Company Profile

Company and Mentor Details

Project Profile

Title: Blood Test System

Abstract: Our project, a comprehensive blood test management system developed using Django, offers a seamless experience for patients seeking blood tests. The system facilitates the booking and management of appointments, allowing patients to schedule appointments conveniently. Additionally, patients can provide feedback on their experience, contributing to continuous improvement. Moreover, the system provides a secure platform for patients to access and view their test reports online, ensuring efficient communication of critical health information. With user-friendly interfaces and robust functionalities, our project aims to enhance the efficiency and effectiveness of blood test management while prioritizing patient satisfaction and convenience.

Key-Words: Patient, Appointments, Laboratory, Feedbacks

Modules: Laboratory Admin, Patient, Home, Lab Worker

Technology:

Front-End - HTML, CSS, JavaScript, Bootstrap, ReactJS

Back End - Django web framework (Python)

Database – SQLite

Day 1

“Unraveling the Strengths of Python and Django: A Detailed Exploration”

Prompt: Craft a summary outlining Python and the Django framework, highlighting Python's capabilities while also presenting an accompanying diagram for clarity and brevity.

Python, renowned for its simplicity and versatility, serves as a powerful programming language for a myriad of applications. With its clear syntax and extensive standard library, Python facilitates rapid development and readability, making it a favorite among developers worldwide. Its dynamic typing and strong support for object-oriented, functional, and procedural programming paradigms enable developers to tackle diverse tasks efficiently.

When coupled with the Django framework, Python becomes even more formidable for web development. Django, a high-level web framework, streamlines the creation of robust, scalable web applications. Built on the principles of DRY (Don't Repeat Yourself) and convention over configuration, Django promotes clean, maintainable code and accelerates development through its built-in features and utilities.

Django's architecture follows the Model-View-Controller (MVC) pattern, with slight variations, such as the Model-View-Template (MVT) pattern. Models define the data structure, views handle user requests and generate responses, and templates render HTML pages. Additionally, Django includes an Object-Relational Mapping (ORM) layer, facilitating database interactions without requiring direct SQL queries.

The framework provides built-in security features, such as protection against common web vulnerabilities like SQL injection and Cross-Site Scripting (XSS). Furthermore, Django offers authentication, session management, and

authorization mechanisms out of the box, ensuring robust security for web applications.

In summary, Python's versatility and simplicity, coupled with Django's powerful features and conventions, make them a formidable combination for web development, enabling developers to create sophisticated and scalable web applications efficiently.

[Python Functionality]

- Clear Syntax
- Extensive Standard Library
- Dynamic Typing
- Support for Multiple Programming Paradigms
  - Object-Oriented
  - Functional
  - Procedural

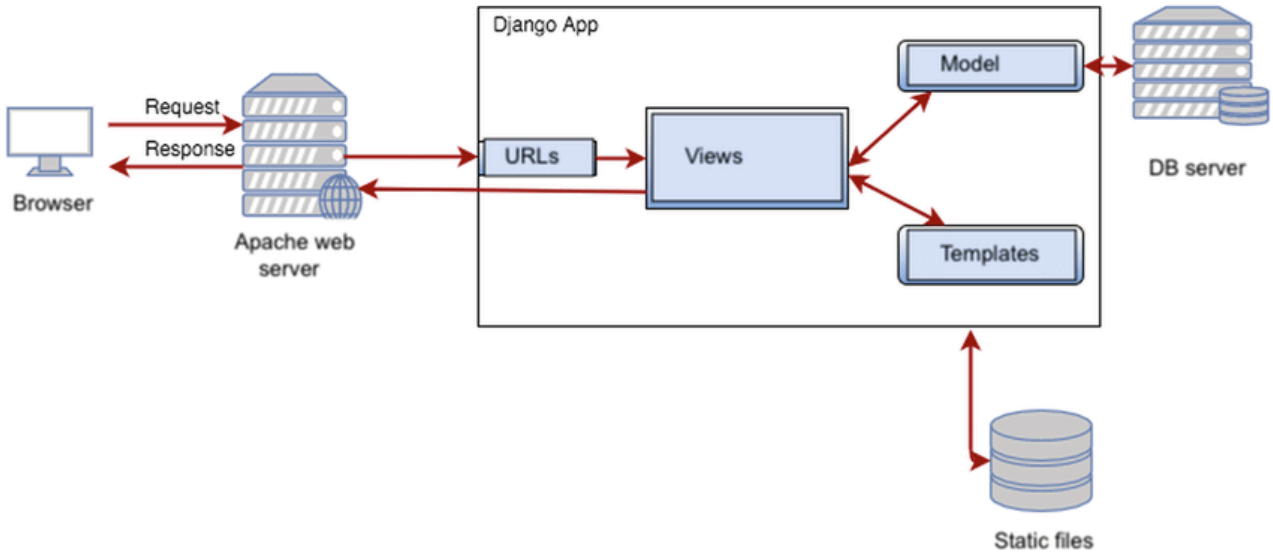
[Django Framework]

- High-Level Web Framework
- Follows MVC/MVT Architecture
- Built-in Features:
  - ORM Layer
  - Security Mechanisms
  - Authentication & Authorization

- Promotes Clean, Maintainable Code
- Accelerates Development

[Web Applications]

- Robust
- Scalable
- Secure
- Efficiently Developed



Prompt: Compare **python with other programming languages**

Language	Characteristics	Pros	Cons	Typical Uses
C	Compiled language. Simple and very widely used. A building block of many other languages.	Highly efficient programming language. Stable. Fast.	Low-level language, which means it can be difficult to use due to technical details a programmer must remember. It can lead to developers creating problems as they work on it. Modern coding techniques are challenging to implement within the language.	Used in many embedded systems.

Day 2

“Building a web application with Django”

Prompt: Django project development, featuring detailed code and visual aids.

1. **Setup Environment:**
   - Install Django: `pip install django`.
   - Create a new Django project: `django-admin startproject myproject`.
   - Create a new app: `python manage.py startapp myapp`.

2. **Define Models:**

```python
# myapp/models.py
from django.db import models

class MyModel(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.name
```

3. **Migrate Database:**

```
```

```
python manage.py makemigrations
python manage.py migrate
```

4. **Create Views:**

```python
# myapp/views.py
from django.shortcuts import render
from .models import MyModel

def my_view(request):
    objects = MyModel.objects.all()
    return render(request, 'myapp/my_template.html', {'objects': objects})
```

5. **Create Templates:**

   - Create `my_template.html` in `myapp/templates/myapp/` directory.

6. **Define URLs:**

```python
# myapp/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('', views.my_view, name='my_view'),
]
```

```

7. **Configure Project URLs:**

```python
# myproject/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
]
```

8. **Run Development Server:**

```
python manage.py runserver
```

9. **Access your project:** Visit `**http://127.0.0.1:8000/**` in your browser.

This is a basic setup to get you started. From here, you can expand your project by adding more models, views, templates, and functionalities as needed.
```

Day 3

“Login and Register”

Prompt: Creating Login and Register pages for website

Step 1: Go to ‘views.py’ location and write code for login and signup.

```
def Lab_signin(request):
    if request.method=="POST":
        print(request.POST['Lid'])
        try:
            m = LaboratoryRegister.objects.get(Lid=request.POST['Lid'])
            if m.Laboratorypwd == request.POST['Laboratorypwd']:
                request.session['Lid'] = m.Lid
                request.session['name'] = m.Laboratoryfname
                request.session['fname'] = m.Laboratorymname
                request.session['mname'] = m.Laboratorylname

                return redirect('laboratory:index')
            else:
                return render(request,'Lab_signin.html',{'m':'Password incorrect'})
        except:
            return render(request,'Lab_signin.html',{'m':'Password incorrect'})
    return render(request,'Lab_signin.html')




def signin(request):
    if request.POST:
        email = request.POST['uid']
        pass1 = request.POST['userpwd']
        try:
            valid = UserRegister.objects.get(uid=email,userpwd=pass1)
            if valid:
                request.session['user'] = email
                request.session['userId'] = valid.pk
```

```
            return redirect('user:index')
        else:
            return render(request,'signin1.html',{'m':'Password incorrect'})
    except:
        return render(request,'signin1.html',{'m':'Password incorrect'})
        #  return redirect('/signin/')
return render(request,'signin1.html')
```

```
# Registration

def signup(request):
    obj=UserRegisterForm(request.POST)
    if obj.is_valid():
        data=UserRegister.objects.all().filter(uid=request.POST['uid'])
        if len(data)<=0:
            obj.save()
            return redirect('user:signin')
        else:
            return render(request,'signup.html',{'messagekey':"User Already Exists"})
    return render(request,'signup.html')


def Lab_signup(request):
    obj=LaboratoryRegisterform(request.POST,request.FILES)
    if obj.is_valid():
        data=LaboratoryRegister.objects.all().filter(Lid=request.POST['Lid'])
        print(len(data))
        if len(data)<=0:
            obj.save()
            return redirect('laboratory:lab_signin')
        else:
            return render(request,'Lab_signup.html',{'messagekey':"User Already Exists"})
    return render(request,'Lab_signup.html')
```

Step 2: Create HTML templates to render the views.

```
<!— login.html -->

{% load static%}
```

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <title>Ĺogin</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="{% static 'signin1.css' %}">
    <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script
>
    <style>
      .card{
        background-color: transparent;
        color: #017f92;
        }
      a{
        color: #017f92;
      }
      a:hover{
        color: #017f92;
      }
    </style>
  </head>

  <body>
    <div>
<img src="{% static 'img/bg20.jpg' %}" style="height: 100vh; width:100%; position:absolute; z-index: -1; opacity: 1;">
    </div>
    <div style="position:relative; top:100px;">
      <center>
        <div class="col-md-4">
          <div class="card bg-mute">
            <form method="POST" autocomplete="off">
              {% csrf_token %}
                <h1 class="h1 text-center mt-5 mb-5">Laboratory Login</h1>
                <div class="m-1">
```

```
                <input type="email" class="form-control" id="email" placeholder="Enter email"
name="Lid" value="{{ Lid }}">
            </div>
            <div class="m-1">
                    <input type="password" class="form-control" placeholder="Enter password"
name="Laboratorypwd" value="{{ Laboratorypwd }}">
            </div>
            <div class="mt-1 mb-1">
                        <a href="/med/show/"><button type="submit" class="btn btn-info">Sign
Up</button></a>
            </div>
            <hr/>
            <div class="foot-lnk text-center">
                <a href="{% url 'laboratory:lab_signup' %}">Create an account</a>
            </div>
      </form>
</div>
</div>
</center>
      </div>
      {% if m %}
          <script>alert("{{m}}")</script>
         {% endif %}
      </body>
</html>



{% load static %}


<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <title>Ĺogin</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
                                                        <link          rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="{% static 'signin1.css' %}">
    <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
                                                                                <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
```

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script
>
   <style>
     .card{
       background-color: transparent;
       color: #17a2b8;
       }
     a{
       color: #17a2b8;
     }
     a:hover{
       color: #17a2b8;
     }
   </style>
  </head>

  <body>
   <div>
                <img src="{% static 'img/login.jpg' %}" style="height: 100vh; width:100%;
position:absolute; opacity: 0.5;">
   </div>
   <div style="position:relative; top:100px;">
     <center>
       <div class="col-md-4">
         <div class="card">
           <form method="POST" >
             {% csrf_token %}
               <h1 class="h1 text-center mt-5 mb-5">User Login</h1>

               <div class="m-1">
                       <input type="text" class="form-control" id="email" placeholder="Enter email"
name="uid" value="{{ uid }}">
               </div>
               <div class="m-1">
                        <input type="password" class="form-control" placeholder="Enter password"
name="userpwd" value="{{ userpwd }}">
               </div>
               <div class="mt-4 mb-3">
                 <div class="">
                   <button type="submit" class="btn btn-info">Sign In</button>
                 </div>
               </div>
               <hr/>
```

```
            <div class="foot-lnk text-center">
              <a class="#Create" href="{% url 'user:signup' %}" >Create an account</a>
            </div>


          </form>
        </div>
      </div>
    </center>
  </div>
          {% if m %}
            <script>alert("{{m}}")</script>
          {% endif %}
 </body>
</html>
```

```
<!—register.html -->

{% load static %}



<!DOCTYPE html>
<html lang="en" dir="ltr">
      <head>
            <title>Registration</title>
            <meta charset="utf-8">
            <meta name="viewport" content="width=device-width, initial-scale=1">
                                                                                                <link       rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
            <link rel="stylesheet" href="{% static 'signin1.css' %}">
                                                                                                                                <script
src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
                                                                                                                                <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
                                                                                                                                <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script
>
            <style>
                  .card{
```

```
                background-color: transparent;
                color: #17a2b8;
                }
                a{
                color: #17a2b8;
                }
                a:hover{
                color: #17a2b8;
                }
            </style>
        </head>

        <body>
            <div>
                        <img src="{% static 'img/login.jpg' %}" style="height: 100vh; width:100%;
position:absolute; opacity: 0.5;">
            </div>
            <div style="position:relative;">
                <center>
                    <div class="col-md-4">
                        <div class="card">
                            <form method="POST" >
                            {% csrf_token %}

                                <h1 class="h1 text-center mt-2 mb-2">SIGN UP</h1>

                                <div class="m-1">
                                        <input type="email" class="form-control" id="email"
placeholder="Enter email"name="uid" value="{{ uid }}" >
                                </div>
                                <div class="m-1">
                                            <input type="password" class="form-control"
placeholder="Enter password" name="userpwd" value="{{ userpwd }}">
                                </div>
                                <div class="m-1">
                                         <input type="text" class="form-control" id="fname"
placeholder="First Name" name="userfname" value="{{ userfname }}">
                                </div>
                                <div class="m-1">
                                        <input type="text" class="form-control" id="mname"
placeholder="Middle Name" name="usermname" value="{{ usermname }}">
                                </div>
                                <div class="m-1">
```

```
                                        <input type="text" class="form-control" id="lname"
placeholder="Last Name" name="userlname" value="{{ userlname }}">
                                </div>
                                <div class="m-1">
                                                <input type="number" class="form-control"
id="user_age" placeholder="Age" name="user_age" value="{{ user_age }}">
                                </div>
                                <label class="m-1" for="gender"> Select your gender</label>
                                <select name="usergender" class="m-1">
                                    <option value="none" selected>Gender</option>
                                    <option value="Male">Male</option>
                                    <option value="Female">Female</option>

                                </select>
                                <div class="m-1">
                                        <textarea row="3" class="form-control" id="Address"
placeholder="Address"  name="useraddress" value="{{ useraddress }}"></textarea>
                                </div>
                                <div class="m-1">
                                            <input type="text" class="form-control" id="city"
placeholder="City" name="usercity" value="{{ usercity }}">
                                </div>
                                <div class="m-1">
                                           <input type="text" class="form-control" id="area"
placeholder="Area"   name="userarea" value="{{ userarea }}">
                                </div>
                                <div class="m-1">
                                        <input type="text" class="form-control" id="Pincode"
placeholder="Pincode"   name="userpincode" value="{{ userpincode }}">
                                </div>
                                <div class="m-1">
                                                <input type="number" class="form-control"
id="content_no" placeholder="Contact No"   name="usercontactno" value="{{ usercontactno
}}">
                                </div>
                                <div class="mt-1 mb-1">
                                            <button type="submit" class="btn btn-info">Sign
Up</button>
                                </div>
                                <hr/>
                                <div class="foot-lnk text-center">
                                    <a href="{% url 'user:signin' %}">Already Member?</a>
                                </div>
```

```
                                    </form>
                                </div>
                            </div>
                        </center>
                                                {% if messagekey %}
                                                <script>alert("{{ messagekey }}")</script>
                                                {% endif %}
                    </div>
            </body>
</html>


{% load static%}

<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <title>Login</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
                                                                <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="{% static 'signin1.css' %}">
    <script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.slim.min.js"></script>
                                                                                <script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"></script>
                                                                                <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.1/dist/js/bootstrap.bundle.min.js"></script
>
    <style>
      .card{
        background-color: transparent;
        color: #017f92;
        }
      a{
        color: #017f92;
      }
      a:hover{
```

```
      color: #017f92;
    }
  </style>
 </head>

 <body>
  <div>
     <img src="{% static 'img/lab.jpg' %}" style="height: 100vh; width:100%; position:absolute;
z-index: -1; opacity: 0.5;">
  </div>
  <div style="position:relative;">
    <center>
     <div class="col-md-4">
        <div class="card">
        <form method="POST" autocomplete="off" enctype="multipart/form-data">
         {% csrf_token %}
          <h1 class="h1 text-center mt-2 mb-2">Laboratory Signup</h1>
          <div class="">
                <input type="email" class="form-control" id="email" placeholder="Enter email"
name="Lid" value="{{ Lid }}">
          </div>
          <div class="">
                    <input type="password" class="form-control" placeholder="Enter password"
name="Laboratorypwd" value="{{ Laboratorypwd }}">
          </div>
          <div class="">
                   <input type="text" class="form-control" id="fname" placeholder="Laboratory
Name"  name="Laboratoryfname" value="{{ Laboratoryfname }}">
          </div>
          <div class="">
                <input type="text" class="form-control" id="mname" placeholder="First Name"
name="Laboratorymname" value="{{ Laboratorymname }}">
          </div>
          <div class="">
                  <input type="text" class="form-control" id="lname" placeholder="Last Name"
name="Laboratorylname" value="{{ Laboratorylname }}">
          </div>
          <div class="">
                    <textarea row="2" class="form-control" id="Address" placeholder="Address"
name="Laboratoryaddress" value="{{ Laboratoryaddress }}"></textarea>
          </div>
          <div class="">
                              <input type="text" class="form-control" id="city" placeholder="City"
name="Laboratorycity" value="{{ Laboratorycity }}">
```

```
            </div>
            <div class="">
                        <input type="text" class="form-control" id="area" placeholder="Area"
name="Laboratoryarea" value="{{ Laboratoryarea }}">
            </div>
            <div class="">
                  <input type="text" class="form-control" id="Pincode" placeholder="Pincode"
name="Laboratorypincode" value="{{ Laboratorypincode }}">
            </div>
            <div class="">
                              <input type="number" class="form-control" id="content_no"
placeholder="Contact No" name="Laboratorycontactno" value="{{ Laboratorycontactno }}">
            </div>
            <div class="">
                        <input id="id_chemistphoto" type="file" name="Laboratoryphoto"
class="form-control" value="{{ Laboratoryphoto }}">
            </div>
            <div class="mt-2">
                  <a href="/med/show/"><button type="submit" class="btn btn-info">Sign
Up</button></a>

            </div>
            {% if messagekey %}
            <script>alert("{{ messagekey }}")</script>
            {% endif %}
            <hr/>
            <div class="foot-lnk text-center">
                  <a href="{% url 'laboratory:lab_signin' %}">Already Member?</a>
            </div>

      </form>


</div>
</div>
</center>

      </div>

      </body>
</html>
```

Step 3: Go to ‘forms.py’ location and create form for registrations.

```
from django import forms
from laboratoryapp.models import LaboratoryRegister,Appointment,Test_category


class LaboratoryRegisterform(forms.ModelForm):
   class Meta:
      model=LaboratoryRegister
      fields='__all__'




class UserRegisterForm(forms.ModelForm):
   class Meta:
      model=UserRegister
      fields='__all__'
```

Step 4: Go to ‘models.py’ and create register model.

```
from django.db import models


# Create your models here.


# User Details

class UserRegister(models.Model):

   uid=models.EmailField(max_length=50,verbose_name='Email')

   userpwd=models.CharField(max_length=20,verbose_name='Password')

   userfname=models.CharField(max_length=20,default='',verbose_name='First_Name')
```

```
usermname=models.CharField(max_length=20,default='',verbose_name='Middle_Name')
    userlname=models.CharField(max_length=20,default='',verbose_name='Last_Name')
    n='slect gender'
    g = [
        (None, 'select gender'),
        ('Male', 'Male'),
        ('Female', 'Female'),
    ]
    usergender=models.CharField(max_length=12,choices=g,default=None)
    user_age=models.IntegerField(max_length=3,default=None)
    useraddress=models.CharField(max_length=20,default='',verbose_name='Address')
    usercity=models.CharField(max_length=30,default='',verbose_name='City')
    userarea=models.CharField(max_length=20,default='',verbose_name='Area')
    userpincode=models.IntegerField(default='',verbose_name='Pincode')
    usercontactno=models.IntegerField(default='',verbose_name='Contact_No')
    def __str__(self):
        return self.uid



class LaboratoryRegister(models.Model):
    Lid=models.EmailField(max_length=50,verbose_name='Email')
    Laboratorypwd=models.CharField(max_length=20,verbose_name='Password')

Laboratoryfname=models.CharField(max_length=20,default='',verbose_name='Laboratory_
Name')
```

```
Laboratorymname=models.CharField(max_length=20,default='',verbose_name='First_Name')

Laboratorylname=models.CharField(max_length=20,default='',verbose_name='Last_Name')

Laboratoryaddress=models.CharField(max_length=20,default='',verbose_name='Address')
   Laboratorycity=models.CharField(max_length=20,default='',verbose_name='City')
   Laboratoryarea=models.CharField(max_length=20,default='',verbose_name='Area')
   Laboratorypincode=models.IntegerField(default='',verbose_name='Pincode')
   Laboratorycontactno=models.IntegerField(default='',verbose_name='Contact_No')
    Laboratoryphoto = models.FileField(upload_to = 'upload',verbose_name='Laboratory certificate')

   def __str__(self):
      return self.Lid
```

Day 4

“Dashboard”

Prompt: Creating Dashboard for BTS

Step 1: Go to ‘views.py’ location and write code for dashboard.

```
def lab_home(request):
    if 'Lid' in request.session:
        a=LaboratoryRegister.objects.get(Lid=request.session['Lid'])
        b=Test_category.objects.all()
        return render(request,'index.html',{'a':a,'b':b,'c':len(b)})
    else:
        return redirect('laboratory:lab_signin')


def index(request):
    if 'user' in request.session.keys():
        b=UserRegister.objects.get(id=request.session['userId'])
        x=Test_category.objects.filter(email=request.session['user'])
        return render(request,'index1.html',{'b':b,'x':x})
    return redirect('user:signin')
```

Step 2: Create ‘home.html' and ‘profile.html’ and ‘c.html’.

```
<!—index.html -->


{% extends 'admin_base.html' %}
{% load static %}
{% block content %}
```

```
<h1 class="text-center mb-2" style="font-family: 'Adamina';font-size: 40px; color:rgba(255, 165, 0, 1); ">
   Welcome {{ a.Laboratorymname}}
</h1>
<div class="dashboard">

      <div class="row">
         <div class="col-md-6">
            <div class="card">
               <div class="row m-5">
                  <div class="col-md-8">
                              {% comment %} <img src="{% static 'icon/test.png' %}" class="fafa" height="50px" width="50px"><br> {% endcomment %}
                     <span class="text-muted">Total Patient Test Report</span>
                  </div>
                  <div class="col-md-4">
                     <span class="ts" style="font-weight: 900; font-size: 25px;">{{ c }}</span>
                  </div>
               </div>
            </div>
         </div>

      </div>
   <div class="card" style="margin-top:100px; padding:0px 10px 0px 10px ;">
      <div class="baner" style="margin: auto; margin-top:-4%; margin-bottom:20px;">
               <h3 class="text-center" style="font-weight: bold; padding: 10px 0px 10px 0px;">
Patient Records Report</h3>
      </div>
      <table class="table table-hover table-striped" id="myTable">
         <thead>
            <tr>
               <th>No</th>
               <th>Patient NAME</th>
               <th>Patient Email</th>
               <th>Test Name</th>
               <th>Haemoglobin</th>
               <th>Platelet_Count</th>
               <th>Date</th>
               <th>Action</th>
            </tr>
         </thead>
         <tbody>
            {% for i in b %}
```

```
            <tr>
                <td>{{forloop.counter}}</td>
                <td>{{ i.patientname }}</td>
                <td>{{ i.email }}</td>
                <td>{{ i.Test_name }}</td>
                <td>{{ i.Haemoglobin }}</td>
                <td>{{ i.Platelet_Count }}</td>
                <td>{{ i.date_of_test }}</td>
                <td>
                    <a href="{% url 'laboratory:view_test' i.id %}">view</a>
                                <a href="{% url 'laboratory:delete_test' i.id %}"><img src="{% static
'icon/delete.png' %}" height="30px"
                            width="30px"></a>
                </td>
            </tr>
            {% endfor %}
          </tbody>
      </table>
   </div>
</div>

{% endblock %}





<!—index1.html -->

{% extends 'user_base.html' %}
{% load static %}
{% block content %}



<h1 class="text-center mb-2" style="font-family: 'Adamina';font-size: 40px; color:rgba(255,
165, 0, 1); ">
   Welcome {{ b.userfname}}
</h1>
<div class="dashboard">


   <div class="card" style="margin-top:100px; padding:0px 10px 0px 10px ;">
```

```
      <div class="baner" style="margin: auto; margin-top:-4%; margin-bottom:20px;">
              <h3 class="text-center" style="font-weight: bold; padding: 10px 0px 10px 0px;">
Patient Records Report</h3>
      </div>
      <table class="table table-hover table-striped" id="myTable">
        <thead>
          <tr>
            <th>No</th>
            <th>Patient NAME</th>
            <th>Patient Email</th>
            <th>Test Name</th>
            <th>Haemoglobin</th>
            <th>Platelet_Count</th>
            <th>Date</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          {% for i in x %}
          <tr>
            <td>{{forloop.counter}}</td>
            <td>{{ i.patientname }}</td>
            <td>{{ i.email }}</td>
            <td>{{ i.Test_name }}</td>
            <td>{{ i.Haemoglobin }}</td>
            <td>{{ i.Platelet_Count }}</td>
            <td>{{ i.date_of_test }}</td>
            <td>
              <a href="{% url 'user:view_test' i.id %}">view</a>
              {% comment %} <a href="{% url 'user:pdf' i.id %}">Print</a> {% endcomment %}

            </td>
          </tr>
          {% endfor %}
        </tbody>
      </table>
    </div>
</div>

{% endblock %}
```

Step 3: ‘urls.py’ is a Python module where you define the URL patterns for your web application. It acts as a router that maps URLs to views. When a user makes a request to your Django application, Django uses the urls.py file to determine which view should handle the request.

```
from django.urls import path
from laboratoryapp.views import *

urlpatterns = [
   # signin,signup,indexpage
   path('',Lab_signin,name="lab_signin"),
   path('signup/',Lab_signup,name="lab_signup"),
   path('lab/',lab_home,name="index"),
   #profile
   path('lab_profile/',lab_profile,name="lab_profile"),
   #Logout
   path('logout/',logout,name='logout'),
   #add category
   path('add_category/',add_category,name='add_category'),
   #view category
   path('view_category/',view_category,name='view_category'),
   #edit category
   path('edit_category/<int:id>/',edit_category,name='edit_category'),
   path('delete_category/<int:id>/',delete_category,name='del_category'),
   #appointment show
   path('show_appointment/',show_appo,name='show_appointment'),
   # reject appointment
   # path('delete_test/<int:id>',delete_appointment,name='delete_appo'),
   #book appointment
   path('edit/<int:id>/',edit_appo,name="edit_appo"),
   # show approved appointment
   #book appointment
   path('show_test',view_approvedappo,name="view_approvedappo"),
   #take test
   path('take_test/<int:id>/',take_test,name='take_test'),
   #view test
   path('view_test/<int:id>/',view_test,name='view_test'),
   path('delete_test/<int:id>/',delete_test,name='delete_test'),
   path('read_feedback/',view_feedback,name='view_feedback'),
   path('read_feedback/<int:id>/',show_feedback,name='show_feedback'),

]
```

```
from django.urls import path
from  patientapp.views import *



urlpatterns = [


   #User Signin,Signup,Logout,IndexpagenIndexpage1
   path('',signin,name="signin"),
   path('index/',index,name="index"),
   path('signup/',signup,name="signup"),
   path('logout/',logout,name='logout'),
   path('book_appointment/',book_appointment,name="book_appointment"),
   path('view_test/<int:id>',view_test,name='view_test'),
   path('view_appointment_status/',view_appo_status,name="view_approvedappo"),
   path('delete_appo/<int:id>',delete_appointment,name='delete_appointment'),
   path('send_feedback/',user_feedback,name='user_feedback'),
   path('user_profile/',user_profile,name='user_profile'),
   # path('Generatedpdf/<int:id>',GeneratePdf,name="pdf"),
]
```

Day 5

“Patient Details”

Prompt: All the information about patients

Step 1: Go to ‘views.py’ location and write code.

```
def view_test(request,id):
    if 'user' in request.session.keys():
        b=Test_category.objects.get(pk=id)
        c=UserRegister.objects.get(uid=b.email)

        return render(request,'datatable.html',{'b':b,'c':c})
    else:
        return redirect('user:signin')
```

Step 2: Create ‘take_bloodtest.html’

```
{% extends 'admin_base.html' %}
{% load static %}
{% block content %}




<div class="dashboard">
```

```
   <div class="card" style="margin-top:100px; padding:0px 10px 0px 10px ;">
      <div class="baner" style="margin: auto; margin-top:-4%; margin-bottom:20px;">
               <h3 class="text-center" style="font-weight: bold; padding: 10px 0px 10px 0px;">
Patient Approved Records</h3>
      </div>

      <table class="table table-hover table-striped" id="myTable">
         <thead>
            <tr>
               <th>No</th>
               <th>Patient Name</th>
               <th>Phone</th>
               <th>Email</th>
               <th>Schedule</th>
               <th>Test type</th>

               <th>Action</th>

            </tr>
         </thead>

         <tbody>

                        {% for x in b %}
                        <tr>
                           <td>{{forloop.counter}}</td>
                           <td>{{x.name}}</td>
                           <td>{{x.phone}}</td>
                           <td>{{x.email}}</td>
                           <td>{{x.schedule}}</td>
                           <td>{{x.Test_name}}</td>
                              <td><a class="btn btn-danger" href="{% url 'laboratory:take_test' x.id
%}">Test</a></td>
                        </tr>
                        {% endfor %}

                                                                        </tbody>
         </tbody>

      </table>

   </div>
</div>
```

```
{% endblock %}
```

Step 3: Go to ‘models.py’ location and build model for product.

```
# models.py


class Test_category(models.Model):
        l_name=models.CharField(max_length=200,default='',verbose_name='Lab   Assistance
Name')
   lab_name=models.CharField(max_length=200,default='',verbose_name='Lab Name')
         Test_name=models.CharField(max_length=200,default='',verbose_name='Blood   Test
Name')
   patientname=models.CharField(max_length=200,default='',verbose_name='Patient Name')
   email = models.EmailField(default="")
   Haemoglobin=models.PositiveIntegerField(default="")
   RBC_Count=models.PositiveIntegerField(default="")
   PCV=models.PositiveIntegerField(default="")
   MCV=models.PositiveIntegerField(default="")
   MCH=models.PositiveIntegerField(default="")
   MCHC=models.PositiveIntegerField(default="")
   RDW=models.PositiveIntegerField(default="")
   Total_WBC_Count=models.PositiveIntegerField(default="")
   Neutrophils=models.PositiveIntegerField(default="")
   Lymphocytes=models.PositiveIntegerField(default="")
   Eosinophils=models.PositiveIntegerField(default="")
   Monocytes=models.PositiveIntegerField(default="")
   Basophils=models.PositiveIntegerField(default="")
   Platelet_Count =models.PositiveIntegerField(default="")
   WBCs_on_PS=models.CharField(max_length=50,default="")
   date_of_test=models.DateField(auto_created=True,auto_now=True)

   def __str__(self):
      return str(self.Test_name)

class Reference_Test_category(models.Model):
   Haemoglobin_reference = models.TextField(default="male : 14 - 16g \n Female : 12 - 14 g")
   RBC_reference = models.TextField(default="14 - 16g%")
   PCV_reference = models.TextField(default="35 - 45 %")
   MCV_reference = models.TextField(default="80 - 99 fl")
```

```
    MCH_reference = models.TextField(default="28 - 32 pg")
    MCHC_reference = models.TextField(default="30 - 34 %")
    RDW_reference = models.TextField(default="9 - 17 fl")
    Total_WBC_Count_reference = models.TextField(default="4000 - 11000 / cu.mm")
    Neutrophils_reference = models.TextField(default="40 - 75 %")
    Lymphocytes_reference = models.TextField(default="20 - 45 %")
    Eosinophils_reference = models.TextField(default="00 - 06 %")
    Monocytes_reference = models.TextField(default="00 - 10 %")
    Basophils_reference = models.TextField(default="00 - 01 %")
    Platelet_Count_reference = models.TextField(default="150000 - 450000 / cu.mm")

    def __str__(self):
        return str(self.Test_name)
```

Step 4: Migrate Models, after defining models, run the `makemigrations` and `migrate` commands to create corresponding database tables based on the model definitions.

```
python manage.py makemigrations
python manage.py migrate
```

Day 6

“Feedbacks”

Prompt: Get and view feedbacks from the patients

Step 1: Go to ‘views.py’ location and write code for feedback.

```
def user_feedback(request):
    if 'user' in request.session.keys():
        c=UserRegister.objects.get(id=request.session['userId'])
        print(c)
        if request.POST:
            name=request.POST['name']
            email=request.POST['email']
            Feedback=request.POST['Feedback']
            a=Userfeedback()
            a.username=name
            a.useremail=email
            a.feedback=Feedback
            a.save()
            return redirect('user:index')
        return render(request,'user_feedback.html',{'n':c})
    else:
        return redirect('user:signin')
```

```
def view_feedback(request):

    if 'Lid' in request.session:

        obj=Userfeedback.objects.all()

        return render(request,'view_feedback.html',{'b':obj})
```

```
    return redirect('laboratory:lab_signin')


def show_feedback(request,id):
    if 'Lid' in request.session:
        obj=Userfeedback.objects.get(id=id)
        return render(request,'show_feedback.html',{'b':obj})
    return redirect('laboratory:lab_signin')
```

Step 2: Create templates for BTS.

```
<!—user_feedback.html -->

{% extends 'user_base.html' %}
{% load static %}
{% block content %}

<div></div>

<div class="dashboard">
   <div class="card" style="margin-top:100px; padding:0px 10px 0px 10px ;">
      <div class="baner" style="margin: auto; margin-top:-4%; margin-bottom:20px;">
                <h3 class="text-center" style="font-weight: bold; padding: 10px 0px 10px 0px;">
Feedback</h3>
      </div>
      <form method="post" class="col-8 mt-5" style="margin:auto;">
         {% csrf_token%}

         <div class="form-group">
            <label>Patient name</label>
            <input type="text" placeholder="Enter name" class="form-control" name="name"
            value="{{n.userfname}}" readonly>
         </div>
         <div class="form-group">
            <label>Patient email</label>
            <input type="email" placeholder="Enter email" class="form-control" name="email"
               value="{{n.uid}}" readonly>
```

```
        </div>
        <div class="form-group">
           <label>write Feedback</label>
           <textarea type="text" placeholder="Enter password" class="form-control"
              name="Feedback"  required></textarea>
        </div>

        <br>
        <input type="submit" class="btn btn-block" value="submit"  />

     </form>
     </br>
  </div>
</div>


  {% endblock %}
```

```
<!—view_feedback.html 

{% extends 'admin_base.html' %}

{% load static %}

{% block content %}




<div class="dashboard">


   <div class="card" style="margin-top:100px; padding:0px 10px 0px 10px ;">

      <div class="baner" style="margin: auto; margin-top:-4%; margin-bottom:20px;">

            <h3 class="text-center" style="font-weight: bold; padding: 10px 0px 10px 0px;">
Patient Feedback Records </h3>

      </div>

      <table class="table table-hover table-striped" id="myTable">
```

```
        <thead>
          <tr>
            <th>No</th>
            <th>Patient NAME</th>
            <th>Patient Email</th>
            <th>feedback Date</th>
            <th>Read</th>
          </tr>
        </thead>
        <tbody>
          {% for i in b %}
          <tr>
            <td>{{forloop.counter}}</td>
            <td>{{ i.username }}</td>
            <td>{{ i.useremail }}</td>
            <td>{{ i.date_of_feedback }}</td>

            <td>
              <a href="{% url 'laboratory:show_feedback' i.id %}">view</a>

            </td>
          </tr>
          {% endfor %}
        </tbody>
      </table>
    </div>
  </div>
```

```
{% endblock %}
```

```
<!—show_feedback.html 

{% extends 'admin_base.html' %}

{% load static %}

{% block content %}


<div></div>


<div class="dashboard">

   <div class="card" style="margin-top:100px; padding:0px 10px 0px 10px ;">

      <div class="baner" style="margin: auto; margin-top:-4%; margin-bottom:20px;">

                <h3 class="text-center" style="font-weight: bold; padding: 10px 0px 10px 0px;">
Feedback</h3>

      </div>

      <form method="post" class="col-8 mt-5" style="margin:auto;">

         {% csrf_token%}


         <div class="form-group">

            <label>Patient name</label>

            <input type="text"  class="form-control" name="name"

            value="{{b.username}}" readonly>

         </div>

         <div class="form-group">

            <label>Patient email</label>

            <input type="email"  class="form-control" name="email"
```

```
                value="{{b.useremail}}" readonly>
            </div>
            <div class="form-group">
                <label>write Feedback</label>
                <textarea type="text" placeholder="Enter password" class="form-control"
                    required>{{b.feedback}}</textarea>
            </div>


            <br>



        </form>
        </br>
    </div>
</div>



    {% endblock %}
```

Step 3: Go to ‘models.py’ location and build model for feedback.

```
# app1 /models.py
class Userfeedback(models.Model):
    username=models.CharField(max_length=30,default='',verbose_name='Patient Name')
    useremail=models.EmailField(max_length=50,verbose_name='Patient Email')
    feedback=models.TextField(verbose_name='Patient feedback Area')
    date_of_feedback=models.DateField(auto_created=True,auto_now=True)
    def __str__(self):
        return self.username
```

Step 4: Migrate Models, after defining models, run the `makemigrations` and `migrate` commands to create corresponding database tables based on the model definitions.

```
python manage.py makemigrations
python manage.py migrate
```

Day 7

“Patient Appointments”

Prompt: Patients Appointment Information

Step 1: Go to ‘views.py’ location and write code.

```
import datetime
from django.utils.dateparse import parse_datetime

def book_appointment(request):
    if 'user' in request.session.keys():
        a=Name_category.objects.all()
        b=UserRegister.objects.get(id=request.session['userId'])
        form = appointmentform(request.POST)
        #print(form)
        if form.is_valid():
                                    c=Appointment.objects.filter(email=request.POST['email'])        &
Appointment.objects.filter(status='pending')
            if len(c)<=0:
                cDate = datetime.datetime.today()
                postDate = request.POST['schedule']
                postDateArray = postDate.split("T")
                cDateArray = str(cDate).split(" ")
                if parse_datetime(postDateArray[0]) >= parse_datetime(cDateArray[0]):
                    form.save()
                    messages.success(request,'Your appoinement is booked.')
                else:
                                                                                                return
render(request,'book_appointment.html',{'form':form,'ab':a,'b':b,'m':'select      future      date
only'})
            else:
                                                                                                return
render(request,'book_appointment.html',{'form':form,'ab':a,'b':b,'m':'appointment      alredy
booked'})
```

```
        return render(request,'book_appointment.html',{'form':form,'ab':a,'b':b})
    return redirect('user:signin')



def view_appo_status(request):
    if 'user' in request.session.keys():
        b=Appointment.objects.filter(email=request.session['user'])
        return render(request,'view_appointment.html',{'b':b})
    else:
        return redirect('user:signin')

def delete_appointment(request,id):
    if 'user' in request.session.keys():
        obj=Appointment.objects.get(pk=id)
        obj.delete()
        return redirect('user:view_approvedappo')
    return redirect('user:signin')
```

Step 2: Create templates for patient appointments.

```
<!—book_appointment..html -->

{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Book_appoinement</title>
    <!-- <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}"> -->

    <style>
        body{
    margin: 0;
    padding: 300px;

    background: #34495e;
    background-size: 100%;
    background-position: 100%;
```

```
    font-family: sans-serif;
}

body{
    margin: 0;
    padding: 0;
    font-family: sans-serif;
    background: #34495e;
  }

.book-appoinement{
    width: 1px;
    height: 470px;
    background: black;
    color: #fff;
    top: 50%;
    left: 50%;
    position: absolute;
    transform: translate(-50%,-50%);
    box-sizing: border-box;
    padding: 70px 30px;
}



.book-appoinement input[type="text"], input[type="email"]
{
    border: none;
    border-bottom: 1px solid #fff;
    padding-right: 10%;
    background: transparent;
    outline: none;
    height: 40px;
    color: #fff;
    font-size: 19px;
}
.box input[type = "datetime-local"]{
    border:0;
    background: none;
    display: block;
    margin: 20px auto;
    text-align: center;
    border: 2px solid #3498db;
    padding: 14px 10px;
```

```
    width: 200px;
    outline: none;
    color: white;
    border-radius: 24px;
    transition: 0.25s;
  }

  .box input[type = "text"]:focus{
    width: 280px;
    border-color: #2ecc71;
  }

.book-appoinement select[id="id_department"]{
    background-color: transparent;

  color: white;
  padding-right: 5%;
}


.button {
  background-color: #4c90af;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 19px;
}
.time  {
  background-color: #6f7a6f81; /* Green */
  border: none;
  color: white;
  margin-right: 10%;
  /* text-align: center; */
  text-decoration: none;
  /* display: inline-block; */
  font-size: 16px;
}

option{
    color: rgb(48, 47, 47);
    font-size: larger;
```

```
}

.box{
   width: 300px;
   padding: 40px;
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translate(-50%,-50%);
   background: #191919;
   text-align: center;
}



      </style>
</head>

<body>

   <div class="book-appoinement">

   <center>
   <!--  style="padding-top: 250px;
      font-size: x-large;" -->
      <form method="POST" class="box">
      {% csrf_token %}
         <div>
            <h1>Book AppointMent</h1>
         <label>Name:</label>
        <input type="text" name="name"  value="{{b.userfname}}" readonly>
      </div>
      <br>
      <div class="row">
      <div>
         <label>Email:</label>
        <input type="email" name="email"  value="{{b.uid}}" readonly >
      </div></div>
      <br>
      <div>
         <label>Phone:</label>
                        <input  type="text"  name="phone"  placeholder="Enter  phone  number"
value="{{b.usercontactno}}" readonly>
      </div>
```

```
      <br>
      <div class="d-department">
         <label for="id_department">Test Name:</label>

         <select name="Test_name" required id="id_department">
            {% for i in ab %}
          <option value="{{i.id}}"
          selected="selected">{{i.category_name}}</option>
          {% endfor %}
          </select>

      </div>
      <br>
      <div>
         <label>Schedule:</label>
         <input type="datetime-local" name="schedule" class="time">
      </div>
      <br>
       <input type="hidden" name="status" placeholder="status" value="pending" readonly>
      {% if messages %}
      {% for result in messages %}
     <center>  <b style=color:green;>{{result}}</b></center>
      {% endfor %}
      {% endif %}
      <br>

         <button class="button">Submit here</button>
         <br><br>
         <a href="{% url 'user:index' %}" style="color: teal;"> Home Page </a>
        </form>
</center>
</div>
{% if m %}
<script>alert("{{m}}")</script>
{% endif %}
</body>
</html>
```

Step 3: Go to ‘models.py’ location and build model for appointment.

```
# models.py


class Appointment(models.Model):
    g = (
        ('pending', 'pending'),
        ('approved', 'approved'),
        ('rejected', 'rejected'),
    )
    name = models.CharField(max_length=500)
    phone = models.IntegerField()
    email = models.EmailField(max_length=100)
    schedule = models.DateTimeField()
    Test_name=models.ForeignKey(Name_category, on_delete=models.CASCADE)
    status = models.CharField(max_length=12, choices=g, default = 'pending')
    appointment_booked=models.BooleanField(default=False)
    def __str__(self):
        return self.name
```

Step 4: Migrate Models, after defining models, run the `makemigrations` and `migrate` commands to create corresponding database tables based on the model definitions.

```
python manage.py makemigrations
python manage.py migrate
```

Day 8

“Status Details”

Prompt: Appointment Status on Lab Side

Step 1: Go to ‘app1/views.py’ location and write code for HBS.

```
def show_appo(request):
   results=Appointment.objects.filter(status='pending')
   return render(request,'show_appointment.html',{'book':results})



#book appointment
def edit_appo(request,id):
   if 'Lid' in request.session.keys():
      a=LaboratoryRegister.objects.get(Lid=request.session['Lid'])
      book=Appointment.objects.get(pk=id)
      if request.POST:
         book.status=request.POST['boked']
         book.save()
         return redirect('laboratory:show_appointment')
      return render(request,'edit_appointment.html',{'book':book,'owner_data':a})
   else:
      return redirect('laboratory:lab_signin')

def view_approvedappo(request):
   if 'Lid' in request.session:
                                                b=Appointment.objects.filter(status='approved')        &
Appointment.objects.filter(appointment_booked=False)

      return render(request,'take_bloodtest.html',{'b':b})
   else:
      return redirect('laboratory:lab_signin')
```

Step 2: Create template “edit_appointment.html” for BTS.

```
{% extends 'admin_base.html' %}
{% load static %}
{% block content %}
<!--/header-->

<section class="container">
   <div class="row justify-content-center p-4">
      <div class="col-12 col-md-6 p-5 card">
         <h3 class="text-uppercase font-weight-bold text-center">Appointment Edit</h3>
         <form method="POST" enctype="multipart/form-data" >
            {% csrf_token %}
            <br />
            <div class="form-group row">
               <label for="staticName" class="col-4 col-form-label">Patient Name</label>
               <div class="col-auto">
                        <input type="text" readonly class="form-control" id="staticName" value="{{
book.name }}" />
               </div>
            </div>
            <div class="form-group row">
               <label for="staticOwner" class="col-4 col-form-label">Phone</label>
        <div class="col-auto">
          <input
            type="text"
            readonly
            class="form-control"
            id="staticOwner"
            value="{{ book.phone }}"
          />
        </div>
      </div>
      <div class="form-group row">
        <label for="staticProperty" class="col-4 col-form-label"
          >Email</label
        >
        <div class="col-auto">
          <input
            type="text"
            readonly
```

```
          class="form-control"
          id="staticProperty"
          value="{{ book.email }}"
         />
        </div>
      </div>
      <div class="form-group row">
        <label for="staticName" class="col-4 col-form-label">Schedule</label>
        <div class="col-auto">
                    <input type="text" readonly class="form-control" id="staticName" value="{{
book.schedule }}" />
        </div>
      </div>
      <div class="form-group row">
        <label for="staticName" class="col-4 col-form-label">Test type</label>
        <div class="col-auto">
                    <input type="text" readonly class="form-control" id="staticName" value="{{
book.Test_name }}" />
        </div>
      </div>
      <div class="form-group row">
        <label
         class="col-auto col-form-label d-inline-block"
         for="customCheck1"
         >Want to Approve:</label
        >
        <div class="col-7">
         <select class="form-control w-100 text-center" id="boked" name="boked">
           <option value="none" selected>select</option>
           {% comment %} <option value="pending">pending</option> {% endcomment %}
           <option value="approved">Approve</option>
           <option value="rejected">Reject</option>
         </select>
        </div>
      </div>
      <div class="form-group row">
        <div class="col-12">
        <input type="submit" value="submit" class="btn btn-primary btn-block" />
        </div>
      </div>
    </form>
  </div>
 </div>
</section>
```

```
<!--  //News section -->

<!-- footers 20 -->
{% endblock content %}
```

Day 9

“Profile”

Prompt: Fetch profile details

Step 1: Go to ‘views.py’ location and write code for profile.

```
def lab_profile(request):
    if 'Lid' in request.session:
        a=LaboratoryRegister.objects.get(Lid=request.session['Lid'])
        if request.POST:
            name=request.POST['name']
            email=request.POST['email']
            password=request.POST['passowrd']
            a.Laboratoryfname=name
            a.Lid=email
            a.Laboratorypwd=password
            a.save()
            return redirect('laboratory:index')
        return render(request,'lab_profile.html',{'n':a})
    else:
        return redirect('laboratory:lab_signin')


def user_profile(request):
    if 'user' in request.session.keys():
        a=UserRegister.objects.get(id=request.session['userId'])
        if request.POST:
            name=request.POST['name']
            email=request.POST['email']
            password=request.POST['passowrd']
            a.userfname=name
            a.uid=email
            a.userpwd=password
            a.save()
            return redirect('user:index')
        return render(request,'user_profile.html',{'n':a})
```

```
else:
    return redirect('user:signin')
```

Step 2: Template “lab_profile.html”.

```
{% extends 'admin_base.html' %}

{% load static %}

{% block content %}


<div></div>


<div class="dashboard">

   <div class="card" style="margin-top:100px; padding:0px 10px 0px 10px ;">

      <div class="baner" style="margin: auto; margin-top:-4%; margin-bottom:20px;">

         <h3 class="text-center" style="font-weight: bold; padding: 10px 0px 10px 0px;"> Edit
Laboratory Profile</h3>

      </div>

      <form method="post" class="col-8 mt-5" style="margin:auto;">

         {% csrf_token%}


         <div class="form-group">

            <label>Enter name</label>

            <input type="text" placeholder="Enter name" class="form-control" name="name"

            value="{{n.Laboratoryfname}}" required>

         </div>

         <div class="form-group">

            <label>Enter email</label>

            <input type="email" placeholder="Enter email" class="form-control" name="email"
```

```
            value="{{n.Lid}}" readonly>
        </div>
        <div class="form-group">
          <label>Enter password</label>
          <input type="password" placeholder="Enter password" class="form-control"
            name="passowrd" value="{{n.Laboratorypwd}}" required>
        </div>


        <br>
        <input type="submit" class="btn btn-block" value="submit"  />


      </form>
      </br>
    </div>
  </div>



    {% endblock %}
```

Day 10

“Test Reports”

Prompt: Final Test Reports Information

Step 1: Go to ‘views.py’ location and write code.

```
def take_test(request,id):
    if 'Lid' in request.session:
        a=request.session['fname']
        c=request.session['name']
        b=Appointment.objects.get(pk=id)
        form =Testform(request.POST)
        if form.is_valid():
            b.appointment_booked=request.POST['test']
            form.save()
            b.save()

            return redirect('laboratory:index')
        return render(request,'bloodtest.html',{'a':a,'b':b,'c':c})
    else:
        return redirect('laboratory:lab_signin')


def view_test(request,id):
    if 'user' in request.session.keys():
        b=Test_category.objects.get(pk=id)
        c=UserRegister.objects.get(uid=b.email)

        return render(request,'datatable.html',{'b':b,'c':c})
    else:
        return redirect('user:signin')
```

Step 2: Go to ‘models.py’ location and build model.

```
# models.py


class Reference_Test_category(models.Model):
    Haemoglobin_reference = models.TextField(default="male : 14 - 16g \n Female : 12 - 14 g")
    RBC_reference = models.TextField(default="14 - 16g%")
    PCV_reference = models.TextField(default="35 - 45 %")
    MCV_reference = models.TextField(default="80 - 99 fl")
    MCH_reference = models.TextField(default="28 - 32 pg")
    MCHC_reference = models.TextField(default="30 - 34 %")
    RDW_reference = models.TextField(default="9 - 17 fl")
    Total_WBC_Count_reference = models.TextField(default="4000 - 11000 / cu.mm")
    Neutrophils_reference = models.TextField(default="40 - 75 %")
    Lymphocytes_reference = models.TextField(default="20 - 45 %")
    Eosinophils_reference = models.TextField(default="00 - 06 %")
    Monocytes_reference = models.TextField(default="00 - 10 %")
    Basophils_reference = models.TextField(default="00 - 01 %")
    Platelet_Count_reference = models.TextField(default="150000 - 450000 / cu.mm")

    def __str__(self):
        return str(self.Test_name)
```

Step 4: Migrate Models, after defining models, run the `makemigrations` and `migrate` commands to create corresponding database tables based on the model definitions.

```
python manage.py makemigrations
python manage.py migrate
```

Day 11

“Queries handling in Django framework”

Object-Relational Mapping (ORM) is an integral part of the framework, enabling developers to interact with databases using Python objects. Django's ORM abstracts away the complexity of SQL queries, allowing developers to focus on application logic rather than database operations directly. Here's an overview of how ORM queries work in Django:

1. Define Models: Django models are Python classes that represent database tables. Each attribute in the class corresponds to a field in the table. Models are defined in the `models.py` file of Django apps.

```
class Blog(models.Model):
    name = models.CharField(max_length=50)
    tagline = models.TextField()
    # author = models.CharField(max_length=20,default = "")

    def __str__(self):
        return self.name


class Author(models.Model):
    name = models.CharField(max_length=200)
    email = models.EmailField()

    def __str__(self):
        return self.name
```

2. Migrate Models: After defining models, run the `makemigrations` and `migrate` commands to create corresponding database tables based on the model definitions.

```
python manage.py makemigrations
python manage.py migrate
```

3. Perform Queries: Django provides a powerful API for querying databases using model objects. Various methods are available to perform CRUD operations, filter data, perform aggregations, and more.

```
def table(request):
   a = userregister.objects.get(name = 'a')
    print(a)
    for i in a:
      print(i.email)
   return render(request,'table.html',{'data':a})
```

Day 12

Custom Validations in the Django Models

Custom field validations in Django models can be implemented using the `clean()` method or by overriding the `save()` method. Here's how you can implement custom field validations using both methods:

```
class vendor(models.Model):

    name = models.CharField(max_length=50)

    email = models.EmailField()

    mob = models.CharField(max_length=10)

    add = models.TextField()

    password = models.CharField(max_length=10)

    typeofbusiness = models.CharField(max_length = 100)


    def __str__(self):

        return self.name




class order(models.Model):

    userid = models.CharField(max_length = 10)

    proid = models.CharField(max_length = 10)

    totalamount = models.CharField(max_length = 10)
```

```
qty = models.CharField(max_length = 10)

add = models.TextField()

city = models.CharField(max_length = 30)

state =models.CharField(max_length = 30)

pincode = models.CharField(max_length = 6)

paymentype = models.CharField(max_length = 30)

transactionid = models.CharField(max_length = 100)

datetime = models.DateTimeField(auto_now = True)


def __str__(self):

    return self.userid
```

The `clean()` method is overridden to perform custom validation on the `your_field` attribute. If the validation fails, a `ValidationError` is raised. Make sure to call the parent class's `clean()` method at the end of your custom `clean()` method.

```
class cart(models.Model):

    productid = models.CharField(max_length=10)

    userid = models.CharField(max_length=10)

    quantity = models.CharField(max_length=10)

    totalprice = models.CharField(max_length=10)

    orderid = models.CharField(max_length=10)



    def __str__(select):

        return self.userid
```

The `save()` method is overridden to perform custom validation before saving the object. If the validation fails, a `ValidationError` is raised. Make sure to call the parent class's `save()` method at the end of your custom `save()` method.

Choose the method that best fits your use case. Generally, the `clean()` method is preferred for field-specific validations, while the `save()` method is more suitable for object-level validations.

In Django, the `Meta` inner class within a model allows you to define metadata about the model. This metadata can include options such as database table name, ordering, indexes, permissions, and more. Here's an example of how you can use the `Meta` class in Django models:

- The `Meta` class is used to define metadata for the `YourModel` class.

- `verbose_name` specifies a human-readable name for the model. It's used in various Django admin interfaces.

- `verbose_name_plural` specifies the plural form of the model name.

- `ordering` specifies the default ordering for queries on this model. In this case, it orders the query results by the `field2` attribute in descending order.

You can include other options in the `Meta` class based on your requirements. Here are some other common options:

- `db_table`: Specifies the name of the database table for the model.

- `unique_together`: Specifies sets of fields that, taken together, must be unique.

- `indexes`: Specifies database indexes for one or more fields.

- `permissions`: Specifies permissions associated with the model.

Day 13
Filters in Python

In Python, filters are a way to selectively extract elements from a collection (like a list) based on a condition. They're particularly useful when you want to apply a certain criterion to each element of a collection and only retain those elements that satisfy the criterion.

The `filter()` function in Python takes two arguments: a function (or `None`) and an iterable (like a list). It returns an iterator yielding those items of the iterable for which the function returns true. If the function is `None`, it simply returns the elements that are true.

Here's a basic example to illustrate how `filter()` works:

```
def index(request):
    if 'email' in request.session:
        b = userregister.objects.get(email = request.session['email'])
        a = category.objects.all()
        return render(request,'index.html',{'category':a,'session':b})
    elif 'vendoremail' in request.session:
        a = category.objects.all()
        b = vendor.objects.get(email = request.session['vendoremail'])
        return render(request,'index.html',{'category':a,'vensession':b})
    else:
        a = category.objects.all()
        return render(request,'index.html',{'category':a})
```

```
def reg(request):
```

```
    # c = userregister.objects.get(name = 'b')
    if request.method == 'POST':
        a = userregister()
        a.name = request.POST['uname']
        a.email = request.POST['email']
        a.mob = request.POST['mob']
        a.add = request.POST['add']
        pasword = request.POST['password']
        print(pasword,"1111111111111111111111111111111111111")
        encrypted_pass = encrypt(pasword)
        print(encrypted_pass,"2222222222222222222222")
        # check_password(encrypted_pass, pasword)
        a.password = encrypted_pass
        b = userregister.objects.filter(email = request.POST['email'])
        error_msg = None
        if a.email:
            if len(a.mob) == 10:
                if len(b) > 0:
                    return render(request,'register.html',{'email':'This email is already registered..'})
                else:
                    if request.POST['password'] == request.POST['cp']:
                        a.save()
                        return render(request,'register.html',{'save':'Data stored succesfully....'})
                    else:
                        return render(request,'register.html',{'pass':'Passwords did not matched...'})
            else:
                error_msg = "Phone number must be 10 didgits.."
                return render(request,'register.html',{'error':error_msg})
        else:
            error_msg = "Email field is required.."
            return render(request,'register.html',{'error':error_msg})
    else:
        return render(request,'register.html',{})
```

This is a basic example of how you can implement product functionality using ORM queries with SQLAlchemy in Python. Depending on your specific requirements and database schema, you may need to adjust the code accordingly.

Day 14
Web Page and Navigation using Session

To build a common web page with navigation using sessions in Python, you typically use a web framework like Flask or Django. Here's an example using Flask, a lightweight and easy-to-use web framework:

1. First, you need to install Flask if you haven't already:

```
pip install flask
```

2. Now, let's create a simple Flask application that demonstrates session-based navigation. We'll have three pages: Home, About, and Contact.

```
from flask import Flask, render_template, session, redirect, url_for

app = Flask(__name__)
app.secret_key = 'your_secret_key'  # Needed for session encryption

@app.route('/')
def home():
    return render_template('home.html')

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/contact')
def contact():
    return render_template('contact.html')

@app.route('/login')
def login():
    # Here you can set session variables like user authentication status, etc.
    session['logged_in'] = True
    return redirect(url_for('home'))

@app.route('/logout')
```

```
def logout():
    # Clear session variables
    session.clear()
    return redirect(url_for('home'))

if __name__ == '__main__':
    app.run(debug=True)
```

3. Create HTML templates for each page. You can use Jinja2 templating engine provided by Flask for dynamic content.

```
`templates/home.html`:
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home</title>
</head>
<body>
    <h1>Home Page</h1>
    <p>Welcome to our website!</p>
    <nav>
        <a href="/">Home</a> |
        <a href="/about">About</a> |
        <a href="/contact">Contact</a>
        {% if session.logged_in %}
            | <a href="/logout">Logout</a>
        {% else %}
            | <a href="/login">Login</a>
        {% endif %}
    </nav>
</body>
</html>
```

```
`templates/about.html`:
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About</title>
</head>
<body>
    <h1>About Us</h1>
    <p>We are a company that does amazing things.</p>
    <nav>
        <a href="/">Home</a> |
        <a href="/about">About</a> |
        <a href="/contact">Contact</a>
        {% if session.logged_in %}
            | <a href="/logout">Logout</a>
        {% else %}
            | <a href="/login">Login</a>
        {% endif %}
    </nav>
</body>
</html>
```

```
`templates/contact.html`:
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact</title>
</head>
<body>
    <h1>Contact Us</h1>
    <p>You can reach us at: example@example.com</p>
    <nav>
        <a href="/">Home</a> |
        <a href="/about">About</a> |
        <a href="/contact">Contact</a>
        {% if session.logged_in %}
            | <a href="/logout">Logout</a>
        {% else %}
            | <a href="/login">Login</a>
        {% endif %}
```

```
    </nav>
</body>
</html>
```

4. Run your Flask application:

```
python your_app_name.py
```

Now, you can navigate through the pages and see that the navigation bar updates based on whether the user is logged in or not. The `session` object is used to keep track of the user's authentication status across requests. When the user logs in or logs out, the `session` object is updated accordingly.

Activity 2:Wishlist Management for the ecommerce in python with django
Prompt: Wishlist Management for the ecommerce in python with django

Implementing wishlist management for an e-commerce platform using Django involves creating models, views, templates, and handling user authentication. Here's a step-by-step guide to achieve this:

1. Set up Django Project: If you haven't already, create a Django project and app:

```bash
django-admin startproject ecommerce_project
cd ecommerce_project
python manage.py startapp wishlist
```

2. **Define Models**: Define the models for the wishlist functionality. A basic model might look like this:

```python
# In wishlist/models.py

from django.db import models
from django.contrib.auth.models import User

class WishlistItem(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
```

```
    product_name = models.CharField(max_length=100)
    product_description = models.TextField()
    added_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.product_name
```

3. **Create Views**: Create views to handle adding, viewing, and removing items from the wishlist.

```python
# In wishlist/views.py

from django.shortcuts import render, redirect
from django.contrib.auth.decorators import login_required
from .models import WishlistItem

@login_required
def wishlist(request):
    wishlist_items = WishlistItem.objects.filter(user=request.user)
    return render(request, 'wishlist/wishlist.html', {'wishlist_items': wishlist_items})

@login_required
def add_to_wishlist(request, product_id):
    # Logic to add product to wishlist
    pass

@login_required
def remove_from_wishlist(request, product_id):
    # Logic to remove product from wishlist
    pass
```

4. **Create URLs**: Define URLs for the wishlist views.

```python
# In wishlist/urls.py

from django.urls import path
```

```
from . import views

urlpatterns = [
    path('', views.wishlist, name='wishlist'),
    path('add/<int:product_id>/', views.add_to_wishlist, name='add_to_wishlist'),
    path('remove/<int:product_id>/', views.remove_from_wishlist, name='remove_from_wishlist'),
]
```

5. **Set up Templates**: Create templates for displaying the wishlist.

```html
<!-- In templates/wishlist/wishlist.html -->

{% extends 'base.html' %}
{% block content %}
    <h1>Wishlist</h1>
    {% if wishlist_items %}
        <ul>
            {% for item in wishlist_items %}
                <li>{{ item.product_name }} - {{ item.product_description }}</li>
            {% endfor %}
        </ul>
    {% else %}
        <p>Your wishlist is empty.</p>
    {% endif %}
{% endblock %}
```

6. **Wire up Authentication**: Ensure user authentication is set up properly in your Django project.

7. **Include Wishlist URLs**: Include wishlist URLs in the main project's URL configuration.

```python
# In ecommerce_project/urls.py

from django.contrib import admin
from django.urls import path, include
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('wishlist/', include('wishlist.urls')),
]
```

8. **Run Migrations**: Run migrations to create the necessary database tables.

```bash
python manage.py makemigrations
python manage.py migrate
```

9. **Add Links to Wishlist**: In your e-commerce platform's product pages, add links to add/remove products from the wishlist.

This is a basic outline to get you started with wishlist management in Django. Depending on your specific requirements, you may need to extend this functionality further, such as adding AJAX support for adding/removing items, handling quantity, etc.

Day 15
SQLite Database integration in Django

SQLite is a lightweight and serverless database engine that is well-suited for development and small-scale applications. Django, a high-level Python web framework, supports SQLite out of the box, making it easy to get started with database-driven web applications.

Here's an overview of how to use SQLite database with Django and how to establish a connection:

1. **Configure Django Settings**:
   In your Django project's settings file (`settings.py`), ensure that Django is configured to use SQLite as the default database engine. By default, Django comes pre-configured to use SQLite for development purposes.

```python
# In settings.py

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

2. **Create Django Models**:
   Define your application's data models using Django's ORM (Object-Relational Mapping). These models will be used to create corresponding database tables.

```python
# In models.py

from django.db import models

class Product(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
```

```
    price = models.DecimalField(max_digits=10, decimal_places=2)
```

3. **Make Migrations**:
   Django's migration system allows you to propagate changes you make to your models (adding fields, deleting models, etc.) into your database schema.

```bash
python manage.py makemigrations
```

This command creates migration files based on the changes detected in your models.

4. **Apply Migrations**:
   Apply the migrations to create the corresponding database tables.

```bash
python manage.py migrate
```

This command executes the migration files to create the database tables.

5. **Interact with the Database**:
   Now that your database is set up, you can interact with it using Django's ORM. You can perform CRUD (Create, Read, Update, Delete) operations on your models.

```python
# In views.py

from django.shortcuts import render
from .models import Product

def product_list(request):
    products = Product.objects.all()
    return render(request, 'product_list.html', {'products': products})
```

6. **Run Django Development Server**:
   Start the Django development server to test your application.

```bash
python manage.py runserver
```

7. **Access Admin Interface (Optional)**:
   Django provides a built-in admin interface that allows you to perform CRUD operations on your models. To access it, you need to create a superuser first.

```bash
python manage.py createsuperuser
```

Follow the prompts to create a superuser, and then you can access the admin interface at `http://localhost:8000/admin`.

That's it! You've established a connection to SQLite database in Python with Django and created models to interact with the database. You can now start building your web application with Django and SQLite.

To use MySQL with Django, you need to set up your Django project to use the MySQL database engine. Here's how you can create and operate a MySQL database in Python with Django:

1. **Install MySQL Database Server**:
   First, ensure that you have MySQL installed on your system. You can download and install MySQL from the official website: [MySQL Downloads](https://dev.mysql.com/downloads/)

2. **Install MySQL Client Library**:
   You also need to install the MySQL client library for Python. You can install it using pip:

   ```bash
   pip install mysqlclient
   ```

3. **Create MySQL Database**:
   Create a new database in your MySQL server that will be used by your Django project.

4. **Configure Django Settings**:
   In your Django project's settings file (`settings.py`), update the database configuration to use MySQL.

```python
# In settings.py

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'your_database_name',
        'USER': 'your_mysql_username',
        'PASSWORD': 'your_mysql_password',
        'HOST': 'localhost',   # Or the hostname of your MySQL server
        'PORT': '3306',        # Default MySQL port
    }
}
```

Replace `'your_database_name'`, `'your_mysql_username'`, and `'your_mysql_password'` with your actual MySQL database name, username, and password.

5. **Create Django Models**:
   Define your Django models as you would normally. Here's an example:

```python
# In models.py

from django.db import models

class Product(models.Model):
    name = models.CharField(max_length=100)
    description = models.TextField()
    price = models.DecimalField(max_digits=10, decimal_places=2)
```

6. **Make Migrations and Apply Migrations**:
   Run the following commands to create and apply migrations:

```bash
python manage.py makemigrations
python manage.py migrate
```

7. **Interact with the Database**:
   Now you can interact with your MySQL database using Django's ORM. For example, you can create, read, update, and delete records:

   ```python
   # In views.py

   from django.shortcuts import render
   from .models import Product

   def product_list(request):
       products = Product.objects.all()
       return render(request, 'product_list.html', {'products': products})
   ```

8. **Run Django Development Server**:
   Start the Django development server to test your application:

   ```bash
   python manage.py runserver
   ```

By following these steps, you can create and operate a MySQL database in Python with Django. Ensure that your MySQL server is running and accessible from your Django project.

Day 16
Razorpay Django

To integrate Razorpay payment gateway with Django, you need to follow these steps:

1. **Create a Razorpay Account**:
   Sign up for a Razorpay account if you haven't already. Once signed up, you will get access to the Razorpay dashboard where you can generate API keys.

2. **Install Razorpay Python SDK**:
   You can use the official Razorpay Python SDK to integrate Razorpay with your Django project. Install it using pip:

   ```bash
   pip install razorpay
   ```

3. **Configure Razorpay API Keys**:
   In your Django settings file (`settings.py`), add your Razorpay API keys:

   ```python
   # In settings.py

   RAZORPAY_API_KEY = 'your_api_key'
   RAZORPAY_API_SECRET = 'your_api_secret'
   ```

4. **Create Payment Form**:
   Create a form in your Django template where users can enter payment details:

   ```html
   <!-- In payment_form.html -->

   <form id="paymentForm" action="{% url 'process_payment' %}" method="POST">
       {% csrf_token %}
       <input type="text" name="amount" placeholder="Amount">
       <input type="submit" value="Pay Now">
   </form>
   ```

5. **Process Payment View**:
   Create a view in Django to handle payment processing:

   ```python
   # In views.py

   from django.shortcuts import render
   from django.http import JsonResponse
   import razorpay
   from django.conf import settings

   def process_payment(request):
       if request.method == 'POST':
           amount = int(request.POST['amount']) * 100  # Convert amount to paisa (Razorpay expects amount in paisa)
           client = razorpay.Client(auth=(settings.RAZORPAY_API_KEY, settings.RAZORPAY_API_SECRET))
           payment_data = {
               'amount': amount,
               'currency': 'INR',
               'receipt': 'order_rcptid_11',
               'payment_capture': '1'
           }
           payment = client.order.create(data=payment_data)
           return JsonResponse(payment)
       return render(request, 'payment_form.html')
   ```

6. **Handle Payment Response**:
   After the payment is processed, Razorpay will send a webhook or redirect the user back to a URL specified by you. You need to handle this response to update the payment status in your database.

7. **Update Payment Status View**:
   Create a view to handle the payment status update:

   ```python
   # In views.py

   def update_payment_status(request):
   ```

```
    if request.method == 'POST':
        # Process Razorpay payment webhook or redirect response
        # Update payment status in your database
        return JsonResponse({'status': 'Payment status updated successfully'})
```

8. **URL Configuration**:
   Configure URLs to map views:

```python
# In urls.py

from django.urls import path
from . import views

urlpatterns = [
    path('', views.process_payment, name='process_payment'),
    path('update_payment_status/', views.update_payment_status, name='update_payment_status'),
]
```

9. **Update Templates**:
   Update your templates to include the payment form and handle payment responses.

10. **Testing**:
    Test the payment flow in your development environment. Ensure that payments are processed successfully and payment status is updated in your database.

Remember to handle exceptions and errors gracefully, and follow best practices for security, such as validating input and using HTTPS for sensitive transactions. Additionally, refer to the Razorpay documentation for more details on handling webhooks and verifying payment responses.

Day 17
Git hub version control

Version control and collaboration using Git and GitHub are essential for managing Python and Django projects effectively, especially when working in teams or contributing to open-source projects. Here's an overview of how Git and GitHub are used in Python and Django development:

Version Control with Git:

1. **What is Version Control?**
   Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. Git is one of the most popular distributed version control systems.

2. **Basic Concepts:**
   - **Repository (Repo):** A repository is a collection of files and their revision history. It can exist locally on your machine or remotely on a server.
   - **Commit:** A commit is a snapshot of the repository at a specific point in time. It records changes to one or more files.
   - **Branch:** A branch is a parallel version of a repository. It allows you to work on new features or fixes without affecting the main codebase.
   - **Merge:** Merging is the process of combining changes from one branch into another.
   - **Pull Request (PR):** A pull request is a request to merge changes from one branch into another. It's commonly used for code review and collaboration.

3. **Workflow:**
   - **Feature Branch Workflow:** Developers create a new branch for each new feature or bug fix. Once the changes are complete, they create a pull request to merge their branch into the main branch.

Collaboration with GitHub:

1. **What is GitHub?**
   GitHub is a web-based platform that provides hosting for Git repositories. It offers collaboration features such as issue tracking, pull requests, and project management tools.

2. **Key Features:**
   - **Repositories:** GitHub hosts repositories where you can store your code.

- **Issues:** You can create, assign, and track issues (bugs, feature requests, etc.) using GitHub's issue tracker.
- **Pull Requests:** Pull requests facilitate code review and collaboration. They allow contributors to propose changes and discuss them before merging into the main codebase.
- **Branch Protection:** GitHub allows you to protect certain branches from direct pushes and require pull requests for changes, ensuring code quality and stability.
- **Actions and Workflows:** GitHub Actions enable you to automate workflows such as testing, building, and deployment directly from your repository.

3. **Collaboration Workflow:**
   - **Forking:** Contributors typically fork a repository to their account, make changes in their forked repository, and then create a pull request to merge those changes back into the original repository.
   - **Code Review:** Pull requests enable code review by allowing team members to comment, suggest changes, and discuss code changes before merging.
   - **Continuous Integration (CI) / Continuous Deployment (CD):** GitHub Actions can be used to set up CI/CD pipelines, automating the testing and deployment process.

Summary:

Version control with Git and collaboration with GitHub are fundamental for managing Python and Django projects efficiently. By leveraging these tools, developers can track changes, collaborate effectively, and maintain code quality throughout the development lifecycle. Whether working solo or in a team, understanding version control and collaboration workflows is crucial for successful project management in Python and Django development.

Activity: Exception handling and ORM Queries for Login Management
Prompt: Exception handling and ORM Queries for Login Management in Python with Django

Exception handling and ORM queries are crucial aspects of implementing login management in Python with Django. Below, I'll provide a basic example of how to handle exceptions and perform ORM queries for login management in a Django project:

1. **Exception Handling**:
   Exception handling is essential to gracefully handle errors that may occur during login operations, such as invalid credentials or database errors. Django provides built-in mechanisms for handling exceptions.

```python
```

```
from django.contrib.auth import authenticate, login
from django.contrib.auth.models import User
from django.http import HttpResponse
from django.shortcuts import render, redirect

def user_login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            # Handle invalid credentials
            return render(request, 'login.html', {'error': 'Invalid username or password'})
    else:
        return render(request, 'login.html')
```

2. **ORM Queries**:

Django's ORM (Object-Relational Mapping) provides a powerful interface for interacting with the database. You can use ORM queries to retrieve user information from the database for authentication purposes.

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

def user_login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        try:
            user = User.objects.get(username=username)
            if user.check_password(password):
                login(request, user)
                return redirect('home')
            else:
                # Handle invalid password
                return render(request, 'login.html', {'error': 'Invalid password'})
```

```
        except User.DoesNotExist:
            # Handle user not found
            return render(request, 'login.html', {'error': 'User does not exist'})
    else:
        return render(request, 'login.html')
```

In this example, we attempt to retrieve the user from the database using `User.objects.get()` and then validate the password using `user.check_password()`. We use exception handling (`User.DoesNotExist`) to handle cases where the user does not exist in the database.

Ensure that you have a login template (`login.html`) where users can input their credentials and submit the login form. Handle and display any error messages appropriately in the login template.

By combining exception handling and ORM queries, you can implement robust login management in Python with Django, ensuring that your application gracefully handles various scenarios that may arise during the authentication process.

Day 18
Django Email integration

Integrating email functionality into a Django project is a common requirement, whether it's for user registration, password reset, or sending notifications. Django provides a convenient way to handle email functionality through its built-in `send_mail()` function and EmailMessage class. Here's how you can integrate email functionality into your Django project:

1. **Configure Email Settings in `settings.py`**:
   First, configure the email backend settings in your Django project's `settings.py` file. You can use SMTP for sending emails. Here's an example configuration using Gmail SMTP:

```python
# settings.py

EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_USE_TLS = True
EMAIL_HOST_USER = 'your_email@gmail.com'
EMAIL_HOST_PASSWORD = 'your_email_password'
```

   Replace `'your_email@gmail.com'` and `'your_email_password'` with your Gmail email address and password. Alternatively, you can use other SMTP servers or email providers.

2. **Sending Email Using `send_mail()` Function**:
   You can use the `send_mail()` function provided by Django to send simple emails. Here's an example of sending a basic email:

```python
from django.core.mail import send_mail

send_mail(
    'Subject here',
    'Here is the message.',
    'from@example.com',
    ['to@example.com'],
    fail_silently=False,
)
```

```

You can call this function from any Django view or task where you want to send an email.

3. **Sending Email Using `EmailMessage` Class**:
For more advanced email functionality, such as sending HTML emails or attaching files, you can use the `EmailMessage` class. Here's an example:

```python
from django.core.mail import EmailMessage

email = EmailMessage(
    'Subject here',
    'Here is the message.',
    'from@example.com',
    ['to@example.com'],
)
email.content_subtype = "html"  # Set the content type to HTML
email.attach_file('/path/to/file.pdf')  # Attach a file
email.send()
```

4. **Sending Templated Emails**:
You can use Django's template system to create HTML email templates. First, create an HTML template for your email (e.g., `email_template.html`), then render it using Django's template engine:

```python
from django.core.mail import EmailMessage
from django.template.loader import render_to_string

html_content = render_to_string('email_template.html', {'context_variable': 'value'})
email = EmailMessage(
    'Subject here',
    html_content,
    'from@example.com',
    ['to@example.com'],
)
email.content_subtype = "html"  # Set the content type to HTML
email.send()
```
```

```

Ensure that you have created a template file (`email_template.html`) in your Django templates directory.

5. **Asynchronous Email Sending**:
For long-running tasks or when you want to offload email sending to a background process, you can use Django's `django.core.mail.send_mail()` method with `asyncio` or `celery`.

Here's a basic example using `asyncio`:

```python
import asyncio
from django.core.mail import send_mail

async def send_email_async():
    await asyncio.sleep(1)  # Simulate asynchronous operation
    send_mail(
        'Subject here',
        'Here is the message.',
        'from@example.com',
        ['to@example.com'],
        fail_silently=False,
    )

asyncio.run(send_email_async())
```

For `celery`, you need to set up a Celery worker and task to handle email sending asynchronously.

With these steps, you can integrate email functionality into your Python Django project, enabling you to send various types of emails, including simple text emails, HTML emails, templated emails, and asynchronously sent emails.
```

Day 19

Ecom Cart

Implementing cart management for an e-commerce website using Python and Django involves several steps. Below is a basic guide on how to implement cart functionality:

1. **Create Cart Model**:
First, create a model to represent the cart and its items in your Django application.

```
def cartdata(request):
    if 'email' in request.session:
        prolist = []
        b = userregister.objects.get(email = request.session['email'])
        a = cart.objects.filter(userid = request.session['userid'],orderid = "0")
        print(a,111111111111111111111111111111111111111)
        totalamount = 0
        for i in a:
            totalamount += int(i.totalprice)
            pro = product.objects.get(id = i.productid)
            prodict = {'id':i.productid,'proimage':pro.image,'proprice':pro.price,'total':i.totalprice,'userqty':i.quantity}
            prolist.append(prodict)
        if 'stock' in request.session:
            del  request.session['stock']
            return render(request,'cart.html',{'session':b,'prolist':prolist,'totalamount':totalamount,'stock':"Kale avje!!"})
        else:
            return render(request,'cart.html',{'session':b,'prolist':prolist,'totalamount':totalamount})
        return render(request,'cart.html',{'session':b,'prolist':prolist,'totalamount':totalamount})
    else:
        return redirect('login')
```

2. **Add Cart**:

Associate each user with their cart. You can do this by extending the user model or using a OneToOneField to link users to their carts.

```
def additem(request,id):
    if 'email' in request.session:
        # b = userregister.objects.get(email = request.session['email'])
        a = cart.objects.get(userid = request.session['userid'],productid = id,orderid = "0")
        b = product.objects.get(id = a.productid)
        if b.qty <= 0:
            request.session['stock'] = 0
            return redirect('cart')
        else:
            a.quantity = int(a.quantity) + 1
            a.totalprice= int(a.totalprice) + int(b.price)
            a.save()
            b.qty = b.qty - 1
            b.save()
            print(a,"cart qunatity 1111111111111111111111111111111111")
            return redirect('cart')
    else:
        return redirect('login')
```

Day 20
Remove Cart Function

```
def minus(request,id):
    if 'email' in request.session:
        a = cart.objects.get(userid = request.session['userid'],productid = id,orderid = "0")
        b = product.objects.get(id = a.productid)
        if int(a.quantity) <= 1:
            a.delete()
            return redirect('cart')
        else:
            a.quantity = int(a.quantity) - 1
            a.totalprice= int(a.totalprice) - int(b.price)
            a.save()
            b.qty = int(b.qty) + 1
            b.save()
            return redirect('cart')
    else:
        return redirect('login')
```

Remove all items makes cart empty, so user have to add again products into the cart. It will also completely remove the data from the model itself.

```
def removeall(request):
    if 'email' in request.session:
        a = cart.objects.filter(userid = request.session['userid'])
        for i in a:
            b = product.objects.get(id = i.productid)
            b.qty = int(b.qty) + int(i.quantity)
            b.save()
        a.delete()
        return redirect('cart')
    else:
```

```
    return redirect('login')
```

Day 21

Cart Template

1. cart.html

```
{% extends "nav.html" %}
{% load static %}
{% block abc %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>E-commerce</title>
    <link
      rel="stylesheet"
      45
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css"

integrity="sha512-z3gLpd7yknf1YoNbCzqRKc4qyor8gaKU1qmn+CShxbuBusANI9QpRohGBreCFkKxLhei6S9CQXFEbbKuqLg0DA=="
      crossorigin="anonymous"
      referrerpolicy="no-referrer"
    />
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
```

```
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href={% static "css/ecom.css" %} />
 </head>
 <body>
  <!-------------------------- Navbar ------------------------->


  <!-------------------------- Cart Section ------------------------->
  <div class="container mt-5">
   <h1 class="mb-4">Your Shopping Cart</h1>
   <div class="row">
    <div class="col-md-8">
     <table class="table">
      <thead>
       <tr>
        <th>Product</th>
        <th>Price</th>
        <th>Quantity</th>
        <th>Total</th>
        <th class="align-bottom pb-2">
         <a href={% url "removeall" %}><button class="btn
btn-outline-danger">Remove All</button></a>
        </th>
       </tr>
      </thead>
      <tbody>
       {% for i in prolist %}
       <tr class="align-content-center">
        <td>
```

```
            <img src={{i.proimage.url}} alt="" height="200px" width="200px" />
          </td>
          <td>₹{{i.proprice}}</td>
          <td>
            <a href={% url "add1" i.id %}><button class="increase items-count btn mr-1" type="button">
              <i class="fa fa-plus text-dark"></i>
            </button></a>
            <button class="border-0 btn">{{i.userqty}}</button>
            <a href={% url "minus" i.id %}><button class="increase items-count btn" type="button">
              <i class="fa fa-minus text-dark"></i>
            </button></a>
          </td>
          <td>₹{{i.total}}</td>
          <td class="align-bottom pb-3">
            <button class="btn btn-outline-danger">Remove</button>
          </td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
  <div class="col-md-3 ml-auto mt-0 pt-0">


    <div class="card p-2">
      <div class="card-body">
        <h5 class="card-title">Summary</h5>
        <hr />
```

```
          <p><strong>Total: </strong>₹{{totalamount}}</p>
          <a href={% url "checkout" %}>
          <button class="btn btn-outline-success btn-block">
            Checkout
          </button>
        </a>
        </div>
      </div>
    </div>
    {% if stock %}
    <script>
      alert('{{stock}}')
    </script>
    {% endif %}
  </div>
</div>
<!-----------------------Footer Section----------------------->
<footer class="bg-dark text-light py-4">
  <div class="container">
    <div class="row">
      <div class="col-md-4">
        <h5>Contact Us</h5>
        <p>Email: info@example.com</p>
        <p>Phone: 123-456-7890</p>
      </div>
      <div class="col-md-4">
        <h5>Links</h5>
        <ul class="list-unstyled">
```

```
          <li><a href="#">Home</a></li>
          <li><a href="#">Products</a></li>
          <li><a href="#">About Us</a></li>
          <li><a href="#">Contact Us</a></li>
        </ul>
      </div>
      <div class="col-md-4">
        <h5>Follow Us</h5>
        <ul class="list-inline">
          <li class="list-inline-item">
            <a href="#"><i class="fab fa-facebook"></i></a>
          </li>
          <li class="list-inline-item">
            <a href="#"><i class="fab fa-twitter"></i></a>
          </li>
          <li class="list-inline-item">
            <a href="#"><i class="fab fa-instagram"></i></a>
          </li>
        </ul>
      </div>
    </div>
    <hr />
    <div class="row">
      <div class="col text-center">
        <p>&copy; 2023 Your E-Commerce Store</p>
      </div>
    </div>
  </div>
```

```
    </footer>
  </body>
</html>
{% endblock abc %}
```

Day 22

Checkout Template

1. checkout.html

```
{% extends 'nav.html' %}

{% block abc %}

<!doctype html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

    <meta name="description" content="">

    <meta name="author" content="Mark Otto, Jacob Thornton, and Bootstrap
contributors">

    <meta name="generator" content="Hugo 0.101.0">

    <title>Checkout example · Bootstrap v4.6</title>



    <link rel="canonical"
href="https://getbootstrap.com/docs/4.6/examples/checkout/">





    <!-- Bootstrap core CSS -->

<link href="https://getbootstrap.com/docs/4.6/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHsD9ESMhqIYd0nLMwNL
D69Npy4HI+N" crossorigin="anonymous">
```

```
  <!-- Favicons -->
<link rel="apple-touch-icon"
href="/docs/4.6/assets/img/favicons/apple-touch-icon.png" sizes="180x180">
<link rel="icon" href="/docs/4.6/assets/img/favicons/favicon-32x32.png"
sizes="32x32" type="image/png">
<link rel="icon" href="/docs/4.6/assets/img/favicons/favicon-16x16.png" sizes="16x16"
type="image/png">
<link rel="manifest" href="/docs/4.6/assets/img/favicons/manifest.json">
<link rel="mask-icon" href="/docs/4.6/assets/img/favicons/safari-pinned-tab.svg"
color="#563d7c">
<link rel="icon" href="/docs/4.6/assets/img/favicons/favicon.ico">
<meta name="msapplication-config"
content="/docs/4.6/assets/img/favicons/browserconfig.xml">
<meta name="theme-color" content="#563d7c">



  <style>
    .bd-placeholder-img {
      font-size: 1.125rem;
      text-anchor: middle;
      -webkit-user-select: none;
      -moz-user-select: none;
      -ms-user-select: none;
      user-select: none;
    }

    @media (min-width: 768px) {
      .bd-placeholder-img-lg {
```

```
      font-size: 3.5rem;
    }
  }
  .reveal-if-active {
    opacity: 0;
    max-height: 0;
    overflow: hidden;
  }
  input[type="radio"]:checked ~ .reveal-if-active{
    opacity: 1;
    max-height: 100px; /* little bit of a magic number :( */
    overflow: visible;
  }

</style>



<!-- Custom styles for this template -->
<link href="form-validation.css" rel="stylesheet">
</head>
<body class="bg-light">

<div class="container">
<div class="py-5 text-center">
{% comment %} <img class="d-block mx-auto mb-4" src="https://getbootstrap.com/docs/4.6/assets/brand/bootstrap-solid.svg" alt="" width="72" height="72"> {% endcomment %}
<h2>Checkout form</h2>
{% comment %} <p class="lead">Below is an example form built entirely with Bootstrap's form controls. Each required form group has a validation state that can be
```

```
triggered by attempting to submit the form without completing it.</p> {%
endcomment %}
  </div>


  <div class="row">
    <div class="col-md-4 order-md-2 mb-4">
      <h4 class="d-flex justify-content-between align-items-center mb-3">
        <span class="text-muted">Your cart</span>
        <span class="badge badge-secondary badge-pill">3</span>
      </h4>
      {% for i in prolist %}
      <ul class="list-group mb-3">
        <li class="list-group-item d-flex justify-content-between lh-condensed">
          <div>
            <div>
              <img src="{{i.proimg.url}}" alt="" style="height: 25%; width: 25%;">
            </div>
            <h6 class="my-0">{{i.proname}}</h6>
            <small class="text-muted">Discrption:{{i.prodis}}</small>
            <div>
              <span class="text-muted">Price:{{i.proprice}}</span>
            </div>
            <span class="text-muted">Quantity:{{i.cartqty}}</span>
            <span class="text-muted">price of your qty:{{i.prototalprice}}</span>


          </div>
        </li>
        {% endfor %}
        <li class="list-group-item d-flex justify-content-between">
```

```
        <span>Total </span>
        <strong>{{grandtotal}}</strong>
      </li>


    </ul>



    <form class="card p-2">
      <div class="input-group">
        {% comment %} <input type="text" class="form-control" placeholder="Promo
code">
        <div class="input-group-append">
          <button type="submit" class="btn btn-secondary">Redeem</button>
        </div> {% endcomment %}
      </div>
    </form>
  </div>
  <div class="col-md-8 order-md-1">
    <h4 class="mb-3">Billing address</h4>
    <form class="needs-validation" novalidate method="post">
      {% csrf_token %}
      <div class="row">
        <div class="col-md-6 mb-3">
          <label for="firstName">First name</label>
          <input type="text" class="form-control" id="firstName" placeholder=""
value="{{session.name}}" required name="name">
          <div class="invalid-feedback">
            Valid first name is required.
          </div>
```

```
        </div>
      </div>


      <div class="mb-3">
        <label for="email">Email <span class="text-muted"></span></label>
        <input type="email" class="form-control" id="email"
placeholder="you@example.com" name='email' required value={{session.email}}
readonly>
        <div class="invalid-feedback">
          Please enter a valid email address for shipping updates.
        </div>
      </div>


      <div class="mb-3">
        <label for="email">Mobile Number <span class="text-muted"></span></label>
        <input type="text" class="form-control" id="email" placeholder="+91 "
name='mob' required value={{session.mob}} >
        <div class="invalid-feedback">
          Please enter a valid email address for shipping updates.
        </div>
      </div>


      <div class="mb-3">
        <label for="address">Address</label>
        <input type="text" class="form-control" id="address" placeholder="1234 Main St"
required name='add' value="{{session.add}}" >
        <div class="invalid-feedback">
          Please enter your shipping address.
        </div>
      </div>
```

```
<div class="mb-3">
  <label for="address2">City  <span class="text-muted"></span></label>
  <input type="text" class="form-control" id="address2" placeholder="city" name='city' required value="">
</div>


<div class="row">


  <div class="col-md-4 mb-3">
    <label for="state">State</label>
    <input type="text" class="form-control" id="address2" placeholder="state" name='state' required value="{{a.state}}">
  </div>
  <div class="col-md-3 mb-3">
    <label for="zip">Zip</label>
    <input type="text" class="form-control" id="zip" placeholder="Pincode" required name='pin' value="{{a.pincode}}">
    <div class="invalid-feedback">
      Zip code required.
    </div>
  </div>
</div>
<hr class="mb-4">
<div class="custom-control custom-checkbox">
  <input type="checkbox" class="custom-control-input" id="same-address">
  <label class="custom-control-label" for="same-address">Shipping address is the same as my billing address</label>
</div>
<div class="custom-control custom-checkbox">
```

```
        <input type="checkbox" class="custom-control-input" id="save-info">
        <label class="custom-control-label" for="save-info">Save this information for next time</label>
      </div>
      <hr class="mb-4">


      <h4 class="mb-3">Payment</h4>


      <div class="d-block my-3">
        <div class="custom-control custom-radio">
          <input id="online" name="paymentvia"  type="radio" class="custom-control-input" value="online">
          <label class="custom-control-label" for="online">Online</label>
        </div>
        <div class="custom-control custom-radio">
          <input id="cod" name="paymentvia"  type="radio" class="custom-control-input" value="cod">
          <label class="custom-control-label" for="cod">Cash on Delivery</label>
        </div>
      </div>
        <hr class="mb-4">
        <button class="btn btn-primary btn-lg btn-block" type="submit">Continue to checkout</button>
        {% comment %} <button class="btn btn-primary btn-lg btn-block" type="submit" id="pay-btn">Make Payment</button> {% endcomment %}
     </form>
   </div>
  </div>


  <footer class="my-5 pt-5 text-muted text-center text-small">
```

```
    <p class="mb-1">&copy; 2017-2022 Company Name</p>
    <ul class="list-inline">
      <li class="list-inline-item"><a href="#">Privacy</a></li>
      <li class="list-inline-item"><a href="#">Terms</a></li>
      <li class="list-inline-item"><a href="#">Support</a></li>
    </ul>
  </footer>
</div>



    <script src="https://cdn.jsdelivr.net/npm/jquery@3.5.1/dist/jquery.slim.min.js"
integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUe
w+OrCXaRkfj" crossorigin="anonymous"></script>
      <script>window.jQuery || document.write('<script
src="/docs/4.6/assets/js/vendor/jquery.slim.min.js"><\/script>')</script><script
src="/docs/4.6/dist/js/bootstrap.bundle.min.js"
integrity="sha384-Fy6S3B9q64WdZWQUiU+q4/2Lc9npb8tCaSX9FK7E8HnRr0Jz8D6O
P9dO5Vg3Q9ct" crossorigin="anonymous"></script>


        </body>
</html>
{% endblock abc %}
```

Day 23

Product Template

1. product.html

```
{% extends "nav.html" %}
{% block abc %}
{% load static %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>E-commerce</title>
    <link
      rel="stylesheet"
      45
      href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    />
    <link
      rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css"

integrity="sha512-z3gLpd7yknf1YoNbCzqRKc4qyor8gaKU1qmn+CShxbuBusANI9QpRohGBreCFkKxLhei6S9CQXFEbbKuqLg0DA=="
      crossorigin="anonymous"
      referrerpolicy="no-referrer"
    />
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
```

```
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href={% static "css/ecom.css" %} />
 </head>
 <body>
  <!-------------------------- Navbar ------------------------->



  <!-------------------------- Main Section ------------------------->
  <div class="container my-5">
   <div class="row mt-4 mr-auto ml-auto">
    <div class="col-lg-4">

     <img
      src={{prodetails.image.url}}
      alt="Product Image"
      class="img-fluid"
      height="500px"
      width="300px"
     />

    </div>
    <div class="col-lg-6 mr-auto ml-auto mt-5">
     <h1>{{prodetails.name}}</h1>
     <p>
      {{prodetails.discrption}}
     </p>
     <p><strong>Price:</strong> ₹{{prodetails.price}}</p>
     <form action="" method="post">
```

```
      {% csrf_token %}
      <input type="number" minlength="1" name="qty">
      <i class="fa-solid fa-star ratingcolor"></i>
      <i class="fa-solid fa-star ratingcolor"></i>
      <i class="fa-solid fa-star ratingcolor"></i>
      <i class="fa-regular fa-star-half-stroke ratingcolor"></i>
      <i class="fa-regular fa-star ratingcolor"></i>
      <br />
      <a href="cart.html">
      <button class="btn btn-outline-primary mt-3">Add to Cart</button>
    </a>
  </form>
  {% if chej %}
  <script>
    alert('{{chej}}')
  </script>
  {% endif %}
    </div>
  </div>
</div>

<!-----------------------Footer Section----------------------->
<footer class="bg-dark text-light py-4">
  <div class="container">
    <div class="row">
      <div class="col-md-4">
        <h5>Contact Us</h5>
        <p>Email: info@example.com</p>
```

```
      <p>Phone: 123-456-7890</p>
    </div>
    <div class="col-md-4">
      <h5>Links</h5>
      <ul class="list-unstyled">
        <li><a href="#">Home</a></li>
        <li><a href="#">Products</a></li>
        <li><a href="#">About Us</a></li>
        <li><a href="#">Contact Us</a></li>
      </ul>
    </div>
    <div class="col-md-4">
      <h5>Follow Us</h5>
      <ul class="list-inline">
        <li class="list-inline-item">
          <a href="#"><i class="fab fa-facebook"></i></a>
        </li>
        <li class="list-inline-item">
          <a href="#"><i class="fab fa-twitter"></i></a>
        </li>
        <li class="list-inline-item">
          <a href="#"><i class="fab fa-instagram"></i></a>
        </li>
      </ul>
    </div>
  </div>
  <hr />
  <div class="row">
```

```
        <div class="col text-center">
          <p>&copy; 2023 Your E-Commerce Store</p>
        </div>
      </div>
    </div>
  </footer>
 </body>
</html>
{% endblock abc %}
```

Day 24

“Deployment”

Prompt: Website Deployment on Server

Deploying a Django project typically involves the following steps:

1. **Prepare your Django project for deployment:**
   - Ensure your project is properly configured for production, including settings related to database, static files, media files, security, and debugging.
   - Update your `ALLOWED_HOSTS` setting in `settings.py` to include the domain names or IP addresses of your production server(s).

2. **Choose a hosting provider:**
   - Select a hosting provider that supports Django applications. Popular options include Heroku, DigitalOcean, AWS (Amazon Web Services), Google Cloud Platform, PythonAnywhere, and others.
   - Consider factors such as pricing, scalability, performance, ease of use, and support when choosing a hosting provider.

3. **Set up your server environment:**
   - Provision a server instance (virtual private server or VPS) with your hosting provider.
   - Install necessary software on the server, such as Python, a web server (e.g., Nginx or Apache), and a database server (e.g., PostgreSQL, MySQL, or SQLite).
   - Set up a domain name and configure DNS settings to point to your server's IP address.

4. **Deploy your code to the server:**
   - Upload your Django project files to the server using SSH, FTP, or a version control system like Git.

- Set up a virtual environment on the server and install project dependencies using pip.
- Configure the server to serve static files and media files (if applicable) using the appropriate web server configuration.

5. **Configure the web server:**
   - Set up Nginx or Apache to serve as a reverse proxy for your Django application.
   - Configure the web server to pass requests to the Django application using WSGI (Web Server Gateway Interface) or ASGI (Asynchronous Server Gateway Interface).

6. **Set up database and environment variables:**
   - Create a database for your Django project and configure the database connection settings in your `settings.py` file.
   - Set up environment variables for sensitive information such as database credentials, secret keys, and API keys.

7. **Collect static files:**
   - Run the `collectstatic` management command to collect static files from your Django apps into one location.
   - Configure your web server to serve static files directly or use a content delivery network (CDN) for improved performance.

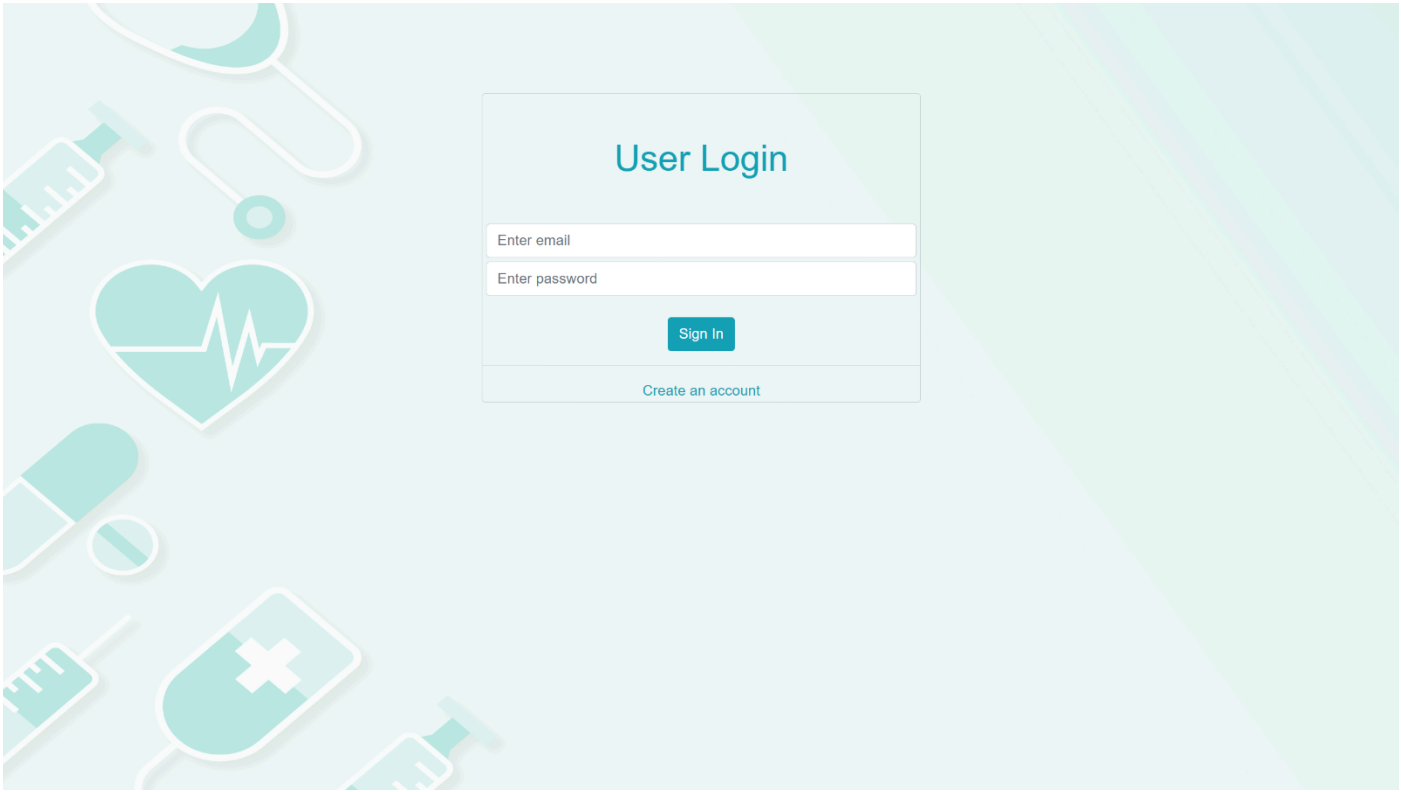
8. **Test your deployment:**
   - Access your Django application through the domain name or IP address of your server.
   - Test all functionality to ensure everything is working correctly in the production environment.
   - Monitor server logs and error messages for any issues that may arise.

9. **Implement security measures:**
   - Enable HTTPS to encrypt traffic between the client and server.
   - Implement security best practices, such as using strong passwords, keeping software up-to-date, and configuring firewalls.

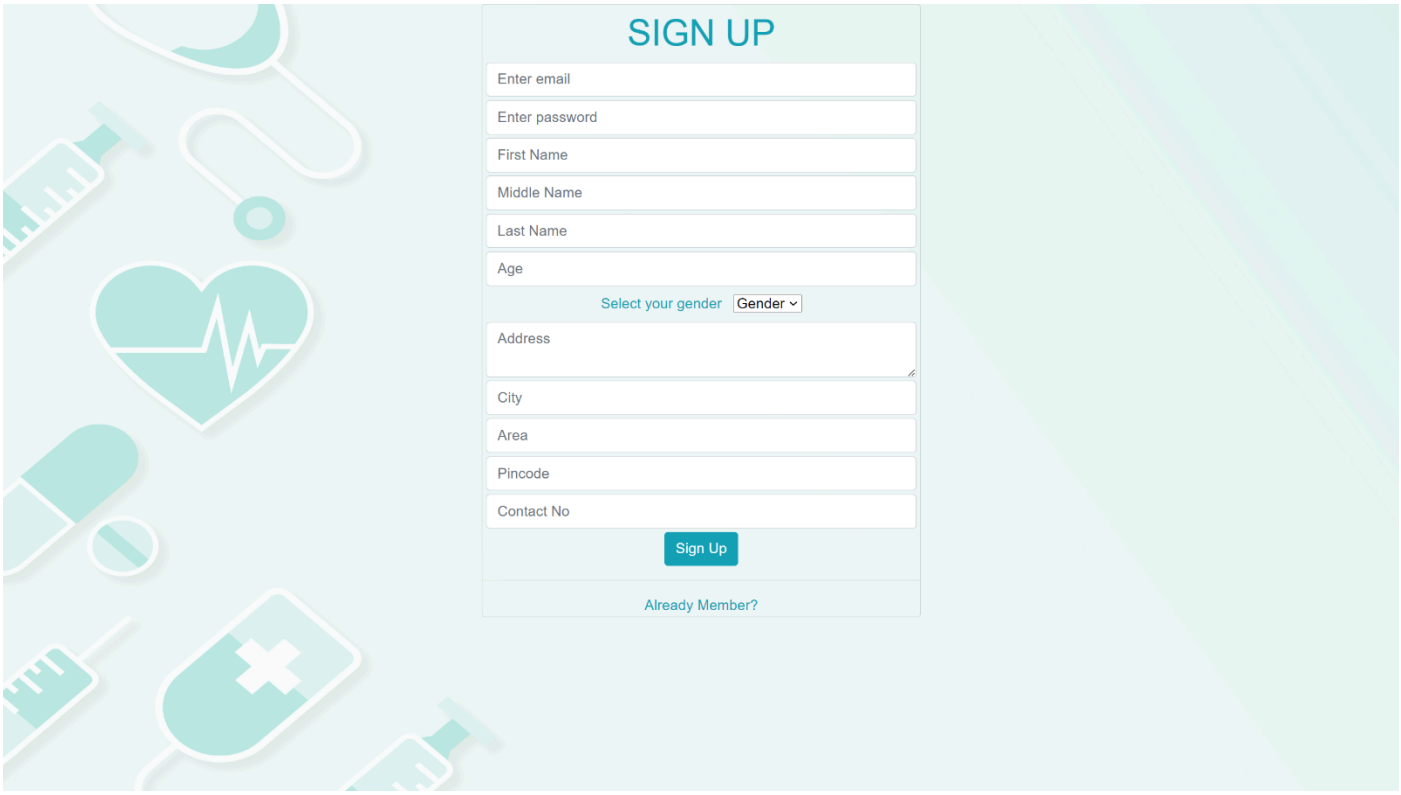
10. **Set up backups and monitoring:**
    - Configure regular backups of your Django project files, database, and server configuration.
    - Set up monitoring tools to track server performance, uptime, and security threats.

Remember to refer to the documentation of your hosting provider and follow their specific guidelines for deploying Django applications. Additionally, consider automating the deployment process using tools like Fabric, Ansible, or Docker for easier management and scalability.

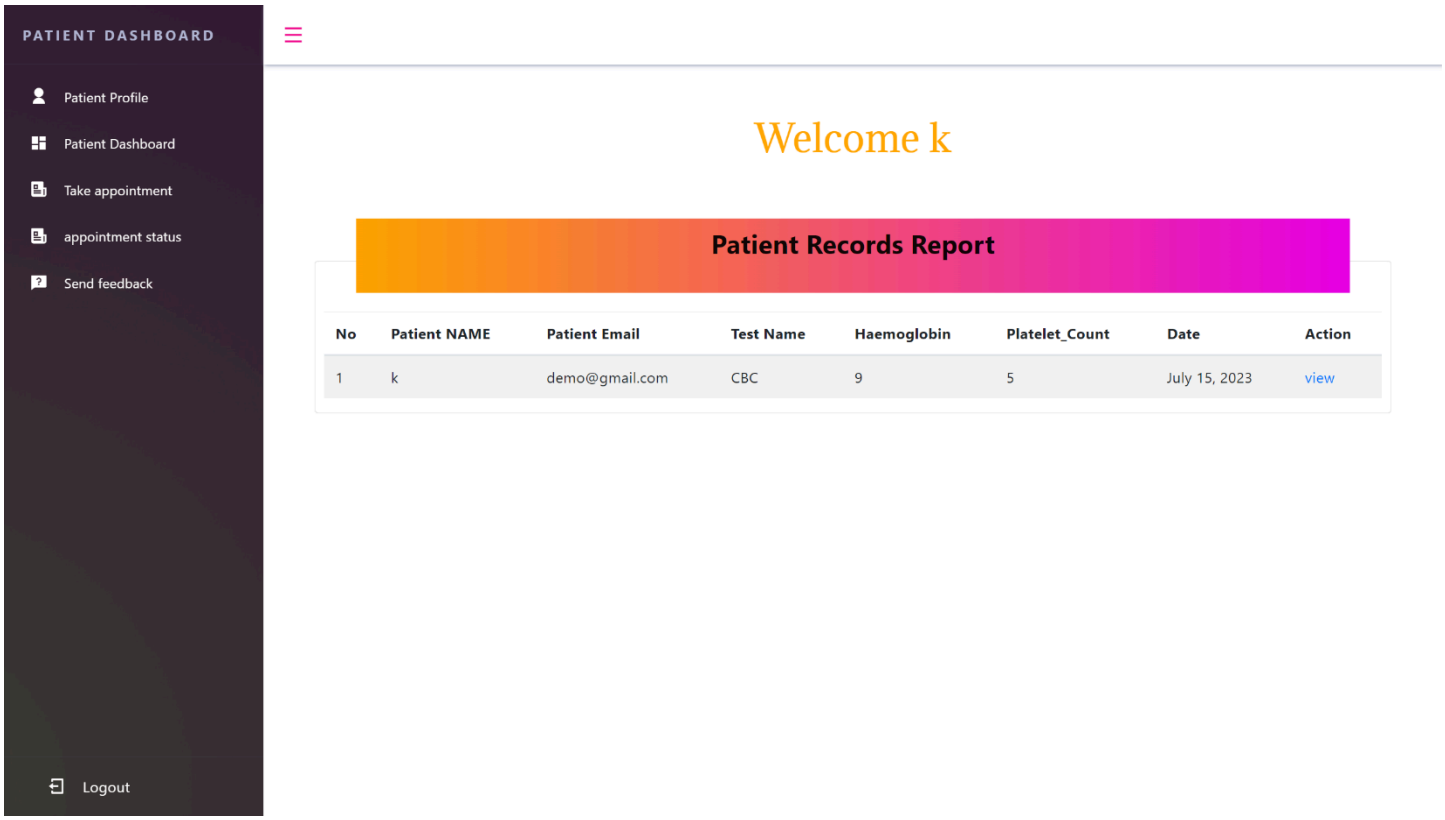
Day 25


User Login
Enter email
Enter password
Sign In
Create an account

Day 26


SIGN UP
Enter email
Enter password
First Name
Middle Name
Last Name
Age
Select your gender
Gender
Address
City
Area
Pincode
Contact No
Sign Up
Already Member?

Day 27


PATIENT DASHBOARD
Patient Profile
Patient Dashboard
Take appointment
appointment status
Send feedback
Logout
Welcome k
Patient Records Report
No
Patient NAME
Patient Email
Test Name
Haemoglobin
Platelet_Count
Date
Action
1
k
demo@gmail.com
CBC
9
5
July 15, 2023
view

Day 28

PATIENT NAME:	k	DATE:	July 15, 2023
REF BY DR.	DR. soury M.B.B.S.	SEX:	Male
SMPLE COLL AT:	dev lab	AGE	18 Years

COMPLETE BLOOD COUNT

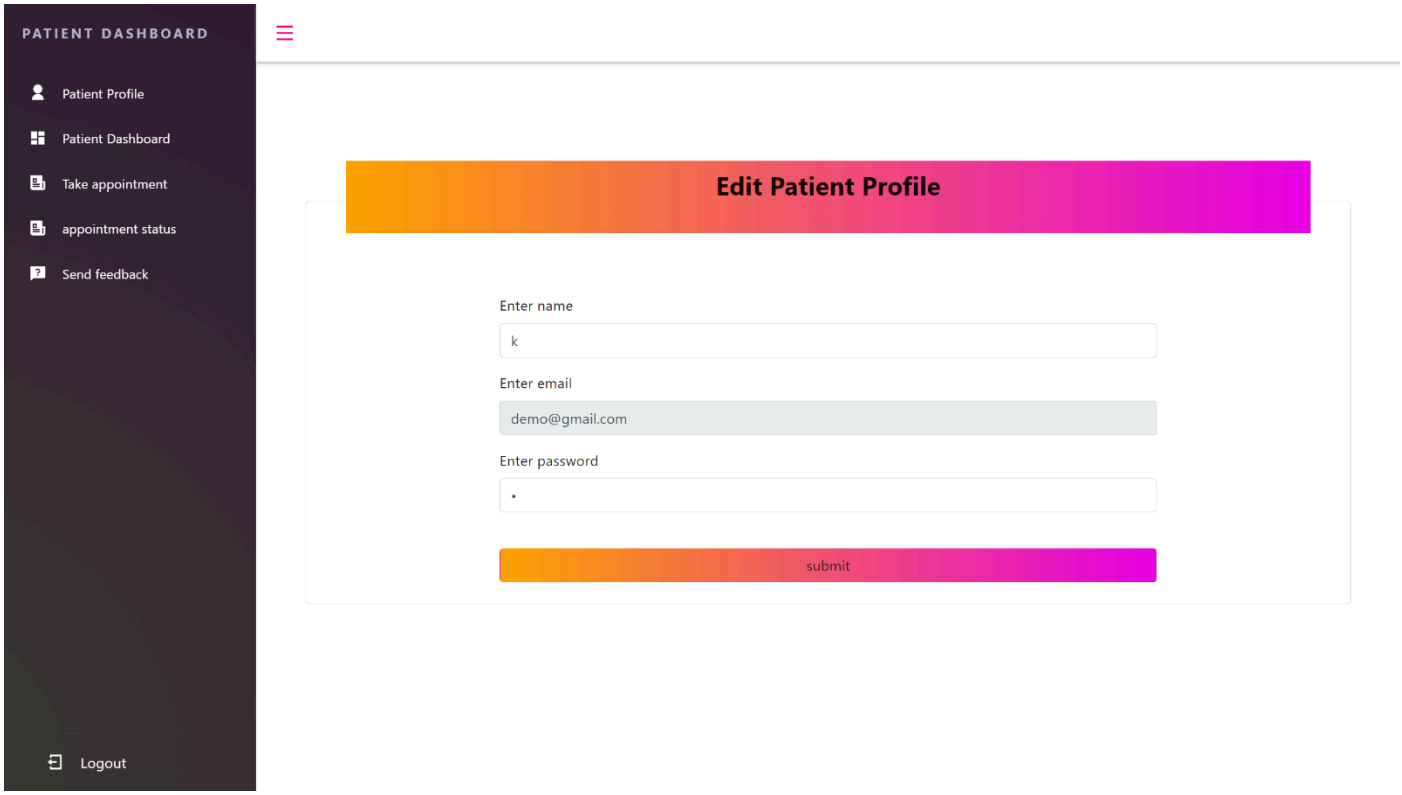
Test	Result	Reference Range
Haemoglobin	9	Male: 14-16 g%
		Female: 12-14 g%
RBC Count	5	14-16 g%
PCV	5	35-45%
RBC INDICES		
MCV	5	80-99 fl
MCH	5	28-32 pg
MCHC	5	30-34%
RDW	5	9-17 fl
TOTAL WBC COUNT		
Total WBC count	5	4000-11000 /cu.mm
Neutrophils	5	40-75%

Day 29

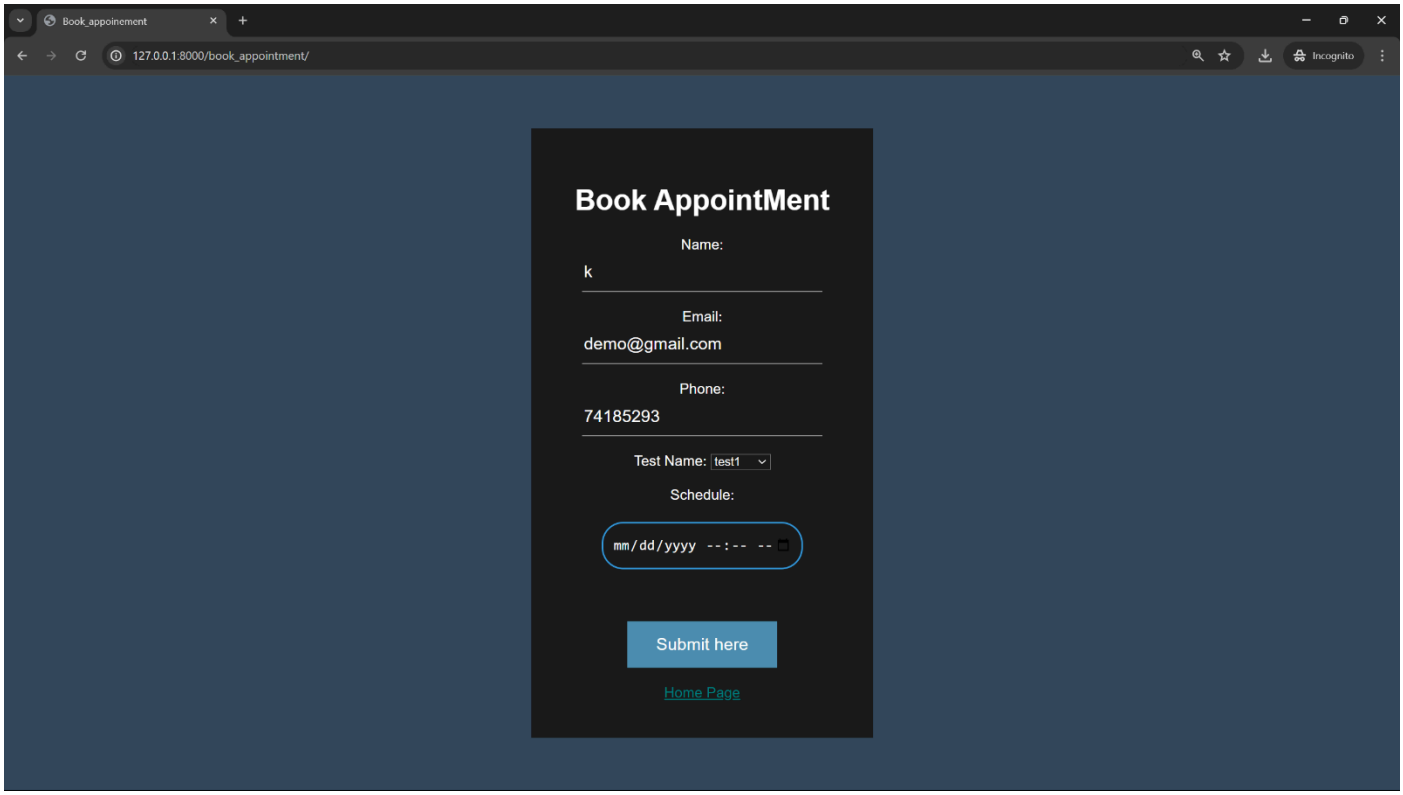
RBC INDICES		
MCV	5	80-99 fl
MCH	5	28-32 pg
MCHC	5	30-34%
RDW	5	9-17 fl
TOTAL WBC COUNT		
Total WBC count	5	4000-11000 /cu.mm
Neutrophils	5	40-75%
Lymphocytes	5	20-45%
Eosinophils	5	00-06%
Monosytes	5	00-10%
Basophils	55	00-01%
PLATELETS		
Platelet Count	5	150000-450000/cu.mm
PERIPHERAL BLOOD SMEAR		
WBCs on PS	5	

---- End Of Report. ----

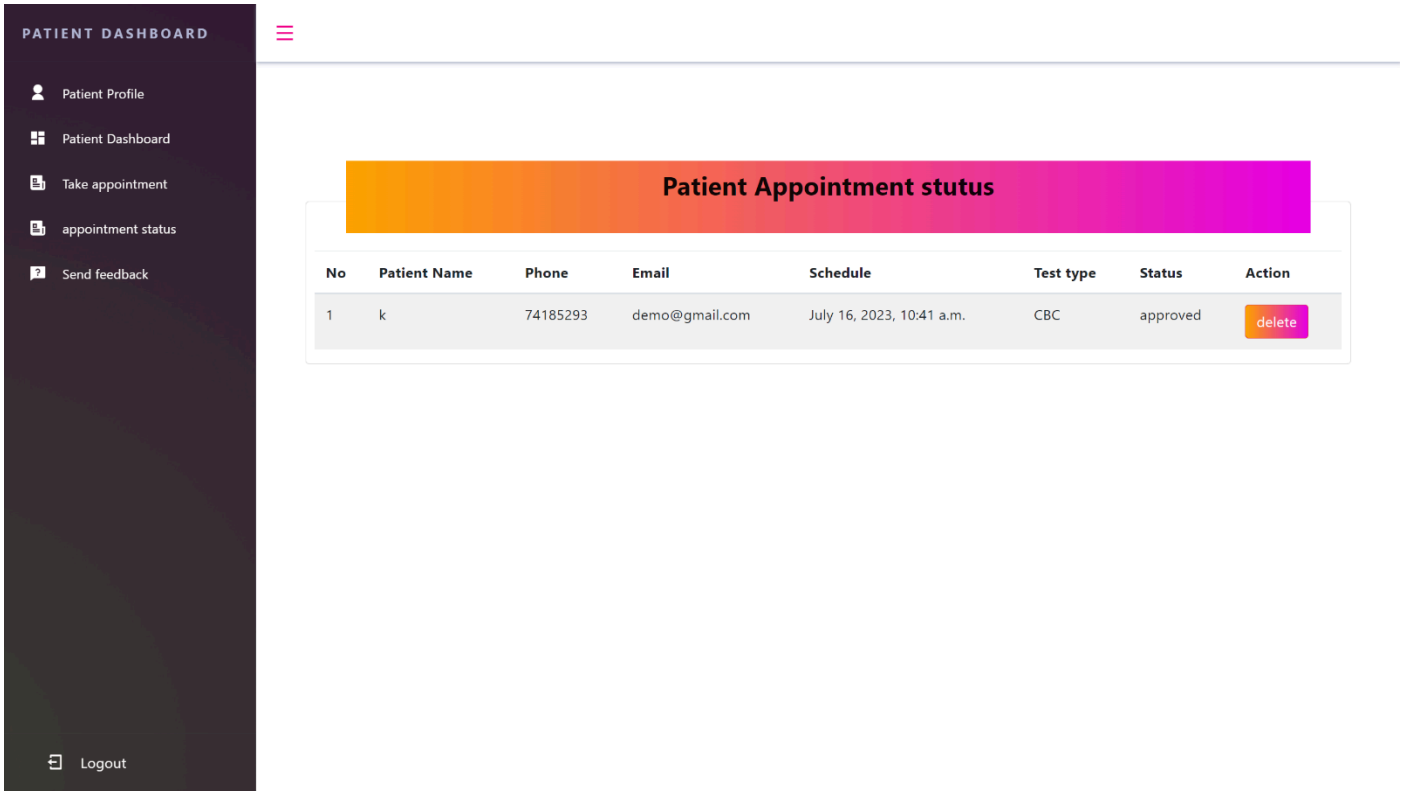
Day 30


PATIENT DASHBOARD
Patient Profile
Patient Dashboard
Take appointment
appointment status
Send feedback
Logout
Edit Patient Profile
Enter name
k
Enter email
demo@gmail.com
Enter password
submit

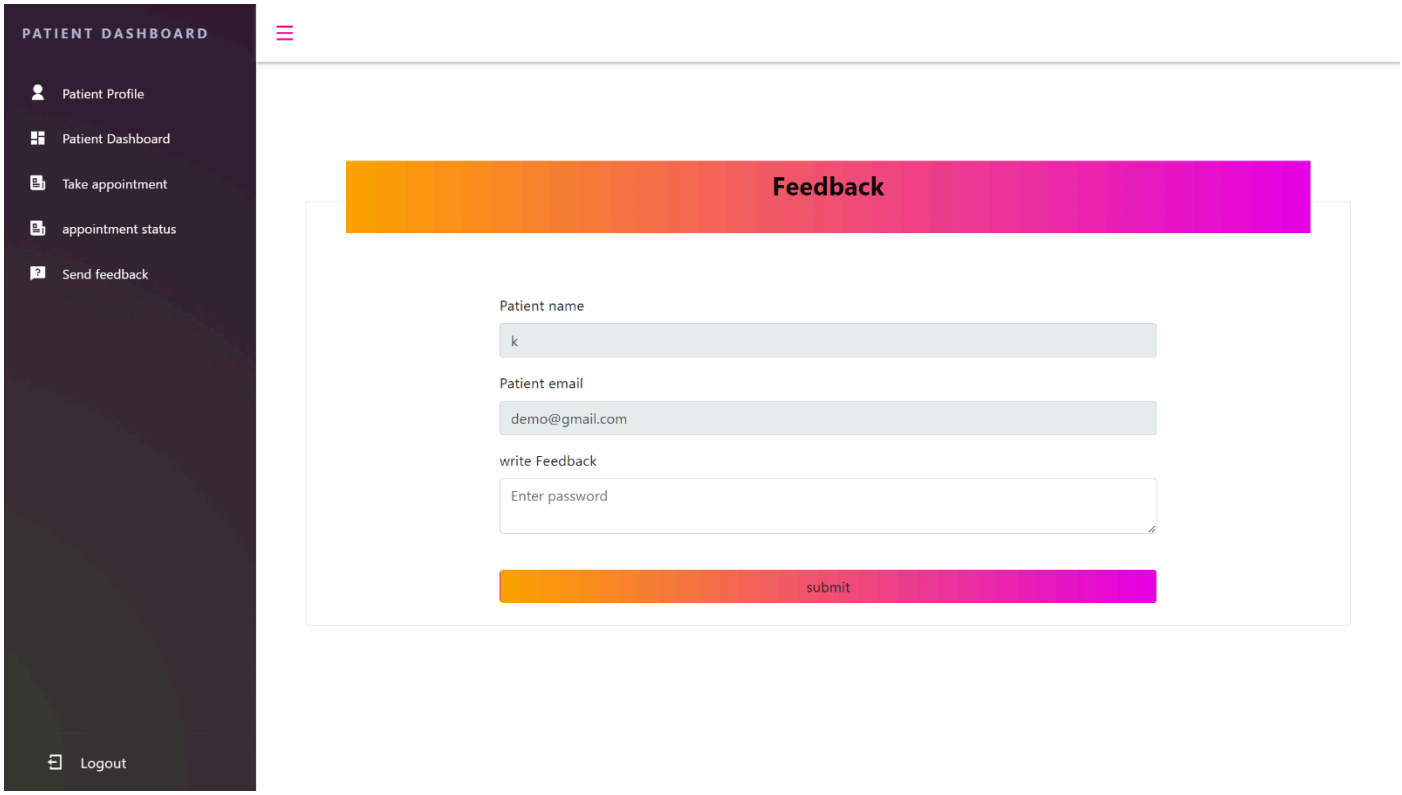
Day 31


Book_appoinement
127.0.0.1:8000/book_appointment/
Incognito
Book AppointMent
Name:
k
Email:
demo@gmail.com
Phone:
74185293
Test Name: test1
Schedule:
mm/dd/yyyy --:-- --
Submit here
Home Page

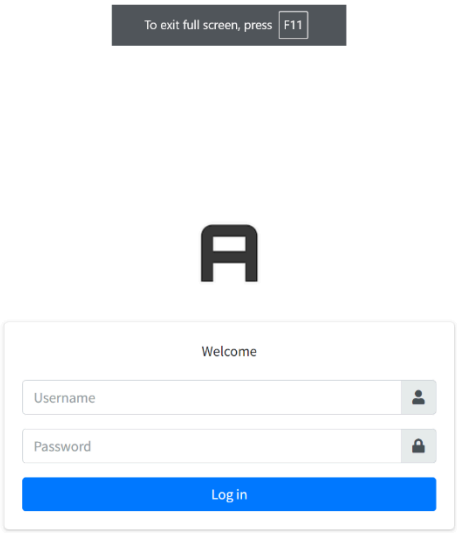
Day 32


PATIENT DASHBOARD
Patient Profile
Patient Dashboard
Take appointment
appointment status
Send feedback
Logout
Patient Appointment stutus
No
Patient Name
Phone
Email
Schedule
Test type
Status
Action
1
k
74185293
demo@gmail.com
July 16, 2023, 10:41 a.m.
CBC
approved
delete

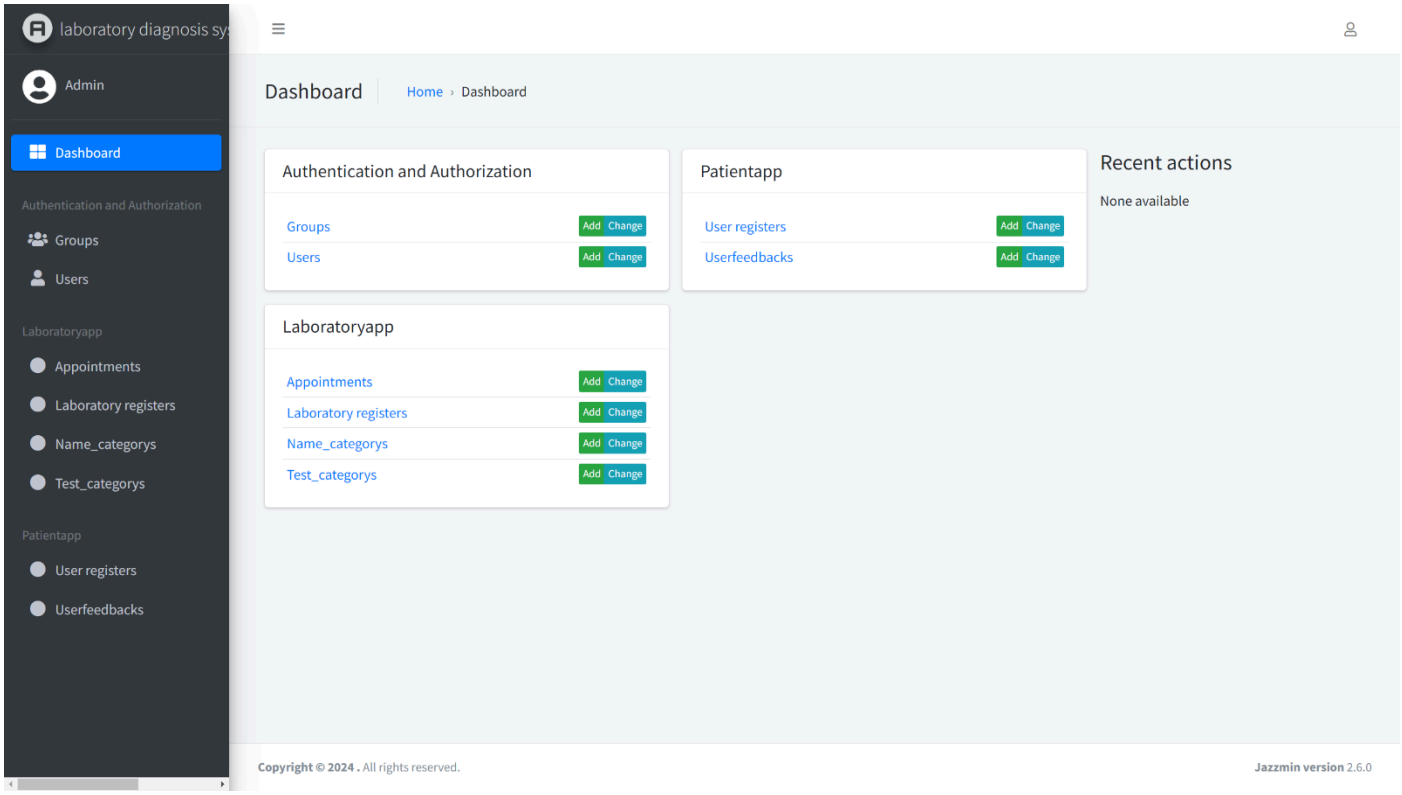
Day 33


PATIENT DASHBOARD
Patient Profile
Patient Dashboard
Take appointment
appointment status
Send feedback
Logout
Feedback
Patient name
k
Patient email
demo@gmail.com
write Feedback
Enter password
submit

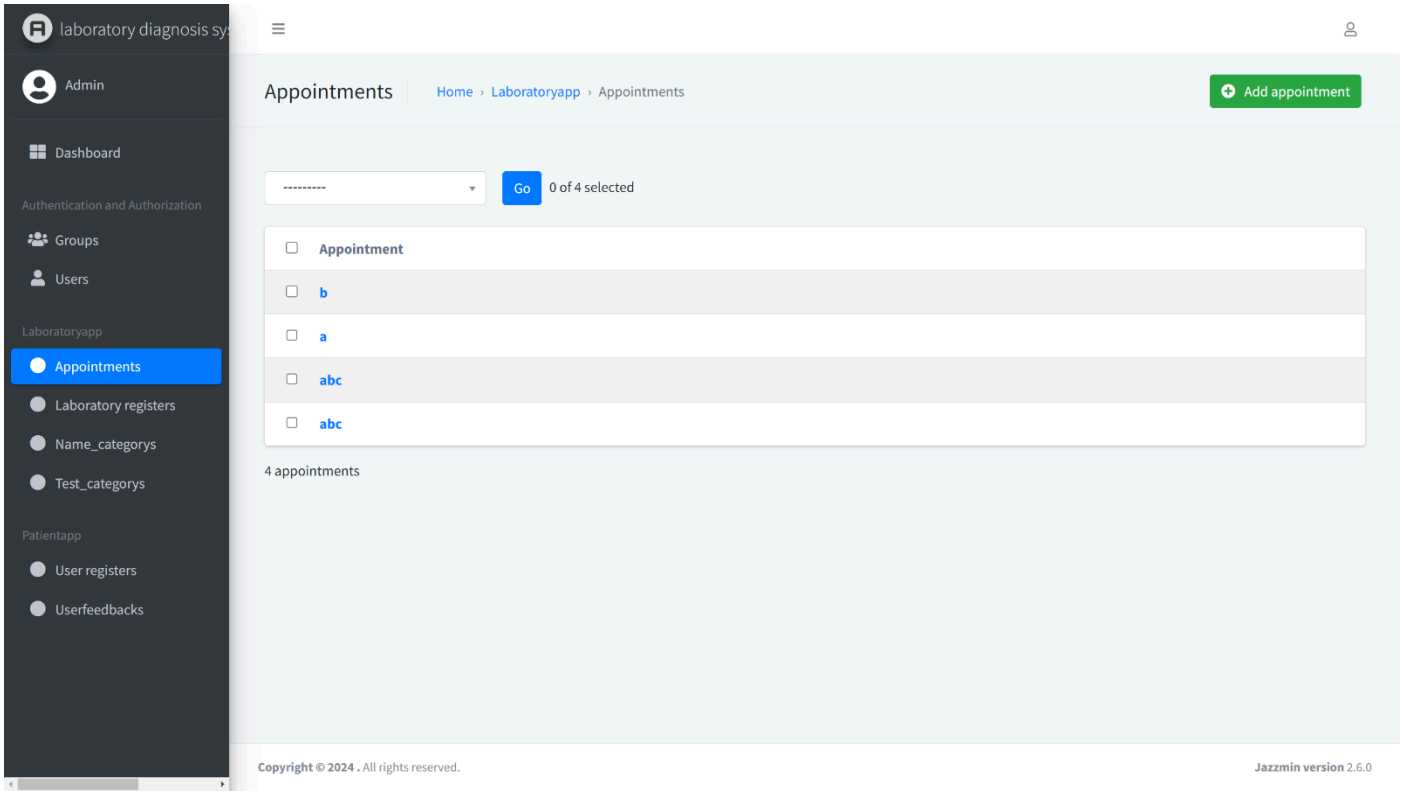
Day 34


To exit full screen, press F11
Welcome
Username
Password
Log in

Day 35

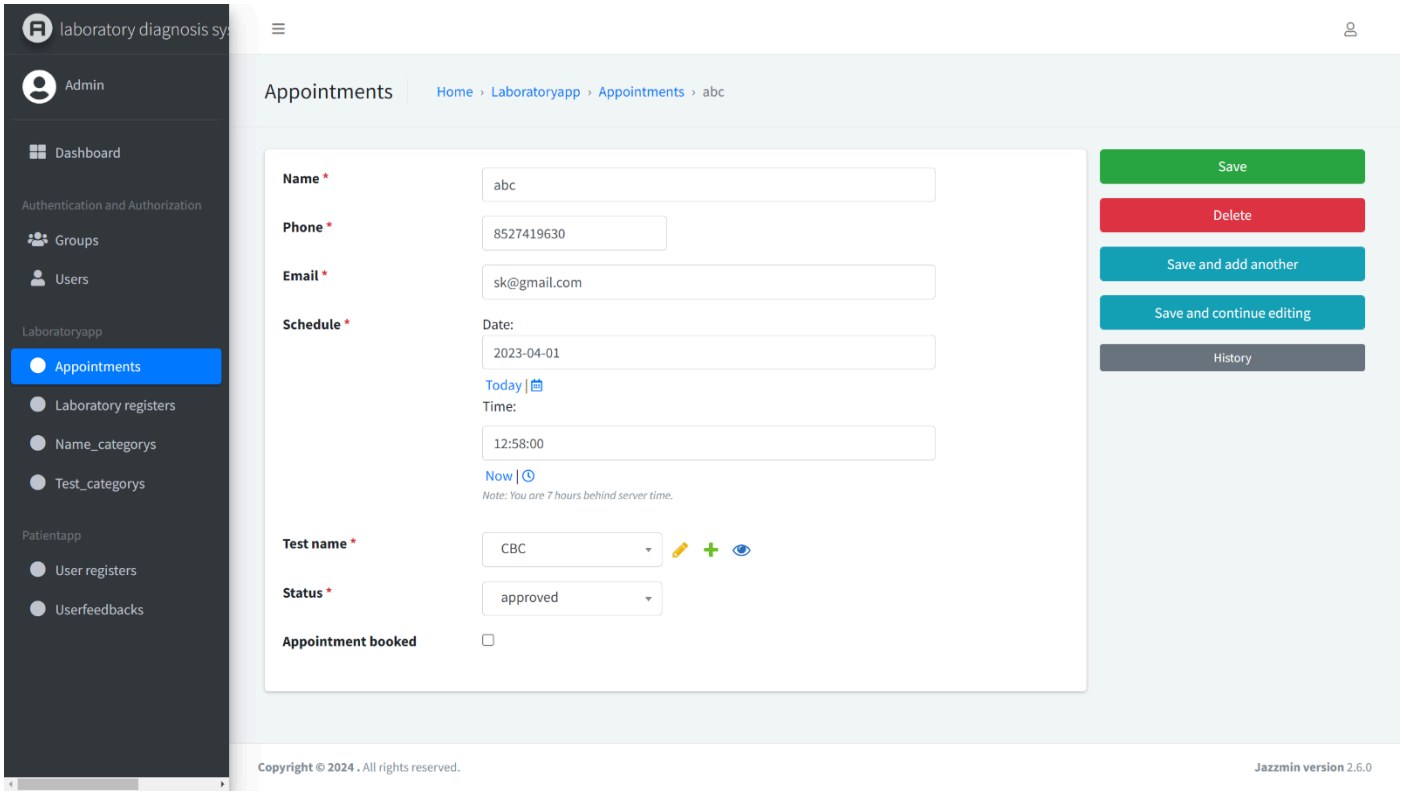

laboratory diagnosis sys
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Dashboard
Home › Dashboard
Authentication and Authorization
Groups Add Change
Users Add Change
Patientapp
User registers Add Change
Userfeedbacks Add Change
Laboratoryapp
Appointments Add Change
Laboratory registers Add Change
Name_categorys Add Change
Test_categorys Add Change
Recent actions
None available
Copyright © 2024 . All rights reserved.
Jazzmin version 2.6.0

Day 36

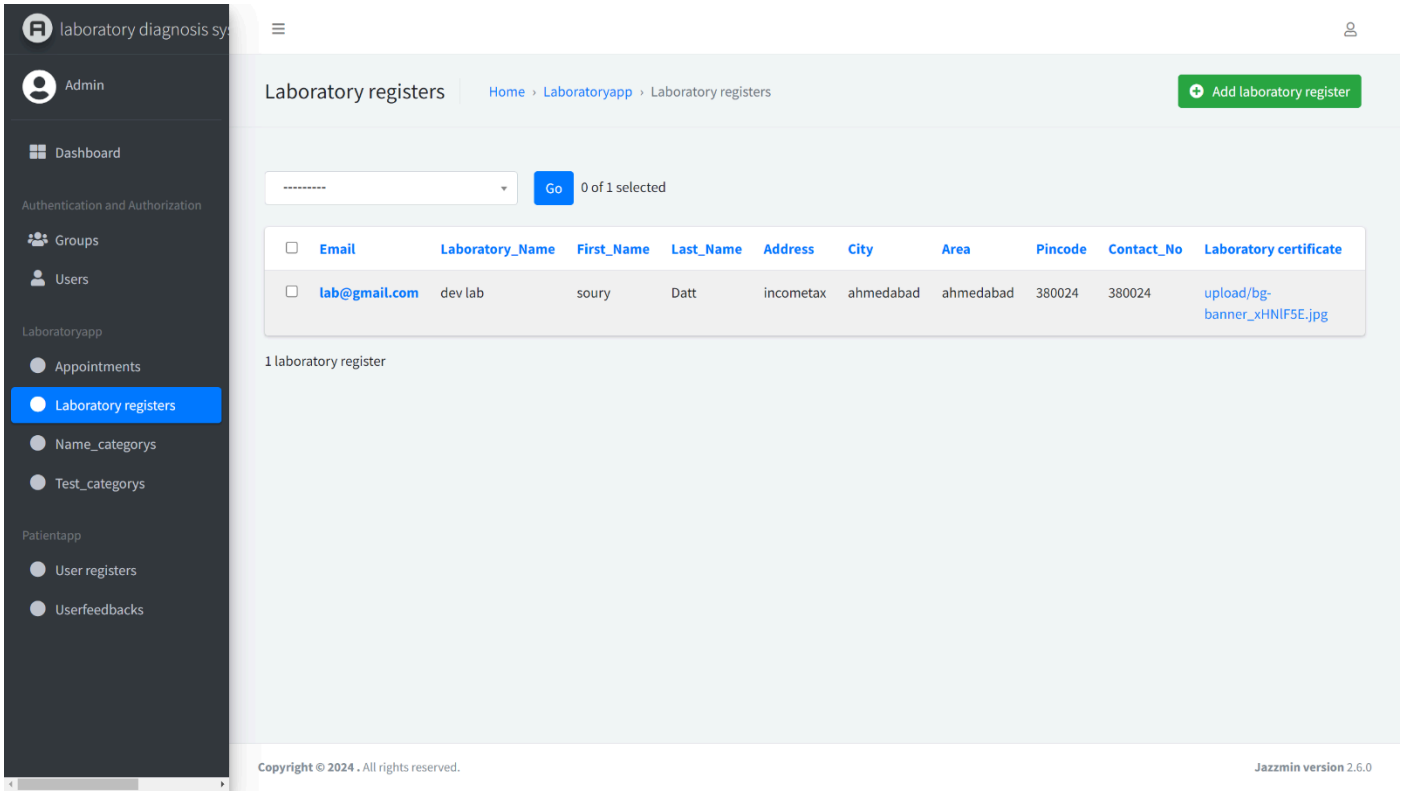

laboratory diagnosis sys
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Appointments
Home › Laboratoryapp › Appointments
Add appointment

Go
0 of 4 selected
Appointment
b
a
abc
abc
4 appointments
Copyright © 2024 . All rights reserved.
Jazzmin version 2.6.0

Day 37

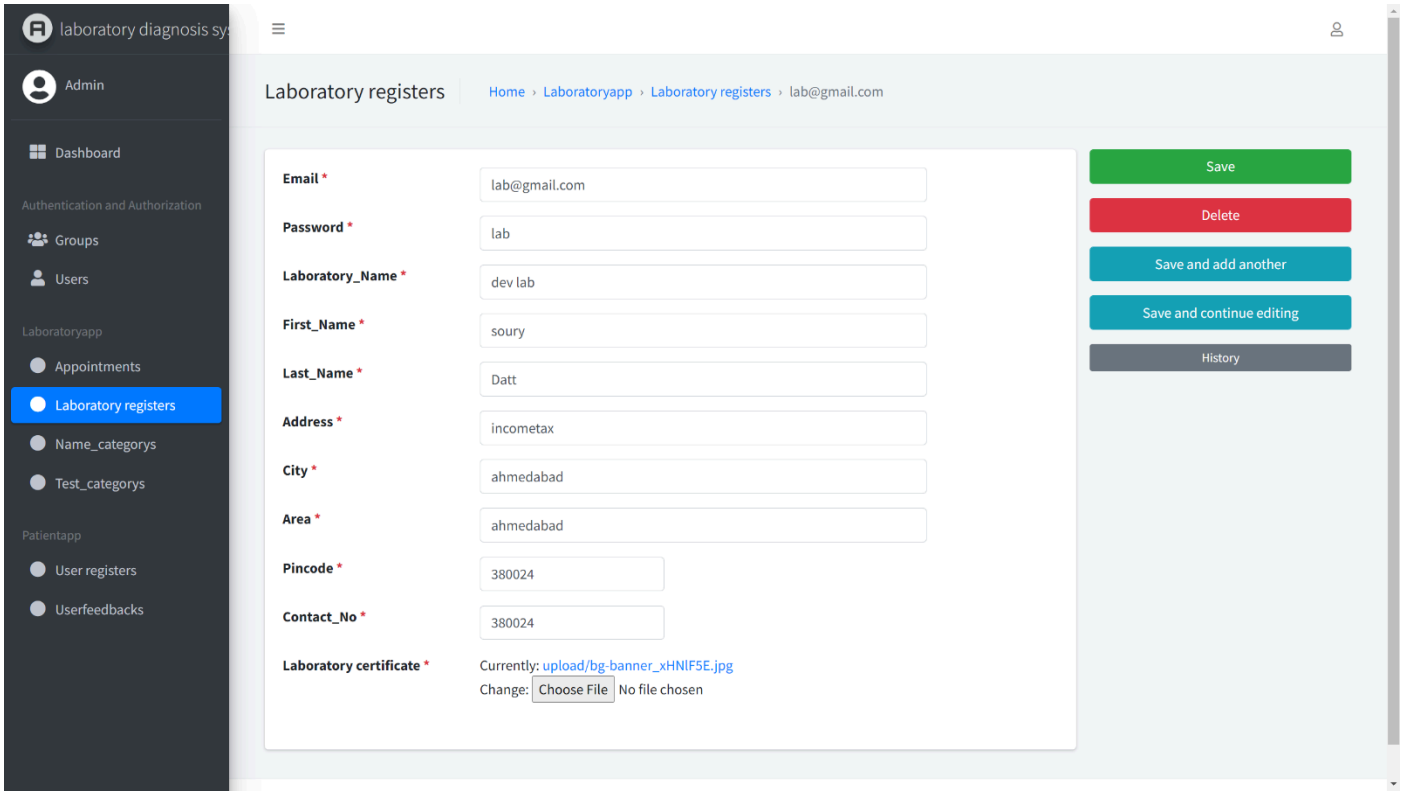

laboratory diagnosis sy
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Appointments
Home › Laboratoryapp › Appointments › abc
Name *
abc
Phone *
8527419630
Email *
sk@gmail.com
Schedule *
Date:
2023-04-01
Today |
Time:
12:58:00
Now |
Note: You are 7 hours behind server time.
Test name *
CBC
Status *
approved
Appointment booked
Save
Delete
Save and add another
Save and continue editing
History
Copyright © 2024 . All rights reserved.
Jazzmin version 2.6.0

Day 38

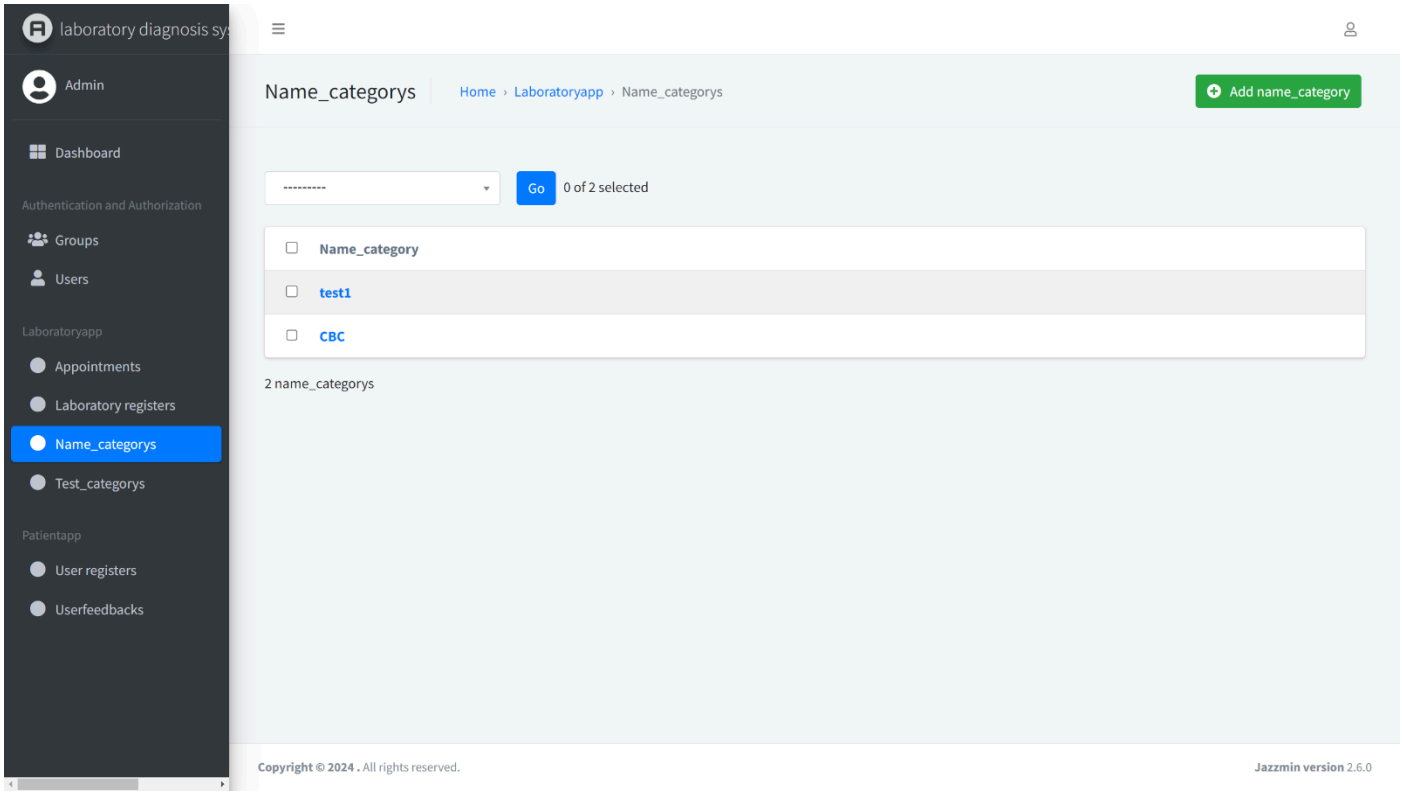

laboratory diagnosis sys
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Laboratory registers
Home › Laboratoryapp › Laboratory registers
Add laboratory register

Go
0 of 1 selected
Email
Laboratory_Name
First_Name
Last_Name
Address
City
Area
Pincode
Contact_No
Laboratory certificate
lab@gmail.com
dev lab
soury
Datt
incometax
ahmedabad
ahmedabad
380024
380024
upload/bg-banner_xHNlF5E.jpg
1 laboratory register
Copyright © 2024 . All rights reserved.
Jazzmin version 2.6.0

Day 39

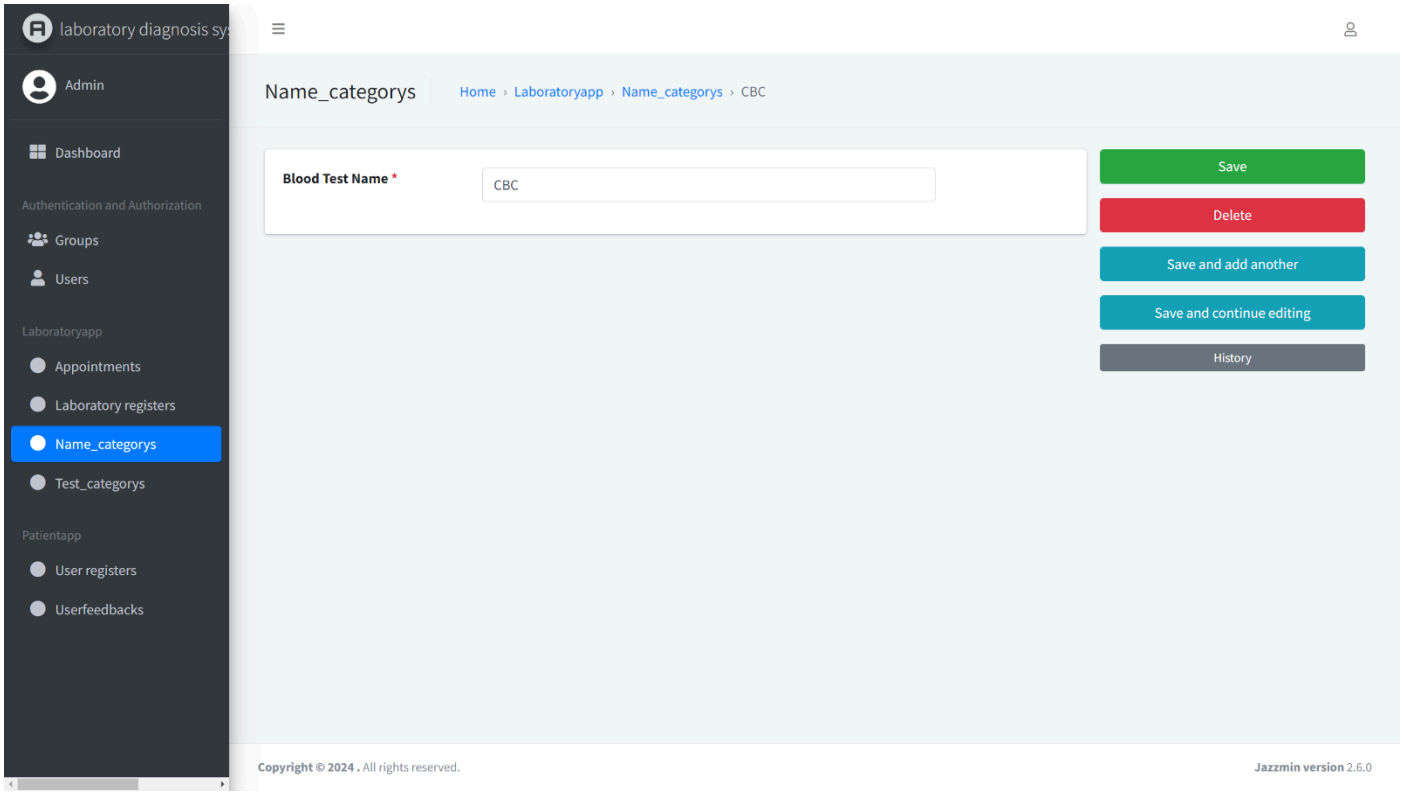

laboratory diagnosis sys
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Laboratory registers
Home › Laboratoryapp › Laboratory registers › lab@gmail.com
Email *
lab@gmail.com
Password *
lab
Laboratory_Name *
dev lab
First_Name *
soury
Last_Name *
Datt
Address *
incometax
City *
ahmedabad
Area *
ahmedabad
Pincode *
380024
Contact_No *
380024
Laboratory certificate *
Currently: upload/bg-banner_xHNlF5E.jpg
Change: Choose File No file chosen
Save
Delete
Save and add another
Save and continue editing
History

Day 40

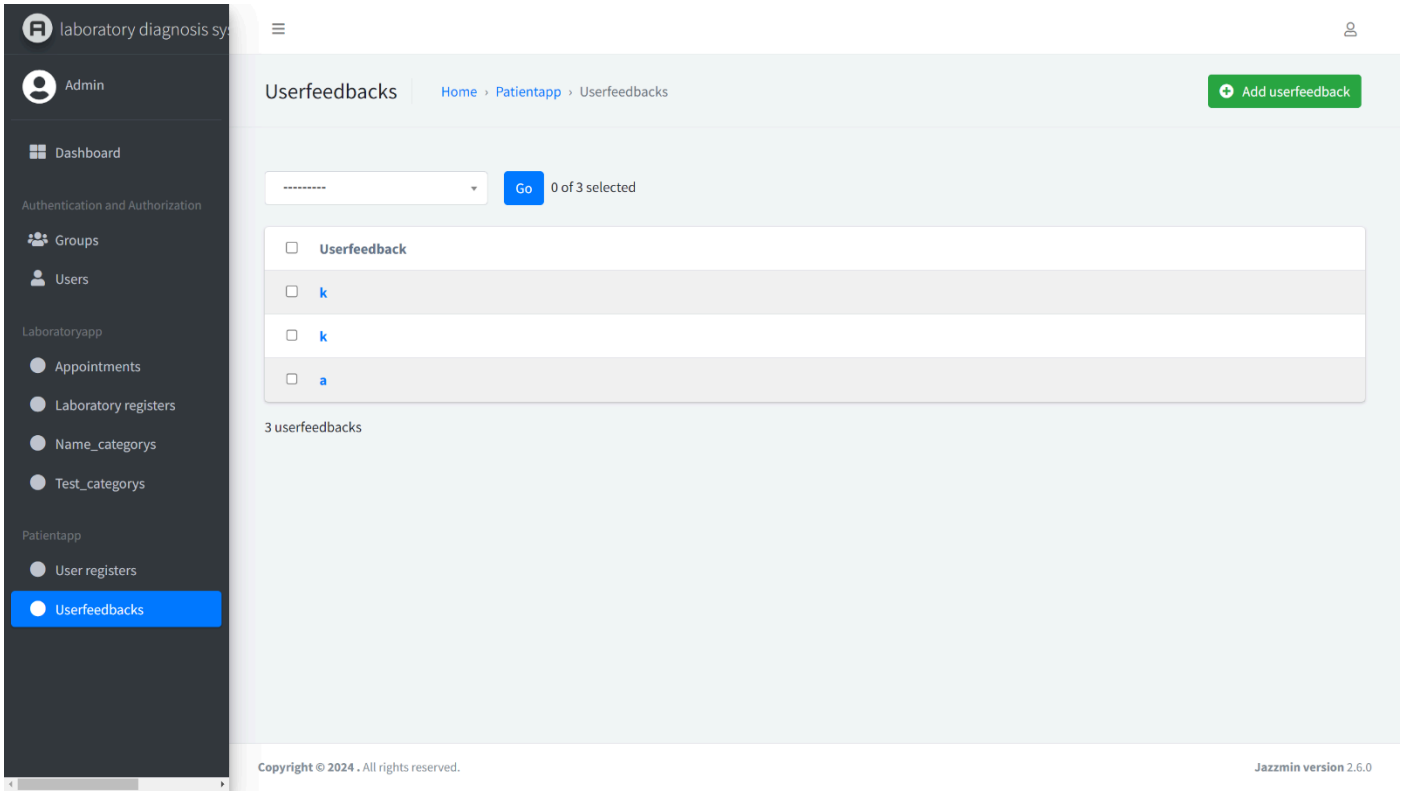

laboratory diagnosis sys
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Name_categorys
Home › Laboratoryapp › Name_categorys
Add name_category

Go
0 of 2 selected
Name_category
test1
CBC
2 name_categorys
Copyright © 2024 . All rights reserved.
Jazzmin version 2.6.0

Day 41

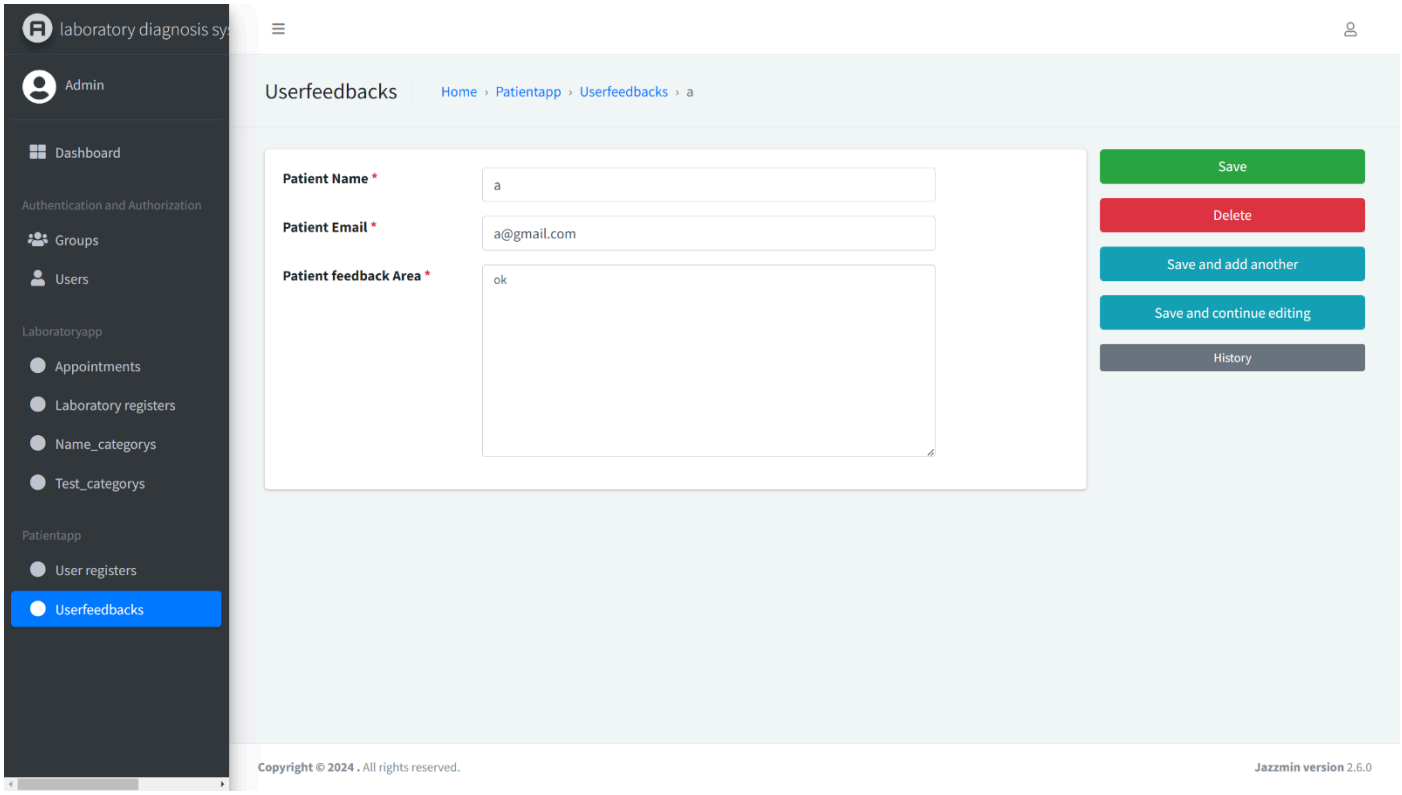

laboratory diagnosis sys
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Name_categorys
Home › Laboratoryapp › Name_categorys › CBC
Blood Test Name *
CBC
Save
Delete
Save and add another
Save and continue editing
History
Copyright © 2024 . All rights reserved.
Jazzmin version 2.6.0

Day 42

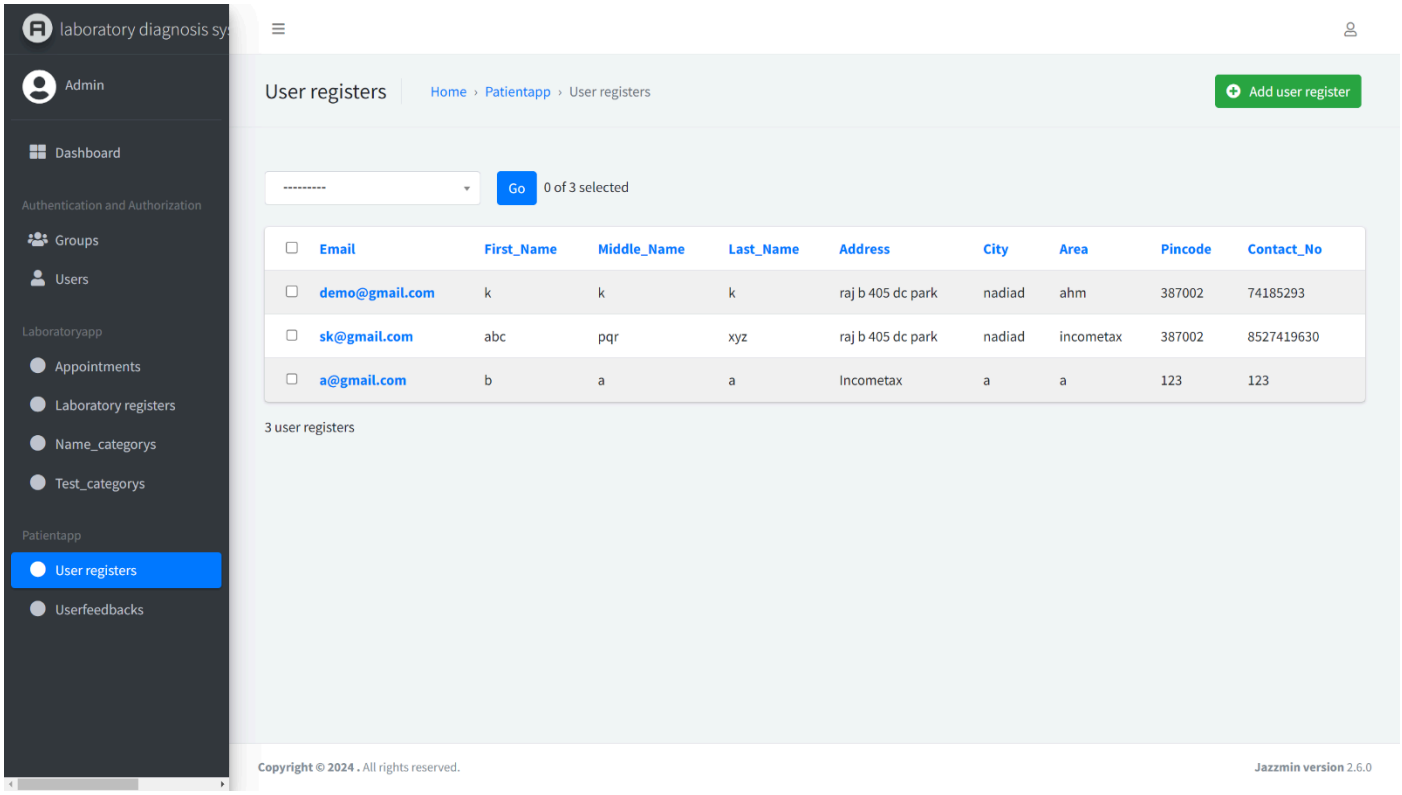

laboratory diagnosis sys
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Userfeedbacks
Home › Patientapp › Userfeedbacks
Add userfeedback

Go
0 of 3 selected
Userfeedback
k
k
a
3 userfeedbacks
Copyright © 2024 . All rights reserved.
Jazzmin version 2.6.0

Day 43

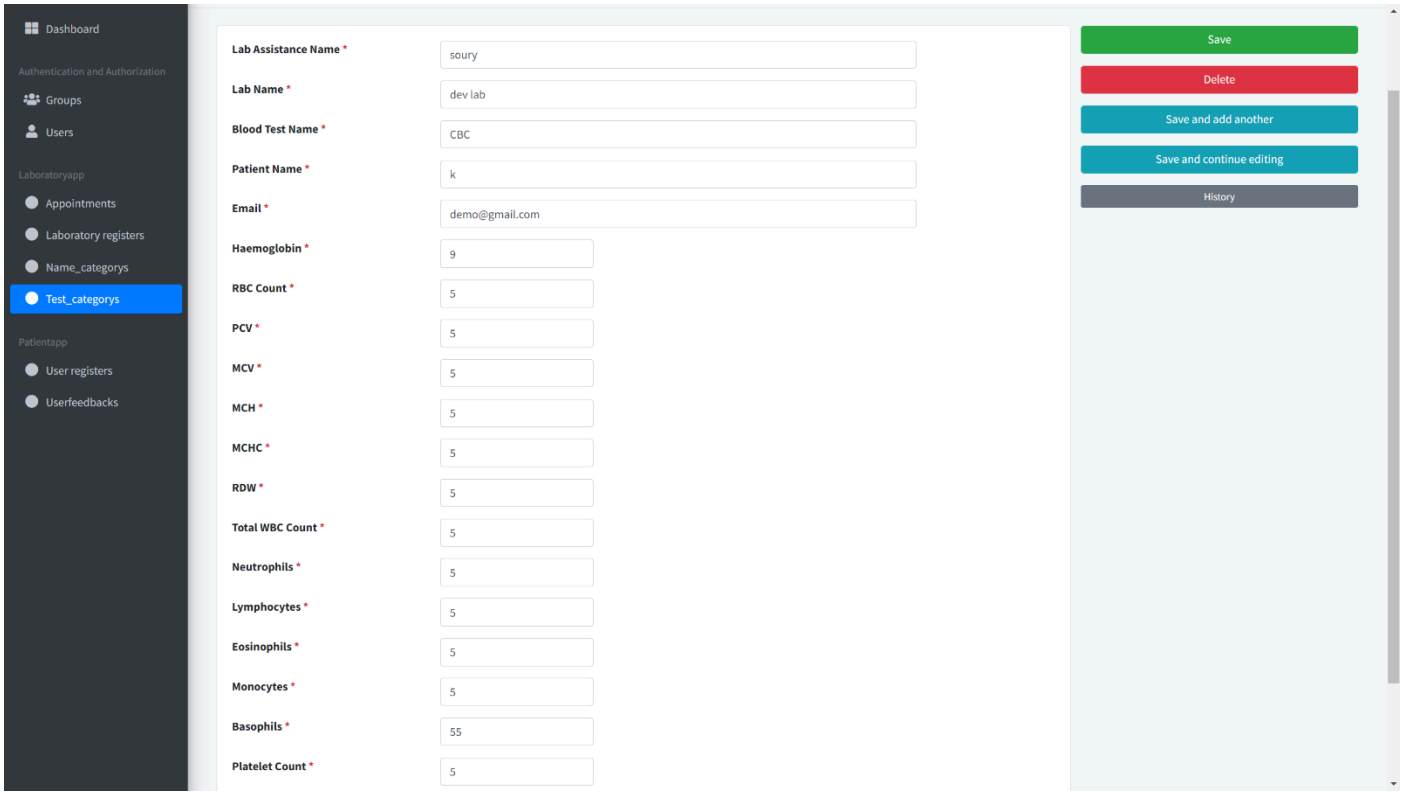

laboratory diagnosis sys
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Userfeedbacks
Home › Patientapp › Userfeedbacks › a
Patient Name *
a
Patient Email *
a@gmail.com
Patient feedback Area *
ok
Save
Delete
Save and add another
Save and continue editing
History
Copyright © 2024 . All rights reserved.
Jazzmin version 2.6.0

Day 44


laboratory diagnosis sys
Admin
Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
User registers
Home › Patientapp › User registers
Add user register

Go
0 of 3 selected
Email | First_Name | Middle_Name | Last_Name | Address | City | Area | Pincode | Contact_No
demo@gmail.com | k | k | k | raj b 405 dc park | nadiad | ahm | 387002 | 74185293
sk@gmail.com | abc | pqr | xyz | raj b 405 dc park | nadiad | incometax | 387002 | 8527419630
a@gmail.com | b | a | a | Incometax | a | a | 123 | 123
3 user registers
Copyright © 2024 . All rights reserved.
Jazzmin version 2.6.0

Day 45


Dashboard
Authentication and Authorization
Groups
Users
Laboratoryapp
Appointments
Laboratory registers
Name_categorys
Test_categorys
Patientapp
User registers
Userfeedbacks
Lab Assistance Name *
soury
Lab Name *
dev lab
Blood Test Name *
CBC
Patient Name *
k
Email *
demo@gmail.com
Haemoglobin *
9
RBC Count *
5
PCV *
5
MCV *
5
MCH *
5
MCHC *
5
RDW *
5
Total WBC Count *
5
Neutrophils *
5
Lymphocytes *
5
Eosinophils *
5
Monocytes *
5
Basophils *
55
Platelet Count *
5
Save
Delete
Save and add another
Save and continue editing
History

Tasks

Task 1: Project Planning

Define project scope, objectives, and requirements

```
# project_settings.py
PROJECT_NAME = "Ecommerce Website"
SCOPE = "To build a fully functional ecommerce platform using Django."
OBJECTIVES = ["Implement user authentication", "Create product management system",
"Integrate payment gateway"]
```

Task 2: Django Setup

Install Django framework.

```
pip install django
```

Task 3: Create Django Project

```
django-admin startproject ecommerce_project
```

Task 4: Database Design

Design the database schema.

```
# models.py
from django.db import models
class Product(models.Model):
    name = models.CharField(max_length=100)
    price = models.DecimalField(max_digits=10, decimal_places=2)
```

```
    description = models.TextField()
```

Task 5: Research and Analysis

- Analyze competitor websites for insights.
- Conduct market research on ecommerce trends.

Task 6: User Authentication

Implement user registration and login functionality.

```
# views.py
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth import login, authenticate

def register(request):
    form = UserCreationForm()
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('/')
    return render(request, 'registration/register.html', {'form': form})
```

Task 7: Product Management

Develop CRUD operations for products.

```
# views.py
```

```
from .models import Product
from django.shortcuts import render, get_object_or_404

def product_detail(request, product_id):
    product = get_object_or_404(Product, pk=product_id)
    return render(request, 'product/detail.html', {'product': product})
```

Task 8: Shopping Cart

Create a shopping cart system.

```
# models.py
class Cart(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    items = models.ManyToManyField(Product)
    quantity = models.PositiveIntegerField(default=1)
```

Task 9: Checkout Process

Design and implement a multi-step checkout process.

```
# views.py
def checkout(request):
    if request.method == 'POST':
        # Handle payment processing
        return redirect('order_confirmation')
    return render(request, 'checkout.html')
```

Task 10: User Profiles and Orders

Develop user profile pages.

```
# views.py
def profile(request):
    user = request.user
    orders = Order.objects.filter(user=user)
    return render(request, 'profile.html', {'user': user, 'orders': orders
```

Task 11: Backend Administration

Create an admin panel for site management.

```
# admin.py
from django.contrib import admin
from .models import Product, Order

admin.site.register(Product)
```

```
admin.site.register(Order
```

Task 12: Base Page

Convert wireframes into responsive web pages.

```
<!-- base.html -->
<html>
<head>
  <title>{% block title %}{% endblock %}</title>
</head>
<body>
  {% block content %}
  {% endblock %}
</body>
</html>
```

Task 13: Encrypt Util

```
from cryptography.fernet import Fernet
import base64
import logging
import traceback
from django.conf import settings

def encrypt(pas):
    try:
        pas = str(pas)
        cipher_pass = Fernet(settings.ENCRYPT_KEY)
        encrypt_pass = cipher_pass.encrypt(pas.encode('UTF-8'))
        encrypt_pass = base64.urlsafe_b64encode(encrypt_pass).decode("UTF-8")
        return encrypt_pass
    except Exception as e:
        logging.getLogger("error_logger").error(traceback.format_exc())
        return None
```

Task 14: Decrypt Util

```
def decrypt(pas):
    try:
        pas = base64.urlsafe_b64decode(pas)
        cipher_pass = Fernet(settings.ENCRYPT_KEY)
        decod_pass = cipher_pass.decrypt(pas).decode("UTF-8")
        return decod_pass
    except Exception as e:
        logging.getLogger("error_logger").error(traceback.format_exc())
        return None
```

Task 15: App Info

```
from django.apps import AppConfig


class App1Config(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'app1'
```

Task 16: Mapping

```
from django.urls import path
from .views import *

urlpatterns = [
    path('reg/',reg,name='userreg'),
    # path('reg/',),
    path('',index,name='index'),
    path('table/',table,name='table'),
    path('cat/',cat,name='cat'),
    path('data/',data,name='data'),
    path('img/',image,name='img'),
    path('login/',login,name='login'),
    path('logout/',logout,name='logout'),
    path('profile/',profile,name='profile'),
    path('card/<int:id>',card,name='card'),
    path('pro/<int:id>',pro,name='pro'),
    path('prodetails/<int:id>',prodetails,name='prodetails'),
    path('cart/',cartdata,name='cart'),
    path('add1/<int:id>',additem,name='add1'),
    path('minus/<int:id>',minus,name='minus'),
    path('removeall/',removeall,name='removeall'),
    path('venreg/',venreg,name='venreg'),
    path('venlogin/',venlogin,name='venlogin'),
    path('addpro/',addproducts,name = 'addpro'),
    path('venpro/',venpro,name = 'venpro'),
    path('updatepro/<int:id>',updatepro,name='updatepro'),
    path('deletepro/<int:id>',deletepro,name='deletepro'),
    path('checkout/',checkout,name='checkout'),
    path('orderhistory/',orderhistory,name='orderhistory'),
    path('soldpro/',soldpro,name='soldpro'),
    path('razorpay/',razorpaypayment,name= 'razorpay'),
```

```
    path('paymenthandler/',paymenthandler,name='paymenthandler'),
    path('otp/',otp,name='otp'),

]
```

Task 17: Database Config

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'app1',
        'HOST': '127.0.0.1',
        'PORT': '3306',
        'USER': 'root',
        'PASSWORD': 'root',
    }
}
```

Task 18: Static & Media Files

```
STATIC_URL = '/static/'
```

```
import os
MEDIA_ROOT = os.path.join(BASE_DIR,'media/img/')
MEDIA_URL = '/media/img/'
```

Task 19: Email Config

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_HOST_USER = 'pqr6997@gmail.com'
EMAIL_HOST_PASSWORD = 'howk ouoo xljq lqwe'
EMAIL_USE_TLS = True
EMAIL_PORT = 587
```

Task 20: Table Config

```
def table(request):
    a = userregister.objects.get(name = 'a')
    # print(a)
    # for i in a:
    #     print(i.email)
    return render(request,'table.html',{'data':a})
```

Task 21: Categories

```
def cat(request):
    if 'email' in request.session:
        b = userregister.objects.get(email = request.session['email'])
        a = category.objects.all()
        return render(request,'cat.html',{'cat':a,'session':b})
    else:
        a = category.objects.all()
        return render(request,'cat.html',{'cat':a})
```

```
class category(models.Model):
    catname = models.CharField(max_length=50)
    image  =  models.ImageField(upload_to='imgcat')

    def __str__(self):
        return self.catname
```

Task 22: Ecom Data Store

```
def data(request):
    if request.method == 'POST':
        a = Blog()
        a.name = request.POST['uname']
        a.tagline = request.POST['data']
        a.save()
        print("data stored succesfullyy....")
        return render(request,'form.html')
    else:
        print("failed to store data.......")
        return render(request,'form.html')
```

Task 23: Image Processing

```
def image(request):
    if request.method == 'POST' and request.FILES['img']:
            a = category()
            a.catname = request.POST['uname']
            a.image = request.FILES['img']
            a.save()
            # return render(request,'img.html')
            return redirect('venlogin')
    else:
            return render(request,'img.html')
```

Task 24: OTP Config

```
def otp(request):
    if request.method == 'POST':
        if int(request.session['otp']) == int(request.POST['otp']):
            return redirect('index')
        else:
            return render(request,'otp.html',{'invalid':"Invalid OTP"})

    else:
        return render(request,'otp.html')
```

Task 25: Profile Fetch

```
def profile(request):
    if 'email' in request.session:
        a = userregister.objects.get(email = request.session['email'])
        if request.method == 'POST':
            a.name =  request.POST['uname']
            a.mob = request.POST['mob']
            a.add = request.POST['add']
            a.save()
            return render(request,'profile.html',{'session':a})
        else:
            return render(request,'profile.html',{'session':a})
    else:
        return redirect('login')
```

Task 26: Card

```
def card(request,id):
    if 'email' in request.session:
        b = userregister.objects.get(email = request.session['email'])
        a = category.objects.get(id = id)
        return render(request,'card.html',{'cat':a,'session':b})
    else:
        return redirect('login')
```

Task 27: All Products Fetch

```
def pro(request,id):
    if 'email' in request.session:
        b = userregister.objects.get(email = request.session['email'])
        a = product.objects.filter(category = id)
        return render(request,'pro.html',{'pro':a,'session':b})
    else:
        return redirect('login')
```

Task 28: Products Details

```
def prodetails(request,id):
    if 'email' in request.session:
        b = userregister.objects.get(email = request.session['email'])
        a = product.objects.get(id = id)
        if request.method == 'POST':
            cartdata = cart()
            cartdata.productid = id
            cartdata.userid = request.session['userid']
            cartdata.quantity = request.POST['qty']
            cartdata.totalprice = int(request.POST['qty']) * int(a.price)
            cartdata.orderid = "0"
            c = cart.objects.filter(productid = id,userid =
request.session['userid'],orderid = "0")
            if c:
                return
render(request,'product.html',{'session':b,'prodetails':a,'chej':'cheejjjj!!!'}
)
            else:
                cartdata.save()
                a.qty = a.qty - int(request.POST['qty'])
                a.save()
                return
render(request,'product.html',{'session':b,'prodetails':a})
        else:
            return render(request,'product.html',{'session':b,'prodetails':a})
    else:
        return redirect('login')
```

Task 29: Update Product

```
def updatepro(request,id):
    if request.method == 'POST':
        a = product.objects.get(id = id)
        a.price = request.POST['price']
        a.qty = request.POST['qty']
        a.discrption = request.POST['des']
        a.save()
        # return render(request,'updatepro.html',{'proupdatedetail':a})
        return redirect('venpro')
    else:
        a = product.objects.get(id = id)
        return render(request,'updatepro.html',{'proupdatedetail':a})
```

Task 30: Delete Product

```
def deletepro(request,id):
    if 'vendoremail' in request.session:
        ven = vendor.objects.get(email = request.session['vendoremail'])
        a = product.objects.get(id = id)
        a.delete()
        return redirect('venpro')
    else:
        return redirect('venlogin')
```

Task 31: Product Sold

```
def soldpro(request):
    if 'vendoremail' in request.session:
        cartdata = cart.objects.all()
        prolist= []
        for i in cartdata:
            print(i,"carrrtttt datattttttaa")
            if i.orderid != "0":
                print(i, "ordered dataaaaaa.......")
                pro = product.objects.get(id = i.productid)
                print(pro.name,"prooduct nameeeeeeeeeee......")
                print(request.session['vendorid'],pro.vendor,"vendooorrrrr
idd......")
                if int(pro.vendor) == int(request.session['vendorid']):
                    print(pro.name,"product name of vendorr......")
                    userdetails = userregister.objects.get(id = i.userid)
                    prodict =
{'proimg':pro.image,'proname':pro.name,'qty':i.quantity,'uname':userdetails.nam
e}
                    prolist.append(prodict)
        return
render(request,'soldpro.html',{'prolist':prolist,'vensession':True})
    else:
        return redirect('venlogin')
```

Tast 32: Order History

```
def orderhistory(request):
    if 'email' in request.session:
        a = cart.objects.filter(userid = request.session['userid'])
        prolist = []
        for i in a:
            if i.orderid != "0":
                orderdt = order.objects.get(id = i.orderid)
                prodetails = product.objects.get(id = i.productid)
                prodict =
{'proname':prodetails.name,'proimg':prodetails.image,'prodis':prodetails.discrp
tion,'datetime':orderdt.datetime,'qty':i.quantity,'tp':i.totalprice,'sp':prodet
ails.price}
                prolist.append(prodict)
```

```
            else:
                pass
        # print(total,"totalll producttsss........")
        return
render(request,'orderhistory.html',{'order':a,'prolist':prolist})
    else:
        return redirect('login')
```

Task 33: Landing Page

```
def index(request):
    if 'email' in request.session:
        b = userregister.objects.get(email = request.session['email'])
        a = category.objects.all()
        return render(request,'index.html',{'category':a,'session':b})
    elif 'vendoremail' in request.session:
        a = category.objects.all()
        b = vendor.objects.get(email = request.session['vendoremail'])
        return render(request,'index.html',{'category':a,'vensession':b})
    else:
        a = category.objects.all()
        return render(request,'index.html',{'category':a})
```

Task 34: Razorpay Config

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script src="https://checkout.razorpay.com/v1/checkout.js"></script>
  <script>
    var options = {

      // Enter the Key ID generated from the Dashboard
      key: "{{ razorpay_merchant_key }}",

      // Amount is in currency subunits.
      // Default currency is INR. Hence,
      // 50000 refers to 50000 paise
      amount: "{{ razorpay_amount }}",
      currency: 'INR',

      // Your/store name.
      name: "morning E-commerce",

      // Pass the `id` obtained in the response of Step 1
      order_id: "{{ razorpay_order_id }}",
      callback_url: "{{ callback_url }}",
    };
```

```
    // initialise razorpay with the options.
    var rzp1 = new Razorpay(options);
    rzp1.open();

  </script>
</body>
</html>
```

Task 35: Vendor Product

```
{% extends "nav.html" %}
{% block abc %}
{% load static %}
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>E-commerce</title>
    <link
      rel="stylesheet"
      45

href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    />
    <link
      rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css
"

integrity="sha512-z3gLpd7yknf1YoNbCzqRKc4qyor8gaKU1qmn+CShxbuBusANI9QpRohGBreCF
kKxLhei6S9CQXFEbbKuqLg0DA=="
      crossorigin="anonymous"
      referrerpolicy="no-referrer"
    />
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></scr
ipt>
    <link rel="stylesheet" href= {% static "ecom.css" %}/>
  </head>
  <body>
    <!--====================== Navbar ======================-->

    <!--======================Carousel Section======================-->
    <div id="myCarousel" class="carousel slide" data-ride="carousel">
      <ol class="carousel-indicators">
        <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
        <li data-target="#myCarousel" data-slide-to="1"></li>
        <li data-target="#myCarousel" data-slide-to="2"></li>
      </ol>

      <div class="carousel-inner">
        <div class="carousel-item active">
```

```
        <img src="Img/151.jpg" alt="First Slide" />
      </div>
      <div class="carousel-item">
        <img src="Img/6068656.jpg" alt="Second Slide" />
      </div>
      <div class="carousel-item">
        <img src="Img/3671942.jpg" alt="Third Slide" />
      </div>
    </div>

    <a
      class="carousel-control-prev"
      href="#myCarousel"
      role="button"
      data-slide="prev"
    >
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="sr-only">Previous</span>
    </a>
    <a
      class="carousel-control-next"
      href="#myCarousel"
      role="button"
      data-slide="next"
    >
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="sr-only">Next</span>
    </a>
  </div>
  <!--=====================Service Section=====================-->
  <section class="container-fluid py-2">
    <div class="container">
      <div class="row">
        <div
          class="col-4 d-flex justify-content-center align-items-center p-4"
        >
          <i class="fa-solid fa-truck-fast fa-2x mr-3"></i>
          <p class="text-left p-0">
            Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dolorum
          </p>
        </div>
        <div
          class="col-4 d-flex justify-content-center align-items-center p-4"
        >
          <i class="fa fa-plane-circle-check fa-2x mr-3"></i>
          <p class="text-left p-0">
            Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dolorum
          </p>
        </div>
        <div
          class="col-4 d-flex justify-content-center align-items-center p-4"
        >
          <i class="fa fa-handshake fa-2x mr-3"></i>
          <p class="text-left p-0">
            Lorem ipsum dolor sit amet, consectetur adipisicing elit. Dolorum
          </p>
        </div>
```

```
      </div>
    </div>
  </section>
  <!--======================Collection Section======================-->
  <section class="container">
    <div class="mt-3">
      <div class="row justify-content-center">
        {% for i in  prod %}
        <div class="col-lg-3 col-md-6 p-0">
          <div class="card p-1 border-0 scale-style">
            <a href=>
              <img
                class="card-img-top img-hover"
                src={{i.image.url}}
                alt="Card image cap"
                style="height: 300px"
              />
            </a>
            <div class="card-body text-center">
              <p class="card-text text-center">{{i.catname}}</p>
              <a href={% url "deletepro" i.id  %}>
                <button type="button" class="btn btn-outline-danger mt-2">
                  Delete
                </button>
              </a>
              <a href={% url "updatepro" i.id %}>
                <button type="button" class="btn btn-outline-success mt-2">
                  Update
                </button>
              </a>
            </div>
          </div>
        </div>
        {% endfor %}

      </div>
    </div>
  </section>
  <!--======================Footer Section======================-->
  <footer class="bg-dark text-light py-4">
    <div class="container">
      <div class="row">
        <div class="col-md-4">
          <h5>Contact Us</h5>
          <p>Email: info@example.com</p>
          <p>Phone: 123-456-7890</p>
        </div>
        <div class="col-md-4">
          <h5>Links</h5>
          <ul class="list-unstyled">
            <li><a href="#">Home</a></li>
            <li><a href="#">Products</a></li>
            <li><a href="#">About Us</a></li>
            <li><a href="#">Contact Us</a></li>
          </ul>
        </div>
        <div class="col-md-4">
```

```
            <h5>Follow Us</h5>
            <ul class="list-inline">
              <li class="list-inline-item">
                <a href="#"><i class="fab fa-facebook"></i></a>
              </li>
              <li class="list-inline-item">
                <a href="#"><i class="fab fa-twitter"></i></a>
              </li>
              <li class="list-inline-item">
                <a href="#"><i class="fab fa-instagram"></i></a>
              </li>
            </ul>
          </div>
        </div>
        <hr />
        <div class="row">
          <div class="col text-center">
            <p>&copy; 2023 Your E-Commerce Store</p>
          </div>
        </div>
      </div>
    </footer>
  </body>
</html>
{% endblock abc %}
```

Task 36: Vendor Product Fetch

```
def venpro(request):
    if 'vendoremail' in request.session:
        b = vendor.objects.get(email = request.session['vendoremail'])
        c = product.objects.filter(vendor = b.pk)
        return render(request,'venpro.html',{'vensession':b,'prod':c})
    else:
        return redirect('venlogin')
```

Task 37: Navigation

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
        <title>E-commerce</title>
        <link
          rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"

integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6
JXm"
          crossorigin="anonymous"
        />
        <link
          rel="stylesheet"

href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css
"

integrity="sha512-z3gLpd7yknf1YoNbCzqRKc4qyor8gaKU1qmn+CShxbuBusANI9QpRohGBreCF
kKxLhei6S9CQXFEbbKuqLg0DA=="
          crossorigin="anonymous"
          referrerpolicy="no-referrer"
        />
        <script
          src="https://code.jquery.com/jquery-3.2.1.slim.min.js"

integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5
KkN"
          crossorigin="anonymous"
        ></script>
        <script

src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"

integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0
b4Q"
          crossorigin="anonymous"
        ></script>
        <script

src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"

integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVC
mYl"
          crossorigin="anonymous"
        ></script>
        <link rel="stylesheet" href="Style.css" />
      </head>
<body>
        <!-=======================- Navbar =-=======================-->
        <header class="container-fluid navbar-light bg-light">
```

```
            <nav class="navbar container navbar-expand-lg navbar-light
bg-light">
              <a class="navbar-brand" href="index.html">Navbar</a>
              <button
                class="navbar-toggler"
                type="button"
                data-toggle="collapse"
                data-target="#navbarSupportedContent"
                aria-controls="navbarSupportedContent"
                aria-expanded="false"
                aria-label="Toggle navigation"
              >
                <span class="navbar-toggler-icon"></span>
              </button>

              <div class="collapse navbar-collapse"
id="navbarSupportedContent">
                <ul class="navbar-nav ml-auto">
                  <li class="nav-item">
                    <a class="nav-link" href={% url "index" %}>Home</a>
                  </li>
                  {% if session %}
                  <li class="nav-item">
                    <a class="nav-link" href={% url "logout" %}>Logout</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href="orderhistory.html">Order
History</a>
                  </li>
                  {% elif vensession %}
                  <li class="nav-item">
                    <a class="nav-link" href={% url "logout" %}>Logout</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href={% url "addpro" %}>Start
selling..!</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href={% url "soldpro" %}>Sold
Products</a>
                  </li>
                  {% else %}
                  <li class="nav-item">
                    <a class="nav-link" href={% url "login" %}>User Login</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href={% url "venlogin" %}>vendor
Login</a>
                  </li>
                  {% endif %}
                  <li class="nav-item">
                    <a class="nav-link" href={% url "profile" %}>Profile</a>
                  </li>

                  <li class="nav-item">
                    <a class="nav-link" href="addproduct.html">
                      <i class="fa-solid fa-plus-square text-dark"
```

```
aria-hidden="true">
                        Add product</i
                      >
                    </a>
                  </li>
                  <li class="nav-item">
                    <input
                      class="input-group-text"
                      type="text"
                      placeholder="Search"
                    />
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href={% url "cart" %}
                      ><i class="fa fa-cart-shopping text-dark"
aria-hidden="true"></i
                    ></a>
                  </li>
                  {% if session %}
                  <li class="nav-item">
                    <a class="nav-link" href="#">Welcome,{{session.name}}</a>
                  </li>
                  {% elif vensession %}
                  <li class="nav-item">
                    <a class="nav-link"
href="#">Welcome,{{vensession.name}}</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href={% url "venpro" %}>My products</a>
                  </li>
                  {% endif %}
                </ul>
              </div>
            </nav>
          </header>
          <!-- content -->
          {% block abc %}
          {% endblock abc %}
</body>
</html>
```

Task 38: Generalized Form

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form method="post">
        {% csrf_token %}
        <label for="">Enter name: </label>
        <input type="text" name="uname">
```

```
        <label for="">Enter text: </label>
        <input type="text" name="data">

        <button type="submit"> Submit </button>
    </form>
</body>
</html>
```

Task 39: Payment Success Check

```
<h2>Payment Success.......</h2>
```

Task 40: Necessary Settings

```
INSTALLED_APPS = [
    'jazzmin',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'app1',

]
```

Task 41: Define Encryption Key

```
import os
ENCRYPT_KEY = b'DJ3r34r9zSRPM8OWucrsv2PDBEA6BYaGpfJx67C9_us='
```

Task 42: Testing and Optimization

Write unit tests for each module and functionality.

```
# tests.py
from django.test import TestCase
from .models import Product

class ProductTestCase(TestCase):
    def setUp(self):
        Product.objects.create(name="Test Product", price=10.0,
description="Test Description")

    def test_product_creation(self):
        product = Product.objects.get(name="Test Product")
        self.assertEqual(product.price, 10.0)
```

Task 43: Deployment

Set up hosting environment.

```
# Example: Heroku deployment
heroku create
git push heroku master
heroku open
```

Task 44: Documentation

Document codebase, APIs, and configurations.

```
## API Documentation

### Products
- **GET /api/products**: Get all products
- **POST /api/products**: Create a new product
```

Configuration

- **Database**: PostgreSQL

- **Payment Gateway**: Stripe

Task 45: Post-Deployment Support

- Monitor website performance and user feedback.
- Provide ongoing maintenance and support.