MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
*(A constituent unit of MAHE, Manipal)*

# CSE 3243 Web Programming Lab

# Mini Project Report on

# "CampusSpaces - Booking spaces and equipment in your campus made easy"

**SUBMITTED BY**


| | |
|---|---|
| Pranamya G Kulal | 220905018 |
| Aditya Amit Kinjawadekar | 220905006 |
| Ayush Prabhu | 220905010 |
| Havish Shetty | 220905233 |
| Areen Chakraborty | 220905496 |


## Section A


Under the Guidance of:
Mr. Manamohana K., Dr. Vidya K. and Dr. Neelima B.
Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal, Karnataka – 576104


2024-25

## Acknowledgement

We would like to express our deepest gratitude to everyone who contributed to the successful development of *Campus Spaces*—a platform designed to simplify the booking of classrooms, lecture halls, seminar halls, and equipment on campus.

First and foremost, we extend our heartfelt appreciation to our project guides, Prof. Manamohana Krishna, Prof. Vidya Kamath and Prof. Neelima B. for their invaluable support, insightful guidance, and constant encouragement throughout the project. Their expertise and mentorship played a crucial role in shaping our work.

We also wish to thank our institution, faculty members, and peers for providing us with the necessary resources, feedback, and motivation to complete this project successfully.

A special note of appreciation to the security guards and the student council, whose cooperation and support helped us better understand the logistics and real-world challenges of managing campus spaces efficiently. Their insights and assistance were instrumental in refining our platform.

Finally, we acknowledge the hard work, dedication, and collaboration of our entire project team. This project would not have been possible without the collective effort and determination of each team member.

Acknowledged by:

Pranamya G Kulal
Aditya Amit Kinjawadekar
Ayush Prabhu
Havish Shetty
Areen Chakraborty

## Abstract

The Campus Spaces Project is a web-based room and equipment booking system designed to streamline resource management at the Manipal Academy of Higher Education (MAHE). The previous manual process led to inefficiencies such as double bookings, approval delays, and miscommunication among stakeholders. The system automates the booking workflow to address these issues, integrating automated approvals, conflict resolution, and real-time notifications for all relevant personnel.

The project is implemented using Django (backend), PostgreSQL (database), and Bootstrap (frontend) to provide a scalable, secure, and user-friendly experience. Key functionalities include role-based access control, hourly slot-based reservations, automatic acceptance of nonconflicting bookings, and faculty/admin-based arbitration for disputes. Additionally, HR personnel receive automated notifications when rooms or equipment under their supervision are booked.

The development followed an iterative methodology, including database normalisation (BCNF compliance), an efficient booking approval workflow, rigorous unit, integration, system, and performance testing to ensure system reliability. The system successfully enhances efficiency, reduces manual workload, and improves transparency in campus resource management. Future enhancements may include recurring bookings, advanced analytics, and integration with MAHE's SLCM/Outlook system for seamless user management.

## Table of Contents

# 1. Introduction

The Campus Spaces Project is designed to streamline and automate the room and equipment booking system at Manipal Academy of Higher Education (MAHE). Previously, the booking process was handled manually, leading to inefficiencies such as long wait times, double bookings, approval delays, and miscommunication. The student/faculty was supposed to get approval from the student welfare department and student council separately and provide proof of approval to the person in charge of the resource beforehand. This system automates the processes and improves efficiency.

This system provides a centralised platform where students, faculty, and administrators can manage bookings efficiently. Key features include automated approval workflows, conflict resolution based on predefined rules, and real-time notifications for relevant stakeholders, including non-teaching personnel responsible for maintaining the booked spaces and equipment.

By implementing a modern web-based solution using Django (backend), PostgreSQL (database), and Bootstrap (frontend), the project ensures an efficient, transparent, and userfriendly booking experience.

## 2. System Analysis

The Campus Spaces Project aims to address inefficiencies in room and equipment booking at the Manipal Academy of Higher Education (MAHE). The primary issues being solved include:

- Inefficient manual booking: Previously, room bookings were handled manually, leading to errors and double bookings.
- Automated approval process: Faculty members now handle approvals through an automated system based on student club requests.
- Conflict resolution: If booking conflicts arise, arbitration is conducted based on faculty seniority in the organisation. In case this does not resolve the issue, the admin has the privilege to resolve the conflict.
- Automated notifications: HR personnel (e.g., technicians, cleaners) are automatically notified when a resource under their responsibility is booked.

### 2.1 Functional Requirements

The system is designed to fulfil the following must-have functionalities:

- User Roles & Permissions:

  Students: Club representatives can request room bookings but cannot approve them.
  Faculty: Approves/rejects booking requests for their respective clubs and can book rooms/equipment.
  Admin: Manages users, oversees the system, and resolves booking conflicts

- Room & Equipment Booking:

  Users can book classrooms, lecture halls, seminar rooms, and labs.
  Equipment (e.g., projectors, PCs, microphones) may be associated with rooms or available centrally.
  Bookings follow a fixed hourly slot system.

- Approval & Conflict Resolution Workflow:

  Faculty approves bookings for their respective clubs.
  Admin arbitrates in case of booking conflicts, following predefined rules. Automatic acceptance occurs if no conflicts exist.

- Notifications System:

  HR personnel receive email notifications when resources they manage are booked.
  Users receive notifications only if their booking is cancelled due to a conflict.

- Human Resource (HR) Management: HR personnel do not directly interact with the system but receive notifications when assigned tasks.

## 2.2 Non-Functional Requirements

- Scalability: The system should handle a growing number of users and bookings without performance issues.
- Security: Only authenticated users can access the system, with role-based permissions enforced.
- Usability: The system should have an intuitive UI for seamless room and equipment booking.
- Performance: Booking processing, approvals, and notifications should happen in realtime.

## 2.3 Data Flow Diagrams

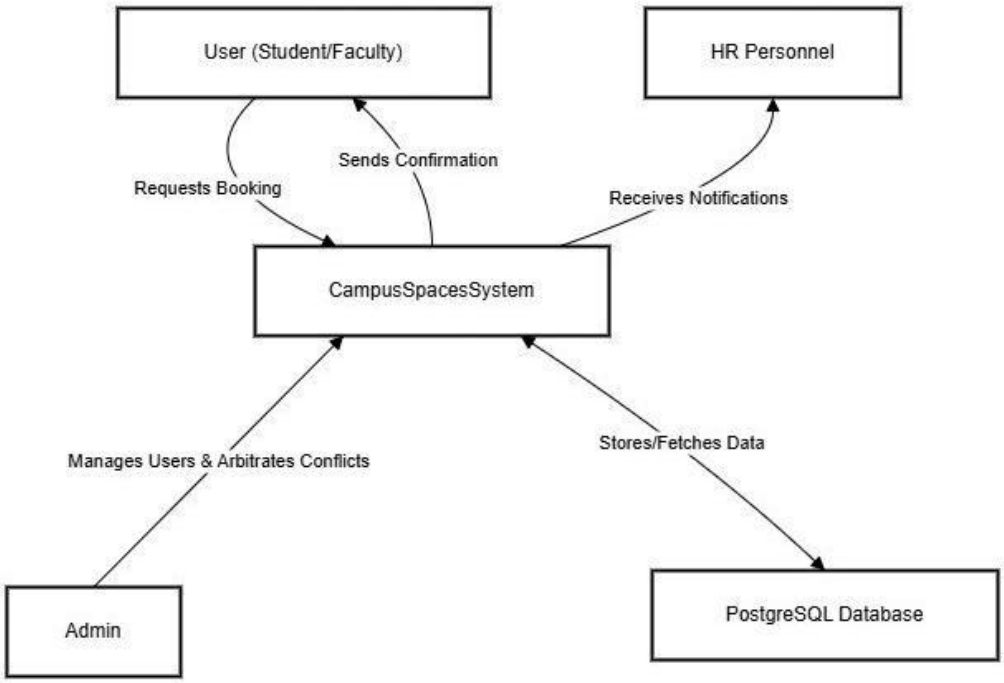


Fig 2.3.1 Level 0 - Context Diagram

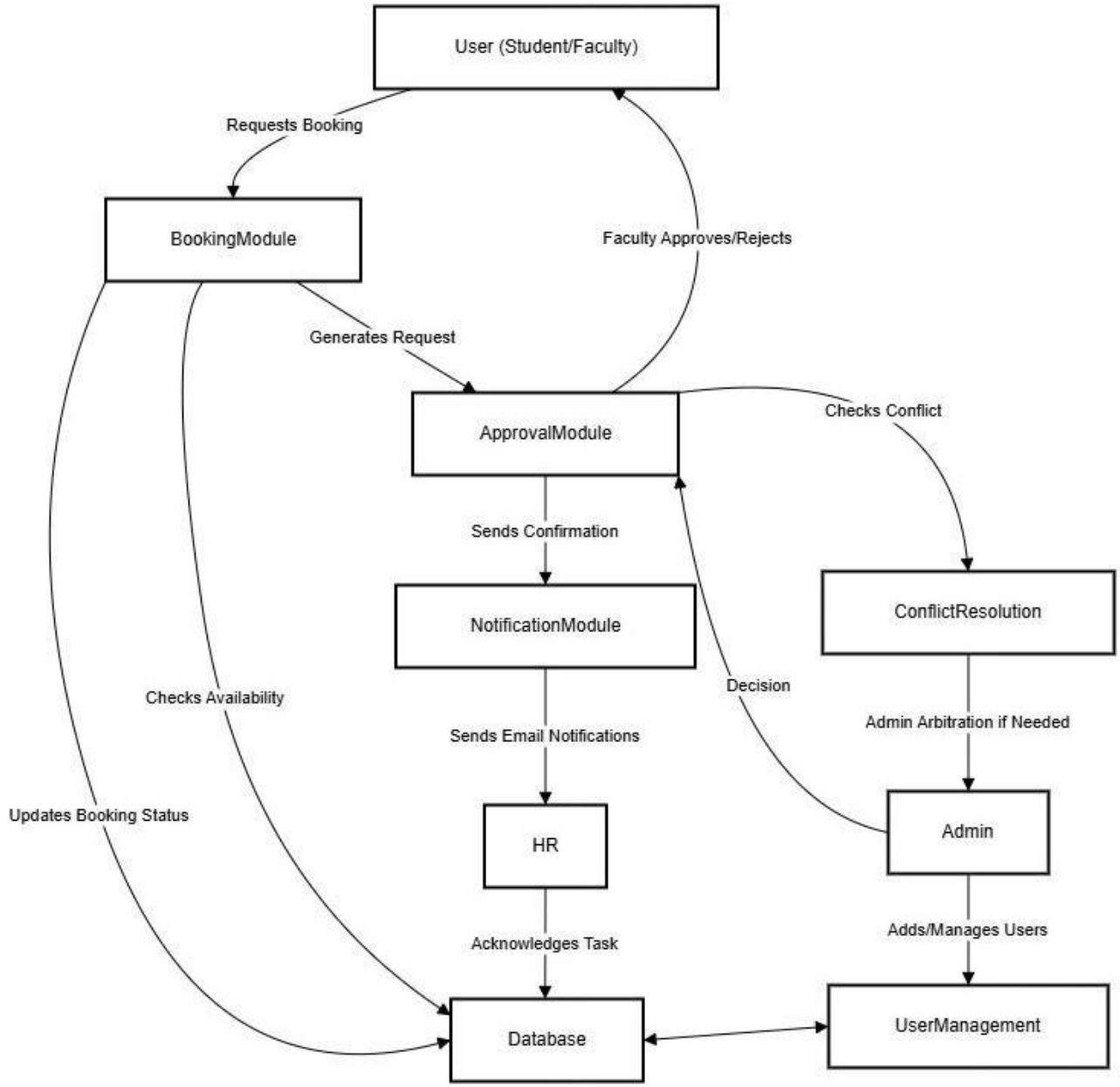


Fig 2.3.2 Level 1 DFD: Module-level representation of the system.

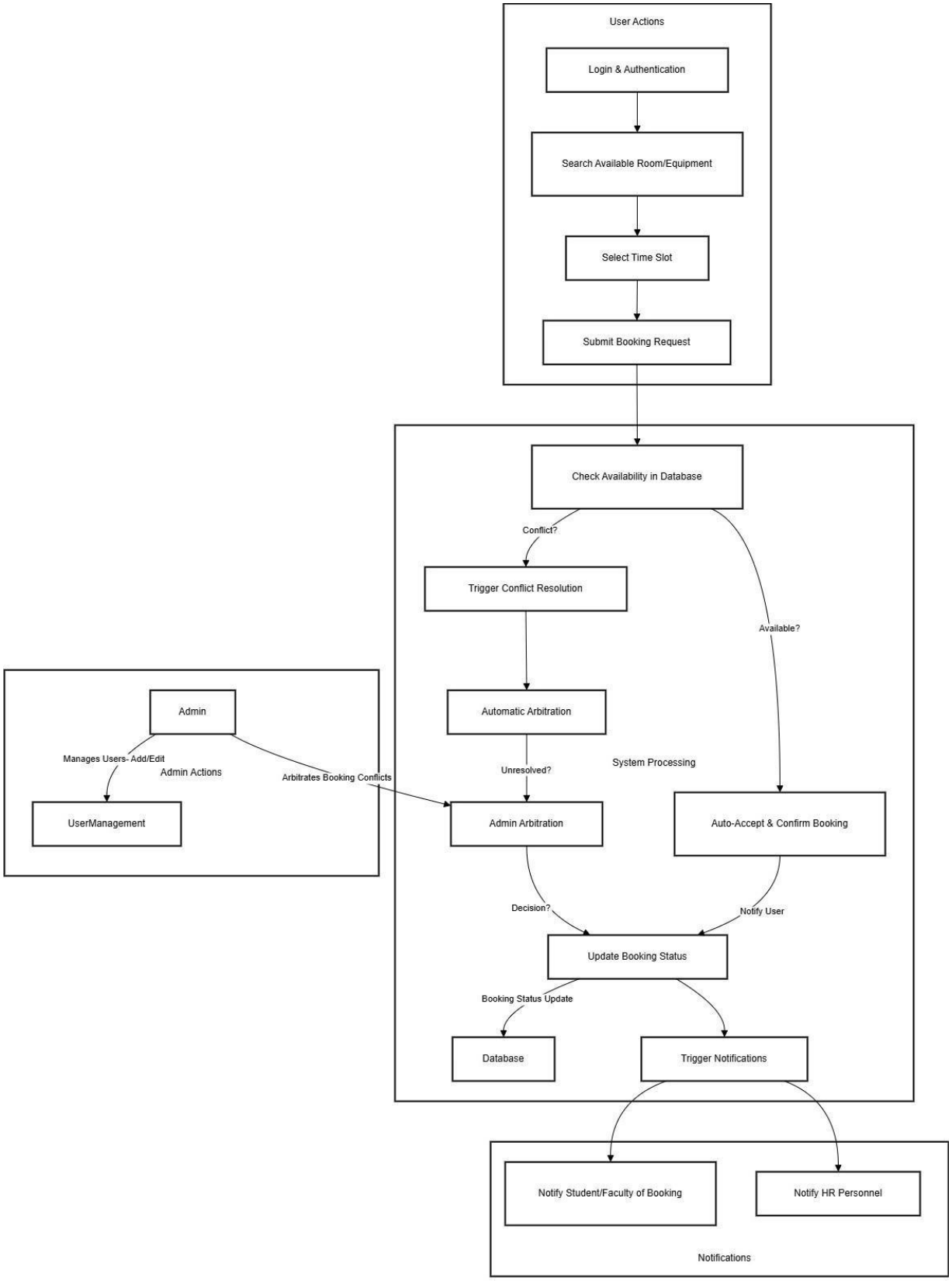


Fig 2.3.3 Level 2 DFD: Refactored DFD to represent the internal functioning of each module

# 3. System Design

## 3.1 System Architecture

The system is built on a web-based architecture with the following components:

Frontend (User Interface)

- Built using Bootstrap for a responsive design.
- Key pages include:

  Login Page: For user authentication.

  Dashboard: Allows users to search and book rooms/equipment.

  Booking Confirmation Page: Displays booking details.

  Admin Panel: Enables admins to manage users and resolve conflicts.

Backend (Business Logic & API)

- Developed using Django, providing a robust backend framework.
- Implements Django ORM for database interactions.
- Handles:

  Booking logic (checking availability, creating bookings).

  Conflict resolution (faculty arbitration, admin intervention).

  Email notifications (booking confirmations, HR task notifications).
- Authentication: Uses Django's built-in authentication system with session management.

Database (Storage & Data Management)

- Database (Storage & Data Management)
  The database is designed following the BCNF (Boyce-Codd Normal Form) to ensure minimal redundancy and maintain data integrity. It is implemented using PostgreSQL for efficient data storage, retrieval, and support for complex queries, particularly for managing bookings and resource allocation.

Database Schema

- Users Table
  Stores user details such as user ID, username, email, password, and their associated role.
  Ensures authentication and authorization for different functionalities within the system.
- User Profile Table
  Extends the Users table to store additional user details such as role, phone number, and organization memberships.
  Supports many-to-many relationships between users and organizations.
- Organization Table
  Stores details of various organizations (e.g., clubs, departments, administration).
  Fields include organization ID, name, type, and description.

- User-Organization Table
  Manages the many-to-many relationship between users and organizations.
  Contains user ID, organization ID, and the hierarchy level (e.g., Member, Representative, Leader).
- Building Table
  Stores building-related information, including building ID, name, and location.
  A building can contain multiple rooms and resources.
- Rooms Table
  Stores details of various rooms within buildings, including room ID, name, building ID, room type, capacity, floor number, and room number.
  Used for room booking and allocation.
- Equipment Table
  Stores details of equipment available in rooms.
  Fields include equipment ID, name, equipment type, model number, serial number, and associated room ID.
- HR Personnel Table
  Stores details about HR personnel responsible for managing resources.
  Fields include HR ID, name, phone, email, and HR type (e.g., Technician, Cleaner, Security).
- Resources Table
  Represents abstract resources such as rooms and equipment.
  Fields include resource ID, name, description, status, associated HR personnel, and building ID.
- Bookings Table
  Stores booking records for rooms and equipment.
  Fields include booking ID, user ID, organization ID, room ID, title, description, start time, end time, status, timestamps, approved by, and rejection reason.
- Booking Equipment Table
  Manages many-to-many relationships between bookings and equipment.
  Fields include booking ID and equipment ID to track which equipment is associated with which booking.
- Notifications Table
  Stores system notifications related to bookings and HR tasks.
  Fields include notification ID, user ID, title, message, notification type, timestamp, read status, and related booking ID.

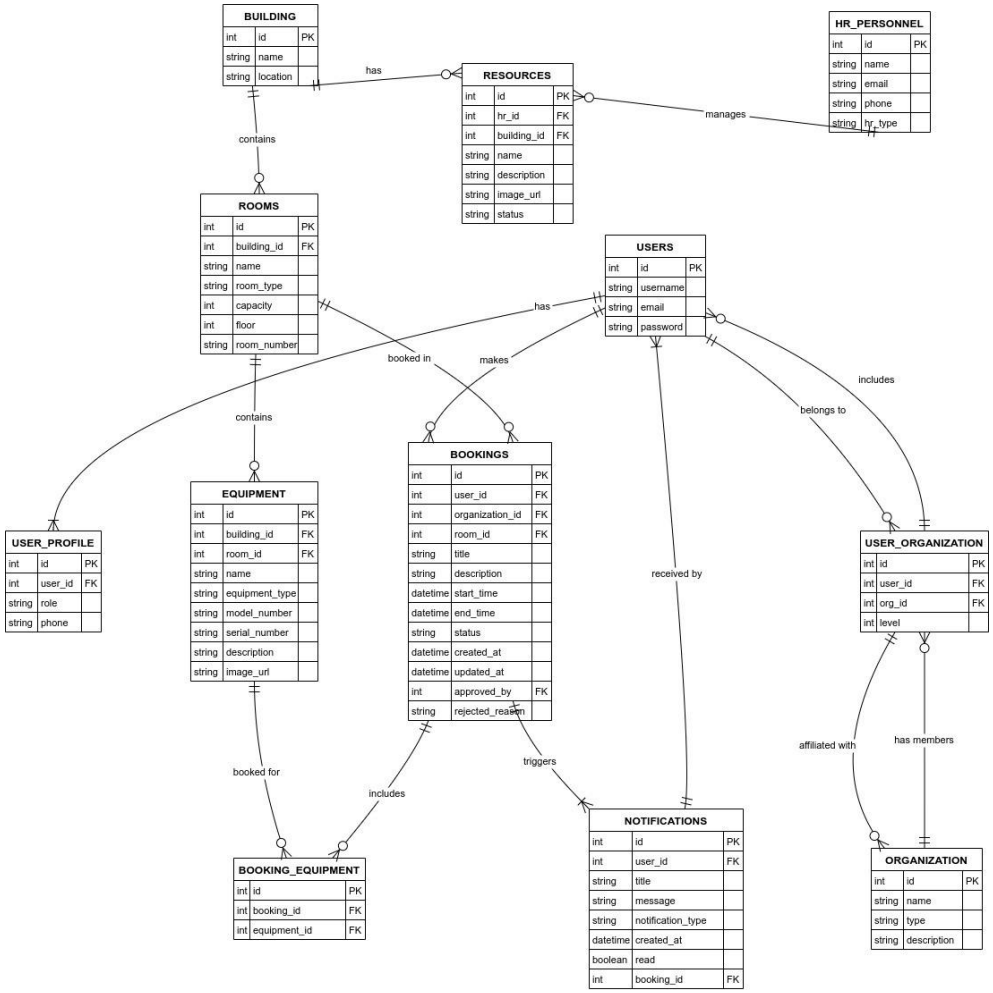


Fig 3.1 ER Diagram

## 3.2 Booking Process Flow

The step-by-step flow of the system is as follows:

1. User Authentication: Users log in via Django authentication (with session management).
2. Room & Equipment Search: Users can filter available rooms/equipment by type, capacity, and slot availability.
3. Booking Request: A user selects a resource and applies for a booking.
4. Conflict Detection (T-24 hours):

- If no conflict exists, the booking is automatically accepted.
- If a conflict arises, the priority decides. If unresolved, the admin intervenes.

5. Approval & Notification:

- Faculty or admin approves/rejects bookings.
- Users and HR personnel receive email notifications.

## 4. Implementation Steps

Phase 1: Setup & Database Design

- Initialize Django project and set up PostgreSQL.
- Define database schema using Django ORM.

Phase 2: Authentication & User Management

- Implement role-based access control.
- Enable authentication for secure session handling.

Phase 3: Booking System Development

- Develop room and equipment booking logic with conflict detection.
- Implement automatic approval and admin arbitration.

Phase 4: Notifications & Admin Panel

- Configure email notifications for bookings and HR personnel.
- Build an admin dashboard for conflict management and user oversight.

Phase 5: Frontend Development & Integration

- Develop Bootstrap-based UI.
- Implement search and booking features.

Phase 6: Testing & Deployment

- Conduct unit testing and integration testing.
- Deploy the system to a production environment.

## 5. Testing Methodology

The testing process involved multiple stages, each addressing specific aspects of system functionality:

### 5.1 Unit Testing

Unit tests were conducted to verify the behaviour of individual modules, such as authentication, booking logic, conflict detection, and email notifications. This phase ensured that each component operated correctly in isolation before integration. The following unit tests were performed:

- User Authentication Test: Ensured that login, logout, and password recovery mechanisms function correctly.
- Booking Logic Test: Verified that users could book available rooms and equipment without errors.
- Conflict Detection Test: Checked that the system correctly identifies and flags double bookings.
- Email Notification Test: Confirmed that notifications are sent to users and HR personnel as expected.

### 5.2 Integration Testing

Integration tests were performed to assess the interaction between different modules. The objective was to ensure seamless data flow and proper execution of processes, such as the transition from booking submission to faculty approval and conflict resolution. The following tests were conducted:

- User Role Validation Test: Ensured that students, faculty, and admin users have appropriate permissions and cannot perform unauthorised actions.
- Approval Workflow Test: Verified that booking requests are routed correctly to faculty for approval and that admin intervention works in conflict resolution cases.
- Booking Confirmation and Notification Test: Ensured that successful bookings trigger appropriate email notifications and updates in the system.
- Database Consistency Test: Checked that bookings, approvals, and cancellations are correctly reflected in the database.

### 5.3 System Testing

The entire system was tested to validate that it meets the specified requirements. Various scenarios were simulated, including successful bookings, booking conflicts, cancellations, and faculty intervention in case of disputes. Multiple scenarios were simulated, including:

- End-to-End Booking Process Test: Verified that users can log in, search for rooms/equipment, book resources and receive approval.

- Conflict Resolution Test: Ensured the system correctly enforces priority rules when resolving booking conflicts.
- Admin Intervention Test: Tested the admin's ability to manually resolve disputes and override approvals when necessary.

## 5. 4 Performance Testing

The system's ability to handle multiple simultaneous requests was evaluated. Load testing assessed response times and system stability under high user activity.

## 5.5 Security Testing

Basic security tests were conducted to check for vulnerabilities such as SQL injection, crosssite scripting (XSS), and unauthorised access attempts. These tests were essential to ensure data protection and system integrity.

The testing process confirmed that the system functions as expected, with all critical workflows operating correctly. The system successfully processed room and equipment bookings without errors, and the conflict resolution mechanism performed effectively in cases of overlapping requests.

The performance tests demonstrated that the system could handle many concurrent users without degradation. Security testing identified no significant vulnerabilities, ensuring that user data remains protected.

## 6. Conclusion and Future Enhancements

The Campus Spaces Project successfully streamlines the room and equipment booking process at Manipal Academy of Higher Education (MAHE). By automating approvals, enhancing conflict resolution, and notifying relevant personnel, the system significantly reduces manual errors, double bookings, and inefficiencies. Adopting a Django-based backend, PostgreSQL database, and a user-friendly Bootstrap frontend ensures a scalable, maintainable, and intuitive solution.

Moving forward, the system will continue to be monitored for usability, performance, and scalability, ensuring it meets evolving needs while providing a robust and reliable solution for campus resource management.

Several improvements can be implemented in future iterations of the system to enhance functionality and user experience:

1. Recurring Bookings: Implement a weekly or monthly recurring booking feature, allowing clubs and faculty to reserve rooms on a regular schedule.
2. Reports and Analytics: Generate detailed insights into room usage trends, peak booking hours, and equipment demand to optimise space allocation.
3. Feedback Mechanism: Allow users to rate rooms and equipment, helping administrators track maintenance needs and improve resource quality.
4. Integration with SLCM/Outlook to automate user creation and priority labelling.

# 7. Screenshots and Output

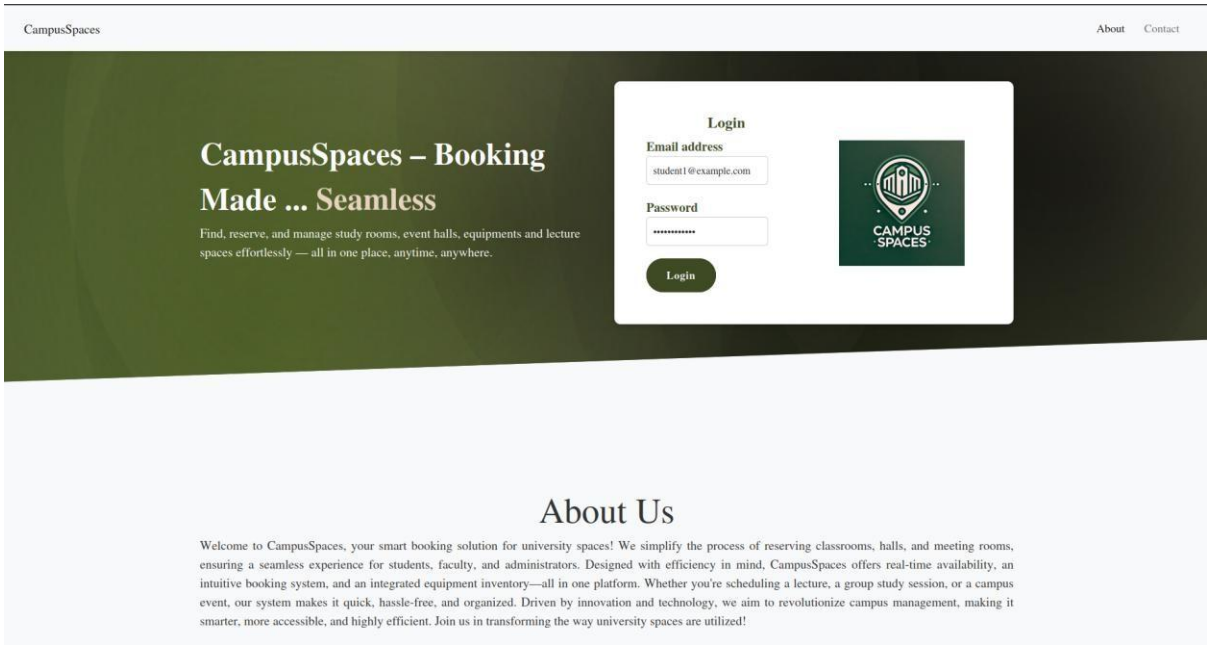


Fig 7.1 Home Page

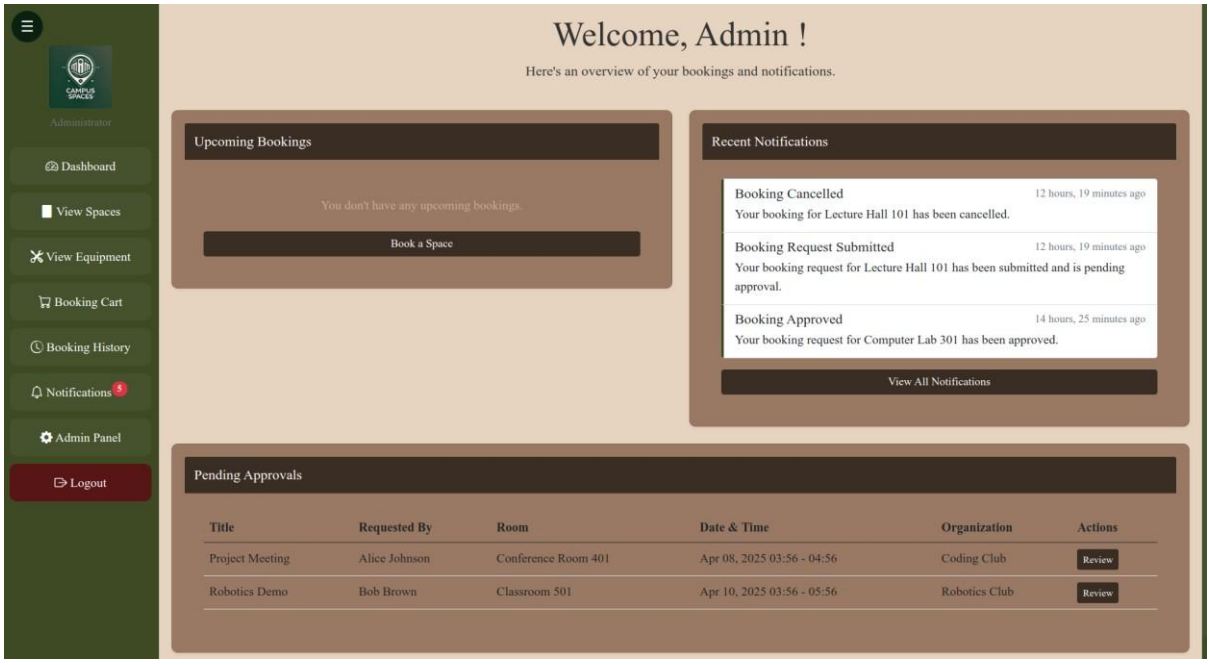


Fig 7.2 Dashboard (Admin View)

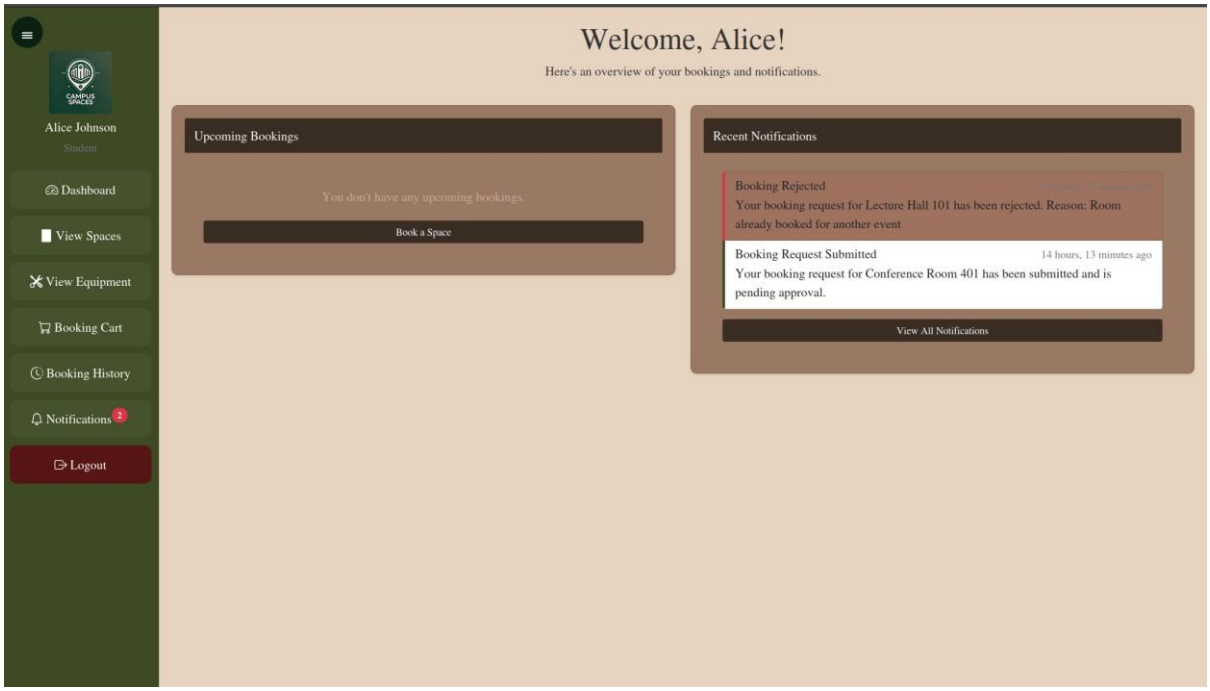


Fig 7.3 Dashboard (Non Admin)

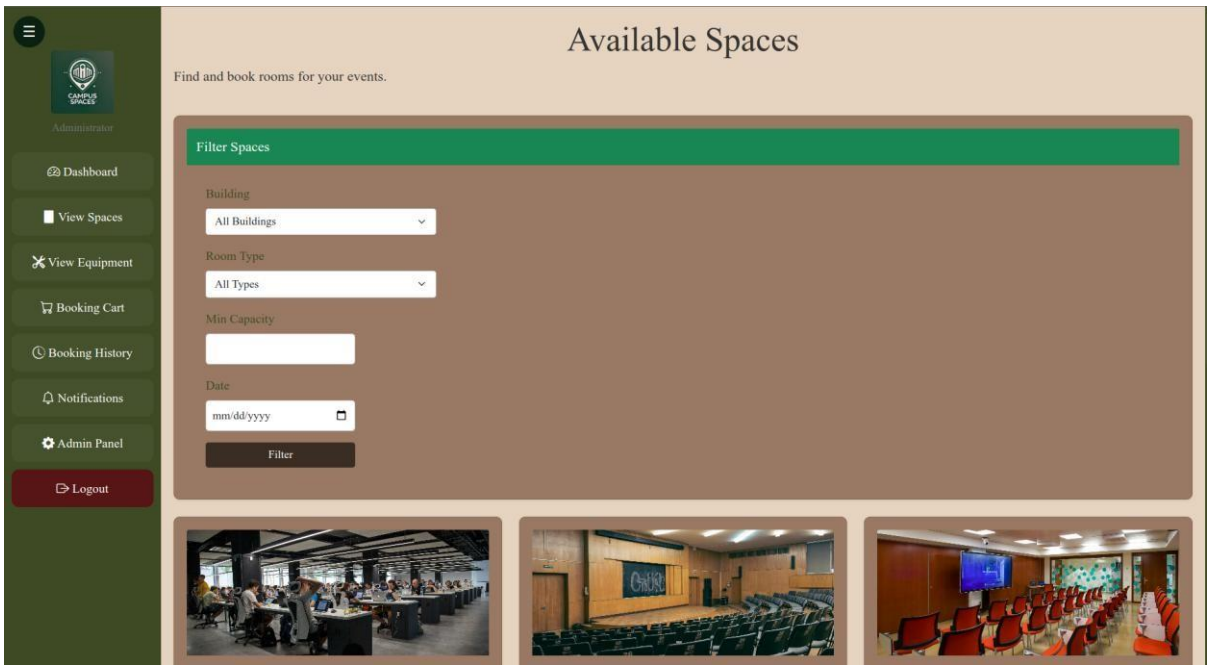


Fig 7.4 View Spaces

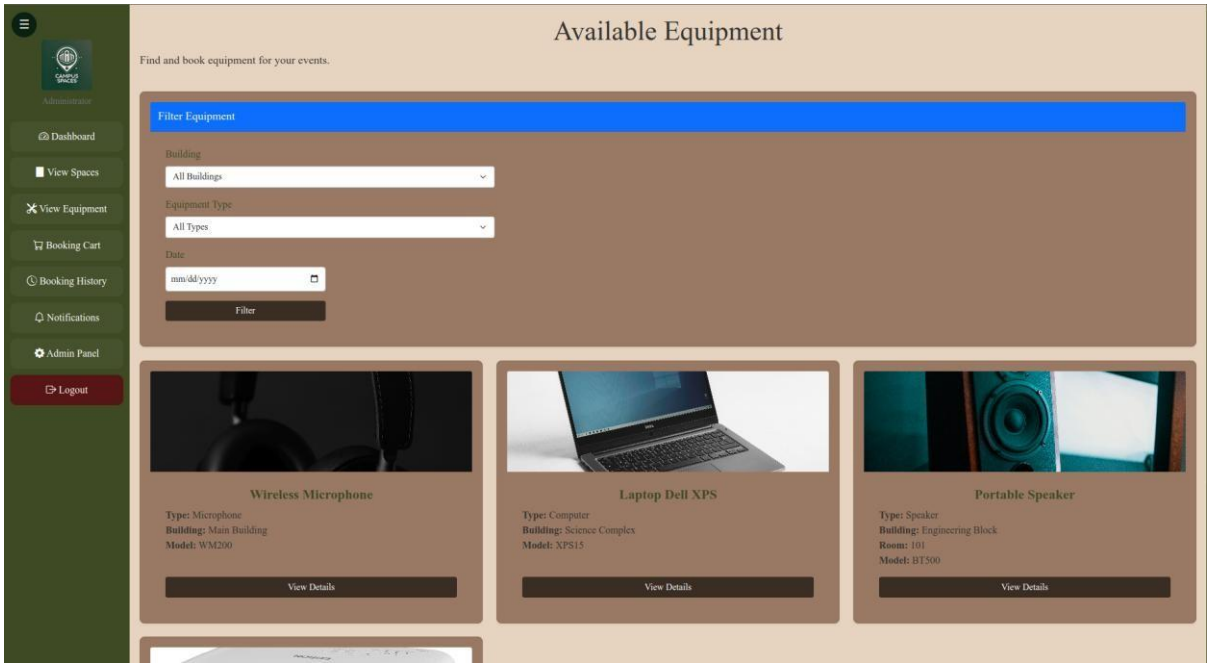


Fig 7.5 View Equipment

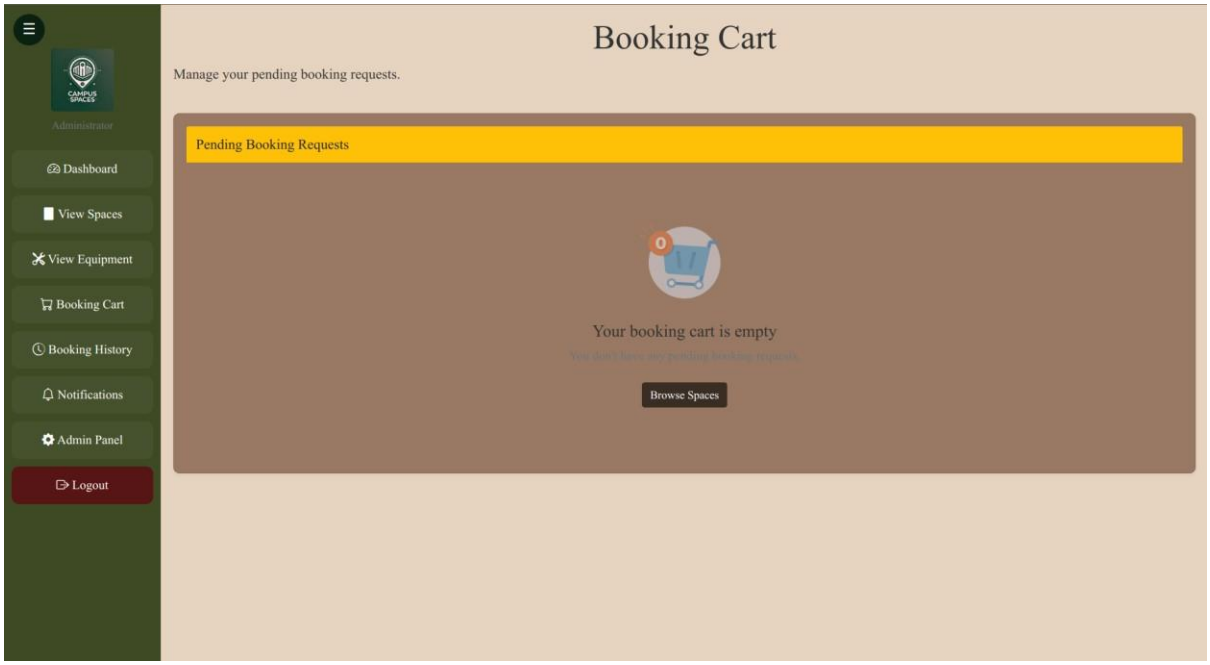


Fig 7.6 Booking Cart

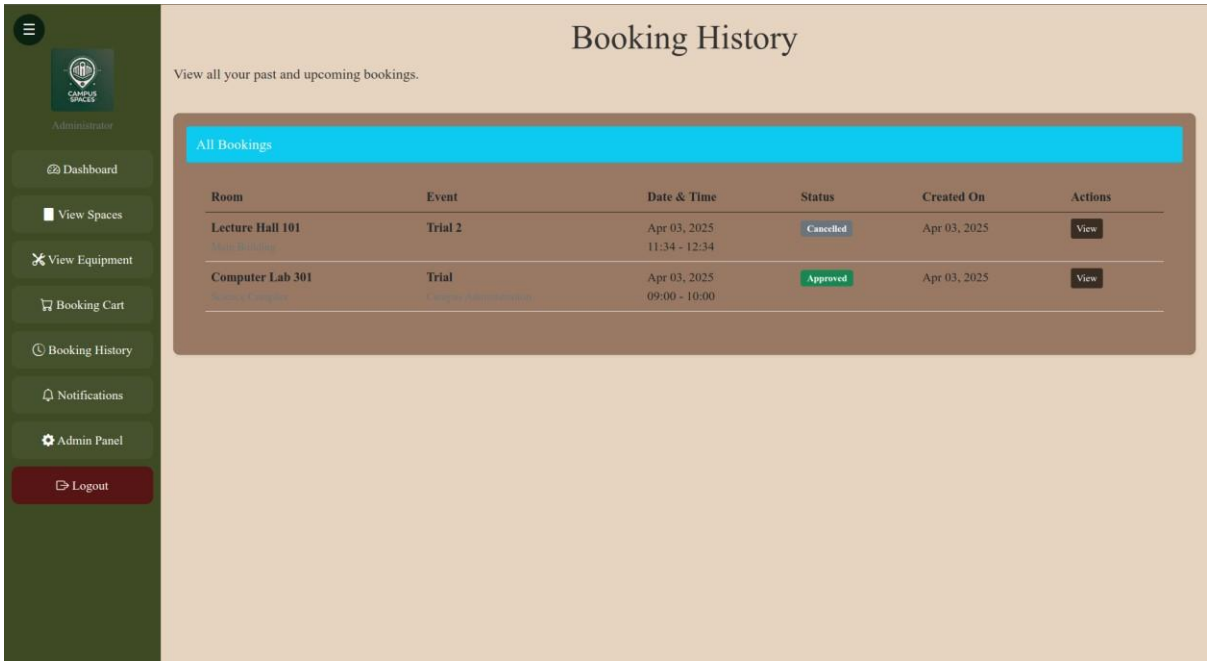


Fig 7.7 Booking History

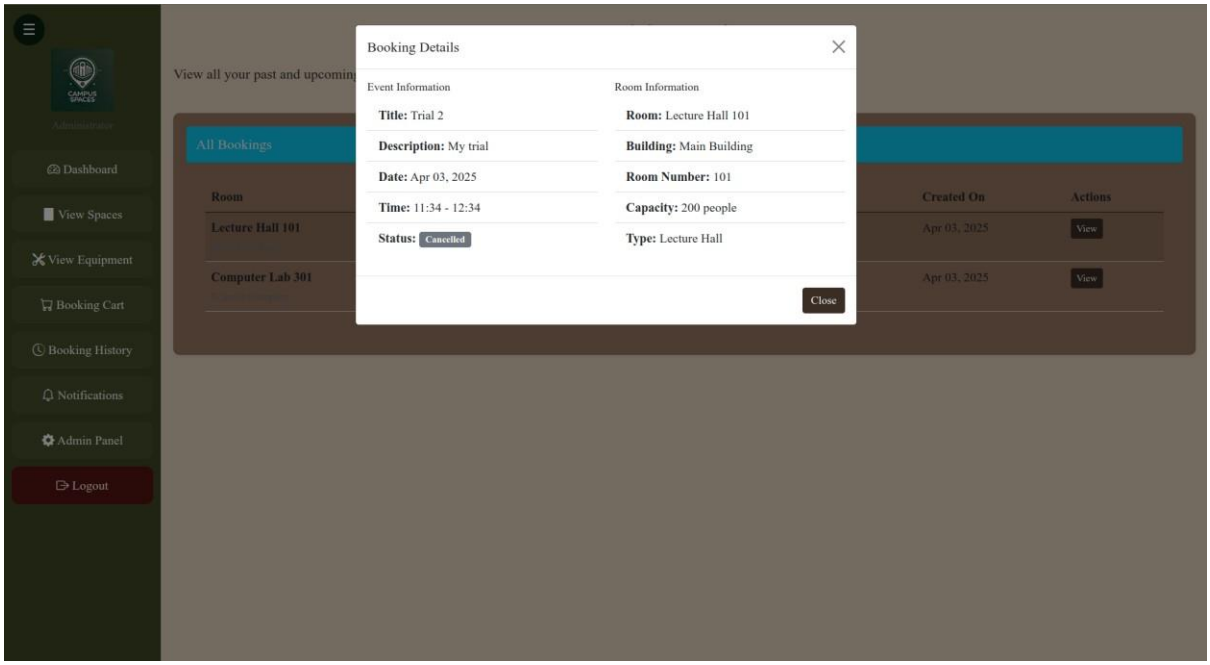


Fig 7.8 Booking Details View

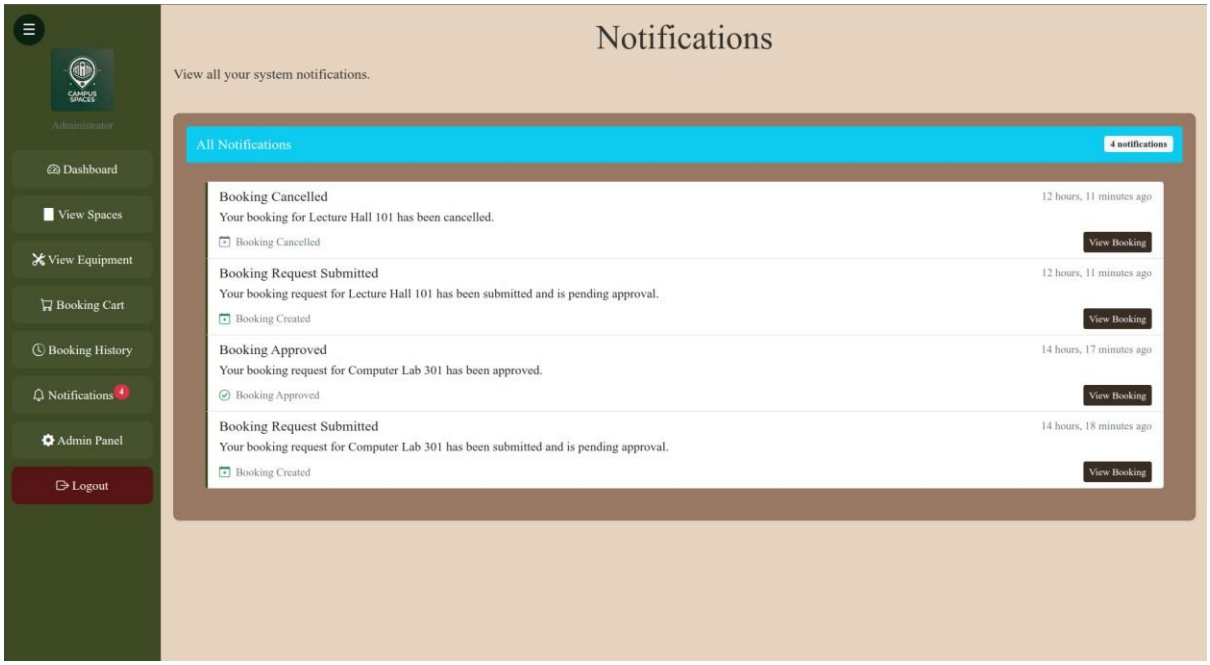


Fig 7.9 Notifications Page

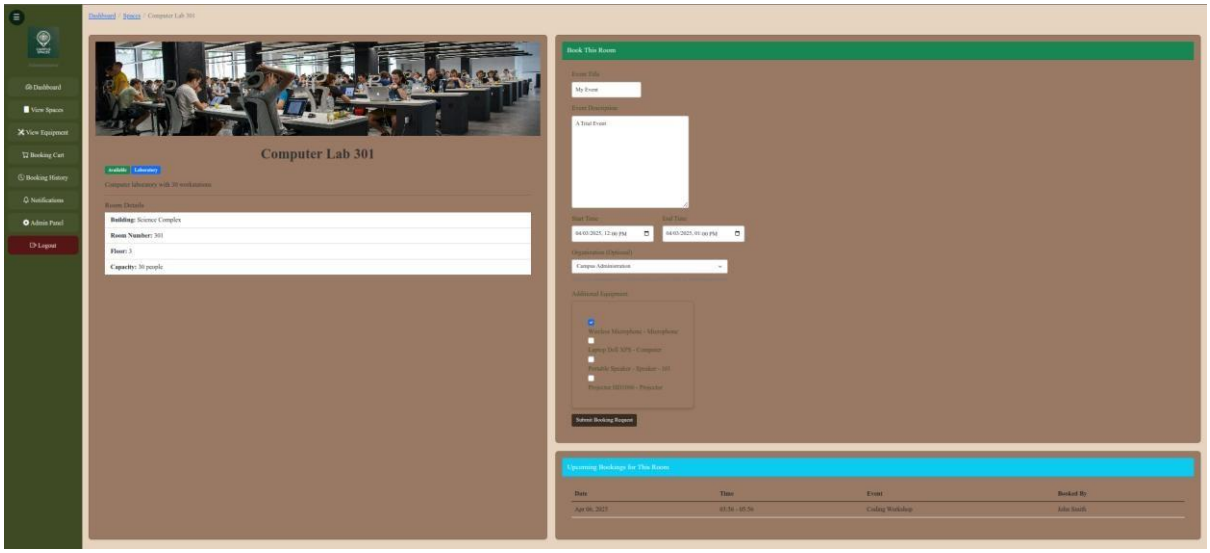


Fig 7.10 Room Booking Page

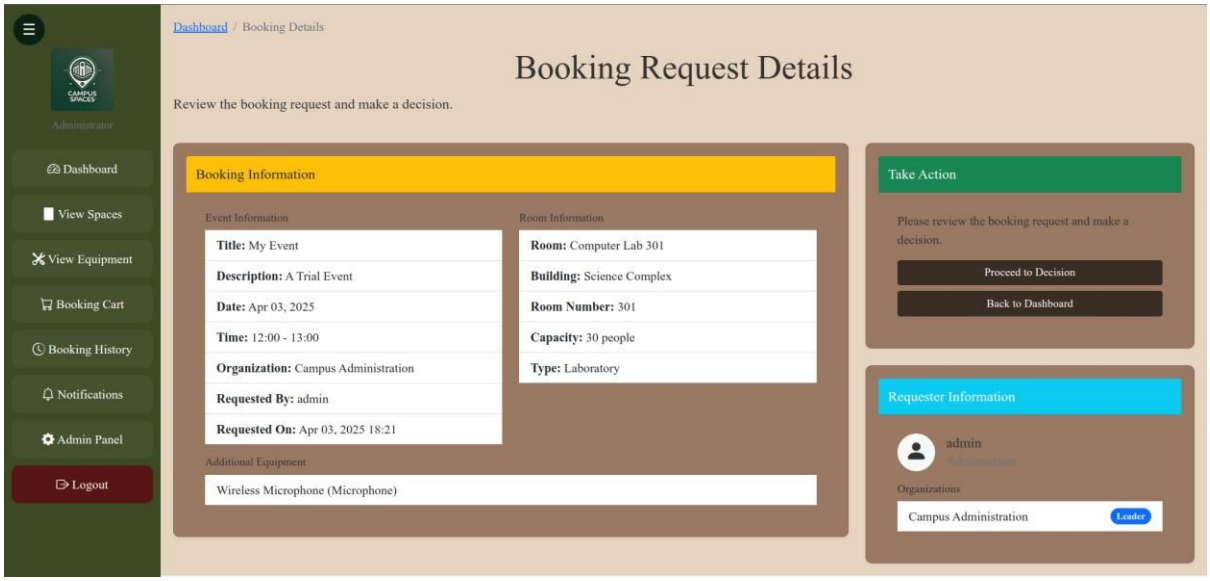


Fig 7.11 Booking Review Page (Admin)

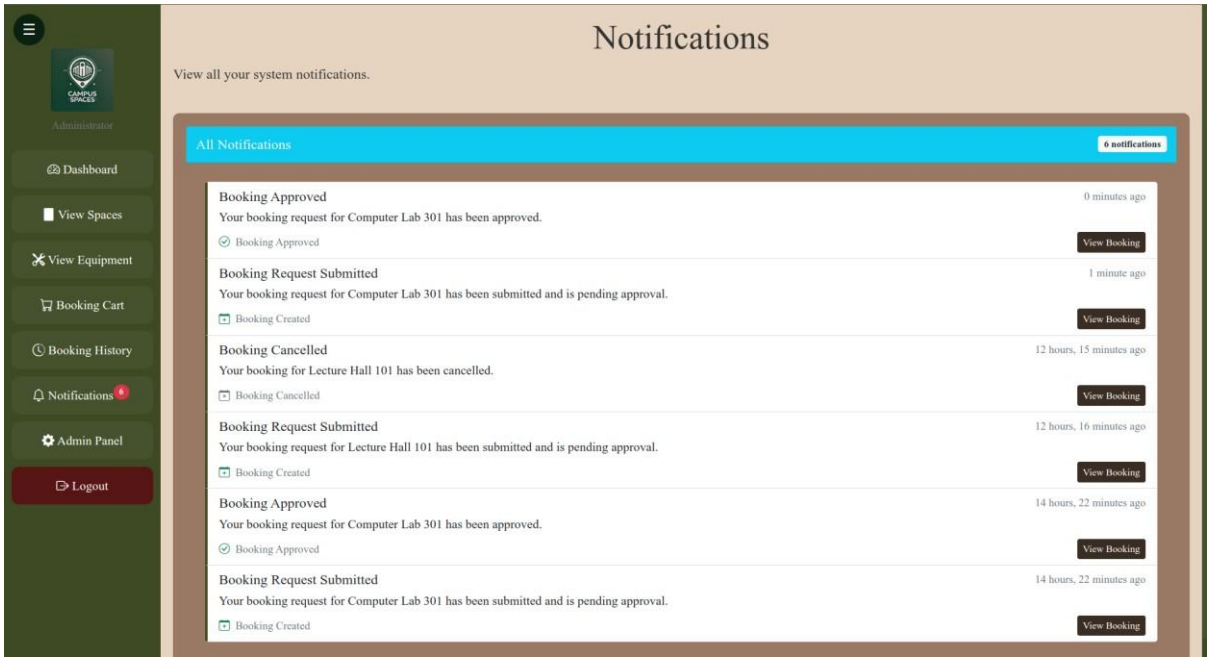


Fig 7.12 Updated Notifications Page

## 8. References

[1] “Normalization,” *Griffith.edu.au*, 2015. https://www.ict.griffith.edu.au/normalization_tools/normalization/ind.php (accessed Feb. 02, 2025).

[2] “Django Tutorial,” www.w3schools.com. https://www.w3schools.com/django/index.php (accessed Feb. 08, 2025).

[3] “PostgreSQL: Documentation,” www.postgresql.org. https://www.postgresql.org/docs/ (accessed Mar. 15, 2025).