Batch Name: Summer Internship 2018

Project Title- Retail Store Management System

Team ID-500054466

Team Member Details-

S.no	SAP ID	Name	Enrollment no
1	500054466	Pratyush Sharma	R103216072
2	500053452	Srishti Nigam	R103216100
3	500054431	Ayush Chaturvedi	R103216124

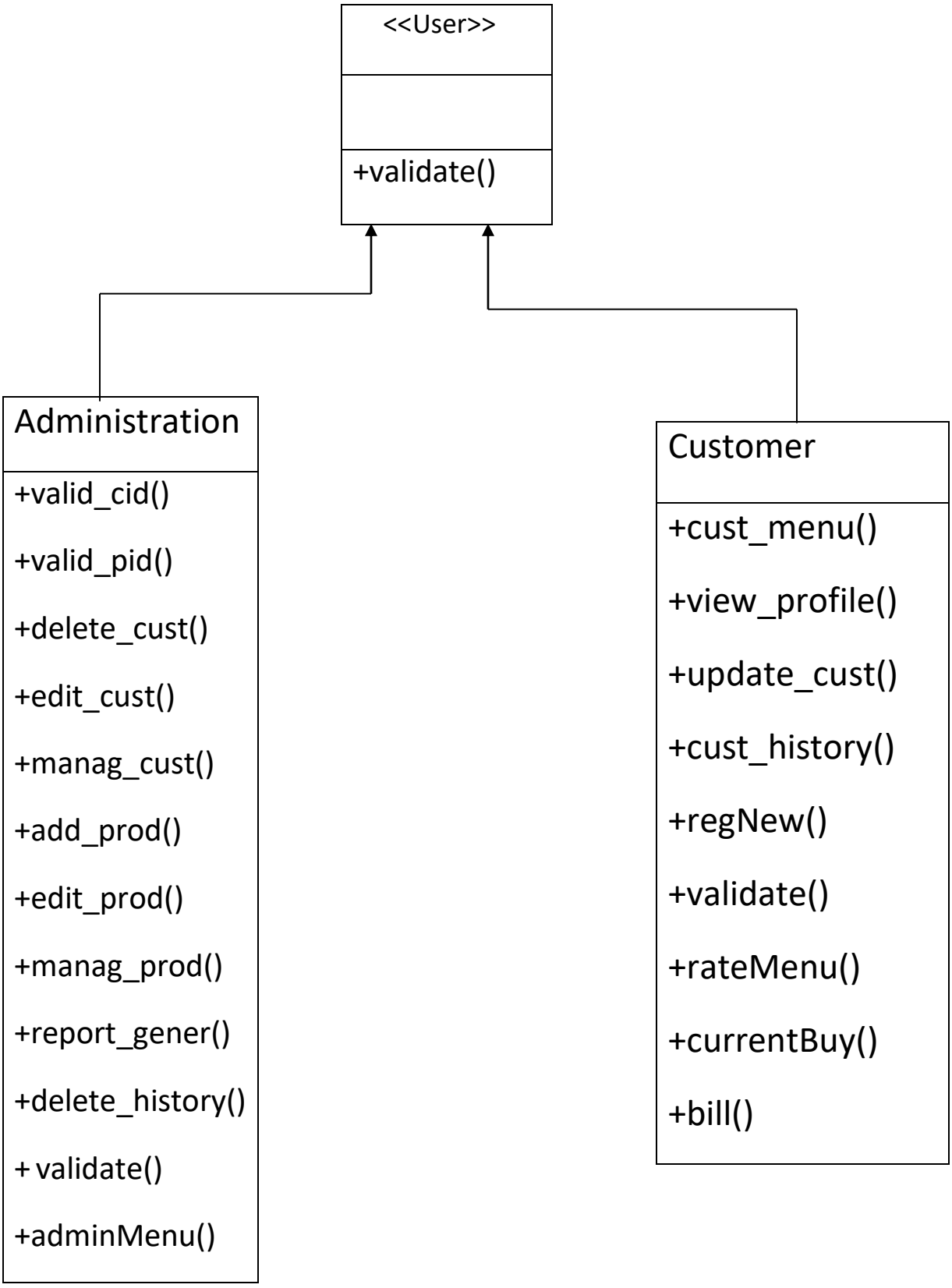
INTRODUCTION-

Scope of the project- Generalize software for the General store, Electronic Appliances Store, Medical Store or all type of store where management of customer and product are required.

Why Project is Made- In order to make our decisions on how to choose among a variety of options in Retail Store as a customer and functionality to admin to manage to customer and Product.

System Requirement-My SQL 6.3 and above, Database, Python Shell 3.6 and above.

UML diagram


<<User>>
+validate()
Administration
+valid_cid()
+valid_pid()
+delete_cust()
+edit_cust()
+manag_cust()
+add_prod()
+edit_prod()
+manag_prod()
+report_gener()
+delete_history()
+ validate()
+adminMenu()
Customer
+cust_menu()
+view_profile()
+update_cust()
+cust_history()
+regNew()
+validate()
+rateMenu()
+currentBuy()
+bill()

Modules Function Descriptions

User module -

- **Admin module –** This module show all functionalities provided to Admin.

1. +valid_cid()-This function check the validation of the entered Customer id with the Customer id from the database.

2. +valid_pid()-This function check the validation of the entered Product id with the Product id from the database

3. +delete_cust() –This Function helps the admin to delete the customer with the entered customer id

4. +edit_cust()-This function helps the Admin to edit the customer name ,date of registration and the phone number.

5. +manag_cust()-This Function helps the admin to to edit customer and to delete customer.

6. +add_prod()-This function helps the admin to add the product to the database with the given information like id ,name, category, GST ,and the Quantity .

7. +edit_prod()-This function helps the admin to edit the id, name, category, GST, and Quantity

8. +manag_prod()-This function helps the admin to add product and edit product to the database.

9. +report_gener()-This function shows the Report of the customer, product and the shopping history to the admin.

10. +delete_history()-This function helps the admin to Delete the one customer history and the history.

11. + validate()-Check the validation of the Admin

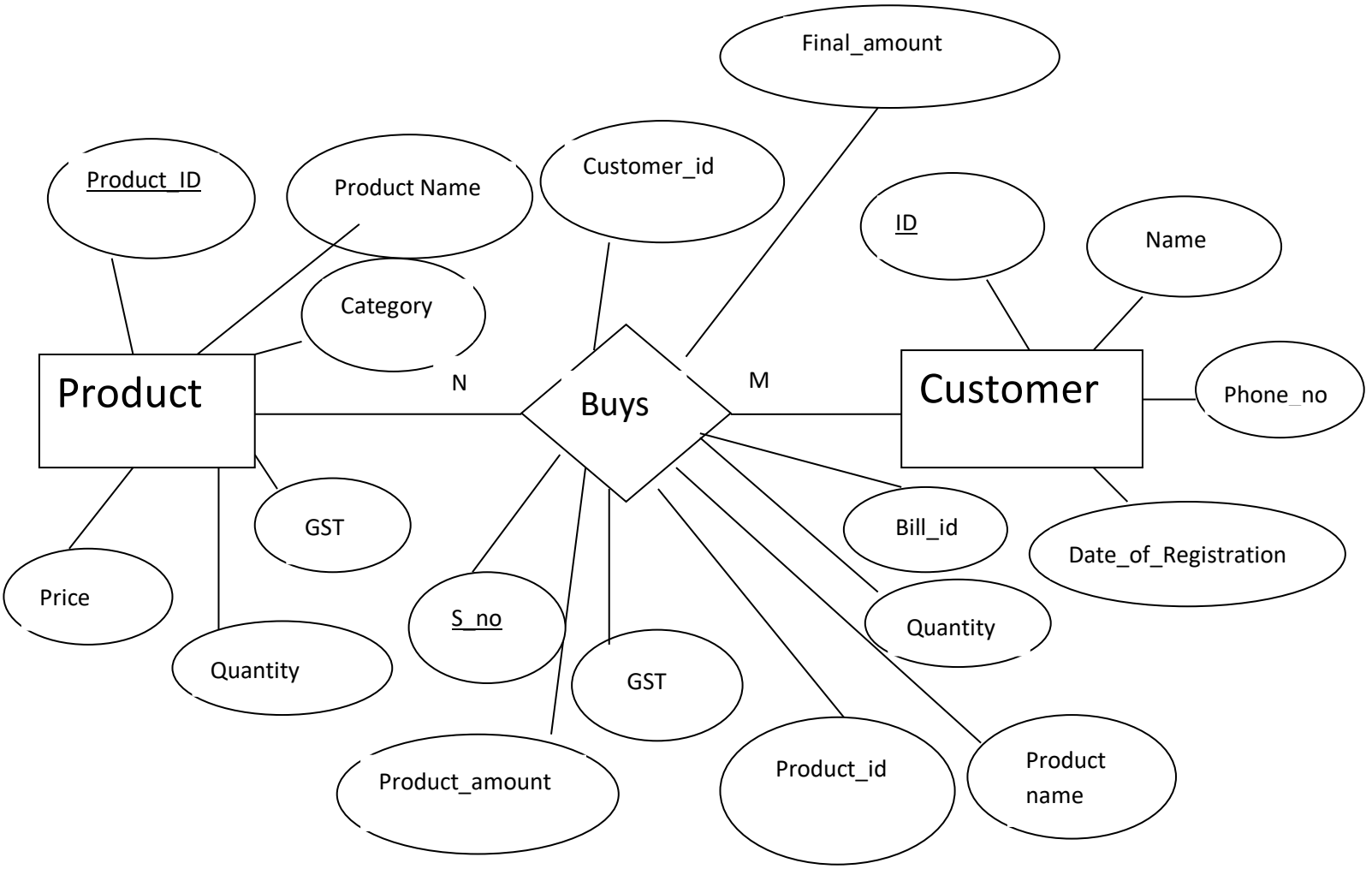
12. +adminMenu()-the function shows the admin menu with Manage custom ,Manage product, Report generation, Delete history, Logout as admin.

- **Customer Module –** This module show all functionalities provided to customer.

1. +cust_menu()-This function shows the customer to buy product ,view my customer info, update my account info and view shopping history with the entered customer id.
2. +view_profile()—Shows the profile of the customer with the valid customer id.
3. +update_cust()-This function give the customer to Edit his name and the phone number.

4. +cust_history()-This Function shows the customer shows the customer history.
5. +regNew()-This function is used to register the new customer and generates its customer id.
6. +validate()-This function check the validation of the customer
7. +rateMenu()-This Function helps the customer to see the Rate menu and the buy the product.
8. +currentBuy()-This function help the customer to by the product.
9. +bill()-The Function generates bill of all the product customer buys.

ER Diagram


Final_amount
Product_ID
Product Name
Customer_id
ID
Name
Category
Product
N
Buys
M
Customer
Phone_no
GST
Bill_id
Date_of_Registration
Price
S_no
Quantity
Quantity
GST
Product_amount
Product_id
Product name

Retail store Management system provides following services:

. A User contains the following activity:

1. Admin Activities

2. Customer Activities

· A customer can do following activities:

1. First select whether you are the Existing customer or the new customer, if you are the new customer then you have to register yourself and get the unique customer ID and if you are the Existing customer you already have the customer ID.

2. The Existing customer inserts the customer ID and sees the different option like to buys the product, view his customer account details, update his account info and view shopping history then View final GST added to the product. Then if he wants, he may purchase the product. Take quantity as input from the user. Update the now available quantity accordingly in the database.

- Admin can do following activities:

1. Manage Customer:

-Add customer

-Delete customer

2. Manage Product:

-Add product

-Edit product (name/category/price/quantity/ GST)

3. Report Generation:

-view report of all customers

-view report of all products

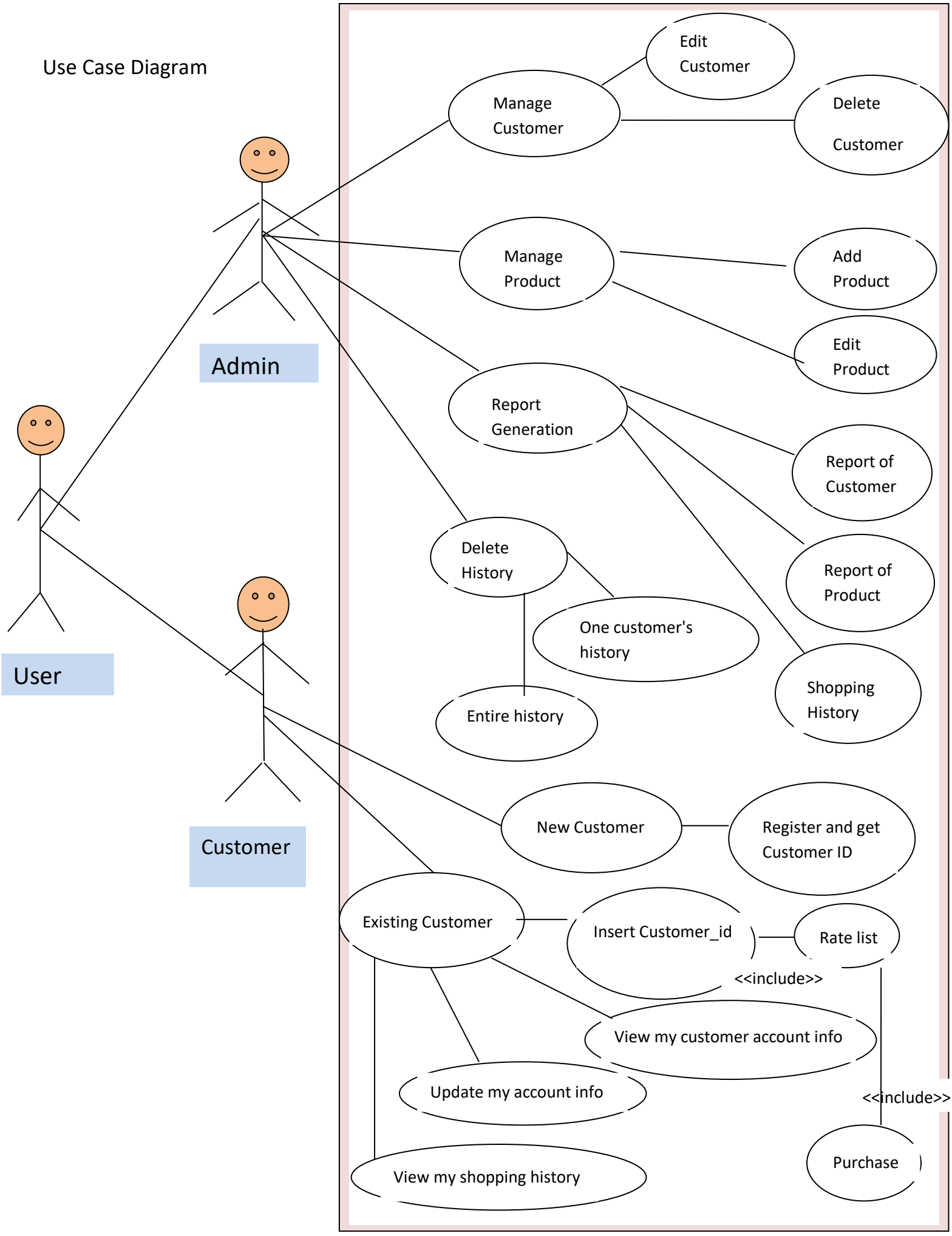
- Report of shopping history

4. Delete History

-One Customer History

-Entire History

Use Case Diagram



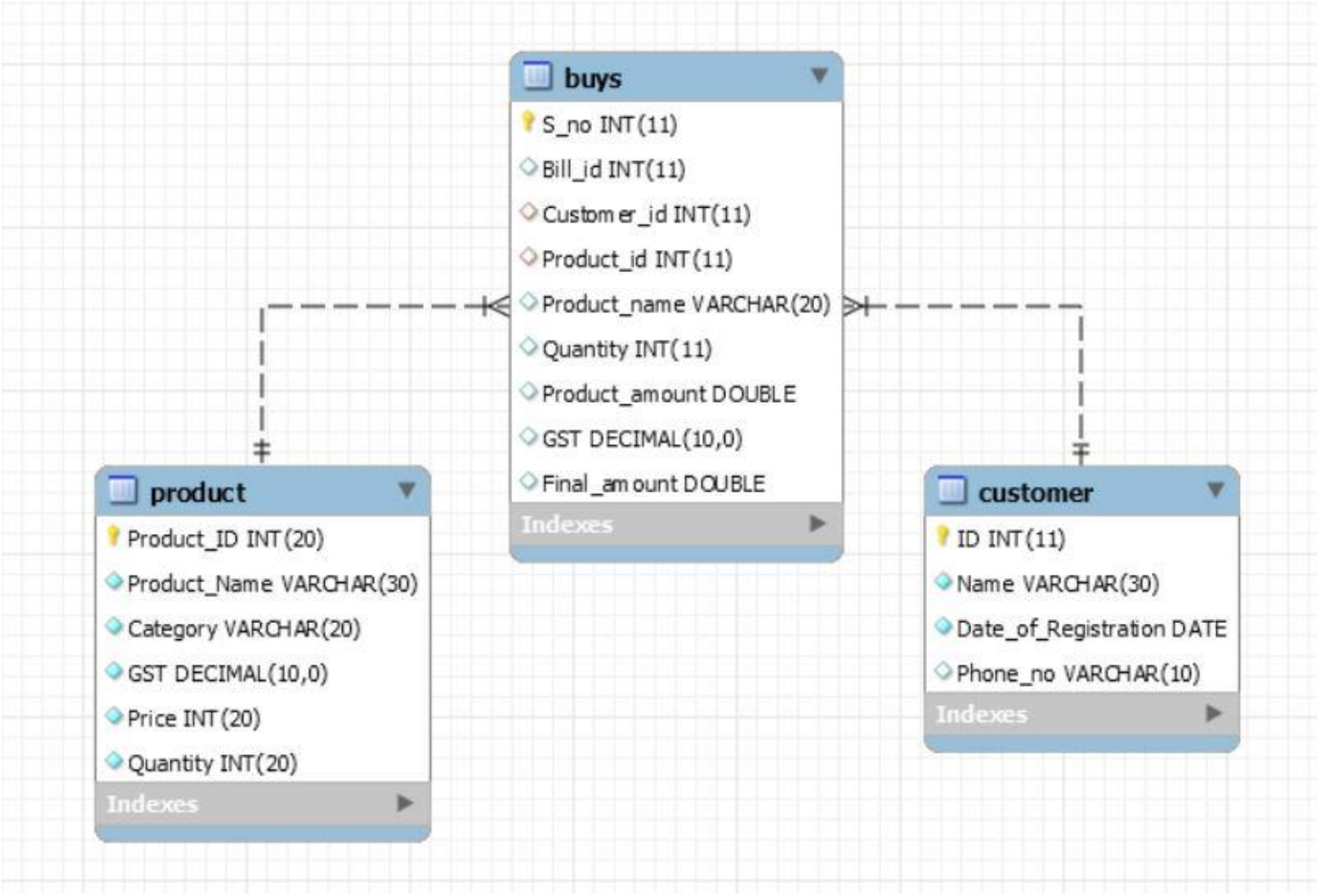
List of all Tables and Schema:

The list of all table used in database is:

1. product

2. customer

3.buys



Code-

User Module-

```
from abc import ABC, abstractmethod

class User(ABC):

    @abstractmethod

    def validate(self):

        pass
```

Admin Module-

```
import mysql.connector

from mysql.connector import Error

import sys

from prettytable import from_db_cursor

from userMod import User

from datetime import date

import time


try:
```

```
  con = mysql.connector.connect(host = 'localhost', db =
'practice', user = 'root', password = 'password')
  cur = con.cursor(buffered=True)
except Error as e:
  print(e)


def cls():
  print("\n" * 1)


class Administration(User):



  def __init__(self):
    print("Welcome Admin!")


  def valid_cid(self, cid):
    try:
      cur.execute("SELECT ID FROM customer")
      flag = 0
      for value in cur.fetchall():
        if cid == value[0]:
          flag = 1
```

```
            return True
        if flag == 0:
            return False
    except Error as e:
        print(e)


def valid_pid(self, pid):
    try:
        cur.execute("SELECT Product_ID FROM product")
        flag = 0
        for value in cur.fetchall():
            if pid == value[0]:
                flag = 1
                return True
                break
        if flag == 0:
            return False
    except Error as e:
        print(e)
```

```
  def delete_cust(self):
    try:
      cid = int(input("Enter customer id:"))
      if self.valid_cid(cid):

        try:
          sql = "DELETE from customer where ID = %s"          # Delete the customer with customer id
          cur.execute(sql,(cid,))
          print("Customer deleted")
          con.commit()
          cls()
          self.manag_cust()

        except Error as e:
          print(e)

      else:
        print("Invalid cid, try again!")
```

```
            self.manag_cust()

        except ValueError as e:
            print(e)
            self.delete_cust()



    def edit_cust(self):

        print('''1.Edit customer name
2.Edit customer's date of registration
3.Edit customer's phone number
4.Return to previous menu''')
        t = int(input("Please enter your choice: "))

        if t == 1:
            try:
                cus_id = int(input("Enter the customer id: "))
                if self.valid_cid(cus_id):
                    new_name = input("Enter new name: ")
                    try:
```

```
                sql = "UPDATE customer SET Name = %s WHERE ID =
%s"       # Edits the customer id
                cur.execute(sql,(new_name, cus_id))
                print("Name updated")
                con.commit()
                cls()
                self.edit_cust()
              except Error as e:
                print(e)
            else:
              print("This customer ID does not exist. Try again!")
              self.edit_cust()
          except ValueError as e:
            print(e)

        elif t == 2:
          try:
            cus_id = int(input("Enter customer id: "))
            if self.valid_cid(cus_id):
              new_date = input("Enter new date(yyyy-mm-dd):")
              newdate1 = time.strptime(new_date, "%Y/%m/%d")
              today = str(date.today().strftime('%Y/%m/%d'))
```

```
            today1 = time.strptime(today, "%Y/%m/%d")
            if newdate1 > today1:
              print("Date of registration exceeds today's date. This
is not possible. Try again!")
              self.edit_cust()
            else:
              try:
                cur.execute("UPDATE customer SET
Date_of_Registration=%s WHERE ID = %s",(new_date, cus_id))
# Edits the customer date of registration
                print("Date updated")
                #con.commit()
                cls()
                self.edit_cust()

              except Error as e:
                print (e)
          else:
            print("This customer id does not exist. Try again!")
            self.edit_cust()

        except ValueError as e:
          print(e)
```

```
            self.edit_cust()


        elif t == 3:
            newPhone = input("Enter new phone number: ")
            if len(newPhone) != 10:
                print("Invalid phone number entered. Try again!")
                cls()
                self.edit_cust()
            else:
                try:
                    cur.execute("update customer SET Phone_no = %s",
(newMobile,))
                    print("Phone number updated")
                    con.commit()
                except Error as e:
                    print(e)
            cls()
            self.edit_cust()




        elif t == 4:
```

```
            cls()
            self.manag_cust()


        else:
            print("Invalid option entered. Try again!")
            cls()
            self.edit_cust()


    def manag_cust(self):
        print('''1.Edit Customer
2.Delete Customer
3.Return to previous menu''')
        e = int(input("Please enter your Choice: "))
        if(e == 1):
            cls()
            self.edit_cust()
            cls()
        elif(e == 2):
            cls()
            self.delete_cust()
            cls()
        elif(e == 3):
```

```
            cls()
            self.adminMenu()



        else:
            print("Invalid option entered. Try again!")
            cls()
            self.manag_cust()


    def add_prod(self):
        try:
            pid = int(input("Enter product id:"))
            pname = input("Enter product name:")
            cat = input("Enter category to which product belongs:")
            cg = int(input("Enter the GST of the product:"))
            price = int(input("Enter price of the product:"))
            quant = int(input("Enter the Quantity of the product:"))
            if cg < 0 or price < 0 or quant < 0:
                print(''' The following cannot be negative:
1. GST
2. Price
3.Quantity
```

```
Please check the values you have entered and try again!''')
            self.add_prod()


        else:
            try:
                sql = "INSERT INTO product values(%s, %s, %s, %s,
%s, %s)"   # Adds the Product
                cur.execute(sql,(pid, pname, cat, cg, price, quant))
                con.commit()
                print("Product added")
                con.commit()
                self.manag_prod()


            except Error as e:
                print(e)
                self.add_prod()
    except ValueError as e:
        print(e)
        self.add_prod()


  def edit_prod(self):
```

```
        print('''1. Edit product name
2. Edit category
3. Edit GST  on a category
4. Edit  price
5. Edit quantity
6. Return to previous menu''')
        m = int(input("Please enter your choice:"))

        if m == 1:
            try:
                id_p = int(input("Enter product id of the product:"))
                if self.valid_pid(id_p):
                    new_pname = input("Enter new name of the
product:")
                    try:
                        cur.execute("UPDATE product SET Product_Name
= %s WHERE Product_ID = %s",(new_pname, id_p))   # Edit
Product name
                        con.commit()
                        print("Product updated")
                        cls()
                        self.edit_prod()
                    except Error as e:
```

```
                print(e)
            else:
                print("Invalid pid, try again")
                self.edit_prod()
        except ValueError as e:
            print(e)
            self.edit_prod()



    elif m == 2:


        try:


            id_p = int(input("Enter product id of the product: "))
            if self.valid_pid(id_p):
                new_cat = input("Enter new category of the
product:")
                new_cg = int(input("Enter new category's
corresponding GST: "))
                try:
                    cur.execute("UPDATE product SET Category = %s
WHERE Product_ID = %s",(new_cat, id_p))  # Edit the category
and also edits its corresponding GST
```

```
                cur.execute("UPDATE product SET GST = %s
WHERE Product_ID = %s",(new_cg, id_p))
                con.commit()
                print("Category updated!")
                cls()
                self.edit_prod()
            except Error as e:
                print(e)
        else:
            print("This product id does not exist. Try again!")
            self.edit_prod()
    except ValueError as e:
        print(e)
        self.edit_prod()



elif m == 3:


    try:
        id_p = int(input("Enter product id of the product:"))
        if self.valid_pid(id_p):
```

```
                new_gst = int(input("Enter GST of the corresponding
Product id:"))
                if new_gst<0:
                    print("GST can't be negative!")
                    self.edit_prod()
                else:
                    try:
                        cur.execute("UPDATE product SET GST=%s
WHERE Product_ID = %s",(new_gst, id_p))  # Edit GST of the
product
                        con.commit()
                        print("GST updated!")
                        cls()
                        self.edit_prod()

                    except Error as e:
                        print(e)
            else:
                print("This product id does not exist. Try again!")
                self.edit_prod()
        except ValueError as e:
            print(e)
            self.edit_prod()
```

```
        elif m == 4:
            try:
                id_p = int(input("Enter product id of the product:"))
                if self.valid_pid(id_p):
                    new_price = int(input("Enter the new price of the
product:"))
                    if new_price<0:
                        print("Price cannot be negative.")
                        self.edit_prod()
                    else:
                        try:
                            cur.execute("UPDATE product SET Price=%s
WHERE Product_ID = %s",(new_price, id_p))  # Edits price
                            con.commit()
                            print("Price updated!")
                            cls()
                            self.edit_prod()

                        except Error as e:
                            print(e)
```

```
            else:
                print("This product id does not exist. Try again!")
                self.edit_prod()
            except ValueError as e:
                print(e)
                self.edit_prod()


        elif m == 5:
            try:
                id_p = int(input("Enter product id of the product:"))
                if self.valid_pid(id_p):
                    new_q = int(input("Enter new quantity of the
product:"))
                    if new_q<0:
                        print("Quantity can't be negative")
                        self.edit_prod()
                    else:
                        try:
                            cur.execute("UPDATE product SET Quantity=%s
WHERE Product_ID = %s",(new_q, id_p))  # Edits Product
quantity
                            con.commit()
                            print("Quantity updated!")
```

```
                cls()
                self.edit_prod()

            except Error as e:
                print(e)
        else:
            print("This product ID does not exist. Try again!")
            self.edit_prod()
    except ValueError as e:
        print(e)
        self.edit_prod()

elif m == 6:
    cls()
    self.manag_prod()

else:
    print("Invalid option entered, please try again!")
    cls()
    self.edit_prod()
```

```
  def manag_prod(self):
      print('''1.Add Product
2.Edit Product
3.Return to previous menu''')
      f= int(input("please Enter your Choice:"))
      if(f == 1):
         cls()
         self.add_prod()
         cls()
      elif(f == 2):
         cls()
         self.edit_prod()
         cls()
      elif(f == 3):
         cls()
         self.adminMenu()
      else:
         print("Invalid option entered, please try again!")
         cls()
         self.manag_prod()
```

```
  def report_gener(self):
      print('''1.View report of all Customer
2.View report of All Products
3.View all customer's shopping history
4.Return to previous menu''')
      h = int(input("Please Enter your Choice:"))

      if(h == 1):
         try:
            cur.execute("SELECT * FROM customer")     # Report of
all Customer
            print(from_db_cursor(cur))
         except Exception as e:
            print(e)
         self.report_gener()

      elif(h == 2):
         try:
            cur.execute("SELECT * FROM product")     # Report of
All Products
            print(from_db_cursor(cur))
```

```
            except Exception as e:
                print(e)
            self.report_gener()

        elif(h == 3):
            try:
                cur.execute("SELECT * FROM buys")       # All customer
shopping history
                print(from_db_cursor(cur))
            except Exception as e:
                print(e)
            self.report_gener()

        elif(h == 4):
            cls()
            self.adminMenu()

        else:
            print("Invalid option entered, try again!")
            cls()
            self.report_gener()
```

```
  def delete_history(self):
      o = int(input('''Do you want to delete:
1. One customer's history
2. Entire history
3. Return to previous menu
Enter your choice: '''))
      if o == 1:
          try:
              cid = int(input("Enter the cid of customer whose record you want to delete: "))
              if self.valid_cid(cid):
                  try:
                      cur.execute("delete from buys where Customer_id = %s",(cid,))
                      print(cur.rowcount," rows deleted")
                      con.commit()
                  except Error as e:
                      print(e)
              else:
                  print("Invalid customer id entered, please try again!")
                  self.delete_history()
```

```
        except ValueError as e:
            print(e)
            cls()
            self.delete_history()



    elif o == 2:
        try:
            cur.execute(("delete from buys"))
            print("Data deleted")
            con.commit()
            cls()
            self.delete_history()

        except Error as e:
            print(e)



    elif o == 3:
        cls()
        self.adminMenu()
```

```
        else:
            print("Invalid option entered, please try again!")
            cls()
            self.delete_history()




    def validate(self):
        password = input("Please enter password:")
        if(password == "admin123"):
            cls()
            self.adminMenu()


        else:
            print('''Incorrect password!
1. Try again
2. Quit''')
            valOption = int(input("Option(1/2): "))
            if valOption == 1:
                        cls()
```

```
                self.validate()
        elif valOption == 2:
                return
        else:
                print("Invalid option, try again!")
                cls()
                self.validate()


  def adminMenu(self):
    print('''1.Manage customer
2.Manage product
3.Report generation
4.Delete history
5.Logout as admin''')
    d = int(input("Please enter your Choice:"))
    if(d == 1):
      cls()
      self.manag_cust()
    elif(d == 2):
      cls()
      self.manag_prod()
```

```
    elif(d == 3):
        cls()
        self.report_gener()
    elif(d == 4):
        cls()
        self.delete_history()
    elif(d == 5):
        return
    else:
        print("Invalid option entered, try again!")
        cls()
        self.adminmenu()
```

Customer Module

```
import mysql.connector
from mysql.connector import Error
from prettytable import from_db_cursor
from userMod import User
from datetime import date
import time
```

```
try:
    con = mysql.connector.connect(host = 'localhost', db =
'practice', user = 'root', password = 'password')
    cur = con.cursor(buffered = True)

except Error as e:
    print(e)

def cls():
    print("\n" * 1)

class Customer(User):

    def __init__(self):
        print("Welcome Customer !")

    def cust_menu(self, cid, bill_id):
        self.bill_id = bill_id
        self.cid = cid
        print('''1. Buy products
2. View my customer account info
```

```
3. Update my account info
4. View my shopping history
5. Exit''')

        custOption = int(input("Choose an option: "))

        if custOption == 1:
            cls()
            self.rateMenu(cid, bill_id)
        elif custOption == 2:
            cls()
            self.view_profile(cid, bill_id)
        elif custOption == 3:
            cls()
            self.update_cust(cid, bill_id)
        elif custOption == 4:
            cls()
            self.cust_history(cid, bill_id)
        elif custOption == 5:
            cls()
            return
        else:
```

```
            print("Invalid input. Try again!")
            cls()
            self.cust_menu(cid, bill_id)

    def view_profile(self, cid, bill_id):
        self.cid = cid
        self.bill_id = bill_id
        try:
            cur.execute("SELECT * from customer where ID = %s",
(cid,))
            res = from_db_cursor(cur)
            print(res)
            con.commit()
        except Error as e:
            print(e)
        cls()
        self.cust_menu(cid, bill_id)

    def update_cust(self, cid, bill_id):
        self.bill_id = bill_id
        self.cid = cid
        print('''1. Want to update name
```

```
2. Want to update phone number
3. Return to previous menu''')

        updateOption = int(input("Enter your option: "))

        if updateOption == 1:
            newName = input("Enter updated name: ")
            try:
                cur.execute("update customer SET Name = %s",
(newName,))
                print("Profile Updated")
                con.commit()
            except Error as e:
                print(e)
            cls()
            self.update_cust(cid, bill_id)

        elif updateOption == 2:
            newMobile = input("Enter new phone number: ")
            if len(newMobile) != 10:
                print("Invalid phone number entered. Try again!")
                cls()
```

```
            self.update_cust(cid, bill_id)
        else:
            try:
                cur.execute("update customer SET Phone_no = %s",
(newMobile,))
                print("Phone number updated")
                con.commit()
            except Error as e:
                print(e)

        cls()
        self.update_cust(cid, bill_id)

    elif updateOption == 3:
        cls()
        self.cust_menu(cid, bill_id)

    else:
        print("Invalid value entered. Try again!")
        cls()
        self.update_cust(cid, bill_id)
```

```
    def cust_history(self, cid, bill_id):
        self.bill_id = bill_id
        self.cid = cid
        try:
            cur.execute("SELECT * from buys where Customer_id =
%s", (cid,))
            res = from_db_cursor(cur)
            print(res)
        except Error as e:
            print(e)
        cls()
        self.cust_menu(cid, bill_id)



    def regNew(self):
        name = input("Enter your name:")
        dor = str(date.today().strftime('%Y/%m/%d'))


        mobile = input("Enter your Phone number (10 digits only):
")
        if (len(mobile) != 10):
            print("Invalid Phone number. Try again!")
```

```
            self.regNew()


        else:
            try:
                sql = ("INSERT INTO Customer values (null, %s, %s,
%s)")  #Adds the customer with customer name, DOR and auto
customer ID
                cur.execute(sql,(name, dor, mobile))
                print("New customer account created")
                cur.execute("SELECT * from customer where Name =
%s", (name,))
                res = from_db_cursor(cur)
                print("Your recorded info is: ")
                print(res)
                print("You are now registered! Glad to have you with
us!")
                print("Kindly remember the customer ID (ID) for future
reference")
                con.commit()
                cls()
            except Error as e:
                print(e)
```

```
def validate(self, cid):
    self.cid = cid
    flag = 0
    try:
        cur.execute("SELECT ID FROM customer")

        for value in cur.fetchall():
            if cid == value[0]:
                print("Logged in as ID:",cid)
                cls()
                flag = 1
                break
        return flag

    except Error as e:
        print(e)
    except ValueError as e:
        print(e)

def rateMenu(self, cid, bill_id):
    self.bill_id = bill_id
    self.cid = cid
```

```
        print("----------------------Product Menu----------------------")
        try:
            cur.execute("SELECT Product_ID, Product_Name, Price,
Quantity, GST FROM Product")
            y = from_db_cursor(cur)
            print(y)
            cls()
        except Error as e:
            print(e)


        buy = input("Want to buy? (y/n): ").lower()
        if buy == 'y':
            if 1!= 0:                                   # do condition of
do while loop implementation
                cls()
                self.currentBuy(cid, bill_id)
                cls()
                while True:
                    buyMore = input("Want to buy more ? (y/n):
").lower()
                    if buyMore == 'y':
                        cls()
                        self.currentBuy(cid, bill_id)
```

```
                continue
            else:
                break
        else:
            pass
    else:
        pass

    if buy == 'y':
        self.bill(cid, bill_id)
    else:
        print("-----------------")
        print("Nothing ordered")
        print("-----------------")
        cls()
        self.cust_menu(cid, bill_id)

def currentBuy(self, cid, bill_id):
    self.bill_id = bill_id
    self.cid = cid
    productBuy = int(input("Enter product ID of the product to
buy: "))
```

```
        quantityBuy = int(input("Enter quantity to buy: "))


        #query1 (check entered product ID matches one of in
product table)
        try:
            cur.execute("SELECT Product_ID FROM Product")
            flag = 0
            for value in cur.fetchall():
                if productBuy == value[0]:
                    cur.execute("SELECT Quantity FROM Product
WHERE Product_ID = %s",(productBuy,))
                    q = cur.fetchone()[0]
                    updated_quantity = q - quantityBuy
                    if updated_quantity < 0:
#query2 (check quantity enetred does not become negative
after dec it from current quantity in product table of respective
product
                        print("Not sufficient stock")
#if quantity becomes negative print "not sufficient stock"
                        print("Refer product menu for available quantity.
Try again")
                        cls()
                        self.currentBuy(cid, bill_id)
```

```
            else:
                cur.execute("UPDATE product set Quantity = %s
where Product_ID = %s",(updated_quantity, productBuy))


                #now fetching all required info one by one to
calculate gst and enter info into buys table
                cur.execute("SELECT Product_Name FROM
Product WHERE Product_ID = %s",(productBuy,))
                name = cur.fetchone()[0]
                cur.execute("SELECT Price FROM Product WHERE
Product_ID = %s",(productBuy,))
                price = cur.fetchone()[0]
                cur.execute("SELECT GST FROM Product WHERE
Product_ID = %s",(productBuy,))
                GST = cur.fetchone()[0]
                product_amt = quantityBuy * price
                total_gst = (GST/100) * product_amt
                final_amt = product_amt + total_gst
                print("Amount for selected item(s): ",final_amt)
#query3 (insert these entered values into buys table)
                cur.execute("INSERT INTO buys VALUES(null, %s,
%s, %s ,%s ,%s ,%s ,%s, %s)",(bill_id, cid, productBuy, name, q,
product_amt, total_gst, final_amt))
                con.commit()
```

```
                flag = 1
                break

        if flag == 0:                           #if print invalid product
name enter again and call currentBuy() again
            print("Invalid Product name! Try again.")
            cls()
            self.currentBuy(cid, bill_id)                   #call
currentBuy() again

    except Error as e:
        print(e)

def bill(self, cid, bill_id):
    self.bill_id = bill_id
    self.cid = cid
    try:
        cur.execute("select Bill_id, Customer_id, Product_id,
Product_name, Quantity, Product_amount, GST, Final_amount
from buys group by Product_id  having Bill_id = %s order by
Quantity",(bill_id,))#query4 (fetch products ordered with
quantity, group by pid)
```

```
            x = from_db_cursor(cur)
            print(x)
            cls()
            cur.execute("SELECT sum(Final_amount) FROM buys")
#query5 (fetch final amount with gst (sum of all rows))
            print("---------------------------------")
            print("Total bill amount:",cur.fetchone()[0])
            print("Thank you for shopping with us !!")
            print("---------------------------------")

        except Error as e:
            print(e)

        self.cust_menu(cid, bill_id)
```

Menu Module-

```
import custMod
import adminMod
import sys

def cls():
```

```
    print("\n" * 1)


bill_id = 0


def userMenu():
    print('========= Welcome to SAP Retail Store =========')
    print()



    userOption =int(input('''Please select your option:
    1. Admin
    2. Customer
    3. Exit

option(1/2/3): '''))
    if userOption == 1:
        admin = adminMod.Administration()           #admin object
created
        admin.validate()
        cls()
        userMenu()
```

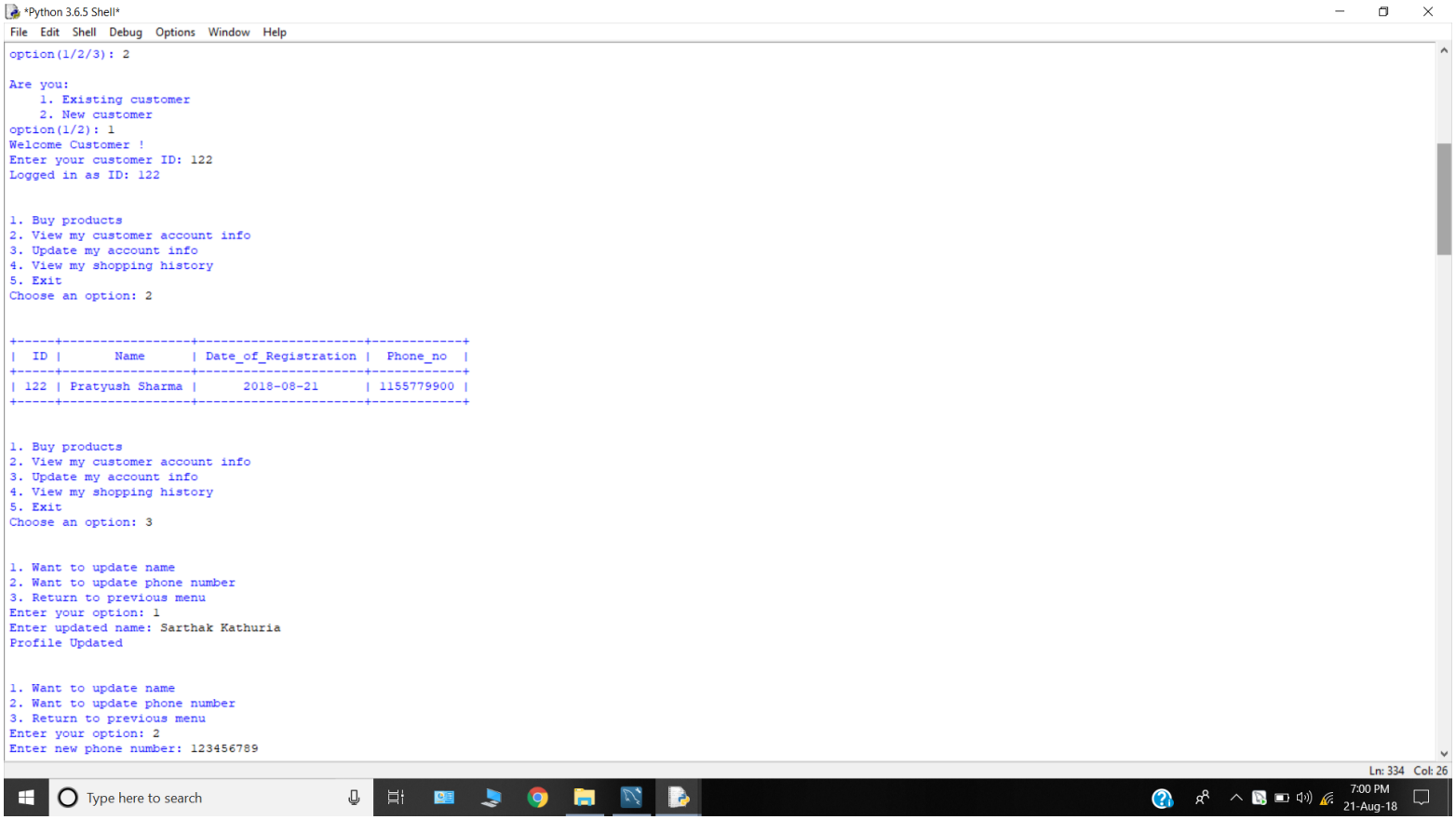
```
    elif userOption == 2:
        print()
        print('''Are you:
    1. Existing customer
    2. New customer''')



        custOption = int(input("option(1/2): "))
        if (custOption == 1):
            global bill_id
            bill_id = bill_id + 1
            customer = custMod.Customer()     #customer object
created
            cid = int(input("Enter your customer ID: "))
            flag_value = customer.validate(cid)
            if flag_value == 1:
                customer.cust_menu(cid, bill_id)
                cls()
                userMenu()
            else:
                print("Invalid ID entered. Try again!")
```

```
            userMenu()

        elif (custOption == 2):
            customer = custMod.Customer()
            customer.regNew()
            cls()
            userMenu()
        else:
            print("Invalid option entered. Try again!")
            cls()
            userMenu()



    elif userOption == 3:
        sys.exit("Exiting program")


    else:
        print("Invalid option entered, Try again!")
        cls()
        userMenu()


userMenu()
```

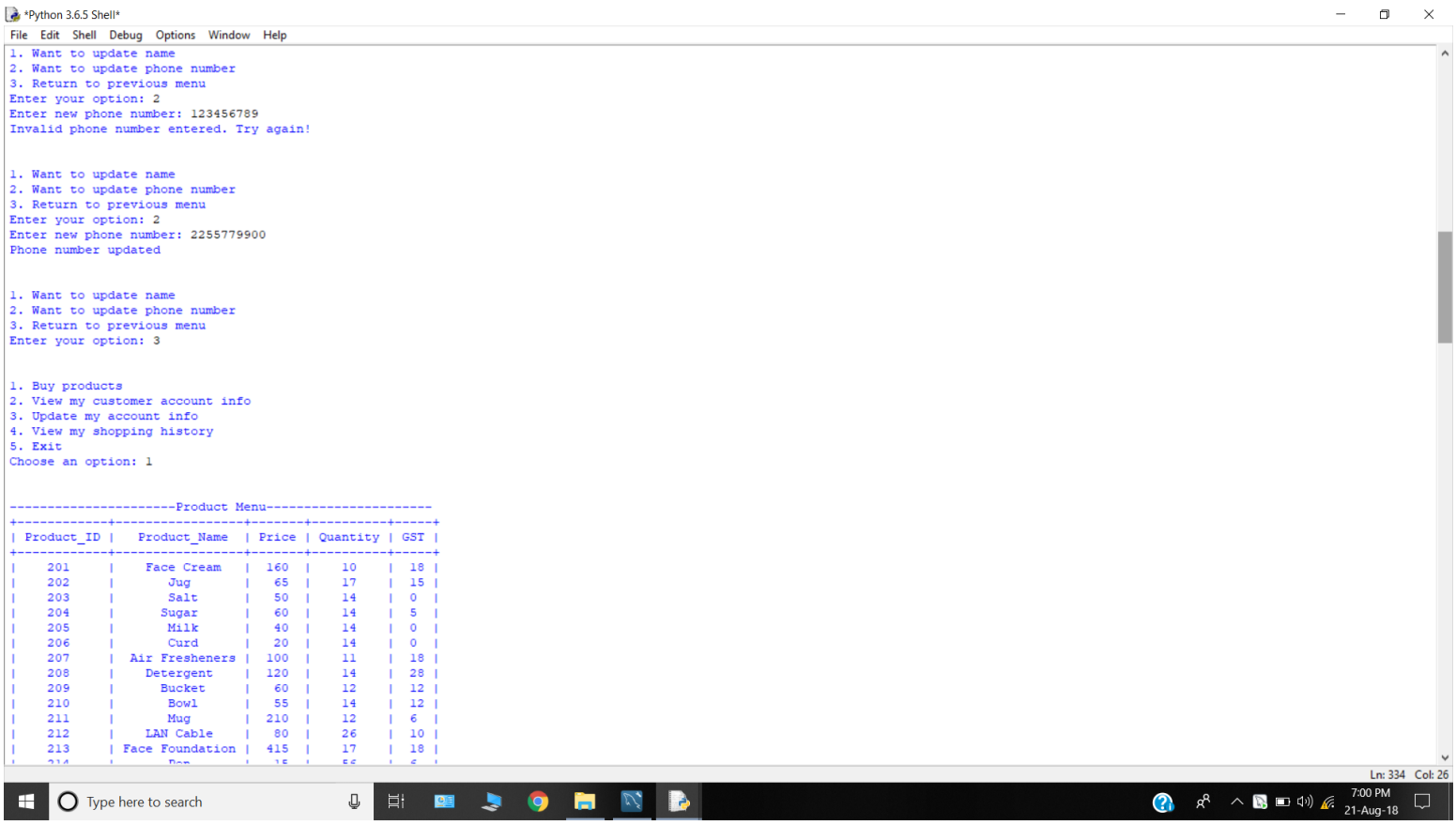
Project Workin with Screenshots-

Customer Side-

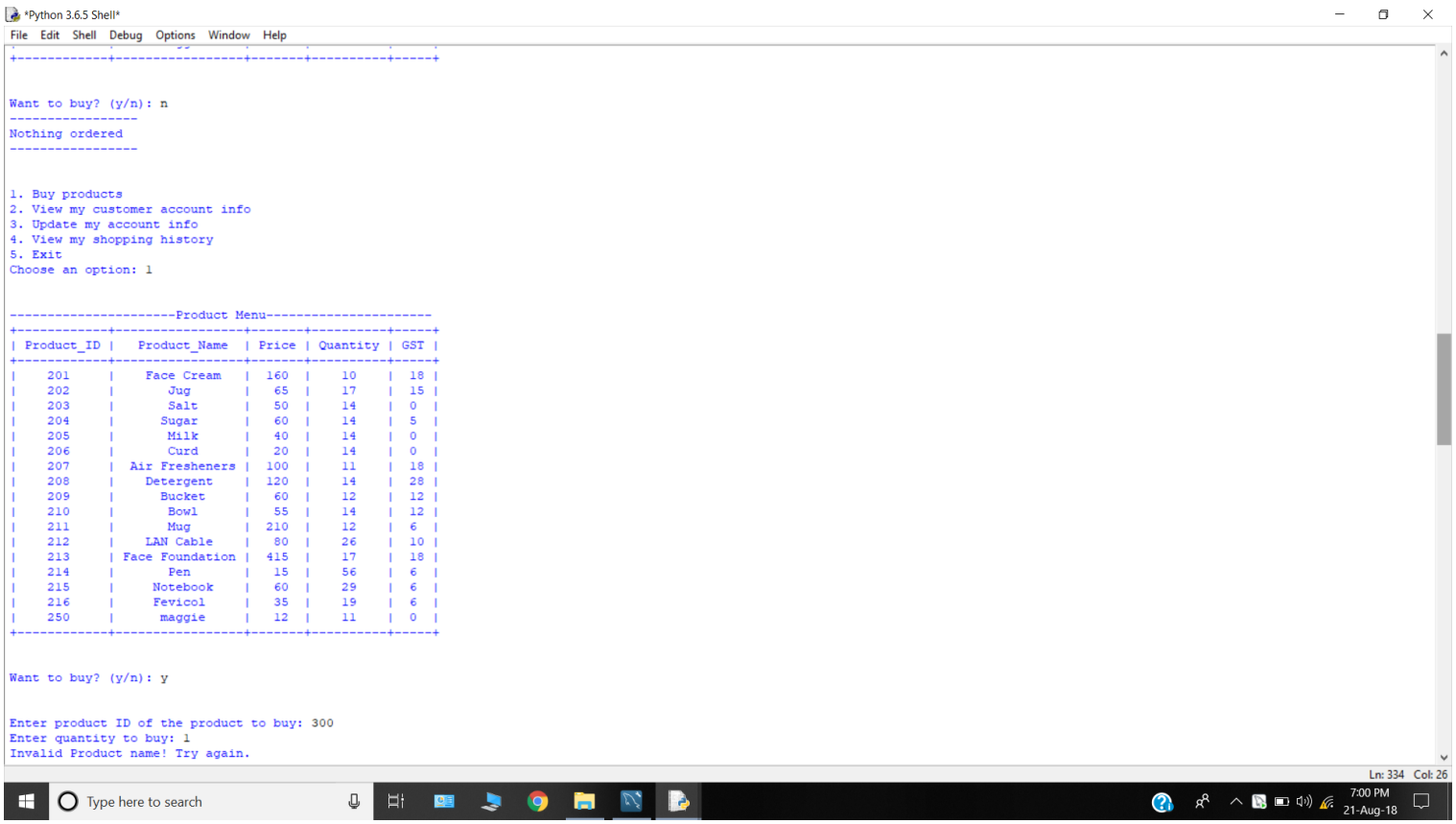
At the start there are option Admin and customer. Here we choose The Customer Side. And add the new Customer with the following Information. And get the unique Customer id.



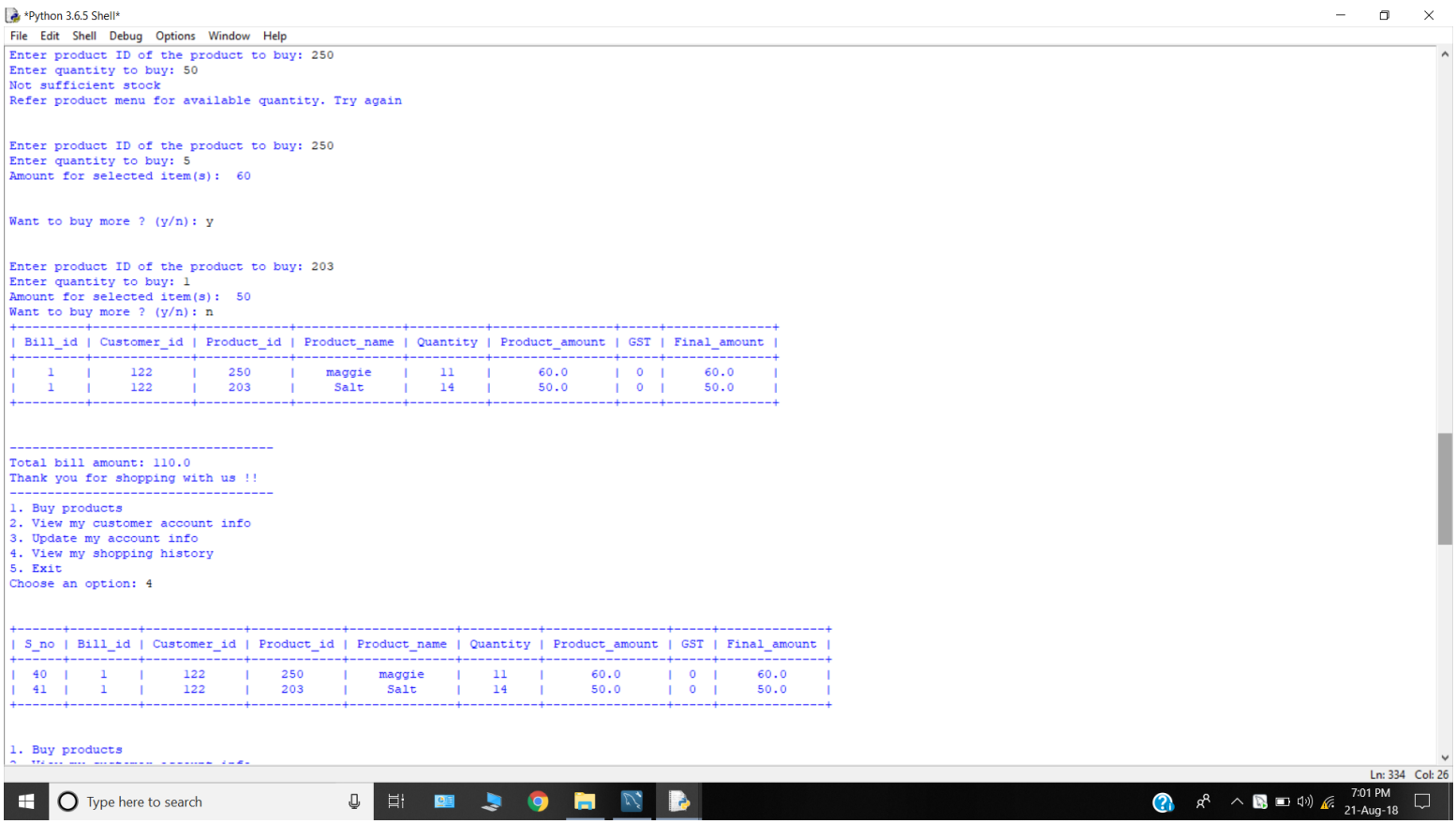
Now the existing customer get its information by this option .



Here the customer update its phone number and wants to buy some product.



It Customer Want to Buy then Just enter “y” and enter the product id of the product from the list.



After Buying from the Store you will generate the Bill.

Python 3.6.5 Shell

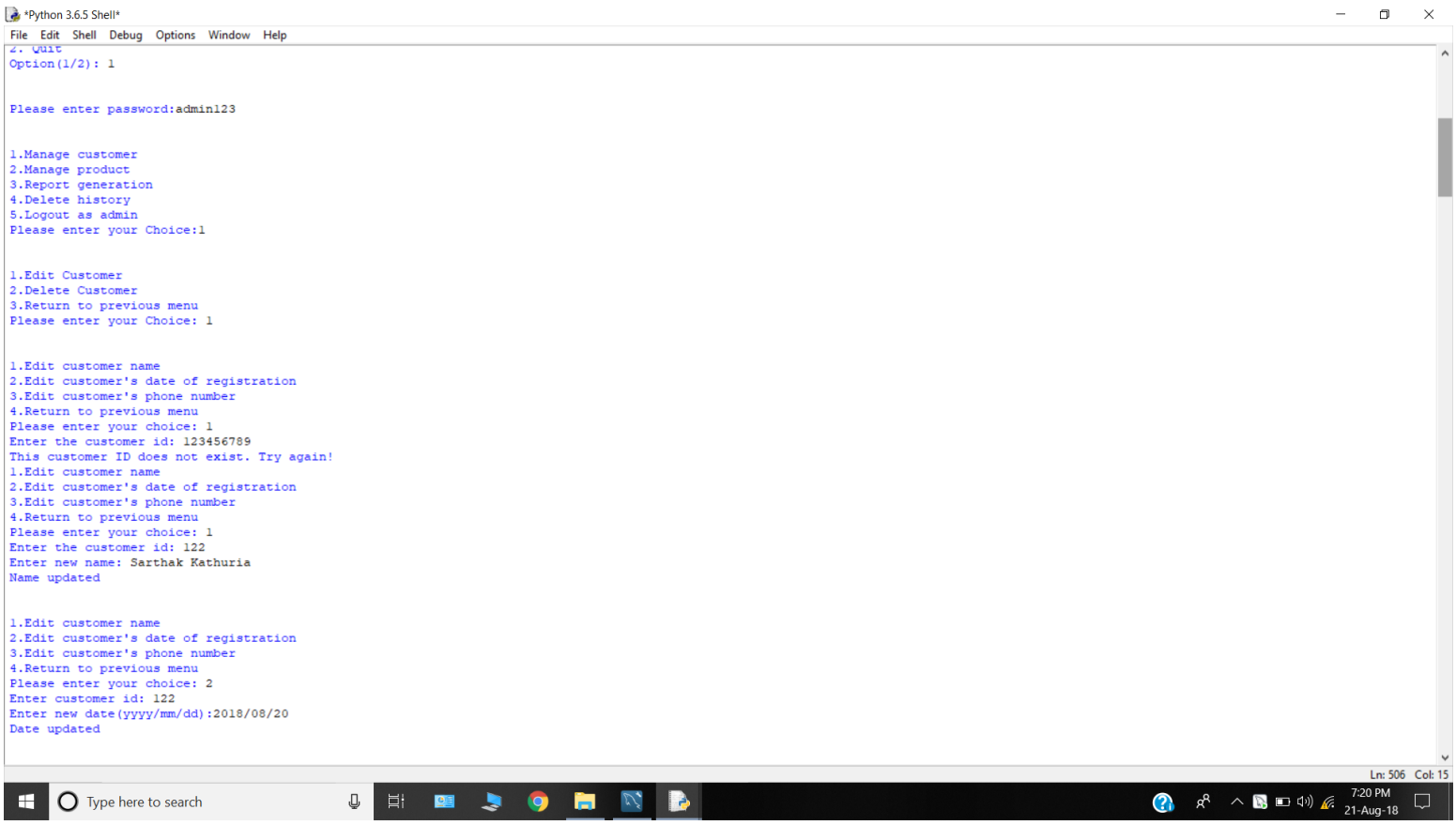
File Edit Shell Debug Options Window Help

```
Enter product ID of the product to buy: 250
Enter quantity to buy: 5
Amount for selected item(s):  60


Want to buy more ? (y/n): y


Enter product ID of the product to buy: 203
Enter quantity to buy: 1
Amount for selected item(s):  50
Want to buy more ? (y/n): n
+---------+------------+------------+--------------+----------+----------------+-----+--------------+
| Bill_id | Customer_id | Product_id | Product_name | Quantity | Product_amount | GST | Final_amount |
+---------+------------+------------+--------------+----------+----------------+-----+--------------+
|    1    |     122    |    250     |    maggie    |    11    |      60.0      |  0  |     60.0     |
|    1    |     122    |    203     |     Salt     |    14    |      50.0      |  0  |     50.0     |
+---------+------------+------------+--------------+----------+----------------+-----+--------------+


----------------------------------
Total bill amount: 110.0
Thank you for shopping with us !!
----------------------------------
1. Buy products
2. View my customer account info
3. Update my account info
4. View my shopping history
5. Exit
Choose an option: 4


+------+---------+------------+------------+--------------+----------+----------------+-----+--------------+
| S_no | Bill_id | Customer_id | Product_id | Product_name | Quantity | Product_amount | GST | Final_amount |
+------+---------+------------+------------+--------------+----------+----------------+-----+--------------+
|  40  |    1    |     122    |    250     |    maggie    |    11    |      60.0      |  0  |     60.0     |
|  41  |    1    |     122    |    203     |     Salt     |    14    |      50.0      |  0  |     50.0     |
+------+---------+------------+------------+--------------+----------+----------------+-----+--------------+


1. Buy products
2. View my customer account info
3. Update my account info
4. View my shopping history
5. Exit
Choose an option: 5
```

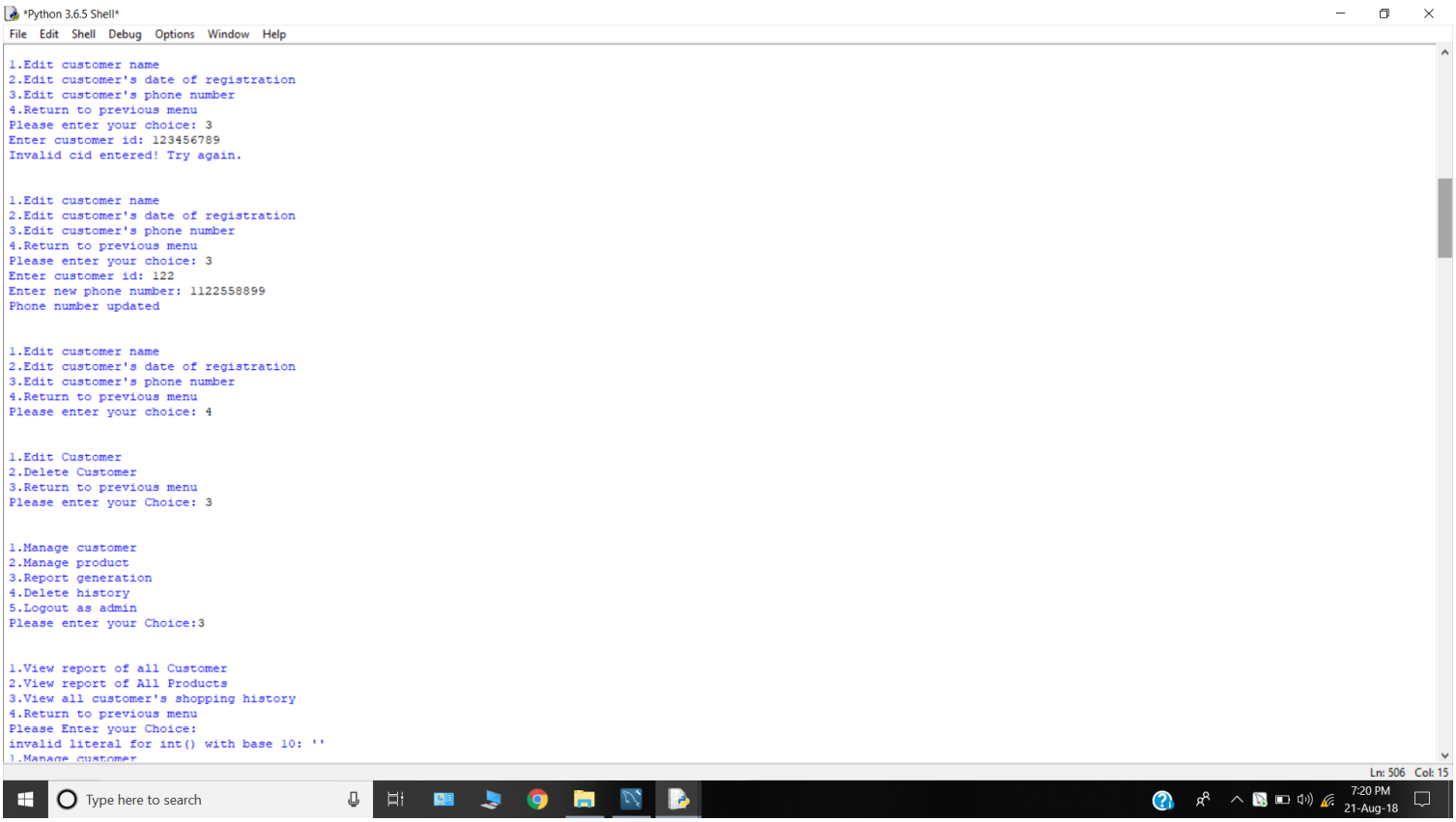
Ln: 334 Col: 26

Admin Side

Now here we Enter in the Admin Menu and enter the Admin password. And enter in the Report Generation of Customer ,Product and the Shopping History.



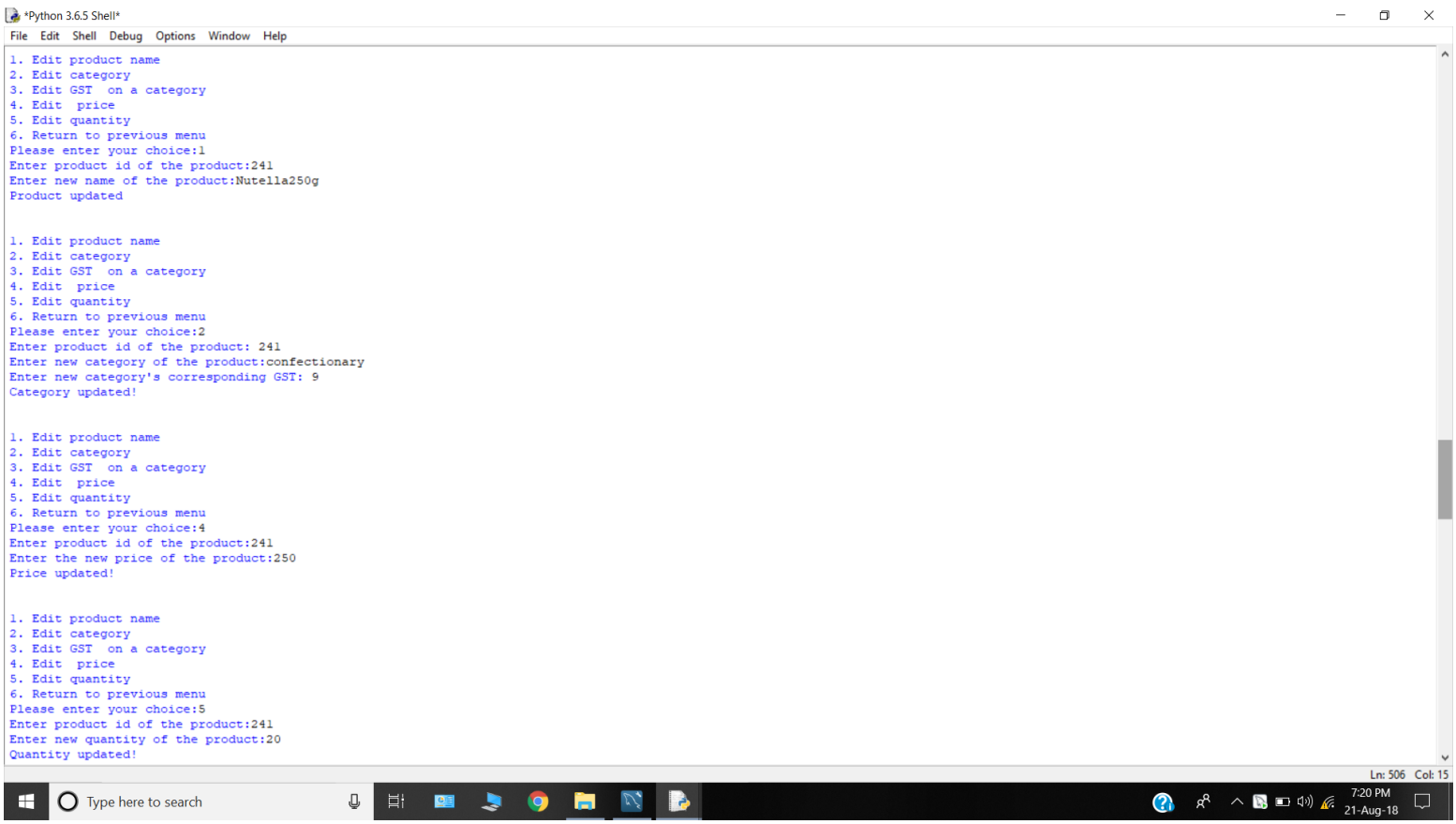
Admin manages the customer information like name ,date of registration and phone number.



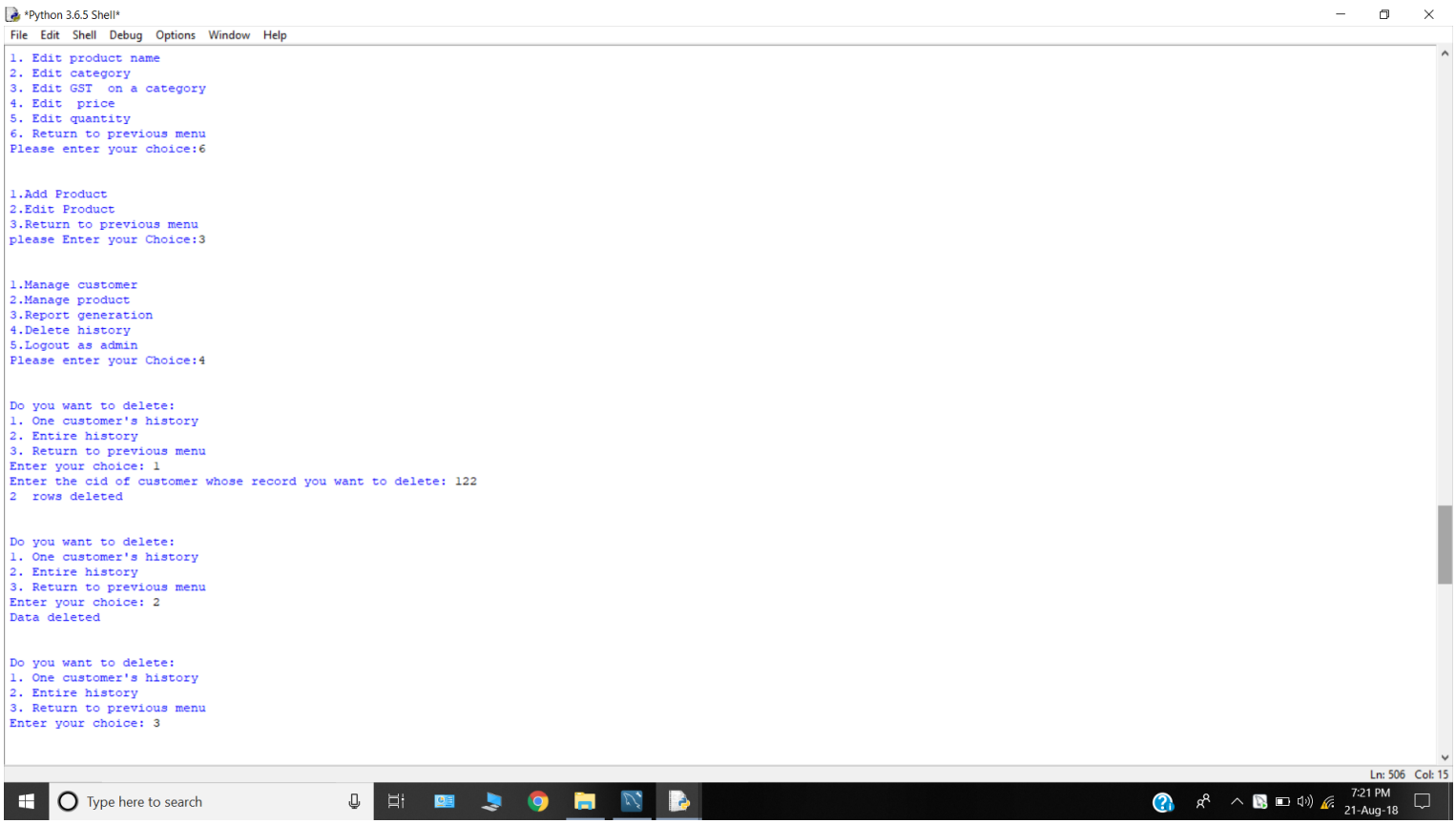
Admin can generates the report of customer, product and the shopping history.

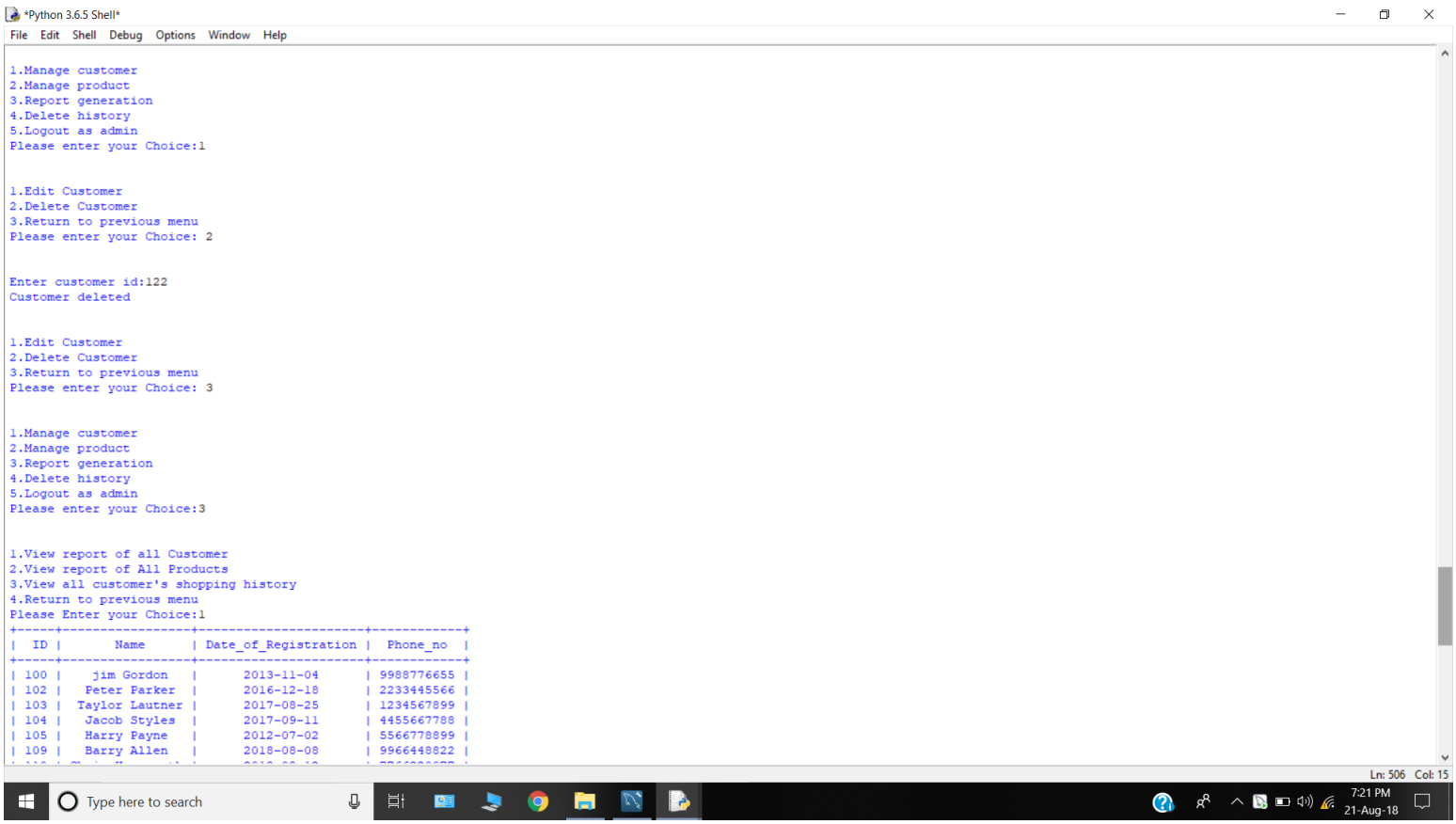
The Customer Report and the Product History.

Admin can manages the product of the retail store.

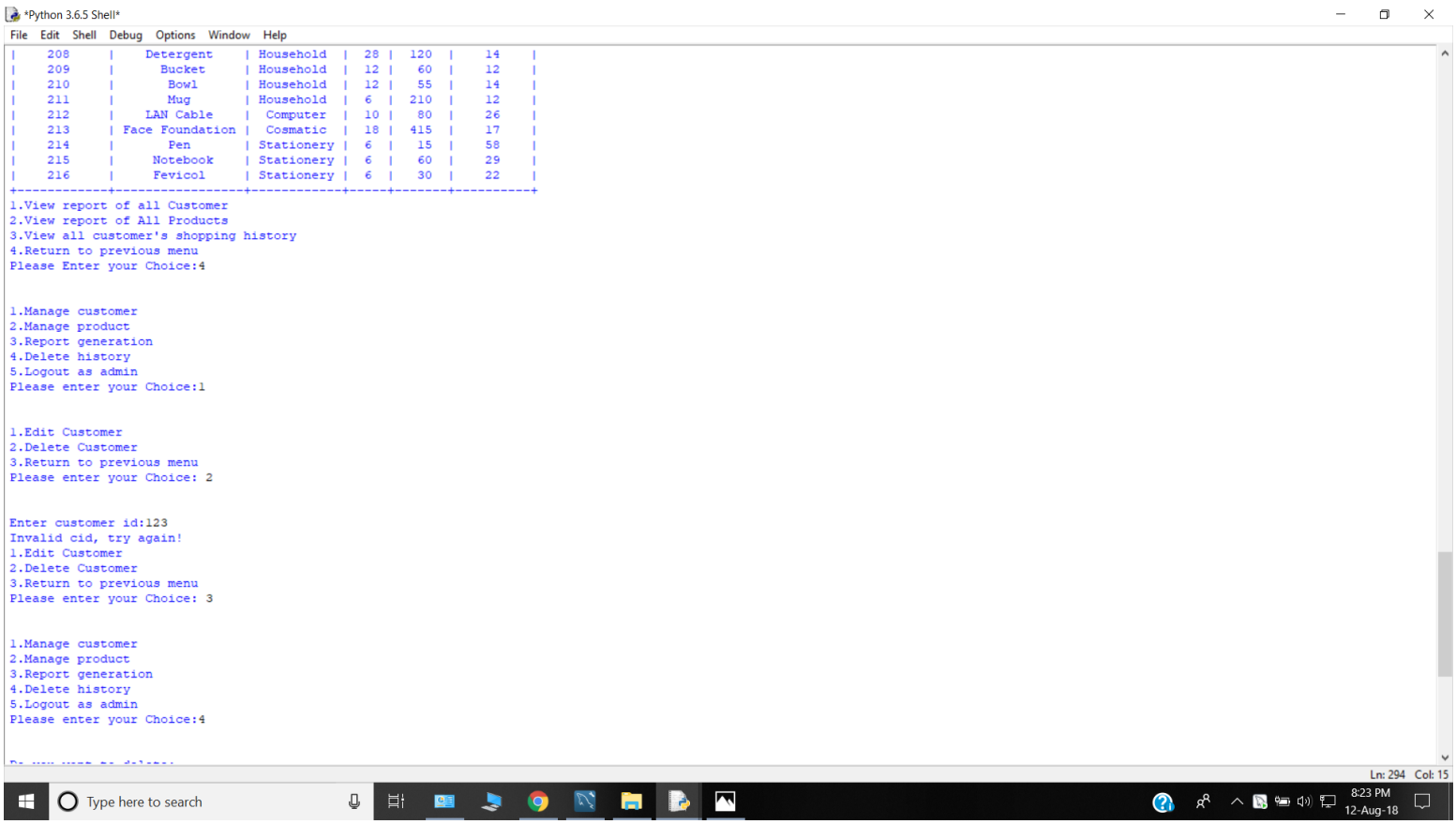


Admin updated the product information like name, category, gst, price and quantity.

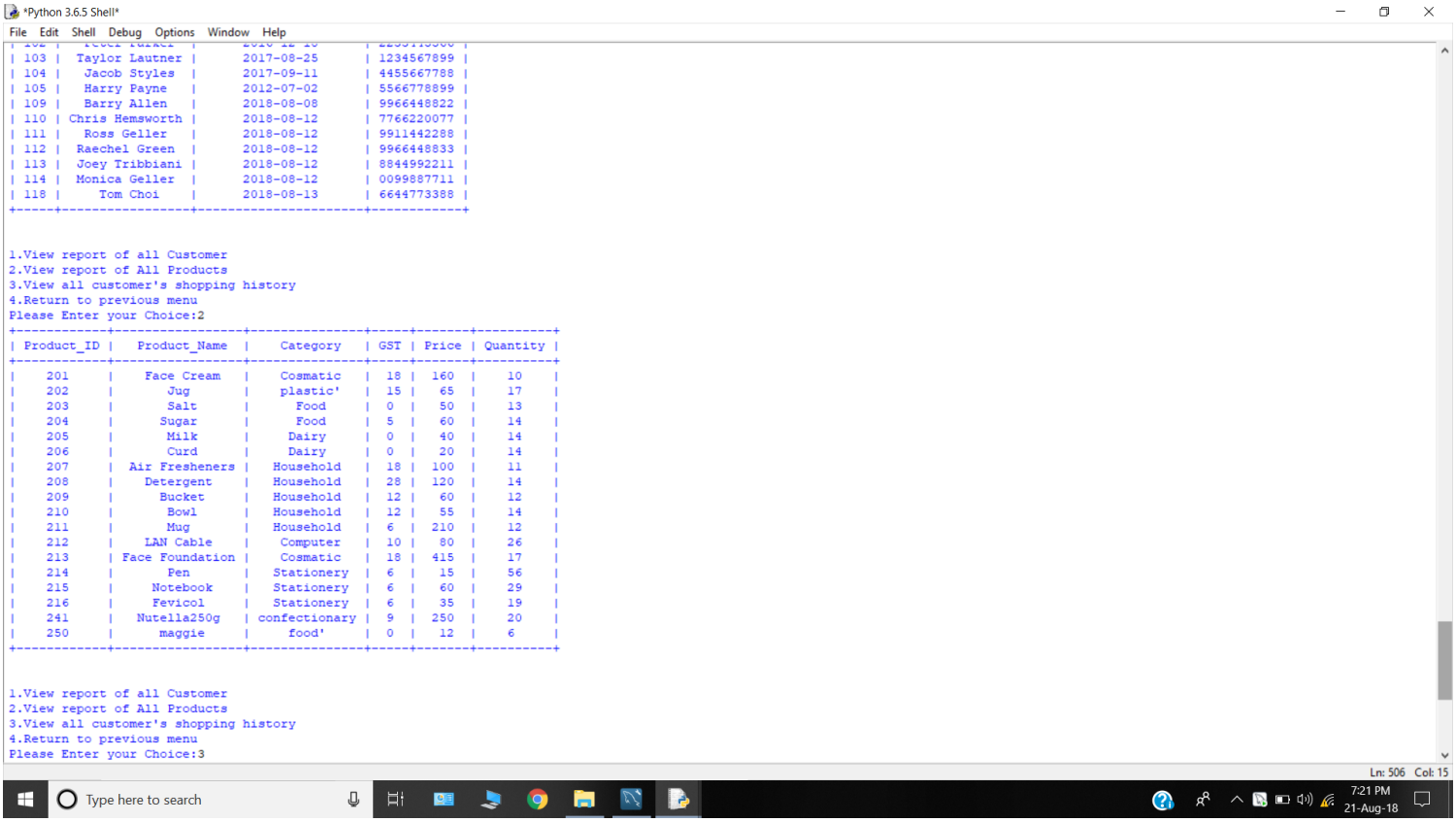

Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
1. Edit product name
2. Edit category
3. Edit GST on a category
4. Edit price
5. Edit quantity
6. Return to previous menu
Please enter your choice:6
1.Add Product
2.Edit Product
3.Return to previous menu
please Enter your Choice:3
1.Manage customer
2.Manage product
3.Report generation
4.Delete history
5.Logout as admin
Please enter your Choice:4
Do you want to delete:
1. One customer's history
2. Entire history
3. Return to previous menu
Enter your choice: 1
Enter the cid of customer whose record you want to delete: 122
2 rows deleted
Do you want to delete:
1. One customer's history
2. Entire history
3. Return to previous menu
Enter your choice: 2
Data deleted
Do you want to delete:
1. One customer's history
2. Entire history
3. Return to previous menu
Enter your choice: 3
Ln: 506 Col: 15
Type here to search
7:21 PM
21-Aug-18



Admin generate the report of customer, product and shopping history.



Admin Delete the History of one customer or Entire history of the Customer.


Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
103	Taylor Lautner	2017-08-25	1234567899
104	Jacob Styles	2017-09-11	4455667788
105	Harry Payne	2012-07-02	5566778899
109	Barry Allen	2018-08-08	9966448822
110	Chris Hemsworth	2018-08-12	7766220077
111	Ross Geller	2018-08-12	9911442288
112	Raechel Green	2018-08-12	9966448833
113	Joey Tribbiani	2018-08-12	8844992211
114	Monica Geller	2018-08-12	0099887711
118	Tom Choi	2018-08-13	6644773388
1.View report of all Customer
2.View report of All Products
3.View all customer's shopping history
4.Return to previous menu
Please Enter your Choice:2
Product_ID	Product_Name	Category	GST	Price	Quantity
201	Face Cream	Cosmatic	18	160	10
202	Jug	plastic'	15	65	17
203	Salt	Food	0	50	13
204	Sugar	Food	5	60	14
205	Milk	Dairy	0	40	14
206	Curd	Dairy	0	20	14
207	Air Fresheners	Household	18	100	11
208	Detergent	Household	28	120	14
209	Bucket	Household	12	60	12
210	Bowl	Household	12	55	14
211	Mug	Household	6	210	12
212	LAN Cable	Computer	10	80	26
213	Face Foundation	Cosmatic	18	415	17
214	Pen	Stationery	6	15	56
215	Notebook	Stationery	6	60	29
216	Fevicol	Stationery	6	35	19
241	Nutella250g	confectionary	9	250	20
250	maggie	food'	0	12	6
1.View report of all Customer
2.View report of All Products
3.View all customer's shopping history
4.Return to previous menu
Please Enter your Choice:3
Ln: 506 Col: 15
Type here to search
7:21 PM
21-Aug-18


Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
215	Notebook	Stationery	6	60	29
216	Fevicol	Stationery	6	35	19
241	Nutella250g	confectionary	9	250	20
250	maggie	food'	0	12	6
1.View report of all Customer
2.View report of All Products
3.View all customer's shopping history
4.Return to previous menu
Please Enter your Choice:3
S_no	Bill_id	Customer_id	Product_id	Product_name	Quantity	Product_amount	GST	Final_amount
1.View report of all Customer
2.View report of All Products
3.View all customer's shopping history
4.Return to previous menu
Please Enter your Choice:4
1.Manage customer
2.Manage product
3.Report generation
4.Delete history
5.Logout as admin
Please enter your Choice:5
========= Welcome to SAP Retail Store =========
Please select your option:
1. Admin
2. Customer
3. Exit
option(1/2/3):
Ln: 506 Col: 15
Type here to search
7:21 PM
21-Aug-18

Limitations

- If program is re-run bill_id gets reset and again starts from “1” leading to its inconsistency.
- No GUI Feature was implemented and the project is text based only.

Conclusion-

Instead of using contentious method of paper-based records and documentation, electronic methods are prepared for Product Ordering, customer management and GST calculation. This is the time of growing insisting of online Retail Store. Every small business now requires a chosen management system with software that is accomplished of handling all the business ends.