Jeevandeep Mahavidyalaya

TITLE OF THE Mini PROJECT

Customer Management System

SUBMITTED BY

Aryan Singh
Enrollment No.**: KA2K22/132870010**

DATE OF SUBMISSION
10/03/2024

Under The Guidance Of: ***.Mr. Shivchand Sir***

Submitted in partial fulfillment of the requirements for qualifying Bsc.

Jeevandeep Mahavidyalaya
Affilated to MGKVP Varanasi
Bada Lalpur ,P.O.Lamhi,Varanasi

Project certificate

This is to certify that the Project/Dissertation entitled, **Customer Management System** is a bona fide work done by **Aryan Singh** (Enrollment No. **KA2K22/132870010**) in partial fulfillment of Bsc examination has been carried out under my direct supervision and guidance. This report or a similar report on the topic has not been submitted for any other examination and does not form part of any other course undergone by the candidate.

Jeevandeep Mahavidyalaya

Affilated to MGKVP Varanasi

Bada Lalpur ,P.O.Lamhi,Varanasi

Submmited by:

Aryan Singh

Jeevandeep Mahavidyalaya

Roll No : KA2K22/132870010

Guided by:

Shiv Chand Sir

Computer Faculty

Jeevandeep Mahavidyalaya

Lamhi,Varanasi

Signature :

Signature :

Preface

The main aim of science technology is to understand natural event. We feel this event with our mind, which is main part of our body. Computer is also gift of science, which also works like a brain, but the manner speed at so many other properties are much more than a men's brain. Modern age is an age of computer. Computer has just become a part of our life.

The computer education in India was started in somewhere in sixties. Now with the advent of micro computer, it has become possible to have mass education in this field. The major advantage of computers is the speed which calculated are the and relative of output comes.

Computer can store much information and solve our problem within very short duration.

So keeping these things in views, I have computerized my small project work "**Customer Management System".**I have maintained this project work on the suitable and advance language i.e. "**Python**".

Table of contents

Project Title

Introduction

Welcome to **CUSTOMER MANAGEMENT SYSTEM**. This project is for many purposes because I want to give full satisfaction of the user.

In my project I provide many facilities to user. In Customer Management System it is very difficult to manage all things but with the help of my software it becomes very easy to handle. It saves our time and paper work.

So we consider the above problem and make this software. In this software you store all kinds of information like-

Objective of the Project

Our consideration while developing this software was to achieve the following goals:-

1- This software is for Customer Management so it provides those facilities, which is used in any office.

2- It also saves the valuable time of the shop owner and lots of paper works.

3- This will save lots of effort of the user. He/She is able to manage lots of information after using this software and also able for seeing in different angles.

4- User can manage all the records very easily. You can generate all kinds of reports like bundle receiving, Transport receiving, customer detail and many more reports for management.

We also try to maintain security of data. User cannot enter directly to the software without using login and password

Project Category

This project belongs to the category of application software as use the applications of Customer Management System can it. It is a desktop application, At a time only One user can use it. It is a type of commercial software so terms and conditions are provides by the vender to buyer.

Software Platform

Implementation Platform

The following software is required.

• Operating System	Window 10
• Back End	SQLite
• Front End	Python

Development Platform

The following S/W was used n developing the system.

• Operating System	Window 10
• Back End	SQLite
• Front End	Python

Hardware Platform

Development platform

The following H/W components were used in developing the system project.

RAM	- more than 64 MB
HDD	- 20 GB or greater
Monitor	- EGA/VGA
Keyboard	- Input Device
Mouse	- As a pointer Device
Processor	- Pentium Minx

Implementation Platform

The following H/W is required

RAM	- more than 64 MB
HDD	- 20 GB or greater
Monitor	-EGA/VGA
Keyboard	-Input Device
Mouse	- As a pointing device
Processor	- Pentium Minx

Feasibility Study

It has very important place in the study of any software because if the project are not feasible then working on this working on this software is not a good work.

So I conducted the feasibility study in order to check out whether our software “CUSTOMER MANAGEMENT SYSTEM” is feasible or not. So we study about thee types of feasibility:-

- Economic Feasibility
- Technical Feasibility
- Operational Feasibility

So brief description of above are following

1 **Economic Feasibility:-**

In this study, I was concerned about broad range of factors. Our software has no need of any other equipment or software. It saves very much time and lots of manual work. It also saves the paper work of the person.

So, in short we can say that it is economic feasible software and saves lots of expenses of the person and time as well.

2 Technical Feasibility:-

In technical feasibility we studied that the system is technical feasible or not. If system is not feasible then, a system analyst takes a decision about software that will be proceeding out or not.

So, we take care of that our software is overall technically feasible. I take care of the development and technology of the Varanasi region for ensuring availability of the hardware and software needs for implementation of this project and that is minimal.

3 Operational feasibility:-

Our software is user friendly. So a person who does not have any technical background can operate this software, prior knowledge of computer is not mandatory for operating the software. Icons menus and its options describing its meaning itself, input and output form are designed in user friendly way for providing proper instructions with user.

So we specially take care of that software is operational feasible.

Testing

Testing phase is one of the important phase in between developed and implementation phase. We perform testing to ensure code optimization and to ensure accuracy and validity of entered data.

The system development phases involve many activities where chances for occurrence of human errors are enormous. Logical error, carelessness, improper communication, the need to hurry through the whole process of software development due to time constraint, cost constraints etc. provide ways for errors to creep in. The system must be tested thoroughly so that such error to creep in. The system must be tested thoroughly so that such error are detected and corrected as early as possible. A successful text is one that uncovers every possible error.

There are following testing methods which are below:-

1 <u>BLACK BOX TESTING:</u>-

In black box testing we checked modules by giving test data prepared by keeping all possibilities.

2 <u>UNIT TESTING:</u>-

In unit testing each module checked for following:

1- Interface

2- Local data structure

3- Boundary conditions

4- Independents paths

5- Error handling paths

3 <u>SYSTEM TESTING</u>:-

Modules added to project in incremental approach and overall testing performed on system to evaluate its overall performance with stated requirement of user.

4 <u>WHITE BOX TESTING:</u>-

We ensure following by performing white box testing:

1. Guarantee that all independents path within a module exercised at least one.
2. Exercise all logical decision on their true and false sides.
3. Execute all loops at their boundary.
4. Exercise internal data structure to ensure validity.

By performing glass box testing, we achieved code optimization by removing unnecessary codes.

Python

Python is a high-level, versatile, and dynamically typed programming language known for its simplicity and readability. Developed by Guido van Rossum and first released in 1991, Python has gained widespread popularity and has become one of the most used languages in various domains, including web development, data science, artificial intelligence, automation, and more.

Key features of Python include:

Readability: Python's syntax is designed to be clear and readable, emphasizing code readability and reducing the cost of program maintenance.

Versatility: Python is a multiparadigm programming language, supporting object-oriented, imperative, and functional programming styles. This flexibility allows developers to choose the most suitable approach for their projects.

Extensive Libraries: Python's extensive standard library provides modules and packages for a wide range of tasks, from web development (Django, Flask) to scientific computing (NumPy, SciPy) and machine learning (TensorFlow, PyTorch).

Community Support: Python has a large and active community of developers who contribute to the language's growth. This community support is evident in the availability of resources, documentation, and third-party packages.

Cross-Platform: Python is a cross-platform language, meaning code written in Python can run on various operating systems without modification. This makes it easy to develop applications that work seamlessly across different platforms.

Interpreted Language: Python is an interpreted language, which means that the source code is executed line by line, making it easy to test and debug code. This also contributes to the language's dynamic nature.

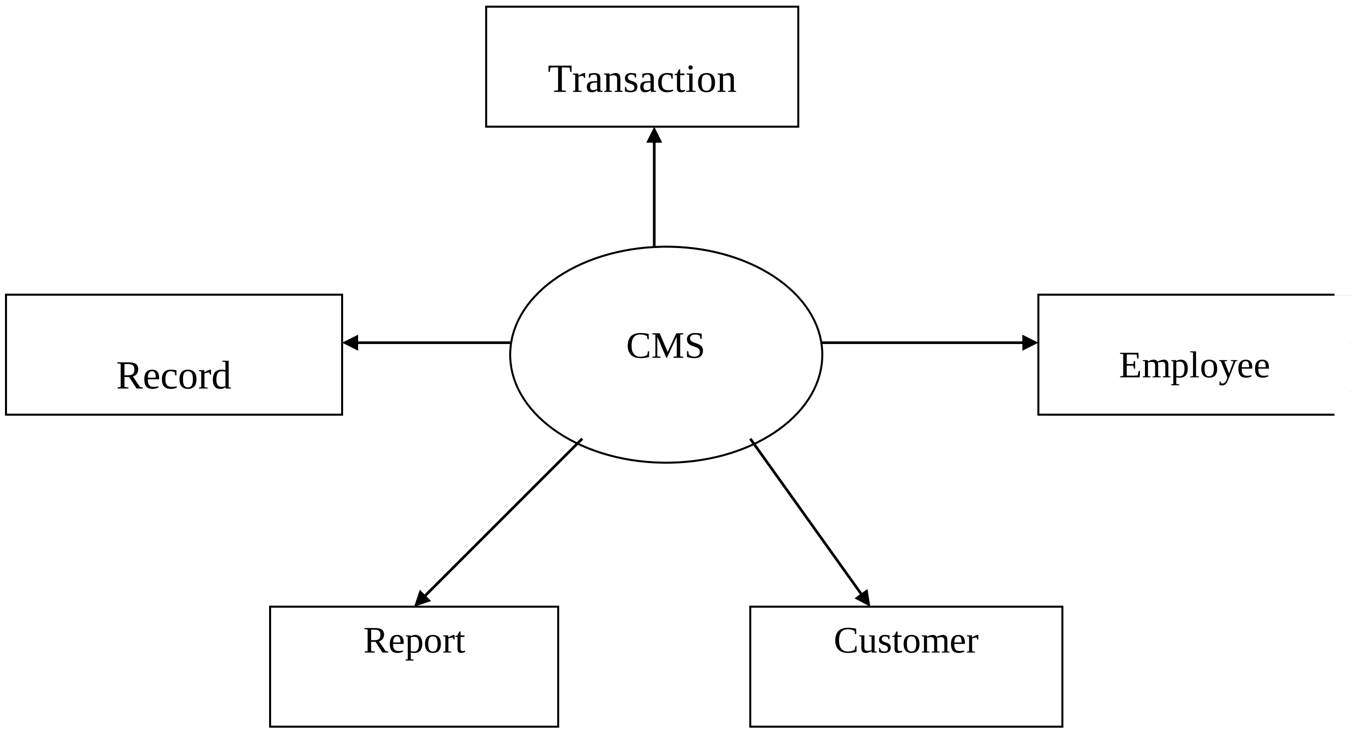
Open Source: Python is an open-source language, allowing developers to access and modify the source code according to their needs. This fosters collaboration and the continuous improvement of the language.

Large Ecosystem: Python's ecosystem includes a vast number of third-party libraries and frameworks, expanding its capabilities and making it suitable for various domains, such as web development, data analysis, machine learning, and more.

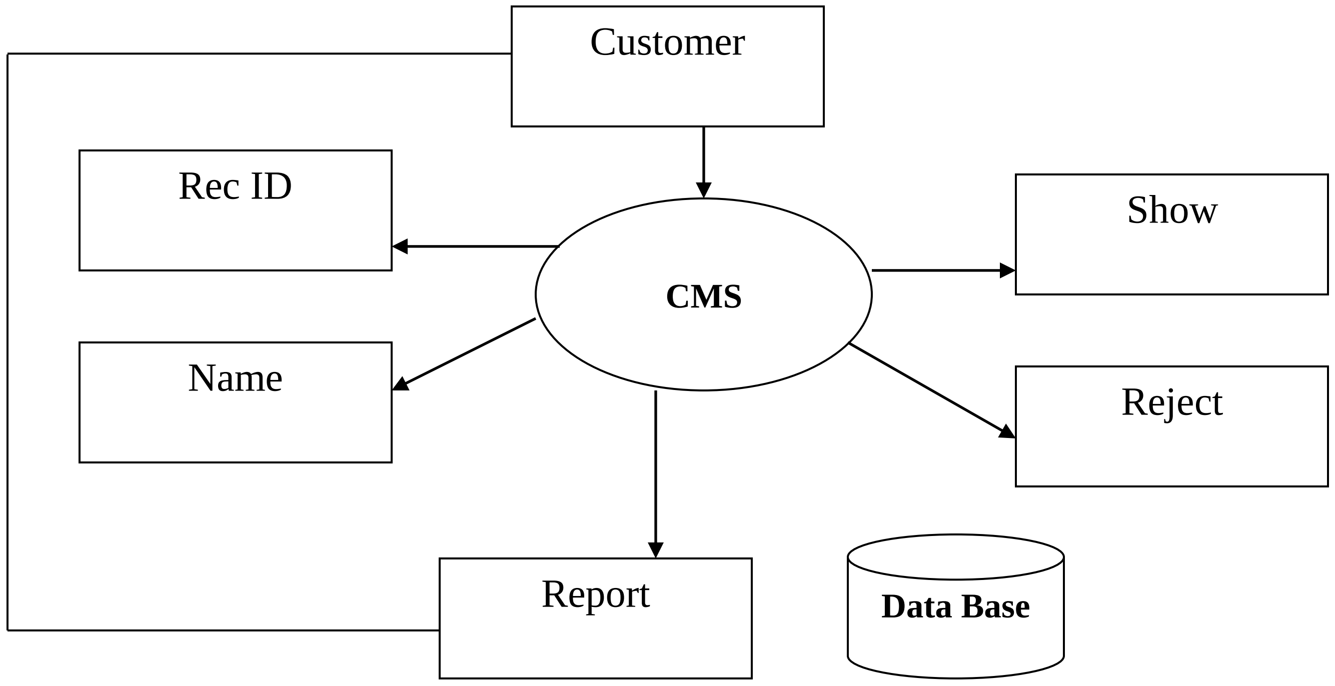
Overall, Python's simplicity, readability, and wide range of applications make it an excellent choice for both beginners and experienced developers, contributing to its sustained popularity in the software development community.

DFD

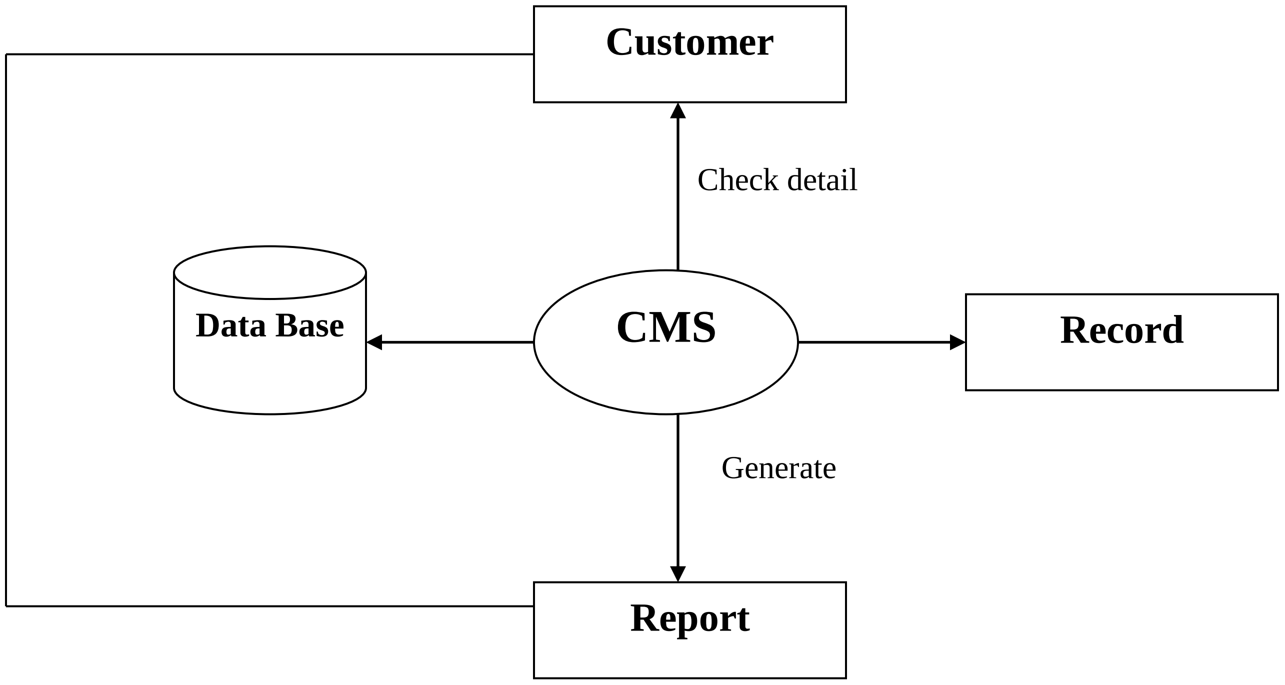
I) DFD on book shop management system

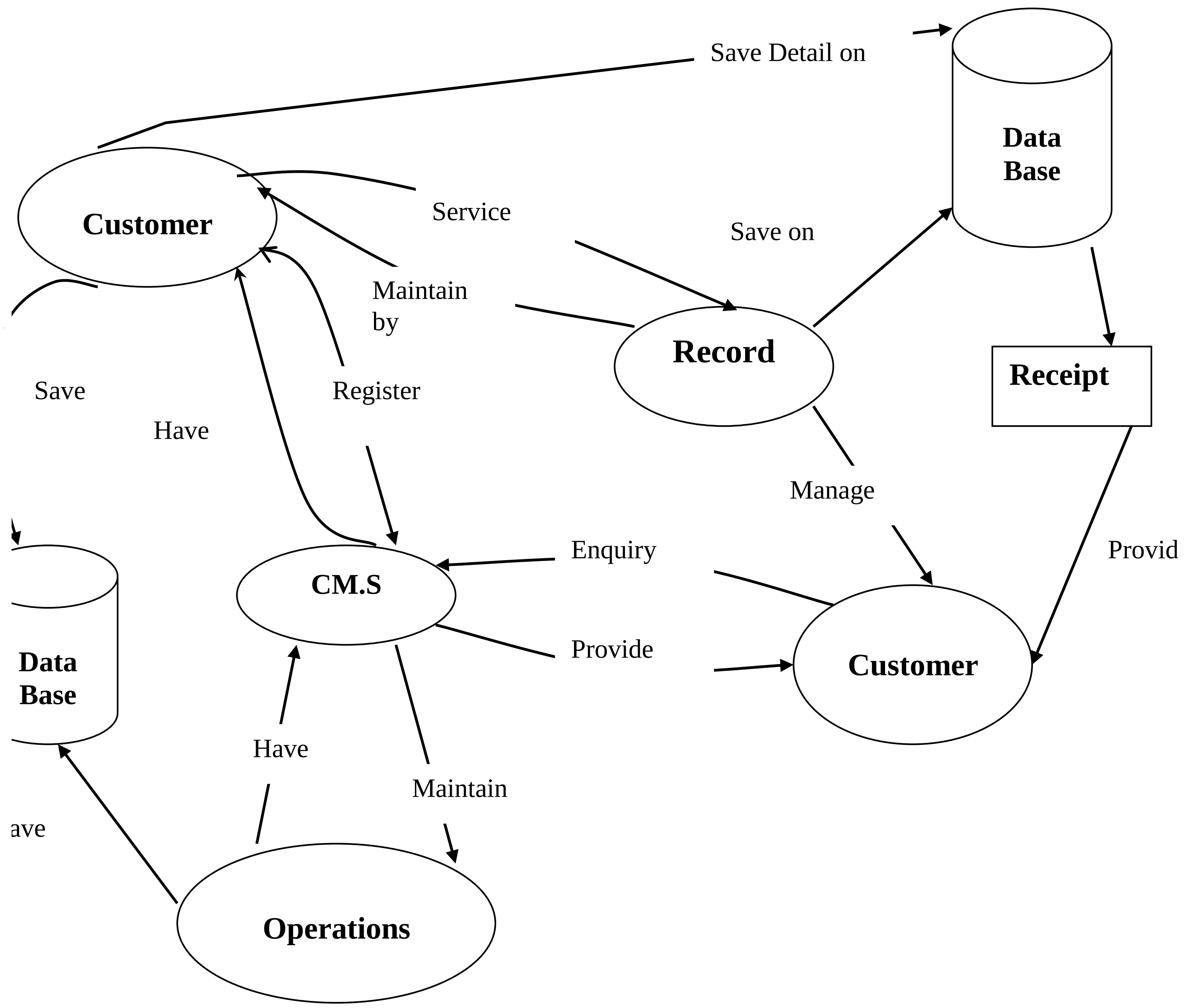


II) DFD ON CUSTOMER TABLE



III) DFD ON BOOKS STOCK:




Save Detail on
Data Base
Customer
Service
Save on
Maintain by
Record
Receipt
Save
Register
Have
Manage
Enquiry
Provid
CM.S
Provide
Customer
Data Base
Have
Maintain
ave
Operations

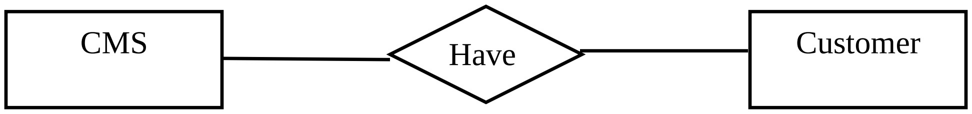
ER Diagram

I) Entity of book Shop Management System:

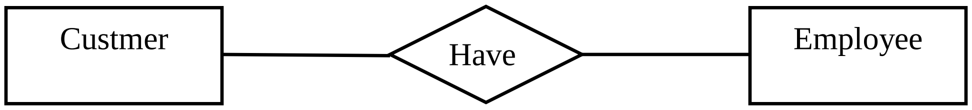
First Name, Last Name, Gender, Age, Address, Contact

III) Entity and their relationship

1) Relationship between Admin & Customer



2) Relationship between Office & Employee



Data Structure

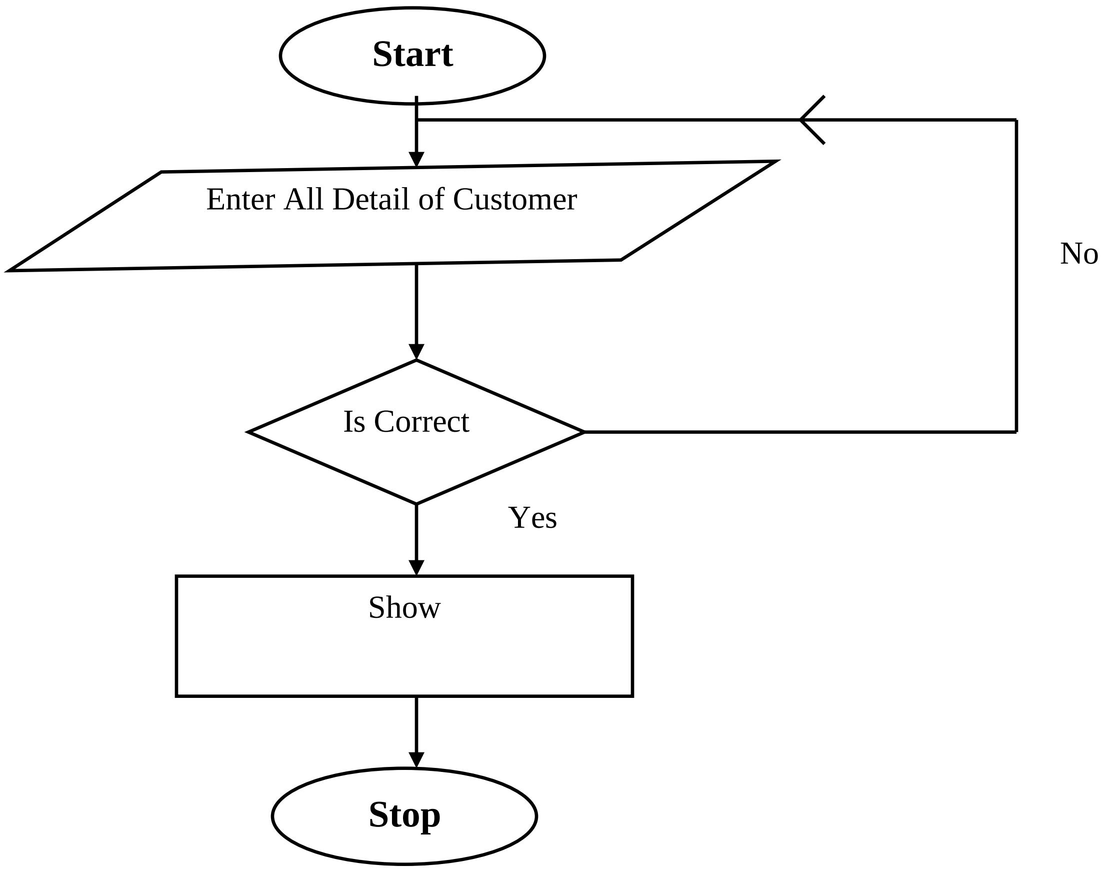
Customer Module:

This table contains employee related information

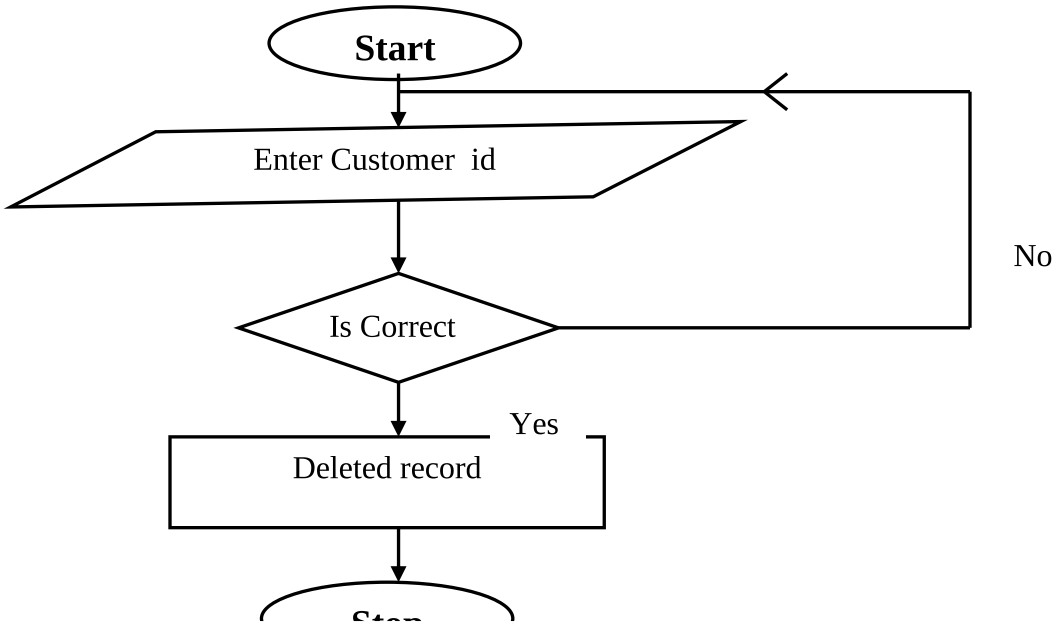
Attribute	Data types	Constraint
id	Integer	Primnary key
FirstName	String	Not Null
LastName	String	Not Null
Gender	String	Not Null
Age	Integer	Not Null
Address	String	Not Null
Contact	String	Not Null

Process Logic

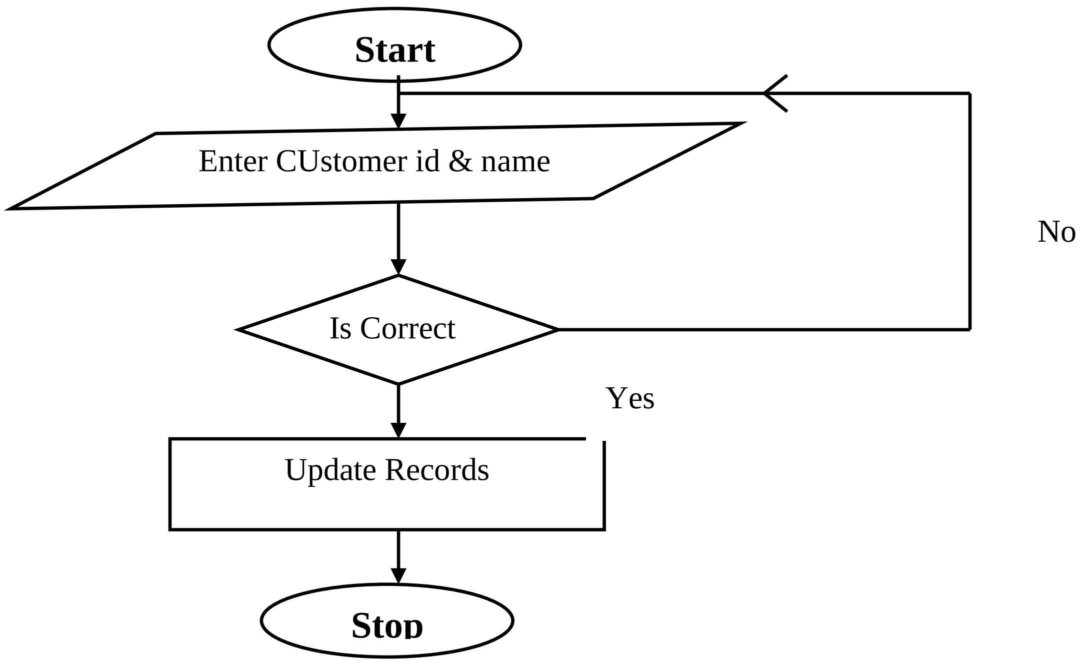
Logic For Management System



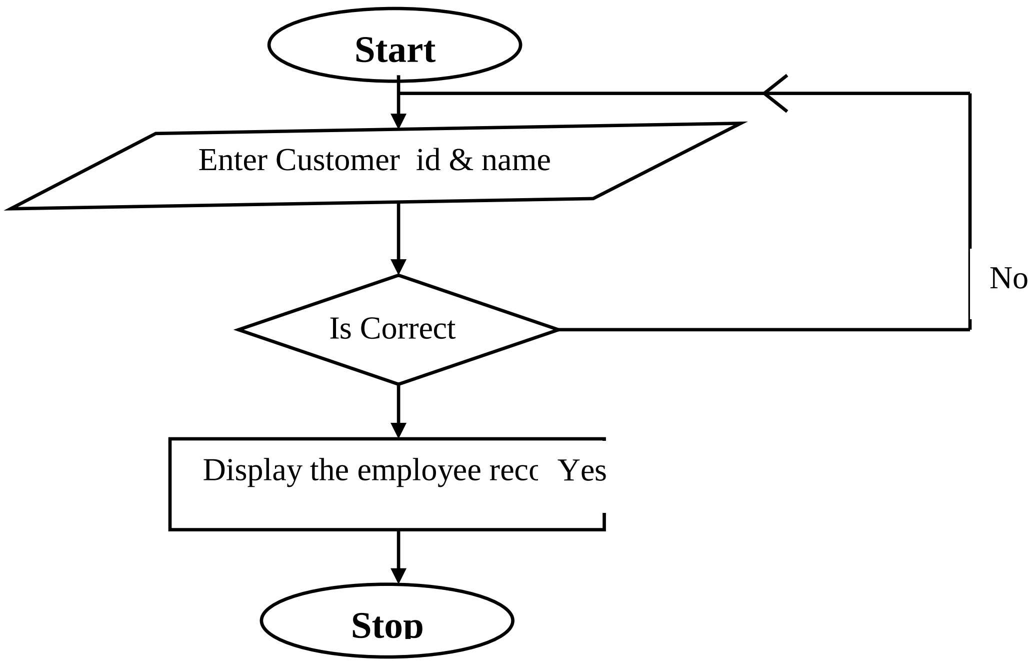
Logic for Delete Record form :



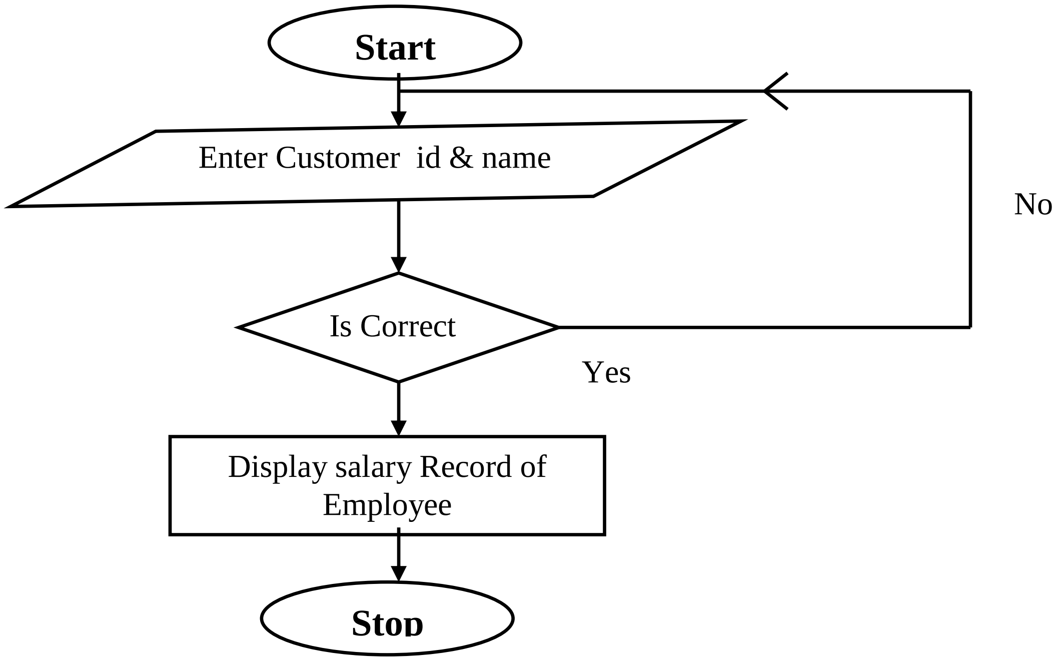
Logic for update Record form :



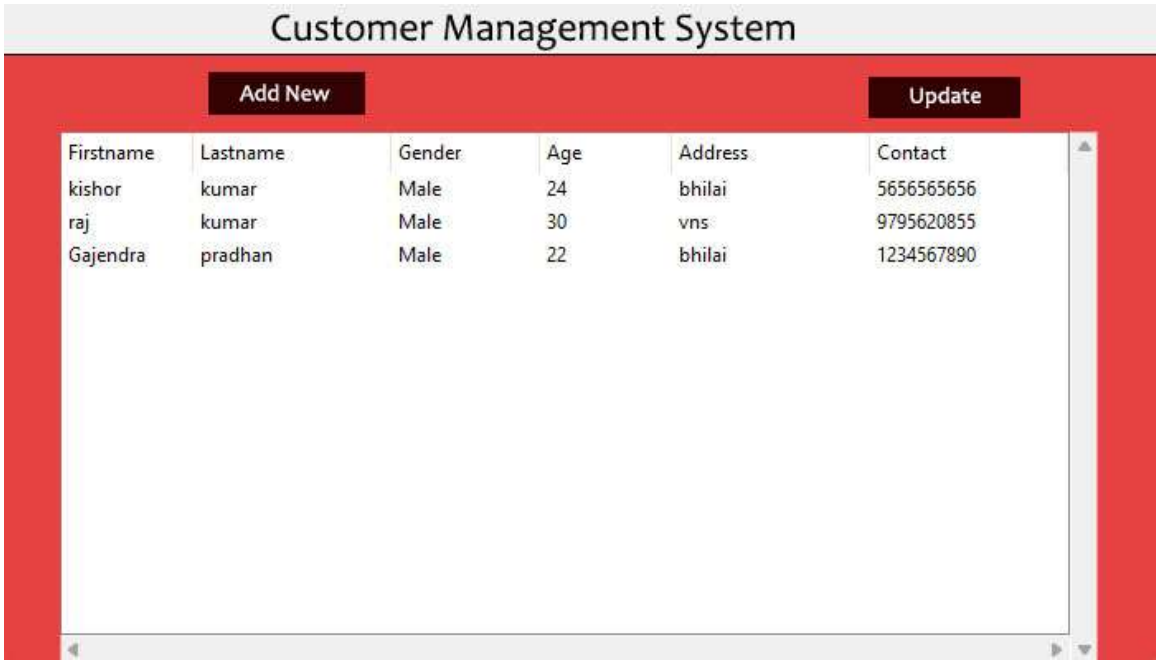
Logic for find Record form:



Logic for submit Employee salary details :

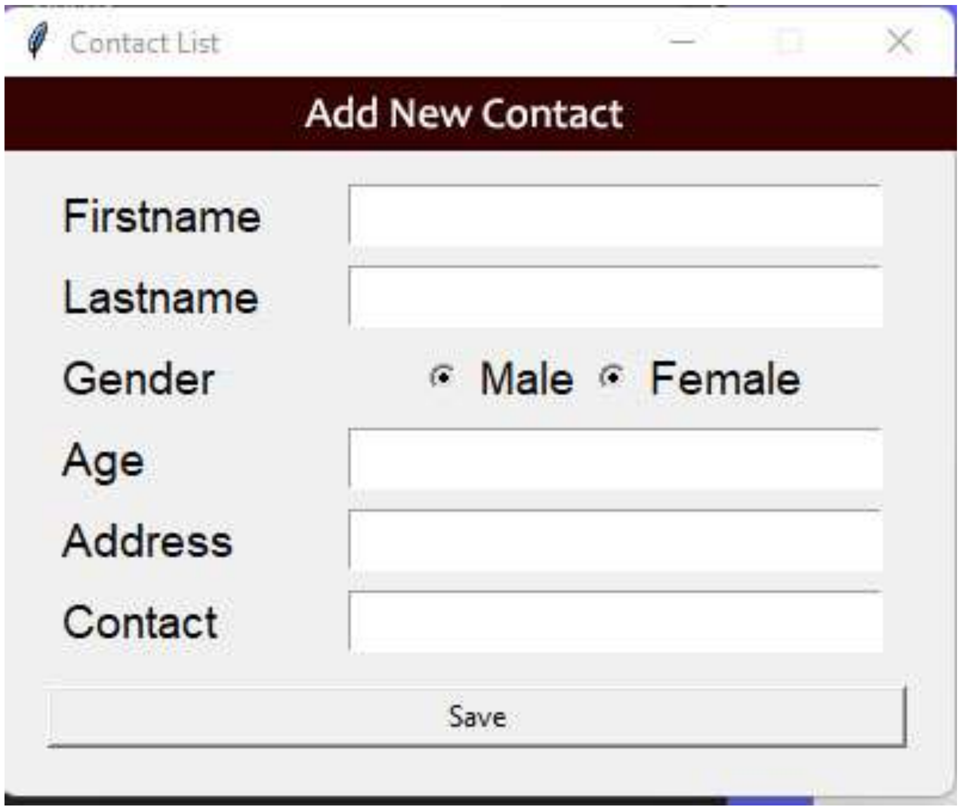


Output



Customer Management System

Firstname	Lastname	Gender	Age	Address	Contact
kishor	kumar	Male	24	bhilai	5656565656
raj	kumar	Male	30	vns	9795620855
Gajendra	pradhan	Male	22	bhilai	1234567890

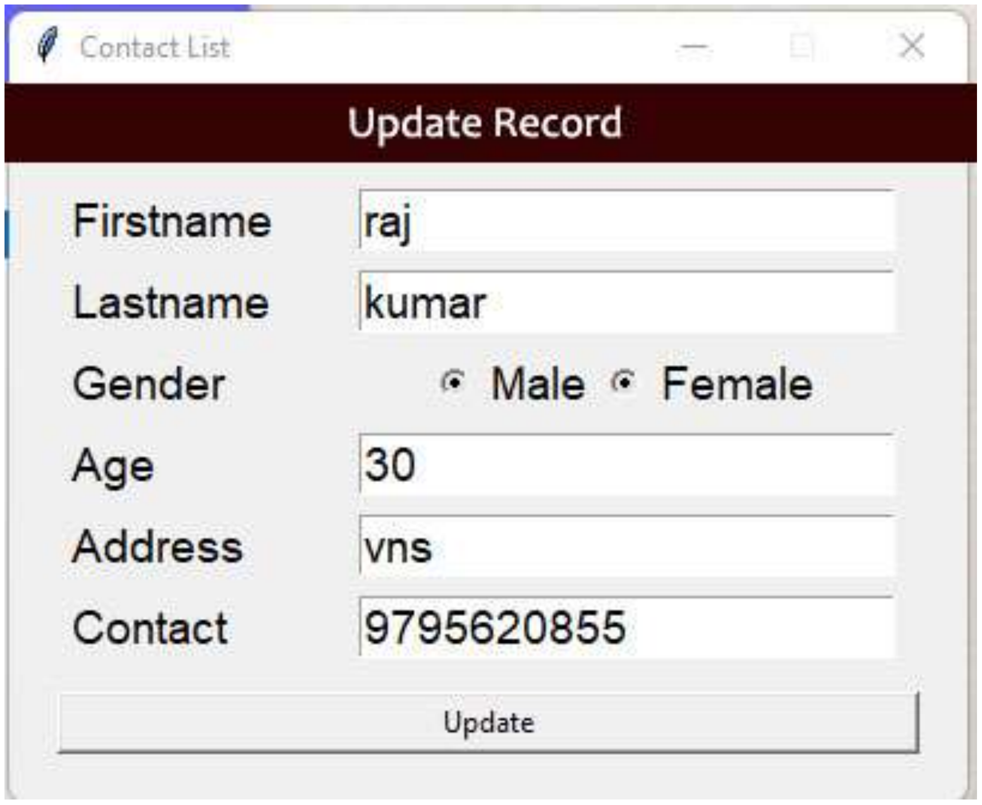


Contact List

Add New Contact

Firstname	
Lastname	
Gender	Male Female
Age	
Address	
Contact	

Save



Update Record

Firstname	raj
Lastname	kumar
Gender	● Male ● Female
Age	30
Address	vns
Contact	9795620855

Coding

```
from tkinter import *
import sqlite3
import tkinter.ttk as ttk
import tkinter.messagebox as tkMessageBox

#DEVELOPED BY Mark Arvin
root = Tk()
root.title("Contact List")
width = 700
height = 400
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x = (screen_width/2) - (width/2)
y = (screen_height/2) - (height/2)
root.geometry("%dx%d+%d+%d" % (width, height, x, y))
root.resizable(0, 0)
root.config(bg="#6666ff")

#==============================VARIABLES=========
===========================
FIRSTNAME = StringVar()
LASTNAME = StringVar()
GENDER = StringVar()
AGE = StringVar()
ADDRESS = StringVar()
CONTACT = StringVar()



#==============================METHODS==========
============================
def Database():
    conn = sqlite3.connect("pythontut.db")
    cursor = conn.cursor()
    cursor.execute("CREATE TABLE IF NOT EXISTS `member`
(mem_id INTEGER NOT NULL  PRIMARY KEY
AUTOINCREMENT, firstname TEXT, lastname TEXT, gender
TEXT, age TEXT, address TEXT, contact TEXT)")
    cursor.execute("SELECT * FROM `member` ORDER BY
`lastname` ASC")
    fetch = cursor.fetchall()
    for data in fetch:
        tree.insert('', 'end', values=(data))
    cursor.close()
    conn.close()

def SubmitData():
    if  FIRSTNAME.get() == "" or LASTNAME.get() == "" or
GENDER.get() == "" or AGE.get() == "" or ADDRESS.get() ==
"" or CONTACT.get() == "":
        result = tkMessageBox.showwarning('', 'Please Complete The
Required Field', icon="warning")
    else:
        tree.delete(*tree.get_children())
        conn = sqlite3.connect("pythontut.db")
        cursor = conn.cursor()
        cursor.execute("INSERT INTO `member` (firstname,
lastname, gender, age, address, contact) VALUES(?, ?, ?, ?, ?, ?)",
(str(FIRSTNAME.get()), str(LASTNAME.get()),
str(GENDER.get()), int(AGE.get()), str(ADDRESS.get()),
str(CONTACT.get())))
        conn.commit()
        cursor.execute("SELECT * FROM `member` ORDER BY
`lastname` ASC")
        fetch = cursor.fetchall()
        for data in fetch:
            tree.insert('', 'end', values=(data))
        cursor.close()
        conn.close()
        FIRSTNAME.set("")
        LASTNAME.set("")
        GENDER.set("")
        AGE.set("")
        ADDRESS.set("")
        CONTACT.set("")

def UpdateData():
    if GENDER.get() == "":
       result = tkMessageBox.showwarning('', 'Please Complete The
Required Field', icon="warning")
    else:
        tree.delete(*tree.get_children())
        conn = sqlite3.connect("pythontut.db")
        cursor = conn.cursor()
        cursor.execute("UPDATE `member` SET `firstname` = ?,
`lastname` = ?, `gender` =?, `age` = ?,  `address` = ?, `contact` = ?
WHERE `mem_id` = ?", (str(FIRSTNAME.get()),
str(LASTNAME.get()), str(GENDER.get()), str(AGE.get()),
str(ADDRESS.get()), str(CONTACT.get()), int(mem_id)))
        conn.commit()
        cursor.execute("SELECT * FROM `member` ORDER BY
`lastname` ASC")
        fetch = cursor.fetchall()
        for data in fetch:
            tree.insert('', 'end', values=(data))
        cursor.close()
        conn.close()
        FIRSTNAME.set("")
        LASTNAME.set("")
        GENDER.set("")
        AGE.set("")
        ADDRESS.set("")
        CONTACT.set("")


def OnSelected(event):
    global mem_id, UpdateWindow
    curItem = tree.focus()
    contents =(tree.item(curItem))
    selecteditem = contents['values']
    mem_id = selecteditem[0]
    FIRSTNAME.set("")
    LASTNAME.set("")
    GENDER.set("")
    AGE.set("")
    ADDRESS.set("")
    CONTACT.set("")
    FIRSTNAME.set(selecteditem[1])
    LASTNAME.set(selecteditem[2])
    AGE.set(selecteditem[4])
    ADDRESS.set(selecteditem[5])
    CONTACT.set(selecteditem[6])
    UpdateWindow = Toplevel()
    UpdateWindow.title("Contact List")
    width = 400
    height = 300
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    x = ((screen_width/2) + 450) - (width/2)
    y = ((screen_height/2) + 20) - (height/2)
    UpdateWindow.resizable(0, 0)
    UpdateWindow.geometry("%dx%d+%d+%d" % (width, height,
x, y))
    if 'NewWindow' in globals():
        NewWindow.destroy()
```

```
#===================FRAMES=====================
==========
    FormTitle = Frame(UpdateWindow)
    FormTitle.pack(side=TOP)
    ContactForm = Frame(UpdateWindow)
    ContactForm.pack(side=TOP, pady=10)
    RadioGroup = Frame(ContactForm)
    Male = Radiobutton(RadioGroup, text="Male",
variable=GENDER, value="Male",  font=('arial',
14)).pack(side=LEFT)
    Female = Radiobutton(RadioGroup, text="Female",
variable=GENDER, value="Female",  font=('arial',
14)).pack(side=LEFT)


#===================LABELS=====================
=========
    lbl_title = Label(FormTitle, text="Updating Contacts",
font=('arial', 16), bg="orange",  width = 300)
    lbl_title.pack(fill=X)
    lbl_firstname = Label(ContactForm, text="Firstname",
font=('arial', 14), bd=5)
    lbl_firstname.grid(row=0, sticky=W)
    lbl_lastname = Label(ContactForm, text="Lastname",
font=('arial', 14), bd=5)
    lbl_lastname.grid(row=1, sticky=W)
    lbl_gender = Label(ContactForm, text="Gender", font=('arial',
14), bd=5)
    lbl_gender.grid(row=2, sticky=W)
    lbl_age = Label(ContactForm, text="Age", font=('arial', 14),
bd=5)
    lbl_age.grid(row=3, sticky=W)
    lbl_address = Label(ContactForm, text="Address", font=('arial',
14), bd=5)
    lbl_address.grid(row=4, sticky=W)
    lbl_contact = Label(ContactForm, text="Contact", font=('arial',
14), bd=5)
    lbl_contact.grid(row=5, sticky=W)


#===================ENTRY======================
=========
    firstname = Entry(ContactForm, textvariable=FIRSTNAME,
font=('arial', 14))
    firstname.grid(row=0, column=1)
    lastname = Entry(ContactForm, textvariable=LASTNAME,
font=('arial', 14))
    lastname.grid(row=1, column=1)
    RadioGroup.grid(row=2, column=1)
    age = Entry(ContactForm, textvariable=AGE,  font=('arial', 14))
    age.grid(row=3, column=1)
    address = Entry(ContactForm, textvariable=ADDRESS,
font=('arial', 14))
    address.grid(row=4, column=1)
    contact = Entry(ContactForm, textvariable=CONTACT,
font=('arial', 14))
    contact.grid(row=5, column=1)



#==================BUTTONS====================
==========
    btn_updatecon = Button(ContactForm, text="Update",
width=50, command=UpdateData)
    btn_updatecon.grid(row=6, columnspan=2, pady=10)
```

```
#fn1353p
def DeleteData():
    if not tree.selection():
       result = tkMessageBox.showwarning('', 'Please Select
Something First!', icon="warning")
    else:
        result = tkMessageBox.askquestion('', 'Are you sure you want
to delete this record?', icon="warning")
        if result == 'yes':
            curItem = tree.focus()
            contents =(tree.item(curItem))
            selecteditem = contents['values']
            tree.delete(curItem)
            conn = sqlite3.connect("pythontut.db")
            cursor = conn.cursor()
            cursor.execute("DELETE FROM `member` WHERE
`mem_id` = %d" % selecteditem[0])
            conn.commit()
            cursor.close()
            conn.close()

def AddNewWindow():
    global NewWindow
    FIRSTNAME.set("")
    LASTNAME.set("")
    GENDER.set("")
    AGE.set("")
    ADDRESS.set("")
    CONTACT.set("")
    NewWindow = Toplevel()
    NewWindow.title("Contact List")
    width = 400
    height = 300
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()
    x = ((screen_width/2) - 455) - (width/2)
    y = ((screen_height/2) + 20) - (height/2)
    NewWindow.resizable(0, 0)
    NewWindow.geometry("%dx%d+%d+%d" % (width, height, x,
y))
    if 'UpdateWindow' in globals():
        UpdateWindow.destroy()


#===================FRAMES=====================
==========
    FormTitle = Frame(NewWindow)
    FormTitle.pack(side=TOP)
    ContactForm = Frame(NewWindow)
    ContactForm.pack(side=TOP, pady=10)
    RadioGroup = Frame(ContactForm)
    Male = Radiobutton(RadioGroup, text="Male",
variable=GENDER, value="Male",  font=('arial',
14)).pack(side=LEFT)
    Female = Radiobutton(RadioGroup, text="Female",
variable=GENDER, value="Female",  font=('arial',
14)).pack(side=LEFT)


#===================LABELS=====================
=========
    lbl_title = Label(FormTitle, text="Adding New Contacts",
font=('arial', 16), bg="#66ff66",  width = 300)
    lbl_title.pack(fill=X)
    lbl_firstname = Label(ContactForm, text="Firstname",
font=('arial', 14), bd=5)
    lbl_firstname.grid(row=0, sticky=W)
    lbl_lastname = Label(ContactForm, text="Lastname",
font=('arial', 14), bd=5)
    lbl_lastname.grid(row=1, sticky=W)
    lbl_gender = Label(ContactForm, text="Gender", font=('arial',
14), bd=5)
    lbl_gender.grid(row=2, sticky=W)
    lbl_age = Label(ContactForm, text="Age", font=('arial', 14),
bd=5)
    lbl_age.grid(row=3, sticky=W)
    lbl_address = Label(ContactForm, text="Address", font=('arial',
14), bd=5)
    lbl_address.grid(row=4, sticky=W)
    lbl_contact = Label(ContactForm, text="Contact", font=('arial',
14), bd=5)
    lbl_contact.grid(row=5, sticky=W)


#===================ENTRY======================
=========
    firstname = Entry(ContactForm, textvariable=FIRSTNAME,
font=('arial', 14))
    firstname.grid(row=0, column=1)
    lastname = Entry(ContactForm, textvariable=LASTNAME,
font=('arial', 14))
    lastname.grid(row=1, column=1)
    RadioGroup.grid(row=2, column=1)
    age = Entry(ContactForm, textvariable=AGE,  font=('arial', 14))
    age.grid(row=3, column=1)
    address = Entry(ContactForm, textvariable=ADDRESS,
font=('arial', 14))
    address.grid(row=4, column=1)
    contact = Entry(ContactForm, textvariable=CONTACT,
font=('arial', 14))
    contact.grid(row=5, column=1)
```

```
#==================BUTTONS====================
==========
    btn_addcon = Button(ContactForm, text="Save", width=50,
command=SubmitData)
    btn_addcon.grid(row=6, columnspan=2, pady=10)




#============================FRAMES===========
============================
Top = Frame(root, width=500, bd=1, relief=SOLID)
Top.pack(side=TOP)
Mid = Frame(root, width=500,  bg="#6666ff")
Mid.pack(side=TOP)
MidLeft = Frame(Mid, width=100)
MidLeft.pack(side=LEFT, pady=10)
MidLeftPadding = Frame(Mid, width=370, bg="#6666ff")
MidLeftPadding.pack(side=LEFT)
MidRight = Frame(Mid, width=100)
MidRight.pack(side=RIGHT, pady=10)
TableMargin = Frame(root, width=500)
TableMargin.pack(side=TOP)
#============================LABELS===========
==========================
lbl_title = Label(Top, text="Contact Management System",
font=('arial', 16), width=500)
lbl_title.pack(fill=X)

#============================ENTRY============
==========================
#=============================BUTTONS==========
===========================
btn_add = Button(MidLeft, text="+ ADD NEW", bg="#66ff66",
command=AddNewWindow)
btn_add.pack()
btn_delete = Button(MidRight, text="DELETE", bg="red",
command=DeleteData)
btn_delete.pack(side=RIGHT)

#=============================TABLES============
===========================
scrollbarx = Scrollbar(TableMargin, orient=HORIZONTAL)
scrollbary = Scrollbar(TableMargin, orient=VERTICAL)
tree = ttk.Treeview(TableMargin, columns=("MemberID",
"Firstname", "Lastname", "Gender", "Age", "Address", "Contact"),
height=400, selectmode="extended",
yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
scrollbary.config(command=tree.yview)
scrollbary.pack(side=RIGHT, fill=Y)
scrollbarx.config(command=tree.xview)
scrollbarx.pack(side=BOTTOM, fill=X)
tree.heading('MemberID', text="MemberID", anchor=W)
tree.heading('Firstname', text="Firstname", anchor=W)
tree.heading('Lastname', text="Lastname", anchor=W)
tree.heading('Gender', text="Gender", anchor=W)
tree.heading('Age', text="Age", anchor=W)
tree.heading('Address', text="Address", anchor=W)
tree.heading('Contact', text="Contact", anchor=W)
tree.column('#0', stretch=NO, minwidth=0, width=0)
tree.column('#1', stretch=NO, minwidth=0, width=0)
tree.column('#2', stretch=NO, minwidth=0, width=80)
tree.column('#3', stretch=NO, minwidth=0, width=120)
tree.column('#4', stretch=NO, minwidth=0, width=90)
tree.column('#5', stretch=NO, minwidth=0, width=80)
tree.column('#6', stretch=NO, minwidth=0, width=120)
tree.column('#7', stretch=NO, minwidth=0, width=120)
tree.pack()
tree.bind('<Double-Button-1>', OnSelected)

#=============================INITIALIZATION=====
==========================
if __name__ == '__main__':
    Database()
    root.mainloop()
```

Further Modification

The project is implemented in a very short span of time. There can be many extensions, which can be made to the existing project. Some of the extension, which we thought doing, are summarized below:-

If any body wishes to continue this project further, he or she use the suggestion made below as guideline for start of some the extension.

- We can give the facility of More feature for Customer Management
- We can give the facility of Online Management.
- We can give the facility of to attach this system to other oraganization.

There are so many facilities, which are necessary for any Shop, we can include in this project.

Bibliography

With tremendous advance in software technology, it is very essential for software to be up to date user interactive keeping this fact in view, I have tried my best to keep it to such level as of today's so no make it to this level, I had to consult many books and media.
These are the list of books and media, I have consulted: -

- https://docs.python.org/3/tutorial/index.html
- **https://www.w3schools.com/python/**
- **https://www.tutorialspoint.com/python/index.htm**
- https://www.pythontutorial.ne