Final Year Project

Opti-Reserve

Project Team

Om -2280156

Gulzar Hussain -2280176

Project Supervisor

Miss Sadia Aziz

Date: JULY 2nd 2026

Submitted partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science

in the

Faculty of Computing and Engineering Sciences

Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology University

SZABIST University Karachi

Table of Content

Project Proposal 1
1. Introduction 2
2. Objective 2
3. Problem Description 2
4. Target Industry 2
5. Methodology 3
6. Project Scope 3
7. Feasibility Study 3
8. Correlation with the Industry practices and Trends 3
9. Emerging Technologies Used 4
9.1 Hardware 4
9.2 Software 4
10. Expertise of the Team Members 4
11. Milestones 5
12. Project Schedule 6
13. Work Breakdown Structure 9
13.1 FYP-1 9
13.2 FYP-2 10
14. References 10
Software Requirement Specifications 11
1. Introduction 12
1.1. Purpose 12
1.2. Document Conventions 12
2. Overall Description 14
2.1. Product Perspective 14
2.2. Product Functions 14
2.3. User Classes and Characteristics 15
2.3.1. Administrator 15
2.3.2. Faculty Member 15
2.3.3. Student / Public User 16
2.4. Operating Environment 16
2.4.1. Hardware Platform 16

2.4.2. Operating System 16
2.4.3. Software Components 16
2.5. Design and Implementation Constraints 16
2.6. User Documentation 17
2.6.1. Faculty, User Manual 17
2.6.2. Online Help Section 17
2.7. Assumptions and Dependencies 17
3. External Interface Requirements 18
3.1. User Interfaces 18
3.2. Sign Up 18
3.3. Login 19
3.4. Forget Password 19
3.5. About 20
3.6. Admin Board 20
3.7. Users 21
3.8. Schedules 21
3.9. Account Setting 22
3.10. Hardware Interfaces 22
3.11. Software Interfaces 22
3.11.1. Backend 22
3.11.2. Database 22
3.11.3. Email Services: 22
3.11.4. Version Control 22
3.11.5. Communication Flow 23
3.12. Communications Interfaces 23
4. System Features 23
4.1. User Authentication and User Management 23
4.1.1. User Registration 23
4.2. User Login 24
4.2.1. Description 24
4.3. Forgotten Password 26
4.3.1. Description 26
4.4. Account Activation 27
4.4.1. Description 27

4.5. View Available Classroom 28
4.5.1. Description 28
4.5.2. Stimulus/Response Sequences 29
4.5.2.1. Functional Requirements 29
4.6. Admin Management System 29
4.7. User Management 29
4.7.1. Description 29
4.7.2. Stimulus/Response Sequences 29
4.7.2.1. Functional Requirements 30
4.8. Session Management 30
4.8.1. Description 30
4.8.2. Stimulus/Response Sequences 31
4.8.2.1. Functional Requirements 31
4.9. Analytics and Reporting 32
4.10. System Analytics 32
4.10.1. Description 32
4.10.2. Stimulus/Response Sequences 32
4.10.2.1. Functional Requirements 32
4.10.3. Facility Booking Management 33
4.10.4. Stimulus/Response Sequences 33
4.10.4.1. Functional Requirements 33
5. Other Nonfunctional Requirements 34
5.1. Performance Requirements 34
5.2. Safety Requirements 34
5.3. Security Requirements 34
5.4. Software Quality Attributes 34
5.5. Business Rules 34
5.6. Business Model Canvas 35
6. Other Requirements 35
Appendix A: Glossary 36
Appendix B: Analysis Models 37
Appendix C: To Be Determined List 38
Software Design Specification 39
1. Introduction 40

1.1 Purpose of this Document 40
1.2 Scope of the development project 40
1.3 Definitions, Acronyms, and Abbreviations 40
1.4 References 41
1.5 Overview of this Document 41
2. System Architecture Description 41
2.1 Section Overview 41
2.2 General Constraints 41
2.3 Data Design 43
2.4 Program Structure 43
2.5 Alternatives Considered 44
2.5.1 Manual / Spreadsheet-Based Scheduling 44
2.5.2 Microservices Architecture 44
2.5.3 Layered MVC Architecture (Chosen) 45
3. Detailed Description of Components 45
3.1 Section Overview 45
3.2 Component and Detail 45
3.2.1 User Registration Component 45
3.2.2 Authentication Component 45
3.2.3 Booking Management Component 46
3.2.4 Conflict Detection Component 46
3.2.5 Check-In / Check-Out Component 46
3.2.6 Maintenance Reporting Component 47
3.2.7 Upload Attachment 47
3.2.8 Resource Catalog 47
3.2.9 Audit Log 47
3.2.10 User Settings / Profile 48
4. User Interface Design 48
4.1 Section Overview 48
4.2 Interface Design Rules 49
4.3 GUI Components 49
4.4 Detailed Description 49
4.4.1 Sign Up 49
4.4.2 Login 50

4.4.3 Forgotten Password 50
4.4.4 About 51
4.4.5 Admin Board 51
4.4.6 Users 52
4.4.7 Schedule 52
4.4.8 User Interfaces 53
4.4.9 Account Setting 53
5. Reuse and Relationship to Other Products 53
6. Design Decision and Tradeoffs 54
7. Pseudocode for Components 55
7.1 User Authentication Logic 55
7.2 Facility Booking Request Logic 55
7.3 Booking Conflict Detection Logic 56
7.4 Booking Approval Logic (Admin) 56
7.5 Check-In / Check-Out Logic 57
7.6 Maintenance Issue Reporting Logic 57
7.7 Open Session Publication Logic 57
7.8 Reporting and Analytics Logic 58
8. Design Diagrams 58
8.1 Class Diagram 58
8.2 Object Diagram 59
8.3 State Chart Diagram 59
8.3.1 Admin State Chart 59
8.3.2 Faculty State Chart 60
8.4 Activity Diagrams 60
8.4.1 Login 60
8.4.2 User Registration & Approval 61
8.4.3 User Login 61
8.4.4 Admin Resource & Category Management 62
8.4.5 Booking Request Creation 62
8.4.6 Booking Approval (Admin) 63
8.4.7 Check-In / Check-Out & No-Show 63
8.4.8 Maintenance Reporting 64
8.4.9 Open Session Publication 64

8.4.10 Public Viewing of Open Sessions 65
8.4.11 Reporting & Analytics 65
8.5 Sequence Diagram 66
8.5.1 Faculty Registration & Approval 66
8.5.2 User Login 66
8.5.3 Booking Request Creation 67
8.5.4 Check-In / Check-Out & No-Show 67
8.5.5 Login 68
8.6 Use Case Diagrams 68
8.7 Component Diagram 69
8.8 Deployment Diagram 70
Test Cases 71
1. Login 72
2. Registration 73
3. Forget-Password 74
4. Profile Management 75
5. Admin Dashboard 75
6. Session Management 76
7. Two Factor Authentication 77
8. Search Functionality 78
9. File Upload Validation 79
10. API Security 80
11. Booking Completion 81
12. Role-Based Access Control 82
13. Admin Panel Real Time Update 82

Table of Figures

Figure 1: FYP 2 Schedule - Gantt Chart 8
Figure 2: FYP 2 Schedule - Gantt Chart 8
Figure 3: FYP 2 Work Breakdown Structure 10
Figure 4: Use case Diagram 14
Figure 5 Screen 18
Figure 6 Sign Up Screen 18
Figure 7 Login Screen 19
Figure 8 Forget Password Screen 19
Figure 9 About Screen 20
Figure 10 Admin Screen 20
Figure 11 Users Screen 21
Figure 12 Schedules Screen 21
Figure 13 Account Setting Screen 22
Figure 14 Business Canvas Model Diagram 35
Figure 15: ERD of FYP 1&2 37
Figure 16: Data Design of FYP 1&2 43
Figure 17 Sign Up Screen 49
Figure 18 Login Screen 50
Figure 19 Forget Password Screen 50
Figure 20 About Screen 51
Figure 21 Admin Board Screen 51
Figure 22 Users Screen 52
Figure 23 Schedule Screen 52
Figure 24 User Profile Screen 52
Figure 25 User Interfaces Screen 53
Figure 26 Account Settings Screen 53
Figure 27: Class Diagram 58
Figure 28: Object Diagram 59
Figure 29: State Diagram 59
Figure 30: Login Activity Diagram 60
Figure 31: User Registration Approval Activity Diagram 61
Figure 32: User login Activity Diagram 61
Figure 33: Admin Resource & Category Activity Diagram 62
Figure 34: Booking Request Activity Diagram 62
Figure 35: Booking Approval Activity Diagram 63
Figure 36: Check In/Check Out Activity Diagram 63
Figure 37: Maintenance Activity Diagram 64
Figure 38: OS Publication Activity Diagram 64
Figure 39: Open session activity Diagram 65
Figure 40: Report Analytics Activity Diagram 65
Figure 41: Faculty registration Sequence Diagram 66
Figure 42: User Login Sequence Diagram 66
Figure 43: Booking Request Sequence Diagram 67
Figure 44: Check In/Check Out Sequence Diagram 67

Figure 45: Login Sequence Diagram 68
Figure 46: Use Case Diagram 68
Figure 47 Component Diagram 69
Figure 48 Deployment Diagram 70

Table of Tables

Table 1:FYP-1 Schedule Table 6
Table 2: FYP-2 Schedule 7
Table 3: Document Convention 12
Table 4 User Registration Stimulus/Response 24
Table 5: User Registration Functional Requirements 24
Table 6: User Login Stimulus/Response 25
Table 7: User Login Functional Requirements 25
Table 8: Forget Password Stimulus Response 26
Table 9: Forget Password Functional Requirement 27
Table 10: Account Activation Stimulus Response 28
Table 11 Account Activation Functional Requirements 28
Table 12: View Available Rooms Stimulus Response 29
Table 13: View available Classroom Functional Requirements 29
Table 14: Manage Users Stimulus Response 30
Table 15: Manage Users Functional Requirements 30
Table 16: Manage Lab Session Stimulus Response 31
Table 17: Management Session Functional Requirements 31
Table 18: View System Analytics Stimulus Response 32
Table 19: View System Analytics Functional Requirements 33
Table 20: Facility Booking Management Stimulus Response 33
Table 21: Facility Booking Management Functional Requirements 33
Table 22: Glossary 36
Table 23:Definitions, Acronyms, and Abbreviations 40
Table 24: User Registration Component Table 45
Table 25: Authentication Component Table 45
Table 26: Booking Component Table 46
Table 27: Conflict Detection Component Table 46
Table 28: Check In/ Check Out Component Table 46
Table 29: Maintenance Reporting Component Table 47
Table 30: Upload Attachment Component Table 47
Table 31: Resource Catalog Component Table 47
Table 32: Audit Log Component Table 48
Table 33: User setting Profile Component Table 48
Table 34 Login Test Case 73
Table 35 User Registration Test Case 74
Table 36 Forget Password Test Case 75
Table 37 Profile Management Test Case 75
Table 38 Admin Dashboard Test Case 76
Table 39 Session Management Test Case 77
Table 40 2 Factor Authentication Test Case 78

Table 41 Search Functionality Test Case 79
Table 42 File Upload Validation Test Case 80
Table 43 API Security Test case 81
Table 44 Booking Completion Test Case 81
Table 45 RBACs Test Case 82
Table 46 Admin Panel Real Time Update Test Case 83

Revision History

Name	Date	Reason For Changes	Version
OM & Gulzar	15-10-25	Filling up the correction in ERD	1.1
OM & Gulzar	16-10-25	Making proper 70% of use cases	1.2
OM & Gulzar	17-10-25	Attacking screenshots of UI	1.3
OM & Gulzar	19-10-25	Discussed FYP-2 features	1.4
OM & Gulzar	21-10-25	Update Document	1.4
OM & Gulzar	28-12-25	Proposal, SRS, SDS, Test Cases Complete	1.5
OM & Gulzar	24-3-26	Updated Admin dashboard	1.6
OM & Gulzar	4-4-26	Email verification added	1.7

Project Proposal

1. Introduction

Opti-Reserve is a smart web-based facility booking system, optimized for reserving shared facilities in a university, university organization or co-working environment. It replaces the inefficiencies of manual bookings like double bookings, tracking and communication availability, and delays. Users can easily access and manage facility bookings with the intuitive interface while the platform provides real-time availability of all facilities to users in a seamless way through reserve slots. Opti- Reserve automates transparencies and makes the reservation process of the facilities easy.

2. Objective

Design and develop an intelligent, user friendly and efficient facility booking system that will allow users to book, manage and monitor the institutional resources in real time. The goal of the system is to reduce the number of scheduling conflicts possible through automatic conflict detection, increase transparency, operational efficiency and maximize the overall use of the resources. It also aims to facilitate the administration and decision making based on its data to manage facilities effectively.

3. Problem Description

Manual facility booking procedures are extremely prone to scheduling issues, double booking and mismanagement of resources altogether. Most of the classrooms, labs, meeting rooms and sports facilities of educational institutions and offices are still reserved by telephone, email or paper. These approaches are not real-time and lack visibility of availability, can lead to miscommunication, and inefficient use of resources. As hybrid learning and working models are adopted, the need for a reliable, and automated facility reservation system is becoming a huge concern. Opti-Reserve's web-based, centralized booking solution is an answer to these issues, delivering real-time availability tracking, automated conflict detection and structured booking workflows. The system uses cutting-edge web technologies to provide increased transparency, efficiency in operations and data-driven insights for optimum use of resources.

4. Target Industry

- **Education:** Educational sectors in universities, colleges and schools.
- **Corporate:** Offices for festivals.
- **Healthcare:** Schedule rooms and equipment in a hospital.
- **Government/Public Sector:** Community centers and event facilities in the Government/Public Sector.

5. Methodology

The project is developed using an Agile software development methodology, which is followed to support an iterative development, continuous feedback and incremental improvement throughout the project life cycle. It is implemented in a modular and maintainable way, following the Model-View-Controller (MVC) architecture pattern, with

the use of Node.js and Express.js for the backend implementation. Prisma ORM is used to interact with a MySQL database with a structured and secure way, to ensure data consistency and scaling. Crypt is used for password hashing, and express sessions are used for the management of sessions. Version Control and Collaborative Development is achieved with the help of GitHub and Visual Studio Code is used as the main development environment.

6. Project Scope

Opti-Reserve system provides one place to reserve facilities and resources, real time conflict detection and database-driven scheduling. It offers role-based access, prioritized workflows for faculty and auto-approval and notification. It features resource usage and no-show analytics, advanced search and multi-criteria filtering, Google Calendar integration and advanced check-in and check-out with auto cancellation. Also, it has built-in maintenance reporting and Open Session Publication functionality, which enable booking to be available in public (for students and other users).

7. Feasibility Study

Technical and operational feasibility of Opti-Reserve system has been extensively studied. A strong database design and automated conflict detection systems can help reduce the risk of recurrence of technical difficulties like booking conflicts. Fall back arrangements are put in place for external services including Google Calendar, email notification services etc. System is capable of storing and retrieving the schedules reliably for their use in optimization and analytics. Controls have been put in place to ensure a fair and an accountable urgent booking override. The project can be developed using common tools and technologies that are used in the web development industry today such as node.js, web frameworks, relational databases, GitHub (version control), cloud-based hosting platforms.

8. Correlation with the Industry practices and Trends

Opti-Reserve's ability to use digital technology to solve the scheduling problem which is traditionally solved using manual and spreadsheet methods makes it a suitable fit for the way in which industrial software is developed today. It contains all the common industry functionality such as role based access control, approval workflow and audit friendly booking records that are typically found in enterprise systems. It adopts a Agile development, modular design, ORM-based database access and MVC architecture, which is in line with professional development practices. While user experience is a top priority, maintainability, seaccommodatedscalability are the operational challenges addressed by Opti-Reserve, which have been adapted to use by numerous fields of education, companies, healthcare and co-working environments.

9. Emerging Technologies Used

The following hardware and software tools/technologies might be needed for the development of the Opti-Reserve platform:

9.1 Hardware: Enough computer or laptop processing power and memory to allow for full stack web development and testing. The application needs to be deployed and run on a server or cloud-based environment. Stable Internet connectivity is crucial for developing, versioning, deployment and integration with external services.

9.2 Software:

- **Integrated Development Environment (IDE):** Visual Studio Code
- **Programming Languages & Frameworks:** Node.js, Express.js
- **Database Management System:** MySQL
- **ORM Tool:** Prisma ORM for database interaction and schema management
- **Templating Engine:** EJS for server-side rendering
- **Authentication & Security:** crypt, express session
- **Version Control:** GitHub
- **Testing Tools:** Web browsers (Chrome, Firefox, Edge)
- **Deployment & Hosting Tools:** Cloud hosting platforms such as AWS, Heroku, or similar services.

10. Expertise of the Team Members

The project team has a solid understanding of the full stack web development ecosystem with Node.js, Express.js and Prisma ORM.

They both have a background with designing relational databases with MySQL and safe session-based authentication with crypt and express session.

The team is also experienced with the MVC architecture, RESTful design principles and collaborative development on GitHub.

11. Milestones

Week 1: Project approval

- Supervisor's approval of project title and project proposal.
- Completion of project goals and objectives.

Weeks 2–4: Initial Planning and Research

- Preside on Project initiating meetings and in-depth requirement sizing up.
- Understand functional and non-functional requirements.
- Develop a plan, schedule and documentation structure for the project.
- Establish technology stack/architectural approach

Weeks 5–7: System & User Interface Design

- Draw system workflows and Use case Diagrams.
- Prioritize and develop wireframes and UI mockups for the Home, Login, Faculty Dashboard, Admin Dashboard, Resource Listing and Booking Interface modules.
- rate basic ER Diagrams & Schema.

Week 8: Mid Term Evaluation

Weeks 9–11: Core Backend Development (Phase I)

- Install Node.js and Express.js project structure (MVC).
- Ensure that users are authenticated and have access based on their roles Admin, Faculty.
- Set up Prisma ORM and MySQL Database connection.

Weeks 12–15: Core Feature Development (FYP-I Completion)

- Implement Resource and Category Management (Create, Read, Update, Delete).
- Implement booking requests, support conflict detection.
- Implement user approval workflows and booking approval workflows.
- Store schedules/booking data in the database.
- **Week 16: Final Evaluation (FYP-1**

Completion) Weeks 17–19: Finalization of Features

- Use faculty priority/urgent booking logic.
- Implement check-in, check-out and no-show detection.
- Include Open Session Publication feature to make use of public sessions.

Weeks 20–23: Advanced Modules & Integration

- Incorporate sophisticated search and filtering by multiple criteria.
- To build maintenance modules and issue reporting modules.
- Implement notification logic (email/In-app as decided).
- Sync Calendar with Google (if applicable).

Week 24: Mid Evaluation

Weeks 25–27: Reporting, Analytics & Optimization

- Create reporting and Analytics dashboard for Admins.
- Institute audit-trail and usage statistics.
- Design and tune databases and systems.

Weeks 28–30: System Integration & Testing

- Integrate Frontend & Backend modules.
- Perform functional, integration and security testing and performance testing.
- Resolve bugs and system related problems.

Week 31: Monitoring and Maintenance

- Analyze the behavior of the system and the usability.
- Apply any last-minute bug fixes and UI/UX enhancements.
- Complete SRS, SDS and Test Cases).

Week 32: Deployment and Launch

- Install system in the host system.
- Do end-to-end testing to finalize.
- Create the final report, presentation and project demonstration.

Week 32: Final Evaluation (FYP-2 Completion)

12. Project Schedule

FYP-1 Schedule

Task Name	Duration	Start	Finish
Topic & Proposal	7	15-Sep-2025	22-Sep-2025
Initial Planning & Research	7	23-Sep-2025	30-Sep-2025
SRS, SDC, UML	14	1-Oct-2025	15-Oct-2025
UI Design	21	16-Oct-2025	06-Nov-2025
Frontend Development	28	7-Nov-2025	05-Dec-2025
Backend Structure APIs	14	6-Dec-2025	20-Dec-2025
User Management Functionality	14	21-Dec-2025	04-Jan-2026

Table 1:FYP-1 Schedule Table

Gantt-Chart FYP-1

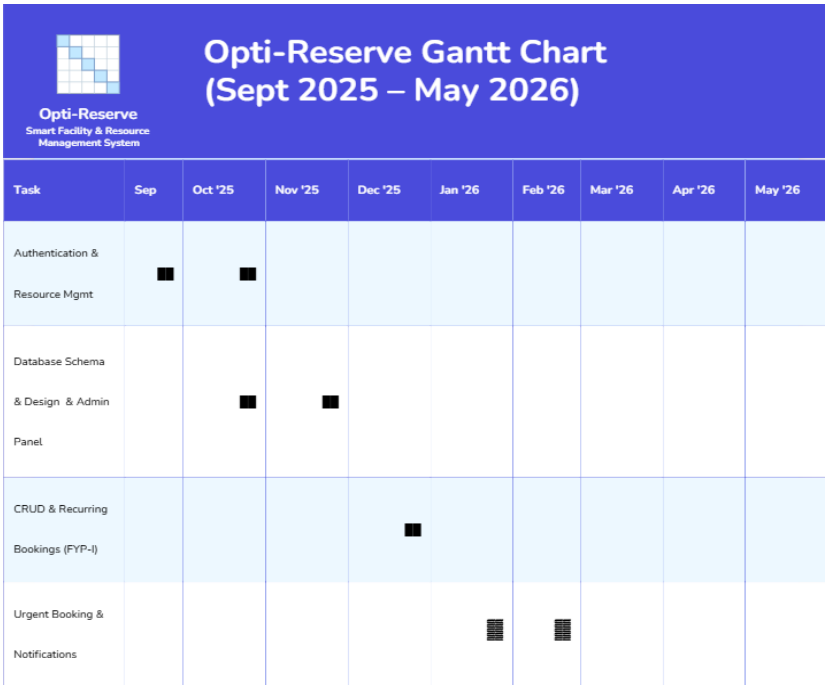


Figure 1 FYP 1 Gantt Chart

FYP-2 Schedule

Task Name	Duration	Start	Finish
Finalization of Features	14	10-Feb-2026	24-Feb-26
Back-end development	35	25-Feb-2026	1-Apr-26
Database Management	21	2-Apr-2026	23-Apr-26
Integration Testing	14	24-Apr-2026	8-May-26
Monitoring and Maintenance	14	9-May-2026	23-May-26
Launch of system	7	24-May-2026	31-May-26

Table 2: FYP-2 Schedule

Gantt-Chart FYP-2

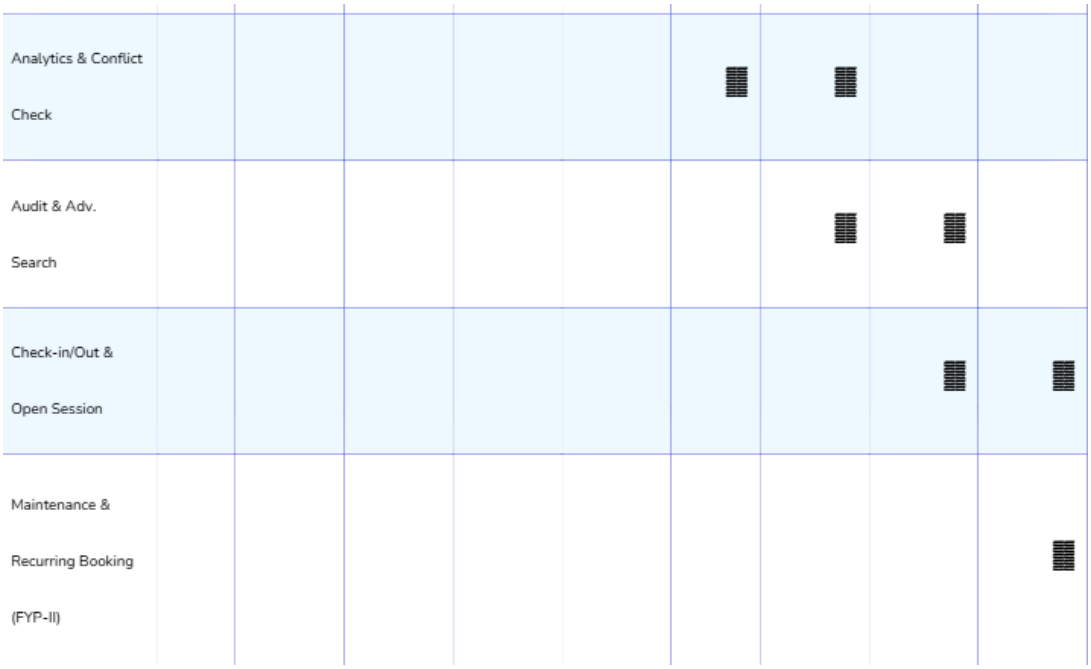
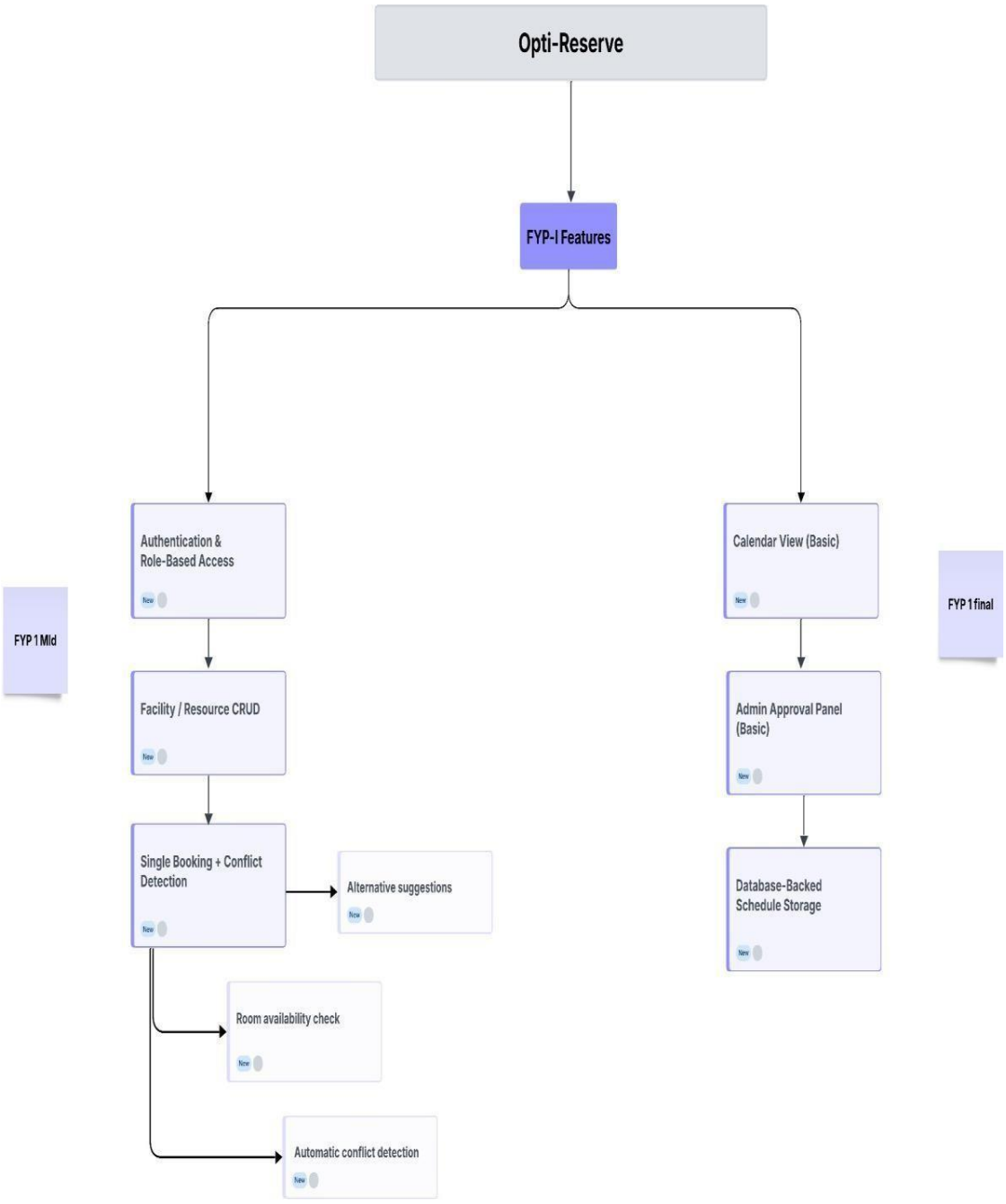


Figure 2: FYP 2 Schedule - Gantt Chart

13. Work Breakdown Structure

13.1 FYP-1



13.2 FYP-2

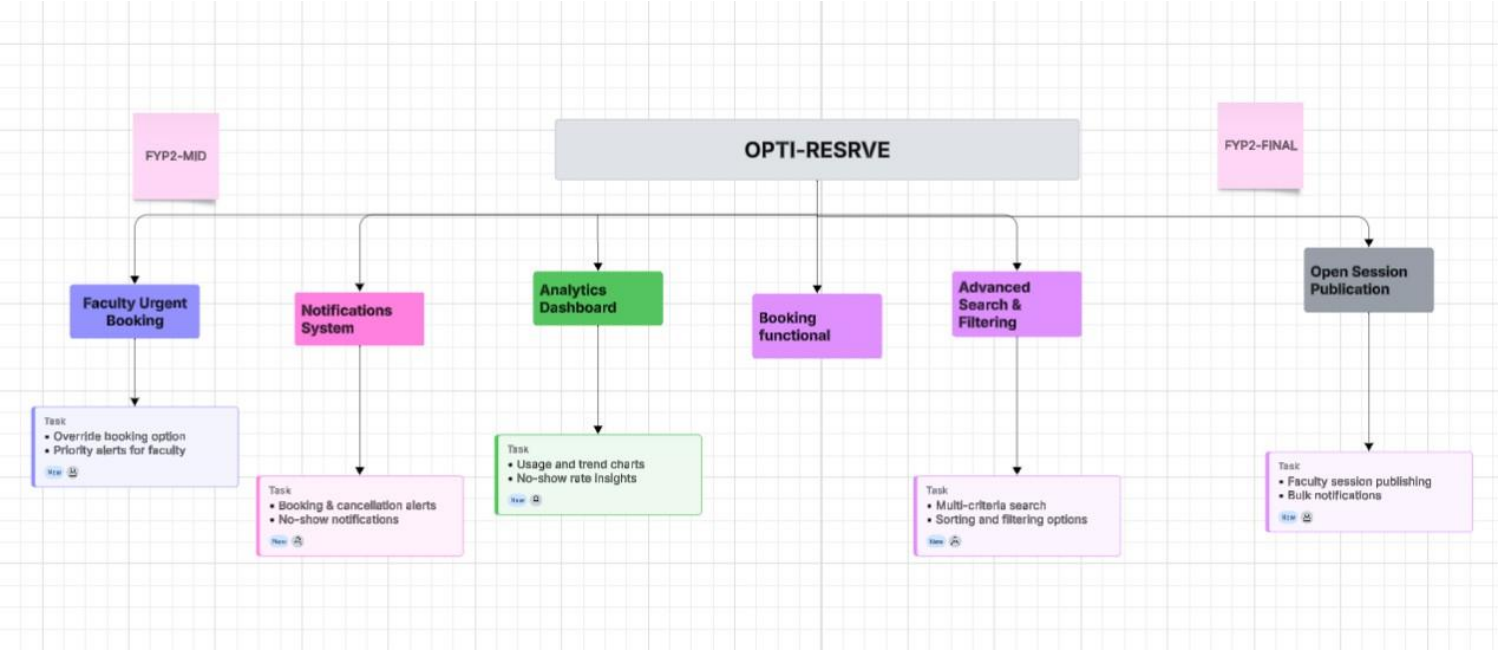


Figure 4: FYP 2 Work Breakdown Structure

14. References

[1] Smith, John. *Web Development with Node.js and React*. O'Reilly Media, 2021.
[2] Doe, Jane. "The Rise of Facility Management Systems." *Tech Trends*, April 2022.
[3] Google Developers. "Google Calendar API Overview." Accessed Sept 22, 2025. https://developers.google.com/calendar/api/guides/overview
[4] Lee, Mary. "Advancements in Facility Management Solutions." *Business Tech Review*, May 2023. Accessed via ProQuest, SZABIST Library, Sept 22, 2025.

Software Requirement Specifications

1. Introduction

1.1. Purpose

This Software Requirements Specification (SRS) document outlines the requirements of the Smart Facility & Resource Management System (Version 1.0) for Opti-Reserve. The system allows to easily book institutional facilities like auditoriums, labs, meeting rooms and equipment. It provides conflict-free scheduling, allows for emergency faculty scheduling, allows for check-in and out tracking, maintenance reporting and publishing of sessions in progress. This is the set of all requirements that the system is required to satisfy, both functional and non-functional – for this systems initial release.

1.2. Document Conventions

Heading	Font Size: 14pts Font Style: Times New Roman
Regular Text	Font Size: 12pts Font Style: Times New Roman

Table 3: Document Convention

1.3. Intended Audience and Reading Suggestions

The following are the major targets of this document:

- **Developers:** To be aware of functions and constraints to be implemented.
- **Testers:** They are responsible for ensuring the platform's functionality, performance, and security. These will focus on Section.
- **Project Supervisors and FYP Advisor:** Supervise the progress of the project, make sure the requirements are fulfilled and offer directions. They should read this entire document paying special attention to Section.
- **Project Team Members (Om & Gulzar Hussain):** Both project team members (Om & Gulzar Hussain) are equally responsible for development, testing and documentation, hence the SRS reflects their responsibilities in the project.
- **System Administrators/Clients:** To ensure the final product is suitable to their needs.

1.4. Product Scope

The Opti-Reserve platform is a single platform to manage institutional facility bookings. The main purpose is to avoid double bookings using the available real-time data and to offer administrative control over all the reservations and users' accounts. The The system will have user authentication, categorize resources, include a request/approval system for users to book, and have easy content management for public facing pages. It's designed to be a professional, responsive and efficient scheduling tool for the institution. There are also some advanced features like check-in/out management, maintenance report, advanced search and open session publication to be provided for better operational transparency and efficiency.

1.5. Reference

- Node.js Documentation (www.nodejs.org)
- Express.js Guide: https://expressjs.com/
- Prisma ORM Documentation (www.prisma.io)

2. Overall Description

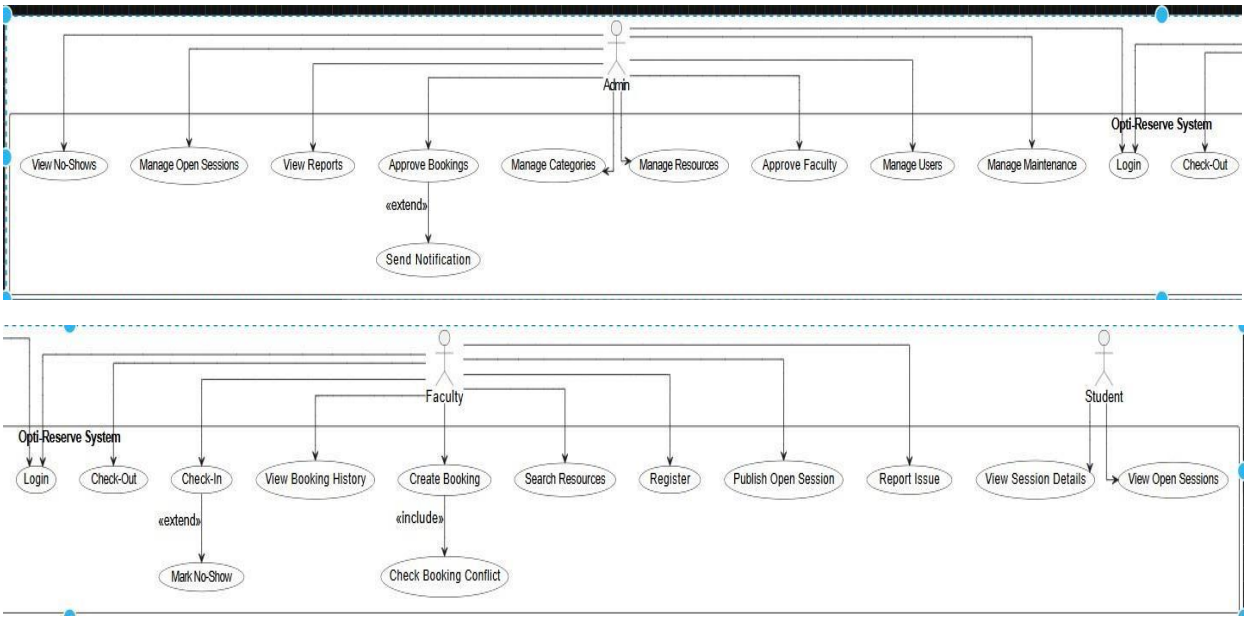


Figure 5: Use case Diagram

2.1. Product Perspective

Opti-Reserve is a new product, and no one is required to use any of the other products. Standalone web-based application designed to be the "single repository" of booking for the facility and equipment in the institution. It does not replace an existing digital system, but it is intended to bring together and automate a process that is done manually or in a fragmented manner.

2.2. Product Functions

- **User Registration and Authentication:** Provides a way for faculty members to register and securely log in, and only after administrative approval is granted may faculty members activate their accounts.
- **Role-Based Access Control:** Restricts the access to the features of the system to different users based on their role, such as Admin, Faculty or Students/Public user.
- **Category and Resource Management:** Allows administrators to add, edit, view, and remove categories and resources like classrooms, labs, auditoriums and equipment.
- **Real-Time Resource Availability Viewing:** Enables users to see real-time availability of facilities and resources, before submitting booking requests.
- **Facility Booking Management:** Allows the faculty to book a facility by choosing the

resources, time slot and purpose and stores all the bookings in a central database.

- **Automated Booking Conflict Detection:** Automatically detects and prevents overlaps in booking same resource at the same time.
- **Booking Approval Workflow:** Gives the administrators the ability to view, approve or deny booking requests, and the status is updated in real time.
- **Booking History and Status Tracking:** Allows faculty members to access the history of their bookings, as well as the status of their bookings.
- **Advanced Search and Filtering:** Enables searching and filtering of resources according to various criteria such as name, type, availability and others.
- **Check-In and Check-Out Management:** Records actual facility usage and provides a check-in/out point for faculty to check in when booking and check out when the facility is used.
- **No-Show Detection and Logging:** f a person doesn't show up to the booking within the grace time, the booking will be automatically marked as a No-show.
- **Open Session Publication:** Open Session Publication – enables users to mark an approved booking as a public session and view it by other students and/or public users.
- **Public Session Viewing:** Allows students and public users to browse and view information about published open sessions, without requiring authentication.
- **Maintenance and Issue Reporting:** Give faculty the ability to submit maintenance requests and/or facility/equipment problems and have them tracked and updated by the administrator.
- **Reporting and Analytics:** Generates system reports and analytics for administrators, such as booking trends, resource utilization and no shows.
- **Audit Trail and Activity Logging:** Maintains a record of critical activities such as approvals, rejections and overrides, ensuring transparency and accountability.
- **Notification Support:** Assists in sending and receiving notifications to and from the system during events like account approval, booking approval/rejection and session updates (via email or in-app as set up).
- **Public Content Management:** Provides the ability to manage public content like the home page, gallery and contact information.

2.3. User Classes and Characteristics

The Opti-Reserve system caters for a variety of classes of users, each with a different role, access rights and responsibilities. Each user class is created to provide for secure access control, effective resource utilization and intuitive user experience.

2.3.1. Administrator

Administrators are members of the administration who have full privileges to the system. Their role is to manage the overall configuration of the system, users and its operations. Admin users can approve or deny faculty registrations and bookings, manage facility categories and resources, reports and analytics to track the system usage, manage the maintenance requests, manage the public content and enforce compliance with institutional policies. This User class can have the most access and priority in this system.

2.3.2. Faculty Member:

Faculty members are those who are approved academic/research staff that are the main users of the booking system. Once registered and administered, faculty members can log

in the system, search and view available resources as well as submit their booking requests. Monitor and history booking status and actions, check-in/out, report maintenance problems and publish approved bookings as open sessions. Faculty users need to have robust access to scheduling tools and are given high priority in operations.

2.3.3. Student / Public User

Access to the system is restricted to students (and public users) who can only read. They are not authenticated and can only see open sessions that are publicly published, and general informational pages (home, contact etc.). This user class is not engaged in booking or management and is the least important one in the system.

2.4. Operating Environment:

2.4.1. Hardware Platform:

The system will be hosted on a cloud-based Linux server (AWS, Azure or Digital Ocean) and the systems used by clients to access the system can be standard desktop or laptop computers, with the minimum requirement of 4 GBs of RAM and an up-to-date web browser.

2.4.2. Operating System

- **Server:** Any modern Linux or cloud environment that is able to run Node.js will do.
- **Database: MySQL** server.
- **Client:** Desktop and mobile browsers – such as Chrome, Firefox, Safari, Edge.

2.4.3. Software Components

- Technology stack: Node.js + Express EJS
- Database: Prisma (SQL database).
- Version Control: CI/CD Git and GitHub.
- Web Server: Nginx or similar reverse proxy server for deployment.
- Authentication Service: Gmail API for Two-Factor Authentication (2FA)

2.5. Design and Implementation Constraints

- Gmail API Integration is required for email notification and two Factor Authentication
- We will comply with secure coding practices and use HTTPS, JWT authentication, and password hashing.
- Application will follow RESTful API standards.
- Code will follow JavaScript conventions and use git for version controlling.
- Hosting will be done on a Linux-based cloud server.

- The platform should run efficiently on standard systems (minimum 4 CPU, 8 GB RAM)
- Integration with external systems is limited to Gmail services only.

2.6. User Documentation

2.6.1. Faculty, User Manual: we will provide a PDF guide explaining system features, login procedures and platform navigation.

2.6.2. Online Help Section: Integrated help page within the web application FAQs and you-tube tutorials.

2.7. Assumptions and Dependencies

- We assumed that users will have a stable internet connection and access to a modern web browser (Chrome, Edge, Firefox).
- We assume that the availability and proper functioning of Gmail services for two-factor authentication and email notifications.
- We assumed that the university hosting server or chosen cloud provider will remain operational and accessible throughout the project lifecycle.
- Our Project depends on regular communication and feedback from the FYP advisor and evaluation panel for timely updates and improvements.
- We assume that any change or deprecation in third-party libraries or APIs (Gmail API, Node modules) could affect system performance and require code updates.
- It is assumed that users will have to register using valid institutional email accounts to maintain system authenticity and security.
- According to the project timeline, we assume both team members (Om and Gulzar Hussain) will contribute consistently according to predefined roles and milestones.

3. External Interface Requirements

3.1. User Interfaces

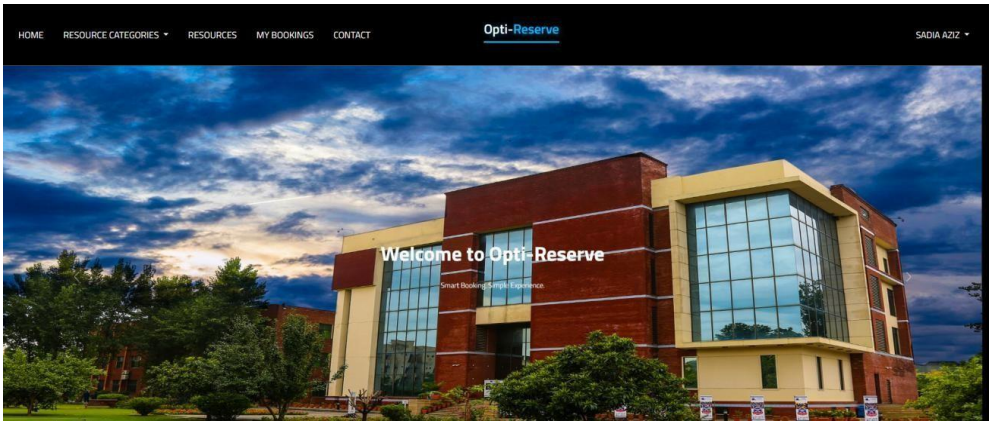


Figure 6 Screen

3.2. Sign Up

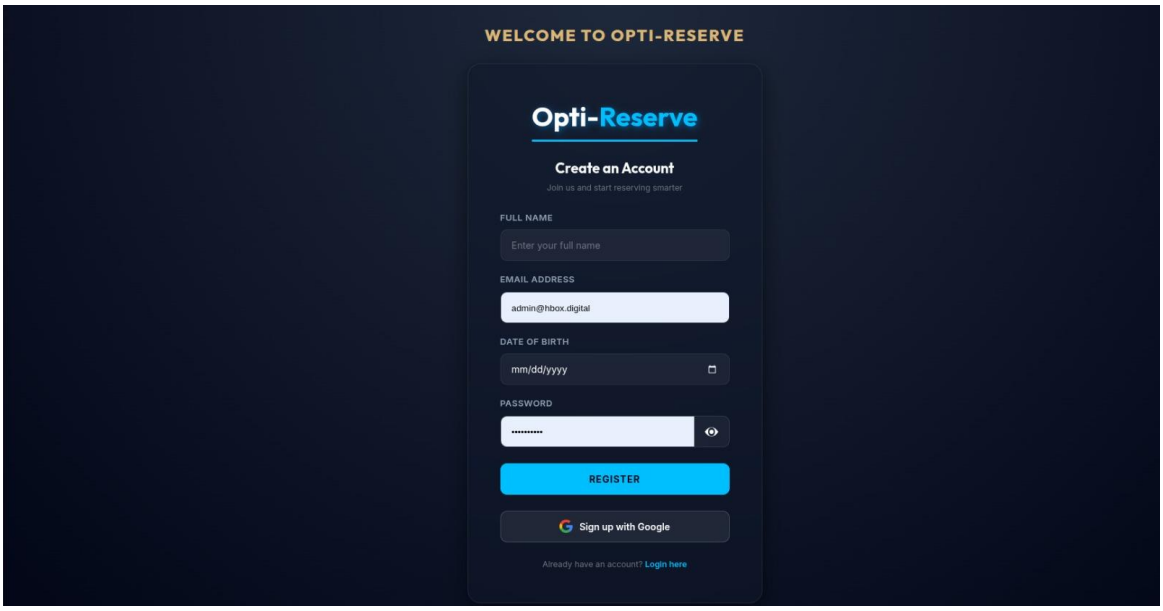


Figure 7 Sign Up Screen

3.3. Login

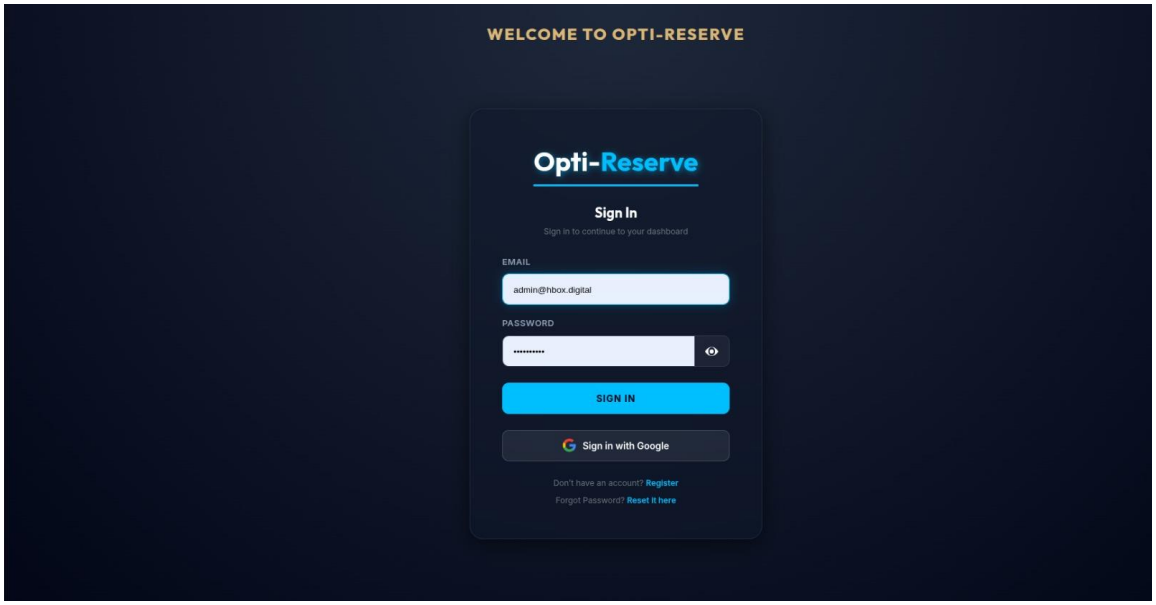


Figure 8 Login Screen

3.4. Forget Password

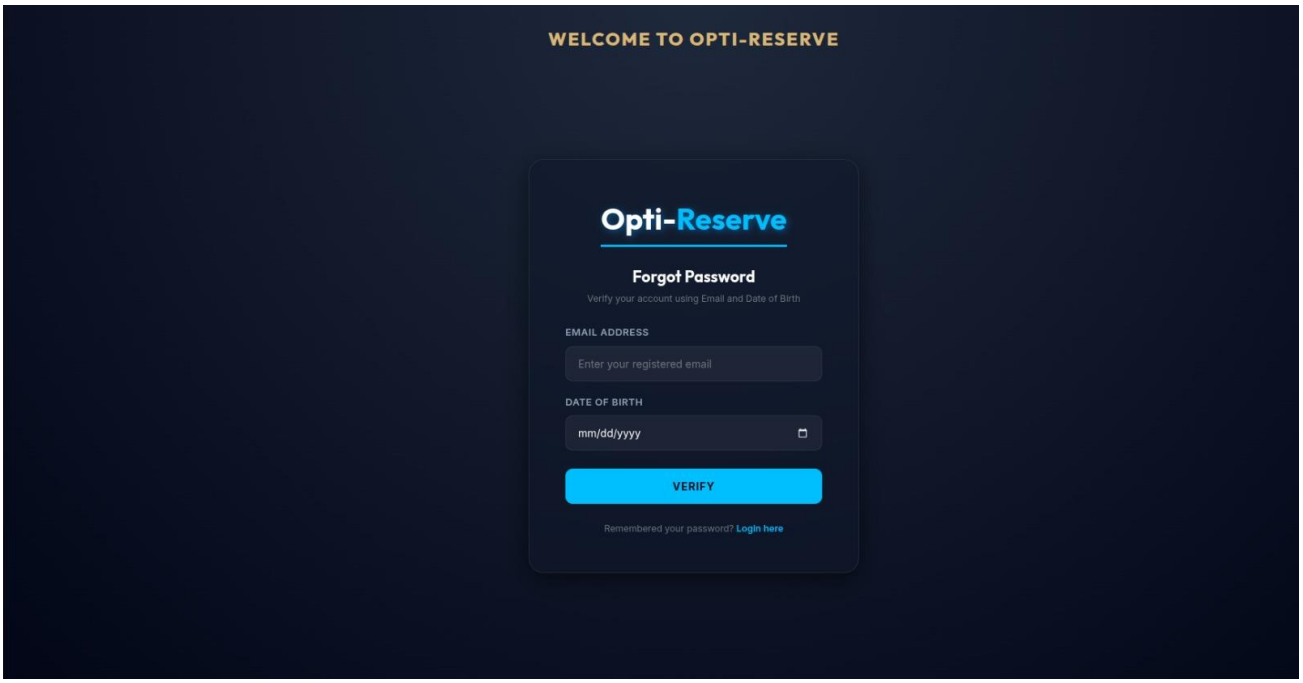


Figure 9 Forget Password Screen

3.5. About

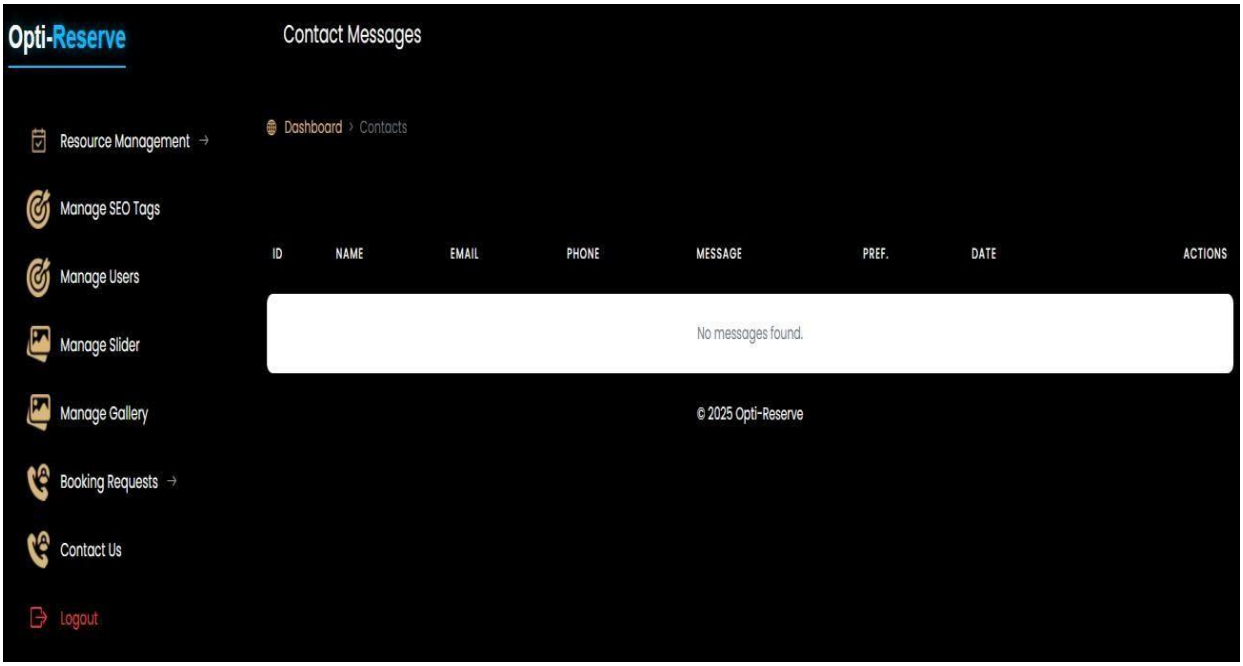


Figure 10 About Screen

3.6. Admin Board

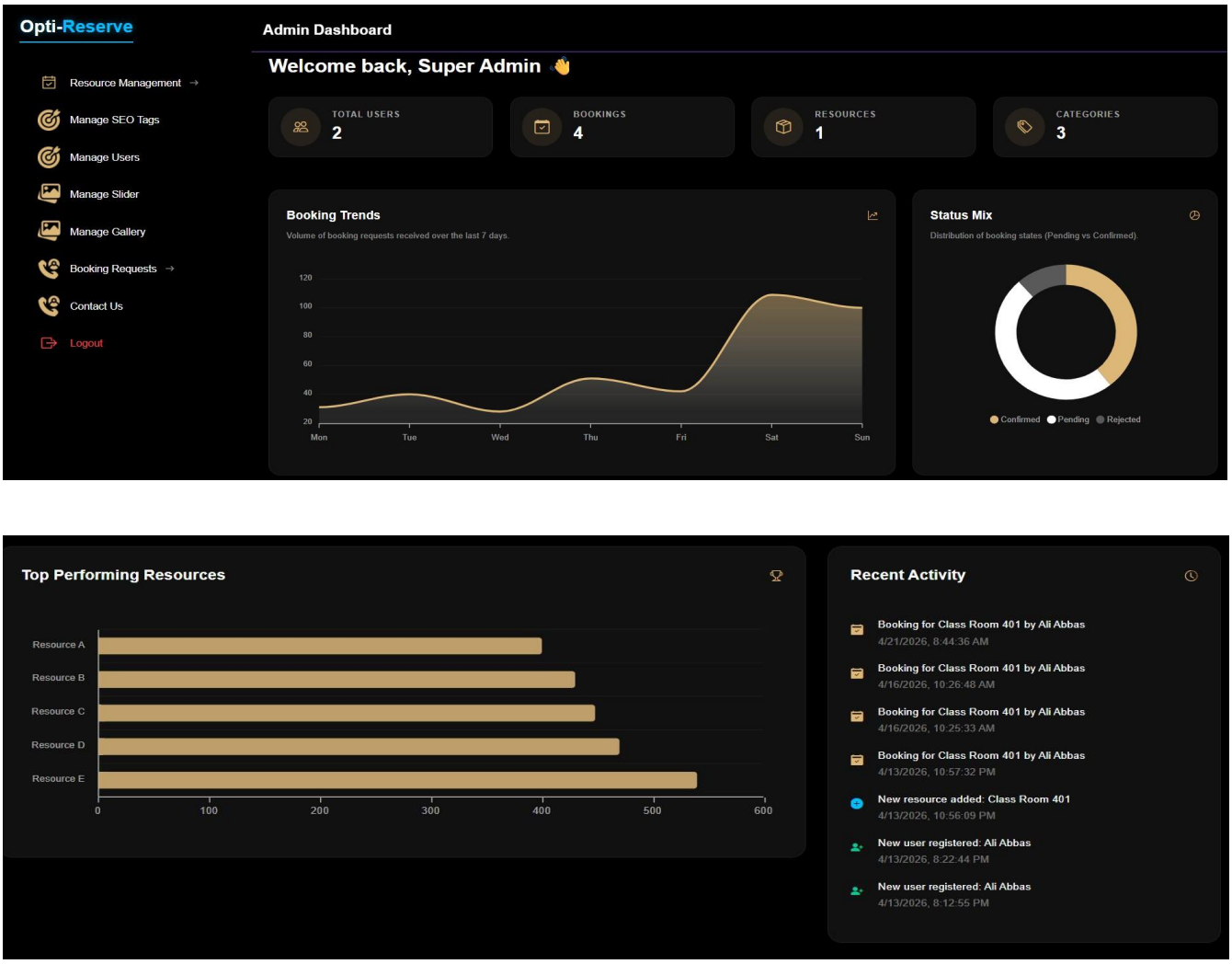


Figure 11 Admin Screen

3.7. Users

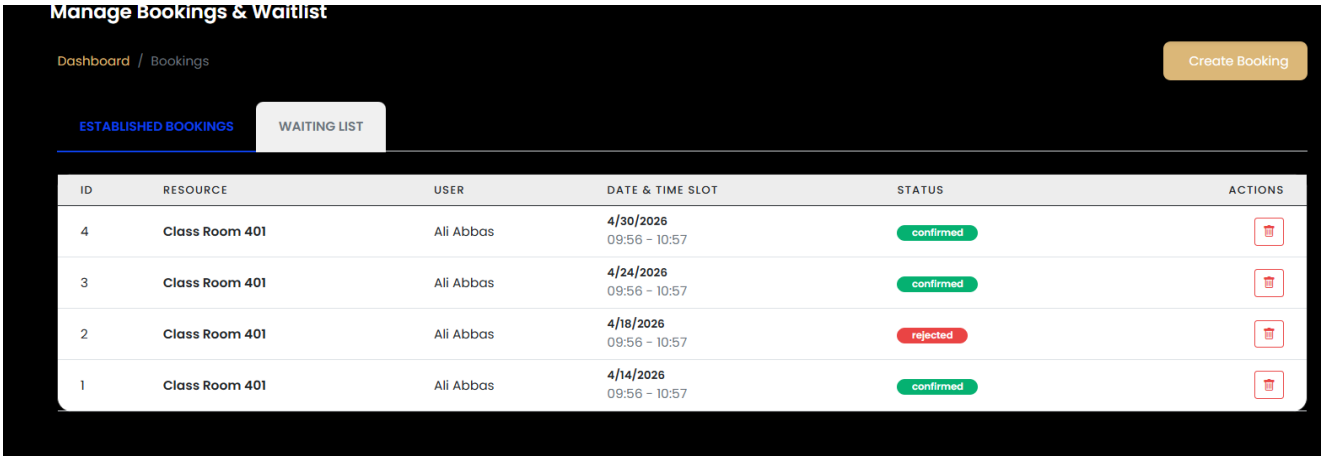


Figure 12 Users Screen

3.8. Schedules

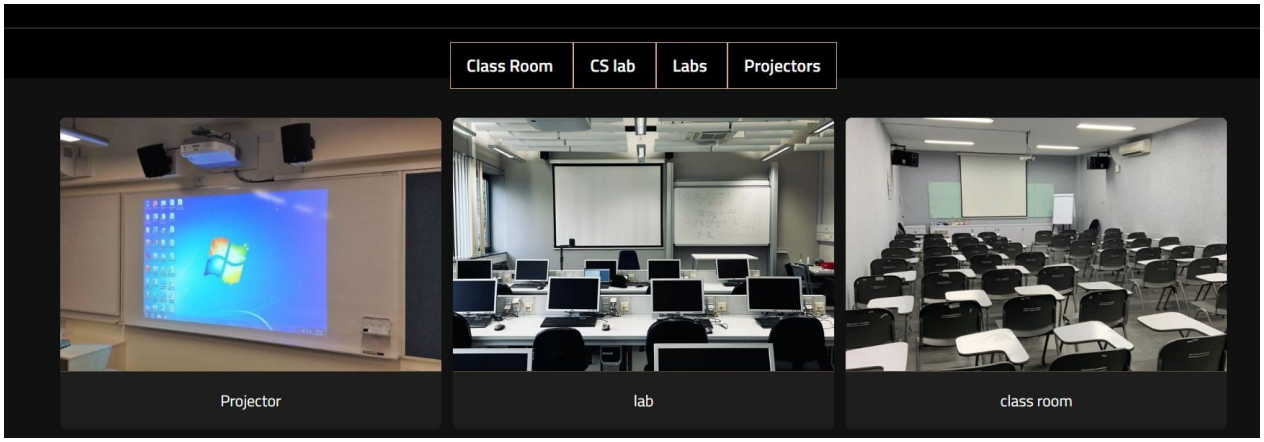


Figure 13 Schedules Screen

3.9. Account Setting

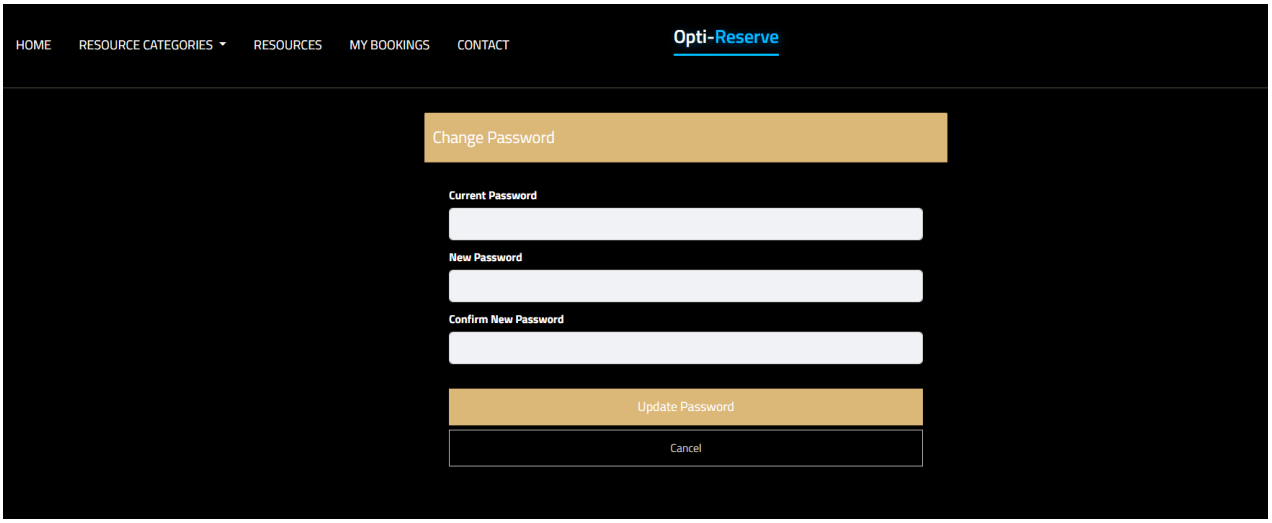


Figure 14 Account Setting Screen

3.10. Hardware Interfaces

- There will be no hardware interface required directly because the only way to use Opti-Reserve will be through a web browser.
- The web server of our application will run on a cloud or university-provided server.

3.11. Software Interfaces

3.11.1. Backend:

Our backend is built using Node.js along with the Express.js framework. It handles all business logic, authentication, and processing of user requests, ensuring efficient communication between the frontend and database.

3.11.2. Database:

We will be using Prisma SQL to securely store user information and storing the data and backend connects to the database through Prisma ORM.

3.11.3. Email Services:

Our system will be integrated using Gmail API to send verification emails and provide two-factor authentication for secure login.

3.11.4. Version Control:

Our project's updates will be managed through GitHub, allowing both members to collaborate efficiently and maintain version control throughout the project lifecycle.

3.11.5. Communication Flow:

Our web application exchanges data between the client and server in JSON format through RESTful API endpoints after considering smooth and reliable communication across all modules of the web application.

3.12. Communications Interfaces

Our platform Opti-Reserve uses HTTPS with TLS encryption to ensure secure communication between the web browser and the server. All data is exchanged through RESTful APIs in JSON format, while SMTP protocols handle email notifications such as verification and authentication messages. The system is fully compatible with modern web standards, and all communication channels remain encrypted and protected to maintain user privacy and data security.

4. System Features

4.1. User Authentication and User Management

4.1.1. User Registration

4.1.1.1. Description

It provides new users to create new accounts on Opti-Reserve by providing essential details and email verification.

4.1.1.2. Stimulus/Response Sequences

Use Case Name: User Registration
Description: Enables new users to create student accounts
Actors: Users
Goal: Create secure accounts for platform access
Pre-Conditions: User doesn’t have an existing account

Basic Course of Events: **Step 1:** User enters the registration page. **Response:** Displays the registration form. **Step 2:** User fills out the form with email, username, password and personal information. **Response:** The system validates and sends an activation email.
Step 3: User clicks the activation link. **Response:** System activates the account and redirects to the login page.
Alternate Flow (Invalid Inputs): · **Step:** User enters existing email or weak password. **Response:** System highlights errors and requests corrections
Exception Flow (System Errors): · **Step**: Email service fails. · **Response**: Displays "Verification email delayed" message
Post-Conditions: User account is created and ready.

Table 4 User Registration Stimulus/Response

4.1.1.3. Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide a form to input user registration details	High	Pending
REQ-2	Validate email format and uniqueness	High	Pending
REQ-3	Enforce strong password policy	High	Pending
REQ-4	Send email verification link	High	Pending

Table 5: User Registration Functional Requirements

4.2. User Login

4.2.1. Description

Delivers secure authentication for registered users to log in to the Opti-Reserve Platform with their credentials.

4.2.2. Stimulus/Response Sequences

Use Case Name: User Login
Description: Authentication users and grant platform access
All Users: Users, Administrator
Goal: Secure access to learning platform

Pre-Condition: User has verified account
Basic Course of Events/Main Flow: **1:** User navigates to login page **Response:** System displays login form with email/username and passwd fields **2:** User enters credentials and submits **Response:** System validates credentials against database **3:** Credentials are correct **Response:** System creates session and redirects to dashboard
Alternate Flow (Invalid Inputs): **Step:** User Enters incorrect password **Response:** System shows invalid credentials retry.
Expectation Flow (System Errors): **Step:** Authentication service unavailable **Response:** Login temporary unavailable.
Post-Conditions: User has logged in successfully and redirected to Dashboard.

Table 6: User Login Stimulus/Response

4.2.2.1. Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide login form with email/usernames and password fields	High	Pending
REQ-2	Validate user credentials against database	High	Pending
REQ-3	Implement secure session management	High	Pending
REQ-4	Redirect to dashboard upon successful login	High	Pending
REQ-5	Limit login attempts to prevent brute force attacks	High	Pending
REQ-6	Show appropriate error messages for failed login	Medium	Pending
REQ-7	Remember user session for specified duration	Medium	Pending
REQ-8	Implement logout functionality	High	Pending

Table 7: User Login Functional Requirements

4.3. Forgotten Password

4.3.1. Description

It enables users to reset password using secure email verification in case they forget their logins.

4.3.2. Stimulus/Response Sequences

Use Case Name: Forgot password
Description: Secure password recovery
Actors: Users, Administrator
Goal: Regain account access without compromising security
Pre-Condition: User has registered account
Basic Course of Events: **Step 1: User clicks forgot password on login page**. **Response:** Opens a message with the email address. **Step 2:** User enters registered e-mail address. **Response:** Email is valid, reset link sent. . **Step 3:** User clicks reset link in email. **Response:** System verifies token and displays a password reset. **Step 4:** User types in a new password. **Response:** Update password is successful.
Alternate Flow (Invalid Email): **Step:** User enters unregistered email **Response:** System shows Email not found message **Alternative Flow (Expired Link):** **Step:** Unregistered email entered by user. **Response:** Enter the email address in the system.
Exception Flow (Email Service Down): **Step:** System can't send reset email. **Response:** System shows that Service temporarily unavailable.
Post-Condition: User password successfully reset and can login with new credentials

Table 8: Forget Password Stimulus Response

4.3.2.1. Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Forgot password option on login page	High	Pending
REQ-2	Validate email against registered users	High	Pending
REQ-3	Generate secure password reset token	High	Pending
REQ-4	Send password reset email with secure link	High	Pending
REQ-5	Verify reset token validity and expiration	High	Pending

REQ-6	Provide secure password reset form	High	Pending
REQ-7	Enforce password policy during reset	High	Pending
REQ-8	Invalid reset token after use	Medium	Pending
REQ-9	Set token expiration time 1 hour	High	Pending

Table 9: Forget Password Functional Requirement

4.4. Account Activation

4.4.1. Description

It verifies user email addresses through activation links sent to the user during activation to ensure account authenticity and prevent spam.

4.4.2. Stimulus/Response Sequences

Use Case Name: Account Activation
Description: Email verification for new user accounts
Actors: Users
Goal: Confirm user email and activation account
Pre-Condition: User completed registration, but account not activated
Basic Course of Events: **Step 1:** User completes registration **Response:** System sends activation email with secure link

Step 2: User clicks activation link in email **Response:** System validates activation token **Step 3:** Token is valid. **Response:** System activates account and shows success message.
Alternative Flow (Expired Link): **Step:** User clicks corrupted activation link **Response:** System shows Invalid activation link error.
Exception Flow (Email Delivery Failed): **Step:** Activation email can't be delivered **Response:** System allows manual to resend from login page
Post-Condition: User account activated and ready for login

Table 10: Account Activation Stimulus Response

4.4.2.1. Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Send activation email after registration	High	Pending
REQ-2	Generate secure activation token	High	Pending
REQ-3	Validate activation taken on link click	High	Pending
REQ-4	Activation user accounts upon successful completion.	High	Pending
REQ-5	Set activation token expiration 1 hour	High	Pending
REQ-6	Provide resend activation email functionality	Medium	Pending
REQ-7	Prevent login for un activated accounts	High	Pending
REQ-8	Shows activation status on login attempt	Medium	Pending
REQ-9	Track number of activation email resends	Medium	Pending

Table 11 Account Activation Functional Requirements

4.5. View Available Classroom

4.5.1. Description

Provides comprehensive progress visualization for Availability of the rooms.

4.5.2. Stimulus/Response Sequences

Use Case Name: View Available Classroom
Description: Display user advancement through curriculum
Actors: Users
Goal: To track the availability of the rooms
Pre-Conditions: User is logged in
Basic Course of Events: **Step 1:** User navigates to progress dashboard **Response:** System displays completion statistics and charts **Step 2:** User filters by room number **Response:** System updates progress view accordingly
Post-Conditions: User Knows the room number he wants to search for

Table 12: View Available Rooms Stimulus Response

4.5.2.1. Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Display visual progress charts and statics	High	Pending
REQ-2	Shows available Room Number	High	Pending
REQ-3	Track on which rooms are available	High	Pending
REQ-4	Provide activity timeline and history	Medium	Pending
REQ-5	Analyze number of rooms available	Medium	Pending

Table 13: View available Classroom Functional Requirements

4.6. Admin Management System

4.7. User Management

4.7.1. Description

It offers the option of changing the password with a secure email confirmation in case the user forgets its login details.

4.7.2. Stimulus/Response Sequences

Use Case Name: Manage Users
Description: Admin controls user accounts and permissions
Actors: Administrator
Goal: Maintain user database and access control
Pre-Conditions: Admin is logged in with privileges
Basic Course of Events: **Step 1:** User clicks forgot password on login page **Response:** System will show the email address. **Step 2:** Admin selects user for management **Response:** System shows user details and activity history **Step 3:** Admin modifies user role or status **Response:** System updates user permissions immediately
Alternate Flow (User Not Found**):** **Step:** Search returns no results **Response:** System shows No users found message
Post-Conditions: User accounts properly managed

Table 14: Manage Users Stimulus Response

4.7.2.1. Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Display all users with permissions	High	Pending
REQ-2	Search users by usernames, email	High	Pending
REQ-3	Modify user roles and permission	High	Pending
REQ-4	View user activity logs	Medium	Pending
REQ-5	Enable or disable user accounts	High	Pending

Table 15: Manage Users Functional Requirements

4.8. Session Management

4.8.1. Description

Track and manage active sessions, including time limit and resources allocation

4.8.2. Stimulus/Response Sequences

Use Case Name: Session Management
Description: Track active users sessions
Actors: System
Goal: Ensure fair resources distribution
Pre-Conditions: User has active session
Basic Course of Events: **Step 1:** User starts session **Response:** System records start time and allocations resources **Step 2:** System monitors session duration **Response:** User receives time remaining warnings **Step 3:** Session time expires **Responses:** System automatically terminates session
Alternate Flow (Early Completion): **Step:** User manually ends session Response: System immediately releases resources
Post-Conditions: Optional resources utilization

Table 16: Manage Lab Session Stimulus Response

4.8.2.1. Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Track session start and end times	High	Pending
REQ-2	Enforce maximum session duration	High	Pending
REQ-3	Provide session time remaining alerts	Medium	Pending
REQ-4	Allow manual session termination	Medium	Pending
REQ-5	Preserve session state for resumption	Low	Pending

Table 17: Management Session Functional Requirements

4.9. Analytics and Reporting

4.10. System Analytics

4.10.1. Description

It provides comprehensive analytics on platform usage, user engagement.

4.10.2. Stimulus/Response Sequences

Use Case Name: View System Analytics
Description: Admin accesses platform performance data
Actors: Administrator
Goal: Monitor platform health and user engagement
Pre-Condition: Admin is logged in
Basic Course of Events: **Step 1:** Admin accesses analytics dashboard **Response:** System displays key metrics and charts **Step 2:** Admin filters data by date range or user group **Response:** System updates analytics displays. **Step 3:** Admin exports reports **Response:** System generates downloads reports
Post-Condition: Data-driven decision making

Table 18: View System Analytics Stimulus Response

4.10.2.1. Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Displays user engagement metrics	High	Pending
REQ-2	Shows schedule completion statistics	High	Pending
REQ-3	Track platform usage patterns	Medium	Pending
REQ-4	Generate exportable reports	Medium	Pending
REQ-5	Provide real-time analytics dashboard	High	Pending

Table 19: View System Analytics Functional Requirements

4.10.3. Facility Booking Management

Allows the Admin to update non-system content like the home page, gallery, contact details, and SEO

4.10.4. Stimulus/Response Sequences

Use Case Name: Facility Booking Management
Description: System saves file and updates Gallery Page view
Actors: Administrator, User, Faculty
Goal: Allows the Admin to update non-system content
Pre-Condition: Must be Admin
Basic Course of Events: **Step 1:** User/Admin requests progress report. **Response:** System complies Schedule data. **Step 2:** System generates visual report **Response:** Report displayed with charts and metrics **Step 3:** User/Admin exports report **Response:** System provides PDF/CSV option
Post-Condition: Schedule progress documented

Table 20: Facility Booking Management Stimulus Response

4.10.4.1. Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Generates individual user progress reports.	High	Pending
REQ-2	Create performance reports.	Medium	Pending
REQ-3	Export reports in multiple formats PDF, CSV	Medium	Pending
REQ-4	Show All the report	Medium	Pending
REQ-5	Compare performance to previous.	Low	Pending

Table 21: Facility Booking Management Functional Requirements

5. Other Nonfunctional Requirements

5.1. Performance Requirements

All web pages should be accessed in less than 3 seconds, and the system should be able to accommodate at least 100 concurrent users. The time to complete a backend (e.g. login, query a database) request must be less than 2 seconds. It should automatically scale as its traffic increases to keep it running smoothly.

5.2. Safety Requirements

Regular backups and encryption of user data should be provided to ensure that user data is not accessible to unauthorized users or that it does not get lost. Deleting account(s) and modifying data requires user's confirmation. The platform should meet the basic cyber security standards to ensure that there are no data breaches or interactions with users, that are damaging.

5.3. Security Requirements

The system needs to utilize safe customer authentication using e-mail verification and two factor authentication (2FA). Strong hashing algorithms are used to store all passwords and data is transmitted over HTTPS and TLS encryption. Only authorised persons can access certain features with role based access control. It also needs to follow the standard web security protocols, such as against SQL injections, CSRF, and XSS attacks.

5.4. Software Quality Attributes

Reliability will be a key value in the system to ensure its performance under high load and usability to ensure its intuitive use. It will ensure security via encryption and authentication measures and ensure maintainability with modular and well documented coding. As the number of users grows, scalability will ensure that it operates seamlessly, and portability will ensure its compatibility with the leading browsers and operating systems. Availability will be given priority to ensure uninterrupted availability and to reduce downtime.

5.5. Business Rules

Platform settings and managing users, for example, can only be done by administrators, who have full control of system operations. Users can view room schedules, their access privileges. All users of the system must abide by security and ethical usage policies and all activities on the system will be monitored and recorded to ensure that users do not breach these policies. If any users use or access the accounts in any manner other than that specifically designated, their accounts will be suspended or terminated.

5.6. Business Model Canvas

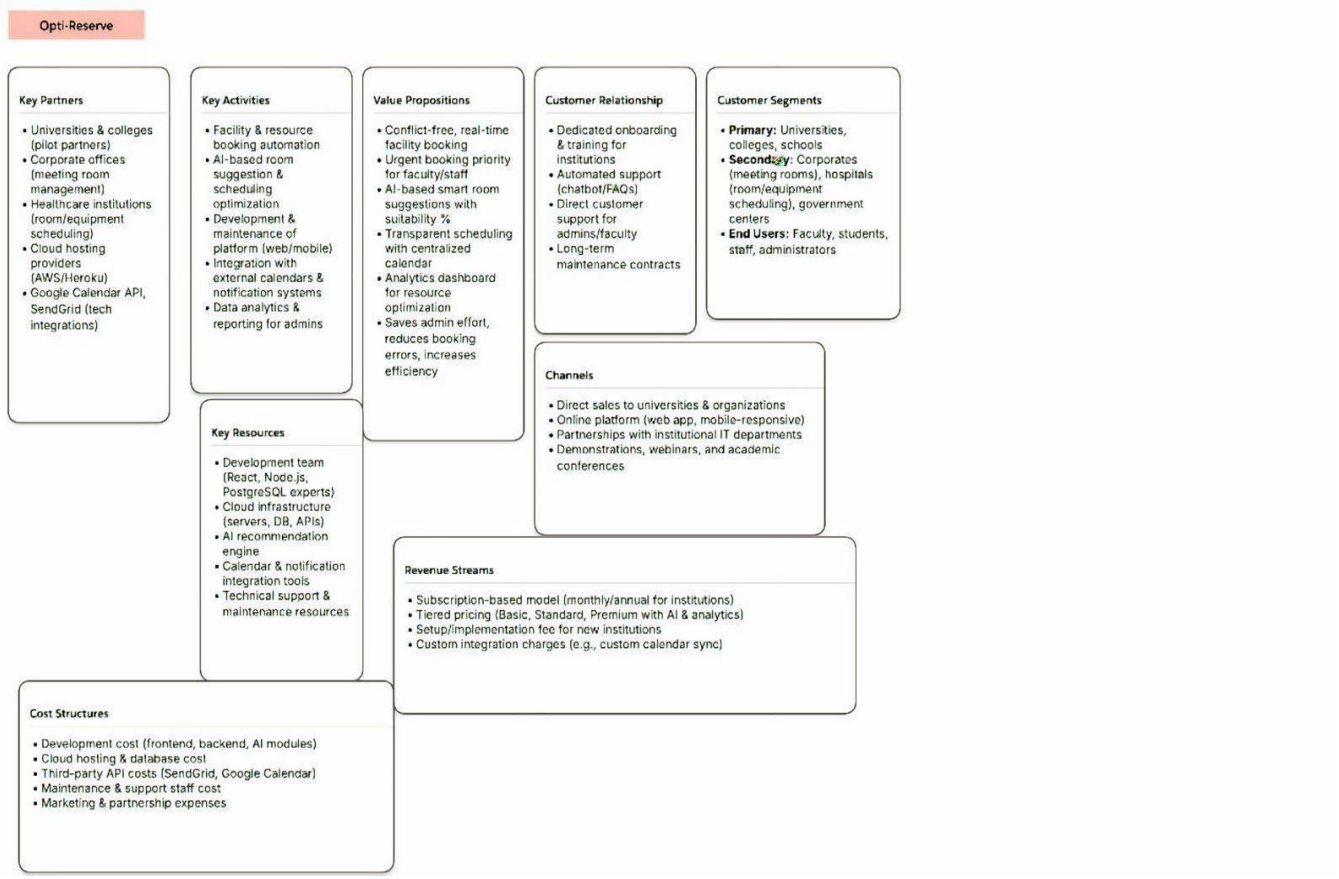


Figure 15 Business Canvas Model Diagram

6. Other Requirements

The Opti-Reserve system will provide regular data backups and recovery to ensure data is reliable and data loss is minimized. The initial release will be in English language, and will be internationalized in the future. It will notify users of important events like changes in their account status or booking status, log user actions into an audit log for important actions, and clearly show error messages, while not giving away any sensitive information. Also, the system will be designed to be modular and extensible for future enhancements and long-term maintenance.

Appendix A: Glossary

S.no	Terms	Description
1	**API**	A set of rules and protocols that allows software components to communicate with each other.
2	**Backend**	The server-side part of the application that handles data processing, storage, and logic.
3	**Admin**	A system user with full administrative privileges (user management, content, resource management).
4	**EJS**	Embedded JavaScript Templating engine used for server-side view generation.
5	**JSON**	A lightweight data format used for exchanging information between the client and server.
6	**Prisma**	The Object-Relational Mapper (ORM) used to interact with the MySQL database.
7	**No Show**	A booking automatically marked unused when Check-In does not occur within the grace period.
8	**Restful.api**	An architectural style for designing networked applications that use HTTP requests for communication.
9	**TLS**	A cryptographic protocol ensures secure communication over a network.
10	**UI**	The visual elements and layout through which users interact with the system.

Table 22: Glossary

Appendix B: Analysis Models

ERD for FYP1 & FYP2

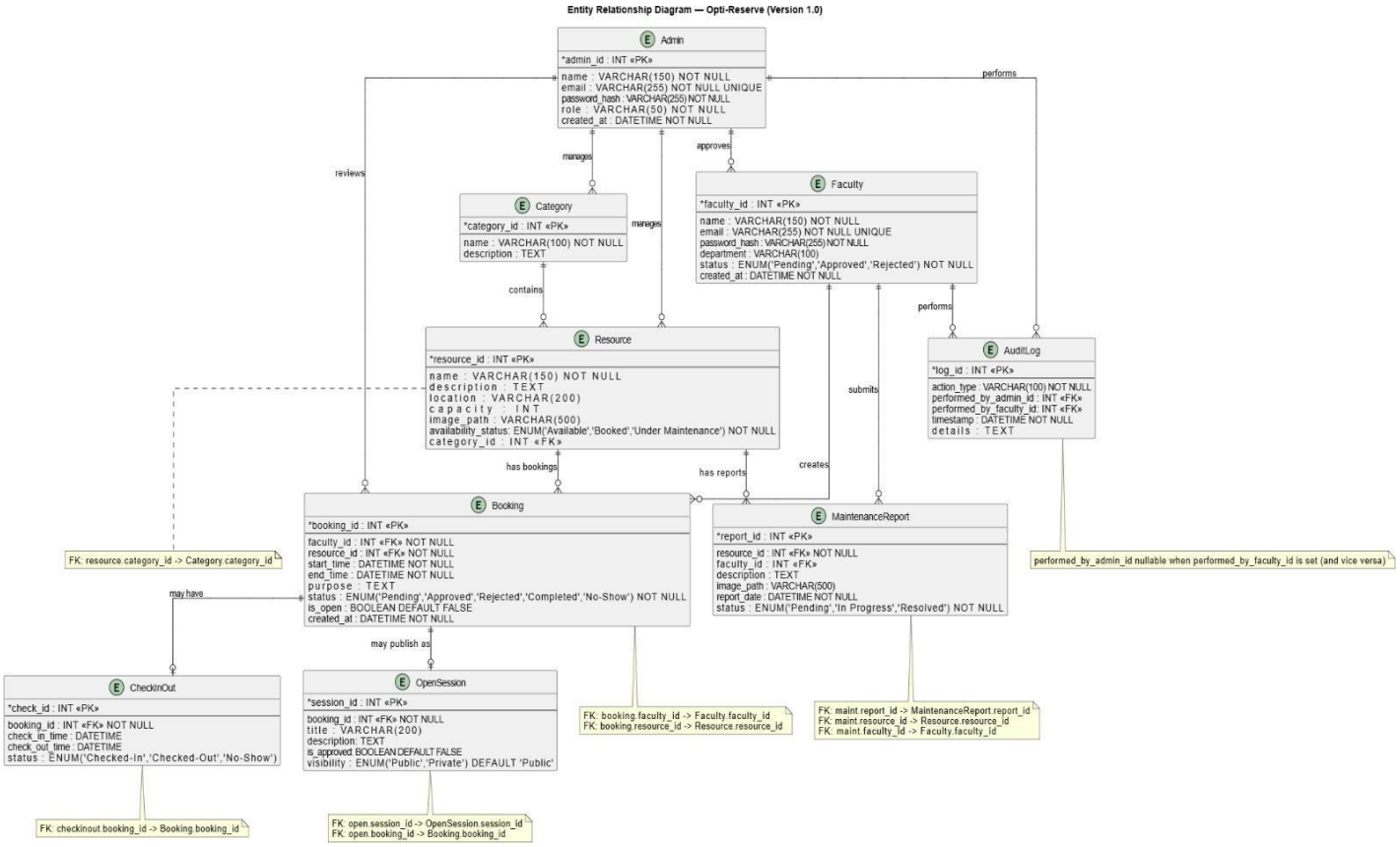


Figure 16: ERD of FYP 1&2

Appendix C: To Be Determined List

N/A

Software Design Specification

1. Introduction

1.1 Purpose of this Document

The purpose of this Software Design Specification (SDS) document is to record the Opti-Reserve system's architecture, components, interfaces, and detailed design. The purpose of this paper is to offer a technical blueprint that will help developers, testers, and stakeholders comprehend, implement, and manage the system. The SDS helps to ensure that the work of the project is aligned with the project's objectives as stated in the SRS and helps to communicate the project's work well throughout its lifecycle.

1.2 Scope of the development project

Opti-Reserve is a web-based facility and resource booking system for academic and organizational environments. This system allows administrators to control the facilities, make bookings, check usage, and report any maintenance issues, and faculty members to request bookings, check schedules and report maintenance issues. It features real-time conflict detection, role-based access control, check-in/out functionality, analytics and open session publishing for students. This SDS provides a detailed design of the implementable features.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
SDS	Software Design Specification: A document that outlines the system's technical architecture and design
SRS	Software Requirements Specification: Describes functional and non-functional requirements of the system.
API	Application Programming Interface: A set of protocols and endpoints enabling system interaction.
MVC	Model-View-Controller
UX	User Experience: The overall experience and satisfaction of a user when using the platform.
ORM	Object Relational Mapping
UI	User Interface
DB	Database
Admin	User with full system privileges
Faculty	Authorized staff member who can create bookings
No-Show	Booking marked unused due to no check-in
FYP	Final Year Project: Academic project demonstrating students' technical knowledge.
IDE	Integrated Development Environment: A tool for coding and debugging.

Table 23:Definitions, Acronyms, and Abbreviations

1.4 References

- Opti-Reserve Software Requirements Specification (SRS)
- IEEE Std 1016 – Software Design Specification
- Node.js Documentation
- Express.js Documentation

- Prisma ORM Documentation

1.5 Overview of this Document

This Software Design Specification (SDS) is organized in such a way that it makes it easy to understand the technical and functional aspects of the project. The document is divided into two sections:

- **Introduction:** Describes the goals and importance of the project and gives a layman's introduction to the project goals and importance.
- **System Overview:** Provides an overview of the system, its main components, and its purpose, helping readers understand the entire system.
- **Functional Specification:** Outlines some of the key features, workflows and behavior of the system as required by the user/business.
- **Design Details:** Technical design of the system, such as the system architecture diagram, module specification and interface design
- **Testing & Validation:** Describes the intended approach to making sure the system is correct, reliable and performs as intended.
- **Glossary and References:** Definitions of technical terms and lists of reference materials for aid in understanding.

The methodology used makes each section specifically targeted to the needs of the reader, so that the developer, tester, and stakeholder will be able to locate the information they need to their advantage.

2. System Architecture Description

2.1 Section Overview

This section explains the general structure of Opti-Reserve, the constraints of the system, how the data is organized, the organization of the programs and the alternative architectures that were considered during the design.

2.2 General Constraints

Opti-Reserve is designed under the conditions of the following constraints:

- **Hardware Constraints:**
    - Cloud or on-premises server capable of running Node.js
    - Sufficient memory and CPU to support concurrent bookings

- **Software Constraints:**
    - Node.js and Express.js backend
    - MySQL database with Prisma ORM
    - EJS templating engine

- Session-based authentication.

- **Performance Constraints:**
  - Process 100 or more active labs
- The time it takes to open a lab on the cluster is less than 10 seconds (on average).

Network Constraints:

- WebSocket might be needed for updates in real time.
- All public endpoint URLs secured with HTTPS.

Security Constraints

- Encrypted password storage
- Role-based authorization
- Audit logging for critical actions

2.3 Data Design

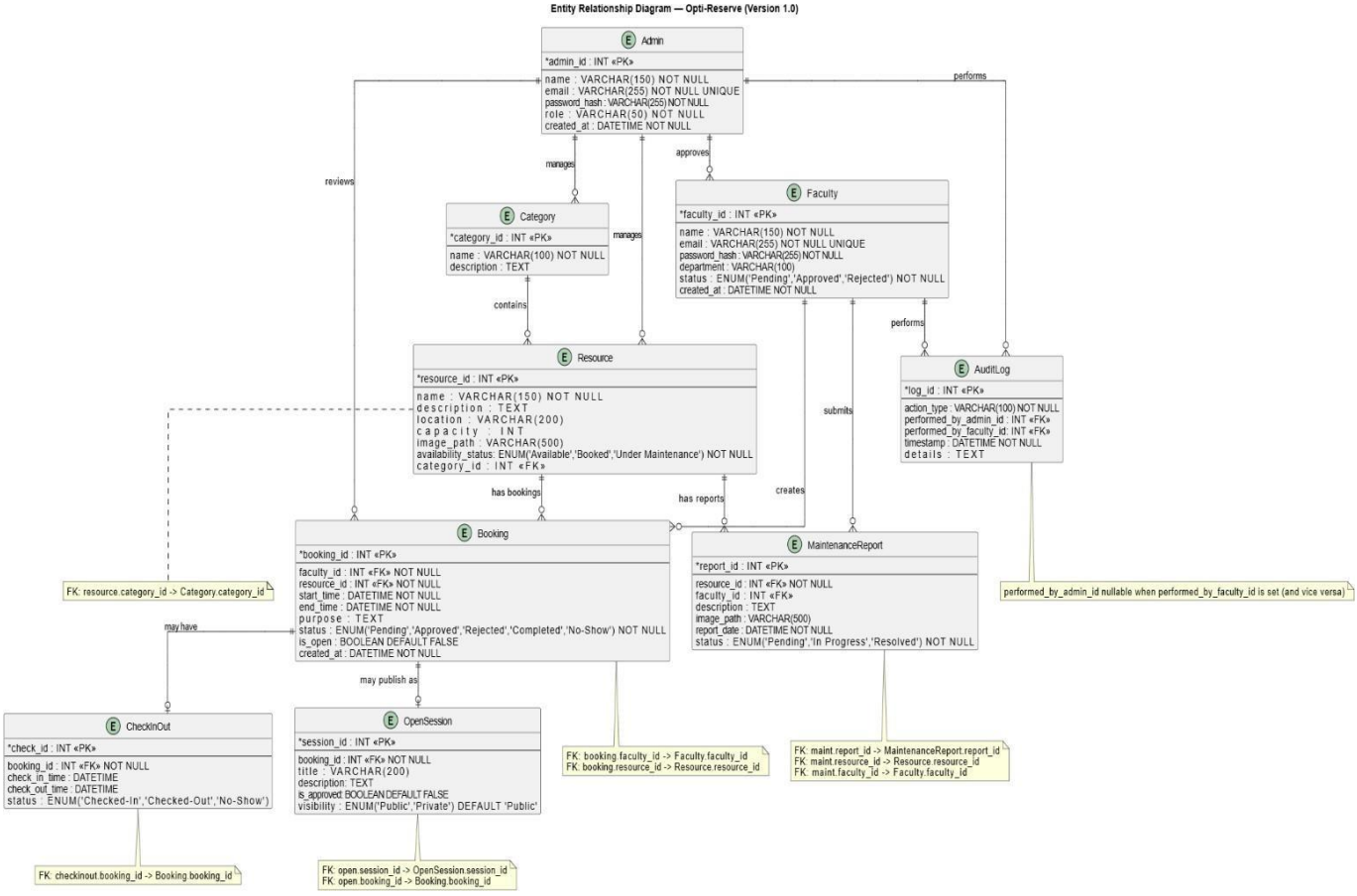


Figure 17: Data Design of FYP 1&2

2.4 Program Structure

1. **Authentication & Authorization**

    - Session-based login
    - Role validation (Admin / Faculty)

2. **Resource Management**

    - Add, Remove and Edit Categories and Resources.Create, Delete, Update categories and resources.

3. **Booking Management**

    - Booking requests
    - Conflict detection
    - Approval workflows

4. **Attendance Tracking**

    - Check-in / Check-out
    - No-show detection

5. **Maintenance Reporting**

    - Issue reporting
    - Resolution tracking

6. **Open Session Publication**
    - The public displays of approved sessions. Public recognition of approved sessions.
7. **Reporting & Analytics**

    - Usage statistics
    - No-show analytics
    - Audit logs

8. **Security & Audit**
    - **Authentication**
        - JWT-based API auth, refresh tokens, logout/invalidate sessions.
    - **Audit Logging**
        - All critical actions logged (login, submissions, admin actions) with timestamps and IPs.
    - **Container Hardening**
        - Run unprivileged containers, read-only filesystems where possible, sec comp/App Armor profiles.

2.5 Alternatives Considered

2.5.1 Manual / Spreadsheet-Based Scheduling

Reason for Rejection:
Lacks is available in real-time but is conflictual and institutional.

2.5.2 Microservices Architecture

Reason for Rejection:
Adds too much complexity and DevOps overhead to an academic system.

2.5.3 Layered MVC Architecture (Chosen)

Reason for Selection:
Clearly defines separation of concerns, helps to maintain, is scalable and fits well current web based institutional systems.

3. Detailed Description of Components

3.1 Section Overview

This section describes in detail the primary components in the Opti-Reserve system such as their purpose, functionality, relationships with other components, functional interfaces, and internal processing logic.

3.2 Component and Detail

3.2.1 User Registration Component

Attribute	Description
Identification	User Registration
Type	Module
Purpose	Allows faculty to register accounts
Function	Stores user data with pending approval
Dependencies	Database, Email (optional)
Interfaces	Registration UI
Processing	Validate input, hash password, store record
Data	Name, email, password hash, status

Table 24: User Registration Component Table

3.2.2 Authentication Component

Attribute	Description
Identification	Login
Type	Module
Purpose	Authenticate users

Function	Validate credentials and create session
Dependencies	User DB, bcrypt
Interfaces	Login UI
Processing	Verify password and role
Data	Session metadata

Table 25: Authentication Component Table

3.2.3 Booking Management Component

Attribute	Description
Identification	Login
Type	Module
Purpose	Authenticate users
Function	Validate credentials and create session
Dependencies	User DB, bcrypt
Interfaces	Login UI
Processing	Verify password and role
Data	Session metadata

Table 26: Booking Component Table

3.2.4 Conflict Detection Component

Attribute	Description
Identification	Conflict Detection
Type	Control Module
Purpose	Prevent overlapping bookings
Function	Time-slot validation
Dependencies	Booking DB, bcrypt
Interfaces	Login UI
Processing	Compare requested slots
Data	Session metadata

Table 27: Conflict Detection Component Table

3.2.5 Check-In / Check-Out Component

Attribute	Description
Identification	Attendance Tracking
Type	Module
Purpose	Track actual usage
Function	Record timestamps
Dependencies	Booking DB
Interfaces	Login UI
Processing	No-show logic
Data	Session metadata

Table 28: Check In/ Check Out Component Table

3.2.6 Maintenance Reporting Component

Attribute	Description
Identification	Maintenance Reporting
Type	Module
Purpose	Report and track issues
Function	Issue creation and resolution
Dependencies	Resource DB
Interfaces	Report UI
Data	Session metadata

Table 29: Maintenance Reporting Component Table

3.2.7 Upload Attachment

Attribute	Description
Identification	Login
Type	Module
Purpose	Authenticate users
Function	Validate credentials and create session
Dependencies	User DB, bcrypt
Interfaces	Login UI
Processing	Verify password and role
Data	Session metadata

Table 30: Upload Attachment Component Table

3.2.8 Resource Catalog

Attribute	Description
Identification	Login
Type	Module
Purpose	Authenticate users
Function	Validate credentials and create session
Dependencies	User DB, bcrypt
Interfaces	Login UI
Processing	Verify password and role
Data	Session metadata

Table 31: Resource Catalog Component Table

3.2.9 Audit Log

Identification	Component "**Audit Log**", immutable activity logger.
Type	Module: System-level logging.

Purpose	Record security-sensitive actions for compliance and debugging.
Function	• Record security-sensitive actions for compliance and debugging. • Provide queryable logs for admins.
Subordinates	• Logger. • Log Query API.
Dependencies	DB or append-only log store.
Interfaces	GET /audit-logs (admin), internal logging calls.
User Interface	Admin log viewer with filters.
Resources	Append-only storage, retention management.
Processing	Write event entries with timestamp, user, action, context.
Data	Log-id, user-id, action, resource, Ip, timestamp.

Table 32: Audit Log Component Table

3.2.10 User Settings / Profile

Identification	Component "**User Settings**", profile & preferences.
Type	Module: Profile management.
Purpose	Allow users to update name, avatar, password, notification preferences
Function	• Edit profile, change password, toggle 2FA.
Subordinates	• Photo Uploader. • Password Change Handler.
Dependencies	Auth, storage, DB.
Interfaces	User settings page and API.
User Interface	Profile form, save button, change password flow.
Resources	Storage for avatars, CPU for hashing.
Processing	• Validate updates. • Hash new password if present • Save changes.
Data	user profile fields, preferences.

Table 33: User setting Profile Component Table

4. User Interface Design

4.1 Section Overview

This section provides a brief overview of the user interface design used in Opti-Reserve It outlines the key design principles, main UI components, and the structure of the interface. The section includes:

- **Interface Design Rules:** Describes the core UI/UX guidelines followed, focusing on simplicity, usability, and responsive.
- **GUI Components:** Lists the primary interface elements used throughout the platform, including reusable components such as navigation bars, cards, forms, and admin tools.

4.2 Interface Design Rules

- Simple and consistent layout

- Responsive design for desktop and mobile
- Role-based dashboards
- Clear feedback and error messages.

4.3 GUI Components

- Login & Registration Screens
- Admin Dashboard
- Faculty Dashboard
- Resource Listing
- Booking Calendar View
- Reports & Analytics Panels

4.4 Detailed Description

4.4.1 Sign Up

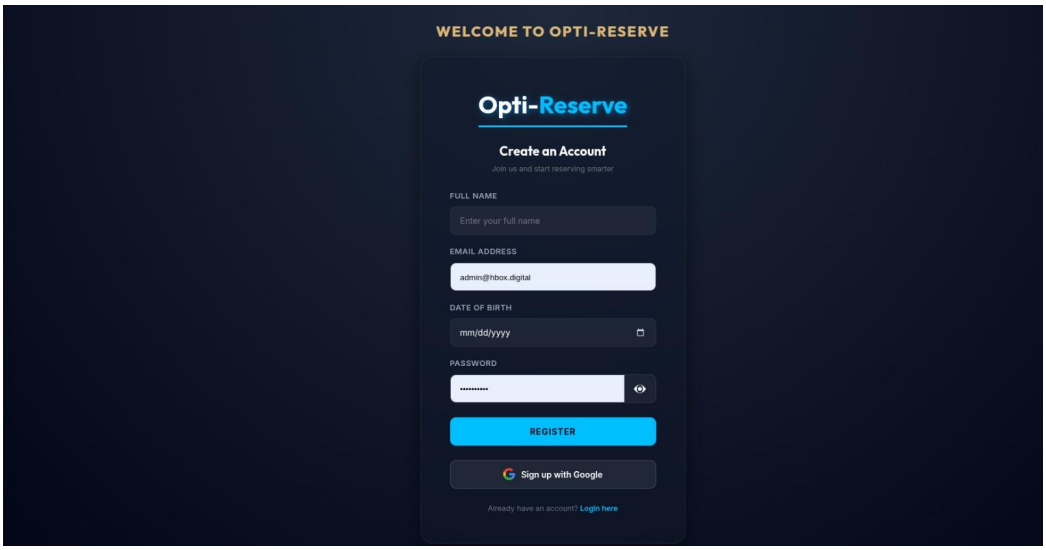


Figure 18 Sign Up Screen

4.4.2 Login

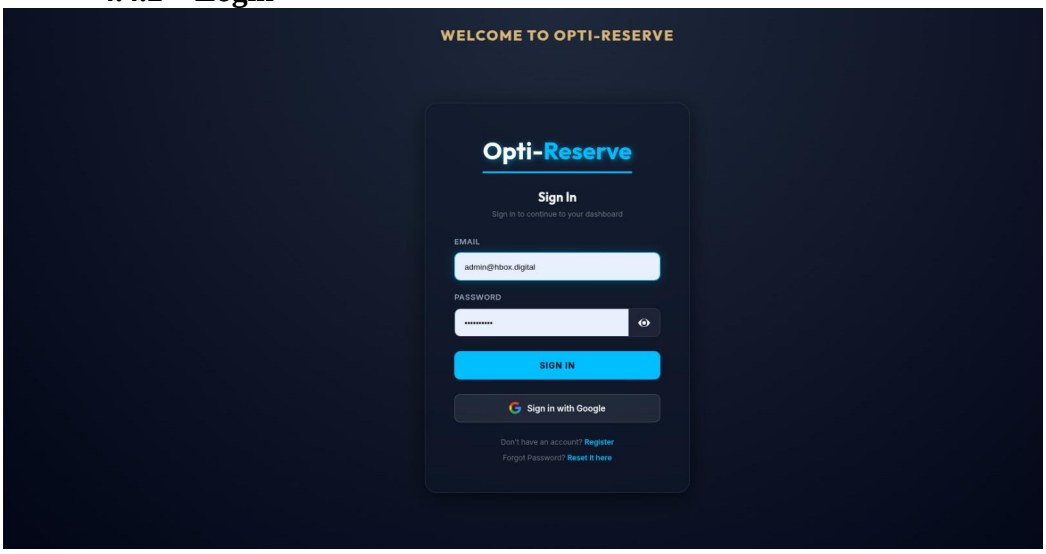


Figure 19 Login Screen

4.4.3 Forgotten Password

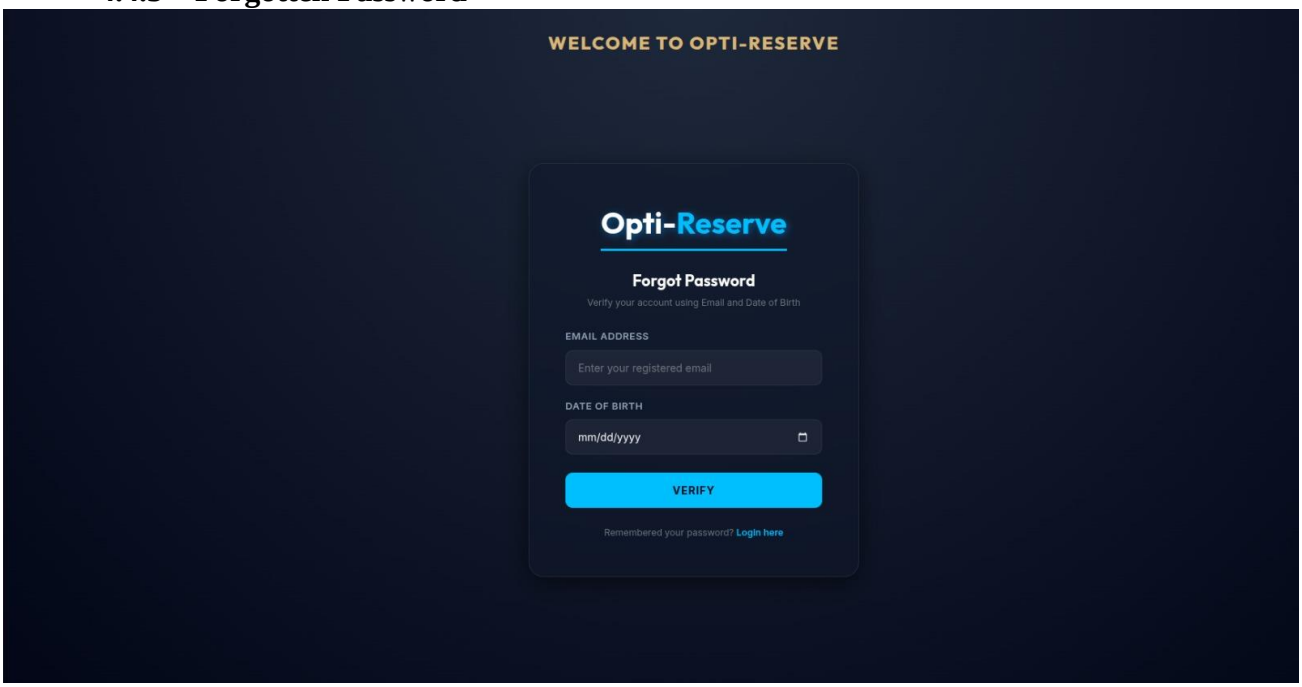


Figure 20 Forget Password Screen

4.4.4 About

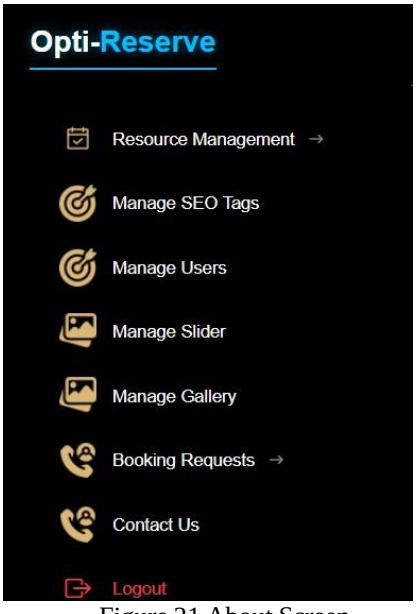


Figure 21 About Screen

4.4.5 Admin Board

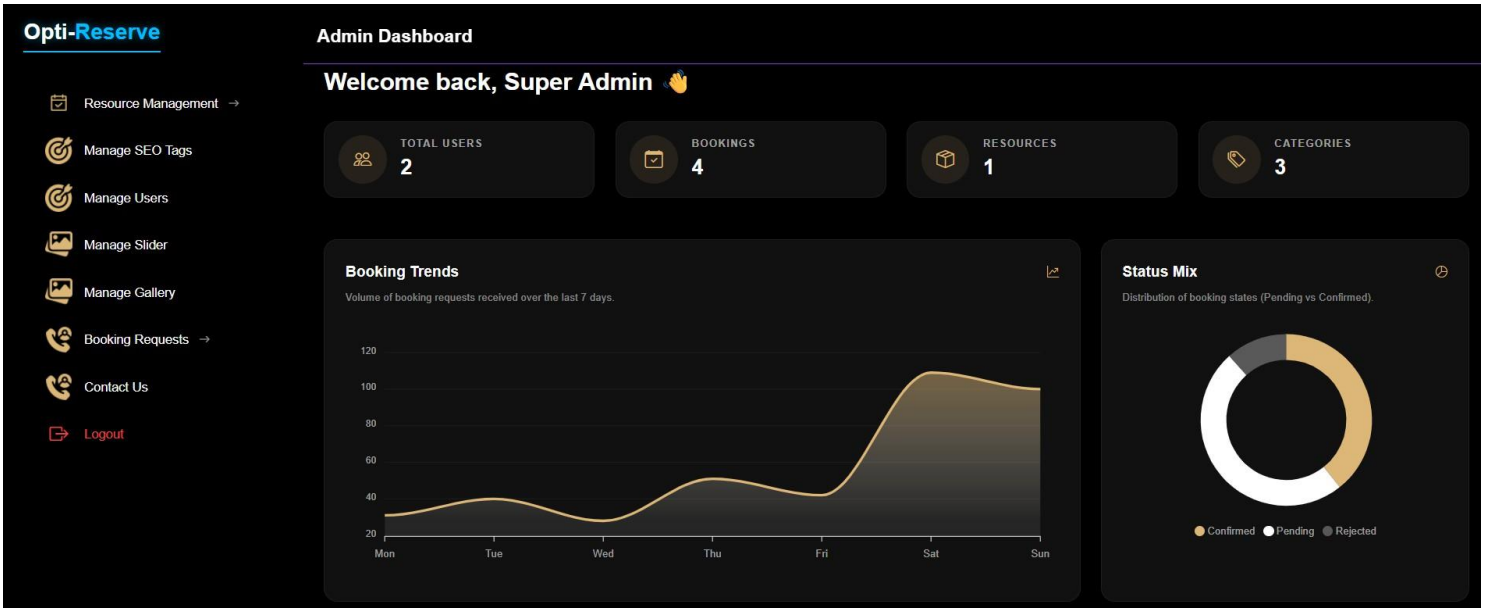


Figure 22 Admin Board Screen

4.4.6 Users

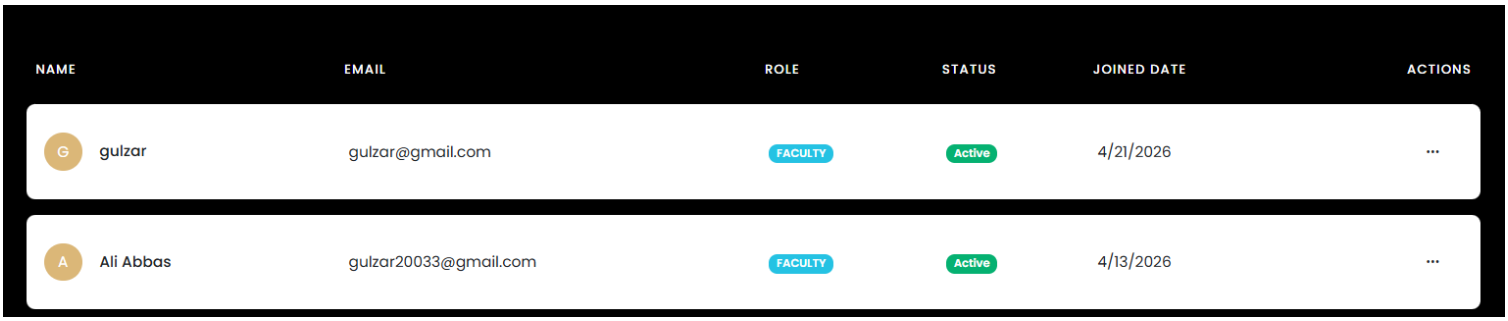


Figure 23 Users Screen

4.4.7 Schedule

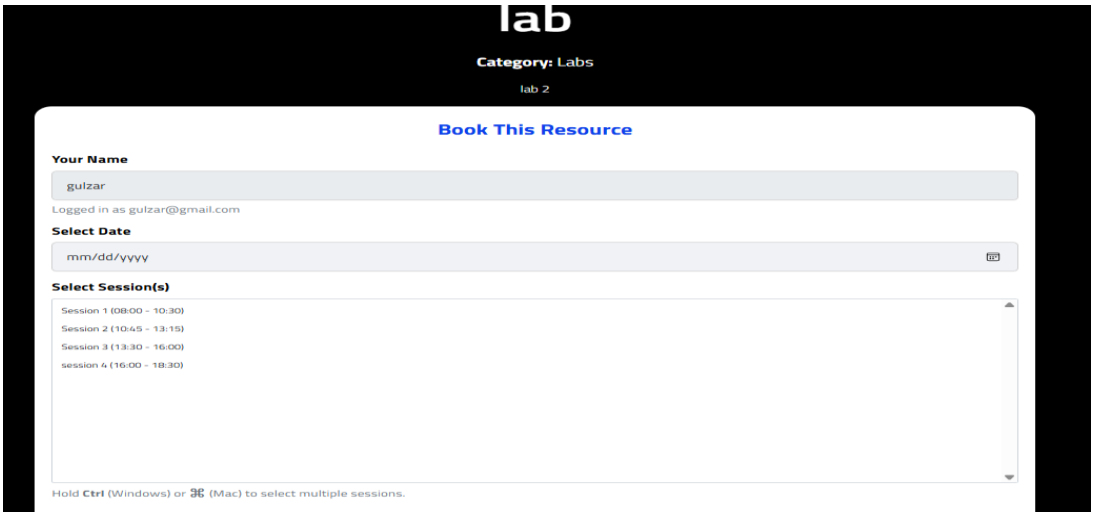


Figure 24 Schedule Screen

4.4.8 User Interfaces

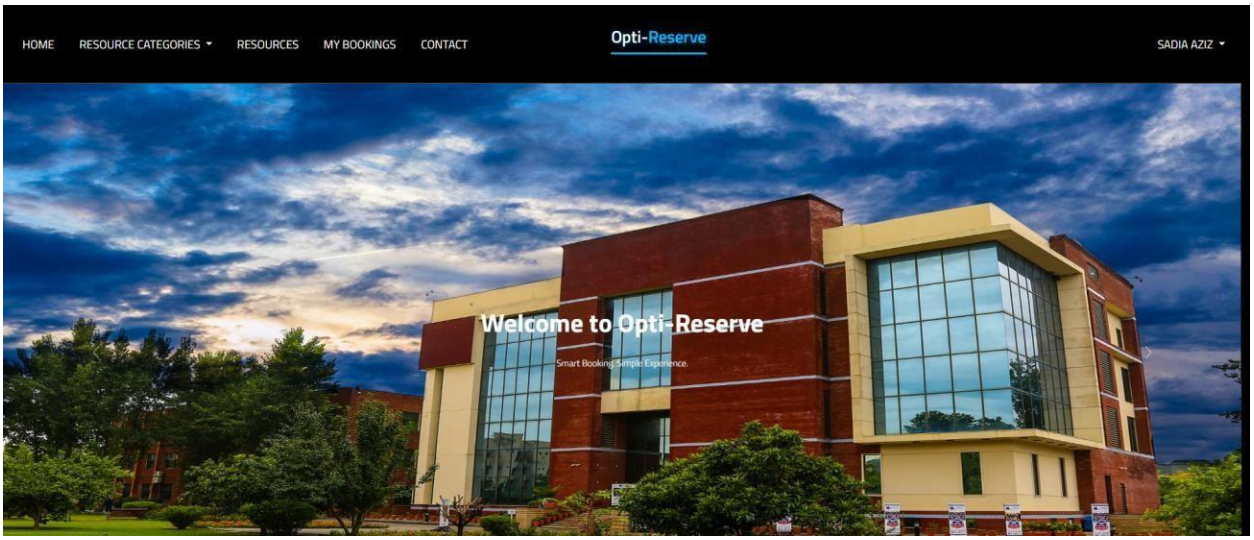


Figure 25 User Interfaces Screen

4.4.9 Email Notify.

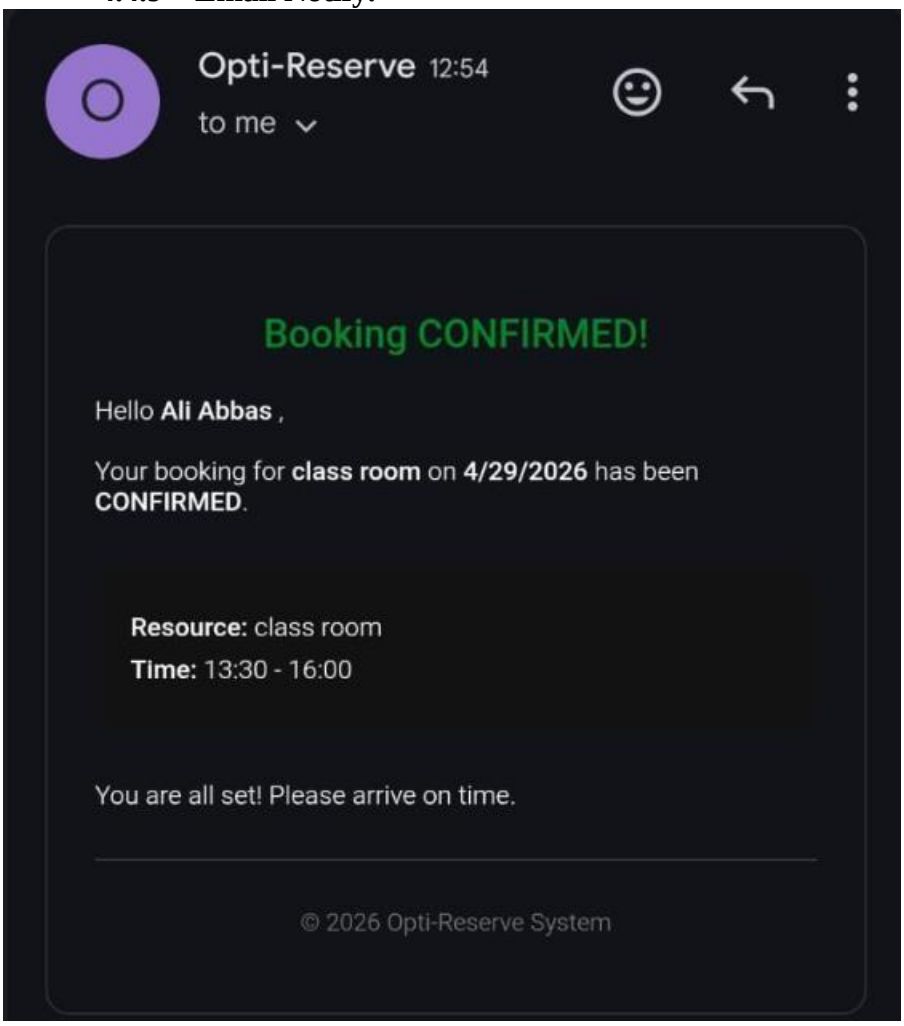


Figure 26 Account Settings Screen

5. Reuse and Relationship to Other Products

- **Role in Product Design:** By utilizing well-known and extensively used frameworks like Node.js, Express.js, Prisma ORM, and EJS, Opti-Reserve's architecture integrates reuse. To guarantee consistency, scalability, and maintainability, common architectural patterns such as the Model-View-Controller (MVC) structure and role-based access control are applied to every module. Standardized dashboards, forms, tables, and approval workflows are examples of reusable design patterns that are used throughout the system.

- **Role in Product Implementation:** Opti-Reserve re-uses the essential backend modules such authentication and session management, role authorisation logic, booking conflict detection, approval workflows, and notification handling during implementation. The reporting, open session publication, facility booking, and check-in/check-out modules are just a few of the many functionalities that make use of these shared services to eliminate code duplication and guarantee consistent system behaviour.
- **Reusable UI Components:** The system utilises many user interface components in the faculty and admin modules such as navigation menus, dashboards, booking forms, resource tables, modal dialogues and alert messages. These reusable UI elements boost development productivity, preserve visual consistency, and improve the overall usability across multiple system interfaces.
- **Motivation for Non-Reuse:** Some parts were deliberately built as custom modules when reuse was not appropriate. This includes booking conflict detection logic, no-show handling techniques and audit logging features that all require domain-specific processing and tight data integrity. These components especially benefit from specialised implementation as generic solutions may compromise system accuracy, performance or institutional requirements.

6. Design Decision and Tradeoffs

the design of Opti-Reserve features reuse through the use of existing and widely used frameworks like Node.js, Express.js, Prisma ORM and EJS. All the modules are consistent, scalable and maintainable, in a way that the common architectural patterns are reused, such as the Model-View-Controller (MVC) structure and the role-based access control. A wide range of reusable design patterns, like common dashboards, forms, tables, and approval processes are used across the system.

- **Simplicity vs Feature Richness:** Reserve is focused on simplicity and the user experience and has a limited number of features needed for the bookings and management. No unnecessary complexity was added to make it easy for faculty and administrators to use, though there were other advanced features which could be added in the future.
- **Session-Based Authentication vs Token-Based Authentication:** It was decided to opt-in for Session Based Authentication with express-session for simple implementation and compatibility with institutional setting. Even though token-based authentication is more scalable, sessions were chosen because they had less implementation complexity and could

handle the expected user load.

- **Data Consistency vs Performance: To** ensure the accuracy of bookings and avoid overlapping bookings, a relational database design was implemented with strict conflict detection logic was implemented. This way, there is a slight overhead during processing, but it assures reliable scheduling and data integrity.
- **Scalability vs Architectural Complexity:** Due to the need to minimize complexity, while providing scalability, a layered architecture of the MVC approach was chosen. Although microservices provide more scalability, they were not seen as being required for an academic and institutional booking system.
- **Web-Based Accessibility:** The system is made web accessible that can be accessed with standard browsers. Not included to save development time has been native mobile applications, but responsive design ensures it works on desktops and on mobile.
- **Abandoned Ideas:** Other parts like AI-powered tour suggestions, rooms occupancy on real time with sensors, fancy graphical analytics dashboards were thought of but not taken on due to their complexity, hardware dependency and scope limitation.

7. Pseudocode for Components

7.1 User Authentication Logic

```
IF login request is received THEN
    Retrieve user record using email
    IF user exists AND password matches hashed password THEN
        IF user status is approved THEN
            Create session
            Redirect user to dashboard based on role
        ELSE
            Display "Account Pending Approval"
        ENDIF
    ELSE
        Display "Invalid Credentials"
    ENDIF
ENDIF
```

7.2 Facility Booking Request Logic

```
IF faculty submits booking request THEN
    Validate resource, date, and time
    Check for booking conflicts
```

```
    IF no conflict exists THEN
   Save booking with status = Pending Approval
   Notify admin
    ELSE
       Display conflict error message
    ENDIF
ENDIF
```

7.3 Booking Conflict Detection Logic

```
FOR each existing booking of selected resource
    IF requested time overlaps existing booking time THEN
       RETURN conflict detected
    ENDIF
ENDFOR
RETURN no conflict
```

7.4 Booking Approval Logic (Admin)

```
IF admin reviews booking request THEN
    IF admin approves booking THEN
       Update booking status to Approved
       Notify faculty
    ELSE
       Update booking status to Rejected
       Notify faculty with reason
    ENDIF
ENDIF
```

7.5 Check-In / Check-Out Logic

```
IF booking start time is reached THEN
   IF faculty checks in within allowed time window THEN
      Record check-in timestamp
   ELSE
      Mark booking as No-Show
   ENDIF
ENDIF

IF faculty checks out THEN
   Record check-out timestamp
ENDIF
```

7.6 Maintenance Issue Reporting Logic

```
IF faculty submits maintenance report THEN
   Validate issue details
   Save report with status = Open
   Notify admin
ENDIF

IF admin marks issue as resolved THEN
   Update report status to Resolved
ENDIF
```

7.7 Open Session Publication Logic

```
IF faculty marks booking as Open Session THEN
   Send request for admin approval
```

IF admin approves THEN

Publish session on public page

ENDIF

ENDIF

7.8 Reporting and Analytics Logic

Admin requests reports

Fetch booking, usage, and no-show data

Aggregate statistics

Display results on analytics dashboard.

8. Design Diagrams

8.1 Class Diagram

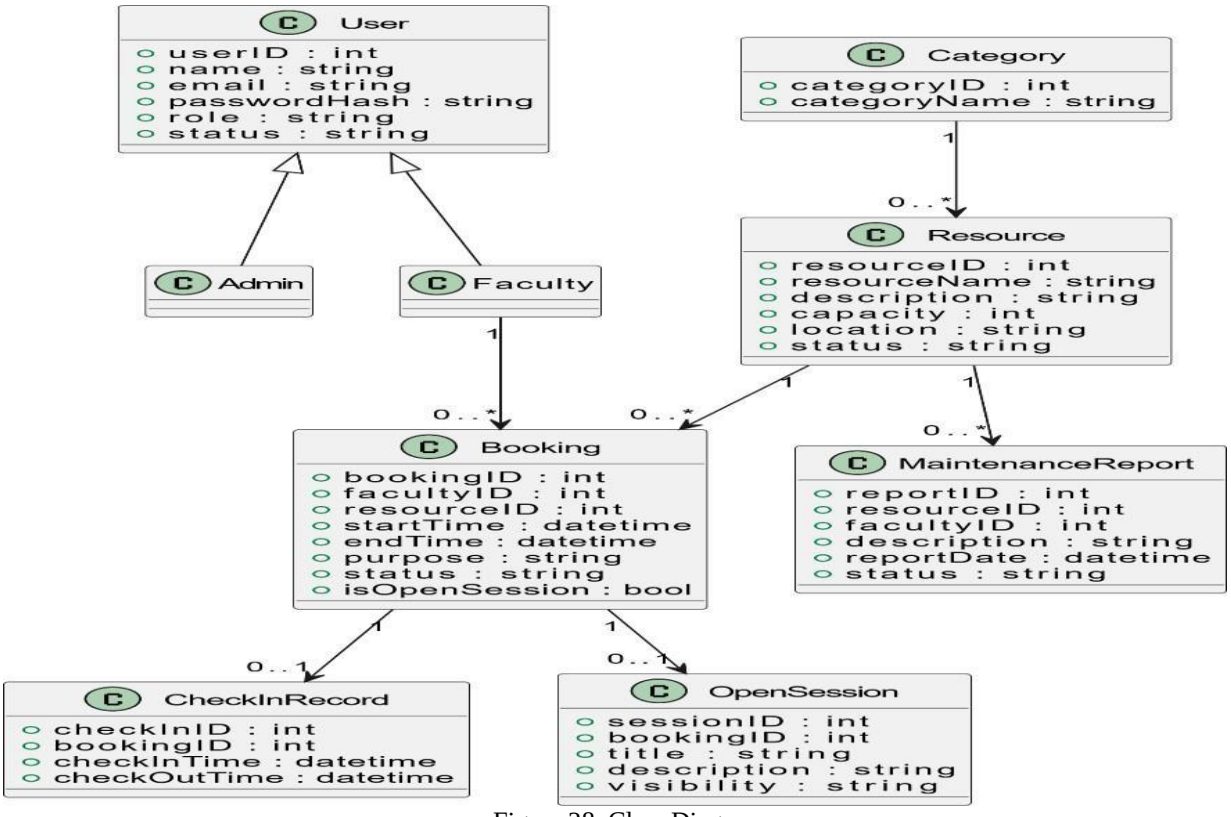


Figure 28: Class Diagram

8.2 Object Diagram

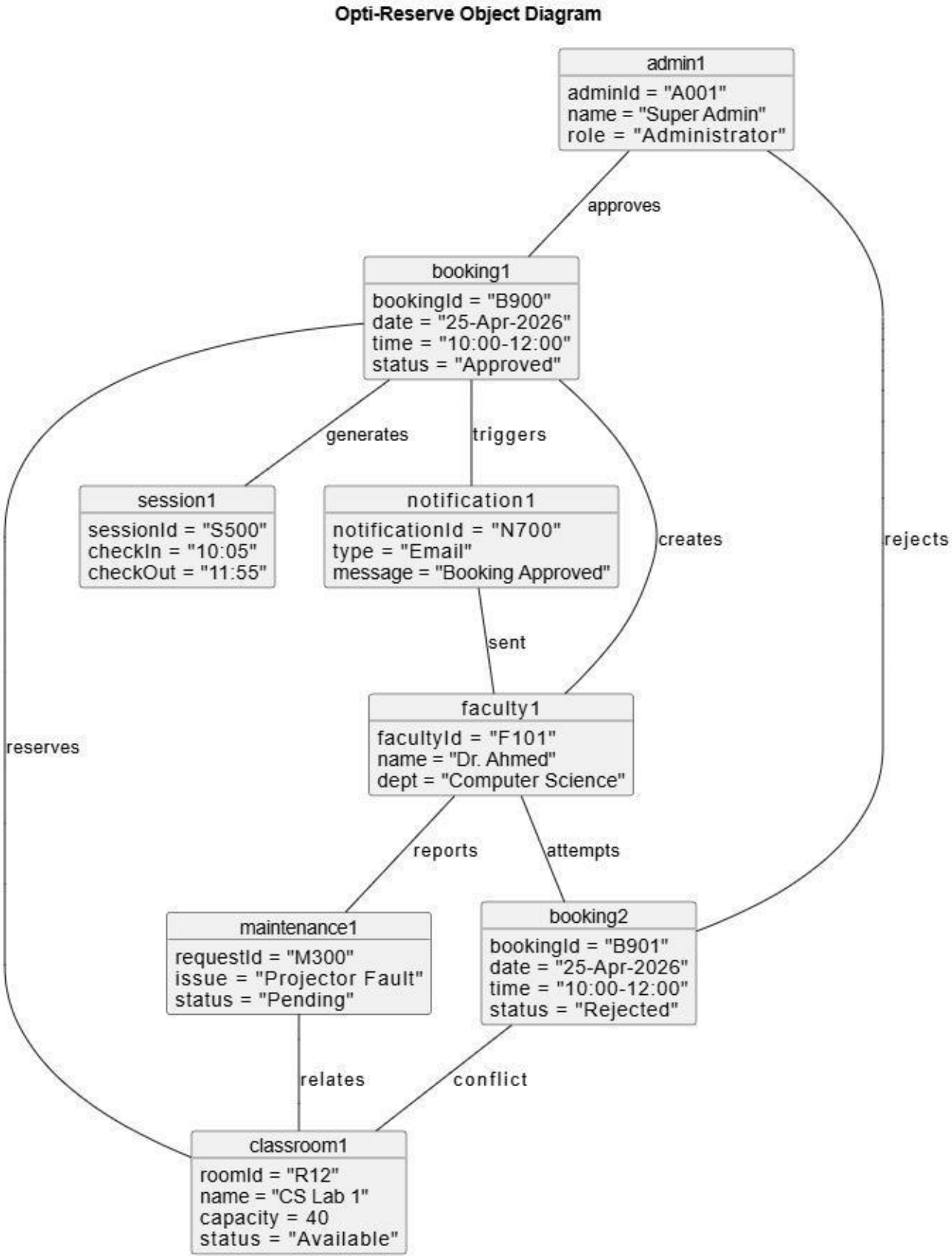


Figure 29: Object Diagram

8.3 State Chart Diagram

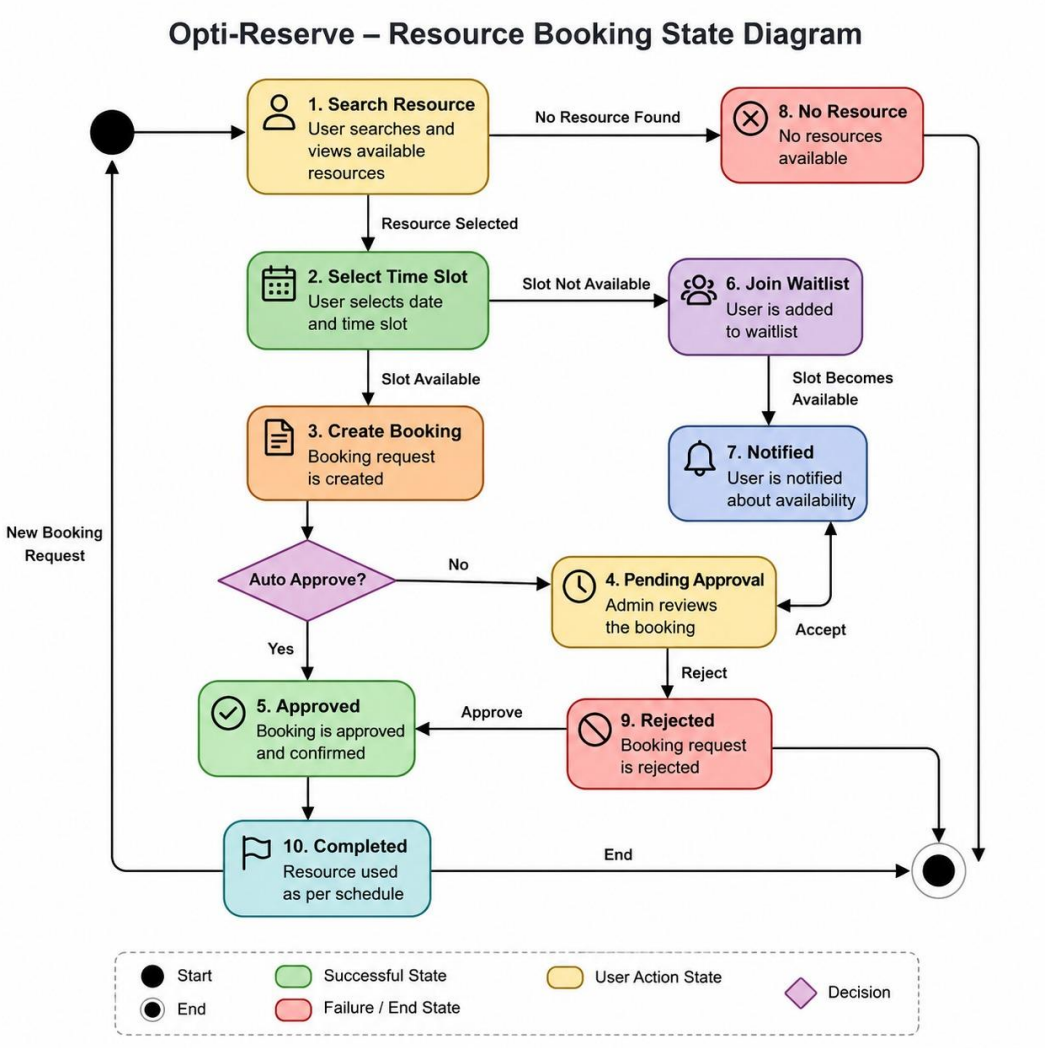


Figure 30: Faculty State Chart

8.4 Activity Diagrams

8.4.1 Login

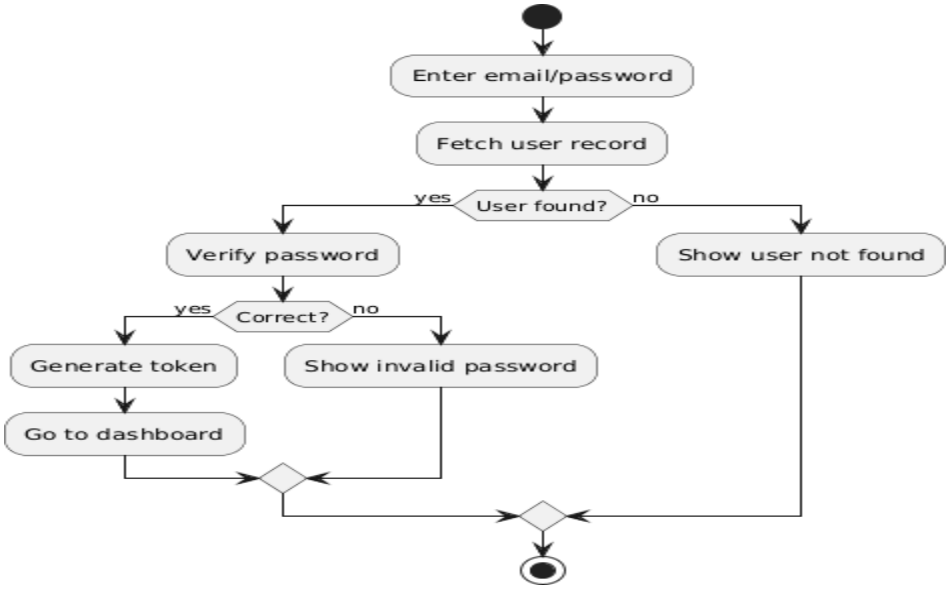


Figure 32: Login Activity Diagram

8.4.2 User Registration & Approval

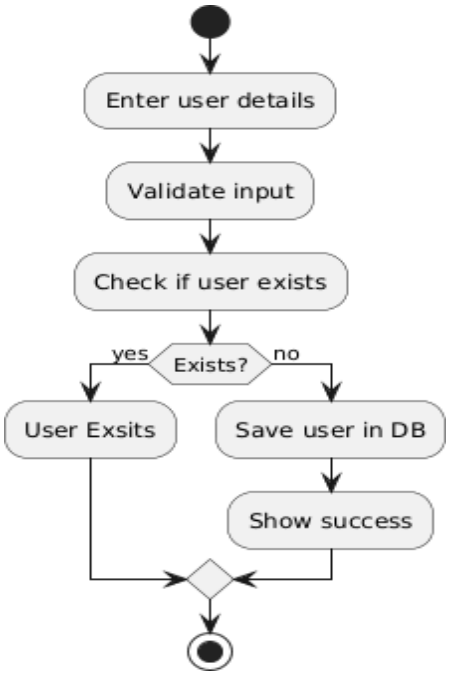


Figure 33: User Registration Approval Activity Diagram

8.4.3 User Login

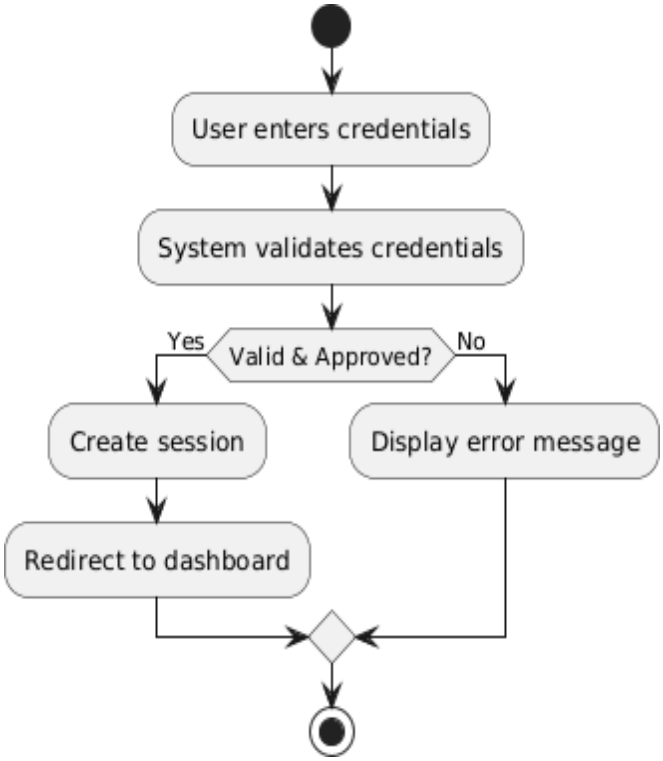


Figure 34: User login Activity Diagram

8.4.4 Admin Resource & Category Management

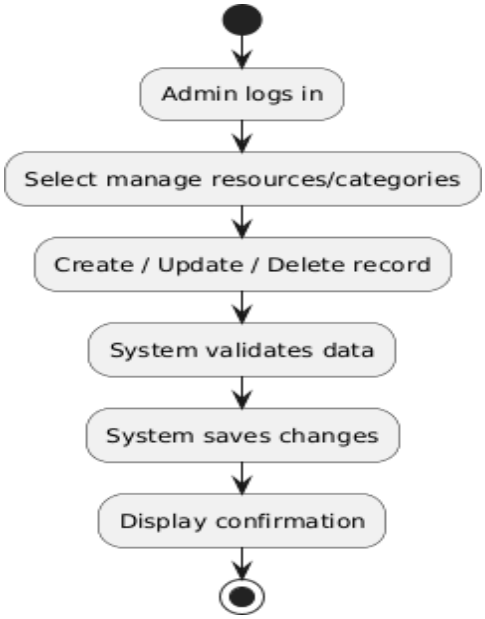


Figure 35: Admin Resource & Category Activity Diagram

8.4.5 Booking Request Creation

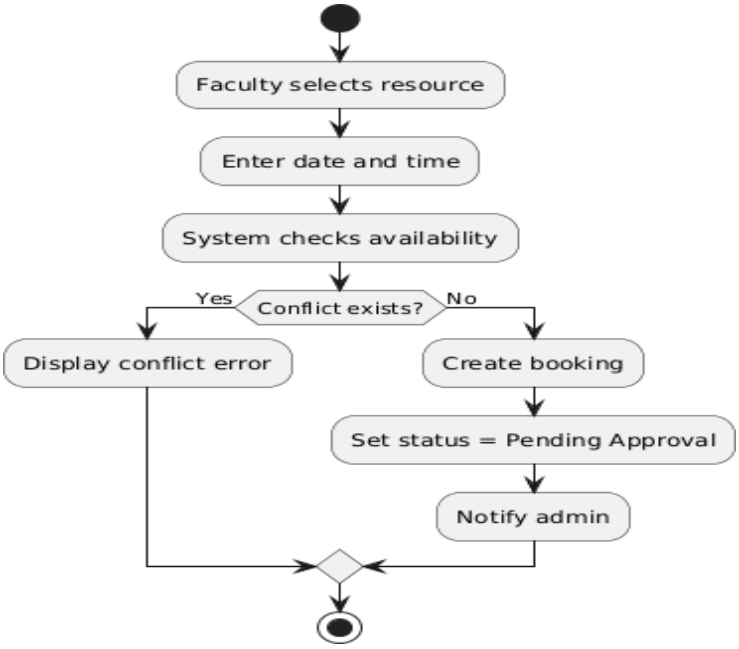


Figure 36: Booking Request Activity Diagram

8.4.6 Booking Approval (Admin)

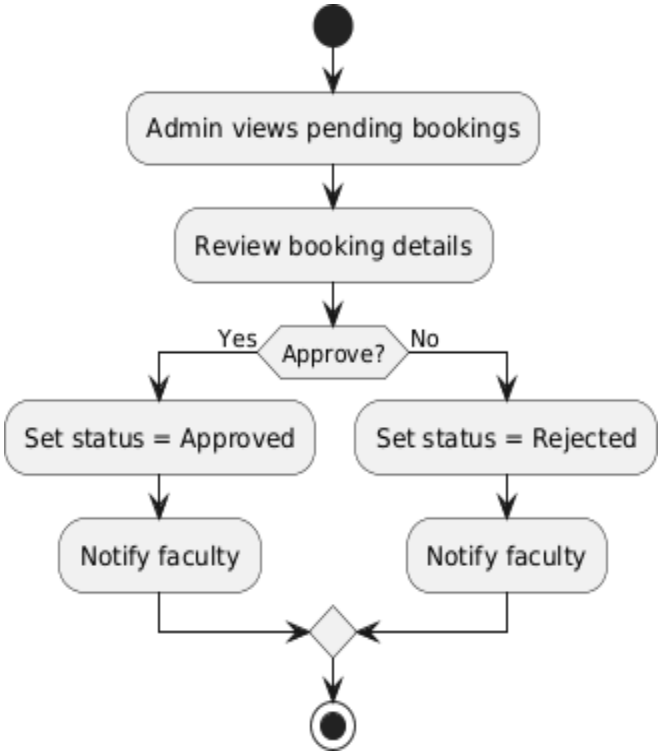


Figure 37: Booking Approval Activity Diagram

8.4.7 Check-In / Check-Out & No-Show

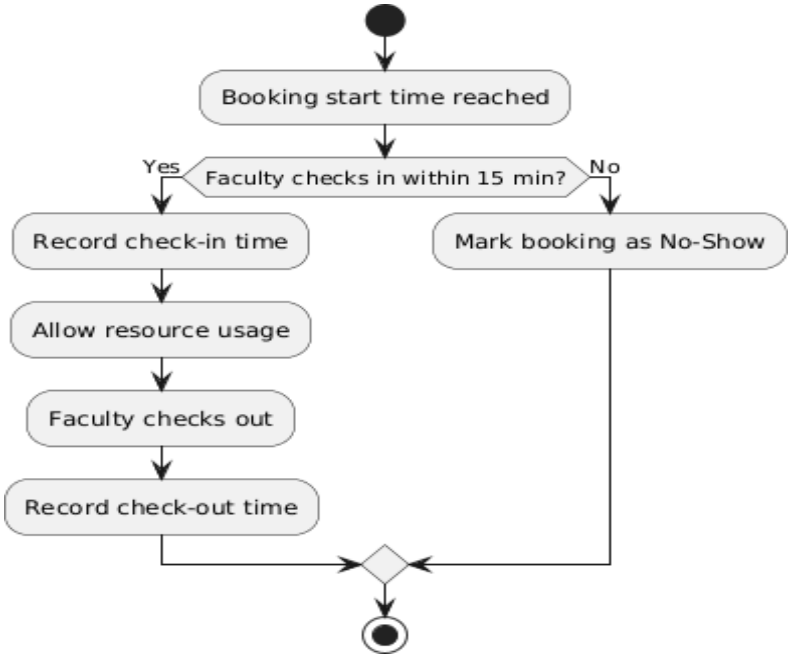


Figure 38: Check In/Check Out Activity Diagram

8.4.8 Maintenance Reporting



Figure 39: Maintenance Activity Diagram

8.4.9 Open Session Publication

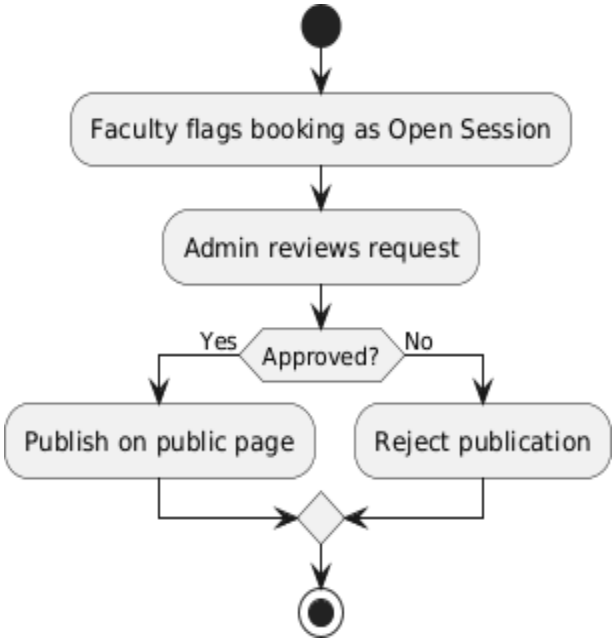


Figure 40: OS Publication Activity Diagram

8.4.10 Public Viewing of Open Sessions

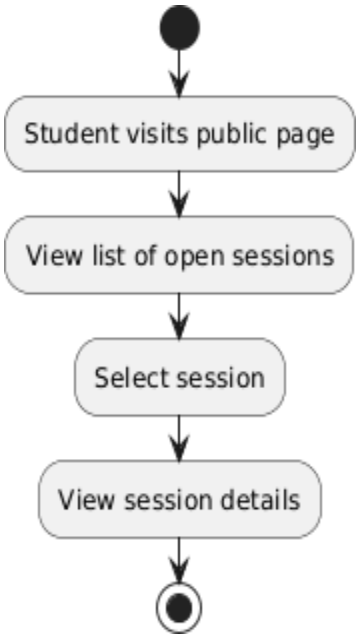


Figure 41: Open session activity Diagram

8.4.11 Reporting & Analytics

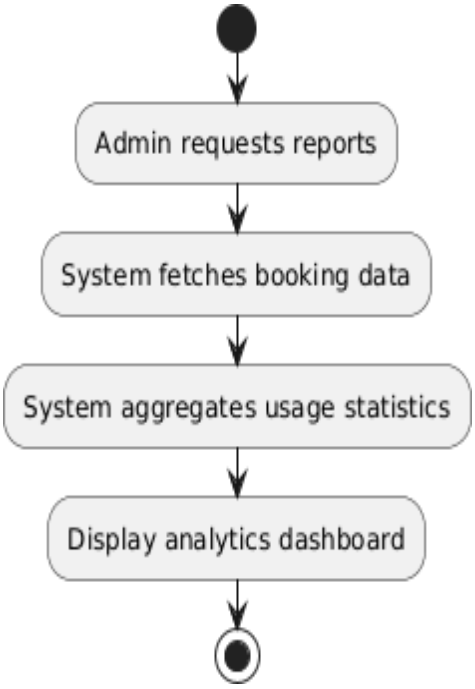


Figure 42: Report Analytics Activity Diagram

8.5 Sequence Diagram

8.5.1 Faculty Registration & Approval

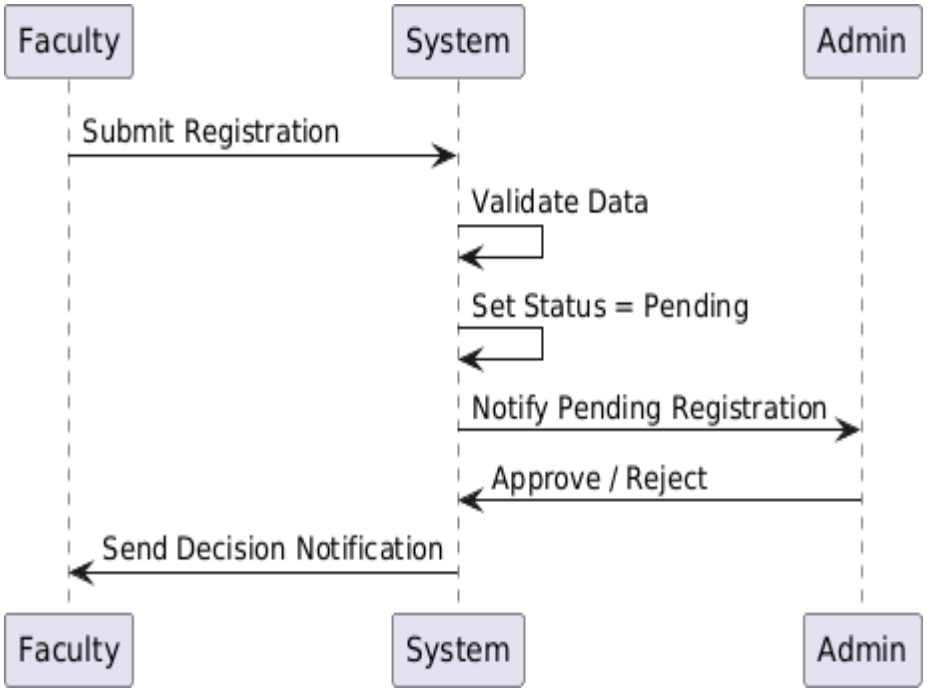


Figure 43: Faculty registration Sequence Diagram

8.5.2 User Login

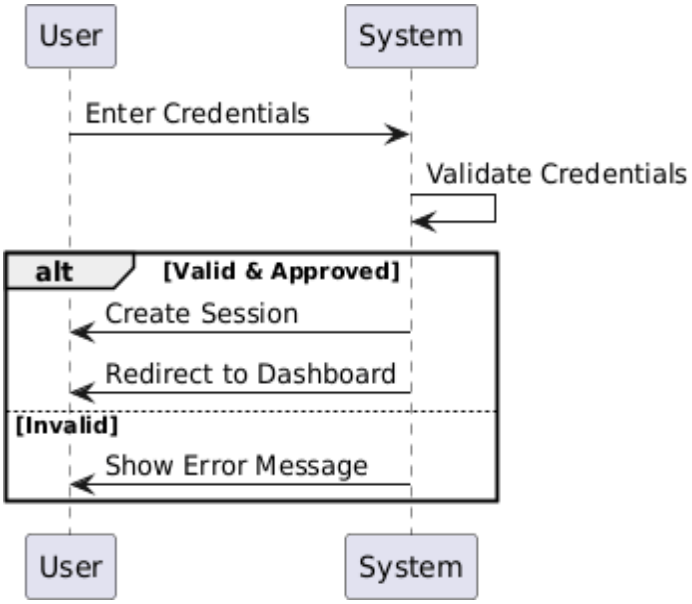


Figure 44: User Login Sequence Diagram

8.5.3 Booking Request Creation

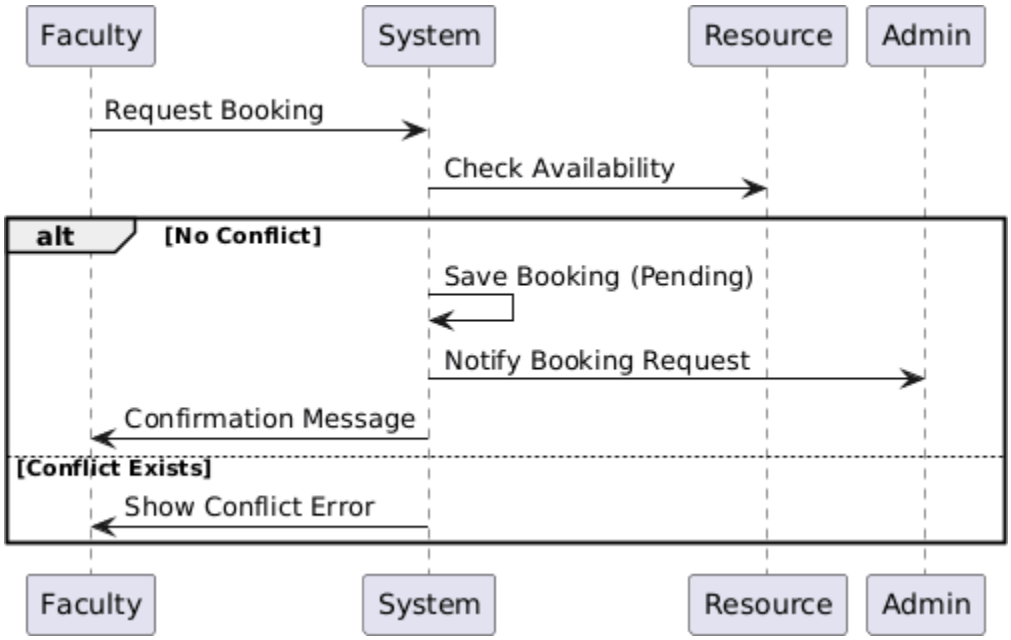


Figure 45: Booking Request Sequence Diagram

8.5.4 Check-In / Check-Out & No-Show

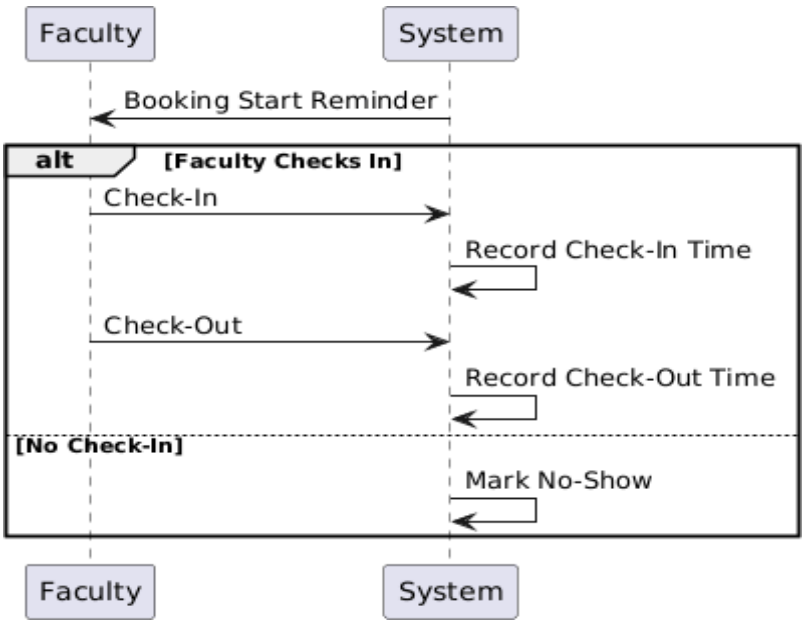


Figure 46: Check In/Check Out Sequence Diagram

8.5.5 Login

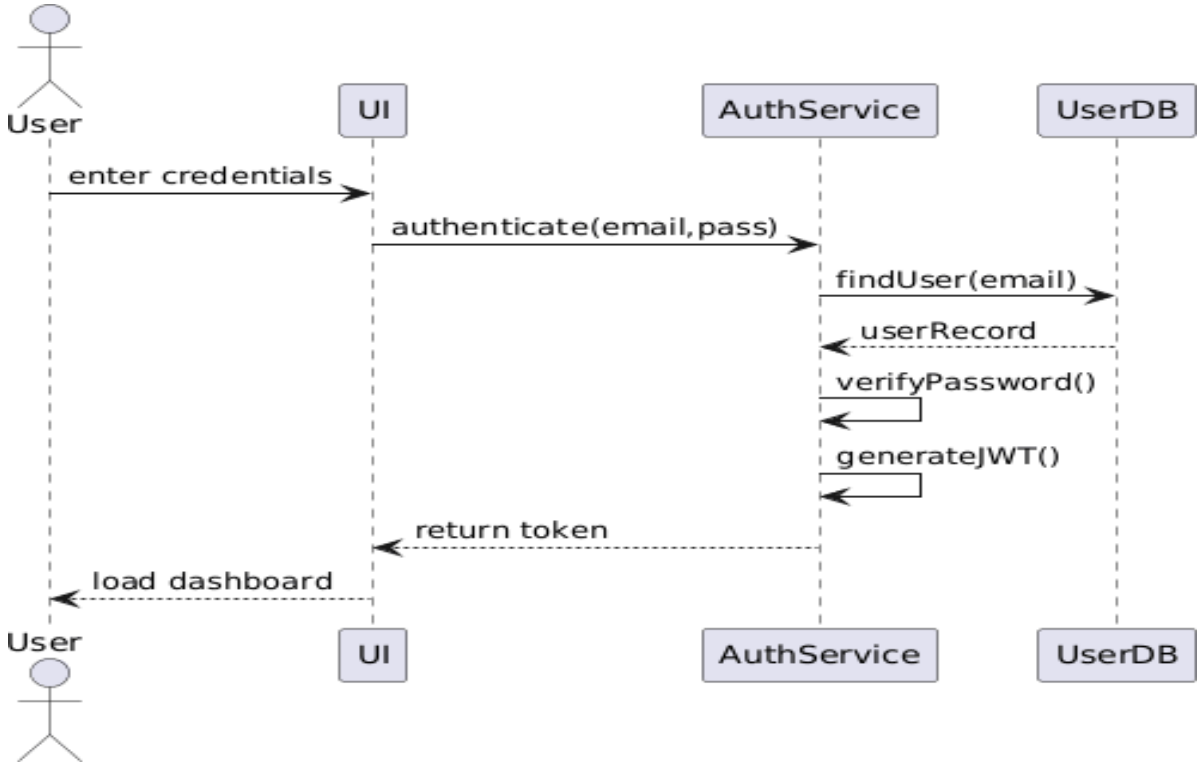


Figure 47: Login Sequence Diagram

8.6 Use Case Diagrams

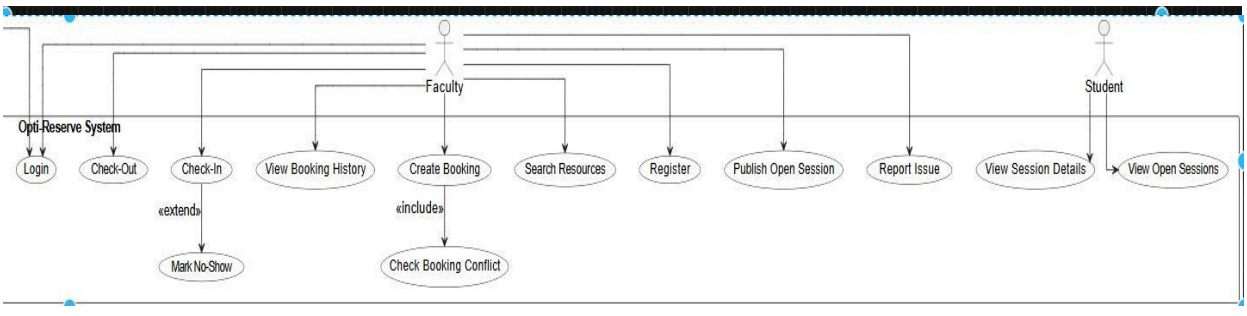


Figure 48: Use Case Diagram

8.7 Component Diagram

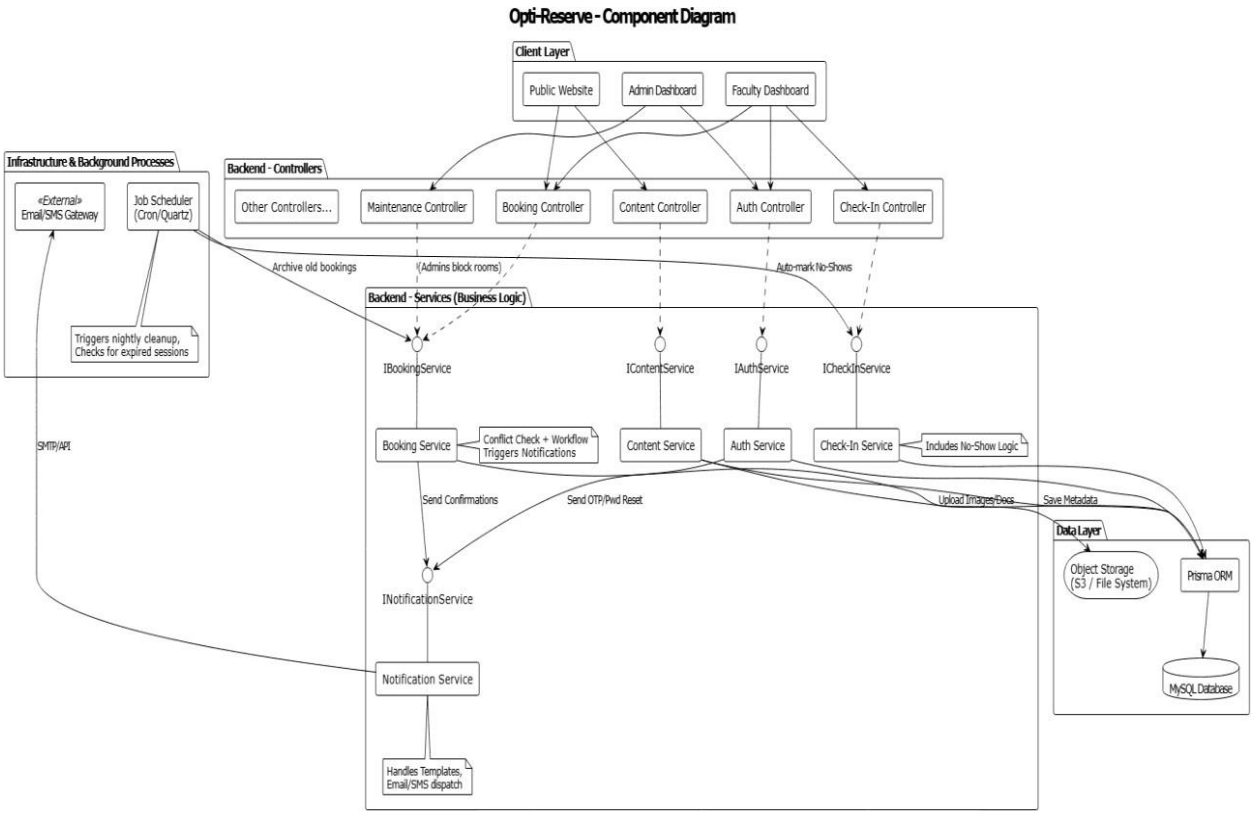


Figure 49 Component Diagram

8.8 Deployment Diagram

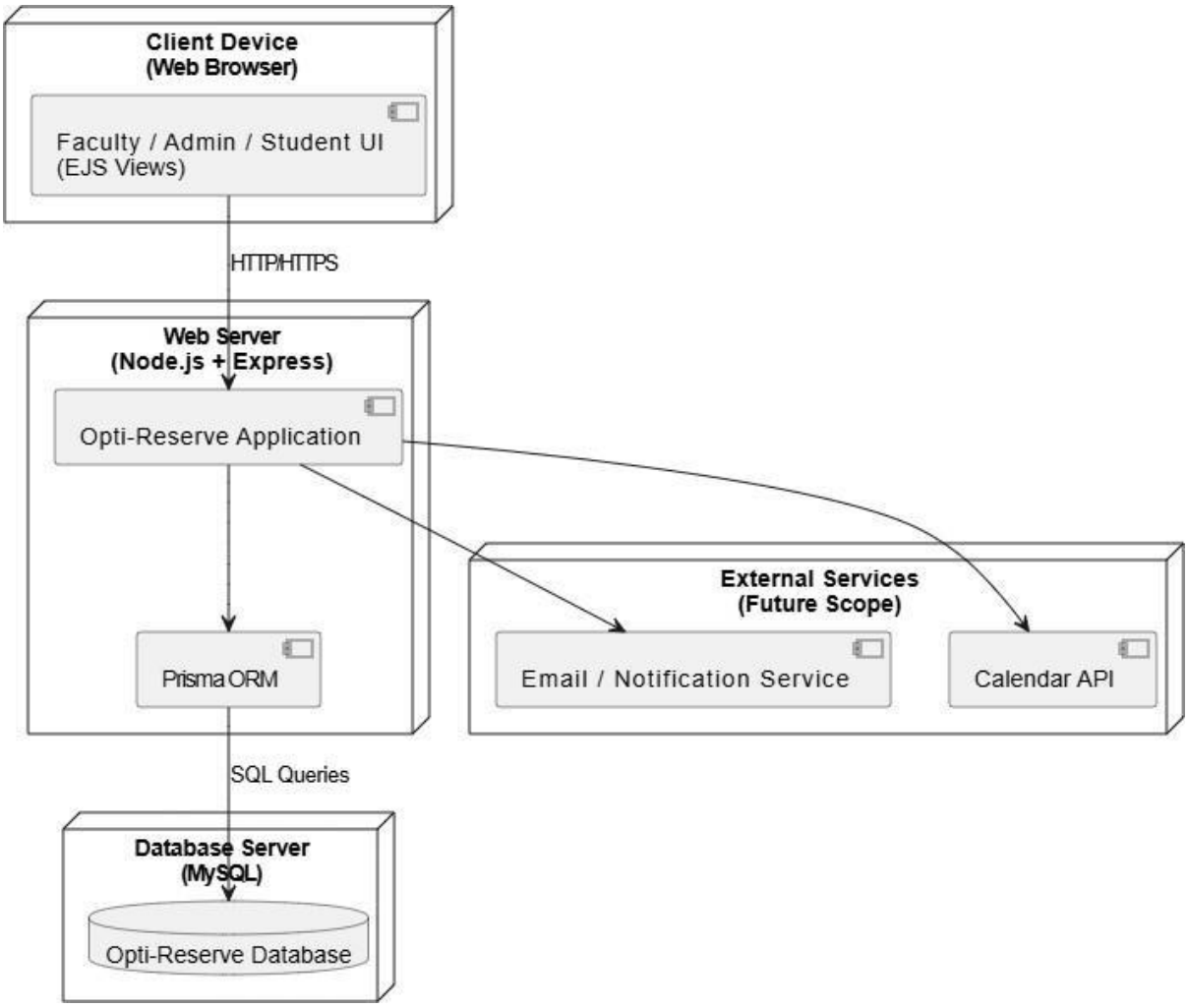


Figure 50 Deployment Diagram

Test Cases

1. Login

Test Case ID		CV_01			
Test Case Name		Login	**Test Case Description**	This test case is use to test the functionality of login feature.	
Created By		QA Team	**Version**	1.0	01 Dec 2025
S.no	**Prerequisites:**				
1	User must have an account				
2	Account must be active				
Test Scenario	Verify on entering valid user id and password, the customer can login				
Step no.					
	Step Details	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**	
1	Enter wrong Email and correct password	Display "Invalid credentials"	Error invalid client ID	Fail	
2	Enter wrong Email and correct password	Display "Invalid credentials"	Generic error message "Invalid credentials" displayed	Pass	
3	Enter a non-existent user ID with a valid password and click Login	Display "Invalid credentials"	Error prevents navigation	Fail	
4	Enter a non-existent user ID with a valid password and click Login	Display "Invalid credentials"	Generic "Invalid credentials" message displayed	Pass	
5	Enter Correct ID and wrong Password	Display "Invalid credentials"	As ExpectedGeneric "Invalid credentials" message displayed	Pass	

6	Enter Correct ID and correct Password	Redirect to Dashboard	User logged in successfully	Pass
7	leave both fields empty	System displays validation message	Validation message shown: “Email and password are required”; form submission blocked	Pass

Table 34 Login Test Case

2. Registration

Test Case ID		CV_02			
Test Case Name		User Registration	**Test Case Description**	User Registration	
Created By		QA Team	**Version**	1	01 Dec 2025
S.no	**Prerequisites:**				
1	User must not have existing account				
2	Account must be active				
Test Scenario	Verify registration with various inputs				
Step no.					
	Step Details	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**	
1	Enter valid username, email, password	Account created successfully	Account created	Pass	
2	Enter existing email	email exist	As Expected	Pass	
3	Enter password less than 6 characters	Display "Password too short"	Weak password accepted by system	Fail	
4	Enter password less than 6 characters	Display "Password too short"	Validation message shown, signup blocked	Pass	
5	Enter username with special chars	Display validation error	Allows special chars	Fail	

6	Enter username with special characters	Display validation error	Validation error shown, form submission blocked	Pass
7	leave both fields empty	System displays validation message	Validation message shown: “Email and password are required”; form submission blocked	Pass

Table 35 User Registration Test Case

3. Forget-Password

Test Case ID		CV_03			
Test Case Name		Forgot Password	**Test Case Description**	Forgot Password	
Created By		QA Team	**Version**	1	01 Dec 2025
S.no	**Prerequisites:**				
1	User has registered account				
2	Account must be active				
Test Scenario	Verify password reset functionality				
Step no.	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**	
1	Enter non-existent email	Display generic message "If email exists, reset code sent"	As Expected	Pass	
2	Enter valid email	Send code	As Expected	Pass	
3	click expired code	Display "code expired"	code still works	Fail	
4	Click expired code	Display "code expired"	"Code expired" message displayed and verification blocked	pass	
5	Enter new password mismatch	Display "Passwords don't match"	As Expected	Pass	

6	Enter same old password	Display "Cannot use old password"	System allows reuse of old password	Pass

Table 36 Forget Password Test Case

4. Profile Management

Test Case ID		CV_04				
Test Case Name		Profile Management	**Test Case Description**	Profile Management		
Created By		QA Team	**Version**	1		01 Dec 2025
S.no	**Prerequisites:**					
1	User logged in					
2	Account must be active					
Test Scenario	Verify profile update functionality					
Step no.	**Step Details**		**Expected Results**	**Actual Results**		**Pass / Fail / Not executed / Suspended**
1	Update profile picture		Image saves successfully	Upload timeout		Fail
2	Update profile picture		Image saves successfully	Image uploaded successfully		Pass
3	Change display name		Name updates across site	As Expected		Pass
4	Change email address		Send verification to new email	As Expected		Pass
5	Delete account		Account removed / Deactivated	Error 500		Fail
6	Delete account		Account removed / Deactivated	Account deactivated successfully		Pass

Table 37 Profile Management Test Case

5. Admin Dashboard

Test Case ID		CV_05				
Test Case Name		Admin Dashboard	**Test Case Description**	Admin Dashboard		
Created By		QA Team	**Version**	1		01 Dec 2025

S.no	**Prerequisites:**			
1	Admin logged in			
2	Account must be active			
Test Scenario	Verify Admin Functionality			
Step no.	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Ban user	User cannot log in	As Expected	Pass
2	Create new Category	appears in list	As Expected	Pass
3	Edit existing Category	Changes reflect on frontend	Cache issue (delay)	Fail
4	Edit existing Category	Changes reflect on frontend	Changes reflected immediately on frontend after save without cache delay	Pass
5	View Analytics	Show user stats	Graph data empty	Fail
6	View Analytics	Show user stats	Analytics graphs display correct user statistics with updated data	Pass

Table 38 Admin Dashboard Test Case

6. Session Management

Test Case ID	CV_06				
Test Case Name	Session Management	**Test Case Description**	Session Management		
Created By	QA Team	**Version**	1		01 Dec 2025
S.no	**Prerequisites:**				
1	User logged in				
2	Account must be active				
Test Scenario	Verify session handling				

Step no.	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Idle for 15 mins	Auto-logout	Stays logged in	Fail
2	Idle for 15 mins	Auto-logout	User automatically logged out after inactivity and redirected to login page	Pass
3	Log in on second device	First session remains or alerts	Both active silently	Pass
4	Log out	Cannot access protected pages	As Expected	Pass
5	Click back button after logout	Redirect to login	Shows cached page	Fail
6	Click back button after logout	Redirect to login	Cached pages cleared; user redirected to login and cannot access protected pages	Pass
7	Close browser tab	Session persists (if 'Remember Me')	As Expected	Pass

Table 39 Session Management Test Case

7. Two Factor Authentication

Test Case ID		CV_07				
Test Case Name		Two-Factor Authentication	Test Case Description	Two-Factor Authentication		
Created By		QA Team	Version	1		01 Dec 2025
S.no	Prerequisites:					
1	User logged in					
2	Account must be active					

Test Scenario	Verify 2FA setup and login			
Step no.	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Enable 2FA	Show QR Code	As Expected	Pass
2	Scan QR and enter code	2FA Activated	As Expected	Pass
3	Login with 2FA enabled	Prompt for code after password	As Expected	Pass
4	Enter wrong 2FA code	Deny access	As Expected	Pass
5	Use backup code	Access granted	Invalid code error	Fail
6	Use backup code	Access granted	Valid backup code accepted; user logged in successfully and code marked as used	Pass

Table 40 2 Factor Authentication Test Case

8. Search Functionality

Test Case ID	CV_08			
Test Case Name	Search Functionality	**Test Case Description**	Search Functionality	
Created By	QA Team	**Version**	1	01 Dec 2025
S.no	**Prerequisites:**			
1	User logged in			
2	Account must be active			
Test Scenario	Verify search functionality			
Step no.	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**

1	Search 'Python'	Show Python rooms	As Expected	Pass
2	Search non-existent term	Show "No results found"	Shows empty list	Pass
3	Search with special chars	Handle safely	Crash / SQL Error	Fail
4	Search with special chars	Handle safely	Special characters handled safely; no crash or SQL error and results shown appropriately	Pass
5	Filter by 'Free'	Show only free rooms	Includes Premium rooms	Fail
6	Filter by 'Free'	Show only free rooms	Only free rooms displayed; premium rooms excluded from results	Pass
7	Sort by 'Newest'	Correct order	As Expected	Pass

Table 41 Search Functionality Test Case

9. File Upload Validation

Test Case ID	CV_09			
Test Case Name	File Upload Security	**Test Case Description**	File Upload Security	
Created By	QA Team	**Version**	1	01 Dec 2025
S.no	**Prerequisites:**			
1	User logged in			
2	Account must be active			
Test Scenario	Verify file upload security			
Step no.	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**

1	Upload .exe file	Block upload	File uploaded	Fail
2	Upload .exe file	Block upload	Upload blocked; system displays "Invalid file type" error message	Pass
1.1	Upload 10MB image	Block (Limit 5MB)	As Expected	Pass
1.2	Upload valid .jpg	Upload success	As Expected	Pass
1.3	Upload .php shell	Block upload	Blocked	Pass
1.4	Upload file with no extension	Block or validate content	As Expected	Pass

Table 42 File Upload Validation Test Case

10. API Security

Test Case ID		CV_010			
Test Case Name		API Security	**Test Case Description**	API Security	
Created By		QA Team	**Version**	1	01 Dec 2025
S.no	**Prerequisites:**				
1	API endpoints available				
2	Account must be active				
Test Scenario	Verify API security				
Step no.	**Step Details**		**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Access API without token		Return 401 Unauthorized	Returns 200 OK	Fail
2	Access API without token		Return 401 Unauthorized	API request rejected; 401 Unauthorized returned when no token is provided	Pass
3	Access Admin API as User		Return 403 Forbidden	Returns 403	Pass

4	Send malformed JSON	Return 400 Bad Request	Returns 500 Error	Fail
5	Send malformed JSON	Return 400 Bad Request	API validates request body and returns 400 Bad Request without server error	Pass
6	SQL Injection in param	Block request	As Expected	Pass

Table 43 API Security Test case

11. Booking Completion

Test Case ID		CV_011			
Test Case Name		Booking Completion	**Test Case Description**		
Created By		QA Team	**Version**	1	01 Dec 2025
S.no	**Prerequisites:**				
1	User logged in				
2	Session not completed before				
Test Scenario	Verify classroom				
Step no.	**Step Details**		**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Complete lab and click “Complete Booking”		Points added and streak updated	Streak not updated	Fail
2	Complete Booking and click “Complete Booking”		Points added and streak updated	Points added, streak updated correctly	Pass
3	Click “Complete Booking” again		Duplicate completion blocked	completion blocked Points added again	Fail
4	Click “Complete Booking” again		Duplicate completion blocked	As Expected	Pass

Table 44 Booking Completion Test Case

12. Role-Based Access Control

Test Case ID		CV_012				
Test Case Name		Role-Based Access Control	**Test Case Description**		Verify normal user cannot become admin	
Created By		QA Team	**Version**		1	01 Dec 2025
S.no	**Prerequisites:**					
1	User logged in					
2	not completed before					
Test Scenario	Verify that a normal user cannot access admin APIs or change user roles through direct API manipulation.					
Step no.	**Step details**		**Expected Result**		**Actual Result**	**Pass / Fail / Not executed / Suspended**
1	Login as normal user		User logged in		As Expected	Pass
2	Try to change own role to admin via API		Access denied		User became admin	Fail
3	Try to change own role to admin via API		Access denied		Request blocked, role unchanged	Pass

Table 45 RBACs Test Case

13. Admin Panel Real Time Update

Test Case ID		CV_013				
Test Case Name		Admin Real-Time Update	**Test Case Description**		Verify Dashboard Real time updates for all users	
Created By		QA Team	**Version**		1	01 Dec 2025
S.no	**Prerequisites:**					
1	User logged in					
2	Account must be activated					

Test Scenario	Verify that admin panel updates in real time for all users when points or ranking change.			
Step no.	**Step Details**	**Expected Result**	**Actual Result**	**Pass/Fail/Not executed/Suspended**
1	User completes Booking-point	Dashboard updates for all	Only one Booking updated	Fail
s2	Admin approves all pending reservation	Dashboard updates for all	All bookings updated	Pass

Table 46 Admin Panel Real Time Update Test Case

User Manual

1. Introduction

1.1 Purpose

The purpose of the Opti-Reserve system is to make it easy to centrally manage and keep track of institutional resources. The system removes the need for manual booking and reduces scheduling conflicts with real-time tracking and automatic booking management, along with administrative control.

This User Manual gives detailed instructions on the use of the Opti-Reserve system, its modules and the features of the system.

1.2 Scope

Opti-Reserve is a web-based facility and resource reservation platform that enables users to:

- Set up and oversee financial accounts
- Set aside funds for the organisation
- Find out what resources are available
- Oversee enquiries for reservations
- Observe the current standing of reservations
- Sign up for waiting lists for resources that are down
- Oversee bookings and timetables
- You can access analytics and reports.
- Keep an eye on how resources are being used.

To learn about KPIs and acquire insights, use analytics and reports.

1.3 Intended Audience

The system is designed for the following users:

- **Administrator:** Responsible for managing the overall system, resources, users, and booking operations.
- **Faculty Members:** Reserves resources, makes bookings and checks schedules.
- **Students and General Users:** Able to see what resources are available, and request reservations as assigned.
- **System Maintainers:** Technical staff who maintain, updates and monitors the system.

1.4 System Overview

Opti-Reserve is built on Web-based current technology and is designed as a client-server application. The system provides a safe and secure environment for managing reservations with automatic scheduling and conflict detection.

Major modules include:

- User Authentication
- Resource Management
- Booking Management
- Waitlist Management
- User Management
- Schedule Management
- Analytics and Reporting

- Account Settings

2. System Requirements

2.1 Hardware Requirements

- **Processor:** Intel Core i3 or higher
- **RAM:** Minimum 4 GB
- **Storage:** 500 MB free disk space
- **Internet:** Stable internet connection

2.2 Software Requirements

- **Operating System:** Windows 10/11, Linux, macOS
- **Web Browser:** Google Chrome, Microsoft Edge, Mozilla Firefox
- **Development Environment:** Node.js, MySQL Database, Visual Studio Code, Git

3. Accessing the System

- **Step 1:** Open a supported web browser.
- **Step 2:** Navigate to the Opti-Reserve application URL.
- **Step 3:** Login using existing credentials or create a new account.

4. User Authentication

4.1 User Registration

New users are required to register in order to use the system.

Registration Procedure:

1. Click **Sign Up**.
2. Enter **Full Name**.
3. Enter **Email Address**.
4. Create **Password**.
5. Confirm **Password**.
6. Submit Registration Form.

System Response:

- Account is created successfully.
- Verification email is sent.
- User receives confirmation notification.

4.2 Email Verification

Procedure:

1. Open verification email.
2. Click verification link.
3. System verifies the account.
4. Account is activated.

4.3 User Login

Procedure:

1. Open Login Page.
2. Enter Email Address.
3. Enter Password.
4. Click **Login**.

- **Successful Login:** The user is redirected to the dashboard.
- **Failed Login:** The system displays an error message.

4.4 Forgot Password

Procedure:

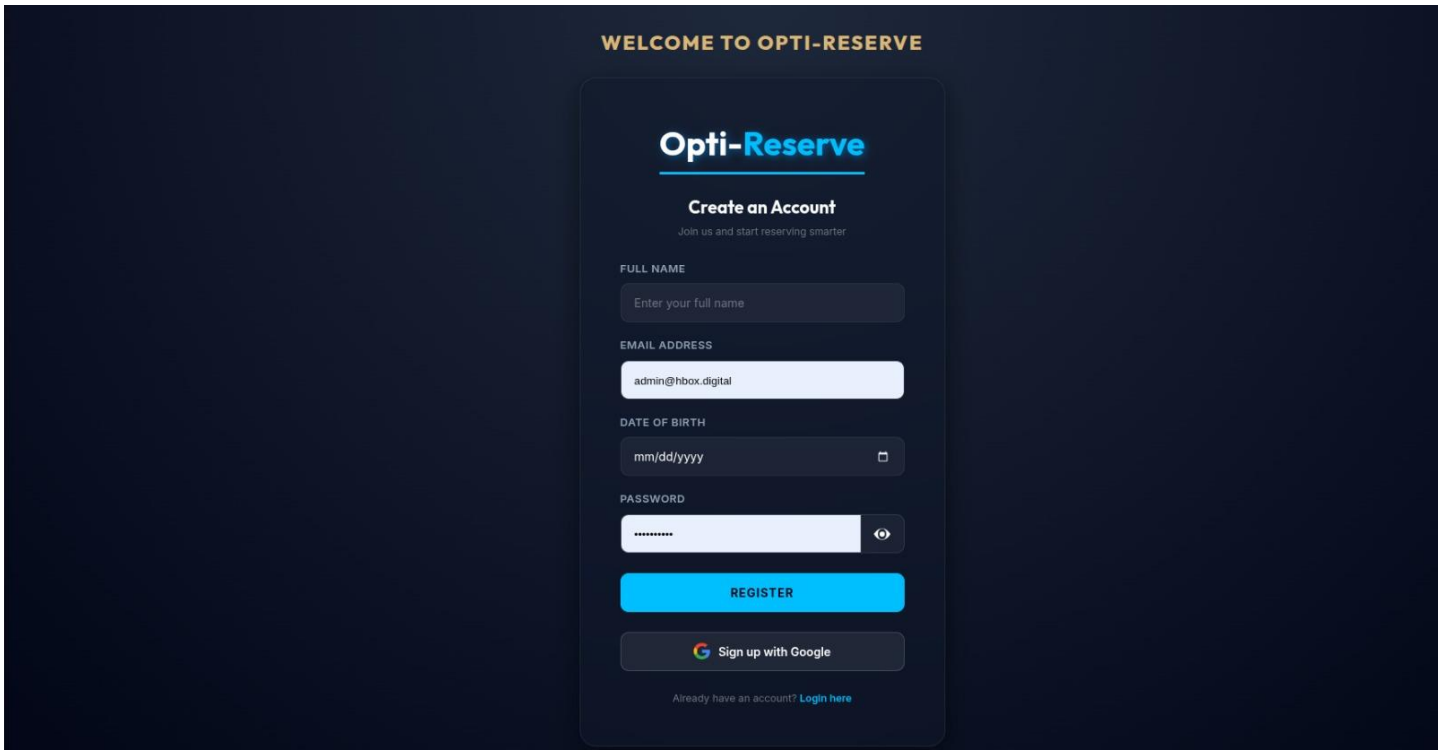
1. Click **Forgot Password**.
2. Enter registered email address.
3. Submit request.
4. Open password reset email.
5. Set a new password.
6. Login using updated credentials.

5. Dashboard

The dashboard serves as the central control panel of the system.

Features:

- Resource Overview
- Booking Overview
- Schedule Overview
- Notifications
- Quick Navigation
- User Information

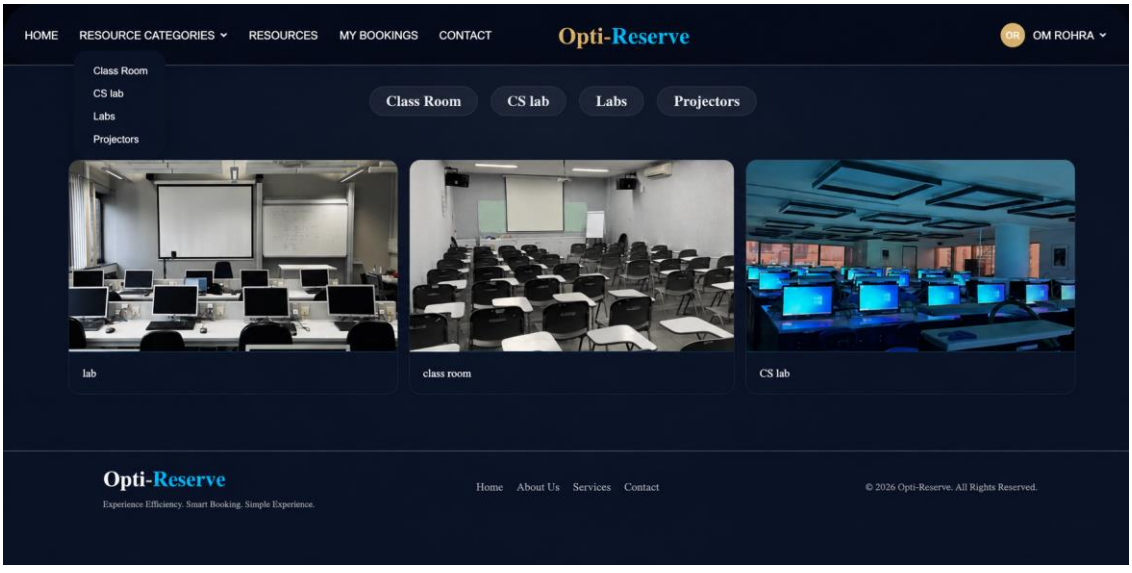


6. Resource Categories

Resources are organized into categories to improve accessibility and management.

Features:

- View Categories
- Search Categories
- Filter Resources
- Browse Resource Types



7. Resource Management

The Resource Management module allows users to view available resources and their details.

Features:

- View Resources
- Check Availability
- Search Resources
- View Resource Information



Resource Information Fields:

- Resource Name
- Category
- Description
- Availability Status
- Booking Schedule

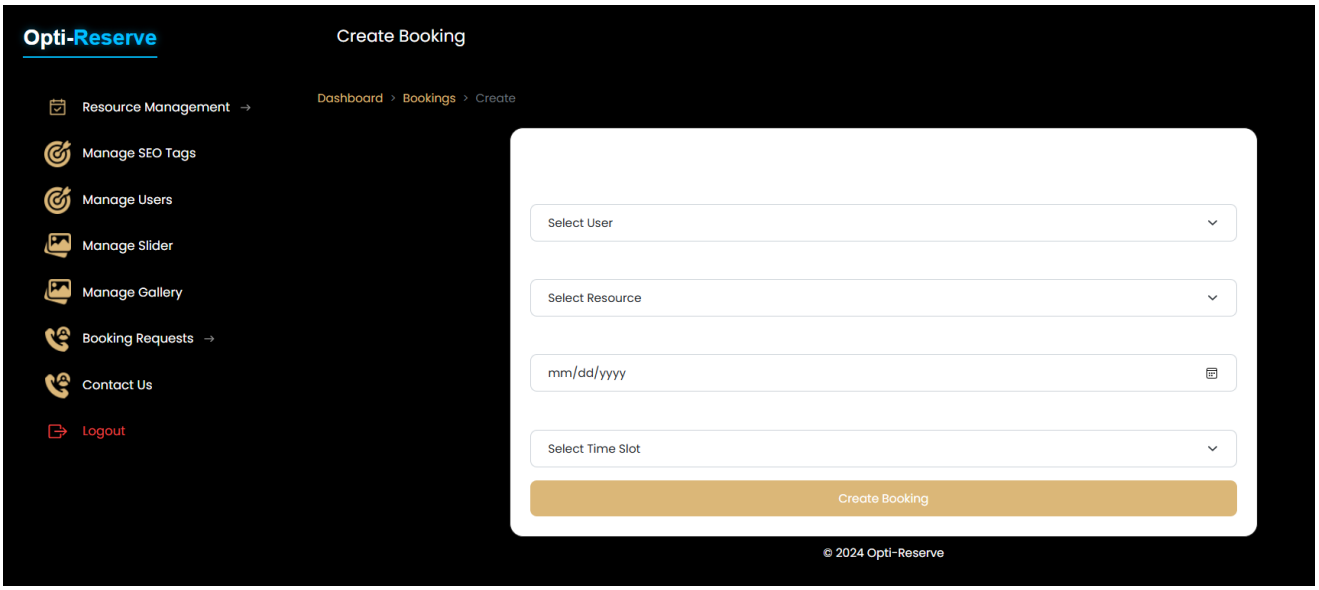
8. Booking Management

Users can reserve available resources through the booking system.

Booking Procedure:

1. Select a resource.
2. Select booking date.
3. Select time slot.
4. Enter booking purpose.
5. Submit booking request.

Booking Statuses: Pending, Approved, Rejected, Completed, Cancelled.



9. Booking Conflict Detection

The system automatically detects scheduling conflicts before confirming a reservation.

Validation Checks:

- Duplicate Bookings
- Overlapping Reservations
- Resource Unavailability
- Invalid Time Slots

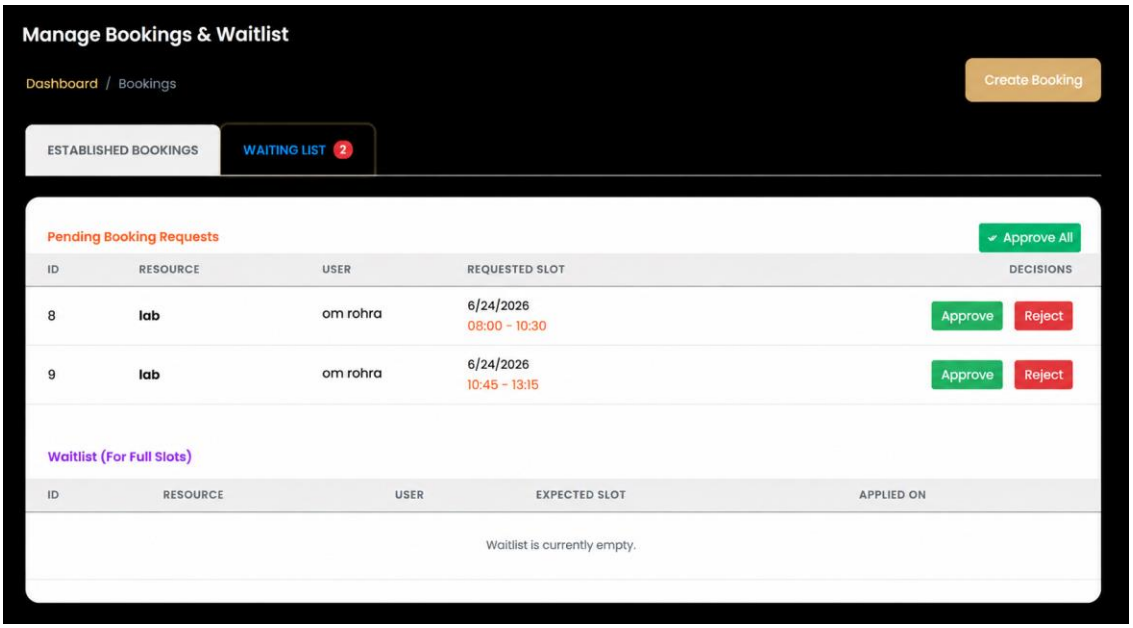
Result: Users receive immediate feedback if conflicts are detected.

10. Waitlist Management

Users may join a waitlist when a resource is unavailable.

Procedure:

1. Select unavailable resource.
2. Choose **Join Waitlist**.
3. Waitlist request is recorded.
4. Notification is sent when a slot becomes available.

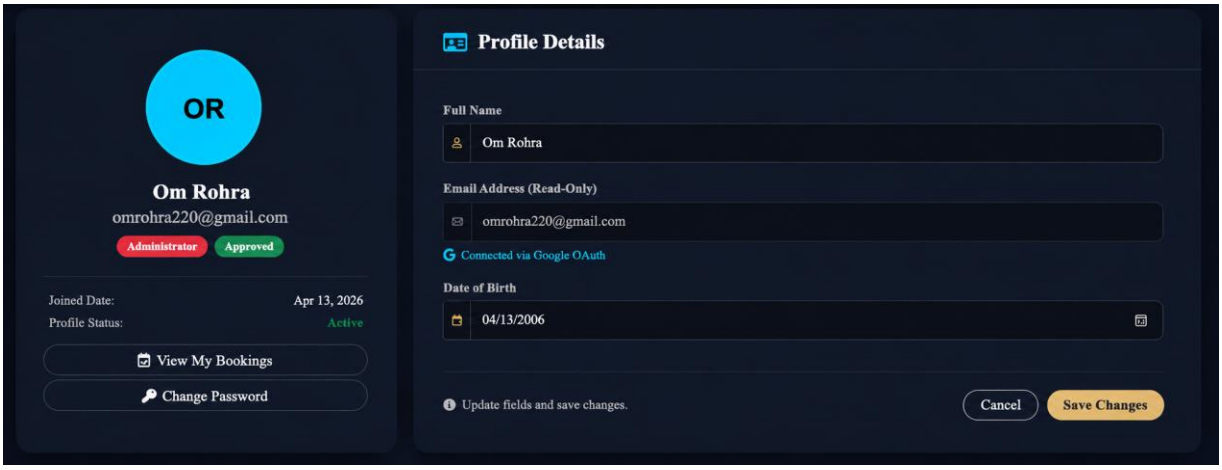


11. User Profile Management

Users can manage personal information and account preferences.

Features:

- View Profile
- Edit Profile
- Change Password
- Update Contact Information
- Account Preferences

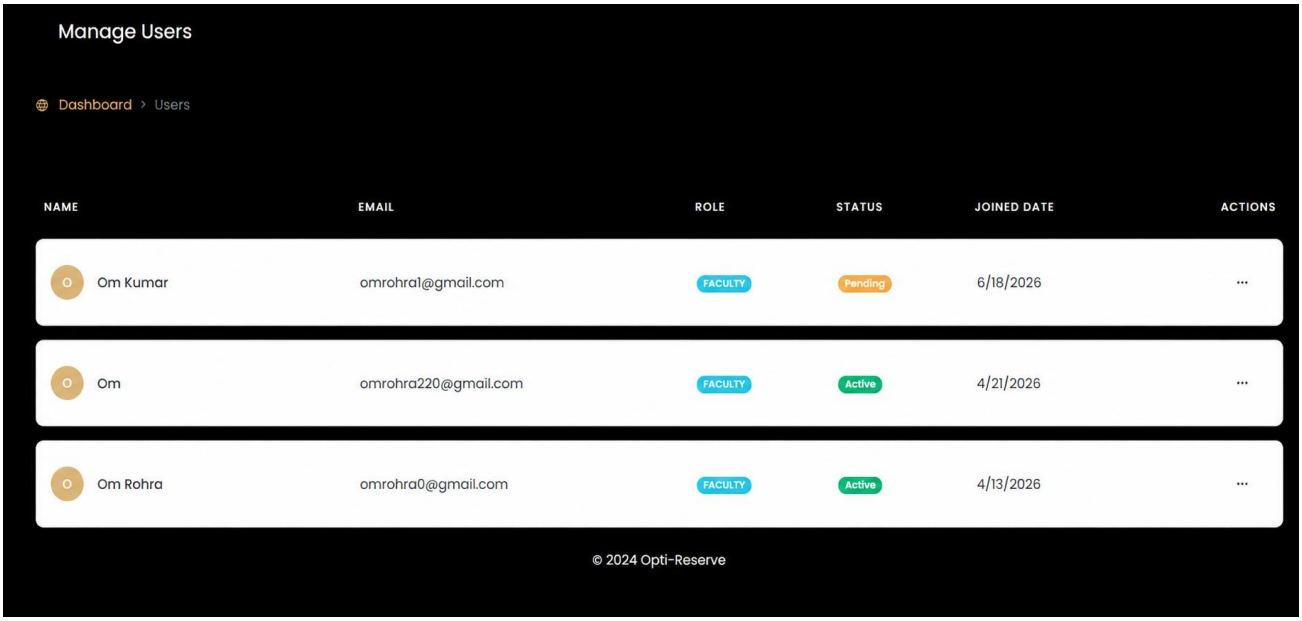


12. User Management

Administrators manage all registered users.

Features:

- View Users / Add Users / Edit Users / Delete Users
- Activate / Deactivate Accounts

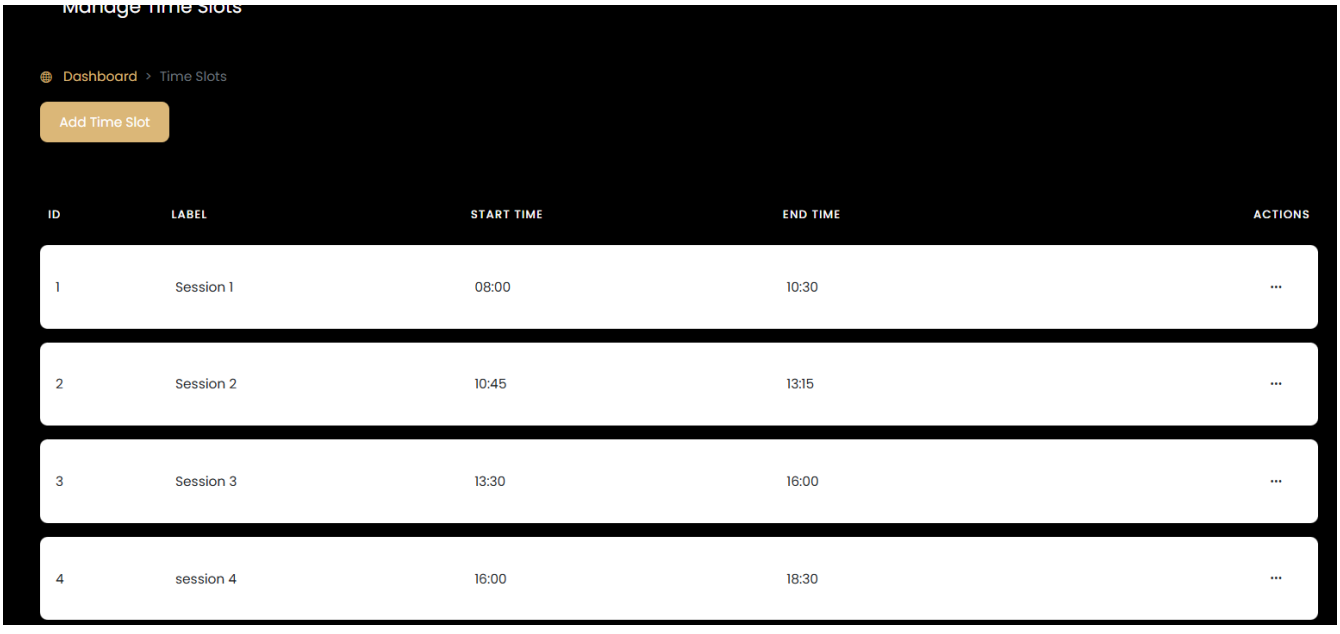


13. Schedule Management

The Schedule Management module enables administrators to manage resource availability and booking schedules.

Features:

- Create, Update, or Delete Schedule
- View Calendar
- Manage Time Slots



14. Analytics and Reporting

The Analytics module gives information about the use of the system and resources.

Available Reports: Total Bookings, Active Users, Resource Utilization, Booking Trends, Resource Usage Statistics.
Benefits: Improved decision making, effective resource allocation and performance monitoring.

15. Account Settings

Users can set up their account preferences and security settings.

Features: Update Account Information, Change Password, Manage Preferences, Security Settings.

16. Security Features

Opti-Reserve safeguards with several security features:

- Secure Authentication
- Password Encryption
- Session Management
- Email Verification
- Role-Based Access Control
- Secure Database Storage

17. Troubleshooting

- **Login Failure:**
  - *Possible Causes:* Incorrect Credentials, Unverified Email, Expired Session.
  - *Solution:* Verify credentials, reset password, or verify email account.
- **Booking Failure:**
  - *Possible Causes:* Resource unavailable, booking conflict, missing required information.
  - *Solution:* Select another time slot, complete all fields, and submit request again.
- **Email Verification Not Received:**
  - *Possible Causes:* Spam filtering, incorrect email address.
  - *Solution:* Check spam folder or request another verification email.

18. Frequently Asked Questions

- **Q1. Can multiple users reserve the same resource simultaneously?**
  - *A:* No. Conflicting reservations are automatically prevented.
- **Q2. Can I cancel a booking?**
  - *A:* Yes. Users are able to cancel bookings on the booking management module.
- **Q3. What happens when a booking is rejected?**
  - *A:* User is alerted and can make another request for a booking.

- **Q4. Can administrators modify bookings?**
  - *A:* Yes. . The administrator can book and update the bookings.
- **Q5. Is my information secure?**
  - *A:* Yes. The system has secure authentication and encrypted password storage.
- **Q6. How does Opti-Reserve prevent double booking?**
  - *A:* Automatic conflict detection takes place prior to confirming a reservation request.
- **Q7. Can administrators modify booking requests?**
  - *A: Yes, bookings can be reviewed and approved/rejected by the administrator from the administration panel*.
- **Q8. What happens when a requested resource is unavailable?**
  - *A: User is informed of a proper time slot and can either change to another time slot or enter the waitlist, if one exists*.

19. Limitations

- Internet connectivity is required.
- The system is accessible through web browsers only.
- System performance depends on server availability.
- Resource availability depends on administrative configuration.

20. Future Enhancements

There are potential improvements that could be made in the future:

- Mobile Application Support
- QR-Based Check-In System
- Smart Booking Recommendations
- Calendar Integration
- Push Notifications
- Advanced Analytics Dashboard
- Multi-Institution Support

21. Contact and Support

Users can reach out to the project development team for technical support, maintenance or project related questions.

- **Project Name:** Opti-Reserve
- **Developers:** Om Kumar, Gulzar Hussain
- **Supervisor:** Ms. Sadia Aziz
- **Department:** Faculty of Computing and Engineering Sciences
- **Institution:** Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology (SZABIST), Karachi
- **Email:** omrohra220@gmail.com
- **Version:** 1.0