UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY

PANJAB UNIVERSITY , CHANDIGARH

"HOSTEL MANAGEMENT SYSTEM"

(DBMS)

Team Members :

Name	Roll Number
Aaryan Singh	UE213003
Akshat Verma	UE213005
Ananya Trinach	UE213007
Aneesh Sayal	UE213009
Bibhuti Singha	UE213024

Why Hostel management system?

- As we all know that the hostels in Panjab University are managed manually by the administration.There is a lot of paper work involved in the whole process.
- It is very difficult to manage the handwritten records of all the students at one place. The current system is inefficient and usually involves a lot of repetition that is time consuming too.
- Also, the new students that are allotted hostels often find the allotement process difficult as they have no idea about it.
- There are no proper resources that provide the complete information regarding the mess details and the facilities that are provided to them in the hostel .
- Moreover , 4 out of 5 team members are hostellers. So we have a better understanding what are the challenges that we often face in the hostel.
- So we decided to create a system that deals with the all these problems faced by the students regularly. With the help of this system ,the whole hostel will be managed in a very effective way and will be convenient for the students as well as for the administration.

Objectives of the project

- To automate each and every process that involves the manual work.
- To register newly admitted students in a very effective way.
- To update , delete or search records of students in a timeless manner.
- To obtain any information regarding mess , hostel amenities , rooms or fee structure .
- To make the system more interactive and user friendly.
- To avoid any kind of inaccuracy of data due to human error and prevent the loss of data.

Software requirement specification

- **MySQL** 

It is an open source Relational Database Management System (RDBMS) and is one of the most widely used. It also allows to scale the project without much overhead. It also has many features such as high availability, query caching, cross platform support and security make it a good candidate for deployment.

- **Python Tkinter** 

Tkinter is the inbuilt Python module to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library.

Operating System : WINDOWS 10/11 

Back End : MySQL 

Front End : PYTHON (TKINTER) 

Database Connectivity : MYSQL CONNECTOR

Code editor: VS code 

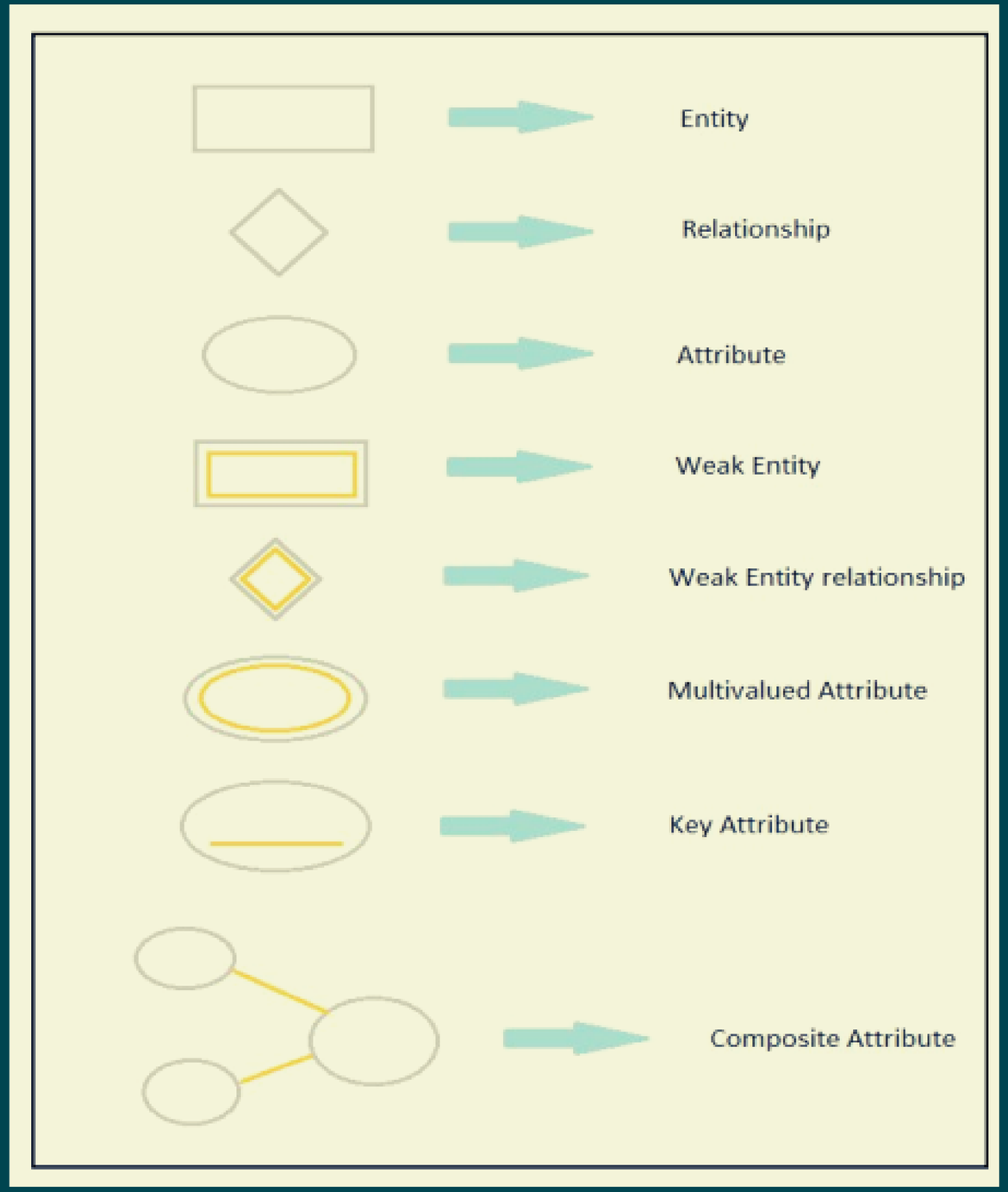
Database Design

ER diagram :

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

ER diagram should have mainly 3 components namely, entity, attribute and relationship.

Our ER diagram consists of seven entities Hemployee1, Hroom1, Fees_bill, Hstudent1 ,Hostel , visitors1, Mess_canteen1 related by the relationtionships pays, stays at, contains, has, resides, visits , getinfo. Each entity consists of many attributes.The following notations can be used for drawing an ER diagram:


Entity
Relationship
Attribute
Weak Entity
Weak Entity relationship
Multivalued Attribute
Key Attribute
Composite Attribute


item_no
availability
Mess_ canteen1
getinfo
visitor_id
visitors1
Visitor_ phone
Visitor_ name
menu
timings
Visit_to
price
Visit_ date
Visitor- addr
Visits
visit_ purpose
stud_id
stud_ year
stud_ address
Hname
stud_ name
Hstudent1
resides
Hostel
hostel_ no
stud_ dept
Student _id
Htype
stud_ phone
contains
has
pays
stays at
emp_id
room_no
stud_id
room_no
cooler_ available
Fees_bill
Hroom1
HEmployee1
emp_ addr
mess_bill
emp_ name
fine
allocated _to
emp_ designation
emp_ department
canteen _bill
paid_sem_ fees
roomate
no_of _chairs
no_of _tables

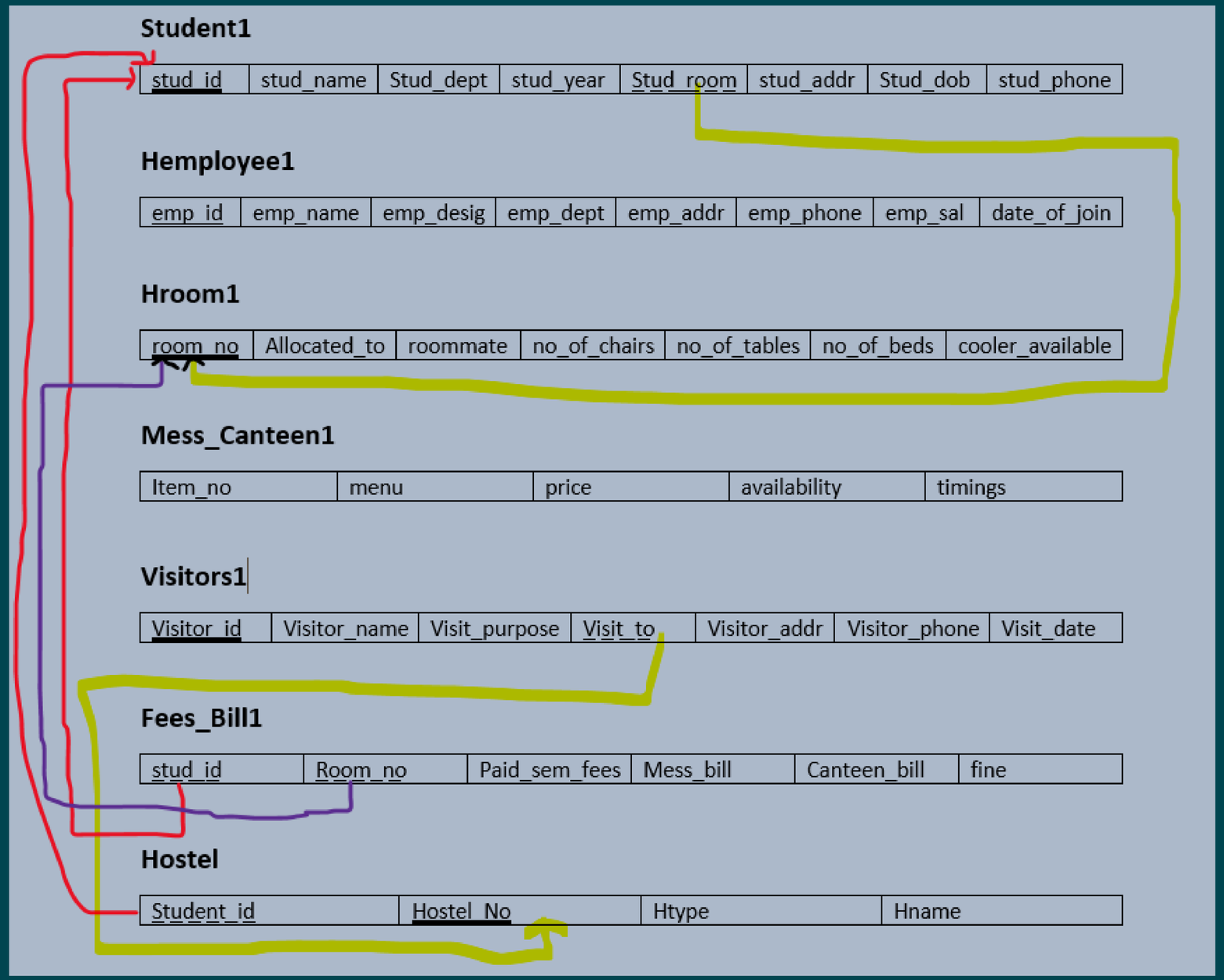
Relational Schema:

The relational schema diagram gives the relation of one entity with another as well as the information about the key constraints. The next figure is the relational schema diagram in which the attributes that are underlined are the primary key and the arrow line is used to represent the mapping. There are totally 7 entities and 7 relationships.

stud_id of Student1 relation , room_no of Hroom1 relation visitor_id of Visitors1 relation and Hostel_no of hostel relation are the primary keys in the relational schema.

stud_room of Student1 relation , visit_to of visitors1 relation , stud_id and room_no of Fees_Bill1 relation are the respective foreign keys.

Hostel Database Relational Schema



Implementation:

RDBMS tables and their description:

Hemployee1:

emp_id	emp_name	emp_designation	emp_dept	emp_addr	emp_phone	emp_salary	emp_date_of_join
E01	Dr.Gajendra Purohit	warden	admin	Chandigarh	1256485963	200000	2021-01-10
E02	Ramesh	Cashier	Finance	Chandigarh	9965485963	120000	2021-01-10
E03	Jatin	Incharge	admin	Chandigarh	8656485963	180000	2021-01-10
E14	Ram	Cook	Mess	Chandigarh	5532485963	25000	2021-01-10
E15	XYZ	Cook	Mess	Chandigarh	7782485963	25000	2021-01-10
E16	ABC	Cook	Mess	Chandigarh	6632885963	16000	2021-01-10
E17	ERD	waiter	Mess	Chandigarh	821485963	9000	2021-01-10
E18	LMN	waiter	Mess	Chandigarh	9648592636	9000	2021-01-10
E19	YUI	waiter	Mess	Chandigarh	8219658665	9000	2021-01-10
E20	CVB	waiter	Mess	Chandigarh	8989626162	9000	2021-01-10
E21	WSC	Security gaurd	Security	Chandigarh	111185963	11000	2021-01-10
E22	ASD	Security Incharge	Security	Chandigarh	223685963	60000	2021-01-10
E23	WSC1	Security gaurd	Security	Chandigarh	963285963	11000	2021-01-10
E24	WSC2	Security gaurd	Security	Chandigarh	777685963	11000	2021-01-10

```
SQL> CREATE TABLE hemployee1 (
  2     emp_id varchar(15) NOT NULL,
  3     emp_name varchar(20) NOT NULL,
  4     emp_designation varchar(30),
  5     emp_dept varchar(15) NOT NULL,
  6     emp_addr varchar(20) NOT NULL,
  7     emp_phone number NOT NULL,
  8     emp_salary number,
  9     emp_date_of_join date,
 10       PRIMARY KEY (emp_id)
 11  );

TABLE created.
```

Student1:

```
SQL> CREATE TABLE student1 (
  2     stud_id varchar(15) NOT NULL,
  3     stud_name varchar(20) NOT NULL,
  4     stud_addr varchar(20) NOT NULL,
  5     stud_dept varchar(15) NOT NULL,
  6     stud_year number NOT NULL,
  7     stud_phone number NOT NULL,
  8     stud_room_no varchar(20) NOT NULL,
  9     PRIMARY KEY (stud_id)
 10  );

TABLE created.
```

stud_id	stud_name	stud_dept	stud_year	stud_room_no	stud_addr	stud_dob	stud_phone
BH1-12	Ananya Trinach	UIET CSE	2	2B/54	Chamba	2002-06-11	8965485926
BH1-16	Anesh Sayal	UIET CSE	2	2A/56	Gurgaon	2003-09-11	9965485926
BH1-17	Akshat Verma	UIET CSE	2	2B/56	Delhi	2004-06-11	9999485926
BH1-22	Aaryan Singh	UIET CSE	2	3B/54	Lucknow	2002-06-11	8865485926
BH1-99	Bibhuti Singha	UIET CSE	2	2B/54	Siliguri	2002-04-11	9932964859

Hroom1:

```
SQL> CREATE TABLE hroom1 (
  2     room_no varchar(20) NOT NULL,
  3     allocated_to varchar(25),
  4     Roommate varchar(25),
  5     no_of_chairs_available varchar(30),
  6     no_of_tables_available varchar(30),
  7     no_of_beds_available varchar(30),
  8     cooler_available varchar(20)
  9  );

TABLE created.
```

Hroom1

room_no	allocated_to	Roommate	no_of_chairs_available	no_of_tables_available	no_of_beds_available	cooler_available
2/54	Bibhuti Singha	Ananya Trinach	3	2	2	yes
2/56	Aneesh Sayal	Akshat Verma	1	1	2	No
1/60	NULL	NULL	1	2	2	NO

Mess_Canteen1:

Mess_canteen1

item_no	menu	price	availibility	timings
1	Aloo paratha	18	everyday	Morning
2	Paneer paratha	18	everyday	Morning
3	Maggi	25	everyday	Morning,Evening
4	Macaroni	25	everyday	Morning,Evening
5	Sandwich	20	everyday	working hours
6	Bread Pakode	20	Weekends	Evening
7	Drinks	25	everyday	working hours
8	Chole Puri	45	Sunday	Lunch
9	Egg Curry	25	Wednesday,Friday	Dinner
10	Regular menu	45	everyday	working hours

```
SQL> CREATE TABLE mess_canteen1 (
  2      item_no number NOT NULL,
  3      menu varchar(20),
  4      price varchar(15),
  5      availibility varchar(20),
  6      timings varchar(20)
  7  );

TABLE created.
```

visitors1:

```
SQL> CREATE TABLE visitors1 (
  2     visitor_id varchar(15) NOT NULL,
  3     visitor_name varchar(20) NOT NULL,
  4     visit_purpose varchar(30),
  5     visit_to varchar(15) NOT NULL,
  6     visitor_addr varchar(20) NOT NULL,
  7     visitor_phone number NOT NULL,
  8     visit_date date,
  9     PRIMARY KEY (visitor_id)
 10  );

TABLE created.
```

Hvisitors1

visitor_id	visitor_name	visit_purpose	visit_to	visitor_addr	visitor_phone	visit_date
V101	Rachit Sharma	meet	BH1-12	Chandigarh	9623562358	2022-12-12
V102	Ram	mess	null	Chandigarh	9965962358	2022-23-12

Fees_bill1:

```
SQL> CREATE TABLE fees_bill1 (
  2      stud_id varchar(15) NOT NULL,
  3      room_no varchar(20),
  4      paid_sem_fees varchar(15),
  5      mess_bill number,
  6      canteen_bill number,
  7      fine number,
  8      PRIMARY KEY (stud_id)
  9  );

TABLE created.
```

Fees_bill1

stud_id	room_no	paid_sem_fees	mess_bill	canteen_bill	fine
BH1-12	2A/54	yes	1500	2500	390
BH1-16	2A/56	yes	1800	300	390
BH1-17	2B/56	yes	900	null	null
BH1-22	3B/54	yes	2215	2500	390
BH1-99	2B/54	yes	650	null	null

Use of Trigger:

```
create or replace TRIGGER employees_audit_trigger
BEFORE INSERT OR DELETE OR UPDATE ON hemployee1
FOR EACH ROW
BEGIN
IF INSERTING THEN
INSERT INTO employee_audit
(emp_id, emp_name, emp_designation, emp_dept, emp_addr, emp_phone, operation_performed, date_op, modified_by)
 VALUES (:NEW.emp_id, :NEW.emp_name, :NEW.emp_designation, :NEW.emp_dept, :NEW.emp_addr, :NEW.emp_phone, :NEW.operation_performed,
  :NEW.date_op, :NEW.modified_by, 'I', sysdate);
ELSIF DELETING THEN
INSERT INTO employee_audit
(emp_id, emp_name, emp_designation, emp_dept, emp_addr, emp_phone, operation_performed, date_op, modified_by
 VALUES (:OLD.emp_id, :OLD.emp_name, :OLD.emp_designation, :OLD.emp_dept, :OLD.emp_addr, :OLD.emp_phone, :OLD.operation_performed,
  :OLD.date_op, :OLD.modified_by, 'D', sysdate);
ELSIF UPDATING THEN
INSERT INTO employee_audit
(emp_id, emp_name, emp_designation, emp_dept, emp_addr, emp_phone, operation_performed, date_op, modified_by
  VALUES (:OLD.emp_id, :OLD.emp_name, :OLD.emp_designation, :OLD.emp_dept, :OLD.emp_addr, :OLD.emp_phone, :OLD.operation_performed,
  :OLD.date_op, :OLD.modified_by, 'U', sysdate);
END IF;
END;
```

Use of Procedure:

```
CREATE OR REPLACE PROCEDURE GRANT_SELECT(
USERNAME VARCHAR2,
GRANTEE VARCHAR2
)
AS
BEGIN
FOR r IN (SELECT TABLE_NAME, OWNER FROM ALL_TABLES
WHERE OWNER = USERNAME)
LOOP
EXECUTE IMMEDIATE
'GRANT SELECT ON ' || r.OWNER || '.' || r.TABLE_NAME || ' TO ' || GRANTEE;
END LOOP;
END;
```

```
GRANT UPDATE(paid_sem_fees,mess_bill,canteen_bill,fine) on fees_bill1 TO cashier1;
```

SQL Commands Used in database:

We have used various SQL commands in our database. These are as follows:

DDL changes the structure of the table like creating a table, altering a table, etc.

- All the command of DDL are auto-committed that means it permanently save all the changes in the database.
- examples : CREATE ,ALTER,DROP

DML commands are used to modify the database. It is responsible for all form of changes in the database.

- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.
- examples :INSERT, UPDATE ,DELETE

DCL commands are used to grant and take back authority from any database user.
Here are some commands that come under DCL:

- Grant
- Revoke

Connecting to MYSQL using mysql.connector Code to Database:

```
from mysql.connector import (connection)

 mydb = mysql.connector.connect(
 host = "localhost",
 user = "root",
 passwd = "hello",
database = "hostel"
)
cursor = mydb.cursor()
```

Tkinter Code to Create GUI :

```
def root():
    root= Tk()


    root.geometry("700x466")
    root.title("HOSTEL DATABASE")
    img =ImageTk.PhotoImage(Image.open(r'C:\coding\project hostel management\1st edit.jpg','r'))
    canvas = Canvas(root, width = 700, height = 466)
    canvas.pack()

    canvas.create_image(20, 20, anchor=NW, image=img)

    ide=Label(root,text='LOGIN PAGE',bg="#041d78",fg="#83e6e6",font=('bold',30))

    ide.place(x=180,y=30)
    but1= Button(root, text="ADMIN LOGIN", font=("italic",20),bg="#83e6e6",command=lambda:[ADMINlogin()])

    but1.place(x=190,y=170)
    but2= Button(root, text="STUDENT LOGIN", font=("italic",20),bg="#83e6e6",command=lambda:[LOG(),root.quit])
    but2.place(x=190,y=250)
    root.mainloop()
root()
exit(0)
```

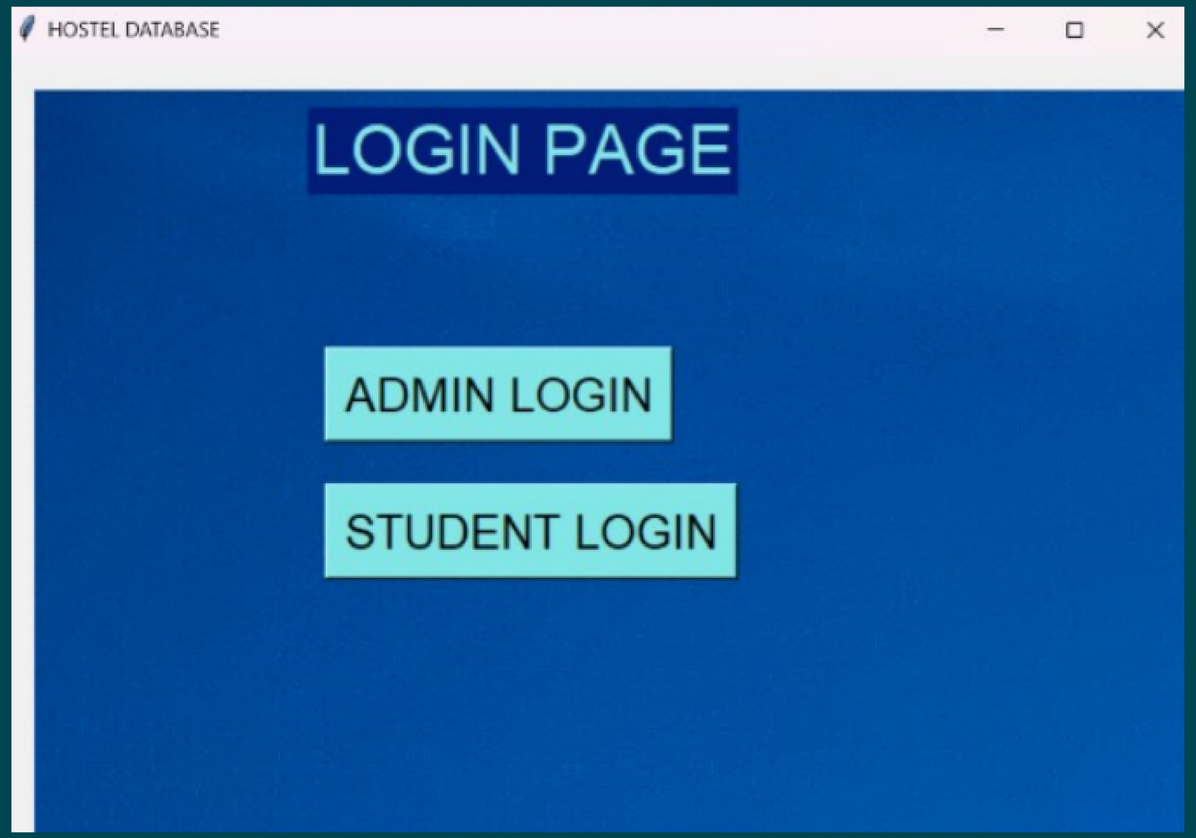
Admin Section:

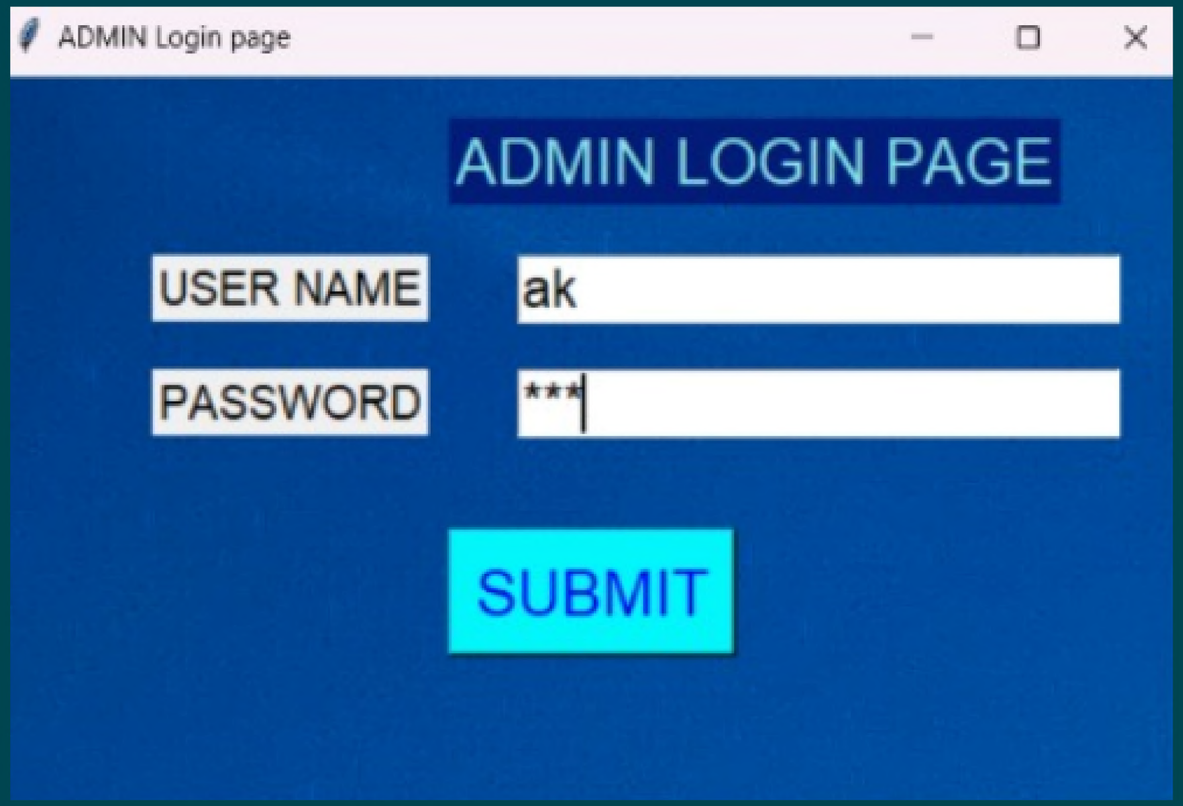
```
def ADMINlogin():

    hostel = Toplevel()
    hostel.geometry("525x328")
    hostel.title("ADMIN Login page")

    bcg=ImageTk.PhotoImage(Image.open(r'C:\coding\project hostel management\1st edit.jpg','r'))
    my_canvas = Canvas(hostel, width=525, height=328)
    my_canvas.pack(fill="both", expand=True)
    my_canvas.create_image(0,0, image=bcg, anchor="nw")
    ide=Label(hostel,text='ADMIN LOGIN PAGE',bg="#041d78",fg="#83e6e6",font=('bold',20))

    ide.place(x=200,y=20)

    name= Label(hostel,text='USER NAME',font=('bold',15))
    name.place(x=70,y=80)


    name = Label(hostel,text='PASSWORD',font=('bold',15))
    name.place(x=70,y=130)
    e1 = Entry(hostel, show=None, font=('Arial', 17))
    e2 = Entry(hostel, show='*', font=('Arial', 17))
    e1.place(x=230,y=80)
    e2.place(x=230,y=130)

    def submit():
        name=e1.get()
        password=e2.get()
        if name=='ak' or password=='123':
            option()
    submits= Button(hostel, text='SUBMIT', font=("italic",20),bg="#05f6fa",fg="blue",command=submit)
    submits.place(x=200,y=200)
    hostel.mainloop()
```

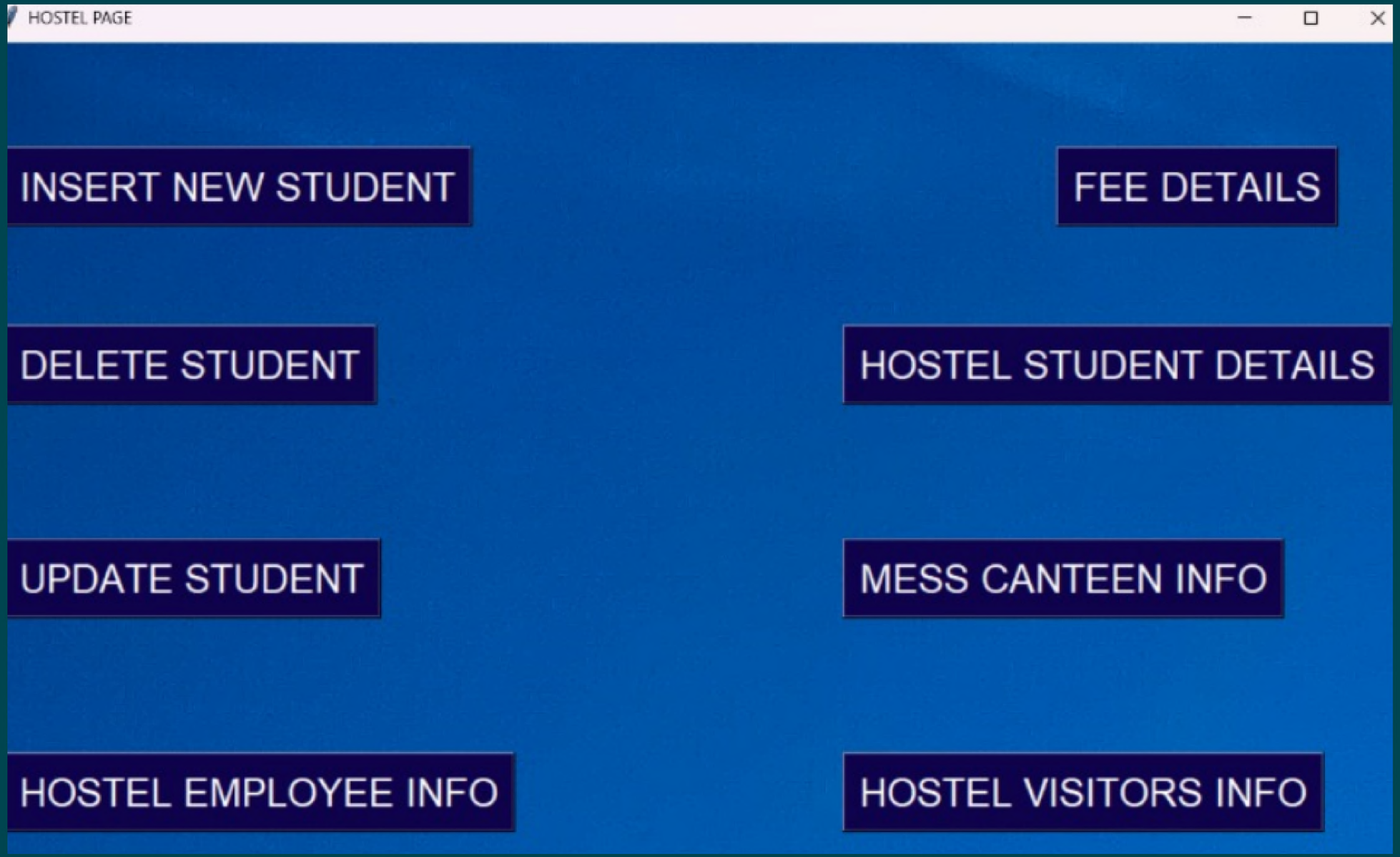
Student Section:

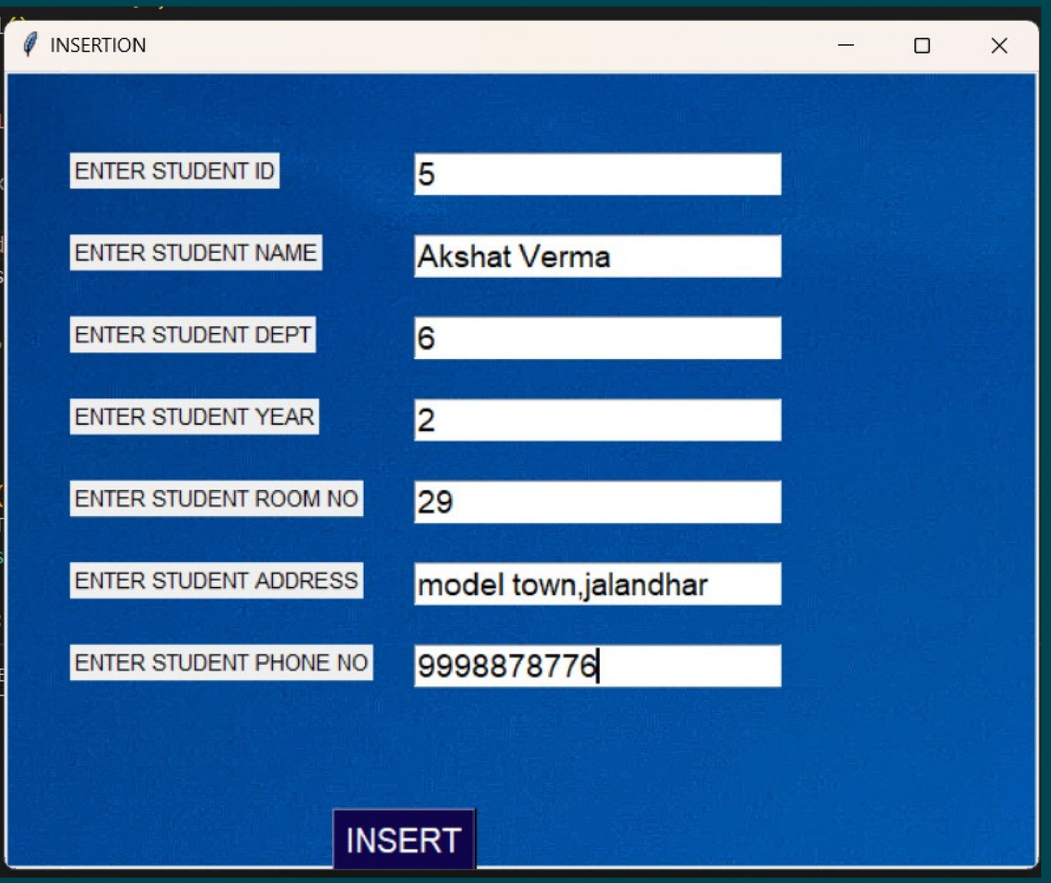
```
def Hstudent_details():
    show1=Tk()
    mydb = connection.MySQLConnection(
                host = "localhost",
                user = "root",
                passwd = "hello",
                database = "hostel"
                )
    cursor = mydb.cursor()
    cursor.execute("SELECT * FROM test;")
    student1=cursor.fetchall()

    show1.title("SHOW")
    show1.configure(bg='#041d78')

    label = Label(show1, text="Student Records", font=("Arial",10)).grid(row=0, columnspan=3)

    cols = ('stud_id', 'stud_name', 'stud_dept','stud_year','stud_room_no','stud_addr','stud_phone')
    listBox = ttk.Treeview(show1, columns=cols, show='headings')
    for col in cols:
        listBox.heading(col, text=col)
        listBox.grid(row=1, column=0, columnspan=2)
    i=0
    for student in student1:
        for j in range(len(student)):
            e = Entry(listBox, width=20, fg='blue')
            e.grid(row=i, column=j)
            e.insert(END, student[j])
        i=i+1
    show1.mainloop()
    mydb.close()
```

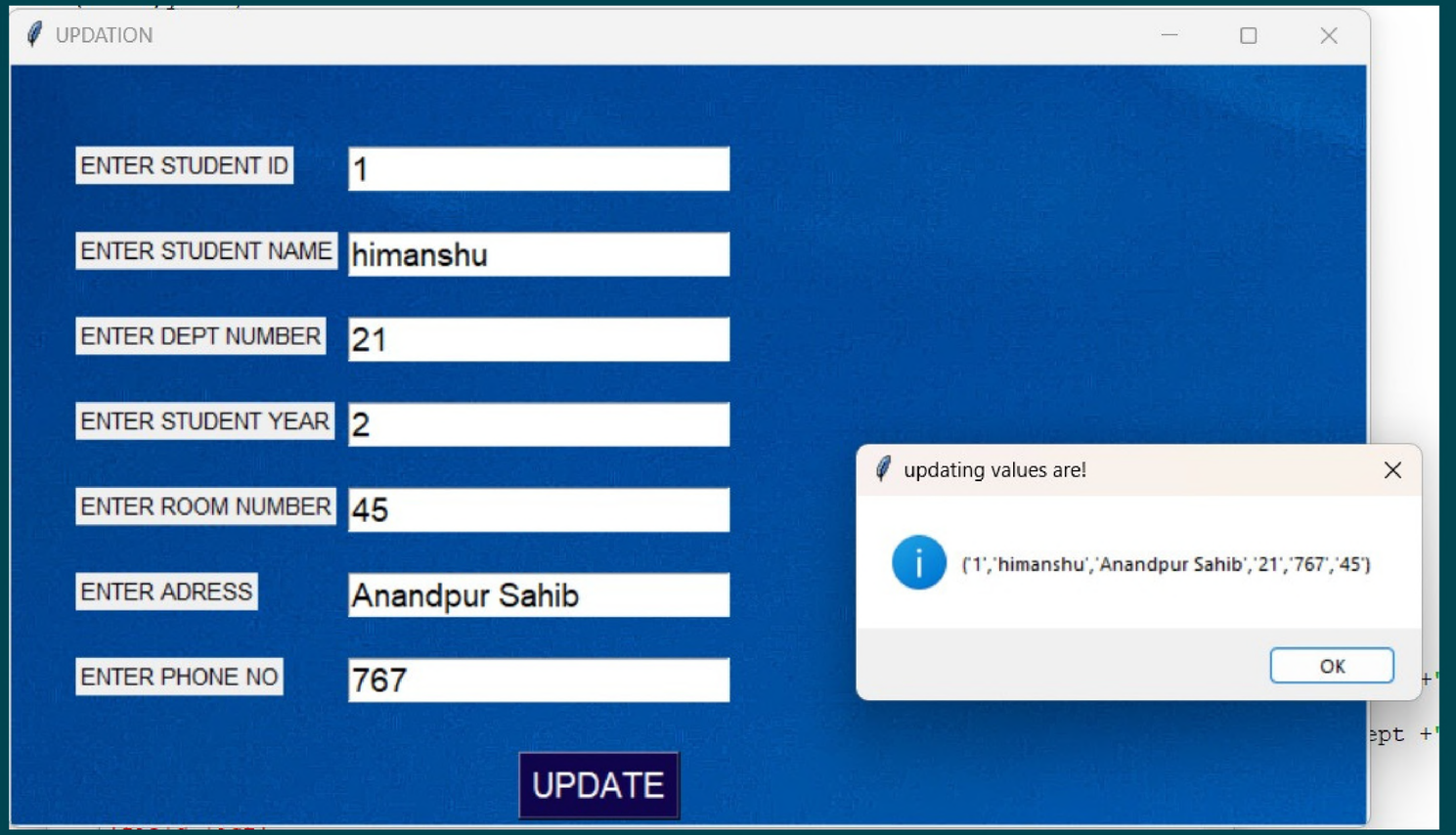
Interface


HOSTEL DATABASE
LOGIN PAGE
ADMIN LOGIN
STUDENT LOGIN


ADMIN Login page
ADMIN LOGIN PAGE
USER NAME
ak
PASSWORD
***
SUBMIT


HOSTEL PAGE
INSERT NEW STUDENT
FEE DETAILS
DELETE STUDENT
HOSTEL STUDENT DETAILS
UPDATE STUDENT
MESS CANTEEN INFO
HOSTEL EMPLOYEE INFO
HOSTEL VISITORS INFO


INSERTION
ENTER STUDENT ID
5
ENTER STUDENT NAME
Akshat Verma
ENTER STUDENT DEPT
6
ENTER STUDENT YEAR
2
ENTER STUDENT ROOM NO
29
ENTER STUDENT ADDRESS
model town,jalandhar
ENTER STUDENT PHONE NO
9998878776
INSERT


UPDATION
ENTER STUDENT ID
1
ENTER STUDENT NAME
himanshu
ENTER DEPT NUMBER
21
ENTER STUDENT YEAR
2
ENTER ROOM NUMBER
45
ENTER ADRESS
Anandpur Sahib
ENTER PHONE NO
767
UPDATE
updating values are!
('1','himanshu','Anandpur Sahib','21','767','45')
OK

Some Limitations :

Although this project covers most of the aspects related to the hostel , there are some sections which are beyond the scope of this project:

- No section for the hostel residents to register their complaints.
- A proper web application for the hostel management would be more effective making the database centralized by which multiuser can work on the system.
- The system is not able to generate the PDF, CSV reporting yet.

Future scope of Enhancement:

1. **Mobile compatibility** : Making the system accessible on mobile devices, allowing hostel managers and residents to manage their operations on-the-go.
2. **Online payments** : Integrating an online payment system to allow residents to pay their fees and other charges through the platform.
3. **Room booking and allocation** : Implementing a room booking and allocation system to automate the process of booking rooms and allocating them to residents.
4. **Maintenance management** : Adding a module to manage maintenance requests and keep track of maintenance activities.
5. **Reporting and analytics** : Providing advanced reporting and analytics features to help hostel managers make informed decisions and track key performance indicators.
6. **User-friendly interface**: Improving the user interface and user experience to make the system even more accessible and user-friendly.

Conclusion

This project gave us the idea about, how large data are stored inside a database and organised so that it can be retrieved easily and in a more efficient way. It also helped us in learning to create application using Python code and connecting the back end with the front end using the python and mysql-connector code, so that any actions that are performed in the front end are reflected in the back end and also any modifications made at the back end can also be seen in the front end. It also gave us complete idea about how the queries retrieve data from multiple tables and the working of structured procedure and the triggers. .

RESOURCES USED :

- www.google.com
- www.geeksforgeeks.org
- Youtube/CodeWithHarry

Acknowledgement

We would like to express our special thanks of gratitude to our teacher ***Dr. Mamta Juneja*** who gave us this project , for which we got to do many research which ultimately helped us in broadening our knowledge for this subject. We learnt a lot of new things while making this project.

Presented to: Dr. Mamta Juneja
Associate professor
UIET , Chandigarh

THANK YOU !