**Student Details**

**FN-5**

**Ramandeep Saroj**

**Kranti Sambhav**

**Hande Rajat Santosh**

**Software Requirements Specification (SRS)**

**1. Introduction**

**1.1 Purpose**

* The purpose of this project is to develop a **Student Details Management System** that maintains a database of students, recording their extracurricular activities.
* The system will facilitate easy access to student records, assist in accreditation processes, and provide features such as event tracking, announcements, and placement records.

**1.2 Scope**

* The system will allow students to record their participation in events (which include sports, arts, drama, music, dance), internships, professional societies as well as the programs held outside of the college for the students.
* It will provide administrative features such as importing data, managing announcements, and maintaining extracurricular activities. Security measures using **Cloudflare** will be implemented to prevent cyber threats.

**1.3 Definitions, Acronyms, and Abbreviations**

* **SRS** - Software Requirements Specification
* **DDoS** - Distributed Denial of Service
* **API** - Application Programming Interface
* **UI/UX** - User Interface/User Experience

**1.4 References**

No references required.

**1.5 Overview**

This document describes the functional, non-functional, and external requirements of the system.

**2. General Description**

**2.1 Product Perspective**

* This system will function as a **centralized platform** for student record management, replacing manual or fragmented databases.
* It will **integrate with existing university databases using the migrations of the database that can be run on our system or a script which can be used directly to fetch all the data from the database directly**.

**2.2 Product Functions**

* **Student Activity Management**: Store records of technical events like tathva, cultural events like ragam, club activities, competitions that happen in these events and we maintain the achievements of the students which later helps in the data collection in the later stages.
* **Placement Tracking**: Maintain details of placements of an individual student as well as the total students placed or who got internships, and allow offer letter uploads by the students which will be cross verified by the faculty advisor whenever the letter reaches him/her when the upload happens and notifications reaches.
* **Publication Integration**: Suggest paper titles from faculty publication databases.
* **Announcements**: Notify students of important updates. A separate announcements page will be maintained in the website by the admins which shows all the users the most important announcements regarding the upcoming events.
* **Security Measures**: Implement **Cloudflare Workers** for DDoS protection, rate limiting on the requests sent by the user, two factor authentication (may or may not be included) , encrypted sensitive data (like passwords) in the database.

**2.3 User Characteristics**

* **Students**: Can add/edit their activities, upload offer letters, and receive announcements.
* **Faculty**: Can manage student records.
* **Admin**: Post announcements, manage website.

**2.4 Constraints**

* Good Internet connectivity is required for real-time updates and fast internet is needed for operations like offer letter uploads.
* File upload size restrictions for documents (e.g., max 5MB per file).
* Compliance with **university data policies** will be checked for any data added for the user or any file uploaded by the respective faculties.

**2.5 Assumptions and Dependencies**

* The system will be hosted as follows => AWS for the code logic and Object stores like S3 for the file uploads.
* Integration with **Cloudflare Workers for security.**

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.1.1 Student Module**

* Students can log in using Google OAuth authentication and are redirected to their personal dashboard.
* The student dashboard provides access to important details, announcements, and navigation options.
* Students can access the following features from the dashboard:
* **Update Details**: Allows students to update their participation records in various events and activities.
* **View Details:** Enables students to review their submitted approval requests and update/resubmit forms if needed.
* **Profile Management:** Students can view and update their personal profiles.
* In the Update Details section, students can edit and submit information for:
* **Technical Events**
  + Students must complete all required details, with essential fields marked '\*'.
  + Students should also upload the necessary documents for verification.
  + After filling out the form, they should click the submit button to send their request for faculty approval**.**
* **Cultural Events**
  + All Cultural event details should be filled in by students, ensuring that fields marked '\*' are completed.
  + Student must also upload the required documents for verification.
  + Once done, they need to click the submit button to forward their request for faculty approval.
* **Sports Events**
  + Students are required to enter all details, especially the mandatory fields marked with '\*', and upload the necessary documents for verification.
  + After that, students should submit the form to send their request for faculty approval.
* **Publication Details**
  + Students can either search for an existent entry for publication in database by entering the name, or can fill up the details on their own.
  + All necessary details should be filled. Click on submit to send the request of approval to faculty.
* **Professional Society Details**
  + Students must provide all necessary information, making sure to complete fields marked with '\*' and upload the required documents for verification.
  + Then, students should click the submit button to initiate the request for faculty approval.
* **Club Details**
  + Students must provide all necessary information, making sure to complete fields marked with '\*' and upload the required documents for verification.
  + Then, students should click the submit button to initiate the request for faculty approval.
* When submitting event details, compulsory fields are marked with a ‘\*’ sign. After filling in the required details (and proofs), students can submit an approval request to the faculty.
* Students can track their submitted requests in the View Details section, where they can also update and resubmit forms if required.
* Students can modify their personal details within the Profile section.

**3.1.2 Faculty Module**

* Faculty members can log in using Google OAuth authentication and are redirected to their personal dashboard.
* The faculty dashboard includes:
* **Approvals Section**: Faculty can review and take action on student requests.
* **Announcements Section**: Displays announcements published by the admin.
* Within the Approvals Section, faculty members can:
* View pending student approval requests.
* Click on a request to see the complete submission details on a separate page.
* Approve or reject requests after reviewing the provided information.
* Provide a reason in case of rejection.
* View previously Rejected and Approved requests.

**3.1.3 Admin Module**

* Admin users can log in using Google OAuth authentication and are redirected to the admin dashboard.
* The admin dashboard provides access to:
* **Add Announcements:** Admins can create and publish new announcements.
* In the Add Announcement section, admins can draft an announcement and publish it upon submission, making it visible to all users.

**3.1.4 Security Features**

* **Multi-Factor Authentication (MFA):** Ensures an additional layer of security for student and admin accounts.
* **Role-Based Access Control (RBAC):** Restricts unauthorized access:
* Admins manage the website and publish announcements.
* Faculty members review and approve student requests.
* Students can submit and manage their participation records.

**3.1.5 Event Calendar**

* An integrated calendar allows students and admins to add events and deadlines.
* Notifications and reminders for upcoming events can be scheduled.
* Students can view their registered events and important academic deadlines.
* Admins can add institution-wide events, and students receive automated reminders.

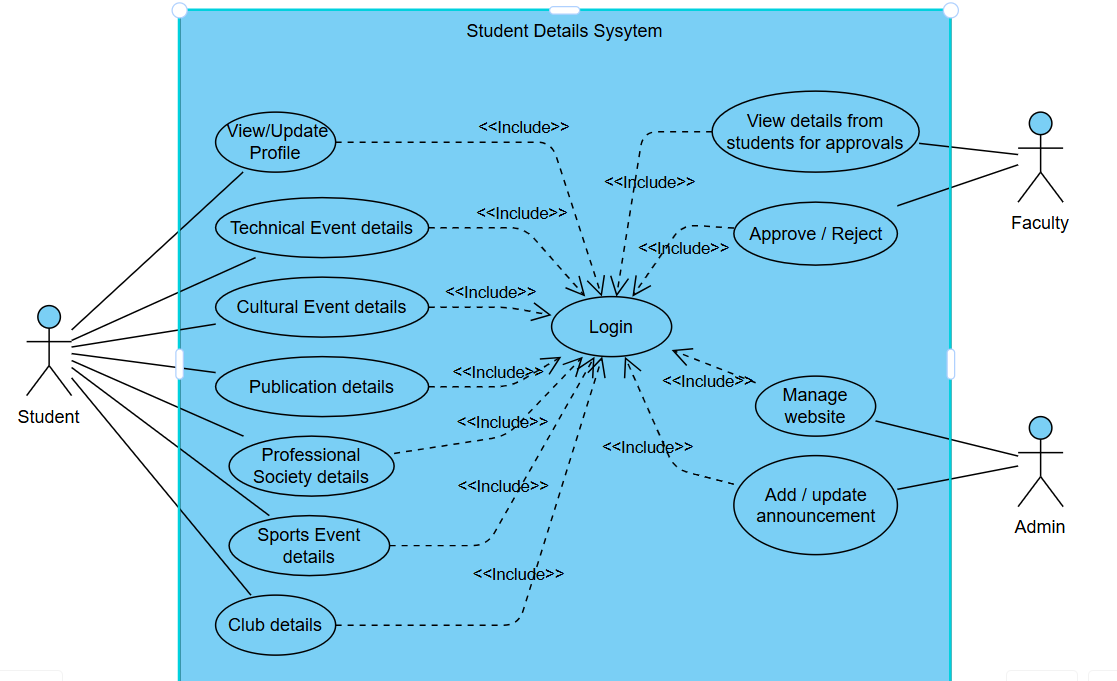
**3.2 Non-Functional Requirements**

* **Performance**: System should load pages within **2 seconds**.
* **Scalability**: Support up to **10,000 student records**.
* **Availability**: 99.9% uptime guarantee.
* **Usability**: Responsive UI for desktop and mobile compatibility.
* **Security:** Implement **Cloudflare** for **DDoS protection**, rate limiting and **firewall rules**.

**3.3 External Interface Requirements**

* **User Interface**: Developed using **React.js** for frontend and **Java/Springboot** for backend. AWS for deployment. Object stores like S3 to stores files uploaded by the student.
* **Database**: Uses **MySQL** for storing student records. May use Prisma and other ORMs to simplify the complexities of the SQL queries.
* **Security Integration**: Uses **Cloudflare API** for security and analytics.

**4. Use-case diagram**

****

**5. Database Design**

**5.1 Table Design**

**5.1.1. STUDENTS**

**Stores details of students including personal, academic, and contact information.**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| StudentID | int | PRIMARY KEY | Unique identifier for each student. |
| Name | string | NOT NULL | Full name of the student. |
| DOB | date | NOT NULL | Date of Birth. |
| Email | string | UNIQUE, NOT NULL | Email ID of the student. |
| RollNo | string | UNIQUE, NOT NULL | Student roll number. |
| ContactNumber | string | NOT NULL | Phone number. |
| Program | string | NOT NULL | Academic program student is enrolled in. |
| YearOfStudy | int | NOT NULL | Current year of study. |
| APARID | string | UNIQUE | Accreditation ID for students. |
| FacultyAdvisor | string | NULLABLE | Assigned faculty advisor. |
| FacultyID | int | FOREIGN KEY | Links to Faculty table. |

**5.1.2. FACULTY**

**Stores information about faculty members.**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| FacultyID | int | PRIMARY KEY | Unique identifier for faculty. |
| Name | string | NOT NULL | Faculty name. |
| Email | string | NOT NULL | Faculty email |

**5.1.3. CLUBS**

**Stores information about the club’s students are part of.**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| ClubID | int | PRIMARY KEY | Unique identifier for each club. |
| StudentID | int | FOREIGN KEY | Links to Students table. |
| ClubName | string | NOT NULL | Name of the club. |
| ClubCategory | string | NOT NULL | Category of the club. |
| Position | string | NOT NULL | Student’s position in the club. |
| DateJoined | date | NOT NULL | Date when the student joined the club. |
| OtherDetails | string | NULLABLE | Additional details. |
| FileLink | string |  | Link to related documents. |
| Status | string |  | Approved/Rejected/Pending |

**5.1.4. PROFESSIONAL SOCIETIES**

**Tracks student memberships in professional societies.**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| SocietyID | int | PRIMARY KEY | Unique identifier for each society. |
| StudentID | int | FOREIGN KEY | Links to Students table. |
| SocietyName | string | NOT NULL | Name of the society. |
| EventCategory | string | NOT NULL | Category of events related to the society. |
| Role | string |  | Student’s role in the society. |
| DateJoined | date | NOT NULL | Date when the student joined. |
| AchievementDetails | string |  | Records of any achievements. |
| FileLink | string |  | Link to documents. |
| Status | string |  | Approved/Rejected/Pending |

**5.1.5. TECHNICAL EVENTS**

**Tracks student participation in technical events.**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| EventID | int | PRIMARY KEY | Unique identifier for event. |
| StudentID | int | FOREIGN KEY | Links to Students table. |
| EventName | string | NOT NULL | Name of the event. |
| EventCategory | string | NOT NULL | Type of technical event. |
| Date | date | NOT NULL | Event date. |
| Role | string |  | Student’s role in the event. |
| Achievement | string |  | awards or recognition received. |
| AchievementDetails | string |  | Description of achievements. |
| OtherDetails | string |  | Additional information. |
| FileLink | string |  | Link to certificates/documents. |
| Status | string |  | Approved/Rejected/Pending |

**5.1.6. CULTURAL EVENTS**

**Tracks student participation in cultural activities.**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| EventID | int | PRIMARY KEY | Unique identifier for event. |
| StudentID | int | FOREIGN KEY | Links to Students table. |
| EventName | string | NOT NULL | Name of the event. |
| EventCategory | string | NOT NULL | Type of cultural event. |
| Date | date | NOT NULL | Event date. |
| Role | string |  | Student’s role in the event. |
| Achievement | string |  | Any awards or recognition received. |
| AchievementDetails | string |  | Description of achievements. |
| OtherDetails | string |  | Additional information. |
| FileLink | string |  | Link to certificates/documents. |
| Status | string |  | Approved/Rejected/Pending |

**5.1.7. SPORTS**

**Tracks student participation in sports activities.**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| ID | int | PRIMARY KEY | Unique identifier for sports record. |
| StudentID | int | FOREIGN KEY | Links to Students table. |
| SportCategory | string | NOT NULL | Type of sport. |
| Date | date | NOT NULL | Event date. |
| Role | string |  | Student’s role in the sport. |
| Achievement | string |  | Any awards or recognition received. |
| AchievementDetails | string |  | Description of achievements. |
| OtherDetails | string |  | Additional information. |
| FileLink | string |  | Link to certificates/documents. |
| Status | string |  | Approved/Rejected/Pending |

**5.1.8. PLACEMENTS AND INTERNSHIPS**

**Tracks student job offers and internships.**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| PlacementID | int | PRIMARY KEY | Unique identifier for placement record. |
| StudentID | int | FOREIGN KEY | Links to Students table. |
| PlacementType | string | NOT NULL | Type (Internship/Full-time). |
| CompanyName | string | NOT NULL | Company offering the placement. |
| StartDate | date | NOT NULL | Start date of the placement. |
| EndDate | date |  | End date (if applicable). |
| Role | string |  | Designation offered. |
| OfferLetterLink | string |  | Link to offer letter. |
| Status | string |  | Approved/Rejected/Pending |

**5.1.9. PUBLICATIONS**

**Tracks student publications.**

| **Column Name** | **Data Type** | **Constraints** | **Description** |
| --- | --- | --- | --- |
| PublicationID | int | PRIMARY KEY | Unique identifier for publication. |
| StudentID | int | FOREIGN KEY | Links to Students table. |
| PublicationTitle | string | NOT NULL | Title of the paper. |
| Authors | string | NOT NULL | Authors of the paper. |
| JournalConference | string | NOT NULL | Journal or conference details. |
| Year | int | NOT NULL | Year of publication. |
| FileLink | string | NULLABLE | Link to document. |
| Status | string |  | Approved/Rejected/Pending |

**5.2 Overall Benefits of the Database Design**

**5.2.1. Structured & Organized – Efficient Data Storage Without Redundancy**

The database is designed using a relational model, ensuring that data is stored in a well-structured manner. Each entity (students, faculty, events, publications, etc.) has its own table, reducing data duplication and redundancy. For example, instead of storing faculty details multiple times for different students, a FacultyID foreign key links students to their assigned faculty. This normalization process prevents unnecessary repetition of data and ensures efficient storage and management.

**5.2.2. Easier Data Retrieval – Quick Access to Student Profiles, Achievements, and History**

Since all tables are properly linked through primary and foreign keys, retrieving information becomes faster and more efficient. A single SQL query can fetch a student’s complete profile, including:

* Personal details (from the students table)
* Events participated in (from Technical Events, Cultural Events, Sports tables)
* Club memberships (from the Clubs table)
* Internship and placement history (from the Placements & Internships table)
* Research publications (from the Publications table)

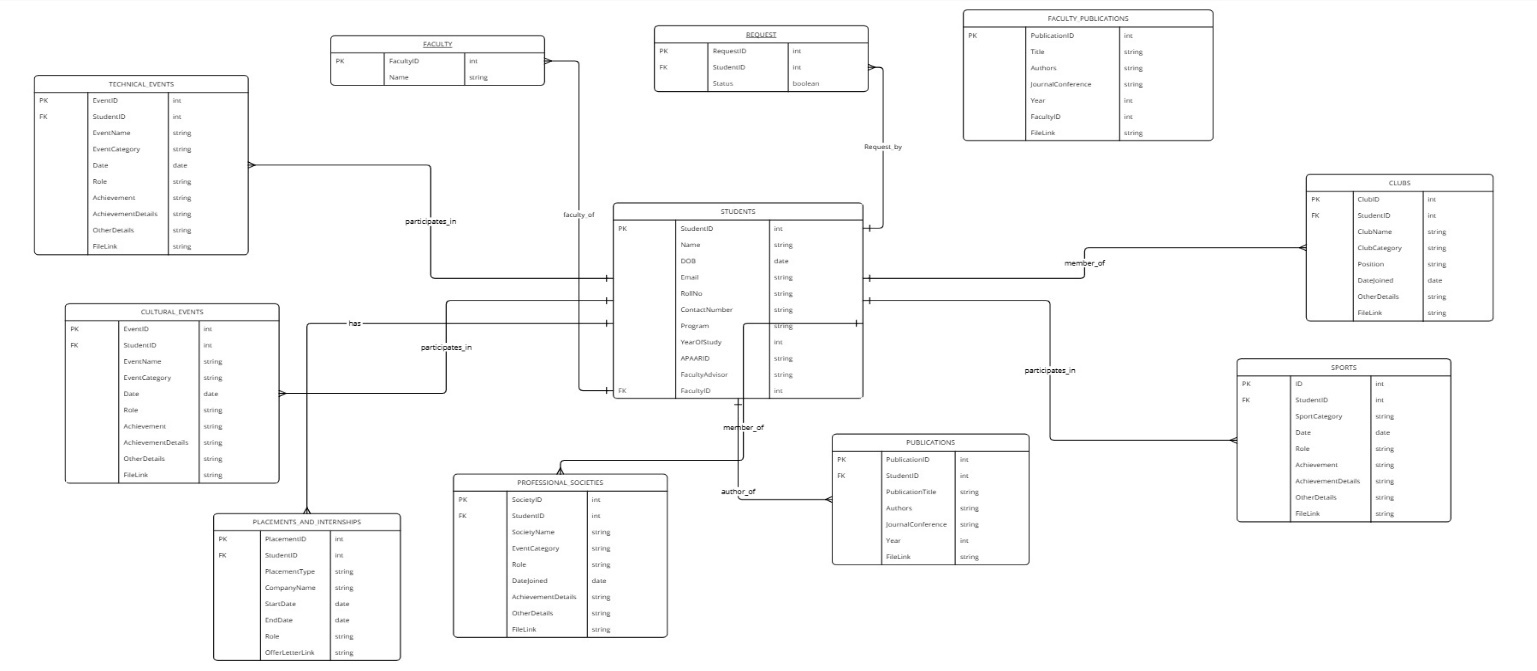
By utilizing JOIN queries and indexing, the system can quickly access and display relevant information, enhancing the overall user experience for students, faculty, and administrators.

**5.2.3. Supports Future Expansions – Easily Scalable for Additional Modules**

One of the key strengths of this database design is its flexibility and scalability.

This ability to expand and adapt ensures that the database remains relevant and useful as institutional needs evolve.

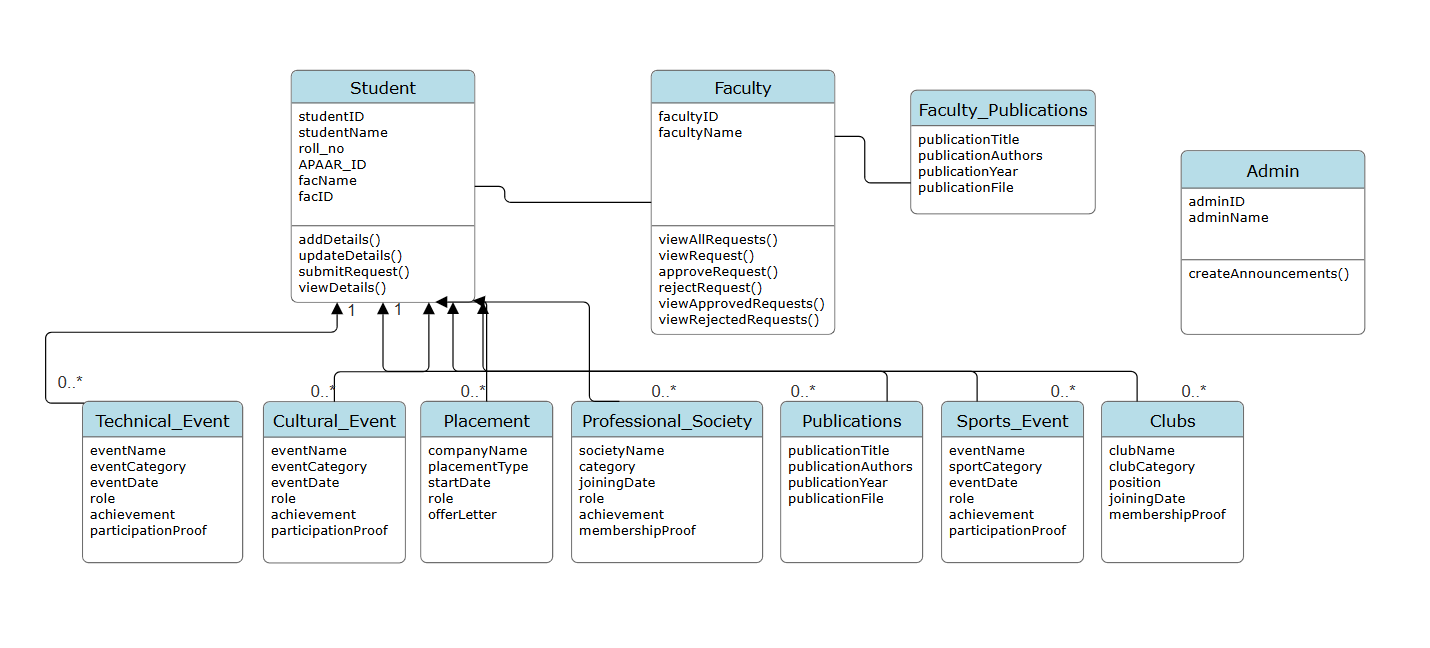
**5.3 ER diagram**

****

**5.4. Conclusion**

This database design ensures structured data storage, quick retrieval and future scalability, making it a powerful tool for student and faculty management. By implementing this system, institutions can enhance efficiency and provide a seamless experience for students and administrators**.**

**6. Class Diagram**

****

**End of the document. Thank you.**