perplexity

There were 8 points in the previous one and security features please redo with all features and clarity

Here is a **full, foolproof, and modern overview** of all tech tools, languages, structures, and security features for your college document management system, mapped to your project's purpose, user roles, and the detailed architecture in your images[1] [2] [3].

1. Users & Permissions

Admin (Super User)

- Set deadlines for each batch/semester.
- Monitor which teachers have submitted/missed deadlines.
- Access all uploaded documents.
- Upload official college policies.
- Manage users (add/remove teachers).

Teacher (Normal User)

- Upload course documents for their subjects.
- Track their own uploads.
- Download all their course documents.
- View college policies.
- Edit/update their uploads (if allowed).

Why:
Clear role-based access ensures only authorized users can perform sensitive actions, supporting compliance and operational clarity[4].

2. Frontend (User Interface Layer)

Tool/Tech	Language	Where Used	Why Used
React.js	TypeScript	Admin & Teacher Dashboards	Modern, component-based UI for dashboards and forms
Redux Toolkit	TypeScript	Frontend state management	Centralized, predictable state (user sessions, permissions, notifications)

Tool/Tech	Language	Where Used	Why Used
Vite	TypeScript	Frontend build tool	Fast builds, hot module reload, optimized dev workflow
Material-UI/AntD	TypeScript	UI components	Accessible, responsive, and professional UI elements
Dropzone.js	TypeScript	File upload components	Drag-and-drop uploads, previews, validation, progress bars
Axios	TypeScript	API communication	Secure HTTP requests from frontend to backend
Prettier	TypeScript	Code formatting	Consistent, automatic code style enforcement
ESLint	TypeScript	Code linting	Enforces code quality and best practices
Jest/React Testing Library	TypeScript	Frontend testing	Unit/integration testing for UI reliability

Why:
Delivers a secure, accessible, and user-friendly experience for all users, while maintaining high code quality and rapid development cycles[5] [6].

3. Backend (API & Business Logic Layer)

Tool/Tech	Language	Where Used	Why Used
Node.js	JavaScript	Backend runtime	High-performance, scalable server-side logic
Express.js	JavaScript	REST API framework	Routing, middleware, and business logic
Multer	JavaScript	File upload middleware	Securely parses and validates file uploads
JWT (jsonwebtoken)	JavaScript	Auth & session mgmt	Stateless, secure authentication and RBAC
Bcrypt.js	JavaScript	Password hashing	Secure credential storage
Helmet.js	JavaScript	HTTP security headers	Protects against common web vulnerabilities
CORS	JavaScript	Cross-origin requests	Controls which domains can access APIs
Rate Limiting	JavaScript	API protection	Prevents brute-force and DDoS attacks
Joi/validator.js	JavaScript	Input validation	Prevents injection and malformed data
Audit Logging	JavaScript	Compliance	Tracks all user/document actions
Mocha/Chai/Supertest	JavaScript	Backend testing	API and integration testing

Why:
Ensures all business logic, file handling, and user management are secure, scalable, and maintainable[5] [6] [7].

4. Database (Persistent Storage Layer)

Tool/Tech	Language	Where Used	Why Used
MongoDB Atlas	JavaScript	Cloud database	Flexible, NoSQL storage for users, docs, deadlines, metadata
Mongoose	JavaScript	ODM for MongoDB	Schema definition, validation, and secure queries

Security Features:

- RBAC at the database level.
- Encrypted at rest and in transit.
- Only backend API has direct access.
- Automated daily backups for disaster recovery[8].

Why:
Supports flexible data models, strong security, and compliance with institutional requirements[5] [8] [6].

5. File Storage (Document Storage Layer)

Tool/Tech	Where Used	Why Used
AWS S3	Cloud file storage	Secure, scalable, highly available document storage
Local Storage	Optional redundancy	Hybrid or backup storage if needed
Pre-signed URLs	File access	Secure, time-limited file upload/download links

Security Features:

- Server-side encryption for all files.
- Access controls: Only authorized users can retrieve files.
- Malware/virus scanning on all uploads.
- Redundant storage for disaster recovery[8].

Why:
Ensures reliable, secure, and scalable storage for all documents and policies[8] [7].

6. Infrastructure (Deployment, Routing, Scaling)

Tool/Tech	Where Used	Why Used
Nginx/Apache	Reverse proxy, SSL	Secure routing, SSL termination, static asset serving
AWS ELB	Load balancing	Distributes requests for high availability and scaling

Tool/Tech	Where Used	Why Used
Docker	Containerization	Consistent, portable deployments for all services
Kubernetes*	Orchestration	Auto-scaling, self-healing, rolling updates (optional, for large scale)
Performance Monitoring (Prometheus, Grafana, AWS CloudWatch)	Real-time health checks and alerting	Proactive issue detection and uptime assurance
Auto-scaling	Resource management	Dynamically adjusts resources based on demand
Firewall/VPC	Network security	Restricts access to only necessary services
DDoS Protection	Network security	Mitigates denial-of-service attacks

Why:
Provides a robust, scalable, and secure backbone for the entire system, ensuring uptime and resilience[1] [3] [8].

7. Security & Compliance (Cross-Cutting)

Feature/Practice	Where Used	Why Used
RBAC	Backend, Database	Granular permissions for all actions and data
Multi-factor Auth (MFA)	Login/authentication	Extra layer of security for sensitive actions
End-to-end Encryption	All data flows	Protects data in transit and at rest
Audit Trails	All actions	Full traceability for compliance and incident response
Regular Security Audits	System-wide	Identifies and fixes vulnerabilities
Automated Backups	DB & file storage	Disaster recovery
File Validation & Malware Scanning	Uploads	Prevents malicious files from entering the system
Secure Password Storage	Backend	Bcrypt for password hashing
Secure HTTP Headers	Backend	Helmet.js for XSS, CSP, and clickjacking protection

Why:
Protects sensitive academic data, ensures only authorized access, and meets regulatory requirements[8] [6] [7].

8. Additional Features & Integrations

Feature/Tool	Where Used	Why Used
Workflow Automation	Backend, Frontend	Automated notifications, reminders, and approval flows
OCR (Tesseract.js)	Backend, Frontend	Converts scanned PDFs to searchable text
Electronic Signatures (DocuSign API)	Backend	For policy documents and approvals
Document Versioning	Backend, DB	Tracks all changes and prevents accidental overwrites
Metadata & Tagging	Backend, DB, Frontend	Advanced search and filtering
Accessibility (WCAG)	Frontend	Inclusive UI for all users
Integration APIs/Webhooks	Backend	Connects with ERP, LMS, HR, or other college systems

Why:
Enhances user experience, compliance, and extensibility for future needs[1] [2] [3] [8].

9. DevOps & Collaboration

Tool/Tech	Where Used	Why Used
Git & GitHub/GitLab	Source control	Team collaboration, versioning, and code reviews
CI/CD (GitHub Actions, GitLab CI)	DevOps	Automated testing, linting, and deployment pipelines
VS Code	Development	Main IDE with rich extension support

Why:
Ensures efficient, collaborative, and automated development and deployment cycles.

Security Features Summary

- **Data Encryption:** AES-256 at rest (DB, files), HTTPS/TLS in transit[8] [9].
- **RBAC:** Granular permissions for all user actions and data[4].
- **Authentication:** JWT, MFA, secure password hashing[6] [7].
- **Input Validation & Sanitization:** Prevents XSS, SQL/noSQL injection, and other attacks[6] [7].
- **Malware/Antivirus Scanning:** All uploads checked for malicious content[8] [7].
- **Redundancy & Disaster Recovery:** Off-site backups, cloud redundancy, auto-recovery[8].
- **Audit Logging:** Tracks all access and changes for compliance and incident response[8] [6].
- **Regular Security Audits:** Automated and manual reviews to identify vulnerabilities[8] [7].

Conclusion

This stack and structure—mapped directly to your project's purpose, user roles, and architecture—delivers a secure, scalable, maintainable, and user-friendly document management system. Every tool and feature is chosen for industry best practices, regulatory compliance, and operational excellence[1] [2] [3] [5] [8] [6] [9] [4] [7].

References:

- Images provided: [1] [2] [3]
- Security and stack best practices: [5] [8] [6] [9] [4] [7]

⁂

1. WhatsApp-Image-2025-06-19-at-19.13.54_5f6be2c8.jpg
2. WhatsApp-Image-2025-06-19-at-19.14.07_994cd473.jpg
3. WhatsApp-Image-2025-06-19-at-19.48.11_bfad8c0d.jpg
4. https://auth0.com/docs/manage-users/access-control/rbac
5. https://www.index.dev/blog/best-tech-stacks-for-web-application-development
6. https://www.guvi.in/blog/best-practices-to-secure-mern-stack-applications/
7. https://massive.io/content-security/how-to-upload-files-securely/
8. https://www.ricoh.com.my/blogs/advanced-security-protocols-in-modern-document-management-solutions
9. https://cronokirby.com/posts/2021/06/e2e_in_the_browser/