Python Libraries

Python libraries are pre-written code that provides a set of functionalities, making it easier to perform specific tasks. They are reusable, well-tested, and widely adopted, saving developers time and effort.

Python Collections - Container Datatypes:

1. **Purpose:** Provides specialized container datatypes that support efficient handling of data.
2. **Key Types:**
   a) List: Ordered, mutable, allows duplicates.
   b) Tuple: Ordered, immutable, allows duplicates.
   c) Set: Unordered, no duplicates, fast membership testing.
   d) Dictionary: Unordered, key-value pairs, fast lookups.
3. **Common Use:** Data manipulation, storing and accessing collections of data in web apps (like user data or API responses).

Tkinter - GUI Applications:

Purpose: Python's standard library for creating graphical user interfaces (GUIs).

Key Features:

1. Widgets: Buttons, labels, text boxes, etc.
2. Event handling: Respond to user interactions like clicks or key presses.
3. Simple layout management.

Common Use: Build desktop applications or tools for local interaction with a web app backend.

To implement the code we need to install the tkinter library:

pip install tk

Code:

```
from tkinter import
Tk, Label
 root =
Tk()
```

```
 root.title("Hello
Window")


Label(root, text="Welcome to Tkinter!").pack()
root.mainloop()
```

Output:

Requests - HTTP Requests

1. Purpose: Simplifies HTTP requests to interact with web APIs.
2. Key Features:
    a. Send GET, POST, PUT, DELETE requests easily.
    b. Handle request parameters, headers, and cookies.
    c. Simple error handling and response handling.
3. Common Use: Interact with REST APIs, download content from the web.

To implement the code we need to install the tkinter library:

pip install tk

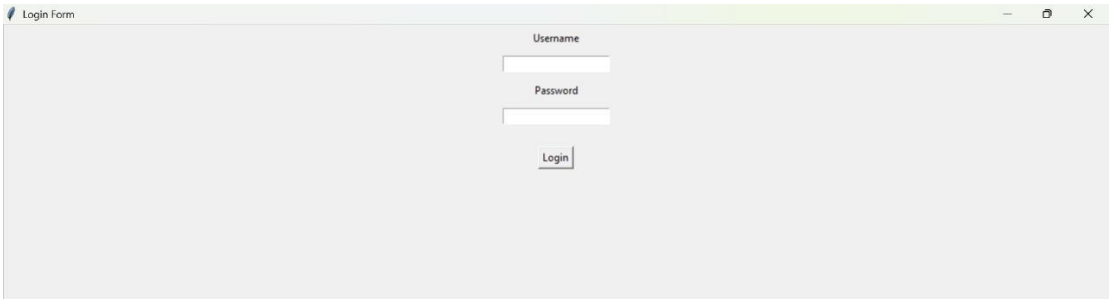
Code:

```
import tkinter as tk from tkinter
import messagebox def
validate_login():     username =
username_entry.get()     password =
password_entry.get()
     # Example credentials     if username == "user" and
password == "password":        messagebox.showinfo("Login
Success", "Login Successful!")     else:
messagebox.showerror("Login Failed", "Invalid username or
password")
```

```
# Create the main window
```

```
root = tk.Tk() root.title("Login
Form") root.geometry("1080x720")

# Create username and password labels and entry widgets
username_label = tk.Label(root, text="Username")
username_label.pack(pady=5)

username_entry = tk.Entry(root) username_entry.pack(pady=5)

password_label = tk.Label(root, text="Password")
password_label.pack(pady=5)

password_entry = tk.Entry(root, show="*") # 'show' hides the
password characters password_entry.pack(pady=5) # Create the
login button
login_button = tk.Button(root, text="Login",
command=validate_login) login_button.pack(pady=20)

# Run the Tkinter event loop root.mainloop()
```

Output:



BeautifulSoup4 - Web Scraping:

1. Purpose: Parses HTML and XML documents to extract data.
2. Key Features:
   a. Easy navigation and searching within HTML.
   b. Supports different parsers like html.parser, lxml, and html5lib.
3. Common Use: Extract data from websites for analysis, e.g., for building datadriven applications.

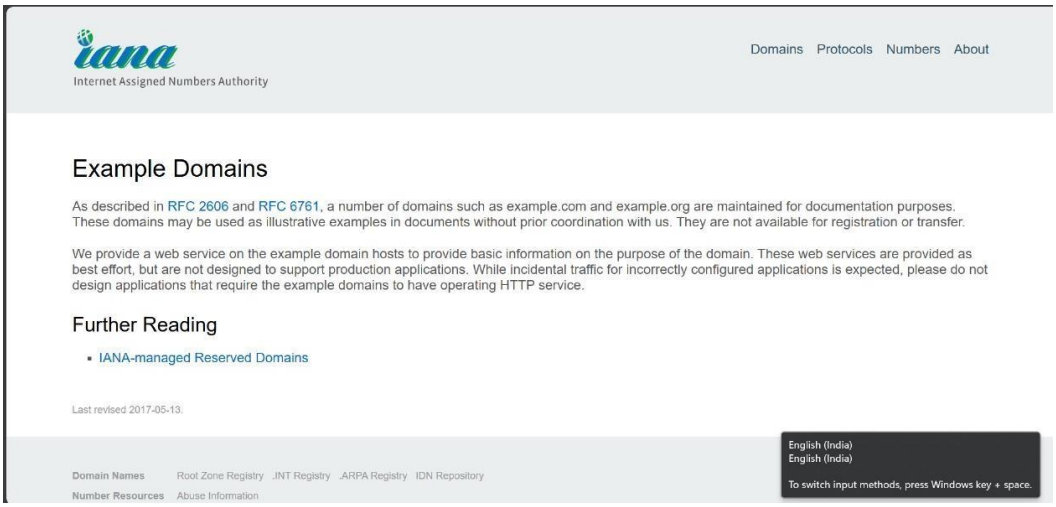
To implement the code we need to install the beautifulsoup4 library:

pip install bs4 (or) pip install beautifulsoup4

Code:

```
import requests
from bs4 import BeautifulSoup

# The URL of the website to scrape
url = 'https://example.com' # Replace with the website you want
to scrape

# Send a GET request to fetch the raw HTML content
response = requests.get(url)

# Parse the raw HTML content with BeautifulSoup soup
= BeautifulSoup(response.text, 'html.parser')
# Find the title of the webpage title
= soup.title.string
print(f"Title of the page: {title}")
# Find all headings (e.g., <h1>, <h2>, <h3>, etc.)
headings = soup.find_all(['h1', 'h2', 'h3']) for heading
in headings:      print(heading.text.strip()) # Print the
heading text
# Find all links (<a> tags) on the page
links = soup.find_all('a', href=True) for
link in links:
print(f"Link: {link['href']}")
```

Output:



CherryPy:

1. Purpose: Minimalistic web framework for building web applications.
2. Key Features:

a. Provides a simple and fast HTTP server.
b. Handles routing, cookies, sessions, and file uploads.

3. Common Use: Building web applications with a lightweight framework.

To implement the code we need to install the cherry py library:

pip install cherrypy

Code:

```
import cherrypy
 class HelloWorld(object):      @cherrypy.expose      def
index(self):          return "Hello, World! This is a CherryPy
web page."      if __name__ == '__main__':
        cherrypy.quickstart(HelloWorld())
```

Output:

Flask:

1. Purpose: Lightweight micro-framework for building web applications.
2. Key Features:
   a. Simple to learn and use, but highly extensible.
   b. Supports extensions for database integration, form handling, authentication, etc.
3. Common Use: Small to medium web applications, APIs, or microservices.

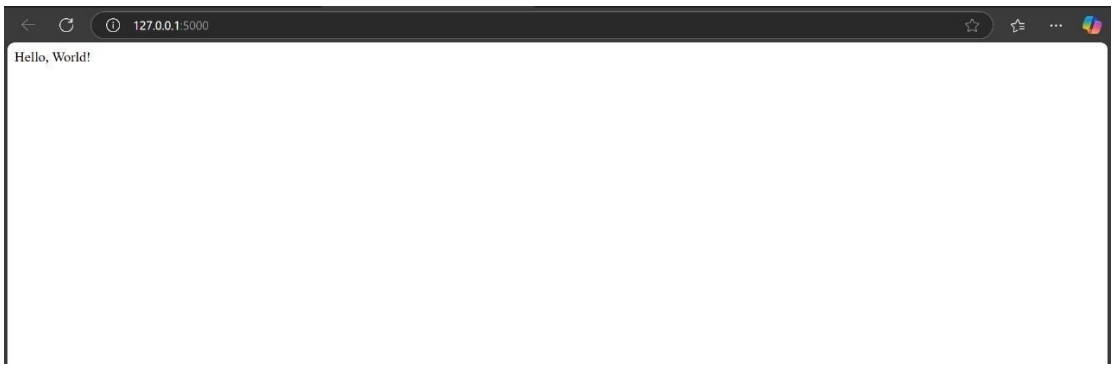
To implement the code we need to install the flask library: pip

install flask

Code:

```
from flask import Flask

# Create a Flask application instance
app = Flask(__name__)

# Define a route for the root URL (“/”)
@app.route(‘/’) def
hello_world():     return
‘Hello, World!’ # Run the
Flask application if
__name__ == ‘__main__’:
   app.run(debug=True)
```

Output:



Bottle:

1. Purpose: Simple and lightweight WSGI micro-framework.
2. Key Features:
    a. Single-file framework, minimalistic, and fast.
    b. No dependencies, supports routing, templates, and form handling.
3. Common Use: Small web applications, APIs, and prototypes.

To implement the code we need to install the bottle library:

pip install bottle

Code:

```
from bottle import route, run

@route('/') def index():     return "Hello, World! This is a
Bottle web page." run(host='localhost', port=8080)
```

Output:

SUMMARY:

1. Flask, Django, Pyramid: Popular web frameworks, each offering flexibility and scalability.
2. Scrapy, BeautifulSoup4: Specialized for web scraping and data extraction.
3. Requests, Zappa, Dash: Tools for making HTTP requests, serverless apps, and interactive data visualizations.
4. Tkinter, Bottle, CherryPy: Libraries for building lightweight desktop and web applications.

STUDENT DIRECTORY

Problem statement:

Create a directory where students can search for contact information and profiles.

Overview:

A Student Directory System is a centralized digital platform for managing student, faculty information. It provides streamlined access, efficient data management, and powerful search capabilities. By offering features like student profiles, user management, and data analytics, it empowers educational institutions to enhance communication, improve operational efficiency, and make data-driven decisions, ultimately fostering a better student experience.

What is the student directory..?

A student directory refers to a collection of information about students, typically maintained by an educational institution. However, it's very important to understand that there are legal and ethical limitations to what information can be shared.

Core Concept:

A student directory aims to provide a way to locate and identify students. It can be used for various administrative and communication purposes.

Directory Information:

Directory information" generally includes data that is not considered harmful or an invasion of privacy if disclosed. They are

1. Student's name
2. Address
3. Telephone number
4. Email address
5. Dates of attendance
6. Major field of study
7. Participation in officially recognized activities

Privacy Considerations:

Institutes need to balance the need for information sharing with the need to protect student privacy. Students should know their rights under FERPA and how to opt out of

directory information disclosures. In essence, a student directory is a tool for organizing student data, but its use is carefully regulated to safeguard privacy.

Tools to build:

We are using Django and SQLite to create the Student directory website.

Django: A Web Framework for Python

Django: Django is a free, open-source Python framework that helps developers build websites.

a) Django: Django is a **high-level Python web framework** that allows developers to build secure, scalable, and maintainable web applications **quickly and efficiently**. It follows the **Model-View-Template (MVT)** architectural pattern.

Key Features of Django:

1. Fast Development – Comes with built-in features like authentication, database management, and an admin panel**.**
2. Scalability – Suitable for small projects to enterprise-level applications.
3. Security – Protects against common security threats (SQL Injection, CSRF, XSS, etc.).
4. ORM (Object-Relational Mapper) – Allows database interaction using Python instead of SQL.
5. Built-in Admin Panel – Auto-generates an admin interface for managing data.
6. Reusable App-Developers can create modular and reusable components.

Django's MVT Architecture:

1. Model (M) – Handles database interactions (e.g., User, Booking).
2. View (V) – Manages business logic and connects models to templates.
3. Template (T) – Renders HTML pages dynamically.

Django as a web framework:

1. Django is used to build many popular websites, including Instagram, YouTube, and Spotify.
2. Django is based on the Model-View-Template (MVT) design pattern.
3. Django includes a built-in testing framework to help ensure applications are reliable and bug-free.

SQLite: SQLite is a free, open-source software library that allows users to store data in a database file on their computer. It's a relational database management system (RDBMS) that's embedded into applications and devices.

Installation of Django:

1. Create a Virtual Environment Using a virtual environment isolates your project's dependencies, preventing conflicts with other Python projects**.**
   --python -m venv venv
2. Create and Activate the virtual environment amed 'venv' venv\Scripts\activate
3. With your virtual environment activated, use pip to install Django:
   --pip install Django
4. Install a specific version of Django:
   --pip install Django==4.2.7
5. Verify the Installation: Check that Django is installed correctly:
   --python -m django –version
6. This command should output the installed Django version.
7. Create and Start new Django project:
   --django-admin startproject myproject
8. django-admin startproject myproject
9. Change into the project directory:
   --cd myproject
10. Start a Django App: Inside your project directory, create a Django app:
    --python manage.py startapp myapp
11. Run the Development Server: Start the Django development server: ***--python manage.py runserver*** **Implementation of Task:**

Creating a Student directory using Django involves several steps. From setting up your project to designing your models and views.

Project Setup:

1. Install Django
2. Create a project
3. Create an app **Define models:**

1. Create the models
2. Migrate the database **Create templates:**

Create a template within your app.

Create URL's:

In the student directory, define your URL patterns

Security:

Implement user authentication and authorization to control access to student data. Use HTTPS to encrypt data in transit. Protect against SQL injection and other security vulnerabilities.

Privacy:

Adhere to privacy regulations (e.g., GDPR, FERPA). Only store necessary student information. Provide users with control over their data.

Search and Filtering:

Add search functionality to allow users to find students by name, ID, or other criteria. Implement filtering options to narrow down the list of students.

Pagination:

Use pagination to handle large numbers of students.

Admin Interface:

Use Django's built-in admin interface to manage student data.

User Profiles:

Consider creating user profiles that students can edit themselves.

Data Import/Export:

Implement functionality to import and export student data in formats like CSV or Excel.

This outline provides a starting point. You can customize and extend it based on your specific requirements.

Connecting Views amd Urls:

Set Up URLs:

Project-level URL Configuration: from django.contrib import admin from django.urls import path, include urlpatterns = [path('admin/', admin.site.urls), path('', include('myapp1.urls')), # Include app URLs] **App-level URLConfiguration**

from django.urls import path from .views import home urlpatterns = [

path('', home, name='home'),] **Create a Sample View:** ***from***

```
django.http import HttpResponse    def home(request):
return HttpResponse("
Welcome to My Django App!
")
```

Run Migrations:

```
python manage.py migrate
```

Run the Server and Test:

```
python manage.py runserver
```

Views.py:

In Django, views.py is the file where you define functions or classes that handle requests and return responses. Views act as the logic layer of a Django web application, controlling how data is processed and which HTML templates are displayed.

Create Views:

```
from django.shortcuts import render, get_object_or_404,
redirect
from .models import Student from
django.contrib import messages from
django.http import JsonResponse from
.forms import EditStudentForm  from
django.db.models import Q def
home(request):
    return render(request,"home.html") def
admin(request):     return
render(request,"admin.html") def
studenthome(request):     return
render(request,"student.html") def login(request):
return render(request,"login.html") def
about_us(request):     return
render(request,"aboutus.html") def
add_student(request):     if request.method ==
'POST':         first_name =
request.POST['first_name']         last_name =
request.POST['last_name']         roll_no =
request.POST['roll_no']         department =
request.POST['department']         year_of_study =
request.POST['year_of_study']         email =
request.POST['email']         phone =
request.POST['phone']

try:
            Student.objects.create(
first_name=first_name,
last_name=last_name,                    roll_no=roll_no,
department=department,
year_of_study=year_of_study,
                email=email,
phone=phone
            )
            messages.success(request, 'Student added
successfully!')
```

```
            return redirect('student_list') # Redirect to the
student list page         except Exception as e:
messages.error(request, f'Error adding student:
{e}')
            return render(request, 'add.html') # Re-render the
form with error message     else:
return render(request, 'add.html')
 def
student_list(request):
    """Renders the list of all students."""     students =
Student.objects.all()     return render(request,
'studentlist.html', {'students': students})
 def
delete_student(request):
    """Handles the deletion of a student via AJAX."""
if request.method == 'POST':          roll_no =
request.POST.get('roll_no')         if roll_no:
try:                  student =
get_object_or_404(Student, roll_no=roll_no)
                student.delete()
                return JsonResponse({'status': 'success',
'message': f'Student with Roll No: {roll_no} deleted
successfully.'})             except Exception as e:
return JsonResponse({'status': 'error', 'message': f'Error
deleting student: {str(e)}'}, status=500)         else:
return JsonResponse({'status': 'error', 'message':
'Roll No not provided.'}, status=400)     else:
return JsonResponse({'status': 'error', 'message':
'Invalid request method.'}, status=405) def
edit_student(request, roll_no):     student =
get_object_or_404(Student, roll_no=roll_no)     if
request.method == 'POST':         form =
EditStudentForm(request.POST, instance=student)
# Populate form with existing instance
if form.is_valid():
            form.save()
            return redirect('student_list') # Redirect after
successful edit     else:         form =
EditStudentForm(instance=student) # Populate form for GET
request
```

```
    return render(request, 'edit.html', {'student': student,
'form': form})
 def
search_students(request):
    """Renders the search students form."""
return render(request, 'search.html')
 def
search_results(request):
    """Handles the search query and displays results."""
    searched = True     results = []     if
request.method == 'GET':         name =
request.GET.get('name', '')         rollno =
request.GET.get('rollno', '')
        department = request.GET.get('department', '')

        query = Q()
         if name:             query &=
Q(first_name__icontains=name) |
Q(last_name__icontains=name)         if rollno:
query &= Q(roll_no__icontains=rollno)         if
department:             query &=
Q(department__icontains=department)

        results = Student.objects.filter(query)

    context = {
        'search_results': results,
        'searched': searched,
    }
    return render(request, 'search.html', context)
 def signup_view(request):     if
request.method == 'POST':         #
Process the form data here
        first_name = request.POST.get('firstName')
last_name = request.POST.get('lastName')
email = request.POST.get('email')         password
= request.POST.get('password')
        confirm_password = request.POST.get('confirmPassword')
birthday = request.POST.get('birthday')         location =
request.POST.get('location')
        terms = request.POST.get('terms')  # This will be 'on'
if checked
         if password != confirm_password:
messages.error(request, "Passwords do not match!")
```

```
            return render(request, 'signin.html')

        # In a real application, you would:
        # 1. Validate the data (e.g., using Django forms).
        # 2. Create a new user in your database.
        # 3. Potentially send a verification email.
        # 4. Redirect the user to a success page or log them
in.
        messages.success(request, "Account created
successfully!")
        return redirect('signup_success') # You'd need to
define this URL

else:
        # If it's a GET request, just display the form
return render(request, 'cprofile.html')
 def signup_success_view(request):     return
render(request, 'cprofile.html') # Create this template
from django.shortcuts import render
from django.contrib.auth.decorators import login_required from
.models import UserProfile

@login_required def view_profile(request):
try:         user_profile =
request.user.profile     except
UserProfile.DoesNotExist:
        # Handle the case where a profile hasn't been created
yet
        user_profile = UserProfile(user=request.user)
user_profile.save()  # Or redirect to a create profile page
    context = {'user_profile': user_profile}
return render(request, 'profile.html', context)

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

@login_required def
edit_profile_view(request):     try:
user_profile = request.user.profile
except UserProfile.DoesNotExist:
```

```
        user_profile = UserProfile(user=request.user)
    if request.method ==
'POST':
        user = request.user
        user.first_name = request.POST.get('firstName')
user.last_name = request.POST.get('lastName')
user.save()

        user_profile.birthday = request.POST.get('birthday')
user_profile.location = request.POST.get('location')
user_profile.save()

        messages.success(request, 'Profile updated
successfully!')
        return redirect('view_profile')
else:         context = {'user_profile':
user_profile}
        return render(request, 'edit_profile.html', context)
 def reports_page(request):     return
render(request, 'reports.html')
```

Apps.py:

Creating urls.py in a Django App:

Each Django app should have its own urls.py file to define app-specific routes.

Steps to Create urls.py in a Django App

1. Inside your Django app folder (myapp1), create a file named urls.py.
2. Define URL patterns to map URLs to views.

```
from django.apps import AppConfig
 class DirectConfig(AppConfig):     default_auto_field =
'django.db.models.BigAutoField'
    name = 'direct'
```

Urls.py:

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

urlpatterns=[
    path('',views.home,name='home'),
path('login/',views.login,name='logo'),
path('admin.html/',views.admin,name='admin'),
path('about_us/',views.about_us,name='about'),
    path('studenthomepage/',views.studenthome,name='stuhome'),
path('add.html/',views.add_student,name='add'),
    path('students/', views.student_list, name='student_list'),
path('student/delete/', views.delete_student,
name='delete_student'),
    path('student/edit/<str:roll_no>/', views.edit_student,
name='edit_student'),
     path('search/', views.search_students,
name='search_students'),
    path('search/results/', views.search_results,
name='search_results'),
    path('signup/', views.signup_view, name='signup'),
path('signup/success/', views.signup_success_view,
name='signup_success'),
    path('profile/', views.view_profile, name='view_profile'),
path('edit_profile/', views.edit_profile_view,
name='edit_profile'),
    path('reports/', views.reports_page, name='reports_page'),
]
```

Templates:

Being a web framework, Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the static parts of the desired HTML output and some special syntax describing how dynamic content will be inserted. We have included some templates.

Templates are such as the Dashboard,Home page, the Login page, the Signin page, and the Admin page

.**Dashboard page:**

Code:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta     name="viewport"     content="width=device-width,
initialscale=1.0">
    <title>Dashboard</title>
    <link rel="icon" href="jntugv_logo.jpg" type="image/jpg">
    <link rel="stylesheet" type="text/css" href="{% static
'dash.css'% }">

</head>
<body>
   <div class="sd">
     <div class="logo"><img
src="https://jntugvcev.edu.in/wpcontent/uploads/2022/07/logo-
min.jpeg" class="logo"></div>
    <div class="hg">
    <h1><b>Jawaharlal Nehru Technology University Gurajada
Vizianagaram</b></h1>
    <h2>Department Of Information Technology</h2>
    <h2><b>DIRECTORY</b></h2>
    <h3>A student directory is essentially a collection of
information about students, typically within an educational
institution. However, it's important to understand that the
contents of a student directory, and who has access to it, are
heavily influenced by privacy considerations.</h3>
    <h3>A student directory aims to provide a way to locate
and, in some cases, contact students.
    </h3>
</div><div class="butt">
    <button><a href="hompage.html">Get Started </a></button>
</div>
</div>
```

```
</body>
```

```
</html>
```

Output:

Home page:

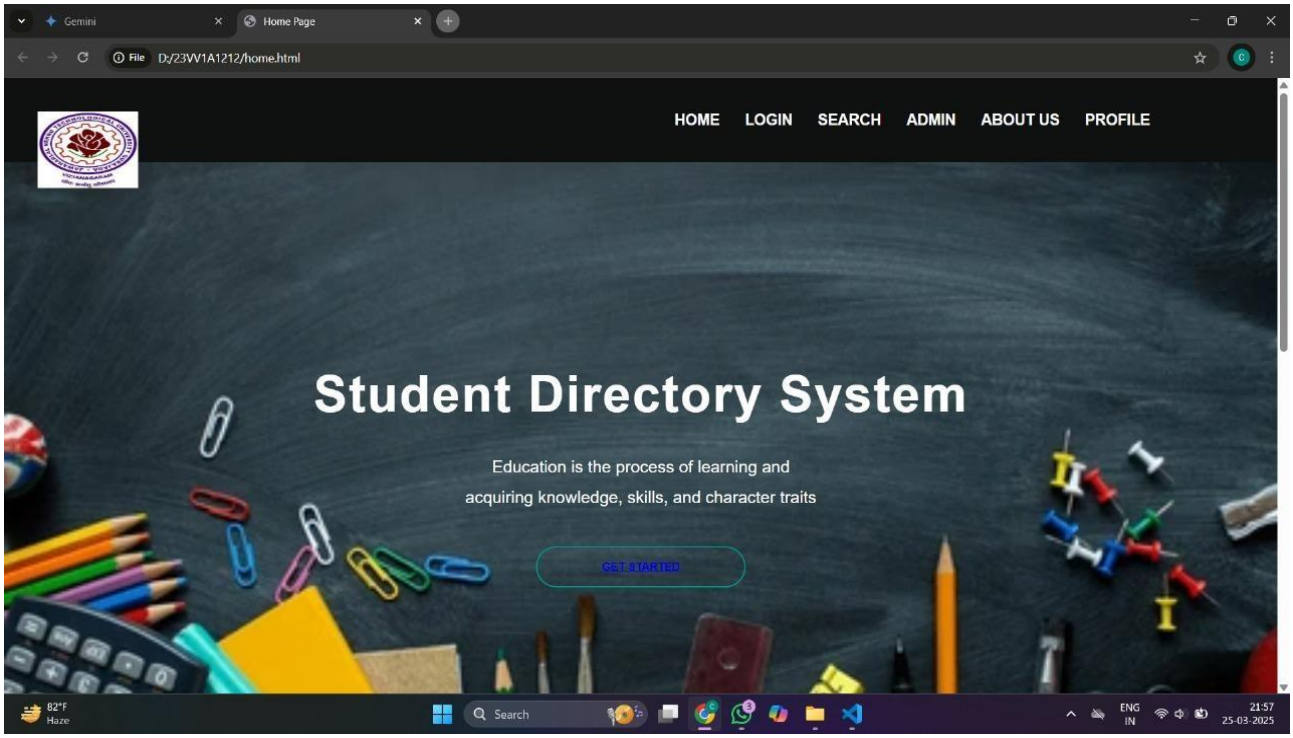
"Homepage" typically refers to the main or initial page of a website or application.

Code:

```
<!DOCTYPE html>
{% load static %}
<html>
<head>
    <title>Home Page</title>
    <link rel="stylesheet" href="{% static 'homepage.css' %}">
</head>
<body>
    <div class="header-txt">
    </div>
    <div class="container">
        <div class="navbar">
            <img
src="https://jntugvcev.edu.in/wpcontent/uploads/2022/07/logo-
min.jpeg" class="logo">
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="loginpage.html">Login</a></li>
                <li><a href="passforgot.html">Sign in</a></li>
                <li><a href="#">Search</a></li>
```

```
                <li><a href="adminpage.html">Admin</a></li>
            </ul>
        </div>
        <div class="content">
            <h1>Student Directory System</h1> <!-- The heading text -
->
            <p>Education is the process of learning and
<br>acquiring knowledge, skills, and character traits</p>
            <div>
                <button type="button"><a
href="passforgot.html"><span></span>Get Started</a></button>
            </div>
        </div>
    </div>
</body>
</html>
```

Output:



Login page:

A login page is a crucial component of many websites and applications, serving as the gateway for users to access secure or personalized content.

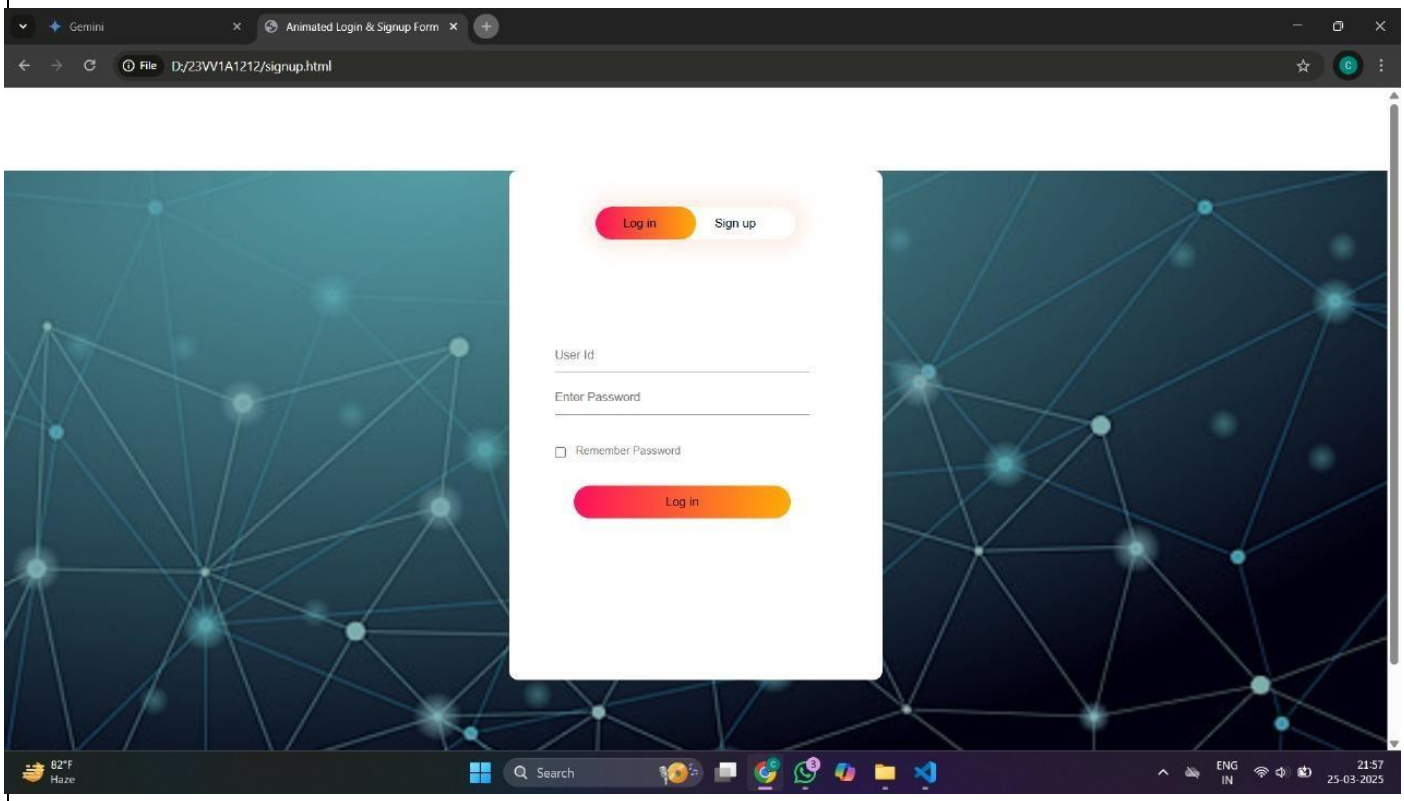
Code:

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>JNTU-GV Login</title>
  <link rel="stylesheet" href="{% static 'myglobal.css' %}">
</head>
<body>
<!-- partial:index.partial.html -->
<div class="wrapper">
  <div class="login-box">
    <form action="" method="POST">
      {% csrf_token %}
      <h2>Login</h2>
      <div class="input-box">
        <span class="icon">
          <ion-icon name="mail"></ion-icon>
        </span>
        <input type="email" name="email" required>
        <label>Email</label>
      </div>
      <div class="input-box">
        <input  type="password"  id="password"  name="password"
required>
        <label>Password</label>
        <span  class="icon"  id="togglePassword"  style="cursor:
pointer;">
          <ion-icon name="eye"></ion-icon>
        </span>
      </div>
      <div class="remember-forgot">
        <label><input type="checkbox" name="remember"> Remember
me</label>
        <a href="passforgot.html">Forgot Password?</a>
      </div>
      <button type="submit">Login</button>
      <div class="register-link">
<p>Don't have an account? <a
href="rest.html">Register</a></p>
      </div>
```

```
</form>
```

```
  </div>
</div>
<!-- partial -->
<script
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.js
"></script>
<script
src="https://unpkg.com/ionicons@5.5.2/dist/ionicons/ionicons.es
m.js"></script>
  <script>
    // JavaScript to toggle password visibility
const togglePassword =
document.querySelector('#togglePassword');
    const password = document.querySelector('#password');
togglePassword.addEventListener('click', function () {
      // Toggle the type attribute
      const type = password.getAttribute('type') === 'password'
? 'text': 'password';
      password.setAttribute('type', type);
      // Toggle the eye icon
      this.querySelector('ion-icon').setAttribute('name', type
=== 'password' ? 'eye': 'eye-off');
    });
  </script>
</body>
</html>
```

Output:



Signin page:

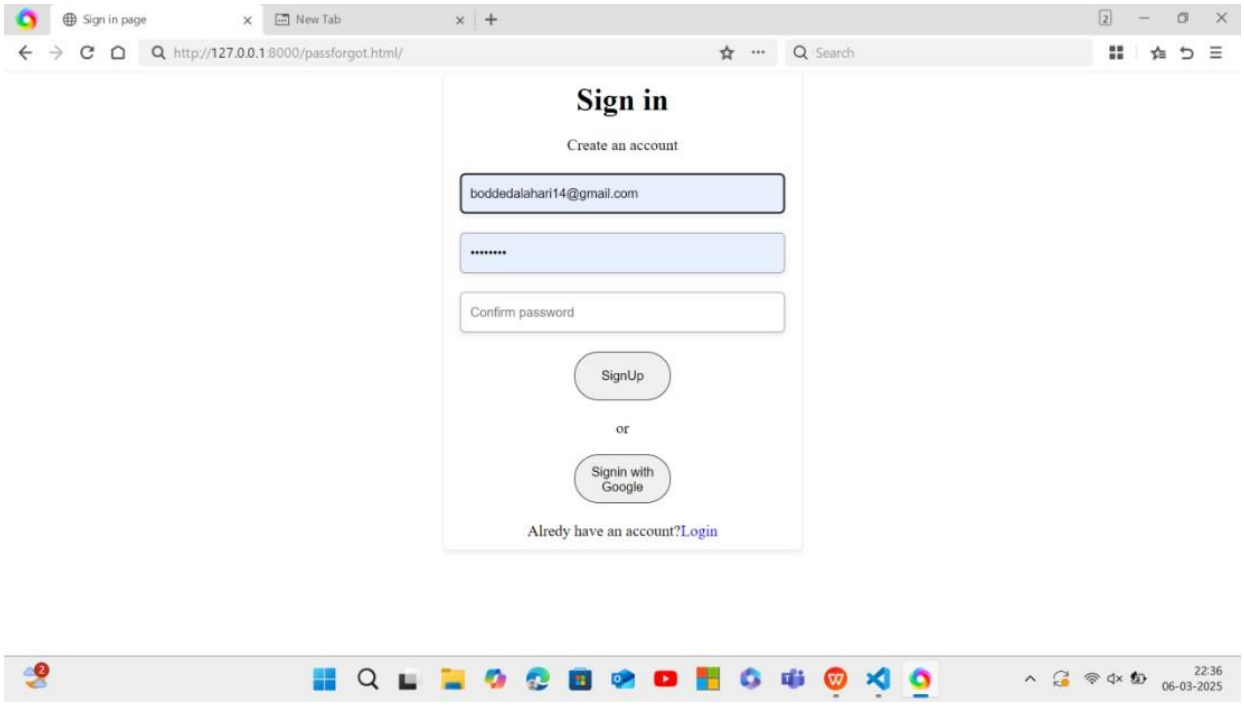
Both "login" and "sign in" refer to the process of a user entering their credentials to gain access to a system, website, or application.

Code:

```
<!DOCTYPE html>
{%load static%}
<html>
<head>
    <title>Sign in page</title>
    <meta name="Saroj" content="author of this page">
    <meta charset="utf-8">
    <link rel="stylesheet" href="{% static 'rest.css' %}">
</head>
<body>
    <div class="container">
        <h1>Sign in</h1>
        <p class="para">Create an account</p>        <input
type="text" placeholder="Email or phone" required><br>
        <input type="password" placeholder="Password" required><br>
```

```
        <input type="password" placeholder="Confirm password"
required><br>
        <button type="button">SignUp</button><br>
        <p>or</p><br>
        <button type="button">Signin with Google </button><br>
        <p> Already have an account?<a
href="login.html">Login</a></p>
    </div>
</body>
</html>
```

Output:



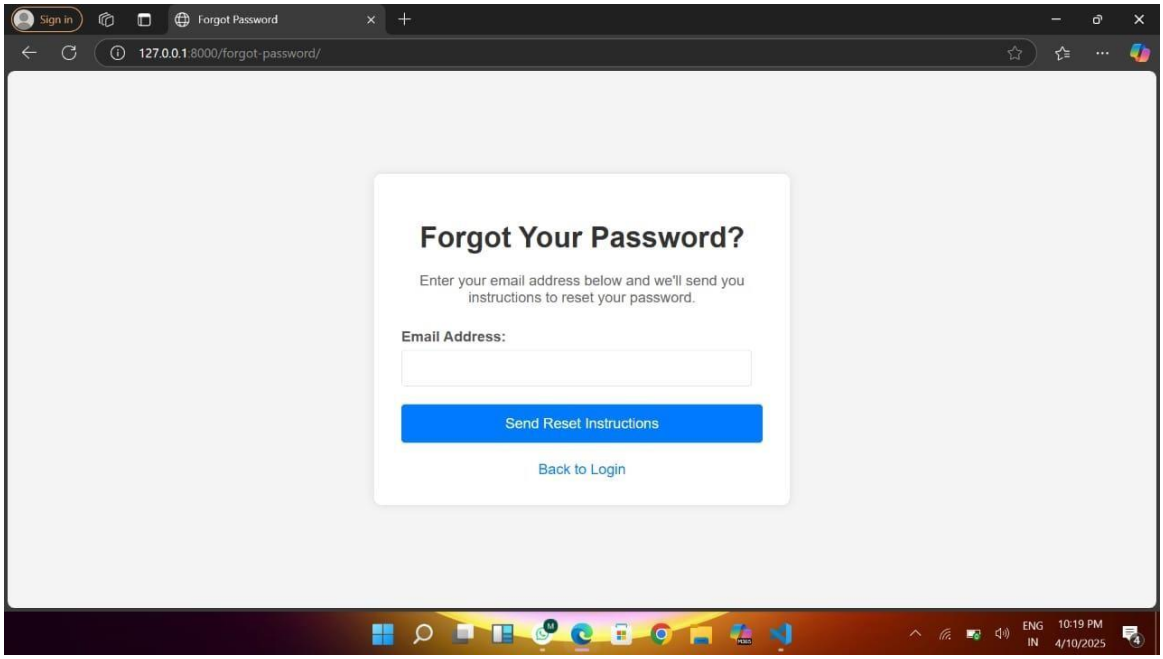
Forgot password page:

Code:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta      name="viewport"      content="width=device-width,
initialscale=1.0">
    <link rel="stylesheet" href="{% static 'style1.css'%}">
<title>JNTU-GV</title>
```

```
</head>
<body>
    <!-- Forgot Password Form -->
<form id="forgot-password-form">
    <h2>Forgot Password</h2>
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <button type="submit">
        <a href="rest.html">Send Reset Link</a>
    </button>
    <p id="error-message"></p>


</form>
</form>
</body>
</html>
```

Output:



Resetpass page:

Code:

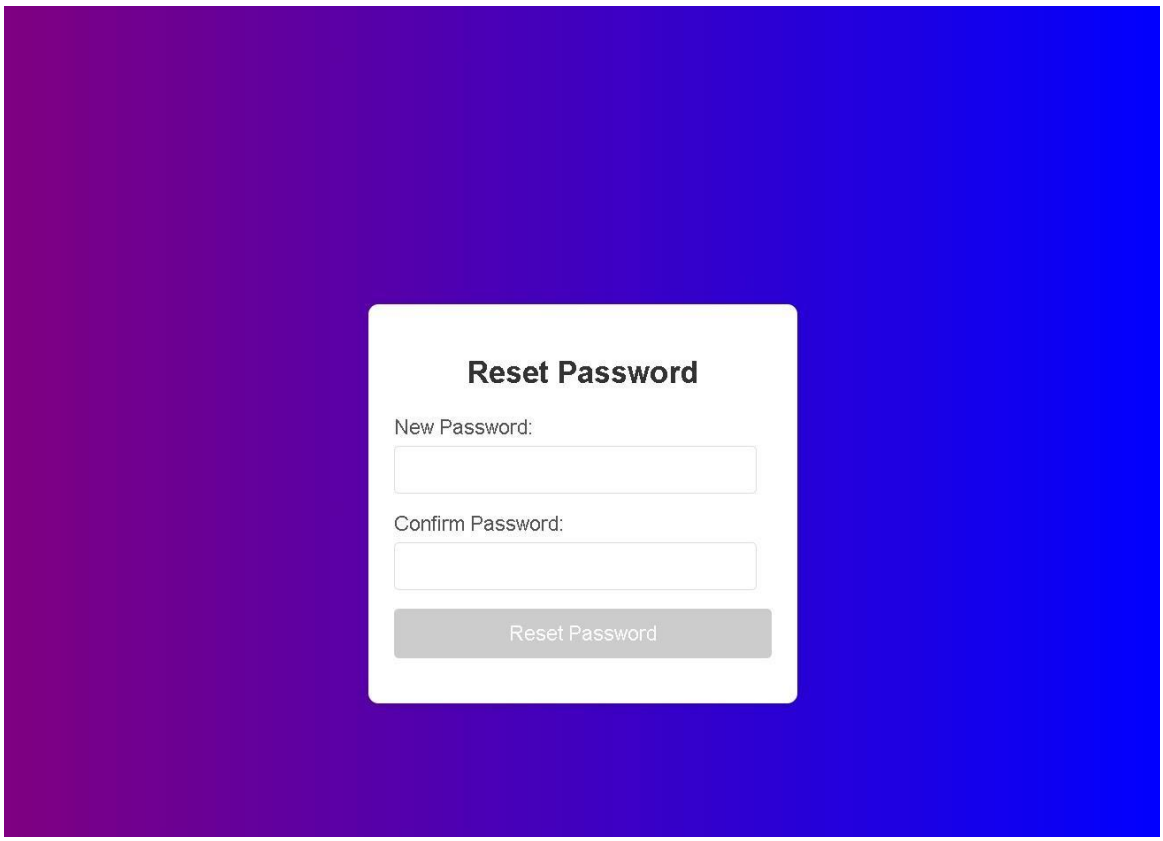
```
{% load static %}
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta      name="viewport"      content="width=device-width,
initialscale=1.0">
    <title>JNTU-GV</title>
    <link rel="stylesheet" href="{% static 'style1.css'%}">
</head>
<body>
  <!-- Reset Password Form -->
   <div class="input-box">
<form id="reset-password-form">
    <h2>Reset Password</h2>
    <label for="password">New Password:</label>
    <input    type="password"    id="password"    name="password"
required>
    <label for="confirm-password">Confirm Password:</label>
<input type="password" id="confirm-password" name="confirm-
password" required>
    <button type="submit" id="submit-btn" disabled>Reset
Password</button>
    <p id="error-message"></p>
</form>
</div>

<script>
    // Get the password and confirm password input fields
const passwordInput = document.getElementById('password');
const confirmPasswordInput =
document.getElementById('confirm-password');
    const submitBtn = document.getElementById('submit-btn');
const errorMessage = document.getElementById('errormessage');

    // Add event listeners to the input fields
    passwordInput.addEventListener('input', checkPasswords);
confirmPasswordInput.addEventListener('input', checkPasswords);

    // Function to check if passwords match
function checkPasswords() {
        if (passwordInput.value === confirmPasswordInput.value)
{
            // Enable the submit button if passwords match
            submitBtn.disabled = false;
errorMessage.textContent = '';
        } else {
            // Disable the submit button and display an error
message if passwords do not match
submitBtn.disabled = true;
            errorMessage.textContent = 'Passwords do not match';
```

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

Output:



Aboutus.html

Code

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initialscale=1.0">
    <title>About Us - Student Directory</title>
```

```
<link rel="stylesheet" href="{% static 'about.css' %}">
```

```
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.0.0/css/al
l.min.css" integrity="sha512-
9usAa10IRO0HhonpyAIVpjrylPvoDwiPUiKdWk5t3PyolY1cOd4DSE0Ga+ri4Au
TroPR5aQvXU9xC6qOPnzFeg==" crossorigin="anonymous"
referrerpolicy="no-referrer" />
</head>
<body>
    <div class="page-wrapper">
        <header class="hero">
            <div class="hero-content">
                <h1>About Our Student Directory</h1>
                <p class="subtitle">Connecting Students,
Simplifying Information.</p>
            </div>
        </header>

        <main class="main-content">
            <section class="section mission-section">
                <div class="container">
                    <div class="mission-image">
                        <img
src="https://img.freepik.com/freephoto/about-us-information-service-
sharing-join-concept_53876-
124056.jpg?t=st=1743609736~exp=1743613336~hmac=e4cf95854bbba465
8f70054396d773ae9170c4a1e19d52555bda90c922b4f9df&w=1480" alt="Our
Mission">
                    </div>
                    <div class="mission-text">
                        <h2 class="section-title"><i class="fas
fa-bullseye"></i> Our Mission</h2>
                        <p>At the heart of the Student
Directory is a clear mission: to empower students by providing a
central hub for information and connections. We believe a well-
informed and connected student body is a stronger one.</p>
<p>Our goal is to simplify the process of finding essential student
details (with a strong focus on privacy), understanding academic
groups, and fostering a collaborative learning environment. We strive
to make student life a little easier and more connected.</p>
                    </div>
                </div>
            </section>

            <section class="section features-section bg-light">
                <div class="container">
```

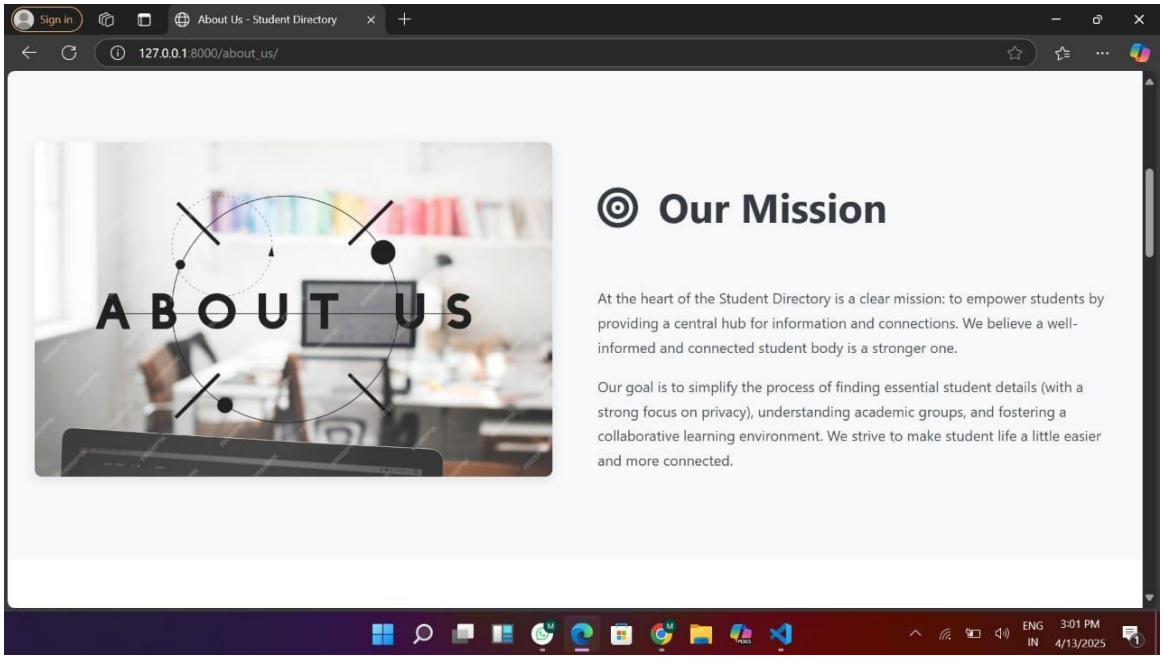
```
                    <h2 class="section-title"><i class="fas
falightbulb"></i> Key Features</h2>
                    <div class="feature-grid">
                        <div class="feature-card">
```

```
                        <i class="fas fa-search fa-2x"></i>
                        <h3>Effortless Search</h3>
<p>Quickly find students by name, roll number, course, and
more.</p>
                    </div>
                    <div class="feature-card">
                        <i class="fas fa-user-circle fa-
2x"></i>
                        <h3>Student Profiles</h3>
<p>Access relevant information about students (privacy
protected).</p>
                    </div>
                    <div class="feature-card">
                        <i class="fas fa-list-alt fa-
2x"></i>
                        <h3>Organized Data</h3>
                        <p>Information presented in a clear
and easy-to-understand format.</p>
                    </div>
                    <div class="feature-card">
                        <i class="fas fa-users fa-2x"></i>
                        <h3>Connect & Collaborate</h3>
<p>Facilitating connections for study groups and
projects.</p>                        </div>
                </div>
            </div>
        </section>

        <section class="section team-section">
            <div class="container">
                <h2 class="section-title"><i class="fas
fausers"></i> Meet Our Team</h2>
                <div class="team-grid">
                    <div class="team-member-card">
<img src="{% static 'images/teammember-1.jpg' %}" alt="[Team
Member 1 Name]">
                        <h3>[Team Member 1 Name]</h3>
<p class="role"><i class="fas fabriefcase"></i> [Your
Role/Team Member 1 Role]</p>
                        <p class="bio">[A short, engaging
bio.]</p>
                    </div>
                    <div class="team-member-card">
<img src="{% static 'images/teammember-2.jpg' %}" alt="[Team
Member 2 Name]">
```

```
                                <h3>[Team Member 2 Name]</h3>
<p class="role"><i class="fas fa-
briefcase"></i> [Team Member 2 Role]</p>
```

```
                            <p class="bio">[A short, engaging
bio.]</p>
                        </div>
                    </div>
                </div>
            </section>

            <section class="section contact-section bg-accent">
                <div class="container">
                    <h2 class="section-title inverted"><i
class="fas fa-envelope"></i> Connect With Us</h2>
                    <p class="inverted">We're always eager to
hear your feedback and suggestions!</p>
<ul class="contact-info">
                        <li><i class="fas fa-envelope"></i>
Email: <a href="mailto:[Your Contact Email]">[Your Contact
Email]</a></li>
                        <li><i class="fab fa-github"></i> <a
href="[Your GitHub Link]" target="_blank">GitHub</a>
(Optional)</li>
                    </ul>
                </div>
            </section>
        </main>

        <footer class="site-footer">
            <div class="container">
                <p>&copy; 2025 Student Directory Project.
Building connections.</p>
            </div>
        </footer>
    </div>
</body>
</html>
```

Output:



add.html

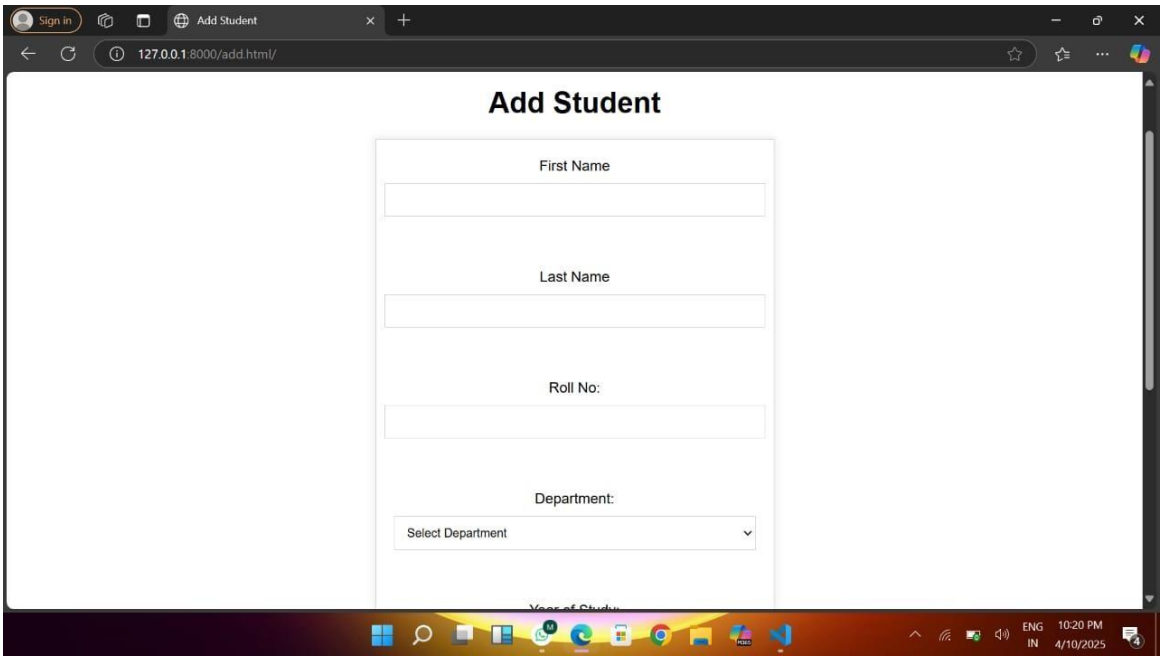
Code:

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta     name="viewport"     content="width=device-width,
initialscale=1.0">
    <title>Add Student</title>
    <link rel="stylesheet" href="{% static 'add.css' %}">
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="{% url 'admin'
%}">Dashboard</a></li>
                <li><a href="">Students</a></li>
<li><a href="">Reports</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h1>Add Student</h1>
        <form id="add-student-form" method="post">
            {% csrf_token %}
            <label for="first_name"> First Name</label>
```

```
            <input type="text" id="first_name"
name="first_name" required>
            <br><br>
            <label    for="last_name">    Last    Name</label>
<input type="text" id="last_name" name="last_name" required>
            <br><br>
            <label     for="roll_no">Roll     No:</label>
<input type="text" id="roll_no" name="roll_no" required>
            <br><br>
            <label for="department">Department:</label>
            <select id="department" name="department"
required>
                <option value="">Select Department</option>
                <option value="Computer Science">Computer
Science</option>
                <option value="Information
Technology">Information Technology</option>
                <option value="ECE">ECE</option>
                <option value="EEE">EEE</option>
                <option
value="Mechanical">Mechanical</option>
                <option value="Civil">Civil</option>
                <option
value="Metallurgy">Metallurgy</option>
            </select>
            <br><br>
            <label        for="year_of_study">Year        of
Study:</label>                    <select  id="year_of_study"
name="year_of_study" required>
                <option value="">Select Year of
Study</option>
                <option value="1">1</option>
                <option value="2">2</option>
                <option value="3">3</option>
                <option value="4">4</option>
            </select>
            <br><br>
            <label for="email">Email:</label>
<input type="email" id="email" name="email" required>
            <br><br>
            <label for="phone">Phone:</label>
<input type="text" id="phone" name="phone" required>
            <br><br>
```

```
            <button type="submit" id="add-student-button">Add
Student</button>
        </form>
    </main>
</body>
</html>
```

Output:



edit_profile.html

Code:

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta     name="viewport"     content="width=device-width,
initialscale=1.0">
    <title>Sign Up</title>
    <style>
body {
            font-family: sans-serif;
background-color: #f4f4f4;
display: flex;             justify-
content: center;             align-
items: center;             min-height:
100vh;             margin: 0;
        }

        .signup-container {
background-color: #fff;             padding:
30px;
            border-radius: 8px;             box-shadow: 0 0
10px rgba(0, 0, 0, 0.1);
            width: 400px;             max-
width: 90%;
        }          h2
{
            text-align: center;
margin-bottom: 20px;             color: #333;
        }

        .form-group {             margin-
bottom: 15px;
        }          label {
display: block;             margin-
bottom: 5px;             color: #555;
font-size: 0.9em;
        }
```

```
        input[type="text"],
input[type="email"],
input[type="password"],
input[type="date"] {             width:
calc(100% - 12px);             padding: 8px;
border: 1px solid #ddd;              border-
radius: 4px;             box-sizing: border-
box;             font-size: 1em;
        }

        .checkbox-group {
display: flex;              align-items:
center;             margin-bottom: 15px;
        }

        .checkbox-group input[type="checkbox"] {
            margin-right: 8px;
        }          button
{
            background-color: #007bff;
color: white;             padding: 10px 15px;
```

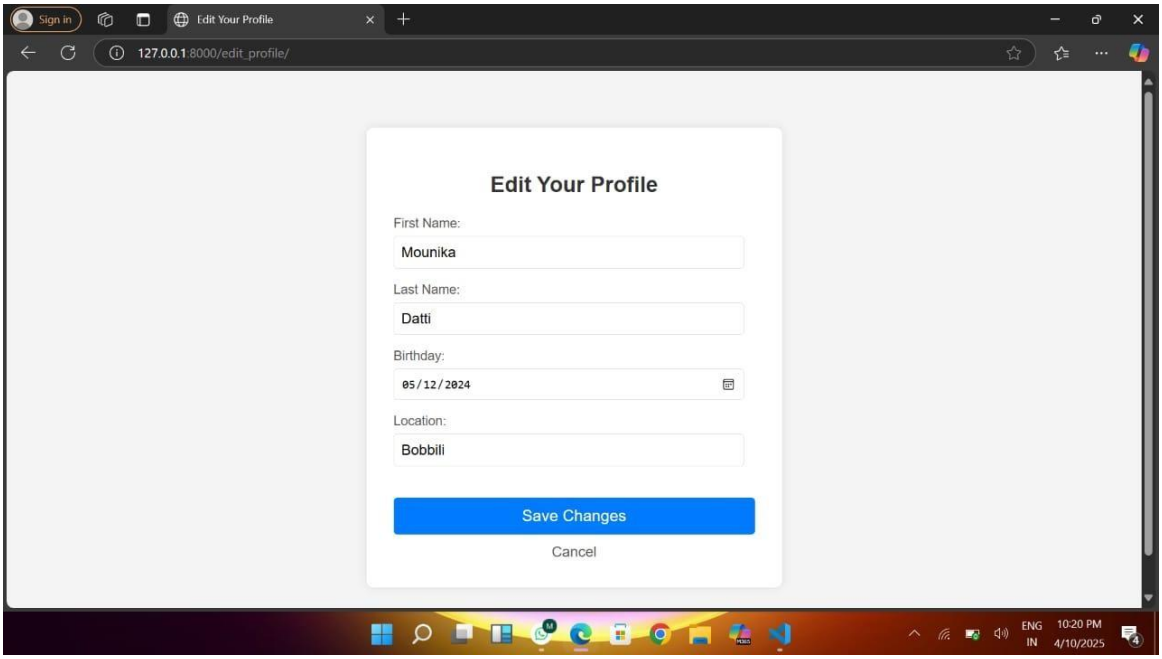
```
            border: none;
border-radius: 4px;
cursor: pointer;            font-
size: 1.1em;            width:
100%;
        }

        button:hover {
            background-color: #0056b3;
        }

        .error-message {
color: red;             font-
size: 0.8em;
margin-top: 5px;
        }
    </style>
</head>
<body>
    <div class="signup-container">
        <h2>Create Your Profile</h2>
        <form id="signupForm" method="post">
            {% csrf_token %}
            <div class="form-group">
                <label for="firstName">First Name:</label>
                <input type="text" id="firstName"
name="firstName" required>
            </div>
            <div class="form-group">
                <label for="lastName">Last Name:</label>
                <input type="text" id="lastName"
name="lastName" required>
            </div>
            <div class="form-group">
                <label  for="email">Email  Address:</label>
<input type="email" id="email" name="email" required>
            </div>
            <div class="form-group">
                <label for="password">Password:</label>
<input type="password" id="password" name="password"
required>
            </div>
            <div class="form-group">
                <label for="confirmPassword">Confirm
Password:</label>
```

```
                <input type="password" id="confirmPassword"
name="confirmPassword" required>
```

```
                <p id="passwordMatch" class="error-
message"></p>
</div>
            <div class="form-group">
                <label
for="birthday">Birthday:</label>
<input type="date" id="birthday" name="birthday">
</div>
            <div class="form-group">
                <label
for="location">Location:</label>
<input type="text" id="location" name="location">
</div>
            <div class="checkbox-group">
                <input type="checkbox" id="terms"
name="terms" required>
                <label for="terms">I agree to the <a
href="#">Terms of Service</a> and <a href="#">Privacy
Policy</a></label>
            </div>
            <button type="submit">Sign Up</button>
        </form>
    </div>

    <script>
        const passwordInput =
document.getElementById('password');
const confirmPasswordInput =
```

```
document.getElementById('confirmPassword');
const passwordMatchMessage =
document.getElementById('passwordMatch');
const signupForm =
document.getElementById('signupForm');

        confirmPasswordInput.addEventListener('keyup', () => {
if (passwordInput.value !== confirmPasswordInput.value) {
                passwordMatchMessage.textContent = "Passwords
do not match!";             } else {
                passwordMatchMessage.textContent = "";
            }
        });

        signupForm.addEventListener('submit', (event) => {
if (passwordInput.value !== confirmPasswordInput.value) {
              event.preventDefault(); // Prevent form
submission
                alert("Please make sure your passwords
match!");
            } else {
                // In a real scenario, you'd send the form
data to a server here
                alert("Account created successfully! (This is a
demo)");             }
        });
    </script>
</body>
</html>
```

Output:



edit.html

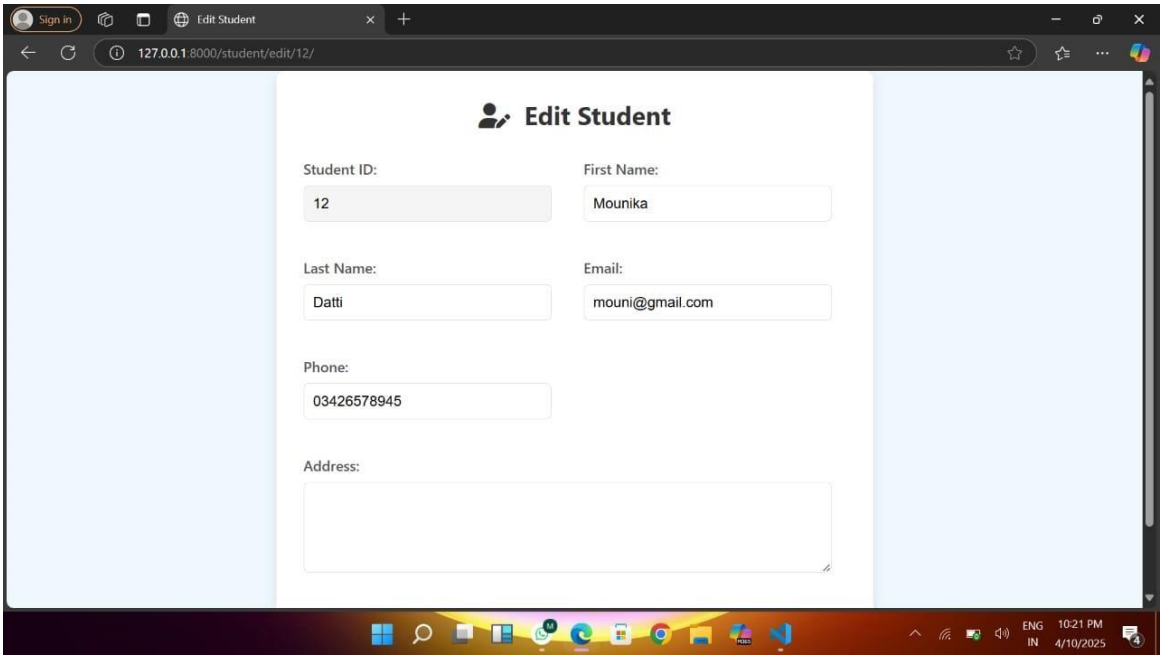
Code:

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initialscale=1.0">
    <title>Edit Student</title>
    <link rel="stylesheet" href="{% static 'edit.css' %}">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.0.0
/css/all.min.css" integrity="sha512-
9usAa10IRO0HhonpyAIVpjrylPvoDwiPUiKdWk5t3PyolY1cOd4DSE0Ga+ri4A
u
TroPR5aQvXU9xC6qOPnzFeg==" crossorigin="anonymous"
referrerpolicy="no-referrer" />
</head>
<body>
    <div class="container">
        <div class="header">
            <h1><i class="fas fa-user-edit"></i> Edit
Student</h1>
        </div>
```

```
        <form id="editStudentForm" method="post">
            {% csrf_token %}
            <div class="form-grid">
                <div class="form-group">
                    <label           for="studentId">Student
ID:</label>                                  <input  type="text"
id="studentId"  name="studentId"  value="{{  student.roll_no
}}" required readonly>
                </div>
                <div class="form-group">
                    <label for="firstName">First
Name:</label>                       <input type="text"
id="firstName" name="firstName" value="{{
student.first_name }}" required>
                </div>
                <div class="form-group">
                    <label for="lastName">Last
Name:</label>                       <input type="text"
id="lastName" name="lastName" value="{{ student.last_name
}}" required>
                </div>
                <div class="form-group">
                    <label for="email">Email:</label>
<input type="email" id="email" name="email" value="{{
student.email }}" required>
                </div>
                <div class="form-group">
                    <label for="phone">Phone:</label>
<input type="tel" id="phone" name="phone" value="{{
student.phone }}">
                </div>
                <div class="form-group address-group">
                    <label for="address">Address:</label>
<textarea id="address" name="address">{{ student.address
}}</textarea>
                </div>
            </div>
            <div class="button-group">
```

```
                <button type="submit" class="save-button"><i
class="fas fa-save"></i> Save Changes</button>
                <button type="button" id="cancelButton"
class="cancel-button"><i class="fas fa-times-circle"></i>
Cancel</button>
            </div>
        </form>
    </div>
<script>
        document.addEventListener('DOMContentLoaded',
function() {
            document.getElementById('cancelButton').addEventLis
tener('click', function(){
                window.location.href = "{% url 'student_list'
%}";
            });

            document.getElementById('editStudentForm').addEvent
Listener('submit', function(event) {
                // The form submission will now be handled by
Django
                // The default action will send a POST request
to the server            });
        });
    </script>
</body>
</html>
```

Output:

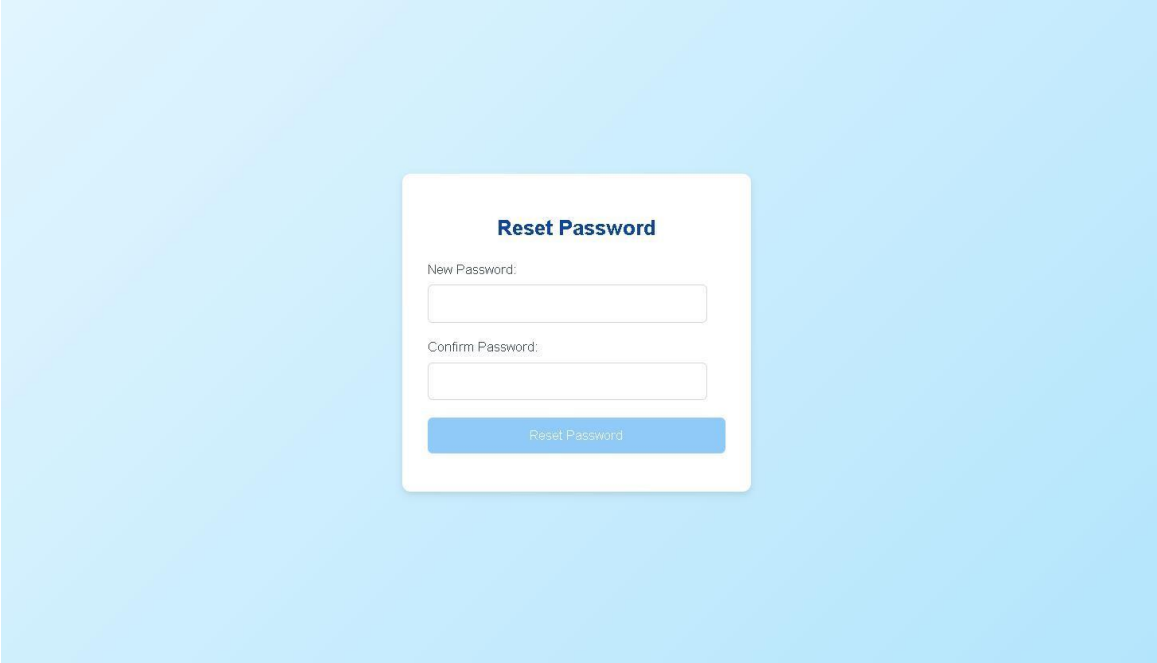


editprofile.html Code:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">     <title>JNTU-GV</title>
    <link rel="stylesheet" href="{% static
'rest.css'%}">
</head>
<body>
  <!-- Reset Password Form -->
   <div class="input-box">
<form id="reset-password-form">
    <h2>Reset Password</h2>
    <label for="password">New Password:</label>
<input type="password" id="password" name="password"
required>
    <label for="confirm-password">Confirm
Password:</label>
    <input type="password" id="confirm-password"
name="confirm-password" required>
    <button type="submit" id="submit-btn"
disabled>Reset Password</button>
    <p id="error-message"></p>
</form>
</div>

<script>
    // Get the password and confirm password input
fields
    const passwordInput =
document.getElementById('password');
const confirmPasswordInput =
document.getElementById('confirm-password');
const submitBtn =
document.getElementById('submitbtn');
```

```
    const errorMessage =
document.getElementById('error-message');

    // Add event listeners to the input fields
passwordInput.addEventListener('input',
checkPasswords);
    confirmPasswordInput.addEventListener('input',
checkPasswords);

    // Function to check if passwords
match     function checkPasswords() {
if (passwordInput.value ===
confirmPasswordInput.value) {
            // Enable the submit button if passwords
match
            submitBtn.disabled        =        false;
errorMessage.textContent = '';
        } else {
            // Disable the submit button and display
an error message if passwords do not match
submitBtn.disabled = true;
            errorMessage.textContent = 'Passwords do
not match';
        }
    }
</script>
</body>
</html>
```

Output:



profile.html Code:

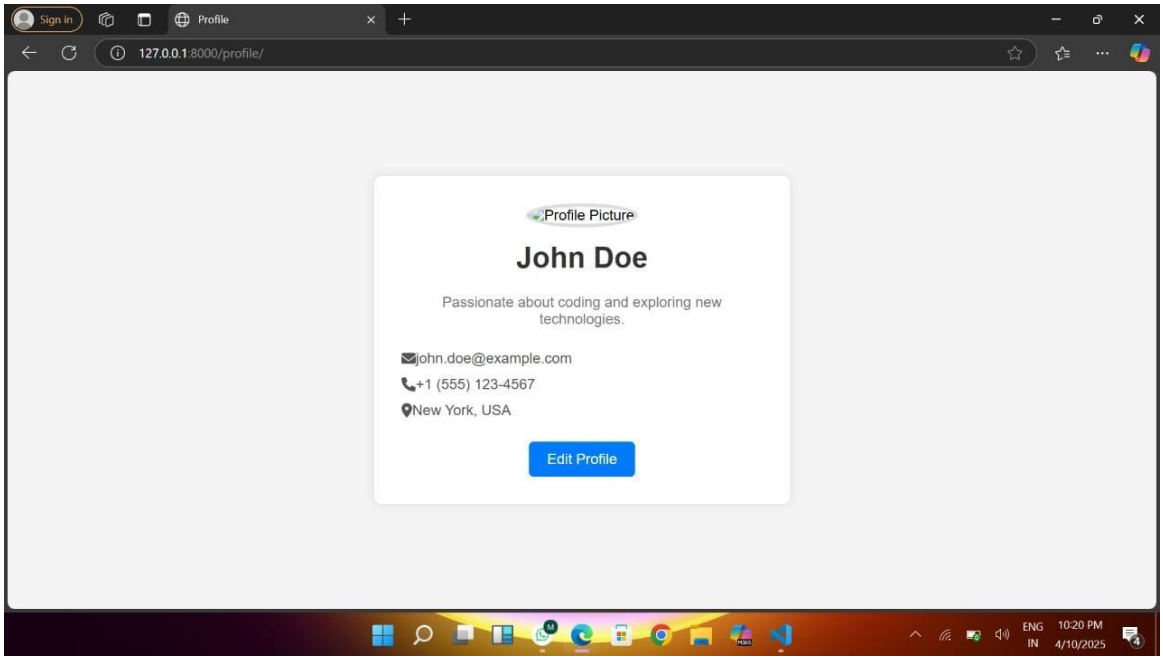
```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta     name="viewport"     content="width=device-width,
initialscale=1.0">
    <title>Your Profile</title>
    <style>
body {
            font-family: sans-serif;
background-color: #f4f4f4;
            margin: 0;
padding: 20px;
display: flex;
            justify-content: center;
        }

        .profile-container {
background-color: #fff;
padding: 30px;             border-radius:
8px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            width: 400px;
max-width: 90%;
        }
```

```
        h2 {
            color: #333;
margin-bottom: 20px;             text-
align: center;
        }

        .profile-info p {
margin-bottom: 10px;             color:
#555;
        }

        .profile-info strong {
font-weight: bold;             color:
#333;             margin-right: 5px;
        }

        .edit-link {
display: block;
margin-top: 20px;
text-align: center;
        }

        .edit-link a {
            text-decoration: none;
color: #007bff;
        }

        .edit-link a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <div class="profile-container">
        <h2>Your Profile</h2>
        <div class="profile-info">
            <p><strong>Username:</strong> <span>{{
user.username }}</span></p>
            <p><strong>First Name:</strong> <span>{{
user.first_name }}</span></p>
            <p><strong>Last Name:</strong> <span>{{
user.last_name }}</span></p>
            <p><strong>Birthday:</strong>
                {% if user_profile.birthday %}
                    <span>{{ user_profile.birthday }}</span>
```

```
                {% else %}
                    <span>Not provided</span>
```

```
                {% endif %}
            </p>
            <p><strong>Location:</strong>
                {% if user_profile.location %}
                    <span>{{ user_profile.location }}</span>
                {% else %}
                    <span>Not provided</span>
                {% endif %}
            </p>
        </div>
        <div class="edit-link">
            <a href="{% url 'edit_profile' %}">Edit Profile</a>
        </div>
    </div>

    <script>
        // The data is now rendered server-side by Django,
// so we don't need JavaScript to populate the fields with
initial data.

        // Any dynamic front-end interactions (like form
submissions on the
        // "Edit Profile" page) would still use JavaScript as
needed.     </script>
</body>
</html>
```

Output:

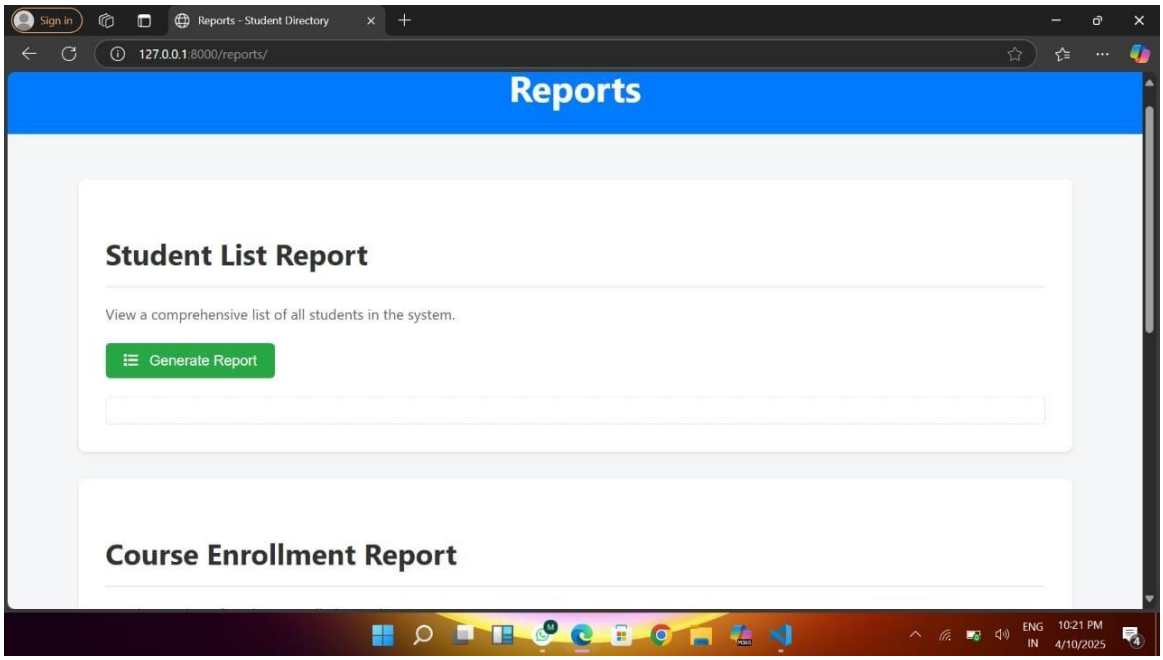


reports.html Code:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta      name="viewport"      content="width=device-width,
initialscale=1.0">
    <title>Reports - Student Directory</title>
    <link rel="stylesheet" href="{% static 'reports.css' %}">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.0.0/
css/all.min.css" integrity="sha512-
9usAa10IRO0HhonpyAIVpjrylPvoDwiPUiKdWk5t3PyolY1cOd4DSE0Ga+ri4Au
TroPR5aQvXU9xC6qOPnzFeg==" crossorigin="anonymous"
referrerpolicy="no-referrer" />
</head>
<body>
    <div class="page-wrapper">
        <header class="site-header">
            <div class="container">
                <h1>Reports</h1>
            </div>
        </header>

        <main class="main-content">
            <div class="container">
                <section class="report-section">
                    <h2>Student List Report</h2>
<p>View a comprehensive list of all students in the
system.</p>
                    <button class="generate-button"><i
class="fas fa-list"></i> Generate Report</button>
<div class="report-placeholder">
                        </div>
                </section>

                <section class="report-section">
                    <h2>Course Enrollment Report</h2>
                    <p>See the number of students enrolled in
each course.</p>
                    <button class="generate-button"><i
class="fas fa-chart-bar"></i> Generate Report</button>
                    <div class="report-placeholder">
                        </div>
                </section>
```

```
                <section class="report-section">
                    <h2>Attendance      Summary</h2>
<p>Summary of student attendance records.</p>
                    <button class="generate-button"><i
class="fas fa-calendar-check"></i> Generate Report</button>
                    <div class="report-placeholder">
                        </div>
                </section>

            </div>
        </main>

        <footer class="site-footer">
            <div class="container">
                <p>&copy; {% now "Y" %} Student Directory
Project. Generating Insights.</p>
            </div>
        </footer>
        <script src="{% static 'js/reports.js' %}"></script>
    </div>
</body>
</html>
```

Output:



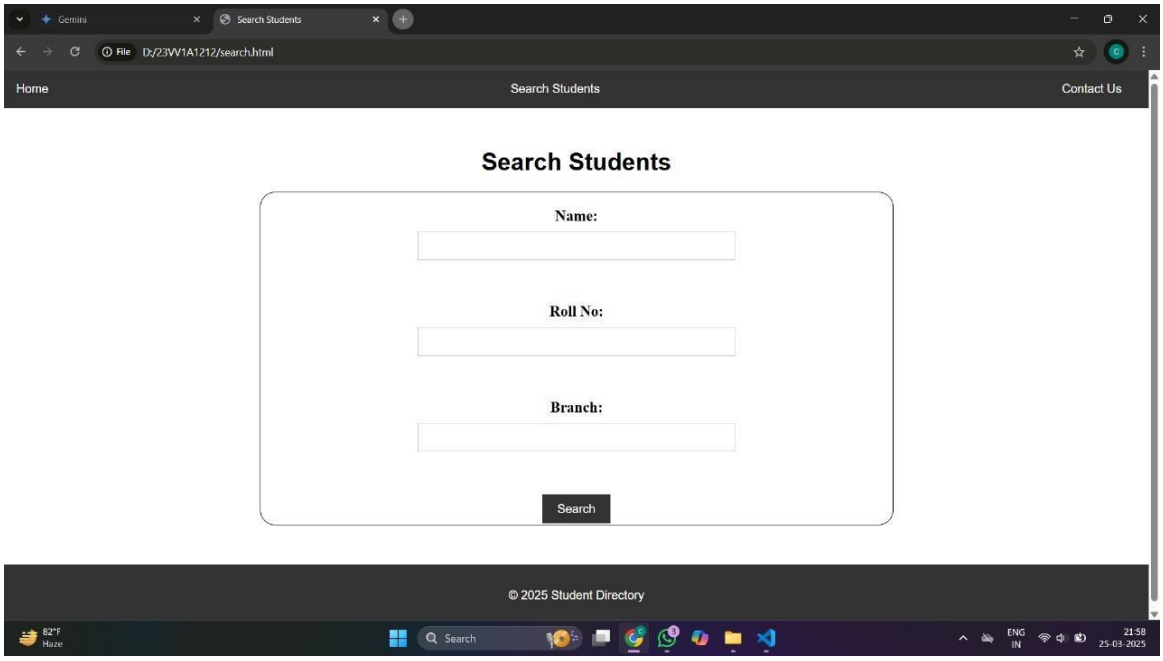
search.html Code:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta     name="viewport"     content="width=device-width,
initialscale=1.0">
    <title>Search Students</title>
    <link rel="stylesheet" href="{% static 'search.css' %}">
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="{% url 'admin' %}">Home</a></li>
                <li class="active"><a href="{% url
'search_students' %}">Search Students</a></li>
                <li><a href="">Contact Us</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h1>Search Students</h1>
        <div class="search">
        <form id="search-form" method="get" action="{% url
'search_results' %}">
            <label for="name" class="name">Name:</label>
            <input type="text" id="name" name="name">
            <br><br>
            <label for="rollno" class="name">Roll No:</label>
            <input type="text" id="rollno" name="rollno">
<br><br>
            <label for="branch" class="name">Branch:</label>
            <input type="text" id="department"
name="department">             <br><br>
            <button id="search-button"
type="submit">Search</button>
        </form>
        </div>
        <div id="search-results">
            {% if search_results %}
                <h2>Search Results:</h2>
                <ul>
```

```
                    {% for student in search_results %}
<li>
```

```
                                {{ student.first_name }} {{
student.last_name }} (Roll No: {{ student.roll_no }},
Department: {{ student.department }})
                            </li>
                        {% endfor %}
                    </ul>
                {% elif searched %}
                    <p>No students found matching your search
criteria.</p>                {% endif %}
            </div>
        </main>
        <footer>
            <p>&copy; 2025 Student Directory</p>
        </footer>
        <script src="{% static 'script.js' %}"></script>
</body>
</html>
```

Output:



Student.html

Code:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta      name="viewport"      content="width=device-width,
initialscale=1.0">
    <title>Student Dashboard</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bo
otstrap.min.css">
    <link rel="stylesheet" href="{% static 'student.css' %}">
</head>
<body>
    <div class="container-fluid">
        <div class="row">
            <nav id="sidebar" class="col-md-3 col-lg-2 d-mdblock
bg-light sidebar">
                <div class="sidebar-sticky">
                    <div class="text-center mt-4">
                        <i class="fa fa-user-circle fa-3x"></i>
<h4>{{ student.name }}</h4> {# Assuming you have a 'student'
object passed in the context #}
                    </div>
                    <ul class="nav flex-column mt-3">
                        <li class="nav-item">
                            <a class="nav-link active" href="{%
url 'home' %}"> {# Replace 'home' with your actual URL name #}
<i class="fa fa-home"></i> Home
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{% url
'signup' %}"> {# Replace 'profile' with your actual URL name #}
                                <i class="fa fa-user"></i>
Create Profile
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{% url
'view_profile' %}"> {# Replace 'profile' with your actual URL
name #}
                                <i class="fa fa-user"></i> View
Profile
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{% url
```

```
'edit_profile' %}"> {# Replace 'edit_profile' with your actual
URL name #}
                                    <i class="fa fa-user"></i> Edit
Profile
                                </a>
```

```
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href=""> {#
Replace 'view_attendance' with your actual URL name #}
                                <i class="fa fa-calendar"></i>
View Attendance
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href=""> {#
Replace 'view_notifications' with your actual URL name #}
                                <i class="fa fa-bell"></i> View
Notifications
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href=""> {#
Replace 'apply_leave' with your actual URL name #}
<i class="fa fa-paper-
plane"></i> Apply For Leave
</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href=""> {#
Replace 'feedback' with your actual URL name #}
                                <i class="fa fa-comment"></i>
Feedback
                            </a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href=""> {#
Replace 'logout' with your actual URL name #}
                                <i class="fa fa-sign-
outalt"></i> Logout
                            </a>
                        </li>
                    </ul>
                </div>
            </nav>

            <main role="main" class="col-md-9 ml-sm-auto collg-
10 px-md-4">
                <div class="d-flex justify-content-between flex-
wrap flex-md-nowrap align-items-center pt-3 pb-2 mb-3 border-
bottom">
```

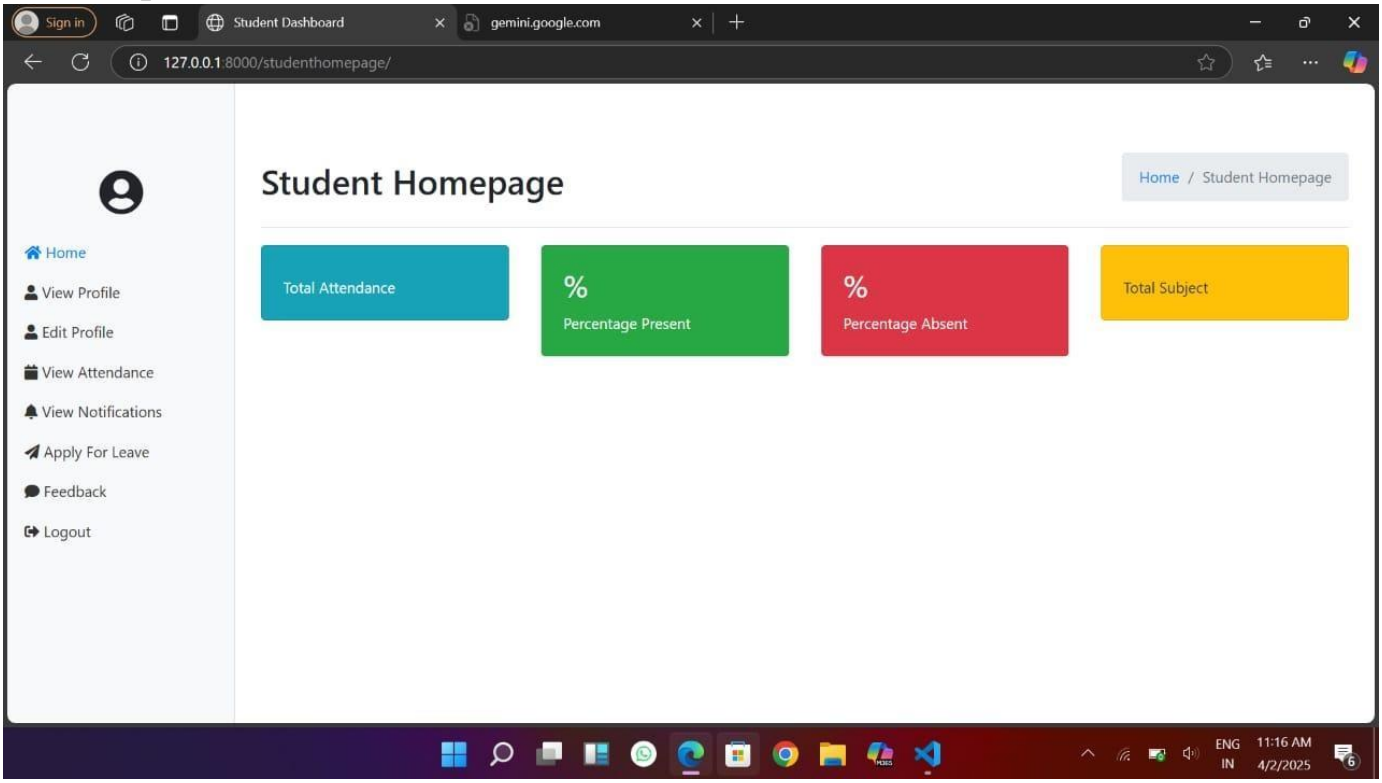
```
<h1 class="h2">Student Homepage</h1>
<nav aria-label="breadcrumb">
    <ol class="breadcrumb">
```

```
                            <li class="breadcrumb-item"><a href="{% url 'home'
%}">Home</a></li> {# Replace 'home' with your actual URL name #}
                            <li class="breadcrumb-item active" aria-
current="page">Student Homepage</li>
                        </ol>
                    </nav>
                </div>

                <div class="row">
                    <div class="col-md-3 mb-4">
                        <div class="card bg-info text-white">
                            <div class="card-body">
                                <h3>{{ total_attendance }}</h3>
{# Assuming you pass this in the context #}
                                <p class="card-text">Total
Attendance</p>
                            </div>
                        </div>
                    </div>
                    <div class="col-md-3 mb-4">
                        <div class="card bg-success textwhite">
                            <div class="card-body">
                                <h3>{{ percentage_present
}}%</h3> {# Assuming you pass this in the context #}
                                <p class="card-text">Percentage
Present</p>
                            </div>
                        </div>
                    </div>
                    <div class="col-md-3 mb-4">
                        <div class="card bg-danger text-white">
                            <div class="card-body">
                                <h3>{{ percentage_absent
}}%</h3> {# Assuming you pass this in the context #}
                                <p class="card-text">Percentage
Absent</p>
                            </div>
                        </div>
                    </div>
                    <div class="col-md-3 mb-4">
                        <div class="card bg-warning text-dark">
                            <div class="card-body">
                                <h3>{{ total_subjects }}</h3>
{# Assuming you pass this in the context #}
                                <p class="card-text">Total
```

```
Subject</p>
                                    </div>
```

```
                    </div>
                </div>
            </div>
        </main>
    </div>
</div>

    <script
src="https://code.jquery.com/jquery3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd
/popper.min.js"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/boot
strap.min.js"></script>     <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css">
    <script
src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script src="{% static 's.js' %}"></script>
</body>
</html>
```

Output:



studentlist.html

Code:

```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initialscale=1.0">
    <title>Student List</title>
    <link rel="stylesheet" href="{% static 'studentlist.css'
%}">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.0.0/c
ss/all.min.css" integrity="sha512-
9usAa10IRO0HhonpyAIVpjrylPvoDwiPUiKdWk5t3PyolY1cOd4DSE0Ga+ri4Au
TroPR5aQvXU9xC6qOPnzFeg==" crossorigin="anonymous"
referrerpolicy="no-referrer" />
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="{% url 'admin'
%}">Dashboard</a></li>
                <li class="active"><a href="{% url
'student_list' %}">Students</a></li>
                <li><a href="{% url
'reports_page'%}">Reports</a></li>
                <li><a href="{% url 'add' %}"
class="addbutton"><i class="fas fa-plus-circle"></i> Add New
Student</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h1><i class="fas fa-graduation-cap"></i> Student
List</h1>
        {% if students %}
            <div class="table-responsive">
```

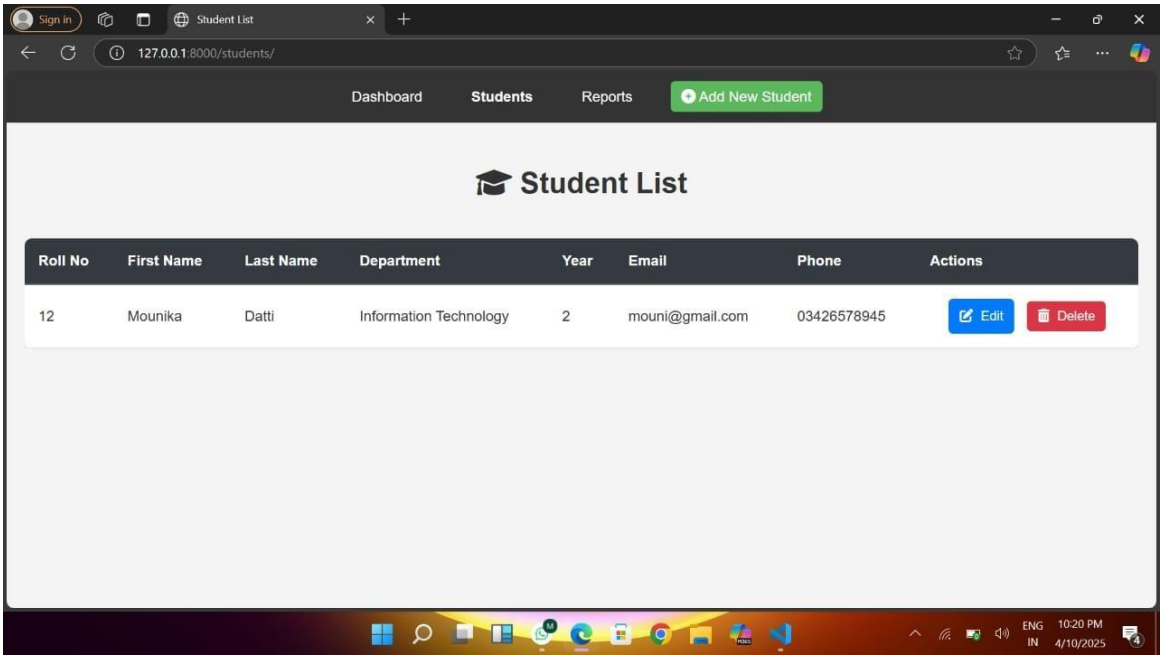
```
<table>
    <thead>
        <tr>
            <th>Roll No</th>
```

```
                            <th>First Name</th>
                            <th>Last Name</th>
                            <th>Department</th>
                            <th>Year</th>
                            <th>Email</th>
                            <th>Phone</th>
                            <th>Actions</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for student in students %}
                            <tr>
                                <td>{{ student.roll_no }}</td>
                                <td>{{ student.first_name
}}</td>
                                <td>{{ student.last_name
}}</td>
                                <td>{{ student.department
}}</td>
                                <td>{{ student.year_of_study
}}</td>
                                <td>{{ student.email }}</td>
                                <td>{{ student.phone }}</td>
                                <td class="actions">
                                    <a href="{% url
'edit_student' student.roll_no %}" class="edit-button"><i
class="fas fa-edit"></i> Edit</a>
                                    <button
class="deletebutton" data-student-id="{{ student.roll_no
}}"><i class="fas fa-trash-alt"></i> Delete</button>
</td>
                            </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        {% else %}
            <p>No students found.</p>
        {% endif %}
    </main>

    <div id="deleteConfirmationModal" class="modal">
        <div class="modal-content">
            <span class="close-button">&times;</span>
            <p>Are you sure you want to delete student with
```

```
Roll No: <span id="delete-student-id"></span>?</p>
            <div class="modal-actions">
                <button id="confirmDelete"
class="confirmbutton">Yes, Delete</button>
```

```
                <button id="cancelDelete"
class="cancelbutton">No, Cancel</button>
            </div>
        </div>
    </div>

    <script src="{% static 'studentlist.js' %}"></script>
</body>
</html>
```

Output:



Admin page:

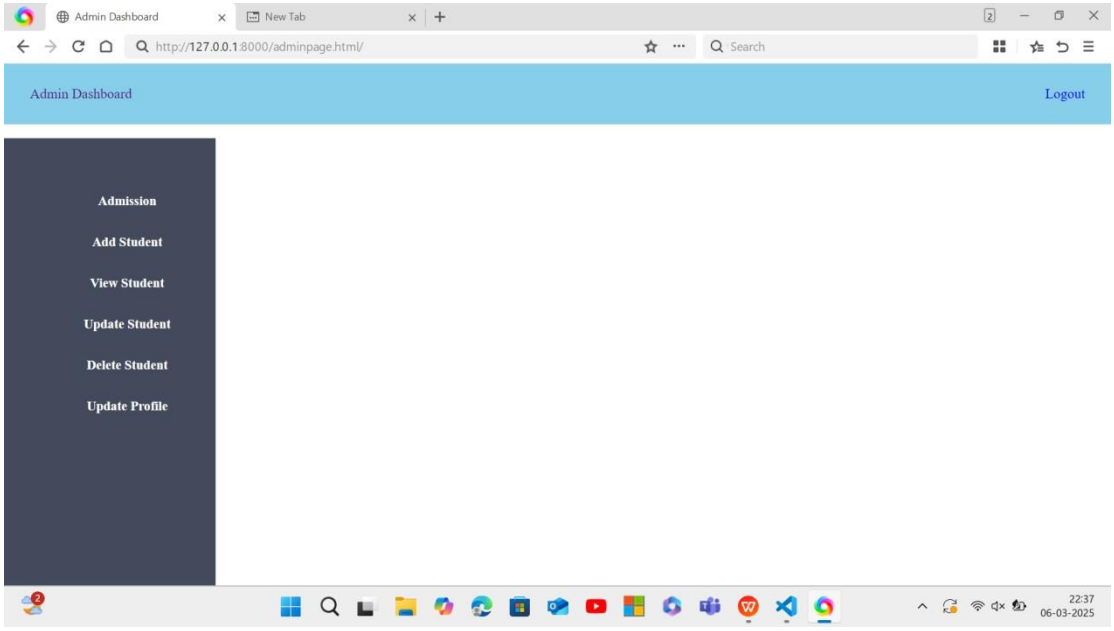
An "admin page" (or "administrator page") is a specialized section of a website or application designed for administrative users. It provides tools and functionalities for managing and controlling the system.

Code:

```
<!DOCTYPE html>
{%load static%}
<html>
<head>
    <title>Admin Dashboard</title>
    <link rel="stylesheet" href="{%static 'admin.css'%}">
</head>
<body>
```

```
    <header class="header">
        <a href="">Admin Dashboard</a>
        <div class="logout">
            <a href="homepage.html">Logout</a>
        </div>
    </header>
    <aside>
        <ul>
            <li><a href="#">Admission</a></li>
            <li><a href="#">Add Student</a></li>
            <li><a href="#">View Student</a></li>
            <li><a href="#">Update Student</a></li>
            <li><a href="#">Delete Student</a></li>
            <li><a href="#">Update Profile</a></li>
        </ul>
    </aside>
</body>
</html>
```

Output:

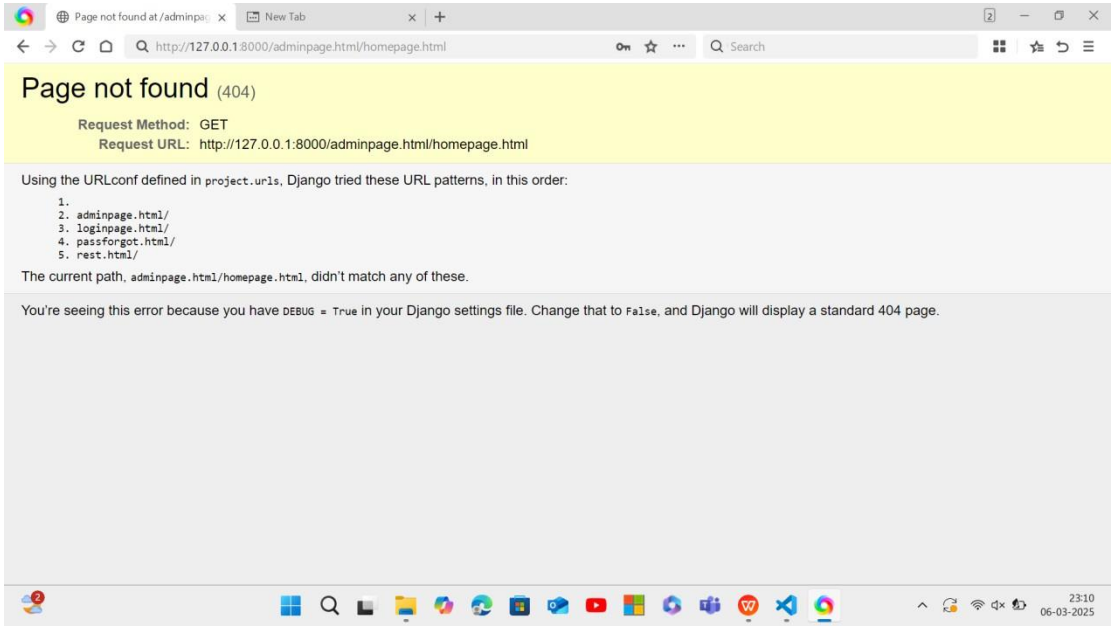


Essential for an exceptional user experience, seamless navigation is the cornerstone of a successful website. HTML page linking, achieved through straightforward hyperlinks, user-friendly navigation menus, and precise section links using anchor tags, empowers users to explore content effortlessly. Developers skillfully employ relative and absolute paths and sophisticated techniques like JavaScript and modern frameworks to forge these vital connections. Critically, effective linking not only dramatically enhances user satisfaction by ensuring effortless site exploration but also significantly boosts search

engine optimization (SEO) by providing clear structural cues that enable search engines to accurately understand and index the site's architecture.

Errors in linking templates:

The template paths don’t matches the HTML pages. It raises an error of PageNotFound (404), the URL patterns are trying to define the templates.



Database in Django:

Django uses SQLite by default. The database is stored as a single file named db.sqlite3 in your project directory.

Code:

```
DATABASES = {

'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

How Databases store in Django:

1. Django models define how data is structured.
2. Migrations (python manage.py migrate) create tables in the database.

Run Migrations to Create Database Tables:

After defining your models, run the following commands to apply them to the database:

1. python manage.py makemigrations
2. python manage.py migrate

Forms.py

What is forms.py in Django:

In Django, forms.py is used to handle user input efficiently and securely. It allows developers to create and manage forms without manually writing HTML and validation logic.

Why Use forms.py:

- Simplifies form creation
- Handles input validation automatically
- Integrates with Django models

- Prevents security risks like SQL Injection & CSRF attacks

Types of Forms in Django:

- Django Forms (forms.Form) – Used for manually creating forms
- **Model Forms (forms.ModelForm)** – Used to create forms directly from **a Django model** Code:

```
from django import forms from
.models import Student
 class
EditStudentForm(forms.ModelForm):
class Meta:
        model = Student
        fields = ['first_name', 'last_name', 'email',
'phone'] # Remove 'address' from this list
widgets = {
            'roll_no':
forms.TextInput(attrs={'readonly': 'readonly'}),
        }
```

Models.PY:

What is models.py in Django:

In Django, models.py is where you define the database structure using Python code. Django models act as a bridge between the database and the application, allowing you to create, read, update, and delete records easily.

Why Use Django Models:

No need to write raw SQL queries, Automatically creates tables in the database and

How to Apply Models:

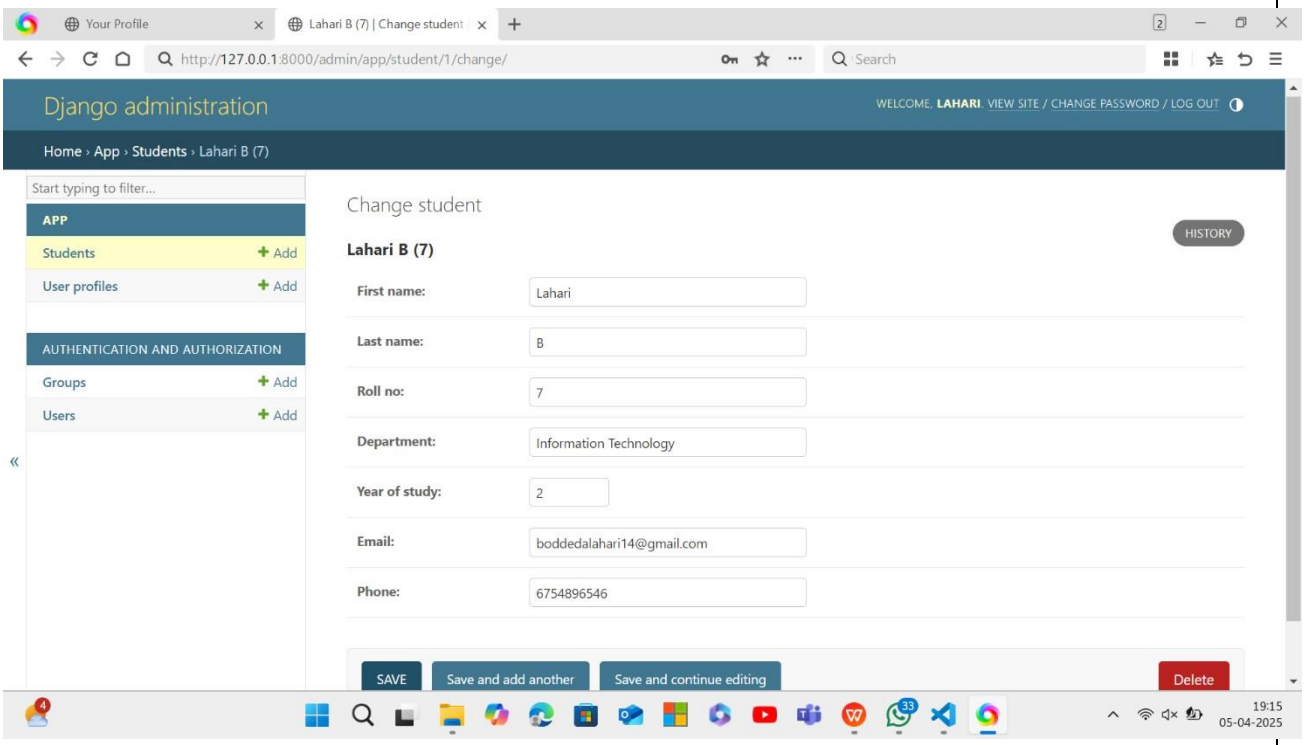
Create the Model in models.py:

Write your models inside the models.py file.

Code:

```
from django.db import models

from django.contrib.auth.models import User
 class
Student(models.Model):

first_name = models.CharField(max_length=100)

last_name = models.CharField(max_length=100)

roll_no = models.CharField(max_length=20, unique=True)

department = models.CharField(max_length=100)

year_of_study = models.IntegerField()

email = models.EmailField(unique=True)

    phone = models.CharField(max_length=20)
 def
__str__(self):

        return f"{self.first_name} {self.last_name}
({self.roll_no})"
     class
UserProfile(models.Model):

user = models.OneToOneField(User, on_delete=models.CASCADE,
 related_name='profile')
birthday = models.DateField(null=True, blank=True)

location = models.CharField(max_length=100, blank=True)
    # Add other profile fields here
 def
_str_(self):

        return self.user.username
```

Output:



Migrations

In Django, migrations are a system for managing changes to your database schema (tables, columns, etc.) in a controlled and versioned manner, ensuring that your database stays in sync with your models.

Key Commands:

1. **Create Migration Files:**
    a. python manage.py makemigrations: This command analyzes your models and generates a migration file (or files) describing your changes.

2. **Apply Migrations:**

    a. python manage.py migrate: This command applies the migration files to your database, updating the schema.

3. python manage.py showmigrations: This command displays the migrations that have been applied to your database.

Userprofile.py:

```
# Generated by Django 5.1.4 on 2025-04-02 06:22

import django.db.models.deletion from
django.conf import settings
from django.db import migrations, models
 class
Migration(migrations.Migration):

    dependencies = [
        ('newapp',
'0002_rename_firstname_student_first_name_and_more'),
        migrations.swappable_dependency(settings.AUTH_USER_MODE
L),
    ]
    operations = [
        migrations.CreateModel(
name='UserProfile',             fields=[
                ('id', models.BigAutoField(auto_created=True,
primary_key=True, serialize=False, verbose_name='ID')),
('birthday', models.DateField(blank=True, null=True)),
                ('location', models.CharField(blank=True,
max_length=100)),                 ('user',
models.OneToOneField(on_delete=django.db.models.deletion.CASCAD
E, related_name='profile', to=settings.AUTH_USER_MODEL)),



            ],
        ),
    ]
```

Rename.py:

```
# Generated by Django 5.1.4 on 2025-04-01 16:59

from django.db import migrations, models
 class
Migration(migrations.Migration):

    dependencies = [
        ('newapp', '0001_initial'),
    ]
    operations = [
        migrations.RenameField(
model_name='student',
old_name='firstName',
new_name='first_name',
        ),
        migrations.RenameField(
model_name='student',
old_name='lastName',
new_name='last_name',
        ),
        migrations.RemoveField(
model_name='student',             name='address',
        ),
        migrations.RemoveField(
model_name='student',             name='studentId',
        ),
        migrations.AddField(
model_name='student',
name='department',
            field=models.CharField(default='None',
max_length=100),         ),
        migrations.AddField(
model_name='student',             name='id',
            field=models.AutoField(default=1,  primary_key=True,
serialize=False),
            preserve_default=False,
```

```
        ),
        migrations.AddField(
model_name='student',              name='roll_no',
            field=models.CharField(default='None',
max_length=20, unique=True),
        ),
        migrations.AddField(
model_name='student',
name='year_of_study',
            field=models.IntegerField(default=1),
preserve_default=False,
        ),
        migrations.AlterField(
model_name='student',              name='email',
            field=models.EmailField(max_length=254,
unique=True),          ),
        migrations.AlterField(
model_name='student',              name='phone',
            field=models.CharField(default=1, max_length=20),
preserve_default=False,
        ),
]
```

Initial.py:

```
# Generated by Django 5.1.4 on 2025-04-01 12:57

from django.db import migrations, models
 class
Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]
    operations = [
        migrations.CreateModel(
name='Student',
```

```
            fields=[
                ('studentId', models.CharField(max_length=10,
primary_key=True, serialize=False)),
('firstName', models.CharField(max_length=100)),
                ('lastName', models.CharField(max_length=100)),
                ('email', models.EmailField(max_length=254)),
('phone', models.CharField(blank=True, max_length=20,
null=True)),
                ('address', models.TextField(blank=True,
null=True)),             ],
        ),
    ]
```

Deploying a Django Web Application on the cloud

What is Deployment?

Deployment is making a Django web application live on the internet so users can access it. This involves hosting your app on a cloud server like AWS, Google Cloud, Digital Ocean, Heroku, or PythonAnywhere Features:

Scalability—You can handle more users without performance issues. Security—You can protect user data with SSL and secure databases. Global Accessibility—Users can access your app from anywhere.

Continuous Deployment – Easily update your app with new features.

Here's a step-by-step guide to register on GitHub, create a Django website with login and registration pages, and Configure Django to handle static files.

Step 1: Register on GitHub

1. Go to GitHub and click Sign up.
2. Enter your Username, Email, and Password.
3. Complete the verification and click Create Account.
4. Verify your email by clicking the link in your inbox.

Step 2: Push to GitHub

Initialize Git in your project: git init

2.Connect to GitHub:

git remote add origin https://github.com/Mounika-datti/Student-Directory-

System.git

3.Add and commit changes: git add.

git commit -m "Initial Commit: Student Directory App"

4.Push to GitHub:

git branch -M main git

push -u origin main

You have successfully built a Django website with login, registration, and static file management.

Your code is now available on GitHub.

GITHUB LINK : https://github.com/Mounika-datti/Student-Directory-System

Conclusion:

In conclusion, the Student Directory Project with Django addresses real-world challenges within educational institutions. It effectively showcases a complete development lifecycle, from meticulous problem identification and strategic solution design to efficient and successful implementation. This project not only delivers a valuable tool for institutions but also serves as an indispensable case study, illustrating the full spectrum of the Django framework. By navigating from initial problem definition to final deployment, it solidifies the practical skills essential for constructing robust and impactful web applications.