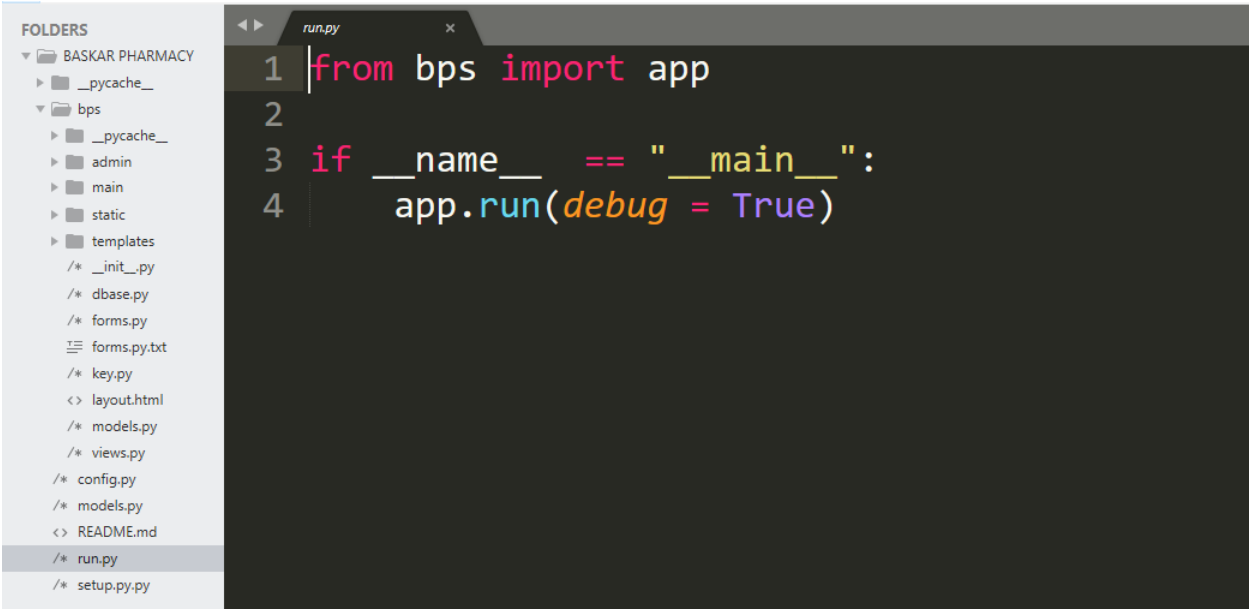
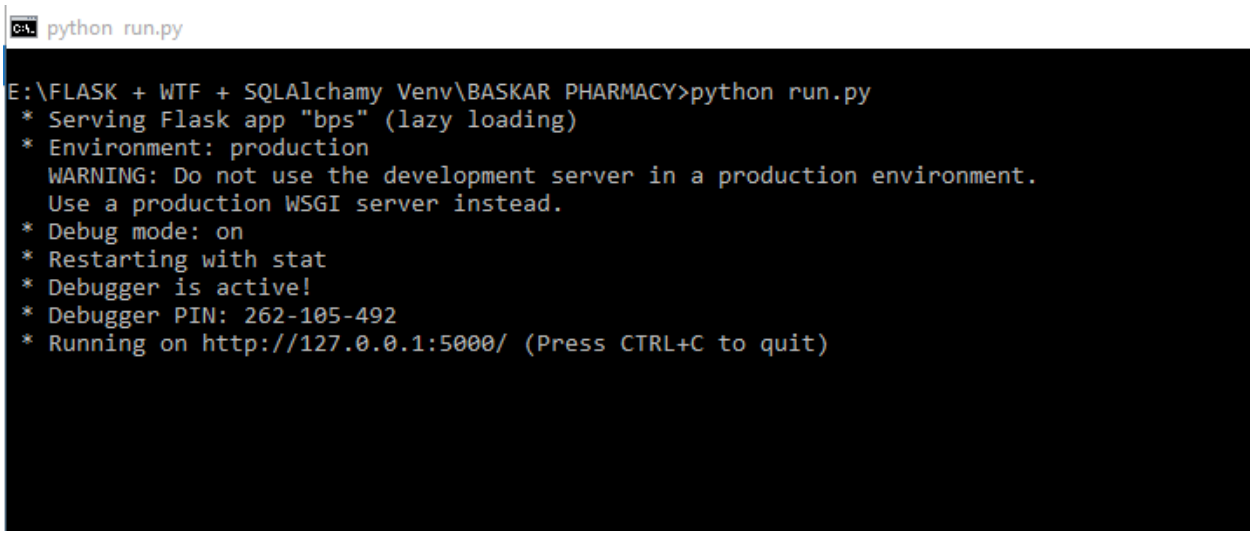


1.the left panel shows the tree structure of our project directory.

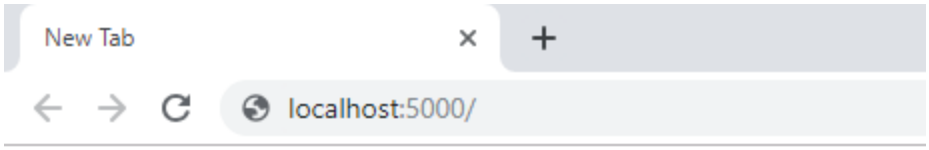
2.run.py file that is placed in the root directory act as the entry gateway to our web app .When the run.py file is run it starts the application server in the debug mode(only because the parameter debug=true is passed.This is removed when the app is finally deployed).



3.Lazy loading will be explained soon when we encounter it in database manipulationis .For now we can access the application using any browser by entering the following link as long as it is in production stage "localhost:5000/" or "127.0..0.1:5000".

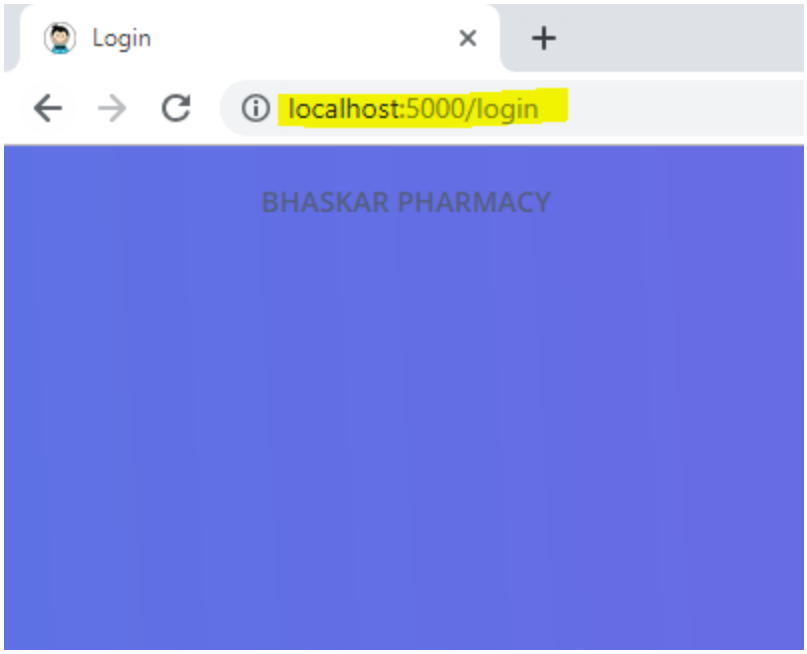
4."127.0.0.1" is the IP address that is used to send the request to any machine from itself.That is,when we use "127.0.0.1" as our IP address in the URL the request is sent to the servers that are running on the same machine and acting from the port number that is followed by the IP address which is "5000" in our case.

4.As said earlier the localhost in the above link is because of the fact that server is running the local machine in the production stage.When the application is deployed in to a commercial web hosting platform the link or URL might change accordingly.



5.Here, when we try to access the localhost the server checks whether the user is logged in. If yes he will be redirected to a page called dashboard where he can get information about stock , invoices ,transactions etc.

6.If the user is not logged in then he is asked to log-in in order to access other features of the application.That can be seen below.



7.Since the user is not logged in the server redirected the request that to the login page and the user gets to see a login form where he can enter the login details .

```python
import pymysql as db

host = "localhost"
user = "Project"
password = "password"
database = "pharmacy"
port = 3306

connection = db.connect(host,user,password,database,port)
cursor = connection.cursor()
```

8.The above code shows the connection to the database.In the first line we are importing a python package that act as a middle-ware between mysql server and web application called "pymysql".

9.Connection to the server can be established using the "connect" method of the pymysql package by passing the parameters such as the host,user,password,database and port.

```
# creating a table
SaleTransaction = "CREATE TABLE sale_transaction(SALE_ID VARCHAR(10)\
 PRIMARY KEY NOT NULL\
 ,SALE_DATE DATE);"
cursor.execute(SaleTransaction)
# commiting request to reflect the changes made
cursor.commit()

# inserting into table
insert = "INSERT INTO sale_transaction VALUES ("15306","30-01-2019");"
cursor.commit()

# querying the database
query = "SELECT * FROM sale_transaction;"
cursor.execute(query);
queryResults = cursor.fectall()
```

10.The above table shows three main actions performed on the database.
viz., (i)creation of a table
(ii)inserting into a table and
(iii)query from the table

11.The "commit()" method called on the cursor object makes sure the changes done on to the tables are actually reflected on the database.This feature is known as the lazy loading that I was talking during the first slide which postpones the actual changes to reflected on the database untill we commit the operations explicitly.

12.Now we are done with the database and would like to explain about the web-server .

13.Below slide shows us the packages that were imported and the blueprints that were registered to the main app.

14.Blueprints are the backend code that were encapsulated into logical blocks called "views" and are stored in different location from the root path in order to improve the maintenance efficiency or to decrease the complexity.

15.Since they are placed in different location ,they need to be registered with the main app using the function that is shown below called "app.register_blueprint()"

```python
from flask import Flask,Blueprint
from bps.admin.homepage import Home
from bps.admin.account import Login,Register
from bps.main.working import Dashboard
from config import Config

app = Flask(__name__)
app.config.from_object(Config)

app.register_blueprint(Home, url_prefix='/')
app.register_blueprint(Login, url_prefix='/')
app.register_blueprint(Register, url_prefix='/')
app.register_blueprint(Dashboard, url_prefix='/')
```

16.The actual work done by these blueprints are stored in their packages and shown below.

```python

Home = Blueprint('Home', __name__)

@Home.route('/')
def index():
    if 'user' in session:
        return redirect(url_for('Dashboard.dashboard'))
    else:
        return redirect(url_for('Login.login'))

```

17.Here when the index method is invoked ,it first checks whether the user is logged in using the condition given in line 10 which checks if the "session" array consists of any user.

18.Below slides show the work done by Blueprints "Login" and "Register".

```python
Login = Blueprint('Login', __name__)

# Login view
@Login.route('/login',methods = ['GET','POST'])
def login():
    if 'user' in session:
        session.pop('user',None)
    form = LoginForm()
    if request.method == 'POST' and form.validate_on_submit():
        email = request.form['email']
        password = request.form['password']
        cursor.execute("SELECT PASSWORD_HASH,FIRSTNAME FROM admin WHERE EMAIL = %s",email)
        resultRows = cursor.fetchall()

        if len(resultRows) == 1:
            passwordHash = resultRows[0][0]
            if check_password_hash(passwordHash, password) :
                session['user'] = resultRows[0][1]
                return render_template("dashboard.html" , title = "Dashboard",session = session)
            else :
                error = "no user found with such details"
        else:
            error = "no user found with such details"
        return render_template("login.html", title = "Login" , form = form , error = error)

    return render_template("login.html" , form = form , title = "Login" )

```

```python
Register = Blueprint('Register',__name__)

# Register view
@Register.route('/register',methods = ['GET','POST'])
def register():
    form = RegisterForm()
    if request.method == 'POST' and form.validate_on_submit():
        firstName = request.form['firstName']
        lastName = request.form['lastName']
        employId = request.form['employId']
        email = request.form['email']
        password = request.form['password']
        confirm = request.form['confirm']

        cursor.execute("SELECT FIRSTNAME FROM admin WHERE EMPLOY_ID = {}".format(employId))
        noOfUsers = cursor.fetchall()
        if len(noOfUsers) > 0:
            error="A user with this Roll Number/Email already exists."
            return redirect(url_for('register'))
        else :
            if password == confirm:
                passwordHash=generate_password_hash(password)
                cursor.execute("INSERT INTO admin VALUES (%s,%s,%s,%s,%s)",(employId,firstName,lastName,email,passwordHash))
                connection.commit()
                return redirect(url_for('Login.login'))
            else :
                error="Passwords do not match"
                return render_template("register.html",error=error)
            return redirect(url_for('Dashboard.dashboard'))
    return render_template("register.html",title = 'Register',form = form)

```