# CONTENTS

# SYNOPSIS

The objective of this program is to create a GUI based Customer Management System. To build this, you will need intermediate understanding of Tkinter library and MySQL messagebox, Ttk modules and CSV module in python.

Our project Customer Management system includes registration of customer, storing their details into the system i.e. computerized the process.

Our software has the facility to give a unique id for every customer and stores the details of every customer. It includes a search facility also – search by name and email.

The data can be retrieved easily. The interface is very User-friendly. The data are well protected for personal use and makes the data processing very fast.

Customer Management System is software which is helpful for markets as well as the shops. In the current system all the activities are done manually.

It is very time consuming and costly. Our Customer Management System deals with the various activities related to managing customer records.

Our Objective is computerizing the process of customer records management.

This program uses python as main language and used for manage Customer details. This program saves the Customer id, Name, Email id, Age.

We can Create, Update and Delete the details in the list, by entering the Customer id we can Delete and Modify them.

This program needs python v3.7 or above to run the tkinter module properly that is used for creating GUI programs.

By importing the mysql.connector module we can use the MySQL workbench to store the data in this program.

We used CSV options to import and export data in the database and manipulate them. And we can save the imported Data from CSV to MySQL Database.

## INTRODUCTION

This project is aimed to automate the customer management system. This project is developed mainly to administrate customer records. The purpose of the project entitled as CUSTOMER MANAGEMENT SYSTEM is to computerize the Front Office Management of customer records in shops, to develop software which is user friendly, simple, fast, and cost – effective. Traditionally, it was done manually. The main function of the system is to register and store customer details, retrieve these details as and when required, and also to manipulate these details meaningfully.

## SCOPE

The proposed software product is the Customer Management System. The system will be used in any shops to get the information from the customer and then storing that data for future usage. The current system in use is a paper-based system. It is too slow and cannot provide updated lists of customers with in a reasonable timeframe. The intentions of the system are to reduce over- time pay and increase the productivity. Requirements statements in this document are both functional and non-functional.

## HARDWARE REQUIREMENT

- Pentium IV processor or higher / AMD
- 512 MB RAM (or above)
- 40 GB or more Hard Disk
- Mouse and Keyboard

## SOFTWARE REQUIREMENT

- OS-Windows 7/8/10 or Linux
- Python Interpreter
- Pycharm IDE or VS code
- Mysql Workbench

## BENEFITS

- ➢ Software provides easy management of customer records.
- ➢ Software has very user friendly interface which is very easy to handle and understand.
- ➢ Software provides security to private data by hiding them.
- ➢ Software uses very less memory and takes less time to startup.

## LIMITATIONS

- Software is limited to Desktop only.
- System requires python interpreter installed on the system.
- All options of customer management are not included in current version.
- Security options provide only low level security against beginner attackers.
- GUI is in English only.

## Project Prerequisites:

To build this project, we will need the following libraries:

**1. Tkinter** – To create the GUI.
**2. MySql** – To connect the program to the database and store information in it.
**3. Tkinter.messagebox** – To show a display box, displaying some information or an error.
**4. Tkinter.ttk** – To create the tree where all the information will be displayed.
**5. messagebox –** to show prompts on actions.
**6. CSV –** to import and export date as .csv formate.

## Python

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

## History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

## Python’s Features includes

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax.This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

## Tkinter

Python offers multiple options for developing a GUI (Graphical User Interface). Out of all the GUI methods, Tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with Tkinter outputs the fastest and easiest way to create GUI applications. Creating a GUI using Tkinter is an easy task.

**To create a Tkinter:**

- Importing the module – Tkinter
- Create the main window (container)
- Add any number of widgets to the main window

- Apply the event Trigger on the widgets.

The name Tkinter comes from Tk interface. Tkinter was written by Fredrik Lundh.

## What is Python GUI Programming?

GUI, also known as Graphical User Interfaces, are computer programs that enable users to make use of visuals in order to interact with underlying systems.

A basic example of the same might include our mobile phones, which are equipped with such GUIs that help us to interact with various functionalities through tap, touch, or display. Similarly, Python GUI is a graphical user interface that is written in this programming language.

As we all know, Python, by far, is considered to be one the most popular programming languages due to its several benefits, such as easy readability, and beginner friendliness, among others. Furthermore, this programming language is also equipped with various frameworks that make it great for developing a graphical user interface.

## Python GUI Programming Use Cases:

Mobile Applications- Instagram, Reddit, and Pinterest are some of the best examples of mobile applications that make use of Python GUI.

Games- Several gaming applications, such as Flappy Bird or Mount & Blade, also make use of Python GUI in order to provide a better user experience, such as flashy graphics, and rewarding interactivity.

Human-Machine Interfaces- Last but not least, human-machine interfaces are present across various industries.

Human-machine interfaces, also known as HMI, are basically graphic user interfaces that are responsible for providing an overview of industrial monitoring and control systems.

Furthermore, this also includes providing appropriate solutions in case of any anomalies in the operating system. Python makes the development of these industrial applications much easier and at a reduced cost, which is one of the most important requirements of all businesses.

### What are some of the drawbacks of using Tkinter Python?

Advanced widgets are not included in Tkinter. It doesn’t have a tool like Qt Designer for Tkinter. It doesn’t have a dependable user interface. Tkinter can be difficult to debug at times. It lacks a native appearance and feel.
Tkinter can be useful for creating simple and quick GUIs for Python scripts, but for more complex programming output, almost all programmers prefer the PyQt capabilities.

# Python - Modules

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

## The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file.

The *import* has the following syntax −

```
import module1
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module.

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

## The *from...import* Statement

Python's *from* statement lets you import specific attributes from a module into the current namespace. The *from...import* has the following syntax −

```
from module_name import name1, name2 etc…
```

For example, to import the function messagebox from the module tkinter, use the following statement :

```
from tkinter import messagebox, filedialog
```

This statement does not import the entire module tkinter into the current namespace; it just introduces the item messagebox from the module tkinter into the global symbol table of the importing module.

The *from...import ** Statement

It is also possible to import all names from a module into the current namespace by using the following import statement

## Python def - keyword

Python def keyword is used to define a function, it is placed before a function name that is provided by the user to create a user-defined function. In python, a function is a logical unit of code containing a sequence of statements indented under a name given using the "**def**" keyword. In python def keyword is the most used keyword.

**Syntax:**

```
def function_name:
    function definition statements...
```

**Use of def keyword:**

- In the case of classes, the def keyword is used for defining the methods of a class.
- def keyword is also required to define the special member function of a class like __init__().

**The possible practical application is that it provides the feature of code reusability rather than writing the piece of code again and again we can define a function and write the code inside the function with the help of the def keyword. It will be more clear in the illustrated example given below. There can possibly be many applications of def depending upon the use cases.**

## Python IF...ELIF...ELSE Statements

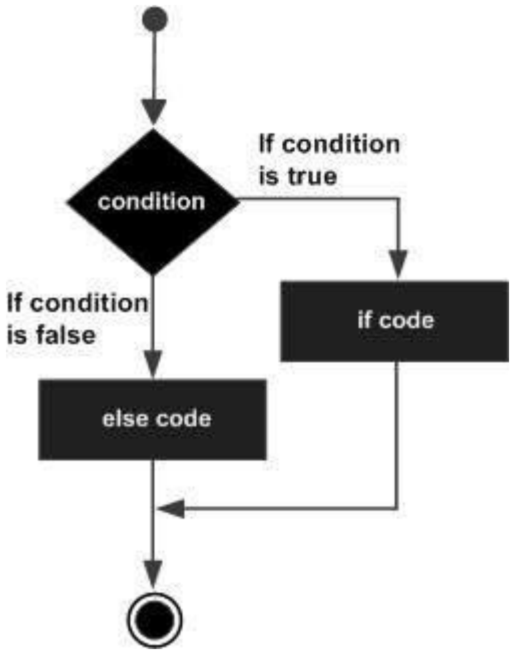
An **else** statement can be combined with an **if** statement. An **else** statement contains the block of code that executes if the conditional expression in the if statement resolves to 0 or a FALSE value.

The *else* statement is an optional statement and there could be at most only one **else** statement following **if**.

**Syntax:**

The syntax of the *if..else* statement is −

```
if expression:
  statement(s)
else:
  statement(s)
```



## The elif Statement

The elif statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

**Similar to the else, the elif statement is optional. However, unlike else, for which there can be at most one statement, there can be an arbitrary number of elif statements following an if.**

**Syntax:**

```
if expression1:
  statement(s)
elif expression2:
  statement(s)
elif expression3:
  statement(s)
else:
  statement(s)
```

## MySQL

Python MySQL Connector is a Python driver that helps to integrate Python and MySQL. This Python MySQL library allows the conversion between Python and MySQL data types. MySQL Connector API is implemented using pure Python and does not require any third-party library.

## Installation

To install the **Python-mysql-connector** module, one must have Python and PIP, preinstalled on their system. If Python and pip are already installed type the below command in the terminal.

```
pip3 install mysql-connector-python
```

## Creating Database

After connecting to the MySQL server let's see how to create a MySQL database using Python. For this, we will first create a cursor() object and will then pass the SQL command as a string to the execute() method.

The SQL command to create a database is –

```
CREATE DATABASE database_name
```

## Creating Tables

For creating tables we will follow the similar approach of writing the SQL commands as strings and then passing it to the execute() method of the cursor object.

SQL command for creating a table is -

```
CREATE TABLE table_name
(
   column_name_1 column_Data_type,
   column_name_2 column_Data_type,
   :
   column_name_n column_Data_type
);
```

## Insert Data into Tables

To insert data into the MySQL table **Insert into** query is used.

**Syntax:**

INSERT INTO table_name (column_names) VALUES (data)

## Fetching Data

We can use the select query on the MySQL tables in the following ways –

- In order to select particular attribute columns from a table, we write the attribute names.

SELECT attr1, attr2 FROM table_name

- In order to select all the attribute columns from a table, we use the asterisk '*' symbol.

SELECT * FROM table_name

## Where Clause

Where clause is used in MySQL database to filter the data as per the condition required. You can fetch, delete or update a particular set of data in MySQL database by using where clause.

**Syntax:**

SELECT column1, column2, …. columnN FROM [TABLE NAME] WHERE [CONDITION];

## Update Data

The update query is used to change the existing values in a database. By using update a specific value can be corrected or updated. It only affects the data and not the structure of the table. The basic advantage provided by this command is that it keeps the table accurate.

**Syntax:**

UPDATE table_name

SET ="new value"

WHERE ="old value";

## Delete Data from Table

We can use the Delete query to delete data from the table in MySQL.

**Syntax:**

```
DELETE FROM table_name WHERE attribute_name = attribute_value
```

## Drop Tables

Drop command affects the structure of the table and not data. It is used to delete an already existing table. For cases where you are not sure if the table to be dropped exists or not DROP TABLE IF EXISTS command is used. Both cases will be dealt with in the following examples.

**Syntax:**

```
DROP TABLE tablename;

DROP TABLE IF EXISTS tablename;
```

## MySQL Methods/functions in Python

- MySQLConnection.close() Method.
- MySQLConnection.commit() Method.
- MySQLConnection.config() Method.
- MySQLConnection.connect() Method.
- MySQLConnection.cursor() Method.
- MySQLConnection.cmd_change_user() Method.
- MySQLConnection.cmd_debug() Method.

## CSV File Reading and Writing

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. CSV format was used for many years prior to attempts to describe the format in a standardized way in RFC 4180.

The lack of a well-defined standard means that subtle differences often exist in the data produced and consumed by different applications.

These differences can make it annoying to process CSV files from multiple sources. Still, while the delimiters and quoting characters vary, the overall format is similar enough that it is possible to write a single module which can efficiently manipulate such data, hiding the details of reading and writing the data from the programmer.

The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, “write this data in the format preferred by Excel,” or “read data from this file which was generated by Excel,” without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats.

The csv module’s reader and writer objects read and write sequences. Programmers can also read and write data in dictionary form using the DictReader and DictWriter classes.

## CSV Module Contents:

csv.reader(csvfile, dialect='excel')

csv.writer(csvfile, dialect='excel')

Dialect.**delimiter**
A one-character string used to separate fields. It defaults to ','.

Dialect.**lineterminator**
The string used to terminate lines produced by the writer. It defaults to '\r\n'.

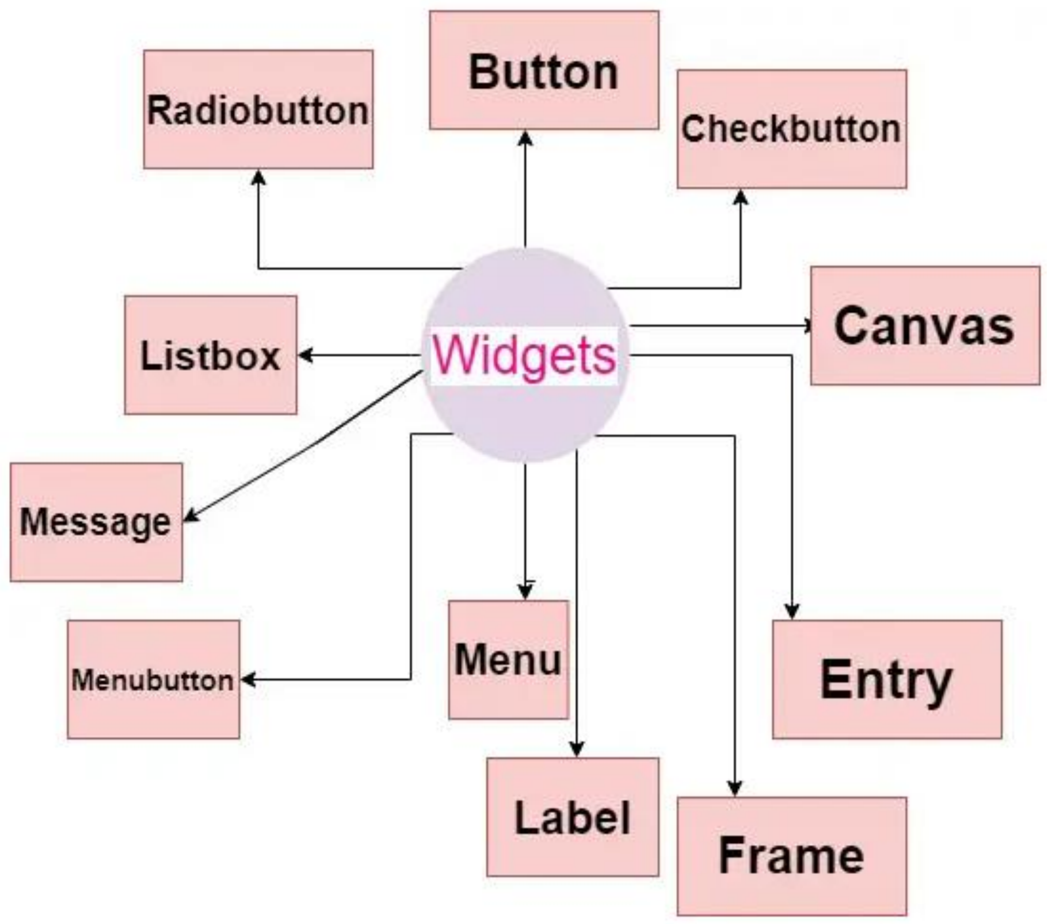
## Tkinter Widgets

In general, Widget is an element of Graphical User Interface (GUI) that displays/illustrates information or gives a way for the user to interact with the OS. In Tkinter , Widgets are objects ; instances of classes that represent buttons, frames, and so on.

Each separate widget is a Python object. When creating a widget, you must pass its parent as a parameter to the widget creation function. The only exception is the “root” window, which is the top-level window that will contain everything else and it does not have a parent.

Tkinter provides widgets that are used to represent in the browser. Tkinter widgets are eventful and represented in the browser, such as textbox, button, etc. In

Python, Tkinter widgets are standard GUI elements used to handle events through elements like button, frames, labels, etc.



In Python, Tkinter provides a small range of widgets that are used in some basic GUI programming, and it also allows us to create specialized widgets as custom widgets. Tkinter widgets usually provide various controls in the GUI applications.

## Widget Classes

Tkinter supports the below mentioned core widgets –

| **Widgets** | **Description** |
|---|---|
| Label | It is used to display text or image on the screen |
| Button | It is used to add buttons to your application |

| | |
|---|---|
| Canvas | It is used to draw pictures and others layouts like texts, graphics etc. |
| Combo Box | It contains a down arrow to select from list of available options |
| Check Button | It displays a number of options to the user as toggle buttons from which user can select any number of options. |
| Radio Button | It is used to implement one-of-many selection as it allows only one option to be selected |
| Entry | It is used to input single line text entry from user |
| Frame | It is used as container to hold and organize the widgets |
| Message | It works same as that of label and refers to multi-line and non-editable text |
| Scale | It is used to provide a graphical slider which allows to select any value from that scale |
| Scrollbar | It is used to scroll down the contents. It provides a slide controller. |
| SpinBox | It is allows user to select from given set of values |
| Text | It allows user to edit multiline text and format the way it has to be displayed |
| Menu | It is used to create all kinds of menu used by an application |

## MessageBox Widget

Python Tkinter – MessageBox Widget is used to display the message boxes in the python applications. This module is used to display a message using provides a number of functions.

messagebox.Function_Name(title, message [options])

### Parameters:

There are various parameters :

- **Function_Name:** This parameter is used to represents an appropriate message box function.
- **title:** This parameter is a string which is shown as a title of a message box.
- **message:** This parameter is the string to be displayed as a message on the message box.

- **options:** There are two options that can be used are:
    1. **default:** This option is used to specify the default button like ABORT, RETRY, or IGNORE in the message box.
    2. **parent:** This option is used to specify the window on top of which the message box is to be displayed

## Function_Name:

There are functions or methods available in the messagebox widget.

1. **showinfo():** Show some relevant information to the user.
2. **showwarning():** Display the warning to the user.
3. **showerror():** Display the error message to the user.
4. **askquestion():** Ask question and user has to answered in yes or no.
5. **askokcancel():** Confirm the user's action regarding some application activity.
6. **askyesno():** User can answer in yes or no for some action.
7. **askretrycancel():** Ask the user about doing a particular task again or not.

**Create a default information message box**.

- class tkinter.messagebox.Message(master=None, **options)

**Information message box:**

- tkinter.messagebox.showinfo(title=None, message=None, **options)

**Warning message boxes:**

- tkinter.messagebox.showwarning(title=None, message=None, **options)
- tkinter.messagebox.showerror(title=None, message=None, **options)

**Question message boxes:**

- tkinter.messagebox.askquestion(title=None, message=None, **options)
- tkinter.messagebox.askokcancel(title=None, message=None, **options)
- tkinter.messagebox.askretrycancel(title=None, message=None, **options)
- tkinter.messagebox.askyesno(title=None, message=None, **options)
- tkinter.messagebox.askyesnocancel(title=None, message=None, **options)
-

## Tkinter filedialog

Python Tkinter (and TK) offer a set of dialogs that you can use when working with files. By using these you don't have to design standard dialogs your self. Example dialogs include an open file dialog, a save file dialog and many others. Besides file dialogs there are other standard dialogs, but in this article we will focus on file dialogs.

File dialogs help you open, save files or directories. This is the type of dialog you get when you click file,open. This dialog comes out of the module, there’s no need to write all the code manually.

Tkinter does not have a native looking file dialog, instead it has the customer tk style. You can see these below.

The tkinter filedialog comes in several types. Which type you need really depends on your applications needs. All of them are methods calls.

You can open a single file, a directory, save as file and much more. Each dialog made with the example below is a different type of dialog.

**from tkinter import filedialog**

- tkinter.filedialog.asksaveasfilename()
- tkinter.filedialog.asksaveasfile()
- tkinter.filedialog.askopenfilename()
- tkinter.filedialog.askopenfile()
- tkinter.filedialog.askdirectory()
- tkinter.filedialog.askopenfilenames()
- tkinter.filedialog.askopenfiles()

## Geometry Managers in Tkinter

Main concept you should know about while working on Python Tkinter projects is geometry managers. Without using a geometry manager, your widgets wouldn’t appear on the screen. There are primarily three ways you can add a geometry manager.

**Grid( )**

It adds a grid to the parent and allows it to show widgets accordingly. The grid has rows and columns for measuring the length and height of its widgets.

**Pack( )**

This method tells the widget to pack itself within the parent. It only adds the widget to a list of children widgets for the parent. In simple terms, it adds them to a block before it places them in a parent widget.

**Place( )**

This method places widgets in a specific location according to the parent widget.

## Advantages of Tkinter Module in Python

Now that you have a basic understanding of what exactly Python Tkinter refers to, and how it operates, let’s take a look at some of the basic advantages of the Tkinter module in Python that are mentioned below.

- When compared with other GUI toolkits, Tkinter is much easier and faster to use.
- You no longer need to download anything extra, since Tkinter is already included in Python.
- Tkinter has a simple syntax, as well as three geometry managers, namely grid, place, and pack.

Last but not least, Python Tkinter is also extremely easy to understand, and can be mastered quite easily.

## Toplevel widget

A Toplevel widget is used to create a window on top of all other windows. The Toplevel widget is used to provide some extra information to the user and also when our program deals with more than one application. These windows are directly organized and managed by the Window Manager and do not need to have any parent window associated with them every time.

**Syntax:**
toplevel = Toplevel(root, bg, fg, bd, height, width, font, ..)

**Optional parameters:**

- **root** = root window(optional)
- **bg** = background colour
- **fg** = foreground colour
- **bd** = border
- **height** = height of the widget.
- **width** = width of the widget.
- **font** = Font type of the text.
- **cursor** = cursor that appears on the widget which can be an arrow, a dot etc.

**Common methods:**

- **iconify** turns the windows into icon.
- **deiconify** turns back the icon into window.
- **state** returns the current state of window.
- **withdraw** removes the window from the screen.
- **title** defines title for window.
- **frame** returns a window identifier which is system spec

## Customer Management System.py

```
from tkinter import *
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox, filedialog
import mysql.connector
import csv
import os
import sys

mydata = []

def update(rows):
    global mydata
    mydata = rows
    trv.delete(*trv.get_children())
    for i in rows:
        trv.insert('', 'end', values=i)

def search():
    q2 = q.get()
    query = "SELECT id, name, email, age FROM customers WHERE name LIKE
'%"+q2+"%' OR email LIKE '%"+q2+"%'"
    cursor.execute(query)
    rows = cursor.fetchall()
    update(rows)

def clear():
    query = "SELECT id, name, email, age FROM customers"
    cursor.execute(query)
    rows = cursor.fetchall()
    update(rows)

def getrow(event):
    rowid = trv.identify_row(event.y)
    item = trv.item(trv.focus())
    t1.set(item['values'][0])
    t2.set(item['values'][1])
    t3.set(item['values'][2])
    t4.set(item['values'][3])

def update_customer():
    name = t2.get()
    email = t3.get()
    age = t4.get()
    custid = t1.get()

    if messagebox.askyesno("Confirm Please", "Are you sure you want to UPDATE
this customer?"):
        query = "UPDATE customers SET name = %s, email = %s, age = %s WHERE id
= %s"
        cursor.execute(query, (name, email, age, custid))
        mydb.commit()
        clear()
    else:
        return True

def add_new():
    name = t2.get()
    email = t3.get()
    age = t4.get()
    if messagebox.askyesno("Confirm Please", "Are you sure you want to ADD this
customer?"):
        query = "INSERT INTO customers(id, name, email, age) VALUES(NULL, %s,
%s, %s)"
        cursor.execute(query, (name, email, age))
        mydb.commit()
        clear()
    else:
        return True

def delete_customer():
    customer_id = t1.get()
    if messagebox.askyesno("Confirm Delete?", "Are you sure you want to delete this
customer?"):
        query = "DELETE FROM customers WHERE id = "+customer_id
        cursor.execute(query)
        mydb.commit()
        clear()
    else:
        return True

def export():
    if len(mydata) < 1:
        messagebox.showerror("No Data", "No data available to export")
        return False

    fln = filedialog.asksaveasfilename(initialdir=os.getcwd(), title="Save CSV",
filetypes=(("CSV File", "*.csv"), ("All Files", "*.*")))
```

```
    with open(fln, mode='w') as myfile:
        exp_writer = csv.writer(myfile, delimiter=',', lineterminator = '\n')
        for i in mydata:
            exp_writer.writerow(i)

    messagebox.showinfo("Data Exported", "Your data has been exported to
"+os.path.basename(fln)+ " successfully.")

def importcsv():
    mydata.clear()
    fln = filedialog.askopenfilename(initialdir=os.getcwd(), title="Save CSV",
filetypes=(("CSV File", "*.csv"), ("All Files", "*.*")))
    with open(fln) as myfile:
        csvread = csv.reader(myfile, delimiter=',')
        for i in csvread:
            mydata.append(i)
    update(mydata)
    messagebox.showinfo("Data Imported", "Your data has been imported
successfully.")

def savedb():
    if messagebox.askyesno("Confirmation", "Are you sure you want to save data to
Database"):
        for i in mydata:
            id = i[0]
            name = i[1]
            email = i[2]
            age = i[3]
            query = "INSERT INTO customers(id, name, email, age) VALUES(NULL,
%s, %s, %s)"
            cursor.execute(query, (name, email, age))
        mydb.commit()
        clear()
        messagebox.showinfo("Data Saved", "Data has been saved to Database
successfully.")
    else:
        return False

mydb = mysql.connector.connect(host="localhost", user="root",
passwd="Uzumaki_706", database="test")
cursor = mydb.cursor()
root = tk.Tk()
```

**#Login Program Section**

```
top = Toplevel()
```

```
top.title("LogIn")
top.geometry('450x450')
top.configure(bg='#333333')

username_entry = Entry(top) #Username entry
password_entry = Entry(top) #Password entry

def login():
    username = "user"
    password = "123"
    if username_entry.get()==username and password_entry.get()==password:
        messagebox.showinfo(title="Login Success", message="You successfully logged
in.")
        root.deiconify()
        top.destroy()
    else:
        messagebox.showerror(title="Error", message="Invalid login.")

def cancel():
    top.destroy() #Removes the toplevel window
    root.destroy() #Removes the hidden root window
    sys.exit() #Ends the script
```

**# Creating widgets**

```
login_label = tk.Label(top, text="Login", bg='#333333', fg="#FF7900", font=("Arial",
30))
username_label = tk.Label(top, text="Username", bg='#333333', fg="#FFFFFF",
font=("Arial", 16))
username_entry = tk.Entry(top, font=("Arial", 16))
password_entry = tk.Entry(top, show="*", font=("Arial", 16))
password_label = tk.Label(top, text="Password", bg='#333333', fg="#FFFFFF",
font=("Arial", 16))
login_button = tk.Button(top, text="Login", bg="GREEN", fg="#FFFFFF",
font=("Arial", 16), command=lambda:login())
cancel_button = tk.Button(top, text="Cancel", bg="RED", fg="#FFFFFF",
font=("Arial", 16), command=lambda:cancel())
```

**# Placing widgets on the screen**

```
login_label.grid(row=0, column=1, columnspan=1, sticky="news", pady=40)
username_label.grid(row=1, column=0)
username_entry.grid(row=1, column=1, pady=20)

password_label.grid(row=2, column=0)
password_entry.grid(row=2, column=1, pady=20)
```

```
login_button.grid(row=3, column=1, columnspan=1, pady=20)
cancel_button.grid(row=4, column=1, columnspan=1, pady=20)

login_img = PhotoImage(file='login.png')
top.iconphoto(False,login_img)
top.resizable(False,False)
root.withdraw()
```

**#Main Program Section**

```
q = StringVar()
t1 = StringVar()
t2 = StringVar()
t3 = StringVar()
t4 = StringVar()

root.configure(bg="#333333")
root.title("Customer Management System")
root.geometry("800x680")
root.resizable(False, False)
main_img = PhotoImage(file='user.png')
root.iconphoto(False,main_img)

wrapper1 = LabelFrame(root, text = "Customer List")
wrapper2 = LabelFrame(root, text = "Search and Modify Data")
wrapper3 = LabelFrame(root, text = "Customer Data")

wrapper1.pack(fill="both", expand="yes", padx=20, pady=10)
wrapper2.pack(fill="both", expand="yes", padx=20, pady=10)
wrapper3.pack(fill="both", expand="yes", padx=20, pady=10)

trv = ttk.Treeview(wrapper1, columns=(1,2,3,4), show="headings", height="7")
trv.pack(side=LEFT)
trv.place(x=0, y=0)

trv.heading(1, text="Customer ID")
trv.heading(2, text="Name")
trv.heading(3, text="Email")
trv.heading(4, text="Age")

trv.bind('<Double 1>', getrow)
```

**#vertical scrollbar**

```
yscrollbar = ttk.Scrollbar(wrapper1 , orient="vertical", command=trv.yview)
```

```
yscrollbar.pack(side=RIGHT, fill="y")
trv.configure(yscrollcommand=yscrollbar.set)

query = "SELECT id, name, email, age from customers"
cursor.execute(query)
rows = cursor.fetchall()
update(rows)
```

**#Search Section**

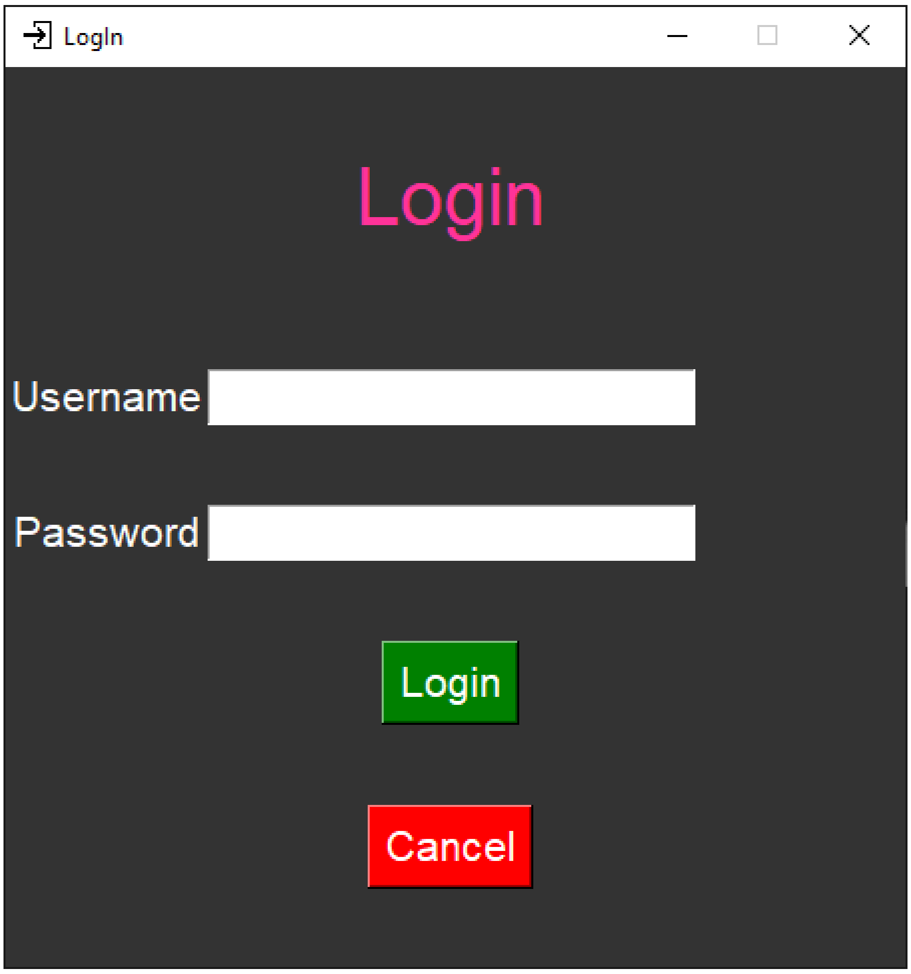
```
lbl = Label(wrapper2, text="Search")
lbl.pack(side=tk.LEFT, padx=10)
ent = Entry(wrapper2, textvariable=q)
ent.pack(side=tk.LEFT, padx=6)
btn = Button(wrapper2, text="Search", command=search)
btn.pack(side=tk.LEFT, padx=6)
cbtn = Button(wrapper2, text="Clear", command=clear)
cbtn.pack(side=tk.LEFT, padx=6)

extbtn = Button(wrapper2, text="Exit", command=lambda: exit())
extbtn.pack(side=tk.LEFT, padx=6)

savebtn = Button(wrapper2, text="Save Data", command=savedb)
savebtn.pack(side=tk.RIGHT, padx=10, pady=10)

expbtn = Button(wrapper2, text="Export CSV", command=export)
expbtn.pack(side=tk.RIGHT, padx=10, pady=10)
impbtn = Button(wrapper2, text="Import CSV", command=importcsv)
impbtn.pack(side=tk.RIGHT, padx=10, pady=10)
```

**#User Data Section**

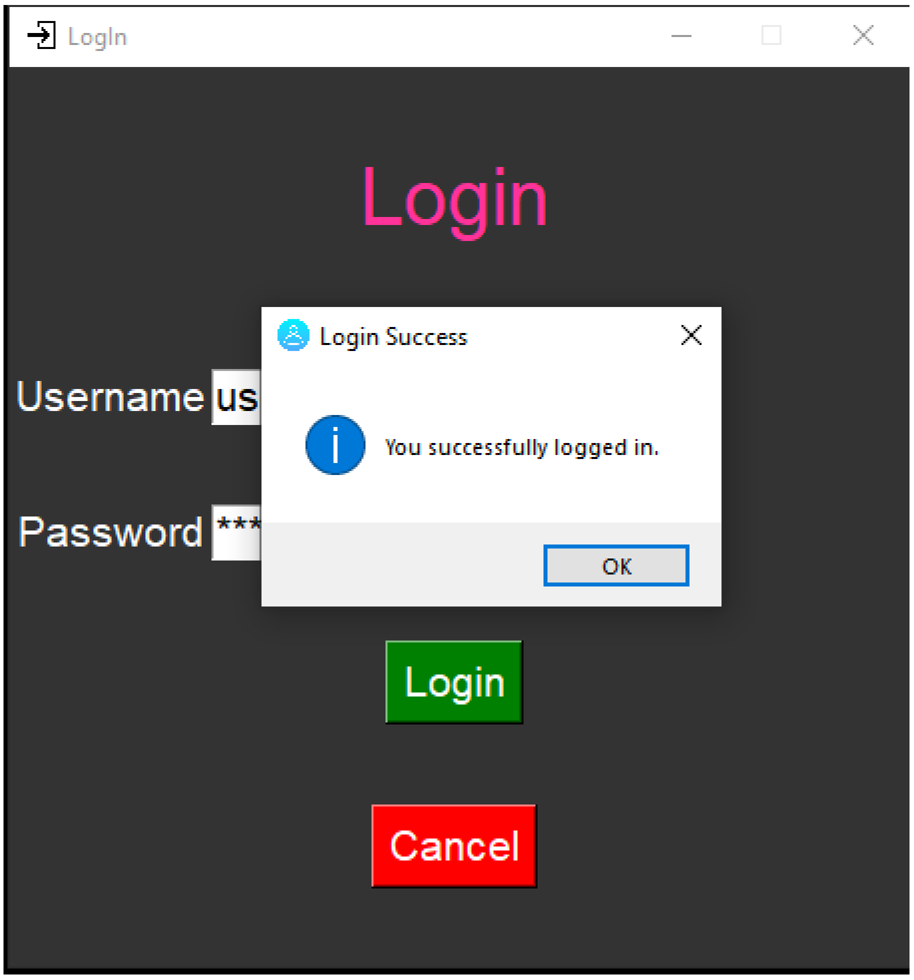
```
lbl1 = Label(wrapper3, text="Customer ID")
lbl1.grid(row=0, column=0, padx=5, pady=3)
ent1 = Entry(wrapper3, textvariable=t1)
ent1.grid(row=0, column=1, padx=5, pady=3)

lbl2 = Label(wrapper3, text="Name")
lbl2.grid(row=1, column=0, padx=5, pady=3)
ent2 = Entry(wrapper3, textvariable=t2)
ent2.grid(row=1, column=1, padx=5, pady=3)

lbl3 = Label(wrapper3, text="Email")
lbl3.grid(row=2, column=0, padx=5, pady=3)
```

```
ent3 = Entry(wrapper3, textvariable=t3)
ent3.grid(row=2, column=1, padx=5, pady=3)
lbl4 = Label(wrapper3, text="Age")
lbl4.grid(row=3, column=0, padx=5, pady=3)
ent4 = Entry(wrapper3, textvariable=t4)
ent4.grid(row=3, column=1, padx=5, pady=3)

up_btn = Button(wrapper3, text="Update", command=update_customer)
add_btn = Button(wrapper3, text="Add New", command=add_new)
delete_btn = Button(wrapper3, text="Delete", command=delete_customer)

add_btn.grid(row=4, column=0, padx=5, pady=3)
up_btn.grid(row=4, column=1, padx=5, pady=3)
delete_btn.grid(row=4, column=2, padx=5, pady=3)

top.mainloop()
root.mainloop()

#End of Program
```

**Login Page:**

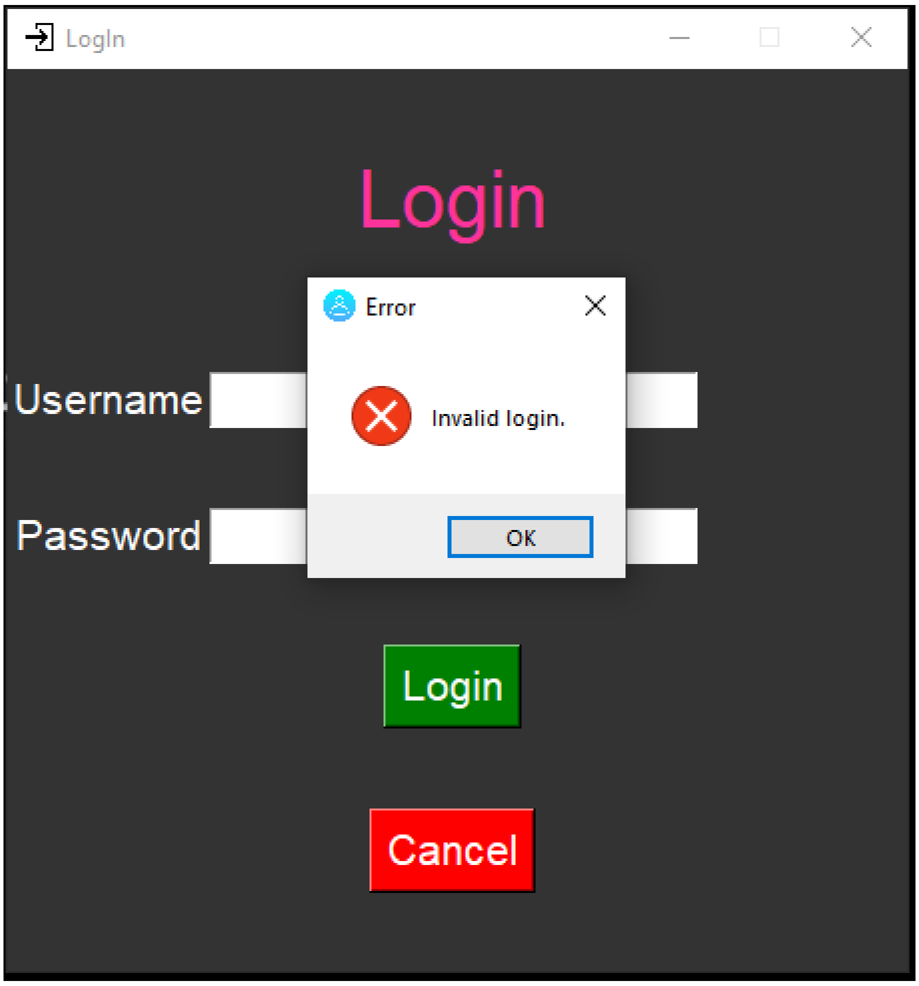


## Login Success:



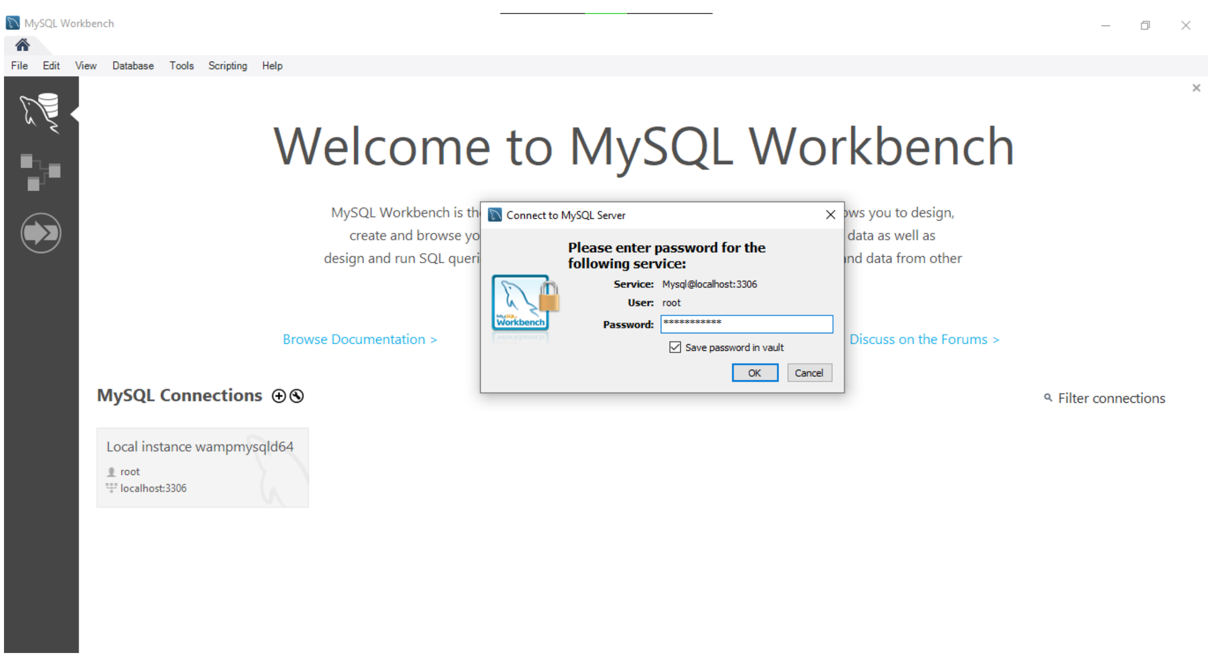
You successfully logged in.

**Invalid Login:**



Invalid login.

## Home page of MySQL Workbench:



Please enter password for the following service:

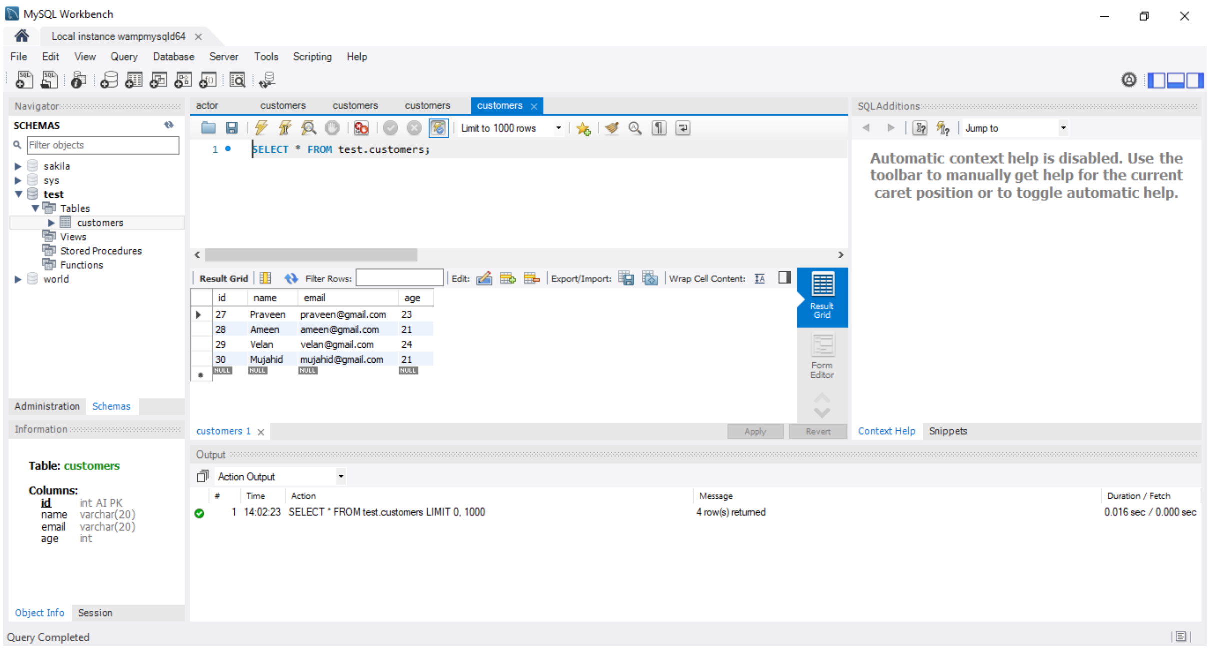
Service: Mysql@localhost:3306

User: root

Password: ***********

☑ Save password in vault

## Data Stored in MySQL DataBase:



```
SELECT * FROM test.customers;
```

| id | name | email | age |
|---|---|---|---|
| 27 | Praveen | praveen@gmail.com | 23 |
| 28 | Ameen | ameen@gmail.com | 21 |
| 29 | Velan | velan@gmail.com | 24 |
| 30 | Mujahid | mujahid@gmail.com | 21 |
| NULL | NULL | NULL | NULL |

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

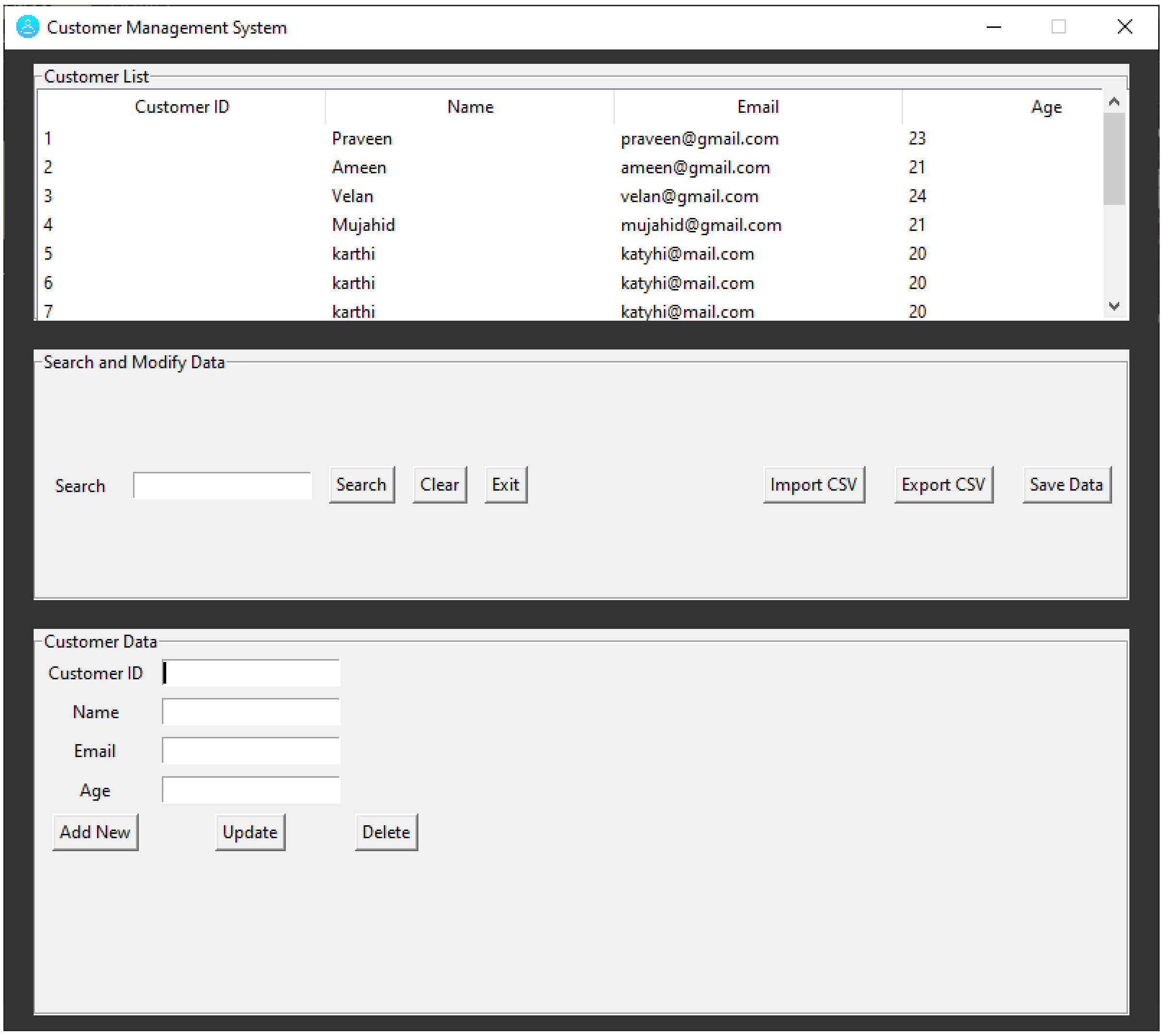
Table: customers

Columns:
id int AI PK
name varchar(20)
email varchar(20)
age int

```
#  Time      Action                                         Message               Duration / Fetch
1  14:02:23  SELECT * FROM test.customers LIMIT 0, 1000     4 row(s) returned     0.016 sec / 0.000 sec
```

Query Completed

## Customer Management System:



Customer Management System

Customer List

| Customer ID | Name | Email | Age |
|---|---|---|---|
| 1 | Praveen | praveen@gmail.com | 23 |
| 2 | Ameen | ameen@gmail.com | 21 |
| 3 | Velan | velan@gmail.com | 24 |
| 4 | Mujahid | mujahid@gmail.com | 21 |
| 5 | karthi | katyhi@mail.com | 20 |
| 6 | karthi | katyhi@mail.com | 20 |
| 7 | karthi | katyhi@mail.com | 20 |

Search and Modify Data

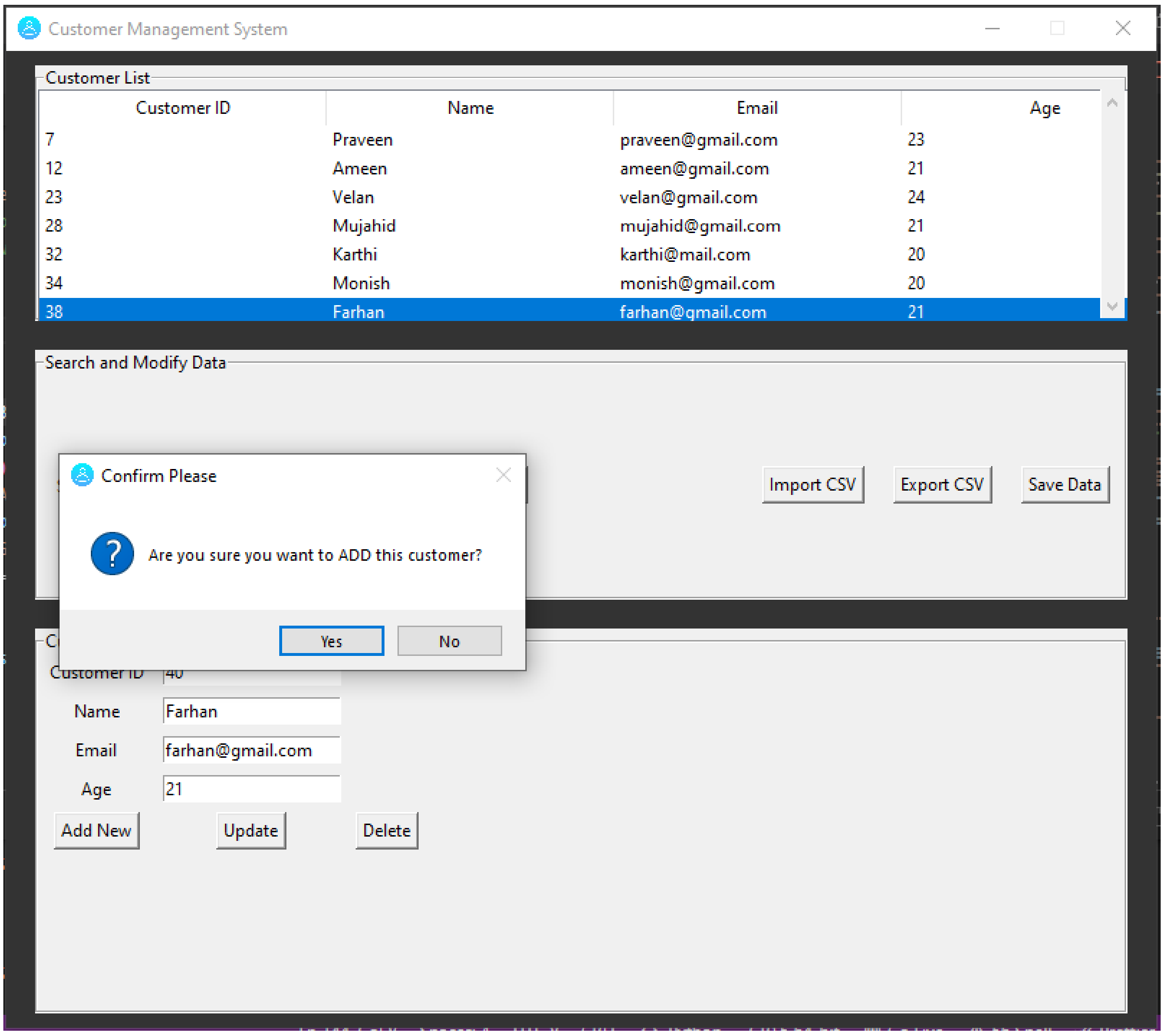
Search [ ] Search Clear Exit Import CSV Export CSV Save Data

Customer Data

| | |
|---|---|
| Customer ID | |
| Name | |
| Email | |
| Age | |

Add New Update Delete

## Add New Customer:



Customer Management System

Customer List

| Customer ID | Name | Email | Age |
|---|---|---|---|
| 7 | Praveen | praveen@gmail.com | 23 |
| 12 | Ameen | ameen@gmail.com | 21 |
| 23 | Velan | velan@gmail.com | 24 |
| 28 | Mujahid | mujahid@gmail.com | 21 |
| 32 | Karthi | karthi@mail.com | 20 |
| 34 | Monish | monish@gmail.com | 20 |
| 38 | Farhan | farhan@gmail.com | 21 |

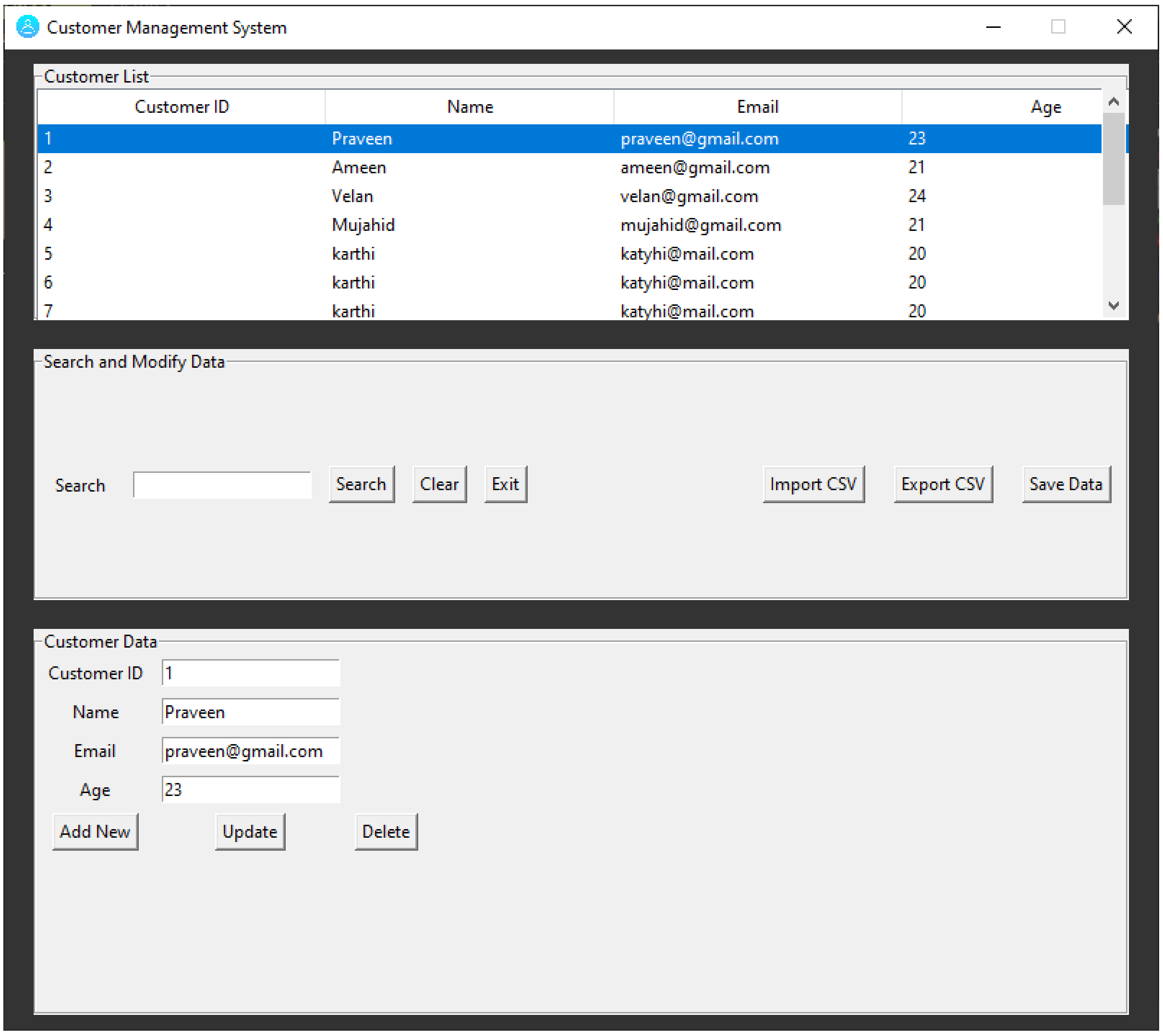
Search and Modify Data

Confirm Please

Are you sure you want to ADD this customer?

| | |
|---|---|
| Customer ID | 40 |
| Name | Farhan |
| Email | farhan@gmail.com |
| Age | 21 |

## Select Data for Update:


Customer Management System

Customer List

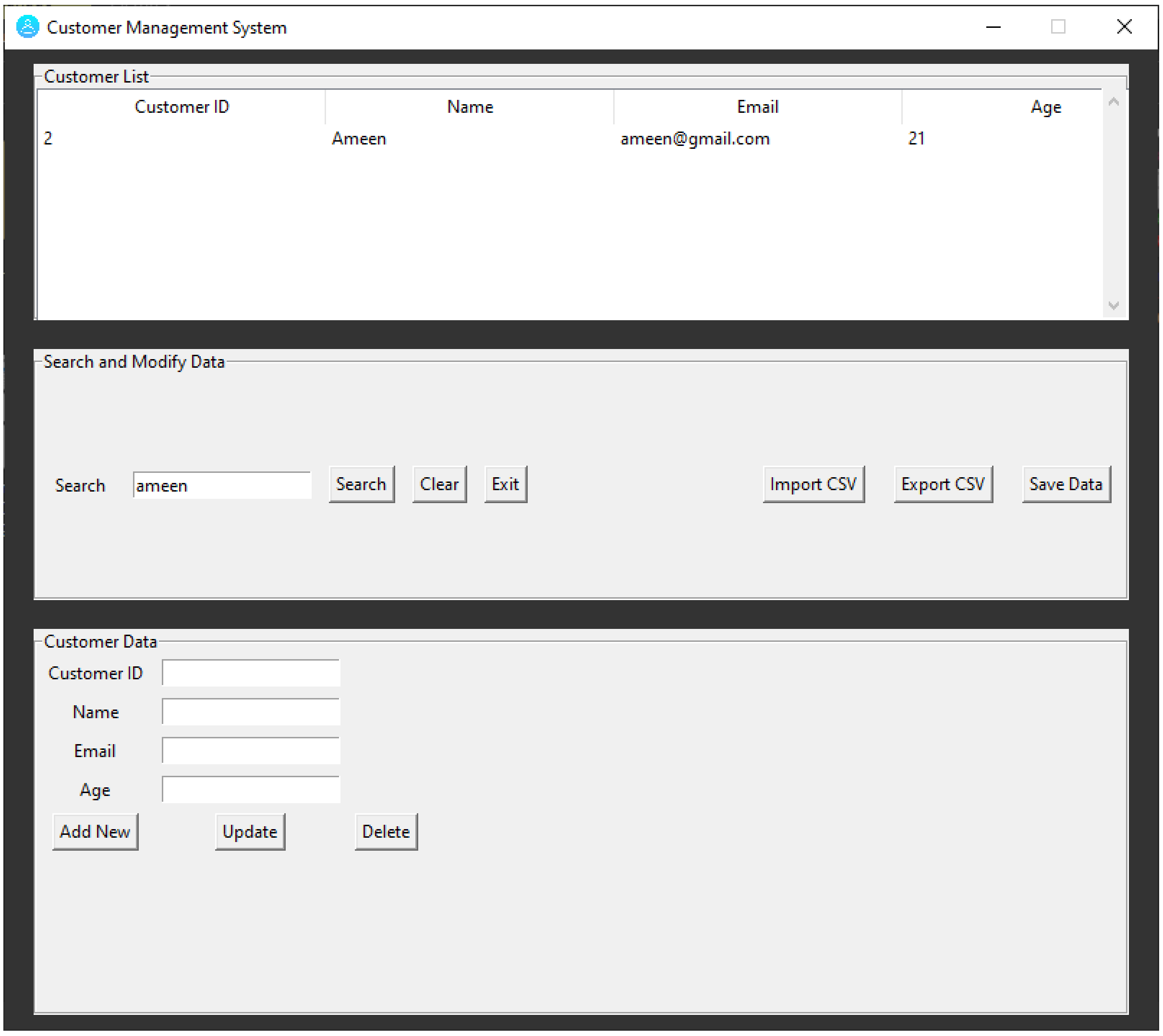
| Customer ID | Name | Email | Age |
|---|---|---|---|
| 1 | Praveen | praveen@gmail.com | 23 |
| 2 | Ameen | ameen@gmail.com | 21 |
| 3 | Velan | velan@gmail.com | 24 |
| 4 | Mujahid | mujahid@gmail.com | 21 |
| 5 | karthi | katyhi@mail.com | 20 |
| 6 | karthi | katyhi@mail.com | 20 |
| 7 | karthi | katyhi@mail.com | 20 |

Search and Modify Data

Customer Data

Customer ID: 1
Name: Praveen
Email: praveen@gmail.com
Age: 23

## Search Data by Name or Email:



Customer Management System

Customer List

| Customer ID | Name | Email | Age |
|---|---|---|---|
| 2 | Ameen | ameen@gmail.com | 21 |

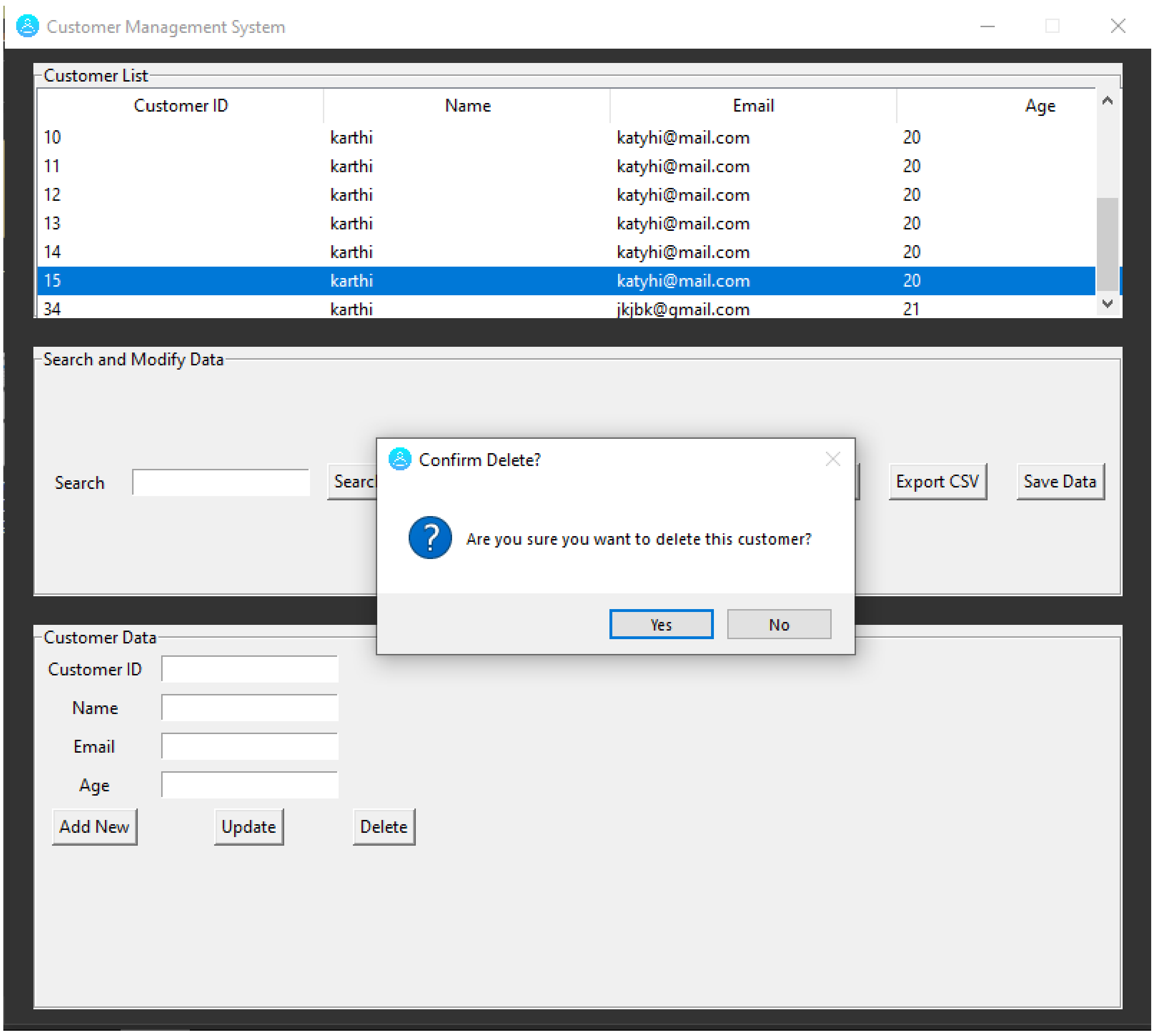
Search and Modify Data

Search: ameen

Customer Data

| | |
|---|---|
| Customer ID | |
| Name | |
| Email | |
| Age | |

## Delete a Data from List:


Customer Management System

Customer List

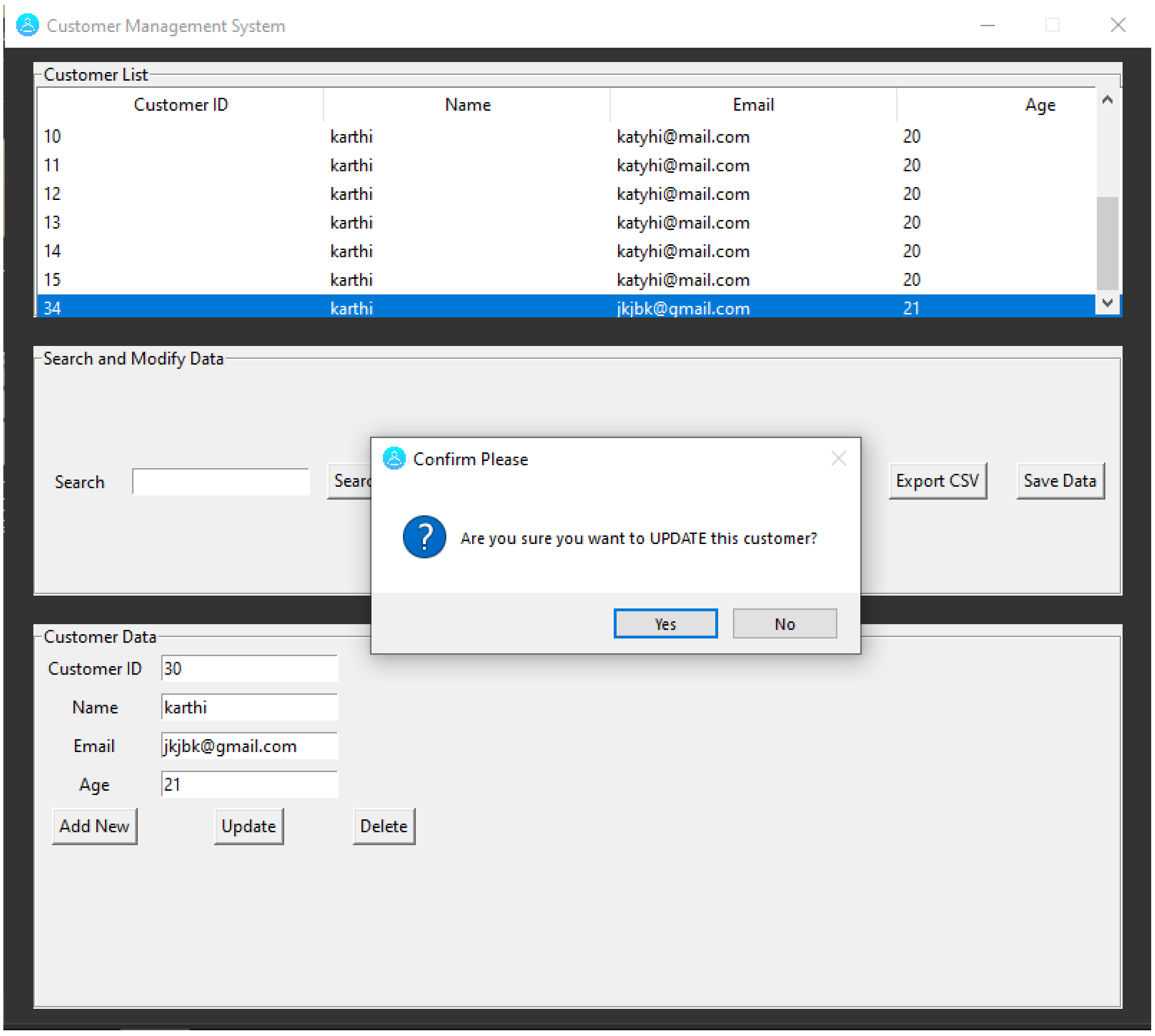
| Customer ID | Name | Email | Age |
|---|---|---|---|
| 10 | karthi | katyhi@mail.com | 20 |
| 11 | karthi | katyhi@mail.com | 20 |
| 12 | karthi | katyhi@mail.com | 20 |
| 13 | karthi | katyhi@mail.com | 20 |
| 14 | karthi | katyhi@mail.com | 20 |
| 15 | karthi | katyhi@mail.com | 20 |
| 34 | karthi | jkjbk@gmail.com | 21 |

Search and Modify Data

Confirm Delete?

Are you sure you want to delete this customer?

## Update a Data in List:



Customer Management System

Customer List

| Customer ID | Name | Email | Age |
|---|---|---|---|
| 10 | karthi | katyhi@mail.com | 20 |
| 11 | karthi | katyhi@mail.com | 20 |
| 12 | karthi | katyhi@mail.com | 20 |
| 13 | karthi | katyhi@mail.com | 20 |
| 14 | karthi | katyhi@mail.com | 20 |
| 15 | karthi | katyhi@mail.com | 20 |
| 34 | karthi | jkjbk@gmail.com | 21 |

Search and Modify Data

Search

Confirm Please

Are you sure you want to UPDATE this customer?

Yes No

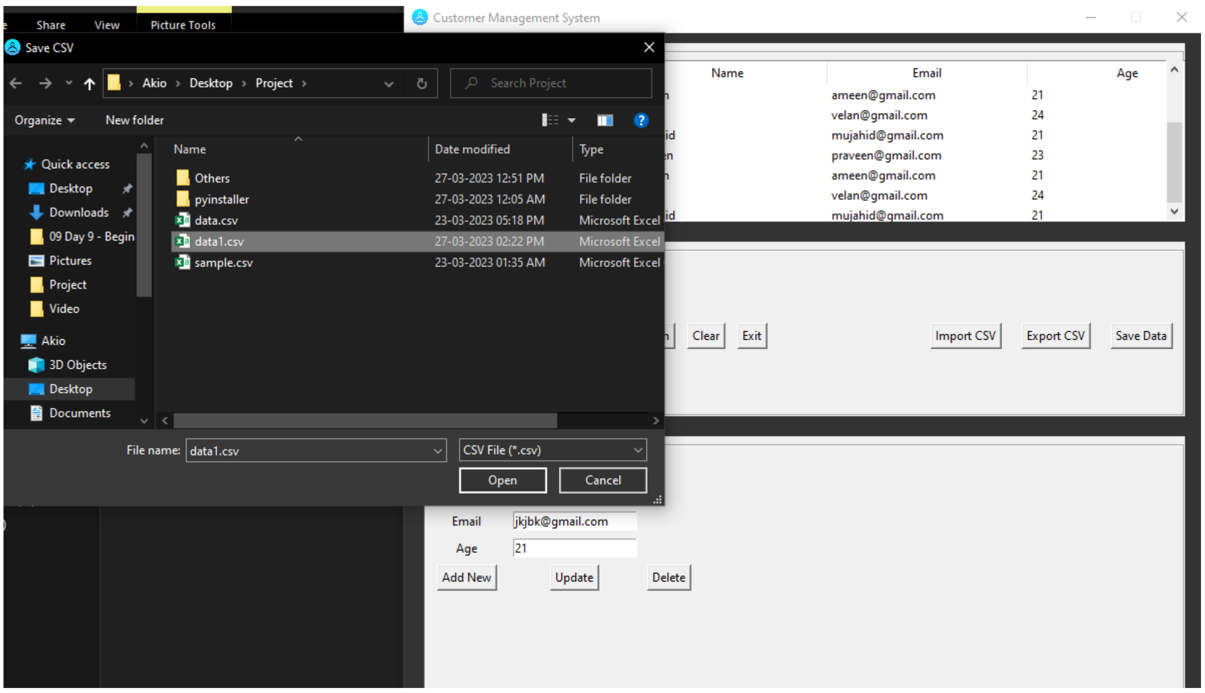
Export CSV Save Data

Customer Data

| | |
|---|---|
| Customer ID | 30 |
| Name | karthi |
| Email | jkjbk@gmail.com |
| Age | 21 |

Add New Update Delete

## Import Data from CSV file:



| Name | Date modified | Type |
|---|---|---|
| Others | 27-03-2023 12:51 PM | File folder |
| pyinstaller | 27-03-2023 12:05 AM | File folder |
| data.csv | 23-03-2023 05:18 PM | Microsoft Excel |
| data1.csv | 27-03-2023 02:22 PM | Microsoft Excel |
| sample.csv | 23-03-2023 01:35 AM | Microsoft Excel |

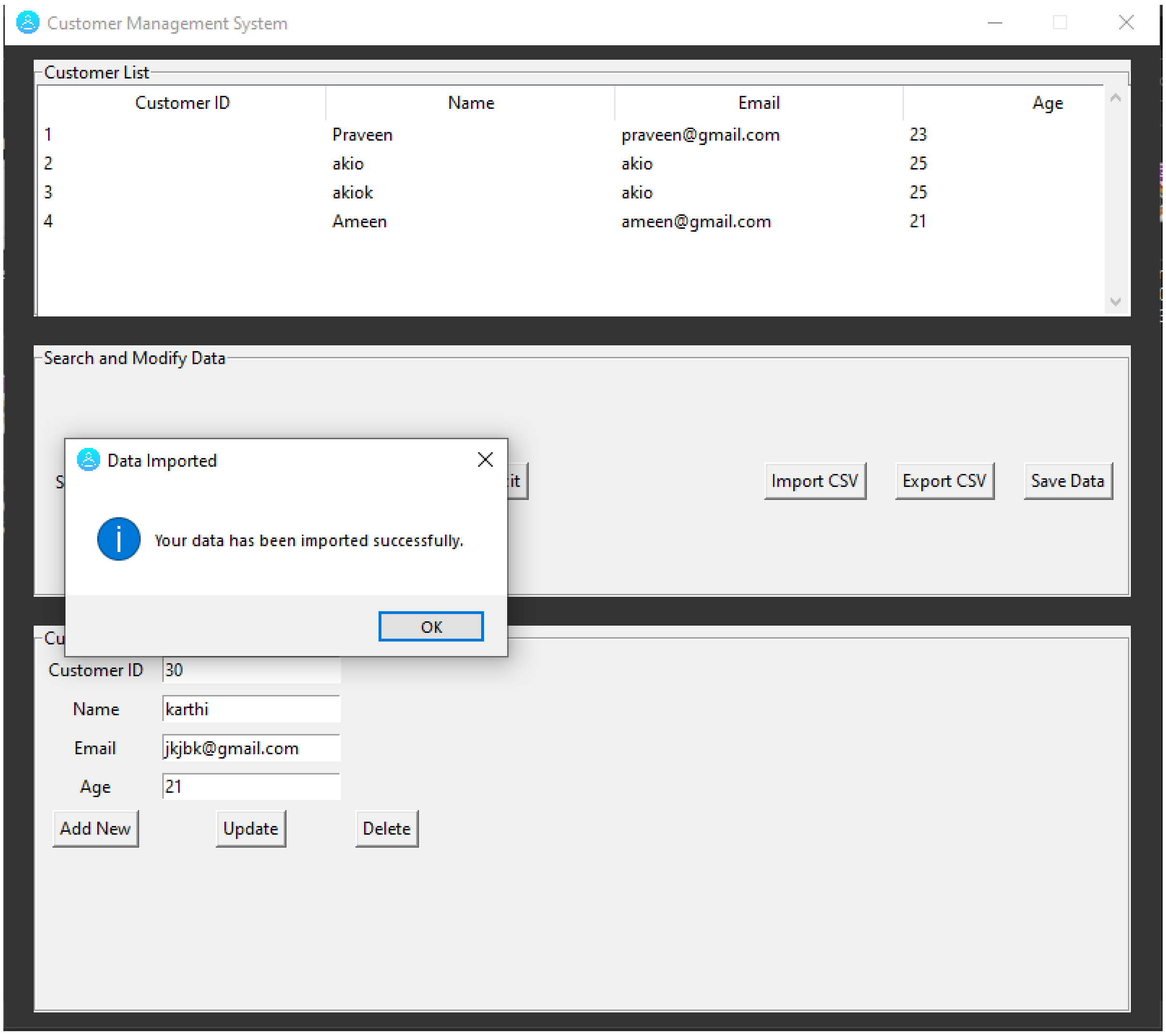
File name: data1.csv

CSV File (*.csv)

| Name | Email | Age |
|---|---|---|
| | ameen@gmail.com | 21 |
| | velan@gmail.com | 24 |
| | mujahid@gmail.com | 21 |
| | praveen@gmail.com | 23 |
| | ameen@gmail.com | 21 |
| | velan@gmail.com | 24 |
| | mujahid@gmail.com | 21 |

Email jkjbk@gmail.com

Age 21

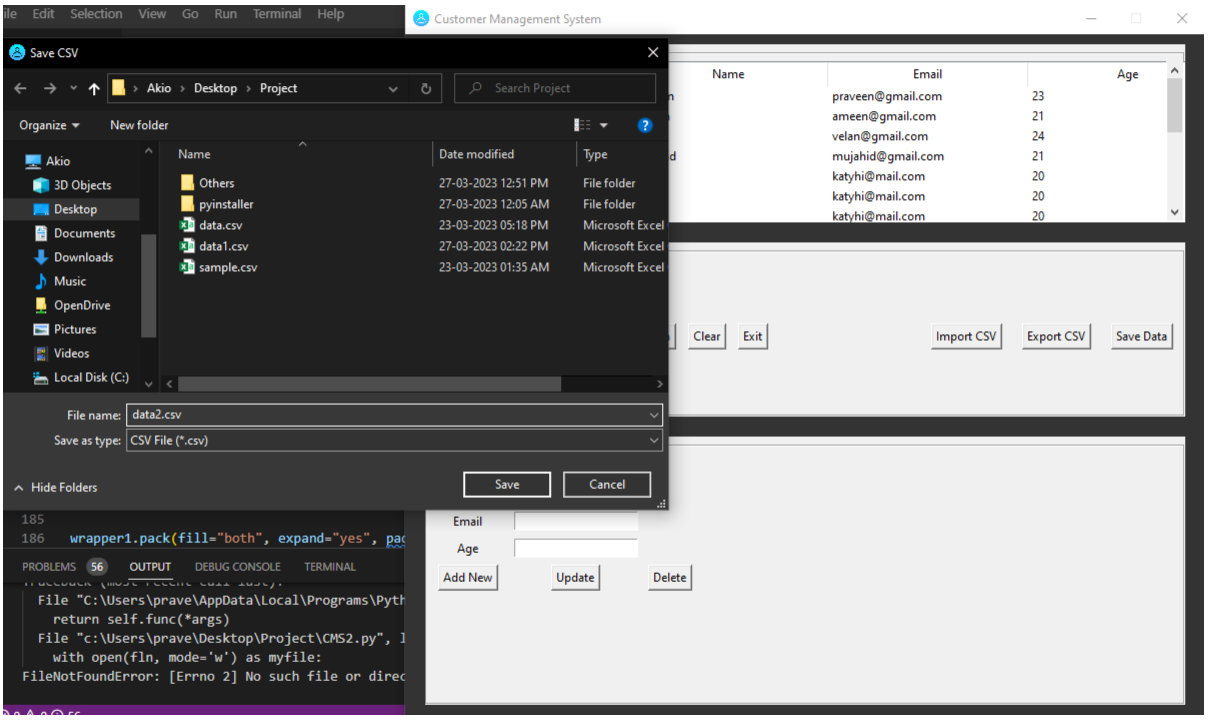


| Customer ID | Name | Email | Age |
|---|---|---|---|
| 1 | Praveen | praveen@gmail.com | 23 |
| 2 | akio | akio | 25 |
| 3 | akiok | akio | 25 |
| 4 | Ameen | ameen@gmail.com | 21 |

Your data has been imported successfully.

| | |
|---|---|
| Customer ID | 30 |
| Name | karthi |
| Email | jkjbk@gmail.com |
| Age | 21 |

## Export Data as CSV file:



| Name | Date modified | Type |
|---|---|---|
| Others | 27-03-2023 12:51 PM | File folder |
| pyinstaller | 27-03-2023 12:05 AM | File folder |
| data.csv | 23-03-2023 05:18 PM | Microsoft Excel |
| data1.csv | 27-03-2023 02:22 PM | Microsoft Excel |
| sample.csv | 23-03-2023 01:35 AM | Microsoft Excel |

File name: data2.csv

Save as type: CSV File (*.csv)

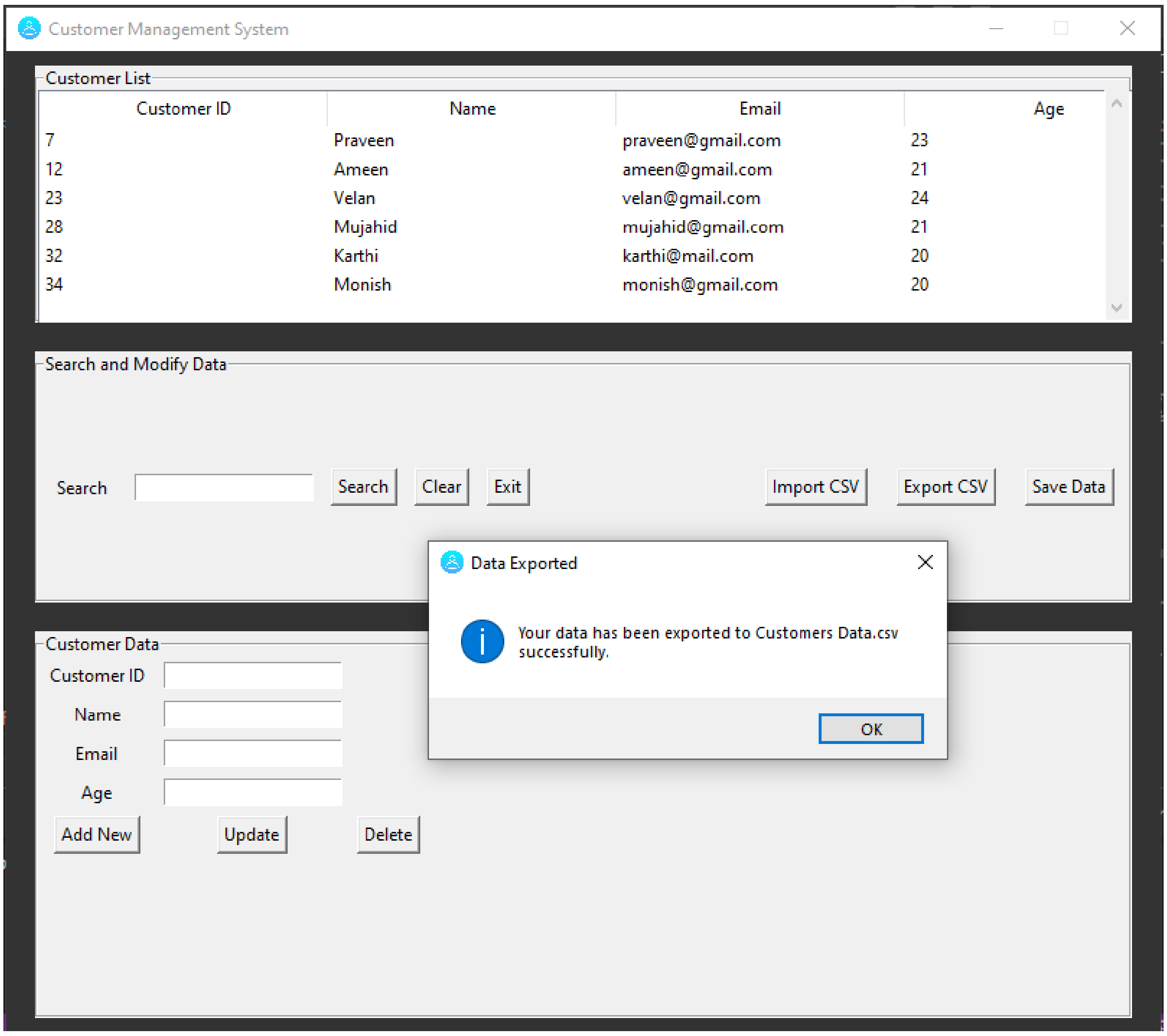
| Name | Email | Age |
|---|---|---|
| | praveen@gmail.com | 23 |
| | ameen@gmail.com | 21 |
| | velan@gmail.com | 24 |
| | mujahid@gmail.com | 21 |
| | katyhi@mail.com | 20 |
| | katyhi@mail.com | 20 |
| | katyhi@mail.com | 20 |

Email

Age

```
wrapper1.pack(fill="both", expand="yes", pa
```

```
  File "C:\Users\prave\AppData\Local\Programs\Pyth
    return self.func(*args)
  File "c:\Users\prave\Desktop\Project\CMS2.py", l
    with open(fln, mode='w') as myfile:
FileNotFoundError: [Errno 2] No such file or direc
```


Customer Management System

Customer List

| Customer ID | Name | Email | Age |
|---|---|---|---|
| 7 | Praveen | praveen@gmail.com | 23 |
| 12 | Ameen | ameen@gmail.com | 21 |
| 23 | Velan | velan@gmail.com | 24 |
| 28 | Mujahid | mujahid@gmail.com | 21 |
| 32 | Karthi | karthi@mail.com | 20 |
| 34 | Monish | monish@gmail.com | 20 |

Search and Modify Data

Search | Search | Clear | Exit | Import CSV | Export CSV | Save Data

Customer Data

Customer ID:
Name:
Email:
Age:

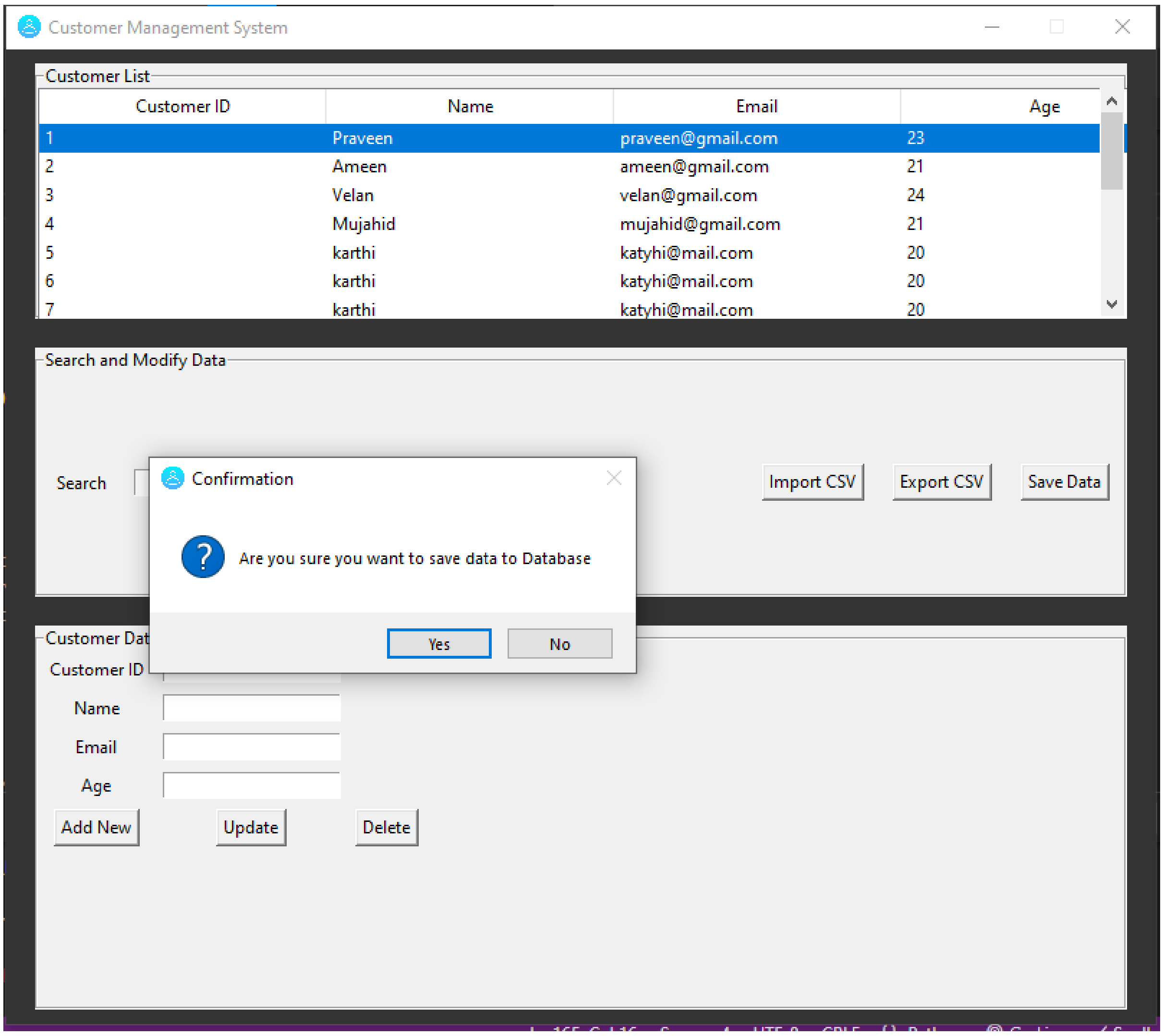
Add New | Update | Delete

Data Exported

Your data has been exported to Customers Data.csv successfully.

OK

## Save Data from CSV to MySQL Database:



Customer Management System

Customer List

| Customer ID | Name | Email | Age |
|---|---|---|---|
| 1 | Praveen | praveen@gmail.com | 23 |
| 2 | Ameen | ameen@gmail.com | 21 |
| 3 | Velan | velan@gmail.com | 24 |
| 4 | Mujahid | mujahid@gmail.com | 21 |
| 5 | karthi | katyhi@mail.com | 20 |
| 6 | karthi | katyhi@mail.com | 20 |
| 7 | karthi | katyhi@mail.com | 20 |

Confirmation

Are you sure you want to save data to Database

## Bibliography

1) https://docs.python.org/3/library/tkinter.html

2) https://en.wikipedia.org/wiki/Tkinter

3) https://www.geeksforgeeks.org/python-gui-tkinter/

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

5) https://books.goalkicker.com/PythonBook/