iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85508

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

ProjectHub: A Retrieval-Augmented Generation Based Academic Project Portal for Discovering and Extending Senior Mini Projects

Sathya B[1], Ezhilarasu L[2], Hariharan R[3], Deepan P[4]
Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, India

Abstract: ***Every year colleges complete many mini projects, but the finished work is usually kept as a printed report or a scattered soft copy and is forgotten when the batch leaves. Juniors cannot easily find out whether an idea has already been done by seniors, so topics are repeated and earlier work is not extended. This paper presents ProjectHub, a web-based academic project portal whose main feature is a retrieval-augmented generation (RAG) assistant over the database of senior projects. Each senior project is stored as a knowledge document with its title, batch, authors, description and technologies, and a local embedding service converts it into a 512-dimension vector. When a student asks a question, the question is converted in the same way, the most similar stored projects are retrieved with their similarity scores, and a large language model served through the Groq API composes the reply from the retrieved projects only. If no project is similar enough, the assistant says so and never claims that the idea is new. Statistics questions are routed to exact database counts instead. The portal also provides project submission, team task tracking, faculty review, a searchable repository and role-based dashboards, built with React, Spring Boot and MySQL. The architecture, algorithms and prototype are described, and a reproducible evaluation plan is given.***

Keywords: ***Retrieval-Augmented Generation (RAG), Groq API, Academic Project Management, Semantic Search, Project Repository, AI Assistant, Spring Boot, React.***

I. INTRODUCTION

Mini projects are an important part of engineering education, because they let students apply theory to a practical problem. In many colleges, however, the project ends with a printed report and a soft copy kept by the team or in the department. When the batch graduates, the title, the technologies and the lessons learned are practically lost. The next batch starts again from an empty page, often choosing the same popular topics, and the earlier prototype is never improved towards a product that could actually be used.

Existing academic project tools address the management side of this problem: they collect proposals, organise teams, track tasks and store files [1]–[9]. They do not help a junior who asks a simple question: “has this idea already been done by the seniors, and what exactly did they build?” A keyword search over titles cannot answer it, and a general chatbot answers from its own training data, may invent projects that do not exist and cannot know what a particular college has done [12].

This paper presents ProjectHub, a unified *web-based academic project management system* whose main feature is a *retrieval-augmented generation (RAG) assistant* over the database of senior projects. The assistant retrieves the most similar earlier projects, shows their title, batch, authors and similarity score, and lets a language model accessed through the *Groq API* compose the reply from those projects only. A junior team can therefore see what has been done and decide either to choose a different topic or to extend the earlier work. The same portal provides project submission, team collaboration, faculty review, a searchable repository and progress dashboards, so that every new project enters the knowledge base through the normal workflow.

II. LITERATURE SURVEY

1) B. Anbarolu, “A Collaborative GIS Programming Course Using GitHub Classroom”, Transactions in GIS (2021). Student teams developed software plugins using GitHub Classroom, which let the lecturer monitor group progress and give timely feedback. The study shows the value of collaborative learning and progress monitoring, but it is limited to one course and programming environment and is not a general platform for project submission, management and showcasing. [1]
2) Cs. Z. Kertész, “Using GitHub in the Classroom – A Collaborative Learning Experience”, SIITME (2015). GitHub is used in laboratory assignments to encourage collaboration and interaction among students, and collaborative practice is compared with traditional approaches. The study concerns laboratory work and gives no system for project review, progress dashboards or showcasing completed projects. [2]

©IJRASET: All Rights are Reserved

3) M. A. Nelson and L. Ponciano, "Experiences and Insights from Using GitHub Classroom to Support Project-Based Courses", SEENG (2021). GitHub Classroom serves as a shared repository for project-based software engineering courses, and the authors point to further opportunities for monitoring skills and student portfolios. The approach depends on a software development platform and offers no integrated management and presentation environment. [3]

4) "CapStone: A Cloud-Based Platform for Multi-Party Collaboration on Capstone Projects", IEEE FIE (2016). A collaborative platform supports students, faculty advisors and industry mentors with planning, scheduling, progress reporting, feedback collection and report preparation, and users found these functions helpful. Its focus is capstone collaboration, and completed projects are not made available for later reference. [4]

5) S. Varikuti, "A Web-Based Online Collaboration Platform for Formulating Engineering Design Projects", M.S. thesis, Purdue University (2014). A web platform lets students and faculty share project ideas, discuss proposals and give feedback before a semester begins. It supports early-stage collaboration, but its emphasis is on project formulation rather than execution, monitoring and showcasing of completed work. [5]

6) "Scholarize: A Digital Ecosystem for Enhanced Research and Collaboration", ISHE (2025). A digital platform combines a research repository with task management and file storage to reduce scattered information between student groups and advisors. It concerns research projects, reports evaluation at one institution and describes no question answering over the stored work. [6]

7) P. Swathi, M. Harshitha, K. Sreekala and N. M. Sultana, "An Online Integration Platform for Student Projects", Zenodo (2024). A platform lets learners from separate institutions create project profiles, find collaborators, hold discussions, manage tasks and deadlines, share files and track versions. It supports teamwork across institutions, but it does not help a student discover whether a similar project already exists. [7]

8) Nachiket, A. Gumate, Md. Bilal, Sharanabasappa and P. Mutta, "Academic Task Manager – A Comprehensive Web-Based Platform for Academic Task Management, Student Progress Tracking, and Collaborative Learning", IJERT (2025). A web platform combines group discussions, assignment setting and marking, portfolios, certificates, event posting and student progress monitoring, built with HTML, CSS, JavaScript, PHP and MySQL. It manages coursework and progress, but offers no semantic search or assistant over earlier student projects. [8]

9) A. D. Dani and K. You, "Leveraging GitHub Classroom and Google Colab for Short-Term Student Machine Learning Team Projects", Journal of Practical Engineering Education (2025). GitHub Classroom hosts team repositories for status updates and submissions, and Google Colab provides GPUs for training, giving one centralised workflow for short machine learning projects. The workflow serves a single course and does not retain projects as a searchable knowledge base. [9]

10) P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", NeurIPS (2020). RAG combines a pre-trained generator with a non-parametric memory that is searched by a retriever, so that answers are based on retrieved documents and their source can be shown. It is the foundation of the ProjectHub assistant, but it was evaluated on open-domain tasks and not on an institutional archive of student projects. [10]

11) Y. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey", arXiv (2023). The survey organises RAG systems into retrieval, augmentation and generation steps and reviews their evaluation. It guides the design of the pipeline, but it does not address academic project portals. [11]

12) Z. Ji et al., "Survey of Hallucination in Natural Language Generation", ACM Computing Surveys (2023). The survey reviews how generated text can be unfaithful to its source or unsupported by facts. It motivates the rule that the ProjectHub assistant answers only from retrieved projects and says so when none is found. [12]

13) G. Salton, A. Wong and C. S. Yang, "A Vector Space Model for Automatic Indexing", Communications of the ACM (1975). Documents and queries are represented as vectors and ranked by their similarity. This is the basis of the similarity ranking used here, but the original model concerns document indexing and not a conversational assistant. [13]

III. RESEARCH GAP

The surveyed project-management works provide submission, teamwork or review, as summarised in Table I, and the RAG works [10]–[12] give the method but not the application. The following gaps remain.

A. Management without Reuse of Past Work

Existing platforms organise the current batch [1]–[9], but none of them describes a way for a student to ask whether a similar project has already been completed, so past work is stored but not reused.

B. Keyword Search Instead of Question Answering

Where an archive exists, it is browsed or filtered by fields such as title or year. A junior who describes an idea in ordinary words cannot obtain the closest earlier projects ranked by similarity.

C. Ungrounded Generated Answers

A general language model answers from its training data and may state projects that were never done in the college [12]. An assistant for this purpose must answer only from the stored projects.

D. Unsupported Claims of Novelty

A reply such as "nobody has done this" cannot be proved from a limited database. The assistant must report that no sufficiently similar project was found, not that the idea is new.

TABLE I.
CAPABILITIES OF RELATED WORK AND PROJECTHUB (YES = PROVIDED; PART. = PARTIALLY; – = NOT DESCRIBED)

Work	Project submission	Team collaboration	Faculty review	Project archive	Semantic search of past work	AI assistant
[1]	–	Yes	Part.	–	–	–
[2], [9]	–	Yes	–	–	–	–
[3]	–	Yes	–	Part.	–	–
[4]	–	Yes	Yes	–	–	–
[5]	Part.	Part.	Part.	–	–	–
[6]	–	Yes	–	Yes	–	–
[7]	Part.	Yes	–	Part.	–	–
[8]	Part.	Yes	Part.	Part.	–	–
ProjectHub	Yes	Yes	Yes	Yes	Yes	Yes

IV. PROBLEM STATEMENT

Colleges lack a unified system that (i) collects project submissions, team work and faculty review on one platform, (ii) keeps approved senior projects as a searchable knowledge base, (iii) lets a junior ask in natural language whether an idea has already been done and answers only from the stored projects, with the similarity of each match, and states plainly when nothing similar is found, and (iv) answers questions about the portal's own numbers exactly.

V. OBJECTIVES

1) To develop a web portal with role-based registration and login for students, faculty and administrators.
2) To let students submit projects with a description, a source-code archive and a deployed link, with a choice of who may see them.
3) To support team tasks with owners, due dates, status and a computed progress percentage, and a faculty review workflow with recorded comments.
4) To store senior projects as knowledge documents and convert each into a vector for similarity search.
5) To build a RAG assistant that retrieves the most similar senior projects and composes a grounded reply through the Groq API, without claiming that an idea is new.
6) To route statistics questions to exact database counts, and to define a reproducible evaluation.

VI. METHODOLOGY

A. System Architecture

ProjectHub has three tiers, as shown in Fig. 1. Students, faculty guides and administrators use a React web frontend with a dashboard, the project submission form, the Explore Projects page, the faculty review page and a floating AI Assistant button. The Spring Boot application server holds the modules for authentication and roles, project submission, team tasks and progress, faculty

review and notifications, and the AI orchestrator. A MySQL database stores the users, projects, teams, tasks, reviews, file records and the knowledge documents with their vectors. The orchestrator calls either a statistics service that reads exact counts, or the RAG retrieval component, which in turn calls the Groq API to compose the reply.

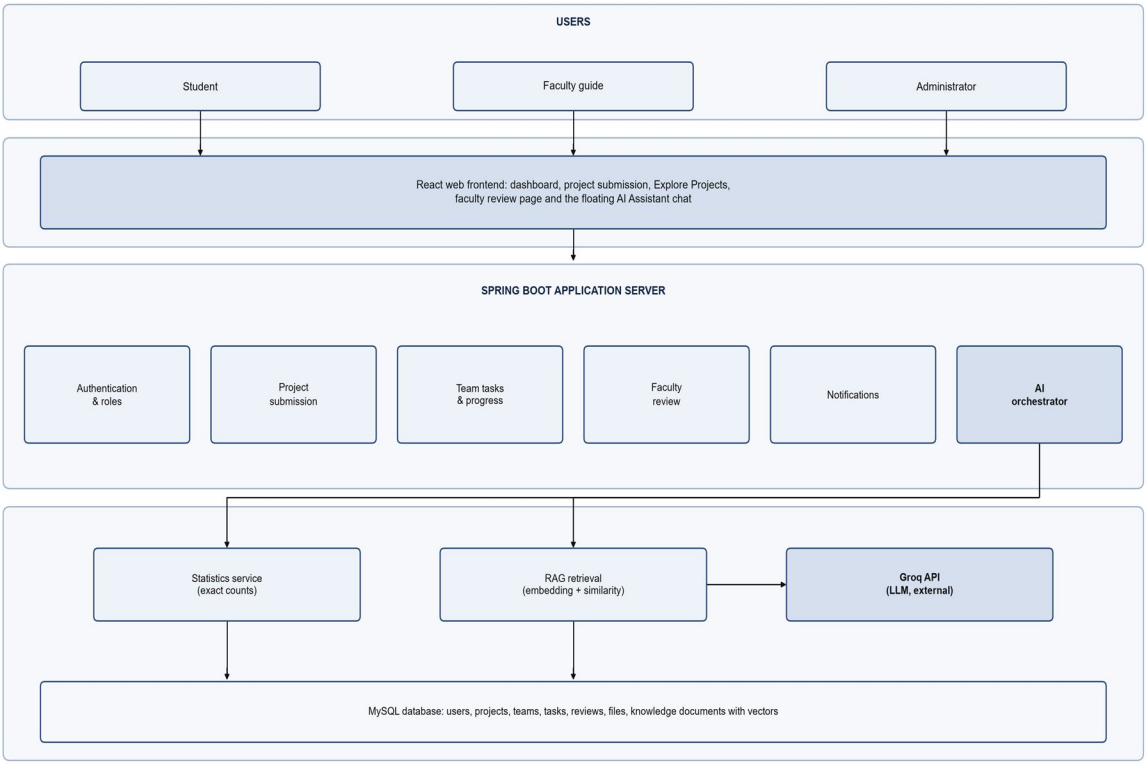


Fig. 1. Architecture of ProjectHub: users, React frontend, Spring Boot application server with the AI orchestrator, and the MySQL database.

B. Project Management Modules

Registration collects name, email, role and password; passwords are stored as bcrypt hashes and a login is accepted only when the entered password matches the stored hash. A student submits a project with a description of 10 to 4000 characters, an optional deployed link and a source-code archive, and chooses whether the code and the link are visible to faculty only or to all students. A submitted project appears on the student's dashboard and in the list of the guide. Team members divide a project into tasks with owners, priorities and due dates, and each task moves through To Do, In Progress and Completed. The progress of a team is

$$Progress\ (\%) = (Completed\ tasks\ /\ Total\ tasks) \times 100 \quad (1)$$

A guide sees only the projects assigned to them, may approve, reject or request changes, and must enter a comment with every decision; the decision, comment, reviewer and date are stored as a review history. Table II lists the submission and storage rules, which are enforced on the server and not only in the interface.

TABLE II.
SUBMISSION AND STORAGE RULES

Item	Rule
Description	10 to 4000 characters
Source-code archive	Up to 50 MB; .zip, .tar, .tar.gz, .gz or .7z only
File name	Directory path removed to prevent path traversal; saved under a unique name
Visibility	Source code and deployed link: faculty only or all students
Editing	A student cannot edit another student's project
Download	Faculty-only code cannot be downloaded by students; faculty can always download

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

C. Database and Knowledge Documents

The relational database holds the tables in Table III, linked by primary and foreign keys. The senior projects that the assistant searches are kept as knowledge documents, separate from the program code, in a fixed text format: each project has a title, batch, authors, description and technologies, and projects are separated by a line of three hyphens. New projects can therefore be added to the knowledge base without changing the application.

TABLE III.
PRINCIPAL DATABASE TABLES

Table	Principal fields
users	name, email, department, role, password hash
projects	title, description, source-code file, deployed link, visibility, status, submission date, guide
teams, tasks	members and roles; task owner, priority, due date, status
reviews	decision, comment, reviewer, date
knowledge documents	title, batch, authors, description, technologies, embedding vector (JSON text)

D. RAG Pipeline

The pipeline has an indexing step and a question-answering step, as shown in Fig. 2. When a knowledge document is added, the embedding service lower-cases its text, replaces punctuation with spaces, discards one-letter words and a list of common stop words, and converts the remaining words into a vector of 512 dimensions, which is stored as JSON text and read back when needed (Table IV). The embedding is computed inside the application and needs no external service.

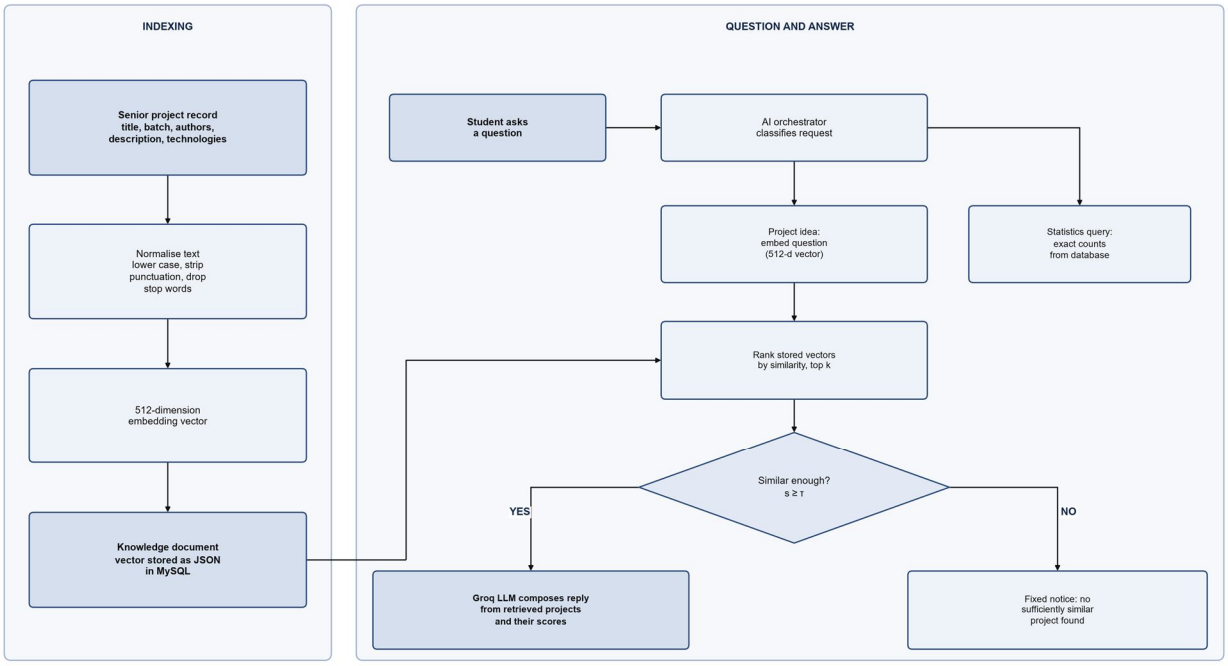


Fig. 2. RAG workflow of the Project Assistant: indexing of senior projects (left) and question answering with the similarity check (right).

TABLE IV.
CONFIGURATION OF THE EMBEDDING SERVICE

Parameter	Value
Vector dimension	512
Text normalisation	Lower case; characters other than letters, digits and spaces removed
Tokens kept	Words longer than one letter that are not in the stop-word list
Empty text	Zero vector
Storage	Vector serialised as JSON text with the knowledge document

A question is converted into a vector q in the same way. With d_i the stored vector of project i, the similarity of the question to that project is

$$s_i = q \cdot d_i / (|q|\,|d_i|) \quad (2)$$

The projects are ranked by s_i and the best k are examined. Only those whose similarity reaches the threshold τ are retained:

$$R = \{\, i \in \text{top-}k : s_i \geq \tau \,\}, \quad reply = LLM(question, R) \text{ if } R \neq \emptyset, \text{ else fixed notice} \quad (3)$$

If R is not empty, the question and the retained projects, each with title, batch, authors and similarity score, are given to a large language model served through the Groq API [14], which writes the reply from them. If R is empty, the assistant states that it could not find a sufficiently similar project in the current senior-project database and does not claim that nobody has done the idea before. Retrieval and the knowledge documents are therefore the only source of project facts in a reply, which addresses the hallucination risk noted in [12].

E. AI Orchestration

The AI orchestrator receives each question together with the identity of the logged-in user and decides how to answer it. A question about numbers, such as “How many projects have been uploaded?”, is routed to the statistics service, which reads exact counts from the database and is returned as a STATISTICS reply. A question about a project idea is routed to the RAG component and returned as a RAG_SEMANTIC reply. Keeping the two routes separate means that numerical answers are exact and project-similarity answers are grounded in the stored projects.

VII. RESULTS AND DISCUSSION

A. Prototype Implementation

The prototype was exercised through its web interface. Fig. 3 shows the Project Assistant on the student dashboard, where a question about a “college related” project is answered with the closest retrieved project, its batch, authors and its low similarity score (0.16), and the statement that no sufficiently similar project was found; and the Explore Projects page. Table V lists the functions that were exercised and the behaviour observed.

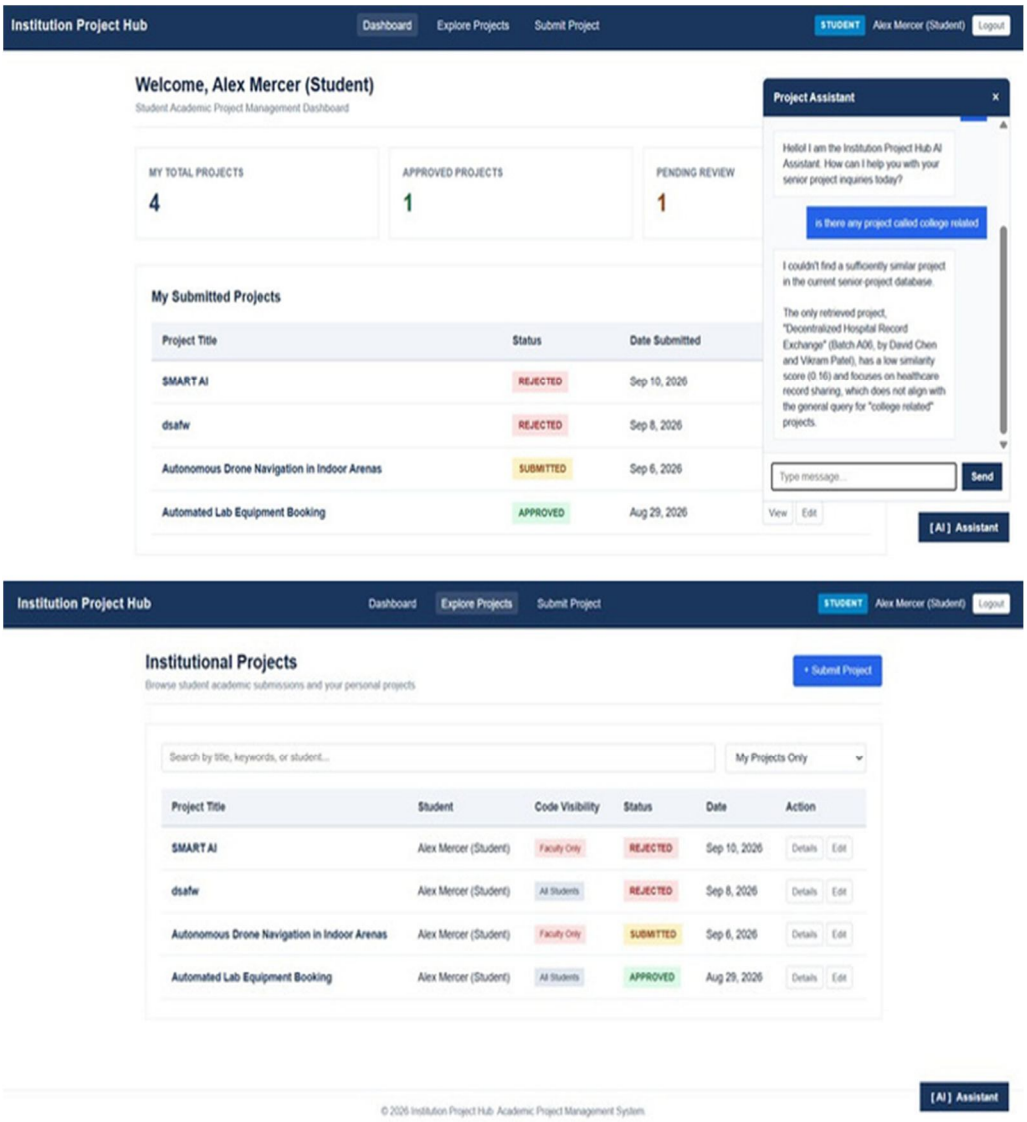


Fig. 3. Prototype screens: student dashboard with the AI Project Assistant, and the Explore Projects page.

TABLE V.
FUNCTIONS EXERCISED ON THE PROTOTYPE

Function	Observed behaviour
Student dashboard	Total, approved and pending projects; table of own projects with status, date, view and edit
Explore Projects	Institutional projects listed with student, code visibility, status and date; searchable
Project submission	Title, description, source-code archive, optional link and visibility; stored as Submitted
Faculty page	Student details, code visibility and status; filter by name, student or status; review and status
Project Assistant	Closest senior project with similarity score, or a notice that no similar project was found
Statistics query	Exact counts read from the database

B. Automated Verification

The security and grounding rules are covered by JUnit tests, listed in Table VI. The last test deletes all knowledge documents, asks whether a submarine laser communication system has been done by seniors, and checks that the reply contains the notice that no sufficiently similar project was found and does not say that no one in the college has ever done it.

TABLE VI.
AUTOMATED TESTS OF THE PROTOTYPE

Test	Rule checked
Project ownership	A student cannot update another student's project
Faculty-only code	Another student cannot download it; faculty can
Statistics	Counts of projects and students are read from the database; "How many projects have been uploaded?" is routed as STATISTICS
Grounding	With no knowledge documents, the RAG_SEMANTIC reply states that no similar project was found and never claims novelty

C. Evaluation Plan

Because the project is at the prototype stage, a formal protocol is defined rather than reporting unmeasured figures. A set Q of questions will be written by seniors and juniors, each with the projects of the knowledge base that a reviewer judges to be relevant. Retrieval quality will be measured with

$$Recall@k = |\{ q \in Q : a\ relevant\ project\ is\ in\ the\ top\ k \}| / |Q|, \quad MRR = (1 / |Q|)\ \Sigma\ 1 / rank_q \tag{4}$$

where $rank_q$ is the position of the first relevant project for question q. The threshold τ will be chosen on a validation subset of questions, including questions with no relevant project, to balance missed matches against false matches. Table VII lists the planned measurements.

TABLE VII.
PLANNED EVALUATION METRICS

Aspect	Metric	Method
Retrieval	Recall@k, MRR	Labelled question set
Threshold τ	False-match and missed-match rate	Questions with and without a relevant project
Grounding	Replies naming a project not retrieved	Manual check of replies
Latency	Time per question	Question to reply, including the Groq call
Usefulness	Topic changes and extensions by juniors	Survey of a junior batch

D. Discussion

The prototype shows that submission, review and archiving can feed a single knowledge base, and that a junior can ask in ordinary words what has been done before and receive the closest projects with their scores. Three design choices make the assistant cautious: replies are composed only from retrieved projects, an explicit threshold separates a match from a non-match, and the no-match reply never claims that an idea is new. Statistics are answered from the database, not from the language model.

The evaluation is limited to the prototype: retrieval accuracy, the best threshold and the usefulness to juniors are not yet measured. Because the vector is built from the words of the text, a project described with different words may score low, so a low score does not prove that no similar project exists. The reply is only as complete as the knowledge base, and each question and the retrieved project text are sent to an external service, which an institution must consider before adoption. The assistant helps a student to see earlier work; it does not judge the quality of a project.

VIII. CONCLUSION AND FUTURE SCOPE

ProjectHub combines project submission, team collaboration, faculty review and a searchable repository with a RAG assistant that retrieves similar senior projects and composes a grounded reply through the Groq API, so that juniors can see what has already been done and build on it instead of repeating it. The assistant refuses to claim that an idea is new when no sufficiently similar project is found, and statistics are answered exactly from the database. Future work includes the quantitative evaluation described above, automatic detection of similar topics when a project is submitted, suggestion of domains and summaries of abstracts and reports, embeddings that match meaning rather than words, GitHub or GitLab integration, a mobile application with push notifications, plagiarism checking and cloud deployment.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

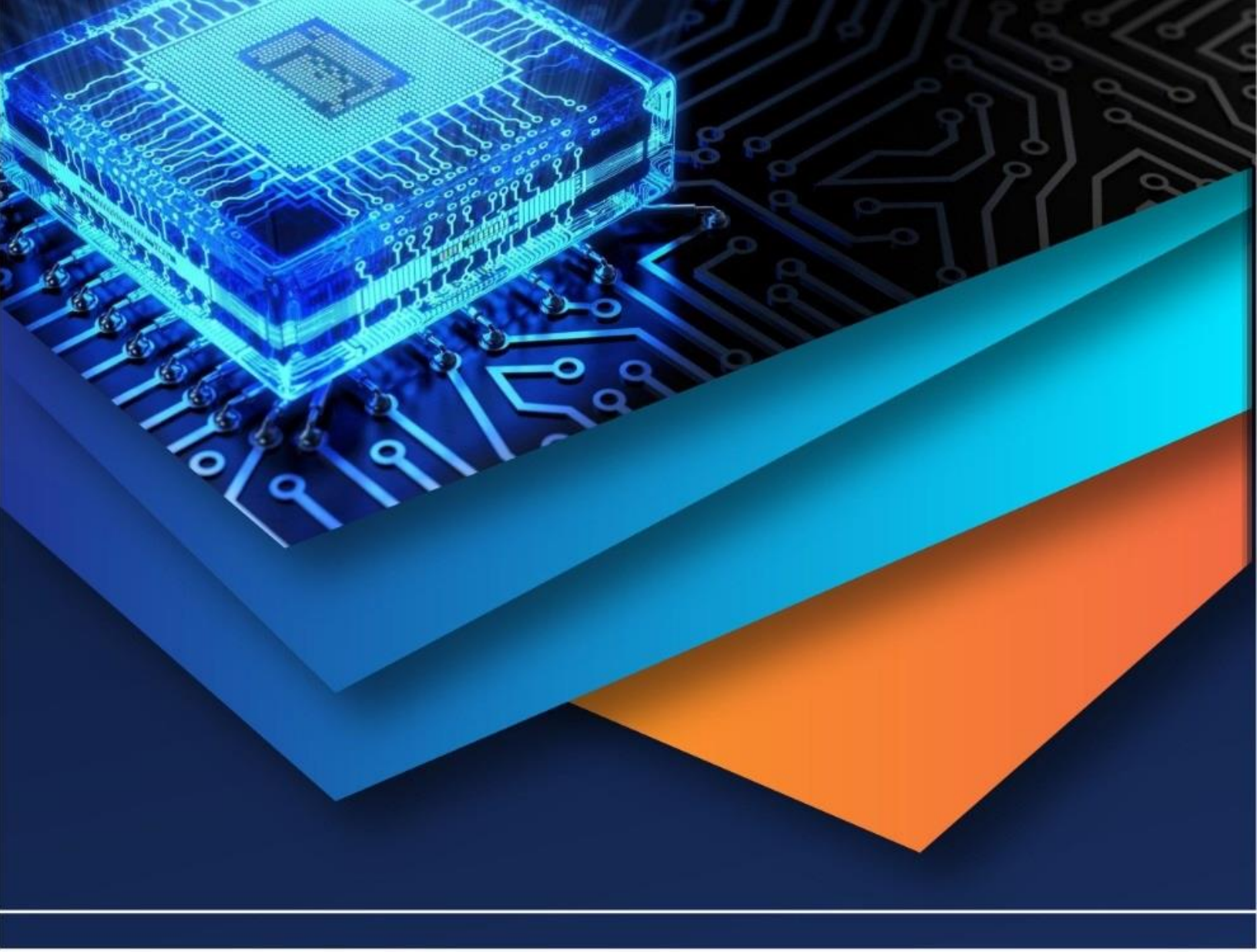
IX. ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, for institutional support.

REFERENCES

[1] B. Anbarolu, “A collaborative GIS programming course using GitHub Classroom,” Trans. GIS, vol. 25, no. 6, pp. 3132–3158, 2021.
[2] Cs. Z. Kertész, “Using GitHub in the classroom – A collaborative learning experience,” in Proc. 21st Int. Symp. Design Technol. Electron. Packag. (SIITME), 2015, pp. 381–386.
[3] M. A. Nelson and L. Ponciano, “Experiences and insights from using GitHub Classroom to support project-based courses,” in Proc. 3rd Int. Workshop Softw. Eng. Educ. Next Gener. (SEENG), 2021, pp. 31–35.
[4] “CapStone: A cloud-based platform for multi-party collaboration on capstone projects,” in Proc. IEEE Frontiers Educ. Conf. (FIE), 2016.
[5] S. Varikuti, “A web-based online collaboration platform for formulating engineering design projects,” M.S. thesis, Purdue Univ., 2014.
[6] “Scholarize: A digital ecosystem for enhanced research and collaboration,” in Proc. 4th Int. Conf. Inf. Syst. Higher Educ. (ISHE), 2025.
[7] P. Swathi, M. Harshitha, K. Sreekala, and N. M. Sultana, “An online integration platform for student projects,” Zenodo, 2024. [Online]. Available: https://zenodo.org/records/12567503
[8] Nachiket, A. Gumate, Md. Bilal, Sharanabasappa, and P. Mutta, “Academic Task Manager – A comprehensive web-based platform for academic task management, student progress tracking, and collaborative learning,” Int. J. Eng. Res. Technol., vol. 14, no. 12, 2025, doi: 10.17577/IJERTV14IS120256.
[9] A. D. Dani and K. You, “Leveraging GitHub Classroom and Google Colab for short-term student machine learning team projects,” J. Pract. Eng. Educ., vol. 17, no. 1, pp. 49–61, 2025.
[10] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 33, 2020, pp. 9459–9474.
[11] Y. Gao et al., “Retrieval-augmented generation for large language models: A survey,” arXiv:2312.10997, 2023.
[12] Z. Ji et al., “Survey of hallucination in natural language generation,” ACM Comput. Surv., vol. 55, no. 12, Art. 248, 2023.
[13] G. Salton, A. Wong, and C. S. Yang, “A vector space model for automatic indexing,” Commun. ACM, vol. 18, no. 11, pp. 613–620, Nov. 1975.
[14] GroqCloud, “GroqCloud developer documentation.” [Online]. Available: https://console.groq.com/docs

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)