MEDEX

A EXPERIENTIAL LEARNING PROJECT REPORT
Course DA1001

Submitted in partial fulfilment of the requirement for the award of the degree

of

BACHELOR OF TECHNOLOGY (B.Tech)
in
Computer Science and Engineering

by

Ameya Rao
209301623

Under the guidance of

Mrs. Shikha Mundra
Assistant Professor

Department of Computer Science and Engineering
School of Computing and Information Technology
Manipal University Jaipur
Jaipur, Rajasthan
2021

INDEX

Acknowledgments

I would like to express my deep and sincere gratitude to my mentor of this project Mrs. Shikha Mundra, who has put in utmost dedication from the very beginning to help make this project a success. Her perseverance and patience have helped put me down the right path to completing this project.

I shall forever be indebted to my PSUC (Problem Solving Using Computers) teacher, Mr. Prashanth Hemrajani and am extremely thankful and pay my gratitude towards his invaluable teachings and suggestions.

I would like to wholeheartedly thank my friends and family whose advice have helped me gain perspective while building this project. I thank them for their relentless support and always helping in boosting my morale.

I extend my gratitude to the Department of Computer Science and Engineering, Manipal University Jaipur for giving me this opportunity.

Introduction:

This project is made to try and replicate the working of a medical store database along with a management prospect and point of sales (POS) view. This project is made to show the inner workings that most stores use to manage the vast amounts of data they receive.

Libraries used:

- Tkinter
- tkinter.ttk
- mysql.connector
- datetime

Motivation:

Ever since I was a child the storing and keeping of this vast amount of data in this world has always piqued my curiosity.

The way major Multi National Companies manage to store all their data in a structured matter that can easily by analysed by a group of people always intrigued me and this is what led me to dive into the world of data science and analytics.

Since there is a covid crisis going on and there is an ever-increasing need for medicine from pharmaceuticals. The medical stores would have more data to deal with and hence I thought of trying to make a simple and efficient way of storing and managing this data.

Statement Of Problem:

The aim of doing this project is to show the inner workings of a medical store. For example, managing medicine stocks, billings and billing history, adding new employees and deleting previous employee records in the company, Viewing customer and employee details and more.

Project Outline:

Everyone will have their own account. Only managers or higher ups in the company are able to add new employees or delete old employee record.

1. We can view the stocks present and order new stocks as well.
2. We can create bills and view the billing history of customers.
3. We can manage the salary, post and more of the employees in the company (if you have your post as manager only).

This allows us to have an overview of all the data flowing into the database with ease of access and manage the employees, items, stocks and customer data all at the same time.

How to use the software:

The first signup into the software must be done with an employee id ‘E001’ and

initial password ‘#00000001#’. This shall be the manager’s signup. After this,

the user shall login with the username and password they set and they will be

directed to the software's main menu. They will then have the following options:

1. Billing:

Make new billing entries along with customer details.

2. Stock:

Add new items to the shop's stock using at least its serial number,

batch code, and the product's name.

3. Customers:

Details of customers can be seen and searched here on the basis of

their phone number.

4. History:

Bill details can be seen here with the help of its bill number. The

customer's details will be shown here too.

5. Employees:

This option is available only for the Manager. Here, the manager can

check an employee’s data, add new employees, update a current

employee’s post or salary or both, and even remove an employee, all

on the basis of the employee id.

Features:

Employees get to add their details and set up their own accounts with the

employee id and initial password given to them by the employer.

- Only managers get to use the employee management option. This option

won’t be visible to other employees.

- Data from the database, and bills are properly shown in the software in

tabular format.

- Changes in the database can be observed in real-time, including reduce in

quantity of an item in the ‘stocks’ table after it is purchased, and changes

in the ‘staff’ table as their salary or post is updated.

- Appropriate errors are shown for wrong inputs.

- Only the phone number needs to be added while making a bill if the

customer had previously made a purchase from the store.

- If the quantity of a product is 0 after a bill is made, it is automatically

removed from the table.

- The customer’s details are shown in the billing history using joins where a

customer’s mobile number is used as the foreign key and primary key in

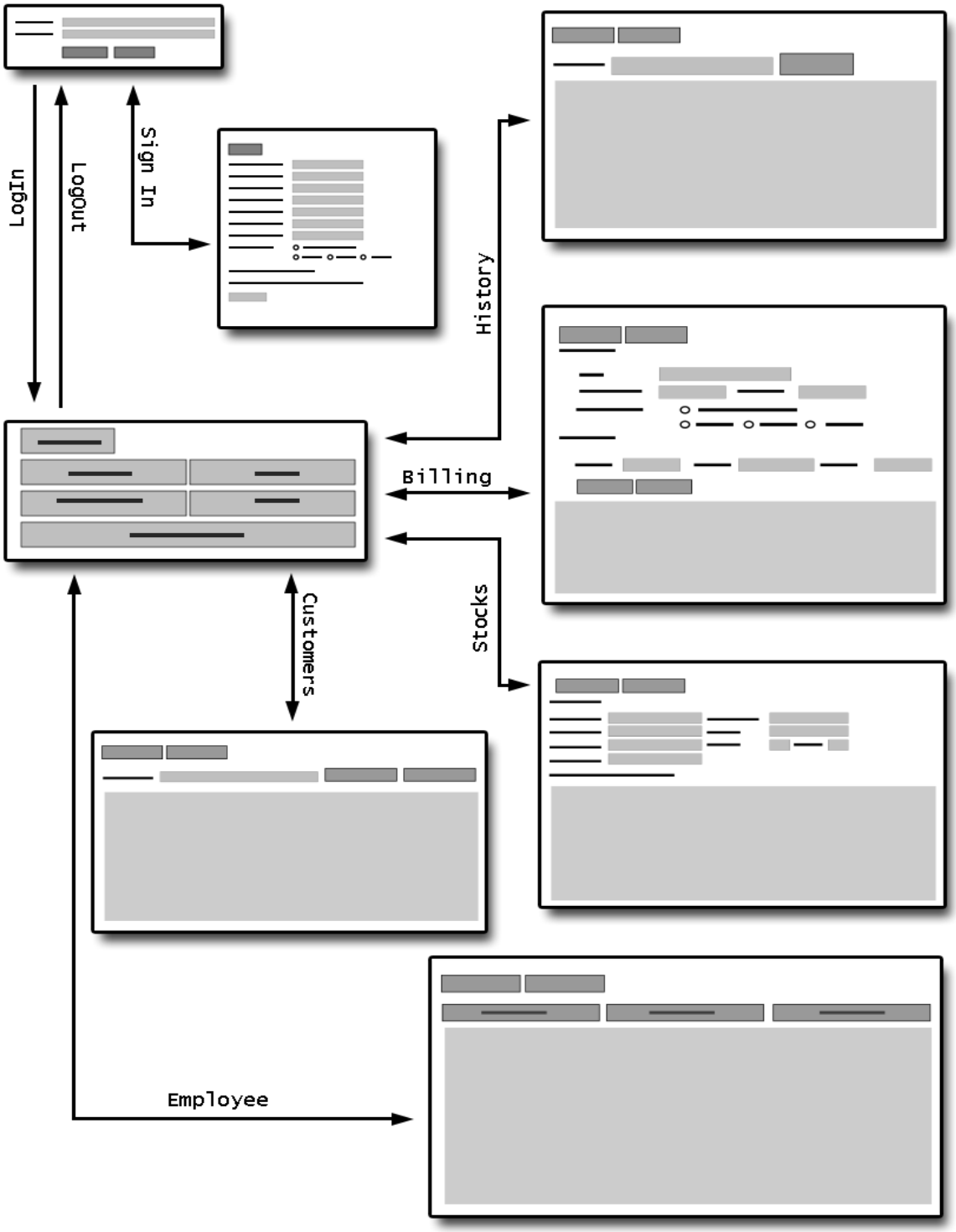
the tables ‘history’ and ‘customers’ respectively.

- Bill number is generated and displayed automatically even before the bill

is made.

- Log out button available on all pages.

Methodology:



Source Code:

MySQL Queries:

```
create database medstore;

use medstore;

create table customer(mob varchar(11) primary key,
name varchar(25), gender char(1), DOB date);

create table staff(eid varchar(5), name varchar(20),
username varchar(20), passwd varchar(15), post
varchar(15), mobile char(11), sal int(6), gender char(1),
dob date, ipass char(10), date_employed date);

create table stock(serialno varchar(15), supplier
varchar(25), manufacturer varchar(20), Itemname
varchar(15), qty int(3), rate float, expiry date, batchcode
varchar(18));

create table history(Billno varchar(5)not null, Serialno
varchar(15), manufacturer varchar(20), itemname
```

varchar(15), batchcode varchar(18), qty int(3), rate float, expiry date, custmob varchar(11) not null, date date not null);

insert into staff values(‘E001’, null,null,null,’Manager’,null,null,null,null,’#00000001#’,curdate());

Python Code:

```
from tkinter import *
from tkinter.ttk import *
import mysql.connector as sql
import datetime

mydb=sql.connect(host="localhost",user="root",passwd="1609",db="Medstore")
cursor=mydb.cursor()
mydb.close()

def click_login(uid,passwd,caller,wrong_c):
    mydb.connect()
    cursor.execute(f"select passwd from staff where username='{uid}'")
    output=cursor.fetchall()
    print(output)
    if cursor.rowcount==1:
        epassw=output[0][0]
        if epassw==passwd:
            cursor.execute(f"select post from staff where username='{uid}'")
            post=(cursor.fetchall())[0][0]
            MainMenu(caller,post)
```

```
        else:
            wrong_c.grid(row=0,column=2)
    else:
        wrong_c.grid(row=0,column=2)


def LoginPage(caller=None):
    if caller!=None:
        caller.destroy()
    login=Tk()
    login.title("Login")
    fm1=Frame(login)
    fm1.grid(row=2,column=1,sticky=W)
    Label(login,text="Username: ").grid(row=0,column=0)
    Label(login,text="Password: ").grid(row=1,column=0)
    wrong_c=Label(fm1, text="Wrong credentials. Try again.")
    username,passwd=StringVar(),StringVar()
    Entry(login, textvariable=username,width=50).grid(row=0,column=1)
    Entry(login, textvariable=passwd,width=50,show='*').grid(row=1,column=1)
   Button(fm1, text="LOG IN", command=lambda:
   click_login(username.get(),passwd.get(),login,wrong_c)).grid(row=0,column=0,sticky=W)
    Button(fm1, text="SIGN UP", command=lambda: SignUp(login)).grid(row=0,column=1)
    login.mainloop()


def MainMenu(caller,post):
    caller.destroy()
    mnmnu=Tk()
    mnmnu.title("Main Menu")
    Button(mnmnu,text="LOGOUT",command=lambda: LoginPage(mnmnu)).grid(row=0,column=0,sticky=W)
    fr1=Frame(mnmnu)
    fr1.grid(row=1,column=0)
    Button(fr1,text="Billing",width=25,command=lambda: billing(mnmnu,post)).grid(row=0,column=0)
    Button(fr1,text="Stock",command=lambda: stock(mnmnu,post),width=25).grid(row=0,column=1)
    Button(fr1,text="Customers",command=lambda: customers(mnmnu,post),width=25).grid(row=1,column=0)
    Button(fr1,text="History",width=25,command=lambda: history(mnmnu,post)).grid(row=1,column=1)
    if post=="Manager":
        Button(fr1,text="Employees",width=50,command=lambda:
employee(mnmnu,post)).grid(row=2,column=0,columnspan=2)
    mnmnu.mainloop()


def SignUp(caller):
```

```
    caller.destroy()

    newacc=Tk()

    newacc.title("Sign up")

    newacc.geometry("500x370")

    Button(newacc,text="BACK",command=lambda: LoginPage(newacc)).grid(row=0,column=0,sticky=W)

    Label(newacc,text="Enter employee id*: ").grid(row=1,column=0,sticky=W)

    Label(newacc,text="Enter initial passcode*: ").grid(row=2,column=0,sticky=W)

    Label(newacc,text="Enter your name*: ").grid(row=3,column=0,sticky=W)

    Label(newacc,text="Enter username*: ").grid(row=4,column=0,sticky=W)

    Label(newacc,text="Enter password*: ").grid(row=5,column=0,sticky=W)

    Label(newacc,text="Re-enter password*: ").grid(row=6,column=0,sticky=W)

    Label(newacc,text="Enter mobile number: ").grid(row=7,column=0,sticky=W)

    Label(newacc,text="Enter date of birth: ").grid(row=8,column=0,sticky=W)

    Label(newacc,text="Select gender: ").grid(row=9,column=0,sticky=NW)

    Label(newacc,text="(Fields marked '*' are important)").grid(row=11,column=0,columnspan=2,sticky=W)

    Label(newacc,text="(Contact employer for the first 2
fields)").grid(row=10,column=0,columnspan=2,sticky=W)


   eid,ipass,name,uname,passwd,repass,mobno,dob,gender=StringVar(),StringVar(),StringVar(),StringVar()
   ,StringVar(),StringVar(),StringVar(),StringVar(),StringVar()

    gender.set(None)

    Entry(newacc, textvariable=eid, width=20).grid(row=1,column=1,sticky=W)

    Entry(newacc, textvariable=ipass, width=20,show='*').grid(row=2,column=1,sticky=W)

    Entry(newacc, textvariable=name, width=30).grid(row=3,column=1,sticky=W)

    Entry(newacc, textvariable=uname, width=30).grid(row=4,column=1,sticky=W)

    Entry(newacc, textvariable=passwd, width=30,show='*').grid(row=5,column=1,sticky=W)

    Entry(newacc, textvariable=repass, width=30,show='*').grid(row=6,column=1,sticky=W)

    Entry(newacc, textvariable=mobno, width=30).grid(row=7,column=1,sticky=W)

    Entry(newacc, textvariable=dob, width=25).grid(row=8,column=1,sticky=W)

    gen=Frame(newacc)

    gen.grid(row=9,column=1,sticky=W)

    Radiobutton(gen,text="Prefer not to
specify",variable=gender,value="null").grid(row=0,column=0,columnspan=2,sticky=W)

    Radiobutton(gen,text="Male",variable=gender,value="'M'").grid(row=1,column=0,sticky=W)

    Radiobutton(gen,text="Female",variable=gender,value="'F'").grid(row=1,column=1,sticky=W)

    Radiobutton(gen,text="Others",variable=gender,value="'O'").grid(row=1,column=2,sticky=W)

    Button(newacc, text="ADD ACCOUNT",command=lambda: click_createacc(eid.get(),
                                                                      name.get(),
                                                                      ipass.get(),
                                                                      uname.get(),
                                                                      passwd.get(),
```

```
                                                                    repass.get(),
                                                                    mobno.get(),
                                                                    dob.get(),
                                                                    gender.get(),

newacc)).grid(row=12,column=0,sticky=W)
    newacc.mainloop()


def click_createacc(eid,name,ipass,uname,passwd,repass,mobno,dob,gender,caller):
    mydb.connect()
    cursor.execute(f"select ipass,name from staff where eid='{eid}'")
    check = cursor.fetchall()
    error=Frame(caller)
    error.grid(row=12,column=1,ipadx=83,sticky=W)
    if cursor.rowcount == 0:
        Label(error, text="Please contact your employer for a valid ID.").pack(fill='x',side=LEFT)
    elif not check[0][1]==None:
        Label(error, text="You have already made an account.").pack(fill='x',side=LEFT)
    elif not ipass==check[0][0]:
        Label(error, text="Please contact your employer for a valid initial
passcode.").pack(fill='x',side=LEFT)
    elif name==='':
        Label(error, text="Please enter your name.").pack(fill='x',side=LEFT)
    elif len(uname)<8:
        Label(error, text="Username must be at least 8 chars long.").pack(fill='x',side=LEFT)
    elif len(passwd)<8:
        Label(error, text="Password must be at least 8 chars long.").pack(fill='x',side=LEFT)
    elif not repass==passwd:
        Label(error, text="Passwords do not match.").pack(fill='x',side=LEFT)
    else:
        dob="'"+dob+"'" if not dob==='' else 'null'
        mobno='null' if mobno==='' else "'"+mobno+"'"
        cursor.execute(f"update staff set\
                        name='{name}',\
                        username='{uname}',\
                        passwd='{passwd}',\
                        mobile={mobno},\
                        gender={gender},\
                        dob={dob} \
                        where eid='{eid}'")
```

```
        mydb.commit()
        mydb.close()
        caller.destroy()
        LoginPage()

def stock(caller,post):
    caller.destroy()
    inventory=Tk()
    inventory.title("Stock")
    topbuttons=Frame(inventory)
    topbuttons.grid(row=0,sticky=W)
    Button(topbuttons,text="LOGOUT",command=lambda: LoginPage(inventory)).grid(row=0,column=1)
    Button(topbuttons,text="BACK",command=lambda: MainMenu(inventory,post)).grid(row=0,column=0)
    additems=Frame(inventory)
    additems.grid(row=2,column=0,columnspan=3,sticky=W)
    Label(inventory,text="Add to stock:-\n").grid(row=1,column=0,sticky=W)
    Label(additems,text="seriel no.*:").grid(row=0,column=0,sticky=W)
    Label(additems,text="Batch code*:").grid(row=0,column=2,sticky=W)
    Label(additems,text="Manufacturer:").grid(row=1,column=0,sticky=W)
    Label(additems,text=" Supplier:").grid(row=1,column=2,sticky=W)
    Label(additems,text="Product name*:").grid(row=2,column=0,sticky=W)
    Label(additems,text=" Quantity:").grid(row=2,column=2,sticky=W)
    Label(additems,text=" Rate*:").grid(row=2,column=4,sticky=W)
    Label(additems,text="Expiry:").grid(row=3,column=0,sticky=W)
    Label(inventory,text="Fields marked '*' are important.").grid(row=3,column=0,sticky=W)

   seriel,batchcode,manu,supp,pname,qty,rate,exp=StringVar(),StringVar(),StringVar(),StringVar(),Strin
   gVar(),StringVar(),StringVar(),StringVar()
    e1=Entry(additems,textvariable=seriel,width=20)
    e1.grid(row=0,column=1,sticky=W)
    e2=Entry(additems,textvariable=batchcode,width=20)
    e2.grid(row=0,column=3,columnspan=3,sticky=W)
    e3=Entry(additems,textvariable=manu,width=30)
    e3.grid(row=1,column=1,sticky=W)
    e4=Entry(additems,textvariable=supp,width=30)
    e4.grid(row=1,column=3,columnspan=3,sticky=W)
    e5=Entry(additems,textvariable=pname,width=30)
    e5.grid(row=2,column=1,sticky=W)
    e6=Entry(additems,textvariable=qty,width=10)
    e6.grid(row=2,column=3,sticky=W)
```

```
    e7=Entry(additems,textvariable=rate,width=12)
    e7.grid(row=2,column=5,sticky=W)
    e8=Entry(additems,textvariable=exp,width=20)
    e8.grid(row=3,column=1,sticky=W)
    entries=[e1,e2,e3,e4,e5,e6,e7,e8]
    Button(inventory,text="Add entry",command=lambda: click_entry("new",inventory,entries,tree,
                                                                  seriel.get(),
                                                                  manu.get(),
                                                                  supp.get(),
                                                                  pname.get(),
                                                                  qty.get(),
                                                                  rate.get(),
                                                                  exp.get(),
batchcode.get())).grid(row=4,column=0,sticky=W)

    tree=Treeview(inventory)
    tree["columns"]=(0,1,2,3,4,5,6,7)
    tree.column("#0",width=0)
    tree.column(0,width=100)
    tree.column(1,width=100)
    tree.column(2,width=100)
    tree.column(3,width=100)
    tree.column(4,width=50)
    tree.column(5,width=50)
    tree.column(6,width=100)
    tree.column(7,width=130)
    tree.heading(0,text="seriel number")
    tree.heading(1,text="Supplier")
    tree.heading(2,text="Manufacturer")
    tree.heading(3,text="Product")
    tree.heading(4,text="Qty.")
    tree.heading(5,text="Rate")
    tree.heading(6,text="Expiry")
    tree.heading(7,text="batchcode")
    tree.grid(row=5,column=0)
    vsb=Scrollbar(inventory,orient="vertical",command=tree.yview)
    tree.configure(yscrollcommand=vsb.set)
    vsb.grid(row=5,column=1,sticky=NS)
```

```
    mydb.connect()
    cursor.execute("select * from stock order by Itemname asc, expiry desc")
    for i in cursor.fetchall():
        tree.insert('',"end",values=(i))
    mydb.close()
    inventory.mainloop()


def
click_entry(mode,caller,entries,tree,serielno,manufacturer,supplier,itemname,qty,rate,expiry,batchcode
):
    error=Frame(caller)
    error.grid(row=6,column=0,sticky=W,ipadx=60)
    if serielno==='':
        Label(error,text="seriel number is necessary.").grid(row=0,column=0,sticky=W)
    elif itemname==='':
        Label(error,text="Product name is necessary.").grid(row=0,column=0,sticky=W)
    elif rate==='':
        Label(error,text="Rate is necessary").grid(row=0,column=0,sticky=W)
    elif batchcode==='':
        Label(error,text="Batch code is necessary").grid(row=0,column=0,sticky=W)
    else:
        manufacturer='null' if manufacturer==='' else "'"+manufacturer+"'"
        supplier='null' if supplier==='' else "'"+supplier+"'"
        qty=0 if qty==='' else qty
        expiry='null' if expiry==='' else "'"+expiry+"'"
        mydb.connect()
        cursor.execute(f"insert into stock values('{serielno}',{supplier},{manufacturer},\
                                                '{itemname}',{qty},{rate},{expiry},'{batchcode}')")
        mydb.commit()
        for i in entries:
            i.delete(0,100)
        cursor.execute("select * from stock order by Itemname asc, expiry desc")
        for i in tree.get_children():
            tree.delete(i)
        for i in cursor.fetchall():
            tree.insert('',"end",values=(i))
        mydb.close()


def employee(caller,post):
    caller.destroy()
```

```
    emp=Tk()

    emp.title("Employee Management")

    topbuttons=Frame(emp)

    topbuttons.grid(row=0,sticky=W)

    Button(topbuttons,text="LOGOUT",command=lambda: LoginPage(emp)).grid(row=0,column=1)

    Button(topbuttons,text="BACK",command=lambda: MainMenu(emp,post)).grid(row=0,column=0)

    fr1=Frame(emp)

    fr1.grid(row=1,column=0,columnspan=2)

    options=Frame(emp)

    options.grid(row=3,column=0,sticky=W)

    tree=Treeview(emp)

    Button(fr1,text="Add new employee",width=48,command=lambda:
add_emp(tree,options)).grid(row=0,column=0)

    Button(fr1,text="Update an employee",width=48,command=lambda:
update_employee(tree,options)).grid(row=0,column=1)

    Button(fr1,text="Remove an employee",width=48, command=lambda:
remove(tree,options)).grid(row=0,column=2)

    tree["columns"]=(0,1,2,3,4,5,6,7,8,9)

    tree.column("#0",width=0)

    tree.column(0,width=50)

    tree.column(1,width=140)

    tree.column(2,width=100)

    tree.column(3,width=80)

    tree.column(4,width=80)

    tree.column(5,width=70)

    tree.column(6,width=50)

    tree.column(7,width=100)

    tree.column(8,width=100)

    tree.column(9,width=110)

    tree.heading(0,text="E.id")

    tree.heading(1,text="Name")

    tree.heading(2,text="Username")

    tree.heading(3,text="Post")

    tree.heading(4,text="Mobile")

    tree.heading(5,text="Salary")

    tree.heading(6,text="Gender")

    tree.heading(7,text="Date of Birth")

    tree.heading(8,text="Initial password")

    tree.heading(9,text="Date employeed")

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

```
    vsb=Scrollbar(emp,orient="vertical",command=tree.yview)

    tree.configure(yscrollcommand=vsb.set)

    vsb.grid(row=2,column=1,sticky=NS)

    mydb.connect()

    cursor.execute("select eid,name,username,post,mobile,sal,gender,dob,ipass,date_employed from
staff")

    for i in cursor.fetchall():

            tree.insert('',"end",values=(i))

    mydb.close()

    emp.mainloop()


def add_emp(tree,fr1):

    for widget in fr1.winfo_children():

        widget.destroy()

    post,sal=StringVar(),StringVar()

    Label(fr1,text="Add post: ").grid(row=0,column=0)

    Entry(fr1,textvariable=post).grid(row=0,column=1)

    Label(fr1,text="Add salary: ").grid(row=0,column=2)

    Entry(fr1,textvariable=sal).grid(row=0,column=4)

    Button(fr1,text="ADD",command=lambda:
click_add_emp(tree,post.get(),sal.get())).grid(row=0,column=5)


def click_add_emp(tree,post,sal):

    mydb.connect()

    cursor.execute("select max(eid),max(date_employed),max(ipass) from staff")

    op=cursor.fetchall()

    today=str(datetime.date.today())

    date_employed=today

    if op[0][0]==None:

        eid='E001'

        ipass='#'+today[2:4]+today[5:7]+today[8:]+'01#'

    else:

        maxeid=op[0][0] #The last eid in the table

        maxde=str(op[0][1]) #The last date_employed in the table

        maxpass=op[0][2] #The last ipass in the table

        eid="E"+str(int(maxeid[1:])+1)

        while len(eid)<4:

            eid=eid[0]+'0'+eid[1:]

        if maxde==today:

            ipassnum=str(int(maxpass[7:9])+1)
```

```
            ipassnum='0'+ipassnum if len(ipassnum)==1 else ipassnum

        else:

            ipassnum='01'

        ipass='#'+today[2:4]+today[5:7]+today[8:]+ipassnum+'#'

    post='null' if post=='' else "'"+post+"'"

    sal='null' if sal=='' else sal

    cursor.execute(f"insert into staff(eid,ipass,date_employed,post,sal)
   values('{eid}','{ipass}','{date_employed}',{post},{sal})")

    mydb.commit()

    cursor.execute("select eid,name,username,post,mobile,sal,gender,dob,ipass,date_employed from
staff")

    for i in tree.get_children():

        tree.delete(i)

    for i in cursor.fetchall():

        tree.insert('',"end",values=(i))

    mydb.close()


def remove(tree,fr1):

    for widget in fr1.winfo_children():

        widget.destroy()

    Label(fr1,text="Enter ID of employee to remove: ").grid(row=0,column=0)

    eid=StringVar()

    Entry(fr1,textvariable=eid).grid(row=0,column=1)

    error=Label(fr1,text="Please enter a valid ID.")

    Button(fr1,text="REMOVE",command=lambda: click_remove(tree,eid.get(),error)).grid(row=0,column=2)


def click_remove(tree,eid,error):

    mydb.connect()

    cursor.execute(f"delete from staff where eid='{eid}'")

    mydb.commit()

    if cursor.rowcount==0:

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

    else:

        error.grid_forget()

        cursor.execute("select eid,name,username,post,mobile,sal,gender,dob,ipass,date_employed from
staff")

        for i in tree.get_children():

            tree.delete(i)

        for i in cursor.fetchall():

            tree.insert('',"end",values=(i))
```

```
    mydb.close()


def update_employee(tree,fr1):
    for widget in fr1.winfo_children():
        widget.destroy()
    eid,new_post,new_sal=StringVar(),StringVar(),StringVar()
    error=Label(fr1,text="Enter a valid ID.")
    Label(fr1,text="Add Employee ID: ").grid(row=0,column=0)
    Entry(fr1,textvariable=eid).grid(row=0,column=1)
    Label(fr1,text="Add new post: ").grid(row=0,column=2)
    Entry(fr1,textvariable=new_post).grid(row=0,column=3)
    Label(fr1,text="Add new salary: ").grid(row=0,column=4)
    Entry(fr1,textvariable=new_sal).grid(row=0,column=5)
   Button(fr1,text="UPDATE",command=lambda:
   click_upd_emp(tree,error,eid.get(),new_post.get(),new_sal.get())).grid(row=0,column=6)


def click_upd_emp(tree,error,eid,new_post,new_sal):
    mydb.connect()
    cursor.execute(f"select post,sal from staff where eid='{eid}'")
    op=list(cursor.fetchall()[0])
    if cursor.rowcount==0:
        error.grid(row=0,column=7)
    else:
        for i in range(2):
            if op[i]==None:
                op[i]='null'
            elif i==0:
                op[i]=f"'{op[i]}'"


        error.grid_forget()
        new_post=op[0] if new_post=='' else "'"+new_post+"'"
        new_sal=op[1] if new_sal=='' else new_sal
        cursor.execute(f"update staff set post={new_post},sal={new_sal} where eid='{eid}'")
        mydb.commit()
        cursor.execute("select eid,name,username,post,mobile,sal,gender,dob,ipass,date_employed from
staff")
        for i in tree.get_children():
            tree.delete(i)
        for i in cursor.fetchall():
            tree.insert('',"end",values=(i))
```

```
        mydb.close()


def billing(caller,post):
    caller.destroy()
    billpage=Tk()
    billpage.title("Billing")
    topbuttons=Frame(billpage)
    topbuttons.grid(row=0,sticky=W)
    Button(topbuttons,text="LOGOUT",command=lambda: LoginPage(billpage)).grid(row=0,column=1)
    Button(topbuttons,text="BACK",command=lambda: MainMenu(billpage,post)).grid(row=0,column=0)
    customer=Frame(billpage)
    customer.grid(row=3,column=0,sticky=W)
    Label(billpage,text="Customer details:\n").grid(row=2,column=0,sticky=W)
    Label(customer,text="\tName: ").grid(row=0,column=0,sticky=W)
    Label(customer,text="\tPhone Number: ").grid(row=1,column=0,sticky=W)
    Label(customer,text="\tGender ").grid(row=2,column=0,sticky=NW)
    Label(customer,text="\t  Date of Birth: ").grid(row=1,column=2,sticky=W)
    mydb.connect()
    cursor.execute("select max(billno) from history")
    op=cursor.fetchall()
    if op[0][0]==None:
        billno="B0001"
    else:
        billno="B"+str(int(op[0][0][1:])+1)
        while len(billno)<5:
            billno=billno[0]+'0'+billno[1:]
    Label(billpage,text=f"Bill number: {billno}",font="Helvetica 10
bold").grid(row=1,column=0,sticky=W)
    name1,ph1,gender,dob1=StringVar(),StringVar(),StringVar(),StringVar()
    e1=Entry(customer, textvariable=name1,width=40)
    e1.grid(row=0,column=1,sticky=W,columnspan=3)
    e2=Entry(customer, textvariable=ph1,width=20)
    e2.grid(row=1,column=1,sticky=W)
    e3=Entry(customer, textvariable=dob1,width=20)
    e3.grid(row=1,column=3,sticky=W)
    gen=Frame(customer)
    gen.grid(row=2,column=1,sticky=W,columnspan=2)
    Radiobutton(gen,text="Prefer not to
specify",variable=gender,value="null").grid(row=0,column=0,columnspan=2,sticky=W)
    Radiobutton(gen,text="Male",variable=gender,value="'M'").grid(row=1,column=0,sticky=W)
```

```
    Radiobutton(gen,text="Female",variable=gender,value="'F'").grid(row=1,column=1,sticky=W)
    Radiobutton(gen,text="Others",variable=gender,value="'O'").grid(row=1,column=2,sticky=W)
    Label(billpage,text="Invoice:\n").grid(row=4,column=0,sticky=W)
    Bill=Frame(billpage)
    Bill.grid(row=5,column=0,sticky=W)
    Label(Bill, text="\tBatch Code:    ").grid(row=0,column=0,sticky=E)
    Label(Bill, text="Item Name:    ").grid(row=0,column=2,sticky=E)
    Label(Bill, text="Quantity:    ").grid(row=0,column=4,sticky=E)
    batch,iname,qty=StringVar(),StringVar(),StringVar()
    e4=Entry(Bill,textvariable=batch)
    e4.grid(row=0,column=1,sticky=W)
    e5=Entry(Bill,textvariable=iname,width=30)
    e5.grid(row=0,column=3,sticky=W)
    e6=Entry(Bill,textvariable=qty,width=10)
    e6.grid(row=0,column=5,sticky=W)
    entries=[e1,e2,e3,e4,e5,e6]
    record=[]
    TotalValue=[0]
    TotalFrame=Frame(Bill)
    TotalFrame.grid(row=3,column=5)
    error1=Label(Bill,text="Please fill the 3 fields.")
    error2=Label(Bill,text="Such item does not exist.")
    error3=Label(Bill,text="Entered quantity exceeds existing quantity.")
    error4=Label(Bill,text="Please make at least ONE entry.")
    cust_error1=Label(Bill, text="Please enter mobile number of the customer.")
    cust_error2=Label(Bill, text="Please enter name of the new customer.")
    Button(Bill,text="ADD ENTRY",command=lambda: add_entry(Bill,tree,record, entries,
                                                            TotalValue,TotalFrame,
                                                            batch.get(),
                                                            iname.get(),
                                                            qty.get(),

error1,error2,error3)).grid(row=1,sticky=E,ipadx=4)
    Button(Bill,text="MAKE BILL",command=lambda: click_makebill(billno,
                                                          name1.get(),
                                                          ph1.get(),
                                                          dob1.get(),
                                                          gender.get(),
                                                          record,entries,
```

```
                                                        cust_error1,cust_error2,

error4,tree,TotalFrame,billpage,post)).grid(row=1,column=1,sticky=W)

    tree=Treeview(Bill)

    tree["columns"]=(0,1,2,3,4,5,6,7)

    tree.column("#0",width=0)

    tree.column(0,width=100)

    tree.column(1,width=100)

    tree.column(2,width=100)

    tree.column(3,width=120)

    tree.column(4,width=50)

    tree.column(5,width=50)

    tree.column(6,width=76)

    tree.column(7,width=64)

    tree.heading(0,text="seriel NO.")

    tree.heading(1,text="MANUFACTURER")

    tree.heading(2,text="PRODUCT NAME")

    tree.heading(3,text="BATCH CODE")

    tree.heading(4,text="QTY")

    tree.heading(5,text="RATE")

    tree.heading(6,text="EXPIRY")

    tree.heading(7,text="PRICE")

    tree.grid(row=2,column=0,columnspan=6,sticky=W)

    vsb=Scrollbar(Bill,orient="vertical",command=tree.yview)

    tree.configure(yscrollcommand=vsb.set)

    vsb.grid(row=2,column=6,sticky=NS)

    Label(Bill,text="TOTAL:",font="Helvetica 10 bold").grid(row=3,column=4,sticky=E)

    billpage.mainloop()


def add_entry(Bill,tree,record,entries,TotalValue,TotalFrame,batch,iname,qty,error1,error2,error3):

    if batch=='' or iname=='' or qty==='':

        error1.grid(row=3,column=0,columnspan=2,sticky=W)

    else:

        error1.grid_forget()

        mydb.connect()

        cursor.execute(f"select serielno,manufacturer,qty,rate,expiry from stock where
        itemname='{iname}' and batchcode='{batch}'")

        result=cursor.fetchall()

        if cursor.rowcount==0:

            error2.grid(row=3,column=0,columnspan=2,sticky=W)
```

```
        else:
            error2.grid_forget()
            result=result[0]
            if int(qty)>result[2]:
                error3.grid(row=3,column=0,columnspan=3,sticky=W)
            else:
                error3.grid_forget()
                for i in entries[3:]:
                    i.delete(0,100)
                record.append((result[0],
                               result[1],
                               iname,
                               batch,
                               qty,
                               result[3],
                               result[4],
                               int(qty)*result[3]))
                tree.insert('',"end",values=(record[-1]))
                TotalValue[0]+=record[-1][-1]
                for i in TotalFrame.winfo_children():
                    i.destroy()
                Label(TotalFrame,text=str(TotalValue[0]),font='Helvetica 10
bold').grid(row=0,column=0,sticky=E)



def
click_makebill(billno,name,mob,dob,gender,record,entries,error1,error2,error3,tree,TotalFrame,billpage
,post):
    if mob=='':
        error1.grid(row=3,column=0,columnspan=3,sticky=W)
    else:
        error1.grid_forget()
        error3.grid_forget()
        mydb.connect()
        cursor.execute(f"select * from customers where mob='{mob}'")
        cursor.fetchall()
        if cursor.rowcount==0:
            if name=='':
                error2.grid(row=3,column=0,columnspan=3,sticky=W)
            else:
```

```
                error2.grid_forget()
                dob="null" if dob=='' else "'"+dob+"'"
                query=f"insert into customers values('{mob}','{name}',{gender},{dob})"
                cursor.execute(query)
                mydb.commit()
                addbill(billno,mob,record,entries,error3,tree,TotalFrame,billpage,post)
        else:
            addbill(billno,mob,record,entries,error3,tree,TotalFrame,billpage,post)


def addbill(billno,mob,record,entries,error,tree,TotalFrame,billpage,post):
        if record==[]:
            error.grid(row=3,column=0,columnspan=3,sticky=W)
        else:
            error.grid_forget()
            today=str(datetime.date.today())
            for i in record:


                cursor.execute(f"insert into history values('{billno}','{i[0]}',\
                                                            '{i[1]}','{i[2]}',\
                                                            '{i[3]}',{i[4]},\
                                                            {i[5]},'{i[6]}',\
                                                            '{mob}',curdate())")
                cursor.execute(f"update stock set qty=qty-{i[4]} where batchcode='{i[3]}' and
Itemname='{i[2]}'")
                cursor.execute("delete from stock where qty=0")
            mydb.commit()
            mydb.close()
            for i in tree.get_children():
                tree.delete(i)
            for i in entries:
                i.delete(0,100)
            for i in TotalFrame.winfo_children():
                i.destroy()
            billing(billpage,post)


def history(caller,post):
    caller.destroy()
    billhist=Tk()
    billhist.title("Billing History")
```

```
topbuttons=Frame(billhist)

topbuttons.grid(row=0,sticky=W)

Button(topbuttons,text="LOGOUT",command=lambda: LoginPage(billhist)).grid(row=0,column=1)

Button(topbuttons,text="BACK",command=lambda: MainMenu(billhist,post)).grid(row=0,column=0)

billno=StringVar()

infoframe=Frame(billhist)

infoframe.grid(row=1,column=0,sticky=W)

Label(infoframe,text="Enter Bill number: ").grid(row=0,column=0,sticky=W)

e1=Entry(infoframe,textvariable=billno)

e1.grid(row=0,column=1,sticky=W)

Button(infoframe,text="SEARCH",command=lambda:
click_search(tree,Custdetails,billno.get(),e1)).grid(row=0,column=2,sticky=W)

tree=Treeview(billhist)

tree["columns"]=(0,1,2,3,4,5,6,7)

tree.column("#0",width=0)

tree.column(0,width=100)

tree.column(1,width=100)

tree.column(2,width=100)

tree.column(3,width=120)

tree.column(4,width=50)

tree.column(5,width=50)

tree.column(6,width=76)

tree.column(7,width=64)

tree.heading(0,text="seriel NO.")

tree.heading(1,text="MANUFACTURER")

tree.heading(2,text="PRODUCT NAME")

tree.heading(3,text="BATCH CODE")

tree.heading(4,text="QTY")

tree.heading(5,text="RATE")

tree.heading(6,text="EXPIRY")

tree.heading(7,text="PRICE")

tree.grid(row=2,column=0,columnspan=3)

vsb=Scrollbar(billhist,orient="vertical",command=tree.yview)

tree.configure(yscrollcommand=vsb.set)

vsb.grid(row=2,column=3,sticky=NS)

Custdetails=Frame(billhist)

Custdetails.grid(row=3,column=0,sticky=W)

billhist.mainloop()
```

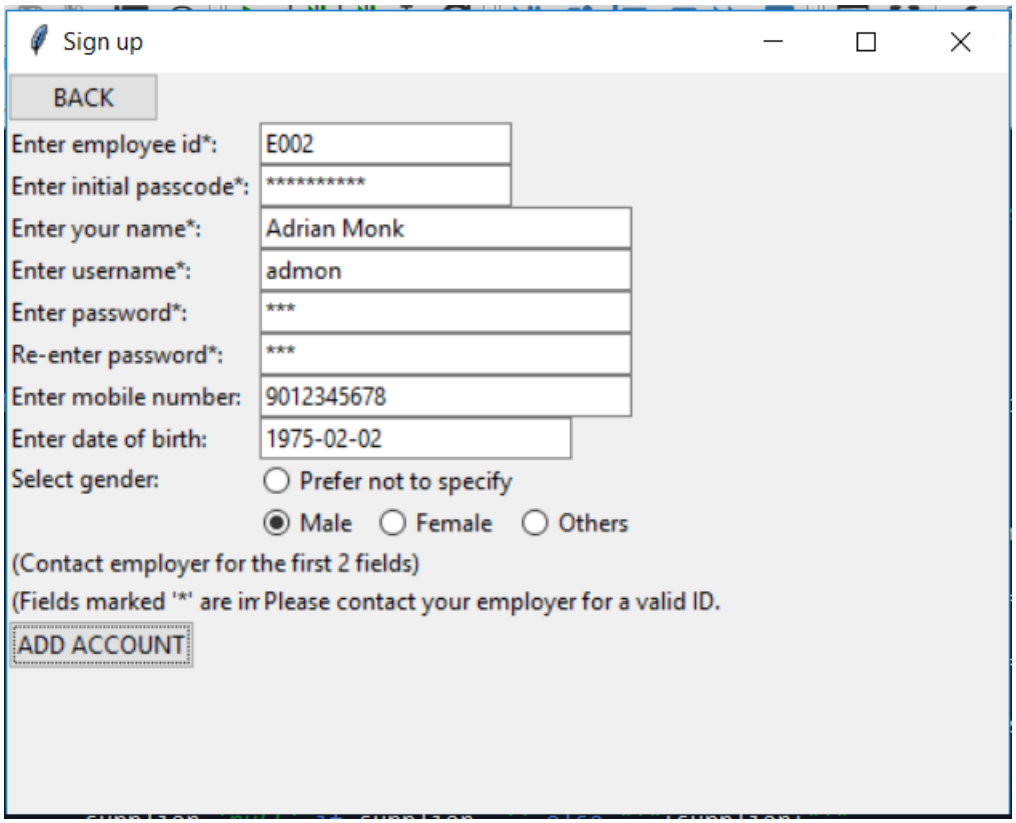
```
def click_search(tree,custframe,billno,e1):
    mydb.connect()
    for widget in custframe.winfo_children():
        widget.destroy()
    for i in tree.get_children():
        tree.delete(i)
    cursor.execute(f"select *,qty*rate from history,customers where history.custmob=customers.mob and
billno='{billno}'")
    result=cursor.fetchall()
    mydb.close()
    if cursor.rowcount==0:
        Label(custframe,text="Enter a valid bill number.").grid(row=0,column=0)
    else:
        total=0
        e1.delete(0,100)
        for i in tree.get_children():
            tree.delete(i)
        for i in result:
            entry=i[1:8]+tuple([i[-1]])
            tree.insert('',"end",values=(entry))
            total+=i[-1]
        name=result[0][11]
        mob=result[0][10]
        date=result[0][9]
        gender=result[0][12]
        dob=result[0][13]
        if gender==None:
            gender="Not specified"
        if dob==None:
            gender="Not specified"
        Label(custframe,text=f"CUSTOMER NAME: {name}").grid(row=0,sticky=W)
        Label(custframe,text=f"CONTACT NUMBER: {mob}").grid(row=1,sticky=W)
        Label(custframe,text=f"GENDER: {gender}").grid(row=2,sticky=W)
        Label(custframe,text=f"DATE OF BIRTH: {dob}").grid(row=3,sticky=W)
        Label(custframe,text=f"DATE PURCHASED: {date}").grid(row=4,sticky=W)
        Label(custframe,text=f"TOTAL COST: Rs. {total}",font='Helvetica 10 bold').grid(row=5,sticky=W)


def customers(caller,post):
    caller.destroy()
```

```
    custpage=Tk()

    custpage.title("Customers")

    topbuttons=Frame(custpage)

    topbuttons.grid(row=0,sticky=W)

    Button(topbuttons,text="LOGOUT",command=lambda: LoginPage(custpage)).grid(row=0,column=1)

    Button(topbuttons,text="BACK",command=lambda: MainMenu(custpage,post)).grid(row=0,column=0)

    search=Frame(custpage)

    search.grid(row=1,sticky=W)

    Label(search,text="Enter contact number: ").grid(row=0,column=0)

    mob=StringVar()

    e1=Entry(search,textvariable=mob)

    e1.grid(row=0,column=1)

    Button(search,text="SEARCH",command=lambda:
click_cust_search(tree,error,mob.get(),e1)).grid(row=0,column=2)

    Button(search,text="SHOW ALL",command=lambda: click_showall(tree,error,e1)).grid(row=0,column=3)

    tree=Treeview(custpage)

    tree["columns"]=(0,1,2,3)

    tree.column("#0",width=0)

    tree.column(0,width=100)

    tree.column(1,width=100)

    tree.column(2,width=80)

    tree.column(3,width=120)

    tree.heading(0,text="CONTACT")

    tree.heading(1,text="NAME")

    tree.heading(2,text="GENDER")

    tree.heading(3,text="DATE OF BIRTH")

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

    vsb=Scrollbar(custpage,orient="vertical",command=tree.yview)

    tree.configure(yscrollcommand=vsb.set)

    vsb.grid(row=2,column=1,sticky=NS)

    mydb.connect()

    cursor.execute("select * from customers order by name asc")

    for i in cursor.fetchall():

        tree.insert('',"end",values=(i))

    error=Label(custpage,text="Please enter a valid contact numer")

    mydb.close()

    custpage.mainloop()


def click_cust_search(tree,error,mob,e1):
```

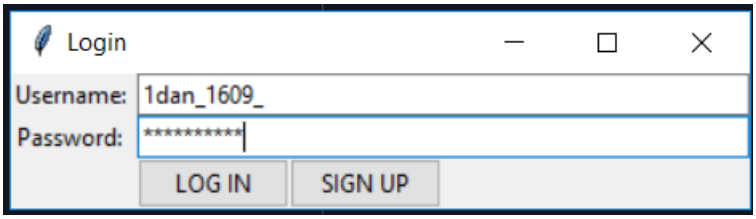
```
    mydb.connect()
    cursor.execute(f"select * from customers where mob='{mob}'")
    result=cursor.fetchall()
    mydb.close()
    if cursor.rowcount==0:
        error.grid(row=3,column=0,sticky=W)
    else:
        e1.delete(0,100)
        error.grid_forget()
        for i in tree.get_children():
            tree.delete(i)
        tree.insert('','end',values=(result[0]))

def click_showall(tree,error,e1):
    error.grid_forget()
    e1.delete(0,100)
    for i in tree.get_children():
        tree.delete(i)
    mydb.connect()
    cursor.execute("select * from customers order by name asc")
    for i in cursor.fetchall():
        tree.insert('',"end",values=(i))
    mydb.close()

LoginPage()
```

Implementation:

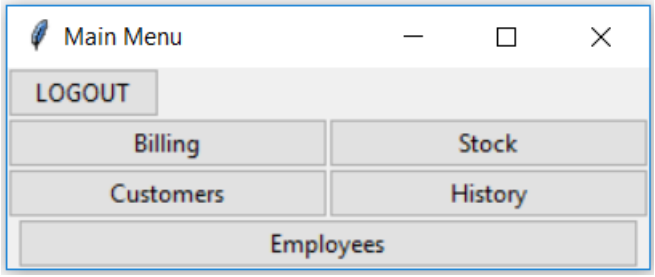
SIGN UP



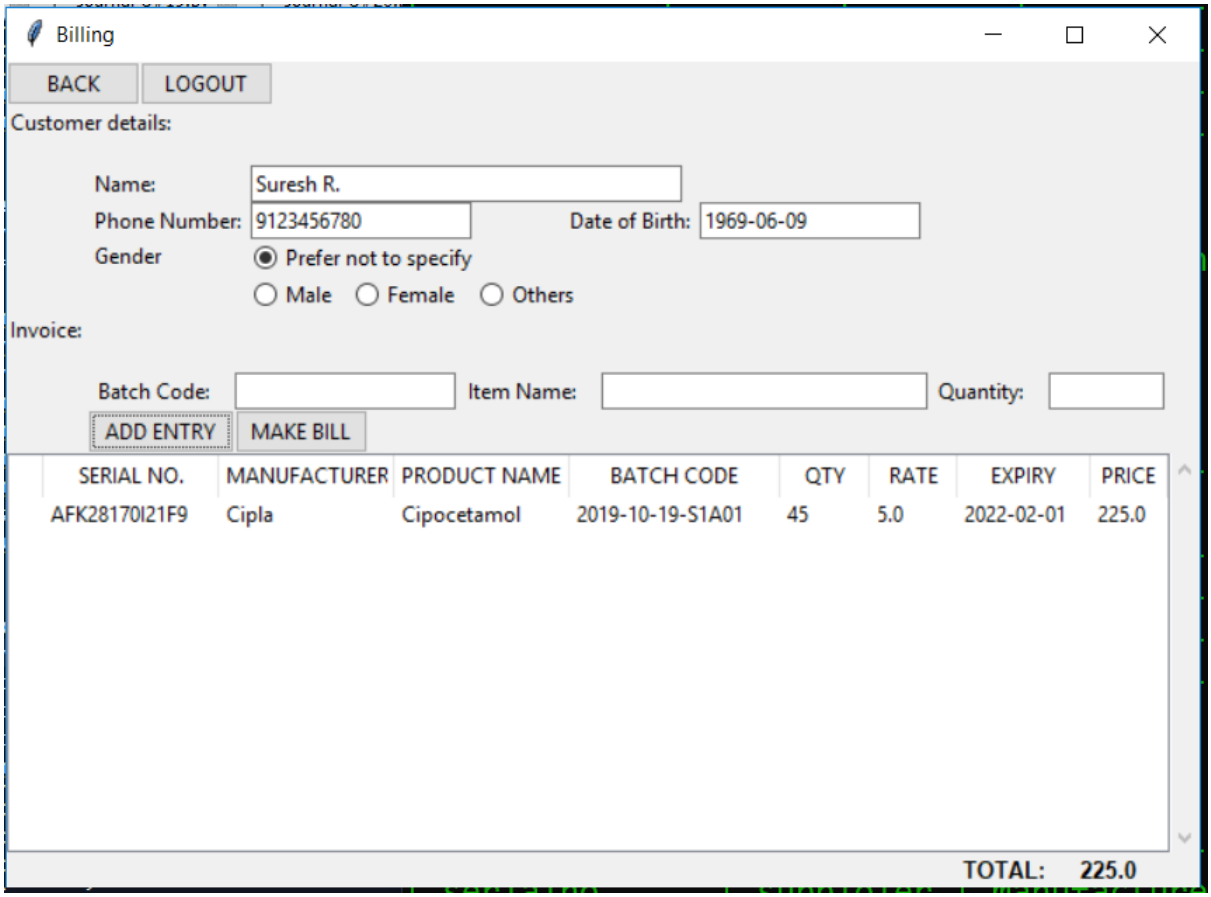
Login



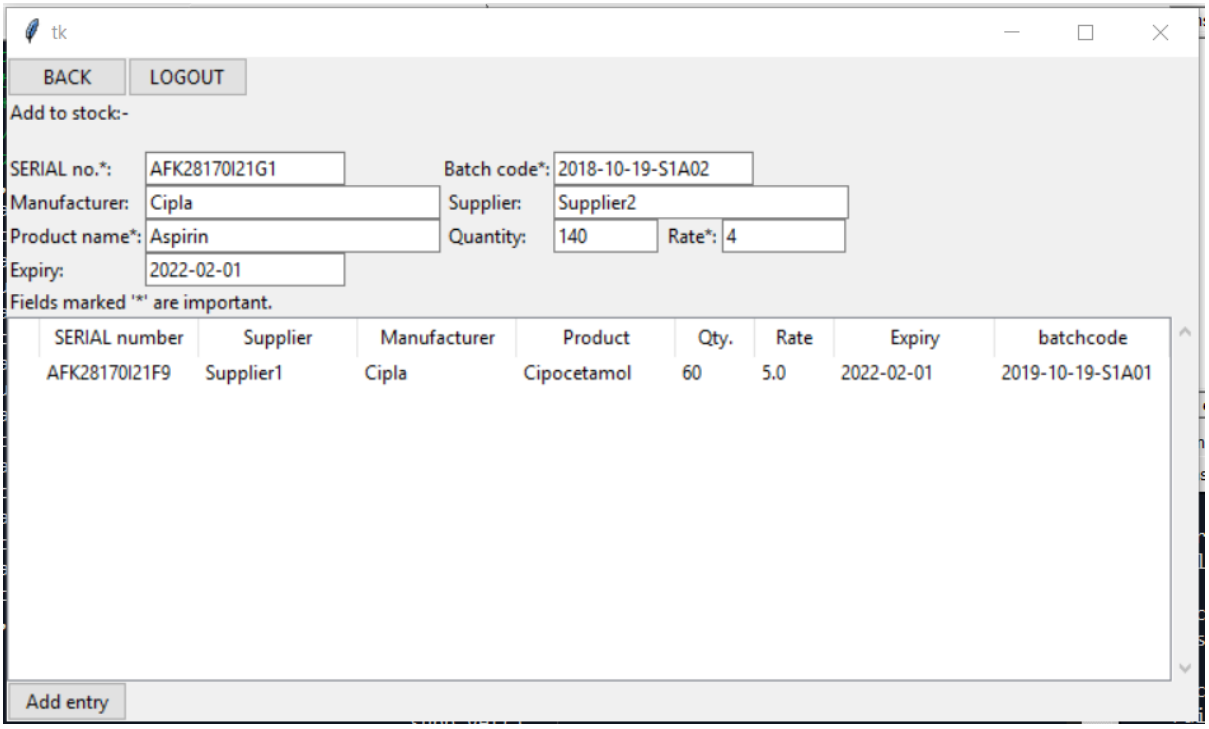
Main Menu



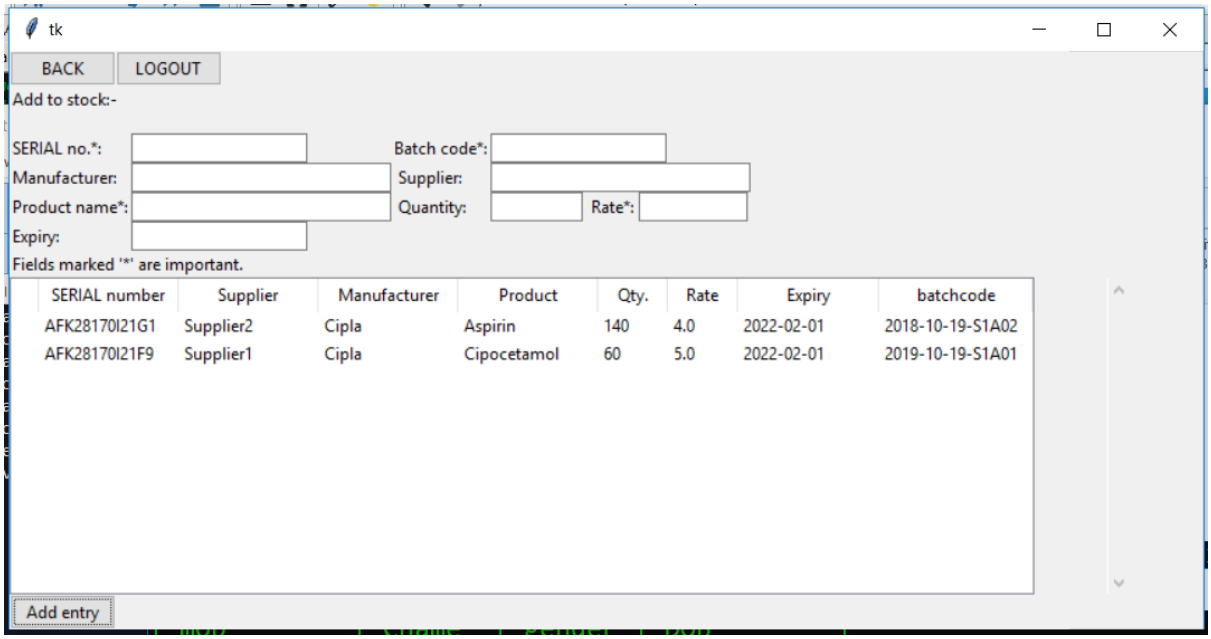
Billing



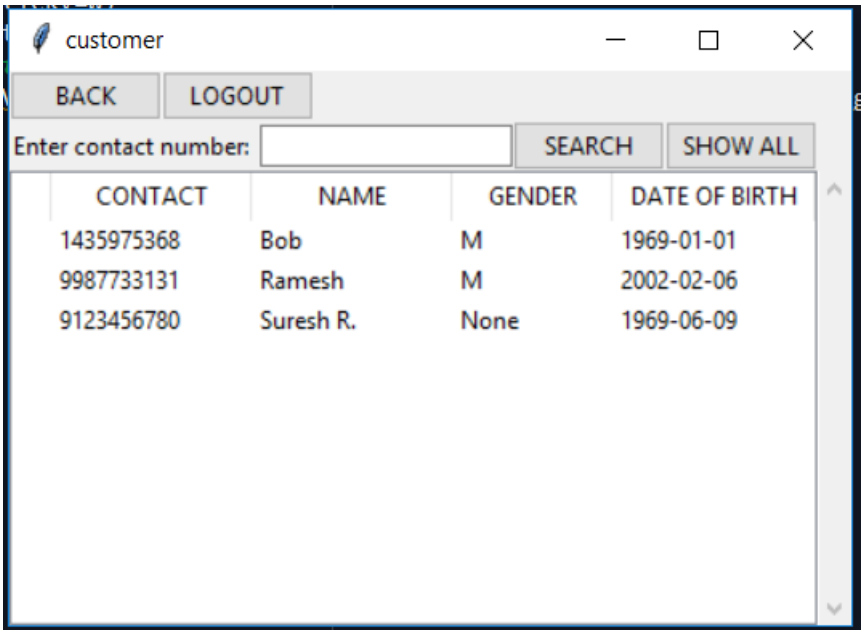
Stock (Before ADD ENTRY)



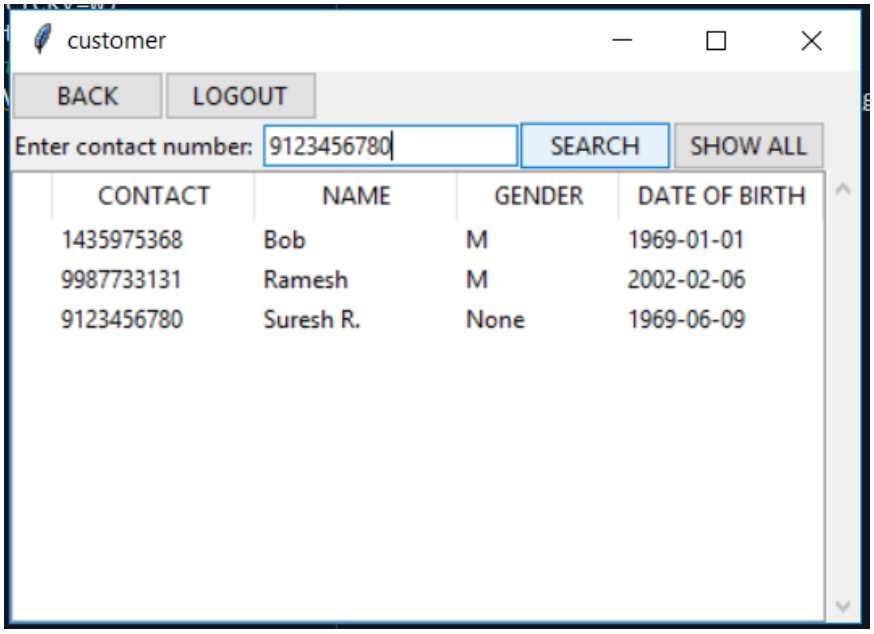
Stock (After ADD ENTRY)



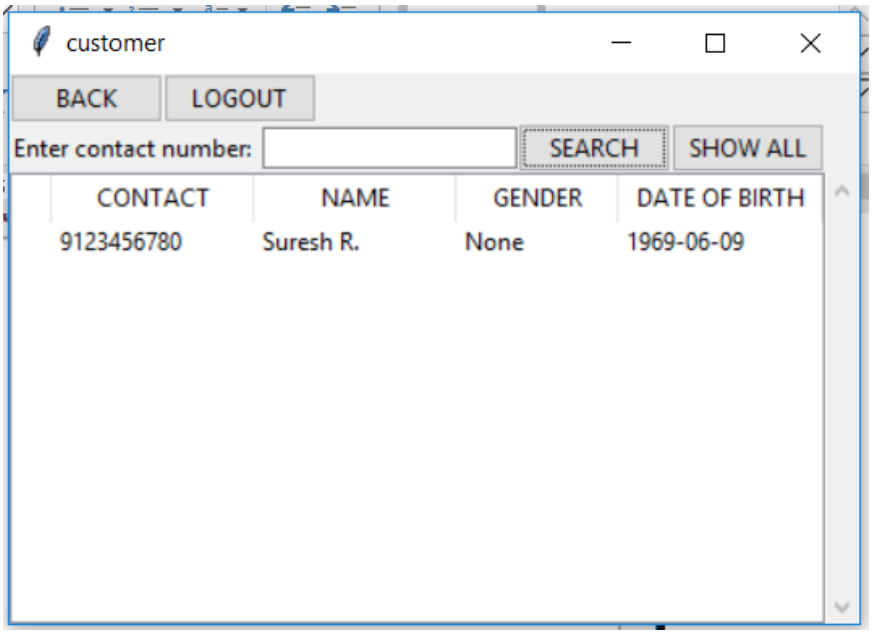
Customer (SHOW ALL)



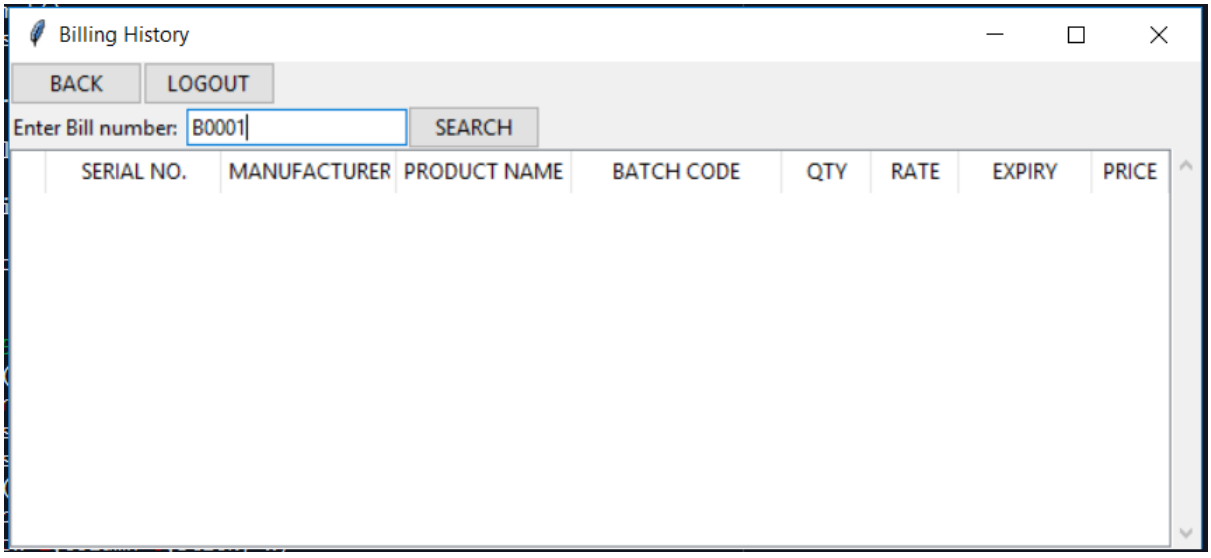
Customer (Before SEARCH)



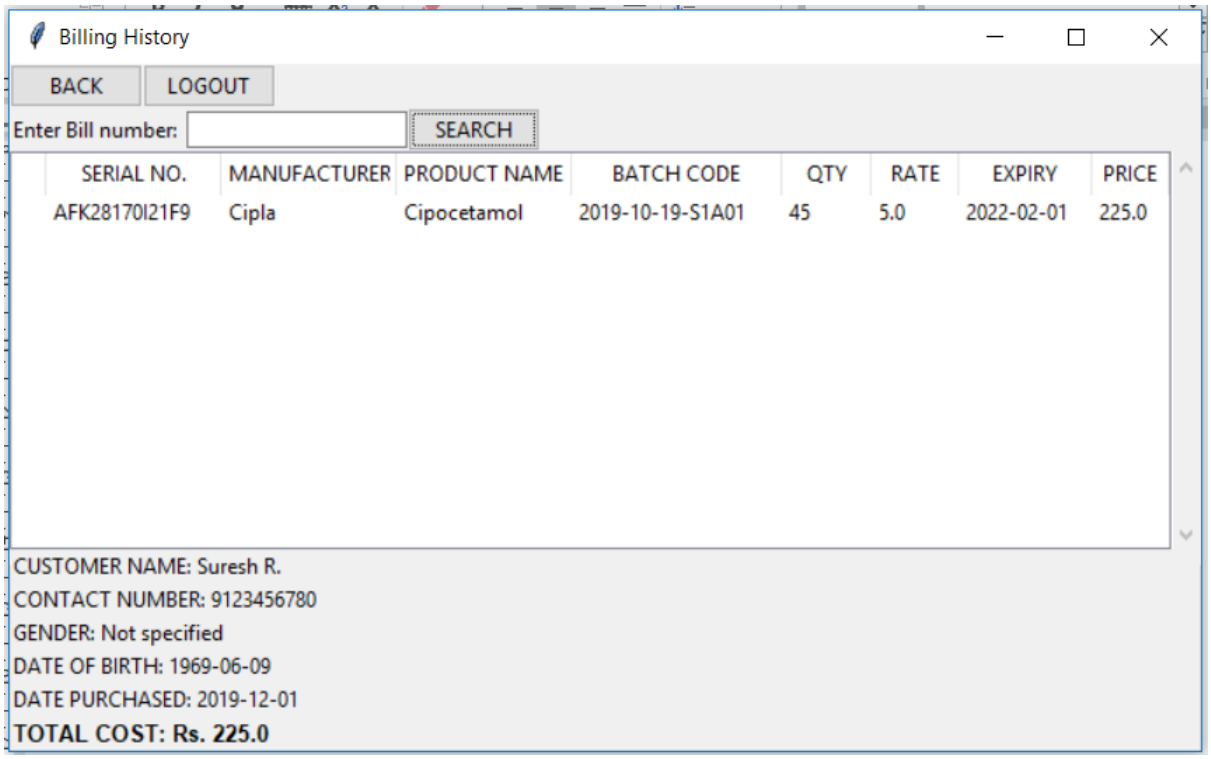
Customer (After SEARCH)



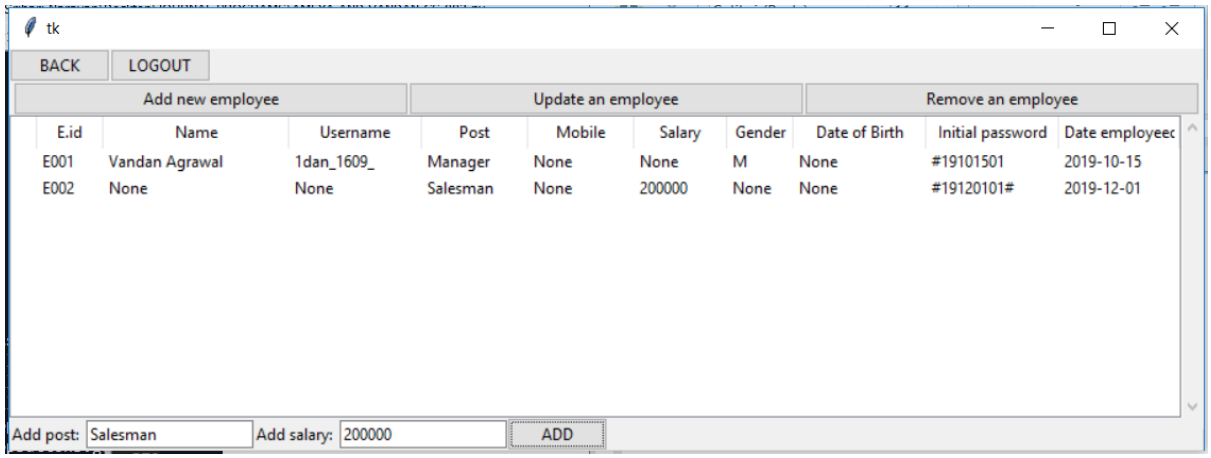
History (Before SEARCH)



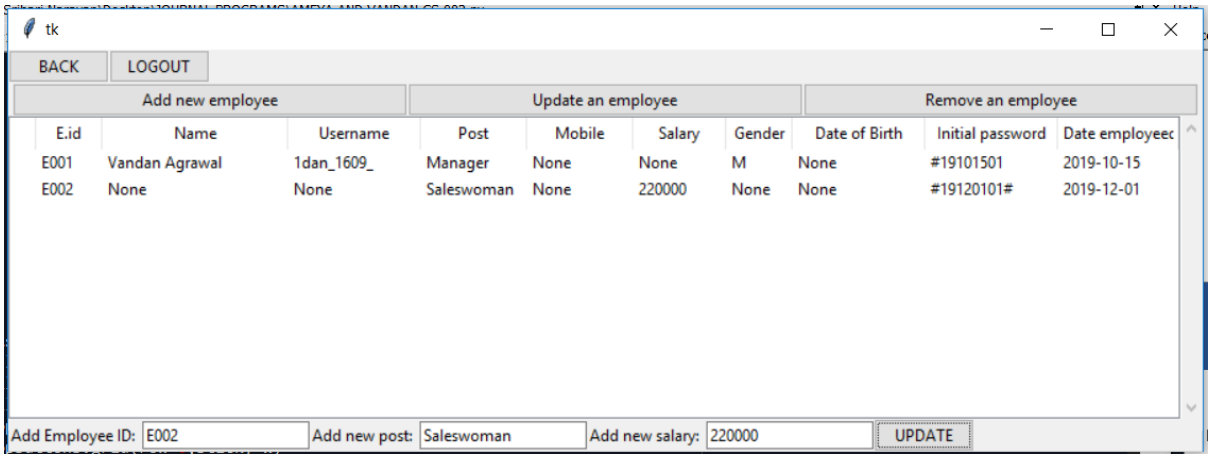
History (After SEARCH)



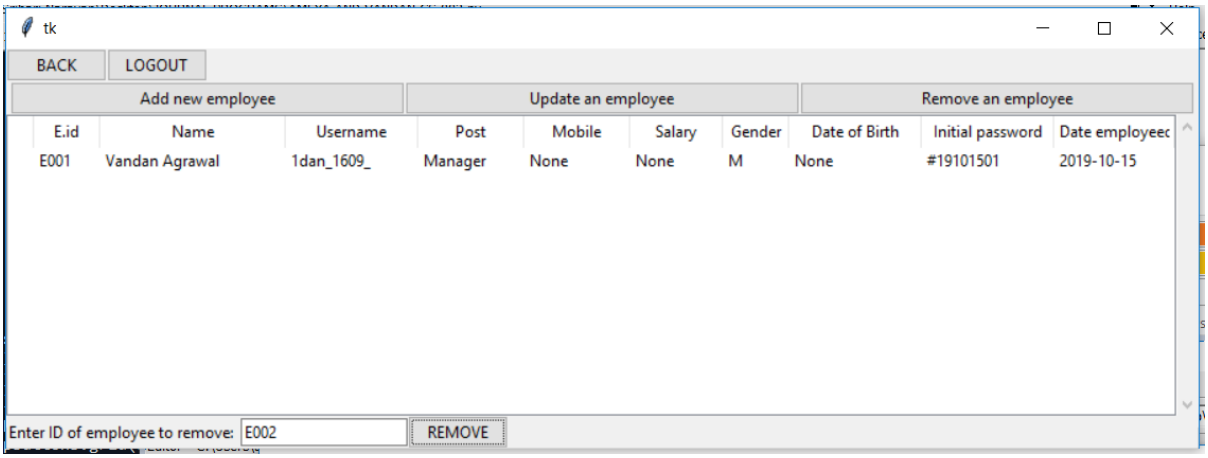
Employees (ADD NEW EMPLOYEE)



Employee (UPDATE AN EMPLOYEE)



Employee (REMOVE AN EMPLOYEE)



Future enhancements

We can add many more improvements to the program such as modifying the styles of the windows and the using of new textures.

We can also add a way to order new stocks automatically or maybe inform the company the moment stocks of a particular medicine/item are running low so that new stocks may be ordered immediately.

Have customer personalised orders so that in case say for example someone needs a month of medication twice a year, the next time they show up to purchase their one-month stock and in case the stock for only 22 days is available they can be given a notification.

Conclusion:

With the ever-increasing amount of big data in today's world we need to find better and more systematic ways of storing and accessing this data.

This project was made to show how a medical store manages the large amounts of data they receive and store it in a structured format(tables) and access this data whenever required and now with this we can see how the handling of data at the backend works with regards to storing it.

BIBLIOGRAPHY

- https://web.archive.org/web/20171018181046/https://spotlessdata.com/blog/exploring-data-analysis
- https://en.wikipedia.org/wiki/Data_science
- https://www.tableau.com/learn/articles/what-is-data-cleaning
- https://www.sas.com/en_in/insights/big-data/what-is-big-data.html
- Stack Overflow for bug fixing
- YouTube video of freecodecamp.org related to tkinter