Advance Programming in Python

Project Title: Inventory management system for pharmacy

Project report

Group members:

Haris Jamal Khan   362535

Aqeel Ahmad   364244

Ammad-ud-din Ghakkar   364715

Project Scope:

In this project we aim to develop an inventory management system for pharmacy. We will create a desktop app using Python frameworks to helps the manager/owner reduce cost, improve operational efficiency, and minimize overstocking and opportunity loss. For a person to effectively manage the inventory, he or she should have complete visibility of the current stock, prices and details of the product. To develop an inventory management system, we use following frameworks.

1. For front end development we have used PyQt5, Qt designer,

2. To store data we have created an inventor using pandas (Python Library)

Task Distribution:

Our inventory management system will have six different features. Each group member will be responsible for development for two features. Details of features is given in the table.

S. NO	Feature	Group Member
1	Store man login/admin login/register new user (front end)	
3	Integration of front end with backend	Aqeel Ahmed
4	New user registration/login (backend)	
5	Add new product (backend)	Haris Jamal Khan
5	Consume products (backend)	
6	Update product	Amad ud Din

GUI pages and descriptions

This is the main login screen

It has the following functionality

- Admin login
- Storeman login
- New user registration

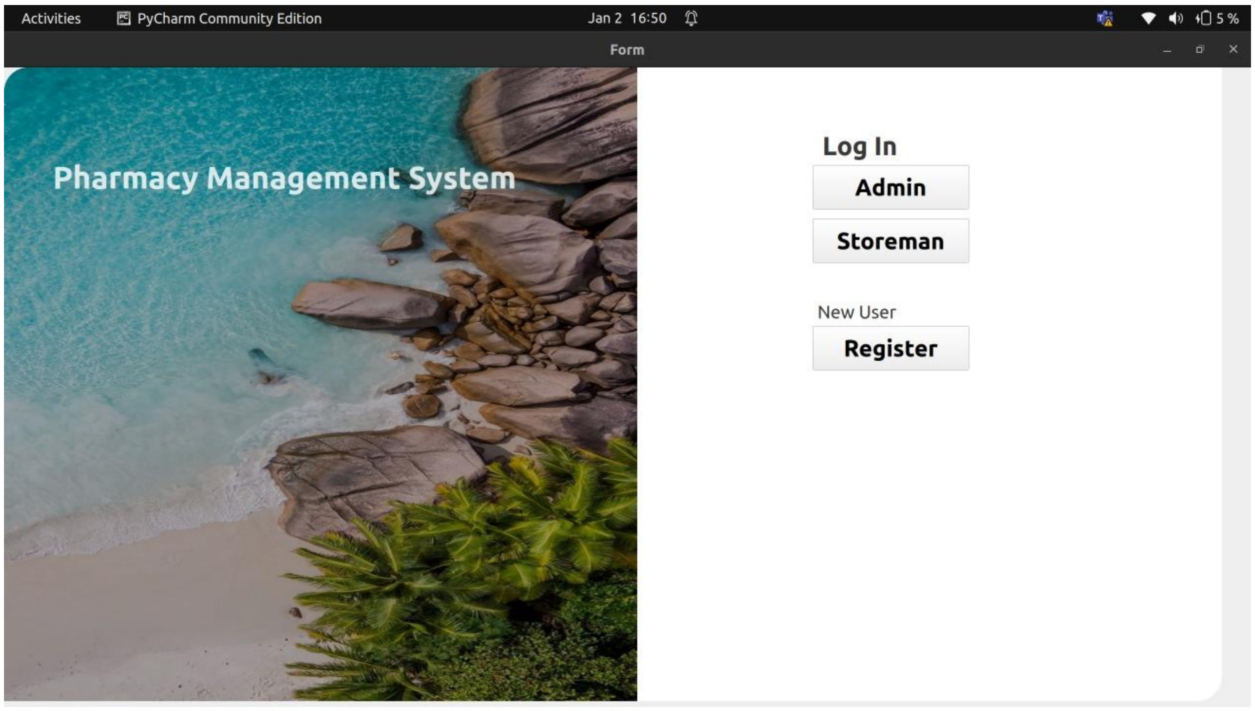


Figure 1 Main login page

Admin login

Note. one admin is present at this stage and has the folowing credentials

Username: admin

Password: admin123

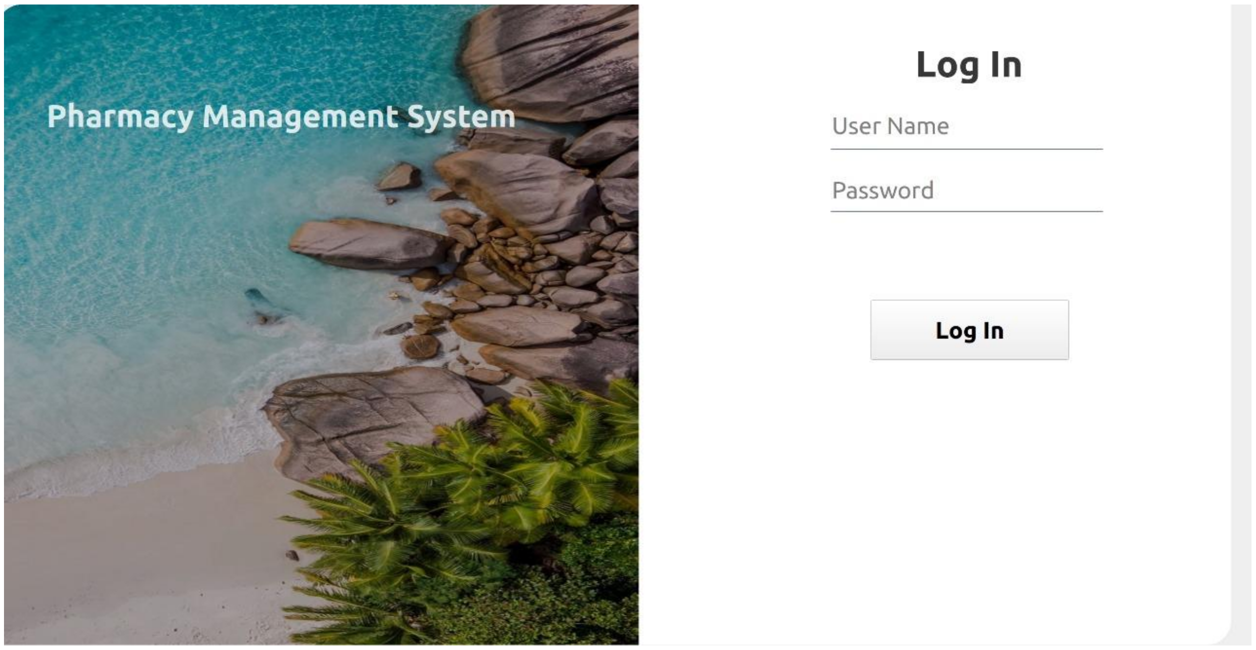
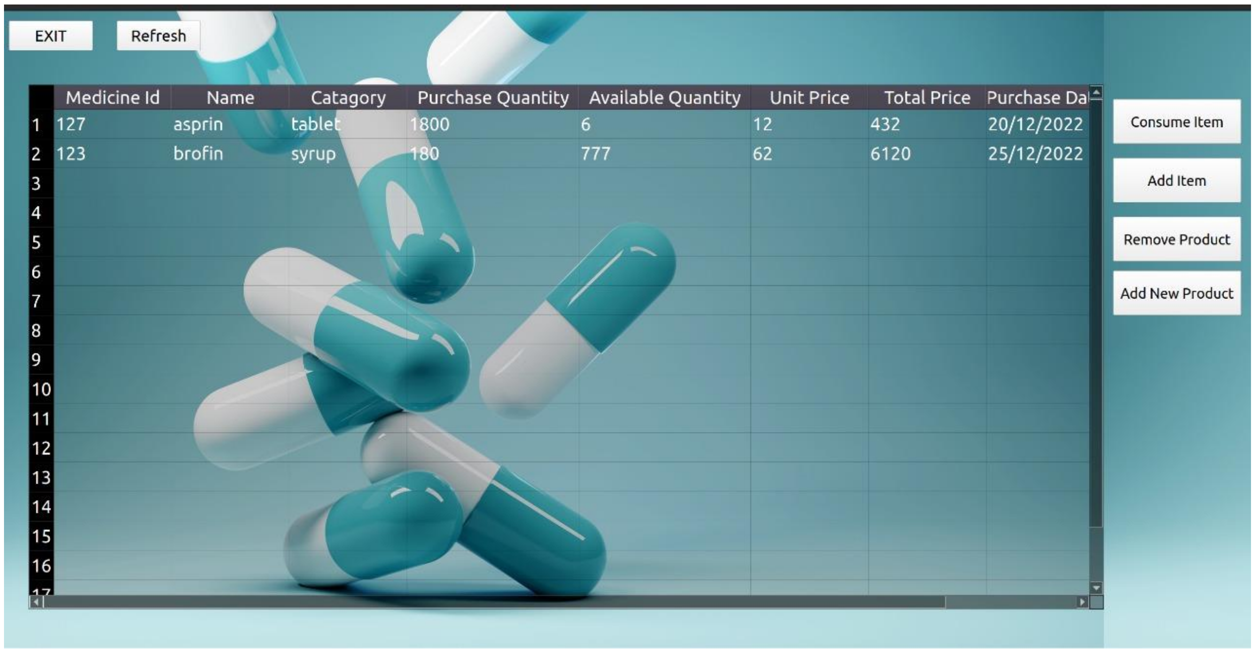


Figure 2 Admin login page

If a user login as an Admin it can perform the flowing functions

- Consume item
- Add item
- Remove product
- Add new product



	Medicine Id	Name	Catagory	Purchase Quantity	Available Quantity	Unit Price	Total Price	Purchase Da
1	127	asprin	tablet	1800	6	12	432	20/12/2022
2	123	brofin	syrup	180	777	62	6120	25/12/2022
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								

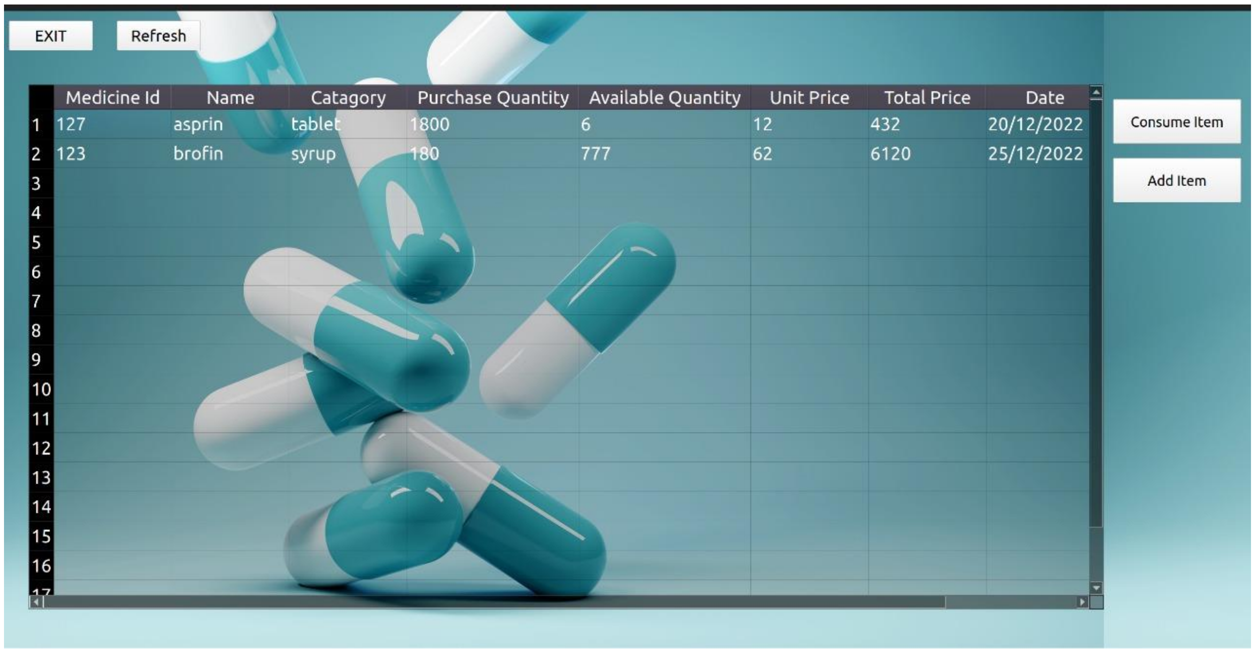
Figure 3 Admin inventory page

Storeman login

A user can login using storeman credentials if present, otherwise user can register by providing username and password

If a user login as a storeamn it can perform the following functions

- Cosume item
- Add item



	Medicine Id	Name	Catagory	Purchase Quantity	Available Quantity	Unit Price	Total Price	Date
1	127	asprin	tablet	1800	6	12	432	20/12/2022
2	123	brofin	syrup	180	777	62	6120	25/12/2022

Figure 4 Storeman inventory page

New product: admin can enter a new product in the database by providing the following informatioon

- Product name
- Product id
- Quantity
- Unit price
- Date of purchase

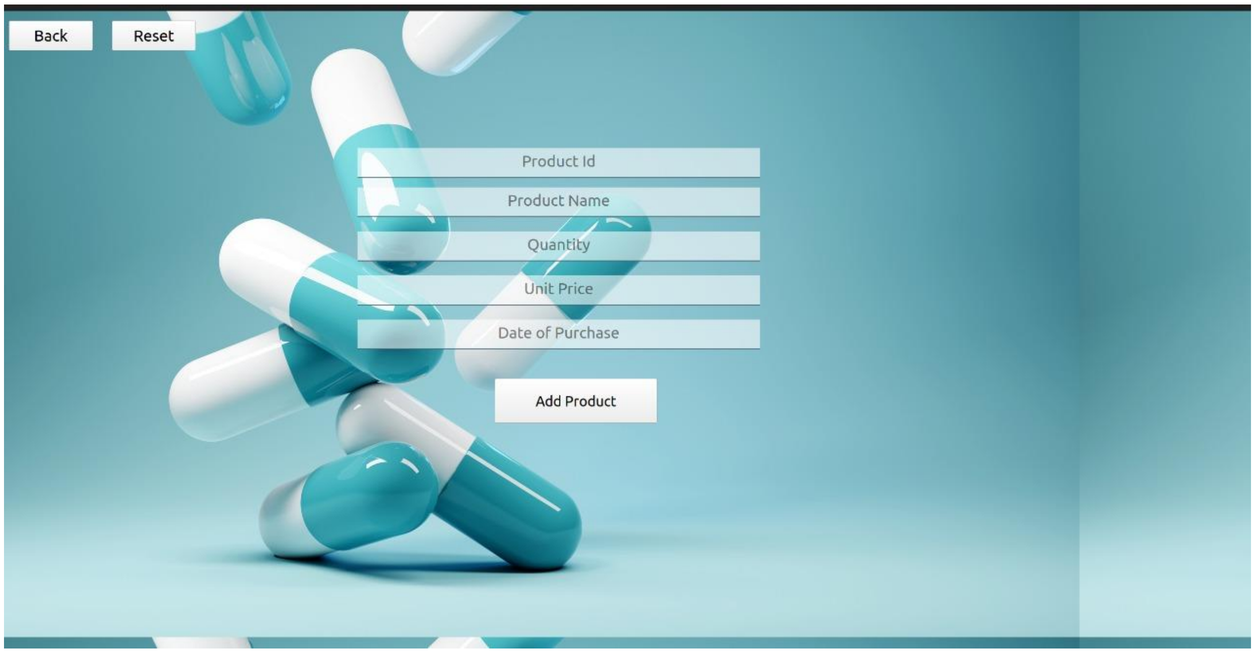


Figure 5 New product page

Add item: one can add quantity to an existing product by providing product name and quantity and it would be updated in the inventory database

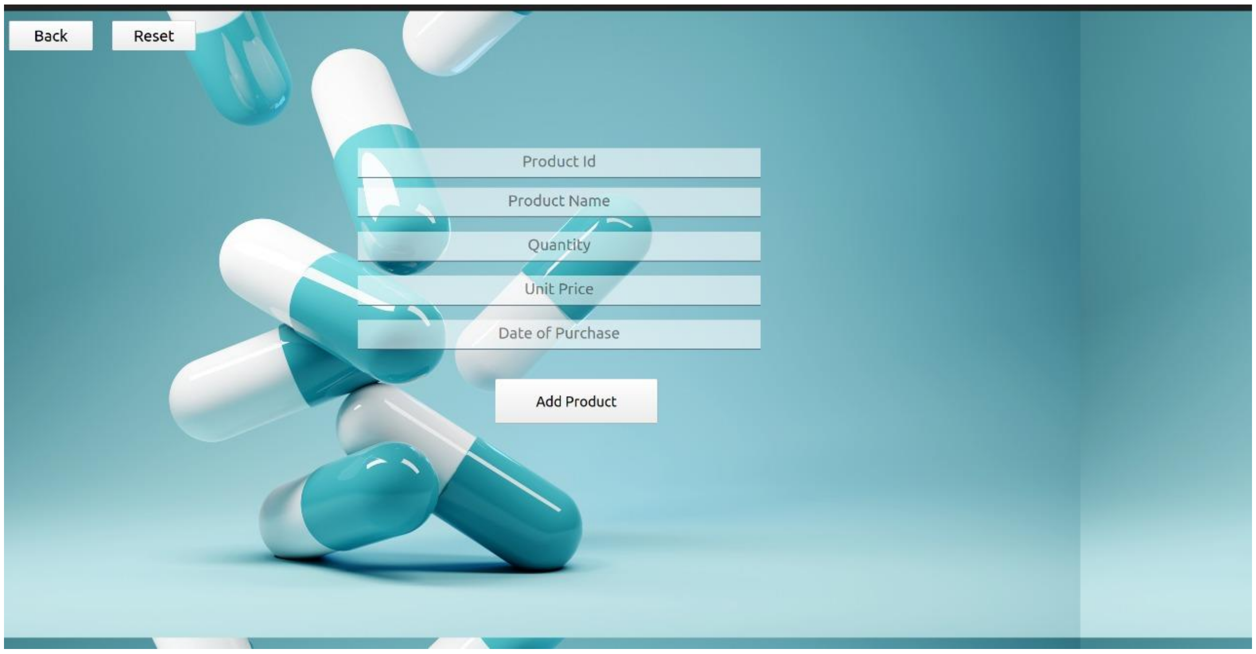


Figure 6 Add item page

Cosnume item : one can update the quantity of a product by providing product name and quantity

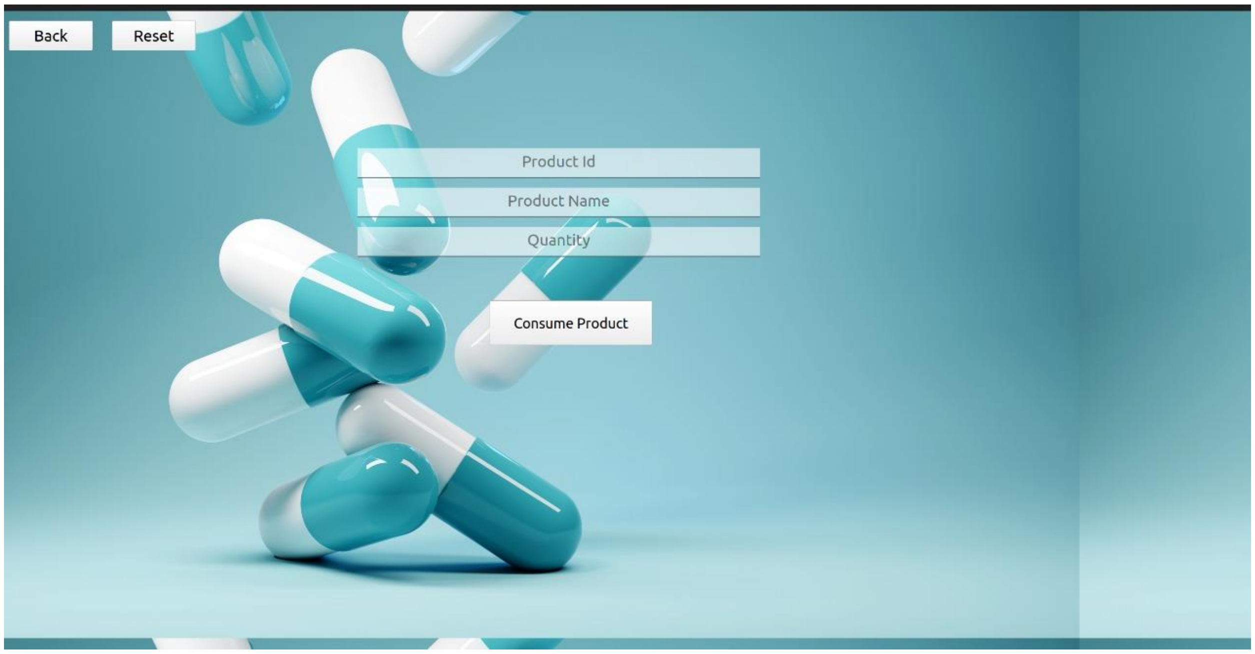


Figure 7 Consume product page

Remove product: product can be deleted by providing the product name.

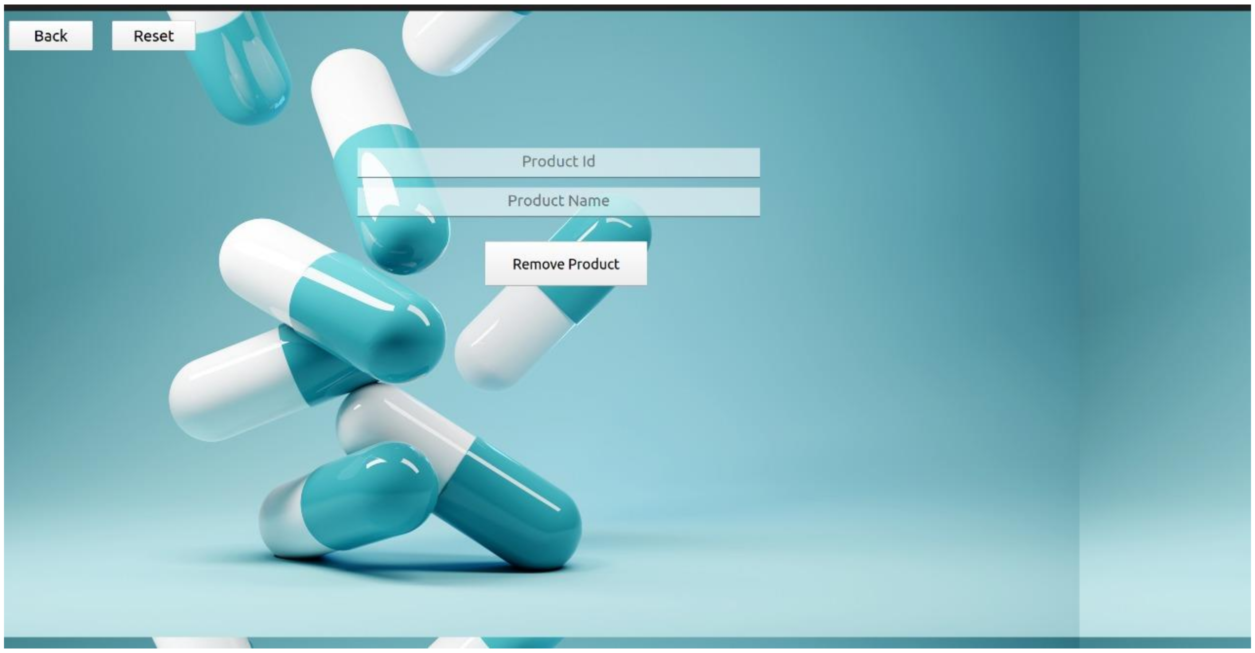


Figure 8 Remove product page

Project code for the main.py script:

```
#
# Created by: PyQt5 UI code generator 5.15.7
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.
import pandas as pd
from PyQt5 import QtCore, QtGui, QtWidgets
from loginUi import Ui_Form_login
from register import Ui_Form_register
from admin import Ui_Form_admin
from storeman import Ui_Form_storeman
from addnewproduct import Ui_Form_addnewprod
from addexisting import Ui_Form_addprod
from remove import Ui_Form_remove
from consume import Ui_Form_consume
import credentials
```

```
from database_backend import Medicine_inventory


class Ui_Form(object):

    def setupUi(self, Form):

        Form.setObjectName("Form")

        Form.resize(2550, 1300)

        self.widget = QtWidgets.QWidget(Form)

        self.widget.setGeometry(QtCore.QRect(0, 0, 2550, 1300))

        self.widget.setStyleSheet("QpushButton#pushButton{\n"

"background-color: qradialgradient(spread:pad, x1:0, y1:0.505682,x2:1,y2:0.477, stop:0 rgba(11, 131, 120, 219), stop:1 rgba(85, 98, 112, 226));\n"

"color:rgba(255,255,255,210);\n"

"border-radius:5px;\n"

"}\n"

"\n"

"QpushButton#pushButton:hover{\n"

"background-color: qradialgradient(spread:pad, x1:0, y1:0.505682,x2:1,y2:0.477, stop:0 rgba(150, 123, 111, 219), stop:1 rgba(85, 81, 84, 226));\n"

"\n"

"}\n"

"\n"

"QpushButton#pushButton:pressed{\n"

"padding-left:5px;\n"

"padding-top:5px;\n"

"background-color:rgba(150,123,111,255);\n"

"\n"

"}")

        self.widget.setObjectName("widget")

        self.label = QtWidgets.QLabel(self.widget)

        self.label.setGeometry(QtCore.QRect(0, 0, 1300, 1300))

        self.label.setStyleSheet("border-image: url(:/images/background.jpg);\n"

"border-top-left-radius: 50px;")

        self.label.setText("")

        self.label.setObjectName("label")

        self.label_2 = QtWidgets.QLabel(self.widget)

        self.label_2.setGeometry(QtCore.QRect(0, 0, 1300, 1300))

        self.label_2.setStyleSheet("background-color: rgba(0,0,0,80);\n"
"border-top-left-radius: 50px;")
        self.label_2.setText("")
        self.label_2.setObjectName("label_2")
        self.label_3 = QtWidgets.QLabel(self.widget)
        self.label_3.setGeometry(QtCore.QRect(1300, 0, 1200, 1300))
        font = QtGui.QFont()
        font.setPointSize(20)
        font.setBold(True)
        font.setWeight(75)
        self.label_3.setFont(font)
        self.label_3.setStyleSheet("background-color:rgba(255,255,255,255);\n"
"border-bottom-right-radius:50px;")
        self.label_3.setText("")
        self.label_3.setObjectName("label_3")
        self.label_4 = QtWidgets.QLabel(self.widget)
        self.label_4.setGeometry(QtCore.QRect(1680, 120, 261, 81))
        font = QtGui.QFont()
        font.setPointSize(20)
        font.setBold(True)
        font.setWeight(75)
        self.label_4.setFont(font)
        self.label_4.setStyleSheet("color:rgba(0,0,0,200);")
        self.label_4.setObjectName("label_4")
        self.AdminBtn = QtWidgets.QPushButton(self.widget)
        self.AdminBtn.setGeometry(QtCore.QRect(1660, 200, 321, 91))
        font = QtGui.QFont()
        font.setPointSize(18)
        font.setBold(True)
        font.setWeight(75)
        self.AdminBtn.setFont(font)
        self.AdminBtn.setStyleSheet("")
        self.AdminBtn.setObjectName("AdminBtn")
        self.label_6 = QtWidgets.QLabel(self.widget)
        self.label_6.setGeometry(QtCore.QRect(100, 160, 1000, 131))
        font = QtGui.QFont()
        font.setPointSize(24)
        font.setBold(True)
        font.setWeight(75)
        self.label_6.setFont(font)
        self.label_6.setStyleSheet("color:rgba(255,255,255,210);")
        self.label_6.setObjectName("label_6")
        self.label_7 = QtWidgets.QLabel(self.widget)
        self.label_7.setGeometry(QtCore.QRect(270, 250, 911, 101))
        font = QtGui.QFont()
        font.setPointSize(18)
        font.setBold(True)
        font.setWeight(75)
        self.label_7.setFont(font)
        self.label_7.setStyleSheet("color:rgba(255,255,255,170);")
        self.label_7.setObjectName("label_7")
        self.pushButton_2 = QtWidgets.QPushButton(self.widget)
        self.pushButton_2.setGeometry(QtCore.QRect(1660, 310, 321, 91))
        font = QtGui.QFont()
        font.setPointSize(18)
        font.setBold(True)
        font.setWeight(75)
        self.pushButton_2.setFont(font)
        self.pushButton_2.setStyleSheet("")
        self.pushButton_2.setObjectName("pushButton_2")
        self.label_5 = QtWidgets.QLabel(self.widget)
        self.label_5.setGeometry(QtCore.QRect(1670, 460, 261, 81))
        font = QtGui.QFont()
        font.setPointSize(14)
        font.setBold(False)
        font.setWeight(50)
        self.label_5.setFont(font)
        self.label_5.setStyleSheet("color:rgba(0,0,0,200);")
        self.label_5.setObjectName("label_5")
        self.RegisterBtn = QtWidgets.QPushButton(self.widget)
        self.RegisterBtn.setGeometry(QtCore.QRect(1660, 530, 321, 91))
        font = QtGui.QFont()
        font.setPointSize(18)
```

```
    font.setBold(True)
    font.setWeight(75)
    self.RegisterBtn.setFont(font)
    self.RegisterBtn.setStyleSheet("")
    self.RegisterBtn.setObjectName("RegisterBtn")
    self.retranslateUi(Form)
    QtCore.QMetaObject.connectSlotsByName(Form)
    self.RegisterBtn.clicked.connect(self.register)
    self.AdminBtn.clicked.connect(self.admin)
    self.pushButton_2.clicked.connect(self.storeman)


######### show register page ##############
def register(self):
    self.Formr = QtWidgets.QWidget()
    self.uir = Ui_Form_register()
    self.uir.setupUi_register(self.Formr)
    self.uir.RegisterBtn.clicked.connect(self.regpage)
    self.Formr.show()
def regpage(self):
    df = pd.read_csv("login_credentials.csv")
    password=self.uir.passwordEdit.text()
    user_name=self.uir.UsrNameEdit.text()
    re_entered_pass=self.uir.ReentrPasswordEdit.text()
    if password==re_entered_pass:
        credentials.add_new_user(df,user_name,password)
        self.Formr.close()

##### show login page for admin ######
def admin(self):
    self.FormAd = QtWidgets.QWidget()
    self.ui1 = Ui_Form_login()
    self.ui1.setupUi_login(self.FormAd)
    self.ui1.pushButton.clicked.connect( self.admin_page)
    self.FormAd.show()

########### show inventory after login###########
```

```
def admin_page(self, FormSt):
    self.username=self.ui1.lineEdit.text()
    self.passwrd=self.ui1.lineEdit_2.text()
    print(self.username)
    print(self.passwrd)
    if self.username=="admin" and self.passwrd=="admin@123":
        print("matched")
        self.FormAd.close()
        self.FormAdmin = QtWidgets.QWidget()
        self.uiAdmin = Ui_Form_admin()
        self.uiAdmin.setupUi_admin(self.FormAdmin)
        self.uiAdmin.addbtn.clicked.connect(lambda:self.addprod(self.uiAdmin))
        self.uiAdmin.addnewbtn.clicked.connect(self.addnewprod)
        self.uiAdmin.rembtn.clicked.connect(self.removeprod)
        self.uiAdmin.consumeBtn.clicked.connect(lambda:self.consumeprod(self.uiAdmin))
        self.uiAdmin.refrshbtn.clicked.connect(self.refresh)
        self.uiAdmin.pushButton_4.clicked.connect(lambda:self.exit(self.FormAdmin))
        self.load_table(self.uiAdmin)
        self.FormAdmin.show()
    else:

        print("passwor incorect")

######### show login page for storeman ########
def storeman(self):
    self.FormSt = QtWidgets.QWidget()
    self.uiSt = Ui_Form_login()
    self.uiSt.setupUi_login(self.FormSt)
    self.FormSt.show()
    self.uiSt.pushButton.clicked.connect(self.storeman_page)
    print("storeman")

############# show inventory  page after login ########
def storeman_page(self):
    df = pd.read_csv("login_credentials.csv")
    self.username=self.uiSt.lineEdit.text()
    self.passwrd=self.uiSt.lineEdit_2.text()

    flag=credentials.check_credentials(df,self.username,self.passwrd)

    if flag==True:

        self.FormSt.close()

        self.FormStman = QtWidgets.QWidget()

        self.uiStman = Ui_Form_storeman()

        self.uiStman.setupUi_storeman(self.FormStman)

        self.uiStman.addbtn.clicked.connect(lambda:self.addprod(self.uiStman))

        self.uiStman.consumeBtn.clicked.connect(lambda:self.consumeprod(self.uiStman))

        self.uiStman.pushButton_4.clicked.connect(lambda:self.exit(self.FormStman))

        self.load_table(self.uiStman)

        self.FormStman.show()


################ add, consume, remove, add new product  ########

def addprod(self,usropt):

    self.usropt=usropt

    self.Formadd = QtWidgets.QWidget()

    self.uiadd = Ui_Form_addprod()

    self.uiadd.setupUi_addprod(self.Formadd)

    self.Formadd.show()

    self.uiadd.addnewbtn.clicked.connect(self.addproduct)


def addproduct(self):

    df = pd.read_csv("medicine_inventory.csv")

    name=self.uiadd.name.text()

    quant=self.uiadd.unitprice_2.text()

    Medicine_inventory.update_product(self,df,name, "available quantity",quant)

    self.Formadd.close()

    self.load_table(self.usropt)


def removeprod(self):

    self.Formrem = QtWidgets.QWidget()

    self.uirem = Ui_Form_remove()

    self.uirem.setupUi_remove(self.Formrem)

    self.Formrem.show()

    self.uirem.addnewbtn.clicked.connect(self.removeproduct)
```

```
def removeproduct(self):
    name=self.uirem.name.text()
    df=pd.read_csv("medicine_inventory.csv")
    Medicine_inventory.remove_product(self,df,  name)
    self.Formrem.close()
    self.FormAdmin = QtWidgets.QWidget()
    self.uiAdmin = Ui_Form_admin()
    self.uiAdmin.setupUi_admin(self.FormAdmin)
    self.load_table(self.uiAdmin)
    self.FormAdmin.show()

def addnewprod(self):
    self.Formaddn = QtWidgets.QWidget()
    self.uiaddn = Ui_Form_addnewprod()
    self.uiaddn.setupUi_addnewprod(self.Formaddn)
    self.Formaddn.show()
    self.uiaddn.addnewbtn.clicked.connect(self.readnewproddata)
def readnewproddata(self):
    id=self.uiaddn.id.text()
    name=self.uiaddn.name.text()
    cat=self.uiaddn.catagory.text()
    unit=self.uiaddn.unitprice.text()
    totalprice=self.uiaddn.totalprice.text()
    purqnt=self.uiaddn.quantity.text()
    date=self.uiaddn.date.text()
    df=pd.read_csv("medicine_inventory.csv")
    Medicine_inventory.add_new_product(self,df, id, name, cat, purqnt, purqnt, unit,totalprice, date)
    self.Formaddn.close()
    self.load_table(self.uiAdmin)
    print(id,name,cat,unit,totalprice,purqnt,date)

def consumeprod(self,useropt):
    self.useropt=useropt
    self.Formcon = QtWidgets.QWidget()
    self.uicon = Ui_Form_consume()
    self.uicon.setupUi_consume(self.Formcon)
```

```
        self.Formcon.show()
        self.uicon.addnewbtn.clicked.connect(self.consumeproduct)
    def consumeproduct(self):
        name=self.uicon.name.text()
        quant=self.uicon.name_2.text()
        df=pd.read_csv("medicine_inventory.csv")
        Medicine_inventory.consume_product(self, df, name, quant)
        self.Formcon.close()
        self.load_table(self.useropt)

    ################# referesh exit ################
    def refresh(self):
        pass
    def exit(self,opt):
        self.opt=opt
        self.opt.close()


    ############ load valued in table from csv ##########
    def load_table(self,uiopt):
        self.uiopt=uiopt
        df= pd.read_csv("medicine_inventory.csv")
        for row in range(len(df.index)):
            data=Medicine_inventory.retrieve_rows(self,df,row)
            print(data["product id"])
            self.uiopt.tableWidget.setItem(row, 0, QtWidgets.QTableWidgetItem(str(data["product id"])))
            self.uiopt.tableWidget.setItem(row, 1, QtWidgets.QTableWidgetItem(data["product name"]))
            self.uiopt.tableWidget.setItem(row, 2, QtWidgets.QTableWidgetItem(data["category"]))
            self.uiopt.tableWidget.setItem(row, 3, QtWidgets.QTableWidgetItem(str(data["purchase quantity"])))
            self.uiopt.tableWidget.setItem(row, 4, QtWidgets.QTableWidgetItem(str(data["available quantity"])))
            self.uiopt.tableWidget.setItem(row, 5, QtWidgets.QTableWidgetItem(str(data["unit price (PKR)"])))
            self.uiopt.tableWidget.setItem(row, 6, QtWidgets.QTableWidgetItem(str(data["total price"])))
            self.uiopt.tableWidget.setItem(row, 7, QtWidgets.QTableWidgetItem(data["purchase date"]))

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.label_4.setText(_translate("Form", "Log In "))
        self.AdminBtn.setText(_translate("Form", "Admin"))
        self.label_6.setText(_translate("Form", "Pharmacy Management System"))
        self.label_7.setText(_translate("Form", " "))
        self.pushButton_2.setText(_translate("Form", "Storeman"))
        self.label_5.setText(_translate("Form", "New User"))
        self.RegisterBtn.setText(_translate("Form", "Register"))
import res
import sys
if __name__ == "__main__":

    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

Inventory backend

```
import pandas as pd

"""

ADMIN ACCESS
1. retrieve rows using rows index - input: row index, output: complete row
2. add new product in dataframe using inputs provided - input: product id, name, category, purchase quantity, available quantity, unit price, total price purchase date; output: write it in database
3. remove product given product name
4. update existing product data using provided inputs
5. consume product - input: product name, quantity consumed; output, update existing inventory of that product

STOREMAN ACCESS
4. update existing product data using provided inputs
5. consume product - input: product name, quantity consumed; output, update existing inventory of that product
```

```
"""

#df = pd.read_csv("medicine_inventory.csv")
# print(df)
#rows = len(df.index)

class Medicine_inventory():

    def __init__(self):
        pass

    def retrieve_rows(self, dataframe, row_id):
        if row_id < len(dataframe.index):
            return dataframe.iloc[row_id]
        else:
            return 'row doesnot exist'

    def add_new_product(self, dataframe, id, product_name, category, purchase_quantity, available_quantity, price,
                    total_price,date):
        if product_name not in dataframe['product name'].values:
            new_row = {'product id': id, 'product name': product_name, 'category': category,
                  'purchase quantity': purchase_quantity, 'available quantity': available_quantity,
                  'unit price (PKR)': price,'total price': total_price, 'purchase date': date}
            dataframe=dataframe.append(new_row, ignore_index=True)
            dataframe.to_csv("medicine_inventory.csv", index=False)
            return True
        else:
            return 'product already exists'

    def remove_product(self, dataframe, product_name):
        if product_name in dataframe['product name'].values:
            dataframe=dataframe[dataframe['product name'] != product_name]
            dataframe.to_csv("medicine_inventory.csv", index=False)
            return True
        else:
            return 'product not available'
    def update_product(self, dataframe, product_name, update_param, update_quantity):
        update_param="available quantity"
        if product_name in dataframe['product name'].values:
            val = dataframe.loc[(dataframe[dataframe['product name'] == product_name].index[0]), update_param]
            dataframe.loc[(dataframe[dataframe['product name'] == product_name].index[0]), update_param] = val + int(update_quantity)


            dataframe.to_csv("medicine_inventory.csv", index=False)
            return True
        else:
            return 'product not available'


    def consume_product(self, dataframe, product_name,  update_quantity):
        update_param="available quantity"
        if product_name in dataframe['product name'].values:
            val=dataframe.loc[(dataframe[dataframe['product name'] == product_name].index[0]), update_param]
            dataframe.loc[(dataframe[dataframe['product name'] == product_name].index[0]), update_param] =val- int(update_quantity)
            dataframe.to_csv("medicine_inventory.csv", index=False)
            return True
        else:
            return 'product not available'


'''
demo_class = Medicine_inventory()
print(demo_class.retrieve_rows(df, 1))
df = demo_class.add_new_product(df, 130, 'flgyl', 'capsule', 14, 12, 15, '01/12/2023')
print(demo_class.remove_product(df, 'asprin'))


# print(df[df['product name'] == 'brofin']['available quantity'])
# print(df)


df.to_csv("medicine_inventory.csv", index=False)
'''
```

User Credentials Backend

```
import pandas as pd
```

```
#print(df)

def add_new_user(dataframe,user_name,password):
    if user_name not in dataframe['user name'].values:
        new_row = {'user name':user_name, 'password':password}
        dataframe=dataframe.append(new_row, ignore_index=True)
        print(dataframe)
        dataframe.to_csv("login_credentials.csv", index=False)
        print('user added')
        return True
    else:
        print('user already exists')
        return False

def check_credentials(dataframe,user_name,password):
    if user_name in dataframe['user name'].values:
        flag=dataframe.loc[(dataframe[dataframe['user name'] == user_name].index[0]),'password']==password
        print("flag",flag)
        return flag
    else:
         print('User doesnot exist')
         return False

#df = pd.read_csv("login_credentials.csv")
#flag,df=add_new_user(df,"aq","123")

#df.to_csv("login_credentials.csv", index=False)

'''
print(check_credentials(df,'hari','haris123'))
df=add_new_user(df,'chemma','asim123')
print(df)
df.to_csv("login_credentials.csv",index=False)
'''
```