

A
PROJECT REPORT
ON
“UNIVERSITY MANAGEMENT SYSTEM”

THE PARTIAL FULFILLMENT OF REQUIREMENT FOR THE
AWARD OF THE DEGREE

OF
MASTER OF COMPUTER APPLICATIONS
(2022-2024)



Under the supervision of:

Dr. Preeti Gulia

Assoc. Professor, DCSA

MDU, Rohtak

Submitted by:

Anurag Vashist

MCA - 2ndSem

Roll No -22110

DEPARTMENT OF COMPUTER SCIENCE&APPLICATIONS

MAHARSHI DAYANAND UNIVERSITY, ROHTAK- 124001

Telephone: +91-1262- 393203, 393202 (O)

E-Mail: hod.computerscience@mdurohtak.ac.in



DEPARTMENT OF COMPUTER SCIENCE & APPLICATIONS
MAHARSHI DAYANAND UNIVERSITY ROHTAK

(www.mdurohtak.ac.in)

(NAAC Accredited 'A+' Grade Haryana State University)

PROF. NASIB SINGH GILL

No. MDU/DCS/___/_____

HEAD OF DEPARTMENT

Date: _____

CERTIFICATE-cum-DECLARATION

I, **Anurag Vashist**, hereby declare that the work presented in the Project Report titled "**UNIVERSITY MANAGEMENT SYSTEM**" and submitted as Industry Internship/Project-I as part of MCA 2nd Semester to Department of Computer Science & Applications, M. D. University, Rohtak is an authentic record of my work carried out during the 2nd semester (Mar-June, 2023) under the supervision of **Dr. Preeti Gulia, Assoc. Professor**, Department of Computer Science & Applications.

Further, I also undertake that the matter embodied in this Project Report is my own work and has not been submitted by me or by any other candidate for the award of any other degree anywhere else.

Student Name – Anurag Vashist

Roll No. – 22110

MCA 2nd Sem

Countersigned by Internal Supervisor

Name: Dr. Preeti Gulia

Designation: Assoc. Professor, DCSA

Forwarded by:

(Head of Department)

ACKNOWLEDGEMENT

Acknowledgement is not mere formality but a genuine opportunity to thank all those people without that active support in this project would not be able to be possible.

Also I am extremely grateful to my project supervisor **Dr. Preeti Gulia Mam** without whose guidance, help, inputs, ideas and creative criticism, this project work would have not been possible. She has been very understanding and friendly.

I would like to express my sincere gratitude to **Dr. Nasib Singh Gill Sir**. His golden words always gave me the inspiration to come out of my tough times, professionally as well as personally. I'd also like to thank all the faculty members of the Department for their valuable suggestions on how to improve this project work, as well as for providing all of the necessary assistance and resources.

Finally I also would like to thank my friends and family for always supporting me. I'm also grateful to everyone for their unwavering support in all of my endeavours.

Signature of the student

Anurag Vashist

MCA 2nd sem

Roll no. 22110

Maharshi Dayanand University, Rohtak

INDEX

Chapter	Contents	PageNo
	Abstract	5
1	Introduction to Project 1.1 Objective 1.2 Project Brief	6
2	Methodology 2.1 Project Requirements Specification 2.2 Feasibility Study	8
3	Requirement Specification 3.1 Hardware Requirements 3.2 Software Requirements 3.3 Programming language used 3.4 Technologies used	16
4	System Design & Analysis 4.1 Description	22
5	Coding 5.1 Backend Code 5.2 Frontend Code	23
6	Screenshot/Snapshot	47
7	Implementation 7.1 Post Implementation	52
8	Testing 8.1 Types of Testing 8.2 Cost Effectiveness 8.3 Customer satisfaction 8.4 Secuirity	53
9	Conclusion And Future Scope 9.1 Future Enhancements	57
10	References	58

Abstract

“University Management System” is a one-stop cloud system for managing and processing academic data about students and staff with additional features. This project is based on web technology. So, it is easily accessible for users. Backend is the core of this project. API (Application Programming Interface) is used for communication between Frontend and Backend.

Chapter 1

Introduction to Project

1.1 Objective

An integrated cloud solution for complete computerization of University & College, build on the most futuristic technologies like- Python, asgiref, Django, Django widget tweaks, pytz, sqlparse denoted as UMS - University Management System. The solution is easy to use web based software created specifically for the management and administration of private and public Universities.

1.2 Project Brief

“University Management System” contains 4 main modules.

1. Home module
2. Student module
3. Teacher module
4. Admin module.

Home Module

It is a public module. That can be accessed by any user. This module contains different webpages like Home page, About page, login and signup page. There are different login pages for students and admin to provide extra layer of security. Homepage shows the key features that are provided in “University Management System”

Student Module

It is a private module. It can only be accessed by using Student’s registration number and password. This module contains Profile page of student and Dashboard of the student. All the notice and attendance are shown on the Student’s dashboard.

Teacher Module

It is also a private model and it can be accessed using Teacher's userID and a password. This module contains the teacher's dashboard and a Subjects page on which a teacher can manage the assignments and test of different subjects.

Admin Module

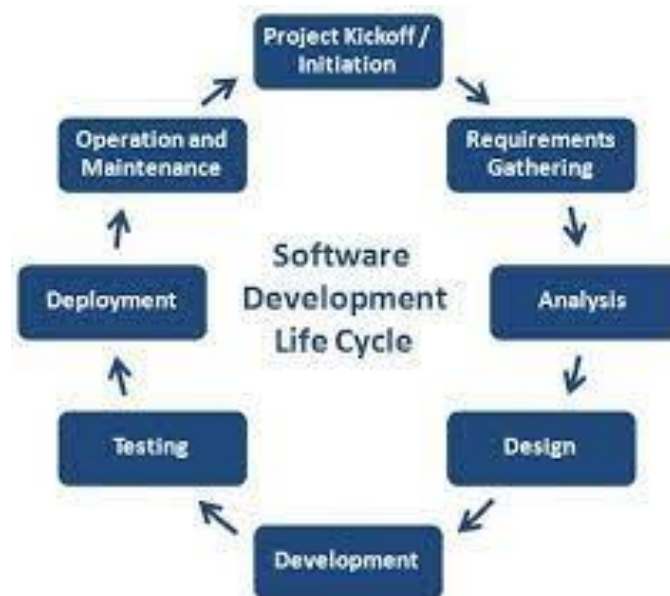
This is the core model of this project. It is an ultra private model. Which can only be accessed via specific IP addresses and after IP authentication, user also need to provide userID and password. So it is very secure model. This module contains Admin's dashboard. From which admin can verify the Students. This module also contains a Departments Page where Admin can add new department, add a new programme, add new subjects to the programme, add new teachers in the department.

Chapter 2

Methodology

The methodology of developing of project will be a step-by-step sequence to design, develop and deliver the application. In software engineering this methodology called 'waterfall model' which one portion of work follows after another in a linear sequence. Following steps will be followed in this methodology:

- Initiation (Requirement Specification);
- Planning and design;
- Execution (construction and coding);
- Validation (Testing);
- Closure (Installation and Maintenance).



2.1 Project Requirements Specification

By project requirements specifications we can analyse the tasks which going to be done by the system. The function and performance of allocated to software as part of system engineering are refined by establishing a complete information description. A detailed functional and behavioural description of the project and concentrating on requirements and constraints of that will provide and good product. The proposed system should follow these requirements:

- i) System should provide support to local and private API routes so that the data is secure.
- ii) System should provide Customizability feature to Students and Teachers.
- iii) System should also provide facility to modify any newly added information. according to their needs.
- iv) System should keep Errors in check by implementing Error Handling

Whether the new system affects the current users in the system?

The new proposed system will affect the users in the following areas :

- Accuracy
- Efficiency
- Productivity
- Robustness
- Lesser time consuming

2.2 Feasibility study

Feasibility study defines all the requirements to performance characteristics of system.

For system to be feasible, the designed sounders take various factors or performance requirements by which the system will be operated.

A feasibility study is short, focused study which aims at selecting the best system that meets performance requirements. Information is gathered regarding the general requirements of the proposed system.

After analysing the scope of the project, the feasibility study is very essential to be held. It is basically keeping the following points in mind.

Hence, we perform feasibility study to make our project compatible for present environment. The project is built with the help of JAVA technology which is reliable and efficient platform to work upon. This concept saves time and lessens the teacher's and student's frustration.

Type of Feasibility Study

There are various measures of feasibility that helps to decide whether a particular project is feasible or not. These measures include

- Operational Feasibility
- Technical Feasibility
- Economic Feasibility
- Behavioural Feasibility

Operational Feasibility:

No doubt that proposed system is fully GUI based that is very user friendly and all inputs to be taken all self-explanatory even to laymen. Besides, a proper training has been conducted to let know the essence of the system to the users so that they feel comfortable with new system. As far our study is concerned the clients are comfortable and happy as the system has cut down their loads and doing.

Economic Feasibility

Economic analysis is most frequently used for evaluation of the effectiveness of the system. More commonly known as cost/benefit analysis the procedure is to determine the benefit and saving that are expected from a system and compare them with costs, decisions is made to design and implement the system

This is very important aspect to be considered while developing a project. We decided the technology based on minimum possible cost factor.

All hardware and software cost has to be borne by the organization.

Overall, we have estimated that the benefits the organization is going to receive from the proposed system will surely overcome the initial costs and the later on running cost for the system.

Behavioural Feasibility

People are inherently resistant to change and computer has been known to facilitate changes. An estimate should be made of how strong the user is likely to move towards the development of computerized system. These are various levels of users in order to ensure proper authentication and authorization and security of sensitive data of the organization.

People are inherently resistant to change and computer has been known to facilitate changes. An estimate should be made of how strong the user is likely to move towards the development of computerized system. These are various levels of users in order to ensure proper authentication and authorization and security of sensitive data of the organization.

Technically feasible:

This website is very much technically feasible. This website is very much concerned with specifying equipment and the website will successfully satisfy almost all the user's requirements. The technical need for this system may vary considerably but might include:

This includes the study of function, performance and constraints that may affect

the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system, as described in the SRS(System Requirement Specification), and checked everything was possible using different type of frontend and backend platforms.

Based on the outline design of the system requirements in terms of inputs, output, procedures, the technical issues raised during technical feasibility include:

- Does the necessary technology exist to do what is proposed?
- Does the proposed equipment have the technical capacity to hold the data required to use in the new system?
- Adequate responses provided by the proposed system?
- Is the system flexible enough to facilitate expansion?
- Is there any technical guarantee of accuracy, reliability, ease of access and data security?

The system developer's task is to view needed capabilities in light of currently available technology. Our site works hand in hand with high technology. A database has to be maintained in order to update and backup data whenever required.

Therefore, the basic input/output of all files is identified. So, the project can easily be builtup and it will also be technically feasible.

2.3 PROBLEMS TO TACKLE DURING WEB DEVELOPMENT

Web development is expediting at an aggressive rate. Better and user-friendly interfaces are in demand. When it comes to developing a successful web application there are a number of factors defining that success. Customers are eager to know different aspects of your product such as it's cost, look and feel, and value for money. To know about the company details, customers may visit the company's website, mobile apps and social media platforms. Thus, it is important how you interact and respond the customers. These are the problems I faced during development phase of this project-

USER INTERFACE AND USER EXPERIENCE

Think a decade ago, the web was a completely different place. Smartphones don't exist. Simpler and customer oriented web application are highly expected now. Sometimes it's the small UI elements that make the biggest impact. In the era of Smartphones, websites should be responsive enough on the smaller screens. If your web applications frustrate or confuse users, then it is difficult to maintain your customer's loyalty for your website. Website navigation is another part often neglected by developers. Intuitive navigation creates a better user experience for the website visitor. Intuitive navigation is leading your audience to the information they are looking without a learning curve. And when the navigation is intuitive, visitors can find out information without any pain, creating a flawless experience preventing them from visiting the competitors.

SCALABILITY

Scalability is neither performance nor it's about making good use of computing power and bandwidth. It's about load balancing between the servers, hence, when the load increases (i.e. more traffic on the page) additional servers can be added to balance it. You should not just throw all the load on a single server but you should design the software such that it can work on a cluster of servers. Service-oriented architecture (SOA) can help in improving scalability when more and more servers are added. SOA gives you the flexibility to change easily. Service oriented architecture is a design where application components provide services to other components through the communication protocol, basically over a network.

PERFORMANCE

Generally, it is accepted that website speed has the major importance for a successful website. When your business is online every second counts. Slow web applications are a failure. As a result, customers abscond your website thus, damaging your revenue as well as reputation. It is said that think about performance first before developing the web application. Some of the

performance issues are Poorly written code, Un-Optimized Databases, Unmanaged Growth of data, Traffic spikes, Poor load distribution, Default configuration, Troublesome third party services, etc. A content distribution network (CDN) is globally distributed network of proxy servers deployed in multiple data centres. It means instead of using a single web server for the website, use a network of servers. Some of the benefits of CDN are that the requests on the server will be routed to different servers balancing the traffic, the files are divided on different CDNs so there will be no queuing and wait for downloading different files like images, videos, text, etc.

KNOWLEDGE OF FRAMEWORK AND PLATFORMS

Frameworks are the kick start for development languages: they boost performance, offer libraries of coding and extend capabilities, so developers need not do hand-coding web applications from the ground up. Frameworks offer features like models, APIs, snippets of code and other elements to develop dynamic web applications. Some of the frameworks have a rigid approach to development and some are flexible. Common examples of web frameworks are PHP, ASP.Net, Ruby on Rails and J2EE. Web platforms provide client libraries build on existing frameworks required to develop a web application or website. A new functionality can be added via external API. Developers and small business owners should have a clear understanding of their company needs related to website and application development. Information delivery and online presence would require a simple web platform such as WordPress or Squarespace but a selling product requires an e-commerce platform such as Magento, Shopify. WooCommerce or BigCommerce). While choosing the perfect platform one should also consider technical skills, learning curve, pricing, customization options and analytics.

SECURITY

In the midst of design and user experience, web app security is often neglected. But security should be considered throughout the software development life cycle, especially when the application is dealing with the vital information such as payment details, contact information, and confidential data. There are many things to consider when it comes to web application security such as denial of

service attacks, the safety of user data, database malfunctioning, unauthorized access to restricted parts of the website, etc. Some of the security threats are Cross-Site Scripting, Phishing, Cross-Site Request Forgery, Shell Injection, Session Hijacking, SQL Injection, Buffer Overflow, etc. The website should be carefully coded to be safe against these security concerns.

Web development can be deliberately difficult as it involves achieving a final product which should be pleasing, builds the brand and is technically up to date with sound visuals.

Chapter-3

Requirement Specifications

3.1 Hardware Requirements

For Development

	Windows requirements	Mac requirements	Linux requirements
Processor	Intel Pentium 4 or later	Intel	Intel Pentium 4 or later
Memory	2 GB minimum, 4 GB recommended		
Screen resolution	1280x1024 or larger		
Application window size	1024x680 or larger		
Internet connection	Required		

For User

The Hardware requirements for this system are as follows:

- 1) 2nd-generation Core i5 (2GHz+), 3rd/4th-generation Core i5 processor, or equivalent
- 2) Memory (RAM) 1 GB or above
- 3) Hard Disk space 1 GB for the database and the client software
- 4) Cache Memory 128 KB or above

3.2 Software Requirements

For Development

- VSCode Editor
- Web browser (Chrome)
- Django
- Python
- Asgiref
- Django widget tweaks
- Pytz
- Sqlparse

For User

- Web browser (Eg. Chrome)

3.3 Programming Languages Used

Python is a popular programming language. It can be used on a server to create web applications.

Why Python?

Python is a popular programming language for web development due to several key reasons:

1. **Readability and Simplicity:** Python has a clean and intuitive syntax that is easy to understand and read, making it a great choice for beginners and experienced developers alike. Its emphasis on code readability promotes maintainability and collaboration within development teams.
2. **Vast Ecosystem and Frameworks:** Python boasts a vast ecosystem of libraries and frameworks specifically designed for web development. The most prominent web frameworks in Python include Django, Flask, and Pyramid, which provide powerful tools and abstractions for building web applications efficiently.

3. **Productivity and Rapid Development:** Python's simplicity and expressive syntax contribute to increased productivity, allowing developers to write code faster and with fewer lines compared to other languages. Python's extensive libraries and frameworks also provide pre-built components and modules, enabling rapid development of web applications.
4. **Cross-Platform Compatibility:** Python is a cross-platform language, meaning that web applications written in Python can run on various operating systems, including Windows, macOS, and Linux, without major modifications. This flexibility simplifies deployment and makes it easier to scale applications across different platforms.
5. **Integration Capabilities:** Python offers excellent integration capabilities with other languages and systems. It can be seamlessly integrated with languages like C/C++, Java, and .NET, allowing developers to leverage existing codebases or take advantage of specialized libraries or services when building web applications.
6. **Large and Active Community:** Python has a vibrant and active community of developers, which means you can find extensive resources, tutorials, and documentation to support your web development journey. The community also contributes to the development of numerous open-source libraries and frameworks, ensuring continuous improvement and innovation.
7. **Data Science and Machine Learning Capabilities:** Python has gained significant popularity in the field of data science and machine learning. Its extensive libraries, such as NumPy, Pandas, and scikit-learn, provide robust capabilities for data manipulation, analysis, and modeling. Python's compatibility with popular machine learning frameworks like TensorFlow and PyTorch makes it an excellent choice for building data-intensive web applications.

While Python excels in web development, it's worth noting that other languages like JavaScript, Ruby, and PHP also have their strengths in this domain. Ultimately, the choice of programming language depends on factors such as project requirements, developer expertise, and personal preferences.

3.4 Technologies Used

These are the modern technologies that is used in building this project.

- 1) HTML
- 2) CSS
- 3) Python
- 4) Django
- 5) SQL Lite 3

1) HTML

HTML(Hypertext Markup Language) is the code used to structure a web page and its content. HTML consists of a series of elements, which you use to surround or wrap different content parts to make it appear a certain way or act.

Every website you open in a web browser uses HTML, from social networks to music services.

2) CSS

CSS or cascading style sheets have made the development of web pages a lot easier. CSS allows you to easily link to other documents in a website. With the help of CSS you can have a control over the various elements in different web pages of your site. CSS only defines the structure and content presentation of a website. It has nothing to do with the design of a website. However, the CSS influences how the design will look like after the final process is over. You can control the font, positioning, color and style information of an entire website with the help of a single CSS sheet.

3) Python

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.

- Python can be used for rapid prototyping, or for production-ready software development.

Why Use Python for Web Development?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

4) Django

- Django is a Python framework that makes it easier to create web sites using Python.
- Django takes care of the difficult stuff so that you can concentrate on building your web applications.
- Django emphasizes reusability of components, also referred to as DRY (Don't Repeat Yourself), and comes with ready-to-use features like login system, database connection and CRUD operations (Create Read Update Delete).
- Django is especially helpful for database driven websites.

How does Django Work?

Django follows the MVT design pattern (Model View Template).

- Model - The data you want to present, usually data from a database.
- View - A request handler that returns the relevant template and content - based on the request from the user.
- Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.



5) SQL Lite 3

- SQLite3 can be integrated with Python using sqlite3 module, which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You do not need to

install this module separately because it is shipped by default along with Python version 2.5.x onwards.

- To use sqlite3 module, you must first create a connection object that represents the database and then optionally you can create a cursor object, which will help you in executing all the SQL statements.

Chapter-4

System Design & Analysis

Systems design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organization.

4.1 Perspective

As we have discussed earlier “University Management System” contains 4 main modules.

1. Home module
2. Student module
3. Teacher module
4. Admin module.

4.1 Description

A systemic approach is required for a coherent and well-running system. Bottom-Up or Top-Down approach is required to take into account all related variables of the system. A designer uses the modelling languages to express the information and knowledge in a structure of system that is defined by a consistent set of rules and definitions. The designs can be defined in graphical or textual modelling languages.

Design methods

1. **Architectural design:** To describes the views, models, behaviour, and Structure of the system.
2. **Logical design:** To represent the data flow, inputs and outputs of the system. Example: ER Diagrams (Entity Relationship Diagrams).
3. **Physical design :** Defined as :
 - How users add information to the system and how the system represents information back to the user.
 - How the data is modelled and stored within the system.
 - How data moves through the system, how data is validated, secured and/or transformed as it flows through and out of the system.

Chapter-5

Coding

The input to the Design Phase is the design document. During the **coding phase**, different modules identified in the design document are coded according to the modules specifications. The **Objectives** of the coding phase is to transform the design of the system, as given by its module specifications, into a high level languages and code and then unit this code. Software developers adhere to some well defined and standard style of coding called coding standards. The reasons for adhering to as standard coding style are the following.

It gives a uniform appearance to the codes written by different engineers. It encourages good programming practices . It encourages code understanding.

Coding Standards

- Rules for limiting the use of global.
- Content of the headers preceding codes for different modules.
- Errors return conventions and exception handling mechanisms.

Code Walk-Through

It is an Informal technique for analysis of the code. A code walk through of a module is undertaken after the coding of the module is complete. Here, after a module has been coded, members of the development team selects some test cases and simulates execution of the by hand.

Code Inspections

They aim explicitly at the discovery of commonly made errors. During which, the code is examined for the presence of certain kinds of errors, in contrast to the hand simulation on code execution as done in code walk-through.

For the purpose of clarity „source code“ is taken to mean any fully executable description of a software system. It is therefore so construed as to include machine code, very high level languages and executable graphical representations of systems.

The code base of a programming project is the larger collection of all the source code of all the computer programs which make up the project.

The project code is divided into two segments-

- **Backend**
- **Frontend**

Backend Code

Python with Django Framework

Admin.py

```
from django.contrib import admin
from .models import Attendance, StudentExtra, TeacherExtra, Notice
# Register your models here. (by Anurag)
class StudentExtraAdmin(admin.ModelAdmin):
    pass
admin.site.register(StudentExtra, StudentExtraAdmin)

class TeacherExtraAdmin(admin.ModelAdmin):
    pass
admin.site.register(TeacherExtra, TeacherExtraAdmin)

class AttendanceAdmin(admin.ModelAdmin):
    pass
admin.site.register(Attendance, AttendanceAdmin)

class NoticeAdmin(admin.ModelAdmin):
    pass
admin.site.register(Notice, NoticeAdmin)
```


apps.py

```
from django.apps import AppConfig
class UniversityConfig(AppConfig):
    name = 'university'
```

views.py

```
from django.shortcuts import render,redirect,reverse
from . import forms,models
from django.db.models import Sum
from django.contrib.auth.models import Group
from django.http import HttpResponseRedirect
from django.contrib.auth.decorators import login_required,user_passes_test
from django.conf import settings
from django.core.mail import send_mail
```

```
def home_view(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect('afterlogin')
    return render(request,'university/index.html')
#for showing signup/login button for teacher(by Anurag)
def adminclick_view(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect('afterlogin')
    return render(request,'university/adminclick.html')
```

```
#for showing signup/login button for teacher(by Anurag)
def teacherclick_view(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect('afterlogin')
    return render(request,'university/teacherclick.html')
```

```
#for showing signup/login button for student(by Anurag)
def studentclick_view(request):
    if request.user.is_authenticated:
        return HttpResponseRedirect('afterlogin')
    return render(request,'university/studentclick.html')
```

```
def admin_signup_view(request):
    form=forms.AdminSigupForm()
    if request.method=='POST':
        form=forms.AdminSigupForm(request.POST)
        if form.is_valid():
            user=form.save()
            user.set_password(user.password)
```

```

        user.save()

        my_admin_group = Group.objects.get_or_create(name='ADMIN')
        my_admin_group[0].user_set.add(user)

        return HttpResponseRedirect('adminlogin')
    return render(request,'university/adminsignup.html',{'form':form})

def student_signup_view(request):
    form1=forms.StudentUserForm()
    form2=forms.StudentExtraForm()
    mydict={'form1':form1,'form2':form2}
    if request.method=='POST':
        form1=forms.StudentUserForm(request.POST)
        form2=forms.StudentExtraForm(request.POST)
        if form1.is_valid() and form2.is_valid():
            user=form1.save()
            user.set_password(user.password)
            user.save()
            f2=form2.save(commit=False)
            f2.user=user
            user2=f2.save()

            my_student_group = Group.objects.get_or_create(name='STUDENT')
            my_student_group[0].user_set.add(user)

            return HttpResponseRedirect('studentlogin')
    return render(request,'university/studentsignup.html',context=mydict)

def teacher_signup_view(request):
    form1=forms.TeacherUserForm()
    form2=forms.TeacherExtraForm()
    mydict={'form1':form1,'form2':form2}
    if request.method=='POST':
        form1=forms.TeacherUserForm(request.POST)
        form2=forms.TeacherExtraForm(request.POST)
        if form1.is_valid() and form2.is_valid():
            user=form1.save()
            user.set_password(user.password)
            user.save()
            f2=form2.save(commit=False)
            f2.user=user
            user2=f2.save()

            my_teacher_group = Group.objects.get_or_create(name='TEACHER')
            my_teacher_group[0].user_set.add(user)

            return HttpResponseRedirect('teacherlogin')

```

```

return render(request,'university/teachersignup.html',context=mydict)

#for checking user is teacher , student or admin(by Anurag)
def is_admin(user):
    return user.groups.filter(name='ADMIN').exists()
def is_teacher(user):
    return user.groups.filter(name='TEACHER').exists()
def is_student(user):
    return user.groups.filter(name='STUDENT').exists()

def afterlogin_view(request):
    if is_admin(request.user):
        return redirect('admin-dashboard')
    elif is_teacher(request.user):
        accountapproval=models.TeacherExtra.objects.all().filter(user_id=request.user.id,status
=True)
        if accountapproval:
            return redirect('teacher-dashboard')
        else:
            return render(request,'university/teacher_wait_for_approval.html')
    elif is_student(request.user):
        accountapproval=models.StudentExtra.objects.all().filter(user_id=request.user.id,status
=True)
        if accountapproval:
            return redirect('student-dashboard')
        else:
            return render(request,'university/student_wait_for_approval.html')

#for dashboard of admin(by Anurag)

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_dashboard_view(request):
    teachercount=models.TeacherExtra.objects.all().filter(status=True).count()
    pendingteachercount=models.TeacherExtra.objects.all().filter(status=False).count()

    studentcount=models.StudentExtra.objects.all().filter(status=True).count()
    pendingstudentcount=models.StudentExtra.objects.all().filter(status=False).count()

    teachersalary=models.TeacherExtra.objects.filter(status=True).aggregate(Sum('salary'))
    pendingteachersalary=models.TeacherExtra.objects.filter(status=False).aggregate(Sum('sa
lary'))

    studentfee=models.StudentExtra.objects.filter(status=True).aggregate(Sum('fee',default=
0))
    pendingstudentfee=models.StudentExtra.objects.filter(status=False).aggregate(Sum('fee'
)
)

```

```

notice=models.Notice.objects.all()

#aggregate function return dictionary so fetch data from dictionary(by Anurag)
mydict={
    'teachercount':teachercount,
    'pendingteachercount':pendingteachercount,

    'studentcount':studentcount,
    'pendingstudentcount':pendingstudentcount,

    'teachersalary':teachersalary['salary__sum'],
    'pendingteachersalary':pendingteachersalary['salary__sum'],

    'studentfee':studentfee['fee__sum'],
    'pendingstudentfee':pendingstudentfee['fee__sum'],

    'notice':notice
}

return render(request,'university/admin_dashboard.html',context=mydict)

#for teacher section by admin(by Anurag)

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_teacher_view(request):
    return render(request,'university/admin_teacher.html')

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_add_teacher_view(request):
    form1=forms.TeacherUserForm()
    form2=forms.TeacherExtraForm()
    mydict={'form1':form1,'form2':form2}
    if request.method=='POST':
        form1=forms.TeacherUserForm(request.POST)
        form2=forms.TeacherExtraForm(request.POST)
        if form1.is_valid() and form2.is_valid():
            user=form1.save()
            user.set_password(user.password)
            user.save()

            f2=form2.save(commit=False)
            f2.user=user
            f2.status=True

```

```

        f2.save()

        my_teacher_group = Group.objects.get_or_create(name='TEACHER')
        my_teacher_group[0].user_set.add(user)

        return HttpResponseRedirect('admin-teacher')
    return render(request,'university/admin_add_teacher.html',context=mydict)

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_view_teacher_view(request):
    teachers=models.TeacherExtra.objects.all().filter(status=True)
    return render(request,'university/admin_view_teacher.html',{'teachers':teachers})

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_approve_teacher_view(request):
    teachers=models.TeacherExtra.objects.all().filter(status=False)
    return render(request,'university/admin_approve_teacher.html',{'teachers':teachers})

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def approve_teacher_view(request,pk):
    teacher=models.TeacherExtra.objects.get(id=pk)
    teacher.status=True
    teacher.save()
    return redirect(reverse('admin-approve-teacher'))

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def delete_teacher_view(request,pk):
    teacher=models.TeacherExtra.objects.get(id=pk)
    user=models.User.objects.get(id=teacher.user_id)
    user.delete()
    teacher.delete()
    return redirect('admin-approve-teacher')

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def delete_teacher_from_university_view(request,pk):
    teacher=models.TeacherExtra.objects.get(id=pk)
    user=models.User.objects.get(id=teacher.user_id)
    user.delete()
    teacher.delete()
    return redirect('admin-view-teacher')

@login_required(login_url='adminlogin')

```

```

@user_passes_test(is_admin)
def update_teacher_view(request,pk):
    teacher=models.TeacherExtra.objects.get(id=pk)
    user=models.User.objects.get(id=teacher.user_id)

    form1=forms.TeacherUserForm(instance=user)
    form2=forms.TeacherExtraForm(instance=teacher)
    mydict={'form1':form1,'form2':form2}

    if request.method=='POST':
        form1=forms.TeacherUserForm(request.POST,instance=user)
        form2=forms.TeacherExtraForm(request.POST,instance=teacher)
        print(form1)
        if form1.is_valid() and form2.is_valid():
            user=form1.save()
            user.set_password(user.password)
            user.save()
            f2=form2.save(commit=False)
            f2.status=True
            f2.save()
            return redirect('admin-view-teacher')
    return render(request,'university/admin_update_teacher.html',context=mydict)

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_view_teacher_salary_view(request):
    teachers=models.TeacherExtra.objects.all()
    return render(request,'university/admin_view_teacher_salary.html',{'teachers':teachers})

#for student by admin(by Anurag)

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_student_view(request):
    return render(request,'university/admin_student.html')

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_add_student_view(request):
    form1=forms.StudentUserForm()
    form2=forms.StudentExtraForm()
    mydict={'form1':form1,'form2':form2}
    if request.method=='POST':
        form1=forms.StudentUserForm(request.POST)
        form2=forms.StudentExtraForm(request.POST)
        if form1.is_valid() and form2.is_valid():
            print("form is valid")

```

```

        user=form1.save()
        user.set_password(user.password)
        user.save()

        f2=form2.save(commit=False)
        f2.user=user
        f2.status=True
        f2.save()

        my_student_group = Group.objects.get_or_create(name='STUDENT')
        my_student_group[0].user_set.add(user)
    else:
        print("form is invalid")
        return HttpResponseRedirect('admin-student')
    return render(request,'university/admin_add_student.html',context=mydict)

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_view_student_view(request):
    students=models.StudentExtra.objects.all().filter(status=True)
    return render(request,'university/admin_view_student.html',{'students':students})

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def delete_student_from_university_view(request,pk):
    student=models.StudentExtra.objects.get(id=pk)
    user=models.User.objects.get(id=student.user_id)
    user.delete()
    student.delete()
    return redirect('admin-view-student')

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def delete_student_view(request,pk):
    student=models.StudentExtra.objects.get(id=pk)
    user=models.User.objects.get(id=student.user_id)
    user.delete()
    student.delete()
    return redirect('admin-approve-student')

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def update_student_view(request,pk):
    student=models.StudentExtra.objects.get(id=pk)
    user=models.User.objects.get(id=student.user_id)
    form1=forms.StudentUserForm(instance=user)
    form2=forms.StudentExtraForm(instance=student)

```

```

mydict={'form1':form1,'form2':form2}
if request.method=='POST':
    form1=forms.StudentUserForm(request.POST,instance=user)
    form2=forms.StudentExtraForm(request.POST,instance=student)
    print(form1)
    if form1.is_valid() and form2.is_valid():
        user=form1.save()
        user.set_password(user.password)
        user.save()
        f2=form2.save(commit=False)
        f2.status=True
        f2.save()
        return redirect('admin-view-student')
return render(request,'university/admin_update_student.html',context=mydict)

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_approve_student_view(request):
    students=models.StudentExtra.objects.all().filter(status=False)
    return render(request,'university/admin_approve_student.html',{'students':students})

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def approve_student_view(request,pk):
    students=models.StudentExtra.objects.get(id=pk)
    students.status=True
    students.save()
    return redirect(reverse('admin-approve-student'))

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_view_student_fee_view(request):
    students=models.StudentExtra.objects.all()
    return render(request,'university/admin_view_student_fee.html',{'students':students})

#attendance related view(by Anurag)
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_attendance_view(request):
    return render(request,'university/admin_attendance.html')
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_take_attendance_view(request,cl):
    students=models.StudentExtra.objects.all().filter(cl=cl)
    print(students)
    aform=forms.AttendanceForm()
    if request.method=='POST':

```



```

form=forms.AttendanceForm(request.POST)
if form.is_valid():
    Attendances=request.POST.getlist('present_status')
    date=form.cleaned_data['date']
    for i in range(len(Attendances)):
        AttendanceModel=models.Attendance()
        AttendanceModel.cl=cl
        AttendanceModel.date=date
        AttendanceModel.present_status=Attendances[i]
        AttendanceModel.roll=students[i].roll
        AttendanceModel.save()
    return redirect('admin-attendance')
else:
    print('form invalid')
return
render(request,'university/admin_take_attendance.html',{'students':students,'aform':aform
})

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_view_attendance_view(request,cl):
    form=forms.AskDateForm()
    if request.method=='POST':
        form=forms.AskDateForm(request.POST)
        if form.is_valid():
            date=form.cleaned_data['date']
            attendancedata=models.Attendance.objects.all().filter(date=date,cl=cl)
            studentdata=models.StudentExtra.objects.all().filter(cl=cl)
            mylist=zip(attendancedata,studentdata)
            return
render(request,'university/admin_view_attendance_page.html',{'cl':cl,'mylist':mylist,'date':
date})
else:
    print('form invalid')
return
render(request,'university/admin_view_attendance_ask_date.html',{'cl':cl,'form':form})

#fee related view by admin(by Anurag)
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_fee_view(request):
    return render(request,'university/admin_fee.html')

@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_view_fee_view(request,cl):
    feedetails=models.StudentExtra.objects.all().filter(cl=cl)

```

```

        return render(request,'university/admin_view_fee.html',{'feedetails':feedetails,'cl':cl})

#notice related views(by Anurag)
@login_required(login_url='adminlogin')
@user_passes_test(is_admin)
def admin_notice_view(request):
    form=forms.NoticeForm()
    if request.method=='POST':
        form=forms.NoticeForm(request.POST)
        if form.is_valid():
            form=form.save(commit=False)
            form.by=request.user.first_name
            form.save()
            return redirect('admin-dashboard')
    return render(request,'university/admin_notice.html',{'form':form})

#for TEACHER LOGIN SECTION(by Anurag)
@login_required(login_url='teacherlogin')
@user_passes_test(is_teacher)
def teacher_dashboard_view(request):
    teacherdata=models.TeacherExtra.objects.all().filter(status=True,user_id=request.user.id)
    notice=models.Notice.objects.all()
    mydict={
        'salary':teacherdata[0].salary,
        'mobile':teacherdata[0].mobile,
        'date':teacherdata[0].joindate,
        'notice':notice
    }
    return render(request,'university/teacher_dashboard.html',context=mydict)

@login_required(login_url='teacherlogin')
@user_passes_test(is_teacher)
def teacher_attendance_view(request):
    return render(request,'university/teacher_attendance.html')
@login_required(login_url='teacherlogin')
@user_passes_test(is_teacher)
def teacher_take_attendance_view(request,cl):
    students=models.StudentExtra.objects.all().filter(cl=cl)
    aform=forms.AttendanceForm()
    if request.method=='POST':
        form=forms.AttendanceForm(request.POST)
        if form.is_valid():
            Attendances=request.POST.getlist('present_status')
            date=form.cleaned_data['date']
            for i in range(len(Attendances)):
                AttendanceModel=models.Attendance()
                AttendanceModel.cl=cl

```

```

        AttendanceModel.date=date
        AttendanceModel.present_status=Attendances[i]
        AttendanceModel.roll=students[i].roll
        AttendanceModel.save()
        return redirect('teacher-attendance')
    else:
        print('form invalid')
    return
render(request,'university/teacher_take_attendance.html',{'students':students,'aform':aform})

@login_required(login_url='teacherlogin')
@user_passes_test(is_teacher)
def teacher_view_attendance_view(request,cl):
    form=forms.DateForm()
    if request.method=='POST':
        form=forms.DateForm(request.POST)
        if form.is_valid():
            date=form.cleaned_data['date']
            attendancedata=models.Attendance.objects.all().filter(date=date,cl=cl)
            studentdata=models.StudentExtra.objects.all().filter(cl=cl)
            mylist=zip(attendancedata,studentdata)
            return
render(request,'university/teacher_view_attendance_page.html',{'cl':cl,'mylist':mylist,'date':date})
    else:
        print('form invalid')
    return
render(request,'university/teacher_view_attendance_ask_date.html',{'cl':cl,'form':form})

@login_required(login_url='teacherlogin')
@user_passes_test(is_teacher)
def teacher_notice_view(request):
    form=forms.NoticeForm()
    if request.method=='POST':
        form=forms.NoticeForm(request.POST)
        if form.is_valid():
            form=form.save(commit=False)
            form.by=request.user.first_name
            form.save()
            return redirect('teacher-dashboard')
    else:
        print('form invalid')
    return render(request,'university/teacher_notice.html',{'form':form})

#FOR STUDENT AFTER THEIR Login(by Anurag)
@login_required(login_url='studentlogin')

```

```

@user_passes_test(is_student)
def student_dashboard_view(request):
    studentdata=models.StudentExtra.objects.all().filter(status=True,user_id=request.user.id)
    notice=models.Notice.objects.all()
    mydict={
        'roll':studentdata[0].roll,
        'mobile':studentdata[0].mobile,
        'fee':studentdata[0].fee,
        'notice':notice
    }
    return render(request,'university/student_dashboard.html',context=mydict)

@login_required(login_url='studentlogin')
@user_passes_test(is_student)
def student_attendance_view(request):
    form=forms.AskDateForm()
    if request.method=='POST':
        form=forms.AskDateForm(request.POST)
        if form.is_valid():
            date=form.cleaned_data['date']
            studentdata=models.StudentExtra.objects.all().filter(user_id=request.user.id,status=T
rue)
            attendancedata=models.Attendance.objects.all().filter(date=date,cl=studentdata[0].c
l,roll=studentdata[0].roll)
            mylist=zip(attendancedata,studentdata)
            return
render(request,'university/student_view_attendance_page.html',{'mylist':mylist,'date':date}
)
        else:
            print('form invalid')
            return
render(request,'university/student_view_attendance_ask_date.html',{'form':form})

# for aboutus and contact us (by Anurag)
def aboutus_view(request):
    return render(request,'university/aboutus.html')

def contactus_view(request):
    sub = forms.ContactusForm()
    if request.method == 'POST':
        sub = forms.ContactusForm(request.POST)
        if sub.is_valid():
            email = sub.cleaned_data['Email']
            name=sub.cleaned_data['Name']
            message = sub.cleaned_data['Message']
            send_mail(str(name)+' || '+str(email),message,settings.EMAIL_HOST_USER,
settings.EMAIL_RECEIVING_USER, fail_silently = False)

```

```
        return render(request, 'university/contactussuccess.html')
    return render(request, 'university/contactus.html', {'form':sub})
```

forms.py

```
from django import forms
from django.contrib.auth.models import User
from . import models

#for admin
class AdminSigupForm(forms.ModelForm):
    class Meta:
        model=User
        fields=['first_name','last_name','username','password']

#for student related form
class StudentUserForm(forms.ModelForm):
    class Meta:
        model=User
        fields=['first_name','last_name','username','password']
class StudentExtraForm(forms.ModelForm):
    class Meta:
        model=models.StudentExtra
        fields=['roll','cl','mobile','fee','status']

#for teacher related form
class TeacherUserForm(forms.ModelForm):
    class Meta:
        model=User
        fields=['first_name','last_name','username','password']
class TeacherExtraForm(forms.ModelForm):
    class Meta:
        model=models.TeacherExtra
        fields=['salary','mobile','status']

#for Attendance related form
presence_choices=(('Present','Present'),('Absent','Absent'))
class AttendanceForm(forms.Form):
    present_status=forms.ChoiceField( choices=presence_choices)
    date=forms.DateField()

class AskDateForm(forms.Form):
    date=forms.DateField()
```

```
#for notice related form
class NoticeForm(forms.ModelForm):
    class Meta:
        model=models.Notice
        fields='__all__'

#for contact us page
class ContactusForm(forms.Form):
    Name = forms.CharField(max_length=30)
    Email = forms.EmailField()
    Message = forms.CharField(max_length=500,widget=forms.Textarea(attrs={'rows': 3,
'cols': 30}))
```

Frontend Code

Index.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en" dir="ltr">
<head>
    <meta charset="utf-8">
    <title></title>
    <style media="screen">
        .jumbotron {
            margin-top: 0px;
            margin-bottom: 0px;
            background-image: url('{% static "images/bgp.jpg" %}');
            background-size: cover;
            background-repeat: no-repeat;
        }
        .jumbotron h1 {
            text-align: center;
        }
        .alert {
            margin: 0px;
        }
        .glow {
            font-size: 70px;
            color: #ffffff;
```

```

text-align: center;
-webkit-animation: glow 1s ease-in-out infinite alternate;
-moz-animation: glow 1s ease-in-out infinite alternate;
animation: glow 1s ease-in-out infinite alternate;
}
@-webkit-keyframes glow {
  from {
    text-shadow: 0 0 10px #eeeeee, 0 0 20px #000000, 0 0 30px #000000, 0 0 40px
#000000, 0 0 50px #9554b3, 0 0 60px #9554b3, 0 0 70px #9554b3;
  }
  to {
    text-shadow: 0 0 20px #eeeeee, 0 0 30px #ff4da6, 0 0 40px #ff4da6, 0 0 50px #ff4da6, 0
0 60px #ff4da6, 0 0 70px #ff4da6, 0 0 80px #ff4da6;
  }
}
</style>
</head>
<body>
{% include "university/navbar.html" %}
<br>
<br>
<div class="jumbotron" style="margin-bottom: 0px;margin-top: 0px;">
  <br>
  <h1 class="display-4 glow">Welcome</h1>
  <br>
  <font color="black"><h2><p>Knowledge is key to success.</p></h2></font>
  <br><br><br><br><br>
  <p class="lead">
    <a class="btn btn-primary btn-lg" href="/studentsignup" role="button">Take
Admission</a>
  </p>
  <br><br>
</div>
<br>
<br>
<br><br>
{% include "university/admin_teacher_student_card.html" %}
<br><br>
<br>
<br>
<br>
<br>
{% include "university/footer.html" %}
</body>
</html>

```

Navbar.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
  <meta charset="utf-8">
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
  <style type="text/css">
    .bs-example {
      margin: 0px;
    }
    .navbar-brand {
      font-size: 20px;
      font-family: sans-serif;
    }
  </style>
</head>
<body>
  <div class="bs-example">
    <nav class="navbar navbar-expand-md bg-dark navbar-dark fixed-top">
      <a href="/" class="navbar-brand">UNIVERSITY MANAGEMENT SYSTEM</a>
      <button type="button" class="navbar-toggler" data-toggle="collapse" data-
target="#navbarCollapse">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse justify-content-between" id="navbarCollapse">
        <div class="navbar-nav">
          <a href="/" class="nav-item nav-link active">Home</a>
          <a href="/adminclick" class="nav-item nav-link">Admin</a>
          <a href="/teacherclick" class="nav-item nav-link">Teacher</a>
          <a href="/studentclick" class="nav-item nav-link">Student</a>
        </div>
        <div class="navbar-nav">
          <a href="/aboutus" class="nav-item nav-link">About Us</a>
          <a href="/contactus" class="nav-item nav-link">Contact Us</a>
        </div>
      </div>
    </nav>
  </div>
</body>
</html>
```



```

        </div>
    </div>
</nav>
</div>
</body>
</html>

```

footer.html

```

<!DOCTYPE html>
<html>

<head>

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css">

    <style>
        /*-----
        Social section
        -----*/
        footer {
            padding: 0px 0px 0px 0px;
            background-color: black;
            margin: 0px;
        }

        .fa {
            padding: 20px;
            font-size: 23px;
            width: 60px;
            text-align: center;
            text-decoration: none;
            margin: 5px 2px;
            border-radius: 50%;
        }

        .fa:hover {
            opacity: 0.5;
            text-decoration: none;
        }

        .fa-facebook {
            background: #3B5998;
            color: white;
            margin-top: 30px;

```

```

}

.fa-whatsapp {
  background: #25d366;
  color: white;
}

.fa-twitter {
  background: #55ACEE;
  color: white;
}

.fa-instagram {
  background: #125688;
  color: white;
}

p {
  text-align: center;
}
</style>
</head>

<footer>

<p>
  <a href="https://facebook.com" class="fa fa-facebook"></a>
  <a href="web.whatsapp.com" class="fa fa-whatsapp"></a>
  <a href="https://instagram.com" class="fa fa-instagram"></a>
  <a href="https://twitter.com" class="fa fa-twitter"></a>
</p>

<br>
<div class="container">
  <div class="row">
    <div class="col-md-12 col-sm-12">
      <div style="color:#ffffff;" class="wow fadeInUp footer-copyright">
        <p>Made in India <br>
        Copyright &copy; 2023 Anurag Vashist | MCA </p>
      </div>
    </div>
  </div>
</div>
</footer>
</html>

```

Admin_dashboard.html

```
{% extends 'university/adminbase.html' %}
{% load static %}
{% block content %}

<head>
<style media="screen">
.alert {
    margin-left: 25%;
    margin-right: 25%;
    padding: 20px;
    background-color: #f44336;
    color: white;
}

.w3-panel p {
    padding-top: 10px;
}

.closebtn {
    margin-left: 15px;
    color: white;
    font-weight: bold;
    float: right;
    font-size: 22px;
    line-height: 20px;
    cursor: pointer;
    transition: 0.3s;
}

.closebtn:hover {
    color: black;
}
</style>
<link rel="stylesheet" href="https://www.w3universitys.com/w3css/4/w3.css">

</head>
<br>

{%include 'university/admin_dashboard_cards.html'%}

<br><br>
<div class="w3-panel w3-blue ">
    <p>Notice Board</p>
</div><br>
{%for n in notice%}
```

```

<div class="alert">
  <span class="closebtn" onclick="this.parentElement.style.display='none';">&times;</span>
  <strong>{{n.date}} | | By :{{n.by}} </strong><br> {{n.message}}
</div>
{%endfor%}

<!--
  written By : Anurag Vashist
-->

{% endblock content %}

```

Teacher_dashboard.html

```

{% extends 'university/teacherbase.html' %}
{% load static %}

{% block content %}

<head>
<style media="screen">
  .alert {
    margin-left: 25%;
    margin-right: 25%;
    padding: 20px;
    background-color: #f44336;
    color: white;
  }

  .w3-panel p {
    padding-top: 10px;
  }

  .closebtn {
    margin-left: 15px;
    color: white;
    font-weight: bold;
    float: right;
    font-size: 22px;
    line-height: 20px;
    cursor: pointer;
    transition: 0.3s;
  }

```

```

        .closebtn:hover {
            color: black;
        }
    </style>
    <link rel="stylesheet" href="https://www.w3universitys.com/w3css/4/w3.css">

</head>
<br>

{%include 'university/teacher_dashboard_cards.html'%}

<br><br>
<div class="w3-panel w3-blue ">
    <p>Notice Board</p>
</div><br>
{%for n in notice%}
<div class="alert">
    <span class="closebtn" onclick="this.parentElement.style.display='none';">&times;</span>
    <strong>{{n.date}} || By : {{n.by}}</strong><br> {{n.message}}
</div>
{%endfor%}

{% endblock content %}

```

Student_dashboard.html

```

{% extends 'university/studentbase.html' %}
{% load static %}
{% block content %}

<head>
    <style media="screen">
        .alert {
            margin-left: 25%;
            margin-right: 25%;
            padding: 20px;
            background-color: #f44336;
            color: white;
        }

        .w3-panel p {
            padding-top: 10px;
        }

        .closebtn {
            margin-left: 15px;
            color: white;
            font-weight: bold;

```

```

        float: right;
        font-size: 22px;
        line-height: 20px;
        cursor: pointer;
        transition: 0.3s;
    }

    .closebtn:hover {
        color: black;
    }
</style>
<link rel="stylesheet"
href="https://www.w3universitys.com/w3css/4/w3.css">

</head>
<br>

{%include 'university/student_dashboard_cards.html'%}

<br><br>
<div class="w3-panel w3-blue ">
    <p>Notice Board</p>
</div><br>
{%for n in notice%}
<div class="alert">
    <span class="closebtn"
onclick="this.parentElement.style.display='none';">&times;</span>
    <strong>{{n.date}} || By : {{n.by}}</strong><br> {{n.message}}
</div>
{%endfor%}

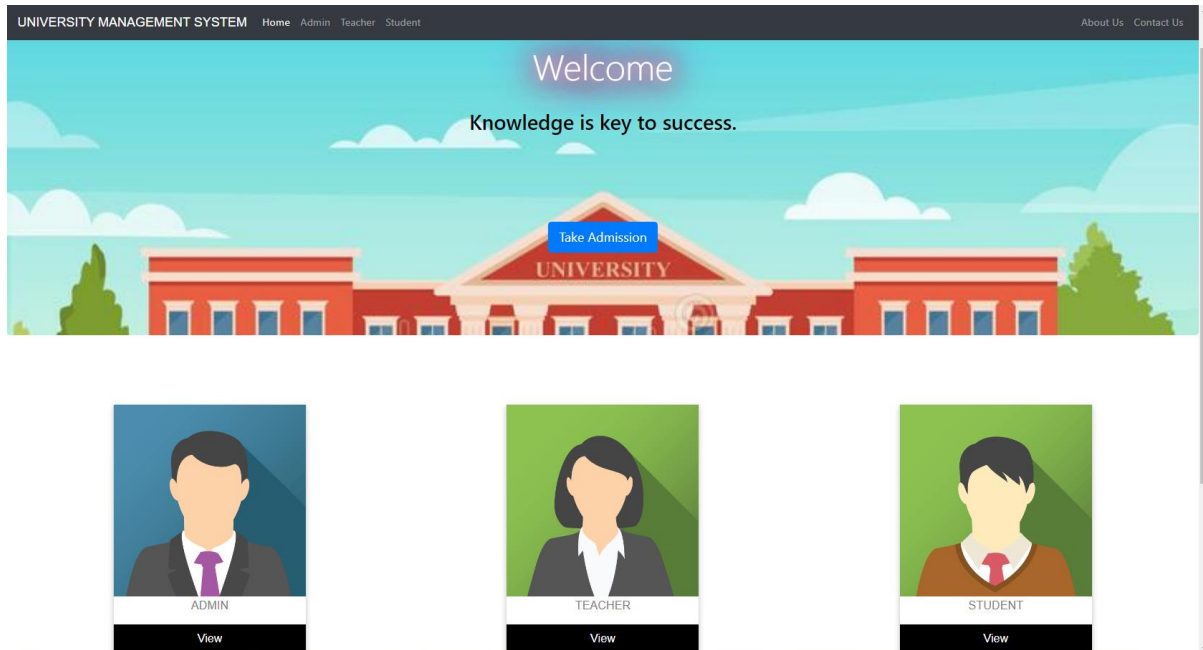
{% endblock content %}

```

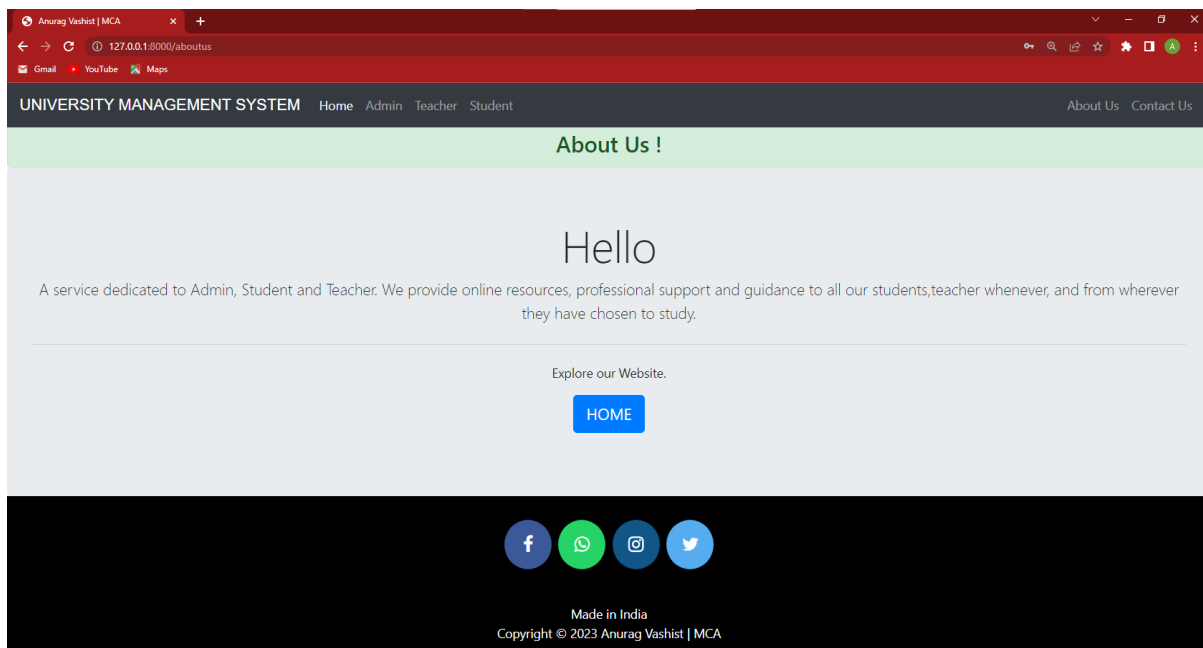
Chapter-5

Screenshot/Snapshot

Home Page



About Us Page



Admin Login Page

UNIVERSITY MANAGEMENT SYSTEM

[Home](#) [Admin](#) [Teacher](#) [Student](#)

[About Us](#) [Contact Us](#)

Admin Login Page

av22110

Login

Do not have account? [Signup here](#)

[f](#) [wa](#) [ig](#) [t](#)

Made in India
Copyright © 2023 Anurag Vashist | MCA

Teacher Login Page

Anurag Vashist | MCA

127.0.0.1:8000/teacherlogin

Gmail YouTube Maps

UNIVERSITY MANAGEMENT SYSTEM

[Home](#) [Admin](#) [Teacher](#) [Student](#)

[About Us](#) [Contact Us](#)

Teacher Login Page

av22110

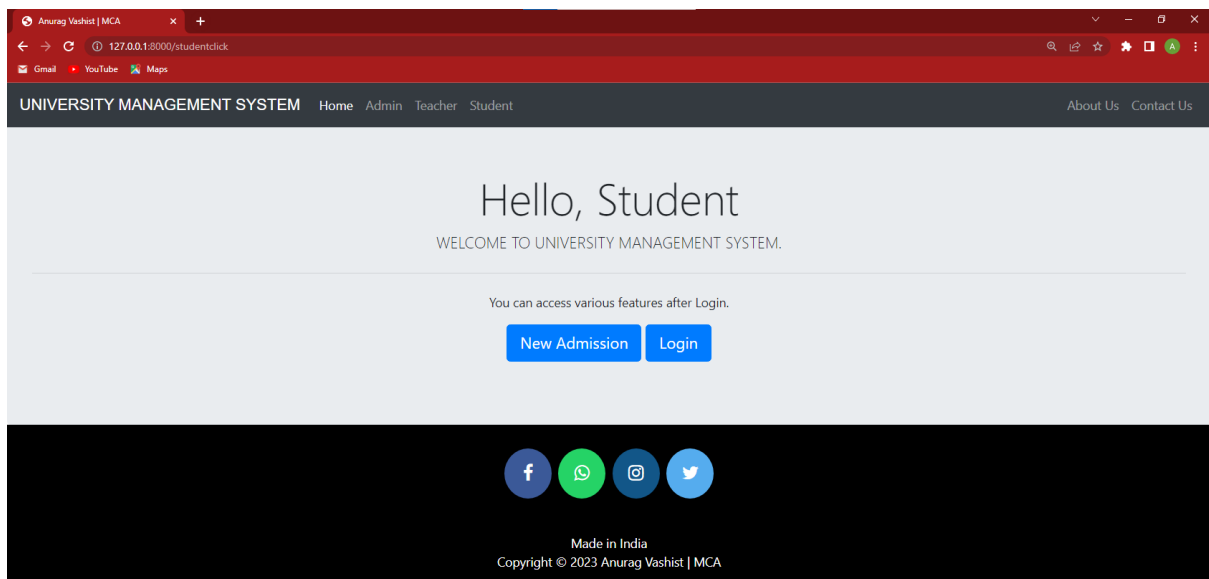
Login

Do not have account? [Signup here](#)

[f](#) [wa](#) [ig](#) [t](#)

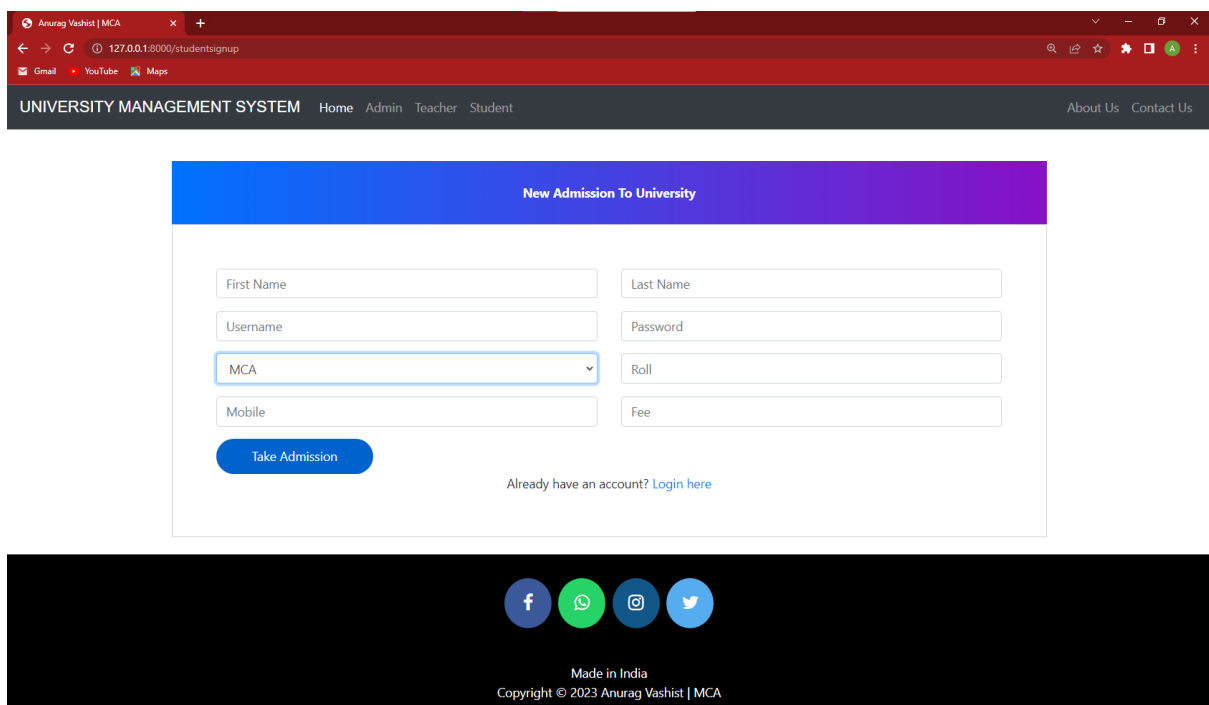
Made in India
Copyright © 2023 Anurag Vashist | MCA

Student Login Page



The screenshot shows a web browser window with the URL `127.0.0.1:8000/studentclick`. The page has a dark red header with the text "UNIVERSITY MANAGEMENT SYSTEM" and navigation links: Home, Admin, Teacher, Student, About Us, and Contact Us. The main content area is light gray and features the text "Hello, Student" and "WELCOME TO UNIVERSITY MANAGEMENT SYSTEM." Below this, a message states "You can access various features after Login." and there are two blue buttons: "New Admission" and "Login". At the bottom, there is a dark blue footer with social media icons for Facebook, WhatsApp, Instagram, and Twitter, and the text "Made in India" and "Copyright © 2023 Anurag Vashist | MCA".

Student Sign Up page



The screenshot shows a web browser window with the URL `127.0.0.1:8000/studentsignup`. The page has a dark red header with the text "UNIVERSITY MANAGEMENT SYSTEM" and navigation links: Home, Admin, Teacher, Student, About Us, and Contact Us. The main content area is white and features a blue header with the text "New Admission To University". Below this, there is a form with the following fields: First Name, Last Name, Username, Password, Roll, Mobile, and Fee. The "MCA" dropdown menu is selected. There is a blue button labeled "Take Admission" and a link that says "Already have an account? Login here". At the bottom, there is a dark blue footer with social media icons for Facebook, WhatsApp, Instagram, and Twitter, and the text "Made in India" and "Copyright © 2023 Anurag Vashist | MCA".

Admin SignUp page

The screenshot shows a web browser window with the URL `127.0.0.1:8000/adminsignup`. The page has a dark red header with the text "UNIVERSITY MANAGEMENT SYSTEM" and navigation links: Home, Admin, Teacher, Student, About Us, and Contact Us. The main content area is titled "Add New Admin To University" in a blue-to-purple gradient bar. Below this, there are four input fields: "First Name", "Last Name", "Username", and "Password". A blue "Submit" button is positioned below the "Username" field. A link "Already have an account? Login here" is located below the "Submit" button. At the bottom of the page, there is a black footer with social media icons for Facebook, WhatsApp, Instagram, and Twitter, followed by the text "Made in India" and "Copyright © 2023 Anurag Vashist | MCA".

Admin Dashboard

The screenshot shows a web browser window with the URL `127.0.0.1:8000/admin-dashboard`. The page has a dark red header with the text "UMS" and a "Logout" button. The main content area is divided into two sections. The top section contains four colored boxes: "Total Teacher" (1) and "Pending Teacher" (0) in blue; "Total Student" (1) and "Pending Students" (0) in green; "Teachers Salary" (100000) and "Pending Salary" (None) in orange; and "Student Fee" (40000) and "Pending Dues" (None) in pink. The bottom section is titled "Notice Board" and contains two red notification boxes. The first box says "May 24, 2023 ||By :Anurag" and "Our Project are to be submitted on 27 May 2023". The second box says "May 24, 2023 ||By :Anurag" and "All students are requested to submit assignment on time." At the bottom of the page, there is a black footer with social media icons for Facebook, WhatsApp, Instagram, and Twitter, followed by the text "Made in India" and "Copyright © 2023 Anurag Vashist | MCA".

Teacher's Dashboard

The screenshot shows the Teacher's Dashboard in a web browser. The browser's address bar displays the URL `127.0.0.1:8000/teacher-dashboard`. The page features a dark sidebar on the left with the 'UMS' logo and a 'Logout' button. The main content area displays the teacher's profile 'Pooja' with a circular avatar. Below the profile, there are four colored boxes containing personal information: Name (Pooja), Mobile (9999999999), Join Date (May 24, 2023), and Salary (₹ 100000). A blue 'Notice Board' section follows, containing two red notification boxes with the text: 'May 24, 2023 || By : Anurag Our Project are to be submitted on 27 May 2023' and 'May 24, 2023 || By : Anurag All students are requested to submit assignment on time.' The footer includes social media icons for Facebook, WhatsApp, Instagram, and Twitter, along with the text 'Made in India' and 'Copyright © 2023 Anurag Vashist | MCA'.

Student's Dashboard

The screenshot shows the Student's Dashboard in a web browser. The browser's address bar displays the URL `127.0.0.1:8000/student-dashboard`. The page features a dark sidebar on the left with the 'UMS' logo and a 'Logout' button. The main content area displays the student's profile 'Sunil' with a circular avatar. Below the profile, there are four colored boxes containing personal information: Name (Sunil), Mobile (8708177066), Roll (22109), and Fee (₹ 40000). A blue 'Notice Board' section follows, containing two red notification boxes with the text: 'May 24, 2023 || By : Anurag Our Project are to be submitted on 27 May 2023' and 'May 24, 2023 || By : Anurag All students are requested to submit assignment on time.' The footer includes social media icons for Facebook, WhatsApp, Instagram, and Twitter, along with the text 'Made in India' and 'Copyright © 2023 Anurag Vashist | MCA'.

Chapter-7

Implementation

In implementation, the designs are translated in to code. Computer programs are written using a conventional programming language or an application generator. Programming tools like compilers, interpreters, debuggers are used to generate the code. Different high-level languages like C, C++, Pascal, Java are used for coding. Detailed logical plans must be developed for all programs before they can be written, tested & documented. After each procedure & program are tested, the system is tested.

- (i) The conversion to the new system is made according to plan developed in the detailed design phase.
- (ii) This stage involves two activities:
 - (a) The User Training.
 - (b) The Conversion.

7.1 POST IMPLEMENTATION

When the system is implemented & conversion is complete, a review should be conducted to determine whether the system is meeting expectations & where improvements are needed. A post Implementation review measures the system's performance against pre-defined requirements.

Chapter-8

Testing

Testing is a process of executing a program with the intend of finding an error. A good test case is one that has high possibility of finding an undiscovered error. A successful test is one that uncovers an undiscovered error.

Testing also is then only way of finding out whether or not there is any error in a system. Therefore, testing is an important part of SDLC.

Testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved.

8.1 TYPES OF TESTING

1. **Unit testing:** -Each component or part of the program is tested individually to verify that the detail design for unit has been correctly implemented.
2. **Module testing:** -In large system a module may be a group of several programs or functions or sub-module testing.
3. **Integration testing:** -It is carried out to find problem in the interface between resembled units' module in a system.
4. **System testing:** - It is executing a program to check logic changes made in it and with the intention of finding errors.
5. **Acceptance testing:** -It is running system with the live data by actual user. Testing the user or reject the system.

Why do we need Software Testing?

Interviewers may ask you this question like “Why do we need Software Testing” or “Why is testing required” or “Why Software Testing”.

When I started in software testing, I had no idea what software testing is and why it is required. I also had no clue of where to start. Maybe you are in the same situation as I

was long back. I say, Software Testing is an art to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product.

What if there is no Software Testing in the Software Development process.

As per the current trend, due to constant change and development in digitization, our lives are improving in all areas. The way we work is also changed. We access our bank online, we do shop online, we order food online and many more. We rely on software's and systems. What if these systems turn out to be defective? We all know that one small bug shows huge impact on business in terms of financial loss and good will. To deliver a quality product, we need to have Software Testing in the Software Development Process.

Some of the reasons why software testing becomes very significant and integral part in the field of information technology areas follows.

- Cost effectiveness
- Customer Satisfaction
- Security
- Product Quality

8.2 Cost effectiveness

As a matter of fact, design defects can never be completely ruled out for any complex system. It is not because developers are careless but because the complexity of a system

is intractable. If the design issues go undetected, then It will become more difficult to trace back defects and rectify it. It will become more expensive to fix it. Sometimes, while fixing one bug we may introduce another one in some other module unknowingly. If the bugs can be identified in early stages of development, then it costs much less to fix them. That is why it is important to find defects in the early stages of the software development life cycle. One of the benefits of testing is cost effectiveness.

It is better to start testing earlier and introduce it in every phase of software development life cycle and regular testing is needed to ensure that the application is developed as per the requirement.

8.3 Customer Satisfaction

In any business, the ultimate goal is to give the best customer satisfaction. Yes, customer satisfaction is very important. Software testing improves the user experience of an application and gives satisfaction to the customers. Happy customers mean more revenue for a business. One of the reasons why software testing is necessary is to provide the best user experience.

8.4 Security

This is probably the most sensitive and vulnerable part of software testing. Testing (penetration testing & security testing) helps in product security. Hackers gain unauthorized access to data. These hackers steal user information and use it for their benefit. If your product is not secured, users won't prefer your product. Users

always look for trusted products. Testing helps in removing vulnerabilities in the product.

8.5 Product Quality

Software Testing is an art which helps in strengthening the market reputation of a company by delivering the quality product to the client as mentioned in the requirements specification documents.

Due to these reasons software testing becomes very significant and integral part of Software Development process.

Now let's move ahead and have a look at some of the principles of Software Testing.

Principles of Software Testing:

Testing of software consist of some principles that play a vital role while testing the project.

The Principles of Software Testing are as follows:

- Testing shows presence of defects
- Exhaustive testing is impossible
- Early testing
- Defect clustering

Chapter-9

CONCLUSION AND FUTURE SCOPE

This project has high scope to do. It meets all the upcoming demands of the system to manage and automate by all means. It integrates all the data and aspects of any system and can easily collaborate them with each other. This system is categorized into different streams. Various users can have access to it and provide their data accordingly. Once the data is entered it will be centralized and collaborated to all the departments. It also has a big scope in future to stream all the aspects of life. All the digitized system can be collaborated by data integration.

9.1 Future Enhancement

For all the projects there is always a scope of future enhancements. There are few enhancements I would like to make in future. They are as follows:

- **Add Classroom functionality, so students can submit their assignment on the platform**
- **Add Grading system**
- **Create a Mobile Application of this project, so it can be easily accessible**
- **Implement Fees Management System**

REFERENCES

Learned Frontend from - [HTML & CSS - YouTube](#)

Learned Backend from – Python and Django - Youtube