

Secure Role-Based Alumni Digital Archive and Networking Platform for Polytechnic Education

Dr. Brindha S¹, Ms. Sountharia I N², Mr. Nikil Vardhan V³, Mr. Jayakrishna J⁴, Mr. Logan Jerry J⁵,
Ms. Nikitha N⁶

¹Head of the Department, Computer Networking, PSG Polytechnic College, Coimbatore

²Lecturer, Computer Networking, PSG Polytechnic College, Coimbatore

³⁻⁶Students, Computer Networking, PSG Polytechnic College, Coimbatore

ABSTRACT

Institutional alumni records are often distributed across spreadsheets, paper archives, informal communication channels, and isolated departmental records, making it difficult to maintain verified information, support alumni networking, and provide controlled access. This paper presents the design and implementation of a web-based Alumni Digital Archive developed for PSG Polytechnic College, Coimbatore, initially serving the Computer Networking and Artificial Intelligence and Machine Learning departments. The platform centralizes alumni records in a MySQL database and provides separate authenticated portals for the Head of Department, Faculty Advisor, Faculty, and three student groups. A role-based access-control model prevents users from obtaining capabilities outside their assigned portal. Administrative users can manage and verify alumni records, while students receive a restricted, read-only directory containing only information required for academic and professional networking. The architecture applies backend SQL projection so protected personal information is excluded before student data is serialized. The system also incorporates moderated registration, reporting, auditability, session persistence, database-backed statistics, bulk Excel import, and profile-photo ingestion. Particular attention is given to refresh persistence and non-destructive read-only initialization so browser reloads do not erase displayed data or overwrite database records. The resulting architecture provides a practical institutional framework for preserving alumni information while balancing accessibility, verification, privacy, and operational control.

Keywords— alumni management, digital archive, role-based access control, MySQL, privacy-preserving directory, institutional information system, bulk data import, web application security.

I. INTRODUCTION

Alumni information is an important institutional asset because it connects current students, graduates, faculty members, academic departments, and professional opportunities. Maintaining an accurate alumni directory becomes increasingly difficult as the number of graduates grows. Records may be collected through registration forms, spreadsheets, departmental documents, and manual updates. Without a centralized system, duplicate records, outdated employment information, inconsistent status values, missing photographs, and uncontrolled access can reduce the usefulness of the archive.

The PSG Polytechnic College Alumni Digital Archive addresses this operational problem through a centralized web application designed around institutional roles. The platform preserves alumni records while making verified information useful for networking and academic engagement. Instead of exposing the same interface to every authenticated user, the system separates capabilities according to portal role. Administrative portals are intended for controlled institutional management, whereas student portals are deliberately simplified and read-only.

Authentication and authorization are treated as separate concerns. A valid HOD credential must not become valid merely because it is entered through a student portal. The requested portal context is therefore conveyed to the backend during login, and the server validates the credential together with the permitted portal category. The resulting session retains

the authenticated identity and portal context across page refreshes.

A second requirement is privacy at the backend boundary. Client-side hiding is insufficient because a browser can inspect network responses. For student requests, the server selects an explicit set of permitted columns from MySQL and excludes protected fields before JSON serialization. The student front sheet is intentionally minimal, displaying Identity, Roll No, Department, Year of Passing, Status, and Actions, with a read-only View Profile operation for additional permitted details.

The remainder of this paper reviews related approaches, defines the problem and research gap, presents the proposed architecture and methodology, describes implementation details, discusses validation considerations, and concludes with limitations and future enhancements.

II. LITERATURE REVIEW

Role-based access control (RBAC) provides a structured method for assigning permissions through organizational roles rather than individual ad hoc rules. This model is appropriate for institutional systems because responsibilities such as department administration, faculty advising, faculty access, and student access naturally correspond to different permission sets. RBAC research emphasizes role hierarchies, separation of responsibilities, and controlled privilege assignment [1], [2].

Privacy-preserving data architectures demonstrate that access control should be enforced at multiple layers. Presenting a restricted interface does not guarantee restricted data because unauthorized fields may still be present in API responses. Secure application guidance therefore encourages server-side authorization and minimization of sensitive data exposure [3], [4]. The proposed alumni platform follows this principle by using explicit SQL projections for student-facing data rather than retrieving complete rows and removing fields only in the browser.

Digital alumni systems commonly emphasize directory management, communication, networking, and institutional engagement. A practical institutional deployment, however, must also address verification

workflows, record lifecycle management, bulk import, data persistence, and differentiated user privileges. These requirements are especially important when a directory contains personal information and students should discover professional connections without receiving unnecessary private attributes.

Database-backed web applications also require consistency between browser state and persistent storage. A common failure mode in client-heavy systems is treating an in-memory array as the authoritative database and synchronizing it automatically during initialization. If an empty client state is synchronized, persistent rows may be overwritten or appear to disappear after refresh. The present work therefore treats MySQL as the source of truth and designs page initialization as a read-only operation.

III. PROBLEM STATEMENT

The central problem addressed by this work is the absence of a unified and access-controlled institutional mechanism for maintaining, verifying, preserving, and selectively sharing alumni information. A system that simply stores alumni rows is insufficient because the same information cannot safely be exposed to HODs, faculty advisors, faculty members, and students with identical permissions.

The problem has four major dimensions. First, authentication must be portal-aware so credentials assigned to one portal cannot be reused to obtain another portal's capabilities. Second, authorization must be enforced on the backend rather than inferred from front-end visibility. Third, student access must expose only information necessary for networking and academic use while excluding sensitive personal and administrative fields. Fourth, persistent data must remain stable during browser refreshes, with database-backed counts and records reloaded without destructive synchronization.

The system must also support institutional workflows such as alumni registration, moderation, profile viewing, reporting, bulk Excel import, profile-photo handling, and database-backed statistics without compromising isolation of existing administrative workflows.

IV. RESEARCH GAP

Existing alumni directory implementations frequently focus on record storage and search but give less attention to strict portal separation and backend data minimization. A student-facing directory can appear read-only while still returning fields that students should never receive. Similarly, a multi-role login screen may visually distinguish portals while accepting the same credentials across them if portal authorization is not bound to the server-side session.

A second gap concerns persistence behavior. Client-side caches are useful for responsiveness, but using them as a write-through source of truth during page loading can create destructive synchronization bugs. An institutional archive requires the opposite behavior: page loading should read from the database, and mutations should occur only through explicit authorized actions.

The proposed work addresses these gaps through portal-aware authentication, backend RBAC, explicit SQL projections, role-aware statistics, read-only initialization, and regression tests that inspect raw API responses rather than relying only on visible user-interface behavior.

V. PROPOSED SYSTEM ARCHITECTURE

The proposed Alumni Digital Archive follows a layered web architecture consisting of browser interfaces, authentication and authorization middleware, REST-style backend services, and a centralized MySQL database. The architecture separates presentation from institutional data services while maintaining a clear authorization boundary between portal categories.

PORTAL	LAYER
HOD • Faculty Advisor • Faculty • Student 1 • Student 2 • Student 3	
AUTHENTICATION	LAYER
Portal-aware login • Session identity • Role and portal validation	
AUTHORIZATION	LAYER
Admin RBAC • Student read-only policy • Protected API endpoints	
APPLICATION	SERVICES
Alumni • Registration • Moderation • Reports • Statistics • Bulk Import • Profiles	

DATA	LAYER
MySQL alumni records • verification state • audit information • profile-photo references	

Fig. 1. Layered architecture of the proposed PSG Alumni Digital Archive.

The portal layer provides six dedicated entry points. HOD, Faculty Advisor, and Faculty portals belong to the administrative category, while the three student portals belong to the student category. The backend receives the requested portal type during authentication and validates it against the authenticated account. This prevents a credential issued for one portal category from silently becoming authorization for another category.

The authorization layer protects administrative APIs such as full alumni data access, mutation operations, pending submissions, audit logs, settings, and recycle-bin functions. Student requests are routed to dedicated student endpoints. These endpoints execute explicit SQL projections that select only approved networking fields. Sensitive attributes such as mobile number, phone, address, date of birth, blood group, marital status, deletion metadata, and approval metadata are excluded at query time.

The data layer remains the authoritative source for alumni records. Statistics are calculated directly from MySQL so current counts do not depend on a browser's in-memory state. Page refreshes therefore reconstruct the interface from persistent data instead of attempting to recreate the database from cached client arrays.

VI. METHODOLOGY

Development follows a security-first, role-aware methodology. Requirements are decomposed into portal capabilities, data visibility rules, and mutation privileges. Each role is mapped to a defined set of API permissions. Authentication lifecycle code is separated from dashboard presentation code so login changes do not unintentionally modify administrative views.

A. Portal-Aware Authentication

Each login request carries an explicit portal identifier such as HOD, Advisor, Faculty, Student 1, Student 2, or Student 3. The backend validates the supplied credentials and verifies that the authenticated

account is authorized for the requested portal category. The server establishes a session containing the user identifier, role, portal type, and authentication timestamp. These values are retained when the browser requests the application again after refresh.

B. Role-Based Authorization

Authorization is enforced on the server. Administrative operations require an administrative role, while student sessions are restricted to student endpoints. The front end hides controls that students should not use, but the security model does not rely on that hiding. Direct requests to protected administrative endpoints are rejected by middleware even when a student attempts to construct such requests manually.

C. Backend Data Projection

For the student directory, the server performs a SELECT statement containing only approved fields. The permitted projection includes identity and networking information such as name, roll number, department, year, admission year, status, organization, job role, education details, LinkedIn, email, profile photograph reference, and verification state where appropriate. Protected PII and moderation fields are not selected.

D. Role-Aware Statistics

Statistics are permission-aware. Administrative sessions may receive institutional management metrics such as total alumni, male and female counts, and pending submissions. Student sessions receive only statistics intended for the student directory, such as total alumni, male count, and female count. Moderation, audit, and recycle-bin metrics are not returned to students.

E. Read-Only Refresh and Persistence

Browser initialization is deliberately read-only. The storage service fetches the appropriate backend dataset and statistics after authentication. It does not call a destructive synchronization endpoint merely because a page has loaded. Mutation endpoints are invoked only after an explicit authorized action. Network failures are represented as loading or error states rather than silently replacing an existing dataset with an empty array.

F. Bulk Import and Profile Photographs

The bulk import workflow supports structured alumni data together with profile photographs. Before photograph extraction is implemented, the database schema must be verified to contain an appropriate photo field or photo reference. During import, the workbook is parsed together with its embedded media package; the importer associates each extracted photograph with the corresponding alumni row using a stable identity such as roll number or another validated key. The photograph is stored according to the application's file-storage strategy, while the database retains the resulting reference.

G. Data Lifecycle and Identifier Integrity

Alumni records use database-generated identifiers. Deleting rows does not necessarily reset an auto-increment counter, so later records may receive higher identifiers. This is normal database behavior and avoids reusing identifiers that may already appear in audit trails or references. The identifier represents record identity rather than the current number of alumni in the system.

VII. IMPLEMENTATION

The implementation uses a web-based client and an Express server backed by MySQL. The front end is divided into landing, authentication, registration, administrative, and student experiences. Authentication is isolated in a dedicated portal-login module so dashboard code is not responsible for deciding identity or portal authorization.

The server exposes dedicated routes for the six portal entry points. After authentication, API access is governed by session state rather than a URL alone. Middleware identifies the authenticated role and portal category before protected operations are processed. This prevents a user from gaining additional access simply by changing a path, query parameter, or client-side variable.

The administrative interface supports alumni information management, detailed reports, pending submissions, recycle-bin functions, settings, and audit logging according to the assigned role. These views are separated from the student experience. The student interface is intentionally minimal: the front sheet displays Identity, Roll No, Department, Year of Passing, Status, and Actions. The only profile

operation available to a student is View Profile. Editing, adding, deleting, importing, exporting, moderation, audit, recycle-bin, settings, and profile-edit controls are unavailable.

The View Profile operation provides a controlled expansion of the directory row. It is not an edit form. The profile view displays only additional information approved for student access, such as academic background, current professional information, organization, role, higher education, LinkedIn, email, and profile photograph where available. Sensitive personal information remains excluded from the student API.

Database-backed statistics are rendered independently from the current table array. Refreshing a page therefore requests current totals from MySQL rather than deriving them from an empty or stale cache. This addresses the failure mode in which alumni records or the total count disappeared after refresh.

The alumni status field is treated as data rather than an automatic synonym for employment. A record should not be labelled employed unless the underlying information or an authorized administrative action supports that status. The application should preserve explicit status values instead of silently assigning employment.

VIII. SYSTEM VISUALIZATION

The landing page presents the archive as an institutional platform rather than merely a login screen. It introduces the college identity, the Alumni Digital Archive, access to the six portals, departmental focus on Computer Networking and AI and ML, workflow explanation, privacy and security principles, and database-backed statistics. The visual hierarchy uses institutional navy and heritage gold with restrained animation and responsive layouts.

The student portal uses a deliberately simpler information architecture. The front sheet prioritizes the six required columns and a View Profile action. Profile information is presented in a separate read-only view so the directory remains scannable while students can still inspect approved professional and academic details.

The administrative experience remains visually and functionally separated from the student experience.

The security model is reflected in both API behavior and interface behavior: users see controls appropriate to their role, while the backend independently enforces the same restrictions.

IX. RESULTS AND DISCUSSION

Evaluation is organized around functional correctness, persistence, authorization, privacy, and data integrity rather than visual appearance alone. The core regression scenario begins by recording the active alumni count directly from MySQL using the condition `isDeleted = 0`. Each of the six portal types is then authenticated independently and tested against the API operations permitted to that role.

For refresh persistence, the authenticated session is retained while the browser requests session information, portal-specific data, and statistics again. The expected behavior is that the same database-backed count and records are reconstructed after refresh. Page loading must not invoke destructive synchronization.

Security verification inspects raw JSON responses rather than relying solely on rendered screens. Student responses must not contain mobile, phone, address, date of birth, blood group, marital status, deletion metadata, or administrative approval metadata. Direct requests from a student session to administrative data and mutation endpoints must return a forbidden response.

Authentication testing verifies portal isolation. A credential associated with the HOD portal must not grant HOD access when submitted through the first-year student portal. Credentials belonging to student portals must not be accepted as administrative credentials. Thus the portal identifier participates in authorization rather than acting only as a visual label.

Bulk import validation includes both spreadsheet records and embedded profile photographs. The test procedure checks that the workbook contains the expected Profile Photo field or, where the implementation uses embedded workbook media, that the importer can extract image objects and associate them with the correct alumni records. Imported photographs are then verified through the stored reference and profile display.

No fabricated accuracy percentage is claimed here. The final paper should report measured pass/fail

counts and database counts from the actual production test run. This keeps the evaluation evidence-based rather than inventing performance results.

A. Verification Matrix

Test Area	Expected Behavior	Evidence
Portal authentication	Each credential works only in its authorized portal	Session role + portal type
Refresh persistence	Records and statistics remain after F5	MySQL count + API response
Student privacy	Protected PII absent from raw JSON	Response-field inspection
Student permissions	Read-only directory and View Profile only	UI + endpoint authorization
Admin permissions	Authorized management functions remain available	Regression suite
Bulk import	Rows and profile photos are associated correctly	Imported row + image reference
Status integrity	Status reflects supplied/verified information	Database inspection

Table I. Functional, security, persistence, and import verification matrix.

X. ADVANTAGES AND IMPACT

The principal advantage is controlled institutional sharing. Alumni information becomes useful to students and faculty without requiring the institution to expose every stored attribute to every authenticated user. The architecture separates administrative management from student networking and reduces the attack surface of the student interface.

A second advantage is database-backed persistence. By treating MySQL as the source of truth and making initialization read-only, the system avoids a class of client-cache synchronization errors. Refreshing a page

becomes a retrieval operation rather than a data mutation.

The platform improves the operational lifecycle of alumni records. Registration, verification, profile viewing, reporting, bulk import, profile photographs, and archival management can be handled through one system. Database-generated identifiers remain stable even when records are deleted, supporting traceability.

The restricted student directory provides a practical bridge between current students and verified alumni. View Profile allows useful professional and academic information to be discovered without turning the student portal into a full administrative information-management interface.

XI. LIMITATIONS

The platform is an institutional application rather than a general-purpose alumni social network. Its effectiveness depends on the quality and completeness of submitted alumni information. Incorrect status, missing photographs, duplicate records, or outdated professional details require verification and periodic maintenance.

Security also depends on secure deployment, session handling, credential storage, transport security, dependency maintenance, rate limiting, and operational monitoring. Backend field projection reduces accidental disclosure but is not by itself a complete security program.

Bulk profile-photo import introduces an additional processing path because spreadsheet-embedded images are workbook media objects rather than ordinary scalar cell values. Robust identity matching and file validation are required before production import.

The current evaluation framework emphasizes functional and security regression testing. A larger longitudinal study involving real alumni and students would be required to quantify networking outcomes, usability, adoption, and institutional value.

XII. FUTURE ENHANCEMENTS

Future development can extend the platform with stronger duplicate detection, alumni verification reminders, richer search and filtering, controlled alumni-to-student mentorship workflows, institutional

email integration, and configurable departmental policies without weakening existing role boundaries.

The bulk-import pipeline can be enhanced with a dedicated media extraction stage that validates image type, dimensions, file size, and identity mapping before committing records. An import preview and row-level error report would allow administrators to correct malformed records without discarding a complete workbook.

The platform can also introduce privacy-aware analytics for administrators, such as batch trends, department distribution, organization categories, and alumni engagement summaries. Student analytics should remain aggregated and should not expose moderation or sensitive personal information.

Infrastructure improvements may include automated backups, structured migration scripts, deployment health checks, stronger session invalidation, rate limiting, centralized logging, and security testing aligned with recognized application-security standards.

XIII. CONCLUSION

This paper presented the design and implementation of a secure Alumni Digital Archive for PSG Polytechnic College that combines centralized MySQL storage, portal-aware authentication, role-based authorization, privacy-preserving student access, institutional verification, reporting, bulk import, and persistent database-backed statistics. The system is designed around a simple principle: authenticated access does not imply unrestricted access.

The architecture separates six portal experiences while keeping authorization decisions on the backend. Administrative users retain controlled management capabilities, whereas students receive a minimal read-only directory with a View Profile operation and only additional details approved for networking. Explicit SQL projection prevents protected personal and administrative fields from entering student JSON responses. Database-backed statistics and read-only initialization ensure browser refreshes reconstruct the interface from persistent records rather than erasing or overwriting them.

The work demonstrates that an institutional alumni platform can move beyond a basic directory by

treating identity, authorization, privacy, persistence, and data lifecycle as first-class engineering requirements. With continued validation, data-quality governance, and production security hardening, the platform can serve as a durable digital foundation for alumni preservation and academic networking.

REFERENCES

- [1] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996.
- [2] D. F. Ferraiolo, D. R. Kuhn, and R. Chandramouli, *Role-Based Access Control*, 2nd ed. Norwood, MA, USA: Artech House, 2007.
- [3] OWASP Foundation, *OWASP Application Security Verification Standard (ASVS)*, Version 4.0.3, 2021.
- [4] OWASP Foundation, *OWASP Top 10: The Ten Most Critical Web Application Security Risks*, 2021.
- [5] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 3rd ed. Indianapolis, IN, USA: Wiley, 2020.
- [6] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*, 7th ed. New York, NY, USA: McGraw-Hill, 2019.
- [7] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA, USA: Pearson, 2015.
- [8] E. Bertino and R. Sandhu, "Database Security—Concepts, Approaches, and Challenges," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 1, pp. 2–19, 2005.