

**Faculty of Information Science & Technology**

**TSE6223 SOFTWARE ENGINEERING FUNDAMENTALS**

**Trimester Marh/April 2025**

**Project Report**

**Title: Hostel Management System**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Student ID | Student Name | Programme (AI/ BIA) | Lab Section |
| 1 | 1211110319 | KEE SHEE HAU | AI | 1C |
| 2 | 1211108357 | ONG QI REN | AI | 1C |
| 3 | 1211108626 | LIM CHIN CHEN | AI | 1C |
| 4 | 1211108213 | SOO TECK SHEN | AI | 1C |

<https://youtu.be/0wd_DkP0erw>



**FACULTY OF INFORMATION SCIENCE & TECHNOLOGY**

**Group Project Declaration**

TSE6223 SOFTWARE ENGINEERING FUNDAMENTALS

Trimester March/April 2025

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Student ID** | **Student name** | **Programme** | **Lab Section** |
| 1 | 1211110319 | KEE SHEE HAU | AI | 1C |
| 2 | 1211108357 | ONG QI REN | AI | 1C |
| 3 | 1211108626 | LIM CHIN CHEN | AI | 1C |
| 4 | 1211108213 | SOO TECK SHEN | AI | 1C |



**Declaration by Group Leader**

I hereby declare that all group members’ names are correctly included in the above section. I hold a copy of this project which I can produce if the original is lost or damaged. I certify that not part of this project has been copied from any other student’s work or from any other source except where due acknowledgement is made in the project. I also acknowledge the use of AI tools (if any) is aimed at brainstorming, rephrasing and enhancing decision-making processes. I affirm our commitment to ethical guidelines, transparency, and responsible utilization of AI technology in acquiring solutions.

Group Leader’s Signature : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Group Leader’s Name : \_\_KEE SHEE HAU\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Group Leader’s ID : \_\_\_1211110319\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Group Member’s Declaration**

(Each group member, including the group leader, must individually fill up and submit this form. This form has to be attached together with submission.)

Group member’s name : \_\_KEE SHEE HAU\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student ID : \_\_\_1211110319\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For the purpose of completing this project, I have performed the following tasks:

I involved in all the discussion with my group members. I was assigned to do the functional and non-functional, requirement modeling and project management of the report part. I was also in charge of doing presentation slide and involve in recording the presentation video

I hereby declare that I have assessed the final submission and I take full responsibility should there be any inaccuracies, incompleteness, omissions, delays or non-submission.



Group member’s signature : \_\_\_ \_\_ \_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Group member’s name : \_\_\_\_KEE SHEE HAU\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Group member’s ID : \_\_\_\_\_\_1211110319\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date : \_\_\_\_\_22/7/2025\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Group Member’s Declaration**

(Each group member, including the group leader, must individually fill up and submit this form. This form has to be attached together with submission.)

Group member’s name : ONG QI REN

Student ID : 121110357

For the purpose of completing this project, I have performed the following tasks:

I participated in every conversation that my group members had. I was tasked with doing the programming portion and explaining to my group member how the system functions generally so they could grasp it better. Along with creating the presentation slides, I was also involved in filming the presentation.

I participated in every conversation that my group members had. I was tasked with doing the programming portion and explaining to my group member how the system functions generally so they could grasp it better. Along with creating the presentation slides, I was also involved in filming the presentation.

I hereby declare that I have assessed the final submission and I take full responsibility should there be any inaccuracies, incompleteness, omissions, delays or non-submission.



Group member’s signature : \_\_\_ \_\_ \_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Group member’s name : ONG QI REN

Group member’s ID : 1211108357

Date : 20/6/25

**Group Member’s Declaration**

(Each group member, including the group leader, must individually fill up and submit this form. This form has to be attached together with submission.)

Group member’s name : Lim Chin Chen

Student ID : 1211108626

For the purpose of completing this project, I have performed the following tasks:

* I involved in all the discussions with my group members.
* I done part 1.0 Introduction, include 1.1 - 1.4.
* I done part 2.0 Background Study.
* I done part 3.0 Requirements Engineering, include 3.1 – 3.7.
* I done part 6.0 Design Modelling, include 6.1 – 6.2.
* I done part 8.0 Project Management, include 8.1 and 8.3.
* I done part 9.0 Conclusion.
* I was also in charge for doing presentation slide and involve in recording the presentation video.

I hereby declare that I have assessed the final submission and I take full responsibility should there be any inaccuracies, incompleteness, omissions, delays or non-submission.

Group member’s signature :



Group member’s name : Lim Chin Chen

Group member’s ID : 1211108626

Date : 20.06.2025

**Group Member’s Declaration**

(Each group member, including the group leader, must individually fill up and submit this form. This form has to be attached together with submission.)

Group member’s name : SOO TECK SHEN

Student ID : 1211108213

For the purpose of completing this project, I have performed the following tasks:

I involved in all the discussion with my group members. I was assigned to do the programming part and providing the overall operation process of the system to my group member for better understanding. I was also in charge of doing presentation slide and involve in recording the presentation video

I hereby declare that I have assessed the final submission and I take full responsibility should there be any inaccuracies, incompleteness, omissions, delays or non-submission.

Group member’s signature : A black line drawn on a white background

AI-generated content may be incorrect.

Group member’s name : SOO TECK SHEN

Group member’s ID : 1211108213

Date : 20.06.2025

[1.0 Introduction 7](#_Toc984206912)

[1.1 Overview of the Project 8](#_Toc979630706)

[1.2 Problem Statements 8](#_Toc297165989)

[1.3 Objectives 8](#_Toc1108954075)

[1.4 Significance of the System 9](#_Toc1990150757)

[2.0 Background Study 9](#_Toc1290390565)

[3.0 Requirements Engineering 10](#_Toc1378110090)

[3.1 Inception 10](#_Toc862892967)

[3.1.1 Stakeholder List 11](#_Toc979048936)

[3.1.2 Points of View 11](#_Toc376550926)

[3.1.3 Target User 12](#_Toc1021627706)

[3.2 Elicitation 12](#_Toc1264889393)

[3.2.1 Gather Requirements through Meeting 13](#_Toc1653899917)

[3.2.2 Elicitation Work Products 14](#_Toc1202490577)

[3.3 Elaboration 15](#_Toc1391150181)

[3.4 Negotiation 15](#_Toc512738135)

[3.5 Specification 16](#_Toc1704832626)

[3.6 Validation 17](#_Toc1380358977)

[3.7 Requirements Management 18](#_Toc2026749660)

[4.0 Functional and Non-Functional 18](#_Toc1685374351)

[4.1 Functional Requirements 18](#_Toc547386135)

[4.2 Non-Functional Requirements 19](#_Toc540815737)

[5.0 Requirements Modelling 20](#_Toc1294128292)

[5.1 Use Case Diagram 21](#_Toc364765571)

[5.2 Class Diagram 21](#_Toc533783782)

[5.3 Sequence Diagram 22](#_Toc398797175)

[5.4 Entity Relation Diagram 24](#_Toc1226802104)

[5.5 Context Diagram 25](#_Toc1052760972)

[5.6 Data Flow Diagram 26](#_Toc243576400)

[6.0 Design Modelling 26](#_Toc624769952)

[6.1 System Implementation 27](#_Toc1247895214)

[6.1.1 Student 28](#_Toc771274068)

[6.1.2 Hostel Administrators 31](#_Toc1793408696)

[6.1.3 Maintenance Staff 34](#_Toc1695473561)

[6.2 Data Dictionary 36](#_Toc457975823)

[7.0 Testing 42](#_Toc1173781015)

[7.1 Admin Test 43](#_Toc869231604)

[7.2 Student Test 43](#_Toc1789864171)

[7.3 Staff Test 44](#_Toc1937739603)

[7.4 Admin Test Result 44](#_Toc2093872589)

[7.5 Student Test Result 45](#_Toc1352037308)

[7.6 Staff Test Result 45](#_Toc313139200)

[8.0 Project Management 46](#_Toc719438542)

[8.1 Project Planning 47](#_Toc777125516)

[8.2 Teamwork 48](#_Toc1646381088)

[8.3 Risk Management 48](#_Toc85822716)

[8.4 Time Management 49](#_Toc401437077)

[9.0 Conclusion 49](#_Toc1451236882)

# 

# **1.0 Introduction**

## 1.1 Overview of the Project

This is a project aimed at ensuring the centralization of a hostel system to make it digitalized and automated through an efficient platform by including a student and a hostel administrator. Through this system, they can see information regarding rooms in real time, make complaints and online payments, check the cleaning schedule of rooms, and submit maintenance requests. The administrators can operate the room information, student information, organize cleaning cycles, take care of maintenance routine processing, and handle complaints.

## 1.2 Problem Statements

* The students want to know the latest details of renting the rooms, as it is important to make the appointment in the management office personally. This will be more hectic since they have to spend time going to the office.
* Administrators work in a day-to-day basis using paper forms, oral communication and manual spreadsheets which retards the efficiency of the processing and also an error-prone method.
* Students are uncertain of the fact that their complaints have not been attended to appropriately by the management, or they have been filed off. This will cause a poor academic experience for students.
* Administrators are supposed to remind the students verbally in case of any additional maintenance costs, and that will consume a lot of time, and this leads to someone as well.
* Absence of a central and unified personalized system of the room, students' records, and reservations causes inefficient management of the hostel and a lack of coordination.

## 1.3 Objectives

* To give students an all-inclusive platform to handle all their lodging activities like room sighting, filing complaints, and tracking payments.
* Enhance the rate of renting the hostels so as to make sure that the rooms that are needed by students can be set up as much as possible.
* To enable the administrators to regulate room distribution, address grievances, modify cleaning arrangements, and observe payment of fees in a single platform effectively.
* To manage requests effectively and to eliminate duplicates in the systems, which consume a lot of time and energy of the administrators, thus freeing their time to do things that matter.
* Lessen the inconvenience of misunderstanding and enhance students' stay.

## 1.4 Significance of the System

The hostel management system is meant to make it transparent and efficient. Accountability and trust are encouraged, as students are able to monitor their request processes, payment, and complaints on a real-time basis. This system will minimize errors and at the same time centralize the data. Digital databases will do away with the discrepancies that generally exist in the manual records, making it easier to report and analyze. Also, the accommodation satisfaction will be found improved among the students, and the administrators can manage the hostel in a better way.

# **2.0 Background Study**

This issue concerns students a lot, and areas of concern about the management of hostels within the institutions of learning are very important, particularly to learning institutions that own residential facilities of their own. The everyday life activities of a hostel are more about keeping things clean, handling maintenance-related problems, allocating rooms, and everything that arises in the minds of students. However, historically, most hostels used manual tools like paper forms, logbooks, or even speech to conduct these tasks. Such types of approach can be effective in the small hostels, and when the residents increase, they are not effective, as well as not error-free, and are hard to handle.

A friendly hostel management system in this region would give a platform to inform the students about all hostel matters in real time. It is a very simple but easy-to-use web/mobile application or tool, which contains all the information and data in one place. The technologies used on this system include relational database, real-time alerts, and safe logging facilities to give the system rapid, safe, and effective. UI/UX Design UI/UX design also plays a key role that ensuring a platform is comfortable and easy to use for all categories of users.

Hostels can be automated using an online-based integrated hostel management system that raises transparency, response time, and staff and student satisfaction. Experience with analogous systems has shown that digital tools can be used to achieve an immense increase in the speed of processing and response to complaints. We take the experiences of implemented systems at the universities in Southeast Asia and ensure their customization to our university.

# **3.0 Requirements Engineering**

## 3.1 Inception

The project of Hostel Management System development will help to cope with problems and challenges that all the stakeholders in the management of the hostel tackle. The main issues in this respect are the inefficiency of even simple tasks, such as making a reservation of the room, a need to visit the premise to ask questions about the availability of the room, large numbers of data and information that is to be manually processed, the inability of the hostels to get feedback and requests as lost opportunities of communication, such as getting feedback and requests. Several other reasons underlie this necessity: there was no centralized, web-based system that helps to store the information about students, assign them to rooms, offer them payment tools, and arrange cleaning procedures.

Following the recognition of those types of needs, MMU asked us to come up with a Hostel Management System to be used in the university hostel systems. The purpose of this system is to ease the management functions and offer an operational ground to students and administrators. The needs and expectations of each of the stakeholder groups in the case are very specific, and the system must reflect the same. Thus, we are interested in constructing a functioning system to provide a solution that could be employed in making life better in the day-to-day operation of living in a hostel.

Stakeholder interviews, such as with the administrators, students, are the foundation we shall base our requirements-gathering process. This way, we will hopefully be exposed fully to their expectations, which can be translated into practical features of the system.

## 3.1.1 Stakeholder List

|  |  |  |
| --- | --- | --- |
| Stakeholder | Role | Interest in System |
| Students | Primary users/hostel occupants | View room availability, reserve visits, book rooms, and submit complaints easily. |
| Hostel Administrators | System managers and data handlers | Manage room assignments, student records, complaints, and cleaning schedules efficiently. |
| Maintenance Staff | Maintain and repair hostel rooms based on requests | Receive clear and timely complaints and maintenance tasks for resolution |

## 3.1.2 Points of View

One of the most active user groups is the students. On their part, hostel life is already full of academic tensions and the process of adjustments. They do not wish to speculate on a case like has the request of a leaking pipe has been resolved or not or have their complaints have not been attended to. They are waiting to interact with an easy-to-use platform and one that updates information on a real-time basis.

To the hostel administrator, the monitoring of all these information manually by using spreadsheets and papers is time-wasting and tiresome. They desire to finish it so that they can access all the information about students and rooms at a central place; this means that they can easily update their records and quickly monitor complaints. A well planned set up would enable them concentrate on decision making processes rather than documentation.

The ones who receive and take any action on requests that are being submitted by students are the maintenance staff. On the one hand, they would prefer a system that provides them with control institutions in the form of clear lists of things to do, accompanied by pertinent information such as room number, types of problems, and emergency nature, among others. Provided that this is possible, they like having images attached to their command so that they can save time and bring the right tools.

## 3.1.3 Target User

The system is built primarily for university students who live in hostels and need a user-friendly platform for managing room rentals, complaints, payments, and cleaning schedules. Their primary concerns include ease of access, instant updates, and transparent fee records. The secondary target user is the hostel administrator, who manages the backend operations such as room allocation, fee updates, and complaint handling. They need a comprehensive platform to help them manage the data or student records and enable them to update new information time-on-time. Finally, maintenance staff interact directly through admin. The system ensures they receive up-to-date assignments without confusion.

## 3.2 Elicitation

In this project, we tried our best to understand the requirements of users in hostel management to adopt a genuine, functional, and practical system. Interviewing, observation, and documentation analysis were used as the three methods of deriving functional and non-functional requirements. With these methods, we can get into the practices of these users in carrying out their activity in everyday life and the issues related to the same.

In the interviews, we came to learn that in the current system, there are times when the students may be spending a lot of time in long queues to book rooms as well as make maintenance requests. According to the administrators, manual management of records was not an easy task. In other cases, the manually stored records may run out of data or have discrepancies. Even in the maintenance department, employees said that they find themselves engaged regularly with ill-phrased or refurbishment office actions that do not add the best to their work.

We had some observations that relate to how day-to-day hostel operations are carried out. As an example, we have observed what is then only the management of processes such as room booking, making visit arrangements, monitoring the maintenance process, process purpose, and activity flow and visibility of bottlenecks. Some documents were also analysed, among which the booking forms and maintenance logs were presented to understand the scope of data that is usually gathered, as well as the general pattern that the information has to make.

The proposed system development aims to address these problems in hostel management, where the current process is manual. This would change to a web-based digital platform. It is built with HTML, CSS, and JavaScript front-end using PHP and MySQL for back-end functionality. XAMPP was chosen as the local development environment because it allows building and testing before deploying the application. This setup ensures that tasks such as booking, record management, and maintenance tracking are faster, more accurate, and easier for all parties involved.

Students, administrators, maintenance staff, and developers are the most important stakeholders who are involved in this system. The system will assist students in viewing the available rooms, reserving their dorms, maintaining requests, and personal information. All the data related to the hostels, such as the room status and booking, will be supervised by the administrators. Repair requests will go to the maintenance staff, who will update their status in real time. Although the system is used at other school departments directly, they still assist in monitoring and effectiveness of the system. Being developers, we investigated the requirements of the user, researched the similar hostel sites, and learned the current procedures to compose a solution that would take care of the hostel community.

## 3.2.1 Gather Requirements through Meeting

Based on the visits to stakeholders and users, we analysed their responses to define the following requirements:

|  |  |
| --- | --- |
| Stakeholders and Users | Requirements |
| Students | * View Room Information: Students can browse dormitory rooms, see availability, and view room details such as type, capacity, and status. * Reserve Hostel Visits: Students want to schedule room viewing appointments online without needing to visit the office physically. * Complaint Tracking: A system that can let them submit and track maintenance complaints, including status updates. * Payment History: Access to user payment status, amount due, and previous transactions in real time. * View and Edit Profile: Students can update their personal information, such as contact number and email. |
| Hostel Administrators | * Manage Room Data: Ability to add, update, and delete data and records. * Manage Student Information: View and update student data related to hostel use. * Handle Complaints: Receive complaints from students, update resolution status, and assign issues to maintenance staff. * Publish Cleaning Schedules: Efficiently create and update cleaning tasks for different blocks. |
| Maintenance Staff | * Update Task Status: Mark tasks as completed or delayed. * Task Dashboard: View a clear, daily list of assigned maintenance tasks. |

## 3.2.2 Elicitation Work Products

As a result of the elicitation process, we produced the following work products to document and communicate the gathered requirements, which are use case diagrams, class diagrams, data flow diagrams, and sequence diagrams. These work products can be used as documentation, planning guides, and alignment tools for both technical and non-technical stakeholders.

## 3.3 Elaboration

Through the elaboration phase, we were able to improve and enhance our comprehension of the system requirements and convert them into technical models that would direct our implementation and design. During this phase, which served as a link between obtaining user requirements and developing the system itself, visual diagrams were made to help in planning, communicating, and verifying our design ideas.

We started by separating each of the functional requirements collected during the collection stage into specific use cases. A Use Case Diagram that put out every potential interaction between the system and its users—including making reservations, submitting complaints, and scheduling cleanings—was then used to explain these. We were able to clearly see which features needed to be developed and how they related to one another as a result.

In order to model the database structure, we created an Entity Relationship Diagram (ERD). Tables, primary and foreign keys, and the connections between data items were displayed. To prevent duplication and ensure effective data access, normalization was used. For backend development and data integrity, this diagram was necessary.

To show the sequence that occurs for important interactions, such as the student login and complaint submitting processes, sequence diagrams were made. These ensured consistency in module communication and helped us in visualizing system behavior over time. Data Flow Diagrams (DFDs) were also created to show how data moves through the system and is processed from input to output.

This process of elaboration gave us a significantly better knowledge of the system's logical and structural operation. Our development effort was guided by these models, which also ensured a more structured, scalable, and maintainable codebase.

## 3.4 Negotiation

In the development process, requirement negotiation was important to ensuring our goals and the demands of our stakeholders were in alignment with what we could really produce. Our goal was to achieve a realistic balance between feasibility and aspirations.

* Prioritization of Requirements: The MoSCoW technique (Must have, Should have, Could have, Won't have) was used to determine the effect and urgency of each stakeholder demand. By delaying optional features like automated email warnings, we were able to concentrate on providing essential services like reserving rooms and handling complaints.
* Trade-offs and Constraints: We were honest with stakeholders about the limits of the system. Early communication of these limitations helped us to establish trust and prevent disappointment later.
* Collaborative Decision-Making: Every significant choice was reached in concert. Stakeholder representatives were asked to participate in conversations involving requirements, and they collaborated to reach resolutions.
* Conflict Resolution: We developed different role-based interfaces for each user group where stakeholders had conflicting demands (for example, administrators wanted detailed control panels while students preferred simplicity).

## 3.5 Specification

The specification phase involved translating the agreed requirements into a formal, structured document that defined exactly how the system should behave. We use it as a blueprint to ensure that the system hits the users' needs and technical requirements.

Functional Requirements:

Student:

1. Room Information and Booking
   * Students can view available rooms with their details, such as room number, type, capacity, and availability status.
   * Students can request to book an available room.
   * The system prevents double booking by checking real-time room status.
2. Complaint Submission and Tracking
   * Students can complain by specifying issue types.
   * The complaint is saved and marked as “Pending” by default.
   * Students can track the complaint status.
3. Payment History
   * Students can view their hostel fee payment history.
   * Information includes payment date, amount, method, and status.
4. Cleaning Schedule
   * Students can check weekly cleaning schedules based on their hostel block.
   * The schedule includes cleaner name, date, and task type.

Hostel Administrators

1. Room Management
   * Admin can add, update, assign, or remove room records.
   * Each room includes status and is linked to student name.
   * Room data is viewable in a dashboard summary.
2. Student Profile Management
   * Admins can view and edit student profiles linked to hostel records.
   * Admins can reassign students to different rooms and update fee statuses.
3. Complaint Management
   * Admins receive a list of all submitted complaints.
   * Each complaint can be assigned to maintenance staff and updated with resolution notes.
   * Admins can mark complaints as "Resolved" or "Rejected."
4. Fee Tracking
   * Admins can input hostel fee details per student, which include amount due, deadline, and remarks.
   * Admins can mark payments as “Paid” or “Unpaid.”
5. Cleaning Schedule
   * Admins can create and assign cleaning tasks to blocks and cleaners.
   * Tasks include type, date, and assigned personnel.
   * Schedules can be updated or deleted on time.

## 3.6 Validation

A validation process is needed to ensure each element of the requirements is consistent with the overall objectives of the system. We organized meetings with the stakeholders to validate the correctness of the implemented features.

* Stakeholders confirmed that all the expected functions had been implemented when we showed the demo to them.
* Feedback from students and admins had been recorded and categorized into different issues and suggestions.
* We developed a validation checklist to verify every point with live system behavior.
* Make sure there are no conflicts between any of the requirements.

## 3.7 Requirements Management

It was important for us to stay flexible but also disciplined to avoid losing track of decisions. To manage this, we adopted a structured process with traceability, change control, versioning, and validation as core principles.

* Traceability: All requirements were labelled with identifiers, making them easy to reference and trace throughout development.
* Change Control: Any change to the system’s scope and requirements is to follow a formal review and approval process.
* Versioning: Each version will include a detailed changelog so we could track how the system evolved and avoid confusion at the same time.
* Validation and Verification: All requirements were validated through stakeholder feedback and verified during system testing.

# 4.0 Functional and Non-Functional

## 4.1 Functional Requirements

**1. Log in System**

* Users can register and create accounts in the system as admin, student or staff
* Users can log in to the account
* If the credentials are not compatible with registration data in database, they will be rejected by the system.

**2. Dashboard**

* Admin can view total room, total student and pending request.
* Users can change the theme
* Student can view my room, due fee and active requests
* Staff can view cleaning task and maintenance tasks

**3. Room Management System**

* Admin can create room, allocate and deallocate room for students.
* Student can apply for room and view room detail

**4. Student Complaint System**

* Admin can update the status of complaints
* Students can submit complaint request

**5. Cleaning Schedule Management**

* Admin can create cleaning schedule for staff
* Staff can view assigned task and mark as completed

**6. Fee Management System**

* Admin can create fee for student
* Student can make payment

**7. Maintenance System**

* Admin can assign task to staff
* Student can submit maintenance request
* Staff can view maintenance task and mark as completed

## 4.2 Non-Functional Requirements

**1. Security**

* The system must be secure like user data should be encrypted when the data is stored and transmitted.
* The system must control access to sensitive information depending on user roles and permissions.

**2. Availability**

* The system need to maintain high availability during times of high demand of users.
* The system should be built to be able to handle increased traffic without downtime or experiencing performance issues.

**3. Usability**

* The user interface employs Bootstrap for responsive design.
* Navigation is simple, with consistent headers, footers, and icons.
* Error messages are clear and actionable (e.g., “Room already assigned”).

**4. Maintainability**

* The system should enable simple maintenance
* The system’s code should be well documented with comments that make it obvious what each module, function and class does.

**5. Performance**

* The system need to be able to support multiple users at once without any performance issue
* The system need to load quickly with respond times not exceed 3 seconds for common task.
* The system need to maintain a high level of stability to prevent system crash and prevent bugs and error.

**6. Scalability**

* The system must be scalable to be able to handle the increases in user numbers and task numbers.
* The system should implement load-balancing mechanisms to ensure it consistently performs under severe loads.

# **5.0 Requirements Modelling**

## 5.1 Use Case Diagram

**A diagram of a diagram

AI-generated content may be incorrect.**

In this Use Case Diagram of Hostel Management System has three actors which are Student, Staff and Admin. The diagram has offered a top-view system function to ensure that common points between developers, stakeholders and users. The student is a typical system user who works with simple room applications options. The main actions that they operate are making payment, submitting complaint and submitting maintenance.

However, the admin can manage some function such as log in, manage rooms, manage fees, assign task and handle complaints. These can help admin to manage the hostel become more easily. In addition, staff can manage some function such as log in, mark task completed and resolve maintenance requests.

## 5.2 Class Diagram

**A screenshot of a computer

AI-generated content may be incorrect.**

## 5.3 Sequence Diagram

**A diagram of a project

AI-generated content may be incorrect.**

**A diagram of a system

AI-generated content may be incorrect.**

**A diagram of a task

AI-generated content may be incorrect.**

## 5.4 Entity Relation Diagram

**A diagram of a computer

AI-generated content may be incorrect.**

1. Users:

* One user only can have one staff id
* One user can allocate one room
* One user can have many fees
* One user can have many maintenances requests
* One user can have many complaints
* One user can have many rooms applications

1. Rooms:

* One room can have many rooms allocations
* One room can have many maintenances request
* One room can have many cleanings schedules
* One room can have many rooms applications

1. Staff:

* One staff can be assigned to many maintenance requests
* One staff can have many cleaning schedules

1. Fees:

* One fee can have many fee payment

## 5.5 Context Diagram

**A diagram of a system

AI-generated content may be incorrect.**

**A diagram of a hotel management system

AI-generated content may be incorrect.**

**A diagram of a task

AI-generated content may be incorrect.**

## 5.6 Data Flow Diagram

**A diagram of a company

AI-generated content may be incorrect.**

# **6.0 Design Modelling**

Design modelling is where our concepts started to take shape. We focused our attention on turning the elaboration stage into real design blueprints that would lead the development of our system. This required establishing the architecture, creating the user interface (UI), and organizing how the various system modules would interact with one another. The goal of design modelling was to make sure that each system component was useful and easy to use in addition to achieving technical requirements.

We began by creating the User Interface (UI), ensuring it was clear and easy to use. The webpage provides a clear interface for students to view room availability, complaint submitting, and payment tracking. On the other side, administrators receive a dashboard summary that includes information on payment statuses, cleaning schedules, complaints, and rates of occupancy.

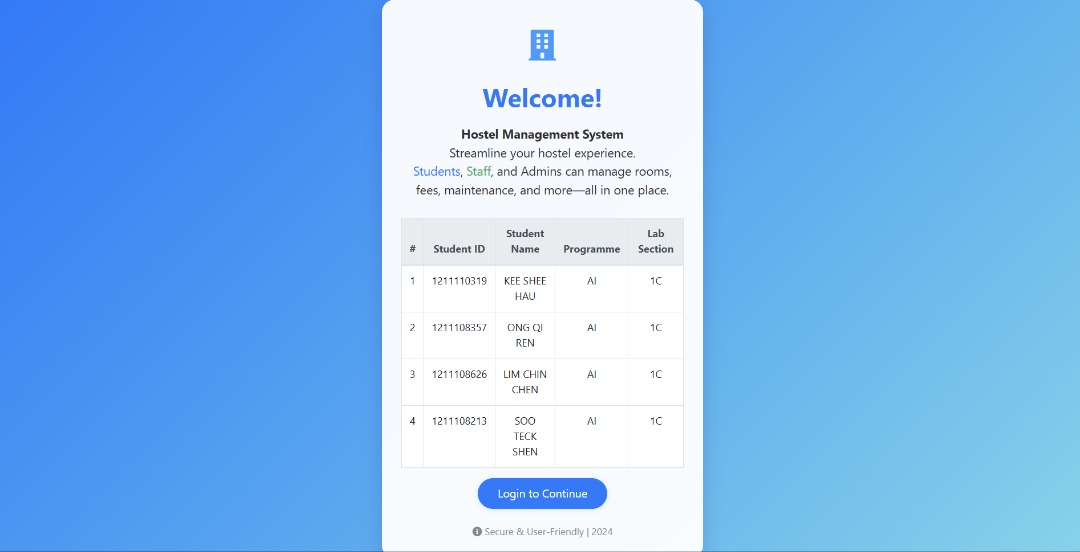
We used PHP and MySQL to provide a modular framework for the backend design. Development and debugging were made easier by the fact that each function, such as room assignment, complaint submission, or login, was controlled by a separate module. For example, the complaint module wouldn't be impacted by changes made to the room management module. In the event that the system needed to incorporate new features like hostel reviews or real-time chat help, this division of responsibilities allowed for efficient communication and future scalability.

The structure of the database followed to a normalised structure, with tables like Students, Rooms, Payments, Complaints, and CleaningSchedules having well-defined relationships with one another. For relational integrity, we added foreign keys, and for faster search execution, we optimised the indexes. In order to ensure that each item and connection was accurately represented and that the data could be accessed quickly, the structure was visualised using ERD diagrams.

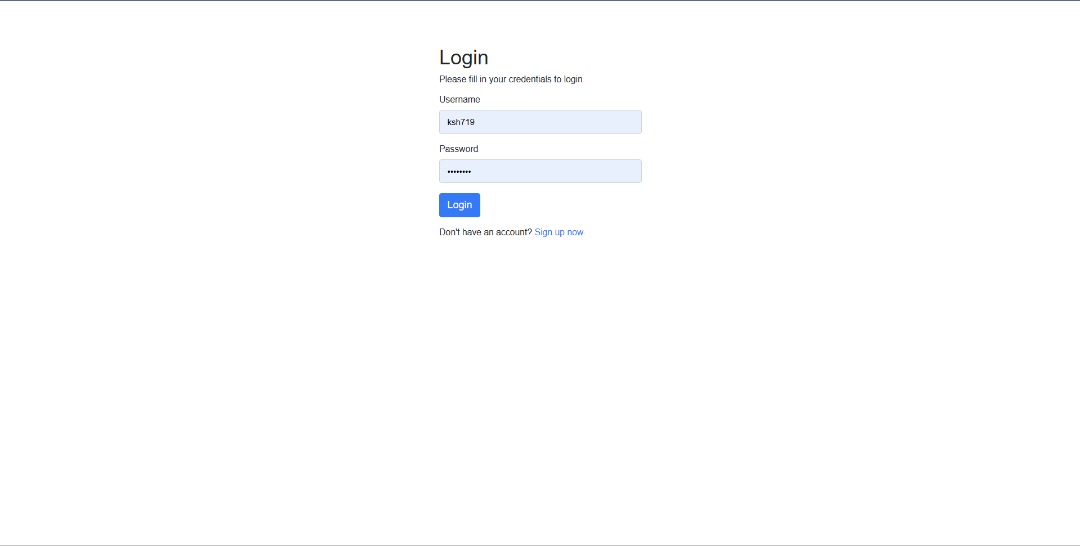
## 6.1 System Implementation

## 6.1.1 Student

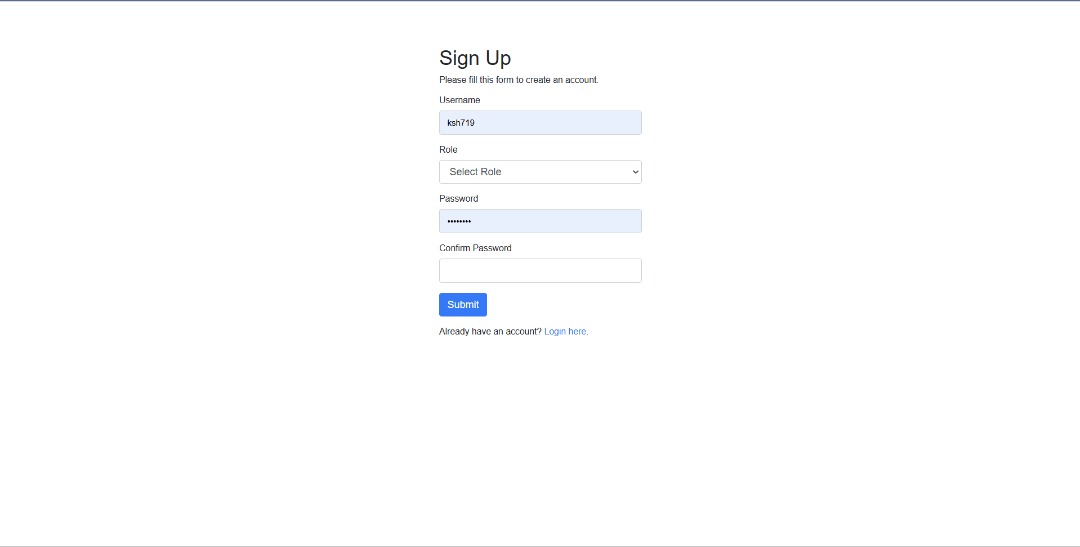
Home page:

This image shows the Hostel Management System homepage; the page shows the profile of the employee, such as ID, full name, program, and lab section.

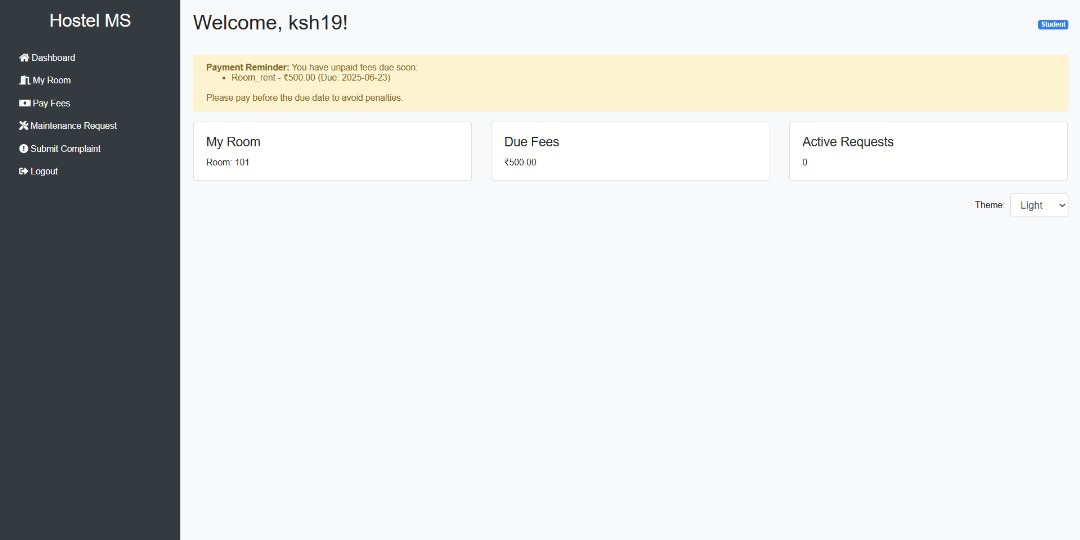
Login Page:

This image shows the login page for students, admins, or staff to log in to their account. They will need to fill in the correct username and password to log in.

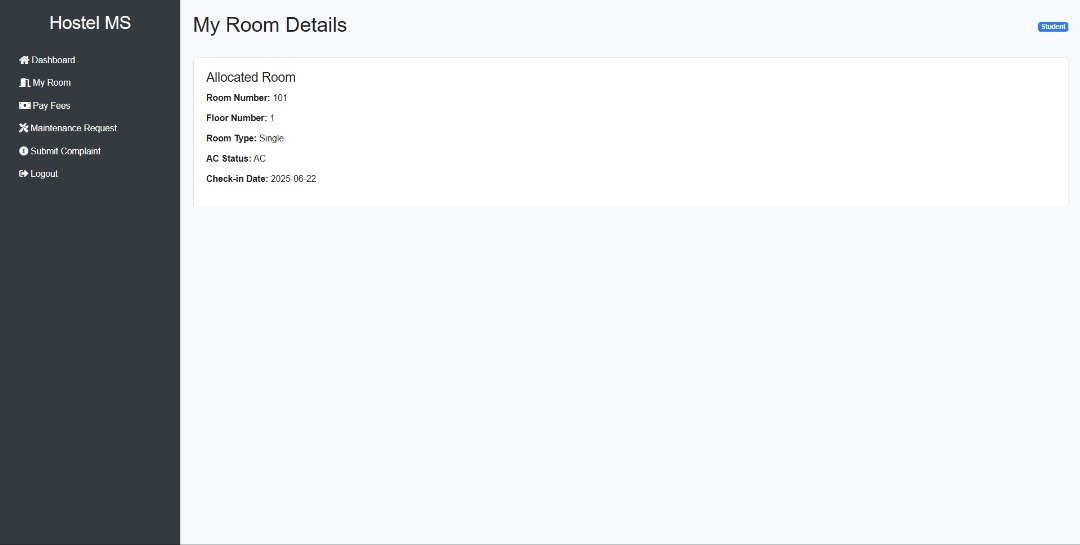
Sign-Up Page:

This image shows a sign-up page for users that haven’t created an account before. They will need to more steps, which are selecting role and confirm a password to submit it.

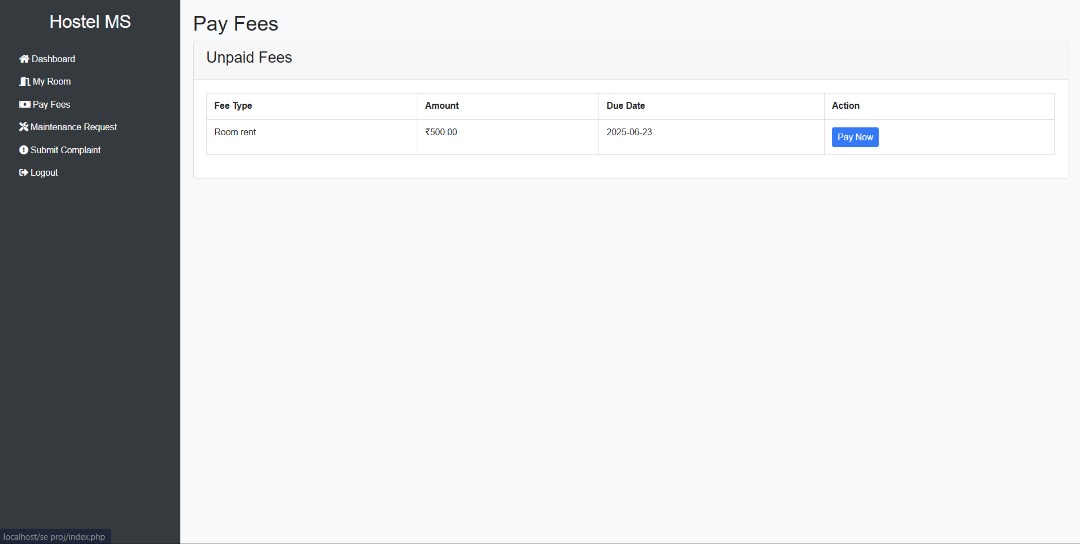
Student Dashboard:

This image shows a student dashboard with their room number, due fees amount, and active request. They also can change the theme into light or dark.

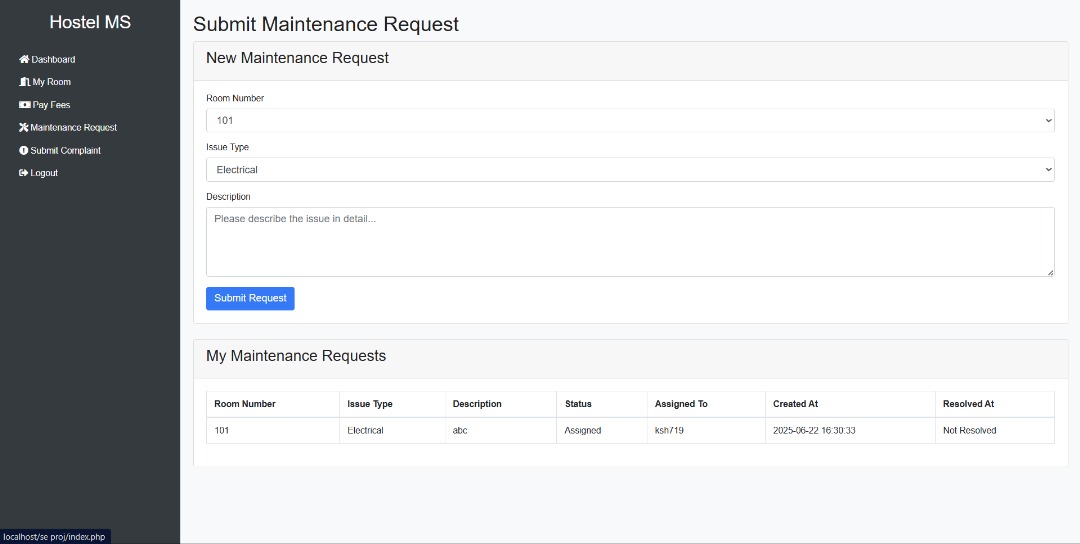
Room Details Page:

This image shows the room details for the user, including room number, floor number, room type, AC status, and check-in date.

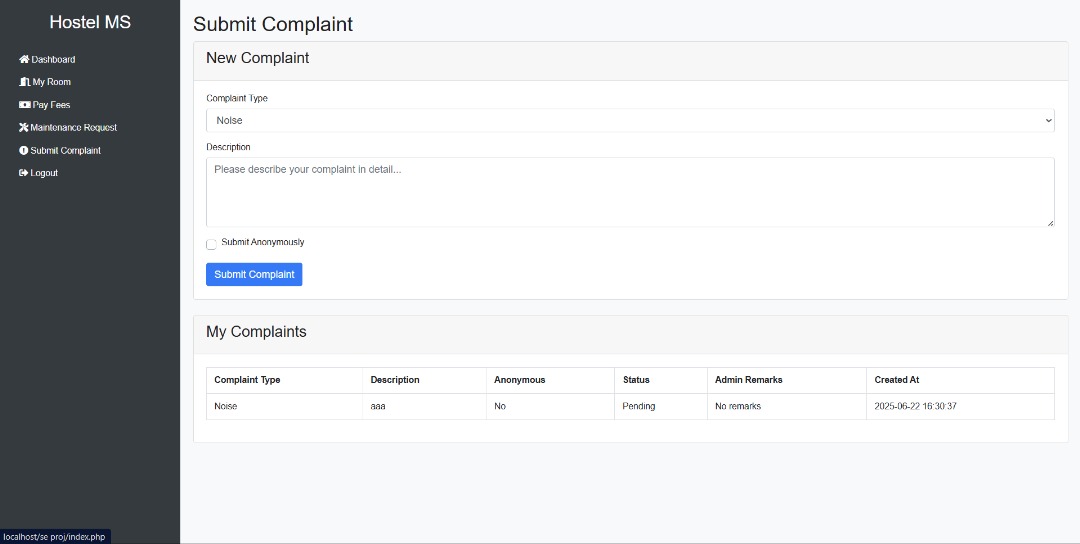
Payment Page:

This image shows that students can pay their fees via this platform. It will list out the fee type, amount, due date, and action.

Submit Maintenance Request Page:

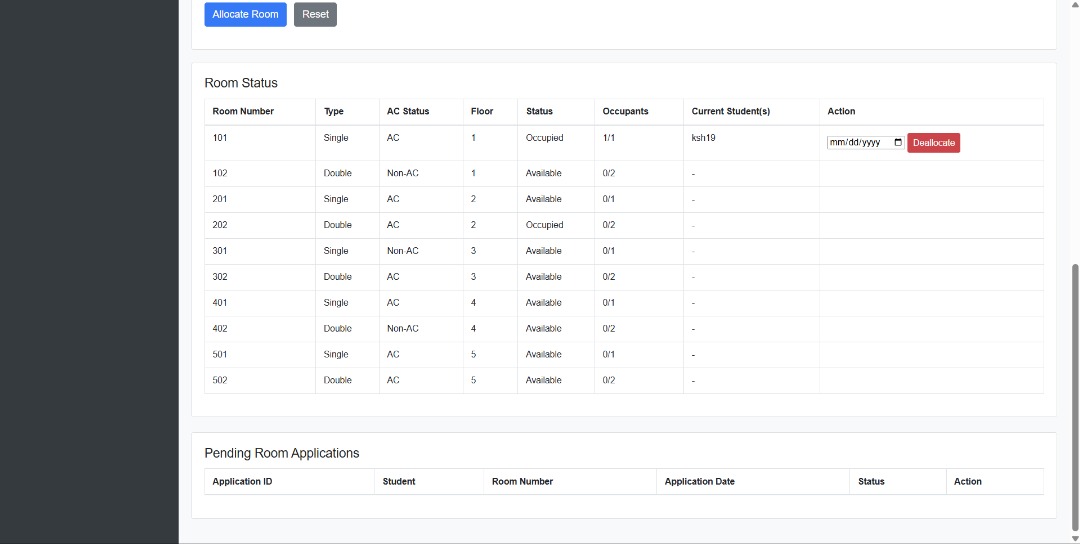
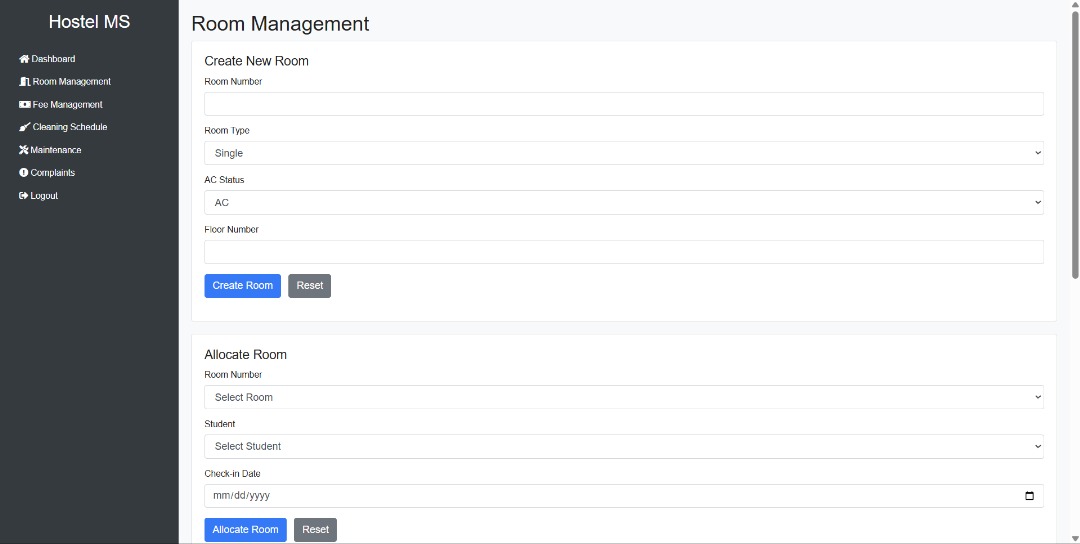
This image shows that students can submit their maintenance requests that include their room number, issue type, and description. They can also track the status of it.

Submit Complaint Page:

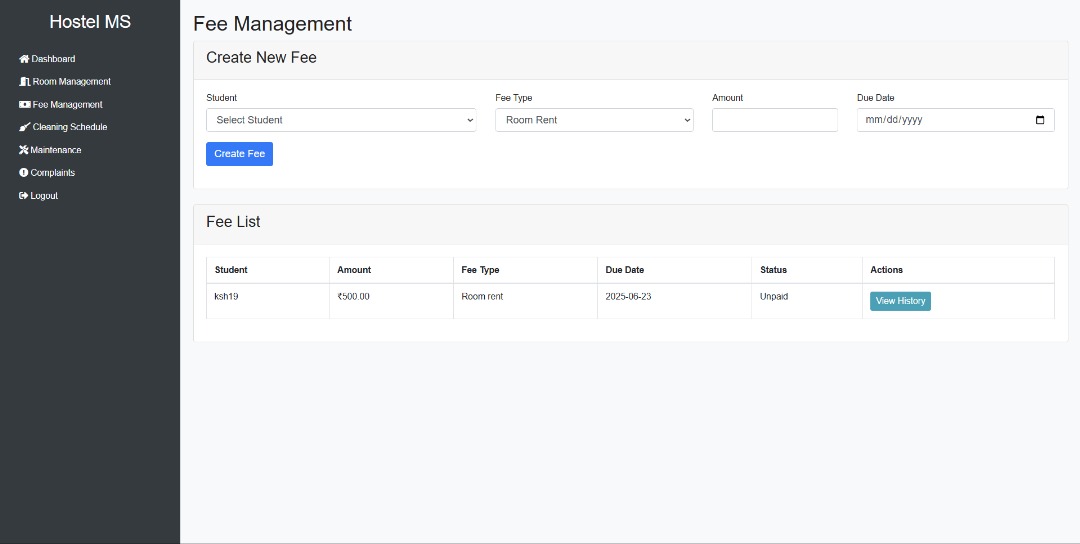
This page shows that students can also submit their complaint to admin. The complaint form will include complaint type and description. Students can choose to submit anonymously and check the status.

## 6.1.2 Hostel Administrators

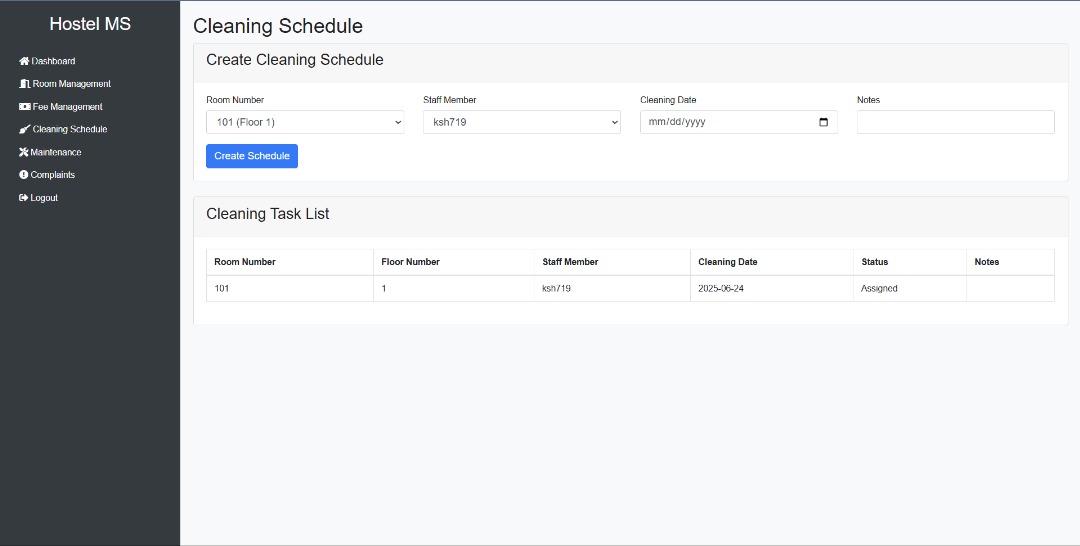
Room Management Page:

This image shows that an admin can create or allocate a room with just simple fill-in-the-blank information needed.

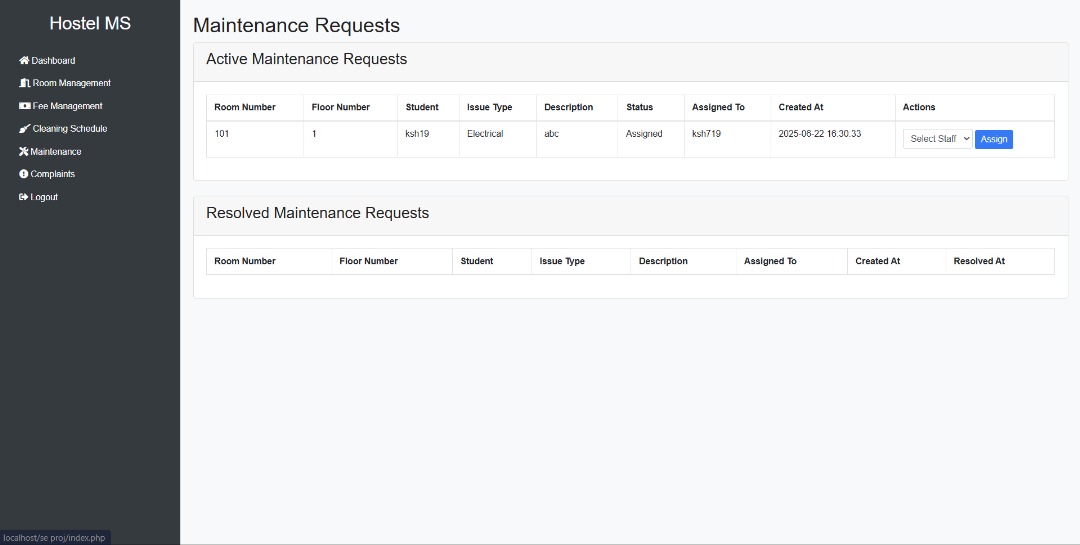
Fee Management Page:

This image shows that the admin can create a new fee directly and view the fee list visually.

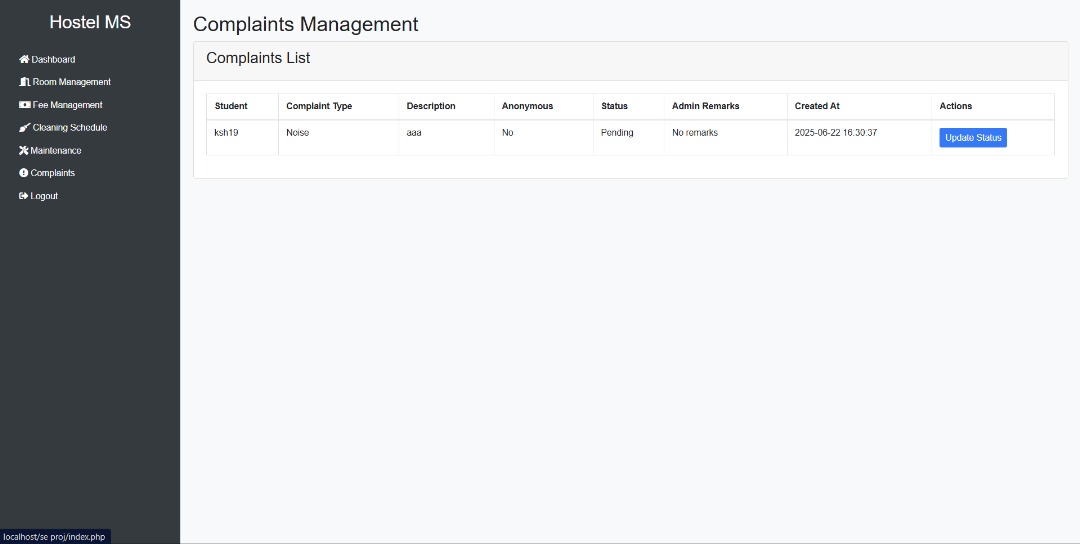
Cleaning Schedule Page:

This image shows that the admin can create a cleaning schedule or view the listed cleaning task.

Maintenance Requests Page:

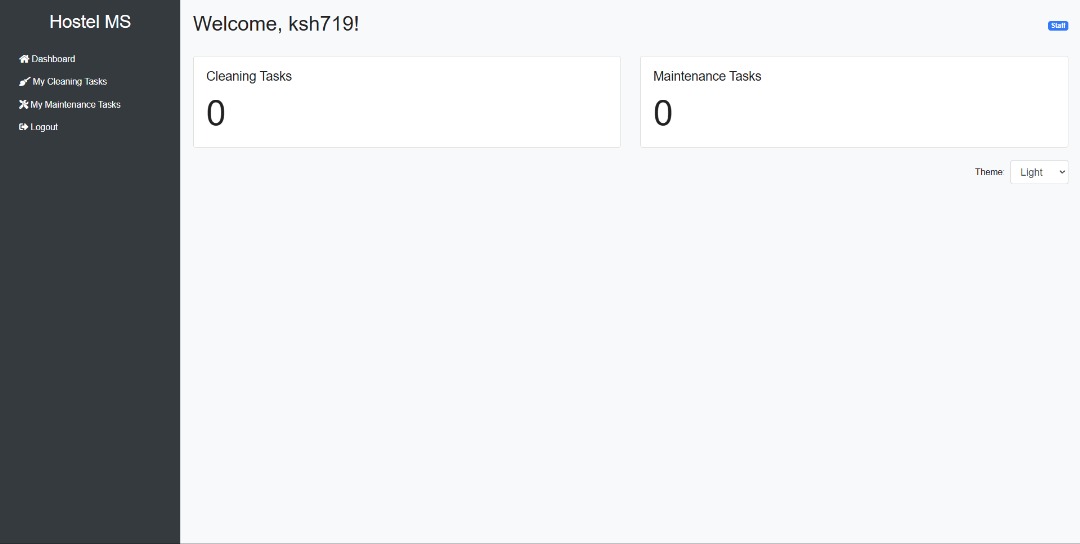
This image shows the maintenance request that the admin receives from students. The status of each request will be marked and include the details of the requests.

Complaints Management Page:

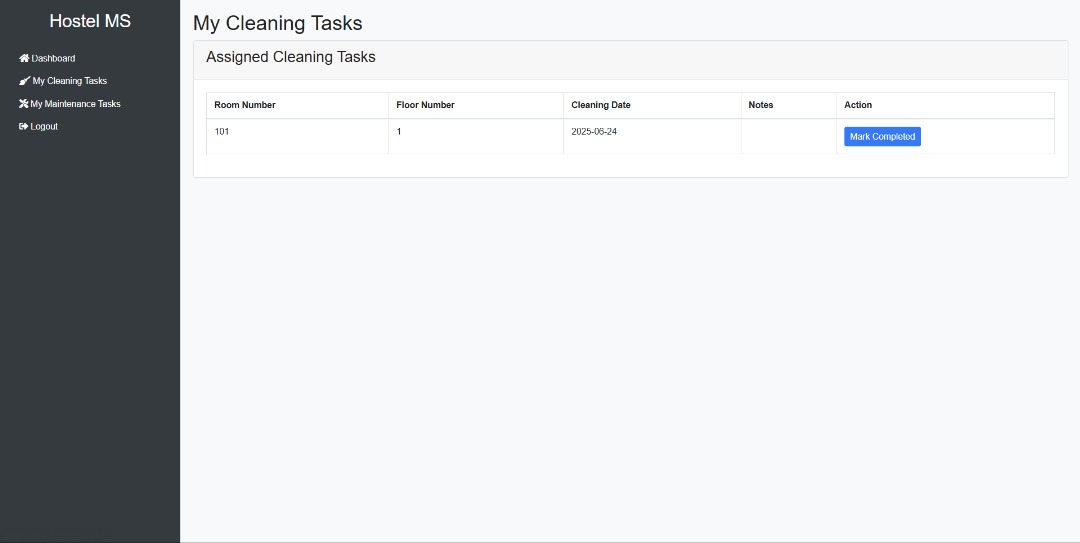
This image shows the complaints that the admin receives from students. The status of each complaint will be marked and include the details of the complaint.

## 6.1.3 Maintenance Staff

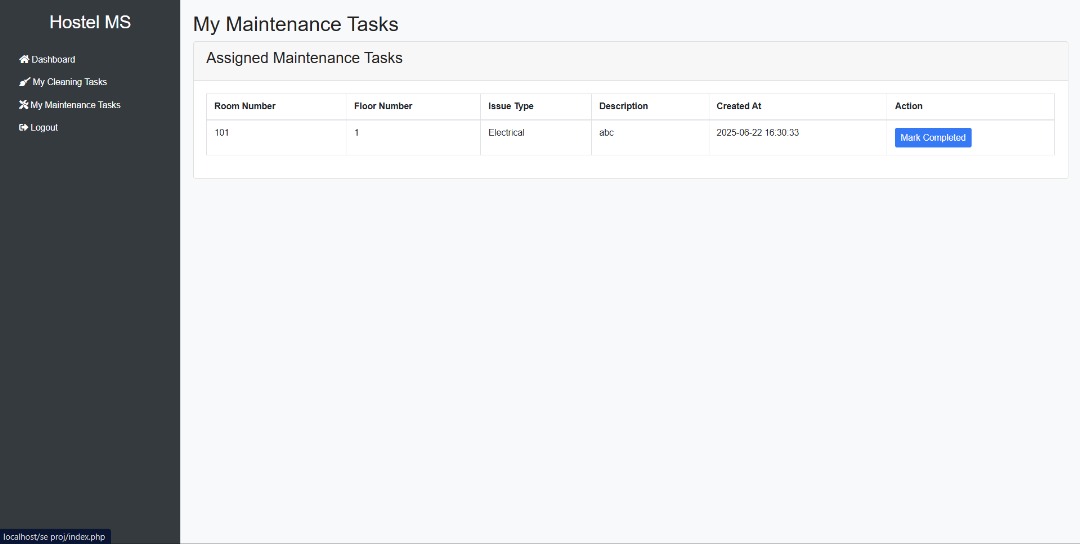
Staff Dashboard:

This image shows the cleaning tasks and maintenance tasks for staff.

My Cleaning Tasks Page:

This image shows the details of the cleaning tasks, and they can mark them as completed after they finish.

My Maintenance Tasks Page:

This image shows the details of the assigned maintenance tasks, and they can also mark them as completed after they finish.

## 6.2 Data Dictionary

Here's the complete data dictionary in a single table format with all field names included:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Table Name** | **Field Name** | **Data Type** | **Field Length/Values** | **Constraint** | **Description** |
| staff | id | INT | 8 | PK, AUTO\_INCREMENT, NOT NULL | Staff ID |
| staff | name | VARCHAR | 100 | NOT NULL | Staff Name |
| staff | created\_at | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Creation timestamp |
| users | id | INT | 8 | PK, AUTO\_INCREMENT, NOT NULL | User ID |
| users | username | VARCHAR | 50 | UNIQUE, NOT NULL | Username |
| users | password | VARCHAR | 255 | NOT NULL | Password |
| users | role | ENUM | ('admin','student','staff') | NOT NULL | User role |
| users | staff\_id | INT | 8 | FK (staff.id) | Reference to staff |
| users | created\_at | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Creation timestamp |
| rooms | room\_number | VARCHAR | 10 | PK, NOT NULL | Room number |
| rooms | room\_type | ENUM | ('single','double') | NOT NULL | Room type |
| rooms | ac\_status | ENUM | ('AC','Non-AC') | NOT NULL | AC status |
| rooms | floor\_number | INT | - | NOT NULL | Floor number |
| rooms | status | ENUM | ('available','occupied','maintenance') | DEFAULT 'available' | Room status |
| rooms | created\_at | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Creation timestamp |
| room\_allocations | id | INT | 8 | PK, AUTO\_INCREMENT, NOT NULL | Allocation ID |
| room\_allocations | room\_number | VARCHAR | 10 | FK (rooms.room\_number) | Room number |
| room\_allocations | student\_id | INT | 8 | FK (users.id) | Student ID |
| room\_allocations | check\_in\_date | DATE | - | NOT NULL | Check-in date |
| room\_allocations | check\_out\_date | DATE | - | - | Check-out date |
| room\_allocations | status | ENUM | ('active','completed') | DEFAULT 'active' | Allocation status |
| room\_allocations | created\_at | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Creation timestamp |
| fees | id | INT | 8 | PK, AUTO\_INCREMENT, NOT NULL | Fee ID |
| fees | student\_id | INT | 8 | FK (users.id) | Student ID |
| fees | amount | DECIMAL | 10,2 | NOT NULL | Fee amount |
| fees | fee\_type | ENUM | ('room\_rent','maintenance','other') | NOT NULL | Fee type |
| fees | due\_date | DATE | - | NOT NULL | Due date |
| fees | status | ENUM | ('paid','unpaid') | DEFAULT 'unpaid' | Payment status |
| fees | created\_at | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Creation timestamp |
| fee\_payments | id | INT | 8 | PK, AUTO\_INCREMENT, NOT NULL | Payment ID |
| fee\_payments | fee\_id | INT | 8 | FK (fees.id) | Fee ID |
| fee\_payments | amount | DECIMAL | 10,2 | NOT NULL | Payment amount |
| fee\_payments | payment\_date | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Payment date |
| fee\_payments | payment\_method | ENUM | ('cash','online transfer','e\_wallet','credit\_debit\_card','other') | NOT NULL | Payment method |
| fee\_payments | transaction\_id | VARCHAR | 100 | - | Transaction ID |
| fee\_payments | status | ENUM | ('success','failed','pending') | DEFAULT 'pending' | Payment status |
| cleaning\_schedule | id | INT | 8 | PK, AUTO\_INCREMENT, NOT NULL | Schedule ID |
| cleaning\_schedule | room\_number | VARCHAR | 10 | FK (rooms.room\_number) | Room number |
| cleaning\_schedule | staff\_id | INT | 8 | FK (staff.id) | Staff ID |
| cleaning\_schedule | cleaning\_date | DATE | - | NOT NULL | Cleaning date |
| cleaning\_schedule | status | ENUM | ('assigned','completed') | DEFAULT 'assigned' | Cleaning status |
| cleaning\_schedule | notes | TEXT | - | - | Notes |
| cleaning\_schedule | created\_at | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Creation timestamp |
| maintenance\_requests | id | INT | 8 | PK, AUTO\_INCREMENT, NOT NULL | Request ID |
| maintenance\_requests | room\_number | VARCHAR | 10 | FK (rooms.room\_number) | Room number |
| maintenance\_requests | student\_id | INT | 8 | FK (users.id) | Student ID |
| maintenance\_requests | issue\_type | ENUM | ('electrical','plumbing','furniture','other') | NOT NULL | Issue type |
| maintenance\_requests | description | TEXT | - | NOT NULL | Description |
| maintenance\_requests | status | ENUM | ('pending','assigned','in\_progress','resolved') | DEFAULT 'pending' | Request status |
| maintenance\_requests | assigned\_to | INT | 8 | FK (staff.id) | Assigned staff |
| maintenance\_requests | created\_at | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Creation timestamp |
| maintenance\_requests | resolved\_at | TIMESTAMP | - | - | Resolution timestamp |
| complaints | id | INT | 8 | PK, AUTO\_INCREMENT, NOT NULL | Complaint ID |
| complaints | student\_id | INT | 8 | FK (users.id) | Student ID |
| complaints | complaint\_type | ENUM | ('noise','food','behavior','other') | NOT NULL | Complaint type |
| complaints | description | TEXT | - | NOT NULL | Description |
| complaints | is\_anonymous | BOOLEAN | - | DEFAULT FALSE | Anonymous flag |
| complaints | status | ENUM | ('pending','in\_progress','resolved') | DEFAULT 'pending' | Complaint status |
| complaints | admin\_remarks | TEXT | - | - | Admin remarks |
| complaints | created\_at | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Creation timestamp |
| complaints | resolved\_at | TIMESTAMP | - | - | Resolution timestamp |
| room\_applications | id | INT | 8 | PK, AUTO\_INCREMENT, NOT NULL | Application ID |
| room\_applications | student\_id | INT | 8 | FK (users.id), NOT NULL | Student ID |
| room\_applications | room\_number | VARCHAR | 10 | FK (rooms.room\_number), NOT NULL | Room number |
| room\_applications | application\_date | TIMESTAMP | - | DEFAULT CURRENT\_TIMESTAMP | Application date |
| room\_applications | status | ENUM | ('pending','approved','denied') | DEFAULT 'pending' | Application status |
| room\_applications | admin\_remarks | TEXT | - | - | Admin remarks |
| room\_applications | decision\_date | TIMESTAMP | - | - | Decision date |

# **7.0 Testing**

We used unit testing to check each PHP functions of the system such as login validation. There is also integration testing to ensure that front-end work well with back-end. System testing has verified full workflows like login, room allocation and task assign.

This part will be to test the system for the webpage to check the functionality of the webpage to determine whether it is passed or failed for student webpage, admin webpage and staff webpage. ( means pass, means failed)

## 7.1 Admin Test

|  |  |  |  |
| --- | --- | --- | --- |
| **Admin** | | | |
| No | Test Case (Page) | Action | Status |
| 1 | Login | Login with correct credentials and login successfully |  |
| 2 | Login with incorrect credentials and unable to login |  |
| 3 | Can log out successfully |  |
| 4 | Dashboard | Can change the theme correctly |  |
| 5 | Room Management | Can create room successfully |  |
| 6 | Can allocate room successfully |  |
| 7 | Can deallocate room successfully |  |
| 8 | Fee Management | Can create fee successfully |  |
| 9 | Can view history payment and approve payment successfully |  |
| 10 | Cleaning Schedule | Can create schedule successfully |  |
| 11 | Maintenance | Can assign task to staff successfully |  |
| 12 | Complaints | Can update status successfully |  |

## 7.2 Student Test

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | | | |
| No | Test Case (Page) | Action | Status |
| 1 | Login | Login with correct credentials and login successfully |  |
| 2 | Login with incorrect credentials and unable to login |  |
| 3 | Can log out successfully |  |
| 4 | Dashboard | Can change the theme correctly |  |
| 5 | My Room | Can apply room successfully |  |
| 6 | Can view room detail successfully |  |
| 8 | Pay Fees | Can make payment successfully |  |
| 11 | Maintenance | Can submit maintenance request successfully |  |
| 12 | Complaints | Can submit complaint request successfully |  |

## 7.3 Staff Test

|  |  |  |  |
| --- | --- | --- | --- |
| **Staff** | | | |
| No | Test Case (Page) | Action | Status |
| 1 | Login | Login with correct credentials and login successfully |  |
| 2 | Login with incorrect credentials and unable to login |  |
| 3 | Can log out successfully |  |
| 4 | Dashboard | Can change the theme correctly |  |
| 5 | My Cleaning Task | Can mark complete task successfully |  |
| 6 | My Maintenance Task | Can mark complete task successfully |  |

## 7.4 Admin Test Result

The data below shows result of the total case tested, total action take, total pass and total fail of the admin webpage.

**Total Case Tested: 7**

**Total Action Taken: 12**

**Total PASS: 12/12(100%)**

**Total FAIL: 0/12(0%)**

From the test results, we can that all the admin test is pass and there is no fail. This has proved that the system functions reliably from the admin’s perspective, allowing smooth operations such as logging in, allocate hostel and ask task. It confirms that the core functionalities designed for employees meet intended requirements.

## 7.5 Student Test Result

The data below shows result of the total case tested, total action take, total pass and total fail of the admin webpage.

**Total Case Tested: 6**

**Total Action Taken: 12**

**Total PASS: 12/12(100%)**

**Total FAIL: 0/12(0%)**

From the test results, we can see that all the student test is pass and there is nothing to fail. This has demonstrated that the student features such as apply room, make payment and submit requests. The successful outcomes confirm that the system is effective in supporting student’s operations with accuracy and efficiency.

## 7.6 Staff Test Result

**Total Case Tested: 4**

**Total Action Taken: 6**

**Total PASS: 6/6(100%)**

**Total FAIL: 0/6(0%)**

From the test results, we can see that all the staff test is pass and there is nothing to fail. This demonstrates that the staff features like mark completed task. The successful outcomes confirm that the system is effective to support staff operation with accuracy and efficiency.

# **8.0 Project Management**

Effective project management was necessary to ensure the successful completion of the Hostel Management System. Our strategy was carefully planned to achieve a balance between time management, planning, teamwork, and risk prediction. Every team member played a specific role in keeping the process efficient at all stages of development.

## 8.1 Project Planning

The table below shows the Gantt Chart of our project planning. The Gantt Chart provides a visual representation of the project timeline, outlining each phase and activity involved in the development process.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Week 5 | Week 6 | Week 7 | Week 8 | Week 9 | Week 10 | Week 11 | Week 12 |
| Introduction and Background Study |  |  |  |  |  |  |  |  |
| Requirements Engineering |  |  |  |  |  |  |  |  |
| Functional & Non-Functional Requirements |  |  |  |  |  |  |  |  |
| Requirements Modelling |  |  |  |  |  |  |  |  |
| Design Modelling |  |  |  |  |  |  |  |  |
| Testing |  |  |  |  |  |  |  |  |
| Project Management |  |  |  |  |  |  |  |  |
| Conclusion |  |  |  |  |  |  |  |  |

The Gantt Chart illustrates the timeline for completing each phase of the project:

* **Introduction and Background Study (Week 5):** This initial phase involves researching the project domain and understanding the problem context.
* **Requirements Engineering (Weeks 6-7):** At this phase, we identify and document all stakeholder needs and system requirements.
* **Functional & Non-Functional Requirements (Weeks 7-8):** Detailed specification of what the system should do and how it should perform.
* **Requirements Modelling (Weeks 7-9):** Creating visual models (use cases, activity diagrams) to represent system requirements.
* **Design Modelling (Weeks 8-10):** Developing the system architecture and detailed design specifications.
* **Testing (Weeks 9-12):** Implementation of test cases, execution of unit, integration, and system tests.
* **Project Management Conclusion (Weeks 11-12):** Finalizing documentation, preparing deliverables, and project review.

## 8.2 Teamwork

A key component of our success was teamwork. The four of us respected each other’s opinions and worked together. One member worked on front-end design and user interface, another on back-end PHP coding, and others helped with testing and documentation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Name  Task | Kee Shee Hau | Ong Qi Ren | Soo Teck Shen | Lim Chin Chen |
| 1 | Introduction and Background Study |  |  |  |  |
| 2 | Requirements Engineering |  |  |  |  |
| 3 | Functional & Non-Functional Requirements |  |  |  |  |
| 4 | Requirements Modelling |  |  |  |  |
| 5 | Design Modelling |  |  |  |  |
| 6 | Testing |  |  |  |  |
| 7 | Project Management |  |  |  |  |
| 8 | Conclusion |  |  |  |  |
| 9 | Student Webpage |  |  |  |  |
| 10 | Admin Webpage |  |  |  |  |
| 11 | Staff Webpage |  |  |  |  |

## 8.3 Risk Management

In the process of developing the hostel management system, we considered some risk factors that may affect the project development, such as technological errors, ill-planned schedules, or poor team communication, etc. To ensure the smooth development and timely completion of the system, we identified several risks and created solutions for each one.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Risk Description | Likelihood | Impact | Risk Level | Solution |
| 1 | Data loss due to file overwrites or corruption | Medium | High | High | Regular backups to cloud storage and version control via GitHub. |
| 2 | Missed deadlines from academic workload | High | Medium | High | Early planning, buffer time in schedule, weekly progress tracking. |
| 3 | Scope creep from new feature suggestions | Medium | High | High | Strict feature freeze after the design phase, backlog list for future updates. |
| 4 | Uneven task distribution | Medium | Medium | Medium | Task rotation and workload balancing during team check-ins. |
| 5 | Miscommunication or misunderstanding | Low | Medium | Medium | Clear task descriptions, group chats, and written summaries for discussions |

## 8.4 Time Management

Effective time management was essential to let the system complete before the deadline. Our team created a planned timetable that matched the Gantt Chart and gave each stage of development the proper amount of time.

|  |  |  |
| --- | --- | --- |
| Phase | Week(s) | Activities |
| Planning and Research | Week 5 | Project initiation, introduction, background study and system overview |
| Requirement Engineering | Week 6-7 | Elicitation, specification, validation, and documentation of requirements. |
| System Design | Week 7-8 | Do use case, diagram, class diagram and webpage design |
| Implementation | Week 8-10 | Developed frontend and backend using PHP and MySQL |
| Testing | Week 10-11 | Conducted unit testing, integration testing and system testing |
| Documentation and Finalization | Week 11-12 | Compile report, do project management task and prepare for presentation. |

# **9.0 Conclusion**

In conclusion, the Hostel Management System project was a meaningful experience that allowed us to apply technical knowledge to solve real-world problems. The discussed Hostel Management System project has shown that a computerized system will help simplify and magnify the daily administrative activities of hostel management. We were able to create centralized, user-friendly software for handling bookings, complaints, cleaning schedules, and payment tracking by identifying redundancies in manual hostel operations. In addition to meeting user demands, our system is scalable and maintainable for future development due to our teamwork and stakeholder participation. We gained important knowledge about communication, cooperation, and the significance of developing solutions with empathy and purpose during the process.

Throughout the project, we encountered several challenges. One of the biggest challenges was managing time, especially balancing academic responsibilities and project deadlines. Technical challenges also occurred while integrating other modules, such as complaint tracking and hotel booking, especially when it came to ensuring data consistency and real-time updates. Furthermore, continuous negotiation and discussion were necessary to bring disparate stakeholders' expectations into alignment.

The system has a lot of potential for improvement in the future. Students could more easily handle their concerns and objections if a mobile application was integrated. In addition, we hope to include analytics dashboards for administrators, automatic notifications, and maybe integration with other university services such as finance or student registration systems. The system is prepared to grow and change to meet the changing demands of the institution and its students due to the foundation we've established.