

VISVESVARAYA TECHNOLOGICAL UNIVERSITY



CONTACT MANAGEMENT SYSTEM

FILE STRUCTURES MINI PROJECT REPORT

Submitted by

SAMAY M SHETTY 1RF19IS047
SHREYANK G 1RF19IS066

Under the Guidance of

Dr. M. Vinoth Kumar

Associate Professor

Information Science and Engineering,

RV Institute of Technology and Management

In partial fulfillment for the award of degree

of

Bachelor of Engineering

IN

Information Science Engineering

**RV INSTITUTE OF TECHNOLOGY AND
MANAGEMENT, BANGALORE-560076**

2021-22

RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT, BANGALORE - 560076
(Affiliated to VTU, Belgaum)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the project work titled ‘ **Contact Management System** ’ is carried out by **Samay M Shetty (1RF19IS047) and Shreyank G (1RF19IS066)**, who are bonafide students of RV Institute of Technology and Management, Bangalore, in partial fulfillment for the award of degree of **Bachelor of Engineering in Information Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year **2021-2022**. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed by the institution for the said degree.

Signature of Guide:

Dr.M.Vinoth Kumar,
Associate Professor
Dept of ISE,
RVITM, Bangalore.

Signature of Head of the Department:

Dr.Latha C A,
Prof. & Head,
Dept of ISE,
RVITM, Bangalore.

Signature of Principal:

Dr.Jayapal R,
Professor,
RVITM,
Bangalore.

External Viva

Name of Examiners

Signature with date

1

2

RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT, BANGALORE - 560076
(Affiliated to VTU, Belgaum)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

DECLARATION

We, **Samay M Shetty and Shreyank G**, the students of sixth semester B.E., **Information Science and Engineering** hereby declare that the project titled **“Contact Management System”** has been carried out by us and submitted in partial fulfillment for the award of degree of Bachelor of Engineering in **Information Science and Engineering**. We do declare that this work is not carried out by any other students for the award of degree in any other branch.

Place: Bangalore

Date:18-07-2022

Names

1.Samay M Shetty

2.Shreyank G

Signature

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the mini project would be incomplete without the mention of those people, who made it possible through their guidance, support and encouragement. We are thankful to all the people who have directly and indirectly involved in this project work.

We would like to thank the **VTU, Belagavi**, for having this mini project work as part of its curriculum, which gave us a wonderful opportunity to work on our project work and presentation abilities.

We extend our deep sense of sincere gratitude to **Dr. Jayapal R**, Principal, RV Institute of Technology and Management (RVITM), Bengaluru, for providing us an opportunity to finish the necessary work.

We express our heartfelt sincere gratitude to **Dr. Latha C A, Professor & Head, Department of Information Science and Engineering, RVITM**, Bengaluru, for her valuable suggestions and support.

We wish to express our sincere thanks to our guide **Dr. Vinoth Kumar M, Associate Professor, Department of Information Science and Engineering, RVITM** for his expertise and encouragement for the successful completion of the project work.

We wish to express our sincere thanks to our project coordinators **Mr. Deepak D J, Assistant Professor and Dr. Vinoth Kumar M, Associate Professor, Department of Information Science and Engineering, RVITM** for the sincere and valuable guidance for the successful completion of the project work

Finally, we would like to thank all the faculty members of Department of Information Science and Engineering, RVITM, Bengaluru, for their support.

Samay M Shetty - 1RF19IS047

Shreyank G - 1RF19IS066

ABSTRACT

The purpose of Contact Management System is to automate the existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Contact Management System, as described above, can lead to error free secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information.

The aim is to automate its existing manual system by the help of computerized equipment and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. Basically the project describes how to manage for good performance and better services for the clients.

The main objective of the project on Contact Management System is to manage the details of Contact, Telephone, Email, and Gender. It manages all the information about Contact, Telephone, Email, and Gender. The project is totally built at administrative and thus only the administrator is guaranteed the access. The purpose of the project is to build and application program to reduce the manual work for managing the Contact, Telephone, Email, Gender. It tracks all the details about the Contact, Telephone, Email, and Gender.

TABLE OF CONTENTS

Acknowledgement	iii
Abstract	iv
1. Introduction	1
2. Theory and Fundamentals	2
3. Design	
3.1 Design and Implications	
4. Implementation	5
4.1 Hardware Requirements	
4.2 Software Requirements	
4.3 Quality Requirements	
4.4 Graphical User Interface	
4.5 File Structures Concept Used	7
4.5.1 Opening Files in Python	
4.5.2 Closing Files in Python	
4.5.3 Writing to Files in Python	
4.5.4 Reading Files in Python	
4.6 Modules	9
4.6.1 Main.py	
4.6.2 Views.py	
5. Screenshots	19
5.1: Viewing all contacts	
5.2: Adding contacts	
5.3: Searching contact	
5.4: Deleting contacts	
5.5: Updating contacts	
6. Conclusions and Future Scope	24
References	25

CHAPTER 1

INTRODUCTION

The Contact Management System has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. Moreover this system is designed for the particular need of the company to carry out operations in a smooth and effective manner.

The application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus by this all it proves it is user-friendly. Contact Management System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources.

Every organization, whether big or small, has challenges to overcome and managing the information of Contact, Telephone, Email, and Gender. Every Contact Management System has different contact needs. Therefore we design exclusive employee management systems that are adapted to your managerial requirements. This is designed to assist in strategic planning, and will help you ensure that your organization is equipped with the right level of information and details for your future goals. Also, for those busy executive who are always on the go, our systems come with remote access features, which will allow you to manage your workforce anytime, at all times. These systems will ultimately allow you to better manage resources.

The scope of the project is to collect management details in perfect. The collection will be obvious, simple and sensible.

CHAPTER 2

THEORY AND FUNDAMENTALS

The principle behind constructing a Contact Management System is to effectively retrieve and implement any information that an organization may have on a pre-existing customer.

All the information related to a particular customer can be linked and archived only to be retrieved later when they are required most. What should be more noteworthy is the fact that all of this can be arranged over every single terminal and at an affordable cost.

Once a company has established a Contact Management System, it can cater to the needs of the customer better. The records of the customer can now be studied to understand his or her preferences and implement them for each individual lead. This gives way to contact personalization.

By managing contacts, the organization can also look for opportunities to upsell and cross-sell a product or service depending on the customer's current situation. By doing so, the company achieves a two-part objective.

On one hand, it establishes a relationship and wins the trust of the customer with consistent assistance. On the other hand, a happy customer can also generate the maximum profits as well as an assured guarantee that he or she will visit the company again sooner or later.

CHAPTER 3

DESIGN

3.1 Design and Implications

Several design suggestions follow from these results. First, our regression analysis is a model for identifying important contacts in email, and this could be implemented directly as an algorithm.

The ability to automatically identify important contacts from communication archives might be used in a number of applications, allowing us to improve messaging applications, support reminding and provide social recommendation. Messaging applications are currently poorly integrated with contact management tools, but future systems could exploit information about important contacts in a variety of ways. These might include alerting, filtering and prioritization of incoming email or voicemail messages based on the sender's importance.

Tighter integration of contact information with messaging logs could be also used to manage relationships with contacts, e.g. reminding the user when they haven't talked to an important contact in a long time. We have implemented contact-based alerting and reminding in a social network based user interface to communication and information .

Finally social recommendation systems might be able to exploit information about a register of important contacts to either direct a user query or guide information access. Other design implications concern contact management tools directly. We could improve address book utility by using our algorithm to automatically recommend that a potentially important contact should be added to the address book, based on their communication history.

But even if we provide ways to better identify significant contacts, data entry is still a major problem. One possible way to address this would be to identify contact information from other sources, such as Internet home pages containing addresses. We may also be able to mine other types of records such as phone and voicemail logs, or use

reverse lookup to provide detailed addresses for contacts.

Having general techniques for populating address books is clearly important. One unexpected finding from our research was that 72% of important contacts came from outside the user's organization. While this may depend on the specific user population, it suggests that corporate address books or intranets have limited utility as a way to provide detailed addresses for contacts.

CHAPTER 4

IMPLEMENTATION

4.1 HARDWARE REQUIREMENTS

- Processor: Pentium 4 or higher
- RAM: 512 MB or more
- Memory Space 80 GB or higher.

4.2 SOFTWARE REQUIREMENTS

- OS : Windows 7/Windows 10/ Windows 11
- Browser : Mozilla Firefox/Google Chrome/Microsoft Edge
- Scripting Language : Python v3.8 or above
- Software used for scripting language : Visual Studio

4.3 QUALITY REQUIREMENTS

- Functional Quality: System should comply with or conforms to the given design, based on functional requirements or specifications.
- Structural quality: It refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly.
- Ease Of Access: The user among the system must easily be able to move forward into the system and easily and interactively use the various features of the application and the project should be able to respond to the users demand successfully and immediately.
- Security: Important requirement of all others is the security. It is the most important part of any project as the information has to be kept secure from malicious users.

Reliability: As it measures the level of risk and the likelihood of potential application failures as well as the defects injected due to modifications made to the software, this is an essential quality requirement.

- **Portability:** The project should be able to swiftly run on any system meeting the mentioned software and hardware requirements.
- **Maintainability:** The maintenance of the project should be easy and the cost required for maintenance should also be efficient.
- **Consistency:** Any operation must be consistent, which means that each operation performed must be performed completely.
- **Size:** While not a quality attribute per se, the sizing of source code is a software characteristic that obviously impacts maintainability.
- **Modularity:** The project must be built after breaking it into various modules so that no point is missed out and the complexity in the analysis, design and coding is reduced.

4.4 GRAPHICAL USER INTERFACE

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

- Import the Tkinter module : `import tkinter`
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

4.5 FILE STRUCTURE CONCEPT USED

File handling is an important part of any web application. Python has several functions for creating, reading, updating, and deleting files.

Python also supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files. The concept of file handling has stretched over various other languages, but the implementation is either complicated or lengthy, but like other concepts of Python, this concept here is also easy and short. Python treats files differently as text or binary and this is important. Each line of code includes a sequence of characters and they form a text file. Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun.

4.5.1 Opening Files in Python

Before performing any operation on the file like reading or writing, first, we have to open that file. For this, we should use Python's inbuilt function `open()` but at the time of opening, we have to specify the mode, which represents the purpose of the opening file.

```
f = open (filename, mode)
```

Where the following mode is supported:

r: open an existing file for a read operation.

w: open an existing file for a write operation. If the file already contains some data then it will be overridden.

a: open an existing file for append operation. It won't override existing data.

r+: To read and write data into the file. The previous data in the file will be overridden.

w+: To write and read data. It will override existing data.

a+: To append and read data from the file. It won't override existing data.

4.5.2 Closing Files in Python

When we are done with performing operations on the file, we need to properly close the file. Closing a file will free up the resources that were tied with the file. It is done using the `close()` method available in Python. Python has a garbage collector to clean up unreferenced objects but we must not rely on it to close the file.

4.5.3 Writing to Files in Python

In order to write into a file in Python, we need to open it in write `w`, append `a` or exclusive creation `x` mode. We need to be careful with the `w` mode, as it will overwrite into the file if it already exists. Due to this, all the previous data are erased. Writing a string or sequence of bytes (for binary files) is done using the `write ()` method. This method returns the number of characters written to the file.

4.5.4 Reading Files in Python

To read a file in Python, we must open the file in reading `r` mode. There are various methods available for this purpose. We can use the `read(size)` method to read in the size number of data. If the size parameter is not specified, it reads and returns up to the end of the file.

4.6 MODULES

View: Allows the user to view all the contact information stored in the system.

Add: It also has the functionality of adding new contacts into the system.

Delete: Users can delete contact information from the list.

Update: Allows users to modify the record details.

Search: It allows users to search details based on their phone numbers.

4.6.1 Main.py

```
from tkinter import *
from tkinter import ttk

from views import *
from tkinter import messagebox

#colors
co0 = "#ffffff"
co1 = "#000000"
co2 = "#4456F0"

window = Tk()
window.title("")
window.geometry('485x450')
window.configure(background=co0)
window.resizable(width=FALSE, height=FALSE)

#frames
frame_up = Frame(window, width=500, height=50, bg=co2)
frame_up.grid(row=0, column=0, padx=0, pady=1)

frame_down = Frame(window, width=500, height=150, bg=co0)
frame_down.grid(row=1, column=0, padx=0, pady=1)
```

```

frame_table = Frame(window, width=500, height=100, bg=co0, relief="flat")
frame_table.grid(row=2, column=0, columnspan=2, padx=10, pady=1, sticky=NW)

#functions
def show():
    global tree

    listheader = ['Name', 'Gender', 'Telephone', 'Email']

    demo_list = view()

    tree = ttk.Treeview(frame_table, selectmode="extended", columns=listheader,
show="headings")

    vsb = ttk.Scrollbar(frame_table, orient="vertical", command=tree.yview)
    hsb = ttk.Scrollbar(frame_table, orient="horizontal", command=tree.xview)

    tree.configure(yscrollcommand=vsb.set, xscrollcommand=hsb.set)

    tree.grid(column=0, row=0, sticky='nsew')
    vsb.grid(column=1, row=0, sticky='ns')
    hsb.grid(column=0, row=1, sticky='ew')

#tree head
tree.heading(0, text='Name', anchor=NW)
tree.heading(1, text='Gender', anchor=NW)
tree.heading(2, text='Telephone', anchor=NW)
tree.heading(3, text='Email', anchor=NW)

# tree columns
tree.column(0, width=120, anchor='nw')
tree.column(1, width=50, anchor='nw')
tree.column(2, width=100, anchor='nw')
tree.column(3, width=180, anchor='nw')

```



```
for item in demo_list:
    tree.insert("", 'end', values=item)

show()

def insert():

    Name = e_name.get()
    Gender = c_gender.get()
    Telephone = e_telephone.get()
    Email = e_email.get()

    data = [Name, Gender, Telephone, Email]

    if Name == "" or Gender == "" or Telephone == "" or Email == "":
        messagebox.showwarning('data', 'Please fill in all fields')

    else:
        add(data)
        messagebox.showinfo('data', 'data added successfully')

        e_name.delete(0, 'end')
        c_gender.delete(0, 'end')
        e_telephone.delete(0, 'end')
        e_email.delete(0, 'end')

        show()

def to_update():
    try:
        tree_data = tree.focus()
        tree_dictionary = tree.item(tree_data)
        tree_list = tree_dictionary['values']
```

```
Name = str(tree_list[0])
Gender = str(tree_list[1])
Telephone = str(tree_list[2])
Email = str(tree_list[3])

e_name.insert(0, Name)
c_gender.insert(0, Gender)

e_telephone.insert(0, Telephone)
e_email.insert(0, Email)

def confirm():
    new_name = e_name.get()
    new_gender = c_gender.get()
    new_telephone = e_telephone.get()
    new_email = e_email.get()

    data = [new_telephone, new_name, new_gender, new_telephone, new_email]

    update(data)

    messagebox.showinfo('Success', 'data updated successfully')

    e_name.delete(0, 'end')
    c_gender.delete(0, 'end')
    e_telephone.delete(0, 'end')
    e_email.delete(0, 'end')

    for widget in frame_table.winfo_children():
        widget.destroy()

    b_confirm.destroy()

    show()
```

```

        b_confirm = Button(frame_down, text="Confirm", width=10, height=1, bg=co2, fg
= co0, font=('Ivy 8 bold'), command=confirm)
        b_confirm.place(x = 290, y = 110)

```

```

except IndexError:

```

```

    messagebox.showerror('Error', 'Select one of them from the table')

```

```

def to_remove():

```

```

    try:

```

```

        tree_data = tree.focus()

```

```

        tree_dictionary = tree.item(tree_data)

```

```

        tree_list = tree_dictionary['values']

```

```

        tree_telephone = str(tree_list[2])

```

```

        remove(tree_telephone)

```

```

        messagebox.showinfo('Success', 'Data has been deleted successfully')

```

```

    for widget in frame_table.winfo_children():

```

```

        widget.destroy()

```

```

    show()

```

```

except IndexError:

```

```

    messagebox.showerror('Error', 'Select one of them from the table')

```

```

def to_search():

```

```

    telephone = e_search.get()

```

```

    data = search(telephone)

```

```

def delete_command():

```

```

    tree.delete(*tree.get_children())

```

```

delete_command()

for item in data:
    tree.insert("", 'end', values = item)

e_search.delete(0, 'end')

#frame_up widgets

app_name = Label(frame_up, text="Contact Management System", height = 1,
font=('Verdana 17 bold'), bg = co2, fg = co0)
app_name.place(x=5, y=5)

#frame_down widgets

l_name = Label(frame_down, text="Name *", width=20, height=1, font=('Ivy 10'),
bg=co0, anchor=NW)
l_name.place(x=10, y=20)
e_name = Entry(frame_down, width=25, justify='left', highlightthickness=1,
relief="solid")
e_name.place(x=80, y=20)

l_gender = Label(frame_down, text="Gender *", width=20, height=1, font=('Ivy 10'),
bg=co0, anchor=NW)
l_gender.place(x=10, y=50)
c_gender = ttk.Combobox(frame_down, width=27)
c_gender['values'] = ['F', 'M']
c_gender.place(x=80, y=50)

l_telephone = Label(frame_down, text="Telephone*", height=1, font=('Ivy 10'), bg=co0,
anchor=NW)
l_telephone.place(x=10, y=80)
e_telephone = Entry(frame_down, width=25, justify='left', highlightthickness=1,
relief="solid")
e_telephone.place(x=80, y=80)

```

```

l_email = Label(frame_down, text="Email *", height=1, font=('Ivy 10'), bg=co0,
anchor=NW)
l_email.place(x=10, y=110)
e_email = Entry(frame_down, width=25, justify='left', highlightthickness=1,
relief="solid")
e_email.place(x=80, y=110)

b_search = Button(frame_down, text="Search", height=1, bg=co2, fg = co0,font=('Ivy 8
bold'), command=to_search)
b_search.place(x=290, y=20)
e_search = Entry(frame_down, width=16, justify='left', font=('Ivy', 11),
highlightthickness=1, relief="solid")
e_search.place(x=347, y=20)

b_view = Button(frame_down, text="View", width=10, height=1, bg=co2, fg =
co0,font=('Ivy 8 bold'), command = show)
b_view.place(x=290, y=50)

b_add = Button(frame_down, text="Add", width=10, height=1, bg=co2, fg =
co0,font=('Ivy 8 bold'), command=insert)
b_add.place(x=400, y=50)

b_update = Button(frame_down, text="Update", width=10, height=1, bg=co2, fg =
co0,font=('Ivy 8 bold'), command=to_update)
b_update.place(x=400, y=80)

b_delete = Button(frame_down, text="Delete", width=10, height=1, bg=co2, fg =
co0,font=('Ivy 8 bold'), command = to_remove)
b_delete.place(x=400, y=110)

window.mainloop()

```

4.6.2 views.py

```
from asyncore import write
import re
import sys
import csv

def add(i):
    with open('data.csv','a+',newline=") as file:
        writer = csv.writer(file)
        writer.writerow(i)

#add(['Anonymous','M','54321','data@gmail.com'])

def view():
    data = []
    with open('data.csv') as file:
        reader =csv.reader(file)
        for row in reader:
            data.append(row)
    print(data)
    return data

view()

def remove(i):

    def save(j):
        with open('data.csv', 'w', newline=") as file:
            writer = csv.writer(file)
            writer.writerows(j)

    new_list = []
```

```

telephone = i

with open('data.csv') as file:

    reader = csv.reader(file)
    for row in reader:
        new_list.append(row)

    for element in row:
        if element == telephone:
            new_list.remove(row)
save(new_list)

def update(i):
    def update_newlist(j):
        with open('data.csv','w',newline=") as file:
            writer = csv.writer(file)
            writer.writerows(j)

    new_list = []
    telephone = i[0]
    # ['123', 'demo', 'M', '123' , 'demo@gmail.com']

    with open('data.csv', 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            new_list.append(row)
            for element in row:
                if element == telephone:
                    name = i[1]
                    gender = i[2]
                    telephone = i[3]
                    email = i[4]
                    data = [name, gender, telephone, email]

```

```
        index = new_list.index(row)
        new_list[index] = data
    update_newlist(new_list)
```

```
def search(i):
    data = []
    telephone = i

    with open('data.csv','r') as file:
        reader = csv.reader(file)
        for row in reader:
            for element in row:
                if element == telephone:
                    data.append(row)

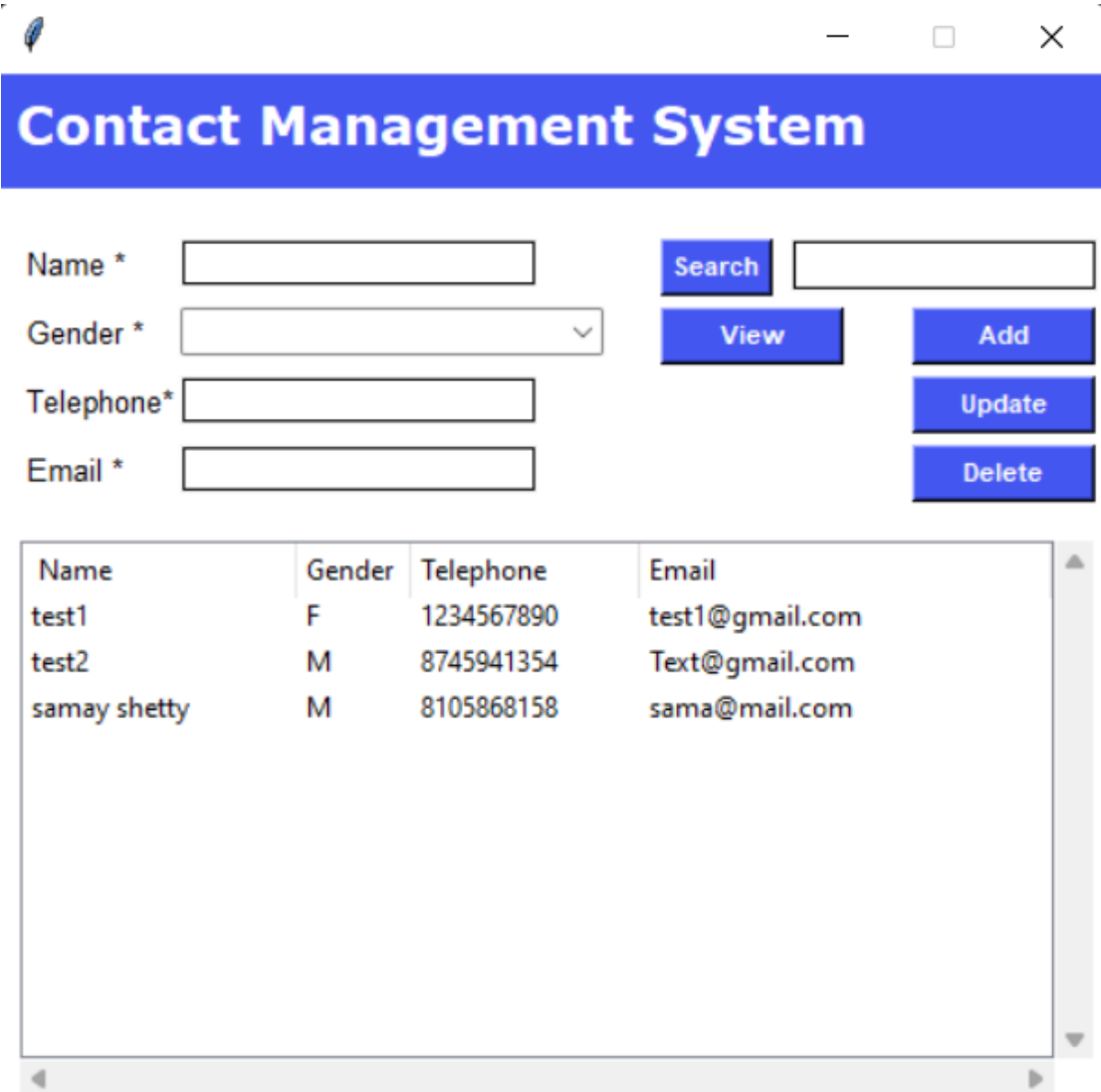
    return data
```

4.6.3 data.csv

test1,F,1234567890,test1@gmail.com

test2,M,8745941354,Text@gmail.com

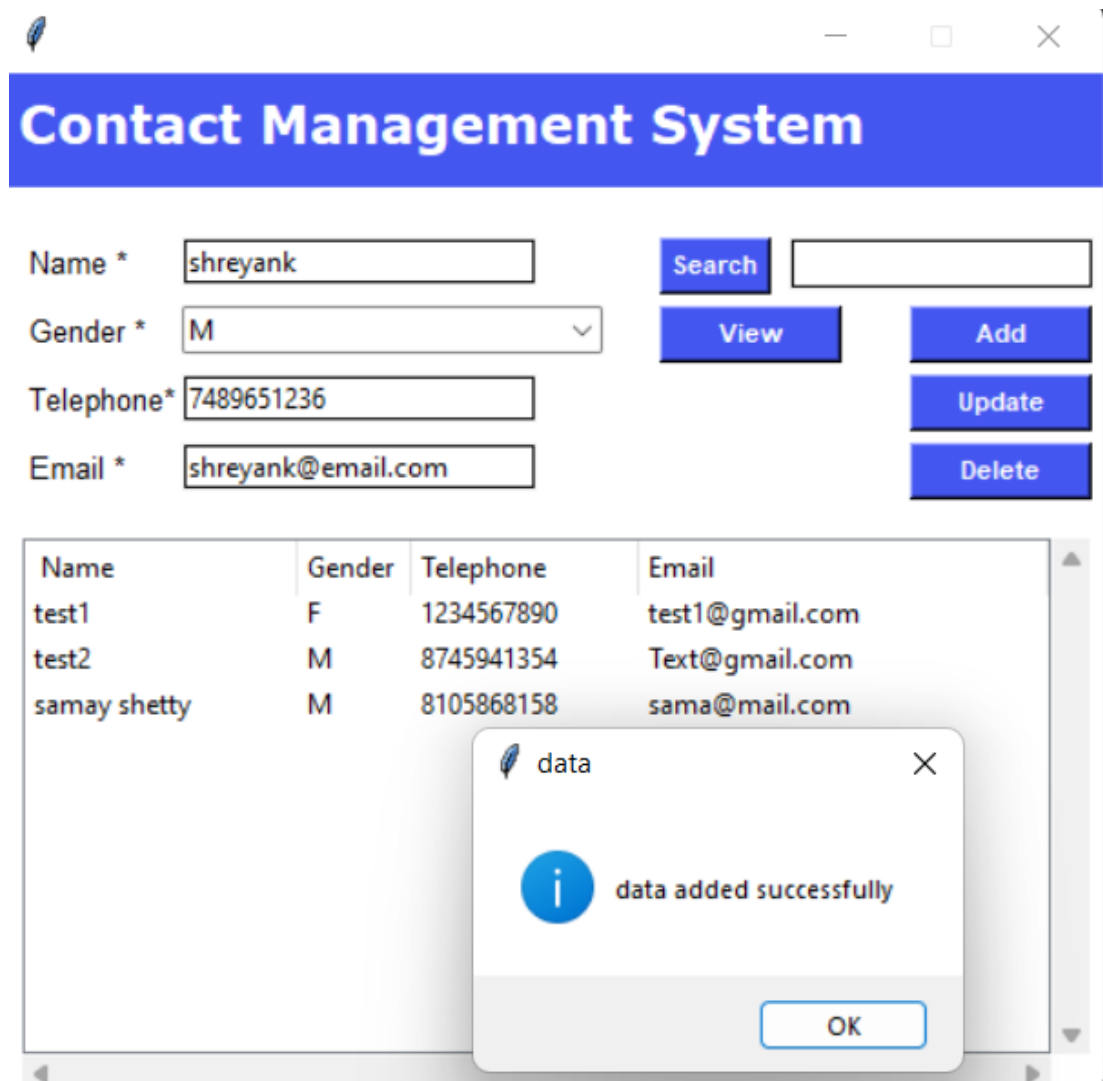
samay shetty,M,8105868158,sama@mail.com

CHAPTER 5**SNAPSHOTS**

The screenshot shows a web application window titled "Contact Management System". It features a form with four input fields: "Name *", "Gender *", "Telephone*", and "Email *". To the right of the form are four buttons: "Search", "View", "Add", "Update", and "Delete". Below the form is a table displaying a list of contacts.

Name	Gender	Telephone	Email
test1	F	1234567890	test1@gmail.com
test2	M	8745941354	Text@gmail.com
samay shetty	M	8105868158	sama@mail.com

5.1 Viewing all contacts



Contact Management System

Name *


Gender *

Telephone*

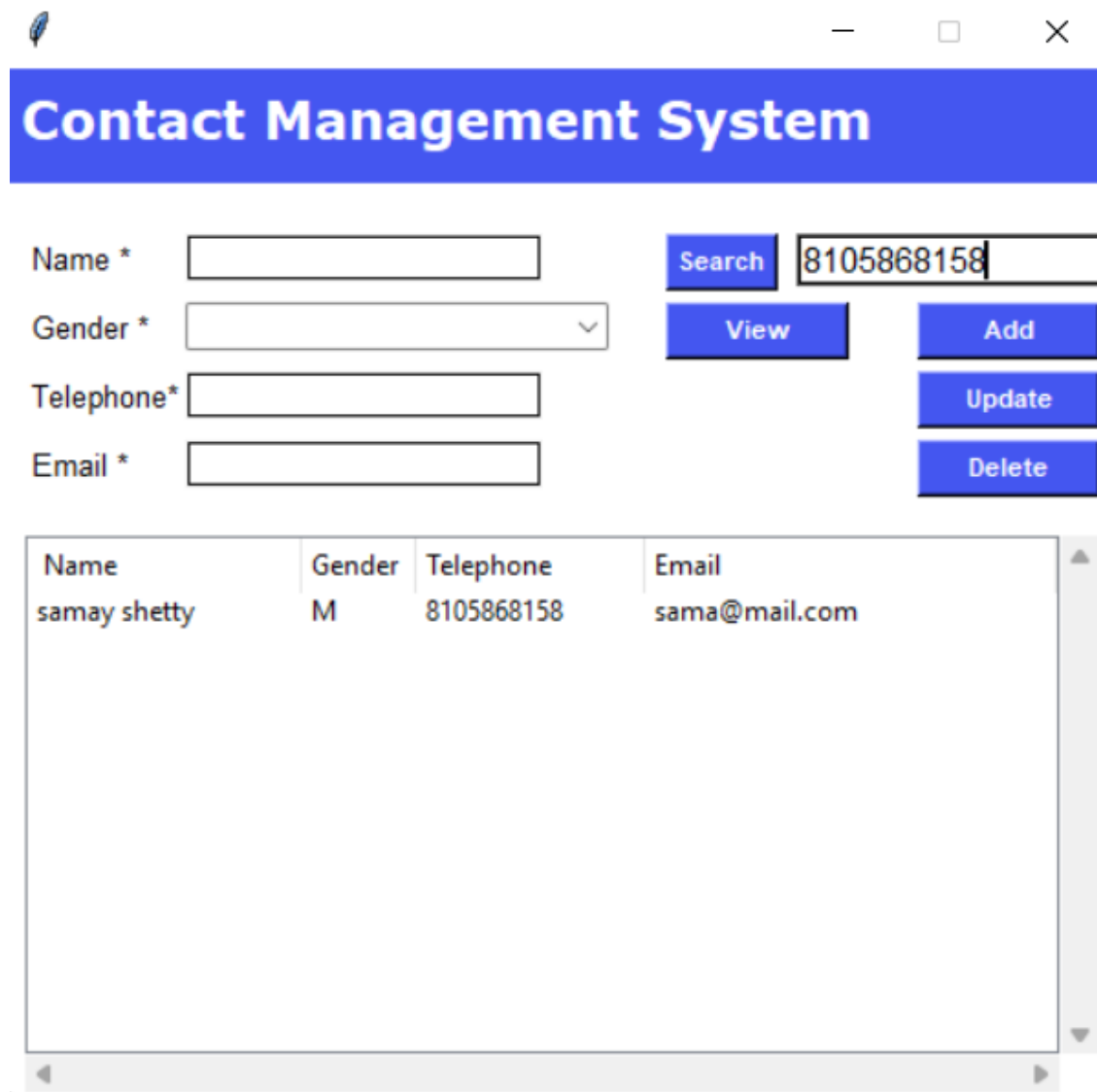
Email *

Name	Gender	Telephone	Email
test1	F	1234567890	test1@gmail.com
test2	M	8745941354	Text@gmail.com
samay shetty	M	8105868158	sama@mail.com

data

 data added successfully

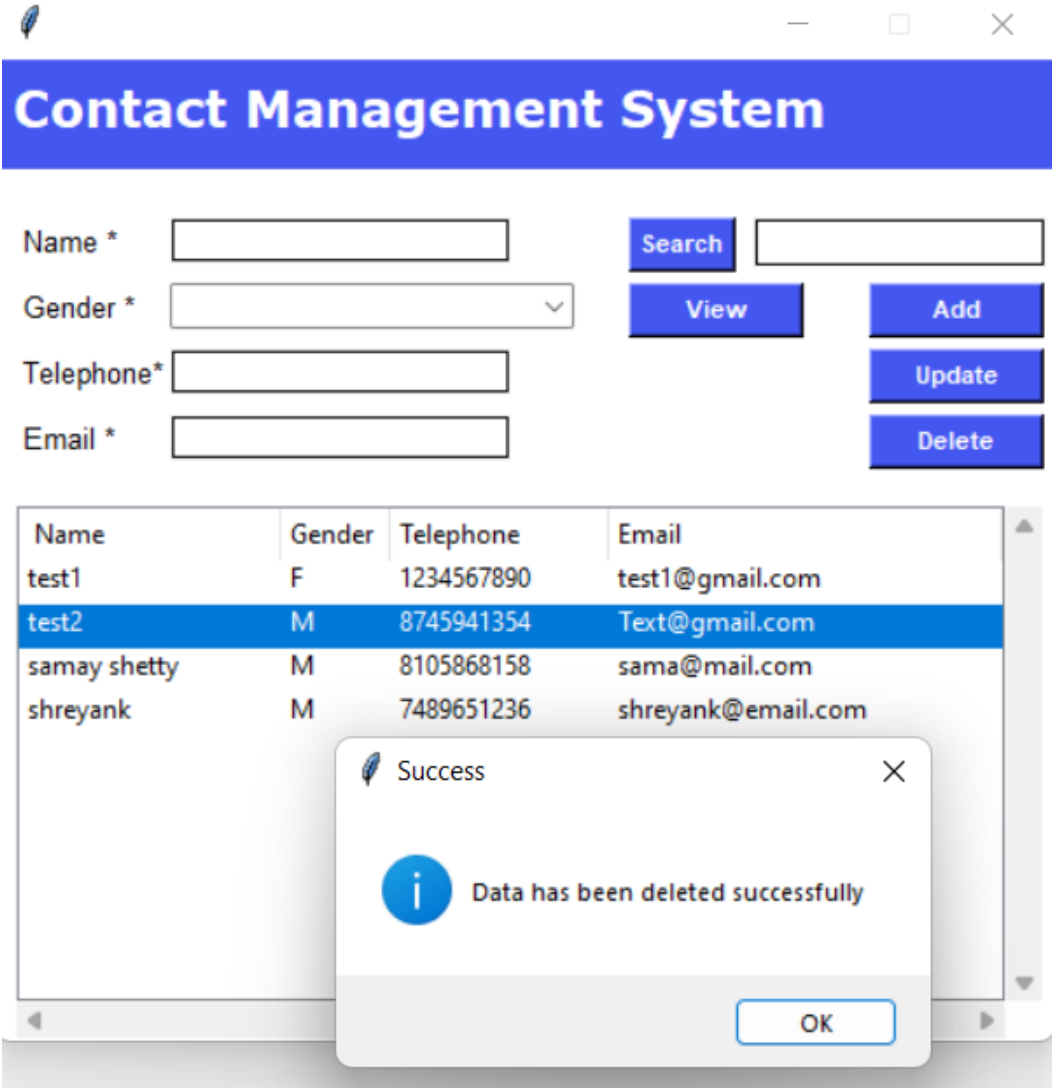
5.2 Adding contacts



A screenshot of a web application window titled "Contact Management System". The window has a blue header bar with the title. Below the header, there are input fields for "Name *", "Gender *", "Telephone*", and "Email *". To the right of these fields are four blue buttons: "Search", "View", "Add", "Update", and "Delete". The "Search" button is highlighted, and the text "8105868158" is entered in the search input field. Below the input fields and buttons is a table with four columns: "Name", "Gender", "Telephone", and "Email". The table contains one row of data: "samay shetty", "M", "8105868158", and "sama@mail.com".

Name	Gender	Telephone	Email
samay shetty	M	8105868158	sama@mail.com

5.3 Searching contacts



The screenshot displays a web application titled "Contact Management System". It features a form with fields for Name, Gender, Telephone, and Email, each with an asterisk indicating it is required. To the right of the form are buttons for Search, View, Add, Update, and Delete. Below the form is a table with four columns: Name, Gender, Telephone, and Email. The table contains five rows of data. A modal dialog box is open in the foreground, displaying a success message: "Data has been deleted successfully".

Name	Gender	Telephone	Email
test1	F	1234567890	test1@gmail.com
test2	M	8745941354	Text@gmail.com
samay shetty	M	8105868158	sama@mail.com
shreyank	M	7489651236	shreyank@email.com

Success

Data has been deleted successfully

OK

5.4 Deleting contacts

Contact Management System

Name *

Gender *

Telephone*

Email *

Name	Gender	Telephone	Email
test1	F	1234567890	test1@gmail.com
samay shetty	M	8105868158	sama@mail.com
shreyank	M	7489651236	shreyank@email.com

Success

data updated successfully

5.5 Updating contacts

CHAPTER 6

CONCLUSION

In this project, the concept of file handling became very clear as it allows us to deal with records by performing operations on it. This project contains generous amount of features, which will help a user to use their phone in a more regular way.

If one's business struggles to keep up with customer data, you need a contact management system. This software automates the whole process of collecting and retrieving customer information. Besides, it leads to increased productivity, collaborations, and customer satisfaction.

This program deals with four operations of adding contacts, deleting them, modifying, searching according the user's choice. Each operation is made as an individual function and so control enters to different structures and all the data added or modified or deleted is going to be stored in a .csv file.

CHAPTER 7

REFERENCES

- [1] [Michael J. Folk, Bill Zoellick, Greg Riccardi \(1998\): File Structures-An Object Oriented Approach with C++, 3rd Edition, Pearson Education.,](#)
- [2] [Al Sweigart](#)(2017): Automate the Boring Stuff With Python: Practical Programming
- [3] [Charles Severance](#)(2016): Python for Everybody: Exploring Data Using Python 3
- [4] [Eric Matthes](#)(2019): Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming
- [5] The Complete Python Course:
<https://www.udemy.com/course/the-complete-python-course/>