# 

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**A**

**MINI PROJECT REPORT**

ON

**“MEDICINE DATABASE MANAGEMENT SYSTEM ”**

Submitted in the partial fulfillment of the requirements in the 4th semester of

**BACHELOR OF ENGINEERING**

**IN**

**INFORMATION SCIENCE AND ENGINEERING**

BY

**DHANUSH BILIGIRI N H**

**1NH18IS030**

***Under the guidance of***

**PROF.Rafega Begam**

Assistant Professor, Dept. of ISE, NHCE

**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**NEW HORIZON COLLEGE OF ENGINEERING**

(Autonomous College Permanently Affiliated to VTU, Approved by AICTE, Accredited by NBA & NAAC with ‘A’ Grade)

Ring Road, Bellandur Post, Near Marathalli,

Bangalore-560103, INDIA



**CERTIFICATE**

Certified that the project work entitled “**MEDICINE DATABASE MANAGEMENT SYSTEM** ***”***carried out by Mr.DHANUSH BILIGIRI N H, USN 1NH18IS030, a bonafide student of 4th sem in partial fulfillment for the award of Bachelor of Engineering in Information Science and Engineering of the Visveswaraiah Technological University, Belgaum during the year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated.The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Signature of the Guide Signature of the HOD Signature of the Principal

Prof.Rafega Dr. R.J Anandhi Dr.Manjunatha

ORIGINALITY REPORT

8%

SIMILARITY INDEX

%

INTERNET SOURCES

8%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

1 Ralph Steyer. "Chapter 16 Grafische Oberflächen (GUI) mit Python – Das Modul tkinter als GUI-Framework", Springer Science and Business Media LLC, 2018

Publication

2 Pro Windows 8 1 Development with XAML and C#, 2014.

Publication

3 James Liech Gor. "Agent-based interoperability system in health insurance", 2017 IST-Africa Week Conference (IST-Africa), 2017

Publication

4 Dr. Ossama Embarak. "Data Analysis and Visualization Using Python", Springer Science and Business Media LLC, 2018

Publication

5 "Advances in Computer Science, Engineering & Applications", Springer Science and Business Media LLC, 2012

Publication

4%

1%

1%

<1%

<1%

6 "Proceedings of International Conference on Remote Sensing for Disaster Management", Springer Science and Business Media LLC, 2019

Publication

7 Hans Petter Langtangen. "Chapter 6 Introduction to GUI Programming", Springer Science and Business Media LLC, 2004

Publication

8 D. Szécsi, Z. Bagoly, J. Kóbori, I. Horváth, L. G. Balázs. " Direction dependent background fitting for the GBM data ", Astronomy & Astrophysics, 2013

Publication

9 K. S. F. Azam, A. Rahman, K. Paul, Md.S. Bari. "Smart Car Parking Strategies For The Entrepreneurially Challenged Through Mobile Application With Real Time Survey", 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2019

Publication

* M Kavya, C Kishore, K S Rajath Kumar. "Verification and validation of data on Wi-Fi Over The Air(OTA)", 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication

<1%

<1%

<1%

<1%

<1%

Technology (RTEICT), 2018

Publication

* Xiangyu Zhang, Sihan Tao, Zhitao Gong, Bo Wu, Ruixin Wang, Bogdan M. Wilamowski. "An improved English to Chinese translation of technical text", 2015 IEEE 19th International Conference on Intelligent Engineering Systems (INES), 2015

Publication

* Santiago L. Rovere, Michael J. North, Guillermo

P. Podesta, Federico E. Bert. "Practical Points

for the Software Development of an Agent- Based Model of a Coupled Human-Natural System", IEEE Access, 2016

Publication

* Janine Dixon. "Chapter 7 Enhancing the Links Between Income Sources, User Expenditures and Taxes in a CGE Database", Springer Science and Business Media LLC, 2017

Publication

* Chet Hosmer. "Defending IoT Infrastructures with the Raspberry Pi", Springer Science and Business Media LLC, 2018

Publication

* Christian Bessiere, Remi Coletta, Gaelle Hisler, Anastasia Paparrizou. "Complexity Results in Optimistic/Pessimistic Preference Reasoning",

<1%

<1%

<1%

<1%

<1%

2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), 2016

Publication

* Ishizaki, Kazuaki, Takeshi Ogasawara, Jose Castanos, Priya Nagpurkar, David Edelsohn, and Toshio Nakatani. "Adding dynamically-typed language support to a statically-typed language compiler : performance evaluation, analysis, and tradeoffs", ACM SIGPLAN Notices, 2012.

Publication

<1%

# CHAPTER 1

## INTRODUCTION

Medicare is a software designed to meet the increasing health concerns among the people by providing quality medicines. The software is developed to interact with the user to add medicines, delete medicines from the stock and search medicines to modify the contents of it. The admin can also bill the medicines purchased by the customer and check the revenue obtained from it.

This is implemented using SQLite studio server where the data of the medicines are stored. The backend is connected to the python script and using graphical user interface the user takes input and stores data in the database. The tables created at the back end are later accessed for retrieving the required information.

Python uses Tkinter tool kit to create user interface. Inserting and accessing data is happening with the help of backend through SQLite3 connection. Tables are created manually in the SQLite3 server, it is invoked through the connect method in the program and the database transaction starts.

SQLite is a relational database that provides vast space for storing data contained in a C library. SQLite is not a client-server database model rather it is embedded into the end program. It is used by various widespread browsers, embedded systems like mobile phones and many others. The table creation and commands used in SQLite is similar to SQL commands and accessing of database is found to be easier.

## 1.1 Motivation of the project

* Every sort of work is done on paper.
* Burden to store information for long period of time.
* Work is Laborious.
* Nearly 2000 drugs are maintained in large records.
* It is very difficult to notify what are the drugs need to be ordered.
* It is very difficult to notify the expiry dates of the every drug and it is very laborious to notify for every drug as every drug has two or more expiry dates in the store.

To overcome the disadvantages discussed above the application medi-care is developed. This software allows the admin to add, modify and delete the medicine stock from the database and it also allows the admin to bill purchased medicine.

## `1.2 Problem statement

To develop an application for managing different types of medicine in a database. This application should identify customers uniquely by vc\_id (valuable customer), name and address, each medicine should have unique id, name, type, quantity left, cost, purpose, rack location and manufactured date. Bill generated should have a unique bill no, cus\_name, items, total cost etc. Here admin should be able to add any number of medicines to the stock and delete expired medicines.

## Methodology

Medicare is a software developed to provide customers with trustworthy medicines. Firstly, the medicines are first added to the database by the admin giving details of the medicine like name, purpose, cost, Expiry date medicine type and symptoms. The admin can modify the medicine contents for future selling based on dosage or exp\_date. The owner can delete medicines from the stock if not required, can keep track of the revenue collected each day based on the medicines being sold. This concept is implemented using python and SQLite acting as front end and back end.

The project is compiled of two parts:

* + - Frontend: Coded in Python and takes input from the user using GUI and retrieves the information from the Database and displays in GUI.
    - Backend: It is usually the database on to which the data is collected and stored. SQLite in used in case of our project to collect information and store in structured format.

The software after being created using the database and python programming takes input from the user using GUI which consists of labels, widgets, buttons that are bind and packed together to form a proper interface. The python program invokes the SQLite database whenever values are entered for the corresponding schemas created.

# CHAPTER 2

## SYSTEM REQUIREMENTS SPECIFICATION

A Software Requirement Specification is the description of the software to be developed. In order to understand the project, it is very important to list out the requirements. The given below hardware system requirements and software system requirements are enough to implement this code without any problems or errors. The web application source code can be easily implemented with these system configurations.

## Hardware requirements:

* + - Processor : Intel core i3 and above
    - Speed : 3.20GHz to 3.60GHz
    - RAM : 512 MB and above
    - Hard disk : 256 GB and above

## Software requirements:

* + - Operating System: Windows 7 or more
    - IDLE (Python 3.7 32-bit)
    - SQLite Studio (3.2.1)

## About the language

**SQL**

SQL stands for Structured Query Language. It’s a computer language used for managing the relational database and manipulation of data. It is used to do all the operation in database like creation of schema, table, inserting, updating, deleting and also to retrieve particular data. SQL

is used as a medium to communicate with the database. SQL is used by various database management system like: MySQL, oracle, Vertica, Sybase, etc.

Some of the commands used in SQL are:

#### 1. DDL commands:

DDL are Data Definition Language used to define the structure of the schema and these commands can manipulate only the table definition but not the contents inside the table.

Example of DDL commands are:

##### CREATE:

This command is used to create a table in a database, generate assertions and triggers in database design. The example of create command implemented in this project is shown below:

Create table med (sl\_no int primary key, name text, type text, qty\_left int, cost int, purpose text, loc text, mfg text);

##### DROP:

This command is used to delete table or the objects from the database. Ex. Drop table med;

##### ALTER:

This command is used to alter the table structure like table definitions, constraints, delete columns, add columns, modify constraints. It has various options like ADD, MODIFY, CHANGE, DELETE.

Ex. Alter table med change purpose side\_effects text; Alter table cus drop column vc\_address;

##### TRUNCATE:

This command is used to delete the contents of the table including all the spaces. Ex. Truncate table log;

#### DQL commands:

DQL stands for Data Query Language for performing database queries. These commands are used to perform specific queries on the existing database using the select statement.

##### SELECT:

This command is used to retrieve the data from the database. It is one of the widely used command and is complex.

Ex: select \*from med;

#### DML commands:

DML stands for Data Manipulation Language and is used to manipulate the rows in the table. The rows in the table can be updated, deleted, inserted, etc.

##### INSERT:

Insert command is used to insert the tuples into the table. Ex: insert into log values (admin, yashwanth);

##### UPDATE:

Update command is used to update the existing values of the tuple. Tuple values can be modified for further access using the update command.

Ex: update med set type=’blood pressure’ where name=’targit’;

##### DELETE:

Delete command is used to delete the tuples from the tables. Ex: delete from med where type=’drowsy’;

#### TCL command:

TCL stands for Transaction Control Language which deals with the transaction within the database.

##### COMMIT:

This command is used to commit the transaction so that the previous transaction will be successfully saved into the database. Once commit is done it is not possible to rollback. Syntax: COMMIT;

##### ROLLBACK:

This command is used to rollback undo the transaction if any error occurs. Syntax: ROLLBACK;

## Python

Python is a high-level programming language designed to be easy to read and simple to implement. It is open source, which means it is free to use, even for commercial applications. Python can run on Mac, Windows, and Unix systems and has also been ported to Java and .NET virtual machines.

### Python Features

##### Brod Standard Library:

Python has huge collection of defined libraries which makes very easy to code in python. Its library is portable and compatible with all platforms like Macintosh, UNIX, and Windows. We need not have to write our own code for each and every thing as it provides rich sets of modules and functions. It has various libraries for web browsing, regular expressions, etc.

##### Interpreted Language:

Python is one of the Interpreted Language as its code is executed line by line at a time. It is not required to compile our code like in other languages like Java, C++, etc. which makes it easier to debug our code. The source code of python is converted into an immediate form called byte code.

##### Support for GUI Programming:

Python provides various modules like PyQt, Tkinter, wxPython through which user can created Graphical User Interface for mobile applications. The most popular for creating graphical apps using python is PyQt5. Tkinter also provides all of the required options to create a beautiful user interface even Gaffer is made importing Tkinter module. This user

interface can be connected to the backend using any one of the DBMS also supported by python, makes it more beautiful.

##### Object Oriented Programming Language:

Python is an object-oriented programming language which include the concept of class and object. It supports all OOPs concepts like inheritance, data abstraction, polymorphism, encapsulation etc.

##### Scalable and Extendable:

Python provides a better structure and support for large programs than shell scripting. You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

##### Free and Open Source:

Python language is available freely at the official website as it is an open source and free of cost, anyone can download it from the official website link as it is open to the public.

##### Python is a Portable-Language:

Python is considered to be a Portable Language because we can run Python program developed on the windows platform on any other platform also like Linux, Unix and Mac.

##### Dynamically-Typed Language:

Python is a dynamically typed language as for instance type of the variable in a program is decided is decides only during the run time whether it is int, float, double and not in advance because we need not specify the type of the variable.

##### Easy to Learn and Use:

Python is easy to learn and use. Python language is developer-friendly and also a high- level programming language.

##### High-Level-Language:

Python is considered to be the most pre-dominant programming language as it a High- LevelLanguage. In Python there is no need to remember the system architecture nor there is no need to manage the memory.

Tkinter Widgets

##### Frame:

A frame is a like container which associates other widgets. It is responsible for holding all the widgets together for grouping and organizing widgets. A bunch of labels, entry, buttons, etc. can be added into the frame and moving frame alone moves other widgets too. Various options like: bg, bd, cursor, relief, width, etc. can be used to configure frame.

### List-box:

List box displays a list of contents which a user interacts with and user can accept any number of times. It looks like a column of a tables that displays values in various rows. It provides option to browse, select multiple, select single through select mode option. It also offers other variety of options like: bg, fg, font, height, width, highlightcolor, etc.

Medicare makes use of List box to display medicine list and the contents of each medicine.

##### Button:

Button is most widely used widget among all widgets in GUI with Tkinter. It is a functional widget that is clickable and on click it performs some action defined in the command option. Buttons are used for linking two functions. They can display text or images same as labels, but also have a whole range of new options used to control their behaviour.

Medicare has many buttons to link two or more functions which on click leads to another window and to select and perform various operations.

##### Label:

A label is a widget which is used to display non-editable text. Label in fact is also used to display images using Photo Image module. The most commonly used label is with ‘text’

configuration option and can change this at any time. Label makes use of many options like fg, bg, font, width, height, etc.

Syntax: Label( window\_name, options) We can add an image into label as below:

project\_img=PhotoImage (file="path with file name")

##### Entry:

An entry is a single line text field user can use to type anything. It’s mostly used in log in form for retrieving username and password. It has a special property to hide/ encrypt the text typed by user by using “show=’\*’ ”option which replaces each and every letter with the specified symbol/letter (in this case every letter typed by user is encrypted with ‘\*’).

Syntax: Entry (window\_name, options)

### ScrolledText:

This widget provides the feature of multiple line input field with scrollbar wherein user can type multiple lines of text. This very much useful for typing paragraphs, letter, essays, etc. It also supports various options like: height, width, etc.

### Scrollbar:

Scrollbar is a kind of widget that is used to scroll down other widget contents like listbox, text and canvas. Horizontal scrollbars can also be created for entry widget. Syntax: w=Scrollbar (top, options).

### Spinbox:

Spinbox is a widget used as a variant of the standard Tkinter entry widget. This Spinbox can be used to select values from a fixed number of values available.

The syntax for the Spinbox to create a widget is shown below : W=Spinbox (window, options).

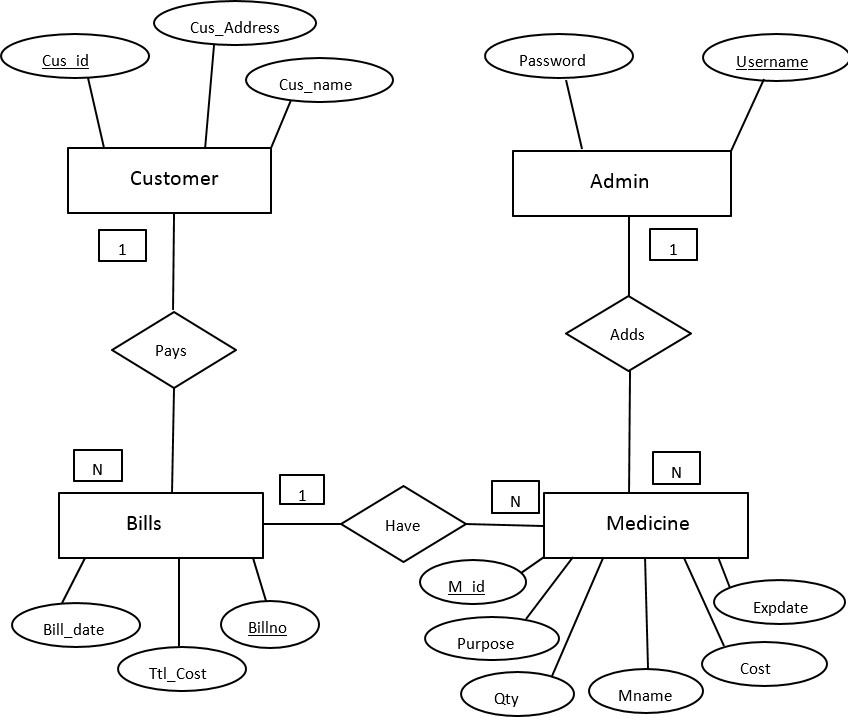
# CHAPTER 3

## SYSTEM DESIGN

System design is the process of defining any project in terms of architecture, modules, interfaces, and data of any system to satisfy the system requirements. It is a collection of methods or procedures used to construct a code.

#### ER Diagram

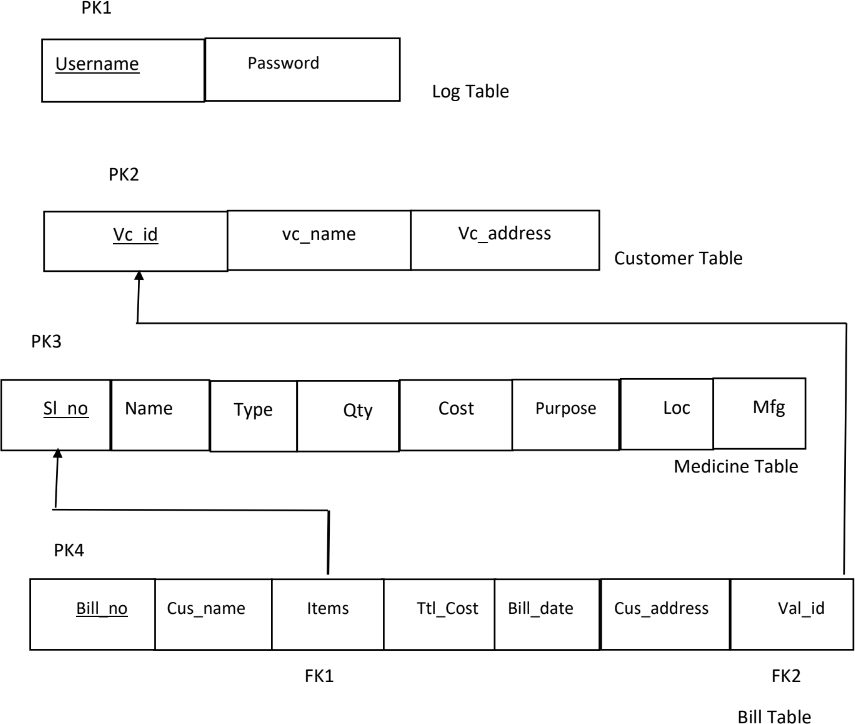
An ER diagram is entity relationship diagram which shows the relationships of entity sets stored in a database. An entity in this context is a component of data. In other words, ER diagrams illustrate the logical structure of databases.



**Fig: 3.1 ER diagram of the system**

* + - The above diagram consists of four entities: customer, bills, medicine and admin.
    - Each entity has a minimum of two attributes.
    - The entities are joined using relationships.

## Relational Model Concepts and Schema



**Fig: 3.2 Relational schema**

#### Algorithm

An algorithm is a step by step method of solving a problem. It is commonly used for data processing, calculation and other related computer and mathematical operations. An algorithm is also used to manipulate data in various ways, such as inserting a new data item, searching for a particular item or sorting an item.

Step1: Start Step2: Login page

Step3: Enter username and password

Step4: If username and password matches go to step 5 Else go to step 2

Step5: stock maintenance dashboard Step6: add valuable costumer details

Step7: add stock or delete stock and check expiry Step8: search from medicine related to symptoms Step9: calculate bill for purchased items from customer Step10: Stop

#### Code and Implementation

In computer science, an implementation is often used to describe the interactions of elements in programming languages used in tech world. The code given below is the code that gives the output of this medicine database management system.

from tkinter import \* import time

import sqlite3 import random import tempfile import win32api import win32print

f=''

flag='' flags=''

login=sqlite3.connect("admin.db") l=login.cursor()

c=sqlite3.connect("medicine.db") cur=c.cursor()

columns=('Sl No', 'Name', 'Type', 'Quantity Left', 'Cost', 'Purpose', 'Expiry Date', 'Rack location', 'Manufacture')

def open\_win(): #OPENS MAIN MENU----------------------------------------------------------------------------

MAIN MENU

global apt, flag flag='apt' apt=Tk()

apt.title("Interface")

Label(apt, text="CHEMIST AND DRUGSHOP").grid(row=0,column=0,columnspan=3) Label(apt, text='\*'\*80).grid(row=1,column=0,columnspan=3)

Label(apt, text='-'\*80).grid(row=3,column=0,columnspan=3)

Label(apt, text="Stock Maintenance").grid(row=2,column=0) Button(apt,text='New V.C.', width=25, command=val\_cus).grid(row=4,column=0)

Button(apt,text='Add product to Stock', width=25,command=stock).grid(row=5,column=0) Button(apt,text='Delete product from Stock',

width=25,command=delete\_stock).grid(row=6,column=0)

Label(apt, text="Access Database").grid(row=2,column=1) Button(apt,text='Modify',width=15, command=modify).grid(row=4,column=1)

Button(apt,text='Search', width=15, command=search).grid(row=5,column=1) Button(apt,text='Expiry Check', width=15, command=exp\_date).grid(row=6,column=1)

Label(apt, text="Handle Cash Flows").grid(row=2,column=2) Button(apt,text="Check Today's Revenue",

width=20,command=show\_rev).grid(row=5,column=2) Button(apt,text='Billing', width=20, command=billing).grid(row=4,column=2)

Label(apt, text='-'\*80).grid(row=12,column=0,columnspan=3) Button(apt,text='Logout',command=again).grid(row=13, column=2) apt.mainloop()

def delete\_stock(): #OPENS DELETE WINDOW DELETES A

PARTICULAR STOCK ITEM

global cur, c, flag, lb1, d apt.destroy()

flag='d' d=Tk()

d.title("Delete a product from Stock")

Label(d,text='Enter Product to delete:').grid(row=0,column=0) Label(d,text='',width=30,bg='white').grid(row=0,column=1) Label(d,text='Product').grid(row=2,column=0)

Label(d,text='Qty. Exp.dt. Cost ').grid(row=2,column=1) ren() b=Button(d,width=20,text='Delete',command=delt).grid(row=0,column=3)

b=Button(d,width=20,text='Main Menu',command=main\_menu).grid(row=5,column=3) d.mainloop()

def ren():

global lb1,d,cur,c def onvsb(\*args):

lb1.yview(\*args) lb2.yview(\*args)

def onmousewheel(): lb1.ywiew=('scroll',event.delta,'units') lb2.ywiew=('scroll',event.delta,'units') return 'break'

cx=0 vsb=Scrollbar(orient='vertical',command=onvsb) lb1=Listbox(d,width=25, yscrollcommand=vsb.set) lb2=Listbox(d,width=30,yscrollcommand=vsb.set) vsb.grid(row=3,column=2,sticky=N+S)

lb1.grid(row=3,column=0) lb2.grid(row=3,column=1) lb1.bind('<MouseWheel>',onmousewheel) lb2.bind('<MouseWheel>',onmousewheel) cur.execute("select \*from med")

for i in cur: cx+=1

s1=[str(i[0]),str(i[1])] s2=[str(i[3]),str(i[6]),str(i[4])] lb1.insert(cx,'. '.join(s1))

lb2.insert(cx,' '.join(s2)) c.commit() lb1.bind('<<ListboxSelect>>', sel\_del)

def sel\_del(e):

global lb1, d, cur, c,p, sl2 p=lb1.curselection() print (p)

x=0 sl2=''

cur.execute("select \* from med") for i in cur:

print (x, p[0])

if x==int(p[0]): sl2=i[0] break

x+=1

c.commit() print (sl2)

Label(d,text=' ',bg='white', width=20).grid(row=0,column=1) cur.execute('Select \* from med')

for i in cur:

if i[0]==sl2:

Label(d,text=i[0]+'. '+i[1],bg='white').grid(row=0,column=1) c.commit()

def delt():

global p,c,cur,d

cur.execute("delete from med where sl\_no=?",(sl2,)) c.commit()

ren()

def modify(): # window for modification-----------------------------------------------------------------------

MODIFY

global cur, c, accept, flag, att, up, n, name\_, apt, st, col,col\_n col=('', '', 'type', 'qty\_left', 'cost', 'purpose', 'expdt', 'loc', 'mfg')

col\_n=('', '', 'Type', 'Quantity Left', 'Cost', 'Purpose', 'Expiry Date', 'Rack location', 'Manufacture')

flag='st' name\_='' apt.destroy() n=[]

cur.execute("select \* from med") for i in cur:

n.append(i[1]) c.commit() st=Tk() st.title('MODIFY')

Label(st, text='-'\*48+' MODIFY DATABASE '+'-'\*48).grid(row=0, column=0,columnspan=6) def onvsb(\*args):

name\_.yview(\*args) def onmousewheel():

name\_.ywiew=('scroll',event.delta,'units') return 'break'

cx=0 vsb=Scrollbar(orient='vertical',command=onvsb) vsb.grid(row=1,column=3,sticky=N+S)

name\_=Listbox(st,width=43,yscrollcommand=vsb.set) cur.execute("select \*from med")

for i in cur: cx+=1

name\_.insert(cx,(str(i[0])+'. '+str(i[1]))) name\_.grid(row=1,column=1,columnspan=2)

c.commit() name\_.bind('<MouseWheel>',onmousewheel) name\_.bind('<<ListboxSelect>>', sel\_mn)

Label(st, text='Enter Medicine Name: ').grid(row=1, column=0) Label(st, text='Enter changed Value of: ').grid(row=2, column=0) att=Spinbox(st, values=col\_n)

att.grid(row=2, column=1) up=Entry(st) up.grid(row=2, column=2)

Button(st,width=10,text='Submit', command=save\_mod).grid(row=2, column=4)

Button(st,width=10,text='Reset', command=res).grid(row=2, column=5) Button(st,width=10,text='Show data', command=show\_val).grid(row=1, column=4) Label(st, text='-'\*120).grid(row=3,column=0,columnspan=6) Button(st,width=10,text='Main Menu',command=main\_menu).grid(row=5,column=5) st.mainloop()

def res(): global st, up up=Entry(st)

up.grid(row=2, column=2)

Label(st,width=20, text=' ').grid(row=5,column=i)

def sel\_mn(e):

global n,name\_, name\_mn, sl, c, cur name\_mn='' p=name\_.curselection()

print (p) x=0

sl=''

cur.execute("select \* from med") for i in cur:

print (x, p[0])

if x==int(p[0]): sl=i[0] break

x+=1

c.commit() print (sl)

name\_nm=n[int(sl)] print (name\_nm)

def show\_val():

global st, name\_mn, att, cur, c, col, col\_n, sl for i in range(3):

Label(st,width=20, text=' ').grid(row=5,column=i) cur.execute("select \* from med")

for i in cur:

for j in range(9):

if att.get()==col\_n[j] and sl==i[0]:

Label(st, text=str(i[0])).grid(row=5,column=0) Label(st, text=str(i[1])).grid(row=5,column=1) Label(st, text=str(i[j])).grid(row=5,column=2)

c.commit()

def save\_mod(): #save modified data

global cur, c, att, name\_mn, st, up, col\_n, sl for i in range(9):

if att.get()==col\_n[i]: a=col[i]

sql="update med set '%s' = '%s' where sl\_no = '%s'" % (a,up.get(),sl) cur.execute(sql)

c.commit()

Label(st, text='Updated!').grid(row=5,column=4)

def stock(): #add to stock window ADD

TO STOCK

global cur, c, columns, accept, flag, sto, apt apt.destroy()

flag='sto' accept=['']\*10 sto=Tk()

sto.title('STOCK ENTRY')

Label(sto,text='ENTER NEW PRODUCT DATA TO THE

STOCK').grid(row=0,column=0,columnspan=2) Label(sto,text='-'\*50).grid(row=1,column=0,columnspan=2) for i in range(1,len(columns)):

Label(sto,width=15,text=' '\*(14- len(str(columns[i])))+str(columns[i])+':').grid(row=i+2,column=0)

accept[i]=Entry(sto) accept[i].grid(row=i+2, column=1)

Button(sto,width=15,text='Submit',command=submit).grid(row=12,column=1) Label(sto,text='-'\*165).grid(row=13,column=0,columnspan=7) Button(sto,width=15,text='Reset',command=reset).grid(row=12,column=0) Button(sto,width=15,text='Refresh stock',command=ref).grid(row=12,column=4) for i in range(1,6):

Label(sto,text=columns[i]).grid(row=14,column=i-1)

Label(sto,text='Exp Rack Manufacturer ').grid(row=14,column=5) Button(sto,width=10,text='Main Menu',command=main\_menu).grid(row=12,column=5) ref()

sto.mainloop()

def ref(): # creates a multi-listbox manually to show the whole database global sto, c, cur

def onvsb(\*args): lb1.yview(\*args) lb2.yview(\*args) lb3.yview(\*args) lb4.yview(\*args) lb5.yview(\*args) lb6.yview(\*args)

def onmousewheel(): lb1.ywiew=('scroll',event.delta,'units') lb2.ywiew=('scroll',event.delta,'units') lb3.ywiew=('scroll',event.delta,'units') lb4.ywiew=('scroll',event.delta,'units') lb5.ywiew=('scroll',event.delta,'units') lb6.ywiew=('scroll',event.delta,'units')

return 'break' cx=0

vsb=Scrollbar(orient='vertical',command=onvsb) lb1=Listbox(sto,yscrollcommand=vsb.set) lb2=Listbox(sto,yscrollcommand=vsb.set) lb3=Listbox(sto,yscrollcommand=vsb.set,width=10) lb4=Listbox(sto,yscrollcommand=vsb.set,width=7) lb5=Listbox(sto,yscrollcommand=vsb.set,width=25) lb6=Listbox(sto,yscrollcommand=vsb.set,width=37) vsb.grid(row=15,column=6,sticky=N+S) lb1.grid(row=15,column=0) lb2.grid(row=15,column=1) lb3.grid(row=15,column=2) lb4.grid(row=15,column=3) lb5.grid(row=15,column=4) lb6.grid(row=15,column=5) lb1.bind('<MouseWheel>',onmousewheel) lb2.bind('<MouseWheel>',onmousewheel) lb3.bind('<MouseWheel>',onmousewheel) lb4.bind('<MouseWheel>',onmousewheel) lb5.bind('<MouseWheel>',onmousewheel) lb6.bind('<MouseWheel>',onmousewheel) cur.execute("select \*from med")

for i in cur: cx+=1

seq=(str(i[0]),str(i[1]))

lb1.insert(cx,'. '.join(seq)) lb2.insert(cx,i[2])

lb3.insert(cx,i[3])

lb4.insert(cx,i[4])

lb5.insert(cx,i[5]) lb6.insert(cx,i[6]+' '+i[7]+' '+i[8])

c.commit()

def reset():

global sto, accept

for i in range(1,len(columns)): Label(sto,width=15,text=' '\*(14-

len(str(columns[i])))+str(columns[i])+':').grid(row=i+2,column=0) accept[i]=Entry(sto)

accept[i].grid(row=i+2, column=1)

def submit(): #for new stock submission global accept, c, cur, columns, sto prev=time.clock()

x=['']\*10

cur.execute("select \* from med") for i in cur:

y=int(i[0])

for i in range(1,9): x[i]=accept[i].get()

sql="insert into med values('%s','%s','%s','%s','%s','%s','%s','%s','%s')" %

(y+1,x[1],x[2],x[3],x[4],x[5],x[6],x[7],x[8])

cur.execute(sql) cur.execute("select \* from med") c.commit()

now=time.clock() print (now-prev) top=Tk()

Label(top,width=20, text='Success!').pack() top.mainloop()

main\_menu()

def chk(): #checks if the medicine is already present so that can be modified global cur, c, accept, sto

cur.execute("select \* from med") for i in cur:

if accept[6].get()==i[6] and i[1]==accept[1].get():

sql="update med set qty\_left = '%s' where name = '%s'" % (str(int(i[3])+int(accept[3].get())),accept[1].get())

cur.execute(sql) c.commit() top=Tk()

Label(top,width=20, text='Modified!').pack() top.mainloop()

main\_menu() else:

submit() c.commit()

def exp\_date(): # expiry window open-----------------------------------------------------------------------------

EXPIRY

global exp, s,c, cur, flag, apt, flags apt.destroy()

flag='exp'

from datetime import date now=time.localtime()

n=[]

cur.execute("select \*from med") for i in cur:

n.append(i[1]) c.commit() exp=Tk()

exp.title('EXPIRY CHECK')

Label(exp,text='Today : '+str(now[2])+'/'+str(now[1])+'/'+str(now[0])).grid(row=0, column=0, columnspan=3)

Label(exp,text='Selling Expired Medicines and Drugs is Illegal').grid(row=1, column=0,columnspan=3)

Label(exp,text='-'\*80).grid(row=2, column=0,columnspan=3) s=Spinbox(exp,values=n)

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

Button(exp,text='Check Expiry date', command=s\_exp).grid(row=3, column=1) Label(exp,text='-'\*80).grid(row=4, column=0,columnspan=3)

if flags=='apt1':

Button(exp,text='Main Menu', command=main\_cus).grid(row=5, column=2) else:

Button(exp,width=20,text='Check Products expiring', command=exp\_dt).grid(row=5, column=0)

Button(exp,text='Main Menu', command=main\_menu).grid(row=5, column=2) exp.mainloop()

def s\_exp(): # shows the expiry date of the medicine entered global c, cur, s, exp, top

from datetime import date now=time.localtime()

d1 = date(now[0],now[1],now[2]) cur.execute("select \* from med") for i in cur:

if(i[1]==s.get()): q=i[6]

d2=date(int('20'+q[8:10]),int(q[3:5]),int(q[0:2])) if d1>d2:

Label(exp, text='EXPIRED! on '+i[6]).grid(row=3, column=2) top=Tk()

Label(top, text='EXPIRED!').pack() else:

Label(exp, text=i[6]).grid(row=3, column=2) c.commit()

def exp\_dt(): # shows medicine to expire in the coming week global c, cur, exp, top

x=0 z=1

from datetime import datetime, timedelta N = 7

dt = datetime.now() + timedelta(days=N) d=str(dt)

from datetime import date now=time.localtime()

d1 = date(now[0],now[1],now[2])

d3 = date(int(d[0:4]),int(d[5:7]),int(d[8:10])) Label(exp,text='S.No'+' '+'Name'+' Qty.

'+'Exp\_date').grid(row=6,column=0,columnspan=2) cur.execute("select \* from med")

for i in cur: s=i[6]

d2=date(int('20'+s[8:10]),int(s[3:5]),int(s[0:2]))

if d1<d2<d3:

Label(exp,text=str(z)+'. '+str(i[1])+' '+str(i[3])+' '+str(i[6])).grid(row=x+7,column=0,columnspan=2)

x+=1

z+=1

elif d1>d2: top=Tk()

Label(top,width=20, text=str(i[1])+' is EXPIRED!').pack() c.commit()

def billing(): # to create bills for customer BILLING

system

global c, cur, apt, flag, t, name, name1, add, st, names, qty, sl, qtys, vc\_id, n, namee, lb1 t=0

vc\_id='' names=[] qty=[]

sl=[]

n=[]

qtys=['']\*10

cur.execute("select \*from med") for i in cur:

n.append(i[1]) c.commit()

if flag=='st': st.destroy()

else:

apt.destroy() flag='st'

st=Tk()

st.title('BILLING SYSTEM')

Label(st,text='-'\*48+'BILLING SYSTEM'+'-'\*49).grid(row=0,column=0,columnspan=7) Label(st,text='Enter Name: ').grid(row=1,column=0)

name1=Entry(st) name1.grid(row=1, column=1)

Label(st,text='Enter Address: ').grid(row=2,column=0) add=Entry(st)

add.grid(row=2, column=1)

Label(st,text="Value Id (if available)").grid(row=3, column=0) vc\_id=Entry(st)

vc\_id.grid(row=3, column=1)

Button(st,text='Check V.C.', command=blue).grid(row=4, column=0) Label(st,text='-'\*115).grid(row=6, column=0,columnspan=7) Label(st,text='SELECT PRODUCT',width=25,relief='ridge').grid(row=7, column=0)

Label(st,text=' RACK QTY LEFT COST ',width=25,relief='ridge').grid(row=7, column=1) Button(st,text='Add to bill',width=15,command=append2bill).grid(row=8, column=6)

Label(st,text='QUANTITY',width=20,relief='ridge').grid(row=7, column=5) qtys=Entry(st)

qtys.grid(row=8,column=5) refresh()

Button(st,width=15,text='Main Menu', command=main\_menu).grid(row=1,column=6) Button(st,width=15,text='Refresh Stock', command=refresh).grid(row=3,column=6) Button(st,width=15,text='Reset Bill', command=billing).grid(row=4,column=6) Button(st,width=15,text='Print Bill', command=print\_bill).grid(row=5,column=6) Button(st,width=15,text='Save Bill', command=make\_bill).grid(row=7,column=6)

st.mainloop() def refresh():

global cur, c, st, lb1, lb2, vsb def onvsb(\*args):

lb1.yview(\*args) lb2.yview(\*args)

def onmousewheel(): lb1.ywiew=('scroll',event.delta,'units') lb2.ywiew=('scroll',event.delta,'units') return 'break'

cx=0 vsb=Scrollbar(orient='vertical',command=onvsb) lb1=Listbox(st,width=25, yscrollcommand=vsb.set) lb2=Listbox(st ,width=25,yscrollcommand=vsb.set) vsb.grid(row=8,column=2,sticky=N+S) lb1.grid(row=8,column=0) lb2.grid(row=8,column=1) lb1.bind('<MouseWheel>',onmousewheel) lb2.bind('<MouseWheel>',onmousewheel) cur.execute("select \*from med")

for i in cur: cx+=1

lb1.insert(cx,str(i[0])+'. '+str(i[1]))

lb2.insert(cx,' '+str(i[7])+' '+str(i[3])+' Rs '+str(i[4])) c.commit()

lb1.bind('<<ListboxSelect>>', select\_mn)

def select\_mn(e): #store the selected medicine from listbox global st, lb1, n ,p, nm, sl1

p=lb1.curselection()

x=0 sl1=''

from datetime import date now=time.localtime()

d1 = date(now[0],now[1],now[2]) cur.execute("select \* from med") for i in cur:

if x==int(p[0]):

sl1=int(i[0]) break

x+=1

c.commit() print (sl1) nm=n[x] print (nm)

def append2bill(): # append to the bill global st, names, nm , qty, sl,cur, c, sl1 sl.append(sl1)

names.append(nm) qty.append(qtys.get()) print (qty)

print (sl[len(sl)-1],names[len(names)-1],qty[len(qty)-1])

def blue(): # check if valued customer global st ,c, cur, named, addd, t, vc\_id cur.execute("select \* from cus")

for i in cur:

if vc\_id.get()!='' and int(vc\_id.get())==i[2]: named=i[0]

addd=i[1]

Label(st,text=named,width=20).grid(row=1, column=1) Label(st,text=addd,width=20).grid(row=2, column=1) Label(st,text=i[2],width=20).grid(row=3, column=1) Label(st, text='Valued Customer!').grid(row=4, column=1) t=1

break c.commit()

def make\_bill(): # makes bill

global t, c, B, cur, st, names, qty, sl , named, addd, name1, add,det, vc\_id price=[0.0]\*10

q=0 det=['','','','','','','','']

det[2]=str(sl)

for i in range(len(sl)):

print (sl[i],' ',qty[i],' ',names[i]) for k in range(len(sl)):

cur.execute("select \* from med where sl\_no=?",(sl[k],)) for i in cur:

price[k]=int(qty[k])\*float(i[4]) print (qty[k],price[k])

cur.execute("update med set qty\_left=? where sl\_no=?",(int(i[3])-int(qty[k]),sl[k])) c.commit()

det[5]=str(random.randint(100,999)) B='bill\_'+str(det[5])+'.txt'

total=0.00

for i in range(10): if price[i] != '':

total+=price[i] #totalling m='\n\n\n'

m+="===============================================\n"

m+=" No :%s\n\n" % det[5] m+=" MEDPLUS CHEMIST AND DRUGGIST\n"

m+=" VIT University, Katpadi, Vellore, T.M.\n\n" m+=" \n"

if t==1:

m+="Name: %s\n" % named m+="Address: %s\n" % addd det[0]=named

det[1]=addd cur.execute('select \* from cus') for i in cur:

if i[0]==named: det[7]=i[2]

else:

m+="Name: %s\n" % name1.get() m+="Address: %s\n" % add.get() det[0]=name1.get() det[1]=add.get()

m+=" \n"

m+="Product Qty. Price\n"

m+=" \n"#47, qty=27, price=8 after 2

for i in range(len(sl)):

if names[i] != 'nil': s1=' '

s1=(names[i]) + (s1 \* (27-len(names[i]))) + s1\*(3-len(qty[i])) +qty[i]+ s1\*(15- len(str(price[i])))+str(price[i]) + '\n'

m+=s1

m+="\n \n"

if t==1:

ntotal=total\*0.8

m+='Total'+(' '\*25)+(' '\*(15-len(str(total)))) + str(total)+'\n'

m+="Valued customer Discount"+ (' '\*(20-len(str(total-ntotal))))+'-'+str(total-ntotal)+'\n' m+=" \n"

m+='Total'+(' '\*25)+(' '\*(12-len(str(ntotal)))) +'Rs '+ str(ntotal)+'\n' det[3]=str(ntotal)

else:

m+='Total'+(' '\*25)+(' '\*(12-len(str(total)))) +'Rs '+ str(total)+'\n' det[3]=str(total)

m+=" \n\n"

m+="Dealer 's signature: \n" m+="===============================================\n"

print (m) p=time.localtime()

det[4]=str(p[2])+'/'+str(p[1])+'/'+str(p[0]) det[6]=m

bill=open(B,'w') bill.write(m) bill.close()

cb=('cus\_name','cus\_add','items','Total\_cost','bill\_dt','bill\_no','bill','val\_id') cur.execute('insert into bills

values(?,?,?,?,?,?,?,?)',(det[0],det[1],det[2],det[3],det[4],det[5],det[6],det[7])) c.commit()

def print\_bill():

win32api.ShellExecute (0,"print",B,'/d:"%s"' % win32print.GetDefaultPrinter (),".",0)

def show\_rev(): # opens revenue window-----------------------------------------------------------------------

TOTAL REVENUE

global c, cur, flag,rev apt.destroy()

cb=('cus\_name','cus\_add','items','Total\_cost','bill\_dt','bill\_no','bill','val\_id') flag='rev'

rev=Tk()

total=0.0 today=str(time.localtime()[2])+'/'+str(time.localtime()[1])+'/'+str(time.localtime()[0]) Label(rev,text='Today: '+today).grid(row=0,column=0)

cur.execute('select \* from bills') for i in cur:

if i[4]==today: total+=float(i[3])

print (total)

Label(rev,width=22,text='Total revenue: Rs '+str(total), bg='black',fg='white').grid(row=1,column=0)

cx=0 vsb=Scrollbar(orient='vertical')

lb1=Listbox(rev,width=25, yscrollcommand=vsb.set) vsb.grid(row=2,column=1,sticky=N+S) lb1.grid(row=2,column=0)

vsb.config( command = lb1.yview ) cur.execute("select \* from bills") for i in cur:

if i[4]==today: cx+=1

lb1.insert(cx,'Bill No.: '+str(i[5])+' : Rs '+str(i[3])) c.commit()

Button(rev,text='Main Menu',command=main\_menu).grid(row=15,column=0) rev.mainloop()

def search(): #search window medicine and symptom details SEARCH

MEDICINE RACK & SYMPTOMS

global c, cur, flag, st, mn, sym, flags flag='st'

apt.destroy()

cur.execute("Select \* from med") symp=['nil']

med\_name=['nil'] for i in cur:

symp.append(i[5]) med\_name.append(i[1])

st=Tk() st.title('SEARCH')

Label(st, text=' SEARCH FOR MEDICINE ').grid(row=0, column=0,columnspan=3) Label(st, text='~'\*40).grid(row=1, column=0,columnspan=3)

Label(st, text='Symptom Name').grid(row=3, column=0)

sym=Spinbox(st,values=symp) sym.grid(row=3, column=1)

Button(st,width=15, text='Search', command=search\_med).grid(row=3, column=2) Label(st, text='-'\*70).grid(row=4, column=0,columnspan=3)

if flags=='apt1':

Button(st,width=15, text='Main Menu', command=main\_cus).grid(row=6, column=2) else:

Button(st,width=15, text='Main Menu', command=main\_menu).grid(row=6, column=2) st.mainloop()

def search\_med():

global c, cur, st, sym, columns cur.execute("select \* from med") y=[]

x=0

for i in cur:

if i[5]==sym.get():

y.append(str(i[0])+'. '+str(i[1])+' Rs '+str(i[4])+' Rack : '+str(i[7])+' Mfg : '+str(i[8])) x=x+1

top=Tk()

for i in range(len(y)): Label(top,text=y[i]).grid(row=i, column=0)

Button(top,text='OK',command=top.destroy).grid(row=5, column=0) c.commit()

top.mainloop()

def val\_cus(): #to enter new valued customer NEW

VALUED CUSTOMER

global val, flag, dbt, name\_vc, add\_vc, cur, c, vc\_id apt.destroy()

cur.execute("select \* from cus") flag='val'

val=Tk()

Label(val,text="ENTER VALUED CUSTOMER DETAILS").grid(row=0,column=0,columnspan=3) Label(val,text="-"\*60).grid(row=1,column=0,columnspan=3)

Label(val,text="Name: ").grid(row=2,column=0) name\_vc=Entry(val)

name\_vc.grid(row=2, column=1) Label(val,text="Address: ").grid(row=3,column=0) add\_vc=Entry(val)

add\_vc.grid(row=3, column=1) Label(val,text="Value Id: ").grid(row=4,column=0)

vc\_id=Entry(val) vc\_id.grid(row=4, column=1)

Button(val,text='Submit',command=val\_get).grid(row=5, column=1) Button(val,text='Main Menu',command=main\_menu).grid(row=5, column=2) Label(val,text='-'\*60).grid(row=6,column=0,columnspan=3)

val.mainloop()

def val\_get(): #to submit new valued customer details global name\_vc, add\_vc, val, dbt ,c, cur, apt, vc\_id

cur.execute("insert into cus values(?,?,?)",(name\_vc.get(),add\_vc.get(),vc\_id.get()))

* 1. xecute("insert into log values(?,?)",(name\_vc.get(),vc\_id.get())) cur.execute("select \* from cus")

for i in cur:

print (i[0], i[1], i[2]) c.commit() login.commit()

def again(): #for login window-----------------------------------------------------------------------------

LOGIN WINDOW

global un, pwd, flag, root, apt if flag=='apt':

apt.destroy() root=Tk()

root.title('YashVone SOFTWARE SOLUTIONS')

Label(root,text='CHEMIST AND DRUG SHOP').grid(row=0,column=0,columnspan=5) Label(root,text="VIT UNIVERSITY, KATPADI, VELLORE,

TM").grid(row=1,column=0,columnspan=5) Label(root,text='-------------------------------------------------------

').grid(row=2,column=0,columnspan=5)

Label(root, text='Username').grid(row=3, column=0) un=Entry(root,width=10)

un.grid(row=3, column=1)

Label(root, text='Password').grid(row=4, column=0) pwd=Entry(root,width=10)

pwd.grid(row=4, column=1) Button(root,width=6,text='Enter',command=check).grid(row=5, column=0) Button(root,width=6,text='Close',command=root.destroy).grid(row=5, column=1) root.mainloop()

def check(): #for enter button in login window global un, pwd, login, l, root

u=un.get()

p=pwd.get() l.execute("select \* from log") for i in l:

if i[0]==u and i[1]==p and u=='admin': root.destroy()

open\_win()

elif i[0]==u and i[1]==p: root.destroy() open\_cus()

login.commit()

def main\_menu(): #controls open and close of main menu window---------------------------------------

-RETURN TO MAIN MENU

global sto, apt, flag, root, st, val, exp, st1,rev if flag=='sto':

sto.destroy() if flag=='rev':

rev.destroy() elif flag=='st':

st.destroy() elif flag=='st1':

st1.destroy() elif flag=='val': val.destroy() elif flag=='exp': exp.destroy()

elif flag=='d': d.destroy()

open\_win()

def main\_cus(): global st, flag, exp if flag=='exp':

exp.destroy() elif flag=='st':

st.destroy() open\_cus()

def open\_cus(): #OPENS MAIN MENU----------------------------------------------------------------------------

MAIN MENU

global apt, flag, flags flags='apt1'

apt=Tk() apt.title("Interface")

Label(apt, text="MEDPLUS CHEMIST AND DRUGGIST").grid(row=0,column=0) Label(apt, text='\*'\*40).grid(row=1,column=0)

Label(apt, text='\* WELCOME \*').grid(row=2,column=0) Label(apt, text='-'\*40).grid(row=3,column=0)

Label(apt, text="Customer Services").grid(row=4,column=0) Label(apt, text='-'\*40).grid(row=5,column=0)

Button(apt,text='Search', width=15, command=search).grid(row=6,column=0) Button(apt,text='Expiry Check', width=15, command=exp\_date).grid(row=7,column=0)

Label(apt, text='-'\*40).grid(row=8,column=0) Button(apt,text='Logout',command=again1).grid(row=9, column=0) apt.mainloop()

def again1(): global flags apt.destroy() flags='' again()

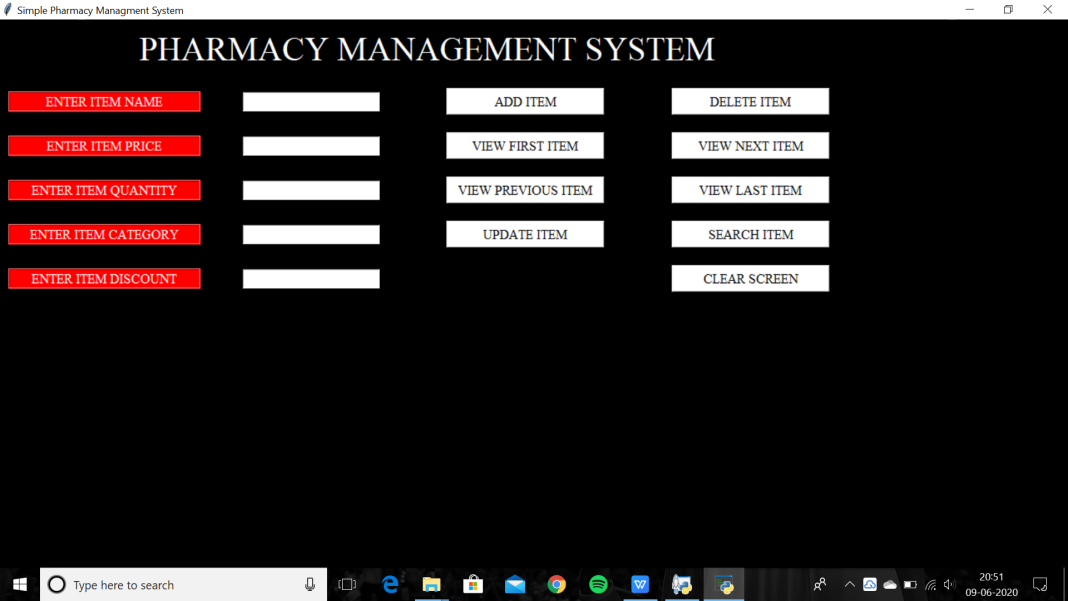
again()

# CHAPTER 4

## Results and Discussion

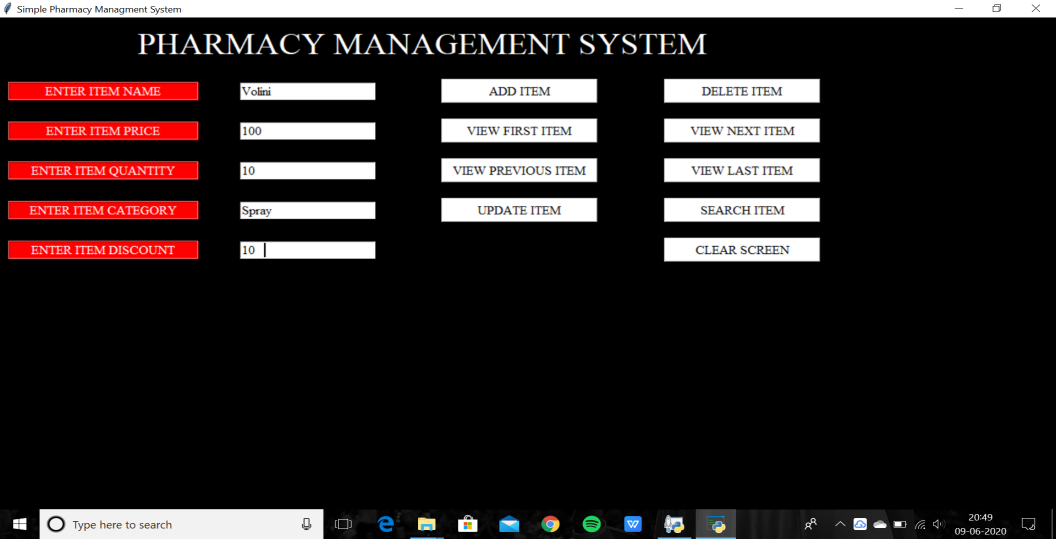
Results provides the outcome of the project wherein the result or the data from the project is described in the discussion. Tables and figures appear here. It helps in process and analysis of the project.

#### Output(snapshots)

****

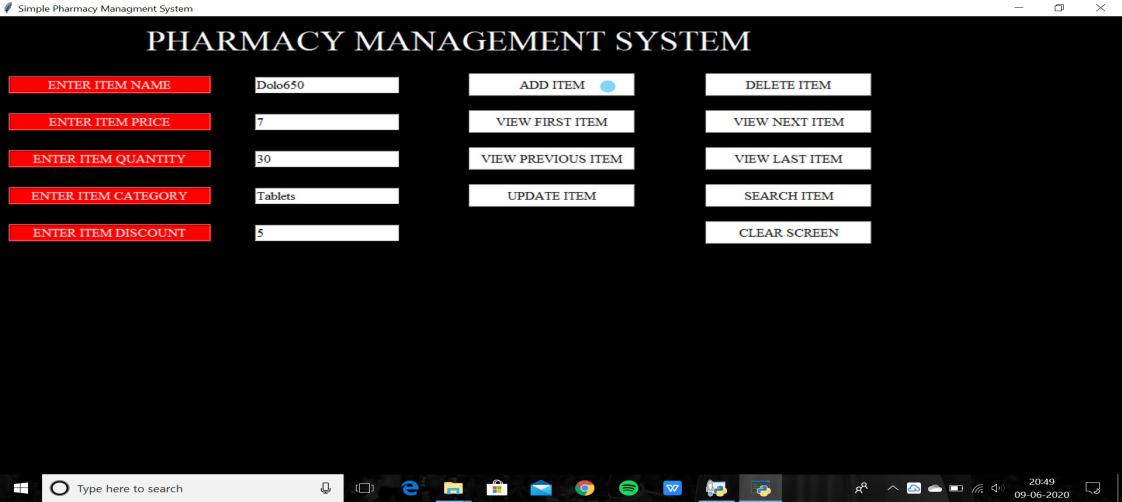
**Fig: 4.1.1 First Page**

* + - Only admin can authorize the login page



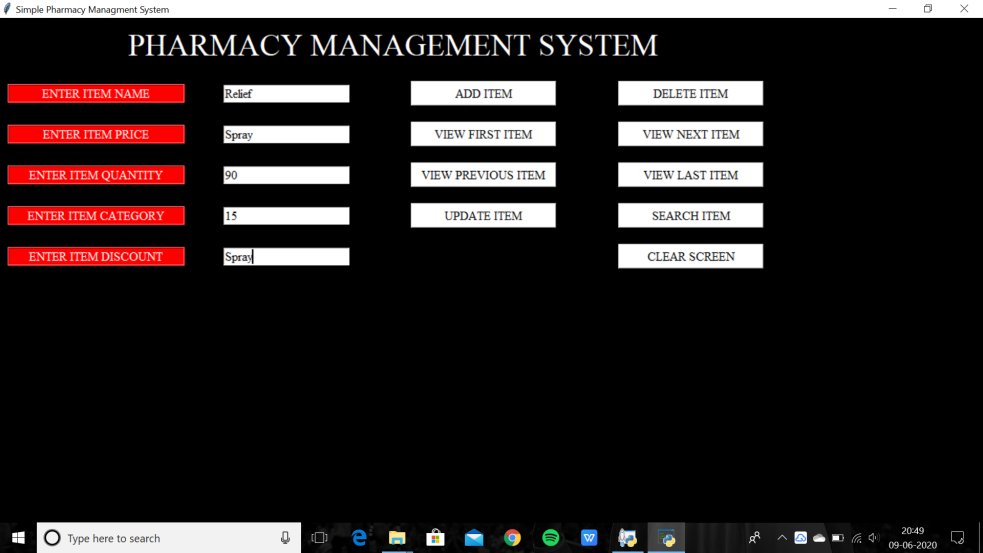
**Fig: 4.1.2 Entering details**

* + - Here owner can access the database of stock and handle cash flows



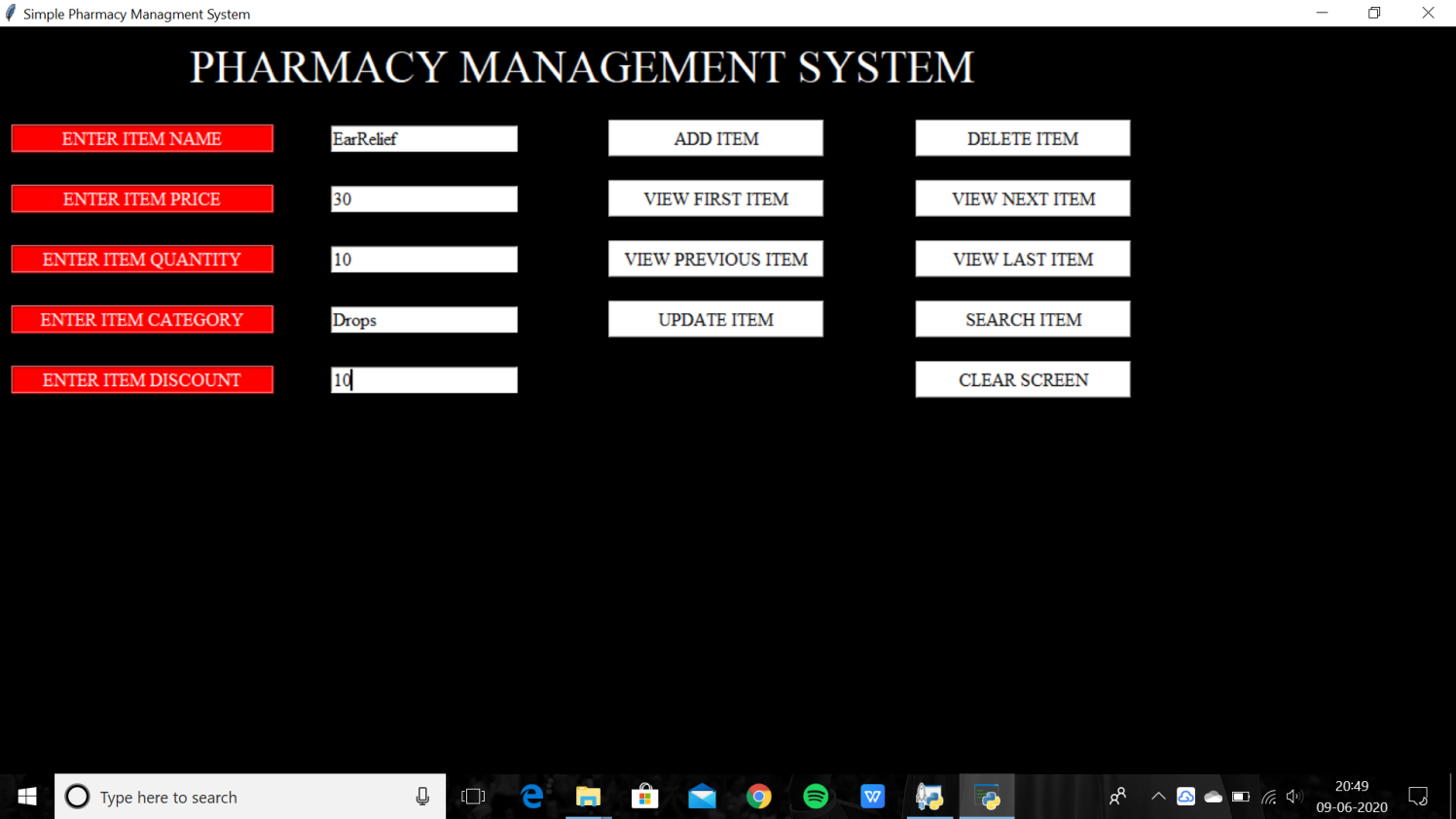
**Fig: 4.1.3 Adding Item**

* + - Entering details of valued customer who are eligible for discounts



**Fig: 4.1.4 Viewing Next Item**

* + - Valuable customers who are eligible for discounts



**Fig: 4.1.5 Viewing Last Item**

* + - The owner can modify the stock database.



**Fig: 4.1.6 Clearing the screen**

# CHAPTER 5

## CONCLUSION

The main objective of Medicare is to provide customers with quality medicines. With increasing health concerns among people demands for efficient medical stores. The project as a whole function in achieving this by providing customers with required medicines. The Medicare takes care of handling the stock by adding new medicines, modifying, checking for expiry of medicines and keep track of the medicines being sold for different diseases purchased by the customers.

The interface developed meets up to the requirements of the owner to achieve profit also to satisfy the needs of the customer, the software developed handles cash flows, billing and the total revenue collected for a day. The software also saves time and money.

The future scope of the project can be extended by adding other functionalities like plotting graphs for most selling medicines and the symptoms relating to viral diseases, detect medicines with fake ingredients and high dosages leading to side effects.

## References

1. Author - John zelle

“Python programming (2nd edition,2011)”

1. Author -Alan Beaulieu

“Learning SQL(Kindle edition,2014)”

1. Author - Mark lutz and david Asher

“learning python (4th edition,2010)”

Websites used:

* + <https://www.tutorialsspoint.com/>
  + <https://www.opensource.com/>
  + https:/[/www.geeksforgeeks.org/c-plus-plus](http://www.geeksforgeeks.org/c-plus-plus)