ДОДАТОК А

Лістинг файлу main.py

Лістинг А.1 – Програмний код файлу main.py

```
from flask import Flask, request, render_template, session, redirect, url_for
import folium
from folium import plugins
import pandas as pd
import datetime
import requests
import json
import os
import joblib

app = Flask(__name__)
app.secret_key = 'your_secret_key'
saved_routes = {}

models = {
    'LightGBM': joblib.load('model_lgb.pkl'),
    'XGBoost': joblib.load('model_xgb.pkl'),
    'CatBoost': joblib.load('model_cat.pkl')
}

def get_direction_icon(maneuver_type, modifier=None):
    if maneuver_type == "arrive":
        return "fa-flag-checkered"
    elif maneuver_type == "depart":
        return "fa-play"
    elif maneuver_type == "turn":
        if modifier == "left":
```

```
                return "fa-arrow-left"
            elif modifier == "right":
                return "fa-arrow-right"
            elif modifier == "slight left":
                return "fa-arrow-alt-circle-left"
            elif modifier == "slight right":
                return "fa-arrow-alt-circle-right"
            elif modifier == "sharp left":
                return "fa-arrow-circle-left"
            elif modifier == "sharp right":
                return "fa-arrow-circle-right"
            elif modifier == "uturn":
                return "fa-arrow-circle-up"
        elif maneuver_type == "new name" or maneuver_type ==
"continue":
            return "fa-arrow-up"
        elif maneuver_type in ["merge", "on ramp", "off ramp"]:
            return "fa-sign"
        elif maneuver_type == "roundabout":
            return "fa-sync"
        return "fa-road"

    def format_instruction(maneuver, distance):
        maneuver_type = maneuver.get('type', '')
        modifier = maneuver.get('modifier', '')
        if 'instruction' in maneuver:
            instruction = maneuver['instruction']
        else:
            if maneuver_type == "turn":
                instruction = f"Поверніть {modifier}" if
modifier else "Поверніть"
            elif maneuver_type == "depart":
                instruction = "Початок маршруту"
```

```
            elif maneuver_type == "arrive":
                instruction = "Прибуття до місця призначення"
            elif maneuver_type == "roundabout":
                instruction = f"На кільцевій розв'язці виїзд {maneuver.get('exit', 1)}"
            elif maneuver_type == "merge":
                instruction = "Злиття з дорогою"
            elif maneuver_type == "on ramp":
                instruction = "В'їзд на рампу"
            elif maneuver_type == "off ramp":
                instruction = "З'їзд з рампи"
            elif maneuver_type in ["continue", "new name"]:
                instruction = "Продовжуйте рух прямо"
            else:
                instruction = f"Рухайтесь {modifier}" if modifier else "Рухайтесь"
        instruction = instruction.replace("Turn left", "Поверніть ліворуч").replace("Turn right", "Поверніть праворуч").replace("Continue", "Продовжуйте рух")
        return {
            "instruction": instruction,
            "distance": round(distance),
            "type": maneuver_type,
            "modifier": modifier,
            "icon": get_direction_icon(maneuver_type, modifier)
        }

    @app.route('/', methods=['GET', 'POST'])
    def index():
        delay = None
        total_duration_min = None
        arrival_str = None
        osrm_time = None
```

```
        instructions = []
        total_distance = None
        error = None
        selected_model_name = request.form.get('model',
session.get('model', 'CatBoost'))
        session['model'] = selected_model_name
        model = models[selected_model_name]
        start_coords = session.get('start_coords', [50.4501,
30.5234])
        end_coords = session.get('end_coords', [50.4530,
30.5235])
        departure = request.form.get('departure',
session.get('departure', '08:00'))
        center_lat = (start_coords[0] + end_coords[0]) / 2
        center_lng = (start_coords[1] + end_coords[1]) / 2
        m = folium.Map(location=[center_lat, center_lng],
zoom_start=13)
        plugins.Fullscreen().add_to(m)
        plugins.LocateControl().add_to(m)

        if request.method == 'POST':
            try:
                start = list(map(float,
request.form['start'].split(',')))
                end = list(map(float,
request.form['end'].split(',')))
                session['start_coords'] = start
                session['end_coords'] = end
                session['departure'] = departure
                url = f"http://router.project-
osrm.org/route/v1/driving/{start[1]},{start[0]};{end[1]},{end[
0]}?overview=full&geometries=geojson&steps=true&annotations=tr
ue"
                response = requests.get(url)
                data = response.json()
                if data.get('code') != 'Ok':
```

```
                    raise Exception(f"OSRM API error: {data.get('message', 'Unknown error')}")
                route = data['routes'][0]
                geometry = route['geometry']['coordinates']
                steps = route['legs'][0]['steps']
                duration_sec = route['duration']
                distance = route['distance'] / 1000
                total_distance = round(distance, 2)
                turns = len(steps)
                route_line = folium.PolyLine(
                    locations=[(lat, lon) for lon, lat in geometry],
                    color='#2c7be5',
                    weight=5,
                    opacity=0.8
                ).add_to(m)
                plugins.PolyLineTextPath(
                    route_line,
                    '➔',
                    repeat=True,
                    offset=8,
                    attributes={'fill': '#2c7be5', 'font-weight': 'bold', 'font-size': '14'}
                ).add_to(m)
                for step in steps:
                    instruction = format_instruction(step['maneuver'], step['distance'])
                    instruction['lat'] = step['maneuver']['location'][1]
                    instruction['lon'] = step['maneuver']['location'][0]
                    instructions.append(instruction)
                    if instruction['type'] in ['turn', 'roundabout', 'merge', 'on ramp', 'off ramp'] and instruction['distance'] > 50:
```

```
                        icon_color = 'blue'
                        if              'left'              in
str(instruction['modifier']).lower():
                            icon_color = 'orange'
                        elif            'right'             in
str(instruction['modifier']).lower():
                            icon_color = 'green'

folium.Marker(location=[instruction['lat'],
instruction['lon']],

icon=folium.Icon(color=icon_color, icon="info-sign")).add_to(m)
                hour_min                                    =
datetime.datetime.strptime(departure, "%H:%M")
                df   =   pd.DataFrame([{'distance':   distance,
'turns': turns, 'hour': hour_min.hour}])
                delay = float(model.predict(df)[0])
                osrm_time = duration_sec / 60
                total_duration_min = osrm_time + delay
                arrival_time        =        hour_min        +
datetime.timedelta(minutes=total_duration_min)
                folium.Marker(location=start,
icon=folium.Icon(color="green",                    icon="play",
prefix="fa")).add_to(m)
                folium.Marker(location=end,
icon=folium.Icon(color="red",            icon="flag-checkered",
prefix="fa")).add_to(m)
                m.fit_bounds([start, end], padding=(30, 30))
                delay = round(delay, 2)
                osrm_time = round(osrm_time, 2)
                total_duration_min = round(total_duration_min,
2)
                arrival_str = arrival_time.strftime('%H:%M')
                map_html = m._repr_html_()
                route_id = len(saved_routes) + 1
                saved_routes[route_id] = {
                    'start': start,
```

```
                    'end': end,
                    'center': [center_lat, center_lng],
                    'geometry': [(lat, lon) for lon, lat in
geometry],
                    'delay': delay,
                    'osrm_time': osrm_time,
                    'total_duration_min': total_duration_min,
                    'arrival_str': arrival_str,
                    'total_distance': total_distance,
                    'instructions': instructions,
                    'departure': departure,
                    'selected_model': selected_model_name
                }
                return    redirect(url_for('show_saved_route',
route_id=route_id))
            except Exception as e:
                error = str(e)

        return                  render_template("index.html",
map=m._repr_html_(), delay=delay,

total_duration_min=total_duration_min, arrival_str=arrival_str,
                                osrm_time=osrm_time,
instructions=instructions,
                                departure=departure,
total_distance=total_distance, error=error,

start_coords=json.dumps(start_coords),
end_coords=json.dumps(end_coords),
                                has_route=bool(instructions),
selected_model=selected_model_name,
                                model_names=models.keys())


    @app.route('/<int:route_id>')
    def show_saved_route(route_id):
```

```
        route_data = saved_routes.get(route_id)
        if not route_data:
            return f"Маршрут №{route_id} не знайдено.", 404
        m = folium.Map(location=route_data['center'],
zoom_start=13)
        folium.PolyLine(route_data['geometry'],
color='#2c7be5', weight=5, opacity=0.8).add_to(m)
        folium.Marker(location=route_data['start'],
icon=folium.Icon(color="green", icon="play",
prefix="fa")).add_to(m)
        folium.Marker(location=route_data['end'],
icon=folium.Icon(color="red", icon="flag-checkered",
prefix="fa")).add_to(m)
        return render_template("index.html",
map=m._repr_html_(),
                               delay=route_data['delay'],
total_duration_min=route_data['total_duration_min'],

arrival_str=route_data['arrival_str'],
osrm_time=route_data['osrm_time'],

instructions=route_data['instructions'],
departure=route_data['departure'],

total_distance=route_data['total_distance'], error=None,

start_coords=json.dumps(route_data['start']),

end_coords=json.dumps(route_data['end']),       has_route=True,
static_view=True)


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

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	65 с.
	Інші документи	
2.	Презентаційні матеріали	__ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Янкович С.Є.			Розробка вебзастосунку для оптимального обрахунку логістики із урахуванням поточної завантаженості дорог на основі рандомізованих ансамблевих алгоритмів	Шифр групи	Код напр./спец.
Перевірив		Бібічков І.Є.				ІТШІ-21-1	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Золотухін О.В.					