Agentic Clinical Trial Operations Assistant

Pranav Jitendra Trivedi & Mohit Manoj Barade
Master's in Software Engineering & Master's in Computer Engineering
San Jose State University
San Jose, United States
pranavjitendra.trivedi@sjsu.edu
mohitmanoj.barade@sjsu.edu

Yashashav Devalapalli Kamalraj & Anupama Singh
Master's in Software Engineering
San Jose State University
San Jose, United States
yashashav.devalapallikamalraj@sjsu.edu
anupama.singh@sjsu.edu

***Abstract*—The Agentic Clinical Trial Operations Assistant (ClinOps) is designed to support clinical operations and regulatory teams in planning, managing, and documenting clinical trials. By leveraging large language models (LLMs), structured project context, and a modern web stack, the system helps coordinators and regulatory advisors generate audit-ready documentation, visualize trial workflows, and monitor trial progress. ClinOps collects essential trial data through guided questionnaires and uses an LLM-driven engine to produce persona- and tab-specific content such as trial overviews, task checklists, regulatory compliance diagrams, and audit preparation guides. The platform persists projects and chat histories in a PostgreSQL database via Prisma and exposes a Next.js-based interface for interactive trial design and operations support. This paper presents the system architecture, core functionalities, personas, and technologies used, as well as testing and observability practices like logging, metrics, and distributed tracing which ensures reliability, performance, and traceability in a clinical operations setting.**

Index Terms: **Clinical Trial Operations, Regulatory Compliance, Next.js, React, Large Language Models, Google Gemini, Mermaid Diagrams, Prisma, PostgreSQL, JWT Authentication, Context-Aware Chat, AI-Assisted Documentation, OpenTelemetry, Prometheus Metrics, Pino Logging, Observability, DevTools for ClinOps, Project-Based Context, Dashboard Widgets.**

I. Introduction

Clinical trials require precise coordination among multiple stakeholders, strict regulatory compliance, and meticulous documentation to stay audit-ready. Traditional tools scatter critical information across spreadsheets, static documents, and inconsistent communication channels, making it difficult to maintain an accurate, real-time view of study progress or compliance status. Teams often recreate similar artifacts like trial summaries, checklists, compliance diagrams, and audit-prep materials, while ensuring alignment with protocols and Good Clinical Practice.

The Agentic Clinical Trial Operations Assistant (ClinOps) solves these issues by providing a unified, project-centric workspace that captures essential trial details, offers a context-aware assistant trained in clinical operations and regulatory standards, and dynamically generates structured content such as study overviews, operational checklists, protocol requirements, and risk-control outputs. It also allows users to save and reuse AI-generated diagrams and dashboards tailored to each project, improving standardization, accuracy, and operational efficiency across the clinical trial lifecycle.

II. System Architecture

The system is designed to provide a seamless user experience while ensuring robust backend processing for data analysis and personalized recommendation generation. The architecture is split into several components, each serving a specific function:

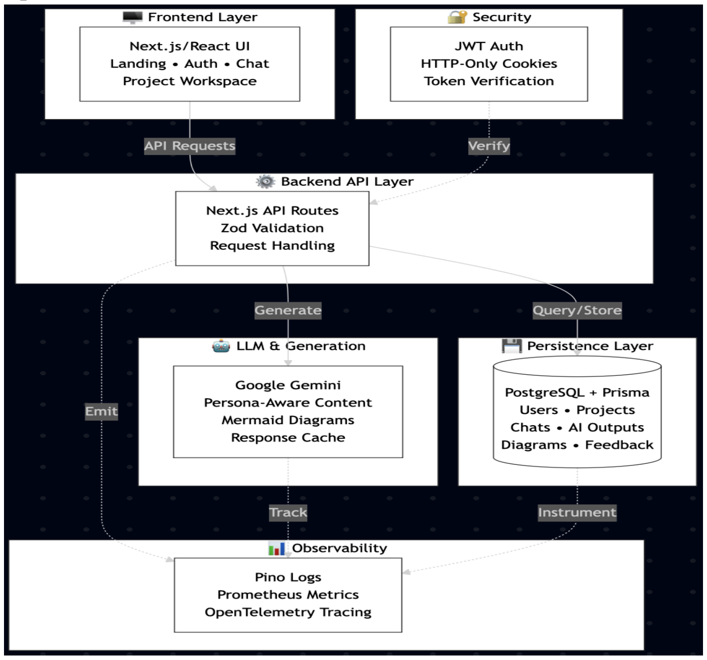


Fig. 1. System Architecture Diagram

A. *Frontend Layer (Next.js / React)*

The frontend is built using Next.js and React, providing a responsive and dynamic interface for clinical operations teams. It includes the landing page, authentication flows, and a project-based chat workspace. Users interact through a context-aware chat UI that supports persona selection, guided workflows, and AI-generated content tabs tailored to clinical trial needs.

B. *Backend API Layer (Next.js API Routes)*

The backend uses Next.js API Routes to handle application logic and communication with the database. It processes requests for authentication, project management, AI generation, and chat persistence. Input validation is performed using Zod, while JWT-based authentication ensures secure access. Prisma enables clean, type-safe interactions with the database.

C. *LLM and Content Generation Layer*

This layer manages persona-aware content generation, workflow-specific outputs, and dynamic diagram creation. It uses Google's Gemini model to produce structured clinical operations content, such as checklists, summaries, and Mermaid diagrams. Optional caching improves performance by storing previous responses for faster retrieval.

D. *Persistence Layer (PostgreSQL + Prisma)*

Data is stored in a PostgreSQL database accessed through Prisma's unified client. The system maintains users, projects, chat logs, AI-generated outputs, diagrams, dashboards, and feedback. Prisma's instrumentation adds transparency and performance insights, supporting scalable and audit-ready clinical operations.

E. *Authentication and Security Layer*

ClinOps uses email/password authentication secured with JWTs stored in HTTP-only cookies. The authentication utilities verify tokens on each request, ensuring only authorized users can access sensitive trial content. This layer maintains the integrity and security required for compliance in regulated environments.

F. *Observability Layer (Logging, Metrics, Tracing)*

The observability stack includes structured logging, metrics, and distributed tracing. Pino provides consistent and efficient logs, while Prometheus-compatible metrics help track system performance. OpenTelemetry offers end-to-end tracing with auto-instrumentation and export capabilities, giving full visibility into application behavior across layer.

III. Functionalities

ClinOps provides several high-level functionalities to support clinical operations and regulatory work:

A. *Project Creation and Workspace Management*

Users can create new projects as authenticated users, with data stored in PostgreSQL, or as guests, whose projects begin in local storage and are later transferred to the database after login. Each project has its own dedicated chat and content workspace accessible via its project URL.

B. *Essential Trial Question Workflow*

The system presents a structured sequence of essential trial questions covering items such as trial name, primary endpoints, and study design details. Each question appears as a styled chat element with the question text, a sample answer, a copy option, and an inline input field. As users provide answers, the responses are saved within the project state, displayed as user chat bubbles, and used as structured inputs for downstream LLM-generated content.

C. *Context-Aware Chat*

Messages are tracked and incorporated into a running conversation history used to construct rich, contextually informed prompts. User queries combine their latest input, conversation context, and project information, enabling both guided and free-form interactions centered on trial operations, compliance, and protocol interpretation.

D. *AI-Generated Tab Content*

For each persona and tab type, the system applies predefined prompt templates that guide the generation of structured outputs, often starting with Mermaid diagrams. Tabs include trial overviews, task checklists, team workflows, timelines, protocol requirements, document control, compliance diagrams, risk controls, audit preparation, and alerting logic. Content generation may involve multiple LLM calls for initial output, refinement, and optional caching and is rendered with Markdown and Mermaid support. Users can persist, export, or convert this content into dashboard elements.

E. *Dashboard and Diagram Management*

Users can promote generated tab content into dashboard widgets for project-level visualization. Mermaid diagrams can also be saved as reusable assets for documentation, reporting, or presentation purposes.

F. *Feedback and Iteration*

Users may provide thumbs-up or thumbs-down ratings and optional comments on AI outputs. All feedback is stored and can be used to refine prompts, improve generation quality, and enhance the overall user experience in the next iteration.

IV. Persona

The system is designed with several stakeholders in mind, each benefiting from its features in different ways:

A. *Trial Coordinators*

Trial Coordinators focus on the operational execution of a study, including enrollment, scheduling, task management, visit adherence, and data collection workflows. They rely on tabs such as Trial Overview, Task Checklists, Team Workflows, Trial Timeline, and Quality Metrics. The assistant helps generate actionable checklists, timelines, and workflow plans aligned with each trial's specific design.

B. *Regulatory Advisors*

Regulatory and compliance professionals ensure that studies remain audit-ready and aligned with standards such as ICH-GCP, FDA, and EMA guidelines. They use tabs including Protocol Requirements, Document Control, Compliance Diagrams, Risk Controls, Audit Preparation, and Smart Alerts. AI-generated diagrams and structured narratives support them in building regulatory binders, risk-management frameworks, and audit preparation materials.

The persona system shapes prompt templates, tone, and diagram structures so that generated content aligns with the user's role and responsibilities.

V. Technologies Used

A. *React and Next.js (Frontend Development)*

React and Next.js power both the marketing pages and the project workspace. The frontend combines client components for rich interactivity with serverless API routes for backend functionality, supported by Tailwind CSS and custom styling to deliver a responsive and modern interface.

B. *Next.js API Routes and Node Environment (Backend)*

The backend runs on Next.js API routes within a Node.js environment. It manages authentication, project CRUD operations, AI orchestration, and metrics export.

C. *JWT (Authentication)*

Authentication is implemented using a JWT-based system. Users sign up or log in with email and password, and the system issues signed tokens stored in HTTP-only cookies. Centralized utilities manage token verification to ensure secure access to protected routes.

D. *PostgreSQL (Database Management)*

PostgreSQL serves as the primary relational database, with Prisma defining the schema for users, projects, chats, AI responses, diagrams, dashboards, and feedback. Metrics instrumentation is added through Prisma middleware to monitor database performance and query behavior.

E. *Google Gemini (LLM)*

The LLM integration layer uses Google's Gemini model through an abstraction that supports retries, backoff, and fallback handling for misconfiguration or transient failures. Both server-side and client-side wrappers manage AI requests, caching metadata, and configurable parameters such as temperature and token limits. Persona and tab-specific prompt logic orchestrates structured outputs and refinement flows.

F. *Pino, Prometheus, Prisma metrics, OpenTelemetry*

ClinOps incorporates a comprehensive observability stack. Logging is handled with structured Pino logs that redact sensitive fields and support correlation IDs. Prometheus-compatible metrics track HTTP request durations, request counts, error rates, query latency, and overall system performance. Prisma metrics middleware adds precise instrumentation around database queries. OpenTelemetry provides distributed tracing with auto-instrumentation and OLTP export, enriched with service and environment metadata for organized trace analysis.

VI. Testing

The Agentic Clinical Trial Operations Assistant underwent comprehensive testing to ensure the reliability and performance of the entire application stack. Jest served as the primary testing framework, enabling unified testing across both backend API routes and frontend React components. The testing process involved unit tests, integration tests, and component tests to verify that all layers functioned as expected and met critical operational requirements.

The testing approach focused on the essential features users rely on most. The automated test suite was structured across four key domains:

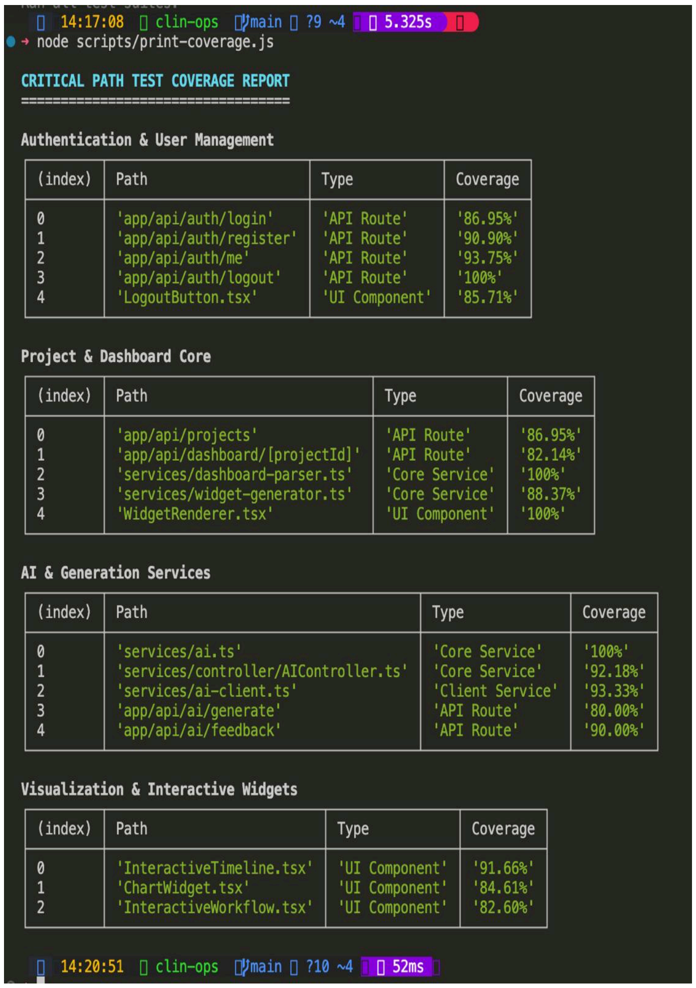
A. **Authentication & User Management**: Individual authentication endpoints and session management services were tested to confirm secure access control. This included testing API routes for login, registration, and session validation, as well as UI components like the logout button. Coverage ranged from 86-100%, ensuring both valid credentials were accepted and invalid ones properly rejected.

B. **Project & Dashboard Core**: Core services responsible for project creation and dashboard rendering were rigorously tested. The dashboard parser service achieved 100% coverage, while API routes for project management and dashboard data retrieval maintained over 82% coverage. React components such as the widget renderer were tested to confirm correct rendering without errors.

C. **AI & Generation Services**: Given that AI-powered assistance is the core value proposition, extensive testing was applied to AI service integration. The base AI service layer achieved 100% coverage, while the AI controller handling complex orchestration maintained 92.18% coverage. API endpoints for AI generation and user feedback were mocked to simulate responses from AI models, allowing validation of both expected results and error handling scenarios.

D. **Visualization & Interactive Widgets**: Interactive UI components including timelines, workflows, and chart widgets were tested for both rendering accuracy and user interaction handling. These tests verified correct display of clinical trial data visualizations and proper response to user interactions such as clicks and hovers, achieving coverage between 82-91%.

E. Test Execution & Results:



```
  14:17:08  clin-ops  main  ?9 ~4   5.325s
→ node scripts/print-coverage.js

CRITICAL PATH TEST COVERAGE REPORT
==================================

Authentication & User Management
┌─────────┬────────────────────────────┬────────────────┬──────────┐
│ (index) │ Path                       │ Type           │ Coverage │
├─────────┼────────────────────────────┼────────────────┼──────────┤
│ 0       │ 'app/api/auth/login'       │ 'API Route'    │ '86.95%' │
│ 1       │ 'app/api/auth/register'    │ 'API Route'    │ '90.90%' │
│ 2       │ 'app/api/auth/me'          │ 'API Route'    │ '93.75%' │
│ 3       │ 'app/api/auth/logout'      │ 'API Route'    │ '100%'   │
│ 4       │ 'LogoutButton.tsx'         │ 'UI Component' │ '85.71%' │
└─────────┴────────────────────────────┴────────────────┴──────────┘

Project & Dashboard Core
┌─────────┬───────────────────────────────────┬────────────────┬──────────┐
│ (index) │ Path                              │ Type           │ Coverage │
├─────────┼───────────────────────────────────┼────────────────┼──────────┤
│ 0       │ 'app/api/projects'                │ 'API Route'    │ '86.95%' │
│ 1       │ 'app/api/dashboard/[projectId]'   │ 'API Route'    │ '82.14%' │
│ 2       │ 'services/dashboard-parser.ts'    │ 'Core Service' │ '100%'   │
│ 3       │ 'services/widget-generator.ts'    │ 'Core Service' │ '88.37%' │
│ 4       │ 'WidgetRenderer.tsx'              │ 'UI Component' │ '100%'   │
└─────────┴───────────────────────────────────┴────────────────┴──────────┘

AI & Generation Services
┌─────────┬────────────────────────────────────────┬──────────────────┬──────────┐
│ (index) │ Path                                   │ Type             │ Coverage │
├─────────┼────────────────────────────────────────┼──────────────────┼──────────┤
│ 0       │ 'services/ai.ts'                       │ 'Core Service'   │ '100%'   │
│ 1       │ 'services/controller/AIController.ts'  │ 'Core Service'   │ '92.18%' │
│ 2       │ 'services/ai-client.ts'                │ 'Client Service' │ '93.33%' │
│ 3       │ 'app/api/ai/generate'                  │ 'API Route'      │ '80.00%' │
│ 4       │ 'app/api/ai/feedback'                  │ 'API Route'      │ '90.00%' │
└─────────┴────────────────────────────────────────┴──────────────────┴──────────┘

Visualization & Interactive Widgets
┌─────────┬─────────────────────────────┬────────────────┬──────────┐
│ (index) │ Path                        │ Type           │ Coverage │
├─────────┼─────────────────────────────┼────────────────┼──────────┤
│ 0       │ 'InteractiveTimeline.tsx'   │ 'UI Component' │ '91.66%' │
│ 1       │ 'ChartWidget.tsx'           │ 'UI Component' │ '84.61%' │
│ 2       │ 'InteractiveWorkflow.tsx'   │ 'UI Component' │ '82.60%' │
└─────────┴─────────────────────────────┴────────────────┴──────────┘

  14:20:51  clin-ops  main  ?10 ~4   52ms
```

Mocking strategies were employed throughout the test suite to isolate components during testing. External dependencies such as database connections via Prisma and AI model integrations were mocked to ensure tests remained focused and independent of external systems. This approach enabled rapid test execution while maintaining comprehensive validation of business logic.

VII. API Security

To ensure secure access to the Agentic Clinical Trial Operations Assistant, the system employs a custom authentication and authorization framework built on industry-standard cryptographic libraries. This solution provides robust control over securing APIs and protecting sensitive clinical trial data. The following are the key aspects of the API security strategy:

A. **Authentication Implementation:** The platform utilizes JSON Web Tokens (JWT) for stateless user authentication, implemented through the jose library. This enables secure verification without requiring server-side session storage, providing users with efficient authentication while maintaining strong security guarantees.

   Secure Token Exchange: Upon successful authentication, the system generates a JWT signed with the HS256 algorithm that is passed between the client and server. The backend validates this token to verify the user's identity and ensure requests originate from authenticated sources.

   HTTP-Only Cookie Storage: Rather than storing tokens in localStorage where they remain vulnerable to XSS attacks, the application uses HTTP-Only cookies for token storage. These cookies include security flags (HttpOnly, Secure, SameSite=Lax) and are configured with a seven-day validity period, ensuring continuous secure access without repeated sign-ins.

B. **Authorization & Access Control:** After successful authentication, the backend validates user authorization by performing database verification through Prisma ORM. Based on the user's account status and permissions, specific resources and endpoints are made accessible. The system employs dual-layer access control, first validating the JWT signature against the server-side secret, then performing a secondary database check to confirm the account remains active. This ensures that suspended or deleted accounts cannot access the system even with valid tokens.

C. **API Security Best Practices:**

   Token Validation: The backend ensures each API request contains a valid token. Tokens are validated using the verification utility, confirming both signature integrity and payload authenticity before processing any request.

   Input Sanitization: All API inputs are validated against strict Zod schemas before processing. This includes enforcement of email formats, password complexity requirements, and data structure validation, protecting against injection attacks and malformed payloads.

   Password Protection: User passwords are hashed using Bcrypt with salt before storage. Authentication attempts verify passwords against stored hashes using constant-time comparison to prevent timing-based attacks.

   Encryption: All communication between frontend and backend is protected using HTTPS in production environments, safeguarding data in transit from unauthorized access.

D. **Logging and Monitoring:** To ensure compliance and detect potential security incidents, the system implements request tracing through custom middleware. Each incoming request receives a unique correlation identifier, enabling end-to-end tracking through the system. This capability allows administrators to trace authentication events, monitor API access patterns, and respond rapidly to suspicious activities.

By implementing this comprehensive security framework, the Agentic Clinical Trial Operations Assistant ensures that only authenticated users can access sensitive clinical trial data and perform critical operations within the system. This security model enhances trust and protects the integrity of the application, ensuring a safe and secure user experience for clinical research teams.

VIII. Conclusion

ClinOps is an AI-powered clinical trial management platform designed to streamline the planning, execution, and monitoring of clinical research projects. The system architecture utilizes Next.js 15 with React 19 for the frontend, Node.js for the backend API layer, and PostgreSQL with Prisma ORM for data persistence. Integration with Google Gemini AI via @google/generative-ai enables intelligent content generation and context-aware recommendations throughout the platform.

The platform provides comprehensive functionality including AI-driven dashboard generation, interactive timeline and workflow visualization, context-aware chat assistance with multiple personas, and real-time widget generation. These capabilities ensure clinical trial managers receive tailored insights to enhance research efficiency and regulatory compliance. The system serves various stakeholders including clinical researchers, project managers, and regulatory teams by providing intelligent analysis of trial data, automated documentation generation, and comprehensive project tracking.

Key technologies employed in the project include Next.js 15, React 19, PostgreSQL, Prisma ORM, Google Gemini AI, and Recharts for data visualization. The system underwent rigorous testing using Jest for full stack validation, achieving high code coverage exceeding 80% across critical paths including authentication, AI services, and dashboard components. This comprehensive testing approach ensures reliability and performance across all system layers.

API security implementation employs JWT with HTTP-Only cookies, Bcrypt password hashing, and Zod schema validation to ensure secure authentication and authorization. These measures protect sensitive clinical trial data and maintain system integrity through request tracing with correlation identifiers. The security framework provides multi-layer verification and real-time monitoring capabilities to detect and respond to potential security incidents.

In conclusion, ClinOps delivers a comprehensive, secure, and efficient solution for organizations seeking to improve clinical trial management and research productivity. By combining advanced AI technology with robust security practices and modern web technologies, the platform provides clinical research teams with powerful tools to manage complex trial operations while maintaining data integrity and regulatory compliance.

Acknowledgment

We extend our heartfelt appreciation to everyone who contributed to the successful development of the Agentic Clinical Trial Operations Assistant. We are profoundly grateful to Prof. Rakesh Ranjan for his unwavering mentorship, insightful guidance, and constructive feedback throughout this project. His expertise and encouragement have been instrumental in navigating the complexities of this endeavor and achieving our objectives.

We acknowledge the collaborative spirit of our colleagues and peers whose thoughtful critiques and suggestions have enriched the quality and depth of this work. Their perspectives have challenged us to refine our approach and deliver a more robust solution.

Our sincere thanks go to the open-source community and the engineering teams behind Next.js, React, PostgreSQL, Prisma ORM, Google Gemini AI, and Jest. Their innovative technologies and comprehensive documentation provided the foundation upon which this platform was built, enabling us to focus on solving domain-specific challenges in clinical trial management.

We are also grateful to the institutions and individuals who provided the necessary resources, infrastructure, and supportive environment that made this research and development possible. Their commitment to fostering innovation has been invaluable to our success.