## horizontal line



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**](#_xz72suvxqw8e) **1**

[Characteristics of Python](#_xmbm5iv3p0v4) **1**

[Django](#_gpjmrnwcd1zl) **2**

[Matplotlib](#_mvjjyaehk5ow) **2**

[**Setting Up**](#_pekbnn2r8z8o) **3**

[**Project Structure**](#_azqkuag49hex) **4**

[**Database Structure**](#_pha2unk4qzuk) **5**

[**URLS**](#_suvoiv6u9edu) **7**

[**Models**](#_fampqfqw3f6v) **9**

[**Modules**](#_3dl00xts36uc) **11**

[**Views**](#_z8iij1heiy3g) **12**

[Imports](#_d8tsa6rh4dzw) **12**

[Home](#_r97ssj73p50i) **13**

[Teacher](#_oks9amo5kj6x) **18**

[Student](#_xt1ni9a6uwcy) **21**

[Test](#_62qzqosx2vlw) **23**

[View test](#_yqhtadvosiv0) **25**

[Student test response](#_p1vmt7hr88z2) **26**

[Create test](#_7uzmqc9b4nwu) **28**

[Response](#_4kqcfm1g8ile) **31**

[Export](#_7y6i2vwvecqj) **34**

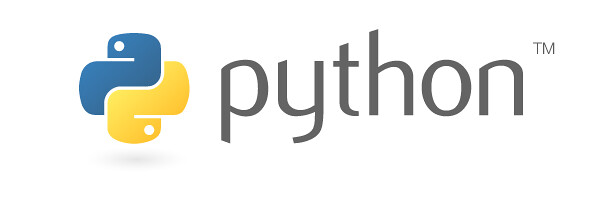
[Classroom](#_3s32vc6qlgua) **35**

[About](#_wb2rv0jju8wi) **37**

[**Plots**](#_o2gnmfbjc2wr) **38**

[**Conclusion**](#_csojsixc0euu) **41**

[**Bibliography**](#_dwfpw9fq03ea) **42**



# 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

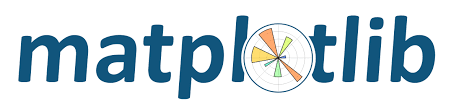
Following are important characteristics of Python Programming −

* 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.

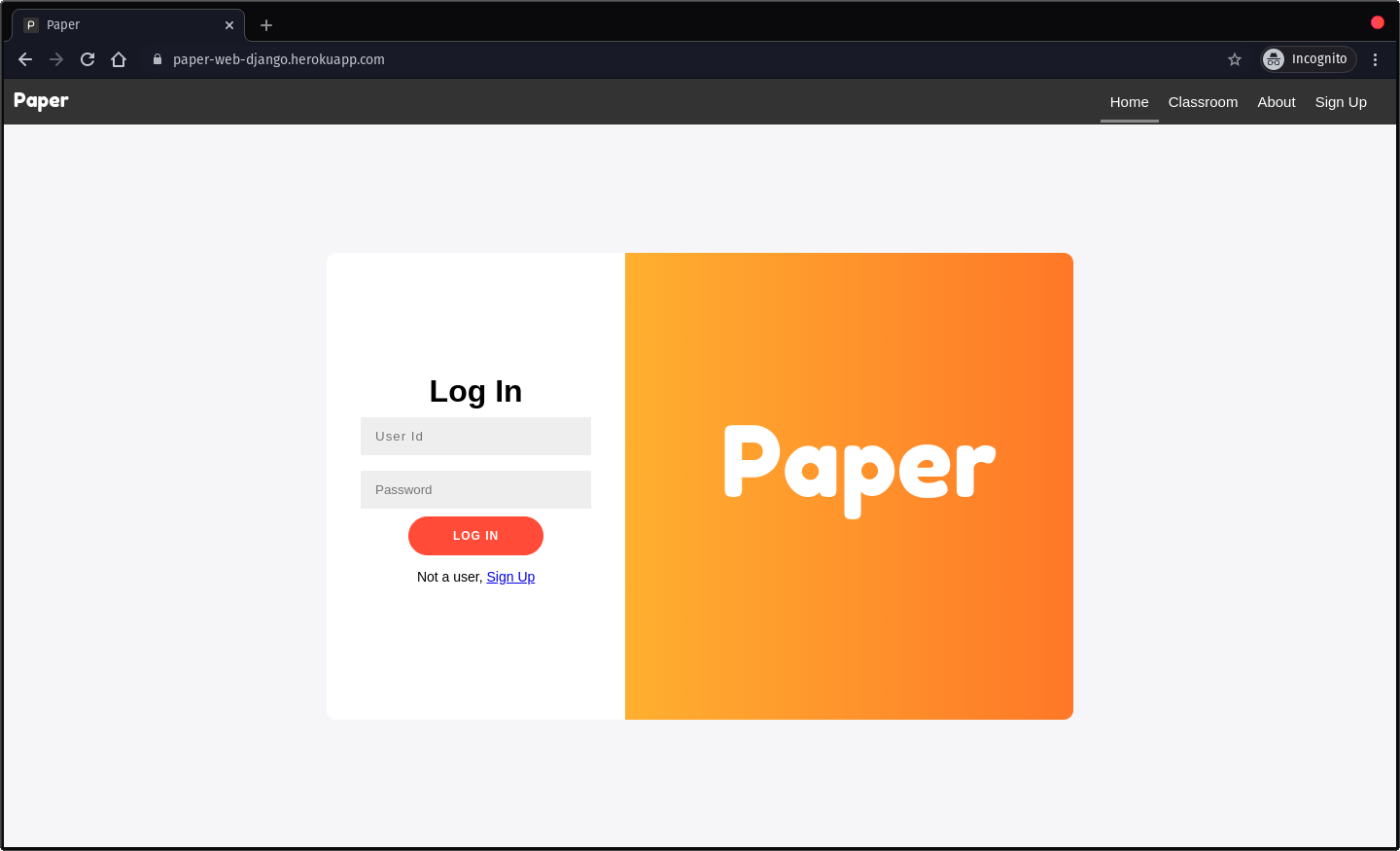


## Matplotlib

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.

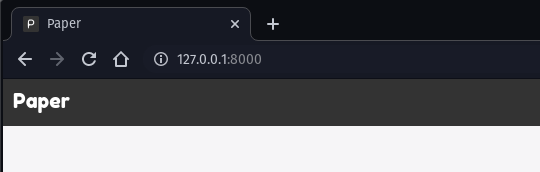
# Setting Up

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

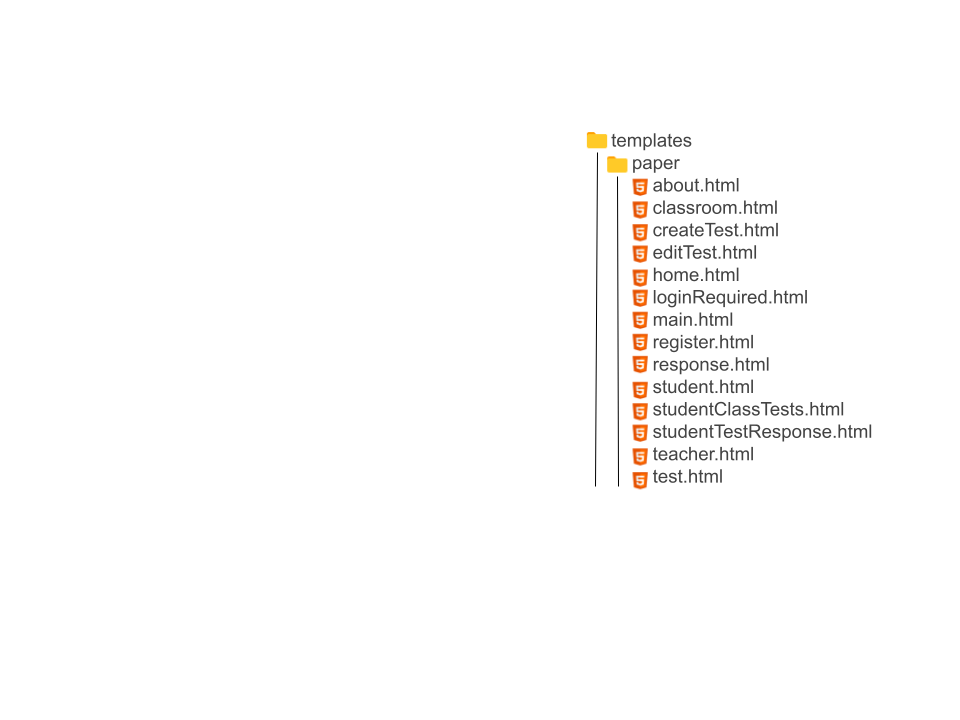


To run the project on your local machine follow the steps-

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.

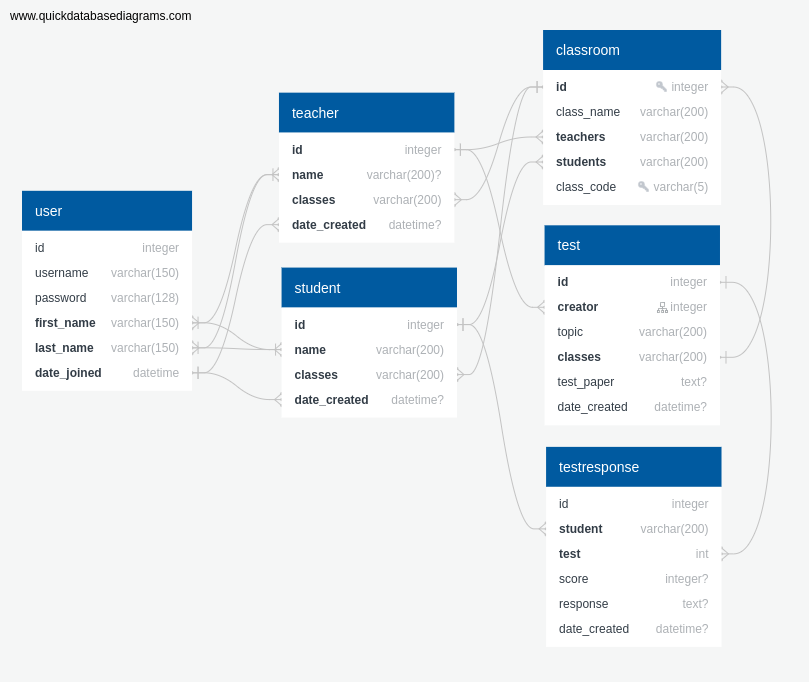


# Project Structure

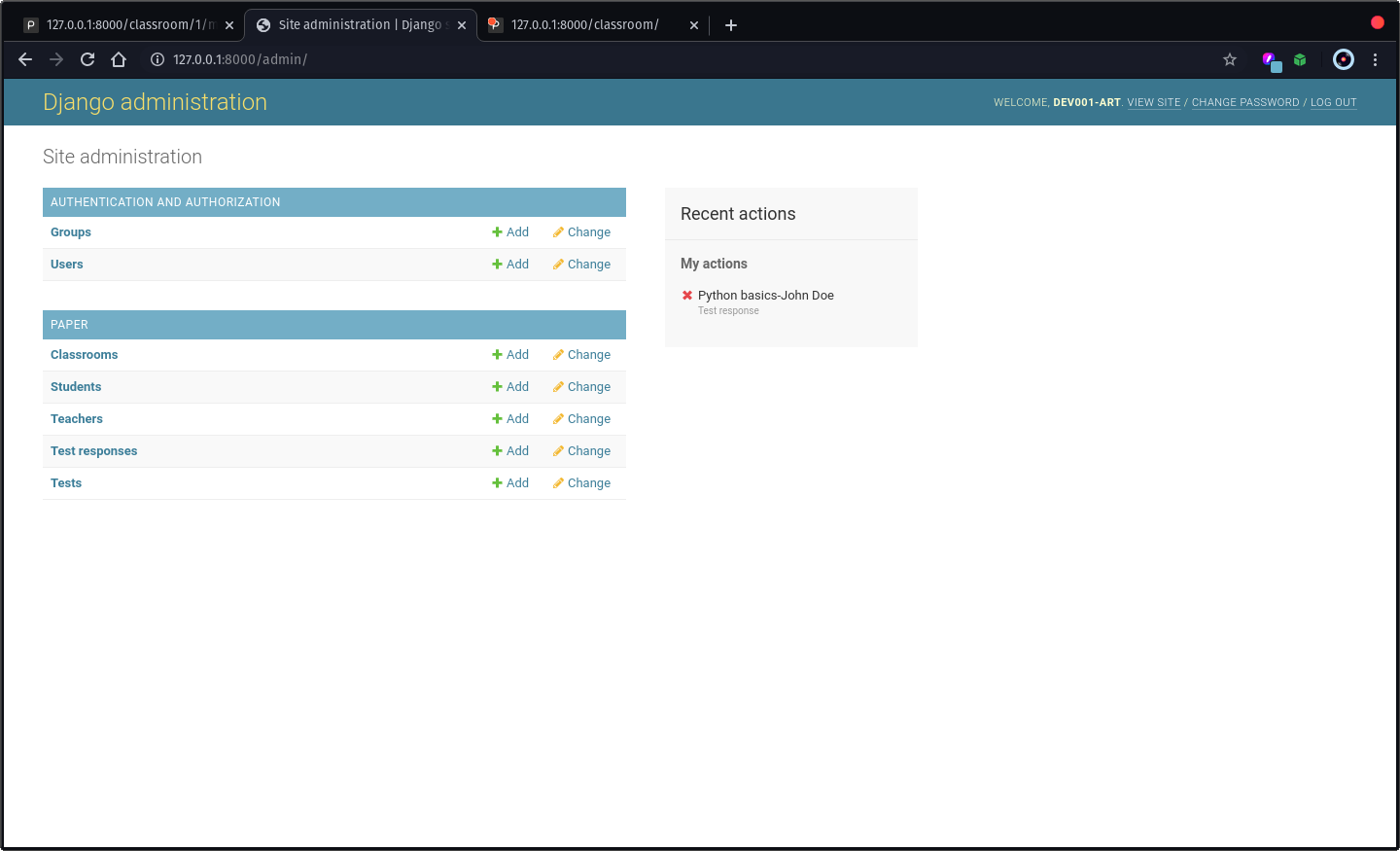
 ****

* **IP> IP** - 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



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



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

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.

### 

### 

### Logged In

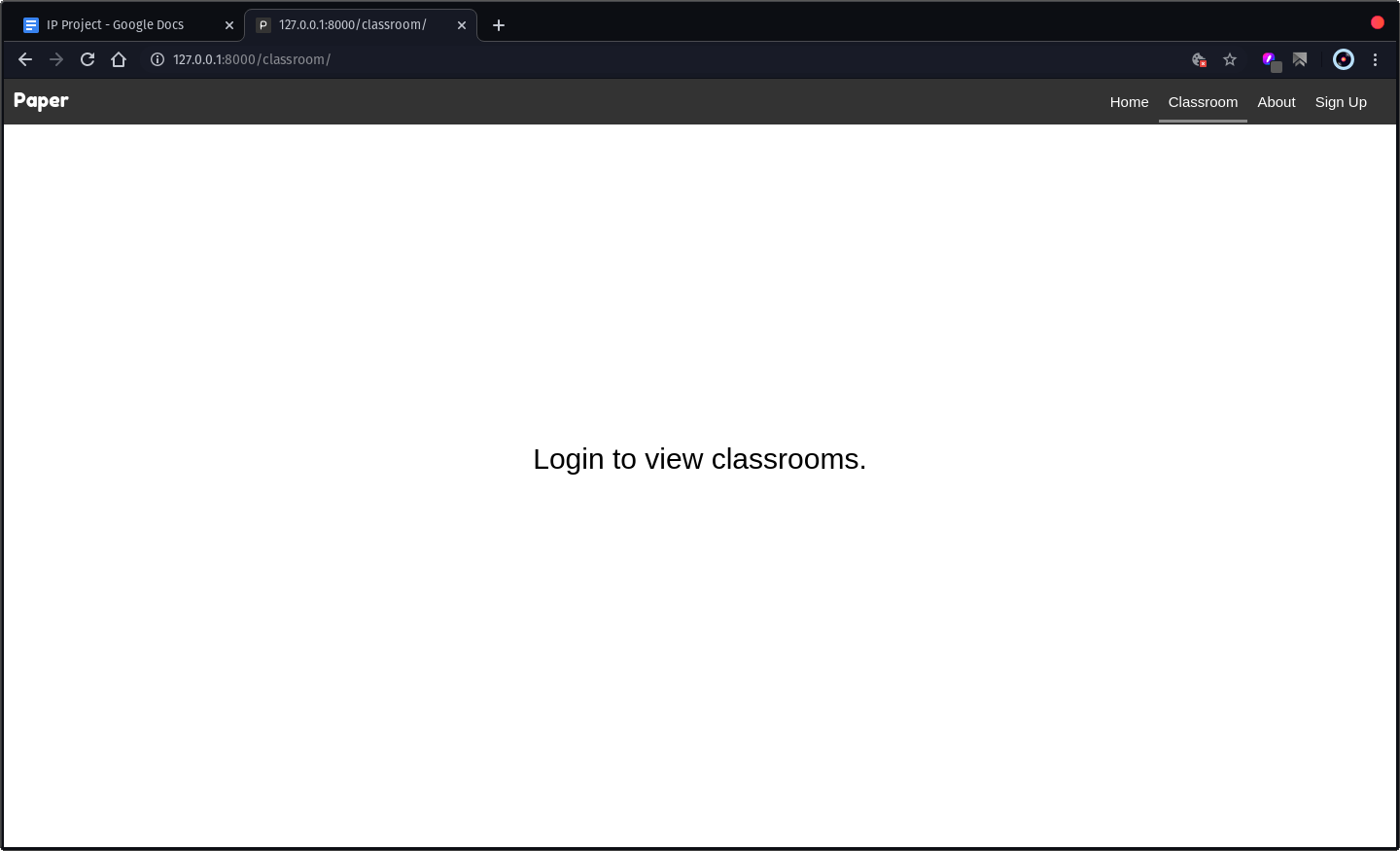
### Logout

|  |
| --- |
| 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 .



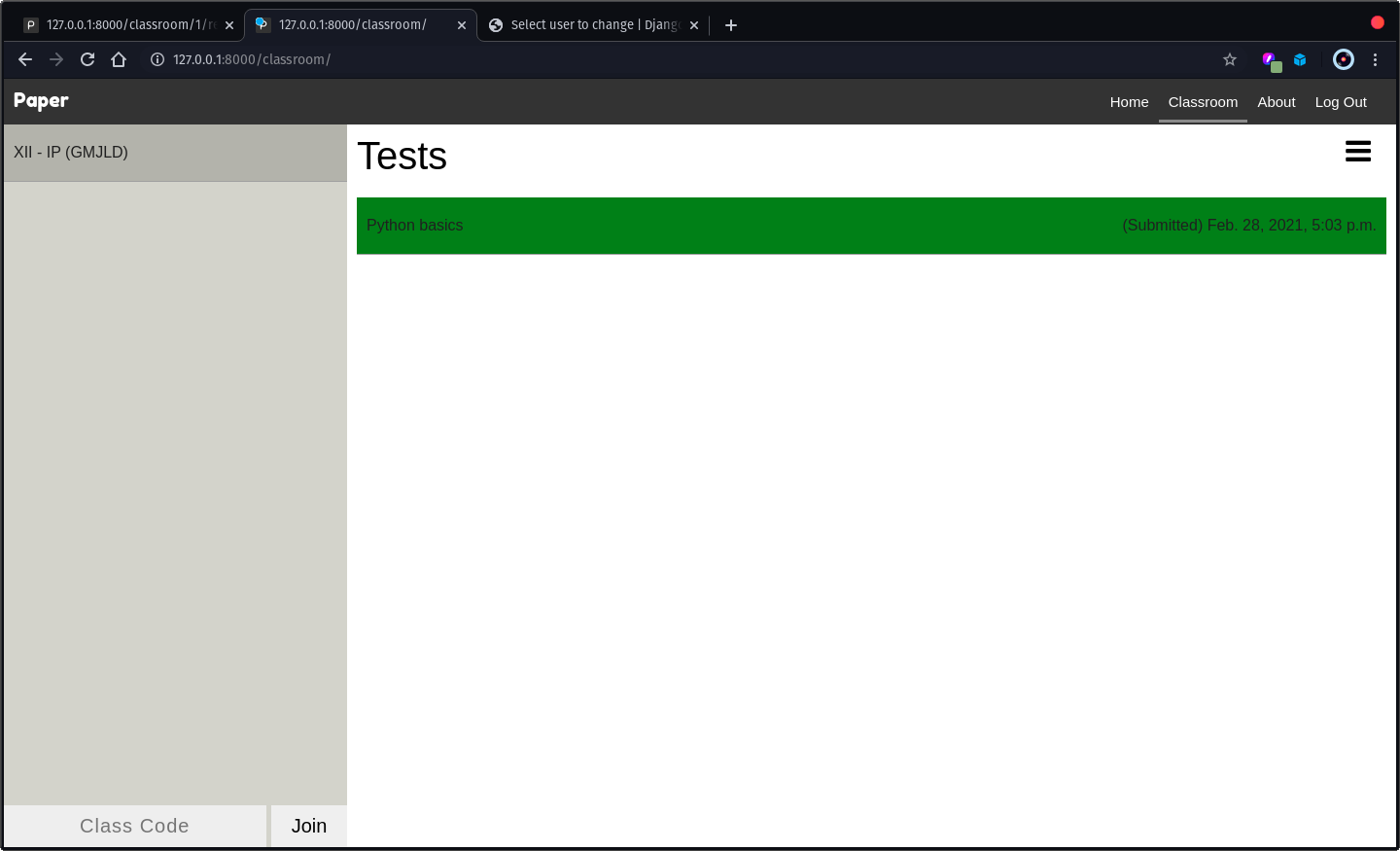
## Teacher

|  |
| --- |
| 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,'userAuthenticated':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

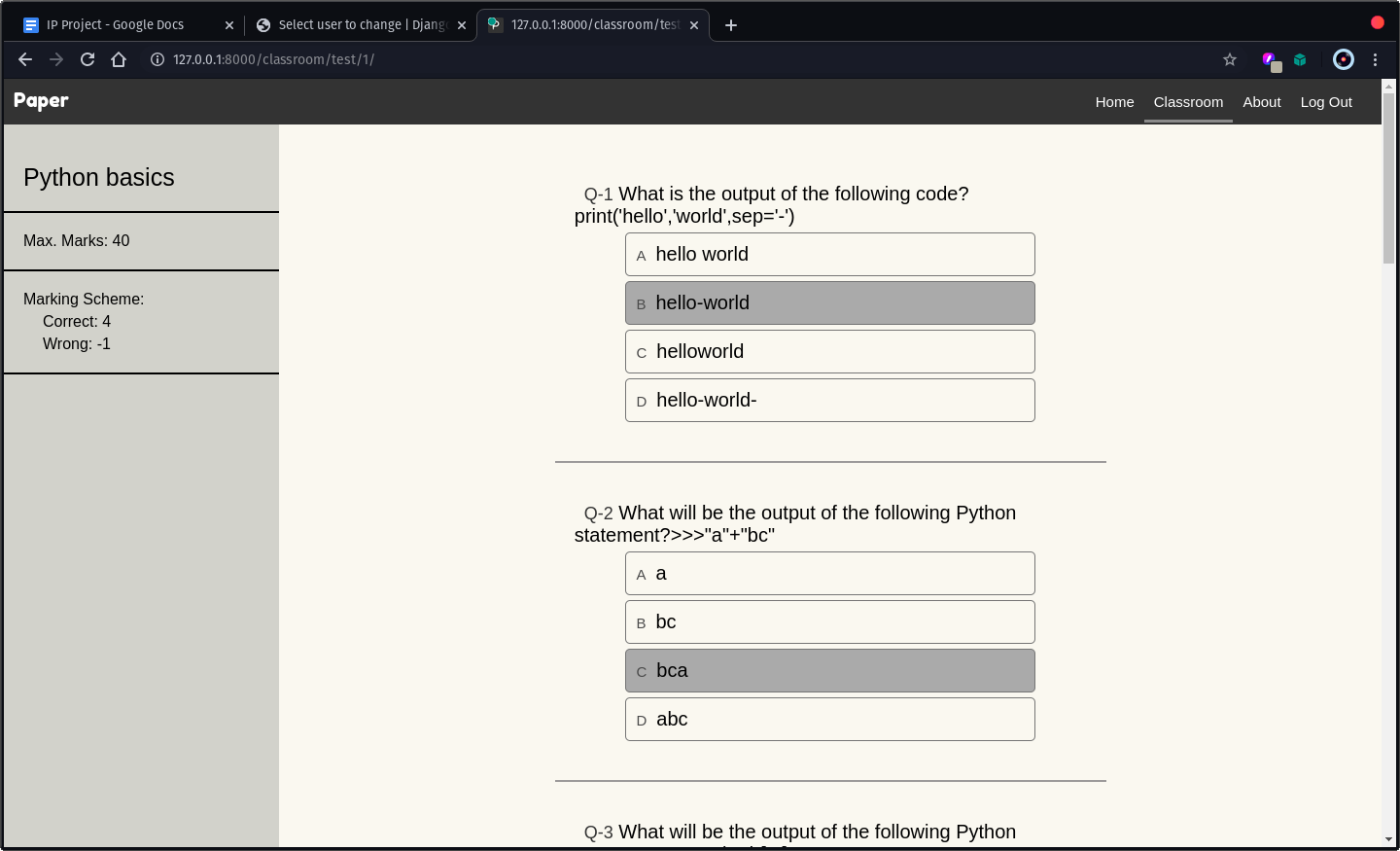
|  |
| --- |
| def student(request,pk\_classroom=0):  currentUser = int(request.user.username)  student = Student.objects.get(id=currentUser)  if pk\_classroom == 0:  try:  className = student.classes.all()  classCode = str(className[0])  tempClass = Classroom.objects.get(class\_code=classCode[0:5])  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,'currentClass':classRoom,'testGiven':testGiven,'noClass':mssg,'hideElement':hideElement,'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.



## 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':answerData1}  form = TestResponseForm(formData)  if form.is\_valid():  form.save()  return HttpResponseRedirect("/classroom/")   return render(request, 'paper/test.html',{'questions':questions,'details':testDetails,'userAuthenticated':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. |



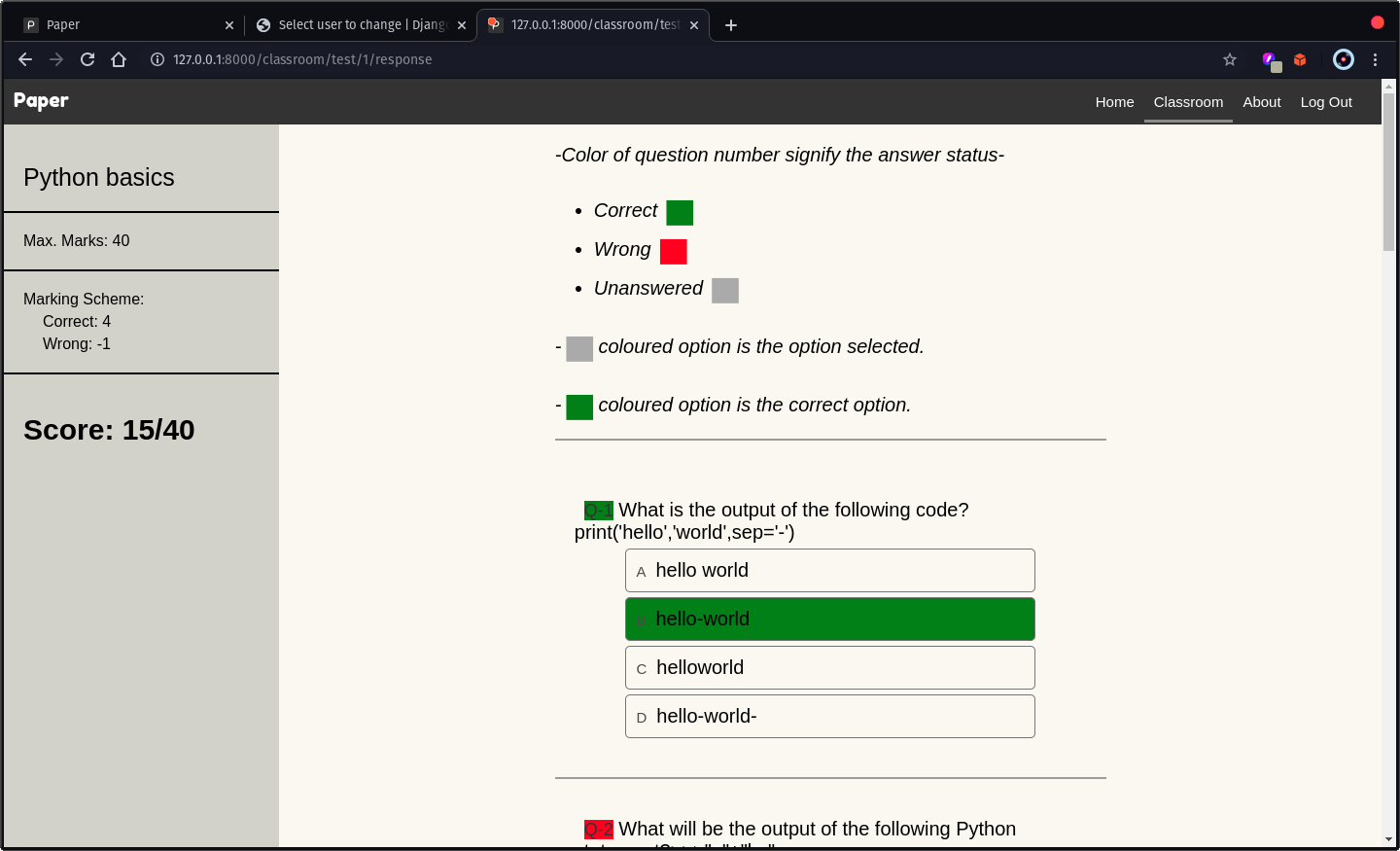
## View test

|  |
| --- |
| 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,'submitBtnClass':'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. |

## 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,'details':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.

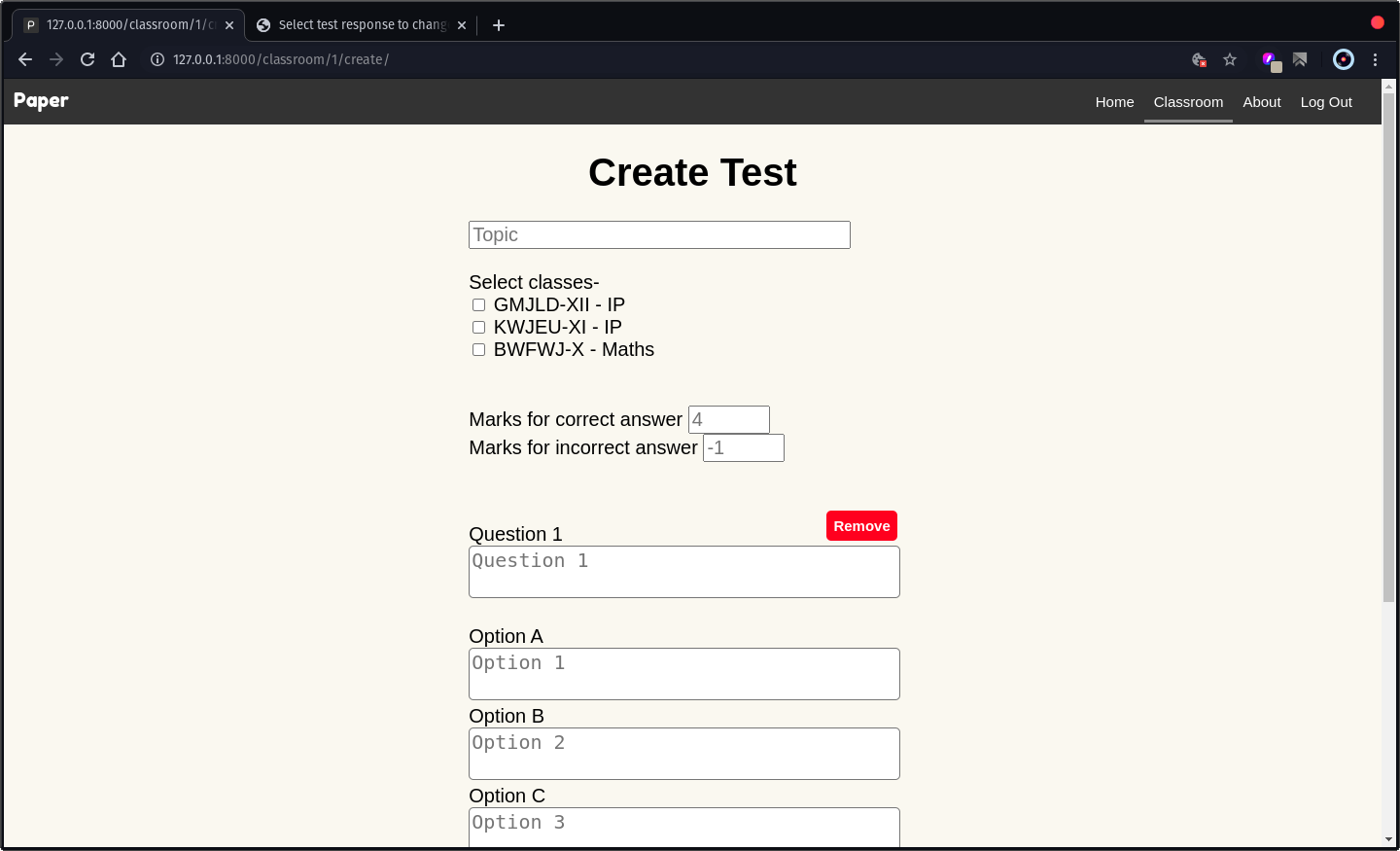


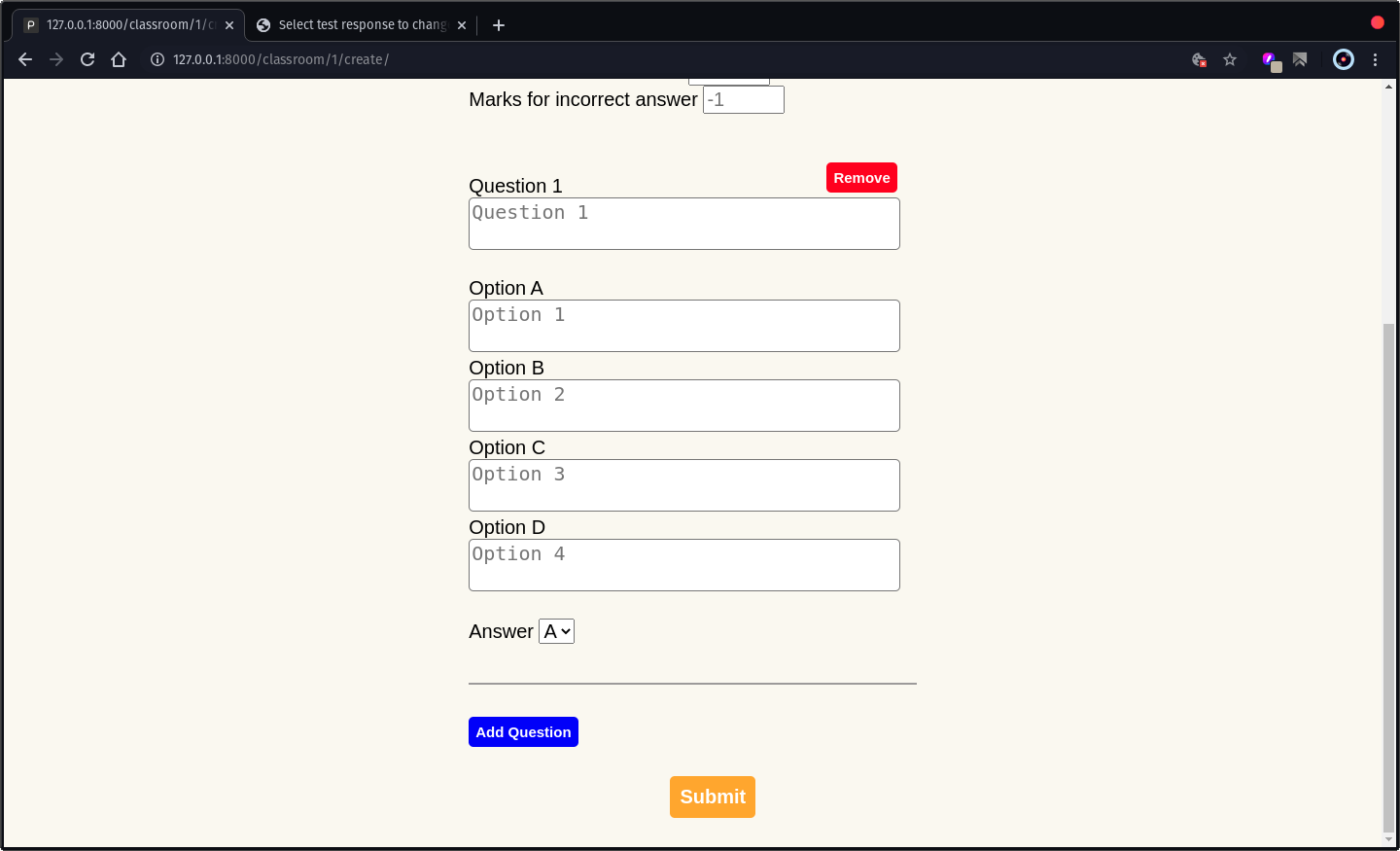
## 

## 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.





## 

## 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('student','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':topic,'currentClass':classRoom,'responseList':responseList,'graphs':graphs,'userAuthenticated':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. |

## 

## 

## 

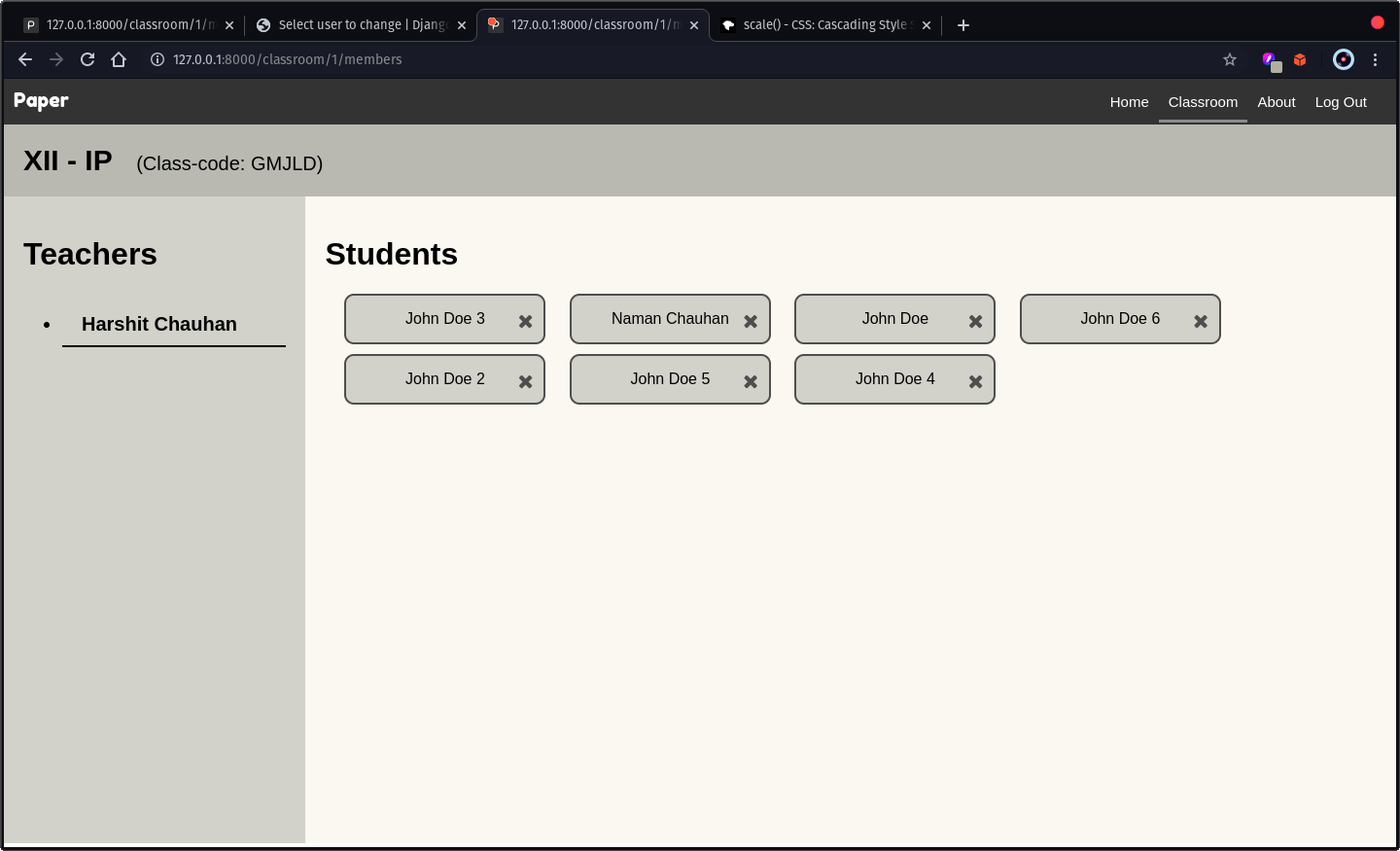
## Export

|  |
| --- |
| 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('student','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,'userAuthenticated':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}) |

# 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](https://www.youtube.com/playlist?list=PL-51WBLyFTg2vW-_6XBoUpE7vpmoR3ztO)