

**INDUSTRIAL INTERNSHIP TRAINING
AT NETWORKING ACADEMY**

On

DJANGO(FRAMEWORK)APPLICATION

Submitted by

HEMANT MISHRA

University Roll No

1809510031

In partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
MGM's College of Engineering & Technology, Noida
1, JANUARY 2021**

COURSE CERTIFICATE



OpenEDG Python Institute Authorized Academy Program | Program Your Future

Statement of Achievement

PCAP: Programming Essentials in Python

During the Cisco Networking Academy® course, administered by the undersigned instructor, the student has studied the following skills:

- the universal concepts of computer programming (i.e. variables, flow control, data structures, algorithms, conditional execution, loops, functions, etc.)
- developer tools, developer tools and the runtime environment;
- the syntax and semantics of the Python language;
- the fundamentals of object-oriented programming and the way they are adopted in Python;
- the means by which to resolve typical implementation problems;
- the writing of Python programs using standard language infrastructure;
- fundamental programming techniques, best practices, customs and vocabulary, including the most common library functions in Python 3.

This Statement of Achievement acknowledges that during the course PCAP: Programming Essentials in Python, the student has been able to accomplish coding tasks related to the basics of programming, and understands the programming techniques, customs and vocabulary used in the Python language.

By completing the course, the student is now ready to attempt the qualification PCAP – Certified Associate in Python Programming certification, from the OpenEDG Python Institute.

Hemant Mishra

Student

Indian Society for Technical Education

Academy Name

TABLE OF CONTENTS

Title Page

Certificate -----2

Acknowledgement-----4

Declaration Certificate-----5

Abstract-----6

1. Introduction -----7

2. Technical Section-----8

3. Conclusion-----21

4. References-----22

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my teacher NISHANT GUPTA as well as our principal SUNIL WAGH who gave me the golden opportunity to do this wonderful project on the topic DJANGO -APPLICATION(python -framework) , which also helped me in doing a lot of Research and i came to know about so many new things I am really thankful to them. Secondly I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame. Thanking you all for helping me complete this wonderful project.

DECLARATION

I affirm that the Industrial Internship Training report titled “INDUSTRIAL INTERNSHIP TRAINING AT NETWORKING ACADEMY” being submitted in partial fulfillment of the requirements for the award of the Degree of BACHELOR OF ENGINEERING IN DJANGO WEB FRAMEWORK is the original work carried out by me. It has not formed the part of any other project work submitted for award of any degree or diploma, either in this or any other Institution.

(Signature)

Name of the Student-HEMANT
University Roll No-1809510031

ABSTRACT

In this process , I learned the concepts of the web framework and dealing with responsive websites and making and dealing with dynamic web pages and learned (Mongodb+django) like framework , how backend technology deal with the user authentication and user login and user password and user data and login and signup system. Many instructors have already discovered the joy of teaching programming using the Python programming language. Now it's time to take Python to the next level. This workshop will introduce Django, an open source Python web framework that saves you time and makes web development fun. It's aimed at Computer Science instructors who want to teach how to build elegant web applications with minimal fuss. Django follows the Model-View-Template (MVT) architectural pattern. Its goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of DRY (Don't Repeat Yourself). Python is used throughout, even for settings, files, and data models. Topics that will be covered during the workshop include: setup and configuration, template language, and database integration through object-relational mapping.

INTRODUCTION

1.COMPANY BACKGROUND

Cisco Networking Academy, established in 1997, teaches students networking and other information technology-related skills, preparing them for jobs as well as for higher education in engineering, computer science and related fields. Since its launch, the program has grown to more than 9,000 Academies in 50 U.S. states and more than 165 countries with a curriculum taught in 16 different languages. More than 900,000 students participate in Academies operating in colleges and universities, technical schools, community-based organizations, and other educational programs around the world. Networking Academy blends face-to-face teaching with web-based curriculum, hands-on lab exercises, and Internet-based assessment.

With Cisco Networking Academy expanding into many different nations, some without the infrastructure present in western nations, Cisco has worked with a business partner to create a remote access router system as well as collaborating with over 200 academies worldwide to test and aid the development of the Packet Tracer application, which offers students and education centers a free networking education tool.

These are available in addition to the in-class practical labs for the Cisco courses.

2.TRAINING OBJECTIVE

My main objective was to learn about the (DJANGO)(python framework),MONGODB,NODEJS DJANGO is backend technology it is the amazing framework that provides some great already built features for handling backend .django is the large framework that has its default database(sqlite).handling a data in backend is very important how to make a server quick responsive and every time technology is going to change.a lot of developer likes to work with different backend technology .node js,django,php(laravel framework).I learned every and each concept of the backend technology django.

3.STUDENT WORK ASSIGNMENT

In this project .I came to know about various concept of backend technology django how to deal django authentication and models and forms and how to handle dynamic web pages.concept of ORM(object relational mapping) .and built a web pages for dealing with user data and information.save it into database.

TECHNICAL SECTION

1.DJANGO-ADMIN STARTPROJECT

writing your first app in django (python-framework) requires that the first thing is set up for django.

install django in system-(**pip install django**)

check version of the django-(**python -m django --version**)

starting of the project (**django-admin startproject <name the project>**)

starting the app inside the project(**python manage.py startapp <name the app>**)

2.Project-Explain

In this project we deal with the user data.and user login into the pages and update his details about medicine.

3.CODE AND EXPLANATION

In this project we deal with the various files-system and there is the explanation of the project file and code.

django-admin startproject medstore command creates various files automatically.

db.sqlite3 manage.py medstore

python manage.py startapp pharma command creating pharma name folder .

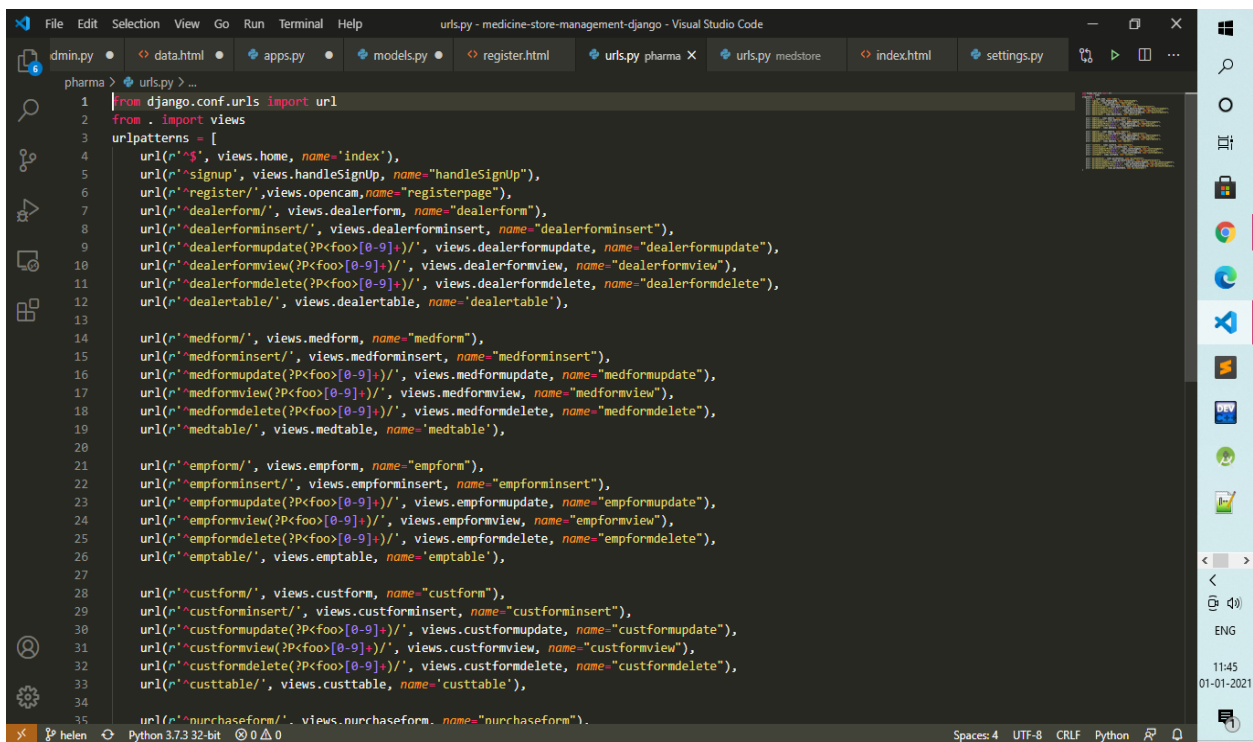
db.sqlite3 manage.py medstore pharma

in the directory name→ medstore/urls.py

```
from django.conf.urls import url, include
from django.contrib import admin
from pharma import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^pharma/', include('pharma.urls')),
    url(r'^$', views.home, name='index'),
    url(r'^login', views.handeLogin, name="handleLogin"),
    #url(r'^logout', views.handelLogout, name="handleLogout"),
    url(r'^signup', views.handleSignUp, name="handleSignUp"),
]
```

→ in the directory medstore/pharma/urls.py



```
1 from django.conf.urls import url
2 from . import views
3 urlpatterns = [
4     url(r'^$', views.home, name='index'),
5     url(r'^signup', views.handleSignUp, name="handleSignUp"),
6     url(r'^register/', views.opencam, name="registerpage"),
7     url(r'^dealerform/', views.dealerform, name="dealerform"),
8     url(r'^dealerforminsert/', views.dealerforminsert, name="dealerforminsert"),
9     url(r'^dealerformupdate(?P<foo>[0-9]+)/', views.dealerformupdate, name="dealerformupdate"),
10    url(r'^dealerformview(?P<foo>[0-9]+)/', views.dealerformview, name="dealerformview"),
11    url(r'^dealerformdelete(?P<foo>[0-9]+)/', views.dealerformdelete, name="dealerformdelete"),
12    url(r'^dealertable/', views.dealertable, name='dealertable'),
13
14    url(r'^medform/', views.medform, name="medform"),
15    url(r'^medforminsert/', views.medforminsert, name="medforminsert"),
16    url(r'^medformupdate(?P<foo>[0-9]+)/', views.medformupdate, name="medformupdate"),
17    url(r'^medformview(?P<foo>[0-9]+)/', views.medformview, name="medformview"),
18    url(r'^medformdelete(?P<foo>[0-9]+)/', views.medformdelete, name="medformdelete"),
19    url(r'^medtable/', views.medtable, name='medtable'),
20
21    url(r'^empform/', views.empform, name="empform"),
22    url(r'^empforminsert/', views.empforminsert, name="empforminsert"),
23    url(r'^empformupdate(?P<foo>[0-9]+)/', views.empformupdate, name="empformupdate"),
24    url(r'^empformview(?P<foo>[0-9]+)/', views.empformview, name="empformview"),
25    url(r'^empformdelete(?P<foo>[0-9]+)/', views.empformdelete, name="empformdelete"),
26    url(r'^empitable/', views.empitable, name='empitable'),
27
28    url(r'^custform/', views.custform, name="custform"),
29    url(r'^custforminsert/', views.custforminsert, name="custforminsert"),
30    url(r'^custformupdate(?P<foo>[0-9]+)/', views.custformupdate, name="custformupdate"),
31    url(r'^custformview(?P<foo>[0-9]+)/', views.custformview, name="custformview"),
32    url(r'^custformdelete(?P<foo>[0-9]+)/', views.custformdelete, name="custformdelete"),
33    url(r'^custtable/', views.custtable, name='custtable'),
34
35    url(r'^purchaseform/', views.purchaseform, name="purchaseform").
```

→ medstore/pharma/views.py

```
from .models import Dealer
from .models import Employee
from .models import Customer
from .models import Medicine
from .models import Purchase
from django.shortcuts import render
from django.contrib import messages
from django.db import IntegrityError
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
import cv2
import numpy as np
from django.contrib.auth import authenticate, login, logout

#import pytesseract extracting text from images and saving it into database
form.

#I am coder boy writing an application for crud data
def home(request):
    return render(request, 'pharma/register.html')

def opencam(request):
```

```

        return render(request, 'pharma/register.html')

def registerpage(request):
    form=UserCreationForm(request.POST)
    if form.is_valid():
        form.save()
    context={'form':form}
    return render(request, 'pharma/register.html')

def dealerform(request):
    dict = {'add': True, }
    return render(request, 'pharma/dealer.html', dict)

def dealerforminsert(request):
    try:
        dealer = Dealer()
        dealer.dname = request.POST['dname']
        dealer.address = request.POST['address']
        dealer.phn_no = request.POST['pno']
        dealer.email = request.POST['email']
        dealer.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')

def dealerformupdate(request, foo):
    try:
        dealer = Dealer.objects.get(pk=foo)
        dealer.dname = request.POST['dname']
        dealer.address = request.POST['address']
        dealer.phn_no = request.POST['pno']
        dealer.email = request.POST['email']
        dealer.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')

def handleSignUp(request):
    if request.method=="POST":
        # Get the post parameters
        username=request.POST['username']
        email=request.POST['email']
        fname=request.POST['fname']
        lname=request.POST['lname']
        pass1=request.POST['pass1']

```

```

pass2=request.POST['pass2']

# check for erroneous input
if (len(username)<10):
    #it is
    messages.error(request, " Your user name must be under 10
characters")
    return render(request,'pharma/register.html')

if not username.isalnum():
    messages.error(request, " User name should only contain
letters and numbers")
    return render(request,'pharma/register.html')
if (pass1!= pass2):
    messages.error(request, " Passwords do not match")
    return render(request,'pharma/register.html')

# Create the user
myuser = User.objects.create_user(username, email, pass1)
myuser.first_name= fname
myuser.last_name= lname
myuser.save()

#messages.success(request, " Your coder has been successfully
created")

return render(request, 'pharma/index.html')

def dealerformview(request, foo):
    dealer = Dealer.objects.get(pk=foo)
    dict = {'dealer': dealer}
    return render(request, 'pharma/dealer.html', dict)

def dealerformdelete(request, foo):
    dealer = Dealer.objects.get(pk=foo)
    dealer.delete()
    return render(request, 'pharma/index.html')

def dealertable(request):
    dealer = Dealer.objects.all()
    dict = {"dealer": dealer}
    return render(request, 'pharma/dealertable.html', dict)

def empform(request):
    dict = {'add': True}
    return render(request, 'pharma/emp.html', dict)

```

```

def empforminsert(request):
    try:
        emp = Employee()
        emp.e_id = request.POST['eid']
        emp.fname = request.POST['fname']
        emp.lname = request.POST['lname']
        emp.address = request.POST['address']
        emp.phn_no = request.POST['pno']
        emp.email = request.POST['email']
        emp.sal = request.POST['sal']
        emp.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')

#Empformda
def empformupdate(request, foo):
    try:
        emp = Employee.objects.get(pk=foo)
        emp.e_id = request.POST['eid']
        emp.fname = request.POST['fname']
        emp.lname = request.POST['lname']
        emp.address = request.POST['address']
        emp.phn_no = request.POST['pno']
        emp.email = request.POST['email']
        emp.sal = request.POST['sal']
        emp.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')

def empformview(request, foo):
    emp = Employee.objects.get(pk=foo)
    li=[]
    dict = {'emp': emp}
    return render(request, 'pharma/emp.html', dict)

def empformdelete(request, foo):
    emp = Employee.objects.get(pk=foo)
    emp.delete()
    return render(request, 'pharma/index.html')

```

```

def emptable(request):
    emp = Employee.objects.all()
    dict = {"emp": emp}
    return render(request, 'pharma/emptable.html', dict)

def custform(request):
    dict = {'add': True}
    return render(request, 'pharma/cust.html', dict)

def custforminsert(request):
    try:
        cust = Customer()
        cust.fname = request.POST['fname']
        cust.lname = request.POST['lname']
        cust.address = request.POST['address']
        cust.phn_no = request.POST['pno']
        cust.email = request.POST['email']
        cust.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')

def custformupdate(request, foo):
    try:
        cust = Customer.objects.get(pk=foo)
        cust.fname = request.POST['fname']
        cust.lname = request.POST['lname']
        cust.address = request.POST['address']
        cust.phn_no = request.POST['pno']
        cust.email = request.POST['email']
        cust.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')

def custformview(request, foo):
    cust = Customer.objects.get(pk=foo)
    dict = {'cust': cust}
    return render(request, 'pharma/cust.html', dict)

```

```
def custformdelete(request, foo):
    cust = Customer.objects.get(pk=foo)
    cust.delete()
    return render(request, 'pharma/index.html')

def custtable(request):
    cust = Customer.objects.all()
    dict = {"cust": cust}
    return render(request, 'pharma/custtable.html', dict)

def medform(request):
    dict = {'add': True}
    return render(request, 'pharma/med.html', dict)

def medforminsert(request):
    try:
        med = Medicine()
        med.m_id = request.POST['mid']
        med.mname = request.POST['mname']
        med.dname = request.POST['dname']
        med.desc = request.POST['desc']
        med.price = request.POST['price']
        med.stock = request.POST['stock']
        med.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')

def medformupdate(request, foo):
    try:
        med = Medicine.objects.get(pk=foo)
        med.m_id = request.POST['mid']
        med.mname = request.POST['mname']
        med.dname = request.POST['dname']
        med.desc = request.POST['desc']
        med.price = request.POST['price']
        med.stock = request.POST['stock']
        med.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')
```

```
def medformview(request, foo):
    med = Medicine.objects.get(pk=foo)
    dict = {'med': med}
    return render(request, 'pharma/med.html', dict)

def medformdelete(request, foo):
    med = Medicine.objects.get(pk=foo)
    med.delete()
    return render(request, 'pharma/index.html')

def medtable(request):
    med = Medicine.objects.all()
    dict = {"med": med}
    return render(request, 'pharma/medtable.html', dict)

def purchaseform(request):
    dict = {'add': True}
    return render(request, 'pharma/purchase.html', dict)

def purchaseforminsert(request):
    try:
        purchase = Purchase()
        purchase.pname = request.POST['pname']
        purchase.fname = request.POST['fname']
        purchase.lname = request.POST['lname']
        purchase.qty = request.POST['qty']
        purchase.phn_no = request.POST['pno']
        purchase.price = request.POST['price']
        a = (int(purchase.price)) * (int(purchase.qty))
        purchase.total = a
        purchase.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')

def purchaseformupdate(request, foo):
    try:
        purchase = Purchase.objects.get(pk=foo)
```

```

        purchase.pname = request.POST['pname']
        purchase.fname = request.POST['fname']
        purchase.lname = request.POST['lname']
        purchase.qty = request.POST['qty']
        purchase.phn_no = request.POST['pno']
        purchase.price = request.POST['price']
        a = (int(purchase.price)) * (int(purchase.qty))
        purchase.total = a
        purchase.save()
    except IntegrityError:
        return render(request, "pharma/new.html")
    return render(request, 'pharma/index.html')

def purchaseformview(request, foo):
    purchase = Purchase.objects.get(pk=foo)
    dict = {'purchase': purchase}
    return render(request, 'pharma/purchase.html', dict)

def purchaseformdelete(request, foo):
    purchase = Purchase.objects.get(pk=foo)
    purchase.delete()
    return render(request, 'pharma/index.html')

def purchasetable(request):
    purchase = Purchase.objects.all()
    dict = {"purchase": purchase}
    return render(request, 'pharma/purchasetable.html', dict)

def handleLogin(request):
    if request.method=="POST":
        # Get the post parameters
        loginusername=request.POST['loginusername']
        loginpassword=request.POST['loginpassword']

        user=authenticate(username= loginusername, password=
loginpassword)
        if user is not None:
            login(request, user)
            #messages.success(request, "Successfully Logged In")
            return render(request,'pharma/index.html')
        else:

```



```
        #messages.error(request, "Invalid credentials! Please try  
again")  
  
        return render(request, 'pharma/register.html')  
  
    return HttpResponseRedirect("404- Not found")
```

in this directory medstore/pharma/models.py

```
from django.db import models  
  
# Create your models here.  
class Dealer(models.Model):  
    dname = models.CharField(max_length=30)  
    address = models.CharField(max_length=100)  
    phn_no = models.BigIntegerField(unique=True)  
    email = models.CharField(max_length=50)  
  
    def __str__(self):  
        return self.email  
  
class Employee(models.Model):  
    e_id = models.IntegerField(unique=True)  
    fname = models.CharField(max_length=30)  
    lname = models.CharField(max_length=30)  
    address = models.CharField(max_length=100)  
    email = models.CharField(max_length=50)  
    sal = models.CharField(max_length=20)  
    phn_no = models.BigIntegerField(unique=True)  
  
    def __str__(self):  
        return self.email  
  
class Customer(models.Model):  
    fname = models.CharField(max_length=30)  
    lname = models.CharField(max_length=30)  
    address = models.CharField(max_length=100)  
    phn_no = models.BigIntegerField()  
    email = models.CharField(max_length=50)  
  
    def __str__(self):  
        return self.email
```

```
class Medicine(models.Model):
    m_id = models.IntegerField(unique=True)
    mname = models.CharField(max_length=30)
    dname = models.CharField(max_length=30)
    desc = models.CharField(max_length=100)
    price = models.CharField(max_length=30)
    stock = models.IntegerField()
```

```
def __str__(self):
    return self.mname
```

```
class Purchase(models.Model):
    pname = models.CharField(max_length=30)
    fname = models.CharField(max_length=30)
    lname = models.CharField(max_length=30)
    phn_no = models.BigIntegerField()
    price = models.BigIntegerField()
    qty = models.BigIntegerField()
    total = models.BigIntegerField()
```

```
def __str__(self):
    return self.pname
```

→ in the directory medstore//pharma/forms.py

```
from django import forms
from django.contrib.auth.models import User
```

```
class UserForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ['cust_id', 'fname', 'lname', 'phn_no', 'address',
'med_name', 'qty']
```

ou

OUTPUT SLIDE.


SLIDE-1

Apps Pinterest - India Lecture 21 - s.aktu... Videos Taking input in Pyth... Left Shift and Right... Home | Google Su... YouTube New Tab All d

HOME Dealer Management Depression Medicine Management Employee Management Customer Management Purchase Information

Medical Store Management

Overview:



Have any queries? Contact us via hemantrishra575@gmail.com or call or Whatsapp us on 851081769.

Projectworlds - Us

SLIDE-2

GUIDELIN Global Ne Cisco Net MINI PRO Writing y ProjectRe Online Di Medicine New Tab

127.0.0.1:8000/pharma/medform/

Apps Pinterest - India Lecture 21 - s.aktu... Videos Taking input in Pyth... Left Shift and Right... Home | Google Su... YouTube New Tab All d

HOME Dealer Management Medicine Management Employee Management Customer Management Purchase Information

Add Medicine Details

Medicine Code:

Medicine Name:

Dealer Name:

Price:

Stock:

Description:

Have any queries? Contact us via email or call or Whatsapp us on +91 6283056779.

Projectworlds - Contact Us

12:10 01-01-2021

SLIDE-3

GUIDELINGlobal NeCisco NetMINI PROdj Writing yProjectReOnline DHomeNew TabNew Tab

127.0.0.1:8000/signup

AppsPinterest - IndiaLecture 21 – saktu...VideosTaking input in Pyth...Left Shift and Right...Home | Google Su...YouTubeNew TabAlld

HOMEDealer ManagementDepressionMedicine ManagementEmployee ManagementCustomer ManagementPurchase Information

Medical Store Management

Overview:

HOMEDealer ManagementMedicine ManagementEmployee ManagementCustomer ManagementPurchase Information

Add Employee Details

Employee ID:

Employee Name:

First Name

Last Name

Address:

Address

Salary:

Salary

Phone Number:

Contact Number

Email:

Email

Cancel

Add Employee

random()class d

Have any queries? Contact us via hemanbmishra575@gmail.com or call or Whatsapp us on 8510081760.

Projectworlds - Contact Us

12-09 01-01-2021

CONCLUSION

My aim is to understand the concept of the django technology and understand the concept of the backend work and how to deal with the user data in the backend and save it into the database
My main objective is to be comfortable with the django and mongodb compass.

REFERENCES

<https://docs.djangoproject.com/en/3.1/>

<https://www.tutorialspoint.com/django/index.htm>

<https://realpython.com/tutorials/django/>

<https://realpython.com/linkedin-social-authentication-in-django/>