

**APARTMENT**

**MANAGEMENT**

**SYSTEM**

**A PROJECT REPORT**

***Submitted by***

**RAMPRABU P-21BIT036**

**RUPAK G B- 21BIT039**

***in partial fulfilment for the award of the***

***degree of***

BACHELOR OF TECHNOLOGY

**in**

INFORMATION TECHNOLOGY **KUMARAGURU**

**COLLEGE OF TECHNOLOGY COIMBATORE**

**(An autonomous institution affiliated to Anna University, Chennai)**

I

**BONAFIDE CERTIFICATE**

Certified that this project report “**APARTMENT MANAGEMENT SYSTEM**” is the bonafide work of ***RAMPRABU P(21BIT036) and RUPAK G B (21BIT039)*** who carried out the project work under my supervision.

SIGNATURE SIGNATURE

**Dr. M. ALAMELU Dr. P.SHENBAGAM**

**HEAD OF THE DEPARTMENT SUPERVISOR**

Professor Assistant Professor

Information Technology Information Technology

**TABLE OF CONTENTS PAGE NO.**

**DECLARATION 4**

**ABSTRACT 5**

**SYSTEM STUDY 6**

**EXISTING SYSTEM 6**

**PROPOSED SYSTEM 7**

**SYSTEM DESIGN 8**

**ER DIAGRAM 8**

**TABLE DESIGN 9**

**TABLE CREATION AND QUERIES 10**

**INTEGRATION OF FRONT END 14**

**SCREENSHOTS 41**

**CONCLUSION 50**

**REFERENCES 51**

# DECLARATION

We ***RAMPRABU P(21BIT036) and RUPAK G B (21BIT039)*,**

hereby declare that the project “**APARTMENT MANAGEMENT SYSTEM**” is done by us and to the study of our knowledge, a similar work has not been submitted to any other institution, for the fulfilment of the required course of study. This report is submitted on the partial fulfilment of the requirements for all awards of the Degree of Bachelor of Information Technology at Kumaraguru College of Technology, Coimbatore. We certify that the declaration made above by the candidates is true.

|  |  |
| --- | --- |
| Place: Coimbatore | **Dr. P. SHENBAGAM**  Assistant Professor |
| Date: | Information Technology |

# ABSTRACT

Our Apartment Management System aims at providing a database system for the residents of the apartment and for the people who are looking for apartment and also for the managing team of the apartment.

The prime objective is to create table for storing the details about blocks, their floors and the apartments in each block and also the people who resides in that apartment.

Our project also includes features like subscriptions for recreations, water and electricity bills, visiting log and parking lot.

The project also contains features to delete the residents and if the owner is deleted then the ownership will be handled well by passing it to senior citizen of the resident.

The visiting log is maintained by keeping track of the current date and time while checking in and out. The subscription log will be added by the admin based on their usage. Then the system will be kept track of the mode of payment the reference id if online payment and the paid date.

# SYSTEM STUDY

# EXISITING SYSTEM:

The current system is a manual based which is not computerized especially for the residents of the apartment. Also this system takes a lot of time for performing different activities and difficult to maintain the visitors’ records.

Limitations of Existing System:

* Existing System is completely hard-copy.
* Time consuming procedure.
* Tracking of records is difficult.

On a whole, every year the company is designing new apartments and its a difficult task to manage the records of each and every apartment in the manual system. It will not only take a lot of time but also increases the chances of errors. This may create a problem when you need details of any particular project.

# PROPOSED SYSTEM:

Ours will be a robust database which stores all the data related to these various apartments, their maintenance related information etc.

It helps to keep track of all the payments done by customers towards maintenance advances or purchases etc.

This tool will ease the user to manage huge data of different customers who own apartments. For the maintenance part of the interface it shall have the data of all the maintenance charges paid by the customers, whether its done quarterly or annually.

It also records the mode of payment. It will also hold all the details such as the apartment number, subscription log, visiting log etc.

**SYSTEM REQUIREMENTS:**

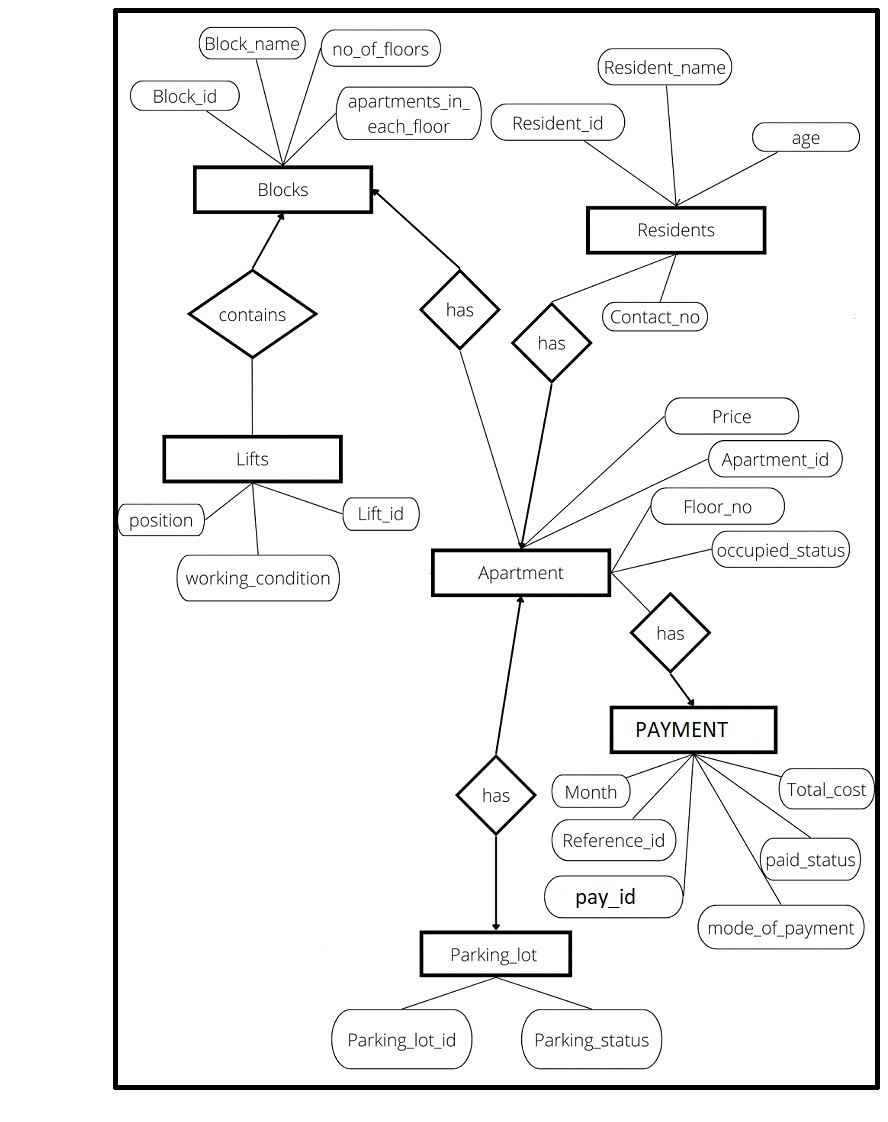
1. Software:
   1. OS: Linux/Windows/Mac OS

* 1. Languages Used: MYSQL – RDBMS, JAVA – Backend Connectivity

* 1. Environment: Eclipse/Net Beans/ VS Code

1. Hardware:
   1. CPU:
   2. RAM:
   3. HARD DISK:

# SYSTEM DESIGN



# TABLE DESIGN

|  |  |  |
| --- | --- | --- |
| **S.no** | **Table name** | **Attributes** |
| **1** | Block | Block(block\_id,block\_name,no\_of\_floors, apartments\_each\_floor,no\_of\_lifts) |
| **2** | Apartment | Apartment(Apartment\_id, floor\_number, price, occupied, owner\_id ) |
| **3** | Residents | Residents(Resident\_id, name, gender, age, blood\_group, contact) |
| **4** | Lifts | Lifts(lift\_id, work\_condition,position) |
| **5** | Parking\_lot | Parking\_lot(parking\_lot\_id,parking\_status) |
| **6** | Payment | Payment(payment\_id, bill\_arrived\_date, bill\_paid\_date, total\_cost, mode\_of\_payment,ref\_id,\_apartment\_id) |
|  |  |  |

**SYSTEM IMPLEMENTATION**

1. **Create Database Comment:**

i. create database Apartment\_Management;

1. **Create Table Comments**

## Blocks

create table Blocks(Block\_id int auto\_increment primary key,block\_name varchar(20),no\_of\_floors int not null,no\_of\_apartments\_in\_each\_floor int not null,no\_of\_lifts int not null);

## Apartment

create table Apartment(Apartment\_id int auto\_increment primary key,floor\_number int not null,price double not null,Block\_id int not null,foreign key(Block\_id) references Blocks(Block\_id));

## Residents

create table Residents(Resident\_id int auto\_increment primary key,name varchar(50) not null,age int not null,Apartment\_id int not null,foreign key(Apartment\_id) references Apartment(Apartment\_id));

### III. Check constraint for Apartment

alter table Apartment add foreign key(owner\_id) references

Residents(Resident\_id);

## occupied

create table occupied(Apartment\_id int not null primary key,owner\_id

int,status boolean,foreign key(Apartment\_id) references Apartment( Apartment\_id),foreign key(owner\_id) references Residents(Resident\_id));

### IV. Check constraint for occupied

alter table occupied add check(status = true or(status = false and owner\_id is null));

## lifts

create table lifts(lift\_id int auto\_increment primary

key,working\_condition boolean,position varchar(7),Block\_id int not null, foreign key(Block\_id) references Blocks(Block\_id));

## Parking\_lot

create table Parking\_lot(parking\_lot\_id int auto\_increment primary key, parking\_status boolean,Apartment\_id int not null,foreign key

(Apartment\_id) references Apartment(Apartment\_id));

## payment

create table payment(pay\_id int auto\_increment primary key, bill\_arrived\_date date not null,bill\_paid\_date date,total\_cost double,mode\_of\_payment varchar(10),ref\_id int,Apartment\_id int not null,foreign key(Apartment\_id) references Apartment(Apartment\_id));

### V. Check constraint for Subscription\_log

alter table Subscription\_log add check(ref\_id is not null or(ref\_id is null and mode\_of\_payment="online"));

Contact create table Contact(Resident\_id int not null,contact\_number varchar(12),primary key(Resident\_id,contact\_number),foreign key(Resident\_id) references Residents(Resident\_id));

## **Sample for Insertion Commands**

a. Blocks Table

insert into Blocks values(1,'A',3,4,2),(2,'B',3,4,3);

### b. Apartment Table

insert into Apartment(Apartment\_id,floor\_number,price,Block\_id) values(101,0,3500000,1),(102,0,3500000,1),(103,0,3500000,1),(10 4,0,3500000,1);

#### c. Residents Table

insert into Residents

values(1,'Vijay','M',30,'O+ve',101),(2,'Shalini','F',29,'B+ve',101);

**Setting owner id for an apartment**

update Apartment set owner\_id=1 where Apartment\_id=101;

## **Join Command to use find the owner name of the Apartment**

select name from Residents inner join Apartment on Apartment.owner\_id=Residents.Resident\_id;

lifts insert into lifts values(1,1,'front',1),(2,1,'back',1);

Parking\_lot insert into Parking\_lot values(1,1,101),(2,1,113);

Contact insert into Contact

values(1,'9095014010'),(2,'9002210555'),(3,'9940891672'),(4,'888 0596912'),(4,'9990011357');

occupied insert into occupied values(101,1,1),(113,4,1); Subsciption\_log insert into Subscription\_log values(1,'2022-05-01','2022-05-

05',10000,'online',10421,101),(2,'2022-05-03','2022-05-

06',20000,'online',10422,113);

**Source Code for JDBC Connectivity and Menu Driven Program**

**ADMIN LOGIN-PAGE:**

from tkinter import \*

from tkinter import messagebox

from PIL import ImageTk

def login():

if unEntry.get() == '' or pwEntry.get() == '':

messagebox.showerror('Error','Fields cannot be empty')

elif unEntry.get()=='admin' and pwEntry.get()=='admin':

messagebox.showinfo('Success','Welcome')

window.destroy()

import admin\_page

else:

messagebox.showerror('Error','Please enter correct credentials')

window=Tk()

window.geometry('1525x760+0+0')

window.title('Admin Login System of Apartment Management System')

window.resizable(0,0)

backImage=ImageTk.PhotoImage(file='lpg.webp')

label1=Label(window,image=backImage)

label1.pack()

loginFrame=Frame(window,bg='white')

loginFrame.place(x=400,y=150)

logoImage=PhotoImage(file='alogo.png')

label2=Label(loginFrame,image=logoImage)

label2.grid(row=0,column=1,columnspan=2,pady=10)

unImage=PhotoImage(file='user.png')

unLabel=Label(loginFrame,image=unImage,text='Adminname',compound=LEFT,font=('times new roman',20,'bold'),bg='white')

unLabel.grid(row=1,column=1,pady=10,padx=20)

unEntry=Entry(loginFrame,font=('times new roman',20,'bold'),bd=5,fg='royalblue')

unEntry.grid(row=1,column=2,pady=10,padx=20)

pwImage = PhotoImage(file='pass.png')

pwLabel = Label(loginFrame, image=pwImage, text='Password', compound=LEFT, font=('times new roman', 20, 'bold'),

bg='white')

pwLabel.grid(row=2, column=1, pady=10, padx=20)

pwEntry = Entry(loginFrame, font=('times new roman', 20, 'bold'), bd=5, fg='royalblue')

pwEntry.grid(row=2, column=2, pady=10, padx=20)

loginButton=Button(loginFrame,text='Login',font=('times new roman', 14, 'bold'),width=15,fg='white'

,bg='cornflowerblue',activebackground='cornflowerblue',activeforeground='white',cursor='hand2'

,command=login)

loginButton.grid(row=3,column=2,pady=10)

window.mainloop()

**ADMIN-PAGE:**

from tkinter import \*

import time

import ttkthemes

from tkinter import ttk,messagebox

import mysql.connector

mydb=mysql.connector.connect(

host="localhost",

username="root",

passwd="ram9486",

database="apart"

)

if mydb.is\_connected():

print("connection established")

mycursor=mydb.cursor()

def exit():

result=messagebox.askyesno('CONFIRM','DO YOU WANT TO LOG-OUT')

if result:

root.destroy()

import admin\_login

else:

pass

def clock():

date=time.strftime('%d/%m/%Y')

currenttime=time.strftime('%H:%M:%S')

datetimeLabel.config(text=f' Date: {date}\nTime: {currenttime}')

datetimeLabel.after(1000,clock)

def addapp():

def add\_data():

global mycursor,mydb

if floorEntry.get()=='' or priceEntry.get()=='' or BlockEntry.get()=='':

messagebox.showerror('Error','All fields are required',parent=addwin)

else:

query='insert into apartment(Apartment\_id,floor\_number,price,block\_id) values(%s,%s,%s,%s)'

mycursor.execute(query,(idEntry.get(),floorEntry.get(),priceEntry.get(),BlockEntry.get()))

result=messagebox.askyesno('Confirm','Data added successfully.Do you want to make any change?',parent=addwin)

if result:

idEntry.delete(0,END)

floorEntry.delete(0,END)

priceEntry.delete(0,END)

BlockEntry.delete(0,END)

else:

pass

mydb.commit()

addwin=Toplevel()

addwin.grab\_set()

addwin.resizable(False,False)

idLabel=Label(addwin,text='Appartment\_id',font=('times new roman',20,'bold'))

idLabel.grid(row=0,column=0,padx=30,pady=15,stick=W)

idEntry=Entry(addwin,font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

idEntry.grid(row=0,column=1,pady=15,padx=10)

floorLabel = Label(addwin, text='Floor no', font=('times new roman', 20, 'bold'))

floorLabel.grid(row=2, column=0, padx=30, pady=15,stick=W)

floorEntry = Entry(addwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

floorEntry.grid(row=2, column=1, pady=15, padx=10)

priceLabel = Label(addwin, text='Price', font=('times new roman', 20, 'bold'))

priceLabel.grid(row=4, column=0, padx=30, pady=15,stick=W)

priceEntry = Entry(addwin,font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

priceEntry.grid(row=4, column=1, pady=15, padx=10)

BlockLabel = Label(addwin, text='Block\_id', font=('times new roman', 20, 'bold'))

BlockLabel.grid(row=6, column=0, padx=30, pady=15,stick=W)

BlockEntry = Entry(addwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

BlockEntry.grid(row=6, column=1, pady=15, padx=10)

addbutton=ttk.Button(addwin,text='SUBMIT',command=add\_data)

addbutton.grid(row=7,columnspan=2,pady=15)

def addres():

def add\_rdata():

global mycursor,mydb

apid = apidEntry.get()

query=f"select Apartment\_id from apartment where Apartment\_id='{apid}'"

mycursor.execute(query)

row = mycursor.fetchone()

if row is None:

messagebox.showerror('Error','Apartment not Avilable ')

addresidwin.destroy()

if ridEntry.get()=='' or nameEntry.get()=='' or apidEntry.get()=='' or ageEntry.get()=='' or phoneEntry.get()=='':

messagebox.showerror('Error','All fields are required',parent=addresidwin)

try:

query='insert into residents(Resident\_id,name,Apartment\_id,age,phone) values(%s,%s,%s,%s,%s)'

mycursor.execute(query,(ridEntry.get(),nameEntry.get(),apidEntry.get(),ageEntry.get(),phoneEntry.get()))

result=messagebox.askyesno('Confirm','Data added successfully.Do you want to make any change?',parent=addresidwin)

if result:

ridEntry.delete(0,END)

nameEntry.delete(0,END)

ageEntry.delete(0,END)

apidEntry.delete(0,END)

phoneEntry.delete(0,END)

else:

pass

mydb.commit()

except:

messagebox.showerror('Error','Id cannot be repeted',parent=addresidwin)

return

query='select \* from residents'

mycursor.execute(query)

fetched\_data=mycursor.fetchall()

resTable.delete(\*resTable.get\_children())

for data in fetched\_data:

datalist=list(data)

resTable.insert('',END,values=datalist)

addresidwin=Toplevel()

addresidwin.grab\_set()

addresidwin.resizable(False,False)

ridLabel=Label(addresidwin,text='Resident\_id',font=('times new roman',20,'bold'))

ridLabel.grid(row=0,column=0,padx=30,pady=15,stick=W)

ridEntry=Entry(addresidwin,font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

ridEntry.grid(row=0,column=1,pady=15,padx=10)

nameLabel = Label(addresidwin, text='Name', font=('times new roman', 20, 'bold'))

nameLabel.grid(row=2, column=0, padx=30, pady=15,stick=W)

nameEntry = Entry(addresidwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

nameEntry.grid(row=2, column=1, pady=15, padx=10)

ageLabel = Label(addresidwin, text='Age', font=('times new roman', 20, 'bold'))

ageLabel.grid(row=4, column=0, padx=30, pady=15,stick=W)

ageEntry = Entry(addresidwin,font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

ageEntry.grid(row=4, column=1, pady=15, padx=10)

phoneLabel = Label(addresidwin, text='Phone\_no', font=('times new roman', 20, 'bold'))

phoneLabel.grid(row=6, column=0, padx=30, pady=15,stick=W)

phoneEntry = Entry(addresidwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

phoneEntry.grid(row=6, column=1, pady=15, padx=10)

apidLabel = Label(addresidwin, text='Apartment\_id', font=('times new roman', 20, 'bold'))

apidLabel.grid(row=8, column=0, padx=30, pady=15, stick=W)

apidEntry = Entry(addresidwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange', width=24)

apidEntry.grid(row=8, column=1, pady=15, padx=10)

addbutton=ttk.Button(addresidwin,text='SUBMIT',command=add\_rdata)

addbutton.grid(row=9,columnspan=2,pady=15)

def editpay():

def submit():

apartment\_id = apartment\_id\_entry.get()

a\_id = int(apartment\_id)

a\_date = a\_date\_entry.get()

cost = total\_cost\_entry.get()

query = 'INSERT INTO payment (bill\_arrived\_date, total\_cost, Apartment\_id) VALUES (%s, %s, %s)'

values = (a\_date, cost, a\_id)

try:

mycursor.execute(query, values)

mydb.commit()

messagebox.showinfo("Success", "Subscription added successfully")

except mysql.connector.Error as e:

messagebox.showerror("Error", str(e))

finally:

if mycursor:

mycursor.close()

edpaywin = Toplevel()

edpaywin.grab\_set()

edpaywin.resizable(False, False)

apartment\_id\_label = Label(edpaywin, text="Enter the apartment id:",font=('times new roman',20,'bold'))

apartment\_id\_label.grid(row=0,column=0,padx=30,pady=15,stick=W)

apartment\_id\_entry = Entry(edpaywin,font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

apartment\_id\_entry.grid(row=0,column=1,pady=15,padx=10)

a\_date\_label = Label(edpaywin, text="Enter the Bill Arrived Date:",font=('times new roman',20,'bold'))

a\_date\_label.grid(row=2,column=0,padx=30,pady=15,stick=W)

a\_date\_entry = Entry(edpaywin,font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

a\_date\_entry.grid(row=2,column=1,pady=15,padx=10)

total\_cost\_label = Label(edpaywin, text="Enter the total cost:",font=('times new roman',20,'bold'))

total\_cost\_label.grid(row=4,column=0,padx=30,pady=15,stick=W)

total\_cost\_entry = Entry(edpaywin,font=('times new roman', 20, 'bold'), bd=5, fg='dark orange',width=24)

total\_cost\_entry.grid(row=4,column=1,pady=15,padx=10)

submit\_button=ttk.Button(edpaywin,text="Submit",command=submit)

submit\_button.grid(row=6,columnspan=2,pady=15)

def parking():

global mycursor, mydb

def topparkin():

pkwin = Toplevel()

pkwin.title('Parking')

pkwin.grab\_set()

pkwin.resizable(False, False)

scrollbarx = Scrollbar(pkwin, orient=HORIZONTAL)

scrollbary = Scrollbar(pkwin, orient=VERTICAL)

parktreeTable = ttk.Treeview(pkwin, columns=('Parking\_lot\_id', 'Parking\_status','Apartment\_id'),

xscrollcommand=scrollbarx.set, yscrollcommand=scrollbary.set)

scrollbarx.config(command=parktreeTable.xview)

scrollbary.config(command=parktreeTable.yview)

scrollbarx.pack(side=BOTTOM, fill=X)

scrollbary.pack(side=RIGHT, fill=Y)

parktreeTable.pack(fill=BOTH, expand=1)

parktreeTable.heading('Parking\_lot\_id', text='Parking\_lot\_id')

parktreeTable.heading('Parking\_status', text='Parking\_status')

parktreeTable.heading('Apartment\_id', text='Apartment\_id')

parktreeTable.column('Parking\_lot\_id', width=50, anchor=CENTER)

parktreeTable.column('Apartment\_id', width=200, anchor=CENTER)

parktreeTable.column('Parking\_status', width=200, anchor=CENTER)

style = ttk.Style()

style.configure('Treeview', rowheight=40, font=('arial', 12, 'bold'), foreground='black', background='white'

, fieldbackground='white')

style.configure('Treeview.Heading', font=('arial', 14, 'bold'))

parktreeTable.config(show='headings')

query = 'select \* from parking\_lot'

mycursor.execute(query)

parktreeTable.delete(\*parktreeTable.get\_children())

fetchdata = mycursor.fetchall()

for data in fetchdata:

parktreeTable.insert('', END, values=data)

def submit():

query = 'delete from parking\_lot where Apartment\_id=%s'

apid = apartment\_id\_entry.get()

mycursor.execute(query, (apid,))

mydb.commit()

choice = parking\_status\_var.get()

a\_id = apartment\_id\_entry.get()

query = "INSERT INTO parking\_lot (parking\_status, Apartment\_id) VALUES (%s, %s)"

mycursor.execute(query, (choice, a\_id))

mydb.commit()

messagebox.showinfo("Success", "Parking entry added successfully.")

topparkin()

parkwin = Toplevel()

parkwin.grab\_set()

parkwin.resizable(False, False)

apartment\_id\_label = Label(parkwin, text="Enter the apartment id:", font=('times new roman', 20, 'bold'))

apartment\_id\_label.grid(row=0, column=0, padx=30, pady=15, stick=W)

apartment\_id\_entry = Entry(parkwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange', width=24)

apartment\_id\_entry.grid(row=0, column=1, pady=15, padx=10)

parking\_status\_label = Label(parkwin, text="Parking Status (1 for available, 0 for occupied):")

parking\_status\_label.grid(row=2, column=0, padx=30, pady=15, stick=W)

parking\_status\_var = IntVar()

park\_status\_radio1 = ttk.Radiobutton(parkwin, text="Yes", variable=parking\_status\_var , value=1)

park\_status\_radio1.grid(row=2, column=1, pady=15, padx=2)

park\_status\_radio2 = ttk.Radiobutton(parkwin, text="No", variable=parking\_status\_var , value=0)

park\_status\_radio2.grid(row=2, column=2, pady=15, padx=2)

submit\_button = ttk.Button(parkwin, text="Submit", command=submit)

submit\_button.grid(row=3, columnspan=2, pady=15)

def searchres():

def search\_rdata():

global mycursor, mydb

query='select \* from residents where Resident\_id=%s or name=%s or Apartment\_id=%s or age=%s or phone=%s '

mycursor.execute(query,(ridEntry.get(),nameEntry.get(),apidEntry.get(),ageEntry.get(),phoneEntry.get()))

resTable.delete(\*resTable.get\_children())

fetchdata=mycursor.fetchall()

for data in fetchdata:

resTable.insert('',END,values=data)

searchwin = Toplevel()

searchwin.title('Search resident')

searchwin.grab\_set()

searchwin.resizable(False, False)

ridLabel = Label(searchwin, text='Resident\_id', font=('times new roman', 20, 'bold'))

ridLabel.grid(row=0, column=0, padx=30, pady=15, stick=W)

ridEntry = Entry(searchwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange', width=24)

ridEntry.grid(row=0, column=1, pady=15, padx=10)

nameLabel = Label(searchwin, text='Name', font=('times new roman', 20, 'bold'))

nameLabel.grid(row=2, column=0, padx=30, pady=15, stick=W)

nameEntry = Entry(searchwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange', width=24)

nameEntry.grid(row=2, column=1, pady=15, padx=10)

ageLabel = Label(searchwin, text='Age', font=('times new roman', 20, 'bold'))

ageLabel.grid(row=4, column=0, padx=30, pady=15, stick=W)

ageEntry = Entry(searchwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange', width=24)

ageEntry.grid(row=4, column=1, pady=15, padx=10)

phoneLabel = Label(searchwin, text='Phone\_no', font=('times new roman', 20, 'bold'))

phoneLabel.grid(row=6, column=0, padx=30, pady=15, stick=W)

phoneEntry = Entry(searchwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange', width=24)

phoneEntry.grid(row=6, column=1, pady=15, padx=10)

apidLabel = Label(searchwin, text='Apartment\_id', font=('times new roman', 20, 'bold'))

apidLabel.grid(row=8, column=0, padx=30, pady=15, stick=W)

apidEntry = Entry(searchwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange', width=24)

apidEntry.grid(row=8, column=1, pady=15, padx=10)

addbutton = ttk.Button(searchwin, text='SEARCH', command=search\_rdata)

addbutton.grid(row=9, columnspan=2, pady=15)

def deleteres():

selitem=resTable.selection()[0]

rid=resTable.item(selitem)['values'][0]

query='delete from residents where Resident\_id= %s '

seldata=(rid,)

mycursor.execute(query,seldata)

mydb.commit()

messagebox.showinfo('Deleted',f'resident {rid} is deleted ')

query1='select \* from residents'

mycursor.execute(query1)

fetchdata=mycursor.fetchall()

resTable.delete(\*resTable.get\_children())

for data in fetchdata:

resTable.insert('',END,values=data)

def viewall():

query1 = 'select \* from residents'

mycursor.execute(query1)

fetchdata = mycursor.fetchall()

resTable.delete(\*resTable.get\_children())

for data in fetchdata:

resTable.insert('', END, values=data)

def unpaybill():

upiwin = Toplevel()

upiwin.title('Unpayed bills')

upiwin.grab\_set()

upiwin.resizable(False, False)

scrollbarx = Scrollbar(upiwin, orient=HORIZONTAL)

scrollbary = Scrollbar(upiwin, orient=VERTICAL)

unpayTable = ttk.Treeview(upiwin, columns=('Pay\_id', 'Apartment\_id','Arrived\_date','Total\_cost'),

xscrollcommand=scrollbarx.set, yscrollcommand=scrollbary.set)

scrollbarx.config(command=unpayTable.xview)

scrollbary.config(command=unpayTable.yview)

scrollbarx.pack(side=BOTTOM, fill=X)

scrollbary.pack(side=RIGHT, fill=Y)

unpayTable.pack(fill=BOTH, expand=1)

unpayTable.heading('Pay\_id', text='Pay\_id')

unpayTable.heading('Apartment\_id', text='Apartment\_id')

unpayTable.heading('Arrived\_date', text='Arrived\_date')

unpayTable.heading('Total\_cost', text='Total\_cost')

unpayTable.column('Pay\_id', width=50, anchor=CENTER)

unpayTable.column('Apartment\_id', width=200, anchor=CENTER)

unpayTable.column('Arrived\_date', width=200, anchor=CENTER)

unpayTable.column('Total\_cost', width=100, anchor=CENTER)

style = ttk.Style()

style.configure('Treeview', rowheight=40, font=('arial', 12, 'bold'), foreground='black', background='white'

, fieldbackground='white')

style.configure('Treeview.Heading', font=('arial', 14, 'bold'))

unpayTable.config(show='headings')

query = 'select pay\_id,Apartment\_id,bill\_arrived\_date,total\_cost from payment where bill\_paid\_date is null and pay\_id is not null'

mycursor.execute(query)

unpayTable.delete(\*unpayTable.get\_children())

fetchdata = mycursor.fetchall()

for data in fetchdata:

unpayTable.insert('', END, values=data)

count=0

text=''

def slider():

global text,count

if count==len(a):

count=0

text=''

text=text+a[count]

sliderLabel.config(text=text)

count+=1

sliderLabel.after(300,slider)

root=ttkthemes.ThemedTk()

root.get\_themes()

root.set\_theme('radiance')

root.geometry('1474x750+0+0')

root.resizable(0,0)

root.title('Apartment Management System')

datetimeLabel=Label(root,font=('times new roman',18,'bold'))

datetimeLabel.place(x=5,y=5)

clock()

a='Apartment Management System'

sliderLabel=Label(root,text=a,font=('arial',28,'italic bold'),width=50)

sliderLabel.place(x=200,y=0)

slider()

connectButton=ttk.Button(root,text='LOG OUT',command=exit)

connectButton.place(x=1300,y=0)

leftFrame=Frame(root)

leftFrame.place(x=50,y=80,width=300,height=600)

logoimage=PhotoImage(file='resident.png')

logoLabel=Label(leftFrame,image=logoimage)

logoLabel.grid(row=0,column=0)

addflatbutton=ttk.Button(leftFrame,text='Add Appartment',cursor='hand2',width=20,command=addapp)

addflatbutton.grid(row=1,column=0,pady=10)

adresibutton=ttk.Button(leftFrame,text='Add new resident',cursor='hand2',width=20,command=addres)

adresibutton.grid(row=2,column=0,pady=10)

adpaybutton=ttk.Button(leftFrame,text='Add Pay cost',cursor='hand2',width=20,command=editpay)

adpaybutton.grid(row=3,column=0,pady=10)

parkbutton=ttk.Button(leftFrame,text='Edit Parking',cursor='hand2',width=20,command=parking)

parkbutton.grid(row=4,column=0,pady=10)

vunpaybutton=ttk.Button(leftFrame,text='View Unpaid bills',cursor='hand2',width=20,command=unpaybill)

vunpaybutton.grid(row=5,column=0,pady=10)

vallresbutton=ttk.Button(leftFrame,text='View All Residents',cursor='hand2',width=20,command=viewall)

vallresbutton.grid(row=6,column=0,pady=10)

searchbutton=ttk.Button(leftFrame,text='Search Resident',cursor='hand2',width=20,command=searchres)

searchbutton.grid(row=7,column=0,pady=10)

deleresbutton=ttk.Button(leftFrame,text='Delete Resident',cursor='hand2',width=20,command=deleteres)

deleresbutton.grid(row=8,column=0,pady=10)

rightFrame = Frame(root, bg='white')

rightFrame.place(x=350, y=80, width=1100, height=650)

scrollbarx = Scrollbar(rightFrame,orient=HORIZONTAL)

scrollbary = Scrollbar(rightFrame,orient=VERTICAL)

resTable=ttk.Treeview(rightFrame,columns=('Resident\_id','Name','Apartment\_id','Age','Contact.no'),

xscrollcommand=scrollbarx.set,yscrollcommand=scrollbary.set)

scrollbarx.config(command=resTable.xview)

scrollbary.config(command=resTable.yview)

scrollbarx.pack(side=BOTTOM,fill=X)

scrollbary.pack(side=RIGHT,fill=Y)

resTable.pack(fill=BOTH,expand=1)

resTable.heading('Resident\_id',text='Resident\_id')

resTable.heading('Name',text='Name')

resTable.heading('Apartment\_id',text='Apartment\_id')

resTable.heading('Age',text='Age')

resTable.heading('Contact.no',text='Contact.no')

resTable.column('Resident\_id',width=50,anchor=CENTER)

resTable.column('Name',width=300,anchor=CENTER)

resTable.column('Apartment\_id',width=200,anchor=CENTER)

resTable.column('Age',width=100,anchor=CENTER)

resTable.column('Contact.no',width=200,anchor=CENTER)

style=ttk.Style()

style.configure('Treeview',rowheight=40,font=('arial',12,'bold'),foreground='black',background='white'

,fieldbackground='white')

style.configure('Treeview.Heading',font=('arial',14,'bold'))

resTable.config(show='headings')

root.mainloop()

**USER LOGIN-PAGE:**

from tkinter import \*

from tkinter import messagebox

from PIL import ImageTk

import mysql.connector

mydb=mysql.connector.connect(

host="localhost",

username="root",

passwd="ram9486",

database="apart"

)

if mydb.is\_connected():

print("login connection established")

mycursor=mydb.cursor()

def login():

username = unEntry.get()

password = pwEntry.get()

query = f"SELECT \* FROM residents WHERE name = '{username}' AND resident\_id = '{password}'"

mycursor.execute(query)

row = mycursor.fetchone()

if row is not None:

messagebox.showinfo("Login", "Login successful!")

window.destroy()

import profile\_page

def uss():

return (username)

def pss():

return (password)

uss(username)

pss(password)

else:

messagebox.showerror("Login", "Invalid username or password.")

window=Tk()

window.geometry('1525x760+0+0')

window.title('Login System of Apartment Management System')

window.resizable(0,0)

backImage=ImageTk.PhotoImage(file='lpg.webp')

label1=Label(window,image=backImage)

label1.pack()

loginFrame=Frame(window,bg='white')

loginFrame.place(x=400,y=150)

logoImage=PhotoImage(file='alogo.png')

label2=Label(loginFrame,image=logoImage)

label2.grid(row=0,column=1,columnspan=2,pady=10)

unImage=PhotoImage(file='user.png')

unLabel=Label(loginFrame,image=unImage,text='Username',compound=LEFT,font=('times new roman',20,'bold'),bg='white')

unLabel.grid(row=1,column=1,pady=10,padx=20)

unEntry=Entry(loginFrame,font=('times new roman',20,'bold'),bd=5,fg='royalblue')

unEntry.grid(row=1,column=2,pady=10,padx=20)

pwImage = PhotoImage(file='pass.png')

pwLabel = Label(loginFrame, image=pwImage, text='Password', compound=LEFT, font=('times new roman', 20, 'bold'),

bg='white')

pwLabel.grid(row=2, column=1, pady=10, padx=20)

pwEntry = Entry(loginFrame, font=('times new roman', 20, 'bold'), bd=5, fg='royalblue')

pwEntry.grid(row=2, column=2, pady=10, padx=20)

loginButton=Button(loginFrame,text='Login',font=('times new roman', 14, 'bold'),width=15,fg='white'

,bg='cornflowerblue',activebackground='cornflowerblue',activeforeground='white',cursor='hand2'

,command=login)

loginButton.grid(row=3,column=2,pady=10)

window.mainloop()

**USER PROFILE:**

from tkinter import \*

import time

import ttkthemes

from tkinter import ttk, messagebox

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

username="root",

passwd="ram9486",

database="apart"

)

if mydb.is\_connected():

print("connection established")

mycursor = mydb.cursor()

# functions

def profile():

rightFrame = Frame(root, bg='white')

rightFrame.place(x=350, y=80, width=1200, height=1000)

query = "SELECT \* FROM residents WHERE resident\_id = %s"

myda=(res\_id,)

mycursor.execute(query,myda)

data = mycursor.fetchone()

namelabel = Label(rightFrame, text="Name : " + data[1], font=('arial', 18), bg='white')

namelabel.place(x=200, y=50)

idlabel = Label(rightFrame, text="Resident id : " + str(data[0]), font=('arial', 18), bg='white')

idlabel.place(x=200, y=100)

agelabel = Label(rightFrame, text="Age : " + str(data[3]), font=('arial', 18), bg='white')

agelabel.place(x=200, y=150)

apalabel = Label(rightFrame, text="Apartment id : " + str(data[2]), font=('arial', 18), bg='white')

apalabel.place(x=200, y=200)

pholabel = Label(rightFrame, text="Phone : " + str(data[4]), font=('arial', 18), bg='white')

pholabel.place(x=200, y=250)

def pay():

def setpay():

query="update payment set bill\_paid\_date=current\_date where Apartment\_id=%s"

mycursor.execute(query,(data[5],))

mydb.commit

messagebox.showinfo('Success','Paid successful')

pay()

rightFrame = Frame(root, bg='white')

rightFrame.place(x=350, y=80, width=1200, height=1000)

mycursor.execute("select pay\_id,bill\_arrived\_date,bill\_paid\_date,total\_cost,mode\_of\_payment,payment.Apartment\_id from residents,payment where residents.Apartment\_id=payment.Apartment\_id and Resident\_id= %s",

(res\_id,))

data = mycursor.fetchone()

if len(data) == 0:

messagebox.showinfo("No Bills", "Sorry!! You don't have any bills arrived yet!")

return

totlabel = Label(rightFrame, text="Bill id : " + str(data[0]), font=('arial', 18), bg='white')

totlabel.place(x=200, y=100)

pidlabel = Label(rightFrame, text="Bill Arrive Date : " + str(data[1]), font=('arial', 18), bg='white')

pidlabel.place(x=200, y=150)

aparlabel = Label(rightFrame, text="Bill Paid Date : " + str(data[2]), font=('arial', 18), bg='white')

aparlabel.place(x=200, y=200)

tcostlabel = Label(rightFrame, text="Total Cost : " + str(data[3]), font=('arial', 18), bg='white')

tcostlabel.place(x=200, y=250)

modelabel = Label(rightFrame, text="Payment Mode : " + str(data[4]), font=('arial', 18), bg='white')

modelabel.place(x=200, y=300)

aplabel = Label(rightFrame, text="Apartment\_Id : " + str(data[5]), font=('arial', 18), bg='white')

aplabel.place(x=200, y=350)

if(data[2] is None):

paycost\_button = ttk.Button(rightFrame, text="PAY", command=setpay)

paycost\_button.place(x=300,y=500)

def park():

rightFrame = Frame(root, bg='white')

rightFrame.place(x=350, y=80, width=1200, height=1000)

query='select \* from parking\_lot where Apartment\_id in(select parking\_lot.Apartment\_id from residents inner join parking\_lot on residents.Apartment\_id=parking\_lot.Apartment\_id and Resident\_id=%s)'

mycursor.execute(query,(res\_id,))

data=mycursor.fetchone()

if len(data) == 0:

messagebox.showinfo("No Lifts", "Sorry!! Lifts not available!")

return

if data[1]==1:

co='Vehicle Parked'

else:

co='Vehicle Not Parked'

pidlabel = Label(rightFrame, text="Working Condition : " +co, font=('arial', 18), bg='white')

pidlabel.place(x=200, y=150)

aparlabel = Label(rightFrame, text="Parking Id : " + str(data[0]), font=('arial', 18), bg='white')

aparlabel.place(x=200, y=200)

tcostlabel = Label(rightFrame, text="Apartment id : " + str(data[2]), font=('arial', 18), bg='white')

tcostlabel.place(x=200, y=250)

def lift():

rightFrame = Frame(root, bg='white')

rightFrame.place(x=350, y=80, width=1200, height=1000)

query='select \* from lifts where Block\_id in(select Block\_id from residents inner join apartment on residents.Apartment\_id=apartment.Apartment\_id and Resident\_id=%s)'

mycursor.execute(query,(res\_id,))

data = mycursor.fetchone()

if len(data) == 0:

messagebox.showinfo("No Lifts", "Sorry!! Lifts not available!")

return

totlabel = Label(rightFrame, text="Lift id : " + str(data[0]), font=('arial', 18), bg='white')

totlabel.place(x=200, y=100)

if data[1]==1:

b='Good'

else:

b='Under repair'

pidlabel = Label(rightFrame, text="Working Condition : " + b, font=('arial', 18), bg='white')

pidlabel.place(x=200, y=150)

aparlabel = Label(rightFrame, text="Position : " + str(data[2]), font=('arial', 18), bg='white')

aparlabel.place(x=200, y=200)

tcostlabel = Label(rightFrame, text="Block id : " + str(data[3]), font=('arial', 18), bg='white')

tcostlabel.place(x=200, y=250)

def exit():

result = messagebox.askyesno('CONFIRM', 'DO YOU WANT TO LOG-OUT')

if result:

root.destroy()

import login\_page

else:

pass

def clock():

date = time.strftime('%d/%m/%Y')

currenttime = time.strftime('%H:%M:%S')

datetimeLabel.config(text=f' Date: {date}\nTime: {currenttime}')

datetimeLabel.after(1000, clock)

def idd():

def submit():

global res\_id

res\_id = res\_id\_entry.get()

idwin.destroy()

idwin = Toplevel()

idwin.grab\_set()

idwin.resizable(False, False)

res\_id\_label = Label(idwin, text="Enter the Resident id:", font=('times new roman', 20, 'bold'))

res\_id\_label.grid(row=0, column=0, padx=30, pady=15, stick=W)

res\_id\_entry = Entry(idwin, font=('times new roman', 20, 'bold'), bd=5, fg='dark orange', width=24)

res\_id\_entry.grid(row=0, column=1, pady=15, padx=10)

submit\_button = ttk.Button(idwin, text="Submit", command=submit)

submit\_button.grid(row=6, columnspan=2, pady=15)

count = 0

text = ''

def slider():

global text, count

if count == len(a):

count = 0

text = ''

text = text + a[count] # a

sliderLabel.config(text=text)

count += 1

sliderLabel.after(300, slider)

# GUI

root = ttkthemes.ThemedTk()

root.get\_themes()

root.set\_theme('radiance')

root.geometry('1474x750+0+0')

root.resizable(0, 0)

root.title('Apartment Management System')

datetimeLabel = Label(root, font=('times new roman', 18, 'bold'))

datetimeLabel.place(x=5, y=5)

clock()

a = 'Apartment Management System'

sliderLabel = Label(root, text=a, font=('arial', 28, 'italic bold'), width=50)

sliderLabel.place(x=200, y=0)

slider()

connectButton = ttk.Button(root, text='LOG OUT', command=exit)

connectButton.place(x=1300, y=0)

leftFrame = Frame(root)

leftFrame.place(x=50, y=80, width=300, height=600)

logoimage = PhotoImage(file='resident.png')

logoLabel = Label(leftFrame, image=logoimage)

logoLabel.grid(row=0, column=0)

profilebutton = ttk.Button(leftFrame, text='PROFILE', cursor='hand2', width=20,command=profile)

profilebutton.grid(row=1, column=0, pady=20)

paybutton = ttk.Button(leftFrame, text='PAYMENT', cursor='hand2', width=20, command=pay)

paybutton.grid(row=3, column=0, pady=20)

parkbutton = ttk.Button(leftFrame, text='PARKING', cursor='hand2', width=20, command=park)

parkbutton.grid(row=5, column=0, pady=20)

liftbutton = ttk.Button(leftFrame, text='LIFT', cursor='hand2', width=20, command=lift)

liftbutton.grid(row=7, column=0, pady=20)

iddbutton = ttk.Button(leftFrame, text='CONNECT ID', cursor='hand2', width=20, command=idd)

iddbutton.grid(row=8, column=0, pady=20)

rightFrame = Frame(root, bg='white')

rightFrame.place(x=350, y=80, width=1200, height=1000)

tlabel = Label(rightFrame, text="WELCOME TO RESIDENT PAGE", font=('times new roman', 25, 'bold'), bg='white',

fg='dark orange')

tlabel.place(x=300, y=250)

root.mainloop()

# SCREENSHOTS:

# A computer screen shot of a computer screen Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

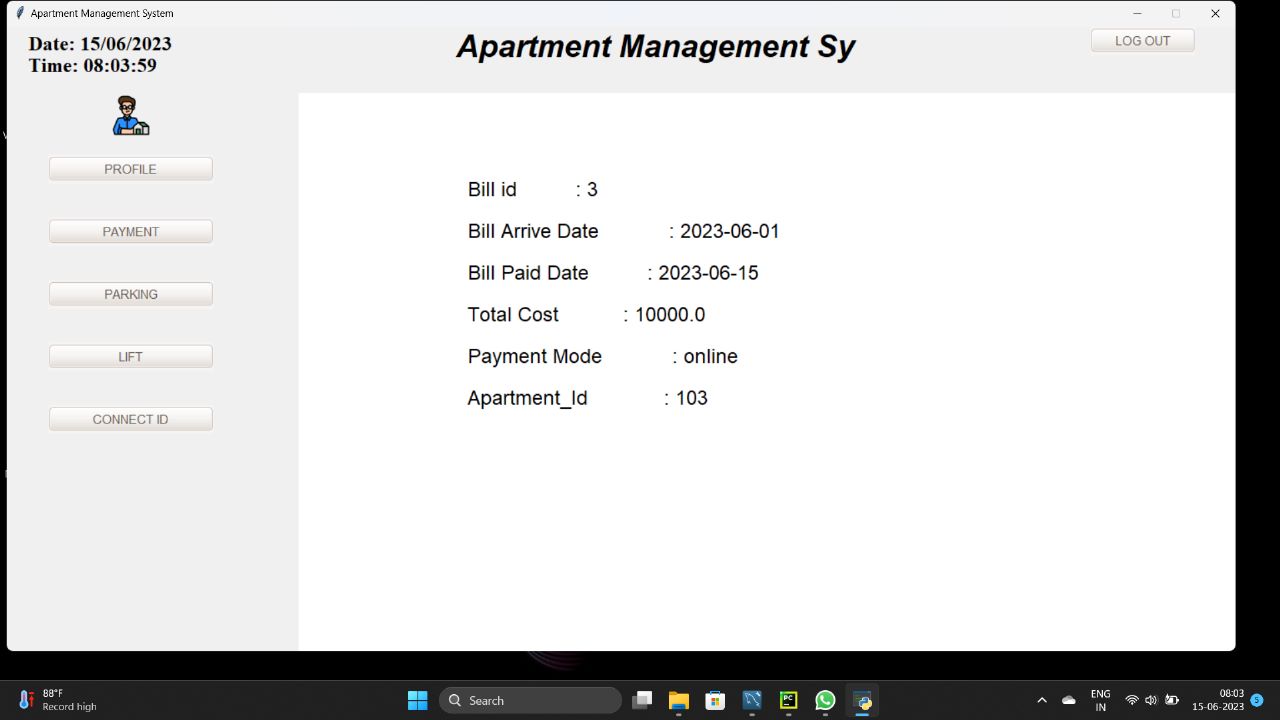
Description automatically generatedA screenshot of a computer

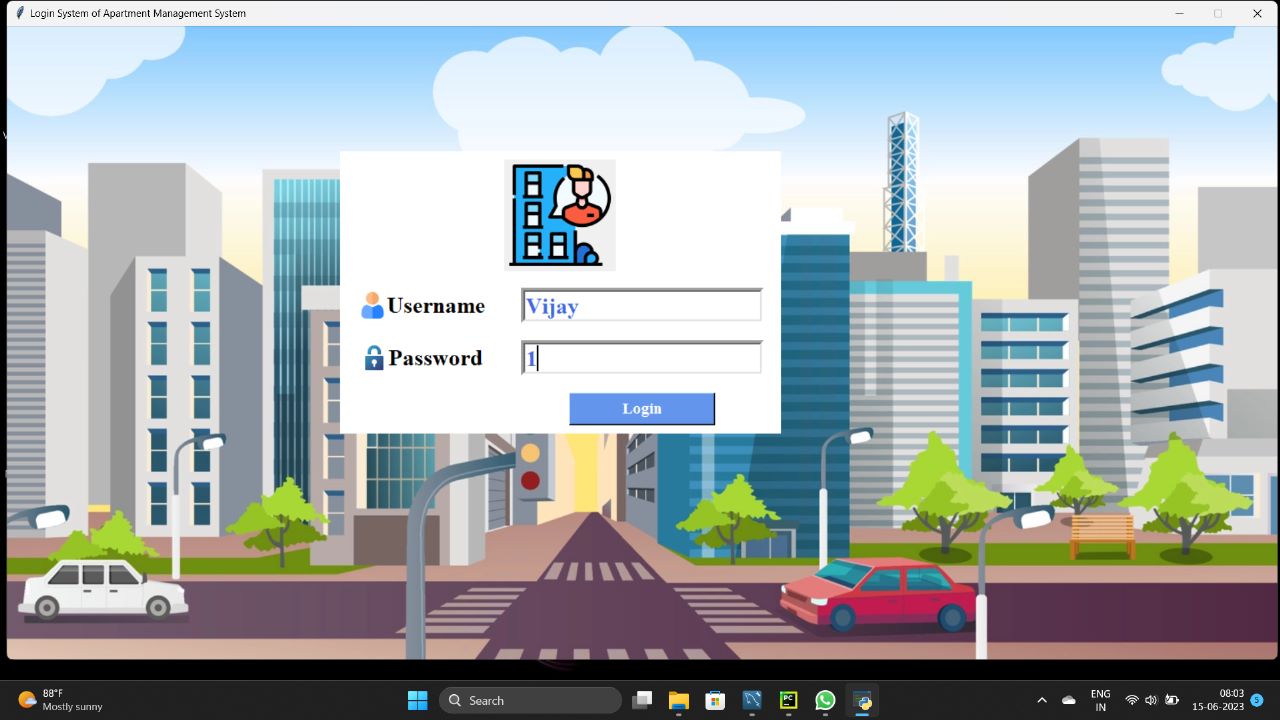
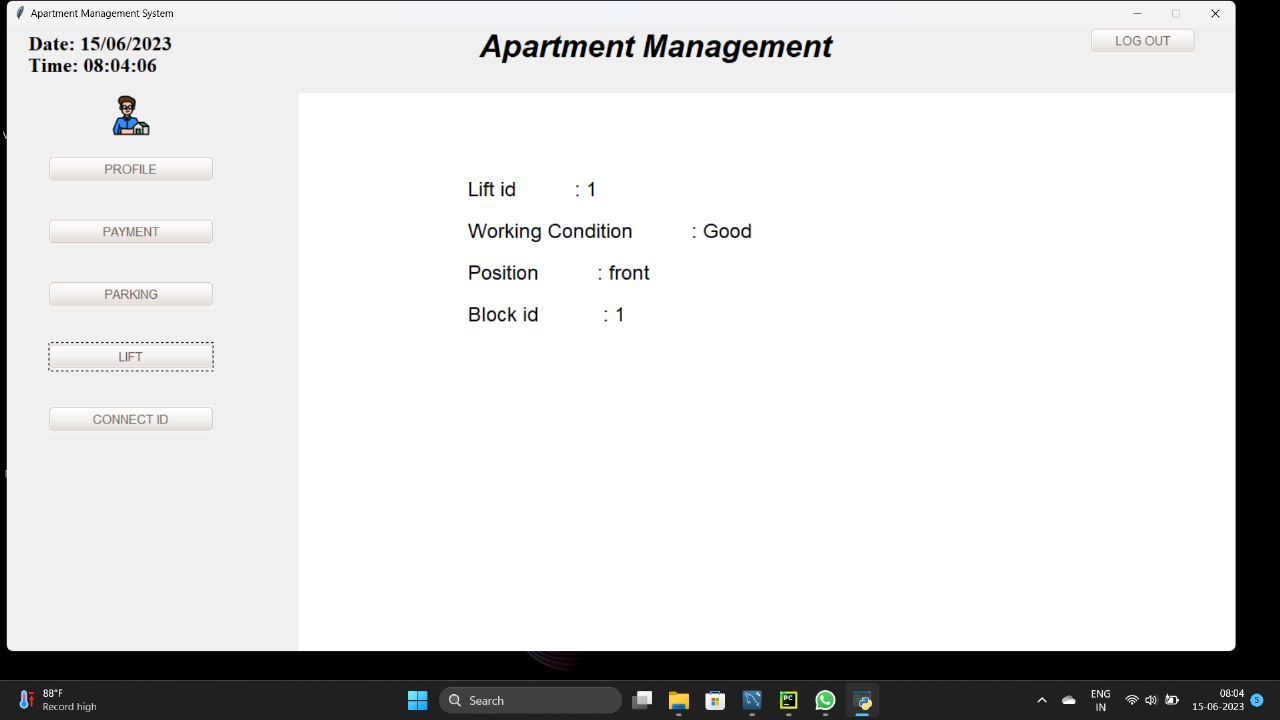
Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generatedA screenshot of a computer

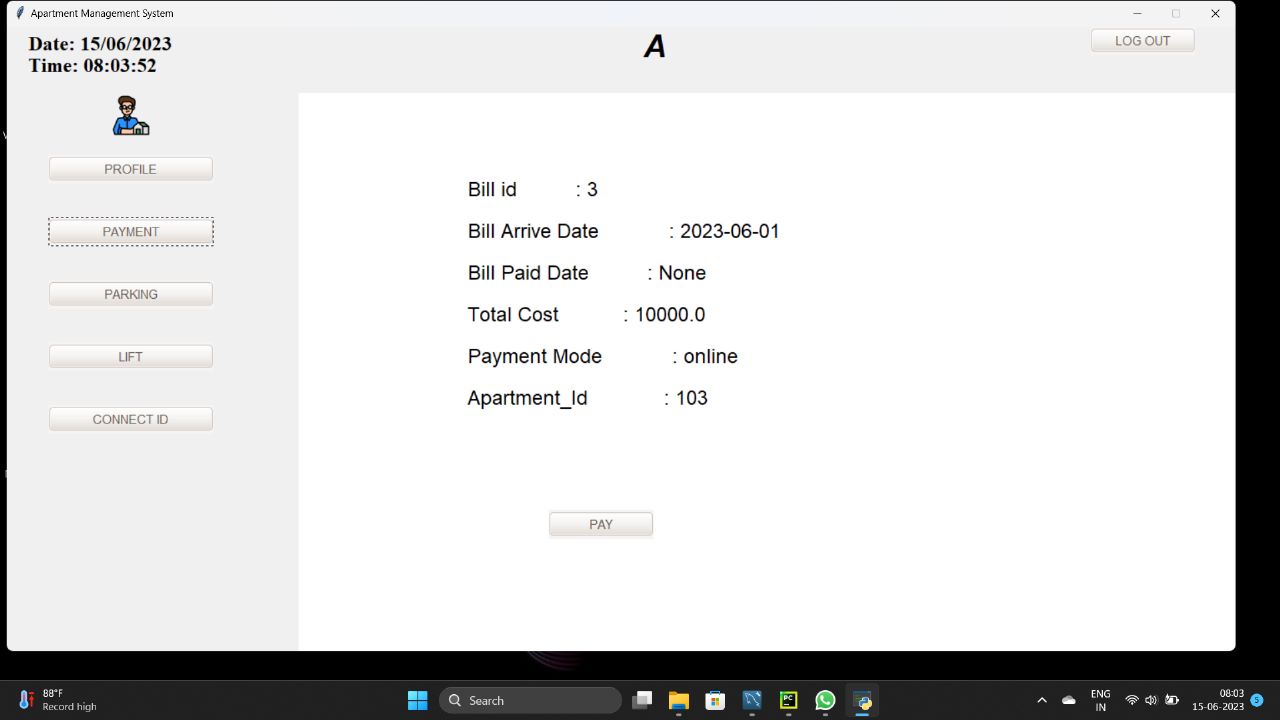
Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidenceA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

# CONCLUSION:

In our project Apartment Management system, we have stored all the information about the Apartments sold and the residents buying apartments and even status of lifts, maintenance bills payment etc.

This data base is helpful for the applications which facilitate residents to view the visitors log and check the details of maintenance receipts and their status from their place itself. It avoids inconvenience of asking the management team to provide bill details.

We had considered the most important requirements only; many more features and details can be added to our project in order to obtain even more userfriendly applications.

These applications are already in progress and in future they can be upgraded and may become part of amazing technology.

# REFERENCES:

SQL INNER JOIN:

<https://www.mysqltutorial.org/mysql-inner-join.aspx>

SQL SELECT:

<https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx>

SQL UPDATE:

<https://www.mysqltutorial.org/mysql-update-data.aspx>

SQL INSERT:

<https://www.mysqltutorial.org/mysql-insert-statement.aspx>