**SOFTWARE REQUIREMENT SPECIFICATIONS (SRS) DOCUMENT**

**NAME**: LAVANYA B

**ROLL NO**: 7376221MC122

**YEAR**: II

**DEPARTMENT**: MTRS

**PROJECT ID**: 04

**PROJECT TITLE:**

“Students Ranking Dashboard”

**TECH STACK**: MERN

**PROBLEM STATEMENT:**

Develop a student ranking dashboard using the MERN stack (MongoDB, Express.js, React, Node.js) to facilitate performance tracking and ranking of college students. The system will have three types of logins: Student, Faculty, and Admin, each with specific functionalities and access controls.

**STUDENT LOGIN:**

**1. Authentication:** Students log in with unique credentials.

**2. Dashboard Features:**

* **Performance Metrics:** Display FA percentage, attendance, completion status of C programming and data structures courses.
* **Skill Assessment:** Track aptitude solving levels, number of attempts to clear each level.
* **Placement Tracking:** Record the number of companies attended and the furthest round reached in each company's recruitment process.
* **Ranking:** View their rank based on the above metrics compared to peers.

**FACULTY LOGIN:**

**1.Authentication:** Faculty members log in with unique credentials.

**2.Dashboard Features:**

* **Student Management:** Access details of 10 to 25 students under their guidance.
* **Student Performance:** View each student's progress in C programming and data structures, number of companies attended, recruitment rounds cleared, department, and roll number.
* **Monitoring:** Oversee attendance, FA percentage, and aptitude levels of students.

**ADMIN LOGIN:**

**1.Authentication:** Admins log in with unique credentials.

**2.Dashboard Features:**

* **Data Import:** Import large datasets of student information via Excel files.
* **Faculty Assignment:** Assign students to faculty members for guidance.
* **Performance Filtering:** Filter and shortlist students based on performance in C programming and data structures.
* **Access Control:** Manage and assign editing rights to specific users for updating student dashboards.

**TECHNICAL REQUIREMENTS:**

* **Frontend:** Use React.js for the user interface, ensuring a responsive and user-friendly experience.
* **Backend:** Utilize Node.js with Express.js to handle server-side logic and APIs.
* **Database:** Implement MongoDB to store student, faculty, and admin data securely.
* **Authentication:** Implement secure login mechanisms for different user roles.
* **Data Handling:** Ensure robust import/export functionality for Excel data.
* **User Interface:** Design intuitive dashboards for students, faculty, and admin with relevant data visualization.

By addressing these requirements, the student ranking dashboard will provide a robust platform for tracking and improving student performance, facilitating faculty guidance, and enabling admin oversight and management.

**SOLUTION:**

Here's a step-by-step approach to implementing the student ranking dashboard based on the problem statement:

**1. PROJECT SETUP**

* **Initialize MERN stack project:** Set up a new project with React.js for the frontend and Node.js with Express.js for the backend. Use MongoDB for the database.

**INSTALL DEPENDENCIES**

* **Backend:** Express.js, Mongoose, JWT, Multer (for file uploads), and any other necessary middleware.
* **Frontend:** React.js, Axios (for API requests), Chart.js or D3.js (for data visualization).

**2. DATABASE DESIGN**

**Define Mongoose Schemas:**

* **User Schema:** Differentiate between students, faculty, and admin using roles.
* **Performance Schema:** Track FA percentage, attendance, programming levels, aptitude progress, companies attended, rounds cleared, etc.
* **Faculty-Student Relationship:** Link students to faculty members.

**3. Authentication System**

* Implement JWT-based authentication:
* Create login and registration endpoints.
* Use JWT to create tokens upon successful login.
* Middleware to protect routes based on user roles.

**4.Data Import and Management**

* Excel Data Handling:
* Use Multer for file uploads.
* Parse Excel files using a library like “xlsx”
* Validate and update MongoDB with parsed data.

**5. Performance Tracking and Visualization**

* Use Chart.js or D3.js in React to create interactive charts showing student progress.
* Implement different chart types (e.g., bar charts, line charts) to display various performance metrics.

**Software Requirement Specifications (SRS):**

1. **Introduction:**

The purpose of this Software Requirement Specification (SRS) is to outline the development of a web-based student ranking dashboard using the MERN stack. The system is designed for college students, faculty, and admins to track, manage, and display student performance metrics and rankings effectively.

1. **Scope:**

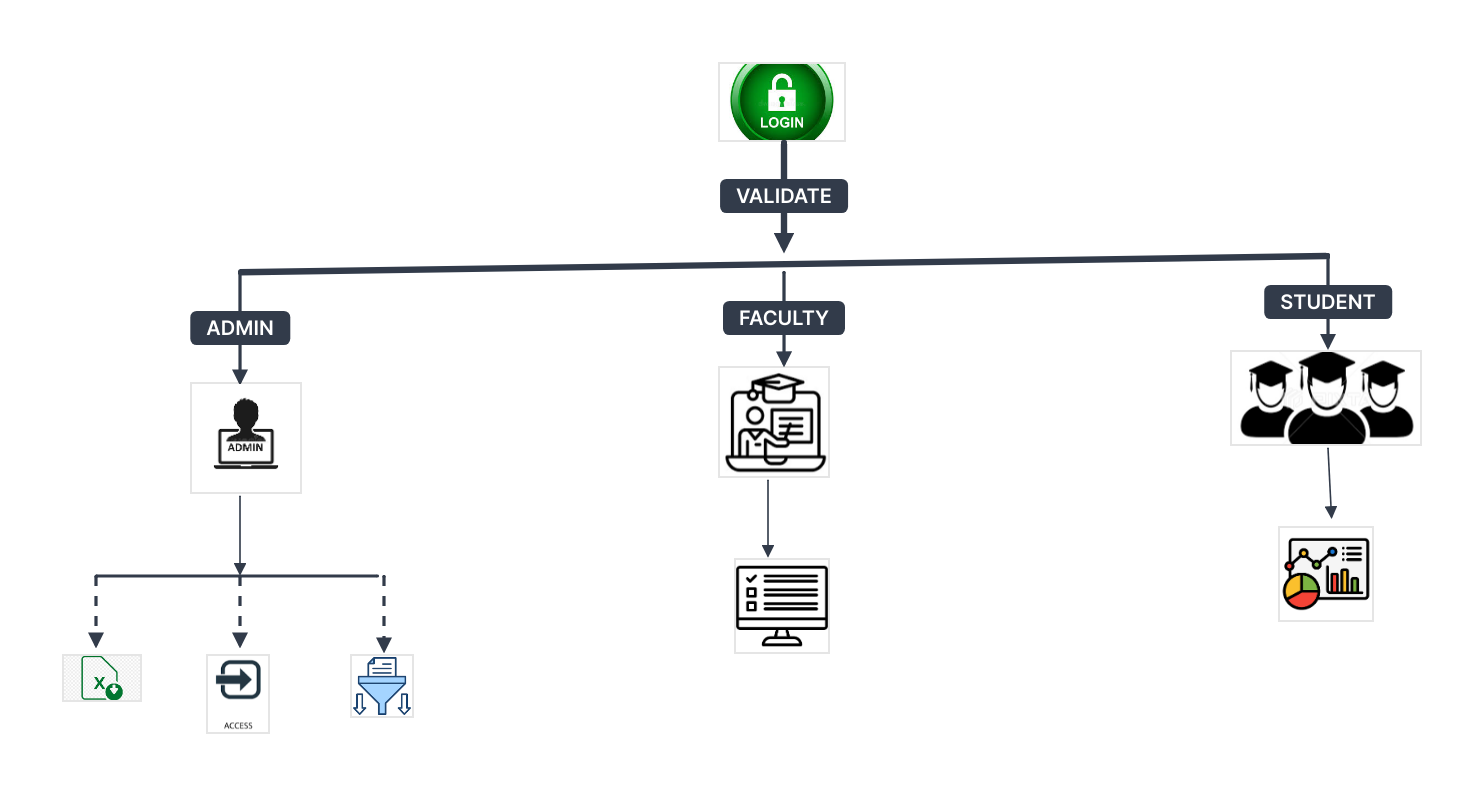
The Student Ranking Dashboard system will support three user roles: Student, Faculty, and Admin. Students can track their academic performance, course completions, and recruitment progress. Faculty members can monitor and manage the progress of their assigned students. Admins will have capabilities to import data, assign faculty mentors, and manage user access and performance filtering. The system aims to enhance performance tracking, faculty guidance, and administrative efficiency within the college environment**.**

1. **INDENTED AUDIENCE AND USE:**

This portal is designed for three primary users:

* **Faculty Members:** To gain insights into the features available for student performance monitoring.
* **Administrators:** To understand the system capabilities for data management, faculty assignment, and performance analysis.
* **Students:** To comprehend the functionalities available for tracking their academic progress and rankings.

1. **SYSTEM FLOW:**

****

The Student Ranking Dashboard system is designed to manage and display student performance data effectively. The system flow can be described as follows:

**1. User Authentication:**

* **Login:** Users (students, faculty, and admins) log in with their unique credentials. Secure authentication mechanisms ensure data privacy and integrity.
* **Role-Based Access:** Upon successful login, users are directed to their respective dashboards based on their roles (Student, Faculty, Admin).

**2. Student Dashboard:**

* **Performance Metrics**: Students view their FA percentage, attendance records, and completion status of C programming and data structures courses.
* **Aptitude Tracking:** Aptitude solving levels and the number of attempts to clear each level are displayed.
* **Recruitment Activity:** Information about companies attended and the furthest recruitment round cleared is provided.
* **Ranking:** Students can see their rank relative to peers based on the combined performance metrics.

**3. Faculty Dashboard:**

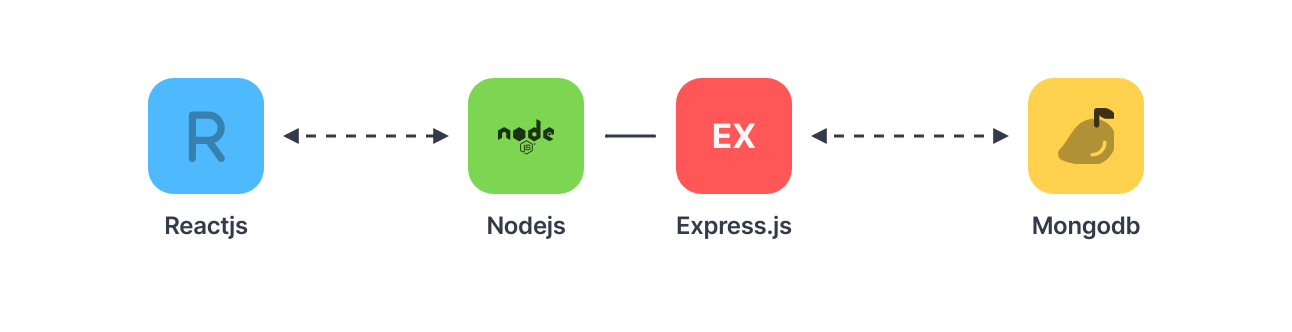
* **Student Management:** Faculty members view a list of students under their guidance.
* **Detailed Insights:** Faculty can access each student's progress in core subjects, attendance, and recruitment activities.
* **Monitoring and Feedback:** Faculty monitor student performance and provide necessary feedback for improvement.

**4. Admin Dashboard:**

* **Data Management:** Admins import and manage large datasets of student information via Excel files.
* **Faculty Assignment:** Admins assign students to faculty members for mentoring.
* **Performance Filtering:** Admins filter and shortlist students based on performance in C programming and data structures.
* **Access Control:** Admins manage user permissions, granting or restricting access to edit student data as needed.

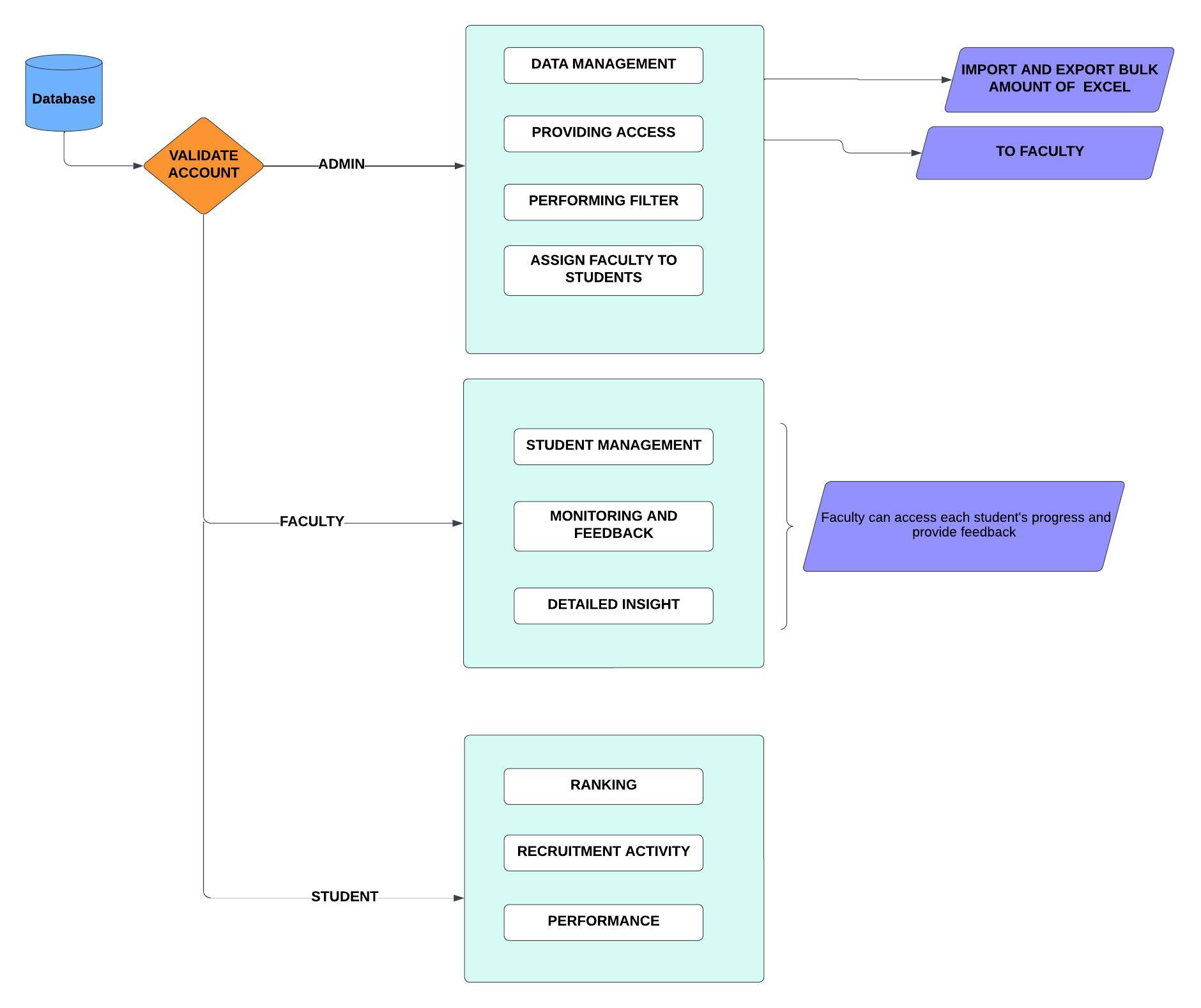
**5.TECHNICAL COMPONNENTS:**

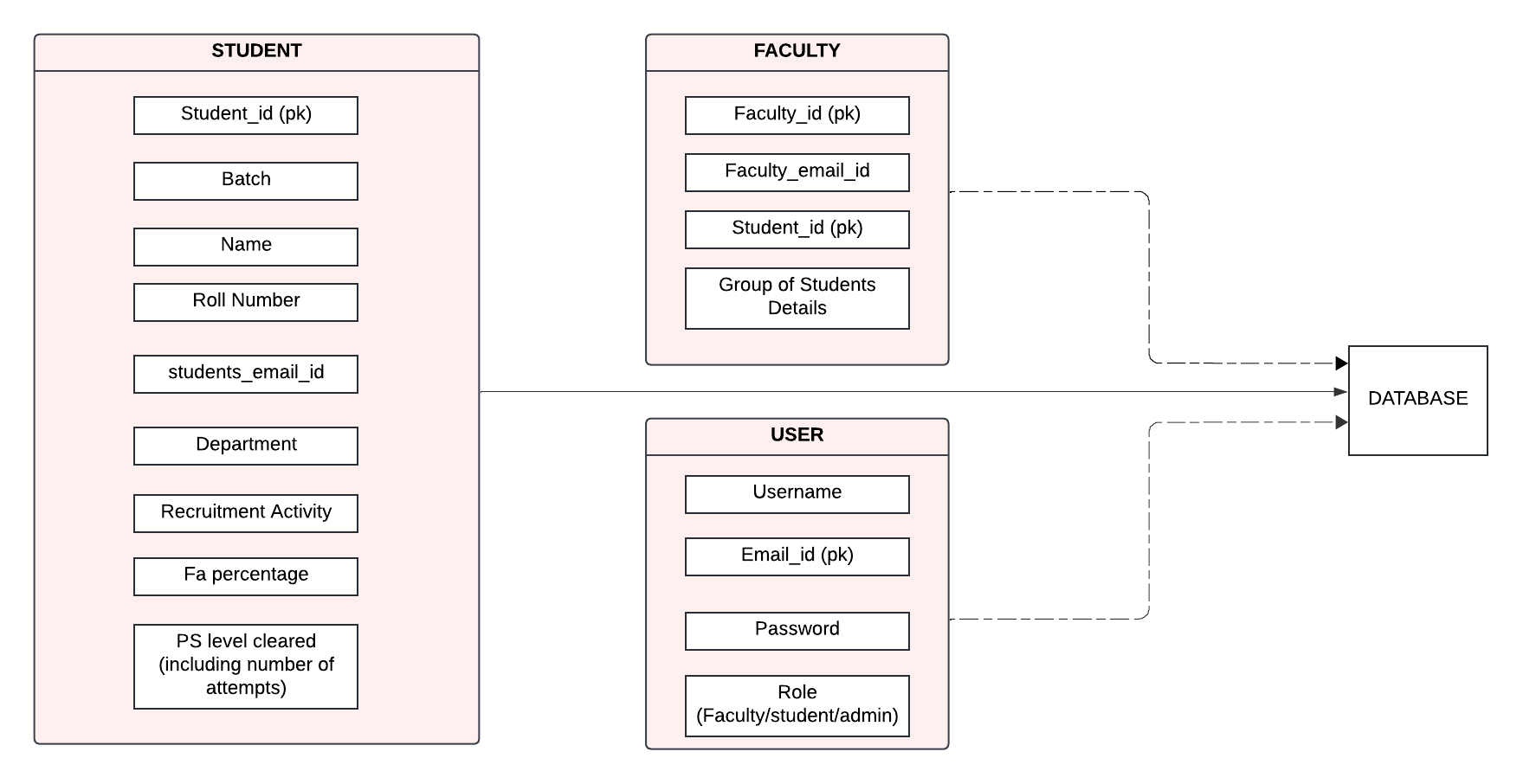
**MERN – MongoDB ExpressJS ReactJS NodeJS**



|  |  |
| --- | --- |
| **Components** | **Tech Stack** |
| Frontend | React.js |
| Backend | Node.js, Express.js |
| Database | MongoDB |

1. **SYSTEM FLOWCHART:**

****

1. **DATABASE:  
   **

**8.FUNCTIONAL REQUIREMENTS:**

The Student Ranking Dashboard system comprises several key functionalities tailored for students, faculty, and administrators to streamline performance tracking and management.

**Student Login:**

* **Authentication:** Secure login for students using unique credentials.
* **Dashboard View:** Display of FA percentage, attendance, and completion status of courses such as C programming and data structures.
* **Aptitude Tracking:** Monitor aptitude solving levels and the number of attempts to clear each level.
* **Recruitment Tracking:** Record of companies attended and the furthest recruitment round reached.
* **Ranking Display:** Show students' rank compared to peers based on combined performance metrics.

**Faculty Login:**

* **Authentication:** Secure login for faculty members with unique credentials.
* **Student Management Dashboard:** Access to the list of assigned students (10 to 25 per faculty member).
* **Performance Monitoring:** Detailed insights into each student's progress in courses, attendance records, and recruitment activities.
* **Student Details:** Access to department and roll number information of students.

**Admin Login:**

* **Authentication:** Secure login for administrators with unique credentials.
* **Data Import/Export:** Ability to import and manage large datasets of student information via Excel files.
* **Faculty Assignment:** Assign students to faculty members for personalized guidance and mentoring.
* **Performance Filtering:** Filter and shortlist students based on their performance in C programming and data structures.
* **Access Management:** Control user permissions, granting or restricting access to edit student data.

**General System Requirements:**

* **Responsive Design:** Ensure the user interface is intuitive and accessible across various devices.
* **Data Security:** Implement robust security measures to protect sensitive student and faculty data.
* **Scalability:** Design the system to handle a growing number of users and data without compromising performance.

**9.Non-Functional Requirements (NFRs)**

The Student Ranking Dashboard must meet several non-functional requirements to ensure reliability, performance, security, and usability.

**Performance:**

* **Response Time:** The system should have a response time of under 2 seconds for most operations, ensuring a smooth user experience.
* **Scalability:** The system must handle a growing number of users and data entries efficiently, supporting up to 10,000 concurrent users without performance degradation.
* **Availability:** The system should maintain 99.9% uptime, ensuring high availability for users.

**Security:**

* **Data Protection:** Implement strong encryption for sensitive data, both in transit and at rest, to prevent unauthorized access.
* **Authentication:** Utilize robust authentication mechanisms, including multi-factor authentication, to secure user accounts.
* **Authorization:** Ensure role-based access control (RBAC) to restrict access to specific functionalities based on user roles (students, faculty, admin).

**Usability:**

* **User Interface:** Design a user-friendly, intuitive interface that is easy to navigate, reducing the learning curve for users.
* **Accessibility:** Ensure the system is accessible to users with disabilities, adhering to WCAG 2.1 guidelines.
* **Responsiveness:** The application should be fully responsive, providing an optimal user experience on various devices (desktops, tablets, smartphones).

**Reliability:**

* **Backup and Recovery:** Implement regular data backup procedures and ensure quick recovery mechanisms in case of data loss or system failure.
* **Fault Tolerance:** Design the system to handle hardware or software failures gracefully, minimizing disruption to users.

**Maintainability:**

* **Code Quality:** Ensure the codebase is well-documented, modular, and adheres to best practices to facilitate easy maintenance and updates.
* **Extensibility:** Design the system architecture to be flexible, allowing easy integration of new features and functionalities in the future.

**Compliance:**

* **Data Privacy:** Ensure compliance with relevant data protection regulations, such as GDPR, to safeguard user privacy and data rights.