ProjectSync-AI: An Intelligent Collaborative Software Engineering Platform for Real-Time Project Tracking and Fair Contribution Assessment

Ankisha Shah[1] Rahul Wagh[2] Prathamesh Shinkar[3] Gauri Randive[4] Mrs. Ashwini R. Garkhedkar[5]
[1,2,3,4,5]Master of Computer Applications (MCA)
[1,2,3,4,5]P.E.S.'s Modern College of Engineering, Pune, Pune, Maharashtra, India

Abstract — Project-based learning is widely used in software engineering courses, yet student projects are usually scattered across separate tools for version control, communication, documentation, task tracking, and submission. This makes it hard to follow progress and to judge what each team member has actually contributed. This paper presents ProjectSync-AI, an intelligent collaborative platform that brings the main activities of academic software development into a single environment, combining skill assessment, task recommendation, project proposal handling, Agile tracking, Git/GitHub activity monitoring, artificial intelligence (AI)-supported contribution analysis, and faculty supervision. The work is organized around four objectives: (1) an AI-driven assessment of students' technical skills, interests, and proficiency; (2) task recommendation based on skills, interests, proficiency, and current workload; (3) real-time tracking of contributions from commits, pull requests, code changes, completed tasks, and code reviews; and (4) a fair and explainable AI-supported evaluation that exposes uneven workloads and gives feedback to students and faculty. The platform is built with React.js, Spring Boot with Spring Security, and PostgreSQL hosted on Supabase. It aims to strengthen teamwork, make contribution assessment more transparent, and reduce the manual workload of faculty supervisors.

Keywords: Software Engineering Education, Artificial Intelligence, Learning Analytics, Academic Project Management, Git Analytics, Contribution Assessment, Task Recommendation, Collaborative Software Development, Educational Technology

I. Introduction

Project-based learning (PBL) has become a mainstay of software engineering education because it gives students first-hand experience of building software [1]–[3]. Through academic projects, learners apply programming, software engineering principles, databases, version control, project management, teamwork, and problem-solving.

In a typical academic project, source code is kept on GitHub, conversations take place in messaging apps, documents are stored on Google Drive or similar services, and planning is handled on task boards such as Trello.

These tools are useful individually, but none of them supports the whole lifecycle of an academic software project. Students must track requirements, tasks, deadlines, development work, and documents across several systems, and faculty must gather information from multiple places just to learn how a project is progressing.

Assessing what each person has contributed to a team project is an especially pressing difficulty. A team can deliver a working product even though its members have participated to very different degrees, and grading that relies chiefly on final demonstrations, documents, and occasional reviews reveals relatively little about how the work was carried out. Earlier studies have examined code-quality metrics, learning analytics, peer evaluation, development logs, and other activity data as evidence of student contributions [4]–[9]. Their findings suggest that ongoing project activity can say more about individual effort than the finished product alone.

ProjectSync-AI draws these ideas together in one academic environment. It monitors project development continuously and applies artificial intelligence (AI) to support skill assessment, task recommendation, and contribution analysis. The study spans software engineering, AI, educational technology, learning analytics, project management, human–computer interaction, collaborative software development, and gamification.

II. Problem Statement and Research Objectives

A. Problem Statement

Software engineering students commonly rely on several unconnected tools for communication, version control, documentation, task management, and project submission. This disjointed workflow hampers collaboration and prevents faculty from following progress across the development lifecycle. Judging fairly what each member has contributed to a group project is a further major problem, because current academic assessment concentrates mostly on the end product rather than on ongoing participation [5]. Moreover, tools such as GitHub, Jira, and Trello were built mainly for professional teams and lack an integrated academic setting that combines proposal management, AI-supported contribution assessment, continuous faculty monitoring, skill assessment, and intelligent task recommendation. An integrated platform is therefore needed that covers the software project lifecycle, gathers meaningful data on development activity, and uses AI to assist in analyzing contributions.

B. Research Objectives

The objectives are adopted as stated in the project synopsis.

1) AI-based skill and interest assessment. The first objective is to build an AI-driven mechanism that assesses students through quizzes and practical coding tasks to determine their technical skills, interests, and proficiency in frontend, backend, database, DevOps, and AI/machine-learning areas. The assessment suggests suitable areas to students and does not push them into fixed roles.
2) Intelligent task recommendation and assignment. The second objective is to pair project tasks with students according to their skills, interests, proficiency, and present workload, supporting project teams without removing human decision-making.
3) Real-time contribution tracking and analysis. The third objective is to capture Git commits, pull requests, code changes, completed tasks, and code reviews through Git/GitHub integration and webhooks, linking each

All rights reserved by www.ijsrd.com

activity to the correct student and project. This matters most when a student works on more than one project, since activity must be attributed to the project it belongs to rather than totaled across everything the student has done.

4) Fair and explainable AI-assisted evaluation. The fourth objective is to examine several kinds of project activity (Git activity, code changes, task completion, pull requests, code reviews, sprint activity, assigned workload, and project milestones) in order to generate insights into contributions, detect uneven workloads, and give feedback to students and teachers. The AI output is meant to help faculty evaluation, not to replace faculty judgment.

C. Research Questions

The objectives give rise to four research questions. RQ1: Can AI-based skill and interest assessment determine the technical proficiency and interests of software engineering students? RQ2: Can a task recommendation mechanism pair project tasks with students by using their skills, interests, proficiency, and workload? RQ3: Can Git and other project activity data serve as continuous evidence of each member's participation in collaborative software projects? RQ4: Can an AI-supported evaluation mechanism merge several activity indicators to deliver explainable insights into contributions and reveal uneven workloads?

III. Literature Review and Research Gap

A. Related Work

Several studies have explored how to assess student contributions and observe collaborative software development. A code-quality evaluation scheme examined how code quality and related development metrics can be used to evaluate student contributions, establishing a basis for treating source code and development data as evidence in academic assessment [4]. Work on unequal contributions in student software engineering teams looked at how participation differs among members and underlines why participation should be observed throughout a project [5]. Learning analytics research has studied how data on student activity can help assess individual contributions, showing that activity logs can reveal further details about how students take part [6]. Function-point-based evaluation has been investigated as a structured way to measure individual contributions in project-based software development [7]. Peer evaluation studies have examined how students can rate the contributions of their teammates [8], [9]; this approach offers helpful insight but would gain from being backed by objective evidence from the development process.

Further work has used learning analytics and activity logs to improve and predict teamwork assessment [10], [11], to assess student behavior from clickstream data [12], and to build dashboards that give instructors a consolidated view of student progress and project status [13]. Multimodal learning analytics has been used to estimate success in collaborative learning [14], and studies of AI support for project management have looked at how AI can be applied to a range of project management tasks [15]. Case studies and surveys on semester-wide PBL, data-science skill development, and software capstone projects add peer assessment practices and assessment rubrics that complement these approaches [1]–[3]. Collectively, these studies lay the groundwork for the four objectives of ProjectSync-AI.

B. Research Gap

Existing research offers several methods for assessing academic projects and analyzing collaborative learning, but each tends to address only one part of software engineering education. The identified gaps are as follows.

1) Fragmentation of project activities. Proposal management, development, task management, documentation, and evaluation are often carried out in separate systems.
2) Limited integration of skill assessment and task allocation. Conventional project management tools usually do not feed information about student competency into task recommendations.
3) Limited continuous contribution monitoring. Conventional evaluation stresses the final result, whereas ongoing development activity can supply extra evidence of participation.
4) Multiple activity sources. A single measure such as the number of commits cannot adequately capture contribution; activity has to be viewed together with tasks, code changes, reviews, and other project work.
5) Limited AI-assisted contribution analysis. There remains room to combine AI with learning analytics and development activity so that faculty receive explainable insights into contributions.

ProjectSync-AI responds to these gaps with an integrated platform for academic software engineering.

IV. Proposed System and Architecture

A. System Overview

ProjectSync-AI is conceived as a single, central environment for collaborative academic software development. Its main components are user authentication, student skill assessment, team formation, project proposal management, faculty supervision, Agile project management, intelligent task recommendation, Git/GitHub integration, contribution analytics, AI-assisted contribution evaluation, a faculty dashboard, notifications, project documentation, and portfolio generation. Together, these components serve the four research objectives.

B. System Architecture

The proposed architecture consists of frontend, backend, database, version-control, and AI layers, as shown in Fig. 1. The React.js frontend supplies interfaces tailored to each user role. Spring Boot delivers the application services and Representational State Transfer (REST) application programming interfaces (APIs), and PostgreSQL hosted on Supabase holds the persistent academic and project data. Software development activity data come from the Git/GitHub integration, and the AI services process the assessment results and project activity data.

All rights reserved by www.ijsrd.com

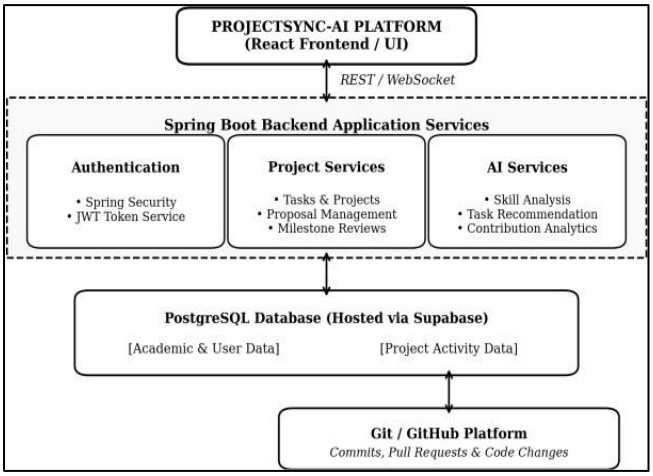


Fig. 1: Architecture of the ProjectSync-AI collaborative platform.

V. SKILL ASSESSMENT AND TASK RECOMMENDATION

A. Student Skill and Interest Assessment

The first objective is to understand which technical skills and interests students have. The assessment may combine multiple-choice technical questions, programming questions, practical coding tasks, and skill-specific questions, from which proficiency information is derived for each technical category (Fig. 2). Conceptually, the skill score of student *s* can be expressed as

$$SkillScore_S = f(Q_S, C_S, P_S) \quad (5.1)$$

where Q_S, C_S, and P_S denote quiz, coding, and practical assessment performance, respectively. These results allow the system to propose suitable technical areas. The outcome is offered as a recommendation only, and students remain free to choose the project role they prefer.

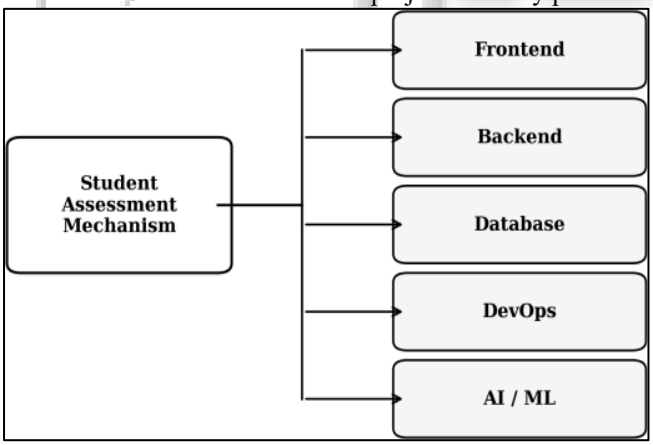


Fig. 2: Student technical skill assessment and recommendation structure.

B. Intelligent Task Recommendation

The second objective concerns pairing project tasks with students. The factors considered are student skills, interests, competency, current workload, task requirements, and project phase. A conceptual recommendation function for student s and task t can be written as

$$R(s, t) = f(S_{match}, I_{match}, C_{match}, W_{availability}) \quad (5.2)$$

where S_{match}, I_{match}, and C_{match} denote skill, interest, and competency compatibility, and $W_{availability}$ denotes workload availability. The system can generate a ranked list of suitable tasks for an individual student or for a whole team. Recommendations do not force a particular allocation, since team members and project supervisors keep control over how tasks are assigned.

VI. PROJECT MANAGEMENT MODULES

A. Project Proposal Management

Proposal management forms part of the academic project lifecycle. A student team selects a faculty guide and prepares a proposal, and the team leader submits it. The faculty either approves the proposal, after which the project is created, or requests modification, after which the team resubmits (Fig. 3). A proposal contains the project title, problem statement, objectives, proposed technology stack, expected outcomes, team members, faculty guide, and project phases. A history of proposals is retained so that earlier submissions and faculty decisions are preserved.

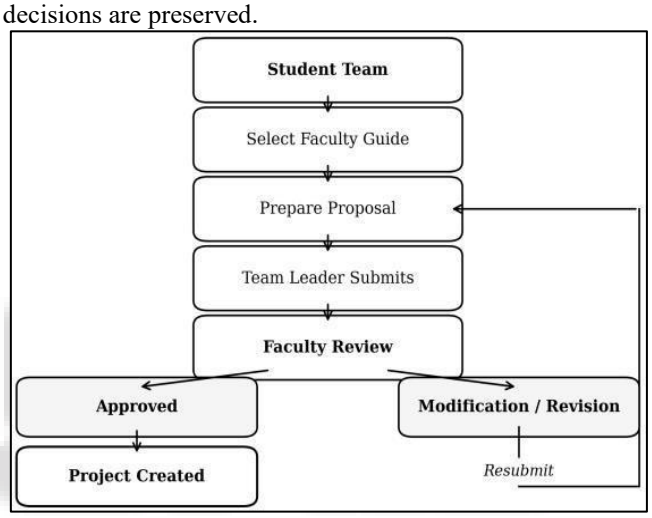


Fig. 3: Project proposal submission, review, and approval lifecycle.

B. Agile Project Tracking

Once a proposal is approved, the project is split into development phases and sprints, for instance requirements, design, development, testing, and deployment (Fig. 4). Every phase may include a start date, end date, deliverables, tasks, a review milestone, and a completion status. Students can be reminded of upcoming deadlines and review milestones, and faculty can adjust academic deadlines where institutional calendars or holidays call for it.

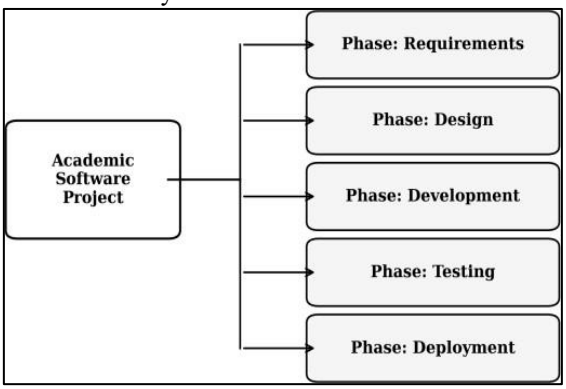


Fig. 4: Academic software engineering project development phases.

All rights reserved by www.ijsrd.com

VII. CONTRIBUTION TRACKING AND AI-ASSISTED EVALUATION

A. Real-Time Contribution Tracking

The third objective concerns the collection of development activity. The system gathers Git commits, pull requests, code changes, code reviews, commit timestamps, files changed, insertions and deletions, task completion, and sprint activity [6]. The proposed workflow, shown in Fig. 5, provisions a dedicated repository for each approved project using JGit, captures students' commits and file differences, maps repository events to authenticated users, projects, tasks, and sprints, and passes the resulting evidence to contribution analysis and the faculty dashboard. Each repository has to be linked to the correct project so that activity from different projects is not wrongly merged.

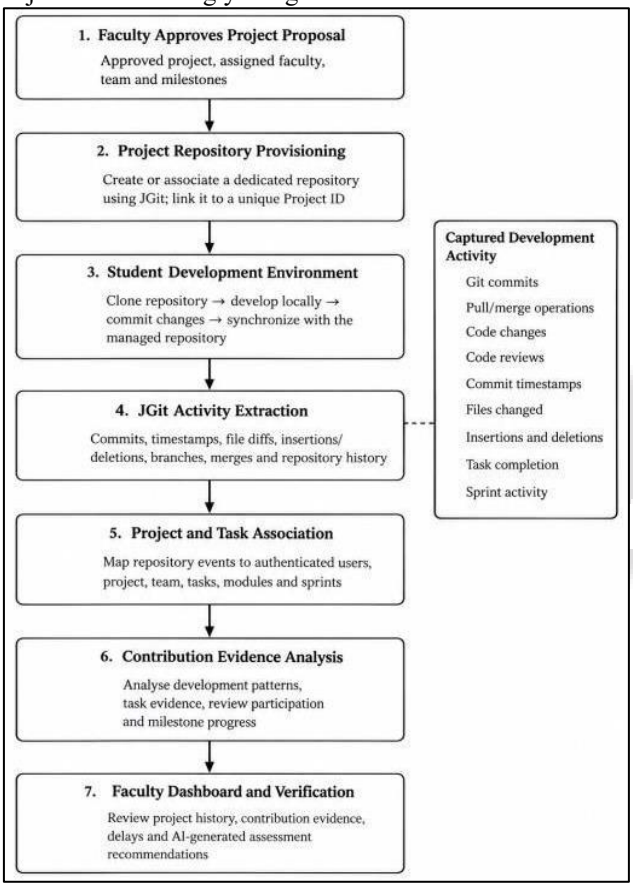


Fig. 5: Real-time Git and GitHub development activity capture workflow.

B. Contribution Analytics

Instead of relying only on the number of commits, the system brings together several indicators. Conceptually, the contribution of student i can be represented as

$$C_i = f(G_i, P_i, T_i, R_i, M_i, W_i) \quad (7.1)$$

where G_i is Git activity, P_i is pull-request activity, T_i is task activity, R_i is review activity, M_i is milestone activity, and W_i is workload. The precise weights for these indicators ought to be settled through experimentation and validation.

This is necessary because the amount of activity alone cannot fully reflect a person's contribution: a few commits may hold a great deal of work, while many commits may hold only trivial changes [4].

C. AI-Assisted Contribution Evaluation

The fourth objective is a fair and explainable AI-supported evaluation mechanism. As shown in Fig. 6, project activity data are collected, features are extracted, contributions are analyzed by an AI model, and explainable insights are passed to faculty review. By analyzing several indicators, the system can identify contribution patterns, workload imbalance, low recent activity, uneven task distribution, development consistency, and differences between assigned and completed work. The result is accompanied by supporting evidence and is not just a single number, as illustrated by the comparison of two students in Table I, from which the AI then produces an explanation.

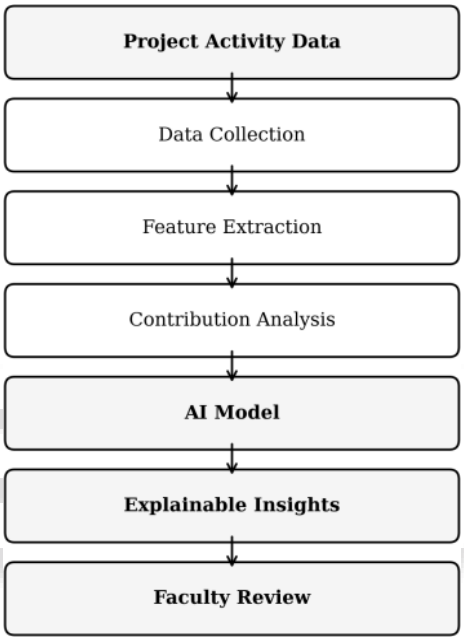


Fig. 6: Multi-stage AI-assisted contribution evaluation and feedback process.

Indicator	Student A	Student B
Commits	Repository data	Repository data
Tasks completed	Project data	Project data
Code reviews	Project data	Project data
Sprint activity	Project data	Project data
Assigned workload	Project data	Project data

Table I: Comparative Activity Indicator Mapping

D. Explainability

Explainability is a key element of the fourth objective. Rather than returning only a score such as "Contribution Score = 82%," the system describes the factors that led to the analysis. Development activity, task completion, code review activity, workload, and milestone completion feed the AI analysis, which yields an explanation together with supporting evidence (Fig. 7). In this way, faculty can examine the evidence behind a result before reaching an academic decision.

All rights reserved by www.ijsrd.com

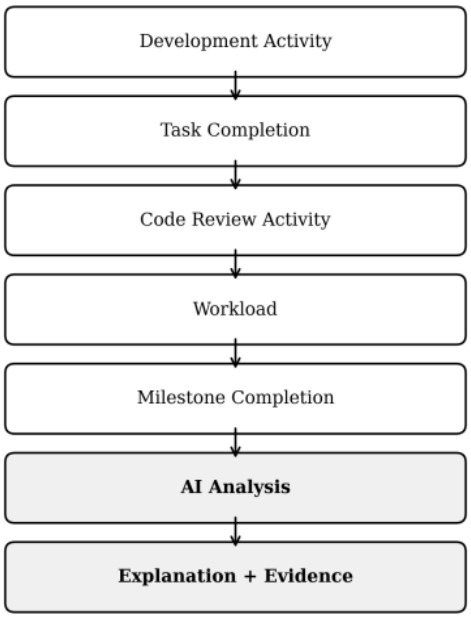


Fig. 7: Flow of explainable factors leading to the faculty evidence summary.

VIII. Faculty Dashboard and Project Monitoring

The faculty dashboard gives a single, central view of all projects under a faculty member's supervision, and the same dashboard allows faculty to supervise several projects [13]. The project listing follows the structure in Table II, and choosing a project reveals the approved proposal, proposal history, project phases, deadlines, tasks, Git activity, contribution analytics, AI-generated insights, and review information.

Project	Leader	Team Members	Phase	Progress	Status
—	—	—	—	—	—

Table II: Faculty Dashboard Project Overview Structure

AI can also help faculty spot the projects that need their attention [15]. Possible indicators are delayed milestones, incomplete tasks, low recent development activity, workload imbalance, missed deadlines, and differences between planned and completed activities. Projects needing attention are sorted into categories by predefined, measurable conditions, with the goal of helping faculty set priorities in their monitoring work.

IX. Security and Database Design

A. Authentication and Security

ProjectSync-AI provides role-based authentication for students, faculty, and administrators. The proposed security mechanisms are JSON Web Token (JWT) authentication, BCrypt password hashing, role-based authorization, faculty authentication and approval, first-login one-time password (OTP) verification, forgot-password functionality, password-strength validation, and unique student and faculty IDs. Security matters greatly because the system holds student details, project data, records of development activity, and academic evaluation information.

B. Database Design

The database is built on PostgreSQL and hosted on Supabase. Its principal entities (users, students and faculty, teams, projects, proposals, reviews, proposal history, tasks, task history, sprints, and Git repositories with their commits and activity) and their relationships are shown in Fig. 8. The relationship between projects and students is designed so that development activity can be attributed correctly when a student takes part in several projects.

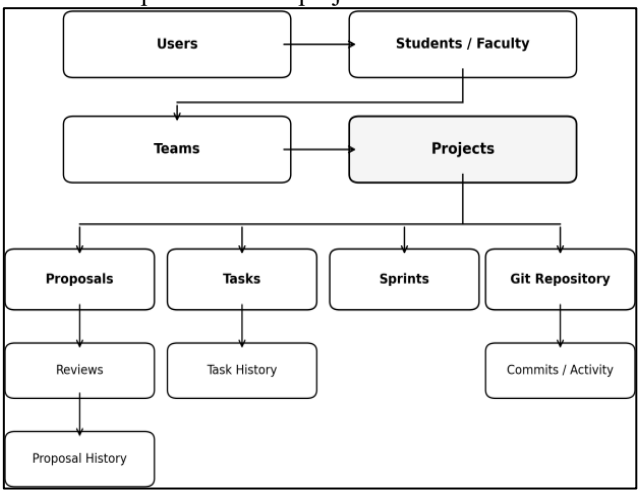


Fig. 8: Relational database schema structure and entity relationships.

X. Research Methodology

This research adopts a design-and-evaluation approach in seven phases. In Phase 1, the problems of academic project management are determined and the system architecture is designed. Phase 2 builds the core database, REST APIs, authentication, authorization, and user management. Phase 3 builds the student, faculty, project, and task modules, covering proposal management, team features, project tracking, and Agile task management. Phase 4 develops and evaluates the mechanisms for skill assessment and task recommendation. Phase 5 integrates Git/GitHub activity and gathers commits, pull requests, code changes, and related project data. Phase 6 carries out functional, integration, AI, security, and usability testing and deployment. Phase 7 compiles the technical documentation, research findings, results, and final project documentation.

XI. Experimental Evaluation

Each of the four research objectives needs its own evaluation. Skill assessment can be evaluated through assessment accuracy, agreement with manually evaluated competency, student feedback, and recommendation relevance. The task recommendation mechanism can be evaluated by comparing its suggested tasks against student skills, interests, faculty assessment, and student-selected tasks, using measures such as recommendation precision and relevance. The Git integration can be evaluated on commit capture accuracy, pull-request capture, code-change detection, project-to-student mapping, timestamp accuracy, and repository synchronization. Finally, the AI-supported contribution analysis can be evaluated by comparing it with manually reviewed project activity, using the metrics

All rights reserved by www.ijsrd.com

$$\text{Accuracy} = \text{Correct Results} / \text{Total Results} \quad (11.1)$$
$$\text{Precision} = TP / (TP + FP) \quad (11.2)$$
$$\text{Recall} = TP / (TP + FN) \quad (11.3)$$
$$\text{F1-Score} = 2PR / (P + R) \quad (11.4)$$

where TP, FP, and FN denote true positives, false positives, and false negatives, and P and R denote the precision and recall from (11.2) and (11.3). The evaluation dataset and its labels should be fixed in advance, before any numerical results are reported.

XII. Expected Outcomes and Scope

The expected outcomes correspond to the four objectives: (1) an AI-driven skill and interest assessment mechanism that analyzes students' quiz and practical assessment results; (2) a task recommendation and assignment mechanism built on skills, interests, proficiency, and workload; (3) a real-time contribution tracking mechanism based on Git commits, pull requests, code changes, completed tasks, and code reviews; and (4) a fair and explainable AI-supported evaluation mechanism that analyzes several kinds of project activity, detects uneven workloads, and gives feedback to students and teachers. The scope of the research covers academic software engineering projects, including student team formation, proposal submission and approval, Agile sprint and task management, Git/GitHub integration, development activity monitoring, AI-assisted contribution analytics, faculty dashboards, skill assessment, task recommendation, and collaborative project tracking.

XIII. Limitations and Future Enhancements

A. Limitations

First, Git activity alone cannot establish the full quality of a person's contribution, and commit counts, lines of code, or pull requests should not be taken as direct measures of the value of software work. Second, analyses and recommendations produced by AI can be wrong and so need to be validated. Third, students follow different development practices, which makes it hard to compare raw activity figures directly. Fourth, the quality of task recommendations depends on the quality of the skill assessment and of the project task metadata. Lastly, the effectiveness of the proposed system still has to be shown through controlled testing and real project data.

B. Future Enhancements

The platform could be extended with more fully integrated version-control features and a browser-based integrated development environment (IDE) that lets students build their projects directly inside it. Gamification through experience points (XP), badges, levels, and achievements could encourage greater student engagement and participation. Future studies might also explore more advanced machine-learning models for analyzing contributions, together with more refined analytics for project risk.

XIV. Conclusion

ProjectSync-AI is put forward as an intelligent collaborative platform for managing academic software projects and assessing contributions. It rests on four objectives: AI-driven assessment of student skills and interests, intelligent task recommendation, real-time tracking of contributions, and fair, explainable AI-supported evaluation. These are combined with proposal management, Agile project tracking, Git/GitHub integration, and faculty dashboards, giving students a central place to collaborate while faculty receive up-to-date information about their projects. Development activity such as commits, pull requests, code changes, completed tasks, and code reviews supplies several independent sources of evidence that AI can process into structured insights for faculty.

The research does not regard AI as a substitute for faculty evaluation. Instead, AI serves as a supporting tool for analyzing project information, recommending tasks, spotting patterns, and presenting explainable evidence. Further experimental evaluation will establish how effective the proposed mechanisms for skill assessment, task recommendation, contribution tracking, and AI-supported evaluation really are.

Acknowledgment

The authors thank their guide, Mrs. Ashwini R. Garkhedkar, and the Department of MCA, Progressive Education Society's Modern College of Engineering, for their guidance and support throughout this work.

References

[1] "Teamwork and peer assessment within semester-wide project-based learning: A case study on an industrial management and engineering degree," IEEE, 2024.
[2] "Project-based learning for scaffolding data scientists' skills," IEEE, 2021.
[3] "Assessment and support for software capstone projects at the undergraduate level: A survey and rubrics," IEEE Trans. Educ., 2012.
[4] "Code-quality evaluation scheme for assessment of student contributions to programming projects," J. Syst. Softw., Elsevier, 2022.
[5] "The impact of unequal contributions in student software engineering team projects," J. Syst. Softw., Elsevier, 2023.
[6] "Assessing individual contribution in a team project using learning analytics," Procedia Comput. Sci., Elsevier, 2020.
[7] "Proposal of an evaluation method of individual contributions using the function point in the implementation phase in project-based learning of software development," Procedia Comput. Sci., Elsevier, 2021.
[8] "A three-year study on peer evaluation in a software engineering project course," IEEE, 2021.
[9] "Exploring peer evaluation methods in group projects of software engineering education," IEEE, 2025.
[10] "Using learning analytics to improve teamwork assessment," Comput. Hum. Behav., Elsevier, 2015.
[11] "Predicting teamwork group assessment using log data-based learning analytics," Comput. Hum. Behav., Elsevier, 2019.

All rights reserved by www.ijsrd.com

[12] "Clickstream for learning analytics to assess students' behavior with Scratch," Future Gener. Comput. Syst., Elsevier, 2024.
[13] "Using learning analytics dashboards to monitor student progress: The case of a blended computer science university course," IEEE, 2024.
[14] "Estimation of success in collaborative learning based on multimodal learning analytics features," IEEE, 2017.
[15] "Multi-criteria measurement of AI support to project management," IEEE, 2023.

All rights reserved by www.ijsrd.com