APARTMENT MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

NIDARSHANA K S	**220701185**
NEELA A	**220701184**

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2023-2024

BONAFIDE CERTIFICATE

Certified that this project report "**APARTMENT MANAGEMENT SYSTEM**" is the bonafide work of "**NEELA A (220701184), NIDARSHANA K S (220701185)** who carried out the project under my supervision.

Submitted for the Practical Examination held on ____________________

SIGNATURE	**SIGNATURE**
Dr.R.SABITHA	**Ms.D.KALPANA**
Professor and II Year Academic Head	**Assistant Professor (SG),**
Computer Science and Engineering,	**Computer Science and Engineering,**
Rajalakshmi Engineering College	**Rajalakshmi Engineering College**
(Autonomous),	**(Autonomous),**
Thandalam, Chennai - 602 105	**Thandalam, Chennai - 602 105**

ABSTRACT

The Apartment Management System is a user-friendly application which enables efficient management of residents information by allowing users to insert, update, delete, and view records seamlessly. The intuitive interface facilitates easy data entry and retrieval, ensuring effective administration of apartment complexes.

Overall, the Apartment Management System offers a comprehensive solution for managing apartment resident data, combining the simplicityof Tkinter for the GUI and the robustness of MySQL for database operations. This integration facilitates effective data management and enhances user experience in apartment administration tasks.

TABLE OF CONTENTS

CHAPTER 1

1.1 INTRODUCTION

The Apartment Management System provides essential functionalities to handle various aspects of resident information management. Users can effortlessly add new resident details, update existing records, delete outdated or incorrect information, and view all resident data in an organized table format. The intuitive interface ensures that users can navigate the system with ease, making data entry and retrieval straightforward and efficient.

1.2 EXISTING SYSTEM

The apartment management system that focuses solely on tenant information and apartment details includes

1.SimplifyEm (Tenant and Apartment Management)

Overview: SimplifyEm offers a basic version that can be tailored to focus on managing tenant and apartment details without including maintenance requests, lease tracking, accounting, or document management features.

Key Features:

Tenant Management: Store and manage tenant contact information and household members.

Apartment Management: Track apartment details such as block number and door number.

Communication Tools: Basic tools for sending messages and notifications to tenants.

Functions:

Add: New tenant information and apartment details.

Delete: Tenant records and apartment details.

Update: Tenant information and apartment details.
Search: Tenant records and apartment details.

2. Propertyware (Basic Tenant and Apartment Management):

Overview: Propertyware provides flexible property management solutions that can be customized to include only basic tenant and apartment management functionalities.

Key Features:
Tenant Management: Manage tenant details and household members.
Apartment Management: Store apartment information such as block number and door number
Communication Tools: Basic communication tools to interact with tenants.

Functions:
Add: Tenant details and apartment information.
Delete: Tenant records and apartment information.
Update: Tenant information and apartment details.
Search: Tenant records and apartment information.

PROPOSED SYSTEM

Objective

The proposed apartment management system is designed to efficiently manage tenant and apartment information without including lease tracking, accounting, document management, or maintenance requests. The primary functions include adding, deleting, updating, and searching tenant and apartment details.

Key Features:
Tenant Management:
    Store and manage tenant contact information and household members.
    Update tenant details as needed.
Apartment Management:
    Track apartment details such as block number and door number.
    Update apartment information as needed.

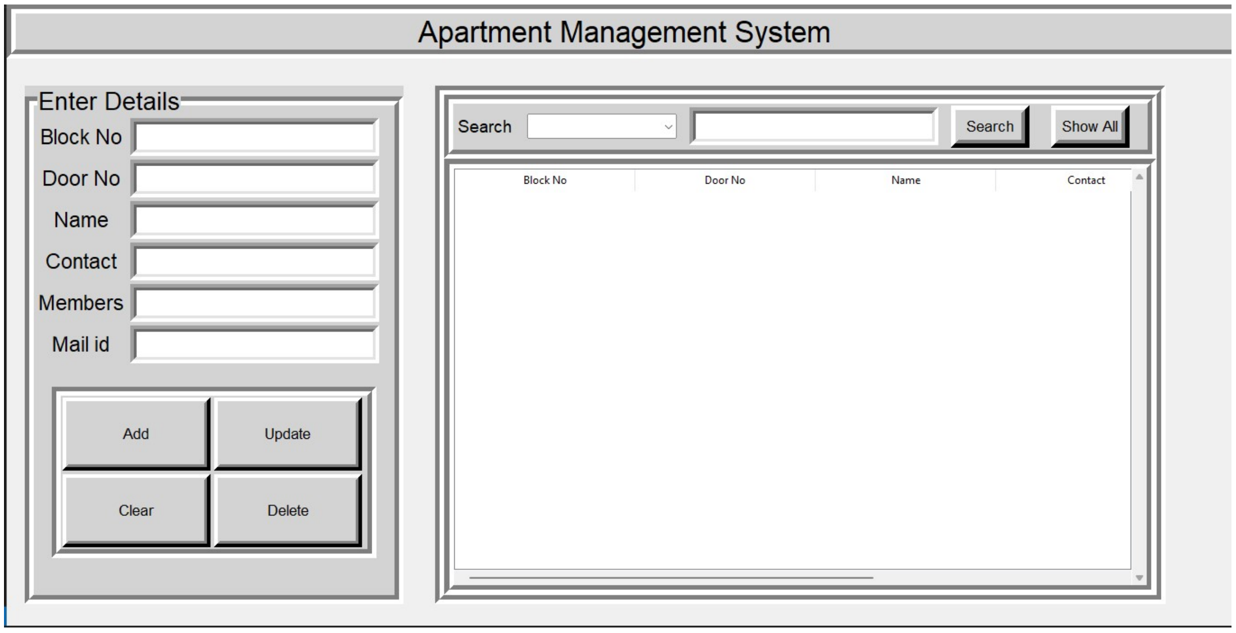
Search Functionality:
Search for tenant records and apartment details using various criteria (e.g., tenant name, block number, door number).

1.3 OBJECTIVES

The primary objectives of the Apartment Management System are to streamline the management of resident data through an efficient and user-friendly interface, thereby reducing the time and effort required for data entry and retrieval. The system aims to enhance data accuracy and integrity by incorporating robust error handling and validation mechanisms, ensuring that all information is reliable. It facilitates easy access to and updates of resident information, allowing users to add new records, update existing ones, and delete outdated data, thus maintaining a current database.

1.4 MODULES

Resident information module



CHAPTER-2

2.1 SOFTWARE DESCRIPTION

Visual studio Code:

Visual Studio Code serves as a comprehensive development environmentthat supports every stage of the application lifecycle from design and development to testing and deployment.

2.2 LANGUAGES

1. Python:

It provides the necessary tools to create a feature-rich, scalable, andsecure system.

2. Tkinter:

Tkinter is a Python library used for creating graphical user interfaces (GUIs). In the context of an Apartment Management System (AMS), Tkinter provides a simple way to create interactive elements such as buttons, text fields, and tables

3. MySQL:

MySQL serves as the backend database where all the data related to the apartments, residents, and their related operations are stored, managed, and retrieved securely.

CHAPTER-3

REQUIREMENT AND ANALYSIS

3.1 REQUIREMENT SPECIFICATION:

The Apartment Management System must enable users to efficiently manage various store operations. Users should be able to add, view, delete ,update and store details such as Block number,Door number, Name,Contact,Number of people residing in the house.User should be able to search the details of resident by specifying any value in the database.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS:

Hardware Requirements :

- Processor: 1 GHz or faster processor

- RAM: 2 GB or more

- Storage: At least 500 MB of available disk space

- Display: Minimum resolution of 1024x768

- Input Devices: Keyboard and mouse

Software Requirements:

- Operating System: Windows 7 or later, macOS, or Linux

- Python: Version 3.6 or higher

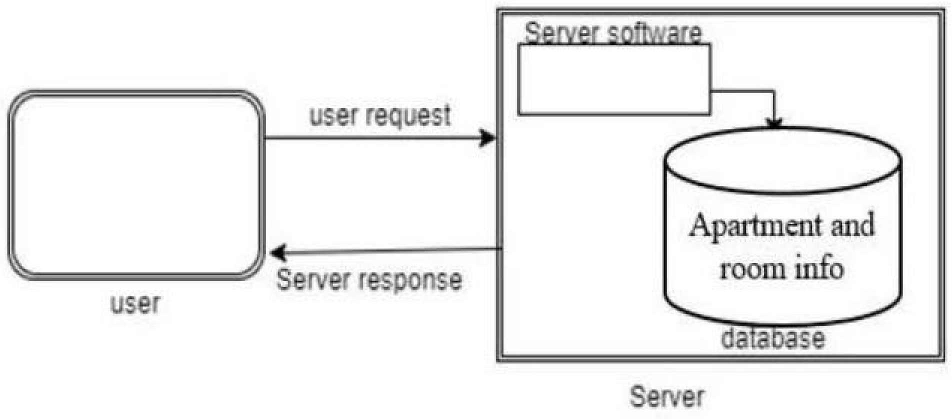
- Mysql: Version 3 or higherPython

  Libraries:

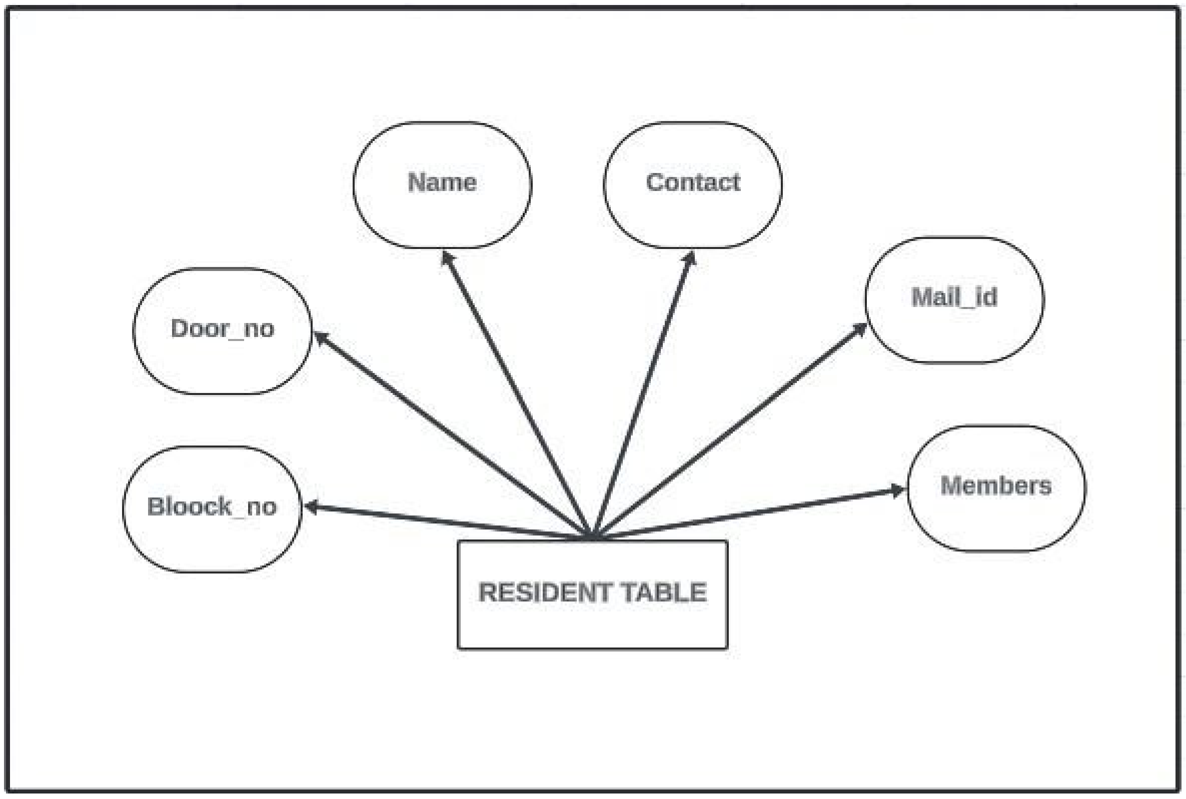
  - ‘tkinter’ for GUI development (included with Python)

  - ‘mysql’ for database management (included withPython)

3.1 ARCHITECTURE DIAGRAM:



3.2 ER DIAGRAM



CHAPTER 4
PROGRAM CODE

```
import tkinter as tk
from tkinter import messageboxfrom tkinter
import ttk
from typing import Selffrom
weakref import ref
import mysql.connector.connectionfrom
requests import delete

win=tk.Tk() win.geometry('13750x700+0+0')
win.title('Apartment  Management  System')
title_label=tk.Label(win,text='Apartment Management
System',font=("Arial",25),border=10,relief=tk.GROOVE,bg='lightgrey')
title_label.pack(side=tk.TOP,fil=tk.X)

 detail_frame=tk.LabelFrame(win,text="Enter
Details",font=('arial',20),bg='lightgrey',bd=10,relief=tk.GROOVE)
detail_frame.place(x=20,y=90,width=420,height=575)
data_frame=tk.LabelFrame(win,bd=10,bg='lightgrey',relief=tk.GROOV E)
data_frame.place(x=475,y=90,width=810,height=575)


blockno=tk.StringVar() doorno=tk.StringVar()
name_id=tk.StringVar()
```

```
contactno=tk.StringVar() members=tk.StringVar()
mailid=tk.StringVar() search_by=tk.StringVar()
search_val=tk.StringVar()



block_no=tk.Label(detail_frame,text='Block No',font=('arial',17),bg='lightgrey')
block_no.grid(row=0,column=0,padx=2,pady=2)
block_no_ent=tk.Entry(detail_frame,bd=7,font=('arial',17),textvariable= blockno)
block_no_ent.grid(row=0,column=1,padx=2,pady=2)

door_no=tk.Label(detail_frame,text='Door No',font=('arial',17),bg='lightgrey')
door_no.grid(row=1,column=0,padx=2,pady=2)
door_no_ent=tk.Entry(detail_frame,bd=7,font=('arial',17),textvariable=d oorno)
door_no_ent.grid(row=1,column=1,padx=2,pady=2)

name=tk.Label(detail_frame,text='Name',font=('arial',17),bg='lightgrey')
name.grid(row=2,column=0,padx=2,pady=2)
name_ent=tk.Entry(detail_frame,bd=7,font=('arial',17),textvariable=nam e_id)
name_ent.grid(row=2,column=1,padx=2,pady=2)
contact_no=tk.Label(detail_frame,text='Contact',font=('arial',17),bg='lig htgrey')
contact_no.grid(row=3,column=0,padx=2,pady=2)
```

```
contact_no_ent=tk.Entry(detail_frame,bd=7,font=('arial',17),textvariable
=contactno) contact_no_ent.grid(row=3,column=1,padx=2,pady=2)

members_no=tk.Label(detail_frame,text='Members',font=('arial',17),bg= 'lightgrey')
members_no.grid(row=4,column=0,padx=2,pady=2)
members_no_ent=tk.Entry(detail_frame,bd=7,font=('arial',17),textvariab le=members)
members_no_ent.grid(row=4,column=1,padx=2,pady=2)

mail_id=tk.Label(detail_frame,text='Mail id',font=('arial',17),bg='lightgrey')
mail_id.grid(row=5,column=0,padx=2,pady=2)
mail_id_ent=tk.Entry(detail_frame,bd=7,font=('arial',17),textvariable=m ailid)
mail_id_ent.grid(row=5,column=1,padx=2,pady=2) def
fetch_apartment():
con=mysql.connector.connect(host='localhost',user='root',password='Ns
@2005@',database='ams')cur=con.cursor()
    cur.execute("SELECT * FROM apartment")rows=cur.fetchall()
    if len(rows)!=0: resident_table.delete(*resident_table.get_children())
        for row in rows:
            resident_table.insert('',tk.END,values=row) con.commit()
    con.close()



def add_func():
    if blockno.get() =="" or doorno.get()=="" : messagebox.showerror("error!","please fill all the
        details")
    else:

con=mysql.connector.connect(host='localhost',user='root',password='nee la@2004',database='ams')
        cur=con.cursor() cur.execute("insert into
        apartment
values(%s,%s,%s,%s,%s,%s)",(blockno.get(),doorno.get(),name_id.get()
,contactno.get(),members.get(),mailid.get())) con.commit()
        con.close()
        fetch_apartment()
def get_cursor(event):
    cursor_row = resident_table.focus()
    content=resident_table.item(cursor_row) row =
    content['values'] blockno.set(row[0])
    doorno.set(row[1]) name_id.set(row[2])
    contactno.set(row[3])members.set(row[4])
    mailid.set(row[5])
```

```
def clear(): blockno.set("")
    doorno.set("")
    name_id.set("")
    contactno.set("")
    members.set("")
    mailid.set("")



def update_func():

con=mysql.connector.connect(host='localhost',user='root',password='nee la@2004',database='ams')
    cur=con.cursor() cur.execute("update
    apartment  set
blockno=%s,doorno=%s,name_id=%s,contactno=%s,members=%s where
mailid=%s",(blockno.get(),doorno.get(),name_id.get(),contactno.get(),m
embers.get(),mailid.get()))
    con.commit() con.close()
    fetch_apartment()clear()


def delete():
    selected_item  =  resident_table.selection()if not
    selected_item:
        messagebox.showwarning("Delete Error", "Please select a residentto delete")
        return

    resident_data = resident_table.item(selected_item, 'values')blockno, doorno,
    name_id, contactno, members, mailid =
resident_data



con=mysql.connector.connect(host='localhost',user='root',password='nee la@2004',database='ams')
      if con:
         try:
            cur = con.cursor()
            cur.execute("DELETE FROM apartment WHERE blockno = %sAND doorno = %s
AND name_id = %s AND contactno = %s AND members = %s AND mailid = %s",
                           (blockno, doorno, name_id, contactno, members,
mailid))
            con.commit() fetch_apartment()
            messagebox.showinfo("Success", "Resident deletedsuccessfully")
        except mysql.connector.Error as err: messagebox.showerror("Database Error", f"Error:
            {err}")
        finally:
            con.close()

# Function to search for residentsdef
search_resident():
    search_by_value = search_by.get() search_text_value =
    search_val.get()

    if not search_by_value or not search_text_value: messagebox.showwarning("Search Error",
       "Please select a search
criterion and enter a search term")return


con=mysql.connector.connect(host='localhost',user='root',password='nee la@2004',database='ams')
      if con:
         try:
            cursor = con.cursor()

            # Mapping user-friendly column names to actual databasecolumn names
            column_mapping = { "Block No":
                "blockno","Door No": "doorno",
                "Name": "name_id", "Contact":
                "contactno", "Members":
                "members","Mail id": "mailid"
            }

            # Get the actual column name from the mapping db_column =
            column_mapping.get(search_by_value)

            # If the column name is not found, raise an error
            if not db_column:
                messagebox.showwarning("Search  Error",  "Invalid  searchcriterion selected")
                return

            query = f"SELECT * FROM apartment WHERE {db_column}LIKE %s"
            cursor.execute(query, ('%' + search_text_value + '%',))rows =
            cursor.fetchall()
            resident_table.delete(*resident_table.get_children()) if rows:
                for row in rows:
                    resident_table.insert('', 'end', values=row)
            else:
                messagebox.showinfo("Search Result", "No matching records
found")
        except mysql.connector.Error as err: messagebox.showerror("Database  Error",  f"Error:
            {err}")
        finally:
            con.close()

bt_frame=tk.Frame(detail_frame,bg='lightgrey',bd=10,relief=tk.GROOV E)
bt_frame.place(x=20,y=300,width=360,height=190)

add_btn=tk.Button(bt_frame,bg='lightgrey',text='Add',bd=7,font=('Arial'
,13),width=16,height=3,command=add_func) add_btn.grid(row=0,column=0,padx=2,pady=2)
```

```
update_btn=tk.Button(bt_frame,bg='lightgrey',text='Update',bd=7,font=( 'Arial',13),width=16,
height=3,command=update_func) update_btn.grid(row=0,column=1,padx=2,pady=2)

clear_btn=tk.Button(bt_frame,bg='lightgrey',text='Clear',bd=7,font=('Ari
al',13),width=16,height=3,command=clear) clear_btn.grid(row=1,column=0,padx=2,pady=2)

delete_btn=tk.Button(bt_frame,bg='lightgrey',text='Delete',bd=7,font=('
Arial',13),width=16,height=3,command=delete)
delete_btn.grid(row=1,column=1,padx=2,pady=2)

search_frame=tk.Frame(data_frame,bg='lightgrey',bd=10,relief=tk.GRO OVE)
search_frame.pack(side=tk.TOP,fill=tk.X)

search_label=tk.Label(search_frame,text='Search',bg='lightgrey',font=('a rial',14))
search_label.grid(row=0,column=0,padx=2,pady=2)

search_in=ttk.Combobox(search_frame,font=('arial',14),state='readonly',
textvariable=search_by,width=13)
search_in['values']=("Block No","Door No","Name","Contact","Members","Mail id")
search_in.grid(row=0,column=1,padx=12,pady=2)
search_in1_ent=tk.Entry(search_frame,bd=7,font=('arial',17),textvariabl e=search_val)
search_in1_ent.grid(row=0,column=2,padx=2,pady=2)
```

```
search_btn=tk.Button(search_frame,bg='lightgrey',text='Search',bd=9,fo
nt=('Arial',13),width=7,height=1,command=search_resident)
search_btn.grid(row=0,column=3,padx=12,pady=2)
show_all_btn=tk.Button(search_frame,bg='lightgrey',text='Show
All',bd=9,font=('Arial',13),width=7,height=1,command=fetch_apartmen)
show_all_btn.grid(row=0,column=4,padx=12,pady=2)


db_frame=tk.Frame(data_frame,bg='lightgrey',bd=11,relief=tk.GROOVE)
db_frame.pack(fill=tk.BOTH,expand=True)

y_scroll=tk.Scrollbar(db_frame,orient=tk.VERTICAL)
x_scroll=tk.Scrollbar(db_frame,orient=tk.HORIZONTAL)

resident_table=ttk.Treeview(db_frame,columns=("Block No","Door
No","Name","Contact","Members","Mail
id"),yscrollcommand=y_scroll.set,xscrollcommand=x_scroll.set)
y_scroll.config(command=resident_table.yview) x_scroll.config(command=resident_table.xview)
y_scroll.pack(side=tk.RIGHT,fill=tk.Y) x_scroll.pack(side=tk.BOTTOM,fill=tk.X)

resident_table.heading("Block No",text="Block No")resident_table.heading("Door No",text="Door
No") resident_table.heading("Name",text="Name") resident_table.heading("Contact",text="Contact")
resident_table.heading("Members",text="Members") resident_table.heading("Mail id",text="Mail
id")

resident_table['show']='headings'
```

```
resident_table.pack(fill=tk.BOTH,expand=True)

resident_table.column("Block No",width=200)
resident_table.column("Door No",width=200)
resident_table.column("Name",width=200)
resident_table.column("Contact",width=200)
resident_table.column("Members",width=200)
resident_table.column("Mail id",width=200)

resident_table.pack(fill=tk.BOTH,expand=True) fetch_apartment()
resident_table.bind("<ButtonRelease-1>",get_cursor)

win.mainloop()
```

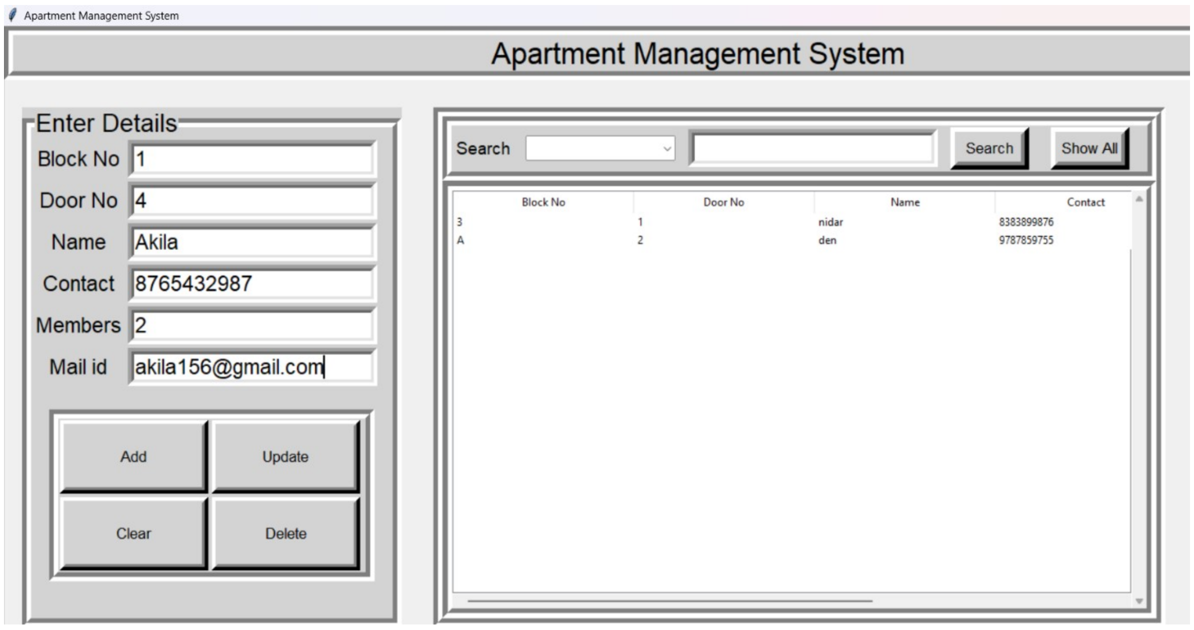
CHAPTER-5

RESULTS

AND DISCUSSION

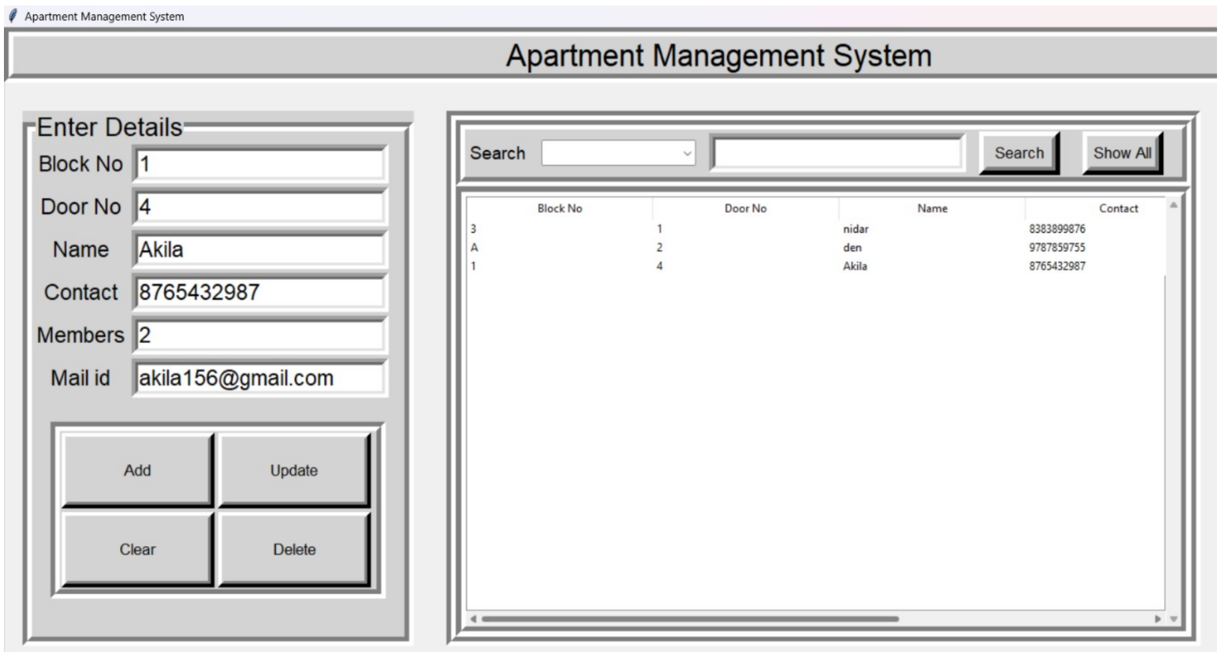
5.1 USER DOCUMENTATION:

ADD FUNCTION:



Block No	1
Door No	4
Name	Akila
Contact	8765432987
Members	2
Mail id	akila156@gmail.com

Block No	Door No	Name	Contact
3	1	nidar	8383899876
A	2	den	9787859755



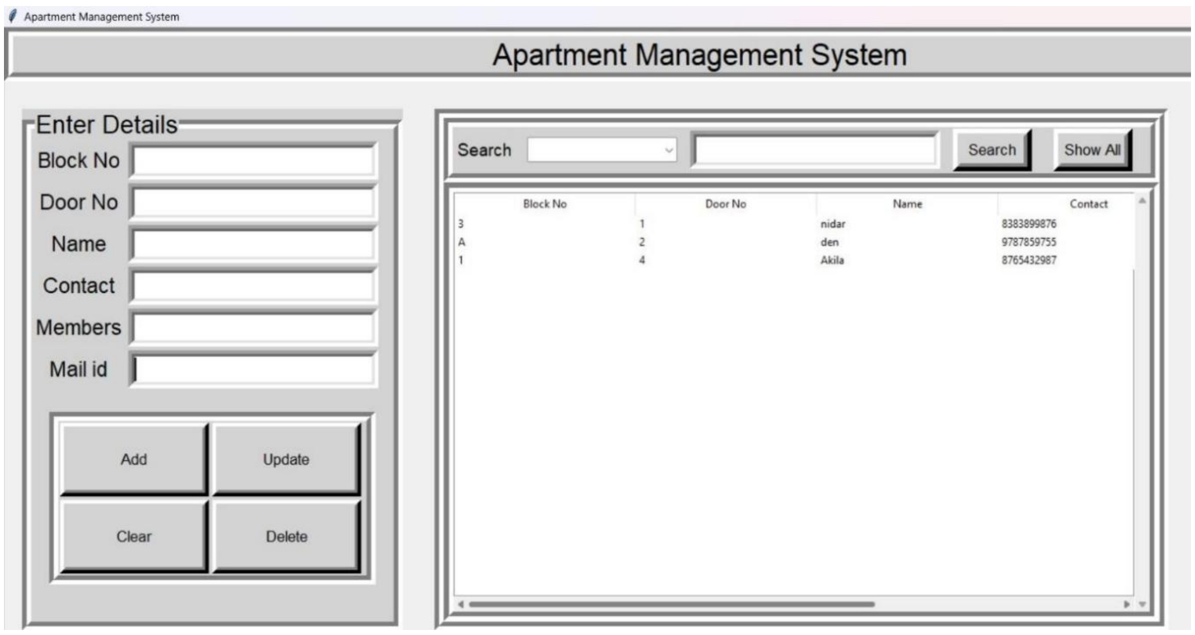
Apartment Management System

Enter Details

Block No	1
Door No	4
Name	Akila
Contact	8765432987
Members	2
Mail id	akila156@gmail.com

Block No	Door No	Name	Contact
3	1	nidar	8383899876
A	2	den	9787859755
1	4	Akila	8765432987

CLEAR FUNCTION:



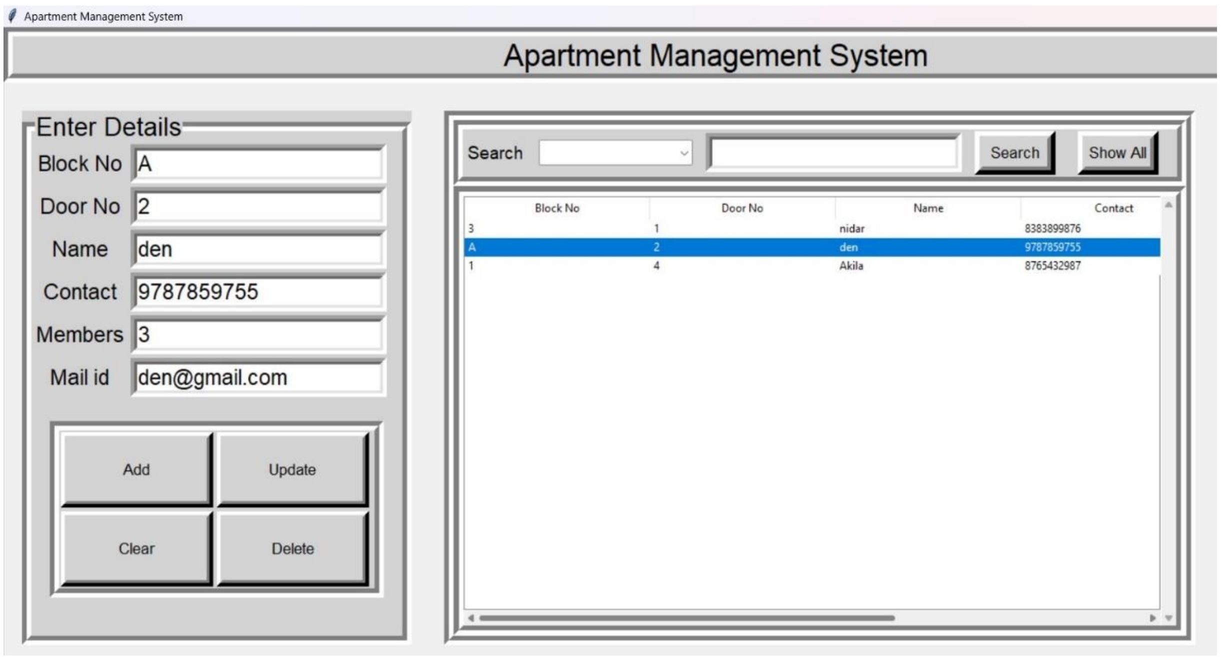
Apartment Management System

Enter Details

Block No	
Door No	
Name	
Contact	
Members	
Mail id	

Block No	Door No	Name	Contact
3	1	nidar	8383899876
A	2	den	9787859755
1	4	Akila	8765432987

UPDATE FUNCTION:

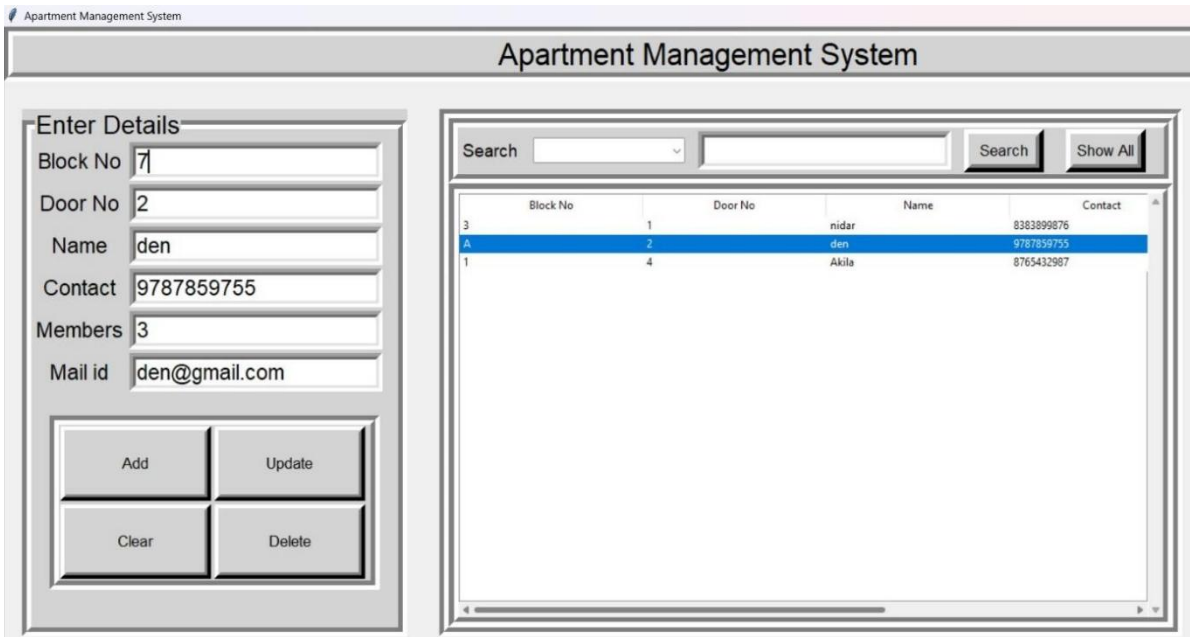

Apartment Management System

Apartment Management System

Enter Details

Block No	A
Door No	2
Name	den
Contact	9787859755
Members	3
Mail id	den@gmail.com

Block No	Door No	Name	Contact
3	1	nidar	8383899876
A	2	den	9787859755
1	4	Akila	8765432987

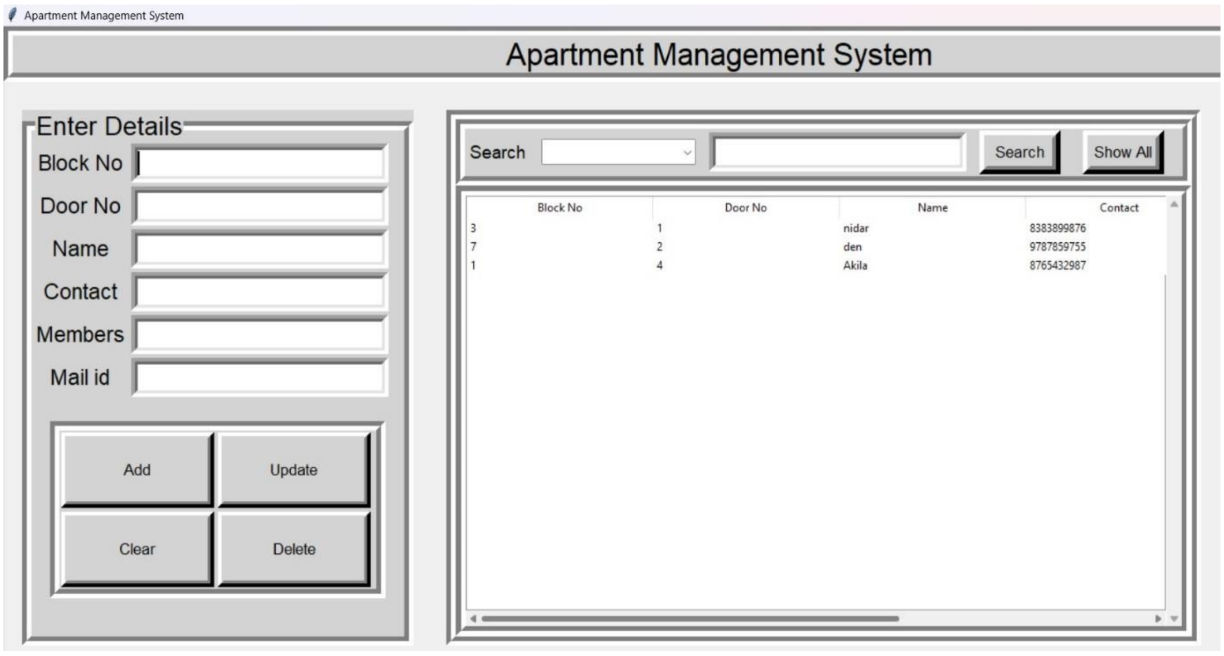

Apartment Management System

Apartment Management System

Enter Details

Block No	7
Door No	2
Name	den
Contact	9787859755
Members	3
Mail id	den@gmail.com

Block No	Door No	Name	Contact
3	1	nidar	8383899876
A	2	den	9787859755
1	4	Akila	8765432987


Apartment Management System

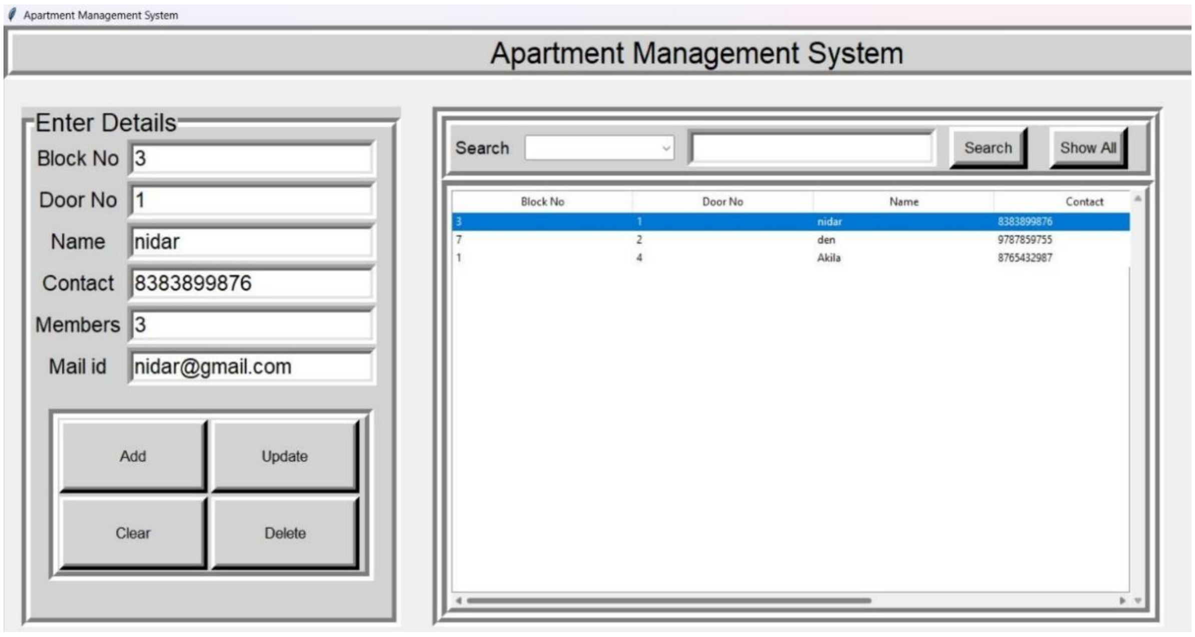
Apartment Management System

Enter Details

Block No	
Door No	
Name	
Contact	
Members	
Mail id	

Block No	Door No	Name	Contact
3	1	nidar	8383899876
7	2	den	9787859755
1	4	Akila	8765432987

DELETE FUNCTION:

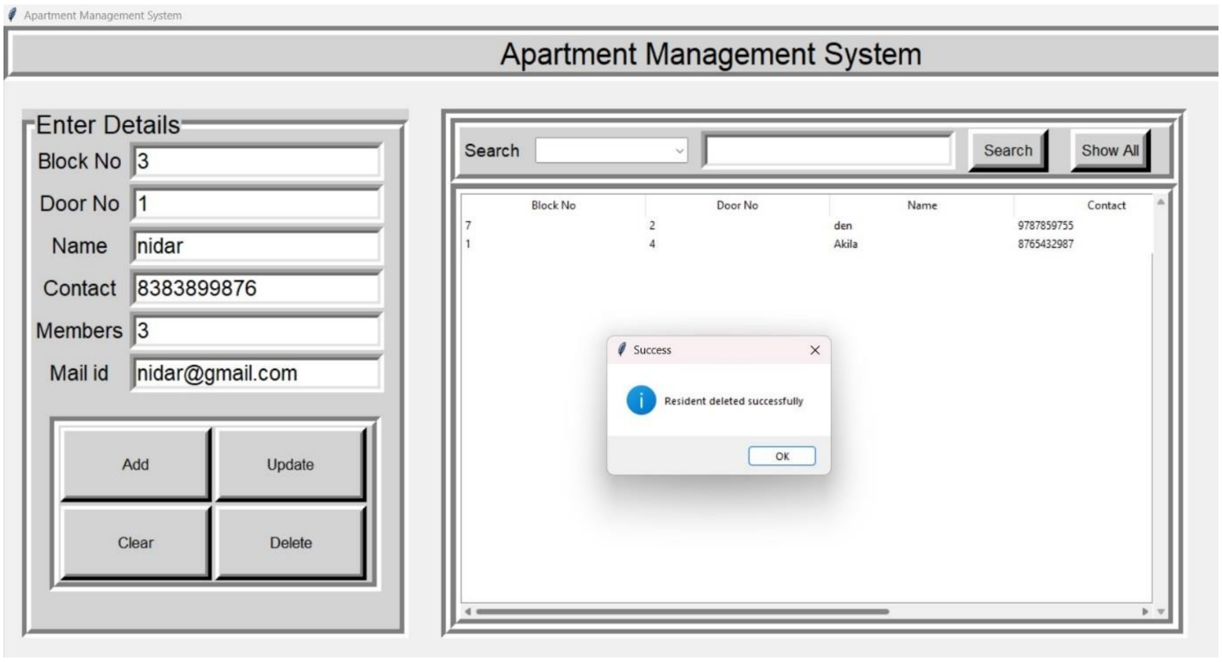

Apartment Management System

Apartment Management System

Enter Details

Block No	3
Door No	1
Name	nidar
Contact	8383899876
Members	3
Mail id	nidar@gmail.com

Block No	Door No	Name	Contact
3	1	nidar	8383899876
7	2	den	9787859755
1	4	Akila	8765432987


Apartment Management System

Enter Details

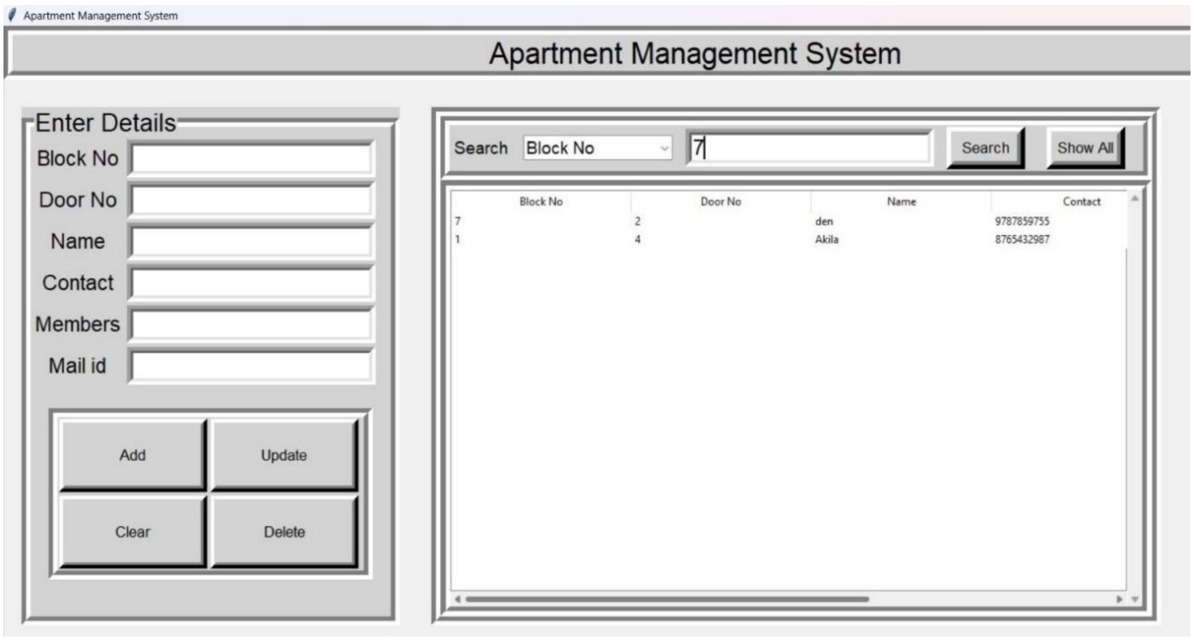
Block No	3
Door No	1
Name	nidar
Contact	8383899876
Members	3
Mail id	nidar@gmail.com

Block No	Door No	Name	Contact
7	2	den	9787859755
1	4	Akila	8765432987

Success

Resident deleted successfully

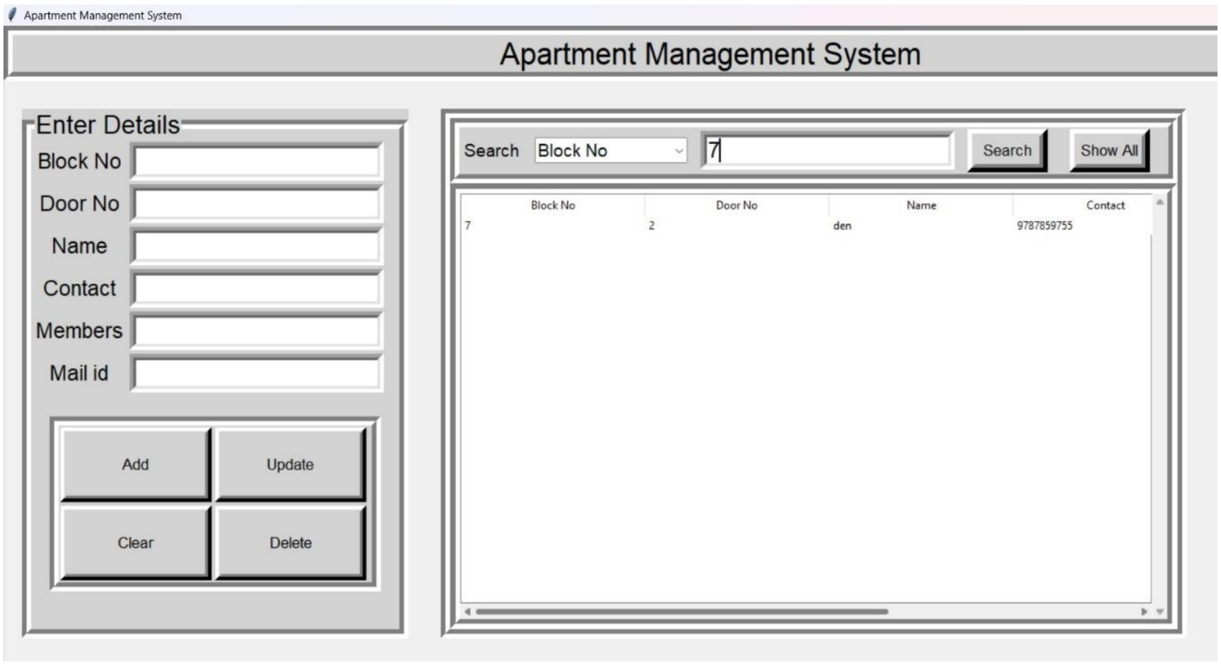
SEARCH FUNCTION:


Apartment Management System

Enter Details

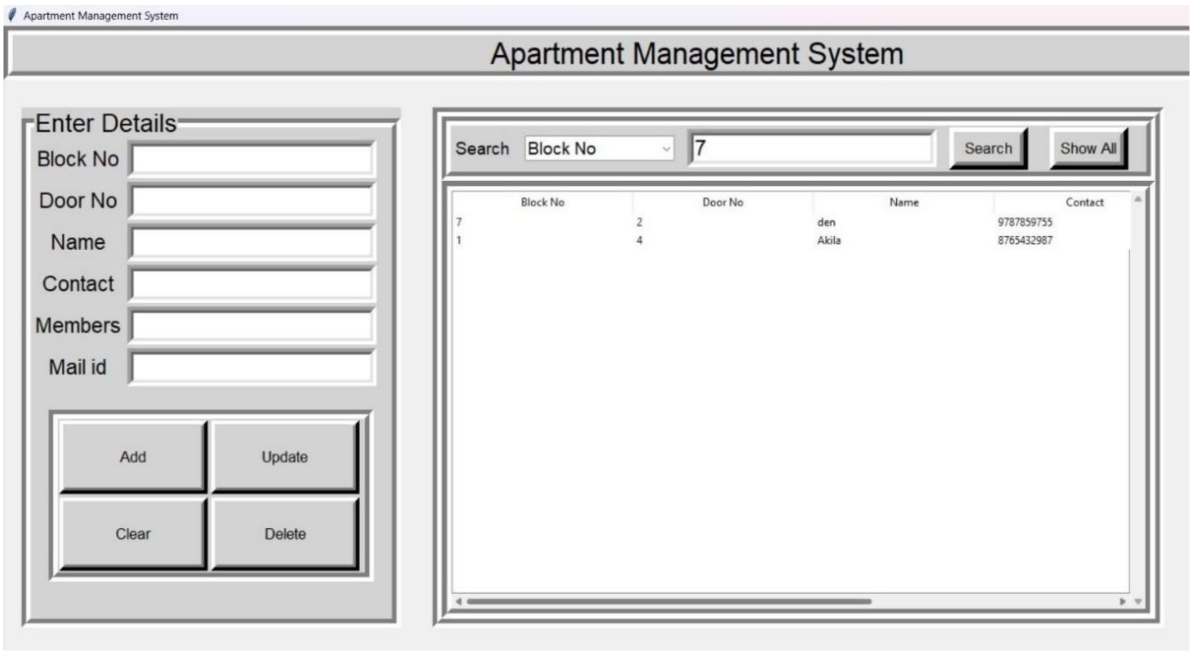
Search: Block No — 7

Block No	Door No	Name	Contact
7	2	den	9787859755
1	4	Akila	8765432987



Block No	Door No	Name	Contact
7	2	den	9787859755

SHOW ALL FUNCTION:



Block No	Door No	Name	Contact
7	2	den	9787859755
1	4	Akila	8765432987

CHAPTER-6

6.1 CONCLUSION

The completion of the Apartment Management System project, specifically designed to add, update, delete, and view all details of residents, signifies a pivotal step towards modernizing property management practices. This system is tailored to streamline the complexand often cumbersome processes associated with managing residential properties, ensuring a more organized, efficient, and responsive management approach.

In conclusion, the Apartment Management System successfully addresses the essential needs of property management by providing a reliable, efficient, and secure platform for managing resident details. Theimplementation of this system marks a significant improvement in operational workflows, data management, and overall service quality.

CHAPTER-7

7.1 REFERENCES:

1.Python's Official Documentation:

Tkinter Documentation: https://docs.python.org/3/library/tkinter.html2.MySQL Connector/Python Documentation:

https://dev.mysql.com/doc/connector-python/en/3.Python GUI Programming With Tkinter:

https://realpython.com/python-gui-tkinter/

4. Python Tkinter Tutorial:

https://www.tutorialspoint.com/python/python_gui_programming.htm

5. Python MySQL Tutorial:

https://www.tutorialspoint.com/python/python_database_access.htm