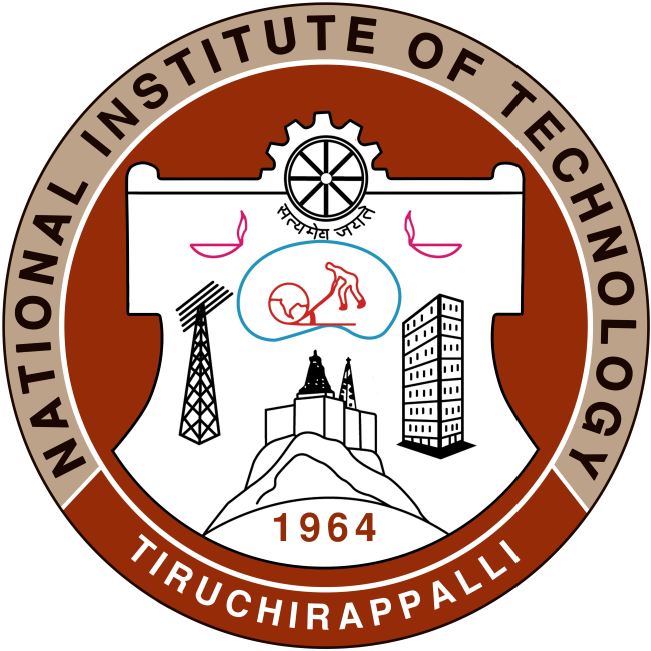
**NATIONAL INSTITUTE OF** **TECHNOLOGY, TIRUCHIRAPPALLI**

**Tamil Nadu-620015**



# *‘Database Management System’*

**PROJECT REPORT**

**STORE MANAGEMENT**

Submitted To: Submitted By:

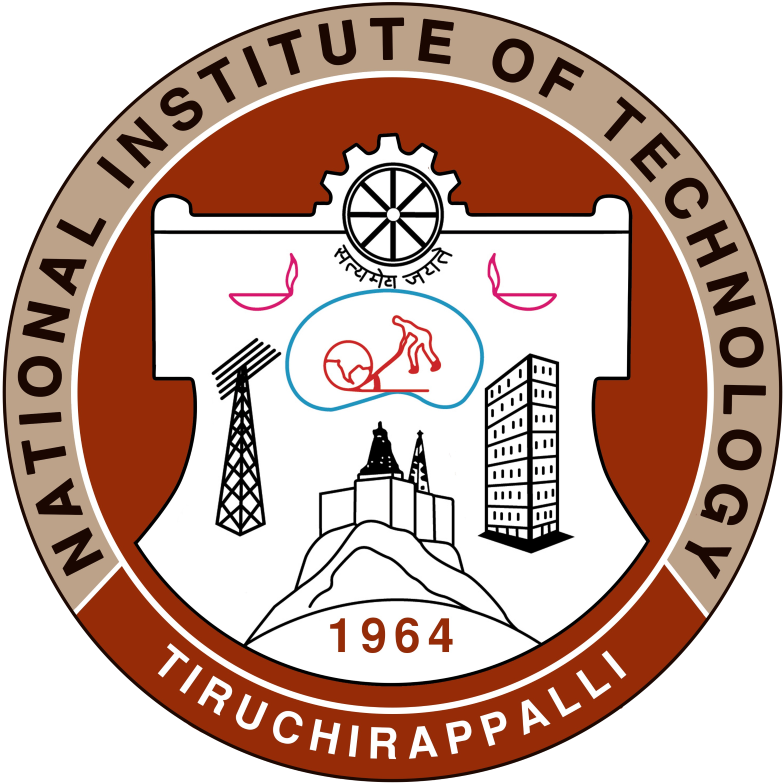
**Dr. U. Vignesh Akshay Jain**

**Roll No. – 205119007**

**MCA - II Semester ‘A’**

**NATIONAL INSTITUTE OF TECHNOLOGY,**

**TIRUCHIRAPPALLI-15**



## **CERTIFICATE**

*This is to certify that* ***Mr. AKSHAY JAIN,*** *student of 2nd semester MCA (batch 2019-2022) of National Institute of Technology, Tiruchirappalli has successfully completed the project* ***STORE MANAGEMENT*** *in Tkinter(Python)/MySQL**under the guidance of* ***Dr. U. Vignesh.***

Signature

***Dr. U. Vignesh***

## **Abstract**

The main aim of **Store Management** project is to keep a track of sales, purchases and their effect on inventory. We aim to demonstrate the use of create, read, select, update and delete MySQL operations through this project. The project starts by add items to the inventory by the seller, then a customer, purchase some products from seller (dealer) and note the changes to inventory. The purchased goods can be modified later which demonstrates the update functionality of project. Finally, we can record a sale and note the changes to product quantity in inventory page. The Application is built using Tkinter (Python) and MySQL technologies.

**ACKNOWLEDGEMENT**

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to NIT, Trichy for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards **Dr. U. Vignesh** for his kind co-operation and encouragement which help me in completion of this project.

**Dr. U. Vignesh**

(Department of Computer Applications)

# INTRODUCTION

A database management system (DBMS) refers to the technology for creating and managing databases. Basically DBMS is a software tool to organize (create, retrieve, update and manage) data in a database.

The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database.

Database systems are meant to handle large collection of information. Management of data involves both defining structures for storage of information and providing mechanisms that can do the manipulation of those stored information. Moreover, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

This project is aim at computerizing the manual process of wedding system. Front end and backend are implemented using Tkinter and MySQL respectively. The project consists of different forms(entity) namely Add, Update, Billing which are used for maintaining stock of store. The forms have number of entries. As well as each entry will be used to hold the information of items in the inventory.

**The services of a Store Management System can include:**

* Holding information about the items in stock.
* Adding information of new stocks.
* Updating information of current stocks.
* Searching information of item with the ID.
* Generating Invoice of items purchased by the customer.
* Keeping records of daily transactions.

**Database Management System**

DBMS stands for Database Management System. We can break it like this DBMS = Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can define DBMS like this: DBMS is a collection of inter-related data and set of programs to store and access those data in an easy and effective manner.

Database system are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: Storage of data and retrieval of data. According to the principles of database systems, the data is stored in such a way that it acquires a lot less space as the redundant data(duplicate data) has been removed before storage.

Along with storing the data in an optimized and systematic manner, It is also important that we retrieve the data quickly when needed. Database system ensures that data is retrieved as quickly as possible.

## **Applications of DBMS**

The development of computer graphics has been driven both by the needs of the user community and by the advances in hardware and software. The applications of database are many and varied; it can be divided into four major areas

1. Hierarchical and network system
2. Flexibility with relational database.
3. Object oriented application.
4. Interchanging the data on the web for e-commerce.

### **Display information**

In this particular project, we have taken Tkinter as a front end in order to display the information which are stored in the backend database called MySQL.

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the Tkinter module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

### **User Interfaces**

Our interactions with computers has become dominated by a visual paradigm that includes windows, buttons, menus, pointing device, such as a mouse. Although we are familiar with the syntax of MySQL, advances in MySQL have made possible other forms of advantages.

**What is MySQL?**

MySQL is multithreaded, multi user SQL database management System (DBMS). The basic program run as server providing multiuser access to a number of databases. The project’s source code is available under terms of the GNU(General Public Union), as well as under a variety of property arguments. MySQL is a database. The data in a MySQL is stored in a Database objects called tables. A table is a collection of related data entries and it consists of columns and rows. The databases are useful when storing information categorically.

MySQL is a central components of the LAMP open source web application software stack (and other “AMP” stacks). LAMP is an acronym for Linux, Apache, MySQL, Perl/PHP/Python. Application that use the MySQL database include PyCharm, TYP03, MODx, Joomla, WordPress, PHPBB, MyBB and Drupal. MySQL is also used in many high profile, large scale web sites, including Google(Though not for the searches).

**MySQL Command Syntax**

As you might have observed from the simple program in the previous section, MySQL uses mainly uses six commands in which SELECT is used to retrieve rows selected from one or more tables. FROM refers to the table from which we need to select the attributes. WHERE clause, if given, indicates condition or conditions that rows must satisfy to be selected. where\_ condition is expression that evaluates to true for each row to be selected. This statement selects all rows if there is no where clause. GROUP BY clause used to group the values of the attributes provided that values must be same. HAVING clause is applied nearly last, just before items are sent to the client, with no optimization. If the HAVING clause refers to a column that is ambiguous, warning occurs. ORDER BY clause is used for the purpose of sorting the values of the attributes in a result. If you use GROUP BY, output rows are sorted according to GROUP BY columns as if you had an ORDER BY for the same columns.

**Purpose**

The purpose of this project is to outline Inventory data and to recommend data management solutions and to provide a information regarding the stock. The purpose of this project is to develop a data management system to consolidate, organize, document, store and distribute information related to Store Management System.

A centralized database created to consolidate data, allowing integrated, long term analyses, and dynamic search ability with user friendly query tools to be performed to support adaptive management. Many data collection, analysis and presentation software programs that are currently being used must be able to interface with any new data management system. Continuity with consistent data collection methodology is enforced by a common database system, allowing for standardized format for forms ad reports between projects.

**Scope**

The scope of the project is managing a consistency and storage of data by dedicated data administrator. It provides most of the features that a Database Management System should have. It is developed by using MySQL database. It has been implemented in WINDOWS platform.

## **Hardware specification**

Processor : i3 Core Processor

Clock speed : 2.5GHz

Monitor : 1024 \* 768 Resolution Color

Keyboard : QWERTY

RAM : 1 GB

Input Output Console for interaction

## **Software specification**

MySQL Libraries

MySQL Client Server

PhyCham

Operating system : Windows7

## **DESIGN OF THE PROJECT**

This project has been developed using MySQL software which is queries oriented. Changes at the queries and the way in which it uses a system state may cause anticipated changes in the behaviour of other result.

## **Schema and Tables Description**

Following tables are used in the project:-

1. Inventory:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int | NO | PRI | NULL | auto\_increment |
| name | varchar(50) | NO |  | NULL |  |
| stock | int | NO |  | NULL |  |
| cp | int | YES |  | NULL |  |
| sp | int | YES |  | NULL |  |
| totalcp | int | YES |  | NULL |  |
| totalsp | int | YES |  | NULL |  |
| assumed\_pofit | int | YES |  | NULL |  |
| vendor | varchar(50) | YES |  | NULL |  |
| vendor\_phoneno | bigint | YES |  | NULL |  |

1. Transactions:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra |
| id | int | NO | PRI | NULL | auto\_increment |
| product\_name | varchar(50) | NO |  | NULL |  |
| quantity | int | NO |  | NULL |  |
| amount | int | YES |  | NULL |  |
| tdate | date | YES |  | NULL |  |

## **IMPLEMENTATION**

The project is implemented using MySQL database along with Tkinter.

**Implementation of Table Creation:**

1. Inventory –

create table if not exists inventory (id int not null auto\_increment, name varchar(50) not null, stock int not null, cp int, sp int, totalcp int, tatalsp int, assumed\_profit int, vendor varchar(50), vendor\_phoneno bigint, primary key(id));

A screen shot of a social media post

Description automatically generated

1. Transactions –

create table if not exists transactions (id int not null auto\_increment, product\_name varchar(50) not null, quantity int not null, amount int, tdate date, primary key(id));

A screenshot of a person

Description automatically generated

**Project Description:**

**DBMS Modules –**

1. Inventory: As the name suggests, records of items are hold in this table with the help of Primary key(id).
2. Transactions: Each and every transactions happening in the store are placed in this table with their respective date of selling.

**GUI Modules –**

1. Add items to Database: This form is used to add each and every item in the store to the inventory table of the store database by using insert query with their name, stock, cost price, selling price, total cost price, total selling price, vendor, vendor phone number.
2. Update information of items to database: This form is used to update information of any item present in the inventory table of store database with the help of it’s item’s id (Primary Key).
3. Search items: This form is used to search information about any item present in the database with it’s item’s id.
4. Generate Bill: This form is used to Generate Bill of all the items that a customer wants to buy and to calculate the total cost which is to be paid by the customer.
5. Change Button: This button is used to find how much change is to be returned to the customer if he paid more than the bill amount.
6. Print Invoice: By clicking generate a bill a process is initiated which will contact to the printing device so the invoice can be printed.
7. Clear Button: This Button is used to clear all entry field of the form.

**Source Code**

**GIT Link:**

<https://github.com/rudeakshay/205119007_DBMS_Project_StoreManagement>

**Note:** Source codes are in text format not snapshots.

//source code for **add\_to\_db.py**

# import all the modules

import tkinter

from tkinter import \*

import mysql.connector

import tkinter.messagebox

conn = mysql.connector.connect(*host*='localhost', *user*='root', *passwd*='7729', *database*='store',*use\_pure*=True )

con = conn.cursor()

s = "create table if not exists inventory(id int not null auto\_increment, name varchar(50) not null, stock int not null," \

    "cp int, sp int, totalcp int, tatalsp int, assumed\_profit int, vendor varchar(50), vendor\_phoneno bigint, primary key(id))"

con.execute(s)

conn.commit()

con.execute("SELECT Max(id) FROM inventory")

result = con.fetchall()

if result:

    for r in result:

        id = r[0]

*class* Database:

*def* \_\_init\_\_(*self*, *master*, \**args*, \*\**kwargs*):

        self.master = master

        self.heading = Label(master, *text*="Add To The Database", *font*=('arial 40 bold'), *fg*='steelblue')

        self.heading.place(*x*=450, *y*=0)

        # labels for the window

        self.name\_l = Label(master, *text*="Enter Product Name", *font*=('arial 18 bold'))

        self.name\_l.place(*x*=0, *y*=70)

        self.stock\_l = Label(master, *text*="Enter Stocks", *font*=("arial 18 bold"))

        self.stock\_l.place(*x*=0, *y*=120)

        self.cp\_l = Label(master, *text*="Enter Cost Price", *font*=("arial 18 bold"))

        self.cp\_l.place(*x*=0, *y*=170)

        self.sp\_l = Label(master, *text*="Enter Selling Price", *font*=("arial 18 bold"))

        self.sp\_l.place(*x*=0, *y*=220)

        self.vendor\_l = Label(master, *text*="Enter Vendor Name", *font*=("arial 18 bold"))

        self.vendor\_l.place(*x*=0, *y*=270)

        self.vendor\_phone\_l = Label(master, *text*="Enter Vendor Phone Number", *font*=("arial 18 bold"))

        self.vendor\_phone\_l.place(*x*=0, *y*=320)

        self.id\_l = Label(master, *text*="Enter ID", *font*=("arial 18 bold"))

        self.id\_l.place(*x*=0, *y*=370)

        # entries for lables

        self.name\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.name\_e.place(*x*=400, *y*=70)

        self.stock\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.stock\_e.place(*x*=400, *y*=120)

        self.cp\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.cp\_e.place(*x*=400, *y*=170)

        self.sp\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.sp\_e.place(*x*=400, *y*=220)

        self.vendor\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.vendor\_e.place(*x*=400, *y*=270)

        self.vendor\_phone\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.vendor\_phone\_e.place(*x*=400, *y*=320)

        self.id\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.id\_e.place(*x*=400,*y*=370)

        # button to add to the database

        self.btn\_clear = Button(master, *text*="Clear All Fields", *width*=25, *height*=2, *bg*='red', *fg*='white', *command*=self.clear\_all)

        self.btn\_clear.place(*x*=320, *y*=420)

        self.btn\_add = Button(master, *text*="Add To Database", *width*=25, *height*=2, *bg*='steelblue', *fg*='white', *command* = self.get\_items)

        self.btn\_add.place(*x*=520, *y*=420)

        # text box for the logs

        self.tBox = Text(master, *width*=60, *height*=20)

        self.tBox.place(*x*=800, *y*=70)

        self.tBox.insert(END, "ID has reached upto: " + *str*(id))

*def* get\_items(*self*, \**args*, \*\**kwaargs*):

        conn = mysql.connector.connect(*host*='localhost', *user*='root', *passwd*='7729', *database*='store',*use\_pure*=True )

        con = conn.cursor()

        # get data from entries

        self.name = self.name\_e.get()

        self.stock = self.stock\_e.get()

        self.cp = self.cp\_e.get()

        self.sp = self.sp\_e.get()

        self.vendor = self.vendor\_e.get()

        self.vendor\_phone = self.vendor\_phone\_e.get()

        # dynamic entries

        self.totalcp = *float*(self.cp) \* *float*(self.stock)

        self.totalsp = *float*(self.sp) \* *float*(self.stock)

        self.assumed\_profit = *float*(self.totalsp - self.totalcp)

        if self.name == '' or self.stock == '' or self.cp == '' or self.sp == '':

            tkinter.messagebox.showinfo("Error", "Please Fill All Entries.")

        else:

            sql = "INSERT INTO inventory (name, stock, cp, sp, totalcp, tatalsp, assumed\_profit, vendor, vendor\_phoneno) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"

            con.execute(sql, (self.name, self.stock, self.cp, self.sp, self.totalcp, self.totalsp, self.assumed\_profit, self.vendor, self.vendor\_phone))

            conn.commit()

            self.tBox.insert(END, "\n\nInserted " + *str*(self.name) + " into the database with code " + *str*(self.id\_e.get()))

            tkinter.messagebox.showinfo("Success", "Successfully added to the Database.")

        conn.close()

*def* clear\_all(*self*, \**args*, \*\**kwargs*):

        # num=id+1

        self.name\_e.delete(0, END)

        self.stock\_e.delete(0, END)

        self.cp\_e.delete(0, END)

        self.sp\_e.delete(0, END)

        self.vendor\_e.delete(0, END)

        self.vendor\_phone\_e.delete(0, END)

        self.id\_e.delete(0, END)

root = Tk()

b = Database(root)

root.geometry("1366x768+0+0")

root.title("Add To Database")

root.mainloop()

//source code for **update.py**

# import all the modules

import tkinter

from tkinter import \*

import mysql.connector

import tkinter.messagebox

conn = mysql.connector.connect(*host*='localhost', *user*='root', *passwd*='7729', *database*='store', *use\_pure*=True)

con = conn.cursor()

s = "create table if not exists inventory(id int not null auto\_increment, name varchar(50) not null, stock int not null," \

    "cp int, sp int, totalcp int, tatalsp int, assumed\_profit int, vendor varchar(50), vendor\_phoneno bigint, primary key(id))"

con.execute(s)

conn.commit()

con.execute("SELECT Max(id) FROM inventory")

result = con.fetchall()

if result:

    for r in result:

        id = r[0]

*class* Database:

*def* \_\_init\_\_(*self*, *master*, \**args*, \*\**kwargs*):

        self.master = master

        self.heading = Label(master, *text*="Update To The Database", *font*=('arial 40 bold'), *fg*='steelblue')

        self.heading.place(*x*=400, *y*=0)

        # label and entry for id

        self.id\_le = Label(master, *text*="Enter ID", *font*=("arial 18 bold"))

        self.id\_le.place(*x*=0, *y*=70)

        self.id\_leb = Entry(master, *width*=10, *font*=('arial 20 bold'))

        self.id\_leb.place(*x*=400, *y*=70)

        self.btn\_search = Button(master, *text*="Search", *width*=15, *height*=2, *bg*='orange', *command*=self.search)

        self.btn\_search.place(*x*=580, *y*=70)

        # labels for the window

        self.name\_l = Label(master, *text*="Enter Product Name", *font*=('arial 18 bold'))

        self.name\_l.place(*x*=0, *y*=120)

        self.stock\_l = Label(master, *text*="Enter Stocks", *font*=("arial 18 bold"))

        self.stock\_l.place(*x*=0, *y*=170)

        self.cp\_l = Label(master, *text*="Enter Cost Price", *font*=("arial 18 bold"))

        self.cp\_l.place(*x*=0, *y*=220)

        self.sp\_l = Label(master, *text*="Enter Selling Price", *font*=("arial 18 bold"))

        self.sp\_l.place(*x*=0, *y*=270)

        self.totalcp\_l = Label(master, *text*="Enter Total Cost Price", *font*=("arial 18 bold"))

        self.totalcp\_l.place(*x*=0, *y*=320)

        self.totalsp\_l = Label(master, *text*="Enter Total Selling Price", *font*=("arial 18 bold"))

        self.totalsp\_l.place(*x*=0, *y*=370)

        self.vendor\_l = Label(master, *text*="Enter Vendor Name", *font*=("arial 18 bold"))

        self.vendor\_l.place(*x*=0, *y*=420)

        self.vendor\_phone\_l = Label(master, *text*="Enter Vendor Phone Number", *font*=("arial 18 bold"))

        self.vendor\_phone\_l.place(*x*=0, *y*=470)

        # entries for lables

        self.name\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.name\_e.place(*x*=400, *y*=120)

        self.stock\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.stock\_e.place(*x*=400, *y*=170)

        self.cp\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.cp\_e.place(*x*=400, *y*=220)

        self.sp\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.sp\_e.place(*x*=400, *y*=270)

        self.totalcp\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.totalcp\_e.place(*x*=400, *y*=320)

        self.totalsp\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.totalsp\_e.place(*x*=400, *y*=370)

        self.vendor\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.vendor\_e.place(*x*=400, *y*=420)

        self.vendor\_phone\_e = Entry(master, *width*=25, *font*=('arial 18 bold'))

        self.vendor\_phone\_e.place(*x*=400, *y*=470)

        # button to add to the database

        self.btn\_clear = Button(master, *text*="Clear All Fields", *width*=25, *height*=2, *bg*='red', *fg*='white')

        self.btn\_clear.place(*x*=320, *y*=520)

        self.btn\_update = Button(master, *text*="Update Database", *width*=25, *height*=2, *bg*='steelblue', *fg*='white', *command*=self.update)

        self.btn\_update.place(*x*=520, *y*=520)

        # text box for the logs

        self.tBox = Text(master, *width*=60, *height*=20)

        self.tBox.place(*x*=800, *y*=70)

        self.tBox.insert(END, "")

*def* search(*self*, \**args*, \*\**kwargs*):

        sql = "SELECT \* FROM inventory WHERE id=%s"

        con.execute(sql, (self.id\_leb.get(), ))

        result = con.fetchall()

        for r in result:

            self.n1 = r[1]  # name

            self.n2 = r[2]  # stock

            self.n3 = r[3]  # cp

            self.n4 = r[4]  # sp

            self.n5 = r[5]  # totalcp

            self.n6 = r[6]  # tatalsp

            self.n7 = r[7]  # assumed\_profit

            self.n8 = r[8]  # vendor

            self.n9 = r[9]  # vendor\_phoneno

        conn.commit()

        # insert into the entries to update

        self.name\_e.delete(0, END)

        self.name\_e.insert(0, *str*(self.n1))

        self.stock\_e.delete(0, END)

        self.stock\_e.insert(0, *str*(self.n2))

        self.cp\_e.delete(0, END)

        self.cp\_e.insert(0, *str*(self.n3))

        self.sp\_e.delete(0, END)

        self.sp\_e.insert(0, *str*(self.n4))

        self.totalcp\_e.delete(0, END)

        self.totalcp\_e.insert(0, *str*(self.n5))

        self.totalsp\_e.delete(0, END)

        self.totalsp\_e.insert(0, *str*(self.n6))

        self.vendor\_e.delete(0, END)

        self.vendor\_e.insert(0, *str*(self.n8))

        self.vendor\_phone\_e.delete(0, END)

        self.vendor\_phone\_e.insert(0, *str*(self.n9))

*def* update(*self*, \**args*, \*\**kwargs*):

        con = conn.cursor()

        # get all updated values

        self.u1 = self.name\_e.get()

        self.u2 = self.stock\_e.get()

        self.u3 = self.cp\_e.get()

        self.u4 = self.sp\_e.get()

        self.u5 = self.totalcp\_e.get()

        self.u6 = self.totalsp\_e.get()

        self.u7 = self.vendor\_e.get()

        self.u8 = self.vendor\_phone\_e.get()

        query = "UPDATE inventory SET name=%s, stock=%s, cp=%s, sp=%s, totalcp=%s, tatalsp=%s, vendor=%s, vendor\_phoneno=%s WHERE id=%s"

        con.execute(query, (self.u1, self.u2, self.u3, self.u4, self.u5, self.u6, self.u7, self.u8, self.id\_leb.get()))

        conn.commit()

        tkinter.messagebox.showinfo("Success","Database Updated!!!")

root = Tk()

b = Database(root)

root.geometry("1366x768+0+0")

root.title("Update To Database")

root.mainloop()

//source code for **main.py**

# import all the modules

from tkinter import \*

import mysql.connector

import tkinter.messagebox

import datetime

import os

import random

conn = mysql.connector.connect(*host*='localhost', *user*='root', *passwd*='7729', *database*='store',*use\_pure*=True )

con = conn.cursor()

# creating transactions table

s = "create table if not exists transactions(id int not null auto\_increment, product\_name varchar(50) not null, quantity int not null," \

    "amount int, tdate date, primary key(id))"

con.execute(s)

conn.commit()

# date

date = datetime.datetime.now().date()

# temporary lists like sessions

products\_list = []

product\_price = []

product\_quantity = []

product\_id =[]

# labels list

lables\_list = []

*class* Application:

*def* \_\_init\_\_(*self*, *master*, \**args*, \*\**kwargs*):

        self.master = master

        # frames

        self.left = Frame(master, *width*=700, *height*=768, *bg*='white')

        self.left.pack(*side*=LEFT)

        self.right = Frame(master, *width*=666, *height*=768, *bg*='lightblue')

        self.right.pack(*side*=RIGHT)

        # components

        self.heading = Label(self.left, *text*="2k Market", *font*=('arial 40 bold'), *bg*='white')

        self.heading.place(*x*=0,*y*=0)

        self.date\_l = Label(self.right, *text*="Today's Date: " + *str*(date), *font*=('arial 18 bold'), *bg*='lightblue', *fg*='white')

        self.date\_l.place(*x*=0, *y*=0)

        # table invoice ==================================================================

        self.tproduct = Label(self.right, *text*="Products", *font*=('arial 18 bold'), *bg*='lightblue', *fg*='white')

        self.tproduct.place(*x*=0,*y*=60)

        self.tquantity = Label(self.right, *text*="Quantity", *font*=('arial 18 bold'), *bg*='lightblue', *fg*='white')

        self.tquantity.place(*x*=300, *y*=60)

        self.tamount = Label(self.right, *text*="Amount", *font*=('arial 18 bold'), *bg*='lightblue', *fg*='white')

        self.tamount.place(*x*=500, *y*=60)

        # enter stuff

        self.enterid = Label(self.left, *text*="Enter Product's ID", *font*=('arial 18 bold'), *bg*='white')

        self.enterid.place(*x*=0, *y*=80)

        self.enteride = Entry(self.left, *width*=25, *font*=('arial 18 bold'), *bg*='lightblue')

        self.enteride.place(*x*=230, *y*=80)

        self.enteride.focus()

        self.search\_btn = Button(self.left, *text*="Search", *width*=22, *height*=2, *bg*='orange', *command*=self.ajax)

        self.search\_btn.place(*x*=350, *y*=120)

        # fill it by the function ajax

        self.productname = Label(self.left, *text*="", *font*=('arial 27 bold'), *bg*='white', *fg*='steelblue')

        self.productname.place(*x*=0, *y*=250)

        self.pprice = Label(self.left, *text*="", *font*=('arial 27 bold'), *bg*='white', *fg*='steelblue')

        self.pprice.place(*x*=0, *y*=290)

        # total label

        self.total\_l= Label(self.right, *text*="", *font*=('arial 40 bold'), *bg*='lightblue', *fg*='white')

        self.total\_l.place(*x*=0, *y*=600)

        self.master.bind("<Return>", self.ajax)

        self.master.bind("<Up>", self.add\_to\_cart)

        self.master.bind("<space>", self.generate\_bill)

*def* ajax(*self*, \**args*, \*\**kwargs*):

        self.get\_id=self.enteride.get()

        # get the productsinfo with there id and fill in the labels above

        query="SELECT \* FROM inventory WHERE id=%s"

        con.execute(query,(self.get\_id, ))

        result=con.fetchall()

        for self.r in result:

            self.get\_id= self.r[0]

            self.get\_name= self.r[1]

            self.get\_price= self.r[4]

            self.get\_stock= self.r[2]

        self.productname.configure(*text*="Product's Name: " + *str*(self.get\_name))

        self.pprice.configure(*text*="Price: Rs. " + *str*(self.get\_price))

        # create the quatity and discount label

        self.quantity\_l=Label(self.left, *text*="Enter Quantity", *font*=('arial 18 bold'), *bg*='white')

        self.quantity\_l.place(*x*=0, *y*=370)

        self.quantity\_e= Entry(self.left, *width*=25,*font*=('arial 18 bold'), *bg*='light blue')

        self.quantity\_e.place(*x*=190, *y*=370)

        self.quantity\_e.focus()

        self.discount\_l = Label(self.left, *text*="Enter Discount", *font*=('arial 18 bold'), *bg*='white')

        self.discount\_l.place(*x*=0, *y*=410)

        self.discount\_e = Entry(self.left, *width*=25, *font*=('arial 18 bold'), *bg*='light blue')

        self.discount\_e.place(*x*=190, *y*=410)

        self.discount\_e.insert(END, 0)

        # add to cart button

        self.add\_to\_cart\_btn = Button(self.left, *text*="Add To Cart", *width*=22, *height*=2, *bg*='orange', *command*=self.add\_to\_cart)

        self.add\_to\_cart\_btn.place(*x*=350, *y*=450)

        #generate bills and change

        self.change\_l=Label(self.left,*text*="Given Amount", *font*=('arial 18 bold'), *bg*='white')

        self.change\_l.place(*x*=0, *y*=550)

        self.change\_e= Entry(self.left, *width* =25, *font*=('arial 18 bold'), *bg*='lightblue')

        self.change\_e.place(*x*=190, *y*=550)

        #button change

        self.change\_btn = Button(self.left, *text*="Calculate Change", *width*=22, *height*=2, *bg*='orange', *command*=self.change\_fun)

        self.change\_btn.place(*x*=350, *y*=590)

        #generate bill button

        self.generate\_bill\_btn = Button(self.left, *text*="Generate Bill", *width*=80, *height*=2, *bg*='red', *command*=self.generate\_bill)

        self.generate\_bill\_btn.place(*x*=60, *y*=640)

*def* add\_to\_cart(*self*, \**args*, \*\**kwargs*):

        #get the quantity value from the database

        self.quantity\_value= *int*(self.quantity\_e.get())

        if self.quantity\_value > *int*(self.get\_stock):

            tkinter.messagebox.showinfo("Error","Not that many Products in our Inventory.")

        else:

            #calculate the price

            self.final\_price= *float*(self.quantity\_value)\**float*(self.get\_price)-(*float*(self.discount\_e.get()))

            products\_list.append(self.get\_name)

            product\_price.append(self.final\_price)

            product\_quantity.append(self.quantity\_value)

            product\_id.append(self.get\_id)

            self.x\_index=0

            self.y\_index=100

            self.counter=0

            for self.p in products\_list:

                self.tempname= Label(self.right, *text*=*str*(products\_list[self.counter]), *font* =('arial 18 bold') , *bg*='lightblue', *fg*='white')

                self.tempname.place(*x*=0, *y*=self.y\_index)

                lables\_list.append(self.tempname)

                self.tempqt = Label(self.right, *text*=*str*(product\_quantity[self.counter]), *font*=('arial 18 bold'), *bg*='lightblue',*fg*='white')

                self.tempqt.place(*x*=300, *y*=self.y\_index)

                lables\_list.append(self.tempqt)

                self.tempprice = Label(self.right, *text*=*str*(product\_price[self.counter]), *font*=('arial 18 bold'), *bg*='lightblue',*fg*='white')

                self.tempprice.place(*x*=500, *y*=self.y\_index)

                lables\_list.append(self.tempprice)

                self.y\_index+=40

                self.counter+=1

                # total configure

                self.total\_l.configure(*text*="Total Rs. " + *str*(sum(product\_price)))

                # delete

                self.productname.configure(*text*="")

                self.pprice.configure(*text*="")

                self.quantity\_l.place\_forget()

                self.quantity\_e.place\_forget()

                self.discount\_l.place\_forget()

                self.discount\_e.place\_forget()

                self.add\_to\_cart\_btn.destroy()

                # autofocus to the enter id

                self.enteride.focus()

                self.enteride.delete(0, END)

*def* change\_fun(*self*):

        # get the amount given by the customer and the amount generated by the computer

        self.amount\_given = *float*(self.change\_e.get())

        self.our\_total = *float*(sum(product\_price))

        self.to\_give = self.amount\_given - self.our\_total

        #label change

        self.c\_amount = Label(self.left, *text*="Change: Rs. "+*str*(self.to\_give), *font*=('arial 18 bold'), *fg*='red', *bg*='white')

        self.c\_amount.place(*x*=0, *y*=600)

*def* generate\_bill(*self*, \**args*, \*\**kwargs*):

        # create the bill before updating the database

        directory = "C:/New folder/StoreManagement/Invoice/" + *str*(date) + "/"

        if not os.path.exists(directory):

            os.makedirs(directory)

        # templates for the bill

        company = "\t\t\t\t\t2K Market\n"

        address = "\t\tNational Institute of Technology, Tiruchirappalli\n"

        phone = "\t\t\t\t\t9876543210\n"

        sample = "\t\t\t\t\tInvoice\n"

        dt = "\t\t\t\t\t" + *str*(date)

        table\_header = "\n\n\t-----------------------------------------------------------\n\t\tSNo.\t\tProducts\t\tQty\t\tAmount\n\t-----------------------------------------------------------"

        final = company + address + phone + sample + dt + "\n" + table\_header

        # open a file to write it to

        file\_name = *str*(directory) + *str*(random.randrange(5000,10000)) + ".rtf"

        f = open(file\_name, 'w')

        f.write(final)

        # fill dynamics

        r = 1

        i = 0

        for t in products\_list:

            f.write("\n\t\t" + *str*(r) + "\t\t" + *str*(products\_list[i] + "     ")[:10] + "\t\t" + *str*(product\_quantity[i]) +"\t\t" + *str*(product\_price[i]))

            r += 1

            i += 1

        f.write("\n\n\t\t\t\tTotal Amount Rs. " + *str*(sum(product\_price)))

        f.write("\n\t\t\t\tThanks for Visiting.")

        os.startfile(file\_name, "print")

        f.close()

        # decrease the stock

        self.x = 0

        for i in products\_list:

            initial = "SELECT \* FROM inventory WHERE id=%s"

            con.execute(initial, (product\_id[self.x],))

            result = con.fetchall()

            for r in result:

                self.old\_stock = r[2]

            self.new\_stock = *int*(self.old\_stock)-*int*(product\_quantity[self.x])

            # updating the stock

            sql="UPDATE inventory SET stock=%s WHERE id=%s"

            con.execute(sql, (self.new\_stock, product\_id[self.x]))

            conn.commit()

            # insert into the transaction

            sql2 = "INSERT INTO transactions(product\_name, quantity, amount, tdate) VALUES (%s,%s,%s,%s)"

            con.execute(sql2,(products\_list[self.x], product\_quantity[self.x], product\_price[self.x], date))

            conn.commit()

            self.x += 1

        for a in lables\_list:

            a.destroy()

        del(product\_id[:])

        del(products\_list[:])

        del(product\_quantity[:])

        del(product\_price[:])

        self.total\_l.configure(*text*="")

        self.c\_amount.configure(*text*="")

        self.change\_e.delete(0, END)

        self.enteride.focus()

        tkinter.messagebox.showinfo("Success", "Done Everything Smoothly.")

root = Tk()

b = Application(root)

root.geometry("1366x768+0+0")

root.mainloop()

**Snapshots**

1. **Form for Adding Item’s Information’s to Database:**

**A screenshot of a cell phone

Description automatically generated**

1. **Adding Item’s Information’s to Database:**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

1. **Form for Updating Item’s Information’s to Database:**

**A screenshot of a cell phone

Description automatically generated**

1. **Updating Item’s Information’s to Database:**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

1. **2k Market Form:**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

1. **Inventory Table After Adding Data:**

**A screenshot of a computer screen

Description automatically generated**

1. **Inventory Table After Updating Data:**

**A screen shot of a computer

Description automatically generated**

1. **Transactions Table After Generating Bills:**

**A screenshot of a computer screen

Description automatically generated**

1. **Generated Invoice:**

**A screenshot of a social media post

Description automatically generated**

**References**

* <https://www.w3schools.com/sql/sql_intro.asp>
* <https://docs.python.org/3/library/tkinter.html>
* <https://stackoverflow.com/>
* <https://www.jetbrains.com/pycharm/>
* <https://www.youtube.com/>
* <https://www.tutorialspoint.com/python/python_gui_programming.htm>
* <https://www.tutorialspoint.com/dbms/index.htm>
* <https://www.javatpoint.com/dbms-tutorial>
* <https://realpython.com/python-gui-tkinter/>

**-Thank You-**

**--END--**