ACKNOWLEDGEMENT

We consider it a privilege to whole-heartedly express my gratitude and respect to each and every one who guided and helped me in the successful completion of this Project Report.

We are very thankful to the principal Dr G PUNDARIKA for being kind enough to provide me an opportunity to work on a Project in this institution.

We are also thankful to Prof. VASANTH G, HOD, Department of Computer Science, for her co-operation and encouragement at all moments of my approach.

We would greatly mention the enthusiastic influence provided by Dr. Shabeen Taj G A, Associate Professor, as our Project Guide, for her ideas and cooperation showed on us during my venture and making this Project a great success.

We would also like to thank my parents and well-wishers as well as my dear classmates for their guidance and their kind co-operation.

Finally, it is my pleasure and happiness to the friendly co-operation showed by all the staff members of Computer Science Department,

INDEX

CHAPTER 1

ABSTRACT

The main aim of the project is the management of the database of the pharmaceutical shop. This project is insight into the design and implementation of a Pharmacy Management System. One of the most important responsibilities of pharmacy management is to supervise and manage the pharmacy employees in order to ensure healthy working relationships and outcomes. Each of these functions is critical to the pharmacy's operation and should be explained by the management. This is done by creating a database of the available medicines in the shop. The primary aim of pharmacy management system is to improve accuracy and enhance safety and efficiency in the pharmaceutical store. The aim of this project is to develop software for the effective management of a pharmaceutical store. We have developed this software for ensuring effective policing by providing statistics of the drugs in stock.

CHAPTER 2

INTRODUCTION

Description on topic:

This program can be used in any pharmaceutical shops having a database to maintain. The software used can generate reports, as per the user's requirements. The software can print invoices, bills, receipts etc. It can also maintain the record of supplies sent in by the supplier. Here, the admin who are handling the organization will be responsible to manage the record of the employee. Each employee will be given with a separate username and password.

Program definition:

The aim of the project is to create an effective software to help the pharmacist to maintain the records of the medicines, handle user details, generate invoice, check and renew validity and provide a scope of communication between users by using inbuilt messaging system. Pharmacy management system deals with the maintenance of drugs and consumables in the pharmacy unit. This pharmacy management system is user friendly.

CHAPTER 3

SYSTEM REQUIREMENTS

Hardware and software tools:

The system service and goals are established by consultation with system user. They are then defined in details and serve as a system specification . System requirement are those on which the system runs.

Hardware Requirements:

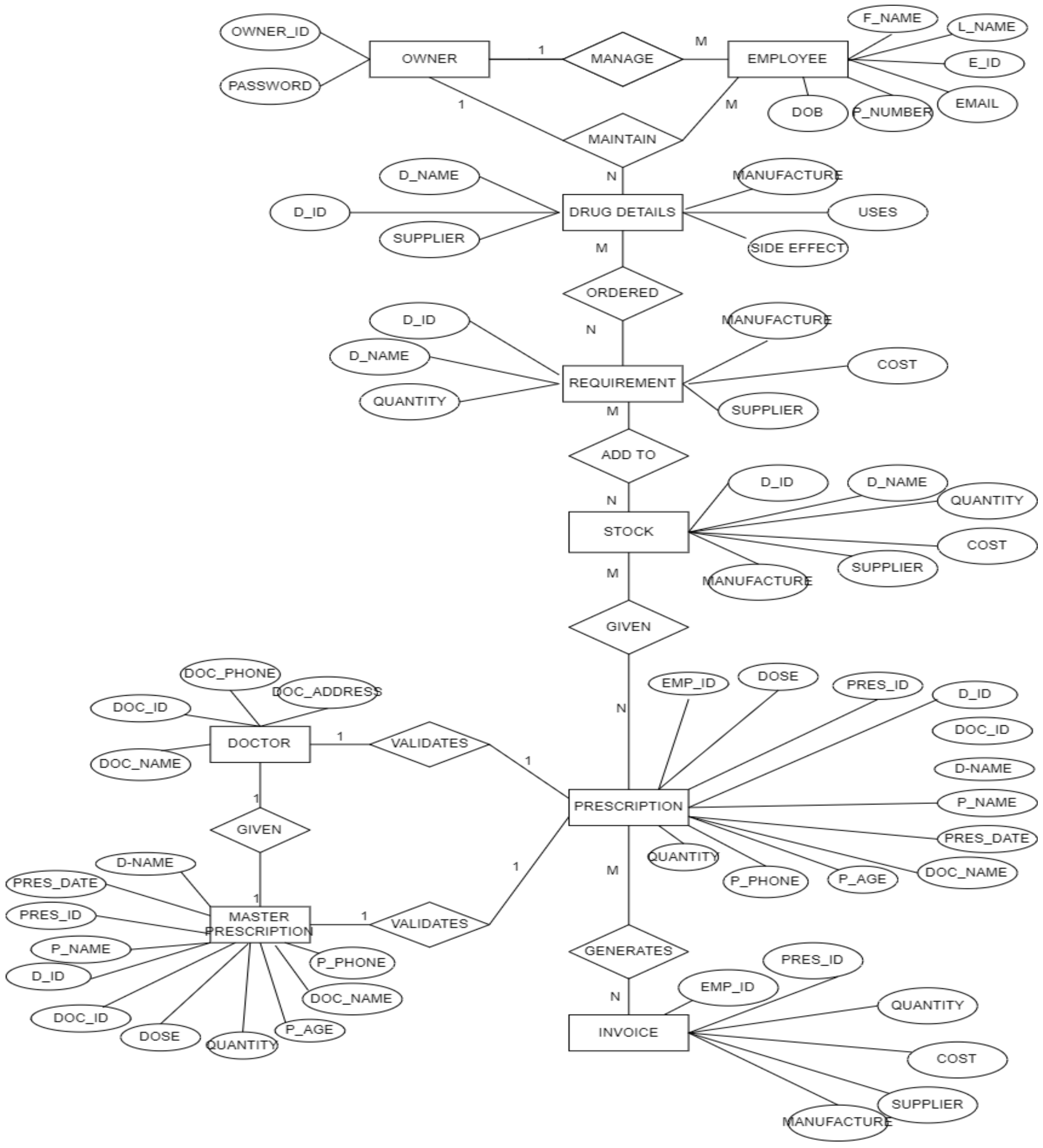
- Computer with either Intel core or AMD processor.
- 1GB +DDR RAM.
- 40GB HARD DISK DRIVE.

Software Requirements:

- Windows/macOS/Linux operating system .
- Flask,Html,Css.
- Mysql.

CHAPTER 4

ENTITY RELATIONSHIP DIAGRAM



CHAPTER **5**

IMPLEMENTATION

Description on Implementation

The goal this application is to manage the medicines and the various function of the pharmacy.

Modules:

Login page: Login page give the access for owner and employee. Login page used to login to the pharmacy database system. Login page is very common among any type of secures website it is used server for authentication .This help to secure the web pages and database.

Owner and Employee page: Owner page get all the access of the tables. But the employee page access only particular access given by the owner.

Drug page: Drug page contain the drugs information which are recommended for particular diseases. It must be best medicine for patient and fast recovery the health.

Requirement page: Requirement page contain the required medicine which are mentioned in drug page. That medicines can give the order by owner.

Prescription page: The prescription page having all the details of patient ,drug, and employee .

Invoice page: Invoice page is the reference for patient, employee and owner of the pharmacy. It contains detailed information of the amount, prescription, patient , employee details.

Stock page: The stock page contains quantity of medicine are available in stock.

Master prescription page : The page contains the doctor prescription that would be match to the patient prescription while give the medicine in pharmacy.

Doctor page : It show the prescription for patient given by doctor.

Source code:

1.Database initiate code:

```
  import mysql.connector as msc

db =msc.connect(user="root",passwd="root",auth_plugin='mysql_native_password')

  cursor=db.cursor()

  def initialize():

    cursor.execute('CREATE DATABASE PHARMACY_MANAGEMENT')

    cursor.execute('USE PHARMACY_MANAGEMENT')

    cursor.execute('CREATE TABLE OWNER(OWNER_ID INT(10) PRIMARY KEY,PASSWORD VARCHAR(15))')

    cursor.execute('CREATE TABLE EMPLOYEE(F_NAME VARCHAR(10) NOT NULL, L_NAME VARCHAR(10), EMP_ID INT PRIMARY KEY, EMAIL VARCHAR(20), PHONE BIGINT(10), DOB VARCHAR(10))')

    cursor.execute('CREATE TABLE DRUG(D_ID INT NOT NULL, D_NAME VARCHAR(50) NOT NULL, SUPPLIER VARCHAR(50) NOT NULL, MANUFACTURER VARCHAR(50) NOT NULL,USES VARCHAR(50) NOT NULL, SIDE_EFFECT VARCHAR(50) NOT NULL)')

    cursor.execute('CREATE TABLE REQUIREMENT(D_ID INT NOT NULL,D_NAME VARCHAR(50) NOT NULL, QUANTITY INT NOT NULL, COST FLOAT NOT NULL, SUPPLIER VARCHAR(100) NOT NULL, MANUFACTURER VARCHAR(100) NOT NULL)')

    cursor.execute('CREATE TABLE PRESCRIPTION(EMP_ID INT NOT NULL,PRES_ID INT NOT NULL, QUANTITY INT NOT NULL, DOSE INT NOT NULL,D_ID INT NOT NULL,D_NAME VARCHAR(50) NOT NULL, P_NAME VARCHAR(100) NOT
```

NULL,PRES_DATE VARCHAR(10) NOT NULL, P_AGE INT NOT NULL,P_PHONE BIGINT(10) NOT NULL)')

cursor.execute('CREATE TABLE INVOICE(EMP_ID INT NOT NULL,PRES_ID INT NOT NULL, QUANTITY INT NOT NULL, DOSE INT NOT NULL,D_ID INT NOT NULL,D_NAME VARCHAR(50) NOT NULL, P_NAME VARCHAR(100) NOT NULL,PRES_DATE VARCHAR(10) NOT NULL, P_AGE INT NOT NULL,P_PHONE BIGINT(10) NOT NULL,PRICE FLOAT NOT NULL)')

cursor.execute('CREATE TABLE STOCK(D_ID INT NOT NULL,D_NAME VARCHAR(50) NOT NULL, QUANTITY INT NOT NULL, COST FLOAT NOT NULL, SUPPLIER VARCHAR(100) NOT NULL, MANUFACTURER VARCHAR(100) NOT NULL)')

db.commit()

initialize()

Execution code:

from flask import Flask, render_template,request, redirect

import mysql.connector as msc

db= msc.connect(user="root",passwd="root",auth_plugin='mysql_native_password')

cursor=db.cursor(buffered=True)

cursor.execute("USE PHARMACY_MANAGEMENT")

app=Flask(__name__)

```
@app.route('/', methods=['POST','GET'])

def login():

    if request.method == 'POST':

     uid=request.form['username']

    pwd=request.form['password']

        try:

            cursor.execute('SELECT * FROM OWNER WHERE OWNER_ID=\'%s\''%uid)

            c=0

            for t in cursor

            c+=1;

                 if c!=0:

                return redirect('/owner')

            else:

  cursor.execute('SELECT * FROM EMPLOYEE WHERE       EMP_ID=\'%s\''%uid)

                c=0

                for t in cursor:

                    c+=1

                if c!=0:

                    return redirect('/employee')

                else:

                    return redirect('/')
        except:
            return redirect('/')
    else:
        return render_template('login.html')


@app.route('/owner',methods=['POST','GET'])
def owner():
    return render_template('owner.html')



@app.route('/employee',methods=['POST','GET'])
def employee():
    return render_template('employee.html')
@app.route('/addemp',methods=['POST','GET'])
def addemp():
    if request.method == 'POST':
        fname=request.form['fname']
        lname=request.form['lname']
        eid=request.form['eid']
        mail=request.form['mail']
        ph=request.form['ph']
        dob=request.form['dob']
        cursor.execute(r"INSERT INTO EMPLOYEE VALUES('%s','%s',%s,'%s','%s','%s')"%(fname,lname,eid,mail,ph,dob))
        db.commit()
        return redirect('/empo')
    else:
        return render_template('addemployee.html')
@app.route('/empo',methods=['POST','GET'])
def empo():
    cursor.execute('SELECT * FROM EMPLOYEE')
    tasks=cursor.fetchall()
    return render_template('viewemployeeo.html',tasks=tasks)
@app.route('/delempo/<int:eid>', methods=['POST', 'GET'])
def delempo(eid):
    cursor.execute(r"DELETE FROM EMPLOYEE WHERE EMP_ID=%d" % (eid))
    db.commit()
    return redirect('/empo')
@app.route('/upempo/<int:eid>',methods=['POST','GET'])
def upempo(eid):
    cursor.execute('SELECT * FROM EMPLOYEE WHERE EMP_ID=%d' % eid)
    tasks = cursor.fetchone()
    if request.method == 'POST':
        fname=request.form['fname']
        lname=request.form['lname']
        mail=request.form['mail']
        ph=request.form['ph']
        dob=request.form['dob']
        cursor.execute(r"UPDATE EMPLOYEE SET F_NAME='%s',L_NAME='%s',EMAIL='%s',PHONE='%s',DOB='%s' WHERE EMP_ID='%s'" %(fname,lname,mail,ph,dob,eid))
        db.commit()
        return redirect("/empo")
    else:
        return render_template('upemployee.html',tasks = tasks)


@app.route('/requiremento',methods=['POST','GET'])
def requiremento():
    cursor.execute('SELECT * FROM REQUIREMENT')
    tasks = cursor.fetchall()
    return render_template('requiremento.html',tasks=tasks)
@app.route('/addrequiremento',methods=['POST','GET'])
def addrequiremento():
    if request.method == 'POST':
        did = request.form['did']
        dname=request.form['dname']
        quantity=request.form['quantity']
        cost=request.form['cost']
        sup=request.form['sup']
        manu = request.form['manu']
        if(check_did(did)):
            sentence = 'Drug detail not available. Please add drug details and complete this process'
            return render_template('addrequiremento.html',sentence = sentence)
        if(check_did_available(did)):
            sentence = 'Still stock is available. Raise new request for after completion exciting stock'
            return render_template('addrequiremento.html',sentence = sentence)
        cursor.execute(r"INSERT INTO REQUIREMENT (D_ID,D_NAME,QUANTITY,COST,SUPPLIER,MANUFACTURER) VALUES('%s','%s','%s','%s','%s','%s')"%(did,dname,quantity,cost,sup,manu))
        db.commit()
        return redirect('/requiremento')
    else:
        return render_template('addrequiremento.html')
@app.route('/uprequiremento/<int:did>',methods=['POST','GET'])
```

```
def uprequiremento(did):
    cursor.execute('SELECT * FROM REQUIREMENT WHERE D_ID=%d' % did)
    tasks = cursor.fetchone()
    if request.method == 'POST':
        quantity=request.form['quantity']
        sup=request.form['sup']
        manu = request.form['manu']
        cursor.execute(r"UPDATE REQUIREMENT SET QUANTITY='%s',SUPPLIER='%s',MANUFACTURER='%s' WHERE D_ID = '%s'"%(quantity,sup,manu,did))
        db.commit()
        return redirect('/requiremento')
    else:
        return render_template('uprequiremento.html',tasks = tasks)
@app.route('/upstocka/<float:cost>',methods=['POST','GET'])
def upstocka(cost):
    cursor.execute('SELECT * FROM STOCK WHERE COST=%d' % cost)
    tasks = cursor.fetchone()
    if request.method == 'POST':
        num=request.form['quantity']
        cursor.execute(r"UPDATE STOCK SET QUANTITY = %s WHERE COST = %f"%(num,cost))
        db.commit()

        return redirect('/stocka')

    else:

        return render_template('upstocka.html', tasks=tasks)

@app.route('/delrequiremento/<int:did>',methods=['POST','GET'])

def delrequiremento(did):

    cursor.execute(r"DELETE FROM REQUIREMENT WHERE D_ID=%d"%(did))

    db.commit()

    return redirect('/requiremento')

@app.route('/costupdate/<int:did>',methods=['POST','GET'])

def costupdate(did):



    cursor.execute('SELECT * FROM REQUIREMENT WHERE D_ID=%d' % did)

    tasks = cursor.fetchone()

    if request.method == 'POST':

        num=request.form['cost']

        #print(num)

        cursor.execute(r"UPDATE REQUIREMENT SET COST = %s WHERE D_ID = 
%d"%(num,did))

        db.commit()

        return redirect('/requiremento')
    else:

        return render_template('costupdate.html',tasks = tasks)



@app.route('/stockreceived/<int:did>',methods=['POST','GET'])

def stockreceived(did):



    cursor.execute(r"SELECT * FROM REQUIREMENT WHERE D_ID=%d"%(did))

    tasks = cursor.fetchone()

    did = tasks[0]

    dname= tasks[1]

    quantity= tasks[2]

    cost=tasks[3]

    sup=tasks[4]

    manu = tasks[5]

    cursor.execute(r"INSERT INTO STOCK
(D_ID,D_NAME,QUANTITY,COST,SUPPLIER,MANUFACTURER)
VALUES('%s','%s','%s','%s','%s','%s')"%(did,dname,quantity,cost,sup,manu))

    cursor.execute(r"DELETE FROM REQUIREMENT WHERE D_ID=%d"%(did))

    db.commit()

    return redirect('/requiremento')
```

```
@app.route('/requiremente',methods=['POST','GET'])

def requiremente():

    cursor.execute('SELECT * FROM REQUIREMENT')

    tasks = cursor.fetchall()

    return render_template('requiremente.html',tasks=tasks)


@app.route('/addrequiremente',methods=['POST','GET'])

def addrequiremente():

    if request.method == 'POST':

        did = request.form['did']

        dname=request.form['dname']

        quantity=request.form['quantity']

        cost=request.form['cost']

        sup=request.form['sup']

        manu = request.form['manu']

        if(check_did(did)):

            sentence = 'Drug detail not available. Please add drug details and complete this process'

            return render_template('addrequiremente.html',sentence = sentence)
        if(check_did_available(did)):
            sentence = 'Still stock is available. Raise new request for after completion exciting stock'
            return render_template('addrequiremente.html',sentence = sentence)
        cursor.execute(r"INSERT INTO REQUIREMENT (D_ID,D_NAME,QUANTITY,COST,SUPPLIER,MANUFACTURER) VALUES('%s','%s','%s','%s','%s','%s')"%(did,dname,quantity,cost,sup,manu))
        db.commit()
        return redirect('/requiremente')
    else:
        return render_template('addrequiremente.html')



@app.route('/uprequiremente/<int:did>',methods=['POST','GET'])
def uprequiremente(did):
    cursor.execute('SELECT * FROM REQUIREMENT WHERE D_ID=%d' % did)
    tasks = cursor.fetchone()
    if request.method == 'POST':
        quantity=request.form['quantity']
        sup=request.form['sup']
        manu = request.form['manu']
        cursor.execute(r"UPDATE REQUIREMENT SET QUANTITY='%s',SUPPLIER='%s',MANUFACTURER='%s' WHERE D_ID = '%s'"%(quantity,sup,manu,did))

        db.commit()

        return redirect('/requiremente')

    else:

        return render_template('uprequiremente.html',tasks = tasks)




@app.route('/delrequiremente/<int:did>',methods=['POST','GET'])

def delrequiremente(did):

    cursor.execute(r"DELETE FROM REQUIREMENT WHERE D_ID=%d"%(did))

    db.commit()

    return redirect('/requiremente')

@app.route('/stocke',methods=['POST','GET'])

def stocke():

    cursor.execute('SELECT * FROM STOCK')

    tasks = cursor.fetchall()

    return render_template('stocke.html', tasks=tasks)
```

```
@app.route('/stocko',methods=['POST','GET'])

def stocko():

    cursor.execute('SELECT * FROM STOCK')

    tasks = cursor.fetchall()

    return render_template('stocko.html',tasks=tasks)



@app.route('/updatestock/<int:did>',methods=['POST','GET'])

def updatestock(did):

    cursor.execute('SELECT * FROM STOCK WHERE D_ID=%d' % did)

    tasks = cursor.fetchone()

    if request.method == 'POST':

        quantity=request.form['quantity']

        sup=request.form['sup']

        manu = request.form['manu']

        cursor.execute(r"UPDATE STOCK SET QUANTITY='%s',SUPPLIER='%s',MANUFACTURER='%s' WHERE D_ID = '%s'"%(quantity,sup,manu,did))

        db.commit()

        return redirect('/stocko')

    else:

        return render_template('updatestock.html',tasks = tasks)
```

```
@app.route('/delstock/<int:did>',methods=['POST','GET'])
def delstock(did):
    cursor.execute(r"DELETE FROM STOCK WHERE D_ID=%d"%(did))
    db.commit()
    return redirect('/stocko')



@app.route('/costupdatestock/<int:did>',methods=['POST','GET'])
def costupdatestock(did):


    cursor.execute('SELECT * FROM STOCK WHERE D_ID=%d' % did)
    tasks = cursor.fetchone()
    if request.method == 'POST':
        num=request.form['cost']
        #print(num)
        cursor.execute(r"UPDATE STOCK SET COST = %s WHERE D_ID = %d"%(num,did))
        db.commit()
        return redirect('/stocko')
    else:
        return render_template('costupdatestock.html',tasks = tasks)
```

```
@app.route('/addstock',methods=['POST','GET'])

def addstock():

    if request.method == 'POST':

        dname=request.form['dname']

        quantity=request.form['quantity']

        cost=request.form['cost']

        sup=request.form['sup']

        cursor.execute(r"INSERT INTO STOCK
VALUES('%s',%s,%s,'%s')"%(dname,quantity,cost,sup))

        db.commit()

        return redirect('/stocko')

    else:

        return render_template('addstock.html')

@app.route('/prescripo',methods=['POST','GET'])

def prescripo():

    if request.method == 'POST':

        pname = request.form['Patient Name']

        page = request.form['Patient Age']

        pphone = request.form['Patient Phone']

        eid = request.form['Employee id']

        pid = request.form['Prescription id']
        dna = request.form['Drug Name']

        quantity = request.form['Quantity']

        dose = request.form['Dose']

        did = request.form['Drug ID']

        date = request.form['Date']

        print(eid,did)

        if(check_did(did)):

            sentence = 'Drug detail not available. Please add drug details and complete this process'

            return render_template('prescripo.html',sentence = sentence)

        if(check_eid(eid)):

            sentence = 'Employee detail not available. Please add employee details and complete
this process'

            return render_template('prescripo.html',sentence = sentence)

        if(check_quantity(quantity,did)):

            sentence = 'Required quantity not available. Please order for requirement details and
complete this process'

            return render_template('prescripo.html',sentence = sentence)



        cursor.execute("SELECT * FROM STOCK where D_ID = '%s'" % (did))

        tasks=cursor.fetchone()

        price = tasks[3]*int(quantity)
```

```
        cursor.execute("INSERT INTO PRESCRIPTION VALUES('%s','%s','%s','%s','%s','%s','%s','%s','%s','%s')" % (eid,pid, quantity, dose, did,dna,pname,date,page,pphone))

        cursor.execute("INSERT INTO INVOICE VALUES('%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s')" % (eid,pid, quantity, dose, did,dna,pname,date,page,pphone,price))

        cursor.execute(r"UPDATE STOCK SET QUANTITY=QUANTITY-'%s' WHERE D_ID='%s'"%(quantity,did))

        db.commit()

        return redirect("/invo")

    else:

        return render_template('prescripo.html')



@app.route('/prescripe',methods=['POST','GET'])

def prescripe():

    if request.method == 'POST':

        pname = request.form['Patient Name']

        page = request.form['Patient Age']

        pphone = request.form['Patient Phone']

        eid = request.form['Employee id']

        pid = request.form['Prescription id']

        dna = request.form['Drug Name']

        quantity = request.form['Quantity']
        dose = request.form['Dose']

        did = request.form['Drug ID']

        date = request.form['Date']



        if(check_did(did)):

            sentence = 'Drug detail not available. Please add drug details and complete this process'

            return render_template('prescripe.html',sentence = sentence)

        if(check_eid(eid)):

            sentence = 'Employee detail not available. Please add employee details and complete this process'

            return render_template('prescripe.html',sentence = sentence)

        if(check_quantity(quantity,did)):

            sentence = 'Required quantity not available. Please order for requirment details and complete this process'

            return render_template('prescripe.html',sentence = sentence)




        cursor.execute("SELECT * FROM STOCK where D_ID = '%s'" % (did))

        tasks=cursor.fetchone()

        price = tasks[3]*int(quantity)
```

```
        cursor.execute("INSERT INTO PRESCRIPTION VALUES('%s','%s','%s','%s','%s','%s','%s','%s','%s','%s')" % (eid,pid, quantity, dose, did,dna,pname,date,page,pphone))

        cursor.execute("INSERT INTO INVOICE VALUES('%s','%s','%s','%s','%s','%s','%s','%s','%s','%s','%s')" % (eid,pid, quantity, dose, did,dna,pname,date,page,pphone,price))

        cursor.execute(r"UPDATE STOCK SET QUANTITY=QUANTITY-'%s' WHERE D_ID='%s'"%(quantity,did))

        db.commit()

        return redirect("/inve")

    else:

        return render_template('prescripe.html',sentence = '')





def check_did(did):

    test = True

    cursor.execute('SELECT * FROM DRUG WHERE D_ID=%d' % int(did))

    c=0

    for t in cursor:

        c+=1;

        if c!=0:

            test = False
    return test


def check_eid(eid):
    test = True
    cursor.execute('SELECT * FROM EMPLOYEE WHERE EMP_ID=%d' % int(eid))
    c=0
    for t in cursor:
        c+=1;
        if c!=0:
            test = False
    return test


def check_quantity(quantity,did):
    test = False
    cursor.execute('SELECT * FROM STOCK WHERE D_ID=%d' % int(did))
    tasks = cursor.fetchone()
    if((int(tasks[2])-int(quantity)) < 0):
        test = True
    return test


def check_did_available(did):
    test = False

    cursor.execute('SELECT * FROM STOCK WHERE D_ID=%d' % int(did))

    c=0

    for t in cursor:

        c+=1;

        if c!=0:

            test = True

    return test




@app.route('/invo',methods=['POST','GET'])

def invo():

    cursor.execute('SELECT * FROM INVOICE')

    tasks=cursor.fetchall()

    return render_template('invo.html',tasks=tasks)


@app.route('/inve',methods=['POST','GET'])

def inve():

    cursor.execute('SELECT * FROM INVOICE')

    tasks = cursor.fetchall()
    return render_template('inve.html', tasks=tasks)




@app.route('/viewdrugo',methods=['POST','GET'])

def viewdrugo():

    cursor.execute('SELECT * FROM DRUG')

    tasks=cursor.fetchall()

    return render_template('viewdrugo.html',tasks=tasks)


@app.route('/viewdruge',methods=['POST','GET'])

def viewdruge():

    cursor.execute('SELECT * FROM DRUG')

    tasks = cursor.fetchall()

    return render_template('viewdruge.html', tasks=tasks)




@app.route('/adddrugo',methods=['POST','GET'])

def adddrugo():

    if request.method == 'POST':

        did = request.form['did']
        dname=request.form['dname']

        sup=request.form['sup']

        manu = request.form['manu']

        uses = request.form['uses']

        side = request.form['side']

        cursor.execute(r"INSERT                    INTO                    DRUG
(D_ID,D_NAME,SUPPLIER,MANUFACTURER,USES,SIDE_EFFECT)
VALUES('%s','%s','%s','%s','%s','%s')"%(did,dname,sup,manu,uses,side))

        db.commit()

        return redirect('/viewdrugo')

    else:

        return render_template('adddrugo.html')



@app.route('/updrugo/<int:did>',methods=['POST','GET'])

def updrugo(did):

    cursor.execute('SELECT * FROM DRUG WHERE D_ID=%d' % did)

    tasks = cursor.fetchone()

    if request.method == 'POST':

        sup=request.form['sup']

        manu = request.form['manu']
        uses = request.form['uses']

        side = request.form['side']

        cursor.execute(r"UPDATE DRUG SET USES='%s',SUPPLIER='%s',MANUFACTURER='%s',SIDE_EFFECT='%s' WHERE D_ID = '%s'"%(uses,sup,manu,side,did))

        db.commit()

        return redirect('/viewdrugo')

    else:

        return render_template('updrugo.html',tasks = tasks)




@app.route('/deldrugo/<int:did>',methods=['POST','GET'])

def deldrugo(did):

    cursor.execute(r"DELETE FROM DRUG WHERE D_ID=%d"%(did))

    db.commit()

    return redirect('/viewdrugo')

if __name__== '__main__':

    app.run(debug=True)
```

CHAPTER 7

CONCLUSION

- Detailed information gathering has to be done. Without that the purpose for using the software won't be satisfied properly.
- However, it can give good profits in the long run.
- Implementing the software requires change in the business practices.
- Efficient organization of all knowledge is the analysis company and easy analysis access and retrieval of information is possible.
- Company using this software will always be able to plan in future and always be aware of their financial position in the market.
- It leads to ease in functioning of business processes.
- The project can be made more robust by including biometric verification.
- There is also a scope to expand by implementing newer technologies like cloud etcetera.

CHAPTER 8

BIBLIOGRAPY

- R. R. Berardi, L. V. Allen, E. M. DeSimone (eds.), *Handbook of Nonprescription Drugs*, 14th ed., American Pharmaceutical Association, Washington, DC, 2004.
- G. Briggs, R. K. Freeman, S. J. Yaffe (eds.), *Drugs in Pregnancy and Lactation*, 7th ed., Williams & Wilkins, Baltimore, MD, 2005.
- J. T. DiPiro, T. L. Schwinghammer, B. Wells (eds.), *Pharmacotherapy: A Pathophysiologic Approach*, 5th ed., Appleton & Lange, Stamford, CT, 2002.
- R. Gennaro, et al. (eds.), *Remingtonâs The Science and Practice of Pharmacy*, 20th ed., Mack Publishers, Easton, PA, 2000.
- J. D. Grabenstein, *Immunofacts: Vaccines and Immunologic Drugs*, Drug Facts and Comparisons, St. Louis, MO, 1995.