from flask import Flask, jsonify, render\_template

from flask\_socketio import SocketIO, emit

import random

import time

import threading

app = Flask(\_\_name\_\_)

socketio = SocketIO(app)

# Modem signal data (could be more complex in real-world applications)

modem\_signals = {

"modem1": {"frequency": 0.5, "signal\_strength": 75},

"modem2": {"frequency": 0.6, "signal\_strength": 80},

"modem3": {"frequency": 0.45, "signal\_strength": 70}

}

# Simulate real-time modem signal updates

def simulate\_signal():

while True:

time.sleep(5) # Simulate low-frequency signal update every 5 seconds

# Randomly select a modem to update its signal data

modem\_id = random.choice(list(modem\_signals.keys()))

modem\_signals[modem\_id]["signal\_strength"] = random.randint(60, 100)

modem\_signals[modem\_id]["frequency"] = round(random.uniform(0.4, 1.0), 2)

print(f"Updated {modem\_id}: {modem\_signals[modem\_id]}")

# Emit updated signal data to the frontend

socketio.emit('signal\_update', modem\_signals)

# Start the simulation in a separate thread

threading.Thread(target=simulate\_signal, daemon=True).start()

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/get\_signal\_data', methods=['GET'])

def get\_signal\_data():

return jsonify(modem\_signals)

# WebSocket to handle signal transmission from CesiumJS

@socketio.on('send\_signal')

def handle\_send\_signal(data):

print(f"Received signal data from Cesium: {data}")

# Handle data from Cesium if necessary, e.g., logging or processing it.

if \_\_name\_\_ == '\_\_main\_\_':

socketio.run(app, debug=True)