**FINAL YEAR PROJECT**

**Smart Dorm**

Project Code: 20F-27

****

Final Year Project Report by

**Zainab Fatima Bhutto**

**Aisha**

**Syed Muzamil Shah**

In Partial Fulfillment

Of the Requirements for the degree Bachelors of Science (CS/SE)

Sukkur IBA University, Sukkur, Sindh Pakistan

### Smart Dorm

By

### Zainab Fatima Bhutto

### Aisha

### Syed Muzamil Shah

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE IN PARTIAL

FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE / SOFTWARE ENGINEERING**

At the

SUKKUR IBA University

May 2024

**© 2024 Zainab Fatima Bhutto, Aisha, Syed Muzamil Shah**

**All rights reserved.**

The author hereby grants permission to Sukkur IBA to reproduce and distribute publicly, paper and electronic copies of this thesis and grant others the right to do so.

Signature of Author(s)

Certified by: Internal Examiner



External Examiner:

Accepted by

### DECLARATION

We hereby declare that this project report entitled “Smart Dorm” submitted to the

“**Computer Science/ Software Engineering”**, is a record of an original work done by us under the guidance of Supervisor “**Dr. Faheem Rajput**” and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Computer Science.

###### Team Members Signature

Zainab Fatima

Aisha

Muzamil Shah

###### Supervisor: Signature

Supervisor Name

###### Date:

**Place:**

**Acknowledgements:**

We extend our heartfelt gratitude to the Almighty Allah, the Most Merciful, for granting us the opportunity to embark on this project. We are deeply thankful for the blessings and guidance that have enabled us to undertake our studies in Computer Science at Sukkur IBA University, where we have been privileged to delve into the realm of technology.

We honor the memory of the late Professor Nisar Ahmad Siddiqi, the founding figure of Sukkur IBA University. His dedication to education and the nurturing of this institution has left an indelible legacy. His teachings and commitment continue to inspire us, shaping our journey in the field of Computer Science.

We express our deepest appreciation to the esteemed faculty members and mentors in the Computer Science department. Their expertise, encouragement, and guidance have been invaluable throughout our project. We are profoundly grateful for the opportunities to learn and grow under their tutelage.

Lastly, we extend our heartfelt thanks to our beloved parents, friends, and colleagues. Their unwavering belief in us has been a constant source of motivation. We are deeply grateful for their support and encouragement throughout this project and beyond.

**Table of Contents**

**Abstract-------------------------------------------------------------------------------------------------8**

1. **Introduction-----------------------------------------------------------------------------------9**
   1. Purpose of Document---------------------------------------------------------------------------------9
   2. Intended Audience-------------------------------------------------------------------------------------9
   3. Document Convention--------------------------------------------------------------------------------9
   4. Project Scope------------------------------------------------------------------------------------------10
   5. Not In Scope-------------------------------------------------------------------------------------------10
2. **Literature Review and Comparison-----------------------------------------------------11**
   1. Existing Systems--------------------------------------------------------------------------------------11
   2. Smart Dorm Project-----------------------------------------------------------------------------------11
3. **Overall System Description----------------------------------------------------------------12**
   1. Project Background-----------------------------------------------------------------------------------12
   2. Project Objectives-------------------------------------------------------------------------------------12
   3. Stakeholders--------------------------------------------------------------------------------------------12
   4. Operating Environment-------------------------------------------------------------------------------12
   5. System Constraints------------------------------------------------------------------------------------13
   6. Assumptions & Dependencies-----------------------------------------------------------------------14
4. **Project Breakdown---------------------------------------------------------------------------15**
5. **Proposed Methodology-------------------------------------------------------------------------16**
   1. Work Flow ---------------------------------------------------------------------------------------------16
6. **Functional Requirements-------------------------------------------------------------------18**
   1. Functional Hierarchy----------------------------------------------------------------------------------18
   2. User Class 1: The Resident Interaction-------------------------------------------------------------19
   3. User Class 2: The Administrative Interaction------------------------------------------------------19
   4. Use Cases-----------------------------------------------------------------------------------------------20
      1. Room Allocation---------------------------------------------------------------------------------20
      2. Leave Request------------------------------------------------------------------------------------21
      3. Lost and Found Report--------------------------------------------------------------------------22
      4. Room Availability Updates---------------------------------------------------------------------23
      5. Communication Portal---------------------------------------------------------------------------24
7. **Non-Functional Requirements-------------------------------------------------------------25**
   1. Performance Requirements---------------------------------------------------------------------------25
   2. Safety Requirements-----------------------------------------------------------------------------------25
   3. Security Requirements---------------------------------------------------------------------------------25
   4. User Documentation-----------------------------------------------------------------------------------25
8. **Detailed Design Architecture ---------------------------------------------------------------26**
   1. Architecture Design Approach------------------------------------------------------------------------26
   2. Entity Relationship Diagram -----------------------------------------------------------------------27
   3. Sequence Diagram------------------------------------------------------------------------------------28
      1. Login Sequence Diagram------------------------------------------------------------ ----------28
      2. Communication Module Sequence Diagram-------------------------------------- ----------29
      3. Room Allocation Module Sequence Diagram-----------------------------------------------30
      4. Lost and Found Module Sequence Diagram-------------------------------------------------31
      5. Room Availability Updates Sequence Diagram---------------------------------------------32
      6. Notification Module Sequence Diagram-----------------------------------------------------32
      7. Leave Request Module Sequence Diagram--------------------------------------------------33
   4. Architecture Design Diagram-----------------------------------------------------------------------34
   5. Database Diagram------------------------------------------------------------------------------------35
   6. Class Diagram-----------------------------------------------------------------------------------------36
   7. Test Cases----------------------------------------------------------------------------------------------37
9. **Milestones Achieved-------------------------------------------------------------------------39**
10. **Results and Discussion-----------------------------------------------------------------------41**
    1. Website Application---------------------------------------------------------------------------41
    2. Mobile Application----------------------------------------------------------------------------44
11. **Conclusion and Future Work--------------------------------------------------------------47**
    1. Conclusion--------------------------------------------------------------------------------------47
    2. Future Directions-------------------------------------------------------------------------------47
12. **References--------------------------------------------------------------------------------------48**

#### List of Figures

#### Figure 1: Functional Hierarchy Diagram--------------------------------------------------18

#### Figure 2: User Class of Resident Interaction---------------------------------------------19

#### Figure 3: User Class of Administrative Interaction--------------------------------------19

#### Figure 4: Entity Relationship Diagram----------------------------------------------------27

#### Figure 5: Sequence diagram-----------------------------------------------------------------28

#### Figure 5.1: Login Sequence Diagram ------------------------------------------------28

#### Figure 5.2: Communication Module Sequence Diagram--------------------------29

#### Figure 5.3: Room Allocation Module Sequence Diagram-------------------------30

#### Figure 5.4: Lost and Found Module Sequence Diagram --------------------------31

#### Figure 5.5: Room Availability Updates Sequence Diagram----------------------32

#### Figure 5.6: Notification Module Sequence Diagram ------------------------------32

#### Figure 5.7: Leave Request Module Sequence Diagram ---------------------------33

#### Figure 6.0: Architecture Design Diagram -----------------------------------------------34

#### Figure 7.0: Database Diagram -------------------------------------------------------------35

#### Figure 8.0: Class Diagram -----------------------------------------------------------------36

#### Figure 9.0: Login Screen -------------------------------------------------------------------41

#### Figure 10: Signup Screen-------------------------------------------------------------------41

#### Figure 11: Room Allocation Screen ------------------------------------------------------42

#### Figure 12: Leave Request Screen ---------------------------------------------------------42

#### Figure 13: Chat Screen----------------------------------------------------------------------43

#### Figure 14: Lost and Found Screen --------------------------------------------------------43

#### Figure 15: Splash Screen -------------------------------------------------------------------44

#### Figure 16: Login Screen ---------------------------------------------------­­­­­­­­­­-----------------44

#### Figure 17: User Dashboard Screen --------------------------------------------------------45

#### Figure 18: Leave Request Form Screen --------------------------------------------------45

#### Figure 19: Room Availability Screen -----------------------------------------------------46

#### Figure 20: Notification Screen -------------------------------------------------------------46

#### List of Tables

#### Table 1: Project Breakdown ----------------------------------------------------------------15

#### Table 2: Room Allocation-------------------------------------------------------------------20

#### Table 2: Leave Request Processing --------------------------------------------------------21

#### Table 4: Lost and Found Report -----------------------------------------------------------22

#### Table 5: Room Availability Updates ------------------------------------------------------23

#### Table 6: Real-Time Communication ------------------------------------------------------24

#### Table 7: Test cases----------------------------------------------------------------------------37

#### Abstract

#### This project introduces the idea of creating a "Smart Dorm," an advanced Hostel Management System that makes hostel life smoother. Our system will utilize the latest technologies to build a user-friendly platform. It will include features like room assignments, instant messaging, leave requests, lost and found tracking, updates on room availability, and notifications for room assignments. One exciting feature will be a Lost and Found portal, where residents can report lost items, and anyone who finds something can easily inform the owner. This system aims to make hostel living more efficient, safe, and convenient by harnessing cutting-edge technology.

#### Introduction

* 1. Purpose of Document

This document serves as the definitive guide for the "Smart Dorm" project, providing a comprehensive overview of the project's objectives, functionalities, and expected outcomes. It is intended to ensure a shared understanding of the project's goals and how it will revolutionize hostel management, enhance security, and create a more efficient and connected living environment. It will also provide an illustration and the purpose of declaration for the development of the system. The document is intended to provide the complete scope, objectives, outcomes, operational environment and limitations, assumptions or dependencies related to the proposed system.

* 1. Intended Audience

The intended audience for the "Smart Dorm" project consists of hostel administrators, residents, technical teams, project supervisors, system administrators, external service providers, and other stakeholders associated with the hostel environment. The system is designed to meet the unique needs of each group, facilitating efficient hostel management, enhanced resident experiences, and streamlined technical operations

* 1. Document Convention
* Main Heading Titles:
  + Font: Times new roman
  + Face: Bold black
  + Size: 18
* Sub Heading Titles:
  + Font: Times new roman
  + Face: Bold black & underlined
  + Size: 14
* Main Points:
  + Font: Times new roman
  + Face: bold black & displayed with bullets.
  + Size: 12
* Other Text Explanation:
  + Font: Times new roman
  + Face: Normal black
  + Size: 12
  1. Project Scope

The "Smart Dorm" project encompasses a range of functionalities aimed at optimizing hostel management. It will include features such as room allocation, real-time communication, leave request processing, a Lost and Found portal, room availability updates, and automated notifications.

* 1. Not in Scope

The following functionalities are explicitly not in the scope of the "Smart Dorm" project:

* **External Maintenance**: Physical maintenance and repairs in residents' rooms are not included.
* **Supply Procurement:** Procurement of items and financial transactions for purchases are not part of the project.
* **Physical Security Infrastructure:** Installation and management of physical security systems are outside the project's scope.
* **Healthcare Services:** Medical services are not provided by the system.
* **Transportation Services:** Transportation arrangements are not within the project's scope.

1. **Literature Review and Comparison:**
   1. **Existing Systems:**

* **Functionality**: Common functionalities include room allocation, maintenance requests, and communication platforms. While functional, these systems often lack real-time updates and comprehensive leave management.
* **User Experience**: Many systems struggle with user-friendly interfaces, making navigation and operation cumbersome for both administrators and students.
* **Technology**: Predominantly built on traditional frameworks, existing solutions may not fully leverage modern web technologies, limiting scalability and integration capabilities.  
  1. **Smart Dorm Project:**
* **Innovative Features**: Beyond standard functionalities, Smart Dorm introduces a Lost and Found portal and real-time communication, addressing specific pain points in hostel management.
* **User-Centric Design**: Focused on providing an intuitive user experience, Smart Dorm is designed with the end-user in mind, ensuring ease of use and accessibility.
* **Modern Stack**: Utilizing the MERN stack (MongoDB, Express.js, React.js, Node.js), Smart Dorm benefits from a scalable, efficient, and flexible architecture, allowing for rapid development and deployment.

#### Overall System Description

* 1. Project Background

The "Smart Dorm" project emerges in response to challenges in hostel management. Traditional methods have been marked by inefficiencies, manual processes, and communication gaps. This project aims to leverage modern technology to address these issues, making hostel life more efficient, safe, and connected.

* 1. Project Objectives

The primary objectives of the "Smart Dorm" project are as follows:

* Streamline hostel management operations: Optimize and simplify the administrative tasks and processes in hostel management for greater efficiency.
* Enhance security measures: Improve security protocols and mechanisms to ensure the safety of residents and their belongings.
* Foster efficient communication: Facilitate effective and rapid communication between residents and administrators.
* Simplify leave request processing: Make the process of requesting and approving leave more straightforward and efficient.
* Implement a Lost and Found portal: Create a system for reporting and retrieving lost items, streamlining the process for residents.
* Provide real-time room availability updates: Offer up-to-the-minute information on room availability and occupancy.
  1. Stakeholders

Stakeholders in the "Smart Dorm" project include:

* Hostel administrators
* Residents
* Development and maintenance team
* Supervisor and Co.
* Evaluators
  1. Operating Environment

The software will operate in the following environment:

Hardware Platform

* Use: Accessible via standard computer systems such as desktops and laptops for administrators, with responsive design for end-users on smartphones and tablets.

Operating System

* Compatibility: Cross-platform compatibility (Windows, macOS, Linux) for development and operational use, enabling access from any OS with a web browser.

Network Environment

* Servers: Utilizes centralized cloud servers for scalable and efficient data management.
* Security: Emphasizes network security with encryption, secure access protocols, and continuous monitoring to protect sensitive data and prevent unauthorized access.

APIs: APIs play a crucial role in retrieving real-time data and facilitating seamless integration with various components of the hostel management system.

* 1. System Constraints

The "Smart Dorm" project operates under the following constraints:

Software Constraints:

* **Framework and Tools Compatibility**: Smart Dorm is built using the MERN stack, which dictates compatibility requirements with MongoDB, Express.js, React.js, and Node.js. The system adheres to the limitations and capabilities of these technologies.
* **Programming Languages and Databases**: The application is developed with JavaScript and utilizes MongoDB as its database, requiring alignment with their syntax, functionalities, and best practices.

Hardware Constraints:   
Device Compatibility: Smart Dorm's software components are optimized for operation on standard computing devices, including desktops, laptops, and mobile phones, ensuring broad accessibility.

Legal Constraints:

Data Protection and Privacy: The system is compliant with data protection, ensuring user data is handled securely and privacy is maintained.

Environmental Constraints:   
Operational Versatility: Designed to be operational in diverse environmental settings, Smart Dorm ensures consistent functionality in both quiet and noisy backgrounds, accommodating various user scenarios

User Constraints:

* **University Affiliation**: Access to Smart Dorm is restricted to students of the associated university, requiring valid identification for user registration and system access.
* **Language Proficiency**: The primary language of the Smart Dorm interface is English, necessitating basic proficiency by users to navigate and utilize the system effectively.

Cultural Constraints:

The system will be in English; thus, the user must at least be familiar with it.

* 1. **Assumptions & Dependencies**For the Smart Dorm project, our progress and development are guided by certain assumptions and dependencies crucial for its successful implementation:

**Assumptions**

* **Availability of Standard Computer Systems**: We assume that our development and testing teams have access to standard computer systems that are essential for the software development lifecycle.
* **Operating System Access**: It's assumed that the team has access to necessary operating systems (Windows, macOS, Linux) for both development and testing purposes, ensuring cross-platform compatibility.
* **Stable Network Connectivity**: The project presupposes stable internet access for both the development team and the end-users (administrators and residents) to facilitate seamless development, testing, and eventual deployment.

**Dependencies**

* **Hardware and Software Reliability**: Smart Dorm’s development and operational effectiveness depend on the reliable performance of chosen hardware and software components, including servers, development machines, and the MERN stack technologies.
* **Compliance with Legal Standards**: The success of the project is significantly dependent on adhering to legal and regulatory frameworks governing data protection and privacy

1. **Project Breakdown:**At the beginning of the seventh semester, our team gathered to brainstorm ideas. After exploring various collaborative websites, we came up with the idea of Smart Dorm. In August 2023, we submitted an abstract of our idea to our supervisor for the first phase. Following that, we began gathering requirements. In the second phase, around September, we had our first defense where we presented our idea along with details such as frontend designs and backend functionalities. By November 2023, during the third phase, we had completed 50% of our implementation along with SRS and SDS and showcased it to the committee members. Now, in March 2024, as we write our report, we have accomplished 100% of our work, although some minor functionalities are still in progress.

| **Phase** | **Timeframe** | **Milestones and Activities** |
| --- | --- | --- |
| Brainstorming | Beginning of 7th semester | - Team gathered to brainstorm ideas |
|  |  | - Explored various collaborative websites |
|  |  | - Conceptualized the idea of Smart Dorm |
| Abstract | August 2023 | - Submitted abstract of the idea to supervisor for review |
| Submission |  |  |
| Requirements | August - September 2023 | - Started gathering requirements for the project |
| Gathering |  |  |
| First Defense | September 2023 | - Presented idea, frontend designs, and backend |
|  |  | functionalities |
| Implementation | September - | - Initiated implementation phase |
|  | November 2023 | - Completed 50% of the implementation |
| SRS and SDS | November 2023 | - Completed Software Requirements Specification (SRS) |
|  |  | and Software Design Specification (SDS) |
|  |  | - Showcased progress to committee members |
| Finalization | March 2024 | - Completed 100% of the work |
|  |  | - Minor functionalities still in progress |

**Table 1:** Project Breakdown

1. **Proposed Methodology**

Proposed Methodology for Smart Dorm Project Using the MERN Stack

**Workflow**:

* **Requirement Analysis and Data Compilation:**

Kick off the project with a deep dive into the functional and non-functional requirements specific to the Smart Dorm application. This step involves understanding the needs of all stakeholders involved, including students (residents), hostel administrators, and technical support teams.

Compile essential data and requirements such as user data models, room details, leave and maintenance request workflows, security requirements, and real-time communication needs.

* **Selection of Tools and Technologies:**

Confirm the use of the MERN stack as the core technology suite, leveraging MongoDB for database management, Express.js and Node.js for the backend framework and server environment, and React for the frontend development.

Identify supplementary tools for version control (Git), testing (Jest for backend and frontend testing, React Testing Library for React components), and deployment (Docker for containerization, AWS or Heroku for hosting).

* **Architectural Design and Framework Selection:**

Design a system architecture that supports the Smart Dorm's scalability, security, and real-time data handling needs. Opt for a RESTful API design to facilitate communication between the frontend and backend.

Choose React for its component-based architecture to handle the dynamic nature of the user interface, Express.js for routing and middleware functionalities, and MongoDB for its schema-less nature which allows flexibility in handling diverse data types.

* **Development Using the MERN Stack:**

Initiate frontend development with React, focusing on creating a responsive and intuitive user interface that aligns with the user stories and use cases identified earlier. Implement real-time updates for room availability and communications using Socket.IO.

Develop the backend with Node.js and Express.js, creating RESTful APIs that interact with the MongoDB database for CRUD operations.

* **Building and Integrating Backend and Frontend:**

Develop and test API endpoints in the backend for functionality such as room allocations, leave requests, and real-time chat, ensuring they meet performance and security benchmarks.

Seamlessly integrate these APIs with the React frontend, ensuring that data flow and user interactions are smooth and performant across all application features.

* **System Testing and Validation:**

Implement comprehensive testing across both backend and frontend components, including unit testing and integration testing.

* **Deployment and Launch:**

Prepare the application for deployment, setting up the production environment on platforms configuring databases, and ensuring all environmental variables and dependencies are properly managed.

Deploy the Smart Dorm application, monitor its performance, and address any immediate issues post-launch.

#### Functional Requirements

* 1. Functional Hierarchy

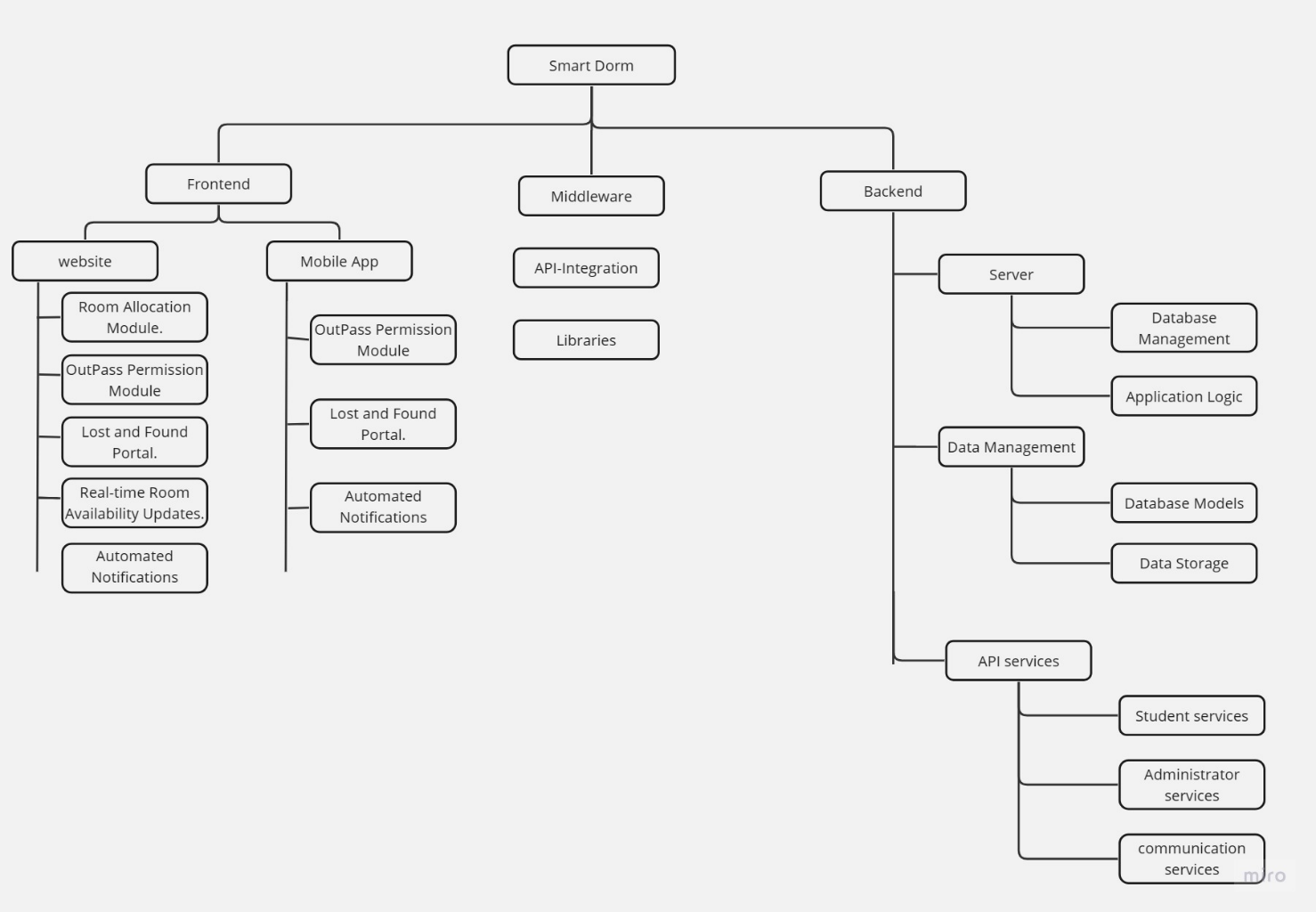
All possible features of System are listed in below diagram

Figure 1 Functional Hierarchy Diagram

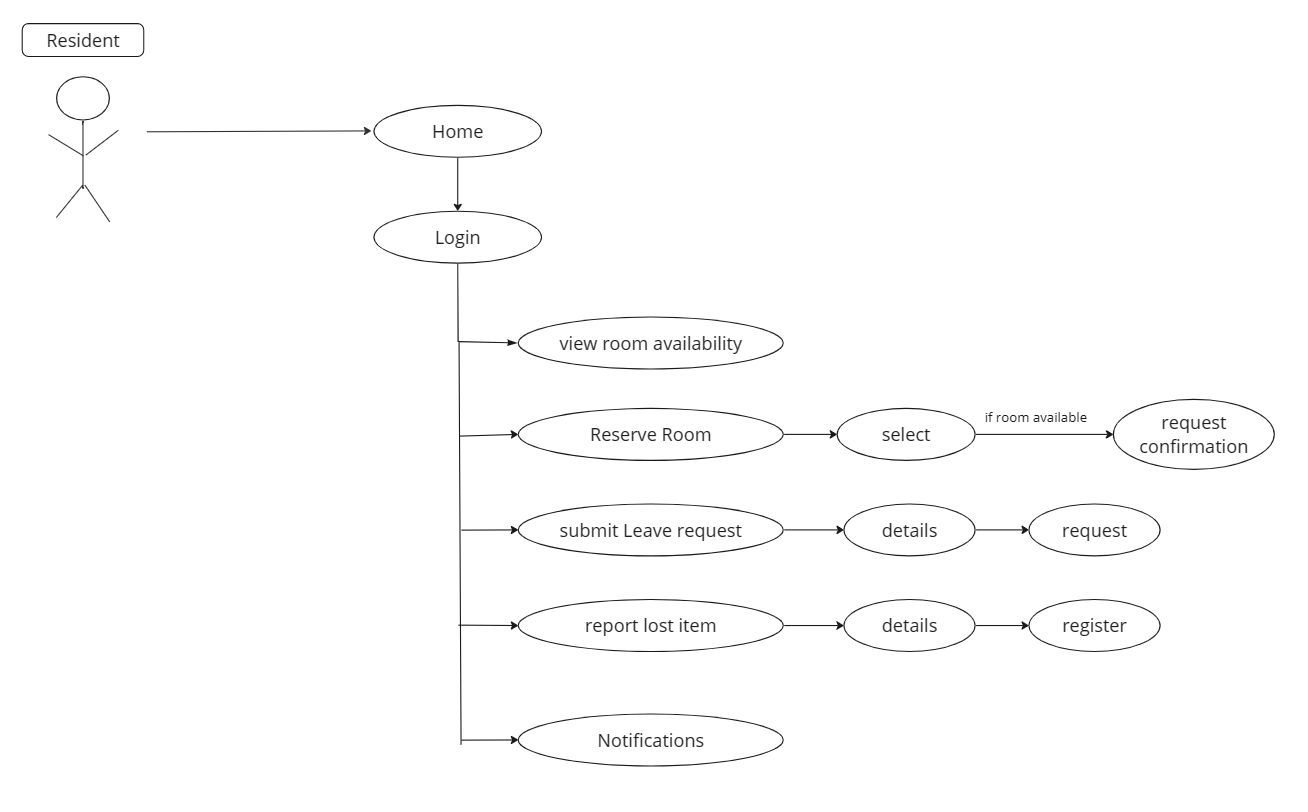
User class 1: the resident interaction

Figure 2 User Class of Resident Interaction

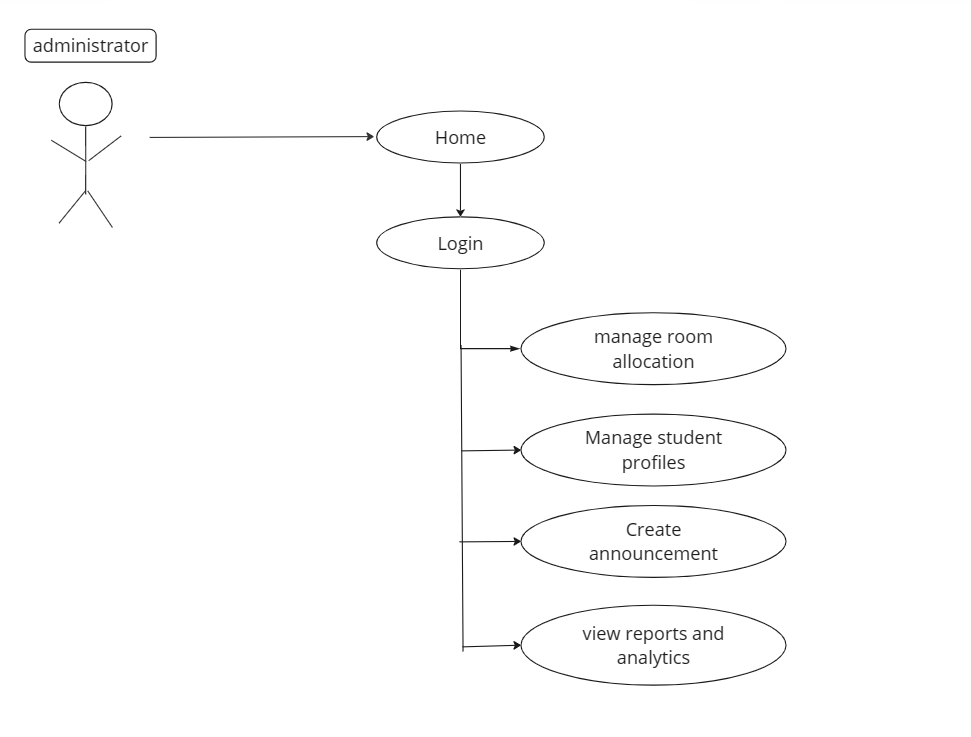
User class 2: the Administration interaction

Figure 3 User Class of Administrative Interaction

6. 2 Use Cases

6.2.1 Room Allocation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UC-001: Room Allocation** | | | | |
| **Use case Id:** | | 01 | | |
| **Actors:** | | Hostel administrators, Residents | | |
| **Feature:** | | Room allocation | | |
| **Pre-condition:** | | System is operational, user authentication is successful | | |
| **Scenarios** | | | | |
| **Step#** | **Action** | | | **Software Reaction** |
| **1.** | Hostel administrator selects room allocation | | | System displays room allocation options. |
| **2.** | Hostel administrator chooses room preferences. | | | System checks room availability. |
| **3.** | System allocates a room based on preferences. | | | System updates room availability status. |
| **Alternate Scenarios:** | | | | |
| **1:** User authentication fails. System prompts for valid credentials.  **2:** Room preferences exceed availability. System suggests alternative options. | | | | |
| **Post Conditions** | | | | |
| **Step#** | **Description** | | | |
| **1.** | Room allocated successfully. | | | |
| **2.** | Room availability updated. | | | |
| **Use Case Cross referenced** | | | None | |

Table 2 Room Allocation

6.2.2 Leave Request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **UC-002: Leave Request Processing** | | | | |
| **Use case Id:** | | | | 02 |
| **Actors:** | | | | Hostel administrators, Residents |
| **Feature:** | | | | Leave request processing |
| **Pre-condition:** | | | | System is operational, user authentication is successful |
| **Scenarios** | | | | |
| **Step#** | | **Action** | | **Software Reaction** |
| **1.** | | Resident submits a leave request | | System records the leave request. |
| **2.** | | Hostel administrator reviews the leave request. | | System displays the request details. |
| **3.** | | Hostel administrator approves or denies the request. | | System sends a notification to the resident with the decision. |
| **Alternate Scenarios:** | | | | |
| **1**    **2** | Resident submits an incomplete leave request. System prompts for necessary information.  Hostel administrator requests additional information. System notifies the resident of the request. | | | |
| **Post Conditions** | | | | |
| **Step#** | | **Description** | | |
|  | | Leave request processed with an approval or denial. | | |
|  | | Notification sent to the resident regarding the decision. | | |
| **Use Case Cross referenced** | | | None | |

Table 3 Leave Request Processing

6.2.3 Lost and Found report

|  |  |  |  |
| --- | --- | --- | --- |
| **UC-003: Lost and Found Report** | | | |
| **Use case Id:** | | | 03 |
| **Actors:** | | | Residents |
| **Feature:** | | | Lost and found reporting |
| **Pre-condition:** | | | System is operational, user authentication is successful |
| **Scenarios** | | | |
| **Step#** | **Action** | | **Software Reaction** |
| **1.** | Resident reports a lost item. | | System records the lost item report. |
| **2.** | Other residents or administrators find a lost item and report it. | | System records found items. |
| **3.** | Resident contacts the finder to retrieve the item. | | Resident receives a notification about a found item matching their lost item description. |
| **Alternate Scenarios:** | | | |
| **1:** Resident submits an incomplete lost item report. System prompts for necessary information. | | | |
| **Post Conditions** | | | |
| **Step#** | **Description** | | |
|  | Lost and found items are documented. | | |
|  | Residents are notified of matching lost and found items for recovery. | | |
| **Use Case Cross referenced** | | None | |

Table 4 Lost and Found Report

6.2.4 Room Availability Updates

|  |  |  |
| --- | --- | --- |
| **UC-004:Room Availability Updates** | | |
| **Use case Id:** | | 04 |
| **Actors:** | | Hostel administrators, Residents |
| **Feature:** | | Room availability updates |
| **Pre-condition:** | | System is operational, user authentication is successful |
| **Scenarios** | | |
| **Step#** | **Action** | **Software Reaction** |
| **1.** | Resident checks room availability. | System displays current room availability. |
| **2.** | Hostel administrator updates room availability status. | System reflects the changes in real-time. |
| **Alternate Scenarios:** No alternate scenarios | | |
| **Post Conditions** | | |
| **Step#** | **Description** | |
|  | Current room availability status is displayed and updated as needed. | |

**Table 5**Room Availability Updates

**6.2.5 Real-Time Communication**

|  |  |  |  |
| --- | --- | --- | --- |
| **UC-005:Real-Time Communication** | | | |
| **Use case Id:** | | | 05 |
| **Actors:** | | | Hostel administrators, Residents |
| **Feature:** | | | Real-Time Communication |
| **Pre-condition:** | | | System is operational, user authentication is successful |
| **Scenarios** | | | |
| **Step#** | **Action** | | **Software Reaction** |
| **1.** | Resident initiates a real-time chat | | System creates a chat session. |
| **2.** | Resident selects the recipient (another resident or hostel administrator). | | System identifies the recipient. |
| 3. | Resident sends a message. | | System delivers the message to the recipient... |
| **Alternate Scenarios:** | | | |
| **1:** Resident initiates a group chat. System creates a group chat session.  **2:** Resident sends a file attachment. System uploads and delivers the file.  **3**: Recipient is offline. System stores the message for future delivery. | | | |
| **Post Conditions** | | | |
| **Step#** | **Description** | | |
|  | Real-time communication session successfully initiated. | | |
|  | Messages delivered, read, and responded to. | | |
|  |  | | |
| **Use Case Cross referenced** | | None | |

**Table 6**Real Time Communication

#### 7. Non-functional Requirements

7.1 Performance Requirements

The performance of the Smart Dorm system is crucial for a seamless user experience.

The system should:

* Ensure a rapid response time to user interactions.
* Support concurrent access by users without significant performance degradation.
* Maintain an uptime to ensure availability.

7.2 Safety Requirements:

The system must address safety concerns, including:

* Implementation of data encryption during transmission and storage to safeguard user data.
* User authentication with secure password storage to prevent unauthorized access.
* Measures to prevent data breaches and protect the privacy of users.
* Compliance with safety certifications, policies, and relevant regulations in the hostel management domain.

7.3 Security Requirements:

Security is paramount to the Smart Dorm system to protect user data and system integrity. The system should include:

* Robust user authentication and authorization mechanisms to control access to sensitive data and features.
* Safeguards to prevent unauthorized access, data breaches, and cyber threats.

7.4 User Documentation:

Comprehensive user documentation is essential for a user-friendly experience. The system will provide:

* User manuals to guide users through system functionalities and processes.
* Online help resources for instant assistance and problem resolution.
* Context-sensitive help within the user interface to aid users in understanding specific features.

# 8. Detailed Design Architecture

**Client-Side Application**

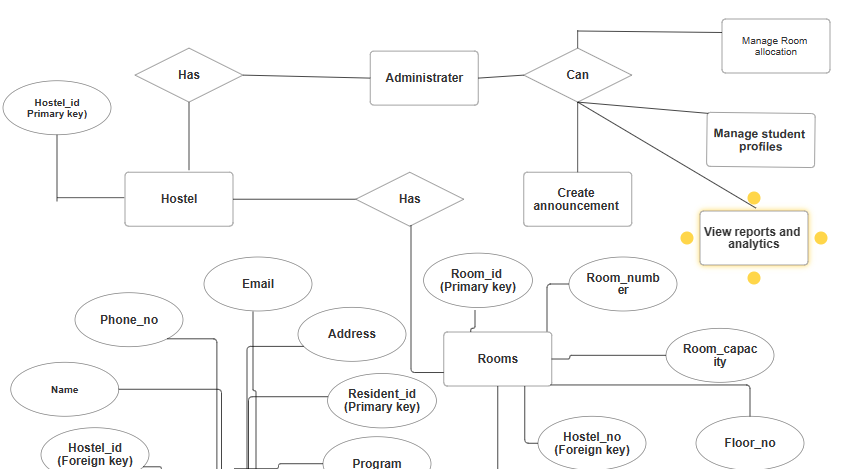
The client-side application component of the Smart Dorm project is responsible for rendering the user interface and managing user interactions. It serves as the front end of the system, providing users with a visually appealing and interactive interface to interact with the hostel management functionalities.

**Server-Side Backend**

The server-side backend component of the Smart Dorm project serves as the core engine that powers the system's functionality. It handles critical operations such as user authentication, session management, real-time collaboration features, and interaction with the database. The backend is responsible for processing client requests, executing business logic, and delivering data to the client-side application.

**Architecture Design Approach**

The architecture design approach for the Smart Dorm project follows a client-server model, integrating modern web technologies and real-time communication features. On the client side, React.js is utilized for dynamic UI rendering, ensuring a responsive and intuitive user experience. WebSocket technology facilitates real-time communication between clients and the server, enabling seamless interaction and synchronization of data. The backend, powered by Node.js and Express.js, manages the system's state and handles communication between clients. Data storage is handled by MongoDB, ensuring efficient storage and retrieval of hostel management data. This architectural setup ensures scalability, reliability, and real-time responsiveness, providing a solid foundation for efficient hostel management operations.

**8.1. Entity Relationship Diagram**   
The ER diagram for the Smart Dorm project depicts the essential entities and relationships for efficient hostel management. Entities include User, Hostel, and Room, with attributes like Resident\_id and Room\_id. Relationships like User-Hostel and Room-User signify residency and room allocation. The diagram visually encapsulates the structural framework, highlighting key elements crucial for streamlined hostel operations.  


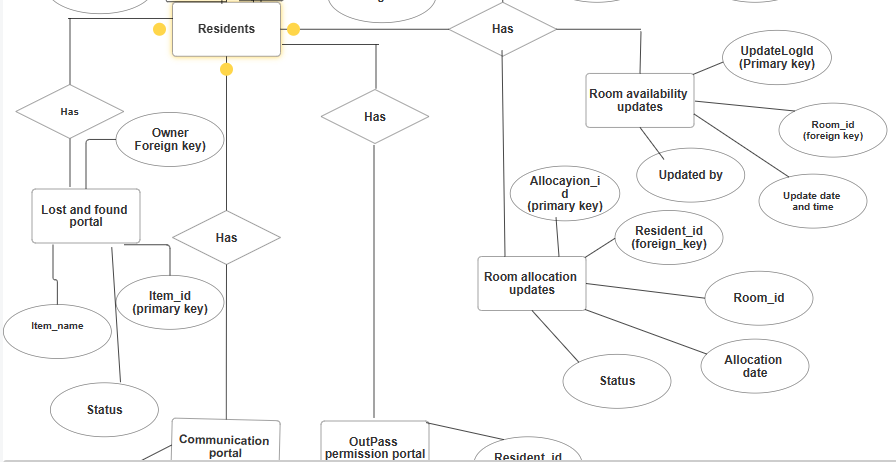
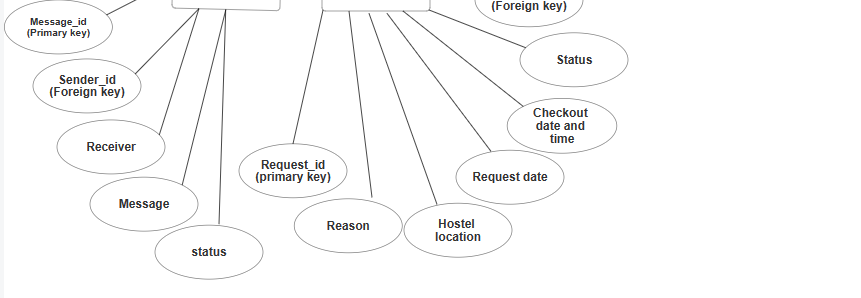
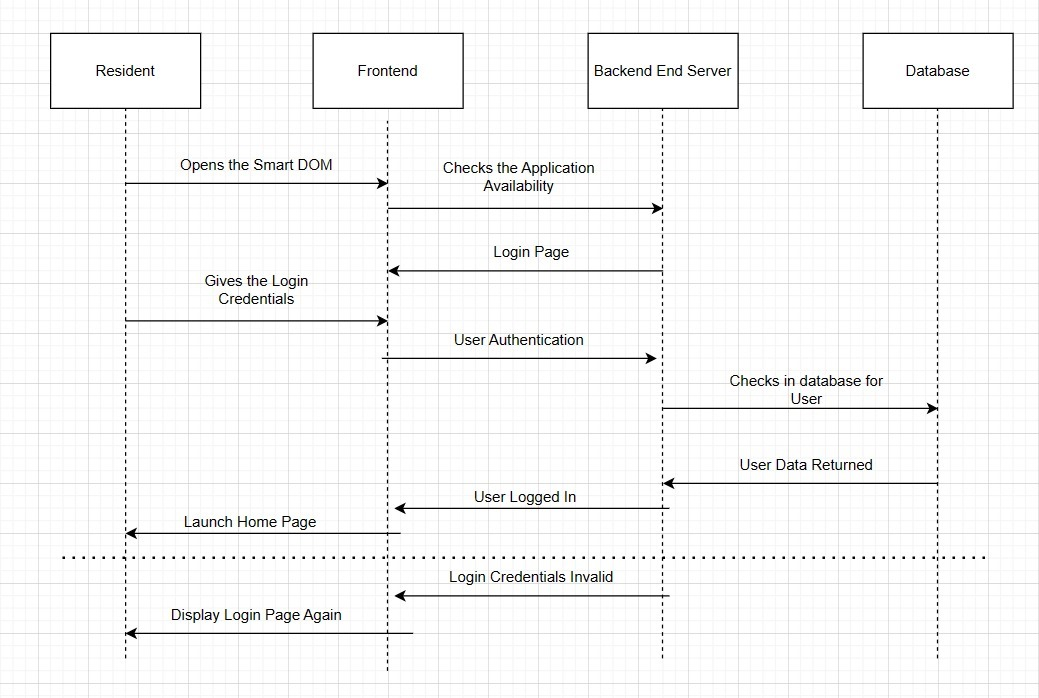
  


Figure: 4 Entity Relationship Diagram.

**8.2. Sequence Diagram:**

* + 1. **Log in sequence diagram:**  
       The Login Sequence Diagram in Smart Dorm showcases user authentication, where inputting valid credentials grants system access. In case of invalid credentials, users are prompted for the correct information. To enhance security, the system enforces a temporary account lockout after repeated login failures. Successful login provides access to the Smart Dorm interface, while unsuccessful attempts trigger user notification and prompt for valid credentials.  
         
         
         
        Figure 5.1 Login Sequence Diagram

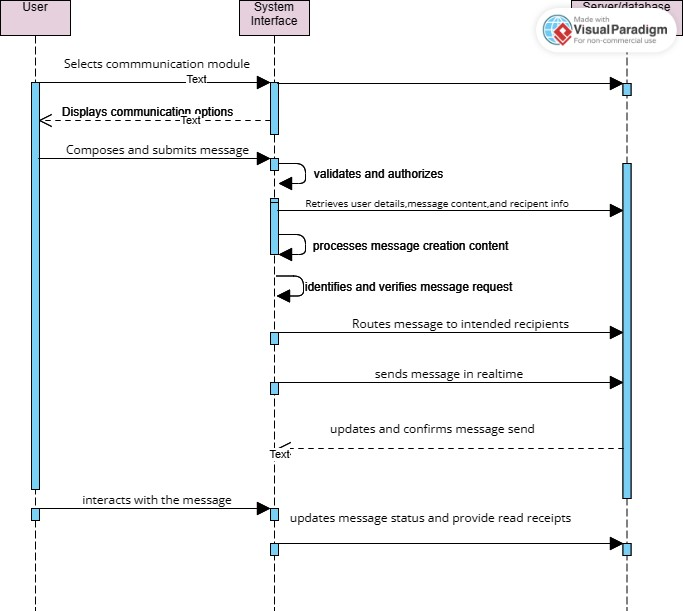
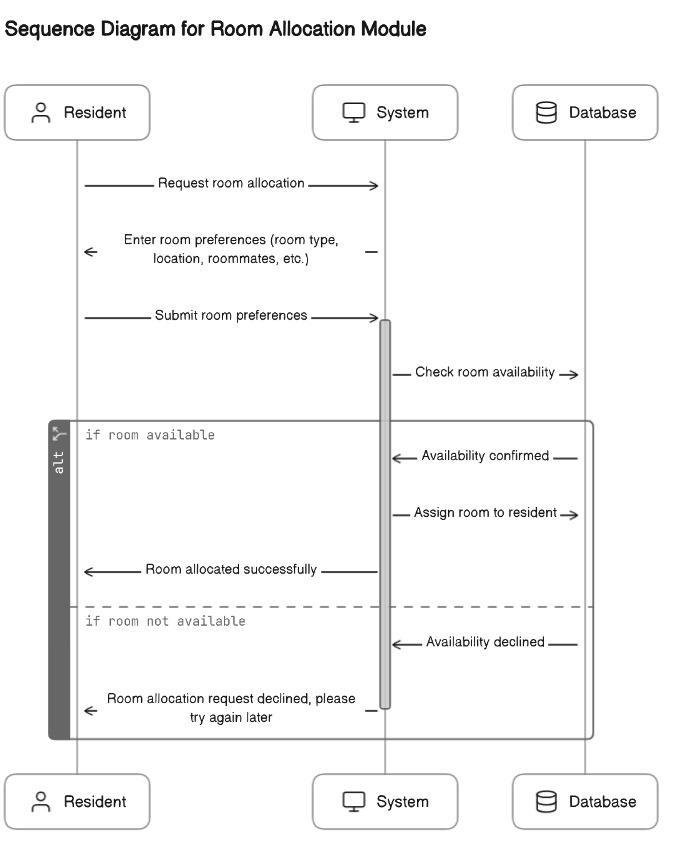
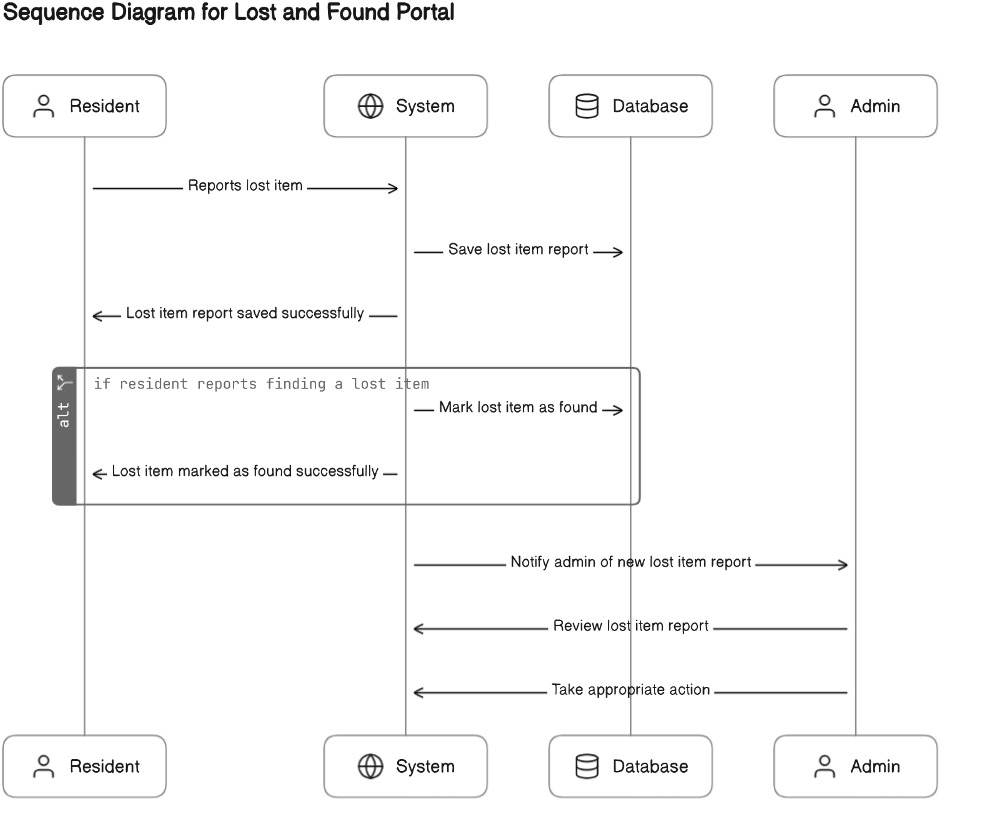
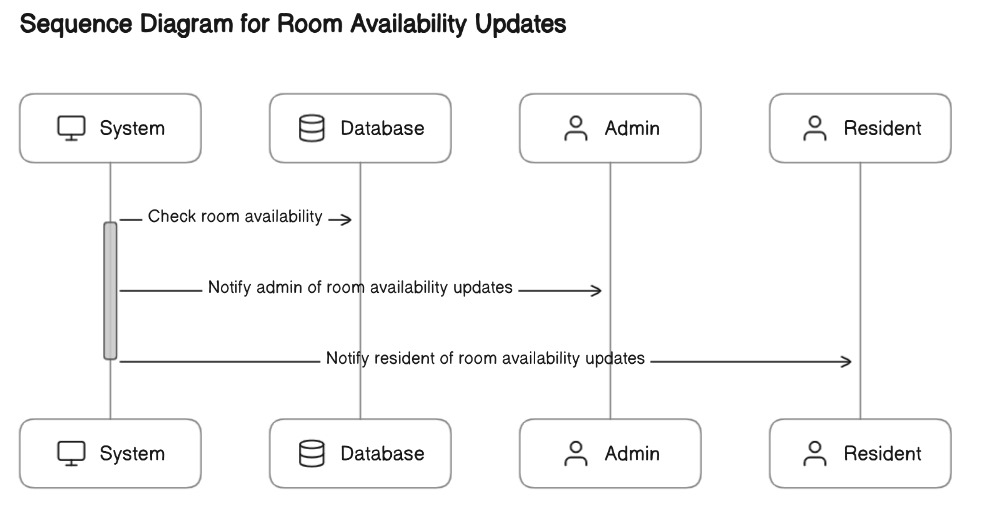
**8.2.2 Communication module sequence diagram:**The Communication Module Sequence Diagram illustrates resident-initiated real-time chats, including features like group chats, file attachments, and offline message storage. It provides a succinct overview of the dynamic communication processes within the Smart Dorm system.  
  


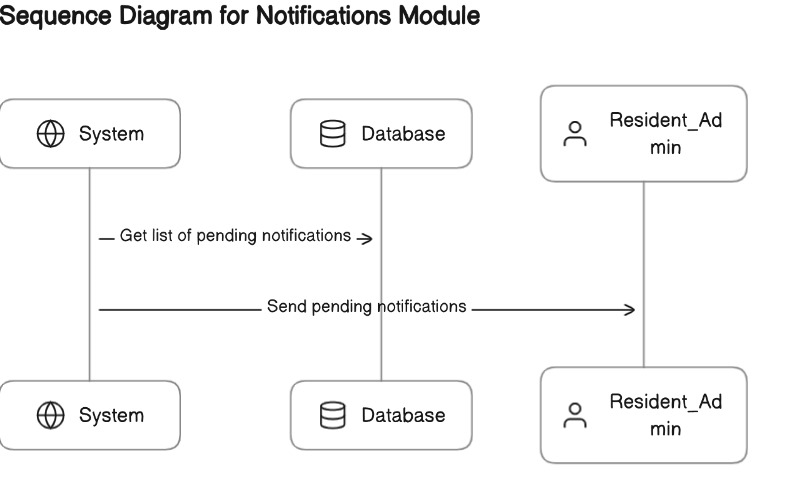
Figure: 5.2 Communication module sequence diagram.

**8.2.3** **Room Allocation Module Sequence Diagram:**The Room Allocation Module Sequence Diagram showcases hostel administrators selecting room allocation, checking availability, and updating status. It provides a concise overview of dynamic room allocation processes in the Smart Dorm system.  
  
  
  
 Figure: 5.3 Room allocation module sequence diagram.

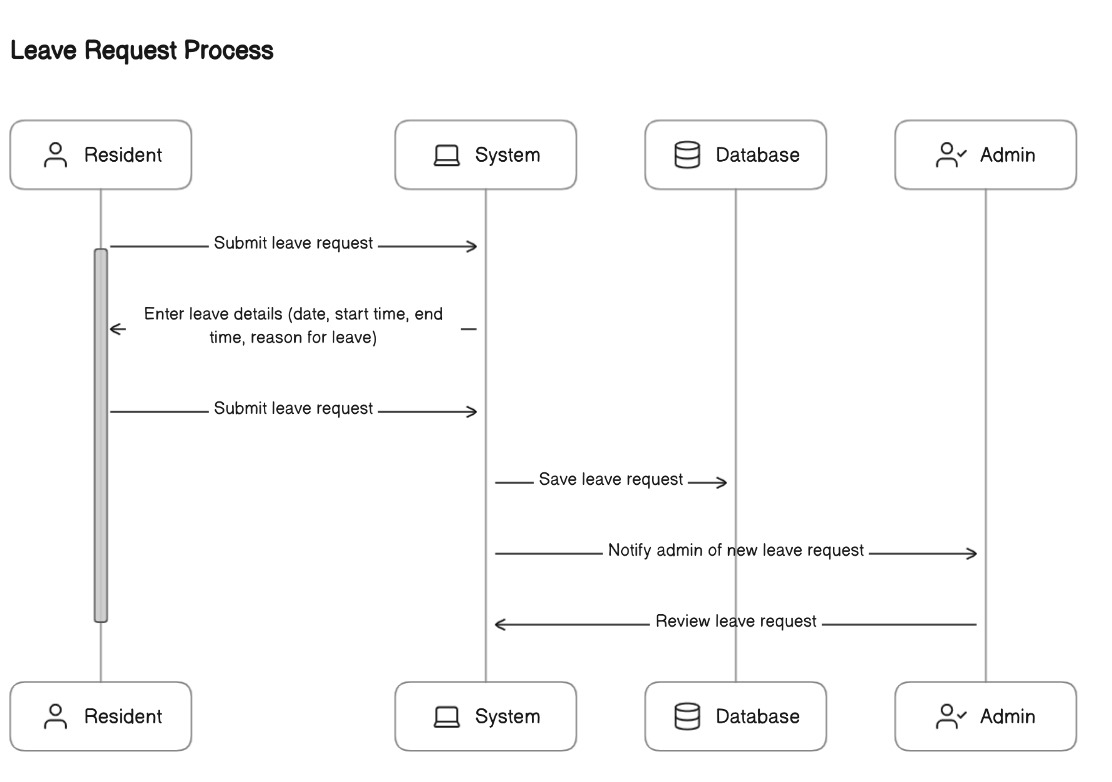
**8.2.4 Lost and Found module sequence diagram:**The Lost and Found Module Sequence Diagram succinctly illustrates resident-initiated reporting of lost items, administrators documenting found items, and residents receiving notifications about matching lost and found items for recovery. This diagram provides a brief overview of the dynamic processes in the Smart Dorm system related to lost and found items.  
  
  
  
 Figure: 5.4 Lost and Found module sequence diagram.

**8.2.5 Room Availability Updates Sequence Diagram:**

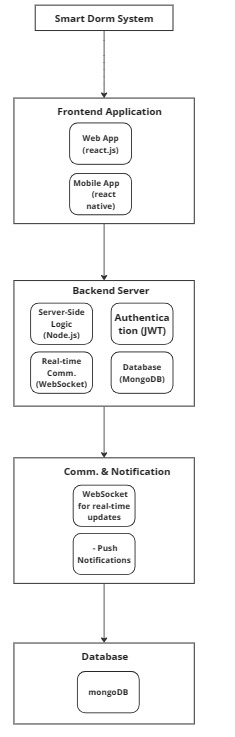
The Room Availability Updates Module Sequence Diagram showcases residents checking room availability, hostel administrators updating availability status, and real-time reflection of changes in the system. This diagram offers a concise depiction of the dynamic processes related to room availability updates within the Smart Dorm system.  
  
 Figure: 5.5 Room availability updates sequence diagram.

**8.2.6 Notification Module Sequence Diagram:**The Notification Module Sequence Diagram outlines the steps involved in initiating and processing notifications within the Smart Dorm system. It includes the creation of notifications, system delivery, and user receipt. This diagram provides a brief overview of the dynamic processes related to notifications within the Smart Dorm system.  
  
  
 Figure: 5.6 Notification Module sequence diagram.

**8.2.7 Leave Request Module Sequence Diagram:**

The Leave Request Module Sequence Diagram illustrates a resident submitting a leave request, administrators reviewing the request, and the subsequent approval or denial process with notification delivery. It provides a succinct overview of the dynamic processes related to leave requests within the Smart Dorm system  
  
  
 Figure: 5.7 Leave request module sequence diagram.

* 1. **Architecture Design Diagram:**The architecture design diagram for Smart Dorm provides a concise overview of the system's structure, showcasing the interaction between user interface, application logic, and the database. It illustrates the flow of user requests through the application logic layer for processes like room allocation and communication, emphasizing the efficient coordination between system components.

  
 Figure: 6.0 Architecture Design Diagram.

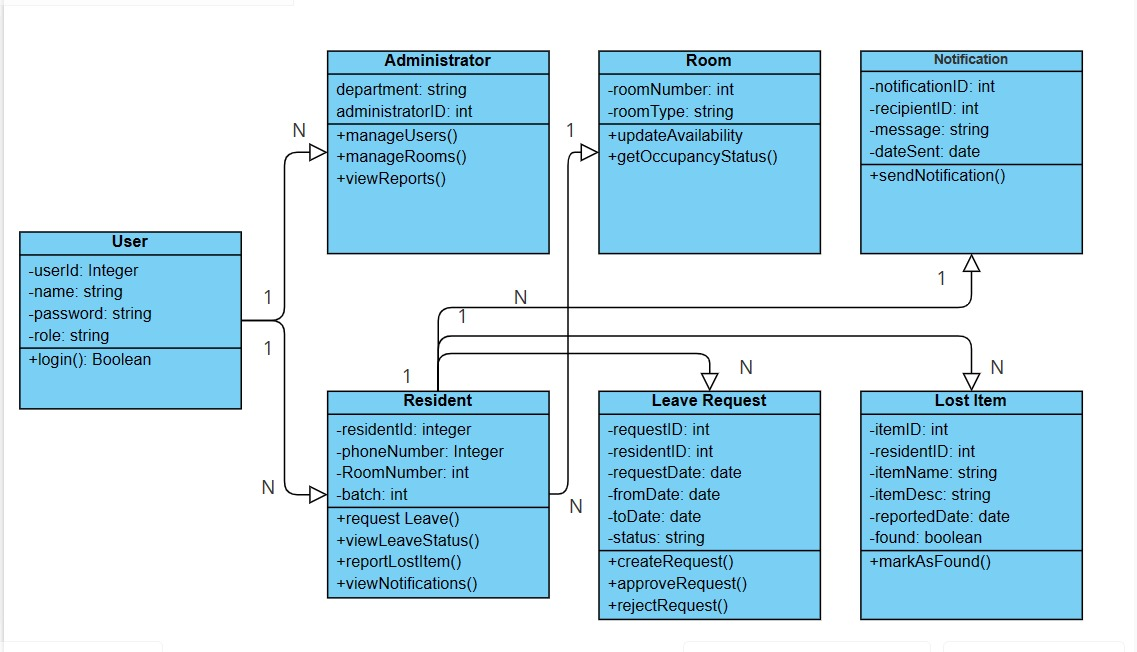
* 1. **Database Diagram:**The diagram illustrates the organized storage of user details, hostel information, and room allocations, facilitating efficient data management within the Smart Dorm system.



Figure: 7.0 Database Diagram

* 1. **Class Diagram**:

The Class Diagram for the Smart Dorm project succinctly outlines the system's static structure by representing classes, their attributes, and relationships. It encapsulates essential entities like User, Hostel, and Room, showcasing their respective attributes and associations. The diagram visually communicates the core components of the Smart Dorm system, providing a foundational understanding of how classes and their interactions contribute to the overall structure and functionality of the hostel management system.

Figure: 8.0 Class Diagram.

**8.6. Test cases:**

**8.6.1 Room Allocation Module:**

**Test Case1: Successful Room Allocation**

|  |  |
| --- | --- |
| Steps: | Description |
| 1 | Log in as an administrator. |
| 2 | Access the Room Allocation Module. |
| 3 | Allocate a room to a resident based on preferences. |
| 4 | Verify that the room allocation is successful. |

**Test Case2**: **Room Availability Update**

|  |  |
| --- | --- |
| Steps: | Description |
| 1 | Log in as an administrator. |
| 2 | Access the Room Allocation Module. |
| 3 | Mark a room as unavailable. |
| 4 | Verify that the room availability is updated in real-time. |

**8.6.2 Communication Module:**

**Test Case3**: **Sending a Message.**

|  |  |
| --- | --- |
| Steps: | Description |
| 1 | Log in as a user. |
| 2 | Access the Communication Module. |
| 3 | Compose and send a message to another user. |
| 4 | Verify that the message is sent successfully. |

**Test Case4**: **Real-time Messaging**

|  |  |
| --- | --- |
| Steps: | Description |
| 1 | Log in as two users on different devices. |
| 2 | Access the Communication Module for both users. |
| 3 | Send a message from one user. |
| 4 | Verify that the other user receives the message in real-time. |

**8.6.3 Out Pass Permission Module:**

**Test Case5: Submitting Leave Request**

|  |  |
| --- | --- |
| Steps: | Description |
| 1 | Log in as a resident. |
| 2 | Access the Out Pass Permission Module. |
| 3 | Submit a leave request. |
| 4 | Verify that the leave request is recorded. |

**Test Case6: Leave Request Approval**

|  |  |
| --- | --- |
| Steps: | Description |
| 1 | Log in as an administrator. |
| 2 | Access the Out Pass Permission Module. |
| 3 | Approve a leave request. |
| 4 | Verify that the resident receives notification of the approval. |

**8.6.4 Lost and Found Portal:**

**Test Case7**: **Reporting a Lost Item**

|  |  |
| --- | --- |
| Steps: | Description |
| 1 | Log in as a user. |
| 2 | Access the Lost and Found Portal. |
| 3 | Report a lost item. |
| 4 | Verify that the lost item report is recorded. |

**Test Case8**: **Returning a Found Item**

|  |  |
| --- | --- |
| Steps: | Description |
| 1 | Log in as a user. |
| 2 | Access the Lost and Found Portal. |
| 3 | Report a found item and return it. |
| 4 | Verify that the owner receives notification of the found item. |

9. Milestones Achieved:

Milestones Achieved for the Smart Dorm Project

9.1. Project Initialization and Planning

Achieved: Defined project scope, objectives, and requirements through comprehensive analysis.

Methodology: Employed Agile development methodologies for flexible, iterative progress.

Results: Established a solid foundation for the project with a clear roadmap and Agile sprints planned.

9.2. System Architecture Design

Achieved: Designed a robust system architecture suitable for the Smart Dorm application.

Methodology: Followed best practices in software architecture, focusing on scalability, security, and performance.

Tools & Techniques: Created architectural diagrams and models using tools like Lucidchart.

Results: A scalable and secure architecture that supports all intended functionalities of the Smart Dorm project.

9.3. Development of Core Functionalities

Achieved: Completed the development of key functionalities such as room allocation, real-time communication, and leave requests.

Methodology: Used the MERN stack (MongoDB, Express.js, React, Node.js) for full-stack development.

Tools & Techniques: React for the frontend, Node.js and Express.js for the backend API, MongoDB for the database. Employed RESTful API principles for efficient server-client communication.

Results: Successfully implemented and tested core features, ensuring they perform as expected and provide a smooth user experience.

9.4. Implementation of Real-Time Features

Achieved: Developed real-time communication and notifications for room availability updates.

Methodology: Integrated WebSockets for real-time bidirectional communication between clients and the server.

Tools & Techniques: Used Socket.IO with Node.js to facilitate real-time features without needing to refresh the browser.

Results: Enabled live updates and real-time chat functionality, enhancing user interaction within the Smart Dorm application.

9.5. User Interface Design and Usability Testing

Achieved: Designed an intuitive and responsive user interface, followed by usability testing with potential users.

Methodology: Adopted user-centered design principles to ensure the interface is accessible and easy to use.

Tools & Techniques: React was used for the frontend to create a dynamic and responsive design. Conducted usability tests to gather feedback and make iterative improvements.

Results: A user-friendly interface that has been positively received by test users, with adjustments made based on their feedback to improve navigation and functionality.

9.6. System Testing and Validation

Achieved: Conducted extensive testing across the application to ensure reliability and performance.

Methodology: Implemented a mix of unit testing, integration testing, and end-to-end testing, following test-driven development (TDD) practices.

Tools & Techniques: Used Jest for backend testing, React Testing Library for frontend components, and Postman for API testing.

Results: Identified and resolved bugs and performance issues, ensuring the application meets all functional requirements and user expectations.

9.7. Preliminary Deployment and Feedback Collection

Achieved: Deployed the application to a staging environment for preliminary user testing and feedback collection.

Methodology: Early deployment to gather real-world user feedback and identify areas for improvement before the final launch.

Tools & Techniques: Utilized Google Forms and direct interviews for feedback collection.

Results: Gained valuable insights into user experiences and preferences, leading to several important adjustments to improve usability and functionality.

10. Results and Discussion:

10.1. Website Application

10.1.1 Login Screen

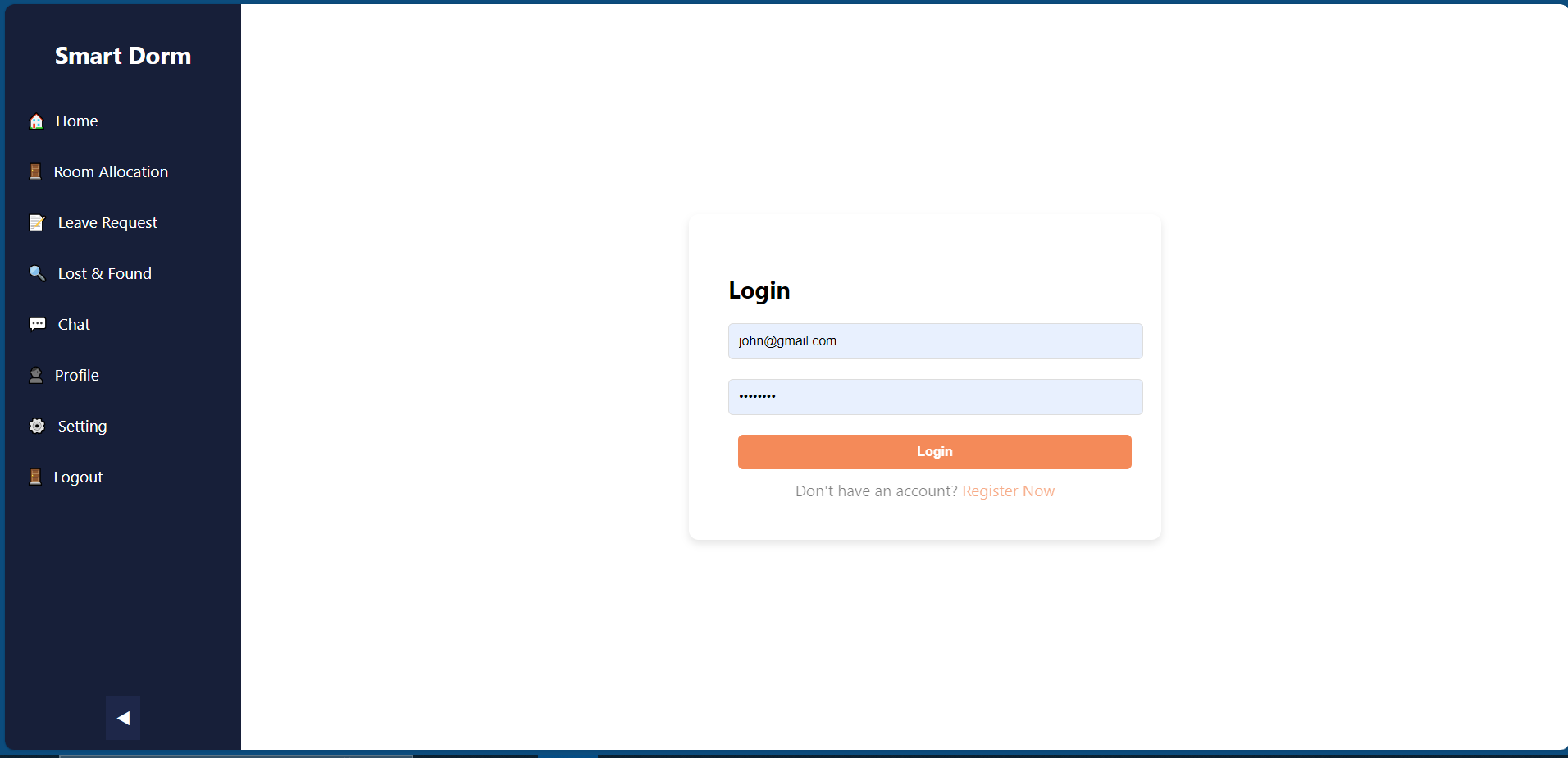


Figure: 9.0 Login Screen

10.1.2 Signup Screen

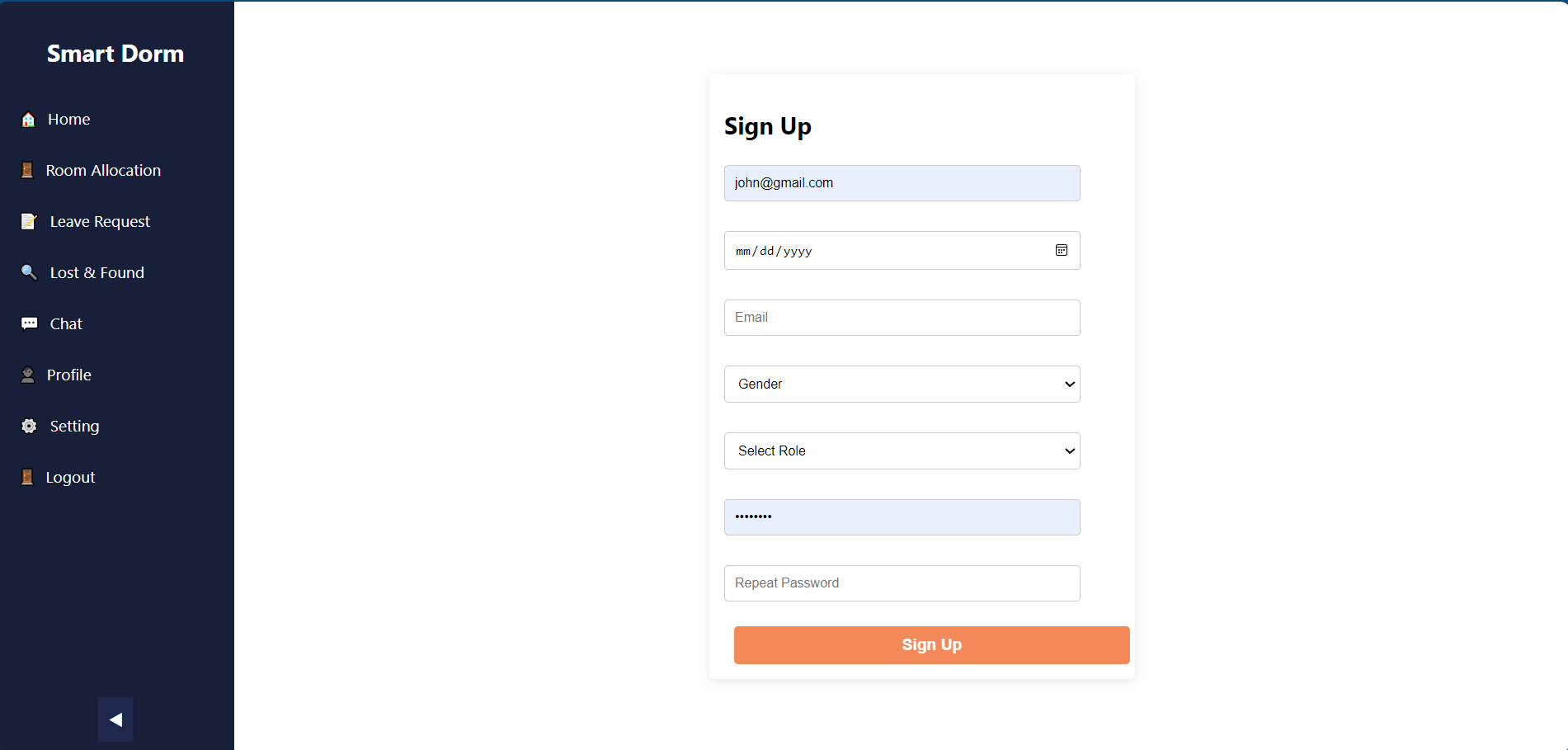


Figure: 10 Signup Screen

10.1.3 Room Allocation Screen

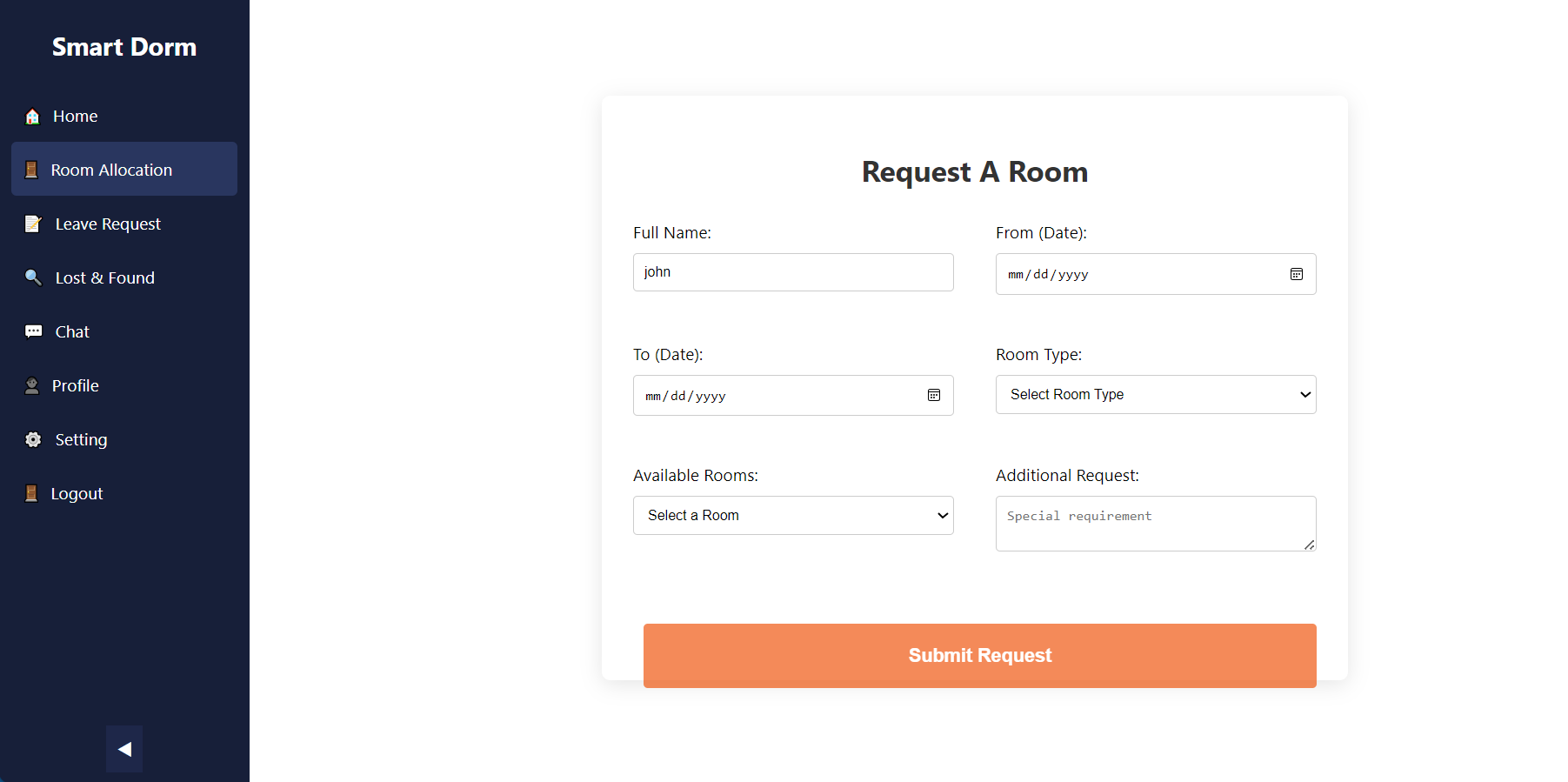


Figure: 11 Room Allocation Screen

10.1.4 Leave Request Screen

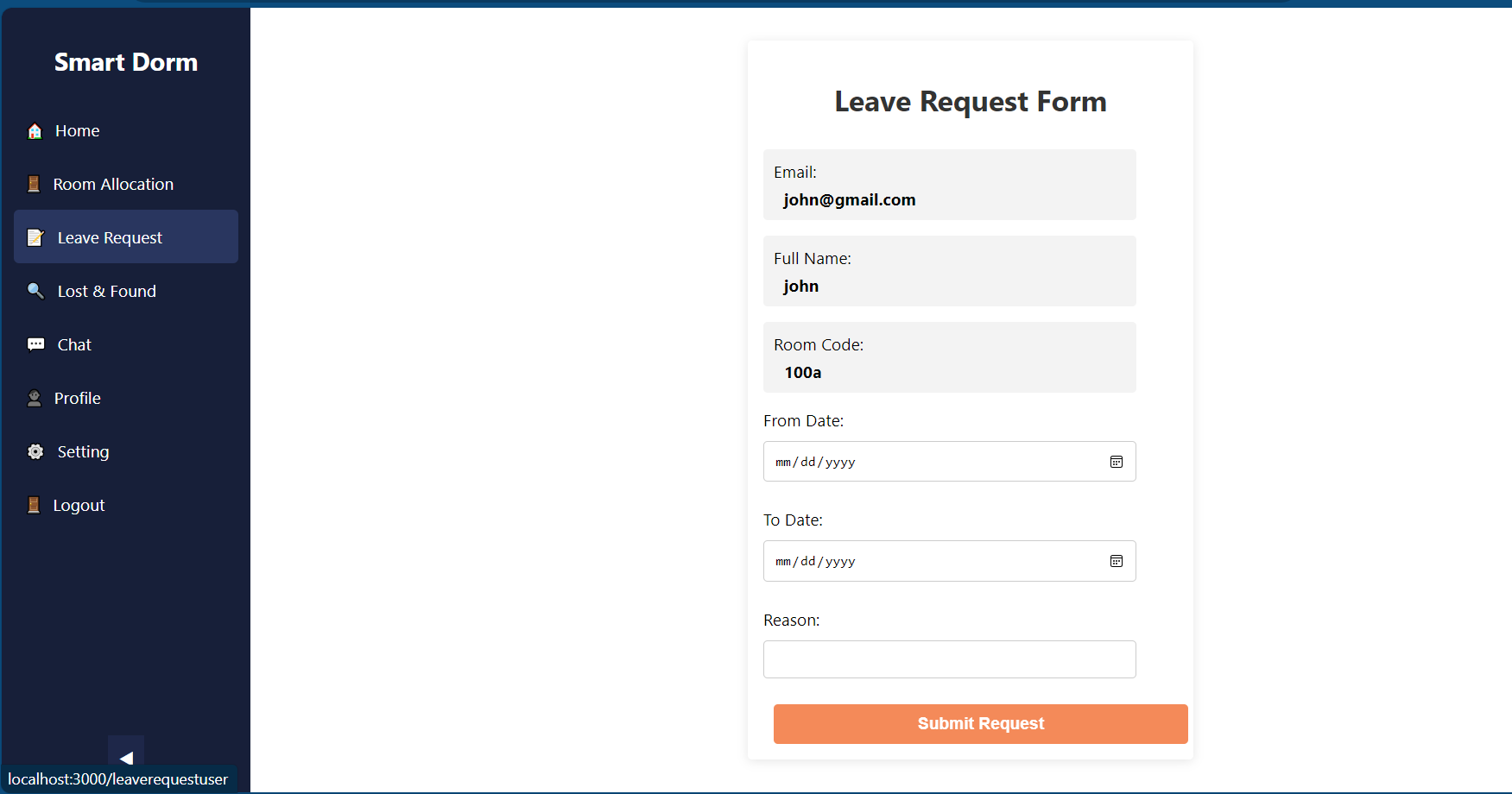


Figure: 12 Leave Request Screen

10.1.5 Chat Screen

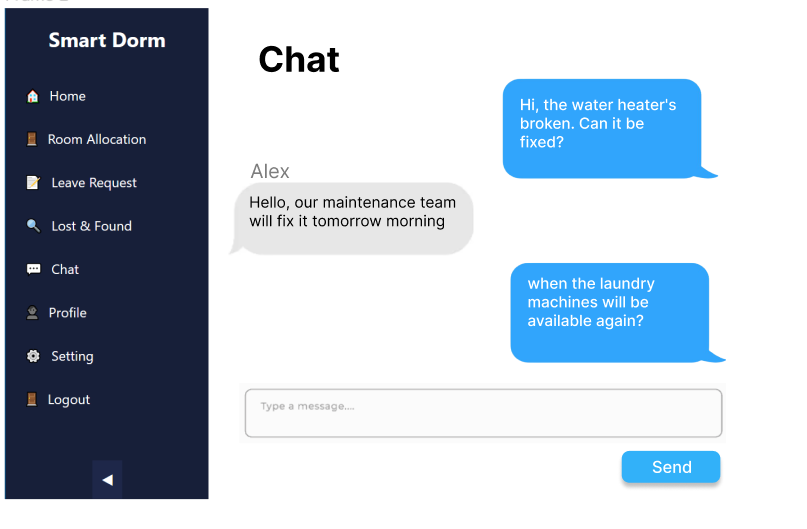


Figure: 12 Chat Screen

10.1.6 Lost and Found Screen

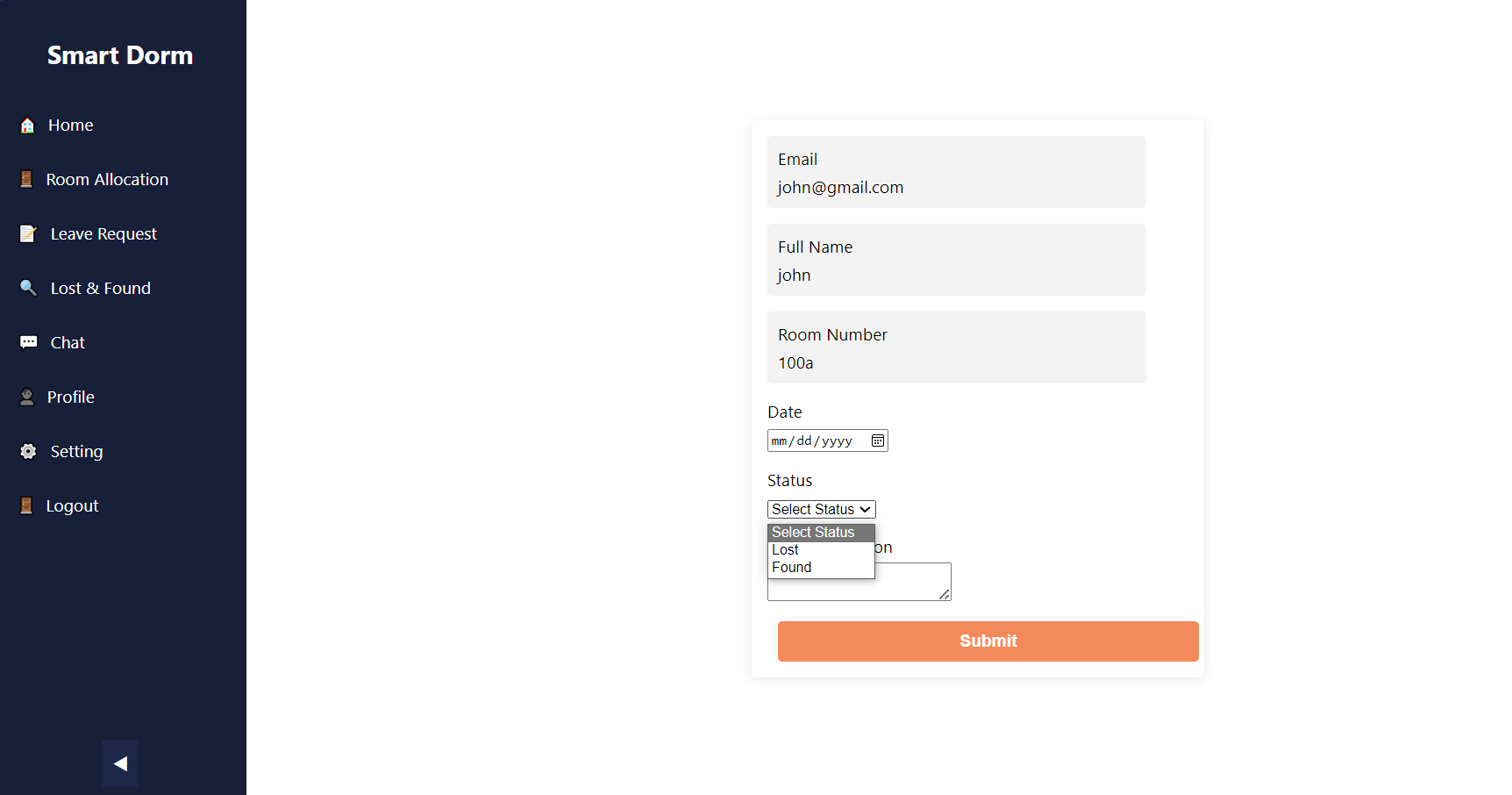


Figure: 13 Lost and Found Screen

10.2. Mobile Application

10.2.1 Splash Screen:



Figure: 14 Splash Screen

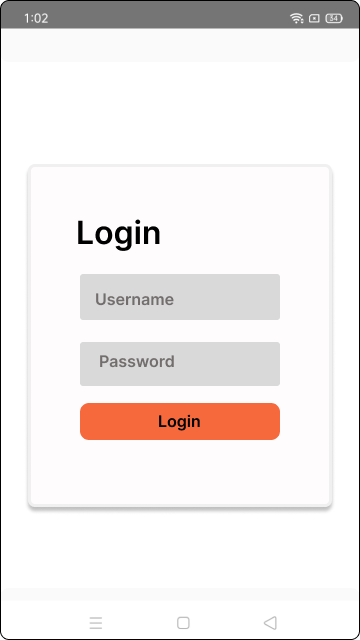
10.2.2 Login Screen:

Figure: 15 Login Screen

10.2.3 User Dashboard Screen:

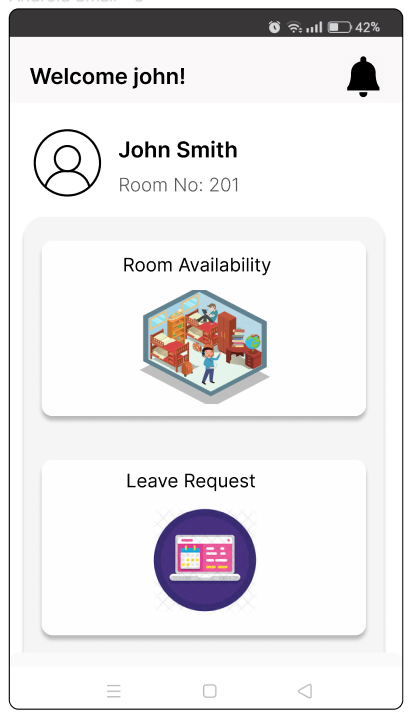


Figure: 16 User Dashboard Screen

10.2.4 Leave Request form Screen:

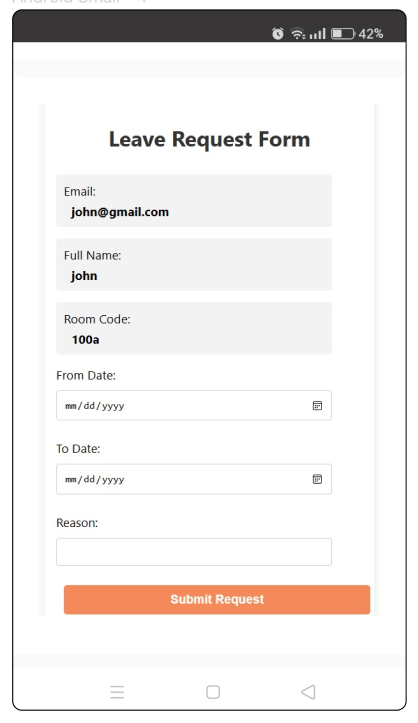


Figure: 17 Leave Request form Screen

10.2.5 Room Availability Screen:

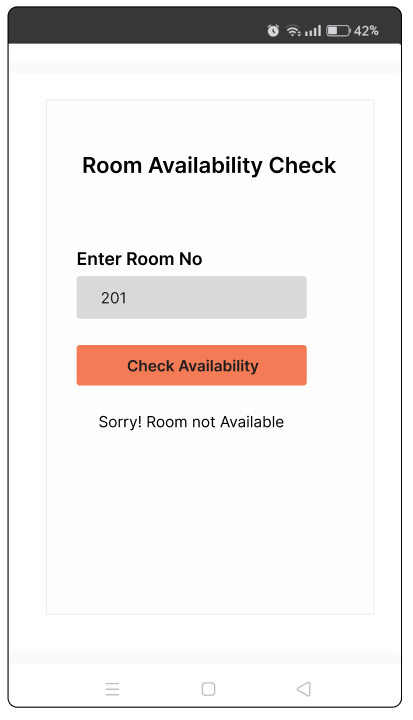


Figure: 18 Room Availability Screen

10.2.6 Notification Screen:

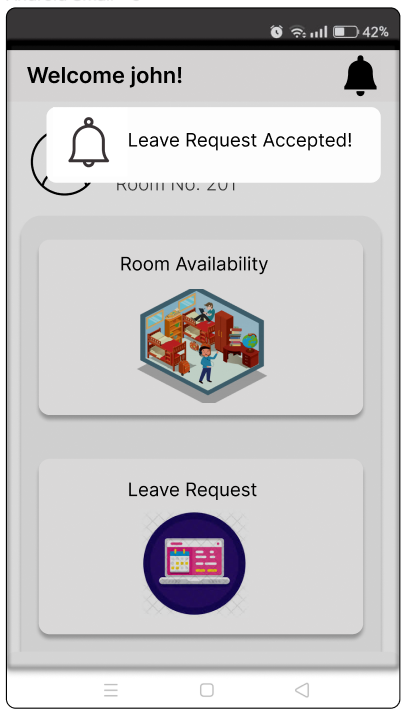


Figure: 19 Notification Screen

11. Conclusion and Future Plan:

* 1. **Conclusion:**

The Smart Dorm project stands out for its commitment to enhancing the hostel living experience through innovative features and a user-centric approach. By leveraging modern technology, Smart Dorm addresses limitations found in existing systems, offering a more scalable, secure, and intuitive solution. As hostel management practices evolve, Smart Dorm is well-positioned to lead with its advanced functionalities and potential for future integration and expansion.

**11.2** **Future Directions:**

To remain competitive and relevant, the Smart Dorm project will consider exploring AI and machine learning for predictive analytics in room allocation and maintenance. Integrating with campus-wide systems for a holistic approach to student services and incorporating feedback mechanisms for continuous improvement will further solidify Smart Dorm's position as a forward-thinking solution in hostel management technology.

By addressing current shortcomings and focusing on technological advancement, Smart Dorm is poised to redefine hostel management, offering a system that is not only functionally superior but also more aligned with the expectations of modern users.

#### 12. References

* Kartik Chaudhri, Riddhi Kevat. "Study of Digitalized Hostel Management System", vol. 7, no. 2, pp. 366-371, March-April 2021. Ayanlowo, K., Shoewu, O., & Olatinwo, S. O. (2014).
* "Design and Implementation of a Hostel Management System" by Abiola Ayodeji and Ahmed Rufai. (Link: https://www.researchgate.net/publication/328774914\_Design\_and\_Implementation\_of\_a\_Hostel\_Management\_System)
* "Real-Time Web Applications with WebSocket, PHP, and jQuery" by Mehul Mohan. (Link: https://ieeexplore.ieee.org/document/6679155)
* "Design and Implementation of Hostel Management System Based on Web" by Jun Wei, Xuefei Wang, and Lingjun Li. (Link: https://ieeexplore.ieee.org/document/7754667)
* "Real-Time Web Application Development using AngularJS and WebSockets" by Miguel Angel Durán García. (Link: https://www.researchgate.net/publication/304949965\_Real-Time\_Web\_Application\_Development\_using\_AngularJS\_and\_WebSockets)