# Lab Experiments Report

## Hostel Management System

## 1. Develop requirements specification for a given problem(Hostel Management System) (28/Jan2022)

## 1. Introduction

### 1.1 Purpose

The purpose of this SRS document is to provide a detailed overview of our software product, its parameters and goals. This document describes the project's intended audience and its user interface, hardware and software requirements for the product. It defines how our end client, team and audience see the product and its functionality.

### 1.2 Intended Audience

The intended audience is the students who are staying in hostels. The students face a lot of problems while they are staying in their hostels, so we want to automate the entire hostel management so that they can enjoy staying at hostels without facing any problems and if there are no problems faced by students in the hostel then few more students would be interested to use the hostel facility.

### 1.3 Intended Use

This document is to provide a detailed overview of our software product, its parameters and goals and also serves as a contract between the manager of the software and the developers and testers where the manager can clearly see what and how the developers intend to do to make the software.

### 1.4 Scope

The goal is to design robust software for hostel management. In the project, we will fully automate the entire work of the hostels. Whenever a student chooses to stay in the hostel, one should visit our website and then he/she should create an account by filling required details and then once the account was created he/she should choose their choice of rooms in the available rooms such as (2person, 4person, etc.) and then we will be proceeding with the payment and confirmation of the room. When a student residing in a hostel requests to get his room cleaned, a notification will be sent to the help desk staff and then the help desk staff will send the housekeeping staff employee to clean the room at the time mentioned in the request. In case a student faces a problem related to electrical works in the room, a notification will be sent to the help desk again and they will notify the electrician and he will reach the room in the mutually accepted time and any other

complaints regarding carpentry works, plumbing issues and AC repairs will be catered similarly.
The software must be able to perform the following operations:
**I. Student details:** It must be able to take the required student details to create an account.
**II. Room details:** It must be able to show the available rooms for the hostellers to choose the rooms of their choice to stay in the hostel.
**III. Payment and confirmation:** It must be able to confirm if the payment is done for the chosen room and allocate that room.
**IV. Take requests:** It must be able to take requests from the students in the hostel about the type of work required i.e. plumbing, room cleaning, AC repair etc.
**V. Notify staff:** It must be able to notify the help desk staff about the request made by hostellers and once the help desk staff got the request then a person is allotted to do that task.

Initially, we plan to implement the project for only one of the hostels in the city, in which only limited students residing in different rooms of the same hostel will be given access as a part of the Pilot Phase. Once the Pilot Phase is successful then we plan to implement it in the entire hostel. After we get to know all the vulnerabilities of the project, we can implement it in the different hostels and slowly increase the number.

The scope of this project is not just limited to the one hostel or hostels in the same city or one college campus only as the same mechanism can be reused in other campuses as well. There are an uncountable number of institutes in the country where automation of the hostels is required so that students can enjoy their stay at hostels. This system can also be implemented in big cities where service apartments are quite common.

## 1.5 Document conventions

We will display the Hostel Management System (HRM) on a user-friendly and user-centric website.

## 1.6 Contact information/SRS team members

K.Sravan

T.Anudeep

C.Abhinav

**1.7 Definitions and Acronyms**

- SRS – System Requirement Specifications
- DFD – Data Flow Diagram
- ERD – Entity Relationship Diagram
- HMS – Hostel Management System
- User – Student who lives in the hostel
- Database – Records of every hosteller

## 2. Overall Description

The product will run as a website wherein when the concerned person opens the homepage of the website, the person will be asked to sign up if it's the first time he or she is using the portal. If they have been using the site, they will be asked to log in. Once they log in, they can see the different options available in the portal and choose the required option and then proceed with the room details and payment methods there will also be an option like help/form where they complain about the issues they are facing and these are taken care by help desk staff and then they will inform the required ff about the issues and then they will be resolved.

### 2.1 User Neons

The product is going to be used by colleges and universities who are providing hostel facilities to the students and college management is the primary user as the primary user is in direct contact with the system interface and they are having access to the database too and students are the secondary users as they create an account on the portal and choose the rooms of the hostel to stay in them and raise the complaints in their rooms if they have any. The purchaser of this project would be mostly colleges and universities and these can be purchased by people living in big cities where service apartments are quite common and end-users are students in case university had brought it and people in the apartment in case service apartment people had purchased that.

### 2.2 Assumptions and Dependencies

The following list prevents the assumptions, dependencies or guidelines that are imposed upon the implementation of System:
I. The product must have a user-friendly interface that is simple enough for all types of users to understand.
II. Response time should not be longer than 5 seconds

III. General knowledge of basic computer skills or usage of a smartphone knowledge is required to use the product.
IV. They should have the internet to use the product.

## 2.3 Operating environment

Operating environment for the hostel management system includes

Operating system: Windows XP, 7/ Mac OS

Front End: HTML, CSS, JavaScript, PHP

Database(Back End): SQL SERVER

Admin and Student side system

The admin and student side components of the software system must operate within common web browser environments. The browsers that must be included and supported are

- Apple Safari 7+
- Google Chrome 44+
- Microsoft Internet Explorer 10+
- Mozilla Firefox 40+

## 2.5 User environment

Any user is the target audience for the software system that is provided. It gives users access and guidelines to register and view different modules that are present in the software. The users will be given a detailed description of how to register and about every step on the website.

## 2.6 Design/implementation constraints

The Administration Department should implement a Security Policy to ensure the safety of the information contained in our database. A Bug Bounty Team can help catch these errors in the initial stages.

Since there would be many departments in a hostel for the maintenance of hostel information and student databases in any hostel. All these departments provide various records regarding students. Most of the records must contain information about the students. General details like student name, address, number, etc. are stored there. All the modules in hostel administration are interdependent and maintained manually. So they need to be automated and centralized to avoid redundancy.

## 3. System Features and Requirements

### 3.1 Functional Requirements

- Admin can be able to enter the student details and he can perform a few actions such as deletion of student records or adding new student records. Admins will no longer have to maintain a book and register the student details. They can register them with one click through or web portal and can save them or edit them whenever they want.
- Admin can search for the student records and the data stored should be obtained within the stipulated time Delays in time may result in the usage of the software and usage of the software is preferred over the book so that we can save time, while manually it takes a lot of time.
- Provide hostlers with the ability to complain about problems facing the hostel. Residents will no longer have to write about the problems and complaints in the register which are not read by anyone in the management. They can register a complaint with one click through or web portal. Taking into consideration different parameters, a form is made to monitor the quality of the hostel. After filling out the form, the residents can also give suggestions or register a complaint in the other box.
- Admin can maintain the track of the payment records by using the website and they can go through that whenever it is necessary.
- Students are provided with different options of rooms and bed choices in the website so that they can choose according to their choice and proceed with the payment gateway and while registration they are requested to enter some personal details about them for a successful registration.
- **Admin Login:** This facility is for authorizing access to the system. (Admin)Then he should be able to retrieve the information of his students and enter the choice of hostel room details accordingly. It is the job of the administrator to insert, update and monitor the whole process.
- **Student Login:** They will be able to log in and proceed with the registration for a room for having a comfortable stay.

### 3.2 External Interface Requirements

**User Interfaces:**
The goal is to design the software used for the proper management of hostels and automate the current process. The user types are listed as followed
I. Students/Hostellers
II. Hostel staff(people like electrician/plumber)
III. Help desk staff
IV. Administrator

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:
I.The user is a computer-literate and has little or no difficulty in using the software keeping in mind the software is user friendly.
II. In order to use software a user must be aware of the internal working and expected to know how things work.
III. 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.

**Hardware Interfaces:**
I.Computer: A computer will be required to open the website and use the software
II. Smartphone: A smartphone can also be required in case there is no availability of computers for them
III. Internet: A good internet connection is required to access the website and without this if the person is having a computer or smartphone then there would be no use.

**Software Interfaces:**
I. A SQL Database Server will be required to store and retrieve data of the activities done in the hostel management system.
II. A web browser will be required to open the website.

**Communication Interfaces:**
I.The system shall be a standalone product that does not require any communication interfaces.

## 3.3 System Features

**System feature- A**

In the Student View, he can visit all the following pages:

- Rooms and bed details
- Payment related details
- Help desk page to lodge a complaint

### 3.3.1 Description and priority

The hostel management system maintains information about rooms, beds, information about student details, personal information requirements and hostel registration. This project has a high priority because it is very difficult for the hostel administrator to keep track of the hostlers data without this software and it is very beneficial for the students to check in quickly without having any delay and they don’t need anyone to help them out and paperwork is eliminated with this software.

### 3.3.2 Action/result

Gives desired output of the students based on his interaction with the software. It finally allocates the room booked and helps students to have an identity.

**System feature B**

- Student Helpdesk is staffed by hostel wardens who are there to help you and You may contact the help desk at any point of time and raise a query that needs to be addressed.
- Queries are solved by contacting authorized persons and then students are notified about that and changing rooms or editing personal information also can be done through software.

## 3.4 Non-functional Requirements

**Performance Requirements:**
The application shall be based on web technology and has to be run on any platform. the application shall task initial load time depending on performance of the operating system. The performance shall depend upon hardware and software components of the computer or smartphone that we are using.

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

**Security Requirements:**

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

**Quality Requirements:**
The software is prepared to define six quality characteristics such as functionality, reliability, usability, maintainability, portability, and efficiency.

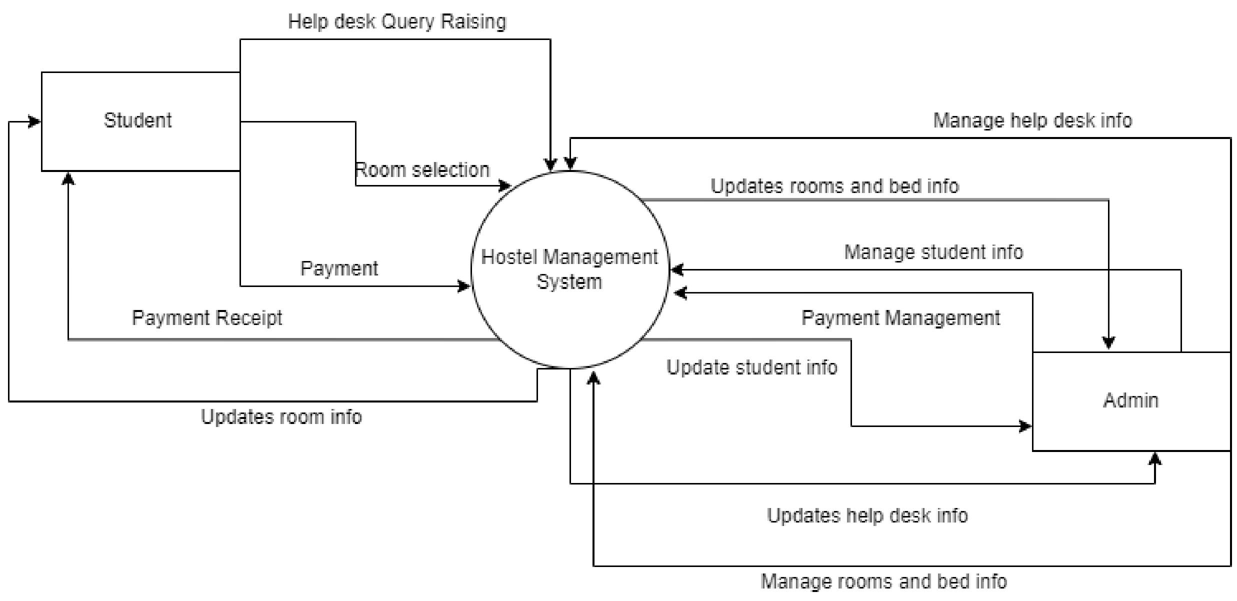
**Conclusion:**
The SRS was made successfully by following the steps described above.

**2. Develop DFD Model (Level 0, Level 1 DFD and data dictionary) of the sample problem (04/Feb/2022)**

**Results:**

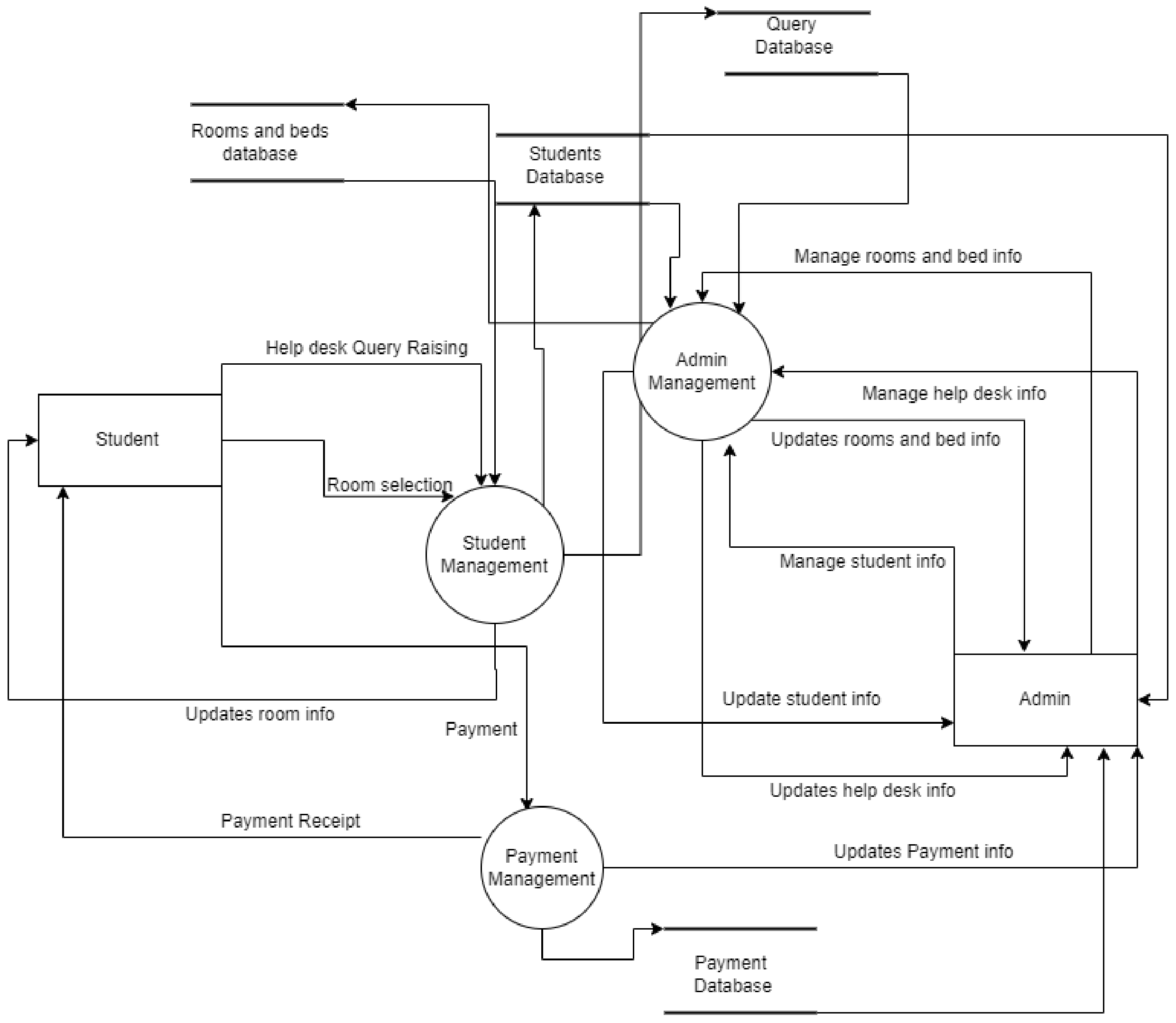
**Level 0 DFD**

This is the Zero Level DFD of HMS, Where we have elaborated the high-level process of System. It's a basic overview of the whole hostel management system or process being analyzed or modeled. It should be easily understood by a wide audience, including Users,Admin,Students. In zero level DFD of Hostel management System, we have described the high-level flow of the Hostel system.
https://app.diagrams.net/#G1edPuFm9UTnMlsDfoEBpsiWe5IMId51Ls



**Level 1 DFD**

First level DFD(1st level) of Hostel Management System shows how the system is divided into subsystems ( process), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the HMS System as a whole. It also identifies internal data stores of payments ,Rooms ,beds, that must be present in order for the Hostel system to do its job, and shows the flow of data between the various parts of the system. DFD level 1 provides a more detailed breakout of pieces of the 1st level DFD.
https://app.diagrams.net/#G1jppwoqmfW8Bt4J_VnXHDcHY5hOwTdEZ1



**DATA DICTIONARY: -**

**a)**

1. Name: - Login
2. Aliases: - None
3. Where used/how used: - User(Staff/Administrator) wants to login
4. Description: - Stores the login ID and Password

| SI.No | Field Name | Data Type | Description |
|---|---|---|---|
| 1 | Login ID | Char(10) | User name |
| 2 | Password | Char(10) | Password of the |

| | | | |
|---|---|---|---|
| | | | user(admin / student) |

**b)**

1. Name: - Student details
2. Aliases: - None
3. Where used/how used: - Details provided by student to book a room
4. Description: - Stores student information

| SI.No | Field Name | Data Type | Description |
|---|---|---|---|
| 1 | First name | Char(50) | First name of the student |
| 2 | Last name | Char(50) | Last name of the student |
| 3 | Address | Char(100) | Address of the student |
| 4 | Mobile number | Int(10) | Mobile number of the student |
| 5 | ID of Student | Int(10) | ID number of the student |

**c)**

1. Name: - Rooms information
2. Aliases: - None
3. Where used/how used: - Students selecting the rooms available
4. Description: - Stores rooms and bed information

| SI.No | Field Name | Data Type | Description |
|---|---|---|---|
| 1 | Rooms | Char(100) | Details related to rooms |
| 2 | ID | Int(10) | ID of the room |
| 3 | Beds | Char(100) | Details related to |

| | | | |
|---|---|---|---|
| | | | beds |

**d)**

1. Name: - Student room booking
2. Aliases: - None
3. Where used/how used: - Student to book a room
4. Description: - Stores student room information

| SI.No | Field Name | Data Type | Description |
|---|---|---|---|
| 1 | Room type | Char(50) | Type of room such as single person. |
| 2 | Room number | Int(10) | Number of the room booked |
| 3 | Bed type | Char(20) | Type of bed such as a single person bed or two person sharing bed. |

**e)**

1. Name: - Help desk Queries
2. Aliases: - None
3. Where used/how used: - Students raising problems that they are facing
4. Description: - Stores Query information

| SI.No | Field Name | Data Type | Description |
|---|---|---|---|
| 1 | Name | Char(50) | Name of the student |
| 2 | ID | Int(10) | ID of the student |
| 3 | Complaint | Char(100) | Problems faced by the student |

**f)**

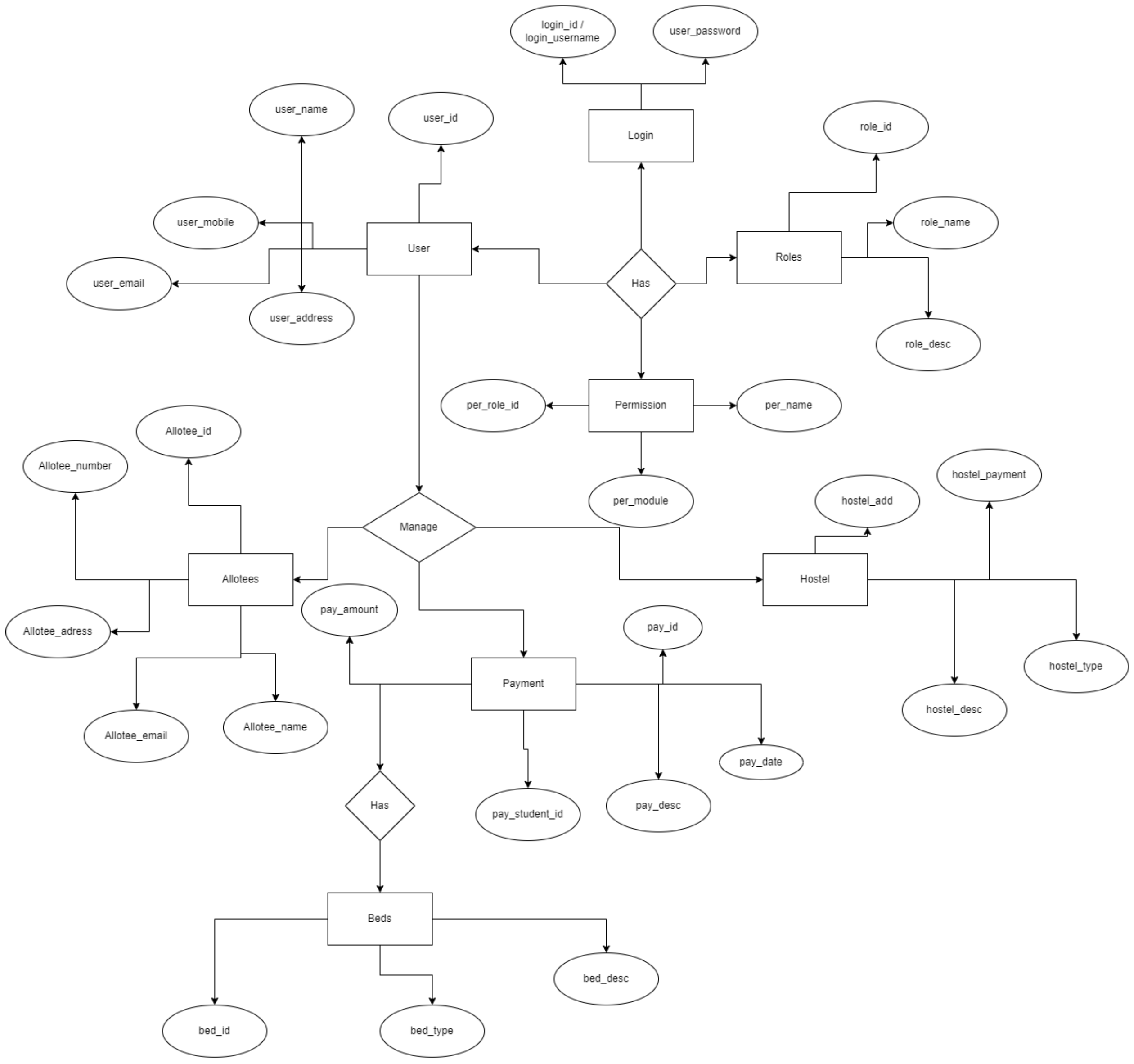
1. Name: - Payment
2. Aliases: - None

3. Where used/how used: - Information about payments done to book a room
4. Description: - Stores payment information

| SI.No | Field Name | Data Type | Description |
|---|---|---|---|
| 1 | Name | Char(50) | Name of the student |
| 2 | ID | Int(10) | ID of the student |
| 3 | Total amount | Int(10) | Amount paid to reserve a room |
| 4 | Mode of payment | Char(20) | Method used for paying the amount |
| 5 | Date of payment | Int(10) & Char(20) | The day of payment made |

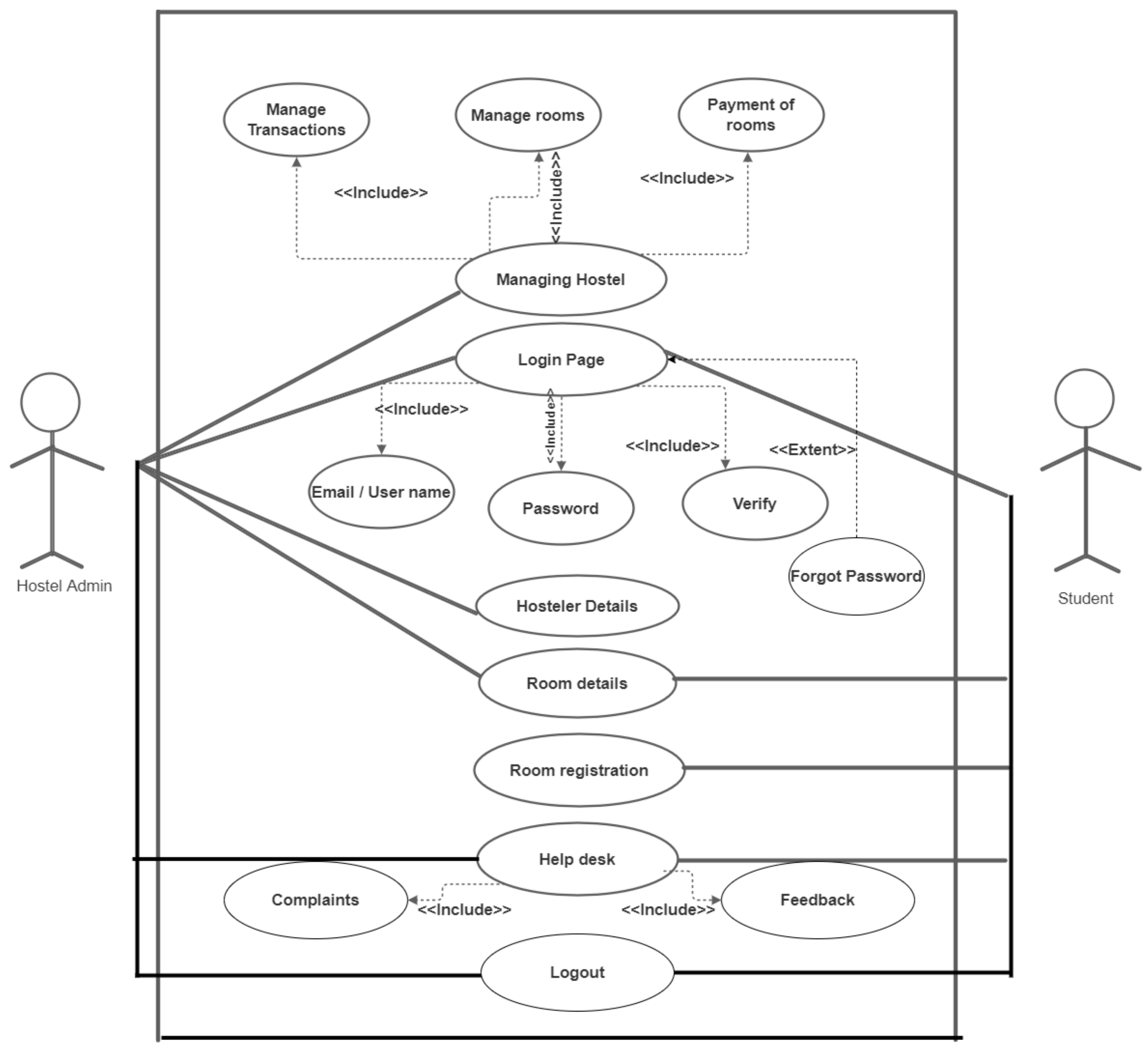
**Conclusion:**
The data flow diagrams of Level-0 and Level-1 and Data dictionaries were made successfully by following the steps described above.

**3. To perform the function oriented diagram: ER diagram (11/Feb/2022)**
**Results:**
https://app.diagrams.net/#G1ekQjnFRIlXnCab-Cd5CrXZPYCfPwhcRy



**Conclusion:**
The Entity Relationship diagram was made successfully by following the steps described above.

**4. To perform the user's view analysis: Use case diagram (18/Feb/2022)**
**Results:**
https://app.diagrams.net/?src=about#G1DgXsgccCWsUGlvrfI6R_RRTIokkeSlua



**Conclusion:**
The use case diagram was made successfully by following the steps described above.

**5. To perform the user's view analysis: Use case diagram Scenarios (25/Feb/2022)**
**Results:**

| | |
|---|---|
| Use Case ID | UC-100 |
| Use Case | Manage Hostel |
| Actors | (H) Admin |
| Description | Admin logins and Opens the hostel room booking, Manage rooms, Manage student transactions and payments. |
| Pre-Conditions | Login page opens. |
| Flow of Events | 1. Admin will log in.<br>2. Then they're directed to the dashboard.<br>3. They can open room bookings based on availability.<br>4. Then they can manage rooms, transactions and payments made by the hostlers.<br>5. Once the admin logs out of the software, then the application is closed. |
| Post-Conditions | Login Page is back on. |
| Alternative Flows | Server may have some issues due to heavy traffic then the server doesn't allow all of the users to log in concurrently. |
| Etc. | |

| | |
|---|---|
| Use Case ID | UC-200 |
| Use Case | Login Page |
| Actors | (I) Hostler/Admin |
| Description | Students and admin fill in their details to log in and view the dashboard, can book rooms and pay the fee accordingly. |

| | |
|---|---|
| Pre-Conditions | Login page opens. |
| Flow of Events | 1. Students and admin fill in their respective details like ID/Username and password.<br>2. They are directed to the dashboard and can view room details and proceed with the payment..<br>3. Then, the person is logged out once the logout button is selected. |
| Post-Conditions | Login page is back on. |
| Alternative Flows | Server may have some issues due to heavy traffic then server doesn’t allow all of the users to log in concurrently. |
| Etc. | |

| | |
|---|---|
| Use Case ID | UC-300 |
| Use Case | Hostler Details |
| Actors | (J) Admin |
| Description | Admin logins and enters student details respectively. |
| Pre-Conditions | Login page opens. |
| Flow of Events | 1. Admin login into the portal.<br>2. Admin selects student details.<br>3. Admin enters the student details based on the data obtained from the students.<br>4. Admin enters details such as mobile number, address, parent details etc.<br>5. Then, the person is logged out once the logout button is selected. |

| | |
|---|---|
| Post-Conditions | Login page is back on. |
| Alternative Flows | Server may have some issues due to heavy traffic then server doesn’t allow all of the users to log in concurrently. |
| Etc. | |

| | |
|---|---|
| Use Case ID | UC-400 |
| Use Case | Room details |
| Actors | (K) Student |
| Description | Students check for the rooms available and details regarding them. |
| Pre-Conditions | Login page opens. |
| Flow of Events | 1. Students will log in.<br>2. They are directed to the dashboard.<br>3. They choose the room details option.<br>4. They can see the available rooms in that hostel.<br>5. Once they go through different types of rooms available then they can select a room based on their choice.<br>6. Then they should book a room and then proceed with the payment gateway.<br>7. The student is then logged out once the logout button is selected. |
| Post-Conditions | Login page is back on. |
| Alternative Flows | Server may have some issues due heavy traffic then server doesn’t allow all of the users to log in concurrently. |
| Etc. | |

| | |
|---|---|
| Use Case ID | UC-500 |
| Use Case | Room Registration |
| Actors | (L) Student |
| Description | Student logins, register for rooms in the hostel, and pay fees. |
| Pre-Conditions | Login page opens. |
| Flow of Events | 1. Students login into the portal using respective login details like ID/Username and password<br>2. User then directs to the hostel room Registration page on selecting.<br>3. Students pay the room registration fee.<br>4. Registration completes after paying the fee and room is allocated to students.<br>5. Then, the person is logged out once the logout button is selected. |
| Post-Conditions | Login page is back on. |
| Alternative Flows | Transaction failure.<br>Server may have some issues due to heavy traffic then server doesn't allow all of the users to log in concurrently. |
| Etc. | |

| | |
|---|---|
| Use Case ID | UC-600 |
| Use Case | Help desk |
| Actors | (M) Hostler/Admin |
| Description | Students and admin fill in their details to log in and view the help desk where student requests |

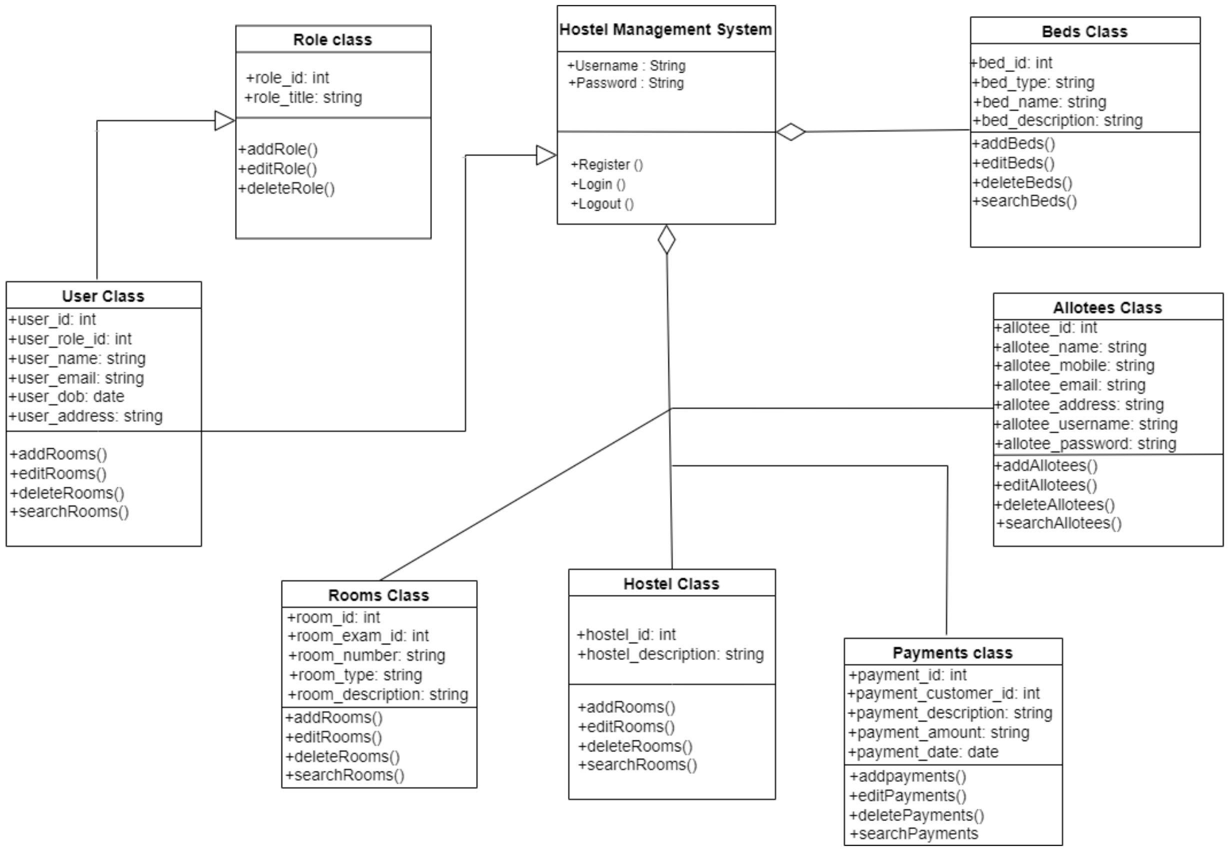
| | |
|---|---|
| | for help and admin providing the required help. |
| Pre-Conditions | Login page opens. |
| Flow of Events | 1. Students/Admin login into the portal using respective login details like ID and password.<br>2. User then directs to the help desk page on selecting.<br>3. Then students can request for any help regarding hardware problems faced in the hostel room or any help regarding hostile rooms.<br>4. Once students raise a request for any help then admin follow up the request and make sure to solve it by providing the required help.<br>5. Then, the person is logged out once the logout button is selected. |
| Post-Conditions | Login page is back on. |
| Alternative Flows | Network Failure.<br>Server may have some issues due to heavy traffic then server doesn’t allow all of the users to log in concurrently. |
| Etc. | |

**Conclusion:**
The use case scenario was made successfully by following the steps described above.

**6. To draw the structural view diagram: Class diagram (04/Mar/2022)**

**Results:**

https://app.diagrams.net/#G1AVQhevFljhXorQOxEyyk9pT86Ojfah3b



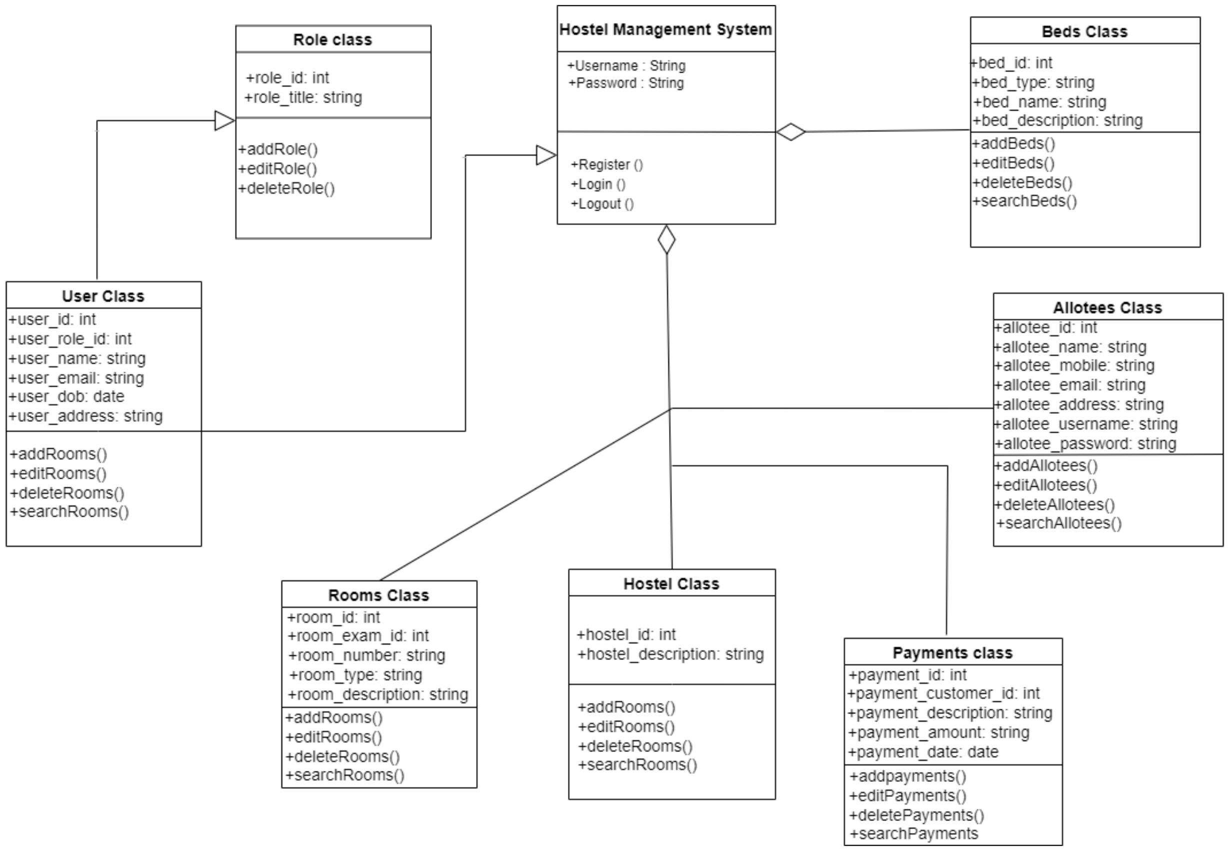
**Conclusion:**

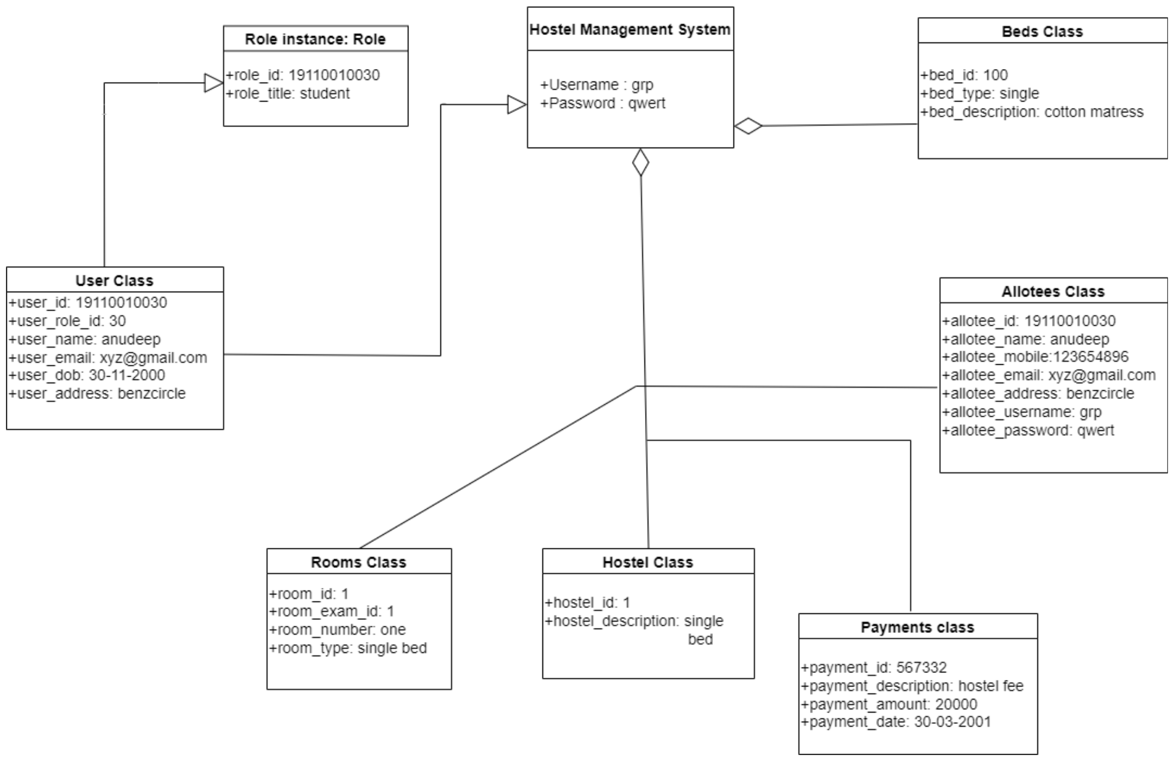
The class diagram was made successfully by following the above steps.

## 7. To draw the structural view diagram: Object diagram (11/Mar/2022)

**Results:**

https://app.diagrams.net/#G1EdTFTIzHArlMFZSt8yJR4Yc95Iv-9LCq





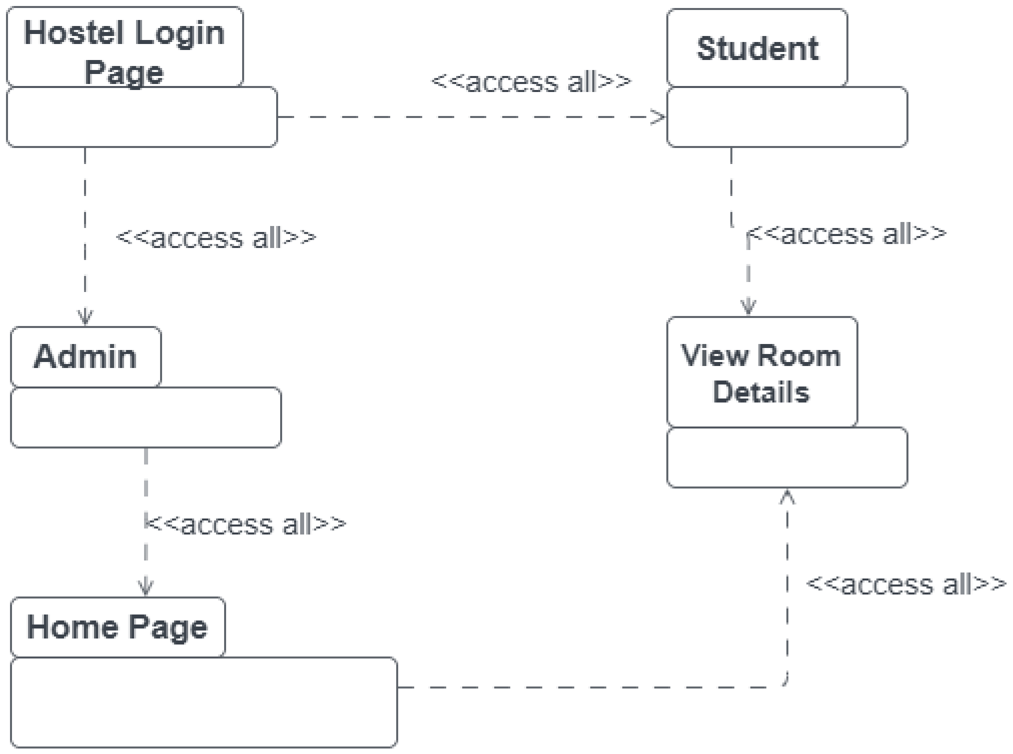
**Conclusion:**

The object diagram was made successfully by following the above steps.

**8. To draw the structural view diagram: Package diagram (25/Mar/2022)**

**Results:**

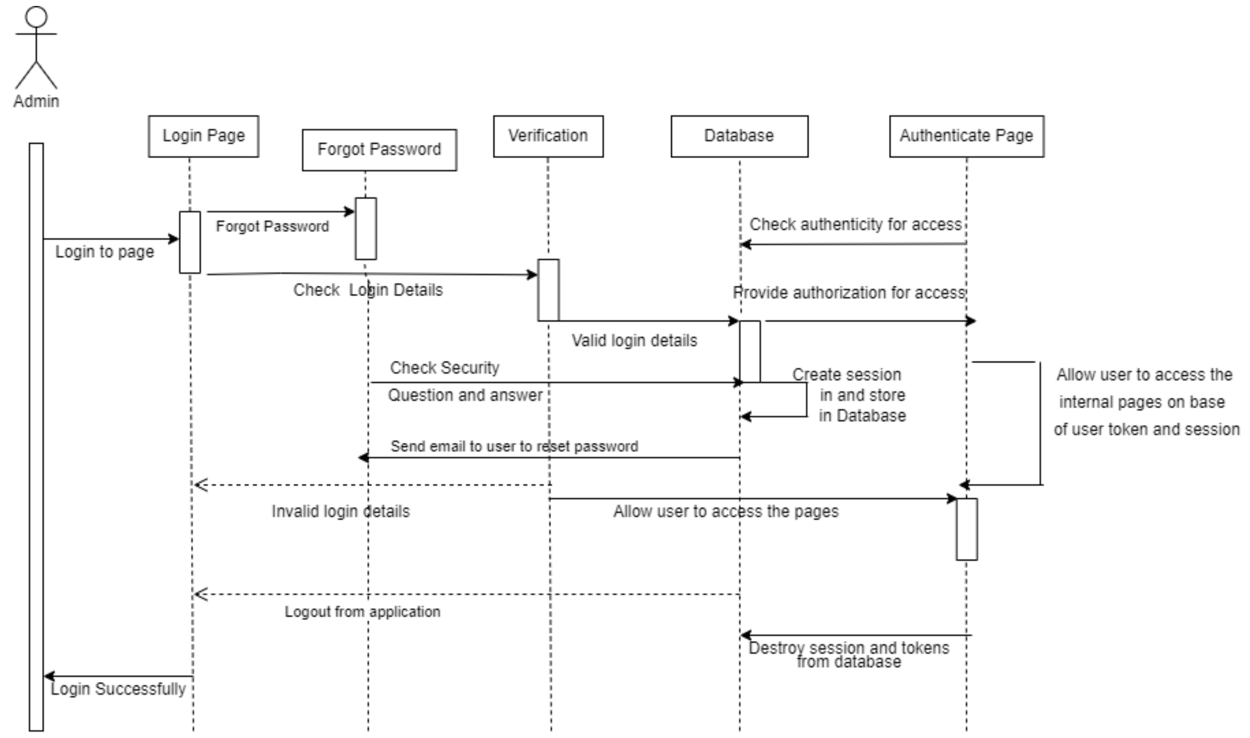
https://app.diagrams.net/?src=about#G1oDlc7yqFPA3m68P2Nh16OnMkTqXdUdOp



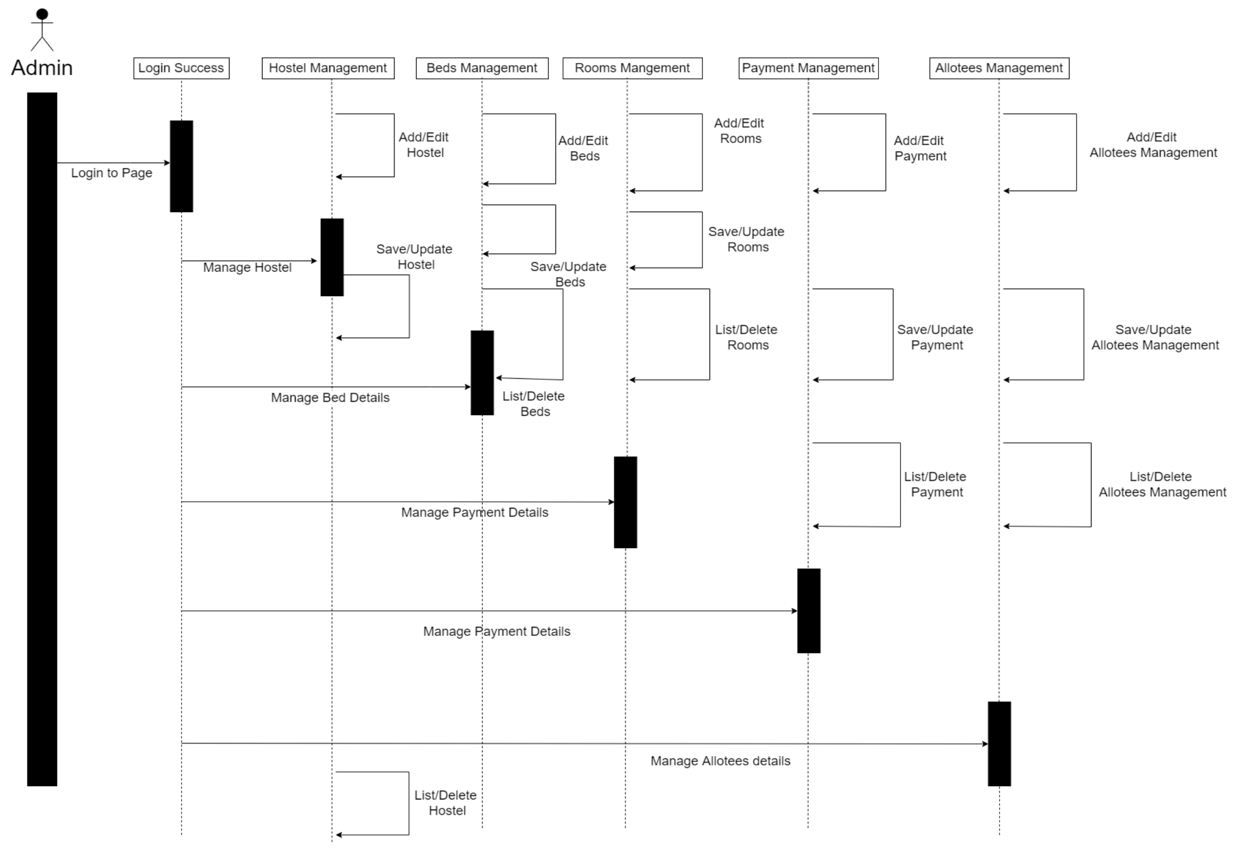
**Conclusion:**

The package diagram was made successfully by following the above steps.

**9. To draw the behavioral view diagram: Sequence diagram (25/Mar/2022)**
**Results:**
https://app.diagrams.net/#G1NhqyqjhMP8xupWYiopMnjTMTZBjiQi4t



https://app.diagrams.net/#G1_AzuXdG7r3ny_JOFzuJCQoShLzJ4bnGY



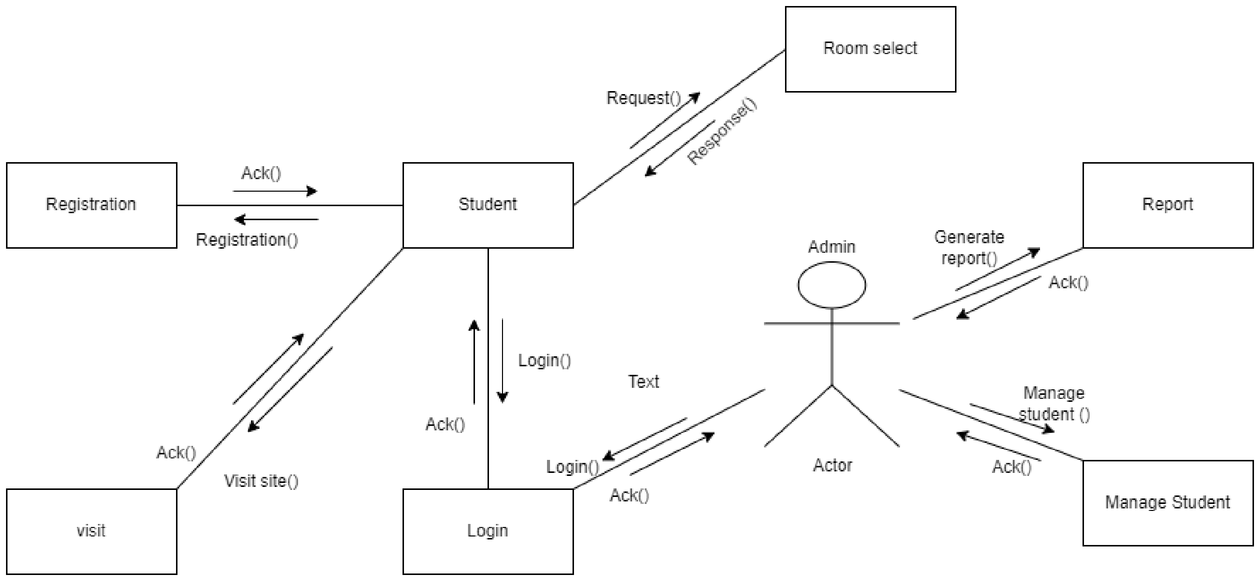
## Conclusion:

The sequence diagram was made successfully by following the above steps.

## 10. To draw the behavioral view diagram: Collaboration diagram (08/Apr/2022)

**Results:**

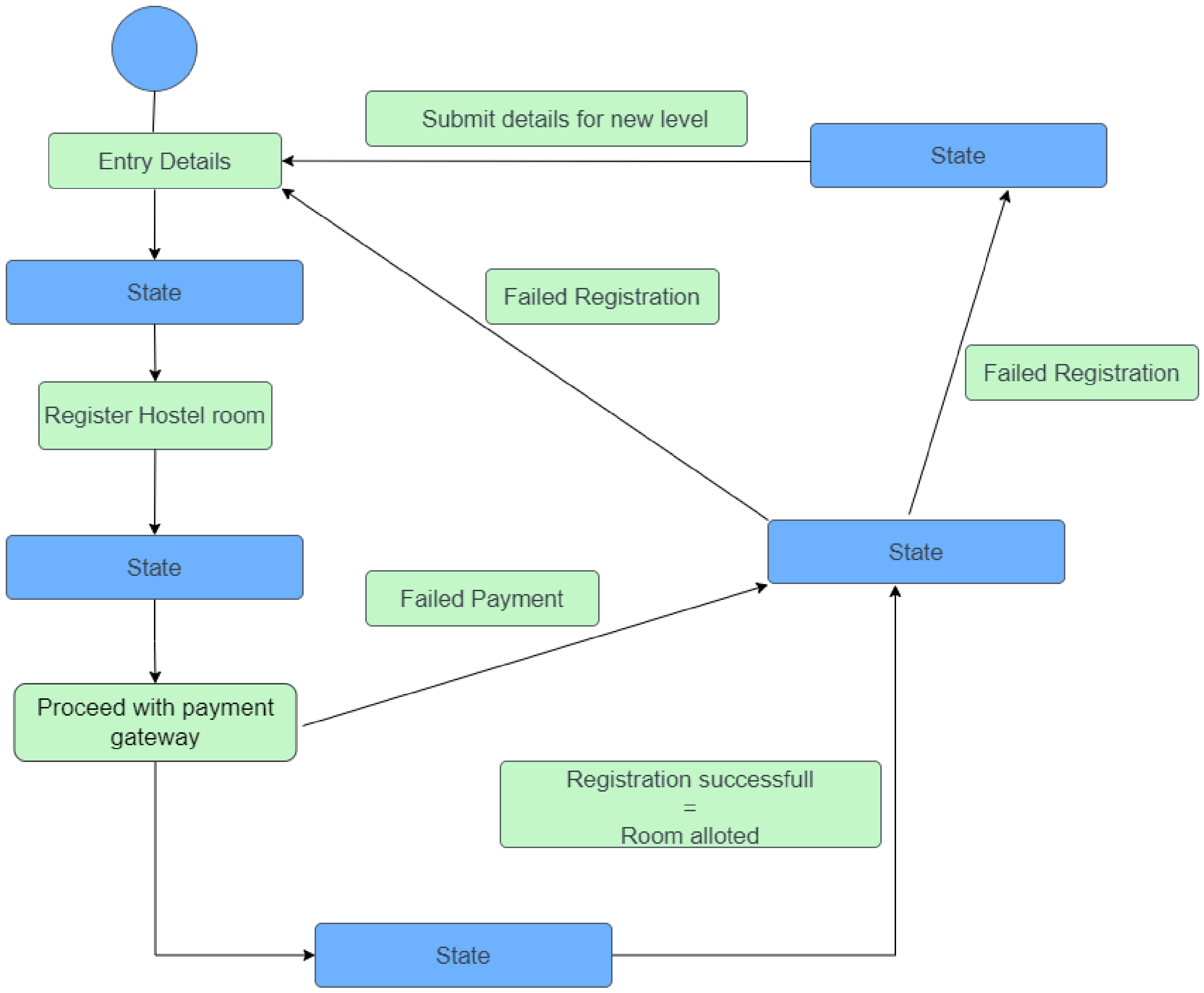
https://app.diagrams.net/?src=about#G1-0sNnlp7wQ7-pBySs3dKSM-PR55IKB2x



**Conclusion:**

The collaboration diagram was made successfully by following the above steps.

**11. To draw the behavioral view diagram: State-chart diagram (08/Apr/2022)**
**Results:**
https://app.diagrams.net/?src=about#G1muvCUqWEEslLCF9HQ5ir0CFFumJhBIQ1

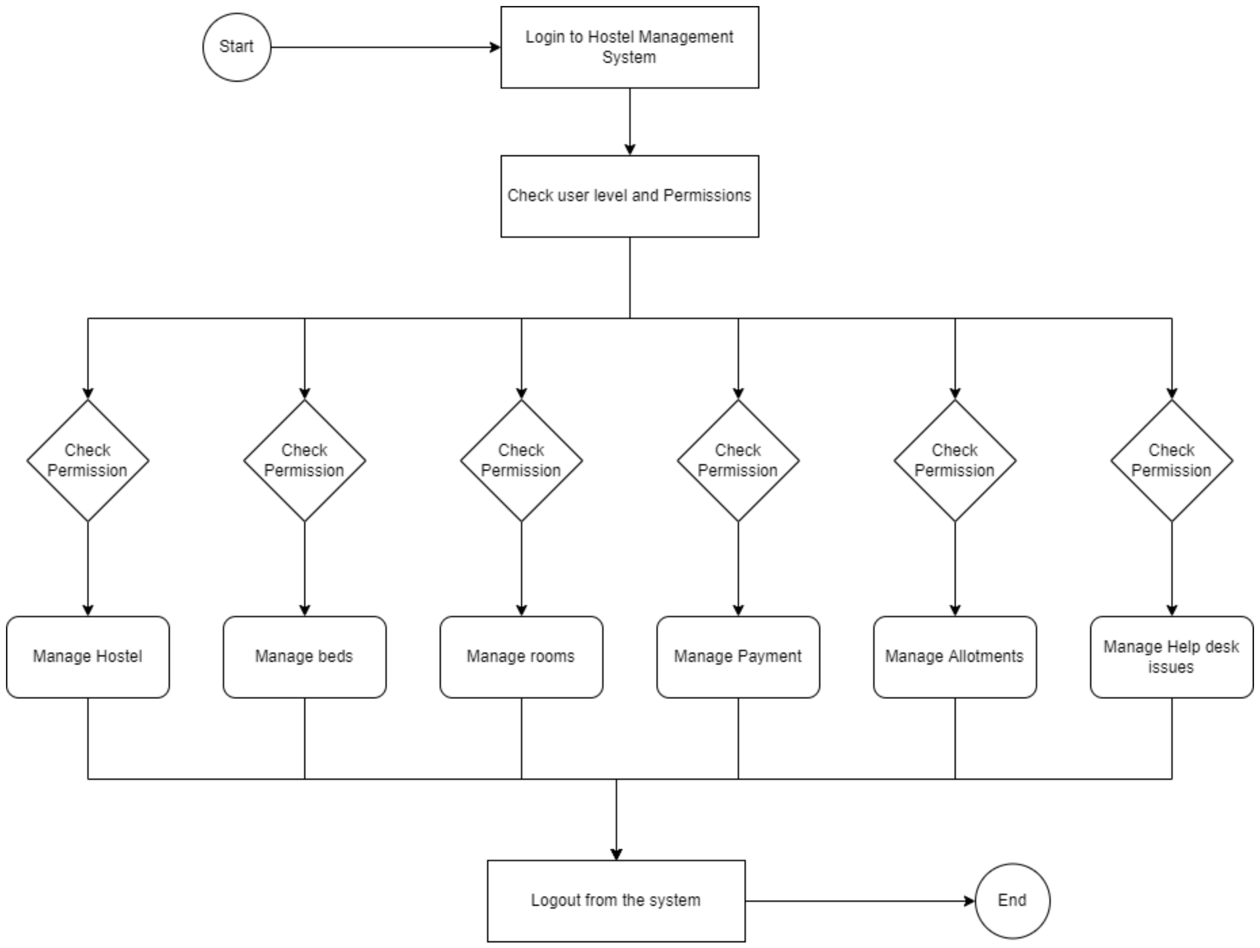


**Conclusion:**
The state chart diagram was made successfully by following the above steps.

**12. To draw the behavioral view diagram: Activity diagram (22/Apr/2022)**

**Results:**

https://app.diagrams.net/?libs=general;uml&src=about#G1CHIJkQXSDrxnD4sIHkgvaYyIHlutxb-A



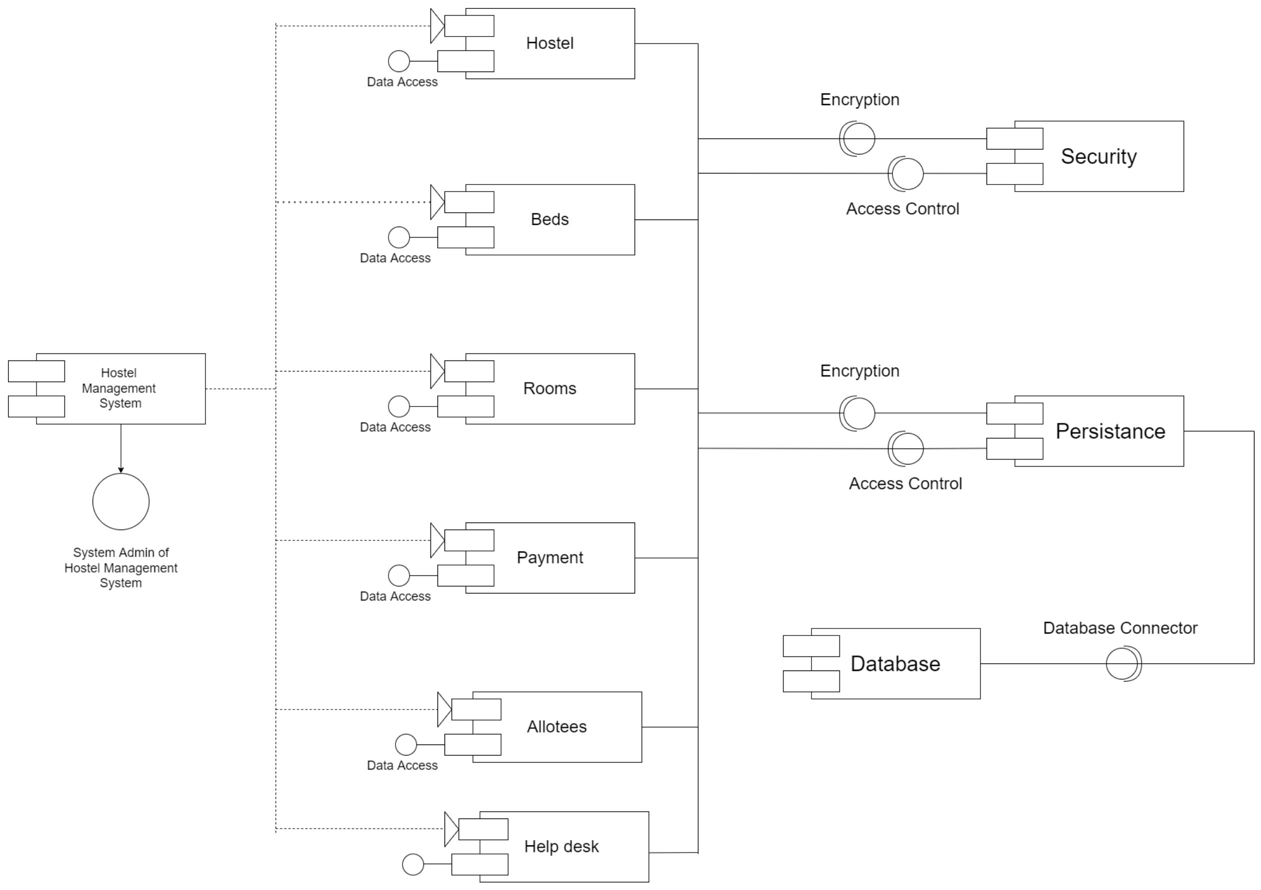
**Conclusion:**

The activity diagram was made successfully by following the above steps.

## 13. To draw the implementation view diagram: Component diagram (22/Apr/2022)

**Results:**

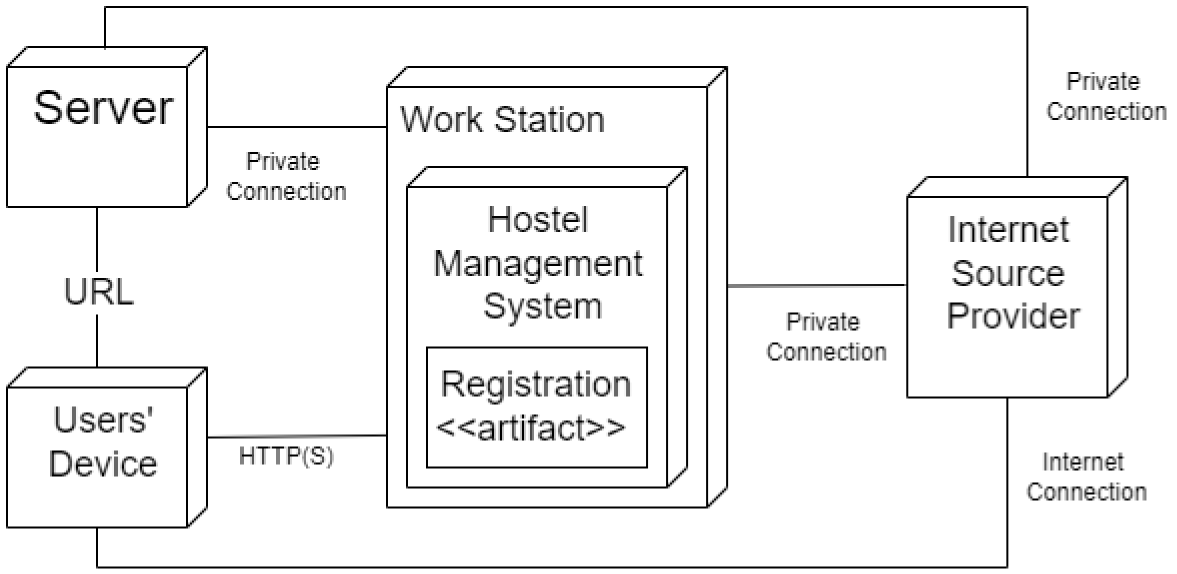
https://app.diagrams.net/#G1aQF6BJanXgfxv55D3zL74Dl8u6GPsTfz



**Conclusion:**

The component diagram was made successfully by following the above steps.

**14. To draw the environmental view diagram: Deployment diagram (29/Apr/2022)**
**Results:**
https://app.diagrams.net/#G1BNcZm_hQrBC_6xjAXnVJEX7VJ8HSe9re



**Conclusion:**
The component diagram was made successfully by following the above steps.

**15.To perform various testing using the testing tool unit testing, integration testing (29/Apr/2022)**

**Results:**

**Login testing code:**

```
session_start();
include('includes/config.php');
if(isset($_POST['login']))
{
$email=$_POST['email'];
$password=$_POST['password'];
$stmt=$mysqli->prepare("SELECT email,password,id FROM userregistration WHERE
email=? and password=? ");
            $stmt->bind_param('ss',$email,$password);
            $stmt->execute();
            $stmt -> bind_result($email,$password,$id);
            $rs=$stmt->fetch();
            $stmt->close();
            $_SESSION['id']=$id;
            $_SESSION['login']=$email;
            $uip=$_SERVER['REMOTE_ADDR'];
            $ldate=date('d/m/Y h:i:s', time());
            if($rs)
            {
          $uid=$_SESSION['id'];
          $uemail=$_SESSION['login'];
$ip=$_SERVER['REMOTE_ADDR'];
$geopluginURL='http://www.geoplugin.net/php.gp?ip='.$ip;
$addrDetailsArr = unserialize(file_get_contents($geopluginURL));
$city = $addrDetailsArr['geoplugin_city'];
$country = $addrDetailsArr['geoplugin_countryName'];
$log="insert into userLog(userId,userEmail,userIp,city,country)
values('$uid','$uemail','$ip','$city','$country')";
$mysqli->query($log);
if($log)
{
header("location:dashboard.php");
                }
}
                else
                {
                    echo "<script>alert('Invalid Username/Email or password');</script>";
                }
            }
                ?>
```

**<u>Room booking based on availability testing code:</u>**

```
function checkAvailability() {
$("#loaderIcon").show();
jQuery.ajax({
url: "check_availability.php",
data:'roomno='+$("#room").val(),
type: "POST",
success:function(data){
$("#room-availability-status").html(data);
$("#loaderIcon").hide();
},
error:function (){}
});
}
</script>
```

```
<script type="text/javascript">

$(document).ready(function() {
   $('#duration').keyup(function(){
       var fetch_dbid = $(this).val();
       $.ajax({
       type:'POST',
       url :"ins-amt.php?action=userid",
       data :{userinfo:fetch_dbid},
       success:function(data){
       $('.result').val(data);
       }
       });
```

**Logout testing code:**

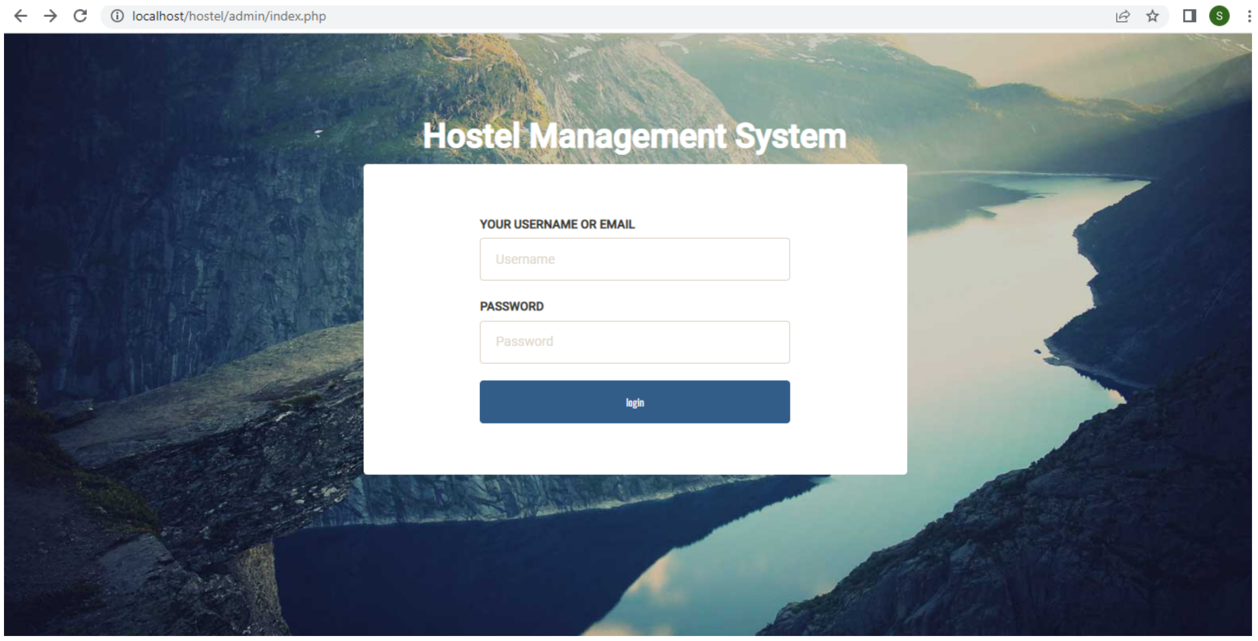
```
<?php
session_start();
unset($_SESSION['id']);
session_destroy();
header('Location:../index.php');
?>
```

**Room creating testing code:**

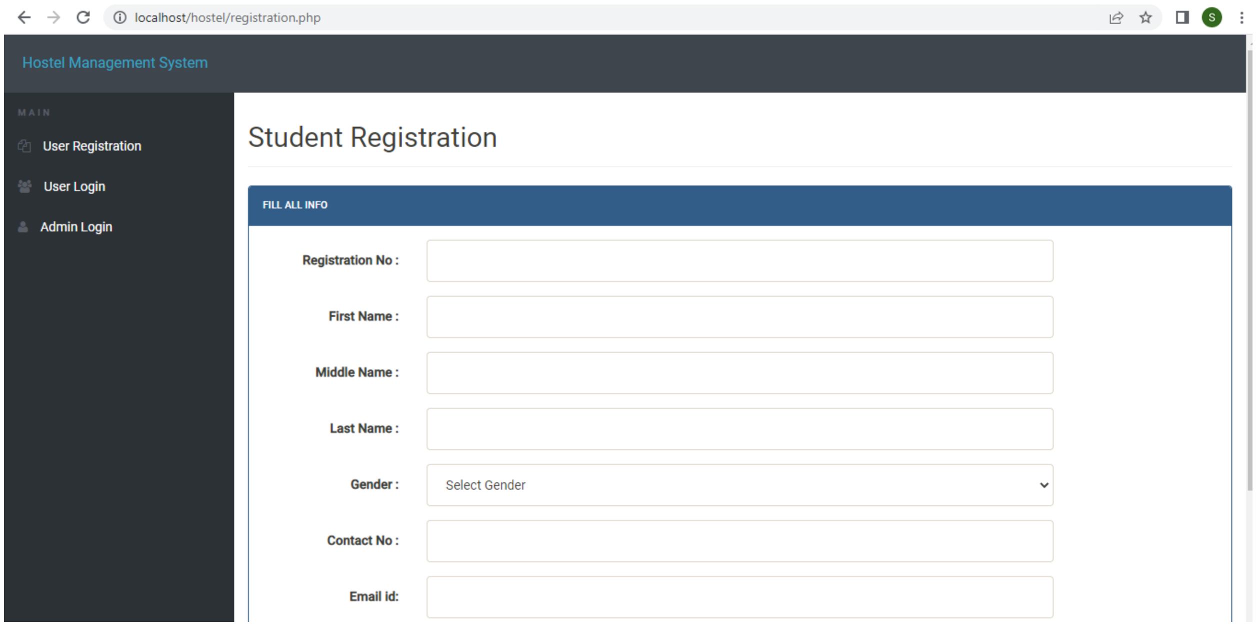
```
<?php
session_start();
include('includes/config.php');
include('includes/checklogin.php');
check_login();
//code for add courses
```

```
if($_POST['submit'])
{
$seater=$_POST['seater'];
$roomno=$_POST['rmno'];
$fees=$_POST['fee'];
$sql="SELECT room_no FROM rooms where room_no=?";
$stmt1 = $mysqli->prepare($sql);
$stmt1->bind_param('i',$roomno);
$stmt1->execute();
$stmt1->store_result();
$row_cnt=$stmt1->num_rows;;
if($row_cnt>0)
{
echo"<script>alert('Room alreadt exist');</script>";
}
else
{
$query="insert into  rooms (seater,room_no,fees) values(?,?,?)";
$stmt = $mysqli->prepare($query);
$rc=$stmt->bind_param('iii',$seater,$roomno,$fees);
$stmt->execute();
echo"<script>alert('Room has been added successfully');</script>";
}
}
?>
```

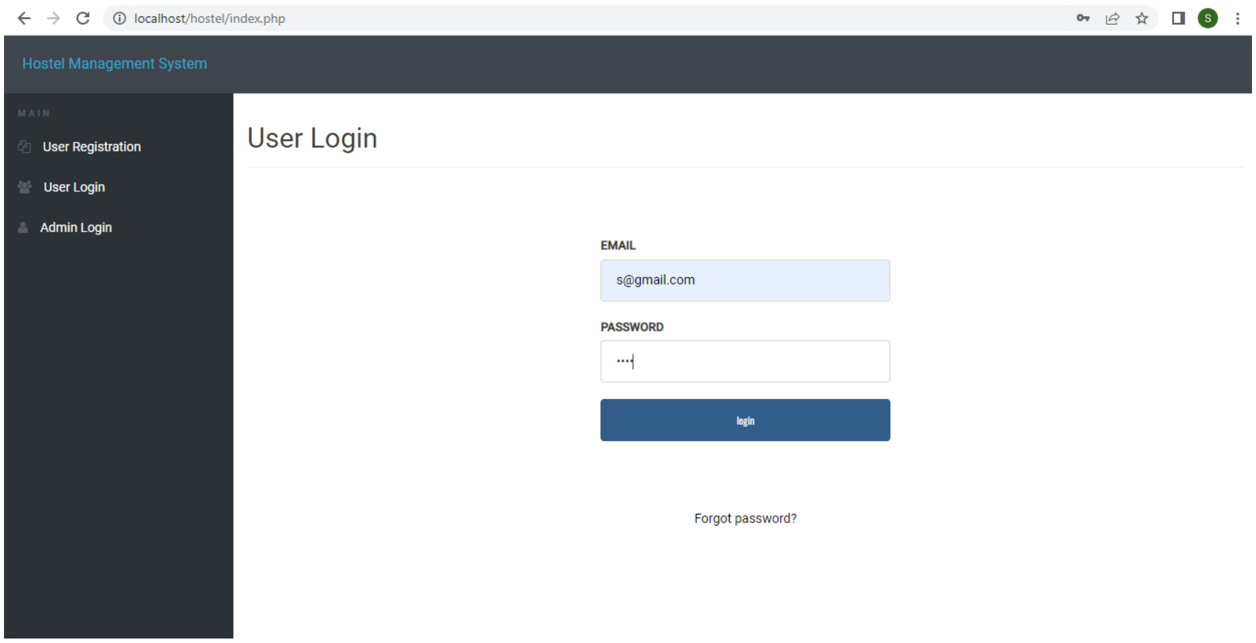
## Admin login page:



## Student registration:



## User login:



localhost/hostel/index.php

Hostel Management System

MAIN
- User Registration
- User Login
- Admin Login

User Login

EMAIL

s@gmail.com

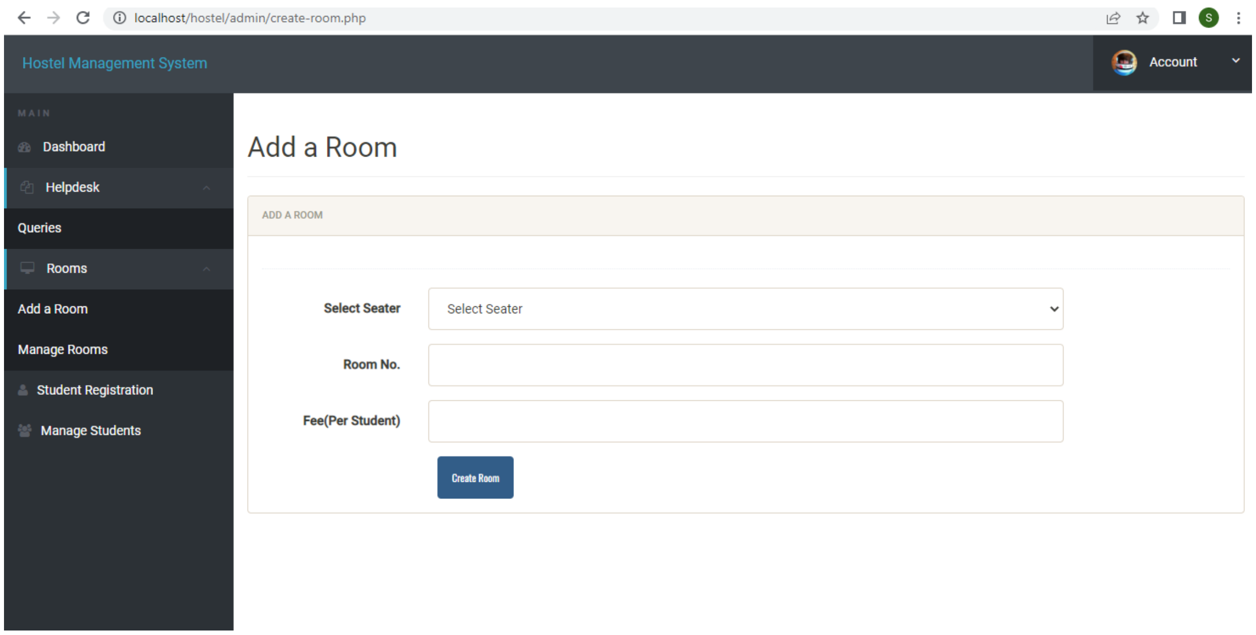
PASSWORD

····

login

Forgot password?

## Admin adding a room:



localhost/hostel/admin/create-room.php

Hostel Management System

Account

MAIN
- Dashboard
- Helpdesk
- Queries
- Rooms
- Add a Room
- Manage Rooms
- Student Registration
- Manage Students

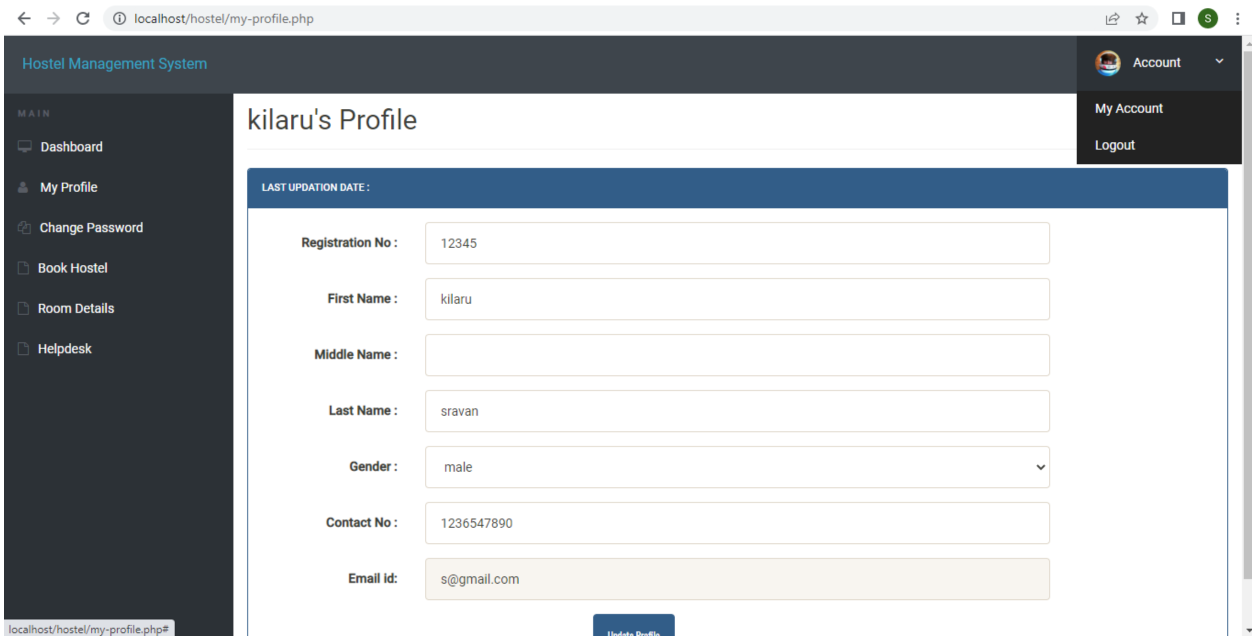
Add a Room

ADD A ROOM

| | |
|---|---|
| Select Seater | Select Seater |
| Room No. | |
| Fee(Per Student) | |

Create Room

## Hostler's profile:



localhost/hostel/my-profile.php

Hostel Management System

Account

My Account

Logout

MAIN

Dashboard

My Profile

Change Password

Book Hostel

Room Details

Helpdesk

kilaru's Profile

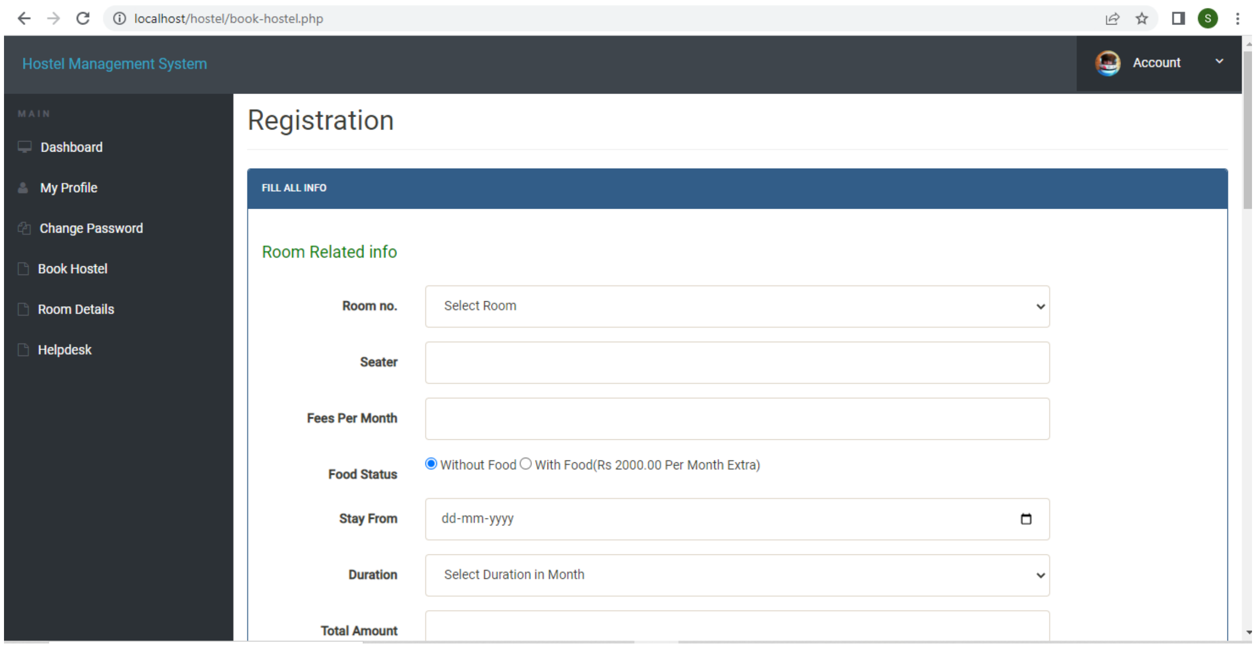
LAST UPDATION DATE :

| | |
|---|---|
| Registration No : | 12345 |
| First Name : | kilaru |
| Middle Name : | |
| Last Name : | sravan |
| Gender : | male |
| Contact No : | 1236547890 |
| Email id: | s@gmail.com |

Update Profile

localhost/hostel/my-profile.php#

## Registration of a room:



localhost/hostel/book-hostel.php

Hostel Management System

Account

MAIN

Dashboard

My Profile

Change Password

Book Hostel

Room Details

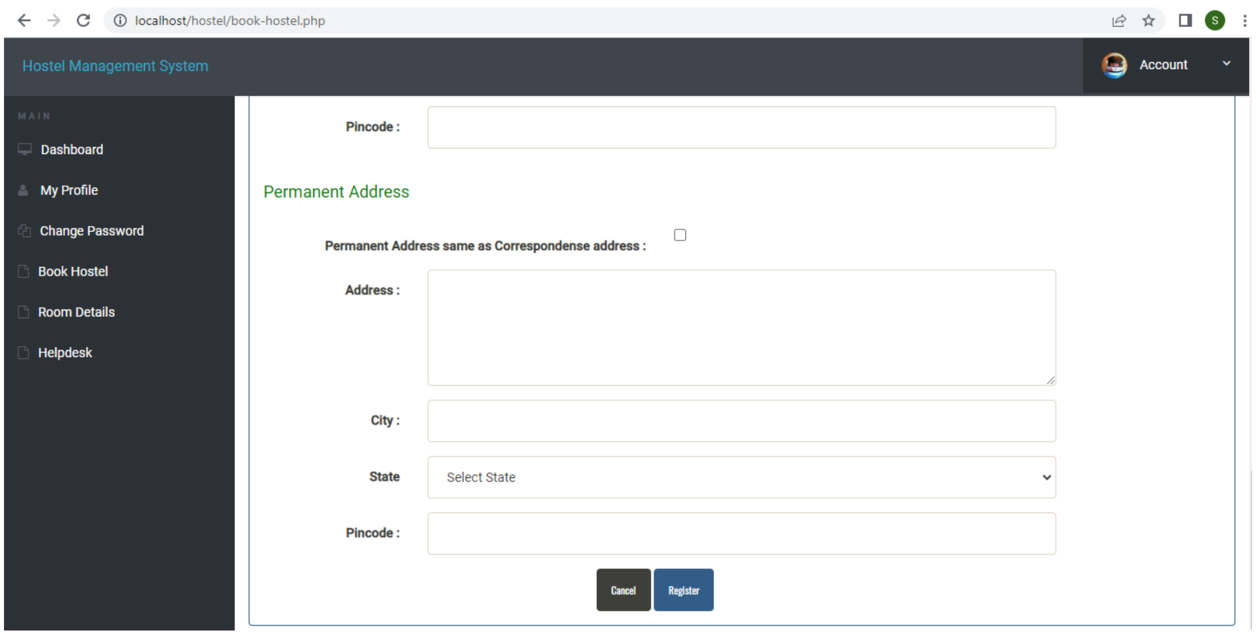
Helpdesk

Registration

FILL ALL INFO

Room Related info

| | |
|---|---|
| Room no. | Select Room |
| Seater | |
| Fees Per Month | |
| Food Status | [✓ Without Food] With Food(Rs 2000.00 Per Month Extra) |
| Stay From | dd-mm-yyyy |
| Duration | Select Duration in Month |
| Total Amount | |



Pincode :

Permanent Address

Permanent Address same as Correspondense address :

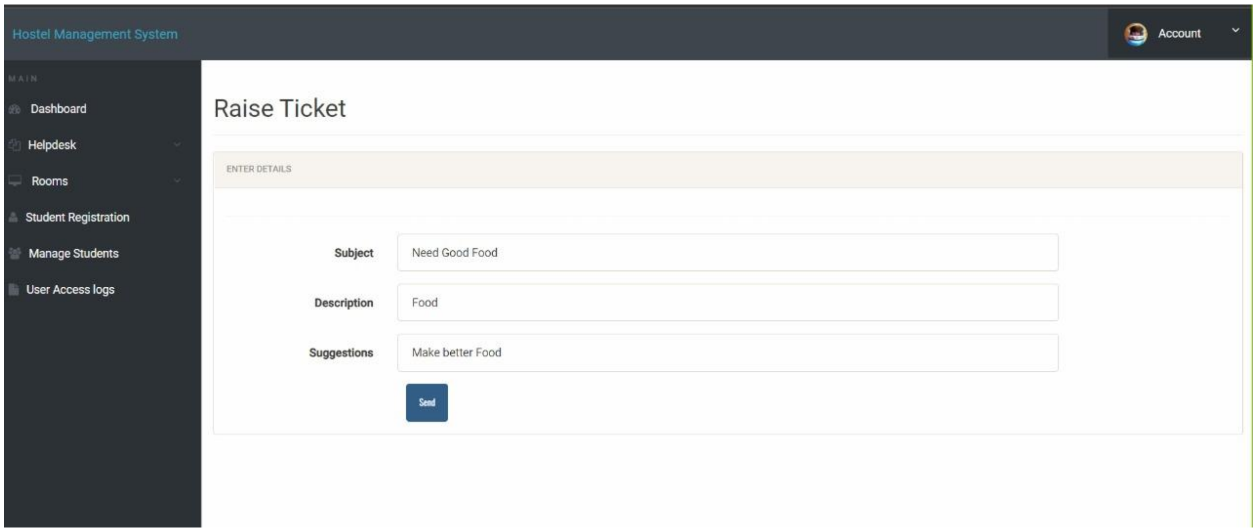
Address :

City :

State: Select State

Pincode :

## Help desk (ticket raising):



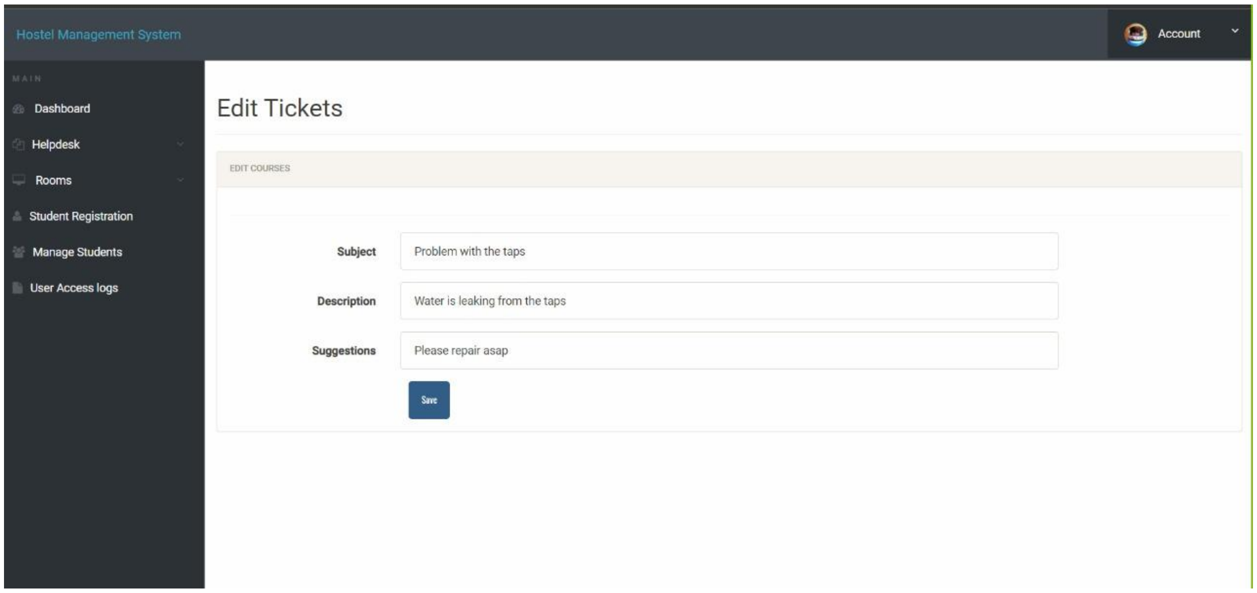
Raise Ticket

ENTER DETAILS

Subject: Need Good Food

Description: Food

Suggestions: Make better Food



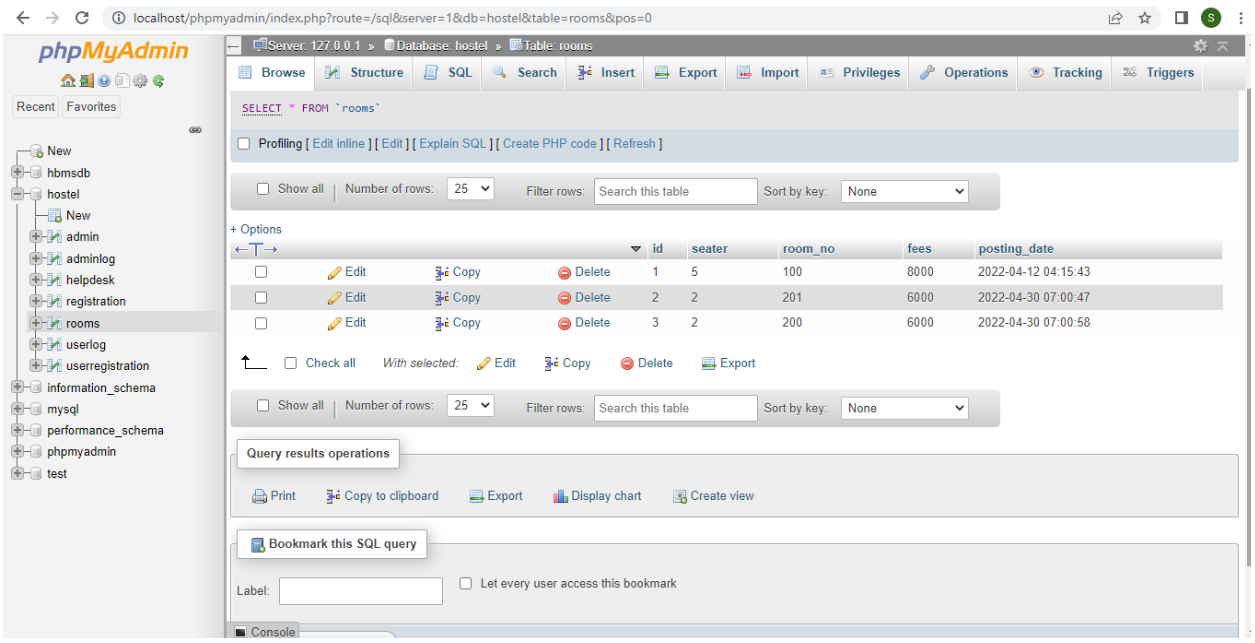
Hostel Management System

Edit Tickets

EDIT COURSES

| | |
|---|---|
| Subject | Problem with the taps |
| Description | Water is leaking from the taps |
| Suggestions | Please repair asap |

## Available rooms database:



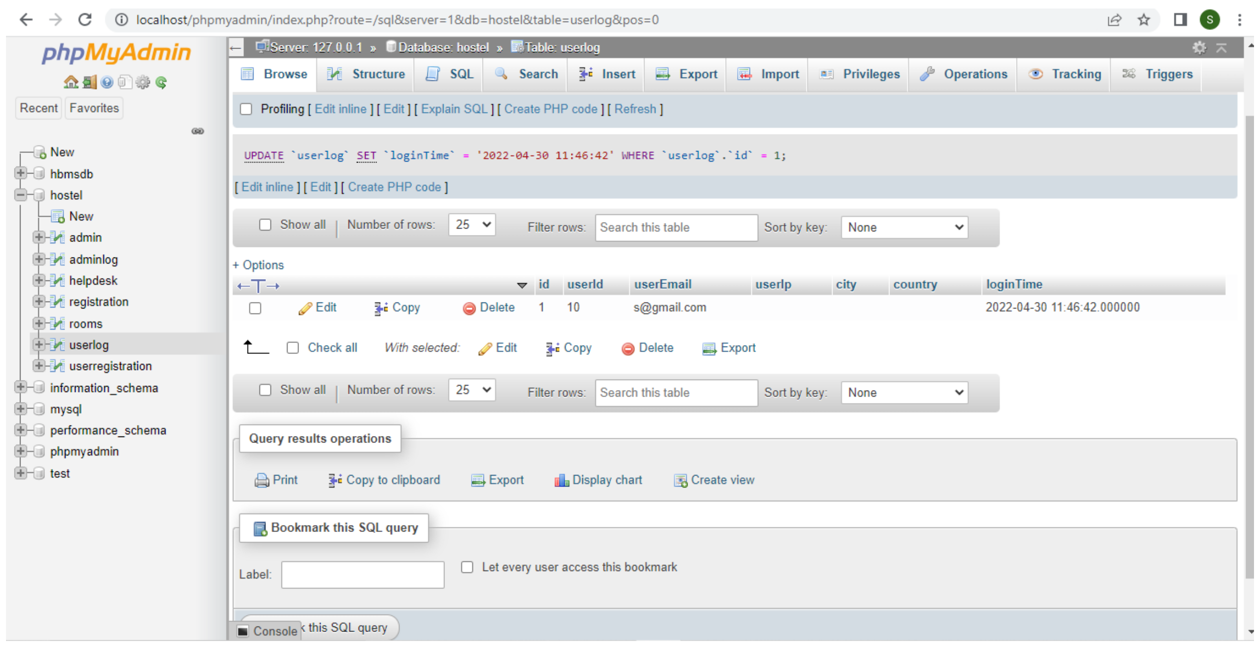
localhost/phpmyadmin/index.php?route=/sql&server=1&db=hostel&table=rooms&pos=0

Server: 127.0.0.1 » Database: hostel » Table: rooms

```
SELECT * FROM `rooms`
```

| id | seater | room_no | fees | posting_date |
|---|---|---|---|---|
| 1 | 5 | 100 | 8000 | 2022-04-12 04:15:43 |
| 2 | 2 | 201 | 6000 | 2022-04-30 07:00:47 |
| 3 | 2 | 200 | 6000 | 2022-04-30 07:00:58 |

## User login database:



```
UPDATE `userlog` SET `loginTime` = '2022-04-30 11:46:42' WHERE `userlog`.`id` = 1;
```

[ Edit inline ] [ Edit ] [ Create PHP code ]
Show all
Filter rows: Search this table
+ Options
Check all
With selected: Edit Copy Delete Export
Query results operations
Print
Copy to clipboard
Export
Display chart
Create view
Bookmark this SQL query
Let every user access this bookmark
Console

Number of rows: 25 | Sort by key: None

| | id | userId | userEmail | userIp | city | country | loginTime |
|---|---|---|---|---|---|---|---|
| Edit Copy Delete | 1 | 10 | s@gmail.com | | | | 2022-04-30 11:46:42.000000 |

Label:

## Conclusion:

Unit testing and Integration testing was made successfully by following the above steps.