, with local demo behavior when no provider is configured. [7]-[9]

VI. SYSTEM ARCHITECTURE

The system follows a simple layered architecture. Students, faculty, and administrators access a React-based web interface, which communicates with the Node.js application/API layer. The API layer coordinates

authentication, academic workflows, analytics, notifications, course and marketplace operations, OCR services, and AI services. Academic data and uploaded files are stored locally in the current project architecture. Tesseract.js and PDF.js provide OCR and PDF processing, while optional Ollama provides local AI support; a deterministic fallback allows core demonstrations to continue without an external model. This local-first design reduces infrastructure requirements and keeps the system understandable and reproducible for academic use.

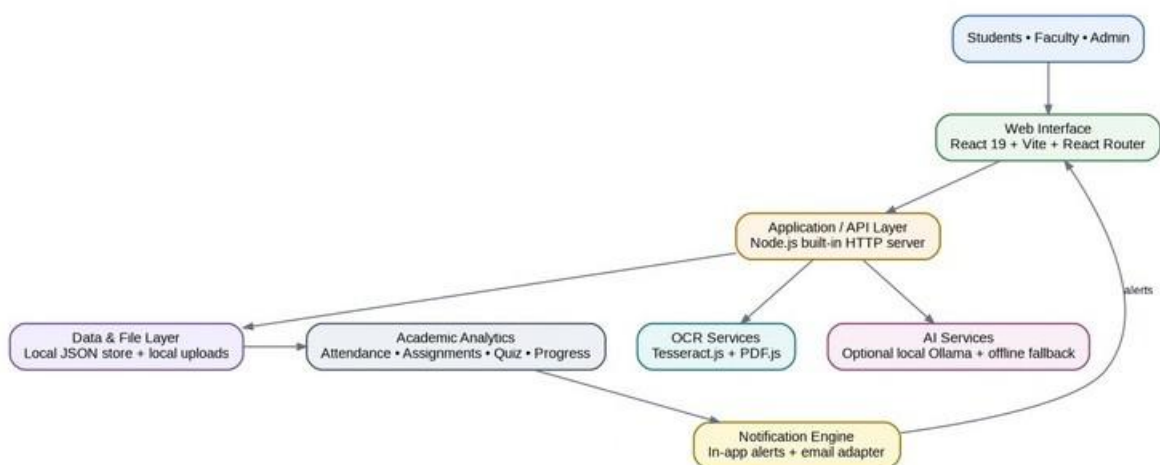


Fig. 1. Simple layered software architecture of the UPStudy platform.

The principal data flow is: user interface -> API/application layer -> local data and file layer, with OCR, analytics, AI, and notification components invoked as required by each workflow. Faculty review remains part of the attendance OCR process before records are finalized, supporting controlled and explainable data entry.

VII. METHODOLOGY

Development followed a modular and iterative approach. Requirements were first grouped around Student, Faculty, and Admin roles; common academic data and notification workflows were then defined across modules. UI/UX screens were designed for role-specific dashboards and learning pages, followed by frontend implementation using React and Vite and backend API development using Node.js built-in HTTP facilities.

OCR integration used Tesseract.js and PDF.js, while optional local Ollama and a deterministic offline fallback were added for AI functionality. Data and uploaded files were maintained locally, and the implementation was tested with isolated synthetic data and saved evaluation scripts. The methodology intentionally avoids unnecessary cloud dependencies so that the software can be reproduced in a college environment.

VIII. FUNCTIONAL MODULES

Module		
Student	Dashboard	Attendance, assignments, notifications, AI support, quizzes, courses, progress.
Faculty Dashboard		Attendance review, assignments, resources, course management, analytics.
Admin	Dashboard	User oversight, moderation, content control, system-level summaries.
Attendance	& OCR	Image/PDF processing, OCR extraction, review, validation, attendance storage, risk alerts.
Assignments & Resources		Creation, submission tracking, pending work, reminders, notes, PDFs, recorded lessons, PYQs.
AI Assistant	& Quiz	Concept explanation, coding help, exam support, summarization, question generation, scoring.
<u>Courses</u> & Marketplace		Course/resource creation, browsing, preview, enrollment/demo purchase, progress, certificates.
Analytics & Notifications		Attendance, assignment, quiz and course indicators; in-app alerts and notification adapters.

Table 1. Summary of the main functional modules implemented in UPStudy.

IX. SECURITY CONSIDERATIONS

Security is treated as a practical design requirement. Authentication and role authorization are enforced at the server/API layer rather than relying only on frontend route guards. Input validation is applied to academic values and uploaded files, passwords are handled using salted hashing in the existing implementation, and environment configuration is kept outside published source content. AI prompts and uploaded documents are treated as untrusted inputs, and the OCR workflow is reviewable by faculty before final persistence. The saved security evaluation also documents weaknesses observed at an earlier prototype boundary, including unauthenticated access and an assignment-mutation case; therefore, the system is

presented as a college-level prototype rather than a production security product unless full server-side hardening is deployed and re-tested. [6]

X. RESULTS AND DISCUSSION

Testing used isolated synthetic data to evaluate software behavior without treating seeded demonstration records as real student observations. The saved evaluation artifacts include unit tests, functional checks, analytics fixtures, security probes, and a local performance benchmark. The primary functional and analytics suites

completed without failure, while security probes recorded boundary observations for the prototype.

Evaluation scope	Size	Result
Unit tests	8	8 passed, 0 failed
Functional checks	17	17 passed
analytics fixtures	5	5 passed
security probes	10	10 executed; 0 execution errors; boundary observations recorded
Performance	8 operations	0 recorded failures

Table 2. Summary of controlled software evaluation.

The warm local benchmark used 20 calls after warm-ups for each operation and intentionally used offline AI fallback. Reported p50 values ranged from 8.334 ms to 25.078 ms, p95 values from 13.564 ms to 34.721 ms, and the recorded failure rate was 0% for all benchmark groups. The benchmark was collected on the local Windows x64 environment and should not be generalized to large concurrent deployments. The screenshots below show representative implemented interfaces for faculty monitoring, attendance/OCR review, AI quiz generation, and the learning marketplace.

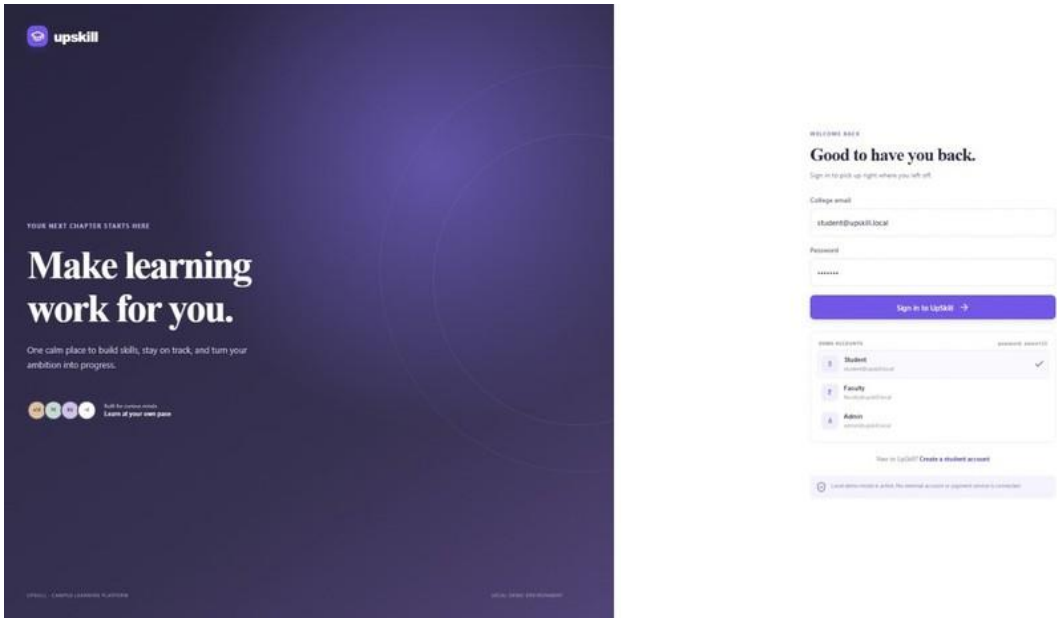


Fig. 2. Representative UPStudy interfaces: (a) faculty dashboard, (b) attendance/OCR, (c) AI quiz generation, and (d) learning marketplace.

XI. LIMITATIONS AND FUTURE SCOPE

OCR accuracy has not been established on a representative labelled attendance dataset; future work should compare OCR output with verified ground truth.

AI answer and quiz quality require structured human evaluation using fixed prompts, materials, and a scoring rubric. The local storage design is suitable for a college prototype but should be extended with a production database and stronger deployment security for multi-user use.

Email and other external services should be enabled through configured providers with delivery and failure tracking. Future work can add predictive analytics after sufficient, ethically collected data is available, together with real payment settlement, refunds, complaints, and scalable video delivery through modular adapters.

XII. CONCLUSION

UPStudy integrates academic management and learning support in a single platform. Its main contribution is the connection between academic records and actionable learning workflows: attendance can be processed and reviewed, low-attendance conditions can trigger intervention, assignments can be monitored, resources can be organized, AI can support study activities, quizzes can be generated and scored, and academic performance can be visualized. The course and learning-marketplace modules further support sharing of educational material and structured learning experiences. The current evaluation provides evidence for functional behavior, analytics calculations, local performance, and security-boundary testing. The system is intentionally simple and low-cost for reproducible academic demonstration, while future work should focus on stronger OCR and AI evaluation, production-grade storage and security, and broader user studies.

REFERENCES

- [1] R. Sajja, Y. Sermet, D. Cwiertny, and I. Demir, "Integrating AI and Learning Analytics for Data-Driven Pedagogical Decisions and Personalized Interventions in Education," *Technology, Knowledge and Learning*, 2025, doi: 10.1007/s10758-025-09897-9.
- [2] S. Manickam, M. Al Ismaili, R. Chidambaram Sivamani, and S. Dileesh, "An integrated system for just-in-time formative assessment and attendance monitoring," *Discover Education*, vol. 4, 2025, doi: 10.1007/s44217-025-00960-2.
- [3] R. Smith, "An Overview of the Tesseract OCR Engine," *Proc. Ninth International Conference on Document Analysis and Recognition (ICDAR)*, 2007, doi: 10.1109/ICDAR.2007.4376991.
- [4] I. Rizvi, S. Shaikh, J. Kaur, I. Kampoo, and S. Thakkar, "AttendAI: Automated Attendance Detection," *International Journal of Engineering Research & Technology*, vol. 15, issue 06, 2026.
- [5] K. Jothikumar, G. Velusamy, A. J. Meenakshi, S. Aruna Varshini, and S. Anisha, "Design and Implementation of an AI Integrated Educational Assistant for Learning," *SN Computer Science*, vol. 7, 2026, doi: 10.1007/s42979-026-04984-9.
- [6] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100- 1, 2023.
- [7] React Documentation, Meta Platforms Inc., <https://react.dev/>.

-
- [8] Tesseract.js Documentation, open-source JavaScript OCR library documentation.
- [9] Ollama Documentation, local large-language-model runtime documentation.
- [10] Google Classroom, Brainly, Vedantu, and Moodle were reviewed as representative educational platforms during project planning. [11] UPStudy Project Progress Seminar-02, Department of Computer Science and Engineering, G H Rasoni University, Amravati, Session 2026–27.

Declaration of Conflicting Interests

The authors declare no potential conflicts of interest with respect to the research, authorship and publication of this article.

Funding

The author received no financial support for the research, authorship and publication of this article.