ProjectBuddy: Your Hackathon Teammate Matchmaker

Manish M Banait, Gaurav V Patil, Om S Patil, Yash A Shete
School of Computer Science and Engineering, Sandip University, Trimbak Road, Nashik, Maharashtra –422213, India
Email: banaitmanish2509@gmail.com / Corresponding Author: Manish M Banait
Project Guide: Dr. P. R. Patil, Professor, SOCSE, Sandip University

Abstract: The growing culture of student hackathons has created a pressing need for structured mechanisms that support team assembly and mentorship access. In practice, most participants form teams through informal channels such as personal messaging groups, physical notice boards, or word-of-mouth referrals, resulting in unbalanced compositions and significant time loss. This paper introduces ProjectBuddy, a purpose-built web application that enables engineering students to publish structured technical profiles, discover teammates through skill-based and domain-based search filters, and receive ranked compatibility recommendations generated by a weighted cosine-similarity algorithm. The platform additionally provides a mentor discovery module, a post-collaboration review system, and a community knowledge-sharing section. The technology stack comprises HTML5, CSS3, Bootstrap 5, JavaScript, a Node.js/Express backend, a relational SQL database, and Microsoft Azure cloud hosting. A comparative user study involving sixty participants across two institutions demonstrated that platform-assisted team formation reduced average assembly time by 74 percent, improved team skill diversity scores by 52.9 percent, and raised post-event team satisfaction from 2.9 to 4.0 out of 5.0. System Usability Scale evaluation returned a score of 79.4, placing the platform in the 'Good' usability category.

Keywords — hackathon team formation, skill-based matching, cosine similarity, web platform, mentor discovery, collaborative learning, Microsoft Azure, student networking.

I. INTRODUCTION

Over the past decade, student hackathons have transformed from informal coding marathons into structured learning events that sit at the intersection of academic education and industry practice. Universities across India and internationally now recognise hackathon participation as a measurable indicator of problem-solving aptitude, collaborative competence, and technical breadth. The scale of participation has expanded considerably; reports from Major League Hacking indicate that more than one hundred thousand students registered for sanctioned hackathon events in 2022 alone, a figure that continues to grow as institutions embed such events into their engineering curricula.

Within this expanding landscape, however, a foundational challenge persists: the process through which student teams are assembled prior to competition remains largely unstructured and socially biased. Participants typically rely on existing friendship groups, department-level messaging threads, or last-minute announcements posted on physical notice boards. These informal mechanisms share a critical limitation: they privilege individuals who are already embedded in strong peer networks while systematically excluding those who are new to the institution, transferring between departments, or simply less socially visible. The result is a pattern of team formation that is both inequitable and operationally inefficient.

The consequences of poor team composition extend beyond the competition itself. A team assembled without regard for skill complementarity is likely to lack coverage across essential technical domains — a web-focused team may have no member with data analytics experience, or a hardware-oriented group may be unable to develop a functional user interface. Such structural gaps reduce submission quality and diminish the educational value of the event for all members involved. Moreover, beginner-level participants who are unable to join any team disengage from collaborative learning culture at precisely the stage when engagement would be most formative.

A parallel deficiency exists in the domain of mentorship. Hackathon participants, particularly those in their first or second year of study, benefit substantially from access to experienced mentors who can guide project scoping, architectural decisions, and presentation strategy. In the absence of a structured discovery mechanism, such mentorship is distributed through informal alumni networks and departmental contacts that are accessible only to students at well-resourced institutions. This creates a significant equity gap in the quality of support available to participants from smaller or less-connected colleges.

ProjectBuddy is a web-based platform developed to systematically address these challenges. It provides a structured environment in which students can publish searchable technical profiles, use domain-based and skill-based filters to identify potential teammates, and receive algorithmically ranked compatibility recommendations. A dedicated mentor module enables practitioners to register their availability and domain expertise, making mentorship discoverable by any registered participant. The platform was designed, developed, and evaluated as a final-year engineering project at Sandip University, and its effectiveness was assessed through a controlled user study described in detail in subsequent sections.

II. LITERATURE REVIEW

Research into team formation, collaborative matching, and recommender systems provides a rich theoretical foundation upon which ProjectBuddy is constructed. This section surveys the most relevant prior contributions and identifies the gaps that motivated the present work.

A. Automated Group Formation in Education

Ounnas, Davis, and Millard [1] proposed a framework for semantic group formation in educational settings, representing student competencies through ontological structures and applying constraint-based assignment to generate groups with complementary coverage. Their system demonstrated that structured competency modelling produces more balanced groups than random or self-directed assignment. However, the framework was designed for instructor-administered classroom contexts with predefined competency vocabularies, making direct application to the self-directed, event-specific context of hackathons impractical. Similarly, Carro, Pulido, and Rodríguez [2] examined constraint satisfaction approaches to adaptive group formation, focusing primarily on academic achievement metrics rather than technical skill compatibility.

B. Content-Based Recommender Systems

The theoretical basis for profile-driven teammate recommendation draws heavily from content-based filtering, surveyed comprehensively by Lops, de Gemmis, and Semeraro [3]. Their analysis establishes that vector-space representations of user attributes, combined with cosine similarity computation, provide an effective and interpretable baseline for profile-matching tasks. The approach scales well to sparse, high-dimensional attribute vectors of the kind produced by multi-hot encoding of skill and domain tags, and avoids the popularity bias inherent in collaborative filtering. The weighted extension of cosine similarity — in which different attribute groups contribute differentially to the final score — provides additional flexibility appropriate to contexts where some features, such as domain alignment, are empirically more predictive of collaboration quality than others.

C. Hackathon Dynamics and Team Quality

Briscoe and Mulligan [4] conducted an ethnographic examination of digital innovation hackathons and identified team composition at the outset of the event as one of the strongest determinants of both submission quality and participant satisfaction. Their fieldwork revealed that the informal, socially-mediated nature of team formation represents a structural weakness in most hackathon formats — one that disproportionately disadvantages participants who lack pre-existing peer connections. Despite this documented finding, purpose-built digital platforms that address team formation at scale remain scarce. Existing platforms such as Devfolio and Devpost provide event registration and project submission infrastructure but offer no structured teammate-discovery or algorithmic matching functionality.

D. Reputation and Engagement in Collaborative Platforms

Hamari, Koivisto, and Sarsa [5] studied the role of gamification and social proof in online collaborative environments, finding that visible reputation signals — such as peer ratings and contribution histories — substantially improve both user engagement and inter-user trust. This finding informed the design of ProjectBuddy's review and rating module, through which participants can submit structured post-collaboration feedback that contributes to a visible reputation score on each user's profile. The incorporation of GitHub activity data as a proxy for technical productivity is further motivated by research demonstrating that version-control contribution metrics carry meaningful information about software development engagement levels.

III. PROBLEM STATEMENT AND OBJECTIVES

A. Problem Statement

A structured requirement elicitation exercise involving forty-five engineering students across two Maharashtra institutions identified four principal failure modes in existing hackathon team formation practice. First, technical skills remain invisible: no searchable, standardised record of student competencies exists that is accessible to peers outside an individual's immediate social circle, causing participants with valuable but niche skills to be overlooked. Second, team access is network-dependent: students without established peer connections are effectively excluded from competitive team pools, creating inequitable participation outcomes. Third, mentorship is inaccessible: the absence of a structured mentor discovery channel means that novice participants either proceed without guidance or obtain it through informal contacts unavailable to all. Fourth, the formation process is time-intensive: without digital tools, team assembly consumes a disproportionate share of the limited time available before competition begins.

B. Objectives

Drawing directly from these identified failure modes, the objectives of the ProjectBuddy system are as follows: (i) to provide a web platform enabling structured technical profile creation covering skills, domain preferences, GitHub activity, and prior project experience; (ii) to implement multi-criteria search and filtering that surfaces potential teammates matching specified domain and skill requirements; (iii) to develop a weighted cosine-similarity algorithm that generates ranked teammate recommendations; (iv) to build a mentor discovery module supporting domain-based mentor search and direct contact initiation; (v) to integrate a review and rating system that builds verifiable reputation over time; and (vi) to deploy the platform on scalable cloud infrastructure that remains reliably available during high-demand hackathon registration periods.

IV. SYSTEM ARCHITECTURE

ProjectBuddy adopts a three-tier client-server architecture comprising a Presentation Layer, an Application Layer, and a Data Layer, deployed on Microsoft Azure cloud infrastructure. This organisation ensures clean separation of concerns between user interface rendering, business logic execution, and data persistence.

A. Presentation Layer

The client-facing interface is rendered in the user's web browser using HTML5 semantic markup, CSS3 custom properties, Bootstrap 5.3 responsive grid components, and vanilla JavaScript. A custom tag-input component allows users to select skill and domain tags interactively during profile creation. Search results and match recommendations are loaded asynchronously via the Fetch API, eliminating full-page reloads and providing a fluid browsing experience without the overhead of a frontend JavaScript framework.

B. Application Layer

The backend server is implemented using Node.js 18 LTS and the Express.js framework. It exposes RESTful API endpoints organised into four route modules: /auth for registration and session management, /profile for profile creation and retrieval, /match for search and recommendation, and /mentor for mentor discovery and contact. All database interactions use parameterised queries via the node-mssql driver, providing systematic protection against SQL injection. Input validation on all mutation endpoints uses the express-validator middleware with structured RFC 7807 error responses.

C. Data Layer

Persistent data is maintained in an Azure SQL Database instance. The schema comprises seven normalised tables: Users, Skills, UserSkills, Domains, UserDomains, Mentors, and Reviews. Referential integrity is enforced through foreign key constraints with ON DELETE CASCADE behaviour. Composite indexes on the UserSkills(user_id, skill_id) and UserDomains(user_id, domain_id) junction tables accelerate the multi-join queries executed by the matching engine during recommendation generation.

D. Architecture Diagram Description

Figure 1 depicts the system flow. Client browsers communicate over HTTPS with the Azure App Service instance hosting the Node.js Express server. The server dispatches requests to five controller modules (Auth, Profile, Match, Mentor, Review), which execute SQL queries against the Azure SQL Database and optionally invoke the GitHub REST API to retrieve activity scores. Static assets are served from Azure Blob Storage through a CDN layer. All application secrets, including database credentials and session keys, are stored in Azure Key Vault and injected as environment variables at runtime.

Figure 1: ProjectBuddy System Architecture (Conceptual)

V. METHODOLOGY

Development followed an iterative hybrid methodology that combined upfront requirements documentation with agile sprint cycles for implementation and testing. Five sequential phases were completed: requirement analysis, system design, development, testing, and cloud deployment.

A. Requirement Analysis

A structured survey administered to forty-five students across two engineering colleges in Maharashtra yielded the following statistically notable findings: 82 percent reported difficulty locating teammates with the required technical profile; 67 percent had participated in at least one hackathon with a suboptimally composed team; 91 percent expressed willingness to use a structured digital platform for teammate discovery; and 78 percent identified mentor access as a priority need. These results were used to rank features by priority and construct the system's functional requirement specification.

B. System Design

The design phase produced three artefacts. The Entity-Relationship diagram identified six primary entities — User, Skill, Domain, Mentor, Review, and BlogPost — and defined their relationships. The database schema normalised these entities into SQL tables with appropriate constraints. User interface wireframes produced in Figma defined interaction patterns for all primary screens, including registration, profile building, search, match recommendations, and mentor discovery, with layouts conforming to Bootstrap's twelve-column responsive grid.

C. Development and Testing

Frontend development proceeded screen by screen against the Figma wireframes. Backend service functions were covered by unit tests written in Mocha and Chai, achieving 78 percent code coverage. Integration tests validated each primary user journey end-to-end. User Acceptance Testing with fifteen student volunteers using the System Usability Scale produced a mean score of 79.4, confirming 'Good' usability. The GitHub Actions CI/CD pipeline automated test execution on feature branches and triggered zero-downtime blue-green deployments to Azure App Service on main-branch merge.

VI. MATCHING ALGORITHM

A. Problem Formulation

Teammate matching is formalised as a ranked retrieval task. Given a query profile Q and a database of candidate profiles C_1, C_2, …, C_n, the objective is to produce an ordered list of candidates sorted by decreasing compatibility with Q. Compatibility is operationalised as a weighted aggregate of pairwise attribute similarities across five feature groups: technical skills, domain interests, experience level, GitHub activity, and general interests.

B. Feature Vector Construction

Each profile is encoded as a composite feature vector V by concatenating five sub-vectors. V_s is a binary multi-hot vector over all registered skill tags (e.g., Python, React, TensorFlow, Figma). V^d is a binary multi-hot vector over eight domain categories (AI/ML, Web Development, UI/UX, Cybersecurity, App Development, Data Science, IoT, Blockchain). V_e is an ordinal scalar normalised to [0,1]: Beginner−0.25, Intermediate−0.5, Advanced−0.75, Expert−1.0. V^g is a scalar derived from GitHub public repository count and recent commit frequency via the GitHub REST API, normalised to [0,1]. V^l is a binary

multi-hot vector over general interest tags such as open source, competitive programming, and research.

C. Similarity Formula

The weighted similarity score S(Q, C) is defined as:

$$S(Q, C) = \Sigma_i\, w_i \cdot \cos(V_i^L, V_i^c)$$
$$= \Sigma_i\, w_i \cdot (V_i^L \cdot V_i^c) / (\|V_i^L\| \cdot \|V_i^c\|) \quad \ldots(1)$$

In equation (1), i indexes the five attribute groups {s, d, e, g, i}; w_i is the weight assigned to group i; and the cosine of the angle between the query and candidate sub-vectors for group i captures directional similarity independent of vector magnitude. Since Σw_i 1.0, the composite score S(Q,C) lies in [0, 1], where 1.0 represents complete attribute alignment.

D. Weight Assignment

Attribute group weights were determined from the normalised mean importance ratings obtained during requirement analysis. Students ranked five teammate attributes on a five-point Likert scale. The resulting normalised weights are: domain alignment w^d 0.35, skill overlap w_s 0.30, interests compatibility w^l 0.15, experience proximity w_e 0.12, and GitHub activity w^g 0.08. Domain alignment received the highest weight, consistent with the survey finding that shared technical domain is the single most important criterion in teammate selection.

E. Output and Example

For each query user, the algorithm computes $S(Q, C_k)$ for all registered candidates and returns the top ten results with $S \geq 0.20$ sorted in descending order. Results are presented as profile cards showing the candidate's name, top three skills, domain tags, and numeric match percentage. As a practical illustration: a student with Python and TensorFlow skills seeking an AI/ML hackathon teammate would receive recommendations for candidates with complementary skills such as React (frontend), Data Visualisation, or Cloud DevOps, rather than candidates who also list TensorFlow and thereby duplicate an already-covered competency.

VII. IMPLEMENTATION DETAILS

The ProjectBuddy prototype is implemented using the following technology stack and module structure.

A. Frontend

The interface layer uses HTML5, CSS3 with custom properties for theming, Bootstrap 5.3 for responsive layout, and ES6+ JavaScript for interactivity. Skill and domain tag selection uses a custom badge-input component. All API calls use the browser-native Fetch API with async/await syntax, providing non-blocking data loading for search and match screens without requiring a frontend framework.

B. Backend

The server is organised into route, controller, and service layers following the Repository pattern. The four route modules (/auth, /profile, /match, /mentor) each delegate to a corresponding controller, which calls service functions that encapsulate all database interaction. Session state is maintained server-side using express-session with the session store backed by Azure SQL, enabling horizontal scaling without sticky sessions. All POST and PUT endpoints validate payloads using express-validator before any business logic is executed.

C. Database and Cloud

The Azure SQL Database instance uses the General Purpose service tier with two virtual cores. Seven normalised tables store all application data. The App Service Plan (B1 tier) hosts the Node.js process and is configured to auto-scale to three instances when sustained CPU utilisation exceeds 70 percent. GitHub Actions pipelines run the Mocha test suite on every feature-branch push and deploy to a staging slot on the App Service on every main-branch merge, with slot swap completing the blue-green deployment.

D. Key Modules

Six functional modules were implemented: (1) User Management — registration, login, and role-based access control; (2) Profile Builder — multi-step form with skill-tag selection and GitHub URL integration; (3) Search and Filter — faceted filtering by domain, skills, year, and institution with dynamic SQL generation; (4) Matching Engine — weighted cosine-similarity computation returning top-ten ranked recommendations; (5) Mentor Discovery — domain-filtered mentor registry with in-app contact initiation; and (6) Review System — structured post-collaboration feedback collection with aggregate rating display.

VIII. RESULTS AND DISCUSSION

A controlled user study evaluated ProjectBuddy against conventional informal team formation using sixty student participants from two Maharashtra engineering colleges. Group A (n = 30) used the platform; Group B (n = 30) used informal methods. Both groups tackled identical problem statements judged by a common evaluation panel. All sixty participants completed post-study questionnaires measuring efficiency, team quality, and platform usability.

TABLE I

Performance Metrics — User Study Results

Metric	Grp A	Grp B	Change
Team formation time	11.2 min	43.7 min	−74%
Skill diversity (/10)	7.8	5.1	+52.9%
Satisfaction (/5.0)	4.2	N/A	—
SUS score (/100)	79.4	N/A	—
Profile completion	87%	N/A	—
Mentor connections	22/30	4/30	+450%
Top match score	0.71	N/A	—
Post-event sat. (/5)	4.0	2.9	+37.9%

The most pronounced finding concerns team assembly time. Group A participants completed team formation in an average of 11.2 minutes, compared to 43.7 minutes for

Group B, a reduction of 74 percent. This gain arises from eliminating the iterative, unstructured outreach that characterises informal formation — participants using ProjectBuddy could identify and contact compatible candidates through a single filtered search session rather than through sequential individual inquiries across multiple messaging channels.

Team skill diversity, measured by counting distinct technical skill categories represented within each assembled team and normalising to a ten-point scale, was meaningfully higher for Group A (7.8 vs. 5.1). Post-study qualitative feedback corroborated this finding: multiple Group A participants specifically noted that they had connected with individuals whose complementary expertise — such as embedded hardware integration or UI motion design — they would not have sought without algorithmic prompting.

The 450 percent increase in successful mentor connections (22 of 30 Group A participants vs. 4 of 30 in Group B) demonstrates that the mentor discovery module delivers substantial value beyond team formation. Post-event team satisfaction scores (4.0 vs. 2.9 out of 5.0) confirm that teams assembled through ProjectBuddy experienced qualitatively better collaborative outcomes, consistent with the literature finding that structured formation predicts team satisfaction [4]. The SUS score of 79.4 indicates that the platform is usable by its target audience without specialised training.

IX. SOCIAL IMPACT

The societal implications of ProjectBuddy reach beyond operational efficiency. Conventional hackathon team formation functions as a socially gated process in which access to strong peer networks — a form of accumulated social capital — determines access to capable teammates. Students at well-connected institutions or those in their senior years hold structural advantages over first-year students, transfer students, and those from smaller affiliated colleges. ProjectBuddy reconfigures this access structure by substituting skills and domain expertise as the primary discovery criteria, enabling any registered participant to be found and contacted by peers whose needs align with their profile.

The mentor discovery module carries particular significance in the context of Indian engineering education, where access to industry mentorship correlates strongly with institutional prestige. Students at elite national institutions benefit from extensive informal alumni and industry networks, while those at smaller state-affiliated colleges often have no equivalent access. By creating a discoverable, domain-indexed register of mentors willing to engage with any registered student, ProjectBuddy functions as a digital equaliser that redistributes mentorship access more equitably across the institutional hierarchy.

At a broader level, improving team formation quality has downstream effects on the innovation outcomes that hackathons are intended to generate. Teams that are well-composed and well-mentored produce higher-quality solutions, submit more complete projects, and are more likely to continue developing their ideas beyond the event itself. By improving the structural conditions under which student innovation begins, ProjectBuddy contributes indirectly to the robustness of the student-driven technology ecosystem.

X. CONCLUSION AND FUTURE WORK

A. Conclusion

This paper presented ProjectBuddy, a web platform for hackathon team formation and mentor discovery that addresses the documented limitations of informal team assembly through structured profiles, multi-criteria search, and weighted cosine-similarity matching. Implemented on a Node.js/Express backend with an Azure SQL relational database and deployed on Microsoft Azure, the platform was evaluated in a controlled study of sixty participants. Results confirmed a 74 percent reduction in team formation time, a 52.9 percent improvement in team skill diversity, and a 37.9 percent improvement in post-event team satisfaction. A SUS score of 79.4 and a mean satisfaction rating of 4.2 out of 5.0 validate that the platform is both effective and usable by its target audience.

B. Future Work

Four priority directions for future development are identified. First, integrating verified external skill signals — including parsed GitHub contribution graphs, competitive programming rankings from platforms such as Codeforces or LeetCode, and digitally-verified certification records — would improve the objectivity and reliability of compatibility scores beyond self-reported data. Second, introducing a feedback loop in which implicit signals from successful team formations are used to iteratively recalibrate attribute weights would move the system toward adaptive personalisation. Third, extending the platform with live-event features such as real-time team formation boards, in-app collaborative workspaces, and milestone tracking would increase utility across the full hackathon lifecycle rather than only the pre-event preparation phase. Fourth, developing institutional administrator accounts that allow event organizers to define custom domain taxonomies, curate participant pools, and manage mentor assignments would transform ProjectBuddy from a student-facing utility into a comprehensive institutional hackathon management solution suitable for adoption across multiple colleges simultaneously.

XI. COMPARATIVE ANALYSIS WITH EXISTING PLATFORMS

To contextualise the contribution of ProjectBuddy, it is useful to examine how the platform compares with tools that students presently use as informal substitutes for structured team formation. Three categories of substitute tools were commonly cited by survey respondents: general-purpose professional networking services,

hackathon event management portals, and informal group messaging applications.

A. General-Purpose Networking Platforms

Professional networking services such as LinkedIn offer rich profile infrastructure that supports skill listing, endorsement, and portfolio attachment. However, these platforms are architected for long-horizon career networking rather than event-specific, time-constrained team assembly. Their search functionality is not optimised for multi-criteria technical skill matching, and they provide no concept of a compatibility score or ranked recommendation relative to a specific hackathon domain. Additionally, the professional context of such services creates a social barrier for undergraduate students who do not yet have substantial work histories, making profile creation and peer outreach feel incongruent with the casual collaborative intent of hackathon participation.

B. Hackathon Event Portals

Platforms such as Devfolio and Devpost serve as the primary digital infrastructure for hackathon registration and project submission in the Indian student community. They provide event discovery, team registration, and submission management at scale. However, neither platform offers a structured teammate discovery mechanism. Participants may indicate that they are seeking a team, but the subsequent matching process is entirely manual. There is no skill-based search interface, no domain filter, and no algorithmic recommendation engine. The absence of these features means that the fundamental problem of unstructured, socially-mediated team formation persists even when students are actively using these portals.

C. Feature Comparison

Table II presents a structured comparison of ProjectBuddy against LinkedIn, Devfolio, and WhatsApp-based informal coordination across five capability dimensions relevant to hackathon team formation.

TABLE II

Platform Feature Comparison

Feature	LinkedIn	Devfolio	WhatsApp	ProjectBuddy
Skill-based search	Partial	No	No	Yes
Domain filtering	No	No	No	Yes
Match score ranking	No	No	No	Yes
Mentor discovery	Partial	No	No	Yes
Peer review system	No	No	No	Yes

As Table II illustrates, ProjectBuddy is the only platform in this comparison that provides all five capabilities within a single integrated environment. LinkedIn offers partial skill visibility through its endorsement mechanism but lacks domain filtering, algorithmic compatibility ranking, and a project-oriented peer review system. Devfolio and informal messaging tools such as WhatsApp provide none of these capabilities. This comparison confirms that ProjectBuddy addresses a genuine and presently unmet functional gap in the hackathon participation ecosystem, rather than duplicating existing tooling.

XII. LIMITATIONS AND THREATS TO VALIDITY

A. Self-Reported Profile Data

The matching algorithm operates exclusively on self-reported skill and domain information. Participants may overstate competency levels, list skills in which they have only incidental exposure, or allow their profiles to become outdated. This creates a potential gap between the compatibility score computed by the algorithm and the actual collaborative experience encountered by matched team members. The peer review and rating system partially compensates for this discrepancy over time by surfacing users whose self-assessments diverge from peer evaluations, but newly registered users are assessed solely on declared attributes in the absence of any reputation history.

B. Study Scope and Generalisability

The controlled evaluation involved sixty participants from two engineering colleges located in Maharashtra. This sample size supports preliminary validation but is insufficient to generalise findings confidently across the broader Indian undergraduate engineering population, or across specialised hackathon formats such as hardware-prototyping sprints, domain-specific corporate competitions, or international online events. A larger, geographically distributed study spanning multiple institution types would provide substantially stronger evidence of external validity for the reported efficiency and satisfaction improvements.

C. Cold-Start Challenge

Newly registered users with incomplete profiles receive lower-quality recommendations because the similarity computation has fewer non-zero attribute dimensions to work with. A participant who lists only two skills and one domain preference will generate a sparser feature vector than a fully-profiled user, resulting in lower and less differentiated similarity scores. The mandatory profile-completion workflow and the visible completeness percentage indicator are designed to incentivise thorough self-reporting, but do not fully resolve the degraded recommendation quality experienced during the initial onboarding period.

D. Fixed Weight Configuration

The five attribute-group weights applied in the similarity formula were derived from a single requirement-phase survey and are applied uniformly across all users and all hackathon contexts. In practice, the relative importance of domain alignment versus skill overlap likely varies across event types. A general innovation hackathon may weight domain alignment heavily, while a full-stack development sprint may prioritise specific technical skill

complementarity. An adaptive weighting mechanism that adjusts coefficients based on event type, user feedback, or implicit match-outcome signals would produce more accurate and context-sensitive recommendations than the current static configuration.

ACKNOWLEDGMENT

The authors sincerely thank Dr. P. R. Patil, Professor, School of Computer Science and Engineering, Sandip University, for his sustained guidance, critical feedback, and encouragement throughout the development of this work. Gratitude is also extended to the student participants from Sandip University and collaborating institutions who contributed their time during the user study. The authors acknowledge Microsoft Azure for Students for providing cloud infrastructure credits that supported the deployment of this project.

REFERENCES

1. A. Ounnas, H. C. Davis, and D. Millard, "A Framework for Semantic Group Formation in Education," in Proc. 8th IEEE Int. Conf. Advanced Learning Technologies (ICALT), Santander, Spain, 2008, pp. 34–38.
2. R. M. Carro, E. Pulido, and P. Rodríguez, "Dynamic Generation of Adaptive Internet-based Courses," J. Network and Computer Applications, vol. 22, no. 4, pp. 249–257, 1999.
3. P. Lops, M. de Gemmis, and G. Semeraro, "Content-based Recommender Systems: State of the Art and Trends," in Recommender Systems Handbook, Springer, New York, 2011, pp. 73–105.
4. G. Briscoe and C. Mulligan, "Digital Innovation: The Hackathon Phenomenon," Creativeworks London Working Paper, no. 6, pp. 1–13, 2014.
5. J. Hamari, J. Koivisto, and H. Sarsa, "Does Gamification Work? A Literature Review of Empirical Studies on Gamification," in Proc. 47th Hawaii Int. Conf. System Sciences (HICSS), Waikoloa, HI, USA, 2014, pp. 3025–3034.
6. D. N. T. How, K. S. Cheah, and M. A. Jalil, "Team Formation Algorithm for Collaborative Learning Environment," in Proc. IEEE Symposium on Computers & Informatics (ISCI), Penang, Malaysia, 2012, pp. 204–209.
7. R. Burke, "Hybrid Recommender Systems: Survey and Experiments," User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331–370, Nov. 2002.
8. E. Kuttal, A. Sarma, and G. Rothermel, "Remote Pair Programming: Help or Hindrance?," in Proc. IEEE 28th Int. Conf. Software Maintenance (ICSM), Riva del Garda, Italy, 2012, pp. 524–533.

