INDEX

ABSTRACT

1. Background:

The Campus Recruitment System is designed to streamline and automate the process of campus placements for educational institutions, companies, and students. Traditionally, this process has been manual, time-consuming, and prone to errors. The implementation of an automated system aims to address these challenges and provide a more efficient and transparent platform for all stakeholders.

2. Purpose:

The purpose of this project is to develop a comprehensive and user-friendly platform that facilitates the entire campus recruitment process. This includes registration and management of stakeholders (Admins, Companies, and Students), job postings, application submissions, and selection processes. The system aims to provide a seamless experience for both recruiters and students, enhancing the efficiency of placements.

3. Methods:

The project employs a systematic approach starting with requirement gathering and analysis. Frontend interfaces are developed using modern web technologies, while backend services are implemented server-side technologies. The system integrates database management for storing and retrieving user and job-related data securely.

4. Findings:

The Campus Recruitment System successfully automates the placement process, reducing manual effort and improving accuracy. The system allows Admins to manage user accounts to oversee student placements, Companies to view and recruit suitable candidates, and Students to apply for jobs and track their application status. The integration of notification systems ensures timely communication between all stakeholders.

5. Implications:

The implementation of this system has several implications. It enhances the efficiency of campus placements by providing a centralized platform for all stakeholders. For educational institutions, it reduces administrative burden and enhances the placement success rate. For companies, it provides access to a pool of qualified candidates. For students, it simplifies the job application process and increases transparency.

LIST OF FIGUREURES

CONTENTS	PAGE NUMBER

LIST OF TABLES

LIST OF TABLES

S.NO	ACRONYM	ABBREVATIONS
1	CRS	Campus Recruitment System
2	SWOT	Strengths, Weaknesses, Opportunities, and Threats
3	UI	User Interface
4	HTML	Hypertext Markup Language
5	CSS	Cascading Style Sheets
6	MYSQL	My Structured Query Language
7	UAT	User Acceptance Testing
8	RDBMS	Relational Database Management System
9	HTTP	HyperText Transfer Protocol
10	CRUD	Create, Read, Update, Delete
11	UML	Unified Modeling Language
12	DFD	Data Flow Diagram
13	CI	Continuous Integration
14	IDE	Integrated Development Environment
15	UT	Unit Testing
16	IT	Integration Testing
17	WB	White Box
18	BB	Black Box

CHAPTER - 1

INTRODUCTION

1. INTRODUCTION

1.1. PREAMBLE

The traditional campus recruitment process often relies on manual methods, leading to inefficiencies for students, , and college placement officers. This project aims to address these challenges by developing a web-application of campus recruitment system**.**

1.2. MOTIVATION FOR THE PROJECT

The current manual system for campus recruitment suffers from several drawbacks. These include:

- **Tedious record keeping and data retrieval:** Manually maintaining and searching through student and company data can be time-consuming and prone to errors.
- **Inefficient communication:** Coordinating between students, , and placement officers can be cumbersome and slow.
- **Limited accessibility:** Students may not have easy access to information about available job opportunities.

This project is motivated by the need for a more **efficient, streamlined, and accessible** campus recruitment process. By automating many of the manual tasks involved, the proposed system can benefit all stakeholders:

- **Students:** Easier job search, faster application process, and improved communication with .
- **Company:** Efficient candidate selection, quicker recruitment process, and wider reach to potential hires.
- **College Placement Officer:** Reduced workload, better organization of recruitment activities, and improved tracking of student job placements.

1.3. SCOPE OF THE PROJECT

This project focuses on developing a core system for managing campus recruitment activities. It will encompass the following functionalities:

- **User Management:** Create separate modules for College Management (Admin), Company Management, and Student user roles.
- **Data Management:** Facilitate storing, editing, and retrieving data for students, companies, and job postings.
- **Job Application Management:** Allow students to search and apply for jobs, and enable companies to view and select applicants.

1.4. OVERVIEW OF THE PROJECT [2,5]

The Campus Recruitment System is a web-based application designed to facilitate and streamline the recruitment process within educational institutions. The system provides a platform where employers can post job vacancies and interact with students directly. Students, on the other hand, can search for job opportunities, apply online, and manage their applications through the system.

- **User Profiles:** students can create on the platform and update detailed profiles.
- **Job Posting and Management:** Placement officer verify the job vacancies and post them, manage job listings, and review student applications status.
- **Application and Selection Process:** Students can search for job listings, apply online, and track their application status.
- **Admin Dashboard:** Admin have access to a centralized dashboard to manage user accounts, monitor activities, and generate reports.

The system aims to replace traditional manual processes with a digital, efficient, and user-friendly solution that benefits all stakeholders involved in the campus recruitment process.

1.5. PROBLEM STATEMENT [1, 2, 5, 7, 8]

The traditional campus recruitment process is often inefficient and tough for students, companies, and educational institutions. Students typically have to manually search for job postings, submit paper resumes, and wait for callbacks without any clear timeline or communication. This process can be time-consuming and stressful, as students may miss out on important opportunities due to lack of organization or timely information.

Companies, on the other hand, face challenges in reaching a large pool of qualified candidates efficiently. They often have to coordinate with multiple departments within the institution, handle a large volume of paper applications, and manually schedule interviews, which can lead to delays and errors. This not only increases the workload for recruiters but also hampers the overall effectiveness of the recruitment process.

Educational institutions struggle to manage and oversee the recruitment activities happening on their campuses. They often lack a centralized system to monitor job postings, track student applications, and generate useful reports. This makes it difficult for them to ensure a fair and transparent recruitment process and to provide necessary support to students and employers. The Campus Recruitment System aims to address these issues by providing a digital platform that streamlines and automates the recruitment process. By allowing students to create profiles, search for jobs, and apply online, the system makes it easier for them to find opportunities and stay informed. Employers can efficiently manage job postings, review applications, and schedule interviews, reducing their administrative burden. Educational institutions benefit from having a centralized system to oversee the entire process, ensuring better organization, transparency, and support for all parties involved.

1.6. OBJECTIVES OF THE PROJECT

The primary objectives of this campus recruitment system project are:

- **To automate the manual tasks** currently involved in the campus recruitment process, such as record keeping, data retrieval.
- **To improve the efficiency** of campus recruitment for all stakeholders: students, companies, and college placement officers.

- **To facilitate easier access to job opportunities** for students by providing a centralized platform for job postings and applications.
- **To streamline the recruitment process** for companies by offering efficient candidate selection tools and communication channels.
- **To reduce the workload** of college placement officers by automating administrative tasks and providing better organization of recruitment activities.
- **To develop a user-friendly and accessible** web-based application that caters to the specific needs of each user role.
- **To demonstrate the feasibility** of implementing a technology-driven solution for campus recruitment and its potential benefits for educational institutions and businesses.

CHAPTER - 2

SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

2.1 INTRODUCTION [3, 8]

System analysis is a critical phase in the development of the Campus Recruitment System. It involves understanding and specifying in detail what the system should do to meet the needs of its users. This phase includes gathering and analysing requirements, studying the current system and defining the functional and non-functional requirements. The goal is to ensure that the new system will improve the recruitment process, making it more efficient and effective for students, companies, and educational institutions. System analysis involves a thorough examination of the existing manual recruitment process to identify its strengths, weaknesses, opportunities, and threats (SWOT analysis). This analysis will help determine the requirements and functionalities needed for the proposed system.

2.2 SYSTEM STUDY [9, 22]

The system study will encompass the following activities:

- **Data Gathering:** Information about the current recruitment process will be collected through various methods:
    - Interviews with college placement officers, students, and company representatives.
    - Document analysis of existing recruitment materials, forms, and data records.
    - Surveys to understand user needs and preferences for a new system.
- **Requirement Analysis:** Based on the gathered data, the functional and non-functional requirements of the system will be identified. This includes:
    - **Functional requirements:** Specific actions the system should perform for each user role (placement officer, company, student). Examples include functionalities for managing user accounts, posting jobs, searching for jobs, applying for jobs, sending notifications, etc.

  - **Non-functional requirements:** Qualities of the system such as security, scalability, performance, usability, and reliability.
- **Feasibility Analysis:** The feasibility of developing and implementing the proposed system will be assessed based on three key factors:
  - **Technical Feasibility:** Evaluation of the available technology, resources, and expertise to develop and maintain the system.
  - **Economic Feasibility:** Cost-benefit analysis to determine if the project is financially viable within the allocated budget.
  - **Social Feasibility:** Assessment of user acceptance, potential impact on workflows, and the need for training and support.

The findings from the system study will be crucial in defining the system specifications and laying the foundation for the design and development phases of the project.

2.3. EXISTING SYSTEM [6, 8, 23]

- Since whole of the system to be maintained manually, the process of keeping, maintaining, and retrieving the information was very tedious and lengthy.
- It was difficult to find errors while entering the records.
- If any information was to be found, it was required to go through different registers, documents etc.
- Colleges, Companies and Students had to do lot of work for their recruitments.

2.3.1. DISADVANTAGES [1, 7, 24]

The manual system for campus recruitment suffers from several drawbacks that hinder its efficiency and effectiveness. These disadvantages include:

- **Tedious record keeping and data retrieval:** Maintaining student and company data in physical files or spreadsheets can be time-consuming and prone to errors. Searching for specific information can be cumbersome and slow.
- **Inefficient communication:** Coordinating communication between students, companies, and placement officers often involves emails, phone calls, and physical notices. This can be slow, lead to missed information, and create difficulties in tracking communication history.

- **Limited access to information:** Students may not have easy access to information about available job opportunities. They may rely on physical job postings or outdated information boards.
- **Difficulty tracking applications and placements:** Tracking the application status of students and the overall placement success rate can be challenging with a manual system. Data may be scattered across various documents, making it difficult to analyse trends or identify areas for improvement.
- **Lack of transparency:** The manual process may lack transparency for students regarding job selection criteria and company decisions. Students may not receive timely updates on their applications, leading to frustration and uncertainty.
- **Increased paperwork:** The manual system likely relies heavily on physical forms and documents for applications, job postings, and communication. This can be time-consuming to manage, prone to loss or damage, and not environmentally friendly.
- **Scalability issues:** As the number of students, companies, and job postings increases, the manual system becomes difficult to manage and maintain.

2.4. PROPOSED SYSTEM [22, 26]

- Separate modules for College Management (Placement Officer), Company Management, and Students ensure focused functionalities.
- Students, companies, and job details are stored electronically, enabling easy search, update, and retrieval.
- All student profiles, job postings, and employer information are stored in a single, unified database.
- Students can submit applications online through a user-friendly portal.
- Companies can view applications electronically.
- Track the status of applications and recruitment activities in real-time.

2.4.1. ADVANTAGES [23, 27]

The campus recruitment system aims to offer several advantages compared to the limitations of the manual process. These benefits encompass all stakeholders involved in campus recruitment:

❖ **Increased Efficiency:**

- Automates record keeping and data retrieval, saving time and effort for all users.
- Simplifies job application and selection processes.

❖ **Improved Accessibility:**

- Provides a centralized platform for students to access job postings and apply for positions.
- Allows companies to reach a wider pool of potential candidates.

❖ **Enhanced Transparency:**

- Students can track the status of their applications and receive timely updates.
- Clearer communication channels between all parties involved.

❖ **Reduced Paperwork:**

- Eliminates the need for physical documents for applications, job postings, and communication.
- More environmentally friendly solution.

❖ **Improved Organization:**

- Easier management of student, company, and job data.
- Tracking of application trends and placement success rates.

❖ **Scalability:**

- The system can accommodate a growing number of users and job postings without significant challenges.
- Easier to manage and maintain compared to a manual system.

❖ **User-friendliness:**

- Designed with a user-friendly interface for each user role (placement officer, company, student).
- Easy to navigate and access features relevant to each user.
- Potential for self-service functionalities (e.g., students updating profiles).

2.5. PROPOSED METHODOLOGY [2, 3, 4]

The development of the campus recruitment system will follow a structured methodology to ensure a well-defined and efficient development process.

1. **Requirement Gathering:** This initial phase involves collecting detailed information about the functionalities and user needs of the system. This will be achieved through:

    - Interviews with college placement officers, company representatives, and students.
    - Analysis of existing recruitment documents, forms, and data records.
    - Surveys to understand user preferences and expectations for the new system.

2. **System Design:** Based on the gathered requirements, this stage focuses on designing the overall architecture of the system. This includes:

    - Defining user interfaces (UI) for each user role (company, student, admin) to ensure a user-friendly experience.

- Designing data models to represent students, companies, jobs, applications, and other relevant entities.
- Developing a system workflow that outlines the interaction between users and the system functionalities.

3. **Development:** This stage involves translating the system design into a functional application. It will encompass:

    - Coding the system functionalities using the chosen programming language (e.g., Java).
    - **Frontend Development:** Develop the user interface using HTML, CSS, and JavaScript to ensure a responsive and user-friendly design.
    - **Backend Development:** Implement the server-side logic using PHP and MySQL, ensuring secure data handling and processing.
    - **Integration:** Integrate the frontend with the backend, ensuring seamless communication between client-side and server-side components.
    - Utilizing a web server (e.g.,) to deploy the application and make it accessible through a web browser.

4. **Testing:** Rigorous testing is crucial to ensure the system functions as intended and meets all requirements. This will involve:

    - Unit testing to verify the functionality of individual program modules.
    - Integration testing to ensure different system components work seamlessly together.
    - User Acceptance Testing (UAT) with representatives from each user role to gather feedback and identify any usability issues.

5. **Deployment:** Once testing is successful, the system will be deployed in a live environment. This may involve:

    - Deploying the application on a web server accessible to authorized users.
    - Providing user training materials and support for placement officers, companies, and students.

6. **Maintenance:** The system will require ongoing maintenance to address bugs, incorporate new features, and ensure security updates. This will involve:

   - Monitoring system performance and user feedback.
   - Implementing bug fixes and updates as needed.
   - Enhancing the system with new functionalities based on user feedback or future requirements.

7. Documentation and Training:

User Manuals: Create comprehensive user manuals and documentation to guide users on how to use the system effectively.

Training Sessions: Conduct training sessions for administrators, companies, and students to ensure they are familiar with the system's features and functionality.

2.6. SOFTWARE ENVIRONMENT [10, 11, 12]

Java Technology

Java technology is both a programming language and a platform.

The Java Programming Language

The Java programming language is a high-level language that can be characterized by all the following buzzwords:

- ♦ Simple
- ♦ Architecture neutral
- ♦ Object oriented
- ♦ Portable
- ♦ Distributed
- ♦ High performance
- ♦ Interpreted
- ♦ Multithreaded
- ♦ Robust

- Dynamic
- Secure

The software environment for the Campus Recruitment System is carefully chosen to ensure that the system is robust, scalable, and user-friendly. This section outlines the key components of the software environment, including the operating system, database, server, frontend technologies, scripting languages, and development tools used to build and deploy the system.

Operating System:

Windows XP or later versions: The system is designed to operate efficiently on any version of Windows XP or later, providing compatibility with a wide range of hardware configurations.

Database:

MySQL: The system uses MySQL as its relational database management system (RDBMS). MySQL is chosen for its reliability, ease of use, and strong support for web applications. It handles all data storage, retrieval, and management tasks efficiently.

Server:

: The HTTP Server is used as the web server. is a highly popular and powerful web server that ensures the reliable delivery of web content to users. It supports a wide range of functionalities and is highly configurable.

Frontend Technologies:

- **HTML (Hypertext Markup Language):**

  HTML is used for structuring the web pages and forms the backbone of the system's user interface.

- **CSS (Cascading Style Sheets):**

  CSS is used for styling the web pages, ensuring that the user interface is visually appealing and consistent across different browsers.

- **JavaScript:**

  JavaScript is used to add interactivity to the web pages, enabling dynamic content updates and enhancing user experience.

Scripting Language:

- **PHP (Hypertext Preprocessor):**

  PHP is the primary server-side scripting language used to develop the system's backend logic. PHP is widely used for web development and can easily integrate with the MySQL database, making it an ideal choice for this project.

Integrated Development Environment (IDE):

- **Visual Studio Code (VS Code):**

  VS Code is the chosen IDE for this project due to its powerful features, extensive extensions, and support for various programming languages. It provides a user-friendly environment for writing and managing code efficiently.

Additional Tools and Technologies:

- **Browser Compatibility:**

  The system is designed to be compatible with modern web browsers such as Mozilla Firefox (version 4.0 or later), Internet Explorer (version 6.0 or later), and Google Chrome. This ensures that users can access the system from different browsers without any issues.

2.6.1. FUNCTIONAL REQUIREMENTS [14, 15]

Functional requirements define what a system or product must do to fulfil its intended purpose. They describe the features, functions, and behaviours that the system should have.

- **Operations and workflows:** This specifies the tasks and processes that the system should be able to perform. For instance, in a library management system, a functional requirement might be the ability to search for books by title, author, or genre.

- **Data inputs and outputs:** This describes the format and validity of the data that the system can handle. An example would be a requirement specifying that a system should accept dates in YYYY-MM-DD format.
- **User interface behaviour:** This outlines how users will interact with the system. This could include requirements for menus, buttons, error messages, and other UI elements.
- **Data integrity and security:** This ensures that the system protects and manages data accurately. Functional requirements in this area might specify encryption methods or access control mechanisms.

Functional requirements are essential for ensuring that a system meets the needs of its users. They provide a clear blueprint to follow during the development process. By meeting these requirements, the system becomes functional and achieving its goals.

1. User Authentication:

- ♦ **Secure Login:**
    - All users (students, companies, Admin) must log in using a secure mechanism.
    - Implement password policies (e.g., minimum length, complexity) to ensure secure passwords.
    - Use encryption to protect user credentials during transmission and storage.
- ♦ **Role-Based Access Control:**
    - Different roles (student, company, admin) have different access levels and permissions.
    - Ensure users can only access the functionality appropriate for their role.
- ♦ **Student Profile Management:**
    - Students must be able to create their profiles by filling in required personal and academic information.
    - Provide an interface for students to update their profile information as needed.
- ♦ **Job Application:**
    - Students can browse and apply for job vacancies posted in the portal.

- Allow students to track the status of their applications.

2. Company Profile:

- Companies mail about job vacancies with detailed descriptions, requirements, and application deadlines to the placement officer.

Student Profile Viewing:

- Companies can view student profiles based on specific criteria (e.g., skills, academic background).

Recruitment :

- Companies can recruit the students regarding their interest in recruiting students.

3. Admin Management:

- ♦ **Student Management:**
  - Placement officers can add and manage student profiles.
  - Placement officer can Track students' job status.
- ♦ **Company Collaboration:**
  - Placement officers can view and select companies for collaboration.
  - Send notifications to companies regarding collaboration opportunities.
- ♦ **Recruitment Drives Management:**
  - Manage and schedule recruitment drives and company visits.
  - Track and manage the outcomes of recruitment activities.
- ♦ **User Management:**
  - Provide an interface for admin to monitor overall system activities and user interactions.

2.6.2. Non-Functional Requirements [16, 17]

Non-functional requirements (NFRs) address how a system should behave, rather than what specific actions it should perform. They define the system's quality attributes.
Non-functional requirements focus on criteria like performance, security, usability, reliability, and scalability. These characteristics influence the overall user experience and how well the system functions in real-world use. NFRs don't necessarily dictate what the

system does, but rather how well it does it. For instance, a non-functional requirement might specify that a web application should load pages in under three seconds.

1. Usability:

- Design a clean, intuitive interface for easy navigation by students, companies, and placement officers.
- Ensure consistency in design and functionality across all user roles.
- Provide clear instructions, tooltips, and help documentation for all features.
- Incorporate feedback mechanisms for continuous usability improvements.
- The system shall be accessible to users with varying levels of technical expertise.

2. Accessibility:

- Comply with web accessibility standards (e.g., WCAG) to ensure the system is usable by people with disabilities.
- Implement keyboard navigation and screen reader compatibility.

3. Performance:

- Ensure that the system responds to user actions within an acceptable time frame (e.g., page loads within 2 seconds).
- Optimize database queries and server interactions to minimize latency.
- Design the system to handle high volumes of concurrent users without degradation in performance.
- The system shall be available during peak usage times to ensure uninterrupted service.

4. Security:

- Encrypt sensitive data both in transit and at rest using industry-standard encryption protocols.
- Ensure compliance with data protection regulations (e.g., GDPR, CCPA).
- Conduct regular security audits and vulnerability assessments.
- Develop and maintain an incident response plan for handling security breaches.
- Train staff on security best practices and incident management.

4. Scalability:

- Design the system to support vertical (enhancing machine capacity) scaling.
- Utilize scalable cloud infrastructure to handle increased load as the user base grows.
- Implement a modular architecture to allow independent scaling of different system components.
- Continuously monitor system performance and resource usage.
- Use analytics to predict and plan for scaling needs proactively.

5. Maintainability:

- Adhere to coding standards and best practices to ensure clean, maintainable code.
- Conduct regular code reviews and refactoring sessions.
- Maintain comprehensive documentation for system architecture, codebase, and APIs.
- Ensure documentation is up to date with the latest system changes.

6. Portability:

- Design the system to be easily deployable across different environments
- Use containerization to ensure consistency across deployment environments.
- Provide tools and processes for smooth data migration during system upgrades or changes.

These non-functional requirements are essential for ensuring the system's overall quality, user satisfaction, and long-term viability.

2.7. Summary

This section comprehensively analyses the Campus Recruitment System, starting with an introduction to the current scenario and the need for an efficient recruitment solution. The system study highlights the importance of understanding user requirements and system capabilities, paving the way for an effective solution design. The existing system's analysis reveals several shortcomings, such as manual processes, inefficiencies, and a lack of integration, which hinder the recruitment process. Key disadvantages include time-consuming tasks, high error rates, and difficulties in managing large volumes of data.

To address these issues, the proposed system offers several improvements. It aims to automate and streamline the recruitment process, enhancing accuracy, efficiency, and user satisfaction. Key advantages include reduced manual intervention, improved data management, and enhanced user experience through a centralized platform. The proposed methodology focuses on developing a user-friendly system incorporating modern technologies and best practices in software development.

The software environment for the project includes tools and technologies essential for developing, deploying, and maintaining the system. These may include programming languages, frameworks, databases, and development tools that ensure a scalable and maintainable solution.

The system's capabilities, emphasizing user authentication, profile management, job posting and management, application processing, interview scheduling, shortlisting, and selection, reporting and analytics, and an admin dashboard. The system also emphasizes security and privacy, ensuring user data protection and compliance with relevant regulations. Document management features facilitate easy uploading and storage of resumes and other documents, while a feedback mechanism allows continuous improvement based on user input. The integration of these functionalities into the Campus Recruitment System aims to provide a seamless and efficient recruitment process, benefiting students, recruiters, and administrators alike. By addressing the limitations of the existing system and leveraging advanced technologies, the proposed system enhances the overall recruitment experience, making it more reliable, efficient, and user-centric. This comprehensive approach ensures that the system meets the diverse needs of its users and supports the institution's goal of effective and streamlined campus recruitment.

A structured development methodology (Section 2.5) will guide the creation of the system. This involves requirement gathering through interviews, surveys, and document analysis. Then, system design will define user interfaces and data models. Development follows, translating the design into a functional application using Java, MySQL, and web technologies (HTML, PHP, JavaScript). will serve as the web server, and VS code will be the development environment. Rigorous testing ensures the system functions as intended before deployment. Finally, ongoing maintenance will address bugs, incorporate new features, and ensure security updates.

The chosen software environment (Section 2.6) leverages open-source technologies for a cost-effective and scalable solution. Java provides a robust programming language, and MySQL offers efficient database management. Web technologies (HTML, PHP, JavaScript) create the user interface, and Tomcat acts as the web server. VS code streamlines the development process.

Functional requirements (Section 2.6.1) detail the specific actions the system must perform. These include user login, profile creation and management, company searching and student profile viewing, job application submission, and notification delivery based on user roles.
Non-functional requirements (Section 2.6.2) focus on the overall qualities of the system. Usability ensures user-friendly interfaces for different roles. Performance emphasizes responsiveness and handling multiple users. Security involves protecting sensitive data through encryption and access controls. Scalability ensures the system can accommodate growth without compromising performance.
By considering both functional and non-functional requirements, the campus recruitment system can be developed to be efficient, user-friendly, secure, and adaptable to future needs.

CHAPTER - 3

SYSTEM DESIGN

3. SYSTEM DESIGN

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing, and designing systems which satisfies the specific needs and requirements of a business or organization.

A systemic approach is required for a coherent and well-running system. Bottom-Up or Top-Down approach is required to consider all related variables of the system. A designer uses the modelling languages to express the information and knowledge in a structure of system that is defined by a consistent set of rules and definitions. The designs can be defined in graphical or textual modelling languages.

3.1. INPUT DESIGN [20, 21]

Input design is a crucial part of system design as it ensures that the data entering the system is accurate, complete, and secure. In the Campus Recruitment System, the input design focuses on how data is entered by different users such as administrators, placement officers, companies, and students. Proper input design can significantly reduce errors and improve the overall efficiency and usability of the system.

Objectives of Input Design:

1. Accuracy: Ensure that the data entered the system is correct and valid.

2. Simplicity: Make the input process straightforward and user-friendly.

3. Consistency: Maintain a consistent design across different input forms to avoid confusion.

4. Error Handling: Provide mechanisms to handle and correct input errors.

Types of Inputs:

1. Text Inputs:

- ✓ **Username and Password:** Used for logging into the system.
- ✓ **Names and Addresses:** Used for entering personal information of students, placement officers, and companies.

- ✓ **Job Titles and Descriptions:** Used by companies to post job vacancies.
- ✓ **Educational Details:** Used by students to provide their academic background.

2. Dropdowns:

- ✓ **Application Status:** Used by placement officers and companies to update the status of job applications.

3. Radio Buttons:

- ✓ **Gender Selection:** Used for demographic data collection in student profiles.

4. File Uploads:

- ✓ **Resumes:** Used by students to upload their resumes.
- ✓ **Company Logos:** Used by companies to upload their logos.

Input Forms:

1. Login Form:

- ✓ **Fields:** Username, Password
- ✓ **Validation:** Ensure that both fields are filled out and match existing records.

2. Registration Forms:

- ✓ **Company Registration:**
  - ➢ **Fields:** Company Name, Industry, Contact Person, Email, Phone Number, Address, Username, Password
  - ➢ **Validation:** Check for unique company name and valid contact information.
- ✓ **Student Registration:**
  - ➢ **Fields:** Full Name, Date of Birth, Gender, Email, Phone Number, Address, Department, Course, Year of Study, Username, Password
  - ➢ **Validation:** Ensure all fields are filled, validate email and phone number formats.

3. Profile Update Forms:

Administrator, Company, and Student Profiles:

- ✓ **Fields:** Various personal and contact information fields.

- ✓ **Validation:** Ensure all mandatory fields are filled, and the data format is correct.

4. Job Posting Form (for Companies):

- ✓ **Fields:** Job Title, Job Description, Required Skills, Job Type, Location, Application Deadline
- ✓ **Validation:** Ensure all fields are filled, and the deadline is a future date.

5. Job Application Form (for Students):

- ✓ **Fields:** Select Job, Attach Resume.
- ✓ **Validation:** Ensure job selection is made and required documents are attached.

Error Handling:

1. Inline Validation:

- ✓ Provide real-time feedback as users fill out forms, highlighting incorrect fields immediately.

2. Error Messages:

- ✓ Display clear and specific error messages to guide users in correcting their input.

3. Confirmation Messages:

- ✓ Show confirmation messages after successful data submission to inform users that their input has been received.

4. Default Values and Placeholders:

- ✓ Use default values and placeholders to guide users on what type of data is expected.

3.2. OUTPUT DESIGN [23, 24]

Output design is an essential aspect of the Campus Recruitment System as it determines how information is presented to the users in a clear, concise, and meaningful manner. Effective output design enhances user experience by providing them with the necessary information in an easily accessible format.

User Roles and Outputs:

- ✓ **Student:**
  - ➢ **Search Results:** List of available job postings matching their search criteria (job title, company, location, skills). Each entry should display relevant details like job title, company name, location, and a brief description.
  - ➢ **Job Details:** Detailed information about a specific job posting, including job description, requirements, and application instructions.
  - ➢ **Profile View:** Ability to view and update their profile information.
  - ➢ **Application Status:** Updates on the status of their applications (e.g., submitted, shortlisted, rejected, interview invite).
  - ➢ **Notifications:** Alerts and messages from the system regarding applications, upcoming deadlines, or other relevant information.
- ✓ **Company:**
  - ➢ **Student Profiles:** Displaying student names, educational background, relevant skills, and resumes .
  - ➢ **Application List:** List of applications received for their job postings, including student information, application documents, and a way to shortlist or reject candidates.
  - ➢ **Job Posting List:** Ability to view and manage their listed job postings.
  - ➢ **Notifications:** Alerts and messages from the system regarding applications, shortlisted candidates, or other relevant information.
- ✓ **Admin:**
  - ➢ **Student Applications:** View applications submitted by students for various job postings.
  - ➢ **Company List:** List of registered companies with their profile information.
  - ➢ **Reports:** Reports on job applications, or other relevant data by date-to-date report is available

3.3. ARCHITECTURE DIAGRAM [18]

An architecture diagram is a visual representation of a system's elements that helps people understand its layout. It can be used to show the overall outline of a software system, and to build relationships, boundaries, and constraints between its components. Architecture diagrams can also help identify potential risks in system development, such as faulty logic, incorrect assumptions, or inadequate testing.

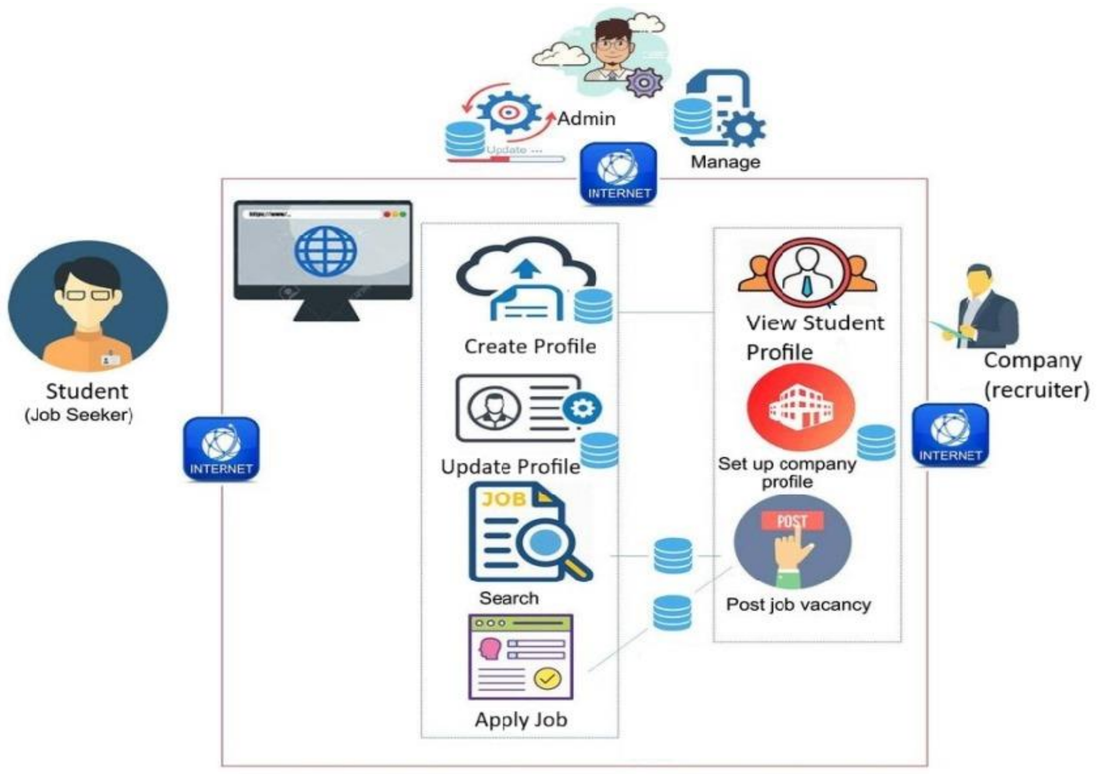


Fig 3.2 Architecture diagram of Campus Recruitment System

User Interface (UI):

- Web-based interface accessible through a standard web browser.
  Separate UIs for Students, Companies, and Placement Officers, each catering to specific functionalities.

Presentation Layer:

- Web technologies (HTML, CSS, JavaScript) for creating dynamic and interactive web pages.

- Java code handles user interactions, data processing, and business rules of the system.
- Communicates with the Data Access Layer to retrieve and store data.

Data Access Layer:

- Interfaces with the database (MySQL) to perform CRUD (Create, Read, Update, Delete) operations on student, company, job posting, and application data.

Database (MySQL):

- Stores all system data in a structured and organized manner.
- Ensures data integrity and efficient retrieval.

Web Server (Apache):

- Hosts the web application and makes it accessible to users through the internet.

3.4. USE CASE DIAGRAM [16]

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from the external point of view. The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system.

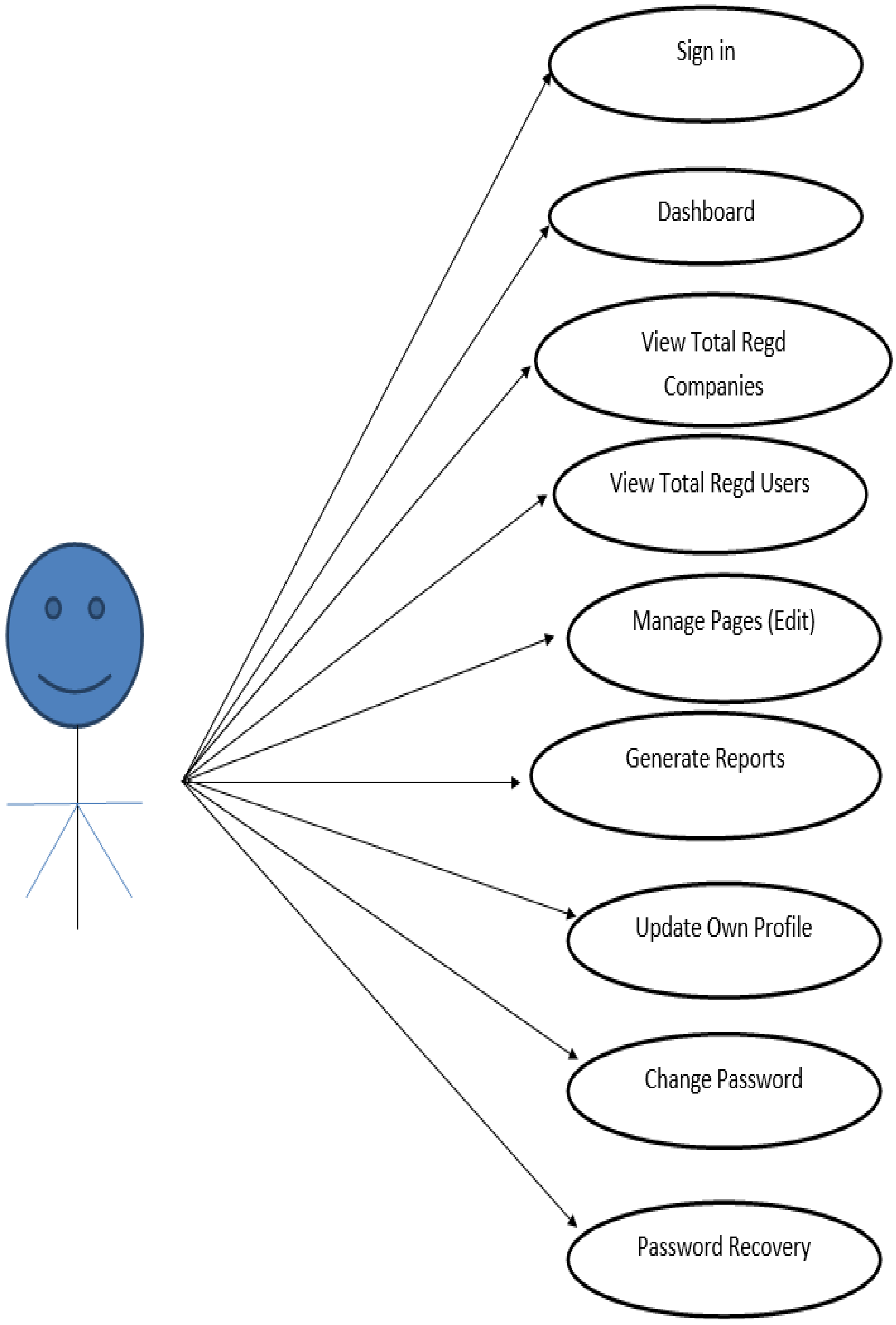


Fig 3.3 (a) Use-case Diagram of Admin

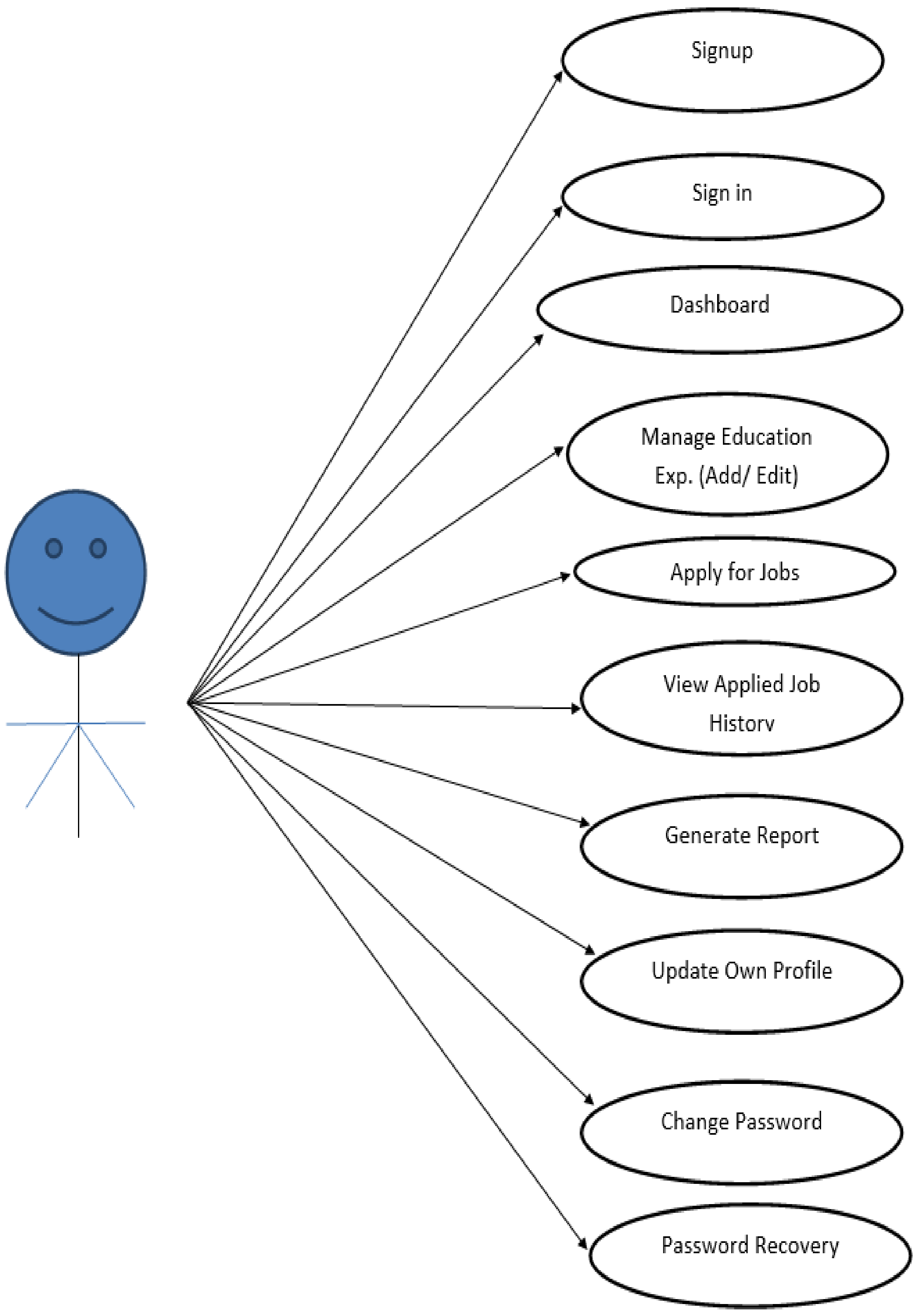


Fig 3.3 (b) Use-case Diagram of Student

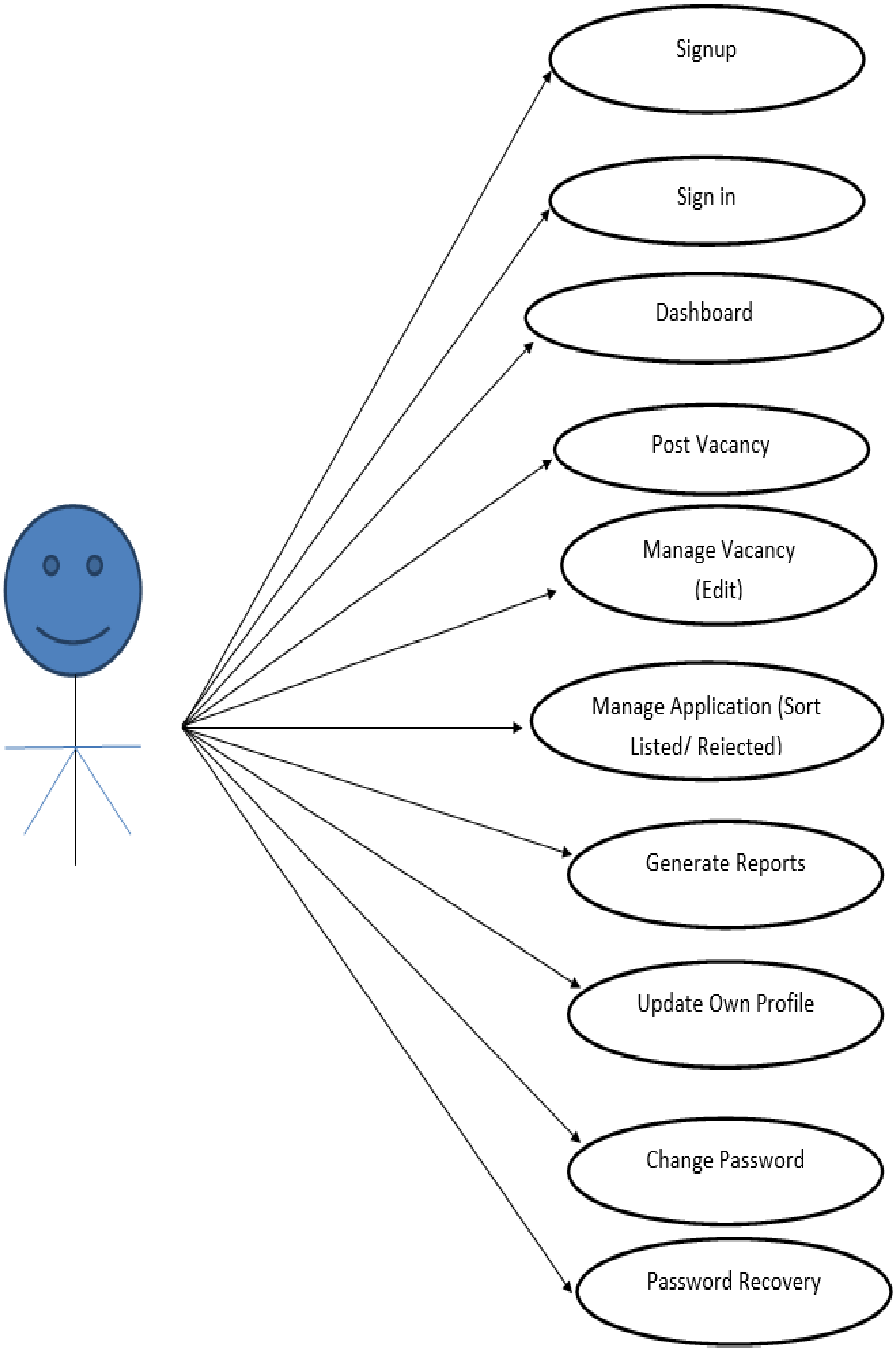


Fig 3.3 (c) Use-case Diagram of Company

3.5. CLASS DIAGRAM [19]

Class diagrams are fundamental to the object modelling process and model the static structure of a system. Depending on the complexity of a system, you can use a single class diagram to model an entire system, or you can use several class diagrams to model the components of a system. Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design. In the analysis stage, a class diagram can help you to understand the requirements of your problem domain and to identify its components. In an object-oriented software project, the class diagrams that you create during the early stages of the project contain classes that often translate into actual software classes and objects when you write code.

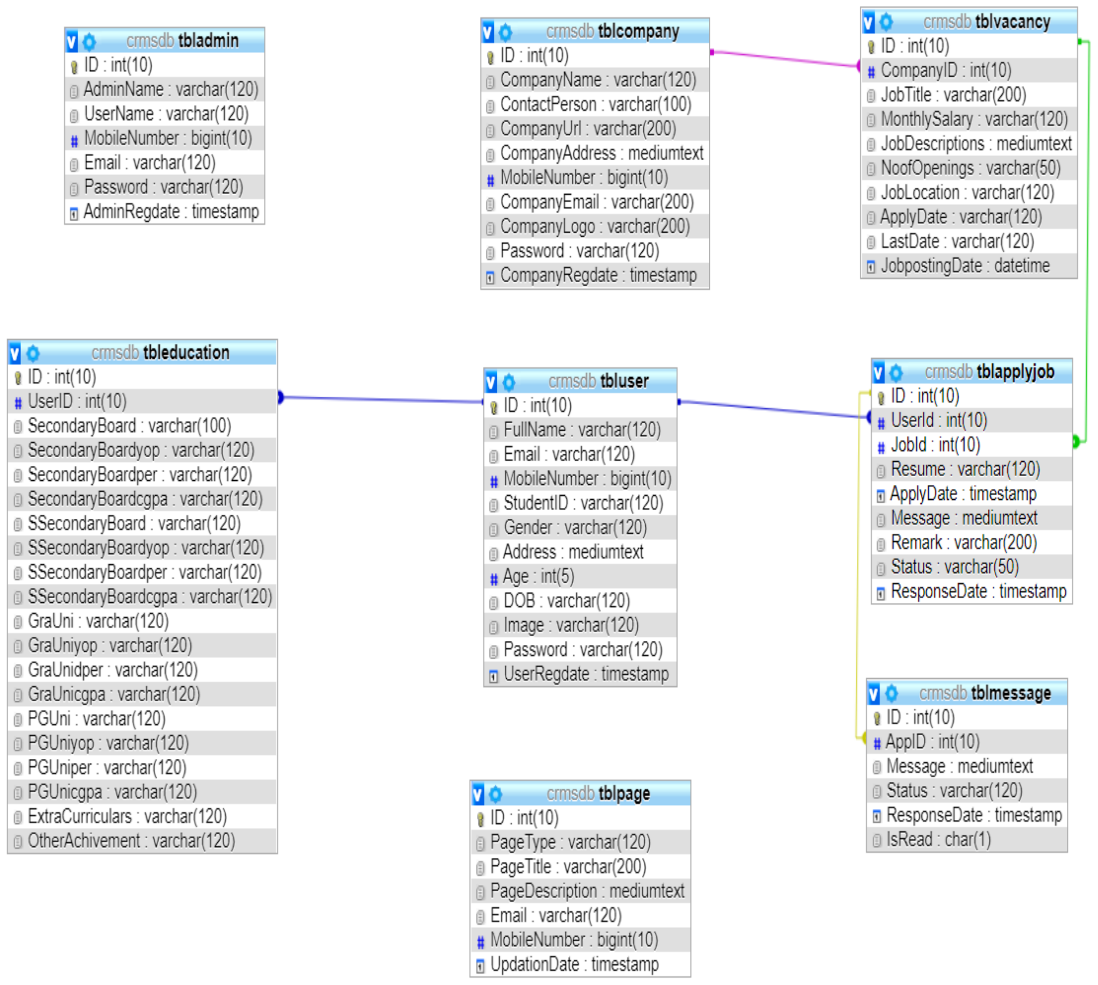


Fig 3.4. Class Diagram of Campus Recruitment System

3.6. ENTITY-RELATIONSHIP DIAGRAMS [11]

E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in the table.

The symbols used in E-R diagrams are:

SYMBOL	PURPOSE
▭	Represents Entity sets.
⬭	Represent attributes.
◇	Represent Relationship Sets.
—	Line represents flow

Structured analysis is a set of tools and techniques that the analyst.

To develop a new kind of a system:

The traditional approach focuses on the cost benefit and feasibility analysis, Project management, and hardware and software selection a personal consideration.

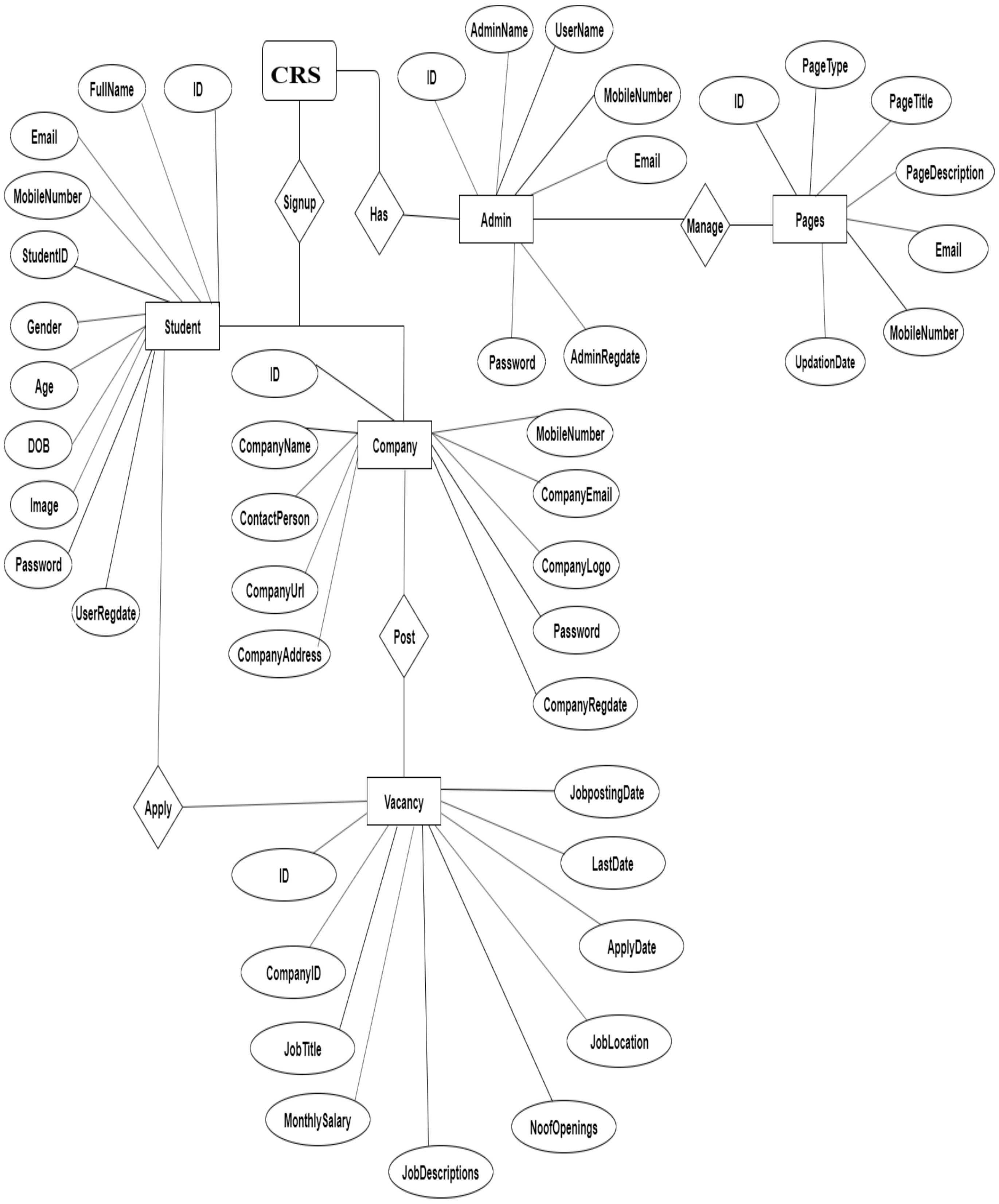
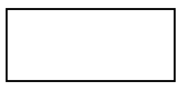


Fig 3.5. E-R (Entity-Relationship) Diagram of Campus Recruitment System

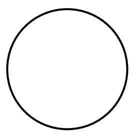
3.7. DATA FLOW DIAGRAMS [18]

A DFD does not show a sequence of steps. A DFD only shows what the different process in a system is and what data flows between them.

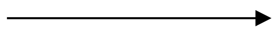
The following are some DFD symbols used in the project



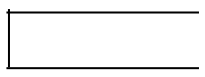
External entities



Process: A transaction of information that resides within the bounds of the system to be module.



Dataflows



DATASTORE: A repository of data that is to be stored for use by one or more processes, may be as simple as buffer of queue or as a relational database.

RULES FOR DFD:

- Fix the scope of the system by means of context diagrams.
- Organize the DFD so that the main sequence of the actions reads left to right and top to bottom.
- Identify all inputs and outputs.
- Identify and label each process internal to the system with rounded circles.
- A process is required for all the data transformation and transfers. Therefore, never connect a data store to a data source or the destinations or another data store with just a data flow arrow.
- Do not indicate hardware and ignore control information.

- Make sure the names of the processes accurately convey everything the process is done.
- There must not be unnamed process.
- Indicate external sources and destinations of the data, with squares.
- Number each occurrence of repeated external entities.
- Identify all data flows for each process step, except simple Record retrievals.
- Label data flow on each arrow.
- Use details flow on each arrow.
- Use the details flow arrow to indicate data movements.
- There can't be unnamed data flow.
- A data flow can't connect two external entities.

LEVELS OF DFD:

The complexity of the business system means that it is a responsible to represent the operations of any system of single data flow diagram. At the top level, an Overview of the different systems in an organization is shown by the way of context analysis diagram. When exploded into DFD

They are represented by:

- LEVEL-0: SYSTEM INPUT/OUTPUT
- LEVEL-1: SUBSYSTEM LEVEL DATAFLOW FUNCTIONAL
- LEVEL-2: FILE LEVEL DETAIL DATA FLOW.

The input and output data shown should be consistent from one level to the next.

LEVEL - 0: SYSTEM INPUT/OUTPUT LEVEL

A level-0 DFD describes the system-wide boundaries, dealing inputs to and outputs from the system and major processes. This diagram is similar to the combined user-level context diagram.

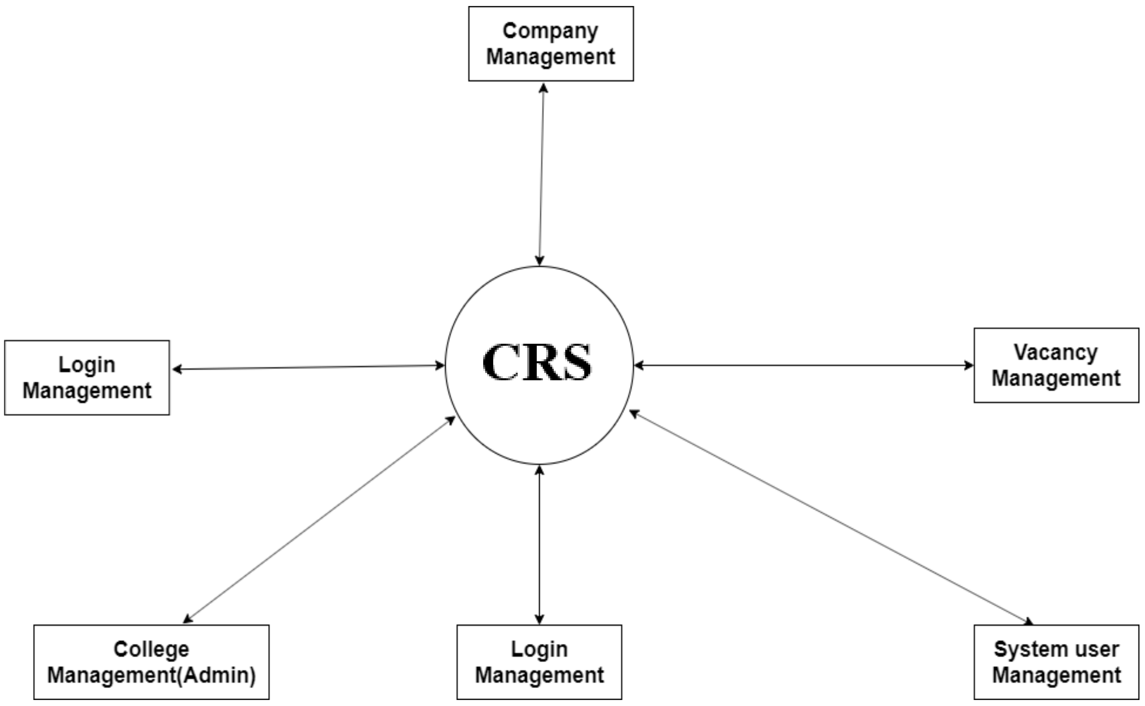


Fig 3.6. (a) Zero level Data Flow Diagram of Campus Recruitment System

LEVEL - 1: SUBSYSTEM LEVEL DATA FLOW

A level-1 DFD describes the next level of details within the system, detailing the data flows between subsystems, which makeup the whole.

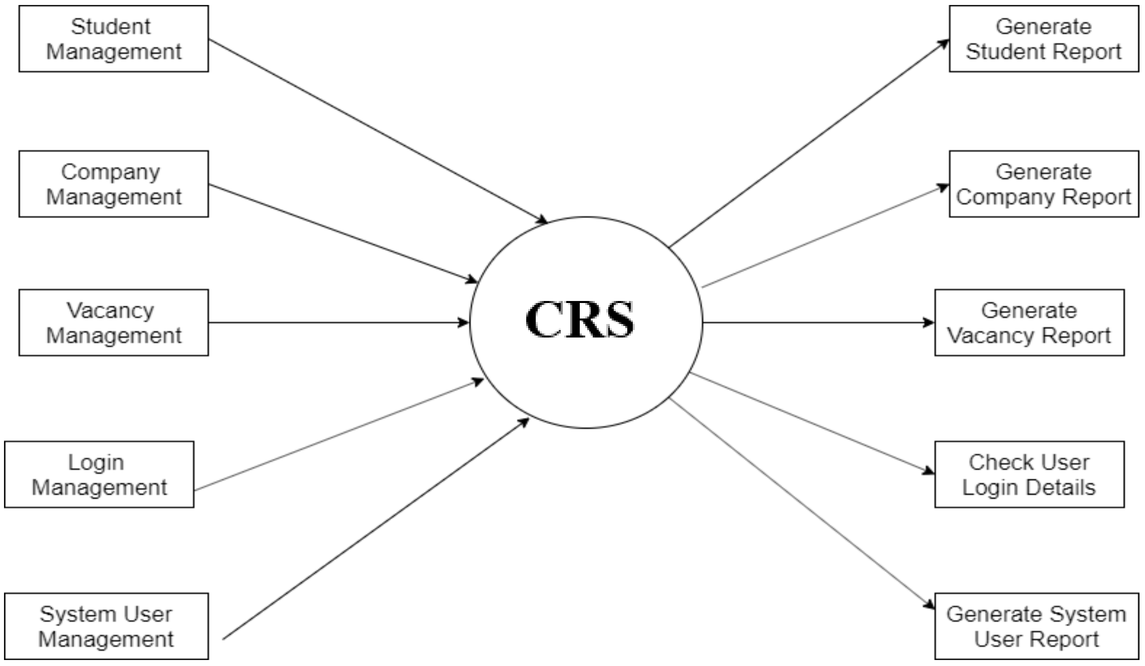


Fig 3.6.(b) First level Data Flow Diagram of Campus Recruitment System

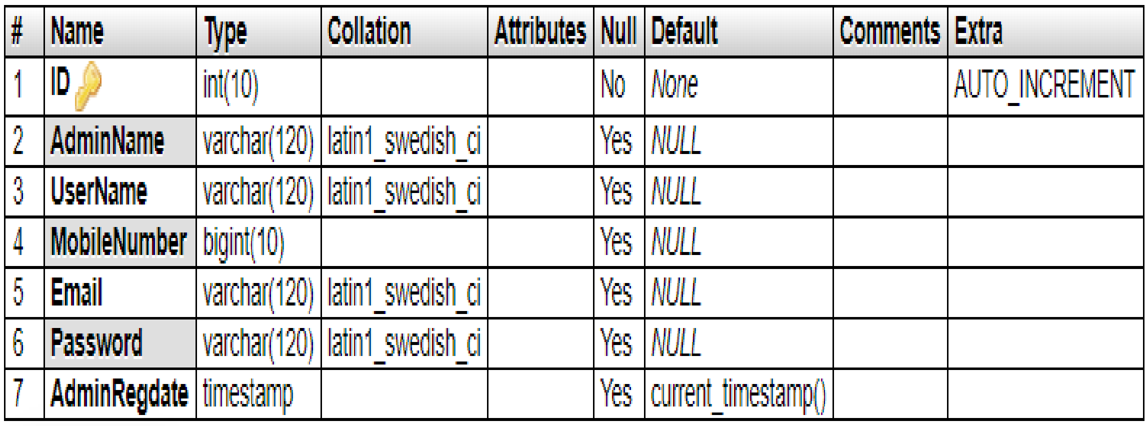
3.8. DATABASE DESIGN [17]

The data in the system has to be stored and retrieved from database. Designing the database is part of system design. Data elements and data structures to be stored have been identified at analysis stage. They are structured and put together to design the data storage and retrieval system.

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make database access easy, quick, inexpensive, and flexible for the user. Relationships are established between the data items and unnecessary data items are removed. Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data storage required, minimizing chances of data inconsistencies and optimizing for updates. The MySQL Database has been chosen for developing the relevant databases.

Campus Recruitment Management System (CRMS) contains 8 MySQL tables:

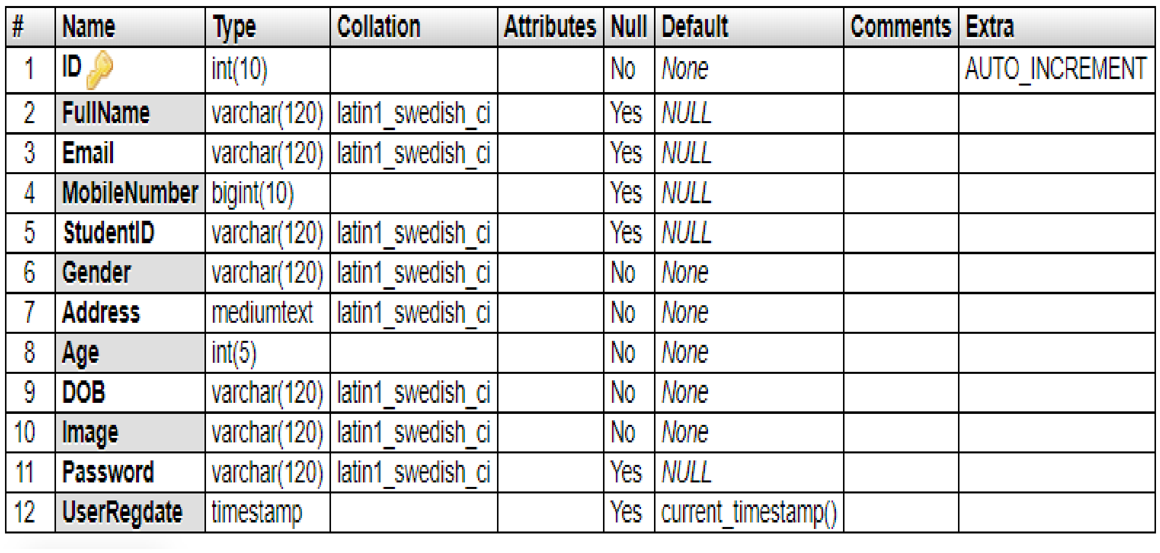
1. tbladmin table Structure : This table store the admin login and personal Details.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(10)			No	*None*		AUTO_INCREMENT
2	AdminName	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
3	UserName	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
4	MobileNumber	bigint(10)			Yes	*NULL*		
5	Email	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
6	Password	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
7	AdminRegdate	timestamp			Yes	current_timestamp()		

Table 3.8. (a) tbladmin

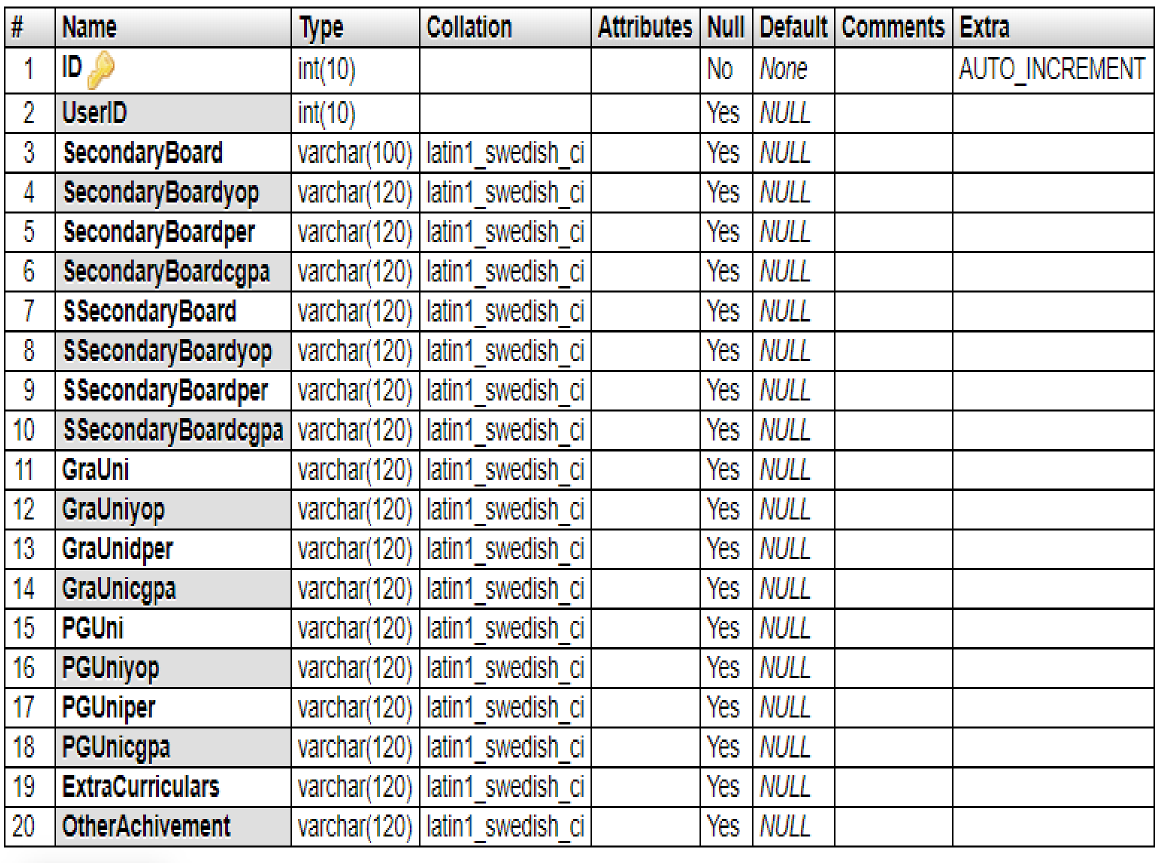
2. tbluser table Structure : This table store the student login and personal Details.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(10)			No	*None*		AUTO_INCREMENT
2	FullName	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
3	Email	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
4	MobileNumber	bigint(10)			Yes	*NULL*		
5	StudentID	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
6	Gender	varchar(120)	latin1_swedish_ci		No	*None*		
7	Address	mediumtext	latin1_swedish_ci		No	*None*		
8	Age	int(5)			No	*None*		
9	DOB	varchar(120)	latin1_swedish_ci		No	*None*		
10	Image	varchar(120)	latin1_swedish_ci		No	*None*		
11	Password	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
12	UserRegdate	timestamp			Yes	current_timestamp()		

Table 3.8. (b) tbluser

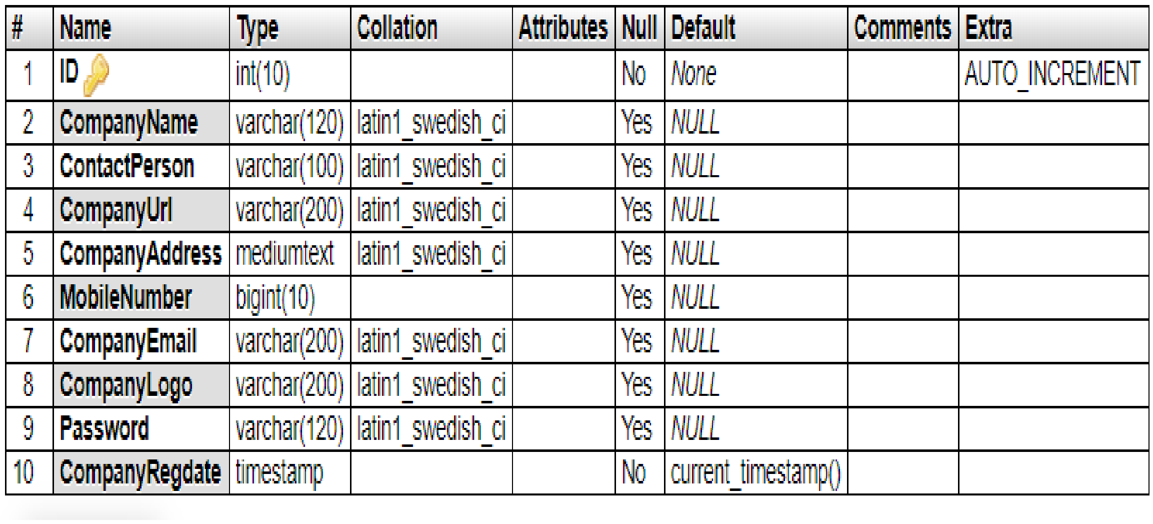
3. tbleducation table Structure : This table store the student education details.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(10)			No	*None*		AUTO_INCREMENT
2	UserID	int(10)			Yes	*NULL*		
3	SecondaryBoard	varchar(100)	latin1_swedish_ci		Yes	*NULL*		
4	SecondaryBoardyop	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
5	SecondaryBoardper	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
6	SecondaryBoardcgpa	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
7	SSecondaryBoard	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
8	SSecondaryBoardyop	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
9	SSecondaryBoardper	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
10	SSecondaryBoardcgpa	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
11	GraUni	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
12	GraUniyop	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
13	GraUnidper	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
14	GraUnicgpa	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
15	PGUni	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
16	PGUniyop	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
17	PGUniper	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
18	PGUnicgpa	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
19	ExtraCurriculars	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
20	OtherAchivement	varchar(120)	latin1_swedish_ci		Yes	*NULL*		

Table 3.8. (c) tbleducation

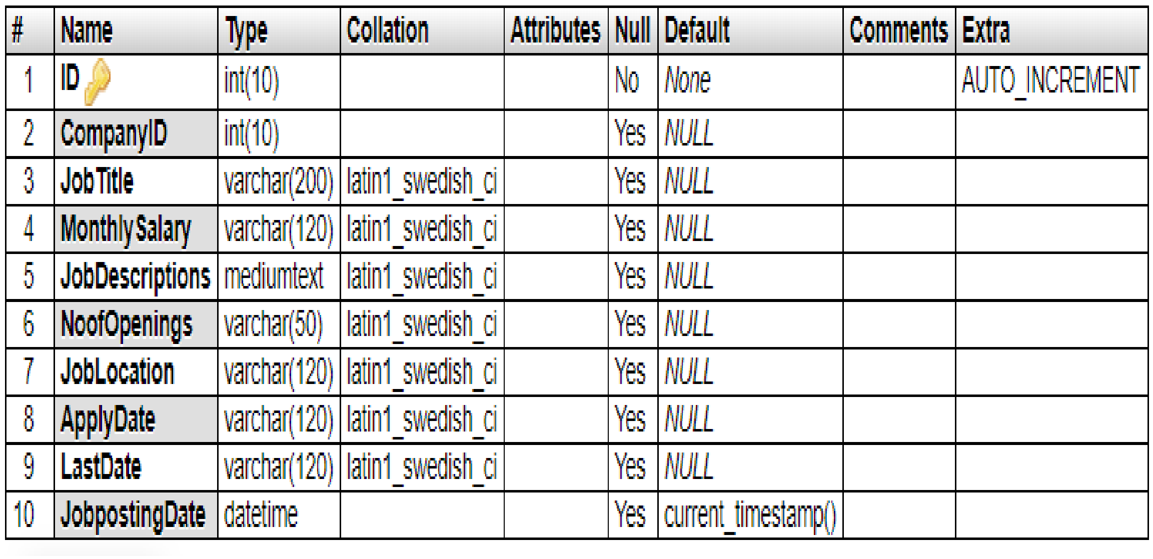
4. tblcompany table Structure : This table store the company details.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(10)			No	*None*		AUTO_INCREMENT
2	CompanyName	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
3	ContactPerson	varchar(100)	latin1_swedish_ci		Yes	*NULL*		
4	CompanyUrl	varchar(200)	latin1_swedish_ci		Yes	*NULL*		
5	CompanyAddress	mediumtext	latin1_swedish_ci		Yes	*NULL*		
6	MobileNumber	bigint(10)			Yes	*NULL*		
7	CompanyEmail	varchar(200)	latin1_swedish_ci		Yes	*NULL*		
8	CompanyLogo	varchar(200)	latin1_swedish_ci		Yes	*NULL*		
9	Password	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
10	CompanyRegdate	timestamp			No	current_timestamp()		

Table 3.8. (d) tblcompany

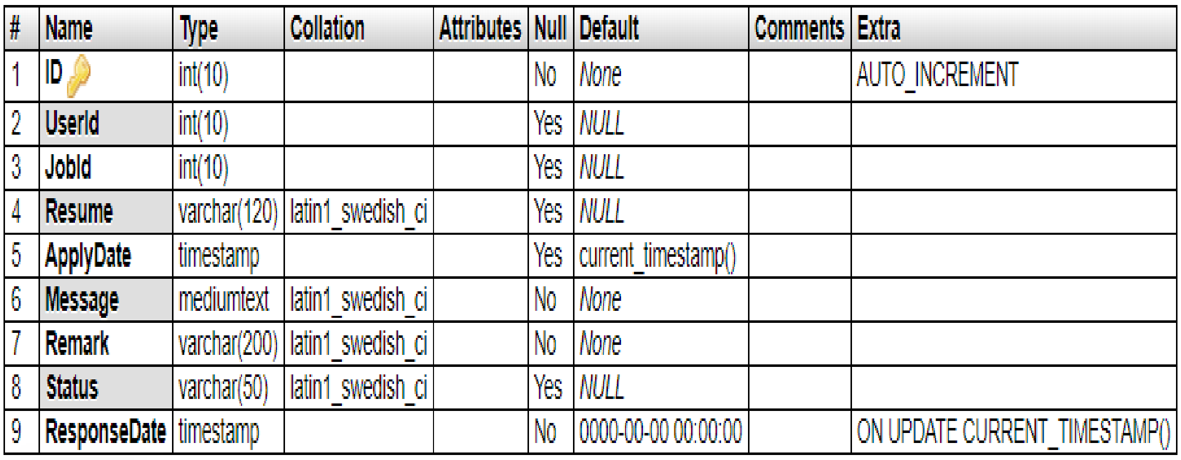
5. tblvacancy table Structure : This table store the company vacancy details.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(10)			No	*None*		AUTO_INCREMENT
2	CompanyID	int(10)			Yes	*NULL*		
3	JobTitle	varchar(200)	latin1_swedish_ci		Yes	*NULL*		
4	MonthlySalary	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
5	JobDescriptions	mediumtext	latin1_swedish_ci		Yes	*NULL*		
6	NoofOpenings	varchar(50)	latin1_swedish_ci		Yes	*NULL*		
7	JobLocation	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
8	ApplyDate	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
9	LastDate	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
10	JobpostingDate	datetime			Yes	current_timestamp()		

Table 3.8. (e) tblvacancy

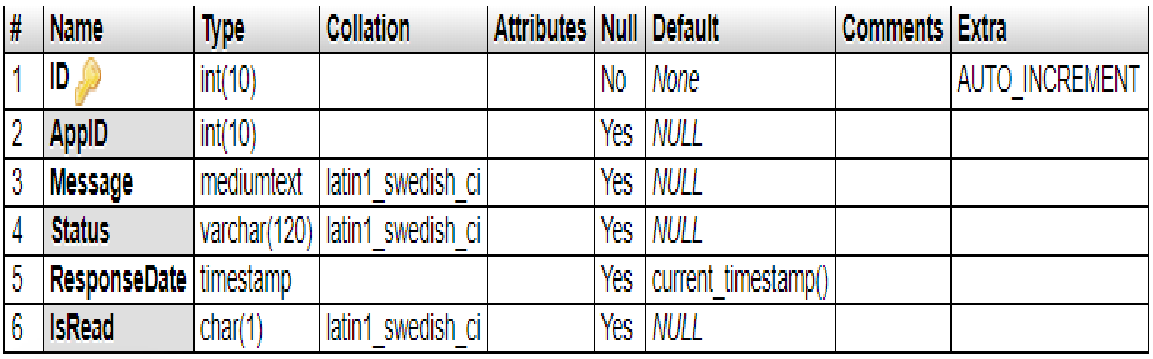
6. tblapplyjob table Structure : This table store the vacancy application details .



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(10)			No	*None*		AUTO_INCREMENT
2	UserId	int(10)			Yes	*NULL*		
3	JobId	int(10)			Yes	*NULL*		
4	Resume	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
5	ApplyDate	timestamp			Yes	current_timestamp()		
6	Message	mediumtext	latin1_swedish_ci		No	*None*		
7	Remark	varchar(200)	latin1_swedish_ci		No	*None*		
8	Status	varchar(50)	latin1_swedish_ci		Yes	*NULL*		
9	ResponseDate	timestamp			No	0000-00-00 00:00:00		ON UPDATE CURRENT_TIMESTAMP()

Table 3.8. (f) tblapplyjob

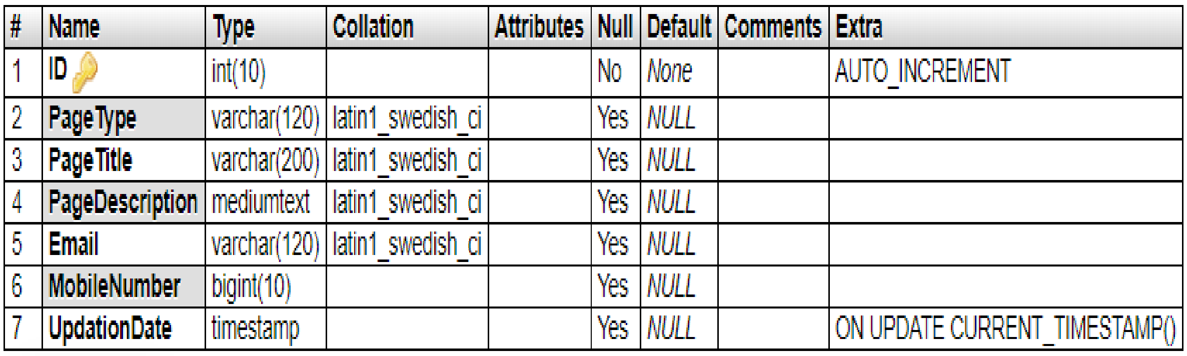
7. tblmessage table Structure : This table store the company message against any vacancy application .



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(10)			No	*None*		AUTO_INCREMENT
2	AppID	int(10)			Yes	*NULL*		
3	Message	mediumtext	latin1_swedish_ci		Yes	*NULL*		
4	Status	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
5	ResponseDate	timestamp			Yes	current_timestamp()		
6	IsRead	char(1)	latin1_swedish_ci		Yes	*NULL*		

Table 3.8. (g) tblmessage

8. tblpage table Structure : This table store the pages information .



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	ID	int(10)			No	*None*		AUTO_INCREMENT
2	PageType	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
3	PageTitle	varchar(200)	latin1_swedish_ci		Yes	*NULL*		
4	PageDescription	mediumtext	latin1_swedish_ci		Yes	*NULL*		
5	Email	varchar(120)	latin1_swedish_ci		Yes	*NULL*		
6	MobileNumber	bigint(10)			Yes	*NULL*		
7	UpdationDate	timestamp			Yes	*NULL*		ON UPDATE CURRENT_TIMESTAMP()

Table 3.8. (h) tblpage

3.9. SUMMARY

The system design phase of the Campus Recruitment System focuses on defining the architecture, components, modules, interfaces, and data for the system to satisfy specified requirements. Input Design emphasizes how the system captures and inputs data from users, ensuring that input methods are efficient, user-friendly, and error-resistant. Output Design deals with how information is presented to users, ensuring clarity, relevance, consistency, accessibility, and aesthetic appeal. This includes various output formats like web pages, PDFs, and emails tailored for different user types like administrators, companies, placement officers, and students.

The Architecture Diagram illustrates the high-level structure of the system, showing the major components and their interactions. It provides a blueprint of the system, highlighting the client-server interactions, database connections, and the flow of data between different modules. The Use Case Diagram identifies the different user roles and their interactions with the system, detailing the functionalities available to each type of user and how they use the system to achieve their goals.

The Class Diagram models the static structure of the system, showing the system's classes, attributes, methods, and the relationships between the classes. It is crucial for understanding the system’s object-oriented design and how different parts of the system interrelate. The Entity-Relationship Diagrams (ERDs) focus on the database design, illustrating the entities within the system, their attributes, and the relationships between these entities. This diagram is essential for designing the database schema and ensuring data integrity and normalization.

Data Flow Diagrams (DFDs) depict the flow of information within the system, from input to processing and output. They provide a clear visualization of how data moves through the system, identifying sources, destinations, and storage points. This helps in understanding the logical flow of data and identifying potential bottlenecks or inefficiencies. Database Design details the structure of the database, including tables, fields, and the relationships between tables. It ensures that the database is optimized for performance, scalability, and reliability.

The summary encapsulates the comprehensive design approach undertaken to develop a robust Campus Recruitment System. Each design aspect, from input and output specifications to the architectural and database design, ensures the system is well-structured, user-friendly, and capable of meeting the diverse needs of its users. By integrating detailed diagrams and thoughtful design principles, the system is geared to provide an efficient, secure, and scalable platform for managing campus recruitment activities effectively. This holistic approach to system design guarantees that all functional and non-functional requirements are addressed, paving the way for successful implementation and deployment.

CHAPTER – 4

SYSTEM IMPLEMENTATION

4. SYSTEM IMPLEMENTATION

4.1. INTRODUCTION [20, 21]

System implementation is a crucial phase in the development of the Campus Recruitment System, where the theoretical design is transformed into a working system. This section provides an overview of the implementation process, including the technologies used, development environment setup, and the challenges encountered during implementation. The implementation phase involves translating the design documents into actual code and integrating various software components to ensure the system functions as intended. It encompasses coding, database design, testing, deployment, and maintenance of the system. The successful implementation of the Campus Recruitment System requires adherence to the design specifications and the utilization of appropriate development tools and techniques. This section details the steps taken to ensure that the system meets the specified requirements and objectives.

- **Purpose:** This section details the steps involved in translating the blueprint we meticulously crafted in Section 3 into a functional web application. We'll walk through the implementation process, transforming the design into a reality.
- **Activities:** The implementation phase encompasses various activities like development, testing, deployment, and maintenance. Each stage plays a critical role in ensuring a robust and user-friendly system.
- **Deliverables:** The successful culmination of this phase is a fully-functional, rigorously tested, and deployed web application. This system will be readily accessible to students, companies, and placement officers, forever transforming the campus recruitment landscape.

4.2. MODULES [14, 15]

Module Description:

The three types of modules present in this project are

1. Admin
2. Company
3. User(Candidates/ Students)

❖ **Admin Module:**

- **Dashboard:** In this section, admin can see all detail in brief like Total Company Registered, Total User (Candidates) Registered and Total Vacancy Listed.
- **Total Registered Company:** In this section, admin can view detail of registered company.
- **Total Registered Users:** In this section, admin can view detail of users.
- **Pages:** In this section, the admin can manage about us and contact us pages.
- **Reports:** In this section admin can view how many company has been registered in particular period and also view how many vacancy counts listed by particular company in particular periods.

Admin can also update his profile, change the password and recover the password.

❖ **Company Module:**

- **Dashboard:** In this section, company can see all detail in brief like Total Number of applications received, Total number of new applications, Total number selected application, Total number of rejected applications.
- **Post Vacancy:** In this section, company can manage job posting(Add/Manage).
- **Job Application:** In this section, company can view total new applications receive, total sorted applications and total rejected applications and company also have right to sort application and reject application and this selected and rejected message send to candidates.
- **Reports:** In this section, company can view job posting in a particular period and see how many applications has been received in a particular period.

Company can also update his/her company profile, change the password, see the notifications of new applications received and recover the password.

❖ **Student Module:**

a. Guest Student
b. Registered Student

a) Guest Student

- **Home Page:** Student can see latest job posted on home page.
- **About Us:** Student can view about us page.
- **Listed Jobs:** Student can view total listed jobs.
- **Contact us:** Student can view contact us page.
- **Candidates:** In this guest Student can registered himself/herself.

b) Registered Student

- **Home Page:** Student can see latest job posted on home page.
- **About Us:** Student can view about us page.
- **Listed Jobs:** Student can view total listed jobs.
- **Contact us:** Student can view contact us page.
- **Student Dashboard:** After click on this Student can do following activities

i. View his/her applied(Today's applied jobs, Yesterday applied jobs, Last seven days applied jobs and total applied jobs)

ii. **Fill Educations forms:** In this section, Student can fill his/her own education details.

iii. **View Vacancy:** In this section, Student view the vacancy of companies and apply the jobs.

iv. **History of Applied Jobs:** In this section, Student can view his/her own applied job and see response of companies.

v. **Reports:** In these sections, Student can view his/her applied jobs in a particular period.

vi. **Search Jobs:** In this section, Student can search jobs according to job titles.

Student can also update his/her own profile, change the password, see notification message of companies and recover the password.

4.3. IMPLEMENTATION PROCEDURE [16,21]

The implementation of the Campus Recruitment System follows a systematic and structured approach to ensure that the system is developed, tested, and deployed effectively. The procedure involves several key steps that are critical for transforming the design into a fully functional application.

Step 1: Setting Up the Development Environment

Install Necessary Software: Install the required software and tools such as Apache server, MySQL database, PHP, and Visual Studio Code (VS Code) as the integrated development environment (IDE).

Configure the Server: Set up the Apache server to host the application, ensuring proper configuration for handling PHP files and connecting to the MySQL database.

Database Setup: Create the database schema in MySQL as per the design specifications. This involves creating tables, defining relationships, and setting up initial data.

Step 2: Coding and Development

Front-end Development: Develop the front-end of the application using HTML, CSS, and JavaScript. This includes creating user interfaces for the admin, company, and student modules.

Back-end Development: Implement the back-end logic using PHP. This involves writing code to handle user authentication, data processing, and business logic.

Database Integration: Write PHP scripts to interact with the MySQL database, performing CRUD (Create, Read, Update, Delete) operations as required by the system functionalities.

Step 3: Module Integration

Admin Module: Integrate and test functionalities such as viewing registered companies and users, managing pages, and generating reports.

Company Module: Implement and test functionalities for job posting, application management, and generating reports.

Student Module: Develop and test both guest and registered student functionalities, including viewing job listings, applying for jobs, and managing profiles.

Step 4: Testing

Unit Testing: Conduct unit testing to ensure that individual components and functions work correctly. Test cases should cover all possible scenarios to identify any bugs or issues.

Integration Testing: Perform integration testing to ensure that different modules and components work together seamlessly. This includes testing the interaction between the front-end and back-end, as well as database operations.

Performance Testing: Evaluate the system's performance under various conditions to ensure it meets the required performance standards. This includes testing response times, load handling, and scalability.

Step 5: Deployment

Deployment Preparation: Prepare the server environment for deployment. This includes setting up a live server, configuring security settings, and ensuring database backups.

Deployment: Deploy the application to the live server. Ensure all necessary files and configurations are transferred correctly.

Post-Deployment Testing: Conduct thorough testing on the live server to ensure that the deployment is successful and the system functions as expected.

Step 6: Maintenance

Monitoring: Continuously monitor the system for any issues or performance bottlenecks. Set up logging and monitoring tools to track system health and user activities.

Updates and Bug Fixes: Regularly update the system to fix any bugs, improve performance, and add new features. Ensure that updates are tested thoroughly before being applied to the live system.

4.4. SOURCE CODE

Admin Dashboard

```
<?php session_start();
error_reporting(0);
include('includes/dbconnection.php');
error_reporting(0);
if (strlen($_SESSION['crmsaid']==0)) {
  header('location:logout.php');
  } else{ ?>
<!DOCTYPE html>
<html lang="zxx">
<head>
    <title>Campus Recruitment System-Admin Dashboard</title>
    <!-- CSS -->
    <link rel="stylesheet" href="assets/css/app.css">
</head>
<body class="light">
<!-- Pre loader -->
<div id="loader" class="loader">
    <div class="plane-container">
       <div class="preloader-wrapper small active">
       </div>
    </div>
</div>
<div id="app">
<?php include_once('includes/sidebar.php');?>

<div class="page has-sidebar-left">

 <?php include_once('includes/header.php');?>

 <div class="container-fluid animatedParent animateOnce my-3">
  <div class="animated fadeInUpShort">
     <div class="card">
     <div class="card-header white">
     <h6> Admin Dashboard </h6>
     </div>
  </div>

<div class="row my-3">

<?php $query=mysqli_query($con,"Select * from  tblcompany");
$comcounts=mysqli_num_rows($query);
?>
```

```
<div class="col-md-4">
   <a href="total-regcompany.php" target="blank">
   <div class="counter-box white r-5 p-3">
   <div class="p-4">
   <div class="float-right">
   <span class="icon icon-building text-light-blue s-48"></span>
</div>
<div class="counter-title">Listed Companies</div>
<h5 class="sc-counter mt-3"><?php echo $comcounts;?></h5>
</div>
<div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div>
</div>
</div></a>
</div>
<?php $query1=mysqli_query($con,"Select * from  tbluser");
$userscounts=mysqli_num_rows($query1);
?>
<div class="col-md-4">
   <a href="total-regusers.php" target="blank">
   <div class="counter-box white r-5 p-3">
   <div class="p-4">
   <div class="float-right">
   <span class="icon icon-users s-48"></span>
</div>
<div class="counter-title ">Regsitered Users </div>
  <h5 class="sc-counter mt-3"><?php echo $userscounts;?></h5>
</div>
<div class="progress progress-xs r-0">
 <div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div>
</div>
</div> </a> </div>
<?php
$query2=mysqli_query($con,"Select * from  tblvacancy");
$vaccounts=mysqli_num_rows($query2);
?>
<div class="col-md-4">
      <a href="listed-jobs.php" target="blank">
      <div class="counter-box white r-5 p-3">
      <div class="p-4">
      <div class="float-right">
      <span class="icon icon-file text-blue s-48"></span>
```

```
</div>
<div class="counter-title">Posted Vacancies/Jobs</div>
<h5 class="sc-counter mt-3"><?php echo $vaccounts;?></h5> </div>
<div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div>
</div> </div>
</a> </div>
<?php
$query1=mysqli_query($con,"Select * from  tblapplyjob join tblvacancy
on  tblvacancy.ID=tblapplyjob.JobId");
$totalapplications=mysqli_num_rows($query1);
?>
<div class="col-md-4" style="margin-top:2%">
      <a href="all-applications.php" target="_blank">
      <div class="counter-box white r-5 p-3">
      <div class="p-4">
      <div class="float-right">
      <span class="icon icon-mail-envelope-open s-48"></span>
</div>
<div class="counter-title ">Total Job Applcations</div>
<h5 class="sc-counter mt-3"><?php echo $totalapplications;?></h5>
</div>
<div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div>
</div> </div> </a> </div>
<?php
$query12=mysqli_query($con,"Select * from  tblapplyjob join tbluser on
tblapplyjob.UserId=tbluser.ID join tblvacancy on
tblapplyjob.JobId=tblvacancy.ID  where  tblapplyjob.Status is null ||
tblapplyjob.Status=''");
$newapplication=mysqli_num_rows($query12);
?>
<div class="col-md-4" style="margin-top:2%">

      <a href="new-applications.php" target="_blank">

      <div class="counter-box white r-5 p-3">

      <div class="p-4">

<div class="float-right"> <span class="icon icon-mail-envelope-open text-
blue s-48"></span> </div>
<div class="counter-title">New Job Applications</div>
<h5 class="sc-counter mt-3"> <?php echo $newapplication;?> </h5> </div>
<div class="progress progress-xs r-0">
```

```
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div> </div></div>

 </a> </div>
<?php $query3=mysqli_query($con,"Select * from  tblapplyjob join
tblvacancy on  tblvacancy.ID=tblapplyjob.JobId where
tblapplyjob.Status='Sorted'");
$totalselapp=mysqli_num_rows($query3); ?>
<div class="col-md-4" style="margin-top:2%;">
      <a href="sort-listed-applications.php" target="blank">
      <div class="counter-box white r-5 p-3">
      <div class="p-4"> <div class="float-right">
      <span class="icon icon-mail-envelope-open text-yellow s-48"></span>
</div> <div class="counter-title">Sort Listed Job Applications</div>
<h5 class="sc-counter mt-3"><?php echo $totalselapp;?></h5> </div>
<div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div> </div>
</div> </a> </div>
<?php $query3=mysqli_query($con,"Select * from  tblapplyjob join
tblvacancy on  tblvacancy.ID=tblapplyjob.JobId where
tblapplyjob.Status='Selected'");
$totalselapp=mysqli_num_rows($query3); ?>
<div class="col-md-4" style="margin-top:2%;">
<a href="selected-applications.php" target="blank">
<div class="counter-box white r-5 p-3">
<div class="p-4"> <div class="float-right">
<span class="icon icon-mail-envelope-open text-green s-48"></span>
</div>
<div class="counter-title">Selected Job Applications</div>
<h5 class="sc-counter mt-3"><?php echo $totalselapp;?></h5> </div>
<div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div>
</div> </div> </a> </div>
<?php $query4=mysqli_query($con,"Select * from  tblapplyjob join
tblvacancy on  tblvacancy.ID=tblapplyjob.JobId where
tblapplyjob.Status='Rejected'");
$totalrejapp=mysqli_num_rows($query4); ?>
<div class="col-md-4" style="margin-top:2%;">
      <a href="rejected-applications.php" target="blank">
      <div class="counter-box white r-5 p-3">
      <div class="p-4"> <div class="float-right">
      <span class="icon icon-mail-envelope-open text-red s-48"></span>
</div>
<div class="counter-title">Rejected Job Applications</div>
```

```
<h5 class="sc-counter mt-3"> <?php echo $totalrejapp;?> </h5> </div>
<div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div> </div>
</div> </a> </div> </div> </div> </div> </div>
<div class="control-sidebar-bg shadow white fixed"></div> </div>
<!--/#app -->
<script src="assets/js/app.js"></script>
</body> </html> <?php } ?>
```

Student Dashboard

```
<?php
session_start();
error_reporting(0);
include('includes/dbconnection.php');
error_reporting(0);
if (strlen($_SESSION['crmsuid']==0)) {
  header('location:logout.php');
  } else{ ?>
<!DOCTYPE html>
<html lang="zxx">
<head>
    <title>Campus Recruitment System-User Dashboard</title>
    <!-- CSS -->
    <link rel="stylesheet" href="assets/css/app.css">
</head>
<body class="light">
<!-- Pre loader -->
<div id="loader" class="loader">
<div class="plane-container">
<div class="preloader-wrapper small active"> </div> </div>
</div>
<div id="app">
<?php include_once('includes/sidebar.php');?>
<div class="page has-sidebar-left">
<?php include_once('includes/header.php');?>
<div class="container-fluid animatedParent animateOnce my-3">
<div class="animated fadeInUpShort">
<div class="card">
<div class="card-header white">
<h6> User Dashboard </h6> </div> </div>
<div class="row my-3">
<?php $query4=mysqli_query($con,"select ID from tblvacancy");
$count_total_vacancy=mysqli_num_rows($query4); ?>
```

```
<div class="col-md-4" >
      <a href="view-vacancy.php" target="blank">
      <div class="counter-box white r-5 p-3">
      <div class="p-4"> <div class="float-right">
      <span class="icon icon-note-list text-light-blue s-48"></span>
</div> <div class="counter-title">Listed Vacancies/Jobs</div>
<h5 class="sc-counter mt-3"><?php echo $count_total_vacancy;?></h5>
</div>
<div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div> </div> </div>
</a> </div>
<?php $uid=$_SESSION['crmsuid'];
$query3=mysqli_query($con,"select ID from tblapplyjob where
UserId='$uid'");
$count_total_appjob=mysqli_num_rows($query3);
?>
<div class="col-md-4"> <a href="histroy-applied-job.php" target="blank">
<div class="counter-box white r-5 p-3">
<div class="p-4"> <div class="float-right">
<span class="icon icon-mail-envelope-open s-48"></span> </div>
<div class="counter-title ">Total Applied Jobs </div>
<h5 class="sc-counter mt-3"><?php echo $count_total_appjob;?></h5> </div>
<div class="progress progress-xs r-0"> <div class="progress-bar"
role="progressbar" style="width: 100%;" aria-valuenow="25" aria-
valuemin="0" aria-valuemax="128"></div> </div> </div> </a> </div>
<?php
$query=mysqli_query($con,"select ID from tblapplyjob where UserId='$uid'
&& date(ApplyDate)=CURDATE();");
$count_today_appjob=mysqli_num_rows($query);
?>
<div class="col-md-4">
<a href="histroy-applied-job.php" target="blank">
<div class="counter-box white r-5 p-3"> <div class="p-4">
<div class="float-right">
<span class="icon icon-mail-envelope-open text-blue s-48"></span>
</div> <div class="counter-title">Today's Applied Jobs</div>
<h5 class="sc-counter mt-3"><?php echo $count_today_appjob;?></h5>
</div> <div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div> </div> </div>
</a> </div>
<hr />
<?php  $query2=mysqli_query($con,"select ID from tblapplyjob where
UserId='$uid' && date(ApplyDate)=DATE(NOW()) - INTERVAL 1 DAY");
$count_yesterday_appjob=mysqli_num_rows($query2);
```

```
?>
<div class="col-md-4" style="margin-top:2%;">
      <a href="histroy-applied-job.php" target="blank">
      <div class="counter-box white r-5 p-3">
      <div class="p-4"> <div class="float-right">
      <span class="icon icon-mail-envelope-open text-yellow s-48"></span>
</div>
<div class="counter-title">Yesterdays Applied Jobs</div>
<h5 class="sc-counter mt-3"><?php echo $count_yesterday_appjob;?></h5>
</div>
<div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div> </div>
</div> </a> </div>
<?php  $query2=mysqli_query($con,"select ID from tblapplyjob where
UserId='$uid' && date(ApplyDate)=DATE(NOW()) - INTERVAL 1 DAY");
$count_sevenday_appjob=mysqli_num_rows($query2);
?>
<div class="col-md-4" style="margin-top:2%;">
      <a href="histroy-applied-job.php" target="blank">
      <div class="counter-box white r-5 p-3"> <div class="p-4">
      <div class="float-right">
      <span class="icon icon-mail-envelope-open text-green s-48"></span>
</div>
<div class="counter-title">Applied Jobs in Last 7 Days</div>
h5 class="sc-counter mt-3"><?php echo $count_sevenday_appjob;?></h5>
</div> <div class="progress progress-xs r-0">
<div class="progress-bar" role="progressbar" style="width: 100%;" aria-
valuenow="25" aria-valuemin="0" aria-valuemax="128"></div>
</div> </div> </a> </div> </div> </div> </div> </div>
<div class="control-sidebar-bg shadow white fixed"></div>
</div>
<!--/#app -->
<script src="assets/js/app.js"></script>
</body>
</html> <?php } ?>
```

4.5. FLOW OF EXECUTION [27]

The flow of execution outlines the step-by-step process of how different users interact with the Campus Recruitment System. It details the sequence of actions from logging in to performing various tasks within the system. This flow ensures that each user type (Admin, Company, and Student) can navigate and utilize the system effectively.

- **Admin Module Flow**
  1. **Login:** The admin logs in using their credentials.
  2. **Dashboard:** Upon successful login, the admin is directed to the dashboard displaying an overview of total registered companies, users, and listed vacancies.
  3. **View Registered Companies:** The admin can navigate to the section to view detailed information about all registered companies.
  4. **View Registered Users:** The admin can view detailed information about all registered users.
  5. **Manage Pages:** The admin can manage content for the 'About Us' and 'Contact Us' pages.
  6. **Generate Reports:** The admin can generate reports on company registrations and job vacancies over specific periods.
  7. **Profile Management:** The admin can update their profile, change the password, and recover the password if needed.
- **Company Module Flow**
  1. **Login/Registration:** The company representative logs in or registers for a new account.
  2. **Dashboard:** After logging in, the company is directed to a dashboard displaying details like total applications received, new applications, selected applications, and rejected applications.
  3. **Post Vacancy:** The company can add or manage job postings.
  4. **View Job Applications:** The company can view new applications, sort them, and send notifications for selected or rejected applications.

5. **Generate Reports:** The company can generate reports on job postings and received applications over specific periods.
6. **Profile Management:** The company can update its profile, change the password, see notifications of new applications, and recover the password if needed.

❖ **Student Module Flow**

a. **Guest Student Flow:**

   1. **Home Page:** The guest student visits the home page to see the latest job postings.
   2. **About Us:** The guest student can view the 'About Us' page.
   3. **View Listed Jobs:** The guest student can browse through the total listed jobs.
   4. **Contact Us:** The guest student can view the 'Contact Us' page.
   5. **Registration:** The guest student can register themselves to become a registered user.

b. **Registered Student Flow:**

   1. **Login:** The student logs in using their credentials.
   2. **Dashboard:** After logging in, the student is directed to their dashboard where they can see the latest job postings and perform various actions.
   3. **View Applied Jobs:** The student can view their applied jobs, categorized by today, yesterday, last seven days, and total applied jobs.
   4. **Fill Education Forms:** The student fills in their educational details.
   5. **View Vacancies:** The student views available job vacancies and applies for suitable positions.
   6. **History of Applied Jobs:** The student can view the history of their applied jobs and see the responses from companies.
   7. **Generate Reports:** The student can generate reports on their applied jobs over specific periods.
   8. **Search Jobs:** The student can search for jobs based on job titles.
   9. **Profile Management:** The student can update their profile, change the password, see notification messages from companies, and recover the password if needed.

4.6. SUMMARY

In the system implementation phase of the Campus Recruitment System project, we transformed the detailed design into a functional and user-friendly application. Starting with the Introduction (Section 4.1), we outlined the importance of this phase and the steps required to bring the project from a conceptual design to a working system. This included setting up the development environment, choosing appropriate technologies, and addressing any challenges encountered during the process.

In Modules (Section 4.2), we detailed the primary components of the system, which are the Admin, Company, and Student (Candidates) modules. Each module has specific functionalities tailored to meet the needs of its respective users, ensuring that all stakeholders have the tools they need to interact with the system effectively.

The Implementation Procedure (Section 4.3) provided a step-by-step guide on how the system was developed. This included coding, integrating different software components, testing, and deploying the system. The methodology ensured that the system was built according to the specified requirements and design.

Source Code (Section 4.4) encompassed the actual written code for the project. This section covered the various scripts and programs used to implement the system's functionalities, adhering to best practices in coding standards to ensure maintainability and scalability.

Flow of Execution (Section 4.5) illustrated the user journey through the system. It detailed how admins, companies, and students interact with the system from login to performing their respective tasks, ensuring a smooth and intuitive experience for all users.

Together, these sections highlight the comprehensive effort put into implementing the Campus Recruitment System. The careful planning, development, and testing phases culminate in a robust system that meets the needs of administrators, companies, and students. This implementation phase is crucial for ensuring the system is operational, efficient, and ready for deployment in a real-world environment.

CHAPTER - 5

SYSTEM TESTING

5. SYSTEM TESTING

5.1. INTRODUCTION [23, 24]

System testing is a critical phase in the software development lifecycle of the Campus Recruitment System. This phase involves verifying that the entire system functions as intended and meets the specified requirements. System testing encompasses various types of testing, including functional, integration, and performance testing, to ensure that all components of the system work together seamlessly. The primary goal of system testing is to identify and resolve any defects or issues before the system is deployed for actual use. This section will provide an overview of the objectives, methodologies, and processes involved in testing the Campus Recruitment System, ensuring it is robust, reliable, and ready for deployment. Through rigorous testing, we aim to deliver a high-quality product that enhances the recruitment process for students, companies, and administrators.

5.2. TESTING PROCESS [2,6]

The testing process for the Campus Recruitment System involves several phases to ensure that the system meets all functional and performance requirements. This structured approach ensures thorough validation of each component and the entire system.

1. Test Planning

Objective: Define the scope, approach, resources, and schedule of the testing activities.

Activities:

- Identify testing objectives and deliverables.
- Develop a test plan document outlining the testing strategy, resources required, test environment, testing schedule, and risk analysis.

2. Test Design

Objective: Design test cases based on the system requirements and specifications.

Activities:

- Review the functional and non-functional requirements of the system.
- Develop detailed test cases and scenarios that cover all aspects of the system, including user interfaces, functionalities, performance, and security.
- Create a traceability matrix to ensure all requirements are covered by test cases.

3. Test Environment Setup

Objective: Prepare the testing environment where tests will be executed.

Activities:

- Configure the hardware and software required for testing.
- Set up the database and load test data.
- Ensure all tools and resources needed for testing are available and operational.

4. Test Execution

Objective: Execute the designed test cases and document the results.

Activities:

- Execute unit tests on individual components to ensure they function correctly in isolation.
- Perform integration tests to verify that different modules and components work together as expected.
- Conduct system testing to validate the overall functionality, performance, and reliability of the entire system.
- Log any defects or issues found during testing for further analysis and resolution.

5. Defect Reporting and Tracking

Objective: Document and track defects found during testing.

Activities:

- Record details of each defect, including steps to reproduce, severity, and impact.

- Use a defect tracking system to manage and prioritize defects.
- Communicate defects to the development team for resolution.

6. Test Closure

Objective: Ensure all testing activities are completed, and the system is ready for deployment.

Activities:

- Verify that all test cases have been executed and all defects have been resolved.
- Review and evaluate the test execution process.
- Prepare a test summary report, detailing the testing outcomes, defect status, and overall system quality.

7. Post-Implementation Testing

Objective: Conduct additional testing after deployment to ensure the system works correctly in the live environment.

Activities:

- Perform smoke testing to verify the basic functionality of the system.
- Conduct user acceptance testing (UAT) with real users to validate the system meets their needs.
- Monitor the system's performance and stability in the live environment and address any issues that arise.

5.3. UNIT TESTING [6,7]

Unit testing is a critical phase in the software testing lifecycle. It focuses on verifying the functionality of individual components or units of the software. For the Campus Recruitment System, unit testing ensures that each module performs as expected in isolation, without interference from other components.

Objectives:

- To validate the functionality of each unit (module) in isolation.
- To identify and fix defects at an early stage, reducing the cost and complexity of fixing issues later.
- To ensure that each unit meets its design and behaves as intended.

Scope:

Unit testing will cover all core modules of the Campus Recruitment System, including but not limited to:

- Admin functionalities such as user management and report generation.
- Company functionalities such as job postings and application processing.
- Student functionalities such as profile management and job application.

Methodology:

Unit testing for the Campus Recruitment System will follow a structured approach using a combination of manual and automated testing techniques. The following steps outline the methodology:

1. **Test Case Design**
    - **Identify Test Scenarios:** Based on the module specifications and requirements, identify key test scenarios for each unit.
    - **Write Test Cases:** Develop detailed test cases for each scenario, specifying inputs, expected outputs, and execution steps.
    - **Prepare Test Data:** Create the necessary test data for executing the test cases.
2. **Test Case Execution**
    - **Set Up Test Environment:** Ensure the development environment is configured correctly with all necessary dependencies and tools.
    - **Execute Test Cases:** Run the test cases, recording the actual outputs and comparing them against the expected outputs.
    - **Log Results:** Document the outcomes of each test case, noting any discrepancies or defects.
3. **Defect Reporting and Resolution**

- **Log Defects:** For any failed test cases, log the defects in a tracking system, providing detailed information about the issue.
- **Fix Defects:** Developers will analyze and fix the reported defects, followed by retesting to ensure the issues are resolved.
- **Regression Testing:** After fixing defects, perform regression testing to ensure that the changes have not introduced new issues.

4. **Automation**
    - **Automate Repetitive Tests:** Use testing frameworks (e.g., PHPUnit for PHP) to automate repetitive and regression test cases.
    - **Continuous Integration:** Integrate unit tests into the continuous integration (CI) pipeline to ensure ongoing validation of the codebase with every change.

Tools and Frameworks

- **PHPUnit:** A unit testing framework for PHP that will be used to write and execute automated unit tests.
- **VS Code:** Integrated Development Environment (IDE) for writing and running unit tests.
- **MySQL:** Database used for setting up test data and validating database interactions.
- **Apache:** Web server used to host the application during testing.

5.4. INTEGRATION TESTING [8, 9]

Integration testing focuses on verifying the interaction between integrated units or modules to ensure they function correctly as a group. The goal is to uncover issues in the interfaces and interactions between the modules that unit testing may have missed. For the Campus Recruitment System, integration testing is critical to ensure seamless interaction among the Admin, Company, and Student modules.

Integration Testing Strategy

1. **Big Bang Integration Testing:** All or most of the developed modules are coupled together to form a complete system and tested in one go.

2. **Top-Down Integration Testing:** Testing starts from the top-level modules and progresses through lower levels.
3. **Bottom-Up Integration Testing:** Testing starts from the lower-level modules and progresses through upper levels.

5.5. BLACK BOX TESTING [3, 4]

Black box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to every level of software testing: unit, integration, system, and acceptance. It is also called functional testing.

For the Campus Recruitment System, black box testing focuses on validating the system's inputs and outputs against the specified requirements without considering the internal code structure.

Black Box Testing Strategy

1. **Requirement-Based Testing:** Verifying that the system meets the specified requirements.
2. **Boundary Value Analysis:** Testing the boundaries between partitions.
3. **Equivalence Partitioning:** Dividing inputs into equivalent partitions that can be used to derive test cases.
4. **Decision Table Testing:** Using decision tables to represent combinations of inputs and outputs.

5.6. WHITE BOX TESTING [13, 14]

White box testing, also known as clear box testing, glass box testing, transparent box testing, and structural testing, is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e., black box testing). In white box testing, the tester needs to have knowledge of the internal code, design, and structure of the software. This type of testing is useful for optimizing code, checking for security vulnerabilities, and ensuring thoroughness in testing all code paths.

White Box Testing Techniques

1. **Control Flow Testing:** Examines the flow of control through the program to identify all executable statements and ensure each one is executed at least once.
2. **Data Flow Testing:** Focuses on the points at which variables receive values and the points at which these values are used.
3. **Branch Testing:** Ensures that each possible branch from each decision point is executed at least once.
4. **Path Testing:** Ensures that every possible path through the program is executed.
5. **Statement Coverage:** Ensures that every statement in the code is executed at least once.
6. **Decision Coverage:** Ensures that each decision in the program evaluates to both true and false.

5.7. TEST CASES

Test Scenarios and Cases for Campus Recruitment System

Unit Testing:

Admin Module:

Test Case ID: UT-Admin-01

- **Module:** Admin Dashboard
- **Test Scenario:** Verify that the total number of registered companies is displayed correctly on the admin dashboard.
- **Preconditions:** The database contains records of registered companies.
- **Test Steps:**
    1) Log in as an admin.
    2) Navigate to the dashboard.
    3) Check the total number of registered companies displayed.
- **Expected Result:** The displayed number matches the count of company records in the database.
- **Status:** [Pass]

Test Case ID: UT-Admin-02

- **Module:** Admin Dashboard
- **Test Scenario:** Verify that the total number of registered users is displayed correctly on the admin dashboard.
- **Preconditions:** The database contains records of registered users.
- **Test Steps:**
  1) Log in as an admin.
  2) Navigate to the dashboard.
  3) Check the total number of registered users displayed.
- **Expected Result:** The displayed number matches the count of user records in the database.
- **Status:** [Pass]

Company Module:

Test Case ID: UT-Company-01

- **Module:** Company Dashboard
- **Test Scenario:** Verify that the total number of applications received is displayed correctly on the company dashboard.
- **Preconditions:** The database contains records of job applications.
- **Test Steps:**
  1) Log in as a company representative.
  2) Navigate to the dashboard.
  3) Check the total number of applications received displayed.
- **Expected Result:** The displayed number matches the count of application records in the database.
- **Status:** [Pass]

Test Case ID: UT-Company-02

- **Module:** Job Application Management
- **Test Scenario:** Verify the functionality of sorting applications into accepted and rejected categories.
- **Preconditions:** The database contains job application records.
- **Test Steps:**

1) Log in as a company representative.
2) Navigate to the Job Application Management section.
3) Sort a list of applications, marking some as accepted and others as rejected.
4) Check that the applications are correctly categorized.

- **Expected Result:** Applications are correctly moved to their respective categories and notifications are sent to candidates.
- **Status:** [Pass]

Student Module:

Test Case ID: UT-Student-01

- **Module:** Student Profile
- **Test Scenario:** Verify that a registered student can successfully update their profile information.
- **Preconditions:** The database contains student records.
- **Test Steps:**
  1) Log in as a registered student.
  2) Navigate to the profile section.
  3) Update profile information (e.g., contact details, educational background).
  4) Save the changes.
  5) Log out and log back in to verify the updated information.
- **Expected Result:** The updated profile information is saved correctly and displayed accurately.
- **Status:** [Pass]

Test Case ID: UT-Student-02

- **Module:** Job Application
- **Test Scenario:** Verify that a registered student can apply for a listed job.
- **Preconditions:** The database contains job vacancy records and student records.
- **Test Steps:**
  1) Log in as a registered student.
  2) Navigate to the job listings section.
  3) Select a job vacancy.

4) Apply for the job.
5) Verify that the application is submitted successfully.

- **Expected Result**: The application is successfully submitted and reflected in the student's application history.
- **Status:** [Pass]

Integration Testing:

Scenario 1: Admin Module Interaction with Company and Student Modules
Test Case ID: IT-01

- **Modules Involved:** Admin, Company, Student
- **Test Scenario:** Verify that the admin can view the list of registered companies and students, and generate reports.
- **Preconditions:** The database contains records of companies and students.
- **Test Steps:**
  1) Log in as admin.
  2) Navigate to the section to view registered companies.
  3) Verify that the list of companies is displayed correctly.
  4) Navigate to the section to view registered students.
  5) Verify that the list of students is displayed correctly.
  6) Generate a report for registered companies and students.
  7) Verify that the report contains accurate information.
- **Expected Result:** The admin can view lists and generate reports accurately.
- **Status:** [Pass]

Scenario 2: Company Module Interaction with Student Module
Test Case ID: IT-02

- **Modules Involved:** Company, Student
- **Test Scenario:** Verify that companies can view student profiles and send notifications.
- **Preconditions:** The database contains records of students.
- **Test Steps:**
  1) Log in as a company representative.

2) Navigate to the section to view student profiles.
3) Verify that student profiles are displayed correctly.
4) Send a notification to a student.
5) Log in as the student and check for the notification.
6) Verify that the notification is received and displayed correctly.

- **Expected Result:** Companies can view student profiles and send notifications, which are received correctly by students.
- **Status:** [Pass]

Scenario 3: Student Module Interaction with Job Vacancies

Test Case ID: IT-03

- **Modules Involved:** Student, Company
- **Test Scenario:** Verify that students can view and apply for job vacancies posted by companies.
- **Preconditions:** The database contains records of job vacancies posted by companies.
- **Test Steps:**
  1) Log in as a registered student.
  2) Navigate to the job listings section.
  3) Verify that job vacancies are displayed correctly.
  4) Apply for a job.
  5) Log in as a company representative and check the application.
  6) Verify that the application is received and displayed correctly.
- **Expected Result:** Students can view and apply for job vacancies, and companies can view received applications.
- **Status:** [Pass]

Scenario 4: End-to-End Workflow
Test Case ID: IT-04

- **Modules Involved:** Admin, Company, Student

- **Test Scenario:** Verify the complete workflow from admin registering a company, company posting a job, student applying for the job, and company processing the application.
- **Preconditions:** The database is initialized and empty.
- **Test Steps:**
  1) Admin registers a new company.
  2) Company logs in and posts a new job vacancy.
  3) Student logs in and applies for the job.
  4) Company processes the application (accepts/rejects).
  5) Student checks the application status.
- **Expected Result:** The entire workflow is executed without errors, and all interactions are recorded accurately.
- **Status:** [Pass]

Black Box Testing:

Scenario 1: Admin Login Functionality
Test Case ID: BB-01

- **Test Scenario:** Verify that the admin can log in with valid credentials.
- **Test Steps:**
  1) Navigate to the admin login page.
  2) Enter a valid username.
  3) Enter a valid password.
  4) Click the "Login" button.
- **Expected Result:** Admin is successfully logged in and redirected to the admin dashboard.
- **Status:** [Pass]

Scenario 2: Invalid Admin Login
Test Case ID: BB-02

- **Test Scenario:** Verify that the admin cannot log in with invalid credentials.
- **Test Steps:**
  1) Navigate to the admin login page.

   2) Enter an invalid username.
   3) Enter an invalid password.
   4) Click the "Login" button.
- **Expected Result:** Admin is not logged in, and an error message is displayed.
- **Status:** [Pass]

Scenario 3: Student Registration
Test Case ID: BB-03

- **Test Scenario:** Verify that a new student can register.
- **Test Steps:**
   1) Navigate to the student registration page.
   2) Enter all required information (name, email, password, etc.).
   3) Click the "Register" button.
- **Expected Result:** Student is successfully registered, and a confirmation message is displayed.
- **Status:** [Pass]

Scenario 4: Job Application by Student
Test Case ID: BB-04

- **Test Scenario:** Verify that a student can apply for a job.
- **Test Steps:**
   1) Log in as a registered student.
   2) Navigate to the job listings page.
   3) Select a job and click "Apply".
- **Expected Result:** The application is submitted successfully, and a confirmation message is displayed.
- **Status:** [Pass]

Scenario 5: Company Posting a Job
Test Case ID: BB-05

- **Test Scenario:** Verify that a company can post a job.
- **Test Steps:**

1) Log in as a company representative.
2) Navigate to the job posting section.
3) Enter job details and click "Post Job".

- **Expected Result:** The job is posted successfully, and a confirmation message is displayed.
- **Status:** [Pass]

Scenario 6: Viewing Applied Jobs
Test Case ID: BB-06

- **Test Scenario:** Verify that a student can view the jobs they have applied for.
- **Test Steps:**
  1) Log in as a registered student.
  2) Navigate to the "Applied Jobs" section.
- **Expected Result:** The list of jobs the student has applied for is displayed.
- **Status:** [Pass]

Scenario 1: Admin Login Functionality
Test Case ID: BB-01

- **Test Scenario:** Verify that the admin can log in with valid credentials.
- **Test Steps:**
  5) Navigate to the admin login page.
  6) Enter a valid username.
  7) Enter a valid password.
  8) Click the "Login" button.
- **Expected Result:** Admin is successfully logged in and redirected to the admin dashboard.
- **Status:** [Pass]

Scenario 2: Invalid Admin Login
Test Case ID: BB-02

- **Test Scenario:** Verify that the admin cannot log in with invalid credentials.
- **Test Steps:**

5) Navigate to the admin login page.
6) Enter an invalid username.
7) Enter an invalid password.
8) Click the "Login" button.

- **Expected Result:** Admin is not logged in, and an error message is displayed.
- **Status:** [Pass]

Scenario 3: Student Registration

Test Case ID: BB-03

- **Test Scenario:** Verify that a new student can register.
- **Test Steps:**
  4) Navigate to the student registration page.
  5) Enter all required information (name, email, password, etc.).
  6) Click the "Register" button.
- **Expected Result:** Student is successfully registered, and a confirmation message is displayed.
- **Status:** [Pass]

Scenario 4: Job Application by Student

Test Case ID: BB-04

- **Test Scenario:** Verify that a student can apply for a job.
- **Test Steps:**
  4) Log in as a registered student.
  5) Navigate to the job listings page.
  6) Select a job and click "Apply".
- **Expected Result:** The application is submitted successfully, and a confirmation message is displayed.
- **Status:** [Pass]

Scenario 5: Company Posting a Job

Test Case ID: BB-05

- **Test Scenario:** Verify that a company can post a job.

- **Test Steps:**
    4) Log in as a company representative.
    5) Navigate to the job posting section.
    6) Enter job details and click "Post Job".
- **Expected Result:** The job is posted successfully, and a confirmation message is displayed.
- **Status:** [Pass]

Scenario 6: Viewing Applied Jobs
Test Case ID: BB-06

- **Test Scenario:** Verify that a student can view the jobs they have applied for.
- **Test Steps:**
    3) Log in as a registered student.
    4) Navigate to the "Applied Jobs" section.
- **Expected Result:** The list of jobs the student has applied for is displayed.
- **Status:** [Pass]

White Box Testing:

Scenario 1: Admin Login Functionality

Test Case ID: WB-01

- **Objective:** Verify that the login function for admin checks username and password correctly.
- **Code Coverage:** Statement coverage, branch coverage.
- **Test Steps:**
    1) Check if the function verifies that both username and password are not null.
    2) Check if the function correctly matches the input with the stored admin credentials.
    3) Check if the function correctly handles failed login attempts.
- **Expected Result:** Proper validation and error messages for null inputs, correct matching of credentials, and handling of failed attempts.

- **Status:** [Pass]

Scenario 2: Student Registration

Test Case ID: WB-02

- **Objective:** Verify the student registration function for proper input validation and data storage.
- **Code Coverage:** Path coverage, statement coverage.
- **Test Steps:**
    1) Validate all input fields for correct data types and non-null values.
    2) Ensure that the function correctly stores the student data in the database.
    3) Check if the function handles duplicate registrations.
- **Expected Result:** All input validations pass, data is stored correctly, and duplicates are handled appropriately.
- **Status:** [Pass]

Scenario 3: Job Application by Student

Test Case ID: WB-03

- **Objective:** Verify the job application function for proper processing of job applications.
- **Code Coverage:** Statement coverage, decision coverage.
- **Test Steps:**
    1) Ensure the function checks if the student is logged in.
    2) Validate that the job ID exists and is valid.
    3) Check if the function correctly records the job application in the database.
- **Expected Result:** The student is authenticated, the job ID is validated, and the application is recorded.
- **Status:** [Pass]

Scenario 4: Company Posting a Job

Test Case ID: WB-04

- **Objective:** Verify the job posting function for correct input handling and data storage.
- **Code Coverage:** Path coverage, statement coverage.
- **Test Steps:**
    1) Validate all input fields for correct data types and required values.
    2) Ensure that the function correctly stores the job details in the database.
    3) Check if the function handles invalid inputs and displays appropriate error messages.
- **Expected Result:** All input validations pass, data is stored correctly, and invalid inputs are handled properly.
- **Status:** [Pass]

Scenario 5: Viewing Applied Jobs

Test Case ID: WB-05

- **Objective:** Verify the function that retrieves and displays the list of applied jobs for a student.
- **Code Coverage:** Statement coverage, branch coverage.
- **Test Steps:**
    1) Ensure the function checks if the student is logged in.
    2) Validate that the function retrieves the correct job applications from the database.
    3) Check if the function correctly formats and displays the job application list.
- **Expected Result:** The student is authenticated, the correct data is retrieved, and the list is displayed properly.
- **Status:** [Pass]

Scenario 6: Password Recovery

Test Case ID: WB-06

- **Objective:** Verify the password recovery function for proper input handling and email functionality.

- **Code Coverage:** Path coverage, decision coverage.
- **Test Steps:**
    1) Validate the input email for correct format and existence in the database.
    2) Ensure that the function generates a password recovery token.
    3) Check if the function sends the recovery email correctly.
- **Expected Result:** The email is validated, the token is generated, and the recovery email is sent properly.
- **Status:** [Pass]

5.8. SUMMARY

System testing is a critical phase in the software development lifecycle, ensuring that the Campus Recruitment System functions as intended and meets all specified requirements. This phase involves various testing techniques and processes designed to identify and rectify defects before the system is deployed. The testing process starts with defining the objectives and scope, emphasizing the intent to find and fix errors. Unit testing, the first step, focuses on the smallest units of code, ensuring each module functions correctly in isolation. Integration testing follows, where individual modules are combined and tested as a group to identify issues in their interactions. Black box testing evaluates the system's functionality without peering into its internal structures, verifying that inputs produce the expected outputs. Conversely, white box testing examines the internal workings of the application, ensuring that all possible code paths are executed and error-free.

Test cases play a pivotal role in system testing, providing specific scenarios to validate different functionalities of the system. These test cases ensure comprehensive coverage, testing various user interactions and system responses. By meticulously executing these tests, the development team can identify and address issues related to data flow, security, usability, and performance. Performance testing, another critical aspect, measures the system's responsiveness and stability under various conditions, ensuring it can handle real-world usage scenarios.

Overall, system testing integrates various methodologies and techniques to deliver a robust, reliable, and user-friendly Campus Recruitment System. It ensures that the system not only meets the functional requirements but also provides a seamless and secure experience for administrators, companies, and students. The summary of the testing phase highlights the thoroughness and depth of the process, reflecting the commitment to quality and excellence in software development.

CHAPTER – 6

RESULT AND DISCUSSION

6. RESULT AND DISCUSSION [10]

The Campus Recruitment System has significantly streamlined campus placements, automating tasks like data entry, application sorting, and communication, thus reducing manual effort, and saving time. Administrators can now efficiently manage user accounts and placements, while recruiters benefit from easier job postings and candidate shortlisting. Students enjoy a simplified application process with real-time tracking and updates, improving their chances of securing placements. The system also ensures secure and accurate data handling.

This system reduces administrative burdens for educational institutions, potentially increasing placement success rates, while giving companies access to a larger pool of qualified candidates and providing students with more transparency. Challenges such as adoption, data privacy, and continuous improvement remain, but future enhancements like advanced analytics, mobile accessibility, and AI integration could further optimize the process. Overall, the system modernizes and enhances the efficiency of campus placements.

6.1 Home Page

The homepage of the Campus Recruitment System for Narayana Engineering College features a user-friendly interface designed to streamline the recruitment process for all stakeholders. At the top, users can find options to "Sign In" or "Register," allowing both new and returning users to access their accounts easily. The navigation bar includes links to various sections such as "Home," "About Us," "Listed Job," and "Contact," providing a comprehensive overview of the system's capabilities. These links help users navigate the platform seamlessly and access essential information and services.

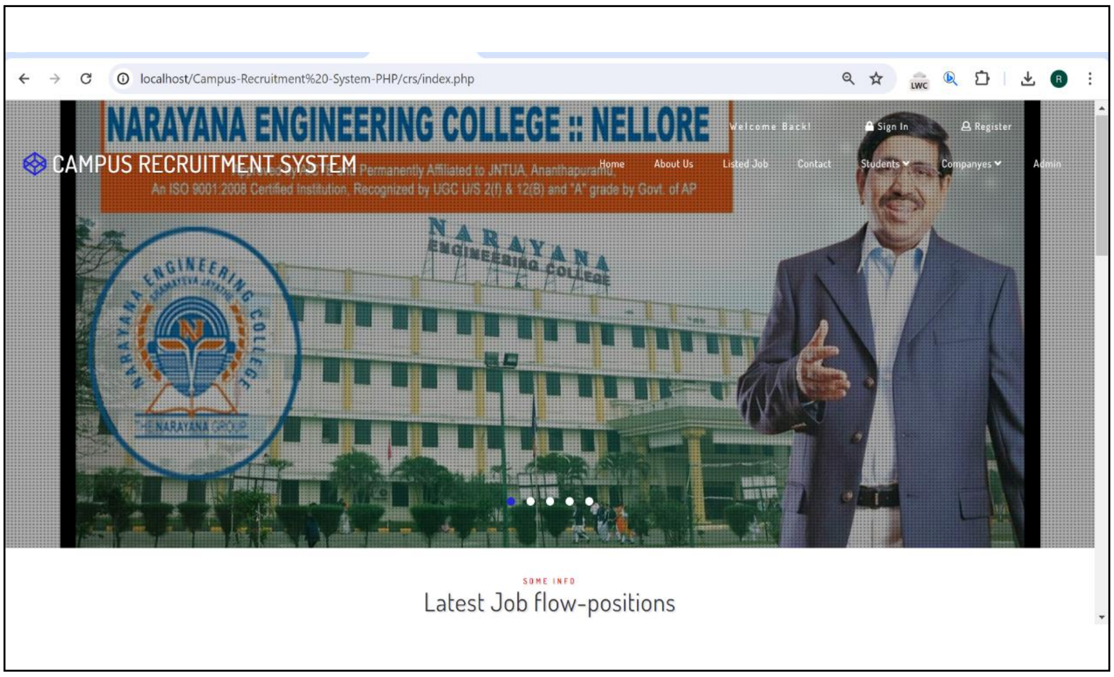

localhost/Campus-Recruitment%20-System-PHP/crs/index.php
NARAYANA ENGINEERING COLLEGE :: NELLORE
CAMPUS RECRUITMENT SYSTEM
Welcome Back!
Sign In
Register
Home
About Us
Listed Job
Contact
Students
Companyes
Admin
SOME INFO
Latest Job flow-positions

Figure 6.1: Home Screen

For students, the system offers a dedicated "Students" dropdown menu, likely featuring tools and resources tailored to their needs, such as job application tracking, resume building, and interview preparation. Similarly, companies have a "Companies" dropdown menu that probably includes functionalities for posting job openings, reviewing applications, and shortlisting candidates. The "Admin" section is designed for administrative users who manage the overall system, ensuring smooth operations and user management. This modular approach ensures that each user group has access to the specific features they need, enhancing the efficiency and effectiveness of the recruitment process.

Additionally, the page prominently displays a section titled "Latest Job flow-positions," which highlights the most recent job openings and application updates. This feature keeps users informed about new opportunities and their application statuses, promoting timely and effective communication. By providing real-time updates and a clear overview of available positions, the system enhances transparency and helps students stay on top of their job search. Overall, the design and features of this webpage aim to create a seamless and efficient recruitment experience for students, companies, and administrators.

6.2 Student Portal

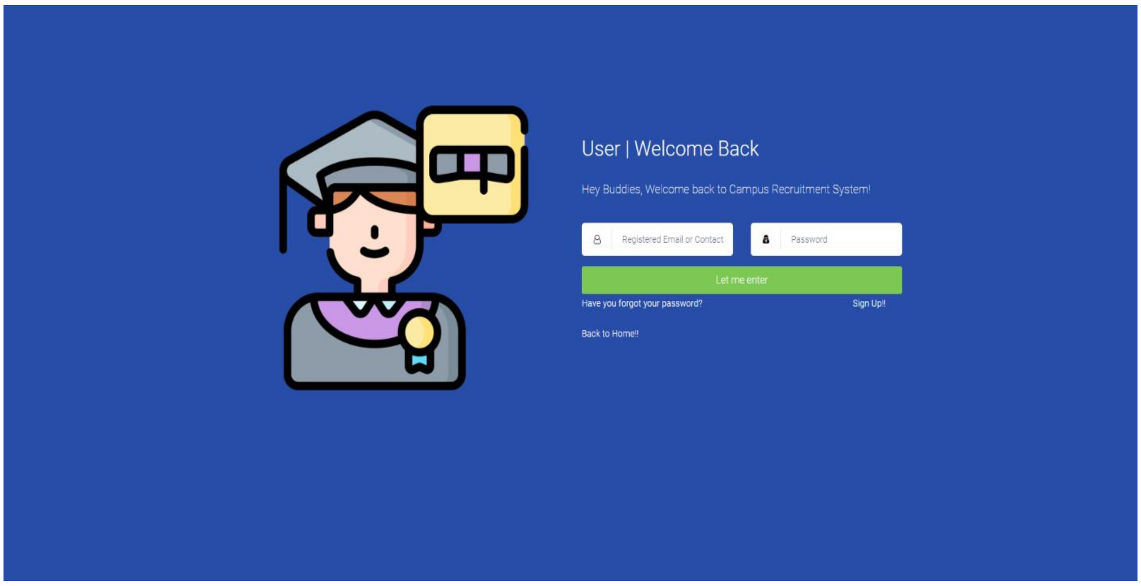
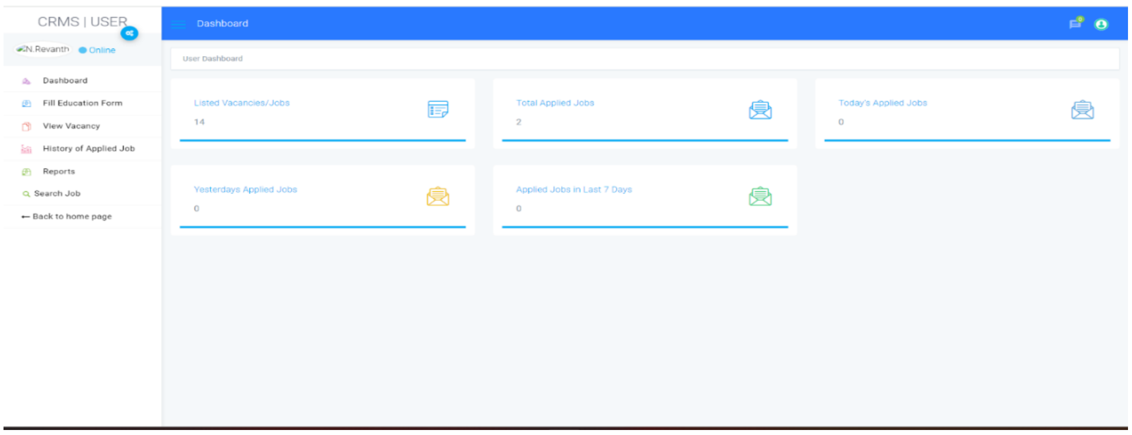

User | Welcome Back
Hey Buddies, Welcome back to Campus Recruitment System!
Registered Email or Contact
Password
Let me enter
Have you forgot your password?
Sign Up!!
Back to Home!!

Figure 6.2: Student Login

The login page for the Campus Recruitment Management system features fields for users to enter their registered email or contact information and password, along with a green "Let me enter" button to submit their credentials. It includes a link for password recovery, allowing users to reset forgotten passwords, and a "Sign Up" link for new users to create an account. Additionally, there is a "Back to Home!!" link for easy navigation back to the homepage. The page also displays a personalized welcome message for returning users, enhancing the user experience.

6.2.1: Student Dashboard



Listed Vacancies/Jobs
14

Total Applied Jobs
2

Today's Applied Jobs
0

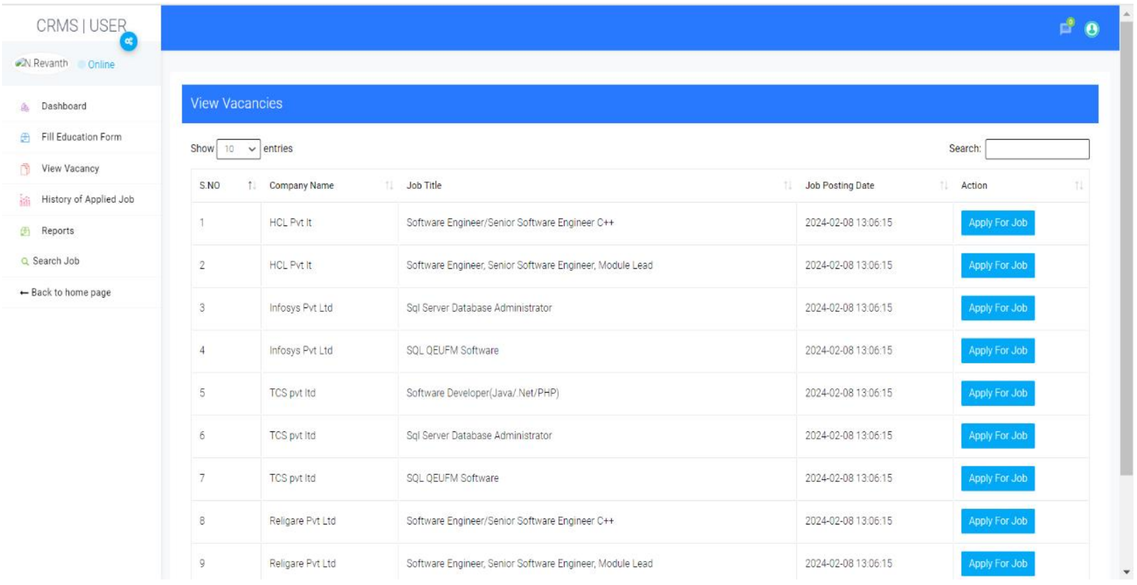
Yesterdays Applied Jobs
0

Applied Jobs in Last 7 Days
0

Figure 6.2.1 Student Dashboard

Campus Recruitment System (CRS), which is designed to streamline the campus placement process. The dashboard provides a summary of job-related activities, displaying metrics such as the number of listed vacancies , total applied jobs , today's applied jobs , yesterday's applied jobs and jobs applied for in the last 7 days. The navigation menu on the left offers various functionalities, including filling out an education form to manage educational information, viewing available job vacancies, checking the history of applied jobs, generating reports related to job applications and placements, and searching for jobs. The user information section at the top left displays the user's online status, with an icon for accessing settings and preferences. This comprehensive interface allows students to efficiently manage their job applications, track their progress, and access all necessary information and features in one centralized platform.

6.2.2. View Vacancies



View Vacancies

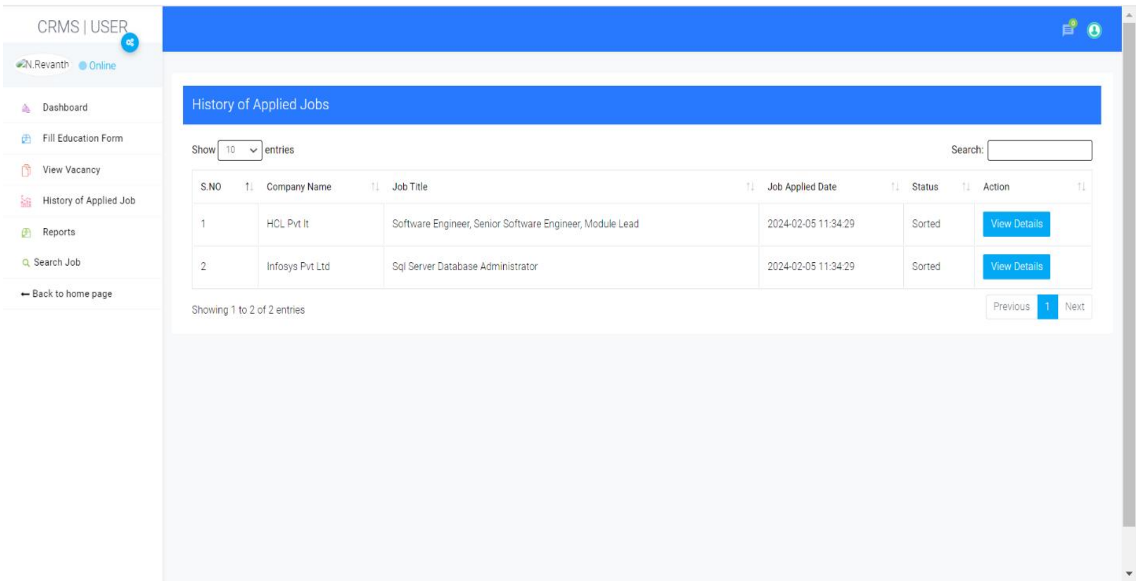
Show 10 entries Search:

S.NO	Company Name	Job Title	Job Posting Date	Action
1	HCL Pvt lt	Software Engineer/Senior Software Engineer C++	2024-02-08 13:06:15	Apply For Job
2	HCL Pvt lt	Software Engineer, Senior Software Engineer, Module Lead	2024-02-08 13:06:15	Apply For Job
3	Infosys Pvt Ltd	Sql Server Database Administrator	2024-02-08 13:06:15	Apply For Job
4	Infosys Pvt Ltd	SQL QEUFM Software	2024-02-08 13:06:15	Apply For Job
5	TCS pvt ltd	Software Developer(Java/.Net/PHP)	2024-02-08 13:06:15	Apply For Job
6	TCS pvt ltd	Sql Server Database Administrator	2024-02-08 13:06:15	Apply For Job
7	TCS pvt ltd	SQL QEUFM Software	2024-02-08 13:06:15	Apply For Job
8	Religare Pvt Ltd	Software Engineer/Senior Software Engineer C++	2024-02-08 13:06:15	Apply For Job
9	Religare Pvt Ltd	Software Engineer, Senior Software Engineer, Module Lead	2024-02-08 13:06:15	Apply For Job

Figure 6.2.2. View Vacancies

The "View Vacancies" page of the Campus Recruitment Management System is designed to help students efficiently browse and apply for job opportunities. Users can view a list of job vacancies, including company names, job titles, and posting dates. Each entry has an "Apply For Job" button, allowing users to submit their applications directly through the platform. The left sidebar provides easy access to various features like filling out educational forms, viewing the history of applied jobs, generating reports, and searching for specific jobs. This page streamlines the job search process, making it simple for students to find and apply for positions that match their qualifications and interests.

6.2.3. Applied Jobs



History of Applied Jobs

S.NO	Company Name	Job Title	Job Applied Date	Status	Action
1	HCL Pvt lt	Software Engineer, Senior Software Engineer, Module Lead	2024-02-05 11:34:29	Sorted	View Details
2	Infosys Pvt Ltd	Sql Server Database Administrator	2024-02-05 11:34:29	Sorted	View Details

Showing 1 to 2 of 2 entries

Figure 6.2.3. Applied Jobs

The "History of Applied Jobs" feature in the Campus Recruitment System allows users to track their job application history efficiently. This section provides a detailed record of all the jobs a user has applied for, including the company name, job title, application date, and current status. Additionally, the "View Details" button enables users to access more comprehensive information about each application. This feature helps users manage their job applications by offering a clear overview of their past activities, ensuring they can monitor the progress and outcomes of their applications. It enhances transparency and organization, allowing users to stay informed about their application status and follow up as needed.

6.3 Company Portal

The company portal in the Campus Recruitment System is designed to streamline and enhance the hiring process for companies. It allows employers to create and post job listings with detailed descriptions and qualifications, manage and review applications, and shortlist candidates for further recruitment stages. The portal facilitates communication between companies and candidates through messaging and notification features, ensuring timely interactions. Companies can update their profiles and preferences, and use analytical tools and reports to evaluate the effectiveness of their recruitment efforts. Additionally, the portal supports collaborative features, enabling multiple recruiters from the same company to coordinate and manage the recruitment process efficiently. This centralized platform ultimately helps companies attract and recruit top talent from educational institutions.

6.3.1. Company Dashboard

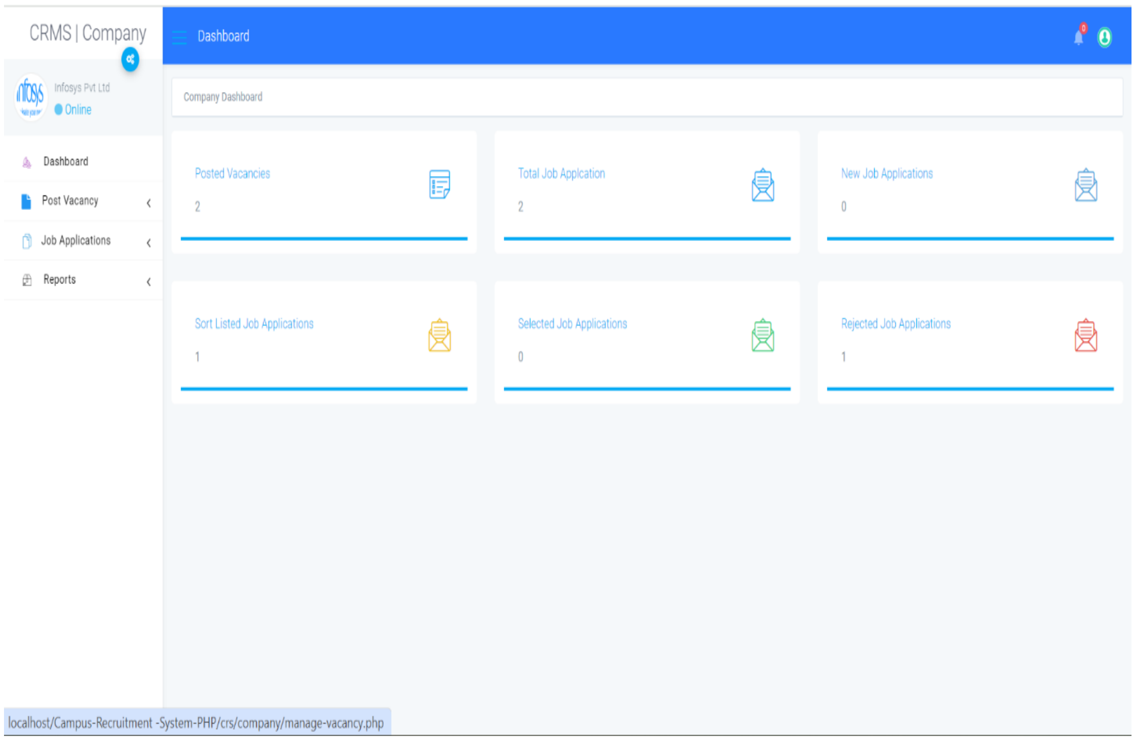
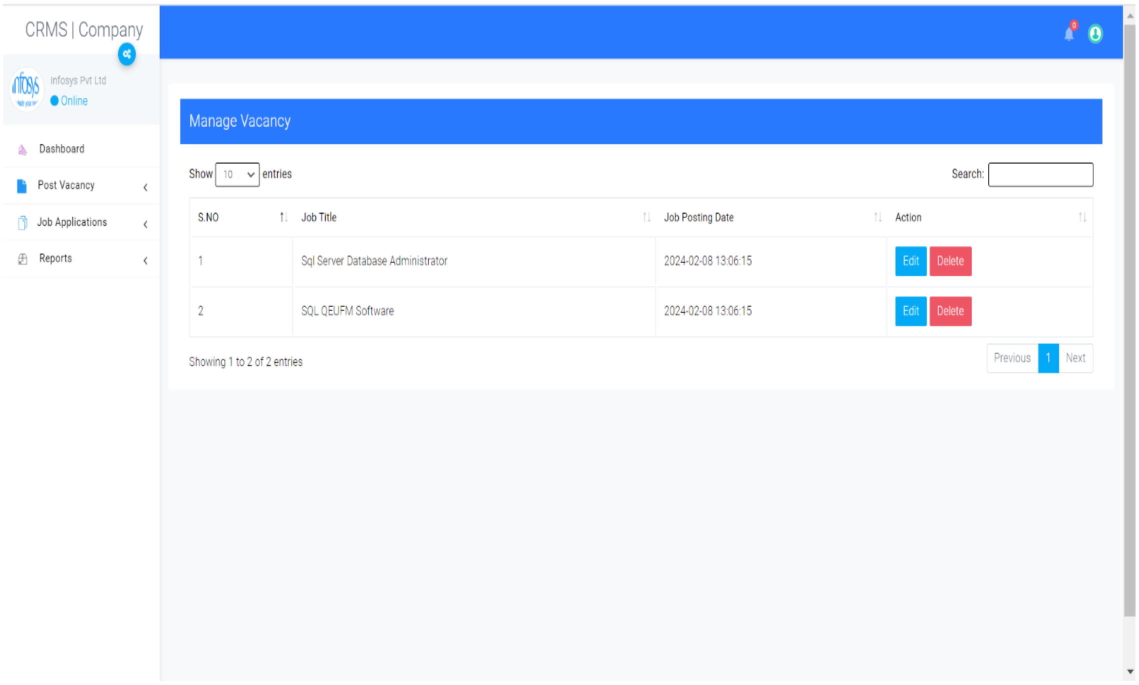


Figure 6.3.1:Company Dashboard

The company portal dashboard of the Campus Recruitment System (CRS), highlighting several key features. The dashboard includes metrics such as the number of posted vacancies , total job applications received , new job applications , shortlisted job applications , selected job applications , and rejected job applications . These features provide companies with a comprehensive overview of their recruitment activities. The navigation menu on the left offers additional functionalities, including posting new vacancies, viewing and managing job applications, and accessing various reports related to job postings and applications. This user-friendly interface enables companies to efficiently handle their recruitment processes within the CRS platform.

6.3.2. Manage Vacancies



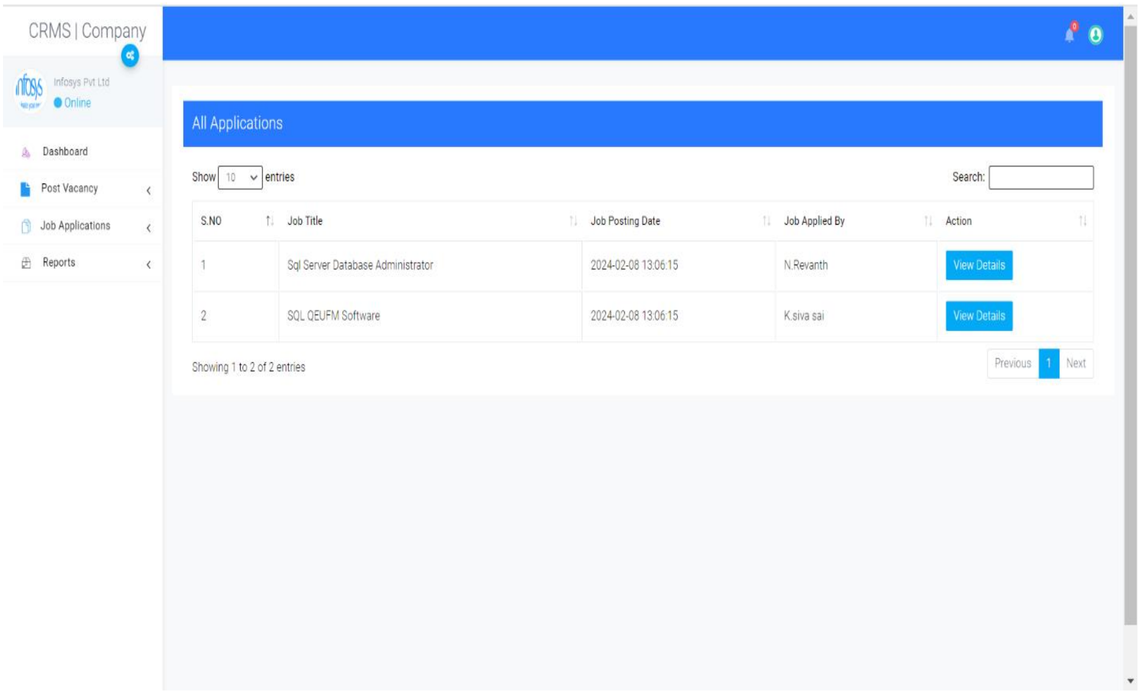
S.NO	Job Title	Job Posting Date	Action
1	Sql Server Database Administrator	2024-02-08 13:06:15	Edit Delete
2	SQL QEUFM Software	2024-02-08 13:06:15	Edit Delete

Showing 1 to 2 of 2 entries

Figure 6.3.2. Manage Vacancies

The "Manage Vacancies" feature in the Campus Recruitment System (CRS) project is designed to streamline the process of posting and managing job openings for companies. This feature allows companies to create, update, and delete job postings through a user-friendly interface. Companies can specify detailed job descriptions, required qualifications, and application deadlines, ensuring that all necessary information is communicated to potential candidates. Additionally, this feature enables companies to view and track the status of each vacancy, including the number of applications received, shortlisted candidates, and final selections. By providing a centralized platform for vacancy management, the CRS project enhances the efficiency of the recruitment process, making it easier for companies to attract and manage potential hires. This reduces administrative burden and helps ensure that all job postings are up-to-date and accessible to students seeking employment opportunities.

6.3.3. Job Applications



All Applications

Show 10 entries

Search:

S.NO	Job Title	Job Posting Date	Job Applied By	Action
1	Sql Server Database Administrator	2024-02-08 13:06:15	N.Revanth	View Details
2	SQL QEUFM Software	2024-02-08 13:06:15	K.siva sai	View Details

Showing 1 to 2 of 2 entries

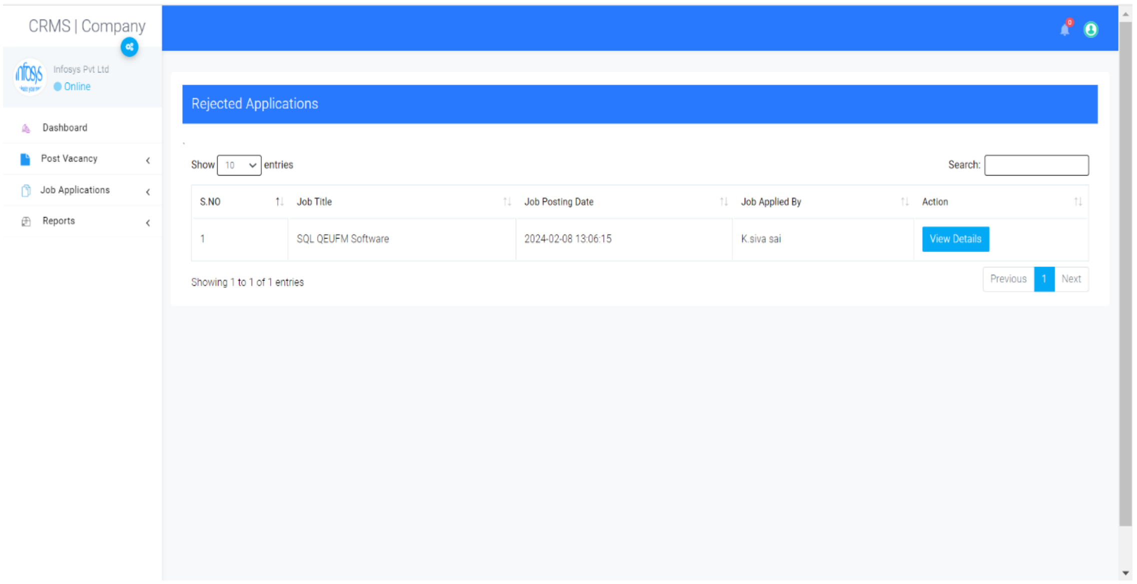
Figure 6.3.3. Job Applications

The "Job Applications" feature in the Campus Recruitment System (CRS) project is an essential tool for managing the entire lifecycle of job applications. This feature allows companies to view, sort, and evaluate applications submitted by students. When a student applies for a job, their application is automatically recorded in the system, where it can be accessed and reviewed by the company's recruitment team.

Within the "Job Applications" section, companies can see detailed information about each applicant, including their resume, cover letter, and any other relevant documents. The system also allows recruiters to sort applications based on various criteria, such as qualifications, experience, or the date of submission. This helps in efficiently identifying the most suitable candidates for further steps in the hiring process.

Overall, the "Job Applications" feature in CRS simplifies and organizes the recruitment process, making it easier for companies to manage and assess applications, and for students to track the progress of their job applications.

6.3.4. Rejected Applications



Rejected Applications

Show 10 entries Search:

S.NO	Job Title	Job Posting Date	Job Applied By	Action
1	SQL QEUFM Software	2024-02-08 13:06:15	K.siva sai	View Details

Showing 1 to 1 of 1 entries

Previous 1 Next

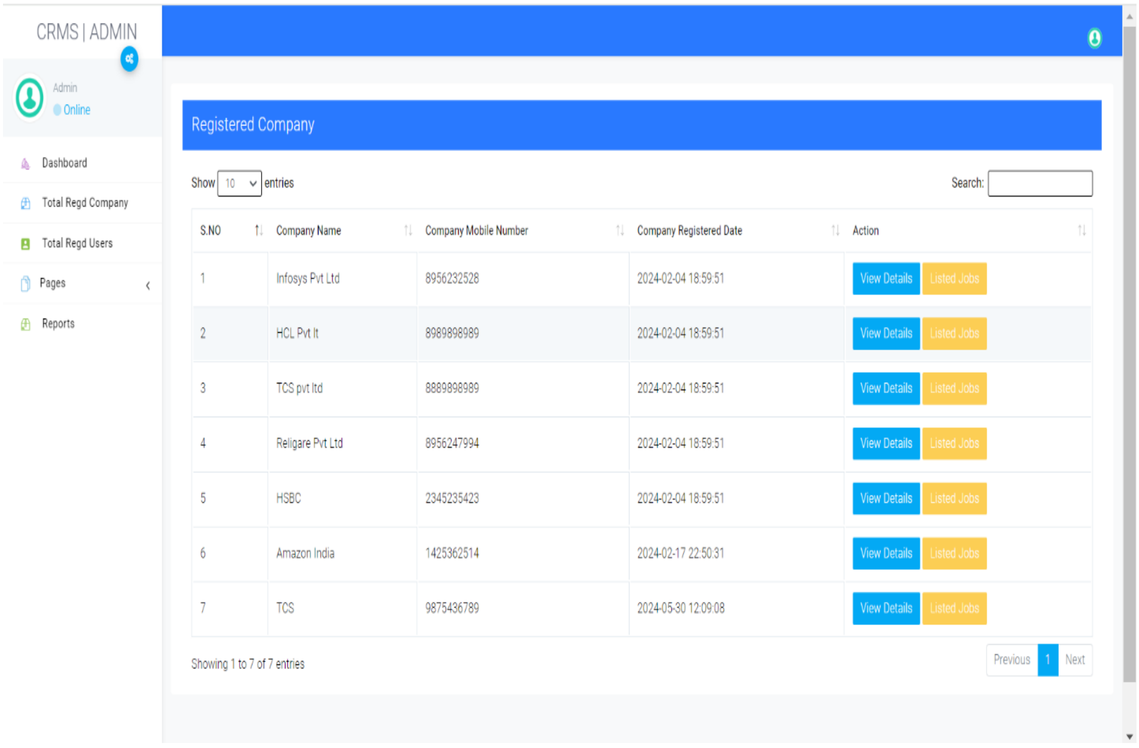
Figure 6.3.4. Rejected Applications

The "Rejected Applications" feature in the company portal of the Campus Recruitment System (CRS) allows companies to efficiently manage and communicate rejections for job applications. When an application is deemed unsuitable, recruiters can mark it as rejected, moving it to a designated section where it is stored for future reference. This feature helps keep the workflow organized by separating rejected applications from others and ensures transparency by providing clear status updates to applicants. Automated notifications inform candidates of their application's outcome, maintaining professionalism and courtesy. Overall, this feature streamlines the recruitment process, allowing companies to handle high volumes of applications effectively and focus on potential candidates while ensuring all applicants are promptly informed about their status.

6.4 Admin Portal

The admin portal in the Campus Recruitment System (CRS) is a comprehensive control hub designed to manage and oversee all aspects of the recruitment process within an educational institution. This portal equips administrators with powerful tools to handle user accounts, including students, companies, and other administrative users. Administrators can create, update, and delete accounts, ensuring that the user base is accurately maintained and secure. Additionally, the portal allows admins to monitor and manage job postings from various companies, ensuring they meet the institution's standards and requirements. Admins can review and approve new job vacancies, edit existing ones, and remove outdated or inappropriate listings, maintaining a high-quality job board for students.

6.4.1. Registered Company



Registered Company

Show 10 entries Search:

S.NO	Company Name	Company Mobile Number	Company Registered Date	Action
1	Infosys Pvt Ltd	8956232528	2024-02-04 18:59:51	View Details Listed Jobs
2	HCL Pvt lt	8989898989	2024-02-04 18:59:51	View Details Listed Jobs
3	TCS pvt ltd	8889898989	2024-02-04 18:59:51	View Details Listed Jobs
4	Religare Pvt Ltd	8956247994	2024-02-04 18:59:51	View Details Listed Jobs
5	HSBC	2345235423	2024-02-04 18:59:51	View Details Listed Jobs
6	Amazon India	1425362514	2024-02-17 22:50:31	View Details Listed Jobs
7	TCS	9875436789	2024-05-30 12:09:08	View Details Listed Jobs

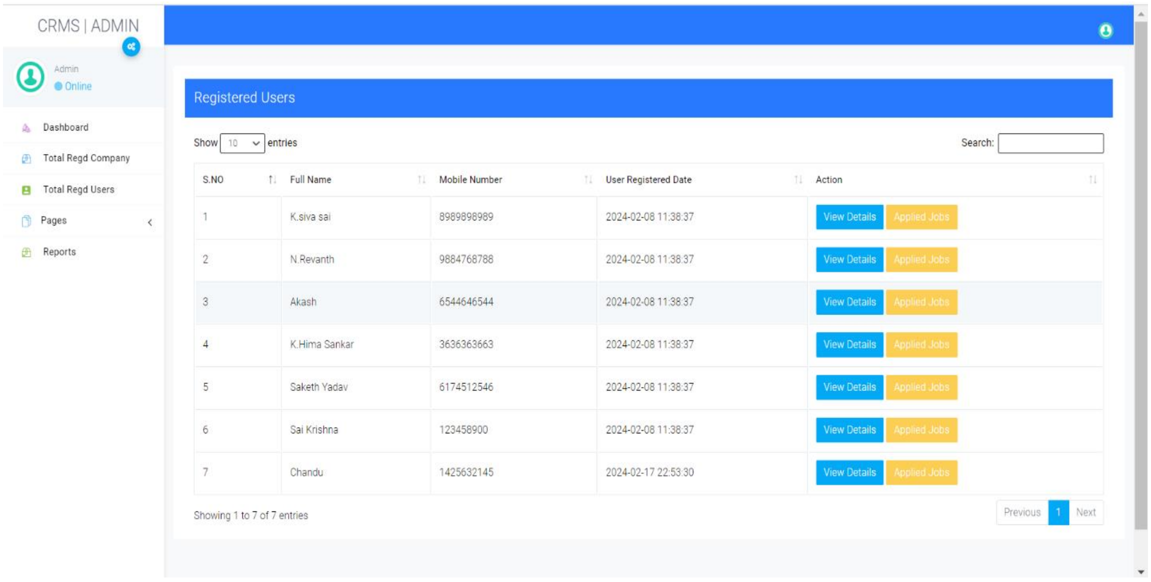
Showing 1 to 7 of 7 entries

Previous 1 Next

Figure 6.4.1: Registered Company

The "Registered Companies" feature in the admin portal of the Campus Recruitment System (CRS) allows administrators to manage and oversee the companies participating in the campus recruitment process. This feature provides a centralized list of all registered companies, including detailed profiles with essential information such as company name, contact details, industry type, and job posting history. Administrators can review and verify the authenticity of each company's registration, ensuring that only legitimate and reputable companies participate in the recruitment process. They can also edit company details, approve or reject new company registrations, and track the performance and engagement of each company with students. This comprehensive management capability helps maintain the quality and integrity of the recruitment platform, providing students with reliable opportunities and fostering strong relationships between the educational institution and recruiting companies.

6.4.2. Registered Users



Registered Users

Show 10 entries Search:

S.NO	Full Name	Mobile Number	User Registered Date	Action
1	K.siva sai	8989898989	2024-02-08 11:38:37	View Details Applied Jobs
2	N.Revanth	9884768788	2024-02-08 11:38:37	View Details Applied Jobs
3	Akash	6544646544	2024-02-08 11:38:37	View Details Applied Jobs
4	K.Hima Sankar	3636363663	2024-02-08 11:38:37	View Details Applied Jobs
5	Saketh Yadav	6174512546	2024-02-08 11:38:37	View Details Applied Jobs
6	Sai Krishna	123458900	2024-02-08 11:38:37	View Details Applied Jobs
7	Chandu	1425632145	2024-02-17 22:53:30	View Details Applied Jobs

Showing 1 to 7 of 7 entries

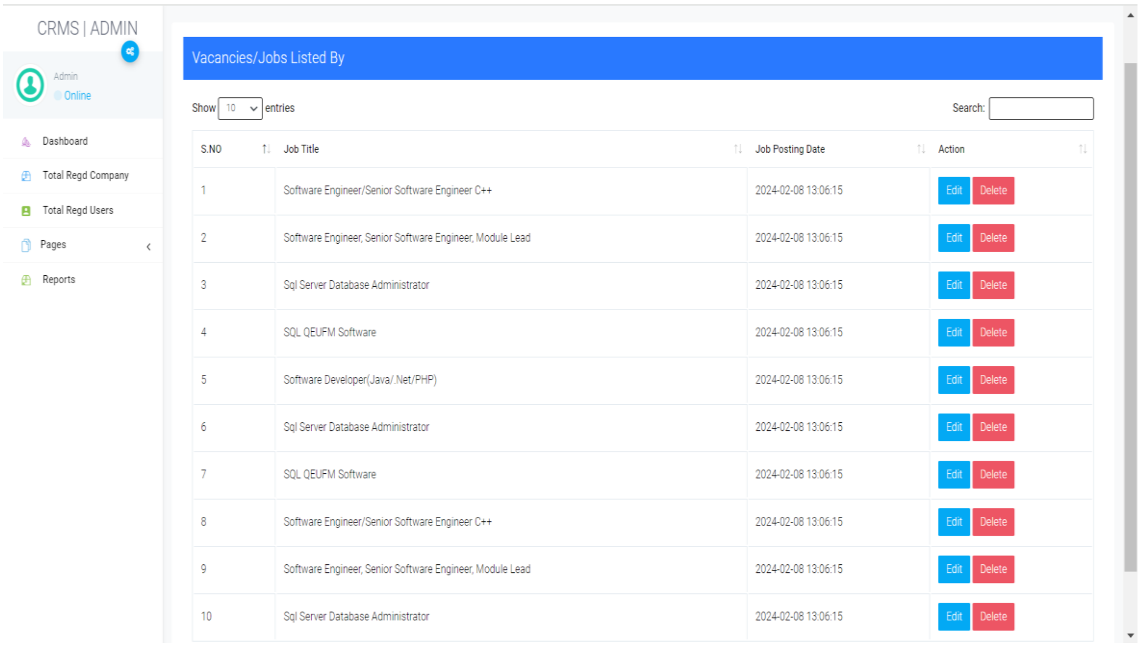
Previous 1 Next

Figure 6.4.2. Registered Users

The "Registered Users" feature in the admin portal of the Campus Recruitment System (CRS) provides administrators with the ability to manage and monitor all user accounts within the system, including students, companies, and other administrative personnel. This feature displays a comprehensive list of all registered users, complete with detailed profiles containing vital information such as user names, contact details, roles, and registration dates. Administrators can efficiently manage these accounts by creating, updating, or deleting user profiles as needed to ensure the accuracy and security of the system. They can also verify user credentials, approve or reject new user registrations, and track user activity and engagement within the platform. This functionality is essential for maintaining the integrity of the recruitment process, ensuring that all users are properly authenticated and that the system operates smoothly and securely.

6.4.3. Vacancies/Jobs Listed By

The "Posted Vacancies and Jobs" feature in the admin portal of the Campus Recruitment System (CRS) enables administrators to oversee and manage all job listings posted by companies on the platform. This feature provides a centralized view of all active and past job vacancies, complete with detailed information such as job titles, descriptions, requirements, posting dates, application deadlines, and the number of applications received. Administrators can review these details to ensure that job postings are clear, accurate, and adhere to the institution's standards and policies. They have the authority to approve new job listings before they become visible to students, edit existing postings to correct any discrepancies or update information, and remove outdated or inappropriate listings to maintain the quality and relevance of the job board.



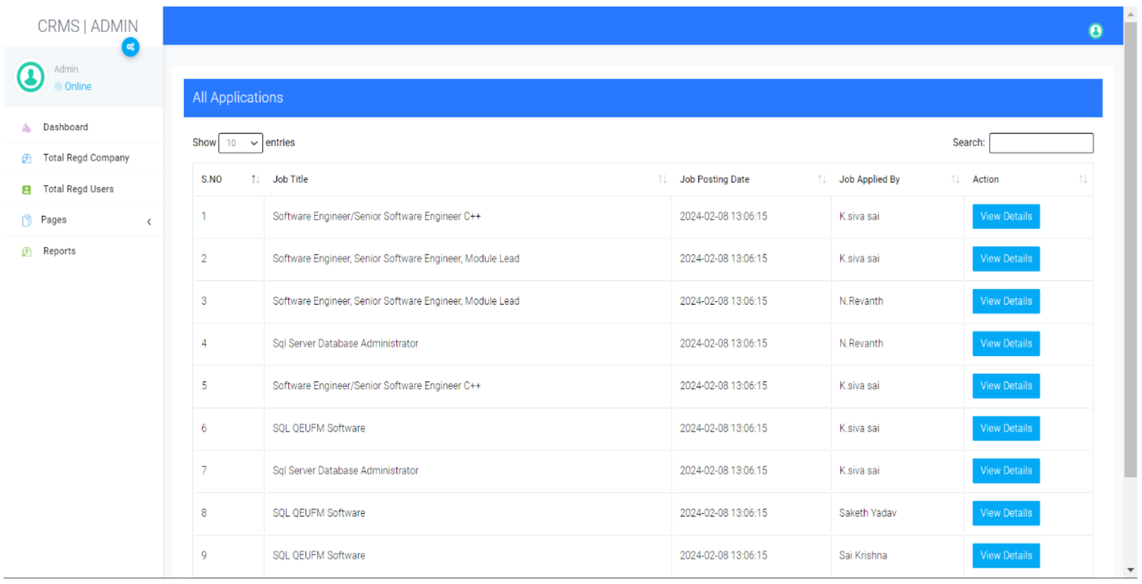
S.NO	Job Title	Job Posting Date	Action
1	Software Engineer/Senior Software Engineer C++	2024-02-08 13:06:15	Edit Delete
2	Software Engineer, Senior Software Engineer, Module Lead	2024-02-08 13:06:15	Edit Delete
3	Sql Server Database Administrator	2024-02-08 13:06:15	Edit Delete
4	SQL QEUFM Software	2024-02-08 13:06:15	Edit Delete
5	Software Developer(Java/.Net/PHP)	2024-02-08 13:06:15	Edit Delete
6	Sql Server Database Administrator	2024-02-08 13:06:15	Edit Delete
7	SQL QEUFM Software	2024-02-08 13:06:15	Edit Delete
8	Software Engineer/Senior Software Engineer C++	2024-02-08 13:06:15	Edit Delete
9	Software Engineer, Senior Software Engineer, Module Lead	2024-02-08 13:06:15	Edit Delete
10	Sql Server Database Administrator	2024-02-08 13:06:15	Edit Delete

Figure 6.4.3: Vacancies/Jobs Listed By

6.4.4. All Applications

The "All Applications" feature in the admin portal of the Campus Recruitment System (CRS) serves as a comprehensive tool for administrators to oversee and manage the entire spectrum of job applications submitted by students.

This feature provides a centralized repository where administrators can view all applications in detail, including information such as applicant names, the positions they applied for, submission dates, and status (e.g., pending, shortlisted, rejected, or accepted). By having access to this detailed information, administrators can monitor the progress of applications, ensuring that each step of the recruitment process is proceeding smoothly and efficiently. They can also identify any bottlenecks or issues that might arise, such as delays in processing applications or a lack of communication between companies and applicants.



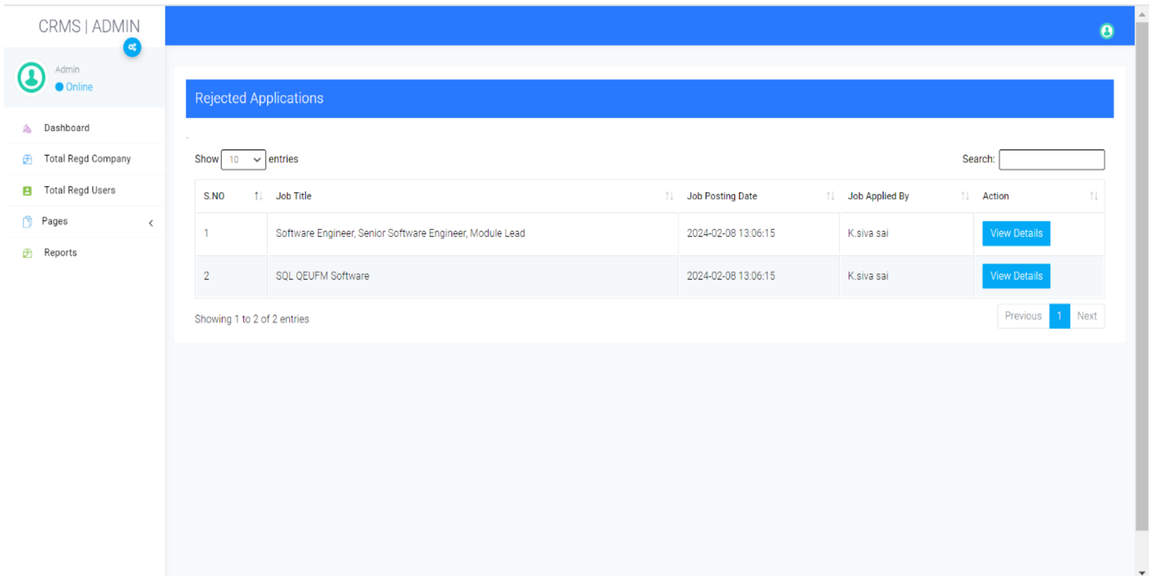
S.NO	Job Title	Job Posting Date	Job Applied By	Action
1	Software Engineer/Senior Software Engineer C++	2024-02-08 13:06:15	K.siva sai	View Details
2	Software Engineer, Senior Software Engineer, Module Lead	2024-02-08 13:06:15	K.siva sai	View Details
3	Software Engineer, Senior Software Engineer, Module Lead	2024-02-08 13:06:15	N.Revanth	View Details
4	Sql Server Database Administrator	2024-02-08 13:06:15	N.Revanth	View Details
5	Software Engineer/Senior Software Engineer C++	2024-02-08 13:06:15	K.siva sai	View Details
6	SQL QEUFM Software	2024-02-08 13:06:15	K.siva sai	View Details
7	Sql Server Database Administrator	2024-02-08 13:06:15	K.siva sai	View Details
8	SQL QEUFM Software	2024-02-08 13:06:15	Saketh Yadav	View Details
9	SQL QEUFM Software	2024-02-08 13:06:15	Sai Krishna	View Details

Figure 6.4.4: All Applications

Administrators can also use this feature to ensure compliance with institutional policies and standards, verifying that all applications are handled fairly and transparently. By maintaining a thorough and organized overview of all job applications, the "All Applications" feature enhances the ability of administrators to support both students and companies, contributing to a more efficient and effective campus recruitment system.

6.4.5. Rejected Applications

The "Rejected Applications" feature in the admin portal of the Campus Recruitment System (CRS) allows administrators to manage and review applications that have been marked as rejected by recruiters. This feature provides a detailed list of all rejected applications, including applicant names, the positions they applied for, the reasons for rejection, and the dates they were rejected.



Rejected Applications

Show 10 entries | Search:

S.NO	Job Title	Job Posting Date	Job Applied By	Action
1	Software Engineer, Senior Software Engineer, Module Lead	2024-02-08 13:06:15	K.siva sai	View Details
2	SQL QEUFM Software	2024-02-08 13:06:15	K.siva sai	View Details

Showing 1 to 2 of 2 entries

Previous 1 Next

Figure 6.4.5: Rejected Applications

Administrators can use this information to ensure that the rejection process is fair and consistent, verifying that all candidates are evaluated based on the same criteria. Additionally, this feature enables administrators to track patterns or trends in rejections, which can help identify potential areas for improvement in the recruitment process or in student preparation. By maintaining oversight of rejected applications, administrators can ensure transparency and accountability within the system, and provide feedback or support to students to help them improve their future applications.

CHAPTER - 7
CONCLUSION & FUTURE SCOPE

7. CONCLUSION & FUTURE SCOPE

7.1. CONCLUSION [22]

The Campus Recruitment System project has been a transformative initiative aimed at modernizing the recruitment processes within educational institutions. This project encompasses the development and implementation of a comprehensive platform that serves as an interface between students, companies, and campus administrators, streamlining the various stages of recruitment from job postings to final placements.

From the outset, the project was driven by a need to address inefficiencies and bottlenecks in traditional recruitment processes. By leveraging advanced technologies, the system offers a user-friendly and intuitive interface that facilitates seamless interactions among all stakeholders. Students can effortlessly browse job listings, apply for positions, and track their application status. Companies, on the other hand, can post job vacancies, manage applications, and communicate with potential candidates efficiently. Campus administrators benefit from enhanced oversight capabilities, allowing them to monitor recruitment activities, generate insightful reports, and ensure a smooth operation throughout the recruitment cycle.

A major highlight of the project is the robust testing regime that was employed to ensure the system's reliability and performance. Unit testing validated the smallest components of the software, ensuring they functioned correctly in isolation. Integration testing verified that different modules of the system worked together as intended, while black box and white box testing methods were used to scrutinize the system from both external and internal perspectives. These rigorous testing processes uncovered and addressed numerous potential issues, resulting in a system that is both stable and secure.

Data security and privacy were prioritized throughout the development process. With sensitive information such as student profiles, company details, and job application data being managed by the system, stringent security measures were implemented. These included data encryption, secure authentication mechanisms, and strict access control protocols, all designed to protect user data from unauthorized access and breaches.

The project also faced and overcame significant challenges. Gathering comprehensive requirements from a diverse user base required extensive consultations and iterations.

Integrating the new system with existing institutional frameworks involved careful planning and execution, ensuring seamless data migration and interoperability. Additionally, user training and support mechanisms were put in place to facilitate the adoption of the system, including detailed user manuals and responsive helpdesk support.

Looking ahead, the Campus Recruitment System has the potential for further enhancements. Incorporating advanced analytics and AI capabilities could provide deeper insights and more personalized experiences for users. Developing a mobile application would enhance accessibility, allowing users to engage with the system on-the-go. Continuous updates and improvements, guided by user feedback and technological advancements, will ensure that the system remains relevant and effective.

In conclusion, the Campus Recruitment System project has successfully created a robust, efficient, and user-centric platform that revolutionizes campus recruitment processes. By bridging the gap between students, companies, and administrators, it fosters a more efficient and transparent recruitment environment. The system not only meets current needs but is also adaptable for future enhancements, positioning it as a critical tool for educational institutions aiming to optimize their recruitment strategies and outcomes. This project stands as a testament to the power of innovative technology in transforming traditional processes, delivering significant benefits to all stakeholders involved.

7.2. SCOPE FOR FUTURE WORK [8]

The Campus Recruitment System, while comprehensive and effective in its current state, has ample potential for further enhancements and expansion. Several avenues for future work can be pursued to ensure the system remains state-of-the-art, meets evolving user needs, and leverages emerging technologies. The following outlines key areas for future work:

1. **Advanced Analytics and Reporting:**
    - **AI and Machine Learning Integration:** Implementing AI and machine learning algorithms can provide predictive analytics for job market trends, personalized job recommendations for students, and hiring trends for companies.

- **Enhanced Reporting:** Develop more sophisticated reporting tools that allow for customized reports, real-time data visualization, and deeper insights into recruitment metrics.

2. **Mobile Application Development:**
   - **Cross-Platform Mobile App**: Creating a mobile application for iOS and Android would enhance accessibility, allowing users to interact with the system anytime and anywhere. The app could offer push notifications for job alerts, interview schedules, and updates.
   - **Mobile-Specific Features:** Incorporate features such as location-based job recommendations, mobile resume upload, and quick apply options to improve the user experience.
3. **User Experience Improvements:**
   - **User Interface Enhancements:** Continuously improve the user interface based on user feedback to ensure it remains intuitive and easy to navigate. Implementing a responsive design will ensure the system works well on various devices.
   - **Personalization:** Enhance personalization features to tailor the user experience based on individual user behavior and preferences.
4. **Integration with Social Media and Professional Networks:**
   - **Social Media Integration:** Allow users to connect their profiles with social media platforms such as LinkedIn, enabling easy import of profile information and job application sharing.
   - **Networking Features:** Introduce networking features that allow students to connect with alumni, mentors, and industry professionals.
5. **Enhanced Security Measures:**
   - **Advanced Authentication:** Implement multi-factor authentication (MFA) to enhance security. Explore the use of biometric authentication for added security.
   - **Data Encryption:** Continuously update and improve data encryption methods to protect sensitive information against emerging cyber threats.

6. **Automation of Administrative Tasks:**
    - **Automated Scheduling:** Develop features to automate the scheduling of interviews and events, including integration with calendar systems like Google Calendar and Outlook.
    - **Chatbots and Virtual Assistants:** Implement AI-driven chatbots to handle routine inquiries and provide instant support to users.
7. **Broadened Scope for Companies and Jobs:**
    - **Global Reach:** Expand the system to support multinational companies and global job opportunities, including features to handle multiple languages and currencies.
    - **Industry-Specific Modules:** Develop industry-specific modules to cater to specialized recruitment needs in sectors such as healthcare, engineering, and IT.
8. **Feedback and Continuous Improvement:**
    - **User Feedback Mechanisms:** Implement robust mechanisms for collecting and analyzing user feedback to inform ongoing improvements and new feature development.
    - **Regular Updates:** Establish a schedule for regular updates to the system, ensuring it stays current with technological advancements and user requirements.
9. **Partnerships and Collaborations:**
    - **Educational Partnerships:** Collaborate with other educational institutions to broaden the system's user base and share best practices.
    - **Industry Partnerships:** Partner with leading companies to offer exclusive job opportunities and internships to students using the system.
10. **Legal and Compliance Updates:**
    - **Regulatory Compliance**: Ensure the system remains compliant with changing legal requirements and data protection regulations such as GDPR and CCPA.
    - **Accessibility Compliance:** Continuously improve the system to meet accessibility standards, ensuring it is usable by individuals with disabilities.

8. REFERENCES

1. Kashyap, N., & Singh, N. (2019). "Campus Recruitment System Using Machine Learning". International Journal of Scientific Research in Computer Science, Engineering, and Information Technology.
2. Ahsan, K., & Nasir, M. (2018). "Automated Campus Recruitment System". International Journal of Computer Applications.
3. Rao, P. (2017). "An Efficient Campus Recruitment System". International Journal of Advanced Research in Computer Science and Software Engineering.
4. Sharma, R., & Kaur, P. (2020). "Online Campus Recruitment System: A Framework". Journal of Emerging Technologies and Innovative Research.
5. Gupta, A., & Sharma, A. (2016). "Campus Recruitment System: A Comprehensive Approach". International Journal of Computer Applications.
6. Mathur, S., & Choudhary, S. (2019). "Cloud-Based Campus Recruitment System". International Journal of Scientific Research in Computer Science, Engineering, and Information Technology.
7. Kumar, M., & Kaur, M. (2018). "Enhancing Recruitment Process through Campus Recruitment System". International Journal of Computer Applications.
8. Pandey, S., & Jain, A. (2017). "Automated Campus Placement System". International Journal of Engineering Research and Applications.
9. National Institute of Standards and Technology (NIST). (2015). Guide to Software Assurance (SP 800-53 Rev. 4). NIST.
10. Pope, M. (2016). Pro Git (2nd ed.). Apress. Retrieved from https://git-scm.com/book/en/v2
11. Scrum Alliance. (n.d.). Scrum Guide. Retrieved from https://www.scrumguides.org/scrum-guide.html
12. Google. (n.d.). Google Cloud Platform Documentation. Retrieved from https://cloud.google.com/docs
13. Microsoft. (n.d.). Azure DevOps Documentation. Retrieved from https://docs.microsoft.com/en-us/azure/devops/
14. SeleniumHQ. (n.d.). Selenium WebDriver Documentation. Retrieved from https://www.selenium.dev/documentation/en/webdriver/

15. The Apache Software Foundation. (n.d.). Apache JMeter User Manual. Retrieved from https://jmeter.apache.org/usermanual/index.html
16. W3C. (n.d.). Web Content Accessibility Guidelines (WCAG) 2.1. Retrieved from https://www.w3.org/WAI/WCAG21/quickref/
17. University of Cambridge. (n.d.). Cambridge Advanced Learner's Dictionary & Thesaurus. Retrieved from https://dictionary.cambridge.org/
18. Jackson, M. (1995). Software Requirements & Specifications: A Lexicon of Practice, Principles and Prejudices. Addison-Wesley.
19. The PostgreSQL Global Development Group. (n.d.). PostgreSQL Documentation. Retrieved from https://www.postgresql.org/docs/
20. Oracle Corporation. (n.d.). Java SE Documentation. Retrieved from https://docs.oracle.com/en/java/
21. GitHub. (n.d.). GitHub Documentation. Retrieved from https://docs.github.com/en
22. Cohn, M. (2004). User Stories Applied: For Agile Software Development. Addison-Wesley.
23. Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.
24. IEEE. (2017). IEEE Standard for Software and Systems Test Documentation (IEEE Std 829-2017). IEEE.
25. Mehta, S., & Sinha, R. (2018). "E-Recruitment System for Campus Placements". International Journal of Innovative Research in Computer Science & Technology.
26. Malhotra, R., & Gupta, S. (2017). "Campus Placement Automation". International Journal of Advanced Research in Computer and Communication Engineering.
27. Bansal, P., & Singh, T. (2019). "Data-Driven Campus Recruitment System". International Journal of Engineering Research and Applications.