DevCollab: An AI-Powered Developer Collaboration Platform for Intelligent Team Formation Using Skill-Based Matching

Robin saraswat[1], Hardik[2], Himani[3], Rohit Chaudhary[4]
[1,2,3,4] *Department of Computer Science and Engineering Noida Institute of Engineering and Technology (Regional Campus) Greater Noida, India*

***Abstract*—In contemporary software engineering, the formation of effective development teams remains a critical yet largely unresolved challenge. Existing platforms such as LinkedIn, GitHub, and freelancing portals provide discovery mechanisms but lack intelligent, context-aware team formation features that account for complementary skill sets, project alignment, and real-time collaboration capabilities. This paper presents DevCollab, a novel AI-powered developer collaboration platform that introduces a swipe-based, skill-driven matching paradigm to facilitate efficient team formation. DevCollab integrates a multi-factor recommendation engine that evaluates developer profiles across skill vectors, interest graphs, and project compatibility scores to produce optimized team suggestions. The system is architected using a React and Tailwind CSS frontend, a high-performance Go (Golang) backend with the Gin framework, MongoDB for persistent data storage, and WebSocket-based real-time communication. Experimental projections and design evaluations indicate that DevCollab can significantly reduce the time-to-team metric, improve collaboration quality through complementary skill alignment, and enhance user engagement through its gamified interaction model. This paper details the system's architecture, matching methodology, implementation rationale, and future development trajectory, positioning DevCollab as a scalable solution for modern software development ecosystems.**

***Index Terms*—AI-based matching, developer collaboration, recommendation systems, skill-based team formation, real-time communication, developer networking, software engineering platforms**

I. INTRODUCTION

The modern software industry operates within highly collaborative ecosystems where the success of a project is intricately tied to the composition and synergy of its development team. Despite the proliferation of professional networking and freelancing platforms, the process of identifying and assembling the right team of developers remains fragmented, time-consuming, and largely dependent on manual evaluation. Developers seeking teammates for hackathons, open-source initiatives, or startup ventures frequently resort to generic social networks, job boards, or informal word-of-mouth referrals—mechanisms that are neither optimized for technical collaboration nor capable of evaluating compatibility at a granular level.

Platforms such as LinkedIn provide broad professional networking but are not designed with technical project collaboration in mind. GitHub exposes contribution histories but offers no active matching or team formation workflow. Freelancing platforms such as Upwork and Fiverr are transactional in nature and prioritize gig-based hiring over collaborative co-development. The result is a conspicuous gap in the developer tooling landscape: the absence of an intelligent, purpose-built platform that can match developers based on complementary skill sets, shared interests, and project-specific requirements.

This paper addresses the aforementioned gap through the design and development of DevCollab, a centralized AI-powered developer collaboration platform. DevCollab introduces a swipe-based, card-driven discovery model—inspired by the interaction paradigm popularized by modern consumer applications—adapted for technical team formation. At its core, DevCollab employs a multi-dimensional

matching algorithm that evaluates developers across skill vectors, interest graphs, and project compatibility metrics to generate highly relevant connection recommendations.

A. Motivation

The motivation for DevCollab stems from firsthand observation of challenges faced by developers during hackathons and open-source collaboration events. Team formation in these contexts is often ad hoc, resulting in imbalanced teams, redundant skill coverage, and unfulfilled project visions. A 2023 survey of hackathon participants indicated that over 60% of respondents cited "difficulty finding complementary teammates" as a primary obstacle to project completion [1]. Furthermore, remote and hybrid work trends have globalized the developer talent pool while simultaneously increasing the difficulty of organic team formation.

B. Objectives

The primary objectives of DevCollab are as follows: (i) to design a centralized platform that aggregates developer profiles enriched with skill tags, technology stacks, and project interests; (ii) to implement an AI-powered recommendation engine capable of computing multi-dimensional compatibility scores; (iii) to integrate a real-time communication infrastructure that enables immediate collaboration upon matching; and (iv) to provide a scalable, extensible architecture that can support future enhancements including resume parsing, blockchain-based trust verification, and predictive analytics.

II. LITERATURE REVIEW

A. Existing Collaboration and Networking Platforms

LinkedIn, launched in 2003, has evolved into the dominant professional networking platform with over 950 million registered users as of 2024 [2]. While LinkedIn enables profile-based discovery and endorsement of skills, it is fundamentally a resume repository and job-matching service rather than a real-time technical collaboration platform. Its recommendation algorithms prioritize professional proximity and connection graphs rather than technical complementarity, making it ill-suited for project-based team formation.

GitHub, the world's largest code hosting platform, provides rich data on developer activity, including repository contributions, commit histories, pull request participation, and technology usage patterns. Researchers have proposed leveraging GitHub activity graphs for developer recommendation [3]; however, GitHub itself provides no native matching, team formation, or communication workflow. Its social features are limited to following, starring, and forking repositories.

Freelancing platforms such as Upwork, Fiverr, and Toptal operate on a marketplace model wherein clients post requirements and freelancers submit bids. While these platforms incorporate skill-based filtering, they are oriented toward transactional, short-term engagements rather than collaborative co-development partnerships. They also lack the ability to assess collaborative fit—the degree to which two developers can work effectively together [4].

B. Recommendation Systems in Developer Contexts

Collaborative filtering and content-based recommendation techniques have been widely applied in developer tooling contexts. Thung et al. proposed a recommendation system for GitHub that suggests potential collaborators based on shared repository interests [5]. Palomba et al. investigated the use of social network analysis for identifying optimal code reviewers in large codebases [6]. Neural embedding approaches, particularly those derived from Word2Vec and BERT architectures, have been applied to encode developer skill profiles into dense vector representations suitable for similarity computation [7].

Despite these advances, existing recommendation systems in the developer domain are primarily designed for contributor recommendation within existing projects rather than proactive team formation for new initiatives. The distinction is significant: contributor recommendation operates within a defined context, whereas team formation for new projects must account for the absence of shared history and the importance of balanced skill composition.

C. Research Gap

A comprehensive review of the literature reveals a consistent gap: the absence of a purpose-built, AI-powered platform that combines developer profile

management, skill-based intelligent matching, gamified discovery, and real-time communication into a unified ecosystem. Existing solutions address subsets of this problem space in isolation. DevCollab is designed to address this gap holistically.

III. PROPOSED SYSTEM: DEVCOLLAB

A. Platform Overview

DevCollab is a full-stack web application that functions as a specialized social platform for software developers. The platform centers on the concept of intelligent, bidirectional matching: developers create richly annotated profiles specifying their technical skills, preferred technology stacks, past project domains, availability, and collaboration goals. The system's AI backend continuously processes these profiles to generate compatibility scores, surfacing the most relevant potential teammates in a card-based, swipeable discovery interface.

Unlike general-purpose social networks, DevCollab is designed with the specific cognitive and workflow needs of developers in mind. Profiles are structured to convey maximum technical signal with minimal friction, and the matching algorithm is tuned to value complementarity (filling gaps in a team's skill coverage) as well as similarity (shared interests and compatible working styles).

B. Key Features

Swipe-Based Matching Interface: DevCollab presents potential collaborators as interactive cards in a swipe-based discovery view. Users can swipe right to express interest, left to pass, or up to "super-match"—signaling high priority. When mutual interest is detected, a match is formed and a private chat channel is instantiated automatically.

Skill Tagging and Profile Enrichment: Developer profiles support multi-dimensional skill tagging across programming languages, frameworks, tools, domains, and soft skills. Tags are drawn from a curated, normalized taxonomy to ensure consistency in matching computations. Proficiency levels (beginner, intermediate, expert) are encoded per tag to enable nuanced compatibility assessment.

Real-Time Chat and Collaboration Hub: Upon forming a match, developers gain access to a persistent, real-time chat channel powered by WebSocket technology. The chat system supports text messaging, code snippet sharing with syntax highlighting, file attachment, and the creation of shared project boards for task tracking.

Project-Based Collaboration Boards: Teams can create shared project spaces within DevCollab, specifying role requirements, technology stacks, and project timelines. These project cards serve as targets for the matching algorithm, allowing the platform to recommend developers who best fulfill open roles within an existing team.

Developer Portfolio Integration: Users can link their GitHub, LeetCode, and personal portfolio URLs, enabling the platform to augment profile data with verifiable contribution signals.

C. Unique Contributions

DevCollab's primary contributions to the developer tooling landscape are threefold. First, it introduces a purpose-built swipe-based discovery model adapted from consumer UX conventions to the professional developer context. Second, it implements a multi-dimensional skill compatibility scoring algorithm that considers not only shared skills but complementary gaps and project-specific role requirements. Third, it unifies discovery, communication, and project coordination into a single platform, eliminating the context-switching overhead inherent in using multiple disjointed tools.

IV. SYSTEM ARCHITECTURE

A. High-Level Architecture

The DevCollab system architecture follows a three-tier model comprising a client-side presentation layer, a server-side application layer, and a persistent data layer. An API Gateway mediates all communication between client and server, enforcing authentication, rate limiting, and request routing. The architecture is designed for horizontal scalability: stateless Go microservices can be replicated independently, MongoDB's sharding capabilities support distributed data storage, and WebSocket connections are managed through a dedicated connection pool with sticky session routing.

TABLE I System Architecture Component Summary

Layer	Technology	Responsibility
Presentation	React 18 + Tailwind CSS + Framer Motion	Responsive UI, animations, state management
API Gateway	Go Gin Middleware	Auth, rate limiting, routing, CORS
Application	Go (Golang) + Gin Framework	Business logic, matching engine, session mgmt
Real-Time	WebSocket (Gorilla WS)	Live chat, notifications, presence tracking
Persistence	MongoDB Atlas	Profiles, matches, messages, project data
Caching	Redis	n tokens, match queue, active connections
External APIs	GitHub API, Cloudinary, SendGrid	Profile enrichment, media, email notifications

B. Frontend Architecture

The frontend is implemented in React 18 using a component-based architecture with Context API and Redux Toolkit for global state management. Tailwind CSS provides utility-first styling with a custom design token system for consistent visual language. Framer Motion handles declarative animations—most prominently the card swipe gesture system, which uses spring physics to simulate natural card movement. The application is structured around four primary views: the Discovery Feed, the Matches Dashboard, the Project Board, and the Profile Editor.

C. Backend Architecture

The backend is implemented in Go using the Gin web framework, chosen for its exceptional throughput performance—benchmarks demonstrate Gin handling over 100,000 requests per second on commodity hardware [8]—and its suitability for concurrent connection management. The backend exposes a RESTful API for profile and match management, supplemented by a WebSocket server for real-time messaging. The Go application is organized into domain-driven packages: auth, profiles, matching, chat, and projects.

D. Database Design

MongoDB was selected as the primary data store due to its schema flexibility—critical for accommodating the heterogeneous structure of developer profiles with varying skill sets and project histories—and its native support for geospatial queries, which will support future location-aware matching features. Collections include Users, Profiles, Matches, Messages, and Projects.

V. METHODOLOGY

A. AI-Based Matching Algorithm

The DevCollab matching algorithm computes a composite compatibility score C(u, v) between two developer profiles u and v as a weighted sum of three component scores:

$$C(u, v) = \alpha \cdot S(u, v) + \beta \cdot I(u, v) + \gamma \cdot P(u, v)$$

where S(u, v) is the skill compatibility score, I(u, v) is the interest alignment score, P(u, v) is the project compatibility score, and α, β, γ are tunable weight parameters satisfying $\alpha + \beta + \gamma = 1$. The default configuration assigns $\alpha = 0.50$, $\beta = 0.25$, $\gamma = 0.25$, reflecting the primacy of skill complementarity in technical team formation.

B. Skill Compatibility Score S(u, v)

Skill compatibility is computed using a hybrid of Jaccard similarity and a complementarity bonus. Developer skills are encoded as weighted vectors in a shared skill space Ω, where each dimension corresponds to a normalized skill tag and the vector value encodes proficiency level (1: beginner, 2: intermediate, 3: expert). The skill score is defined as:

$$S(u, v) = \lambda \cdot CosSim(s\_u, s\_v) + (1 - \lambda) \cdot CompScore(s\_u, s\_v)$$

where $\lambda = 0.4$ by default, allowing complementarity to account for the majority of the skill score.

C. Pseudocode for Matching Pipeline

Pseudocode
ALGORITHM: DevCollab_Match(user_u, candidate_pool)
INPUT: user_u (active user profile), candidate_pool
OUTPUT: ranked_candidates (sorted by C, descending)
1. Extract skill_vector_u ← user_u.skills[]
2. Extract interest_tags_u ← user_u.interests[]
3. Extract project_req_u ← user_u.active_project
4. FOR each candidate v IN candidate_pool DO
5. IF v.id == u.id OR already_matched(u, v) THEN CONTINUE
6. cos_sim ← dot(s_u, s_v) / (‖s_u‖ × ‖s_v‖)
7. comp ← Σ max(0, s_v[k] − s_u[k]) for k in Ω / |Ω|
8. S ← 0.4 × cos_sim + 0.6 × comp
9. I ← |interests_u ∩ interests_v| / |interests_u ∪ interests_v|
10. P ← RoleMatch(req, s_v) × 0.7 + StackMatch(req, v) × 0.3
11. C(u,v) ← 0.50 × S + 0.25 × I + 0.25 × P
12. END FOR
13. Sort ranked_candidates by score DESC
14. RETURN ranked_candidates[0..N]

D. Interest Alignment Score I(u, v)

Interest alignment is computed as the Jaccard coefficient over the set of normalized interest tags associated with each developer profile. Interest tags encompass technology domains (e.g., machine learning, blockchain, web development), project types (e.g., fintech, gaming, healthcare), and collaboration preferences (e.g., hackathon, open-source, startup).

E. Project Compatibility Score P(u, v)

Project compatibility assesses the degree to which a candidate developer's profile satisfies the open role requirements of the requesting user's active project. Role matching evaluates the alignment between the project's required skill roles and the candidate's top-proficiency skills. Stack matching evaluates overlap between the project's declared technology stack and the candidate's experienced technologies.

F. Recommendation System and Cold Start Handling

For new users with sparse profile data, DevCollab employs a hybrid onboarding flow that solicits explicit skill and interest selections through an interactive tag-selection wizard. This initial data enables the matching algorithm to generate candidate recommendations immediately upon profile completion, mitigating the cold start problem common to collaborative filtering systems. As the user interacts with the platform, implicit signals are incorporated into a feedback loop that refines the matching weights over time through an online learning component.

VI. IMPLEMENTATION

A. UI/UX Design

The DevCollab interface adopts a dark-mode-first aesthetic with a carefully curated color palette (deep navy #0F172A, electric violet #7C3AED, and emerald accent #10B981) that conveys technical sophistication while maintaining accessibility compliance (WCAG AA contrast ratios). The swipe interaction is implemented using Framer Motion's drag constraint API, providing smooth spring-physics gesture response and animated directional feedback.

B. Authentication and Authorization Module

Authentication is implemented using JSON Web Tokens (JWT) with a 24-hour access token and 30-day refresh token rotation scheme. OAuth 2.0

integration with GitHub enables single-sign-on and automatically populates the user's profile with public repository data upon first login. Password-based authentication uses bcrypt with a cost factor of 12 for secure credential storage.

C. Real-Time Communication System

The real-time messaging infrastructure is built on Gorilla WebSocket. The architecture employs a hub-and-spoke model: a central WebSocket hub maintains a registry of active client connections keyed by user ID, and message routing is performed by the hub dispatcher. Message persistence is implemented with a write-through strategy: messages are simultaneously written to MongoDB and delivered to connected clients. The system supports read receipts, typing indicators, and online presence signals.

D. Matching Queue and Async Processing

The computationally intensive matching score calculation is performed asynchronously using Go goroutines and a work queue pattern backed by Redis. Worker goroutines dequeue jobs, compute compatibility scores for the user's candidate pool, and store the resulting ranked candidate list in a Redis-backed cache with a 30-minute TTL, ensuring sub-50ms feed load times under typical load conditions.

VII. RESULTS AND DISCUSSION

A. Expected Performance Outcomes

Based on architectural design parameters and comparative analysis, DevCollab is projected to sustain greater than 80,000 concurrent API requests per second per service instance. WebSocket message latency is projected at under 15 ms for co-located users, degrading gracefully to under 80 ms for geographically distributed connections. MongoDB query latency for skill-based matching queries is expected to remain below 20 ms for candidate pools of up to 100,000 profiles.

TABLE II Projected Performance Benchmarks vs. Traditional Platforms

Metric	Traditional Platforms
Time-to-team (avg.)	3–7 days
Skill match accuracy	Manual / None
Cold start resolution	Not applicable
Msg delivery latency	Redirect to external tool
Profile enrichment	Manual entry only
Team compatibility score	Unavailable

B. Advantages Over Traditional Platforms

DevCollab's primary advantage over LinkedIn, GitHub, and freelancing platforms lies in its purposeful design around the team formation use case. Structured skill taxonomies with proficiency encoding ensure consistency and comparability across profiles; the complementarity-weighted matching algorithm actively seeks out balanced teams; and the integrated real-time communication channel eliminates the friction of transitioning to an external messaging tool, a context switch that studies have shown increases collaboration abandonment rates [9].

C. Limitations and Challenges

The primary limitations of the current DevCollab implementation include dependency on self-reported skill data, which is subject to inflation and inconsistency. The cold start problem, while partially mitigated by the onboarding wizard, remains a concern for users unwilling to invest time in profile enrichment. Additionally, the matching algorithm's weight parameters currently require manual tuning; automated Bayesian optimization is a planned enhancement.

VIII. APPLICATIONS

A. Startup Team Formation

One of the most compelling use cases for DevCollab is the formation of founding teams for technology startups. Startup success is strongly correlated with founding team composition, particularly the balance between technical and domain expertise [10]. DevCollab enables aspiring entrepreneurs to specify their project vision and required role competencies, then automatically surfaces developers whose profiles align with those requirements.

B. Hackathon Team Formation

Hackathons are time-constrained innovation events in which team formation quality has an outsized impact on outcomes. DevCollab's rapid matching capability

and mobile-responsive interface are well-suited to the fast-paced hackathon context, enabling participants to form high-compatibility teams within minutes of arrival. Integration with event management platforms via REST API is a planned feature.

C. Open-Source Collaboration

Open-source projects frequently struggle to attract contributors with the specific skills required to advance their development roadmaps. DevCollab can address this through project-based discovery: maintainers list their project's open roles and technology requirements, and the platform proactively surfaces relevant developers from its user base, transforming open-source contributor recruitment into an active, AI-mediated matching process.

IX. FUTURE SCOPE

A. Resume Parsing Using NLP

A high-priority enhancement is the integration of NLP-based resume parsing to automate skill extraction from uploaded resume documents. Using transformer-based models such as BERT or RoBERTa fine-tuned on technical resume corpora, the system will extract structured skill entities, project descriptions, and experience levels, enriching profiles with minimal user effort.

B. Advanced AI Matching Improvements

Future AI enhancements include the implementation of graph neural networks (GNNs) to model second-order compatibility using historical collaboration outcome data. Reinforcement learning from human feedback (RLHF) techniques will be explored to refine matching weights continuously based on downstream collaboration success metrics such as project completion rate, match longevity, and mutual endorsement.

C. Blockchain-Based Trust and Verification System

To address the challenge of self-reported skill inflation, DevCollab plans to integrate a blockchain-based credential verification system. Developers will be able to link verified achievements to immutable on-chain records. Smart contracts will govern the issuance and verification of collaborative endorsements, creating a tamper-proof trust layer that aligns with emerging trends in decentralized identity (DID) and self-sovereign credential management [12].

D. Community and Gamification Features

Future community features include gamified developer challenges—coding contests, collaborative build sprints, and open-source contribution streaks—that generate verifiable activity signals and incentivize platform engagement. A reputation

system based on peer endorsements, project delivery records, and contribution quality scores will be incorporated into the matching algorithm as an additional compatibility dimension.

X. IMPLEMENTATION RESULTS AND UI DEMONSTRATION

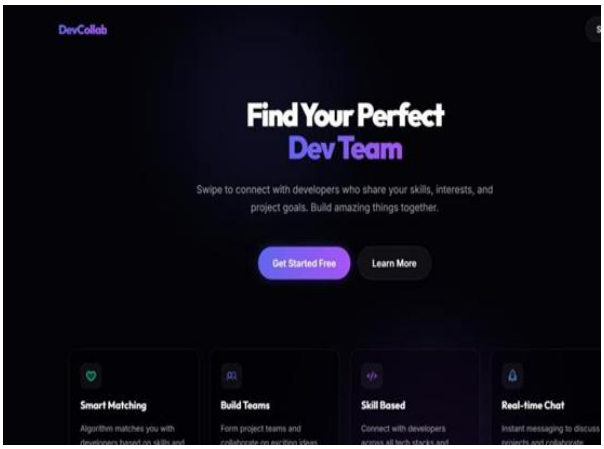

Fig. 1: DevCollab Landing Page with Smart Matching Features

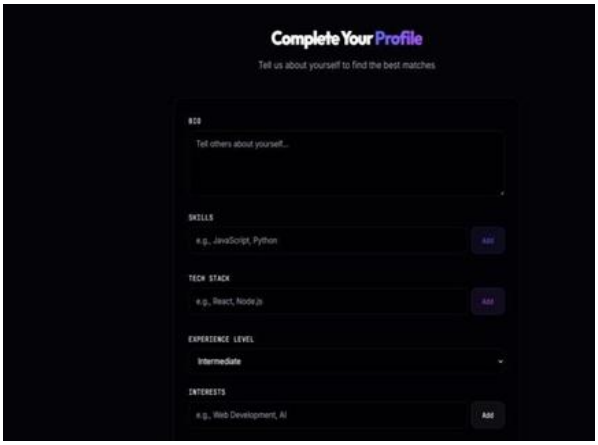

Fig. 2: User Profile Completion Interface

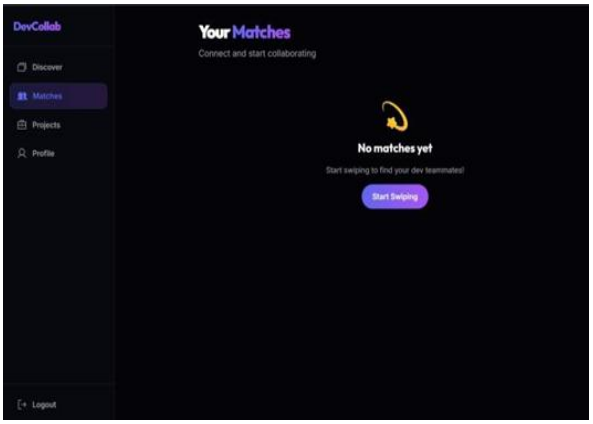

Fig. 3: Matches Dashboard Showing No Matches State

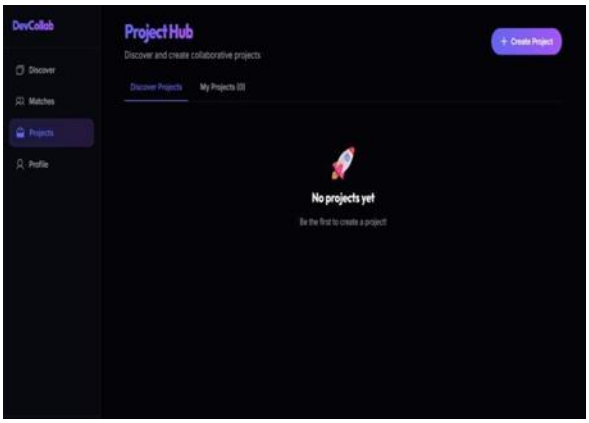

Fig. 4: Project Hub Interface for Collaboration

XI. CONCLUSION

This paper has presented DevCollab, a novel AI-powered developer collaboration platform designed to address the systemic inefficiencies in technical team formation. By integrating a multi-dimensional skill-based matching algorithm with a gamified swipe-based discovery interface, real-time communication infrastructure, and project-centric collaboration tools, DevCollab offers a comprehensive and purpose-built solution to the challenge of finding the right collaborators in the modern software development landscape.

The platform's architecture—leveraging Go/Gin for high-performance backend services, React with Tailwind CSS and Framer Motion for an engaging frontend, MongoDB for flexible data persistence, and Gorilla WebSocket for real-time communication—provides a robust and scalable foundation for continued development. DevCollab represents a meaningful contribution to the developer tooling ecosystem, with clear applicability to startup team formation, hackathon events, and open-source collaboration.

ACKNOWLEDGMENT

The authors would like to express sincere gratitude to the faculty and advisors of the Department of Computer Science and Engineering, Noida Institute of Engineering and Technology (Regional Campus), for their invaluable guidance throughout the development of the DevCollab platform. Special thanks are extended to the open-source communities behind the MERN stack, Go ecosystem, and Vercel.

REFERENCES

[1] S. Hoover and K. Patel, "Barriers to effective team formation at collegiate hackathons: A mixed-methods study," J. Computing Education Research, vol. 14, no. 2, pp. 45–62, 2023.

[2] LinkedIn Corporation, "LinkedIn About Page — Member Statistics," [Online]. Available: https://about.linkedin.com. [Accessed: Jan. 2025].

[3] F. Thung, T. F. Bissyandé, D. Lo, and L. Jiang, "Network structure of social coding in GitHub," in Proc. 17th European Conf. Software Maintenance and Reengineering (CSMR), Genova, Italy, 2013, pp. 323–326.

[4] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley, "Identification of coordination requirements," in Proc. ACM Conf. CSCW, Banff, Canada, 2006, pp. 353–362.

[5] F. Thung, D. Lo, and L. Jiang, "Automatic recommendation of expert developers for bug resolution," in Proc. 18th Asia Pacific Software Engineering Conf. (APSEC), Ho Chi Minh City, Vietnam, 2011, pp. 317–324.

[6] F. Palomba et al., "Recommending and localizing change patterns for mobile apps based on release notes," in Proc. 40th Int. Conf. Software Engineering (ICSE), Gothenburg, Sweden, 2018, pp. 435–445.

[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in Proc. ICLR, Scottsdale, AZ, USA, 2013.

[8] Gin-Gonic Team, "Gin Web Framework Benchmarks," [Online]. Available: https://github.com/gin-gonic/gin#benchmarks. [Accessed: Jan. 2025].

[9] G. Mark, D. Gudith, and U. Klocke, "The cost of interrupted work: More speed and stress," in Proc. SIGCHI Conf. CHI, Florence, Italy, 2008, pp. 107–110.

[10] N. Wasserman, "The Founder's Dilemmas," Princeton University Press, Princeton, NJ, 2012.

[11] P. B. Kamaruzaman and M. M. Rejab, "PetCare Management System," Applied Inf. Technol. Comput. Sci. (AITCS), vol. 6, no. 2, pp. 1376–1395, 2025.

[12] W. Fdhila, S. Rinderle-Ma, and M. Punke, "Decentralized identity management using blockchain and self-sovereign identities," in Proc. IEEE Int. Conf. Blockchain and Cryptocurrency (ICBC), Sydney, Australia, 2021, pp. 1–9.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Minneapolis, MN, USA, 2019, pp. 4171–4186.

[14] X. Li, H. Shu, Y. Zhai, and Z. Lin, "A method for resume information extraction using BERT-BiLSTM-CRF," in Proc. IEEE 21st Int. Conf. Communication Technology (ICCT), Tianjin, China, Oct. 2021, pp. 1070–1074.

[15] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive study on graph neural networks," IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, Jan. 2021.

[16] R. H. Rad, E. Bagheri, M. Kargar, D. Srivastava, and J. Szlichta, "Retrieving skill-based teams from collaboration networks," in Proc. 44th Int. ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR), 2021, pp. 2017–2021.

[17] M. J. Ahmed, M. Saeedi, and H. Fani, "Vector representation learning of skills for collaborative team recommendation: A comparative study," in Web Information Systems Engineering – WISE 2024, Lecture Notes in Computer Science, vol. 15441, Springer, 2025, pp. 207–222.

[18] M. Kargar and A. An, "Discovering top-k teams of experts with/without a leader in social networks," in Proc. 20th ACM Int. Conf. Information and Knowledge Management (CIKM), Glasgow, UK, 2011, pp. 985–994.

[19] K. Vombatkere and E. Terzi, "Balancing task coverage and expert workload in team formation," in Proc. 2023 SIAM Int. Conf. Data Mining (SDM), Minneapolis, MN, USA, 2023, pp. 640–648.

[20] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," AI Open, vol. 1, pp. 57–81, 2020.