Customer Management System

Prepared For:

Python Programming AML-1204

Prepared by-

First name	Last Name	Student number
Amandeep	Rathee	C0802428
Fahad Ahmed	Mohammed	C0790295
Milan	Lakhani	C0804324
Pratikgiri	Goswami	C0806977
Rabjot	Singh	C0805209

Guided By:

Ms. Parwinder Kaur

April 19, 2021

Lambton College in Toronto, Canada

Contents

Abstract

The aim of this project is to develop Customer Management System (CMS) which is one of the systems included in Enterprise Resource Planning (ERP). The objective of the project is to build a system that can help business to decrease the defection rate in billing and in turn enhance customer's experience. This project not only improves the speed of billing but also keeps track of existing customers.

The project includes several modules such as customer addition, customer modification, customer identification, and billing information. Each customer's information is stored in database using MySQL. Customer identification and modification help keep track of customer's purchase history. Billing module assists employer enter and display billing amount for customer. The tools used for this project are Python and MySQL.

Introduction

In today's commercial world, practice of dealing with existing customers and thriving business by getting more customers into loop is predominant. Tools like Customer Management System (CMS) can improve the situation and help in challenging the new ways of business in efficient manner.

CMS consists of a historical view of all acquired or to be acquired customers. This helps in reduced searching and correlating customers and to foresee customer needs effectively and increase business. CSM contains details of every customer, hence it is very easy to keep track a customer accordingly and can be used to determine which customer can be profitable and which not. It is not only used to deal with the existing customers but is also useful in acquiring new customers. The process first starts with identifying a customer and maintaining all the corresponding details into the system which is also called an 'Opportunity of Business'.

All the details in system are kept centralized which is available anytime on fingertips. This reduces the process time and increases productivity. The strongest aspect of Customer Management System is that it is very cost-effective. The advantage of decently implemented CMS is that there is very less need of paper and manual work which requires lesser staff to manage and lesser resources to deal with. The technologies used in implementing this CMS are also very cheap and smooth as compared to the traditional way of business.

The report further describes the methods used to implement this CMS, results of the same and concluded with the future scope.

Methods

This section describes different modules and components of the project along with the tools and technologies used.

Pre-Requisites

This section describes the basic requirements to run the project.

Technologies:

- **Python-3:** It is a general-purpose, high-level programming language. Python is interpreted, object-oriented, and easily readable which can also be used as a scripting language. It supports dynamic data types and dynamic type checking (tutprialspoint, n.d.) We have chosen Anaconda python distribution**.**

  Anaconda can be downloaded and installed from- Python.Org .

- **MySQL:** It is a web-accessed database system. MySQL is a server-based system that is an excellent option for both small and large projects. It is a fast, dependable, and simple database and makes use of regular SQL. MySQL runs on a variety of platforms; it is a free database to download and use. Oracle Corporation creates, distributes, and supports MySQL (w3schools, n.d.).

Tools:

- **Python IDE:** An IDE (or Integrated Development Environment) is a software development program. IDEs, as the name suggests, combine several resources that are specifically developed for software creation. It contains an editor made to work with code (with, for example, syntax highlighting and auto-completion)—tools for design, execution, and debugging, and Source control of some kind. We have chosen Spyder IDE which is an open-source environment integrated with the Anaconda distribution.
- **MySQL Workbench:** It is used by architects, developers, and DBAs, MySQL Workbench is a cohesive visual method. MySQL Workbench includes data modeling,

SQL creation, and server configuration, user administration, backup, and other administration tools. MySQL Workbench is a database management system that runs on Windows, Linux, and Mac OS X (MySQL Workbench, n.d.). To use MySQL, download MySQL Community Server and MySQL Workbench from the MySQL website download section.

Libraries:

- **MySQL- Connector:** Download and install MySQL Connector/Python from MySQL community and install the MySQL Connector library in the Anaconda Prompt as:

```
(base) C:\Users\hp>pip install mysql-connector-python
```

Project Files:

- **main.py:** The python file consists of all the modules and classes of the project. The file consists of a 'customer' class containing methods to perform various operations like adding customer details to the database, modifying, deleting, searching customer details, and displaying all records.
- **Testdb_customer.sql:** This file is the database file. It contains a database, namely ‘testdb,’ which includes the table ‘customer,’ which contains attributes such as id, name, items, and bill details of a customer.

Workflow

This section highlights the steps that we followed to create our project.

- ***Creating Database*:** To create a custom database, access the 'root' user by entering the password. Create a new database; in our project, we created a database as 'testdb'. The query to make a new database is:

```
create database testdb;
```

- ***Creating Customer Table*:** To create a new table navigate to '*testdb'* and in editor area type the following commands.

```
use testdb;

create table customer(
    id INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(30) NOT NULL,
    items VARCHAR(30) NOT NULL,
    bill int,
    PRIMARY KEY ( id )
);
```

- ***Importing Modules***

  As explained in the earlier sections of this report, we need the 'mysql.connector' library, and therefore the first step is to import this library using the code shown below.

```
import mysql.connector
```

- ***Connecting to MySQL Connector***

  There are three lines of code here for connecting to the MySQL Connector. Following is the screenshot of the three lines used in the project.

```
con = mysql.connector.connect(host="localhost", user="root", password="tiger10          se="testdb")
myCursor = con.cursor()
myCursor.execute("USE testdb")
```

Let us take these lines one by one and explain each attribute in them.

- The first line is as follows:

  Here we have used '. connect' function of the 'mysql.connector' library in which there are four significant arguments, 'Host', 'User,' 'Password' and 'Database'.

  Host - The name of the domain or IP (Internet Protocol) address where MySQL is installed. You should use localhost or its IP 127.0.0.1 if you're working on localhost.

  The ***username*** in which you interact with MySQL Server. The MySQL database's standard login name is root.

  The user specifies the ***password*** when the MySQL server is installed. You won't need the password if you're using root.

  The ***database*** title to which you want to bind and run the operations.

- The second code line is as follows:

```
myCursor = con.cursor()
```

  The MySQL Cursor class creates instances that can run commands like SQL statements. Cursor objects use a MySQL Connection object to communicate with the MySQL server.

- The third line in this step is as follows:

```
myCursor.execute("USE testdb")
```

We use the execute function for the cursor object created in the last line of code to tell the SQL system to “USE testdb”.

- ***Creating* Customer Class**

We have created a class named “Customer” with the following variables and functions:

Methods- Init (Constructor), addCus, modifyCus, searchCustomer, deleteCustomer and displayCustomer.

Variables – ‘id’ (Numerical), ‘name’ (String), ‘items’(Numerical) and ‘bill’ (String).

Let us look at every function in the class one by one.

- ***def __init__(self)***

Init function in python works like a constructor for the class variables. Therefore, we initialize the values for the variables id, name, items and bill in this function. Following is a screenshot to show how we did the latter.

```
def __init__(self):
    self.id = 0  # Instance variable
    self.name = ""
    self.items = ""
    self.bill = 0
```

- ***def addCus(self)***

This function purpose is to add a new customer to the database using SQL queries. For that first we create a cursor object. Then we create a string variable named ‘strQuery’ to store the query language in it. Customer data is stored in the

variable ‘customer1’ and using the execute function, we give the function strQuery and customer1 as arguments to run the SQL query to finally add the customer to the database. To run the code, we use the line “con.commit”. Following is the code snippet:

```
def addCus(self):
    myCursor = con.cursor()
    strQuery = "insert into customer values(%s,%s,%s,%s)"
    customer1=(self.id, self.name, self.items, self.bill)
    myCursor.execute(strQuery,customer1)
    con.commit()
```

- ***def modifyCus(self)***

In this function, we change the details of the customer according to the user’s preference, modifying the variables (is, name, items, bill). Like the rest of the function, we create a cursor variable, assign the SQL query to a string variable ‘strQuery’ and add the modified customer data. Lastly, we put as arguments both strQuery and the modified customer data in the execute function. To run the code, we use the line “con.commit”. Following is a snippet for the function code.

```
def modifyCus(self):
    myCursor = con.cursor()
    strQuery = "update Customer set name=%s, items=%s,bill=%s where id=%s"
    rowaffected = myCursor.execute(strQuery, (self.name, self.items, self.bill, self.id))
    con.commit()
    if (rowaffected == 0):
        raise Exception("The customer id doesn't exist")
```

- ***def searchCustomer()***

Here we have defined the function *searchCustomer()* to search the details of a customer using ID of the customer. We have used *cursor()* in the function which is a database object used to retrieve data from the customer in list format and using the *fetchone()* method we extracted the data in python string format, and then assigned values to the attributes name, item and bill. The method *fetchone()*

is used to retrieve data of a query from MySQL server, which is converted to Python objects.

```
def searchCustomer(self):
    myCursor = con.cursor()
    strQuery = "select * from Customer where id=%s"
    rowaffected = myCursor.execute(strQuery,(self.id,))
    if (rowaffected == 0):
        raise Exception("The customer id doesn't exist")
    data = myCursor.fetchone()
    self.name = data[1]
    self.items = data[2]
    self.bill = data[3]
```

- ***def deleteCustomer()***

We have defined the function *deleteCustomer()* to take the input of customer ID and then find it in the database to delete it, and if the ID is not found it throws an exception stating, “The customer id doesn’t exist”. Inside the function we have used *cursor()* method, *execute()* method and *commit()* method. The method *commit()* sends a statement to the MySQL server to COMMIT the current transaction. It is necessary to call this method after every transaction that changes the data.

```
def deleteCustomer(self):
    myCursor = con.cursor()
    strQuery = "delete from Customer where id=%s"
    rowaffected = myCursor.execute(strQuery,(self.id,))
    con.commit()
    if (rowaffected == 0):
        raise Exception("The customer id doesn't exist")
```

- ***def displayCustomer()***

We have defined a function *displayCustomer()* to display the details of the customer using ID as input. Inside the function we have used function *cursor()* to

retrieve the data, *fetchall()* to fetch the rows of a query result, *execute()* to update the database and then we used print function to print the customer details ID, Name, item and bill.

```
def displayCustomer(self):
    myCursor = con.cursor()
    strQuery = "select * from customer"
    myCursor.execute(strQuery)
    print("ID     Name     item    Bill")
    for row in myCursor.fetchall():
        for cell in row:
            print(cell, end='\t')
        print()
```

- ***Main Class:***

```
if(__name__=="__main__"):
```

Here, we have compared the *__name__* module with *__main__* module to prevent certain code from being run. In the main class we printed the instructions and given the user 6 choices to enter, search, modify, delete, retrieve the customer details or exit. We have used *if-elif-else* function to execute the choice made by the user. The choices are as given below.

```
print("\tEnter 1 to add New customer")
print("\tEnter 2 to search customer by id")
print("\tEnter 3 to Modify customer by id")
print("\tEnter 4 to Delete customer by id")
print("\tEnter 5 to display all customer")
print("\tEnter 6 to Exit")
```

- In **choice 1** we have asked to enter the details of new customer and stored it in the database using the function *addCus()* defined earlier.

```
Customer.addCus(cus)  #Calling Function
```

- In **choice 2** we have asked to enter the ID of the customer and searched the database using the function searchCustomer() defined earlier.

```
cus.searchCustomer() #Calling function Syntax 2
```

- In **choice 3** we have asked to enter the new details of the existing customer to modify and store it in the database using the function *modifycus()* defined earlier.

```
cus.modifyCus() #Calling function
```

- In **choice 4** we have asked to enter the ID of the customer and deleted it from the database using the function *deleteCustomer()* defined earlier.

```
Customer.deleteCustomer(cus)
```

- In **choice 5** we have asked to enter the ID of the customer to display it using the defined function *displayCustomer().*

```
cus.displayCustomer()
```

- In **choice 6** we exit the program using *break.*

```
break
```

Class Diagram:

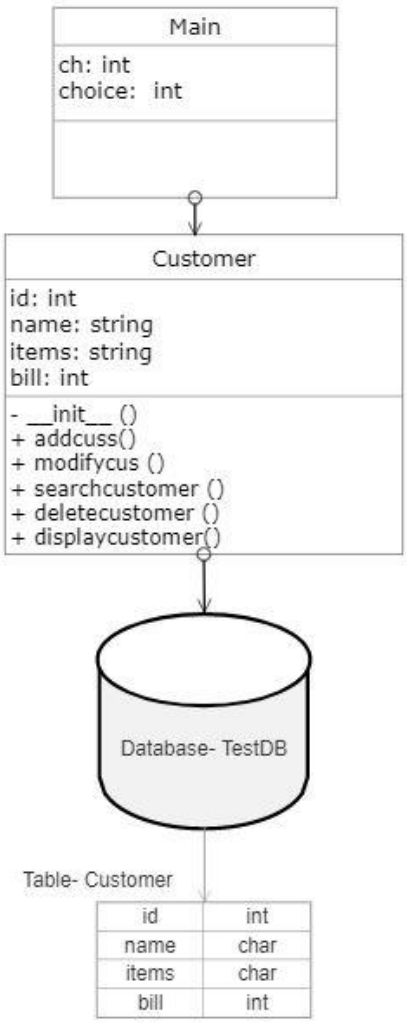

Fig1: UML Diagram

Results

The Database was created successfully and along with a successful implementation of Python code. The code implementation is as follows:

- Go to Command prompt or Anaconda prompt and navigate to the python file directory and open the file as ‘python main.py.’ for example-

```
E:\Python\Megastore_management>python main.py
```

- Choose from the menu different options such as:

```
        Enter 1 to add New customer
        Enter 2 to search customer by id
        Enter 3 to Modify customer by id
        Enter 4 to Delete customer by id
        Enter 5 to display all customer
        Enter 6 to Exit
Enter your choice:
```

- **Use Case 1**- Adding Customer:

```
Enter your choice: 1
Enter Customer id: 1
Enter Name: Aman
Enter items: Bread
Enter Bill Amount: 125
```

- **Use Case 2-** Search Customer:

```
Enter your choice: 2
Enter Id of customer to be searched: 1
Customer Name= Aman
Customer items= Bread
Customer Bill= 125
```

- **Use Case 3-** Modify Customer Details:

```
Enter your choice: 3
Enter Id of customer to be modified: 1
Enter  new name: Amandeep
Enter New items: Milk
Enter new Billing Amount: 50
Data Modified successully
```

- **Use Case 4-** Delete Customer from database:

```
Enter your choice: 4
Enter Id of customer to be deleted: 1
Customer Deleted successully
```

- **Use Case 5-** Display all Customer Records:

```
Enter your choice: 5
ID     Name      item     Bill
1       Aman     Milk     50
2       Pratik   Bread    25
3       Fahad    Chips    120
4       Rabjot   Chicken  200
5       Milan    Fish,Pie         250
Do you want to continue?
 Enter 'Y/y/1'- Yes and 'N/n' - No :
```

- **Use Case 6-** Exit from Application:

```
Enter your choice: 6
```

Conclusion & Future Work

The report exhibits the project goal, implementation, tools used and working of a Customer Management System. The project is a dependable and reliable solution for managing the customers. It encompasses the essential CRUD (Create, Read, Update and Delete) operations; necessary for maintaining customers in a business.

We have used the latest tools and technology for impeccable implementation and flawless execution. SQL is used as our database to ensure the security and robustness of one's valuable data.

There were many things we couldn't add to the project due to lack of time. Following is the list of functionality and ideas that we would love to add or improvise to our project in future:

1. We will have a better user experience by adding GUI (Graphical User Interface), making the app attractive and having smooth navigation.
2. We will make an API for our application, which will enable us to make desktop apps, web apps and mobile apps. It will also help to integrate our application into any existing business models.
3. We could add real-time analytics and sales forecasting with the help of machine learning algorithms and display results in GUI to help make businesses better decisions.
4. Auto-messaging and email services can notify customers of essential updates in future versions.

References

(n.d.). Retrieved from tutprialspoint: https://www.tutorialspoint.com/python3/index.htm

(n.d.). Retrieved from w3schools: https://www.w3schools.com/php/php_mysql_intro.asp

(n.d.). Retrieved from MySQL Workbench: https://www.mysql.com/products/workbench/