Event Aggregator For College Students: A Unified Web-Based Platform For Centralizing Career And Academic Opportunities

Prathamesh Shinde[1], Omkar Phapale[2], Saurabh Kanade[3], Sahil Zaware[4]

Abstract- *In the current digital era, college students frequently miss internships, hackathons, job openings, and coding contests because relevant information is fragmented across dozens of platforms, social media groups, and institutional portals. This paper presents the design and development of an Event Aggregator for College Students — a centralized, rolebased web application that aggregates, verifies, and delivers academic and professional opportunities through three dedicated portals: the Student Portal, the Company Portal, and the Admin Portal. The Student Portal enables profile creation, skillbased filtering, application tracking, and personalized deadline reminders. The Company Portal allows verified organizations to post and manage opportunities and evaluate applicants. The Admin Portal ensures platform integrity through opportunity moderation and company verification. The system is built using React.js for the frontend, Java Spring Boot or Node.js for the backend, and MySQL/MongoDB as the database. The Agile SDLC model guides iterative development across sprints. This paper details the system's motivation, related work, software requirements, architecture, data flow, risk management, and future directions including AI-based recommendations and automated resume screening.*

Keywords: Event aggregation, opportunity management, college student portal, web-based platform, job recommendation, hackathon, internship, React.js, Spring Boot, Agile SDLC

I. INTRODUCTION

1.1 Background

The transition from academic education to professional employment represents one of the most significant milestones in a student's career journey. In today's competitive and technology-driven world, academic knowledge alone is insufficient for achieving employability and professional success. Students are increasingly expected to possess practical exposure, technical competencies, communication skills, and industry awareness before graduation. Consequently, opportunities such as internships, hackathons, coding competitions, workshops, research programs, campus placement drives, certification courses, and part-time jobs play a crucial role in enhancing students' practical knowledge and preparing them for real-world challenges.

However, despite the growing importance of such opportunities, students face substantial difficulties in discovering and accessing them efficiently. Information regarding internships, job openings, technical events, and placement drives is highly fragmented across multiple platforms including corporate career portals, LinkedIn, Internshala, HackerEarth, Unstop, college notice boards, Telegram and WhatsApp groups, emails, and informal peer networks [1]. As a result, students are often required to spend considerable time searching, validating, and organizing information from numerous unreliable or inconsistent sources. This fragmented ecosystem creates significant information overload and reduces the efficiency with which students can identify opportunities relevant to their interests and qualifications.

The lack of centralized access further contributes to unequal opportunity distribution among students. Students studying in premier institutions typically benefit from well established placement cells, active alumni networks, and stronger industry connections, allowing them to gain early access to high-quality opportunities. In contrast, students from smaller, rural, or newly established colleges frequently lack such support systems, resulting in reduced awareness and limited participation in valuable professional activities. This imbalance creates an information asymmetry where opportunities become concentrated among students with stronger institutional or social networks, thereby widening the employability gap.

Another major challenge lies in the absence of personalization in existing opportunity platforms. Most currently available systems present opportunities in a generalized manner without considering a student's academic background, technical skills, career interests, preferred domains, or eligibility criteria. Consequently, students are

forced to manually filter through large volumes of irrelevant listings, which consumes time and increases the likelihood of missing important deadlines. Additionally, many platforms suffer from problems such as duplicate postings, spam advertisements, fraudulent opportunities, and outdated information, reducing user trust and platform reliability.

From an organizational perspective, companies and recruiters also encounter operational inefficiencies during the recruitment process. Employers often need to publish the same job or internship opportunity across multiple portals to maximize visibility. Furthermore, the absence of structured applicant filtering mechanisms leads to large volumes of unqualified applications, increasing the burden on recruitment teams and delaying candidate selection processes. Many recruiters also struggle to identify students possessing the precise technical and soft skills required for specific roles.

Similarly, educational institutions and college administrators lack analytical visibility into student engagement with external opportunities. Placement cells frequently face challenges in tracking how many students apply for internships, participate in hackathons, or develop in-demand industry skills. Without centralized monitoring systems, colleges cannot effectively identify skill gaps, measure placement preparedness, or design targeted training initiatives for students.

To address these challenges, this research proposes the design and development of a comprehensive Event Aggregator for College Students, a centralized web-based platform that acts as a unified hub for academic, technical, and professional opportunities. The proposed system aims to streamline opportunity discovery by integrating multiple categories of events and career resources into a single platform accessible to all students regardless of institutional background.

The proposed platform introduces a multi-role architecture consisting of three dedicated user portals:

1. Student Portal– Enables students to discover, filter, bookmark, and apply for opportunities based on their skills, interests, academic profile, and career goals. Students can also receive personalized notifications and track their application status in real time.
2. Company Portal– Allows organizations and recruiters to post internships, jobs, hackathons, workshops, and recruitment drives while managing applications through a structured and centralized interface.
3. Admin Portal– Provides administrative oversight for validating opportunity listings, preventing spam or fraudulent postings, managing users, and maintaining overall platform integrity and data quality.

A key feature of the proposed system is its verification and moderation workflow, which ensures that all published opportunities are authenticated before becoming visible to students. This mechanism enhances trustworthiness and reduces misinformation. In addition, the platform incorporates skillbased recommendation and filtering systems, enabling students to efficiently identify opportunities relevant to their technical expertise, preferred domains, and eligibility criteria.

The platform also includes a notification and alert infrastructure that informs users about upcoming deadlines, newly posted opportunities, interview schedules, and application updates. Such automation minimizes the risk of missed opportunities and improves user engagement. Furthermore, the centralized database architecture enables educational institutions to generate analytical insights regarding student participation, skill trends, and placement readiness.

The proposed Event Aggregator system contributes to solving multiple practical challenges simultaneously: it reduces information fragmentation, democratizes access to opportunities, minimizes manual effort for students and recruiters, improves transparency, and supports data-driven decision-making within educational institutions. By leveraging modern web technologies, database management systems, and scalable software architecture principles, the system seeks to create a reliable and efficient ecosystem for connecting students with academic and professional growth opportunities.

In broader terms, the platform aligns with the ongoing digital transformation in higher education and recruitment processes. As educational institutions increasingly adopt technology-driven solutions for student development and employability enhancement, centralized opportunity aggregation systems can significantly improve accessibility, inclusiveness, and operational efficiency. The proposed system therefore has the potential to become an essential component of modern academic and career support infrastructure.

The remainder of this research paper is organized as follows: Section II presents a review of related literature and existing systems; Section III discusses the software and hardware requirements of the proposed platform; Section IV explains the overall system architecture, database design, and implementation methodology; Section V outlines the project planning and development lifecycle; Section VI highlights the

advantages, limitations, and practical applications of the system; and finally, Section VII concludes the paper and suggests possible future enhancements and research directions.

II. LITERATURE SURVEY

The design of the proposed system draws on multiple bodies of research spanning job recommendation systems, skill extraction, information retrieval, and hackathon management.

Ertugrul et al. [2] conducted a comprehensive systematic literature review of job recommender systems from the past decade, categorizing hybrid, content-based, and collaborative approaches alongside their evaluation metrics and known research gaps. Their work highlights the growing role of machine learning in matching candidates to positions — a dimension we plan to incorporate in future iterations of our platform.

Senger et al. [3] surveyed deep-learning and traditional NLP techniques for extracting and classifying skills from job descriptions, discussing challenges in ontology mapping. This work directly informs the skill-tagging mechanism in our Company Portal, where job descriptions are structured for accurate filtering.

Akkasi et al. [4] proposed transformer-based models with explainability components to parse job descriptions into structured fields (skills, role, seniority). Their approach demonstrates the feasibility of automated opportunity parsing, which we plan to integrate for auto-categorization in future releases.

Singh [5] provided a broad overview of informationextraction (IE) techniques from unstructured text, including rule-based and statistical/ML models. This provides the theoretical foundation for our web-scraping and opportunity aggregation modules.

Mhamdi et al. [6] presented a recommender system where job offers are clustered and matched to candidate profiles using similarity measures — a technique that influences our personalized recommendation design. Brambilla et al. [7] analyzed hackathons from an organizer and innovation perspective, including metadata structures and evaluation criteria, which directly informed our hackathon opportunity schema.

Soni and Shah [8] described a hybrid recommendation architecture combining BERT embeddings, knowledge-graph reasoning, and reinforcement learning for adaptive personalization — a roadmap for our planned AI-based recommendation module. Kumar et al. [9] raised critical concerns about fairness and legal compliance in recruitment recommenders, which influenced our decision to implement transparent admin moderation rather than purely automated filtering.

Wang et al. [10] demonstrated gains in match relevance through personalized recommendation algorithms tailored to student profiles, reinforcing the importance of profile completeness in our Student Portal design.

III. SOFTWARE REQUIREMENT SPECIFICATION

Three primary user classes interact with the system:

- Student Users: Primary consumers of the platform.Typically undergraduate or postgraduate students with varying technical familiarity, requiring a simple and intuitive interface.
- Company/Recruiter Users: Verified representatives of or-ganizations who post and manage opportunities and evaluate applicant profiles.
- Admin Users: Super-users with full control over platformintegrity, responsible for verifying companies, approving posts, and moderating content.

C. Functional Requirements

The core functional requirements, organized by system feature, are as follows:

User Registration and Authentication: The system shall allow students and companies to register with basic credentials and role-based details. Role-based login shall redirect users to their respective dashboards. Password management, account verification for companies, and session security are enforced.

Student Portal: Students can browse and filter all verified opportunities by domain, location, skills, deadline, and type. They can apply directly through the portal, upload resumes, track application status (Applied → Reviewed → Shortlisted → Selected/Rejected), bookmark opportunities, and receive personalized deadline reminders.

Company Portal: Verified companies can create, edit, and delete opportunity listings. They can view and manage submitted applications, shortlist candidates, download resumes, and update applicant status.

Admin Portal: Admins can verify and approve company registrations, review and publish opportunity posts, remove duplicates or spam, and monitor overall platform activity.

D. Non-Functional Requirements

- Performance: The system shall support concurrent accessby multiple users with response times under 3 seconds for standard queries.
- Security: All data transmission occurs over HTTPS.Role-based access control prevents unauthorized operations.
  Sensitive credentials are stored using industry-standard hashing.
- Scalability: The architecture shall support horizontalscaling on cloud infrastructure to handle growing user bases.
- Availability: Target uptime of 99.5% with fail-safedatabase backups.
- Usability: The UI shall be responsive and accessible ondesktops, tablets, and smartphones.

E. System and Hardware Requirements

Client-side requirements include a dual-core processor, minimum 2 GB RAM (4 GB recommended), and any modern browser (Chrome, Firefox, Edge, Safari). Server-side requirements include a multi-core processor, 8–16 GB RAM, 100 GB+ storage, and stable high-speed internet. The platform may be deployed on cloud environments such as Google Cloud Platform, AWS, or Vercel.

F. Technology Stack

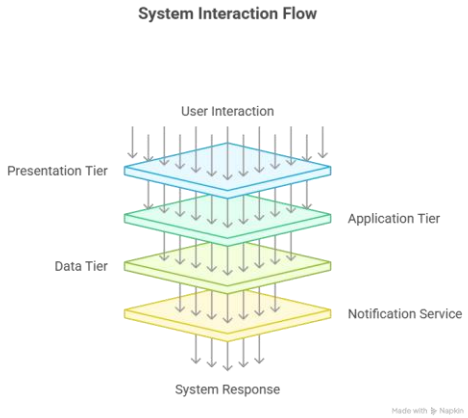
Frontend: React.js, HTML5, CSS3, JavaScript. Backend: Java with Spring Boot, or Node.js with Express.js. Database: MySQL (relational) or MongoDB (documentoriented). APIs: RESTful services over HTTPS. Notification Services: SMTP or SendGrid for email alerts. Version Control: Git/GitHub. Cloud Storage: Google Cloud or Firebase for document and resume uploads.

IV. SYSTEM DESIGN

A. System Architecture

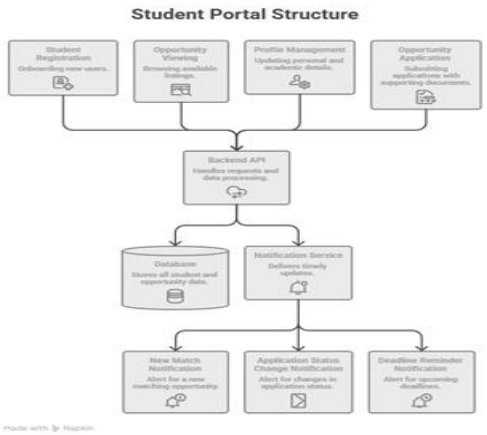
The system follows a three-tier client-server architecture. The presentation tier consists of the three role-specific portals (Student, Company, Admin) built with React.js. The application tier hosts the backend RESTful API server (Java Spring Boot or Node.js), which handles business logic, authentication, and data management. The data tier comprises the database server (MySQL/MongoDB) and cloud storage for files. A dedicated Notification and Email Service module handles asynchronous alerts. All communication between tiers uses HTTPS.

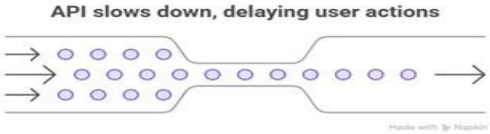
The Admin Portal acts as the central governance layer, connecting to all other components to enforce content quality. The Company Portal and Student Portal interact with the Backend Server and Database through the API layer, ensuring separation of concerns and modularity.



B. Data Flow

The Student Portal structure encompasses four key flows: Student Registration (onboarding new users), Opportunity Viewing (browsing available listings), Profile Management (updating personal and academic details), and Opportunity Application (submitting applications with supporting documents). Each flow routes through the backend API and interacts with the database, while the Notification Service delivers timely updates at key lifecycle events (new match, application status change, deadline reminder).



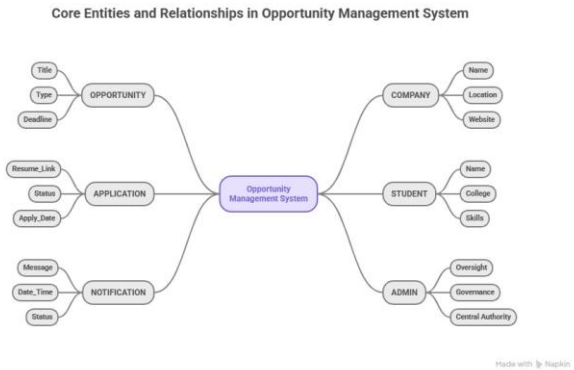


C. Entity-Relationship Model

The core entities and their key attributes are:

- OPPORTUNITY: Title, Type (intern-ship/job/hackathon/contest), Deadline
- COMPANY: Name, Location, Website
- APPLICATION: Resume Link, Status, Apply Date
- STUDENT: Name, College, Skills
- NOTIFICATION: Message, Date _Time, Status
- ADMIN: (governance entity linking to all above)

The ADMIN entity maintains oversight relationships with OPPORTUNITY, COMPANY, APPLICATION, STUDENT, and NOTIFICATION entities, reflecting the admin's role as the central authority. Students submit Applications for Opportunities posted by Companies, and the system generates Notifications for relevant events.



D. UML Use Case

The use case diagram identifies three primary actors — Student, Company, and Admin — interacting with the Event Aggregator System. Students can Register/Login, View Opportunities, and Apply for Opportunities. Companies can Register/Login, Manage Opportunities, and view applicant profiles. Admins can Register/Login and perform the exclusive action of Approving Posts, in addition to sharing the Manage Opportunities use case with companies for administrative actions. The Approve Posts use case is the critical governance boundary that ensures platform quality

V. PROJECT PLAN

A. SDLC Model

The Agile SDLC model was selected for its support of iterative development and rapid adaptation to evolving requirements. Since the platform encompasses multiple independent modules (Student, Company, and Admin portals), Agile sprints allow each module to be developed, tested, and refined independently before integration. Agile is particularly suitable for web-based platforms where new features — such as AI-based recommendations, analytics dashboards, or third-party integrations — are likely to emerge during development based on early user feedback.

B. Implementation Plan

Phase	Duration	Deliverables
Requirement Analysis	1–2 weeks	SRS, UML Diagrams
System Design	1 week	Architecture, DB Schema
Backend Development	3–4 weeks	RESTful API, Database
Frontend Development	3–4 weeks	UI Portals, Dashboards
Integration	1 week	Full-Stack Integration
Testing	2 weeks	Test Reports, Bug Fixes
Deployment	1 week	Live System
Maintenance	Ongoing	Updates, Documentation

Table 1: System Implementation Plan

C. Cost Estimates

The platform leverages entirely open-source technologies (React.js, Node.js, Java Spring Boot, MySQL, MongoDB) resulting in zero licensing costs. Cloud hosting on a basic-tier environment (AWS/GCP/Azure) is estimated at 3,000–5,000 per month during testing and deployment. Domain registration and SSL certificate costs are estimated at 1,200–2,000 per year. Total projected development cost is approximately 10,000–15,000, making the platform highly cost-effective. D. Risk Management

Seven primary risk categories were identified and mitigated:

- Algorithmic Complexity: Recommendation and match-ing algorithms may exhibit exponential

growth. Mitigation: Use approximate algorithms, caching, and pagination.

- Performance & Scalability: High concurrent load may degrade response times. Mitigation: Load balancing, database indexing, and CDN deployment.
- Data Consistency: Multi-user concurrent writes risk raceconditions. Mitigation: ACID-compliant database transactions and optimistic locking.
- Security & Privacy: Sensitive student and company datarequires strong protection. Mitigation: HTTPS, role-based access control, encrypted storage, and regular audits.
- Integration Risk: Three-portal integration may introduceAPI incompatibilities. Mitigation: API versioning, contract testing, and staged rollout.
- Resource Allocation: Server resource contention underpeak load. Mitigation: Auto-scaling policies and performance dashboards.
- Requirement Volatility: Evolving needs may requirearchitectural changes. Mitigation: Agile sprints, modular design, and structured change management.

VI. ADVANTAGES, LIMITATIONS, AND APPLICATIONS

A. Advantages

- Single centralized platform eliminates multi-sourcebrowsing for students.
- Role-based portals ensure each stakeholder sees onlyrelevant functionality.
- Admin verification enforces data quality and preventsspam or fake listings.
- Automated notifications and deadline reminders increasestudent participation rates.
- Transparent application tracking reduces ambiguity forstudents and recruiters alike.
- Open-source technology stack minimizes deploymentcosts.
- Modular architecture supports future feature additionswithout major rework.

B. Limitations

- Requires stable internet connectivity, disadvantagingstudents in low-connectivity areas.
- System still depends on companies to provide accurateinformation despite admin verification.
- Initial user adoption may be slow, particularly amonginstitutions with established workflows.
- Admin moderation creates a bottleneck; delayed approvals slow opportunity publication.
- Performance may degrade at very high concurrencywithout infrastructure investment.

C. Applications

- College Career Portal: Unified resource for exploringinternships, jobs, hackathons, and contests.
- Campus Placement Support: Colleges can share place-ment drives with their students directly.
- Company Recruitment: Organizations can find candi-dates from multiple institutions efficiently.
- Hackathon & Competition Promotion: Event organizerscan increase participation through the platform.
- Skill Gap Analysis: Colleges can analyzeapplicationpatterns to identify areas needing training.
- Resume Repository: Students maintain updated profilesaccessible across all applications.

VII. CONCLUSION AND FUTURE WORK

This paper presented the design and specification of an Event Aggregator for College Students, a role-based web platform that centralizes the discovery, verification, and management of academic and professional opportunities. The three-portal architecture (Student, Company, Admin) cleanly separates concerns while enabling rich interaction among stakeholders. The system's technical design — grounded in RESTful APIs, a relational/document database, and cloudhosted infrastructure — ensures scalability, reliability, and security. Risk management strategies drawn from NP-hard algorithmic analysis ensure the platform remains performant under realistic load conditions. By eliminating information scatter and democratizing access to opportunities, the platform has the potential to meaningfully improve student career outcomes, particularly for those from institutions with limited placement infrastructure.

Several extensions are planned for future development:

- AI-Based Personalized Recommendation: Integrate MLalgorithms (collaborative filtering, BERT embeddings) to recommend opportunities matching student skills and application history [8][10].
- Automated Resume Screening: Implement NLP-basedresume parsing and scoring to help companies shortlist candidates efficiently [3][4].

- Real-Time Interview Scheduling: Add calendar integration for scheduling and reminders.
- Mobile Application: Native Android and iOS apps forpush notifications and mobile-first access.
- Learning Platform Integration: Connect with Coursera,Udemy, and LinkedIn Learning to suggest skill-development courses based on identified gaps.
- Real-Time Chat: Enable direct messaging between students and company HR teams within the platform.
- Fairness-Aware Filtering: Implement bias-mitigationmeasures in recommendation outputs, informed by fairness research in recruitment AI [9].

REFERENCES

[1] ” L. Wenle, ”Research on personalised recommendation algorithm for college students’ employment,” Applied Mathematics and Nonlinear Sciences, vol. 8, no. 2, 2023.
[2] “D. Celik Ertugrul et al., ”Job Recommender Systems: A Systematic Literature Review,” 2025.
[3] “E. Senger et al., ”A Survey on Skill Extraction and Classification from Job Postings,” 2024.
[4] “A. Akkasi et al., ”Job Description Parsing with Explainable Transformer-Based Models,” 2024.
[5] “ S. Singh, ”Natural Language Processing for Information Extraction,” arXiv, 2018.
[6] ”D. Mhamdi et al., ”Job Recommendation Based on Job Profile Clustering and Matching,” 2020.
[7] ”F. R. Brambilla et al., ”Hackathons as a Strategy for Open Innovation,” 2022.
[8] ”S. Soni and S. Shah, ”Job Recommendation Systems Through AI and Machine Learning,” International Journal of Basic and Applied Sciences, 2025.
[9] ”D. Kumar, T. Grosz, N. Rekabsaz, E. Greif, and M. Schedl, ”Fairness of Recommender Systems in the Recruitment Domain: Technical and Legal Perspectives,” 2023.
[10] ”Y. Wang et al., ”Design and Implementation of Student Job Matching System Based on Personalized Recommendation Algorithm,” 2025.
[11] ”F. Guo et al., ”Implementation and Application of Employment Recommendation System for College Students with Personalized Preferences,” Applied Mathematics and Nonlinear Sciences, vol. 9, no. 1, 2024.
[12] ”S. R. Mandalapu, B. Narayanan, and S. Putheti, ”Job Recommendation System Using Deep Reinforcement Learning (DRL),” IJRITCC, vol. 11, 2023.