International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Smart Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

DevSync OS – Unified Engineering Lifecycle Orchestrator: A Centralized Web-Based Platform for Software Development Workflow Integration

Vrushali P. Ghaywankar
Department of Science and Technology, G.H.Raisoni Skill Tech University, Nagpur, India

ABSTRACT

Modern software engineering teams depend on a fragmented set of tools to manage planning, development, version control, collaboration, and monitoring activities. While each tool is effective individually, the lack of seamless integration among them results in fragmented workflows, limited visibility, and increased operational complexity. This paper presents DevSync OS — Unified Engineering Lifecycle Orchestrator, a centralized web-based platform designed to integrate and streamline the complete software development lifecycle (SDLC) into a single cohesive environment. The system connects project lifecycle management, GitHub repository activity, workflow visualization, and team collaboration through a unified operational interface. Built on the MERN stack (MongoDB, Express.js, React.js, Node.js), DevSync OS integrates with GitHub via the Octokit API for real-time repository synchronization. Workflow and system architecture visualization are powered by React Flow and Mermaid.js. The system implements role-based access control (RBAC) through JSON Web Tokens (JWT) to ensure secure, structured access across Administrator, Project Manager, and Developer roles. Test results demonstrate API response times under 200ms, 100% database uptime during load testing, and stability for up to 20 concurrent simulated users. DevSync OS successfully reduces context switching, improves transparency, and increases team coordination, offering a scalable solution aligned with modern Agile and DevOps engineering practices.

Keywords: Engineering Lifecycle Management; MERN Stack; GitHub Integration; Workflow Visualization; Role-Based Access Control; DevOps; Agile Software Development.

1. Introduction

The rapid growth of software-based systems has significantly increased the complexity of modern software development. Engineering teams are required to manage multiple activities simultaneously — requirement analysis, system design, coding, testing, version control, and collaboration. To support these activities, teams depend on various specialized tools and platforms. While these tools improve efficiency for individual tasks, the absence of a unified system results in fragmented workflows and limited visibility across the development lifecycle [1].

In a typical development environment, project planning, code development, repository management, documentation, and progress tracking are handled using separate applications. This separation creates

challenges such as frequent context switching, scattered project information, difficulty in monitoring real-time progress, and reduced coordination among team members. As project size and team distribution increase, these challenges become more prominent and negatively impact productivity, quality, and delivery timelines [2].

Existing solutions such as Jira, Trello, and GitHub each address individual aspects of software development, but they fail to provide an end-to-end orchestration of the complete engineering lifecycle. There is a clear need for a centralized system that integrates project workflows, repository activities, system architecture, and team collaboration into a single platform [3].

This paper presents DevSync OS — Unified Engineering Lifecycle Orchestrator, a web-based platform that resolves these challenges by consolidating all critical lifecycle components. The system provides a single operational interface where project data, workflows, repository activity, and team interactions are seamlessly integrated, enabling teams to maintain a clear and consistent view of project progress and system structure.

1.1. Motivation

Software engineering teams working in distributed environments face the compounding challenge of maintaining coherence across multiple disparate tools. Context switching between project management platforms, code repositories, documentation systems, and communication tools consumes valuable engineering time and increases the risk of information loss. Furthermore, the absence of a real-time, unified view of the development lifecycle makes it difficult for project managers and stakeholders to identify bottlenecks, track contributions, and make timely decisions.

The motivation for DevSync OS stems from the need to build a system that acts as a single 'operating system' for the software engineering team — one that centralizes lifecycle data, automates integration between development components, and provides interactive visualizations that make complex project states understandable at a glance.

1.2. Contribution

The key contributions of this paper are as follows:

- Design and implementation of a unified full-stack MERN platform for end-to-end engineering lifecycle management.
- Real-time GitHub repository synchronization via the Octokit API, bridging the gap between project planning and actual code execution.
- Interactive workflow and system architecture visualization using React Flow and Mermaid.js, enabling dynamic, data-driven lifecycle diagrams.
- Implementation of a secure role-based access control system using JWT authentication across Administrator, Project Manager, and Developer tiers.
- A backend caching strategy for external API data that reduces average GitHub synchronization latency to 1.2 seconds after an initial cache warm-up.
- A comprehensive test report validating functional, performance, and compatibility requirements of the integrated system.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 describes the system architecture and methodology. Section 4 details the implementation. Section 5 presents experimental results and analysis. Section 6 concludes with future scope.

2. Related Work

The literature on software engineering lifecycle management, project coordination tools, and integrated development platforms reveals several important findings relevant to the design of DevSync OS.

Sommerville [1] emphasizes that effective software engineering requires structured processes, clear communication, and continuous monitoring across all lifecycle stages. The study highlights that fragmented tools and poor coordination frequently result in delays, increased defects, and reduced productivity — establishing the foundational need for unified lifecycle management systems.

Research on version control systems, particularly Git and GitHub, confirms that repository-based collaboration significantly improves team coordination and code quality [4]. GitHub's features — commit tracking, pull requests, and contributor analytics — enhance transparency and accountability within development teams. However, these platforms focus exclusively on source code management and do not offer complete project lifecycle visibility or integration with planning tools.

Widely used project management tools such as Jira, Trello, and Asana are effective for task tracking and workflow management [5]. Literature related to Agile and Scrum methodologies highlights their effectiveness in iterative development and team collaboration. Despite these strengths, these tools operate independently from version control systems, creating disconnected views of planning and actual development activities.

Visualization frameworks such as Mermaid.js and React Flow have been studied for their ability to represent system architecture and workflows in an intuitive manner [6]. Research shows that visual models significantly improve understanding of complex systems, especially for non-technical stakeholders. However, most existing systems use visualization only for documentation purposes and not as an active, data-driven component of lifecycle management.

Recent studies on full-stack web development using the MERN stack highlight its suitability for building scalable and maintainable applications [7]. The combination of React.js for dynamic UIs, Node.js and Express.js for backend services, and MongoDB for flexible data storage is widely recommended in modern web application development literature.

From the reviewed literature, it is evident that while several tools and frameworks address individual aspects of software development, there is a clear gap: no single platform integrates project management, version control activity, and dynamic workflow visualization into a cohesive engineering lifecycle orchestration system. This research gap forms the foundation for DevSync OS.

3. System Design and Methodology

3.1. Problem Statement

Modern software development teams rely on a collection of independent tools — project trackers, code repositories, documentation systems, and communication platforms. The critical problem is the absence of a unified platform that provides complete, real-time visibility into the entire engineering lifecycle. This fragmentation forces developers and managers to context-switch between applications, complicates traceability from requirements to deployed code, and obscures the true status of projects from stakeholders.

DevSync OS addresses this problem by designing a centralized orchestration layer that connects all lifecycle phases — planning, development, testing, and deployment — into a single, coherent system backed by live data from code repositories and a responsive visualization engine.

3.2. System Architecture

The architecture of DevSync OS follows a three-tier design comprising a Presentation Layer, an Application Layer, and a Data Layer, with an additional external integration channel to GitHub.

The Presentation Layer is built with React.js and styled using Tailwind CSS. It provides the Dashboard, Project Views, Workflow Visualization, and Activity Feed as key UI modules. Visualization components leverage React Flow for interactive node-based lifecycle diagrams and Mermaid.js for text- based architectural representations.

The Application Layer is implemented with Node.js and Express.js. It handles all business logic including JWT-based authentication and authorization, project management operations, engineering lifecycle state transitions, and GitHub API integration via the Octokit library. A Controller-Service-Route pattern is used to maintain separation of concerns and code modularity.

The Data Layer employs MongoDB with Mongoose ODM. The schema models include USER, PROJECT, WORKFLOW_LIFECYCLE, REPOSITORY, and ACTIVITY entities, with appropriate foreign key relationships enforced at the application layer. GitHub repository data is cached in a dedicated data store (D4: Repository Activity Data) and refreshed at configured intervals to reduce API call frequency.

3.3. Core Modules

DevSync OS is divided into five interconnected functional modules:

Authentication and Security Module: Manages user identity through JWT stored in secure HTTP-only cookies, supporting Secure Login, Registration, and Role-Based Access Control (RBAC) across Administrator, Project Manager, and Developer roles.

Project Management Module: Allows Project Managers to define project scopes and deadlines, track current project status (Active, On Hold, Completed), and assign team members to specific projects.

Engineering Lifecycle Orchestrator: Maps work into sequential lifecycle phases (Planning → Design

→ Development → Testing → Deployment) and renders these phases as dynamic, interactive diagrams using React Flow and Mermaid.js.

GitHub Integration Module: Uses the Octokit library to fetch real-time repository metadata, commit history, pull request statuses, and contributor analytics, mapping this data to the project lifecycle context.

Analytics and Reporting Module: Aggregates data across all modules into a centralized Dashboard, provides chronological Activity Feeds for full traceability, and generates visual charts to identify bottlenecks.

4. Implementation

4.1. Technology Stack

Category	Technology / Tool	Purpose
Frontend	React.js, Tailwind CSS	Responsive UI and styling
Frontend	React Flow, Mermaid.js	Lifecycle and architecture visualization
Backend	Node.js, Express.js	RESTful API and business logic
Database	MongoDB, Mongoose	Data storage and schema management
Auth	JSON Web Tokens (JWT)	Stateless, role-based authentication
API Integration	GitHub Octokit	Repository and commit synchronization
Dev Tools	VS Code, Postman, Git	Development, testing, version control

Table 1. Technology stack of DevSync OS.

4.2. Development Methodology

An Agile-Scrum approach was adopted for development. The project was divided into iterative sprints, each focused on a specific module: the Dashboard and Authentication layer were built first, followed by the Project Management module, GitHub Integration, and finally the Workflow Visualization and Analytics modules. Regular sprint retrospectives enabled continuous UI/UX improvements, particularly in the dark-themed interface and visualization components.

4.3. Key Implementation Details

Frontend components were built as reusable React.js units including Project Cards, Status Badges, and Navigation Bars. React Query and custom Axios-based hooks manage data fetching and ensure real- time UI synchronization with the backend. The Engineering Lifecycle visualization is rendered by passing project stage data from MongoDB directly into React Flow, which maps these stages into a visual node- based diagram.

Backend controllers follow the Controller-Service-Route pattern. The GitHub Sync controller performs asynchronous fetches via the @octokit/rest library, processes raw JSON data, and delivers filtered results (commits, branches, contributors) to the frontend. Password hashing is performed using bcryptjs, and all API routes are protected by JWT middleware that verifies tokens and enforces role-based access.

A significant optimization involved transitioning from direct GitHub API calls on every page load to a caching strategy where repository data is stored in MongoDB and refreshed at specific intervals. This reduced GitHub API call frequency and eliminated the risk of hitting GitHub's rate limits under normal usage conditions.

5. Experimental Results and Analysis

5.1. Functional Test Results

A multi-layered testing strategy was applied including unit testing of individual API endpoints via Postman, integration testing of the MERN stack data flow, functional user-journey testing of role-based access and lifecycle transitions, and cross-browser UI compatibility testing.

Test Case ID	Feature Under Test	Expected Outcome	Actual Outcome	Status
TC-01	User Registration	Data encrypted, saved in MongoDB	User created; password hashed	Pass
TC-02	JWT Authentication	Valid token grants access to protected routes	Access granted; invalid token redirected	Pass
TC-03	Project Creation	Project metadata stored and visible on dashboard	Project saved and displayed in UI	Pass
TC-04	GitHub Sync	Octokit fetches repo commits and contributors	Real-time data fetched and mapped to UI	Pass
TC-05	Role Access Control	Developer cannot access Manage Users page	Access denied with 403 Forbidden	Pass

Table 2. Unit and Functional Test Results.

5.2. Performance and Integration Results

Metric	Description	Result
API Response Time	Average time for backend to return project list	< 200ms
GitHub Sync Latency	Time taken to fetch and render repository data	1.2s (optimized via caching)
Database Connectivity	MongoDB connection stability under load	100% uptime during testing
Concurrent Users	System stability with multiple active logins	Stable up to 20 simulated users

Table 3. Performance and Integration Summary.

5.3. UI and Compatibility Test Results

Environment / Feature	Test Description	Observation	Status
Google Chrome	Rendering of React Flow diagrams	Smooth interaction and node dragging	Pass
Mobile (iOS/Android)	Responsive layout using Tailwind CSS	Sidebar collapses; cards stack vertically	Pass
Dark Mode Contrast	Readability of charts and activity logs	High visibility with chosen color palette	Pass
Form Validation	Handling of empty fields in Add Project	Error messages appear immediately	Pass
Mozilla Firefox / Edge	Cross-browser diagram rendering	Mermaid.js and React Flow render correctly	Pass

Table 4. UI and Compatibility Test Results.

5.4. Comparative Analysis of Development Tools

DevSync OS was evaluated against the closest existing solutions to highlight its differentiated value proposition.

Feature	GitHub	Jira / Trello	DevSync OS
Lifecycle Management	No	Partial	Full (5 stages)
Repository Sync	Native	Plugin only	Real-time via Octokit
Workflow Visualization	No	Board only	Interactive node graphs
Role-Based Access	Limited	Yes	JWT-based RBAC
Unified Dashboard	No	Partial	Fully centralized
Dark Mode UI	Yes	Limited	Full premium dark theme

Table 5. Comparative analysis: DevSync OS vs. existing tools.

6. Conclusion and Future Scope

6.1. Conclusion

This paper has presented DevSync OS — Unified Engineering Lifecycle Orchestrator, a centralized web-based platform that successfully addresses the challenge of fragmented engineering workflows in modern software development environments. By unifying project management, version control tracking, lifecycle orchestration, and workflow visualization into a single MERN-based system, DevSync OS provides engineering teams with the transparency, structure, and coordination necessary to deliver high-quality software efficiently.

The system demonstrates strong functional correctness across all test cases, API response times below 200ms, and stable operation under concurrent user loads. The GitHub Octokit integration effectively bridges the gap between project planning and actual development activity, while the React Flow and Mermaid.js visualization engine translates complex lifecycle data into intuitive, interactive representations. JWT-based RBAC provides a secure, scalable access control mechanism across user tiers.

DevSync OS goes beyond conventional project management tools by acting as a specialized orchestrator aligned with modern Agile and DevOps practices, effectively reducing operational friction and providing engineers and stakeholders with real-time visibility into the engineering lifecycle.

6.2. Limitations

The current version of DevSync OS carries several identified limitations. First, the platform is exclusively integrated with GitHub; it does not yet support GitLab, Bitbucket, or Azure DevOps. Second, while a caching layer mitigates GitHub API rate limits, very large organizations with thousands of repositories may still experience synchronization delays. Third, the system lacks built-in real-time messaging or commenting capabilities, requiring teams to rely on external communication tools. Fourth, the lifecycle module does not trigger actual deployments; it functions as a management and visualization layer rather than a CI/CD execution engine. Fifth, complex React Flow diagrams are better experienced on desktop or tablet screens, as small mobile screens limit the full interactivity of node-based visualizations.

6.3. Future Scope

Several planned enhancements will extend the platform's capabilities. Multi-platform VCS integration with GitLab, Bitbucket, and Azure DevOps will broaden applicability across hybrid cloud environments. Direct integration with CI/CD tools including GitHub Actions, Jenkins, and CircleCI will enable users to monitor build statuses, test coverage, and deployment logs within the Lifecycle module, creating a true single pane of glass for DevOps.

Real-time collaborative features including live commenting on lifecycle stages and in-app push notifications for PR approvals and build failures will improve team synergy. Advanced predictive analytics leveraging historical development cycle data will enable the system to forecast project bottlenecks and estimate completion dates. Automated technical documentation generation from system metadata and a native React Native mobile application for on-the-go access are also planned for future iterations.

References

1. I. Sommerville, Software Engineering, 10th ed. Boston, MA: Pearson Education Limited, 2016.
2. P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile Software Development Methods: Review and Analysis," VTT Technical Research Centre of Finland, 2002.
3. G. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, 3rd ed. Addison-Wesley Professional, 2012.
4. GitHub, "Octokit.js: The all-batteries-included GitHub SDK for Browsers, Node.js, and Deno," GitHub Docs, 2025. [Online]. Available: https://github.com/octokit/octokit.js
5. K. Beck et al., "Manifesto for Agile Software Development," 2001. [Online]. Available: https://agilemanifesto.org
6. xyflow, "React Flow Documentation: Node-based UI for React," 2025. [Online]. Available: https://reactflow.dev

7. Mermaid.js, "Mermaid: Diagramming and charting tool," 2025. [Online]. Available: https://mermaid.js.org
8. MongoDB Inc., "MongoDB Documentation: Manual and Data Modeling," 2025. [Online]. Available: https://www.mongodb.com/docs/manual/
9. Meta Open Source, "React Documentation: Managing State and Component Lifecycle," 2025. [Online]. Available: https://react.dev
10. OpenJS Foundation, "Node.js v20.x Documentation," 2025. [Online]. Available: https://nodejs.org/docs/
11. ExpressJS, "Express - Node.js web application framework," 2025. [Online]. Available: https://expressjs.com
12. Tailwind Labs Inc., "Tailwind CSS Documentation: Utility-First Fundamentals," 2025. [Online]. Available: https://tailwindcss.com/docs
13. Auth0, "JSON Web Token (JWT) Introduction," 2025. [Online]. Available: https://jwt.io/introduction
14. A. Chaube, "ACO-Enhanced Siamese Networks for Robust Feature Matching in Copy-Move Image Forgery Detection," 2024 International Conference on Artificial Intelligence and Quantum Computation-Based Sensor Application (ICAIQSA), Nagpur, India, 2024, pp. 1-6, doi: 10.1109/ICAIQSA64000.2024.10882433.
15. Usha Prashant Kosarkar, Gopal Sakarkar, Mahesh Naik, " A Hybrid Deep Learning Model for Robust Deepfake Detection", International Conference on Advanced Communications and Machine Intelligence(MICA), 30th & 31st October 2023,pp 117-127, https://doi.org/10.1007/978-981-97-6222-4_9