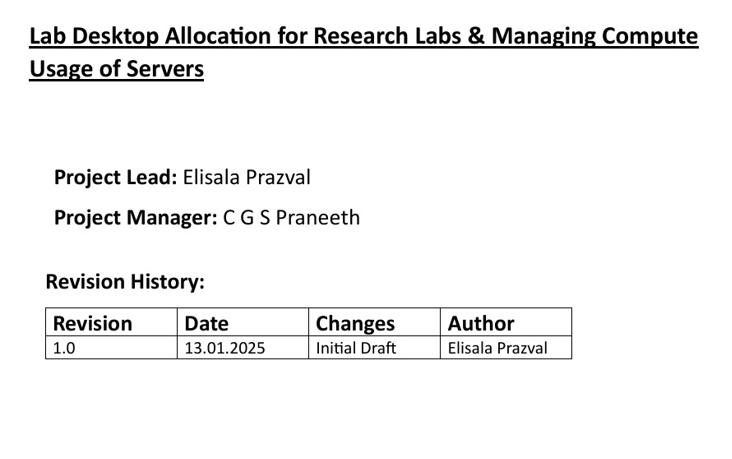
****

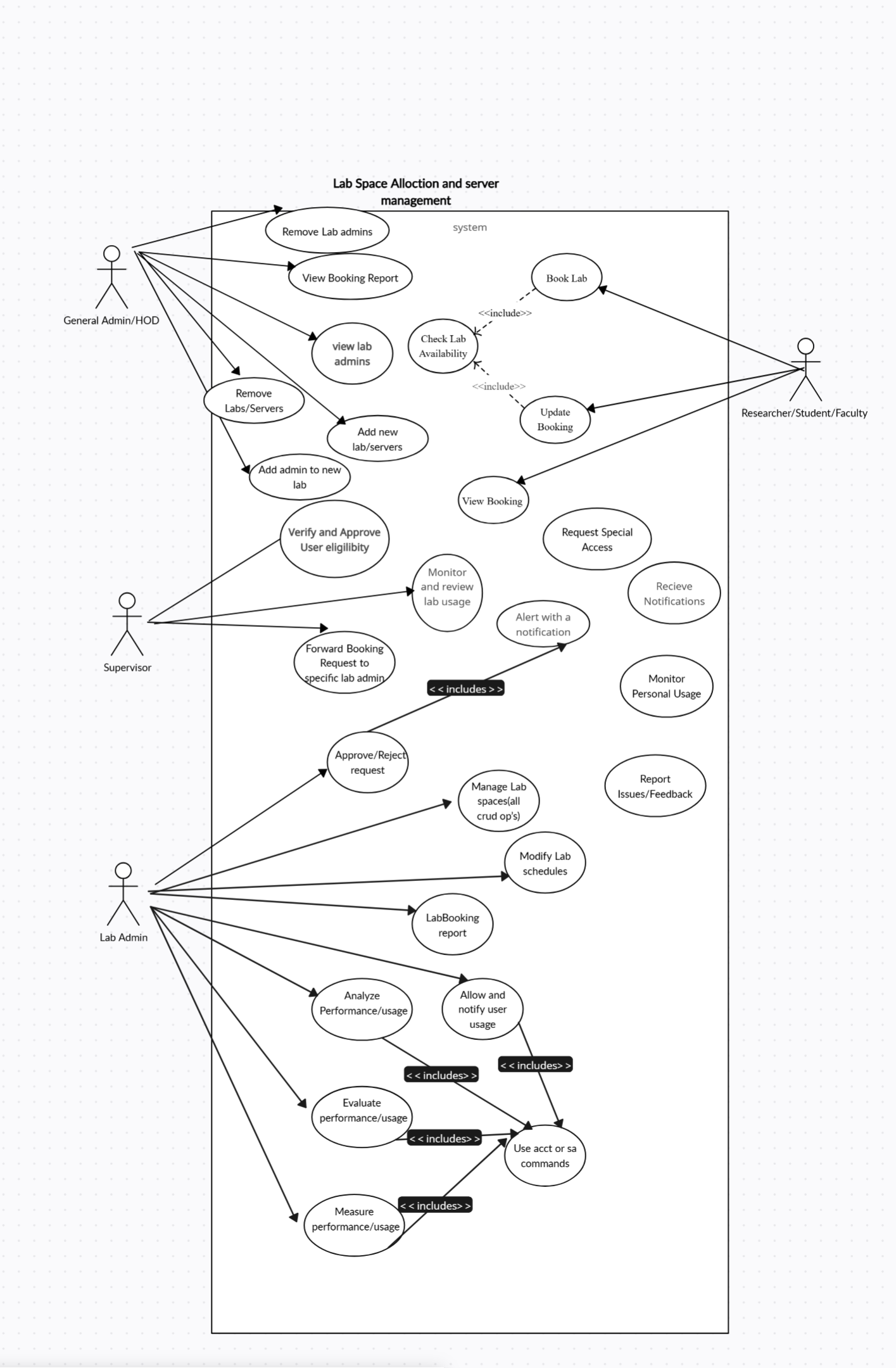
**TEAM NUMBER – 2 (FN)**

**1. ELISALA PRAZVAL - B220841CS**

**2. C G S PRANEETH - B220234CS**

**3. C. HIMAJALA REDDY – B220243CS**

**USE-CASE DIAGRAM**



**Functional Requirements**

**1. Inventory Management**

**F1.1** The system shall maintain a centralized inventory of all desktops, lab equipment, and server resources, accessible to administrators.  
**F1.2** The system shall track real-time availability of all resources, including desktops, lab equipment, and servers.  
**F1.3** The system shall allow administrators to update the status of resources as available, booked, blocked, under maintenance, or decommissioned.

**2. Resource Booking & Allocation**

**F2.1** The system shall allow students, researchers, and faculty to book lab desktops and resources based on real-time availability.  
**F2.2** The system shall implement a priority-based allocation mechanism where faculty and researchers get priority over students.  
**F2.3** The system shall allow manual or automated approval of bookings based on admin-defined policies.  
**F2.4** The system shall provide a booking history log for users and admins to track past and upcoming reservations.

**3. Server Allocation Mechanism**

**F3.1** The system shall allow faculty, researchers, and students to request server resources for computing tasks.  
**F3.2** The system shall support dynamic allocation of server resources based on real-time demand and workload.  
**F3.3** The system shall allow manual or automated allocation of servers by administrators.  
**F3.4** The system shall implement priority-based allocation where faculty and researchers get higher priority over students.  
**F3.5** The system shall allow admins to reallocate server resources dynamically based on system load.  
**F3.6** The system shall enforce role-based access control (RBAC) to restrict server access based on user roles.  
**F3.7** The system shall provide real-time monitoring of allocated servers, including resource usage and active sessions.  
**F3.8** The system shall send alerts to users and admins for resource overuse, failures, or nearing capacity limits.

**4. User Access & Authentication**

**F4.1** The system shall implement Role-Based Access Control (RBAC) to assign permissions based on user roles:

* Admin: Full access to resource management, user management, and reports.
* Lab Admin: Manage bookings, approve/reject requests, and monitor lab usage.
* Faculty/Researchers/Students: Request and manage bookings within their assigned permissions.

**F4.2** The system shall support Single Sign-On (SSO) integration with institutional login credentials.  
**F4.3** The system shall allow administrators to manage access rights and modify permissions for specific users.  
**F4.4** The system shall enforce role-based login functionality to restrict users from accessing unauthorized resources.

**5. Real-Time Monitoring & Dashboard**

**F5.1** The system shall provide an interactive dashboard for administrators to monitor real-time resource usage.  
**F5.2** The dashboard shall display user-specific resource usage (e.g., total lab hours used, number of bookings).  
**F5.3** The system shall send alerts and notifications for critical events such as resource overuse, system failures, and maintenance schedules.

**6. Integration with Existing Research Tools**

**F6.1** The system shall support API-based integration with research tools, lab software, and cloud computing services.  
**F6.2** The system shall allow users to export usage data and reports in formats such as CSV, PDF, and JSON.

**7. Feedback & Reporting Mechanism**

**F7.1** The system shall provide a feedback system for users to report issues related to lab equipment, servers, or booking problems.  
**F7.2** The system shall allow administrators to view, categorize, and resolve reported issues.  
**F7.3** The system shall generate usage analytics and reports for administrators, including:

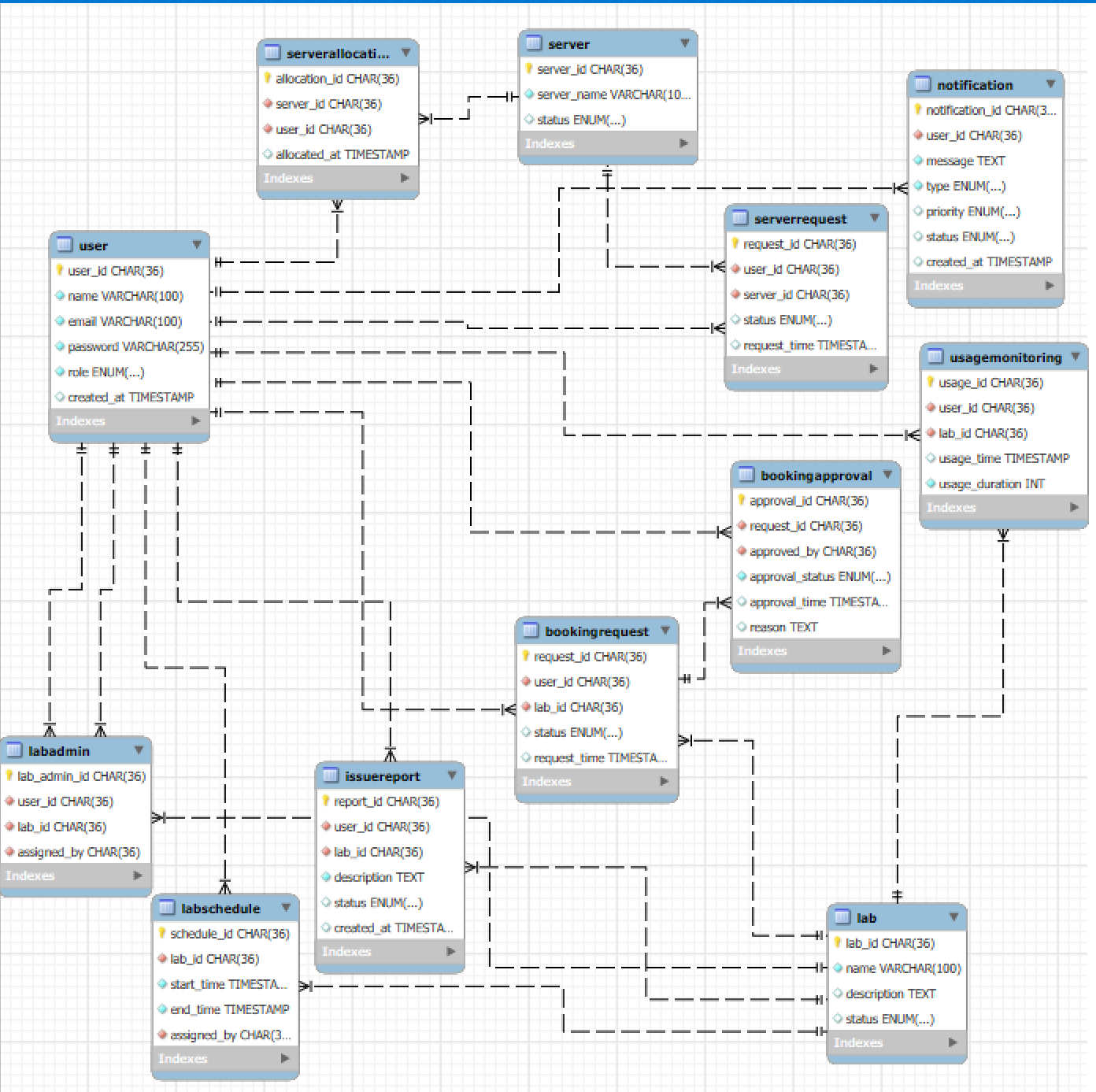
* Resource utilization statistics.
* Server allocation efficiency.
* Lab occupancy trends.

**F7.4** The system shall allow users to report malfunctioning desktops, servers, or other resources, which will be assigned to lab admins for resolution.

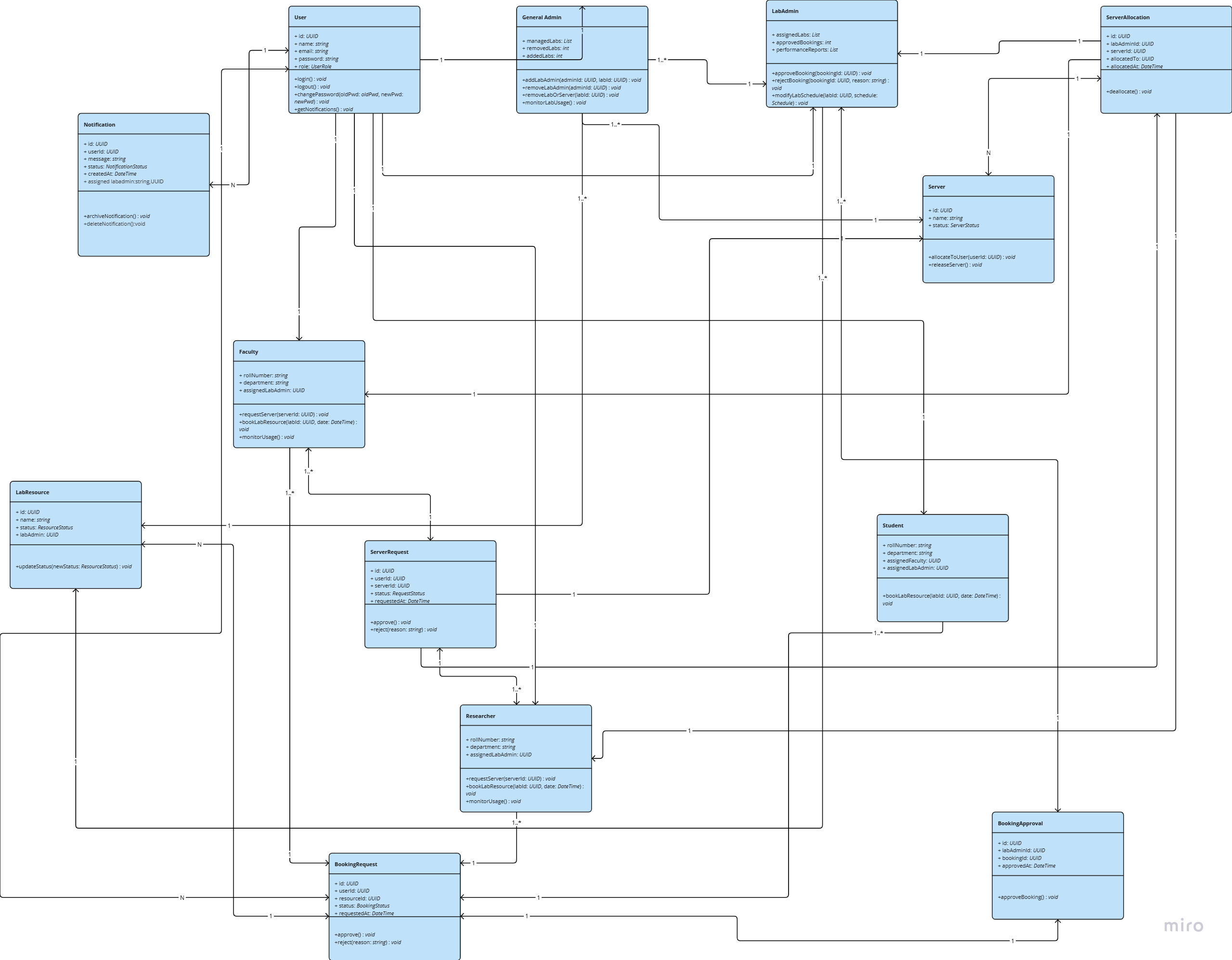
**Non-Functional Requirements:**

1. **Performance**:
   * The platform should be able to handle up to 10,000 concurrent users without performance degradation.
   * Response time for contest submissions should be under 2 seconds.
2. **Scalability**:
   * The system should be designed to scale horizontally for handling increasing numbers of users and contests.
   * The backend architecture should support adding additional servers as needed without major changes to the system.
3. **Availability**:
   * The platform should have an uptime of 99.9% or higher.
   * Maintenance windows should be scheduled with prior user notifications.
4. **Security**:
   * User data, including passwords and sensitive profile information, should be stored securely with encryption (e.g., bcrypt for passwords).
   * The system should have measures to prevent common web vulnerabilities like SQL injection, XSS, and CSRF.
   * User authentication should be managed with secure, token-based mechanisms (e.g., JWT).
5. **Usability**:
   * The UI/UX should be intuitive and responsive, providing an optimal experience across devices (desktop, tablet, mobile).
   * The platform should include detailed tooltips and instructions for new users.
6. **Maintainability**:
   * Code should be modular and well-documented to facilitate ease of maintenance and future feature additions.
   * There should be a version control system (like Git) in place for collaboration.
7. **Backup and Recovery**:
   * Regular backups of the database and user data should be scheduled.
   * The platform should have a disaster recovery plan in place, allowing restoration within a defined time frame (e.g., 4 hours).

**DATABASE DESIGN**

****

**CLASS DIAGRAM**

****

**Workflow :** user(researcher,faculty/student)=>booking request=>lab admin(approves)=>record added to the booking approval class=>server request(If needed by the user)=>serverallocation(done by the lab admin and checks availability in server class to know info about servers )=>notification(if every thing done)