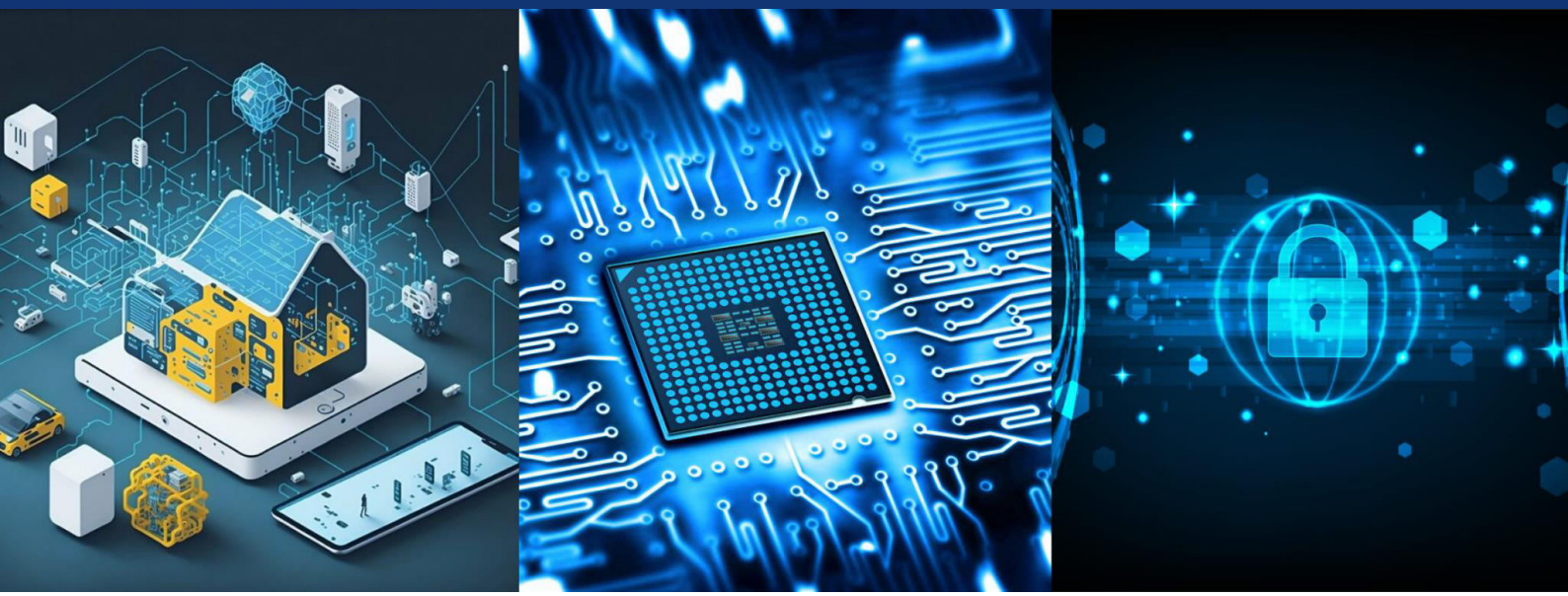
IJIRCCE

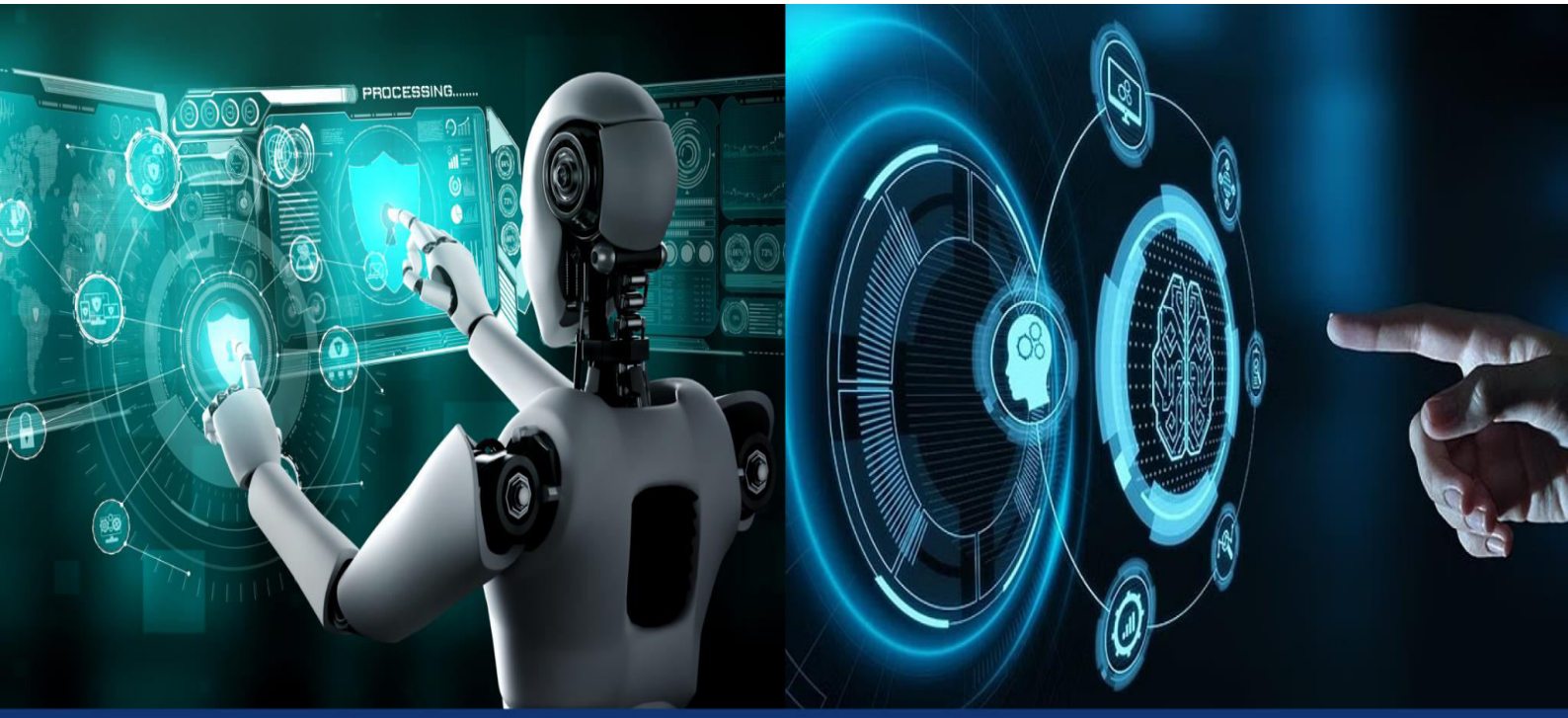
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.927

Volume 14, Issue 7, July 2026

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407043

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

NexusCode: Unified AI-Powered Collaborative IDE and Project Management Platform

Thipparthi Raviteja, Dr. V. Uma Rani

Post-Graduate Student, Department of Computer Science Engineering, Computer Networks and Information Security, Jawaharlal Nehru Technological University Hyderabad, Hyderabad, India

Professor, Department of Computer Science Engineering, Jawaharlal Nehru Technological University Hyderabad, Hyderabad, India

ABSTRACT: Contemporary software engineering practice is marked by a reliance on disjointed tools, encompassing project tracking utilities, code editing environments, messaging platforms, artificial intelligence helpers, and administrative dashboards, each governed by its own authentication layer and data repository. This arrangement amplifies context-switching overhead and constrains AI capabilities due to a lack of holistic project awareness. NexusCode offers a consolidated, AI-augmented, role-sensitive platform that unifies essential development workflows. The platform is realized through three independently deployable TypeScript services: a React and Vite web frontend, also packaged as an Electron desktop client; a Node.js Express API backend; and a dedicated authentication service. It furnishes three role-specific dashboards tailored to project managers, developers, and administrators. The Project Manager Dashboard encompasses views addressing kanban, backlog, sprint planning, roadmap, and analytics, underpinned by heuristic models for velocity forecasting, completion prediction, story-point estimation, and burnout risk assessment. The Developer Dashboard delivers a Monaco Editor-based browser IDE incorporating an embedded terminal, debugging support, Source Control Management, inline AI assistance, and autonomous agent capability. The Administrator Dashboard furnishes tooling for user administration, system observability, and cost analytics. A collaboration layer facilitates secure messaging, WebRTC-based communication, and Y.js CRDT-driven real-time editing. Authentication is handled through JWT, TOTP-based two-factor authentication, and RBAC enforcement, validated via automated testing and CI/CD pipelines.

KEYWORDS: integrated development platform, AI-driven software engineering, role-based access control, Monaco Editor, Y.js CRDT, WebRTC, multi-provider AI orchestration, JWT, TOTP.

I. INTRODUCTION

Modern software development teams operate within a fragmented ecosystem of digital tools, including project management systems, code editors, communication platforms, and version control systems. In a typical workflow, developers frequently switch between these tools to manage tasks, write and test code, collaborate with team members, and track progress. Such repeated context switching disrupts cognitive flow, forcing developers to reconstruct their working context multiple times a day, leading to reduced productivity and increased mental overhead.

The fundamental issue is not the inadequacy of individual tools—platforms like Jira, Visual Studio Code, and Microsoft Teams are highly capable—but their inability to share context and provide a unified view of development activities. This fragmentation introduces systemic inefficiencies that cannot be resolved by improving any single tool in isolation.

To address this challenge, this project proposes NexusCode, an AI-enabled, unified software engineering platform that integrates project management, a browser-based IDE, AI assistance, real-time collaboration, and administrative monitoring into a single application. Built using a modular three-service architecture, NexusCode provides role-based dashboards for project managers, developers, and administrators. By combining these functionalities under one system, NexusCode aims to eliminate fragmentation, reduce context switching, and enhance overall development efficiency.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. LITERATURE SURVEY

Paper 1: Grounded Copilot: How Programmers Interact with Code Generating Models
Abstract:
By watching twenty participants complete programming tasks in four different languages, Barke, James, and Polikarpova (2023) provided the first grounded-theory investigation of how programmers actually engage with an AI coding helper. The main conclusion of the study is that developer interaction is bimodal: in exploration mode, the developer utilizes the assistant to locate alternatives; in acceleration mode, the developer uses the assistance to move more quickly. The two interaction modes that NexusCode's IDE AI surfaces are specifically built around are ghost-text inline completion for acceleration mode and the conversational AI chat panel for exploration mode, both of which are based on real workspace context. This makes the study pertinent.

Paper 2: Near Real-Time Peer-to-Peer Shared Editing on Extensible Data Types
Abstract:
Nicolaescu, Jahns, Derntl, and Klamma (2016) introduced a conflict-free replicated data type (CRDT) system that allows peer-to-peer collaborative editing in the browser in almost real-time without the need for a central server to mediate merge order. The study shows that several clients can apply concurrent updates in any sequence while always converging to the same final document state thanks to CRDT-based synchronization. The theoretical and practical basis of the Y.js library, which NexusCode employs to provide multi-user collaborative editing of the same source file inside the Monaco-based IDE, is the framework that makes this study immediately applicable.

Paper 3: The NIST Model for Role-Based Access Control: Towards a Unified Standard
Abstract:
In order to provide a single consensus model for Role-Based Access Control (RBAC), Sandhu, Ferraiolo, and Kuhn (2000) reconciled previous RBAC concepts, commercial product implementations, and research prototypes. The study proposes that a uniform standard lowers uncertainty for implementers and divides RBAC into four increasingly capable levels: flat RBAC, hierarchical RBAC, limited RBAC, and symmetric RBAC. The three-role access model used in NexusCode—Project Manager, Developer, and Administrator, each limited to their own routes at the middleware layer—is a direct application of the flat RBAC level outlined in this foundational model, which makes this study important to the current endeavor.

III. PROPOSED METHODOLOGY

The objective of the proposed NexusCode platform is to build a unified collaborative IDE and project management workspace powered by multi-provider AI agents. Figure 1 illustrates the overall Proposed System Architecture. The system is designed with three core layers: Client Workspace, Gateway Orchestrator, and AI Integration engine.

A. System Architecture

NexusCode's modular, pipeline-based architecture consists of the following components:

Authentication and Identity Verification

- Compiles user credentials and compares them to the specialized authentication service.
- Issues a signed JWT with the user's role claim, verifies the password hash, and optionally verifies a TOTP code.
- Captures the audit log and the session in Redis for traceability and revocation.

Project and Workspace Data Processing

- Uses Knex.js to load project, sprint, issue, and module records from MySQL.
- Verifies and arranges structured data so dashboards and AI context builders can use it.

Role-Based Routing and Dashboard Rendering

- Before any controller logic is executed, RBAC middleware verifies the role claim on each protected request.
- The dashboard and tab set that correspond to the authorized role—PM, Developer, or Admin—are rendered by the frontend.

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407043

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

AI Context Assembly and Workspace Intelligence

- Before each AI call, context builders compile real sprint, problem, and team data.
- A multi-provider gateway with automated fallback that routes to Groq, NVIDIA NIM, or OpenRouter.

Collaboration and Real-Time Synchronization

- Group chats, channel communication via WebSocket, and end-to-end encrypted direct messages.
- COTURN relay fallback for NAT bypassing in WebRTC audio and video calls.
- Y.js CRDT document state with presence broadcasting and an awareness-based cursor.

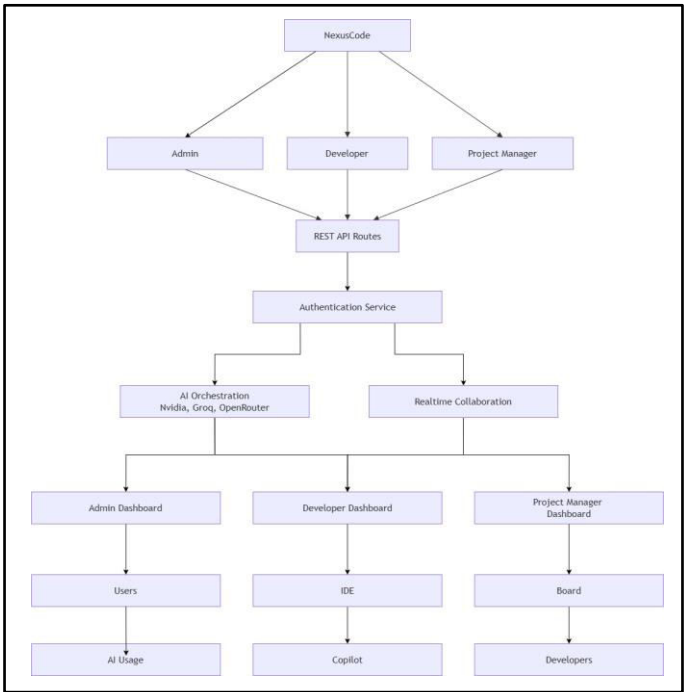


Figure 1: Proposed System Architecture

B. Design Considerations

Design considerations prioritized security, consistency, isolation, and AI resilience. JWT with 15-minute access tokens and RFC 6238 TOTP via Speakeasy enforce authentication. Monaco Editor provides a standards-compliant coding environment. Y.js CRDT enables conflict-free collaborative editing. A three-provider AI failover chain — Groq, OpenRouter, NVIDIA NIM — ensures service continuity.

IV. IMPLEMENTATION MODULES

The platform is implemented across six core workspaces, each taken from the corresponding system modules defined in the project documentation:

A. Authentication and Role Management

- The system initially verifies user credentials before granting secure, role-aware access.
- The authentication process involves an assigned role (PM, Developer, or Admin), an email address, a password, and an optional TOTP code.
- Access to each subsequent request is controlled by JWT issuance, Redis session caching, and RBAC middleware enforcement.
- A personalized, role-appropriate workspace is rendered using the role assigned at registration, which can only be changed by an Administrator.

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407043

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

B. Project and Sprint Data Processing

The project and sprint data processing module is necessary to supply organized project data for the dashboards and the AI orchestration layer. The authentication process involves an assigned role (PM, Developer, or Admin), an email address, a password, and a TOTP code. For the recommendation and reporting logic to use project, sprint, issue, and module records, they must be loaded, verified, and arranged. With parameters like issue status, story points, priority, assignee, sprint dates, and module ownership, NexusCode employs a relational MySQL schema.

Steps Involved:

1. **Data loading:** Knex.js parameterized queries are used to load project, sprint, and issue records from MySQL.
2. **Data validation:** Before being saved or returned, necessary fields are verified at the controller layer.
3. **Attribute organization:** For kanban, backlog, and reporting views, crucial fields like story points, priority, and sprint association are maintained.
4. **Audit logging:** The audit log documents notable changes including status updates and sprint creation.

C. Developer Dashboard and IDE Module

In terms of implementation surface, NexusCode's Developer Dashboard is the largest component. The Monaco Editor lifecycle, all WebSocket connections for the terminal, language server, and debugger, the AI panel state, the source control panel, the keybinding registry, the theme system, and the command palette are all managed by the IDE shell, which essentially acts as the coordinator for everything operating below it.

- **Monaco Editor:** Loads files as discrete models with distinct undo histories. After a short typing delay, inline AI ghost-text shows up and is accepted via Tab Key.
- **Terminal:** Streams output to the browser and creates a real pseudo-terminal on the server. Several tabs operate concurrently as distinct, independent sessions.
- **Language Server Protocol:** Launches a language server for each type of file and establishes a WebSocket connection with the editor to automatically provide completions, hover docs, diagnostics, and go-to definitions.
- **Source Control Management:** Shows modified files, staging, commit, and push controls right within the IDE by executing git commands on the server.
- **AI Surfaces inside the IDE:** Three modes — ghost-text inline completion, a session-persistent chat panel pre-loaded with sprint and issue context, and an agent mode that plans and executes multi-step coding tasks autonomously.

D. Administrator Dashboard Module

The Administrator Dashboard guarantees governance, visibility, and effective management of platform activities by enabling centralized control over users, AI operations, and system monitoring.

- **User Management:** Ensures secure access and appropriate role-based permissions throughout the platform by enabling administrators to establish, amend, deactivate, assign roles, and enforce authentication controls.
- **AI Interaction Logs:** Enables auditing, debugging, and transparency of AI operations by offering a filterable view of all AI requests, including model, provider, tokens, latency, and status.
- **Cost and Usage Analytics:** Enables cost analysis, resource management, and well-informed decision-making for the use of AI by aggregating token consumption and usage patterns across models and workspaces.
- **Alerts and Monitoring:** Enables threshold-based notifications on mistake rates and usage limitations, assisting administrators in early anomaly detection and proactive system stability maintenance.

E. AI-Powered Workspace Intelligence

The AI-powered workplace intelligence module is responsible for producing contextual support for each role based on actual project and sprint data. This is the core intelligence layer of the system and is essential to transforming AI support from generic to truly helpful. The AI pipeline for this project uses heuristic scoring, multi-provider orchestration, and context assembly.

- **Context Assembly:** Before each AI call, specialized context builders for the PM, Dev, and Admin workspaces search the database for sprint objectives, issue descriptions, and team activity.
- **Multi-Provider Orchestration:** Depending on the kind of task, a provider gateway sends requests to Groq, NVIDIA NIM, or OpenRouter with automatic fallback in case of an error or rate limit.
- **Inline and Conversational Surfaces:** A streaming AI conversation panel, an autonomous agent/planning mode with tool-calling, and ghost-text inline completion are all customized to specific latency and complexity requirements.

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407043

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

F. Collaboration and Progress Tracking

Through the reporting views on the PM dashboard, the system tracks engineering progress and facilitates real-time collaboration across all three roles. This module improves the platform's usefulness for daily team coordination by combining sprint-level reporting with messaging, calling, and collaborative editing.

- **Messaging:** WebSocket is used to transmit direct messages, group chats, and channel messages that are encrypted end-to-end utilizing AES-GCM and ECDH key exchange.
- **Calling and Screen Sharing:** DTLS-SRTP encryption, ICE negotiation, and COTURN relay fallback are used to establish WebRTC audio and video calls; screen sharing is enabled during calls.
- **Collaborative Editing:** Y.js CRDT documents use awareness-based cursor sharing to synchronize concurrent updates to the same file across several developers in the IDE.
- **Presence Tracking:** All connected users' online, offline, busy, and away statuses are monitored via a heartbeat-based presence system.
- **Sprint Progress Reporting:** The PM Dashboard's Reports view compiles team-wide velocity, completion trends, and burnout indicators.

V. SIMULATION RESULTS AND SCREENS

To validate the NexusCode platform, extensive simulation studies were conducted on local environments. Workspaces (Admin, PM, Developer) were tested for authentication flows, project backlog updates, AI completions, and collaborative session syncs.

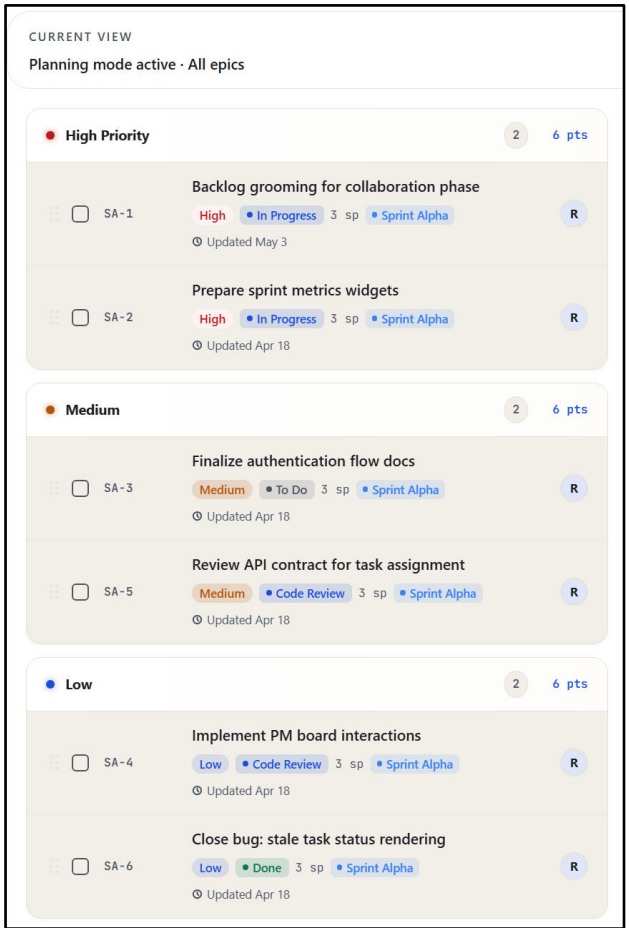


Figure 2: PM Dashboard – Project Manager Backlog

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407043

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

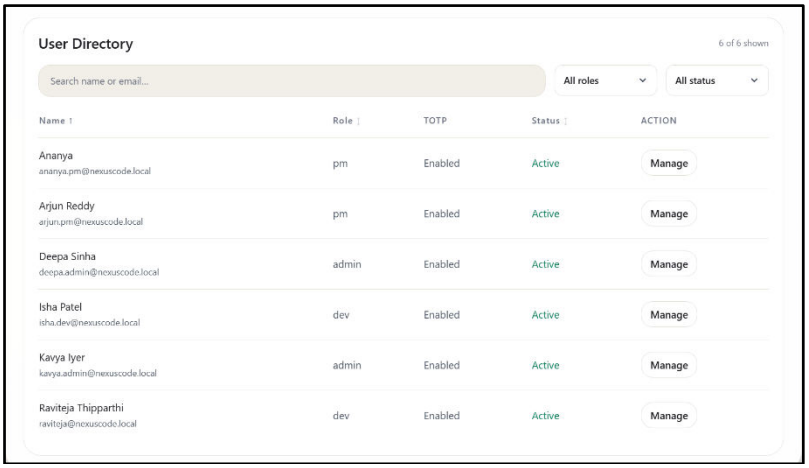


Figure 3: Admin Dashboard – Admin User Management

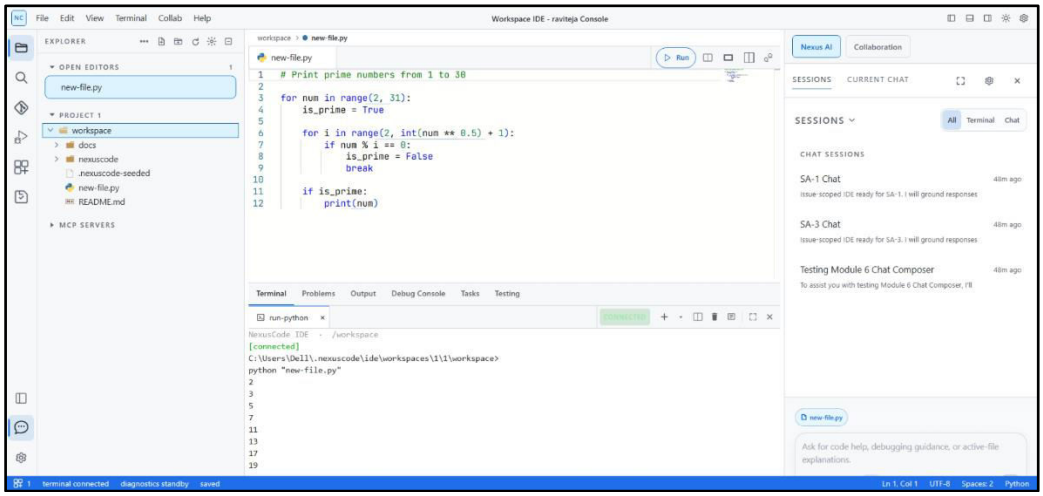


Figure 4: Developer IDE – Monaco Editor and Workspace

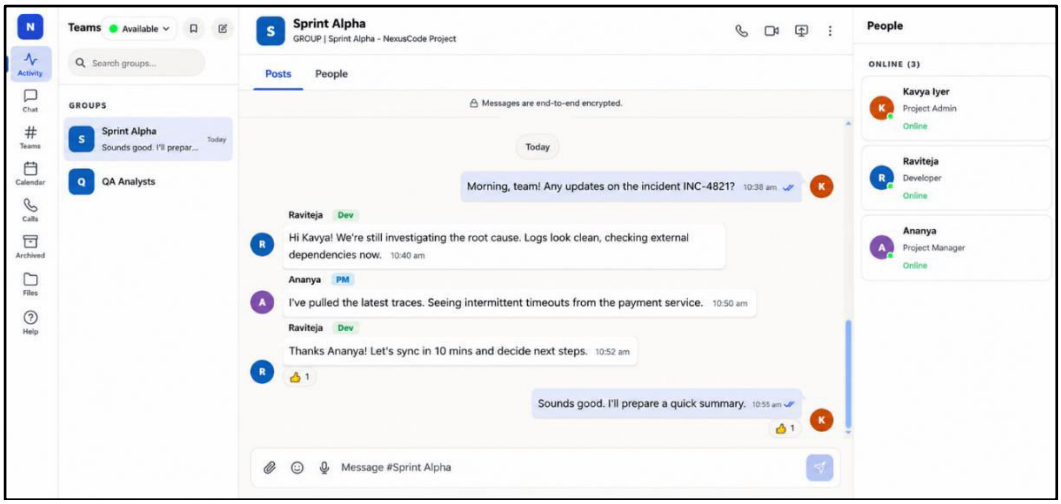


Figure 5: Collaboration Module – Teams Messaging and Presence

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

VI. CONCLUSION AND FUTURE WORK

The core rationale behind NexusCode is simple: development teams lose efficiency when forced to switch between multiple disconnected tools, while a unified workspace preserves focus and continuity. The problem is not the individual tools themselves, but the lack of integration between them. NexusCode addresses this by combining project management, a browser-based IDE, context-aware AI, and real-time collaboration into a single authenticated platform.The system is built on a three-service architecture covering authentication, backend logic, and frontend interaction. It incorporates modules for collaboration, AI orchestration, IDE functionality, project and sprint management, and role-based dashboards. Security is enforced at the middleware level using RBAC, ensuring strict access control independent of the UI. Authentication is handled centrally through JWT and TOTP.A key differentiator is the AI orchestration layer, which uses context builders to supply real project and sprint data before generating responses. This is enhanced by multi-provider fallback across Groq, NVIDIA NIM, and OpenRouter. The collaboration layer integrates messaging, calling, and CRDT-based editing.Notably, the entire system is built using open-source technologies, demonstrating that production-grade platforms can now be developed without expensive infrastructure or licensing.

REFERENCES

[1] Barke, S., James, M. B.James, & Polikarpova, N. (2023). Grounded Copilot: How programmers interact with code-generating models. Proceedings of the ACM on Programming Languages, 7(OOPSLA1), Article 78, 85–111. https://doi.org/10.1145/3586030

[2] Nicolaescu, P., Jahns, K., Derntl, M., & Klamma, R. (2016). Near real-time peer-to-peer shared editing on extensible data types. Proceedings of the 19th International Conference on Supporting Group Work (GROUP 2016), 39–49. https://doi.org/10.1145/2957276.2957310

[3] Sandhu, R., Ferraiolo, David. F., & Kuhn, R. (2000). The NIST model for role-based access control: Towards a unified standard. Proceedings of the 5th ACM Workshop on Role-Based Access Control (RBAC 2000), 47–63. https://doi.org/10.1145/344287.344301

[4] M. Jones, J. Bradley, N. Sakimura (2015). JSON Web Token (JWT) (RFC 7519). Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc7519

[5] D. M'Raihi, S. Machani, M. Pei, J. Rydell (2011). TOTP: Time-Based One-Time Password Algorithm (RFC 6238). Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc6238

[6] I. Fette, A. Melnikov (2011). The WebSocket protocol (RFC 6455). Internet Engineering Task Force. https://www.rfc-editor.org/rfc/rfc6455

[7] W3C. (2023). WebRTC 1.0: Real-time communication between browsers (W3C Recommendation). World Wide Web Consortium. https://www.w3.org/TR/webrtc/

[8] Microsoft Corporation. (2015). Monaco Editor: The editor that powers VS Code. https://microsoft.github.io/monaco-editor/

[9] Microsoft Corporation. (2018). Language Server Protocol specification. https://microsoft.github.io/language-server-protocol/

[10] Microsoft Corporation. (2016). Debug Adapter Protocol specification. https://microsoft.github.io/debug-adapter-protocol/

[11] NIST Computer Security Resource Center. (2024). Role Based Access Control project page. National Institute of Standards and Technology. https://csrc.nist.gov/projects/role-based-access-control

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details