**Research Lab Equipment Booking System**

*Final Project Report*

Christian Manangan 11755967, Kyle Hawkins 11759096, Ethan Frazier 11792962

Dr. Kumar

04/27/2025

CptS 451: Introduction to Database Systems

**Abstract**: The Research Lab Equipment Booking System aims to make it easier for students and researchers to reserve lab equipment. Users can register and login, view equipment availability, book equipment, and track their past reservations. Administrators oversee bookings, enforce rules, and generate reports to optimize resource use. The system includes user authentication, equipment tracking, and real time updates to prevent scheduling conflicts and ensure fair access. This project focuses on developing a database-driven solution with a three-tier client server architecture, ensuring scalability, security, and maintainability. The system will be created using Render as the cloud platform, and PostgreSQL as the relational database. These two technologies were chosen because Render integrates seamlessly with PostgreSQL which is very convenient for database creation, smooth workflow, and ease of access of information. Key features include role-based access control, an approval process for reserving equipment, notifications, and detailed usage logs. By creating a system in this manner, it reduces inefficiencies, helps prevent scheduling conflicts, and overall improves the accessibility and management of lab resources.

**TABLE OF CONTENTS**

[I. Introduction 4](#_Toc196619851)

[I.1. Overview of the Project 4](#_Toc196619852)

[I.2. Problem Statement 4](#_Toc196619853)

[I.3. Objectives and Scope of the Project 4](#_Toc196619854)

[I.4. Importance and Potential Impact 4](#_Toc196619855)

[II. Functional and Non-Functional Requirements 4](#_Toc196619856)

[II.1. Functional Requirements 4](#_Toc196619857)

[II.2. Writing User Stories 5](#_Toc196619858)

[II.3. Non-Functional Requirements 7](#_Toc196619859)

[III. Database Design: ER Diagram & Tables 8](#_Toc196619860)

[III.1. Entity-Relationship (ER) Diagram 8](#_Toc196619861)

[III.2. Conversion of ER Diagram to Relational Tables 9](#_Toc196619862)

[IV. SQL Implementation 9](#_Toc196619863)

[IV.1. SQL DDL Scripts 9](#_Toc196619864)

[IV.2. Sample DML Scripts 10](#_Toc196619865)

[V. Feature Implementations 10](#_Toc196619866)

[VI. Query Design 11](#_Toc196619867)

[VII. User Interface Snapshots 11](#_Toc196619868)

[VIII. Limitations & Future Scope 14](#_Toc196619869)

[IX. Conclusion 15](#_Toc196619870)

# Introduction

# Overview of the Project

The project is a lab equipment rental system that allows users to log on to an online website and create reservations for the equipment they desire. The reservations and usage logs of those who used the equipment will all be handled internally by the system and can be manually edited by a system admin with high tier credentials. Students and Researchers have the same level of clearance in the database and can rent any equipment that is available in the system.

# Problem Statement

Developing a secure and efficient way for students and researchers to reserve equipment with ease and no conflict. If equipment is broken and should not be used by another lab member, the online system for rentals and tracking equipment will display whether an item is broken or not and can then be properly removed and disposed of instead of someone not knowing the equipment is broken- and mistakenly reserving it for a potentially risky experiment.

# Objectives and Scope of the Project

The most important goal of this system is to effectively rent equipment and gear to any person who has logged onto the site and requested equipment that is available and not rented by someone else. The scope of this project is somewhat limited in a grand scheme of who can access it, and the global usage of this system as it is only for a laboratory, designed to help with equipment rental, and tracking. Many people may be a part of the lab, and the database can hold a lot of records, but users that access this system will all be a part of the same category and in the same location at one time.

# Importance and Potential Impact

Science lab equipment can be very limited, or hard to track when the facility does not use an efficient database to store records of the equipment usage and last renter. Without a system to track this information, trying to keep watch of the equipment with physical notes and people’s word of mouth on where something has gone is inefficient. Using a database is much faster, and secure for renting equipment to users who have credentials they can verify through the system and log their equipment condition and time used so that all other users may view that information and avoid confusion about availability.

# Functional and Non-Functional Requirements

# Functional Requirements

Functional Requirements are key features, actions, and workflows of what the system must do. Below is a detailed and exhaustive list of functional requirements we defined.

1. **User Registration & Authentication**: Users such as students, researchers, and administrators must be able to register and authenticate using a secure login system.
2. **Equipment Listing & Details**: The system must be able to store a list of lab equipment along with its details like specifications and availability status, in a retrievable and modifiable way like a database.
3. **Equipment Booking System**: Users should be able to reserve equipment for a specific time slot based on availability status
4. **Usage Logging System**: The system must maintain logs of equipment usage. This may include user details and timestamps
5. **Booking Approval**: The system must allow administrators to review and to make a decision for booking requests
6. **Usage Reports & Analytics**: Administrators should be able to generate a report on user activity and equipment usage, and booking patterns
7. **User Can View Reservation Statuses**: Users should be able to see a page that shows a list with the status of their booking reservation requests and see if it's pending, denied, or approved.
8. **Admin Approve/Deny Reservation Requests**: An admin should be able to make decisions on pending reservation requests to either approve or deny requests.
9. **Admin View Reservation Requests**: An admin should be able to see a page of reservation requests that are pending approval.
10. **Lab Equipment Status Is Shared and Accurate**: The availability status of equipment should be accurately reflected/shared and seen by all users.

# Writing User Stories

User stories describe different interactions of users with the system. Below are all the user stories the team has come up with based on the functional requirements.

1. As a student, I want to register as a student through the system so that I can verify my role as a student and reserve equipment.
2. As a researcher, I want to register as a researcher through the system so that I can verify my role as a researcher and have priority reserving equipment.
3. As an administrator, I want to register as an admin through the system so that I can handle administrative tasks like generating usage reports.
4. As a student, researcher, or administrator, I want to log into the system securely so that I can access my account and perform role-specific actions.
5. As a student or researcher, I want to be able to view equipment along with its details so that I can rent and achieve my academic goals.
6. As a student or researcher, I want to be able to see if equipment is available or not, so that I can better plan for what I can do.
7. As a student or researcher, I want to be able to reserve equipment for a specific time slot based on availability status so that I can reserve equipment in an orderly and structured manner.
8. As a student or researcher or administrator, I want to be able to view system logs of equipment usage that includes user details and timestamps so that I can use that information to improve scheduling and identify misuse
9. As an administrator, I want to view a list of current booking requests, so that I can handle and see equipment demands.
10. As an administrator, I want to review and make decisions for booking requests so that I can prevent misuse and maintain resource availability
11. As an administrator, I want the system to store and to be able to see usage analytics, so that I can better perform administrative duties and understand equipment demands.
12. As an administrator, I want to be able to see a user's equipment rental history, so that I can understand and track use.
13. As a student or researcher, I want to see a page that shows a list with the status of their booking reservation so that I can see if it’s pending, denied, or approved
14. As a student or researcher, I want to be notified of whether my booking request was approved or denied so that I can plan accordingly.
15. As a student or researcher, I want to modify or cancel my booking request so that others can reserve the equipment
16. As an administrator, I want to see a page of reservation requests that are pending approval so that I can be informed about the amount of booking requests and make decisions.
17. As a student or researcher, I want to see the status of equipment so that I can see if equipment is available for a given day or has already been booked by another user for that date.

# Non-Functional Requirements

Non-functional requirements outline the operational qualities of the system, such as performance, system availability, maintainability, and security. The system needs to ensure user experience to ensure that it meets quality standards beyond core functionality. Details of non-functional requirements are as follows.

|  |  |  |
| --- | --- | --- |
| **Non-Functional Requirements** |  | **Description** |
| Security |  | User data such as passwords should be encrypted and stored |
| Usability |  | Users should be able to use the system intuitively and with minimal |
| Scalability |  | System must be able to accommodate various expansions |
| Availability |  | System must be up and running for the majority of time |
| Maintainability |  | Bug fixes and updates should be simple and easy for the system |
| Compliance |  | Meets legal and regulatory requirements |
| Loose Coupling of UI and Database |  | The systems UI should be loosely coupled to the database meaning that if the underlying way the data is stored in the database changes, it won’t also require remaking the UI. |
| Role Based Access |  | The system should ensure role-based access and views so that unauthorized users can't see or perform administrative functions. |
| Performance |  | System must process booking requests in a quick/timely manner, no longer than ½ a minute for a user's booking request to be propagated and viewable by the admin page. |
| Accuracy |  | The accuracy of the status of data should always be above 99%, across all user devices. Meaning that all users should be able to see the same, accurate, data and details for the lab equipment. |

# Database Design: ER Diagram & Tables

# Entity-Relationship (ER) Diagram

An ER diagram represents the database schema with entities, attributes, and relationships. Primary keys are attributes that are underlined. The cardinality (1:1, 1:M, M:1, and M:M) for each entity relationship set are labeled on the diagram. The ER diagram below was created using a UML tool Draw.io.

A diagram of a company

AI-generated content may be incorrect.

# Conversion of ER Diagram to Relational Tables

To turn the ER diagram into tables we make the squares into tables, and relationships that are many to many tables too. Then we make the attributes into columns. The primary keys are the underlined or id fields. Foreign keys are the relationships between entities. Then we normalized the tables and made multi-value attributes into their own tables. Please see below our relational tables.

A diagram of a computer

AI-generated content may be incorrect.

# SQL Implementation

To implement the database schema for the Lab Equipment Booking system, we used PostgreSQL. This section shows and explains the main SQL scripts used to define and initialize the database. There are two main components to the implementation:

* Data Definition Language (DDL): SQL statements used to create tables, constraints, and indexes
* Data Manipulation Language (DML): SQL statements used to insert test data for development and demonstration purposes

# SQL DDL Scripts

The database schema was created using CREATE TABLE statements with proper data types, constraints, and foreign keys references. Each table supports a distinct system domain:

* roles: Holds predefined user roles such as Admin, Researcher, and User
* users: Stores registered users, including passwords that have been hashed
* equipment: Tracks lab equipment, their categories, current status, and total inventory
* reservation: Booking requests by users along with their request date, start date, end date, quantity, and status
* reservation\_admin: Provides a map of administrators who made the decisions on the booking requests
* admins: A list of users who have admin privileges
* usage\_logs: Keeps track of use occurrences to support administrative analytics and equipment demand tracking
* suppliers and supplied: monitor equipment’s history of supply and origin
* notifications: Stores messages sent to users regarding their reservations

Each table enforces data integrity through:

* Primary keys (SERIAL used for auto-incrementing IDs)
* Foreign key constraints (ON DELETE CASCADE to ensure referential cleanup)
* Unique constraints to avoid duplicates
* Check constraints to enforce valid values for status and quantity fields

Sample creation of the reservation table:

CREATE TABLE IF NOT EXISTS reservations (

    id SERIAL PRIMARY KEY,

    user\_id INTEGER REFERENCES users(id) ON DELETE CASCADE,

    equipment\_id INTEGER REFERENCES equipment(id) ON DELETE CASCADE,

    res\_request\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    res\_start\_date TIMESTAMP NOT NULL,

    res\_end\_date TIMESTAMP NOT NULL,

reservation\_status VARCHAR(50) CHECK (reservation\_status IN ('pending', 'approved',

'denied', 'cancelled', 'in\_use')),

    reserved\_quantity INTEGER CHECK (reserved\_quantity > 0),

    CONSTRAINT unique\_reservation UNIQUE(user\_id, equipment\_id, res\_start\_date)

);

# Sample DML Scripts

To enable development, testing, and demonstration of the Lab Equipment Booking system, the database was populated with sample data using INSERT INTO statements. Important notes are:

* Preloading roles to ensure role integrity
* Seeding users with bcrypt-hashed passwords using PostgreSQL’s crypt() and gen\_salt() functions
* Added equipment entries with varied statues
* Created sample reservations, each with a distinct user\_id, equipment\_id, and start\_date combination to adhere to the unique constraint
* Assigning reservation decisions by admin users
* Populating usage logs to support analytics and usage history
* Providing suppliers and provided equipment data
* Creating user notifications that are uniquely constrained to prevent sending out the same message more than once

Below are sample entries from the reservations table:

INSERT INTO reservations\_admins (reservation\_id, admin\_id, approval\_date, decision) VALUES

(1, 1, CURRENT\_TIMESTAMP, 'approved'),

(2, 1, CURRENT\_TIMESTAMP, 'approved'),

(3, 1, CURRENT\_TIMESTAMP, 'denied')

ON CONFLICT (reservation\_id, admin\_id) DO NOTHING;

The use of ON CONFLICT DO NOTHING ensures that scripts can be run multiple times without duplicating data or causing insert errors.

# Feature Implementations

The following is a list of the key features that were developed as part of our system, things like user authentication, equipment booking, reservation management, analytics, admin features, etc.

* **User Authentication System**
* **Purpose:** Secure access with role-based permissions (Admin, Researcher, Student). Handles user registration, login, and profile management.
* **Flow from UI to Database:**
  1. User fills in registration/login form
  2. Frontend sends request to backend auth endpoints
  3. Backend validates credentials (hashed via Werkzeug)
  4. On login: generates JWT token; on registration: creates new user record
  5. Frontend stores token in local storage for session persistence
* **Database Integration:** Uses *users* and *roles* tables with foreign-key relationships; passwords stored as secure hashes.
* **Equipment Booking System**
* **Purpose:** Enables users to browse and reserve lab equipment for specific dates and quantities.
* **Flow from UI to Database:**
  1. User views equipment list (fetched from *equipment* table)
  2. Clicks “Book” → opens booking modal
  3. Backend endpoint checks availability for chosen dates/quantity
  4. If available, a new record is inserted into reservations with status = “pending”
  5. Notification entry created; equipment availability updated if fully booked
* **Database Integration:** Reads from *equipment*; writes to *reservations* and *notifications*; may update *equipment* status.
* **Reservation Management**
* **Purpose:** Allows users to view/edit/cancel their own reservations; admins can approve or reject pending requests.
* **Flow from UI to Database:**
  + **User actions:**
    - Edit/cancel → frontend issues *PUT/DELETE* to */api/reservations/:id* after availability check
  + **Admin actions:**
    - Approve/deny → updates *reservations.status* and inserts a record into *reservation\_admins*; creates a notification
* **Database Integration:** Interacts with *reservations, reservation\_admins,* and *notifications* tables; ensures consistency of equipment availability.
* **Notifications System**
* **Purpose:** Informs users about reservation status changes and other system events.
* **Flow from UI to Database:**
  1. Backend creates rows in *notifications* on key events (new reservation, status change)
  2. Frontend fetches */api/notifications*, filters by *user\_id,* and displays messages
* **Database Integration:** Simple CRUD on *notifications* table; each notification links to a user.
* **Admin Analytics Dashboard**
* **Purpose:** Gives administrators insights into equipment usage, reservation trends, and popular items.
* **Flow from UI to Database:**
  1. Frontend calls analytics endpoints (e.g. */api/admin/analytics/equipment-usage*)
  2. Backend runs SQL queries with *JOIN, GROUP BY,* and aggregates over *equipment, reservations,* and *usage\_logs*
  3. JSON results are rendered as charts and tables in the dashboard
* **Database Integration:** Complex read-only queries for reporting; does not modify data.

# Query Design

The following are SQL queries used against our PostgresSQL database which is hosted on Render.

* **Equipment Usage Analytics**

SELECT

e.id,

e.equip\_name,

COUNT(r.id) AS reservation\_count

FROM Equipment e

JOIN Reservation r

ON r.equipment\_id = e.id

GROUP BY e.id, e.equip\_name;

*Purpose:* Counts total reservations per equipment item to feed the admin’s usage statistics.

* **Reservation Trends Analysis**

SELECT

DATE\_TRUNC('month', res\_request\_date) AS month,

COUNT(id) AS reservation\_count

FROM Reservation

WHERE res\_request\_date >= NOW() - INTERVAL '365 days'

GROUP BY month

ORDER BY month;

*Purpose:* Shows monthly counts of new reservations over the past year for trend visualization in the analytics.

* **Popular Equipment Identification**

SELECT

e.id,

e.equip\_name,

COUNT(r.id) AS reservation\_count

FROM Equipment e

JOIN Reservation r

ON r.equipment\_id = e.id

GROUP BY e.id, e.equip\_name

ORDER BY reservation\_count DESC

LIMIT 10;

*Purpose:* Lists the top 10 most-reserved items to show in the analytics.

* **Equipment Availability Check**

SELECT

COALESCE(SUM(r.reserved\_quantity), 0) AS total\_reserved

FROM Reservation r

WHERE r.equipment\_id = :equipment\_id

AND r.reservation\_status IN ('pending','approved')

AND r.res\_start\_date <= :end\_date

AND r.res\_end\_date >= :start\_date;

*Purpose:* Gets how many units are already reserved in the requested period to determine availability.

* **User Reservation History**

SELECT

r.id,

r.res\_start\_date,

r.res\_end\_date,

r.reservation\_status,

e.equip\_name

FROM Reservation r

JOIN Equipment e

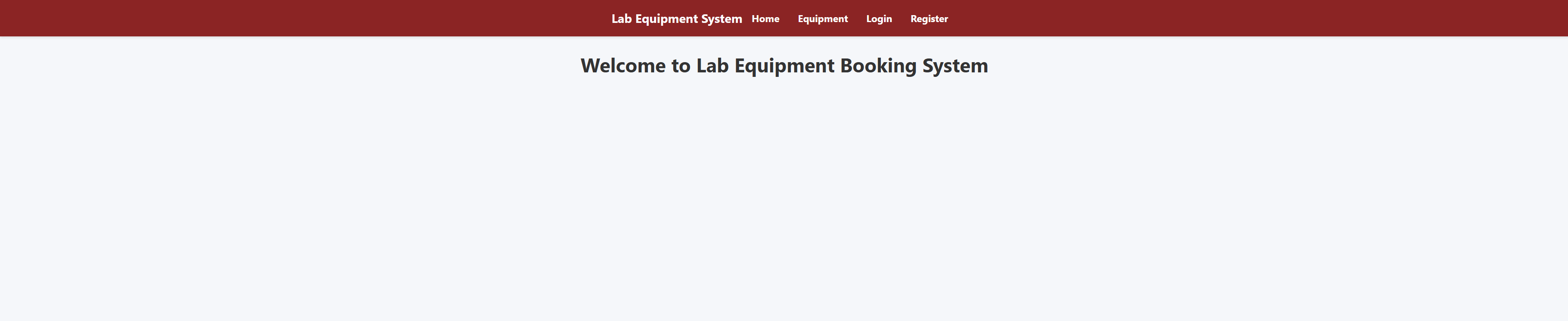
ON r.equipment\_id = e.id

WHERE r.user\_id = :user\_id;

*Purpose:* Retrieves every reservation made by a given user, including equipment names, for display in their “My Reservations” page.

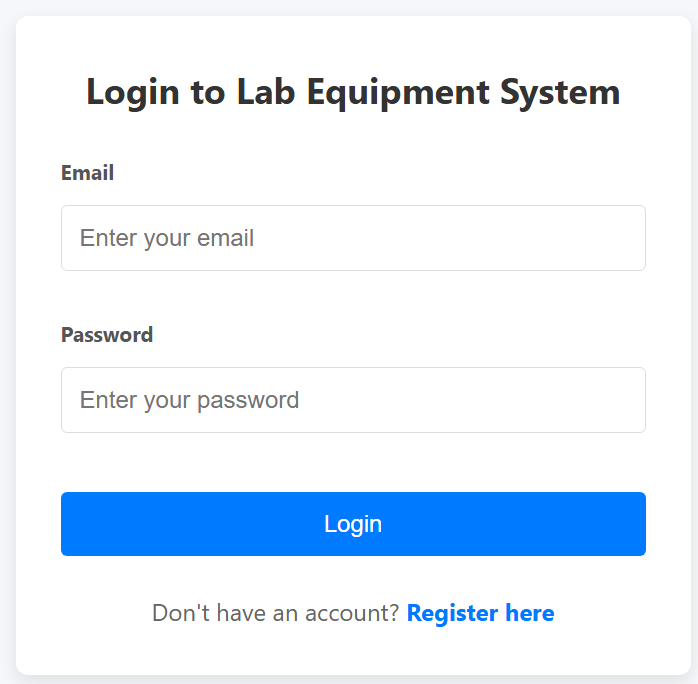
# User Interface Snapshots

This section showcases the main user interfaces we created for the Lab Equipment Booking system. A brief explanation of each screenshot’s function and database connection is included.



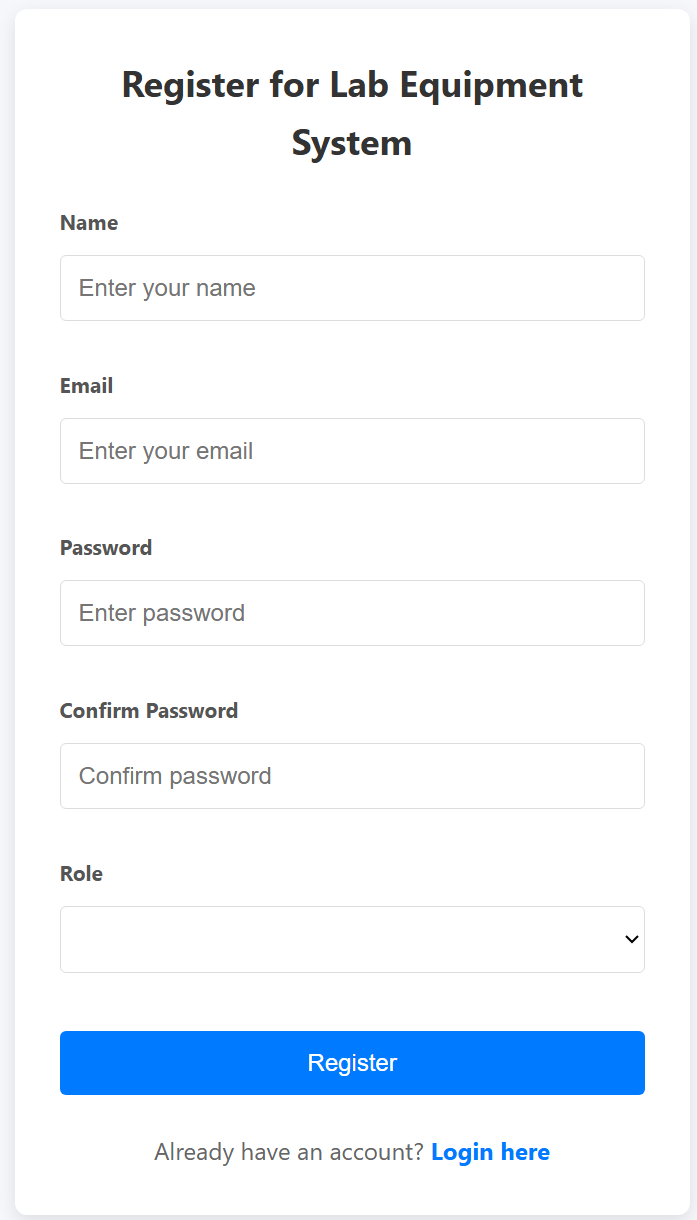
Screenshot 1: Home Page

The home page welcomes users to the Lab Booking Equipment Booking system and provides navigation links through its navbar.



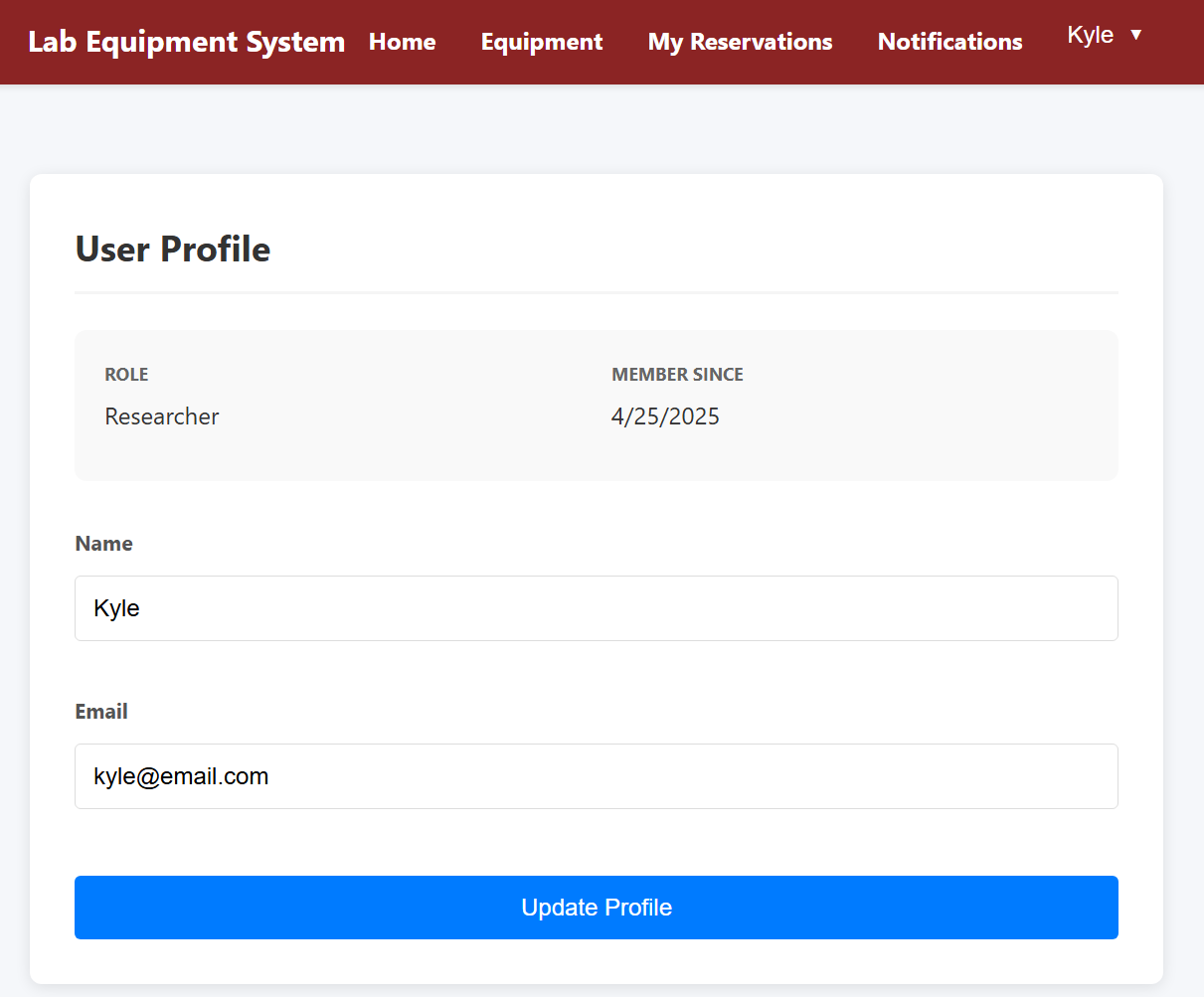
Screenshot 2: Login Page

The login page allows existing users to provide their email address and password which are compared to the users table in the database.



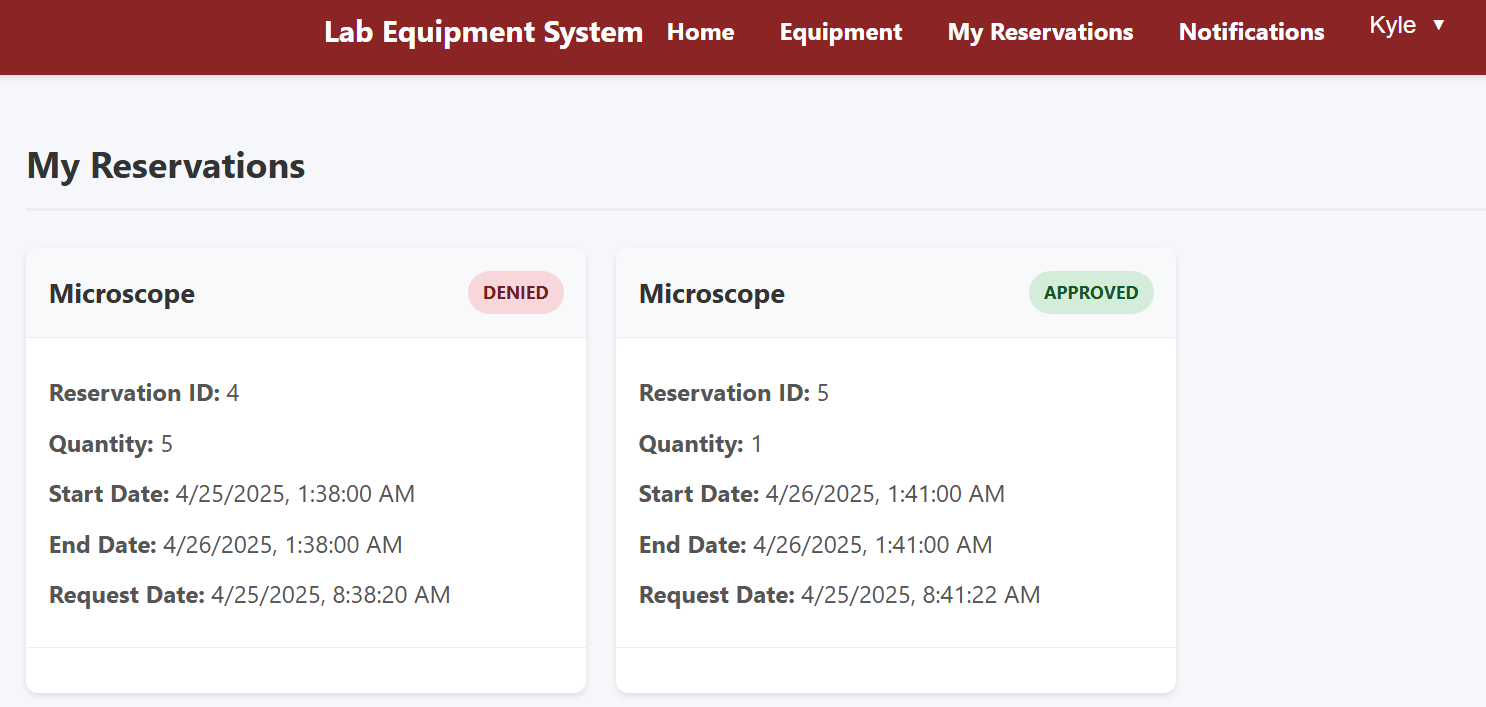
Screenshot 3: Register Page

New and interested users can create an account. They must provide their name, email address, and confirm their password. Finally, they must select a role. Upon submission, the data is inserted into the users table.



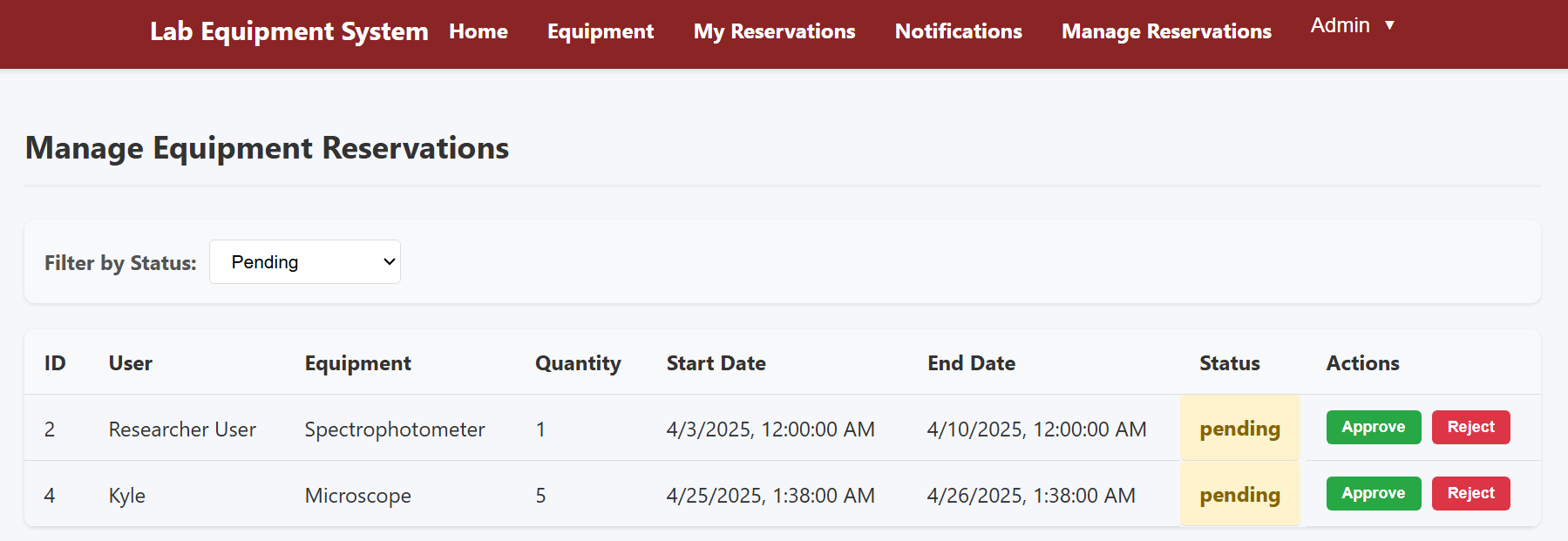
Screenshot 4: User Profile Page

Users can view and update their profile information. Changes here trigger an update to the corresponding record in the users table.



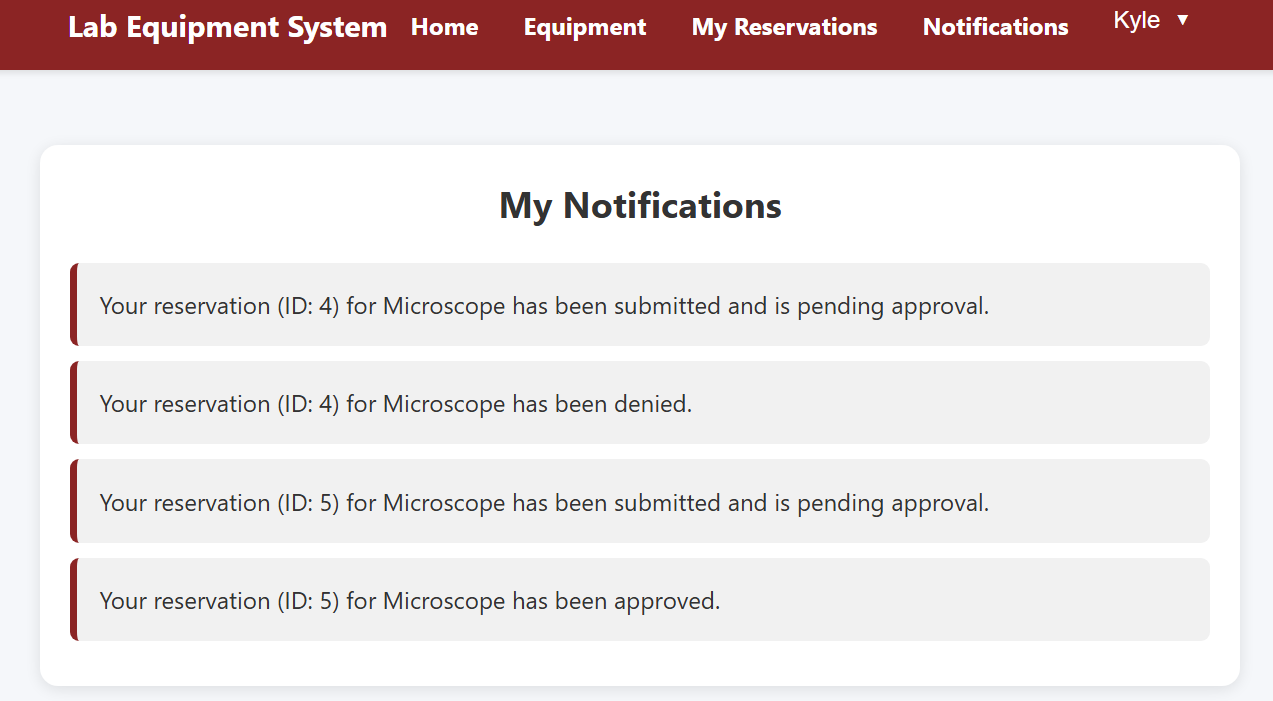
Screenshot 5: My Reservations Page

Reservations made by a user can be viewed on the My Reservations page. Users can view current and past reservations. The reservations table is used to dynamically fetch data which is connected to the equipment table for item names.



Screenshot 6: Management Equipment Reservations (Admin Page)

Admins can view all pending equipment reservations submitted by users and make a decision on them (approve or reject them). Actions here update the reservations table and create entries in the reservations\_admins table for tracking decisions.



Screenshot 7: Notifications Page

Users can view notifications generated and issued by the system about their reservation requests (approved, denied, pending). These notifications are retrieved from the notifications table where each new notification is generated for every decision made by the administrators.

A screenshot of a computer

AI-generated content may be incorrect.

Screenshot 8: Booking Equipment Modal (Booking Page)

The equipment booking model is used to select the dates and quantity for equipment that a user wants to book and when you submit the request it creates a reservation in the *reservations* table.

A screenshot of a computer

AI-generated content may be incorrect.

Screenshot 9: Analytics Dashboard (Admin Page)

This shows the admin analytics dashboard which shows analytics for equipment and booking and rental history which aggregates data from the *equipment* and *reservations* tables to provide analytical insights.

A screenshot of a computer

AI-generated content may be incorrect.

Screenshot 10: Analytics Users List (Admin Page)

This shows the view where you can search users and see their account details as well as the option to view their reservation history which pulls data from the *user* table.

A screenshot of a computer

AI-generated content may be incorrect.

Screenshot 11: View User Reservation History (Admin Page)

This view shows the reservation history for a given user pulling from the user and reservation table.

# Limitations & Future Scope

While our Lab Equipment Booking system meets the core functionality requirements, there are several limitations that must be discussed. The limitations of this system are mainly due to time constraints and design choices:

* No Email integration: Notifications are stored in the database and shown in the UI. There is no real-time emailing or SMS functionality
* No Calendar View: Reservations are shown in a list. Perhaps a calendar view would improve visibility and scheduling
* Limited Security Features: No multi-factor authentication or session management
* No Equipment Maintenance Scheduling: No way to schedule or track equipment that is in maintenance
* No Approval Queue Prioritization: Admins see pending reservations without any prioritization based on urgency.

For future scope, the following improvements and extensions are recommended:

* Integrate calendar-based reservation views, making it easier to navigate overlapping bookings
* Add real-time conflict warnings if multiple users attempt to book the same item
* Advanced Search & Filtering: Enable users to filter equipment or reservations by status, category, and more
* Mobile App Version: Phone app compatibility for booking and notifications on the go

These improvements would further enhance the reliability and scope of the Lab Equipment Booking system.

# Conclusion

Building the Lab Equipment Booking system was a rewarding experience that taught us the importance of database design. As a team, we strengthened our full-stack development skills using a tech stack of PostgreSQL, Flask and React. Working on features such as availability validation and notification systems enhanced our understanding of complex SQL operations and RESTful API design.

For this project, we learned a lot about usability, validating inputs across systems, and managing relational data in complex workflows. Overall, we are proud and confident that this system has a strong foundation for future development. The skills and insights we gained will stay with us as we continue in our academic and professional journeys.