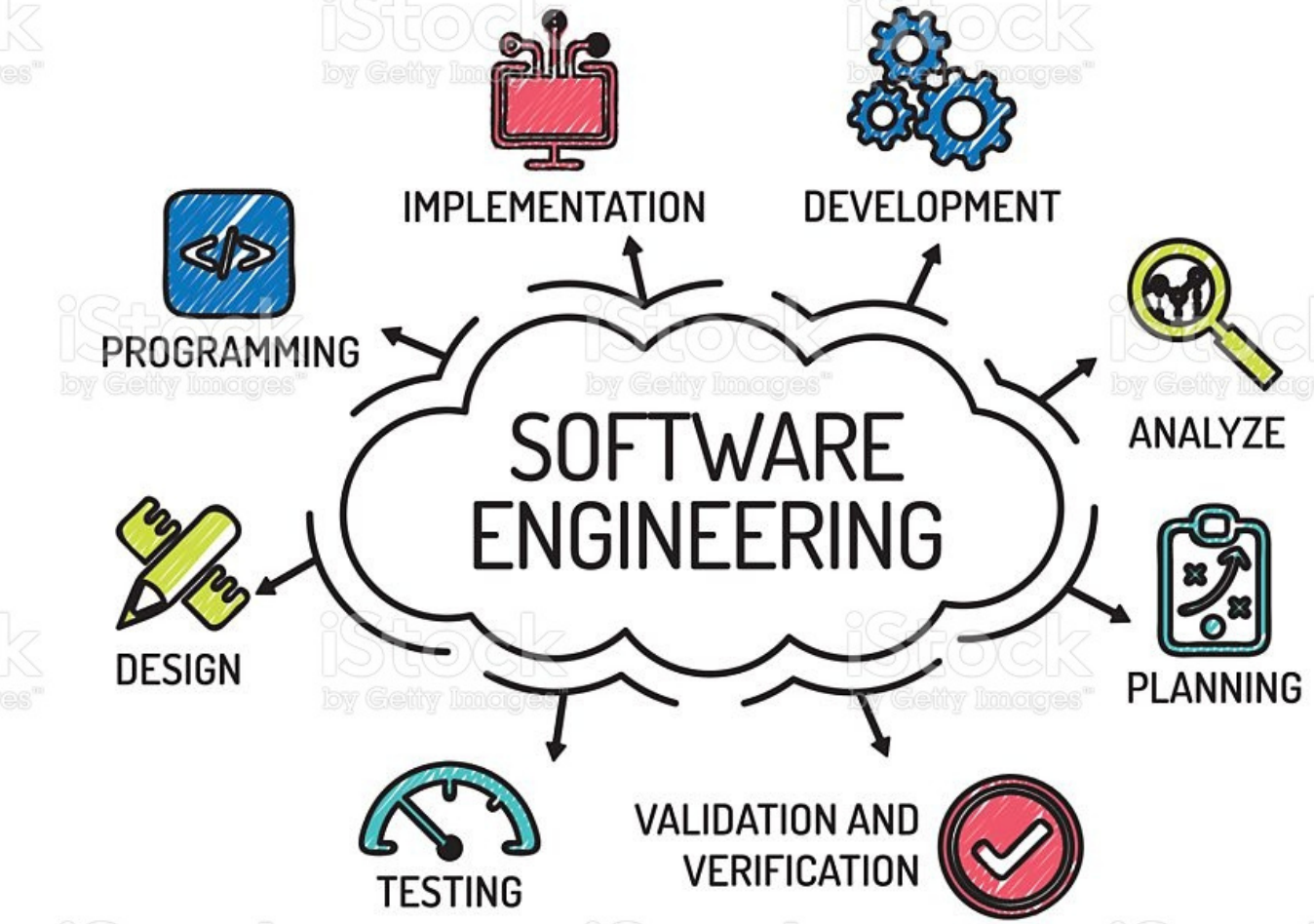
# PROJECT REPORT ON HOSTEL MANAGEMENT SYSTEM





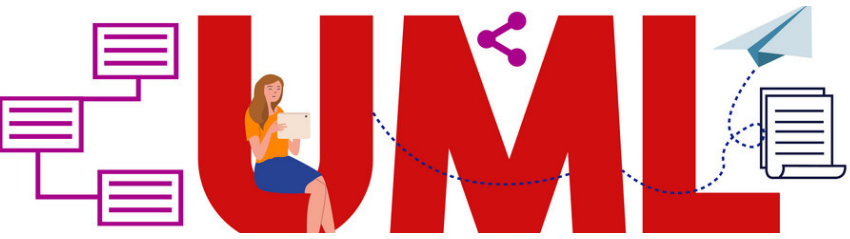
B18CS015

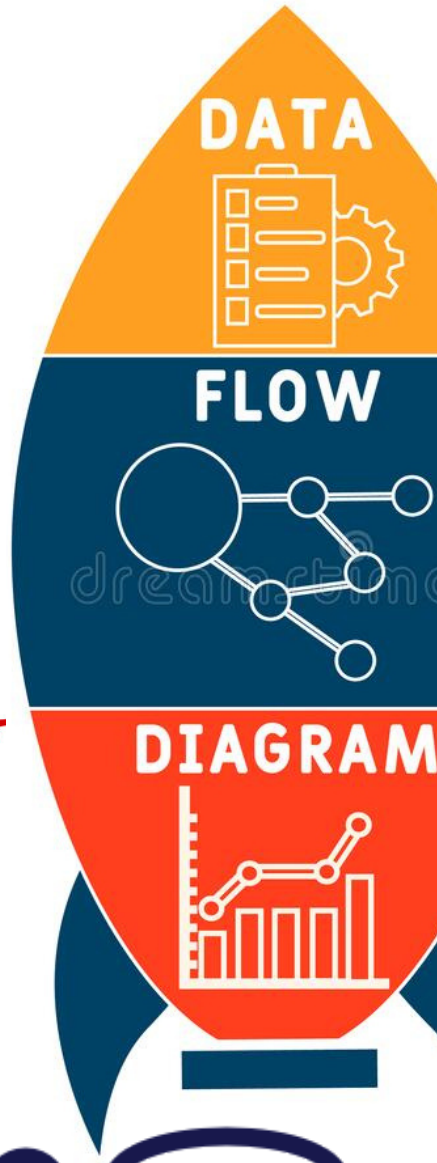
# JOURNEY TO DESIGN A SOFTWARE

# TABLE OF CONTENTS


PROJECT


PLAN
PLANNING
ARCHITECTURAL
STRATEGY
SCHEDULE
PROJECT
MARKETING
ARCHITECTURE
BRAINSTORMING
STRATEGIC
BLUEPRINT
APARTMENT
PROCESS
NOTEBOOK
WORKING
PLANE
MEETING
LIVING
ACCOUNTING
INVESTMENT
DESK
BOTTLE
ADVISER
COFFEE
MOBILE
WRITING
BATHROOM
TEAMWORK
DRAWING
ACTION
CHART
BOARD
NOTES
SIMPLE
DISCUSSING
BEDROOM
HEALTHY
PLANS
PAPERS
STUDIO
GRAPHIC
PAPERWORK
ROOM
DOCUMENTS
AROUND
KITCHEN
GROWTH
ANALYSIS
FLOOR
PHOTO
PENCIL
WOOD
FOCUS
CUP
COUPLE
PUTTING
RESULTS
POINTING
FURNITURE
CREATIVE
ADVERTISING
IDEAS
STARTUP
TEAM
INTERIOR
ARCHITECT
FUTURE
DOCUMENT
INSCRIPTION
AMERICA
EUROPE
BEDROOMS
ANIMALS
CHARTS
APPLE
ARROW
AGENDA
ARTS
OPEN
SHOT
LINEAR
TRIP
ADVISOR
SHOWING
STICKER
APPLICATION
AMBER
ASSOCIATES
HOTEL
CORPORATE
ADVERTISEMENT
VACATION
INTENTIONAL
ORIENTATIONS
SIGNS
AGE
STANDARD
ORGANIC
PEN
MANAGERS
GRAPHS
JOB
BLAN
ANGLE
BEHIND
ACCESSORIES
BASED
BANANA
CREW
CALCULATOR
WORKPLACE
KEY
BASIC
AQUA
CROPPED
GLARE
SELECTIVE
STANDING
AVOCADO
STORE
TOGETHER
AZURE
LAYOUT
TRAVEL
DAILY
EDUCATION
ANALYZE
TECHNICAL
SENIOR
LIGHT
SINGLE
DIET
RELEASES
DOODLE
BACKGROUNDS
BATHROOMS
ACHIEVEMENT
CONTEMPORARY
MARKER
TOWN
ANALYZING
ARCHITECTS
EFFECT


UML


DATA
FLOW
DIAGRAM


FEASIBILITY STUDY
resources
factors
operations
plan
management
schedule
new
testing
research
development
legal
funding
costs
project
money
idea
economic
project
analysis
capital
research
market
input
investor
investment
marketing
company
implement
finance
risk
decision
business
value
evaluate
cost

B18CS015

# TABLE OF CONTENTS

# INTRODUCTION

The Student hostel management system is web based software to provide college students accommodation to the university hostel more efficiently. This project also keeps details of the hostelers and applied students. It is headed by Warden. He will be the administrator. For accommodate a large number of students into hostel. This document is intended to minimize human works and make hostel allocation is an easier job for students and hostel authorities by providing online application for hostel, automatically select the students from the waiting list and mess calculation, complaint registration, notice board etc. etc. Students will get approval notification in their mails. Hostelers can view notice board, hostel fee, mess menu by login into the online system. 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 which is more users friendly and more GUI oriented. We can improve the efficiency of the system, thus overcome the drawbacks the existing system.

# OBJECTIVES

- Process allotment list.
- Admin can send the approval notification to every approved student via email .
- Students can register their complaints.
- Admin can edit notice board and each student can view it.
- Hostelers can check the status of every month's hostel fee.
- Hostel secretary can calculate hostel fee including mess fee and can edit mess menu.

# MAJOR FUNCTIONS

- It tracks all the information of Room, Facility, Student, etc.
- Manage the information of Room.
- Shows the information and description of the Hostel, Bed.
- To increase efficiency of managing the Hostel rooms.
- Integration of all records of Student Registration.
- Provides the searching facilities based on various factors. Such as Hostel, Bed,Student, Student Registration Id etc.
- Manage the information of a Student.

# PERFORMANCE ISSUES

- Hostelers photo can't be stored in database.
- Hostelers photo can't be stored in folder
- Online transaction facility is not included.

# SCOPE AND LIMITATIONS

- Hostel Managements System is designed for Hostel (like schools,universities).
- There will be predefined criteria for the Reserve to the hostels.
- He checks the attested application forms of the students obtained from the internet and verify it with the student database.
- If the students are found eligible then they are allotted to the hostel Room.

# FEASIBILITY

## TECHNICAL FEASIBILITY

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 latest technology or not. It happens that after a system is prepared a new technology arises and the user wants the system based on that technology. This system use windows platform, .net as front end technology and SQL server as back-end technology. Thus ONLINE HOSTEL MANAGEMENT SYSTEM is technically feasible.

- Hardware : - Hardware requirements include following points:
  1. Any Computer (Pentium PC, Laptop etc)
  2. RAM.
  3. Hard-disk.
  4. Printer.
  5. VDU(CRT or LCD).
  6. Peripherals (Like. Mouse and Keyboard).
- Software : - It includes following programs:
  1. Operating System(32 bits or higher)
  2. Visual Basic.NET.
  3. Office Package.
  4. Photoshop.
  5. Bandicam.

# ECONOMICAL FEASIBILITY

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis. .net using visual C# and SQL database easily available in internet.

| S.N. | Details | Rs |
|---|---|---|
| 1 | Computer (Laptop) | N/a |
| 2 | Required Software | 3,000 |
| 3 | Portable devices (Pen drives, CD etc) | 1,500 |
| 4 | Paper work cost | 500 |
| 5 | Miscellaneous | 3,000 |
| | **Total Cost** | **8,000** |

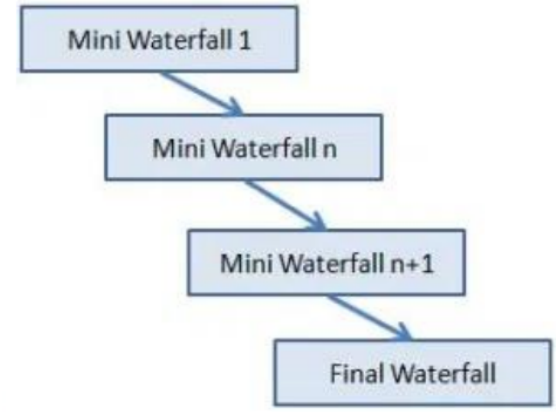
# OPERATIONAL FEASIBILITY

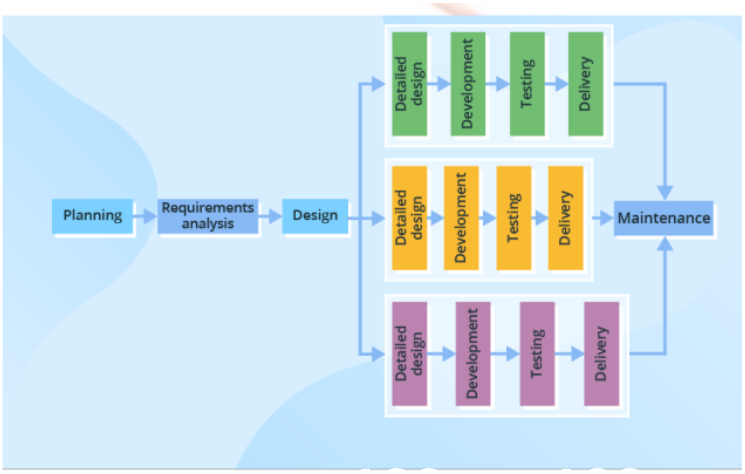
The project has been developed in such a way that it becomes very easy even for a person with little computer knowledge to operate it. This software is very user friendly and does not require any technical person to operate .Thus the project is even operationally feasible.

# PROJECT MANAGEMENT APPROACH

## ITERATIVE MODEL

It is developed to overcome the weaknesses of the waterfall model. It starts with an initial planning and ends with deployment with the cyclic interactions in between. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental), allowing software developers to take advantage of what was learned during the development of earlier parts or versions of the system. It can consist of mini waterfalls or mini V-Shaped model




## USESFULNESS (FOR HOSTEL MANAGEMENT SYSTEMS)

- Produces business value early in the development life cycle
- Better use of scarce resources through proper increment definition.
- Can accommodate some change requests between increments.
- More focused on customer value than the linear approaches.
- We can detect project issues and changes earlier.

IMPORTANT!

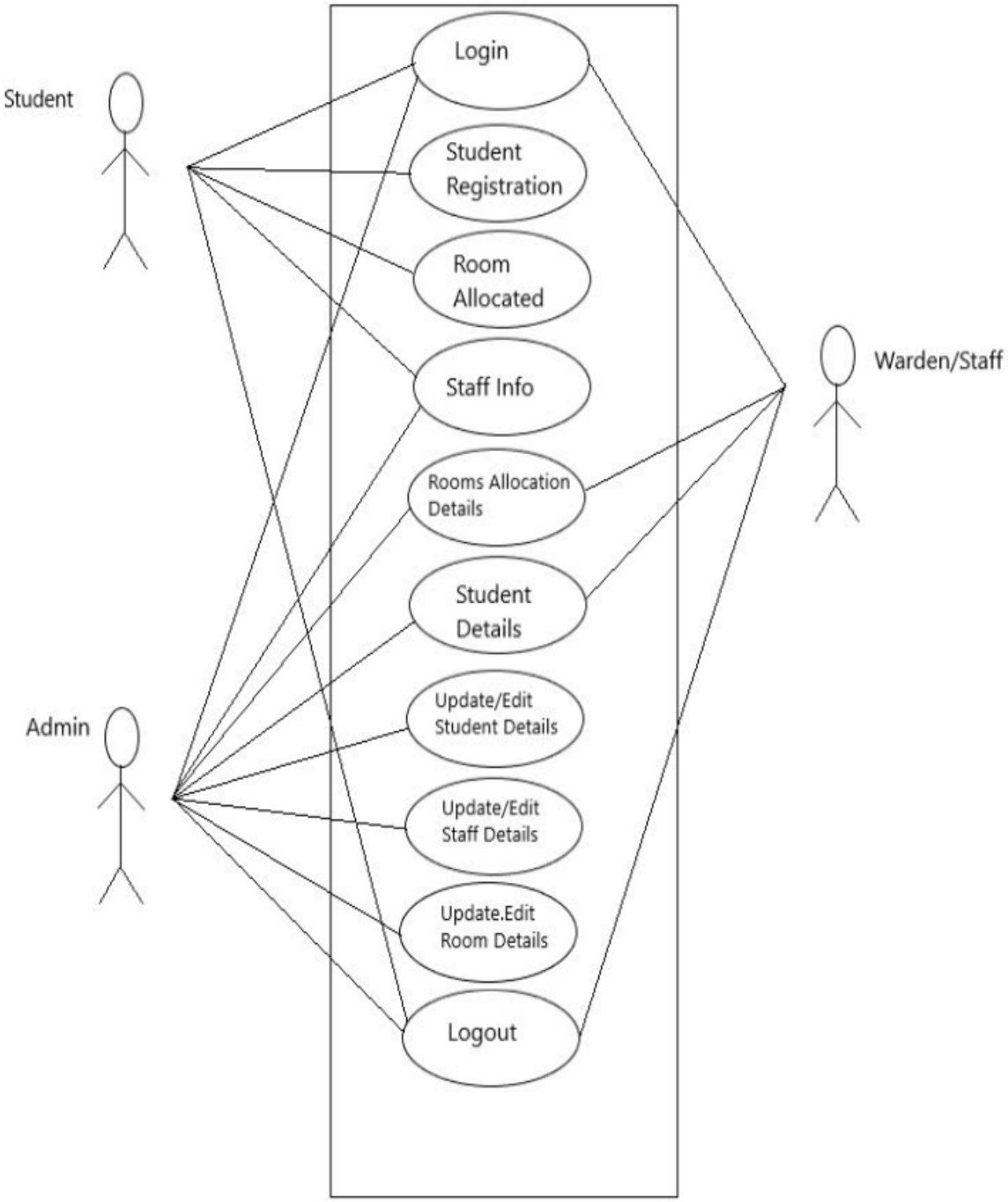
- **Results are obtained early and periodically.**
- **Parallel development can be planned.**
- **Progress can be measured.**
- **It supports changing requirements.**
- **Initial Operating time is less.**

# SYSTEM DESIGN

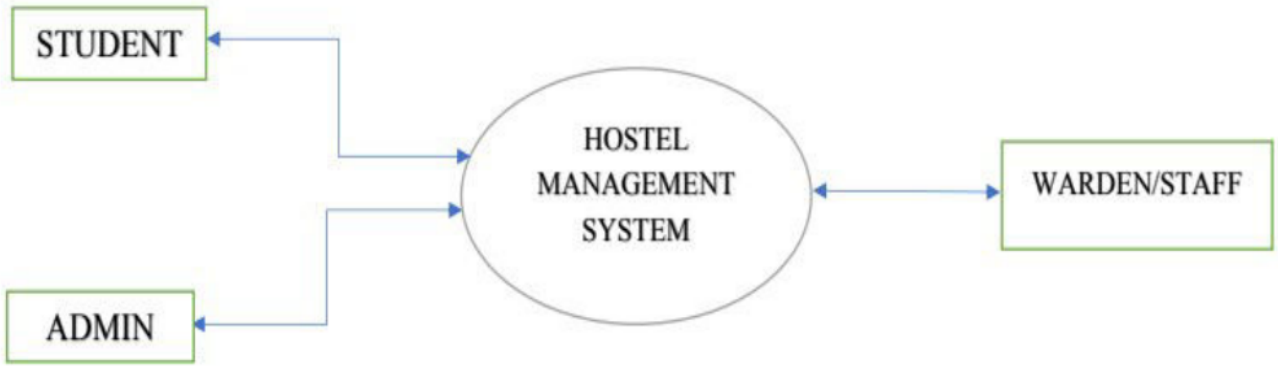
## INTRODUCTION

In this chapter we are introduce Use Case diagram, HMS system architecture, principal system object, design model and object interface.

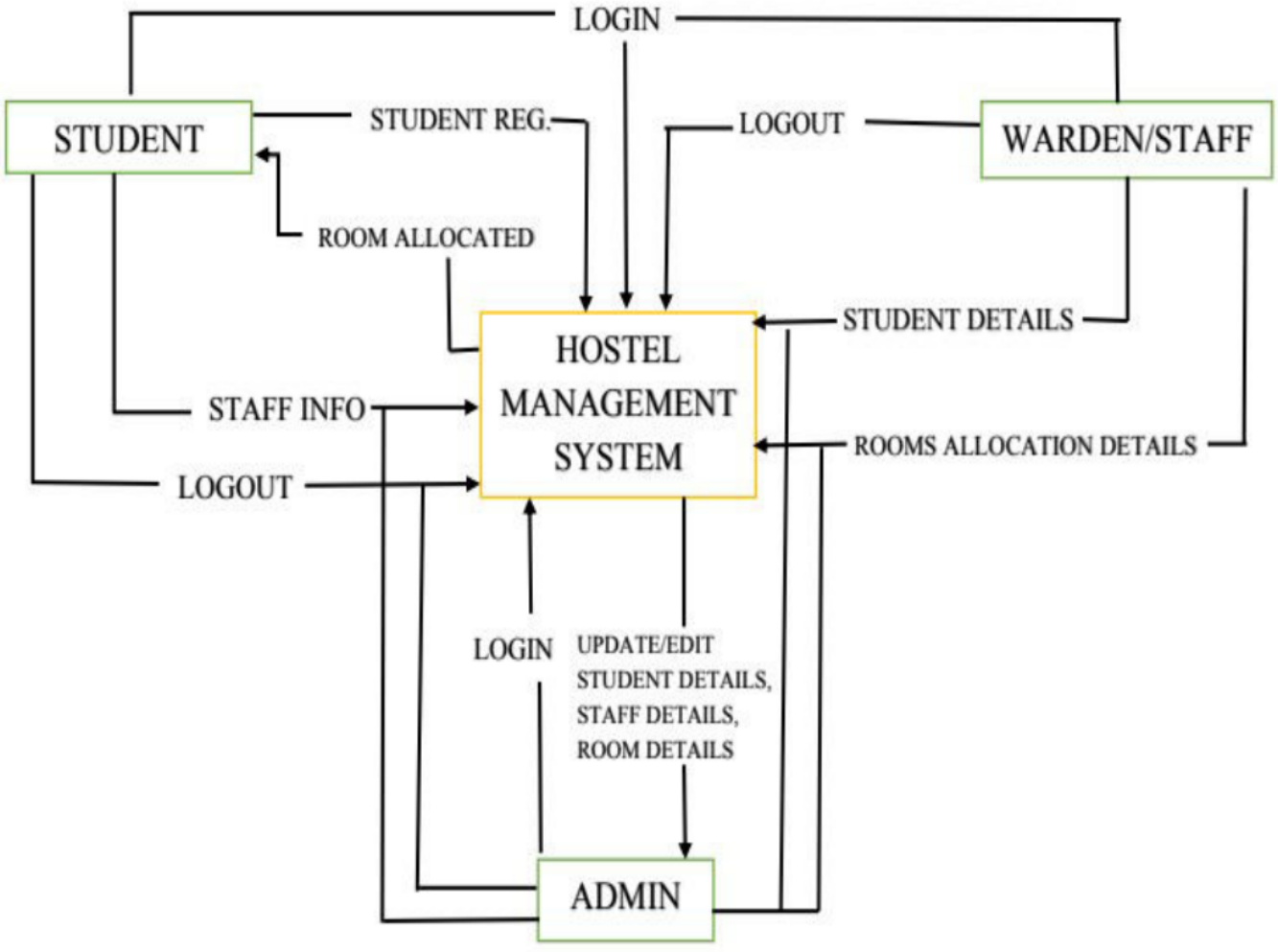
## USE CASE DIAGRAM



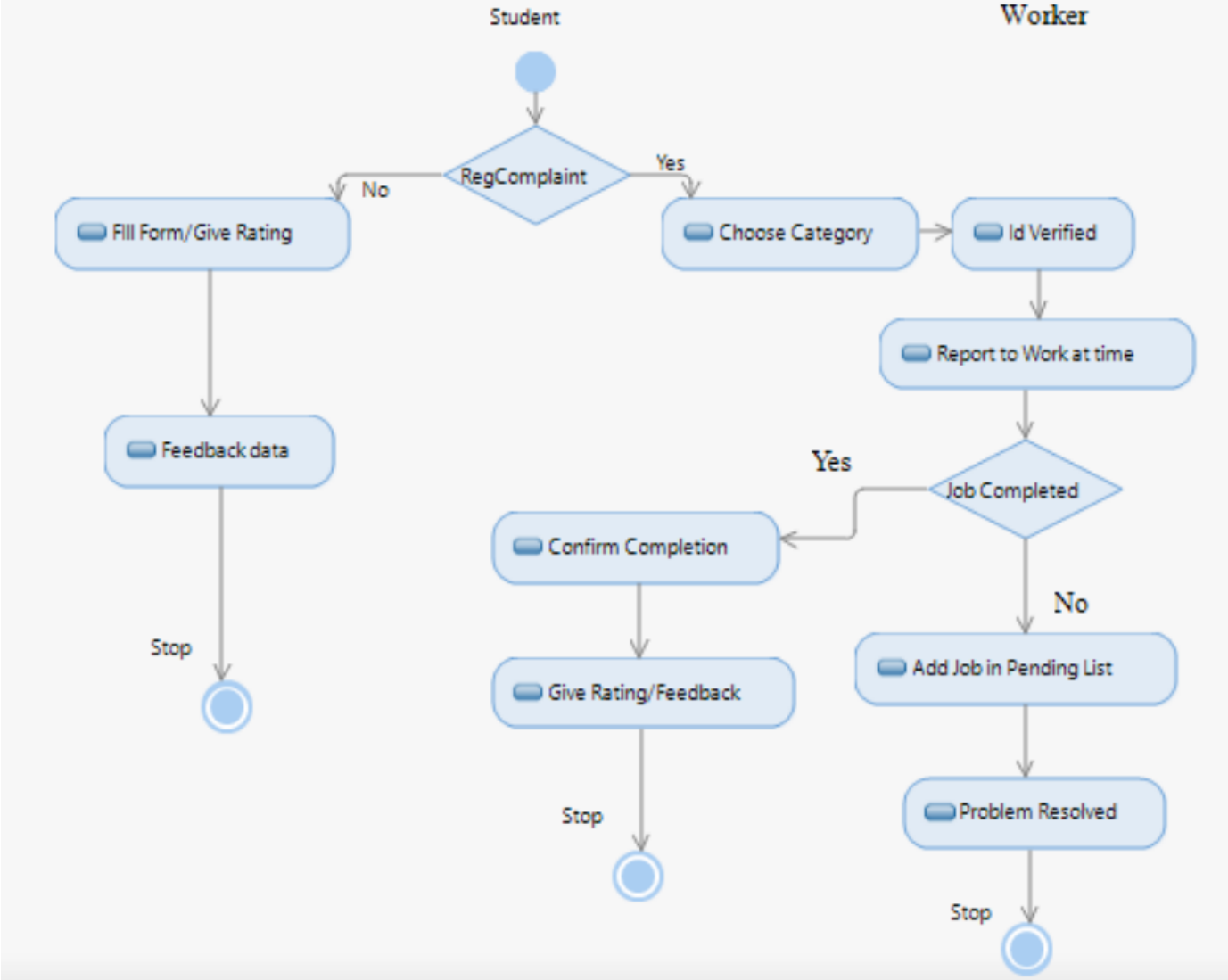
# CONTEXT DIAGRAM



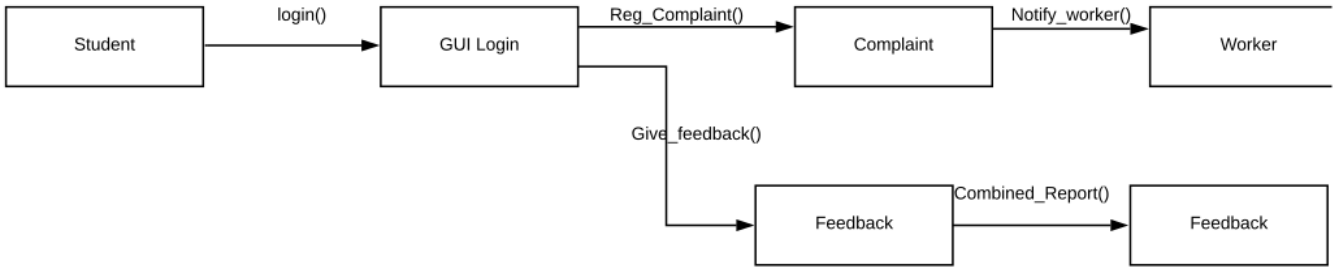

# DATA FLOW DIAGRAM



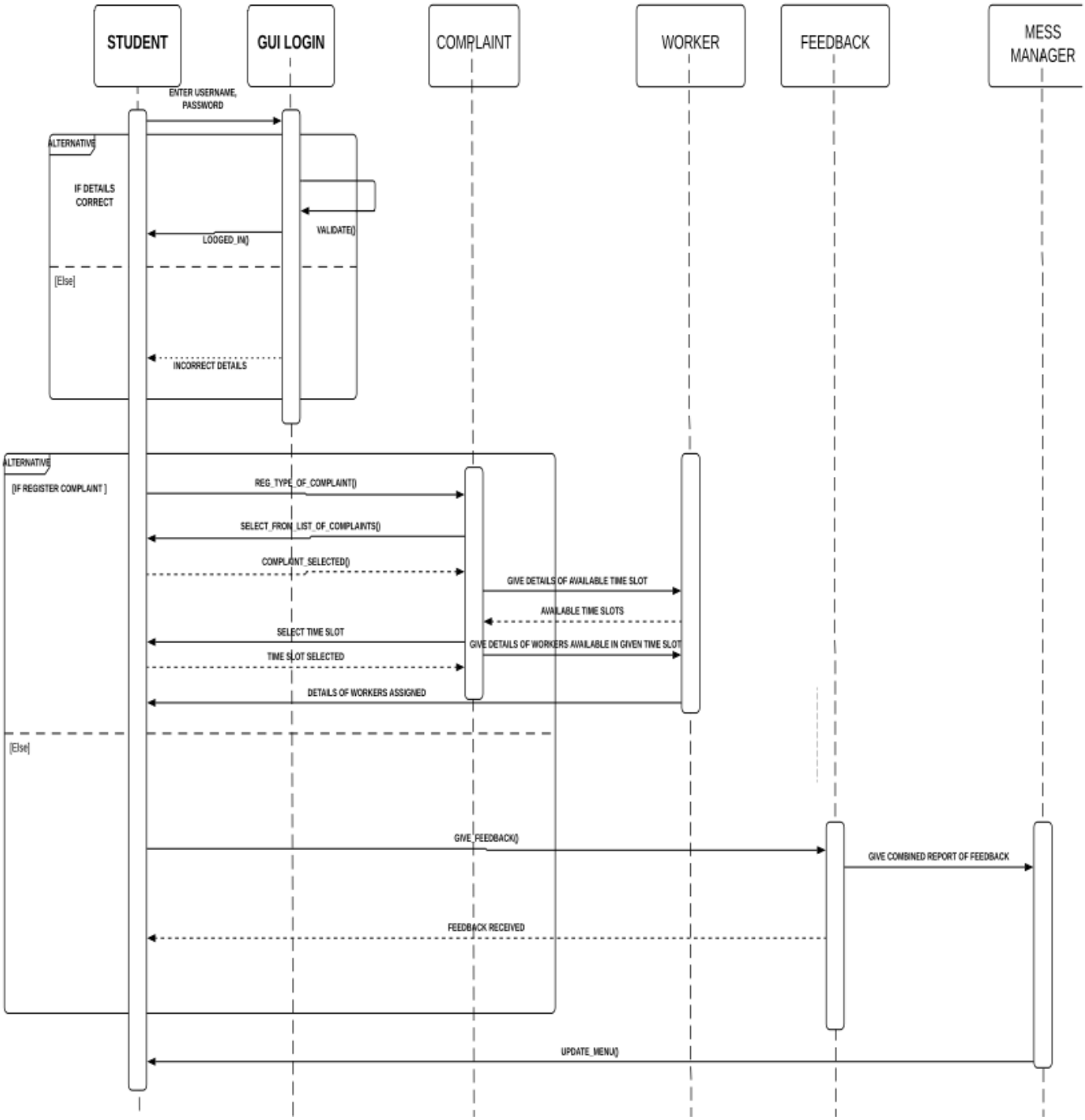
# ACTIVITY DIAGRAM



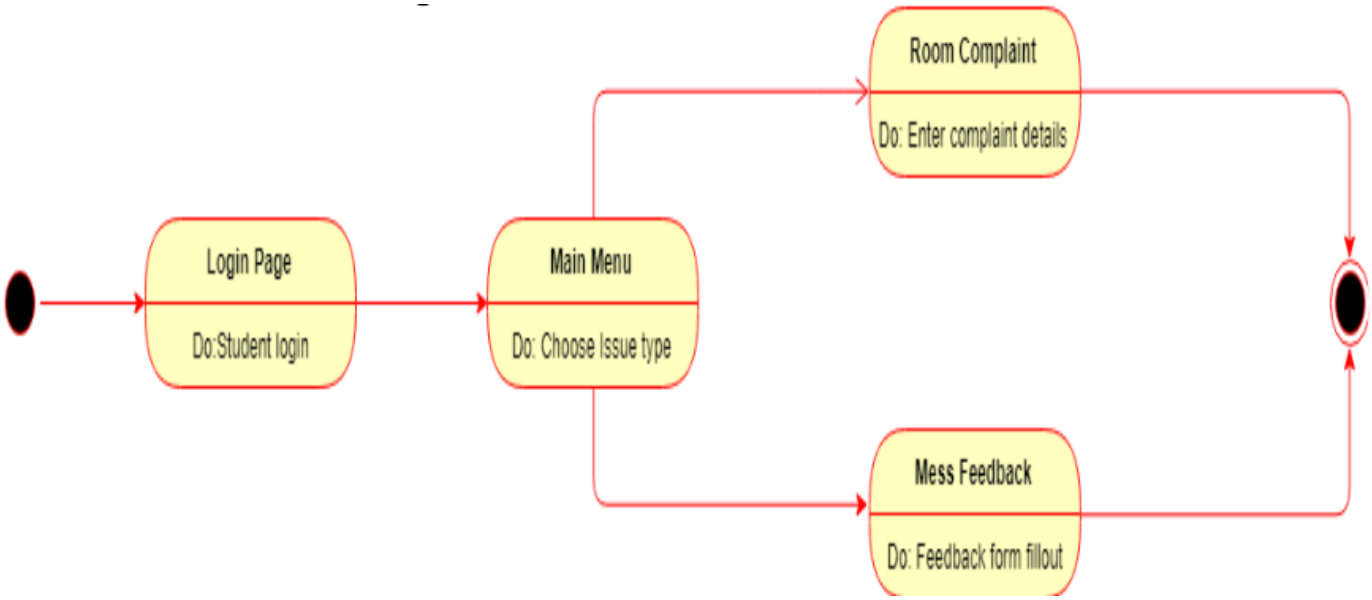

# COLLABORATION DIAGRAM



# SEQUENCE DIAGRAM



# STATE CHART DIAGRAM



# E-R DIAGRAM

Hostel Management System entities and their attributes:

### 1. Hostel Entity:

Attributes of hostel are Hostel_ID, H_name, H_Type, H_address. We are taking the entity named as a hostel because from here data will move to other entities and we will manage the database. This entity will manage the data of students in the hostel.

### 2. Rooms Entity:

Attributes of Rooms are Room_ID, Room_type. Rooms will be allotted to different students, so the students database can be accessed by the attributes of the room as well.

### 3. Student Entity:

Attributes of Student are Student_ID, S_name. The reason for choosing this entity is that students are the main entity in the hostel and we will keep the database of students and logically related things in the students entity.

### 1. Employee Entity:

Attributes of Employee are Employee_ID, Employee_name. There will be staff in the hostel. Some will work in mess, others may work as sweepers, and various other designations in hostels.

## Relationships between different entities:

- There exists a one-to-many relationship(stays) between Hostel and Student entities.
- There exists a one-to-many relationship(works) between Hostel and Employee entities.
- There exists a one-to-many relationship(accomodation) between Hostel and Rooms entities.

## Description of Hostel Management System Database:

- The details of Hostel are stored in Hostel tables, Similar to other tables as well.
- Each entity contains a Primary key and some contain both Primary and Foreign keys.
- Primary key of an entity should always be unique and not Null.
- All the entities- Hostel, Rooms, Student, Employee are normalized and then reduced to eliminate duplicates of records.

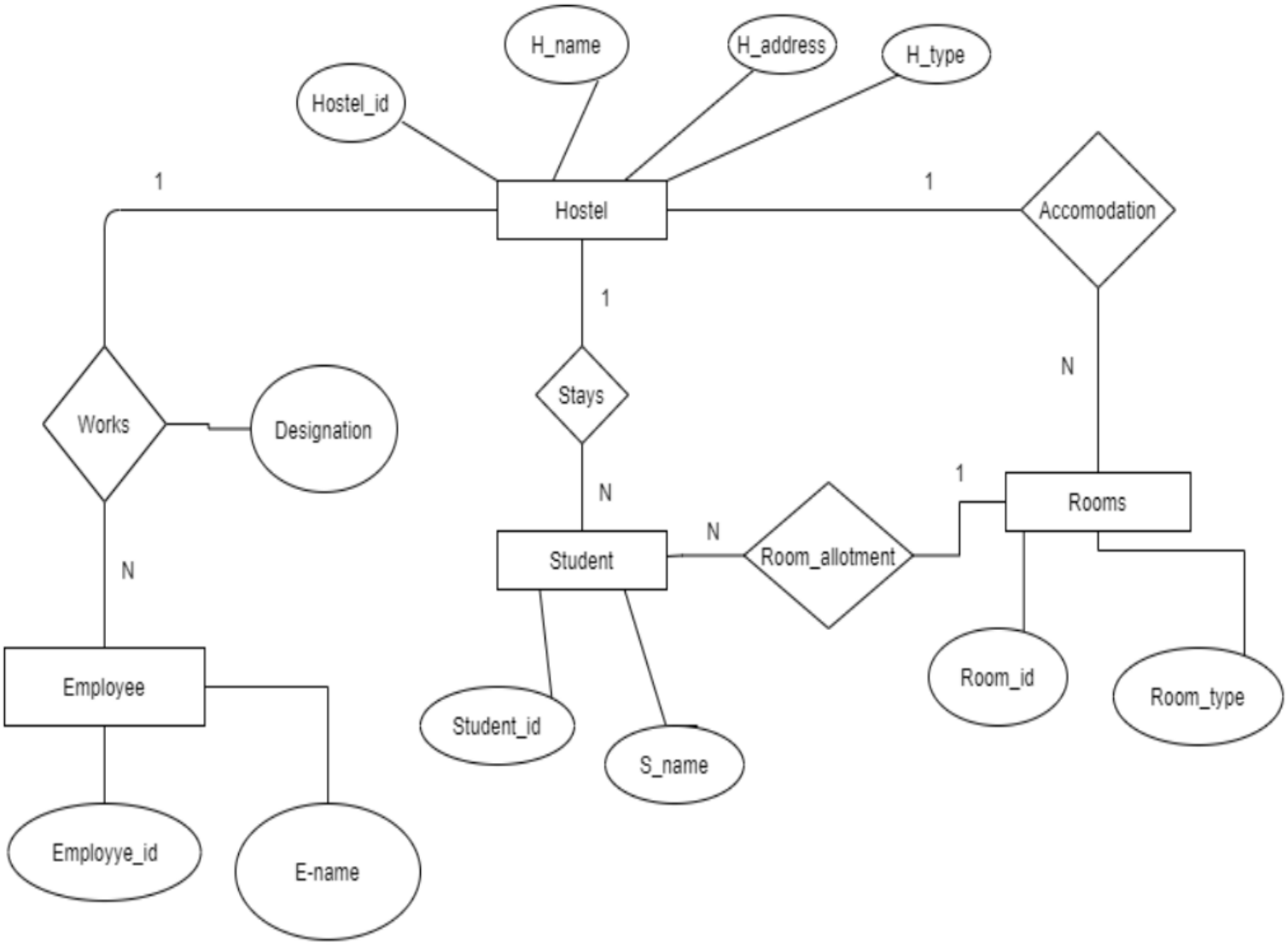



Fig:ER Diagram for Hostel Management System

# SOFTWARE REQUIREMENTS SPECIFICATION FOR

# HOSTEL MANAGEMENT SYSTEM


B18CS015

# Table of Contents

# INTRODUCTION

## 1.1 Purpose

The purpose of this project to make an automated system to carry out different operations of a hostel. The system will provide the ease, comfort of use to the staff of the hostel by performing all work on computer system rather than following a paper pen approach. This approach helps improving the reliability of data maintained and provides a fast and efficient for users of the software. Hostel Management System is a customize and user friendly software for hostel which provide. Hostel information, Hostel room information and Hostel accounts information. It helps admin to manage Student record, staff record & generating report of students etc.

## 1.2 Document Conventions

- Main Heading Titles:
  Font:Times
  Face:Bold Black
  Size:28
- Sub Heading Titles:
  Font:Times
  Face:Bold Blue
  Size:20
- Main Points:
  Font:Times
  Face:Bold black underline & displayed with bullets
  Size:14

## 1.3 Intended Audience and Reading Suggestions

The project is being designed for the students of NIT Meghalaya. The students face a lot of problems while they are staying in their hostels Thus, we want to automate the entire hostel management so that they can enjoy staying at hostels. This document also serves as a contract between the owner of the software and the developers where the owner can clearly see what and how the developers intend to do to make the software.

## 1.4 Product Scope

The software product "Hostel Management System" will be an application that will be used for maintaining the records in an organized manner and to replace old paper working on the system.This projects aims at automating the hostel management for smooth working of the hostel by automating almost all the calculations and accounting work would be accurate

- Hostel Management System is designed for hostel like (Schools and Universities).
- He checks the attested application form of students obtained from the internet & verify it with student database.
- If the students are found eligible then they allotted to the hostel room.

OBJECTIVES:

- Room Allocation
- Bill Generation
- Maintaining Student's Records
- Provide to student's Complaints
- Maintaining Employee Records

GOALS:

- Centralizing the database and thus providing consistent data to all the employees in the Hostel.
- Make the system more user friendly by providing an intensive user interface.
- Easy access through reports.
- Restricted data access to employees thus providing additional security to data.

## 1.5 Reference


- http//freestudentprojects.com
- www.iitm.ipu.ac.in
- http//bzupages.net//
- www.du.ac.in
- http://en.wikipedia.org/wiki/secondry_data
- http//www.slidshare.net/fahadchishti/hostel-management-system
- www.student projectguide.com
- http://t4tataurials.com//srs

# OVERALL DESCRIPTION

## 2.1 Product Perspective

The HMS is fully independent product. Our product is not a part of any other system. We have user interfaces. User interfaces will be divided like

- Admin Interfaces:

In this view User will add remove, new staff in the system. In administrator view,administrator will confirm newly added department. Administrator will be able to access all information about hostel staff, students & managed them.

- Staff Interfaces

At the start there will be a login screen where the user has to enter its login name & password to authenticate himself or herself.After the login a homepage will be displayed showing all the information and operations provided by Hostel Management System.

## 2.2 Product Function

“Hostel Management system” is an attempt to simulate the basic management system. The system enables to perform the following functions.

- Maintaining the resident information.
- Maintaining room information.
- Maintaining fee information.
- Maintaining employee information
- Searching, sorting and retrieval of data.

The various other functions covered by the requirement specifications which follows are provided to meet the requirements of database administrator students and administrator with role based updating and viewing rights.

The student details are filled by as:

- Student Name
- Student father’s Name
- Student year
- Room No

Following information given by the Admin:

- Student details
- Room details
- Attendance details
- Mess details
- Mess details per month

## 2.3 User Classes And Characteristics

User Characteristics:

- Educational Level:

At least user of the system should be comfortable with English language.

- Technical Expertise:

User should be comfortable using general purpose applications on the system.

User Classes with Functionality:

- Admin:

The administrator can Allot different student to different hostels.Vacate the Students for the hostels. Control the status of the fee payment.Edit the details of the students and modify the student record.

- Students:

Every student who have room in hostel have a database and a student account to access his/her data. These permition shall be showed after administrator approval. Student can check his or her data weather its true or need to some changes. Also he or she can check the monthly reports of their fines, mess bills other stuffs.

## 2.4 Operating Environment

The user will use this application to maintain the database of students and the rooms mess. The application of HMS has a very user friendly interface. The software provides accuracy along with a pleasant interface. Make the present manual system more interactive, speedy & user friendly.

## 2.5 Design And Implementation Constraints

- The developed system should run under any platform i.e. Unix, Linux, windows etc.
- There can be any security risk involved.
- Details provided by individual during his sign up should be stored in database.
- Student details can update or change by only administrator or database manager.
- The Hostel Id card is necessary to use mess
- Time constraint

## 2.6 Assumption And Dependencies

- The details related to the students, room, mess.
- Administrator is created in the database already
- Roles and tasks are predefined.

# EXTERNAL INTERFACE REQUIREMENTS

## 3.1 User Interface

The goal is to design the software used for proper management of hostels and automate the current process. The user types are listed followed

- Students
- Warden/Staff
- Admin

Our goal is to develop a software that should be easy to use for all types of users. Thus while designing the software one can assume that each user type has the following characteristics:

- The user is a computer-literate and has little or no difficulty in using the software keeping in mind the software is user friendly.
- In order to use software a user must be aware of the internal working and expected to know how things work.
- All the guidelines about the use of software will be informed to the user once the user signs up on the software or web page.

## 3.2 Hardware Interface

The section of hardware configuration is an important task related to the software development insufficient random access memory may affect adversely on the speed and efficiency of the entire system. The process should be powerful to handle the entire operations. The hard disk should have sufficient capacity to store the file and application.

- Processor :Pentium IV and above
- Processor speed : 1.4 GHz On wards
- System memory : 128 Mb minimum 256 Mb recommended
- Cache size : 512 KB RAM : 512 MB(Minimum)
- Network card : Any card can provide a 100mbps speed
- Network connection : UTP or Coaxile cable connection
- Printer : Inkjet/Laser Color printer provides at least 1000 Dpi
- Hard disk : 80Gb
- Monitor : SVGA Color 15”
- Mouse : 104 keys US Key Serial,
- USB or PS/2 Modem : 56.6 Kbps

## 3.3 Software Interface

A major element in building a system is the section of compatible software since the software in the market is experiencing in geometric progression. Selected software should be acceptable by the firm and one user as well as it should be feasible for the system. This document gives a detailed description of the software requirement specification. The study of requirement specification is focused specially on the functioning of the system. It allow the developer or analyst to understand the system, function to be carried out the performance level to be obtained and corresponding interfaces to be established.

- Front end tool : HTML,CSS,JAVASCRIPT
- Backed : SQL/XML,JSP,C++
- Operating system : Windows (XP/7/8/10)/Linux
- Client Side : HTML, Photoshop

## 3.4 Communication Interface

The system shall be a standalone product that does not require any communication interfaces.

# SYSTEM FEATURE

The functions which are used in this project and their functioning

- Student _detail () - To enter the details of the students
- Room details () - To enter the details of hostel rooms
- Attendances details () - Maintain the student attendance detail.
- Mess _details - To keep the record of challan of mess bill.

## 1. Student:

- **Request for room:** The student will request for room.
- **Pay the Bill:** The student will pay the bill to the manager.
- **Check its Profile:** The registered student will see his or her profile.
- **Check Mess Bill Report**: The student will see the mess report.

## 2. Admin:

- **Add students:** The HMS allows the administrator to add new student in to the database.
- **Room Allocation:** The HMS will allocate a room to student. The room no will store in user profile.
- **Delete Student:** The HMS allows the administrator to remove the student.
- **Mess account:** A mess account also will generate.

## 3. Database Administrator:

- **Mess Bill calculation:** The database administrator can calculate the student mess bill.
- **Search Student:** The HMS shall allow the students to search from database according to different criteria
- **Check the availability of Room:** The HMS allows to the database administrator to check the availability of room

# OTHER NON-FUNCTIONAL REQUIREMENTS

There are the following non-functional requirements of hostel management system

## 1. Performance Requirements

The application shall be based on java and has to be run on any platform. the application shall task initial load time depending on performance of operating system. The performance shall depend upon hardware and software components of the computer.

## 2. Safety Requirements

The database may get crashed any certain time due to the virus or operating system failure. Therefore, it is required to take the backup of database.

## 3. Security Requirements

This project provides a genuine security to all those individuals who are having their account on the database as they are password protected. This is very important aspect of the design and should cover areas of hardware reliability fall, back procedures, physical security of data and provision for detection of fraud and abuse.

## 4. Software Quality Attributes

- **Reliability and Availability:** The project shall provide storage of all database on redundant computer with oracle database.
- **Maintainability:**The system shall provide the capability to back up the database.
- **Portability:**The Hostel Management System shall run on any Microsoft Windows environment.
- **Flexibility:** Ability to add new features to the system and handle them conveniently.
- **Re-usability:**What is ability to use the available components of the system in other.
- **Efficiency:**How much less number of resources and time are required to achieve a particular task through the system.

## 5. Business Rules

The system is desired to handle all the activities of the students as well as the administrative level. The system will have the ability to search the student's information about his/her room mess and all the other things. Once the current and previous record is entered then the database will be updated for the new students automatically. This system is for hostel so that the primary users of the system are the students and the administrative penal. The main constraint is the system registration is valid if the department has been approving that student is valid for the department. The constrains are the amount of the hostel dues and the mess dues that are calculated in the system. These dues should be paid within 10 days. If anyone could not do the payment for some reason the system will notify the name of the student.

- System will use warden of the hostel.
- The Hostel id card is necessary to use mess.
- Time constraint

# CODING REPORT

## HOSTEL MANAGEMENT SYSTEM

B18CS015

# CONTENTS



<CODE/>

# LOGIN PAGE

## FRONT-END CODE:

```
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JCheckBox jCheckBox1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel4;
private javax.swing.JPasswordField
jPasswordField1;
private javax.swing.JTextField jTextField2;
```

# LOGIN PAGE

## BACK-END CODE:

Login Button:

```
private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
if(jTextField2.getText().equals("hostel") &&
jPasswordField1.getText().equals("admin"))
{setVisible(false);
new home().setVisible(true);}
else
{JOptionPane.showMessageDialog(null, "Incorrect
Username or Password");}
}
```

Checkbox Button:

```
private void
jCheckBox1ActionPerformed(java.awt.event.ActionEvent evt) {
if(jCheckBox1.isSelected())
{jPasswordField1.setEchoChar((char)0);
}else
{jPasswordField1.setEchoChar('*');}}
}
```

# HOME PAGE

## FRONT-END CODE:

```
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton12;private
javax.swing.JButton jButton13;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JLabel jLabel1;
```

# HOME PAGE

## BACK-END CODE:

Source Code:

```
private void
formWindowGainedFocus(java.awt.event.WindowEvent evt) {
// TODO add your handling code here:
jButton1.setForeground(new JButton().getForeground());
jButton1.setBackground(new JButton().getBackground());
jButton2.setForeground(new JButton().getForeground());
jButton2.setBackground(new JButton().getBackground());
jButton3.setForeground(new JButton().getForeground());
jButton3.setBackground(new JButton().getBackground());
jButton4.setForeground(new JButton().getForeground());
jButton4.setBackground(new JButton().getBackground());
jButton5.setForeground(new JButton().getForeground());
jButton5.setBackground(new JButton().getBackground());
}
```

New Student Button:

```
private void
jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
jButton2.setForeground(Color.white);
jButton2.setBackground(Color.gray);
new NewStudent().setVisible(true);}
```

Manage Room Button:

```
private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
jButton1.setForeground(Color.white);
jButton1.setBackground(Color.gray);
new ManageRoom().setVisible(true);
}
```

# HOME PAGE

## BACK-END CODE:

Update Delete Students button:

```
private void
jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
jButton3.setForeground(Color.white);
jButton3.setBackground(Color.gray);
new UpdateDeleteStudents().setVisible(true);
}
```

Student Fees Button:

```
private void
jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
jButton4.setForeground(Color.white);
jButton4.setBackground(Color.gray);
new StudentFees().setVisible(true);
}
```

Logout Button:

```
private void
jButton13ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
int a=JOptionPane.showConfirmDialog(null,"Do you want to
Logout?","Select",JOptionPane.YES_NO_OPTION);
if(a==0){
setVisible(false);
new login().setVisible(true);}}
```

Exit Button:

```
private void
jButton12ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
int a=JOptionPane.showConfirmDialog(null,"Do you want to
Exit?","Select",JOptionPane.YES_NO_OPTION);
if(a==0) {System.exit(0);}}
```

# MANAGE ROOM/HOSTEL

## FRONT-END CODE:

```
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;private javax.swing.JButton
jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JCheckBox jCheckBox1;
private javax.swing.JCheckBox jCheckBox2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JTable jTable1;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
```

# MANAGE ROOM/HOSTEL

## BACK-END CODE:

### Connecting database to table:

```
public void tableDetails()
{DefaultTableModel dtm=(DefaultTableModel)
jTable1.getModel();
dtm.setRowCount(0);
try
{Connection con = ConnectionProvider.getCon();Statement st=con.createStatement();
ResultSet rs=st.executeQuery("select *from room");
while(rs.next())
{dtm.addRow(new
Object[]{rs.getString(1),rs.getString(2),rs.getString(3)});}
}catch(Exception e)
{JOptionPane.showMessageDialog(null, e);}}
```

### Save Button:

```
private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
String RoomNumber=jTextField1.getText();
String activate;
String roomStatus="Not Booked";
if(jCheckBox1.isSelected())
{activate="Yes";}
else
activate="No";
try{Connection con=ConnectionProvider.getCon();
PreparedStatement ps=con.prepareStatement("insert into
room values(?,?,?)");ps.setString(1,RoomNumber);ps.setString(2,activate);
ps.setString(3,roomStatus);
ps.executeUpdate();
JOptionPane.showMessageDialog(null, "Successfully
Updated");
tableDetails();
clear();}catch(Exception e){JOptionPane.showMessageDialog(null, "Room already
exists");}
```

# MANAGE ROOM/HOSTEL

## BACK-END CODE:

Update Button:

```
private void
jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
String roomnumber=jTextField2.getText();
String activate;
if(jCheckBox2.isSelected())
activate="Yes";
elseactivate="No";
try{Connection con=ConnectionProvider.getCon();
Statement st=con.createStatement();
st.executeUpdate("update room set
activate='"+activate+"'where number='"+roomnumber+"'");
JOptionPane.showMessageDialog(null, "Successfully
Updated");tableDetails();
clear();}
catch(Exception e){
JOptionPane.showMessageDialog(null, e);}}
```

Delete Button:

```
private void
jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
String roomnumber=jTextField2.getText();
try{Connection con=ConnectionProvider.getCon();
Statement st=con.createStatement();
st.executeUpdate("delete from room where
number='"+roomnumber+"'");
JOptionPane.showMessageDialog(null, "Successfully
Deleted");
tableDetails();
clear();}catch(Exception e){
JOptionPane.showMessageDialog(null, e);}}
```

# STUDENTS MANAGER

## FRONT-END CODE:

```
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;private javax.swing.JButton
jButton3;
private javax.swing.JComboBox<String> jComboBox1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JTextField jTextField4;
```

# STUDENTS MANAGER

## BACK-END CODE:

### Save Button:

```
private void
jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
String mobilenumber=jTextField1.getText();
String name=jTextField2.getText();
String fathername=jTextField3.getText();
String mothername=jTextField4.getText();String email=jTextField5.getText();
String address=jTextField6.getText();
String college=jTextField7.getText();
String aadhaar=jTextField8.getText();
String roomnumber=(String)jComboBox1.getSelectedItem();
String status="living";
try{Connection con=ConnectionProvider.getCon();
PreparedStatement ps=con.prepareStatement("insert into
student values(?,?,?,?,?,?,?,?,?,?)");
ps.setString(1,mobilenumber);
ps.setString(2,name);
ps.setString(3,fathername);
ps.setString(10,status);
ps.executeUpdate();
PreparedStatement ps1=con.prepareStatement("update
room set roomStatus='Booked' where number=?");
ps1.setString(1, roomnumber);
ps1.executeUpdate();
JOptionPane.showMessageDialog(null, "Successfully Updated");
clear();}
catch(Exception e){
JOptionPane.showMessageDialog(null, e); }}Clear Function:
public void clear(){
roomNumber();
}
```

# ROOM ALLOTMENT

## FRONT-END CODE:

```
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;
private javax.swing.JComboBox<String> jComboBox1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JTextField jTextField4;
private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
private javax.swing.JTextField jTextField7;
private javax.swing.JTextField jTextField8;
private javax.swing.JTextField jTextField9;
```

# ROOM ALLOTMENT

## BACK-END CODE:

### Search Button:

```
private void
jButton2ActionPerformed(java.awt.event.ActionEvent evt) {String mobileNo;
mobileNo = jTextField1.getText();
try
{
Connection con=ConnectionProvider.getCon();
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("select *from student where
mobileNo='"+mobileNo+"'");
if(rs.next())
{
jTextField1.setEditable(false);
jTextField2.setText(rs.getString(2));
jTextField3.setText(rs.getString(3));
jTextField4.setText(rs.getString(4));
jTextField5.setText(rs.getString(5));
jTextField6.setText(rs.getString(6));
jTextField7.setText(rs.getString(7));
jTextField8.setText(rs.getString(8));
jTextField9.setText(rs.getString(9));
jTextField9.setEditable(false);
if(rs.getString(10).equals("living"))
{jComboBox1.addItem("living");
jComboBox1.addItem("out");
}else{
jComboBox1.addItem("out");
jComboBox1.addItem("living");}}
else
{JOptionPane.showMessageDialog(null, "Student does not
exist");
clear();}
catch(Exception e){
JOptionPane.showMessageDialog(null, e);
}
}
```

# ROOM ALLOTMENT

## BACK-END CODE:

Save Button:

```
private void
jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
String mobileNo=jTextField1.getText();
String name=jTextField2.getText();
String fathername=jTextField3.getText();
String mothername=jTextField4.getText();
String college=jTextField7.getText();
String aadhaar=jTextField8.getText();
String roomNo=jTextField9.getText();
String status=(String)jComboBox1.getSelectedItem();
try{
Connection con=ConnectionProvider.getCon();
Statement st=con.createStatement();
if(status.equals("living"))
st.executeUpdate("update room set
roomStatus='Booked' where number='"+roomNo+"'");
elsest.executeUpdate("update room set roomStatus='Not
Booked' where number='"+roomNo+"'");
PreparedStatement ps=con.prepareStatement("update
student set
name=?,father=?,mother=?,email=?,address=?,college=?,aadhar
=?,status=? where mobileNo=?");
ps.setString(1, name);
ps.setString(2, fathername);
ps.setString(3, mothername);

ps.setString(9, mobileNo);
ps.executeUpdate();
JOptionPane.showMessageDialog(null,"Successfully
Updated");
clear();}
catch (Exception e){
JOptionPane.showMessageDialog(null, e);}
```

# FEES MANAGEMENT

## FRONT-END CODE:

```
private javax.swing.JButton jButton1;private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JTextField jTextField4;
private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
```

# FEES MANAGEMENT

## BACK-END CODE:

Search Button:

```
private void
jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
String mobileNo=jTextField1.getText();
SimpleDateFormat dFormat=new
SimpleDateFormat("MMM-yyyy");
Date date=new Date();
String month=dFormat.format(date);try
{Connection con=ConnectionProvider.getCon();
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("select *from student where
mobileNo='"+mobileNo+"' and status='living'");
if(rs.next())
{jTextField1.setEditable(false);
jTextField2.setText(rs.getString(2));
j(9));
jTextField5.setText(month);
jTextField6.setText("8000");
}else
{JOptionPane.showMessageDialog(null,"Student does not
exist");
clear();}
tableDetails();
ResultSet rs1=st.executeQuery("select *from fees inner join
student where student.status='living' and
fees.month='"+month+"' and fees.mobileNo='"+mobileNo+"'
and student.mobileNo='"+mobileNo+"'");
if(rs1.next())
{jButton3.setVisible(false);
JOptionPane.showMessageDialog(null,"Fees Paid
Already");
}}catch(Exception e)
{JOptionPane.showMessageDialog(null, e);}}
```

# FEES MANAGEMENT

## BACK-END CODE:

### Pay Button:

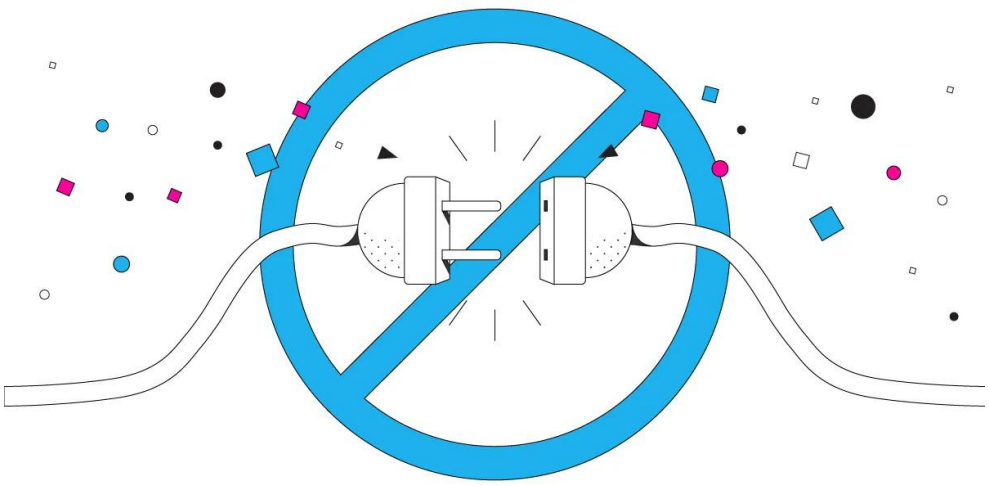
```
private void
jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
String mobileNo=jTextField1.getText();
String month=jTextField5.getText();
String amount=jTextField6.getText();
try
{
Connection con=ConnectionProvider.getCon();
PreparedStatement ps=con.prepareStatement("insert into
fees values(?,?,?)");
ps.setString(1, mobileNo);
ps.setString(2, month);
ps.setString(3, amount);
ps.executeUpdate();
tableDetails();
JOptionPane.showMessageDialog(null, "Successfully
Updated");
clear();
}
catch(Exception e)
{
JOptionPane.showMessageDialog(null, e);
}
}Clear Button:
private void
jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
clear();
}
```

# CONNECTION PROVIDER

## FRONT-END CODE:

### Source Code:

```
public class ConnectionProvider {
public static Connection getCon()
{
try
{
Class.forName("com.mysql.jdbc.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:330
6/hostel","root","12345");
return con;
}
catch(Exception e)
{
return null;
}
}
}
```



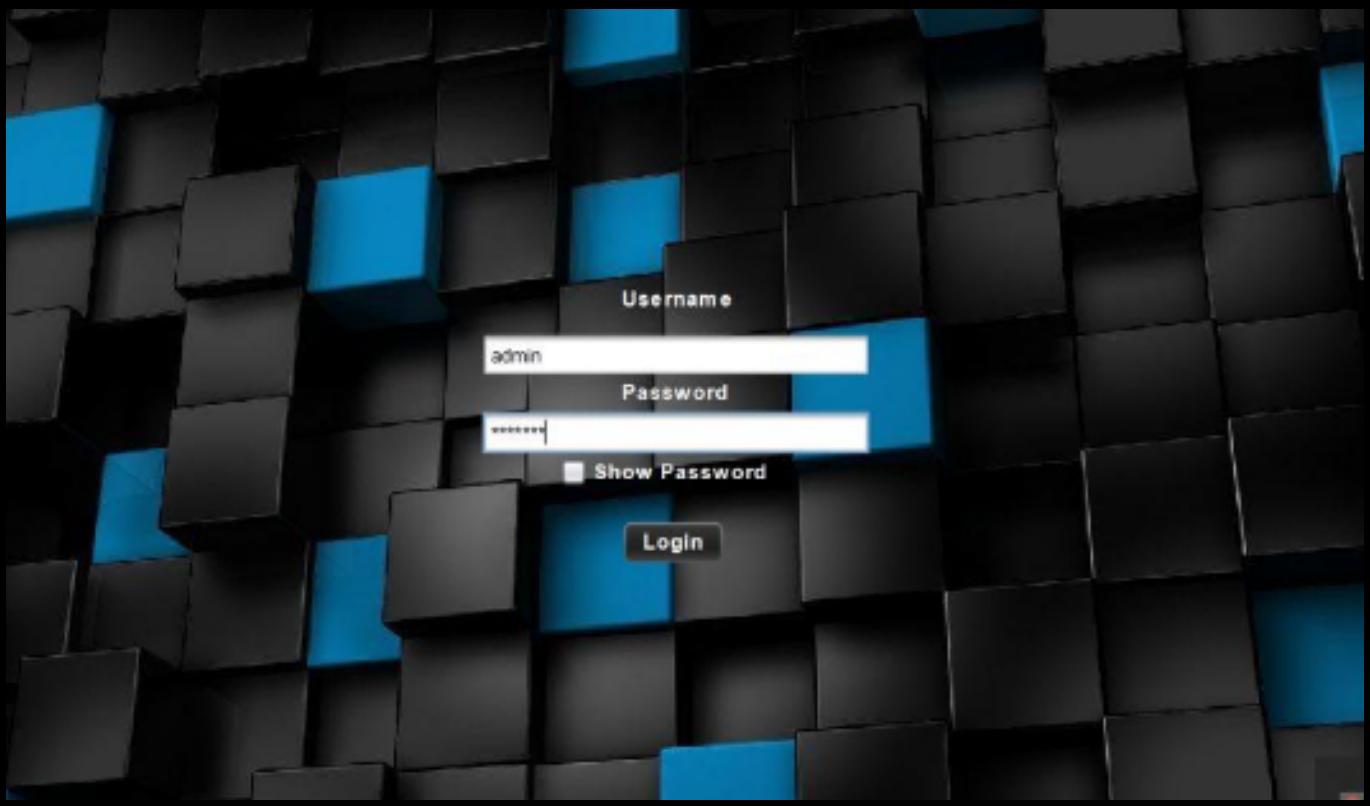
# USER MANUAL

## HOSTEL MANAGEMENT SYSTEM

B18CS015

# CONTENTS

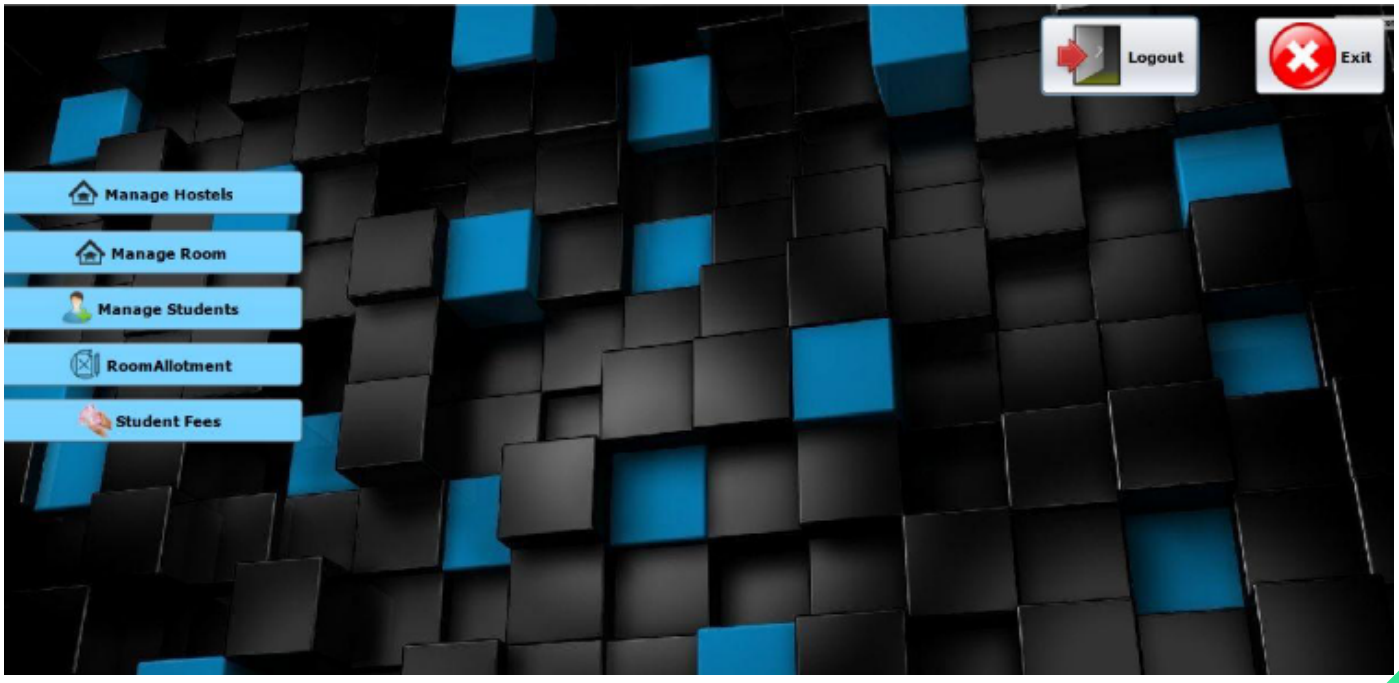
# LOGIN PAGE



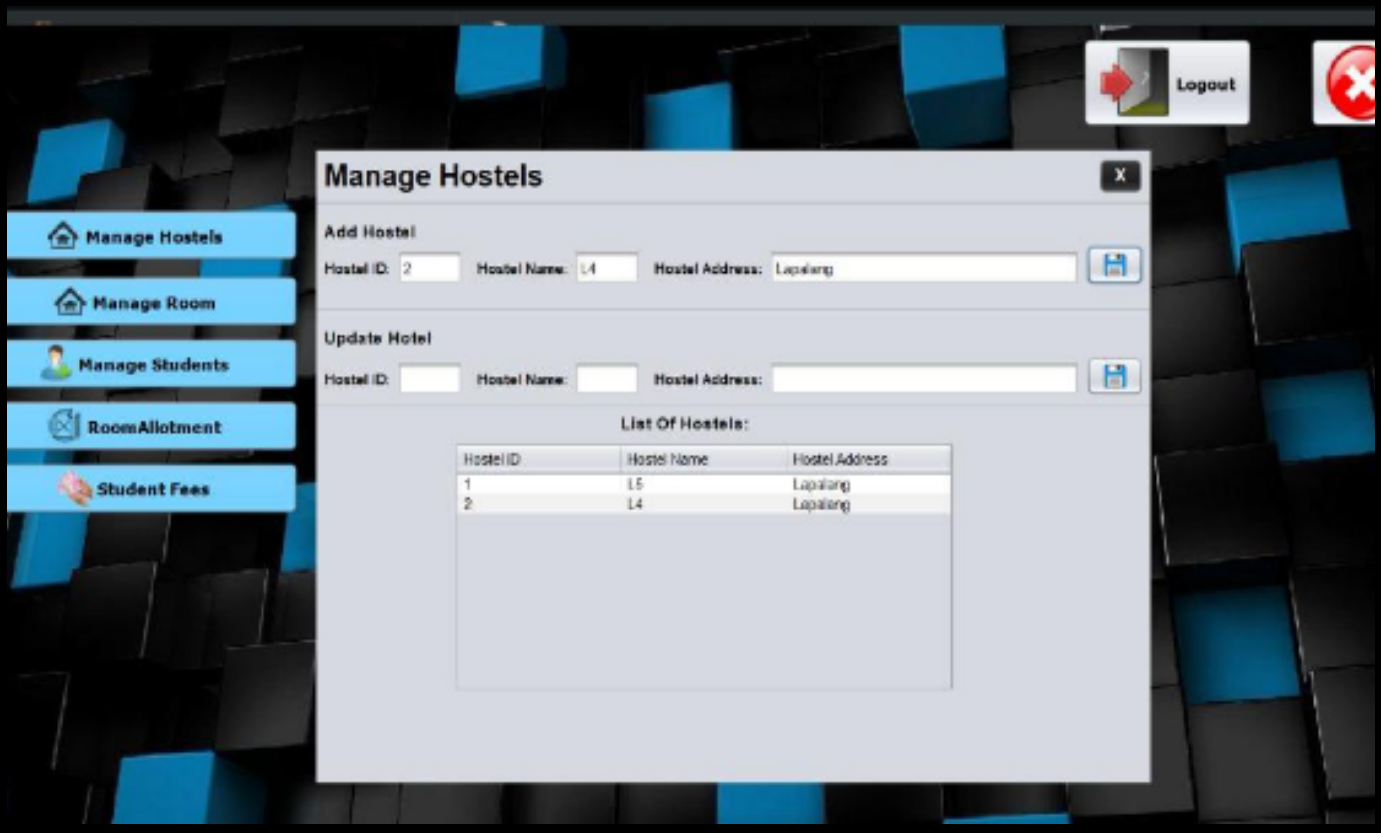

USERS WILL BE PROVIDED WITH AN AUTHORIZED USER NAME AND PASSWORD ALONG WITH THE SOFTWARE. THEY CAN LOG IN TO THE MANAGEMENT SYSTEM USING THE CREDENTIALS.ON CLICKING THE LOGIN BUTTON , THE USER WILL ADVANCE TO THE HOMEPAGE AND START USING THE APPLICATION.

# HOME PAGE

THE HOMEPAGE BASICALLY CONTAINS A NAVIGATION BAR WHICH TAKES THE USER TO DIFFERENT PAGES TO PERFORM DIFFERENT ACTIVITIES.



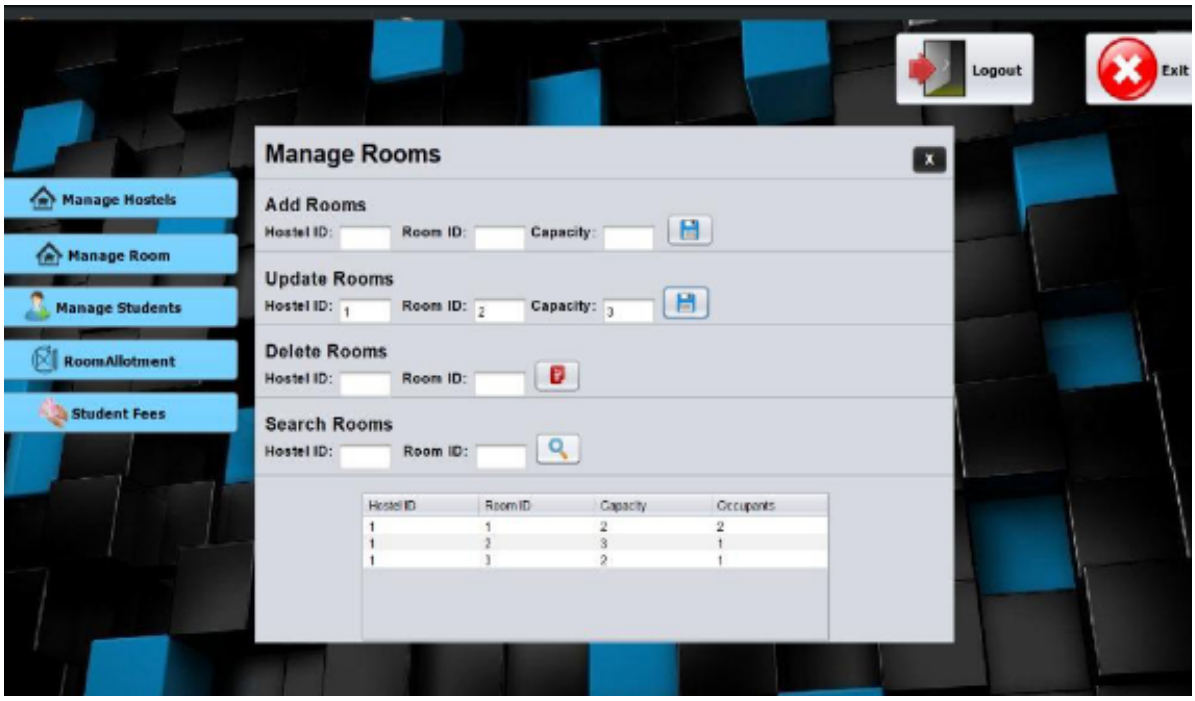
# MANAGING HOSTELS




THIS PAGE SHOWS THE LIST OF ALL HOSTELS ADDED TO THE SYSTEM BY THE USER. USER CAN ADD/UPDATE THE HOSTEL DETAILS IN THIS PAGE. THE HOSTEL DETAILS INCLUDE: HOSTEL ID , HOSTEL NAME AND HOSTEL ADDRESS.

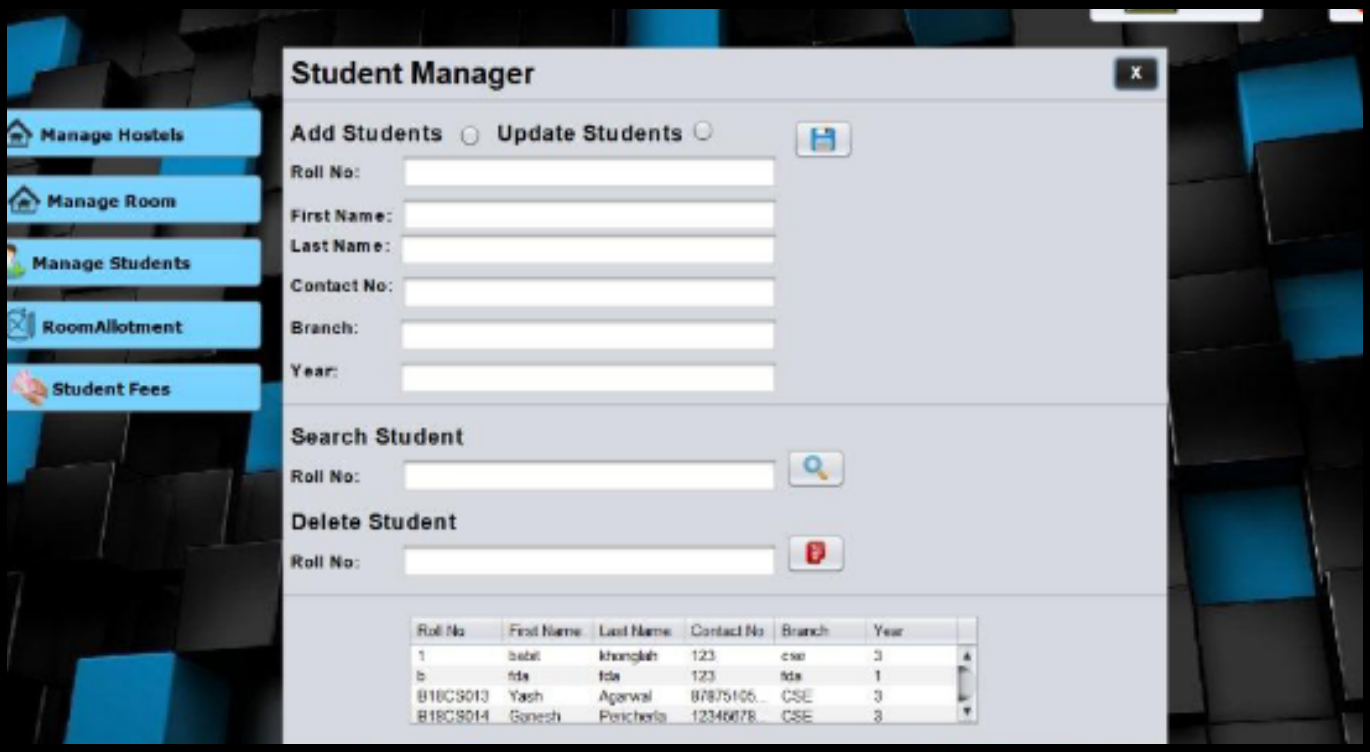
# MANAGING ROOMS

HERE, THE USER CAN LOOK AT THE ROOM DETAILS OF EACH HOSTEL. USER CAN ADD/UPDATE/DELETE THE ROOM DETAILS IN THIS PAGE. THE ROOM DETAILS INCLUDE :HOSTEL ID, ROOM ID , TOTAL CAPACITY AND NUMBER OF OCCUPANTS.




# STUDENT MANAGER

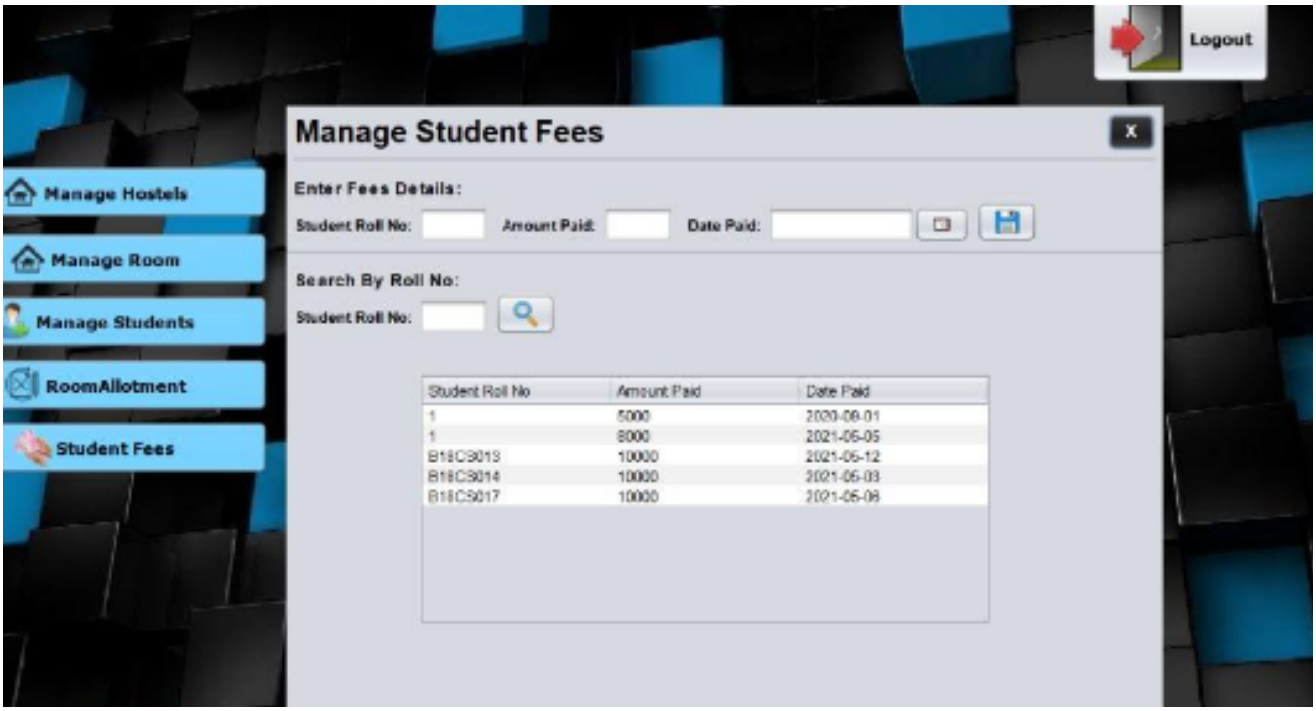
HERE, THE USER CAN VIEW DETAILS OF ALL THE HOSTELERS. THE HOSTEL MANAGER CAN ALSO ADD A NEW STUDENT, UPDATE THE EXISTING DETAILS OF THE HOSTELER AND DELETE THE DETAILS OF OUTGOING HOSTELER AS WELL.




# ROOM ALLOTMENT

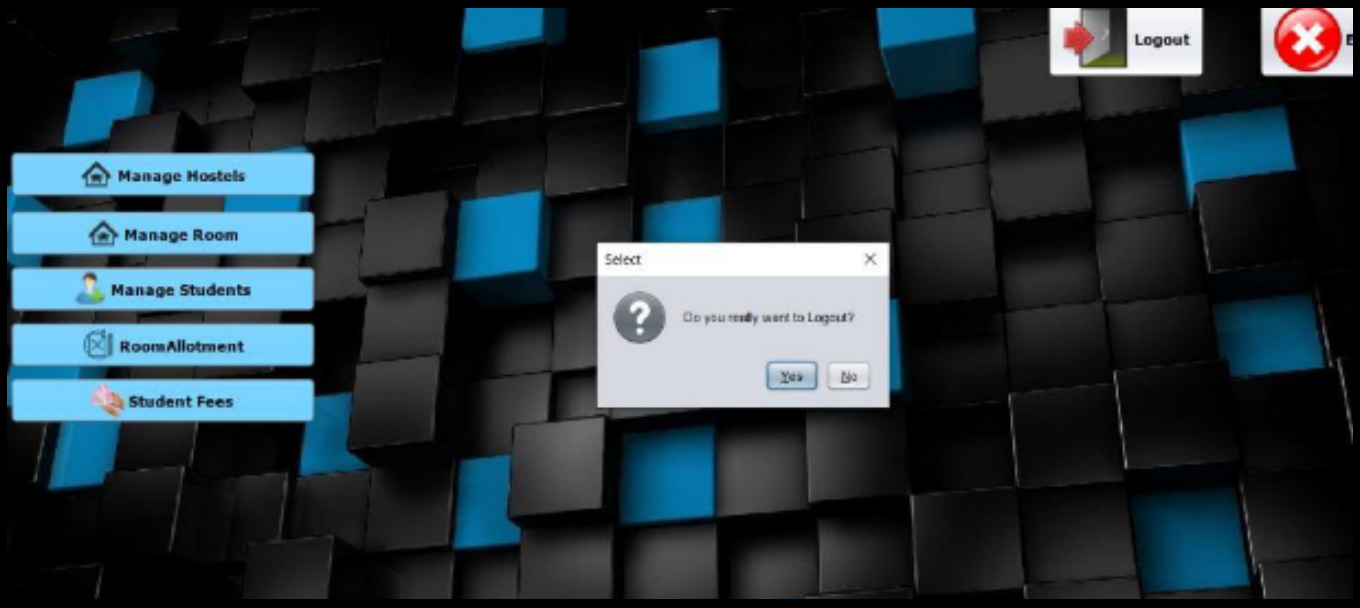
HERE, THE HOSTEL MANAGER CAN VIEW THE ROOMS ALLOTTED THE STUDENTS. HOSTEL MANAGER CAN ALLOCATE/DE-ALLOCATE THE ROOM FOR THE STUDENTS IN THIS PAGE.

# FEES MANAGEMENT

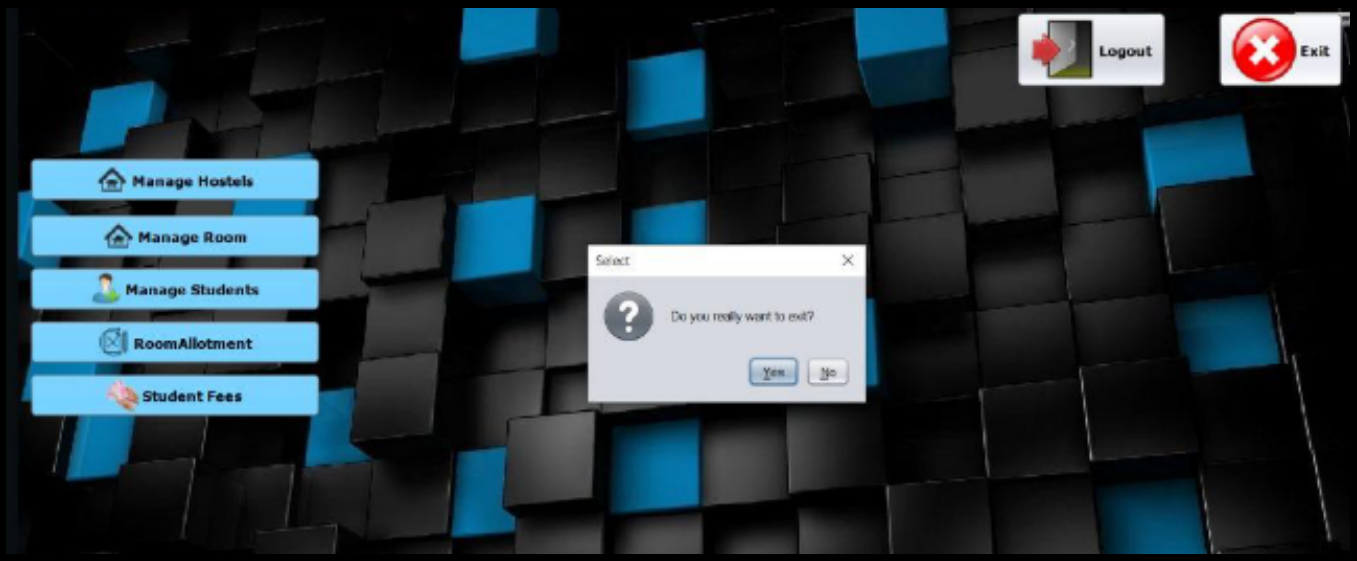



HERE, THE USER CAN UPDATE THE FEE DETAILS OF EVERY HOSTELER AND CAN VIEW THE DETAILS OF PAYMENTS LIKE AMOUNT PAID AND DATE OF TRANSACTION.

# LOGOUT AND EXIT




USER CAN LOGOUT FROM HIS/HER ACCOUNT BY CLICKING ON LOGOUT BUTTON.




USER CAN DIRECTLY EXIT FROM THE APPLICATION BY CLICKING ON EXIT BUTTON.