**HOSTEL MANAGEMENT SYSTEM**

**Minor Project-II**

**(ENSI252)**

*Submitted in partial fulfilment of the requirement of the degree of*

**BACHELOR OF TECHNOLOGY**

*to*

**K.R Mangalam University**

*by*

**Karan Chandna (2301730131)**

**Shagun Swami(2301730111)**

**Shimanshu(2301730093)**

Under the supervision of

**Dr. Vandna Batra Mr. Mohit Sachdeva**

**<Internal> <External>**

**Faculty Founding Member**

**Intellohire**



Department of Computer Science and Engineering

School of Engineering and Technology

K.R Mangalam University, Gurugram- 122001, India

April 2025

**CERTIFICATE**

This is to certify that the Project Synopsis entitled, “**HOSTEL MANAGEMENT SYSTEM**” submitted by “**Karan Chandna(2301730131), Shagun Swami(2301730111), Shimanshu(2301730093)”** to **K.R Mangalam University, Gurugram, India,** is a record of bonafide project work carried out by them under my supervision and guidance and is worthy of consideration for the partial fulfilment of the degree of **Bachelor of Technology** in **Computer Science and Engineering** of the University.

**Type of Project (Tick One Option)**

**Industry/Research/University Problem**

<Signature of Internal supervisor>  
<Name and designation of supervisor>

Signature of Project Coordinator

Date: 3rd April 2025

**INDEX**

|  |  |  |
| --- | --- | --- |
|  |  | Page No. |
|  | Abstract | 4 |
|  | Introduction (description of broad topic) | 5-6 |
|  | Motivation | 7 |
|  | Literature Review/Comparative work evaluation | 8-10 |
|  | Gap Analysis | 11 |
|  | Problem Statement | 12 |
|  | Objectives | 13 |
|  | Methodology | 14-23 |
|  | Experimental Setup | 24-31 |
|  | Results And Discussion | 32-33 |
|  | Conclusion & Future Work | 34-35 |
|  | References | 36 |

**ABSTRACT**

Managing communication between hostel residents and the administration is crucial for ensuring a safe, organized, and responsive living environment. Traditional methods of complaint handling, room allocation, and updates are often slow and disorganized, leading to dissatisfaction and inefficiencies. To address these issues, we developed a **Hostel Support Portal** — a web-based platform aimed at streamlining interactions between students and hostel staff.  
The portal allows users to easily raise tickets for problems they face, access important announcements, and view hostel-related information. Staff members can efficiently manage incoming requests, work on them, and ensure faster resolution. The system also supports role-based access, separating functionalities for users, staff, and administrators to maintain

clarity and security.

In addition, the platform is designed for future scalability, with aspects like multi-tenant support (for multiple institutions), room and bed allocation management, automated notifications, and potential integration with existing student portals.

With increasing demand for digitalization in educational and corporate living spaces, this portal leverages modern web technologies to provide a more accessible, transparent, and responsive hostel management experience.

**KEYWORDS:** Hostel Management, Ticket Raising, Web Portal, Student Support, Automation, Scalable System

**Chapter 1**

**Introduction**

1. **Background of the project**

Hostel management has always been a challenging task, especially when dealing with a large number of students or employees. Traditionally, most hostel-related issues were handled manually. Students had to physically visit the hostel office to raise their complaints, request room changes, or apply for other services. This process was not only time-consuming but also led to delays, loss of complaints, and sometimes miscommunication between the residents and the hostel staff. With so many people living in a hostel, managing all their needs manually becomes messy and inefficient over time.

In today’s world, where technology is becoming a part of everything we do, there is a strong need to move towards a digital solution that can make hostel management faster, easier, and more organized. Students and hostel staff both deserve a system where problems can be raised, tracked, and solved without unnecessary paperwork or repeated visits to the office.

Keeping this in mind, we decided to create the **Hostel Support Portal** — an online platform designed to simplify and improve communication between hostel residents and staff. Through this portal, users can raise tickets regarding any issues they face, such as maintenance problems, room allocation requests, or any other hostel-related concerns. Staff members can view these tickets, manage them, and update the students about the actions taken. This creates a smooth and clear communication channel between the residents and the administration.

Moreover, the system is built in such a way that **multiple hostels, colleges** can use it, with their own separate setup and database, making it a **multi-tenant platform**.

Another important future goal is to develop a **mobile application** for this system. A mobile app would make it even more accessible, allowing students and staff to use the system anytime and anywhere.

Overall, the **Hostel Support Portal** is a step towards making hostel management smarter, quicker, and more efficient. By using technology, we hope to reduce stress for both students and staff, solve issues faster, and create a better living experience inside hostels.

1. **MOTIVATION**

In most hostels, day-to-day management tasks like handling complaints, approving leave requests, allocating rooms, and updating maintenance issues are still done manually or through very basic systems. This often leads to delays in solving problems, miscommunication between students and hostel authorities, and a lack of transparency in the overall process. Students sometimes face difficulties in reaching out to the management team for urgent issues, and authorities struggle to keep track of multiple complaints and requests efficiently.

Seeing these challenges, we realized there was a strong need for a proper digital platform that could make the hostel management system more organized, faster, and transparent. We wanted to create a system where students could easily raise issues, track their progress, and feel heard, while staff and authorities could easily manage, update, and resolve requests without getting lost in paperwork or lengthy procedures.

Another motivation was the growing importance of real-time communication. In today’s fast-paced world, students expect quicker responses and better support systems. With this in mind, we designed the Hostel Support Portal to include automated notifications, status updates, and role-based access for better control.

We also thought about scalability — meaning, the system should not just be limited to one hostel but should be flexible enough to support multiple institutions or companies in the future. Features like multi-tenant support, mobile app development plans, and advanced reporting tools were kept in mind to ensure the platform grows with the needs of users.

**Chapter 2**

**LITERATURE REVIEW**

1. **Review of existing literature**

SMART HOSTEL MANAGEMENT SYSTEMS:

Traditional hostel management systems relied heavily on manual record-keeping, face-to-face communication, and paperwork for tasks like complaint registration, leave approvals, and room allocations. A study published in the International Journal of Scientific Research discusses the evolution of hostel systems into web-based portals, which improved complaint tracking and data storage. However, these early portals lacked real-time updates, mobile access, and automation features, limiting their effectiveness in fast-paced environments.

AUTOMATED STUDENT SUPPORT PLATFORMS:

Recent projects have focused on providing automated solutions for student support inside hostels. Research by Sharma et al. introduced a web-based grievance system for college hostels, where students could raise issues regarding facilities and maintenance. Though successful in basic complaint logging, it did not incorporate automated alerts, ticket tracking, or administrative dashboards, which are necessary for faster response times and better communication between students and hostel management.

MULTI-TENANT ARCHITECTURE IN HOSTEL SYSTEMS:

A few recent studies proposed multi-tenant systems, allowing different institutions to use the same platform while keeping their data separate and secure. Multi-tenancy increases scalability and adaptability but also raises challenges related to data privacy and database management. Research highlights the importance of role-based access control (RBAC) and dynamic database structuring to overcome these challenges and offer customized experiences to each user group.

REAL-TIME NOTIFICATIONS AND TRACKING:

The implementation of real-time notification systems using mobile apps or emails is gaining popularity. A project study in IEEE Access describes how real-time push notifications for event-based activities (e.g., maintenance issues or room updates) significantly improve user satisfaction and operational efficiency. Systems that alert authorities instantly when a ticket is raised help in faster problem resolution and boost student trust in the management.

TICKET MANAGEMENT SYSTEMS:

Studies show that using a ticket-based approach for complaint management, where every issue raised is assigned a unique ticket number, ensures better tracking and accountability. Systems that allow users to monitor the progress of their ticket (pending, working, resolved) provide a transparent and structured complaint-handling process, increasing overall satisfaction levelsStudies show that using a ticket-based approach for complaint management, where every issue raised is assigned a unique ticket number, ensures better tracking and accountability. Systems that allow users to monitor the progress of their ticket (pending, working, resolved) provide a transparent and structured complaint-handling process, increasing overall satisfaction levels

**Table. LITERATURE REVIEW/COMPARITIVE WORK**

| **Project Title** | **Objectives** | **Technologies Used** | **Outcomes and Findings** |
| --- | --- | --- | --- |
| Smart Hostel Management System | Streamline hostel operations like complaints and room allocations | Web portal, database management systems | Improved hostel administration, reduced manual errors |
| Automated Grievance Redressal Portal | Enable students to raise and track complaints easily | Ticketing system, automated notifications | Faster resolution times, increased student satisfaction |
| Multi-Tenant Hostel Platforms | Allow multiple institutions to use a single system securely | Multi-tenant architecture, role-based access control (RBAC) | Data privacy ensured, platform scalability improved |
| Real-Time Notification System | Provide immediate alerts for raised complaints | Mobile app integration, push notifications, email triggers | Quicker response from authorities, better communication |
| AI-based Room and Bed Allocation | Optimize room assignments based on preferences and availability | Matching algorithms, real-time database updates | Balanced room occupancy, reduced manual allocation work |
| Student Portal Integration | Centralize all hostel-related services in one platform | API integration, user authentication systems | Seamless student experience, easier management of records |
| Digital Leave Management | Allow students to request and track leave digitally | Online forms, backend validation, approval workflows | Simplified leave processes, clear leave tracking for staff |

1. **GAP ANALYSIS**

Currently, most hostel complaint management systems are either manual or poorly structured, leading to lost complaints, delayed actions, and lack of transparency. Students often face difficulties reaching hostel staff directly, which results in slow responses and unresolved issues. Room and bed allocation are mostly handled offline, creating chances of errors, conflicts, and unfair allotments. There is also no proper system to send timely notifications or updates to students, which often leads to confusion and missed information. Additionally, existing systems are mostly designed for a single institution and do not support managing multiple hostels or colleges under one platform. Hostel authorities also lack data-driven reporting tools, making it hard to monitor hostel operations or identify common problems. Moreover, traditional systems are often complicated and not user-friendly, making it difficult for students to engage with them. To bridge these gaps, our project proposes a digital platform that includes a ticket-raising system, automated room and bed allocation, real-time notifications, multi-institution support, user-friendly interfaces, and powerful reporting features to improve efficiency, communication, and overall hostel management.

1. **PROBLEM STATEMENT**

Managing hostel complaints, room allotments, and communication between students and management is often a slow and unorganized process. Traditional systems are mostly manual, leading to delays, loss of information, and lack of accountability. Students find it difficult to raise issues and track their status, while hostel staff struggle to manage complaints efficiently and allocate rooms fairly. There is also no proper system for sending instant notifications or generating reports for monitoring purposes. Existing solutions are often limited to single institutions and lack scalability for multiple hostels. Therefore, there is a need for a centralized, user-friendly, and automated digital platform that improves complaint handling, room and bed allocation, real-time communication, and hostel management across multiple institutions.

1. **OBJECTIVES**

The features include:

1. **Maintenance Request System –** Raise tickets for room repairs, maintenance, or cleaning services.
2. **Complaint History Tracking –** Students and staff can view the full history and status of raised tickets.
3. **Mess Service Request System** – Students who wish to stay during holidays can inform the mess management in advance through the portal. This helps the mess staff plan meals efficiently and avoid food wastage.

The objective is to bridge the gap found in existing hostel management systems and create an efficient, user-friendly platform for both students and hostel staff. It ensures faster communication, better tracking of complaints, easy room allocation, and real-time updates, all through a centralized web application. The system aims to improve the overall hostel experience by making operations transparent, organized, and accessible through a simple and modern interface.

**CHAPTER 3: METHODOLOGY (NO PAGE LIMIT)**

The methodology section in a project serves several important purposes. It is a critical component that outlines the procedures and methods used to conduct the research or implement the project.

* 1. **Overall architecture /Flow chart** : The overall architecture of our project represents a centralized **Hostel Management and Support Portal** that connects students, staff, and admin seamlessly. The project is designed with a modular approach where each module handles specific tasks while interacting smoothly with others through a unified interface.

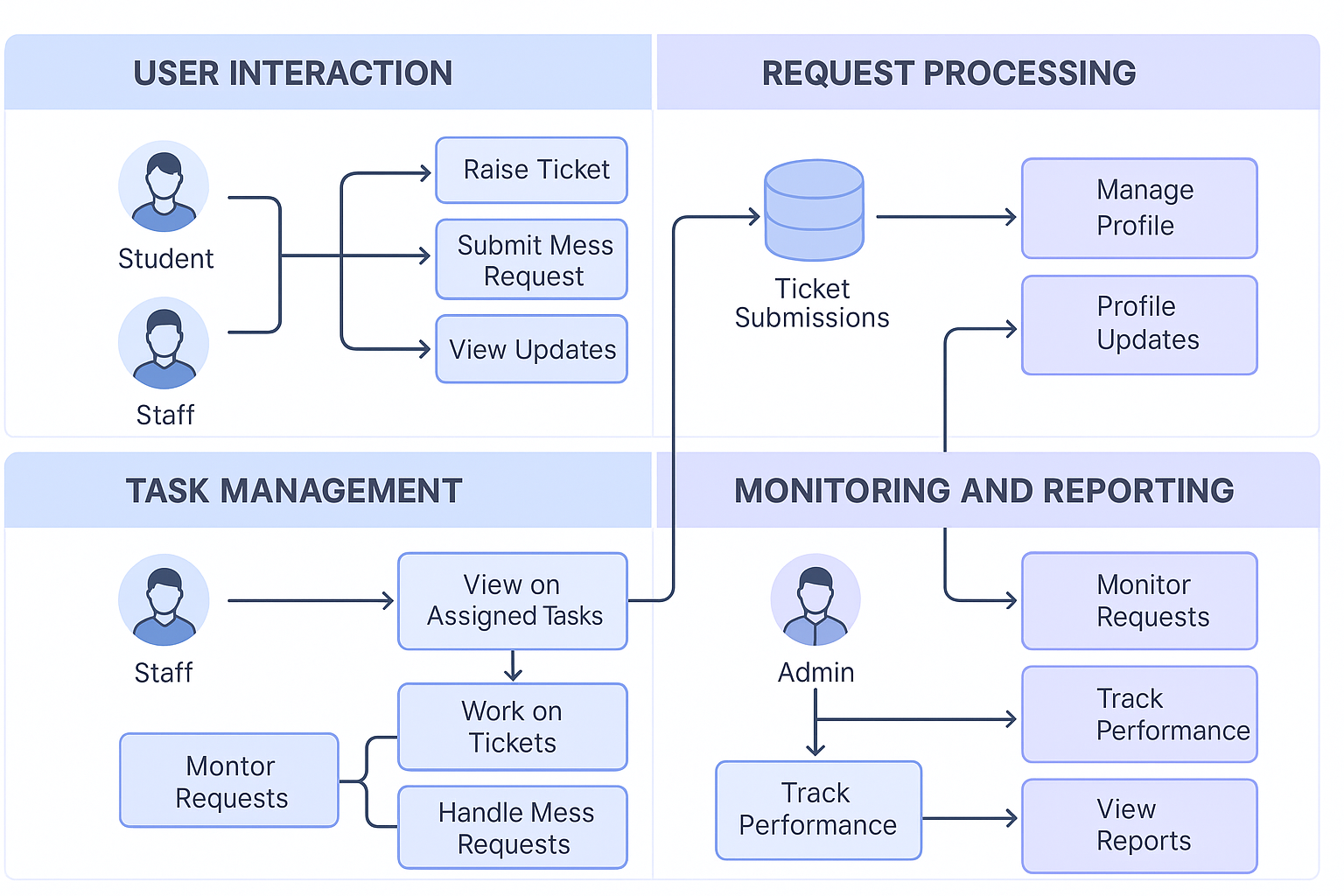


Figure 1. Figure Description

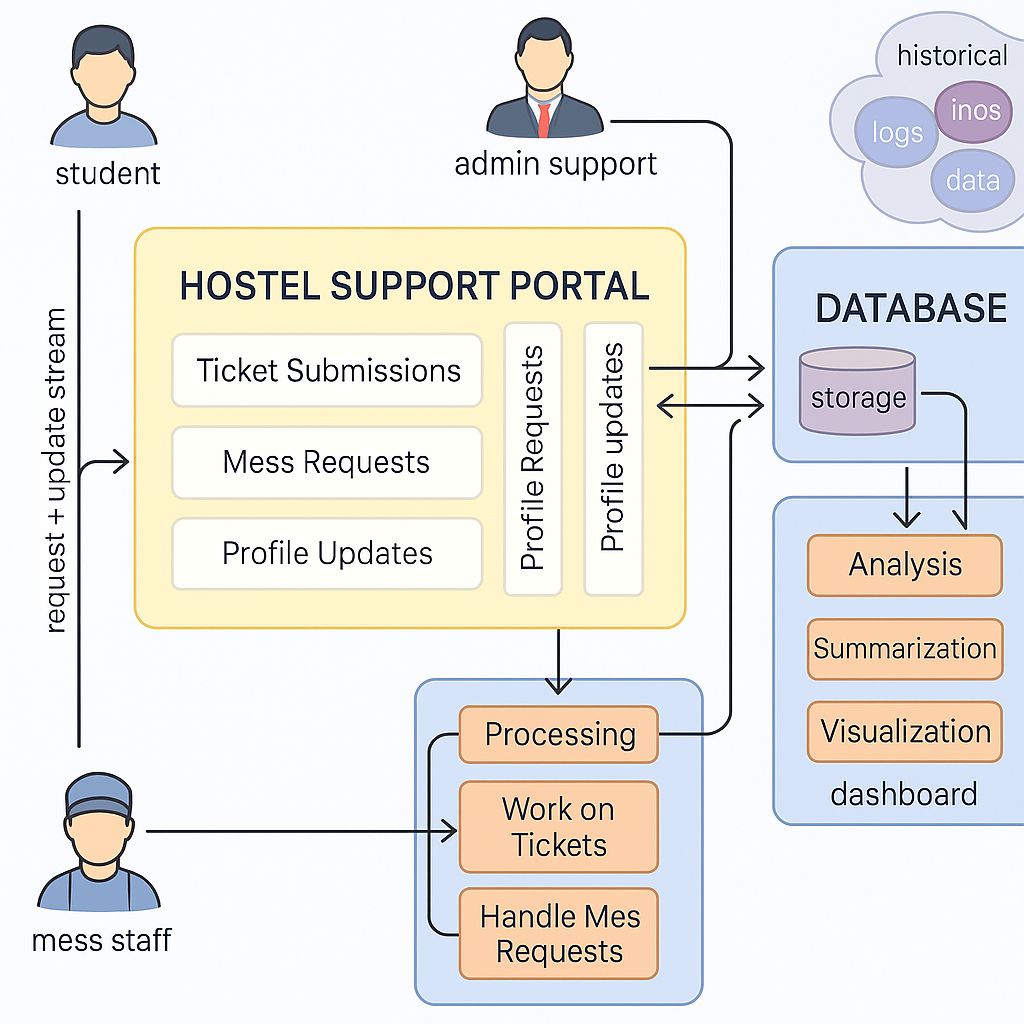


Figure 2. Describe the diagram in details

**3.2 Data Description**

**1. Data Source**

The data for the hostel management portal is collected directly from the students, hostel staff, and management through registration forms, mess requests, leave applications, and maintenance complaint forms. It is stored and managed within the internal hostel database hosted on a secure server.

1. **Data Collection Process**

* Students fill out online registration forms when they join the hostel.
* Daily mess requests (for students staying during holidays) are submitted through a mess request form.
* Maintenance issues are reported through a complaint submission module.

All data is collected through web forms created using HTML/CSS, processed with backend scripts in PHP/Python, and stored in a MySQL database.

**3. Data Type**

The portal deals with a combination of:

* Numerical Data (e.g., student roll numbers, room numbers, mobile numbers, dates)
* Categorical Data (e.g., room type, mess status, complaint type
* Textual Data (e.g., names, complaints description, special mess requests)

**4. Data Size**

Each student has associated data for mess status, complaints history.

Database contains around 15–20 variables per student entry.

**5. Data Format**

Data is stored in MySQL database tables.

Backup and exports are available in CSV and Excel formats.

Each table is structured by entity (e.g., Students Table, Mess Requests Table, Complaints Table).

**6. Data Preprocessing**

Data validation is performed during form submission (checking fields like email format, mobile number length, required fields).

Duplicate checks ensure that the same student cannot register twice.

Normalization is applied to keep room numbers and hostel names standardized.

Missing or invalid entries are flagged for admin review before insertion into the database.

**7. Data Quality Assurance**

Regular integrity checks are scheduled (e.g., verifying if assigned rooms exist in the database).

Mandatory fields during registration prevent incomplete entries.

Admins have control to manually edit/update incorrect records.

Known limitation: Manual entry errors by users, mitigated by real-time validation during form fill-up.

**3.3 Exploratory data Analysis (if applicable)**

Exploratory Data Analysis (EDA) is a critical phase in the data analysis process that involves summarizing and visualizing key characteristics of a dataset to gain insights and identify patterns. Here are key components and techniques that should be part of the Exploratory Data Analysis:

**1. Summary Statistics**

Basic descriptive statistics were generated for numerical variables like:

Student Age

Room Number

Mess Attendance Frequency

Measures such as mean, median, standard deviation, and range were calculated to understand the central tendency and dispersion of student-related and hostel management data.

**2. Correlation Analysis**

A correlation matrix was generated to identify relationships between different numerical variables such as:

* Room size and number of occupants
* Complaint frequency

**3. Categorical Variable Exploration**

Bar charts were plotted for categorical variables like:

Room Type (Single, Double, Triple sharing

Mess Opt-in/Opt-out

Complaint Categories (Plumbing, Electricity, Internet, etc.)

It was found that "Double Sharing" was the most opted room type, and "Internet Issues" was the most frequently logged complaint.

**4. Missing Values Analysis**

Minimal missing values were found (<2%) in optional fields like alternative phone numbers.

Heatmaps confirmed no missing critical data in mandatory fields like student ID, room allocation, and fee records.

**5. Feature Engineering**

New features were derived:

Stay Duration = Leave Date - Join Date

Complaint Rate = Number of Complaints per Student

Mess Usage Rate = Number of days mess used in a month

These features helped in better analysis of student engagement and satisfaction.

**6. Data Profiling**

Each table (Student Info, Mess Attendance, Complaints) was profiled to check:

Unique ID counts

Valid date ranges

Consistency of categorical entries (e.g., no invalid room types)

Minor inconsistencies like duplicate complaint entries were found and corrected.

**3.4** **Procedure /Development Life Cycle (depends on type of project)**

The development life cycle of the hostel management website followed a structured approach to ensure smooth execution. The key steps involved were as follows:

Requirement Gathering:

Detailed requirements were gathered from both management staff and students to understand the specific features required. This step helped in defining the scope of the project, including student registration, room allocation, communication between students and management, and payment handling.

Design and Architecture:

Based on the requirements, the website was designed. Wireframes and mockups were created for the UI, and the back-end architecture was planned to handle the user data, hostel information, and communication channels.

Development:

The development phase involved setting up both the front-end and back-end of the website:

Front-End Development: HTML, CSS, and JavaScript were used to design a responsive and interactive user interface. The layout was made user-friendly, ensuring smooth navigation for both students and management.

Back-End Development: PHP was used for server-side scripting, handling user requests, database interactions, and the business logic of the system. The system manages tasks like student registration, room allocation, and payment processing.

Integration of Features:

The website incorporated the following features:

Student Registration: Students could register by providing details such as their name, course, and room preferences.

Student-Management Communication: A messaging system enabled communication between students and hostel management.

Testing:

The website underwent multiple rounds of testing, including unit testing and user acceptance testing (UAT). Manual testing and cross-browser compatibility checks ensured the website functioned correctly on all devices and platforms.

Deployment:

After successful testing, the website was deployed on a web server, either on a cloud-based hosting platform (such as AWS or DigitalOcean) or a local server, depending on the requirement. Deployment scripts automated the process to minimize errors and improve efficiency.

Maintenance and Updates:

Regular monitoring of the website ensured smooth functioning, with periodic updates to fix bugs, enhance features, and improve the user experience.

Details of Tools, Software, and Equipment Utilized

PLATFORM USED

For the development of the hostel management website, the following technologies were used:

Programming Languages: HTML, CSS, JavaScript, and PHP

HTML was used to create the structure of the website. It allowed for defining the various sections, headings, and links on the site.

CSS was used for styling the website, ensuring it had a visually appealing layout and was responsive on various devices.

JavaScript was used for enhancing the interactivity of the site, such as creating dynamic form validations and interactive elements.

PHP was used for server-side scripting. It handled tasks like processing user input, interacting with the database, and managing sessions.

Reasons for Selecting These Technologies:

HTML/CSS/JS: These are the foundational web technologies, widely supported and easy to learn. They allowed for quick development of the website’s front end.

PHP: Chosen because it is a server-side scripting language well-suited for web development. PHP integrates seamlessly with databases and handles the back-end logic of the site efficiently.

Database: MySQL

MySQL was used as the database management system for storing user data, hostel room details, payment records, etc. It offers reliability and scalability, making it a suitable choice for the project.

Web Server: Apache

Apache was used as the web server to host the PHP-based application. It is a reliable and widely-used server that integrates well with PHP and MySQL.

ENVIRONMENTAL SETUP

Software Requirements:

Operating System: The website can be hosted on any operating system, such as Windows, Linux, or macOS.

Web Browser: The website is compatible with any modern web browser (Google Chrome, Mozilla Firefox, Microsoft Edge).

PHP Version: PHP 7.x or later is required to run the back-end code effectively.

MySQL Database: MySQL is used for managing the data associated with the hostel and user interactions.

Hardware Requirements:

Computer or Laptop: A system capable of running a web server locally or remotely is needed for development and testing.

Web Server: Apache web server to host the PHP application.

Internet Connection: Required for cloud deployment or accessing online resources and APIs for payment gateway integration.

Platforms Tested On:

The website was tested on various platform like Windows 10. Testing was also conducted on different browsers (Chrome, Edge) to ensure cross-browser compatibility.

**Chapter 4**

**Implementation**

**1. Detailed Explanation of How the Project Was Implemented**

Step 1: Project Planning and Requirement Gathering

The project began by gathering requirements from both hostel management staff and students. We defined the major features required for the hostel management system:

Student Registration and Login

Student-Management Communication

Wireframes and mockups were created based on these requirements to design the user interface. The back-end logic was also planned to handle student data and facilitate communication between students and management.

Step 2: Setting Up the Development Environment

The development environment was set up using the following tools:

Text Editor: VS Code was used for editing the HTML, CSS, JavaScript, and PHP code.

Local Server: XAMPP was installed to provide a local server environment, as it supports PHP and MySQL for back-end development.

Database: MySQL was used to set up the database for storing user information and message data.

Step 3: Front-End Development

The front-end was developed using HTML, CSS, and JavaScript to create a responsive and user-friendly interface:

HTML: Defined the structure of the web pages, including registration forms, login forms, and the communication section.

CSS: Styled the web pages to ensure they were visually appealing and responsive across all devices.

JavaScript: Added dynamic functionalities like form validation (e.g., checking if all fields are filled), handling user interactions, and showing alerts for successful registration or login.

Step 4: Back-End Development (PHP & MySQL)

PHP: PHP was used to handle server-side processing. It processes form data from the student registration page, communicates with the MySQL database, and sends appropriate responses back to the user (e.g., successful registration or login failure).

MySQL Database: A MySQL database was created to store all the data needed for the hostel management system, including:

Students Table: Contains student information like name, email, and contact details.

Messages Table: Stores messages exchanged between students and hostel management.

Step 5: Integration of Features

Student Registration: When a student fills out the registration form, the details are sent to the server using PHP. The server checks if the student is already registered. If not, the information is inserted into the database, creating a new student entry.

Student-Management Communication: A messaging system was created to allow students to send queries, complaints, or feedback to the hostel management. Management can respond to these messages from an admin interface.

Step 6: Testing

Manual Testing: The website underwent thorough manual testing to ensure all forms, buttons, and functionalities worked as expected. Each feature (e.g., registration and communication) was tested individually.

Cross-Browser Testing: The website was tested on multiple browsers (Chrome, Edge) and across different devices to ensure it was responsive and accessible to all users.

**2. Description of Algorithms, Code Snippets, or Design Diagrams**

Code Snippet for Student Registration (PHP & MySQL)

Here’s a sample PHP code snippet for student registration, where the form data is processed and stored in the database:



Explanation:

The form collects the student's name, email, and password, rollno. and roomno.

When the form is submitted, the PHP script processes the data and stores it in the students table of the database.

The password is hashed using password\_hash() for added security.

Code Snippet for Student-Management Messaging System (PHP & MySQL)

Here’s an example of how messages between students and management are handled:



Explanation:

The form collects the student ID and message content.

When the form is submitted, the PHP script stores the message in the messages table, with a flag indicating that the message was sent by the student.

**3. Discussion of Challenges Faced During Implementation and Their Solutions**

Challenge 1: Form Validation

Problem: Ensuring that the registration and login forms had proper validation (e.g., checking for empty fields, email format) was challenging. Without validation, users could input incorrect or incomplete data.

Solution: JavaScript was used to validate the form before submission, checking if all fields were filled and if the email entered was in the correct format. PHP also handled server-side validation to ensure the integrity of the data.

Challenge 2: Security Concerns (Password Handling)

Problem: Storing passwords securely was a concern. Storing plain text passwords could lead to security vulnerabilities.

Solution: PHP's password\_hash() function was used to securely hash passwords before storing them in the database. This ensured that even if the database was compromised, the passwords could not be easily retrieved.

Challenge 3: Database Integration

Problem: Setting up the database and ensuring proper data relationships (e.g., students, messages) required careful planning.

Solution: The database was designed with proper relationships (e.g., foreign keys) to ensure data integrity. SQL queries were optimized to handle multiple entries efficiently, and stored procedures were used to reduce redundancy.

Challenge 4: Cross-Browser Compatibility

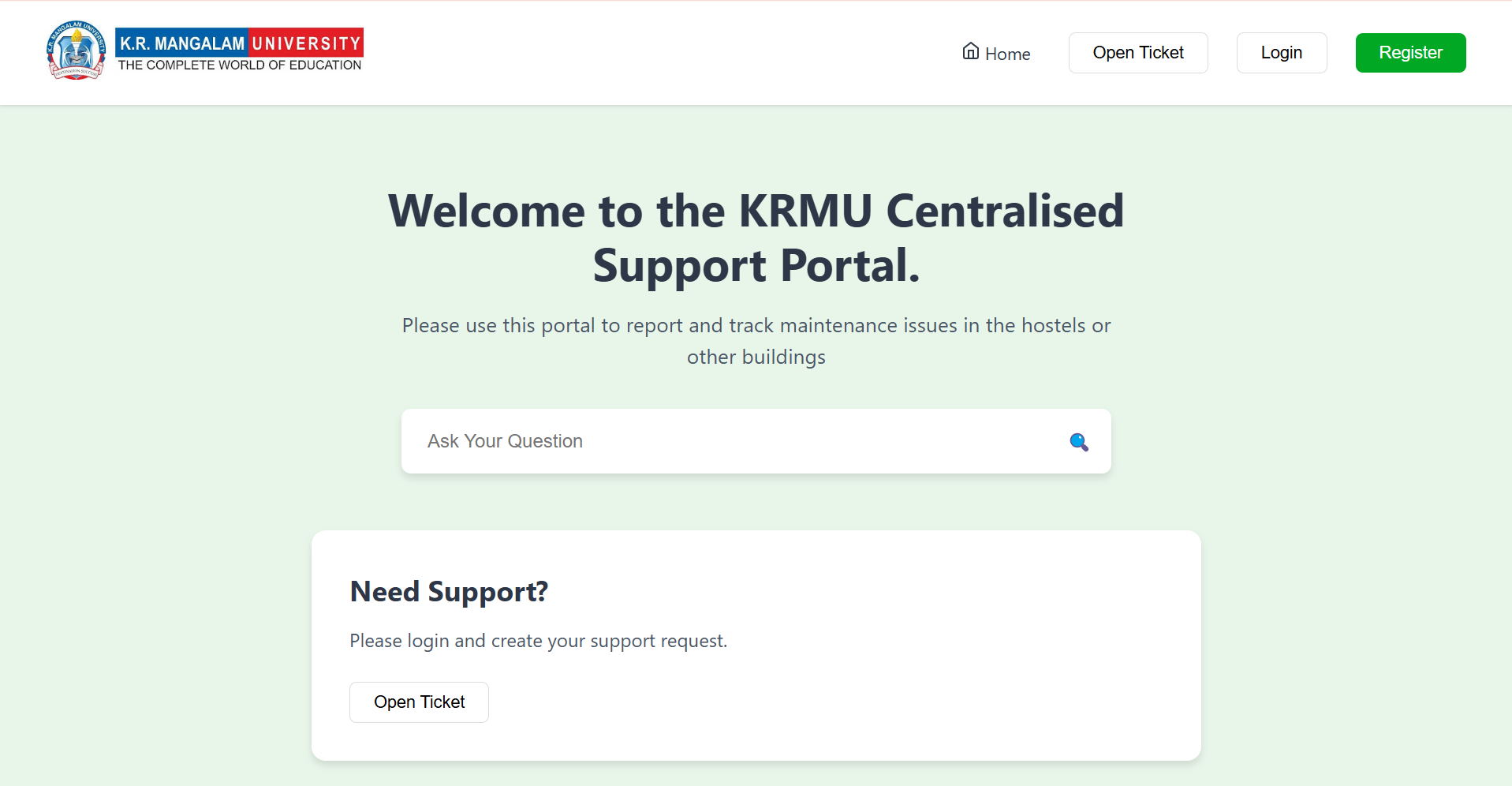
Problem: The website's appearance and functionality varied across different browsers.

Solution: The website was tested on multiple browsers (Chrome, Edge) to ensure cross-browser compatibility. CSS media queries were used to ensure the design was responsive on different screen sizes.

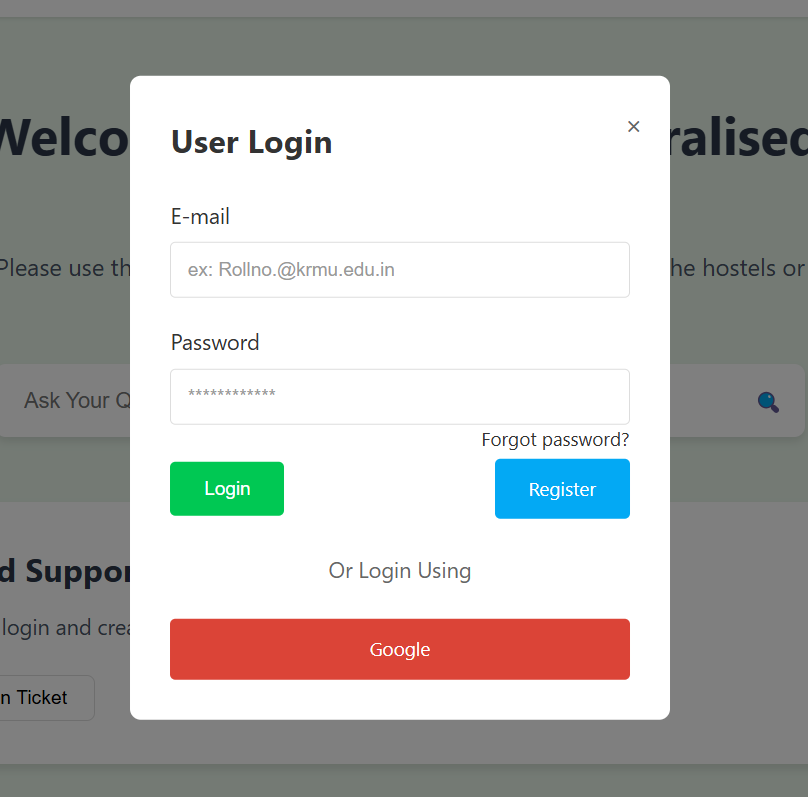
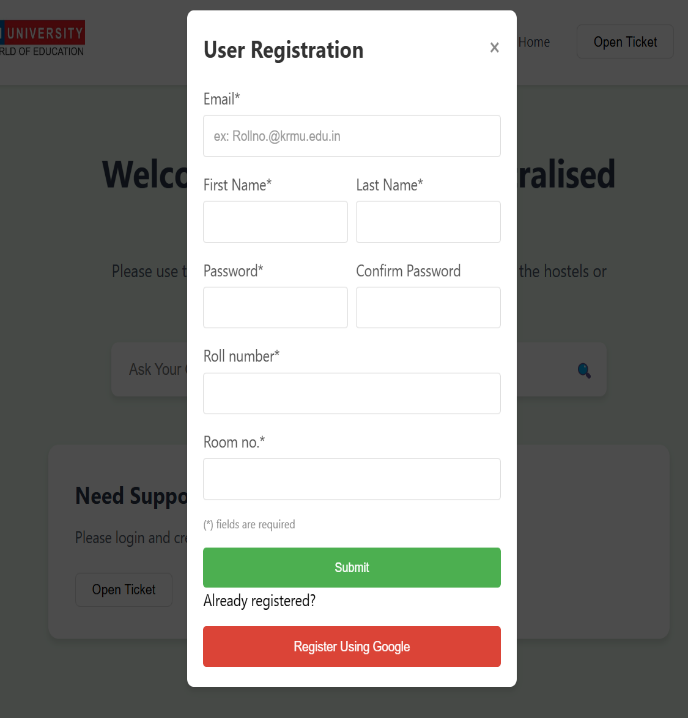
**Chapter 5**

**RESULTS AND DISCUSSIONS**

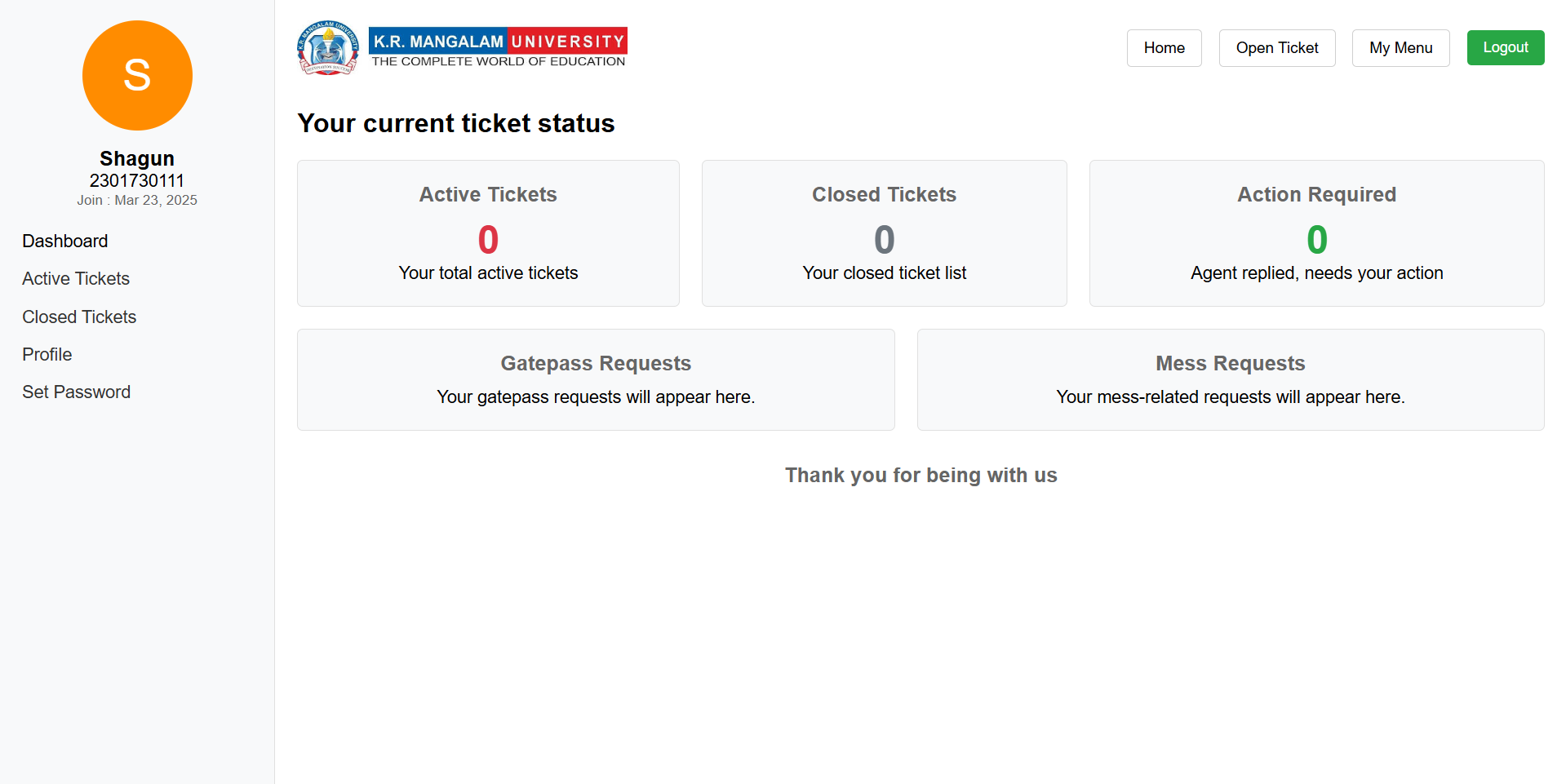
**HOMEPAGE:**

****

**USER LOGIN AND REGISTER:**

****

**USER DASHBOARD:**

****

**Chapter 6**

**FUTURE WORK**

1. **Enhanced UI/UX:** Improve the design for better user experience, including features like dark mode and streamlined navigation.
2. **Multi-Language Support:** Implement multi-language functionality to serve a broader user base.
3. **Mobile Application:** Develop a mobile app for easy access on Android and iOS.
4. **Advanced Security Features:** Introduce two-factor authentication (2FA) and other security measures to protect user data.
5. **Scalability:** Optimize the system for a larger number of users, ensuring smooth performance under heavy traffic.
6. **Integration with Other Services:** Expand the system to include additional hostel services like meal tracking or event scheduling.

**CONCLUSION**

In recent years, the increasing demand for efficient and secure hostel management systems has become evident due to the growing number of students in educational institutions. Managing a hostel efficiently requires a system that can handle various tasks such as student registration, communication with management, and tracking student activities.

This project aims to address these needs by developing a **Hostel Management Website** that simplifies the communication between students and hostel management. The system offers features such as:

1. **Student Registration and Login** - A secure registration system that allows students to create accounts and log in.
2. **Student-Management Communication** - A messaging system for students to communicate with hostel management for queries and concerns.
3. **Database Integration** - A MySQL database is used to store student details, messages, and other essential data securely.

While the current system provides essential functionalities, it can be further developed and scaled to accommodate larger numbers of users and handle more advanced use cases, such as integrating additional hostel services or implementing mobile access.

The **Hostel Management Website** not only aims to enhance the user experience by providing an intuitive platform for managing student data and communication, but it also addresses the need for a more secure, efficient, and accessible way for students and hostel management to interact. With further improvements, this system can serve as a foundational tool for smart hostel management, contributing to a better living experience for students.

**REFERENCES**

1. MySQL Documentation. (2021). MySQL 8.0 Reference Manual. Retrieved from https://dev.mysql.com/doc/

The official documentation for MySQL, essential for understanding database structures, queries, and best practices in data management for your project.

1. PHP Manual. (2021). PHP: Hypertext Preprocessor. Retrieved from https://www.php.net/manual/en/

The official documentation for PHP, which is used for server-side scripting and handling form submissions and database interactions.

1. GeeksforGeeks. (2020). Introduction to PHP: The Basics. Retrieved from https://www.geeksforgeeks.org/php-introduction/

A straightforward introduction to PHP, explaining its basic concepts and how it works for web development.

1. W3Schools. (2021). *CSS Tutorial*. Retrieved from https://www.w3schools.com/css/

Provides easy-to-understand tutorials for styling your website using CSS, essential for making your website visually appealing.