Lab Desktop Allocation for Research Labs & Managing Compute Usage of Servers

Project Lead: Elisala Prazval

Project Manager: C G S Praneeth

Revision History:

Revision	Date	Changes	Author
1.0	13.01.2025	Initial Draft	Elisala Prazval

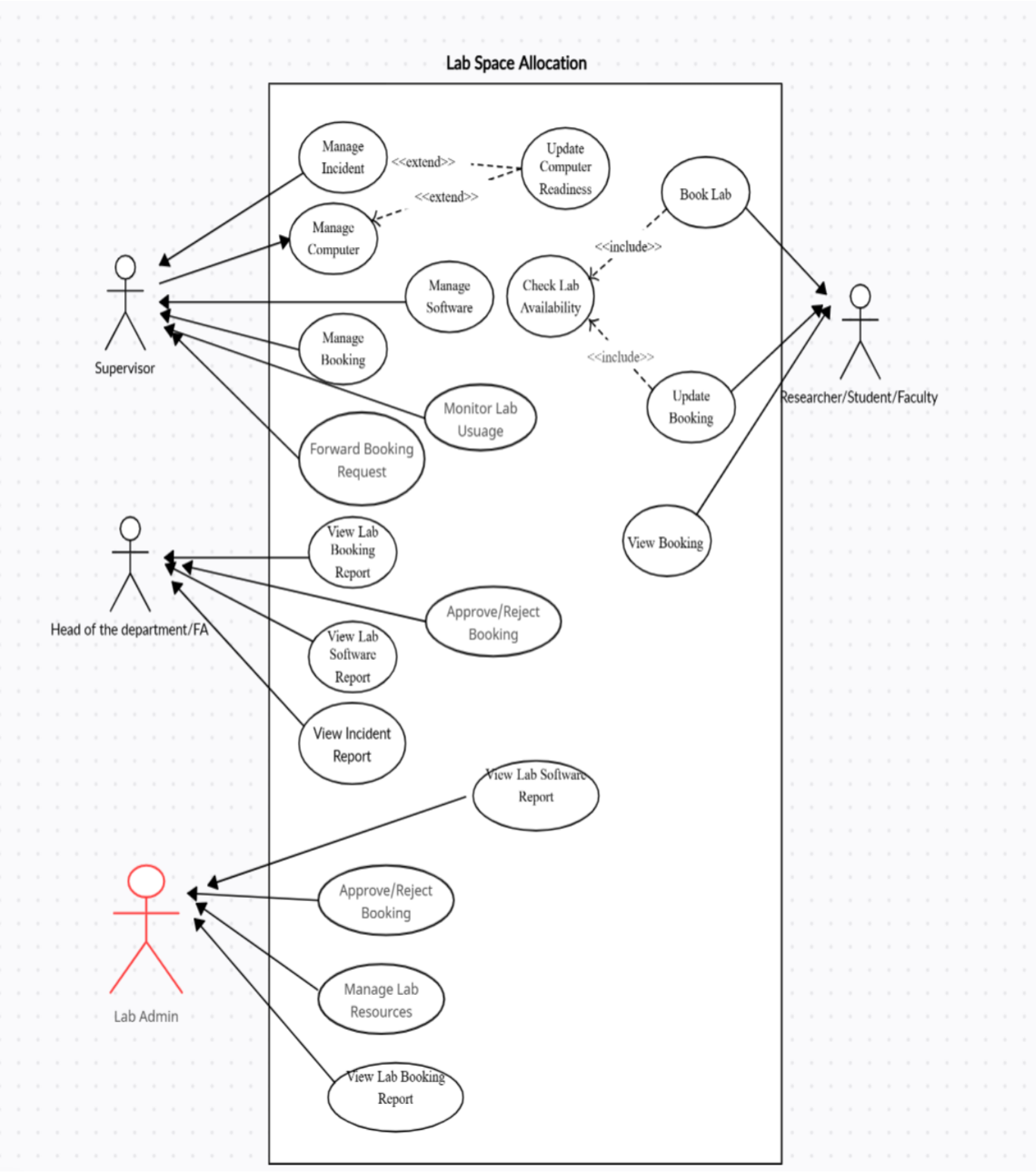
TEAM NUMBER – 2 (FN)

1. ELISALA PRAZVAL - B220841CS

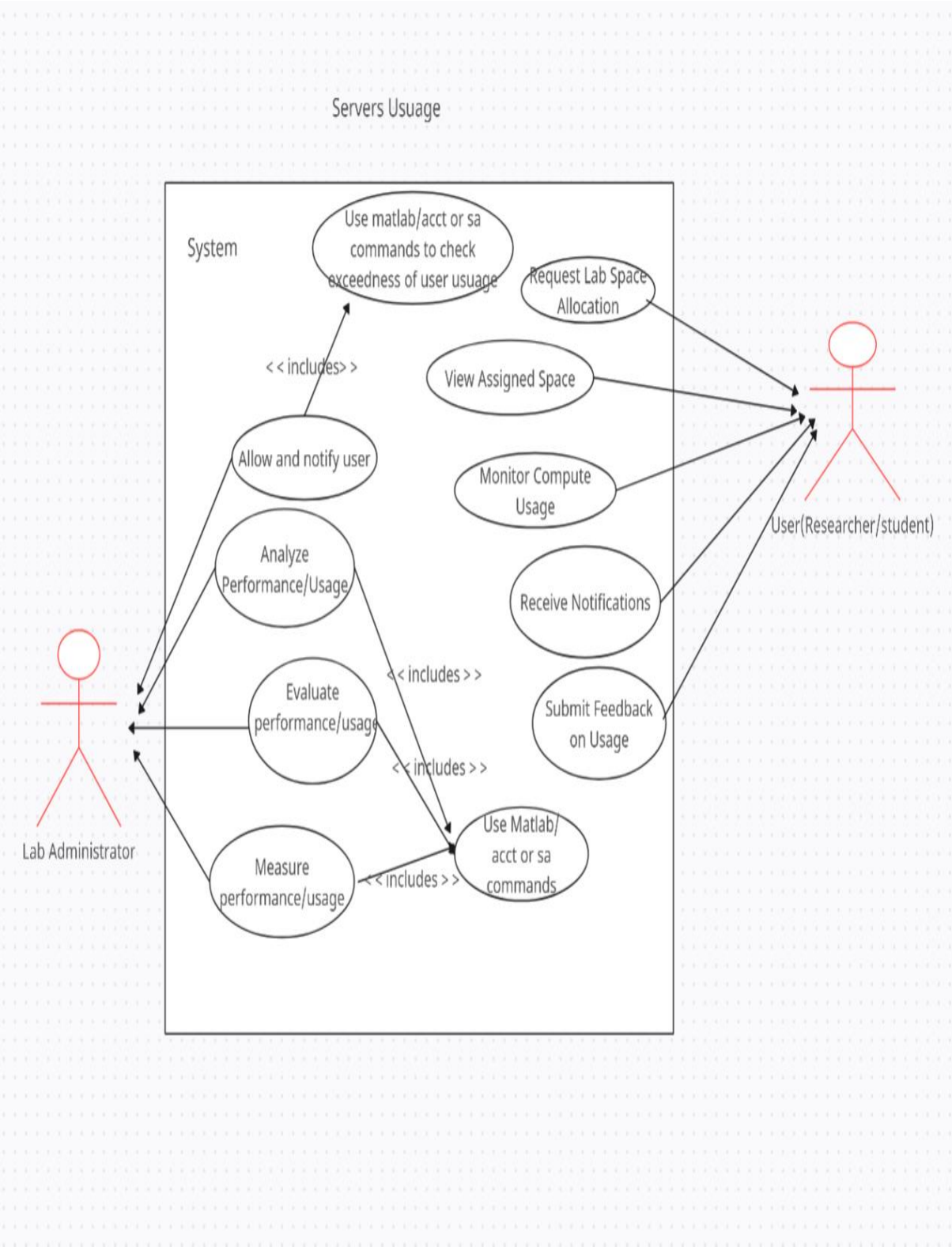
2. C G S PRANEETH - B220234CS

3. C. HIMAJALA REDDY – B220243CS

USE CASE DIAGRAM FOR LAB SPACE ALLOCATION



USE CASE DIAGRAM FOR SERVERS USUAGE



FUNCTIONAL REQUIREMENTS:

These define the core functionalities of the system.

Inventory Management:

- Maintain a centralized inventory of all desktops, lab equipment, and resources.
- Track resource availability in real-time.
- Update resource status (available, booked, blocked, etc.).

Resource Booking & Allocation:

- Allow users to book lab desktops and resources based on availability.
- Implement a priority-based allocation system (e.g., faculty, researchers, students).
- Enable automated/manual approval for certain bookings if required.

Server Allocation Mechanism:

- Allocate server resources dynamically based on user requests and workload.
- Allow manual or automated allocation of servers for specific tasks.
- Implement priority-based allocation (e.g., faculty, researchers, students).
- Enable reallocation of resources based on real-time demand.
- Restrict server access based on user roles (admin, faculty, student, researcher).
- Provide real-time monitoring of allocated servers.
- Send alerts for resource overuse or failures.

User Access & Authentication:

- Implement role-based access control (RBAC) (e.g., admin, faculty, student).
- Enable SSO (Single Sign-On) or integration with institutional login for authentication.

- Allow admin privileges to manage access and permissions for different users.
- Implement role-based login functionality to ensure users can only access permitted resources.

Real-Time Monitoring & Dashboard:

- Provide an interactive dashboard for administrators to monitor resource usage.
- Display only user usage time instead of live system health metrics.
- Send alerts and notifications for critical issues or high resource consumption.

Integration with Existing Research Tools:

- Enable API-based integration with existing lab tools, research software, and cloud services.
- Allow exporting data and reports in multiple formats for research documentation.

Feedback & Reporting Mechanism:

- Provide a feedback system for users to report issues or suggest improvements.
- Generate usage analytics and reports for administrators to track system efficiency.
- Allow incident reporting for malfunctioning resources.

2. Non-Functional Requirements

These define the system's quality attributes and constraints.

Performance:

- Ensure fast response time for system interactions.
- Optimize server efficiency to handle multiple users.
- Use caching to reduce database load.

Scalability:

- Support expansion of server allocation capabilities as demand increases.

Availability & Reliability:

- Ensure 99.9% uptime with automatic failover mechanisms for uninterrupted service.
- Maintain redundant data backups to prevent data loss.
- Implement proactive monitoring with automated alerts for potential failures.
- Use load distribution strategies to prevent system overloads.

Security:

- Use encryption for secure data transmission.
- Implement multi-factor authentication (MFA).
- Restrict unauthorized access with role-based permissions.
- Enable session timeouts and auto-logout.

Usability:

- Provide a user-friendly interface with documentation and tooltips.

Maintainability & Extensibility:

- Use a modular system for easy updates.
- Support automatic updates with minimal downtime.

Compliance & Legal Requirements:

- Adhere to data privacy laws (e.g., GDPR, institutional policies).
- Maintain audit trails for compliance tracking.