GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

MEDICAL STORE MANAGEMENT SYSTEM

A Project Report of Capstone Project 1

Submitted by

AASHI JAIN

(1713104065/17SCSE104067)

in partial fulfillment for the award of the degree

of

Bachelor of Computer Applications

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Ms. J. ANGELIN BLESSY,M.C.A.

Assistant Professor

APRIL / MAY- 2020

GALGOTIAS UNIVERSITY

Certified that this project report **"MEDICAL STORE MANAGEMENT SYSTEM"** is the bonafide work of "**AASHI JAIN (1713104065)"** who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL, Ph(Management), PhD (CS)

Professor & Dean,

School of Computing Science & Engineering

SIGNATURE OF SUPERVISOR

Ms. J. ANGELIN BLESSY,M.C.A.

Assistant Professor

School of Computing Science & Engineering

ABSTRACT

The purpose of Medical Store Management System is to automate the existing manual system by the help of computerized equipments and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same. The required software and hardware are easily available and easy to work with.

Medical Store Management System, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. Thus it will help organization in better utilization of resources. The organization can maintain computerized records without redundant entries. That means that one need not be distracted by information that is not relevant, while being able to reach the information

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

List of abbreviation:

MSMS: Medical Store Management system

DBMS: Database Management System

MS Office: Microsoft Office

CHAPTER 1: INTRODUCTION

1 Introduction

The following subsections of the Software Requirements Specifications (SRS) document provide an overview of the entire SRS.

Purpose

The Software Requirements Specification (SRS) will provide a detailed description of the requirements for the Medical Store Management System (MSMS). This SRS will allow for a complete understanding of what is to be expected of the MSMS to be constructed. The clear understanding of the MSMS and its' functionality will allow for the correct software to be developed for the end user and will be used for the development of the future stages of the project. This SRS will provide the foundation for the project. From this SRS, the MSMS can be designed, constructed, and finally tested.

This SRS will be used by the software engineers constructing the MSMS and the medical store end users. The software engineers will use the SRS to fully understand the expectations of this MSMS to construct the appropriate software. The medical store end users will be able to use this SRS as a "test" to see if the software engineers will be constructing the system to their expectations. If it is not to their expectations the end users can specify how it is not to their liking and the software engineers will change the SRS to fit the end users' needs.

Scope

The software product to be produced is a Medical Store Management System which will automate the major medical store operations. The first subsystem is to keep record of the stock of the medicines available and expiry date of the medicines. The second subsystem is for billing and keep record of the customer. The third subsystem is keep record of the sale and the supplier and keep record of the profits and losses . These three subsystems' functionality will be described in detail in section 2-Overall Description.

There are two end users for the MSMS. The end users are the chemists and the store manager.

It is the complete medical shop management software is so designed as to ease the

work load of medical shop professionals. The main feature includes invoicing, inventory and stock control, accounting, client and vendor management.

This software helps you to track all the profits, loss, profitable clients and products of medical shop moreover it's a medical shop accounting software. Flexible and adaptive software suited to medical shops or stores or pharmacies of any size.

.

Definitions, Acronyms, and Abbreviations.

SRS – Software Requirements
Specification MSMS – Medical Store
Management System
Subjective satisfaction – The overall satisfaction of the system

End users – The people who will be actually using the system

Overview

The SRS is organized into two main sections. The first is The Overall Description
and the second is the Specific Requirements. The Overall Description will describe the requirements of the MSMS from a general high level perspective. The Specific Requirements section will describe in detail the requirements of the system.

CHAPTER 2:THE OVERALL DESCRIPTION

The Overall Description

Describes the general factors that affect the product and its requirements. This section does not state specific requirements. Instead it provides a background for those requirements, which are defined in section 3, and makes them easier to understand.

Product Perspective

The MSMS is an independent stand–alone system. It is totally self contained.

Hardware Interfaces

The MSMS will be placed on PC's throughout the medical store.

Software Interfaces

All databases for the MSMS will be configured using MS Access. These databases include medicine stocks ,customers information and their billing. These can be modified by the end users. The medicine stock database will include the number of stock of medicines available and keep track of their expiry dates. The customers information database will contain all the information of the customer such as first name, last name,phone number,id, number of

medicines purchased, details of medicine purchased, total amount, mode of payment(cash/debit or credit card) .

Product Functions

Medicine Stock Record Management System

- Allows to keep record of the medicine such as medicine name, medicine id, quantity of medicine, price of the medicine etc.
- Allow to keep record of the expiry date of the medicines such as to remove those medicines from the stock whose expiry date is over.
- Allows to manage the availability of the medicine in the store such as to order those stock of medicines which are over on time.
- Allows the user to arrange the medicine systematically according to their comfort.

Customer information and Billing System

- Keeps track of the customer information.
- Billing is done with their proper record.

Sale and Supplier Management System

- Keep in record the details of their supplier such as supplier name, id, quantity of stock of medicines supplied, bill of the stock supplied etc.
- Keep track of the profit and losses of the shop.
- Keep record of the sales of shop.
- Complete record of all the purchases is maintained.

User Characteristics

Educational level of MSMS computer software –
Low Experience of MSMS software – None
Technical Expertise – Little

Apportioning of Requirements

The audio and visual alerts will be deferred because of low importance at this time.

Assumptions and Dependencies

- The system is not required to save generated reports.
- Credit card payments are not included

CHAPTER 3: SPECIFIC REQUIREMENTS

2 Specific Requirements

This section contains all the software requirements at a level of detail, that when combined with the system context diagram, use cases, and use case descriptions, is sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements.

External Interfaces

The Hotel Management System will use the standard input/output devices for a personal computer. This includes the following:

- Keyboard
- Mouse
- Monitor
- Printer

User Interfaces

The User Interface Screens are described in table 1.

Table 1: Medical Store Management User Interface Screens

Screen Name	Description
Login	Log into the system as a pharmacist or Manager
Medicine stock	Retrieve button, update/save new medicine stocks, remove expired stock, availability of stock.
Billing	Record all the medicine purchases and generate bill.
Payment	Payment details should be stored.
Supplier Info	Complete details of the supplier is maintained and updated/ saved with time.
Sale	Complete record of sale is maintained with the profit or losses and updated/ saved with time to time.

Software Interfaces

The system shall interface with Python.

Hardware Interfaces

The system shall run on Python based system.

Communication Interfaces

The system shall be a standalone product that does not require any communication interfaces.

Functional Requirements

Functional requirements define the fundamental actions that system must perform.
The functional requirements for the system are divided into three main categories,Medicine stock, Customer information and billing and Sale and supplier info. For further details, refer to the use cases.

1. Medicine Stock

   The system shall record stock of medicines.
   The system shall be updated with arrival of new stock.
   The system shall notify the expired stock of medicines.
   The system shall keep record of medicine details,

2. Customer info and billing

   The system will display the customer info.
   The system will generate the bill.
   The system shall store the customer information.
   The system shall keep record of the billing.

3. Sale and supplier info
   The system shall display the supplier information and update it from time to time.
   The system shall display the number of sale with record of profit and losses.

Nonfunctional Requirements

Functional requirements define the needs in terms of performance, logical database requirements, design constraints, standards compliance, reliability, availability, security, maintainability, and portability.

Performance Requirements

Performance requirements define acceptable response times for system functionality.

- The load time for user interface screens shall take no longer than ten seconds.
- The log in information shall be verified within five seconds.
- Queries shall return results within five seconds.

Logical Database Requirements

The logical database requirements include the retention of the following data elements. This list is not a complete list and is designed as a starting point for development.

Design Constraints

The Medical Store Management System shall be a stand-alone system running in a Windows environment. The system shall be developed using c/c++ language.

Standards Compliance

There shall be consistency in variable names within the system. The graphical user interface shall have a consistent look and feel.

Reliability

Specify the factors required to establish the required reliability of the software system at time of delivery.

Availability

The system shall be available during normal hotel operating hours.

Security

Pharmacist and Managers will be able to log in to the Medical Store Management System. Pharmacist will have access to the Medcine stock and customer info and billing system.

Managers will have access to the Sale and Supplier info as well as the Medcine stock and customer info and billing system. Access to the various subsystems will be protected by a user log in screen that requires a user name and password.

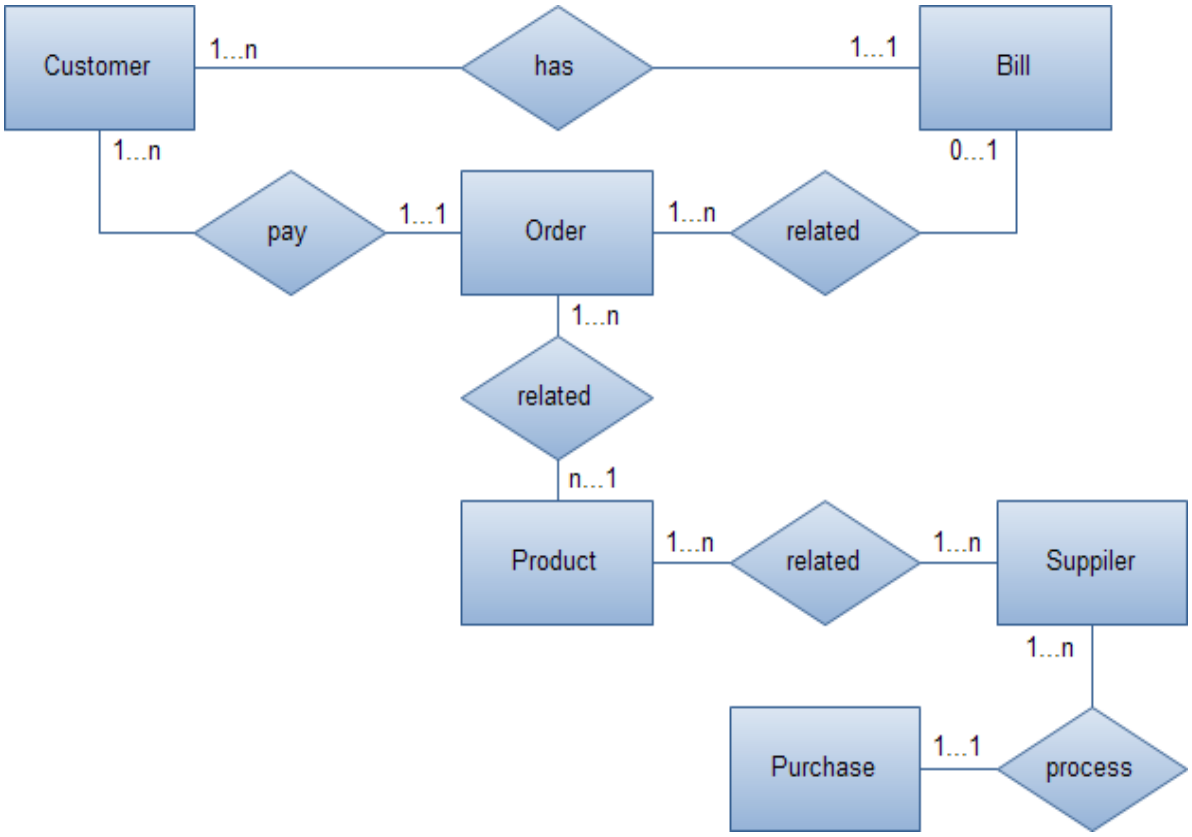
Maintainability

The Medical Store Management System is being developed in Turbo c . It shall be easy to maintain.

Portability

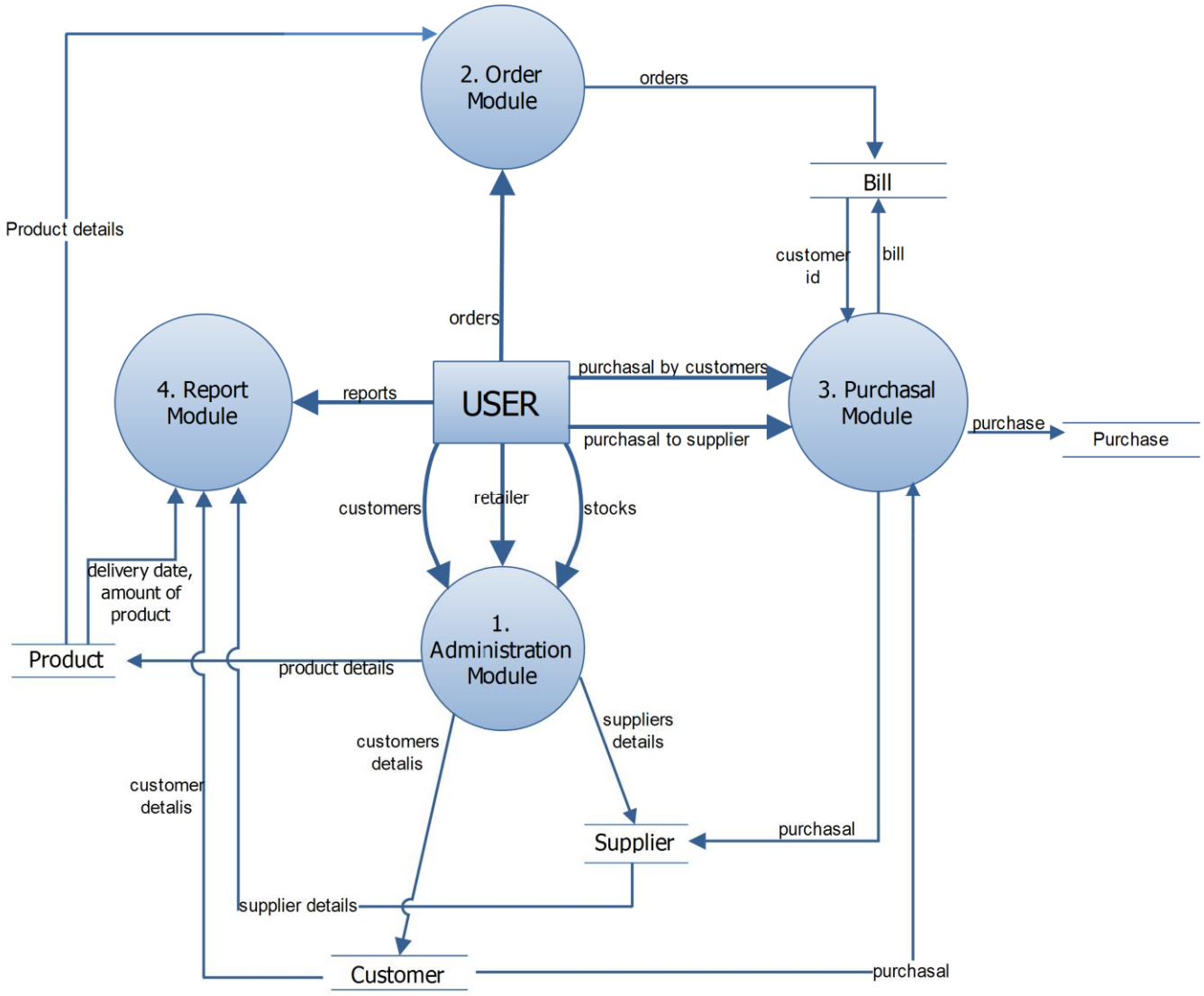
The Medical Store Management System shall run in any Microsoft Windows environment that contains Microsoft Access database.

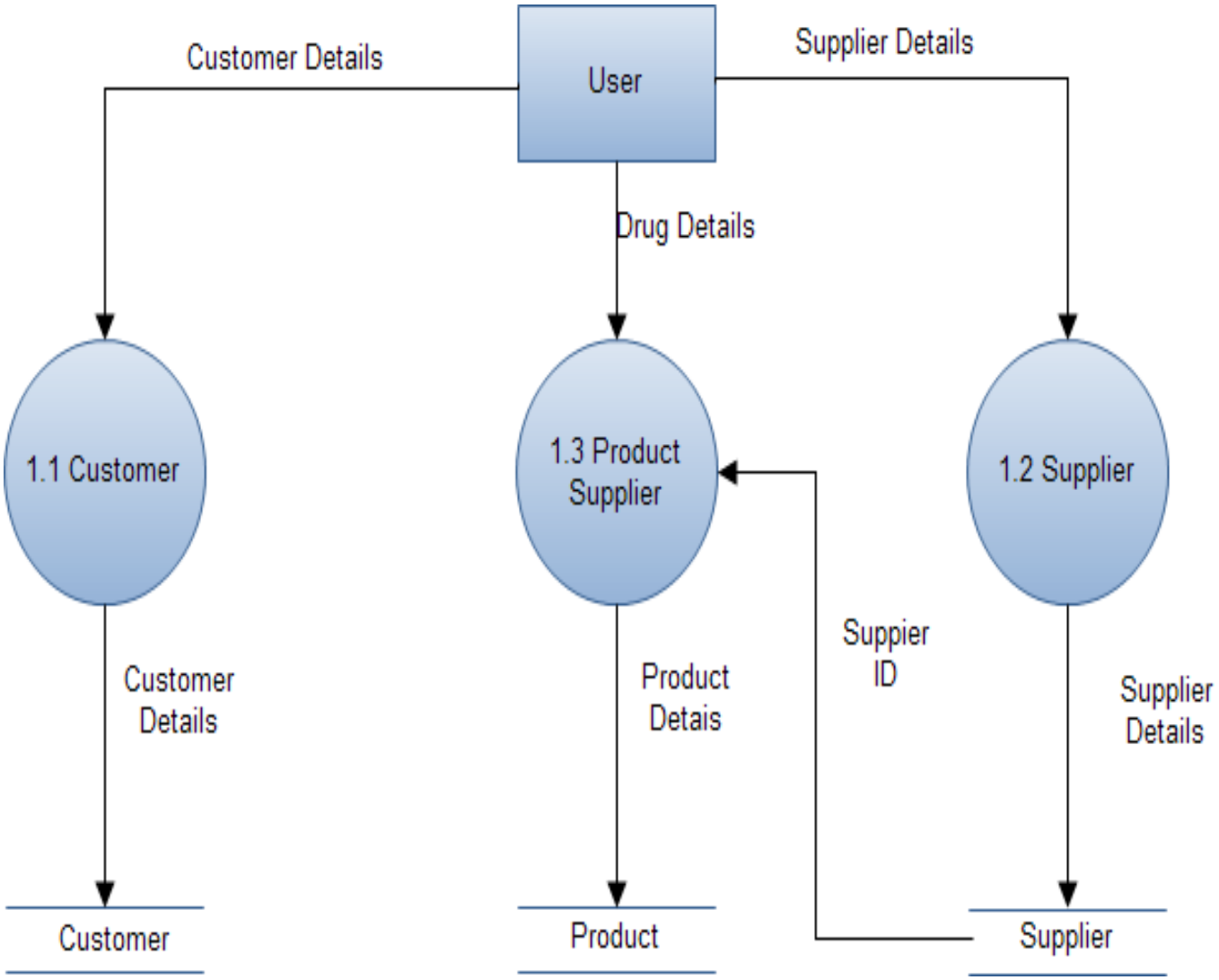
ENTITY-RELATIONSHIP DIAGRAM



DATA FLOW DIAGRAMS

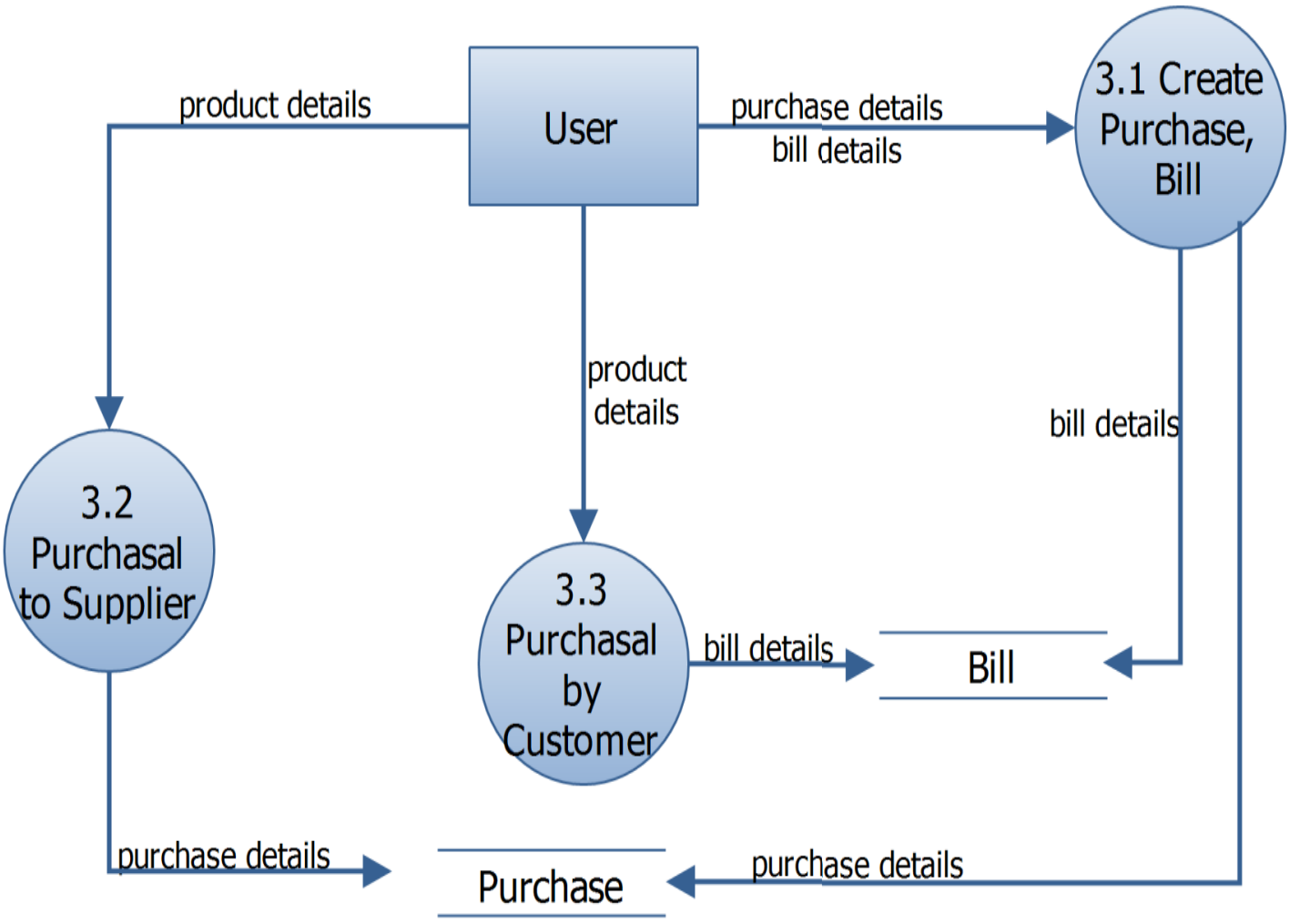
LEVEL 1:





DFD LEVEL 2-ADMINISTRATION

DFD LEVEL2-PURCHASE



DATA DICTIONARY

AIM:- Data dictionary for Medical Store Management System.

- **Customer Information** :- Customer_Id + Customer Name + Email Id + Address_Id.
    - Customer_Id – INT
    - Customer Name – VARCHAR
    - Email Id – VARCHAR
    - Address_Id - VARCHAR

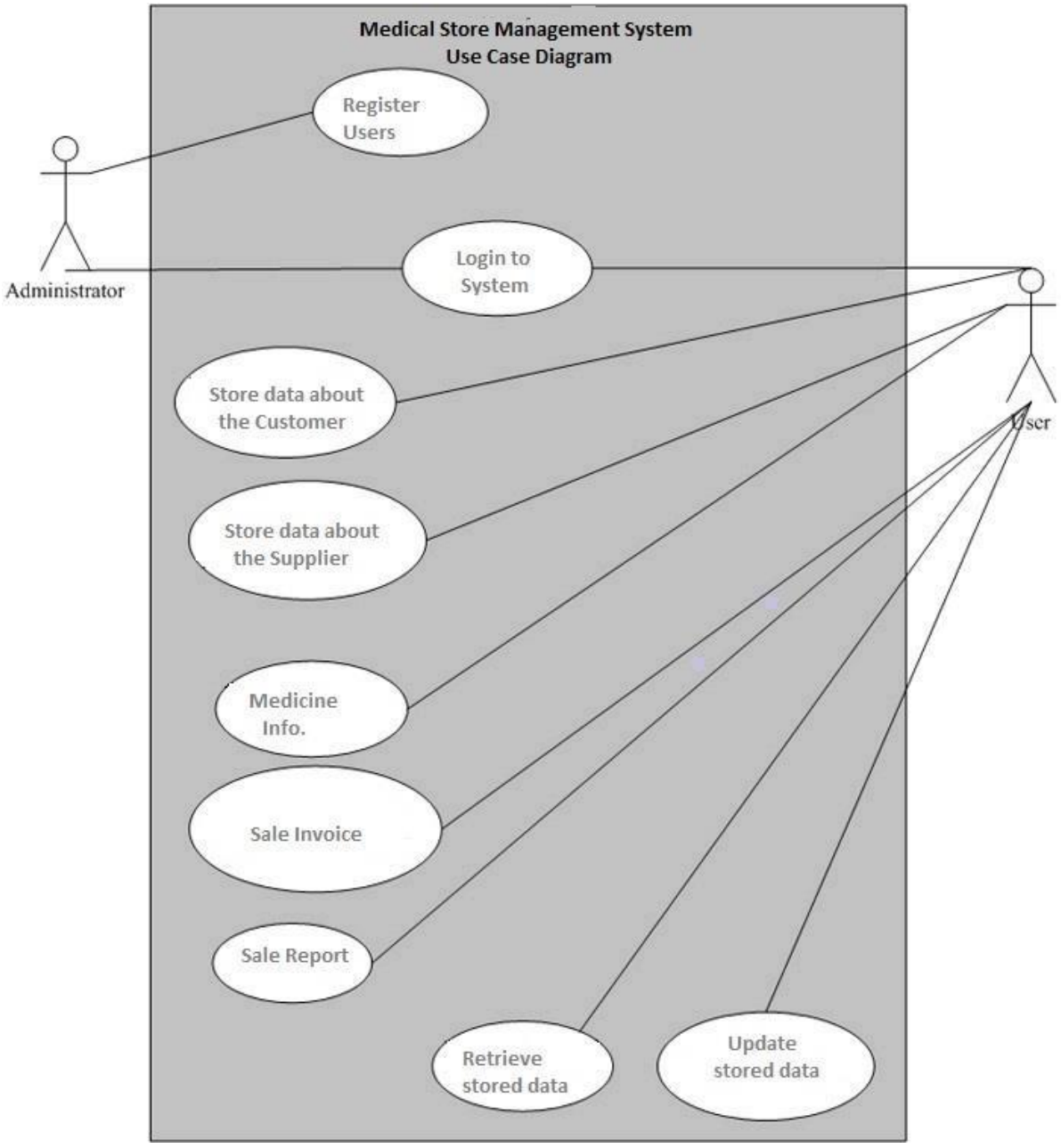
- **Supplier Information:-** Supplier_Id + Supplier Name + Email Id + Address_Id.
    - Supplier_Id – INT
    - Supplier Name – VARCHAR
    - Email Id – VARCHAR
    - Address_Id - VARCHAR

- **Medicine Information:-** Product Code + Product Name + Expiry Date + Price.
    - Product Code – INT
    - Product Name – VARCHAR
    - Expiry Date – DATE
    - Price – INT

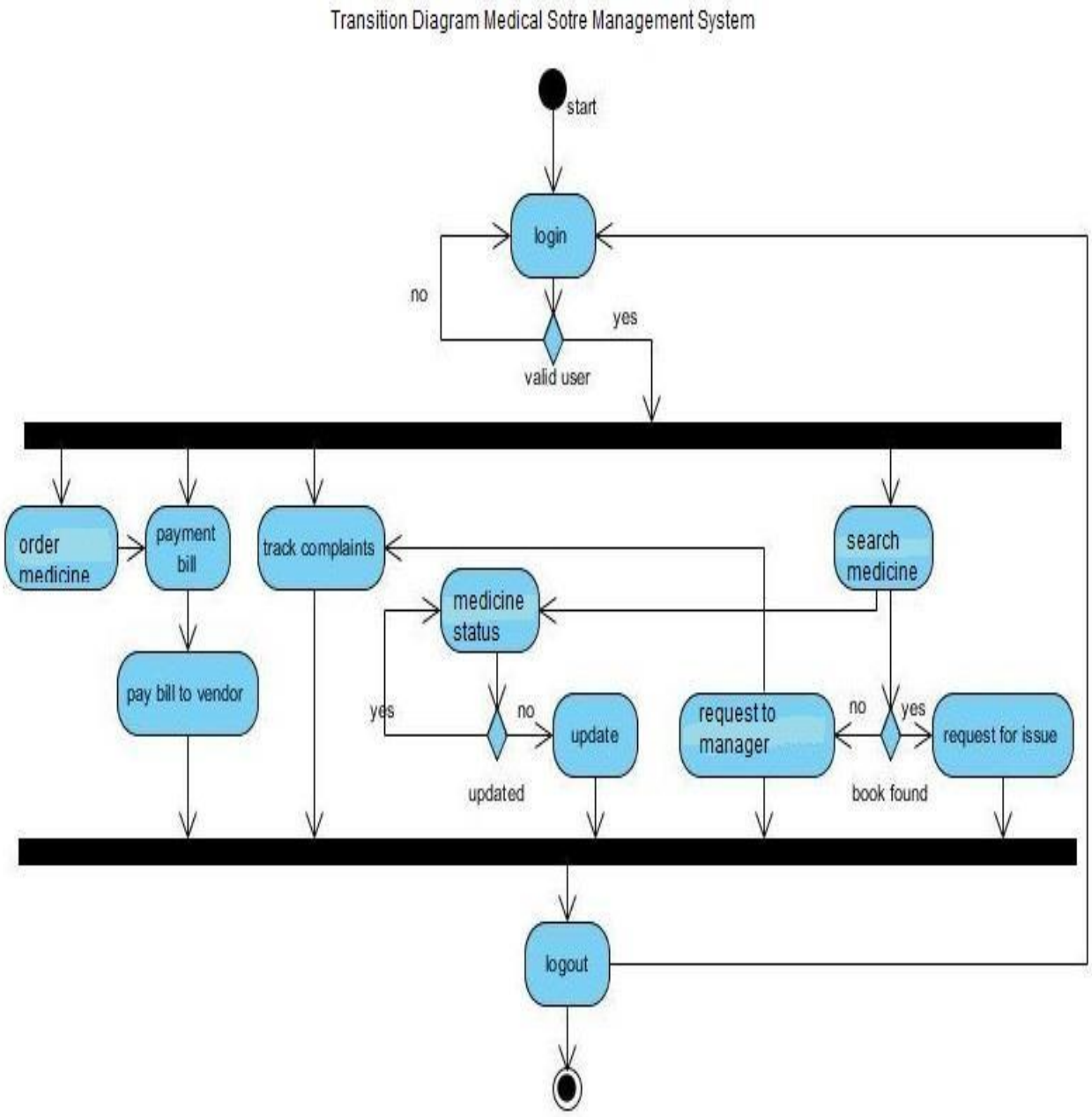
Sale Invoice:- Product Code + Product Name + Quantity + Price + Discount(%) + Total.

- Product Code – INT
- Quantity – INT
- Price – INT
- Total – INT

USE CASE DIAGRAM



STATE VTRANSITION DIAGRAM



CODING

```
import supplier_functions
#Medicine Menu
def mclickedbtn1():
    menu_functions.medicine_menu()
def mclickedbtn2():
    menu_functions.customer_functions()
def mclickedbtn3():
    menu_functions.supplier_functions()
def mclickedbtn4():
    menu_functions.report_functions()
def mclickedbtn5():
    menu_functions.invoicing_functions()
def mclickedbtn6():
    medicine_menu.add_medicine()
def clickedbtn1():
    medicine_menu_window = Tk()
    medicine_menu_window.configure(width=1500,height=600)
    medicine_menu_window.title("Medical Store Management System")
    lbl = Label(medicine_menu_window, text="Medicine
Menu!",bg="RED",fg="white",font=("Times", 30))
    lbl.grid(column=0, row=0)
    lbl2 = Label(medicine_menu_window, text="What would you like to do!")
    lbl2.grid(column=0, row=1)
```

```
    btn1 = Button(medicine_menu_window, text="Add New Medicine",fg="white",bg="SKY BLUE",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn1)

    btn1.grid(column=0, row=2)

    btn2 = Button(medicine_menu_window, text="Search Medicine",fg="white",bg="SKY BLUE",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn2)

    btn2.grid(column=0, row=3)

    btn3 = Button(medicine_menu_window, text="Update Medicine",fg="white",bg="SKY BLUE",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn3)

    btn3.grid(column=0, row=4)

    btn4 = Button(medicine_menu_window, text="Medicines to be purchased",fg="white",bg="SKY BLUE",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn4)

    btn4.grid(column=0, row=5)

    btn4 = Button(medicine_menu_window, text="Return to main menu",fg="white",bg="SKY BLUE",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn4)

    btn4.grid(column=0, row=6)

    medicine_menu_window.mainloop()

#Customer Menu

def clickedbtn2():

    c_menu_window = Tk()

    c_menu_window.configure(width=1500,height=600)

    c_menu_window.title("Medical Store Management Software")

    lbl = Label(c_menu_window, text="Customer Menu!",bg="black",fg="white",font=("Times", 30))

    lbl.grid(column=0, row=0)

    lbl2 = Label(c_menu_window, text="What would you like to do!")

    lbl2.grid(column=0, row=1)
```

```
    btn1 = Button(c_menu_window, text="Search
Customer",fg="white",bg="black",relief="ridge", font=("Times", 12),width=25,
command=mclickedbtn1)

    btn1.grid(column=0, row=2)

    btn2 = Button(c_menu_window, text="New
Customer",fg="white",bg="black",relief="ridge", font=("Times", 12),width=25,
command=mclickedbtn2)

    btn2.grid(column=0, row=3)

    btn3 = Button(c_menu_window, text="Update Customer
Info",fg="white",bg="black",relief="ridge", font=("Times", 12),width=25,
command=mclickedbtn3)

    btn3.grid(column=0, row=4)

    btn4 = Button(c_menu_window, text="Return to main
menu",fg="white",bg="black",relief="ridge", font=("Times", 12),width=25,
command=mclickedbtn4)

    btn4.grid(column=0, row=5)

    c_menu_window.mainloop()

#Supplier Menu

def clickedbtn3():

    s_menu_window = Tk()

    s_menu_window.configure(width=1500,height=600)

    s_menu_window.title("Medical Store Management Software")

    lbl = Label(s_menu_window, text="Supplier Menu!",bg="black",fg="white",font=("Times",
30))

    lbl.grid(column=0, row=0)

    lbl2 = Label(s_menu_window, text="What would you like to do!")

    lbl2.grid(column=0, row=1)
```

```
    btn1 = Button(s_menu_window, text="Search
Supplier",fg="white",bg="black",relief="ridge", font=("Times", 12),width=25,
command=mclickedbtn1)

    btn1.grid(column=0, row=2)

    btn2 = Button(s_menu_window, text="New Supplier",fg="white",bg="black",relief="ridge",
font=("Times", 12),width=25, command=mclickedbtn2)

    btn2.grid(column=0, row=3)

    btn3 = Button(s_menu_window, text="Update Supplier
Info",fg="white",bg="black",relief="ridge", font=("Times", 12),width=25,
command=mclickedbtn3)

    btn3.grid(column=0, row=4)

    btn4 = Button(s_menu_window, text="Return to main
menu",fg="white",bg="black",relief="ridge", font=("Times", 12),width=25,
command=mclickedbtn4)

    btn4.grid(column=0, row=5)

    s_menu_window.mainloop()

#Report Menu

def clickedbtn4():

    r_menu_window = Tk()

    r_menu_window.configure(width=1500,height=600)

    r_menu_window.title("Medical Store Management Software")

    lbl = Label(r_menu_window, text="Report Menu!",bg="black",fg="white",font=("Times",
30))

    lbl.grid(column=0, row=0)

    lbl2 = Label(r_menu_window, text="What would you like to do!")

    lbl2.grid(column=0, row=1)

    btn1 = Button(r_menu_window, text="Day Sales",bg="sky blue",fg="black",relief="ridge",
font=("Times", 12),width=25, command=mclickedbtn1)
```

```
    btn1.grid(column=0, row=2)

    btn2 = Button(r_menu_window, text="Month Sales",bg="sky
blue",fg="black",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn2)

    btn2.grid(column=0, row=3)

    btn3 = Button(r_menu_window, text="Day Purchase",bg="sky
blue",fg="black",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn3)

    btn3.grid(column=0, row=4)

    btn3 = Button(r_menu_window, text="Month Purchase",bg="sky
blue",fg="black",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn3)

    btn3.grid(column=0, row=5)

    btn3 = Button(r_menu_window, text="Profit Report",bg="sky
blue",fg="black",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn3)

    btn3.grid(column=0, row=6)

    btn4 = Button(r_menu_window, text="Return to main menu",bg="sky
blue",fg="black",relief="ridge", font=("Times", 12),width=25, command=mclickedbtn4)

    btn4.grid(column=0, row=7)

    r_menu_window.mainloop()

#Invoicing Menu

def clickedbtn5():

    r_menu_window = Tk()

    r_menu_window.geometry('1500x600')

    r_menu_window.title("Medical Store Management Software")

    lbl = Label(r_menu_window, text="Invoice Menu!",bg="black",fg="white",font=("Times",
30))

    lbl.grid(column=0, row=0)

    lbl2 = Label(r_menu_window, text="What would you like to
do!",bg="red",relief="ridge",fg="white",font=("Times", 20),width=25)
```

```
    lbl2.grid(column=0, row=1)

    btn1 = Button(r_menu_window, text="Supplier
Invoice",fg="white",bg="black",relief="ridge", font=("Times",
12),width=25,command=mclickedbtn1)

    btn1.grid(column=0, row=2)

    btn2 = Button(r_menu_window, text="Customer
Invoice",fg="white",bg="black",relief="ridge", font=("Times", 12),width=25,
command=mclickedbtn2)

    btn2.grid(column=0, row=3)

    btn4 = Button(r_menu_window, text="Return to main
menu",fg="white",bg="black",relief="ridge", font=("Times", 12),width=25,
command=mclickedbtn4)

    btn4.grid(column=0, row=4)

    r_menu_window.mainloop()


#Main Menu

window = Tk()

window.geometry('1500x600')

window.title("Medical Store Management Software")

lbl = Label(window, text="Welcome to Medical Store Management
System!",bg="BLACK",fg="white",font=("Times", 30))

lbl.grid(column=0, row=0)

lbl2 = Label(window, text="What would you like to
do!",bg="RED",relief="ridge",fg="WHITE",font=("Times", 20),width=25)

lbl2.grid(column=0, row=3)

btn1 = Button(window, text="Medicine Menu",fg="BLACK",bg="SKY BLUE",relief="ridge",
font=("Times", 12),width=25, command=clickedbtn1)

btn1.grid(column=0, row=5)
```

```
btn2 = Button(window, text="Customer Menu",fg="BLACK",bg="SKY
BLUE",relief="ridge", font=("Times", 12),width=25, command=clickedbtn2)

btn2.grid(column=0, row=7)

btn3 = Button(window, text="Supplier Menu",fg="BLACK",bg="SKY BLUE",relief="ridge",
font=("Times", 12),width=25, command=clickedbtn3)

btn3.grid(column=0, row=9)

btn4 = Button(window, text="Report Menu",fg="BLACK",bg="SKY BLUE",relief="ridge",
font=("Times", 12),width=25, command=clickedbtn4)

btn4.grid(column=0, row=11)

btn5 = Button(window, text="Invoicing Menu",fg="BLACK",bg="SKY BLUE",relief="ridge",
font=("Times", 12),width=25, command=clickedbtn5)

btn5.grid(column=0, row=13)

window.mainloop()
```

MENU FUNCTION:

```
from tkinter import *

from tkinter import messagebox

import os

f=open("database_proj",'a+')

root = Tk()

root.title("Medical Store Managment System")

root.configure(width=1500,height=600,bg='WHITE')

var=-1


def additem():
```

```
    global var
    num_lines = 0
    with open("database_proj", 'r') as f10:
        for line in f10:
            num_lines += 1
    var=num_lines-1
    e1= entry1.get()
    e2=entry2.get()
    e3=entry3.get()
    e4=entry4.get()
    e5=entry5.get()
    f.write('{0} {1} {2} {3} {4}\n'.format(str(e1),e2,e3,str(e4),e5))
    entry1.delete(0, END)
    entry2.delete(0, END)
    entry3.delete(0, END)
    entry4.delete(0, END)
    entry5.delete(0, END)



def deleteitem():
    e1=entry1.get()
    with open(r"database_proj") as f, open(r"database_proj1", "w") as working:
        for line in f:
            if str(e1) not in line:
                working.write(line)
```

```
    os.remove(r"database_proj")
    os.rename(r"database_proj1", r"database_proj")
    f.close()
    working.close()
    entry1.delete(0, END)
    entry2.delete(0, END)
    entry3.delete(0, END)
    entry4.delete(0, END)
    entry5.delete(0, END)

def firstitem():
    global var
    var=0
    f.seek(var)
    c=f.readline()
    v=list(c.split(" "))
    entry1.delete(0, END)
    entry2.delete(0, END)
    entry3.delete(0, END)
    entry4.delete(0, END)
    entry5.delete(0, END)
    entry1.insert(0,str(v[0]))
    entry2.insert(0,str(v[1]))
    entry3.insert(0,str(v[2]))
    entry4.insert(0,str(v[3]))
```

```
    entry5.insert(0,str(v[4]))


def nextitem():
    global var
    var = var + 1
    f.seek(var)
    try:
        c=f.readlines()
        xyz = c[var]
        v = list(xyz.split(" "))
        entry1.delete(0, END)
        entry2.delete(0, END)
        entry3.delete(0, END)
        entry4.delete(0, END)
        entry5.delete(0, END)
        entry1.insert(0, str(v[0]))
        entry2.insert(0, str(v[1]))
        entry3.insert(0, str(v[2]))
        entry4.insert(0, str(v[3]))
        entry5.insert(0, str(v[4]))
    except:
        messagebox.showinfo("Title", "SORRY!...NO MORE RECORDS")
def previousitem():
        global var
        var=var-1
```

```
        f.seek(var)
        try:
            z = f.readlines()
            xyz=z[var]
            v = list(xyz.split(" "))
            entry1.delete(0, END)
            entry2.delete(0, END)
            entry3.delete(0, END)
            entry4.delete(0, END)
            entry5.delete(0, END)

            entry1.insert(0, str(v[0]))
            entry2.insert(0, str(v[1]))
            entry3.insert(0, str(v[2]))
            entry4.insert(0, str(v[3]))
            entry5.insert(0, str(v[4]))
        except:
            messagebox.showinfo("Title", "SORRY!...NO MORE RECORDS")


def lastitem():
    global var
    f4=open("database_proj",'r')
    x=f4.read().splitlines()
    last_line= x[-1]
```

```
    num_lines = 0
    with open("database_proj", 'r') as f8:
        for line in f8:
            num_lines += 1
    var=num_lines-1
    print(last_line)
    try:
        v = list(last_line.split(" "))
        entry1.delete(0, END)
        entry2.delete(0, END)
        entry3.delete(0, END)
        entry4.delete(0, END)
        entry5.delete(0, END)

        entry1.insert(0, str(v[0]))
        entry2.insert(0, str(v[1]))
        entry3.insert(0, str(v[2]))
        entry4.insert(0, str(v[3]))
        entry5.insert(0, str(v[4]))
    except:
        messagebox.showinfo("Title", "SORRY!...NO MORE RECORDS")



def updateitem():
```

```
    e1 = entry1.get()
    e2 = entry2.get()
    e3 = entry3.get()
    e4 = entry4.get()
    e5 = entry5.get()
    with open(r"database_proj") as f1, open(r"database_proj1", "w") as working:
        for line in f1:
            if str(e1) not in line:
                working.write(line)
            else:
                working.write('{0} {1} {2} {3} {4}'.format(str(e1), e2, e3, str(e4), e5))
    os.remove(r"database_proj")
    #brought to you by code-projects.org
    os.rename(r"database_proj1", r"database_proj")



def searchitem():
    i=0
    e11 = entry1.get()
    with open(r"database_proj") as working:
        for line in working:
            i=i+1
            if str(e11) in line:
                break
        try:
```

```
            v = list(line.split(" "))
            entry1.delete(0, END)
            entry2.delete(0, END)
            entry3.delete(0, END)
            entry4.delete(0, END)
            entry5.delete(0, END)
            entry1.insert(0, str(v[0]))
            entry2.insert(0, str(v[1]))
            entry3.insert(0, str(v[2]))
            entry4.insert(0, str(v[3]))
            entry5.insert(0, str(v[4]))
        except:
            messagebox.showinfo("Title", "error end of file")
    working.close()



def clearitem():
    entry1.delete(0, END)
    entry2.delete(0, END)
    entry3.delete(0, END)
    entry4.delete(0, END)
    entry5.delete(0, END)
#fn1353
label0= Label(root,text="MEDICAL STORE MANAGEMENT SYSTEM
",bg="black",fg="white",font=("Times", 30))
```

```
label1=Label(root,text="ENTER ITEM NAME",bg="SKY
BLUE",relief="ridge",fg="white",font=("Times", 12),width=25)

entry1=Entry(root , font=("Times", 12))

label2=Label(root, text="ENTER ITEM PRICE",bd="2",relief="ridge",height="1",bg="SKY
BLUE",fg="WHITE", font=("Times", 12),width=25)

entry2= Entry(root, font=("Times", 12))

label3=Label(root, text="ENTER ITEM QUANTITY",bd="2",relief="ridge",bg="SKY
BLUE",fg="white", font=("Times", 12),width=25)

entry3= Entry(root, font=("Times", 12))

label4=Label(root, text="ENTER ITEM CATEGORY",bd="2",relief="ridge",bg="SKY
BLUE",fg="white", font=("Times", 12),width=25)

entry4= Entry(root, font=("Times", 12))

label5=Label(root, text="ENTER ITEM DISCOUNT",bg="SKY
BLUE",relief="ridge",fg="white", font=("Times", 12),width=25)

entry5= Entry(root, font=("Times", 12))

button1= Button(root, text="ADD ITEM", bg="PINK", fg="black", width=20, font=("Times",
12),command=additem)

button2= Button(root, text="DELETE ITEM", bg="PINK", fg="black", width =20,
font=("Times", 12),command=deleteitem)

button3= Button(root, text="VIEW FIRST ITEM" , bg="PINK", fg="black", width =20,
font=("Times", 12),command=firstitem)

button4= Button(root, text="VIEW NEXT ITEM" , bg="PINK", fg="black", width =20,
font=("Times", 12), command=nextitem)

button5= Button(root, text="VIEW PREVIOUS ITEM", bg="PINK", fg="black", width =20,
font=("Times", 12),command=previousitem)

button6= Button(root, text="VIEW LAST ITEM", bg="PINK", fg="black", width =20,
font=("Times", 12),command=lastitem)

button7= Button(root, text="UPDATE ITEM", bg="PINK", fg="black", width =20,
font=("Times", 12),command=updateitem)
```

```
button8= Button(root, text="SEARCH ITEM", bg="PINK", fg="black", width =20,
font=("Times", 12),command=searchitem)

button9= Button(root, text="CLEAR SCREEN", bg="PINK", fg="black", width=20,
font=("Times", 12),command=clearitem)

label0.grid(columnspan=6, padx=10, pady=10)

label1.grid(row=1,column=0, sticky=W, padx=10, pady=10)

label2.grid(row=2,column=0, sticky=W, padx=10, pady=10)

label3.grid(row=3,column=0, sticky=W, padx=10, pady=10)

label4.grid(row=4,column=0, sticky=W, padx=10, pady=10)

label5.grid(row=5,column=0, sticky=W, padx=10, pady=10)

entry1.grid(row=1,column=1, padx=40, pady=10)

entry2.grid(row=2,column=1, padx=10, pady=10)

entry3.grid(row=3,column=1, padx=10, pady=10)

entry4.grid(row=4,column=1, padx=10, pady=10)

entry5.grid(row=5,column=1, padx=10, pady=10)

button1.grid(row=1,column=4, padx=40, pady=10)

button2.grid(row=1,column=5, padx=40, pady=10)

button3.grid(row=2,column=4, padx=40, pady=10)

button4.grid(row=2,column=5, padx=40, pady=10)

button5.grid(row=3,column=4, padx=40, pady=10)

button6.grid(row=3,column=5, padx=40, pady=10)

button7.grid(row=4,column=4, padx=40, pady=10)

button8.grid(row=4,column=5, padx=40, pady=10)

button9.grid(row=5,column=5, padx=40, pady=10)

root.mainloop()
```

CUSTOMER FUNCTION

```
import csv
from tempfile import NamedTemporaryFile
import shutil

def customer_id_generator():
    with open('cus_men.csv','r') as csvfile:
        reader=csv.DictReader(csvfile)
        i=1
        for row in reader:
            if int(row['customer_id'])==i:
                i=i+1
    return i

def new_customer():
    with open('cus_men.csv','a+') as csvfile:
        names=['customer_name','customer_id','customer_phone','customer_address']
        writer=csv.DictWriter(csvfile,fieldnames=names)
        writer.writeheader()
        customer_name=input('Enter the name of the customer : ')
        customer_id=customer_id_generator()
        print('Unique customer ID generated : ',customer_id)
        customer_phone=input('Enter the phone number of the customer : ')
        customer_address=input('Enter the address : ')
```

```
writer.writerow({'customer_name':customer_name,'customer_id':customer_id,'customer_phone'
:customer_phone,"customer_address":customer_address})


def search_customer():
    with open('cus_men.csv','r') as csvfile:
        name=input('Enter the name of customer:\n')
        reader=csv.DictReader(csvfile)
        for row in reader:
            if row['customer_name']==name:
                print("----------------------------------------")
                print(" Name : ",row['customer_name'],'\n',"ID : ",row['customer_id'],'\n',"Phone :
",row['customer_phone'],'\n',"Address : ",row['customer_address'])
                print("----------------------------------------")
def update_customer_info():
    tempfile = NamedTemporaryFile(mode='w', delete=False)
    names=['customer_name','customer_id','customer_phone','customer_address']
    with open('cus_men.csv', 'r') as csvfile, tempfile:
        reader = csv.DictReader(csvfile)
        writer = csv.DictWriter(tempfile, fieldnames=names)
        writer.writeheader()
        idno =input('Enter the id of the customer you want to modify!\n')
        for row in reader:
            if row['customer_id'] == idno:
                print('----------------------------------------')
                print("|Enter 1 to change name                 |")
```

```
            print('-------------------------------------------')
            print('|Enter 2 to change phone number           |')
            print('-------------------------------------------')
            print('|Enter 3 to change address                |')
            print('-------------------------------------------')
            choice=int(input("Enter Your Choice!\n"))

            if(choice==1):
                row['customer_name']=input("Enter the new name : ")

            elif(choice==2):
                row['customer_phone']=input("Enter the new phone number : ")

            elif(choice==3):

                row['customer_address']=input("Enter the new address : ")

        row =
{'customer_name':row['customer_name'],'customer_id':row['customer_id'],'customer_phone':ro
w['customer_phone'],"customer_address":row['customer_address']}
        writer.writerow(row)

    shutil.move(tempfile.name, 'cus_men.csv')
```

SUPPLIER FUNCTION:

```
import csv
from tempfile import NamedTemporaryFile
import shutil
def supplier_id_generator():
    with open('supplier.csv', 'r') as csvfile:
        reader=csv.DictReader(csvfile)
        i=1
        for r in reader:
            if int(r['sup_id'])==i:
                i=i+1
    return i
def create_supplier():
    with open('supplier.csv', 'a+') as csvfile:
        columns = ['sup_name', 'sup_id', 'sup_city', 'sup_contact', 'sup_email']
        writer = csv.DictWriter(csvfile, fieldnames = columns)

        sup_name = input("Enter New Supplier's Name : ")
        sup_id = supplier_id_generator()
        print('Unique Supplier ID Generated : ', sup_id)
        sup_city = input("Enter New Supplier's City : ")
        sup_contact = int(input("Enter New Supplier's Contact Number : "))
        sup_email = input("Enter New Supplier's Email Id : ")
        writer.writerow({'sup_name':sup_name, 'sup_id':sup_id, 'sup_city':sup_city,
'sup_contact':sup_contact, 'sup_email':sup_email})
```

```
def s_searchbyname():
    with open('supplier.csv','r') as csvfile:
        name=input('Enter Supplier Name!\n')
        reader=csv.DictReader(csvfile)
        for r in reader:
            if r['sup_name'] == name:
                print('Name : ', r['sup_name'], '\n', 'Id : ', r['sup_id'],'\n', 'City : ', r['sup_city'], '\n', 'Contact No :', r['sup_contact'], '\n', 'Email id : ', r['sup_email'])
def s_searchbyid():
    with open('supplier.csv','r') as csvfile:
        id=int(input('Enter Supplier ID!\n'))
        reader=csv.DictReader(csvfile)
        for r in reader:
            if r['sup_id'] == id:
                print('Name : ', r['sup_name'], '\n', 'Id : ', r['sup_id'],'\n', 'City : ', r['sup_city'], '\n', 'Contact No :', r['sup_contact'], '\n', 'Email id : ', r['sup_email'])
def search_supplier():
    ss_choice=0
    while(ss_choice!=3):
        print('-------------------------------------------')
        print("|Enter 1 to search supplier by name!       |")
        print('-------------------------------------------')
        print("|Enter 2 to search supplier by id!         |")
        print('-------------------------------------------')
        print("|Enter 3 to exit supplier search!          |")
```

```
        print('-----------------------------------------')
        ss_choice=int(input("Enter your choice!\n"))
        if ss_choice==1 :
            s_searchbyname()
        elif ss_choice==2 :
            s_searchbyid()
        else:
            print("Invalid Input! Try again!\n")
def update_supplier_info():
    tempfile = NamedTemporaryFile(mode='w', delete=False)
    columns = ['sup_name', 'sup_id', 'sup_city', 'sup_contact', 'sup_email']
    with open('supplier.csv', 'r') as csvfile, tempfile:
        reader = csv.DictReader(csvfile)
        writer = csv.DictWriter(tempfile, fieldnames=columns)
        writer.writeheader()
        suppp_name=input('Enter the name of the supplier you want to modify!\n')
        for r in reader:
            if r['sup_name'] == suppp_name:
                print('-----------------------------------------')
                print('|Enter 1 to update supplier name.        |')
                print('-----------------------------------------')
                print('|Enter 2 to update supplier id.          |')
                print('-----------------------------------------')
                print('|Enter 3 to update supplier city.        |')
                print('-----------------------------------------')
```

```
                print('|Enter 4 to update supplier contact no.     |')
                print('-------------------------------------------')
                print('|Enter 5 to update supplier email id.       |')
                print('-------------------------------------------')
                choice=int(input('Enter your choice!\n'))
                if(choice==1):
                    r['sup_name']=input("Enter updated name : ")
                elif(choice==2):
                    r['sup_id']=int(input("Enter updated id : "))
                elif(choice==3):
                    r['sup_city']=input("Enter updated city : ")
                elif(choice==4):
                    r['sup_contact']=int(input("Enter updated contact : "))
                elif(choice==5):
                    r['sup_email']=int(input("Enter updated email id : "))
                else:
                    print("Invalid Input!\n")
            r = {'sup_name':r['sup_name'], 'sup_id':r['sup_id'], 'sup_city':r['sup_city'], 'sup_contact':r['sup_contact'], 'sup_email':r['sup_email']}
            writer.writerow(r)
    shutil.move(tempfile.name, 'supplier.csv')
```

REPORT FUNCTION:

```
import csv
from tempfile import NamedTemporaryFile
```

```
import shutil
import datetime
d = datetime.datetime.now()
date= d.strftime("%d")
month= d.strftime("%m")
year = d.strftime("%Y")

def day_sale():
    print('Enter Date : ')
    date = input()
    print('Enter Month : ')
    month = input()
    print('Enter Year : ')
    year = input()
    count=0.0
    with open('sales.csv','r+') as csvfile:
        reader = csv.DictReader(csvfile)
        print('-----------------Day\'s Sales-----------------')
        for r in reader:
            if r['sale_date']==date and r['sale_month']==month and r['sale_year']==year :
                count=count+float(r['total'])
                print('Medicine Name : ', r['medi_name'])
                print('Medicine Id : ', r['med_id'])
                print('Sale : ', r['sale'])
                print('Quantity : ', r['quantity'])
```

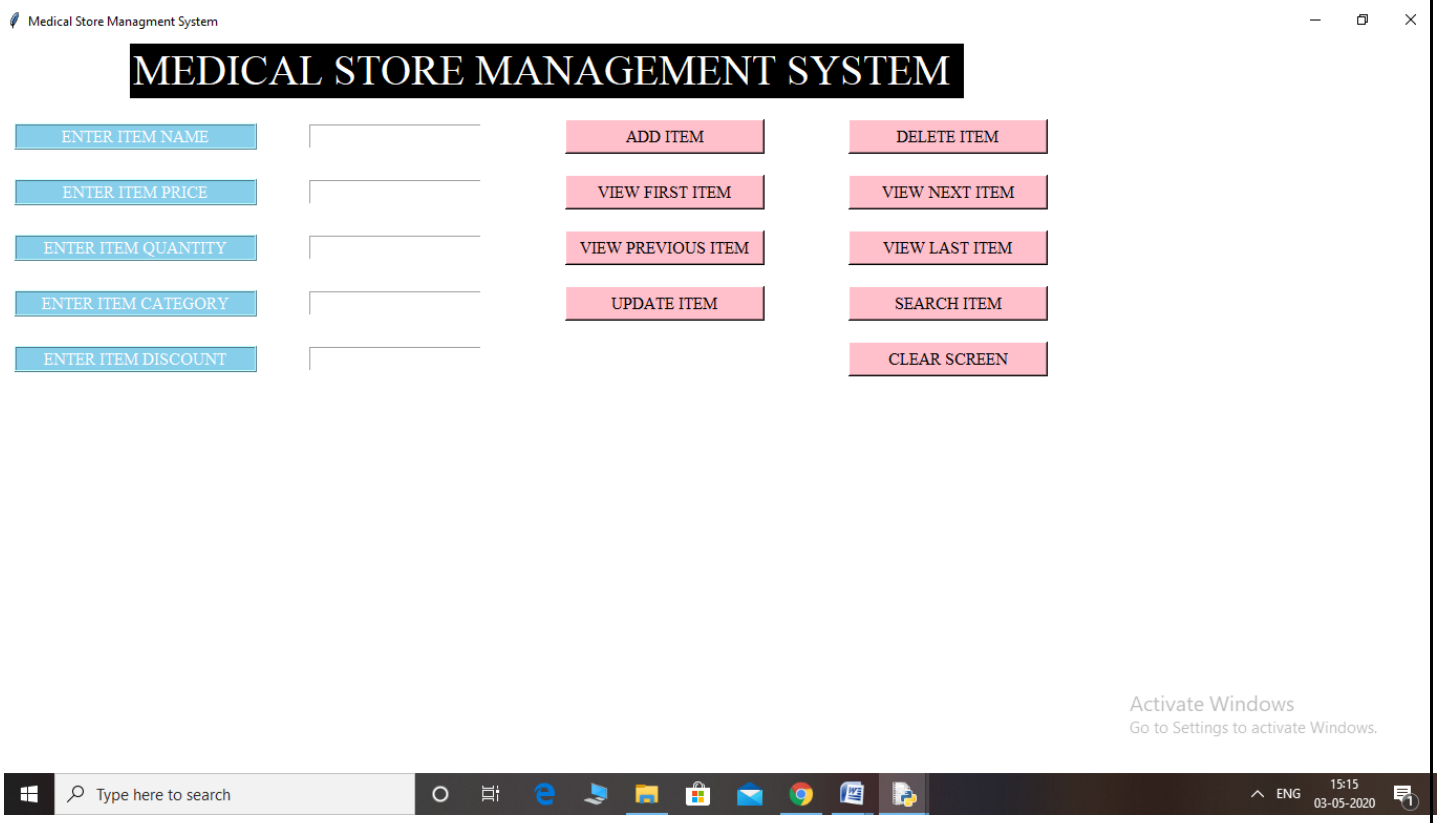
```
            print('Total : ', r['total'])
            print('\n')
        print('-----------------------------------------------')
        print('Total sales for the day : ', count)
        print('-----------------------------------------------')
def month_sale():
    print('Enter Month : ')
    month = input()
    print('Enter Year : ')
    year = input()
    count=0.0
    with open('sales.csv','r+') as csvfile:
        reader = csv.DictReader(csvfile)
        print('-----------------Day\'s Sales-----------------')
        for r in reader:
            if r['sale_month']==month and r['sale_year']==year :
                count=count+float(r['total'])
                print('Medicine Name : ', r['medi_name'])
                print('Medicine Id : ', r['med_id'])
                print('Sale : ', r['sale'])
                print('Quantity : ', r['quantity'])
                print('Total : ', r['total'])
                print('\n')
        print('-----------------------------------------------')
        print('Total sales for the month : ', count)
```

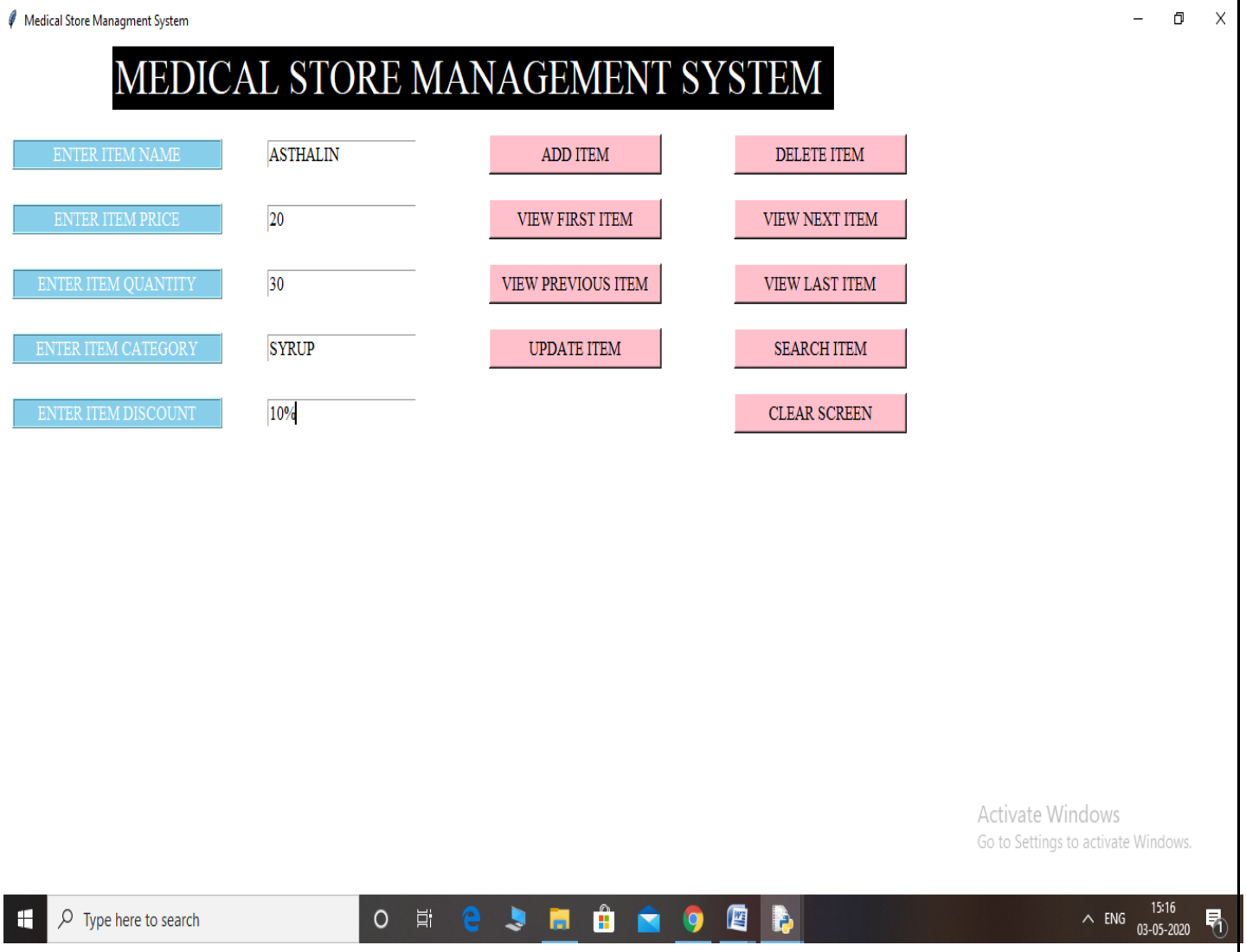
```
        print('-----------------------------------------------')
def day_purchase():
    print('Enter Date : ')
    date = input()
    print('Enter Month : ')
    month = input()
    print('Enter Year : ')
    year = input()
    count=0.0
    with open('purchase.csv','r+') as csvfile:
        reader = csv.DictReader(csvfile)
        print('-----------------Day\'s Purchase-----------------')
        for r in reader:
            if r['pur_date']==date and r['pur_month']==month and r['pur_year']==year :
                count=count+float(r['total'])
                print('Medicine Name : ', r['medi_name'])
                print('Medicine Id : ', r['med_id'])
                print('Purchase cost per item : ', r['unit'])
                print('Quantity : ', r['quantity'])
                print('Total : ', r['cost'])
                print('\n')
        print('-----------------------------------------------')
        print('Total Purchase cost for the day : ', count)
        print('-----------------------------------------------')
def month_purchase():
```

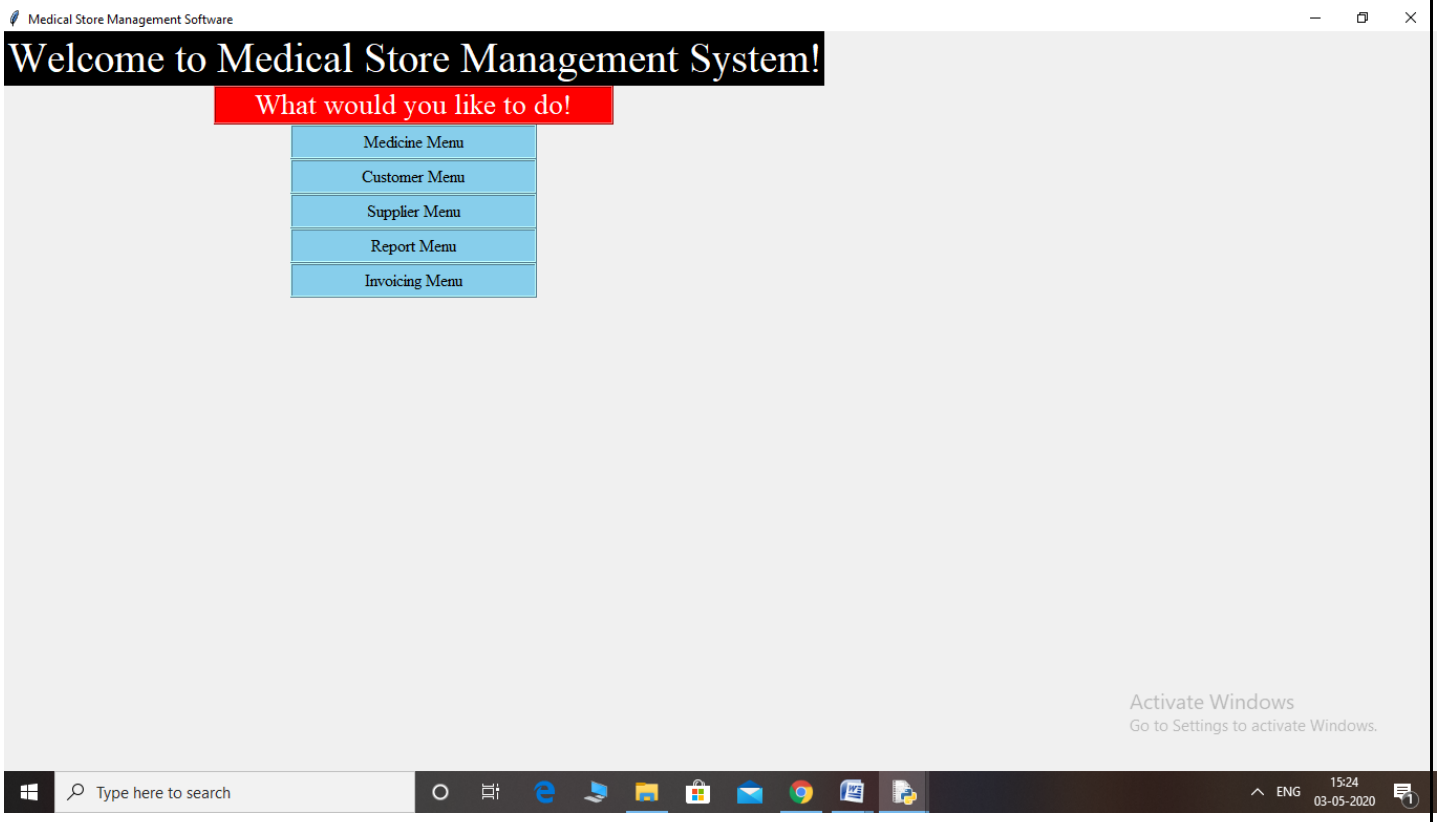
```
    print('Enter Month : ')
    month = input()
    print('Enter Year : ')
    year = input()
    count=0.0
    with open('purchase.csv','r+') as csvfile:
        reader = csv.DictReader(csvfile)
        print('-----------------Day\'s Purchase-----------------')
        for r in reader:
            if r['pur_month']==month and r['pur_year']==year :
                count=count+float(r['total'])
                print('Medicine Name : ', r['medi_name'])
                print('Medicine Id : ', r['med_id'])
                print('Purchase cost per item : ', r['unit'])
                print('Quantity : ', r['quantity'])
                print('Total : ', r['cost'])
                print('\n')
        print('------------------------------------------------')
        print('Total Purchase cost for the month : ', count)
        print('------------------------------------------------')
def profit_report():

        print('Enter Month : ')
        month = input()
        print('Enter Year : ')
```

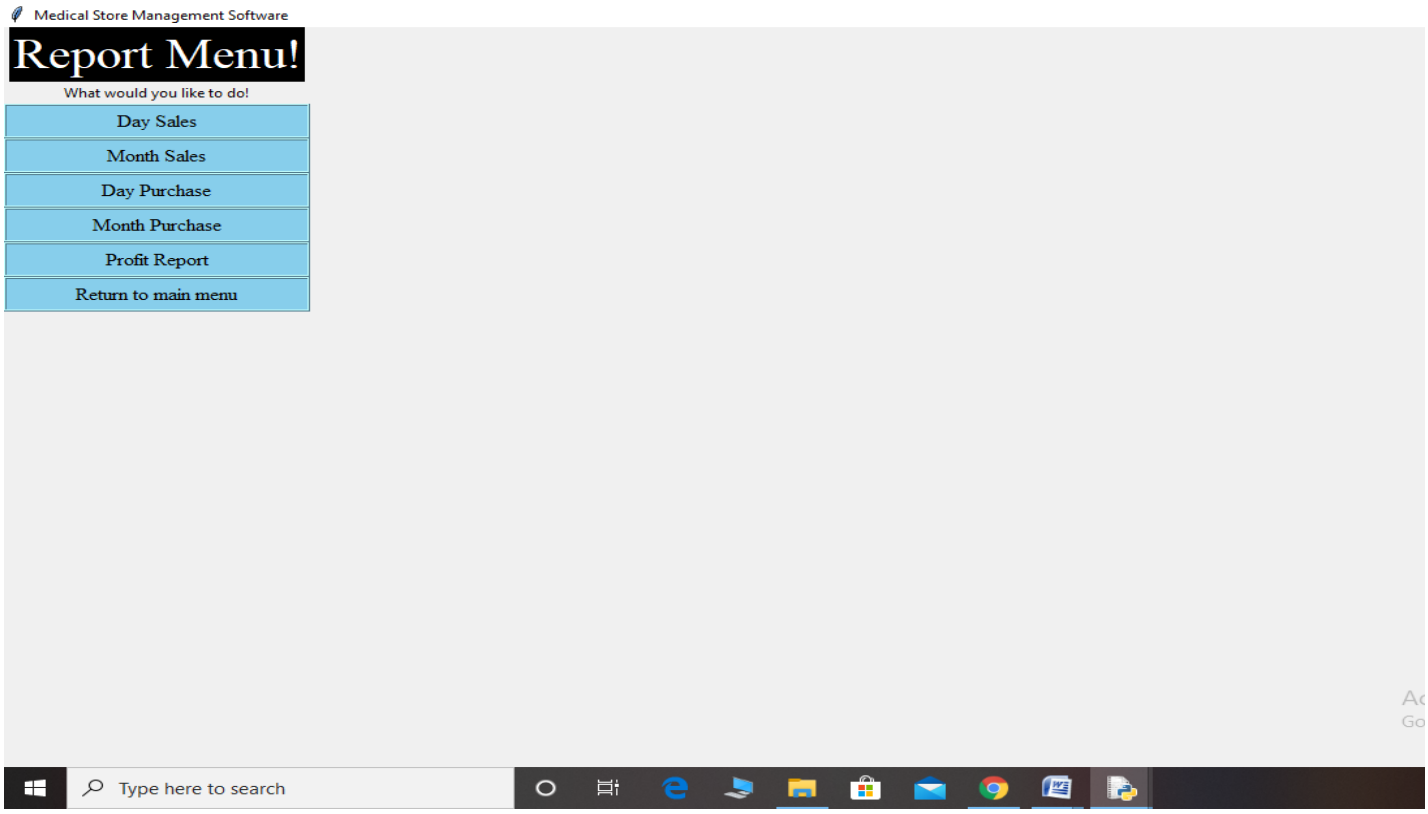
```
year = input()
count1=0.0
count2=0.0
with open('sales.csv','r+') as csvfile :
    reader = csv.DictReader(csvfile)
    for r in reader:
        if r['sale_month']==month and r['sale_year']==year :
            count1=count1+float(r['total'])
with open('purchase.csv', 'r+') as csvfile :
    reader = csv.DictReader(csvfile)
    for r in reader:
        if r['pur_month']==month and r['pur_year']==year :
            count2=count2+float(r['cost'])
profit = count1-count2
print("Profit for ", month, " - ", year, " is ", profit, "!\n")
```

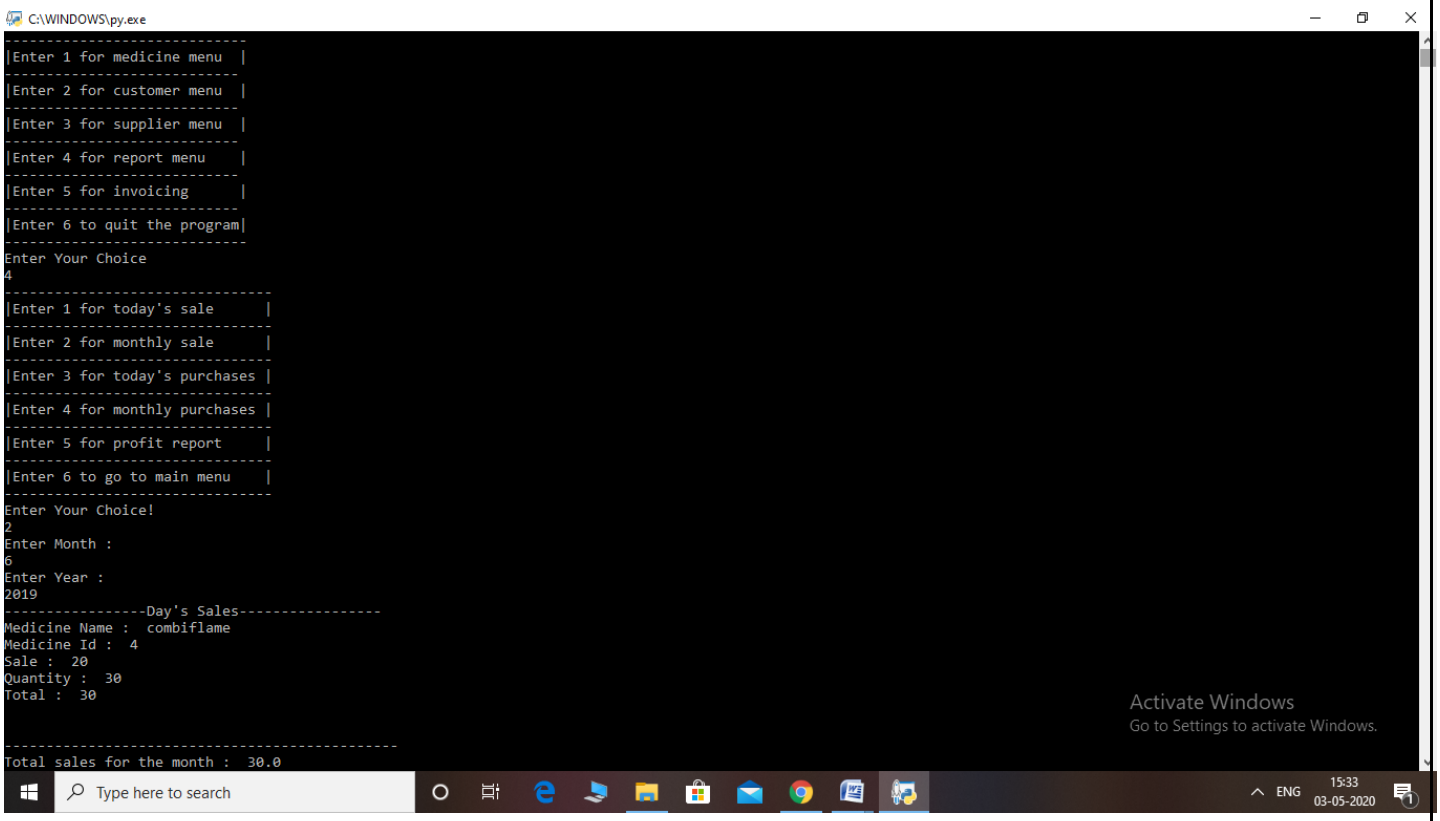
OUTPUT


Medical Store Managment System
MEDICAL STORE MANAGEMENT SYSTEM
ENTER ITEM NAME
ENTER ITEM PRICE
ENTER ITEM QUANTITY
ENTER ITEM CATEGORY
ENTER ITEM DISCOUNT
ADD ITEM
VIEW FIRST ITEM
VIEW PREVIOUS ITEM
UPDATE ITEM
DELETE ITEM
VIEW NEXT ITEM
VIEW LAST ITEM
SEARCH ITEM
CLEAR SCREEN
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
15:15
03-05-2020


Medical Store Managment System
MEDICAL STORE MANAGEMENT SYSTEM
ENTER ITEM NAME
ASTHALIN
ADD ITEM
DELETE ITEM
ENTER ITEM PRICE
20
VIEW FIRST ITEM
VIEW NEXT ITEM
ENTER ITEM QUANTITY
30
VIEW PREVIOUS ITEM
VIEW LAST ITEM
ENTER ITEM CATEGORY
SYRUP
UPDATE ITEM
SEARCH ITEM
ENTER ITEM DISCOUNT
10%
CLEAR SCREEN
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
15:16
03-05-2020


Medical Store Management Software
Welcome to Medical Store Management System!
What would you like to do!
Medicine Menu
Customer Menu
Supplier Menu
Report Menu
Invoicing Menu
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
15:24
03-05-2020


Medical Store Management Software
Report Menu!
What would you like to do!
Day Sales
Month Sales
Day Purchase
Month Purchase
Profit Report
Return to main menu
Type here to search


C:\WINDOWS\py.exe
|Enter 1 for medicine menu |
|Enter 2 for customer menu |
|Enter 3 for supplier menu |
|Enter 4 for report menu |
|Enter 5 for invoicing |
|Enter 6 to quit the program|
Enter Your Choice
4
|Enter 1 for today's sale |
|Enter 2 for monthly sale |
|Enter 3 for today's purchases |
|Enter 4 for monthly purchases |
|Enter 5 for profit report |
|Enter 6 to go to main menu |
Enter Your Choice!
2
Enter Month :
6
Enter Year :
2019
----------Day's Sales----------
Medicine Name : combiflame
Medicine Id : 4
Sale : 20
Quantity : 30
Total : 30
Total sales for the month : 30.0
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
15:33
03-05-2020

SIZE AND EFFORT

AIM:- Size and effort required for Medical Store Management System.

Size and effort:

Size of code: 2004 LOC

2.004 KLOC

Applying the COCOMO model for estimation of effort, duration and productivity of the project, we found the basic model suitable for our needs. Further our project fell under the organic type in the basic model.

Effort = E = a *

$(KLOC)^b$ Duration

= D = c * (E)^d

Productivity =

KLOC/E

where a, b, c and d are variables that have predefined values given as follows:

a = 2.4
b = 1.05
c = 2.5
d = 0.38

Effort = 4.9797 PM

Duration = 3.2558 Months (Maximum time)

Productivity = 0.4024 KLOC/PM

TEST CASES

AIM:- Test cases for Medical Store Management System.

Introduction:

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Testing presents an interesting of a system using various test data. Preparation of the test data plays a vital role in the system testing. After preparation the test data, the system under study is tested those test data. Errors were found and corrected by using the following testing steps and corrections are recorded for future references. Thus, series of testing is performed on the system before it is already for implementation.

The development of software systems involves a series of production activities where opportunities for injection of human errors are enormous. Errors may begin to occur at the very inception of the process where the objectives may be erroneously or imperfectly specified as well as in later design and development stages. Because of human in ability to perform and communicate with perfection, software development is followed by assurance activities.

Quality assurance is the review of software products and related documentation for completeness, correctness, reliability and maintainability. And of course it includes assurances that the system meets the specification and the requirements for its intended use and performance. The various levels of quality assurance are described in the following sub sections.

System Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specifications, design and coding. The testing phase involves the testing of system using various test data; Preparation of test data plays a vital role in the system testing. After preparation the test data, the system under study is tested.

Those test data, errors were found and corrected by following testing steps and corrections are recorded for future references. Thus a series testing is performed on the system before it is ready for implementation.

The various types of testing on the system are:

- Unit testing
- Integrated testing

- Validation testing
- Output testing
- User acceptance testing

Unit testing

Unit testing focuses on verification effort on the smallest unit of software design module. Using the unit test plans. Prepared in the design phase of the system as a guide, important control paths are tested to uncover errors within the boundary of the modules. The interfaces of each of the modules under consideration are also tested. Boundary conditions were checked.

All independent paths were exercised to ensure that all statements in the module are executed at least once and all error-handling paths were tested. Each unit was thoroughly tested to check if it might fall in any possible situation. This testing was carried out during the programming itself. At the end of this testing phase, each unit was found to be working satisfactorily, as regarded to the expected out from the module.

Integration Testing

Data can be across an interface one module can have an adverse effect on another's sub function, when combined may not produce the desired major function; global data structures can present problems. Integration testing is a symmetric technique for constructing tests to uncover errors associated with the interface. All modules are combined in this testing step. Then the entire program was tested as a whole.

Validation Testing

At the culmination of integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and final series of software test-validation testing begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the consumer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two conditions exists.

- The function or performance characteristics confirm to specification that are accepted.
- A validation from specification is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of the project with the help of user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since a system is useful if it does not produce the required output in the specific format required by them tests the output generator displayed on the system under consideration. Here the output is considered in two ways: - one is onscreen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system design phase according to the user needs. As far as hardcopies are considered it goes in terms with the user requirement. Hence output testing does not result any correction in the system.

User Acceptance Testing

User acceptance of the system is a key factor for success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with prospective system and user at the time of developing and making changes whenever required.

TEST RESULT: UNIT TESTING

LOGIN FORM:

SL.No	Test Case	Excepted Result	Test Result
1	Enter valid name and password & click on login button	Software should display main window	Successful
2	Enter invalid	Software should not display main window	successful

CUSTOMER / SUPPLIER / MEDICINE ENTRY FORM:

SL.No	Test Case	Excepted Result	Test Result
1	On the click of ADD button	At first user have to fill all fields with proper data , if any Error like entering text data instead of number or entering number instead of text..is found then it gives proper message otherwise Adds Record To the Database	successful
2.	On the Click of DELETE Button	This deletes the details in database.	Successful
3.	On the Click of UPDATE Button	Modified records are Updated in database by clicking UPDATE button.	Successful
4.	On the Click of SEARCH Button	Displays the Details by entering name. Otherwise gives proper Error message.	Successful
5.	On the Click of EXIT button	Exit the form.	successful

SALE INVOICE FORM:

SL.No	Test Case	Excepted Result	Test Result
1	On the click of Calculate button	At first user have to fill all fields with proper data , if any Error like entering text data instead of number or entering number instead of text..is found then it gives proper message otherwise after clicking calculate button it calculates the the total amount.	successful
2.	On the Click of Save Button	This saves the details of sale in database.	Successful
3.	On the Click of Load Table Button	It load the sale table from the database and displays it on the form.	Successful
4.	On the Click of EXIT button	Exit the form.	successful

REFERENCES

1. http://www.w3school.com

2. http://en.wikipedia.org/wiki/Waterfall_model

3. http://en.wikipedia.org/wiki/Software_development_process,

4. http://en.wikipedia.org/wiki/Spiral_model

5. http://en.wikipedia.org/wiki/Prototype

6. https://www.google.com.pk/?gws_rd=cr,ssl&ei=ey3HVPXGC6HmyQOCxIDQCA#q=prototype
   +model+diagrams

7. https://www.google.com.pk/?gws_rd=cr,ssl&ei=9LXNVMaeI9bnatzKgng#q=
   history%20of%20o nline%20fir%20system%20in%20pakistan

8. http://www.incpak.com/editorial/govt-must-upgrade-police-computerized-fir-online-