Ed-Analytics- Educational Analytics Dashboard

A PROJECT REPORT

Submitted by

Amandeep Verma

in partial fulfilment for the award of the degree

of

M.Sc. Data Science and Artificial Intelligence – Part 1

RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ART'S,

SCIENCE & COMMERCE (AUTONOMOUS),

GHATKOPAR (W)

April– 2023

RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ART’S, SCIENCE & COMMERCE (AUTONOMOUS), GHATKOPAR (W)

(Affiliated to University of Mumbai)

Certificate

This is to certify that the Project entitled **Ed-Analytics- Educational Analytics Dashboard** *is bonafide work of* **Amandeep Verma** *bearing Seat No* **722** *submitted in partial fulfilment of the requirements for the award of Degree* ***Master of Science*** *in* ***Data Science & Artificial Intelligence.***

Signature of Internal Guide **Signature of Co-ordinator**

College Seal and Date **Signature Examiner**

Abstract

Ed-Analytics is a data science project developed using Django framework to help teachers and students analyze academic performance. The project collects data from R.J.College of students of BAMMC course from the years 2019-2022, which contains 3 years of data. The data is processed and stored in an excel file that contains 5 columns, including name, subject, internal marks, external marks, and semester. The project has three types of users: students, teachers, and admins. Students can view their performance in the latest semester, including average internal and external marks, bar graphs of their marks in each subject and semester. Teachers can see the average marks of students in the subject they teach, top 5 most and least scoring students, and bar graphs of all students' marks in their subject. Admins can view the performance of both students and teachers, total number of students, subjects, average internal and external marks, and an interactive pie chart of subjects in the latest semester. Additionally, admins can manage roles and access rights for different users.

Acknowledgement

Before we get into thick of things, I would like to add a few heartfelt words for the people who were part of Ed-Analytics project in numerous ways, people who gave unending support right from the stage the project idea was conceived.

A project report is such a comprehensive coverage; it would not have been materialized without the help of many. The four things that go on to make a successful endeavour are dedication, hard work, patience and correct guidance. Able and timely guidance not only helps in making an effort fruitful but also transforms the whole process of learning and implementing into an enjoyable experience.

In particular, I would like to thank our Mentor/Director Dr. (Mrs.) Usha Mukundan, R.J. College. I would like to give a very special honor and respect to our teacher, Prof. Mujtaba Shaikh and our project guide Yousoufi Waisullah who took keen interest in checking the minute details of the project work and guided us throughout the same. A sincere quote of thanks to the non-teaching staff for providing us software their time. I appreciate outstanding co-operation by them, especially for the long Lab timings that we could receive.

Declaration

I hereby declare that the Project entitled, ***“Ed-Analytics- Educational Analytics Dashboard”*** done at R. J. COLLEGE, Ghatkopar(W), Mumbai, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my Knowledge other than me, No one has submitted to any other University. The Project is done in partial fulfilment of the requirements for the award of degree of ***Master of Science Data Science and Artificial Intelligence*** to be submitted as mini project as part of our curriculum.

Index

Introduction

Ed-Analytics is a data science project developed using the Django framework to help teachers and students analyze academic performance. The primary objective of the project is to provide an efficient platform for students, teachers, and administrators to access and evaluate academic data. The project collects data from the R.J.College of students pursuing BAMMC course from the years 2019-2022, which contains 3 years of data.

The project uses data science techniques to process and store the data in an excel file containing five columns, including name, subject, internal marks, external marks, and semester. The data is analyzed and presented in a user-friendly manner to facilitate easy interpretation and understanding.

The project has three types of users: students, teachers, and administrators. Each user type has specific access rights and privileges that enable them to access and analyze data that is relevant to them.

Students can view their performance in the latest semester, including average internal and external marks, bar graphs of their marks in each subject and semester. This feature helps students track their academic progress and identify areas where they need improvement.

Teachers can view the average marks of students in the subject they teach, top 5 most and least scoring students, and bar graphs of all students' marks in their subject. This feature helps teachers identify the strengths and weaknesses of their students and tailor their teaching methods accordingly.

Administrators can view the performance of both students and teachers, total number of students, subjects, average internal and external marks, and an interactive pie chart of subjects in the latest semester. Additionally, admins can manage roles and access rights for different users. This feature enables administrators to maintain the integrity and security of the system.

Problem Statement

Academic performance analysis is a critical aspect of education that helps both students and teachers identify strengths and weaknesses, make informed decisions, and improve learning outcomes. However, the traditional methods of performance analysis are often time-consuming, cumbersome, and lack real-time insights. This project aims to provide an interactive platform that enables teachers and students to analyze academic performance efficiently and effectively. The platform collects and processes data from R.J.College of students of BAMMC course from the years 2019-2022 and uses data analysis techniques and visualization tools to provide real-time insights into academic performance. The platform's goal is to help students and teachers identify strengths and weaknesses, improve learning outcomes, and make informed decisions based on data-driven insights. The project's problem statement is to create an interactive platform that facilitates academic performance analysis, enhances decision-making, and improves learning outcomes for students and teachers.

Dataset

The dataset used in the Ed-Analytics project is named stud_cd.csv, which contains data on the academic performance of students in R.J. College pursuing the BAMMC course from the years 2019-2022. The dataset has a total of 5620 rows and includes data on multiple students and subjects across three academic years. The dataset consists of five columns:

- Name: This column contains the name of the student, represented as a string.

- Subject: This column contains the subject of the course, represented as a string.

- Internal Marks: This column contains the internal marks scored by the student in the subject, represented as an integer.

- External Marks: This column contains the external marks scored by the student in the subject, represented as an integer.

- Semester: This column contains the semester in which the student took the course, represented as an integer.

The dataset is a comprehensive representation of the academic performance of students at R.J.College, covering three academic years. The dataset is designed to be used as an input for the Ed-Analytics project, which utilizes data science techniques to analyze and present the data in a user-friendly manner.

The dataset includes data on unique students, with multiple rows for each student representing their performance in different courses and semesters. The dataset was processed and saved in an Excel file format, which is then used as input for the Ed-Analytics project.

The Ed-Analytics project uses this dataset as its primary data source to generate various visualizations and insights related to student performance. The dataset's size and scope make it ideal for identifying patterns and trends in academic performance across different courses and semesters. The dataset's five columns provide a comprehensive overview of student performance, enabling the Ed-Analytics project to generate detailed reports and visualizations that are relevant to students, teachers, and administrators alike.

Overall, the stud_cd.csv dataset used in the Ed-Analytics project is a rich and comprehensive dataset that provides valuable insights into the academic performance of students at R.J.College. The dataset's size and scope make it ideal for identifying areas for improvement and promoting academic excellence.

Methodology

The Ed-Analytics project follows a robust and data-driven methodology for analyzing academic performance using the latest data analysis techniques and visualization tools. The project's architecture is based on the Django web framework, which provides a scalable and secure platform for developing web applications.

The project's methodology starts by collecting data on student performance from the stud_cd.csv dataset, which contains information on the academic performance of students in the BAMMC course at R.J. College for the years 2019-2022. The dataset has five columns, including student name, subject name, internal marks, external marks, and semester, and a total of 5620 rows. The data is processed and saved in an excel file format, which serves as the primary data source for the Ed-Analytics project.

The project follows a Model-View-Controller (MVC) architecture, with separate modules for data processing, data analysis, and visualization. The data processing module preprocesses the data by cleaning, filtering, and transforming it into a suitable format for analysis. The data analysis module uses various Python libraries, including NumPy, Pandas, Matplotlib, and Seaborn, to analyze student performance data and generate insights into academic performance.

The visualization module uses various interactive and user-friendly visualization tools, including bar graphs, pie charts, and tables, to present the analysis results visually. The interactive interface of the Ed-Analytics project enables users to navigate and explore data quickly, filter and sort data, and customize visualizations to suit their needs.

The Ed-Analytics project also includes separate modules for user management, access control, and authentication, ensuring that user data is secure and protected. The project's methodology leverages advanced security features, such as SSL encryption and two-factor authentication, to ensure that user data is protected from unauthorized access.

Overall, the Ed-Analytics project's methodology uses a data-driven approach to analyze academic performance, leveraging data analysis techniques and visualization tools to provide real-time insights into student performance and facilitate informed decision-making. The project's methodology ensures that user data is secure, accessible, and customizable, providing an efficient and effective platform for academic performance analysis.

The Ed-Analytics project utilizes a robust and scalable web development framework called Django to create a seamless and efficient user experience. The project follows a Model-View-Controller (MVC) architecture, with separate modules for data processing, data analysis, visualization, and user management. The project also includes separate modules for access control and authentication, ensuring that user data is protected.

The Django framework provides a powerful toolkit for building web applications with high security and customization. The project includes several apps within the Django framework that work seamlessly together to create a cohesive user experience. These apps include:

Data Processing: This app is responsible for processing the raw student performance data from the stud_cd.csv dataset and storing it in an excel file. The app also includes data validation and cleaning functions to ensure that the data is accurate and consistent.

Data Analysis: This app is responsible for analyzing the processed data and generating various insights and visualizations related to student performance. The app uses Python libraries, including NumPy, Pandas, Matplotlib, and Seaborn, to generate insights into academic performance. The app also includes various visualizations, including bar graphs, pie charts, and tables, to help users analyze academic performance.

User Management: This app is responsible for managing user roles, access rights, and authentication. The app ensures that only authorized users can access sensitive data and perform specific actions. It includes functions for creating user accounts, assigning roles and access rights, and managing passwords.

Frontend: This app is responsible for rendering the user interface and making it interactive. The app includes HTML templates, CSS stylesheets, and JavaScript code to create a seamless user experience. The app enables users to navigate and explore data quickly, filter and sort data, and customize visualizations to suit their needs.

Challenges

While developing the Ed-Analytics project, we encountered several challenges that required careful consideration and planning. Here are some additional details on the challenges we faced:

Data Quality: To ensure the accuracy and reliability of the visualizations and insights generated by the Ed-Analytics project, we had to overcome several data quality challenges. These included inconsistencies and errors in the data, missing values, and data in different formats. We had to clean and preprocess the data extensively to ensure that it was ready for analysis.

Security: The Ed-Analytics project deals with sensitive student data, and ensuring the security of user data was a top priority. We implemented several security measures, such as access control, authentication, and data encryption, to ensure that user data was protected and secure.

User Interface Design: Another significant challenge we faced was designing an intuitive and user-friendly interface that was easy to navigate and understand. We conducted several rounds of user testing and feedback to ensure that the interface was accessible and easy to use for all users, including students, teachers, and administrators.

Performance: As the dataset used in the project was quite large, ensuring that the application was performant and responsive was a significant challenge. We optimized the data processing and analysis modules to ensure that the application remained fast and responsive even when handling large amounts of data.

Maintenance: As the project is designed to be used over an extended period, ensuring that the application remains up-to-date and functional was a challenge. We established a maintenance plan and update the project regularly to ensure that it remained compatible with the latest versions of the Python libraries and frameworks.

In addition to these challenges, we also had to tackle some issues related to the Django framework and different apps. We had to ensure that the different apps in the project were seamlessly integrated, and the user data was consistent across all the modules. We also had to optimize the database queries to reduce the response time of the application.

Despite these challenges, we were able to develop a robust and scalable platform for analyzing academic performance using data-driven techniques and visualization tools. The Ed-Analytics project provides real-time insights into student performance, enabling informed decision-making and improving educational outcomes.

Code

My Project Ed-Analytics is made in Django framework and uses MVT Architecture and if I incude all the code that is all the templates too than the documentation will be unnecessary longer so I am including only my models and views code to make it a bit brief

Dashboard models.py

```
from django.db import models
# Create your models here.

class Records(models.Model):
    No          = models.IntegerField()
    Name        = models.CharField(max_length = 200 )
    Internal    = models.FloatField(max_length = 200)
    External    = models.FloatField(max_length = 200 )
    Subject     = models.CharField(max_length = 200 )
    Semester    = models.CharField(max_length = 200 )
    Total       = models.FloatField(max_length = 200 )

    class Meta:
        ordering = ['Name']

    def __str__(self):
        return f"{self.Name} {self.Subject} {self.Semester} {self.Total}"
```

Accounts models.py

```
class Contact(models.Model):
    Name        = models.CharField(max_length = 200, blank = True, null = True)
    Email       = models.EmailField(max_length= 254, blank = True, null = True)
    Subject     = models.CharField(max_length = 200, blank = True, null = True)
    Message     = models.TextField(max_length = 400, blank = True, null = True)

    class Meta:
        ordering = ['Name']

    def __str__(self):
        return f"{self.Name} {self.Email} {self.Subject} {self.Message}"
```

Rolerights models.py

```
from django.db import models

from django.contrib.auth.models import User
from django_userforeignkey.models.fields import UserForeignKey

# Create your models here.


class role_master(models.Model):
    rolename = models.CharField("Role Name",max_length=250)
    rolecode = models.CharField("Role Code",max_length=50)
    roledesc = models.CharField("Role Description",max_length=250)
    STATUS_CHOICES = (
        ('Active', 'Active'),
        ('Inactive', 'Inactive'),
        )
    status = models.CharField(max_length=25,choices=STATUS_CHOICES,default='Active')
    created_by = UserForeignKey(auto_user_add=True, verbose_name="The user that is
automatically assigned", related_name="role_master_created_by_user")
    updated_by = UserForeignKey(auto_user=True, verbose_name="The user that is automatically
assigned", related_name="role_master_updated_by_user")

    class Meta:
        db_table = "role_master"
    def __str__(self):
        return "%s " % (self.rolename)



class menu_master1(models.Model):
    menuname = models.CharField("Menu Name",max_length=250)
    menu_parent = models.CharField("Parent Menu",max_length=250)
    menuurl = models.CharField("Menu URL",max_length=50)
    menudesc = models.CharField("Menu Description",max_length=250)
    STATUS_CHOICES = (
        ('Active', 'Active'),
        ('Inactive', 'Inactive'),
        )
    status = models.CharField(max_length=25,choices=STATUS_CHOICES,default='Active')
    menu_parent =models.ForeignKey('self', on_delete=models.CASCADE ,
related_name='parent_menu_master1_fk', default=None, blank=True, null=True)
    created_by = UserForeignKey(auto_user_add=True, verbose_name="The user that is
automatically assigned", related_name="menu_master1_created_by_user", null=True)
    updated_by = UserForeignKey(auto_user=True, verbose_name="The user that is automatically
assigned", related_name="menu_master1_updated_by_user", null=True)
    iconn = models.CharField("Icon", max_length=50, blank=True, null=True, default = 'fa fa-circle')

    class Meta:
        db_table = "menu_master1"
    def __str__(self):
        return "%s " % (self.menuname)
```

```
class role_menu_map(models.Model):
    role_master =models.ForeignKey(role_master, on_delete=models.CASCADE ,
related_name='role_master_fk', default=None)
    menu_master =models.ForeignKey(menu_master1, on_delete=models.CASCADE ,
related_name='menu_master_fk', default=None)
    created_by = UserForeignKey(auto_user_add=True, verbose_name="The user that is
automatically assigned", related_name="role_menu_map_created_by_user")
    updated_by = UserForeignKey(auto_user=True, verbose_name="The user that is automatically
assigned", related_name="role_menu_map_updated_by_user")

    class Meta:
        db_table = "role_menu_map"
        constraints = [models.UniqueConstraint(fields=['role_master','menu_master'],
name="role_master_Unique")]
    def __str__(self):
        return "%s " % (self.role_master)



class user_map(models.Model):
    role_master =models.ForeignKey(role_master, on_delete=models.CASCADE ,
related_name='role1_master_fk', default=None)
    user_master =models.ForeignKey(User, on_delete=models.CASCADE , related_name='user_fk',
default=None)
    # lab_master =models.ForeignKey(lab_master, on_delete=models.CASCADE ,blank=True,
null=True, related_name='lab_fk', default=None)
    created_by = UserForeignKey(auto_user_add=True, verbose_name="The user that is
automatically assigned", related_name="user_map_created_by_user")
    updated_by = UserForeignKey(auto_user=True, verbose_name="The user that is automatically
assigned", related_name="user_map_updated_by_user")

    class Meta:
        db_table = "user_map"

    def __str__(self):
        return "%s  " % (self.role_master)
```

Dashboard views.py:

```
# Create your views here.
from django.http import HttpResponse
from django.shortcuts import render,redirect, get_object_or_404
import pandas as pd
import numpy as np
from .models import Records, Organisation_master, Student, Teacher,Class
import plotly.express as px
import dash
from dash import dcc, html
from django_plotly_dash import DjangoDash
import dash
from dash import dcc
import dash_bootstrap_components as dbc
from dash.dependencies import Input, Output
import plotly.express as px
import pandas as pd
from django.contrib.auth import get_user_model
from django.contrib import messages
from accounts.decorators import allowed_users
from django.contrib.auth.decorators import login_required

@login_required
#@allowed_users(allowed_roles=['Admin'])
def upload_marks(request):
    if request.method == "POST":
        myfile = request.FILES.get('doc')
        df = pd.read_csv(myfile)
        df.rename(columns = {'Roll No':'No','Student Name':'Name'}, inplace = True)
        try:
            cols = ['IA', 'SEE']
            df[cols] = df[cols].apply(pd.to_numeric, errors='coerce')
            df.fillna(0,inplace = True)
            df.describe(include="all")
            df['Total'] = df['IA']+df['SEE']
            vv = df.Name.value_counts().loc[lambda x: x>35].reset_index()['index']
            df2 = df[df['Name'].isin(vv)]
            list1 = df2.values.tolist()
            print(list1)
            for i in list1:
                if Records.objects.filter(No=i[0],Name = i[1], Internal = i[2], External = i[3], Subject = i[4],
Semester = i[5], Total = i[6] ).exists():
                    pass
                else:
                    h = Records(No=i[0],Name = i[1], Internal = i[2], External = i[3], Subject = i[4], Semester =
i[5], Total = i[6])
                    h.save()
        except:
            messages.warning(request,'The required excel is not provided. Please resubmit the proper
excel.')
            return redirect('dashboard:upload_marks')
    return render(request,'upload_marks.html')



@login_required
#@allowed_users(allowed_roles=['Admin'])
def index(request):
    stu = Records.objects.order_by().values('Name').distinct().count()
    sub = Records.objects.order_by().values('Subject').distinct().count()
    all_data = Records.objects.order_by().values('Name').distinct()
    item = Records.objects.all().values('id','No','Name','Internal','External','Subject','Semester','Total')
    df = pd.DataFrame(item)
    a =  df["Internal"].mean()
    meani = round(a, 2)
    b = df['External'].mean()
    meanx = round(b, 2)
    return render(request,'front.html',{'Student':stu,'Subjects':sub,'all':all_data,'reA':meani, 'reB':
meanx})




@login_required
#@allowed_users(allowed_roles=['Admin'])
def students(request):
    stu = Records.objects.order_by().values('Name').distinct().count()
    sub = Records.objects.order_by().values('Subject').distinct().count()
    all_data = Records.objects.order_by().values('Name').distinct()
    all_dat = Records.objects.order_by().values().distinct()
    return render(request, 'student.html',{'Student':stu,'Subjects':sub,'all':all_data})


@login_required
#@allowed_users(allowed_roles=['Admin'])
def stud_plot(request, pk):
    print(pk)
    item =
Records.objects.all().values('id','No','Name','Internal','External','Subject','Semester','Total').filter(Na
me = pk)
    z = pd.DataFrame(item)
    print(z.head())
    a =  z["Internal"].mean()
    meani = round(a, 2)
    b = z['External'].mean()
    meanx = round(b, 2)
    fig_graph = px.bar(z, x = "Semester", y = "Total", color = "Internal", text = "Subject",  title="Marks
scored with respect to Internal marks",hover_data=['Subject', 'External'])
    fig_graph1 = px.bar(z, x = "Semester", y = "Total", color = "External", text = "Subject",title="Marks
scored with respect to External marks",hover_data=['Subject', 'External','Internal'])
    fig1=fig_graph.to_html(full_html=True,config={'displayModeBar':False})
    fig2=fig_graph1.to_html(full_html=True,config={'displayModeBar':False})
    return render(request,'plot.html',{'bar':fig1,'int':meani,'ext':meanx,'bar1':fig2})


@login_required
#@allowed_users(allowed_roles=['Admin'])
def only_stud(request):
    us = 'GAIKWAD SHUBHANGI MANSINGH SHEETAL'
    item =
Records.objects.all().values('id','No','Name','Internal','External','Subject','Semester','Total').filter(Na
me = us)
```

```
   z = pd.DataFrame(item)
   print(z.head())
   a =  z["Internal"].mean()
   meani = round(a, 2)
   b = z['External'].mean()
   meanx = round(b, 2)
   fig_graph = px.bar(z, x = "Semester", y = "Total", color = "Internal", text = "Subject", title="Marks
scored with respect to Internal marks",hover_data=['Subject', 'External'])
   fig_graph1 = px.bar(z, x = "Semester", y = "Total", color = "External", text = "Subject", title="Marks
scored with respect to External marks",hover_data=['Subject', 'External','Internal'])
   fig1=fig_graph.to_html(full_html=True,config={'displayModeBar':False})
   fig2=fig_graph1.to_html(full_html=True,config={'displayModeBar':False})
   return render(request,'plot.html',{'bar':fig1,'int':meani,'ext':meanx,'bar1':fig2})




item = Records.objects.values('id','No','Name','Internal','External','Subject','Semester','Total')
df2 = pd.DataFrame(item)


r = 'SEM_VI'
z4 = df2.query('Semester == @r')
figg = px.pie(z4, names = "Subject", values = "Total",height = 500, width = 1000, title="Marks
Distribution According to Subjects in 6th Semester")





app = DjangoDash('SimpleExample')
app.layout = dbc.Container([
   dbc.Card([
         dbc.Button('⍰', id='back-button', outline=True, size="sm",
                   className='mt-2 ml-2 col-1', style={'display': 'none'}),
         dbc.Row(
            dcc.Graph(
                   id='graph',
                   figure=figg
               ), justify='center'
         )
   ], className='mt-3')
])

#Callback
@app.callback(
   Output(component_id = 'graph',component_property =  'figure'),
   Output(component_id = 'back-button',component_property =  'style'), #to hide/unhide the back
button
   Input(component_id = 'graph', component_property = 'clickData'),    #for getting the vendor name
from graph
   Input(component_id = 'back-button', component_property = 'n_clicks')
)



def drilldown(click_data,n_clicks):
    ctx = dash.callback_context
    if len(ctx.triggered) !=0:
        trigger_id = ctx.triggered[0]["prop_id"].split(".")[0]
        if trigger_id == 'graph':
            s = click_data['points'][0]['label']
            print(s)
            if s in z4.Subject.unique():
                x = 'DIGITAL MEDIA'
                vendor_sales_df =z4.query('Subject == @s')
                fig = px.bar(vendor_sales_df, x = "Name", y = "Total", color =
"External",hover_data=['Internal', 'External'], height = 600, width = 1000)
                return fig, {'display':'block'}     #returning the fig and unhiding the back button
            else:
                return figg, {'display':'none'}
    return figg, {'display':'block'}




@login_required
#@allowed_users(allowed_roles=['Admin'])
def trial(request):

    menu_list = Records.objects.order_by().values('Subject','Semester').distinct()
    print(len(menu_list))
    if request.method == 'GET':
        menuname =request.GET.get('menuname')

        if request.GET.get('menuname') == '' :

            a = Records.objects.order_by().values('Subject','Semester').distinct()

        else:

             a =
Records.objects.order_by().values('Subject','Semester').filter(Subject=menuname).distinct()
    return render(request,'trial_forms.html',{'menu_list':menu_list, 'ab':a})

@login_required
#@allowed_users(allowed_roles=['Admin'])
def teacher_specific_student(request,ak,bk):
    item =
Records.objects.all().values('id','No','Name','Internal','External','Subject','Semester','Total').filter(Na
me = ak)
    name = ak
    print(ak)
    print(bk)
    z = pd.DataFrame(item)
    r = bk
    df = z.query('Subject == @r')
    print(df)
    a =  df["Internal"].mean()
    meani = round(a, 2)
    b = df['External'].mean()
    meanx = round(b, 2)
    item2 = Records.objects.all().values('id','No','Name','Internal','External','Subject','Semester','Total')
    for_avg = pd.DataFrame(item2)
    a =  for_avg["Internal"].mean()
    meaniA = round(a, 2)
    b = for_avg['External'].mean()
    meanxA = round(b, 2)
    fig_graph = px.bar(df, x = "Semester", y = "Total", color = "Internal", text = "Subject", title="Marks 
scored with respect to Internal marks",hover_data=['Subject', 'External'])
    fig_graph1 = px.bar(df, x = "Semester", y = "Total", color = "External", text = "Subject", 
title="Marks scored with respect to External marks",hover_data=['Subject', 'External','Internal'])
    fig1=fig_graph.to_html(full_html=True,config={'displayModeBar':False})
    fig2=fig_graph1.to_html(full_html=True,config={'displayModeBar':False})
    return render(request, 
'teacher_specific_student.html',{'bar':fig1,'int':meani,'ext':meanx,'bar1':fig2, 'reA':meaniA, 'reB': 
meanxA,'nam':name,'sub':r})


@login_required
#@allowed_users(allowed_roles=['Admin'])
def teachers_view_dynamic(request,pk,jk):
    print(pk)
    print(jk)
    item = 
Records.objects.all().values('id','No','Name','Internal','External','Subject','Semester','Total').filter(Sem
ester = jk)
    z = pd.DataFrame(item).sort_values('Total')
    r = pk
    df = z.query('Subject == @r')
    top = Records.objects.values().order_by('-Total').filter(Semester = jk,Subject = r)[0:5]
    low = Records.objects.values().order_by('Total').filter(Semester = jk,Subject = r)[0:5]
    # print(df.head())
    print(low)
    all_data = Records.objects.order_by().values().distinct()
    a =  df["Internal"].mean()
    meani = round(a, 2)
    b = df['External'].mean()
    meanx = round(b, 2)
    print(df.head())
    figg1 = px.bar(df,x = 'Name',y = 'Total',color = 'External',text_auto='.2s',hover_data=['Internal', 
'External'],height=600, width = 1000)
    figg1.update_traces(textfont_size=12, textangle=0, textposition="outside", cliponaxis=False)
    figg=figg1.to_html(full_html=True,config={'displayModeBar':False})
```

```
   return
render(request,'teacher_dynamic.html',{'graphh':figg,'all':all_data,'int':meani,'ext':meanx,'sub':r,'lo':
low,'to':top})
```

Accounts views.py

```
from django.shortcuts import render

# Create your views here.
from django.shortcuts import render,redirect,HttpResponse
from django.contrib.auth.models import User
from django.contrib import messages
from django.views.decorators.cache import cache_control
from django.contrib import auth
from django.contrib.auth.decorators import login_required
import pandas as pd
from django.db.models import Sum
import plotly.express as px
from django.db import connection
from django.contrib.auth.models import User,Group
from rolerights.models import role_master, menu_master1, role_menu_map, user_map
from dashboard.models import Records
import json
from django.http import JsonResponse
import numpy as np




from .models import Contact

def main_page(request):
   if request.method == 'POST':
      name =request.POST.get('na')
      email =request.POST.get('em')
      subject =request.POST.get('su')
      message =request.POST.get('me')
      print(name)
      if Contact.objects.filter(Name = name, Email = email, Subject = subject, Message =
message).exists():
         pass
      else:
         z = Contact(Name = name, Email = email, Subject = subject, Message = message)
         z.save()
      return render(request, 'index1.html')
   return render(request, 'index1.html')

def about(request):
   return render(request, 'about.html')
def service(request):
    return render(request, 'service.html')



@cache_control(no_cache=True, must_revalidate=True, no_store=True)
def login(request):
    if request.method =='POST':
        user=
auth.authenticate(username=request.POST['username'],password=request.POST['password'])

        if user is not None:
            auth.login(request,user)
            if  request.user.is_active:
                userid = request.user.id
                roleid = user_map.objects.filter(user_master=userid).values('role_master')
                # allowed_roles = ['Student']
                rolename = role_master.objects.filter(id=roleid[0]['role_master']).values('rolename')
                # print(rolename)
                if request.user.is_superuser:
                    return redirect('dashboard:dash')
                # if rolename[0]['rolename'] == 'Student':
                #     print(rolename[0]['rolename'])
                #     return redirect('dashboard:only_stud')
                elif rolename[0]['rolename'] == 'Teacher':
                    a = 'CA'
                    b = 'SEM_I'
                    return redirect('dashboard:teacher_dynamic',pk=a,jk=b)
                else:
                    return redirect('accounts:base')
        else:
            messages.error(request,'User Name or Password is Incorrect')
            return redirect('accounts:login')

    else:
        return render(request, 'alt_login.html')


@login_required
#@allowed_users(allowed_roles=['Admin'])
def index2(request):
    stu = Records.objects.order_by().values('Name').distinct().count()
    sub = Records.objects.order_by().values('Subject').distinct().count()
    all_data = Records.objects.order_by().values('Name').distinct()
    item = Records.objects.all().values('id','No','Name','Internal','External','Subject','Semester','Total')
    df = pd.DataFrame(item)
    a =  df["Internal"].mean()
    meani = round(a, 2)
    b = df['External'].mean()
    meanx = round(b, 2)
    return render(request,'base_login.html',{'Student':stu,'Subjects':sub,'all':all_data,'reA':meani,
'reB': meanx})
```

```
@cache_control(no_cache=True, must_revalidate=True, no_store=True)
@login_required
def logout(request):
    auth.logout(request)
    return redirect('accounts:login')


@login_required
def lab_menu(request):
    ui = request.user.id
    ro = user_map.objects.filter(user_master_id = ui).values_list('role_master_id',flat=True)
    # print(ro)
    ro1 = user_map.objects.all().values()
    rmid = list(ro)
    mo = role_menu_map.objects.filter(role_master_id =
rmid[0]).values_list('menu_master_id',flat=True)

    # result_data = menu_master1.objects.filter(id__in =  mo).values()
    # print(result_data)
    result_data = menu_master1.objects.filter(id__in =  mo, status = 'Active').values()
    result_data1 = menu_master1.objects.all().values()
    # print(result_data1)
    dff =
pd.DataFrame(list(result_data1),columns=['id','menu_parent_id','menuname','menuurl','iconn'])
    df =
pd.DataFrame(list(result_data),columns=['id','menu_parent_id','menuname','menuurl','iconn'])
    df['depth'] = np.where(df['menuurl'] == '#', 1, 2)
    df = df.sort_values(by=['depth'])
    # print(df)
    return JsonResponse(json.dumps(df.to_dict(orient='records')),safe=False)
```

Rolerights views.py

```
from django.shortcuts import render
from django.http import HttpResponse
from .forms import roleForm, menuForm, SearchContactForm, role_menu_mapForm,
user_mapForm
from .models import role_master,menu_master1, role_menu_map, user_map
from django.db import connection
from django.contrib import messages
from django.shortcuts import render,redirect, get_object_or_404
from django.contrib.auth.models import User
from django.http import JsonResponse
import json
from accounts.decorators import allowed_users
from django.contrib.auth.decorators import login_required



@login_required
@allowed_users(allowed_roles=['Admin'])
def menu(request,success=None):
   menu_list =menu_master1.objects.all().values()
   form = SearchContactForm()
   if request.method == 'GET':
      menuname =request.GET.get('menuname')

      if request.GET.get('menuname') == '' :

         a = menu_master1.objects.all()

      else:
          a = menu_master1.objects.filter(menuname=menuname)
         #  print(a)
   return render(request,'menu_search.html',{'form': form,'ab':a, 'menu_list':menu_list})



@login_required
@allowed_users(allowed_roles=['Admin'])
def menu_add(request, template_name='menu_add.html'):
   status_list = {"Active": "Active", "Inactive": "Inactive"}
   ic = {'Home':'fa fa-home','Teacher':'fa fa-users' ,'Link':'fa fa-link', 'Mail':'fa fa-envelope-o',
'Circle':'fa fa-circle', 'Clock':'fa fa-clock-o', 'User':'fa fa-user', 'Task':'fa fa-tasks', 'Money':'fa fa-
money', 'Upload':'fa fa-upload', 'Database':'fa fa-database'}
   # print(ic)
   menu_list =menu_master1.objects.all().values()
   if request.method == 'POST':
      st = request.POST.get('status')
      ico = request.POST.get('icon_name')
      # print(st)
      # print(ico)
      form = menuForm(request.POST, request.FILES)

      if form.is_valid():

         menu1= form.cleaned_data['menuname']
         menu2= form.cleaned_data['menuurl']
         menu3= form.cleaned_data['menudesc']
         menu4= form.cleaned_data['status']
         menu5= form.cleaned_data['menu_parent']
         menu6= form.cleaned_data['iconn']
         # print(form.errors)
         if
menu_master1.objects.filter(menuname=menu1,menuurl=menu2,menudesc=menu3,status=menu4
,menu_parent=menu5, iconn=menu6).exists():
```

```
            messages.warning(request,'Menu already exists')
            form = menuForm()
            return render(request, template_name,{'form':form,'status_list':status_list})
         else:
            # messages.warning(request, 'Saved successfully')
            form.save()
            return redirect('rolerights:menu_search')
   else:
      form =menuForm()
      return render(request, template_name, {'form':
form,'status_list':status_list,'menu_list':menu_list, 'ic':ic})



@login_required
@allowed_users(allowed_roles=['Admin'])
def menu_edit(request, pk, template_name='menu_edit.html'):
   menu_list =menu_master1.objects.all().values()
   status_list = {"Active": "Active", "Inactive": "Inactive"}
   post = get_object_or_404(menu_master1, pk=pk)
   form = menuForm(request.POST or None, instance=post)
   ic = {'Home':'fa fa-home', 'Link':'fa fa-link', 'Mail':'fa fa-envelope-o', 'Circle':'fa fa-circle', 'Clock':'fa
fa-clock-o', 'User':'fa fa-user', 'Task':'fa fa-tasks', 'Money':'fa fa-money', 'Upload':'fa fa-upload',
'Database':'fa fa-database'}
   # print(ic)
   if form.is_valid():
         print('valid')
         me1= form.cleaned_data['menuname']
         me2= form.cleaned_data['menuurl']
         me3= form.cleaned_data['menudesc']
         me4= form.cleaned_data['status']
         me5= form.cleaned_data['menu_parent']
         me6= form.cleaned_data['iconn']
         if
menu_master1.objects.filter(menuname=me1,menuurl=me2,menudesc=me3,status=me4,menu_pa
rent=me5, iconn=me6).exists():
            messages.warning(request,'Menu already exists')
            form = menuForm()
            return redirect('rolerights:menu_search')
         else:
            # messages.warning(request, 'Saved successfully')
            form.save()
            return redirect('rolerights:menu_search')
   else:
      form =menuForm(request.POST or None, instance=post)
      return render(request, template_name, {'form':
form,'status_list':status_list,'menu_list':menu_list, 'ic':ic})
```

```
@login_required
@allowed_users(allowed_roles=['Admin'])
def role_search(request,success=None):
   bgg = role_master.objects.all()
   form = SearchContactForm()
   if request.method == 'GET':
      rolename =request.GET.get('rolename')
      if request.GET.get('rolename') == '' :

         b = role_master.objects.all()

      else:
         b = role_master.objects.filter(rolename = rolename)

   return render(request,'role_search.html',{'form': form,'ba':b, 'bgg':bgg})




@login_required
@allowed_users(allowed_roles=['Admin'])
def role_add(request, template_name='role_add.html'):
   status_list = {"Active": "Active", "Inactive": "Inactive"}
   if request.method == 'POST':
      form = roleForm(request.POST, request.FILES )

      if form.is_valid():
         role1= form.cleaned_data['rolename']
         role2= form.cleaned_data['rolecode']
         role3= form.cleaned_data['roledesc']
         role4= form.cleaned_data['status']

         if role_master.objects.filter(rolename=role1,rolecode=role2,roledesc=role3,
status=role4).exists():
            messages.warning(request,'Role already exists')
            form = roleForm()
            return render(request, template_name,{'form':form,'status_list':status_list})
         else:
            messages.success(request, 'Saved successfully')
            form.save()
            return redirect('rolerights:role_search')
   else:
      form =roleForm()
      return render(request, template_name, {'form': form,'status_list':status_list})



@login_required
@allowed_users(allowed_roles=['Admin'])
```

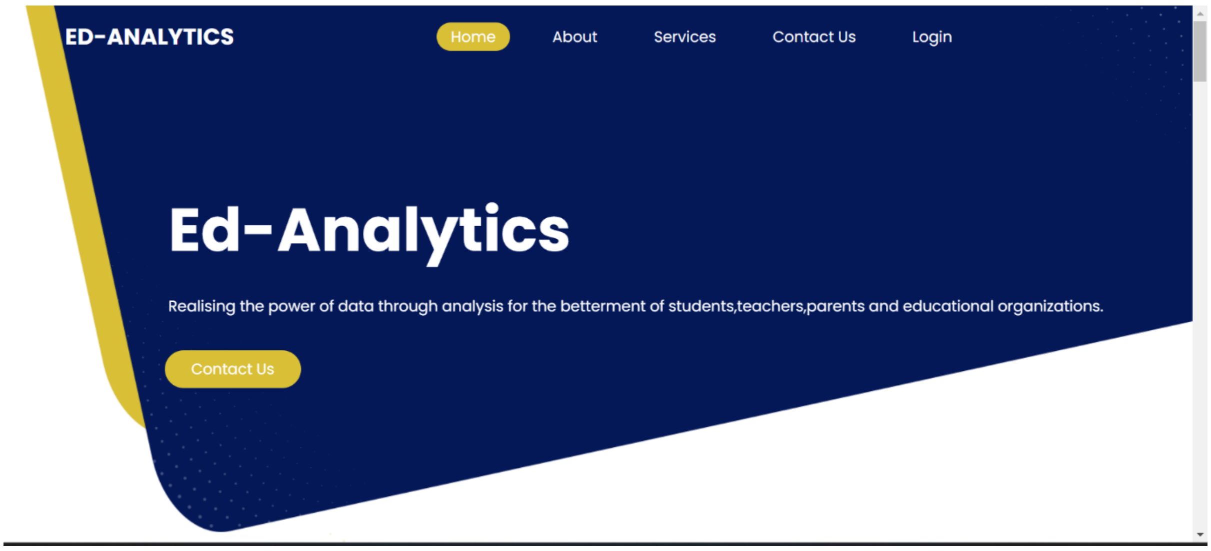
```
def role_edit(request, pk, template_name='role_edit.html'):
   status_list = {"Active": "Active", "Inactive": "Inactive"}
   post = get_object_or_404(role_master, pk=pk)
   form = roleForm(request.POST or None, instance=post)
   if form.is_valid():
         # print('valid')
         role1= form.cleaned_data['rolename']
         role2= form.cleaned_data['rolecode']
         role3= form.cleaned_data['roledesc']
         role4= form.cleaned_data['status']

         if role_master.objects.filter(rolename=role1,rolecode=role2,roledesc=role3, status=
role4).exists():
            messages.warning(request,'Role already exists')
            form = roleForm(request.POST or None, instance=post)
            return redirect('rolerights:role_search')
         else:
            # messages.warning(request, 'Saved successfully')
            success = True
            form.save()
            return redirect('rolerights:role_search')
   else:
      form =roleForm(request.POST or None, instance=post)
      return render(request, template_name, {'form': form,'status_list':status_list})



@login_required
@allowed_users(allowed_roles=['Admin'])
def role_menu_map_add(request,template_name='role_map_add.html'):
   global sub_menu
   role_list =role_master.objects.all().filter(status = 'Active').values()
   menu_list =menu_master1.objects.all().filter(status = 'Active').values()
   main_menu = menu_master1.objects.all().filter(status = 'Active', menuurl = '#').values()
   if 'mainmenu' in request.GET:
      print(request.GET.getlist('menu_master_m'))
      sub_menu = menu_master1.objects.filter(id = request.GET.get('mainmenu'), status =
'Active').values('id')
      # print(sub_menu)
      a = menu_master1.objects.filter(menu_parent_id__in= sub_menu).values_list('menuname','id')
      return JsonResponse(list(a),safe=False)
   # print(main_menu)
   if request.method == 'POST':
      form = role_menu_mapForm(request.POST, request.FILES)
      su = sub_menu[0]['id']
      # print(su)
      f1= request.POST.get('role_master')
      f2=  request.POST.getlist('menu_master')
      f2.append(su)
      print(f2)
      for i in f2:
          if role_menu_map.objects.filter(role_master_id=f1, menu_master_id=i).exists():
            print('heloo')
            messages.warning(request,'Mapping already exists')
            form = role_menu_mapForm()
          else:
            ff = role_menu_map(role_master_id=f1, menu_master_id=i)
            ff.save()
      return redirect('rolerights:role_menu_map_search')
    else:
      form =role_menu_mapForm()
      return render(request, template_name, {'form': form,'role_list':role_list,'menu_list':menu_list,
'main_menu':main_menu})



@login_required
@allowed_users(allowed_roles=['Admin'])
def role_menu_map_search(request,template_name='role_menu_map_search.html'):
  role_list =role_master.objects.all().values()
  form =role_menu_mapForm()
  results =[]
  if 'rolename' in request.GET:
    rolename =request.GET.get('rolename')
    print(rolename)
    if rolename == '':
      act = role_master.objects.all().filter(status = 'Active').values('id')
      # print(act)
      print('hi')
      results = role_menu_map.objects.all().filter(role_master_id__in =
act).values('role_master_id__rolename','role_master_id').distinct()
      # print(results)
      r1 = role_menu_map.objects.all().values()
    else:
      act = role_master.objects.all().filter(status = 'Active').values('id')
      results = role_menu_map.objects.all().filter(role_master_id__in = act, role_master_id =
rolename).values('role_master_id__rolename','role_master_id').distinct()
      print('hellp')
      print(results)
      # results = role_menu_map.objects.all().values().distinct()
      # results = role_menu_map.objects.all().filter(role_master_id__in = act,
role_master_id__rolename =
rolename).values('role_master_id__rolename','role_master_id').distinct()
      # results = role_menu_map.objects.filter(role_master=rolename)
  return render(request, template_name, {'form': form,'role_list':role_list,'results':results})




@login_required
@allowed_users(allowed_roles=['Admin'])
def role_menu_map_edit(request, role_master_id):
  roleid = role_master_id
  roles = role_master.objects.all().filter(id = roleid).values()
  r2 = role_menu_map.objects.filter(role_master_id =
role_master_id).values('menu_master_id','role_master_id__rolename','role_master_id')
  main_menu = menu_master1.objects.all().filter(status = 'Active', menuurl = '#').values()
  menu_list = role_menu_map.objects.values('id','menu_master_id__menuname','role_master_id',
'menu_master_id')
  # if 'mainmenu' in request.GET:
  #     sub_menu = menu_master1.objects.filter(id = request.GET.get('mainmenu'), status =
'Active').values('id')
  #     # print(sub_menu)
  #     hjj =list( role_menu_map.objects.filter(role_master_id =
role_master_id).values_list('menu_master_id__menuname'))
  #     a = list(menu_master1.objects.filter(menu_parent_id__in=
sub_menu,).values_list('menuname','id'))
  #     print(hjj)
  #     print(a)
  #     return JsonResponse({'a':a,'hjj':hjj},safe=False)
  hj =list( role_menu_map.objects.filter(role_master_id =
role_master_id).values_list('menu_master_id'))
  b =[]
  for  i in range(len(hj)):
     for j in hj[i]:
      b.append(j)
  mk = menu_master1.objects.all().filter(status = 'Active').values()
  print(mk)
  if request.method == "POST":
     ro = request.POST.get('role_master')
     menus = request.POST.getlist('menu_m')
     # print(ro)
     # print(menus)
     al = role_menu_map.objects.filter(role_master_id = ro).values_list('menu_master_id')
     # for i in menus:
     if role_menu_map.objects.filter(role_master_id = ro).exists():
      role_menu_map.objects.all().filter(role_master_id = ro).delete()
      # print('deleted')
      for i in menus:
            zz = role_menu_map(role_master_id = ro, menu_master_id = i )
            zz.save()
            # print(role_menu_map.objects.filter(role_master_id = ro).values())
      return redirect('rolerights:role_menu_map_search')
  return render(request, 'role_menu_map_edit.html', {'menu_list':menu_list, 'roles':roles ,
'roleid':roleid ,'r2': r2, 'mk':mk, 'hj':b, 'main_menu':main_menu})


@login_required
@allowed_users(allowed_roles=['Admin'])
def user_map_add(request,template_name='user_map_add.html'):
  role_list =role_master.objects.all().filter(status = 'Active').values()
  User_list =User.objects.all().values()
  r = user_map.objects.all().values()
```

```
    # print(r)
    if request.method == 'POST':
        form = user_mapForm(request.POST, request.FILES)
        # print(form.errors)

        if form.is_valid():

            r1= form.cleaned_data['role_master']
            r2= form.cleaned_data['user_master']
            if user_map.objects.filter(role_master=r1,user_master=r2).exists():
                messages.warning(request,'Mapping already exists')
                form = user_mapForm()
                return render(request, template_name,{'form':form,'role_list':role_list,'User_list':User_list})
            else:
                # messages.warning(request, 'Saved successfully')
                form.save()
                return redirect('rolerights:user_map_search')
    else:
        form =user_mapForm()
        return render(request, template_name, {'form':form,'role_list':role_list,'User_list':User_list})



@login_required
@allowed_users(allowed_roles=['Admin'])
def user_map_search(request,template_name='user_map_search.html'):
    # lab_list =lab_master.objects.all().values()
    use_list =user_map.objects.all().values()
    role_list =role_master.objects.all().values()
    # print(use_list)
    form =user_mapForm()
    results =[]
    if 'rolename' in request.GET:
        form = user_mapForm(request.GET)
        rn =request.GET.get('rolename')
        if rn == '0':
            results = user_map.objects.all()
        else:
            r1 = user_map.objects.all().values()
            ro = request.GET.get('rolename')
            results = user_map.objects.filter(role_master_id = ro)
    return render(request, template_name, {'form': form, 'role_list':role_list, 'results':results})


@login_required
@allowed_users(allowed_roles=['Admin'])
def user_map_edit(request, pk, template_name='user_map_edit.html'):
    role_list =role_master.objects.all().filter(status = 'Active').values()
    User_list =User.objects.all().values()
    post = get_object_or_404(user_map, pk=pk)
    form = user_mapForm(request.POST or None, instance=post)
```

```
if form.is_valid():
   r1= form.cleaned_data['role_master']
   r2= form.cleaned_data['user_master']
   if user_map.objects.filter(role_master=r1,user_master=r2).exists():
      messages.warning(request, 'Mapping exists')
      form = menuForm()
      return render(request, template_name,{'form':form,'role_list':role_list,'User_list':User_list})
   else:
      success = True
      form.save()
      return redirect('rolerights:user_map_search')
return render(request, template_name, {'form': form,'role_list':role_list,'User_list':User_list})
```

Output Images

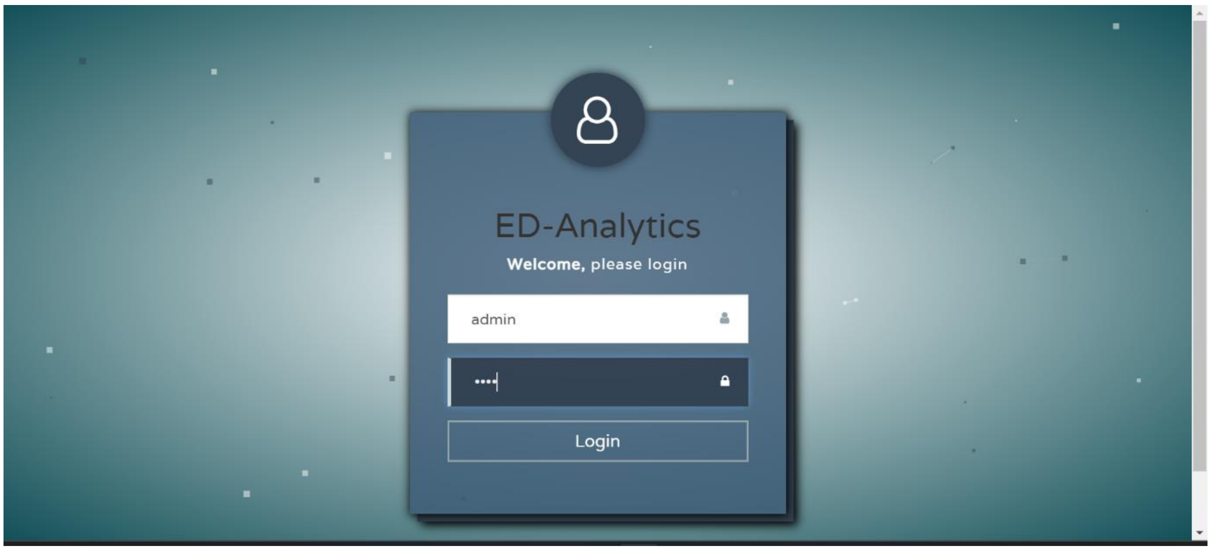
My Front Page



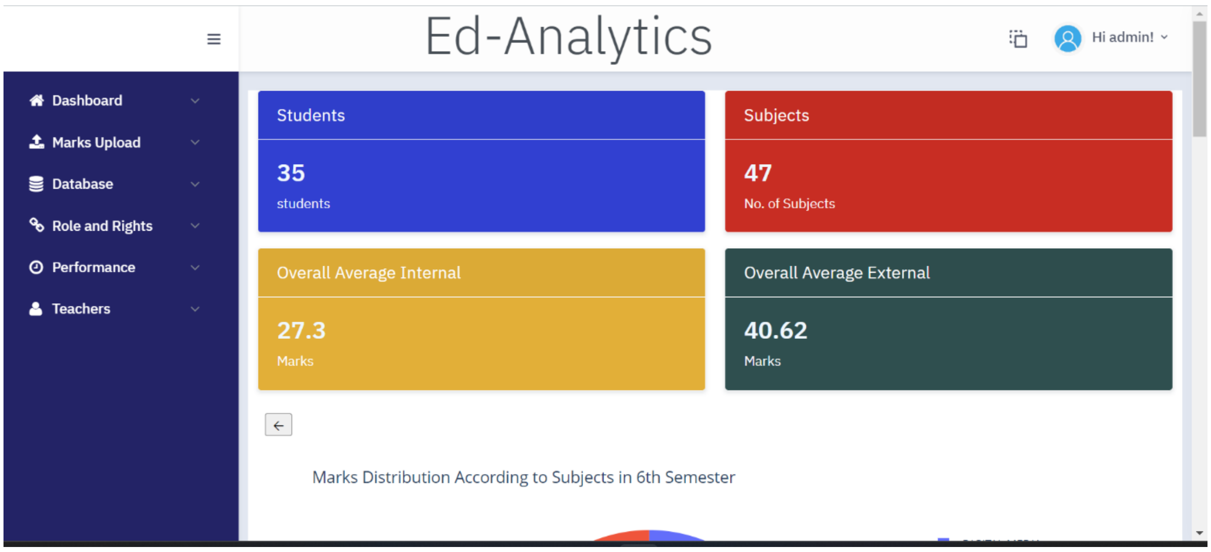
Ed-Analytics

Realising the power of data through analysis for the betterment of students,teachers,parents and educational organizations.

My Login Page



Dashboard Page



Students
35
students

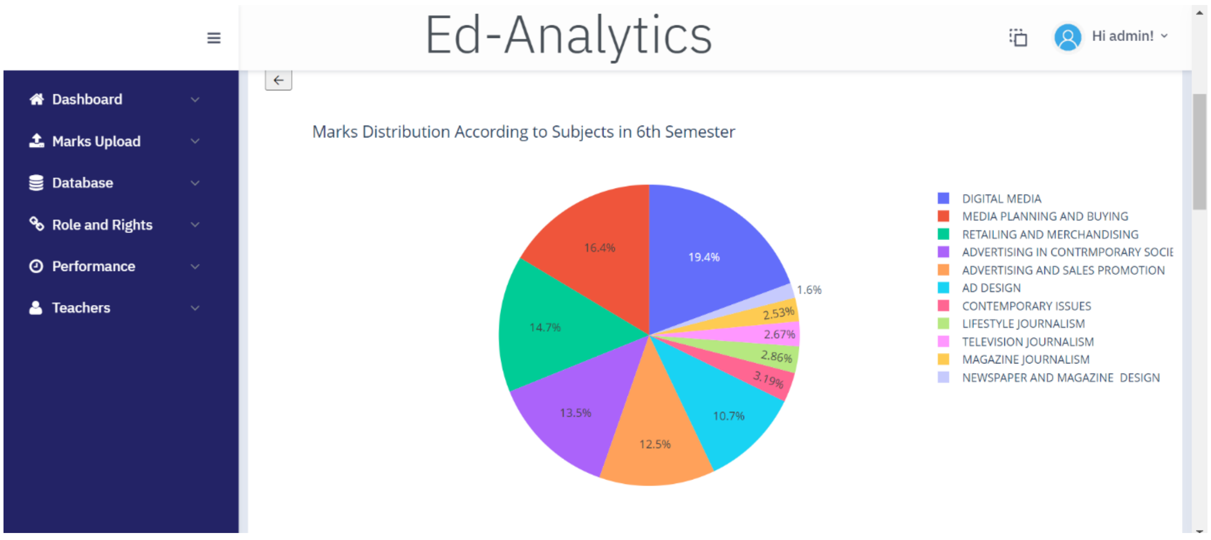
Subjects
47
No. of Subjects

Overall Average Internal
27.3
Marks

Overall Average External
40.62
Marks

Marks Distribution According to Subjects in 6th Semester

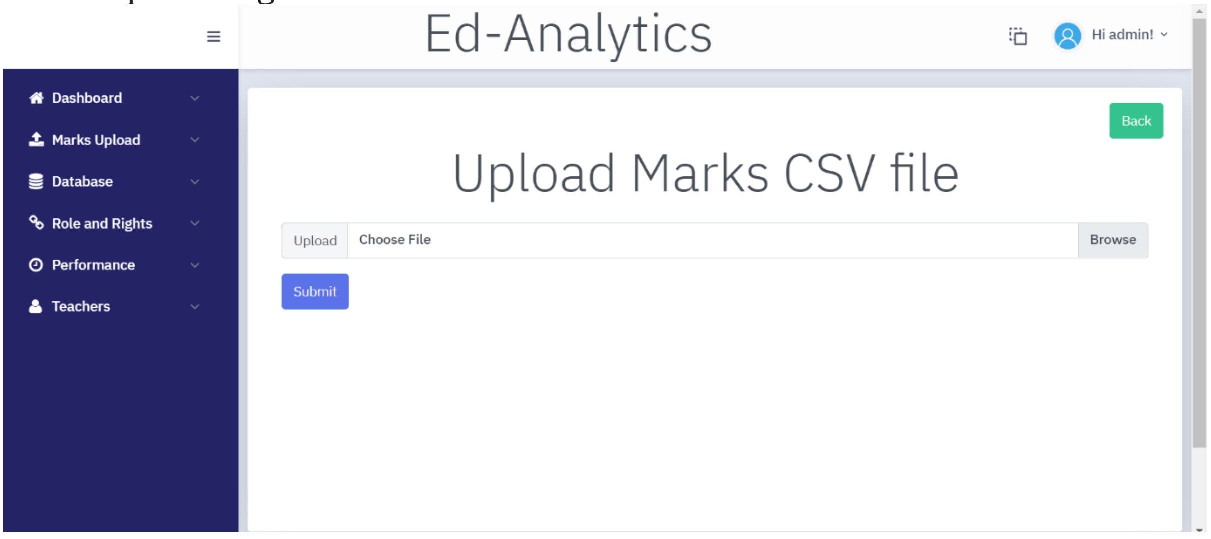
Pie Chart



Marks Distribution According to Subjects in 6th Semester

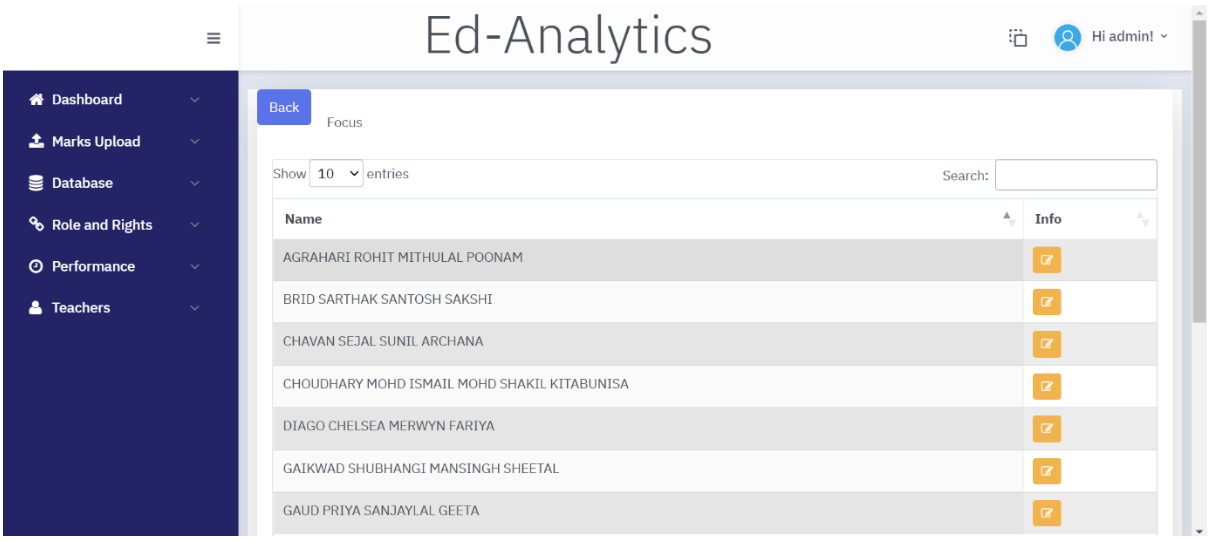
🟦	DIGITAL MEDIA	19.4%
🟥	MEDIA PLANNING AND BUYING	16.4%
🟩	RETAILING AND MERCHANDISING	14.7%
🟪	ADVERTISING IN CONTRMPORARY SOCIE	13.5%
🟧	ADVERTISING AND SALES PROMOTION	12.5%
🟦	AD DESIGN	10.7%
🟥	CONTEMPORARY ISSUES	3.19%
🟩	LIFESTYLE JOURNALISM	2.86%
🟪	TELEVISION JOURNALISM	2.67%
🟨	MAGAZINE JOURNALISM	2.53%
⬜	NEWSPAPER AND MAGAZINE DESIGN	1.6%

Marks upload Page



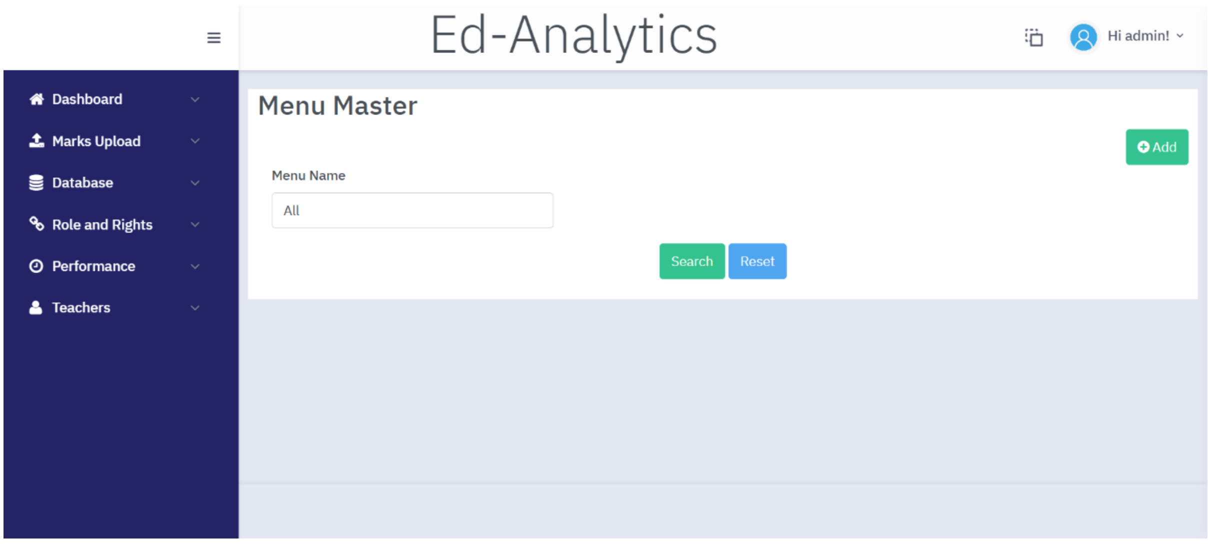
Upload Marks CSV file

Student Database Page



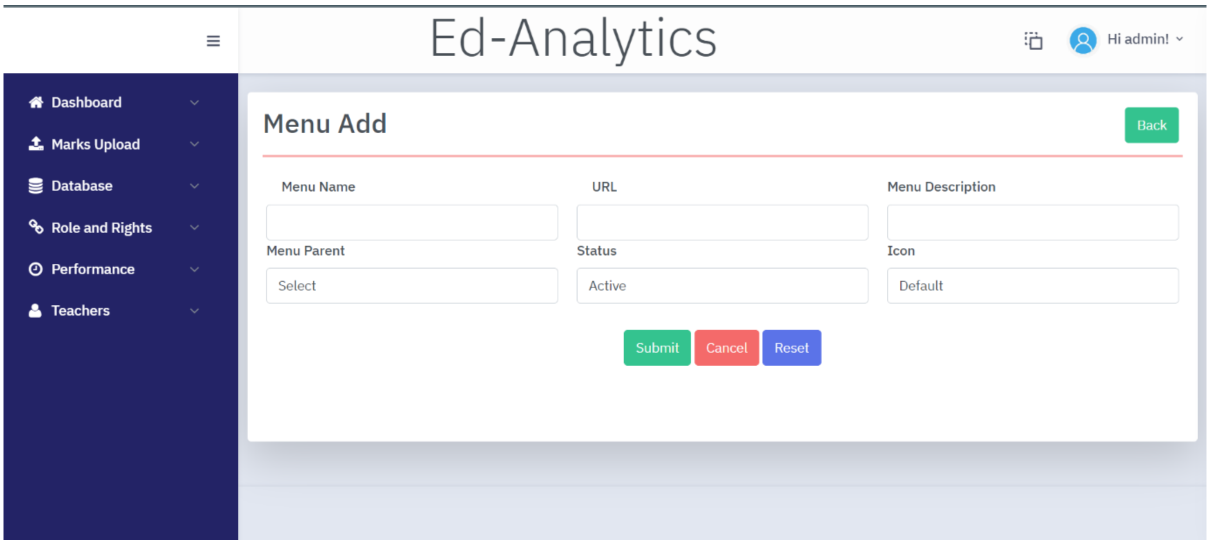
Name	Info
AGRAHARI ROHIT MITHULAL POONAM	
BRID SARTHAK SANTOSH SAKSHI	
CHAVAN SEJAL SUNIL ARCHANA	
CHOUDHARY MOHD ISMAIL MOHD SHAKIL KITABUNISA	
DIAGO CHELSEA MERWYN FARIYA	
GAIKWAD SHUBHANGI MANSINGH SHEETAL	
GAUD PRIYA SANJAYLAL GEETA	

Menu Master Page



Menu Name: All

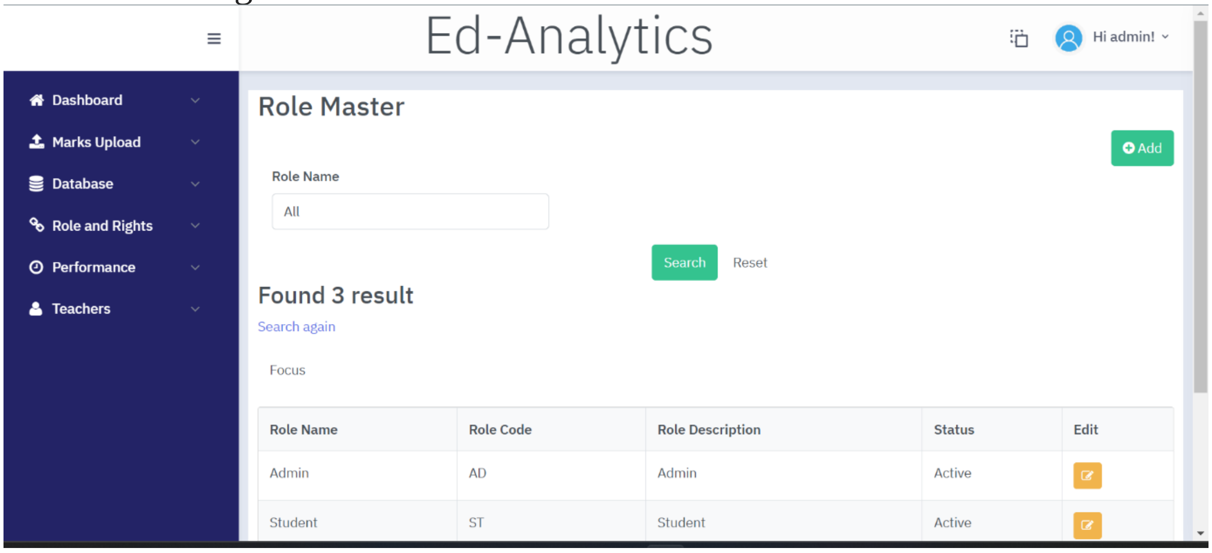
Menu add Page



Menu Add

Menu Name	URL	Menu Description
Menu Parent	Status	Icon
Select	Active	Default

Role Master Page



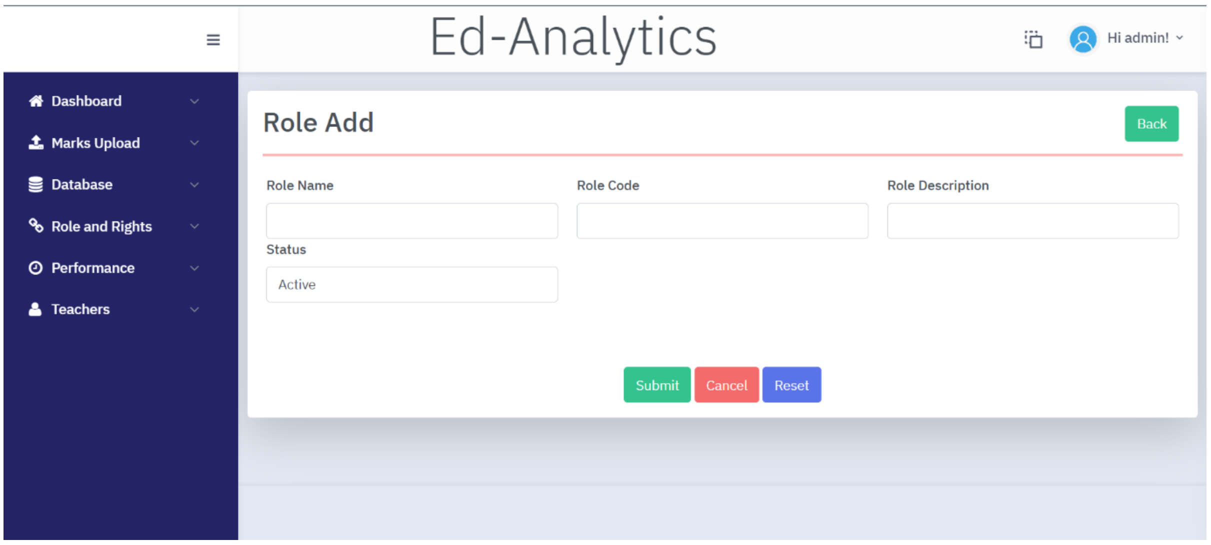
Role Master

Role Name: All

Found 3 result

Role Name	Role Code	Role Description	Status	Edit
Admin	AD	Admin	Active	
Student	ST	Student	Active	

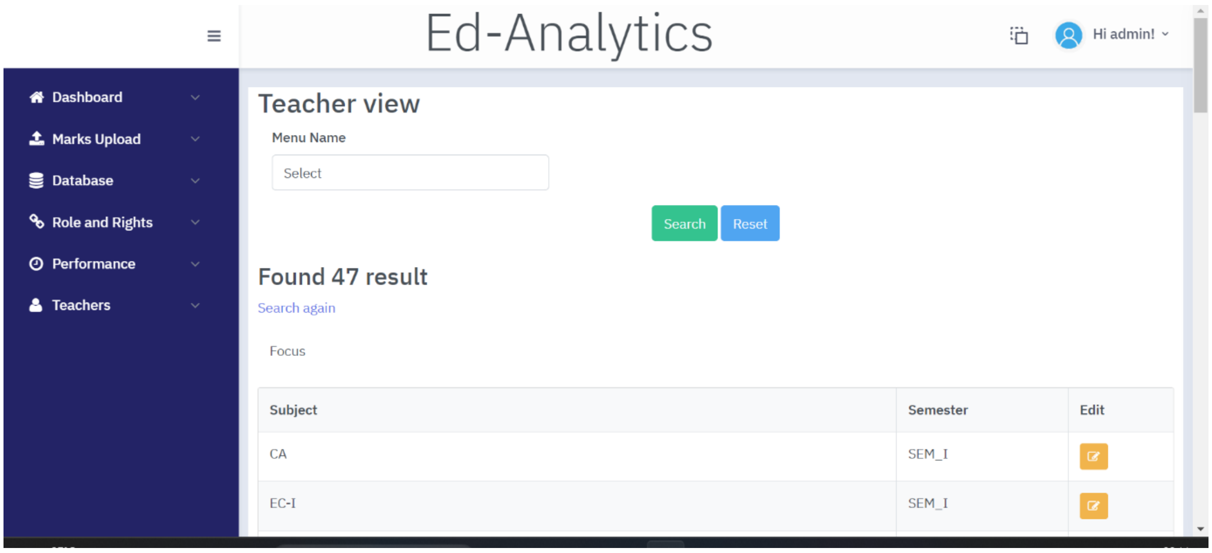
Role Add Page



Role Add

Role Name	Role Code	Role Description
Status		
Active		

Teacher's View Page



Teacher view

Menu Name

Select

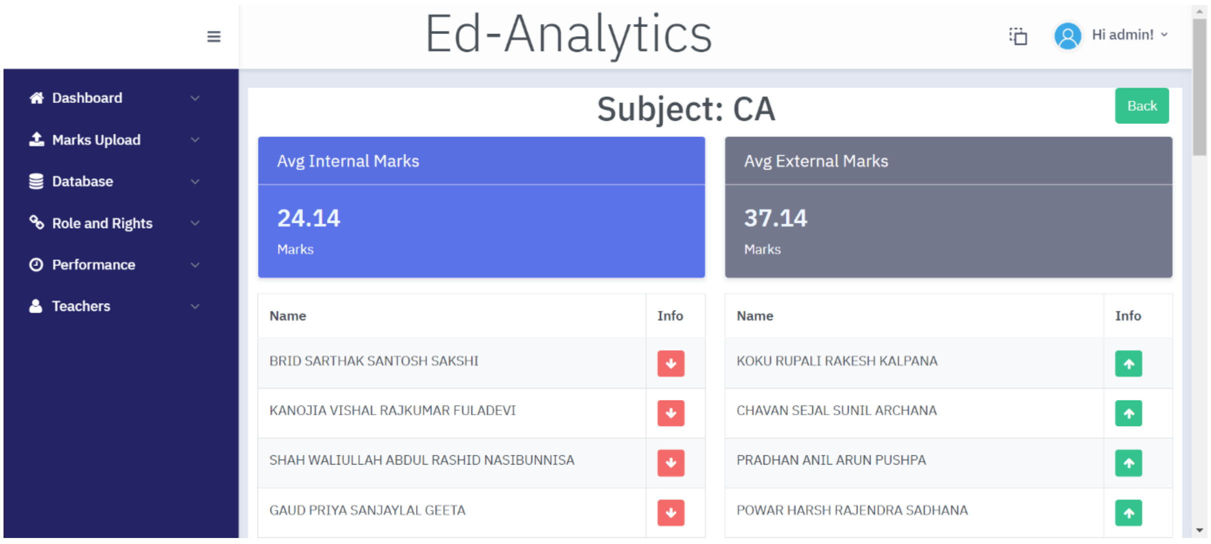
Found 47 result

Search again

Focus

Subject	Semester	Edit
CA	SEM_I	
EC-I	SEM_I	

Teacher's View Page



Subject: CA

Avg Internal Marks

24.14
Marks

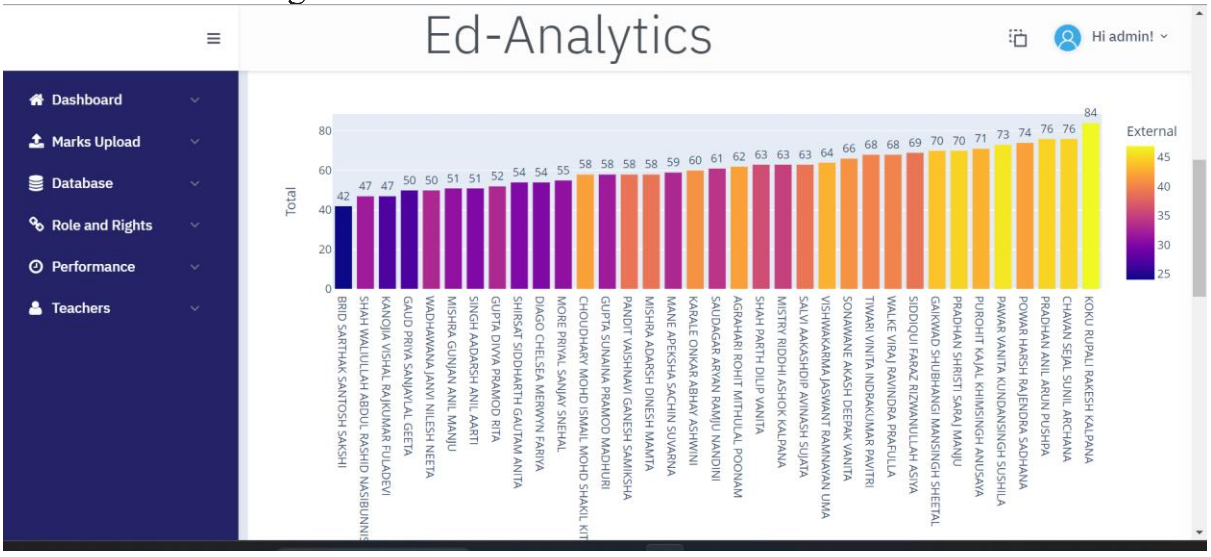
Avg External Marks

37.14
Marks

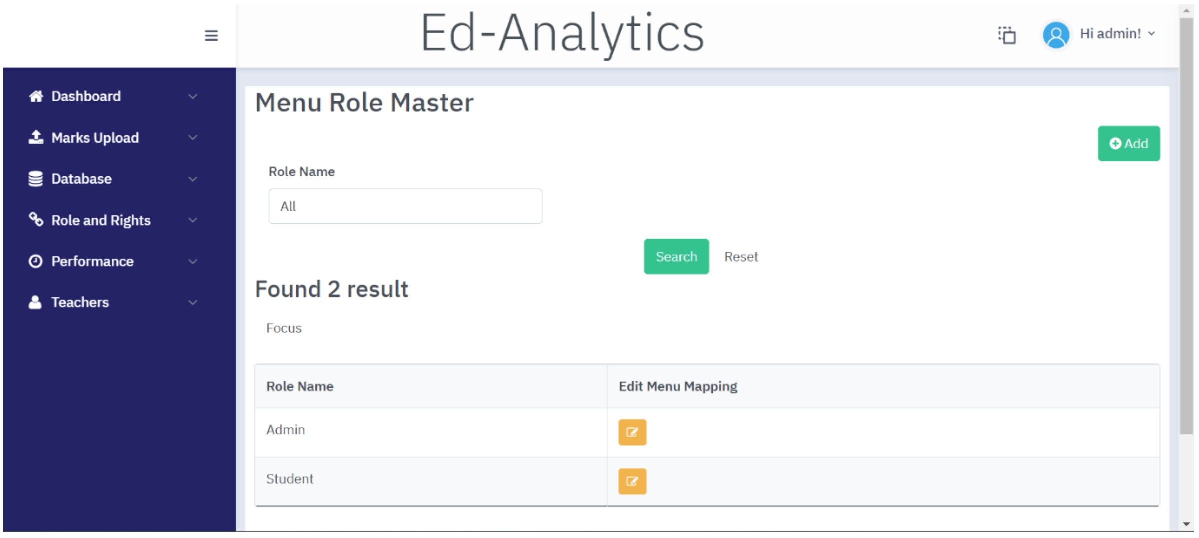
Name	Info
BRID SARTHAK SANTOSH SAKSHI	↓
KANOJIA VISHAL RAJKUMAR FULADEVI	↓
SHAH WALIULLAH ABDUL RASHID NASIBUNNISA	↓
GAUD PRIYA SANJAYLAL GEETA	↓

Name	Info
KOKU RUPALI RAKESH KALPANA	↑
CHAVAN SEJAL SUNIL ARCHANA	↑
PRADHAN ANIL ARUN PUSHPA	↑
POWAR HARSH RAJENDRA SADHANA	↑

Teacher's View Page


≡
Ed-Analytics
Hi admin!
Dashboard
Marks Upload
Database
Role and Rights
Performance
Teachers
Total
0
20
40
60
80
42
47
47
50
50
51
51
52
54
54
55
58
58
58
58
59
60
61
62
63
63
63
64
66
68
68
69
70
70
71
73
74
76
76
84
BRID SARTHAK SANTOSH SAKSHI
SHAH WALIULLAH ABDUL RASHID NASIBUNNIS
KANOJIA VISHAL RAJKUMAR FULADEVI
GAUD PRIYA SANJAYLAL GEETA
WADHAWANA JANVI NILESH NEETA
MISHRA GUNJAN ANIL MANJU
SINGH AADARSH ANIL AARTI
GUPTA DIVYA PRAMOD RITA
SHIRSAT SIDDHARTH GAUTAM ANITA
DIAGO CHELSEA MERWYN FARIYA
MORE PRIYAL SANJAY SNEHAL
CHOUDHARY MOHD ISMAIL MOHD SHAKIL KIT
GUPTA SUNAINA PRAMOD MADHURI
PANDIT VAISHNAVI GANESH SAMIKSHA
MISHRA ADARSH DINESH MAMTA
MANE APEKSHA SACHIN SUVARNA
KARALE ONKAR ABHAY ASHWINI
SAUDAGAR ARYAN RAMJU NANDINI
AGRAHARI ROHIT MITHULAL POONAM
SHAH PARTH DILIP VANITA
MISTRY RIDDHI ASHOK KALPANA
SALVI AAKASHDIP AVINASH SUJATA
VISHWAKARMA JASWANT RAMNAYAN UMA
SONAWANE AKASH DEEPAK VANITA
TIWARI VINITA INDRAKUMAR PAVITRI
WALKE VIRAJ RAVINDRA PRAFULLA
SIDDIQUI FARAZ RIZWANULLAH ASIYA
GAIKWAD SHUBHANGI MANSINGH SHEETAL
PRADHAN SHRISTI SARAJ MANJU
PUROHIT KAJAL KHIMSINGH ANUSAYA
PAWAR VANITA KUNDANSINGH SUSHILA
POWAR HARSH RAJENDRA SADHANA
PRADHAN ANIL ARUN PUSHPA
CHAVAN SEJAL SUNIL ARCHANA
KOKU RUPALI RAKESH KALPANA
External
45
40
35
30
25

Menu Role Master Page



Menu Role Master

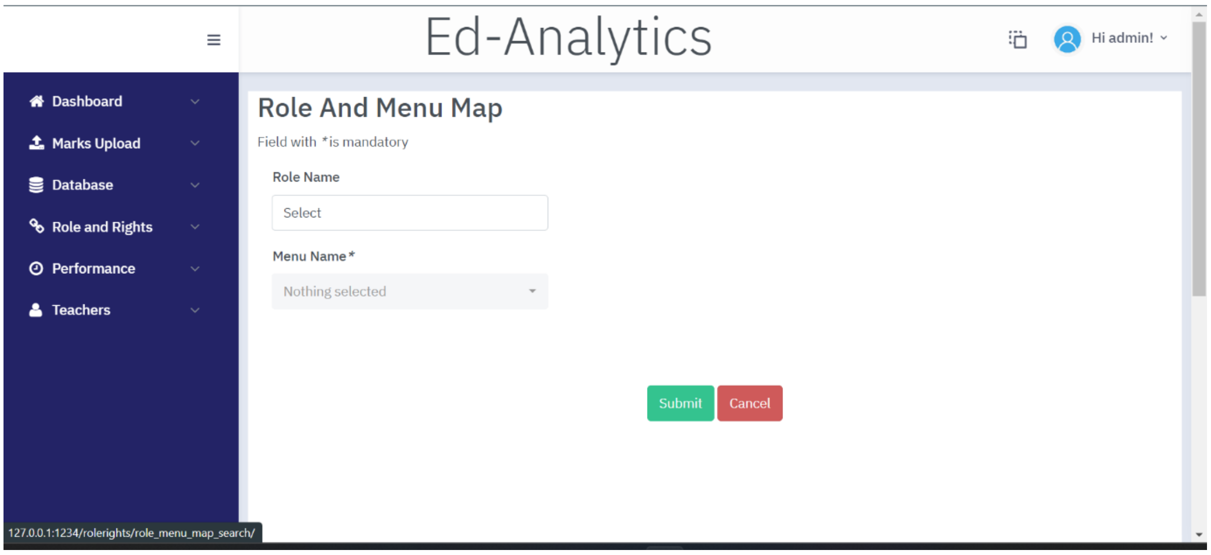
Role Name: All

Found 2 result

Focus

Role Name	Edit Menu Mapping
Admin	
Student	

Menu Role Master Page



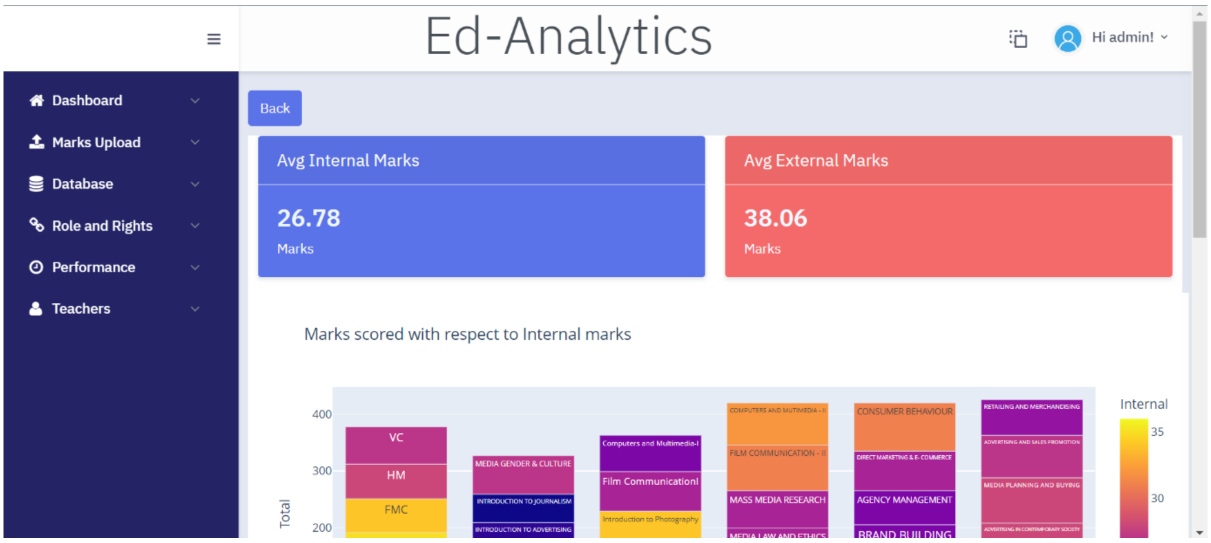
Role And Menu Map

Field with *is mandatory

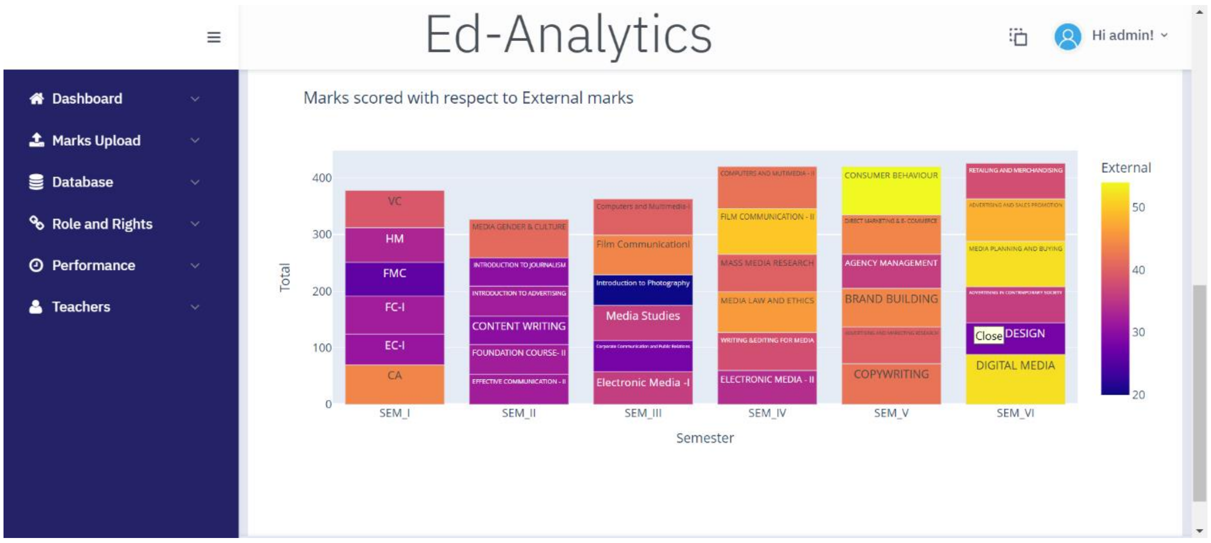
Role Name: Select

Menu Name*: Nothing selected

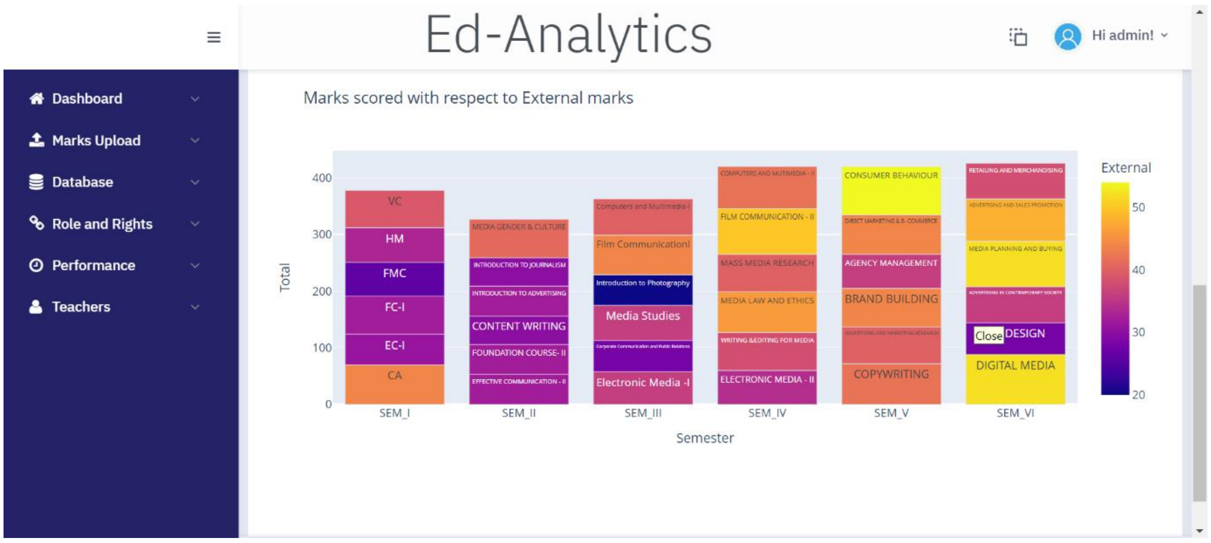
Student’s view Page



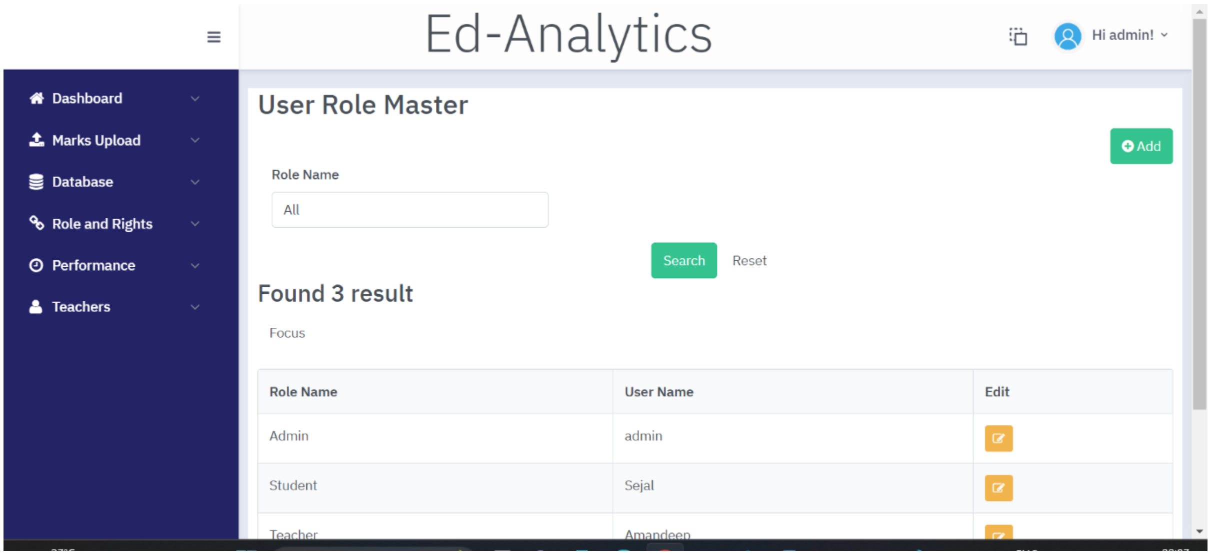
Student’s view Page



Student’s view Page



User Role Master Page



User Role Master

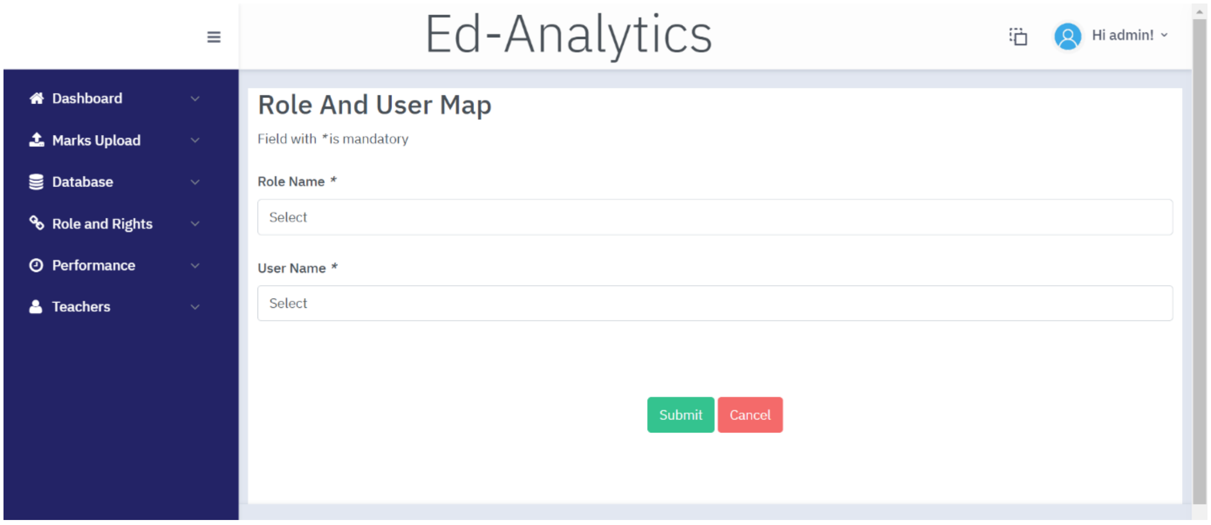
Role Name: All

Found 3 result

Focus

Role Name	User Name	Edit
Admin	admin	
Student	Sejal	
Teacher	Amandeep	

User Role Master Page



Role And User Map

Field with *is mandatory

Role Name *

Select

User Name *

Select

Submit Cancel

Future works

The Ed-Analytics project is a robust and scalable platform for analyzing academic performance using data-driven techniques and visualization tools. However, there are several areas where the project could be extended and improved to provide even more insights into academic performance. Some of the possible areas for future work include:

Incorporating Machine Learning Techniques: One of the most significant areas for future work in the Ed-Analytics project is the incorporation of machine learning techniques. Machine learning algorithms could be used to analyze patterns in academic performance data and provide insights into the factors that influence academic performance. For example, machine learning algorithms could be used to predict the grades of students based on their past performance, demographic data, and other factors.

Integration with Other Data Sources: The Ed-Analytics project could be extended to incorporate data from other sources, such as online learning platforms and social media platforms. The integration of data from multiple sources could provide a more comprehensive picture of academic performance and help identify factors that influence academic performance.

Customizable Visualizations: The project's current visualizations are pre-built and not customizable. In the future, the project could be extended to allow users to create custom visualizations based on their specific needs and preferences. This would provide users with greater flexibility in analyzing academic performance data and enable them to create visualizations that are tailored to their needs.

Real-Time Data Processing: The project's current data processing module requires the data to be manually uploaded and processed. In the future, the project could be extended to support real-time data processing, allowing users to access and analyze academic performance data in real-time.

Collaborative Tools: The project could be extended to include collaborative tools, such as chat rooms and discussion forums, to facilitate collaboration and knowledge sharing among users. Collaborative tools could be used to help teachers and students work together to identify areas where students need additional support and to share best practices for academic success.

Enhanced User Management: The project's current user management module is basic and could be improved to provide more features and functionality. For example, the project could be extended to support role-based access control, allowing administrators to control user access to specific features and modules based on their roles and responsibilities.

Integration with Learning Management Systems: The Ed-Analytics project could be integrated with Learning Management Systems (LMS) to provide seamless access to academic performance data. Integration with LMS could also enable the project to leverage data from LMS to provide more insights into academic performance.

In conclusion, the Ed-Analytics project is a powerful platform for analyzing academic performance using data-driven techniques and visualization tools. However, there are several areas where the project could be extended and improved to provide even more insights into academic performance. The future work described above represents some of the possible areas for future development and improvement.

Conclusion

The Ed-Analytics project is an essential tool that utilizes data analysis techniques and visualization tools to provide insights into academic performance. It is a powerful platform that helps students, teachers, and administrators identify areas that require additional support to improve academic outcomes.

The project leverages the power of the Django web framework and various Python libraries, including NumPy, Pandas, Matplotlib, and Seaborn, to generate insights into academic performance. It includes various visualizations, such as bar graphs, pie charts, and tables, to help users analyze academic performance and make informed decisions.

Despite the challenges encountered while developing the project, the Ed-Analytics project is a significant step forward in using data-driven techniques to improve academic performance and enhance the educational experience for students. The platform's data-driven approach ensures that insights into academic performance are accurate and reliable, providing users with valuable information for decision-making.

In conclusion, the Ed-Analytics project is an effective tool for analyzing academic performance, and it has the potential to enhance the educational experience for students, teachers, and administrators alike. The project's methodology ensures that user data is secure, accessible, and customizable, providing an efficient and effective platform for academic performance analysis. While there is still room for improvement, the Ed-Analytics project represents a significant achievement in the field of data-driven educational analysis, and it has the potential to revolutionize the way we approach academic performance analysis in the future.

References

Django: https://www.djangoproject.com/

Raniranjan Jhumjhunwala College: https://www.rjcollege.edu.in/