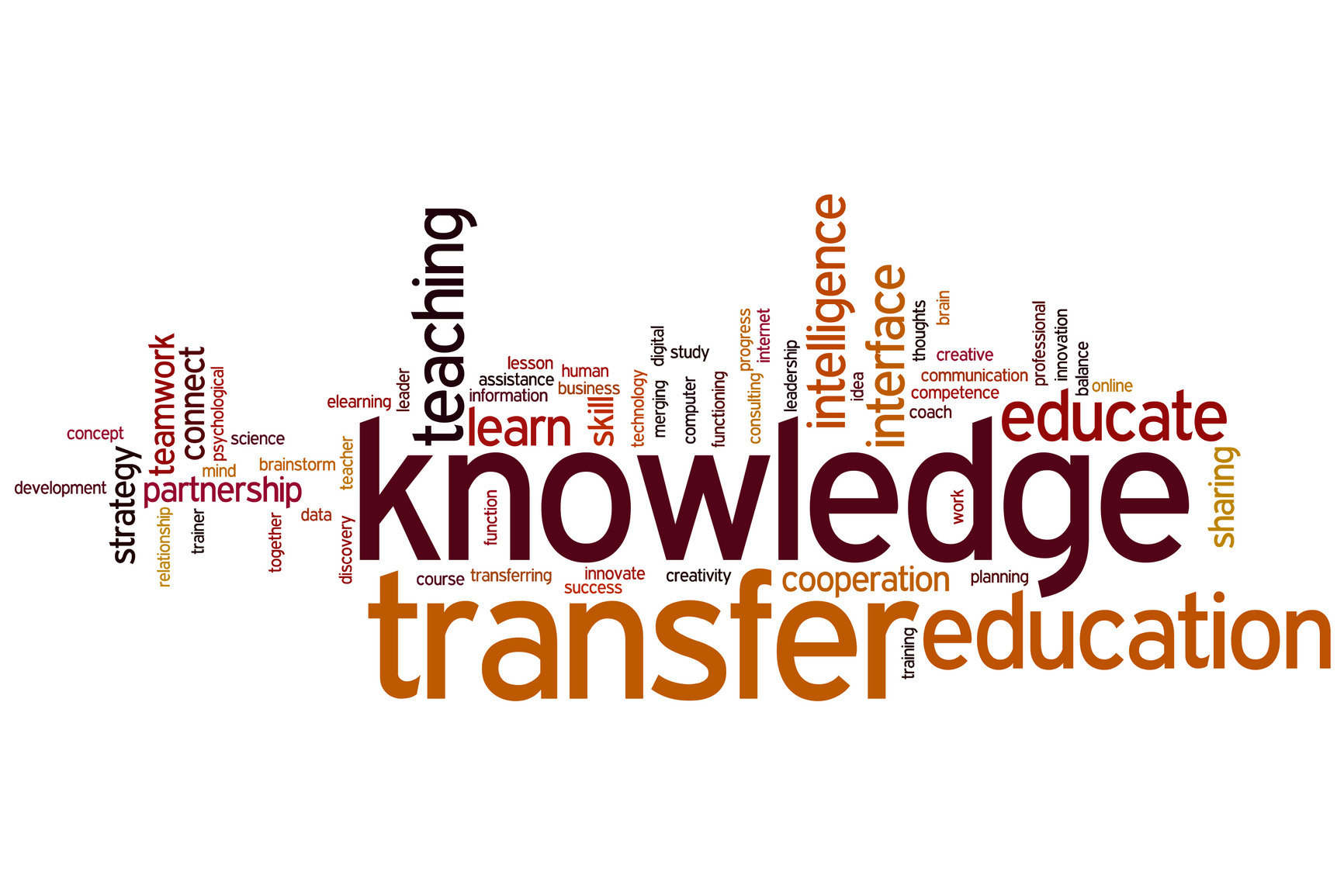
|  |
| --- |
| .000000000000000000 |

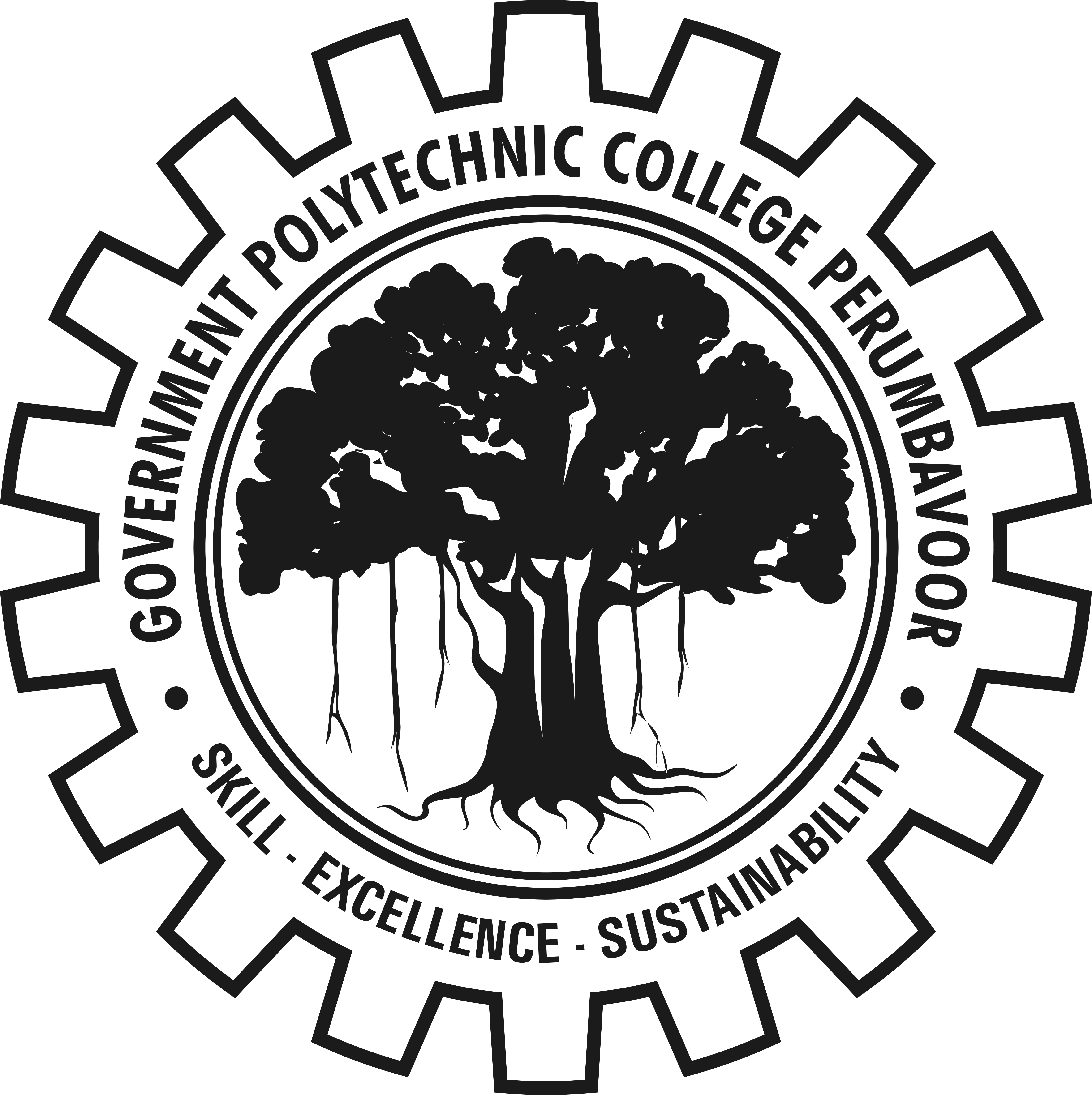
****

Application Development

|  |
| --- |
|  |
|  |



APPLICATION DEVELOPMENT



**Department of Computer Engineering**

**GOVERNMENT POLYTECHNIC COLLEGE, PERUMBAVOOR**

Koovappady P O, Perumbavoor, Ernakulam – 683544www.gptcperumbavoor.ac.in

**GOVERNMENT POLYTECHNIC COLLEGE**

**PERUMBAVOOR, KOOVAPPADY**

****

**LABORATORY RECORD**

Name

Year 2022-23 Branch Computer Engineering

Class No. Register No.

*Certified that this is the bonafide record of work done in the* **Application Development***Laboratory of the Government Polytechnic College, Perumbavoor,*

By Sri. /Kum. ……………………………………………………………………………………

Head of Section Lecturer in Charge

Date :

Place :

Internal Examiner External Examiner

**Index**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sl | Name of Experiment | | | | | | Date | | Remarks | |
|  | Vision and Mission statement | | | | | |  | |  | |
|  | PEO, PO and PSO | | | | | |  | |  | |
|  | Safety Procedures | | | | | |  | |  | |
| 1 | Program to find the largest number among the three input numbers | | | | | |  | |  | |
| 2 | Write a program to check whether givenstringispalindromeor not | | | | | |  | |  | |
| 3 | Write a program to demonstrate list , tuple,anddictionariesinPython | | | | | |  | |  | |
| 4 | Write a recursive function to find the factorial of an integer. | | | | | |  | |  | |
| 5 | Write a program to find area of room using class | | | | | |  | |  | |
| 6 | Demonstratethe installation of djangoframework | | | | | |  | |  | |
| 7 | Writeaprogramtoprinthelloworldin Djangoframework | | | | | |  | |  | |
| 8 | Createa simpleloginpagewithusername | | | | | |  | |  | |
| 9 | Writeapythonprogramtovalidate Name,E- mail,Passwordbycreatingaregistration form | | | | | |  | |  | |
| 10 | Writeapython program for creating sample models, managing them at admin level and adding data to model from user side | | | | | |  | |  | |
| 11 | Write a python program to checkconnectivitywithSQLdatabase | | | | | |  | |  | |
| 12 | Open ended project | | | | | |  | |  | |
|  |  | | | | | |  | |  | |
|  |  | | | | | |  | |  | |
|  |  | | | | | |  | |  | |
|  |  | | | | | |  | |  | |
| General Remarks: (For office use only ) | | | | | | | | | | |
| Test 1: | |  | Test 2: |  | Assign 1: |  | | Assign 2 | |  |
|  | | | | | | | | | | |

**VISION AND MISSION**

**Government Polytechnic College, Perumbavoor**

**Vision:** Excel as a centre of skill education moulding professionals who sincerely strive for the betterment of society

**Mission:**

* To impart state of the art knowledge and skill to the graduate and mould them to be competent, committed and responsible for the well-being of society.
* To apply technology in the traditional skills, thereby enhancing the living standards of the community.

**Department of Computer Engineering**

**Vision:** Excel as a centre of skill education in Computer Engineering, moulding professionals who sincerely strive for the betterment of themselves and the society.

**Mission:**

* To impart state of the art knowledge, skill and attitude to the graduates and contribute to their sustainable development.
* To merge technologies in the field of computer engineering with occupational skills, thereby improving the quality of living.

**PEO, PO and PSOs of the Program**

**Program Educational Outcome (PEOs)**

**PEO1:** To produce technically competent diploma holders in engineering with scientific,  
analytical, mathematical and problem solving skills.

**PEO2:** To develop the habit of quality, safety, selflearning along with environmental awareness.

**PEO3:**To equip diploma holders with good management practices, interpersonal skills and entrepreneurial discipline with strong adherence to ethics and values.

**Program Outcomes (POs)**

**PO1: Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science andengineering fundamentals and engineering specialization to solve the engineering problems.

**PO2: Problem analysis:** Identify and analyse well-defined engineering problems using codified standard methods.

**PO3: Design/ development of solutions:** Design solutions for well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

**PO4: Engineering Tools, Experimentation and Testing:** Apply modern engineering tools andappropriate technique to conduct standard tests and measurements.

**PO5: Engineering practices for society, sustainability and environment:** Apply appropriatetechnology in context of society, sustainability, environment and ethical practices.

**PO6: Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

**PO7: Life-long learning:** Ability to analyse individual needs and engage in updating in the context oftechnological changes.

**Program Specific Outcome (PSO):**

**PSO1: Specialization knowledge:** The computer engineering diploma graduate will be able to work in information technology industry in the area of development, implementation, testing and maintenance.

**PSO2: Professional growth:** The computer engineering diploma graduate will be fit for real time software projects.

**PSO3: Entrepreneurship:** A successful career as an entrepreneur with a passion, social commitment and ethical responsibility for real-world applications using optimal resources.

|  |  |  |  |
| --- | --- | --- | --- |
| **CO** | **Description** | **Duration(Hours)** | **CognitiveLevel** |
| CO1 | Develop simple programs in python programminglanguage | 9 | Applying |
| CO2 | Developasimpleweb applicationwithform | 10 | Applying |
| CO3 | Developawebapplicationwithmultipleforms | 12 | Applying |
| CO4 | Design and Develop a Web Application using front-endframework and MVC. | 12 | Applying |

**SYSTEM/SOFTWAREREQUIREMENTS**

* + AnyGNU/Linux
  + python3 and django

|  |
| --- |
| **SAFETY PROCEDURES** |

***Problem Statement:***

The safety instructions are presented to the attention of the students as a mean of preventing accidents while performing experiments and activities in the Data Structures lab of the department .The purpose is to draw attention to the risks involved in lab activities to prevent human suffering and damage to equipment.

### Safety in the laboratory:

Working in the lab is not allowed without following electricity precautions displayed.

No individual work is allowed in the lab.

Laboratory in charge is responsible for the arrangements of your lab activities; Listen carefully to his/her instructions and follow them.

### To do and not to do:

**Do’s**

Do wear ID card and follow dress code.

♣ Do log off the computers when you finish.

♣ Do ask for assistance if you need help.

♣ Do keep your voice low when speaking to others in the LAB.

♣ Do ask for assistance in downloading any software.

♣ Do make suggestions as to how we can improve the LAB.

♣ In case of any hardware related problem, ask LAB in charge for solution.

♣ If you are the last one leaving the LAB, make sure that the staff in charge of the LAB is informed to close the LAB.

♣ Be on time to LAB sessions.

♣ Do keep the LAB as clean as possible.

**Don’ts**

♣ Do not use mobile phone inside the lab.

♣ Don’t do anything that can make the LAB dirty (like eating, throwing waste papers etc).

♣ Do not carry any external devices without permission.

♣ Don’t move the chairs of the LAB.

♣ Don’t interchange any part of one computer with another.

♣ Don’t leave the computers of the LAB turned on while leaving the LAB.

♣ Do not install or download any software or modify or delete any system files on any lab computers.

♣ Do not damage, remove, or disconnect any labels, parts, cables, or equipment.

♣ Don’t attempt to bypass the computer security system.

♣ Do not read or modify other user’s file.

♣ If you leave the lab, do not leave your personal belongings unattended. We are not responsible for any theft.

### Electrical Safety:

Consult Electrical Engineering section available in the campus for electrical safety queries.

The lab equipment is powered from electrical sockets installed on the tables. Do not use equipment that is powered from a damaged socket.

Do not use equipment that is powered from flexible cable with damaged insulation or if it’s plug is not assembled properly.

Do not repair or disassemble electrical equipment including replacement of fuses installed in the equipment.

Do not open the main fuse box, unless it is an emergency and you need to switch off main circuit breaker.

### Emergency Switches:

The laboratory has circuit breakers, which is located in the main panel. Identify the place.  
In an emergency condition, switch off circuit breakers immediately.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 1 | Date: |  |  | - |  |  | - |  |  |

**AIM:**program to find the largest number among the three input numbers

**Program:**

# change the values of num1, num2 and num3

# for a different result

num1 = 10 num2 = 14 num3 = 12

# uncomment following lines to take three numbers from user

#num1 = float(input("Enter first number: "))

#num2 = float(input("Enter second number: "))

#num3 = float(input("Enter third number: "))

if (num1 >= num2) and (num1 >= num3):

largest = num1 elif (num2 >= num1) and (num2 >= num3):

largest = num2 else: largest = num3 print("The largest number between",num1,",",num2,"and",num3,"is",largest)

**Output :** The largest number between 10 , 14 and 12 is 14

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 2 | Date: |  |  | - |  |  | - |  |  |

**Aim:**Write a program to check whether givenstringispalindromeor not

**Program**

my\_str = 'aIbohPhoBiA'

# make it suitable for caseless comparison

my\_str = my\_str.casefold()

# reverse the string

rev\_str = reversed(my\_str)

# check if the string is equal to its reverse

if list(my\_str) == list(rev\_str):

print("The string is a palindrome.")

else:

print("The string is not a palindrome.")

**Output**

The string is not a palindrome

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 3 | Date: |  |  | - |  |  | - |  |  |

**Aim:** Python program for explaining use of list, tuple, set and dictionary

**Program**

# Lists

l = []

# Adding Element into list

l.append(5)

l.append(10)

print("Adding 5 and 10 in list", l)

# Popping Elements from list

l.pop()

print("Popped one element from list", l)

print()

# Set

s = set()

# Adding element into set

s.add(5)

s.add(10)

print("Adding 5 and 10 in set", s)

# Removing element from set

s.remove(5)

print("Removing 5 from set", s)

print()

# Tuple

t = tuple(l)

# Tuples are immutable

print("Tuple", t)

print()

# Dictionary

d = {}

# Adding the key value pair

d[5] = "Five"

d[10] = "Ten"

print("Dictionary", d)

# Removing key-value pair

del d[10]

print("Dictionary", d)

**Output**

Adding 5 and 10 in list [5, 10]

Popped one element from list [5]

Adding 5 and 10 in set {10, 5}

Removing 5 from set {10}

Tuple (5,)

Dictionary {5: 'Five', 10: 'Ten'}

Dictionary {5: 'Five'}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 4 | Date: |  |  | - |  |  | - |  |  |

**Aim:**write a recursive function to find factorial of a number

**Program**

def factorial(x):

if x == 1:

return 1

else:

return (x \* factorial(x-1))

num = 3

print("The factorial of", num, "is", factorial(num))

**output:**

the factorial of 3 is 6.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 5 | Date: |  |  | - |  |  | - |  |  |

**Aim:**Write a python program to find area of room using class.

**Program**

# create a class

class Room:

length = 0.0

breadth = 0.0

# method to calculate area

defcalculate\_area(self):

print("Area of Room =", self.length \* self.breadth)

# create object of Room class

study\_room = Room()

# assign values to all the attributes

study\_room.length = 42.5

study\_room.breadth = 30.8

# access method inside class

study\_room.calculate\_area()

**ouput:**

Area of room=1309.0

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 6 | Date: |  |  | - |  |  | - |  |  |

**Aim:**to understand the installation and prerequsites of django and python IDE

Django is a free and open-source web framework written in Python that adheres to the **model template view (MTV)** software architectural pattern. The MTV pattern is Django’s take on the [model–view–controller (MVC)](https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller) pattern. According to the Django Software Foundation, the model is the single definitive source of your data, the view describes the data that gets represented to the user via a Python callback [function](https://www.digitalocean.com/community/tutorials/how-to-define-functions-in-python-3) to a specific URL, and the template is how Django generates HTML dynamically.

## Prerequisites

## Step 1 — Install Python and pip

To install Python we must first update the local APT repository

$ sudo apt-get update && sudo apt-get -y upgrade

it is recommended by the Django Software Foundation to use Python 3,

sudo apt-get install python3

To verify the successful installation of Python 3, run a version check with the **python3** command:

python3 -V

install **pip** in order to install packages from PyPi, Python’s package repository.

$ sudo apt-get install -y python3-pip

$ pip3 -V to check pip version

To install virtualenv, we will use the **pip3** command, as shown below

pip3 install virtualenv

## Step 2 — Install Django

## **Install Django within a**virtualenv**.**

## Run the following command to create a directory called django-apps

## mkdirdjango-apps

## cddjango-apps

## $ pip install django

steps to create a new project

1.. create new project by using django-admin startprojectnewproject

2....go to the new created project by using cd newproject

3...create a new app inside the project using the command django-admin startappnewapp

4...open the project in vscode

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 7 | Date: |  |  | - |  |  | - |  |  |

**Aim:**Writeaprogramtoprinthelloworldin Djangoframework

step 1 After creating Project and app folder, go to setting.py and add line import os in import section

go to the section Installed Apps in settings.py and add our app name in single quotes.

For example see below ,

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'newapp', # App name

]

Step 2 create file urls.py inside the new app. this is for keeping the urls of the project

: add the following lines to urls.py (initially empty)

fromdjango . urls import path

from . import views

urlpatterns = [

path('',views.index,name='index'),

]

Here we have to import path and views from django first. Then inside urlpatterns , you have to give the path of

eachurls in the app. Remember each urls should have curresponding views in the views.py file.

Step 3: Go to the newapp and open views.py file. Create a view for the index page, ie, the first page. The views.py then will look like

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

def index(request):

returnHttpResponse ("<h1>Hello World </h1>")

Step 4: There is a file named urls.py inside the project folder.(in project folder, not in the app folder).

Open that file.There you can see the default code as

from django.contrib import admin

from django.urls import path

urlpatterns = [

path('admin/', admin.site.urls),

]

add import for include statement so that you can include all the urls in the newapp so that the final urls.py

file in the project folder will be as follows.

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path('admin/', admin.site.urls),

path('',include('newapp.urls'))

]

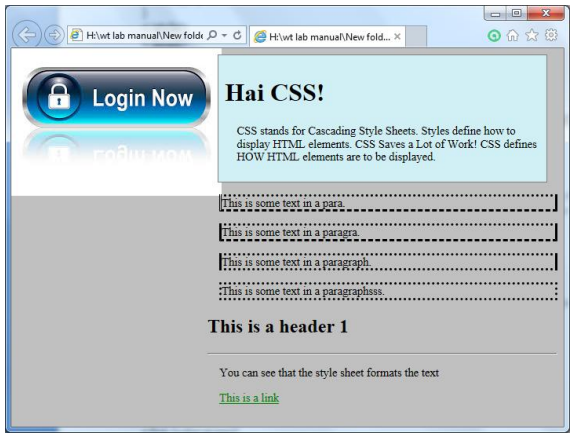
This says django to include whatever the urls in our newapp into the project.

Step 5: We are ready to go.

Run the following on terminal

python manage.py runserver(to run the server on 127.0.0.1:8000)

**Output:** Hello World



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 8 | Date: |  |  | - |  |  | - |  |  |

**Aim:**Createa simpleloginpagewithusername.

Step 1:

create folders templates and static inside the new app(static for storing images and template for keeping html files)

step 2:

Go to templates section of your app and create a new html file named index.html and write html code for creating a login form there (The same name we gave for urls,views)

Step 3: Go to the newapp and open views.py file. Create a view for the index page, ie, the first page. The views.py then will look like

fromdjango.shortcuts import render

# Create your views here.

def index(request):

return render(request,'index.html')

Step 4:

add the following lines to urls.py (initially empty)

fromdjango.urls import path

from .import views

urlpatterns = [

path('',views.index,name='index'),

where render is used to redirect to an html page named index.html, which we have to create in the template folder.

Step 5:

Run the following on terminal

python manage.py makemigrations ( This is to migrate everything we done, important when we do edits on model file)

python manage.py migrate

python manage.py runserver(to run the server on 127.0.0.1:8000)

**Output:** You can see whatever you done on index.html as output

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 9 | Date: |  |  | - |  |  | - |  |  |

**Aim:**Writeapythonprogramtovalidate Name,E- mail,Passwordbycreatingaregistration form

step 1: create a view for register inside views.py

def register(request):

return render(request,'register.html')

Our final views.py will look like

fromdjango.shortcuts import render

# Create your views here.

def index(request):

return render(request,'index.html')

def register(request):

return render(request,'register.html')

step 2:include the path in urls.py inside the newapp so that our final urls.py inside app will look like

fromdjango.urls import path

from .import views

urlpatterns = [

path('',views.index,name="index"),

path('register/',views.register,name="register"),

]

step 3:For basic registration, we have a basic model named "User" in django. You have to import the

user model. The user model is provided with fields username,email,password,first\_name,last\_name by default. For a basic

registration, use as it is.

3.a ) open forms.py . create a class named RegisterForm and add the following

fromdjango import forms

fromdjango.contrib.auth.models import User

from .import models

classRegisterForm(forms.ModelForm):

class Meta:

model=User

fields=['username','email','password','first\_name','last\_name']

widgets={

'password':forms.PasswordInput(),

'email':forms.EmailInput()

}

We are done with forms. Here model=User means we are created a User model (in built in django).

step 4: complete the view for registration in views.py.

You have to import the new form , model we done using the below statement

"from .forms import \*

from .models import \*

fromdjango.contrib import messages"

Here messages is to print a confirmation message that registration is done or not.

Our final views.py file will be as below

fromdjango.shortcuts import render,redirect

from .forms import \*

from .models import \*

fromdjango.contrib import messages

# Create your views here.

def index(request):

return render(request,'index.html')

def register(request):

form=RegisterForm()

dict={'form':form}

ifrequest.method=="POST":

form=RegisterForm(request.POST)

ifform.is\_valid():

user=form.save()

messages.success(request,"succeessfull")

return render(request,'register.html',dict)

Step 5:Create register.html inside templates folder and add the following code.

<html>

{% load widget\_tweaks %}

<body>

<form method="POST">

{% csrf\_token %}

Username

{% render\_fieldform.username class="form-control" %}<br>

Email

{% render\_fieldform.email class="form-control" %}<br>

Password

{% render\_fieldform.password class="form-control" %}<br>

Firstname

Last Name

{% render\_fieldform.last\_name class="form-control" %}<br>

<button type="submit">Register</button>

</form>

{%if messages%}

<ul> {%for message in messages%}

<li {% if message.tags %}

{% endif %}>

{{message}}

</li>

{% endfor %}

</ul>

{% endif %}

</body>

</html>

Step 6: create a super user by using the command python manage.py createsuperuser and assign a username and password.

step 7: make necessary migrations and run the project. Go to admin section and view the registered data.

NB: You may install widget\_tweaks if its not installed using pip install django-widget\_tweaks and add it to setting.py---

Installed App section if its your first time. So the Installed App section will look like

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

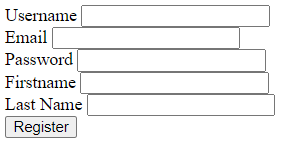
'django.contrib.staticfiles',

'widget\_tweaks',

'newapp',

]

**Output:** Registration form created.



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 10 | Date: |  |  | - |  |  | - |  |  |

**Aim:**file for creating sample models, managing them at admin level and adding data to model from user side

create new model and insert new data into model

step 1:create a new model in models.py

for example,

class posts(models.Model):

title=models.CharField(max\_length=20)

content=models.CharField(max\_length=40)

will create a new model named posts.

step 2: register that model in admin.py

so that superuser can view and manage it in the admin panel

admin.site.register(feedbacks)

step 3: try login into admin and add some dummy data into it.

Now to create a form to insert data into post model

step 4: create an html file addpost.html and add html content accordingly. for example for the above model,

<html>

<body>

<h1>Create a Post </h1>

<form action="{% url 'addpost' %}" method="POST">

{% csrf\_token %}

Title: <input type="text" name="title"/><br/>

Content: <br/>

<textarea cols="35" rows="8" name="content">

</textarea><br/>

<input type="submit" value="Post"/>

</form>

</body>

</html>

step 5: create function for addpost in views.py

defaddpost(request):

return render(request,'addpost.html')

step 6: create url for addpost accordingly in urls.py file. the urls.py will look like

fromdjango.urls import path

from .import views

urlpatterns = [

path('',views.index,name="index"),

path('register/',views.register,name="register"),

path('login/',views.log,name="login"),

path('dashboard/',views.dashboard,name="dashboard"),

path('addpost/',views.addpost,name="addpost"),

]

step 7:inside views.py, import the model name using "from .models import \*"

step 8:save the model as done in register page (refer register section) ,

defaddpost(request):

ifrequest.method == 'POST':

post=posts()

post.title= request.POST['title']

post.content= request.POST['content']

post.save()

return render(request, 'addpost.html')

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 11 | Date: | D | D | - | M | M | - | Y | Y |

**Aim:**Write a program to checkconnectivitywithSQLdatabase

**views.py file**

fromdjango.shortcuts import render,redirect

from .forms import \*

from .models import \*

fromdjango.contrib import messages

fromdjango.contrib.auth import authenticate, login

fromdjango.conf import settings

fromdjango.core.mail import send\_mail

# Create your views here.

def index(request):

return render(request,'index.html')

def dashboard(request):

return render(request,'dashboard.html')

defaddpost(request):

ifrequest.method == 'POST':

post=posts()

post.title= request.POST['title']

post.content= request.POST['content']

post.save()

return render(request, 'addpost.html')

else:

return render(request,'addpost.html')

defviewall(request):

cr=posts.objects.all().order\_by('-title')

#select \* from posts order by title

return render(request,'listall.html',{'cr':cr})

defviewone(request):

ifrequest.method=='POST':

t=request.POST['title']

cr=posts.objects.filter(title=t)

return render(request,'listall.html',{'cr':cr})

else:

return render(request,'viewone.html')

defdeleteone(request):

ifrequest.method=='POST':

t=request.POST['title']

cr=posts.objects.filter(title=t)

cr.delete()

#return render(request,'listall.html')

return render(request,'deleteone.html')

else:

return render(request,'deleteone.html')

def register(request):

form=RegisterForm()

dict={'form':form}

ifrequest.method=="POST":

form=RegisterForm(request.POST)

ifform.is\_valid():

user=form.save()

user.set\_password(user.password)

subject = 'welcome to FDP'

message = 'Hi , thank you for registering in FDP.'

email\_from = 'nefsal003@gmail.com'

recipient\_list = [user.email, ]

send\_mail( subject, message, email\_from, recipient\_list )

user.save()

messages.success(request,"succeessfull")

#return redirect('login')

return render(request,'register.html',dict)

def log(request):

ifrequest.method == 'POST':

# Process the request if posted data are available

username = request.POST['username']

password = request.POST['password']

# Check username and password combination if correct

user = authenticate(username=username, password=password)

if user is not None:

# Save session as cookie to login the user

login(request, user)

# Success, now let's login the user.

return render(request, 'dashboard.html')

else:

# Incorrect credentials, let's throw an error to the screen.

return render(request, 'login.html', {'error\_message': 'Incorrect username and / or password.'})

else:

# No post data availabe, let's just show the page to the user.

return render(request, 'login.html')

**urls.py in the app** # create corresponding html file in the templates folder

romdjango.urls import path

from .import views

urlpatterns = [

path('',views.index,name="index"),

path('register/',views.register,name="register"),

path('login/',views.log,name="login"),

path('dashboard/',views.dashboard,name="dashboard"),

path('addpost/',views.addpost,name="addpost"),

path('listall/',views.viewall,name="viewall"),

path('viewone/',views.viewone,name="viewone"),

path('deleteone/',views.deleteone,name="deleteone")

]

**models.py**

fromdjango.db import models

# Create your models here.

class posts(models.Model):

title=models.CharField(max\_length=20)

content=models.CharField(max\_length=40)

class feedbacks(models.Model):

heading=models.CharField(max\_length=20)

content=models.CharField(max\_length=40)

**forms.py**

fromdjango import forms

fromdjango.contrib.auth.models import User

from .import models

classRegisterForm(forms.ModelForm):

class Meta:

model=User

fields=['username','email','password','first\_name','last\_name',]

widgets={

'password':forms.PasswordInput(),

'email':forms.EmailInput()

}

**app.py**

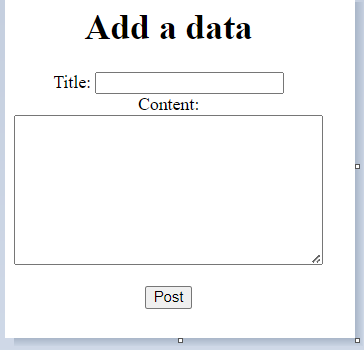
fromdjango.apps import AppConfig

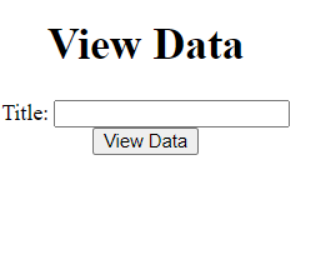
class Newapp2Config(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'newapp2'

**output**:





|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Exp No. | 12 | Date: | D | D | - | M | M | - | Y | Y |

### Open-ended Experiments

**Aim:**Open experiment experiment using python django.

* **views.py file**

from django.shortcuts import render,redirect

from django.contrib.auth.models import User

from django.contrib.auth.forms import UserCreationForm,AuthenticationForm

from django.db import IntegrityError

from django.contrib import messages

from django.contrib.auth import login,logout,authenticate

#Method for create a new User

def create\_user(request):

context={

"form":UserCreationForm

}

if request.method == "GET" :

return render(request, "signup.html",context)

else:

password=request.POST.get("password1")

password\_2=request.POST.get("password2")

username=request.POST.get("username")

if password==password\_2:

try:

user=User.objects.create\_user(

username= username,

password= password,

)

user.save()

login(request,user)

return render(request, "home.html")

except IntegrityError:

messages.error(request,"The User is registered")

return redirect("signup")

else:

messages.error(request,"Passwords must be the same")

return redirect("signup")

#LogIn Method

def signin(request):

context={

"form":AuthenticationForm

}

if request.method == 'GET':

return render(request, "login.html", context)

else:

user = authenticate(request, username=request.POST['username'], password=request.POST['password'])

if user is None:

messages.error(request, "Username or password is incorrect")

return redirect('login')

login(request,user)

return render(request, "home.html")

#close session method

def close\_session(request):

logout(request)

context={

"form":AuthenticationForm

}

return render(request, "login.html", context)

def home(request):

return render(request, "home.html")

* **urls.py in the app**

from django.contrib import admin

from django.urls import path,include

urlpatterns = [

path('admin/', admin.site.urls),

path("",include("login\_app.urls"))

]

* **models.py**

from django.db import models

from django.contrib.auth.models import User

class Login(models.Model):

title = models.CharField(max\_length=200)

description = models.TextField()

created\_at = models.DateTimeField(auto\_now\_add=True)

datecompleted = models.DateTimeField(null=True, blank=True)

importtant = models.BooleanField(default=False)

user = models.ForeignKey(User, on\_delete=models.CASCADE)

def \_\_str\_\_(self) -> str:

return self.title + " - " + self.user.username

class Meta:

db\_table = "login"

* **apps.py**

from django.apps import AppConfig

class LoginAppConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'login\_app'

* **forms.py**

fromdjango import forms

fromdjango.contrib.auth.models import User

from .import models

classRegisterForm(forms.ModelForm):

class Meta:

model=User

fields=['username','email','password','first\_name','last\_name',]

widgets={

'password':forms.PasswordInput(),

'email':forms.EmailInput()

}

* **urls.py**

from django.urls import path

from . import views

urlpatterns = [

path("", views.signin, name="Login"),

path("home/", views.home, name="Home"),

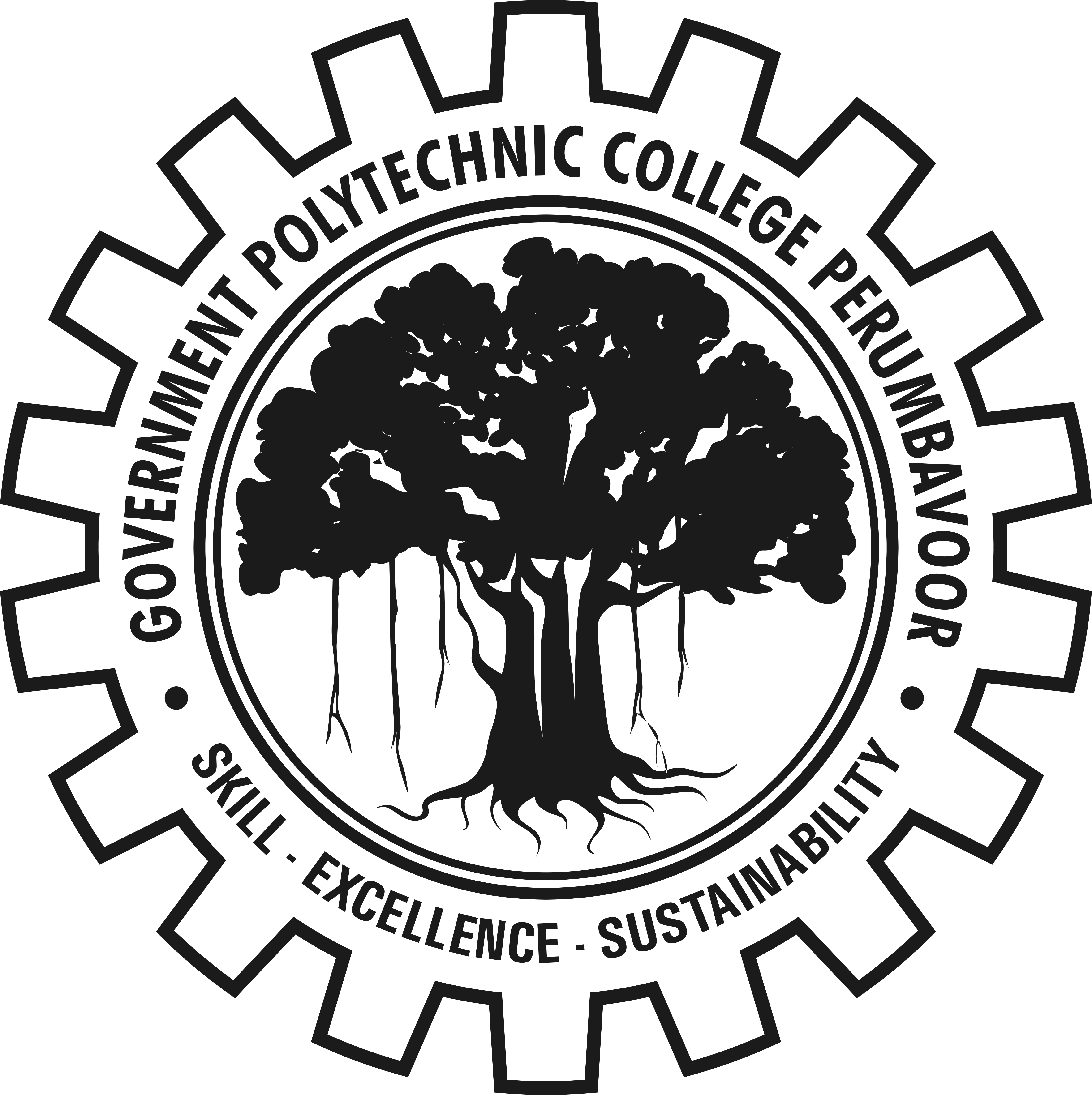
path("signup/", views.create\_user, name="signup"),

path("logout/", views.close\_session, name="logout"),

path("login/", views.signin, name="login")

]

**Output**:



**GOVERNMENT POLYTECHNIC COLLEGE, PERUMBAVOOR**

Koovappady P O, Perumbavoor, Ernakulam – 683544

www.gptcperumbavoor.ac.in