Creating a Smart Weather Dashboard involves developing an application that displays real-time weather information for different locations. Below is an outline and example code using Python with Flask for the backend and HTML/CSS/JavaScript for the frontend, integrating with a weather API (e.g., OpenWeatherMap).

---

1. Prerequisites

Python 3.x

Install necessary libraries:

pip install flask requests

Sign up for a free weather API key from OpenWeatherMap.

---

2. Backend Code (Flask)

```
# app.py
from flask import Flask, render_template, request, jsonify
import requests

app = Flask(__name__)

# OpenWeatherMap API key
API_KEY = "your_openweathermap_api_key"
BASE_URL = "http://api.openweathermap.org/data/2.5/weather"

@app.route("/")
def index():
    return render_template("index.html")

@app.route("/get_weather", methods=["POST"])
def get_weather():
    city = request.form.get("city")
    if not city:
        return jsonify({"error": "City is required!"}), 400
```

```
    # Fetch weather data
    params = {"q": city, "appid": API_KEY, "units": "metric"}
    response = requests.get(BASE_URL, params=params)
    data = response.json()

    if response.status_code == 200:
        weather = {
            "city": data["name"],
            "temperature": data["main"]["temp"],
            "description": data["weather"][0]["description"],
            "icon": data["weather"][0]["icon"]
        }
        return jsonify(weather)
    else:
        return jsonify({"error": "City not found!"}), 404

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

---

3. Frontend Code

HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Smart Weather Dashboard</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Smart Weather Dashboard</h1>
        <form id="weatherForm">
            <input type="text" id="city" name="city" placeholder="Enter city" required>
            <button type="submit">Get Weather</button>
        </form>
        <div id="weatherResult" class="hidden">
            <h2 id="cityName"></h2>
            <p><span id="temperature"></span>°C</p>
```

```
            <p id="description"></p>
            <img id="icon" alt="Weather Icon">
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

CSS (styles.css)

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    background-color: #f4f4f9;
    margin: 0;
    padding: 0;
}

.container {
    max-width: 600px;
    margin: 50px auto;
    padding: 20px;
    background: #ffffff;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    border-radius: 8px;
}

h1 {
    margin-bottom: 20px;
}

form {
    margin-bottom: 20px;
}

input {
    padding: 10px;
    width: 70%;
    margin-right: 10px;
}
```

```
button {
    padding: 10px 15px;
    background-color: #007bff;
    color: #fff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

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

.hidden {
    display: none;
}
```

JavaScript (script.js)

```
document.getElementById("weatherForm").addEventListener("submit", async (event) => {
    event.preventDefault();

    const city = document.getElementById("city").value;
    const resultDiv = document.getElementById("weatherResult");

    try {
        const response = await fetch("/get_weather", {
            method: "POST",
            headers: { "Content-Type": "application/x-www-form-urlencoded" },
            body: `city=${encodeURIComponent(city)}`,
        });

        if (!response.ok) {
            throw new Error("City not found!");
        }

        const data = await response.json();

        document.getElementById("cityName").textContent = data.city;
        document.getElementById("temperature").textContent = data.temperature;
        document.getElementById("description").textContent = data.description;
```

```
        document.getElementById("icon").src =
`http://openweathermap.org/img/wn/${data.icon}.png`;

        resultDiv.classList.remove("hidden");
    } catch (error) {
        alert(error.message);
        resultDiv.classList.add("hidden");
    }
});
```

---

4. Running the Application

1. Start the Flask server:

```
python app.py
```

2. Open your browser and visit http://127.0.0.1:5000.

This code provides a basic, responsive weather dashboard. You can expand the project by adding features like a 5-day forecast, geolocation-based weather, or a favorite cities list.