

--

Program No: 1

Problem statement:

A program that creates a simple web server and static HTML page.

Aim:

To solve this problem statement.

Problem Description:

Develop a Python program using Flask to create a simple web server serving a static HTML page. The objective is to use the `flask --app <YourAppName> run` command instead of running the script directly. Ensure proper project organization, write a basic HTML file, and confirm functionality by accessing `http://127.0.0.1:5000/` in a web browser.

Algorithm:

Step 1: start

Step 2: Install Flask

Step 3: create a Project Structure

Step 4: Write a HTML file and save as a index.html file

Step 5: Write a Flask file and save as a app.py file

Step 6: Run that Flask file.

Step 7: Stop

HTML FILE:

Index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1">
```

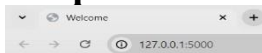
```
</body>  
</html>
```

FLASK FILE:

app.py

```
from flask import Flask,  
render_template app = Flask(__name__  
_____  
# Define a route for the root URL "/"  
@app.route("/")  
def home():  
    # Render the static HTML page located in the "templates" folder  
    return render_template("index.html")  
  
# Run the app if this script is the main  
program if __name__ == "__main__":  
    app.run(debug=True)
```

Output:



Hello World!

Conclusion:

Here We have successfully creates a simple web server and

Program No: 2

Problem statement:

A program that creates a web application that allows users to register and login.

Aim:

To solve this problem statement.

Problem Description:

The web application will be built using a combination of front-end and back-end technologies. The front-end will be responsible for creating the user interface, while the back-end will handle user registration, login authentication, and data storage.

Algorithm:

Step 1: start

Step 2: create a Project Structure

Step 3: Write a HTML file and save as a index.html ,login.html & register.html file

Step 4: Write a Flask file and save as a app.py file

Step 5: Run that Flask file.

Step 6: Stop

HTML FILE:

Index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8" />
```

```
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

```
@import url("https://fonts.googleapis.com/css2?
family=Poppins:wght@500&display=sw ap");
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
body {
  height: 100vh;
  width: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction:
  column; background:
  #ff5a5f;
}
h1 {
  font-family: "Poppins", sans-serif;
  color: #fff;
  margin: 30px
  50px; font-size:
  3rem;
}
input {
  padding: 10px 20px;
  border: 3px solid
  #fff; border-radius:
  10px;
  background: rgb(16, 208,
  16); font-size: 1.5rem;
  color: white;
  font-family: "Poppins", sans-
  serif; font-weight: 300;
  transition: .3s;
  &:hover{ backgr
```

}

```
</style>
<body>
  <h1>Hello, this is a static HTML page served by Flask!</h1>
  <form action="{{ url_for('register') }}">
    <input type="submit" value="Register" />
  </form>
</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>User Login</title>
    <style>
      * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
      }
      body {
        height: 100vh;
        width: 100%;
        display: flex;
        align-items: center;
        justify-content: center;
        flex-direction:
          column;
        background: rgb(9, 9, 121);
        background: linear-gradient(
          30deg,
          rgba(9, 9, 121, 1) 0%
```



```
.container {
  display: flex;
  align-items: center;
  justify-content: space-
evenly; flex-direction:
column; width: 600px;
border-radius: 20px;
height: 500px;
background: #ffffff5a;
backdrop-filter:
blur(20px); & h1 {
  font-family: Arial, Helvetica, sans-serif;
  color: #fff;
  margin: 30px 0;
}
& li {
  list-style: none;
}
& form {
  & label {
    color: white;
    font-family: Arial, Helvetica, sans-
serif; font-size: 1.4rem;
    margin: 10px 20px;
  }
  & .log_button {
    color: #fff;
    background: red;
    border: none;
    outline: none;
    padding: 5px 10px;
    border-radius:
10px; font-size:
1.2rem; transition:
0.3s;
    transform: translateX(130px);
```



```
    }  
  }  
  & .password{ paddin  
    g: 10px 20px;  
    border-radius:  
    20px; outline: none;  
    border: none;  
  }  
  & .username{ paddin  
    g: 10px 20px;  
    border-radius:  
    20px; outline: none;  
    border: none;  
  }  
  & input {  
    margin: 10px 20px;  
  }  
}  
  
.error {  
  color: red;  
}  
.  
success {  
  color: green;  
}  
.  
default {  
  color: black;  
}  
  </style>  
</head>  
<body>  
  <div class="container">  
    <h1>User Login</h1>  
    {% with messages = get_flashed_messages() %} {% if messages %}  
    <ul>  
      {% for message in messages %}
```

```
>
  {{ message }}
</li>
{% endfor %}
</ul>
{% endif %} {% endwith %}
<form method="post" action="{{ url_for('login') }}">
  <label for="username" class="username_label">Username:</label>
  <input type="text" name="username" class="username" required />
  <br />
  <label for="password" class="password_label">Password:</label>
  <input type="password" name="password" class="password" required />
  <br />
  <input type="submit" class="log_button" value="Log in" />
</form>
<p>
  Don't have an account?
  <a href="{{ url_for('register') }}">Register here</a>.
</p>
</div>
</body>
</html>
```

register.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>User Registration</title>
  <style>
    * {
      margin: 0;
      padding: 0;
```

```
height: 100vh;
width: 100%;
display: flex;
align-items: center;
justify-content: center;
flex-direction:
column;
background: rgb(9, 9, 121);
background: linear-gradient(
30deg,
  rgba(9, 9, 121, 1) 0%,
  rgba(2, 0, 36, 1) 29%,
  rgba(0, 212, 255, 1) 100%
);
}
.container {
display: flex;
align-items: center;
justify-content: space-
evenly; flex-direction:
column; width: 600px;
border-radius: 20px;
height: 500px;
background: #ffffff5a;
backdrop-filter:
blur(20px); & h1 {
font-family: Arial, Helvetica, sans-serif;
color: #fff;
margin: 30px 0;
}
& li {
list-style: none;
}
& form {
& label {
color: white;
```



```
& .register_button {
    color: #fff;
    background: red;
    border: none;
    outline: none;
    padding: 5px 10px;
    border-radius:
    10px; font-size:
    1.2rem; transition:
    0.3s;
    transform: translateX(130px);
    &:hover {
        background: #fff;
        color: #000;
        cursor: pointer;
    }
}
& .password {
    padding: 10px
    20px; border-
    radius: 20px;
    outline: none;
    border: none;
}
& .username {
    padding: 10px
    20px; border-
    radius: 20px;
    outline: none;
    border: none;
}
& input {
    margin: 10px 20px;
}
}
```


color: green;
}

```
.default {
    color: black;
}
</style>
</head>
<body>
<div class="container">
<h1>User Registration</h1>
{% with messages = get_flashed_messages() %} {% if messages %}
<ul>
    {% for message in messages %}
    <li
        class="{% if 'error' in message %}error{% elif 'success' in message %}success{% else %}default{% endif %}"
    >
        {{ message }}
    </li>
    {% endfor %}
</ul>
{% endif %} {% endif %}
<form method="post" action="{% url_for('register') %}">
    <label for="username" class="username_label">Username:</label>
    <input type="text" name="username" class="username" required />
    <br />
    <label for="password" class="password_label">Password:</label>
    <input type="password" name="password" class="password" required />
    <br />
    <input type="submit" class="register_button" value="Register" />
</form>
<p>
    Already have an account?
    <a href="{% url_for('login') %}">Log in here</a>.
</p>
</div>
</body>
</html>
```

FLASK FILE:

app.py

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

```
from flask_sqlalchemy import SQLAlchemy
```

```
from werkzeug.security import generate_password_hash, check_password_hash
```

```
import secrets
```

```
# print(secrets.token_hex(16))
```

```
app = Flask(__name__)
```

```
app.secret_key = secrets.token_hex(16)
```

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
```

```
# Change the database URI to match the location of the database, change for other databases
```

```
db = SQLAlchemy(app)
```

```
# Define the User model
```

```
class User(db.Model):
```

```
    id = db.Column(db.Integer, primary_key=True)
```

```
    username = db.Column(db.String(50), unique=True, nullable=False)
```

```
    password = db.Column(db.String(256), nullable=False)
```

```
# Ensure the creation of all tables inside the application context
```

```
with app.app_context():
```

```
    db.create_all()
```

```
@app.route("/")
```

```
def home():
```

```
    # Render the static HTML page located in the "templates" folder
```

```
    return render_template("index.html")
```

```
# Define a route for the registration page
```

```
@app.route('/register', methods=['GET', 'POST'])
```



```
password = request.form['password']

# Check if the username is already taken
if User.query.filter_by(username=username).first():
    flash('Username already taken. Please choose another.', 'error')
else:
    # Hash the password before storing it
    hashed_password = generate_password_hash(password,
method='pbkdf2:sha256')

    # Create a new user instance
    new_user = User(username=username, password=hashed_passw

    # Add the user to the database
    db.session.add(new_user)
    db.session.commit()

    flash('Registration successful. You can now log in.', 'success')
    return redirect(url_for('login'))

return render_template('register.html')

# Define a route for the login page
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        # Query the user from the database
        user = User.query.filter_by(username=username).first()

        # Check if the username exists and the password is correct
        if user and check_password_hash(user.password, password):
            session['username'] = username
            flash('Login successful!', 'success')
```

```
return render_template('login.html')

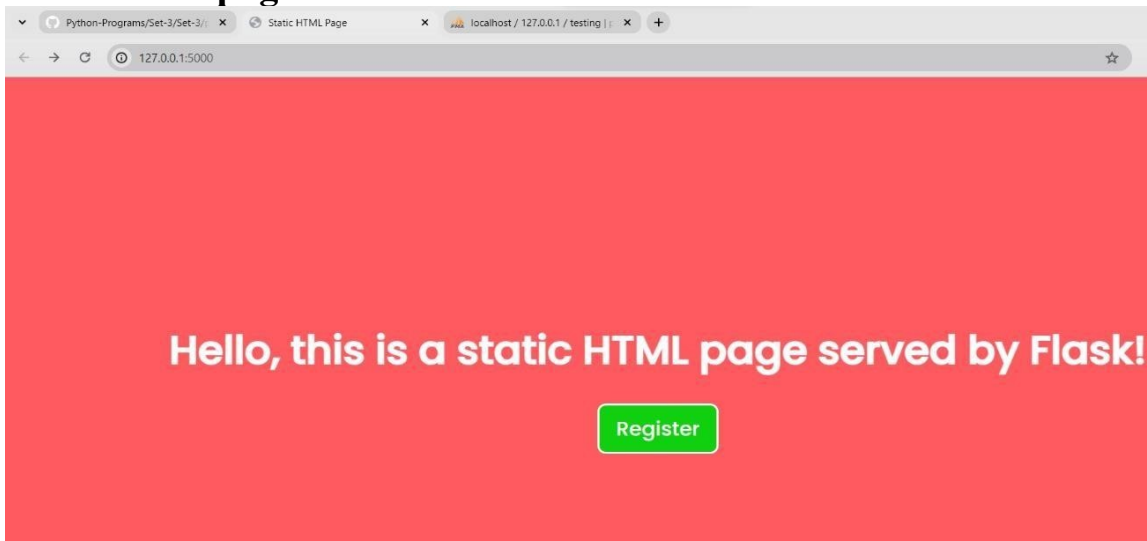
# Define a route for the dashboard (a protected route accessible only after login)
@app.route('/dashboard')
def dashboard():
    if 'username' in session:
        return f'Welcome to the dashboard, {session["username"]}!'
    else:
        flash('Please log in to access the dashboard.', 'info')
        return redirect(url_for('login'))

# Logout route
@app.route('/logout')
def logout():
    session.pop('username', None)
    flash('You have been logged out.', 'info')
    return redirect(url_for('login'))

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

Output:

#static html page:



#Registration page

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/register?'. The page has a dark blue background with a lighter blue gradient. In the center, there is a white rounded rectangle containing the text 'User Registration' in bold. Below this, there is a green message 'Login successful!'. The form includes two input fields: 'Username:' and 'Password:'. A red 'Register' button is positioned below the password field. At the bottom of the form, there is a link that says 'Already have an account? [Log in here.](#)'. The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons.

#Login page

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/login'. The page has a dark blue background with a lighter blue gradient. In the center, there is a white rounded rectangle containing the text 'User Login' in bold. Below this, there is a green message 'Login successful!'. The form includes two input fields: 'Username:' and 'Password:'. A red 'Log in' button is positioned below the password field. At the bottom of the form, there is a link that says 'Don't have an account? [Register here.](#)'. The browser's taskbar at the bottom shows the Windows logo, a search bar, and various application icons.

#Dashboard



Conclusion:

Here We have successfully creates a web application allows users to register and login.

Program No:3

Problem statement:

A program that creates a web application that allows users to upload and download files.

Aim:

To solve this problem statement.

Problem Description:

Develop a Flask-based web application featuring:

1. File Upload:

- Users can upload files through a web form.
- Prevent form submission without selecting a file.
- Save uploaded files to a server directory (e.g., uploads).

2. File Display:

- Display a dynamic list of uploaded files on the main page.
- Each file entry includes a "Download" button.

3. File Download:

- Enable users to download files by clicking the corresponding "Download" button.

4. User Interface:

- Design a clean and user-friendly interface using HTML templates.
- Separate application code (app.py) and HTML templates.

Algorithm:

Step 1: start

Step 2: create a Project Structure

Step 3: Write a HTML file and save as a

index.htmlfile **Step 4:** Write a Flask file and save as a

app.py file **Step 5:** Run that Flask file.

Step 6: Stop

HTML FILE

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>File Upload and Download</title>
</head>
<body>
  <h1>File Upload and Download</h1>
  <form action="/upload" method="post" enctype="multipart/form-data">
    <label for="file">Choose a file:</label>
    <input type="file" name="file" id="file" required>
    <br>
    <input type="submit" value="Upload">
  </form>

  <h2>Uploaded Files</h2>
  {% for filename in filenames %}
    <div>
      <span>{{ filename }}</span>
      <a href="{{ url_for('download_file', filename=filename) }}">
        <button>Download</button>
      </a>
    </div>
  {% endfor %}
</body>
</html>
```

FLASK FILE

app.py

```
from flask import Flask, render_template, request, send_from_directory
redirect_url_for
```

```

UPLOAD_FOLDER = 'uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
@app.route('/')
def index():
    filenames = os.listdir(app.config['UPLOAD_FOLDER'])
    return render_template('index.html', filenames=filenames)
@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return "No file part"
    file = request.files['file']
    if file.filename == "":
        return "No selected file"
    file.save(os.path.join(app.config['UPLOAD_FOLDER'], file.filename))
    return redirect(url_for('index'))
@app.route('/download/<filename>')
def download_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)

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

```

Output:



Program No:4

Problem statement:

A program that creates a web application that displays data from a database in a tabular format.

Aim:

To solve this problem statement.

Problem Description:

The program utilizes Flask, SQLAlchemy, and Pandas to create a web application. It defines a simple SQLAlchemy model (Person) to represent data with attributes like name and age. Sample data is inserted into an SQLite database. The main route (/) queries the database, converts the results to a Pandas DataFrame, and renders it as an HTML table using a Flask template (index.html).

Algorithm:

Step 1: start

Step 2: create a Project Structure

Step 3: Write a HTML file and save as a

index.htmlfile **Step 4:** Write a Flask file and save as a
app.py file **Step 5:** Run that Flask file.

Step 6: Stop

HTML FILE

index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1">
```

```
  <title>Data Display</title>
```

```
  <link rel="stylesheet" href="style.css">
```

```
<h1>Data Display</h1>
<!-- Render the HTML table -->
{{ table_html | safe }}
</div>
</body>
</html>
```

FLSK FILE

app.py

```
from flask import Flask,
render_template import pandas as pd

app = Flask(__name__)
sample_data = {'name': ['John', 'Alice', 'Bob'],
               'age': [25, 30, 22]}
df = pd.DataFrame(sample_data)
@app.route('/')
def display_data():
    table_html = df.to_html(classes='table table-striped', index=False)
    return render_template('index.html', table_html=table_html)

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

Output:



Data Display

name	age
Rohan	25
Tushar	30
Mohit	22

Conclusion:

Program No:5

Problem statement:

A program that creates a web application that takes user input and sends it to a server-side script for processing.

Aim:

To solve this problem statement.

Problem Description:

You are tasked with creating a web application using Flask that enables users to input data on the main page through a form. The entered data should be sent to the server, processed by a server-side script, and the result displayed on the same page. The provided code includes a basic structure for achieving this, where user input is obtained from a form, and simple processing logic is applied.

Algorithm:

Step 1: start

Step 2: create a Project Structure

Step 3: Write a HTML file and save as a

index.html file **Step 4:** Write a Flask file and save as a

app.py file **Step 5:** Run that Flask file.

Step 6: Stop

HTML FILE

index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8" />
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<title>User Input</title>
```

```
</head>
```

```
padding: 0;
box-sizing: border-box;
}
body {
height: 100vh;
width: 100%;
background: #a2d2ff;
display: flex;
align-items: center;
justify-content: center;
flex-direction:
column;
}
.container {
display: flex;
align-items: center;
justify-content: space-
evenly; flex-direction:
column; width: 500px;
height: 600px;
border-radius:
20px;
background: #ffffff5a;
backdrop-filter:
blur(20px); & h1 {
font-family: Arial, Helvetica, sans-serif;
color: #3a86ff;
font-size: 2rem;
}
& label{
color: #3a86ff;
font-family: Arial, Helvetica, sans-
serif; font-size: 1.2rem;
padding: 10px;
margin: 10px 20px;
}
```

border-radius:
20px;

```

    }
    & .submit{
        padding: 10px 20px;
        color: #fff;
        background: #2a9d8f;
        outline: none;
        border: none;
        border-radius:
        10px; transition:
        .3s;
        transform: translateX(150px);
        margin: 30px;
        &:hover{ colo
            r: #000;
            cursor: pointer;
            background: #fff;
        }
    }
}
& h2{
    font-family: Arial, Helvetica, sans-serif;
    color: #3a86ff;
    font-size: 2rem;
}
}
</style>
<body>
    <div class="container">
        <h1>User Input Form</h1>
        <form method="post" action="/">
            <label for="user_input">Enter something:</label>
            <input type="text" class="enter" name="user_input" id="user_inp
required />
            <br />
            <input class="submit" type="submit" value="Submit" />
        </form>

```



```
</div>
    {% endif %}
</div>
</body>
</html>
```

FLSK FILE

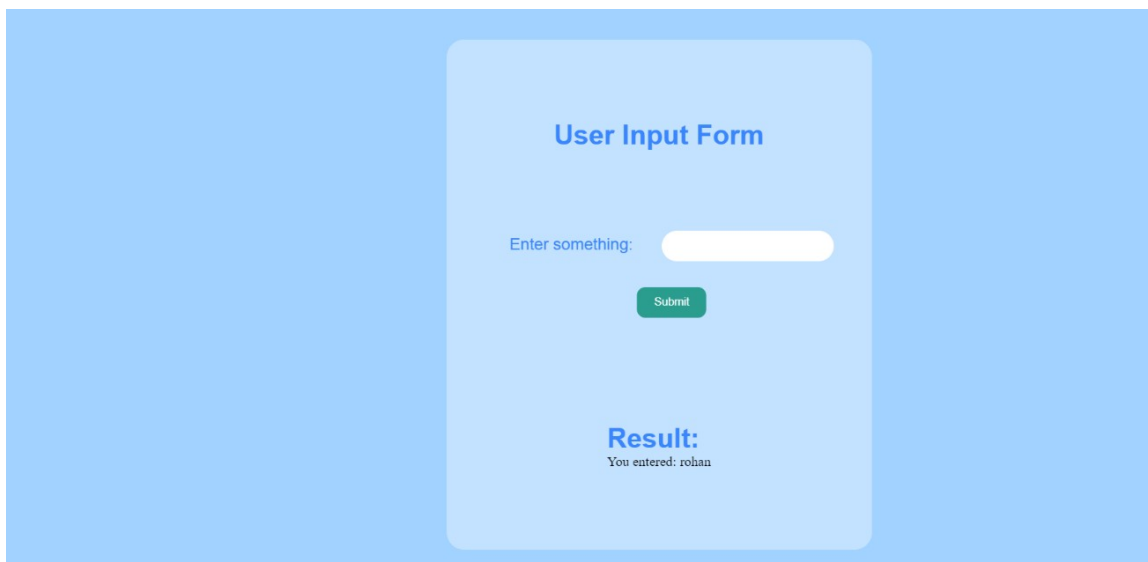
app.py

```
from flask import Flask, render_template, request
```

```
app = Flask(__name__)
@app.route('/', methods=['GET', 'POST'])
def index():
    result = None
    if request.method == 'POST':
        # Get user input from the form
        user_input = request.form.get('user_input')
        result = f"You entered: {user_input}"
    return render_template('index.html', result=result)

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

Output:



The screenshot shows a web application interface with a light blue background. In the center, there is a white rounded rectangle containing the form. At the top of this rectangle is the title 'User Input Form' in blue. Below the title, the text 'Enter something:' is followed by a white text input field. Underneath the input field is a green 'Submit' button. At the bottom of the white rectangle, the word 'Result:' is displayed in blue, followed by the text 'You entered: rohan'.



COMPUTER SCIENCE AND ENG
FACULTY OF ENGINEERING &
TECHNOLOGY PFSD (303105258)
SEM ENROLLMENT NO: 2203051



COMPUTER SCIENCE AND ENG
FACULTY OF ENGINEERING &
TECHNOLOGY PFSD (303105258)
SEM ENROLLMENT NO: 2203051

