IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429

Volume 6, Issue 3, January 2026

Impact Factor: 7.67

College Placement System Using Python

Pavan Subhash Dighe[1], Shivam Dnyaneshwar Deshmukh[2], Omsing Chainsing Sulane[3]
Prof. N. S. Kharatmal[4]
Student, Computer Science and Engineering[1.2.3]
Lecture, Computer Science and Engineering[4]
Matsyodari Shikshan Sanstha College of Engineering and polytechnic, Jalna,India
[1]dighepavan555@gmail.com, [2]shivamdeshmukh1188@gmail.com, [3]chensingsulane@gmail.com,
nanditakhartmal27@gmail.com[4]

Abstract: *The College Placement System is a software application developed using Python to streamline and automate the placement process in educational institutions. The system aims to reduce manual effort, improve accuracy, and enhance communication between students, placement officers, and recruiting companies. It provides a centralized platform to manage student records, company details, eligibility criteria, and placement status efficiently. The application allows students to register their academic and personal information, view eligible job opportunities, and apply for companies based on predefined criteria such as CGPA, branch, and skills. The system also facilitates automated notifications, status tracking of applications, and report generation, reducing manual effort and minimizing errors. By leveraging Python's simplicity and powerful libraries, the system offers an efficient, scalable, and secure solution to enhance the overall placement process in educational institutions. Placement officers can manage company profiles, schedule interviews, track student participation, and generate placement reports. Recruiters can access eligible student data and update selection results. Overall, the College Placement System not only reduces administrative effort but also improves transparency, accessibility, and efficiency in managing campus placements, contributing significantly to the career development of students*

Keywords: College Placement System, Python Programming, Student Data Management, Recruitment Automation, Placement Portal, Database Integration, Resume Management, Eligibility Filtering, Company–Student Matching, Web-Based Application, Backend Development, Data Validation, Secure Authentication, Reporting and Analytics

I. INTRODUCTION

The College Placement System is an application designed to simplify and automate the campus recruitment process. Traditionally, placement activities involve manual record keeping, eligibility verification, and coordination between students, placement officers, and recruiters. To overcome these challenges, a computerized system is essential. This system serves as a centralized platform where students can register, maintain their profiles, and apply for suitable job opportunities. Administrators can efficiently manage student records, company profiles, and placement events, while companies can post job openings and monitor applicant progress. By leveraging Python's simplicity, versatility, and powerful libraries, the system ensures smooth operation, enhanced data security, and user-friendly interfaces. Additionally, features such as automated notifications, real-time updates, and report generation reduce manual effort and improve overall efficiency.

Using Python as the core programming language, the College Placement System provides an efficient and user-friendly platform for managing placement-related data. The system stores student details, academic records, and company information in a centralized database. Python's flexibility and database support enable faster processing, improved accuracy, and easy maintenance.

Scope:

The scope of the College Placement System includes:

- Managing student registration, academic details, and skill information.
- Storing and updating company profiles and job requirements.
- Automatically checking student eligibility based on predefined criteria.

The system can be expanded in the future to include features such as resume upload, email notifications, analytics dashboards, and integration with online assessment platforms.

II. MOTIVATION

The motivation for developing the College Placement System arises from the challenges faced in traditional placement management, such as data inconsistency, time-consuming processes, and lack of transparency. Manual systems often make it difficult to track student eligibility and placement progress efficiently.

By developing this system using Python, the project aims to demonstrate how software solutions can simplify real-world institutional problems. It also provides practical exposure to programming, database management, and system design, making it an ideal academic project.

III. LITERATURE SURVEY

The placement process in colleges involves managing large volumes of student data, company requirements, and recruitment outcomes. Traditional placement management methods rely heavily on manual record-keeping and spread sheets, which are time-consuming and prone to errors.

1) Automation of Placement Activities: Previous studies highlight the need to automate manual placement processes such as student registration, eligibility checking, and interview scheduling. Python-based systems reduce human effort and minimize errors. Automation improves efficiency and ensures faster processing of placement-related tasks.

2) Centralized Student Data Management: Literature emphasizes maintaining a centralized database for student academic and personal information. Python applications integrated with databases allow secure storage and quick retrieval of student records. Centralization avoids data duplication and ensures consistency. It also simplifies data updates and reporting.

3) Eligibility Filtering and Shortlisting: Many existing systems focus on automating eligibility criteria based on academic performance and skill requirements. Python enables logical filtering and dynamic shortlisting of candidates. This reduces manual screening time for recruiters. Accurate filtering improves the quality of shortlisted candidates.

4) Company and student interaction platform: Research highlights the importance of an interactive platform connecting companies and students. Python-based web systems allow recruiters to post job requirements and students to apply online. This improves transparency and accessibility. It also ensures real-time communication between stakeholders.

5) Secure Authentication and Role Management: Literature stresses the need for secure login systems with role-based access. Python supports authentication mechanisms for students, recruiters, and administrators. Proper access control prevents unauthorized data usage. Security enhances trust and system reliability.

6) Resume Management and Profile Analysis: Studies show that digital resume handling improves placement efficiency. Python systems allow students to upload, update, and manage resumes. Recruiters can easily review candidate profiles. This simplifies resume screening and improves hiring decisions.

7) Reporting and Performance Analytics: Existing research highlights the role of analytics in evaluating placement performance. Python enables generation of reports such as placement statistics and company-wise results. Analytical insights help institutions improve placement strategies. Data-driven decisions enhance overall system effectiveness.

IV. EXISTING MODEL

Many existing systems focus on automating eligibility criteria based on academic performance and skill requirements. Python enables logical filtering and dynamic shortlisting of candidates.

1) Manual and Semi-Automated Operations: In the existing model, many placement activities such as data entry and eligibility checking are partially manual. Python is sometimes used only for basic record handling.
2) Fragmented Data Storage: Existing placement systems often store student and company data in separate files or databases. Python scripts are used individually without proper integration.
3) Basic Eligibility Screening: The current model supports simple eligibility checks based on predefined academic criteria. Python logic is applied in a limited and static manner.
4) Limited User Interaction: Existing systems provide minimal interaction between students and recruiters. Python-based interfaces are often text-based or poorly designed.

V. METHODOLOGY

1. Hardware setup.

- Processor: Intel Core i3 or higher
- Hard Disk: 20 GB free storage
- Display: Standard monitor with minimum 1366×768 resolution

2. Software Installation.
Python, VS Code, Django or Flask, Linux

3. Database/Data Management
Use Python data structures or a database (like SQLite) to store and retrieve data securely. Implement CRUD operations (Create, Read, Update, Delete) for students and companies.

4. Testing and Validation
Test the system with sample data to verify correctness, handle errors, and ensure accurate placement results. Fix bugs and optimize performance before final deployment.

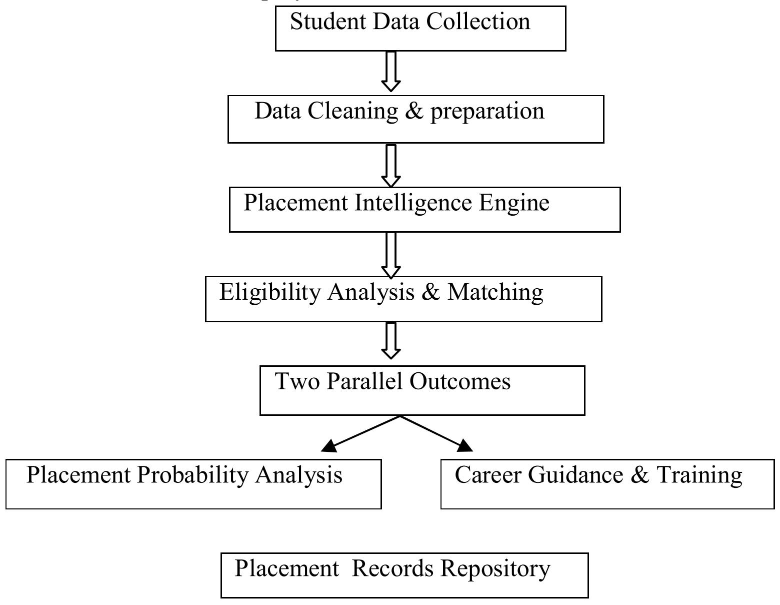
Student Data Collection
Data Cleaning & preparation
Placement Intelligence Engine
Eligibility Analysis & Matching
Two Parallel Outcomes
Placement Probability Analysis
Career Guidance & Training
Placement Records Repository

Fig 1. System Architecture.

IJARSCT

International Journal of Advanced Research in Science, Communication and Technology

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

ISSN: 2581-9429

Volume 6, Issue 3, January 2026

Impact Factor: 7.67

VI. ALGORITHM USED IN EXISTING SYSTEM AND PROPOSED SYSTEM

All category	Existing System Algorithms / Techniques	Limitations in Existing System	Proposed System Algorithms / Techniques (Python-Based)
Student Data Management	Manual records, Excel sheets	Data redundancy, human errors, difficult tracking	Centralized database using Python (MySQL/SQLite) with CRUD operations
Eligibility Checking	Manual shortlisting based on marks	Time-consuming, error-prone	Automated eligibility filtering using Python conditional logic
Resume Screening	Manual resume verification	Inconsistent evaluation, slow process	Resume parsing using Python (keyword-based matching)
Company Requirement Matching	Manual comparison of student profiles	Inefficient matching, missed candidates	Rule-based matching algorithm using Python
Placement Notification	Notice boards, emails	Delays, missed notifications	Automated email/SMS alerts using Python (SMTP APIs)
Student Performance Analysis	Basic percentage calculation	No predictive insights	Statistical analysis using Python (Pandas, Numpy)

VII. OUTPUT/RESULT AND DISCUSSION

ALL TYPES AND CATEGORY	Input / Source	Processing Tool / Module	Output / Results
System Initialization	Admin login credentials	Python Authentication Module	Secure login established; admin dashboard loaded successfully
Registration of the student	Personal data and academic data	Validation of python from Database Module	Student profile must be stored unique id
Company Registration	Company details and job criteria	Python CRUD Operations	Company profile added and verified successfully
Eligibility Evaluation	All of the students marks and skills	List -based Python Logic	Selected student sorted automatically
Performance Analytics	Student & placement records	Data analysis, (pandas, Numpy)	Department-wise and year-wise placement statistics
Report Generation	Placement summary data	Python report generator	Downloadable placement report created (PDF/CSV)

VIII. CONCLUSION

The College Placement System developed using Python provides an effective and structured solution to the challenges faced in traditional placement management processes. By automating student registration, eligibility verification, resume evaluation, and company matching, the system significantly reduces manual workload and operational delays. The integration of rule-based logic and data-driven techniques ensures fair and accurate shortlisting of candidates. The system also enhances communication through automated notifications and organized interview scheduling, minimizing conflicts and information gaps between students, companies, and placement administrators. The inclusion of analytical modules enables real-time monitoring of placement activities and offers meaningful insights through reports and visual representations.

REFERENCES

[1]. V. Kalsarpe, H. Ahire, A. More, S. Jadhav, and V. Walke, "College Campus Placement System Using Python,"
[2]. J. Antony Jerold, A. George Antony, S. Hariharan, L. Saranhariharajeyan, P. Sowkarthika, and N. Suguna, "Automated Placement Coordinator System with Automated Proctoring Assessment using Python Django,"
[3]. A. Banu and M. B. S. K., "A Research on Placement Management System.
[4]. A. Tyagi, P. Agrawal, P. Srivastav, P. Sagar, and A. Sharma, "College Placement Portal System.
[5]. H. Bhaskaran, R. Rakesh, R. S. S., K. S. K., and H. G., "Student Placement Management System.
[6]. "College Placement Management System".
[7]. K. Priyanga, A. Divakar, A. S. E. Vyshnavee, S. Ibrahim Basha, and D. Harsitha, "Smart and Automated College Placement System.
[8]. S. Kumar H. and S. V. R., "Online Training and Placement Management System.
[9]. V. Nageswara Rao and P. Dhanalakshmi, "Campus Placement Prediction using Machine Learning.