

**SESSION 2022-23**

**PROJECT FILE**

GROCERY STORE MANAGEMENT  
SYSTEM

**Submitted By**

**Abhik Ghosh**

**XII – Aryabhatta**

**Roll No : 2**

**CONTENTS**

* Certificate
* Acknowledgement
* Introduction
* Features
* Validations and Checks
* Hardware Used
* Software Used
* System Structure
* Screenshots
* Program Code
* Annexure
* Bibliography

**CERTIFICATE**

This is to certify that ABHIK GHOSH of class XII Aryabhatta has prepared the Grocery Store Management System Project. The project report is the result of his efforts and endeavours. The report is found worthy of acceptance as the project report under the subject computer science of class XII. He has prepared the report under my guidance.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ms Parul Kapil)

**ACKNOWLEDGEMENT**

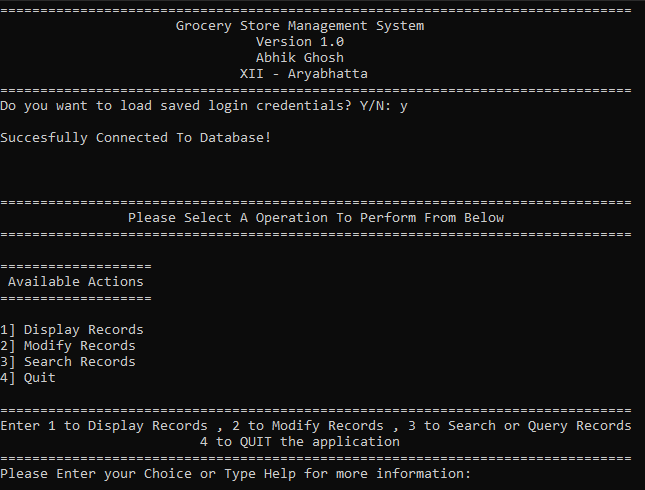
I would like to express my special thanks of gratitude to my teacher MS.PARUL KAPIL as well as our principal DR.RUCHI SETH who gave me the golden opportunity to do this wonderful project on the topic Grocery Store Management System, which also helped me in doing a lot of Research and I came to know about python and MYSQL I am really thankful to them. Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

**INTRODUCTION**

What is a Grocery Store Management System ?

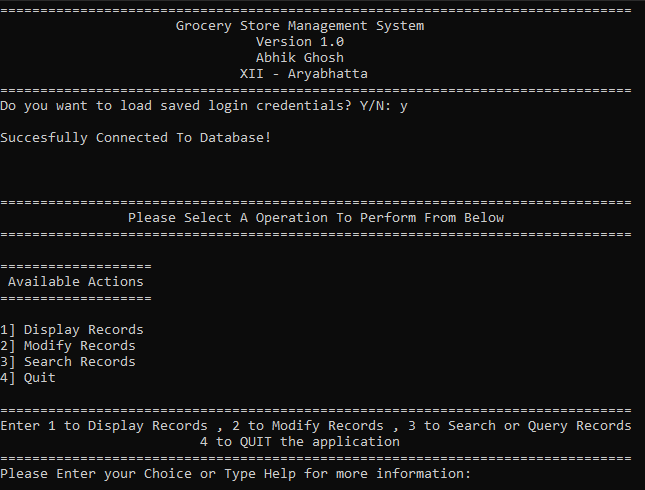
A Grocery Store Management System is a piece of software that can handle inventory checking , price checking , discount calculations etc of all products to be sold at a grocery store and is a must have for all stores.

This is achieved by using a database which can be setup in cloud server, offline network or single computer. Grocery Shop Management System is **managing all the information about products, product brand, product category, vendors, customers and product stock**. Multiple user can access this software.



**FEATURES**

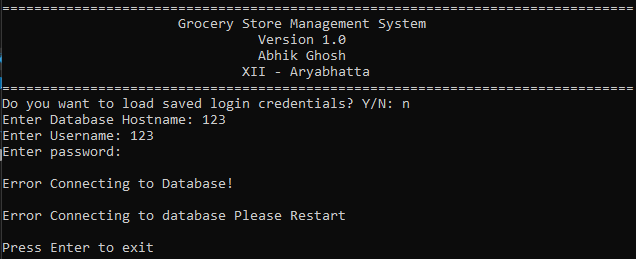
* Shipment Management
* Real-time inventory tracking
* Data is displayed in user friendly Tabular Format
* Extensive Error Checking
* Ability to save login credentials for ease of access
* Modify Existing Records
* Search from the list of Records
* Auto Creation of necessary files when missing
* Password Field is Protected to prevent password from being seen



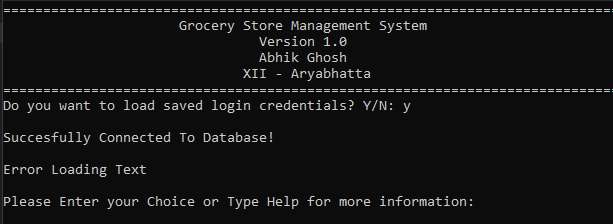
**VALIDATIONS AND CHECKS**

Checks for various forms of invalid input

Database Connection issues



Error Loading Information from files



Database Related Error Management





**HARDWARE USED**

Device Specifications

Dell Inspirion 15 5000

Device name : DESKTOP-OF1F2KB

Processor: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz

Installed RAM: 8.00 GB (7.77 GB usable)

Device ID: A53DF0FE-978E-43BB-BCA0-1201BDD5A2DF

Product ID: 00327-35870-30987-AAOEM

System type: 64-bit operating system, x64-based processor

Pen and touch: No pen or touch input is available for this display

**SOFTWARE AND PROGRAMMING LANGUAGE USED**

Windows Specifications

Edition : Windows 11 Home Single Language

Version : 21H2

Installed on : ‎09-‎10-‎2021

OS build : 22000.348

Experience : Windows Feature Experience Pack 1000.22000.348.0



Python 3.9.5 MYSQL 8.0.29 was used to write this code

**SYSTEM STRUCTURE**

USER CHOICE

START

DISPLAY OUTPUT

END

DO YOU WANT TO CONTINUE ?

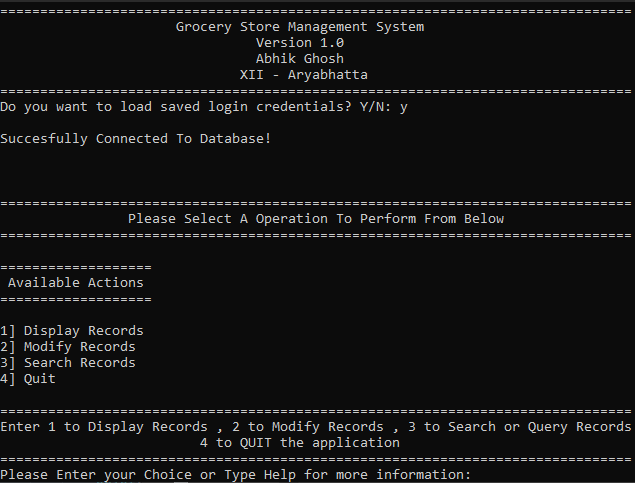
NO

END

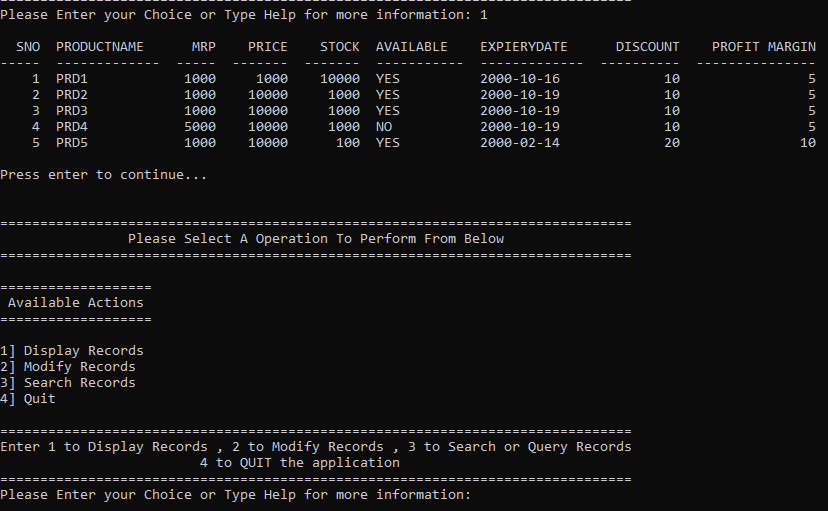
EXECUTE USER CHOICE

YES

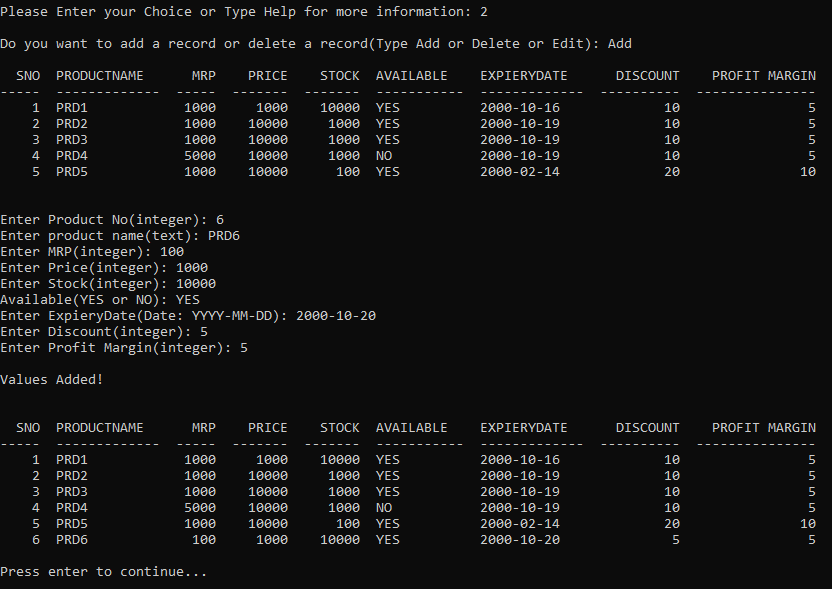
**SCREENSHOTS**



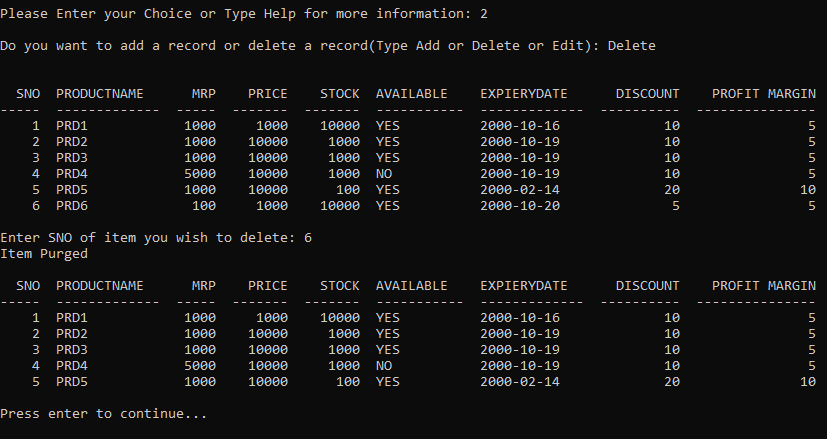
DISPLAY FUCNTION



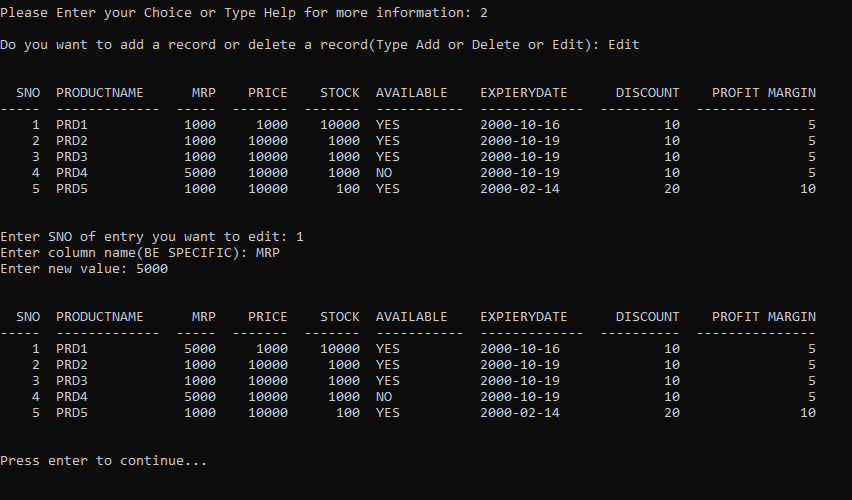
EDIT FUNCTION ADD OPERATION



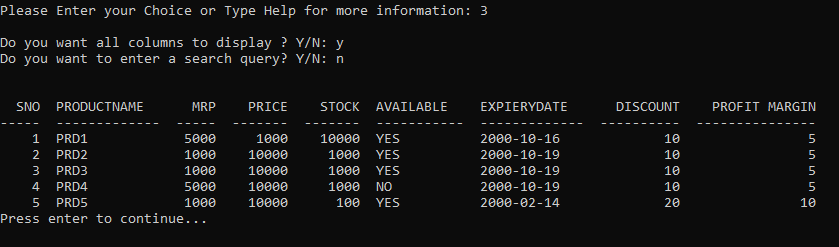
EDIT FUNCTION DELETE OPERATION



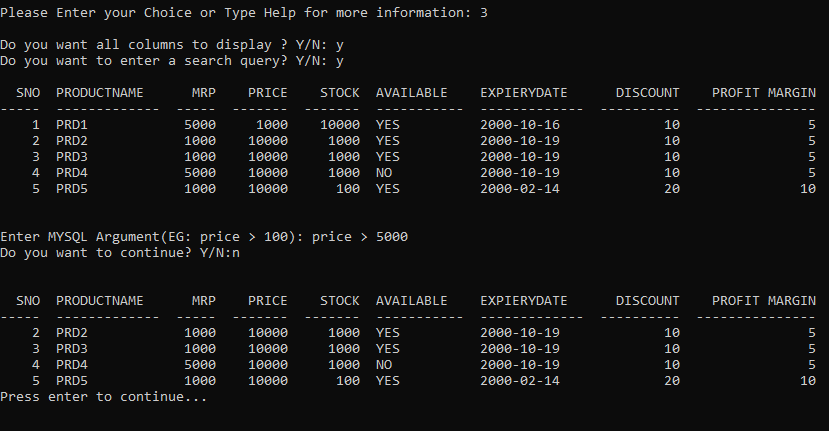
EDIT FUNCTION EDIT OPERATION



SEARCH FUNCTION DISPLAY ALL



SEARCH FUNCTION QUERY



**CODE**

main.py

import mysql.connector as c  
from Utils import \*  
try:  
 import tabulate as t  
except:  
 print('Missing Required Library: Tabulate (pip install tabulate)')  
 input()  
  
isrunning = False  
option = None  
  
  
def app():  
 try:  
 global isrunning  
 global option  
  
 loadtext('WelcomeText.txt', 'p')  
 db = connect(c)  
 if db is not None:  
 cur = db.cursor()  
 else:  
 print("Error Connecting to database Please Restart")  
 return  
 checkdb(db)  
 isrunning = True  
 while isrunning:  
 if option is None:  
 loadtext('choice.txt', 'p')  
 option = 0  
 print()  
 cm = input("Please Enter your Choice or Type Help for more information: ")  
 if cm.lower() == "quit" or cm.lower() == 'q' or cm.lower() == "4":  
 isrunning = False  
 break  
 elif cm.lower() == "1":  
 option = 1  
 elif cm.lower() == "2":  
 option = 2  
 elif cm.lower() == "3":  
 option = 3  
 elif cm.lower() == "5" or cm.lower() == "help" or cm.lower() == "h":  
 loadtext("help.txt", 'p')  
  
 process(option, db)  
 db.commit()  
 input('Press enter to continue...')  
 option = None  
 except:  
 print('Fatal Error Application has crashed!')  
 input()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app()  
 print()  
 input("Press Enter to exit")

Utils.py

import getpass  
from tabulate import tabulate  
  
  
def load\_data():  
 try:  
 with open('parameter.txt', 'r') as f:  
 x = f.readlines()  
 out = []  
 for i in x:  
 out.append(i.replace('\n', ''))  
 return out  
 except:  
 print("Value not found error")  
  
  
def loadtext(file, mode):  
 try:  
 with open(file, 'r') as f:  
 x = f.readlines()  
 if mode.lower() == 'return' or mode.lower() == 'r':  
 return x  
 else:  
 for i in x:  
 print(i, end='')  
 except:  
 if mode.lower() == 'return' or mode.lower() == 'r':  
 return "Error Loading Text"  
 else:  
 print("Error Loading Text")  
  
  
def savedpass(mode, data=None):  
 try:  
 if mode.lower() == 'g':  
 with open('details.txt', 'r') as f:  
 x = f.readlines()  
 if len(x) == 3:  
 return tuple(x)  
 else:  
 return 'error', "error", 'error'  
 elif mode.lower() == 's':  
 with open('details.txt', 'w') as f:  
 f.writelines(data)  
 f.flush()  
  
 except:  
 return 'error', "error", 'error'  
  
  
def connect(connector):  
 global d  
 x = input("Do you want to load saved login credentials? Y/N: ")  
 if x.lower() == 'y':  
 hostname, username, passwd = savedpass('g')  
 else:  
 hostname = input("Enter Database Hostname: ")  
 username = input("Enter Username: ")  
 passwd = getpass.getpass('Enter password:')  
 try:  
 db = connector.connect(  
 host=hostname.replace('\n', ''),  
 user=username.replace('\n', ''),  
 password=passwd.replace('\n', '')  
 )  
  
 print()  
 print("Succesfully Connected To Database!")  
 print()  
  
 if x.lower() != 'y':  
 c = input("Do you want to save your login credentials? Y/N: ")  
 if c.lower() == 'y':  
 savedpass('s', [hostname + '\n', username + '\n', passwd + '\n'])  
 d = db  
 return db  
 except:  
 print()  
 print("Error Connecting to Database!")  
 print()  
  
  
def execute(cursor, command):  
 cursor.execute(command)  
 return cursor.fetchall()  
  
  
def checkdb(db, database\_name=load\_data()[0], table\_name=load\_data()[1]):  
 cur = db.cursor()  
 data = execute(cur, 'show databases')  
 dbs = tuple()  
 exist = False  
 for i in data:  
 for j in i:  
 if j not in dbs:  
 dbs += (j,)  
  
 for i in dbs:  
 if i == database\_name:  
 exist = True  
  
 if not exist:  
 execute(cur, "create database store")  
 execute(cur , 'use store')  
 execute(cur,  
 "create table " + table\_name.lower() + " (SNO integer(255) NOT NULL PRIMARY KEY,PRODUCTNAME varchar(30),MRP integer(255),PRICE integer(255),STOCK integer(255),AVAILABE varchar(4),EXPIARYDATE date,DISCOUNT integer(255),PROFITMARGIN integer(255))")  
 else:  
 execute(cur, 'use store')  
 data = execute(cur, "show tables")  
 tbls = tuple()  
 there = False  
  
 for i in data:  
 for j in i:  
 if j not in tbls:  
 tbls += (j,)  
  
 for i in tbls:  
 if i == table\_name.lower():  
 there = True  
  
 if there:  
 return  
 else:  
 execute(cur, 'use store')  
 execute(cur,  
 "create table " + table\_name + " (SNO integer(255) NOT NULL PRIMARY KEY,PRODUCTNAME varchar(30),MRP integer(255),PRICE integer(255),STOCK integer(255),AVAILABE varchar(4),EXPIARYDATE date,DISCOUNT integer(255),PROFITMARGIN integer(255))")  
  
  
def display(cur):  
 execute(cur, 'use ' + load\_data()[0])  
 x = execute(cur, 'select \* from ' + load\_data()[1])  
 print()  
 print(tabulate(x, ['SNO', 'PRODUCTNAME', 'MRP', 'PRICE', 'STOCK', 'AVAILABLE', 'EXPIERYDATE', 'DISCOUNT',  
 'PROFIT MARGIN']))  
 print()  
  
  
def modify(cur):  
 print()  
 v = input("Do you want to add a record or delete a record(Type Add or Delete or Edit): ")  
 if v.lower() == 'add' or v.lower() == 'a':  
 display(cur)  
 print()  
 SNO = input("Enter Product No(integer): ")  
 PRODUCTNAME = input("Enter product name(text): ")  
 MRP = input("Enter MRP(integer): ")  
 PRICE = input("Enter Price(integer): ")  
 STOCK = input("Enter Stock(integer): ")  
 AV = input("Available(YES or NO): ")  
 EXPIERYDATE = input("Enter ExpieryDate(Date: YYYY-MM-DD): ")  
 DISCOUNT = input("Enter Discount(integer): ")  
 PROFIT = input("Enter Profit Margin(integer): ")  
 try:  
 change\_values(v, cur, (  
 int(SNO), PRODUCTNAME, int(MRP), int(PRICE), int(STOCK), AV, EXPIERYDATE, int(DISCOUNT), int(PROFIT)))  
 except:  
 print()  
 print('Operation Failed Input Data Error Pls check datatype of inputted values!')  
 print()  
 return  
 print()  
 print("Values Added!")  
 print()  
 display(cur)  
 elif v.lower() == 'delete' or v.lower() == 'd':  
 print()  
 display(cur)  
 n = int(input("Enter SNO of item you wish to delete: "))  
 try:  
 change\_values(v, cur, n)  
 except:  
 print("Operation Failed!")  
 print('Item Purged')  
 display(cur)  
 elif v.lower() == 'edit' or v.lower() == 'e':  
 print()  
 display(cur)  
 print()  
 x = load\_data()  
 no = input('Enter SNO of entry you want to edit: ')  
 c = input("Enter column name(BE SPECIFIC): ")  
 v = input("Enter new value: ")  
 try:  
 execute(cur, 'update ' + x[1] + ' set ' + c.upper() + '= "' + v + '" where SNO=' + no)  
 print()  
 display(cur)  
 print()  
 except Exception:  
 print()  
 print('Input Error')  
 print()  
 return  
 else:  
 return  
  
  
def search(cur):  
 print()  
 c = None  
 arg = None  
 ch = input('Do you want all columns to display ? Y/N: ')  
 if ch.lower() == 'n':  
 c = []  
 display(cur)  
 print()  
 while True:  
 \_ = input('Enter Column you want to display Name: ')  
 c.append(\_)  
 e = input('Do you want continue? Y/N:')  
 if e.lower() == 'no' or e.lower() == 'n':  
 break  
 ch1 = input('Do you want to enter a search query? Y/N: ')  
 if ch1.lower() == 'y':  
 arg = ''  
 display(cur)  
 print()  
 while True:  
 \_ = input("Enter MYSQL Argument(EG: price > 100): ")  
 arg += \_  
 e = input('Do you want to continue? Y/N:')  
 if e.lower() == 'no' or e.lower() == 'n':  
 break  
 else:  
 v = input('Do you want a AND Logical Operator or OR Logical Operator-> AND/OR/A/R: ')  
 if v.lower() == 'and' or v.lower() == 'a':  
 arg += ' and '  
 elif v.lower() == 'or' or v.lower() == 'r':  
 arg += ' or '  
 else:  
 break  
 print()  
 find(cur, c, arg)  
  
  
def find(cur, columns=None, condiitons=None):  
 try:  
 d = load\_data()  
 x = None  
 selection = ''  
  
 execute(cur, 'use ' + d[0])  
  
 if columns is not None:  
 for i in columns:  
 if len(selection) == 0:  
 selection += i  
 else:  
 selection += ',' + i  
 if columns is None:  
 if condiitons is None:  
 x = execute(cur, 'select \*' + ' from ' + d[1])  
 else:  
 x = execute(cur, 'select \*' + ' from ' + d[1] + ' where ' + condiitons + ';')  
 else:  
 if condiitons is None:  
 x = execute(cur, 'select ' + selection + ' from ' + d[1])  
 else:  
 x = execute(cur, 'select ' + selection + ' from ' + d[1] + ' where ' + condiitons + ';')  
 print()  
 if columns is not None:  
 print(  
 tabulate(x,  
 columns)  
 )  
 else:  
 print(tabulate(x, ['SNO', 'PRODUCTNAME', 'MRP', 'PRICE', 'STOCK', 'AVAILABLE', 'EXPIERYDATE', 'DISCOUNT',  
 'PROFIT MARGIN']))  
 except:  
 print("Search Error Please Check Your Arguments For SQL Syntax Errors")  
  
  
  
def change\_values(action, cursor, values, db\_name=load\_data()[0], table\_name=load\_data()[1]):  
 if action.lower() == "add" or action.lower() == 'a':  
 if values is None:  
 return  
 else:  
 execute(cursor, 'use ' + db\_name)  
 execute(cursor, 'insert into ' + table\_name + ' values' + str(values))  
 elif action.lower() == 'delete' or action.lower() == 'd':  
 if values is None:  
 return  
 else:  
 execute(cursor, 'use ' + db\_name)  
 execute(cursor, 'delete from ' + table\_name + ' where SNO=' + str(values))  
  
  
def process(option, db):  
 if option == 1:  
 display(db.cursor())  
 db.commit()  
 elif option == 2:  
 modify(db.cursor())  
 db.commit()  
 elif option == 3:  
 search(db.cursor())  
 db.commit()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 input("This file cannot be run on its own Execute main.py!")

**ANNEXURE**

* Import mysql.connector : adds mysql functionaly to program.
* Import tabulate : imports required library TABULATE.
* Import getpass: enables pythons built-in password utils library.

CLASS MAIN.PY

* app() : main program logic all codes executes from within this function.

CLASS UTILS.PY

* load\_data() : function to read data from storage files.
* load\_text(file , mode) : function to read text from file.
* savedpass(mode , data=None) : function to load password from file.
* connect() : function to connect to MYSQL database.
* execute() : Helper function to run SQL commands.
* checkdb(db,database\_name=load\_data()[0],table\_name=load\_data()[1]) : Function to check for missing files.
* display(cur) : function to display text in a cursor.
* modify(cur) : function to modify table.
* search(cur) : function to search/Query database.
* find(cur) : Helper function for search().
* change\_values(action,cursor,values,db\_name=load\_data()[0],table\_name=load\_data()[1]) : function to modiy the table.
* process(option, db) : loader function which selects options.

The if \_\_name\_\_ == “\_\_main\_\_” statement ensures that the program run only when the main class is directly executed to allow to import the class in a project without issue.