UNIVERSITY OF NAIROBI

FACULTY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTING AND INFORMATICS

HOSTEL MANAGEMENT SYSTEM

Submitted by

Nangulu Hezron Wekesa

REG NO: ***P15/136414/2019***

Submitted to

SUPERVISOR: Mr. Eric Masiga Ayienga

A second year project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science in Computer Science of the University of Nairobi.

March 2021

Declaration

This section is to certify that this project was carried out by Nangulu Hezron Wekesa at the University of Nairobi, faculty of science and technology and department of computing and informatics as a second-year project.

STATEMENT OF AUTHENTICITY

I have read the university regulations relating to plagiarism and certify that this report is all my own work and does not contain any unacknowledged work from any other source. I also declare that I have been under supervision for this report here is submitted.

Name	Registration Number	Signature	Date
Nangulu Hezron Wekesa	P15/136414/2019	……[illegible]………	……15th March 2022

SUPERVISORS'S DECLARATION

I hereby declare that the preparation and presentation of this project report were supervised in accordance with the guidelines on supervision Laid down by The University of Nairobi.

Mr. Eric Masiga Ayienga	…………………….	….…………………..
Supervisor Name	Signature	Date

Acknowledgement

Presenting this project report on “Hostel Management System” as part of the 2nd semester of Bachelor in Computer Science at The University of Nairobi. I would like to thank my Project Supervisor, Mr. Eric Masiga Ayienga, for helping me with this project, and the project documentation. He allowed me to work on this project. My special thanks go to Mr. Eric Masiga Ayienga for the keen interest shown in completing this thesis successfully. Along with that, I would like to thank my lecturers who gave me a golden opportunity to apply knowledge learned from them on this project.

I’d also like to express my gratitude to the department of computing and informatics wholeheartedly. I also thank all the staff of the department for their kindness.

I must also thank my parents and friends for their support and help during this project. Without their help, completing this project would have been very difficult. I am extremely grateful to my parents for their love, prayers, care and sacrifices; for educating and preparing me for my future. Also, I express my thanks to my sisters, brother, sister-in-law and brothers-in-law for their support and valuable prayers.

Finally, I would like to thank God for being able to complete this project with success, and for blessing me with his grace and taking our endeavours to a successful culmination. The Lord has been faithful in granting me the strength, wisdom, knowledge, and the courage needed throughout this period of study. I thank the Almighty God and Father above, for the life and success of this study.

Abstract

Hostel Management System is software developed to manage various activities in the hostel. This particular project deals with the problems of managing the hostel and avoids the problems that occurs when carried out manually process (using pens and papers). Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible to the existing system with the system that is more users friendly and more GUI oriented. For the past 25 years, the number of students joining universities has increased. This has also resulted in an increase in the number of hostels that provide accommodation for students studying at different institutions. Due to the rapid and huge influx of students in hostels, it has caused a lot of strain on the caretakers or the landlords who are running the hostels. Currently, the majority of hostels are managed manually, which leads to load and time consumption.

This particular project, that I am developing will deal with the problems of managing a hostel and avoid the problems which occur when managing hostels manually. With this system, we can easily manage the room details, student records, hostel expenditure, rent payment and calculation, and other hostel details. It will provide an easy way of room allocation and hostel attendance.

Identifying the demerits of the existing system led us to the design of a computerized hostel management system that would be compatible with the existing system. We can improve the efficiency of the system, thus overcoming the drawbacks of the existing system.

During the development of this project, I visited the hostel administration for collection of data and to analyze their problems. We have tried to fulfill most of the desires of the hostel requirements, but due to the lack of time, some of the requirements were solved. The problems analyzed for the hostel were.

- Check in/checkout system for students to keep records of rooms that have been reserved.
- Room Reservation system to keep records of the students and find their room number.
- To generate a report of each room reserved.

Hostel Management System is an individual project that I will develop using Python as frontend and backend, and MySQL for the database.

Table of Contents

CHAPTER ONE: INTRODUCTION

1.1 BACKGROUND

Computer as an indispensable tool in the total life of human endeavor has gained precedence in the world especially the development grows of the Kenya society. Due to this advancement of Technology, both software and hardware, the application areas of computers are rising day by day. Thus, it is not advisable to use manual systems.

The gradual increase in human population has equally affected the education sector as higher institutions in the country today are currently faced, with the problem of how to allocate hostels/halls of residence to students. It is common that the accommodation of students usually out numbers the accommodation available. Hence, there is the need for computerization of the hostel process to manage accommodation of students. Hostels without a computerized management system are usually done manually, meaning any records related to the hostels are stored on sheets of paper. The drawbacks of existing systems have led to the design of a computerized system that will help reduce a lot of manual inputs. With a computerized system in place, we can improve the efficiency as well as the effectiveness of the system, thus overcoming the drawbacks of the existing manual system.

The hostel management system was developed in favor of hostel management staff. It will help them save the records of the students, to know which room is vacant, manage hostel rent payments, and other hostel-related details. This system will also make it easier for staff to run the hostels.

The idea behind developing this system was to automate the manual work and make it easy to allocate rooms to students. The system doesn't require such efficient personnel to manage the hostel affairs. All they have to do is to log in as admin and from there they will be able to obtain all the information of all the students registered and allocate new students to available rooms/hostel.

I chose hostel management software as the system seemed simple and easy to use and this system gives solutions to existing problems in most hostels here in Kenya and other parts of the world that haven't automated their hostels. This system is user-friendly and more graphic user interface (GUI) oriented, thus improving and overcoming the drawbacks of managing a hostel.

1.2 PROBLEM STATEMENT

The growing number of students in higher institutions of learning all over the world has posed a lot of accommodation problems on the part of students and school management, especially in Kenya. Students at the beginning of each session spend a lot of time looking for accommodation. It is common to find students in need of accommodation are not aware when there is a free space in a hostel. The warden sometimes labors to check physically from hostel to hostel to ascertain the available spaces. This is because there is no central automated repository to always help the warden to know which room. Due to human error in the current paper-based system, there are some times double allocations where more students than the room can accommodate are allocated to one room, which sometimes brings conf1icts among the students. All these challenges can be answered by the development of computer-based automated hostel management software which will enable secure, easy, timely and effective way of assigning hostel room to students and generating/ retrieving any specific information that can be required from time to time about students in the different university hostel.

Despite advancements in technology, there are still hostels in many institutions in Kenya and the world who are still using manual systems to manage the hostels. The difference between computers and humans is that computers are faster and more accurate, which reduces mistakes.

The current manual system(paper-based system) has the following difficulties;

1.Slow data retrieval

2.Poor data storage

3.Poor data security

One scenario of the manual system is, the hostel staff might find it hard to know the number of students in the hostels. Thus, he or she has to go room by room to check if the room is occupied or not. Usually, all records related are saved on paper or huge notebooks, and sometimes receipts. If the books should go missing or stolen, it would be a huge loss for the hostel since it's business running around that book.

1.3 Objectives

The aim of the proposed system is to provide solutions to the problems stated above and help the user to manage the hostels effectively and efficiently throughout.

The general objective of this project, Hostel Management System, is to:

- TO MAKE IT EASIER FOR DATA COLLECTION, STORAGE AND REFERENCING RELIABLE.
- ENABLE HOSTEL MANAGERS TO EASILY TRACK THE PAYMENT AND STUDENTS' DATA.
- REDUCE THE COST THE ADMINISTRATION SPENDS ON PAPER WORKS AND STUFF.
- MAKE It EASY FOR HOSTEL MANAGERS TO VIEW PAYMENT REPORT USING THEIR COMPUTERS.
- TO MAINTAIN THE STUDENTS AS HOSTELLERS AND WAITING LIST STUDENTS SEPARATELY.
- AUTOMATE THE BOOKINGS AND NOTIFICATION.
- TO PROCESS ALLOTMENT LIST.

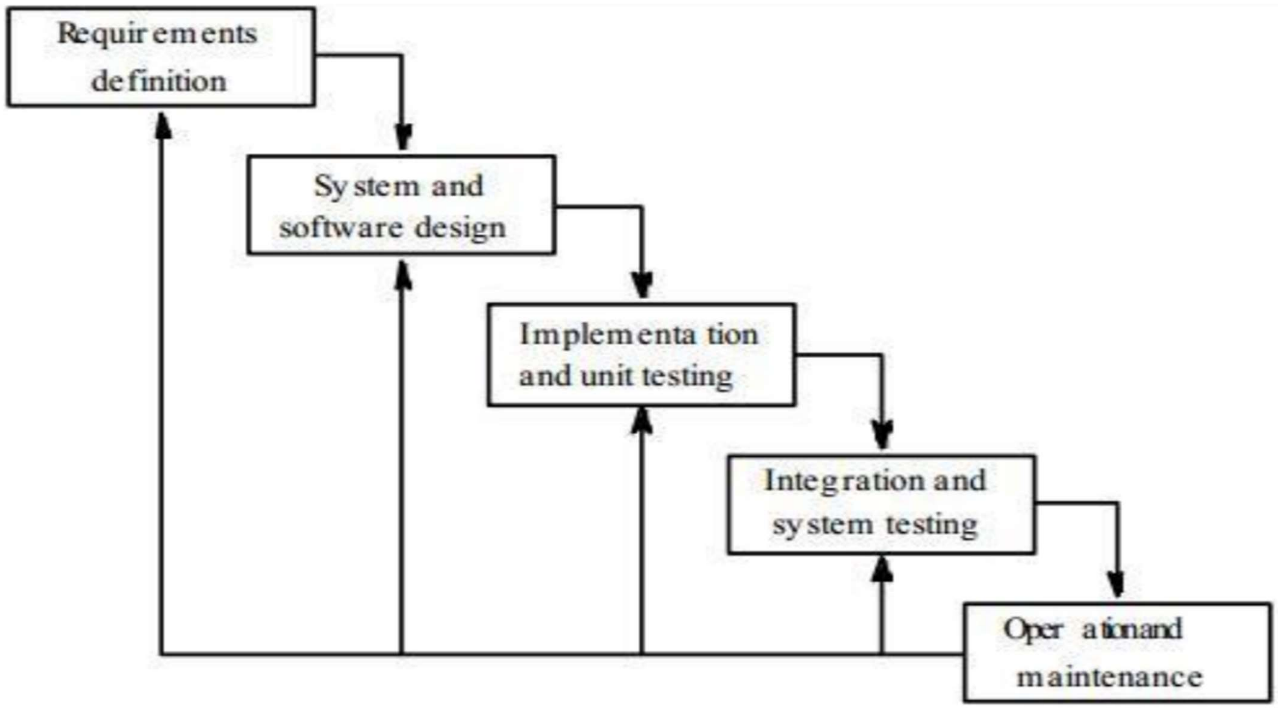
STAKE HOLDER	**OBJECTIVE**
STUDENT	THE STUDENT CAN STORE HIS OR HER INFORMATION
ADMINISTRATIVE	THE WARDEN CAN SEE THE DATA OF STUDENTS

O Other objectives that this project intends to achieve are;

- Automate the current manual system.
- Quick retrieval of students' information.
- Find out the possible benefits of the new system.
- To gather necessary requirements for developing a convenient and flexible hostel allocation
- management system
- To test, validate and implement the system
- To develop a central database system that will serve as hostel database, which will contain
- information on all the available rooms in the hostels.
- Reduce the cost the management spends on labor running the hostels

1.4 METHODOLOGY

This project follows the waterfall model.


Requir em ents de finition
Sy stem and software design
Implem enta tion and unit testing
Integ ration and sy stem testing
Oper a tionand maintenance

Requirement Analysis

At this stage, the whole system needs should be obtained in this phase, including the expected utility, user and system limitations.
This information can will be obtained through interviews , surveys or discussions and online research. The information is analyzed to obtain user documentation needs to be used at a later stage

System Design

This section is done before encoding. This section aims to provide a summary of what to do and what to look like. This section helps to define hardware and system requirements as well as define the overall system architecture.

Implementation

At this stage of planning. Software development is subdivided into smaller modules that will be integrated into the next phase. And at this stage also checks in the module are made, whether it meets the function you want or not

Integration & Testing

At this stage, the installation of modified and tested modules is done to determine if the software is compliant with the design and there are errors or not.

Operation & Maintenance

At this stage the Software that has been so run down and do maintenance Maintenance included in repair errors not found in previous phases.

1.5 RESEARCH OBJECTIVES

Based on the need for an effective and efficient system for hostels and some amount of probing into the matter, the following questions were identified:

- What are the easier forms for student collection, storage, and referenced by the system?

• How can information about students be collected and stored for future reference?

• How can the system reduce errors as compared to the manual system?

Research Questions

• Does inadequate hostel accommodation affect student study habits?

• Are students happy to be accommodated outside the two campuses?

• Does inadequate hostel accommodation affect student study habits?

1.6 SYSTEM OBJECTIVES

- Reduce costs such a labor.
- Make it easier for the engineering team to receive and respond to customer feedback.
- Improve performance reviews of the hostels.

1.7 SCOPE OF THE STUDY

The Hostel Management System project is an interface to keep information in an orderly way to replace the old paperwork system. The objective of this project is to automate the management of the hostel for its smooth operation by automating almost the whole business. Updates and changes are easy to implement, and both estimates and accountability activities are reliable.

This system is a Desktop application which will capture all the management activities involved with the hostels. The administrator module is concerned with all administration details, while the student module is concerned with management of student details in regards to the hostel. Students are allocated rooms based on their availability. The students' module enables them to apply for a room and manage their details. They then check their application status to check whether a room has been allocated to them. The systems database will capture both the administrator details and the students' details.

Different tasks done by the management team will be automated and integrated into the system; for example, keeping record of all transactions done by the hostel or students; check-in or checkout of the hostel and other hostel-related tasks.

Out of Scope

The following features will not be delivered by the system: Employee Payroll, Inventory Management, Resident attendance, and Accounting Details

8 JUSTIFICATION

Prior to 2010, most Hostels operated in a fairly traditional manner. More than 2/3 of the Hostels in Kenya have no computerized system put in place to help manage them. The number of institutions of higher learning has increased in the last few years; so, has the number of students, according to government figures. The number of hostels to accommodate students has also increased rapidly. The stuff that runs the student hostels usually have a hard time managing the hostels. With the Hostel Management System, the hostel stuff can have an easy time managing the hostels and any operations related to the hostels.

1.9 Project Assumptions

- It is assumed that the system developed will work perfectly; it is going to be developed under the Windows OS and MySQL database.
- Personnel costs will not change during the project cycle.
- The overall cost of day-to-day operations will not increase.

- The scope of the project will not change throughout the life cycle.

1.1.0 Project constraints

Quality constraint

- If the project scope extends due to scope creep, I may not have the time or resources to deliver the promised quality.
- If delivery time is cut or rushed, project costs may rise and quality will very likely decline.
- If you are unable to meet a sudden rise in cost, the project scope may shrink and the quality may decline.

Cost constraint

- Budget and materials: as limited by teacher and school
- Budget Constraint: Due to a limited budget, HMS is intended to be very simple and just for basic functionalities. The UI is going to be very simple.

Implementation Constraint

- Applications should be based on Python only, and the database on MySQL.

1.1.1 LIMITATIONS

The limitation of this project includes the difficulties encountered in getting information and relevant fact about the existing system from the inquisitive and critical management and staff in the hotel.

Some of the other constraints encountered during this design project include the following:

- Financial Constraints: The design was achieved but not without some financial involvement. One had to pay for the computer time. Also, the typing and planning of the work have its own financial involvement.
- High programming Technique: The programming aspect of this project posed a lot of problematic bugs that took us some days to solve. Problems such as database connections using Python and MySQL databases posed a lot of challenges.
- Few Literature Sources: The topic, though, seems to be a common term; it is not a popular topic to surf on the Internet. It had fewer literature sources.

1.1.2 GLOSSARY

Short Name	Description
HSM	Hostel Management System
Administrator	is responsible to create a new allotment of room, delete reserved rooms. The person who controls the system
Resident	Student living in the hostels
UML	Unified Modeling Language
ERD	Entity Relationship Diagram
DFD Data Flow Diagram	Data Flow Diagram
MySQL	My Structure Query Language

1.1.3 Reference:

1) Shaw m,2002-international journal of technology

2) Ruso A.s.Easterbrook and Nuisebeh (2001)journal of system and software

3) Muhammed Shaheer,KA,Muhammad Shiras-(HMS 2009)

4) College Hostel Management Software by Initio (2010).

5) Loventis Booking System by Loventis Systems (2005).

6) Indocon Hostel Management Software by Indocon Micro Engineers Limited

7) IEEE Standards 1016-1998, Recommended Practice for Software Design Descriptions

CHAPTER TWO: LITERATURE REVIEW

In the literature review we consider and examine the work done by other scholars and researchers who have broached this particular topic of Hostel Management Systems. The objective of this literature review is to analyze the work related to this project and the mechanisms used in previous studies. This chapter briefly describes the review on existing technique related with "Hostel allocation management system" that will be developed later

2.1 INTRODUCTION

Technology has made a considerable impact on many industries in recent years and will continue to do so with the increasing use of computers, controlled equipment and the growth of information technology in general. In the last two decades, technology has become far more advanced and far more widely used throughout all types of industry, and the hospitality industry is no exception. Indeed, many hostels' establishments rely on technological systems for the vast majority of their operations. They use a range of computer programs, from everything to bookings, communications, security, and payments. If a hostel establishment does not use any sort of advanced technological system in its operations, it is deemed to be out of date and disorganized.

Part of the reason why hostels utilize technological systems in their operations is because it keeps them up to date in terms of where they are placed in the market. It makes work easier for staff members, allowing them to work more efficiently and taking away time-consuming activities which can be carried out by technology. In some hostels, the utilization of technological systems means that fewer staff members are needed and this saves considerable costs.

With this in mind, hostels and hostel managers can implement hostel management systems through which they are able to monitor and manage operations related to the hostels and react to this information. This Hostel Management System was developed in favor of the hostel management team, which helps them to save the records of the students about their rooms and other things. It helps them with the manual work, from which it is very difficult to find the records of the students and the information about those who had left the hostel years before.

The development of the hostel management system seeks to solve the challenges that are experienced by the current manual system. Some of the challenges are redundancy, data security, too much paper work, data retrieval and inaccuracy. This proposed system will computerize most of the operations in the Tuk and other organizations with hostels.

In an attempt to review existing literature, I came across a number of similar products that are in use in many colleges worldwide. Some of them are described below:

> Microbes Hostel system is another software product that automates the hostel facility management exercise. It has several compelling features, like powerful reservation management, synchronization of computers, reception and cash box administration, point of sale, accounts statistics and reports.
>
> The Loventis booking system is another innovation from Loventis systems (2005). It has features like a property management system (PMS), channel manager and booking engine, plug and play.
>
> College Hostel Management Software developed by Initio has six modules, such as the library module, the transport module, the hostel module, the inventory/store module, the enquiry module and the visitors tracking module. It offers information on the building, rooms, and students. .

According to (Stein,2006) an organization or institution can't be competitive or successful today's business without a computer system that can make operations efficient and easy to run. Automated systems reduce the workload and paperwork experienced by the current manual system. Its clear that computerized systems are needed in institutions and organizations to make operations efficient and improve customer experience.

Despite advancements in technology, there are institutions and organizations which still use manual systems to keep records and other operations in the hostel. Though (**Griffin,2011**) stated that many organizations can still function quite well with manual systems using paper documents, it's/ still considered high risk and inaccurate. Therefore, the hostel management system will automate the current system to reduce these high risks and inaccuracies.

The difference between computer and manual execution of jobs as stated by (**Watford,1998**) is that computers perform tasks much faster than humans. To stress this (**O'Connor,2000**) describes that the accuracy of computers greatly reduces mistakes and maintains high accuracy level; hence customer satisfaction and easy control of hostel operations. According to (**O'Connor,2000**) the computer systems have greater benefits and advantages than the manual system because of its mode record keeping, speeding, no repetition of data and accuracy. As per (**Griffin 2011**) and (**O'Connor 2000**) it's evident that computer systems are more accurate and appropriate compared to human or manual systems.

This proposed system seeks to change this scenario. Some of the benefits of this new system compared to the manual system are:

1) The lower cost of management that comes with this system will be designed to reduce the cost of paperwork and reduce the number of people managing staff.

2) The system will offer a common platform for admins to share information and notices regarding the exams and admission by allowing the management to create a virtual notice board.

3) Increased transparency This proposed system would ensure transparency of records and data maintenance.

After the development of a system, testing is one of the stages it undergoes. (Farrow,2010) elaborated that system software testing provides a "status report of the actual product in comparison with the product requirement". Testing a system verifies if the system really fulfills the intended purpose, checks for errors, and makes sure it meets the required quality.

O'Brien, (2002) recommends the following steps while developing a hostel allocation management system:

> Review the existing system, defining the data needed for relevant units within organization, determine the most appropriate and effective data flow, Design the data collection and reporting tools, Develop the procedures and mechanisms for data processing, Develop and implement a training program for program for data provides and data users Pre-test, and if necessary re-design the system for data collection(,n. data flow, data processing and data utilization, Monitor and evaluate the system, Develop effective data dissemination and feedback mechanisms and Evaluate the system

2.2 EXISTING SYSTEM

This section is to review the current system and existing system that related to hostelmanagement system. These systems involve the use of pens, pieces of papers to manage, control and capture all students' details and allotment. In spite the fact that paper based systems are cheap and seem to be easy to use by many people, these systems arc time consuming, laborious, not accurate,less information back up or security, and prone to human errors.(**www.infocntrcprcncurs.org/)**

The existing system is manual based and needs a lot of effort and it consumes a lot of time. In the existing system, we can apply for the hostels online but the allotment processes are done manually. It may lead to corruption in the allocation process as well as hostel fee calculation. The existing system does not deal with mess calculation and complaint registration.

Miss AzlindaBinti Alias2009 defines hostel management system as a system specialty designed to centrally manage hostel Allocation. This system is standalone system. It is customize and user friendly software for hostel. All administrative and application system data has been designed to be kept centrally and unique for entire population. Hostel Management System (HMS) is a system which helps in managing various activities in the

hostel.

Disadvantages of the existing system include.

- More human power
- More strength and strain of manual labour needed
- Repetition of same procedure.
- Low security.
- Data redundancy.
- Difficulty to handle.
- Difficulty to update data.
- Record keeping is difficult.
- Backup data can be easily generated.

2.3 PROPOSED SYSTEM

This project is aimed at developing a system for keeping records and showing information about or in a hostel. This system will help the hostel officer to be able to manage the affairs of the hostel. This system will provide full information about a student in the hostel. It will show rooms available or not, and the number of people in a particular room. This will also provide information on students who have paid in full or are still owing. This system will also provide a report on the summary details regarding fees and bills students are owing. Also included is a user module for employees or the hostel officer. There will also be an administrator module which will be accessed by the administrator and has the ability to delete, add and edit employee records.

This system will be developed based on the Software Development Life Cycle (SDLC) with Python and MySQL database.

This solution was developed based on the plight of the hostel management team. Through this, they do not require so efficient people to handle and manage the affairs of the students in the hostel. All you need to do is to login as administrator and you can see the information of all the students who have obtained and registered their hostel form, click verify to ascertain their eligibility and allocate them to the available hostel.

Identification of the problems of the existing hostel management leads to the development of a computerized solution that will be compatible with the existing hostel management with a solution which is more user-friendly and more GUI-oriented. We can improve the efficiency of the hostel management, thus overcoming the drawbacks of the existing management.

Python Tkinter module is used as the front end tool and MySQL is used as the backend tool. Python is one of the driven programming languages. The application wizards, menu editor, and data reports, etc is very useful for creating very good professional software.

2.4 CONCLUSION

In conclusion many scholars have done publication about allocation and management system as a more convenient way and an urgent requirement and a lot of research has been carried out though implementation is still insufficient. However, according to the literature available, there are numerous benefits that accrue from an allocation and management system when compared with manual systems. For example there will be no duplication of records, sharing of information is made possible, the problem of missing and misplaced records is reduced and the information is available at any time. In order to continually improve the quality of services, Kampala International University Western Campus need to put in place a hostel allocation and management system backed up with electronic databases replacing all existing paper-based allocation system. The following chapter of methodology gives the steps of how the system is go111g to be developed.

CHAPTER THREE: FEASIBILITY STUDY

The purpose of this report to study that the particular parameters which are necessary to make the hostel system.

3.1 TECHNICAL FGEASIBLITY

The technical feasibility in the proposed system deals with the technology used in the system. It deals with the hardware and software used in the system, whether they are of the latest technology or not, and if it happens that after a system is prepared, a new technology arises, and the user wants the system based on that technology. This system uses Windows OS platform, MySQL for database, Python as the language, and python 'mysql-connector' as the user interface. Thus, HOSTEL MANAGEMENT SYSTEM is technically feasible.

3.2 ECONOMIC FEASIBILTY

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis. Php, html, xml and sql database are easily available on internet.

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis. Python, and MySQL database are easily available on the internet. Thus, this project is Economically Feasible.

3.3 OPERATIONAL FEASIBILTY

The project has been developed in such a way that it becomes very easy, even for a person with little computer knowledge to operate it. This software is very user-friendly and does not require any technical person to operate .Thus, the project is even operationally feasible.

3.4 STSTEM MODULES

USER MODULE

This helps the administrator and user to login to homepage only if password and usernamematches

CHANGE PASSWORD MODULE

Allows the user to change the password

STUDENT MODULE

This module is used to store student details i.e. information like profile details, contactinformation, educational details etc. Users can search according different criteria such asname, course, room number etc

ROOM ALLOTMENT MODULE

This deals with allocation of room to students according to education details, section orcourse. Rooms will be allocated to students and an ID will be generated for it. It will displaydetails students staying in the room or rooms. When a student leaves the room after thesemester, the left date will be also saved

VISITORS MODULE

This allows the visitor details to view the visitors depending on various search criteria

REPORT GENERATION MODULE

This is provided to view summary detail regarding hostel

GUEST MODULE

This section is to enable guest to request for booking, and inquire more information about the Hostels and services they offer

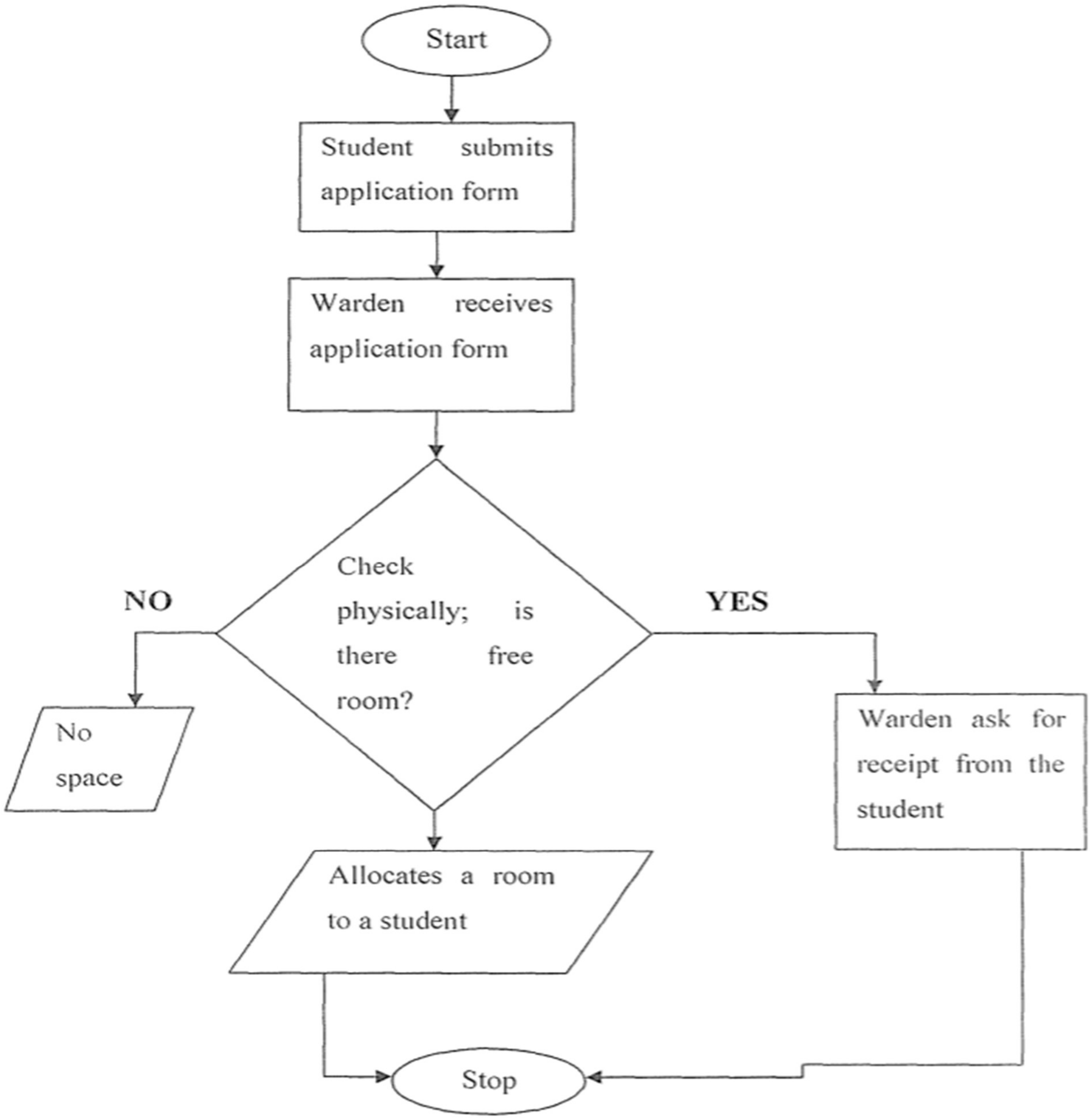
CHAPTER FOUR: SYSTEM ANALYSIS AND DESIGN

4.1 SYSTEM ANALYSIS

4.1.0 ANALYSIS OF THE CURRENT SYSTEM

The current system uses a lot of paperwork which takes much time as compared to other methods of keeping information. It also makes it difficult for the users to retrieve a particular file because one has to go through all the files first. Congestion was also another weakness since files kept increasing as the number of students also was increasing and this resulted into need of more shelves which increased congestion in the office. Another weakness was loss of some files because as files increase in number, the old ones are seen as useless hence misused. The current system was insecure in terms of data access and confidentiality. This was more so because there mare no strict measures implemented that denies unauthorized entities from accessing the information and this led to data being inaccurate.

The flow chart of the existing hostel allocation management system


Start
Student submits application form
Warden receives application form
Check physically; is there free room?
NO
YES
No space
Warden ask for receipt from the student
Allocates a room to a student
Stop

4.1.1 USER REQUIREMENTS

The staffs in charge of students' hostel allocation and management are the primary users of the system in fact "Administrators". Each user is required to have the username and password that is authenticates first before gaining access into the system. Depending on their level of privileges, they perform different activities as follows: Administrator users (add information, views all the information, makes modification, clear all-in formation, update a student and generate reports)

4.1.1 REQUIREMENT ANALYSIS AND SPECIFICATION

Requirement analysis is the process of looking into software specifications which intends to communicate the customer needs to system developers.so this will deal with resources to be used in building this system.

Functions and features delivered to end users. The end users of the proposed system are:

- User Module: This helps the administrator and user to login to the homepage only if password and username match.
- Change password module: Allows the user to change the password.
- Student module: This module is used to store student details, i.e. information like profile details, contact information, educational details etc. Users can search according to different criteria such as name, course, room number etc.
- Room allotment module: This deals with allocation of room to students according to education details, section, or course. Rooms will be allocated to students and an ID will be generated for it. It will display details of students staying in the room or rooms. When a student leaves the room after the semester, the left date will also be saved.
- Room fees module: This displays fee records, student dues status, and balance amount status. It is also used to renew students' rent every semester.
- Visitors module: This allows the visitor details to view the visitors depending on various search criteria.
- Report generation module: This is provided to view summary details regarding hostel fees and bills. Students can check hostel fees and bill details by entering the unique hostel ID.
- Settings module: In this module, only the administrator can access it. The administrator has a unique account with a lot of special access and permissions over normal users. The module allows adding, editing, deleting and employee records, building block information, room details, course details etc.

4.1.2 HARDWARE CONFIGURATION

The section of hardware configuration is an important task related to software development; insufficient random-access memory may adversely affect the speed and efficiency of the entire system. The process should be effective to handle the entire operation. The hard disk should have sufficient capacity to store the files and applications.

Processor :	Pentium IV and above
Processor speed :	1.4 GHz Onwards
System memory :	128 Mb minimum 256 Mb recommended
Cache size :	512 KB
RAM :	512 MB(Minimum)
Hard disk :	80Gb minimum
Monitor :	SVGA Color 15”
Code Editor:	PyCharm code editor
Operating System:	Microsoft Windows 8, 10, 11

4.1.3 SOFTWARE CONFIGURATION

This section gives a detailed description of the software requirement specification. The study of requirement specifications is focused specially on the functioning of the system. It allows the developer or analyst to understand the system, functions to be carried out the performance level to be obtained and corresponding interfaces to be established.

Front end tool:	Python Tkinter Module.
Backend:	Python scripting language.
Database:	MySQL (workbench)
Database Connector:	Python “MySQL Connector”
Operating system:	Windows OS (64-bit machine)

The software requirement is classified into 2

1)Functional

2)non functiona

4.1.4 FUNCTIONAL SYSTEN REQUIREMENT

These are specific functions, tasks or behaviors the system must support. They include the following: -

- The system allows the management to capture and store a particular student's details
- It enables management to view and search a particular file within a short time
- it authenticates users
- The system does enable the users to update, delete and save information about the student. Therefore, the system has the ability to permanently save data into the database.
- The system does generate necessary reports.

4.1.5 NON-FUNCTIONAL REQUIREMENT

The system does authenticate users and provide different levels of access to avoid unauthorized access.

The system does perform in a standard relative to the activities carried out within the office. In other words, it retrieves information very fast.

The system should be easy for users to learn and use with dynamic access of information by both skilled and unskilled users.

The system is cost effective with less effect on its implementation and maintenance.

The system is portable and light in order not to affect the throughput system does not require a lot of storage space

-Performance Requirements

• The load time for user interface screens should take no longer than two seconds.

• The log-in information shall be verified within five seconds.

• Queries shall return results within five seconds

• Data in database should be updated within 2 seconds

• Query results must return results within 5 seconds

• Load time of UI should not take more than 2 seconds

• Login Validation should be done within 3 seconds

• Response to user inquiry must be done within 5 minutes.

• The database shall be able to accommodate a thousand records to store.

• The software shall support the use of multiple users at a time.

• There are no other specific performance requirements that will affect development

Reliability

• The system database connectivity has been designed with a persistent connection to ensure system reliability. The system runs on a dedicated server to ensure that it is reliable at all times

Availability

• The system is available at all times, but a room application is open at the beginning of every semester to ensure that rooms are available.

• Report should be generated automatically every day for manager and anytime upon request

Security

• The system has an authorization mechanism for users to identify their personal profiles. Therefore, different users will have different authorization levels to access the data. The system should utilize certain cryptographic techniques, for instance the SHA1 algorithm, for encrypting passwords.

• All external communications between the data's server and client must be encrypted.

• All data must be stored and protected.

Maintainability

• The system is developed using Python, and Python is a cross-platform programming language which is easy to maintain.

Portability

● The HMS runs on any desktop computer or laptop

Safety Requirements

• Database should be backed up every hour.
• Under failure, the system should be able to come back to normal operation in under an hour.

4.1.6 SYSTEM METHODOLOGY

In this system we are use waterfall model. The waterfall model helps us to separate each step and when we finish a one phase the output of it is the input to the next phase. Also, we can backwards if there is a new requirement or to apply any update.

The steps of waterfall model are:

1. Requirement Definition
2. System and Software Design
3. Implementation
4. Integration and System Testing
5. Operation and Maintenance

4.2 SYSTEM DESIGN

4.2.0 INTRODUCTION

System design gives the details of how the system will meet the information requirements determined in the system analysis, it is the blueprint for the Hostel Management System that is to be developed. It gives the very detail about every component of the system that is to be built. The various procedures use for the new system is given here; i.e., how to, what to, and do, what shall the system be used on. The importance of the design is to enable the system designer or researcher to know the cost consequence of the product on the user and the developer. In that, the effectiveness of the system will not be obsolete.

In this chapter we will introduce UML diagrams, HMS system architecture, principal system object, design models, and object interfaces.

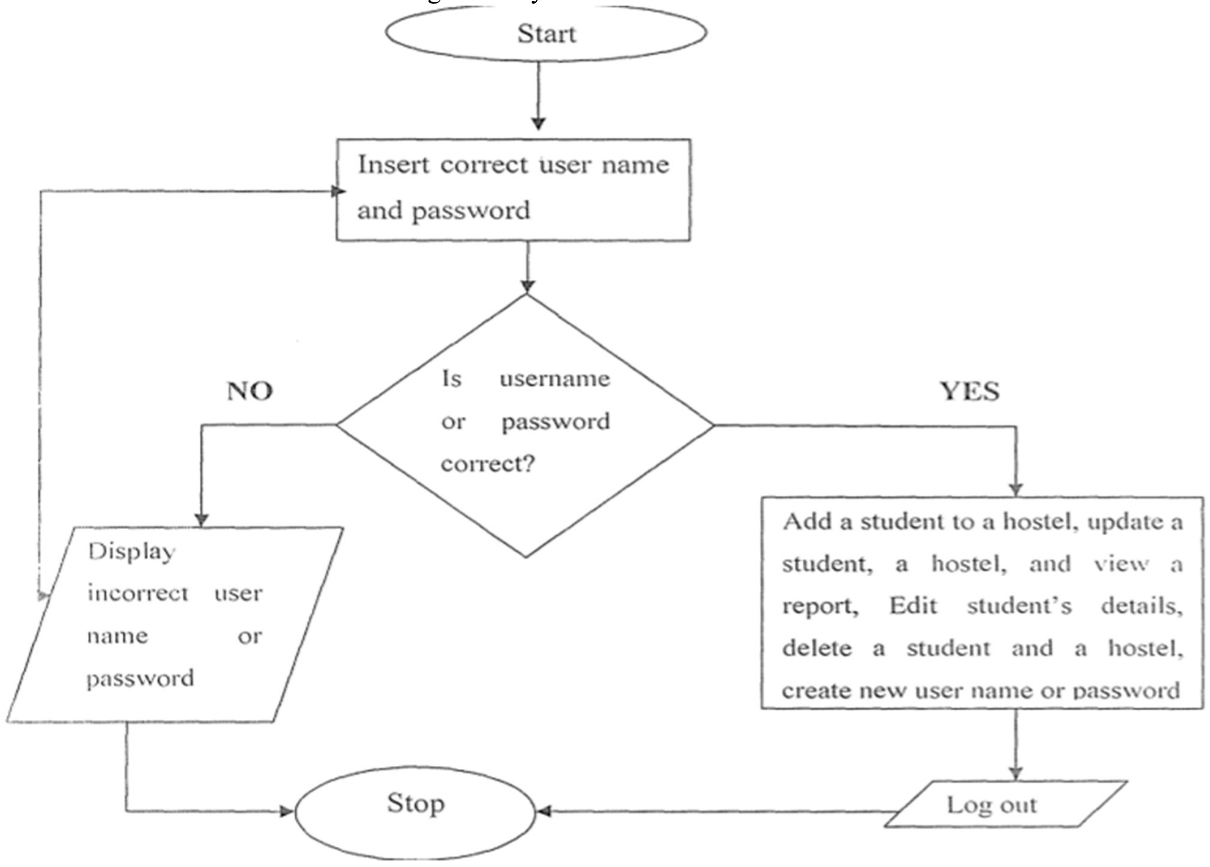
The project used two techniques to design the system. These techniques are logical design and physical design

4.2.1 DESIGN OBJECTIVE

The systems design meets the following objectives:

- Produce the system that overcomes the inefficiencies identified during the system study and
- optimally uses computer resources.
- Allow accurate input of correct and valid data in the database.
- Creates a user-friendly system with acceptable, clear and easy to use interfaces for the users of average computer knowledge

A flow Chart of a Hostel Allocation Management System

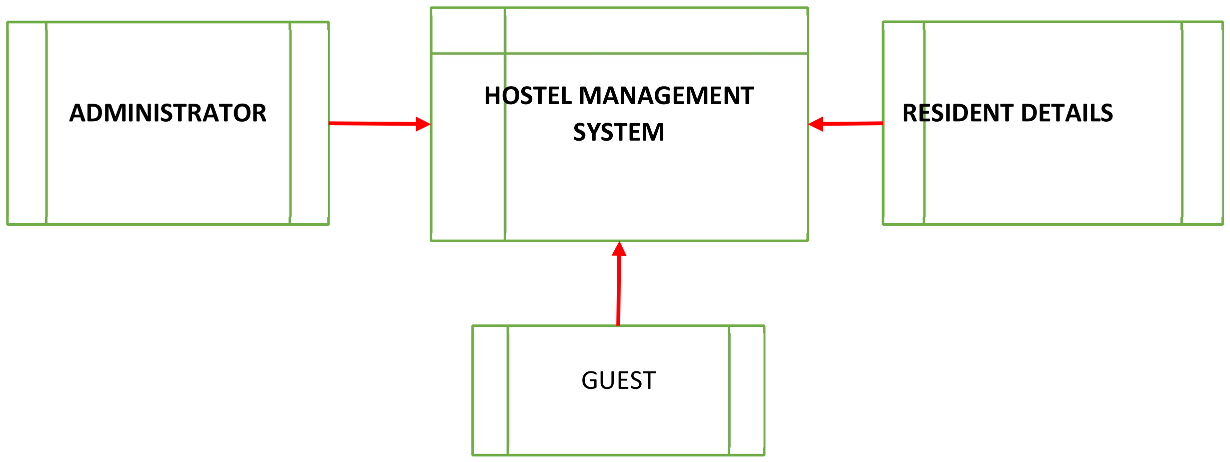

Start
Insert correct user name and password
Is username or password correct?
NO
YES
Display incorrect user name or password
Add a student to a hostel, update a student, a hostel, and view a report, Edit student's details, delete a student and a hostel, create new user name or password
Stop
Log out

4.2.2 DATAFLOW DIAGRAM

Data flow diagrams are used to illustrate how data are processed by a system in terms of inputs and outputs; what activities are occurring to fulfill a business relationship or accomplish a business task, not how these activities are to be performed. It shows the logical sequence of associations and activities, not physical processes.

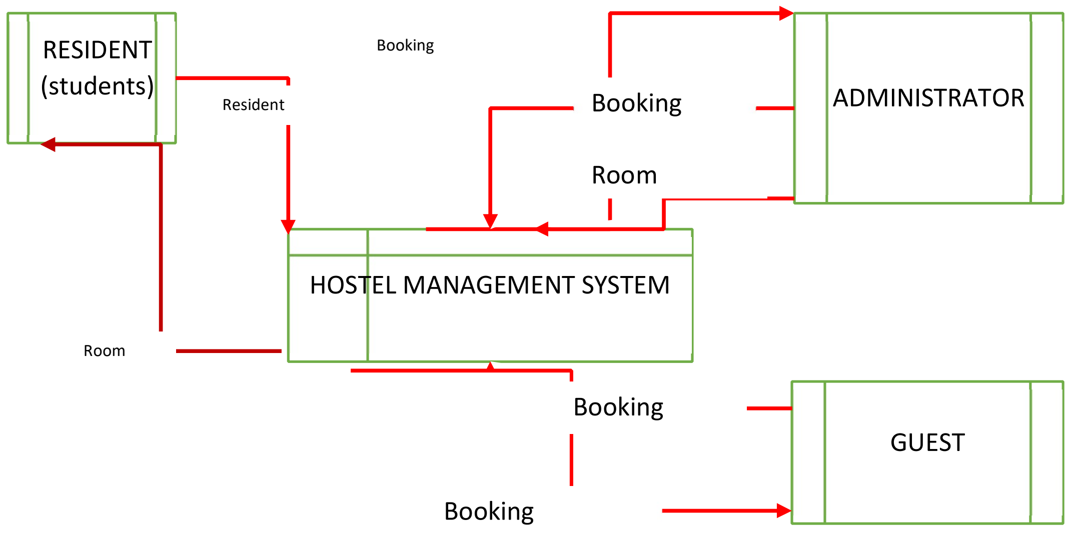
Context Dataflow Diagram

This diagram represents the boundaries and scope are of the Hostel Management System project. It describes the main objective of the system and its entities involved.

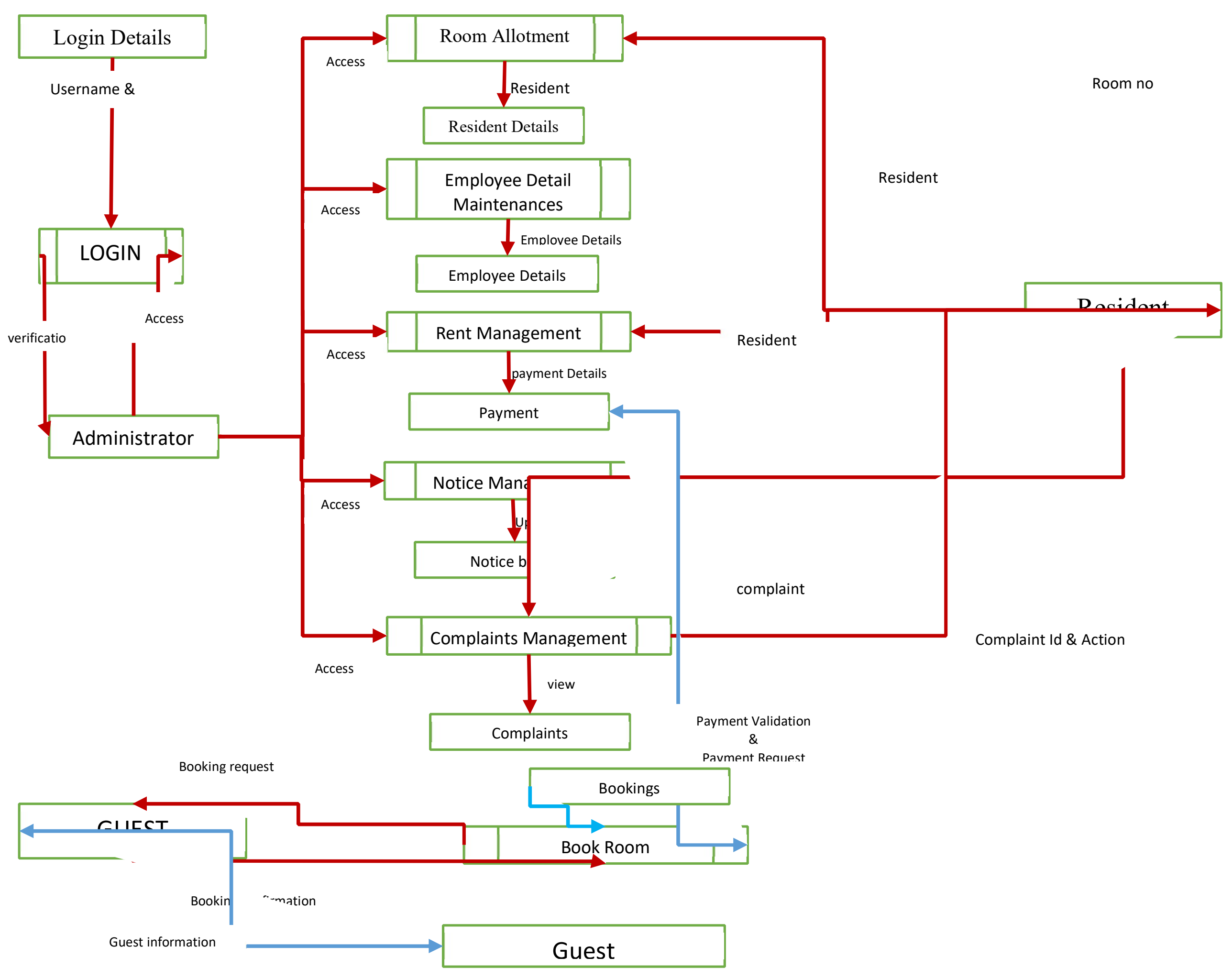


LEVEL 1 DFD

Level 1 DFD is an expansion of the context diagram that shows more processes and how they interact with the system in terms of inputs and outputs.



LEVEL 2 DFD



4.2.3 USE CASE MODE

The USE-CASE diagrams describe what a system does from the standpoint of an external observer. The emphasis of use case diagrams is on what a system does rather than how. They are used to show interactions between users of the system and the system. A use case represents the various users called actors and the different ways in which they interact with the system. various

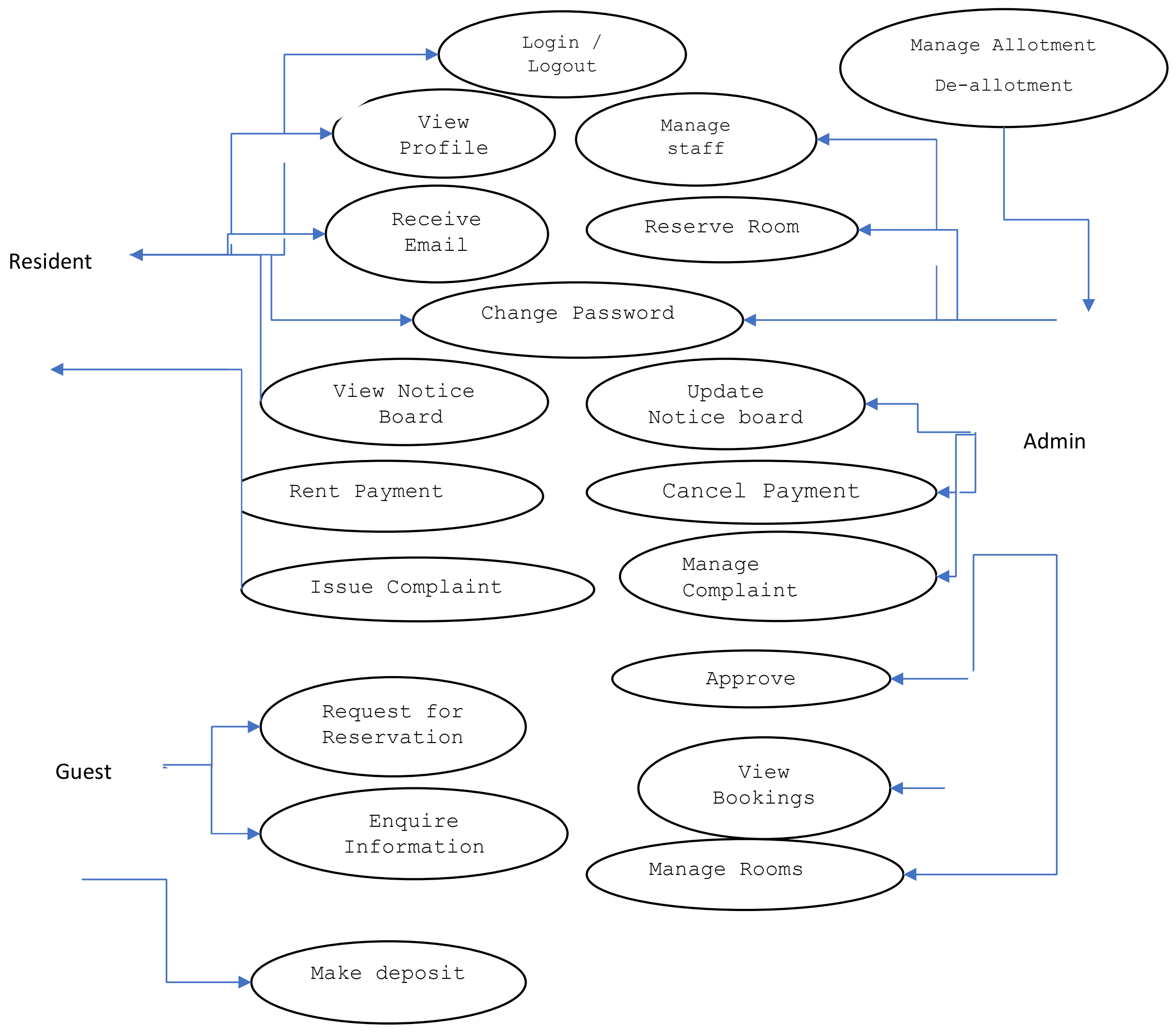
Actors

- Resident
- Admin
- Guests

Use Cases

- Apply
- Setup hostels
- rooms
- View status
- Delete setups
- View applications
- Change password
- Login

Below is the diagram of the use case for the proposed system



Use Case Name	Login
Introduction	This use case describes how a user logs into the HMS.
Actors	Administrator and Student
Pre-Conditions	None

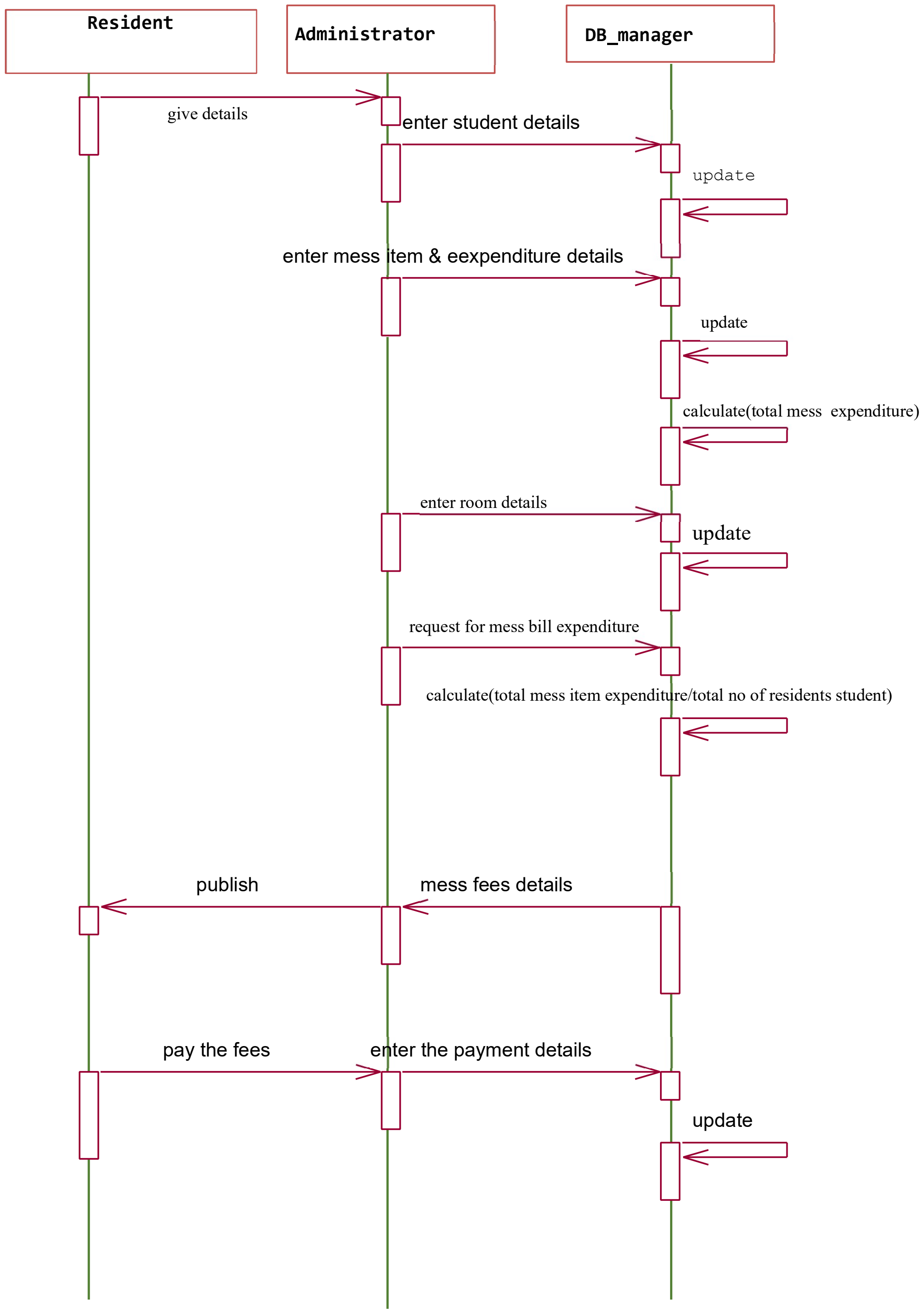
Basic Flow	This use case starts when the actor wishes to login to the HMS which requests that the actor enter his/her username and password. The actor enters his/her username & password. System validates username and password, and if finds correct allow the actor to log into the system.
Alternate Flow	Invalid name and password. If in the basic flow, the actor enters an invalid name and/or password, the system displays an error message. The actor can choose to either return to the beginning of the basic flow or cancel the login, at that point, the use case ends.
Post-Condition	If the use case is successful, the actor is logged into the system. If not, the system state is unchanged

Use Case Name	Manage Student Activities
Introduction	Allows the student to manage their various activities. This includes View Profile, Apply Room, View Status of Application, View History, and Change Password
Actors	Student.
Pre-Conditions	Student must be logged into the system.
Post-Condition	If use case is successful, the student can view their profiles, apply room, view status of application, view history and change password. Otherwise, the system state is unchanged.
Basic Flow	Starts when student wishes to view Profile, Apply Room, view Status of Application, and Change Password. The system requests the student to specify the function; he/she would like to perform.
Alternate Flow	The system displays an error message if student information is unavailable

‘

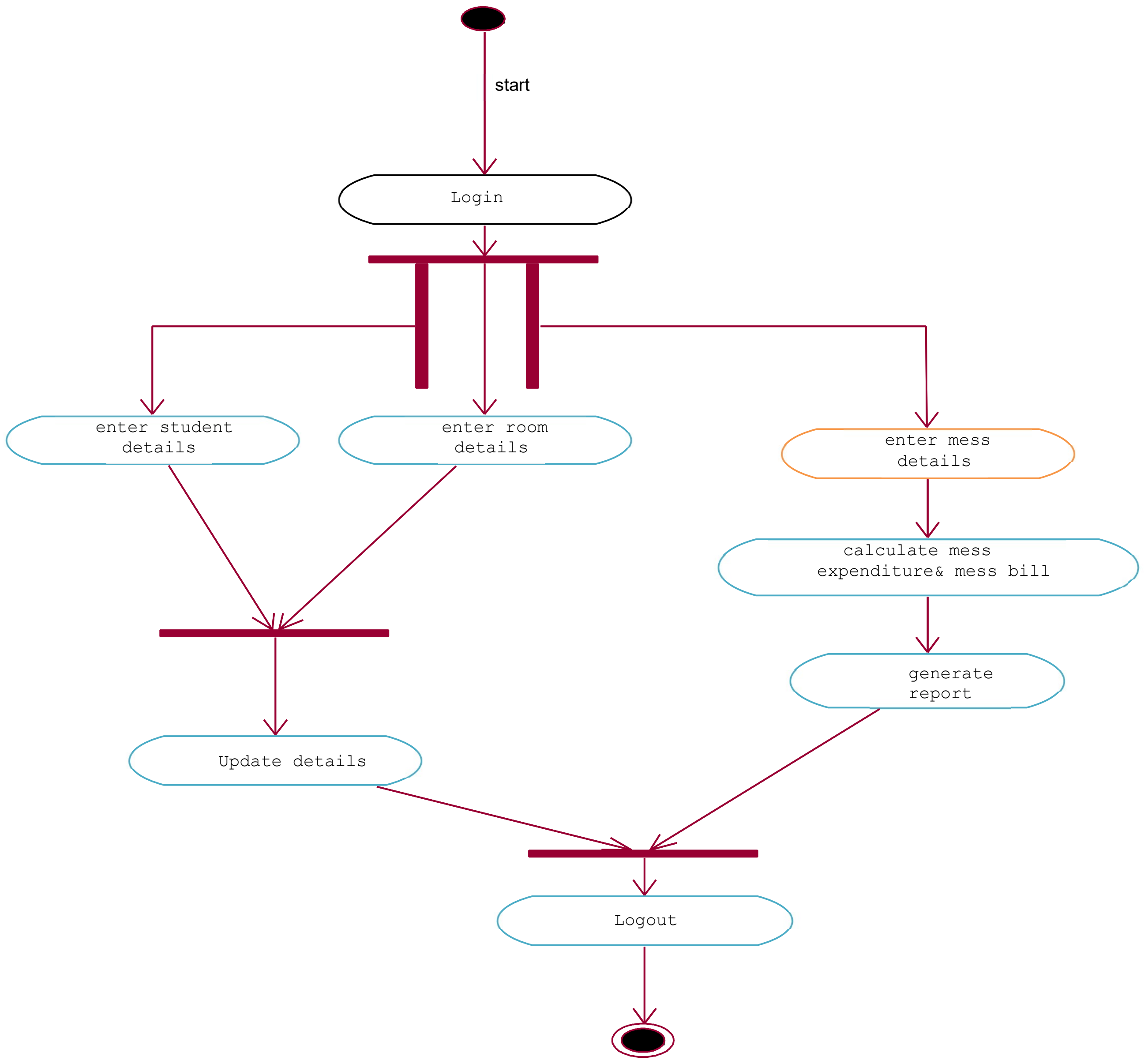
4.2.4 SEQUENCE DIAGRAM

The sequence diagram shows an interaction arranged in a time sequence. It is an alternate way to understand the overall flow of the control of the system program.


Resident
Administrator
DB_manager
give details
enter student details
update
enter mess item & eexpenditure details
update
calculate(total mess expenditure)
enter room details
update
request for mess bill expenditure
calculate(total mess item expenditure/total no of residents student)
publish
mess fees details
pay the fees
enter the payment details
update

4.2.5 ACTIVITY DIAGRAM

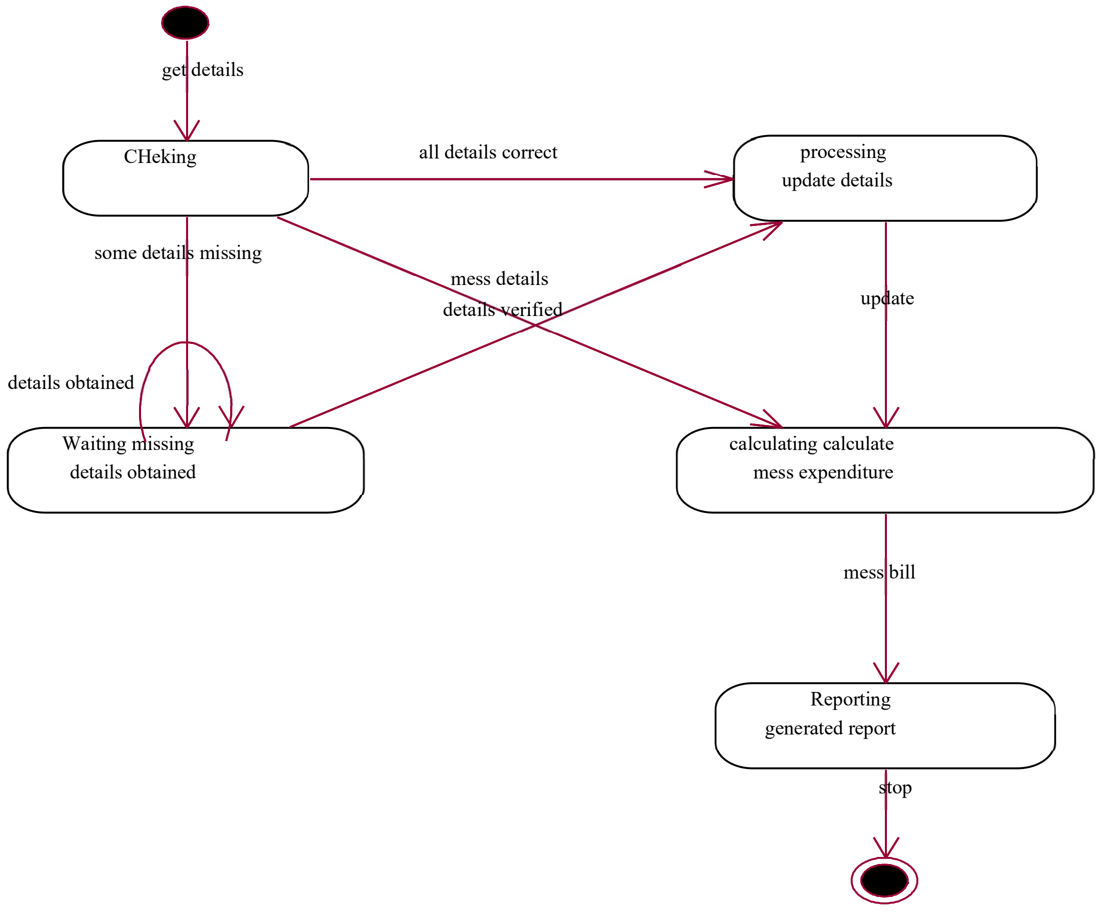
It describes the sequence of activity it supports for conditional and parallel behavior. It is a variant of a state diagram in which most of the states are activity states.



-

4.2.6. STATE DIAGRAM

State diagrams are a familiar technique for describing the behavior of a system. They describe all of the possible states that a particular object can get into and how the object's state changes as a result of events that reach the object.

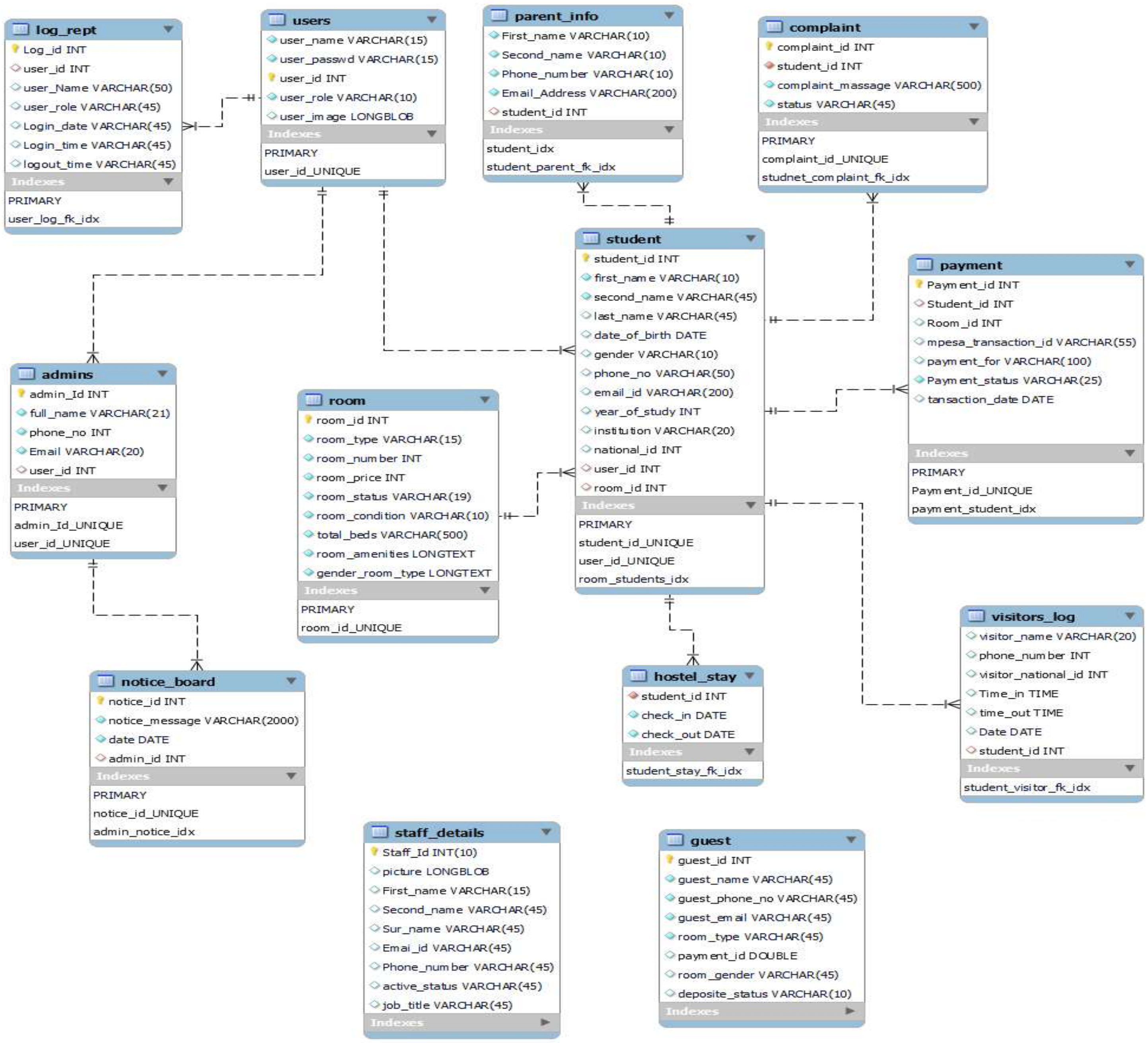


4.3 DATABASE DESIGN

The database design composed of the following objects: tables, security, indexes, keys, fields and data types

4.3.1 Entity Relationship Model

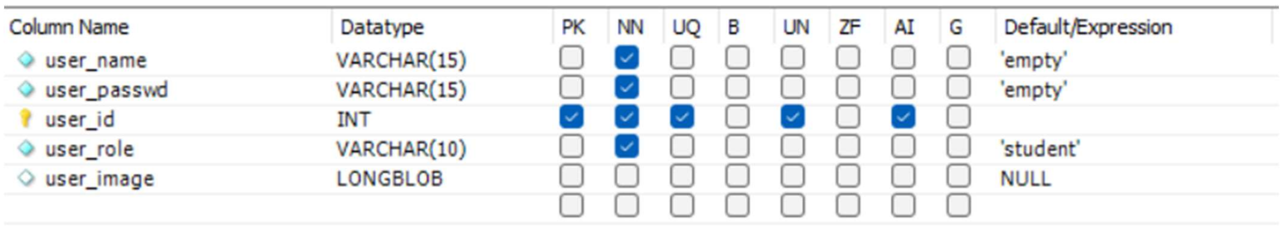
An entity relationship model (ER model) is a systematic way of describing and defining a business process. The process is modeled as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties(attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity relationship diagrams. An ER model is typically implemented as a database. In the case of a relational database, which stores data in tables, every row of each of table represents one instance of an entity. Some data fields in these tables point to indexes in other tables; such pointers represent the relationship



4.3.2 Table Specification

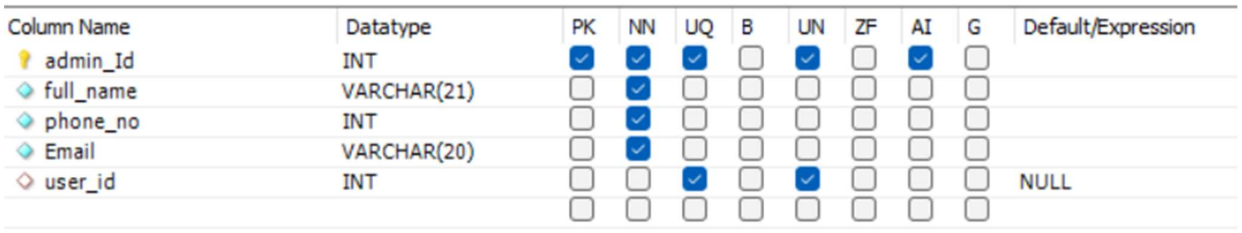
User Table

This table holds the password that is used as the security measures. Simple changing the user's name and password in the table makes a change to the password. The program starts up by reading this table to confirm authorization to access the package and disallow access if password is in valid. And if a user name and password are valid, gives you access to the module according to your user role.



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
user_name	VARCHAR(15)		✓							'empty'
user_passwd	VARCHAR(15)		✓							'empty'
user_id	INT	✓	✓	✓		✓		✓		
user_role	VARCHAR(10)		✓							'student'
user_image	LONGBLOB									NULL

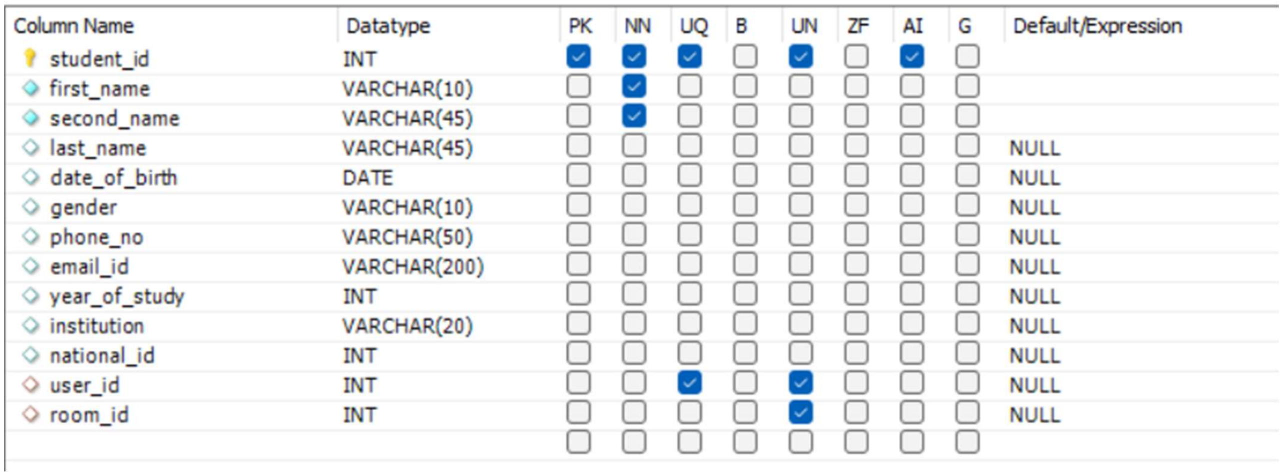
Admin Table



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
admin_Id	INT	✓	✓	✓		✓		✓		
full_name	VARCHAR(21)		✓							
phone_no	INT		✓							
Email	VARCHAR(20)		✓							
user_id	INT			✓		✓				NULL

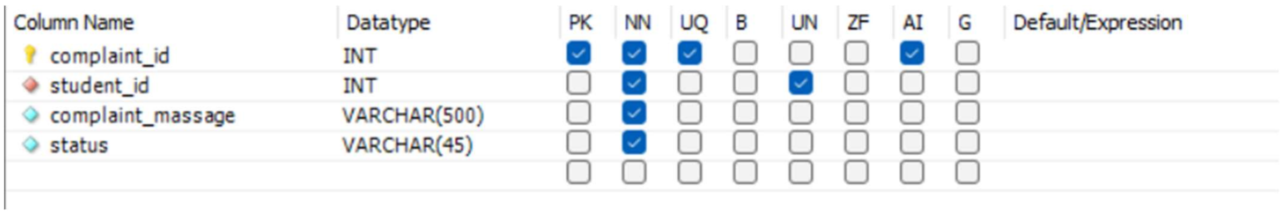
Student Table

This table holds the information of the bona-fide students who are allocated rooms.



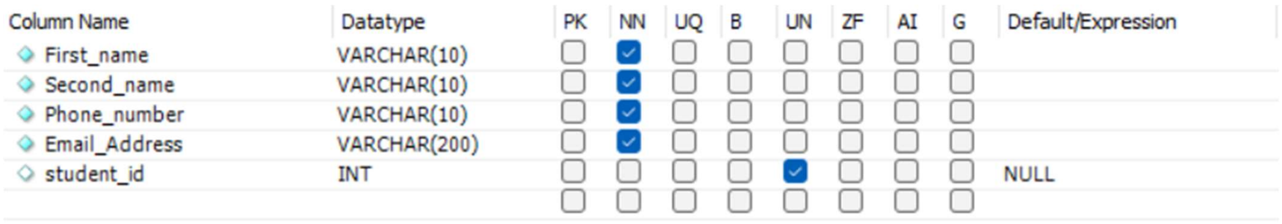
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
student_id	INT	✓	✓	✓		✓		✓		
first_name	VARCHAR(10)		✓							
second_name	VARCHAR(45)		✓							
last_name	VARCHAR(45)									NULL
date_of_birth	DATE									NULL
gender	VARCHAR(10)									NULL
phone_no	VARCHAR(50)									NULL
email_id	VARCHAR(200)									NULL
year_of_study	INT									NULL
institution	VARCHAR(20)									NULL
national_id	INT									NULL
user_id	INT			✓		✓				NULL
room_id	INT					✓				NULL

Complaint Table



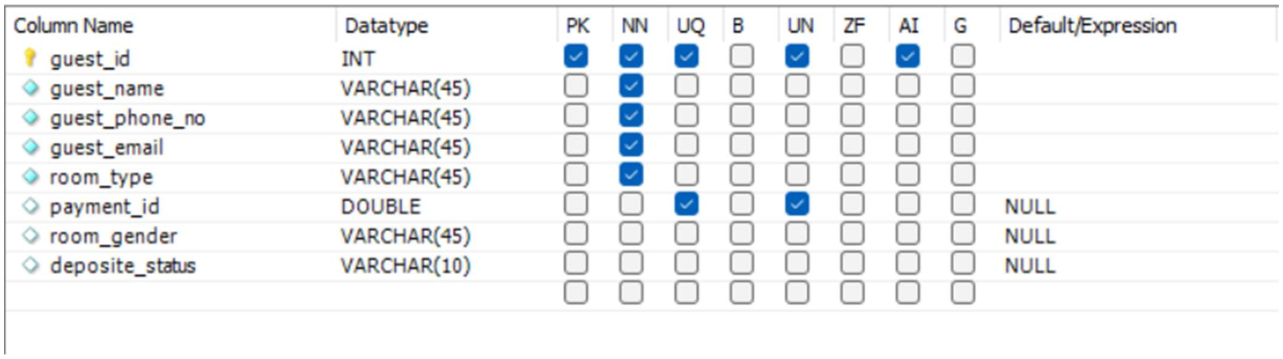
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
complaint_id	INT	✓	✓	✓				✓		
student_id	INT		✓			✓				
complaint_massage	VARCHAR(500)		✓							
status	VARCHAR(45)		✓							

Parent Table



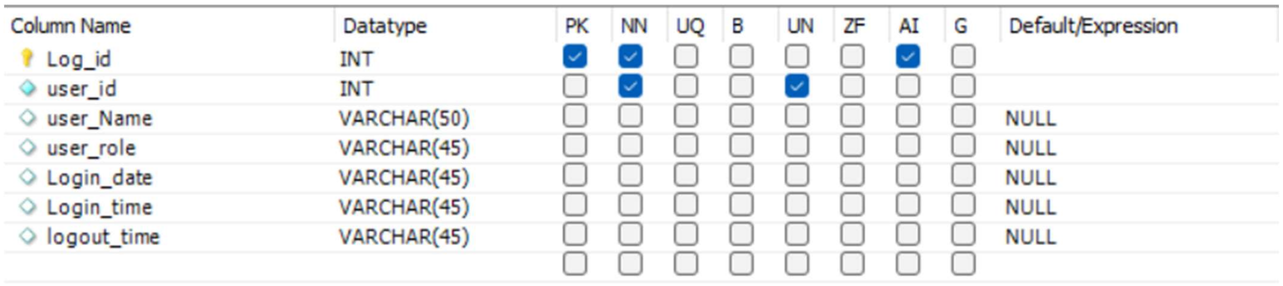
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
First_name	VARCHAR(10)		✓							
Second_name	VARCHAR(10)		✓							
Phone_number	VARCHAR(10)		✓							
Email_Address	VARCHAR(200)		✓							
student_id	INT					✓				NULL

Guest Table



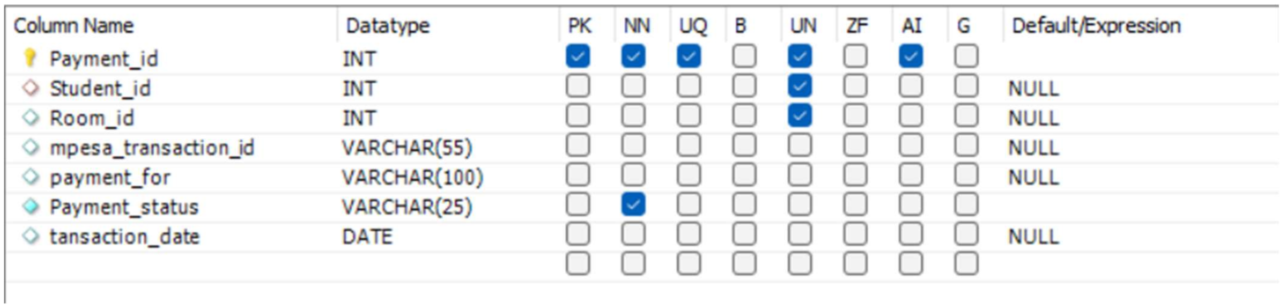
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
guest_id	INT	✓	✓	✓		✓		✓		
guest_name	VARCHAR(45)		✓							
guest_phone_no	VARCHAR(45)		✓							
guest_email	VARCHAR(45)		✓							
room_type	VARCHAR(45)		✓							
payment_id	DOUBLE			✓		✓				NULL
room_gender	VARCHAR(45)									NULL
deposite_status	VARCHAR(10)									NULL

log Table



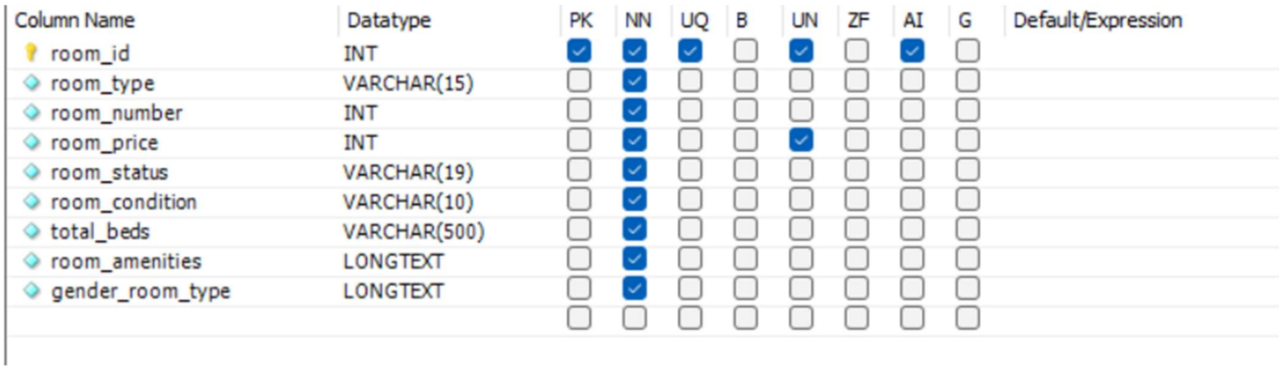
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Log_id	INT	✓	✓					✓		
user_id	INT		✓			✓				
user_Name	VARCHAR(50)									NULL
user_role	VARCHAR(45)									NULL
Login_date	VARCHAR(45)									NULL
Login_time	VARCHAR(45)									NULL
logout_time	VARCHAR(45)									NULL

Payment Table



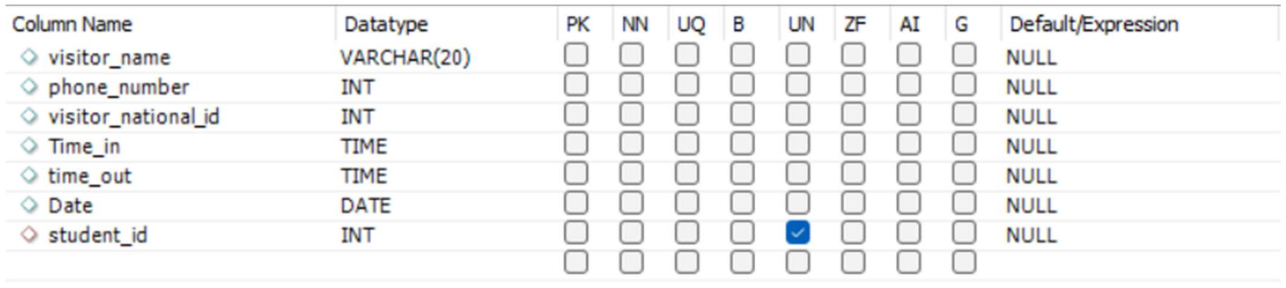
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Payment_id	INT	✓	✓	✓		✓		✓		
Student_id	INT					✓				NULL
Room_id	INT					✓				NULL
mpesa_transaction_id	VARCHAR(55)									NULL
payment_for	VARCHAR(100)									NULL
Payment_status	VARCHAR(25)		✓							
tansaction_date	DATE									NULL

Room Table



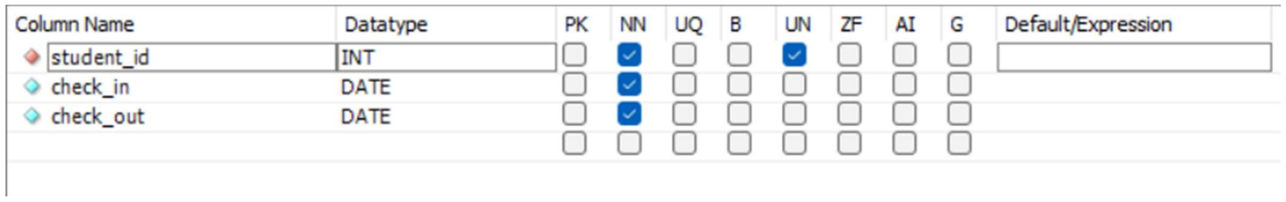
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
room_id	INT	✓	✓	✓		✓		✓		
room_type	VARCHAR(15)		✓							
room_number	INT		✓							
room_price	INT		✓			✓				
room_status	VARCHAR(19)		✓							
room_condition	VARCHAR(10)		✓							
total_beds	VARCHAR(500)		✓							
room_amenities	LONGTEXT		✓							
gender_room_type	LONGTEXT		✓							

Visitor Table



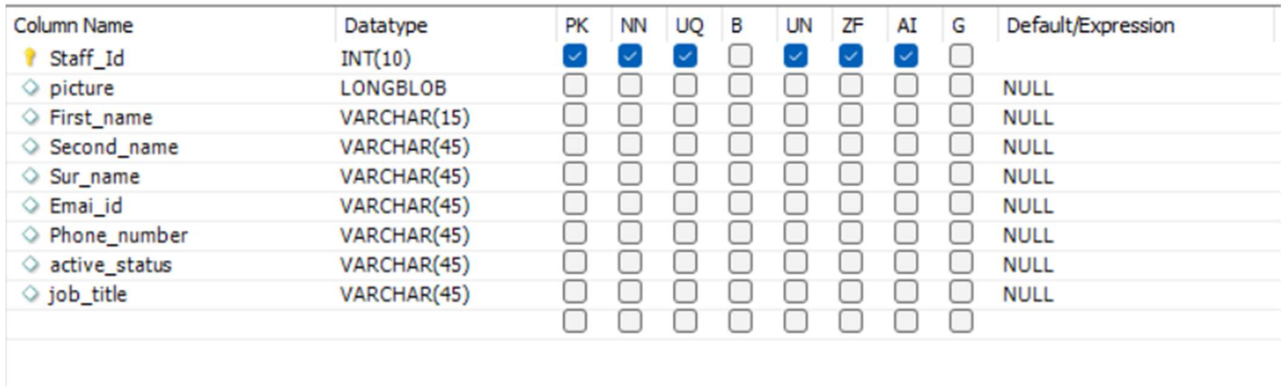
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
visitor_name	VARCHAR(20)									NULL
phone_number	INT									NULL
visitor_national_id	INT									NULL
Time_in	TIME									NULL
time_out	TIME									NULL
Date	DATE									NULL
student_id	INT					✓				NULL

Hostel Stay Table



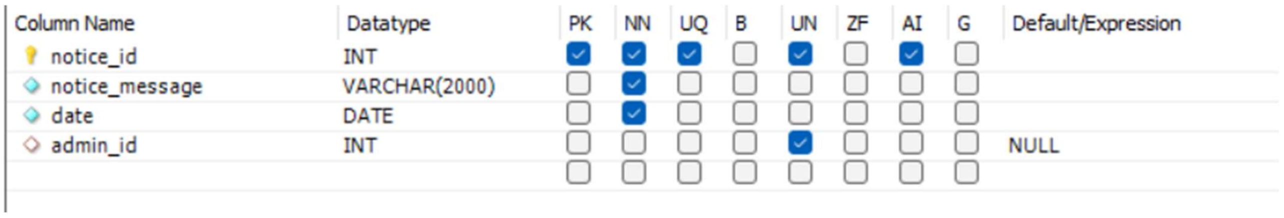
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
student_id	INT		✓			✓				
check_in	DATE		✓							
check_out	DATE		✓							

Staff Table



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
Staff_Id	INT(10)	✓	✓	✓		✓	✓	✓		
picture	LONGBLOB									NULL
First_name	VARCHAR(15)									NULL
Second_name	VARCHAR(45)									NULL
Sur_name	VARCHAR(45)									NULL
Emai_id	VARCHAR(45)									NULL
Phone_number	VARCHAR(45)									NULL
active_status	VARCHAR(45)									NULL
job_title	VARCHAR(45)									NULL

Notice board Table

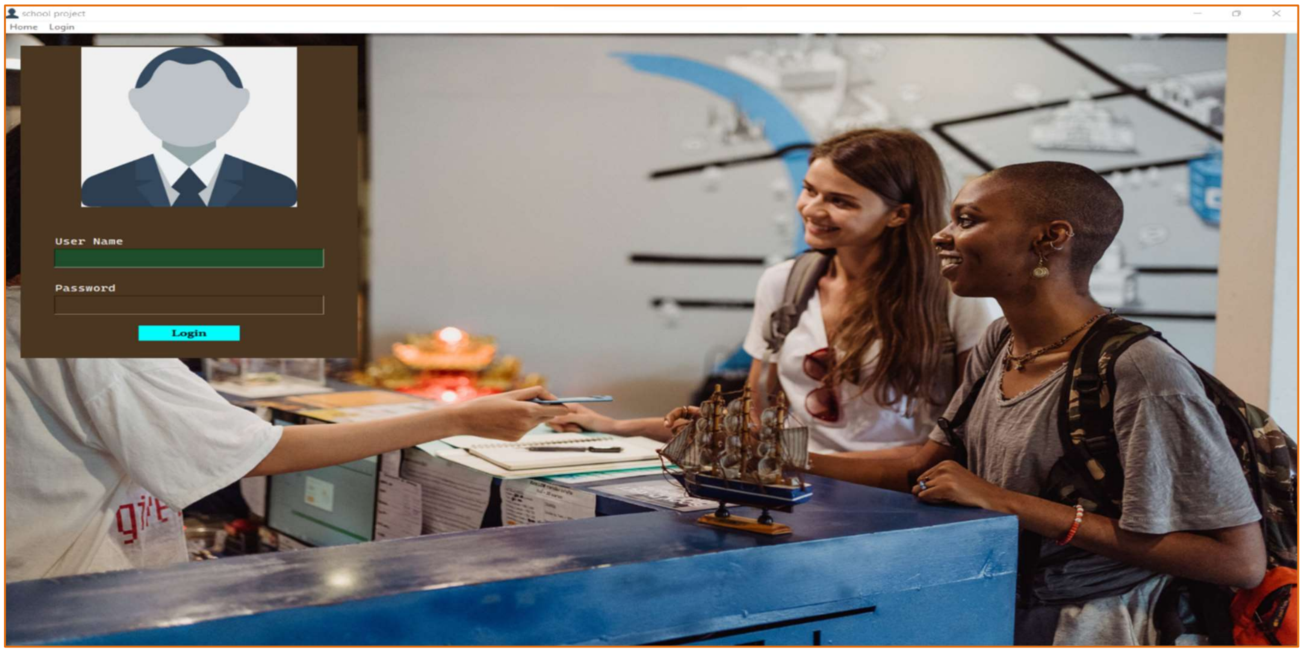


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
notice_id	INT	✓	✓	✓		✓		✓		
notice_message	VARCHAR(2000)		✓							
date	DATE		✓							
admin_id	INT					✓				NULL

4.4 USER INTERFACE

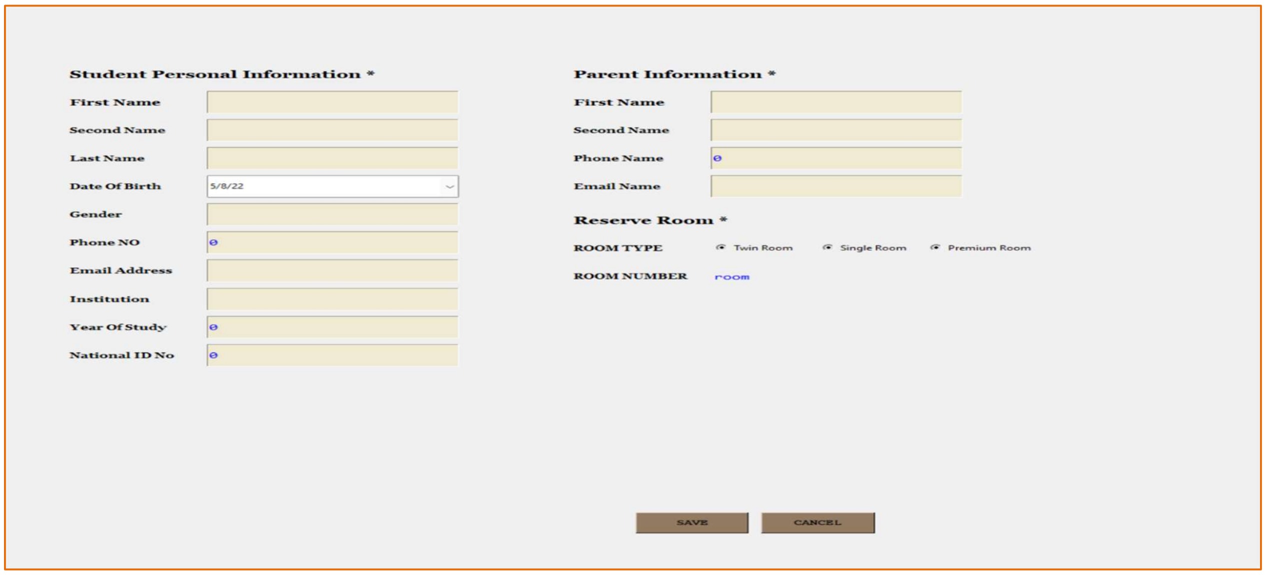
Login Page

This page consists of the administrators and student’s login. One has to have a username or password to be able to access the site



Allotment Page

This panel allows the administrator to add students to the system. He can collect student details in order to assign rooms for the student



Student Personal Information *

First Name	
Second Name	
Last Name	
Date Of Birth	5/8/22
Gender	
Phone NO	0
Email Address	
Institution	
Year Of Study	0
National ID No	0

Parent Information *

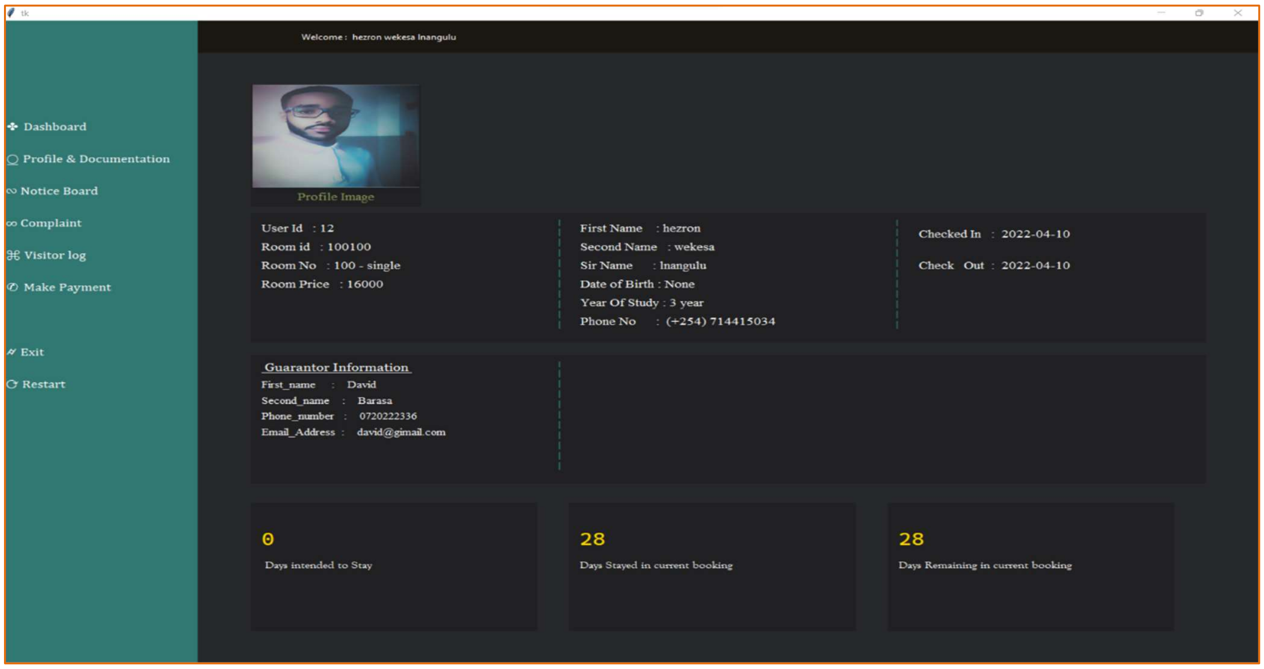
First Name	
Second Name	
Phone Name	0
Email Name	

Reserve Room *

ROOM TYPE	Twin Room / Single Room / Premium Room
ROOM NUMBER	room

SAVE CANCEL

Student Module



Welcome : hezron wekesa lnangulu

User Id : 12
Room id : 100100
Room No : 100 - single
Room Price : 16000

First Name : hezron
Second Name : wekesa
Sir Name : lnangulu
Date of Birth : None
Year Of Study : 3 year
Phone No : (+254) 714415034

Checked In : 2022-04-10
Check Out : 2022-04-10

Guarantor Information
First_name : David
Second_name : Barasa
Phone_number : 0720222336
Email_Address : david@gimail.com

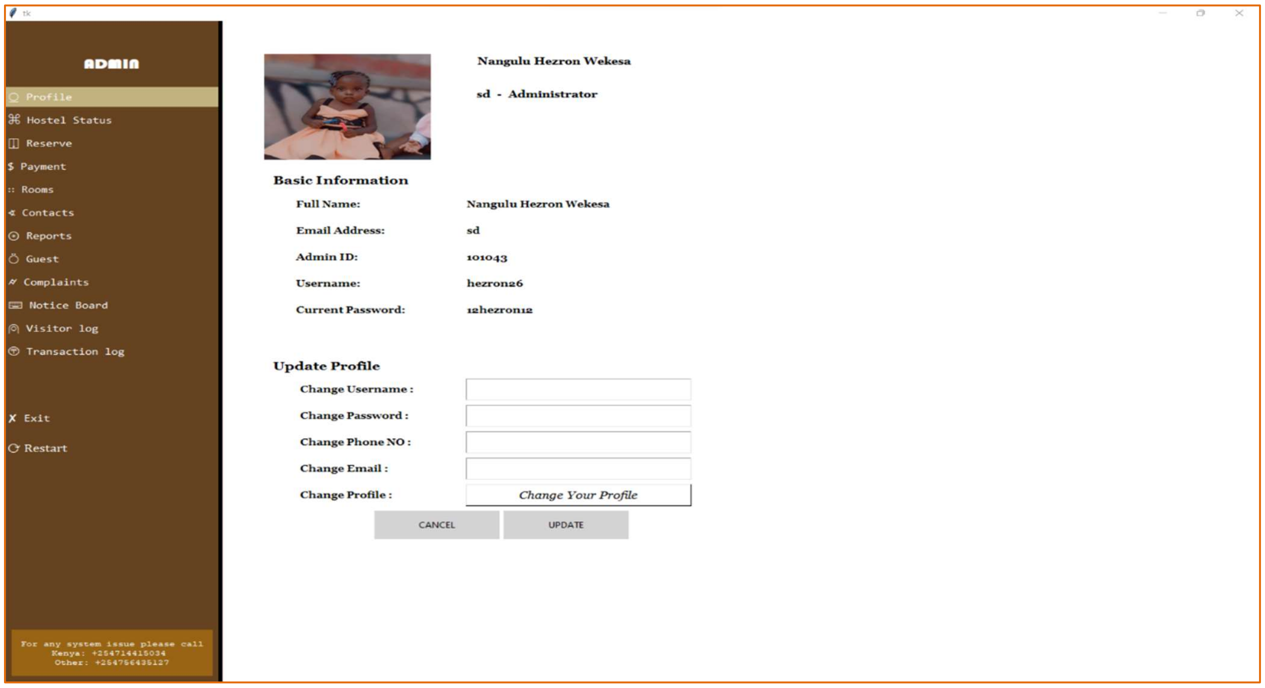
0
Days intended to Stay

28
Days Stayed in current booking

28
Days Remaining in current booking

Admin Page

This is page for the administrator. He has the ability to control who and what goes into the system. He can add, remove and update information in this system. In this panel is used to register a student, view fees, view rooms, view mess bills, allot students to blocks and rooms, view courses of students and assign them to special or specific rooms. He can also add new students to the system



Nangulu Hezron Wekesa

sd - Administrator

Basic Information

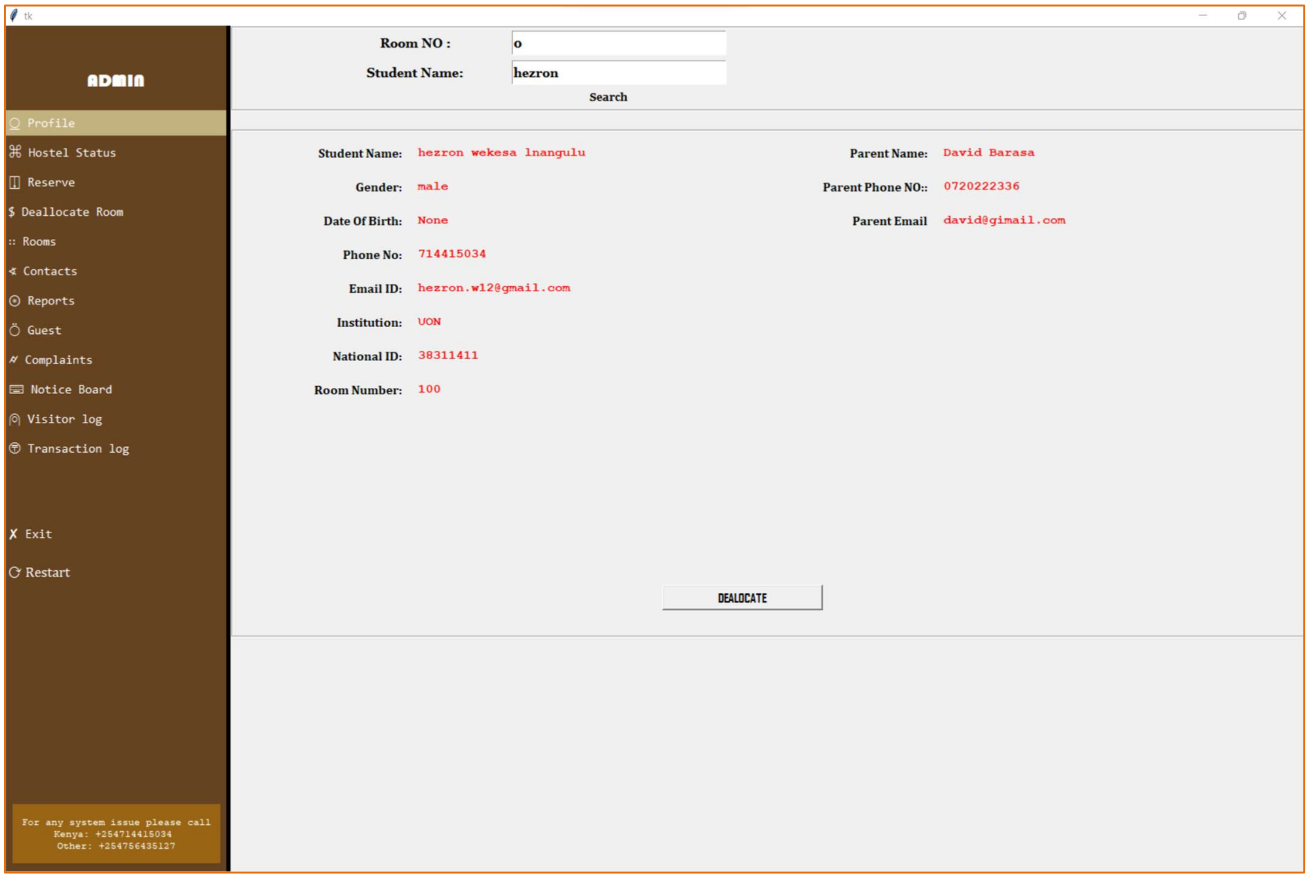
Full Name: Nangulu Hezron Wekesa
Email Address: sd
Admin ID: 101043
Username: hezron26
Current Password: 12hezron12

Update Profile

Change Username :
Change Password :
Change Phone NO :
Change Email :
Change Profile : Change Your Profile

For any system issue please call
Kenya: +254714415034
Other: +254756435127

Deallocation module



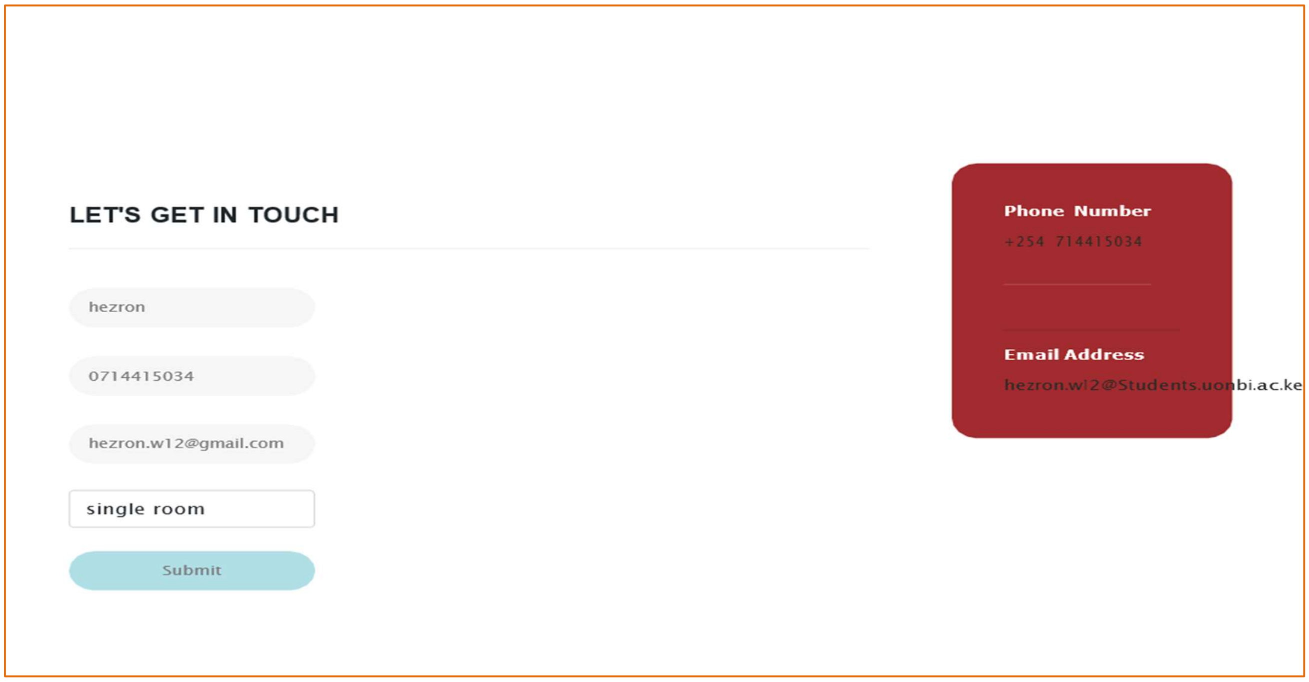
Room NO : o
Student Name: hezron

Student Name:	hezron wekesa lnangulu	Parent Name:	David Barasa
Gender:	male	Parent Phone NO::	0720222336
Date Of Birth:	None	Parent Email	david@gimail.com
Phone No:	714415034		
Email ID:	hezron.w12@gmail.com		
Institution:	UON		
National ID:	38311411		
Room Number:	100		

For any system issue please call
Kenya: +254714415034
Other: +254756435127

Guest Interface

Guest interface will be a website, with Python as backend and HTM as Frontend. Guest information will be stored in JSON. Portion of the Guest User Interface below:



LET'S GET IN TOUCH

hezron
0714415034
hezron.w12@gmail.com
single room

Phone Number
+254 714415034

Email Address
hezron.w12@Students.uonbi.ac.ke

CHAPTER FIVE: TESTING

5.1 Introduction of Testing

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations, and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is quality assurance.

5.2 System Testing

As the part of system testing, we execute the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is quality assurance. Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is clone. Using detailed testing strategies, a test plan is carried out on.

5.2 Unit Testing

The software units in a system are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches the type and size supported by Python. The various controls are tested to ensure that each performs its action as required.

5.3 Integration Testing

Data can be lost across any interface; one module can have an adverse effect on another, sub-functions when combined, may not produce the desired major functions. Integration testing is systematic testing to discover errors associated with the interface. The objective is to take unit-tested modules and build a program structure. All the modules are combined and tested as a whole. Here the Server module and Client module options are integrated and tested. This testing provides the assurance that the application is well integrated functional unit with smooth transition of data.

5.4 User Acceptance Testing

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 keeping in touch with the system users at the time of developing and making changes whenever required.

5.2: System Functionality

The system displays the login form that allows authorized user to access the system contems and

details

CHAPTER SIX: SYSTEM IMPLIMENTATION

5.0 Introduction

This chapter emphasizes the actual system implementation.Implementation is the stage in the project where the theoretical design is turned into a working system and gives confidence in the new system for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over and evaluation of changes to methods. Apart from planning, major tasks for preparing the implementation are education and training of users. The implementation process begins by preparing a plan for the implementation of the system. According to this plan, activities are to be carried out; discussions are to be made regarding the equipment and resources, and additional equipment has to be acquired to implement the new system. In a network backup system, no additional resources are needed. Implementation is the final and the most important phase. The most critical stage in achieving a successful new system is giving users confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it is found to be working according to the specification. This method also offers the greatest security, since the old system can take over if errors are found or an inability to handle certain types of transactions while using the new system.

The purpose of system implementation was to make sure that the correct application is delivered to the encl user. Besides that, this chapter also emphasizes on how the testing is done to confirm it meets user requirements.

The system will be implemented after thorough testing to make sure it is error free and working as per the user specifications.

5.1 User Training

After the system is implemented successfully, training of the user is one of the most important subtasks of the developer

5.2 Security and Maintenance

Maintenance is based on fixing the problems reported, changing the interface with other software or hardware and enhancing the software.

Security measures are provided to prevent unauthorized access of the database at various levels. Password protection and simple procedures to prevent unauthorized access are provided for the users . The system allows the user to enter the system only through the proper user name and password.

CHAPTER SEVEN: CONCLUSION

The new system automatically builds and maintains fast access to each record in order to maximize retrieval speed as compared to the previous system where data entry had to be done manually by writing clown on forms for particular records

> Performance
> The system allows a user to enter new records into a given file, modify, correct, or delete existing records, and displays reports either in detail or summaries according to the user's information requirements
>
> Security
> The system offers security at user level thereby allowing you to maximize the security of the data. Users have to log on using passwords however; passwords have to be kept confidential so as to prevent unauthorized users from gaining access to the system.
>
> Accuracy
> The system offers a high degree of accuracy. The system can perform millions of data entry with same accuracy unlike in the previous system where sometime administrators would get tired and commit errors

To conclude the description about the project The project, developed using Python and MySQL database. This hostel management software is designed for people who want to manage various activities in the hostel. For the past few years, the numbers of educational institutions have been increasing rapidly. Thereby the numbers of hostels are also increasing for the accommodation of the students studying in this institution. And hence there is a lot of strain on the people who are running the hostel and software's are not usually used in this context. This particular project deals with the problems of managing a hostel and avoids the problems which occur when carried manually.

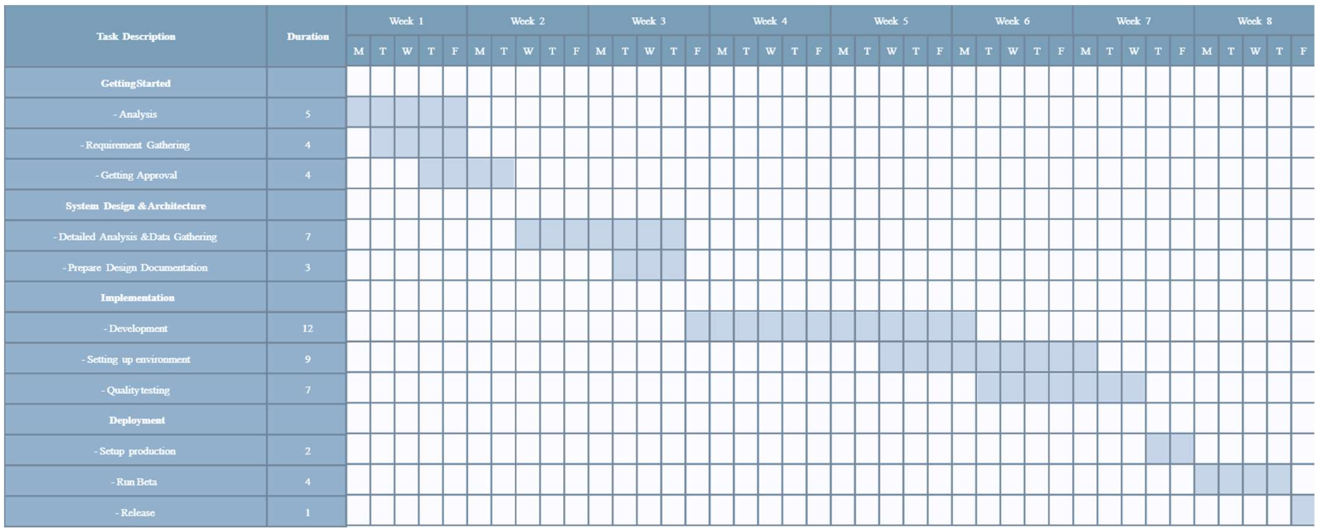
Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible with the existing system with the system which is more user-friendly and more GUI-oriented.
This new system will deal with the problems of managing a hostel and avoids the problems which occur when carried manually.

Identification of the drawbacks of the existing system leads to the designing of computerized system that will be compatible with the existing system with the system which is more user-friendly and more GUI-oriented

References & Citation

[1] Software Engineering 9th Edition, Ian Sommerville
[2] Fundamentals of Database System, 6th Edition, Ramez Elmasri, Shamkant B. Navathe
[3] ER Diagram Tutorial: https://www.tutorialspoint.com/dbms/er_diagram_representation.htm
[4] Requirement Engineering: http://morse.inf.unideb.hu/valseg/gybitt/07/ch02.html.
[5] Hotel Management System: https://www.scribd.com/doc/63824633/Hotel-ManagementSystem
[6] Case Study: https://www.scribd.com/doc/27927992/Hotel-Management-Case-Study
[7] Data Flow Diagram: http://myyee.tripod.com/cs457/dfd.htm
[8] Requirement Engineering: https://en.wikipedia.org/wiki/Requirements_engineering
[9] J. R. Groff and P. N Weinberg. Complete Reference SQL Second Edition.
[10] https://www.coursehero.com/file/129466515/Proposaldocx//
[11] https://www.scribd.com/presentation/561218838/Enabling-Consolidating-and-Codifying-Statutes/
[12] https://www.scribd.com
[13]: https://docplayer.net/37880829-Waterfall-model-application-in-development-dorm-student-information-management-system-west-kutai.html
[14] www.bing.com
[15] https://codeshoppy.com/shop/product/hostel-management-system
[16] https://www.modishproject.com/design-and-implementation-of-hostel-management-system//
[17] https://studylib.net/doc/25760269/hostel-management-system-full-project-1
[18] https://www.ijraset.com/research-paper/hostel-management-system-hms/
[19] https://www.coursehero.com/file/113222636/1634817424870-hostel-management-systemsdocx.
[20] https://www.ijraset.com/fileserve.php?FID=31517/
[21] https://pdfcoffee.com/hostel-management-system-9-pdf-free.html
[22] http://www.smconsultant.ae/page/130-feasibility-study-and-business-plan.html#:~:text=Economic%20Feasibility,and%20compare%20them%20with%20costs.
[23] https://studylib.net/doc/25760269/hostel-management-system-full-project-1
[24] https://www.academia.edu/40487624/Hostel_Automation_System_
[25] https://ir.kiu.ac.ug/bitstream/20.500.12306/13009/1/img-0162.pdf
[26] https://www.coursehero.com/file/140241330/elahi4docx/
[27] https://www.passeidireto.com/arquivo/4532972/entity-relationship-model.
[28] https://www.relationaldbdesign.com/database-design/module6/intro-entityRelationship-diagrams.php
[29] https://www.academia.edu/36649813/DESIGN_AND_IMPLEMENTATION_OF_ONLINE_HOSTEL_MANAGEMENT_SYSTEM
[30] https://docshare.tips/hotel-management-system_57513d25b6d87fd2b88b4fc7.html/
[31] https://www.coursehero.com/file/p34ru1l/Testing-is-used-to-show-incorrectness-and-considered-to-success-when-an-error-is//
[32] https://docshare.tips/hotel-management-system_57513d25b6d87fd2b88b4fc7.html/
[33] http://scholarsmepub.com/wp-content/uploads/2017/04/SJEAT-22114-118.pdf/
[34]: https://www.coursehero.com/file/142816511/Sqe-Assignment-3docx//
[35] https://docshare.tips/final-report_5856b465b6d87f224e8b6151.html/
[36] https://www.wikitechy.com/final-year-project/dotnet/room-booking-management-system
[37] https://www.bing.com/ck/a?!

APPENDIX II: Project Gantt Chart



Task Description	Duration	Week 1					Week 2					Week 3					Week 4					Week 5					Week 6					Week 7					Week 8				
		M	T	W	T	F	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F	M	T	W	T	F
GettingStarted																																									
- Analysis	5	▨	▨	▨	▨	▨																																			
- Requirement Gathering	4		▨	▨	▨	▨																																			
- Getting Approval	4				▨	▨	▨	▨																																	
System Design & Architecture																																									
- Detailed Analysis &Data Gathering	7								▨	▨	▨	▨	▨	▨	▨																										
- Prepare Design Documentation	3												▨	▨	▨																										
Implementation																																									
- Development	12															▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨														
- Setting up environment	9																							▨	▨	▨	▨	▨	▨	▨	▨	▨									
- Quality testing	7																										▨	▨	▨	▨	▨	▨	▨	▨							
Deployment																																									
- Setup production	2																																	▨	▨						
- Run Beta	4																																			▨	▨	▨	▨		
- Release	1																																							▨	

APPENDIX II: PROJECT TASK SCJEDULE

TASK NAME	DURATION	START DATE	FINISH DATE
Problem Definition	1 week	10/02/2022	16/02/2022
Analysis and Site-Map and Functionality	1 week	15/02/2022	29/02/2022
Design: coming-up with DFD and Relational Schemas	1 week	24/02/2022	01/02/2022
Backend Implementation: MySQL and Python	3 weeks	13/03/2022	05/04/2022
Frontend Implementation: User Interface: Tkinter and HTML	3 weeks	06/04/2022	03/05/2022
Preparation Of Final Report and testing	2 weeks	14/04/2022	17/05/2022

APPENDIX III: IMPORTANT CODE

Database connection

```
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="*********",
    database="hostel"
)
mycursor = mydb.cursor()
```

Creating a GUI instance with tkinter module

```
import tkinter as tk
root = tk.Tk()
root.state('zoomed')
root.minsize(1050, 800)
root.maxsize(1350, 950)
```

Login Function

```
def Login_function(username, password):
    if password == '' and username == '':
            stut1 = tk.Label(Login_Page, text='╳ Error:\n\n User name or Password is Empty/blank', bg='#BA0021',fg='black', font='-family {Georgia}  -size 10 -weight bold')
            stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
            stut1.after(4100, lambda: stut1.place_forget())
            return
    else:
           mycursor.execute("select * from hostel.users where user_name=%s and user_passwd=%s", (username, password))
           myresult = mycursor.fetchall()
           if len(myresult) == 0:
                stut1 = tk.Label(Login_Page, text='╳ LOGIN ERROR:\n\n Invalid Username or Password', bg='#FF0000',fg='black', font='-family {Georgia}  -size 10 -weight bold')
                stut1.place(relx=0.6, rely=0.05, relwidth=0.25, relheight=0.09)
                stut1.after(4100, lambda: stut1.place_forget())
                return
            else:
                    user_id = myresult[0][2]
                    with open('session.txt', 'w') as f:
                        f.write(f'{user_id}')
                    f.close()
                    if myresult[0][3] == 'student':
                            mycursor.execute("SELECT * FROM hostel.student WHERE user_id = %s;", [user_id])
                            fetg = mycursor.fetchall()
                            mycursor.execute("INSERT INTO hostel.log_rept (user_id, user_Name, user_role, Login_date, Login_time) VALUES (%s, %s, %s, CURDATE(),
CURTIME());",(myresult[0][2],f'{fetg[0][1]} {fetg[0][2]} {fetg[0][2]}',myresult[0][3]))
                            mydb.commit()
                            import subprocess
                            cmd = 'python StudentPage.py'
                            p = subprocess.Popen(cmd, shell=True)
                            out, err = p.communicate()
                            print(err)
                            print(out)

                    if myresult[0][3] == 'admin':
                            mycursor.execute("SELECT * FROM hostel.admins WHERE admin_Id = %s;", [user_id])
                            fetg = mycursor.fetchall()
                            mycursor.execute("INSERT INTO hostel.log_rept (user_id, user_Name, user_role, Login_date, Login_time) VALUES (%s, %s, %s, CURDATE(),
CURTIME());",(myresult[0][2], fetg[0][1], myresult[0][3]))
                            mydb.commit()

                            import subprocess
                            cmd = 'python AdminPage.py'
                            p = subprocess.Popen(cmd, shell=True)
                            out, err = p.communicate()
                            print(err)
                            print(out)
```

profile Update Function

```
def open_file():
   from tkinter import filedialog
   filepath = filedialog.askopenfilename(title="Select Profile Photo", filetypes=((" ","*.jpg"), (" ","*.png")))
   if filepath != None:
       print(filepath)
       # Open a file in binary mode
       file = open(filepath, 'rb').read()
       # We must encode the file to get base64 string
       file = base64.b64encode(file)
       # Execute the query and commit the database.
       mycursor.execute('UPDATE hostel.users SET user_image = %s WHERE user_id = %s', [file,user_id])
       mydb.commit()
```

Room Allotment Function

```
def reserve_room(f_name, s_name, l_name, d_birth, s_gender, s_phone, s_email, year_study, s_institution, s_national_is,
                 p_first_name, p_second_name, p_phone_no, p_email, room_ty):
    if f_name == '':
        stut1 = tk.Label(Reserve_Room_frame, text='╳ Error:\n\n Student First Name Field is Empty', bg='#BA0021',
                         fg='white', font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return
    if s_name == '':
        stut1 = tk.Label(Reserve_Room_frame, text='╳ Error:\n\n Student Second Name Field is Empty', bg='#BA0021',
                         fg='white', font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return
    if l_name == '':
        stut1 = tk.Label(Reserve_Room_frame, text='╳ Error:\n\n Last First Name Field is Empty', bg='#BA0021',
                         fg='white', font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return

    if s_gender != '':
        if s_gender == 'male' or s_gender == 'Male':
            s_gender = 'Male'

        elif s_gender == 'female' or s_gender == 'Female':
            s_gender = 'Female'

        elif (s_gender != 'female') or (s_gender != 'Female') or (s_gender != 'male') or (s_gender != 'Male'):
            messagebox.showwarning("ERROR", "Gender Specification either male or female")
            stut1 = tk.Label(Reserve_Room_frame, text='╳ ERROR:\n\n Student Gender !', bg='#BA0021', fg='white',
                             font='-family {Georgia}  -size 10 -slant italic')
            stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
            stut1.after(3100, lambda: stut1.place_forget())

    if s_phone == 0:
        stut1 = tk.Label(Reserve_Room_frame, text='╳ ERROR:\n\n Student Phone Number', bg='#BA0021', fg='white',
                         font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return

    if s_email != '':
        regex = '^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}$'
        if (re.search(regex, s_email)):
            print('valid s email')
        else:
            stut1 = tk.Label(Reserve_Room_frame, text='╳ Error:\n\n Student Invalid Emali', bg='#BA0021', fg='white',
                             font='-family {Georgia}  -size 10 -slant italic')
            stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
            stut1.after(3100, lambda: stut1.place_forget())
    else:
        messagebox.showwarning("ERROR", "No Email Provided")
        stut1 = tk.Label(Reserve_Room_frame, text='╳ Error:\n\n No Student Email Provided', bg='#BA0021', fg='white',
                         font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())

    sssdtoday = ty.today()
    db = abs((d_birth - sssdtoday).days)
    age = db / 365
    if d_birth != sssdtoday:
        if d_birth > sssdtoday:
            stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n Date of Birth Erro', bg='#BA0021', fg='white',
                             font='-family {Georgia}  -size 10 -slant italic')
            stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
            stut1.after(3100, lambda: stut1.place_forget())
            return
        if age < 17:
            print(age)
            stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n Age is Less Than 17', bg='#BA0021', fg='white',
                             font='-family {Georgia}  -size 10 -slant italic')
            stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
            stut1.after(3100, lambda: stut1.place_forget())
            return
    else:
        stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n Select Age', bg='#BA0021', fg='white',
                         font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return

    if year_study <= 0 or year_study >= 7:
        stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n Invalid Year Of Study', bg='#BA0021', fg='white',
                         font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return

    if s_institution == '':
        stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n Please Fill Student Institution Name', bg='#BA0021',
                         fg='white', font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return

    if s_national_is == 0:
        stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n National ID field is empty', bg='#BA0021', fg='white',
                         font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return

    if p_first_name == '':
        stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n Parent First Name Field is Empty', bg='#BA0021',
                         fg='white', font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return
    if p_second_name == '':
        stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n Parent Second Name Field is Empty', bg='#BA0021',
                         fg='white', font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return

    if p_email != '':
        regex = '^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}$'
        if (re.search(regex, p_email)):
            print('Valid Parent Email')
        else:
            messagebox.showwarning("ERROR", "Invalid Parent Email")
            stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n Invalid Parent Email', bg='#BA0021', fg='white',
                             font='-family {Georgia}  -size 10 -slant italic')
            stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
            stut1.after(3100, lambda: stut1.place_forget())
            return
    else:
        stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n No Parent Email Provided', bg='#BA0021', fg='white',
                         font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return

    if p_phone_no == 0:
        stut1 = tk.Label(Reserve_Room_frame, text='✗ Error:\n\n Parent Phone Number', bg='#BA0021', fg='white',
                         font='-family {Georgia}  -size 10 -slant italic')
        stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.09)
        stut1.after(3100, lambda: stut1.place_forget())
        return

    mycursor.execute(
        "SELECT * FROM hostel.student where  first_name = %s and second_name =  %s and last_name = %s and gender = %s and phone_no = %s and email_id = %s;",
        (f_name, s_name, l_name, s_gender, s_phone, s_email))
    check_inf = mycursor.fetchall()
    print(check_inf)
    if check_inf == []:
        mycursor.execute(
            "SELECT * FROM hostel.room where room_type = %s and room_status='not occupied' and room_condition = 'good'",
            [room_ty])
        r_output = mycursor.fetchall()
        if r_output != []:
            room_number = r_output[0][2]
            room_id = r_output[0][0]
            mycursor.execute("UPDATE hostel.room SET room_status = 'occupied' WHERE  room_number = %s", [room_number])
            mydb.commit()

            user_name = f_name + str(random.randrange(50, 1000))
            pass_word = s_name + str(random.randrange(50, 1000))
            mycursor.execute("INSERT INTO hostel.users (user_name, user_passwd, user_role) VALUES (%s, %s, %s);",
                             (user_name, pass_word, 'student'))
            mydb.commit()
            mycursor.execute("SELECT * FROM hostel.users where user_name = %s and user_passwd = %s and user_role= %s;",
                             (user_name, pass_word, 'student'))
            user_det = mycursor.fetchall()
            newUser_id = user_det[0][2]
            spr = "INSERT INTO hostel.student(first_name, second_name, last_name, date_of_birth, gender, phone_no, email_id, year_of_study, institution,
national_id, user_id,  room_id) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);"
            var = (
            f_name, s_name, l_name, d_birth, s_gender, s_phone, s_email, year_study, s_institution, s_national_is,
            newUser_id, room_id)
            mycursor.execute(spr, var)
            mydb.commit()

            mycursor.execute(
                'SELECT * FROM hostel.student where first_name = %s and second_name = %s and last_name = %s and date_of_birth = %s and gender = %s;',
                (f_name, s_name, l_name, d_birth, s_gender))
            student_id = mycursor.fetchall()

            spr1 = "INSERT INTO hostel.parent_info (First_name, Second_name, Phone_number, Email_Address, student_id) VALUES (%s, %s, %s, %s, %s);"
            var1 = (p_first_name, p_second_name, p_phone_no, p_email, student_id[0][0])
            mycursor.execute(spr1, var1)
            mydb.commit()

            import smtplib
            sender_add = 'hezron.w12@gmail.com'
            receiver_add = 'mdenish057@gmail.com'
            password = '2608Ann12'

            smtp_server = smtplib.SMTP("smtp.gmail.com", 587)
            smtp_server.ehlo()
            smtp_server.starttls()
            smtp_server.ehlo()
            smtp_server.login(sender_add, password)
            msg_to_be_sent = f'''
                            Hello {s_name} {l_name},

                                     Hope you are doing well.
                                     Welcome to Students Apartments (Hostels)!
                                     your login Cridential are :
                                               USERNAME : {user_name}
                                               PASSWORD : {pass_word}
                                      you can login and change your password and upload your profile
a
                            My name is Nangulu Hezron, and welcome to the Qwetu community.
                            regards
                            '''
            # sending the mail by specifying the from and to address and the message
            smtp_server.sendmail(sender_add, receiver_add, msg_to_be_sent)
            print('Successfully the mail is sent')
            smtp_server.quit()

            stut1 = tk.Label(Reserve_Room_frame, text='✓ Successfuly Reserved', bg='green', fg='#6B4423',
                             font='-family {Georgia}  -size 14 -slant italic')
            stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.07)
            stut1.after(2100, lambda: stut1.place_forget())
        else:
            stut1 = tk.Label(Reserve_Room_frame, text=' ERROR \n no available room-type ', bg='red', fg='#6B4423',
                             font='-family {Georgia}  -size 14 -slant italic')
            stut1.place(relx=0.7, rely=0.04, relwidth=0.25, relheight=0.07)
            stut1.after(3000, lambda: stut1.place_forget())
```

deallocation Module

```
Deallocate_frame = tk.Frame(root)
Deallocate_frame.place(relx=0.173, rely=0, relwidth=0.83, relheight=1)
section_1_deall = tk.LabelFrame(Deallocate_frame)
section_1_deall.place(relx=0.001, rely=0.001, relwidth=0.998, relheight=0.1)
def Search_del_student(room_num, student_name):

        mycursor.execute('SELECT * FROM hostel.student WHERE first_name = %s OR second_name = %s OR last_name = %s;', [student_name,student_name,student_name])
        deloc_s = mycursor.fetchall()
        if deloc_s == []:
            de_conf = tk.Label(Deallocate_frame, bg='yellow', text=' ! SOORY ! \nStudent Name Not IN the System', font='-family {Courier New}  -size 10 -weight
bold')
            de_conf.place(relx=0.25, rely=0.4, relwidth=0.4, relheight=0.1)
            de_conf.after(4000, lambda: de_conf.place_forget())
            return
        room_id_del = deloc_s[0][12]
        mycursor.execute('SELECT * FROM hostel.room WHERE room_id = %s;', [room_id_del])
        deloc_r = mycursor.fetchall()
        mycursor.execute('SELECT * FROM hostel.parent_info WHERE student_id = %s;', [deloc_s[0][0]])
        deloc_par = mycursor.fetchall()
        if room_num == deloc_r[0][2]:

                    tk.Label(section_2_deall, fg='red', text=f'{deloc_s[0][1]} {deloc_s[0][2]} {deloc_s[0][3]}', anchor='w', font='-family {Courier New}  -size
11 -weight bold').place(rely=0.01, relx=0.17, relwidth=0.27, relheight=0.065)
                    tk.Label(section_2_deall, fg='red', text=f'{deloc_s[0][5]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.077, relx=0.17, relwidth=0.27, relheight=0.065)
                    tk.Label(section_2_deall, fg='red',text=f'{deloc_s[0][4]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.144, relx=0.17, relwidth=0.27, relheight=0.065)
                    tk.Label(section_2_deall, fg='red',text=f'{deloc_s[0][6]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.211, relx=0.17, relwidth=0.27, relheight=0.065)
                    tk.Label(section_2_deall, fg='red',text=f'{deloc_s[0][7]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.278, relx=0.17, relwidth=0.27, relheight=0.065)
                    tk.Label(section_2_deall, fg='red',text=f'{deloc_s[0][9]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.345, relx=0.17, relwidth=0.27, relheight=0.065)
                    tk.Label(section_2_deall, fg='red',text=f'{deloc_s[0][10]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.412, relx=0.17, relwidth=0.27, relheight=0.065)
                    tk.Label(section_2_deall, fg='red',text=f'{deloc_r[0][2]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.479, relx=0.17, relwidth=0.27, relheight=0.065)

                    tk.Label(section_2_deall,  fg='red',text=f'{deloc_par[0][0]} {deloc_par[0][1]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.01, relx=0.66, relwidth=0.27, relheight=0.065)
                    tk.Label(section_2_deall, fg='red',text=f'{deloc_par[0][2]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.077, relx=0.66, relwidth=0.27, relheight=0.065)
                    tk.Label(section_2_deall, fg='red',text=f'{deloc_par[0][3]}', anchor='w', font='-family {Courier New}  -size 11 -weight
bold').place(rely=0.144, relx=0.66, relwidth=0.27, relheight=0.065)

                    def delete_student_records():
                                def YES():
                                       mycursor.execute('DELETE FROM hostel.student WHERE (student_id = %s);', [deloc_s[0][0]])
                                       mydb.commit()
                                       mycursor.execute('DELETE FROM hostel.parent_info WHERE (student_id = %s);', [deloc_s[0][0]])
                                       mydb.commit()
                                       mycursor.execute('DELETE FROM hostel.users WHERE (user_id = %s);', [deloc_s[0][11]])
                                       mydb.commit()
                                       mycursor.execute('DELETE FROM hostel.complaint WHERE (student_id = %s);', [deloc_s[0][0]])
                                       mydb.commit()
                                       mycursor.execute("UPDATE hostel.room  SET room_status = 'not occupied' WHERE(`room_id` = '100102');", [deloc_s[0][12]])
                                       mydb.commit()
                                def NO():
                                       de_conf.place_forget()

                                de_conf = tk.Frame(Deallocate_frame, bg=side_bar_frame_bg_color)
                                de_conf.place(relx=0.25, rely=0.4, relwidth=0.4, relheight=0.1)
                                tk.Label(de_conf, text='Are you Sure You Want To Dealocate This Student !',bg=side_bar_frame_bg_color,font='-family {Courier
New}  -size 10 -weight bold').place(relx=0,rely=0, relwidth=1, relheight=0.3)
                                tk.Label(de_conf, text='! !This Action Is Can not be Reversed! !', bg=side_bar_frame_bg_color, font='-family {Courier New}  -
size 10 -weight bold').place(relx=0, rely=0.31, relwidth=1, relheight=0.3)
                                tk.Button(de_conf, text='YES', bg='green' , command=YES).place(relx=0, rely=0.7, relwidth=0.5, relheight=0.3)
                                tk.Button(de_conf, text='NO', bg='red', command=NO).place(relx=0.5, rely=0.7, relwidth=0.5, relheight=0.3)

                    del_bt = tk.Button(section_2_deall, bg='#EFDECD',text='DEALOCATE', font='-family {Agency FB}  -size 11 -weight bold', command= lambda
:delete_student_records())
                    del_bt.place(relx=0.4, rely=0.9, relwidth=0.15, relheight=0.05)
                    changeOnHover(del_bt, 'brown', '#EFDECD')

        else:
            de_conf = tk.Label(Deallocate_frame,  bg='pink', text=' ! SOORY ! \nSTUDENT NAME DOSE-NOT MATCH ROOM NUMBER', font='-family {Courier New}  -size 10
-weight bold')
            de_conf.place(relx=0.25, rely=0.4, relwidth=0.4, relheight=0.1)
            de_conf.after(4000, lambda :de_conf.place_forget())



deallocate_room = tk.IntVar()
deallocate_student = tk.StringVar()
tk.Label(section_1_deall,  text='Room NO :', font='-family {Georgia}  -size 11 -weight bold').place(relx=0.1, rely=0.03, relheight=0.3, relwidth=0.14)
tk.Label(section_1_deall,  text='Student Name:', font='-family {Georgia}  -size 11 -weight bold').place(relx=0.1, rely=0.4, relheight=0.3, relwidth=0.14)
tk.Entry(section_1_deall,  textvariable=deallocate_room, font='-family {Georgia}  -size 11 -weight bold').place(relx=0.26, rely=0.03, relheight=0.3,
relwidth=0.2)
tk.Entry(section_1_deall,  textvariable=deallocate_student, font='-family {Georgia}  -size 11 -weight bold').place(relx=0.26, rely=0.4, relheight=0.3,
relwidth=0.2)
del_ser = tk.Button(section_1_deall, bg="#EFDECD",text='Search', font='-family {Georgia}  -size 9 -weight bold', borderwidth=0, command= lambda:
Search_del_student(deallocate_room.get(), deallocate_student.get()))
del_ser.place(relx=0.3, rely=0.723, relheight=0.27, relwidth=0.1)
changeOnHover(del_ser,'Green', '#EFDECD')
section_2_deall = tk.LabelFrame(Deallocate_frame)
section_2_deall.place(relx=0.001, rely=0.123, relwidth=0.998, relheight=0.6)
tk.Label(section_2_deall, text='Student Name:', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.01, relx=0.01, relwidth=0.15,
relheight=0.065)
tk.Label(section_2_deall,text='Gender:', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.077, relx=0.01, relwidth=0.15,
relheight=0.065)
tk.Label(section_2_deall, text='Date Of Birth:', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.144, relx=0.01, relwidth=0.15,
relheight=0.065)
tk.Label(section_2_deall, text='Phone No:', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.211, relx=0.01, relwidth=0.15,
relheight=0.065)
tk.Label(section_2_deall, text='Email ID:', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.278, relx=0.01, relwidth=0.15,
relheight=0.065)
tk.Label(section_2_deall, text='Institution:', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.345, relx=0.01, relwidth=0.15,
relheight=0.065)
tk.Label(section_2_deall, text='National ID:', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.412, relx=0.01, relwidth=0.15,
relheight=0.065)
tk.Label(section_2_deall, text='Room Number:', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.479, relx=0.01, relwidth=0.15,
relheight=0.065)
# parent info
tk.Label(section_2_deall, text='Parent Name:', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.01, relx=0.5, relwidth=0.15,
relheight=0.065)
tk.Label(section_2_deall, text='Parent Phone NO::', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.077, relx=0.5, relwidth=0.15,
relheight=0.065)
tk.Label(section_2_deall, text='Parent Email', anchor='e', font='-family {Cambria}  -size 11 -weight bold').place(rely=0.144, relx=0.5, relwidth=0.15,
relheight=0.065)
```