**WEB APPLICATION WITH CRUD OPERATIONS**

**Aim:**

To build a simple and a small web application which can perform the below CRUD operations (CREATE, READ, UPDATE, DELETE).

**Technical skills used :**

• Language: Python, HTML, CSS

• Framework: Django

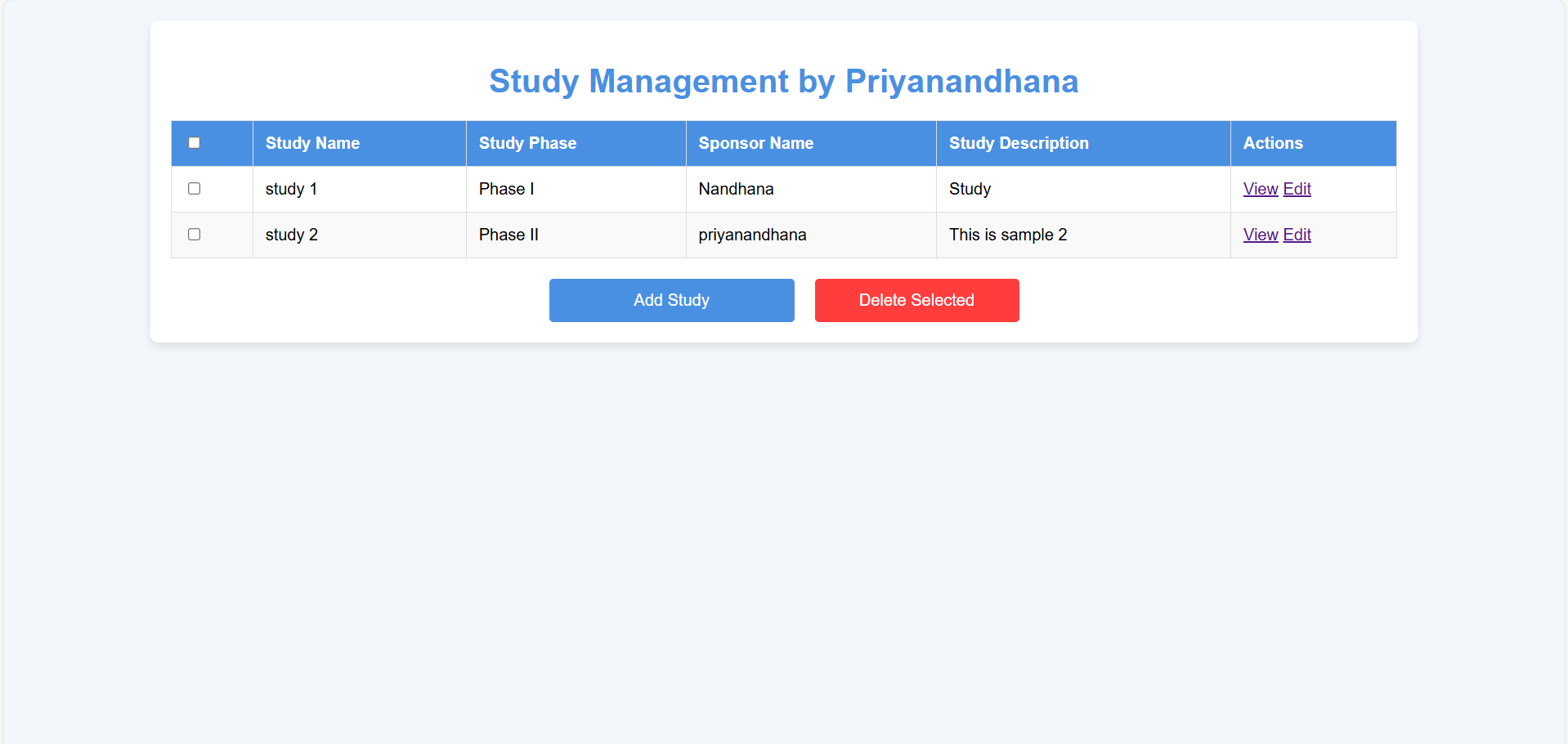
• Database: MySQL

• Logging

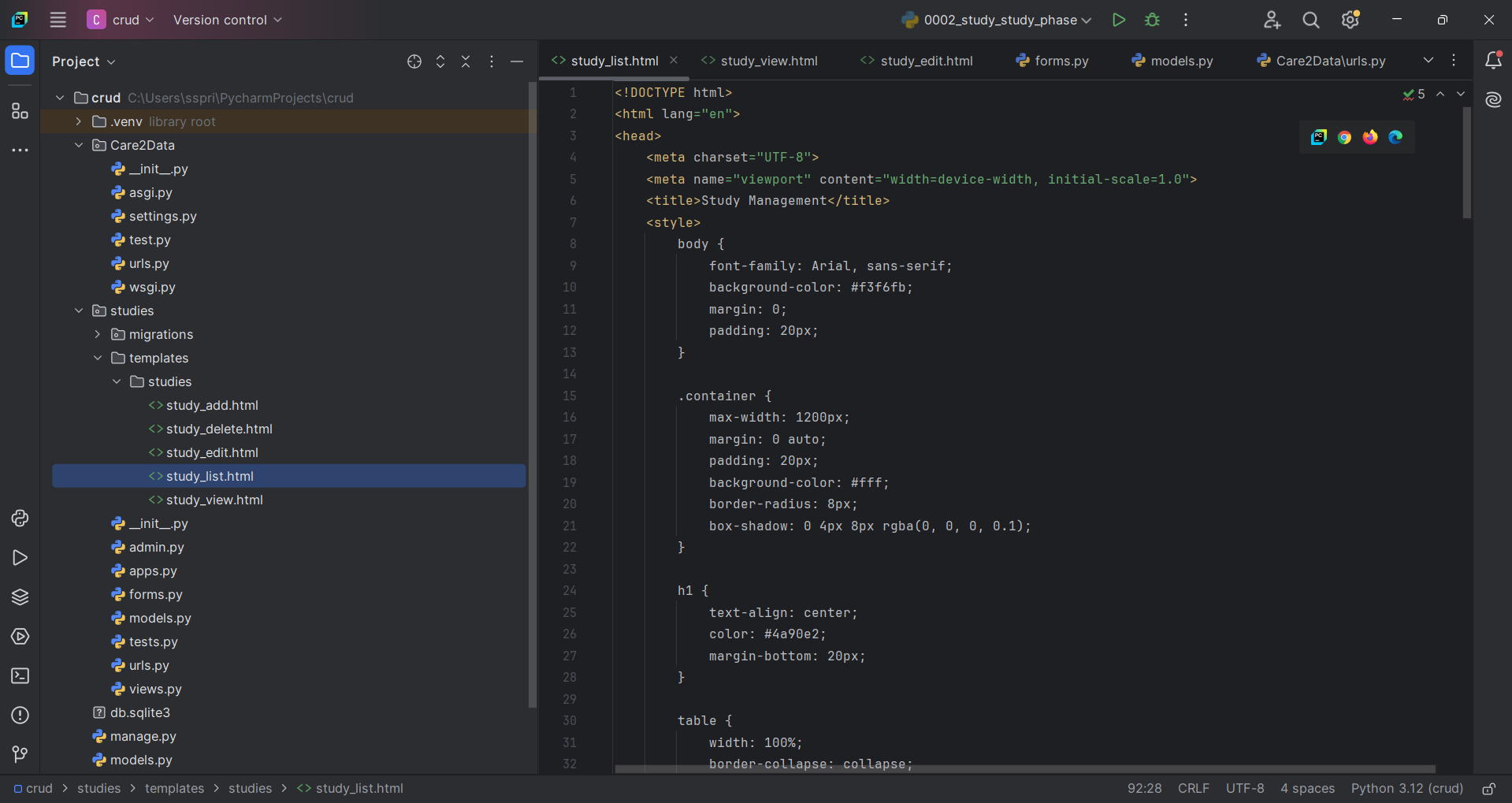
• Exception handling

**Test Duration:** 6 hours

**Main Page :**



**IDE: PyCharm**



**Code:**

**Settings.py**

*"""  
Django settings for Care2Data project.  
  
Generated by 'django-admin startproject' using Django 5.1.3.  
  
For more information on this file, see  
https://docs.djangoproject.com/en/5.1/topics/settings/  
  
For the full list of settings and their values, see  
https://docs.djangoproject.com/en/5.1/ref/settings/  
"""*from pathlib import Path  
  
# Build paths inside the project like this: BASE\_DIR / 'subdir'.  
BASE\_DIR = Path(\_\_file\_\_).resolve().parent.parent  
  
  
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = 'django-insecure-e@j+uw\_^@1nd5v39xck4!2pv)ef)n7msye$&#3e(t0enn0xiat'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
ALLOWED\_HOSTS = []  
  
  
# Application definition  
  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'studies',  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'Care2Data.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [BASE\_DIR / "templates"],  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'Care2Data.wsgi.application'  
  
  
# Database  
# https://docs.djangoproject.com/en/5.1/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql', # Use MySQL backend  
 'NAME': 'crud', # Replace with your database name  
 'USER': 'root', # Replace with your MySQL username (default is 'root')  
 'PASSWORD': 'root', # Replace with your MySQL password  
 'HOST': 'localhost', # Or use your MySQL server hostname  
 'PORT': '3306', # Default MySQL port  
 }  
}  
  
  
  
  
  
# Password validation  
# https://docs.djangoproject.com/en/5.1/ref/settings/#auth-password-validators  
  
AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 },  
]  
  
  
# Internationalization  
# https://docs.djangoproject.com/en/5.1/topics/i18n/  
  
LANGUAGE\_CODE = 'en-us'  
  
TIME\_ZONE = 'UTC'  
  
USE\_I18N = True  
  
USE\_TZ = True  
  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/5.1/howto/static-files/  
  
STATIC\_URL = 'static/'  
  
# Default primary key field type  
# https://docs.djangoproject.com/en/5.1/ref/settings/#default-auto-field  
  
DEFAULT\_AUTO\_FIELD = 'django.db.models.BigAutoField'

**urls.py**

from django.contrib import admin  
from django.urls import path, include  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('studies.urls')), # Include studies app URLs  
]

**studies/templates/studies/study\_list.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Study Management</title>  
 <style>  
 body {  
 font-family: Arial, sans-serif;  
 background-color: #f3f6fb;  
 margin: 0;  
 padding: 20px;  
 }  
  
 .container {  
 max-width: 1200px;  
 margin: 0 auto;  
 padding: 20px;  
 background-color: #fff;  
 border-radius: 8px;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
 }  
  
 h1 {  
 text-align: center;  
 color: #4a90e2;  
 margin-bottom: 20px;  
 }  
  
 table {  
 width: 100%;  
 border-collapse: collapse;  
 margin-bottom: 20px;  
 }  
  
 table thead {  
 background-color: #4a90e2;  
 color: #fff;  
 }  
  
 table th, table td {  
 text-align: left;  
 padding: 12px;  
 border: 1px solid #ddd;  
 }  
  
 table tr:nth-child(even) {  
 background-color: #f9f9f9;  
 }  
  
 table tr:hover {  
 background-color: #f1f1f1;  
 }  
  
 .actions a {  
 text-decoration: none;  
 padding: 8px 12px;  
 border-radius: 4px;  
 font-size: 14px;  
 }  
  
 .actions .view-btn {  
 background-color: #4caf50;  
 color: white;  
 }  
  
 .actions .edit-btn {  
 background-color: #ff9800;  
 color: white;  
 }  
  
 .buttons {  
 display: flex;  
 justify-content: center; /\* Centers the buttons horizontally \*/  
 gap: 20px;  
 margin-top: 20px;  
 }  
  
 .buttons a, .buttons button {  
 text-decoration: none;  
 background-color: #4a90e2; /\* Blue background for Add Study \*/  
 color: white;  
 padding: 12px 20px;  
 border-radius: 4px;  
 font-size: 16px;  
 border: none;  
 cursor: pointer;  
 width: 200px; /\* Same width for both buttons \*/  
 text-align: center; /\* Ensures text is centered \*/  
 }  
  
 .buttons a:hover, .buttons button:hover {  
 background-color: #357abd; /\* Darker blue for hover effect \*/  
 }  
  
 .buttons button {  
 background-color: #ff3d3d; /\* Red for delete button \*/  
 }  
  
 .buttons button:hover {  
 background-color: #e53935;  
 }  
  
 /\* Style for the "Add Study" button \*/  
 .buttons .add-btn {  
 background-color: #4a90e2; /\* Blue for Add Study \*/  
 }  
  
 .buttons .add-btn:hover {  
 background-color: #357abd; /\* Darker blue on hover \*/  
 }  
  
 /\* Style for the "Delete Selected" button \*/  
 .buttons .delete-btn {  
 background-color: #ff3d3d; /\* Red for delete \*/  
 }  
  
 .buttons .delete-btn:hover {  
 background-color: #e53935; /\* Darker red on hover \*/  
 }  
 </style>  
</head>  
<body>  
  
 <div class="container">  
 <h1>Study Management by Priyanandhana</h1>  
  
 <!-- Form for bulk deletion -->  
 <form method="POST" action="{% url 'study\_delete\_bulk' %}">  
 {% csrf\_token %}  
  
 <table class="table">  
 <thead>  
 <tr>  
 <th><input type="checkbox" id="select\_all" onclick="selectAll()"></th>  
 <th>Study Name</th>  
 <th>Study Phase</th>  
 <th>Sponsor Name</th>  
 <th>Study Description</th>  
 <th>Actions</th>  
 </tr>  
 </thead>  
 <tbody>  
 {% for study in studies %}  
 <tr>  
 <td><input type="checkbox" name="study\_ids" value="{{ study.id }}"></td>  
 <td>{{ study.study\_name }}</td>  
 <td>{{ study.study\_phase }}</td>  
 <td>{{ study.sponsor\_name }}</td>  
 <td>{{ study.study\_description }}</td>  
 <td>  
 <a href="{% url 'study\_view' study.id %}" class="btn btn-success">View</a>  
 <a href="{% url 'study\_edit' study.id %}" class="btn btn-warning">Edit</a>  
 </td>  
 </tr>  
 {% empty %}  
 <tr>  
 <td colspan="6">No studies available</td>  
 </tr>  
 {% endfor %}  
 </tbody>  
 </table>  
  
 <!-- Buttons to submit the selected studies for deletion -->  
 <div class="buttons">  
 <a href="{% url 'study\_add' %}" class="add-btn">Add Study</a>  
 <button type="submit" class="delete-btn">Delete Selected</button>  
 </div>  
 </form>  
 </div>  
  
 <script>  
 // JavaScript for "Select All" functionality  
 function selectAll() {  
 const checkboxes = document.querySelectorAll('input[name="study\_ids"]');  
 checkboxes.forEach(function(checkbox) {  
 checkbox.checked = document.getElementById('select\_all').checked;  
 });  
 }  
 </script>  
</body>  
</html>

**studies/templates/studies/study\_add.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Add Study</title>  
 <style>  
 body {  
 font-family: Arial, sans-serif;  
 background-color: #f3f6fb;  
 margin: 0;  
 padding: 20px;  
 }  
  
 .container {  
 max-width: 600px;  
 margin: 0 auto;  
 padding: 20px;  
 background-color: #fff;  
 border-radius: 8px;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
 }  
  
 h1 {  
 text-align: center;  
 color: #4a90e2;  
 margin-bottom: 20px;  
 }  
  
 form {  
 display: flex;  
 flex-direction: column;  
 }  
  
 label {  
 margin: 10px 0 5px;  
 font-weight: bold;  
 }  
  
 input[type="text"], input[type="textarea"], select {  
 padding: 10px;  
 margin-bottom: 20px;  
 border: 1px solid #ddd;  
 border-radius: 4px;  
 }  
  
 button {  
 background-color: #4a90e2;  
 color: white;  
 padding: 12px 20px;  
 border: none;  
 border-radius: 4px;  
 font-size: 16px;  
 cursor: pointer;  
 }  
  
 button:hover {  
 background-color: #357abd;  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <h1>Add Study</h1>  
 <form method="POST">  
 {% csrf\_token %}  
 {{ form.as\_p }} <!-- Render the form fields -->  
 <button type="submit">Add Study</button>  
 </form>  
 </div>  
</body>  
</html>

**studies/templates/studies/study\_edit.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Edit Study</title>  
 <style>  
 body {  
 font-family: Arial, sans-serif;  
 background-color: #f3f6fb;  
 margin: 0;  
 padding: 20px;  
 }  
 .container {  
 max-width: 900px;  
 margin: 0 auto;  
 padding: 20px;  
 background-color: #fff;  
 border-radius: 8px;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
 }  
 h1 {  
 text-align: center;  
 color: #4a90e2;  
 margin-bottom: 20px;  
 }  
 label {  
 font-size: 18px;  
 color: #333;  
 margin-bottom: 5px;  
 display: inline-block;  
 }  
 input[type="text"], textarea, select {  
 width: 100%;  
 padding: 10px;  
 margin-bottom: 15px;  
 border: 1px solid #ddd;  
 border-radius: 4px;  
 font-size: 16px;  
 }  
 textarea {  
 resize: vertical;  
 height: 120px;  
 }  
 button {  
 background-color: #4a90e2;  
 color: white;  
 padding: 12px 20px;  
 border-radius: 4px;  
 font-size: 16px;  
 border: none;  
 cursor: pointer;  
 margin-right: 10px;  
 }  
 button:hover {  
 background-color: #357abd;  
 }  
 .buttons {  
 display: flex;  
 justify-content: center;  
 gap: 15px;  
 }  
 a {  
 text-decoration: none;  
 }  
 </style>  
</head>  
<body>  
 <div class="container">  
 <h1>Edit Study</h1>  
 <form method="post" action="{% url 'study\_edit' study.id %}">  
 {% csrf\_token %}  
  
 <!-- Render the form fields using {{ form.as\_p }} -->  
 {{ form.as\_p }}  
  
 <div class="buttons">  
 <button type="submit">Update</button>  
 <a href="{% url 'study\_list' %}"><button type="button">Cancel</button></a>  
 </div>  
 </form>  
 </div>  
</body>  
</html>

**studies/templates/studies/study\_view.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>View Study</title>  
 <style>  
 body {  
 font-family: Arial, sans-serif;  
 background-color: #f3f6fb;  
 margin: 0;  
 padding: 20px;  
 }  
  
 .container {  
 max-width: 900px;  
 margin: 0 auto;  
 padding: 20px;  
 background-color: #fff;  
 border-radius: 8px;  
 box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);  
 }  
  
 h1 {  
 text-align: center;  
 color: #4a90e2;  
 margin-bottom: 20px;  
 }  
  
 p {  
 font-size: 18px;  
 margin-bottom: 10px;  
 color: #333;  
 }  
  
 strong {  
 color: #4a90e2;  
 }  
  
 .buttons {  
 display: flex;  
 justify-content: center;  
 margin-top: 20px;  
 }  
  
 .buttons a {  
 text-decoration: none;  
 background-color: #4a90e2;  
 color: white;  
 padding: 12px 20px;  
 border-radius: 4px;  
 font-size: 16px;  
 border: none;  
 cursor: pointer;  
 }  
  
 .buttons a:hover {  
 background-color: #357abd;  
 }  
  
 /\* Ensure paragraph content does not get cut off \*/  
 .study-details p {  
 word-wrap: break-word;  
 }  
 </style>  
</head>  
<body>  
  
 <div class="container">  
 <h1>View Study</h1>  
  
 <div class="study-details">  
 <p><strong>Study Name:</strong> {{ study.study\_name }}</p>  
 <p><strong>Study Description:</strong> {{ study.study\_description }}</p>  
 <p><strong>Study Phase:</strong> {{ study.study\_phase }}</p>  
 <p><strong>Sponsor Name:</strong> {{ study.sponsor\_name }}</p>  
 </div>  
  
 <div class="buttons">  
 <a href="{% url 'study\_list' %}"><button type="button">Cancel</button></a>  
 </div>  
 </div>  
  
</body>  
</html>

**studies/templates/studies/study\_delete.html**

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>Delete Study</title>  
</head>  
<body>  
 <h1>Delete Study</h1>  
 <p>Are you sure you want to delete the study "{{ study.study\_name }}"?</p>  
 <form method="post" action="{% url 'study\_delete' study.id %}">  
 {% csrf\_token %}  
 <button type="submit">Yes, Delete</button>  
 </form>  
 <a href="{% url 'study\_list' %}"><button type="button">Cancel</button></a>  
</body>  
</html>

**apps.py**

from django.apps import AppConfig  
  
class StudiesConfig(AppConfig):  
 default\_auto\_field = 'django.db.models.BigAutoField'  
 name = 'studies'

**forms.py**

from django import forms  
from .models import Study  
  
class StudyForm(forms.ModelForm):  
 class Meta:  
 model = Study  
 fields = ['study\_name', 'study\_phase', 'sponsor\_name', 'study\_description']  
 phase = forms.ChoiceField(choices=Study.PHASE\_CHOICES)

**models.py**

from django.db import models  
  
class Study(models.Model):  
 objects = None  
 PHASE\_CHOICES = [  
 ('Phase I', 'Phase I'),  
 ('Phase II', 'Phase II'),  
 ('Phase III', 'Phase III'),  
 ('Phase IV', 'Phase IV'),  
 ]  
  
 study\_name = models.CharField(max\_length=100, db\_column='study\_name')  
 study\_description = models.TextField(db\_column='study\_description')  
 study\_phase = models.CharField(max\_length=50, choices=PHASE\_CHOICES, db\_column='study\_phase')  
 sponsor\_name = models.CharField(max\_length=100, db\_column='sponsor\_name')  
  
 def \_\_str\_\_(self):  
 return self.study\_name

**studies/urls.py**

from django.urls import path  
from . import views  
  
urlpatterns = [  
 path('', views.study\_list, name='study\_list'),  
 path('add/', views.study\_add, name='study\_add'),  
 path('view/<int:id>/', views.study\_view, name='study\_view'),  
 path('edit/<int:id>/', views.study\_edit, name='study\_edit'),  
 path('delete/<int:study\_id>/', views.study\_delete, name='study\_delete'),  
 path('delete\_bulk/', views.study\_delete\_bulk, name='study\_delete\_bulk'),  
]

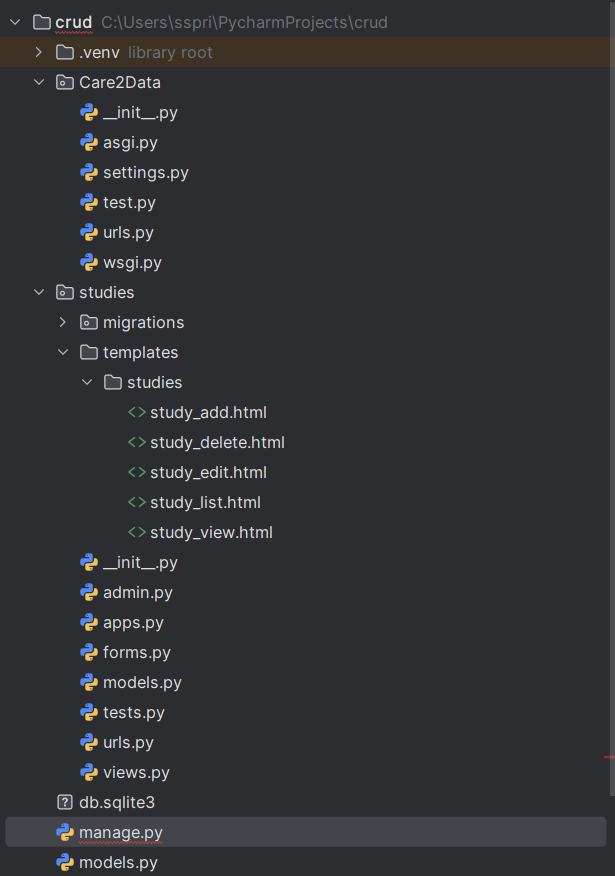
**studies/views.py**

from django.http import HttpResponse  
from django.shortcuts import render, get\_object\_or\_404, redirect  
from .models import Study  
from .forms import StudyForm  
  
# View for displaying the list of studies  
def study\_list(request):  
 studies = Study.objects.all() # Get all studies  
 return render(request, 'studies/study\_list.html', {'studies': studies})  
  
  
# View for adding a new study  
def study\_add(request):  
 if request.method == 'POST':  
 form = StudyForm(request.POST)  
 if form.is\_valid():  
 form.save() # Save the new study record  
 return redirect('study\_list') # Redirect to the study list after adding  
 else:  
 print(form.errors) # Log errors for debugging  
 else:  
 form = StudyForm() # Show an empty form for a GET request  
 return render(request, 'studies/study\_add.html', {'form': form})  
  
  
# View for viewing a study's details  
def study\_view(request, id):  
 study = get\_object\_or\_404(Study, id=id) # Get the study object or return 404  
 return render(request, 'studies/study\_view.html', {'study': study})  
  
  
# View for editing an existing study  
def study\_edit(request, id):  
 # Get the study object based on the id  
 study = get\_object\_or\_404(Study, pk=id)  
  
 if request.method == 'POST':  
 form = StudyForm(request.POST, instance=study) # Pre-populate the form with the study instance  
 if form.is\_valid():  
 form.save() # Save the updated study object  
 return redirect('study\_list') # Redirect to the study list page  
 else:  
 form = StudyForm(instance=study) # Populate form with current study data  
  
 return render(request, 'studies/study\_edit.html', {'form': form, 'study': study})  
  
  
  
# View for deleting a study  
def study\_delete(request, study\_id):  
 study = get\_object\_or\_404(Study, id=study\_id)  
  
 if request.method == 'POST':  
 study.delete() # Delete the study  
 return redirect('study\_list') # Redirect to study list after deletion  
  
 return render(request, 'studies/study\_delete.html', {'study': study})  
  
from django.shortcuts import render, redirect  
from .models import Study  
  
def study\_delete\_bulk(request):  
 if request.method == 'POST':  
 study\_ids = request.POST.getlist('study\_ids')  
 if study\_ids:  
 Study.objects.filter(id\_\_in=study\_ids).delete()  
 return redirect('study\_list') # Redirect to a page after deletion

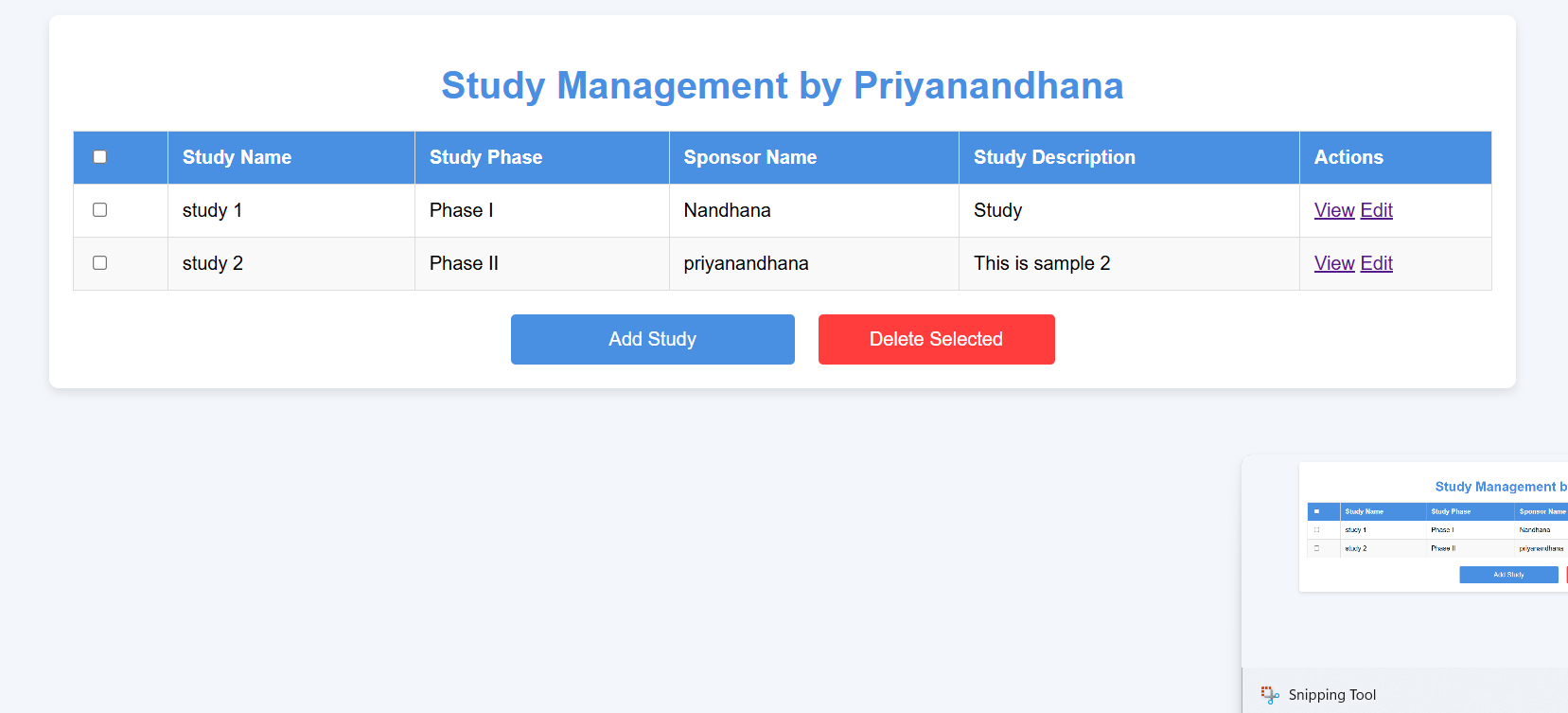
**models.py**

# This is an auto-generated Django model module.  
# You'll have to do the following manually to clean this up:  
# \* Rearrange models' order  
# \* Make sure each model has one field with primary\_key=True  
# \* Make sure each ForeignKey and OneToOneField has `on\_delete` set to the desired behavior  
# \* Remove `managed = False` lines if you wish to allow Django to create, modify, and delete the table  
# Feel free to rename the models, but don't rename db\_table values or field names.  
from django.db import models  
  
  
class AuthGroup(models.Model):  
 name = models.CharField(unique=True, max\_length=150)  
  
 class Meta:  
 managed = False  
 db\_table = 'auth\_group'  
  
  
class AuthGroupPermissions(models.Model):  
 id = models.BigAutoField(primary\_key=True)  
 group = models.ForeignKey(AuthGroup, models.DO\_NOTHING)  
 permission = models.ForeignKey('AuthPermission', models.DO\_NOTHING)  
  
 class Meta:  
 managed = False  
 db\_table = 'auth\_group\_permissions'  
 unique\_together = (('group', 'permission'),)  
  
  
class AuthPermission(models.Model):  
 name = models.CharField(max\_length=255)  
 content\_type = models.ForeignKey('DjangoContentType', models.DO\_NOTHING)  
 codename = models.CharField(max\_length=100)  
  
 class Meta:  
 managed = False  
 db\_table = 'auth\_permission'  
 unique\_together = (('content\_type', 'codename'),)  
  
  
class AuthUser(models.Model):  
 password = models.CharField(max\_length=128)  
 last\_login = models.DateTimeField(blank=True, null=True)  
 is\_superuser = models.IntegerField()  
 username = models.CharField(unique=True, max\_length=150)  
 first\_name = models.CharField(max\_length=150)  
 last\_name = models.CharField(max\_length=150)  
 email = models.CharField(max\_length=254)  
 is\_staff = models.IntegerField()  
 is\_active = models.IntegerField()  
 date\_joined = models.DateTimeField()  
  
 class Meta:  
 managed = False  
 db\_table = 'auth\_user'  
  
  
class AuthUserGroups(models.Model):  
 id = models.BigAutoField(primary\_key=True)  
 user = models.ForeignKey(AuthUser, models.DO\_NOTHING)  
 group = models.ForeignKey(AuthGroup, models.DO\_NOTHING)  
  
 class Meta:  
 managed = False  
 db\_table = 'auth\_user\_groups'  
 unique\_together = (('user', 'group'),)  
  
  
class AuthUserUserPermissions(models.Model):  
 id = models.BigAutoField(primary\_key=True)  
 user = models.ForeignKey(AuthUser, models.DO\_NOTHING)  
 permission = models.ForeignKey(AuthPermission, models.DO\_NOTHING)  
  
 class Meta:  
 managed = False  
 db\_table = 'auth\_user\_user\_permissions'  
 unique\_together = (('user', 'permission'),)  
  
  
class DjangoAdminLog(models.Model):  
 action\_time = models.DateTimeField()  
 object\_id = models.TextField(blank=True, null=True)  
 object\_repr = models.CharField(max\_length=200)  
 action\_flag = models.PositiveSmallIntegerField()  
 change\_message = models.TextField()  
 content\_type = models.ForeignKey('DjangoContentType', models.DO\_NOTHING, blank=True, null=True)  
 user = models.ForeignKey(AuthUser, models.DO\_NOTHING)  
  
 class Meta:  
 managed = False  
 db\_table = 'django\_admin\_log'  
  
  
class DjangoContentType(models.Model):  
 app\_label = models.CharField(max\_length=100)  
 model = models.CharField(max\_length=100)  
  
 class Meta:  
 managed = False  
 db\_table = 'django\_content\_type'  
 unique\_together = (('app\_label', 'model'),)  
  
  
class DjangoMigrations(models.Model):  
 id = models.BigAutoField(primary\_key=True)  
 app = models.CharField(max\_length=255)  
 name = models.CharField(max\_length=255)  
 applied = models.DateTimeField()  
  
 class Meta:  
 managed = False  
 db\_table = 'django\_migrations'  
  
  
class DjangoSession(models.Model):  
 session\_key = models.CharField(primary\_key=True, max\_length=40)  
 session\_data = models.TextField()  
 expire\_date = models.DateTimeField()  
  
 class Meta:  
 managed = False  
 db\_table = 'django\_session'  
  
  
class StudiesStudy(models.Model):  
 id = models.BigAutoField(primary\_key=True)  
 study\_name = models.CharField(max\_length=100)  
 study\_description = models.TextField()  
 sponsor\_name = models.CharField(max\_length=100)  
 study\_phase = models.CharField(max\_length=50, blank=True, null=True)  
  
 class Meta:  
 managed = False  
 db\_table = 'studies\_study'

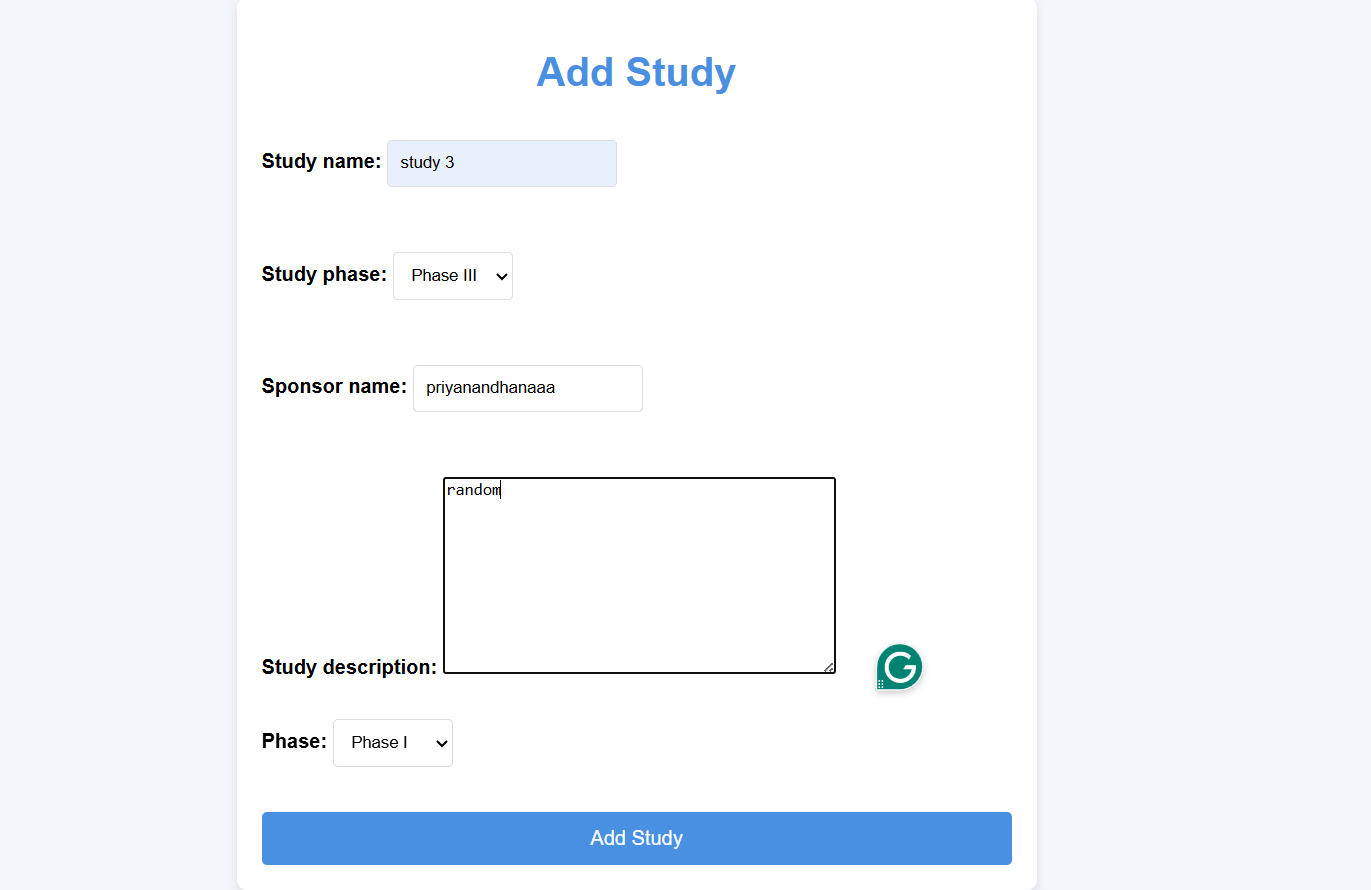
**My folders:**



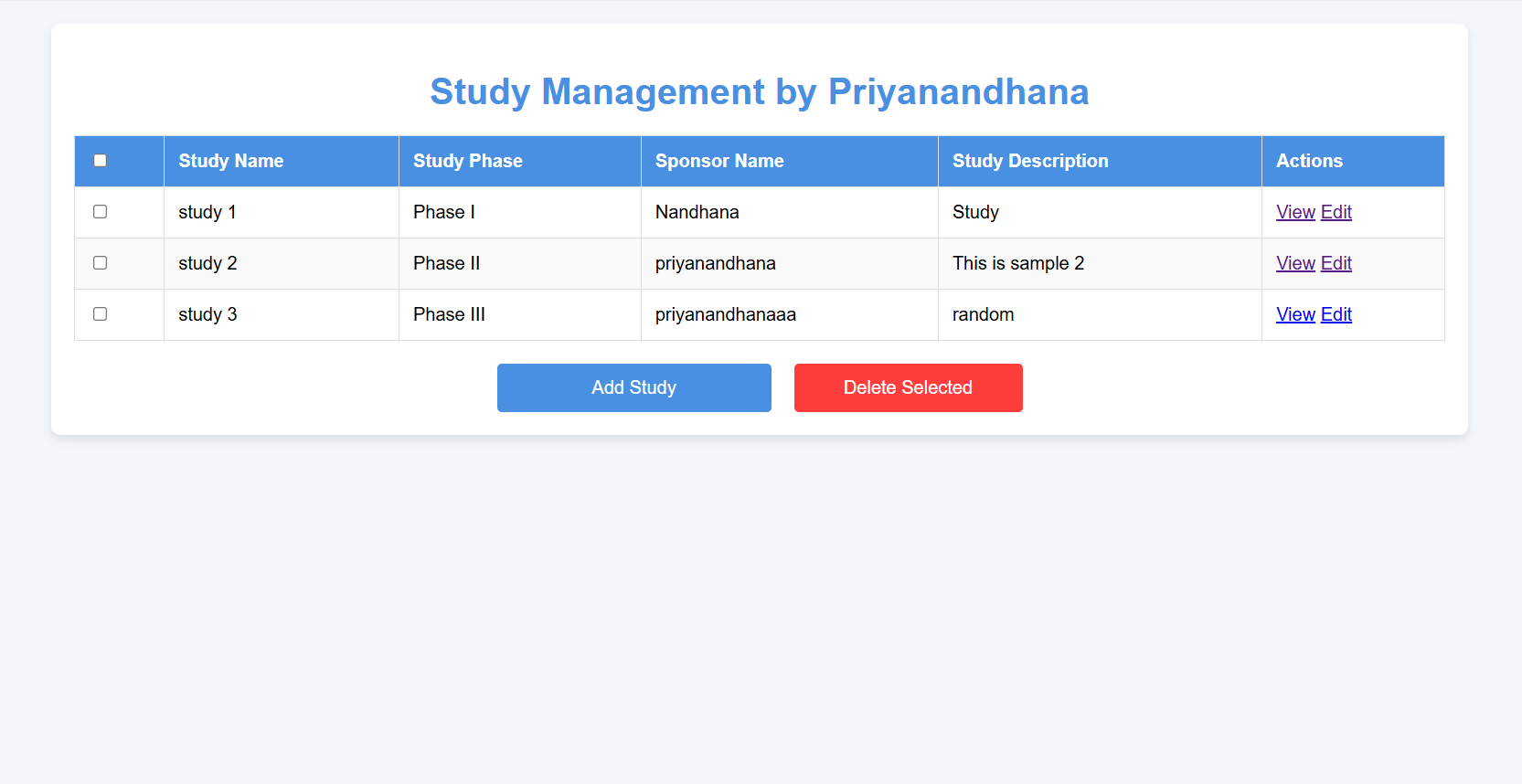
**Output :**



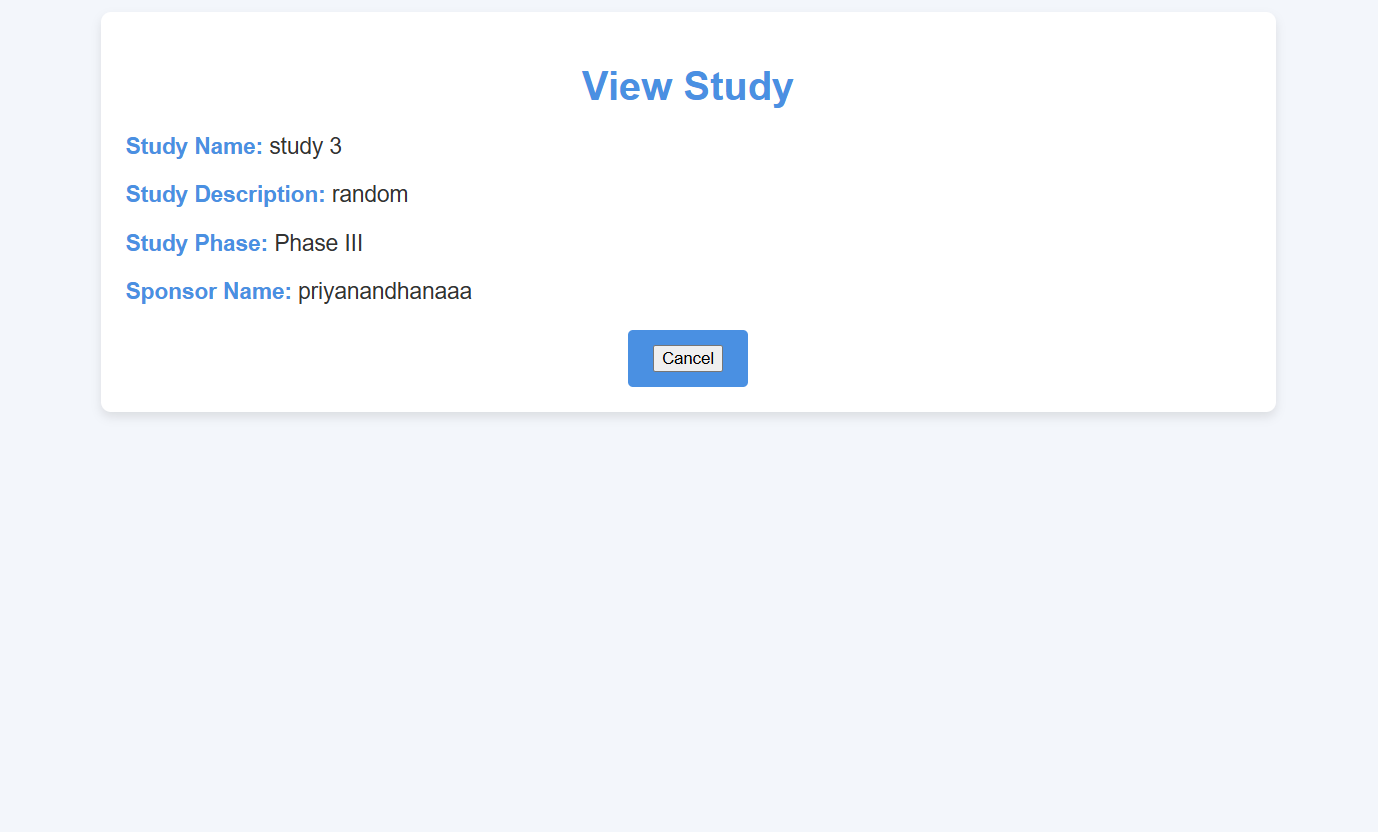
**MAIN PAGE**



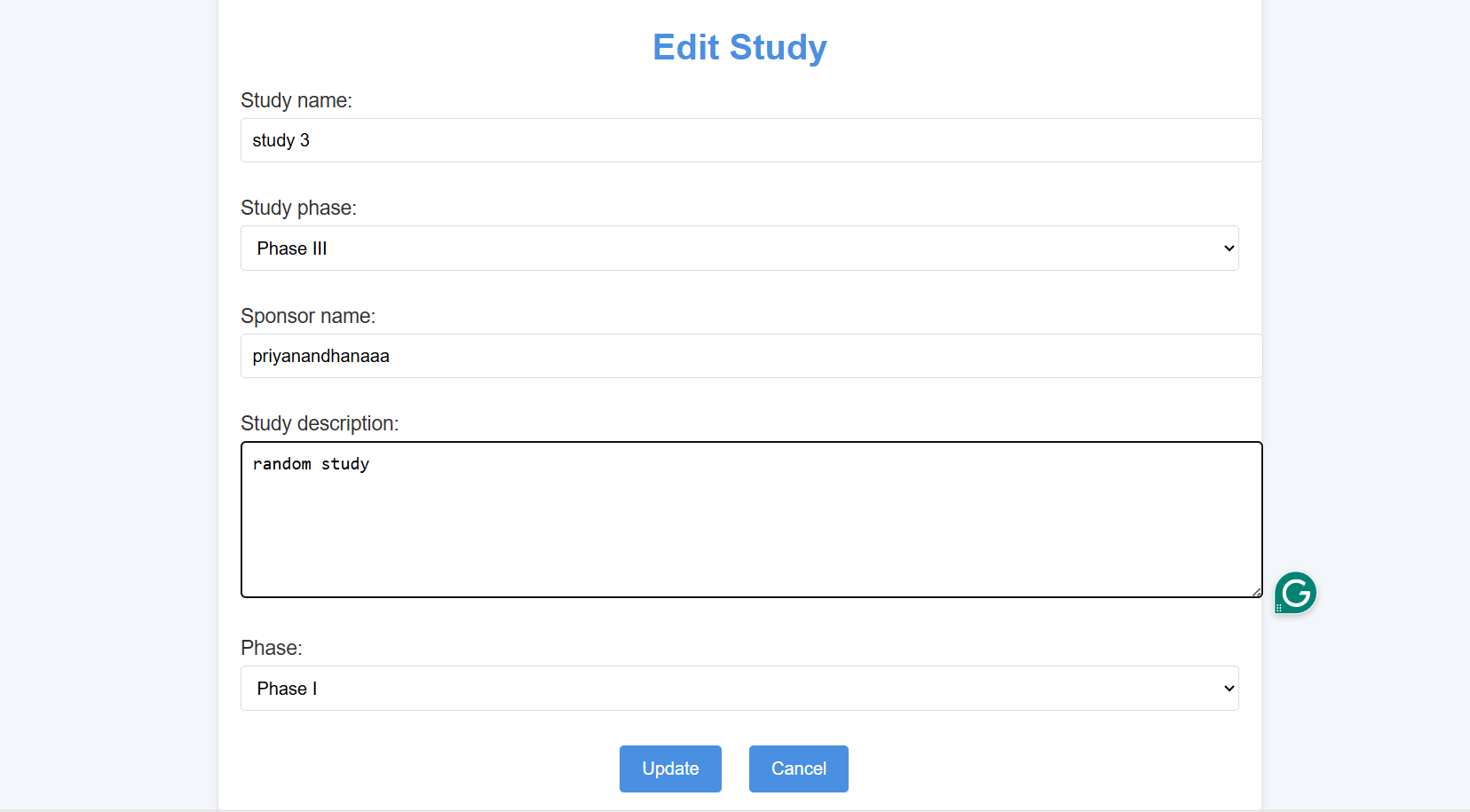
**ADD STUDY**



**ADDED**



**VIEW STUDY**



**EDIT STUDY**

**Conclusion:**

The **Study Management System** is developed as user-friendly web application to streamline the management of studies, providing features for adding, editing, viewing, and deleting records. It’s interface, responsive design, and robust functionality ensure efficient handling of study data, making it an effective tool for administrators. This project demonstrates the integration of simplicity and efficiency, offering a reliable solution for study management.

The project was started with the setup of a **Django framework**, where models were created to define the database structure for study data. Views were implemented to handle CRUD operations, and templates were designed for displaying and interacting with data.

CSS was applied to ensure an appealing and user-friendly interface. The system was thoroughly tested and debugged, resulting in a functional and efficient Study Management System. The Python exercise is successfully completed.