MOBILE SHOP MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

R.KABILESHKUMAR – 312322205075

J.S.MELVIN FREDRICK-312322205100

of

BACHELOR OF

TECHNOLOGY *in*

INFORMATION TECHNOLOGY

St. JOSEPH'S COLLEGE OF ENGINEERING

(An Autonomous Institution)

St. Joseph's Group of

Institutions OMR, Chennai 600

119

ANNA UNIVERSITY: CHENNAI

May-2024

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report is the bonafide work of

R.KABILESHKUMAR (312322205075)
J.S.MELVIN FREDRICK(312322205100)
who carried out the project under my supervision.

SIGNATURE	SIGNATURE
Supervisor,	**Head of the department,**
Mr.Manikandan B.E., M.E.,	**Mrs. G. Lathaselvi, B.E., M.E.,**
Assistant Professor,	**Head of Department,**
Department of IT,	**Department of IT,**
St. Joseph's College of Engineering, OMR, Chennai- 600119.	St. Joseph's College of Engineering, OMR, Chennai- 600119.

ACKNOWLEDGEMENT

At the outset we would like to express our sincere gratitude to the beloved **Chairman, Dr. Babu Manoharan, M.A., M.B.A., Ph.D.,** for his constant guidance and support.

We would like to express our heartfelt thanks to our respected **Managing Director Mrs. S. Jessie Priya, M.Com.,** for her kind encouragement and blessings.

We wish to express our sincere thanks to our **Executive Director Mr. B. Shashi Sekar, M.Sc.,** for providing ample facilities in the institution.

We express our deepest gratitude and thanks to our beloved **Principal Dr. Vaddi Seshagiri Rao, M.E., M.B.A., Ph.D., F.I.E.,** for his inspirational ideas during the course of the project.

We wish to express our sincere thanks and gratitude to **Mrs. G. Lathaselvi, B.E., M.E.,** Head of the Department, Department of Information Technology - St. Joseph’s College of Engineering for her guidance and assistance in solving the various intricacies involved in the project.

It is with deep sense of gratitude that we acknowledge our gratitude to our supervisor **Mr.Manikandan** for her expert guidance and connoisseur suggestion.

Finally, we thank our department staff members who helped us in the successful completion of this project.

TABLE OF CONTENTS

INTRODUCTION TO PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting language to connect existing components together. Python's simple and easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python's features include –

- **Easy to code:** Python is a high-level programming language and it is very easy to code. It is also a developer-friendly language.

- **Free and Open Source:** Python is freely available. You can download it from the Python Official Website. Secondly, it is open-source. This means that its source code is available to the public. You can download it, change it, use it, and distribute it.

- **Object-Oriented Language:** One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects, modularization etc.

- **GUI Programming Support:** Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python.

- **High-Level Language:** Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

- **Extensible feature:** If needed, you can write some of your Python code in other languages like **C++**. This makes Python an extensible language, meaning that it can be extended to other languages.

- **Python is Portable language:** Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

- **Python is Integrated language:** Python is also an Integrated language because we can easily integrated python with other languages like c, c++, etc.

- **Interpreted Language:** Python is an Interpreted Language because Python code is executed line by line at a time. Like other languages C, C++, Java, etc. there is no need to compile entire python code, this makes it easier to debug the code.

- **Large Standard Library**: Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, web browsers, etc.

- **Dynamically Typed Language:** Python is dynamically-typed. This means that the type for a value is decided at runtime, not in advance. This is why we don't need to specify the type of data while declaring it.

INTRODUCTION TO MYSQL (RDBMS)

A **database** is a collection of information related to a particular subject or purpose, such as tracking customer orders or maintaining a product collection. Using any RDBMS application software like MS SQL Server, MySQL, Oracle, Sybase etc, all information can be managed from a single database file. Within the file, data can be divided into separate storage containers called tables. Data can be retrieved using queries.

A table is a collection of data about a specific topic, such as products or suppliers. Using a separate table for each topic means you can store that data only once, which makes your database more efficient and reduces data-entry errors. Table organises data into columns (called fields) and rows (called records).

A **Primary key** is one or more fields whose value or values uniquely identify each record in a table. In a relationship, a primary key is used to refer to specific record in one table from another table. A foreign key is a column in a relational database table that provides a link between data in two tables. It acts as a cross-reference between tables because it references the primary key of another table, thereby establishing a link between them.

Role of RDBMS Application Program:

A computer database works as a electronic filing system, which has a large number of ways of cross-referencing, and this allows the user many different ways in which to re-organize and retrieve data. A database can handle business inventory, accounting and filing and use the information in its files to prepare summaries, estimates and other reports. The management of data in a database system is done by means of a general-purpose software package called a Database Management System (DBMS). Some commercially available DBMS are MS SQL Server, MS ACCESS, INGRES, ORACLE, and Sybase. A database management system, therefore, is a combination of hardware and software that can be used to set up and monitor a database, and can manage the updating and retrieval of database that has been stored in it. Most of the database management systems have the following capabilities:

- Creating of a table, addition, deletion, modification of records.
- Retrieving data collectively or selectively.
- The data stored can be sorted or indexed at the user's discretion and direction.
- Various reports can be produced from the system. These may be either standardized report or that may be specifically generated according to specific user definition.
- Mathematical functions can be performed and the data stored in the database can be manipulated with these functions to perform the desired calculations.
- To maintain data integrity and database use.

The management of data in a database system is done by means of a general-purpose software package called a Database Management System (DBMS). Some commercially available RDBMS are MS SQL Server, MS ACCESS, INGRES, ORACLE, and Sybase. MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.

The Main Features of MySQL

- MySQL is written in C and C++.
- **Easy to use:** MySQL is easy to use. We have to get only the basic knowledge of SQL. We can build and interact with MySQL by using only a few simple SQL statements.
- **Secure:** It consists of a solid data security layer that protects sensitive data from intruders. Also, passwords are encrypted in MySQL.
- **Client/ Server Architecture**: It follows the working of a client/server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they can query data, save changes, etc.
- **Free to download:** It is free to use so that we can download it from MySQL official website without any cost.

- **Speed:** It is considered to be one of the very fast database languages, backed by a large number of the benchmark test.
- **High Flexibility:** It supports a large number of embedded applications, which makes it very flexible.
- **Compatible on many operating systems:** It is compatible to run on many operating systems, like Novell NetWare, Windows, Linux, many varieties of UNIX etc.
- **Allows roll-back:** It allows transactions to be rolled back, commit, and crash recovery.
- **Memory efficiency:** It's efficiency is high because it has a very low memory leakage problem.
- **High Performance:** It is faster, more reliable, and cheaper because of its unique storage engine architecture. It provides very high-performance results in

- comparison to other databases without losing an essential functionality of the software. It has fast loading utilities because of the different cache memory.
- **High Productivity:** MySQL uses Triggers, Stored procedures, and views that allow the developer to give higher productivity.
- **Platform Independent:** It can be downloaded, installed, and executed on most of the available operating systems.
- **Supports large databases:** MySQL Server can be used with databases that contain 50 million records. There are users who use MySQL Server with 200,000 tables and about 50,000,000 rows.

PROJECT SYNOPSIS

1. **Title of the Project :** Mobile shop Management system

2. **Introduction** :

   This project is aimed at developing a software application that depicts mobile shop management and maintain the inventory . Using this software, companies can improve the efficiency of those services. This application involves all the features of the inventory management.

3. **Objective of the Project:**

   This software helps user to find different mobiles, their features, and new updates easily. It is designed such a way that one can view all the mobile models based on their requirement criteria . The software will help in easy maintaining and updating products by the administrator, also for quick and easy comparison of different products by the customers.

4. **Scope of the Project:**

   This system will reduce the manual operation required to maintain all the records of booking information. And also generates the various reports for analysis. Main concept of the project is to enter transaction reports and to maintain customer records. Hence this software can be used in any mobile showroom to maintain the record easily.

MODULES AND DESCRIPTION

The modules used in this software are as follows:

1. **Login:** This module has 2 options and we have to select either ADMIN or USER. The admin has all the rights in the software including updating the status of the site. The Other fields in login are username and password. If the username and password are correct then it is directed accordingly.

2. **New User:** This module is for the users who do not have their account. Here the user is allowed to create an account to login. The account creation is done by filling the registration form with user details such as name, phone, email etc.

3. **Add New phone**: This module is for the Admin users who can update the nw phone details in the inventory. Here the Admin user is allowed to create new entry or record for new phone . Update it's selling price , quantity, RAM ,Memory capacity , Front and Rear Camera details

4. **Search:** This module helps the user to ease the search based on its budget or interest. the search can be done on different categories like mobile model name model number, colour, price etc.

5. **Remove User :** This Module helps to remove the user from the Database

PYTHON SOURCE CODE

```
from tkinter import Label,Entry,Button,END,StringVar,Tk
from tkinter import messagebox
import os
py=os.sys.executable
#import MySQLdb.connector
import mysql.connector
from mysql.connector import Error

#creating window
class Lib(Tk):
  def __init__(self):
    super().__init__()
    self.a = StringVar()
    self.b = StringVar()
    self.maxsize(1200, 700)
    self.minsize(1200, 700)
    self.configure(bg="gray")
    self.title("KPJ MOBILESHOP MANAGEMENT SYSTEM")


#verifying input
    def chex():
      if len(self.user_text.get()) < 0:
        messagebox.showinfo(" INVALID USERNAME OR PASSWORD" )
      elif len(self.pass_text.get()) < 0:
        messagebox.showinfo(" INVALID USERNAME OR PASSWORD")
      else:
        try:
          conn = mysql.connector.connect(host='localhost',
                    database='mobile',
                    user='root',
                    password='root')
          print("connection established")
          cursor = conn.cursor()
          user = self.user_text.get()
          password = self.pass_text.get()
          print(user,password)
          cursor.execute('Select * from admin where user= %s AND password
= %s ',(user,password))
          pc = cursor.fetchone()
          print(pc)
          if pc:
            self.destroy()
            os.system('%s %s' % (py, 'options.py'))
          else:
            print(pc)
            messagebox.showinfo('Error',  'Username and password not
found')
            self.user_text.delete(0, END)
            self.pass_text.delete(0, END)
        except Error:
          messagebox.showinfo('Error',"Something  Goes Wrong,Try
restarting",Error)

    def check():

          self.label = Label(self, text="LOGIN", bg = 'gray' , fg = 'black',
font=("courier-new", 24,'bold'))
          self.label.place(x=550, y=90)
          self.label1 = Label(self, text="User-Id" , bg = 'gray' , fg = 'black',
font=("courier-new", 18, 'bold'))
          self.label1.place(x=370, y=180)
          self.user_text = Entry(self, textvariable=self.a, width=45)
          self.user_text.place(x=480, y=190)
          self.label2 = Label(self, text="Password" , bg = 'gray' , fg = 'black',
font=("courier-new", 18, 'bold'))
```

```
                self.label2.place(x=340, y=250)
                self.pass_text = Entry(self, show='*', textvariable=self.b, width=45)
                self.pass_text.place(x=480, y=255)
                self.butt = Button(self, text="Login",bg ='white', font=10, width=8,
command=chex).place(x=580, y=300)
                self.label3 = Label(self, text="KPJ MOBILESHOP  MANAGEMENT
SYSTEM", bg='pink', fg='black', font=("courier-new", 24, 'bold'))
                self.label3.place(x=350, y=30)
        check()
Lib().mainloop()
from tkinter import *
from tkinter import messagebox
import re
from tkinter import ttk
import mysql.connector
from mysql.connector import Error
import os,sys
py=sys.executable

#creating window
class reg(Tk):
    def __init__(self):
        super().__init__()
        self.iconbitmap(r'libico.ico')
        self.maxsize(500, 417)
        self.minsize(500, 417)
        self.title('Add User')
        self.canvas = Canvas(width=500, height=417, bg='gray')
        self.canvas.pack()
#creating variables Please chech carefully
        u = StringVar()
        n = StringVar()
        p = StringVar()
    def insert():
        try:
            self.conn = mysql.connector.connect(host='localhost',
                                  database='mobile',
                                  user='root',
                                  password='root')
            print(self.conn)
            self.myCursor = self.conn.cursor()
            print(self.myCursor)
            print(u.get(), n.get(), p.get())
            self.myCursor.execute("Insert into admin(user,name,password)
values (%s,%s,%s)",[u.get(), n.get(), p.get()])
            print("executed")
            self.conn.commit()
            messagebox.showinfo("Done", "User Inserted Successfully")
            ask = messagebox.askyesno("Confirm", "Do you want to add another
user?")
            if ask:
                self.destroy()
                os.system('%s %s' % (py, 'Reg.py'))
            else:
                self.destroy()
                self.myCursor.close()
                self.conn.close()
        except Error:
            messagebox.showinfo("Error", "Something Goes Wrong",Error)
#label and input
    Label(self, text='User Details', bg='gray', fg='black', font=('Courier new',
25, 'bold')).place(x=130, y=22)
    Label(self, text='Username:', bg='gray', font=('Courier new', 10,
'bold')).place(x=70, y=82)
    Entry(self, textvariable=u, width=30).place(x=200, y=84)
    Label(self, text='Name:', bg='gray', font=('Courier new', 10,
'bold')).place(x=70, y=130)
        Entry(self, textvariable=n, width=30).place(x=200, y=132)
        Label(self, text='Password:', bg='gray', font=('Courier new', 10,
'bold')).place(x=70, y=180)
        Entry(self, textvariable=p, width=30).place(x=200, y=182)
        Button(self, text="Submit", width=15, command=insert).place(x=230,
y=220)
reg().mainloop()

from tkinter import *
from tkinter import messagebox
import mysql.connector
from mysql.connector import Error
#creating widow
class Rem(Tk):
    def __init__(self):
        super().__init__()
        self.iconbitmap(r'libico.ico')
        self.maxsize(400, 300)
        self.minsize(400, 300)
        self.title("Remove  User")
        self.canvas = Canvas(width=1366, height=768, bg='gray')
        self.canvas.pack()
        a = StringVar()
        def ent():
            if len(a.get()) ==0:
                messagebox.showinfo("Error","Please  Enter A Valid Id")
            else:
                d = messagebox.askyesno("Confirm",  "Are you sure you want to
remove the user?")
                if d:
                    try:
                        self.conn = mysql.connector.connect(host='localhost',
                                         database='mobile',
                                         user='root',
                                         password='root')
                        self.myCursor = self.conn.cursor()
                        self.myCursor.execute("Delete from admin where id =
%s",[a.get()])
                        self.conn.commit()
                        self.myCursor.close()
                        self.conn.close()
                        messagebox.showinfo("Confirm","User Removed Successfully")
                        a.set("")
                    except:
                        messagebox.showerror("Error","Something goes wrong")
        Label(self, text = "Enter User Id: ",bg='gray',fg='black',font=('Courier new',
15, 'bold')).place(x = 5,y = 40)
        Entry(self,textvariable = a,width = 37).place(x = 160,y = 44)
        Button(self, text='Remove', width=15, font=('arial', 10),command =
ent).place(x=200, y = 90)
Rem().mainloop()

from tkinter import *
from tkinter import ttk
from tkinter import messagebox
#from PIL import ImageTk,Image
import os,glob
import mysql.connector
from mysql.connector import Error

class Search(Tk):
    def __init__(self):
        super().__init__()
        f = StringVar()
        g = StringVar()
        self.title("Search Customer")
        self.maxsize(800,520)
        self.canvas = Canvas(width=1366, height=768, bg='gray')
        self.canvas.pack()
        self.iconbitmap(r'libico.ico')
        l1=Label(self,text="Search  Customer",bg='gray', font=("Courier
new",20,'bold')).place(x=290,y=40)
        l = Label(self, text="Search By",bg='gray', font=("Courier new", 15,
'bold')).place(x=180, y=100)


        def insert(data):
            self.listTree.delete(*self.listTree.get_children())
            for row in data:
                self.listTree.insert("","end",text = row[0], values =
(row[1],row[2],row[3]))


        def ge():
            if (len(self.entry.get())) == 0:
                messagebox.showinfo('Error',  'First select a item')
            elif (len(self.combo.get())) == 0:
                messagebox.showinfo('Error',  'Enter the '+self.combo.get())
            elif self.combo.get() == 'Name':
                try:
                    self.conn = mysql.connector.connect(host='localhost',
                                     database='mobile',
                                     user='root',
                                     password='root')
                    self.mycursor = self.conn.cursor()
                    name = self.entry.get()
                    self.mycursor.execute("Select  * from mobile.customer  where name
like %s",['%'+name+'%'])
                    pc = self.mycursor.fetchall()
                    if pc:
                        insert(pc)
                    else:
                        messagebox.showinfo("Oop's","Customer  Name not found")
                except Error:
                    messagebox.showerror("Error", "Something goes wrong")
            elif self.combo.get() == 'ID':
                try:
                    self.conn = mysql.connector.connect(host='localhost',
                                     database='library',
                                     user='root',
                                     password='')
                    self.mycursor = self.conn.cursor()
                    id = self.entry.get()
                    self.mycursor.execute("Select  * from mobile.customer  where
cust_id like %s", ['%' + id + '%'])
                    pc = self.mycursor.fetchall()
                    if pc:
                        insert(pc)
                    else:
                        messagebox.showinfo("Oop's",  "Id not found")
                except Error:
                    messagebox.showerror("Error", "Something goes wrong")


        self.b= Button(self,text="Find",width=8,font=("Courier
new",8,'bold'),command= ge )
        self.b.place(x=400,y=170)

self.combo=ttk.Combobox(self,textvariable=g,values=["Name","ID"],width=40,
state="readonly")
        self.combo.place(x = 310, y = 105)
        self.entry = Entry(self,textvariable=f,width=43)
        self.entry.place(x=310,y=145)
        self.la = Label(self, text="Enter",bg = 'gray', font=("Courier new", 15,
'bold')).place(x=180, y=140)
```

```
        def handle(event):
            if self.listTree.identify_region(event.x,event.y) == "separator":
                return "break"
        self.listTree = ttk.Treeview(self, height=13,columns=('Customer Name',
'Phone Number', 'Address'))
        self.vsb =
ttk.Scrollbar(self,orient="vertical",command=self.listTree.yview)
        self.listTree.configure(yscrollcommand=self.vsb.set)
        self.listTree.heading("#0", text='Customer ID', anchor='w')
        self.listTree.column("#0", width=100, anchor='w')
        self.listTree.heading("Customer Name", text='Customer Name')
        self.listTree.column("Customer Name", width=200, anchor='center')
        self.listTree.heading("Phone Number", text='Phone Number')
        self.listTree.column("Phone Number", width=200, anchor='center')
        self.listTree.heading("Address", text='Address')
        self.listTree.column("Address", width=200, anchor='center')
        self.listTree.place(x=40, y=200)
        self.vsb.place(x=743,y=200,height=287)
        ttk.Style().configure("Treeview", font=('Times new Roman', 15))

Search().mainloop()

from tkinter import *
from tkinter import messagebox
from tkinter import filedialog
import os
import sys
import mysql.connector
from mysql.connector import Error
py = sys.executable

#creating window
class Add(Tk):
    def __init__(self):
        super().__init__()
        self.iconbitmap(r'libico.ico')
        self.maxsize(500,417)
        self.minsize(500,417)
        self.title('Add Customer Details')
        self.canvas = Canvas(width=500, height=417, bg='gray')
        self.canvas.pack()
        n = StringVar()
        p = StringVar()
        a = StringVar()
#verifying input
        def asi():
            if len(n.get()) < 1:
                messagebox.showinfo("Oop's",  "Please Enter Customer Name")
            elif len(p.get()) < 1:
                messagebox.showinfo("Oop's","Please  Enter Customer Phone
Number")
            elif len(a.get()) < 1:
                messagebox.showinfo("Oop's",  "Please Enter Customer Address")
            else:
                try:
                    self.conn = mysql.connector.connect(host='localhost',
                                 database='mobile',
                                 user='root',
                                 password='root')
                    self.myCursor = self.conn.cursor()
                    name1 = n.get()
                    pn1 = p.get()
                    add1 = a.get()
                    self.myCursor.execute("Insert into
mobile.customer(name,phone_number,address)  values
(%s,%s,%s)",[name1,pn1,add1])
                    self.conn.commit()
                    messagebox.showinfo("Done","Customer  Details Inserted
Successfully")
                    ask = messagebox.askyesno("Confirm","Do  you want to add another
Customer?")
                    if ask:
                     self.destroy()
                     os.system('%s %s' % (py, 'Add_Customer.py'))
                    else:
                     self.destroy()
                     self.myCursor.close()
                     self.conn.close()
                except Error:
                    messagebox.showerror("Error","Something  goes wrong",Error)

        # label and input box
        Label(self, text='Customer Details',bg='gray', fg='white', font=('Courier
new', 25, 'bold')).pack()
        Label(self, text='Customer Name:',bg='gray', font=('Courier new', 10,
'bold')).place(x=70, y=82)
        Entry(self, textvariable=n, width=30).place(x=200, y=84)
        Label(self, text='Phone Number:',bg='gray', font=('Courier new', 10,
'bold')).place(x=70, y=130)
        Entry(self, textvariable=p, width=30).place(x=200, y=132)
        Label(self, text='Address:',bg='gray', font=('Courier new', 10,
'bold')).place(x=70, y=180)
        Entry(self, textvariable=a, width=30).place(x=200, y=182)
        Button(self, text="Submit",width = 15,command=asi).place(x=230,
y=220)

Add().mainloop()

from tkinter import *
from tkinter import messagebox
import os
import sys
from tkinter import ttk

import mysql.connector
from mysql.connector import Error
py=sys.executable

#creating window
class MainWin(Tk):
    def __init__(self):
        super().__init__()
        self.iconbitmap(r'libico.ico')
        self.configure(bg='gray')
        self.canvas = Canvas(width=1366, height=768, bg='gray')
        self.canvas.pack()
        self.maxsize(1320, 768)
        self.minsize(1320,768)
        self.state('zoomed')
        self.title('KPJ MOBILESHOP  MANAGEMENT SYSTEM')
        self.a = StringVar()
        self.b = StringVar()
        self.mymenu = Menu(self)
#calling scripts
        def a_s():
            os.system('%s %s' % (py, 'Add_Customer.py'))

        def a_b():
            os.system('%s %s' % (py, 'Add_Phone.py'))

        def r_b():
            os.system('%s %s' % (py, 'remove_book.py'))

        def r_s():
            os.system('%s %s' % (py, 'Remove_student.py'))
        def sea():
            os.system('%s %s' % (py,'Search.py'))

        def log():
            conf = messagebox.askyesno("Confirm",  "Are you sure you want to 
Logout?")
            if conf:
             self.destroy()
             os.system('%s %s' % (py, 'Main.py'))



      # def handle(event):
        #     if self.listTree.identify_region(event.x,event.y)  == "separator":
        #         return "break"
        def add_user():
            os.system('%s %s' % (py, 'Reg.py'))
        def rem_user():
            os.system('%s %s' % (py, 'Rem.py'))
        def sest():
            os.system('%s %s' % (py,'Search_Customer.py'))

#creating table

        self.listTree = 
ttk.Treeview(self,height=14,columns=('Brand','Model','RAM','Memory','Camer
a','RCamera','Price','Quantity'))
        self.vsb = 
ttk.Scrollbar(self,orient="vertical",command=self.listTree.yview)
        self.hsb = 
ttk.Scrollbar(self,orient="horizontal",command=self.listTree.xview)

self.listTree.configure(yscrollcommand=self.vsb.set,xscrollcommand=self.hsb.
set)
        self.listTree.heading("#0", text='ID')
        self.listTree.column("#0", width=150,minwidth=150,anchor='center')
        self.listTree.heading("Brand", text='Brand')
        self.listTree.column("Brand", width=150, minwidth=150,anchor='center')
        self.listTree.heading("Model", text='Model')
        self.listTree.column("Model", width=100, minwidth=100,anchor='center')
        self.listTree.heading("RAM", text='RAM')
        self.listTree.column("RAM", width=100, minwidth=100,anchor='center')
        self.listTree.heading("Memory", text='Memory')
        self.listTree.column("Memory", width=100,
minwidth=100,anchor='center')
        self.listTree.heading("Camera", text='Front Camera')
        self.listTree.column("Camera", width=100,
minwidth=100,anchor='center')
        self.listTree.heading("RCamera", text='Rear Camera')
        self.listTree.column("RCamera", width=100,
minwidth=100,anchor='center')
        self.listTree.heading("Price", text='Price')
        self.listTree.column("Price", width=100, minwidth=100,anchor='center')
        self.listTree.heading("Quantity", text='Avilable Quantity')
        self.listTree.column("Quantity", width=100, minwidth=100,
anchor='center')
        self.listTree.place(x=240,y=360)
        self.vsb.place(x=1238,y=361,height=300)
        #self.hsb.place(x=320,y=650,width=700)
        ttk.Style().configure("Treeview",font=('Times new Roman',15))

        list1 = Menu(self)
        list1.add_command(label="Customer", command=a_s)
        list1.add_command(label="Mobilephone Inventory", command=a_b)

        list3 = Menu(self)
        list3.add_command(label = "Add User",command = add_user)
```

```
    list3.add_command(label = "Remove User",command = rem_user)


    self.mymenu.add_cascade(label='Add',  menu=list1)
    self.mymenu.add_cascade(label = 'Admin Tools', menu = list3)

    self.config(menu=self.mymenu)

    def ser():
      if(len(self.studid.get())==0):
        messagebox.showinfo("Error",  "Empty Field!")
      else:

       try:
        conn = mysql.connector.connect(host='localhost',
                          database='mobile',
                          user='root',
                          password='root')
        cursor = conn.cursor()
        brand = self.studid.get()
        print(brand)
        cursor.execute("Select *  from mobile.phone where brand LIKE
%s",['%'+brand+'%'])
        pc = cursor.fetchall()
        if pc:
          self.listTree.delete(*self.listTree.get_children())
          for row in pc:
            self.listTree.insert("",'end',text=row[0]  ,values =
(row[1],row[2],row[3],row[4], row[5], row[6], row[7], row[8]))
        else:
          messagebox.showinfo("Error", "Mobilephone details not found in
the Inventory")
       except Error:
        #print(Error)
```

```
              messagebox.showerror("Error","Something  Goes Wrong")
        def ent():
            if (len(self.bookid.get())  == 0):
                messagebox.showinfo("Error",  "Empty Field!")
            else:
             try:
                self.conn = mysql.connector.connect(host='localhost',
                                         database='mobile',
                                         user='root',
                                         password='root')
                self.myCursor = self.conn.cursor()
                ram = self.bookid.get()
                print(ram)
                self.myCursor.execute("Select  *  from mobile.phone where ram LIKE
%s",['%'+ram+'%'])
                self.pc = self.myCursor.fetchall()
                if self.pc:
                    self.listTree.delete(*self.listTree.get_children())
                    for row in self.pc:
                        self.listTree.insert("", 'end', text=row[0],values=(row[1], row[2],
row[3], row[4], row[5], row[6], row[7], row[8]))
                else:
                    messagebox.showinfo("Error",  "Mobilephone details not found in
the Inventory")
             except Error:
                messagebox.showerror("Error", "Something  Goes Wrong")

        def check():
            try:
                conn = mysql.connector.connect(host='localhost',
                                         database='mobile',
                                         user='root',
                                         password='root')
                mycursor = conn.cursor()
                mycursor.execute("Select * from admin")
                z = mycursor.fetchone()
                if not z:
                    messagebox.showinfo("Error", "Please Register A user")
                    x = messagebox.askyesno("Confirm","Do you want to register a
user")
                    if x:
                        self.destroy()
                        os.system('%s %s' % (py, 'Reg.py'))
                else:
                    #label and input box
                    self.label3 = Label(self, text='KPJ MOBILESHOP MANAGEMENT
SYSTEM',fg='white',bg="pink" ,font=('Courier new', 30, 'bold'))
                    self.label3.place(x=350, y=22)
                    self.label4 = Label(self, text="ENTER MOBILE BRAND
NAME",bg="gray", font=('Courier new', 18, 'bold'))
                    self.label4.place(x=75, y=107)
                    self.studid = Entry(self, textvariable=self.a, width=90)
                    self.studid.place(x=405, y=110)
                    self.srt = Button(self, text='Search', width=15, font=('arial',
10),command = ser).place(x=1000, y=106)
                    self.label5 = Label(self, text="ENTER THE RAM
CAPACITY",bg="gray", font=('Courier new', 18, 'bold'))
                    self.label5.place(x=75, y=150)
                    self.bookid = Entry(self, textvariable=self.b, width=90)
                    self.bookid.place(x=405, y=160)
                    self.brt = Button(self, text='Find', width=15, font=('arial',
10),command = ent).place(x=1000, y=150)
                    self.label6 = Label(self, text="MOBILEPHONE INFORMATION
DETAILS",fg="white",bg="blue", font=('Courier new', 15, 'underline', 'bold'))
                    self.label6.place(x=560, y=300)
                    self.button = Button(self, text='Search Customer',
width=25,bg="green", font=('Courier new', 10),
command=sest).place(x=240,y=250)
```

```
            self.button = Button(self, text='Search Mobile Phone',
width=25,bg="yellow", font=('Courier new', 10),
command=sea).place(x=1000,y=250)
            self.brt = Button(self, text="LOGOUT", width=15,bg="red",
font=('Courier new', 10), command=log).place(x=1150, y=105)
        except Error:
            messagebox.showerror("Error", "Something Goes Wrong")
    check()

MainWin().mainloop()

from tkinter import *
from tkinter import ttk
from tkinter import messagebox
import mysql.connector
from mysql.connector import Error
class Search(Tk):
  def __init__(self):
    super().__init__()
    f = StringVar()
    g = StringVar()
    self.title("Search Mobile Phone Details")
    self.maxsize(1000,800)
    self.minsize(1000,800)
    self.canvas = Canvas(width=1200, height=800, bg='gray')
    self.canvas.pack()
    self.iconbitmap(r'libico.ico')
    l1=Label(self,text="Search Mobile Phone",bg='gray', font=("Courier
new",20,'bold')).place(x=290,y=20)
    l = Label(self, text="Search By",bg='gray', font=("Courier new", 15,
'bold')).place(x=60, y=96)
    def insert(data):
      self.listTree.delete(*self.listTree.get_children())
      for row in data:
            self.listTree.insert("", 'end', text=row[0], values=(row[1], row[2],
row[3], row[4], row[5]))
      def ge():
         if (len(g.get())) == 0:
            messagebox.showinfo('Error', 'First select a item')
         elif (len(f.get())) == 0:
            messagebox.showinfo('Error', 'Enter the '+g.get())
         elif g.get() == 'Brand':
            try:
               self.conn = mysql.connector.connect(host='localhost',
                                 database='mobile',
                                 user='root',
                                 password='root')
               self.mycursor = self.conn.cursor()
               self.mycursor.execute("Select * from mobile.phone where brand
LIKE %s",['%'+f.get()+'%'])
               self.pc = self.mycursor.fetchall()
               if self.pc:
                  insert(self.pc)
               else:
                  messagebox.showinfo("Oop's","Either Brand Name is incorrect or
it is not available")
            except Error:
               messagebox.showerror("Error","Something goes wrong")
         elif g.get() == 'RAM':
            try:
               self.conn = mysql.connector.connect(host='localhost',
                                 database='mobile',
                                 user='root',
                                 password='root')
               self.mycursor = self.conn.cursor()
               self.mycursor.execute("Select * from mobile.phone where ram LIKE
%s", ['%'+f.get()+'%'])
               self.pc = self.mycursor.fetchall()
```

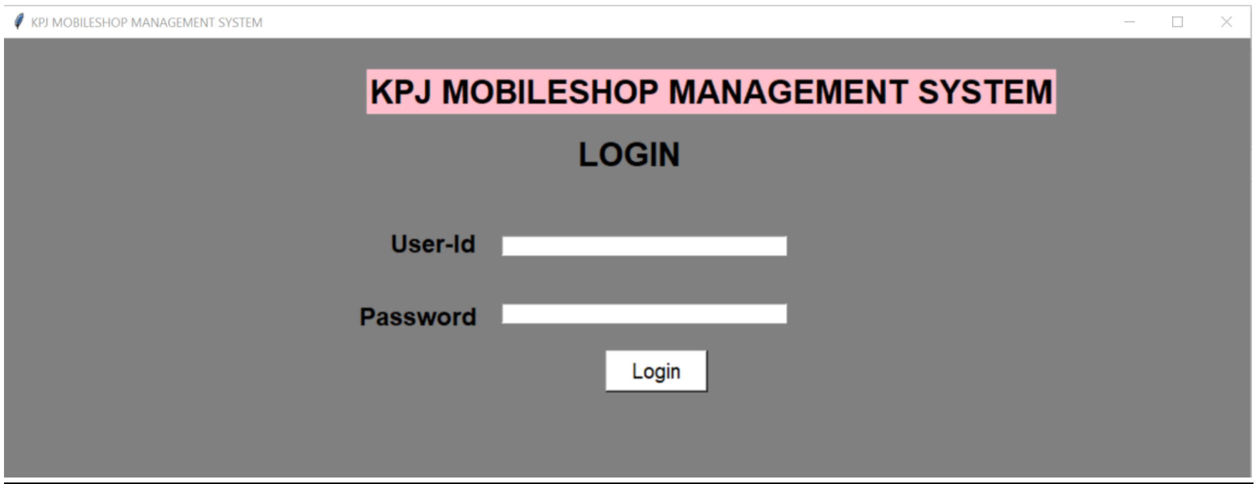
```
                if self.pc:
                    insert(self.pc)
                else:
                    messagebox.showinfo("Oop's","RAM  size not avilable")
            except Error:
                messagebox.showerror("Error","Something  goes wrong")
        elif g.get() == 'Memory':
            try:
                self.conn = mysql.connector.connect(host='localhost',
                                    database='mobile',
                                    user='root',
                                    password='root')
                self.mycursor = self.conn.cursor()
                self.mycursor.execute("Select  * from mobile.phone  where memory
LIKE %s", ['%'+f.get()+'%'])
                self.pc = self.mycursor.fetchall()
                if self.pc:
                    insert(self.pc)
                else:
                    messagebox.showinfo("Oop's","Internal  Memory Size not
available")
            except Error:
                messagebox.showerror("Error","Something  goes wrong")
    b=Button(self,text="Find",width=15,bg='green',font=("Courier
new",10,'bold'),command=ge).place(x=460,y=148)

c=ttk.Combobox(self,textvariable=g,values=["Brand","RAM","Memory"],width
=40,state="readonly").place(x = 180, y = 100)
    en = Entry(self,textvariable=f,width=43).place(x=180,y=155)
    la = Label(self, text="Enter",bg='gray', font=("Courier new", 15,
'bold')).place(x=100, y=150)

    def handle(event):
        if self.listTree.identify_region(event.x,event.y)  == "separator":
            return "break"


    self.listTree = ttk.Treeview(self, height=13,columns=('Brand Name',
'Model','RAM','Memory', 'Availabil Quantity'))
    self.vsb =
ttk.Scrollbar(self,orient="vertical",command=self.listTree.yview)
    self.listTree.configure(yscrollcommand=self.vsb.set)
    self.listTree.heading("#0", text='Phone ID', anchor='center')
    self.listTree.column("#0", width=120, anchor='center')
    self.listTree.heading("Brand Name", text='Brand Name')
    self.listTree.column("Brand Name", width=200, anchor='center')
    self.listTree.heading("Model", text='Model')
    self.listTree.column("Model", width=200, anchor='center')
    self.listTree.heading("RAM", text='RAM')
    self.listTree.column("RAM", width=100, anchor='center')
    self.listTree.heading("Memory", text='Memory')
    self.listTree.column("Memory", width=100, anchor='center')
    self.listTree.heading("Availabil Quantity", text='Availabil Quantity')
    self.listTree.column("Availabil Quantity", width=100, anchor='center')
    self.listTree.bind('<Button-1>', handle)
    self.listTree.place(x=40, y=200)
    self.vsb.place(x=820,y=200,height=287)
    ttk.Style().configure("Treeview", font=('Times new Roman', 15))
    Search().mainloop()
```

SQL QUERY

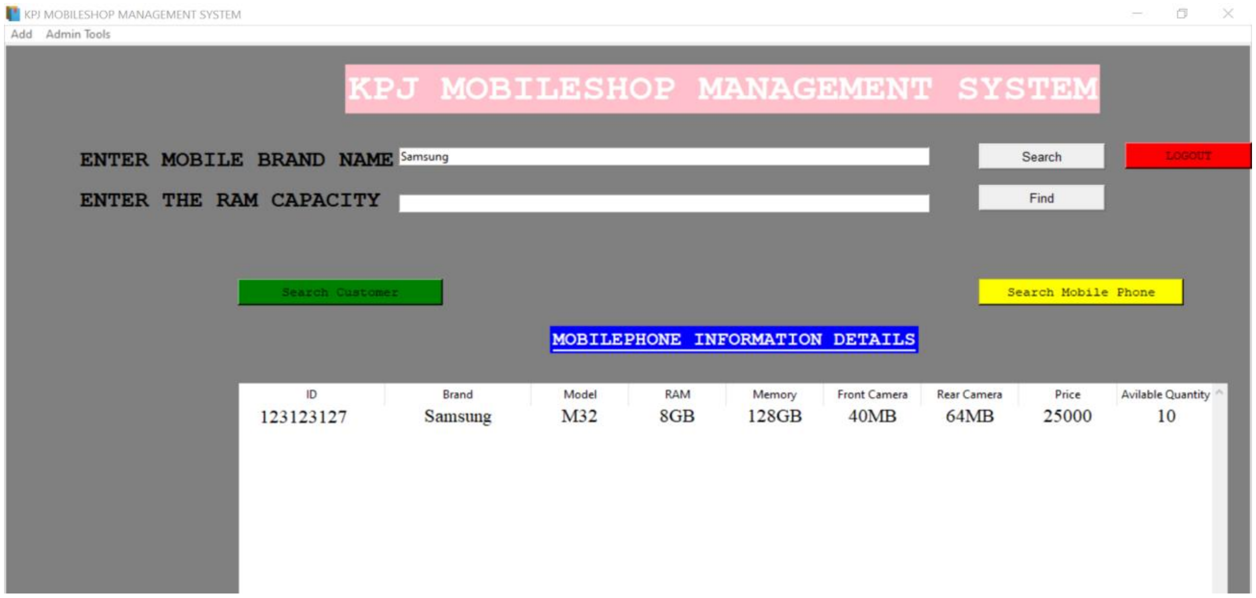
```
CREATE TABLE mobile.admin (
  `id` int(11) NOT NULL,
  `user` varchar(30) NOT NULL,
  `name` text NOT NULL,
  `password` text NOT NULL
);

ALTER TABLE mobile.admin   ADD PRIMARY KEY (`id`);
ALTER TABLE mobile.admin   MODIFY `id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=14;

CREATE TABLE mobile.phone(
  phone_id int(11) NOT NULL,
  brand varchar(300) NOT NULL,
  model varchar(300) NOT NULL,
  ram varchar(30) NOT NULL,
  memory varchar(30) NOT NULL,
  front_camera varchar(30) NOT NULL,
  rear_camera varchar(30) NOT NULL,
  price varchar(30) NOT NULL,
  quantity varchar(30) NOT NULL
);
ALTER TABLE mobile.phone ADD PRIMARY KEY (phone_id);
ALTER TABLE mobile.phone  MODIFY phone_id int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=202200;

CREATE TABLE mobile.customer (
  cust_id int(11) NOT NULL,
  name varchar(300) NOT NULL,
  phone_number varchar(30) NOT NULL,
  address text NOT NULL
);

ALTER TABLE mobile.customer  ADD PRIMARY KEY (cust_id);
ALTER TABLE mobile.customer  MODIFY cust_id int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=4;
```

SCREENSHOTS OF EXECUTION

Admin Login Page


KPJ MOBILESHOP MANAGEMENT SYSTEM
KPJ MOBILESHOP MANAGEMENT SYSTEM
LOGIN
User-Id
Password
Login

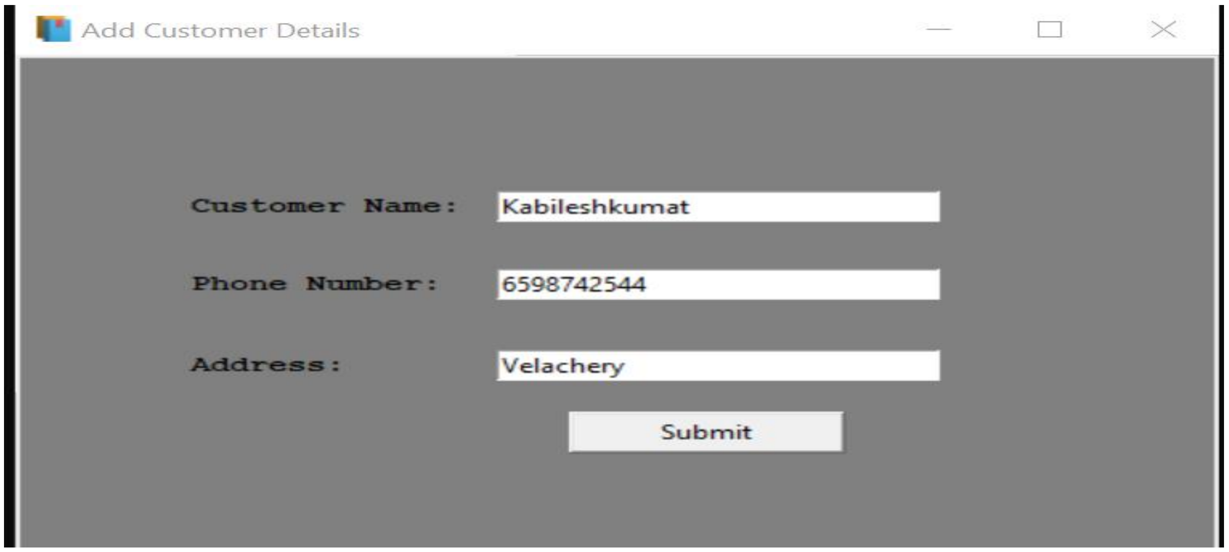
Main Page



ENTER MOBILE BRAND NAME: Samsung

ID	Brand	Model	RAM	Memory	Front Camera	Rear Camera	Price	Avilable Quantity
123123127	Samsung	M32	8GB	128GB	40MB	64MB	25000	10

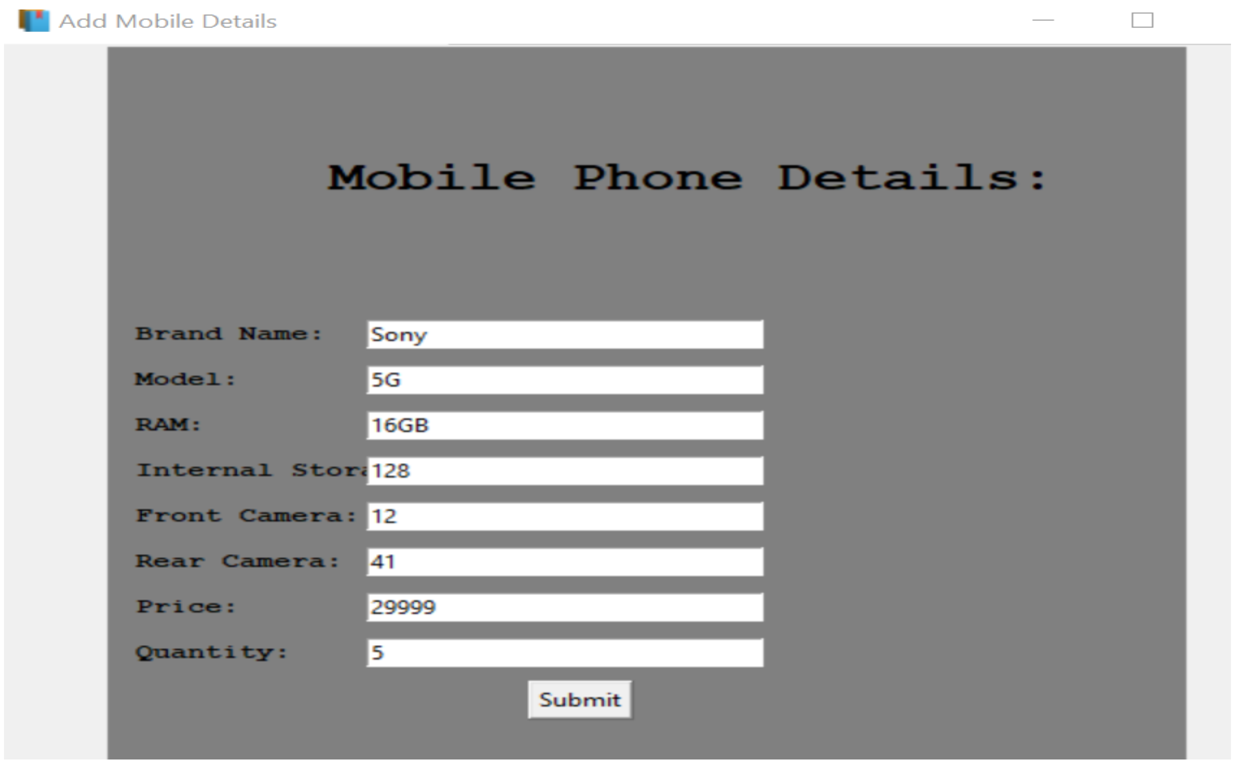
Add New Customer Details Page



Add Customer Details

Customer Name:	Kabileshkumat
Phone Number:	6598742544
Address:	Velachery

Add New Mobile Page

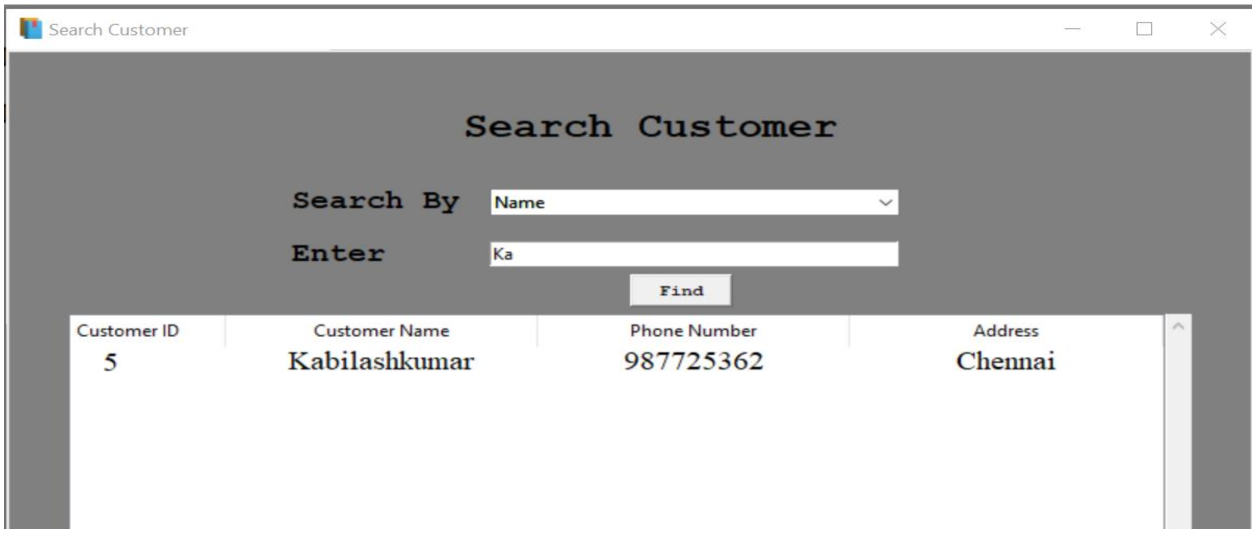


Add Mobile Details

Mobile Phone Details:

Brand Name:	Sony
Model:	5G
RAM:	16GB
Internal Stor	128
Front Camera:	12
Rear Camera:	41
Price:	29999
Quantity:	5

Search Customer page



Search Customer

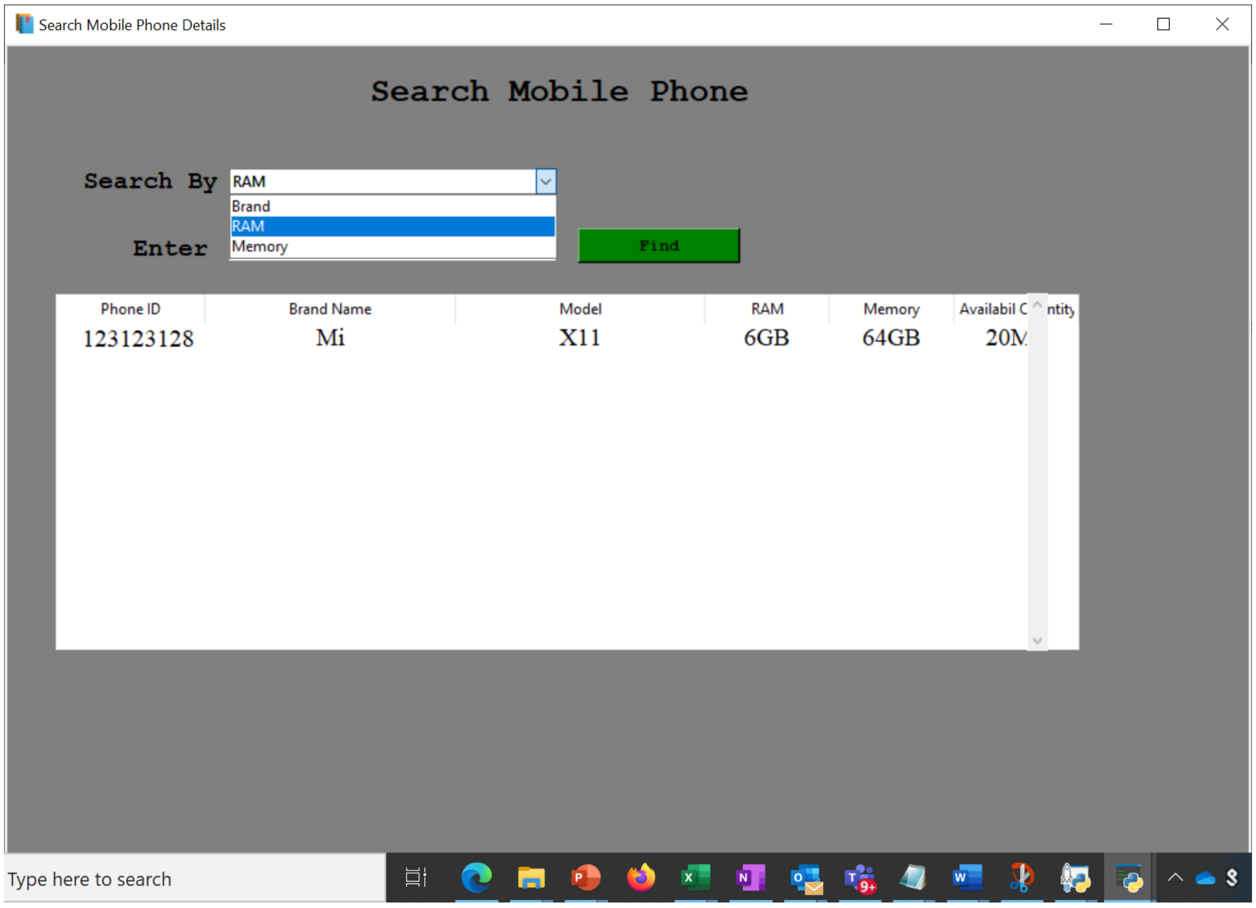
Search Customer

Search By: Name

Enter: Ka

Customer ID	Customer Name	Phone Number	Address
5	Kabilashkumar	987725362	Chennai

Search Mobile Phone



Search Mobile Phone Details

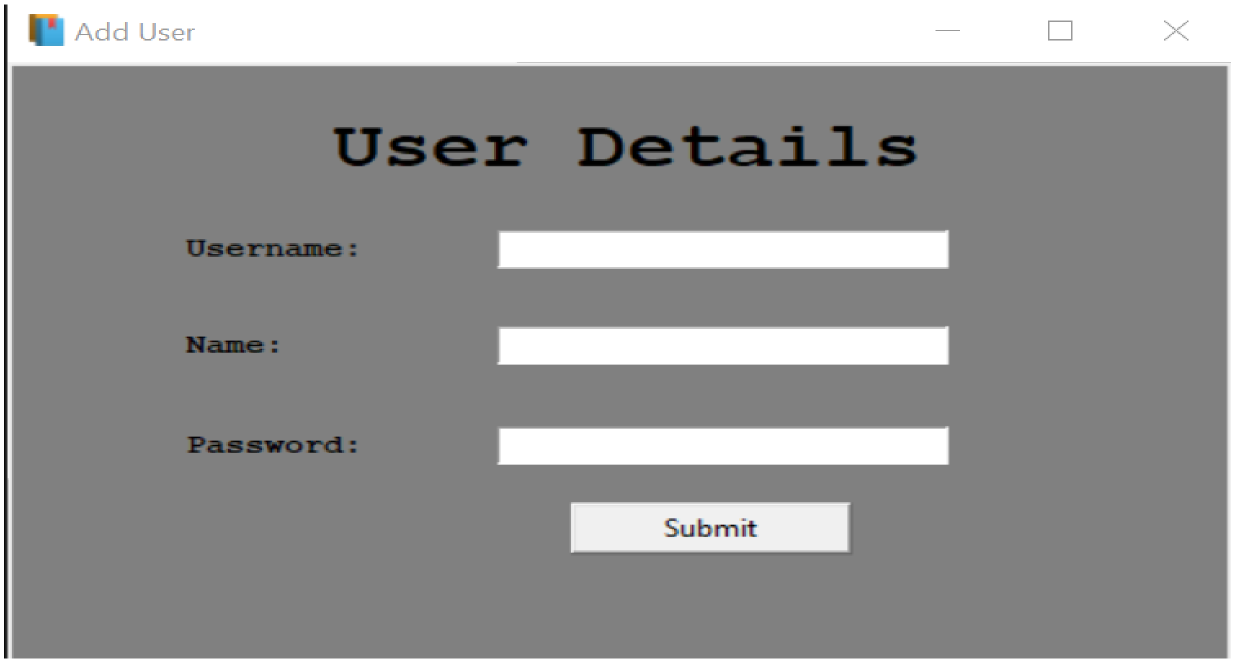
Search Mobile Phone

Search By: RAM

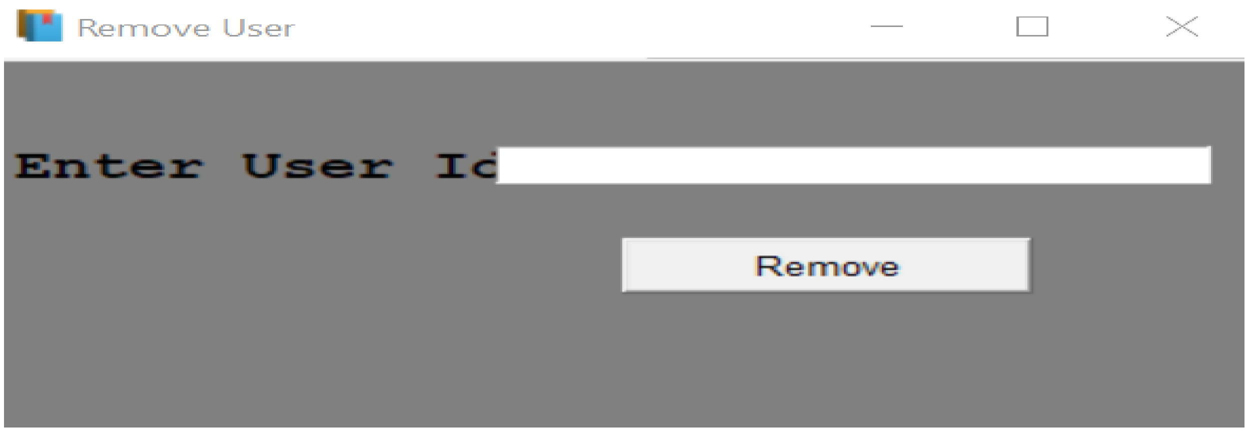
Enter

Phone ID	Brand Name	Model	RAM	Memory	Availabil C ntity
123123128	Mi	X11	6GB	64GB	20M

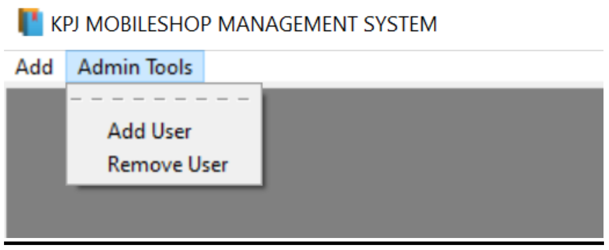
Add User Page

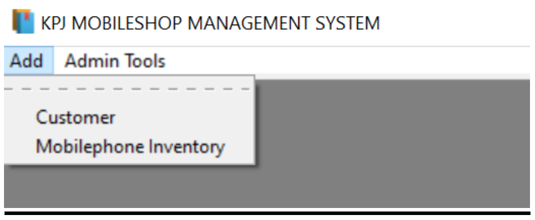


Remove User Page

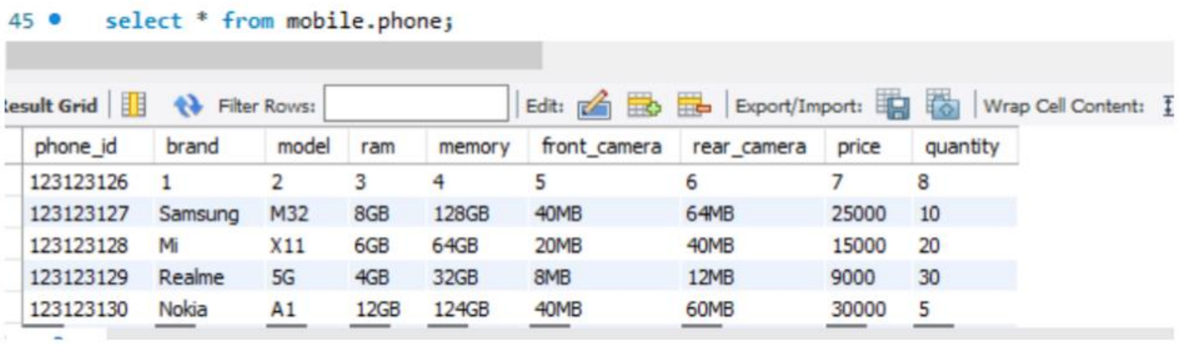


Menu Items



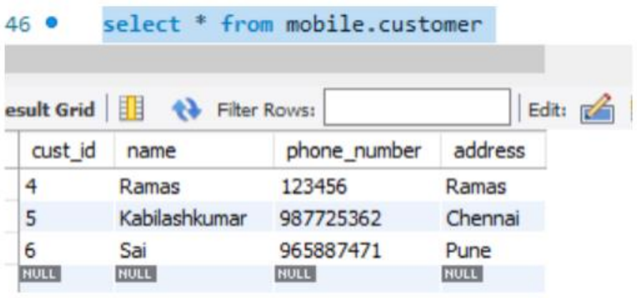


SQL Commandline Output



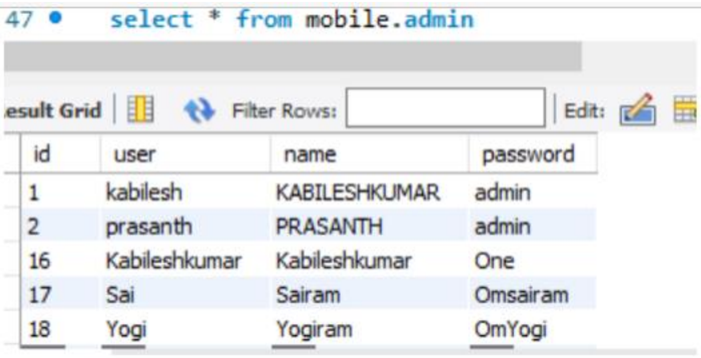
```
select * from mobile.phone;
```

phone_id	brand	model	ram	memory	front_camera	rear_camera	price	quantity
123123126	1	2	3	4	5	6	7	8
123123127	Samsung	M32	8GB	128GB	40MB	64MB	25000	10
123123128	Mi	X11	6GB	64GB	20MB	40MB	15000	20
123123129	Realme	5G	4GB	32GB	8MB	12MB	9000	30
123123130	Nokia	A1	12GB	124GB	40MB	60MB	30000	5



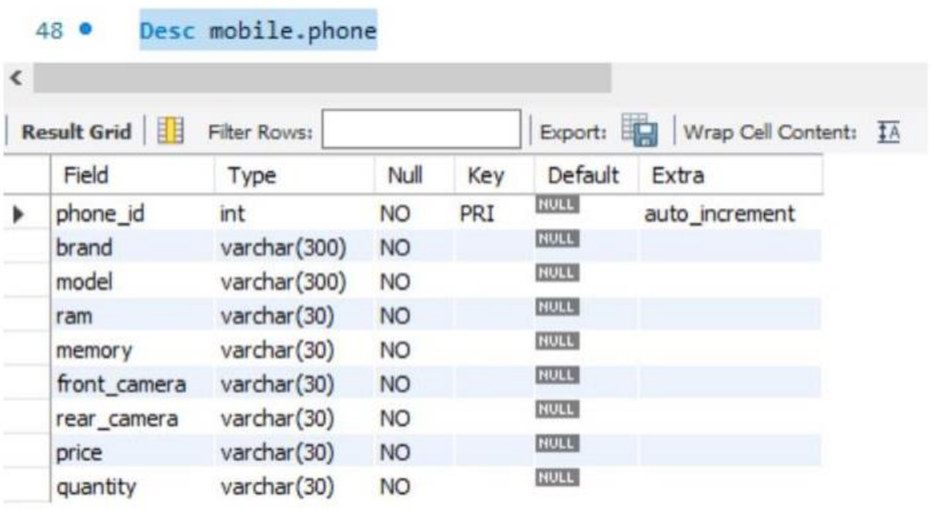
```
select * from mobile.customer
```

cust_id	name	phone_number	address
4	Ramas	123456	Ramas
5	Kabilashkumar	987725362	Chennai
6	Sai	965887471	Pune
NULL	NULL	NULL	NULL



```
select * from mobile.admin
```

id	user	name	password
1	kabilesh	KABILESHKUMAR	admin
2	prasanth	PRASANTH	admin
16	Kabileshkumar	Kabileshkumar	One
17	Sai	Sairam	Omsairam
18	Yogi	Yogiram	OmYogi



```
Desc mobile.phone
```

Field	Type	Null	Key	Default	Extra
phone_id	int	NO	PRI	NULL	auto_increment
brand	varchar(300)	NO		NULL	
model	varchar(300)	NO		NULL	
ram	varchar(30)	NO		NULL	
memory	varchar(30)	NO		NULL	
front_camera	varchar(30)	NO		NULL	
rear_camera	varchar(30)	NO		NULL	
price	varchar(30)	NO		NULL	
quantity	varchar(30)	NO		NULL	

SYSTEM REQUIREMENTS

HARDWARE:

- Proccesor: Pentium III and above
- Printer- to print the required documents of the project.
- Minimum memory - 2GB

SOFTWARE:

- Windows 7 or higher
- My-SQL server 5.5 or higher(as backend)
- Python idle 3.6 or higher or Spyder (as frontend).
- Microsoft Word 2010 or higher for documentation.

BIBLIOGRAPHY

In order to work on this project titled – Online Mobile Shopping, the following books and literature are referred by me during the various phases of development of the project.

- Computer Science with python - by Sumita Arora
- www.python.org/download
- www.py2exe.org
- www.mysql.org