EDUCAREER:SMART LEARNING AND PLACEMENT ANALYTICS HUB

A PROJECT REPORT

Submitted by

PRIYA T **(111421104092)**

SUNITA SHAKUNIYA **(111421104105)**

ROSARIN TEENA **(111421104101)**

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

PRATHYUSHA ENGINEERING COLLEGE

THIRUVALLUR-602 025

ANNA UNIVERSITY :: CHENNAI 600 025

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this project report "**EDUCAREER:SMART LEARNING AND PLACEMENT ANALYTICS HUB**" is the bonafide work of the "**PRIYA T (111421104092)**" **,** , "**ROSARIN TEENA (111421104101)**" , "**SUNITA SHAKUNIYA (111421104105)** " who carried out the project work under my supervision

SIGNATURE
Dr.THAMBA MESHACH W,
PROFESSOR,
HEAD OF THE DEPARTMENT,
Department of CSE,
Prathyusha Engineering College,
Thiruvallur – 602 025

SIGNATURE
Dr.THAMBA MESHACH W,
PROFESSOR,
SUPERVISOR,
Department of CSE,
Prathyusha Engineering College,
Thiruvallur – 602 025

Place: Thiruvallur
Date:

Submitted for the Project Viva-Voce held on…….…. at Prathyusha Engineering College, Thiruvallur- 602 025.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Our sincere thanks to **Shri. P. RAJARAO,** Chairman, **Shri. CHARAN TEJA,** Vice Chairman and **Smt. P. PRATHYUSHA**, CEO, Prathyusha Engineering College for facilitating us to do this project.

We would like to express our heartfelt gratitude to **Dr. R.S. KUMAR**, Principal of Prathyusha Engineering College, for supporting our efforts on this initiative.

We are sincerely thanking **Dr. THAMBA MESHACH W,** Professor, Head of the Department, for his guidance for the successful completion of this project.

We are grateful to our project coordinator **Mrs. SOUMYA T.R,** Assistant Professor, for his valuable suggestions and guidance.

We are more thankful to our project guide **Dr. THAMBA MESHACH W,** Professor**,** for assisting us with his suggestions to complete our project.

We also wish to extend our sincere thanks to all the committee members for their constant support throughout the review.

We wish to express our sincere gratitude to **PARENTS AND FRIENDS** for valuable help, cooperation, and encouragement during this project work.

Last but not least, we wish to express our sincere thanks to the entire teaching faculty and non-teaching staff for their support.

ABSTRACT

Educational placement systems are transforming with digital solutions to effectively bridge the gap between students and employment opportunities. Our project, the EduCareer: Smart Learning And Placement Analytics Hub, embodies this evolution by integrating advanced web technologies into academic placement processes to enhance career readiness and institutional efficiency. The system is built on a robust architecture containing modules for student profiling, drive management, and analytics, enabling precise opportunity matching. By leveraging modern frameworks such as React.js for the frontend and Node.js with Express.js for the backend, the platform achieves seamless performance, making it ideal for integration into college placement cells and corporate recruitment systems to facilitate data-driven hiring practices. The role-based access control (RBAC) approach creates distinct workflows efficiently for different user types. Unlike traditional placement systems that rely on manual coordination and spreadsheet tracking, our solution utilizes Firebase Authentication and real-time MySQL databases to manage user permissions dynamically. The proposed method eliminates dependency on paper-based processes and removes the need for fragmented communication, resulting in an optimized and transparent placement ecosystem. The organization of placement activities into structured digital workflows is essential for effective career services and institutional reputation management. Our project addresses placement challenges by facilitating collaboration between students, colleges and recruiters via an intuitive web interface. Advanced dashboard analytics provide insights into placement trends, skill gaps, and recruitment patterns, enhancing placement cell efficiency. While the system currently excels in drive management and student-recruiter coordination, future updates will incorporate AI-based resume analysis and expand mobile accessibility features.
capabilities.

Keywords: Placement Management, Student Recruitment, React.js, Node.js, Firebase Authentication, MySQL, Data Analytics, Role-Based Access Control, Drive Scheduling, Employability, Institutional Collaboration, Career Services..

TABLE OF CONTENTS

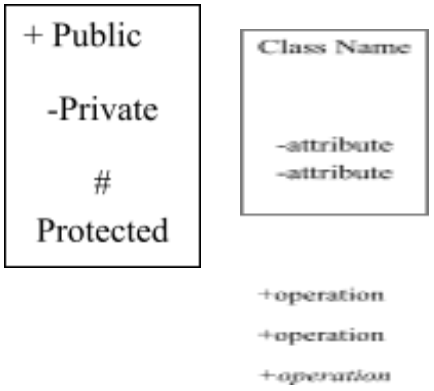
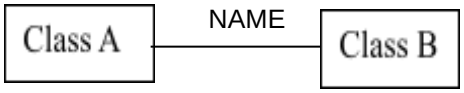
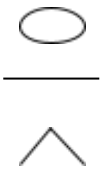
CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	I
	LIST OF FIGURES	IV
	LIST OF SYMBOLS	V
1	**INTRODUCTION**	1
	1.1 OVERVIEW	2
	1.2 OBJECTIVE	2
	1.3 LITERATURE SURVEY	3
2	**SYSTEM ANALYSIS**	6
	2.1 EXISTING SYSTEM	7
	2.1.1 DISADVANTAGES	7
	2.2 PROPOSED SYSTEM	7
	2.2.1 ADVANTAGES	8
3	**SYSTEM REQUIREMENTS**	9
	3.1 HARDWARE REQUIREMENTS	10
	3.2 SOFTWARE REQUIREMENTS	11
	3.3 SOFTWARE DESCRIPTION	12
4	**SYSTEM DESIGN**	15
	4.1 SYSTEM ARCHITECTURE	16
	4.2 UML DIAGRAM	19
	4.2.1 USE CASE DIAGRAM	19
	4.2.2 ACTIVITY DIAGRAM	20
	4.2.3 SEQUENCE DIAGRAM	21

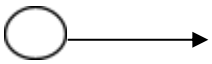
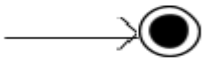
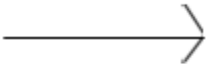
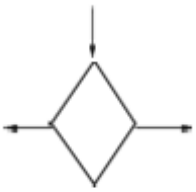
CHAPTER NO	TITLE	PAGE NO
5	**SYSTEM IMPLEMENTATION**	22
	5.1 LIST OF MODULES	23
	5.2 MODULE DESCRIPTION	23
	5.2.1: USER AUTHENTICATION MODULE	23
	5.2.2: STUDENT AND RECRUITER MANAGEMENT MODULE.	24
	5.2.3: PLACEMENT DRIVE AND RESOURCE MANAGEMENT MODULE	25
	5.2.4: ANALYTICS, AND REPORTING MODULE.	25
	5.2.5: DATABASE MANAGEMENT MODULE	26
	5.2.6: ADMINISTRATOR DASHBOARD MODULE	26
6	**TESTING**	28
	6.1 UNIT TESTING	29
	6.2 INTEGRATION TESTING	30
	6.3 USER ACCEPTANSE TESTING	31
7	**RESULTS AND DISCUSION**	33
	7.1 RESULTS	33
	7.2 DISCUSSION	33
8	**CONCLUSION AND FUTURE ENHANCEMENT**	35
	8.1 CONCLUSION	36
	8.2 FUTURE ENHANCEMENTS	36
	ANNEXURE	38
	APPENDIX 1 SAMPLE CODE	39
	APPENDIX 2 OUTPUT SCREENSHOTS	58
	REFERENCES	66

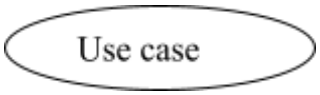
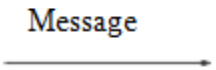
LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
4.1	System architecture of the Platform.	18
4.2	Working process of the Platform.	19
4.3	Use Case Diagram for Platform End User Navigation.	20
4.4	Activity Diagram for Admin Navigation.	21
4.5	Sequence Diagram for End Recruiter Navigation.	22

LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class		Represents a collection of similar entities grouped together
2.	Association		Associations represents static relationships between classes
			Roles represents the way the two classes see each other
3.	Actor		Specifies a role played by a user that interacts with the subject.

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
4.	Communication		Communication between various use cases.
5.	Initial State		Initial state of the object
6.	Final State		Final state of the object
7.	Control Flow		Represents various control flow between the states.
8.	Decision Box		Represents decision making process from a constraint.

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
9.	Use Case		Interaction between the system and external environment.
10.	Externalenity		Represents externalentities such as keyboard , sensors etc.
11.	Transition		Represents communication that occurs between Processes
12.	Object Life Line		Represents the vertical dimensions that the object communicates.
13.	Message		Represents the Messages exchanged

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The integration of technology into placement management systems has become essential in addressing the challenges faced by educational institutions in organizing and tracking student placements. Our application aims to streamline the placement process by offering a comprehensive solution that includes centralized access to resources, company drive management, and placement analytics functionalities. Users interact with our application through an intuitive interface where students, placement coordinators, and companies can register and authenticate their accounts securely. One of the key features of our application is the management of placement drives, where coordinators can schedule, track, and analyze upcoming and completed drives, while students can register for drives and submit their details for personalized opportunities. By leveraging deep learning and real-time data storage capabilities provided by MySQL and Firebase, our application maintains a robust database of student academic records, company details, and placement statistics. This data is utilized to provide coordinators with actionable insights, such as CTC trends, department-wise placement percentages, and skill-based analytics, while enabling students to explore internships, courses, and hackathons tailored to their profiles. Through the integration of technology and placement best practices, our application offers a practical solution to the challenges of inefficient placement processes. By empowering students, coordinators, and companies with tools for effective collaboration and data-driven decision-making, we strive to create a more organized, transparent, and successful placement ecosystem for educational institutions.

1.2 OBJECTIVE

The objective of this project is to develop an integrated the Smart Learning And Placement Analytics Hub application that streamlines the placement process for educational institutions by providing centralized access to resources, company drive management, and placement analytics. By leveraging advanced technologies such as deep learning, real-time data storage, and role-based authentication, the aim is to create a one-stop platform where students can explore and utilize resources like internships, courses, and hackathons from various platforms (e.g., GFG, w3 School, UnStop,

Coursera, Udemy) in one place. The application will also enable placement coordinators to track and analyze placement trends, manage drives, and generate actionable insights, while companies can efficiently schedule and conduct placement activities. Through the integration of technology and placement best practices, the project aims to enhance the efficiency, transparency, and success of the placement process, ultimately contributing to a more organized, resourceful, and data-driven placement ecosystem.

1.3 LITERATURE SURVEY

[1] Monika Tripathi, Dimpal Jain, Khushboo Sharma, Anuradha, Daulatram, "Web Development Framework", International Journal of Advanced Research in Science, Communication and Technology, vol. 4, issue 1, December 2024.

This article provides a comprehensive review of web development frameworks, covering front-end, back-end, and full-stack frameworks like React, Angular, Express.js, and Django. It discusses their role in creating efficient, scalable, and maintainable web applications, highlighting key features, advantages, and use cases. The study also explores emerging trends such as serverless architecture and AI integration, along with challenges in implementation and future prospects for web development practices.

[2]Ivan Suster, Tamara Ranisavljevic, "Optimization of MySQL Database", Journal of Process Management and New Technologies, vol. 11, issue 1-2, pp. 141-151, 2023.

This paper explores optimization techniques for MySQL databases, focusing on physical programming and data tuning to enhance performance, scalability, and reliability. Physical programming involves optimizing data storage through partitioning, indexing, and selecting appropriate data types and storage engines, while data tuning focuses on query optimization, configuration adjustments, and hardware improvements. The study highlights the benefits of these techniques, such as improved query efficiency, reduced storage requirements, and better resource utilization.

[3] Wen-Tin Lee, Chih-Hsien Chen, "Agile Software Development and Reuse Approach with Scrum and Software Product Line Engineering", Electronics, vol. 12, issue 15, 2023.

This research paper introduces SPLE-Scrum, a hybrid approach combining Scrum and Software Product Line Engineering (SPLE) to enhance agile software development and reuse. It describes the integration of SPLE's core asset reuse with Scrum's iterative practices, demonstrated through a blockchain online store case study. The study evaluates the approach's effectiveness in improving requirement engineering, design, and project success, supported by expert interviews and practitioner surveys. The paper highlights SPLE-Scrum's potential for optimizing agile development while enabling systematic software reuse.

[4] Mahesh Elango, "UI/UX Design: Next Generation Perspectives", Technical Report, August 2024.

This technical paper discusses the integration of advanced technologies, including artificial intelligence, augmented reality, and biometrics, in next-generation UI/UX design. The paper highlights the importance of sustainability-centric design and introduces the "Contextual Fusion Design" framework for crafting interfaces that evolve with user needs and contexts.

[5] Michael Adelusola, "Efficient Model Deployment Strategies for LLMs in Web Applications", Research Article, June 2023.

This paper explores efficient deployment strategies for Large Language Models (LLMs) in web applications, focusing on performance optimization, scalability, and cost management. It discusses techniques like model compression, cloud and edge computing, and hybrid architectures to reduce latency and resource usage. The paper also highlights cost-saving measures such as serverless computing and spot instances, alongside security practices like data encryption and access control to ensure user privacy.

[6] Yash Jani, "Optimizing Database Performance for Large-Scale Enterprise Applications", International Journal of Science and Research (IJSR), vol. 11, issue 10, pp. 1394-1396, 2022.

An efficient database performance is critical for large-scale enterprise applications to ensure quick data retrieval, seamless user experiences, and robust backend operations. This paper explores advanced optimization strategies, including indexing, query optimization, caching, multithreading, and NoSQL database utilization. It highlights techniques such as clustered indexes, materialized views, and distributed caching to enhance performance.

[7] Gokul Pandy, Vigneshwaran Jagadeesan Pugazhenthi, Aravindhan Murugan, "Advances in Software Testing in 2024: Experimental Insights, Frameworks, and Future Directions", International Journal of Advanced Research in Computer and Communication Engineering, vol. 13, issue 11, pp. 41-44, 2024.

This research paper presents the integration of AI and machine learning in software testing to address the growing complexity of modern software systems. It introduces Smart Test, an AI-driven testing framework, and evaluates its performance through experiments, showing improvements in testing coverage, defect detection, and efficiency. The study highlights challenges like scalability, bias mitigation, and ethical concerns, offering insights and recommendations for advancing AI in software testing.

CHAPTER 2

SYSTEM ANALYSIS

CHAPTER 2

SYSTEM ANALYSIS

The system study is to provide a description of the existing system and the proposed system of the project.

2.1 EXISTING SYSTEM

The existing system for managing placements in educational institutions often relies on manual processes and fragmented tools, leading to inefficiencies and lack of transparency. Placement coordinators typically use spreadsheets or basic software to track student details, company drives, and placement statistics, which are time-consuming and prone to errors. Students, on the other hand, rely on multiple platforms to find resources like internships, courses, and hackathons, resulting in a lack of centralized access to opportunities. Companies face challenges in coordinating with institutions due to the absence of a unified system for scheduling and conducting drives. These limitations highlight the need for an integrated, technology-driven solution that streamlines the entire placement process.

2.1.1 DISADVANTAGES

- Communication Challenges.
- Manual Processes.
- Lack Of Centralized Resources.
- Real-Time Monitoring Deficiencies.
- Inadequate Coordination Among Stakeholders.

2.2 PROPOSED SYSTEM

The proposed system aims to enhance the efficiency, transparency, and user experience of the placement process by integrating advanced technologies and addressing the limitations of the existing system. This includes developing a centralized platform that combines student resource management, company drive coordination, and placement analytics into a single, user-friendly application. The system will

leverage deep learning algorithms to provide personalized recommendations for students, such as internships, courses, and hackathons, based on their skills, academic performance, and preferences. For placement coordinators, the system will offer real-time analytics and trend analysis, including CTC trends, department-wise placement percentages, and skill-based insights, enabling data-driven decision-making. Companies will benefit from streamlined drive scheduling, student shortlisting, and automated notifications, ensuring better coordination with institutions. Additionally, the proposed system will incorporate role-based access control and secure authentication to ensure data privacy and security. By integrating advanced technologies like real-time data storage (MySQL and Firebase), image processing for resume analysis, and machine learning for predictive analytics, the proposed system aims to create a comprehensive, efficient, and sustainable solution for placement management in educational institutions.

2.2.1 ADVANTAGES

- Enhanced Accountability and Transparency.
- Efficient Drive Management.
- Improved Coordination Among Stakeholders.
- User-Friendly Interface.
- Integration of Advanced Technologies.
- Data-Driven Decision-Making.

CHAPTER 3

SYSTEM REQUIREMENTS

CHAPTER 3

SYSTEM REQUIREMENTS

INTRODUCTION

The system aims to develop a Placement Management Application to streamline student registration, company drive coordination, and placement analytics processes. It facilitates user authentication, resource management, drive scheduling, real-time data storage, and an administrator dashboard for efficient monitoring and management. This application promotes a transparent and efficient placement ecosystem by enabling students to explore internships, courses, and hackathons in one centralized platform, while providing placement coordinators with actionable insights and tools to manage drives effectively. By integrating advanced technologies such as deep learning, real-time analytics, and role-based access control, the system ensures a seamless and user-friendly experience for all stakeholders, including students, coordinators, and companies.

3.1 HARDWARE REQUIREMENTS

- Operating System: Compatible with Windows, macOS, or Linux distributions.
- Processor: Intel Core i5 or equivalent AMD processor for optimal performance.
- RAM: Minimum 8GB RAM (16GB recommended) for smooth application performance.
- Storage: Sufficient disk space for installing software dependencies, databases, and storing application data.
- Web Browser: The Latest versions of Chrome, Firefox, or Edge for accessing the web-based application.
- Internet Connection: Stable broadband or high-speed internet for real-time data synchronization and API communication.
- Server: A dedicated server or cloud hosting service (e.g., AWS, Azure) for deploying the backend and database.

3.2 SOFTWARE REQUIREMENTS

- Operating System: Compatible with Windows, macOS, or Linux distributions for development and deployment.
- Development Frameworks: **Frontend**: React.js with Vite for building a fast and responsive user interface.**Backend**: Node.js and Express.js for handling API requests, routing, and server-side logic.
- Database Management System (DBMS): MySQL with mysql2for storing student details, company information, placement drives, and resources.
- Authentication Services: Firebase Authentication for secure user authentication, email verification and role-based access control.
- State Management: React Context API or Redux for managing global state in the frontend.
- API Integration: Axios for making HTTP requests and handling API communication between frontend and backend.
- Version Control: Git for version control and collaborative development.
- Web Server: Nginx or Apache for hosting the backend and serving HTTP requests.
- Data Visualization: Libraries like Chart.js for creating interactive charts and graphs for placement analytics.
- Image Processing: Libraries like Multer for handling file uploads (e.g., student resumes).
- Package Manager: npm or Yarn for managing dependencies and scripts.
- Code Editor: Visual Studio Code (VS Code) for development with extensions like ESLint and Prettier for code quality and formatting.
- Development Tools: In the backend used nodemon for automatically restarting the server during development.

3.3 SOFTWARE DESCRIPTION

3.3.1 Development Tools (VS Code and npm):

VS Code (Visual Studio Code): An open-source, lightweight code editor with support for various programming languages, debugging, version control, and extensions. It provides a user-friendly interface for developers to write, edit, and debug code efficiently. Extensions like ESLint and Prettier enhance code quality and formatting.

npm (Node Package Manager): A package manager for JavaScript, used to install and manage dependencies for both frontend and backend development. It simplifies the process of adding libraries, frameworks, and tools to the project.

3.3.2 Frontend Development (React.js and Vite):

React.js: A popular JavaScript library for building user interfaces. React allows developers to create reusable UI components and manage state efficiently, making it ideal for building dynamic and responsive web applications.

Vite: A modern build tool that provides fast development server startup and hot module replacement (HMR). It optimizes the development workflow by bundling and serving code efficiently, ensuring a smooth development experience.

3.3.3 Backend Development (Node.js and Express.js):

Node.js: A runtime environment that allows developers to execute JavaScript on the server side. It is lightweight, efficient, and well-suited for building scalable backend services.

Express.js: A web application framework for Node.js that simplifies the creation of RESTful APIs. It provides robust features for routing, middleware integration, and handling HTTP requests and responses**.**

3.3.4 Database Management (MySQL with mysql2):

MySQL: A relational database management system (RDBMS) used to store and manage structured data. It is chosen for its reliability, scalability, and ease of use in handling placement-related data such as student details, company information, and placement drives.

mysql2: A MySQL client for Node.js that provides efficient database connectivity and supports prepared statements, making it ideal for secure and performant database operations.

3.3.5 Authentication Services (Firebase Authentication):

Firebase Authentication: A secure authentication service provided by Google. It supports multiple authentication methods, including email/password, Google Sign-In, and more. Firebase ensures secure user authentication and role-based access control for the application.

3.3.6 State Management (React Context API and Redux):

React Context API: A built-in feature in React for managing global state without prop drilling. It is used to share data (e.g., user authentication status) across components efficiently.

Redux: A state management library for JavaScript applications. It provides a centralized store for managing application state, making it easier to debug and maintain large-scale applications.

3.3.7 API Integration (Axios):

Axios: A promise-based HTTP client for making API requests. It simplifies communication between the frontend and backend by handling requests, responses, and errors efficiently.

3.3.8 Data Visualization (Charts):

Charts.js: A modern charting library that provides interactive and responsive charts. It is used to visualize placement trends, CTC data, and department-wise performance in the admin dashboard.

3.3.9 Image Processing (Multer):

Multer: A middleware for handling file uploads in Node.js. It is used to manage student resume uploads and other file-related operations.

3.3.10 Deployment Tools (nodemon):

nodemon: A utility that monitors changes in the backend code and automatically restarts the server during development. It enhances productivity by eliminating the need for manual server restarts.

3.3.11 Version Control (Git):

Git: A distributed version control system used to track changes in the codebase. It enables collaborative development and ensures code integrity through branching, merging, and versioning**.**

3.3.12 Mobile Application :

React Native: A framework for building cross-platform mobile applications using React. It allows the application system to be extended to mobile devices, providing a seamless experience for students and coordinators on the go.

3.3.13 Docker :

Docker: A platform for containerizing applications. It ensures consistency across development, testing, and production environments by packaging the application and its dependencies into lightweight containers.

CHAPTER 4

SYSTEM DESIGN

CHAPTER 4
SYSTEM DESIGN

System design is the process of defining the architecture, components, modules, interfaces, and data flow for a system to satisfy specified requirements. It serves as the blueprint for building the system, ensuring that all features and functionalities are well-planned and implemented effectively. The EduCareer:Smart Learning And Placement Analytics Hub is designed to streamline the placement process for educational institutions by providing a centralized platform for students, placement coordinators, and companies. Below is a detailed breakdown of the system design.

4.1 SYSTEM ARCHITECTURE

System architecture is the conceptual model that defines the structure, behavior, and various views of a system. It provides a blueprint for designing and implementing the system, ensuring that all components work together seamlessly to achieve the desired functionality. The architecture description is a formal representation of the system, organized to support reasoning about its structures and behaviours.

For the EduCareer:Smart Learning And Placement Analytics Hub, the architecture is designed to ensure scalability, modularity, and efficiency. It follows a three-tier architecture, which separates the system into three main layers:

Presentation Layer (Frontend): It is responsible for the user interface and user experience. It is the part of the system that users interact with directly.

Application Layer (Backend): The Application Layer handles the business logic, processes requests from the frontend, and interacts with the database. It acts as the bridge between the frontend and the database.

Data Layer (Database): The Data Layer is responsible for storing and managing the system's data. It ensures data integrity, security, and efficient retrieval.

Each layer has a specific role and interacts with the others to deliver a cohesive and functional system

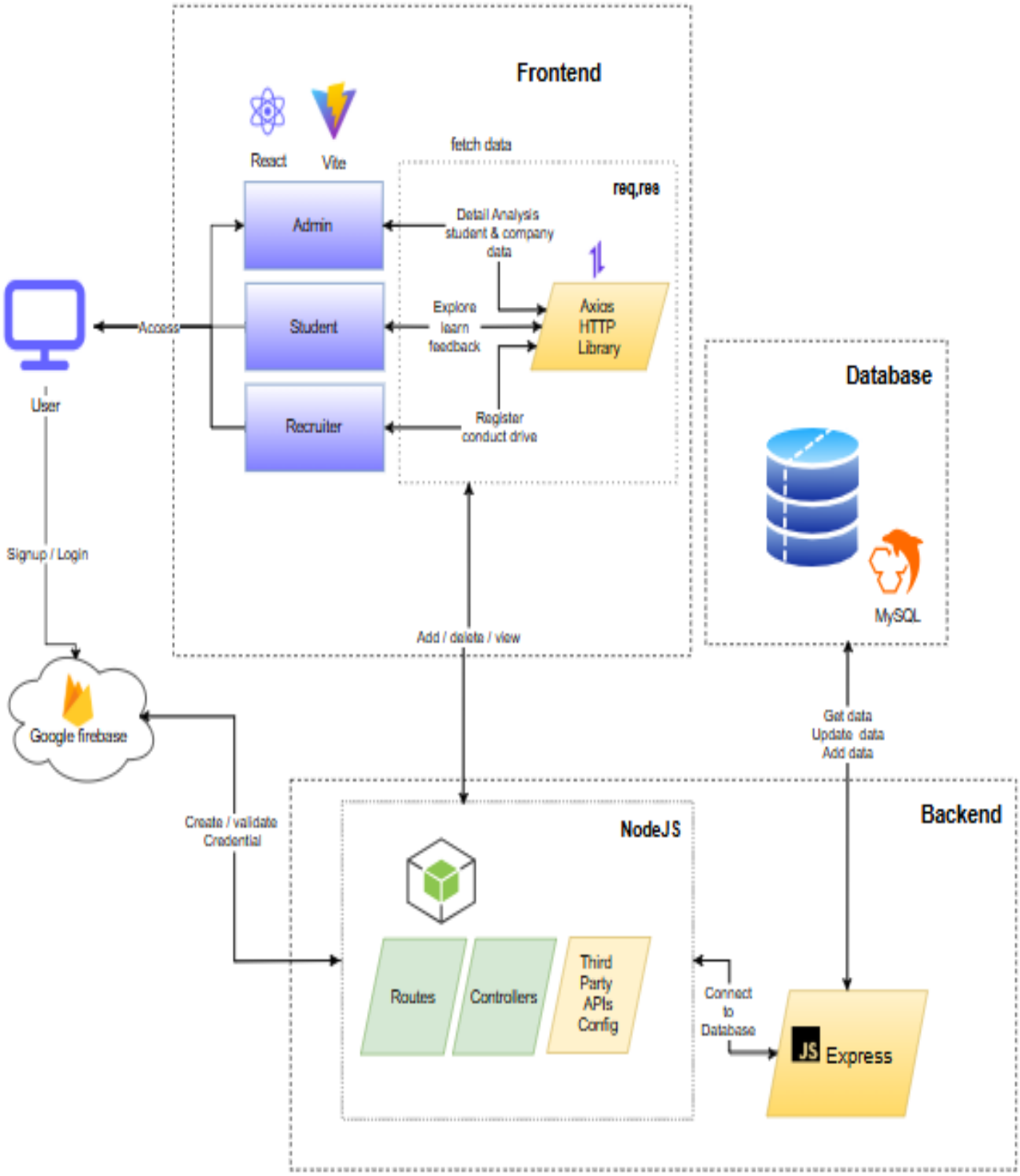


Fig 4.1 Architecture Diagram

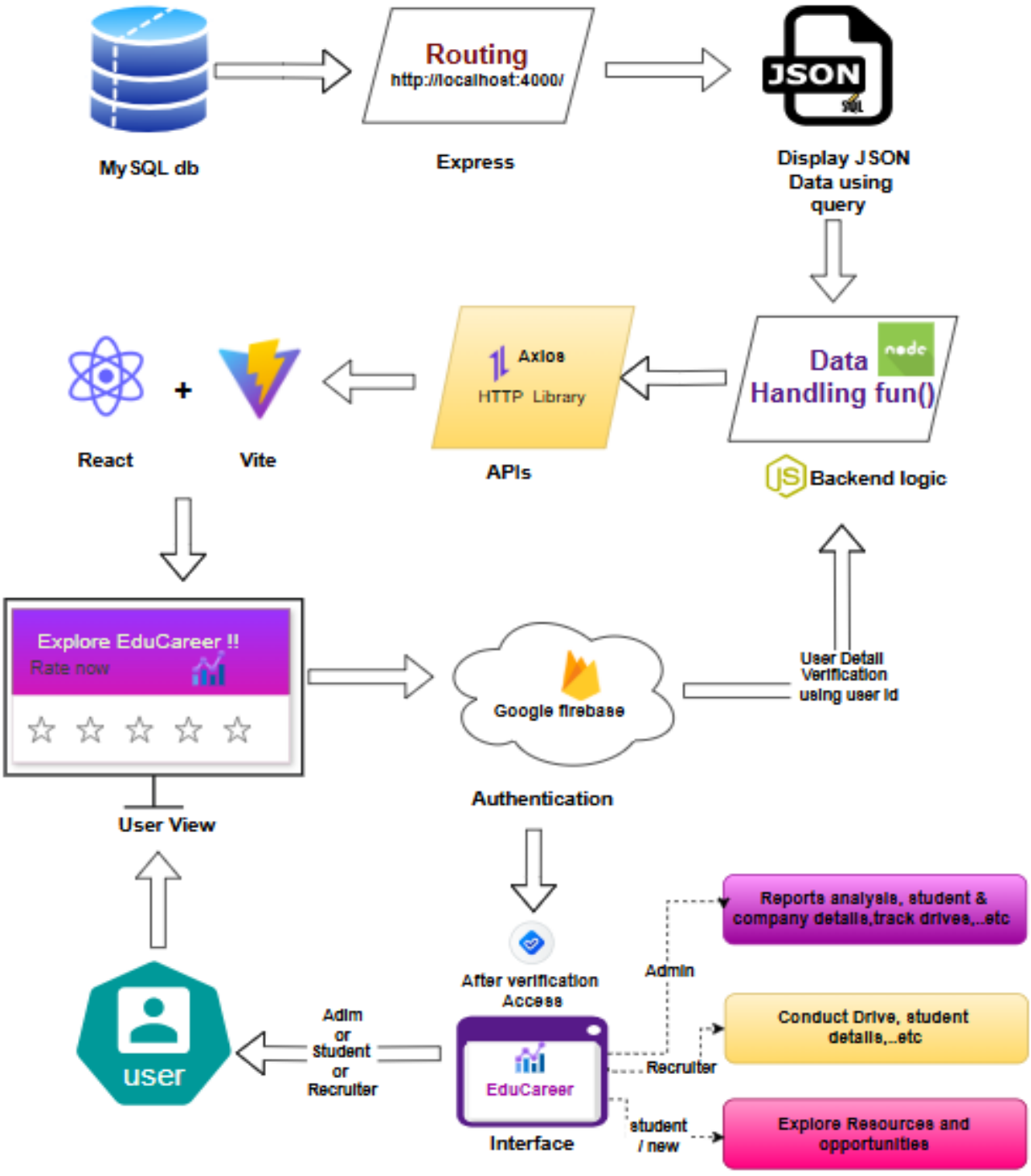


Fig 4.2 Working process of the image detection

4.2 UML DIAGRAM

4.2.1 USE CASE DIAGRAM

Use cases are used to describe the visible interactions that the system will have with users and external systems. They provide a clear understanding of how users perform their roles using the system. The Use Case Diagram for the EduCareer:Smart Learning And Placement Analytics Hub application illustrates the interactions between students, recruiters, and admins with the system, highlighting their access to specific features and functionalities.

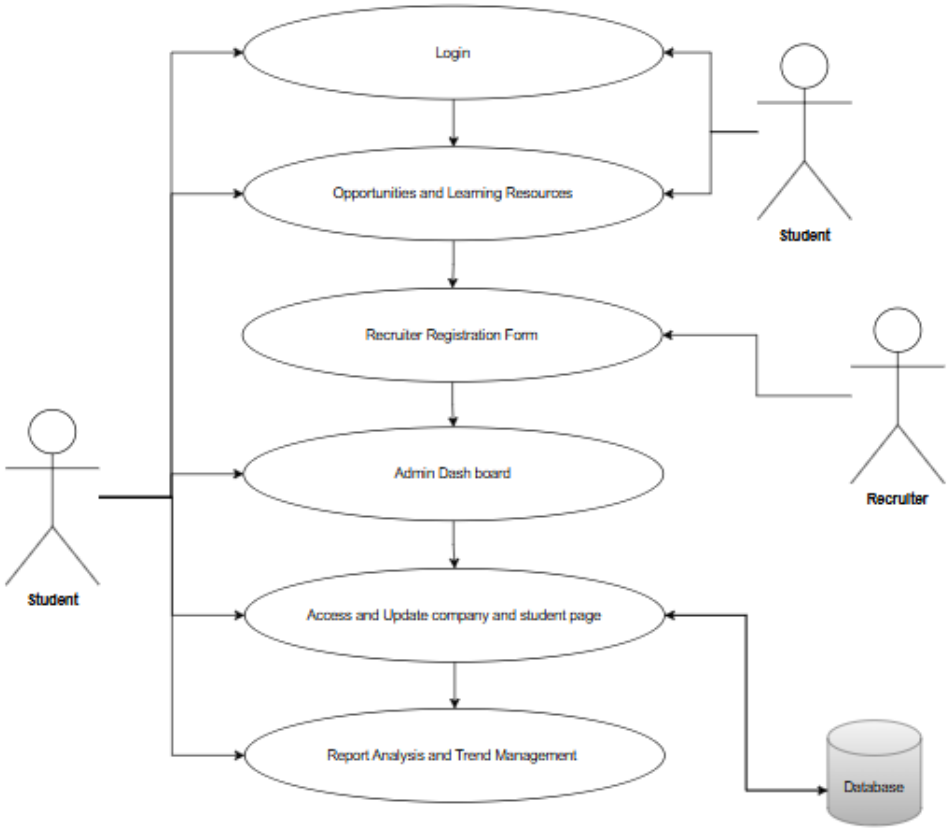


Figure 4.3 Use Case Diagram for Platform End User Navigation

4.2.2 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

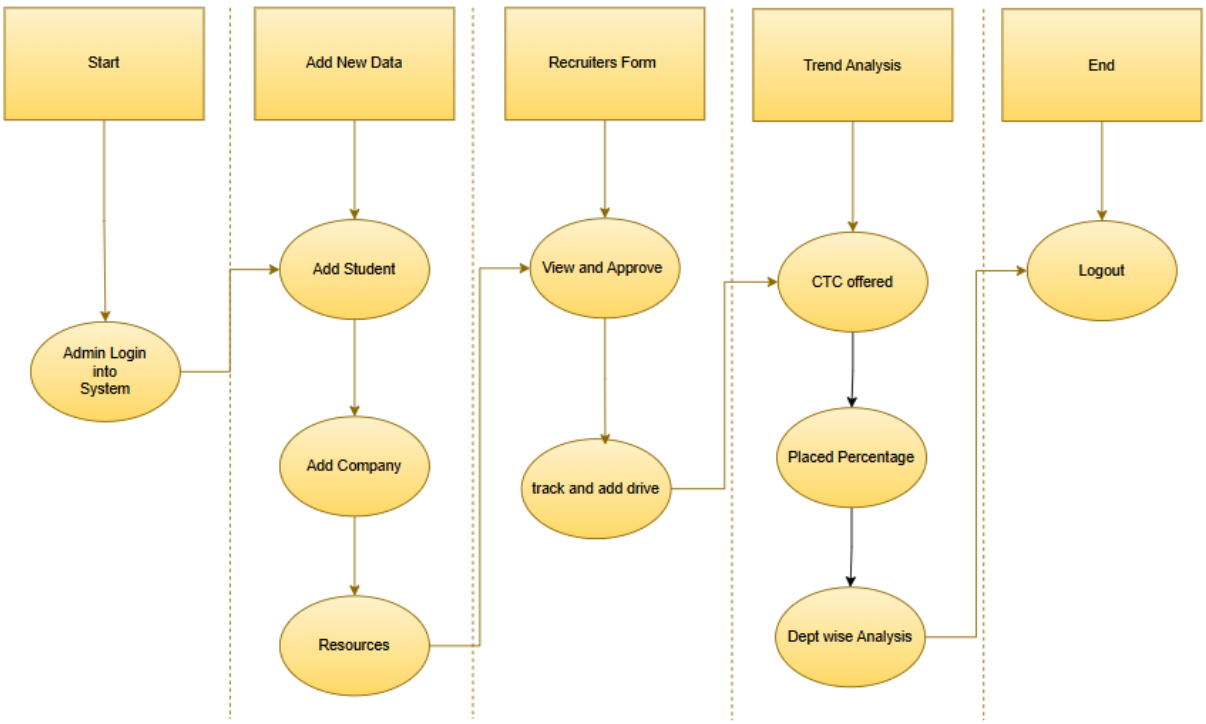


Figure 4.4 Activity Diagram for Admin Navigation

4.2.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

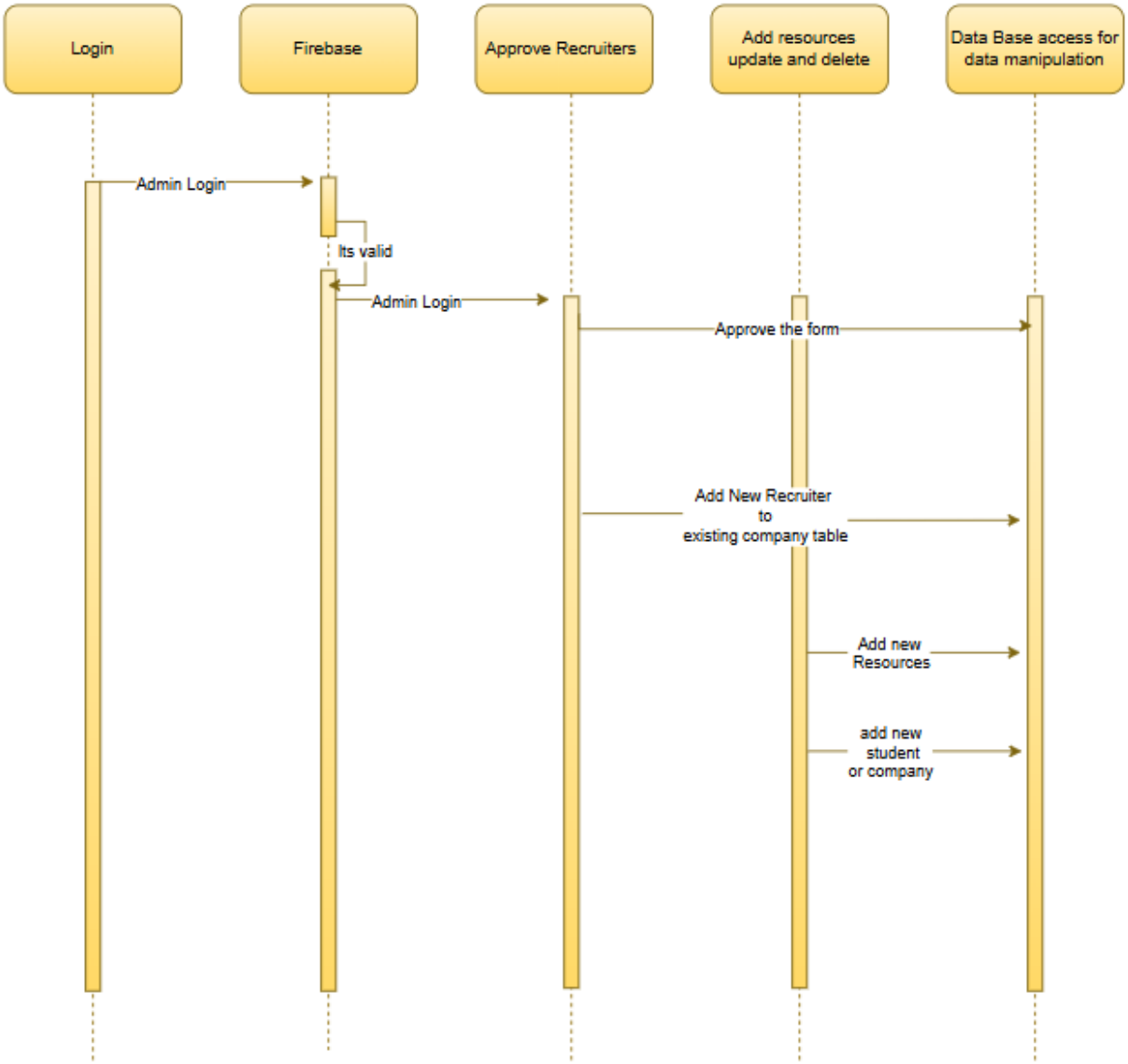


Figure 4.5 Sequence Diagram for End Recruiter Navigation

CHAPTER 5

SYSTEM IMPLEMENTATION

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 LIST OF MODULES

1. User Authentication Module.
2. Student and Recruiter Management Module.
3. Placement Drive and Resource Management Module
4. Analytics, Reporting, and Feedback Module.
5. Database Management Module
6. Administrator Dashboard Module

5.2 MODULE DESCRIPTION

5.2.1: User Authentication Module:

This module handles user authentication using Firebase Authentication, including email verification for secure access to the EduCareer: Smart Learning And Placement Analytics Hub. It ensures the security and authenticity of users, allowing only authorized individuals (students, recruiters, and admins) to access the platform. Key features include email verification, where users receive a verification email upon registration and can log in once verified, and role-based access control, which ensures students, recruiters, and admins can only access features relevant to their role. This secure authentication mechanism fosters trust among users and protects their personal information.

Documentation Components:

- Introduction and Purpose.
- Firebase Authentication Setup Guide.
- Email Verification Process Explanation.

- User Login Instructions.

5.2.2: Student and Recruiter Management Module

This module is designed to manage the core functionalities for students and recruiters, ensuring a seamless experience for both groups. For students, it provides tools to view and update their profiles, upload resumes, and track their placement status. Students can also explore a wide range of resources, including internships, courses, and hackathons, tailored to their skills and preferences. For recruiters, the module allows them to register their company, schedule placement drives, and manage student applications. Recruiters can set drive requirements, track student participation, and shortlist candidates efficiently. This module ensures that students have access to personalized opportunities while recruiters can manage drives effectively.

Documentation Components:

- Student Profile Management Guide.
- Resource Access Explanation.
- Recruiter Drive Management Guide.

5.2.3: Placement Drive and Resource Management Module

This module plays a critical role in managing placement drives and centralizing resources for students. Recruiters can use this module to schedule drives, set specific requirements (e.g., skills, CGPA), and track the progress of drives. Admins can add, update, or delete resources such as internships, courses, and hackathons, ensuring students have

access to the latest opportunities. Students, on the other hand, can register for drives, submit their details, and receive notifications about upcoming drives. The module also integrates with external platforms (e.g., GFG, Coursera, Udemy) to provide a comprehensive list of resources, making it easier for students to explore and apply for opportunities.

Documentation Components:

- Drive Scheduling Guide.
- Resource Management Guide.
- Student Registration Process.

5.2.4: Analytics, And Reporting Module:

This module empowers admins with tools to generate detailed reports, view analytics, and manage feedback from students and recruiters. Admins can analyze CTC trends, placement percentages, and department-wise performance using interactive charts and graphs powered by Chart.js. The module also allows students to submit feedback after attending drives or using resources, providing valuable insights for continuous improvement. By leveraging data visualization and feedback mechanisms, this module ensures data-driven decision-making and enhances the overall placement process.

Documentation Components:

- Analytics and Reporting Guide.
- Feedback Submission Process.
- Data Visualization Overiew.

5.2.5: Database Management Module:

The Database Management Module is the backbone of the EduCareer:Smart Learning And Placement Analytics Hub, ensuring efficient storage, retrieval, and manipulation of data. It uses MySQL to maintain structured tables for students, companies, drives, resources, and offers. The module ensures data integrity by enforcing constraints and relationships between tables, while data security measures protect sensitive information from unauthorized access. Optimized queries ensure fast data retrieval, even with large datasets. This module also supports backup and recovery mechanisms, ensuring data is always available and secure.

Documentation Components:

- Data Schema Overview.
- Data Security Guide
- Query Optimization Tips

5.2.6: Administrator Dashboard Module:

The Admin Dashboard Module provides a centralized interface for admins to manage all aspects of the placement process. Admins can add, update, or delete student and company details, ensuring the database is always up-to-date. They can also view and manage placement drives, add or update resources, and generate detailed reports on placement trends and performance. The dashboard includes interactive charts and graphs for visualizing data, such as CTC trends and department-wise placement percentages, enabling admins to make informed decisions. This module ensures that admins have complete control over the placement process, from managing users to analyzing performance.

Documentation Components:

- Dashboard Overview.
- User Management Guide
- Analytics and Reporting

CHAPTER 6

TESTING

CHAPTER 6
TESTING

SYSTEM TESTING AND IMPLEMENTATION

Testing is the process of executing a program or application with the intent of finding software bugs and verifying that the software product is fit for use. It ensures that the system meets the specified requirements and functions as expected. the EduCareer:Smart Learning And Placement Analytics Hub undergoes rigorous testing at various levels to ensure its reliability, functionality, and performance.

6.1 UNIT TESTING

Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately.

MODULE TESTING:

User Authentication Module Test:

Input: User credentials (email, password).

Expected Output: Successful login and role-based access.

Student Management Module Test:

Input: Student details (name, email, skills, CGPA).

Expected Output: Student profile created, updated, or deleted successfully.

Results: Ensured that student data is managed correctly in the database.

Company Management Module Test:

Input: Company details (name, website, contact information).

Expected Output: Company profile created, updated, or deleted successfully.

Results: Verified that company data is managed accurately.

Placement Drive Management Module Test:

Input: Drive details (date, location, requirements).

Expected Output: Drive scheduled and managed successfully.

Results: Confirmed that drives are scheduled and tracked correctly.

Resource Management Module Test:

Input: Resource details (type, title, platform).

Expected Output: Resource added, updated, or deleted successfully.

Results: Verified that resources are managed and displayed correctly.

Analytics and Reporting Module Test:

Input: Placement data (CTC trends, placement percentages).

Expected Output: Reports generated and displayed accurately.

Results: Confirmed that analytics and reports are generated correctly.

6.2 INTEGRATION TESTING

Integration testing ensures that individual software modules work together as a group. It occurs after unit testing and validates the interactions between modules.

Frontend-Backend Integration Test:

Input: Simulated user interactions through the frontend interface.

Expected Output: Proper routing and response generation from the backend.

Results: Ensured seamless integration between the frontend and backend components.

Database Integration Test:

Input: Data from frontend forms (student details, company details, drive details).

Expected Output: Data stored and retrieved correctly from the database.

Results: Verified that data flows accurately between the frontend, backend, and database.

Authentication Integration Test:

Input: User login credentials.

Expected Output: Secure authentication and role-based access to features.

Results: Confirmed that authentication works seamlessly across the system.

Resource Integration Test:

Input: Resource data from external platforms (e.g., GFG, Coursera, Udemy).

Expected Output: Resources displayed correctly on the frontend.

Results: Verified that resources are integrated and displayed accurately.

6.3 USER ACCEPTANCE TESTING

User Acceptance Testing ensures that the system meets the requirements and expectations of the end-users. It validates the overall functionality and user experience.

Scenario-Based Testing:

Scenarios:

Student registration and profile management.

Recruiter registration and drive scheduling.

Admin management of students, companies, and resources.

Generating reports and viewing analytics.

Expected Outcome:

Students can explore resources, register for drives, and submit feedback.

Recruiters can schedule drives, view student profiles, and manage offers.

Admins can manage users, resources, and generate reports.

Results: Validated that the system meets user requirements and provides a seamless experience.

Performance Testing:

Input: Simulated high traffic (multiple users accessing the system simultaneously).

Expected Output: System responds quickly and handles the load without crashing.

Results: Confirmed that the system performs well under stress.

Usability Testing:

Input: Feedback from students, recruiters, and admins.

Expected Output: Positive feedback on the system's ease of use and functionality.

Results: Verified that the system is user-friendly and meets user expectations.

CHAPTER 7

RESULTS AND DISCUSSION

CHAPTER 7
RESULTS AND DISCUSSION

7.1 RESULTS

The EduCareer:Smart Learning And Placement Analytics Hub developed for students, recruiters, and administrators successfully streamlines the placement process by providing a centralized platform for managing profiles, resources, and placement drives. The system integrates Firebase Authentication for secure user login and role-based access control, ensuring that students, recruiters, and admins can only access features relevant to their roles. Students can efficiently update their profiles, upload resumes, and track their placement status, while recruiters can schedule drives, set requirements, and manage applications seamlessly. The admin dashboard provides comprehensive analytics and reporting tools, enabling admins to monitor placement trends, CTC data, and department-wise performance using interactive charts powered by Chart.js. The system also integrates with external platforms like GFG, Coursera, and Udemy to provide students with a wide range of resources, including internships, courses, and hackathons. Overall, the EduCareer:Smart Learning And Placement Analytics Hub enhances the efficiency of the placement process, fosters collaboration between stakeholders, and ensures data-driven decision-making.

7.2 DISCUSSION

Our project aims to bridge the gap between students, recruiters, and administrators by providing a unified platform for managing the placement process. The system is designed to cater to the needs of all stakeholders, ensuring a seamless and efficient experience. The process begins with user authentication, where students, recruiters, and admins register using their email IDs and passwords. Firebase Authentication ensures secure access, with email verification adding an extra layer of security. Once logged in, users are directed to their respective dashboards based on their roles

.For students, the system provides tools to update their profiles, upload resumes, and track their placement status. They can also explore a wide range of resources, including internships, courses, and hackathons, tailored to their skills and preferences. The integration with external platforms like GFG, Coursera, and Udemy ensures that students have access to the latest opportunities. Additionally, students can register for placement drives, submit their details, and receive notifications about upcoming drives.For recruiters, the system offers a streamlined process for scheduling placement drives, setting specific

requirements (e.g., skills, CGPA), and managing student applications. Recruiters can track the progress of drives, shortlist candidates, and communicate with students efficiently. The system ensures that recruiters have all the tools they need to manage drives effectively.

For admins, the system provides a centralized dashboard for managing all aspects of the placement process. Admins can add, update, or delete student and company details, ensuring the database is always up-to-date. They can also view and manage placement drives, add or update resources, and generate detailed reports on placement trends and performance. The dashboard includes interactive charts and graphs for visualizing data, such as CTC trends and department-wise placement percentages, enabling admins to make informed decisions. While the system has been successful in streamlining the placement process, it currently faces some limitations. For instance, the system relies heavily on MySQL for database management, which may require optimization for handling very large datasets. Additionally, the integration with external platforms is limited to a few providers, and future updates could include support for more platforms. Another limitation is the lack of a mobile application, which could enhance accessibility for students and recruiters on the go. However, these challenges will be addressed in future updates to further improve the system's functionality and effectiveness. Overall, the EduCareer:Smart Learning And Placement Analytics Hub represents a significant step forward in modernizing the placement process, fostering collaboration between stakeholders, and ensuring data-driven decision-making. By addressing the current limitations and incorporating user feedback, the system has the potential to become an indispensable tool for educational institutions and recruiters alike.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

CHAPTER 8
CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

The EduCareer:Smart Learning And Placement Analytics Hub successfully streamlines the placement process by providing a unified platform for students, recruiters, and administrators. Through secure authentication, role-based access control, and seamless data management, the system enhances efficiency, transparency, and collaboration among stakeholders. Students benefit from an organized approach to updating profiles, tracking placement status, and accessing valuable learning resources, while recruiters can efficiently schedule drives and manage applications. The admin dashboard further enables data-driven decision-making with comprehensive analytics and reporting tools. Despite its success, the system has some limitations, such as dependency on MySQL for large datasets, limited integration with external platforms, and the absence of a mobile application. However, these limitations do not overshadow the significant advancements the system brings to the placement process.

8.2 FUTURE ENHANCEMENTS

In the further improve the EduCareer: Smart Learning And Placement Analytics Hub, several enhancements can be introduced in future updates. Optimizing database management for scalability will ensure efficient handling of large datasets, enhancing overall system performance. Expanding integrations with additional external platforms, such as LinkedIn Learning and Kaggle, will provide students with a broader range of learning and career development opportunities. Additionally, developing a dedicated mobile application will enhance accessibility, allowing users to manage placements on the go. Incorporating AI-driven analytics and resume screening features will further refine the recruitment process by providing automated insights and candidate recommendations. Lastly, implementing real-time chat functionality will improve communication between students, recruiters, and admins, ensuring a more interactive and dynamic experience.

These enhancements will significantly elevate the effectiveness of the system, making it an essential tool for placement management in educational institutions.

ANNEXURE

ANNEXURE
APPENDIX I

CODE

Database Configuration:

```
import mysql from "mysql2"
import dotenv from 'dotenv'
dotenv.config()

const pool = mysql.createPool({
    host: process.env.MYSQL_HOST,
    user: process.env.MYSQL_USER,
    password: process.env.MYSQL_PASSWORD,
    database: process.env.MYSQL_DATABASE
  }).promise()
export default pool;
```

Firebase Congiguration:

```
// backend/config/firebaseConfig.js

const admin = require('firebase-admin');
require('dotenv').config();

const serviceAccount = {
  type: "service_account",
  project_id: process.env.FIREBASE_PROJECT_ID,
  private_key_id: process.env.FIREBASE_PRIVATE_KEY_ID,
  private_key: process.env.FIREBASE_PRIVATE_KEY.replace(/\\n/g, '\n'),
  client_email: process.env.FIREBASE_CLIENT_EMAIL,
  client_id: process.env.FIREBASE_CLIENT_ID,
  auth_uri: "https://accounts.google.com/o/oauth2/auth",
  token_uri: "https://oauth2.googleapis.com/token",
  auth_provider_x509_cert_url: "https://www.googleapis.com/oauth2/v1/certs",
  client_x509_cert_url: process.env.FIREBASE_CLIENT_X509_CERT_URL
};

// Initialize Firebase Admin SDK
```

```
admin.initializeApp({
  credential: admin.credential.cert(serviceAccount),
  databaseURL:
`https://${process.env.FIREBASE_PROJECT_ID}.firebaseio.com`
});

const firedb = admin.firestore();
const auth = admin.auth();

module.exports = { firedb, auth };
```

Backend Nodejs Code:

```
import pool from "../config/database.js";

// Get all students
export const getcompanies = async (req, res) => {
   try {
      const [companies] = await pool.query("SELECT * FROM companies");
      res.json(companies);
   } catch (error) {
      res.status(500).json({ error: "Database error while fetching companies
detail at initial " });
   }
};

// Add a company
export const addCompany = async (req, res) => {
   const { company_name, website, contact_email } = req.body;
   try {
      await pool.query(
         "INSERT INTO companies (company_name, website, contact_email)
VALUES (?, ?, ?)",
         [company_name, website, contact_email]
      );
      res.status(201).json({ message: "Company added successfully" });
   } catch (error) {
      console.error(error);
      res.status(500).json({ error: "Error adding company" });
   }
};
```

```
import pool from "../config/database.js";

// Submit recruiter form
export const submitRecruiterForm = async (req, res) => {
  const formData = req.body;

  const query = `
    INSERT INTO recruiter_form (
      organization_name, website_url, email, opportunity_type,
contact_person_name,
      contact_number, alternate_contact_number, mode_of_drive, country,
state,
      city, location, preferred_dates, required_skills, job_description,
      number_of_rounds, rounds, package_offered, role_offered,
eligibility_criteria,
      documents_required, specific_requirements, accommodation_provided,
      travel_reimbursement, other_benefits, terms_and_conditions
    ) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
  `;

  const values = [
    formData.organizationName,
    formData.websiteUrl,
    formData.email,
    formData.opportunityType,
    formData.contactPersonName,
    formData.contactNumber,
    formData.alternateContactNumber,
    formData.modeOfDrive,
    formData.country,
    formData.state,
    formData.city,
    formData.location,
    JSON.stringify(formData.preferredDates),
    formData.requiredSkills,
    formData.jobDescription,
    formData.numberOfRounds,
    JSON.stringify(formData.rounds),
    formData.packageOffered,
    formData.roleOffered,
    formData.eligibilityCriteria,
    JSON.stringify(formData.documentsRequired),
```

```
        formData.specificRequirements,
        formData.accommodationProvided,
        formData.travelReimbursement,
        formData.otherBenefits,
        formData.termsAndConditions
    ];

    try {
        const [result] = await pool.query(query, values);
        res.status(200).json({ message: "Form submitted successfully", id:
result.insertId });
    } catch (error) {
        console.error("Database error while submitting Recruiters form:", error);
        res.status(500).json({ error: "Database error while submitting Recruiters
form" });
    }
};

// Get all recruiter forms
export const getRecruiterForms = async (req, res) => {
    try {
        const [forms] = await pool.query("SELECT * FROM recruiter_form");
        res.json(forms);
    } catch (error) {
        console.error("Error fetching recruiter forms:", error);
        res.status(500).json({ error: "Database error" });
    }
};
// Get a single recruiter form by ID
export const getRecruiterFormById = async (req, res) => {
    const { id } = req.params;

    try {
        const [form] = await pool.query("SELECT * FROM recruiter_form
WHERE id = ?", [id]);
        if (form.length === 0) {
            return res.status(404).json({ error: " recruiters Form not found" });
        }
        res.json(form[0]);
    } catch (error) {
        console.error("Error fetching recruiter form:", error);
```

```
      res.status(500).json({ error: "Database error while fetching recruiters
submited form" });
   }
};

// Move recruiter form data to companies table
export const moveToCompanies = async (req, res) => {
   const { id } = req.params;

   try {
      // Step 1: Fetch the recruiter form data
      const [form] = await pool.query("SELECT * FROM recruiter_form
WHERE id = ?", [id]);
      if (form.length === 0) {
         return res.status(404).json({ error: "Error while fetching rescuiter
Form using id" });
      }

      const { organization_name, website_url, email } = form[0];

      // Step 2: Check if the company already exists in the companies table
      const [existingCompany] = await pool.query(
         "SELECT * FROM companies WHERE company_name = ?",
         [organization_name]
      );

      if (existingCompany.length > 0) {
         return res.status(400).json({ error: "Company already exists" });
      }

      // Step 3: Insert into companies table
      const insertQuery = `
         INSERT INTO companies (company_name, website, contact_email)
         VALUES (?, ?, ?)
      `;
      const values = [organization_name, website_url, email];

      const [result] = await pool.query(insertQuery, values);

      res.status(200).json({
         message: "Data moved to companies table successfully",
         companyId: result.insertId,
```

```
        });
    } catch (error) {
        console.error("Error moving data to companies:", error);
        res.status(500).json({ error: "Database error" });
    }
};

// Delete a recruiter form
export const deleteRecruiterForm = async (req, res) => {
    const { id } = req.params;

    try {
        await pool.query("DELETE FROM recruiter_form WHERE id = ?",
[id]);
        res.status(200).json({ message: "Form deleted successfully" });
    } catch (error) {
        console.error("Error deleting form:", error);
        res.status(500).json({ error: "Database error" });
    }
};
// index.jx file code
import dotenv from 'dotenv'
dotenv.config()
import express from 'express';
import studentRoutes from "./src/routes/student.routes.js";
import offersRoutes from "./src/routes/offers.routes.js";
import CompaniesRouters from './src/routes/companies.routes.js';
import PlacementDriveRoutes from './src/routes/drives.routes.js';
import recruiterFormRoutes from './src/routes/recruiters_forms.routes.js';
import internshipRoutes from './src/routes/Internship.routes.js';
import coursesRoutes from './src/routes/courses.routes.js';
import ResoursesRoutes from './src/routes/resources.routes.js';
import hackathonsRoutes from './src/routes/hackathons.routes.js';

const app = express()
const port = process.env.PORT || 3000; // ☑ Use environment variable or
default port

app.use(express.json()); // Middleware to parse JSON
app.use("/api/students", studentRoutes);
app.use("/api/offers", offersRoutes);
app.use("/api/companies", CompaniesRouters);
```

```
app.use("/api/drives", PlacementDriveRoutes);
app.use("/api/recruiter-forms", recruiterFormRoutes);
app.use("/api/internships", internshipRoutes);
app.use("/api/courses", coursesRoutes);
app.use("/api/resources", ResoursesRoutes);
app.use("/api/hackathons", hackathonsRoutes);

app.listen(process.env.PORT, () => {
  console.log(`Example app listening on port ${port}`)
});
```

Frontend Configuration Code:

```
import React, { Fragment, useContext, useEffect, useRef, useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { Dialog, Transition } from '@headlessui/react';
import { RxCross2 } from 'react-icons/rx';
import myContext from '../../context/data/myContext';

function Navbar() {
    const context = useContext(myContext);
    const [open, setOpen] = useState(false);
    const user = JSON.parse(localStorage.getItem('user'));

    useEffect(() => {
        window.scrollTo(0, 0);
    }, []);

    const Logout = () => {
        localStorage.clear('user');
        window.location.href = "/";
    };

    const currentPath = location.pathname;
    const [isDropdownOpen, setIsDropdownOpen] = useState(false);
    const [isMobileDropdownOpen, setIsMobileDropdownOpen] = useState(false)
  const dropdownRef = useRef(null);
```

```
  const toggleDropdown = () => {
    setIsDropdownOpen(!isDropdownOpen);
  };

  useEffect(() => {
    const handleClickOutside = (event) => {
      if (dropdownRef.current && !dropdownRef.current.contains(event.target)) {
        setIsDropdownOpen(false);
      }
    };

    document.addEventListener("mousedown", handleClickOutside);
    return () => {
      document.removeEventListener("mousedown", handleClickOutside);
    };
  }, []);
  // Toggle function for mobile dropdown
  const toggleMobileDropdown = () => {
    setIsMobileDropdownOpen(!isMobileDropdownOpen);
  };

    return (
      <div style={{ width: "100%" }} className="bg-white sticky top-0 z-50 w-
full">
        {/* Mobile Navigation */}
        <Transition.Root show={open} as={Fragment}>
          <Dialog as="div" className="relative z-40 lg:hidden"
onClose={setOpen}>
            <Transition.Child
              as={Fragment}
              enter="transition-opacity ease-linear duration-300"
              enterFrom="opacity-0"
              enterTo="opacity-100"
              leave="transition-opacity ease-linear duration-300"
              leaveFrom="opacity-100"
              leaveTo="opacity-0"
```

```
          >
            <div className="fixed inset-0 bg-black bg-opacity-25" />
          </Transition.Child>

          <div className="fixed inset-0 z-40 flex">
            <Transition.Child
              as={Fragment}
              enter="transition ease-in-out duration-300 transform"
              enterFrom="-translate-x-full"
              enterTo="translate-x-0"
              leave="transition ease-in-out duration-300 transform"
              leaveFrom="translate-x-0"
              leaveTo="-translate-x-full"
            >
              <Dialog.Panel className="relative flex w-full max-w-xs flex-col
overflow-y-auto bg-white pb-12 shadow-xl">
                {/* Close Button */}
                <div className="flex px-4 pb-2 pt-28">
                  <button
                    type="button"
                    className="-m-2 inline-flex items-center justify-center
rounded-md p-2 text-gray-400"
                    onClick={() => setOpen(false)}
                  >
                    <span className="sr-only">Close menu</span>
                    <RxCross2 />
                  </button>
                </div>

                {/* Sidebar Links */}
                <div className="flex flex-col space-y-6 border-t border-gray
200 px-4 py-6">
                  <Link to="/" className="text-sm font-medium text-gray-
900">Home</Link>
                  <Link to="/about" className="text-sm font-medium text-
gray-900">About</Link>
```

```
                        <Link to="/opportunities" className="text-sm font-
medium text-gray-900">Explore</Link>
                        <Link to="/recruiters" className="text-sm font-medium
text-gray-900">Recruiters</Link>
                        {/* Admin Panel Dropdown */}
                        <div className="flex flex-col space-y-2">
                        <button
                          onClick={toggleMobileDropdown}
                          className="flex items-center justify-between text-sm
font-medium text-gray-900"
                        >
                          <span>Admin Panel</span>
                        </button>
                        {/* Dropdown Menu */}
                        {isMobileDropdownOpen && (
                          <div className="pl-4 space-y-2">
                          <Link
                            to="/admin/dashboard"
                            className="block text-sm text-gray-700 hover:text-
blue-700"
                          >
                            Dashboard
                          </Link>
                          <Link
                            to="/admin/companies"
                            className="block text-sm text-gray-700 hover:text-
blue-700"
                          >
                            Companies
                          </Link>
                          <Link
                            to="/admin/resources"
                            className="block text-sm text-gray-700 hover:text-
blue-700"
                          >
                            Resources
                          </Link>
```

```
                        </div>
                      )}
                      </div>
                      {user ? (
                        <a onClick={Logout} className="text-sm font-medium
text-gray-900 cursor-pointer">Logout</a>
                      ) : (
                        <Link to="/login" className="text-sm font-medium tex
gray-900">Login</Link>
                      )}
                    </div>
                  </Dialog.Panel>
                </Transition.Child>
              </div>
            </Dialog>
          </Transition.Root>

          {/* Main Header */}
          <header className="relative bg-white">
            <nav className="px-4 sm:px-6 lg:px-8 shadow-xl">
              <div className="flex h-16 items-center">
                {/* Mobile Menu Button */}
                <button type="button" className="lg:hidden" onClick={() =>
setOpen(true)}>
                  <span className="sr-only">Open menu</span>
                  <svg xmlns="http://www.w3.org/2000/svg" fill="none"
viewBox="0 0 24 24" strokeWidth={1.5} stroke="currentColor" className="w-
h-6">
                    <path strokeLinecap="round" strokeLinejoin="round"
d="M3.75 6.75h16.5M3.75 12h16.5m-16.5 5.25h16.5" />
                  </svg>
                </button>

                {/* Logo */}
                <div className="ml-4 flex lg:ml-0">
```

```
                <Link to="/" className="text-2xl font-bold text-black px-2 py-
1">EduCareer:<span className='font-medium text-gray-900'>Smart Learning
and Placement  Analytics Hub</span></Link>
              </div>

              {/* Desktop Navigation Links */}
              <div className="ml-auto hidden lg:flex lg:items-center lg:space-x
6">
                <Link to="/" className="text-sm font-medium text-gray-
900">Home</Link>
                <Link to="/about" className="text-sm font-medium text-gray-
900">About</Link>
                <Link to="/opportunities" className="text-sm font-medium
text-gray-900">Explore</Link>
                <Link to="/recruiters" className="text-sm font-medium text-
gray-900">Recruiters</Link>

                {/* Admin Panel Dropdown */}
                <div className="relative" ref={dropdownRef}>
                  <button
                  onClick={toggleDropdown}
                  className="flex items-center text-sm font-medium text-gray-
900 hover:text-blue-700 focus:outline-none"
                  >
                  Admin Panel

                  </button>
                  {/* Dropdown Menu */}
                  {isDropdownOpen && (
                  <div className="absolute right-0 mt-2 w-48 bg-white
rounded-lg shadow-lg py-2 z-50">
                    <Link
                    to="/admin/dashboard"
                    className="block px-4 py-2 text-sm text-gray-700 hover:bg
gray-100"
                    >
                    Dashboard
```

```
                        </Link>
                        <Link
                        to="/admin/companies"
                        className="block px-4 py-2 text-sm text-gray-700 hover:b
gray-100"
                        >
                        Companies
                        </Link>
                        <Link
                        to="/admin/resources"
                        className="block px-4 py-2 text-sm text-gray-700 hover:b
gray-100"
                        >
                        Resources
                        </Link>
                      </div>
                      )}
                    </div>
                    {user ? (
                      <a onClick={Logout} className="text-sm font-medium text-
gray-900 cursor-pointer">Logout</a>
                    ) : (
                      <Link to="/login" className="text-sm font-medium text-
gray-900">Login</Link>
                    )}
                  </div>
                </div>
              </nav>
            </header>
          </div>
    );
}

export default Navbar;

import React, { useState } from 'react'
import Navbar from '../../../components/navbar/Navbar'
```

```
import { addCompany } from '../../../services/companies.service';

function AddCompanyForm() {
   const [formData, setFormData] = useState({
      company_name: "",
      website: "",
      contact_email: "",
   });

   const handleChange = (e) => {
      const { name, value } = e.target;
      setFormData({ ...formData, [name]: value });
   };

   const handleSubmit = async (e) => {
      e.preventDefault();
      try {
         const response = await addCompany(formData);
         alert(response.message); // Show success message
         setFormData({ company_name: "", website: "", contact_email: "" }); //
Reset form
      } catch (error) {
         alert("Error adding company"); // Show error message
      }
   };

   return (
      <div>
         <Navbar/>
         <section className="p-4 ">
         <h1 className="text-2xl font-bold text-center my-4">Add New
company</h1>
         <div className="overflow-x-auto">
         <form onSubmit={handleSubmit}>
         <table className="min-w-full divide-y divide-gray-200 p-4">
               <thead className="bg-gradient-to-r from-blue-500 to-purple-600
text-white">
```

```
                <tr>
                  <th className="px-6 py-3 text-left text-sm font-
semibold">Company Name</th>
                  <th className="px-6 py-3 text-left text-sm font-
semibold">Website link</th>
                  <th className="px-6 py-3 text-left text-sm font-
semibold">Contact Email</th>
                  <th className="px-6 py-3 text-left text-sm font-
semibold">Industry</th>
                  <th className="px-6 py-3 text-left text-sm font-
semibold">Location</th>
                  <th className="px-6 py-3 text-left text-sm font-
semibold">Action</th>
                </tr>
              </thead>
              <tbody className='divide-y  bg-gray-100 divide-gray-200'>
              <tr className="hover:bg-gray-50 transition-colors">
                {/* Company Name */}
                <td className="px-6 py-4">
                <input
                  type="text"
                  name="company_name"
                  placeholder="name"
                  value={formData.company_name}
                  onChange={handleChange}
                  className="w-full p-2 border rounded"
                  required
                />
                </td>

                {/* Website */}
                <td className="px-6 py-4">
                <input
                  type="text"
                  name="website"
                  placeholder="link"
                  value={formData.website}
```

```
            onChange={handleChange}
            className="w-full p-2 border rounded"
          />
        </td>

        {/* Contact Email */}
        <td className="px-6 py-4">
          <input
            type="email"
            name="contact_email"
            placeholder="email"
            value={formData.contact_email}
            onChange={handleChange}
            className="w-full p-2 border rounded"
            required
          />
        </td>

        {/* Industry */}
        <td className="px-6 py-4">
          <input
            type="text"
            name="industry"
            placeholder="industry"
            value={formData.industry}
            onChange={handleChange}
            className="w-full p-2 border rounded"
            required
          />
        </td>

        {/* Location */}
        <td className="px-6 py-4">
          <input
            type="text"
            name="location"
            placeholder="location"
```

```
                    value={formData.location}
                    onChange={handleChange}
                    className="w-full p-2 border rounded"
                    required
                  />
                  </td>

                  {/* Submit Button */}
                  <td className="px-6 py-4">
                  <button type="submit" className="w-full p-2 bg-blue-500 text-
white rounded">
                    Add
                  </button>
                  </td>
                </tr>
                </tbody>
          </table>
          </form>
          </div>
          </section>

        </div>
     );
    }
export default AddCompanyForm

//App.js file
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
import Home from "./pages/home/Home";
import About from "./pages/about_us/About";
import AdminDashboard from "./pages/admin/Dashboard.admin";
import AdminCompanies from "./pages/admin/Companies.admin";
import Drives from "./pages/admin/companyPages/Drives.admin.pages";
import Offers from "./pages/admin/companyPages/Offers.admin.pages";
import Companies from
"./pages/admin/companyPages/Companies.admin.pages";
```

```
import NewCompanies from
"./pages/admin/companyPages/NewCompanies.admin.pages";
import TopCompanies from
"./pages/admin/companyPages/TopCompanies.admin.pages";
import CompanyInfo from
"./pages/admin/companyPages/NewCompanyInfo.admin.pages";
import Opportunities from "./pages/students/Opportunities";
import Result from "./pages/students/pages/Result.pages.students";
import StudentsResources from "./pages/admin/StudentResources.admin";
import CoursesResources from
"./pages/admin/resourcePages/Courses.resourcePages.admin";
import InternshipResources from
"./pages/admin/resourcePages/Internship.reourcePages.admin";
import Resourcelearning from
"./pages/admin/resourcePages/Resource.resourcePages.admin";
import AddCompanyForm from "./pages/admin/add/AddCompany.add.admin"
import AddStudentForm from "./pages/admin/add/AddStudent.add.admin";
import HackathonsResource from
"./pages/admin/resourcePages/Hackthons.resourcePages.admin";
import ConducrDrive from "./pages/recruiters/pages/ConducrDrive";
import SuccessForm from
"./pages/recruiters/pages/SuccessForm.pages.recruiters";
import RecruiterAnalysis from
"./pages/recruiters/pages/RecruiterAnalysis.pages.recruiters.jsx";
import RecruitersStudentData from
"./pages/recruiters/pages/RecruitersStudentData.pages.recruiters";
import RecruitersHome from
"./pages/recruiters/pages/RecruitersHome.pages.recruiters";
import RecruiterTool from
"./pages/recruiters/pages/RecruiterTool.pages.recruiters";

function App() {
 return (
  <Router>
   <Routes>
    <Route path="/" element={<Home/>} />
    <Route path="/admin/dashboard" element={<AdminDashboard />} />
```

```
      <Route path="/admin/companies" element={<AdminCompanies />} />
      <Route path="/about" element={<About />} />
      <Route path="/opportunities" element={<Opportunities />} />
      <Route path="/opportunities/result" element={<Result />} />
      <Route path="/recruiters" element={<RecruitersHome/>} />
      <Route path="/recruiters/drive" element={<ConducrDrive/>} />
      <Route path="/recruiters/success" element={<SuccessForm/>} />
      <Route path="/recruiters/analysis" element={<RecruiterAnalysis/>} />
      <Route path="/recruiters/allstudent-detail"
element={<RecruitersStudentData/>} />
      <Route path="/recruiters/tool" element={<RecruiterTool/>} />
      <Route path="/admin/companies/drives" element={<Drives/>} />
      <Route path="/admin/companies/offers" element={<Offers/>} />
      <Route path="/admin/companies/all" element={<Companies/>} />
      <Route path="/admin/companies/new" element={<NewCcompanies/>} />
      <Route path="/admin/companies/new/info/:id" element={<CompanyInfo/>}
/>
      <Route path="/admin/companies/top" element={<TopCompanies/>} />
      <Route path="/admin/resources" element={<StudentsResources/>} />
      <Route path="/admin/addstudent" element={<AddStudentForm/>} />
      <Route path="/admin/addcompany" element={<AddCompanyForm/>} />
      <Route path="/admin/resources/internships"
element={<InternshipResources/>} />
      <Route path="/admin/resources/hackathons"
element={<HackathonsResource/>} />
      <Route path="/admin/resources/references" element={<Resourcelearning/>
/>
      <Route path="/admin/resources/courses" element={<CoursesResources/>} /
    </Routes>
  </Router>
 );
}

export default App
```

APPENDIX II

OUTPUT SCREENSHOTS

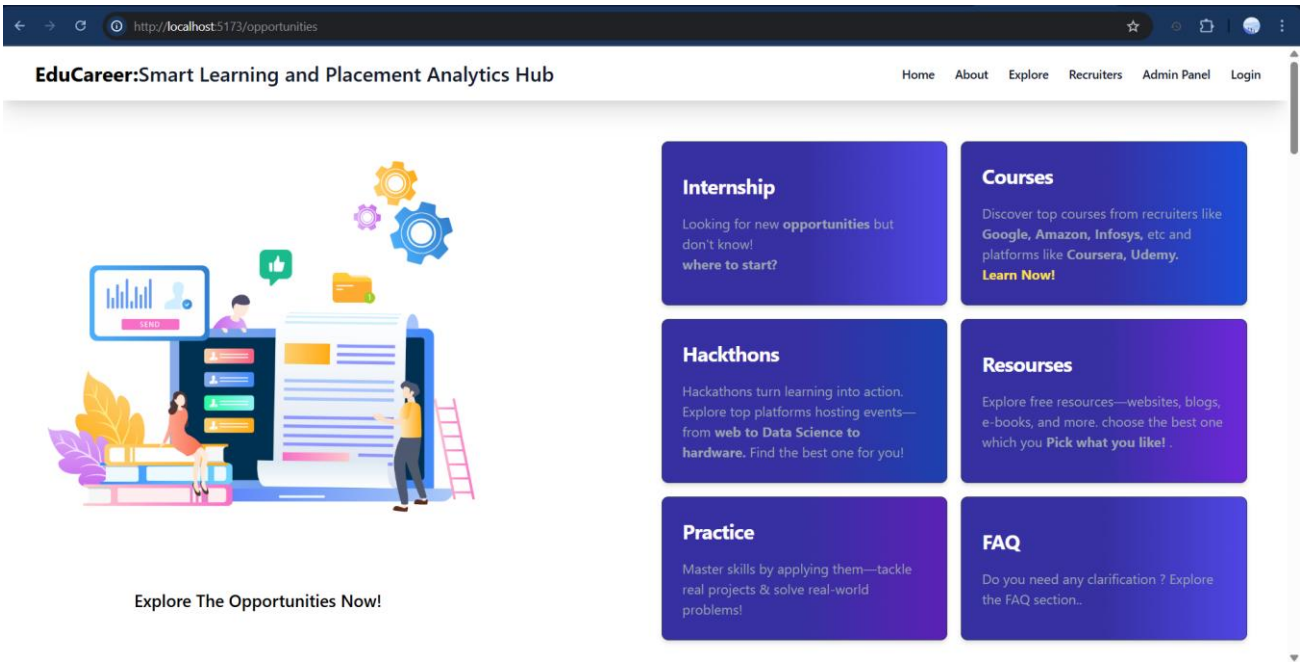


Fig 1: Student interface Hero Section

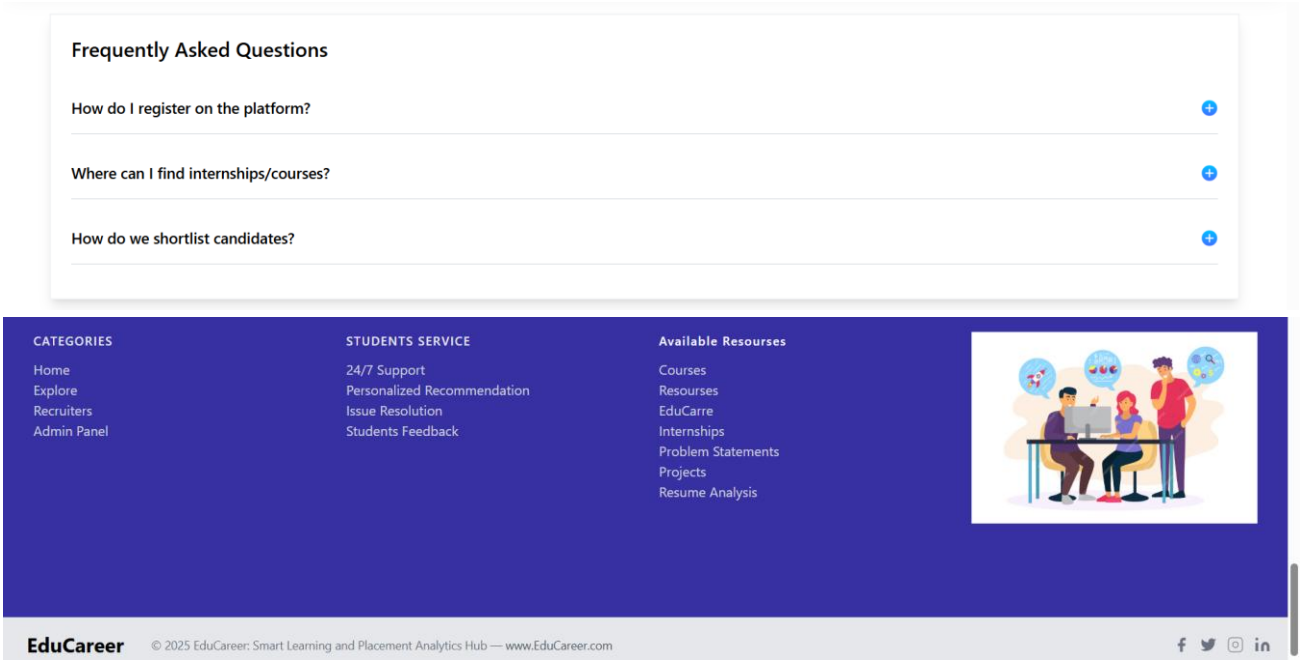


Fig 2: Student Interface FAQ Section and Footer

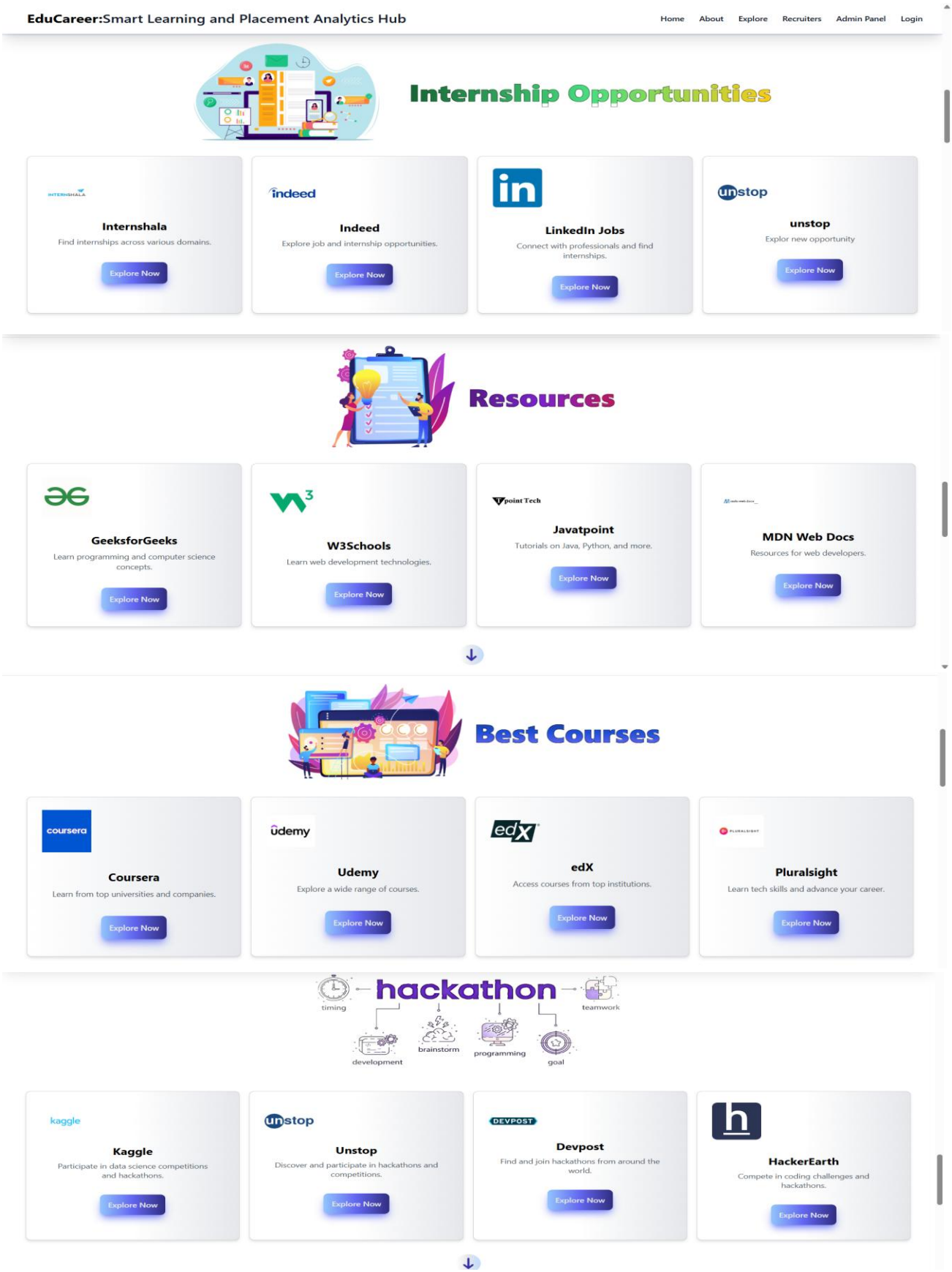


Fig 3: Student interface with all Courses, Internship, Resources, and Hackathons.

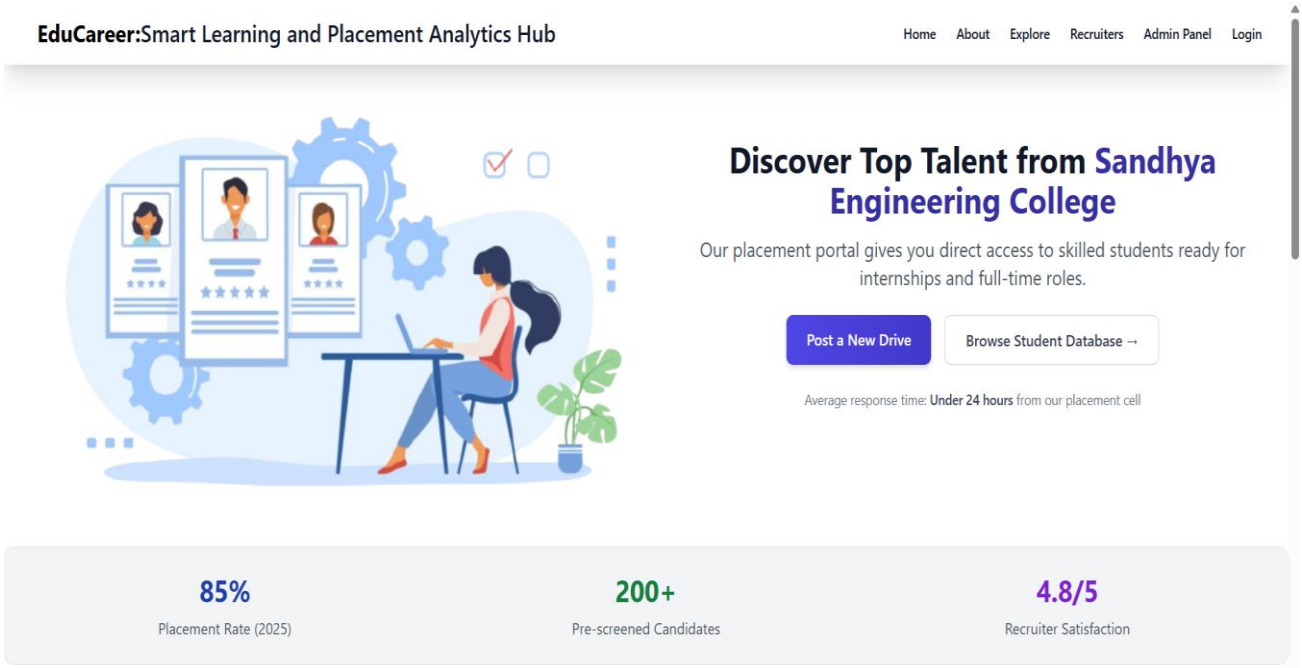


Fig 4: Recruiter Interface Hero Section

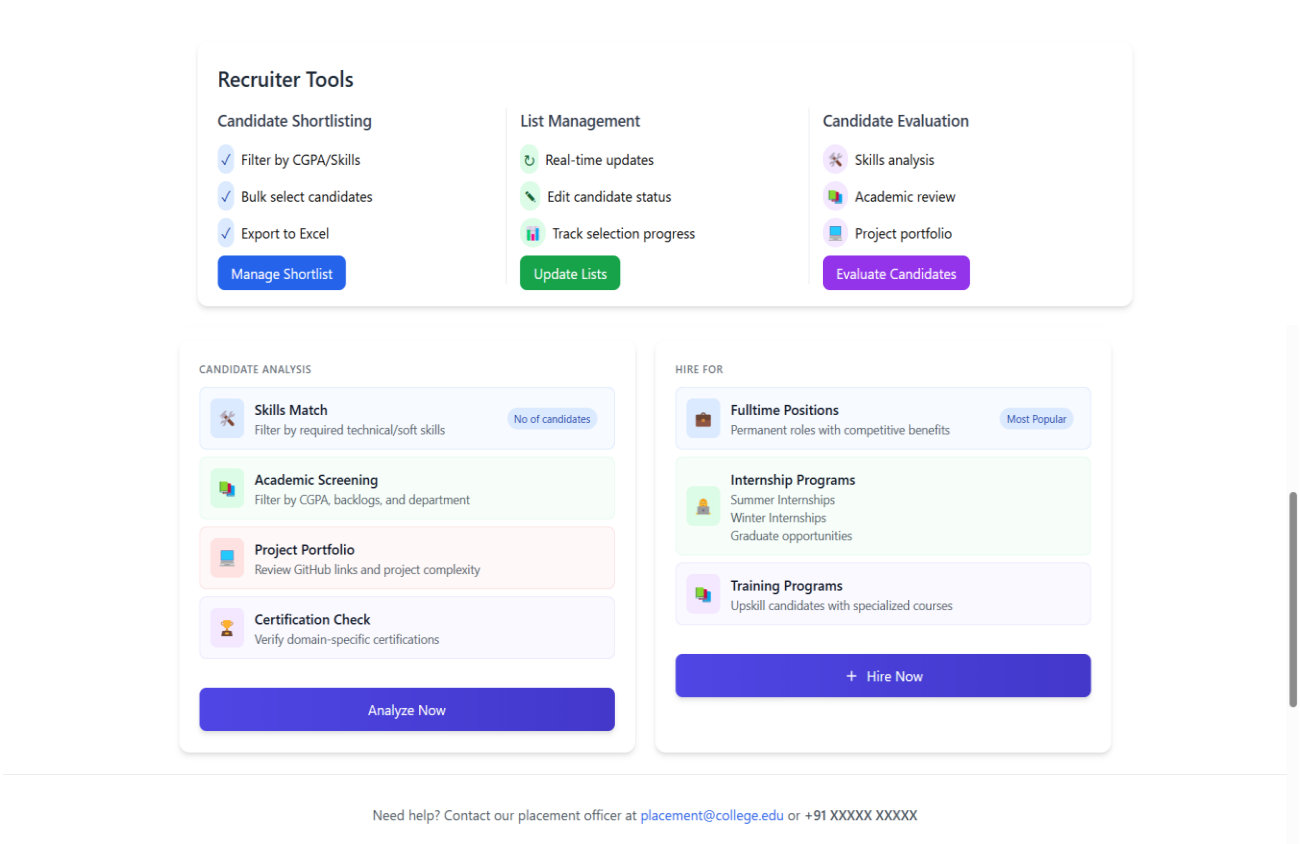


Fig 5: Recruiter Interface Body Section

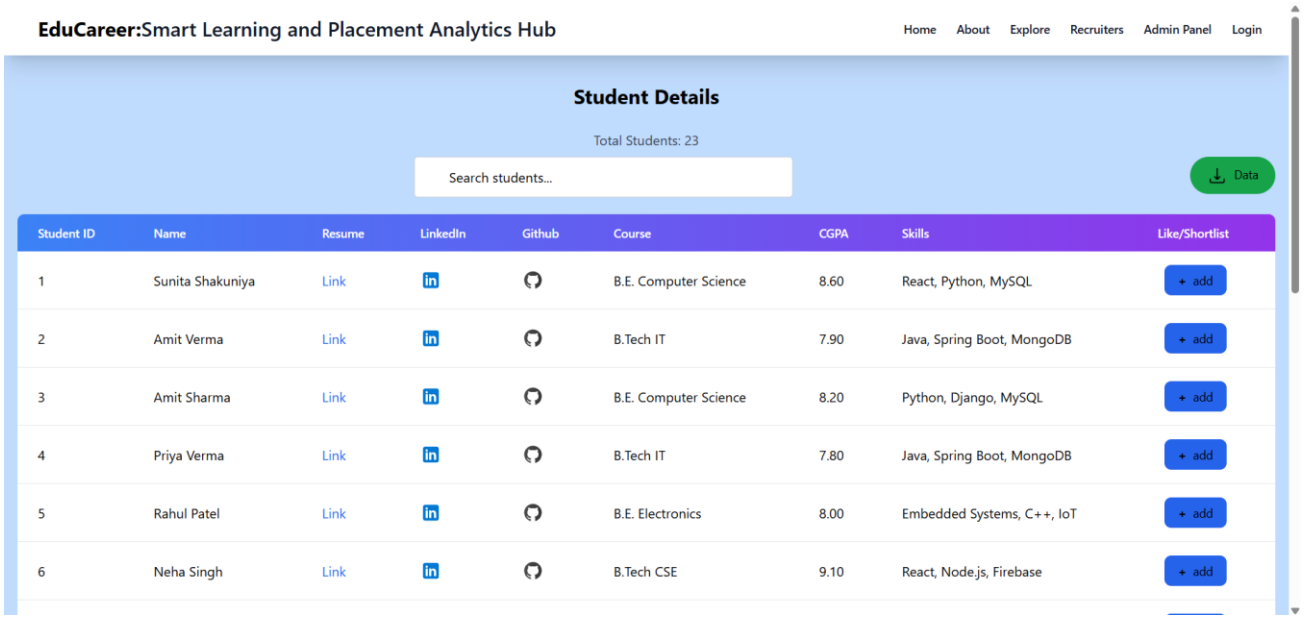


Fig 6: Recruiter Interface Access to Student Database

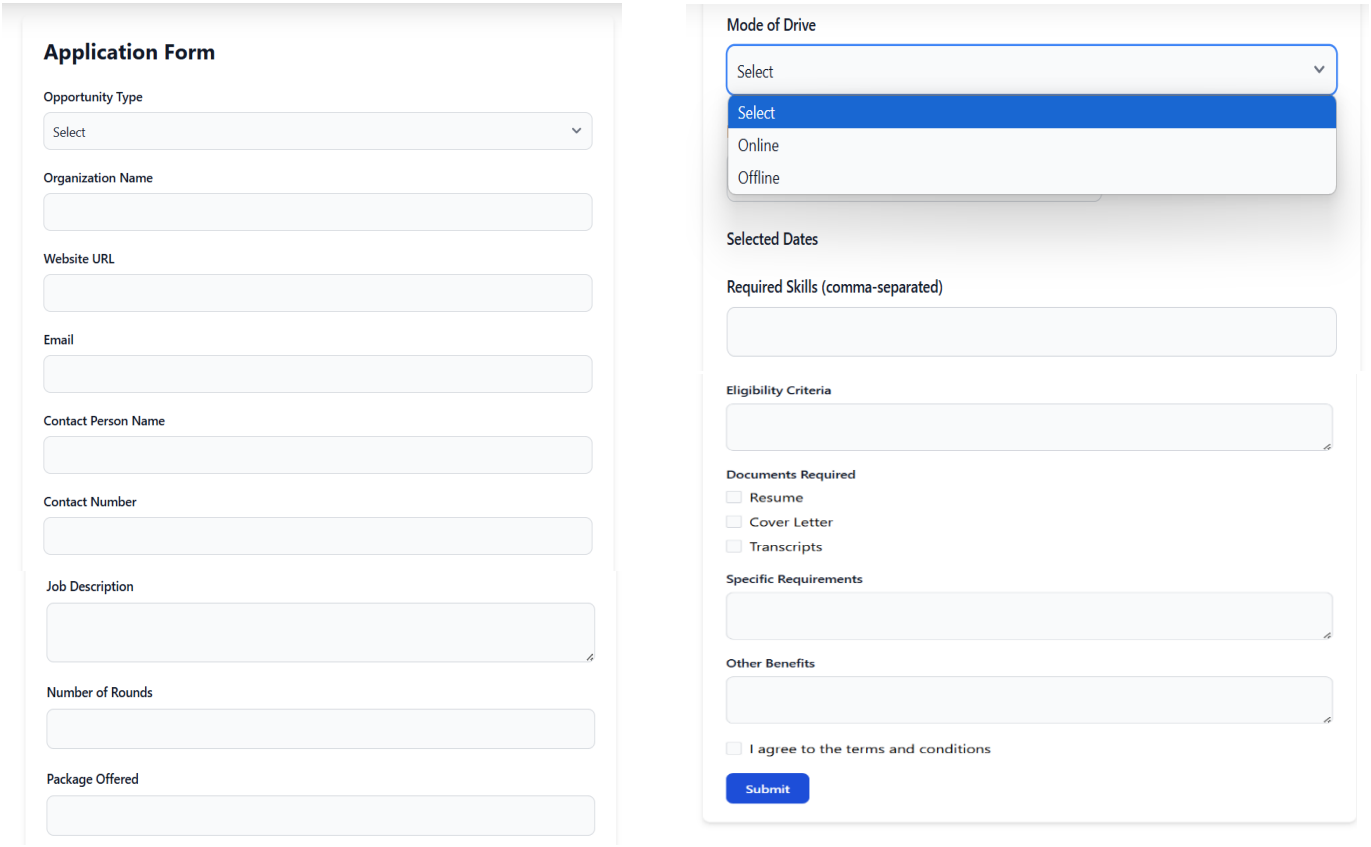


Fig 7: Recruiter Application Form

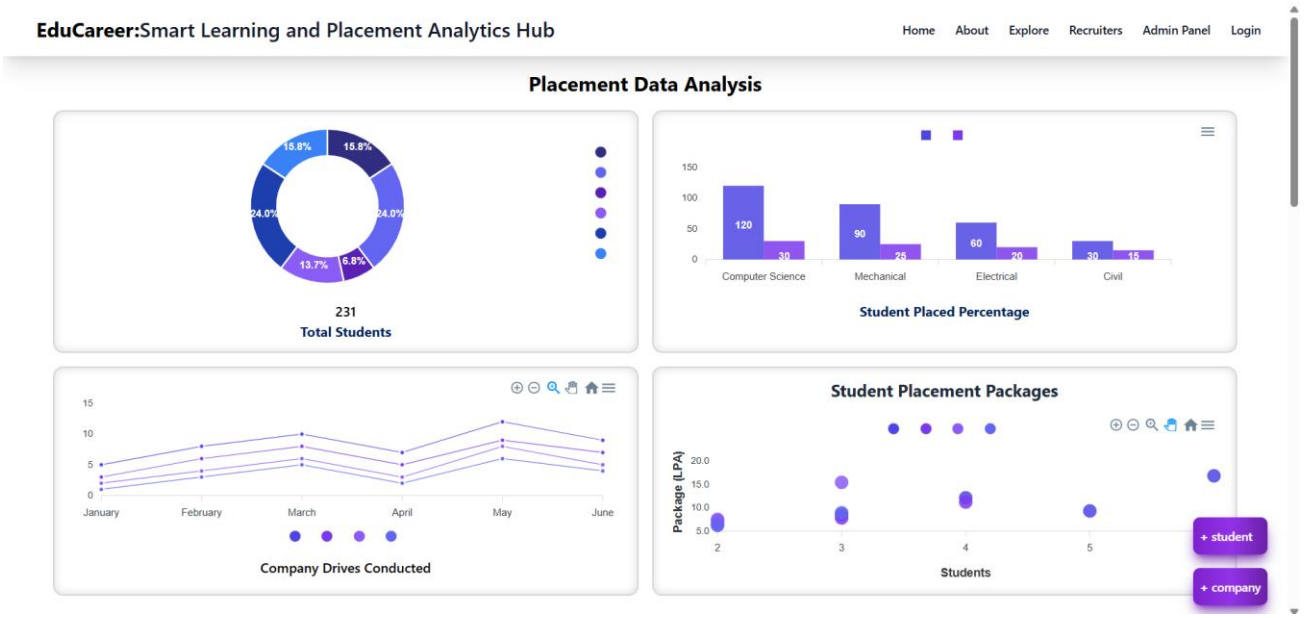


Fig 8: Admin Dashboard Interface – Data Analysis

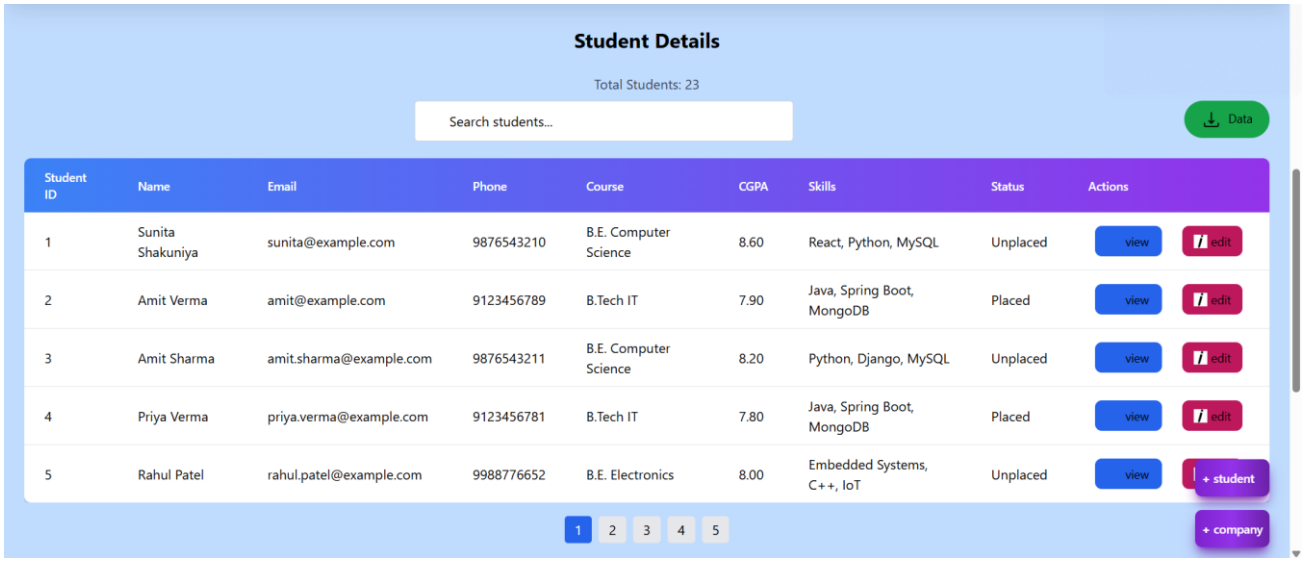


Fig 9: Admin Dashboard Interface –Student Data Access with filtering, sorting and searching function with csv download option.

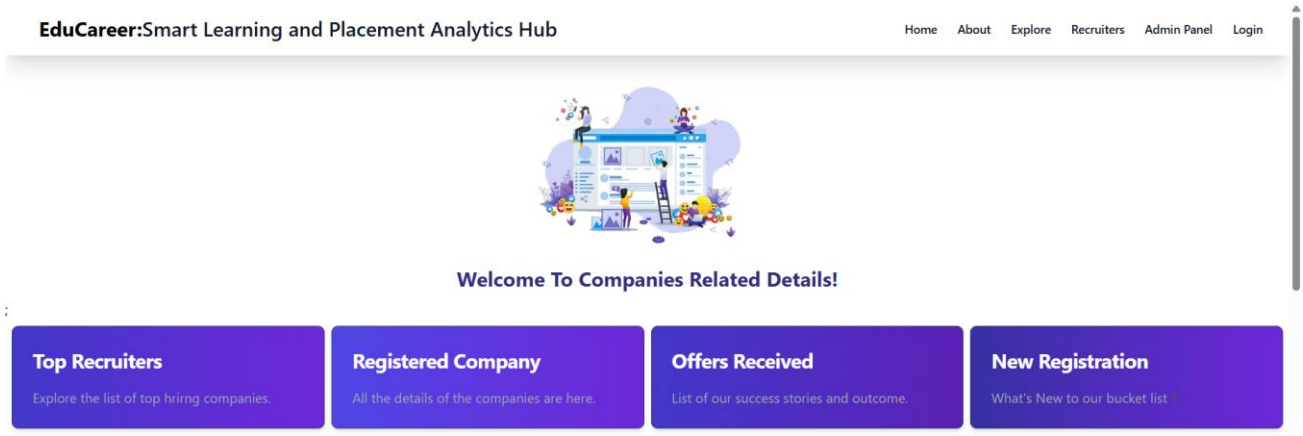


Fig 10: Admin Dashboard Interface – Companies Data Access and Modify Section

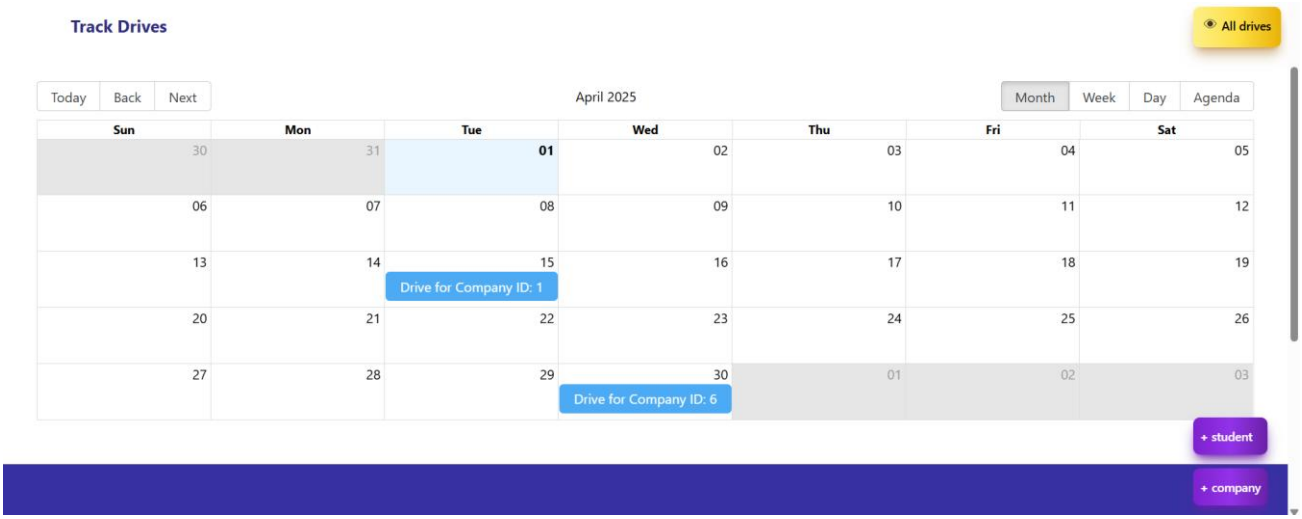


Fig 11: Admin Dashboard Interface – Drive Schedule Calendar view

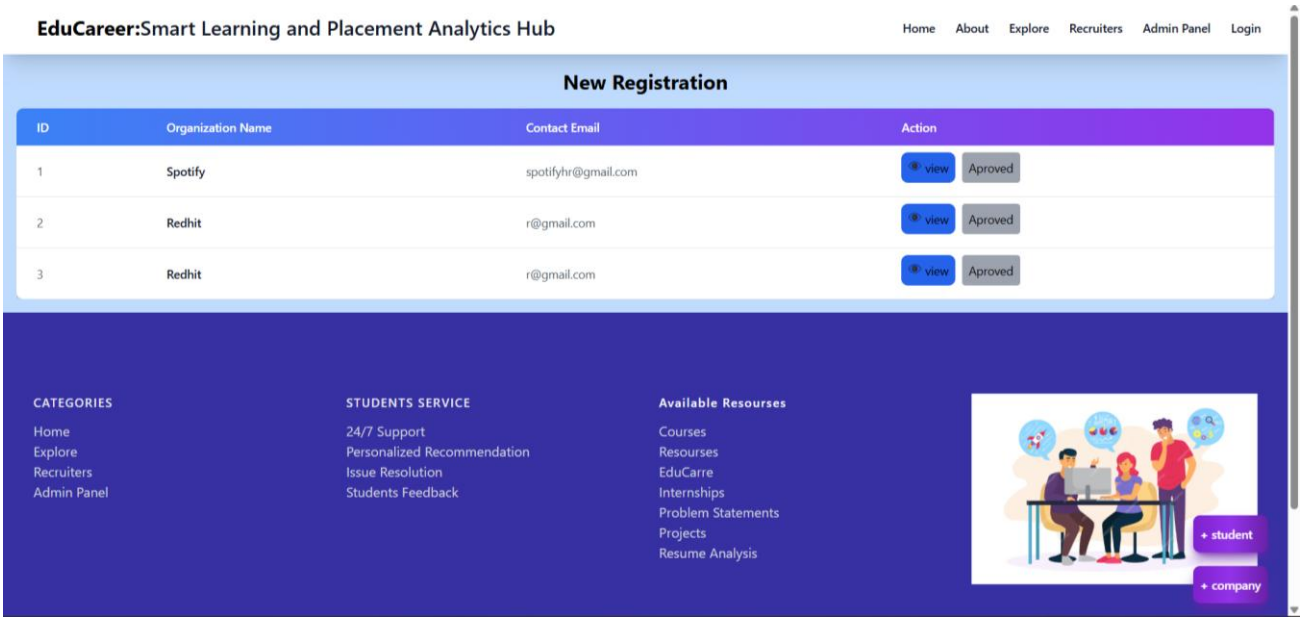


Fig 12: Admin Dashboard Interface – New Company Registration data

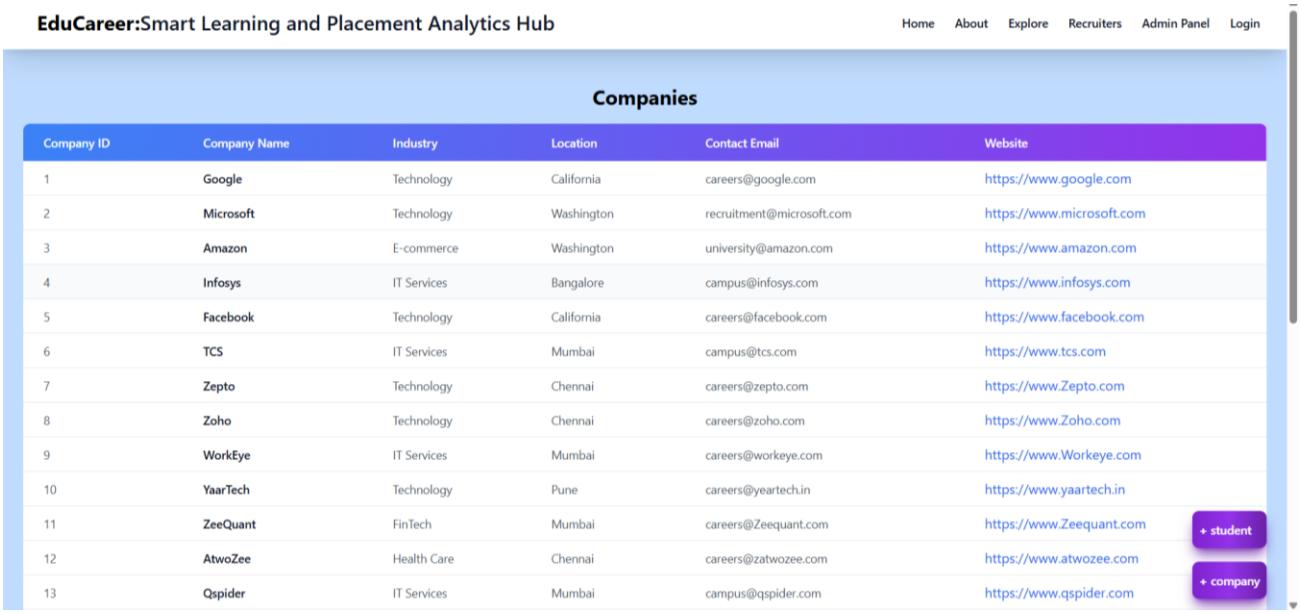


Fig 13: Admin Dashboard Interface – All Approved Companies

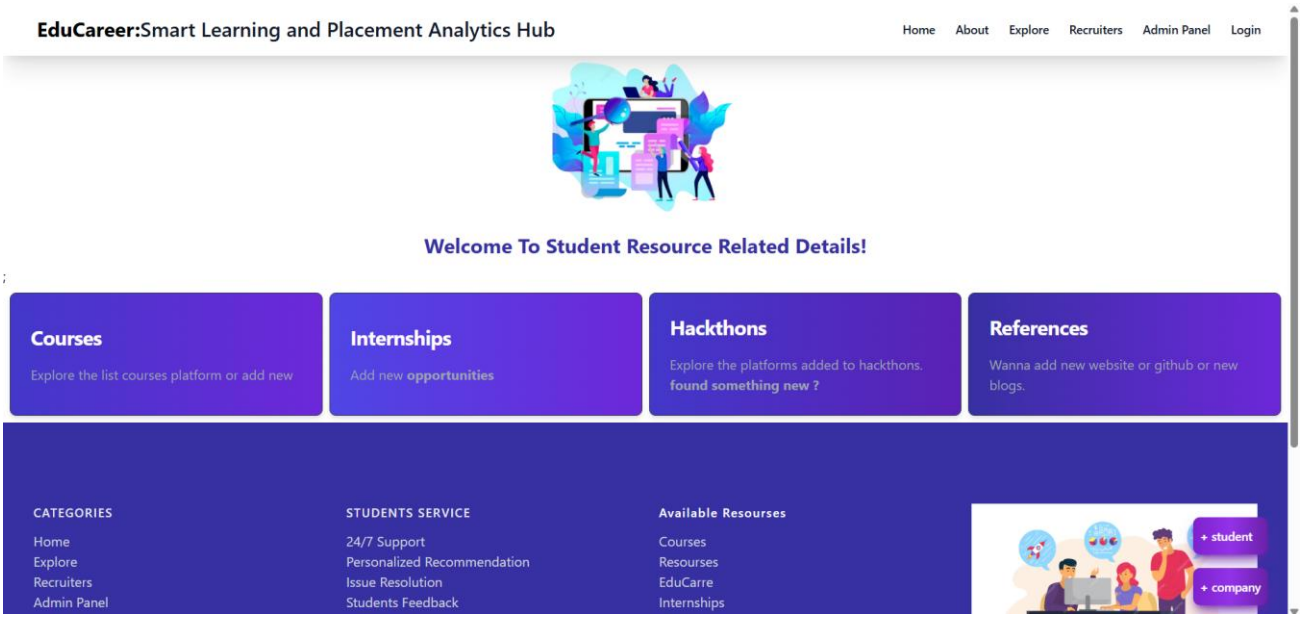


Fig 14: Admin Dashboard Interface – Resources Data Access

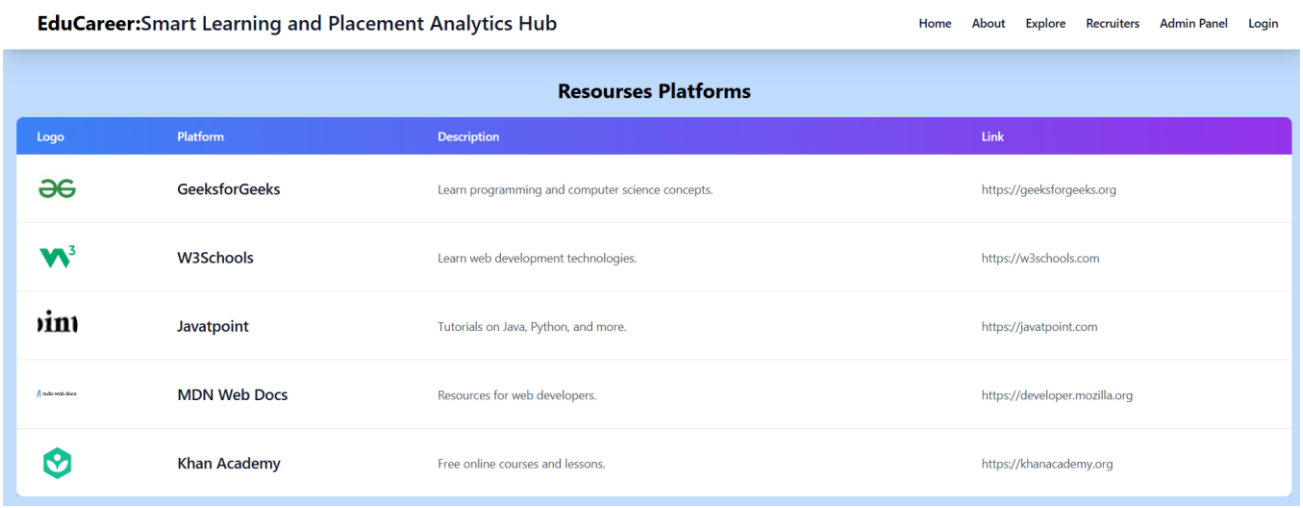


Fig 15: Admin Dashboard Interface – Learning Resources

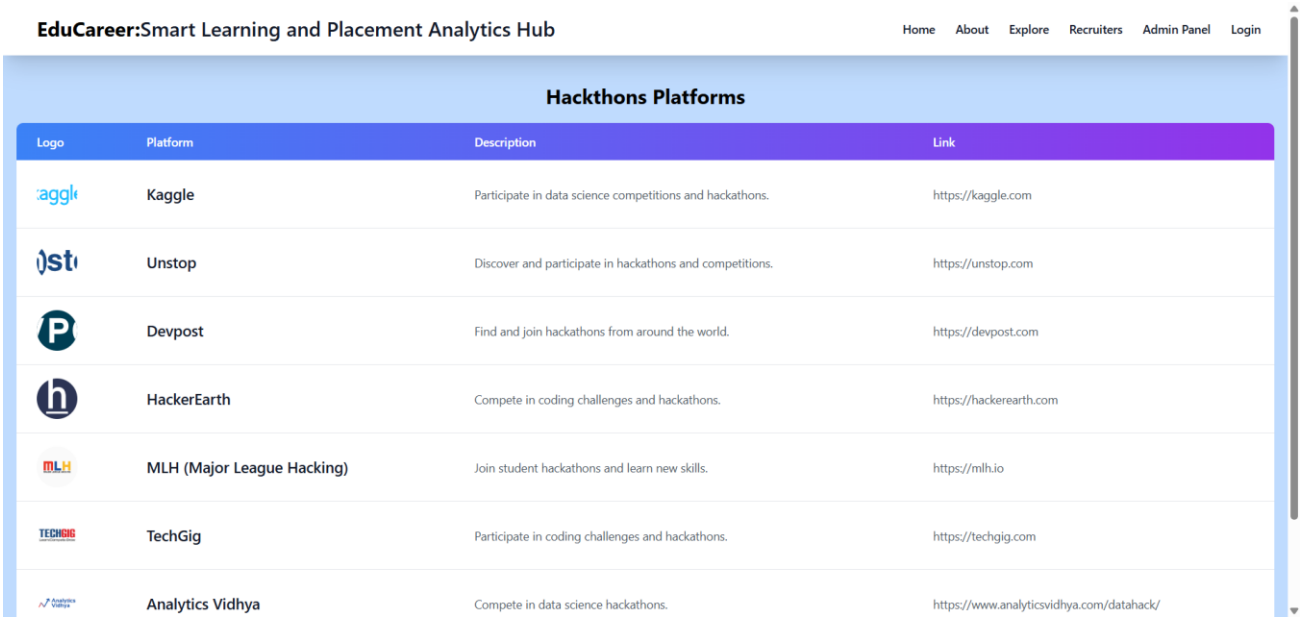


Fig 16: Admin Dashboard Interface –Hackathon Platform

REFERENCES

[1] Monika Tripathi, Dimpal Jain, Khushboo Sharma, Anuradha, Daulatram, “Web Development Framework”, International Journal of Advanced Research in Science, Communication and Technology, vol. 4, issue 1, December 2024.

[2] Ivan Suster, Tamara Ranisavljevic, “Optimization of MySQL Database”, Journal of Process Management and New Technologies, vol. 11, issue 1-2, pp. 141-151, 2023.

[3] Wen-Tin Lee, Chih-Hsien Chen, “Agile Software Development and Reuse Approach with Scrum and Software Product Line Engineering”, Electronics, vol. 12, issue 15, 2023.

[4] Mahesh Elango, “UI/UX Design: Next Generation Perspectives”, Technical Report, August 2024.

[5] Michael Adelusola, “Efficient Model Deployment Strategies for LLMs in Web Applications”, Research Article, June 2023.

[6] Yash Jani, “Optimizing Database Performance for Large-Scale Enterprise Applications”, International Journal of Science and Research (IJSR), vol. 11, issue 10, pp. 1394-1396, 2022.

[7] Gokul Pandy, Vigneshwaran Jagadeesan Pugazhenthi, Aravindhan Murugan, “Advances in Software Testing in 2024: Experimental Insights, Frameworks, and Future Directions”, International Journal of Advanced Research in Computer and Communication Engineering, vol. 13, issue 11, pp. 41-44, 2024.