app.py

```
from flask import Flask, render_template, request, jsonify
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
app = Flask(__name__)
dataset_path = "DisasterTweets.csv"
df = pd.read_csv(dataset_path, dtype={"ID": str})
TWEET_COLUMN = "Tweets"
CATEGORY_COLUMN = "Disaster"
ID_COLUMN = "ID"
df = df[[TWEET_COLUMN, CATEGORY_COLUMN, "Name", "UserName", "Timestamp", "Tags", "Tweet Link", ID_COLUMN]].dropna()
# Preprocess data
X = df[TWEET_COLUMN]
y = df[CATEGORY_COLUMN]
# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
# Vectorize text
vectorizer = TfidfVectorizer(stop_words="english")
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
# Train Naive Bayes classifier
model = MultinomialNB()
model.fit(X_train_vec, y_train)
y_pred = model.predict(X_test_vec)
# Performance evaluation
```

```
precision = precision_score(y_test, y_pred, average='weighted', zero_division=1)
recall = recall_score(y_test, y_pred, average='weighted', zero_division=1)
f1 = f1_score(y_test, y_pred, average='weighted', zero_division=1)
accuracy = accuracy_score(y_test, y_pred)
# Print evaluation metrics
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"Accuracy: {accuracy}")
# Predict on test set
df_test = pd.DataFrame({"Tweet": X_test, "Category": y_test})
df_test["Predicted_Category"] = model.predict(X_test_vec)
# Route for the homepage
@app.route('/')
def home():
    # Group tweets by their predicted categories
    grouped_tweets = df_test.groupby("Predicted_Category")["Tweet"].apply(list).to_dict()
    return render_template("index.html", grouped_tweets=grouped_tweets, df=df)
# Route to view tweet details
@app.route('/tweet/<tweet_id>')
def tweet_details(tweet_id):
    tweet_data = df[df[ID_COLUMN] == tweet_id]
    if tweet_data.empty:
        return render_template("error.html", message="Tweet not found!")
    tweet_dict=tweet_data.iloc[0].to_dict()
    return render_template("tweet_details.html", tweet_data=tweet_dict,
category=tweet_dict[CATEGORY_COLUMN])
# Route for email subscription
@app.route('/subscribe', methods=['POST'])
def subscribe():
    email = request.form.get("email")
    if not email:
        return jsonify({"error": "Email is required"}), 400
    # Save email to a file (or database in production)
    with open("subscribers.txt", "a") as f:
        f.write(f"{email}\n")
    return jsonify({"message": "Subscribed successfully!"})
if __name__ == "__main__":
    app.run(debug=True)
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Disaster Tweet Tracker</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
    <header>
        <h1>Disaster Tweet Tracker</h1>
    </header>
    <div class="container">
        <!-- Grouped Tweets -->
        {% for category, tweets in grouped_tweets.items() %}
        <div class="category">
            <h2>{{ category }}</h2>
            {% for tweet in tweets %}
            <div class="tweet">
```

```
                {% set tweet_id = df[df['Tweets'] == tweet]['ID'].iloc[0] if not df[df['Tweets'] == tweet].empty else None %}
                {% if tweet_id %}
                <a href="/tweet/{{ tweet_id }}">{{ tweet }}</a>
                {% else %}
                {{ tweet }} (Tweet ID not found)
                {% endif %}
            </div>
            {% endfor %}
        </div>
        {% endfor %}
        <!-- Subscription Form -->
        <div class="subscribe-form">
            <h3>Subscribe for Alerts</h3>
            <form id="subscribe-form">
                <input type="email" name="email" placeholder="Enter your email" required>
                <button type="submit">Subscribe</button>
            </form>
            <p id="subscription-message"></p>
        </div>
    </div>
    <script>
        document.getElementById('subscribe-form').addEventListener('submit', function(event) {
            event.preventDefault();
            const formData = new FormData(this);
            fetch('/subscribe', {
                method: 'POST',
                body: formData
            })
            .then(response => response.json())
            .then(data => {
                document.getElementById('subscription-message').textContent = data.message;
            });
        });
    </script>
</body>
</html>
```

tweet_details.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tweet Details</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
    <header>
        <h1>Tweet Details</h1>
    </header>
    <div class="tweet-details">
        <p><strong>Name:</strong> {{ tweet_data['Name'] }}</p>
        <p><strong>Username:</strong> {{ tweet_data['UserName'] }}</p>
        <p><strong>Timestamp:</strong> {{ tweet_data['Timestamp'] }}</p>
        <p><strong>Tweet:</strong> {{ tweet_data['Tweets'] }}</p>
        <p><strong>Disaster Category:</strong> {{ category }}</p>
        <p><strong>Tags:</strong> {{ tweet_data['Tags'] }}</p>
        <p><strong>Tweet ID:</strong> {{ tweet_data['ID'] }}</p>
        <p><strong>Tweet Link:</strong>
```

```
      <a href="{{ tweet_data['Tweet Link'] }}" target="_blank">
        {{ tweet_data['Tweet Link'] }}
      </a>
    </p>
  <footer>
    <button onclick="window.location.href='/'">Go Back to Dashboard</button>
  </footer>
  </div>
</body>
</html>
```

Error.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Error</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='css/styles.css') }}">
</head>
<body>
  <header>
    <h1>Error</h1>
  </header>
  <div class="error-container">
    <p>{{ message }}</p>
    <a href="/">Go Back to Dashboard</a>
  </div>
</body>
</html>
```