A red and yellow logo

Description automatically generated**SUPERVISION CERTIFICATE**

CHECKOUT SYSTEM

SUBMITTED BY:

YUVRAJ SINGH

ISHAAN SINGH BISHT

SUBMITTED TO

Mr. Iqbal Singh

2024

A PROJECT REPORT SUBMITTED TO CENTRAL BOARD OF SECONDARY EDUCATION FOR THE PARTIAL FULFILLMENT OF INFORMATICS PRACTICES PROJECT WORK.

DEPARTMENT OF COMPUTER SCIENCE

ARMY PUBLIC SCHOOL, AMRITSAR

PUNJAB-143107

(2024-2025)

This Is to Certify That Work Presented in The Project Entitled "Checkout System" Submitted for The Partial Fulfilment of CBSE INFORMATICS PRACTICES Exam To The CBSE New Delhi Is A Bona fide Project Work Carried Out By YUVRAJ and ISHAAN Students of Class 12, Army Public School, Amritsar, Punjab Under My Guidance And Supervision.

MR. IQBAL SINGH

PGT COMPUTER SCIENCE

ARMY PUBLIC SCHOOL AMRITSAR

**ACKNOWLEDGEMENT**

I Take This Opportunity to Express My Deep Sense of Gratitude to Mr. Iqbal Singh, PGT (Computer Science), Army Public School, Amritsar for His Guidance, Advice and Support from The Beginning Of My Project Work To The Completion Of The Same. His Keen Interest in MY Project Work, Devotion of Time, And Inspiration During This Project Tenure Has Made This Work to Come in This Form.

I Also Pay Gratitude Towards Our Principal Mrs. Rajdeep Jain, Army Public School Amritsar Permitting Me to Avail Necessary Facilities of The Lab as Well as School for My Work. The Support Received from Friends During My Project Work Cannot Be Acknowledged in Words.

I Express My Deepest Affection, Appreciation and Thank to My Parents, Who Encouraged and Supported me In This Project. All Of Them Have Made Immense Contribution to My Studies and Life by Their Support, Guidance, Understanding and Sacrifices.

…………………………………

Army Public School Amritsar

***PROJECT SYNOPSIS***

**PROJECT DEFINITION**

To Design a project to allow user to create bills and manage inventory in a business.

**TEAM MEMBERS:**

* Ishaan Singh Bisht
* Yuvraj Singh

**REASON FOR CHOOSING THE TOPIC:**

A billing system automates the creation of bills, reducing time spent manually creating them by hand. Thus, ensures that bills are generated quickly and accurately in shop-like scenario.

**OBJECTIVE:**

* To let user create and edit an inventory of items, their prices, stock and discounts.
* To graphically show the sales data of a business.
* To create bills for customer and retrieve the latest one made.
* To automate totalling in bills.
* To automate changes in inventory after each transaction.

**HARDWARE REQUIREMENT:**

* A computer/laptop
* Operating System – Windows 10 or above
* 4 GB RAM
* 512 MB of free disk space

**LIMITATIONS:**

* It is not a web-based project
* UI can be more responsive
* More functionality can be added as per requirements

**REFERENCES:**

* Online Python Documentation
* Textbook – Class XI and XII – INFORMATICS PRACTICES as per NCERT
* Google.com

**TreeMap of Project:**

.

|-- main.py

`-- data/

|-- db.csv

`-- lastbill.csv

**main.py**

**#Help text and imports**

a = """

Commands:

viewitems

add <icode> <cat> <name> <istock> <iprice> <discount>

remove <icode>

stock <icode> <change>

edit <icode> <field> <value>

newbill

viewsales

viewlastbill

savedb

exit

bill commands:

cancel

addedit <icode> <qty>

remove <icode> <qty>

use \"-\" instead of spaces in between of entries

"""

import matplotlib.pyplot as plt

import numpy as np

import datetime

import pandas as pd

import os

fields = ['icode','category','name','istock','iprice',"idiscount","isold"]

store = pd.read\_csv("./data/db.csv", header=None, index\_col=0,

names=['icode','category','name','istock','iprice',"idiscount","isold","netsales"], dtype={"istock":int,"iprice":float,"idiscount":float,"isold":float,"netsales":float})

last\_bill = pd.read\_csv("./data/lastbill.csv",header=None,index\_col=0,

names=['name','qty','price',"discount","amt"])

def parse\_bill(inp: str):

    y = inp.strip().split(" ")

    y[1] = y[1].upper()

    if y[0].upper() == "ADDEDIT":

        try:

            aoe\_bill(y[1],int(y[2]))

        except  IndexError:

            print("Invalid input")

    elif y[0].upper() == "REMOVE":

        try:

            rem\_bill\_item(y[1])

        except AttributeError:

            print("Item not in bill")

    else:

        print("Invalid command")

def parse\_menu(inp: str):

    x = inp.strip().split(" ")

    try:

        x[1] = x[1].upper()

    except:

        pass

    if x[0].upper() == "EXPORT":

        export\_data(store)

    elif x[0].upper() == "ADD":

        add\_item(x[1],x[2].replace("-"," "),x[3].replace("-"," "),

int(x[4]),float(x[5]),float(x[6]),int(x[7]),int(x[8]))

        print("New Item added!")

    elif x[0].upper() == "REMOVE":

        rem\_item(x[1])

        print("Item removed!")

    elif x[0].upper() == "STOCK":

        stock(x[1],int(x[2]))

        print("Stock updated!")

    elif x[0].upper() == "EDIT":

        edit(x[1],x[2],x[3].replace("-"," "))

        print("Data edited!")

    elif x[0].upper() == "HELP":

        print(a)

    elif x[0].upper() == "VIEWITEMS":

        print(store)

    elif x[0].upper() == "NEWBILL":

        makebill()

    elif x[0].upper() == "CLS":

        os.system("cls")

    elif x[0].upper() == "EXIT":

        z = input("Do you want to save changes? (y/n): ")

        if z == "y":

            export\_data(store)

            exit()

        else:

            exit()

    elif x[0].upper() == "SAVEDB":

        export\_data(store)

    elif x[0].upper() == "VIEWSALES":

        viewsales()

    elif x[0].upper() == "VIEWLASTBILL":

        print(last\_bill)

    else:

        print("no such command as", x[0], "try \"help\"")

#main function

def main():

    global store

    print("Welcome to CheckOutApp")

    print("Type \"help\" to get to list of all commands")

    while True:

        inx = input("> ")

        try:

            parse\_menu(inx)

        except IndexError:

             print("Missing Arguments, use \"help\"")

#database commands

def stock(icode: str,change:int):

    if store.loc[icode,"istock"]+change >= 0:

        store.loc[icode,'istock'] += change

    else:

        print("Must have atleast 0 stock")

def edit(icode: int, field: str ,value: any):

    store.loc[icode,field]=value

def add\_item(icode: str,cat: str,name: str,istock: int,iprice:float,

discount: float,isold: int, netsales: float):

store.loc[icode]=[cat,name,istock,iprice,discount,isold, netsales]

def rem\_item(icode: str):

    del store.loc[icode]

def export\_data(df:pd.DataFrame):

    df.to\_csv("./data/db.csv",header=False, index\_label="icode")

#bill commands

def aoe\_bill(icode: str,qty: int): # add if not present/ edit if present

    global bill

    if store.loc[icode,"istock"] >= qty:

        bill.loc[icode] = [store.loc[icode,'name'],

qty,store.loc[icode,'iprice'],

store.loc[icode,'idiscount'],0]

        bill.loc[icode,'amt'] = store.loc[icode,'iprice']\*qty\*

(100-store.loc[icode,'idiscount'])/100

    else:

        print("\n")

        print("Not enough stock")

    print("\n")

    print(bill)

    print('\n')

def rem\_bill\_item(icode): # delete item from bill

    try:

        del bill.loc[icode]

    except:

        print("Item not in bill")

    print("\n")

    print(bill)

    print('\n')

def finalize\_bill(): # print bill and save a copy in lastbill.csv

    global now

    print("Bill finalized")

    now = str(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"))

    print("Timestamp: ",now)

    print("Customer Name: ", cname)

    print("Total Amount: ",bill['amt'].sum())

    print(bill)

    for i in bill.index:

        store.loc[i,"istock"] -= bill.loc[i,'qty']

        store.loc[i,"isold"] += bill.loc[i,'qty']

        store.loc[i,"netsales"] += bill.loc[i,'amt']

    bill.to\_csv('./data/lastbill.csv',index\_label=cname+" "+now,header=False)

def makebill(): # billing command line

    global bill

    bill = pd.DataFrame(columns=['name','qty','price',"discount","amt"])

    global cname

    print(store)

    print("Ready for bill generation")

    cname = input("Enter customer name: ")

    while True:

        iny = input(">bill>")

        if iny.upper() == "CANCEL":

            break

        elif iny.upper() == "FINALIZE":

            finalize\_bill()

            break

        parse\_bill(iny)

#sales graph

def viewsales():

    plt.figure(figsize=(10,6))

    plt.subplot(1, 2, 1)

    plt.bar(store.index,store['isold'])

    plt.title("Units sold")

    plt.subplot(1, 2, 2)

    plt.barh(store.index,store['netsales']//1000)

    plt.title("Sales in thousands")

    plt.tight\_layout()

    plt.show()

main()

**db.csv**

FUR1798,Furniture,Bush Somerset Collection Bookcase,17,20999.0,0.0,5.0,104995.0

OFF2365,Office Supplies,Xerox 1967,3,1199.0,0.2,0.0,0.0

EDI1949,Edibles,Apple,50,30.0,0.2,10.0,299.4

TEC1950,Technology,Cisco SPA 501G IP Phone,18,16999.0,0.2,10.0,169650.02

OFF1234,Office Supplies,Paper A4 500 bundle,70,400.0,0.0,5.0,2000.0

OFF1235,Office Supplies,Paper A4 1000 bundle,49,700.0,5.0,1.0,665.0

**lastbill.csv**

FUR1798,Bush Somerset Collection Bookcase,5,20999.0,0.0,104995.0

EDI1949,Apple,10,30.0,0.2,299.4

OFF1234,Paper A4 500 bundle,5,400.0,0.0,2000.0

OFF1235,Paper A4 1000 bundle,1,700.0,5.0,665.0

TEC1950,Cisco SPA 501G IP Phone,10,16999.0,0.2,169650.02

****

“viewitems” command

“help” command

**A screenshot of a computer

Description automatically generated**

“add” command

**A screen shot of a computer

Description automatically generated**

“remove” command

**A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated**

“edit” command

**A screenshot of a computer

Description automatically generated**

“addedit” command

“newbill” command

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a graph

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

“finalize” command

“remove” command

“viewsales” command command