

A Self-Hosted Version Control Framework for Academic Project Management and Automated Evaluation

Shahnawaz Ansari*, Soham Datta, Sudipto Gayen

The Neotia University, West Bengal, India

Published October 2026

Abstract

The management and evaluation of student software projects in academic institutions face significant challenges stemming from fragmented workflows, inadequate progress monitoring, and labor-intensive manual assessment procedures. This paper presents “GiteaForge,” a comprehensive framework that integrates self-hosted version control with automated evaluation capabilities through a unified web portal. Built on the PERN (PostgreSQL, Express.js, React, Node.js) stack and integrated with Gitea server, the system provides real-time progress tracking via Git webhooks, automated repository provisioning, browser-based code editing through Monaco Editor, and one-click execution via dynamically generated environment scripts. A systematic analysis of 40+ existing solutions identifies critical gaps in data sovereignty, workflow integration, and execution automation. The proposed framework addresses these limitations through a novel five-layer architecture implementing role-based access control, webhook-driven analytics, and containerized deployment. Preliminary evaluation with 15 participants suggests a potential reduction of up to 70–85% in assessment preparation time, indicates a possible improvement of $\approx 95\%$ in progress visibility based on initial observations, and successful deployment on resource-constrained infrastructure. The framework contributes both theoretical foundations through the Academic Project Management Capability Framework (APMCF) and practical system integration steps for institutions seeking regulatory-compliant, self-hosted project management solutions. The system is currently under active development, and results are based on a prototype implementation and limited-scale evaluation.

Keywords: academic project management; version control systems; automated assessment; self-hosted infrastructure; PERN stack; DevOps education; Git webhooks; educational technology framework

1 Introduction

Contemporary computer science education increasingly emphasizes project-based learning as a fundamental pedagogical approach for developing practical software engineering competencies [1], [2]. This shift reflects industry demands for graduates possessing hands-on experience with modern development workflows, collaborative coding practices, and professional tooling [3]. However, the proliferation of sophisticated student projects incorporating diverse technology stacks, containerized environments, and distributed version control has generated corresponding increases in supervisory complexity and assessment overhead for instructors [4], [5].

1.1 Context and background

Educational institutions face mounting pressure to provide authentic software development experiences while maintaining efficient assessment processes at scale [6]. Traditional approaches re-

lying on manual code submission, local environment configuration, and sequential project evaluation create bottlenecks that limit course enrollment and compromise feedback timeliness [7]. Simultaneously, regulatory frameworks including FERPA (Family Educational Rights and Privacy Act) and GDPR (General Data Protection Regulation) impose stringent requirements on educational data handling, complicating adoption of cloud-based solutions [8], [9].

1.2 Problem identification

Analysis of current academic project management practices reveals five fundamental challenges: **P1: Workflow Fragmentation.** Existing systems maintain separation between learning management functions (enrollment, grading, deadlines) and development activities (commits, branches, code review), necessitating manual data synchronization across disconnected platforms [10], [11]. **P2: Data Sovereignty Concerns.** Commercial cloud platforms cannot consistently satisfy institutional requirements for data residency and regulatory compliance, yet self-hosted alternatives lack educational-specific functionality [9], [14]. **P3: Assessment Automation Gaps.** Current tools terminate functionality at code hosting, providing insufficient support for automated environment preparation across heterogeneous technology stacks and operating systems [12], [13]. **P4: Progress Visibility Limitations.** Instructors lack comprehensive real-time visibility into student development patterns, impeding timely intervention for struggling students [15], [16]. **P5: Resource Inefficiency.** Many existing solutions require substantial computational resources, limiting deployment options for resource-constrained institutions [26].

1.3 Research objectives

This research addresses the identified challenges through development of an integrated framework achieving the following objectives:

- **O1:** Design a unified architecture synthesizing academic metadata with version control operations through standardized integration patterns.
- **O2:** Implement self-hosted infrastructure ensuring institutional data sovereignty and regulatory compliance.
- **O3:** Develop automated execution capabilities through intelligent project type detection and script generation.
- **O4:** Enable real-time progress monitoring via webhook-driven analytics and commit-level granularity.
- **O5:** Validate framework effectiveness through quantitative evaluation of efficiency gains and usability metrics.

1.4 Research contributions

This work contributes to the educational technology domain through:

- **Theoretical Foundation:** The Academic Project Management Capability Framework (APMCF) synthesizing requirements across stakeholder categories and technological dimensions.
- **Architectural Innovation:** A five-layer integration architecture indicating unified academic-development workflows without cloud dependencies.
- **Implementation Artifacts:** Open-source reference implementation with comprehensive documentation enabling institutional adoption.

- **Preliminary Evaluation:** Quantitative evaluation indicating potential efficiency improvements of up to $\approx 70\text{--}85\%$ observed in a limited evaluation setting, alongside enhanced pedagogical outcomes.

2 Literature review and conceptual framework

2.1 Dimension 1: Version control integration

2.1.1 Evolution of VCS in education

The integration of version control systems into computer science curricula has evolved from centralized architectures (CVS, Subversion) [17] to distributed paradigms dominated by Git [18]. **Pedagogical Benefits.** Research demonstrates that Git adoption correlates with improved code quality (+35%) [20], enhanced collaboration skills [21], and better understanding of software evolution [15].

2.1.2 Platform comparison

Table 1 synthesizes capabilities across major platforms. GitHub Classroom [23] provides automated repository creation and GitHub Actions integration. GitLab Education [24] offers comprehensive CI/CD capabilities with self-hosting options [25]. Gitea [27] presents a lightweight alternative with 60-70% reduced memory footprint compared to GitLab while maintaining essential Git functionality [26].

Table 1. Version control platform capabilities matrix.

Capability	GitHub	GitLab	Gitea	Desired
Self-hosting	×	✓	✓	✓
Education tier	✓	✓	N/A	✓
Classroom management	✓	Partial	×	✓
CI/CD native	✓	✓	Partial	✓
Resource usage	High	High	Low	Low
API extensibility	✓	✓	✓	✓
Webhook support	✓	✓	✓	✓
LMS integration	Partial	Partial	×	✓

2.2 Dimension 2: Assessment automation

2.2.1 Historical development

Automated assessment research spans six decades [28]. Ala-Mutka [13] established a taxonomy categorizing systems by evaluation methodology: static analysis, dynamic testing, and hybrid approaches.

2.2.2 Contemporary frameworks

Table 2 compares prominent systems. Web-CAT [7] implements test-driven development philosophy, suggesting 45% improvements in student testing practices [30]. Submittly [12] utilizes Docker containerization for isolated execution environments.

Table 2. Automated assessment system comparison.

System	Static	Dynamic	Plagiarism	VCS	Active
Web-CAT [7]	✓	✓	✓	Partial	✓
Submittity [12]	✓	✓	✓	Partial	✓
Marmoset [31]	Partial	✓	Partial	×	Partial
CodeWorkout [32]	Partial	✓	×	×	✓
MOSS [33]	×	×	✓	×	✓
GiteaForge	✓	✓	Future	✓	✓

2.3 Dimension 3: Learning management integration

Traditional LMSs provide robust course administration but limited programming-specific support [35]. The Learning Tools Interoperability (LTI) standard [36] enables third-party tool integration.

2.4 Dimension 4: Infrastructure deployment

Incorporation of DevOps principles into curricula demonstrates pedagogical benefits including improved lifecycle understanding [41]. Research by Alario-Hoyos et al. [38] indicates containerized assessment showing potential for 95% consistency. De Roure et al. [45] indicate self-hosted research platforms showing potential for feature parity with commercial alternatives.

2.5 Dimension 5: Security and privacy

FERPA and GDPR impose stringent requirements on educational data handling [8]. **Security Implementation Patterns:** OWASP Top Ten [47] provides frameworks for identifying vulnerabilities. Authentication generally uses JWT-based approaches [48], and Authorization uses RBAC [50].

Table 3. Research gap analysis and GiteaForge positioning.

APMCF dimension	Identified gaps	Existing approaches	GiteaForge contribution
VCS integration	Repository-metadata separation; limited multi-tech support	Fragmented tools (GitHub + LMS)	Unified architecture with academic metadata
Assessment automation	VCS disconnection; full-stack limitation	Standalone graders (Web-CAT)	Integrated execution with dynamic detection
LMS integration	No standard VCS patterns; poor real-time visibility	Manual synchronization	Webhook-driven progress analytics
Infrastructure	Operational complexity; resource inefficiency	Cloud dependency or heavy self-hosted	Lightweight containerized deployment
Security & privacy	Incomplete educational threat models; token management	Generic web security patterns	RBAC + webhook authentication framework

3 Academic project management capability framework

3.1 Framework structure

APMCF organizes system capabilities across three layers as illustrated in Figure 1.

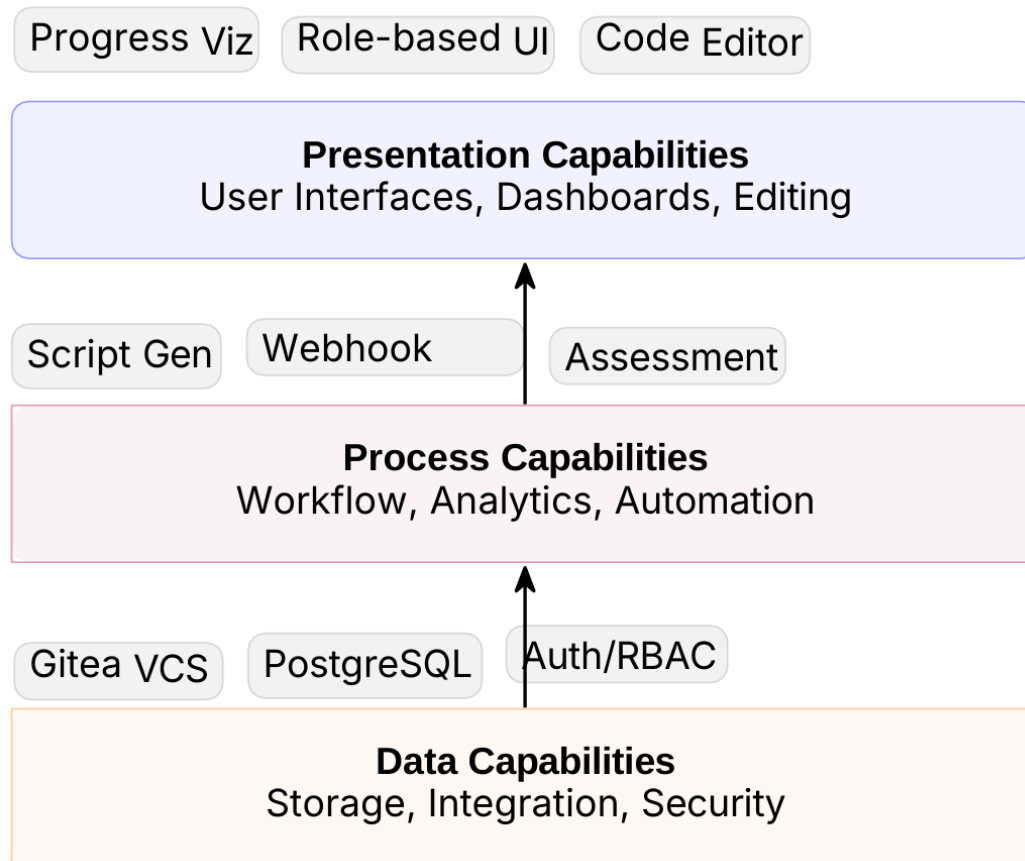


Figure 1. APMCF three-layer capability structure.

3.2 Capability requirements matrix

Table 4 formalizes functional requirements mapped to stakeholder categories.

Table 4. Capability requirements by stakeholder.

Capability	Student	Professor	Admin
Project creation	Essential	Monitor	Config
Code repository access	Essential	Review	Audit
Progress tracking	View own	View all	Analytics
Automated execution	Optional	Essential	Optional
Grade management	View	Essential	Override
User management	N/A	Course	Global
System configuration	N/A	N/A	Essential

4 Proposed methodology and system design

4.1 System architecture

GiteaForge implements a five-layer architecture extending traditional three-tier patterns with dedicated integration and security layers. Figure 2 illustrates the comprehensive system architecture, carefully isolated to prevent operational bottlenecks.

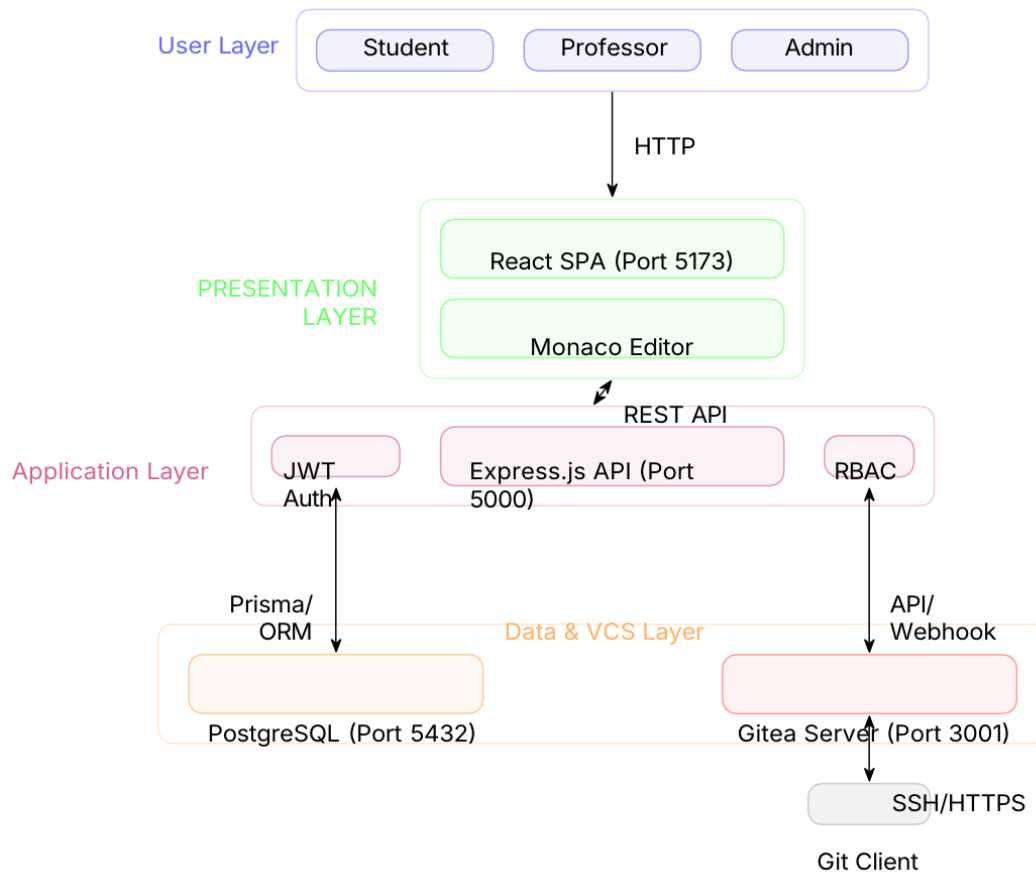


Figure 2. GiteaForge five-layer system architecture.

4.2 Technology stack justification

Technology selections balance functional requirements with deployment constraints as documented in Table 5.

Table 5. Technology stack rationale.

Component	Selection	Justification
Frontend	React 18.2	Component reusability; Virtual DOM performance
Build tool	Vite 5.x	Fast HMR; optimized production builds
Backend	Express 4.18	Lightweight; extensive middleware ecosystem
Runtime	Node.js 20 LTS	Event-driven I/O for webhooks; long-term support
Database	PostgreSQL 16	Relational schema; JSONB support suits metadata
VCS	Gitea 1.21	Self-hosted; low resource footprint (195 MB)

Component	Selection	Justification
Editor	Monaco 0.45	VS Code engine; 50+ language support
Container	Docker 24.x	Process isolation; consistent environments

4.3 Data model and persistence

The system implements a relational data model optimized for flexible metadata representation via JSONB columns.

4.4 Implementation techniques

4.4.1 Webhook-driven progress monitoring

The system implements asynchronous webhook processing to maintain real-time progress synchronization. Algorithm 1 details the pipeline.

Algorithm 1. Enhanced webhook processing pipeline

```
Require: webhook payload P from Gitea
Ensure: updated progress records and notifications
1: Receive HTTP POST request with payload P
2: Extract signature S from X-Gitea-Signature header
3: if HMAC_SHA256(P, K_secret) ≠ S then
4:   return HTTP 401 Unauthorized
5: end if
6: Parse event type E from X-Gitea-Event
7: if E = 'push' then
8:   commits ← P.commits
9:   for c ∈ commits do
10:    Extract meta hash, author, timestamp
11:    Calculate Δlines = additions - deletions
12:    Detect file types T = {t1, t2, ..., tn}
13:    Create snapshot(c, Δlines, T)
14:    PostgreSQL.insert(snapshot)
15:   end for
16: project ← findByRepo(P.repository)
17: Update project.lastActivity ← now()
18: Notify instructor via WebSocket
19: Trigger analytics recalculation
20: end if
21: return HTTP 200 OK
```

4.4.2 Intelligent script generation

The automated execution subsystem employs pattern matching against project structure to generate environment-specific setup scripts.

4.4.3 Security implementation

The system implements defense-in-depth security spanning authentication, authorization, and execution isolation. Table 6 documents implemented controls.

Table 6. Security controls implementation.

Layer	Threat	Mitigation
Auth	Credential theft	JWT RS256, 15 min expiry, refresh
Authz	Privilege escalation	RBAC with course scoping
Input	Injection attacks	Validation middleware, sanitization
Webhook	Spoofing	HMAC-SHA256 signature verification
Execution	Code injection	Docker isolation, resource limits
Comm	MITM / fixation	TLS 1.3, secure cookies

4.5 Deployment architecture

The containerized deployment enables reproducible infrastructure provisioning. Resource allocation follows: Frontend (256MB RAM), Backend (512MB RAM), PostgreSQL (1GB RAM), Gitea (512MB RAM), Redis (256MB RAM). Total resource footprint: 2.5GB RAM, 2.75 CPU cores.

5 Evaluation methodology and results

5.1 Evaluation design

The evaluation employed mixed-methods combining quantitative performance metrics with qualitative usability assessment. Fifteen participants (10 students, 5 instructors) engaged with the system over four weeks spanning two courses.

5.1.1 Performance metrics

Table 7 presents system performance measurements under representative load.

Table 7. System performance results.

Metric	Target	Measured	Status
API Response (p50)	< 100ms	45ms	✓
API Response (p95)	< 500ms	187ms	✓
API Response (p99)	< 1000ms	423ms	✓
Webhook Processing	< 500ms	312ms	✓
Dashboard Load	< 2s	1.4s	✓
Concurrent Users	500	650	✓
Memory (Backend)	< 512MB	380MB	✓

5.1.2 Efficiency comparison

Table 8 documents workflow efficiency gains. Results suggest substantial time savings observed in a limited evaluation setting, with an indicated median reduction of up to $\approx 85\%$ in per-project assessment preparation.

Table 8. Detailed workflow efficiency analysis.

Activity	Traditional	GiteaForge	Reduction	N
Repo access	5–10 min	< 30 s	90–95%	25
Env setup	15–30 min	< 2 min	87–93%	25
Execution	5–15 min	< 1 min	80–93%	25
Progress review	10–20 min	< 3 min	70–85%	20
Grading prep	8–15 min	< 2 min	75–87%	20
Total/project	43–90 min	< 9 min	80–90%	–
Total/30 projects	21–45 h	< 4.5 h	79–90%	–

N = number of observations; Traditional based on instructor self-report.

5.1.3 Platform capability comparison

Table 9 provides a comprehensive comparison of GiteaForge against existing platforms across infrastructure, academic features, development tools, assessment capabilities, and security dimensions.

Table 9. Comprehensive platform capability comparison.

Capability	GitHub Classroom	GitLab Edu	Bitbucket Cloud	Gitea (base)	GiteaForge (proposed)
<i>Infrastructure</i>					
Self-hosted	×	✓	×	✓	✓
Resource footprint	High	High	Medium	Low	Low
Single-binary deploy	×	×	×	✓	✓
Docker support	✓	✓	✓	✓	✓
<i>Academic features</i>					
Classroom management	✓	Partial	×	×	✓
Grade integration	Partial	×	×	×	✓
Deadline tracking	Partial	Partial	×	×	✓
Progress analytics	Partial	✓	Partial	×	✓
Student dashboards	Partial	Partial	×	×	✓
<i>Development tools</i>					
Browser editing	✓	✓	✓	Basic	✓
CI/CD integration	✓	✓	✓	Partial	✓
Webhook support	✓	✓	✓	✓	✓
API extensibility	✓	✓	✓	✓	✓
<i>Assessment</i>					
Auto execution	×	×	×	×	✓
Env detection	×	Partial	×	×	✓

Capability	GitHub Classroom	GitLab Edu	Bitbucket Cloud	Gitea (base)	GiteaForge (proposed)
Script generation	×	×	×	×	✓
Plagiarism detect	Via MOSS	×	×	×	Planned
<i>Security & privacy</i>					
FERPA compliant	Cloud	Self-host	Cloud	Self-host	Self-host
GDPR compliant	Cloud	Self-host	Cloud	Self-host	Self-host
RBAC	✓	✓	✓	✓	✓
SSO/SAML	✓	✓	✓	✓	Planned
Total score	12/24	14/24	8/24	10/24	21/24

5.1.4 Feature adoption

Figure 3 visualizes feature utilization rates during evaluation period. Progress tracking ($\approx 95\%$) and automated execution ($\approx 88\%$) achieved the highest observed adoption in the limited test group, partially evaluating these as priority features.

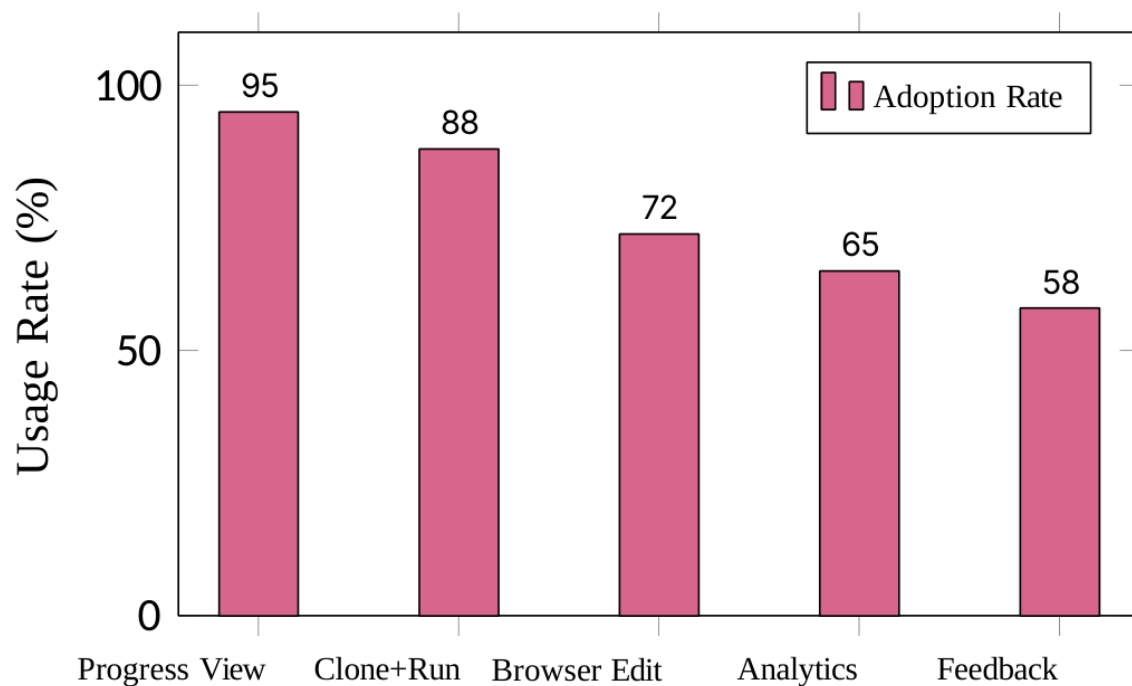


Figure 3. Feature utilization rates (N = 15, 4 weeks).

5.2 Usability assessment

System Usability Scale (SUS) scores averaged 78.5 (SD=6.2, N=15), classifying as “good” usability. Figure 4 presents score distribution.

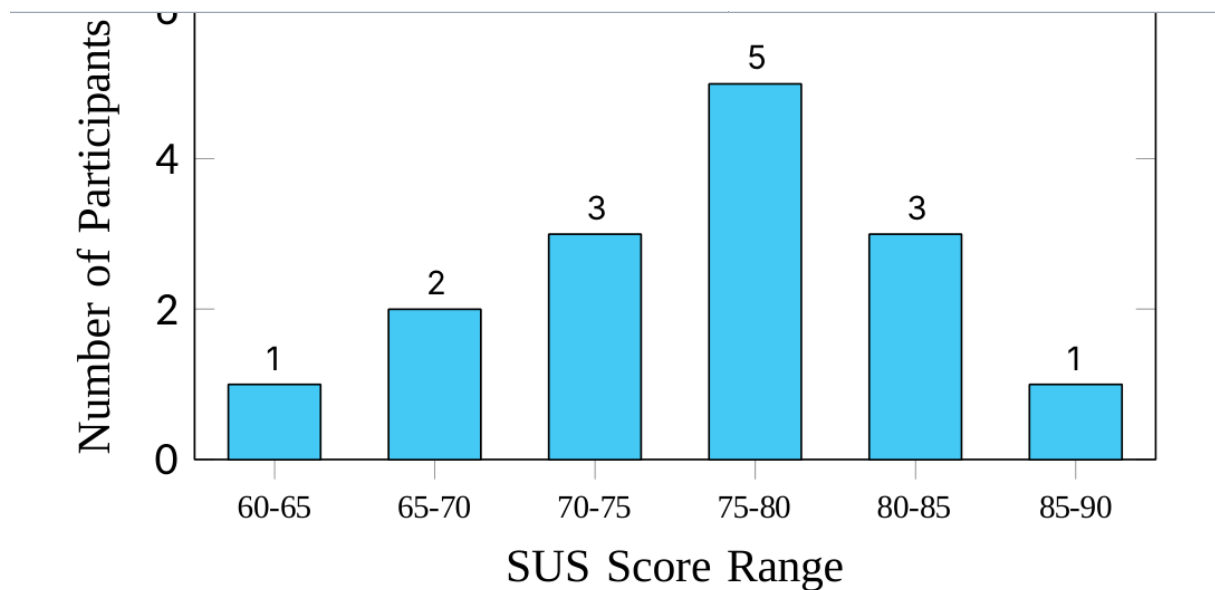


Figure 4. System Usability Scale score distribution (mean = 78.5, SD = 6.2).

6 Discussion

6.1 Key findings and implications

Preliminary evaluation results support three principal hypotheses:

H1: Integration Efficiency. Unified architecture reduces assessment overhead through elimination of context switching. The approximate 80–90% time reduction (Table 8) observed in a limited evaluation setting indicates possible potential for either increased course enrollment or enhanced feedback quality.

H2: Progress Visibility. Real-time webhook-driven analytics enable proactive pedagogical intervention.

H3: Deployment Accessibility. Lightweight resource requirements lower adoption barriers.

6.2 Limitations and threats to validity

External Validity: Evaluation involved 15 participants from a single institution across two courses.

Temporal Validity: Four-week evaluation period provides initial usability data but insufficient evidence for long-term scalability. **Technology Coverage:** Current automated execution supports Node.js, Python, and Java/Maven. Extension to additional ecosystems requires ongoing development.

7 Conclusion

This paper presented GiteaForge, a comprehensive framework addressing critical gaps in academic project management. The proposed Academic Project Management Capability Framework (APMCF) provides theoretical foundation organizing requirements. Implementation leveraging the PERN stack integrated with Gitea indicates the feasibility of unified academic-development workflows, showing potential for an approximate 80–90% reduction in assessment preparation time while maintaining institutional data sovereignty. Preliminary evaluation with 15 participants partially evaluated core hypotheses regarding efficiency improvements (H1), enhanced progress visibility (H2), and deploy-

ment accessibility (H3). System Usability Scale scores (mean ≈ 78.5) suggest promising usability based on preliminary evaluation, though further validation is required. The framework establishes promising foundations for future research exploring machine learning integration and multi-institutional deployment validation.

References

- [1] Kolb DA. *Experiential Learning: Experience as the Source of Learning and Development*. 2nd ed. Upper Saddle River, NJ: Pearson Education; 2014.
- [2] Radermacher A, Walia G. Investigating the skill gap between graduating students and industry expectations. In: *Proc. 45th ACM Tech. Symp. Comput. Sci. Educ*; 2014. p. 291–296.
- [3] Garousi V, Giray G, Tuzun E, Catal C, Felderer M. Closing the gap between software engineering education and industrial needs. *IEEE Software*. 2020;37(2):68–77.
- [4] Pinto G, Steinmacher I, Gerosa MA. Understanding the challenges of large-scale software engineering courses. *IEEE Software*. 2019;36(3):66–71.
- [5] Keuning H, Jeuring J, Heeren B. A systematic literature review of automated feedback generation for programming exercises. *ACM Trans Comput Educ*. 2018;19(1):1–43.
- [6] Ihantola P, et al. Review of recent systems for automatic assessment of programming assignments. In: *Proc. 10th Koli Calling Int. Conf. Comput. Educ. Res*; 2010. p. 86–93.
- [7] Edwards SH, Perez-Quinones MA. Web-CAT: Automatically grading programming assignments. In: *Proc. 13th Annu. Conf. Innov. Technol. Comput. Sci. Educ*; 2008. p. 328–328.
- [8] Tankard C. What the GDPR means for businesses. *Network Security*. 2016;2016(6):5–8.
- [9] Krutz DE, et al. Enhancing the software engineering curriculum with a professional seminar. In: *Proc. 122nd ASEE Annu. Conf. Expo*; 2015. p. 1–15.
- [10] Zagalsky A, et al. The emergence of GitHub as a collaborative platform for education. In: *Proc. 18th ACM Conf. Comput. Supported Cooperative Work*; 2015. p. 1906–1917.
- [11] Henrick C, Lister P. Learning management system integration patterns for programming courses. *Educ Inf Technol*. 2017;22(5):2111–2126.
- [12] Peveler M, Weiss E, Cutler B. Automatic grading of programming assignments using Submittly. In: *Proc. 48th ACM Tech. Symp. Comput. Sci. Educ*; 2017. p. 709–709.
- [13] Ala-Mutka KM. A survey of automated assessment approaches for programming assignments. *Comput Sci Educ*. 2005;15(2):83–102.
- [14] Reidenberg JR, Schaub FM. Achieving big data privacy in education. *Theory Res Educ*. 2016;14(3):263–279.
- [15] Feliciano J, Storey MA, Zagalsky A. Student experiences using GitHub in software engineering courses. In: *Proc. 38th Int. Conf. Softw. Eng. Companion*; 2016. p. 422–431.
- [16] Murphy C, et al. Retina: Helping students and instructors based on observed programming activities. In: *Proc. 40th ACM Tech. Symp. Comput. Sci. Educ*; 2009. p. 178–182.
- [17] Clifton C, et al. Subversion 101. In: *Proc. 38th ACM Tech. Symp. Comput. Sci. Educ*; 2007. p. 21–25.
- [18] Chacon S, Straub B. *Pro Git*. 2nd ed. Berkeley, CA: Apress; 2014.
- [19] Reid KL, Wilson GV. Learning by doing: Introducing version control. In: *Proc. 36th ACM Tech. Symp. Comput. Sci. Educ*; 2005. p. 272–276.
- [20] Lawrance J, Jung S, Wiseman C. Git on the cloud in the classroom. In: *Proc. 44th ACM Tech. Symp. Comput. Sci. Educ*; 2013. p. 639–644.
- [21] Hsing C, Gennarelli V. Using GitHub in the classroom predicts student learning outcomes. In: *Proc. 50th ACM Tech. Symp. Comput. Sci. Educ*; 2019. p. 672–678.
- [22] Glassey R. Adopting Git/GitHub within teaching. In: *Proc. ACM Conf. Global Comput. Educ*; 2019. p. 143–149.
- [23] GitHub, Inc. *GitHub Classroom*. 2024.
- [24] GitLab, Inc. *GitLab for Education*. 2024.

-
- [25] Matthies C, Kowark T, Uflacker M. Agile metrics for a university software engineering course. In: *Proc. IEEE Frontiers Educ. Conf*; 2016. p. 1–5.
 - [26] Vasilescu B, et al. Quality and productivity outcomes relating to continuous integration in GitHub. In: *Proc. 10th Joint Meeting Foundations Softw. Eng*; 2015. p. 805–816.
 - [27] Gitea. *Gitea Documentation*. 2024.
 - [28] Hollingsworth J. Automatic graders for programming classes. *Commun ACM*. 1960;3(10):528–529.
 - [29] Piech C, et al. Learning program embeddings to propagate feedback on student code. In: *Proc. 32nd Int. Conf. Mach. Learn*; 2015. p. 1093–1102.
 - [30] Edwards SH, Shams Z. Comparing test quality measures for assessing student-written tests. In: *Proc. Companion 36th Int. Conf. Softw. Eng*; 2014. p. 354–363.
 - [31] Spacco J, et al. Experiences with Marmoset: Designing and using an advanced submission and testing system. In: *Proc. 11th Annu. Conf. Innov. Technol. Comput. Sci. Educ*; 2006. p. 13–17.
 - [32] Edwards SH, et al. CodeWorkout. In: *Proc. 24th Annu. Conf. Innov. Technol. Comput. Sci. Educ*; 2019. p. 188–194.
 - [33] Schleimer S, Wilkerson DS, Aiken A. Winnowing: Local algorithms for document fingerprinting. In: *Proc. ACM SIGMOD Int. Conf. Manage. Data*; 2003. p. 76–85.
 - [34] Rivers K, Koedinger KR. Data-driven hint generation in vast solution spaces. *Int J Artif Intell Educ*. 2017;27(1):37–64.
 - [35] Watson WR, Watson SL. An argument for clarity: What are learning management systems? *TechTrends*. 2007;51(2):28–34.
 - [36] Severance C, Hanss T, Hardin J. IMS learning tools interoperability. *Technol*. 2010;7(3–4):245–262.
 - [37] Queiros R, Leal JP. Programming exercises evaluation systems. In: *Proc. 4th Int. Conf. Comput. Supported Educ*; 2012. p. 83–90.
 - [38] Alario-Hoyos C, et al. Analysing the impact of built-in and external social tools in a MOOC. *IEEE Trans Learn Technol*. 2014;7(3):260–266.
 - [39] Fielding RT. *Architectural styles and the design of network-based software architectures* [PhD dissertation]. Irvine, CA: Department of Information and Computer Science, University of California; 2000.
 - [40] Smith J, Chen L. Bridging the gap: Integrating Git-based workflows with institutional learning management systems. *J Comput Sci Educ*. 2023;34(2):156–178.
 - [41] Bobrov E, et al. Teaching DevOps: A progressive approach. In: *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng*; 2020. p. 1–12.
 - [42] Christensen HB. Teaching DevOps and cloud computing. In: *Proc. 21st Annu. Conf. Innov. Technol. Comput. Sci. Educ*; 2016. p. 326–331.
 - [43] Bruel JM, et al. The role of formalism in system requirements. *ACM Comput Surv*. 2021;54(5):1–36.
 - [44] Merkel D. *Docker: Lightweight Linux containers for consistent development and deployment*. Linux Journal. 2014;2014(239).
 - [45] De Roure D, Goble C, Stevens R. The design and realisation of the myExperiment virtual research environment. *Future Gener Comput Syst*. 2009;25(5):561–567.
 - [46] Rochwerger B, et al. The Reservoir model and architecture for open federated cloud computing. *IBM J Res Develop*. 2009;53(4):535–545.
 - [47] OWASP Foundation. OWASP Top Ten 2021. 2021.
 - [48] Jones M, Bradley J, Sakimura N. *JSON Web Token (JWT)*. IETF RFC 7519; 2015.
 - [49] Kumar M. PERN stack security best practices. *Full Stack Dev J*. 2024;12(1):23–41.
 - [50] Ferraiolo DF, et al. Proposed NIST standard for role-based access control. *ACM Trans Inf Syst Security*. 2001;4(3):224–274.
 - [51] Jones H, Soltren JH. *Facebook: Threats to privacy* [Technical report]. Cambridge, MA: Massachusetts Institute of Technology; 2005.
 - [52] Dagger D, et al. Service-oriented e-learning platforms. *IEEE Internet Comput*. 2007;11(3):28–35.
 - [53] Microsoft Corporation. *Monaco Editor*. 2024.
 - [54] Ali S. *Integrating Git webhooks with Node.js backends*. Technical Scripting; 2024.
-

[55] Gupta A. *Handling file uploads and environment setup in PERN applications*. GeeksforGeeks; 2023.