```
from flask import Flask, request, jsonify

import torch

from model import NeuralNet

from nltk_utils import bag_of_words, tokenize

import json

from flask_cors import CORS

import logging


app = Flask(_name_)

cors = CORS(app, resources={r"/chat": {"origins": "http://localhost:3000"}})


# Configure logging

logging.basicConfig(level=logging.INFO)

logger = logging.getLogger(_name_)


# Load model and data

FILE = "data.pth"

intents_file = "intents.json"
```

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')


data = torch.load(FILE)

input_size = data["input_size"]

hidden_size = data["hidden_size"]

output_size = data["output_size"]

all_words = data['all_words']

tags = data['tags']

model_state = data["model_state"]


model = NeuralNet(input_size, hidden_size, output_size).to(device)

model.load_state_dict(model_state)

model.eval()


with open(intents_file, 'r') as file:

    intents = json.load(file)
```

```
bot_name = "Bot"


# Define chat endpoint
@app.route('/chat', methods=['POST'])
def chat():
    try:
        data = request.get_json()
        message = data['message']

        sentence = tokenize(message)
        X = bag_of_words(sentence, all_words)
        X = X.reshape(1, X.shape[0])
        X = torch.from_numpy(X).to(device)

        output = model(X)
        _, predicted = torch.max(output, dim=1)

        tag = tags[predicted.item()]
```

```
        probs = torch.softmax(output, dim=1)
        prob = probs[0][predicted.item()]
        logger.info(f"Probability: {prob.item()}")
        print(prob.item())
        if prob.item() >=0.5:
            for intent in intents['intents']:
                if tag == intent["intent"]:
                    if tag == "scheme_application":
                        additional_schemes = intent['additional_schemes']
                        response = f"{bot_name}: Here are some additional schemes:\n"
                        for scheme in additional_schemes:
                            title = scheme['title']
                            how_to_avail = scheme['how_to_avail']
                            description = scheme['description']
                            response += f"Title: {title}\nHow to avail:
{how_to_avail}\nDescription: {description}\n"
                    else:
                        response = f"{bot_name}: {(intent['response'])}"
```

```
        else:
            response = f"{bot_name}: I do not understand..."

        logger.info(f"Response: {response}")
        return jsonify({"message": response})

    except Exception as e:
        logger.exception("An error occurred during chat processing")
        return jsonify({"message": "An error occurred. Please try again later."})

if _name_ == '_main_':
    app.run(debug=True)
```