

HackHub: A Role-Based Web Platform for Project Selection, Abstract Submission and Automated Abstract Grading Using Natural Language Processing

Atmakuri Tagore Nandan¹, CH Pravalika²

¹B.Tech Student, Dept. of Computer Science and Engineering, Hyderabad Institute of Technology and Management (HITAM), Hyderabad, Telangana, India

²Assistant Professor, Dept. of Computer Science and Engineering, Hyderabad Institute of Technology and Management (HITAM), Hyderabad, Telangana, India

Abstract - Final-year engineering students often choose mini and major projects without clear awareness of industry trends, while mentors struggle to guide and evaluate large batches manually. This paper presents HackHub, a role-based web platform that digitizes an academic department's project workflow. Mentors publish problem statements with required skills and seat limits; students browse and select them on a first-come, first-served basis; and abstracts are submitted as PDF files. An automated grading engine built with FastAPI, NLTK, and pre-trained Word2Vec embeddings extracts the most frequent keywords from each abstract, measures their cosine similarity with mentor-defined keywords, and converts the result into a percentage grade after verifying that the abstract contains the project title. The front end is developed with Next.js, Tailwind CSS, and ShadCN UI, while Firebase Authentication, Firestore, and Cloud Storage provide secure sign-in, data management, and file storage. Six functional test cases covering registration, problem statement upload, first-come, first-served allocation, abstract upload, automated grading, and role-based access were executed, and all passed. The platform reduces manual effort, improves transparency in evaluation, and can be extended to hackathons and industry-sponsored problem statements.

Key Words: Project Selection, Automated Abstract Grading, Natural Language Processing, Word Embeddings, Cosine Similarity, FastAPI, Next.js, Firebase.

1. INTRODUCTION

Mini and major projects are a crucial part of the final year in an engineering degree. These projects allow students to apply theoretical knowledge to real-world problems and develop skills in problem-solving, teamwork, and technical expertise. However, many students find it difficult to choose meaningful projects. Three major reasons are common. First, students have limited awareness of current trends and emerging technologies. Second, mentors often lack the time to offer personalized guidance to a large number of students. Third, project selection and evaluation are

generally done manually, which is time-consuming and may be influenced by personal bias. As a result, students often end up with projects that offer limited learning value, reducing their engagement and practical experience.

1.1 Background and Automated Assessment

Automated assessment of written texts has been studied for many years, from early statistical scoring methods [1] to modern systems based on natural language processing and deep learning [2]. These tools are commonly used for grading essays and programming assignments but are rarely integrated into the administrative processes of academic project management. This paper presents HackHub, a web platform that combines problem statement publishing, first-come-first-served project selection, PDF-based abstract submission, and automated abstract grading into a single system tailored for students, mentors, and administrators.

1.2 Objectives

- Create a centralized platform where mentors can upload problem statements along with descriptions, required skills, and seating limits.
- Allow students to select projects using a first-come-first-served (FCFS) approach.
- Accept abstracts in PDF format and automatically grade them based on mentor-defined keywords.
- Implement role-based access control for students, mentors, and administrators.
- Design the system to be adaptable for using in hackathons and industry-sponsored projects.

1.3 Contributions

The main contributions of this work are:

- i. A complete platform covering the entire project selection process.

- ii. A lightweight, explainable grading engine that scores abstracts based on the semantic similarity between the keywords in the abstract and those specified by the mentor.
- iii. A functional prototype that has been validated through testing.

2. LITERATURE REVIEW

2.1 Existing Project Allocation and Evaluation Practices

In traditional academic settings, project selection is usually managed as students approach mentors individually. While this encourages direct communication, it does not scale well and lacks standardization, often leading to dissatisfaction when evaluations are subjective. Open-source platforms like GitHub and SourceForge offer many project ideas, but they were not designed for academic use and do not provide mentorship, structured evaluation workflows, or learning support. Hackathon platforms such as Devpost and HackerEarth support competitive submissions, but they focus on short-term events rather than structured learning. Grading and feedback tools such as Turnitin, CodePost, and Edmodo are used to handle essays, code, and quizzes. However, these tools are not designed to assess project abstracts.

2.2 Word Embeddings and Semantic Similarity

The grading engine in HackHub uses distributed word representations. Mikolov et al. [3] showed that neural word embeddings trained on large corpora place semantically related words close together in vector space, which makes them suitable for measuring word similarity. The closeness of two vectors is commonly measured using cosine similarity, a standard measure in information retrieval [4] that has also been applied to semantic text matching [5]. Practical text processing is supported by the NLTK toolkit [6] for tokenization and stop-word filtering, and by the Gensim library [7] for loading pre-trained vectors.

2.3 Research Gap

From this survey, the following gaps were identified: there is no centralized repository of mentor-curated academic problem statements; manual grading is subjective and time-consuming; existing automated tools are not flexible enough for project-specific evaluation; and conventional processes do not scale to large batches or events. HackHub is designed to address these gaps.

3. METHODOLOGY

3.1 Existing System

In the existing process, students approach faculty manually for guidance, which creates communication gaps. Abstract grading is inconsistent and prone to personal bias. Students lack resources aligned with their chosen projects, and the process does not scale to large batches.

3.2 Proposed System

The proposed system offers a centralized problem-statement repository, role-based access for students, mentors, and administrators, FCFS project selection with seat limits, PDF abstract submission, and an automated grading service that returns a percentage grade to the mentor dashboard.

3.3 Functional Requirements

- **Students** register, log in, create or join a team, select problem statements, upload abstracts, and view results.
- **Mentors** upload problem statements (title, description, required skills, maximum students), view applicants, download abstracts, and review automated grades.
- **Administrators** manage user roles and oversee platform usage and project assignments.
- **Abstract handling:** abstracts are accepted only in PDF format, stored in Firebase Storage and linked to the user record in Firestore.

3.4 Non-Functional Requirements

Scalability: the platform should serve many simultaneous users and submissions. Security: authentication and data storage rely on Firebase Authentication and Firestore. Performance: response time for authentication and data retrieval should not exceed two seconds. Usability: interfaces must be intuitive for all three roles.

3.5 System Architecture

HackHub follows a three-layer architecture (Fig -1). The **frontend layer** is a Next.js web application styled with Tailwind CSS and ShadCN UI components, with separate dashboards for each role. The **backend layer** is a FastAPI service that manages business logic, project assignment, abstract storage, and grading requests. The **data and services layer** consists of Firebase Firestore for structured data, Firebase Storage for PDF files, and Firebase Authentication for sign-in and role

management, together with the natural language processing grading engine hosted on Google Cloud Platform (GCP).

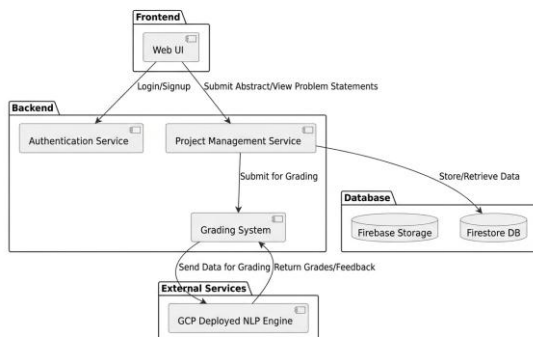


Fig -1: System architecture of HackHub

3.6 Technology Stack

Table -1: Technology stack of HackHub

Layer	Technology
Frontend	Next.js, Tailwind CSS, ShadCN UI
Backend	FastAPI (Python)
Authentication	NextAuth with Firebase Authentication
Database and files	Firebase Firestore, Firebase Storage
Grading engine	NLTK, pdfplumber, Gensim (Word2Vec), NumPy; hosted on GCP
Hosting	Vercel (frontend)

3.7 Abstract Submission Workflow

The abstract submission workflow is shown in Fig -2. After a student submits an abstract, it is saved and the mentor is notified. If the abstract is relevant, the problem statement is allocated, the grading system assigns a grade, feedback is generated, and the student is informed. Otherwise, the student receives rejection feedback. The UML design also includes a notification module and a resource recommendation module. The current prototype implements the submission, storage, and grading path, while notifications and resource recommendations are planned extensions.

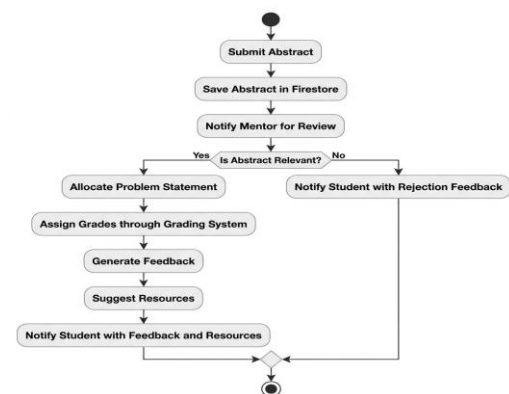


Fig -2: Abstract submission and feedback workflow

3.8 Implementation

3.8.1 Frontend

The frontend was created with Next.js [8]. Tailwind CSS provides utility-based styling, and ShadCN UI provides reusable components such as buttons, inputs, dialogs, and tables. The interface includes a sign-up form with role selection, a login screen, role-specific dashboards, a problem statement management page for mentors, and a problem statement list with PDF upload for students.

3.8.2 Database and Authentication

Firebase [9] provides the data layer. Firestore stores user profiles, problem statements, team details, and submission metadata. Firebase Storage stores the uploaded PDFs and returns download URLs, and storage rules restrict uploads and downloads to authenticated users. Firebase Authentication supports email and password as well as other providers, and the issued tokens are used to validate user actions in the backend through the Firebase Admin SDK.

3.9 Automated Grading Engine

The grading engine is a FastAPI [10] service exposing a POST endpoint that receives a PDF, a comma-separated list of mentor keywords and, optionally, the project title. It proceeds as follows:

- **Validation:** the file must be a PDF and at least one keyword must be supplied; otherwise an error is returned.
- **Text extraction:** pdfplumber reads the text of every page, and pages without text are logged.
- **Title check:** if a title is supplied and does not appear in the extracted text, the grade is 0.
- **Keyword extraction:** the text is lower-cased and tokenized with NLTK, stop words and non-alphanumeric tokens are removed, and the 100

most frequent words are retained with their counts.

- **Semantic correlation:** each mentor keyword and each extracted keyword is mapped to a 300-dimensional vector from the pre-trained word2vec-google-news-300 model, and their cosine similarity is computed. Words missing from the vocabulary are skipped.

For two word vectors u and v , the similarity is

$$\text{sim}(u, v) = (u \cdot v) / (\|u\| \|v\|) \quad (1)$$

If C is the set of all valid (mentor keyword, extracted keyword) pairs, the grade returned to the mentor dashboard is

$$\text{Grade (\%)} = (100 / |C|) \times \sum \text{sim}(u, v) \quad (2)$$

The grade is 0 when C is empty. The response also contains the extracted keywords, their counts, and all pairwise correlations, so the mentor can see why a grade was assigned.

4. RESULTS

The system was tested with unit, integration, and functional testing. Six functional test cases were designed around the main workflows, and all of them passed (Table -2).

Table -2: Functional test cases and outcomes

ID	Test	Expected result	Status
TC1	Register with valid name, email, password and role	User created and stored in Firestore	Pass
TC2	Mentor uploads a problem statement	Saved in Firestore and visible to students	Pass
TC3	Student selects a project (FCFS)	Project assigned and marked unavailable to others	Pass
TC4	Student uploads an abstract PDF	File stored in Firebase Storage, metadata in Firestore	Pass
TC5	Grading of an uploaded abstract	Numeric grade returned	Pass
TC6	Login with student, mentor and admin accounts	Each role sees only its permitted features	Pass

5. DISCUSSION

5.1 Interpretation of the Grade

The prototype shows that the complete project workflow can be handled in one platform, and that a keyword-similarity grade can be produced without manual reading. The grade should be interpreted carefully. Because it is the mean similarity between mentor keywords and the 100 most frequent words, it measures how closely an abstract stays on the mentor-defined topic rather than creativity or feasibility, and values close to 100% are unlikely.

5.2 Limitations

The grades have not yet been compared with mentor-awarded marks, so a correlation study on a larger set of abstracts is the most important next step. Word2vec vectors also cover single words only, and scanned PDFs without a text layer cannot be graded.

6. CONCLUSION

This paper presented HackHub, a role-based web platform that simplifies project selection, abstract submission, and initial evaluation for engineering students and mentors. It combines a Next.js frontend, a FastAPI backend, Firebase services, and an NLP-based grading engine that uses pre-trained word embeddings and cosine similarity. Functional testing of all six core workflows passed. The platform reduces manual coordination, makes evaluation more transparent, and provides a base that can be extended to hackathons and industry collaboration once the grading model is validated against mentor evaluations.

6.1 Future Scope

Planned enhancements include: mentor-configurable grading metrics; sentence-embedding models for richer comparison; plagiarism detection; a resource recommendation module and e-mail or in-app notifications; AI-based project suggestions from a student profile; multilingual and voice-based accessibility; gamification with badges and leaderboards; industry integration; and a mobile application.

REFERENCES

- 1) E. B. Page, "The imminence of grading essays by computer," Phi Delta Kappan, vol. 47, no. 5, pp. 238-243, 1966.
- 2) J. Ke and H. Ng, "Automated essay scoring: A survey of the state of the art," in Proc. 27th Int. Conf. on Computational Linguistics, 2019, pp. 623-632.

- 3) T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781, 2013.
- 4) C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge, U.K.: Cambridge Univ. Press, 2008.
- 5) F. Rahutomo, T. Kitasuka, and M. Aritsugi, "Semantic cosine similarity," in Proc. 7th Int. Student Conf. on Advanced Science and Technology (ICAST), Seoul, South Korea, 2012.
- 6) S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly Media, 2009.
- 7) R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in Proc. LREC Workshop on New Challenges for NLP Frameworks, Valletta, Malta, 2010, pp. 45–50.
- 8) Next.js Documentation. [Online]. Available: <https://nextjs.org/docs>
- 9) Firebase Documentation. [Online]. Available: <https://firebase.google.com/docs>
- 10) FastAPI Documentation. [Online]. Available: <https://fastapi.tiangolo.com>