Smart Health Tracker - Complete Documentation

A beginner-to-advanced guide to building a health risk prediction web app using Flask, HTML, CSS, JavaScript, GSAP, Three.js, and Git.

Table of Contents

---

1. Introduction

The **Smart Health Tracker** is a web-based health monitoring application that predicts health risks based on user input such as age, BMI, and activity level. It uses a machine learning model for health risk prediction and offers an engaging UI with animations.

This documentation provides a step-by-step guide for beginners to advanced developers to build, understand, and improve the project.

---

2. Project Overview

The application consists of:

- A **backend** (Flask) that processes user input and predicts health risks using a trained ML model.
- A **frontend** (HTML, CSS, JavaScript, GSAP, Three.js) that provides an interactive UI.
- **Git & GitHub** for version control.

---

3. Technologies Used

Backend (Python & Flask)

- Flask
- Pandas
- Scikit-Learn (for training the ML model)
- Pickle (for saving and loading the model)

Frontend (HTML, CSS, JS, GSAP, Three.js)

- HTML (structure)
- CSS (styling)
- JavaScript (dynamic interactions)
- GSAP (smooth animations)
- Three.js (3D elements)

Tools & Deployment

- Git & GitHub (version control)
- Docker (optional for containerization)
- AWS/Heroku (optional for deployment)

---

4. Features

✅ Predicts health risks based on age, BMI, and activity level
✅ Interactive UI with animations (GSAP, Three.js)
✅ Dark mode & light mode toggle
✅ Seasonal background effects (snow in winter, sun in summer)
✅ User-friendly, responsive, and engaging UI

---

5. Prerequisites

Before starting, ensure you have:

- **Python (3.x)** installed
- **Flask, Pandas, Scikit-Learn** installed (`pip install flask pandas scikit-learn`)
- **Basic HTML, CSS, JavaScript** knowledge
- **Git & GitHub** set up

---

6. Setting Up the Project

1. Clone the Repository

git clone https://github.com/MishraJi-Devloper/Smart-Health-Tracke.git
cd SmartHealthTracker

2. Create a Virtual Environment

python -m venv venv
source venv/bin/activate # (Mac/Linux)
venv\Scripts\activate # (Windows)

3. Install Dependencies

pip install -r requirements.txt

---

7. Training the Machine Learning Model

To predict health risks, we need a trained machine learning model.

1. Prepare the Dataset

Store health-related data in `health_data.csv`. Example:

Age	BMI	Activity Level	Health Risk
25	22	High	Low
40	30	Low	High

2. Train the Model

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import pickle

# Load dataset
df = pd.read_csv("health_data.csv")

# Features and target variable
X = df[['Age', 'BMI', 'Activity Level']]
y = df['Health Risk']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Save model
pickle.dump(model, open("health_risk_model.pkl", "wb"))
```

---

8. Building the Backend (Flask)

1. Create `app.py`

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

app = Flask(__name__)

# Load trained model
model = pickle.load(open("health_risk_model.pkl", "rb"))
```

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

@app.route('/predict', methods=['POST'])
def predict():
    age = int(request.form['age'])
    bmi = float(request.form['bmi'])
    activity = request.form['activity']

    # Convert activity level to numerical values
    activity_map = {"Low": 0, "Medium": 1, "High": 2}
    activity = activity_map.get(activity, 1)

    prediction = model.predict([[age, bmi, activity]])[0]

    return jsonify({"health_risk": prediction})

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

---

9. Designing the Frontend

1. Create `index.html`

```
<form id="predictForm">
  <input type="text" name="age" placeholder="Enter Age">
  <input type="text" name="bmi" placeholder="Enter BMI">
  <select name="activity">
    <option>Low</option>
    <option>Medium</option>
    <option>High</option>
  </select>
  <button type="submit">Predict</button>
</form>
<p id="result"></p>
<script src="app.js"></script>
```

2. Add JavaScript (`app.js`)

```
document.getElementById("predictForm").onsubmit = function(e) {
  e.preventDefault();
  fetch('/predict', {
    method: "POST",
    body: new FormData(this)
  })
  .then(response => response.json())
  .then(data => document.getElementById("result").innerText = "Health Risk: " +
data.health_risk);
};
```

---

10. Mistakes to Avoid

❌ **Not installing dependencies** (`pip install -r requirements.txt`)
❌ **Not activating virtual environment** (`venv\Scripts\activate`)
❌ **Incorrect file paths for ML model (`pkl` file not found error)**
❌ **Not testing API routes properly** (`app.py` must be running)
❌ **Incorrect HTML form field names (must match Flask backend request names)**

---

11. Deploying the Application

1. Use **Flask + Gunicorn** for production.
2. Deploy on **Heroku / AWS / Render**.
3. Use **Docker** for containerization.

---

12. Future Improvements

✅ Add **user authentication**
✅ Improve **UI animations with GSAP**
✅ Deploy as **mobile-friendly web app**

---

13. Conclusion

Congratulations! 🎉 You have successfully built a **health prediction web app** from scratch. Keep improving, try adding more features, and explore **advanced Flask, React, and AI integrations**! 🚀
