COLLEGE TAB

Minor Project-II Report (IT-608) Section A

In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

Submitted by:

Anushree Kumar (0103IT181027)

Ekveera Joshi (0103IT181044)

Manas Khare (0103IT181055)

Group NO.- 2

Guided by:

Dr. Manish Shrivastava

at

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY

KALCHURI NAGAR, RAISEN ROAD, BHOPAL (INDIA) - 462021

SESSION JULY-DECEMBER 2018

DECLARATION

I hereby declare that the project entitled "**College Tab**" submitted for the B.E.(Information Technology) degree is **our/my** original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Signature of the students with date

Place: **(1) ……**

Date: **(2) ……**

(3) ……

CERTIFICATE

This is to certify that the project titled "**College Tab**" is the bona fide work carried out by **Anushree Kumar (0103IT181027) , Ekveera Joshi (0103IT181044) and Manas Khare (0103IT181055)** student/students of B.E.(Information Technology) of Lakshmi Narain College of Technology, Bhopal affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, Madhya Pradesh (India) during the academic year 2018-19, in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering (**Information Technology**) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Signature & Seal of HOD, **Signature of the Guide with Date**

Information Technology

Lakshmi Narain College of Technology, Bhopal

Signature & Seal of Principal

Lakshmi Narain College of Technology, Bhopal

ACKNOWLEDGEMENT

Apart from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I would like to extend my sincere thanks to all of them. I am highly indebted to our guide Dr. Manish Shrivastava for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. I can't say thank you enough for his tremendous support and help.

I would like to express my gratitude towards Dr. Kamlesh Chandrawanshi for their kind co-operation and encouragement which helped me in completion of this project. I felt motivated and encouraged every time I attended his meeting. Without his encouragement and guidance this project would not have materialized. The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. I am grateful for their constant support and help.

I would also like to express my special gratitude and thanks to my friends and family for giving me their constructive advices and valuable suggestions.

ABSTRACT

.

College Tab is web-based application for technical evaluation of a student's assignments, reports and tests (all on the same platform). Online Quiz Examination System not only replace paperwork but also releases the workload of faculty. It fulfils the requirements of the institutes to conduct the exams online. Students can give exam, submit their work without the need of going to any physical destination. They can view the result at the same time. Thus, the purpose of the site is to provide a system that saves the efforts and time of both the faculty and the students.

Along with this the main feature that gives our application an edge over the others is proctoring. This means that when administering a test, individuals must take the test without cheating.

LIST OF FIGURES

TABLE OF CONTENTS

INTRODUCTION

1.1 Project Background

Education system has changed in many ways over the last decade, with the most important change coming in mode of learning and examination. Educational institutions are slowly moving into online teaching and education, which in a way, is changing the perception of learning among students and parents. Students often raise a query about which is better, online or offline exam. Conducting the exam online has many benefits like less operational and administrative cost, no security threats and use of unfair means (cheating) and feasibility to students and organisers. Online exams have also shortened the timeline of examining the answers sheets and formulating the results beside its numerous other benefits.

Online examinations are becoming extremely popular these days with many universities, institutes and competitive examinations body switching from the Pen and Paper based test. Online examination is basically the exam conducted on a computer with no physical question paper. Online exams are tests with Multiple Choice Questions (MCQs) where candidates have to select the right answer among the four available options and mark on the virtual answer sheet.

Thus, not only does our project help conduct proctored online test (thereby preventing cheating) it also handles students reports, assignments and aims to provide an integrated platform which is convenient for both the student and the teacher.

1.2 Problem Statement

So, during this pandemic we have pretty much completely switched from online to offline mode. And when it comes to the submission of assignments or conduction of exams both the teachers and the students face a lot of issues. Firstly, we are often given the assignments on WhatsApp and are then requested to send it via mail (this ends up leaving the entire process a little disorganised). Along with that we have also faced trouble during our mid semester

examinations and it had to be conducted using google forms. Thus, we aim to create a robust and efficient alternative where all the activities could simultaneously take place.

1.3 Objectives

- The operational, administrative and logistic costs in less in online exams in comparison to offline exams.
- Safety and security of the question paper is not compromised. It is next to impossible to leak online question papers.
- Feasibility for students in attempting the exam.
- Takes less time in checking the answer sheets and preparation of result.
- Question and students' answers can be saved and stored for a long time in online exam.
- Assignments can be uploaded and submitted at any time and would provide flexibility. It also allows students to receive immediate grades of their assignments.

1.4 Project Overview / Specifications

The application launches to display the home screen followed by the login/signup page for both teachers and students.
After the student logs in:
The user then gets to see the dashboard which consists of multiple cards with the name of each subject along with the subject code, the student can click the card to view the assignments provided for each subject. Below this the student also gets to see cards using which the student can attempt examination of the respective subject.
After the teacher logs in:

The user then gets to see the dashboard which consists of multiple cards with the name of each class assigned to the teacher along with the assigned subject along with the subject code, the student can click the card to view the assignments submitted by the students. Along with this the teachers also gets to create new tests.

1.5 Software Specifications

1.5.1 Back End

We have used Django for the back end. Django is an open-source web application frame work which is written in Python. It proves really helpful to build the complex database-driven websites such as ours.

1.5.2 Front End

Front end framework is written in HTML, CSS and Bootstrap that makes designing a user interface easily and also ends up being user friendly. HTML is used for structure and presenting content on the world wide web. An HTML document can provide information for browsers such as what style to use and where to get it. CSS is a stylesheet language that enhances the HTML document. Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development.

LITERATURE SURVEY

2.1 Existing Systems

1. A system that already exists and is currently used by our college as well is LMS. Lurningo is the learning platform of Schoolguru Eduserve, used by the leading Universities across the world to help their students learn online. The platform also offers several employment-linked skill development programs that the students can pursue alongside their degree and diploma programs. But a lot of students find it to be complex and fail to properly understand it. Along with that the technology also failed to administer the mid semester examinations due to heavy traffic.

   Solution - College Tab therefore aims to provide a robust alternative that won't fail under the circumstances of heavy traffic.

2. Currently we receive most of the assignments on WhatsApp and are expected to submit it via email. For this we have a system known as Accsoft (Assignments can be both uploaded and submitted here). But here the issue that persists is that Accsoft also handles students' academic reports, attendance, fees etc. Therefore, college tries to keep minimum unnecessary data (thereby using LMS).

   Solution – Thus College Tab will be the optimum alternative for submitting all assignments, reports and conduction of exams.

2.2 Proposed System

The purpose of College Tab is to create a complete platform where all the activities such as uploading assignments, submitting reports and taking online tests in an efficient manner can take place at once (thereby reducing manual work). The main objective of our project is to create a web application that efficiently evaluates the candidate thoroughly through a fully automated system that not only saves lot of time but also gives fast results.

Scope:

- This can be used in educational institutions as well as in corporate world.

- Can be used anywhere any time as it is a web-based application (user Location doesn't matter).
- No restriction that examiner has to be present when the candidate takes the test.

2.3 Feasibility Study

The prime focus of the feasibility is evaluating the practicality of the proposed system keeping in mind several factors. The following factors are considered before deciding in favour of the new system.

2.3.1 Economic Feasibility

Store organisation process in the proposed system is precise that is the organisation results are generated as per user requirements, which reduces the manual labour. Also, as Retailer's Ally is an android application and none of the software that were used were paid, it is absolutely free.

2.3.2 Technical feasibility

Keeping in view the above fact, nowadays all organizations are automating the repetitive and monotonous works done by humans. The key process areas of the current system are nicely amenable to automation and hence the technical feasibility is proved beyond doubt.

2.3.3 Operational Feasibility

The present system has automated most of the manual tasks. Therefore, the proposed system will increase the operational efficiency of the retailers and other staff in the store.

SYSTEM ANALYSIS AND DESIGN

3.1 Requirement Specifications

3.1.1 Functional Requirements

- It should allow the teacher to work on any new test or assignment and upload it at any time.
- It should also provide the students flexibility to complete and submit their assignments at any time before the deadline according to their convenience.
- Secure registration and profile management facilities for different users.
- It should provide the teachers a list of all the students who have submitted the assignment and test so that they can be graded quickly.
- It should also prevent students cheating during the examination (proctored examination).

3.1.2 Non- Functional Requirements

a) Safety Requirements

If there is extensive damage to a wide portion of the database due to catastrophic failure, such as a disk crash, the recovery method restores a past copy of the database that was backed up to archival storage (typically tape) and reconstructs a more current state by reapplying or redoing the operations of committed transactions from the backed-up log, up to the time of failure.

b) Security Requirements

Security systems need database storage just like many other applications. However, the special requirements of the security market mean that vendors must choose their database partner carefully.

c) Software Quality Attributes

- **AVAILABILITY:** Since we are hosting our project on the server it will be available all the time.

- **CORRECTNESS:** The application should generate an appropriate organisation structure for different items in the store.
- **MAINTAINABILITY:** The application should maintain itself without much human interaction because the machine learning algorithms keep training themselves according to the latest shopping trends.
- **USABILITY:** The system should satisfy the maximum number of user's needs.

3.1.3 Software Requirements

To build such a complicated web system, we need three major parts for each component: database, user interface and the functions to interact in between. Django framework provides sufficient functionalities to implement these three parts. Corresponding to database, user interface and functions in between, Django has model, template and view components to deal with each part respectively. Django's model component helps programmer to define and maintain tables in the database, while its template component helps to write html files using a combination of both html syntax and Django syntax. For those functions in between, Django provides a view component which reads the input from user interface and makes corresponding changes in the database.

- Django framework

  Django is an open-source web application frame work written in Python. The primary goal of Django is to make the development of complex, data-based websites easier. Thus, Django emphasizes the reusability and pluggability of components to ensure rapid developments. Django consists of three major parts: model, view and template.

- Model

  Model is a single, definitive data source which contains the essential field and behaviour of the data. Usually, one model is one table in the database. Each attribute

in the model represents a field of a table in the database. Django provides a set of automatically-generated database application programming interfaces (APIs) for the convenience of users.

- View

  View is short form of view file. It is a file containing Python function which takes web requests and returns web responses. A response can be HTML content or XML documents or a “404 error” and so on. The logic inside the view function can be arbitrary as long as it returns the desired response. To link the view function with a particular URL we need to use a structure called URLconf which maps URLs to view functions.

- Template

  Django’s template is a simple text file which can generate a text-based format like HTML and XML. The template contains variables and tags. Variables will be replaced by the result when the template is evaluated. Tags control the logic of the template. We also can modify the variables by using filters. For example, a lowercase filter can convert the variable from uppercase into lowercase.

- Python

  Python is the language used to build the Django framework. It is a dynamic scripting language similar to Perl and Ruby. The principal author of Python is Guido van Rossum. Python supports dynamic typing and has a garbage collector for automatic memory management. Another important feature of Python is dynamic name solution which binds the names of functions and variables during execution.

3.2 Flowcharts / DFDs / ERDs

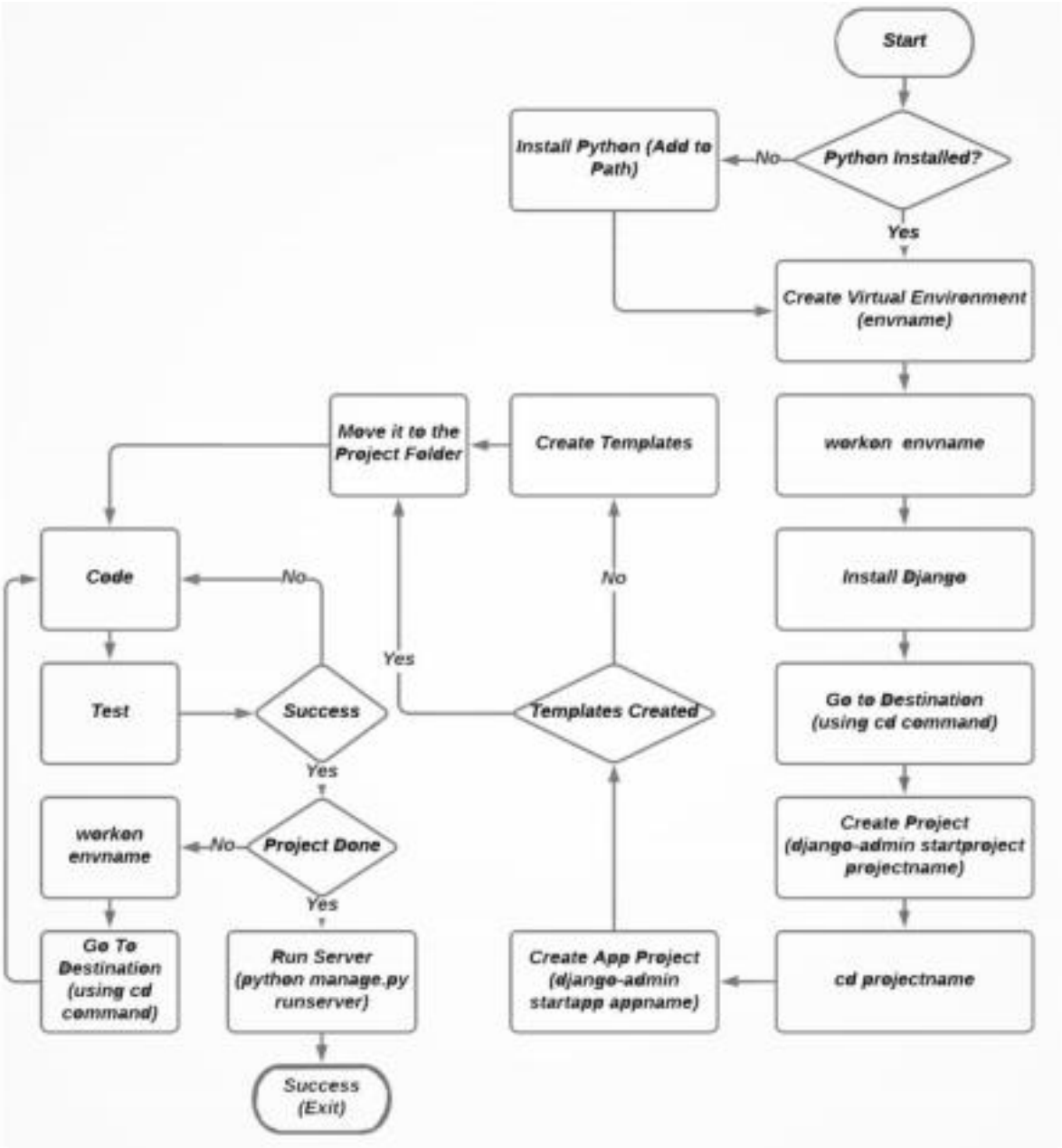
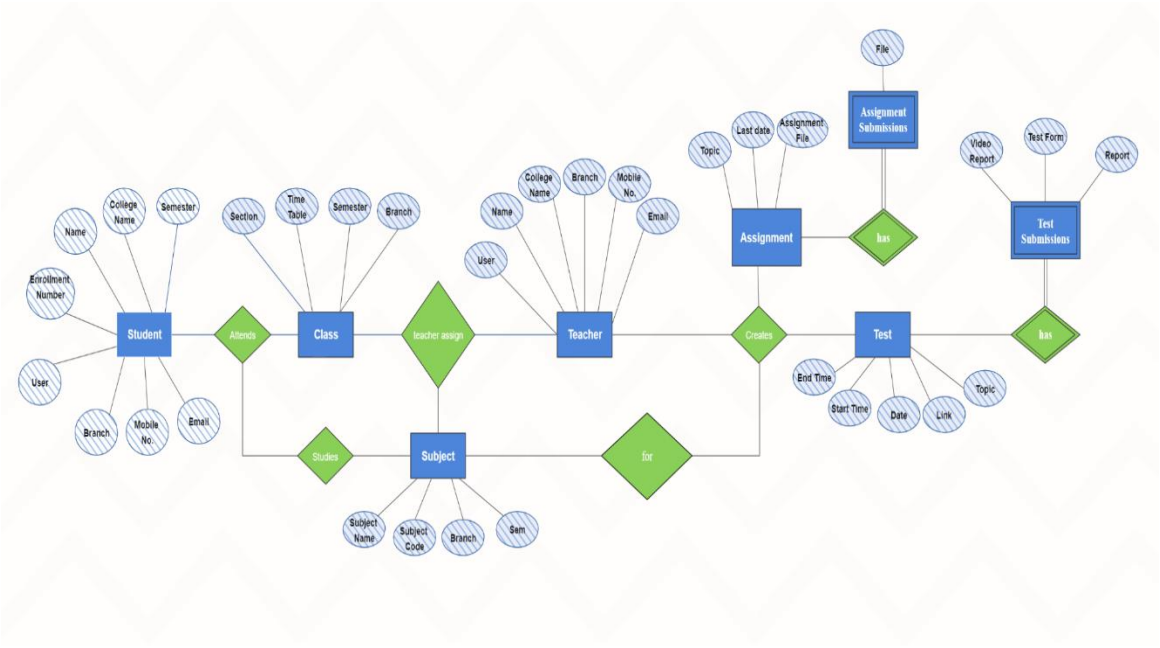


Figure 1



College Tab

ER Diagram

Figure 2

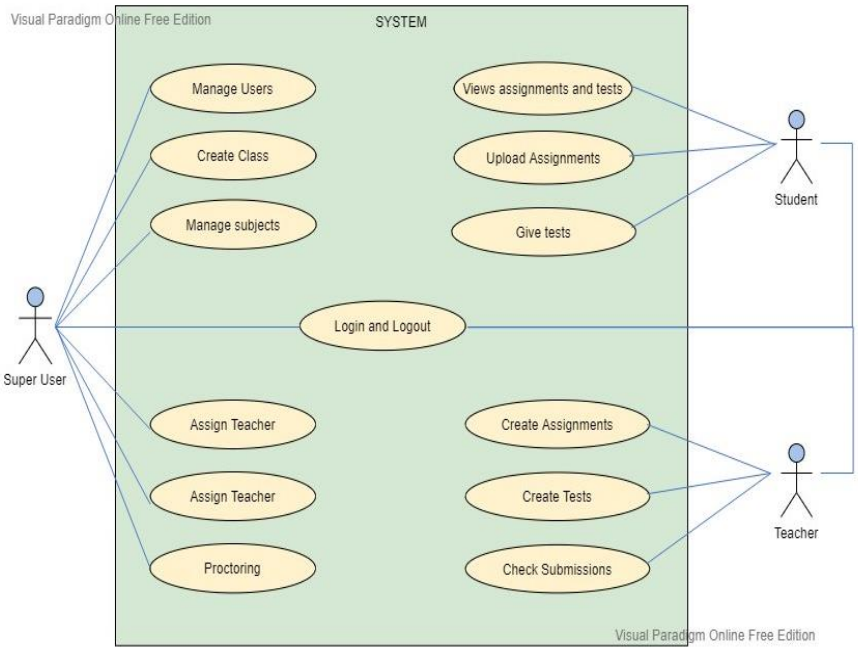


Figure 3

3.3 Design Steps

3.3.3 Database models

Given below are the design of the database tables we have created for our project

```
class student(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,unique=True)
    enrollment_number = models.CharField(max_length = 12,primary_key =
True)
    name = models.CharField(max_length = 30)
    college_name = models.CharField(max_length = 10,choices =
CLG_CHOICES,default = 'LNCT')
    sem = models.CharField(max_length = 20,choices =
SEMESTER_CHOICES,default = '6')
    sec = models.CharField(max_length = 1,choices = SEC_CHOICES,default =
'A')
    branch= models.CharField(max_length = 20,choices =
BRANCH_CHOICES,default = 'IT')
    mobile_no = models.CharField(max_length=10)
    email = models.CharField(max_length=30)
```

```
class teacher(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,unique=True)
    name = models.CharField(max_length=30)
    college_name = models.CharField(max_length = 10,choices =
CLG_CHOICES,default = 'LNCT')
    branch = models.CharField(max_length=20, choices=BRANCH_CHOICES,
default='IT')
    mobile_no = models.CharField(max_length=10)
    email = models.CharField(max_length=30)

    def __str__(self):
        return  self.name


class subject(models.Model):
    subject_code = models.CharField(max_length = 8,primary_key = True)
    name = models.CharField(max_length = 30)
    branch = models.CharField(max_length=20, choices=BRANCH_CHOICES,
default='IT')
    sem = models.CharField(max_length = 20,choices =
SEMESTER_CHOICES,default = '6')

    def __str__(self):
        return str(self.subject_code)+" "+ str(self.name)
```

3.4 Login/signup

3.4.3 Signup

```
def teacher_signup(request):
    if request.method == 'POST':
        uform = UserForm(request.POST)
        form = TeacherForm(request.POST)
        if uform.is_valid():
            user = uform.save()
            if form.is_valid():
                new_teacher = form.save(commit=False)
                new_teacher.user = user
                new_teacher.save()
                print(new_teacher)
    form = TeacherForm()
    uform = UserForm()
    context = {'form': form,'uform':uform}
    template = loader.get_template('student signup.html')
    return HttpResponse(template.render(context, request))
```

3.4.4 Login

```
#teacher views
def teacher_login(request):
    if request.method=="POST":
        username = request.POST["username"]
        password = request.POST["password"]
        user = auth.authenticate(username=username,password=password)
        if user is not None:
            auth.login(request,user)
            tobj = teacher.objects.get(user=user)
            #print(id,username,password)
            #return HttpResponseRedirect('/t_dashboard/%d/' % tobj.id)
            url = reverse('teacher:t_db', kwargs={'id': tobj.id})
            return HttpResponseRedirect(url)
        else:
            messages.info(request,"Invalid Credentials")
            return redirect('/t_login')
    else:
        #form = TeacherForm()
        context = {}
        template = loader.get_template('teacher login.html')
        return HttpResponse(template.render(context, request))
```

3.5 Web Socket

We have used tornado framework which is used for creating persistent web sockets. When the test is active the web socket will we triggered on. From the front end of the

user we will encode the video captured using web cam using javascript to this web socket along with the details of the student. During the entire test duration this will be repeated and web socket will keep on storing the frames when test is over front end will disconnect the connection now these frames will be decoded back into video frames using base64 decoder after which using open cv these frames will be converted To video and stored into the database along with the details of the student.

```
import logging
import base64
import os
import tornado.ioloop
import tornado.options
import tornado.web
import tornado.websocket
from tornado.options import define, options
from django.core.files import File



define("port", default=8001, help="run on the given port", type=int)

class MainHandler(tornado.websocket.WebSocketHandler):
    count = 0
    def check_origin(self, origin):
        return True

    def open(self):
        print("connected")
        self.frames = []
        logging.info("A client connected.")

    def on_close(self):
        logging.info("A client disconnected")

class Application(tornado.web.Application):
    def __init__(self):
        handlers = [(r"/websocket", MainHandler)]
        settings = dict(debug=True)
        tornado.web.Application.__init__(self, handlers, **settings)


def socket():
    import os
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "ctweb.settings")
    tornado.options.parse_command_line()
    #now we start app
    app = Application()
    app.listen(options.port)
    tornado.ioloop.IOLoop.instance().start()
```

3.6 Algorithm

3.6.3 AI proctor using object detection

When the teacher commands to proctor for a test videos of all students who attempted the test will be fetched one by one and fed to the CNN object detection system.

The system is created using yolo_3 and darknet and can detect a large number of objects but we are using it only to detect objectionable objects like cheating notes mobile phone And also to detect the presence of only one person in the video frame who is attempting the test. If there is any violation is detected it will stored in database and the report along with the suspected video will be sent to the teacher in this way teacher can cross check if there was some cheating or it was just false positive. This feedback will also be stored to increase the efficiency of this system

```
def proctor():
    yolo = YoloV3()
    load_darknet_weights(yolo, 'G:/yolov3.weights')
    details = []
    #cap = cv2.VideoCapture(0)
    cap = cv2.VideoCapture('0103IT181055.avi')
    while (cap.isOpened()):
        ret, image = cap.read()
        if ret == False:
            break
        img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        img = cv2.resize(img, (320, 320))
        img = img.astype(np.float32)
        img = np.expand_dims(img, 0)
        img = img / 255
        class_names = [c.strip() for c in open("classes.txt").readlines()]
        boxes, scores, classes, nums = yolo(img)
        count = 0
        for i in range(nums[0]):
            if int(classes[0][i] == 0):
                count += 1
            if int(classes[0][i] == 67):
                details.append('Mobile Phone detected')
        if count == 0:
            details.append('No person detected')
        elif count > 1:
            details.append('More than one person detected')

        image = draw_outputs(image, (boxes, scores, classes, nums),
class_names)

        cv2.imshow('Prediction', image)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
    print(*details)
```

3.7 Testing Process

The reason behind testing was to find errors. We executed the programs with the intention of finding the errors in it. Generally testing is restricted to being performed after the development phase is complete, but we carried it parallelly with all stages of application development, starting with requirement specification.

7.1 Unit Testing

Unit testing was done after the coding phase. The purpose of the unit testing was to locate errors in the current module, independent of the other modules. Some changes in the coding were done during the testing phase. Finally, all the modules were individually tested following bottom to top approach, starting with the smallest and lowest modules and then testing one at a time.

7.2 Black Box Testing

This method of software testing tests the functionality of an application as opposed to its internal structures or working(i.e. white box testing). Specific knowledge of the application's code/internal structure and programming knowledge, in general, is not required. Test cases are built to specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and design to derive test cases. These tests can be functional or non-functional, though usually functional. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure.

7.3 White Box Testing

This method of software testing tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing, an internal
perspective of the system, as well as programming skills, are required and used to design test
cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs.

7.4 Integration Testing

Once the unit was over, all the modules were integrated for integration testing. External and internal interfaces are implemented and work as per design, the performance of the module is not degraded.

7.5 Validation Testing

At the culmination of integration testing, the software is said to be completely assembled as a package; interfacing errors have been uncovered and corrected. Then as a final series of software test, validation tests were carried out.

7. 6 Acceptance Testing

This is the final stage in the testing process before the system is accepted for operational use. Any requirement problem or requirement definition problem revealed from acceptance testing are considered and made error free.

OUTPUT / RESULT

This project report thus introduces how to build part of a course management system using the Django framework. Django is an open-source web application frame work which is written in Python. This course management system built using Django has major components each of which has different functionality but similar architecture. In the project report we demonstrated details of using Django to build proctored examination system integrated with assignment submission platform. Also, the technique and process which is showed here can be applied to build this as well as other complex database-driven websites.

4.1 Screenshots

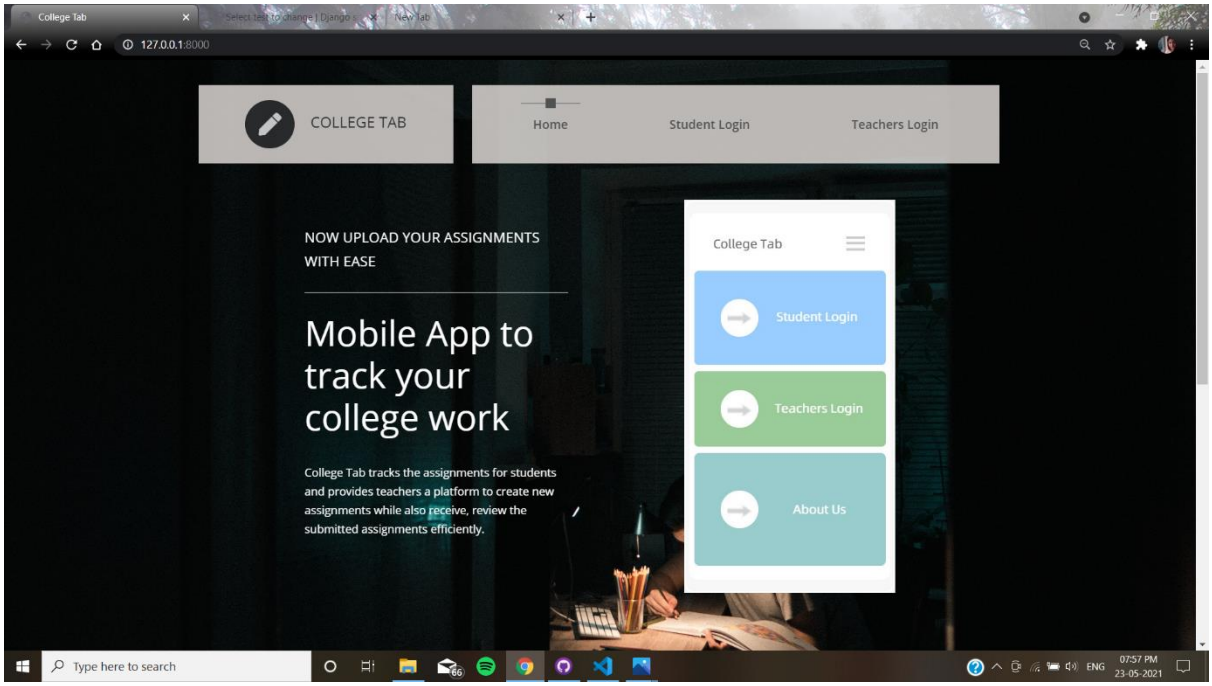


Figure 4

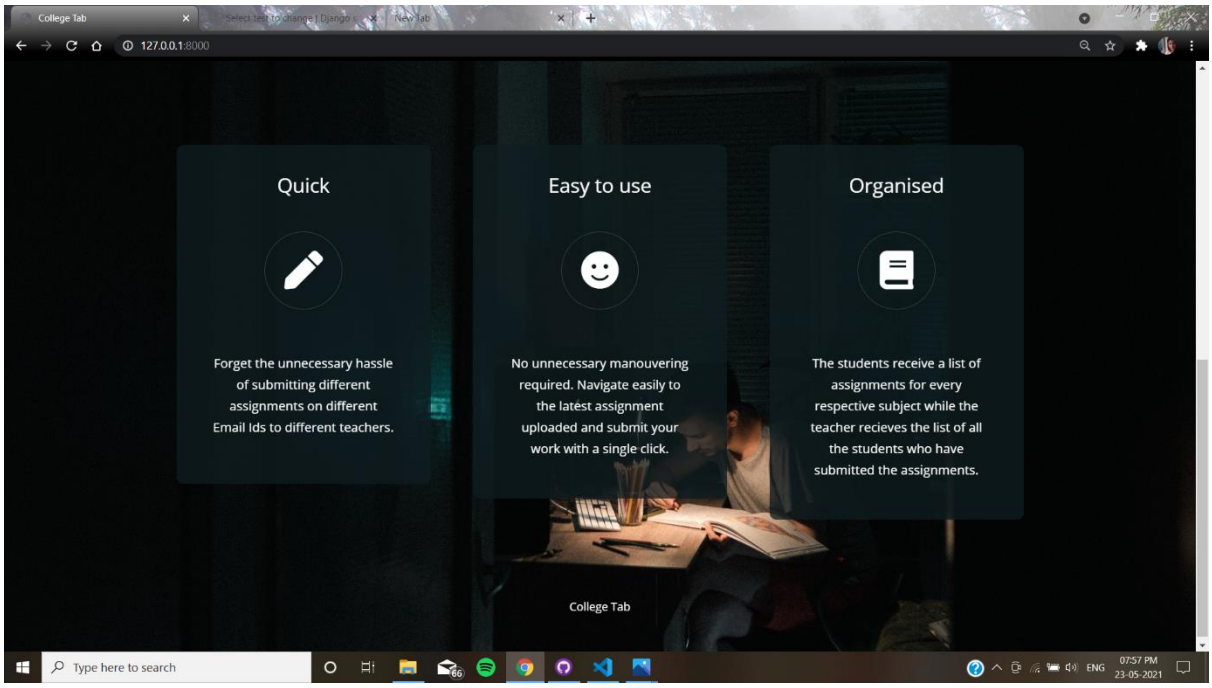


Figure 5

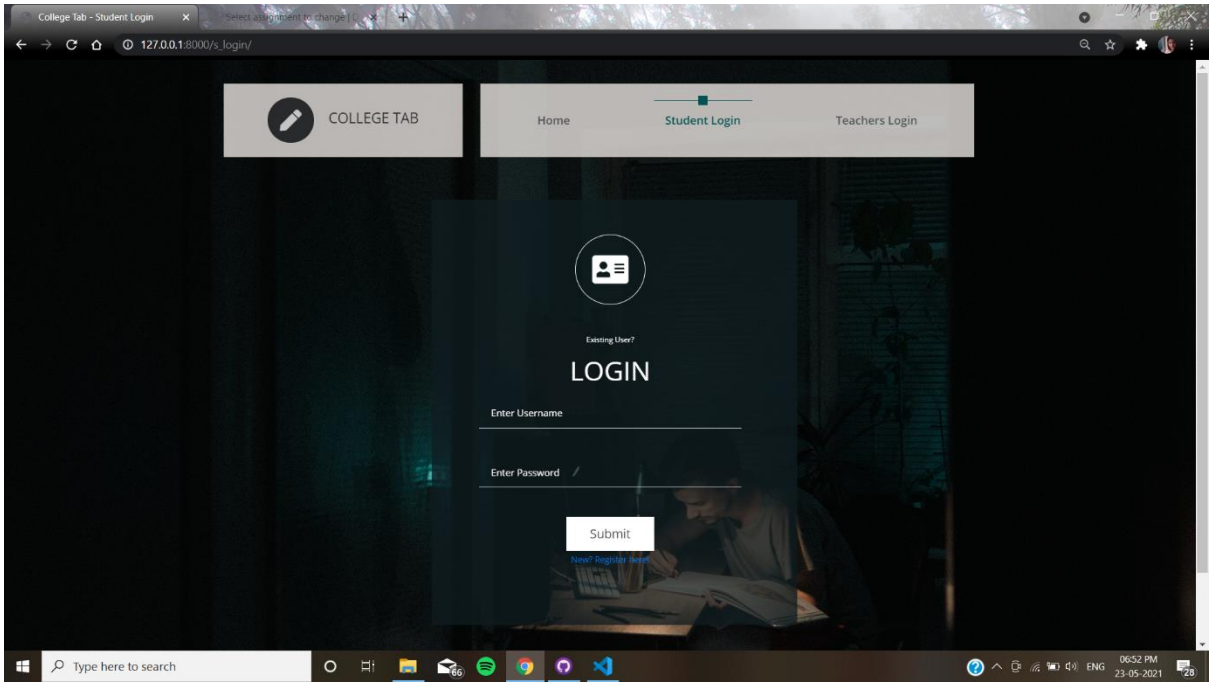


Figure 6

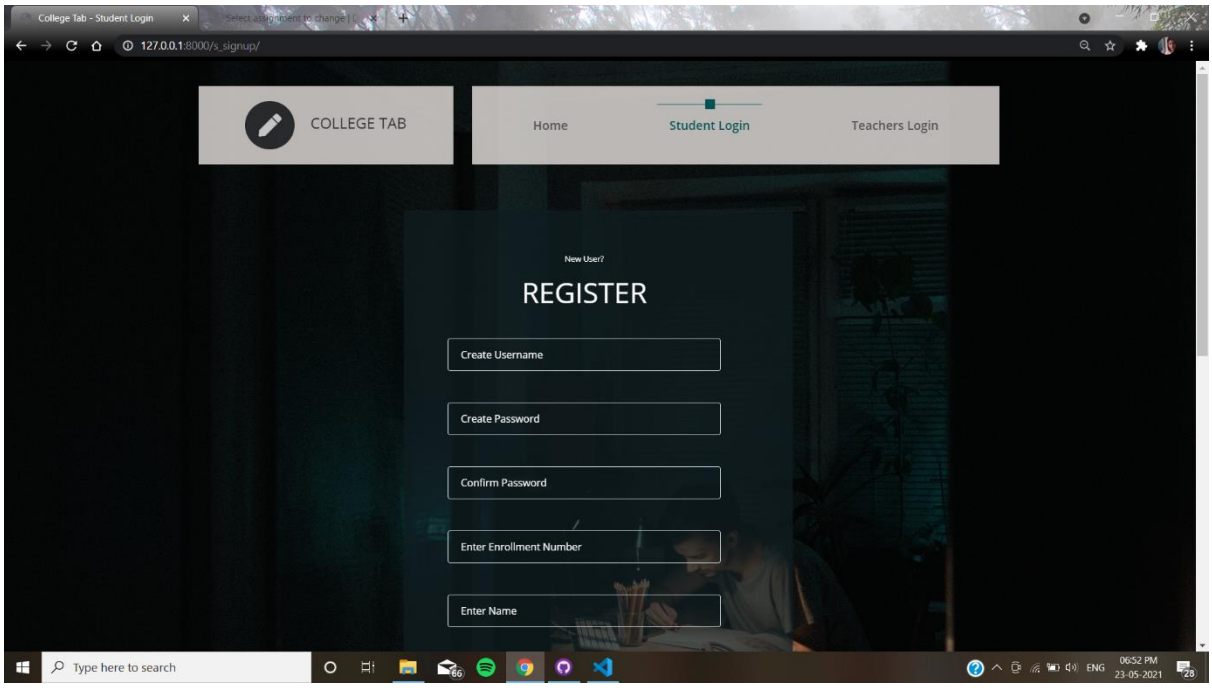


Figure 7

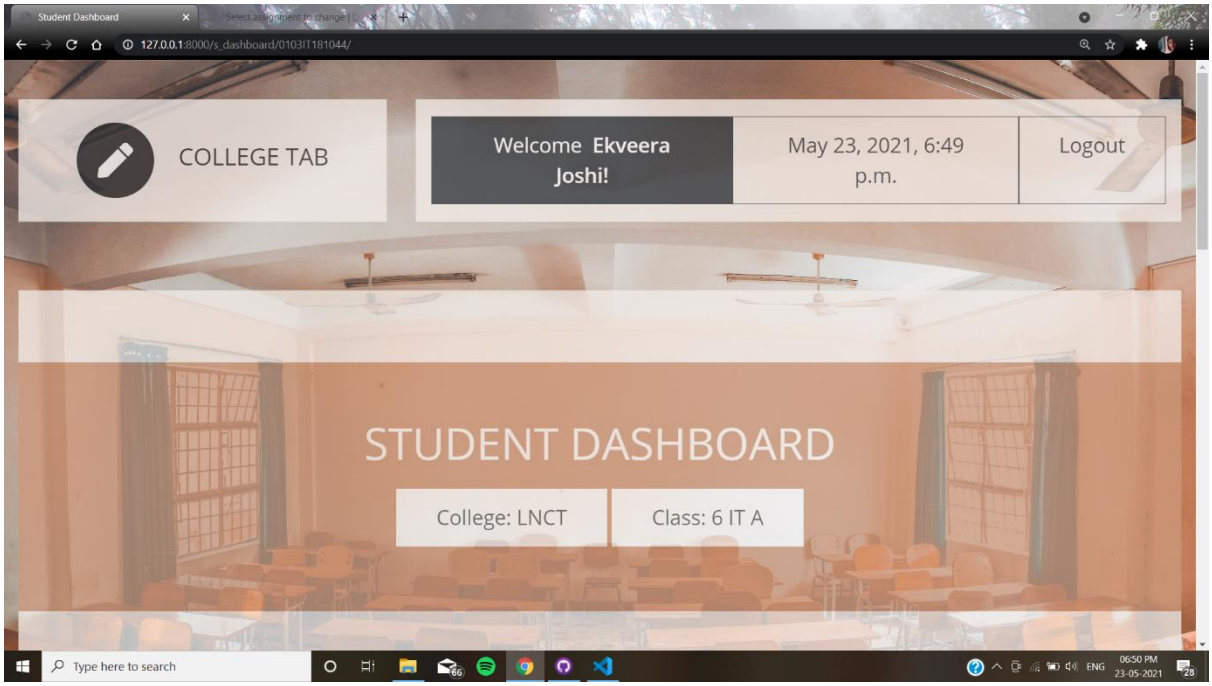


Figure 8

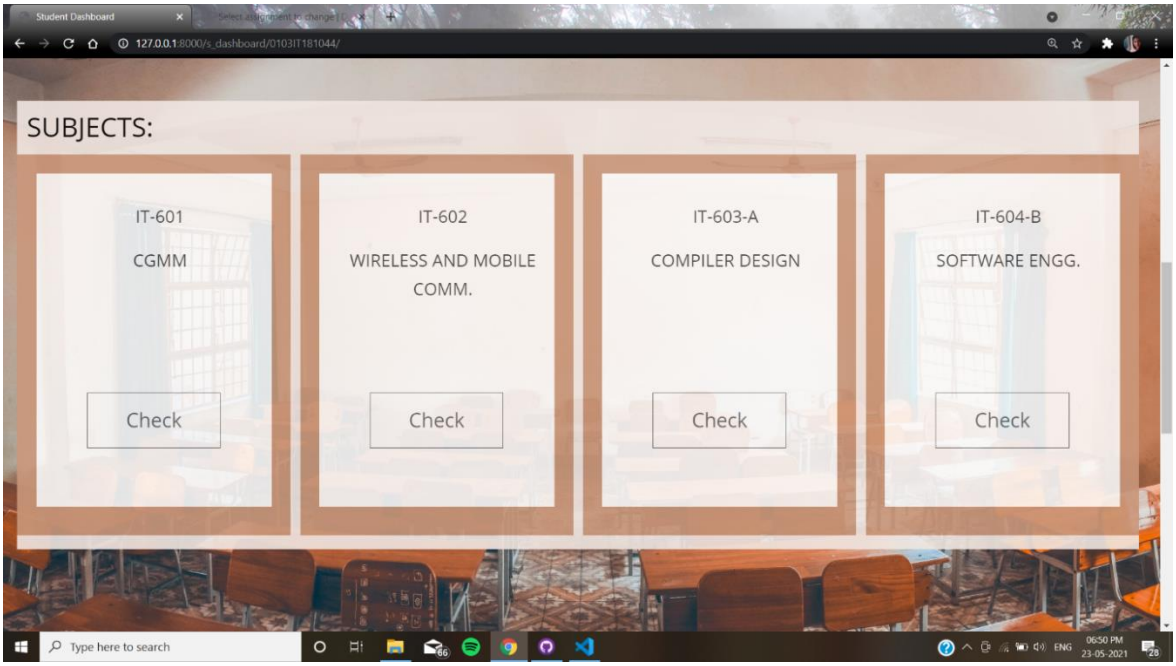


Figure 9

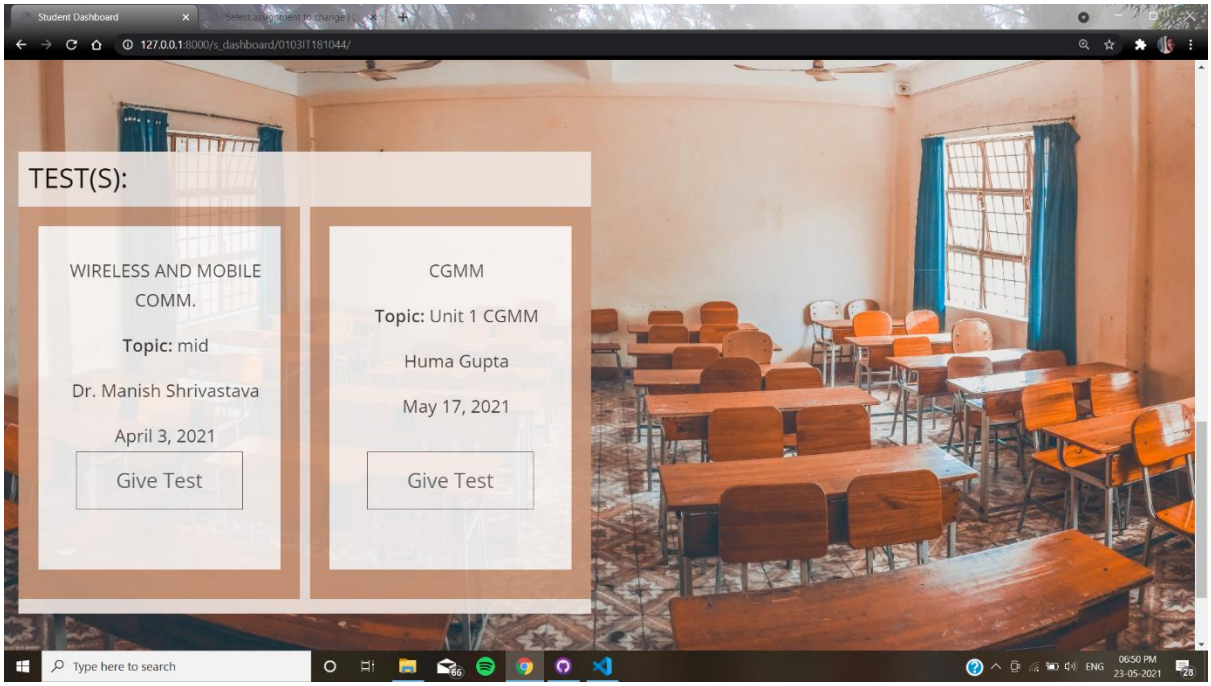


Figure 10

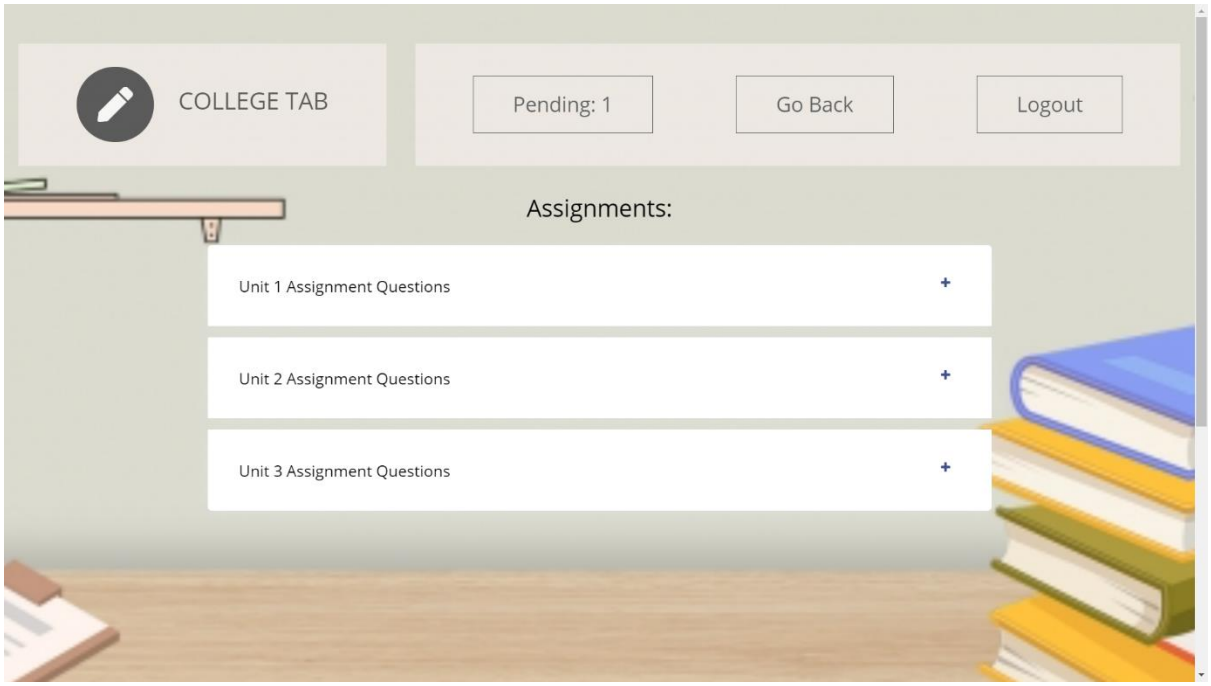


Figure 11

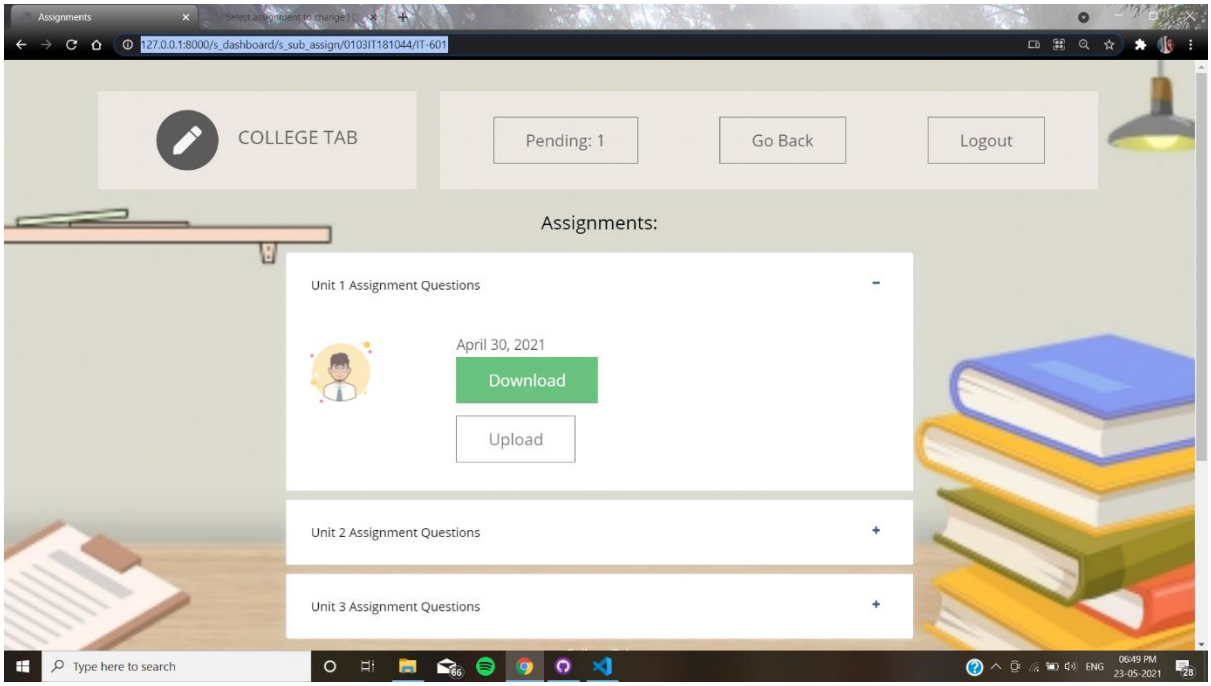


Figure 12

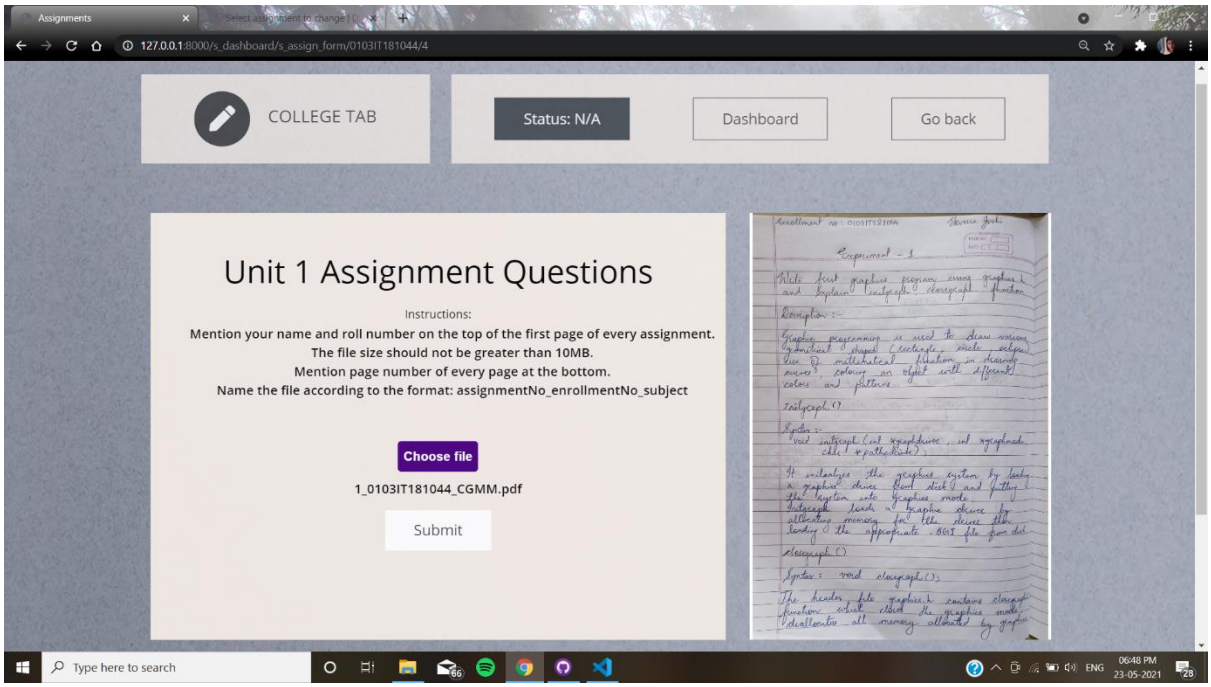


Figure 13

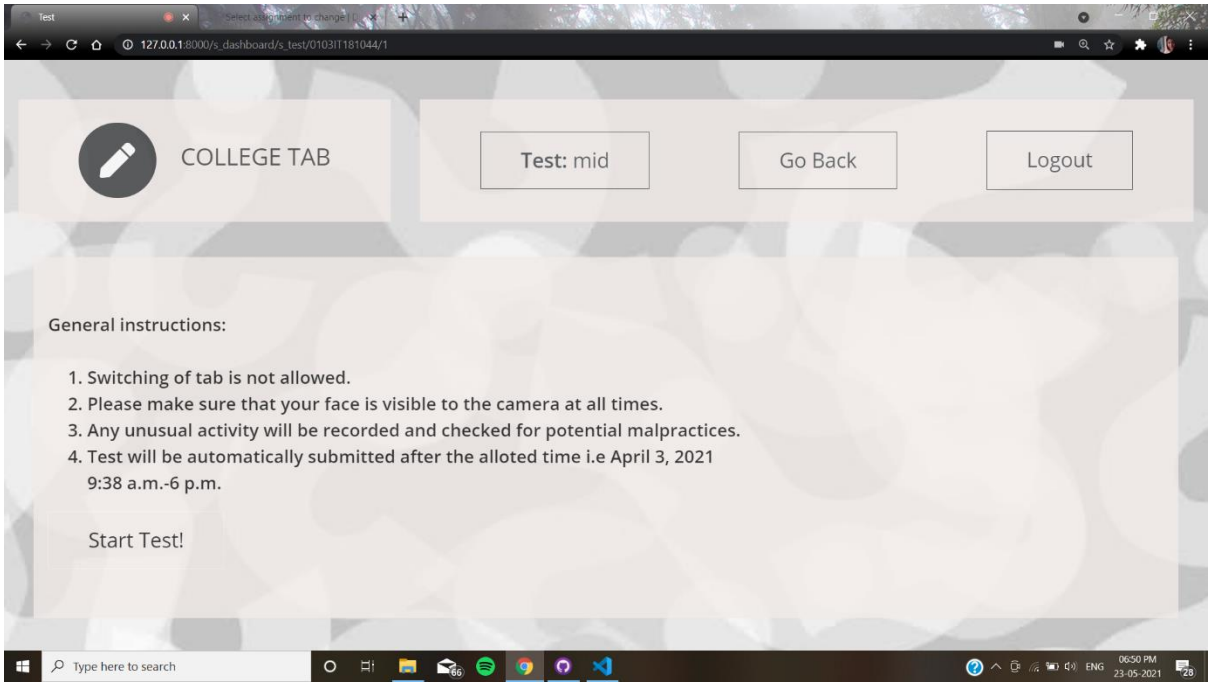


Figure 14

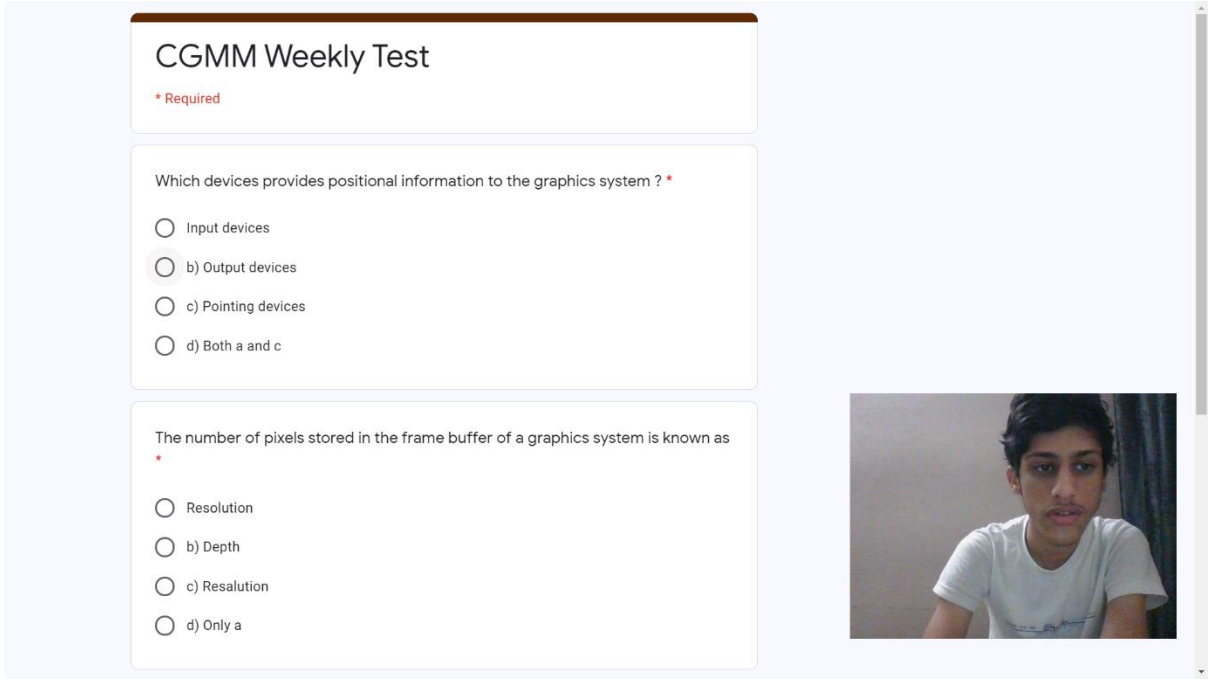


Figure 15

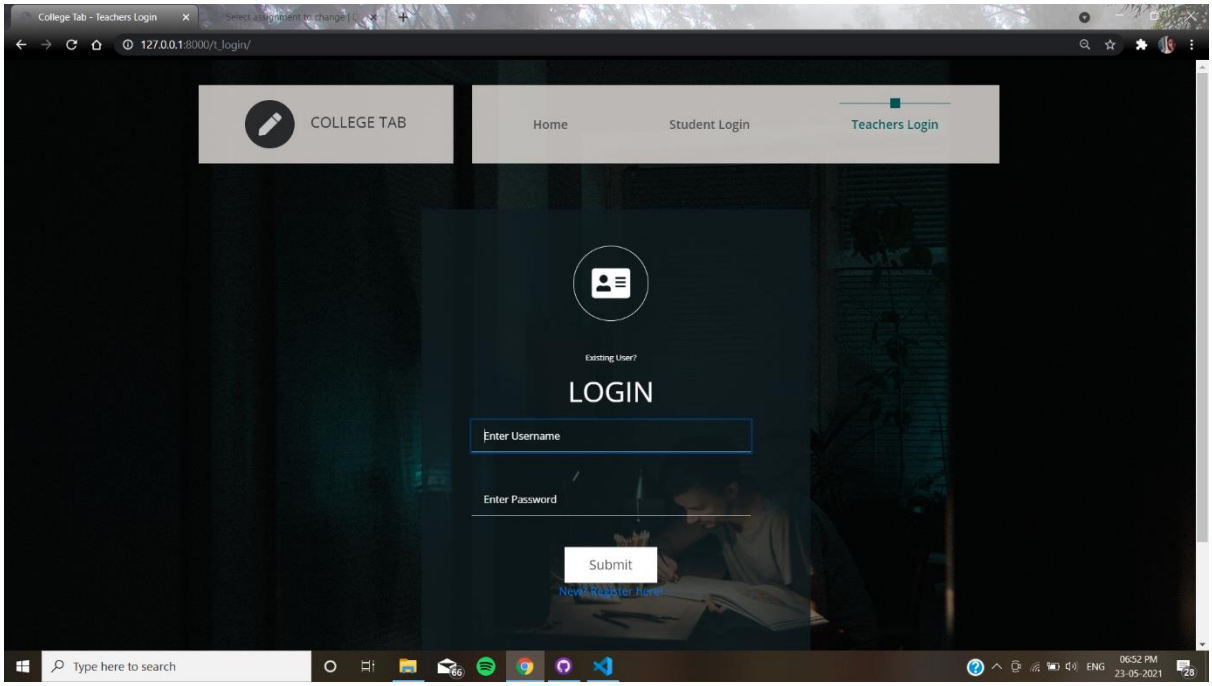


Figure 16

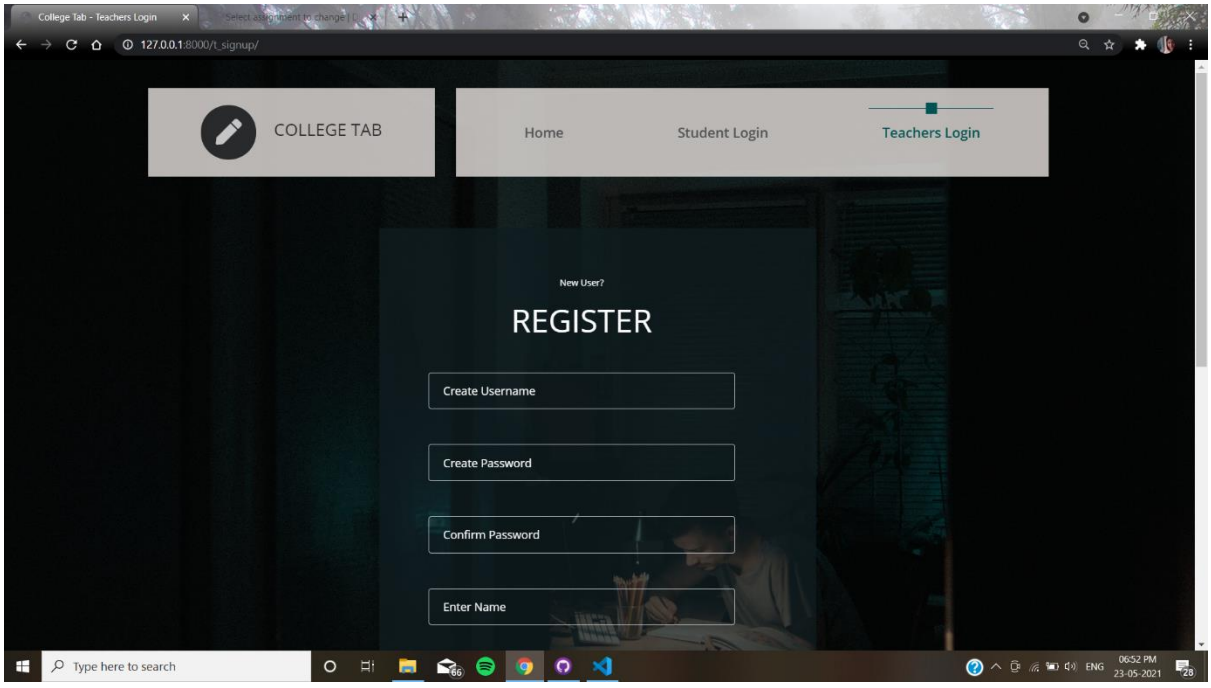


Figure 17

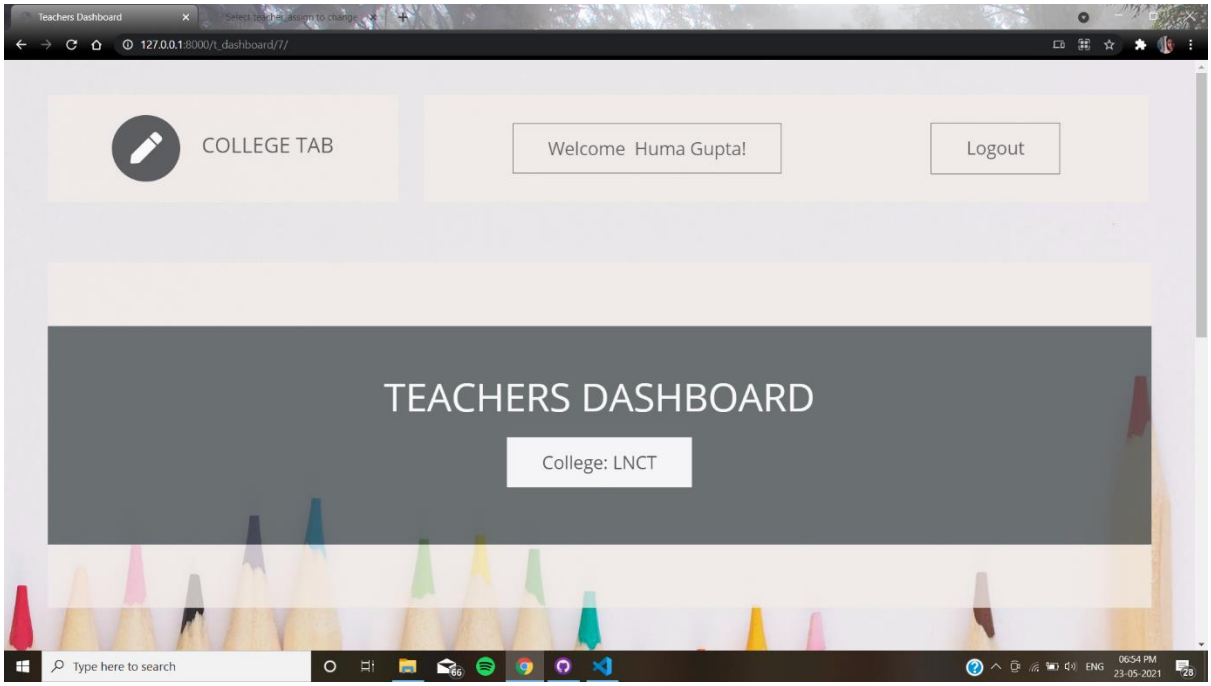


Figure 18

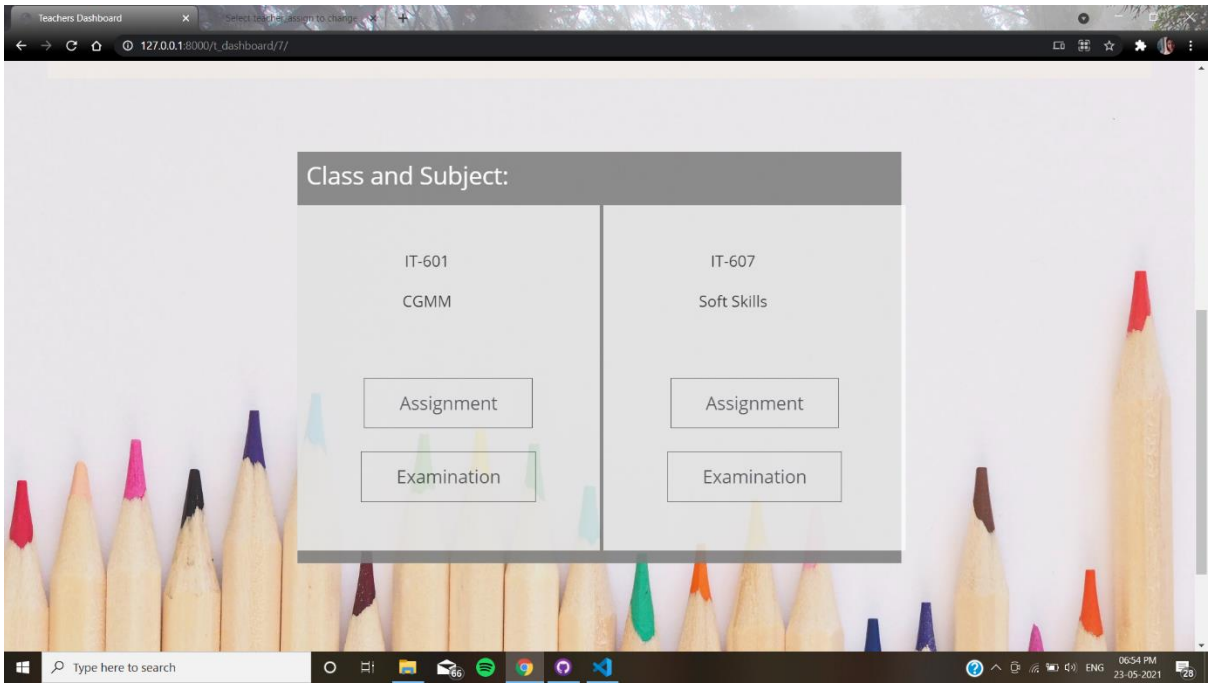


Figure 19

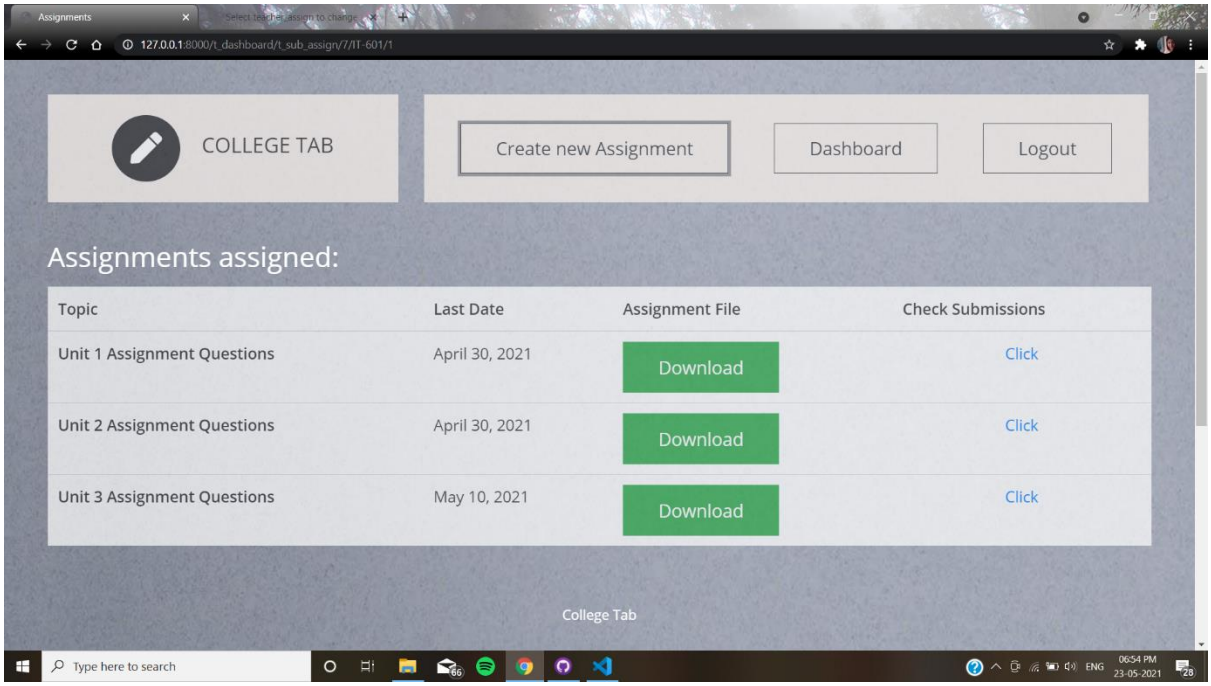


Figure 20

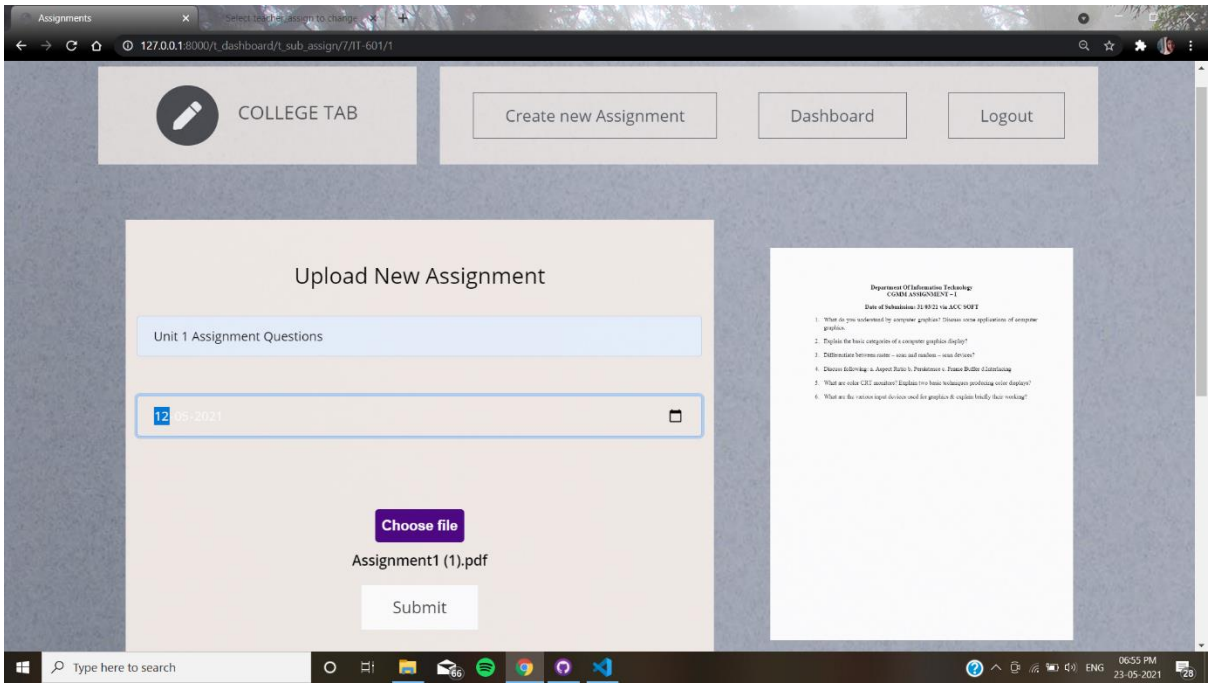


Figure 21

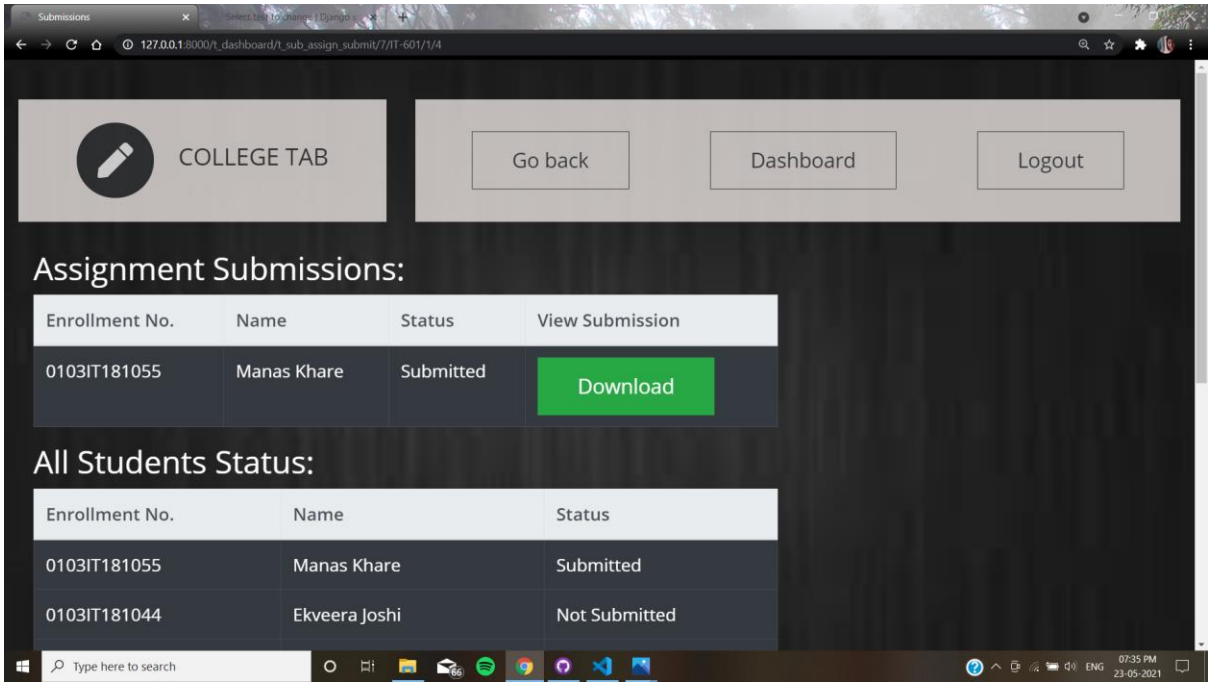


Figure 22

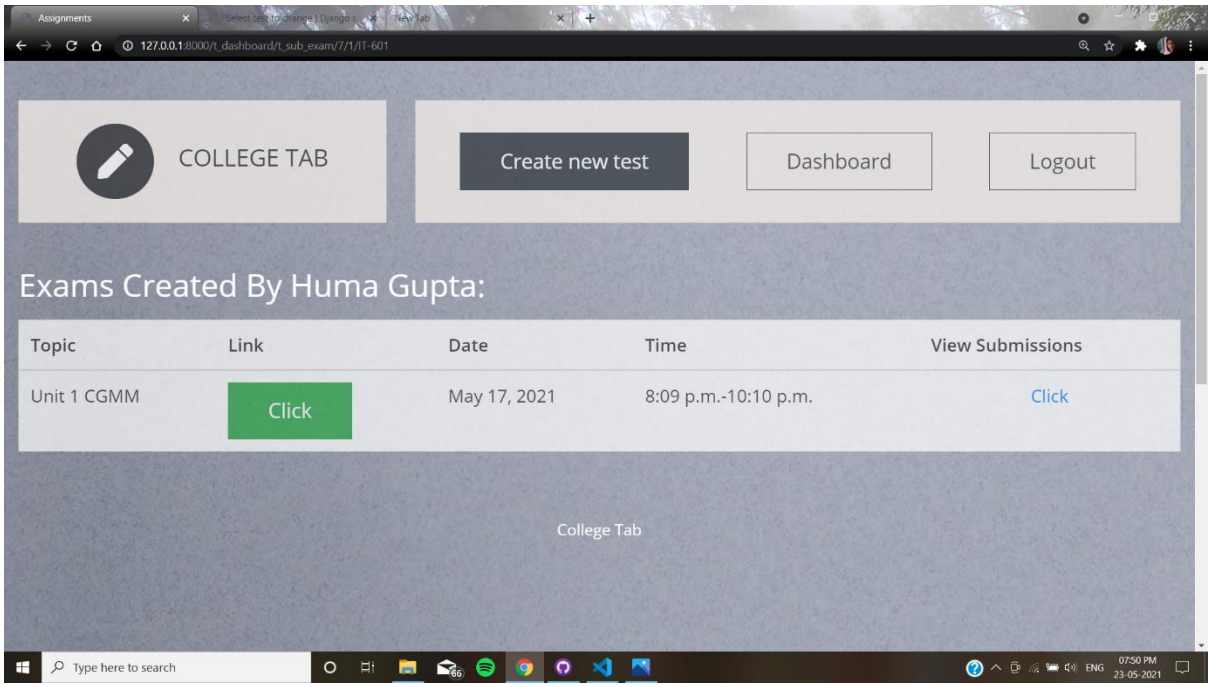


Figure 23

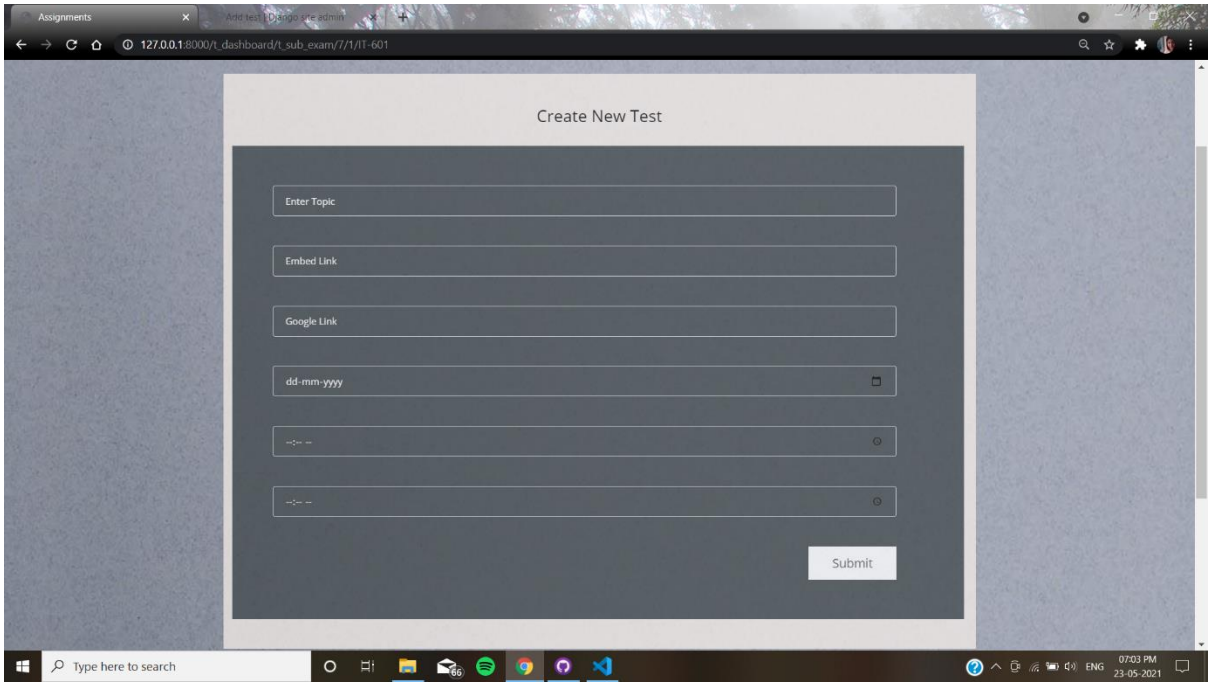


Figure 24

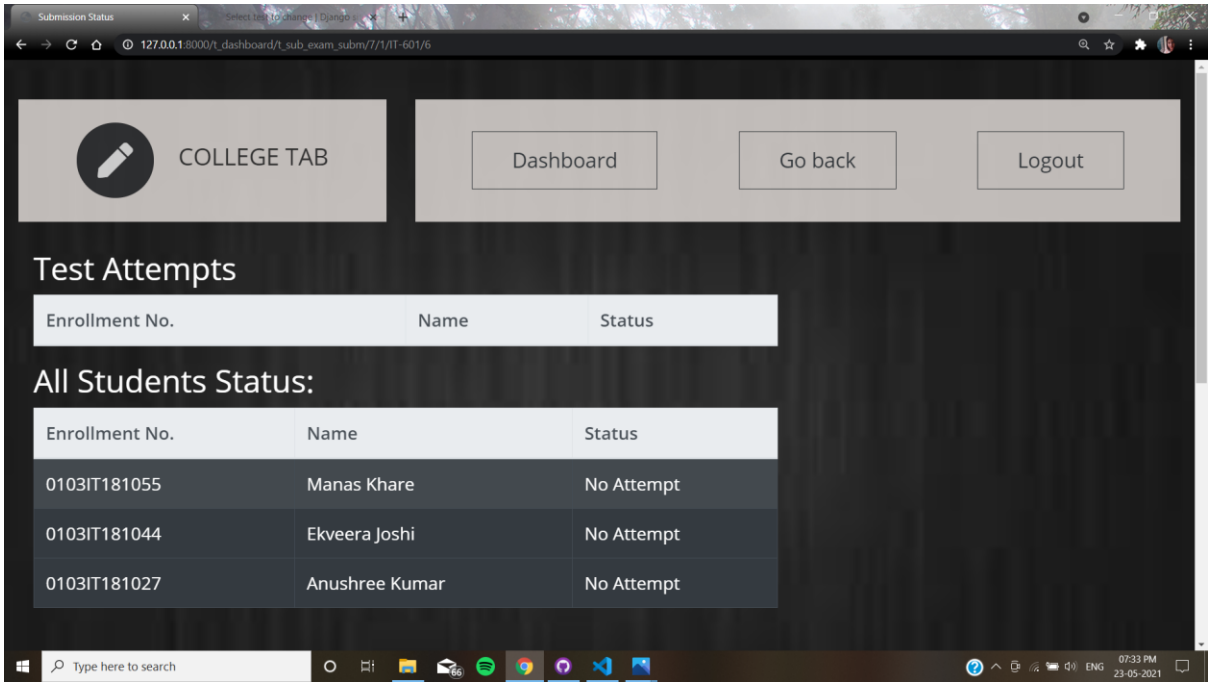


Figure 25

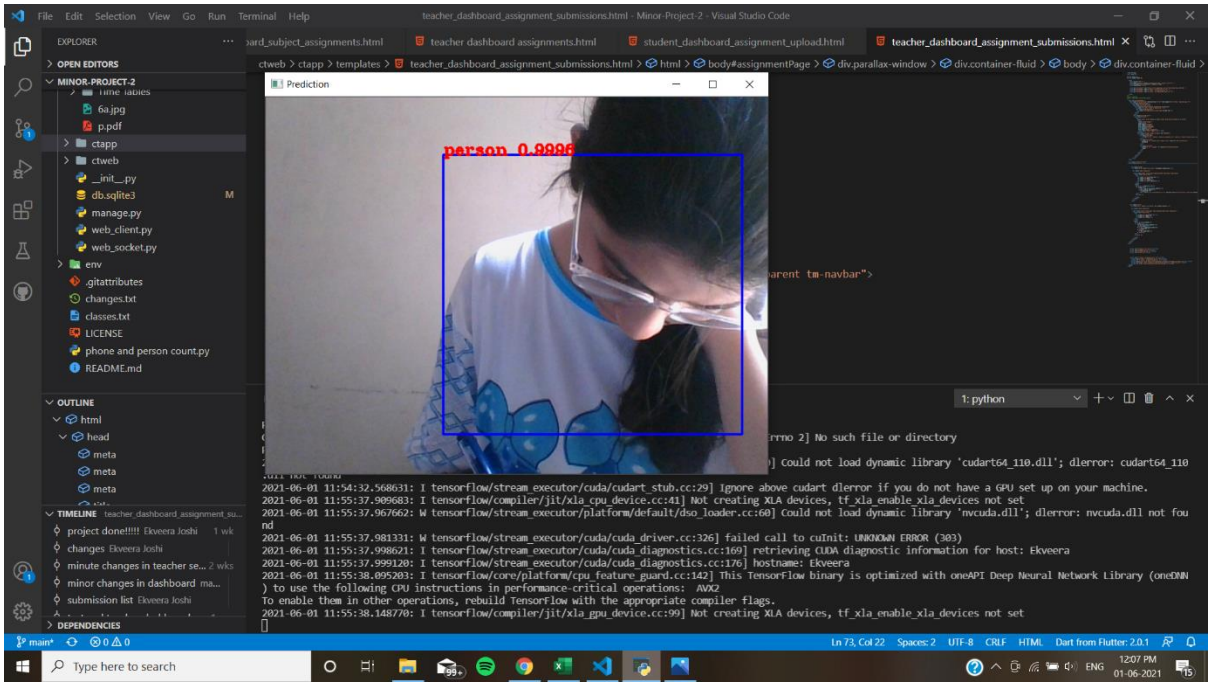


Figure 26

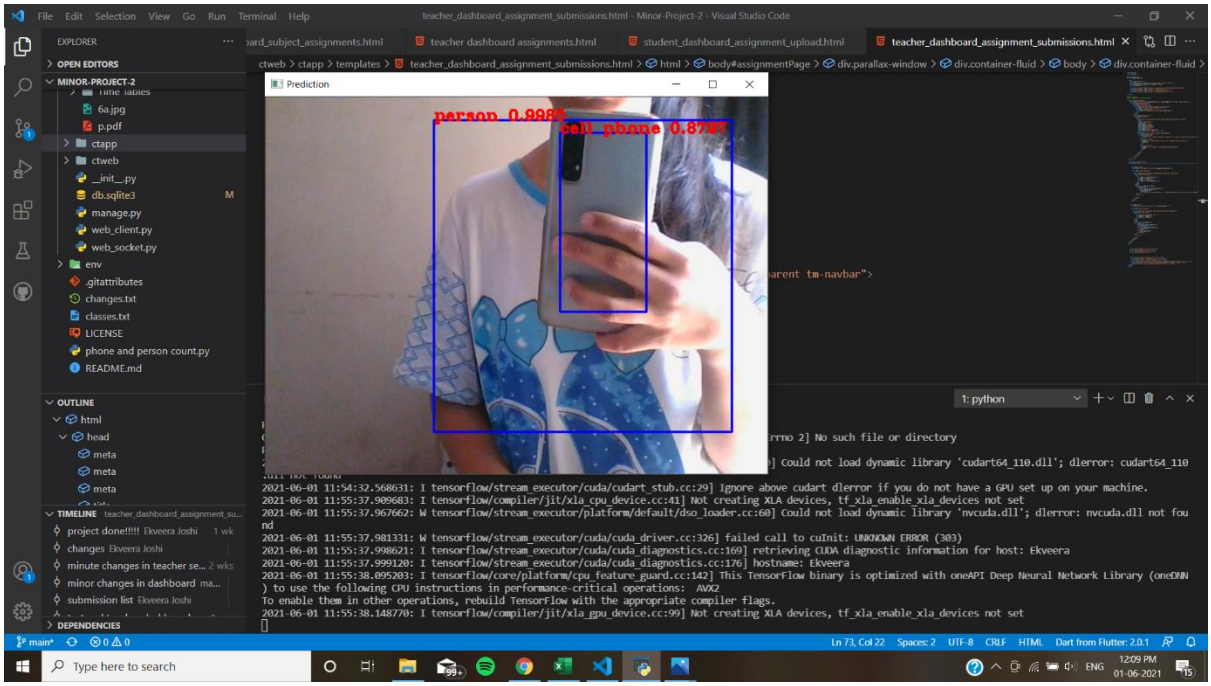


Figure 27

CONCLUSION AND FUTURE SCOPE

5.1 Conclusion

The Django framework gives us a simple and reliable way to create the course management system. It provides powerful functionalities and concise syntax to help programmers deal with the database, the web page and the inner logic. The experience of developing the group component in the system also helped me learning a lot of website development with Django. Within the Django framework, we have successfully accomplished the requirements of the system. Once this system passes the testing phase, it can be used to serve students and instructors and substitute several systems currently in service. It will make the work for instructors to manage the course much easier. It also can simplify the operations for students with assignment submission and conduction of proctored examination all in one system. In short, this system will bring great user experience to both instructors and students.

5.2 Future Scope

We plan to add the following feature in the future:

- The student will be able to receive acknowledgement after the submission of assignment or test paper.

REFERENCES

Book

1. Django for Beginners: Build websites with Python and Django

This is the easiest starting book for your Django learning. The book is fast-paced and the author makes sure you don't feel like you are reading a boring technical textbook. The transition of concepts from easy to difficult is smooth from chapter to chapter. You don't have to be a pro in Python to learn Django from this book. If you are new to Python as well, with help from some online resources, you will be able to follow the concepts of this book well. The author makes you think deeply about a concept and encourages you to try on the concepts by building basic applications. As a beginner, if you buy this book, perhaps you wouldn't need any other guide. From the simple hello world app to more complex websites, the book follows an engaging structure and leaves you seeking more.

2. Django for Professionals: Production websites with Python & Django

If you have worked on a Django project before and want to properly learn the framework, this is the best book. It is written keeping in mind both beginners and advanced level professionals. It is your go-to reference guide for tips and valuable suggestions for best practices. There are plenty of funny but informative diagrams that keep you hooked on the book and the authors have maintained a friendly tone for writing as well. The best part of the book is that you can skip chapters and read them as you like. Each chapter is independent of others. You would also learn to deploy your application to the cloud (PaaS). The authors' experience clearly shows up with the wealth of information they have shared in the form of tips, code samples, tricks, and techniques.

YouTube Video

https://www.youtube.com/watch?v=SIyxjRJ8VNY&list=PLsyeobzWxl7r2ukVgTqIQcl-1T0C2mzau

APPENDICES

7.1 Code

7.1.1 Backend

7.1.1.1 Views

```
@login_required(login_url='t_login/')
def teacher_dashboard(request,id):
    template = loader.get_template('teachers dashboard.html')
    current_teacher = teacher.objects.get(id=id)
    assigned_classes = get_classes(current_teacher)
    context = {'teacher':current_teacher,'class_list':assigned_classes}
    return HttpResponse(template.render(context, request))


@login_required(login_url='t_login/')
def teacher_subject_assign(request, id, sub_id,class_id):            # to
list assignements given subject and class
    if request.method == 'POST':
        form = AssignCreateForm(request.POST,request.FILES)
        if form.is_valid():
            new_assignment = form.save(commit=False)
            new_assignment.t_id = teacher.objects.get(id=id)
            new_assignment.c_id = Class.objects.get(id=class_id)
            new_assignment.s_id = subject.objects.get(subject_code=sub_id)
            new_assignment.save()
            print(new_assignment)
    form = AssignCreateForm()
    template = loader.get_template('teacher dashboard assignments.html')
    print(id,sub_id,class_id)
    current_teacher = teacher.objects.get(id=id)
    current_subject = subject.objects.get(subject_code=sub_id)
    current_class=Class.objects.get(id=class_id)
    assignment_list =
get_assignment_teacher(current_subject,current_teacher)
    context =
{'teacher':current_teacher,'assignment_list':assignment_list,'subject':curr
ent_subject,'class':current_class,'form':form}
    return HttpResponse(template.render(context, request))


#submission for test
@login_required(login_url='t_login/')
def teacher_assignment_submission(request,id,sub_id,class_id,a_id):# list
students who have submitted this assignment
    current_teacher = teacher.objects.get(id=id)
    current_subject = subject.objects.get(subject_code=sub_id)
    current_class=Class.objects.get(id=class_id)
    template =
```

```
loader.get_template('teacher_dashboard_assignment_submissions.html')
    a = assignment.objects.filter(id=a_id)
    student_submission_list = student_submission.objects.filter(a_id=a_id)
    all_students =
student.objects.all().filter(branch=current_class.branch)
    student_who_submit = [s.stud_id for s in student_submission_list]
    context =
{'submission_list':student_submission_list,'assignment':a,'subject':current
_subject,'class':current_class,'teacher':current_teacher,'all_students':all
_students,'student_who_submit':student_who_submit}
    return HttpResponse(template.render(context, request))

@login_required(login_url='t_login/')
def assignment_viewer(request,id):
    submission = student_submission.objects.filter(id=id)
    template = loader.get_template('index.html')
    context = {'submission':submission}
    return HttpResponse(template.render(context,request))
```

7.1.1.2 Models

```
class Class(models.Model):
    timetable = models.ImageField(upload_to='Time
Tables/',default="6a.jpg")
    sem = models.CharField(max_length = 20,choices =
SEMESTER_CHOICES,default = '6')
    sec = models.CharField(max_length = 1,choices = SEC_CHOICES,default =
'A')
    branch= models.CharField(max_length = 20,choices =
BRANCH_CHOICES,default = 'IT')


    def __str__(self):
        return str(self.sem)+" "+str(self.sec)+" - "+str(self.branch)


class teacher_assign(models.Model):
    c_id  = models.ForeignKey(Class,on_delete=models.CASCADE)    #class id
    s_id = models.ForeignKey(subject,on_delete=models.CASCADE)  #subject id
    t_id = models.ForeignKey(teacher,on_delete=models.CASCADE)  #teacher id
    def __str__(self):
        t_name = self.t_id.name     #get name of teacher
        s_name = self.s_id.name     #get subject name
        sem_sec = str(self.c_id.sem) + "TH - " + str(self.c_id.sec)
        return sem_sec+" - "+s_name+' - ' + t_name

#to get separate folder for each assignment created
def get_directory_for_assignment_details(instance, filename):
    exe = filename.split('.')[-1]
    # file will be uploaded to MEDIA_ROOT/sem_sec_subject_topic/<filename>
    cclass = str(instance.c_id.sem) + "_" + str(instance.c_id.sec)
    sub = str(instance.s_id.name)
    top = str(instance.topic)
    filename = "Assignement_Questions"
    return
'Assignments/class_{0}/{1}/{2}/{3}.{4}'.format(cclass,sub,top,filename,exe)


class assignment(models.Model):
    c_id  = models.ForeignKey(Class,on_delete=models.CASCADE)    #class id
```

```
    s_id = models.ForeignKey(subject,on_delete=models.CASCADE)  #subject id
    t_id = models.ForeignKey(teacher,on_delete=models.CASCADE)  #teacher id
    topic = models.CharField(max_length=50)
    last_date = models.DateField()
    assignfile =
models.FileField(upload_to=get_directory_for_assignment_details,default="p.
pdf")

    def __str__(self):
        s_name = str(self.s_id.name)      #get subject name
        sem_sec = str(self.c_id.sem)+"TH - "+str(self.c_id.sec)
        return sem_sec+" - "+s_name+' - ' + str(self.topic)
```

7.1.1.3 Web Socket

```
define("port", default=8001, help="run on the given port", type=int)

class MainHandler(tornado.websocket.WebSocketHandler):
    count = 0
    def check_origin(self, origin):
        return True

    def open(self):
        print("connected")
        self.frames = []
        logging.info("A client connected.")

    def on_close(self):
        logging.info("A client disconnected")

        import time
        time.sleep(5)
        print('now loading images')
        #loading frames to constuct video
        img_array = []
        for filename in self.frames:
            img = cv2.imread(filename)
            img_array.append(img)
            #print(filename)
            #height, width, layers = img.shape
            #size = (width, height)
            #nparr = np.fromstring(base64.b64decode(frame), np.uint8)
            #img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

        print('now making video')
        name = 'Images/'+self.stud_id+'.avi'
        out = cv2.VideoWriter(name, cv2.VideoWriter_fourcc(*'DIVX'), 15,
(600,450))
        for i in range(len(img_array)):
            out.write(img_array[i])
        out.release()
        print('now deleting frames')
        #deleting frames
        import os
        for filename in self.frames:
            os.remove(filename)

        # This logic is necessary to import django files from non-django
python files like this
```

```
        import sys
        sys.path.append('..')
        import django
        django.setup()
        from ctapp.models import student_testsubmission, student, Test
#this might look like error but it's not
        #x = student.objects.get(enrollment_number='0103IT181055')
        #print(x)


    # ADD logic to create a test submission object which has 3 values stud
id , test id ,a video file and proctor description
        stud_object = student.objects.get(enrollment_number=self.stud_id)
        test_obj = Test.objects.get(id=int(self.test_id))
        submission = student_testsubmission()
        submission.stud_id = stud_object
        submission.test_id = test_obj
        f = open(name,'rb')
        file = File(f)
        submission.video = file
        submission.save()
        f.close()

        #delete video locally stored
        os.remove(name)

        print("Successful!!!!")



    def on_message(self, message):
        #from main.detect import get_face_detect_data
        #image_data = get_face_detect_data(message)
        #print("received",message)
        #print(message)
        if message.startswith('stud:'):
            print(message)
            self.stud_id = message.split(':')[1]
            print(self.stud_id)
        elif message.startswith('testid:'):
            #print(message)
            self.test_id = int(message.split(':')[1])
            #print(self.test_id)
        image_data = message[22:]#cropping header
        name = 'Images/check'+ str(MainHandler.count) +'.png'
        image = open(name, 'wb')
        image.write(base64.b64decode((image_data)))
        image.close()
        self.frames.append(name)
        MainHandler.count += 1
        if not image_data:
            image_data = message
        self.write_message(image_data)
```