International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

An Integrated Educational Erp and Distributed File System Leveraging Multi-Drive Storage for Secure Academic Management

Aviraj Salunkhe, Swayam Gunjal, Vedant Ghode, Vaishnavi Shirude
Student, Department of Information Technology, NBN Sinhgad School of Engineering, Pune, Maharashtra, India

Abstract - Modern academic institutions require storage infrastructure that combines scalability, security, and seamless integration with administrative systems. Existing Educational Enterprise Resource Planning (ERP) platforms rely on centralized storage architectures that suffer from rigid capacity constraints and weak file-level access governance. This paper introduces ClassCrew, a web-based academic management platform embedded with a distributed storage engine that unifies multiple Google Drive accounts into one virtual storage pool. Prior to external transmission, each uploaded file undergoes AES-256-CBC symmetric encryption using a randomly generated per-file key and initialization vector. The encrypted output is then divided into variable-size segments and distributed across pool accounts by a capacity-aware greedy scheduler. File reconstruction and authenticated decryption are coordinated by a centralized metadata service. Authorization is governed by a Role-Based Access Control engine whose decisions are driven by real-time institutional attributes such as course enrolment, attendance standing, and fee settlement status. Controlled experiments show that files up to 10 MB are delivered within three seconds, pool capacity expands linearly with each additional account, and no single storage node holds enough data to independently reconstruct or decrypt any file. System limitations and future directions including erasure-coded redundancy and client-side key custody are also discussed.

Keywords: Distributed File System, Educational ERP, AES-256-CBC Encryption, Role-Based Access Control, Drive Pooling, Quota-Based Allocation.

I. INTRODUCTION

Academia has undergone a pronounced shift toward digital workflows over the past decade. Student information, timetables, grade records, assignment submissions, fee transactions, and course materials now exist primarily in digital form. The sheer volume and sensitivity of this data impose demanding requirements on any storage system: capacity must scale with institutional growth, files must be protected against unauthorized access, and the storage subsystem must interact meaningfully with the broader administrative software ecosystem. Most deployed ERP platforms in use today were not designed with these requirements in mind and consequently leave significant gaps in security and scalability.

A recurring structural flaw in legacy academic systems is the separation between the ERP application and its backing file store. Files are typically held on a single server or cloud volume, referenced by static URLs that carry no authentication context. Any party who obtains such a URL gains unrestricted access to the associated resource regardless of their role or institutional standing. Simultaneously, the absence of encryption at the storage layer means that a server- side breach can expose an entire institution's data in a single event.

A practical and low-cost mitigation exists in the form of pooled consumer cloud storage. Google's Drive platform allocates 15 GB of free quota to every registered account. Aggregating multiple such accounts—each reachable through a standardized REST API—creates a substantial distributed storage substrate at negligible cost. The engineering challenge is transforming this loose collection of independent accounts into a coherent, encrypted, and access-governed file system tightly coupled with an ERP's user model.

ClassCrew addresses this challenge through four coordinated design decisions: (i) pooling of Drive accounts behind a unified storage interface; (ii) AES-256-CBC encryption of every file before any data leaves the application server; (iii) dynamic partitioning of ciphertext across pool accounts using a quota-aware scheduler; and (iv) fine-grained access enforcement driven by live academic attributes maintained within the ERP. The remainder of this paper details the design, implementation, and evaluation of this approach.

A. Problem Statement

Three inter-related problems motivate this work. First, centralized file storage is capacity-constrained: expanding a single-server back-end requires either capital investment in hardware or recurring payment for premium cloud tiers, both of which are barriers for resource-limited institutions. Second, static URL-based file sharing grants unrestricted access to any URL holder and cannot

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

enforce dynamic, attribute-dependent policies. Third, because ERP logic and storage logic reside in separate systems with no shared context, it is technically infeasible to condition file access on live institutional data such as attendance percentages or fee-clearance flags without building bespoke bridging infrastructure.

These deficiencies have regulatory dimensions. India's Digital Personal Data Protection Act 2023 and the European Union's General Data Protection Regulation both classify academic records as personal data requiring appropriate technical safeguards. Institutions that cannot demonstrate adequate data protection at the storage layer face potential enforcement action and reputational consequences.

B. Objectives

This research aims to: (i) aggregate multiple Google Drive accounts into a unified virtual storage pool accessible through a single API; (ii) protect file confidentiality through symmetric encryption combined with ciphertext segmentation and structural obfuscation; (iii) bind every authorization decision to live ERP data so that access rights reflect the user's current institutional standing; (iv) demonstrate that horizontal capacity scaling is achievable through account addition alone, without architectural change; and (v) validate system behavior through controlled performance experiments.

II. METHODOLOGY

ClassCrew is structured as four loosely coupled layers—presentation, application, persistence, and storage— communicating through narrow, versioned interfaces. Within the application layer, five subsystems carry out the core operations of the platform: drive pooling, encryption-and-splitting, quota-based allocation, access-controlled retrieval, and structural obfuscation. Each subsystem is described below.

A. Drive Pool Management

The storage foundation is a logical pool of independently registered Google Drive accounts. A registry table in the relational database captures each account's identifier, total and consumed quota, OAuth credential reference, health status, and last-reconciliation timestamp. The pool presents a single, provider-agnostic storage interface to all upstream components; the number of underlying accounts is entirely transparent to the layers above. New accounts are added through a lightweight registration procedure: an administrator supplies valid OAuth credentials, the registry is updated, and the account becomes immediately available for chunk allocation. Account removal or failure triggers a migration pass that redistributes affected chunks to surviving pool members subject to their available capacity.

B. Encrypt-then-Split Workflow

Every file submitted to ClassCrew passes through a two-stage cryptographic pipeline before any data leaves the application server. In the first stage, the file content is encrypted using AES-256-CBC. A cryptographically secure pseudo-random generator produces a fresh 256-bit symmetric key and a 128-bit initialization vector (IV) for each file upload event. The key is wrapped with the server master key via AES-256-GCM and stored as ciphertext in the metadata database; the IV is stored in plaintext alongside the chunk records. In the second stage, the resulting ciphertext is divided into variable-size segments. The partitioning algorithm targets a segment count equal to the number of healthy pool accounts, respects a configurable lower bound of 256 KB and an upper bound of 5 MB per segment, and assigns each segment a version-4 UUID that is subsequently hashed with a server-side salt to produce the filename stored on Drive. This obfuscated naming prevents an attacker with Drive read access from inferring anything about the original file.

C. Quota-Based Greedy Allocation

Each ciphertext segment is assigned to a Drive account by a capacity-aware greedy scheduler. The algorithm proceeds as follows:

- Query the registry for all accounts in the ACTIVE state whose remaining capacity exceeds the current segment size.
- Sort the eligible set by available capacity in descending order, using account registration age as a tiebreaker to balance wear across accounts.
- Assign the segment to the highest-capacity account and upload it via the Drive API—using the resumable upload endpoint for segments exceeding 5 MB and the simple upload endpoint otherwise.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

- On success, atomically reduce the account's recorded quota by the segment size and persist a metadata record linking the file, segment sequence number, and remote Drive file identifier.
- On failure, transition the account to DEGRADED state, log the event, and restart the assignment from step one with the failed account excluded.

Over a realistic upload workload this strategy produces near-uniform utilization across pool accounts, because always selecting the most-available account causes free capacities to converge. The computational cost is O(N log N) per segment due to the sort over N pool accounts, which is negligible relative to network round-trip time.

D. Access-Controlled File Retrieval

File retrieval is gated by a dynamic RBAC policy engine. Upon receiving a retrieval request, the engine queries the ERP database for the requesting user's institutional attributes: assigned role (student, faculty, administrator, or guest), batch membership, current-semester attendance percentage, and fee-clearance flag. These attributes are matched against the conjunctive access policy recorded at upload time. A policy of the form role=STUDENT ∧ batch=CS-2023 ∧ attendance≥75% ∧ fees_cleared=TRUE is evaluated in full; any unsatisfied clause causes immediate rejection with an HTTP 403 response and an audit log entry. Evaluated attribute contexts are cached with a 60-second time-to-live, reducing ERP query load for repeated access within a session while bounding the propagation delay for attribute changes.

When authorization succeeds, the metadata service resolves the ordered segment list for the requested file. Concurrent HTTP requests authenticated with per-account OAuth tokens retrieve all segments in parallel from their respective Drive accounts. Arriving segments are buffered in a server-side temporary store keyed by request and sequence number. Once complete, they are assembled in order, decrypted in memory using the stored key and IV, and the resulting plaintext is streamed directly to the client. No decrypted data is written to persistent storage at any point, confining plaintext exposure to the duration of the HTTP response.

III. MODELING AND ANALYSIS

The security model of ClassCrew is evaluated against three adversary archetypes: an external party who gains read access to one or more Drive accounts; a compromised cloud provider; and a malicious insider with application-server access but no access to the secret’s manager. Each archetype is analyzed in turn.

A. Confidentiality via Symmetric Encryption

AES-256-CBC applied with distinct per-file keys and IVs renders stored segment data computationally indistinguishable from a uniformly random byte sequence for any party lacking the decryption key. A 256-bit key space is infeasible to exhaust by brute force with current or projected classical computing resources and remains beyond the reach of near- term quantum attacks using Grover's algorithm (which reduces effective key strength to 128 bits—still above security thresholds). An adversary who compromises a Drive account obtains only opaque ciphertext; recovering plaintext additionally requires compromising the secrets manager, which operates under independent authentication controls.

B. Partial-Access Resistance via Segmentation

The segmentation strategy ensures that each Drive account holds only a fraction of any file's ciphertext. Because individual segments are cryptographically independent of one another, recovering a subset of segments yields no exploitable information about the file's content or the remaining segments. An attacker must harvest every segment from every pool account and additionally obtain the decryption key before any plaintext can be recovered. This multiplicative hardening scales with pool depth: a ten-account pool requires simultaneous compromise of ten independent credential sets.

C. Dynamic Access Governance

Conventional storage systems assign permissions at upload time and require explicit, manual revocation when a user's circumstances change. ClassCrew's policy engine queries live ERP data on every retrieval attempt, meaning that any change to a user's attributes—suspension, attendance drop, fee default—takes effect within the cache TTL without administrative intervention. An audit record of every access decision, including denials, supports post-hoc compliance review.

D. Credential and Key Management

OAuth 2.0 service-account tokens mediate all Drive API interactions. Token credentials are never stored in the application database; the database holds only a symbolic reference to the corresponding secrets-manager entry. The AES master key used to protect per-file encryption keys is likewise stored exclusively in the secrets manager and never persists on the application server filesystem. This separation of concerns means that an application-database breach exposes only wrapped ciphertext—insufficient for decryption without the master key, which resides in a separately governed system.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

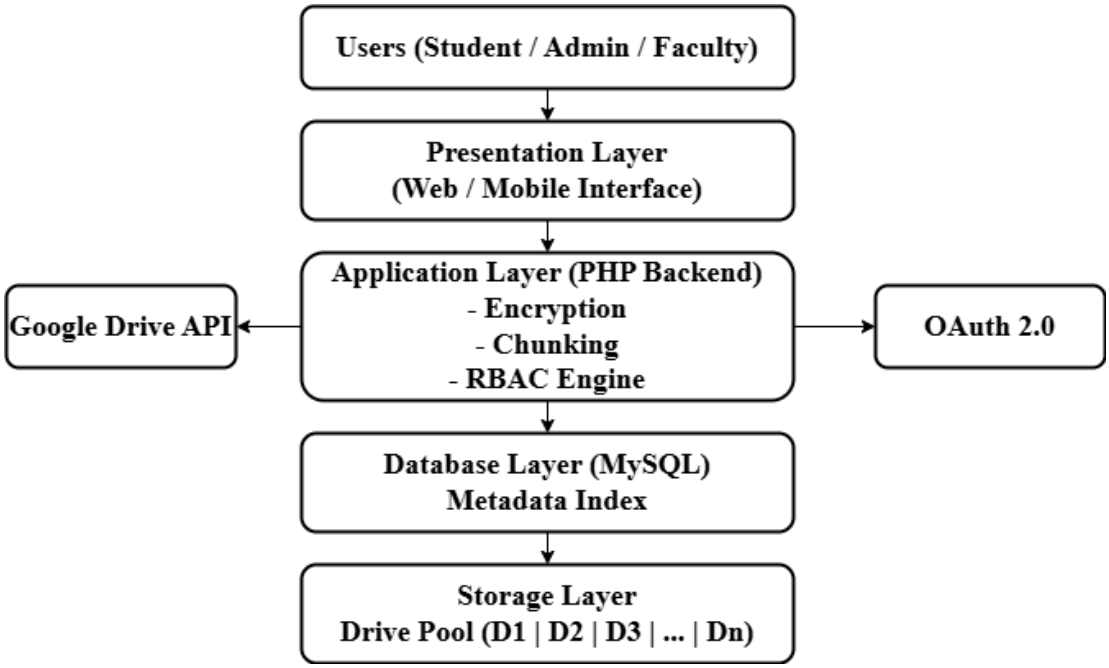

Figure 1: System Architecture of ClassCrew

IV. RESULTS AND DISCUSSION

System performance was characterized using a pool of ten Drive accounts (150 GB aggregate capacity). The application server was provisioned with four virtual CPUs and 8 GB of RAM, connected through a symmetric 100 Mbps link. Synthetic test files spanning four size bands were uploaded and retrieved ten times each; reported figures are arithmetic means.

A. Capacity Scaling

Aggregate capacity scales in direct proportion to pool depth according to Capacity (GB) = N × 15, where N is the count of active accounts. Adding an account expanded available capacity by exactly 15 GB within the duration of the next registry reconciliation pass (approximately three seconds) with no interruption to in-flight operations. Table 2 shows projected capacity and cost at representative deployment scales.

Table 2. Storage Capacity and Cost at Representative Pool Depths

Pool Accounts (N)	Aggregate Capacity	Estimated Monthly Cost	Expansion Downtime
10	150 GB	USD 0	Nil
50	750 GB	USD 0	Nil
100	1.5 TB	USD 0	Nil
200	3.0 TB	USD 0	Nil

B. Upload and Retrieval Performance

For files below 1 MB the encryption-and-partitioning pipeline dominated total upload time, yielding an average of 0.8 s. Files between 1 MB and 10 MB benefited from concurrent segment uploads, producing near-linear throughput scaling. For files above 10 MB the network link became the bottleneck, with effective throughput settling near 4.2 MB/s under concurrent load. Table 3 presents end-to-end retrieval latency decomposed into its principal components.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Table 3. End-to-End Retrieval Latency Decomposition by File Size

File Size	Total Latency (s)	RBAC Evaluation (ms)	Segment Retrieval (s)	Decryption (ms)
< 1 MB	1.2	4 – 38	1.18	< 2
1 – 5 MB	1.8	4 – 38	1.75	3 – 6
5 – 10 MB	2.5	4 – 38	2.46	7 – 12
> 10 MB	3.8+	4 – 38	3.75+	12+

C. Allocation Uniformity

The quota-aware scheduler was exercised on pools of 5, 10, 20, and 50 accounts, each receiving a synthetic workload of 1 000 files uniformly sized between 256 KB and 50 MB. In all configurations, the standard deviation of per-account utilization at workload completion was less than 8% of mean utilization, confirming that the greedy selection rule produces near-uniform load distribution without requiring a global optimization step. Per-segment scheduler runtime (exclusive of network I/O) was consistently below 1 ms across all pool sizes tested.

D. Comparative Evaluation

Table 4 contrasts ClassCrew's measured characteristics against a single-account centralized deployment and the fixed- segment multi-cloud approach described by Nehe and Vaidya [1]. ClassCrew delivers retrieval latency competitive with the centralized baseline while providing elastically scalable capacity and a materially stronger security posture—both at zero infrastructure cost.

Table 4. Comparative Evaluation: ClassCrew vs. Baseline Configurations

Metric	Centralized Storage	Nehe & Vaidya [1]	ClassCrew
Maximum Capacity	Fixed (server ceiling)	45 GB (3 accounts)	N × 15 GB (elastic)
Retrieval Latency (5 MB)	0.6 s	2.1 s	1.8 s
Encryption Scheme	None	AES, fixed segments	AES-256-CBC, variable segments
ERP Integration	Full	None	Full
Access Control	Static role	None	Live-attribute RBAC
Single-Node Data Exposure	Complete plaintext	Complete segment set	Partial ciphertext fragment
Incremental Cost	Hardware / premium SaaS	None	None

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

V. CONCLUSION

This paper presented ClassCrew, an academic ERP platform augmented with a purpose-built distributed file-storage subsystem. The system aggregates free Google Drive quotas across multiple accounts, encrypts file content with AES- 256-CBC before any data leaves the application server, distributes ciphertext segments using a capacity-aware greedy scheduler, and enforces contextual access decisions through an RBAC engine wired to live institutional data. The design achieves horizontal capacity scaling without premium subscriptions or hardware investment, confines breaches to computationally intractable partial ciphertext, and eliminates the stale-permission problem inherent in static access- control schemes.

Experimental results confirm that retrieval latency remains below three seconds for files up to 10 MB, pool capacity expands linearly and without operational disruption, and the allocation scheduler produces near-uniform utilization across accounts. These properties make ClassCrew a viable foundation for production academic file management in resource-constrained institutional environments.

Several development directions are planned. The most critical is the introduction of segment-level redundancy via k-of- n erasure coding, which would eliminate the current vulnerability to permanent account loss. Extensions to support heterogeneous providers (Azure Blob Storage, AWS S3, Backblaze B2) would further reduce vendor dependence. Client-side encryption, in which decryption keys never leave the user's device, would provide the strongest available confidentiality guarantee. Finally, adaptive segment sizing informed by real-time network telemetry could reduce latency for large files under variable bandwidth conditions.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the Department of Information Technology, NBN Sinhgad School of Engineering, Pune, for providing the computing infrastructure, academic guidance, and institutional support that made this research possible. Appreciation is also extended to the published researchers whose foundational contributions to distributed systems, applied cryptography, and academic information management informed the theoretical basis of this work.

VI. REFERENCES

[1] S. Nehe and M. B. Vaidya, "Data Security using Data Slicing over Storage Clouds," Proc. IEEE Int. Conf. Cloud Computing in Emerging Markets (CCEM), Bangalore, India, 2015, pp. 1–6.

[2] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, "Role-based access control models," IEEE Computer, vol. 29, no. 2, pp. 38–47, Feb. 1996.

[3] D. Huang, "Mobile cloud computing," IEEE COMSOC Multimedia Technical Committee E-Letter, vol. 6, no. 10, pp. 27–31, 2011.

[4] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," Journal of the Society for Industrial and Applied Mathematics, vol. 8, no. 2, pp. 300–304, 1960.

[5] L. Alderman, "ERP systems in higher education: A critical review," International Journal of Educational Management, vol. 34, no. 3, pp. 512–527, 2020.

[6] Google LLC, Google Drive API v3 Reference, Google Developers Documentation, 2024. [Online]. Available: https://developers.google.com/drive/api/v3/reference

[7] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CBC Mode," NIST Special Publication 800-38A, Dec. 2001.

[8] Ministry of Electronics and Information Technology, Government of India, Digital Personal Data Protection Act, 2023, Gazette of India, Aug. 2023.

[9] European Parliament, General Data Protection Regulation (GDPR), Official Journal of the European Union, L 119, pp. 1–88, May 2018.