

Resource Management web application

A PROJECT REPORT

Submitted by

Abhishek kumar (reg-19RGVBCA037/19, Roll-214153)

Sunny kumar (reg-19RGVBCA037/19, Roll-2040734)

Navneet kumar (reg-20CCVGCS047/20, Roll-214164)

Ankesh kumar (reg- 20CCVGCS016/20, Roll-214133)

in partial fulfillment of the requirement for the award of the degree

of

BACHELOR OF COMPUTER APPLICATIONS

Submitted to



L.N.D COLLEGE, MOTIHARI

Babasaheb Bhimrao Ambedkar Bihar University

DECEMBER & 2023

CERTIFICATE

This is to certify that project report entitled “Resource. Management” which is submitted by Abhishek kumar, Sunny kumar, Navneet kumar, Shivam raj and Ankesh kumar in partial fulfillment of the requirement for the award of degree Bachelor of Computer Applications from L.N.D COLLEGE, MOTIHARI affiliated to Babasaheb Bhimrao Ambedkar Bihar University, Muzaffarpur is a record of the candidates own work carried out by them under my supervision. The matter embodied in this project report is original and has not been submitted for the award of any other degree.

**Mr. Prabhat kumar
Guide**

**Dr. Pinaki Laha
Head of the Department**

**Arun kumar
Principal**

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my project guide Mr. Prabhat Kumar as well as our course co-ordinator Dr. Pinaki Laha who gave me the golden opportunity to do this wonderful project on the topic Resource Management, which also helped me in doing a lot of research and I came to know about so many new things I am really thankful to them.

Secondly, I would also like to thank my parents and friends who helped me a lot in finalizing this project within the limited time frame.

Date:

Place:

Abhishek Kumar
reg:19RGVBCA037
roll: -214153

Sunny kumar
reg-:19RGVBCA036
roll:2040734

Ankesh kumar
reg-:20CCVGCS016
roll: -214133

Navneet kumar
reg-:20CCVGCS047
roll: -214164

DECLARATION

We, hereby declare that this submission is our own work and that, to the best of our Knowledge and belief. It contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where duly acknowledgement has been made in the text.

we take full responsibility for the authenticity and originality of this project, and we are aware of the consequences of any misrepresentation.

Abhishek Kumar
reg:19RGVBCA037
roll: -214153
signature:

Sunny kumar
reg-:19RGVBCA036
roll:2040734
signature:

Ankesh kumar
reg-:20CCVGCS016
roll: -214133
signature:

Navneet kumar
reg-:20CCVGCS047
roll: -214164
signature:

Table of Contents

	ABSTRACT
1.	Introduction
	1.1 Background
	1.2 Objectives
2.	Project Overview
	2.1 Scope
	2.2 Technology Stack
	2.3 Key Features Overview
3.	System Architecture
	3.1 Django Framework
	3.2 Backend Components
	3.3 Frontend Components
	3.4 Database Design
4.	User Management
	4.1 Account Creation
	4.2 Authentication System
5.	Online Exam Module
	5.1 Exam Creation
	5.2 Exam Taking
	5.3 Result Analysis
6.	Blog System
	6.1 Content Creation
	6.2 Categorization and Search
7.	Notes Repository
	7.1 Upload and Download Features
	7.2 Organization and Accessibility
	7.3 Integration with Other Modules
8.	Discussion Forum
	8.1 Discussion Creation
	8.2 Participation Features
9.	Implementation
	9.1 Development Environment Setup
	9.2 Django Best Practices
	9.3 Testing Strategies
10.	Conclusion

- 10.1 Achievements
- 10.2 Challenges
- 10.3 Future Enhancements

11. **Acknowledgments**

- 11.1 Contributions from Team Members

12. **References**

- 12.1 Django Documentation
- 12.2 Relevant Tutorials and Resources

Introduction

1.1 Background

The project stems come from the need for an integrated system that not only manages resources efficiently but also incorporates a blog system for knowledge sharing and guiding the students come from grassroot level, testing functionalities for quality assessment of one's knowledge and make revision from all subject collectively. a notes application for user productivity and discussion forum for solving student's Problems.

1.2 Objectives

"Unlocking potential, through education"

The main objective behind the Resource Management project is to take a step towards our goal. Generally, we found that the students come Frome villages and small towns don't have access to quality resources and guidance so in order to take a step we built website on which they can improve their productivity and solve their problems. The aim is to enhance collaboration, productivity, and reliability within the students come from such conditions.

Project Overview

2.1 Scope of the Project

The platform will be built using the Django framework for a robust backend, ensuring scalability and security. Users will have the ability to create accounts, access online exams with features like read and write blog etc. The project encompasses the creation of individual web apps for resource management, blogging, testing, and notes. Each web app is designed to function independently while allowing seamless interaction with the others.

While the initial scope is focused on essential features, the project allows for future enhancements, such as the integration of additional functionalities like a chat system, online attendance through face recognition, separate features for registering and selling product, fund rising, extending its features towards

humanities and science students and external API integrations, to adapt to evolving educational needs.

2.2 **Technology Stack:**

Backend Framework: Django

Frontend: HTML, CSS, JavaScript, bootstrap

Database: SQLite (for development)

User Authentication: Django Authentication System

2.3 Key Features Overview

User sign-in and sign-up:

Description: Users can create personalized accounts, for a secure and customized experience within the platform and by sign in their account they can create blog and post question on discussion forum.

Objective: Enable users to access features tailored to their educational needs and maintain a secure user environment.

Online Test Module:

Description: Facilitates the creation, administration, and analysis of online test with features such as adding editing, deleting question in database by log-in as admin, result analysis for students.

Objective: Enhance the assessment and evaluation process, providing students with real-time feedback on their performance and make ready for exam.

Blog System:

Description: Empowers students and teachers to create, read, and engage in discussions around various category of educational content

through a robust blogging system. Features include commenting, categorization, and search functionalities.

Objective: Foster collaboration and knowledge-sharing within the student's community, creating a dynamic learning environment.

Notes Repository:

Description: Provides students and admin with a centralized repository for uploading, downloading, and organizing educational materials, ensuring easy accessibility and resource management.

Objective: Streamline the organization of course-related materials, offering students a convenient way to manage and access study resources.

Discussion Forum:

- **Description:** Establishes a collaborative space for students to participate in discussions on a particular educational topic.
- **Objective:** Promote interactive learning and communication, creating a vibrant community of learners within the platform.

System architecture

3.1 Django framework:

Django is a high-level open-source web framework for building web applications in Python. It is suitable for a wide range of web development projects, from small-scale applications to large, complex systems. Its architecture follows the Model-View-Template (MVT) pattern, which is a slight variation of the more commonly known Model-View-Controller (MVC) pattern. In the Django context, the MVT pattern is comprised of the following components:

Model:

Definition: The Model represents the database structure and business logic of the application.

Responsibilities: Defines the structure of the database tables (schemas) through Django models. Contains methods to manipulate and retrieve data from the database.

Example: In a blogging application, the Model would define the structure of the "Post" and "category" tables, including fields like title, content, publication date, and images details.

View:

Definition: The View handles the presentation logic and user interface of the application.

Responsibilities: Receives user requests and interacts with the Model to fetch or update data. Determines which data and how much of it to display to the user. Renders the appropriate templates to generate HTML or other response formats.

Example: In the blogging application, the View would handle the request to view a blog post. It would interact with the Model to fetch the post data and then render an HTML template to display the post.

Template:

Definition: The Template represents the presentation layer, defining how the data received from the View should be presented.

Responsibilities: Contains HTML code mixed with Django template language syntax to dynamically render data. Defines the structure and layout of the final output sent to the user's browser. Supports the insertion of dynamic data and logic using template tags and filters.

Example: In the blogging application, the Template would define the structure of the HTML page displaying a blog post. It might include tags to insert the post title, content, and category information dynamically.

URL Routing: The framework uses a declarative URL routing system that makes it easy to map URLs to specific views. This promotes clean and readable URL patterns, enhancing maintainability.

Security: Django includes numerous security features by default, such as protection against SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). It encourages secure coding practices to mitigate common web security vulnerabilities.

Overall, Django is widely recognized for its emphasis on simplicity, reusability, and maintainability in web development.

3.2 Backend components

The backend of Resource Management website is implemented using Django web framework, a high-level python web framework that follow the MVT architecture. The backend is responsible for processing user requests, interacting with the database, and serving the necessary data to the frontend.

Django Framework: The core framework handles request processing, routing, and view rendering.

Models: Database models define the structure of the data, including User, Category, post, notes, and others.

Views: The views handle the logic for processing user requests and returning appropriate responses.

URL Routing: The URL patterns are defined to route requests to the corresponding views.

User authentication: session variables are used for login and logout functionality.

3.3 frontend components

The frontend of the Resource Management Website is built using standard web technologies, including HTML, CSS, bootstrap and JavaScript. The user interface is designed to be intuitive, responsive, and accessible mainly through bootstrap.

HTML, CSS, JavaScript:

Bootstrap is integrated into the frontend, utilizing its responsive design components for an adaptive user interface.

HTML templates are designed for each feature, ensuring a consistent and visually appealing user experience.

Custom CSS styles enhance the visual presentation and maintain brand consistency.

JavaScript is used for dynamic frontend interactions.

3.4 Database design

The Resource Management Website uses the SQLite database to persistently store data. The choice of SQLite offers simplicity and ease of use, making it suitable for small to medium-scale applications. The database schema is defined through Django models, ensuring a relational structure that captures the relationships between entities.

SQLite Database: *The lightweight, embedded database engine handles data storage.*

Django Models: *Models define entities such as User, Category, post, and notes, specifying fields and relationships.*

User management

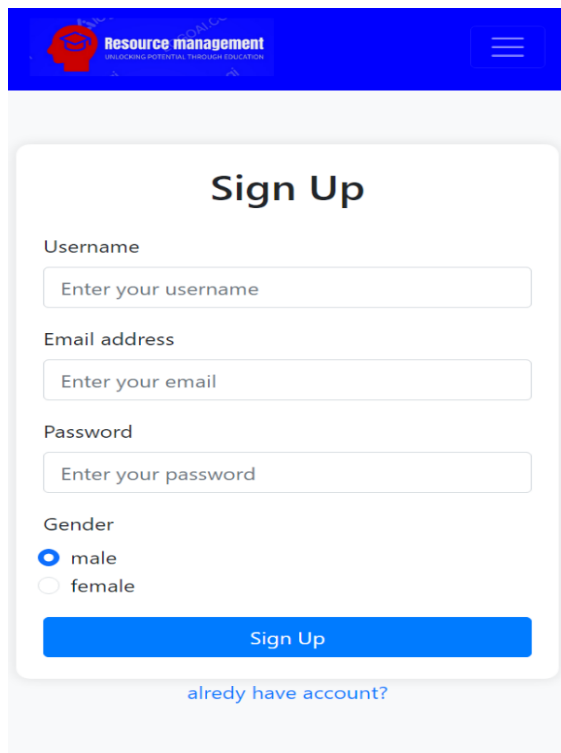
4.1 Account creation

User management is a critical aspect of the Resource Management Project, providing a secure and personalized experience for individuals

engaging in online tests, blogging, discussion forum, and notes uploading. This section outlines the implementation of user authentication, login, and logout features within the project.

User registration:

The user registration process allows individuals to create accounts on the platform. Key user information, such as username, email, and password, gender is



The screenshot shows a 'Sign Up' form with the following fields and elements:

- Username:** A text input field with the placeholder text 'Enter your username'.
- Email address:** A text input field with the placeholder text 'Enter your email'.
- Password:** A text input field with the placeholder text 'Enter your password'.
- Gender:** Two radio button options: 'male' (selected) and 'female'.
- Sign Up:** A prominent blue button.
- Link:** A link below the button that says 'already have account?'.

collected during registration process. Passwords are securely hashed and stored in database to protect user data. Once's user makes their registration he/she will be redirected to sign in form.

Login form:

A login form is provided on the platform's frontend, prompting users to enter their credentials. Validates user input and authenticates users against stored credentials.

Session Management:

Initiates a user session upon successful login.

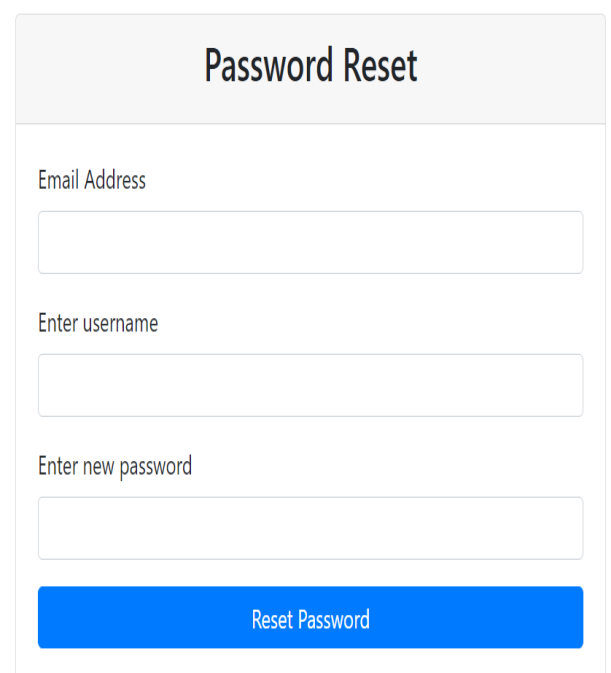
Maintains session data, allowing users to stay authenticated while navigating the platform.

User registration form

Logout features: Users can initiate a logout action, terminating their session securely. Clears session data to ensure the user is no longer authenticated.

Role-Based Access Control: in resource management we provide role base entry for example by login a student can take test, manage resource while an admin can perform crud operations on database.

Forgot Password: Implements a password recovery mechanism for users who forget their passwords.



The screenshot shows a 'Password Reset' form with the following fields and elements:

- Email Address:** A text input field.
- Enter username:** A text input field.
- Enter new password:** A text input field.
- Reset Password:** A prominent blue button.

Username

Password

Sign in


[don't have account?](#)
[forgotten password?](#)

Navigate

[Home](#)
[About Us](#)
[F.A.Q](#)

FOLLOW US



Technical Support 
777-234-098-127

Contact Form

Name

Email address

Message

Submit

Login form for user management

The sign form can display various error message depends on various conditions for example when an user enters wrong details then it display "sorry.. wrong details" while an user try to register himself with existing username or email then after signup validation it redirect to signin page with error message "user already exist with given username or email".

Interface of signin page when an user enters wrong details

Username

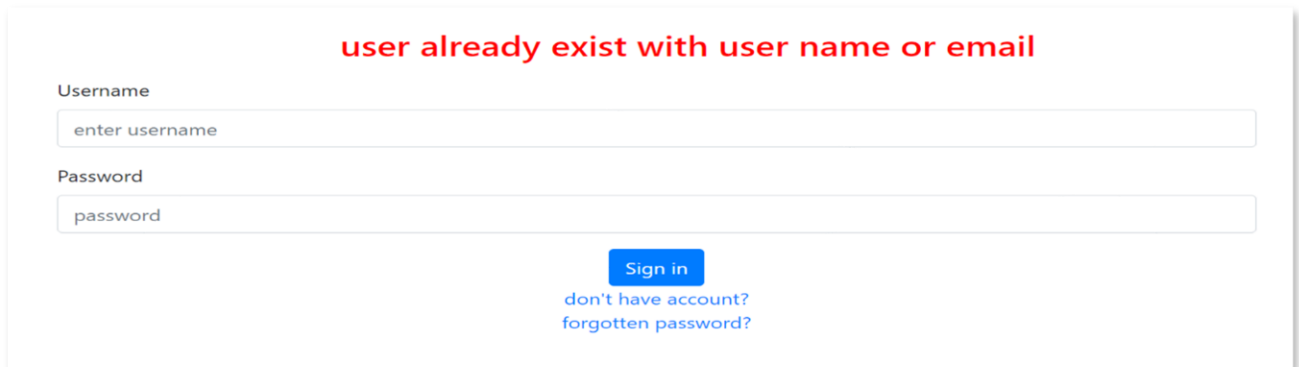
Password

Sign in

[don't have account?](#)
[forgotten password?](#)

sorry... wrong details

User try to register with existing username or email redirected to sign in page



user already exist with user name or email

Username
enter username

Password
password

[Sign in](#)

[don't have account?](#)
[forgotten password?](#)

Template file for sign in page: we have extended the base.html page for sign in page using extend template tag provided by Django.

Template for sign in page(sign_in.html)

```
{% extends "online_exam/base.html" %}
{% block title %}sign_in{% endblock %}
{% block content %}
<form action="/online_exam/sign_in_validation/" method="post">
  {% csrf_token %}
  <div class="container">
    <div class="row justify-content-evenly mt-5">
      <div class="col-10">
        <div class="mb-3">
          <h3 class="text-center mt-5 ml-5 p-2" style="color:
red;">{{message3}} {{message5}}</h3>
          <label for="formGroupExampleInput" class="form-
label">Username</label>
          <input type="text" class="form-control"
id="formGroupExampleInput" placeholder="enter username" name="username">
        </div>
        <div class="mb-3">
          <label for="formGroupExampleInput2" class="form-
label">Password</label>
          <input type="password" class="form-control"
id="formGroupExampleInput2" placeholder="password" name="password">
        </div>
        <div class="text-center">
          <button type="submit" class="btn btn-primary">Sign in</button>
        </div>
        <div class="text-center">
          <a href="/online_exam/sign_up/">don't have account?</a>
        </div>
        <div class="text-center">
```

```

                <a href="/online_exam/forgat_password/">forgotten
password?</a>
            </div>
        </div>
    </div>
</div>
</form>
<div>
    <h3 class="text-center mt-5 ml-5 p-2" style="color: red;">{{message}}
{{message2}}</h3>
</div>

{% endblock %}

```

views.py file for user management

```

def sign_in(request):

    d={}
    demo=user.objects.filter(user_name = 'admin')
    if not demo:
        create_admin()
    try:
        u=request.GET['error']
        if u == str(1):
            d['message']='sorry... wrong details'
        elif u == str(2):
            d['message2']='kya kar raha hai bhai..'
        elif u == str(3):
            d['message3']='user already exist with user name or email'
        elif u == str(4):
            d['message5']='wrong username or password'
        else:
            d['message']=' '
    except:
        pass
    return render(request,'online_exam/sign_in.html',d)

```

function for validating login credentials

```

def sign_in_validation(request):
    try:
        demo=user.objects.get(user_name=request.POST['username'] ,
password=request.POST['password'])
        demo.user_name
        request.session['sessionuser']=demo.user_name
        request.session['email']=demo.email
        request.session['gender']=demo.gender

```



```

        url='http://localhost:8000/online_exam/home/'
    except:
        url='http://localhost:8000/online_exam/sign_in/?error=1'
    return HttpResponseRedirect(url)

```

function for saving user data to database

```

def save_user(request):
    try:
        u=user.objects.filter(user_name=(request.POST['username']))
        if not u:
            User=user()
            User.user_name=request.POST['username']
            User.email=request.POST['email']
            User.password=request.POST['password']
            User.gender=request.POST['gridRadios']
            User.save()
            Return HttpResponseRedirect('http://localhost:8000/online_exam/sign_in/')
        else:
            return HttpResponseRedirect('/online_exam/sign_in/?error=3')
    except:
        return HttpResponseRedirect('/online_exam/sign_in/?error=3')

```

function for clearing session variable

```

def log_out(request):
    request.session.clear()
    return HttpResponseRedirect('/online_exam/sign_in/')

```

models for user management

```

from django.db import models
class user(models.Model):
    user_name=models.CharField(max_length=25,primary_key=True)
    email = models.EmailField(unique=True,null=False)
    password=models.CharField(max_length=25,null=False)
    gender=models.BooleanField(default=True)

```

as we describe above Django works on mvt architecture where render function is used for preparing output which takes html file, python dictionary and request object as input and return output using return keyword. When a user make request then request goes to url.py file and from their it goes to views.py file to executed mapped python function. User management system is built as part of online_exam module so necessary imported file is present in online_exam module.

4.2 Authentication system

Django's built-in authentication system is utilized for user account management.

Sessions are maintained for user login and logout functionality as you can see after sign in validation, we create session variable which is deleted when user logout.

Authentication middleware ensures secure access to different features based on user roles.

csrf mechanism is used for form submission.

Online exam module

5.1 Exam creation:

admin can create exams with multiple-choice question by login through sign in page. He can enjoy various features such as add, modify and delete the question from database.in resource management, there are two admin Pannel, one through login page and one by default provided by Django.

When user login as admin then resource management provide a different view and feature while when a student enters by login then it shows different view.

Welcome to RESOURCE MANAGEMENT

Welcome, Admin!

Create Blog

Compose and publish new blog posts for the community.

Create Blog

Question Management

Manage questions for online tests.

Create Question

View Question

Manage discussion forum

Navigate

Home

About Us

FAQ

FOLLOW US



Technical Support

777-234-098-127

Contact Form

Name

name

Email address

name@example.com

Message

Submit

Admin interface for managing online exam

question statement

enter question

optiona

optionb

optionc

optiond

answer

a

save

It is question setting page, when admin set a question, then he/she will be redirected to question data base where edit or modify option is available. this page also has same nav bar or footer.

Resource management [Home](#) [blog](#) [Test](#) [Notes](#) [Signin](#) [Signup](#) [services](#) set new question view question [create_blog](#) [log_out](#)

question	optiona	optionb	optionc	optiond	answer	edit	delete
1 .choose the oldest programming language?	b language	c language	java language	java script language	a	edit	delete
2 .which is not a language of 8th schedules?	hindi	english	java language	all of these	c	edit	delete
3 .ponnian selvom is also known as...	samundra gupta	akbra	rajendra chola	raj raj chol	d	edit	delete
4 .which is known for its fragrance?	red sandalwood	white sandalwood	both	none	b	edit	delete
5 .ooty is located in which indian state?	bihar	punjab	tamilnadu	uttrakhand	c	edit	delete
6 .choose the immutable element in python.....	list	tuple	string	more than one	d	edit	delete
7 .how many seats of lower house are reserved for p.o.k ?	24	35	20	25	c	edit	delete

Navigate

- [Home](#)
- [About Us](#)
- [F.A.Q](#)

FOLLOW US

Technical Support

777-234-098-127

Contact Form

Name

Email address

Message

[Submit](#)

Enter Title for blog

write blog

upload image No file chosen

category

[upload](#)

blog writing page

question statement

optiona

optionb

optionc

optiond

answer

[update](#)

question editing page

when admin want to edit question then he can click edit button before the question and question editing page will open where question and their options can be edited with update button which redirect to question database table.

If admin click on delete button, then after deleting question it will be redirected to question table. He can also upload a blog by upload button.

Models.py file

```
from django.db import models

class question(models.Model):
    qno=models.IntegerField(primary_key=True,auto_created=True)
    que=models.CharField(max_length=200,blank=True)
    optiona=models.CharField(max_length=100,null=True,blank=True)
    optionb=models.CharField(max_length=100,null=True,blank=True)
    optionc=models.CharField(max_length=100,null=True,blank=True)
    optiond=models.CharField(max_length=100,null=True,blank=True)
    ans=models.CharField(max_length=1)
```

this is model which is created to manage account creation

```
class user(models.Model):
    user_name=models.CharField(max_length=25,primary_key=True)
    email = models.EmailField(unique=True,null=False)
    password=models.CharField(max_length=25,null=False)
    gender=models.BooleanField(default=True)
```

this model is created for managing any particular message send by any user or student.

```
class message(models.Model):
    name=models.CharField(max_length=20)
    email=models.EmailField(primary_key=True,unique=True,null=False)
    message=models.TextField()
    def __str__(self):
        return self.name
```

View.py file of online exam

```
from django.shortcuts import render

from django.http import HttpResponseRedirect,HttpResponseRedirect

from online_exam.models import question,user,message
```

```
import random
```

```
def set_question(request):  
    return render(request,'online_exam/set_question.html')
```

this views function save question to database when admin send new question through question setting page

```
def save_question(request):  
    demo=question()  
    demo.que=request.POST['question']  
    demo.optiona=request.POST['optiona']  
    demo.optionb=request.POST['optionb']  
    demo.optionc=request.POST['optionc']  
    demo.optiond=request.POST['optiond']  
    demo.ans=request.POST['answer']  
    demo.save()  
    return HttpResponseRedirect('http://localhost:8000/online_exam/view_question/')
```

this manage when admin wants to see question database

```
def view_question(request):  
    try:  
        if request.session['sessionuser'] == 'admin':  
            qlist=question.objects.all()  
            return render(request,'online_exam/view_question.html',{'questions':qlist})  
        else:  
            return HttpResponseRedirect('/online_exam/sign_in/')  
    except:  
        return HttpResponseRedirect('/online_exam/sign_in/')
```

this views.py function run when admin edit question and click update button to save edit change

```
def edit_save(request):  
    n=int(request.POST['qnumber'])  
    Q=question.objects.get(qno=n)  
    Q.qno=n  
    Q.que=request.POST['question']  
    Q.optiona=request.POST['optiona']  
    Q.optionb=request.POST['optionb']  
    Q.optionc=request.POST['optionc']  
    Q.optiond=request.POST['optiond']  
    Q.ans=request.POST['answer']  
    Q.save()  
  
    return HttpResponseRedirect('http://localhost:8000/online_exam/view_question/')
```

this views.py function run when admin click edit button before question

```
def edit_question(request):  
    try:  
        if request.session['sessionuser'] == 'admin':  
            n=int(request.GET['qno'])  
            Q=question.objects.get(qno=n)  
            return render(request,'online_exam/edit_question.html',{'question':Q})  
        else:  
            return HttpResponseRedirect('/blog/main_blog/')  
    except:  
        return HttpResponseRedirect('/online_exam/sign_up/')
```

this views.py function run when admin click delete button before question

```
def delete_question(request):
```

```
try:
    n=int(request.GET['qno'])
    ques=question.objects.get(qno=n)
    ques.delete()
    return HttpResponseRedirect('http://localhost:8000/online_exam/view_question/')
except:
    return HttpResponseRedirect('http://localhost:8000/online_exam/sign_in/?error=2')
```

this views.py function is for providing interface after sign in

```
def home(request):
    try:
        if request.session['sessionuser']:
            return render(request,'online_exam/home.html')
        else:
            return HttpResponseRedirect('/online_exam/sign_in/')
    except KeyError:
        return HttpResponseRedirect('/online_exam/sign_in/')
    except:
        return HttpResponseRedirect('/online_exam/sign_in/?error=2')
```

```
def create_admin():
    demo=user()
    demo.user_name='admin'
    demo.password='admin123'
    demo.email='admin123@gmail.com'
    demo.gender='1'
    demo.save()
```

this views.py function run when admin logout

```
def log_out(request):
    request.session.clear()
```

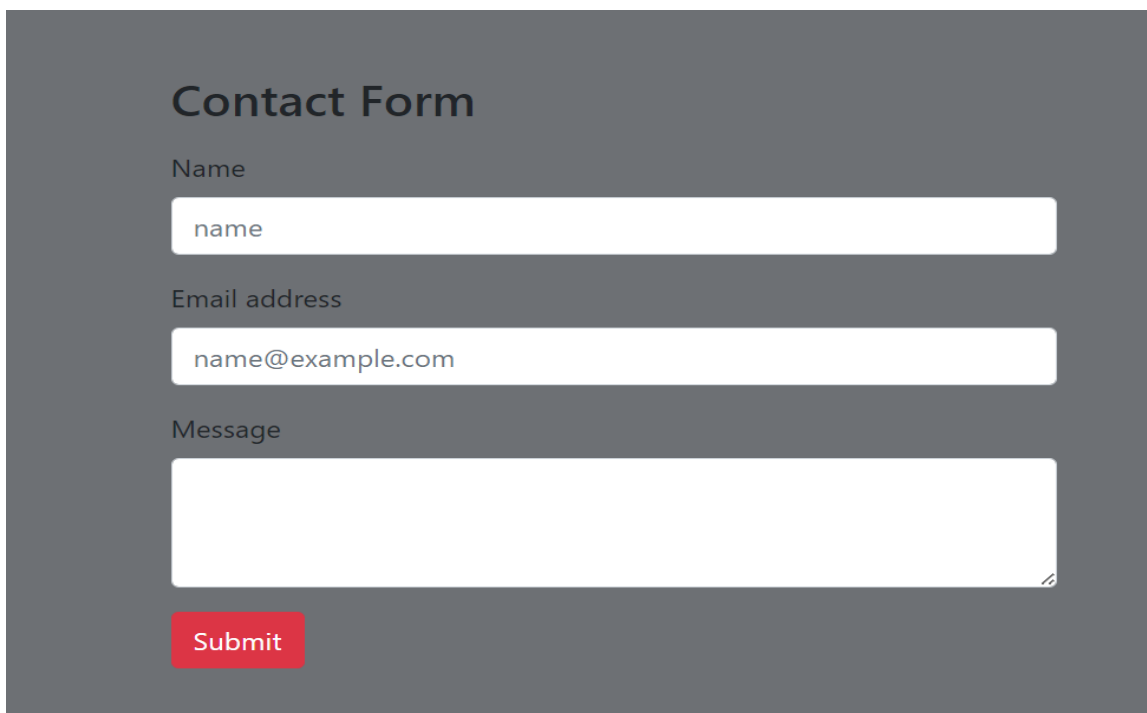


```
return HttpResponseRedirect('/online_exam/sign_in/')
```

this views.py function run when any user send message through footer section

```
def s_message(request):  
    try:  
        demo=message()  
        demo.name=request.POST['name']  
        demo.email=request.POST['email']  
        demo.message=request.POST['message']  
        demo.save()  
        return HttpResponseRedirect('/h2> message successfully submitted </h2>')  
    except:  
        return HttpResponseRedirect('/h2>some error occured .. try after some time</h2>')
```

when any user send message through contact form as shown in picture then the above views.py function will run .



The image shows a contact form on a dark grey background. The form is titled "Contact Form" in a large, bold, dark grey font. Below the title, there are three input fields: "Name" with the placeholder text "name", "Email address" with the placeholder text "name@example.com", and "Message" with a large empty text area. At the bottom of the form, there is a red button with the text "Submit" in white.

template of online exam module

this is the template page for admin and student which dynamically show content based on user.

```
{% extends "online_exam/base.html" %}
{% block title %}home{% endblock %}
{% block content %}
{% if request.session.sessionuser == 'admin' %}

<!-- navigator of admin-->
<ul class="nav justify-content-end">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page"
href="http://localhost:8000/online_exam/set_question/">set new question</a>
  </li>
  <li class="nav-item">
    <a class="nav-link link-warning"
href="http://localhost:8000/online_exam/view_question/">view question</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/blog/create_blog/">create_blog</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/online_exam/log_out/">log_out</a>
  </li>
</ul>
<!-- Admin Dashboard -->
<div class="container mt-4">
  <!-- Admin Options Section -->
  <div class="admin-options">
    <h2>Welcome to <span id="welcome" style="color:
blueviolet;"></span></h2>
    <p>Welcome, {{request.session.sessionuser |capfirst}}!</p>

    <!-- Create Blog Option -->
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Create Blog</h5>
        <p class="card-text">Compose and publish new blog posts for the
community.</p>
        <a href="/blog/create_blog/" class="btn btn-primary">Create
Blog</a>
      </div>
    </div>
  </div>
</div>
```

```

<!-- Question Management Options -->
<div class="card mt-3">
  <div class="card-body">
    <h5 class="card-title">Question Management</h5>
    <p class="card-text">Manage questions for online tests.</p>
    <a href="/online_exam/set_question/" class="btn btn-
success">Create Question</a>
    <a href="/online_exam/view_question/" class="btn btn-
danger">View Question</a>
    <a href="#edit-question" class="btn btn-warning">Manage
discussion forum</a>
  </div>
</div>

<!-- Other Admin Options -->
<!-- Add more admin-specific options here -->

</div>
</div>

{% else %}

<!-- navigator of user-->
<ul class="nav justify-content-end">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page"
href="/online_exam/start_test/">start_test</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/online_exam/log_out/">log-out</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/blog/create_blog/">create_blog</a>
  </li>
</ul>
<!-- user dashboard -->
<div class="container mt-4">
  <!-- students Options Section -->
  <div class="admin-options">
    <h2>Welcome to <span id="welcome" style="color:
blueviolet;"></span></h2>
    <p>Welcome, {{request.session.sessionuser |capfirst}}!</p>

    <!-- Create Blog Option -->
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Create Blog</h5>

```

```

        <p class="card-text">Compose and publish new blog posts for the
community.</p>
        <a href="/blog/create_blog/" class="btn btn-primary">Create
Blog</a>
    </div>
</div>

<!-- Question Management Options -->
<div class="card mt-3">
    <div class="card-body">
        <h5 class="card-title">Resource Management</h5>
        <p class="card-text">Manage Resources.</p>
        <a href="/online_exam/start_test/" class="btn btn-success">Start
Test</a>
        <a href="#" class="btn btn-danger">post question on forum</a>
        <a href="/notes/courses/" class="btn btn-warning">Explore Notes
</a>
    </div>
</div>

<!-- Other Admin Options -->
<!-- Add more admin-specific options here -->
{% endif %}

</div>

</div>
<!--cdn for auto text typing-->
<script src="https://unpkg.com/typed.js@2.1.0/dist/typed.umd.js"></script>
<script>
    var typed = new Typed('#welcome', {
    strings: ['RESOURCE MANAGEMENT', ],
    typeSpeed: 50,
    backspeed:80,
    loop:true
    });
</script>

{% endblock %}

```

This template page is passed to views.py (def set_question(request) function) and this template page has form which send question data to save_question(request): function of views.py which ultimately save data in database.

“Set_question.html”

```

{% extends "online_exam/base.html" %}
{% block title %} set_question{% endblock %}

```

```

{% block content %}
{% load static %}
<link rel="stylesheet" href="{% static 'css/question.css' %}">
{% if request.session.sessionuser == 'admin' %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end list-group list-group-horizontal-sm">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page"
href="http://localhost:8000/online_exam/set_question/">set new question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link link-warning"
href="http://localhost:8000/online_exam/view_question/">view question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
        </li>
        <li class="nav-item">
            <a class="nav-link"
href="http://localhost:8000/online_exam/log_out/">log_out</a>
        </li>

    </ul>

{% else %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end list-group list-group-horizontal-sm">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page"
href="http://localhost:8000/online_exam/start_test/">start_test</a>
        </li>
        <li class="nav-item">
            <a class="nav-link"
href="http://localhost:8000/online_exam/log_out/">log-out</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
        </li>
    </ul>
{% endif %}

<!--edit question-->

<div class="container-fluid">
    <div class="row">

```

```

<div class="col-12 col-sm-12">

    <form action="http://localhost:8000/online_exam/save_question/"
method="post">
    {% csrf_token %}
    <div class="title"> question statement</div>
    <div class="data">
        <textarea name="question" id="" cols="50" rows="5"
placeholder="enter question"></textarea>
    </div>
    <div class="title">optiona</div>
    <div class="data">
        <input type="text" name="optiona">
    </div>
    <div class="title">optionb</div>
    <div class="data">
        <input type="text" name="optionb">
    </div>
    <div class="title">optionc</div>
    <div class="data">
        <input type="text" name="optionc">
    </div>
    <div class="title">optiond</div>
    <div class="data">
        <input type="text" name="optiond">
    </div>
    <div class="title">answer</div>
    <div class="data">
        <select name="answer" id="">
            <option value="a">a</option>
            <option value="b">b</option>
            <option value="c">c</option>
            <option value="d">d</option>
        </select>
    </div>
    <input type="submit" value="save" class="btn btn-primary mt-2">

    </form>
</div>
</div>
</div>
{% endblock %}

```

This template page is passed when admin request to view question database

```
{% extends "online_exam/base.html" %}
{% block title %}question_database{% endblock %}
{% block content %}

{% if request.session.sessionuser == 'admin' %}

    navigator of admin nav bar code is removed for space management
-->
{% else %}

<!-- navigator of student same as which shown in home page -->

{% endif %}

<!--fetching question from question_database-->
<div class="table-responsive">

    <table class="table table-striped table-hover">
        <tr>
            <th>question</th>
            <th>optiona</th>
            <th>optionb</th>
            <th>optionc</th>
            <th>optiond</th>
            <th>answer</th>
            <th>edit</th>
            <th>delete</th>
        </tr>
        {% for q in questions %}
        <tr>
            <td class="que">{{forloop.counter}} .{{q.que}}</td>
            <td>{{q.optiona}}</td>
            <td>{{q.optionb}}</td>
            <td>{{q.optionc}}</td>
            <td>{{q.optiond}}</td>
            <td>{{q.ans}}</td>
            <td><a
href="http://localhost:8000/online_exam/edit_question/?qno={{q.qno}}">edit</a></td>
            <td><a
href="http://localhost:8000/online_exam/delete_question/?qno={{q.qno}}">delete</a></td>

            </tr>

        {% endfor %}

    </table>
```

```
</div>
{% endblock %}
```

Template page which is used to render create_blog functionality called by def create_blog(request): function of blog_views.py file. This template has form which send blog data for storage to def save_post(request): function of same.

```
{% extends "online_exam/base.html" %}
{% block title %}create_blog{% endblock %}
{% block content %}
{% if request.session.sessionuser == 'admin' %}
```

```
<!-- navigator of admin-->
<ul class="nav justify-content-end">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page"
href="http://localhost:8000/online_exam/set_question/">set new question</a>
  </li>
  <li class="nav-item">
    <a class="nav-link link-warning"
href="http://localhost:8000/online_exam/view_question/">view question</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/blog/create_blog/">create_blog</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/online_exam/log_out/">log_out</a>
  </li>
</ul>
```

```
{% else %}
```

```
<!-- navigator of admin-->
<ul class="nav justify-content-end">
  <li class="nav-item">
    <a class="nav-link active" aria-current="page"
href="/online_exam/start_test/">start_test</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/online_exam/log_out/">log-out</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/blog/create_blog/">create_blog</a>
  </li>
```



```

    </ul>
{% endif %}
<!--blog writing-->
    <div class="container">
        <form action="/blog/save_blog/" method="post" enctype="multipart/form-
data">
            {% csrf_token %}
            <div class="title pl-4 ">Enter Title for blog</div>
            <div class="data pl-4">
                <textarea name="title" id="" cols="40" rows="3"></textarea>
            </div>
            <div class="title pl-4">write blog</div>
            <div class="title pl-4"><textarea name="content" id="" cols="40"
rows="25"></textarea></div>
            <div class="title mt-4 pl-4">upload image</div>
            <div class="data pl-4">
                <input type="file" name="picture" >
            </div>
            <div class="title mt-3 pl-4">category</div>
            <div class="data pl-4">
                <select name="category" >
                    {% for c in cat %}

                    <option value="{{c.no}}">{{c.name}}</option>
                    {% endfor %}

                </select>
            </div>
            <input type="submit" value="upload" class="btn btn-success mt-3">
        </form>
    </div>
{% endblock %}

```

This template page is use to render edit question functionality .when admin enter edit button before question then this html page is used by def edit_question(request): function of views.py file of online exam.after editig this send data to edit_save(request): function to updte database.

```

        "Edit_question.html"
{% extends "online_exam/base.html" %}
{% block title %}edit_question{% endblock %}
{% block content %}
{% if request.session.sessionuser == 'admin' %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end list-group list-group-horizontal-sm">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page"
href="/online_exam/set_question/">set new question</a>

```

```

    </li>
    <li class="nav-item">
      <a class="nav-link link-warning"
href="/online_exam/view_question/">view question</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/blog/create_blog/">create_blog</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/online_exam/log_out/">log_out</a>
    </li>
  </ul>

  {% else %}

  <!-- navigator of admin-->
  <ul class="nav justify-content-end list-group list-group-horizontal-sm">
    <li class="nav-item">
      <a class="nav-link active" aria-current="page"
href="/online_exam/start_test/">start_test</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/online_exam/log_out/">log-out</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="/blog/create_blog/">create_blog</a>
    </li>
  </ul>
  {% endif %}
<!--edit question section-->
<div class="container-fluid">
  <form action="/online_exam/edit_save/" method="post">
    {% csrf_token %}
    <input type="hidden" name="qnumber" value="{{question.qno}}">
    <div class="title"> question statement</div>
    <div class="data">
      <textarea name="question" id="" cols="50" rows="5"
>{{question.que}}</textarea>
    </div>
    <div class="title">optiona</div>
    <div class="data">
      <input type="text" name="optiona" value="{{question.optiona}}">
    </div>
    <div class="title">optionb</div>
    <div class="data">
      <input type="text" name="optionb" value="{{question.optionb}}">
    </div>
  </form>

```

```

<div class="title">optionc</div>
<div class="data">
  <input type="text" name="optionc" value="{{question.optionc}}">
</div>
<div class="title">optiond</div>
<div class="data">
  <input type="text" name="optiond" value="{{question.optiond}}">
</div>
<div class="title">answer</div>
<div class="data">
  <select name="answer" id="">
    <option value="a" {% if question.ans == 'a' %}selected{% endif
%}>a</option>
    <option value="b" {% if question.ans == 'b' %}selected{% endif
%}>b</option>
    <option value="c" {% if question.ans == 'c' %}selected{% endif
%}>c</option>
    <option value="d" {% if question.ans == 'd' %}selected{% endif
%}>d</option>
  </select>
</div>
<div>
  <input type="submit" value="update" class="btn btn-secondary mt-
3">
</div>

</form>
</div>
{% endblock %}

```

5.2 exam taking

student can give exams with multiple-choice question by login through sign in page. He can enjoy various features such as add blog, post question, start test.

When user login as student then resource management provide a different view and features while when an admin enters by login then it shows different view.

Welcome to RESOURCE M|

Welcome, Abhishek kumar!

Create Blog

Compose and publish new blog posts for the community.

Create Blog

Resource Management

Manage Resources.

Start Test

post question on forum

Explore Notes

Navigate

[Home](#)

[About Us](#)

[F.A.Q](#)

FOLLOW US



Technical Support 📧

777-234-098-127

Contact Form

Name

name

Email address

name@example.com

Message

Submit

Views.py file for online_exam

when user sign in as student then above view is render dynamically by using this home function and home.html template file which is used for admin interface.

```
def home(request):
    try:
        if request.session['sessionuser']:
            return render(request, 'online_exam/home.html')
        else:
            return HttpResponseRedirect('/online_exam/sign_in/')
    except KeyError:
        return HttpResponseRedirect('/online_exam/sign_in/')
    except:
```

```
return HttpResponseRedirect('/online_exam/sign_in/?error=2')
```

THIS FUNCTION EXECUTE WHEN AN STUDENT CLICK ON START TEST BUTTON

```
def start_test(request):
```

```
    try:
```

```
        if request.session['sessionuser']:
```

```
            qlist=list(question.objects.all())
```

```
            random.shuffle(qlist)
```

```
            qpool=qlist[:5]
```

```
            return render(request,'online_exam/start_test.html',{'qpool':qpool})
```

```
        else:
```

```
            return HttpResponseRedirect('/online_exam/sign_in/')
```

```
    except:
```

```
        return HttpResponseRedirect('/online_exam/sign_in/')
```

*(start_test.html)template file is used by def start_test(request):
function for rendering*

```
{% extends "online_exam/base.html" %}
```

```
{% block title %}test{% endblock %}
```

```
{% block content %}
```

```
{% if request.session.sessionuser == 'admin' %}
```

```
    <!-- navigator of admin-->
```

```
    <ul class="nav justify-content-end">
```

```
        <li class="nav-item">
```

```
            <a class="nav-link active" aria-current="page"
```

```
href="http://localhost:8000/online_exam/set_question/">set new question</a>
```

```
        </li>
```

```
        <li class="nav-item">
```

```
            <a class="nav-link link-warning"
```

```
href="http://localhost:8000/online_exam/view_question/">view question</a>
```

```
        </li>
```

```
        <li class="nav-item">
```

```
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
```

```

        </li>
        <li class="nav-item">
            <a class="nav-link"
href="http://localhost:8000/online_exam/log_out/">log_out</a>
            </li>

    </ul>

    {% else %}

    <!-- navigator of admin-->
    <ul class="nav justify-content-end">
        <li class="nav-item">
            <a class="nav-link active" aria-current="page"
href="http://localhost:8000/online_exam/start_test/">start_test</a>
            </li>
        <li class="nav-item">
            <a class="nav-link"
href="http://localhost:8000/online_exam/log_out/">log-out</a>
            </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
            </li>
    </ul>
    {% endif %}

<div class="container-fluid">
    <div class="row pl-4">
        <div class="col-10">

            <form action="http://localhost:8000/online_exam/test_result/"
method="post">
                {% csrf_token %}
                {% for q in qpool %}
                <input type="hidden" name="qno{{q.qno}}" value="{{q.qno}}">
                <div class="question">{{forloop.counter}} .{{q.que}}</div>
                <div class="option"><input type="radio" name="ans{{q.qno}}" value="a"
>{{q.optiona}}</div>
                <div class="option"><input type="radio" name="ans{{q.qno}}" value="b"
>{{q.optionb}}</div>
                <div class="option"><input type="radio" name="ans{{q.qno}}" value="c"
>{{q.optionc}}</div>
                <div class="option"><input type="radio" name="ans{{q.qno}}" value="d"
>{{q.optiond}}</div>
                {% endfor %}
                <input type="submit" value="submit" class=" btn btn-primary mt-3" >
            </div>

```

```

    </div>
  </form>
</div>
{% endblock %}

```

the interface when a student clicks on start_test then randomly a chunk of 5 question comes from database in multiple choice questions. When one refresh page then questions will be changed.

Resource management Home blog Test Notes Signin Signup services

start_test log-out create_blog

1 .choose the oldest programming language?
 b language
 c language
 java language
 java script language

2 .choose the immutable element in python....
 list
 tuple
 string
 more than one

3 .how many seats of lower house are reserved for p.o.k ?
 24
 35
 20
 25

4 .which is not a language of 8th schedules?
 hindi
 english
 java language
 all of these

5 .ooty is located in which indian state?
 bihar
 punjab
 tamilnadu
 uttrakhand

submit




Navigate


[Home](#)

[About Us](#)

[FAQ](#)

FOLLOW US

Technical Support 

777-234-098-127

Contact Form

Name

Email address

Message

Submit

5.3 Result Analysis:

When a student submits the test then his test data is sent to test result function for result analysis, where it shows where we make a mistake and right wrong questions.

Views.py file of online_exam

```
def test_result(request):
    try:
        if request.session['sessionuser']:
            total_wrong=0
            total_write=0
            attempted_ques=0
            wq=[]
            qlist=[]
            for k in request.POST:
                if k.startswith('qno'):
                    qlist.append(int(request.POST[k]))
            for n in qlist:
                try:
                    q=question.objects.get(qno=n)
                    if q.ans == request.POST['ans'+str(n)]:
                        total_write+=1
                    else:
                        total_wrong+=1
                        wq.append(q)
                        attempted_ques+=1
                except:
                    pass
            d={
                'total_wrong':total_wrong ,
                'total_wright':total_write ,
                'attempted_ques':attempted_ques,
                'wq':wq
            }
            return render(request,'online_exam/test_result.html',d)
    except:
        return
HttpResponseRedirect('http://localhost:8000/online_exam/sign_in/')
```

template file which is used for result analysis

```
{% extends "online_exam/base.html" %}
{% block title %}test_result{% endblock %}
{% block content %}
{% if request.session.sessionuser == 'admin' %}

<!-- navigator of admin-->
<ul class="nav justify-content-end">
```



```

        <li class="nav-item">
            <a class="nav-link active" aria-current="page"
href="http://localhost:8000/online_exam/set_question/">set new question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link link-warning"
href="http://localhost:8000/online_exam/view_question/">view question</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="/blog/create_blog/">create_blog</a>
        </li>
        <li class="nav-item">
            <a class="nav-link"
href="http://localhost:8000/online_exam/log_out/">log_out</a>
        </li>

```

```
</ul>
```

```
{% else %}
```

```

<!-- navigator of admin-->
<ul class="nav justify-content-end">
    <li class="nav-item">
        <a class="nav-link active" aria-current="page"
href="http://localhost:8000/online_exam/start_test/">start_test</a>
    </li>
    <li class="nav-item">
        <a class="nav-link"
href="http://localhost:8000/online_exam/log_out/">log-out</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="/blog/create_blog/">create_blog</a>
    </li>
</ul>
{% endif %}

```

```
<table class="table table-dark table-striped pl-3 mt-3">
```

```
<tr>
```

```
<th>total question:</th>
```

```
<th>5</th>
```

```
</tr>
```

```
<tr>
```

```
<th>attempted question</th>
```

```
<th>{{attempted_ques}}</th>
```

```
</tr>
```

```
<tr>
```

```
<th>wright question</th>
```

```
<th>{{total_wright}}</th>
```

```

        </tr>
        <tr>
            <th>wrong question</th>
            <th>{{total_wrong}}</th>
        </tr>
    </table>
    <table class="table table-striped table-hover pl-3">
        <tr>
            <th class="text-center" style="color: red;"> Questions in which
you make mistake <h1>&#x1F622;</h1></th>
        </tr>

        {% for q in wq %}
        <tr>
            <td>

                <div>{{forloop.counter}} .{{q.que}}</div>
                <div>{{q.optiona}} <input type="radio" {% if q.ans == 'a' %} checked {%
endif %}></div>
                <div>{{q.optionb}} <input type="radio" {% if q.ans == 'b' %} checked {%
endif %}></div>
                <div>{{q.optionc}} <input type="radio" {% if q.ans == 'c' %} checked {%
endif %}></div>
                <div>{{q.optiond}} <input type="radio" {% if q.ans == 'd' %} checked {%
endif %}></div>
            </td>
        </tr>
        {% endfor %}
        <tr>
            <td class="text-center">
                <a href="/online_exam/start_test/">test again</a>
            </td>
        </tr>
    </table>
{% endblock %}

```

total question:	5
attempted question	5
wright question	2
wrong question	3

Questions in which you make mistake



1 .choose the oldest programming language?

- b language
- c language
- java language
- java script language

2 .choose the immutable element in python....

- list
- tuple
- string
- more than one

3 .which is not a language of 8th schedules?

- hindi
- english
- java language
- all of these

[test again](#)

Navigate


[Home](#)

[About Us](#)

[F.A.Q](#)

FOLLOW US



Technical Support 

777-234-098-127

Contact Form

Name

name

Email address

name@example.com

Message

Submit

Blog system

The Blog System is a key feature of the Resource Management Project, providing users with a collaborative platform for sharing insights, knowledge, and fostering community engagement. Developed within the Django framework, this module enhances the educational experience by facilitating communication, information exchange, and collaborative learning.

6.1 content creation:

Users can create and publish blog posts on various different topic. Supports the inclusion of text and images for comprehensive content. When user click blog tab then main blog section is open which show 5 recent blogs. from main blog page any user can read the post and if he likes the post then goes to detailed blog page where he can read blog in details. The architecture of blog features is also based on mvt.

Models.py file of blog app

```
from django.db import models
from django.utils.html import format_html
from tinymce.models import HTMLField

class category(models.Model):
    no=models.IntegerField(primary_key=True,auto_created=True)
    name=models.CharField(max_length=100,null=True)
    discription=models.TextField()
    picture=models.ImageField(upload_to='category/')

    def __str__(self):
        return self.name

    def img_tag(self):
        return format_html(''.format(self.picture))

class post(models.Model):
    post_no=models.IntegerField(primary_key=True,auto_created=True)
    title=models.CharField(max_length=100)
    content=HTMLField()
    picture=models.ImageField(upload_to='post/')
    cat=models.ForeignKey(category,on_delete=models.CASCADE)
    date=models.DateTimeField(auto_now_add=True,null=True)

    def __str__(self):
        return self.title

    def abhi(self):
        return format_html(''.format(self.picture))
```

views.py file of blog app

```
from .models import *
```

the main_blog function of views.py file is responsible for rendering 5 blogs on main blog page filtered by date.

```
def main_blog(request):
    posts=post.objects.all().order_by('-date')[:5]
    cat=category.objects.all()
    return render(request,'blog/blog.html',{'posts':posts,'cats':cat})
```

the detailed_blog function of views.py file is responsible for rendering details blogs from main blog page when user clicks on read more.

```
def detailed_blog(request,postid):
    p=post.objects.get(post_no=postid)
    c=category.objects.all()
    return render(request,'blog/detailedpage.html',{'p':p,'cats':c})
```

the category_blog function of views.py file is responsible for rendering categories wise blogs from main blog page or details blog page.

```
def category_blog(request,catid):
    c=category.objects.get(no=catid)
    p=post.objects.filter(cat=c)
    return render(request,'blog/category_blog.html',{'c':c,'p':p})
```

the create_blog function of views.py file is responsible for providing an interface where students or admin can create blog.

```
def create_blog(request):
    cat=category.objects.all()
    return render(request,'blog/create_blog.html',{'cat':cat})
```

the save_blog function of views.py file is responsible for saving blog post to the database.

```
def save_blog(request):
    try:
        demo=post()
        demo.title=request.POST['title']
        demo.content=request.POST['content']
        demo.picture=request.FILES['picture']
        n=int(request.POST['category'])
        c_id=category.objects.get(no=n)
        demo.cat=c_id
        demo.save()
        return HttpResponseRedirect('http://localhost:8000/blog/main_blog/')
```

```
except:
    return
HttpResponseRedirect('http://localhost:8000/online_exam/sign_in/')
```

template file of blog app

this page of template file is used by main_blog function of views.py file of blog app to show main blog page.

```
{% extends "blog/base.html" %}
{% block title %}welcome to the blog{% endblock %}
{% block content %}
<div class="container">

    <div class="owl-carousel">

        {% for cat in cats %}
        <div class="card">

            <div class="card-body text-center">
                <h6 class="card-title">{{cat.name}}</h6>
                <p class="card-text">{{cat.discription |truncatechars:20}}</p>
                <a href="/blog/category_blog/{{cat.no}}" class="btn btn-
primary">view</a>
            </div>
        </div>
        {% endfor %}

    </div>
</div>
<div class="container ">
    <div class="row ">
        {% for post in posts %}
        <div class="col-md-8 pt-4">
            <h3>{{post.title |upper}}</h3>
            <p>{{post.content |truncatewords:70 }}</p>
            <div class="container pt-4 pl-5 text-center ">
                <a href="/blog/detailed_blog/{{post.post_no}}" class="btn btn-
primary ">read more...</a>
            </div>
        </div>
        <div class="col-md-4 pt-4">
            
        </div>
    </div>
</div>
```

```
        {% endfor %}
    </div>
</div>
{% endblock %}
```

this page of template file is used by category_blog function of views.py file of blog app to show blog category wise.

```
{% extends "blog/base.html" %}
{% block title %}blog{% endblock %}
```

```
{% block content %}
```

```
<div class="container-fluid">
    <div class="row justify-content-evenly text-center">
        <div class="col-lg-8 col-md-12">
            <h3> {{p.title |capfirst}}</h3>
            
            <p class="pt-4 "> {{p.content | safe}}</p>

        </div>
        <div class="col-lg-3 col-md-12">

            {% for cat in cats %}
                <div class="card">

                    <div class="card-body text-center">
                        <h6 class="card-title">{{cat.name}}</h6>
                        <p class="card-text">{{cat.discription
|truncatechars:20}}</p>
                        <a href="/blog/category_blog/{{cat.no}}" class="btn btn-
primary">view</a>
                    </div>
                </div>
            {% endfor %}

        </div>

    </div>

</div>
```



```

        </div>

</div>

{% endblock %}

this page of template file is used by detailed_blog
function of views.py file of blog app to show a detailed
blog from main blog page on clicking read more button.

{% extends "blog/base.html" %}
{% block title %}blog{% endblock %}

{% block content %}
<div class="container-fluid">
    <div class="row justify-content-evenly text-center">
        <div class="col-lg-8 col-md-12">
            <h3> {{p.title | capfirst}}</h3>
            
            <p class="pt-4 "> {{p.content | safe}}</p>

        </div>
        <div class="col-lg-3 col-md-12">

            {% for cat in cats %}
            <div class="card">


                

                <div class="card-body text-center">
                    <h6 class="card-title">{{cat.name}}</h6>
                    <p class="card-text">{{cat.discription
|truncatechars:20}}</p>
                    <a href="/blog/category_blog/{{cat.no}}" class="btn btn-
primary">view</a>
                </div>
            </div>
            {% endfor %}


        </div>
    </div>
</div>
{% endblock %}

```


Resource management
Home [blog](#) [Test](#) [Notes](#) [Signin](#) [Signup](#) [services](#)



Programming Language
Read blog by profes...
[view](#)



Finance
finance involves ma...
[view](#)



Success story
Stories of success ...
[view](#)

HOW CENTRAL BANK MANAGE ECONOMY OF A COUNTRY?

Central banks play a crucial role in managing the economy of a country. Their primary responsibilities often include controlling monetary policy, issuing currency, regulating and supervising financial institutions, and maintaining financial stability. Here are some of the key ways in which central banks manage the economy:

[read more...](#)

HOW DOES BITCOIN WORK?

As a new user, you can get started with Bitcoin without understanding the technical details. Once you've installed a Bitcoin wallet on your computer or mobile phone, it will generate your first Bitcoin address and you can create more whenever you need one. You can disclose your addresses to your friends so that they can pay you or vice versa. In fact, this is pretty similar to how ...

[read more...](#)

KNOW YOUR CUSTOMER: WHAT TRIGGERS THEM TO BUY YOUR PRODUCTS OR SERVICES

As an entrepreneur, it is important to note that customers would not buy your product or service unless something causes them to buy. This is often overlooked, but trust us when we say that buying triggers are one of the most important pieces of information you can collect about your customer. Simply selling your product through different channels is not enough to attract and retain loyal customers. It requires a ...

[read more...](#)

DECODING CODE: A GUIDE TO CHOOSING THE RIGHT PROGRAMMING LANGUAGE FOR YOU"

Selecting the right programming language is a crucial decision for aspiring developers and seasoned coders alike. Each language has its strengths, weaknesses, and specific use cases. In this blog, we'll explore essential factors to consider when choosing a programming language that aligns with your goals and projects. Define Your Goals: Begin by clarifying your objectives. Are you interested in web development, data science, mobile app development, or game programming? Different ...

[read more...](#)

BLOCKCHAIN: WHY AND WHAT?

why blockchain? ब्लॉकचेन लेनदेन का एक निरंतर डिजिटल लेजर है। यह लेनदेन की पूरी अवधारणा को बदल देता है अब हमारे लेनदेन को सत्यापित करने के लिए केंद्रीकृत अधिकारियों की कोड़े आवश्यकता नहीं है, ब्लॉकचेन दूरी विकेन्द्रीकृत और सुरक्षित बनाता है। डेटा माइनिंग नामक प्रक्रिया के माध्यम से पी 2 पी उपयोगकर्ताओं द्वारा किए गए क्रिप्टोग्राफिक प्रक्रियाओं द्वारा सुरक्षा बनाए रखी जाती है, यहां नीचे कुछ महत्वपूर्ण विशेषताएं दी गईं ...

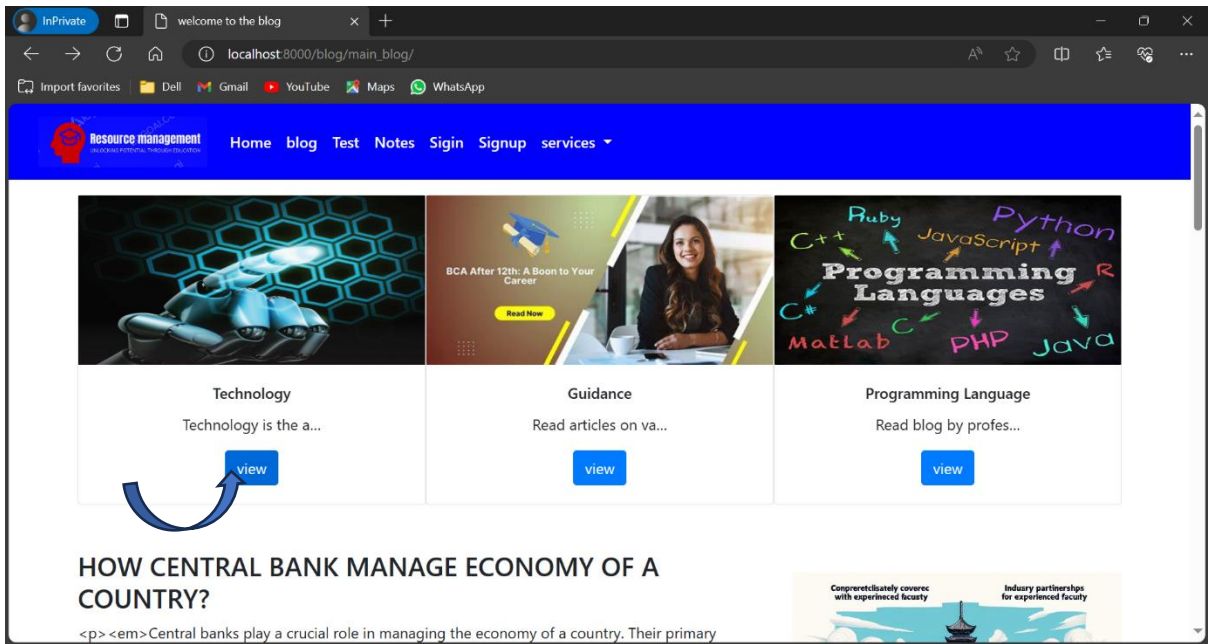
[read more...](#)

Navigate
Contact Form

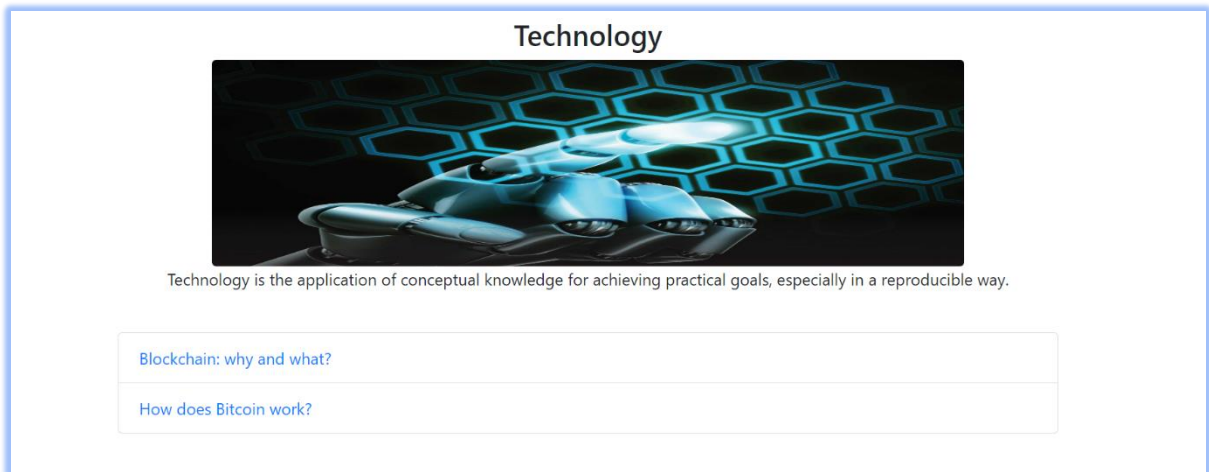
This is the main blog page view where user can see 5 blogs at a time. When user can click on read more then dynamically open detailed blog where category is visible in side bar. The upper side of main blog page has an owl carousel, from there one can find blog category wise.

6.2 categorization and searching:

for seeing category wise blog one can click on top slider view button and can see category wise blog as shown in next image.



On clicking view button of technology category one can view blog category wise as you can see in below image.



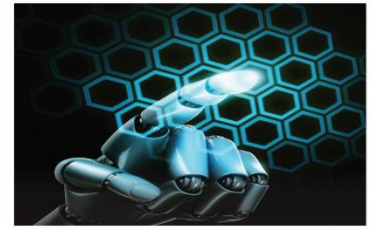
From category blog one can also go for details blog.

Decoding Code: A Guide to Choosing the Right Programming Language for You"



Selecting the right programming language is a crucial decision for aspiring developers and seasoned coders alike. Each language has its strengths, weaknesses, and specific use cases. In this blog, we'll explore essential factors to consider when choosing a programming language that aligns with your goals and projects. Define Your Goals: Begin by clarifying your objectives. Are you interested in web development, data science, mobile app development, or game programming? Different languages excel in specific domains, so understanding your goals narrows down the options. Consider Learning Curve: Evaluate your familiarity with programming. If you're a beginner, languages like Python and JavaScript are known for their readability and ease of learning. For more experienced developers, languages like C++ or Rust might be suitable for performance-critical applications. Community and Support: A vibrant community can be a lifesaver when you encounter challenges. Choose a language with an active community and robust online resources. Forums, documentation, and tutorials play a crucial role in your learning journey and problem-solving. Job Market Demand: Research the demand for specific programming languages in the job market. Languages like JavaScript, Python, and Java are often in high demand, offering diverse opportunities. However, niche languages might be valuable in specific industries or roles. Project Requirements: Tailor your language choice to the project's requirements. For web development, consider HTML, CSS, and JavaScript. Data science tasks often involve Python and R, while mobile app development leans towards languages like Swift (iOS) or Kotlin (Android). Performance Needs: Evaluate the performance requirements of your project. Low-level languages like C and C++ are ideal for system-level programming, while interpreted languages like Python may prioritize ease of development over raw performance. Ecosystem and Libraries: Explore the ecosystem and libraries associated with each language. A robust ecosystem can significantly speed up development by providing pre-built solutions and third-party libraries. For example, Node.js has a vast ecosystem for server-side JavaScript development. Longevity and Trends: Consider the longevity of the language and its current trends. While established languages like Java and C++ continue to be relevant, newer languages might offer modern features and paradigms. Be mindful of industry trends without completely disregarding stability. Conclusion: Choosing a programming language is a personal and project-specific decision. By defining your goals, considering the learning curve, evaluating community support, assessing job market demand, understanding project requirements, considering performance needs, exploring ecosystems, and staying mindful of trends, you can make an informed decision that aligns with your aspirations. Happy coding!

User can go to category page from details blog page.



Technology
Technology is the a...
[view](#)



Guidance
Read articles on va...
[view](#)



Programming Language
Read blog by profes...
[view](#)



Finance
finance involves ma...
[view](#)



Success story
Stories of success ...
[view](#)

Navigate

- [Home](#)
- [About Us](#)
- [FAQ](#)

FOLLOW US



Technical Support ☎
777-234-098-127

Contact Form

Name

Email address

Message

[Submit](#)

Notes repository:

The Notes App is an integral component of the Resource Management Project, offering users a robust platform for creating, organizing, and sharing educational notes. Developed within the Django framework, this module addresses the need for an efficient and user-friendly solution to manage course materials and support collaborative learning.

The Notes App responds to the demand for a centralized and organized repository for educational materials within the Resource Management Project. It empowers users to compile, store, and access course-related notes in a structured and intuitive manner.

Objectives:

Provide users with a platform to create, store, and organize educational notes.

Support collaborative learning through the sharing of notes among users.

Enhance the learning experience by facilitating easy access to course materials.

The Notes App follows the Model-View-Template (MVT) architecture of Django:

Models.py file of notes app

```
from django.db import models
from django.utils.html import format_html

class courses(models.Model):
    c_name=models.CharField(max_length=20,null=False)
    c_desc=models.CharField(max_length=50)
    c_pic=models.ImageField(upload_to='courses/')
    def __str__(self):
        return self.c_name

    def pic(self):
        return format_html(''.format(self.c_pic))

class notes(models.Model):
    sub=models.CharField(max_length=20)
    sub_code=models.CharField(max_length=8)
    semester=models.IntegerField()
    material=models.FileField(upload_to='material/')
    course=models.ForeignKey(courses,on_delete=models.CASCADE)
    uploaded_by=models.CharField(max_length=25)
```

viewss.py file of notes app

```
from django.shortcuts import render
from django.http import HttpResponseRedirect,HttpResponseRedirect
from .models import *
```

this function is used for rendering available courses for notes

```
def course_list(request):
    c=courses.objects.all()
    return render(request,'notes/notes.html',{'c':c})
```

this function is used for rendering notes for particular course's notes

```
def course(request,id):
    c=courses.objects.get(id=id)
    p=notes.objects.filter(course=c)
    return render(request,'notes/bca_semester.html',{'p':p})
```

this function is used for rendering form to upload notes to database

```
def note_upload(request):
    a= courses.objects.all()
    return render(request,'notes/notes_upload.html',{'a':a})
```

this function is used to save data send by note_upload function

```
def note_save(request):
    try:
        d=request.POST['course']
        p=courses.objects.get(id=d)
        demo=notes()
        demo.sub=request.POST['sub']
        demo.sub_code=request.POST['sub_code']
        demo.semester=request.POST['semester']
        demo.course=p
        demo.uploaded_by=request.POST['uploaded_by']
        demo.save()
        return HttpResponseRedirect("successfully uploaded")
    except:
        HttpResponseRedirect('some errors occurred try after some time')
```

Template file for notes

This template html file is used to show courses for notes

```
{% extends "online_exam/base.html" %}
{% block title %}notes{% endblock %}
{% block content %}
{% load static %}
<div class="container ">
<div class="row row-cols-1 row-cols-md-2 row-cols-lg-3 justify-content-
evenly">
  {% for c in c %}
<div class="col pb-4 ">
<div class="card" ">
  
  <div class="card-body text-center">
    <h5 class="card-title">{{c.c_name}}</h5>
    <p class="card-text">{{c.c_desc}}</p>
    <a href="/notes/courses/{{c.id}}" class="btn btn-primary">view</a>
  </div>
</div>
</div>
</div>
{% endfor %}
</div>
</div>
{% endblock %}
```

This template html file is used to notes uploading form

```
<!-- upload_note.html -->
{% extends "online_exam/base.html" %}
{% block title %}note-upload{% endblock %}
{% block content %}
  <h2>Upload Note</h2>

  <form method="post" enctype="multipart/form-data"
action="/notes/note_save/">
    {% csrf_token %}
    <label for="course">Course:</label>
    <select name="course" id="course">
      {% for course in a %}
        <option value="{{ course.id }}">{{ course.c_name }}</option>
      {% endfor %}
    </select><br>

    <label for="title">sub:</label>
    <input type="text" name="sub" required><br>
```



```

<label for="content">sub_code:</label><br>
<input type="text" name="sub_code"><br>

<label for="content">semester:</label><br>
<input type="number" name="semester"><br>

<label for="content">uploadedby:</label><br>
<input type="text" name="uploaded_by"><br>

<label for="file">Upload File:</label>
<input type="file" name="file" accept=".pdf, .doc, .docx"><br>

<button type="submit">Upload Note</button>
</form>
{% endblock %}
This template html file is used to show notes according to
courses

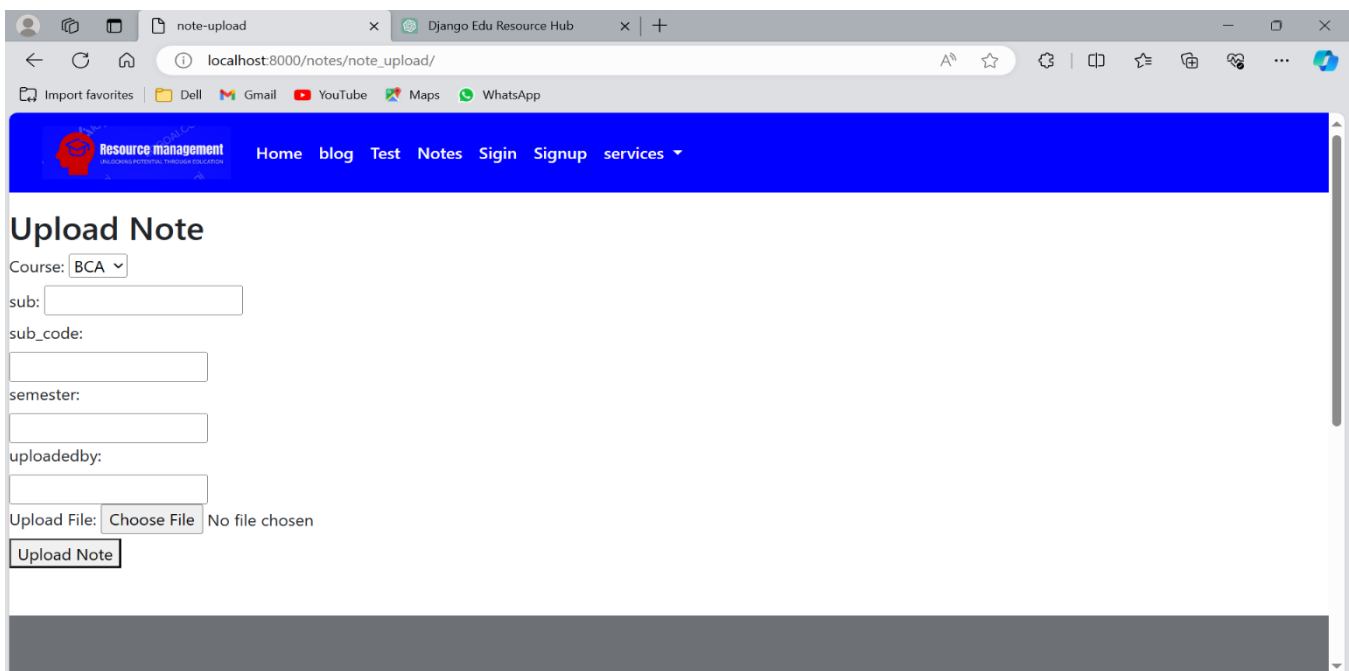
```

```

{% extends "online_exam/base.html" %}
{% block title %}notes{% endblock %}
{% block content %}
<div class="container">
  {% for p in p %}
  <div class="row">
    <div class="col-3">sem-{{p.semester}}th sub-code({{p.sub_code}})</div>
    <div class="col-3">{{p.sub}} <br> upload by- {{p.uploaded_by}}</div>
    <div class="col-6"><embed src="/media/{{p.material}}" type="pdf.pdf"></div>
  </div>
  {% endfor %}
</div>
{% endblock %}

```

The interface for notes uploading





BCA
Bachelor of Computer Applications (BCA)

[view](#)



MCA
Masters of Computer Applications (MCA)

[view](#)



BBA
Bachelor of Business Applications (BBA)

[view](#)



MBA
Masters of Business Applications (MBA)

[view](#)



B.Ed
Bachelor of Education (B.Ed)

[view](#)

Navigate

[Home](#)

[About Us](#)

[F.A.Q](#)

FOLLOW US



Technical Support

777-234-098-127

Contact Form

Name

Email address

Message

[Submit](#)

the upper image shows the courses for notes while below image show the notes of a particular section

sem-4th sub-code-(BC-404)

software engineering
upload by- Abhishek kumar

sem-5th sub-code-(BC-502)


python
upload by- Abhishek kumar

sem-4th sub-code-(BC-402)


java
upload by- raj Kapoor

sem-4th sub-code-(BC-402)


java
upload by- raj Kapoor




This plug-in isn't supported



This plug-in isn't supported



This plug-in isn't supported

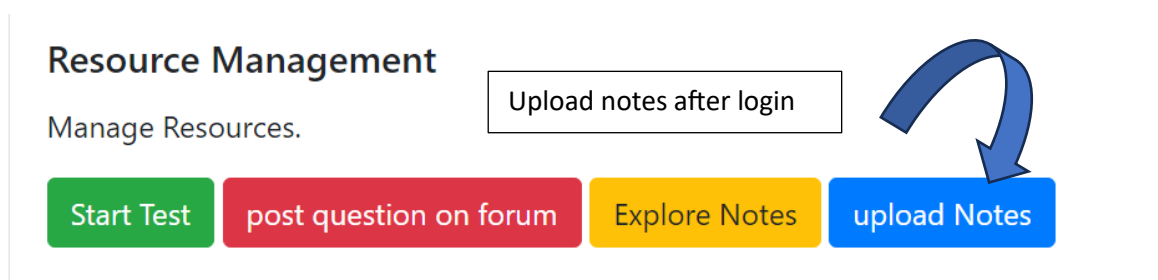


This plug-in isn't supported

7.2 Organization and Accessibility

When user enters as students then on home interface there is a button by clicking on that he can upload notes. in resource management we organize notes according to courses as you can see models where we build a foreign key name 'course' to organise notes according to course.

Any user can access the notes course wise without any login from main landing page.



7.3 Integration with Other Modules:

Notes module is integrated with online_exam. When user enters as student then there are two buttons from there, they can explore notes as well as upload notes.

The Notes App in the Resource Management Project reflects the project's dedication to providing comprehensive and user-centric solutions for the educational community. although it can be improved with feature like adding editing and highlighting features.

Discussion forum

The Discussion Forum module within the Resource Management Website serves as a collaborative space for students to post questions and receive answers from peers.

Discussion creation: - any user by login can create a question on forum and also answer on any discussion. Although any user can explore the forum from main landing page by clicking visit forum. This module is also follow MVT architecture.

Models.py file of discussion forum

```
class DiscussionPost(models.Model):
    author = models.CharField(max_length=100)
    content = models.TextField()
    is_question = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.author} - {self.created_at}"
```

view.py file of discussion forum

this function of views.py is used to display forum

```
def discussion_forum(request):
    questions = DiscussionPost.objects.filter(is_question=True)
    answers = DiscussionPost.objects.filter(is_question=False)
    return render(request, 'discussion_forum/discussion_forum.html',
{'questions': questions, 'answers': answers})
```

this function of views.py is used to save Q and Ans to database.

```
def create_post(request):
    if request.method == 'POST':
        author = request.POST.get('author')
        content = request.POST.get('content')
        is_question = request.POST.get('is_question') == 'on'
        DiscussionPost.objects.create(author=author, content=content,
is_question=is_question)
        return redirect('discussion_forum')
    return render(request, 'discussion_forum/create_post.html')
```

Welcome to Resource Management

Your hub for online tests, blogging, notes, and discussions.

Explore the various features and enhance your learning experience.

[Learn more](#)

Online Tests

Take online tests and track your performance with detailed analytics.

[Explore](#)

Blog

Share your insights and engage in discussions with the community.

[Explore](#)

Notes

Organize and access course materials easily with our notes repository.

[Explore](#)

Discussion Forum

Join discussions, ask questions, and collaborate with fellow learners in our discussion forum.

[Visit Forum](#)

Navigate

[Home](#)

[About Us](#)

[F.A.Q](#)

FOLLOW US



Technical Support

777-234-098-127

Contact Form

Name

Email address

Message

[Submit](#)

from main landing page any one can visit forum and from there they can post question as well as answer for forum.

Question Management

Manage questions for online tests.

[Create Question](#)

[View Question](#)

[Manage discussion forum](#)



Django Edu Resource Hub | Discussion Forum

localhost:8000/forum/discussion/

Discussion Forum

Questions


abhishek kumar - Dec. 19, 2023, 6:07 p.m.
how to write project report for final semester?

sunny - Dec. 19, 2023, 6:13 p.m.
how to gets good marks?

Answers

abhishek kumar - Dec. 19, 2023, 6:14 p.m.
you can get good marks in exam by throughly revise the content teach by teachers.

[Post a Question or Answer](#)



Django Edu Resource Hub | Create Post

localhost:8000/forum/create_post/

Create a New Post

Author:


Content:

Is this a question?

[Back to Discussion Forum](#)

Resource Management

Manage Resources.



Student interface after login

Implementation

9.1 Development Environment Setup: Setting up a well-organized development environment is crucial for the successful implementation of a project using Django. we Follow these steps to ensure a smooth development process:

Virtual Environment: use the command `python -m venv venv`

Activate Virtual Environment: for activating type `venv\Scripts\activate`

Install Dependencies: `pip install -r requirements.txt`

After model creation first makemigrations and then migrate

Database Setup: `python manage.py makemigrations`

`python manage.py migrate`

create super user- `python manage.py createsuperuser`

Static Files:

Configure static files in settings.py file

Configure static files in settings.

Type command -`python manage.py collectstatic`

Project creation-`django-admin createproject projectname`

App creation-`python manage.py startapp app_name`

Run server-use command `python manage.py runserver`

Access Admin Panel:

Visit `http://localhost:8000/admin/` **and log in with the superuser credentials.**

9.2 Django Best Practices

Project Structure:

Organize the project into apps based on functionality (e.g., users, resources).

Follow Django's recommended project structure.

Code Readability:

Adhere to PEP 8 style guidelines for consistent and readable code.

Use meaningful names for variables, functions, and classes.

Models and Migrations:

Design clear and concise models.

Templates:

Utilize Django template tags for dynamic content.

Keep templates modular and organized.

Views and URL Patterns:

Keep views simple and focused on specific functionality.

Use well-structured URL patterns.

Forms:

Leverage Django forms for input validation and handling user data.

Organize form classes effectively.

Middleware:

Utilize middleware for cross-cutting concerns like authentication and security.

Keep middleware logic clean and focused.

Security:

Implement secure coding practices to prevent common web vulnerabilities.

Use Django's built-in security features, such as protection against SQL injection and cross-site scripting.

Testing:

Write comprehensive unit tests and functional tests.

Use Django's testing framework for automated testing.

9.3 Testing Strategies: testing is very important so that a django app or project could perform in intended way.

We generally during development of Resource management website use some common type of testing such as:

Unit testing: tests for individual components, such as models, views, and forms.

integration Testing: Testing the integration of different components within the application.

Ensure seamless collaboration between app functionalities.

Regression Testing:

Implement regression tests to catch unintended side effects during code changes.

Run automated tests regularly.

You can also go for others testing methods like user acceptance testing, load testing etc.

Conclusion

10.1 Achievements:

Module Integration: Successfully integrated and implemented various modules, including the Online Exam Module, Blog System, and Notes App, each contributing to different aspects of the educational journey.

User Engagement: Established a vibrant community for vocational students to participate in discussions, share insights through blogs, and collaboratively manage educational notes.

Scalability and Maintainability: Utilized the Django framework and a modular design approach, ensuring the scalability and maintainability of the system as it evolves.

Security Measures: Implemented robust security measures to safeguard user data and protect against potential vulnerabilities, ensuring a secure online learning environment.

User-Friendly Interfaces: Developed intuitive and responsive user interfaces across modules, ensuring a seamless and enjoyable user experience.

10.2 challenges

Security Implementation: we are facing hurdles in implementing comprehensive security measures to protect user data and prevent unauthorized access. Although we are working on it to overcome.

Collaborative Features: Balancing the implementation of collaborative features in modules like the Blog System and Notes App, ensuring they enhance rather than disrupt the learning experience.

10.2 Future Enhancements

While the initial scope is focused on essential features, the project allows for future enhancements, such as the integration of additional functionalities like a chat system, online attendance through face recognition, separate features for registering and selling product, fund rising, extending its features towards humanities and others streams students.

Adding multimedia in blog and notes section of website to enhance students' productivity.

Acknowledgement

We extend our heartfelt gratitude to the individuals whose dedication and expertise were instrumental in the successful development and implementation of the Resource Management Web app. Their contributions have been invaluable in shaping this innovative platform for online education.

Abhishek kumar
Roll-214153
Reg-19RGVBCA037

Backend of Resource management
Logo of project, project report writing

Sunny kumar
Roll-2140734
Reg-19RGVBCA036

frontend of Resource management
co-ordination between members
bootstrap work of project

Navneet kumar
Roll-214164
Reg-20CCVGCS047

frontend of Resource management
CSS and java script
design of project

Ankesh kumar
Roll-214133
Reg-20CCVGCS016

Testing for Resource management
html
co-writer of project report

References

Django documentation- <https://docs.djangoproject.com/>
W3schools- <https://www.w3schools.com/>
Bootstrap5.3 documentation-<https://www.bootstrap.com/>
Django tutorial on w3c YouTube channel