Backend code-app.py

```
from flask import Flask, render_template, request

import torch

import torch.nn.functional as F

from torch_geometric.nn import GCNConv

from torch_geometric.data import Data

import numpy as np


app = Flask(__name__)


# ----------------------------

# Define GCN Model

# ----------------------------

class GCN(torch.nn.Module):

    def __init__(self, num_features, hidden_dim, num_classes):

        super(GCN, self).__init__()

        self.conv1 = GCNConv(num_features, hidden_dim)

        self.conv2 = GCNConv(hidden_dim, num_classes)


    def forward(self, data):

        x, edge_index = data.x, data.edge_index

        x = self.conv1(x, edge_index)

        x = F.relu(x)

        x = self.conv2(x, edge_index)

        return F.log_softmax(x, dim=1)


# Instantiate model (untrained demo)

model = GCN(num_features=50, hidden_dim=64, num_classes=2)
```

```
model.eval()

# ----------------------------
# Prediction Function
# ----------------------------
def predict_fake_news(news_text):
    # For demonstration: generate random graph data
    num_nodes = 100
    num_features = 50
    num_edges = 300

    x = torch.randn((num_nodes, num_features))
    edge_index = torch.randint(0, num_nodes, (2, num_edges))
    data = Data(x=x, edge_index=edge_index)

    out = model(data)
    pred = torch.argmax(out.mean(dim=0)).item()

    return "Fake News ❌" if pred == 0 else "Real News ✅"

# ----------------------------
# Flask Routes
# ----------------------------
@app.route('/', methods=['GET', 'POST'])
def index():
    result = None
    if request.method == 'POST':
        text = request.form['news']
        result = predict_fake_news(text)
    return render_template('index.html', result=result)


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