PLAN PRACY MAGISTERSKIEJ – LUBOŃSKI JAKUB

1. Wstęp

- Cel, zakres, skąd pomysł, po co i dlaczego

2. Przegląd literatury

- Opis problemu VRP i zastosowania

- Ograniczenia, niepewności (ładowność, okna czasowe, pogoda, ruch uliczny) i zastosowania

- Przegląd algorytmów stosowanych do VRP (genetyczny, mrówkowy, Tabu Search, symulowane wyżarzanie)

3. Analiza wymagań i projekt aplikacji

- Co aplikacja ma robić

- Opis danych wejściowych i wyjściowych

- Architektura aplikacji i wybór technologii

- Opis interfejsu użytkownika

4. Model matematyczny problemu

- Formalny opisVRP z uwzględnieniem ograniczeń i niepewności

- Opis sposobu naliczania niepewności trasy (mnożniki, pogoda, ruch, losowość)

5. Implementacja aplikacji

- Struktura kodu i podział na moduły

- Opis integracji z API (OpenStreetMap, OpenRouteService, OpenWeatherMap)

- Buforowanie i przetwarzanie danych

- Implementacja modelowanie niepewności

- Implementacja algorytmów

- Opis wyników, danych wyjściowych aplikacji

6. Porównanie algorytmów

- Porównanie wyników dla różnych algorytmów (najlepsze trasy, czasy działania, liczba naruszeń ograniczeń)

7. Podsumowanie i wnioski

- Podsumowanie wyników pracy i wnioski

- Ograniczenia aplikacji i propozycje możliwości dalszego rozwoju aplikacji

PLAN APLIKACJI:

1. Cel aplikacji:

Aplikacja służy do wyznaczania optymalnych tras dla pojazdów dostawczych, uwzględniając niepewności takie jak pogoda.

2. Funkcjonalności aplikacji:

- Formularz wejściowy dla użytkownika:

- Wprowadzenie adresu magazynu startowego.

- Wprowadzenie listy pojazdów (liczba, pojemność, opcjonalnie typ).

- Wprowadzenie listy punktów dostaw (adresy, opcjonalnie okna czasowe, typ towaru?).

- Możliwość edycji i usuwania danych wejściowych oraz zapisu.

- Geokodowanie adresów:

- Zamiana adresów na współrzędne GPS za pomocą API (OpenStreetMap).

- Pobieranie danych zewnętrznych:

- Pobieranie czasów przejazdu między punktami z API (OpenRouteService/Google Maps).

- Pobieranie prognozy pogody tras/punktów dostaw (OpenWeatherMap API).

- Pobieranie informacji o ruchu drogowym?

- Modelowanie niepewności:

- Modyfikacja czasów przejazdu na podstawie pogody i/lub ruchu.

- Dodanie losowości do czasów przejazdu (symulacja różnych scenariuszy).

- Optymalizacja tras:

- Implementacja algorytmu VRP (genetyczny, mrówkowy, tabuSearch, symulowane wyżarzanie).

- Uwzględnienie ograniczeń pojazdów i punktów dostaw (okna czasowe).

- Możliwość porównania różnych algorytmów.

- Prezentacja wyników:

- Generowanie raportu tras (do pliku PDF/CSV).

- Wyświetlenie optymalnych tras dla każdego pojazdu, wizualizacja tras na mapie

3. Technologie:

- Backend: Python albo algorytmy python, reszta .NET

- Frontend: cos pythonowego albo angular lub blazor, jeśli reszta backendu będzie w .NET

- API:

* współrzędne GPS (OpenStreetMap),
* czasy przejazdu (OpenRouteService/Google Maps), prognoza pogody (OpenWeatherMap),
* opcjonalnie ruch drogowy (Google Maps Traffic).

- Baza danych: Pliki JSON/CSV lub SQLite ( do zapisu tras, współrzędnych, -> dla testowania oraz wyników )

- Wizualizacja: OpenStreetMap, matplotlib

4. Przykładowy przepływ działania aplikacji:

I. Użytkownik wprowadza dane wejściowe (magazyn, pojazdy, punkty dostaw).

II. Aplikacja zamienia adresy na współrzędne GPS.

III. Aplikacja pobiera czasy przejazdu i prognozę pogody, ruch uliczny

IV. Tworzy macierz czasów przejazdu, uwzględniając niepewności.

V. Algorytm optymalizuje trasy uwzględniając nałożone ograniczenia

VI. Wyniki są prezentowane użytkownikowi (lista tras, mapa, raport).