Amazon't

A Computer Science Project

Woitiwe Simson Abhinav Anil George Kurian

St. Paul's International School Kalamassery

All India Senior School Certificate Examination

Record of Project Work in Computer Science

This is to certify that of class XII has successfully completed his ***Computer Science*** *project work entitled '****Online Shopping Platform****' as prescribed by the Central Board of Secondary Education during the year 2023-2024.*

Register number : _____

Date : _____

Signature of the Internal Examiner : _____

Signature of the External Examiner : _____

Signature of the Principal with school seal : _____

Index

Acknowledgement

I wish to express my deep gratitude to the Principal, Mrs. Sunitha Binu Samuel and St. Paul's International School for the encouragement and the facilities provided for this project work.

I extend my sincere thanks to Ms. Nithya Soman, Computer Teacher, who guided me to the success of this project work on the topic. I take this opportunity to express my deep sense of gratitude for her valuable guidance, constant encouragement and immense motivation which has sustained my effort in all stages of this project.

I take this opportunity to express my sincere thanks to my parents and also my classmates who helped me carry out this project successfully and for the valuable advice and support, I received from time to time to carry out the project.

Introduction

Computer programming has become a necessity in recent times due to the various purposes it can be used for in day-to-day needs. There are many programming languages for these purposes

In this project, Python and MySQL are primarily used. Both are simple, open source, and easily available. Here we connect both softwares using the 'mysql.connector' package which is also open source and easily downloadable

We write the program's source code, which is written in Python and MySQL queries, in the Python interface. Since both are connected, we only have to create the database using the MySQL Command Client. All the other queries (eg: creating tables, inserting values etc) are done within the python interface by mentioning the necessary queries in required areas

In this project, we have used the latest version of both softwares. This project emphasizes on the fact that we are inputting the data through Python and storing it in the MySQL database

System Requirements

Hardware

- Processor
- Keyboard
- Minimum memory - 2 GB

Software

- Operating system: Windows 10
- Python IDLE (3.10.2, 64 bit)
- MySQL (5.5)

Feasibility Study

Feasibility study is a system proposal according to its work, ability, impact on the operations; ability to meet the needs of users, and efficient use of resources. It is an important outcome of preliminary investigations.

Economic feasibility

Economic analysis is the most frequently used method for evaluating the effectiveness of the program, the benefits and savings that are expected from the system, and comparing them with cost.

This program code is not very expensive. So the user records can be maintained at a cheaper rate.

Technical feasibility

Technical feasibility centers on existing computers and to what extent it can support the proposed task. This involves financial consideration to accommodate technical enhancement.

This is technically feasible because whatever technology is needed to develop the software is easily available

Investigative Analysis

Amazon't:- It is an online store that offers the widest collection of items where users can purchase as much as they with no limitations

The entire program consists of four files, the first of which is used to display UI for the user. The second file will handle general functions (checking whether user is admin or not etc) and will redirect to the other two files as necessary

Upon running the given code, we are asked to log in where one of two things will happen:

1. **User mode**: It gives the perspective of a user who is casually shopping on the site
2. **Admin mode**: We are an 'employee' of Amazon't and deal with adding items for sale, confirming transactions and deliveries etc.

Upon logging in as an admin, the following options are available:

1. **Alter items**: to add/remove items available for sale
2. **Check customers**: To verify customer details and to remove customers who have logged out and are now inactive
3. **Check transactions**: to ensure a transaction has been completed before it gets sent out for delivery
4. **Update delivery statuses**: Once the delivery has been completed, the admin sets the status to 'True' which indicates that the order has been completed

Errors and its Types

An error, also called a bug, is some exception in the code that prevents the program from compiling or running as intended. They are broadly classified into three types:

❖ Compile time errors

An error that occurs during compilation of the program is called a compile time error. It is of two types as follows:

a. Syntax Error: it refers to formal rules governing the construction of valid statements in a language
b. Semantic Error: It refers to the set of rules which give the meaning of a statement

❖ Runtime errors

Errors that occur during the execution of the program are called runtime errors. These are harder to detect. Some runtime errors stop the execution of the program which would then be called “Program Crashed”

❖ Logical errors

Sometimes, no errors are encountered during compiling and runtime, but the program does not give the expected results. This is because of the programmer’s mistaken analysis of the problem he/she is trying to solve. Such errors are called logical errors

SQL Tables

Database name: project

Tables used:

```
mysql> use project;
Database changed
mysql> show tables;
+-------------------+
| Tables_in_project |
+-------------------+
| cart              |
| customers         |
| deliveries        |
| items             |
| transactions      |
+-------------------+
5 rows in set (0.33 sec)
```

Table Structures

```
desc cart;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| C_ID  | int(11)     | YES  |     | NULL    |       |
| I_ID  | int(11)     | YES  |     | NULL    |       |
| Name  | varchar(45) | YES  |     | NULL    |       |
| Price | int(11)     | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
4 rows in set (0.64 sec)
```

```
mysql> desc customers;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| C_ID     | int(11)     | NO   | PRI | NULL    |       |
| NAME     | varchar(25) | YES  |     | NULL    |       |
| PHONE_NO | int(11)     | YES  |     | NULL    |       |
| EMAIL    | varchar(55) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.09 sec)
```

```
mysql> desc deliveries;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| T_ID        | int(11)     | NO   | PRI | NULL    |       |
| DESTINATION | varchar(75) | YES  |     | NULL    |       |
| STATUS      | tinyint(1)  | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
3 rows in set (0.09 sec)
```

```
mysql> desc items;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| ITEM_ID   | int(11)     | NO   | PRI | NULL    |       |
| ITEM_NAME | varchar(45) | YES  |     | NULL    |       |
| PRICE     | int(11)     | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
3 rows in set (0.11 sec)
```

```
mysql> desc transactions;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| T_ID            | int(11)     | NO   | PRI | NULL    |       |
| C_ID            | int(11)     | YES  |     | NULL    |       |
| MODE_OF_PAYMENT | varchar(25) | YES  |     | NULL    |       |
| status          | tinyint(1)  | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
4 rows in set (0.06 sec)
```

Tables with sample records

```
mysql> select * from cart;
Empty set (0.00 sec)
```

```
mysql> select * from customers;
+------+------------+------------+----------------------+
| C_ID | NAME       | PHONE_NO   | EMAIL                |
+------+------------+------------+----------------------+
|  201 | S Vasan    | 1234567890 | irockz@gmail.com     |
|  202 | P.P Ramani |  987654321 | woman@gmail.com      |
|  203 | Pedro      |  918273645 | humanbeing@gmail.com |
+------+------------+------------+----------------------+
3 rows in set (0.00 sec)
```

```
mysql> select * from deliveries;
+------+------------------------+--------+
| T_ID | DESTINATION            | STATUS |
+------+------------------------+--------+
|  554 | 553 Trenton St.        |      1 |
|  584 | 56 East Windsor Street |      0 |
|  598 | 94 St Margarets Dr.    |      1 |
+------+------------------------+--------+
3 rows in set (0.00 sec)
```

```
mysql> select * from items;
+---------+-----------+-------+
| ITEM_ID | ITEM_NAME | PRICE |
+---------+-----------+-------+
|     101 | Dum       |   120 |
|     102 | Dum1      |   135 |
|     103 | Dum2      |   175 |
+---------+-----------+-------+
3 rows in set (0.00 sec)
```

```
mysql> select * from transactions;
+------+------+-----------------+--------+
| T_ID | C_ID | MODE_OF_PAYMENT | status |
+------+------+-----------------+--------+
|  554 |  201 | Credit          |      1 |
|  584 |  201 | cash            |      0 |
|  598 |  102 | Cash            |      1 |
+------+------+-----------------+--------+
3 rows in set (0.02 sec)
```

Source Code

File 1: “Main.py”

```
import mysql.connector as mys
import Functions as f
from random import randint

db = mys.connect(host="localhost", user='root', passwd="root",
database="project")
c = db.cursor()

class Login:
    def userLogin():
        global ciD
        ciD = randint(200, 299)

        name = input("Enter name: ")
        email = input("Enter email address: ")

        while True:
            phno = int(input("Enter phone number: "))
            if len(str(phno)) > 10:
                print("Invalid phone number. Try again")
                continue
            break

        c.execute(f"insert into customers values
({ciD},'{name}',{phno},'{email}')")
        db.commit()

        print(f"Your Customer ID is {ciD}")
```

```
    def check(C_ID):
        c.execute("select * from customers")
        table = c.fetchall()

        for i in table:
            if C_ID in i:
                print(f"Hiya {i[1]}!")
                return True
        else:
            return False

    def login():
        print("(Login)")
        print("DEVNOTES: type 1 for admin mode")
        cID = input("Enter customer ID (leave blank if new user): ")
        if cID == '1': #admin
            return 1
        elif cID == '': # new user
            Login.userLogin()
        else: # existing user
            return cID



'''if f.connection():
    print("*"*15,"CONNECTED","*"*15)'''

user = Login.login() # 0 = user | 1 = admin
ciD = user

if user == 1: # admin
    print("Hello Admin!")
    while True:
        print("-$"*15,"ADMIN FUNCTIONS","$-"*15)
        print("1. View/Alter items \n2. Customers \n3. Transactions \n4. Delivery
status \n0. Quit")

        func = int(input("Enter function: "))
```

```
        f.adminFunctions(func)

else: # user
    if user == 2:
        print("Welcome user! \nPlease Login below")
        Login.userLogin()
    else:
        ciD = int(input("Enter it again: "))

        if Login.check(ciD):
            pass
        else:
            print("Invalid customer ID \nClosing program...")
            quit()

    print("-+"*10,"WELCOME TO AMAZON'T","+-"*10,"\n\n")

    while True:
        print("&="*15,"USER","=&"*15)
        print("1. View items \n2. View Cart \n3. Delivery status \n0. Quit")

        func = int(input("Enter function: "))
        f.userFunctions(func)
```

File 2: “Functions.py”

```
import mysql.connector as mys
import adminFunctions as af
import userFunctions as uf


def connection() -> bool:
    try:
        db = mys.connect(host="localhost", user='root', passwd="root",
database="project")
```

```
            c = db.cursor()
            return True
        except:
            return False

    def userFunctions(funcNum):
        uf.Main.start()
        if funcNum == 0:
            quit()
        elif funcNum == 1:
            while True:
                print("-="*10,"ITEMS(USER)","=-"*10)

                print("[ITEM ID, NAME, PRICE]")
                uf.Items.viewTable()

                print("1. Place an order \n2. Search for an item \n0. Go back")
                func1 = int(input("Enter function: "))

                if func1 == 1:
                    uf.Items.order()
                    print("Item(s) added to cart")
                elif func1 == 2:
                    uf.Items.search()
                elif func1 == 0:
                    uf.Main.end()
                    break
        elif funcNum == 2:
            while True:
                print("-="*10,"CART","=-"*10)
                uf.Cart.viewTable()

                print("1. Confirm Order \n0. Go back")
                func2 = int(input("Enter function: "))

                if func2 == 1:
                    uf.Cart.confirm()
                    print("Order confirmed!")
```

```
            elif func2 == 0:
                uf.Main.end()
                break
    elif funcNum == 3:
        while True:
            print("-="*10,"DELIVERY","=-"*10)

            print("1. Check delivery status \n0. Go back")
            func3 = int(input("Enter function: "))

            if func3 == 1:
                uf.deliveryStatus()
            elif func3 == 0:
                uf.Main.end()
                break

def adminFunctions(funcNum):
    af.Main.start()
    if funcNum == 0:
        quit()
    elif funcNum == 1:
        while True:
            print("-="*10,"ITEMS(ADMIN)","=-"*10)
            print("1. Add item \n2. Delete item \n3. Search item \n4. View table
\n0. Go back")
            func1 = int(input("Enter function: "))

            if func1 == 1:
                af.Items.insertItem()
                print("Item added\n")
            elif func1 == 2:
                af.Items.deleteItem()
                print("Item deleted\n")
            elif func1 == 3:
                af.Items.searchItem()
            elif func1 == 4:
                af.Items.viewTable()
            elif func1 == 0:
```

```
                af.Main.end()
                break
            else:
                print("What are you doing here?")
    elif funcNum == 2:
        while True:
            print("-="*10,"CUSTOMERS(ADMIN)","=-"*10)
            print("1. View all customers \n2. Remove a customer \n3. Search for a
customer \n0. Go back")
            func2 = int(input("Enter function: "))

            if func2 == 1:
                af.Customers.viewTable()
            elif func2 == 2:
                af.Customers.removeCustomer()
            elif func2 == 3:
                af.Customers.searchCustomer()
            elif func2 == 0:
                af.Main.end()
                break
    elif funcNum == 3:
        while True:
            print("-="*10,"TRANSACTIONS(ADMIN)","=-"*10)
            print("1. View transactions \n2. Update transaction \n3. Check
transaction\n0. Go back")
            func3 = int(input("Enter function: "))

            if func3 == 1:
                af.Transactions.viewTable()
            elif func3 == 2:
                af.Transactions.update()
            elif func3 == 3:
                af.Transactions.check()
            elif func3 == 0:
                af.Main.end()
                break
    elif funcNum == 4:
        while True:
```

```
            print("-="*10,"DELIVERIES(ADMIN)","=-"*10)
            print("1. Check deliveries \n2. Search a delivery \n3. Update
delivery status \n0. Go back")
            func4 = int(input("Enter function: "))

            if func4 == 1:
                af.Deliveries.viewTable()
            elif func4 == 2:
                af.Deliveries.search()
            elif func4 == 3:
                af.Deliveries.updateStatus()
            elif func4 == 0:
                af.Main.end()
                break
```

File 3: “adminFunctions.py”

```
import mysql.connector as mys

# Common for all admin functions
class Main:
    def start():
        global db ; global c;
        db = mys.connect(host="localhost", user='root', passwd="root",
database="project")
        c = db.cursor()

    def end():
        c.close()
        db.close()


# functions related to table 'ITEMS'
class Items:
    def insertItem():
```

```
        iD = int(input("Enter item iD: "))
        name = input("Enter item name: ")
        price = int(input("Enter price: "))

        c.execute(f"insert into items values ({iD},'{name}',{price})")
        db.commit()

    def deleteItem():
        iD = int(input("Enter iD of item to be deleted: "))
        c.execute(f"delete from items where ITEM_ID={iD}")
        db.commit()

    def searchItem():
        iD = int(input("Enter iD of item to be searched: "))
        c.execute(f"select * from items where ITEM_ID={iD}")
        record = c.fetchone()
        print(record)

    def viewTable():
        c.execute("select * from items")
        table = c.fetchall()
        for i in table:
            print(i)


# functions related to table 'CUSTOMERS'
class Customers:
    def viewTable():
        c.execute("select * from customers")
        table = c.fetchall()
        for i in table:
            print(i)

    def removeCustomer():
        iD = input("Enter customer iD: ")
        c.execute(f"delete from customers where C_ID={iD}")
        db.commit()
```

```
    def searchCustomer():
        iD = int(input("Enter iD of customer to be searched: "))
        c.execute(f"select * from customers where C_ID={iD}")
        record = c.fetchone()
        print(record)


# functions related to table 'TRANSACTIONS'
class Transactions:
    def viewTable():
        c.execute("select * from transactions")
        table = c.fetchall()
        for i in table:
            print(i)

    def update():
        iD = int(input("Enter transaction ID: "))
        value = int(input("Completed/Incomplete (0/1): "))

        c.execute(f"update transactions set STATUS={value} where T_ID={iD}")
        db.commit()

        print("Updated!")

    def check():
        iD = int(input("Enter transaction ID: "))
        c.execute(f"select * from transactions where T_ID={iD}")
        check = c.fetchone()

        if check[-1] == 1:
            print("Transaction completed")
        else:
            print("Transaction incomplete")


# functions related to table 'DELIVERY'
class Deliveries:
    def viewTable():
```

```
        c.execute("select * from deliveries")
        table = c.fetchall()
        for i in table:
            print(i)

    def search():
        iD = int(input("Enter transaction ID: "))
        c.execute(f"select * from deliveries where T_ID={iD}")
        record = c.fetchone()
        print(record)

    def updateStatus():
        iD = int(input("Enter transaction ID: "))
        value = int(input("Enter status (0/1): "))

        c.execute(f"update deliveries set status={value} where T_ID={iD}")
        db.commit()

        print("Status updated!")
```

File 4: “userFunctions.py”

```
import mysql.connector as mys
from random import randint, choice

# A list of addresses for the deliveries table
ADDRESSES = ["8939 High Point St.","233 New Saddle St.","9197 Sunbeam Ave.","7800
E. Lake Forest Lane","224 Lower River St.","8603 Bedford Street","94 St Margarets
Dr.","688 Buckingham Dr.","378 Beaver Ridge Drive","82 W. Main Circle","423
Birchpond Dr.","636 Westport St.","444 Armstrong St.","50 Essex St.","8876 Hudson
Court","19 Spruce Drive","630 Hudson Street","56 East Windsor Street","7799 Queen
Street","89 Meadowbrook Rd.","33 Smith Dr.","8271 Fordham Ave.","522 Prince
Ave.","5 East Lilac Street","553 Trenton St."]

class Main:
```

```
    def start():
        global db ; global c;
        db = mys.connect(host="localhost", user='root', passwd="root",
database="project")
        c = db.cursor()

    def end():
        c.close()
        db.close()


class Items:
    def viewTable():
        c.execute("select * from items;")
        table = c.fetchall()
        for i in table:
            print(i)

    def order():
        try:
            global ciD;
            iD = int(input("Enter item ID: "))
            ciD = int(input("Enter customer ID to confirm: "))

            c.execute(f"select * from items where ITEM_ID={iD}")
            record = c.fetchone()

            temp = list(record)
            temp = [ciD]+temp

            c.execute(f"insert into cart values {tuple(temp)}")
            db.commit()
        except:
            print("Error Occured. Try again")

    def search():
        iD = int(input("Enter item ID: "))
```

```
        c.execute(f"select * from items where ITEM_ID={iD}")
        record = c.fetchone()

        if record is None:
            print("Item not available")
        else:
            print(record)


class Cart:
    def viewTable():
        c.execute("select * from cart")
        table = c.fetchall()
        for i in table:
            print(i)

    def confirm():
        moP = input("Enter mode of payment: (Cash/Credit/UPI): ")

        t_ID = randint(500, 599)
        address = choice(ADDRESSES)

        c.execute(f"insert into transactions values ({t_ID},{ciD},'{moP}', 0)")
        c.execute(f"insert into deliveries values ({t_ID},'{address}', 0)")
        c.execute("delete from cart")
        db.commit()

        print("Your transaction ID is:",t_ID)


def deliveryStatus():
    t_ID = int(input("Enter transaction ID: "))
    try:
        c.execute(f"select * from deliveries where T_ID={t_ID}")
        record = c.fetchone()

        if record[-1] == 0:
            print("Order awaiting delivery")
```

```
        else:
            print("Order has been delivered")
    except:
        print("Transaction ID not found")
```

Outputs

Logged in as user/customer

0. Logging in as new customer

```
(Login)
Enter customer ID (leave blank if new user):
Enter name: Persone
Enter email address: peronsl@gmail.com
Enter phone number: 123459876
Your Customer ID is 210
Enter it again: 210
Hiya Persone!
-+-+-+-+-+-+-+-+-+-+ WELCOME TO AMAZON'T +-+-+-+-+-+-+-+-+-+-


&=&=&=&=&=&=&=&=&=&=&=&=&=&=&= USER =&=&=&=&=&=&=&=&=&=&=&=&=&=&=&
1. View items
2. View Cart
3. Delivery status
0. Quit
Enter function:
```

1. Placing an order

```
(Login)
Enter customer ID (leave blank if new user): 210
Enter it again: 210
Hiya Persone!
-+-+-+-+-+-+-+-+-+-+ WELCOME TO AMAZON'T +-+-+-+-+-+-+-+-+-+-

&=&=&=&=&=&=&=&=&=&=&=&=&=&=&= USER =&=&=&=&=&=&=&=&=&=&=&=&=&=&
1. View items
2. View Cart
3. Delivery status
0. Quit
Enter function: 1
-=-=-=-=-=-=-=-=-=-= ITEMS(USER) =-=-=-=-=-=-=-=-=-
[ITEM ID, NAME, PRICE]
(101, 'Dum', 120)
(102, 'Dum1', 135)
(103, 'Dum2', 175)
1. Place an order
2. Search for an item
0. Go back
Enter function: 1
Enter item ID: 101
Enter customer ID to confirm: 210
Item(s) added to cart
```

2. Searching for an item

```
(Login)
Enter customer ID (leave blank if new user): 210
Enter it again: 210
Hiya Persone!
-+-+-+-+-+-+-+-+-+-+ WELCOME TO AMAZON'T +-+-+-+-+-+-+-+-+-+-
```

```
&=&=&=&=&=&=&=&=&=&=&=&=&=&= USER =&=&=&=&=&=&=&=&=&=&=&=&=&=&=&
1. View items
2. View Cart
3. Delivery status
0. Quit
Enter function:1
-=-=-=-=-=-=-=-=-=-= ITEMS(USER) =-=-=-=-=-=-=-=-=-=-
[ITEM ID, NAME, PRICE]
(101, 'Dum', 120)
(102, 'Dum1', 135)
(103, 'Dum2', 175)
1. Place an order
2. Search for an item
0. Go back
Enter function: 2
Enter item ID: 103
(103, 'Dum2', 175)
```

3. Confirming an order

```
(Login)
Enter customer ID (leave blank if new user): 210
Enter it again: 210
Hiya Persone!
-+-+-+-+-+-+-+-+-+-+ WELCOME TO AMAZON'T +-+-+-+-+-+-+-+-+-+-



&=&=&=&=&=&=&=&=&=&=&=&=&=&= USER =&=&=&=&=&=&=&=&=&=&=&=&=&=&=&
1. View items
2. View Cart
3. Delivery status
0. Quit
Enter function:2


-=-=-=-=-=-=-=-=-=-= CART =-=-=-=-=-=-=-=-=-=-
(210, 101, 'Dum', 120)
1. Confirm Order
0. Go back
Enter function: 1
Enter mode of payment: (Cash/Credit/UPI): Cash
Your transaction ID is: 587
Order confirmed!
```

4. Checking delivery status

```
(Login)
Enter customer ID (leave blank if new user): 210
Enter it again: 210
Hiya Persone!
-+-+-+-+-+-+-+-+-+-+ WELCOME TO AMAZON'T +-+-+-+-+-+-+-+-+-+-



&=&=&=&=&=&=&=&=&=&=&=&=&=&= USER =&=&=&=&=&=&=&=&=&=&=&=&=&=&
1. View items
2. View Cart
3. Delivery status
0. Quit
Enter function:3
-=-=-=-=-=-=-=-=-=-= DELIVERY =-=-=-=-=-=-=-=-=-=-
1. Check delivery status
0. Go back
Enter function: 1
Enter transaction ID: 587
Order awaiting delivery
```

Logged in as admin

1. Adding to table ITEMS

```
(Login)
Enter customer ID (leave blank if new user): 1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 1
-=-=-=-=-=-=-=-=-=-= ITEMS(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. Add item
2. Delete item
3. Search item
4. View table
0. Go back
Enter function: 1
Enter item iD: 104
Enter item name: Dum3
Enter price: 200
Item added
```

2. Deleting from table ITEMS

```
(Login)
Enter customer ID (leave blank if new user): 1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 1
-=-=-=-=-=-=-=-=-= ITEMS(ADMIN) =-=-=-=-=-=-=-=-=-
1. Add item
2. Delete item
3. Search item
4. View table
0. Go back
Enter function: 2
Enter iD of item to be deleted: 104
Item deleted
```

3. Searching in table ITEMS

```
(Login)
Enter customer ID (leave blank if new user): 1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 1
-=-=-=-=-=-=-=-=-= ITEMS(ADMIN) =-=-=-=-=-=-=-=-=-
1. Add item
2. Delete item
3. Search item
4. View table
0. Go back
Enter function: 3
Enter iD of item to be searched: 104
(104, 'Dum3', 200)
```

4. Viewing table ITEMS

```
(Login)
Enter customer ID (leave blank if new user): 1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 1
-=-=-=-=-=-=-=-=-=-= ITEMS(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. Add item
2. Delete item
3. Search item
4. View table
0. Go back
Enter function: 4
(101, 'Dum', 120)
(102, 'Dum1', 135)
(103, 'Dum2', 175)
(104, 'Dum3', 200)
```

5. Viewing table CUSTOMERS

```
(Login)
Enter customer ID (leave blank if new user):
1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 2
-=-=-=-=-=-=-=-=-=-= CUSTOMERS(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. View all customers
2. Remove a customer
3. Search for a customer
0. Go back
Enter function: 1
(201, 'S Vasan', 1234567890, 'irockz@gmail.com')
(202, 'P.P Ramani', 987654321, 'woman@gmail.com')
(203, 'Pedro', 918273645, 'humanbeing@gmail.com')
(210, 'Persone', 123459876, 'peronsl@gmail.com')
```

6. Deleting from table CUSTOMERS

```
(Login)
Enter customer ID (leave blank if new user):
1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 2
-=-=-=-=-=-=-=-=-=-= CUSTOMERS(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. View all customers
2. Remove a customer
3. Search for a customer
0. Go back
Enter function: 2
Enter customer iD: 210
Customer deleted
```

7. Searching in table CUSTOMERS

```
(Login)
Enter customer ID (leave blank if new user):
1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 2
-=-=-=-=-=-=-=-=-=-= CUSTOMERS(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. View all customers
2. Remove a customer
3. Search for a customer
0. Go back
Enter function: 3
Enter iD of customer to be searched: 210
(210, 'Persone', 123459876, 'peronsl@gmail.com')
```

8. Viewing table TRANSACTIONS

```
(Login)
Enter customer ID (leave blank if new user):
1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 3
-=-=-=-=-=-=-=-=-=-= TRANSACTIONS(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. View transactions
2. Update transaction
3. Check transaction
0. Go back
Enter function: 1
(554, 201, 'Credit', 1)
(584, 201, 'cash', 0)
(587, 210, 'Cash', 0)
(598, 102, 'Cash', 1)
```

9. Updating a transaction

```
(Login)
Enter customer ID (leave blank if new user):
1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 3
-=-=-=-=-=-=-=-=-=-= TRANSACTIONS(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. View transactions
2. Update transaction
3. Check transaction
0. Go back
Enter function: 2
Enter transaction ID: 587
Completed/Incomplete (0/1): 1
Updated!
```

10. Checking a transaction

```
(Login)
Enter customer ID (leave blank if new user):
1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 3
-=-=-=-=-=-=-=-=-=-= TRANSACTIONS(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. View transactions
2. Update transaction
3. Check transaction
0. Go back
Enter function: 3
Enter transaction ID: 587
Transaction completed
```

11. Viewing table DELIVERIES

```
(Login)
Enter customer ID (leave blank if new user):
1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 4
-=-=-=-=-=-=-=-=-=-= DELIVERIES(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. Check deliveries
2. Search a delivery
3. Update delivery status
0. Go back
Enter function: 1
(554, '553 Trenton St.', 1)
(584, '56 East Windsor Street', 0)
(587, '522 Prince Ave.', 0)
(598, '94 St Margarets Dr.', 1)
```

12. Searching for a delivery

```
(Login)
Enter customer ID (leave blank if new user):
1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 4
-=-=-=-=-=-=-=-=-=-= DELIVERIES(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. Check deliveries
2. Search a delivery
3. Update delivery status
0. Go back
Enter function: 2
Enter transaction ID: 587
(587, '522 Prince Ave.', 0)
```

13. Updating delivery status

```
(Login)
Enter customer ID (leave blank if new user):
1
Hello Admin!
-$-$-$-$-$-$-$-$-$-$-$-$-$-$-$ ADMIN FUNCTIONS $-$-$-$-$-$-$-$-$-$-$-$-$-$-$-
1. View/Alter items
2. Customers
3. Transactions
4. Delivery status
0. Quit
Enter function: 4
-=-=-=-=-=-=-=-=-=-= DELIVERIES(ADMIN) =-=-=-=-=-=-=-=-=-=-
1. Check deliveries
2. Search a delivery
3. Update delivery status
0. Go back
Enter function: 3
Enter transaction ID: 587
Enter status (0/1): 1
Status updated!
```

Conclusion

The program has been executed successfully and no errors have been found during execution

Thus, the program is economically and technically feasible with absence of compile-time, runtime, and logical errors. Ready for use

The source code can be further modified with more features according to the usability with the help of Python and MySQL

Bibliography

- www.python.org
- "Computer Science with python Class XII" by Sumita Arora
- github.com
- www.geeksforgeeks.org/
- www.programiz.com