

HOSTEL MANAGEMENT SYSTEM (STUDENT & REGISTRAR)

SUBMITTED IN PARTIAL FULFILLMENT OF
REQUIREMENTS FOR THE AWARD OF THE DEGREE OF
MASTER OF SCIENCE IN COMPUTER SCIENCE
OF

UNIVERSITY OF CALICUT

SUBMITTED BY

Ms. MUMTHAZ SULTHANA U K

Reg. No: CUAWCMF013

UNDER THE GUIDANCE OF

Ms. HRIDYA E



**DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF CALICUT**

2024

CERTIFICATE

This is to certify that the project work entitled “HOSTEL MANAGEMENT SYSTEM (STUDENT & REGISTRAR)” is a bonafide record of original work done by MUMTHAZ SULTHANA U K (CUAWCMF013) final year M.Sc. Computer Science in partial fulfillment of the requirements for the award of the degree in Master of Computer Science during the period of 2022-2024.

Project Guide:

Ms. Hridya E

Head of the Department:

Dr. Lajish V L

Certified of the candidate MUMTHAZ SULTHANA U K (CUAWCMF013) is examined by us in the project viva voice.

Examination held on:

Signature of Examiners

1

2

Calicut university

Date:

DECLARATION

I hereby declare that the project entitled “**HOSTEL MANAGEMENT SYSTEM (OFFICE STUDENT & REGISTRAR)**” submitted to the **Department of Computer Science, University of Calicut** in partial fulfilment of the requirements for the award of degree in **MSc. COMPUTER SCIENCE** is a record of original dissertation work done by me, Under the guidance and supervision of **Ms. HRIDYA E Assistant professor, Department of Computer Science, University of Calicut, Thenhipalam** during my study period in **CALICUT UNIVERSITY CAMPUS, MALAPPURAM**.

Place:

MUMTHAZ SULTHANA U K

Date:

Reg.No:CUAWCMF013

ACKNOWLEDGEMENT

I express my sincere thanks to **Dr. Lajish V L**, Associate Professor & Head, Department of Computer Science, University of Calicut, for his valuable support and leadership throughout the course.

Presentation, inspiration, and motivation have always played key roles in the success of any venture. I am deeply grateful to my guide, **Ms. Hridya E**, Assistant Professor, Department of Computer Science, University of Calicut, for her valuable guidance and kind supervision, which significantly contributed to the shaping of this work. Additionally, I am immensely obliged to my friends for their inspiring, encouraging guidance, and kind supervision during the completion of this project.

I extend my deep sense of gratitude to **Dr. Manjula K A**, Associate Professor, **Mr. Ishaque K**, Assistant Professor, and **Dr. Aljinu Khadar K V**, Assistant Professor, Department of Computer Science, University of Calicut, for their valuable suggestions and guidance throughout the project, which helped in its successful development and completion.

I wish to thank **Mr. Abdul Nasar P**, CUCC, University of Calicut, the staffs of the Men's Hostel, Calicut University, and **Dr. Binu R**, Warden of the Men's Hostel, for their guidelines, help and support. Additionally, I am grateful to the peon of the Department of Computer Science for his assistance and sacrifices.

Lastly, I express my heartfelt gratitude to ChatGPT for providing continuous support and assistance throughout the project. Your guidance has been invaluable in shaping this work.

ABSTRACT

We are putting forward HOSTEL MANAGEMENT SYSTEM which is an integrated system to manage the hostel activities, which will help to reduce the chances for making mistakes by humans and thus eliminate the delay of time issues of tracking file records. The drawbacks of the existing manual system leading us to design a computerized, compatible system to ease the efforts of staffs and thus provide better services to all the hostel inmates. Our vision is to make the current manual practices to an automated system. The forwarded system has control over all the activities of a student inside the hostel from registration, room allocation, room vacating, payments, fee dues, etc.

We are suggesting this HOSTEL MANAGEMENT SYSTEM to manage various activities inside the Hostel. The number of educational institutions is increasing rapidly, and thereby the number of hostels is also increasing to accommodate these students of institutions. And the use of software in this context is very limited. And hence make lots of strain to the person who handles the records of the hostels. Our project deals with these issues and solving the problems which is carried out when it is handled manually. The major motive to design such a computerized system is the identification of problems of the existing manual systems. The suggested system is compatible with the existing system. And it is more user-friendly hence it is mostly a GUI oriented system. We can improve the efficiency of the existing system in a convenient manner.

INDEX

Chapter No	Contents	Page No
1	INTRODUCTION	1
2	SYSTEM STUDY	3
	2.1 EXISTING SYSTEM	5
	2.1.1 DRAWBACKS	5
	2.2 PROPOSED SYSTEM	6
	2.2.1 ADVANTAGES	6
	2.3 MODULE DESCRIPTION	7
3	SOFTWARE AND HARDWARE SPECIFICATION	10
	3.1 HARDWARE SPECIFICATION	11
	3.2 SOFTWARE SPECIFICATION	11
	3.3 LANGUAGE DESCRIPTION	12
4	SYSTEM ANALYSIS	15
	4.1 FEASIBILITY STUDY	17
	4.1.1 THECHNICAL FEASIBILITY	18
	4.1.2 OPERATIONAL FEASIBILITY	18
	4.1.3 ECONOMICAL FEASIBILITY	19
	4.1.4 BEHAVIOURAL FEASIBILITY	19
	4.1.5 SOFTWARE FEASIBILITY	19
	4.1.6 HARDWARE FEASIBILITY	19

5	SYSTEM DESIGN		20
	5.1	DATA FLOW DIAGRAM	22
	5.2	ENTITY RELATIONSHIP DIAGRAM	31
	5.3	DATABASE DESIGN	33
6	SYSTEM TESTING		43
	6.1	UNIT TESTING	45
	6.2	INTEGRATION TESTING	45
	6.3	VALIDATION TESTING	46
	6.4	INPUT TESTING	46
	6.5	OUTPUT TESTING	46
	6.6	USER ACCEPTANCE TESTING	46
7	FUTURE ENHANCEMENT		47
8	CONCLUSION		49
9	APPENDIX		51
	9.1	SCREENSHOTS	52
	9.2	SAMPLE CODE	60
10	BIBLIOGRAPHY		80

INTRODUCTION

INTRODUCTION

In the modern digital world all the records are computerized and automated. And it is not advisable to use the file records. Even though our hostels are not yet using the technologies to digitalize the Hostel activities. Registration form verification, fee payment, room allocation, room vacating, etc. are done manually. It is not an ideal way to keep records in file documents since it needs more time and effort. And also the file records are not secure. And there is a lots of chances for repetition of data. These drawbacks of the existing system the motive of us to design a fully computerized system for hostel management which help the staffs to save records of the students and keep track of student data, students can see fee pending, room data, register complaints, apply for vacating room all through the system. No need to see the staffs in person.

Objectives

The major objective of the suggested system is to integrate all the activities of a hostel under a common platform. The admins can keep track of all the activities and transactions of a student. Know the fee pending of the student. Student can also see their pending, register complaints, give feedbacks, apply for room allocation and room vacating. The department HOD can also keep track of their student activities.

- Efficiently completes the admission of student only after the approval from department HOD.
- Room allocation can be more easy .
- Admin add student details after allocation and keeps the details after deallocation for a certain time.
- Admins can keep track of students fee payments and pending.
- Admins can add and edit notice board, departments, courses, view complaints etc.
- Warden / Office staff can allocate and deallocate students.
- Student can add complaint or feedback and also can view reply of the complaint.
- Admins can view complaints and feedbacks provided by the students.

SYSTEM STUDY

SYSTEM STUDY

System study is the first step of a project development life cycle. It gives the clear picture of how the system will look like. The system study is consisting of two phases. The first phase is the preliminary survey and the second phase is the detailed study of the existing system. In the preliminary survey we identify the scope of the system. In the second phase of the system study is more detailed study of the existing system which will helps to identify the user requirements and the limitations and problems of the existing system. After completing the system study the developers will make a proposal for suggested system. In this project we first took survey on scope of the project, this project will be helpful to students, hostel admins, department HOD's. Thus the system possess a huge scope. After studying the existing system we made conclusion that it is very convenient when the hostel management is computerized. This will make easy the records keeping, fee tracking, student tracking, etc. Student can also keep track of their records like fees and rent pending and payments. Register complaints and feedbacks, etc.

EXISTING SYSTEM

The existing system is fully manual and needs more human power and time to keep records of all students. And it is very difficult to track the students details and payments history, needs to check many records a]to find different payment details of a student. This cause waste of time and efforts. And it may also make corruption in the allocation process, fee calculation. The system does not deal with student complaints and feedbacks.

Drawbacks:

- File records/ Paper records needs to be filled manually.
- Need more time and effort.
- Redundancy may occur in data.
- Provides less security for data.
- Same procedures need to be repeated for all students.
- Paper records are difficult to handle.
- Data updating is very difficult when managed manually.
- Keeping the records is very difficult.
- Data cannot be stored for backup and retrieve.

PROPOSED SYSTEM

The proposed system is aimed to overcome the drawbacks of the existing system. It requires less human labor when compared to existing system and it is efficient to keep data. This project is aimed to integrate all the activities related to the hostel. This system will make the hostel officer to manage all the activities inside the hostel. Through this system the hostel officer can get full information about a student in the hostel. It will show rooms available or not and number of people in a particular room. This will provide the details of students who paid their fees and whose are still pending. This system will also provide the facility to students to register complaints and feedbacks, and these complaints can be viewed from the admin side and can take action according to it. The proposed system deals with the allotment process efficiently.

Advantages:

- Provides user friendly GUI.
- No redundant data.
- Easy to manage and update all records.
- Using centralized database.
- Provide more security to data.
- Data can be saved for backup.
- Efficient and time saving.

MODULE DESCRIPTION

Our project HOSTEL MANAGEMENT SYSTEM is an integrated system which is aimed to integrate all the activities of a hostel and bring them under a single platform. It will help to reduce the need of human labor and thus save more time and labor and eliminate the humanly mistakes and delay of time in file records. The drawbacks of the existing system are the major motive of us to design a computerized system to manage the records and activities of the hostels. The system will be compatible to provide better services to the staffs, HOD's and students. The major goal of the proposed system is to eliminate the manual records from offices thus make the record keeping easy, less time consuming, less space consuming and also efficient to keep track of large data. It is provided with overall control on the hostel management like student registration, room allocation, fee and payments, complaints and feedbacks, room vacating, etc.

The system is consisting these modules.

- Admin Module
- Hod Module
- Student Module
- Warden Module
- Office Module
- Registrar Module

ADMIN MODULE

The ADMIN is the super user who manage the master data of the system like server details. ADMIN can also manage other Admin's details. Provide them access and username and passwords. Can control all the functions like student details, hostel details, departments and HOD's, Wardens and office staffs. Can control the fee and payment sections. Generate and view admission fee and other fees

HOD MODULE

The HOD is an admin side user. HOD verifies the details entered by the students and approves for who needs hostel for accommodation. HOD can reject fake applications. The request will be carried to the Hostel warden only after approved by the department HOD. And the HOD can track the student's payment history and fee pending. When a student make application for TC, HOD can check whether the student possess any dues in the hostel.

STUDENT MODULE

The STUDENT is a client-side user. STUDENT need to register to the system to access system functions. When a student register to the system a request is send to the HOD to approve the student for hostel admission. When the HOD approve the request, the STUDENT can pay the admission fee. Then the request is passed to WARDEN with transaction ID. WARDEN can approve the request according to the availability of rooms. Then the student can apply for room allocation. Then the student will be allocated with room and bed. The student needs to pay for rent and water and electricity charges monthly. The STUDENT can register complaint or give feedback about the hostel and system. The student can apply for room vacating only if there are no fee dues left.

WARDEN MODULE

The WARDEN is an admin-side user. The WARDEN is the supreme authority in the hostel. No student can take admission or vacate their room without the permission of the warden. The request is forwarded to the warden with payment details. Then the warden approve the student to take admission. Then the WARDEN can monitor all the activities of the students like payments, dues, complaints, feedbacks. When a student apply for room vacating the warden checks the track of student. And then allow the STUDENT to vacate.

OFFICE MODULE

The OFFICE is an admin side user. It is intended to used by the staffs of the Hostel OFFICE. They are coming under the WARDEN. WARDEN have the authority to supervise the Hostel, while the record keeping, fee payments, verifications are done by Hostel staffs. So they possess a major role in Hostel management. After WARDEN approve a student admission, the STUDENT can apply room allocation, room is allotted from the hostel OFFICE, according to the availability of rooms. And they can check feedbacks, complaints take necessary action. They can check fees and payments. Update rules and Notices. They can give personal notifications to students if needed.

REGISTRAR MODULE

The REGISTRAR is an admin side user. Registrar comes on top of WARDEN in the hierarchy. REGISTRAR can manage WARDEN's. And other admins like office staffs and HOD's. REGISTRAR can also view student details, Hostel details like blocks, rooms, beds, number of vacant rooms, etc. He can view and take action on complaints and check feedbacks. REGISTRAR can produce notices and update the rules if needed.

SOFTWARE AND HARDWARE SPECIFICATION

HARDWARE SPECIFICATION

System	: Intel
Processor	: Core i3
Ram Capacity	: 4GB
Hard Disk Drive	: 256 GB
Keyboard	: Standard
Mouse	: Standard

SOFTWARE SPECIFICATION

Platform	: Windows OS, Linux
Front End	: HTML, CSS, BLADE Template Engine, PHP, JavaScript
Back End	: PHP, PGSQL
Framework	: Laravel
Browser	: Google Chrome, Mozilla Firefox, Edge

LANGUAGE DESCRIPTION

HTML

HTML stands for Hyper Text Markup Language. It is used to design web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. Markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most of markup languages are human readable.

HTML is a markup language which is used by the browser to manipulate text, images and other content to display it in required format. HTML was created by Tim Berners-Lee in 1991. The first ever version of HTML was HTML 1.0. HTML uses predefined tags and elements which tells the browser about content display property. If a tag is not closed then browser applies that effect till end of page.

CSS (Cascading Style Sheet)

A CSS (cascading style sheet) file allows you to separate your web sites HTML content from its style. As always you use your HTML file to arrange the content, but all of the presentation (fonts, colors, background, borders, text formatting, link effects & so on...) are accomplished within a CSS. At this point you have some choices of how to use the CSS, either internally or externally. First, we will explore the internal method. This way you are simply placing the CSS code within the tags of each HTML file you want to style with the CSS. The format for this is shown in the example below. Inline styles are defined right in the HTML file alongside the element you want to style. An external CSS file can be created with any text or HTML editor. A CSS file contains no (X)HTML, only CSS. You simply save it with the .css file extension.

Blade Template Engine

Blade is the simple, yet powerful templating engine that is included with Laravel. Unlike some PHP templating engines, Blade does not restrict you from using plain PHP code in your templates. In fact, all Blade templates are compiled into plain PHP code and cached until they are modified, meaning Blade adds essentially zero overhead to your application. Blade template files use the `.blade.php` file extension and are typically stored in the `resources/views` directory. Blade views may be returned from routes or controllers using the global `view` helper.

PHP

The term PHP is an acronym for Hypertext PreProcessor. PHP is a server-side scripting language designed specifically for web development. It is open-source which means it is free to download and use. It is very simple to learn and use. The file extension of PHP is `.php`. PHP was introduced by **Rasmus Lerdorf** in the first version and participated in the later versions. It is an interpreted language and it does not require a compiler. PHP is primarily used for server-side web development. It enables the creation of dynamic web pages by embedding PHP code within HTML.

JavaScript

JavaScript is a very powerful client-side scripting language used mainly for enhancing the interaction of a user with the webpage. JavaScript is also being used widely in game development and mobile application. Being a scripting language, JavaScript cannot run on its own. In fact, the browser is responsible for running JavaScript code. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it. The main advantage of JavaScript is that all modern web browsers support JavaScript. So, you do not have to worry about whether your site visitor uses Internet Explorer, Google Chrome, Firefox or any other browser. JavaScript will be supported. Also, JavaScript runs on any operating system including Windows, Linux or Mac.

PGSQL (POSTGRESQL)

PostgreSQL also known as Postgres, was developed by Michael Stonebraker of the University of California, Berkley. It started as the **Ingres Project** and later evolved into Postgresql as we know today. In the year 1982, Michael Stonebraker started a **post-Ingres project** to address the problems with contemporary database systems. He was awarded the Turing Award in the year 2014 for the projects and techniques pioneered in them. The POSTGRES project aimed at adding fewest features like the ability to define various data types and to fully describe relationships – something used widely, but maintained completely by the end-user. POSTGRES used various ideas of Ingres, but had its unique source code. The initial version of PostgreSQL was designed to run on UNIX-like platforms. However, it was then evolved to be mobile so that it could run on other platforms such as Mac OS X, Solaris, and Windows.

SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is concerned with analyzing, designing, implementing and evaluating information system in our organization. It is carried out to make the system more effective either by modification or by substantial redesign. In system analysis we identify the problem, study the alternative solution and select the most suitable solution, which meet the technical economic and social demands for analysis, various tools such as dataflow diagram, interviews on site observation, questionnaires etc., are used. System analysis process is also called a life cycle methodology since it relates to four significant phases in life cycle of all information system. They are

1. System Analysis / Study Phase.
2. System Design / Design phase.
3. System Development / Development Phase.
4. Testing and implementation / Operation Phase.

All activities associated with each life cycle phase must be performed managed and documented. So, system analysis is the performance, management and documentation of the activities related to the four life cycle phases of a computer-based system

.

FEASIBILITY STUDY

The most difficult part of feasibility analysis is the identification of the Candidate system and the evaluation of their performance. Feasibility study is a test of a system proposal according to its workability, impact on the organization, ability to meet the user needs and effective use of resources.

A feasibility study is conducted to select the best system that meets performance requirements. The entails an identification description, an evaluation of candidate system, and the selection of the best system for the job. The new system has advantages such as we can easily doing transactions in the shop and this application is more user friendly for the employees. Six key considerations are involved in the feasibility analysis:

1. Technical Feasibility
2. Operational Feasibility
3. Economical Feasibility
4. Behavioral Feasibility
5. Software Feasibility
6. Hardware Feasibility

TECHNICAL FEASIBILITY

A study of function, performance and constraints may improve the ability to create an acceptable system. Technical Feasibility is frequently the most difficult area to achieve at the stage of product engineering process. Considering that are normally associated with the technical feasibility include,

- Development risk
- Resource availability
- Technology

Technical Feasibility study deals with the hardware as well as software requirements. The scope was whether the work for the project is done with the current requirements and existing software technology has to be examined in the feasibility study.

The outcome was found to be positive. In the proposed system, data can be easily stored and managed using database management system software. The reports and results for various queries can be generated easily. Therefore, the system is technically feasible.

OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned into information system that will meet the organization's operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are these major barriers to implementation?

The purpose of the operational feasibility study is to determine whether the new system will be used if it is developed and implemented from users that will undermine the possible application benefits. There was no difficulty in, implementing the system and the proposed system is so effective, user friendly and functionally reliable so that the users in the company will find that the new system reduce their hard steps. If the user of the system is fully aware of the internal working of the system then the users will not be facing any problem in running the system.

ECONOMICAL FEASIBILITY

Proposed system was developed with the available resources. Since cost input for the software is almost nil the output of the software is always a profit. Hence software is economically feasible. In the existing system, manpower is more required. In the proposed system the effort to be involved is reduced drastically. So, the proposed system is said to be economic.

BEHAVIORAL FEASIBILITY

People are inherently resistant to changes and computer is known as facilitating the changes. An estimate should be made of how strongly the users react to the development of the system. The proposed system consumes time. Thus the people are made to engage in some other important work.

SOFTWARE FEASIBILITY

Even though software is developed in very high software environment, it will be supported by many other and environments with minimum changes.

HARDWARE FEASIBILITY

The software can be developed with resource already existing. Here the consideration is that the existing hardware resources support the technologies that are to be used by the new system. No hardware was newly bought for the project and hence software is said to achieve hardware feasibility. The software can be developed with resource already existing. Here the consideration is that the existing hardware resources support the technologies that are to be used by the new system. No hardware was newly bought for the project and hence software is said to achieve hardware feasibility.

SYSTEM DESIGN

SYSTEM DESIGN

System Design involves translating system requirements and conceptual design into technical specification and general flow of processing. After the system requirements have been identified, information has been gathered to verify the problems and after evaluating the existing system a new system is proposed. System Design is the process of planning of new system or to replace or complement an existing system. It must be thoroughly understood about the old system determine how computers can be used to make its operations more effective.

System Design sits at technical the kernel of the system development. Once system requirements have been analyzed and specified system design is the first of the technical activities – design, code generation and test that required to build and verify the software. System Design is the most creative and challenging phases of the system life cycle. The term design describes the final system and the process by which it is to be developed.

System Design is the high-level strategy for solving the problem and building a solution. System Design includes decisions about the organization of the system into subsystems, the allocation of subsystems to hardware and software components and major conceptual and policy decision that forms the framework for detailed design.

DATA FLOW DIAGRAM

A Data Flow Diagram is used to define the flow of data and the resources such as information. Data Flow Diagrams are a way of expressing system requirements in graphical manner. It has the purpose of clarifying system requirements and identifying the major transformation that will become program in the system design. So it is the starting point of design phase that functionally decomposes the requirement specification down into the lowest level of details. The bubbles represent data transformation and the lines represent information flow in the system. Data Flow Diagrams are useful in understanding a system and can be effectively used for partitioning. The system may be an organization, a manual procedure, software system, a mechanical system or any combination of these.

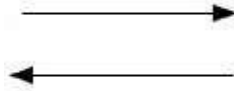
Rules For Constructing a Data Flow Diagram

Process should be named and numbered for easy reference. Each name should be representative of process. The direction of flow is from top to bottom and from left to right. That is information flow should be from source to destination. Numbering is given when a process is exploded into lower-level details. The name of the data stores, source and destination are written in capital letters. Process and Data Flow names have the first letter of each word capitalized. The Data Flow Diagram is particularly designed to aid communication. If it contains dozens of process and data stores it gets too unwieldy. The rule of the thumb is to explode the DFD into a functional level beyond that, it is best to take each function separately and expand it to show the explosion in a single process. If a user wants to know what happens within a given process, then the detailed explosion of that process may be shown.

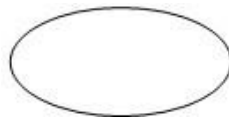
The goal of DFD is to have a commonly understood model of a system. The diagram is the basis of structured system analysis. DFD are supported by other techniques of structured system analysis. DFD are supported by other techniques of structured system analysis such as structured diagrams, and data dictionaries.

DFD SYMBOLS

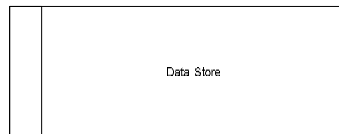
Data Flow Diagrams are composed of the four basis symbols shown below



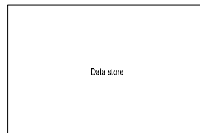
A data is a root, which enables packet of data to travel from one point to another. Data may flow from a source to a processor and from data source or process. An arrow line depicts the flow, with arrowhead pointing in the direction of flow.



A process represents transformation where incoming data changed into outgoing data flows.

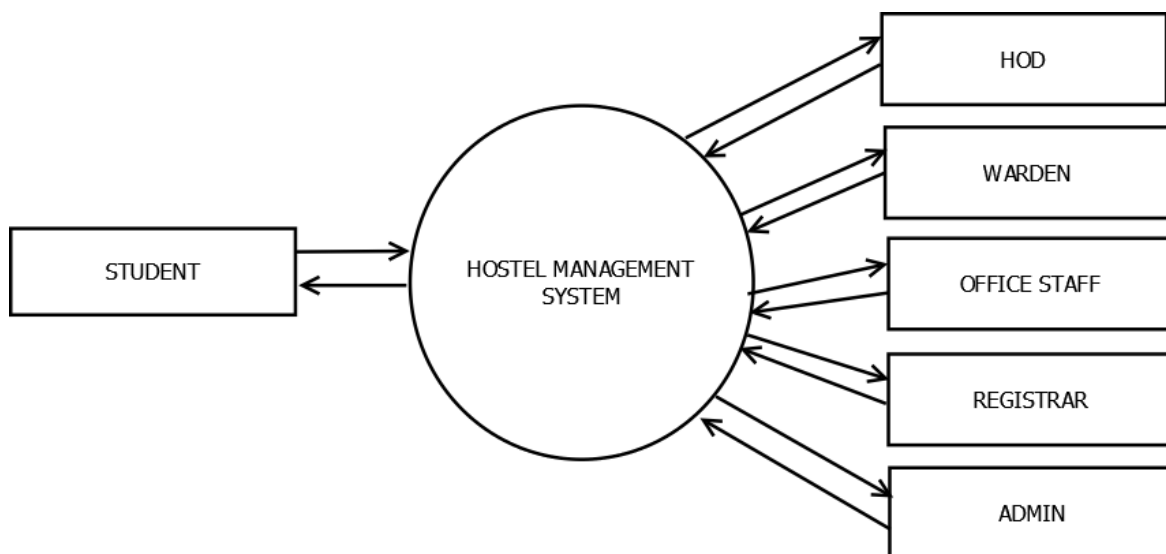


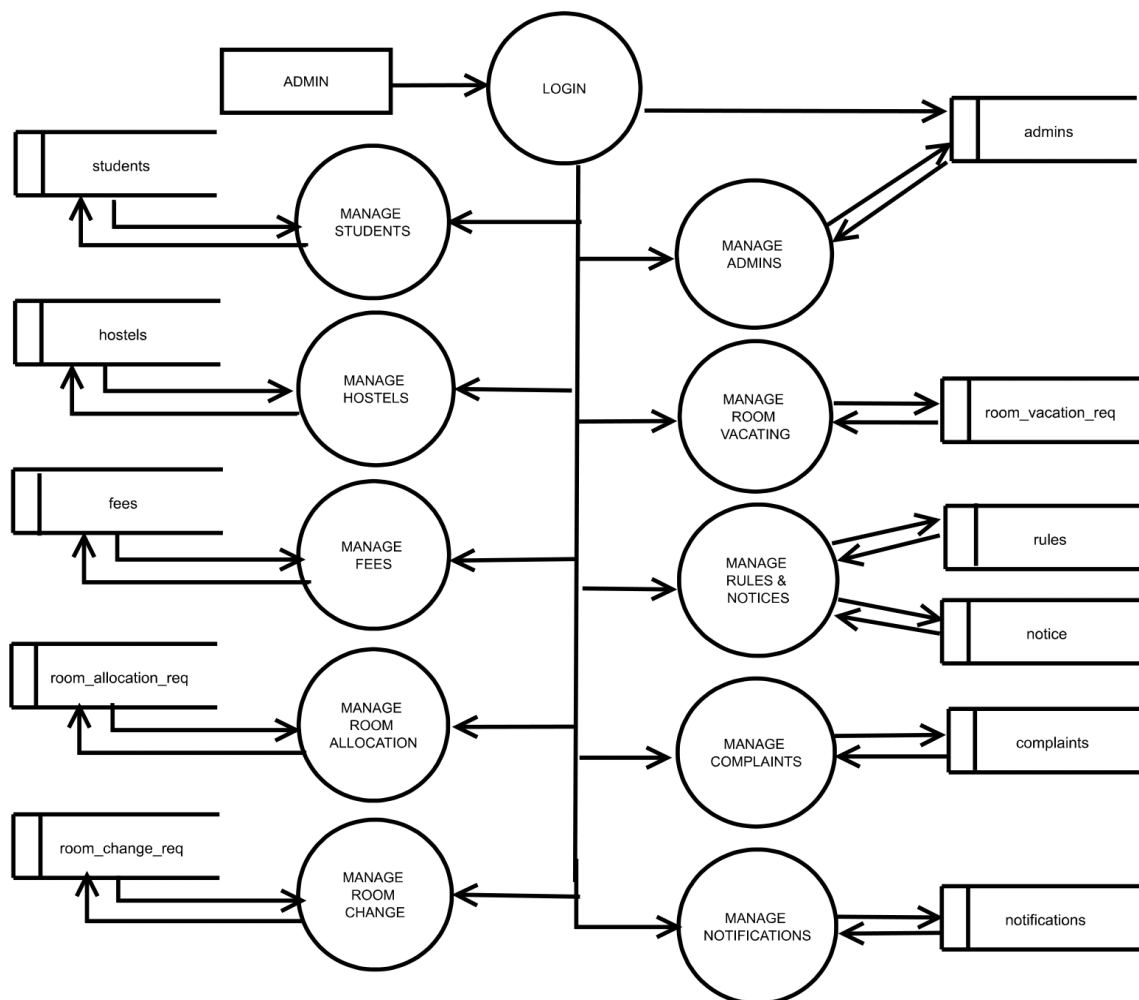
An open-ended box represents a data store, data at rest or a temporary repository of data.

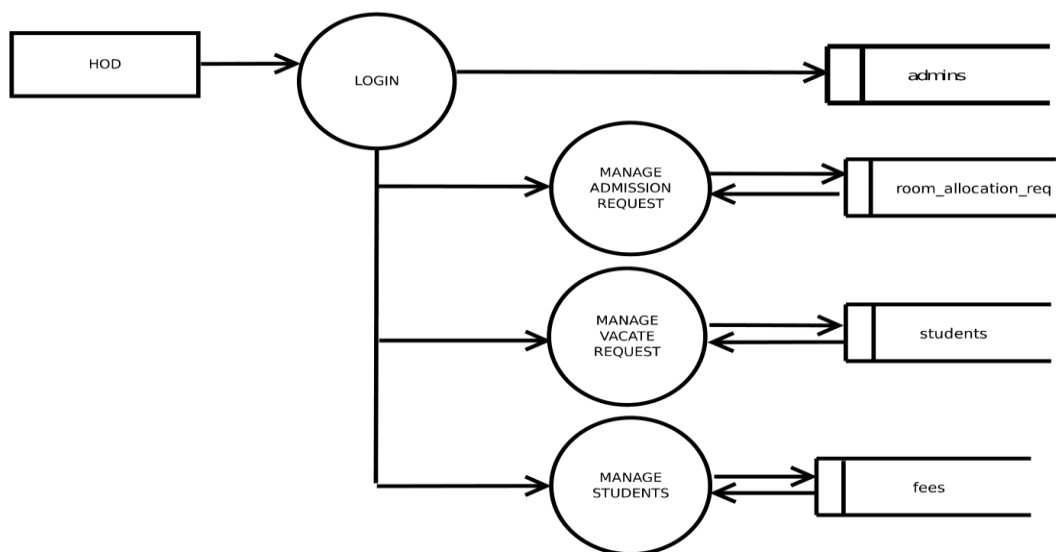


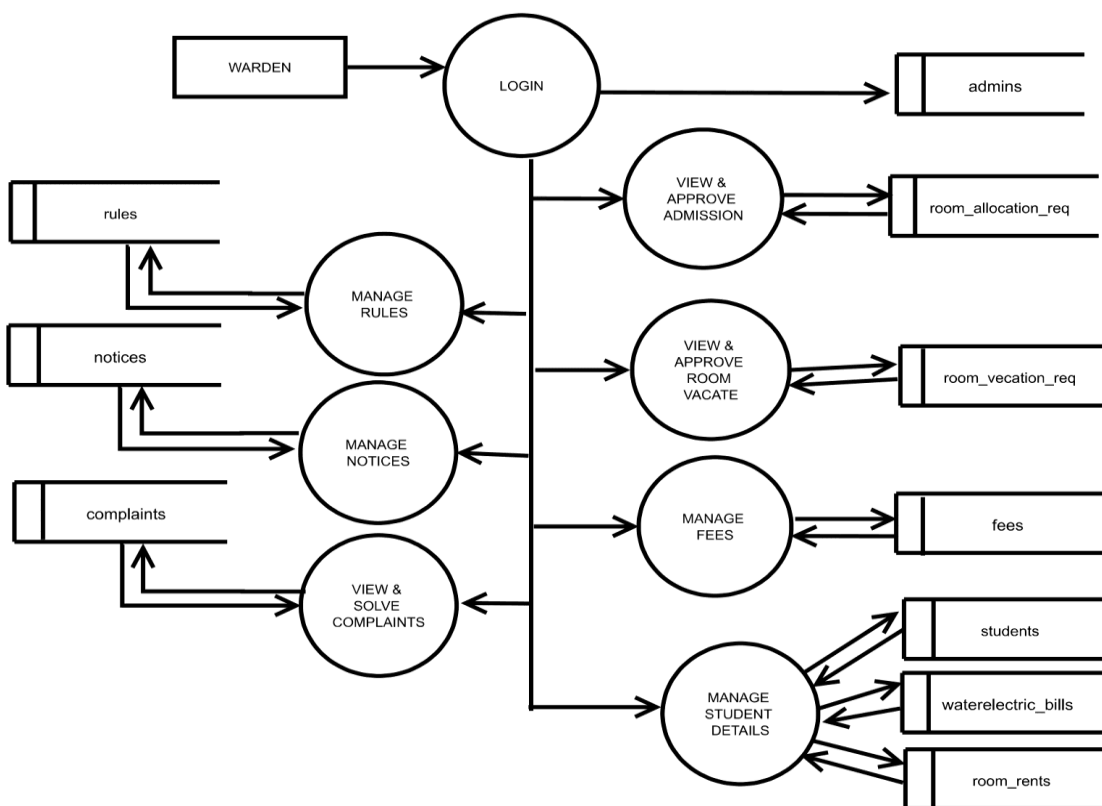
A square defines a source or destination of system data.

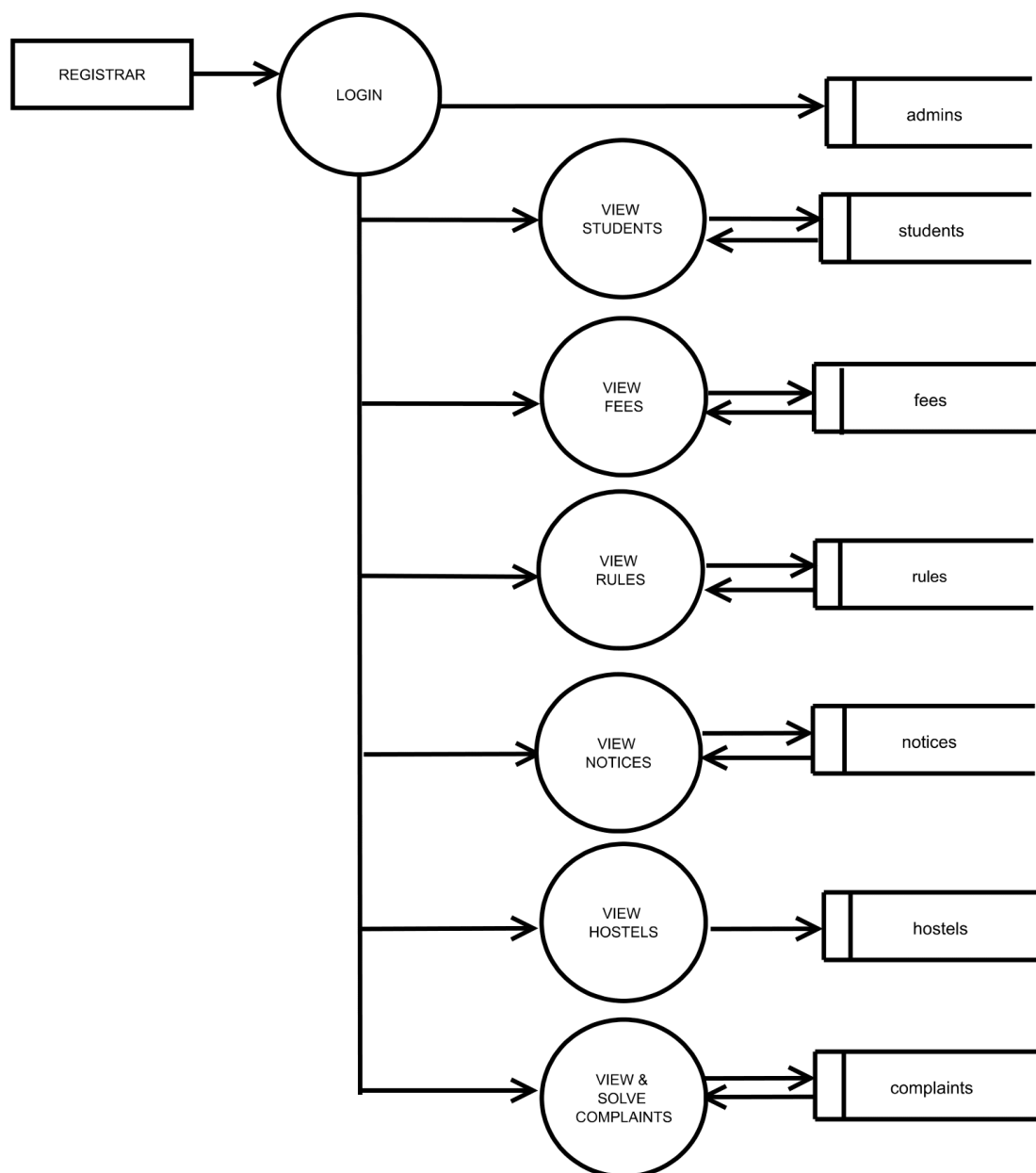
LEVEL 0

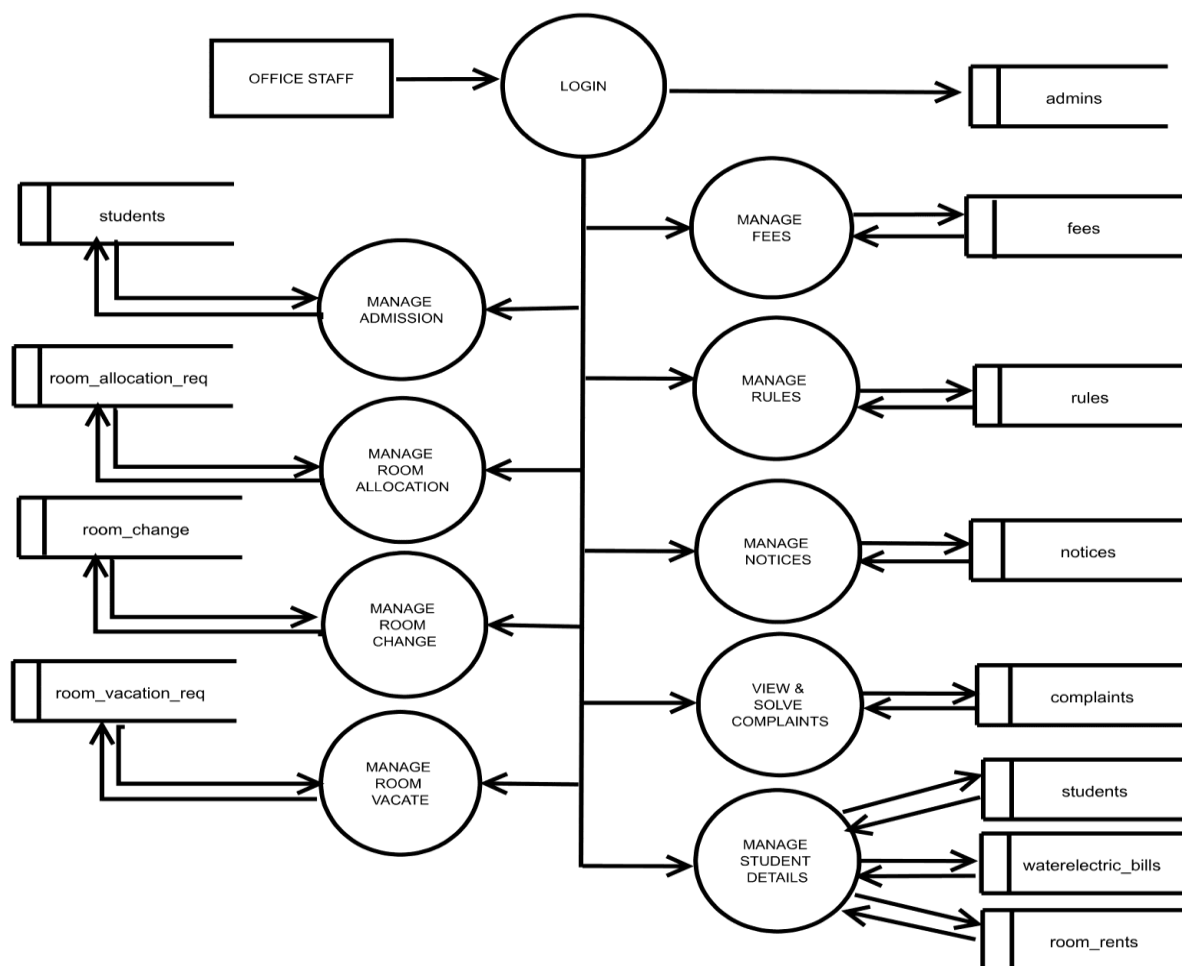


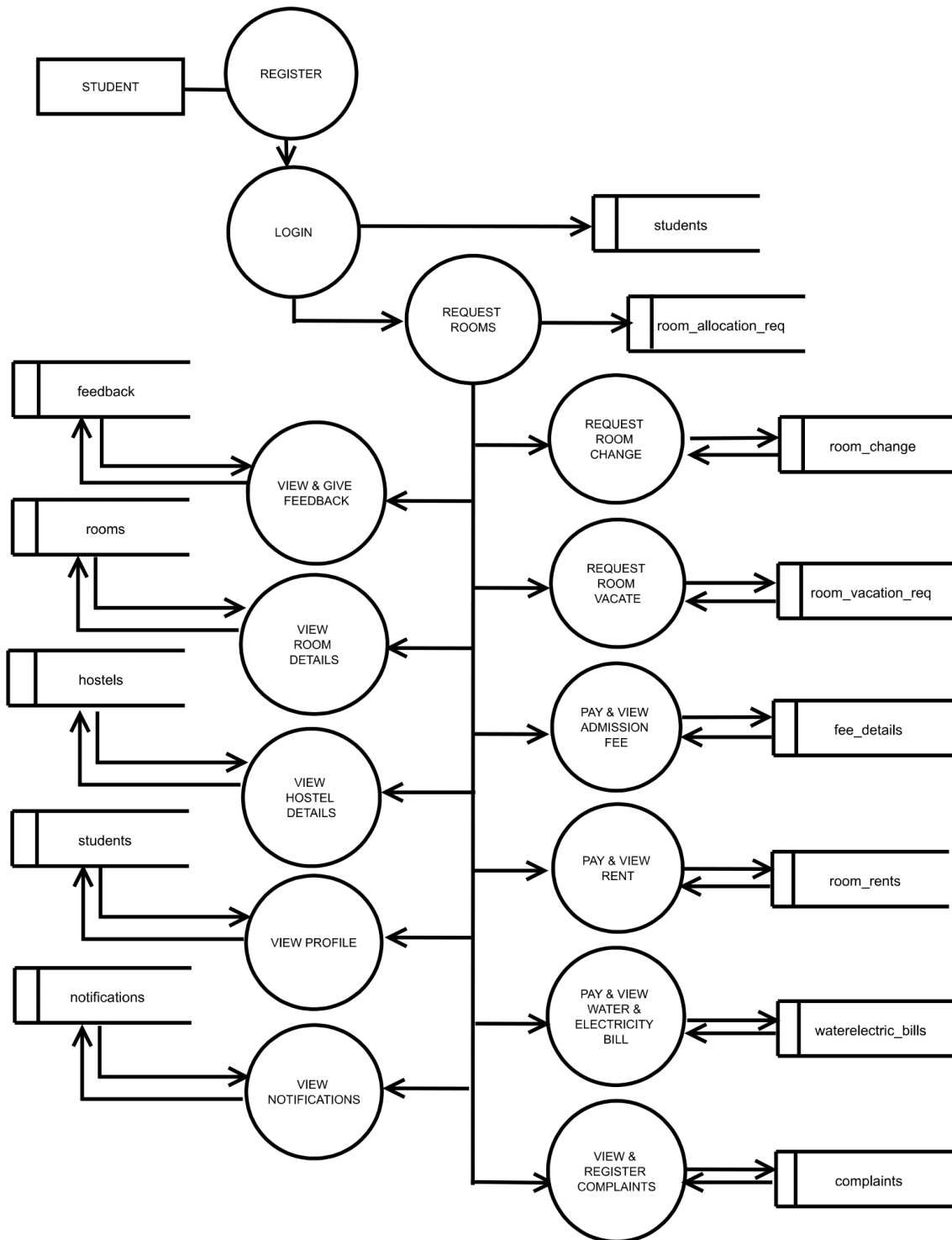
LEVEL 1: ADMIN

LEVEL 1: HOD

LEVEL 1 : WARDEN

LEVEL 1 : REGISTRAR

LEVEL 1 : OFFICE STAFF

LEVEL 1 : STUDENT

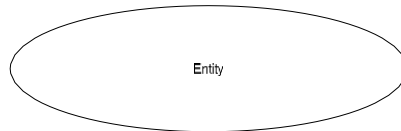
ENTITY RELATIONSHIP DIAGRAM

An E R diagram is a model that identifies the concept or entities that exist in a system and the relationship between those entities. An ERD is often used as a way to visualize a relational database each entity represents a database table and the relationship lines represents the key in one table that point to specific records in related tables.

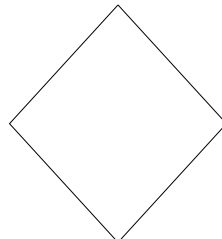
➤ Entity :-



➤ Attribute :-



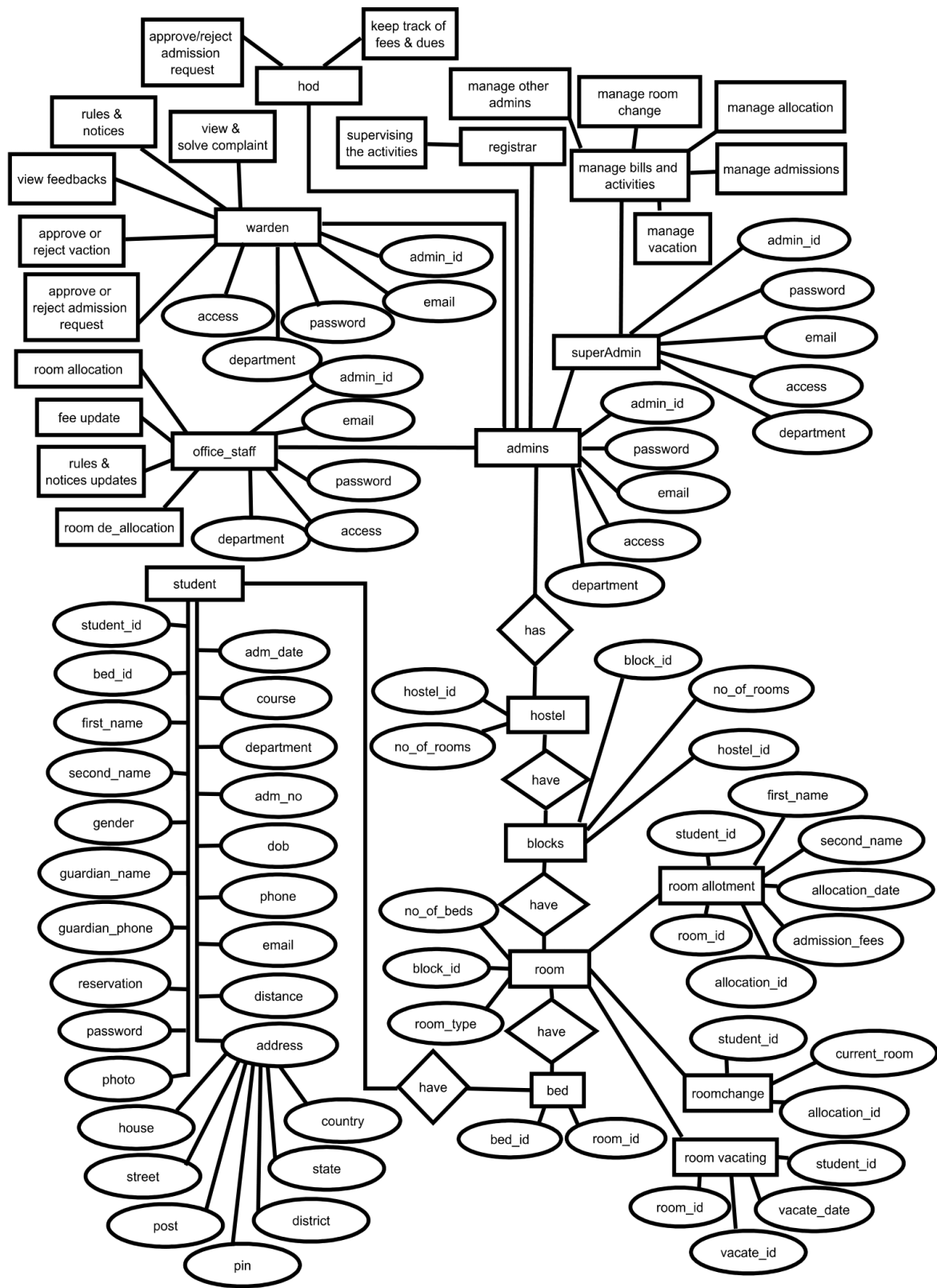
➤ Relationship :-



It depicts the fundamental relations like recording personal information, paying salary and getting a loan. ER involved the student information and payroll system.

Advantage of ER diagram

- Professional and faster development.
- Productivity improved.
- Fewer faults in development.
- Maintenance becomes easy.



DATABASE DESIGN

The data base design is a logical development in the methods used by the computer to access and manipulate data stored in the various parts of the computer system. Database is defined as an integrated collection of data. The overall objective in the development of database technology has been to treat data as organization recourses and as an integrated whole. The main objectives of database are data integration, data integrity and data independence.

PGSQL (POSTGRESQL)

PostgreSQL also known as Postgres, was developed by Michael Stonebraker of the University of California, Berkley. It started as the **Ingres Project** and later evolved into Postgresql as we know today. In the year 1982, Michael Stonebraker started a **post-Ingres project** to address the problems with contemporary database systems. He was awarded the Turing Award in the year 2014 for the projects and techniques pioneered in them. The POSTGRES project aimed at adding fewest features like the ability to define various data types and to fully describe relationships – something used widely, but maintained completely by the end-user. POSTGRES used various ideas of Ingres, but had its unique source code. The initial version of PostgreSQL was designed to run on UNIX-like platforms. However, it was then evolved to be mobile so that it could run on other platforms such as Mac OS X, Solaris, and Windows.

Normalization

Normalization is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency. Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. Database normalization is a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data.

1 st Normal Form (1NF)

In this Normal Form, we tackle the problem of atomicity. Here atomicity means values in the table should not be further divided. In simple terms, a single cell cannot hold multiple values. If a table contains a composite or multi-valued attribute, it violates the First Normal Form.

2 nd Normal Form (2NF)

The first condition in the 2nd NF is that the table has to be in 1st NF. The table also should not contain partial dependency. Here partial dependency means the proper subset of candidate key determines a non-prime attribute.

3 rd Normal Form (3NF)

The same rule applies as before i.e., the table has to be in 2NF before proceeding to 3NF. The other condition is there should be no transitive dependency for non-prime attributes. That means non-prime attributes (which doesn't form a candidate key) should not be dependent on other non-prime attributes in a given table. So a transitive dependency is a functional dependency in which $X \rightarrow Z$ (X determines Z) indirectly, by virtue of $X \rightarrow Y$ and $Y \rightarrow Z$ (where it is not the case that $Y \rightarrow X$)

TABLE DESIGN

1. Table Name: admins

Sl.No	Field	Data type	Constraints
1	Admin_id	Bigint	Primary Key
2	Name	Varchar	
3	Department	Varchar	
4	Designation	Varchar	
5	Access	Varchar	
6	Email	Varchar	
7	Password	Password	
8	Remember_token	Varchar	
9	Created_at	Timestamp	
10	Updated_at	Timestamp	

2. Table Name: beds

Sl.No	Field	Data type	Constraints
1	Bed_id	Bigint	Primary Key
2	Room_id	Bigint	Foreign Key
3	Bed_name	Varchar	
4	Bed_type	Bigint	
5	Status	Varchar	
6	Student_id	Bigint	Foreign key
7	Created_at	Timestamp	
8	Updated_at	Timestamp	

3. Table Name: Blocks

Sl.No	Field	Data type	Constraints
1	Block_id	Bigint	Primary Key
2	Block_name	Varchar	
3	Hostel_id	Bigint	Foreign key
4	No_of_rooms	Varchar	
5	Created_at	Timestamp	
6	Updated_at	Timestamp	

4. Table Name: Complaints

Sl.No	Field	Data type	Constraints
1	Complaint_id	Bigint	Primary Key
2	Student_id	Bigint	Foreign key
3	Category	Varchar	
4	Complaint	Varchar	
5	Status	Varchar	
6	Closedby	Bigint	Foreign key
7	Comment	Varchar	
8	Created_at	Timestamp	
9	Updated_at	Timestamp	

5. Table Name: Courses

Sl.No	Field	Data type	Constraints
1	Course_id	Bigint	Primary Key
2	Course_name	Varchar	
3	Department_id	Bigint	Foreign Key
4	Course type	Varchar	
5	Course_duration	Varchar	
6	Created_at	Timestamp	
7	Updated_at	Timestamp	

6. Table Name : Departments

Sl.No	Field	Data type	Constraints
1	Department_id	Bigint	Primmary key
2	Category	Varchar	
3	Hod	Bigint	Foreign Key
4	Section_Officer	Varchar	
5	Contact_no	Varchar	
6	Created_at	Timestamp	
7	Updated_at	Timestamp	

7. Table Name : Fee Details

Sl.No	Field	Data type	Constraints
1	Fee_id	Bigint	Primary Key
2	Fee_title	Varchar	
3	Amount	Double	
4	Created_at	Timestamp	
5	Updated_at	Timestamp	

8. Table Name: Feedback

Sl.No	Field	Data type	Constraints
1	Feedback_id	Bigint	Primary Key
2	Student_id	Bigint	Foreign Key
3	Review	Varchar	
4	Created_at	Timestamp	
5	Updated_at	Timestamp	

9. Table Name: Fees

Sl.No	Field	Data type	Constraints
1	Fee_id	Bigint	Primary Key
2	Hostel_id	Bigint	Foreign Key
3	Room_type	Varchar	
4	Fee_name	Varchar	
5	Amount	Varchar	
6	Updatedby	Bigint	Foreign Key
7	Created_at	Timestamp	
8	Updated_at	Timestamp	

10. Table Name: Hostels

Sl.No	Field	Data type	Constraints
1	Hostel_id	Bigint	Primary key
2	Hostel_name	Varchar	
3	No_of_blocks	Varchar	
4	Created_at	Timestamp	
5	Updated_at	Timestamp	

11. Table Name: Notice

Sl.No	Field	Data type	Constraints
1	Notice_id	Bigint	Primary key
2	Title	Varchar	
3	Publishedby	Bigint	Foreign key
4	Path	Varchar	
5	Created_at	Timestamp	
6	Updated_at	Timestamp	

12. Table Name: Notifications

Sl.No	Field	Data type	Constraints
1	Notice_id	Bigint	Primary key
2	Title	Varchar	
3	Publishedby	Bigint	Foreign key
4	Path	Varchar	
5	Created_at	Timestamp	
6	Updated_at	Timestamp	

13. Table Name: Room rents

Sl.No	Field	Data type	Constraints
1	Room_rent_id	Bigint	Primary key
2	Fee_id	Bigint	Foreign key
3	Student_id	Bigint	Foreign key
4	Month_of_fee	Varchar	
5	Paid_status	Varchar	
6	Payment_date	Date	
7	Transaction_id	Varchar	
8	Created_at	Timestamp	
9	Updated_at	Timestamp	

14. Table Name: Room allocation req

Sl.No	Field	Data type	Constraints
1	Allocate_req_id	Bigint	Primary key
2	Student_id	Bigint	Foreign key
3	Department_id	Bigint	Foreign key
4	Dep_verification_status	Varchar	
5	Payment_status	Varchar	
6	Allocation_status	Varchar	
7	Transaction_id	Bigint	Foreign Key
8	Warden_verification_status	Varchar	
9	Allocatedby	Varchar	
10	Hostel	Varchar	
11	Created_at	Timestamp	
12	Updated_at	Timestamp	

15. Table Name: Room vacation req

Sl.No	Field	Data type	Constraints
1	Vacate_req_id	Bigint	Primary key
2	Student_id	Bigint	Foreign key
3	Department_id	Bigint	Foreign key
4	Payment_status	Varchar	
5	Vacate_status	Varchar	
6	Office_status	Varchar	
7	Warden_status	Varchar	
8	Hod_status	Varchar	
9	Created_at	Timestamp	
10	Updated_at	Timestamp	

16. Table Name: Rooms

Sl.No	Field	Data type	Constraints
1	Room_id	Bigint	Primary key
2	Block_id	Bigint	Foreign key
3	Room_name	Varchar	
4	Room_type	Varchar	
5	No_of_beds	Varchar	
6	Created_at	Timestamp	
7	Updated_at	Timestamp	

17. Table Name: Room change

Sl.No	Field	Data type	Constraints
1	Room_change_id	Bigint	Primary key
2	Student_id	Bigint	Foreign key
3	Current_room	Bigint	Foreign key
4	Request	Varchar	
5	Reason	Varchar	
6	Status	Varchar	
7	Updatedby	Bigint	Foreign key
8	Created_at	Timestamp	
9	Updated_at	Timestamp	

18. Table Name: Rules

Sl.No	Field	Data type	Constraints
1	Rule_id	Bigint	Primary key
2	Title	Varchar	
3	Description	Varchar	
4	Updatedby	Bigint	Foreign key
5	Created_at	Timestamp	
6	Updated_at	Timestamp	

19. Table Name: Transactions

Sl.No	Field	Data type	Constraints
1	Transaction_id	Bigint	Primary key
2	Student_id	Bigint	Foreign key
3	Purpose	Varchar	
4	Amount	Double precision	
5	Status	Varchar	
6	Created_at	Timestamp	
7	Updated_at	Timestamp	

20. Table Name: Water-electric bills

Sl.No	Field	Data type	Constraints
1	Waterelectric_bills_id	Bigint	Primary key
2	Fee_id	Bigint	Foreign key
3	Student_id	Bigint	Foreign key
4	Month_of_fee	Double precision	
5	Paid-status	Varchar	
6	Payment_date	Date	
7	Transaction_id	Bigint	Foreign key
8	Created_at	Timestamp	
9	Updated_at	Timestamp	

21. Table Name: Students

Sl.No	Field	Data type	Constraints
1	Student_id	Bigint	Primary key
2	First_name	Bigint	Foreign key
3	Second_name	Bigint	Foreign key
4	Gender	Double precision	
5	Department	Varchar	
6	Adm_no	Varchar	
7	Dob	date	
8	Phone	Varchar	
9	E-mail	Varchar	
10	E-mail_verified_at	Timestamp	
11	Password	Varchar	
12	Distance	Varchar	
13	House	Varchar	
14	Street	Varchar	
15	Post	Varchar	
16	District	Varchar	
17	State	Varchar	
18	Pin	Varchar	
19	Country	Varchar	
20	Guardian	Varchar	
21	Guardian_phone	Varchar	
22	pwd	Varchar	
23	Course	Bigint	Foreign key

24	Adm_date	Date	
25	Reservation	Varchar	
26	Image	Varchar	
27	Remember_token	Varchar	
28	Bed_id	Bigint	Foreign key
29	Created_at	Timestamp	
30	Updated_at	Timestamp	

SYSTEM TESTING

SYSTEM TESTING

Testing is the major quality measure employed during software development. After the coding phase, computer programs are available that can be executed for testing purposes. Testing not only has to uncover errors introduced during coding but also locates errors committed during the previous phases. Thus, the aim of testing is to uncover requirements, design or coding in the program. System testing is an expensive but critical process that can take as much as fifty percent of the budget for program development. Consequential, different levels of testing are employed. In-fact a successful test is one that finds an error. The system performance criteria deal with turnaround time backup, file protection and human factor. A test for the user acceptance should be carried out. The package developed was taken through different levels of testing and required modifications were made. Testing is a vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The following points show how testing is essential.

- Existence of program defects or inadequacies is inferred
- Verifies whether the software behaves as intended by its designer.
- Checks conformance with requirements specification/user needs.
- Assesses the operational reliability of the system.
- Test the performance of the system.
- Reflects the frequencies of actual user inputs.
- Find the fault which caused the output anomaly.
- Detect flaws and deficiencies in the requirements.
- Exercise the program using data like real data processed by the program.
- Test the system capabilities.

TYPES OF TESTING

System testing is the state of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is vital to the success of system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to variety of tests.

- Unit Testing
- Integration Testing
- Validation Testing
- Input Testing
- Output Testing
- User Acceptance Testing

Unit Testing

Unit testing focuses on the verification effort on the smallest unit of software design the software component module. Using the component level design as a guide, important control paths are tested to uncover the error within the boundary of the module. The relative complexity of tests and uncovered error is limited by the constrained scope established for unit testing. Each module was tested individually and the errors are corrected.

Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit test components and build a program structure that has been dictated by design. Each module after unit testing were integrated and tested and errors were fixed.

Validation Testing

Here the inputs are given by the user are validated. This is the password validation, format of date are correct, textbox validation. Changes that need to be done after result of this testing. While verification is quality control process, quality assurance process carried out before the software is ready for release is known as validation testing. Its goal is to validate and be confident about the software product or system, that fulfills the requirements given by the customer. The two major areas when it should take place are in the early stages of software development and towards the end, when the product is ready for release. In other words, it is acceptance testing which is a part of validation testing.

Input Testing

Here system is tested with all verifiable combination of inputs. User may type data in situations like entering password, numerical details etc. The system is tested with all the cases and it responded error messages.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output generator or displayed by the system under consideration is tested by asking the user about the format required by them. The output format on the screen is found to be correct as the format was designed in the system design phase according to the user needs. As far as hardcopies are considered it goes in term with the user requirement. Hence output testing does not result any correction in the system.

User Acceptance Testing

User acceptance testing is done in presence of user. User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly in touch with the prospective system users at time of developing and making changes wherever is done in regard to the following points:

- Input screen design
- Output screen design
- Menu driven system

FUTURE ENHANCEMENT

FUTURE ENHANCEMENT

- Real time implementation of the proposed system for hostel management. It will enhance better communication between the students and responsible authority. Thus make an ideal environment inside the hostel. And the human labor can be reduced by using this system.
- Integrate the mess with the Hostel Management System, It will help the mess admins to calculate the bill, publish student mess bills, payments and due checking in an easier way.
- Adding more hostels like international hostel, student home under the same HOSTEL MANAGEMENT SYSTEM. Thus it makes easy to handle all the hostels under a single system.
- Linking the HOSTEL MANAGEMENT SYSTEM with the STUDENT PORTAL using the CAPID . Thus we can integrate all the activities of the student under the University.

CONCLUSION

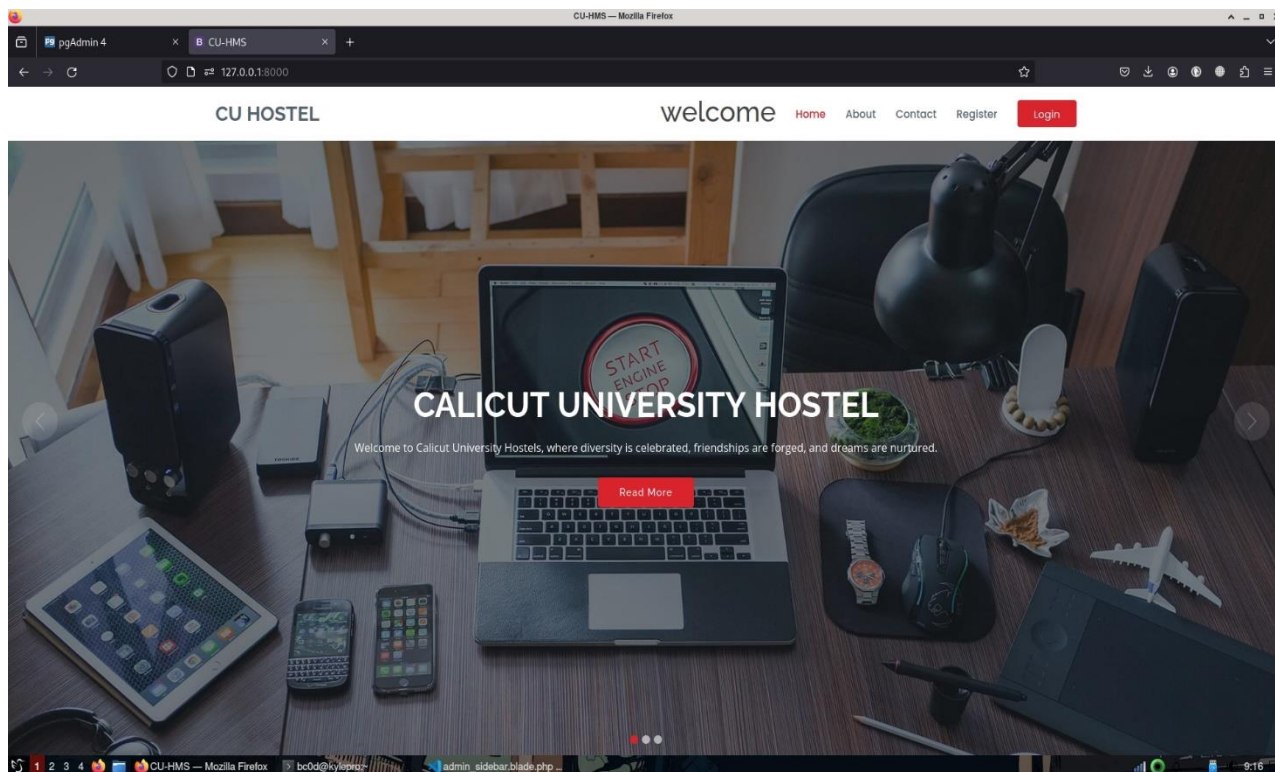
CONCLUSION

To conclude the description about the project, I would like to share some things about the project. This project is fully developed in Laravel framework. Laravel is a framework of the server-side scripting language PHP. This project is based on the user's requirement specification and the analytical study of the existing system,

HOSTEL MANAGEMENT SYSTEM is very useful for hostel admission, room allotment, fee calculation, complaint and feedback management, room deallocation, room vacation, etc. This HOSTEL MANAGEMENT SYSTEM is proposed to handle various activities of a hostel. This particular project deals with various challenges in hostel management and trying to overcome most of the problems and suggests a more convenient, accurate and efficient system to manage and integrate various activities of a hostel. Identification of the drawbacks of the existing system leads to the designing of computerized system that is mostly GUI oriented. Thus the system is more convenient and user friendly .

APPENDIX

SCREENSHOTS



The screenshot shows the "Sign up" form in the application. The browser window is titled "Sign up - Mozilla Firefox" and the address bar shows "127.0.0.1:8000/user-signup". The form is titled "Sign up" and contains the following fields:

- First Name *
- Second Name *
- Gender *
- Department *
- Admission Number *
- Date Of Birth *
- Phone Number *
- Email *
- Password *
- Confirm Password *

The form also includes a "Clear" button and a "Register" button. Below the form, the copyright notice reads: "© Copyright ARM. All Rights Reserved. Developed by ARM". The browser's taskbar at the bottom shows several open windows, including "pgAdmin 4", "Sign up - Mozilla Firefox", and "admin_sidebar.blade.php".

CU HOSTEL

Request room

Department Verification Status:
Approved

Payment Status:
Pending

Fee Name : Hostel Admission Fee Fee Name : Hostel Admission Caution Deposit

Fee Amount : 120 Fee Amount : 2000

Purpose:

Total Fee:

Warden Verification Status:
Pending

Allocation Status:
Pending

CU HOSTEL

University of Calicut

Useful Links
> Home

Login

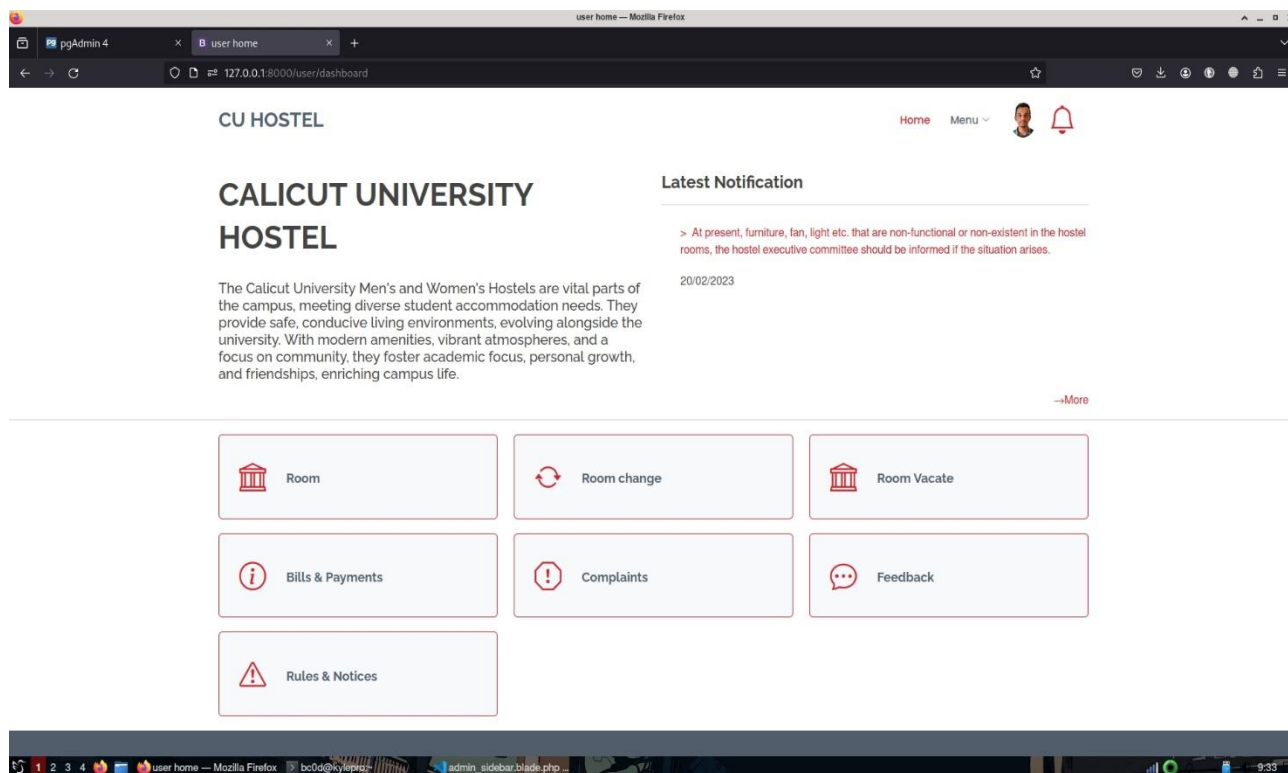
email

Password

New user? [Register](#)

[Staff Login](#)

© Copyright ARM. All Rights Reserved
Developed by ARM



The screenshot shows the 'CU HOSTEL' dashboard in a Mozilla Firefox browser. The page has a header with 'CU HOSTEL' and navigation links 'Home' and 'Menu'. A notification bell icon is visible. The main content area features a large heading 'CALICUT UNIVERSITY HOSTEL' and a paragraph describing the university's hostels. To the right, a 'Latest Notification' section contains a message about non-functional furniture and light in hostel rooms, dated 20/02/2023. Below this, a grid of seven buttons provides access to various functions: Room, Room change, Room Vacate, Bills & Payments, Complaints, Feedback, and Rules & Notices. A 'More' link is also present.

CU HOSTEL

Home Menu

CALICUT UNIVERSITY HOSTEL

The Calicut University Men's and Women's Hostels are vital parts of the campus, meeting diverse student accommodation needs. They provide safe, conducive living environments, evolving alongside the university. With modern amenities, vibrant atmospheres, and a focus on community, they foster academic focus, personal growth, and friendships, enriching campus life.

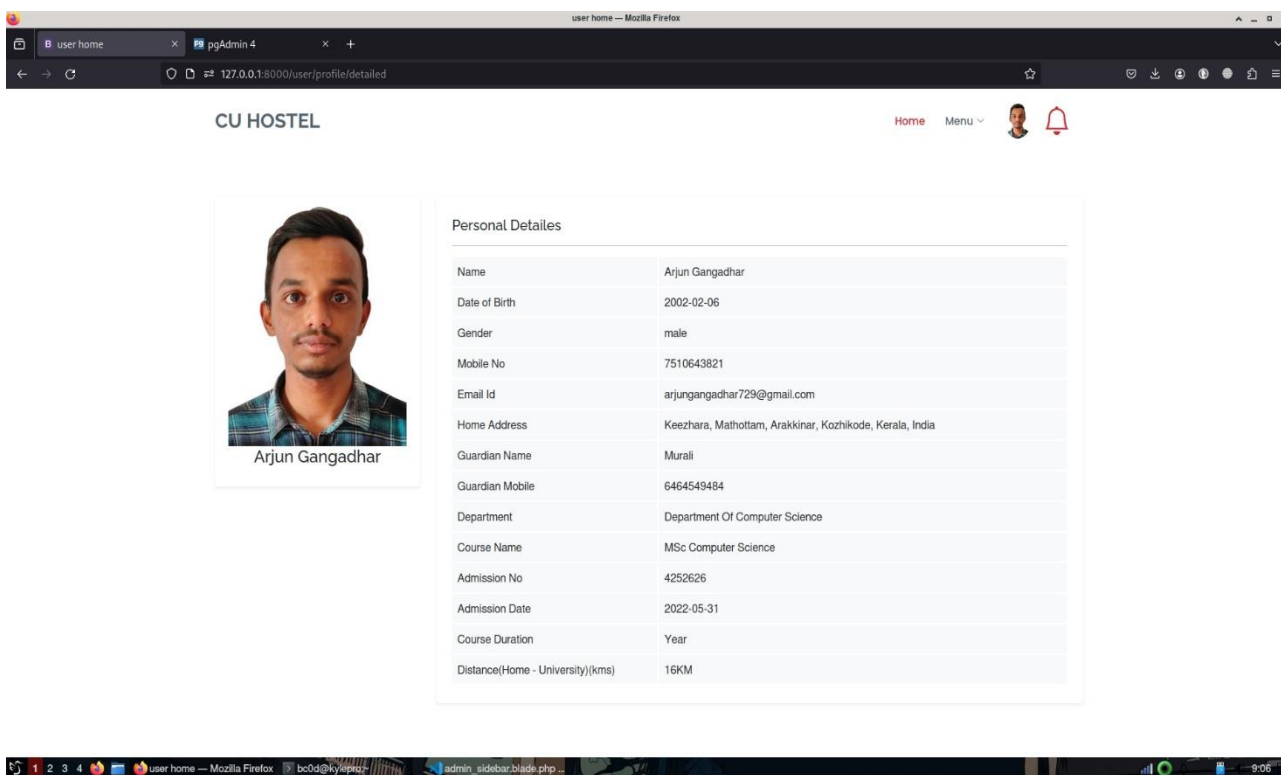
Latest Notification

> At present, furniture, fan, light etc. that are non-functional or non-existent in the hostel rooms, the hostel executive committee should be informed if the situation arises.

20/02/2023

[-->More](#)

- Room
- Room change
- Room Vacate
- Bills & Payments
- Complaints
- Feedback
- Rules & Notices



The screenshot shows the 'CU HOSTEL' user profile page. It features a profile picture of Arjun Gangadhar and a table of 'Personal Details'. The browser address bar shows the URL '127.0.0.1:8000/user/profile/details'. The page includes navigation links 'Home' and 'Menu', and a notification bell icon.

CU HOSTEL

Home Menu

Arjun Gangadhar

Personal Details

Name	Arjun Gangadhar
Date of Birth	2002-02-06
Gender	male
Mobile No	7510643821
Email Id	arjungangadhar729@gmail.com
Home Address	Keezhara, Mathottam, Arakkinar, Kozhikode, Kerala, India
Guardian Name	Murali
Guardian Mobile	6464549484
Department	Department Of Computer Science
Course Name	MSc Computer Science
Admission No	4252626
Admission Date	2022-05-31
Course Duration	Year
Distance(Home - University)(kms)	16KM

The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/user/room". The page title is "CU HOSTEL". The navigation bar includes "Home", "Menu", and a user profile icon. The main heading is "Room Details". Below it, a table displays room information:

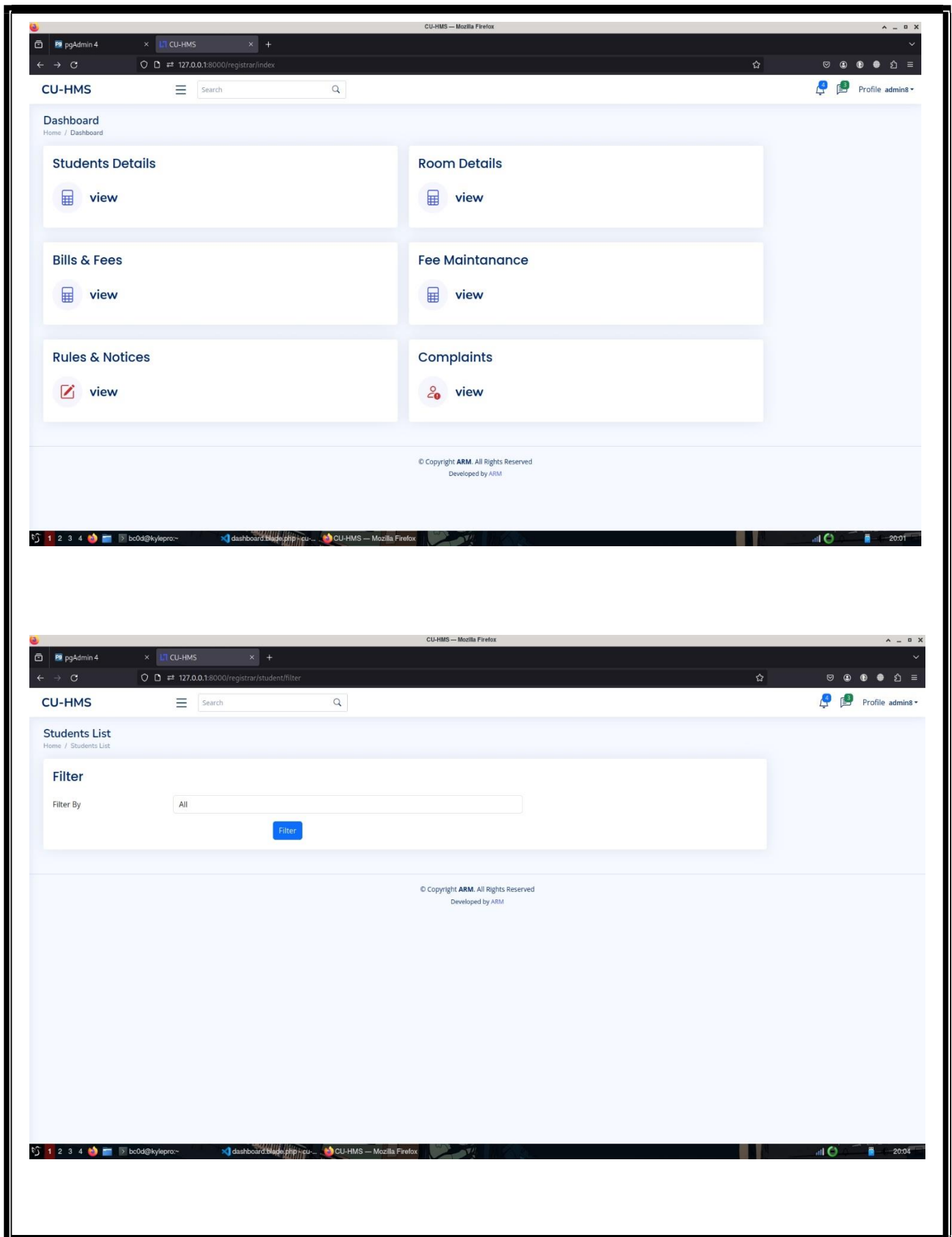
Hostel	Mens Hostel
Block	Old block
Room	MH1-3
Bed	MH1-3-1

Below the table, there are two input fields: "Room Rent" and "Water and Electric Bill". At the bottom, a dark blue footer contains the "CU HOSTEL" logo, contact information (University of Calicut, Thenchipalam PO, Malappuram, 673635, Kerala, India), and a "Useful Links" section with links to Home, About us, Contact, and Terms of service.

The screenshot shows a web browser window with the address bar displaying "127.0.0.1:8000/user/feedback". The page title is "CU HOSTEL". The navigation bar includes "Home", "Menu", and a user profile icon. The main heading is "Feedbacks". Below it, there are three feedback entries, each with a user profile icon, name, and timestamp:

- abhi, 1 day ago, good room
- abhi, 1 day ago, good atmosphere
- abhi, 1 day ago, nice hostel

At the top right of the feedback section, there is a link labeled "give feedback". At the bottom, a dark blue footer contains the "CU HOSTEL" logo, contact information, and a "Useful Links" section.



CU-HMS

Students List

Home / Students List

Filter

Filter By: Mens Hostel

☒ Old block
☒ New block
☐ Annex
☐ New Annex
☐ Phd block

Filter

Name	Age	Hostel	Block	Room	Bed
abhi rajan	1997-08-22	Mens Hostel	New block	MH2-2	MH2-2-1
Arjun Gangadhar	2002-02-06	Mens Hostel	Old block	MH1-3	MH1-3-1

© Copyright ARM. All Rights Reserved
Developed by ARM

CU-HMS

Registrar

Home / Rooms List

Rooms

Hostel: --Select--

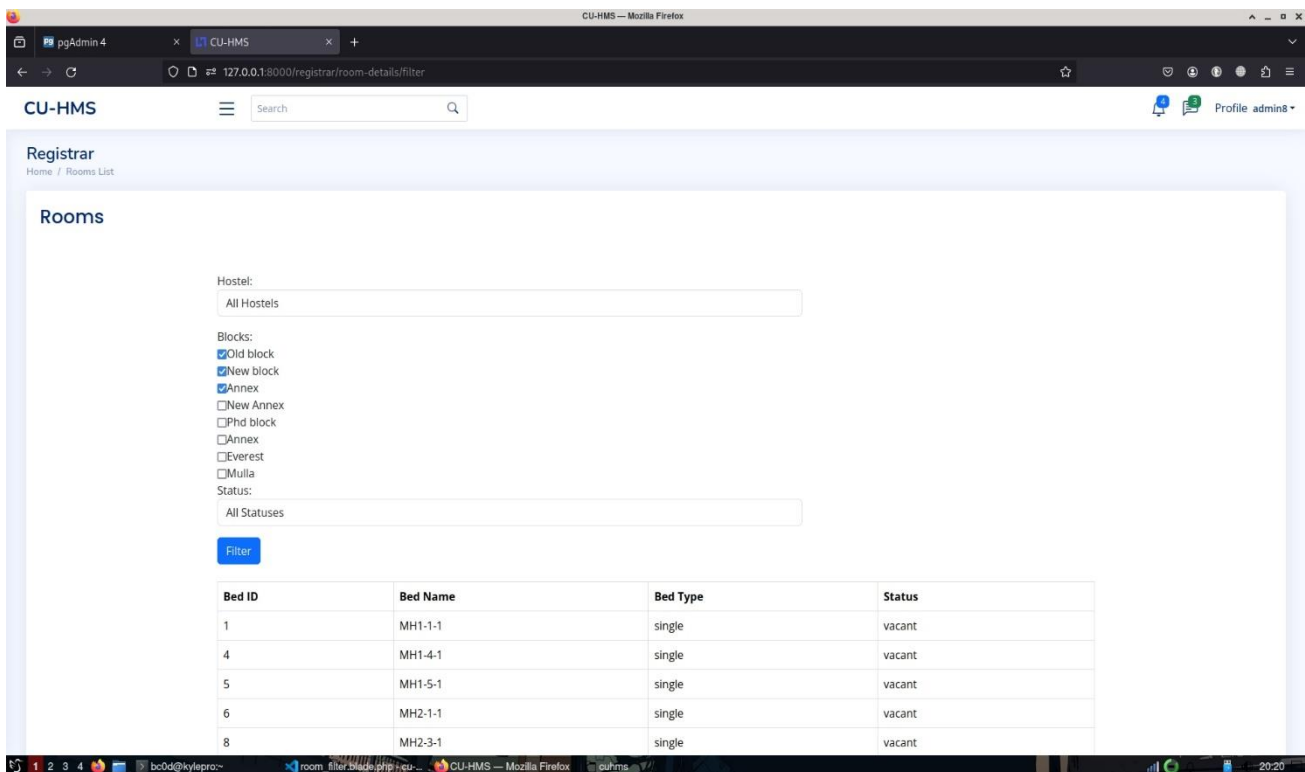
Blocks:

Status: All Statuses

Filter

Bed ID	Bed Name	Bed Type	Status
--------	----------	----------	--------

© Copyright ARM. All Rights Reserved
Developed by ARM



CU-HMS Registrar

Home / Rooms List

Rooms

Hostel:

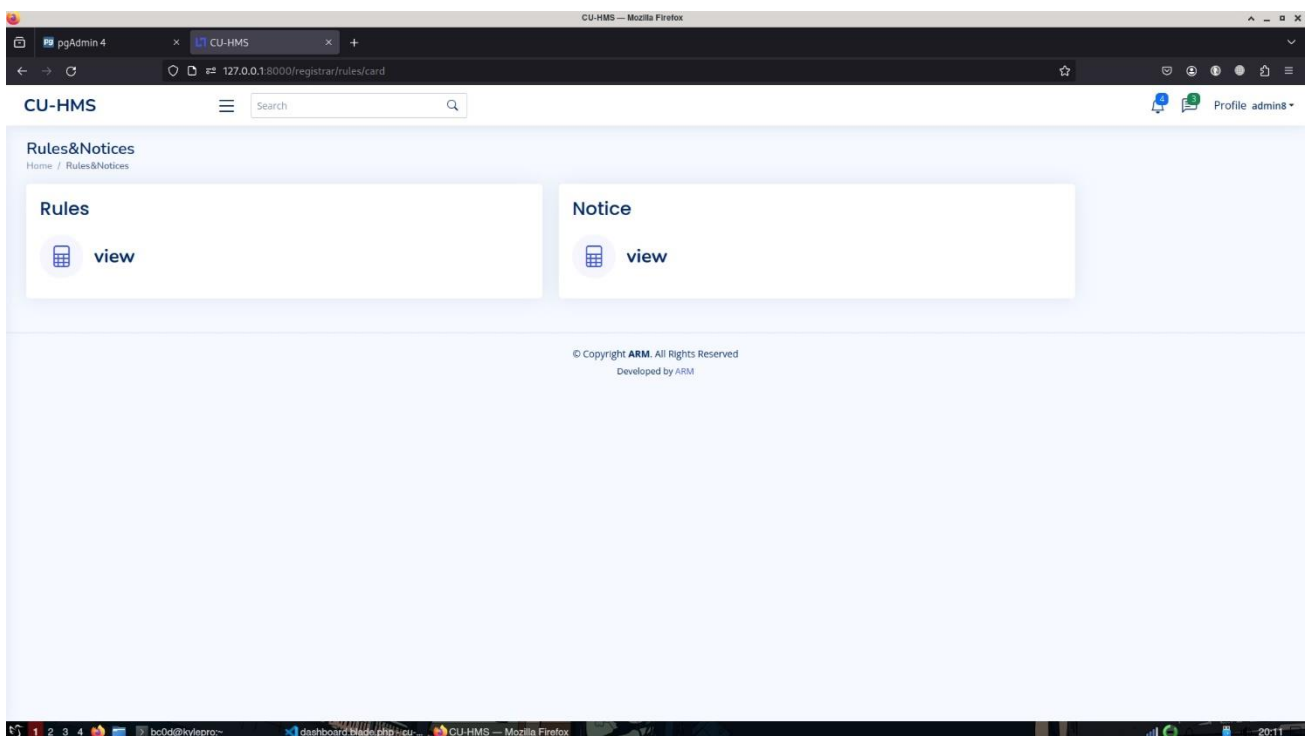
Blocks:

- ☒ Old block
- ☒ New block
- ☒ Annex
- ☐ New Annex
- ☐ Phd block
- ☐ Annex
- ☐ Everest
- ☐ Mulla

Status:

[Filter](#)

Bed ID	Bed Name	Bed Type	Status
1	MH1-1-1	single	vacant
4	MH1-4-1	single	vacant
5	MH1-5-1	single	vacant
6	MH2-1-1	single	vacant
8	MH2-3-1	single	vacant



CU-HMS Rules&Notices

Home / Rules&Notices

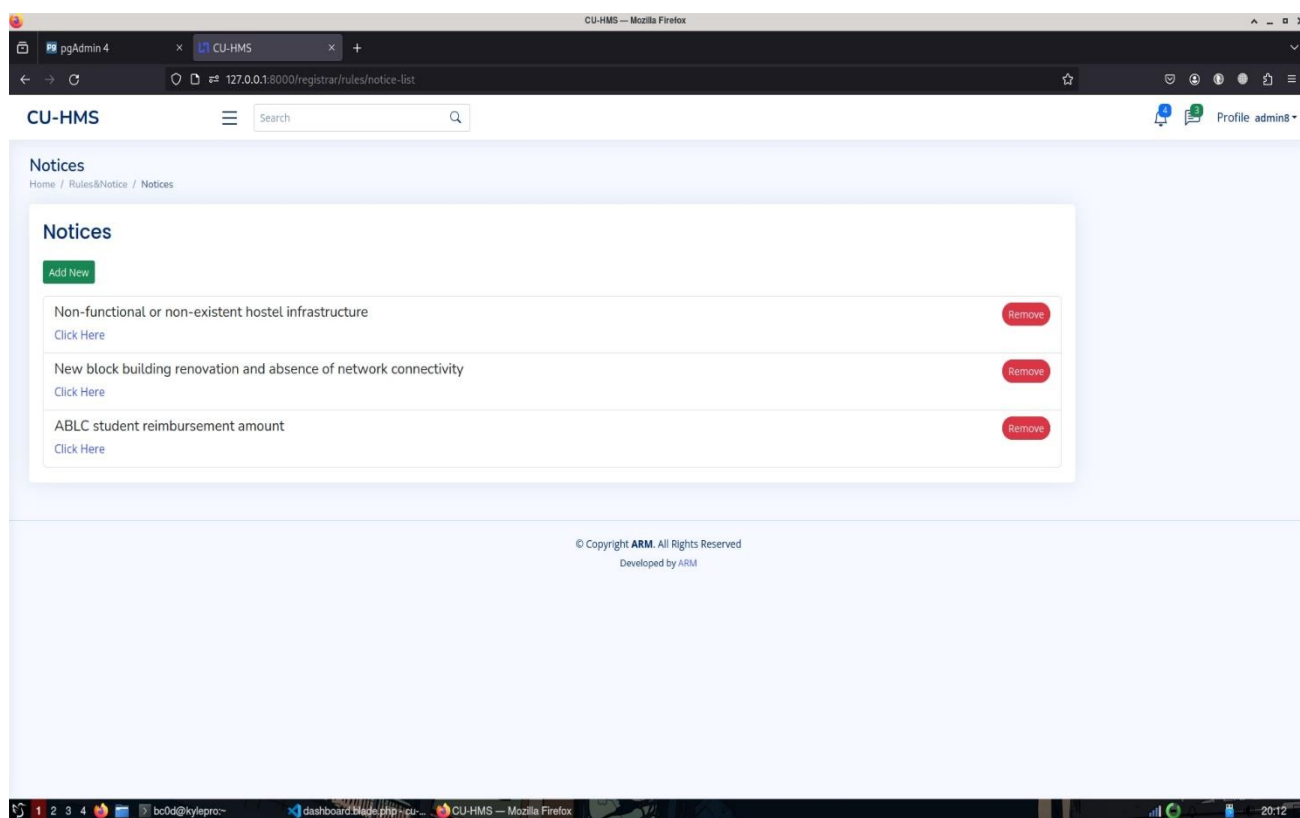
Rules

[view](#)

Notice

[view](#)

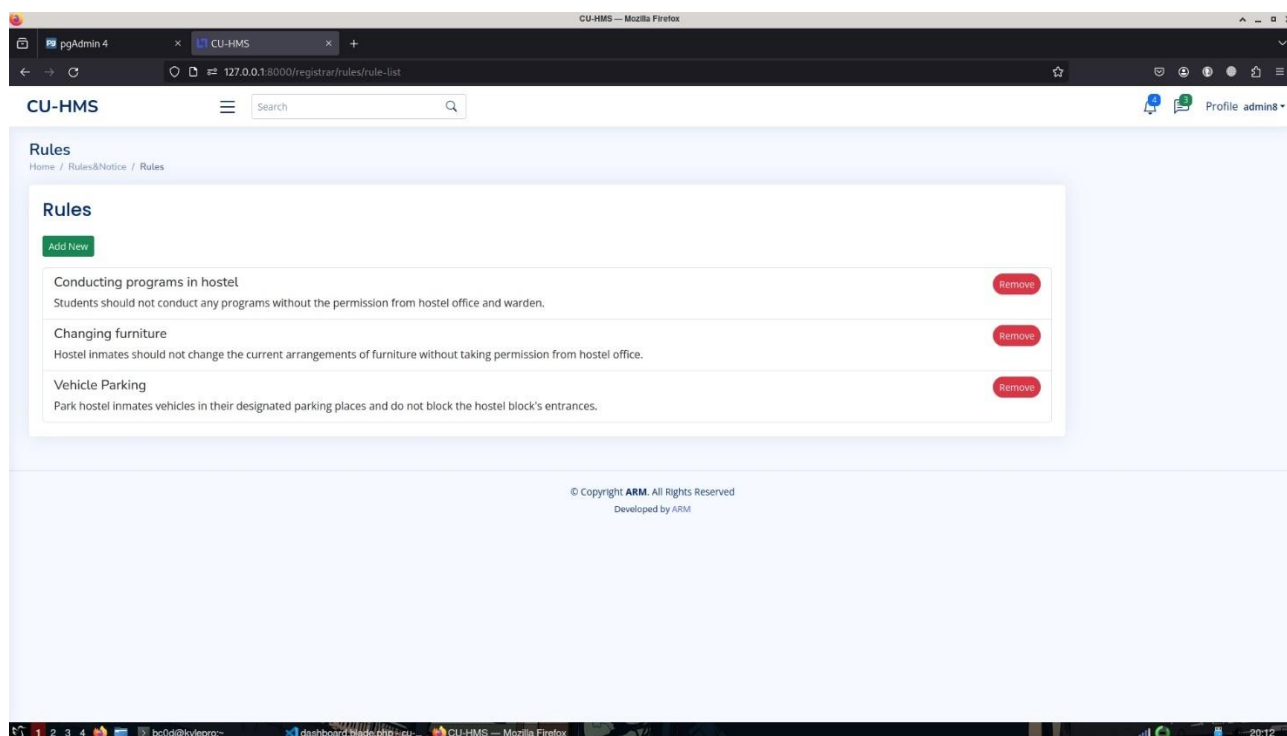
© Copyright ARM. All Rights Reserved
Developed by ARM



The screenshot shows the 'CU-HMS' web application in a Mozilla Firefox browser. The address bar displays '127.0.0.1:8000/registrars/rules/notice-list'. The page title is 'CU-HMS'. Below the title, there is a search bar and a user profile dropdown labeled 'Profile: admin8'. The main content area is titled 'Notices' and includes a breadcrumb trail 'Home / Rules&Notice / Notices'. A green 'Add New' button is located at the top left of the notices list. The list contains three items, each with a title, a 'Click Here' link, and a red 'Remove' button:

- Non-functional or non-existent hostel infrastructure
[Click Here](#) [Remove](#)
- New block building renovation and absence of network connectivity
[Click Here](#) [Remove](#)
- ABLC student reimbursement amount
[Click Here](#) [Remove](#)

At the bottom of the page, there is a copyright notice: '© Copyright ARM. All Rights Reserved. Developed by ARM'.



The screenshot shows the 'CU-HMS' web application in a Mozilla Firefox browser. The address bar displays '127.0.0.1:8000/registrars/rules/rule-list'. The page title is 'CU-HMS'. Below the title, there is a search bar and a user profile dropdown labeled 'Profile: admin8'. The main content area is titled 'Rules' and includes a breadcrumb trail 'Home / Rules&Notice / Rules'. A green 'Add New' button is located at the top left of the rules list. The list contains three items, each with a title, a description, and a red 'Remove' button:

- Conducting programs in hostel
Students should not conduct any programs without the permission from hostel office and warden. [Remove](#)
- Changing furniture
Hostel inmates should not change the current arrangements of furniture without taking permission from hostel office. [Remove](#)
- Vehicle Parking
Park hostel inmates vehicles in their designated parking places and do not block the hostel block's entrances. [Remove](#)

At the bottom of the page, there is a copyright notice: '© Copyright ARM. All Rights Reserved. Developed by ARM'.

SAMPLE CODES

OFFICE

1. web.php

```
<?php
use App\Http\Controllers\User\UserDashboardController;
use App\Http\Controllers\User\UserProfileController;
use App\Http\Controllers\User\UserComplaintsController;
use App\Http\Controllers\User\UserRoomController;
use App\Http\Controllers\User\UserFeedbackController;
use App\Http\Controllers\User\UserRulesAndNoticeController;
use App\Http\Controllers\User\UserFeeAndPaymentController;
use App\Http\Controllers\User\UserNotificationController;

use App\Http\Controllers\Registrar\RegistrarDashboardController;
use App\Http\Controllers\Registrar\RegistrarProfileController;
use App\Http\Controllers\Registrar\RoomDetailsRegistrarController;
use App\Http\Controllers\Registrar\ComplaintsRegistrarController;
use App\Http\Controllers\Registrar\FeeAndPaymentRegistrarController;
use App\Http\Controllers\Registrar\RuleAndNoticeRegistrarController;
use App\Http\Controllers\Registrar\RegistrarStudentDetailsController;

use App\Http\Controllers\Auth\RegisterController;
use App\Http\Controllers\Auth>LoginController;
use App\Http\Controllers\Auth\ResetPasswordController;

Route::get('login', [LoginController::class, 'showStudentLogin']);
Route::post('login', [LoginController::class, 'studentLogin'])->name('login');
Route::post('user-logout', [LoginController::class, 'studentLogout'])->name('logout');

Route::get('user-signup', [RegisterController::class, 'signupPageFirst'])->name('signup');

Route::post('signup/step1', [RegisterController::class, 'signupStep1'])->name('signupstep1');

Route::get('user-signup-dtls', [RegisterController::class, 'signupPageFinal']);

Route::post('signup/step2', [RegisterController::class, 'signupStep2'])->name('signupstep2');

Route::get('user-mail-confirm', function () {
    return view('users.auth.mailconfirm');
```

```
});
Route::get('/', function () {
    return view('index');
});

//payment gateway
Route::get('payment-gateway', [PaymentGatewayController::class, 'showPaymentGateway']);
Route::post('payment-gateway/payment', [PaymentGatewayController::class, 'makePayment'])->name('payment.makepay');

//user start
Route::middleware(['auth:students'])->prefix('user')->group(function () {

    //index
    Route::get('dashboard', [UserDashboardController::class, 'showDashboard'])->name('dashboard');

    Route::prefix('profile')->group(function () {

        //profile
        Route::get('/', [UserProfileController::class, 'showStudentProfile']);
        //Detailed profile
        Route::get('detailed', [UserProfileController::class, 'showMoreDetails']);
        //password reset
        Route::get('password-reset', [ResetPasswordController::class, 'showPasswordReset']);
        Route::post('reset', [ResetPasswordController::class, 'passwordReset'])->name('reset');
    });

    //complaint
    Route::prefix('complaints')->group(function () {

        //complaint-index
        Route::get('/', [UserComplaintsController::class, 'showComplaintSection']);
        //complaint register
        Route::get('register', [UserComplaintsController::class, 'showComplaintRegister']);
        Route::post('register-complaint', [UserComplaintsController::class, 'registerComplaint'])->name('register-complaint');
        //my-complaint
        Route::get('my-complaints', [UserComplaintsController::class, 'showMyComplaints']);
    });

    //start of room
    Route::prefix('room')->group(function () {

        //complaint-index
        Route::get('/', [UserRoomController::class, 'showRoomDetails']);

        Route::get('request', [UserRoomController::class, 'showRoomReq'])->name('room.callback');
        Route::post('room-req', [UserRoomController::class, 'roomRequest'])->name('room.request');
```

```
Route::post('room-req-paymet', [UserRoomController::class, 'roomAllocationPayment'])->
name('room.request.payment');
//endroom request

//start room change section
Route::prefix('change')->group(function () {
    Route::get('request', [UserRoomController::class, 'showRoomChangeRequest']);
    Route::post('request', [UserRoomController::class, 'roomChangeRequest'])->
name('room.change.request');
});
Route::prefix('vacate')->group(function () {

    Route::get('/', [UserRoomController::class, 'showRoomVacate']);
});
}); //end of room

//rules and notice card
Route::prefix('rules')->group(function () {

    Route::get('card', [UserRulesAndNoticeController::class, 'showCard']);
    Route::get('rule-list', [UserRulesAndNoticeController::class, 'viewRules']);
    Route::get('notice-list', [UserRulesAndNoticeController::class, 'viewNotices']);
});

//fee-pending-status
Route::prefix('bills-payments')->group(function() {

    Route::get('card', [UserFeeAndPaymentController::class, 'showBills']);
    Route::get('rent-card', [UserFeeAndPaymentController::class, 'viewRents']);
    Route::get('rent/{id}', [UserFeeAndPaymentController::class, 'payRoomRent']);
    Route::get('bills-card', [UserFeeAndPaymentController::class, 'viewBills']);
    Route::get('bill/{id}', [UserFeeAndPaymentController::class, 'payWaterElectricBill']);
    Route::get('pay', [UserFeeAndPaymentController::class, 'showPayment']);
});

//feedback
Route::prefix('feedback')->group(function () {

    Route::get('/', [UserFeedbackController::class, 'showFeedback']);
    //give feedback
    Route::get('give-feedback', [UserFeedbackController::class, 'showAddFeedback']);

    Route::post('submit-feedback', [UserFeedbackController::class, 'addFeedback'])->
name('user.addfeedback');
});
```

```
//notification
Route::get('notifications', [UserNotificationController::class, 'showNotifications']);

});
//end of user

Route::middleware(['auth:admins'])->prefix('registrar')->group(function () {
    //dashboard
    Route::get('index', [RegistrarDashboardController::class, 'showRegistrarDashboard']);
    //profile
    Route::get('my-profile', [RegistrarProfileController::class, 'showRegistrarProfile']);

    //student card
    Route::prefix('student')->group(function () {

        Route::get('card', [StudentDetailsRegistrarController::class, 'showCard']);
        Route::get('all', [StudentDetailsRegistrarController::class, 'showAllStudentDetails']);
        Route::get('detail/{id}', [StudentDetailsRegistrarController::class, 'showStudentProfileDetails']);
    });

    //rooms details card
    Route::prefix('room-details')->group(function () {

        Route::get('card', [RoomDetailsRegistrarController::class, 'showCard']);
        Route::get('list', [RoomDetailsRegistrarController::class, 'roomDetails']);
    });

    //Complaints
    Route::prefix('complaints')->group(function () {

        Route::get('card', [ComplaintsRegistrarController::class, 'showComplaintsCard']);
        Route::get('new', [ComplaintsRegistrarController::class, 'showNewComplaints']);
        Route::get('view/{id}', [ComplaintsRegistrarController::class, 'showComplaintView']);
        Route::post('edit', [ComplaintsRegistrarController::class, 'complaintEdit'])->
            name('complaint.action');
        Route::get('solved', [ComplaintsRegistrarController::class, 'showSolvedComplaints']);

        Route::get('all', [ComplaintsRegistrarController::class, 'showAllComplaints']);

    });
    //admission card
    Route::prefix('admission')->group(function () {

        Route::get('request', [HostelAdmissionRegistrarController::class, 'showRequests']);
        Route::get('action', [HostelAdmissionRegistrarController::class, 'admissionAction']);
```

```
});

//room allocation
Route::prefix('room')->group(function () {

    Route::get('allocation-list', [RoomAllocationRegistrarController::class, 'showRoomAllocList']);
    Route::get('allocation', [RoomAllocationRegistrarController::class, 'roomAllocAction']);
});

//room change card
Route::prefix('room-change')->group(function () {

    Route::get('request', [RoomChangeRegistrarController::class, 'showRoomChangeReq']);
    Route::get('action', [RoomChangeRegistrarController::class, 'roomChangeAction']);
});

//vacating card
Route::prefix('vacate')->group(function () {

    Route::get('request', [HostelVacateRegistrarController::class, 'showRequests']);
    Route::get('action', [HostelVacateRegistrarController::class, 'vacateAction']);
});

//fee card
Route::prefix('fee')->group(function () {

    Route::get('card', [FeeAndPaymentRegistrarController::class, 'showCard']);
    Route::get('room-rent', [FeeAndPaymentRegistrarController::class, 'roomRentDetails']);

    //fee maintainance
    Route::get('maintanance', [FeeAndPaymentRegistrarController::class, 'feeMaintanance']);
    Route::get('updatation', [FeeAndPaymentRegistrarController::class, 'feeUpdate']);
});

//rules and notice card
Route::prefix('rules')->group(function () {

    Route::get('card', [RuleAndNoticeRegistrarController::class, 'showCard']);

    Route::get('rule-list', [RuleAndNoticeRegistrarController::class, 'viewRules']);
    Route::get('rule-add', [RuleAndNoticeRegistrarController::class, 'viewAddRule']);
    Route::post('add-rule', [RuleAndNoticeRegistrarController::class, 'addRule'])->
name('registrar.rules.add');
    Route::post('remove-rule', [RuleAndNoticeRegistrarController::class, 'removeRule'])->
name('registrar.rules.remove');
```

```

Route::get('notice-list', [RuleAndNoticeRegistrarController::class, 'viewNotices']);
Route::get('notice-add', [RuleAndNoticeRegistrarController::class, 'viewAddNotice']);
Route::post('notice-Add', [RuleAndNoticeRegistrarController::class, 'addNotice'])->>
name('registrar.notice.add');
Route::post('remove-notice', [RuleAndNoticeRegistrarController::class, 'removeNotice'])->>
name('registrar.notice.remove');
});
Route::get('student-details', [RegistrarStudentDetailsController::class, 'showStudentDetails']);
Route::post('list', [RegistrarStudentDetailsController::class, 'viewStudentDetails']);
Route::get('blocks/{hostel}', [RegistrarStudentDetailsController::class, 'getBlocks']);
Route::post('student-details', [RegistrarStudentDetailsController::class, 'filterStudents'])->
>name('registrar.student.list');
});

```

2.complaint_register.blade.php

```
@extends('layout.public_master')
```

```
@section('content')
```

```
<!-- ===== Contact Section ===== -->
```

```
<section id="contact" class="contact sign-sec">
```

```
<h2 class="sign-sec text-center">Give Your Complaint</h2><hr>
```

```
<div class="container">
```

```
@if (session('message'))
```

```
<div class="alert alert-success">
```

```
{{ session('message') }}
```

```
</div>
```

```
@endif
```

```
<div class="row mt-1 justify-content-center">
```

```
<div class="col-lg-8 mt-2 mt-lg-0 pt-2 pt-3">
```

```
<form action="{{ route('register-complaint') }}" method="POST" role="form" class="php-email-form">
```

```
@csrf
```

```
<div class="row ">
```

```
<div class="form-floating mt-1">
```

```
<div class="col-6">
```

```
<label class="form-label pt-2 ">Complaint Category<span aria-hidden="true" class="required-fields"> *</span></label>
```

```
</div>
```

```
<select name="complaint_cat" id="select1" class="form-select ">
```

```
<option value="" selected disabled>Select</option>
```

```
<option value="Hostel">Hostel</option>
```

```
<option value="Mess">Mess</option>
```

```
<option value="Electricity">Electricity</option>
```



```

        <option value="Internet">Internet</option>
        <option value="S&M">Staff and Management</option>
        <option value="Other">Other</option>
    </select>
</div>
</div>

<div class="form-group mt-3 mb-3 pt-2">
    <div class="col-6">
        <label class="form-label">Your Complaint<span aria-hidden="true" class="required-fields">
*</span></label>
    </div>
    <div class="col">
        <textarea class="form-control" name="complaint_msg" rows="3" placeholder=""></textarea>
    </div>
</div>
<div class="text-center">
    <button type="submit" class="btn btn-primary">Submit</button>
</div>
</form>

</div>

</div>

</div>
</section><!-- End Contact Section -->
@endsection

```

3.student_detail_form.blade.php

```

@extends('layout.admin_master')

@section('content')

<div class="pagetitle">

    <h1>Registrar</h1>
    <nav>

        <ol class="breadcrumb">
            <li class="breadcrumb-item"><a href={{ url('registrar/index') }}>Home</a></li>
            <li class="breadcrumb-item active">Profile</li>
        </ol>
    </nav>

```

```
</div><!-- End Page Title -->
```

```
<section class="section">
```

```
<div class="row">
```

```
<div class="col-lg-10">
```

```
<div class="card">
```

```
<div class="card-body">
```

```
<h5 class="card-title">Student Details</h5>
```

```
<!-- Vertical Form -->
```

```
<form method="POST" action="{{ route('registrar.student.list') }}">
```

```
@csrf
```

```
<div>
```

```
<label for="hostel">Hostel:</label>
```

```
<select id="hostel" name="hostel">
```

```
<option value="all">All</option>
```

```
<option value="1">Mens Hostel</option>
```

```
<option value="2">Ladies Hostel</option>
```

```
</select>
```

```
</div>
```

```
<div id="blocks-container" style="display: none;">
```

```
<label>Blocks:</label>
```

```
<div>
```

```
<input type="checkbox" id="select-all-blocks">
```

```
<label for="select-all-blocks">All</label>
```

```
</div>
```

```
<div id="blocks-list"></div>
```

```
</div>
```

```
<button type="submit">Show Students</button>
```

```
</form>
```

```
@if(isset($students))
```

```
<h2>Student List</h2>
```

```
<form method="POST" action="{{ url('office/students/export') }}">
```

```
@csrf
```

```
<input type="hidden" name="hostel" value="{{ request('hostel') }}">
```

```
<input type="hidden" name="blocks" value="{{ json_encode(request('blocks')) }}">
```

```
<button type="submit">Export to Excel</button>
```

```
</form>
```

```
<table border="1">
```

```
<thead>
```

```
<tr>
```

```
<th>Name</th>
```

```
<th>Email</th>
```

```
<th>Bed</th>
```

```
<th>Room</th>
```

```
<th>Block</th>
```

```
<th>Hostel</th>
```

```
</tr>
```

```
</thead>
```

```

        <tbody>
            @foreach($students as $student)
                <tr>
                    <td>{{ $student->name }}</td>
                    <td>{{ $student->email }}</td>
                    <td>{{ $student->bed->bed_name }}</td>
                    <td>{{ $student->bed->room->room_name }}</td>
                    <td>{{ $student->bed->room->block->block_name }}</td>
                    <td>{{ $student->bed->room->block->hostel->hostel_name }}</td>
                </tr>
            @endforeach
        </tbody>
    </table>
</div>
</div>
</div>
</div>
</section>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
    $(document).ready(function() {
        $('#hostel').change(function() {
            var hostel = $(this).val();
            if (hostel && hostel !== 'all') {
                $.ajax({
                    url: '/registrar/blocks/' + hostel,
                    type: 'GET',
                    dataType: 'json',
                    success: function(data) {
                        $('#blocks-container').show();
                        $('#blocks-list').empty();
                        $.each(data, function(key, value) {
                            $('#blocks-list').append(
                                '<div><input type="checkbox" name="blocks[]" value="" + value.block_id + ""> ' +
                                '<label>' + value.block_name + '</label></div>'
                            );
                        });
                    }
                });
            } else {
                $('#blocks-container').hide();
                $('#blocks-list').empty();
            }
        });
    });
    $('#select-all-blocks').change(function() {

```

```

        var isChecked = $(this).is(':checked');
        $('#blocks-list input[type="checkbox"]').prop('checked', isChecked);
    });
});
</script>

@endsection

```

4. CalculateMonthlyFees.php

```

<?php

namespace App\Jobs;

use Illuminate\Bus\Queueable;
use Illuminate\Contracts\Queue\ShouldQueue;
use Illuminate\Foundation\Bus\Dispatchable;
use Illuminate\Queue\InteractsWithQueue;
use Illuminate\Queue\SerializesModels;
use App\Models\Student;
use App\Models\Fee;
use App\Models\RoomRent;
use App\Models\WaterElectricBill;
use Carbon\Carbon;

class CalculateMonthlyFees implements ShouldQueue
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    public function __construct()
    {
        //
    }

    public function handle()
    {
        $currentMonth = Carbon::now()->startOfMonth();

        // Fetch active students
        $students = Student::with('bed.room.block.hostel')->where('status', 'Active')->get();

        foreach ($students as $student) {

```

```
// Calculate room rent
$roomRentFee = Fee::where('hostel_id', $student->bed->room->block->hostel_id)
    ->where('room_type', $student->bed->room->room_type)
    ->where('fee_name', 'room rent')
    ->first();

if ($roomRentFee) {
    RoomRent::create([
        'fee_id' => $roomRentFee->fee_id,
        'student_id' => $student->student_id,
        'month_of_fee' => $currentMonth->toDateString(),
        'paid_status' => 'Pending',
        'payment_date' => null,
        'transaction_id' => null,
        'amount' => $roomRentFee->amount,
    ]);
}

// Calculate water/electric bill
$waterElectricFee = Fee::where('hostel_id', $student->bed->room->block->hostel_id)
    ->where('room_type', $student->bed->room->room_type)
    ->where('fee_name', 'water/electric bill')
    ->first();

if ($waterElectricFee) {
    WaterElectricBill::create([
        'fee_id' => $waterElectricFee->fee_id,
        'student_id' => $student->student_id,
        'month_of_fee' => $currentMonth->toDateString(),
        'paid_status' => 'Pending',

        'payment_date' => null,
        'transaction_id' => null,
        'amount' => $waterElectricFee->amount,
    ]);
}

}

// Log success or perform any additional actions if needed
logger('Monthly fees calculated successfully.');
```

5. UserProfileController.php

```
<?php

namespace App\Http\Controllers\User;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Models\Course;
use App\Models\Bed;

class UserProfileController extends Controller
{
    public function showStudentProfile() {
        $student = Auth::guard('students')->user();
        $course = Course::with('department')->where('course_id', $student->course_id)->first();
        if($student->bed_id === Null) {
            $bed = Null;
            return view('users.profile', compact('student', 'course', 'bed'));
        } else {
            $bed = Bed::with('room.block.hostel')->findOrFail($student->bed_id);
            return view('users.profile', compact('student', 'course', 'bed'));
        }
    }

    public function showMoreDetails() {

        $student = Auth::guard('students')->user();
        $course = Course::with('department')->where('course_id', $student->course_id)->first();
        if($student->bed_id === Null) {
            $bed = Null;
            return view('users.profileDetailed', compact('student', 'course', 'bed'));
        } else {
            $bed = Bed::with('room.block.hostel')->findOrFail($student->bed_id);
            return view('users.profileDetailed', compact('student', 'course', 'bed'));
        }
    }
}
```

6. UserRoomController.php

```
<?php

namespace App\Http\Controllers\User;

use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
```

```
use App\Models\Department;
use App\Models\RoomAllocation;
use App\Models\FeeDetail;
use App\Models\Bed;
use App\Models\Transaction;
use App\Models\RoomChange;
use Illuminate\Http\Request;

class UserRoomController extends Controller
{
    public function showRoomSection() {

        $student = Auth::guard('students')->user();
        return view('users.room_index', compact('student'));
    }

    public function showRoomRent() {

        $student = Auth::guard('students')->user();
        return view('users.room_rent', compact('student'));
    }

    public function showRoomRentPayment() {

        $student = Auth::guard('students')->user();
        return view('users.room_rent_payment', compact('student'));
    }

    public function showRoomRentStatus() {

        $student = Auth::guard('students')->user();
        return view('users.room_rent_status', compact('student'));
    }

    public function showRoomDetails() {

        $student = Auth::guard('students')->user();
        $bed = Bed::with('room.block.hostel')->findOrFail($student->bed_id);
        return view('users.room_details', compact('student', 'bed'));
    }

    public function showRoomReq() {

        $student = Auth::guard('students')->user();
        $department = Department::where('department_id', $student->department)->first();
```

```
$roomAlloc = RoomAllocation::where('student_id', $student->student_id)->first();
$transaction = Transaction::firstWhere([
    'student_id' => $student->student_id,
    'purpose' => 'Hostel Admission'
]);
if(!is_Null($transaction)) {
    $roomAlloc->payment_status = $transaction->status;
    $roomAlloc->transaction_id = $transaction->transaction_id;
    $roomAlloc->save();
}
$fee = FeeDetail::where('fee_title', 'like', '%Admission%')->get();
$total = 0.0;
foreach($fee as $val) {
    $total = $total + $val->amount;
}
return view('users.room_request', compact('student', 'department', 'roomAlloc', 'fee', 'total'));
}
```

```
public function roomRequest(Request $request) {

    $student = Auth::guard('students')->user();
    $data = $request->validate([
        'hostel' => 'required|string',
    ]);

    RoomAllocation::create([
        'student_id' => $student->student_id,
        'department_id' => $student->department,
        'hostel' => $data['hostel'],
        'dep_verification_status' => 'Pending',
        'warden_verification_status' => 'Pending',
        'payment_status' => 'Pending',
        'allocation_status' => 'Pending',
    ]);

    return redirect()->back()->with('message', 'Room Requested');
}
```

```
public function roomAllocationPayment(Request $request) {

    $student = Auth::guard('students')->user();
    $data = $request->validate([
        'student_id' => 'required|string',
        'purpose' => 'required|string',
        'amount' => 'required|numeric',
    ]);
```



```
$paymentDetails = [  
  'student_id' => $data['student_id'],  
  'purpose' => $data['purpose'],  
  'amount' => $data['amount'],  
  'callback_url' => route('room.callback'),  
];  
  
session()->flash('paymentDetails', $paymentDetails);  
return redirect('/payment-gateway');  
}  
  
public function showRoomChangeRequest() {  
  
  $student = Auth::guard('students')->user();  
  return view('users.room_change', compact('student'));  
}  
  
public function roomChangeRequest(Request $request) {  
  
  $student = Auth::guard('students')->user();  
  $data = $request->validate([  
    'preference' => 'required|string',  
    'reason' => 'required|string',  
  ]);  
  
  $bed = Bed::findOrFail($student->bed_id);  
  $roomChange = new RoomChange();  
  $roomChange->student_id = $student->student_id;  
  $roomChange->current_room = $bed->room_id;  
  $roomChange->request = $data['preference'];  
  $roomChange->reason = $data['reason'];  
  $roomChange->status = 'Pending';  
  $roomChange->save();  
  
  return redirect()->back()->with('message', 'Room change request submitted successfully!!');  
}  
  
public function showRoomVacate() {  
  
  $student = Auth::guard('students')->user();  
  
  return view('users.room_vacate', compact('student'));  
}  
}
```

7. UserFeedbackController.php

```
<?php

namespace App\Http\Controllers\User;

use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use App\Models\Feedback;
use Illuminate\Http\Request;

class UserFeedbackController extends Controller
{
    public function showFeedback() {

        $student = Auth::guard('students')->user();
        $feedbacks = Feedback::with('student')->orderBy('updated_at', 'desc')->get();
        return view('users.feedback_index', compact('student', 'feedbacks'));
    }

    public function showAddFeedback() {

        $student = Auth::guard('students')->user();
        return view('users.feedback_give', compact('student'));
    }

    public function addFeedback(Request $request) {

        $student = Auth::guard('students')->user();
        $data = $request->validate(['feedback' => 'required|string']);
        $feedback = new Feedback();
        $feedback->student_id = $student->student_id;
        $feedback->review = $data['feedback'];
        $feedback->save();
        return redirect()->back()->with('message', 'Your feedback added successfully');
    }
}
```

8. RulesAndNoticesRegistrarController

```
<?php

namespace App\Http\Controllers\Registrar;

use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use Illuminate\Http\Request;
```

```
use App\Models\Rule;
use App\Models\Notice;

class RuleAndNoticeRegistrarController extends Controller
{
    public function showCard() {
        $admin = Auth::guard('admins')->user();
        return view('admins.registrar.rules_card',compact('admin'));
    }

    public function viewRules() {
        $admin = Auth::guard('admins')->user();
        $rules = Rule::all();
        return view('admins.registrar.rules_list',compact('admin','rules'));
    }

    public function viewAddRule() {
        $admin = Auth::guard('admins')->user();
        return view('admins.registrar.rules_add', compact('admin'));
    }

    public function addRule(Request $request) {
        $admin = Auth::guard('admins')->user();
        $data = $request->validate([
            'ruleName' => 'required|string',
            'ruleDesc' => 'required|string',
        ]);
        $rules = new Rule();
        $rules->title = $data['ruleName'];
        $rules->description = $data['ruleDesc'];
        $rules->updatedby = $admin->admin_id;
        $rules->save();
        return redirect()->intended('registrar/rules/rule-list');
    }

    public function removeRule(Request $request) {

        $data = $request->validate(['ruleId' => 'required|string']);

        $rule = Rule::findOrFail($data['ruleId']);
        $rule->delete();
        return redirect()->back();
    }

    public function viewNotices() {
        $admin = Auth::guard('admins')->user();
        $notices = Notice::all();
    }
}
```

```

        return view('admins.registrar.notice_list',compact('admin','notices'));
    }

    public function viewAddNotice() {
        $admin = Auth::guard('admins')->user();
        return view('admins.registrar.notice_add',compact('admin'));
    }

    public function addNotice(Request $request) {
        $admin = Auth::guard('admins')->user();
        $noticeData=$request->validate([
            'newNotice' => 'required',
            'noticeSubject' => 'required|string',

        ]);
        $file = $request->file('newNotice');
        $extension = $file->getClientOriginalExtension();
        $filename = time().'.'.$noticeData['noticeSubject'].'.'.$extension;
        $path = 'data/notice/';
        $file->move($path, $filename);
        $notice = new Notice();
        $notice->title = $request->noticeSubject;
        $notice->publishedby = $admin->admin_id;
        $notice->path = $path.$filename;
        $notice->save();
        return redirect('registrar/rules/notice-list');
    }

    public function removeNotice(Request $request) {

        $data = $request->validate(['noticeId' => 'required|string']);
        $notice = Notice::findOrFail($data['noticeId']);
        $notice->delete();
        return redirect()->back();

    }
}

```

9. ComplaintRegistrarController.php

```

<?php

namespace App\Http\Controllers\Registrar;

use App\Http\Controllers\Controller;

```

```
use Illuminate\Support\Facades\Auth;
use App\Models\Complaint;
use Illuminate\Http\Request;

class ComplaintsRegistrarController extends Controller
{
    public function showComplaintsCard() {

        $admin = Auth::guard('admins')->user();
        return view('admins.registrar.complaints_card', compact('admin'));
    }

    public function showNewComplaints() {

        $admin = Auth::guard('admins')->user();
        $complaints = Complaint::where('status', 'Pending')->get();
        return view('admins.registrar.complaints_list', compact('admin', 'complaints'));
    }

    public function showSolvedComplaints() {

        $admin = Auth::guard('admins')->user();
        $complaints = Complaint::where('status', 'Solved')->get();
        return view('admins.registrar.complaints_list', compact('admin', 'complaints'));
    }

    public function showAllComplaints() {

        $admin = Auth::guard('admins')->user();
        $complaints = Complaint::all();
        return view('admins.registrar.complaints_list', compact('admin', 'complaints'));
    }

    public function showComplaintView($id) {

        $admin = Auth::guard('admins')->user();
        $complaint = Complaint::findOrFail($id);

        if ($complaint->status === 'Pending') {

            $complaint->status = 'Viewed';
            $complaint->save();
        }

        return view('admins.office.complaint', compact('admin', 'complaint'));
    }
}
```

```
public function complaintEdit(Request $request) {  
  
    $admin = Auth::guard('admins')->user();  
  
    $data = $request->validate([  
        'status' => 'required|string',  
        'comment' => 'required|string',  
        'id' => 'required|string|exists:complaints,complaint_id', // Add exists rule  
    ]);  
  
    Complaint::where('complaint_id', $data['id'])->update([  
        'status' => $data['status'],  
        'closedby' => $admin->admin_id,  
        'comment' => $data['comment'],  
    ]);  
  
    return redirect()->back()->with('success', 'Complaint updated successfully.');
```

```
    }  
}
```

BIBLIOGRAPHY

BIBILIOGRAPHY

- <https://www.w3schools.com/>
- <https://www.google.com/>
- <https://stackoverflow.com/>
- <https://laracasts.com/>
- <https://chat.openai.com/>
- <https://www.tutorialspoint.com/>
- <https://www.geeksforgeeks.org/>
- <https://learn.jquery.com/>
- <https://laravel.com/>