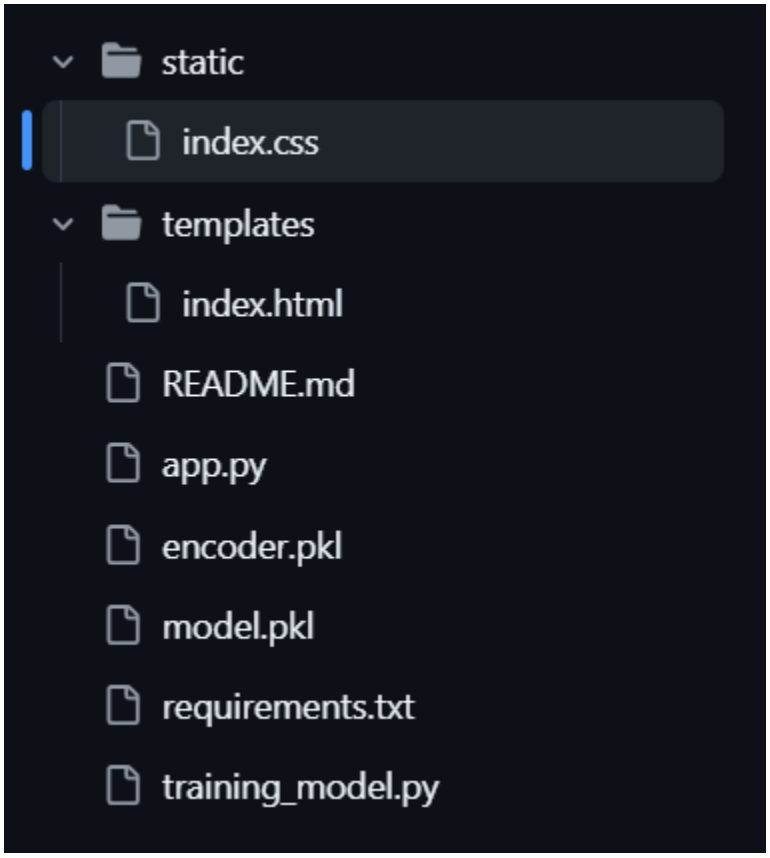
Project Source Code

Project structure



Source Code

```
body {
  font-family: Arial, sans-serif;
  text-align: center;
  background-color: #f2f2f2;
}

form {
  margin-top: 30px;
}

input, select {
```

```
    display: block;
    margin: 10px auto;
    padding: 10px;
    width: 250px;

}
input::-webkit-inner-spin-button,
input::-webkit-outer-spin-button{
    display: none;
}
button {
    padding: 10px 20px;
    background-color: #4CAF50;
    color: white;
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}

h2 {
    margin-top: 20px;
}
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Fraud Detection</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='index.css') }}">
</head>
<body>

  <h1>Online Payment Fraud Detection</h1>

  <form action="/predict" method="post">

    <input type="number" name="step" placeholder="Step" required>

    <select name="type" required >
      <option >choose</option>
      <option value="PAYMENT">PAYMENT</option>
      <option value="TRANSFER">TRANSFER</option>
      <option value="CASH_OUT">CASH_OUT</option>
      <option value="DEBIT">DEBIT</option>
      <option value="CASH_IN">CASH_IN</option>
    </select>

    <input type="number" step="any" name="amount" placeholder="Amount"
required>
```

```
    <input type="number" step="any" name="oldbalanceOrg" placeholder="Old Balance Origin" required>

    <input type="number" step="any" name="newbalanceOrig" placeholder="New Balance Origin" required>

    <input type="number" step="any" name="oldbalanceDest" placeholder="Old Balance Destination" required>

    <input type="number" step="any" name="newbalanceDest" placeholder="New Balance Destination" required>

    <button type="submit">Predict</button>

  </form>

  {% if prediction_text %}
    <h2>{{ prediction_text }}</h2>
  {% endif %}

</body>
</html>
```

requirements.txt

```
flask
pandas
scikit-learn
joblib
numpy
gunicorn
```

Training_model.py

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import joblib


data = pd.read_csv('data.csv')
data = data.dropna()


data = data.sample(200000, random_state=42)


le = LabelEncoder()
data['type'] = le.fit_transform(data['type'])


X = data[['step', 'type', 'amount',
        'oldbalanceOrg', 'newbalanceOrig',
        'oldbalanceDest', 'newbalanceDest']]
y = data['isFraud']
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = RandomForestClassifier(
    n_estimators=100,
    class_weight='balanced',
    random_state=42,
    n_jobs=-1
)

model.fit(X_train, y_train)



joblib.dump(model, 'model.pkl')
joblib.dump(le, 'encoder.pkl')

print("Model trained ")
```

app.py

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

app = Flask(__name__)

model = joblib.load('model.pkl')
le = joblib.load('encoder.pkl')

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

@app.route('/predict', methods=['POST'])
def predict():
    step = int(request.form['step'])
    type_val = request.form['type']
    amount = float(request.form['amount'])
    oldbalanceOrg = float(request.form['oldbalanceOrg'])
    newbalanceOrig = float(request.form['newbalanceOrig'])
    oldbalanceDest = float(request.form['oldbalanceDest'])
    newbalanceDest = float(request.form['newbalanceDest'])

    type_encoded = le.transform([type_val])[0]

    new_data = [[step, type_encoded, amount,
                 oldbalanceOrg, newbalanceOrig,
                 oldbalanceDest, newbalanceDest]]
```

```
    pred = model.predict(new_data)

    result = "Fraud Transaction" if pred[0] == 1 else "Not a Fraud Transaction "

    return render_template("index.html", prediction_text=result)

# import os

# if __name__ == "__main__":
#     port = int(os.environ.get("PORT", 10000))
#     app.run(host="0.0.0.0", port=port)
```