Project Report

—

Harshit Chauhan
XII Non-Medical

Dayanand Public School, Shimla

CERTIFICATE

This is to certify that **Harshit Chauhan** of class **XII Non-Medical** of **Dayanand Public School, Shimla** has completed his project file under my supervision. He has taken proper care and utmost sincerity in completion of the project.

I certify that this project is up to my expectations and as per guidelines issued by CBSE.

Ms. Varsha Sud

(IP teacher, Dayanand Public School)

ACKNOWLEDGEMENT

In the accomplishment of this project website **“Paper”** I would like to thank all those who have helped me in completing this project.

Firstly, I would like to thank my IP teacher Ms. Varsha Sud for her able guidance and constant support. She encouraged us at each step and acted at a critique who pin-pointed us our mistakes.

I would also extend my gratitude to my friends and family who supported me time and time again and helped me in every way possible.

Harshit Chauhan

XII Non-Medical

PREFACE

In the current time of pandamic schools are closed causing studies to be hindered . As a result online learning has become an important part of the life of students and teachers. Online learning can be termed as a tool that can make the teaching–learning process more student-centered, more innovative, and even more flexible.

Online learning is defined as “learning experiences in synchronous or asynchronous environments using different devices (e.g., mobile phones, laptops, etc.) with internet access.

So in this project we are going to try to use our knowledge of python and its libraries to create a website which facilitates online learning.

Introduction

This project aims at studying the computer language Python in an advanced and useful manner by making use of different python libraries like Django, Matplotlib, etc in order to make a functional website.

With that in mind the main goal of the project is to facilitate online learning by allowing teachers to conduct online tests for the classrooms of students.

It allows teachers to create classrooms and easily create tests which the students in the classroom can take. Teachers can get the report of the student response while the students can get their test result which they can analyse in order to improve themselves.

Index

Python

It is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

Characteristics of Python

Following are important characteristics of Python Programming −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Django

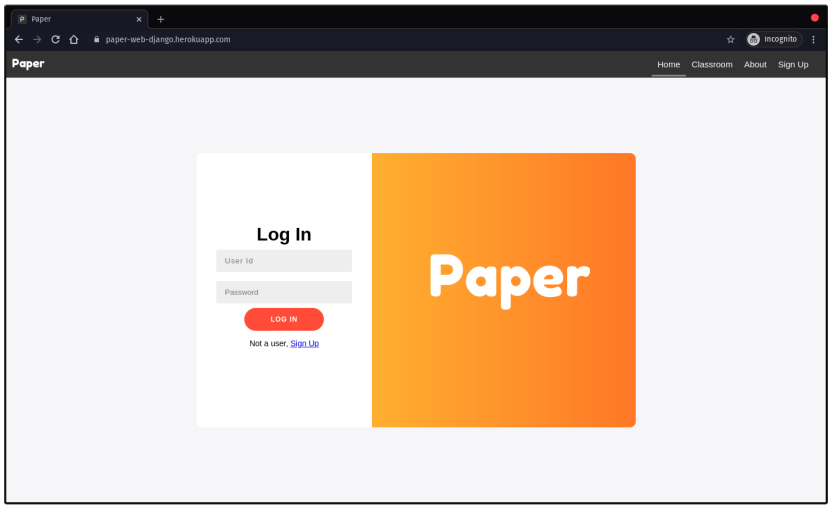
It is a Python Web framework that encourages rapid development and clean pragmatic design. A Web framework is a set of components that provide a standard way to develop websites fast and easily. Django's primary goal is to ease the creation of complex database-driven websites. Some well known sites that use Django include PBS, Instagram, Disqus, Washington Times, Bitbucket and Mozilla.

Matplotlib

It is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

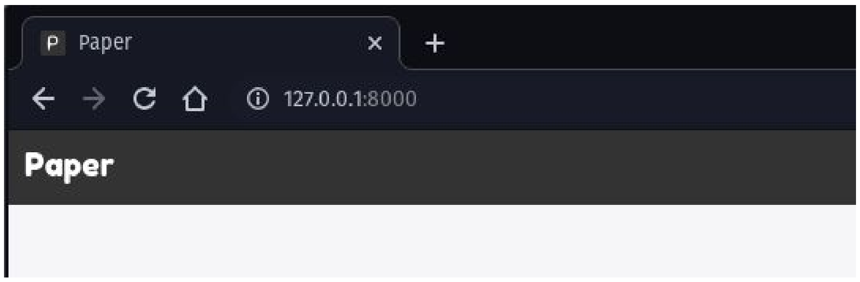
Setting Up

Live version - https://paper-web-django.herokuapp.com/

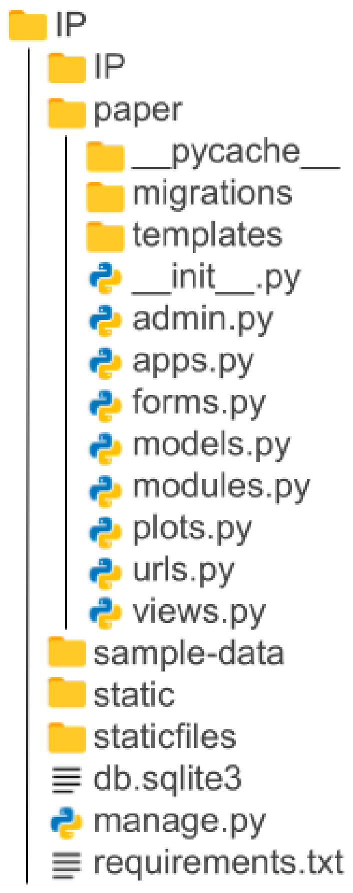


To run the project on your local machine follow the steps-

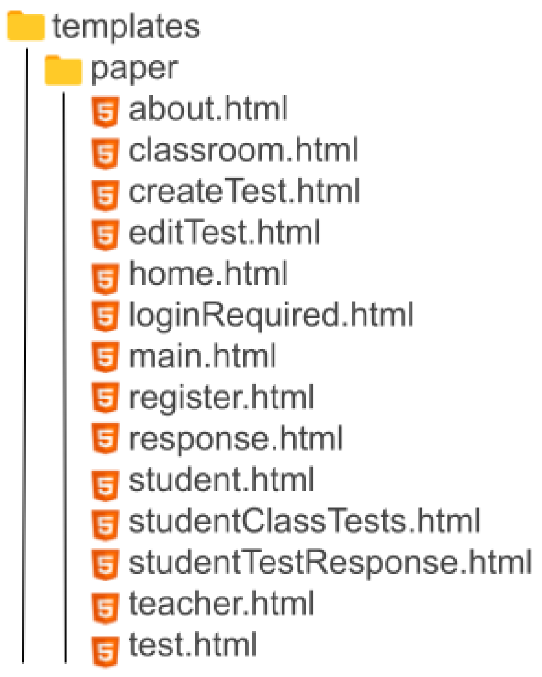
1. Open Command prompt and change directory to the one containing the project.
2. Run `pip install -r requirements.txt to install` the required libraries.
3. Run `python manage.py collectstatic` to collect static files.
4. Run `python manage.py runserver` to deploy it to your localhost.
5. Open a browser(preferably Chrome) and type `127.0.0.1:8000` in the address bar.



Project Structure



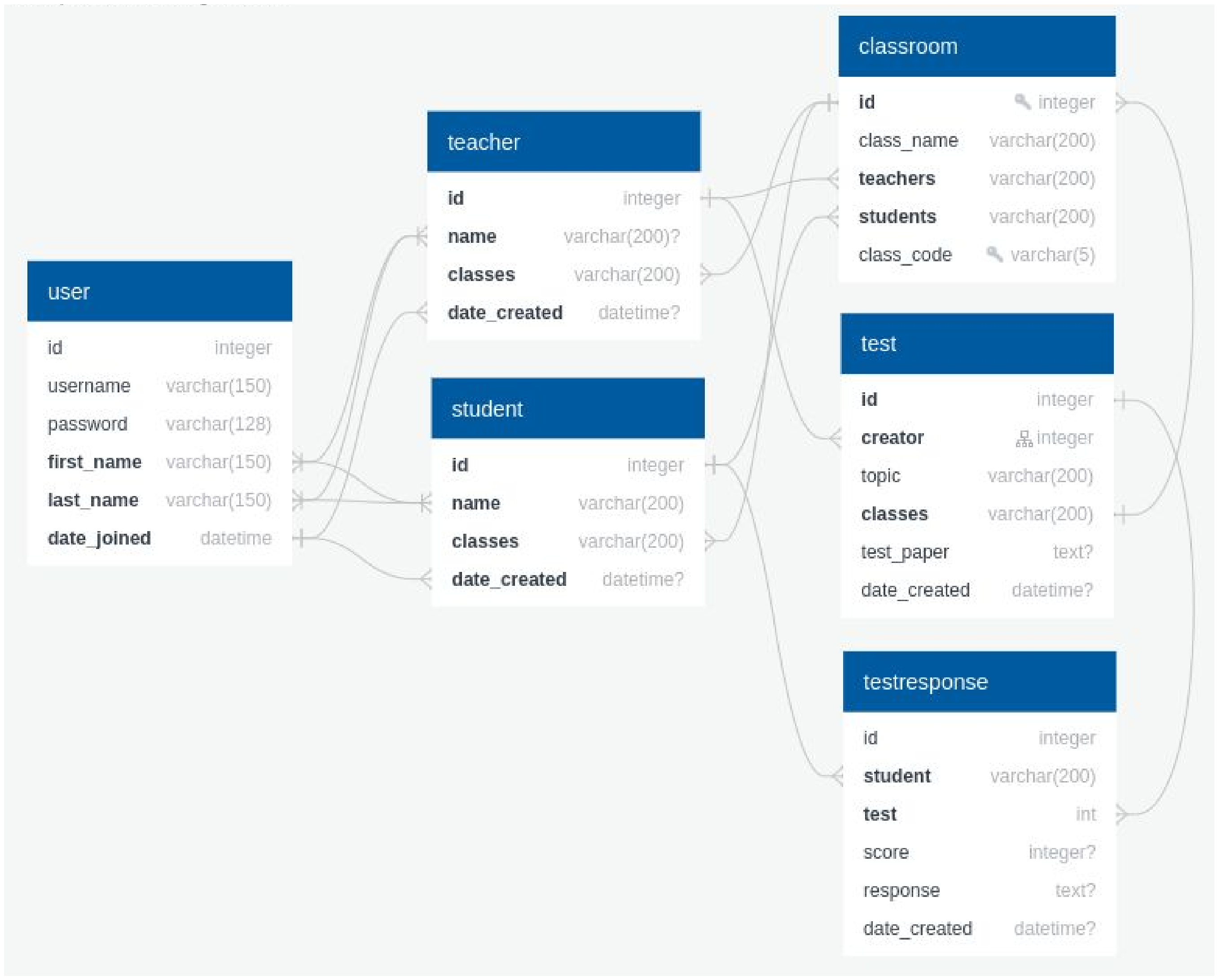
```
IP
  IP
  paper
    __pycache__
    migrations
    templates
    __init__.py
    admin.py
    apps.py
    forms.py
    models.py
    modules.py
    plots.py
    urls.py
    views.py
  sample-data
  static
  staticfiles
  db.sqlite3
  manage.py
  requirements.txt
```



```
templates
  paper
    about.html
    classroom.html
    createTest.html
    editTest.html
    home.html
    loginRequired.html
    main.html
    register.html
    response.html
    student.html
    studentClassTests.html
    studentTestResponse.html
    teacher.html
    test.html
```

➔ ***IP> IP*** *- Default files of django app like settings.py, wsgi.py, etc.*
➔ ***migrations*** *- Changes made in django model(database).*
➔ ***templates*** *- HTML templates of website pages.*
➔ ***static/staticfiles*** *- Static files like HTML, images, etc.*

Database Structure



user	
id	integer
username	varchar(150)
password	varchar(128)
first_name	varchar(150)
last_name	varchar(150)
date_joined	datetime

teacher	
id	integer
name	varchar(200)?
classes	varchar(200)
date_created	datetime?

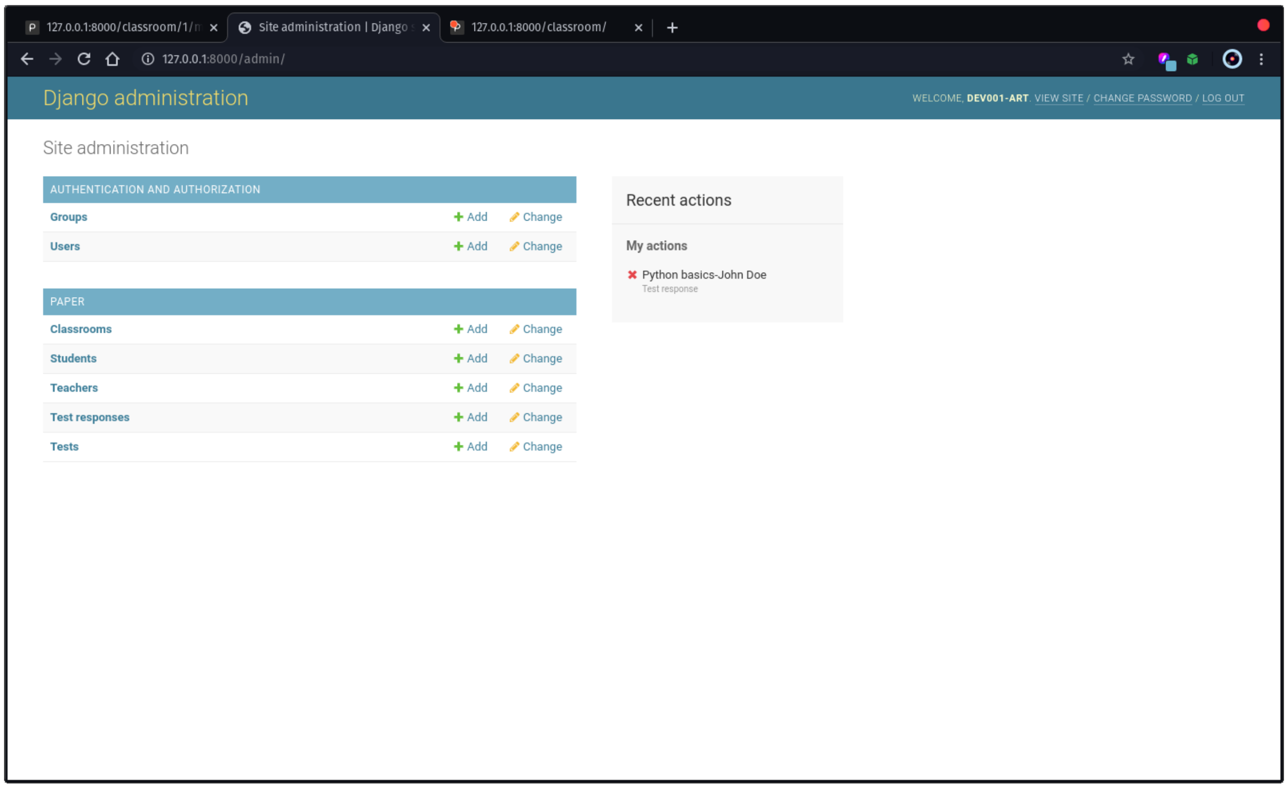
student	
id	integer
name	varchar(200)
classes	varchar(200)
date_created	datetime?

classroom	
id	integer
class_name	varchar(200)
teachers	varchar(200)
students	varchar(200)
class_code	varchar(5)

test	
id	integer
creator	integer
topic	varchar(200)
classes	varchar(200)
test_paper	text?
date_created	datetime?

testresponse	
id	integer
student	varchar(200)
test	int
score	integer?
response	text?
date_created	datetime?

Note- *User table is auto-generated by django with other tables for migrations,etc.*


127.0.0.1:8000/classroom/1/
Site administration | Django
127.0.0.1:8000/classroom/
127.0.0.1:8000/admin/

Django administration

WELCOME, **DEV001-ART**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

PAPER		
Classrooms	+ Add	Change
Students	+ Add	Change
Teachers	+ Add	Change
Test responses	+ Add	Change
Tests	+ Add	Change

Recent actions

My actions

✖ Python basics-John Doe
Test response

Models can be created or altered from the file models.py.

Database can also be viewed or changed by the administrator from the admin panel by logging into it. Admin can also change the permissions, etc. of users.

For admin panel credentials are given below-

Username: admin

Password: admin

URLS

This file helps the website to know what to do when a certain url is requested.

```
from django.urls import path, re_path
from . import views
from .modules import switch_view

urlpatterns = [
   path('', views.home, name='home'),
   path('signup/', views.home, name='signup'),
   path('login/', views.home, name='login'),
   path('logout/', views.logoutUser, name='logout'),
   path('classroom/',
switch_view(views.student,views.teacher,views.loginRequired)),
   path('classroom/<str:pk_classroom>/',
switch_view(views.student,views.teacher,views.loginRequired)),
   path('classroom/<str:pk_classroom>/leave/', views.leaveClassroom),
   path('classroom/<str:pk_classroom>/create/', views.createTest),
   path('classroom/<str:pk_classroom>/edit/<str:pk_test>/', views.editTest),
   path('classroom/test/<str:pk_test>/response', views.studentTestRespone),
   path('classroom/<str:pk_classroom>/response/<str:pk_test>/',
views.response),
   path('classroom/<str:pk_classroom>/response/<str:pk_test>/download',
views.export),
   path('classroom/<str:pk_classroom>/members',views.classroom),
   path('classroom/test/<str:pk_test>/',views.test),
   path('classroom/test/<str:pk_test>/view/',views.viewTest),
   path('about/',views.about),
]
```

In Django, views are Python functions which take requests as parameters and return an HTTP response(Webpage). Each view needs to be mapped to a corresponding URL pattern. This is done via urls.py.

```
path('login/', views.home, name='login')
```

Here if user enters the address **127.0.0.1/login/** or **domainname/login/ ,** home function from views.py will be triggered which in response return html page with content of home page.

In case of **/classroom/** , switch function from modules.py is used which triggers different views depending on the user(student, teacher or unauthorised).

Models

```
from django.db import models

class Classroom(models.Model):
    class_name = models.CharField(max_length=200)
    teachers = models.ManyToManyField('Teacher', blank=True)
    students = models.ManyToManyField('Student', blank=True)
    class_code = models.CharField(max_length=5)
    def __str__(self):
        return str(self.class_code) + '-' + str(self.class_name)
    class Meta:
        unique_together = (("id","class_code"),)

class Teacher(models.Model):
    name = models.CharField(max_length=200, nul
l=True)
    classes = models.ManyToManyField(Classroom, blank=True)
    date_created = models.DateTimeField(auto_now_add=True, null=True)
    def __str__(self):
        return self.name

class Student(models.Model):
    name = models.CharField(max_length=200)
    classes = models.ManyToManyField(Classroom, blank=True)
    date_created = models.DateTimeField(auto_now_add=True, null=True)
    def __str__(self):
        return self.name

class Test(models.Model):
    topic = models.CharField(max_length=200)
    classes = models.ManyToManyField(Classroom, blank=True)
    creator = models.ForeignKey(Teacher, on_delete=models.CASCADE)
    test_paper = models.TextField(null=True)
#complete paper stored as JSON string
    date_created = models.DateTimeField(auto_now_add=True, null=True)
    def __str__(self):
        return str(self.topic)

class TestResponse(models.Model):
    student = models.ForeignKey(Student, related_name='Student', null=True,
on_delete=models.CASCADE)
    test = models.ForeignKey(Test, related_name='Test', null=True,
on_delete=models.CASCADE)
    score = models.IntegerField(null=True)
    response = models.TextField(null=True)   #answers stored as JSON string
    date_created = models.DateTimeField(auto_now_add=True, null=True)
    def __str__(self):
        return str(self.test) + '-' + str(self.student)
    class Meta:
        unique_together = (("student_id", "test_id"),)
```

models.py is used to create models which maps to the corresponding table in the connected database. In this case default sqlite is used as database. The conversion of queries and table structure from models to database is handled by django itself. All we have to do is make changes or queries to the corresponding model.

These models translated to the database structure shown previously.

Modules

```
import random

def switch_view(first_view, second_view, third_view):
   def inner_view(request, *args, **kwargs):
       try:
           if request.user.username[0] == '2':
               return first_view(request, *args, **kwargs)
           elif request.user.username[0] == '1':
               return second_view(request, *args, **kwargs)
       except:
           return third_view(request, *args, **kwargs)
   return inner_view

def generateClassCode(classCodeList):
   code = ''
   for i in range(0,5):
       code += chr(random.randint(65, 90))
   if code in classCodeList:
       generateClassCode()
   elif code != '':
       return code
```

- Switch function is used to call different views based on the user. It return first_view if the user is a student(user id start with 2), second view if teacher(user id start with 1) or third view if the user has not logged in.

- GenerateClassCode function generates a 5 upper case character string which serves as the class code for the classrooms.

Views

views.py contains views. A view function, or view for short, is a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, etc.

Imports

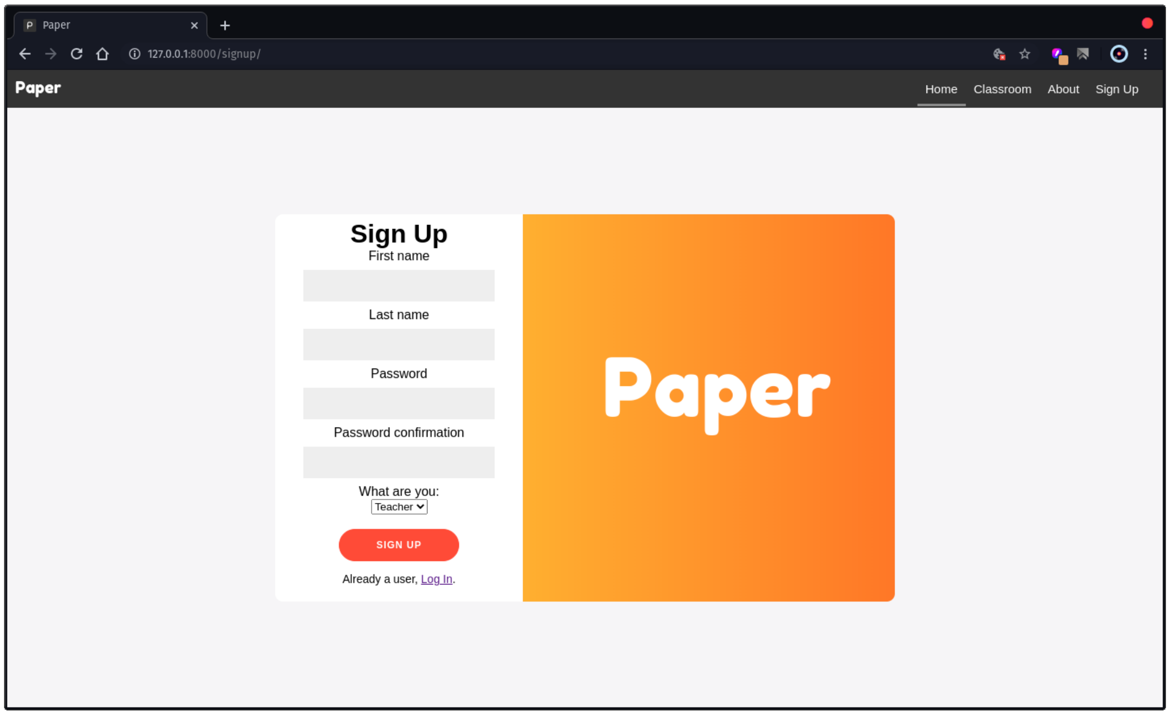
```python
import csv
import json
import random

from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from django.contrib.auth import login, logout, authenticate
from django.core.files.storage import FileSystemStorage
from django.http import HttpResponse, HttpResponseRedirect

from .models import *
from .forms import TestForm, TestResponseForm, ClassroomForm, CreateUserForm
from .modules import *
```

Home

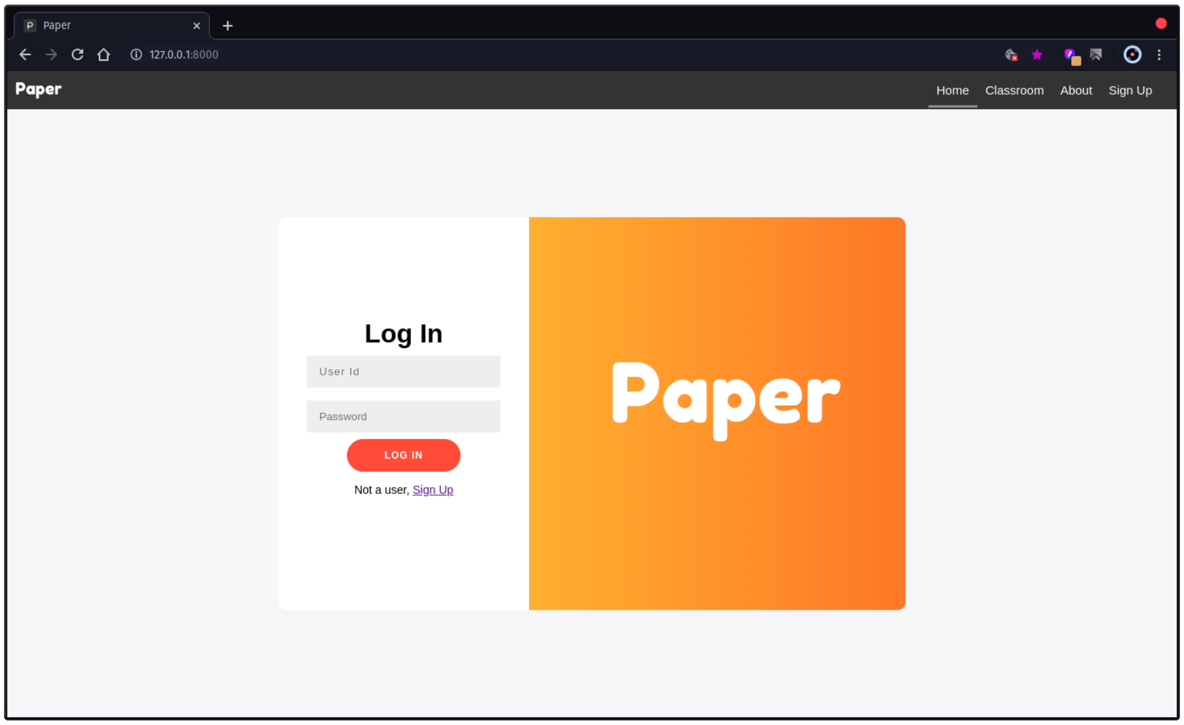
```
def home(request):
   if request.path == '/signup/':
       form = CreateUserForm()
       if request.method == 'POST' and request.POST['action'] == 'sign up':
           data = request.POST
           def generateUserId(userType):
               if userType in ['T','S']:
                   if userType == 'T':
                       userId = '1'
                   else:
                       userId = '2'
                   for i in range(0,5):
                       userId += str(random.randint(0,9))
                   userExist = User.objects.filter(username =
userId).count()
                   if userExist == 0 and userId != '':
                       return int(userId)
                   else:
                       generateUserId(userType)

           formData = {'first_name': data['first_name'], 'last_name':
data['last_name'], 'password1': data['password1'], 'password2':
data['password2'],}
           uid = generateUserId(data['userType'])
           formData['username'] = uid
           form = CreateUserForm(formData)
           if form.is_valid():
               form.save()
               new_user =
authenticate(username=form.cleaned_data['username'],
                                   password=form.cleaned_data['password1'],
                                   )
               login(request, new_user)

               if data['userType'] == 'T':
                   name = data['first_name'] + ' ' + data['last_name']
```

```
                    t = Teacher(name=name, id=uid)
                    t.save()
                else:
                    name = data['first_name'] + ' ' + data['last_name']
                    s = Student(name=name, id=uid)
                    s.save()
                return redirect('/')
        else:
            form = CreateUserForm()
        return render(request, 'paper/home.html',{'form':form})
    else:
        err = ''
        if request.method == 'POST' and request.POST['action'] == 'log in':
            username = request.POST.get('userId')
            password = request.POST.get('password')

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

            if user is not None:
                login(request, user)
                return redirect('/')
            else:
                err = 'User ID or password is incorrect.'

        return render(request, 'paper/home.html',
{'userAuthenticated':request.user.is_authenticated,'error':err})
```

The home view returns the home page as shown in the image. It also serves login and sign up requests.

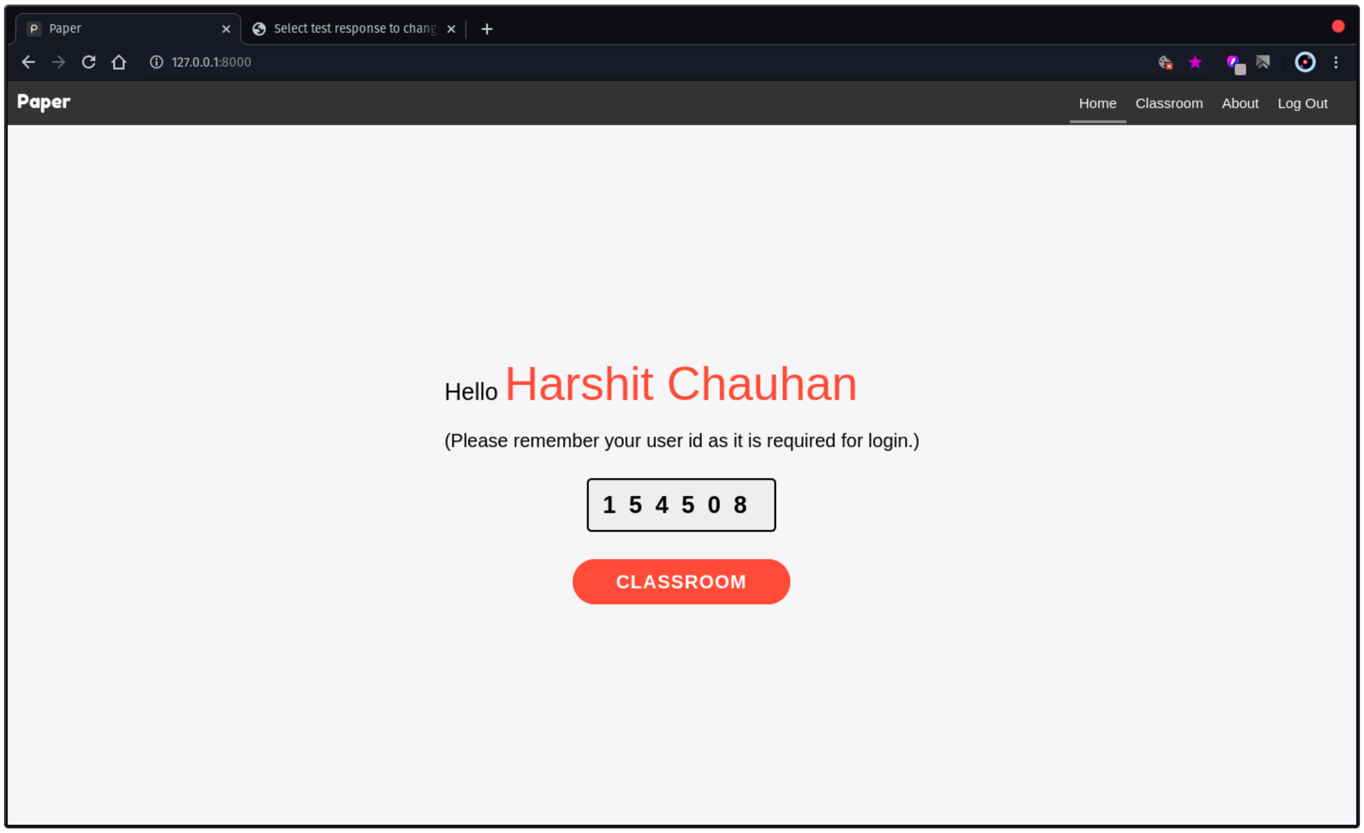


Sign Up



Log In

When signing up or logging in, the data like name, password, etc. is collected by HTML forms and is sent to the backend by POST requests. This data is then used to authenticate or create users and return a suitable HTML response as in the screenshot below.



Hello Harshit Chauhan

(Please remember your user id as it is required for login.)

1 5 4 5 0 8

Logged In

Logout

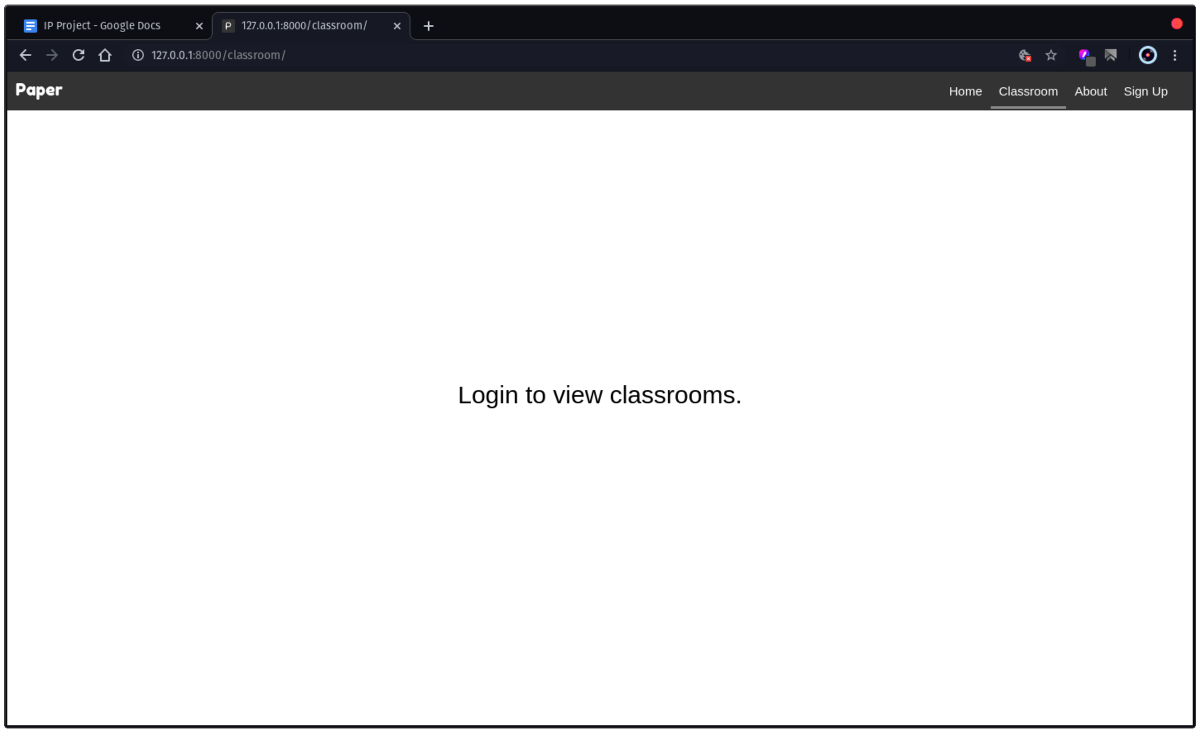
```
def logoutUser(request):
    logout(request)
    return redirect('home')
```

Logs out the user on clicking the logout button in the navigation bar.

Login Required

```
def loginRequired(request):
    return render(request,
'paper/loginRequired.html',{'userAuthenticated':request.user.is_authenticated
})
```

This view is called if the user is not logged in . It prevents the user from accessing the classroom as it requires sign in .



Login to view classrooms.

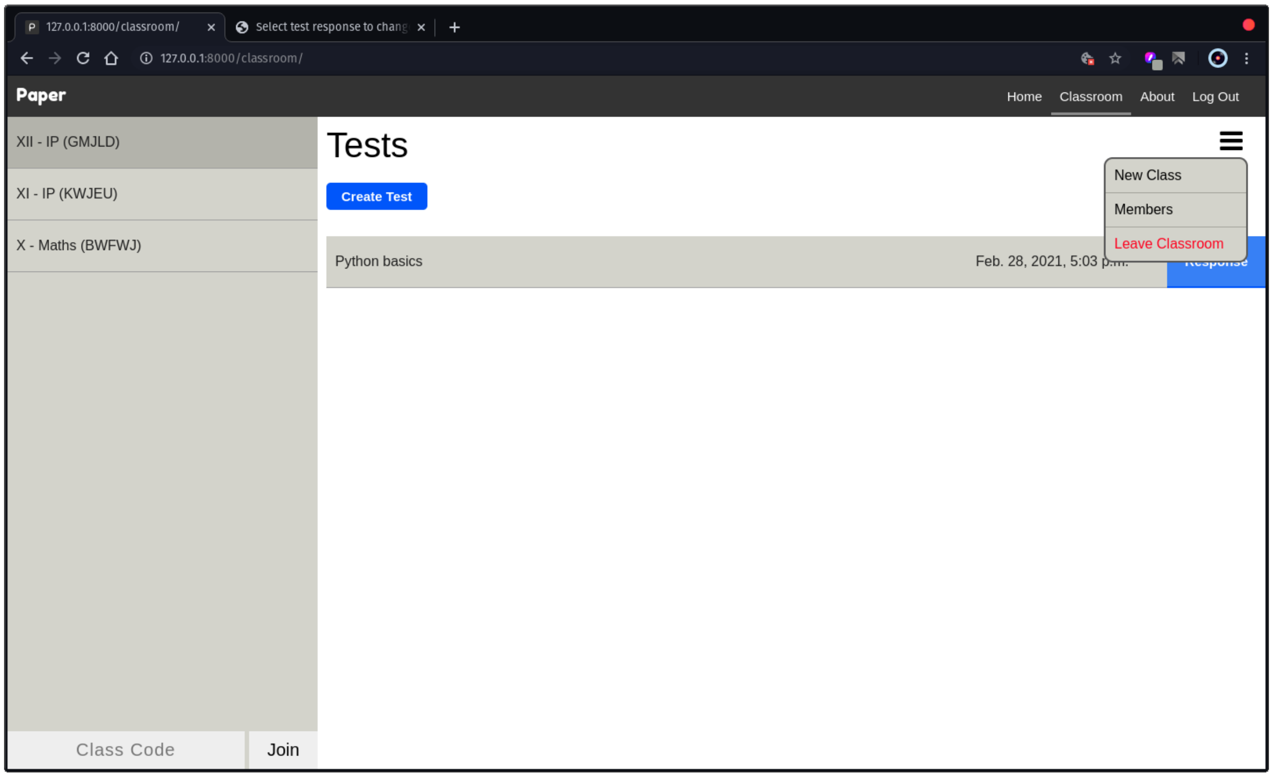
Teacher

```
def teacher(request,pk_classroom=0):
   currentUser = int(request.user.username)
   teacher = Teacher.objects.get(id=currentUser)
   if pk_classroom == 0:
       try:
           className = teacher.classes.all()
           classCode = str(className[0])
           tempClass = Classroom.objects.get(class_code=classCode[0:5])
           pk_classroom = tempClass.id
           classRoom = Classroom.objects.get(id=pk_classroom)
           tests =
Test.objects.filter(classes=pk_classroom,creator=currentUser)
           mssg = ''
       except:
           classRoom = tests = testGiven = testList = ''
           mssg = 'Not part of any class'

   else:
       classRoom = Classroom.objects.get(id=pk_classroom)
       tests =
Test.objects.filter(classes=pk_classroom,creator=currentUser)
       mssg = ''


   if request.method == 'POST':
       if 'className' in request.POST:
           classData = request.POST
           className = classData['className']

           cl = Classroom.objects.values_list('class_code')
           classCodeList = [x[0] for x in cl]
           classCode = generateClassCode(classCodeList)
           formData =
{'class_name':className,'teachers':[teacher],'class_code':classCode}
           form = ClassroomForm(formData)
           if form.is_valid():
               form.save()
               print("form sent")
               currentClassroom =
Classroom.objects.get(class_code=classCode)
               teacher.classes.add(currentClassroom)

       elif 'classCode' in request.POST:
           classroom = request.POST
           classCode = classroom['classCode']
           try:
               currentClassroom =
Classroom.objects.get(class_code=classCode)
               teacher.classes.add(currentClassroom)
               currentClassroom.teachers.add(teacher)
               print('teacher added to class')
           except:
               print('Invalid class code.')

       return HttpResponseRedirect("/classroom/")

   return render(request, 'paper/teacher.html',{'teacher':
teacher,'tests':tests,'currentClass':classRoom,'noClass':mssg,'userAuthenti
cated':request.user.is_authenticated})
```

The teacher view takes the request and pk_classroom arguments to return the teacher's version of the classroom page.

Request contains the user details and and form data that is submitted by the user.

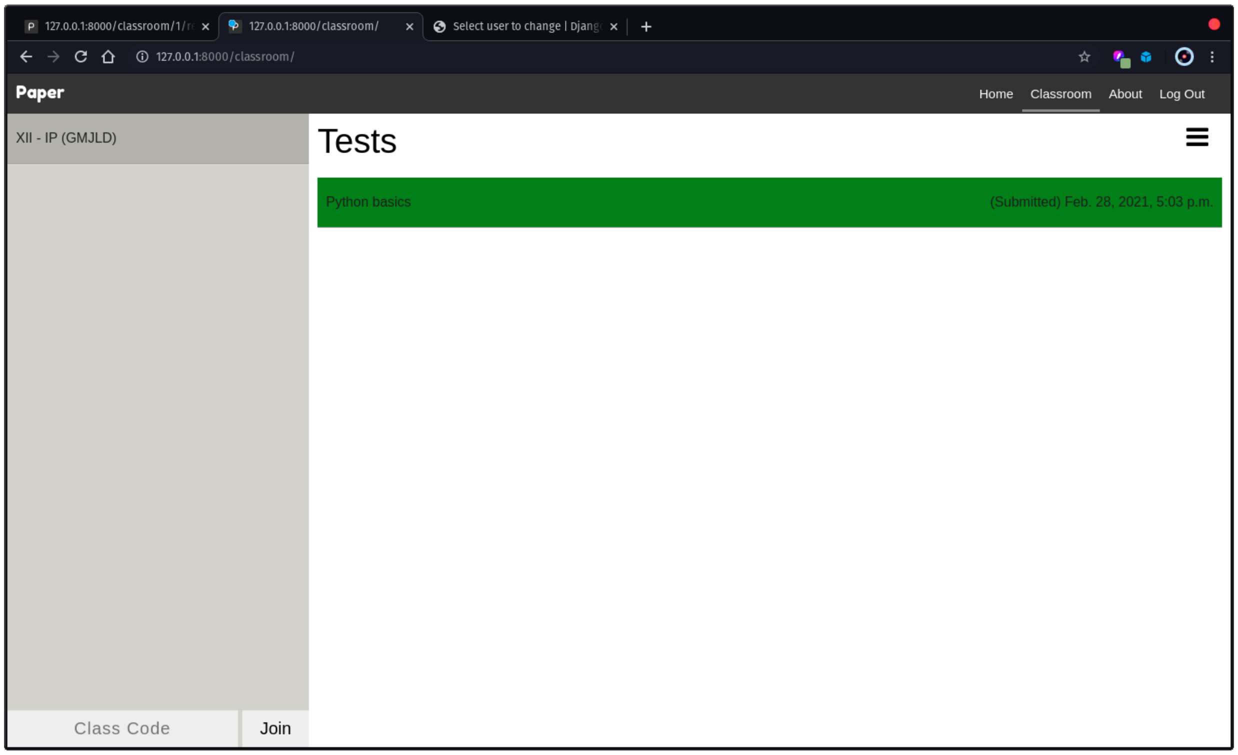
pk_classroom is a variable denoting the class number in the url (e.g. 127.0.0.1:8000/classroom/1/). It is used to get the data of the current classroom which is then returned as an Http response.



Student

```
def student(request,pk_classroom=0):
    currentUser = int(request.user.username)
    student = Student.objects.get(id=currentUser)
    if pk_classroom == 0:
        try:
            className = student.classes.all()
            classCode = str(className[0])
            tempClass = Classroom.objects.get(class_code=classCode[0:5])
            pk_classroom = tempClass.id
            classRoom = Classroom.objects.get(id=pk_classroom)
            tests =  Test.objects.filter(classes=pk_classroom)
            testList = TestResponse.objects.filter(student=currentUser)
            testGiven = []
            for i in testList:
                testGiven.append(i.test.id)
            mssg = ''
            hideElement = ''
        except:
            classRoom = tests = testGiven = testList = ''
            mssg = 'Not part of any class'
            hideElement = 'hide'

    else:
        classRoom = Classroom.objects.get(id=pk_classroom)
        tests =  Test.objects.filter(classes=pk_classroom)
        testList = TestResponse.objects.filter(student=currentUser)
        testGiven = []
        for i in testList:
            testGiven.append(i.test.id)
        mssg = ''
        hideElement = ''

    if request.method == 'POST':
        if 'classCode' in request.POST:
            classroom = request.POST
            classCode = classroom['classCode']
            try:
                currentClassroom = 
Classroom.objects.get(class_code=classCode)
                student.classes.add(currentClassroom)
                currentClassroom.students.add(student)
                print('student added to class')
            except:
                print('Invalid class code.')
            return HttpResponseRedirect("/classroom/")

    return 
render(request,'paper/student.html',{'student':student,'tests':tests,'curre
ntClass':classRoom,'testGiven':testGiven,'noClass':mssg,'hideElement':hideE
lement,'userAuthenticated':request.user.is_authenticated})
```

Student view is similar to the teacher view with the difference being that it returns the student's version of classroom with allow taking test, etc. while the teacher one allowed to create test and see responses.


127.0.0.1:8000/classroom/1/
127.0.0.1:8000/classroom/
Select user to change | Djang
127.0.0.1:8000/classroom/
Paper
Home
Classroom
About
Log Out
XII - IP (GMJLD)
Tests
Class Code
Join

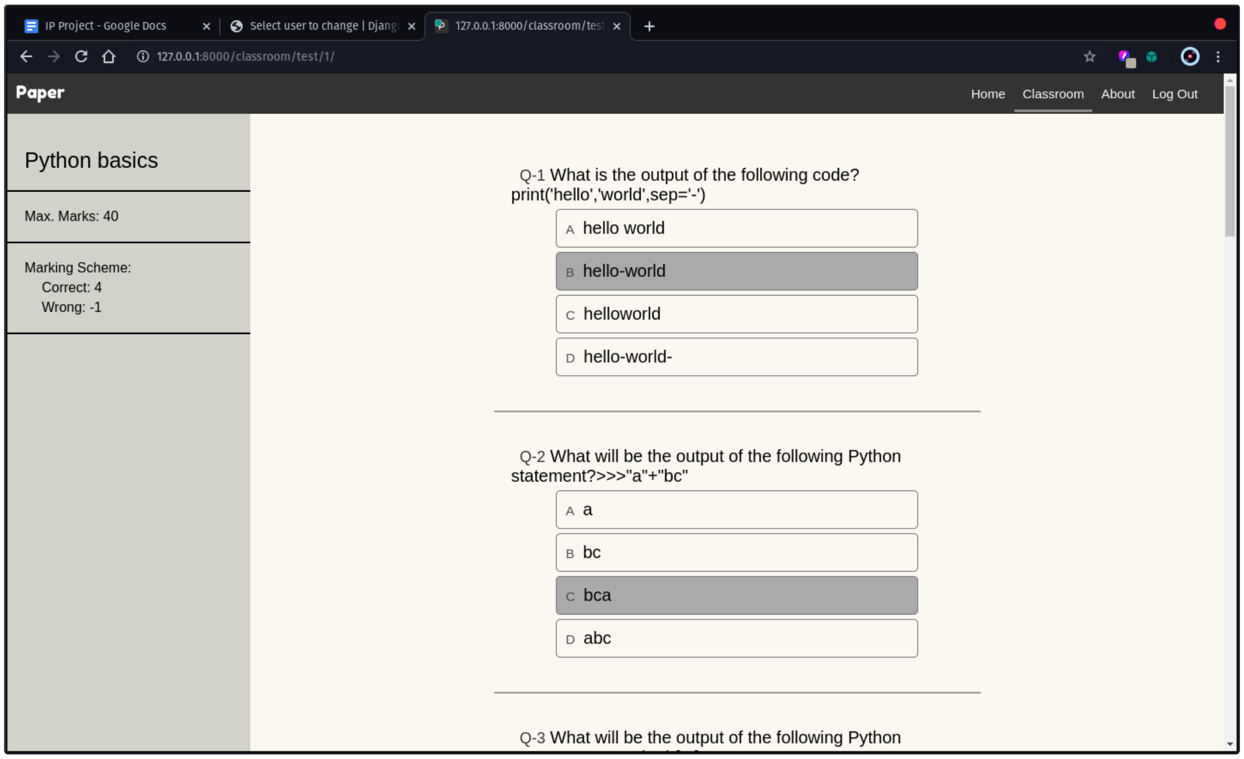
Python basics (Submitted) Feb. 28, 2021, 5:03 p.m.

Test

```
def test(request,pk_test):
    currentUser = int(request.user.username)
    testContent = Test.objects.get(id=pk_test)
    paperStr = testContent.test_paper
    paper = json.loads(paperStr)
    questions = paper['questions']

    student = Student.objects.get(id=currentUser)
    studentId = student.id
    testId = testContent.id

    mm = int(paper['correct']) * len(questions)
    testDetails =
{'topic':testContent.topic,'wrong':paper['wrong'],'correct':paper['correct'
],'mm':mm}
    if request.method == 'POST':
        paperResponse = request.POST
        marksObtained = 0
        answerData = []
        for i in questions:
            answer = i['answer']
            try:
                choice = paperResponse['qID-'+str(i['qid'])]
            except:
                choice = ''
            if choice == answer:
                status = 'Correct'
                marksObtained += int(paper['correct'])
            elif choice != answer and choice in ['A','B','C','D']:
                status = 'Wrong'
                marksObtained += int(paper['wrong'])
            else:
                status = 'Unanswered'
            ans = {'qid':i['qid'],'optionSelected':choice,'status':status}
            answerData.append(ans)
        answerData1 = json.dumps(answerData)
        formData =
{'student':studentId,'test':testId,'score':marksObtained,'response':answerD
ata1}
        form = TestResponseForm(formData)
        if form.is_valid():
            form.save()
        return HttpResponseRedirect("/classroom/")

    return render(request,
'paper/test.html',{'questions':questions,'details':testDetails,'userAuthent
icated':request.user.is_authenticated})
```

Test view takes the data of the test selected by the student from the tests model and return it as Http response which is rendered in the **test.html** template

The student can then take the test and submit it which is sent back by the POST request and after cleaning the data, it is stored in the test responses model.


IP Project - Google Docs
Select user to change | Django
127.0.0.1:8000/classroom/test
127.0.0.1:8000/classroom/test/1/
Paper
Home
Classroom
About
Log Out

Python basics

Max. Marks: 40

Marking Scheme:
Correct: 4
Wrong: -1

Q-1 What is the output of the following code?
print('hello','world',sep='-')

A hello world
[✓ B hello-world]
C helloworld
D hello-world-

Q-2 What will be the output of the following Python statement?>>>"a"+"bc"

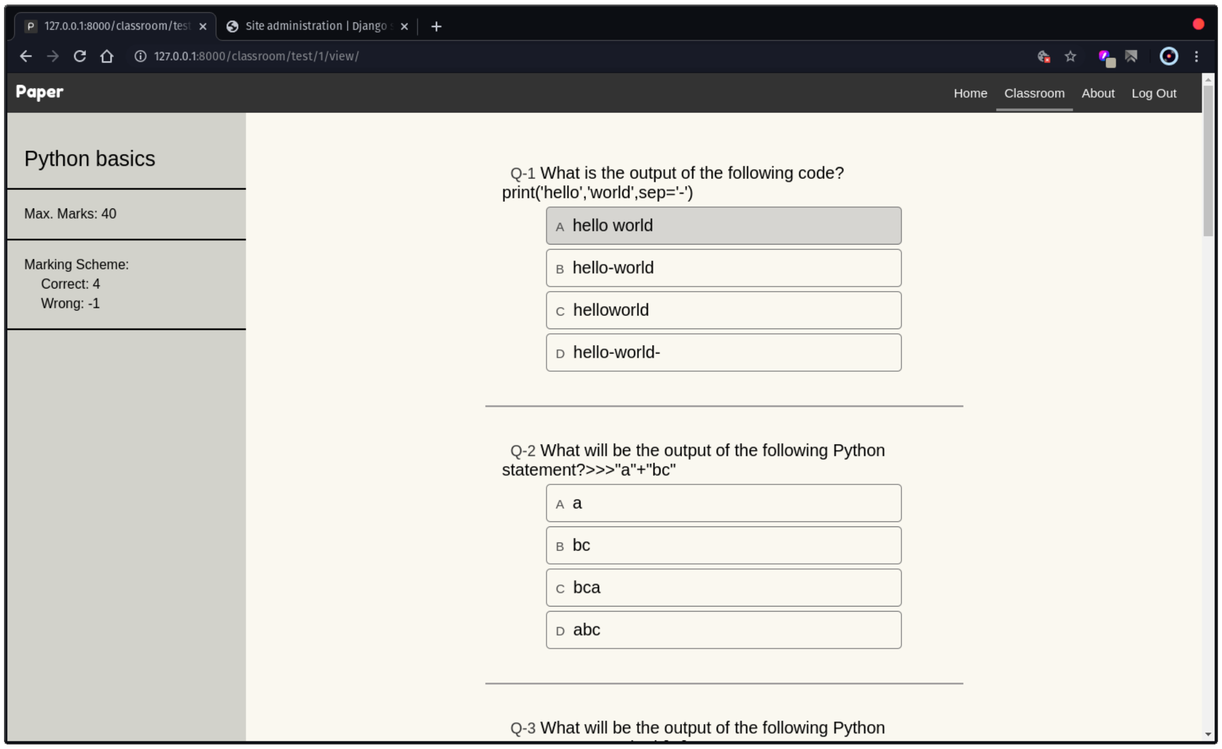
A a
B bc
[✓ C bca]
D abc

Q-3 What will be the output of the following Python

View test

```
def viewTest(request,pk_test):
    currentUser = int(request.user.username)
    testContent = Test.objects.get(id=pk_test)
    paperStr = testContent.test_paper
    paper = json.loads(paperStr)
    questions = paper['questions']

    mm = int(paper['correct']) * len(questions)
    testDetails =
{'topic':testContent.topic,'wrong':paper['wrong'],'correct':paper['correct'
],'mm':mm}

    return render(request,
'paper/test.html',{'questions':questions,'details':testDetails,'submitBtnCl
ass':'hide','userAuthenticated':request.user.is_authenticated})
```

This views renders the test for the teacher so that he/she can check the test paper for errors and other issues after its creation.


127.0.0.1:8000/classroom/test
Site administration | Django
127.0.0.1:8000/classroom/test/1/view/
Paper
Home
Classroom
About
Log Out

Python basics

Max. Marks: 40

Marking Scheme:
Correct: 4
Wrong: -1

Q-1 What is the output of the following code?
print('hello','world',sep='-')

A hello world
B hello-world
C helloworld
D hello-world-

Q-2 What will be the output of the following Python statement?>>>"a"+"bc"

A a
B bc
C bca
D abc

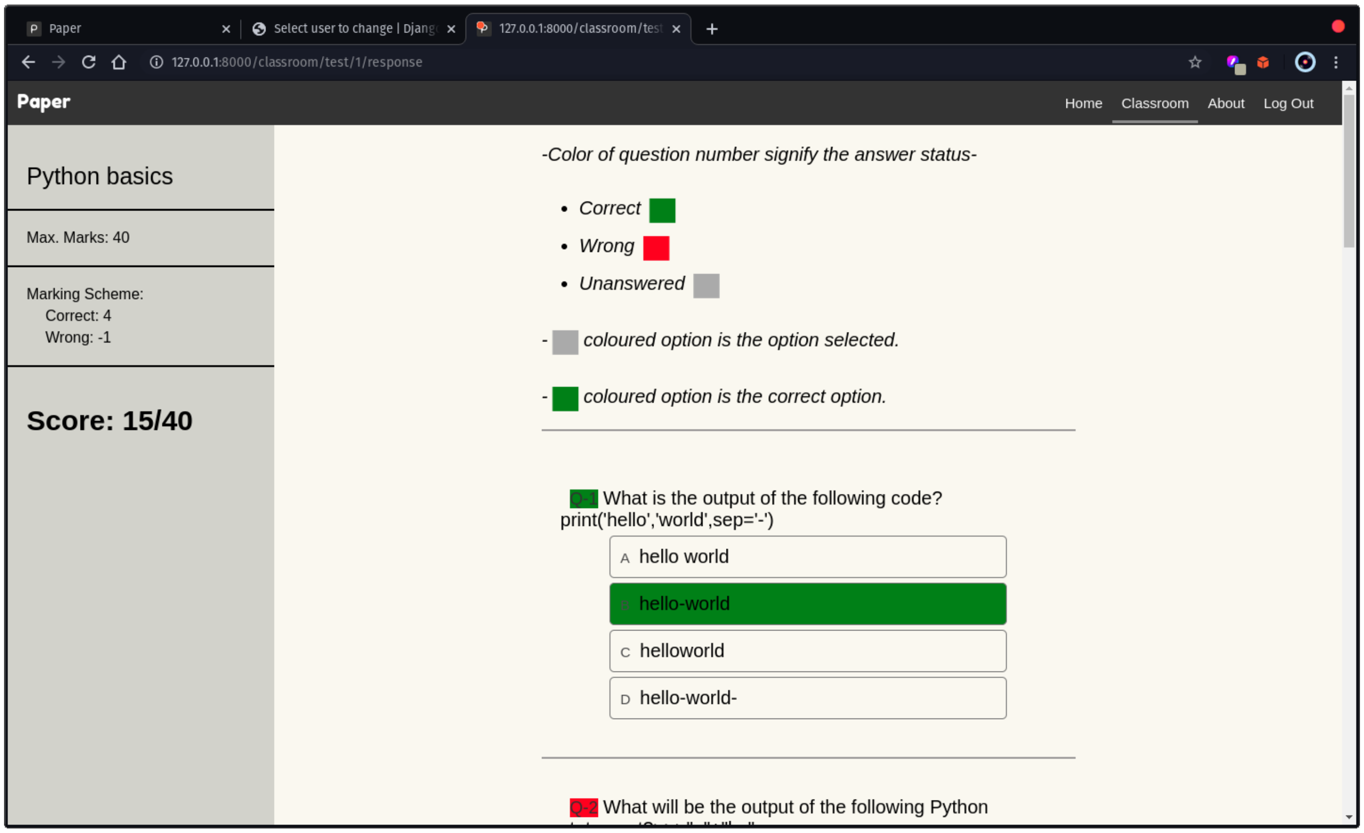
Q-3 What will be the output of the following Python

Student test response

```
def studentTestRespone(request,pk_test):
   currentUser = int(request.user.username)
   student = Student.objects.get(id=currentUser)
   test = Test.objects.get(id=pk_test)
   testResponse = TestResponse.objects.filter(student=student,test=test)
   testResponse = testResponse[0]
   paperResponseStr = testResponse.response
   paperResponse = json.loads(paperResponseStr)

   paperStr = test.test_paper
   paper = json.loads(paperStr)
   questions = paper['questions']
   for i in questions:
       for j in paperResponse:
           if i['qid'] == j['qid']:
               i['status'] = j['status']
               x = 'selectedOption' + j['optionSelected']
               i[x] = 'option-selected'
               y = 'correctOption' + i['answer']
               i[y] = 'answer-selected'
               if j['status'] == 'Correct':
                   i['correct'] = 'correct-q'
               elif j['status'] == 'Wrong':
                   i['incorrect'] = 'incorrect-q'
               else:
                   i['unanswered'] = 'unanswered-q'

           else:
               i['optionSelected'] = ''
               i['status'] = 'Unanswered'
   score = testResponse.score


   mm = int(paper['correct']) * len(questions)
   testDetails =
{'topic':test.topic,'wrong':paper['wrong'],'correct':paper['correct'],'mm':
```

```
mm,'score':score}

    return
render(request,'paper/studentTestResponse.html',{'questions':questions,'det
ails':testDetails,'userAuthenticated':request.user.is_authenticated})
```

This view takes the test paper from the test model and student response from the test response model and returns the student's test paper with markings for the question status, etc. which the student can use in order to analyze his/her performance.


Paper
Select user to change | Djang
127.0.0.1:8000/classroom/test
127.0.0.1:8000/classroom/test/1/response
Paper
Home
Classroom
About
Log Out

Python basics

Max. Marks: 40

Marking Scheme:
Correct: 4
Wrong: -1

Score: 15/40

-Color of question number signify the answer status-

- *Correct* 🟩
- *Wrong* 🟥
- *Unanswered* ⬜

- ⬜ *coloured option is the option selected.*

- 🟩 *coloured option is the correct option.*

Q-1 What is the output of the following code?
print('hello','world',sep='-')

A hello world
B hello-world
C helloworld
D hello-world-

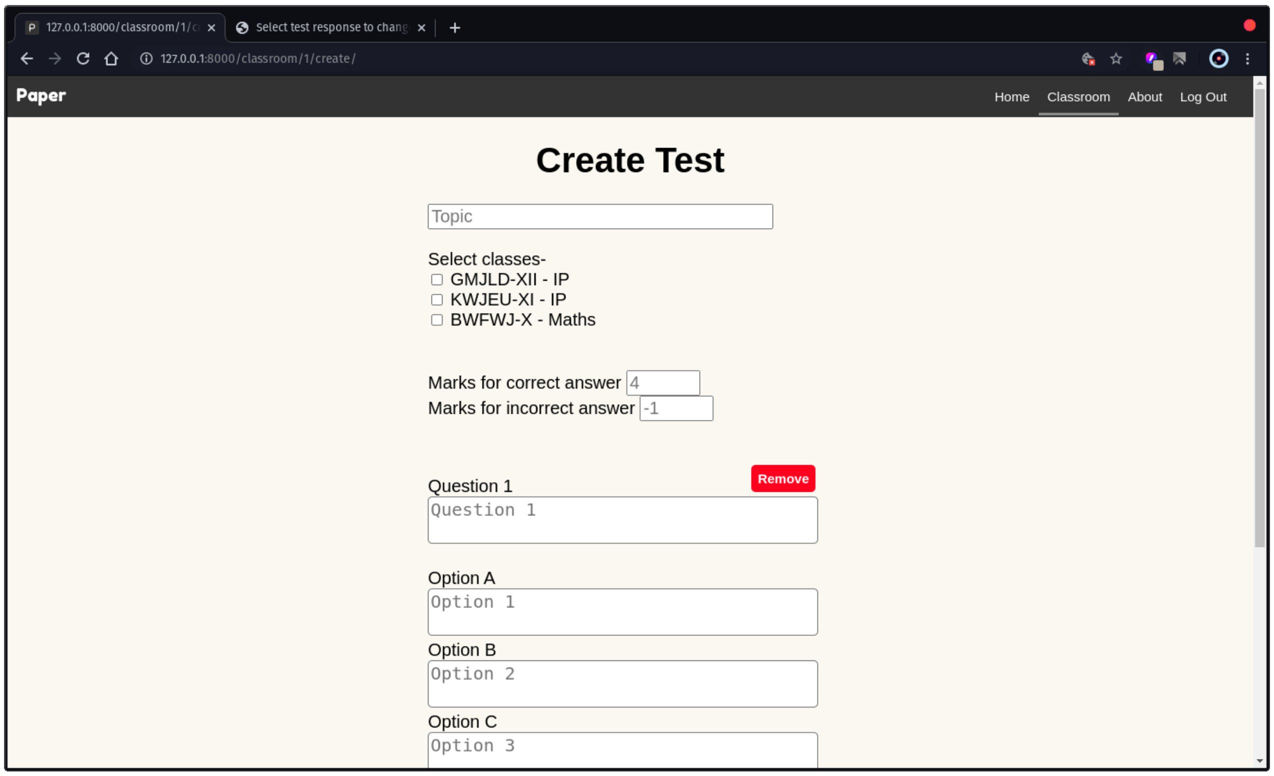
Q-2 What will be the output of the following Python

Create test

```
def createTest(request,pk_classroom):
   currentUser = int(request.user.username)
   creator = Teacher.objects.get(id=currentUser)
   if request.method == 'POST':
       data = request.POST
       classesId = data.getlist('classes')
       correct_marks = data['correct']
       wrong_marks = data['wrong']
       topic = data['topic']
       questions = []
       i = 1
       no = int((len(data)-4)/6)
       for i in range(1,no + 1):
           qId = i
           strId = str(i)
           question = data['question-' + strId]
           optA = data[strId + '-A']
           optB = data[strId + '-B']
           optC = data[strId + '-C']
           optD = data[strId + '-D']
           answer = data['answer-' + strId]
           que = {'qid':qId,'question':question,
'options':{'A':optA,'B':optB,'C':optC,'D':optD},'answer':answer}
           questions.append(que)
       paperData =
{'correct':int(correct_marks),'wrong':int(wrong_marks),'questions':
questions}
       paperData1 = json.dumps(paperData)
       formData = {'topic':topic,'creator':creator,'test_paper':paperData1}
       form = TestForm(formData)
       if form.is_valid():
           form.save()
           test = Test.objects.get(topic=topic)
           for i in classesId:
               classI = Classroom.objects.get(id=i)
               test.classes.add(classI)
```

```
            print("form sent")
            return HttpResponseRedirect("/classroom/")
    return render(request,
'paper/createTest.html',{'teacher':creator,'userAuthenticated':request.user
.is_authenticated})
```

This view takes the test data from the create test page of the teacher's classroom through POST request and then converts it into the suitable JSON format so that it can be saved in the tests model. Sample JSON data for the test and response can be found in the sample data folder.


127.0.0.1:8000/classroom/1/c
Select test response to chang
127.0.0.1:8000/classroom/1/create/
Paper
Home
Classroom
About
Log Out
Remove

Create Test

Topic

Select classes-
- GMJLD-XII - IP
- KWJEU-XI - IP
- BWFWJ-X - Maths

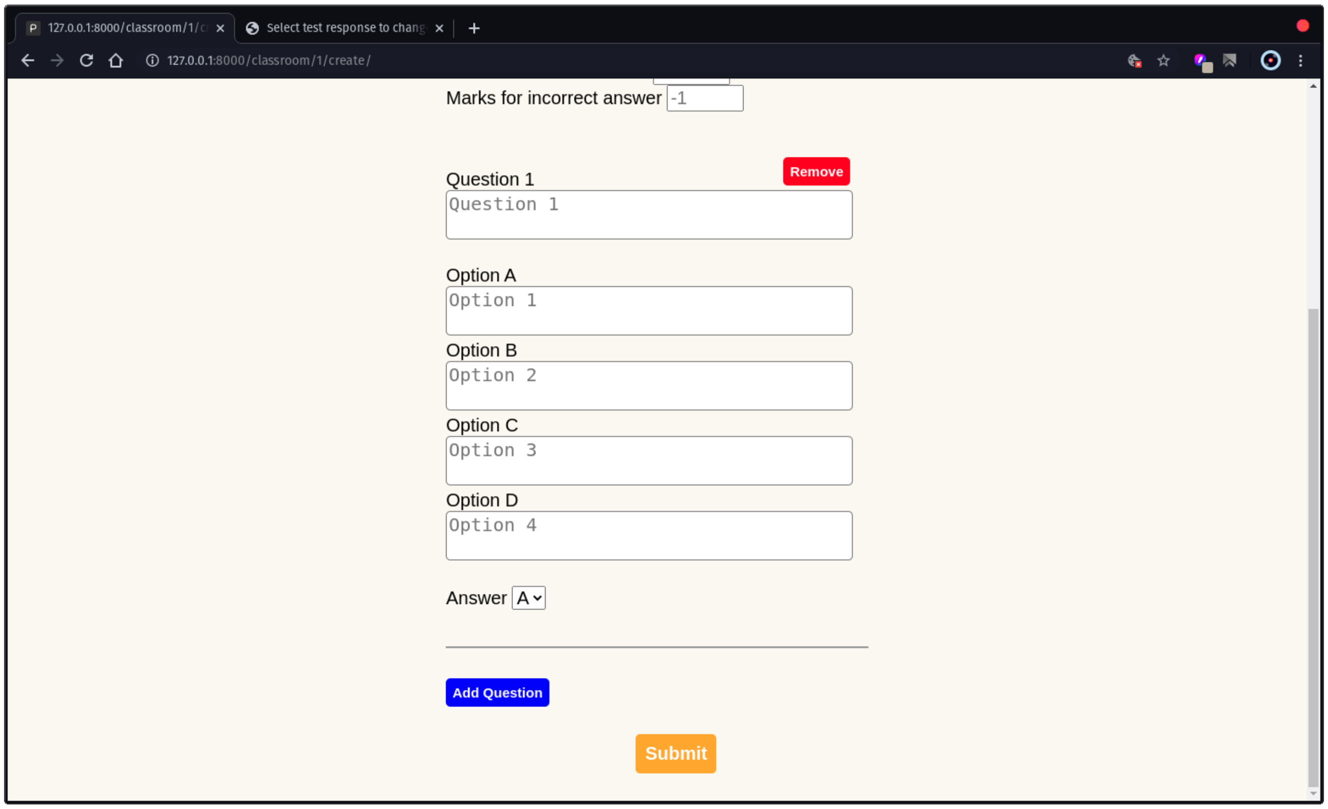
Marks for correct answer 4
Marks for incorrect answer -1

Question 1
Question 1

Option A
Option 1

Option B
Option 2

Option C
Option 3


127.0.0.1:8000/classroom/1/c
Select test response to chang
127.0.0.1:8000/classroom/1/create/

Marks for incorrect answer -1

Question 1 Remove

Question 1

Option A

Option 1

Option B

Option 2

Option C

Option 3

Option D

Option 4

Answer A

Response

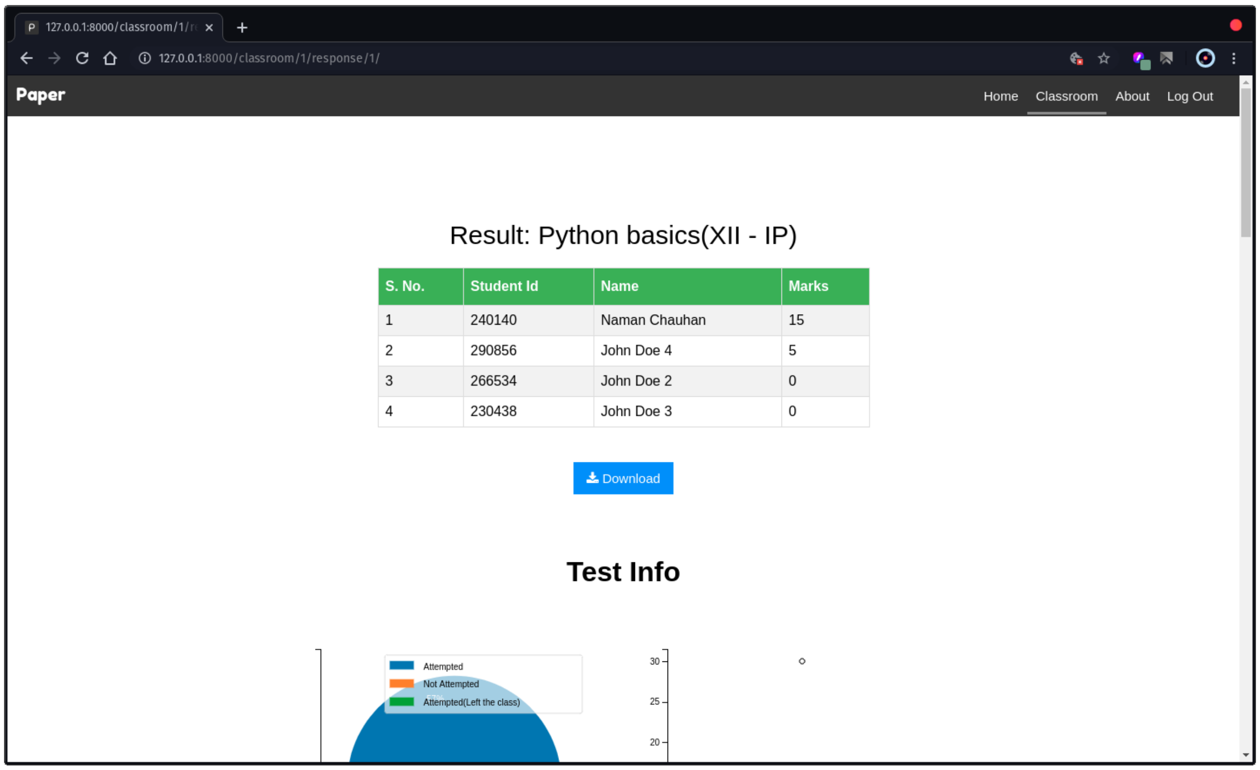
```
def response(request,pk_classroom,pk_test):
   currentUser = int(request.user.username)
   testId = Test.objects.get(id=pk_test)
   classRoom = Classroom.objects.get(id=pk_classroom)
   topic = testId.topic
   responseList =
TestResponse.objects.filter(test=pk_test).order_by('-score').values('studen
t','score')

   j = 1
   for i in responseList:
       if Student.objects.get(id=i['student']) in classRoom.students.all():
           i['rank'] = j
           studentId = i['student']
           i['student'] = Student.objects.get(id=studentId).name
           i['id'] = studentId
           j += 1

   testResponse = TestResponse.objects.filter(test=testId)
   paperStr = testId.test_paper
   paper = json.loads(paperStr)

   graphs = {}
   if len(responseList) > 0:
       graphs['attemptedGraph'] = attemptedGraph(classRoom,responseList)
       graphs['boxPlot'] = boxPlot(responseList)
       graphs['questionWiseBarGraph'] =
questionWiseBarGraph(paper,testResponse)
   else:
       graphs['attemptedGraph'] = ''
       graphs['boxPlot'] = ''
       graphs['questionWiseBarGraph'] = ''
```

```
    return render(request,
'paper/response.html',{'teacherId':currentUser,'testId':pk_test,'topic':top
ic,'currentClass':classRoom,'responseList':responseList,'graphs':graphs,'us
erAuthenticated':request.user.is_authenticated})
```

This function uses the plots forms **plots.py** module to show the class response of the test selected by the teacher.

Response contains the table of students and their marks, renders of the pie chart showing the percentage of students who have given the test, box plot of the test marks and question wise distribution of the options chosen for each of the questions. These graphs are rendered using matplotlib.pyplot library and mpld3 library is used to convert the resultant graph into HTML code.

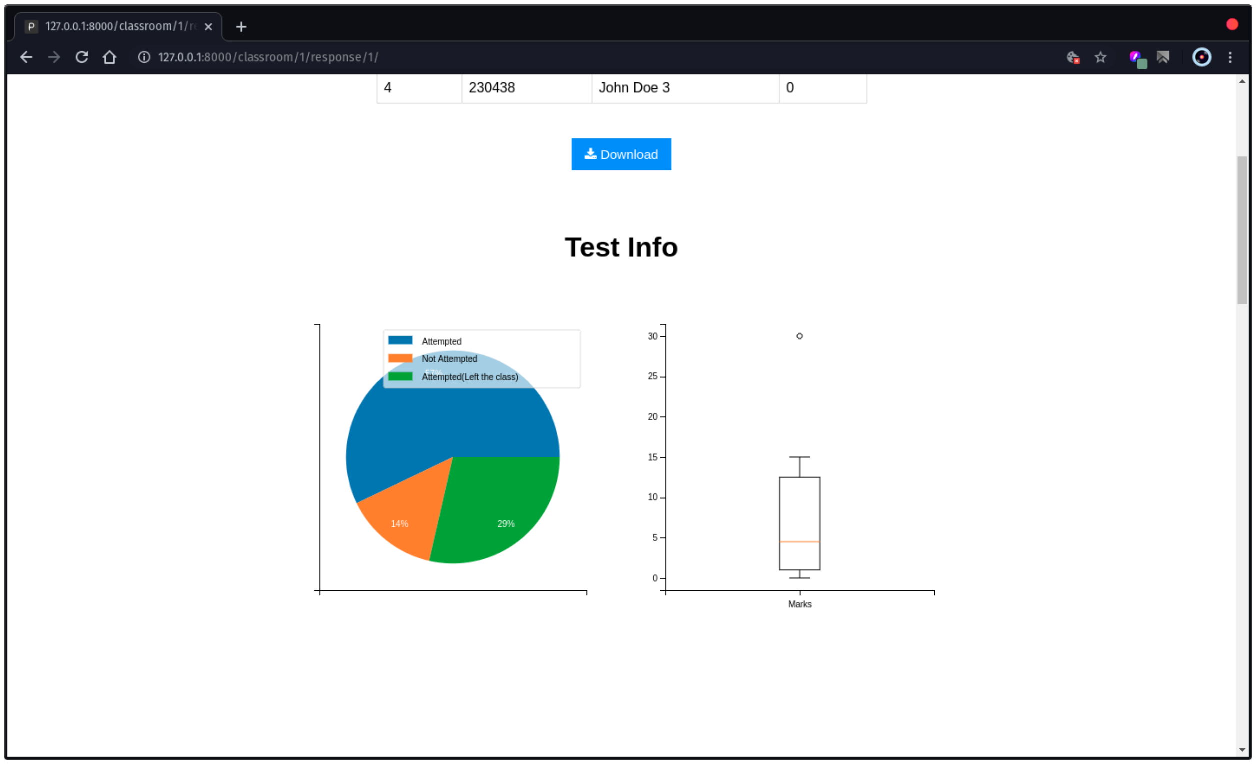
Teacher also has the option to download the csv file of the result table.



Result: Python basics(XII - IP)

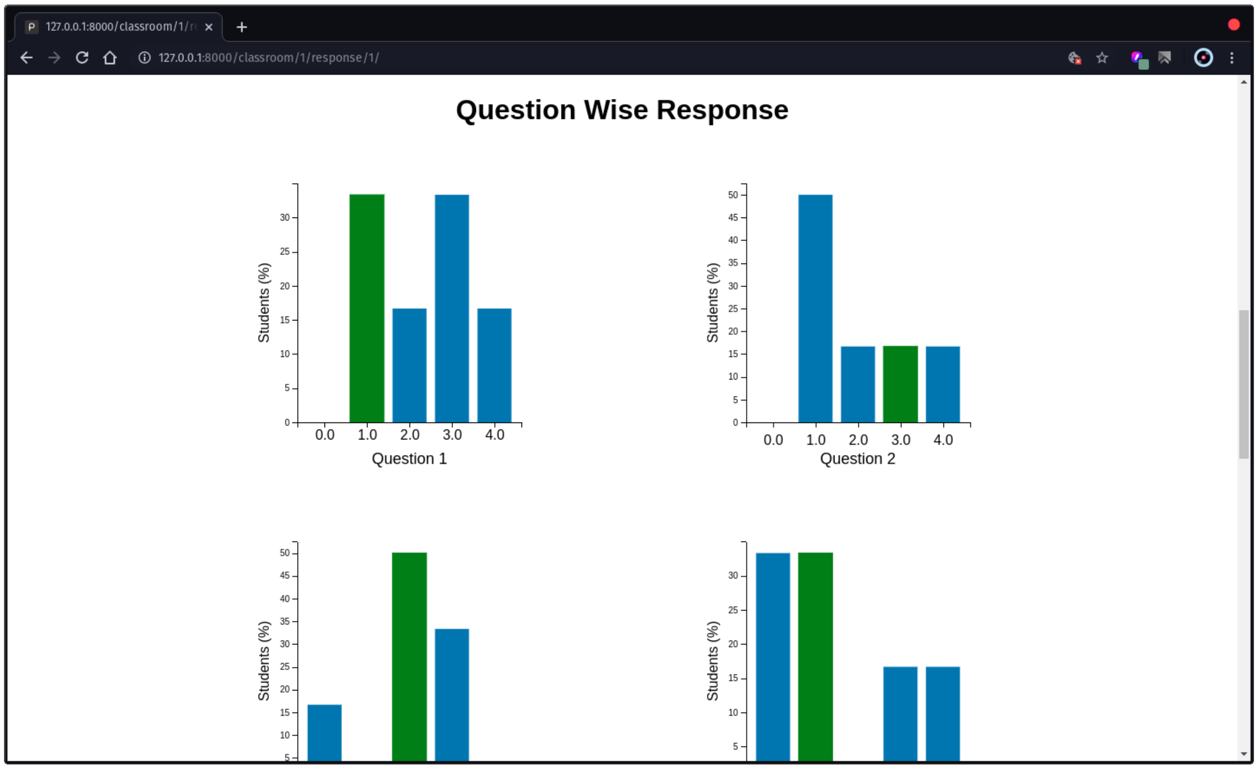
S. No.	Student Id	Name	Marks
1	240140	Naman Chauhan	15
2	290856	John Doe 4	5
3	266534	John Doe 2	0
4	230438	John Doe 3	0

Test Info



4	230438	John Doe 3	0

127.0.0.1:8000/classroom/1/response/1/
Download
Test Info
Attempted
Not Attempted
Attempted(Left the class)
14%
29%
Marks



Export

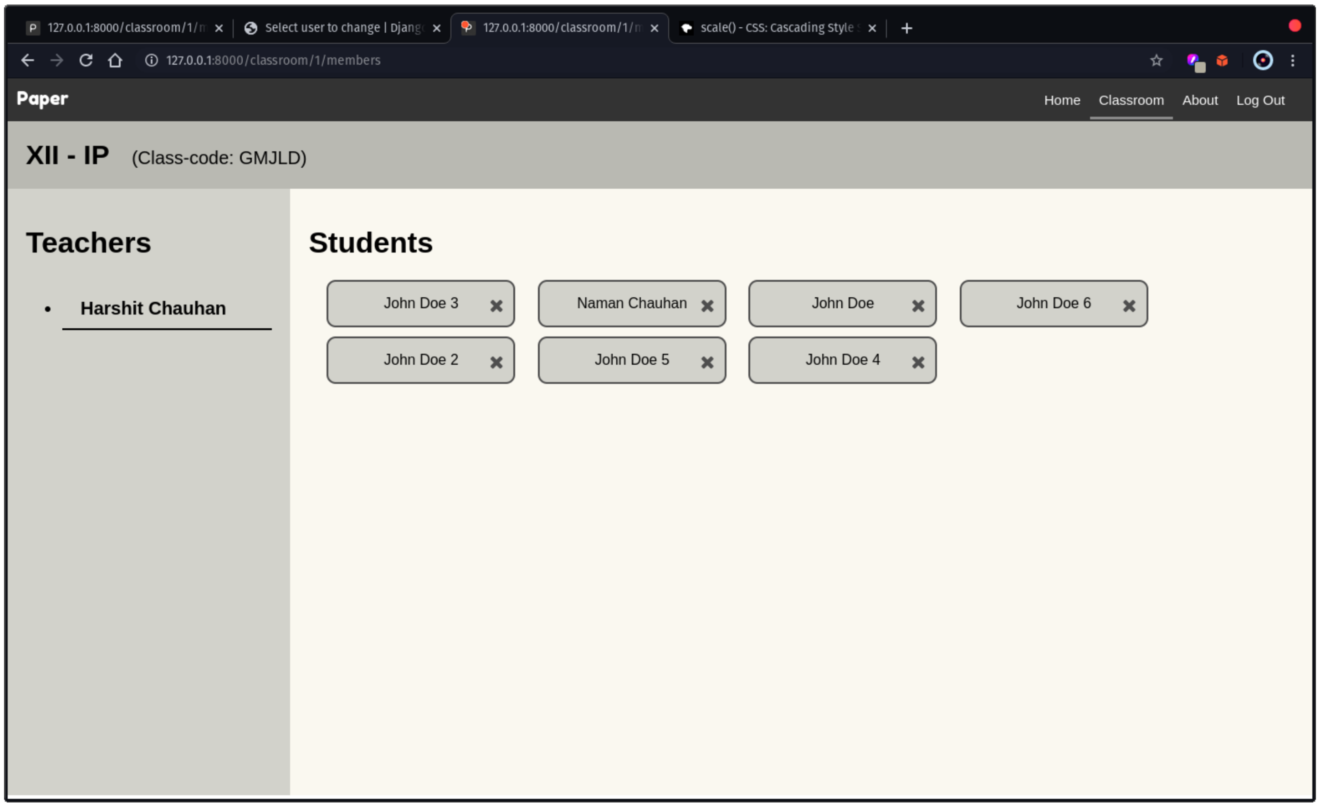
```
def export(request,pk_classroom,pk_test):
   currentUser = int(request.user.username)
   testId = Test.objects.get(id=pk_test)
   classRoom = Classroom.objects.get(id=pk_classroom)
   topic = testId.topic
   responseList = 
TestResponse.objects.filter(test=pk_test).order_by('-score').values('studen
t','score')

   j = 1
   for i in responseList:
       if Student.objects.get(id=i['student']) in classRoom.students.all():
           i['rank'] = j
           studentId = i['student']
           i['student'] = Student.objects.get(id=studentId).name
           i['id'] = studentId
           j += 1

   response = HttpResponse(content_type='text/csv')
   writer = csv.writer(response)
   writer.writerow(['rank','student Id','name','marks'])
   for i in responseList:
       if 'id' in i.keys():
           writer.writerow([i['rank'],i['id'],i['student'],i['score']])
   response['Content-Disposition'] = f'attachment; 
filename="{topic}({classRoom.class_name}).csv"'
   return response
```

Downloads the csv file of the student result on clicking the download button on the response page.

Classroom

```
def classroom(request,pk_classroom):
    currentUser = int(request.user.username)
    classRoom = Classroom.objects.get(id=pk_classroom)
    if currentUser // 100000 == 1:
        isTeacher = True
        teacher = Teacher.objects.get(id=currentUser)
        if request.method == 'POST':
            if 'leaveClassroom' in request.POST:
                print(request.POST)
                try:
                    classRoom.teachers.remove(teacher)
                    teacher.classes.remove(classRoom)
                except:
                    print('Unable to leave classroom.')
                return HttpResponseRedirect("/classroom/")
            else:
                data = request.POST
                print(data)
                for key, value in data.items():
                    if value == '\uf00d':
                        studentId = key.replace("student-","")
                try:
                    student = Student.objects.get(id=studentId)
                    classRoom.students.remove(student)
                    print(student, ' removed')
                except:
                    print('Unable to remove student')
            return
HttpResponseRedirect(f"/classroom/{classRoom.id}/members")

    elif currentUser // 100000 == 2:
        isTeacher = False
        student = Student.objects.get(id=currentUser)
        if request.method == 'POST':
            if 'leaveClassroom' in request.POST:
                print(request.POST)
                try:
                    classRoom.students.remove(student)
                    student.classes.remove(classRoom)
                except:
                    print('Unable to leave classroom.')
                return HttpResponseRedirect("/classroom/")

    return render(request,
'paper/classroom.html',{'classroom':classRoom,'teacher':isTeacher,'userAuth
enticated':request.user.is_authenticated})
```

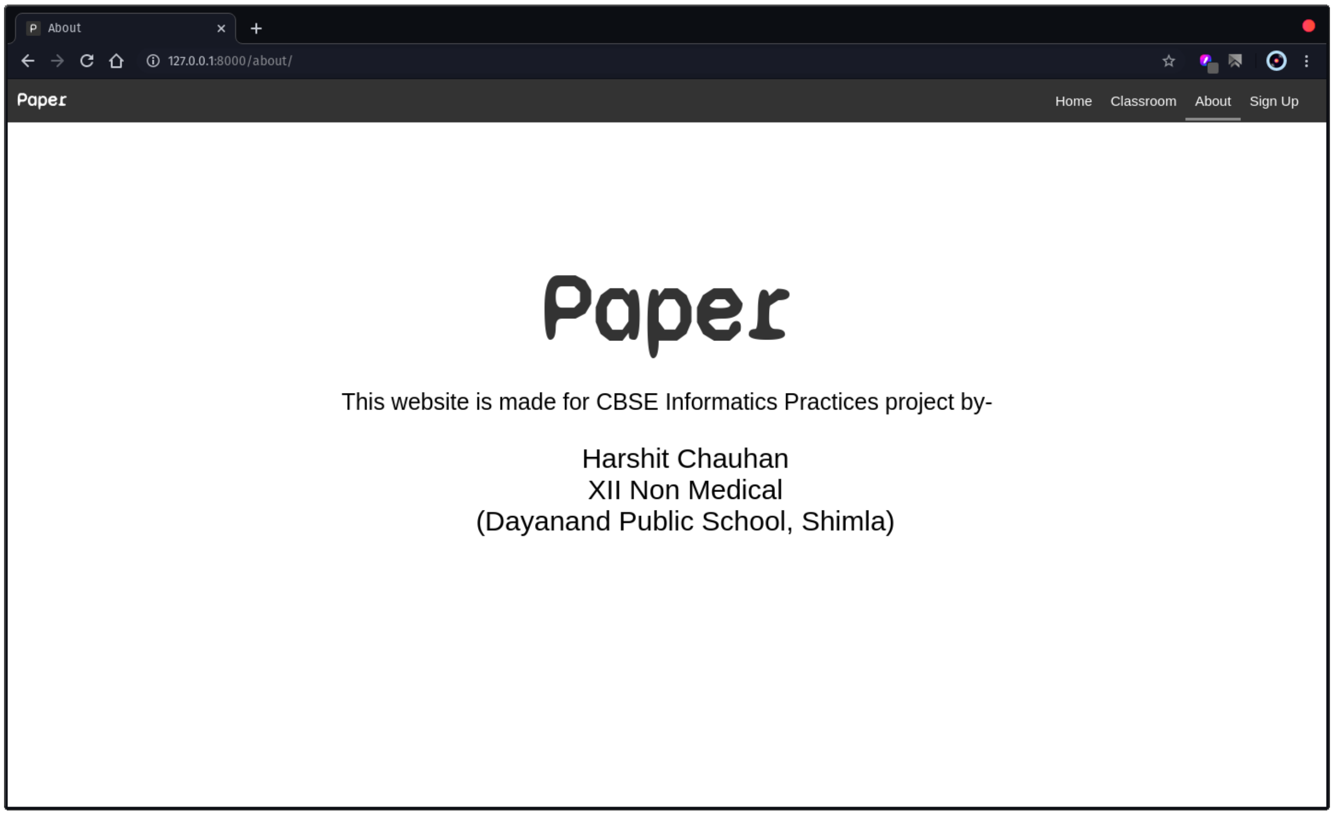
Shows the members of the current class and also allows the teacher to remove certain students from the classroom.



About

Returns the about page of the website.

```
def about(request):
    return render(request, 
'paper/about.html',{'userAuthenticated':request.user.is_authenticated})
```



Paper

This website is made for CBSE Informatics Practices project by-

Harshit Chauhan
XII Non Medical
(Dayanand Public School, Shimla)

Plots

```
import matplotlib.pyplot as plt
import mpld3
import json
import math

def attemptedGraph(classRoom,responseList):
   total = classRoom.students.count()
   attempted = responseList.count()
   attemptedButLeft = 0
   for i in responseList.values('student'):
       if len(classRoom.students.filter(id=i['student'])) != 1:
           attemptedButLeft += 1

   attempted = attempted - attemptedButLeft
   notAttempted = total - attempted

   fig = plt.figure(figsize = (4,4))
   x = [attempted,notAttempted,attemptedButLeft]
   label = ['Attempted','Not Attempted','Attempted(Left the class)']
   plt.pie(x, autopct='%1.0f%%', pctdistance=0.8, textprops={'color':"w"})
   plt.legend(labels=label)
   html_str = mpld3.fig_to_html(fig)
   return html_str

def boxPlot(responseList):
   marks=[]
   for i in responseList.values('score'):
       marks.append(i['score'])

   fig = plt.figure(figsize = (4,4))
   bp = plt.boxplot(marks, vert=1, labels=['Marks'])
   html_str = mpld3.fig_to_html(fig)
   return html_str
```

```
def questionWiseBarGraph(paper,testResponse):
   html_str_list = []
   questions = paper['questions']
   fig = plt.figure(figsize = (10,2.5*len(questions)))
   n = 0
   for i in questions:
       optionChosen = []
       m = 0
       for j in testResponse:
           paperResponseStr = j.response
           paperResponse = json.loads(paperResponseStr)

           for k in paperResponse:
               if i['qid'] == k['qid']:
                   optionChosen.append(k['optionSelected'])

           m += 1

       optionsCount = {i:optionChosen.count(i) for i in optionChosen}

       ans = {'A':0,'B':1,'C':2,'D':3}
       options = ['A','B','C','D','None']

       students = []


       for k in ['A','B','C','D','']:
           try:
               students.append(optionsCount[k])
           except:
               students.append(0)


       students = [x/m*100 for x in students]

       pltNo = n + 1
       pltRow = math.ceil(len(questions) / 2)
       pltCol = 2
```

```
        plt.subplot(pltRow, pltCol, pltNo)
        subplotBars = plt.bar(options,students)
        subplotBars[ans[i['answer']]].set_color('g')

        plt.xticks([0.0,1.0,2.0,3.0,4.0], ['A','B','C','D','None'],
fontsize=16)
        plt.ylabel('Students (%)',fontsize=16)
        plt.xlabel(f'Question {n+1}',fontsize=18)
        plt.subplots_adjust(wspace=1, hspace=0.5)

        n += 1

    html_str = mpld3.fig_to_html(fig)
    return html_str
```

Render the plots showed in the response view.

Conclusion

Bibliography

- Books-
    - Informatics Practices with Python by Preeti Arora
- Websites-
    - https://www.djangoproject.com/
    - https://stackoverflow.com/
    - https://www.tutorialspoint.com/
    - https://www.w3schools.com/
    - Django (3.0) Crash Course Tutorials | Customer Management App