



UE21CS352B - Object Oriented Analysis & Design using Java

Mini Project Report

Placement Management System

Submitted by:

Name 1: Amulya Marali	SRN: PES1UG21CS074
Name 2: Ananya Prasad	SRN: PES1UG21CS079
Name 3: Ashwini Venkataraman Hegde	SRN: PES1UG21CS124
Name 4: Ruchitha V S	SRN: PES1UG21CS909

6 th Semester B Section

Prof. Raghu B A

January - May 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru – 560 085, Karnataka, India

Problem Statement:

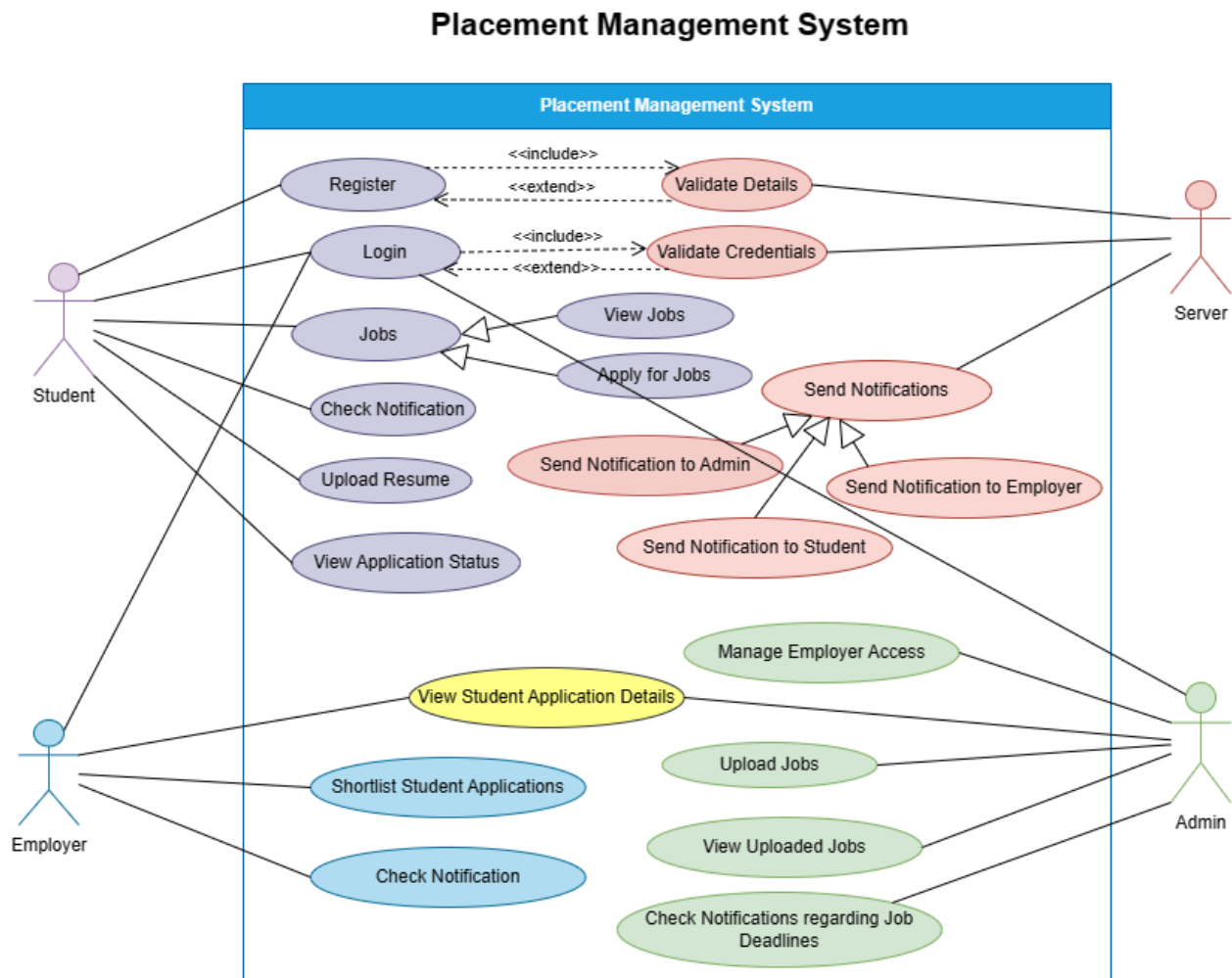
The Placement Management System is a comprehensive platform designed to streamline the placement process for educational institutions, students, and employers. The system facilitates the lifecycle of placement activities, from student registration to job posting, application processing and placement tracking.

Key Features:

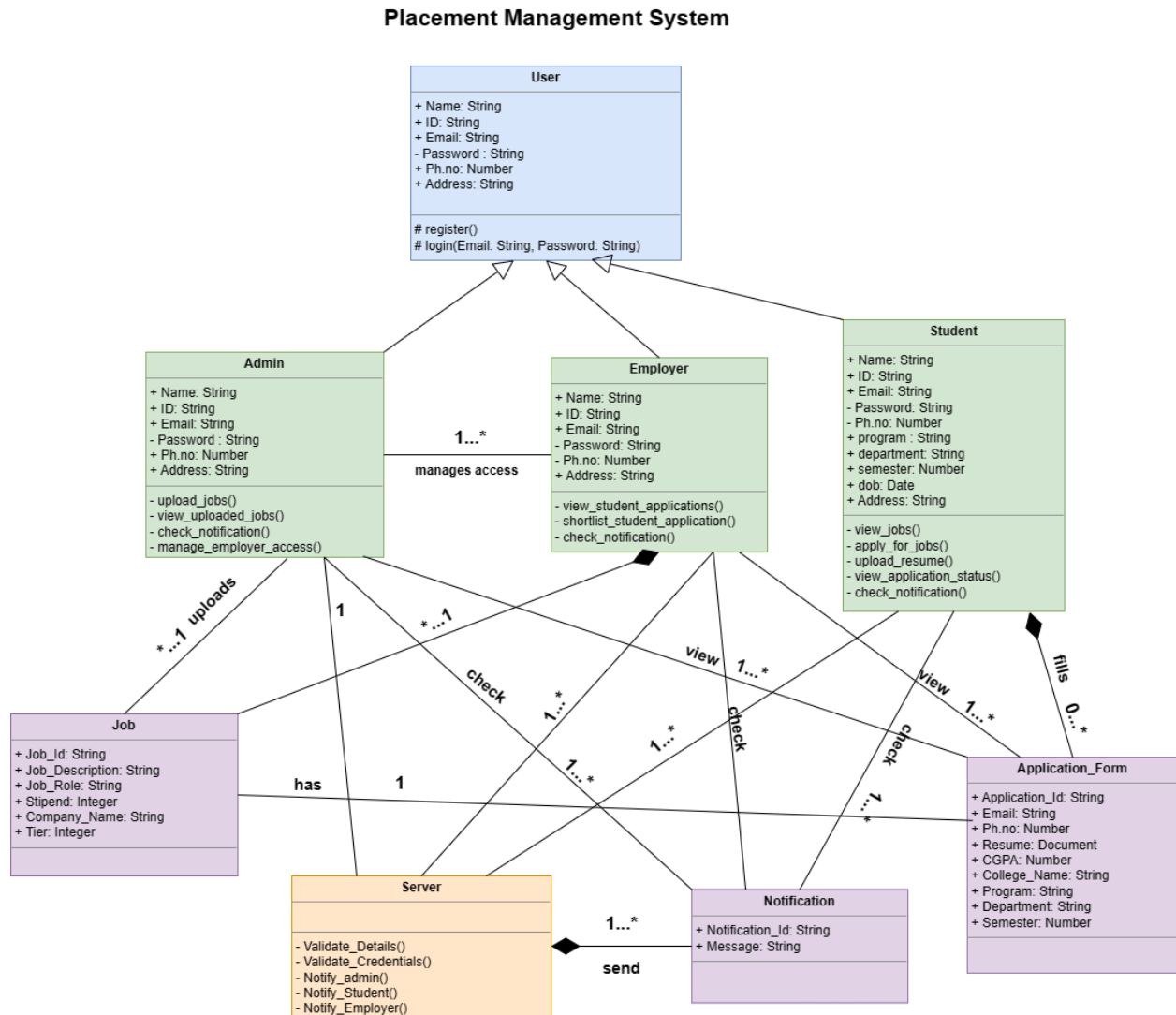
- **User Roles:** The system supports three user roles - Administrator, Student, and Employer, each with specific privileges and responsibilities.
- **Student Registration:** Students can create accounts, provide their educational background, skills, and upload resumes to showcase their qualifications.
- **Job Posting:** Employers can share details to the admin including job descriptions, required skills, and application deadlines and the Admin can upload the same providing students with a wide range of employment prospects.
- **Application Process:** Students can browse job listings, apply to positions, and employers can review student profiles and applications to select suitable candidates.
- **Placement Status Tracking:** Students can track the status of their applications and receive feedback from employers, while employers can monitor the status of job postings and applications.
- **Security and Access Control:** The system employs secure authentication and authorization mechanisms to safeguard data privacy and controls access to features based on user roles.
- **Responsive UI:** The user interface is designed to be responsive, ensuring optimal user experience across different devices and screen sizes.

UML Models:

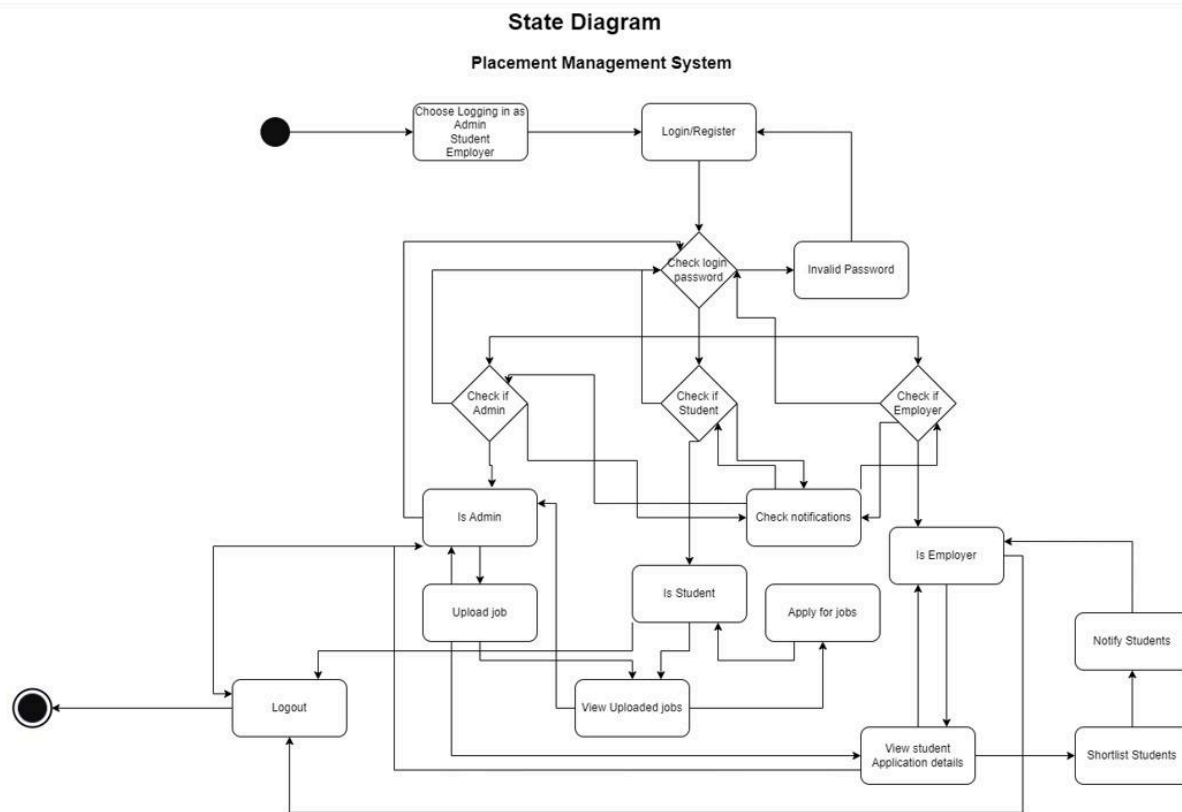
1. Use Case Diagram



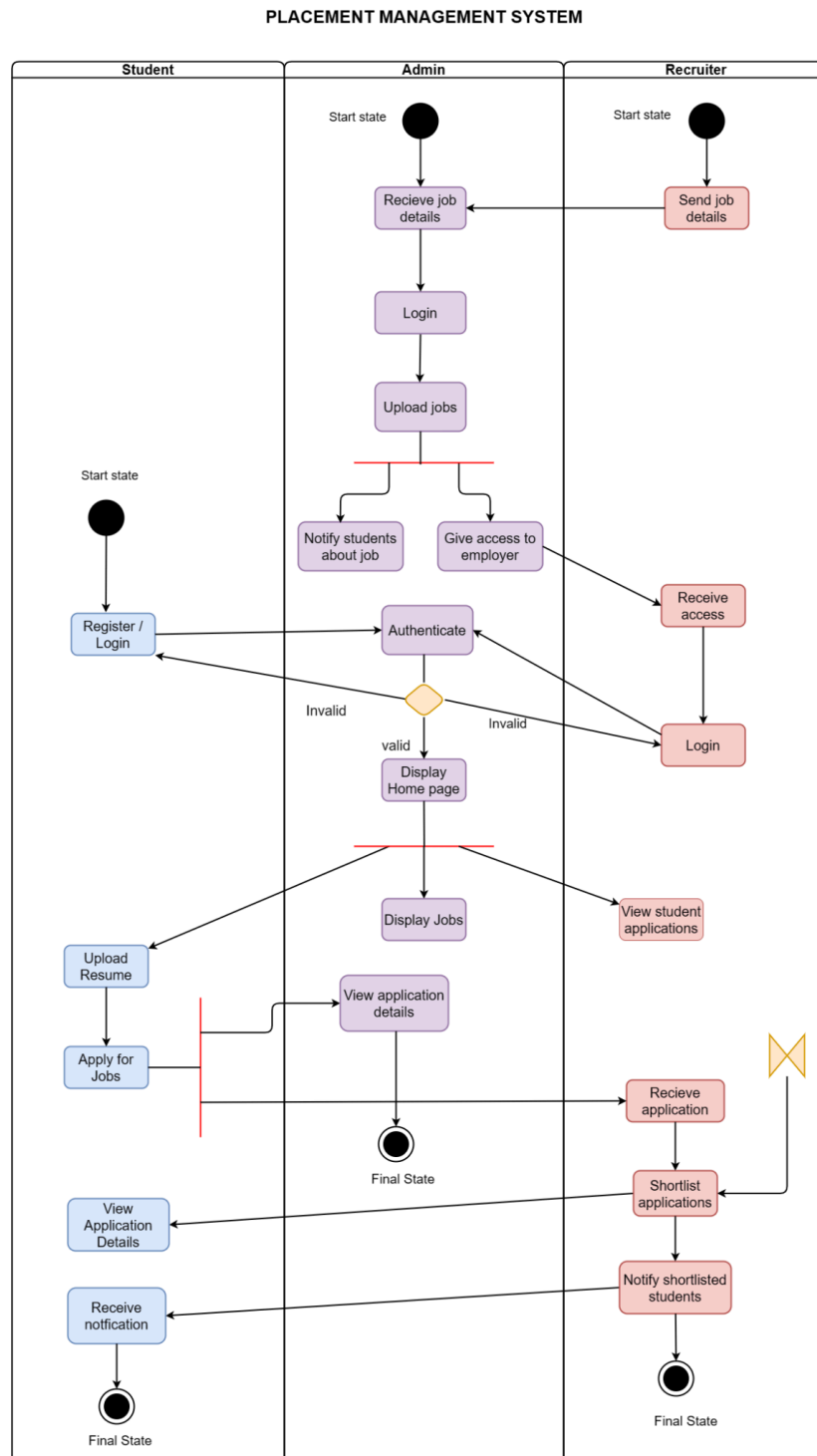
2. Class Diagram:



3. State Diagram:



4. Activity Diagram:



Architecture Patterns:

The Model-View-Controller (MVC) architecture pattern is a widely used approach in software development, particularly in building web applications and user interfaces.

In MVC, the application is divided into three interconnected components:

- **Model:** The Model represents the data and the business logic of the application. It encapsulates the data structure and operations that manipulate the data. The Model responds to requests for information, processes data, and interacts with the database or other sources of persistent data.
We also made use of components like **Entity**, **Repository** and **Service**. Entities represent the data structure of the application, Repositories handle data access and manipulation, and Services contain the application's business logic. Together, these components form the Model layer of the MVC architecture, facilitating a structured and modular approach to application development.
- **View:** The View is responsible for the presentation layer of the application. It displays the data from the Model to the user and handles user interactions such as input events. Views are typically passive components that receive updates from the Controller and render the appropriate content to the user interface.
- **Controller:** The Controller acts as an intermediary between the Model and the View. It receives input from the user via the View, processes it, and updates the Model accordingly. Controllers handle user actions, invoke the appropriate methods in the Model, and update the View to reflect any changes in the data.

Design Principles:

GRASP Principles

- **Creator** - In Placement Management System, admin is responsible for creating a new job and also recruiter in our case. So **Admin** class becomes the creator for **Recruiter** class and **Job** class.
- **Information Expert** - “Objects do things related to the information they have.”
In our system we have Admin, Student, Job and Recruiter as main classes along with respective service classes which handle the functionalities related to the data present in the corresponding classes.

- **Controller** - Responsibility of handling the system events is assigned to the **Controller** classes corresponding to each class in Model, which helps in delegating the request in User Interface layer objects to Model layer objects. Also minimizes the dependencies between the UI components and the system classes.
- **Low Coupling** - Database operations are handled by the intermediate components called **Services** which acts as the mediator between the Data layer and the Model layer. We have Service classes for Student and Job entities which helps in ensuring less dependencies in between the components.

SOLID Principles

- **Single Responsibility Principle (SRP):** Each class / module within the system has single responsibility.
 - **User class:** Handles user data and is responsible for all the actions associated with the user role.
 - **Admin class:** Manages job-related operations such as uploading, updating, deleting job data.
 - **Recruiter class:** Responsible for shortlisting students.
- **Open/Closed Principles (OCP):** Instead of modifying the existing user management code, we can extend the system to accommodate new users. New features can also be implemented by creating a separate module/class that can be plugged into the existing system.
- **Interface Segregation Principle (ISP):** Instead of having a single monolithic 'User' interface with methods for all different roles, we have constructed separate interfaces for each role containing methods relevant to its corresponding role.
- **Dependency Inversion Principle (DIP):** High level modules such as controllers, which are responsible for handling the user requests and coordinating the flow of data do not directly depend on the low-level modules such as data access objects (repositories). In our project, controllers are dependent on abstractions (interfaces / services)

Design Patterns:

- **Facade Pattern:** The Facade pattern has been employed to create a simplified interface for the users, abstracting away the complexities of the underlying subsystems such as authentication, job details management and application processing. This facade interface provides a straightforward way for students to browse job details, apply for positions. shielding them from the intricacies of backend operations.
- **Bridge Pattern:** Decouples an abstraction from its implementation, allowing the two to vary independently. We used the Bridge pattern to separate the UI components from the underlying data storage mechanisms, enabling flexibility in choosing different databases or data sources.
- **Observer Pattern:** The Observer pattern is a behavioral design pattern where an object, known as the subject, maintains a list of its dependents, called observers, and notifies them of any state changes, typically by calling one of their methods. In our system, we used the observer patterns to notify students when they get shortlisted for a job.
- **Singleton Pattern:** The Singleton Pattern is implemented for the Admin class in the Placement Management System to ensure that there is only one instance of the Admin object throughout the application's lifecycle. This design choice simplifies access to administrative functionalities and guarantees a single point of control for administrative tasks.
- **Decorator Pattern:** The Decorator Pattern is a structural design pattern that allows behavior to be added to individual objects dynamically, without affecting the behavior of other objects from the same class. In our project we used it to display which tier the uploaded job belongs to which dynamically changes on runtime.

Github Link: https://github.com/amulyamarali/Placement_Management_System

Individual Contribution:

Amulya: Signup and Login authentication for all users and Decorator Pattern.

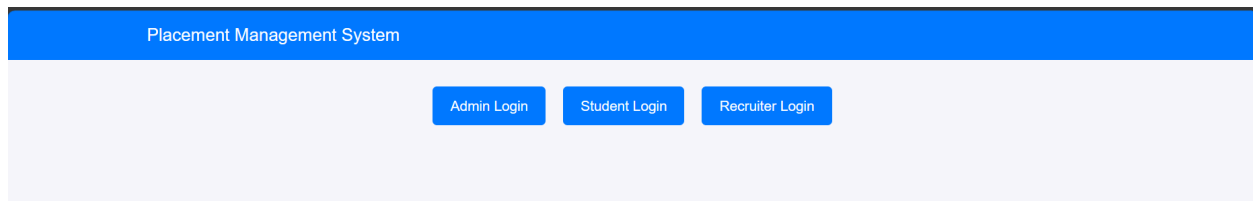
Ananya: Student application form, and student shortlisting and Bridge Pattern.

Ashwini: Display job lists for students, applicants for admin, Observer and Singleton Pattern.

Ruchitha: Display job lists for admin, admin deletion jobs and Facade Pattern.

Input and Output Screenshots:

1. Home Page



2. Admin Dashboard - Displaying jobs before deleting

Uploaded Jobs										Upload new Job
Company	Job Role	Salary	Description	Job ID	Recruiter ID	Recruiter Credentials	Deadline			
Intuit	SDE	30,000.00	Design software systems and contribute to strategic planning. Oversee project technical aspects.	52	IN1	intuit1	2024-04-01	Delete	Applicants	Shortlist
JP Morgan	Data Scientist	40,000.00	Analyze, visualize market trends.	53	JP1	jpm1	2024-04-23	Delete	Applicants	Shortlist

3. Admin Dashboard - Displaying jobs after deleting

Uploaded Jobs										Upload new Job
Company	Job Role	Salary	Description	Job ID	Recruiter ID	Recruiter Credentials	Deadline			
Intuit	SDE	30,000.00	Design software systems and contribute to strategic planning. Oversee project technical aspects.	52	IN1	intuit1	2024-04-01	Delete	Applicants	Shortlist

4. Admin - Upload Jobs

Company	<input type="text" value="JP Morgan"/>
Job Role	<input type="text" value="Data Scientist"/>
Salary	<input type="text" value="40000"/>
Description	<input type="text" value="Analyze, visualize market trends."/>
Recruiter ID	<input type="text" value="JP1"/>
Recruiter Credentials	<input type="text" value="jpm1"/>
Deadline	<input type="text" value="23-04-2024"/>
<input type="button" value="Upload"/>	

5. Student - Display jobs uploaded by admin

Job List					
			Home Profile		
Company	Job Role	Salary	Description	Deadline	
Intuit	SDE	30,000.00	Design software systems and contribute to strategic planning. Oversee project technical aspects.	2024-04-01	Apply

6. Student application form

Application Form

Name

SRN

Gender

☐ Male ☒ Female

Branch

Contact

Email

CGPA

7. Student - after applying for job

[Job List](#)[Home](#)[Profile](#)

No jobs to apply


8. Student Signup

Sign Up

amulya@gmail.com

amulya

....


.... 

Signup

9. Recruiter Login

Recruiter Log In

IN1

..... 

Log In

10. Recruiter Dashboard - Before filter

Recruiter Dashboard

Filter by CGPA (\geq)							Filter	
ID	Name	SRN	Gender	Branch	Email	CGPA	Sem	Resume Link
1	Ashwini Venkataraman Hegde	PES1UG21CS124	female	CSE	hegdeashwini627@gmail.com	9.19	6	https://resume1.com
2	Amulya Marali	PES1UG21CS074	female	CSE	amulya@gmail.com	8.99	6	https://resume2.com

11. Recruiter Dashboard - After filter

Recruiter Dashboard

Filter by CGPA (\geq)							Filter	
ID	Name	SRN	Gender	Branch	Email	CGPA	Sem	Resume Link
1	Ashwini Venkataraman Hegde	PES1UG21CS124	female	CSE	hegdeashwini627@gmail.com	9.19	6	https://resume1.com

12. Admin - Applicants

Applied Students:

ID	Name	SRN	Phone Number	Gender	Branch	Email	CGPA	Sem	Resume Link
1	Ashwini Venkataraman Hegde	PES1UG21CS124		female	CSE	hegdeashwini627@gmail.com	9.19	6	https://resume1.com
2	Amulya Marali	PES1UG21CS074		female	CSE	amulya@gmail.com	8.99	6	https://resume2.com

13. Admin - Shortlisted Students

Shortlisted Students:

ID	Name	SRN	Phone Number	Gender	Branch	Email	CGPA	Sem	Resume Link
1	Ashwini Venkataraman Hegde	PES1UG21CS124		female	CSE	hegdeashwini627@gmail.com	9.19	6	https://resume1.com