Ohm Patel, 2025, 13:6
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Taskflow: A Real-Time, Secure, and Extensible Task Management Web Application Using Next.js, Convex, Clerk, and Edgestore

Ohm Patel, Neel, Devraj, Yashvi
Department of Computer Science and Engineering
Team Lead, Full-Stack Architect and Coordinator Frontend Developer and UI/UX Designer
Authentication and Real-Time Sync Engineer Testing and Environment Specialist

Abstract - Efficient task management and collaborative docu- ment handling are foundational to distributed software teams and modern organizations. This paper presents Taskflow, a web application that combines real-time collaboration, secure authentication, multilingual document translation, AI-assisted document queries, and exportable, presentation-ready documents in a cohesive platform. The system leverages Next.js (App Router) for routing and rendering, React and TailwindCSS for the user interface, Convex for real-time backend logic and persistent state, Clerk for authentication and role-based access control (RBAC), and Edgestore for edge-powered uploads and synchronization. We describe the motivation, design goals, system architecture decisions, development methodology, implementation details, and results from an academic capstone-style build. Our evaluation focuses on developer experience, responsiveness, reliability of syn- chronization, and functional coverage. We also discuss limitations and opportunities for extension, including offline-first operation and advanced AI-driven workflows.

Index Terms - Task management, real-time collaboration, Next.js, Convex, Clerk, Edgestore, RBAC, multilingual transla- tion, AI assistant, web application engineering.

I. INTRODUCTION

Task and project management platforms have matured from simple to-do lists to complex collaboration suites with shared documents, presence indicators, role-based permissions, and AI assistance. However, teams frequently stitch together dis- joint tools for authentication, synchronization, storage, and intelligence, incurring overhead in integration, operability, and maintenance.

Taskflow addresses these challenges by delivering an inte- grated, developer-friendly system that foregrounds: (i) real- time collaboration on tasks and documents, (ii) robust and secure authentication with RBAC, (iii) multilingual document translation with persistent saved variants, (iv) an AI chat assis- tant that answers questions grounded in the active document, and (v) high-quality PDF exports for offline dissemination. The platform is built atop a modern, production-grade stack that privileges simplicity, performance, and maintainability.

This paper makes three contributions: (1) a cohesive design that unifies real-time data, security, and user experience using a minimal set of interoperable technologies; (2) a concrete methodology to organize a small student team around Agile practices and code quality gates; and (3) a reference imple- mentation and qualitative evaluation demonstrating practicality for academic and early-stage industry contexts.

II. RELATED WORK

Mainstream collaborative systems such as Trello, Asana, Notion, and Jira popularized task boards, rich documents, and extensibility. Many offer APIs and

third-party plugins, yet integration depth for live document-aware AI or multilin- gual translation varies. Some tools rely on periodic polling or coarse-grained synchronization that can introduce update latency in high-collaboration scenarios.

Open-source and research systems frequently emphasize one dimension (e.g., real-time text synchronization) but leave security, storage, or AI as future work. In contrast, Taskflow's approach integrates (a) Convex for low-latency queries/muta-tions and serverless functions, (b) Clerk for first-class authen- tication, session management, and RBAC, and (c) Edgestore for edge-proximal file transfer, with (d) a Next.js frontend that supports streaming UI and server actions. This combination reduces integration friction while preserving flexibility for future features.

III. PROBLEM STATEMENT AND OBJECTIVES

Distributed teams need a platform that simultaneously pro- vides:

- Real-time collaboration: Low-latency updates and pres- ence indicators for tasks and documents.
- Security and governance: RBAC, session management, and auditable flows.
- Multilingual reach: Persistent translations and seamless language switching at the document level.
- Actionable intelligence: An AI assistant grounded in project documents for contextual Q&A.
- Portability: Clean PDF exports for reporting, review, and archiving.

Taskflow's objective is to deliver these capabilities in a cohe- sive, developer-friendly system with a maintainable codebase and clear operational pathway to production deployments (e.g., Vercel).

IV. SYSTEM OVERVIEW AND DESIGN DECISIONS

Taskflow employs a modular architecture with clear separa- tion of concerns:

- Presentation Layer: Next.js (App Router) and React man- age routing and rendering; TailwindCSS provides a utility- first styling system to achieve consistent, accessible UI.
- Application Logic: Convex encapsulates server-side logic with queries and mutations that expose real-time reactive data sources to the client.
- Identity and Access: Clerk manages authentication flows, sessions, and RBAC, minimizing custom security code while enabling fine-grained policy.
- File and Edge Sync: Edgestore handles high-performance uploads, storage, and distribution, with real-time change propagation semantics.
- Intelligence: The AI assistant (via OpenAI) performs retrieval-grounded Q&A on current document content to improve discoverability and onboarding.

Key design principles include: (i) eventual simplicity— prefer platform primitives over bespoke infrastructure; (ii) pro- gressive enhancement—graceful degradation when advanced features are unavailable; (iii) observability—structured logging of auth events, mutations, and export operations; and (iv) security by default—deny-by-default RBAC and protected server-side routes.

V. METHODOLOGY

We adopted Agile practices with weekly sprints and daily asynchronous stand-ups. Work items were tracked with clear acceptance criteria and code owners. Pull requests required at least one peer review before merge; CI checks enforced linting and type safety.

Team Structure and Roles

Ohm (Team Lead, Full-Stack Architect). Owns roadmap, milestones, and architecture; coordinates cross-cutting con- cerns such as auth boundaries, data schemas, and deployment. Oversees CI/CD and

ensures implementation fidelity to design. Neel (Frontend Developer, UI/UX Designer). Designs and implements responsive, accessible components; maintains a consistent design system with TailwindCSS; ensures seamless data bindings to Convex queries/mutations; contributes to performance tuning.

Devraj (Authentication and Real-Time Sync Engineer). Integrates Clerk; implements RBAC and session handling; hardens authentication flows; verifies real-time semantics be- tween Convex subscriptions and Edgestore change notifica- tions.
Yashvi (Testing and Environment Specialist). Curates environment configurations for local and production; maintains .env hygiene; builds test plans; conducts manual and auto- mated tests; monitors release readiness and rollback strategies.

Quality Assurance

Unit and integration tests target critical paths: authentication flows, task CRUD, document translation persistence, and PDF export idempotency. Regression suites validate that real-time updates are delivered under concurrent edits. Peer review em- phasizes readability, accessibility (a11y), and error handling.

Implementation

Technology Stack

Next.js & React: The App Router supports server compo- nents and route groups, improving data-fetch strategies while preserving client interactivity. Streaming responses reduce time-to-interactive for authenticated routes.

TailwindCSS: Utility classes speed implementation while encouraging consistency. Custom themes and tokens define a cohesive visual language.

Convex: Schema definitions, queries, and mutations provide a reactive data layer. Client subscriptions deliver updates with- out manual polling. Server-side functions encapsulate business rules and enforce RBAC checks.

Clerk: Provides user registration, sign-in, session lifecycle, and RBAC claims. Middleware guards protect routes and API endpoints; webhooks synchronize identity events with Convex. Edgestore: Edge-proximal uploads improve perceived la- tency. Signed URLs and content validation preserve integrity; metadata supports previews, quotas, and lifecycle policies.

OpenAI Integration: The AI assistant answers questions about the active document. A retrieval step collects rele- vant sections; prompts are grounded and rate-limited. Results stream to the UI to improve responsiveness.

Core Features

Real-Time Document Sharing and Presence: Collabo- rative cursors and document states are synchronized via Convex subscriptions; optimistic UI keeps editing fluid.
Secure Authentication and RBAC: Clerk sessions gate access; role claims (e.g., owner, editor, viewer) are verified server-side before executing mutations.
Multilingual Translation: Documents can be translated, saved as language variants, and switched dynamically. The original language is auto-detected; switching triggers content re-hydration without reloads.
AI Chat-to-Document Assistant: Users ask free-form questions; the assistant fetches salient passages, generates grounded answers, and links to document anchors.
PDF Export: Using html2pdf.js, users export well-styled PDFs suitable for submission or archiving. Exports are deterministic across sessions and include metadata.

Data Model and Access Patterns

A minimal core schema models users, documents, translations, shares, and activities. Documents store structured content; translations reference a base docu- ment and target locale. Shares encode per-user or link-based permissions. Activity logs support audit trails and analytics (e.g., unique collaborators, edit frequency).

Client components read via Convex queries keyed by docu- ment IDs and user claims; mutations validate RBAC, sanitize input, and append activities. Hot

paths (e.g., keystroke-level edits) batch updates for network efficiency.

Operational Concerns

Deployments target Vercel for Next.js and Convex's man- aged runtime for backend logic. Environment variables govern secrets for Clerk, Edgestore, Convex, and OpenAI. Observ- ability relies on structured logs and dashboard alerts for auth anomalies, elevated error rates, or degradation in real-time delivery. Backups cover Convex data and Edgestore buckets with retention windows aligned to institutional policies.

Evaluation and Results

We report qualitative outcomes from end-to-end usage in a small-team setting.

Developer Experience

The stack reduced custom boilerplate: Clerk removed the need for bespoke auth; Convex obviated a separate API gateway; Edgestore simplified uploads and distribution. React Server Components and streaming routes improved perceived performance for authenticated pages. Onboarding new con- tributors was accelerated by a consistent folder structure and typed interfaces.

Responsiveness and Real-Time Behavior

Under concurrent editing by four users on typical broad- band, updates appeared sub-second with no observed merge conflicts in routine usage. Presence indicators and optimistic updates improved collaboration fluency. When network in- terruptions occurred, reconciliation restored consistency upon reconnect without user intervention.

Security and Governance

RBAC prevented privilege escalation in tests where viewers attempted mutations. Session revocation via Clerk immediately invalidated access tokens. Audit logs surfaced share-link mis- use and supported remediation by revoking links or tightening scopes.

Functional Coverage

The translation system preserved formatting and allowed instant language switching. The AI assistant returned context- relevant answers for common tasks (definitions, summaries, and cross-references). PDF exports matched on-screen render- ing closely and were accepted in course submissions without reformatting.

Limitations

Taskflow currently assumes stable connectivity; while tran- sient outages are handled gracefully, a fully offline-first mode is future work. The AI assistant's quality depends on prompt- guardrails and retrieval coverage; long documents may require chunking and advanced ranking. Large binary assets can stress quotas without lifecycle policies.

Discussion

The primary benefit of Taskflow is cohesion: security, collaboration, translation, and intelligence co-exist without brittle glue code. This elevates reliability and reduces long- term maintenance. For student teams, these qualities translate to faster iteration and clearer separation of responsibilities; for early-stage organizations, they enable quicker time-to-value.

Trade-offs include reliance on managed backends, which constrains low-level customization but yields strong defaults for security, availability, and scaling. We found that shaping the domain model early— especially permission boundaries— prevents rework as features expand (e.g., shared links, external collaborators, or document templates).

Future Work

Future enhancements include: (i) offline-first editing with background synchronization and conflict resolution; (ii) deeper AI workflows such as task triage, summarization, and auto- tagging; (iii) compliance features (export controls, retention, DLP); (iv) extended analytics (burndown, velocity, collabo- rator heatmaps); and (v) mobile-native clients leveraging the same Convex-backed APIs.

VI. CONCLUSION

Taskflow demonstrates that a small, well-orchestrated stack can deliver a robust, real-time collaboration platform with strong security, multilingual support, AI assistance, and clean exports. By centering on managed, interoperable services and disciplined engineering practices, teams can achieve production-grade functionality with modest complexity. The implementation and results indicate suitability for academic deployments and a strong foundation for industrial adaptation.

REFERENCES

1. Task Management Support in Information Seeking, 2004.
2. What a To-Do: Studies of Task Management, 2004.
3. Challenges for Business Process and Task Management, 2005.
4. Towards Task-Based Personal Information Management, 2007.
5. Tacit Knowledge Management & Task Efficiency, 2024.
6. Why Task Management Apps Increase Productivity, 2019–2022.
7. Optimizing Database Performance, 2023.
   Microservices vs. Monolithic Architectures, 2022.
8. Agile Software Development, 2021.
9. Introduction to MERN Stack & Comparisons, 2023.
10. Integration of NLP in MongoDB NoSQL, 2025.
11. Clustering of ACS Patients Using Neural Networks, 2019.
12. Data Architecture for DOSM Using DAT, 2023.
13. Web-Based Student Task Management System, 2022.
14. Optimizing Productivity: Task Management Systems, 2024.
15. Project Tracking System Using MERN Stack, 2021.
16. Development of a Task Management Software, 2023.
17. TaDeR: Task Dependency Recommendation, 2022.
18. What a To-Do: Task Management Study, 2004.
19. Challenges for Business Process and Task Management, 2005.