Weather Outfit Recommender using Flask

Riya Parikh (22MID0356)

1. Objective

The objective of this project is to develop a simple Flask web application that recommends suitable outfits based on the current weather conditions and temperature.
The project demonstrates how backend logic (Flask) can be integrated with a visually appealing frontend (HTML and CSS) to build an interactive, real-world web application.

2. Tools and Technologies Used

- **Programming Language:** Python
- **Framework:** Flask
- **Frontend:** HTML, CSS
- **IDE:** Visual Studio Code
- **Operating System:** macOS
- **Browser:** Google Chrome

3. Project Overview

This web app allows users to enter the temperature (in °C) and select the weather condition (Sunny, Rainy, or Windy).
Based on these inputs, the Flask backend suggests what type of clothes the user should wear.

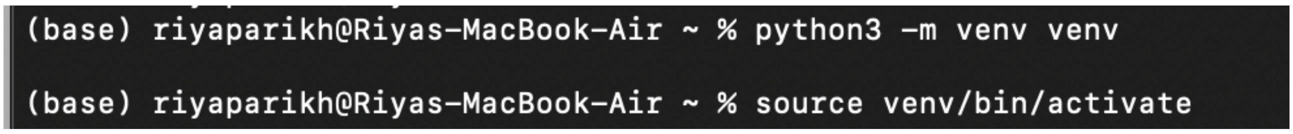
4. Folder Structure

```
flask/
├── app.py
├── static/
│   └── styles.css
└── templates/
    └── index.html
```

5. Setting up the Environment

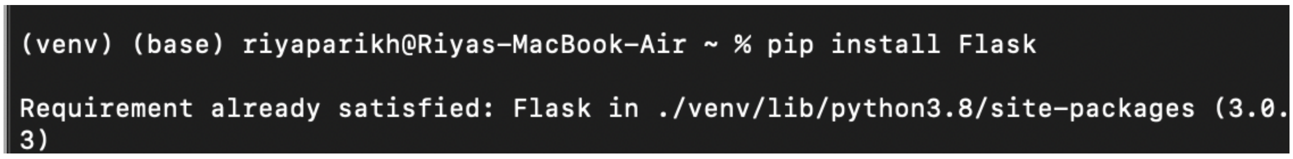
Steps:

1. Create a virtual environment and activate it



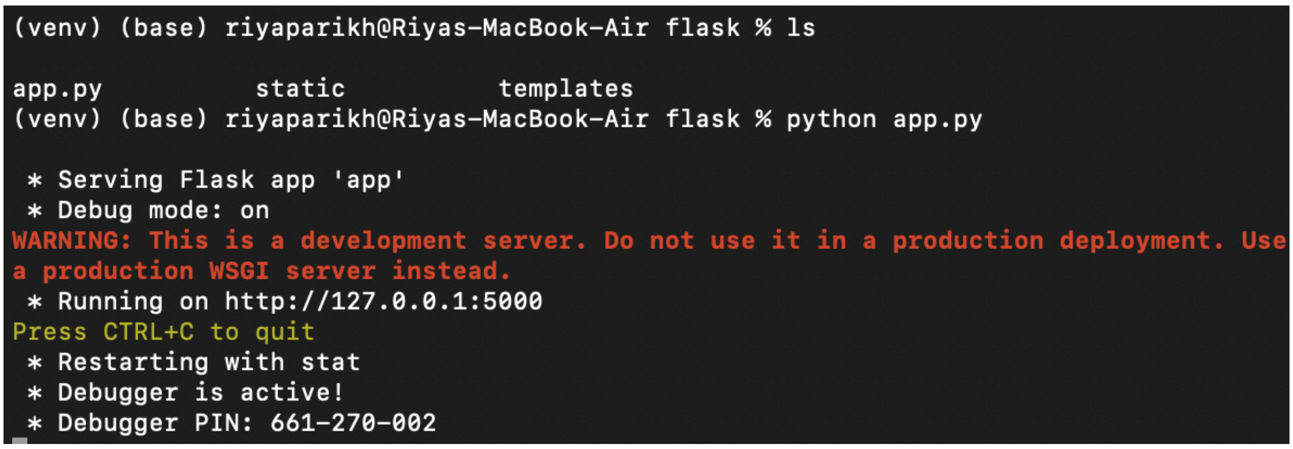
```
(base) riyaparikh@Riyas-MacBook-Air ~ % python3 -m venv venv

(base) riyaparikh@Riyas-MacBook-Air ~ % source venv/bin/activate
```

2.Install Flask:



```
(venv) (base) riyaparikh@Riyas-MacBook-Air ~ % pip install Flask

Requirement already satisfied: Flask in ./venv/lib/python3.8/site-packages (3.0.
3)
```

3. Run the app:



```
(venv) (base) riyaparikh@Riyas-MacBook-Air flask % ls

app.py          static          templates
(venv) (base) riyaparikh@Riyas-MacBook-Air flask % python app.py

 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 661-270-002
```

6. Backend Code (app.py)

The backend is written in Python using Flask.
It receives user input, processes the weather and temperature, and sends an appropriate outfit suggestion to the frontend.

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    suggestion = None
    if request.method == "POST":
        temp = float(request.form["temperature"])
        condition = request.form["condition"]

        if condition == "Rainy":
            outfit = "Wear a waterproof jacket and boots ☔"
        elif condition == "Sunny":
            if temp > 30:
                outfit = "Wear a light cotton t-shirt, shorts, and sunglasses 😎"
            else:
                outfit = "A shirt and jeans would be perfect 🌤️"
        elif condition == "Windy":
            outfit = "Wear a windbreaker and jeans 💨"
        else:
            outfit = "Choose comfortable layers 👕"

        suggestion = {"outfit": outfit}

    return render_template("index.html", suggestion=suggestion)

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

7. Frontend Code(index.html)

The HTML file contains the input form and output section. It uses Jinja templating to dynamically display the outfit recommendation.

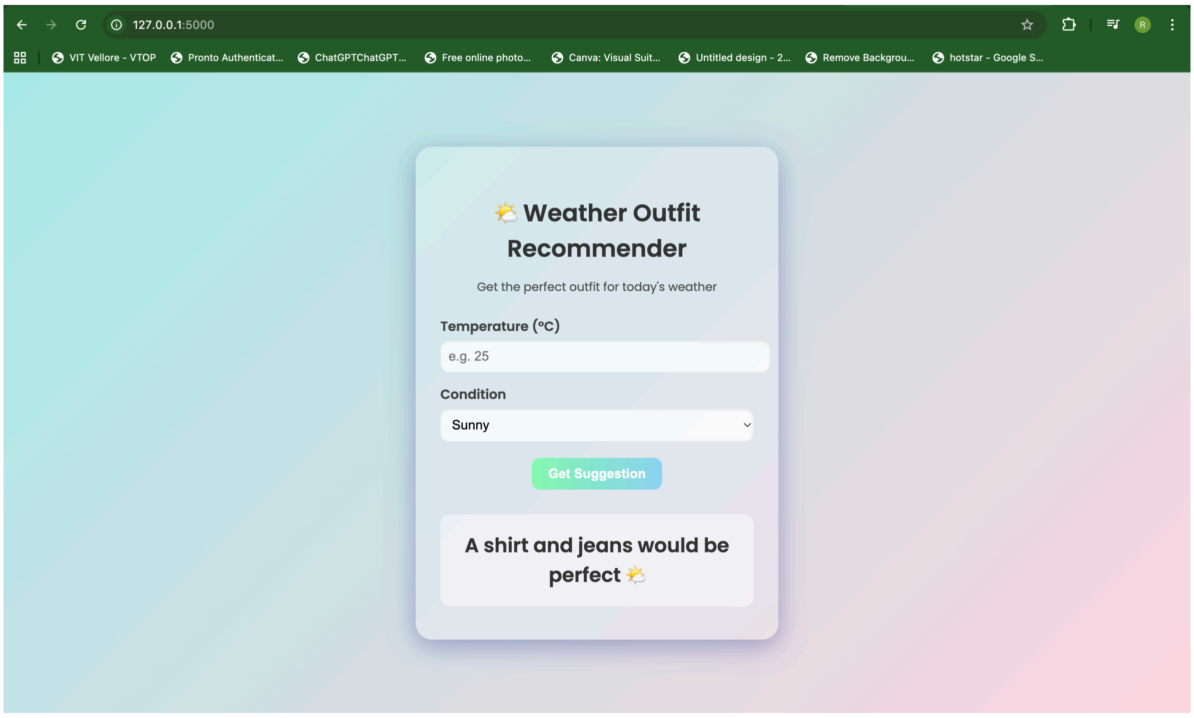
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Weather Outfit Recommender</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
rel="stylesheet">
</head>
<body>
```

```
  <div class="container">
    <div class="card">
      <h1>⛅ Weather Outfit Recommender</h1>
      <p class="tagline">Get the perfect outfit for today's weather</p>

      <form method="POST" action="/">
        <label>Temperature (°C)</label>
        <input name="temperature" type="number" step="0.1" placeholder="e.g. 25"
required>

        <label>Condition</label>
        <select name="condition">
          <option>Sunny</option>
          <option>Rainy</option>
          <option>Windy</option>
        </select>

        <button type="submit">Get Suggestion</button>
      </form>

      {% if suggestion %}
      <div class="suggestion">
        <h2>{{ suggestion.outfit }}</h2>
      </div>
      {% endif %}
    </div>
  </div>
</body>
</html>
```

Styles.css

```
body {
  margin: 0;
  padding: 0;
  height: 100vh;
  font-family: 'Poppins', sans-serif;
  background: linear-gradient(135deg, #a8edea, #fed6e3);
  display: flex;
  align-items: center;
  justify-content: center;
}

.card {
  background: rgba(255, 255, 255, 0.25);
  backdrop-filter: blur(10px);
  border-radius: 20px;
  padding: 40px;
  width: 380px;
  text-align: center;
  box-shadow: 0 8px 32px rgba(31, 38, 135, 0.37);
}

h1 {
  font-size: 1.8rem;
  color: #333;
  margin-bottom: 10px;
}

button {
  margin-top: 20px;
  padding: 10px 20px;
  background: linear-gradient(90deg, #84fab0, #8fd3f4);
  border: none;
  border-radius: 10px;
  color: white;
  font-weight: 600;
  cursor: pointer;
  transition: 0.3s;
}

button:hover {
  transform: scale(1.05);
}
```

OUTPUT



127.0.0.1:5000

⛅ Weather Outfit Recommender

Get the perfect outfit for today's weather

Temperature (°C)

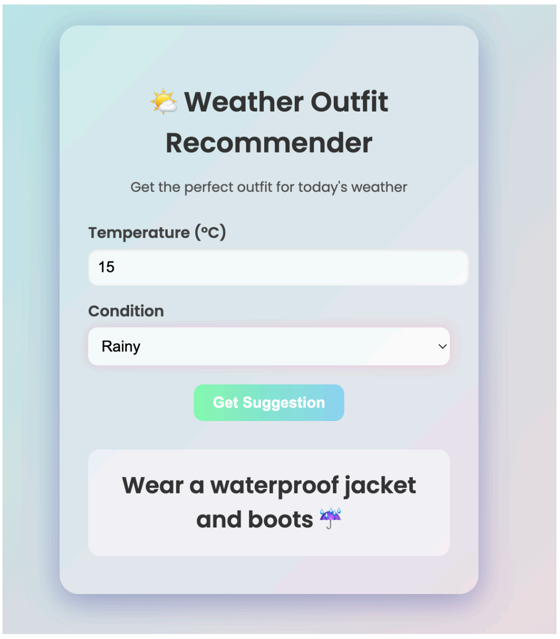
e.g. 25

Condition

Sunny

Get Suggestion

A shirt and jeans would be perfect ⛅


⛅ Weather Outfit Recommender

Get the perfect outfit for today's weather

Temperature (°C): 15

Condition: Rainy

Wear a waterproof jacket and boots ☔

9. Conclusion

The Weather Outfit Recommender web app successfully demonstrates how Flask can be used to build interactive web applications that combine backend logic with frontend presentation.
By entering simple weather parameters, users can instantly receive clothing suggestions. The project showcases the integration of Flask templates, Python logic, and responsive web design.

10. Future Enhancements

- Connect the app to a real-time weather API (like OpenWeatherMap)
- Add outfit images for visual recommendations
- Expand conditions (Snowy, Cloudy, Humid)