МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, КОМП'ЮТЕРНОЇ ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ
КАФЕДРА ПРИКЛАДНОЇ ІНФОРМАТИКИ

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

_____ __Гамаюн В.П.
(підпис) (ПІБ)

"_____"_____2021р.

ДИПЛОМНИЙ ПРОЕКТ

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ "БАКАЛАВР"

Тема: Програмний модуль визначення раціональних маршрутів для служб таксі.

Виконавець: _____ _Слободянюк Михайло Володимирович
(підпис) **(ПІБ)**

Керівник: _____ Водоп'янов Сергій В'ячеславович
(підпис) **(ПІБ)**

Нормоконтролер: _____ Боровик Володимир Миколайович
(підпис) **(ПІБ)**

Київ 2021

ВСТУП

Послуги таксі відіграють важливу роль у повсякденному житті жителів. На відміну від інших громадських перевезень, таких як автобуси та метро, таксі не рухаються періодично фіксованими маршрутами.

Кожного дня водіям таксі приходиться приймати до тисячі рішень щодовибору маршруту на своєму шляху.

Розуміння поведінки таксистів щодо вибору маршруту все ще є теоретичним та емпіричним викликом [1]. Для цього традиційні підходи будують раціональні вибори моделей та детерміновано прогнозують результатз кінцевого набору виборів.

Визначення маршруту є критично важливим для таких адміністративних цілей як містобудування, управління дорожнім рухом та будівництво інфраструктур.

З розширенням дорожньої мережі останніми роками, збільшується кількість альтернативних маршрутів з одного місця в інше, і тим самим збільшується складність аналізу поведінки на маршруті.

Слідування найкоротшому маршруту традиційно вважається однією зголовних особливостей поведінки щодо вибору маршруту водієм. Найкоротший маршрут передбачає мінімальний час подорожі і зменшує витрати з найменшими можливостями уникнення заторів та частоти аварій [1]. Хоча такий підхід є ефективним, він опускає велику частку показників, такі як стан дорожнього покриття, тип дороги, наявність корок та їх потужність, погодні умови на певних ділянках дороги, обмеження швидкості, рівень безпечності дороги, статистика аварій на певних ділянках дороги, дистанція та час подорожі тощо. Такий перелік показників є невід’ємним та напряму впливає на вибір маршрутів у реальному часі.

Тому задачею даного дослідження є застосування алгоритму визначення раціональних маршрутів для служб таксі, який буде

враховувати вище вказані показники

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ВИЗНАЧЕННЯ РАЦІОНАЛЬНИХ МАРШРУТІВ

1.1 Аналітичний огляд систем визначення раціональних маршрутів для служб таксі

На сьогодні, служби таксі все рідше використовують операторів для оформлення замовлення, та контакту з водієм, великі компанії почали активніше просувати та використовувати веб або мобільні додатки для створення замовлення, відслідковування водія або пасажира та в допомозі прокладання маршруту до клієнта або на шляху до пункту призначення. Такі типи систем ідеально підходять для нових або наявних служб таксі. За допомогою таких додатків зручніше замовити поїздку для клієнтів та розширити базу для таксомоторної компанії [2].

Існує досить багато служб таксі, які перейшли з телефонних замовлень, до мобільних додатків та веб систем. Серед таких служб можна виділити три типи.

Перший тип це приватне таксі, коли водій особисто встановлює правила як вижити на ринку таксі, власні тарифи і він працює без посередників. Другий тип це диспетчерська служба. Вона може або не може бути автоматизована і використовувати GPS трекери, радіостанції або інші доступні засоби. Третє, і поки останнє покоління – це автоматизований посередник [2].

На даний час рішення третього типу є досить популярним. Вони допомагають збільшити кількість замовлень у рази, завдяки тому, що не витрачається час диспетчера на пошук авто, реєстрацію замовлення тощо.

Роль диспетчера в таких системах виконує сам додаток, який здатен здійснити реєстрацію клієнта або водія, прийняти замовлення, підібрати найближчого до клієнта водія, оцінити час прибуття та ціну поїздки, та саме головне допомогти водію обрати раціональний маршрут для поїздки.

Таким чином водієві не потрібно планувати власні маршрути під час поїздки, і його доходи багато в чому визначаються вибором цих маршрутів. Однак водієві нелегко скласти графік і вибрати найкращий маршрут, щоб максимально збільшити заробіток. Тому на допомогу водіям приходять автоматизовані системи визначення раціональних маршрутів.

В світі та на території України можна виділити три основних конкурента на ринку систем визначення маршрутів для служб таксі – це Uber, Uklon та Lyft.

Uber – американська компанія, що створила однойменний мобільний додаток (рис 1.1) для пошуку, виклику та оплати таксі або приватних водіїв. використовує мобільний додаток та є посередником між водіями служби таксі та клієнтами [3].

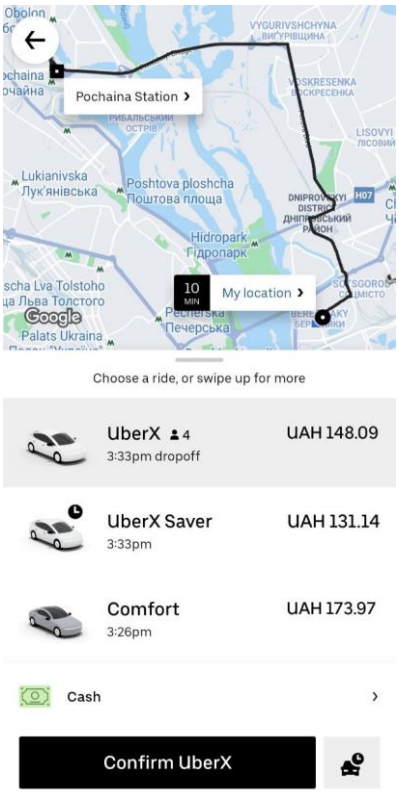


Рис. 1.1. Головне вікно додатку Uber

За допомогою його додатка можливо здійснити реєстрацію, замовлення поїздки, оплату, відслідковування водіїв на мапі в зоні близькій до клієнта та також додаток визначає найкоротший маршрут до клієнта та від місця посадки до точки висадки. Існує дві версії додатка – для водіїв та клієнтів.

Uklon – перший в Україні сервіс виклику авто через інтернет [4]. Uklon дозволяє замовити авто використовуючи як веб так і мобільну версію додатку (рис. 1.2).

Рис. 1.2. Головне вікно додатку

Uklon Серед основних можливостей цієї системи є реєстрація, замовлення та оплата таксі, так як і в системі Uber, в Uklon є також сервіс який відповідає за визначення найкоротшого маршруту для водія, основаного на таких показниках як затори на дорогах, години пік, та кількість запитів. Також в додатку Uklon існує велика кількість додаткових можливостей, таких як вибрати клас авто, замовити послугу «кур'єр», «Англомовний

водій», наявність дитячого сидіння, кондиціонера та можливість перевезення додаткового багажу.

Lyft – є американською компанією, заснованою в Сан – Франциско, яка займається сервісом спільних поїздок [5]. Lyft – друга за величиною компанія, яка займається розвезенням автотранспорту в США з 28 – відсотковою часткою ринку після Uber. Як і інші представники ринку послуг таксі, Lyft (рис. 1.3) пропонує два додатки для використання – для водіїв та для клієнтів.

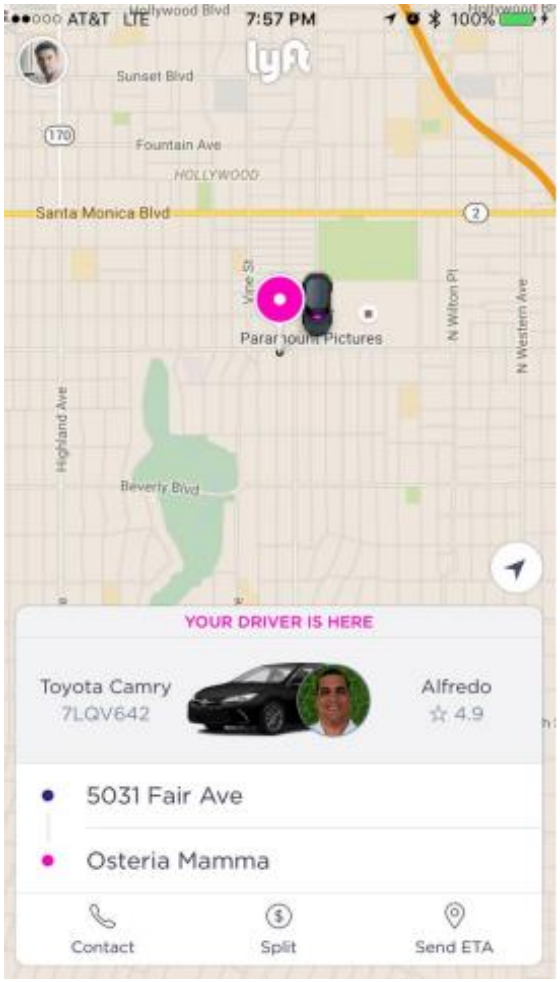


Рис. 1.3. Головне вікно додатку Lyft

Пасажири використовують додаток для створення замовлення на поїздку та оплати, водії використовують додаток для прийняття замовлення, та відстеження найкоротшого маршруту до клієнта, та під час поїздки до точки призначення.

1.2 Відомі підходи до визначення раціональних маршрутів

Проблема визначення раціонального маршруту містить в собі задачу пошуку найкоротшого шляху від заданої початкової вершини до заданої кінцевої вершини.

Для підвищення ефективності роботи таксі слід застосувати алгоритми для знаходження найкоротшого шляху із врахуванням ряду інших важливих чинників (наприклад, стан дорожнього покриття, наявність корок та їх потужність, погодні умови на певних ділянках дороги, рівень безпечності дороги, статистика аварій на певних ділянках дороги тощо).

Існує багато алгоритмів, що дозволяють розв’язати задачу пошуку найкоротшого шляху, проте можна виділити три найбільш популярних та досить ефективних алгоритмів.

Алгоритм Дейкстри – це алгоритм названий на честь його засновника Е. Дейкстра Олівера Дж. (дозволяє знайти найкоротший шлях від однієї вершини графа до усіх інших вершин та надає перевагу шляхам з низькою вартістю) [7].

Цей алгоритм є досить ефективним при використанні його для пошуку найкоротшої відстані в графі від початкової вершини до всіх інших. Але алгоритм не буде працювати якщо граф містить ребра з від’ємною вагою. Найчастіше саме цей алгоритм використовують в протоколах маршрутизації.

Алгоритм такого типу не зовсім підходить для вирішення задачі визначення раціонального маршруту для служб таксі, так як він буде занадто повільним, щоб забезпечувати навігацію зі швидкістю та вартістю яку вимагають служби таксі в реальному часі. Для обчислення маршруту потрібно враховувати лише невеликий розділ карти між початковою точкою, кінцевою точкою та пунктами відвідування, якщо такі існують, оскільки все, що знаходиться за межами коридору між цими точками, майже не цікавить.

Алгоритм пошуку A* є розширенням алгоритма Дейкстри, але дозволяє отримати кращі результати за рахунок застосування деякої евристичної функції. При розгляді кожної окремої вершини здійснюється перехід до

сусідньої вершини, прогнозований шлях від якої до потрібної вершини є найкоротшим [8].

Цей метод використовує "евристичну оцінку" h (x) на кожному вузлі до кожного x вузла шляхом сортування найкращої оцінки маршруту через вузли. У процесі цього методу буде відвідуватися кожен вузол у порядку, що випливає з цієї евристичної оцінки.

Не менш важливим фактором даного алгоритму є його гнучкість та можливість додавання додаткових показників до ваг вузлів або ребр графу.

Таким чином буде доцільно врахувати ряд важливих чинників, щоб, наприклад, "заставити" автомобіль рухатися дорогами, уникаючи бездоріжжя, або засипаних снігом доріг, або, скажімо вибирати безпечніші дороги, використовуючи наявну статистику аварій, або вибирати маршрути, які дозволять менше завдавати шкоди автомобілю під час руху (при допустимому збільшенні відстані), і багато іншого. Потрібно зауважити, що алгоритм A* дозволяє знайти шлях до визначеної кінцевої точки, в той час, як алгоритм Дейкстри може знаходити шляхи для усіх пар точок.

Алгоритм Флойда – Воршелла використовується для розв'язання задачі про найкоротший шлях у зваженому графі з додатними або від'ємними вагами ребр [9]. Ключова ідея алгоритму це розбиття процеса пошуку найкоротшого шляху на фази, перед кожною k – фазою (k=1..n) вважається що в матриці відстаней d[][] збережені довжини таких найкоротших шляхів, які містять в якості внутрішніх вершин тільки вершини із множини {1,2..k – 1}. Таким чином вся робота на k – ій фазі – це перебрати всі пари вершин і перерахувати довжину найкоротшого шляху між ними. В результаті після виконання n – ої фази в матриці відстаней буде записана довжина найкоротшого шляху між і і j, або бескінечніть, якщо такого шляху не існує.

Алгоритм Флойда-Воршалла є хорошим для обчислення шляху між усіма парами вершин в щільних графах, в яких більшість або всі пари вершин, з'єднані ребрами.

Він дозволяє отримати найбільшу продуктивність при невеликій кількості вершин у графі, при збільшенні кількості вершин завдання пошуку найкоротшого шляху стає дещо складнішим. До того ж, цей алгоритм як і алгоритм Дейкстри застосовується для знаходження шляхів із одної вершини до всіх інших.

1.3 Постановка задачі дослідження

Відповідно до проведеного аналізу аналогів, було визначено, що діючі системи визначення раціональних маршрутів для служб таксі не враховують достатньої кількості показників, що напряму впливає на ефективність роботи водія.

Тому пропонується застосувати новий підхід до визначення раціональних маршрутів, який буде враховувати різні додаткові вхідні показники, такі як:

- стан дорожнього покриття;
- наявність корок та їх потужність;
- погодні умови на певних ділянках дороги;
- затори;
- рівень безпечності дороги, врахування статистики аварій на певних ділянках дороги.

Відповідно із врахуванням такого підходу та проведеного аналізу наявних аналогів, було вирішено що програмна реалізація повинна відповідати наступним вимогам:

- можливість реєстрації та авторизації як для клієнтського модуля, так і для модуля водія;
- використання GPS - навігації для встановлення місця перебування користувача та місць пересування водіїв таксі;
- можливість перегляду мапи із усіма вільними авто таксі поблизу користувача для клієнтського модуля;

– можливість перегляду мапи із користувачами які шукають авто для модуля водія;
– можливість визначення місця посадки, м ісця призначення, дати та типу авто для поїздки;
– можливість створення поїздки, та відслідковування водія на його шляху до клієнта;
– можливість перегляду часу очікування та ціни поїздки;
– можливість завантаження файлу з додатковими факторами для врахування їх при визначенні раціонального маршруту;
– для модуля водія можливість визначення та перегляд на мапі раціонального маршруту згідно попередньо заданим на мапі місцям;
– для модуля клієнта та водія можливість перегляду даних про минулі поїздки, а саме початкова адреса, пункт призначення, вартість, час, відстань.

Відповідно до вище вказаних вимог можна визначити які сторінки повинні бути у програмі:

– сторінка реєстрації та авторизації;
– головна сторінка, де можна переглянути мапу та здійснити створення поїздки;
– сторінка перегляду маршруту, ціни та часу поїздки;
– сторінка перегляду попередніх поїздок із інформацією вказаною у вимогах;
– сторінка редагування проф.

Висновки до розділу

Обґрунтовано актуальність задачі визначення раціональних маршрутів для служб таксі. Визначено, що на сьогодні даний програмний модуль є невід’ємним помічником для водіїв таксі у підтримці прийнятті рішень щодо визначення раціональних маршрутів.

Здійснено аналітичний огляд аналогічних систем визначення раціональних маршрутів для служб таксі, огляд показав, що аналоги враховують недостатньо показників, які є важливими під час визначення раціонального маршруту.

Виконано аналіз основних алгоритмів пошуку найкоротшого шляху, розглянуто їх переваги та недоліки.

Виконано постановку задачі розробки програмного модуля з рядом вимог, що мають врахувати основні недоліки розглянутих аналогів.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ВИЗНАЧЕННЯ РАЦІОНАЛЬНИХ МАРШРУТІВ

2.1 Аналіз та обґрунтування вибору алгоритму знаходження найкоротшого шляху

Основним показником під час пошуку раціональних маршрутів є довжина шляху, але як показує практика, у великих містах де є одної довжини недостатньо для визначення раціонального маршруту, тому потрібно розглянути алгоритми пошуку найкоротшого у графі.

В дискретній математиці, граф – це сукупність об'єктів із зв'язками між ними [10]. Об'єкти розглядаються як вершини, або вузли графу, а зв'язки — як дуги, або ребра.

При пошуку раціонального маршруту, представлення мапи може змінити ефективність та якість шляху. Тому доцільним буде перетворити мапу на граф, та здійснювати пошук вже у ньому.

У основі задач пошуку маршрутів в графі лежать алгоритми знаходження найкоротшого шляху.

Традиційна задача про найкоротший шлях полягає у знаходженні такого шляху між двома вершинами (або вузлами) графу, що сума ваг ребр з яких він складається є мінімальною [11].

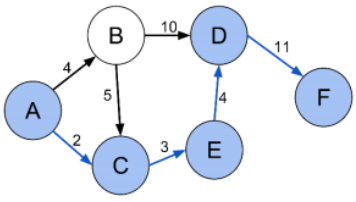


Рис. 2.1. Граф з ребрами

Порівнюючи алгоритми знаходження найкоротшого маршруту при розв’язанні задачі у даному дослідженні потрібно враховувати наступні параметри:

– час роботи
– використання пам’яті
– тип пошуку
– довжина шляху
– ефективність модифікацій
– двома основними задачами пошуку шляху є:
– пошук шляху між двома вузлами на графі
– задача про найкоротший шлях – знайти оптимальний найкоротший шлях

Розглядаючи основні алгоритми, такі як пошук у ширину і пошук у глибину, спрямовані на першу проблему, вичерпуючи всі можливості за допомогою грубого перебору. Починаючи з даного вузла, вони ітераційно досліджують усі потенційні шляхи, доки не досягають цільового вузла. Ці алгоритми використовуються за час

$$O\ (|V| + |E|) \qquad (1)$$

де V – кількість вершин;

E – кількість ребр між вершинами.

Більш складною проблемою є задача про пошук оптимального шляху. Для вирішенні цієї задачі не потрібно вивчати всі можливі шляхи, щоб знайти оптимальний. Алгоритми такі як алгоритм Дейкстри, Белмана-Форда, A* - стратегічно виключають шляхи як за допомогою евристичного алгоритму, так і за допомогою динамічного програмування [12]. Усунувши неможливі шляхи, ці алгоритми можуть досягати низької складності часу, як

$$O\ (|\mathrm{E}|\ \log(|\mathrm{V}|)) \qquad (2)$$

Використання памяті на сьогодні не є проблемою для сучасних компютерів, але цей параметр буде впливати на вартість обслуговування програмного модуля, тому він буде не основним при врахуванні вибору алгоритму.

Розглянувши критерії оціюнювання при виборі алгоритму, можна приступити до їх аналізу.

Алгоритм Дейкстри – алгоритм який відноситься до алгоритмів пошуку в ширину, тобто пошуку шляху з найменшою вагою від вихідного вузла до одного або всіх вузлів у графі. Даний алгоритм спочатку знаходить найближчий вузол до джерела, а потім ітеративно знаходить інші вузли далі і далі. На практиці це створює дерево шляху з мінімальною вагою з початком на початковому вузлі. Якщо потрібно знайти шлях з найменшими витартами до одного вузла, то ітерації можна припинити коли шлях до цього вузла знайдений. Якщо не зупинити пошук, алгоритм обчислить шляхи у всьому графі і таким чином відстані від початкового вузла до всіх інших.

Оригінальна реалізація алгоритму Дейкстри працює у найгіршому випадку із часовою складністю $O(n^2)$, де n – кількість вузлів у графі. Пізніше у 1984 році було винайдено швидшу реалізацію цього алгоритму Фредманом та Тарджаном [12], яка досягнула часу виконання у

$$O(E + N\log N) \tag{3}$$

де E – кількість вершин у графі при найгіршому сценарію.

Алгоритм Дейкстри приймає на вхід граф, початковий вузол, та вузол призначення. Він працює за схожим принципом як пошук у ширину, продвигаючись у графі по всім напрямкам від початкового вузла, відвідуючи вузли що є найближчими до початкового вузла. Як тільки алгоритм досягає вузла призначення, він може відмітити найкоротший шлях від вузла призначення до стартового вузла. При пошуку, алгоритм використовує два набори вузлів, такі як відвідані та передові вузли. Набір передових вузлів це

реалізація структури даних пріоритетна черга [13]. Важливою концепцією алгоритму є G-значення. Кожен вузол в графі має асоційоване G-значення. Для набору відвіданих вузлів G-значення дорівнює довжині найкоротшого шляху назад до початкового вузла.

$$G = (V, E) \tag{4}$$

Для вузлів у передовому наборі, це значення є тільки тимчасовим оцінюванням. Для вузлів які не містяться у цих наборах, значення відсутнє.

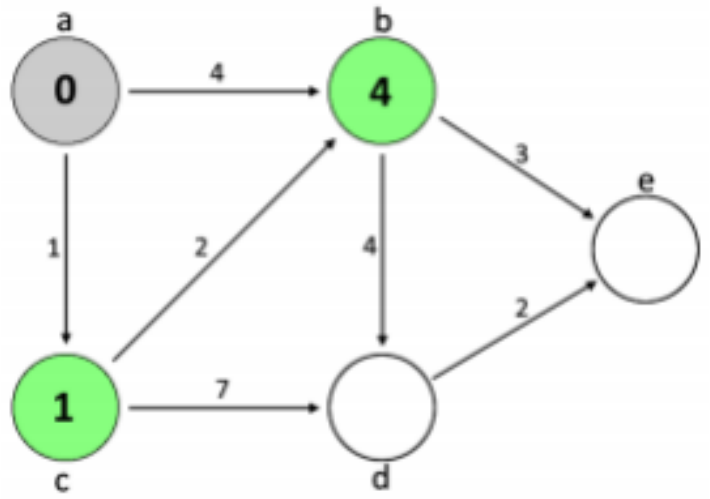


Рис. 2.2. Вигляд вузлів графу після першої ітерації алгоритму Дейкстри

До переваг даного алгоритму можна віднести те, що він працює ефективно коли є декілька вузлів призначення. Алгоритм Дейкстри обирає ребра з найменшою вагою на кожній ітерації та зазвичай покриває велику площину графа. Це є особливо корисно коли є декілька вершин призначення але невідомо яка з них ближче.

Недоліком даного алгоритму є те, що він не робить ніяких оцінок про те як виглядає граф, крім того що граф має додатні ваги ребр.Якщо взяти для прикладу 3 вузла (A, B і C), де вони утворюють непрямий граф з ребрами: AB = 3, AC = 4, BC = -2, оптимальний шлях від A до C коштує 1, а

оптимальний шлях від A до B коштує 2. Якщо застосувати алгоритм Дейкстра: починаючи з A, він спочатку вивчить B, оскільки це найближча вершина. і призначить їй вартість 3, а тому позначить її закритою, що означає, що її вартість ніколи не буде переоцінена. Це означає, що Дайкстра не може оцінити негативні ваги краю.

Алгоритм A* - алгоритм пошуку найкоротшого шляху вперше описаний у 1968 році групою науковців Стендфордського університету. Це модифікація алгоритму Дейкстри що додає евристичну функцію, яка рахує відстань від поточного вузла до пункту призначення, тим самим зменшуючи час роботи.

Алгоритм Дейкстри не використовував місце знаходження вузла призначення для коригування пошуку шляху, що означало що пошук буде проводитись у всіх рівних напрямках.

A* використовує той факт, що всі згенеровані вузли в графі мають свої координати, які можна використати для вимірювання наскільки далеко вузол знаходиться до вузла призначення, використовуючи евристичну функцію.

Якщо правильно обрати функцію яка буде розраховувати дистанцію між вузлами, то A* алгоритм буде виконуватись швидше алгоритма Дейкстри без втрати оптимальності.

Алгоритм A* не змінює загальну структуру алгоритму Дейкстри. Він як і раніше створює набір відвіданих вузлів та набір передових вузлів, і використовує їх для пошуку у графі [14]. На кожній ітерації алгоритм переміщує вузол із передового набору до набору відвіданих вузлів, тільки алгоритм Дейкстри переміщував вузол в якого найменше G-значення, а A* алгоритм переміщує вузол в якого найменше F-значення.

$$F = G + H \tag{5}$$

де H-значення – обраховується використовуючи евристичну функцією, яка оцінює довжину шляху до вузла призначення;

G-значення – ціна переходу до наступної вершини від початкової;

F-значення – є важливим тільки для вузлів у передовому наборі, так як воно використовується при виборі вузла з передового набору до відвіданого.

До переваг даного алгоритму можна віднести те, що він є прямим покращенням алгоритму Дейкстри. Він використовує евристичну функцію щоб коригувати пошук до вузла призначення замість пошуку в ширину по всім вузлам, що призводить до менше відвіданих вузлів та меншого часу виконання пошуку. Що до використання пам’яті, то можна зауважити що чим менше відвіданих вузлів буде відвідано, тим менше пам’яті буде використано.

Рис. 2.3. Порівняння роботи алгоритму Дейкстри та алгоритму A*

Алгоритм Флойда-Воршелла – алгоритм пошуку найкоротшого шляху між усіма вершинами графу. Тобто основну задачу яку вирішує даний алгоритм це задача пошуку найкоротшого шляху між усіма вершинами графу. Він використовує техніку динамічного програмування, і має ЧС O (n^3), де n – кількість ребр у графі, що є достатньо високою швидкістю роботи при графі з невеликою кількістю ребр, але швидкість значно зменшиться при збільшенні ребр графа [15].

Даний алгоритм працює на зваженому графі з невід’ємними вагами ребр. Однак варто зауважити, якщо потрібно, то алгоритм зможе працювати на графі із від’ємними вагами ребр. Якби цикл існував при загальній

від’ємній вазі ребр, тоді для виявлення його може застосовуватись алгоритм Флойда-Воршелла. Якщо від’ємні цикли існують між двома вершинами, то довжина шляху між цими двома вершинами буде від’ємною.

На вхід алгоритму подається презентація графа у вигляді матриці яка містить ваги M, алгоритм обчислює іншу матрицю M', яка представляє граф із усіма довжинами шляхів та містить довжину найкоротшого шляху між двома будь-якими вершинами i та j.

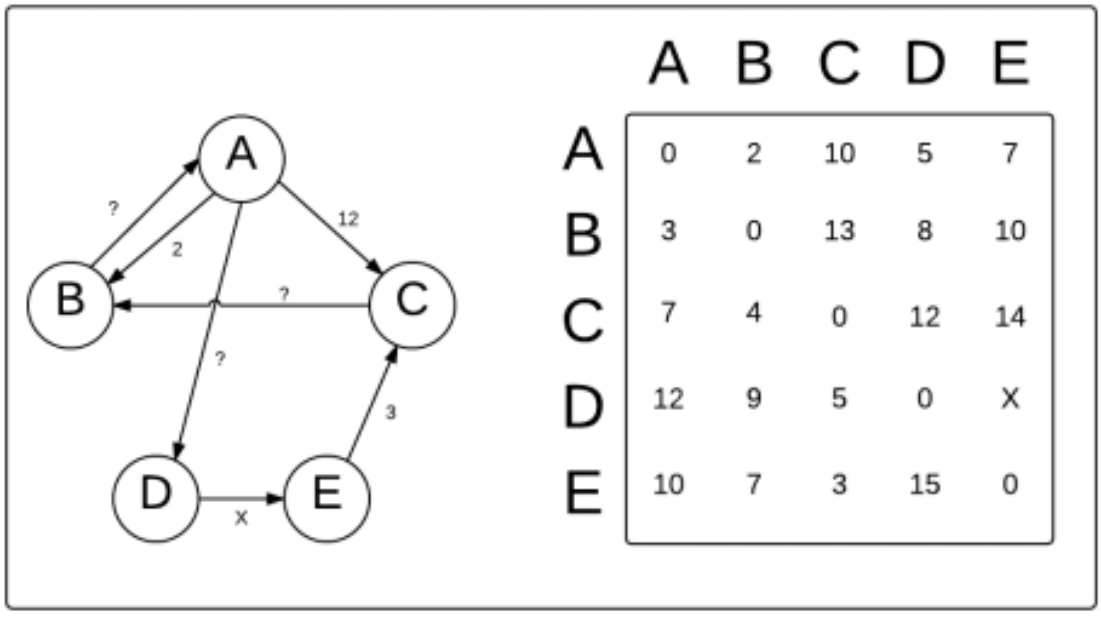


Рис. 2.4. Приклад обчислення матриці у алгоритмі Флойда-Воршелла

Відзначаючи переваги та недоліки даного алгоритму слід зазначити, що цей алгоритм спроектований для пошуку найкоротшого шляху між усіма вершинами графу, тобто не зовсім підходить для вирішення задачі у даному дослідженні, враховуючи ще те, що при переборі дожин ребр для всіх вершин час роботи та кількість використовуваної пам’яті буде збільшуватись відповідно до розміру графа.

Розглянувши детально кожен алгоритм, було прийнято рішення використати саме алгоритм A* як алгоритм знаходження найкоротшого шляху для вирішення задачі пошуку раціональних маршрутів для служб таксі, а саме модифікувати його для покращення пошуку маршрутів згідно

додаткових вхідних показників, які на пряму впливають на вибір маршруту, та будуть додатково обраховані в евристичній функції.

2.2 Аналіз математичної моделі модуля визначення раціональних маршрутів

Оскільки дороги мають різну довжину, для цієї задачі буде доцільно працювати зі зваженими графами.

Зважений граф – граф у якому кожному ребру присвоюється невід'ємне число яке називається вагою. w(e), ребра.

Шлях – це послідовність вершин, така, що:

$$p = (v1 \ldots, vn) \quad (1)$$

такі, що vi ~ vi+1.

Відповідно до цього, тоді довжина шляху p дорівнює:

$$d(p) = \& \text{ Sigma } w\ (ei) \quad (2)$$

Розглядаючи проблему визначення раціонального шляху з s до t та припустимо що ми маємо функцію πt: V→ R, таку що πt (v) повертає оцінену відстань від v до t. Алгоритм A* на кожній ітерації вибирає помічену вершину з найменшим значенням k(v) = ds (v) + πt (t) для обрання наступної.

Для нашого дослідження у формулу евристичної функції необхідно додати додаткові вхідні показники, тоді остаточний вид формули евристичної функції розрахунку ваги ребра графу буде мати вигляд:

$y(x0, x1, x2, x3 \ldots xn) = k0x0 + k1x1 + k2x2 + k3x3 + \cdots + knxn27$

де $x0, x1, x2, x3, xn$ – показники;

$k0, k1, k2, k3, kn,$ - коефіцієнти.

Відповідно, показники з більшими коефіцієнтами будуть найбільше впливати на вихідне значення.

Таким чином, використавши дану формулу як функцію вартості, яка задає кожному ребру його вагу, ми отримаємо зважений граф відповідно до показників та коефіцієнтів, і після цього можна продовжувати пошук раціонального маршруту по звичній формулі:

$$f(n)=g(n)+h(n) \quad (3)$$

Причому функція надання ваги ребрам графа може викликатись повторно, наприклад при різкій зміні погодних умов, дтп чи аварії на дорозі.

У таблиці 1 представлено список показників, які будуть обчислюватись, обраних під час здійснення аналізу.

Показник	Опис
X1	Стан дорожнього покриття
X2	Наявність корок та їх потужність
X3	Погодні умови
X4	Рівень завантаженості дороги, затори
X5	Рівень безпечності дороги

Таблиця 2.1. Список показників які впливають на обчислення ваги ребра графа при пошуку маршруту

2.3 Розробка архітектури та алгоритму функціонування модуля визначення раціональних маршрутів

Після аналізу наявних систем визначення раціональних маршрутів для служб таксі, аналізу вимог та поставленої задачі, було прийнято рішення використати клієнт – серверну архітектуру для розробки програмного модуля.

Даний програмний модуль можна розділити на 4 модуля. Структурна схема функціонування програмного модуля визначення раціональних маршрутів для служб таксі зображена на рисунку 5.

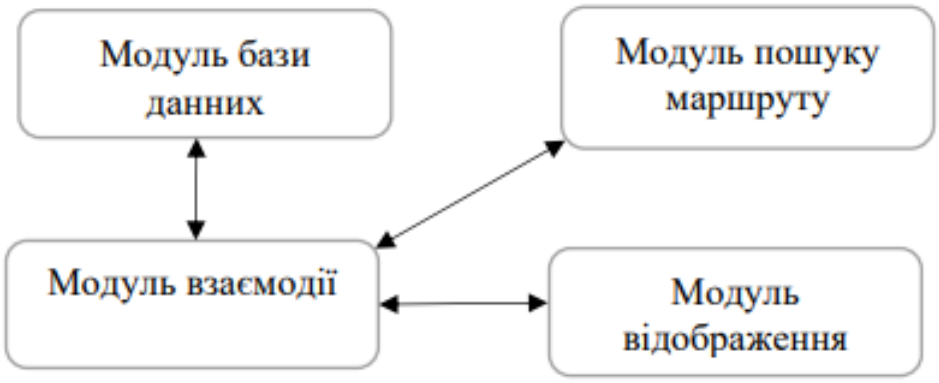


Рис. 2.5. Структурна схема програмного модуля визначення раціональних маршрутів для служб таксі

Модуль бази даних відповідає за створення, збереження, оновлення та видалення даних. Він використовується при реєстрації та авторизації, коли відбувається перевірка чи існує користувач у системі. Цей модуль дозволяє зберігати усю інформацію про користувачів, водіїв та поїздки. Для використання цього модуля, потрібно скористатись модулем взаємодії програми.

Модуль взаємодії програми відповідає взаємодію усіх модулів, у його функціонал входить отримання даних з модуля відображення, перетворення цих даних, та передавання їх у модуль бази даних і модуль пошуку маршруту.

Модуль відображення являє собою будь – яке представлення інформації одержане на виході, дані про користувача отриманні з реєстрації, дані отримані від користувача про поїздку, мапа, маршрут на ній, та дані про минулі поїздки. Через цей модуль користувачі програми взаємодіють як через інтерфейс.

Модуль пошуку маршруту відповідає за створення раціонального маршруту для водіїв, він отримує дані з модуля відображення через модуль взаємодії, такі як інформація про користувача, інформація про поїздку, та інформацію про інші додаткові показники. Отримавши всі необхідні дані модуль виконує необхідні перетворення, враховує всі показники та здійснює пошук раціонального маршруту. Після пошуку маршруту модуль пошуку маршруту передає дані у модуль взаємодії, який у свою чергу передає інформацію у модуль відображення, який відображає знайдений маршрут на мапі, а також обчислену ціну та час поїздки для даного маршруту. В залежності від обраного маршруту користувач та водій можуть або підтвердити поїздку або продовжити роботу з програмою.

На рисунку 2.6 зображено загальний алгоритм роботи програмного модуля визначення маршрутів для служб таксі.

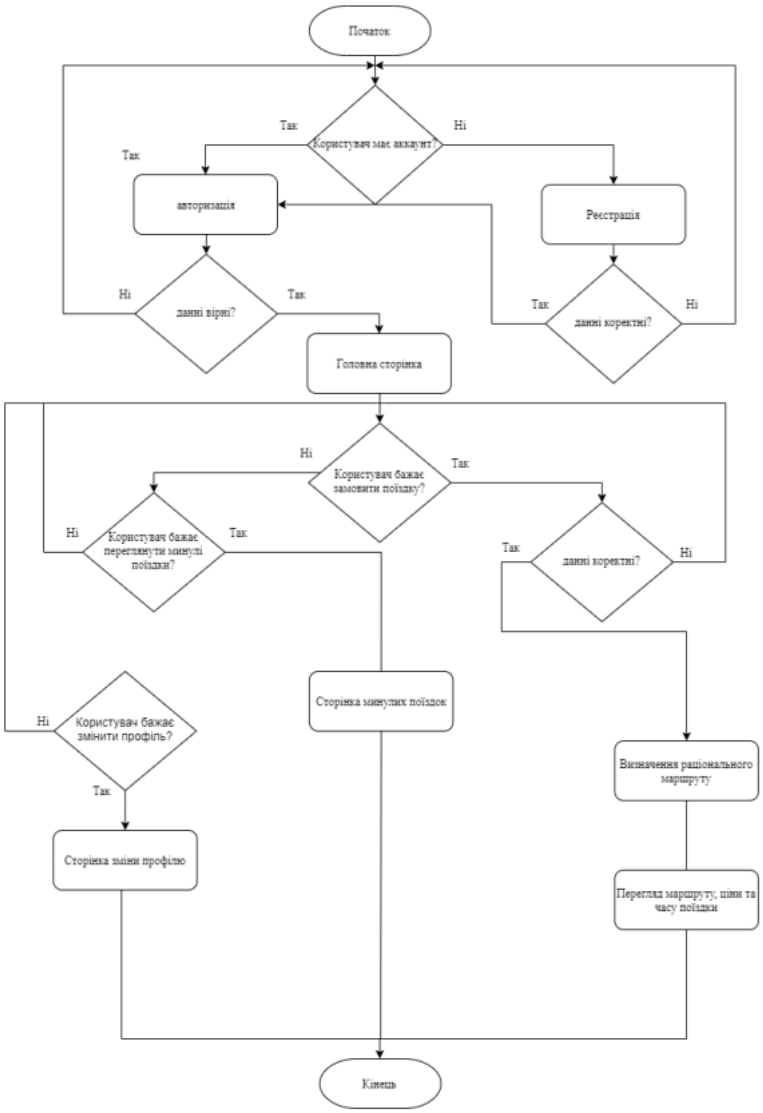


Рис. 2.6. Загальний алгоритм роботи програмного модуля визначення маршрутів для служб таксі

Детальну роботу варіантів взаємодії користувача з програмою можна побачити на діаграмі прецедентів на рисунку 7. На цій діаграмі показано повний список можливостей взаємодії користувача з програмою.

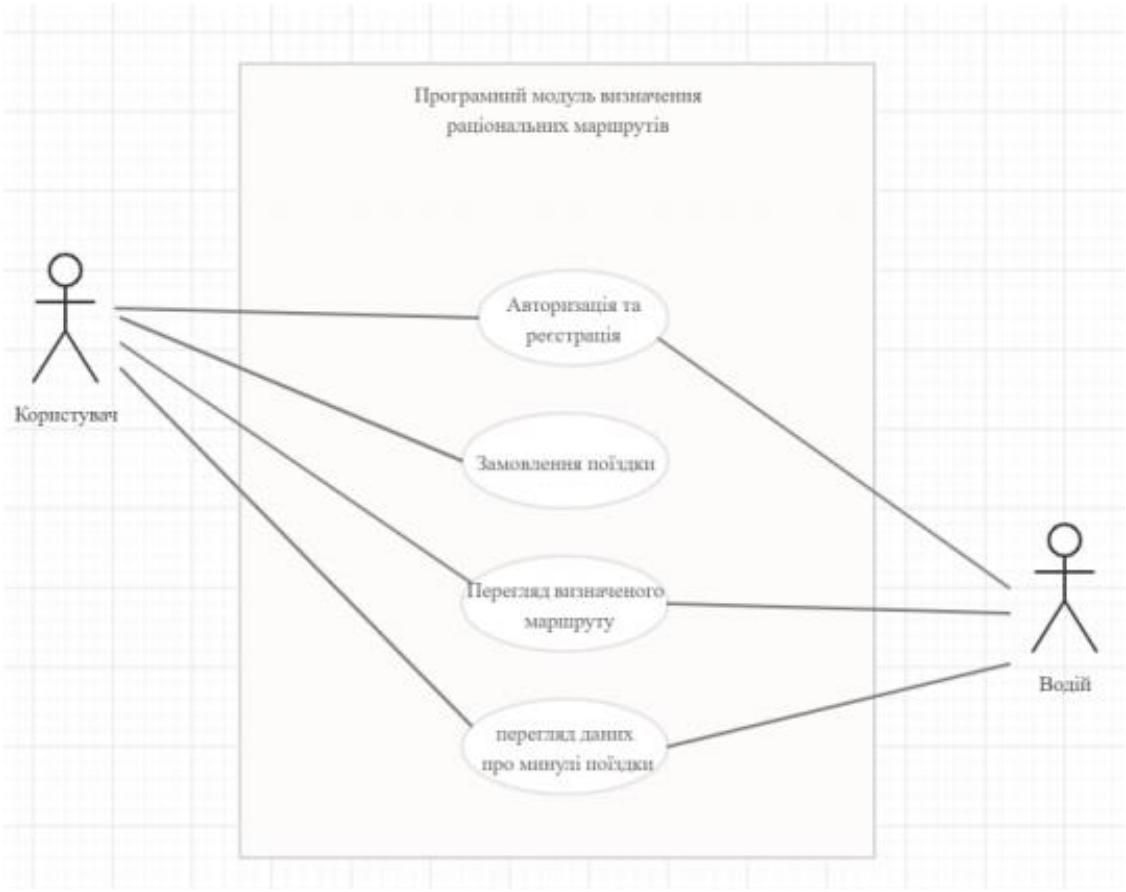


Рис. 2.7. – діаграма прецедентів

При Запуску програми відкривається сторінка авторизації. Якщо користувач вже зареєстрований в системі, він може ввести логін та пароль та продовжити роботу, якщо користувач ще не зареєстрований, він може перейти за посиланням на сторінку реєстрації. На сторінці реєстрації можна вибрати зареєструвати аккаунт для клієнта чи водія.

Після успішної реєстрації програма перенаправляє користувача на сторінку авторизації, де після успішної авторизації у користувача завантажується головна сторінка програми.

Для коректної роботи програми потрібно дозволити надання GPS-навігації, щоб можна було визначити точне місце знаходження користувача.

На головній сторінці програми можна зверху побачити меню навігації, за допомогою якого можна переходити на сторінки профіля, минулих поїздок.

У центрі головної сторінки зображено мапу, на якій видно де знаходиться користувач, та вільні авто таксі. У модулі водія на мапі зображено користувачів які зараз онлайн.

Над мапою знаходиться форма створення поїздки, де користувач може ввести місце посадки, місце призначення, тип авто, дату та час бажаної поїздки.

Висновки до розділу

Проаналізовано та обґрунтовано вибір алгоритму для знаходження найкоротшого шляху а також визначено шлях його модифікації для покращення ефективності визначення раціональних маршрутів.

Обґрунтовано математичну модель визначення раціональних маршрутів для служб таксі.

Розроблено клієнт – серверну архітектуру програмного модуля, що дало змогу спростити розробку програми та реалізувати механізм ефективного доступу великої кількості користувачів до інформації на сервері.

Розроблено загальний алгоритм функціонування модуля визначення раціональних маршрутів для служб таксі, що враховує ряд критеріїв, такі як: стан дорожнього покриття, тип дороги, наявність корок та їх потужність, погодні умови на певних ділянках дороги, обмеження швидкості, рівень безпечності дороги, статистика аварій на певних ділянках дороги.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ МОДУЛЯ ВИЗНАЧЕННЯ РАЦІОНАЛЬНИХ МАРШРУТІВ

3.1 Розробка UML – діаграми класів програмного модуля

Під час проектування програмного модуля визначення раціональних маршрутів для служб таксі розроблено діаграми класів кожного з модулів. Проект має модульну структуру. Основними модулями є:

– controllers;
– mappers;
– pathfinding;
– services.

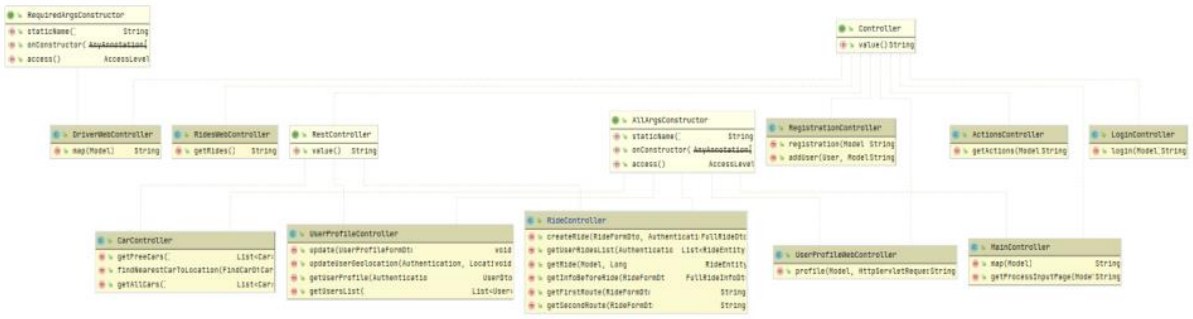


Рис. 3.1. UML діаграма класів модуля Controllers

У модулі Controllers містяться класи які відповідають за отримання запиту на сервер, обробки його та передавання на нижній рівень ієрархії. Основними класами є MainController, RidesControlelr, CarController, UserController. UML – діаграму класів модуля Controllers наведено на рисунку 3.1.

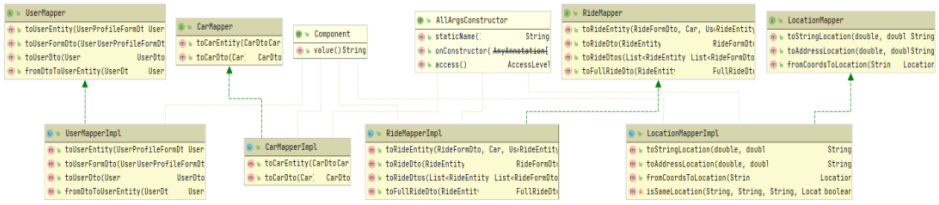


Рис. 3.2. UML-діаграма класів модуля Mappers

У пакеті mappers містяться класи які відповідають за конвертацію обєктів з dto форми до форми у якій приходять обєкти у запиті, у entity – обєкти з якими можна працювати у базі даних як з сущностями. Основними класами є RideController, UserController, CarController, RideService, UserService, CarService, RideMapper, MapsApiFacadeImpl, JsonReader. UML-діаграму класів модуля Mappers наведено на рисунку 3.2.

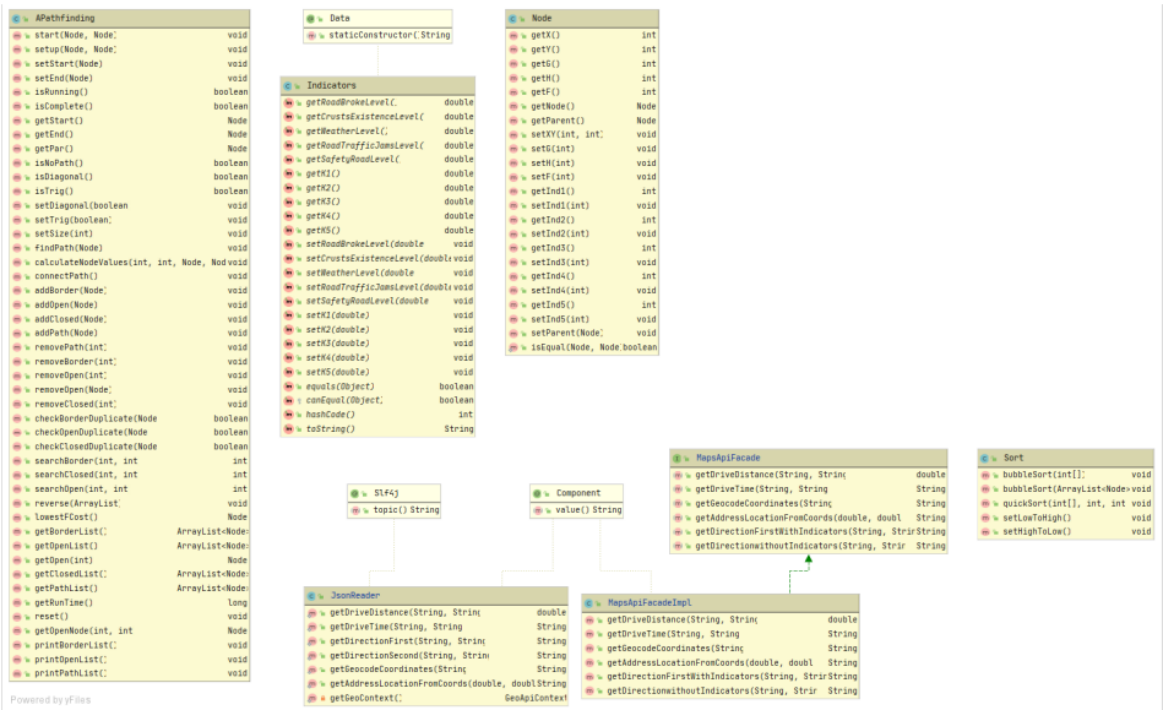


Рис. 3.3. UML-діаграма класів модуля Pathfinding

У пакеті pathfinding містяться класи такі як PathFinding, Node, Indicators, MapsApiFacade – класи які відповідають за роботу з алгоритмом визначення маршрутів та мапою. UML-діаграму класів модуля Pathfinding наведено на рисунку 3.3.

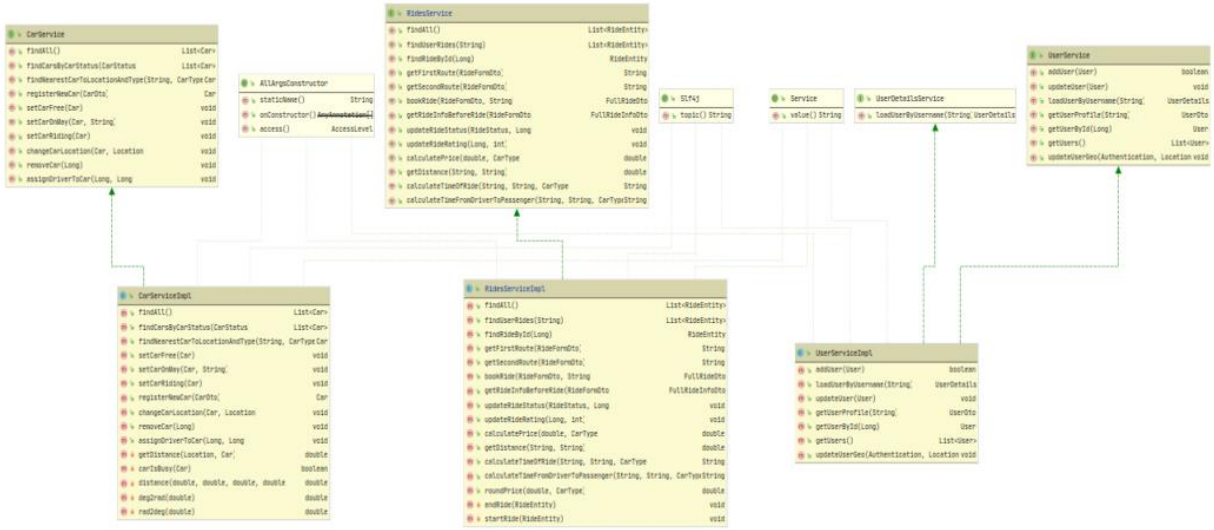


Рис. 3.4. UML-діаграма класів модуля Services

У модулі services містяться класи які виконують усю бізнес-логіку, таку як реєстрація та авторизація клієнта, пошук минулих поїздок клієнта, пошук найближчого авто тощо. UML-діаграму класів модуля Services наведено на рисунку 3.4.

3.2 Обґрунтування вибору мови та середовища програмування

Існує багато мов програмування за допомогою яких можна реалізувати програмний модуль визначення раціональних маршрутів для служб таксі. Але при виборі мови потрібно враховувати те, що дана мова програмування повинна бути об’єктно-орієнтована, та для неї має існувати зручне api google maps.

Серед таких мов програмування можна виділити Java, Python та JavaScript.

Java – об'єктно-орієнтована мова програмування, випущена компанією Sun Microsystems у 1995 році як основний компонент платформи Java [17]. Зараз мовою займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксис мови багато в чому походить від C та C++. У офіційній реалізації, Java програми компілюються у байткод, який при виконанні інтерпретується віртуальною машиною для конкретної платформи. Oracle надає компілятор Java та віртуальну машину Java, які задовольняють специфікації Java Community Process, під ліцезією GNU General Public License.

До переваг Java можна віднести:

- простота синтаксису;
- зручне Api для Google Maps;
- безпечність;
- не залежна від платформи;
- статична типізація;
- драйвери для роботи майже з усіма базами даними;
- збирач сміття;

До недоліків відносять наступне:

- велике використання пам’яті;
- не найкраща швидкість виконання;
- громіздкість синтаксису;
- для деяких версій потрібна платна ліцензія.

Python (рекомендоване прочитання — «Пайтон», запозичено назву збританського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтованамова програмування високого рівня з динамічною семантикою. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання наявних компонентів [11]. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python

та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектноорієнтована.

Переваги Python:

- зручний та простий синтаксис;
- об’єктно-орієнтованість;
- підтримка імперативного та функціонального програмування;
- підтримка веб та мобільних платформ;
- наявність великої кількості бібліотек
- до недоліків Python можна віднести наступне:
- повільна швидкість роботи порівняно з аналогами;
- має обмеження при роботі з базами даними;
- динамічна типізація – збільшення кількості помилок при виконанні.

JavaScript – обєктно-орієнтована скриптова мова програмування і є діалектом мови ECMAScript [6]. Ця мова дозволяє реалізувати складні функції на веб сторінках – кожен раз коли веб-сторінка робить більше, ніж просто відображати статичну інформацію, яку ми бачимо, відображення своєчасного оновлення вмісту, інтерактивні карти, анімована 2D/3D графіка, прокрутка відео тощо.

До переваг JavaScript можна віднести:

- швидкість роботи;
- простота розробки;
- популярність;
- мале навантаження на сервер;
- наявність багатих інтерфейсів;
- універсальність.

До недоліків даної мови можна віднести наступні характеристики:

- недостатня безпека на стороні клієнта;
- підтримка не всіх версій браузерів;

- нестача інструментів для налагодження.

Враховуючи переваги та недоліки розглянутих мов програмування було обрано Java, так як вона має зручне Google Maps Api, драйвери для усіх доступних баз даних, статичну типізацію, що збереже час при налагодженні програми та допоможе уникнути багатьох помилок при виконанні.

При розробці програми досить важливим є наявність зручного середовища розробки, яке б дозволило швидко розробляти, налагоджувати та тестувати програму.

Серед таких середовищ розробки є Intellij Idea.

Intellij Idea – інтегроване середовище програмування розроблено компанією JetBrains та написане на Java [1]. Дане середовише розробки створено для використання його з різними мовами програмування (Java, PHP, Python, Ruby, Scala). Система доступна під ліцензією Apache 2, та існують 2 версії: Community Edition та Ultimate Edition. Безкоштовна версія доступна для завантаження усім користувачам, безкоштовну ліцензію на платну версію продукту мають змогу отримати розробники проектів з відкритим кодом та студенти.

Серед функціональних можливостей які доступні користувачам є:

- допомога у коді, середовище програмування надає такі можливості як автодоповнення коду аналізуючи контекст, зручна навігація по проекту, яка дозволяє напряму перейти до потрібного класу або оголошення методу чи поля, інструменти для рефакторинга коду та налагодження програми.

- встановленні інструменти та інтеграція, середовище програмування має вбудовану систему контроля версій такі к Git, Mercurial, SVN. Підтримка при використанні баз даних Microsoft SQL Server, Oracle, PostgreSQL, MySQL, SQLite до яких можна отримати доступ одразу з середовища розробки.

- інструменти для тестування, аналізу, рефакторингу та налагодження.

- технології та фреймворки, у обох версіях середовища підтримуються такі технології як Junit, Maven, Gradle, TestNG.

- враховуючи усі функціональні можливості, було обрано середовище програмування Intellij Idea, оскільки воно має велику кількість інструментів для налагодження, тестування та рефакторингу коду, а також вбудований зручний інтерфейс та можливість роботи з різними базами даних з коробки.

3.3 Програмна реалізація модуля визначення раціональних маршрутів для служб таксі

При написанні програми були використані Java, Spring, Google Maps Services. Оскільки така зв'язка мови та бібліотек забезпечує максимальну ефективність розробки, налагодження, рефакторингу, тестування та підтримки програмного модуля.

При створенні поїздки, на сервер надходить запит з клієнта з такими даними як місце посадки, місце призначення, тип авто, дата та час бажаної поїздки.

```
package com.taximaps.server.entity.dto.ride;

import ...

@Data
public class RideFormDto {
    @NotNull(message = "Origin must not be null")
    private String origin;
    @NotNull(message = "Destination must not be null")
    private String destination;
    @NotNull(message = "You must choose car type")
    private String carType;
    @NotNull(message = "Date must not be null")
    private Date date;
}
```

Рис. 3.5. Клас із вхідними даними про поїздку які надходять на сервер

Ці дані надходять як запит на сервер, та їх приймає клас RideController, який має наступні методи.

CreateRide – метод який на вхід клас RideFormDto робить необхідні перетворення, та створює поїздку.

GetUserRideList – метод який приймає на вхід клас Authentication, знаходить за допомогою цього класу унікальний ідентифікатор користувача, та отримує інформацію про нього. Маючи необхідну інформацію, цей метод повертає дані про всі поїздки даного користувача з бази даних.

GetRide – метод який виконується як метод getUserRideList, тільки цей метод повертає інформацію про конкретну поїздку, дату та час, ціну, водія, статус поїздки, місце посадки та призначення.

GetInfoBeforeRide – метод який виконує визначення маршруту та розраховує ціну та час поїздки для попереднього ознайомлення, та вибору оформити поїздку чи ні.

Крім цього, важливими компонентами програмного модуля є класи MapsApiFacadeImpl, який виконує всі запити на Google Maps Services, за допомогою яких всі машини, користувачі, водії та маршрути можна відобразити на мапі.

В основному клас RideController відповідає за такі дії з поїздками як створення поїздок, отримання повного списку поїздок для водія та користувача, отримання інформації про конкретну поїздку, та отримання інформації про приблизно розраховану ціну та час майбутньої поїздки. А саме метод createRide, приймає на вхід клас RideFormDto та Authentication клас – клас з якого можна отримати інформацію про користувача який замовив поїздку.

Наступним кроком контроллер клас передає дані у сервіс RidesServiceImpl, який містить у собі код бізнес-логіки. В першу чергу з класа Authentication програма отримує логін користувача, та через запит до бази даних знаходить повну інформацію про клієнта по його логіну.

Тоді програма виконує пошук найближчого авто до користувача. Здійснюється це викликом методу findNearestCarToLocation у класа CarService який приймає на вхід місце посадки та тип авто яке бажає користувач. Після цього алгоритм знаходить найближче авто, присвоює поїздку водієві, та розраховує ціну та час поїздки.

Якщо запит повертає статус success, то користувач на сторінці побачить ціну поїздки та приблизно розрахований час який вона займе, інакше, якщо на якомусь з етапів станеться помилка або не буде знайдено водія, або поїздка буде відмінена користувач залишиться на головній сторінці та зможе продовжити роботу з програмою.

Тим часом у модулі водія, За допомогою класів PathFinding та MapsApiFacade, відбувається запит на Google Maps Services, отримується інформація про зону мапи та дороги у якій буде здійснюватись поїздка, та згідно отриманих з показників даних про погоду, затори, стан доріг тощо алгоритм знаходить раціональний маршрут, та коли поїздка назначається водію, у водія на мапі відображається цей маршрут.

Як тільки почалась поїздка, тобто з моменту виїзду водія до місця посадки користувача, попередні дані про поїздку які можна побачити на рисунку 3.6 зберігаються у базі даних.

Рис. 3.6. Дані про поїздку які зберігаються у базі даних

3.4 Тестування програми

Основною задачею тестування програмного модуля визначення раціональних маршрутів є перевірка наявних функціональних можливостей та відповідність вимогам. Реєстрація користувача, авторизація користувача, процес замовлення поїздки, процес пошуку раціонального маршруту та його відображення, процес запису інформації до бази даних.

Для перевірки функціональності реєстрації користувача, запустимо програму та відкриємо головну сторінку за адресою https://localhost:8443. Одразу можна перевірити безпеку користування програмою, так як для користувача який не увійшов у свій обліковий запис, замість головної сторінки буде відображатись сторінка авторизації яка зображена на рисунку 3.7.

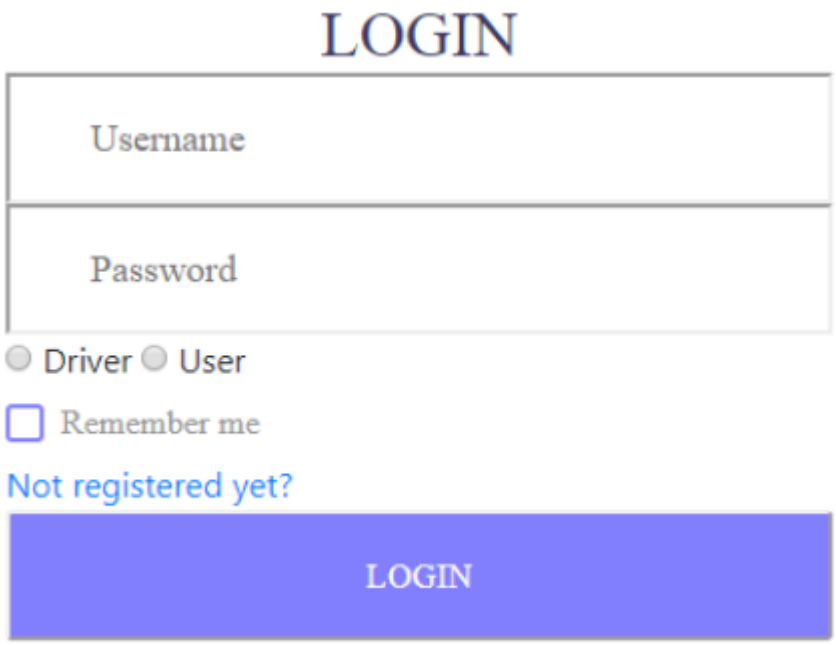


Рис. 3.7. Сторінка авторизації

Спочатку, щоб перевірити процес реєстрації потрібно натиснути на посилання Not registered yet, після цього завантажується сторінка реєстрації користувача яка зображена на рисунку 3.8.

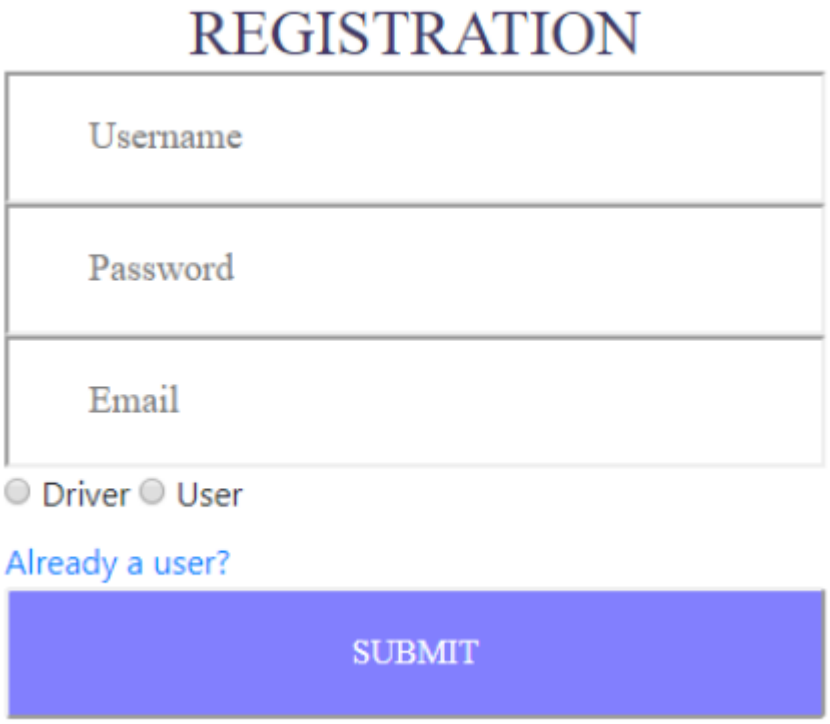


Рис. 3.8. Сторінка реєстрації

Після введення логіну, пароля, пошти та натискання на кнопку Submit, у базі даних повинен створитися запис про користувача. На рисунку 3.9 можна побачити створеного користувача.

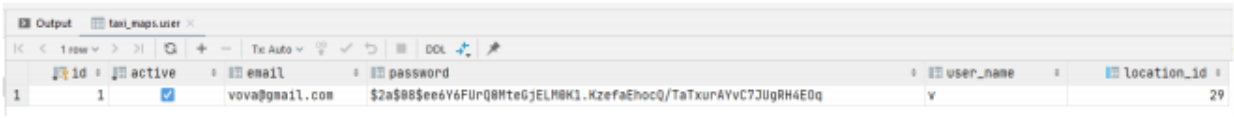


Рис. 3.9. Створений запис про користувача у базі даних

Наступною перевіркою буде перевірка функціональності авторизації. Для цього потрібно перейти на сторінку авторизації та ввести дані введені при реєстрації. При введені коректних даних програма перевіряє чи існує користувач з такими даними у базі даних та в успішному випадку співпадіння інформації перенаправить на головну сторінку програми яку можна побачити на рисунку 3.10.

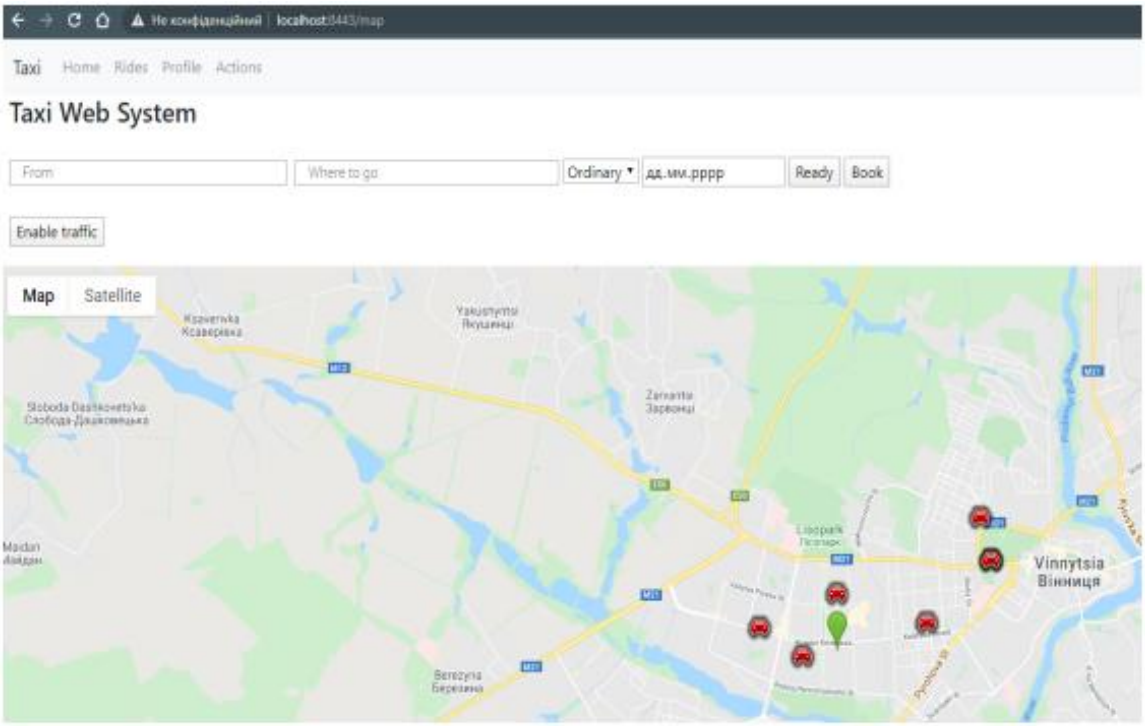


Рис. 3.10. Головна сторінка програми

Для перевірки функціональності визначення раціонального маршруту, на головній сторінці програми користувачу потрібно ввести місце посадки, пункт призначення, тип авто який бажає вибрати користувач, а також дату та час поїздки. Після введення всіх необхідних даних натиснувши кнопку Ready програма перевірить наявність доступних машин, підбере найближчу машину та зобразить маршрут на мапі, який користувач та водій зможе побачити та поруч буде представлено ціну та час поїздки для ознайомлення.

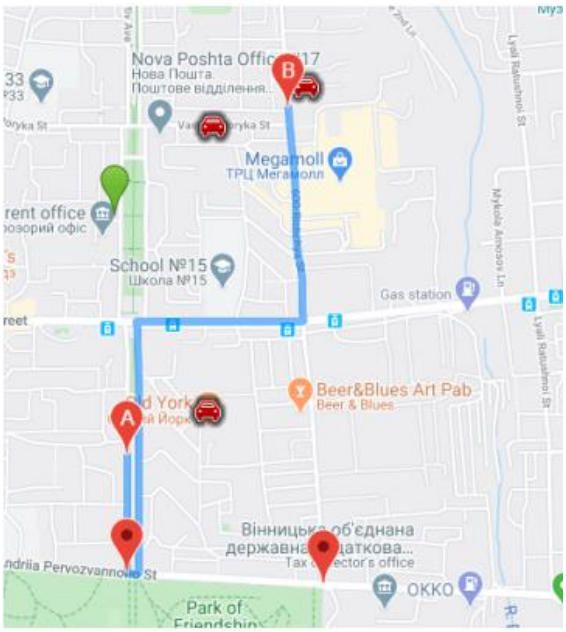


Рис. 3.11. Приклад визначення раціонального маршруту без урахування показників

Як показано на рисунку 3.11, програма скористалась алгоритмом визначення раціонального маршруту без урахування додаткових показників, і в результаті програма визначила маршрут від точки А до точки В який є найближчим, але не найоптимальнішим, оскільки при визначенні цього маршруту програма не брала до уваги такі показники як стан дороги.

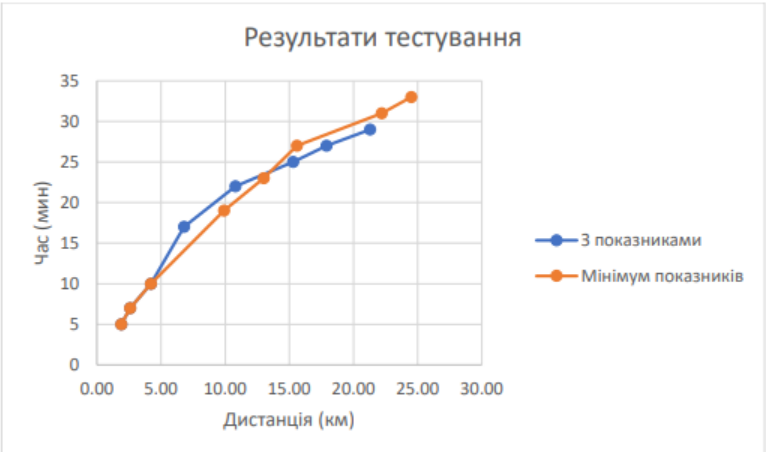
Також визначений маршрут не враховує кількості аварій які сталися на даній ділянці дороги, та наскільки небезпечно може бути тут проїзд. Та обраний маршрут не враховує затори, оскільки по шляху цього маршруту проходить головна дорога міста, залежно від часу проїзду, затримка може бути значною.

Натиснувши кнопку Book відбудеться ті самі процедури що і при натисканні на кнопку Ready, тільки вже створиться запис про поїздку в базі даних, програма визначить найближчого водія та призначить йому цю поїздку, після всіх цих дій буде визначений маршрут із урахуванням додаткових показників попередньо завантажених у систему.

Рис. 3.12. Приклад визначення раціонального маршруту з урахуванням показників

Як можна побачити на рисунку 3.12, визначений маршрут із урахуванням показників є раціональнішим, оскільки у показниках завантажених у систему було зазначено тестові дані, на вулиці Келецька є затори, що створюють затримку поїздки, та також було зазначено що на цій вулиці низький рівень стану дороги, або сталось ДТП та потрібно об’їжджати.

Алгоритм визначення маршруту врахував усі показники, та побудував маршрут від точки А до точки В, який є більш раціональнішим у даній ситуації. Результати тестування можна побачити на графіку 3.1.



Графік 3.1. Результати тестування

Якщо брати до уваги довжину обох маршрутів, та час який потрібен, щоб достатись з точки А до точки В, то можна спостерігати збільшення ефективності визначення маршруту та скорочення часу поїздки на 5 – 12%, в окремих випадках може сягати 22%, але у них не враховуються типи авто.

Висновки до розділу

У даному розділі було зроблено UML – діаграму класів кожного з модулів, наведено структуру та основні класи, що використовуються у програмному модулі.

У результаті порівняльного аналізу об'єктно – орієнтованих мов на основі ряду об'єктивних переваг було обрано мову програмування Java.

Виконано аналіз найпопулярніших середовищ програмування та обґрунтовано вибір середовища програмування Intellij Idea, яке задовольняє усім потребам під час розробки програмного забезпечення, має зручний графічний інтерфейс, а також засоби налагодження.

На основі запропонованих математичної моделі та алгоритму реалізовано програмний модуль визначення раціональних маршрутів для служб таксі.

Протестовано основні функціональні можливості програмного модуля. Під час порівняння визначення маршрутів з меншою та більшою кількостями показників, другий варіант показав збільшення ефективності на 5 – 12%.

ВИСНОВКИ

Під час виконання бакалаврської дипломної роботи обґрунтовано актуальність задачі визначення раціональних маршрутів для служб таксі. Здійснено аналітичний огляд систем визначення раціональних маршрутів для служб таксі. Виконано постановку задачі розробки програмного модуля з рядом вимог, що мають врахувати основні недоліки розглянутих аналогів. Проаналізовано та обґрунтовано вибір алгоритму для знаходження найкоротшого шляху, а також визначено шлях його модифікації для покращення ефективності визначення раціональних маршрутів.

Обґрунтовано математичну модель визначення раціональних маршрутів для служб таксі. Розроблено клієнт – серверну архітектуру програмного модуля, що дало змогу спростити розробку програми та реалізувати механізм ефективного доступу великої кількості користувачів до інформації на сервері. Розроблено загальний алгоритм функціонування модуля визначення раціональних маршрутів для служб таксі, що враховує ряд критеріїв, такі як стан дорожнього покриття, наявність корок та їх потужність, погодні умови, рівень завантаженості дороги, статистика аварій тощо.

Наведено UML – діаграму класів кожного з модулів, наведено структуру та основні класи, що використовуються у програмному модулі. У результаті порівняльного аналізу об'єктно – орієнтованих мов, на основі ряду об'єктивних переваг було обрано мову програмування Java. На основі запропонованих математичної моделі та алгоритму реалізовано програмний модуль визначення раціональних маршрутів для служб таксі. Здійснено тестування основних функціональних можливостей програмного модуля. Під час порівняння визначення маршрутів з меншою та більшою кількостями показників, другий варіант показав збільшення ефективності на 5 – 12%.

В результаті виконання дипломного проекту поставлені задачі були виконані в повній мірі.