International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 11, November 2025)

Examsuite: Empowering Institutions with Centralized Exam Management

R Pradhyumna[1], Arun Kumar H P[2], Kushal Gowda H M[3], Madhu M S[4], Dr. Sharath Kumar Y H[5]

[1,2,3,4]*Department of Information Science and Engineering, Maharaja Institute of Technology, Mysore, Karnataka, India*
[2]*Professor and Head of Department, Information Science and Engineering, Maharaja Institute of Technology, Mysore, Karnataka, India*

***Abstract*—Traditional examination management in educational institutions involves manual paper handling, security vulnerabilities, and lack of accountability mechanisms. This paper presents ExamSuite, a comprehensive cloud-native examination management system designed for Maharaja Institute of Technology, Mysore. The system implements role-based architecture supporting four distinct user roles—Faculty, Controller of Examinations, Board of Examiners, and Principal—each with tailored access controls. Built us- ing React and Supabase with PostgreSQL, ExamSuite lever- ages Row-Level Security policies for database-level access control. The system features time-bound security for sen- sitive operations, comprehensive audit trails, and hierarchi- cal file management. Performance optimizations include React Query caching, lazy loading, and database indexing. ExamSuite eliminates manual workflows, enhances secu- rity through multi-layered authentication and authorization, and provides real-time status tracking throughout the examination paper lifecycle. The system addresses institution-specific requirements while maintaining security and scalability for production deployment.**

***Keywords*—Examination Management System, Role- Based Access Control, Cloud-Native Architecture, Row-Level Security, Academic Workflow Automation**

I. Introduction

Educational institutions face significant challenges in man- aging examination papers throughout their lifecycle. Traditional manual processes involve physical document handling, lack centralized tracking mechanisms, present security vulnerabilities with confidential examination materials, and require time-consuming approval workflows across multiple administrative levels. The absence of comprehensive audit trails makes accountability difficult, while paper-based systems struggle to maintain version control and ensure document integrity.

Maharaja Institute of Technology, Mysore operates across multiple departments with distinct examination schedules, requiring coordination between faculty members, administrative staff, and institutional leadership. The existing manual system created bottlenecks during peak examination periods, delayed approvals, and posed risks of unauthorized access to sensitive examination materials. These challenges motivated the development of ExamSuite, a purpose-built digital solution addressing institution-specific requirements while maintaining scalability and security.

ExamSuite provides a web-based platform that digitizes the entire examination paper workflow from faculty submission through administrative approvals to secure principal download. The system enforces role-based access control at both application and database levels, implements time-bound security for sensitive operations, maintains comprehensive audit trails for compliance, and organizes examination papers using hierarchical folder structures based on academic parameters.

This paper presents the system architecture, implementation details, security mechanisms, and performance optimizations of ExamSuite. The solution demonstrates how modern cloud-native technologies can address complex institutional workflows while maintaining security, scalability, and usability.

II. System Architecture

ExamSuite follows a layered client-server architecture combining separation of concerns with distributed system benefits. The architecture consists of four primary layers: Presentation Layer, Application Layer, Data Access Layer, and Infrastructure Layer.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 11, November 2025)

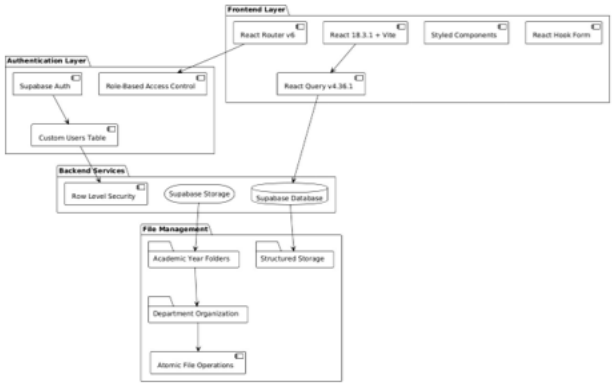

Figure 1: Layered System Architecture

A. Frontend Layer

The presentation layer utilizes React 18 with concurrent rendering features for improved user interface responsiveness. Vite serves as the build tool, providing fast development server startup and optimized production builds through tree-shaking and code splitting. React Router v6 handles client-side routing with protected routes enforcing authentication requirements. Styled Components provides component-scoped styling, eliminating CSS conflicts and enabling theme-based design consistency.

The application implements lazy loading for all major page components (Faculty, CoE, BoE, Principal dashboards) to reduce initial bundle size and improve load times. React Query v4 manages server state with automatic caching, background refetching, and optimistic updates. Custom hooks encapsulate business logic, separating data fetching concerns from presentation components.

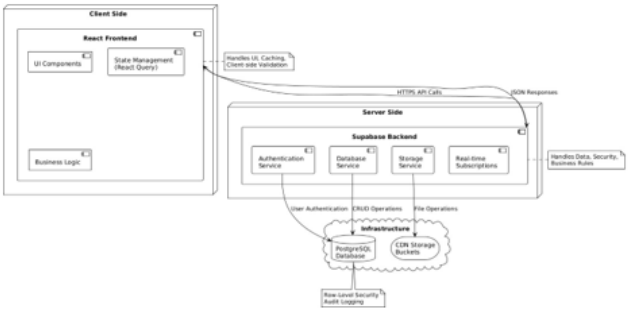

Figure 2: Client-Server Architecture of ExamSuite

B. Authentication Layer

Authentication leverages Supabase Auth providing JWT-based session management with automatic token refresh. The system implements dual-table authentication: Supabase's native auth.users table handles credentials and session tokens, while a custom users table stores institutional metadata including employee ID, department affiliation, and assigned role.

This separation enables institutional data queries without exposing authentication internals while maintaining referential integrity through foreign key constraints linking auth user id to Supabase's authentication system. Single-session enforcement prevents concurrent logins, and rate limiting restricts authentication attempts to 30 requests per 5-minute window per IP address.

C. Backend Services

The backend utilizes Supabase as a Backend-as-a-Service platform, providing PostgreSQL database with ACID com- pliance, CDN-backed file storage with automatic scaling, and real-time subscriptions for live updates. All database interactions use parameterized queries through Supabase's JavaScript client, preventing SQL injection vulnerabilities.

Row-Level Security policies enforce access control at the database level, ensuring unauthorized users cannot access restricted data even if application-level checks fail. The PostgreSQL database includes indexes on frequently queried columns (status, department name, uploaded by, exam datetime) to optimize query performance.

D. File Management

Examination papers are stored in Supabase Storage with hi- erarchical organization following the structure: papers/Academic Year/Department/Semester/Subject/. This organiza- tion facilitates efficient retrieval, maintains logical grouping, and supports atomic file operations with rollback capabili- ties. File versioning tracks modifications when the Board of Examiners uploads corrected versions, maintaining audit trails of all document changes.

III. Role-Based Access Control

ExamSuite implements a four-tier role hierarchy with dis- tinct access patterns and permissions enforced through database-level security policies.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 11, November 2025)

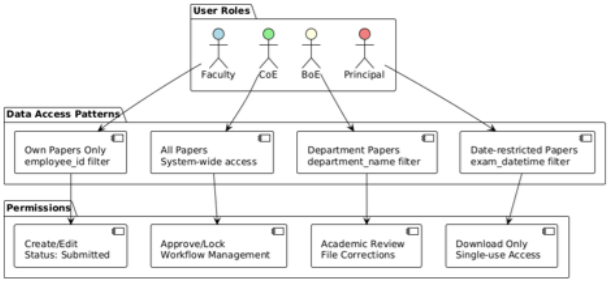

Figure 3: Role-Based Access Control Architecture

```
CREATE POLICY "faculty_own_papers"
  ON exam_papers
  FOR ALL USING (uploaded_by = auth.uid());
```

A. Faculty Role

Faculty members access only examination papers they have uploaded, enforced by the RLS policy:

Faculty can submit new papers with question paper (QP) and answer scheme files, edit papers with status ”Submitted”, track approval progress through workflow stages, and view comprehensive status history. The interface provides a personalized dashboard showing only owned papers with filtering by semester, subject, and academic year.

B. Controller of Examinations (CoE) Role

The CoE has system-wide visibility across all departments and papers. Responsibilities include first-level approval of submitted papers, user management and system adminis- tration, CSV bulk operations for subjects and examination schedules, and locking papers for principal download. The CoE dashboard implements advanced filtering by depart- ment, status, semester, and academic year, with search ca- pabilities across subject codes and names. Approval actions update paper status to ”CoE-approved” and record the ap- prover’s employee ID for audit purposes.

C. Board of Examiners (BoE) Role

BoE access is restricted to their department’s papers with status beyond ”Submitted”, implemented through the policy:

```
CREATE POLICY "boe_department_papers"
  ON exam_papers
  FOR ALL USING (
    department_name = get_user_department() AND
    status != 'Submitted'
  );
```

BoE members perform academic review of examination papers, upload corrected versions maintaining file version history, provide second-level approval advancing status to ”BoE-approved”, and ensure academic quality standards. The system tracks all file modifications with timestamps and uploader identification.

D. Principal Role

Principal access implements time-bound security restricting visibility to papers scheduled for the current date only:

```
CREATE POLICY "principal_today_papers"
  ON exam_papers
  FOR SELECT USING (
    status IN ('Locked', 'Downloaded') AND
    DATE(exam_datetime) = CURRENT_DATE
  );
```

The principal interface displays papers in a matrix format organized by subject and examination time. Download operations are single-use: once downloaded, papers are marked as ”Downloaded” with timestamp recording. Session-based lockout using sessionStorage prevents multiple downloads of the same paper within a single session, providing additional security for sensitive examination materials.

IV. Security Implementation

ExamSuite employs defense-in-depth security with multiple protective layers.

A. Authentication Security

Two-factor authentication support through Supabase Auth enhances login security. JWT tokens with automatic re- fresh maintain secure sessions without requiring frequent re- authentication. Session restoration on page refresh uses se- cure cookie storage with httpOnly and secure flags. CSRF protection is enforced through Supabase security headers validating request origins.

B. Authorization Mechanisms

Authorization operates at three levels: Frontend route pro- tection prevents unauthorized navigation, React components conditionally render based on user roles, and database RLS policies provide final access enforcement. This layered approach ensures security even if frontend checks are by- passed.

Custom PostgreSQL functions get current employee id() and get user department() retrieve authenticated user context for policy evaluation. All policies deny access by default, requiring explicit permission grants.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 11, November 2025)

C. File Security

Structured storage paths prevent unauthorized file access by encoding academic hierarchy in folder names. Public URLs for file downloads are generated on-demand with time-limited validity. File upload operations use atomic transactions: if metadata insertion fails, uploaded files are automatically deleted, maintaining database-storage consistency.

D. Audit Logging

All database operations are logged for 30 days by default through Supabase's audit system. Paper status transitions record timestamps and approver identification. File operations track upload, modification, and download events with user attribution and IP addresses. This comprehensive logging supports compliance requirements and security incident investigation.

V. Implementation Details

A. Technology Stack

The complete technology stack includes:

- Frontend: React 18, React Router v6, React Query v4, React Hook Form v7, Styled Components v6, React Hot Toast v2, Date-fns v2, and React Icons v4.
- Backend: Supabase (PostgreSQL, Row-Level Security, Authentication, Storage), and Papa Parse for CSV op- erations.
- Development Tools: Vite, ESLint, Prettier, and VS Code.

B. Database Schema

The users table stores institutional data with columns for UUID primary key, unique employee ID, username, reference to auth.users, department name, role enum constraint, and creation timestamp.

The exam papers table manages examination metadata including UUID primary key, subject code and name, semester and academic year, department name, file URLs for QP and scheme, file type indicators, storage folder path, uploader employee ID, status enum (Submitted, CoE-approved, BoE- approved, Locked, Downloaded), download flag and times- tamp, approver employee ID, examination date/time, and creation/update timestamps.

C. API Design

The application organizes API functions by role: apiAuth.js for authentication operations (signup, login, getCurrentUser, fetchUserData), apiFaculty.js for faculty operations (cre- ateEditPapers, getPapers), apiCoE.js for CoE operations (getPapers with system-wide access, approvePaper, lockPa- per, manageUsers), apiBoE.js for BoE operations (getPapers with department filter, uploadScrutinizedFiles, approvePa- per), and apiPrincipal.js for principal operations (getPapers with date filter, downloadPaper).

All API functions return promises resolved by React Query hooks, enabling automatic retry logic, optimistic updates, and cache invalidation strategies.

D. Component Architecture

Components follow a hierarchical organization: Page com- ponents serve as layout containers with no business logic, feature components implement domain-specific functional- ity using custom hooks, UI components provide reusable styled elements, and custom hooks encapsulate data fetch- ing and business logic.

This separation of concerns enables independent testing, parallel development by multiple team members, and easy refactoring of business logic without affecting UI components.

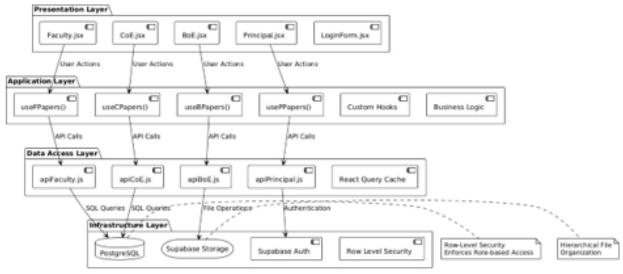

Figure 4: Component Architecture of the Presentation Layer

VI. Performance Optimization

Performance optimizations target frontend rendering, net- work efficiency, and database query speed.

A. Frontend Optimization

Code splitting divides the application into lazy-loaded chunks per route, substantially reducing initial bundle size.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 11, November 2025)

React Query implements a 5-minute stale time for cached data, preventing unnecessary refetches. Prefetching strate- gies load adjacent pagination pages during idle time, en- abling instantaneous navigation.

React 18's concurrent rendering prioritizes user interactions over background updates, maintaining responsiveness during heavy data processing. Memoization using useMemo and useCallback prevents unnecessary re-renders of expensive components.

B. Database Optimization

Strategic indexing on frequently queried columns signifi- cantly reduces query execution time. Pagination limits result sets to 10 records per page, minimizing data transfer and ren- dering overhead. Composite indexes on status and creation timestamp enable efficient sorting of filtered results.

The query planner utilizes index-only scans for common queries, avoiding unnecessary table access. Connection pooling through Supabase manages database connections efficiently under concurrent load.

C. Network Optimization

Supabase Storage integrates with CDN for edge caching of examination paper files with 3600-second cache headers. Gzip compression reduces JSON response sizes. HTTP/2 multiplexing enables parallel resource loading without connection overhead.

VII. Challenges And Solutions

A. Role-Based Access Control Implementation

Implementing granular access control across four distinct roles with overlapping yet differentiated permissions pre- sented significant complexity. The solution combined Su- pabase Row-Level Security policies for database-level en- forcement with frontend component guards for user experi- ence optimization. This dual-layer approach ensured secu- rity while maintaining application performance.

B. Time-Bound Download Security

The principal role required access to examination papers only on the scheduled examination date with single-use download restrictions. The implementation combined PostgreSQL date filtering in RLS policies with JavaScript sessionStorage for per-session download tracking. This approach balanced security requirements with user experience, preventing accidental re-downloads while allowing legitimate recovery from interrupted downloads.

C. System Architecture Design

Designing an enterprise-grade architecture as undergraduate students required extensive research and guidance from the Head of Department. The team adopted layered and client- server patterns after evaluating alternatives, benefiting from faculty mentorship in architectural decision-making. This approach resulted in a maintainable, scalable system suitable for institutional deployment.

VIII. Results And Discussion

ExamSuite has been successfully developed as a production- ready system for institutional deployment. The system demonstrates enterprise-level development practices follow- ing industry-standard architectural patterns. Technical vali- dation by faculty advisors confirmed the system's readiness for deployment, with architectural decisions and security im- plementations meeting institutional requirements.

The system comprises 45+ React components totaling approximately 15,000 lines of well-organized code, with 25+ API service functions, 6 core database tables, comprehensive Row-Level Security policies, and 15+ custom hooks for business logic encapsulation.

A. System Capabilities

ExamSuite eliminates manual paper handling through com- plete digital workflow, streamlines approval processes through automated routing and real-time notifications, en- sures security through multi-layered access control and com- prehensive audit logging, provides real-time status tracking across all workflow stages, and supports bulk operations for administrative efficiency.

B. Architecture Benefits

The layered architecture enables independent development and testing of components, isolated error handling prevent- ing cascade failures, and parallel development with minimal team conflicts. The client-server pattern allows indepen- dent scaling of resources, stronger security through server- side business rule enforcement, and simplified maintenance through clear separation of concerns.

C. Security Validation

Database-level RLS policies provide defense-in-depth secu- rity that cannot be bypassed through application vulnerabilities.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 11, November 2025)

Time-bound access controls ensure examination papers are visible only during appropriate time windows. Comprehensive audit logging supports compliance requirements and security incident investigation.

IX. Future Enhancements

Planned enhancements include:

- Online collaborative editing for real-time coauthoring of examination papers
- Advanced analytics dashboard with performance metrics and usage statistics
- Mobile applications for iOS and Android platforms
- Integration with institutional learning management systems
- Automated quality checks for examination paper formatting

X. Conclusion

ExamSuite demonstrates how modern cloud-native tech- nologies can transform traditional institutional workflows while maintaining security, scalability, and usability. The system successfully addresses examination management challenges through role-based access control, comprehen- sive audit trails, and performance-optimized architecture.

The implementation validates the effectiveness of layered and client-server architectural patterns for complex institutional applications. Database-level security through Row-Level Security policies provides robust protection for sensitive examination materials. Performance optimizations including caching, lazy loading, and indexing ensure responsive user experience under concurrent load.

ExamSuite serves as a model for developing institution-specific applications that balance security requirements with user experience, demonstrating that undergraduate students can deliver enterprise-grade systems with appropriate guidance and modern technology stacks. The system is ready for deployment at Maharaja Institute of Technology, Mysore, providing significant improvements over manual examination management processes.

Acknowledgment

The authors express sincere gratitude to Dr. Sharath Kumar Y H, Head of Department, Information Science and Engi- neering, Maharaja Institute of Technology, Mysore, for in- valuable guidance in system architecture design and project mentorship. We acknowledge the support of the institution in providing resources and infrastructure for development and testing.

Author Profiles

R Pradhyumna is an undergraduate student in Information Science and Engineering at Maharaja Institute of Technology, Mysore. His research interests include full-stack web development, cloud-native architectures, and educational technology systems.

Arun Kumar H P is an undergraduate student in Information Science and Engineering at Maharaja Institute of Technology, Mysore, specializing in web application development and database design.

Kushal Gowda H M is an undergraduate student in Information Science and Engineering at Maharaja Institute of Technology, Mysore, with interests in software architecture and security implementation.

Madhu M S is an undergraduate student in Information Science and Engineering at Maharaja Institute of Technology, Mysore, focusing on frontend development and user interface design.

Dr. Sharath Kumar Y H is Professor and Head of the Department of Information Science and Engineering at Maharaja Institute of Technology, Mysore. He holds a Ph.D. in Computer Science and has over 15 years of teaching and research experience in software engineering, web technologies, and database systems.

References

[1] Sanket Vilas Salunke and Abdelkader Ouda, "A Performance Benchmark for the PostgreSQL and MySQL Databases," 2024.

[2] Shang-Pin Ma et al., "RESTful API Analysis, Recommenda- tion, and Client Code Retrieval," Electronics, 2023.

[3] Muhammad Umar Aftab et al., "Permission-Based Separa- tion of Duty in Dynamic Role-Based Access Control Model," Symmetry, 2019.

[4] Evgeny Nikulchev et al., "Technology Stack Selection Model for Software Design of Digital Platforms," Mathematics, 2021.

[5] Chamalla Sravani et al., "Constructing a Study Buddy Using MERN Stack Technologies," Engineering Proceedings, 2024.

[6] Denis Ulybyshev et al., "End-to-End Database Software Se- curity," 2023.

[7] Gabriel Nyame and Zhiguang Qin, "Precursors of Role-Based Access Control Design in KMS: A Conceptual Framework," 2020.

[8] Maryam Abbasi et al., "Adaptive and Scalable Database Man- agement with Machine Learning Integration: A PostgreSQL Case Study," 2024.

[9] Wisal Khan et al., "SQL and NoSQL Database Software Ar- chitecture Performance Analysis and Assessments—A Sys- tematic Literature Review," 2023.

[10] Deyou Tang et al., "Modeling the Data Provenance of Rela- tional Databases Supporting Full-Featured SQL and Procedu- ral Languages," 2023.

[11] Amit Kumar Goel et al., "Web-ChatLine: An Innovative Chatting Platform," 2022.

[12] Yunliang Li et al., "Role-Based Access Control Model for Inter-System Cross-Domain in Multi-Domain Environment," 2022.

[13] Vojdan Kjorveziroski et al., "IoT Serverless Computing at the Edge: A Systematic Mapping Review," 2021.

[14] Gergo Toth et al., "Edge or Cloud Architecture: The Applica- bility of New Data Processing Methods in Large-Scale Poul- try Farming," 2025.

[15] Wei Tang and Shuili Yang, "Enterprise Digital Management Efficiency under Cloud Computing and Big Data," 2023.

[16] Andrej Grguric et al., "Integration and Deployment of Cloud- Based Assistance System in Pharaon Large Scale Pilots— Experiences and Lessons Learned," 2022.

[17] George F. Fragulis et al., "O.D.E.S.: An Online Dynamic Ex-amination System Based on a CMS WordPress Plugin," 2020.

[18] Kai Cao and Zhe Wang, "Fine-Grained Access Control Sys- tem for Multi-User Environments," 2024.