**Bapuji Salunkhe Polytechnic, Kolhapur**

**Department Of Computer Science**

**Micro:**

Project Report

**Subject:**

Python

**Title:**

CONTACT MANAGEMENT SYSTEM

**Submitted by:**

|  |  |  |
| --- | --- | --- |
| **ROLL NO.** | **STUDENT NAME** | **Enrollment No.** |
| 203371 | Kanishka Desai | 1900430004 |
| 203373 | Pranjal Haval | 1900430009 |
| 203374 | Adinath Sangaj | 1900430011 |

* **Introduction of micro project:**

A contact manager is a software program that enables users to easily store and find contact information, such as names, addresses and telephone numbers, gender, age. They are contact-centric databases that provide a fully integrated approach to tracking of all information and communication activities linked to contacts.

In management terminology, advanced contact managers can be called individual resource management (IRM) or contact management (CM) tools – systems for managing an individual's interactions with current and future contacts, to organize, collaborate, and synchronize health, lifestyle, and financial needs.

The main aim behind the development of an automated system of contact management is to assist any organization to store and retrieve all the information about a pre-existing customer in a way more robust and efficient way.

All the information about a particular customer is stored in a retrievable manner. The fascinating fact about contact management is that all the information about the contact can be made available over every single terminal at an affordable cost.

* **Aim of micro project:**

1. The main aim behind the development of an automated system of contact management
2. Create all the information about a particular customer is stored in a retrievable manner.

* **Intended course outcome:**

1. Display message on screen using python script on IDE.
2. Develop python program to demonstrate use of operators.
3. Perform operations on dada structure in python.
4. Develop functions for given problem.
5. Design classes for given problem.
6. Handle exceptions.

* **literature Review**

TO build our project we took the from following links and books such as:

1: Python book we can refer and name of this book is tech max.

2: We can visit the following websites

A] [www.1000projects.com](http://www.1000projects.com)

In this website, I am collect the various project about the python . Here available various information about the Patient-information-system.

* **Proposed methodology**

The ‘Contact Management’ is the sample project for managing contact details. The project aims at developing contact management system using the python language that enables an organization to maintain its library. The application uses basic python function to generate menus, show message boxes and print text on the screen. To display customized text with colours and fonts according to application requirements, functions have been created in the application, which fetch the exact video memory addresses of a target location, to write text at a particular location. The application also implements the concept of structures to define the contact items.

* **Database design**

A relational database design was used to design the database. A relational database management system (RDBMS) is an excellent tool for organizing large amount of data and defining the relationship between the datasets in a consistent and understandable way.

A RDBMS provides a structure which is flexible enough to accommodate almost any kind of data. Relationships between the tables were defined by creating special columns (keys), which contain the same set of values in each table. The tables can be joined in different combinations to extract the needed data. A RDBMS also offered flexibility that enabled redesign and regeneration of reports from the database without need to re-enter the data.

Data dictionaries were used to provide definitions of the data used; these included the final data structures for the various tables and their corresponding data fields, description and sizes the user application programs and interface were developed using MYSQL.

SQL is a language used to create, manipulate, examine and manage relational databases. SQL was standardized in 1992 so that a program could communicate with most database systems without having to change the SQL commands. Unfortunately one must connect to the database before sending SQL commands and each database vendor has a different interface as well as different extensions of SQL. Though SQL is well suited for manipulating database, it is unsuitable as a general application language and programmers use it primarily as a means of communicating with databases, another language is needed to feed SQL statements to a database and process results for visual display or report generation.

* **System Requirement**
* **Hardware Requirement**

This section describes the hardware components and software requirements needed for effective and efficient running of the system

|  |  |
| --- | --- |
| **Hardware** | **Minimum system requirement** |
| Processor | 2.4 GHZ processor speed |
| Memory | 128 MB RAM (256 MB Recommended) |
| Disk Space | 80 GB (including 20 GB for database management system) |
| Display | 800 x 600 colours (1024 x 768 High colour- 16 bit Recommended ) |

* **Software Requirement**

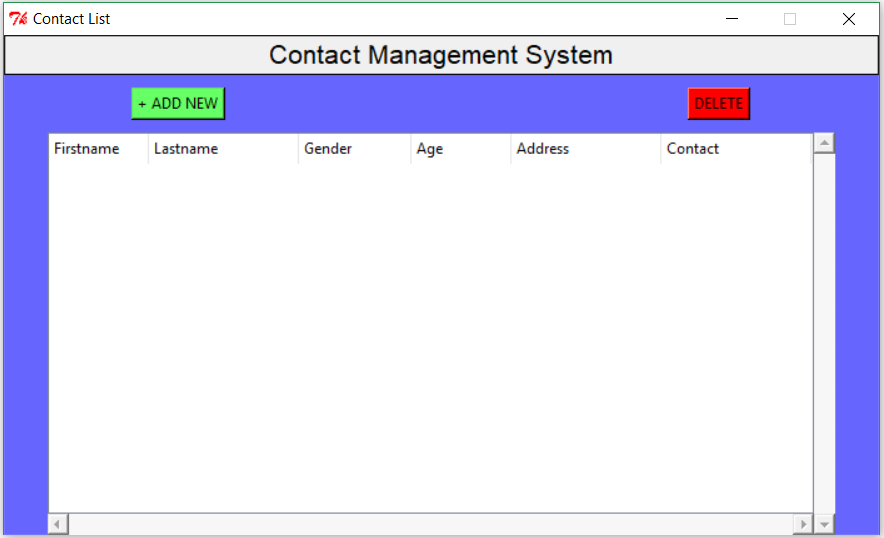
The table above shows software requirements recommended to enable the system to run as required for using PIMS

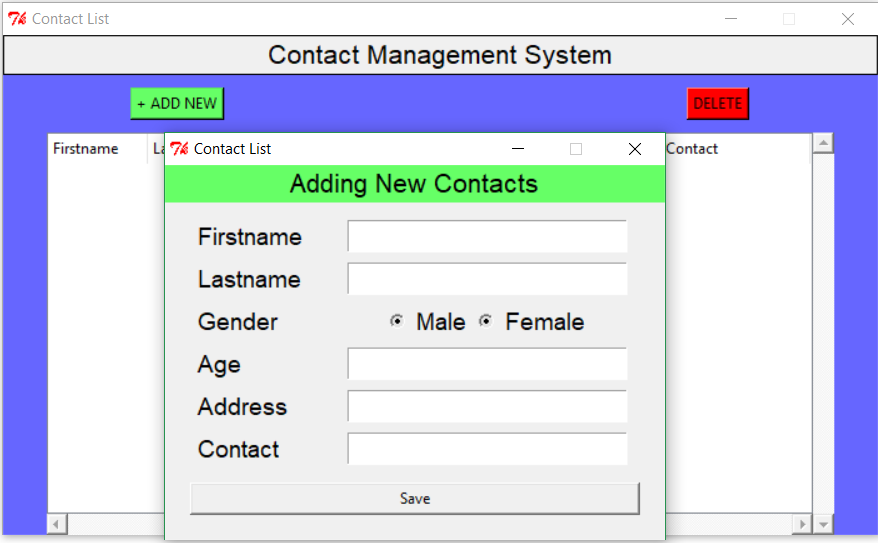
|  |  |
| --- | --- |
| **Software** | **Minimum system requirement** |
| Operating System | Windowa2000 or letter |
| Database Management System | MYSQL, |
| Run-time Environment | Apache/ tomcat 5 server |

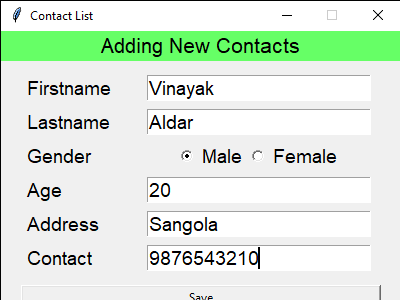
* **Coding:**
* **Python Code:**

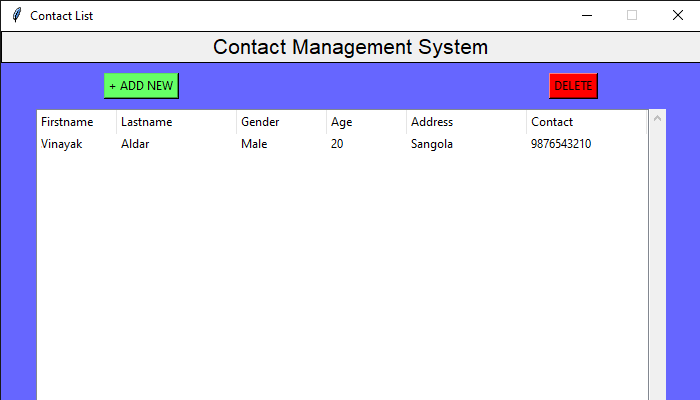
**from** tkinter**import** \*  
**import** sqlite3  
**import** tkinter.ttk**as** ttk  
**import** tkinter.messagebox**as** tkMessageBox  
  
root = Tk()  
root.title(**"Contact List"**)  
width = 700  
height = 400  
screen\_width = root.winfo\_screenwidth()  
screen\_height = root.winfo\_screenheight()  
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"**)  
  
FIRSTNAME = StringVar()  
LASTNAME = StringVar()  
GENDER = StringVar()  
AGE = StringVar()  
ADDRESS = StringVar()  
CONTACT = StringVar()  
  
  
**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()  
  
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)  
  
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)  
  
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)  
  
  
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()  
  
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)  
  
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)  
  
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)  
  
  
btn\_addcon = Button(ContactForm, text=**"Save"**, width=50, command=SubmitData)  
btn\_addcon.grid(row=6, columnspan=2, pady=10)  
  
  
  
  
  
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)  
lbl\_title = Label(Top, text=**"Contact Management System"**, font=(**'arial'**, 16), width=500)  
lbl\_title.pack(fill=X)  
  
  
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)  
  
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)  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 Database()  
root.mainloop()

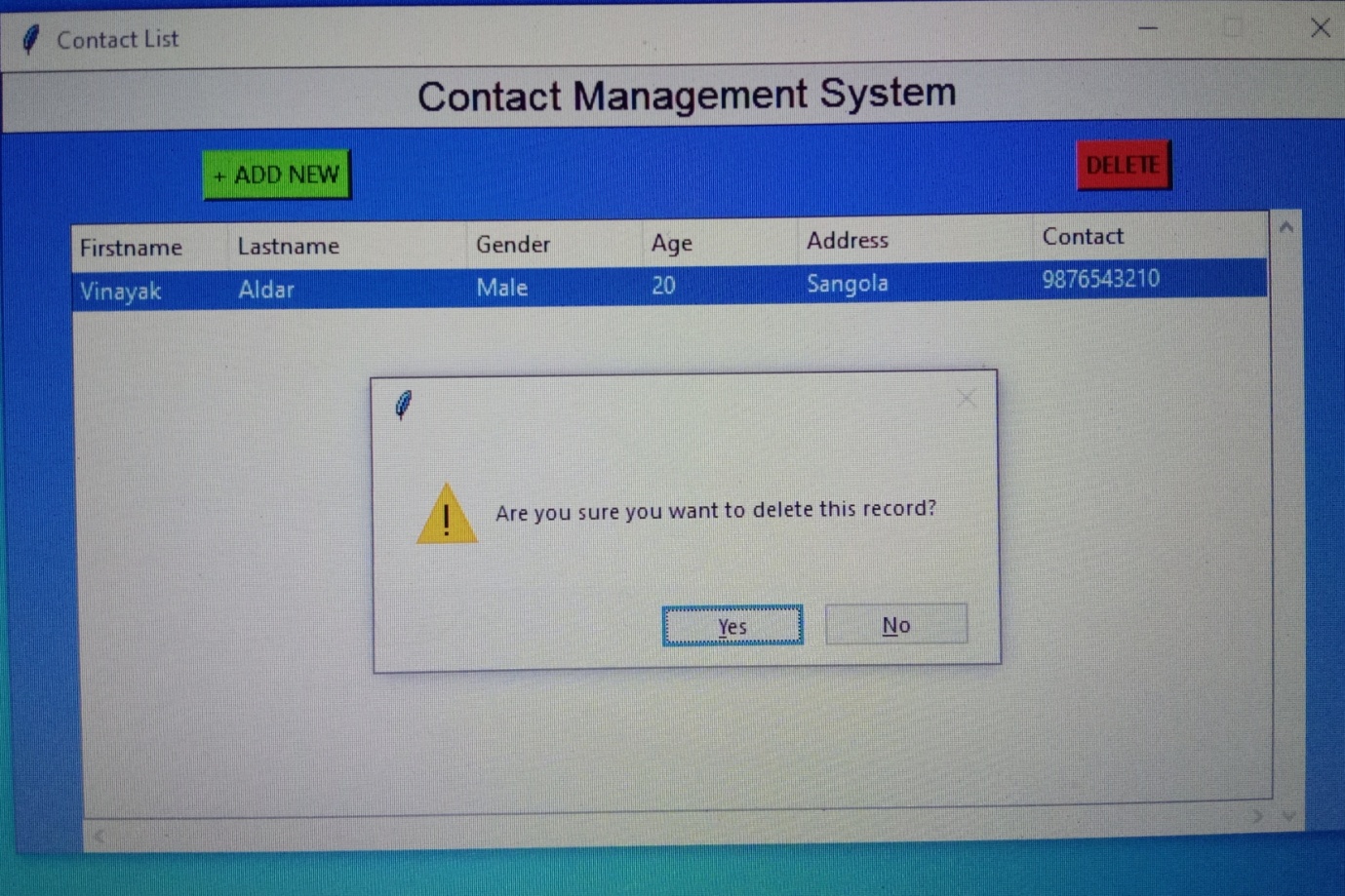
**Outputs:-**

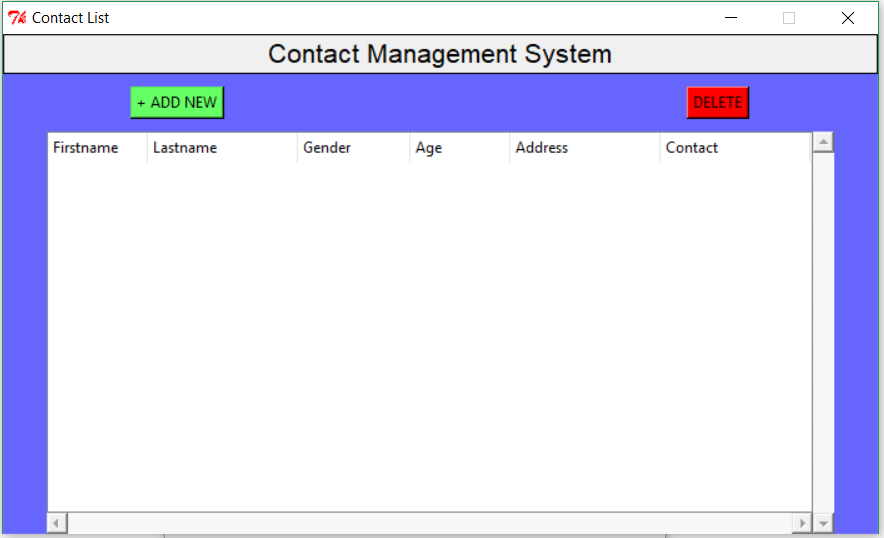
****

****

****

****



****

* **Resources Used:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.no** | **Name of Resource/material** | **Specifications** | **Qty.** | **Remarks** |
| 1 | **Computer system** | AMD E1 processor RAM 4GB 1TB HDD | 1 |  |
| 2 | **Software Used** | Google Chrome, MS world 2016 | 1 |  |
| 3 | **Computer** | IP Address | 1 |  |
| 4 | **Other Resources** | Printer | 1 |  |

* **Skill development**

1. We understand how to write a python program.
2. We understand how to build database.
3. Teamwork.

* **Application of micro project**

1. From this micro-project we understand how to create contact management system.

* **Area of future improvement**
* In future this micro-project will help in contact management system .