MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Information Technology Lab

Student – Teacher Dashboard (Django)

SUBMITTED BY

SUJEET SANJAY AMBERKAR 200905092

Under the Guidance of:
Dr. Roopalkshmi R and Ms Priya Kamath
Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal Karnataka – 576104

April 2024

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
05/03/2024

CERTIFICATE

This is to certify that the project titled Student Teacher Dashboard is a record of the bonafide work done by Student Sujeet Sanjay Amberkar (Reg. No. 200905092) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2022-2023.

Name and Signature of Examiners:

1. Dr. Roopalakshmi R, Associate Professor, CSE Dept.

2. Prof. Priya Kamath, Assistant Professor, CSE Dept.

Index

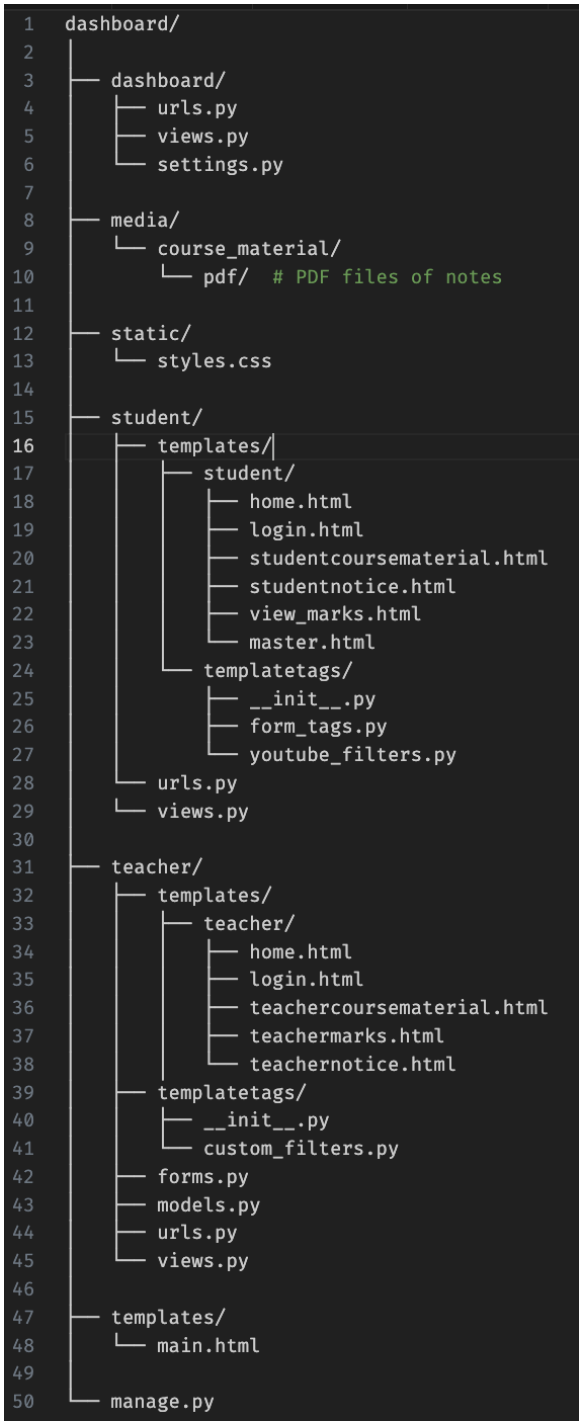
Abstract

The aim of the project “Student-Teacher Dashboard” is to create a structured communication channel between teachers and students. It facilitates various academic and administrative functions, including notice dissemination, access to course materials, and display of marks. This report documents the project’s functionalities, steps of installation and limitations.

Introduction

The student–teacher dashboard is the one-stop solution for all the communication between teacher and student. Using this software, teachers will be able to share study material, manage marks and also publish notices for students.

File Structure



Methodology

Project Structure

The project is structured into a main Django project named "Student-Teacher dashboard," encompassing two key applications: student and teacher.''

- Student Application: Manages student interactions with the platform, including login, notice viewing, access to course materials, and marks review.

- Teacher Application: Handles teacher-related functionalities such as login, posting notices, uploading course materials, and entering student marks.

Implementation

Groups and Permissions

For Students:

1. **Login**: Secure authentication to access personal and course-related information.
2. **Master**: This views the entire dashboard for students. It contains all the notices, lecture notes, video lectures and marks obtained by that particular student.

For Teachers:

1. **Login**: Secure authentication for accessing administrative interfaces.
2. **Post Notices**: Upload notices relevant to students.
3. **Upload Course Materials**: Share educational resources with students.
4. **Enter Marks**: Record and publish student performance metrics.

User Management

1) The "Dashboard" project employs a centralized user management system to maintain control ensure secure access prevent any unauthorized access and maintain the integrity of the records.
2) There is no signup page, The superuser will create the username and password and give it to students and teachers

Database Design

User Management

1) We are using the SQLite database to manage users and groups, which is facilitated through the auth_user and auth_group tables.

Teachers have an option to edit data but students can only view the data

Notices Table:

1) id: Primary key.
2) title: The title of the notice.
3) content: The detailed content of the notice.
4) posted_by: Foreign key linking to the auth_user table to identify the teacher.
5) created_at: Timestamp indicating when the notice was posted.

Course Material

This includes materials uploaded by teachers, such as chapter names, YouTube URLs, and notes in PDF format.

1) CourseMaterial Table:
2) id: Primary key.
3) chapter_name: The name of the chapter.
4) video_url: URL of the corresponding YouTube video.
5) pdf_file: FileField to store the uploaded PDF file.
6) created_at: Timestamp for when the material was uploaded.

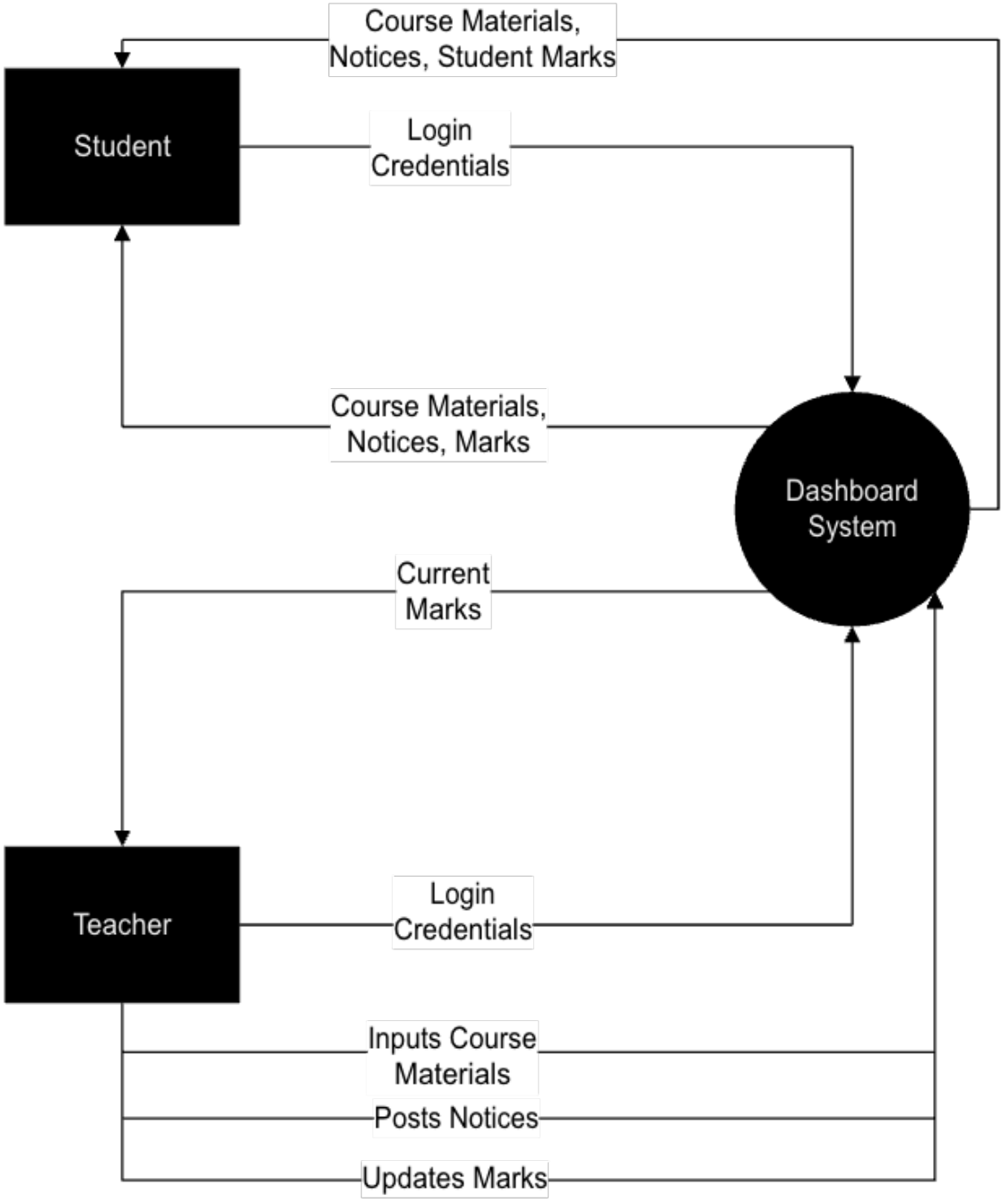
Marks

1) Student (Foreign Key)
2) Marks Obtained
3) Total Marks

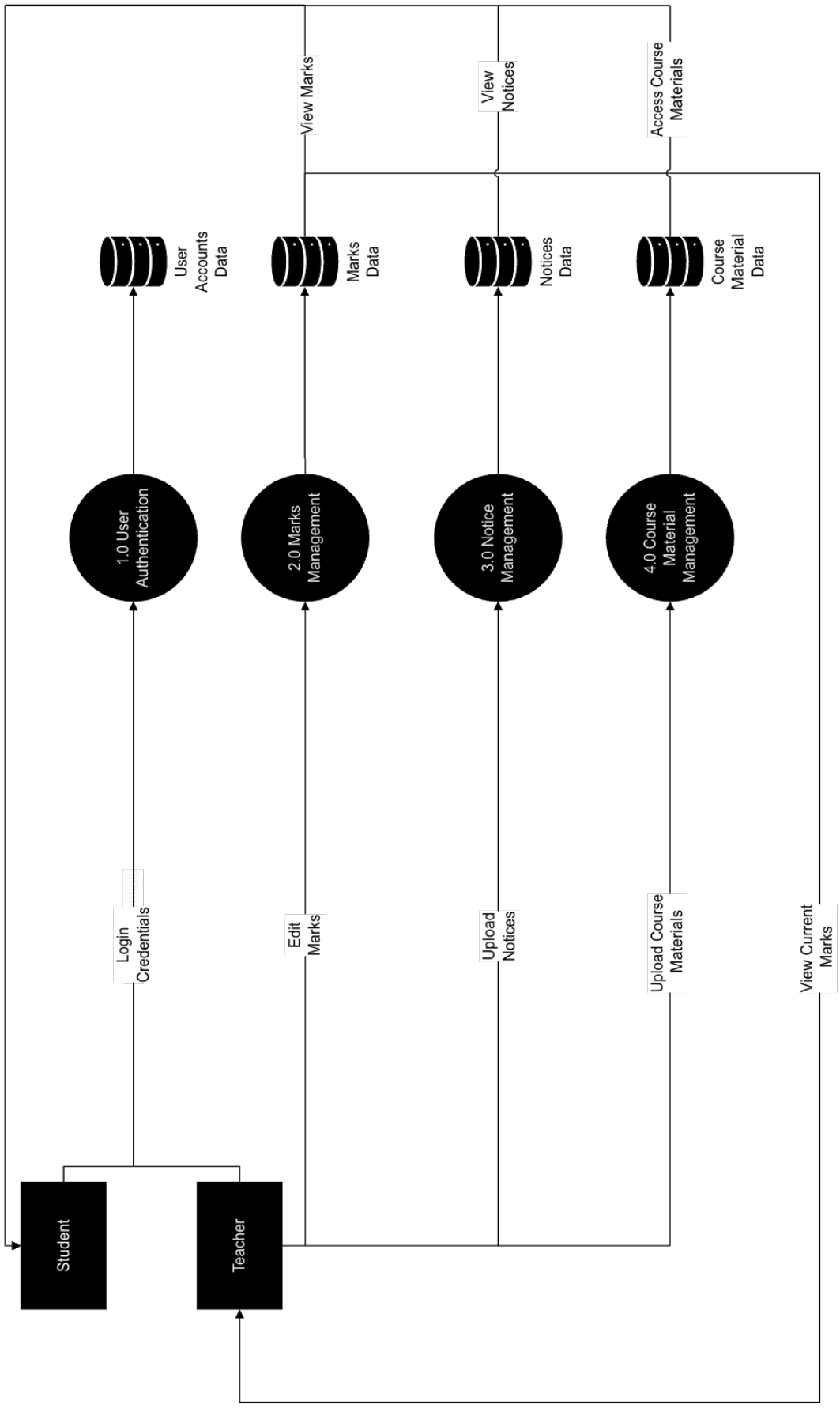
To manage migrations and interact with the database, Django provides the following commands:

- python manage.py makemigrations: Prepares migrations based on model changes.
- python manage.py migrate: Applies migrations to the database, structuring it according to defined models.

Data Flow Diagram



Level 0 : DFD Diagram



Level 1 DFD Diagram

Code:

teacher/views.py

```
from django.shortcuts import render,redirect
from django.http import HttpResponse
from django.shortcuts import redirect
from django.contrib.auth import authenticate, login
from django.contrib.auth.forms import AuthenticationForm
from .models import Notice , StudentMarks
from .forms import CourseMaterialForm, MarksForm
from django.contrib.auth.models import User
from django.forms import modelformset_factory
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User, Group
def home(request):
    return render(request, 'teacher/home.html')
def teacher_login(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(username=username, password=password)
            if user is not None and user.groups.filter(name='teachers').exists():
                login(request, user)
                return redirect('teacher_home')
            else:
                return render(request, 'teacher/login.html', {'form': form, 'error': 'Invalid
credentials or you do not have permission to access this page.'})
    else:
        form = AuthenticationForm()
    return render(request, 'teacher/login.html', {'form': form})
@login_required
def teacher_notice(request):
    if request.method == 'POST':
        notice_content = request.POST.get('notice')
        if notice_content:
            Notice.objects.create(content=notice_content)
            return redirect('teacher_home')
        else:
            return render(request, 'teacher/teachernotice.html', {
                'error': 'Please enter some text for the notice.'
            })
    return render(request, 'teacher/teachernotice.html')
def add_course_material(request):
    if request.method == 'POST':
        form = CourseMaterialForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect('teacher_home')  # Adjust as necessary
    else:
        form = CourseMaterialForm()
    return render(request, 'teacher/teachercoursematerial.html', {'form': form})
@login_required
def enter_update_marks(request):
    student_group = Group.objects.get(name='students')
    students = student_group.user_set.all()
    if request.method == 'POST':
        for student in students:
            marks_obtained = request.POST.get(f'marks_obtained_{student.id}')
            total_marks = request.POST.get(f'total_marks_{student.id}')
            StudentMarks.objects.update_or_create(
                student=student,
                defaults={
                    'marks_obtained': marks_obtained,
                    'total_marks': total_marks
                }
            )
        return redirect('enter_update_marks')
    marks = {mark.student_id: mark for mark in StudentMarks.objects.filter(student__in=students)}
    return render(request, 'teacher/teachermarks.html', {'students': students, 'marks': marks})
```

teacher/ forms.py

```
from django import forms
from .models import CourseMaterial
from .models import StudentMarks
class CourseMaterialForm(forms.ModelForm):
    class Meta:
        model = CourseMaterial
        fields = ['chapter_name', 'pdf_file', 'video_url']
class MarksForm(forms.ModelForm):
    class Meta:
        model = StudentMarks
        fields = ['student', 'marks_obtained', 'total_marks']
```

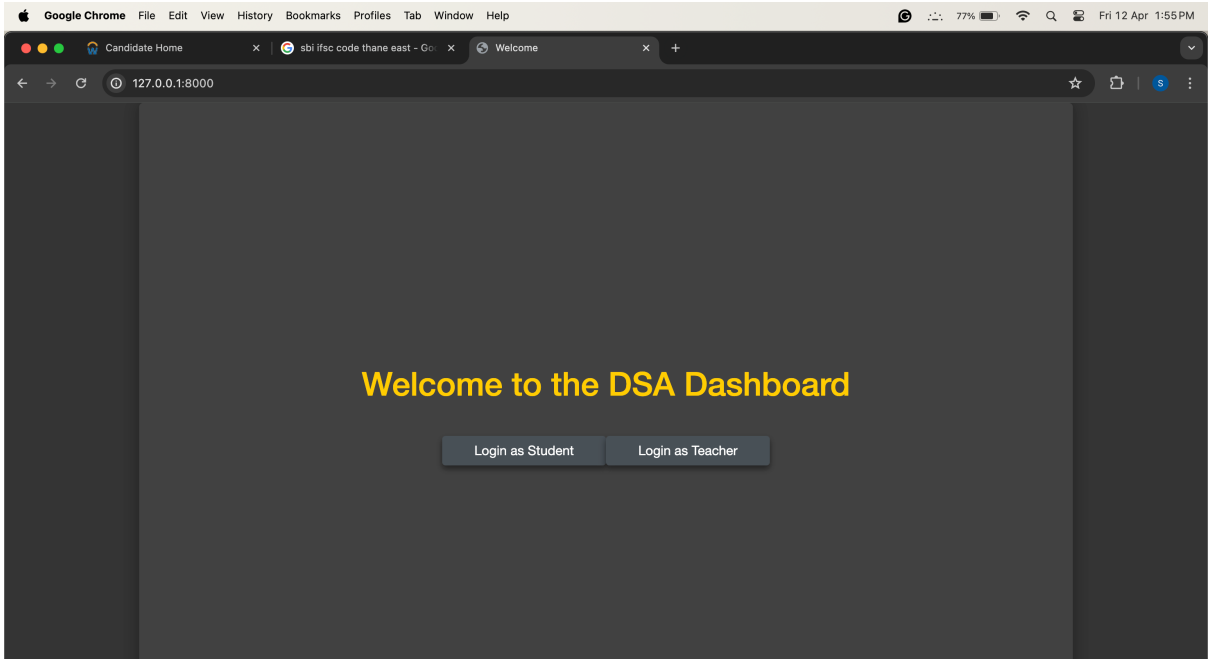
teacher/ models.py

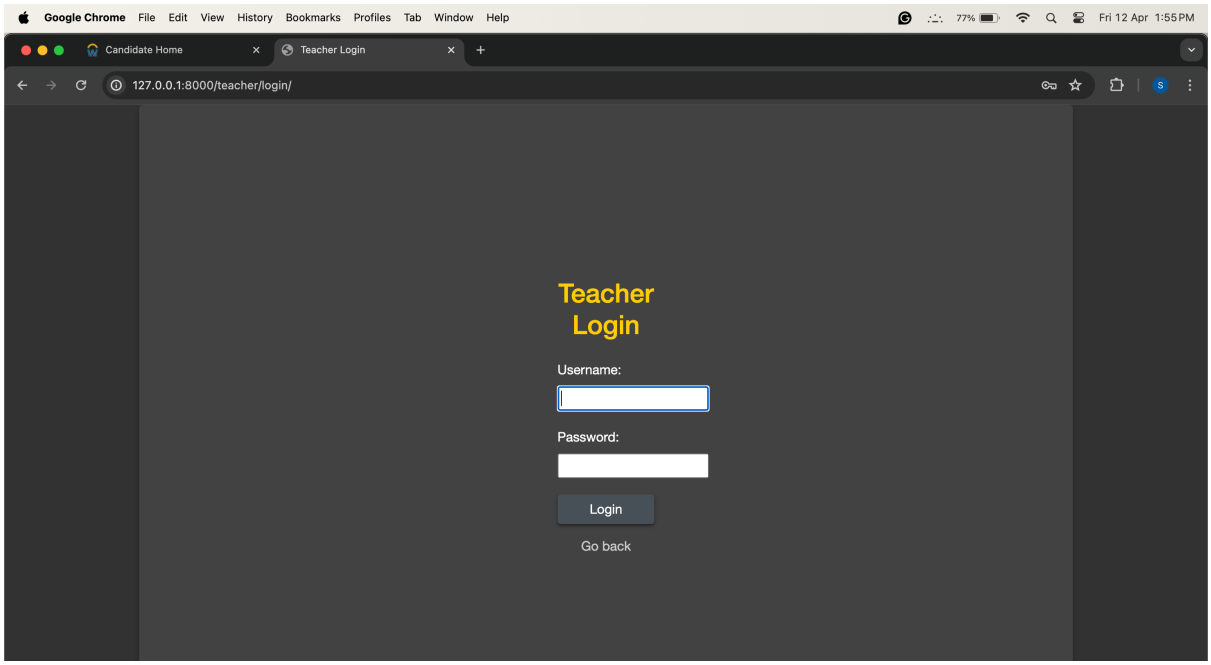
```
from django.db import models
from django.contrib.auth.models import User
from django.conf import settings
class Notice(models.Model):
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
class CourseMaterial(models.Model):
    chapter_name = models.CharField(max_length=255)
    pdf_file = models.FileField(upload_to='course_materials/pdfs/')
    video_url = models.URLField(max_length=1024)

    def __str__(self):
        return self.chapter_name
class StudentMarks(models.Model):
    student = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE,
related_name='marks')
    marks_obtained = models.IntegerField(default=0)
    total_marks = models.IntegerField(default=100)
    def __str__(self):
        return f"{self.student.username} - {self.marks_obtained}/{self.total_marks}"
```

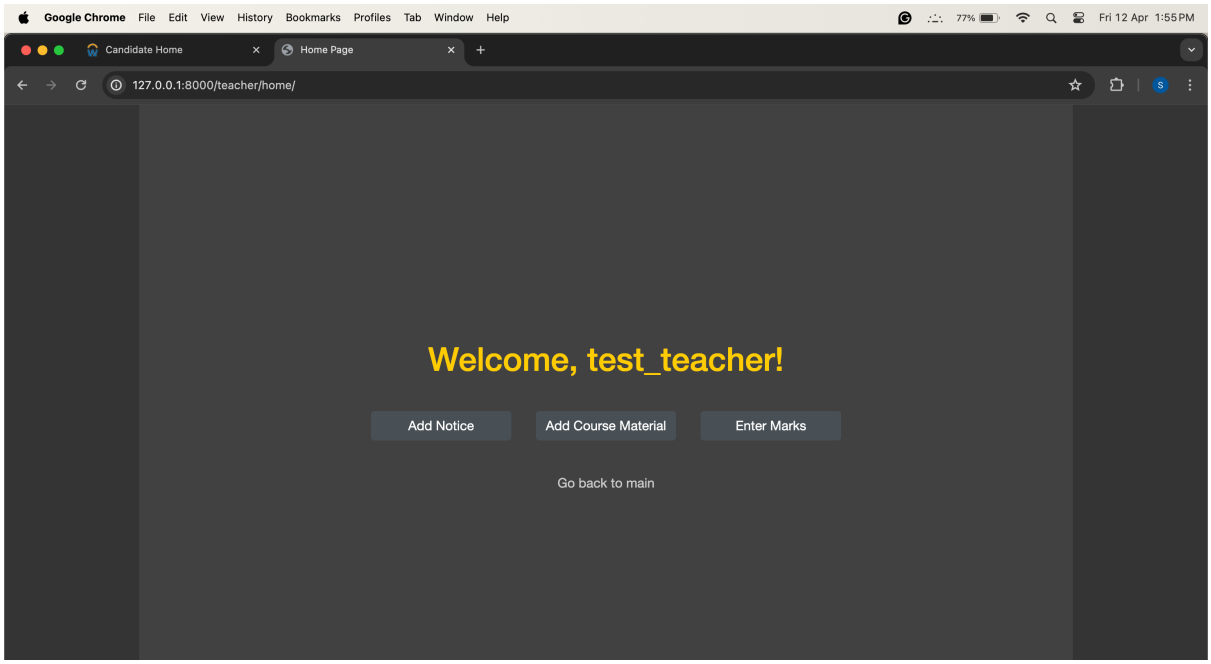
student/views.py

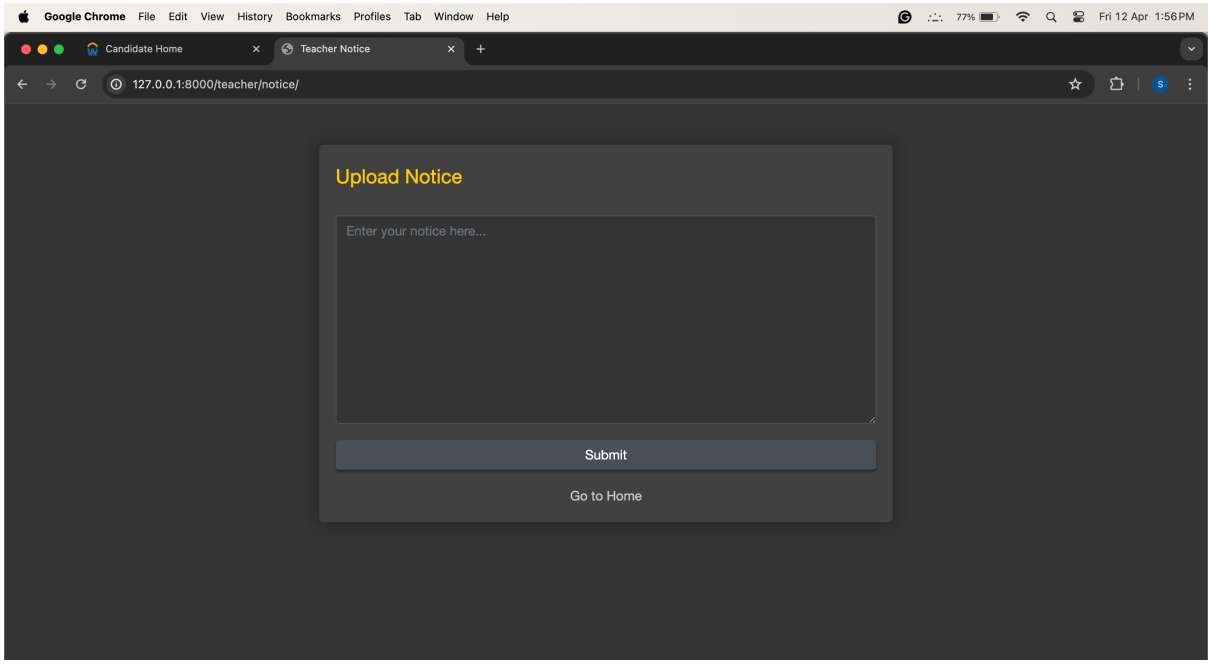
```
from django.shortcuts import render
from teacher.models import Notice
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, login_required
from django.contrib.auth.forms import AuthenticationForm
from django.http import HttpResponseRedirect
from teacher.models import CourseMaterial, StudentMarks
def student_login(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(username=username, password=password)
            if user is not None and user.groups.filter(name='students').exists():
                login(request, user)
                return redirect('master')  # Redirect to the student's home page
            else:
                return render(request, 'student/login.html', {'form': form, 'error': 'Invalid
credentials or you do not have permission to access this page.'})
    else:
        form = AuthenticationForm()
    return render(request, 'student/login.html', {'form': form})
@login_required
def master_view(request):
    notices = Notice.objects.all().order_by('-created_at')
    materials = CourseMaterial.objects.all()
    try:
        marks_instance = StudentMarks.objects.get(student=request.user)
        percentage = (marks_instance.marks_obtained / marks_instance.total_marks) * 100
    except StudentMarks.DoesNotExist:
        marks_instance = None
        percentage = None
    context = {
        'user': request.user,
        'notices': notices,
        'materials': materials,
        'marks_instance': marks_instance,
        'percentage': percentage
    }
    return render(request, 'student/master.html', context)
```

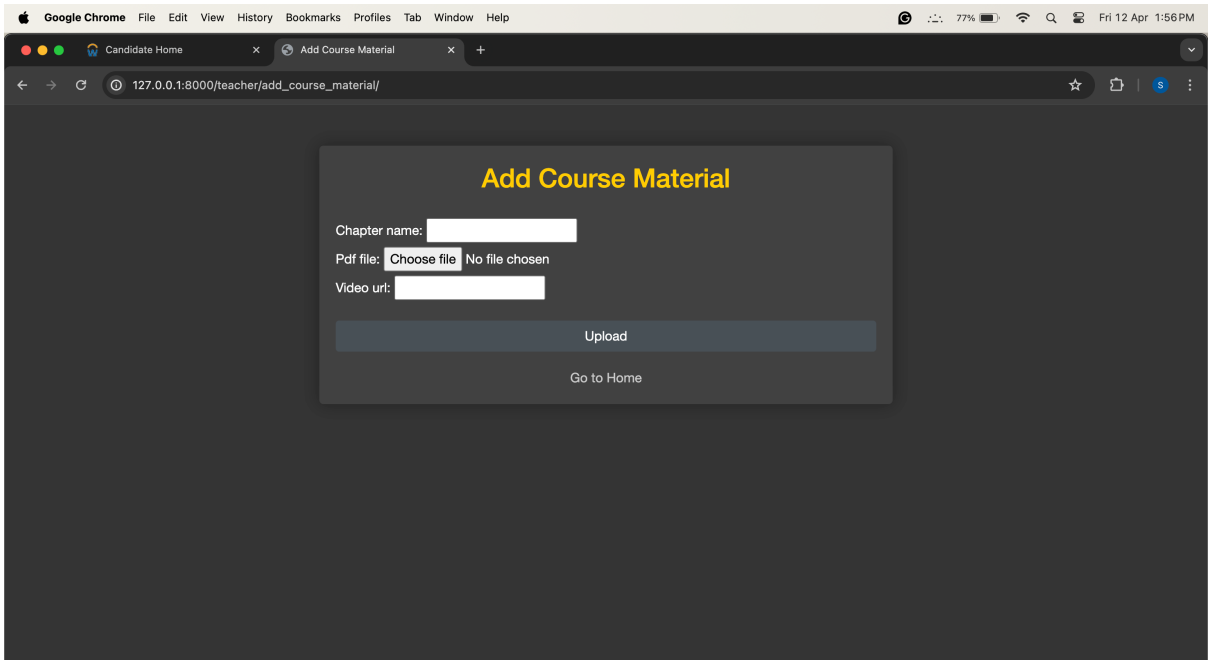
Results And Snapshots

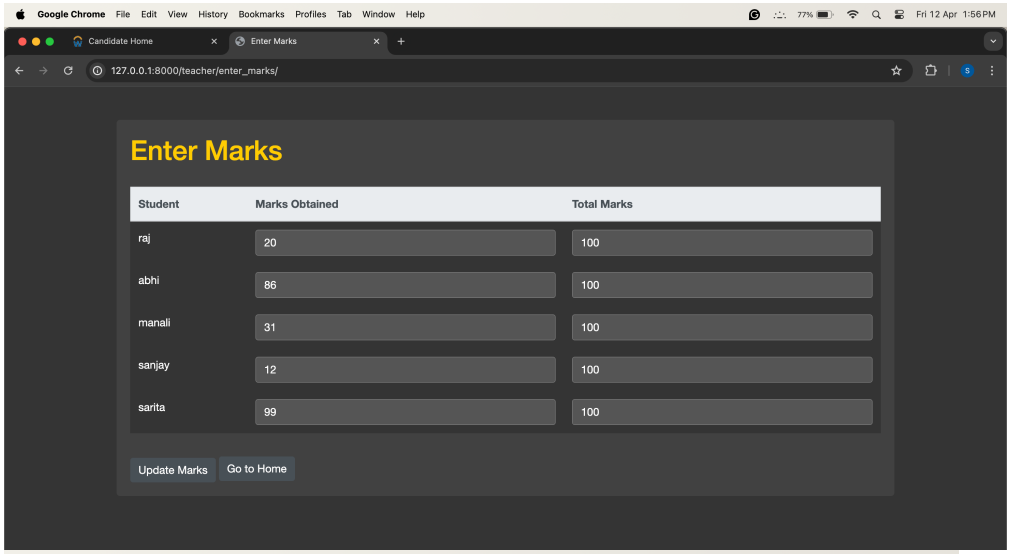


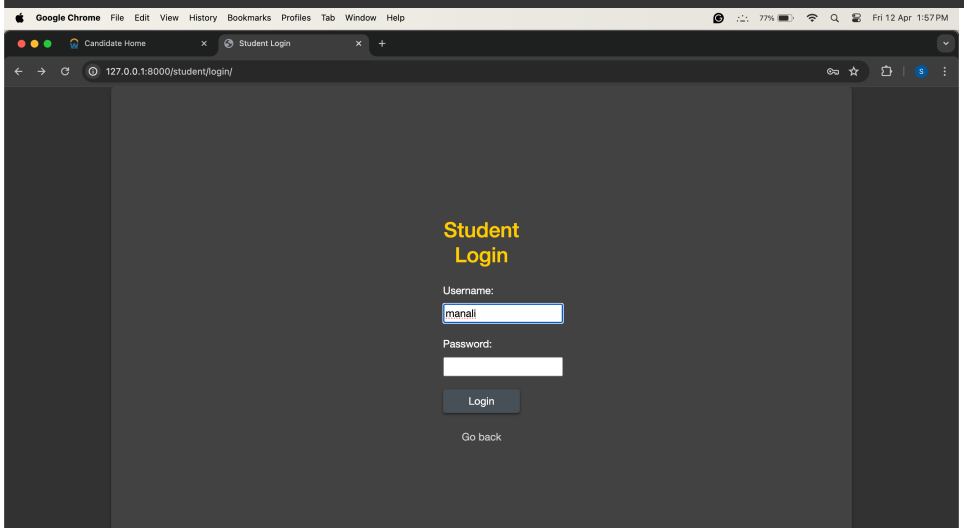


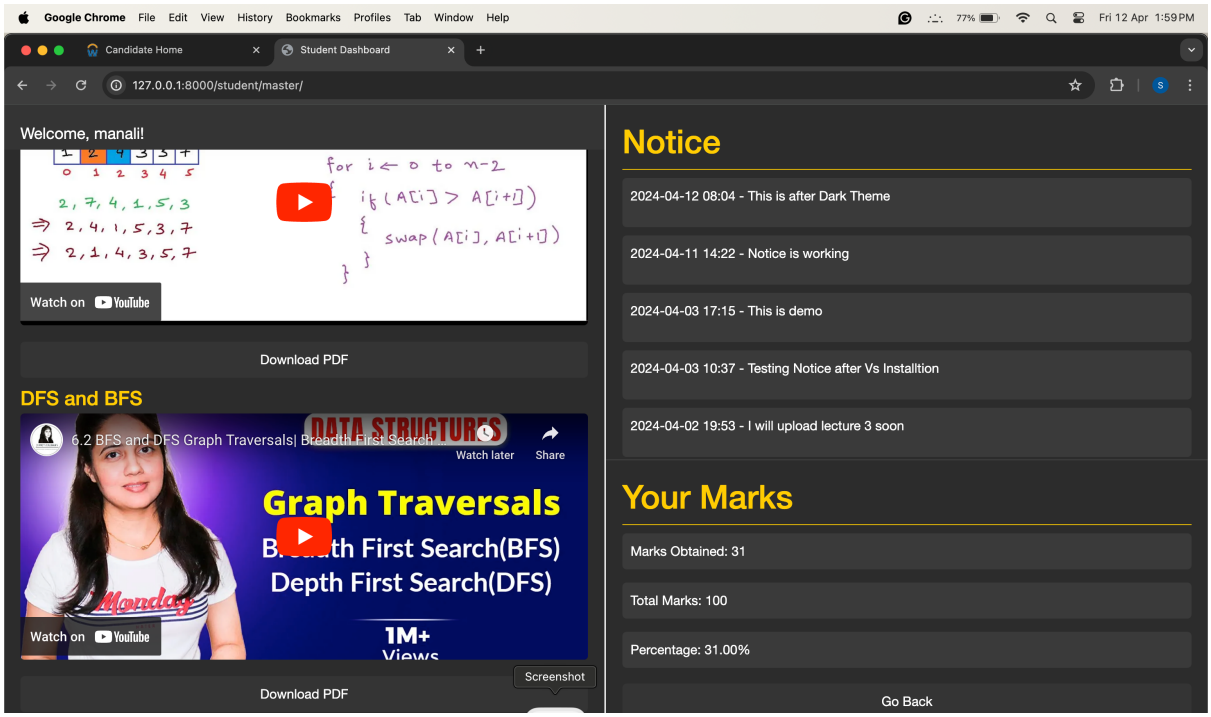

Google Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 12 Apr 1:55 PM
Candidate Home
Home Page
127.0.0.1:8000/teacher/home/
Welcome, test_teacher!
Add Notice
Add Course Material
Enter Marks
Go back to main

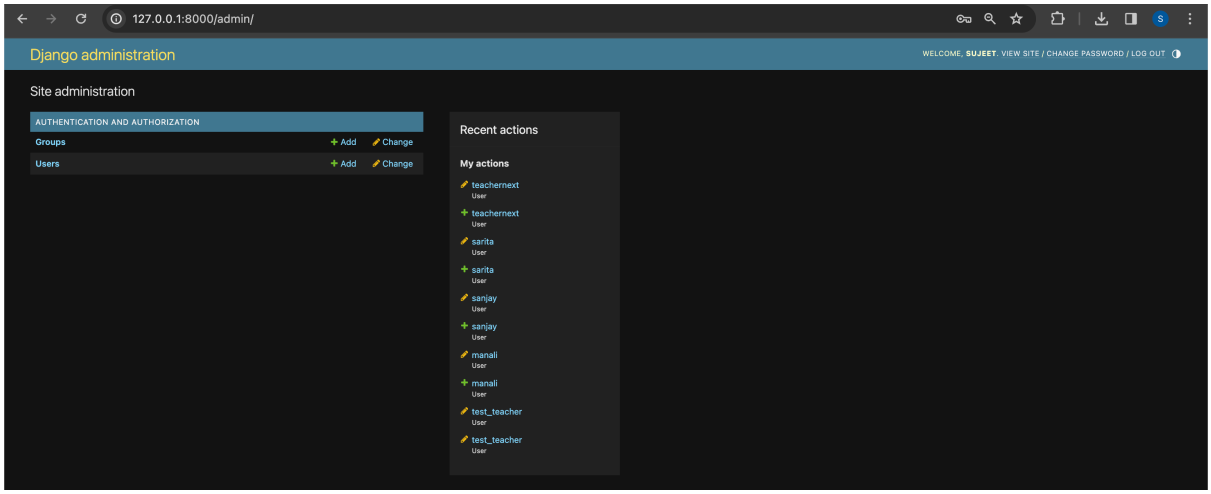

Google Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 12 Apr 1:56 PM
Candidate Home
Teacher Notice
127.0.0.1:8000/teacher/notice/
Upload Notice
Enter your notice here...
Submit
Go to Home


Google Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 12 Apr 1:56 PM
Candidate Home
Add Course Material
127.0.0.1:8000/teacher/add_course_material/
Add Course Material
Chapter name:
Pdf file:
Choose file
No file chosen
Video url:
Upload
Go to Home


Google Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 12 Apr 1:56 PM
Candidate Home
Enter Marks
127.0.0.1:8000/teacher/enter_marks/
Enter Marks
Student
Marks Obtained
Total Marks
raj
20
100
abhi
86
100
manali
31
100
sanjay
12
100
sarita
99
100
Update Marks
Go to Home


Google Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 12 Apr 1:57 PM
Candidate Home
Student Login
127.0.0.1:8000/student/login/
Student Login
Username:
manali
Password:
Login
Go back


Google Chrome File Edit View History Bookmarks Profiles Tab Window Help
Fri 12 Apr 1:59 PM
Candidate Home
Student Dashboard
127.0.0.1:8000/student/master/
Welcome, manali!
Watch on YouTube
Download PDF
DFS and BFS
6.2 BFS and DFS Graph Traversals| Breadth First Search
Watch later
Share
Graph Traversals
Depth First Search(DFS)
Watch on YouTube
1M+
Download PDF
Screenshot
Notice
2024-04-12 08:04 - This is after Dark Theme
2024-04-11 14:22 - Notice is working
2024-04-03 17:15 - This is demo
2024-04-03 10:37 - Testing Notice after Vs Installtion
2024-04-02 19:53 - I will upload lecture 3 soon
Your Marks
Marks Obtained: 31
Total Marks: 100
Percentage: 31.00%
Go Back


127.0.0.1:8000/admin/
Django administration
WELCOME, SUJEET. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups + Add Change
Users + Add Change
Recent actions
My actions
teachernext User
teachernext User
sarita User
sarita User
sanjay User
sanjay User
manali User
manali User
test_teacher User
test_teacher User

Conclusion

The "Dashboard" project represents a significant step forward in simplifying the formal communication between teachers and students.

By leveraging the power of Django we are able to create a user friendly platform to share study material, notices and displaying maths

Future Directions

1) **Feature Expansion**: More features can be added to this project like daily attendance and a way for students to submit their assignments
2) **Use Media**- Query to make give better user experience of mobiles

References

1. Django Documentation: https://docs.djangoproject.com/en/5.0/
2. Bootstrap Documentation: https://getbootstrap.com/docs/4.1/getting-started/introduction/

Github :

https://github.com/sujeetamberkar/it_Project_Dashboard