Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Оптимізація логістичних процесів засобами інтелектуальної маршрутизації
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-4
Артем Пітонов
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник доц. Ігор Магдаліна
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Пітонову Артему Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Оптимізація логістичних процесів засобами інтелектуальної маршрутизації

затверджена наказом університету від 19 травня 20 25 р. № 378Ст
2. Термін подання студентом роботи до екзаменаційної комісії 20 червня 20 25 р.
3. Вихідні дані до роботи набір даних, документація Java, документація Python, документація Google Maps, аналітичні дані ринку доставки їжі в Україні, інтернет-джерела та публікації з досліджуваної проблеми

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Теоретичні дослідження
3) Опис програмної реалізації

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі та постановка задачі	20.05.2025	виконано
3	Теоретичні дослідження	27.05.2025	виконано
4	Опис програмної реалізації	10.06.2025	виконано
5	Подання документів до комісії	19.06.2025	виконано
6	Захист	20.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Ігор Магдаліна
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 91 с., 39 рис., 1 дод., 25 джерел.

ДОСТАВКА «ОСТАННЬОЇ МИЛІ», ЗАДАЧА МАРШРУТИЗАЦІЇ ТРАНСПОРТУ, ІНТЕЛЕКТУАЛЬНА МАРШРУТИЗАЦІЯ, МАШИННЕ НАВЧАННЯ, ОПТИМІЗАЦІЯ ЛОГІСТИКИ, ПРОГНОЗУВАННЯ ЧАСУ ДОСТАВКИ, GOOGLE MAPS API, JAVA SPRING BOOT, REACT, XGBOOST.

Об’єкт дослідження – інформаційна система для оптимізації логістичних процесів у сфері доставки їжі.

Предмет дослідження – методи та технології розробки серверної та клієнтської частини системи, а також моделі машинного навчання для прогнозування часу доставки.

Мета роботи – розробити програмний комплекс для оптимізації та автоматизації процесів доставки їжі з інтеграцією інтелектуальної маршрутизації, що забезпечує точне прогнозування часу доставки та зручне управління для менеджерів та кур’єрів.

Методи дослідження – аналіз літератури, системний аналіз, методи програмної інженерії, методи машинного навчання, експериментальне тестування.

У результаті виконання роботи було досліджено актуальність теми побудови системи доставки із використанням методів інтелектуальної маршрутизації, особливо в умовах воєнного часу в Україні, коли питання логістики набувають особливої ваги. Було проведено аналіз предметної області та сучасних підходів до оптимізації маршрутів та прогнозування часу доставки. Проаналізовано сучасні алгоритми машинного навчання які демонструють високу точність у задачах пов’язаних з логістичними процесами.

ABSTRACT

Bachelor’s thesis contains: 91 pp., 39 fig., 1 ann., 25 references.

DELIVERY TIME FORECASTING, GOOGLE MAPS API, INTELLIGENT ROUTING, JAVA SPRING BOOT, «LAST MILE» DELIVERY, LOGISTICS OPTIMIZATION, MACHINE LEARNING, REACT, VEHICLE ROUTING PROBLEM, XGBOOST.

Object of the study – an information system for optimizing logistics processes in the field of food delivery.

Subject of the study – methods and technologies for developing the server and client parts of the system, as well as machine learning models for delivery time prediction.

Purpose of the work – to develop a software system for optimizing and automating food delivery processes with the integration of intelligent routing, ensuring accurate delivery time prediction and convenient management for both managers and couriers.

Research methods – literature review, systems analysis, software engineering methods, machine learning methods, and experimental testing.

As a result of the work, the relevance of building a delivery system using intelligent routing methods was explored, especially in the context of wartime in Ukraine, where logistics issues are of particular importance. The subject area and modern approaches to route optimization and delivery time prediction were analyzed. Current machine learning algorithms that demonstrate high accuracy in logistics-related tasks were also reviewed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

ETA – Estimated Time of Arrival – очікуваний час доставки;

JWT – JSON Web Token – стандарт для створення токенів доступу, що використовуються для безпечної передачі інформації та автентифікації користувачів у веб-додатках;

KPI – Key Performance Indicators – ключові показники ефективності;

LightGBM – Light Gradient Boosting Machine – популярна бібліотека машинного навчання, що реалізує алгоритм градієнтного бустингу;

ML – Machine Learning – машинне навчання;

OSM – OpenStreetMap – безкоштовна, відкрита та створена спільнотою користувачів географічна карта світу, що є альтернативою комерційним сервісам, як-от Google Maps;

SaaS – Software as a Service – модель розповсюдження програмного забезпечення, за якою доступ до програми надається через інтернет за підпискою, без необхідності встановлювати її на свій комп'ютер;

SVR – Support Vector Regression – метод опорних векторів;

TMS – Transportation Management System – система управління транспортом;

VRP – Vehicle Routing Problems – задачі маршрутизації транспорту;

XGBoost – Extreme Gradient Boosting – бібліотека машинного навчання, що використовує алгоритм градієнтного бустингу для створення високоточних прогнозних моделей.

ВСТУП

У сучасному світі логістична галузь стрімко зростає і постійно адаптується до нових викликів [1], [2]. Останніми роками зростання електронної комерції, зокрема під впливом пандемії COVID-19, змінило правила гри для логістичних компаній. Люди все частіше замовляють товари онлайн, очікуючи швидкої, гнучкої, точної та дешевої доставки [3]. Водночас глобальні ланцюги постачання стають все складнішими, а в Україні додатково ситуація значно ускладнюється війною, яка впливає на економіку, інфраструктуру та безпеку [4]. Усе це змушує логістичні компанії розвиватися та шукати нові рішення, щоб залишатися конкурентоспроможними на ринку.

Традиційні методи планування маршрутів, які часто спираються на статистичні дані або ручні розрахунки вже не відповідають вимогам ефективності в умовах динамічного міського середовища [5]. Такі фактори як обмеження руху, погана інфраструктура, затори на дорогах чи несподівані затримки ускладнюють планування. Що, у свою чергу, призводить до затримань у доставці, зниження якості обслуговування, зменшення довіри клієнтів та збільшення витрат на паливо.

На допомогу у вирішенні даних проблем приходять сучасні технології, а саме інтелектуальна маршрутизація, яка використовує методи штучного інтелекту (ШІ) і машинного навчання (ML). Такі системи є дуже потужними і здатні аналізувати величезні обсяги інформації у режимі реального часу, з високою точністю прогнозувати очікуваний час доставки (ETA) та будувати оптимальні маршрути доставки враховуючи велику кількість змінних, починаючи від стану та завантаженості доріг до пріоритетності замовлень і типу вантажного транспорту. Завдяки таким системам логістичні компанії мають змогу економити ресурси, підвищувати якість сервісу та отримувати більший прибуток. Наприклад, за даними досліджень, оптимізація логістики може скоротити витрати аж до 25% [6].

Саме виходячи з описаних викликів розробка інтелектуальної системи маршрутизації є надзвичайно актуальною. Така система буде особливо корисною для малого та середнього бізнесу в Україні. Крім того, дана система значно спрощує між менеджерами та кур’єрами, що підвищує ефективність роботи. Таким чином, розробка такого рішення є дуже перспективним, адже сприяє розвитку бізнесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі

Логістика «останньої милі» є фінальним етапом доставки товарів від локального розподільчого центру, магазину, ресторану чи складу до кінцевого споживача або пункту видачі. Хоча цей етап є найкоротшим, але він є й найскладнішим і найдорожчим [7]. Повний ланцюг постачання можна бачити на рисунку 1.1. Кур'єр який везе замовлення має уникати затори, їхати через переповнені міські вулиці, знайти місце для паркування та обирати найбільш оптимальний шлях. Кожен з цих кроків може спричинити затримку, що, в свою чергу, впливає на задоволеність клієнтів і витрати та прибуток компанії.

Саме цей етап є ключовим, адже він формує враження клієнтів про сервіс. Наприклад, коли клієнти замовляють їжу, то вони очікують, що їжа прибуде вчасно і гарячою. Якщо ж кур'єр запізниться через затори або неправильно спланований маршрут, то це може зіпсувати досвід клієнта. Для компанії це буде значити втрату клієнтів, прибутку та додаткові трати на паливо.

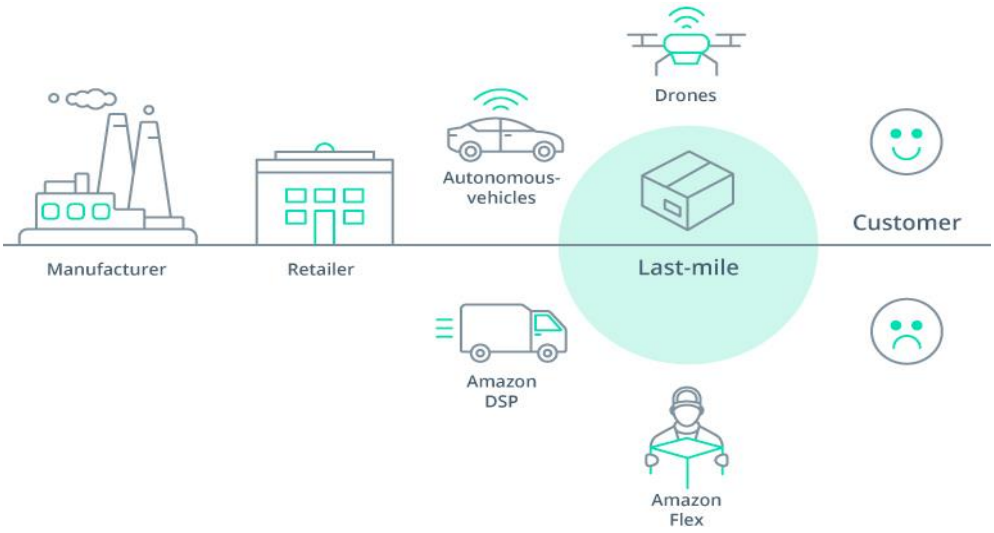

Рисунок 1.1 – Повний ланцюг постачання

Сам термін «остання миля» походить із телекомунікаційної галузі, де він позначав підключення окремих будинків до основної мережі [8]. У логістиці ж його почали використовувати для опису доставки до кінцевого споживача. Хоча доставка до клієнтів існує вже дуже давно, справжньою проблемою вона стала з появою електронної комерції наприкінці 1990-х років. Переломним моментом став запуск Amazon у 1994 році, адже компанія запропонувала швидку доставку, що змусило всіх конкурентів також вдосконалювати свої логістичні процеси. У 2000-х роках поява GPS-трекінгу дозволила відстежувати кур’єрів у реальному часі, що підвищило довіру до доставки [9]. З 2010-х років компанії почали застосовувати штучний інтелект і машинне навчання для оптимізації маршрутів і прогнозування часу доставки. Алгоритми можуть проаналізувати історичні дані про трафік і запропонувати найшвидший шлях, уникаючи заторів, що робить їх незамінними. На сьогоднішній день проводять тестування навіть автономних транспортних засобів та дронів, що у майбутньому може кардинально змінити логістику [10]. Хронологію розвитку логістики можна побачити на рисунку 1.2.

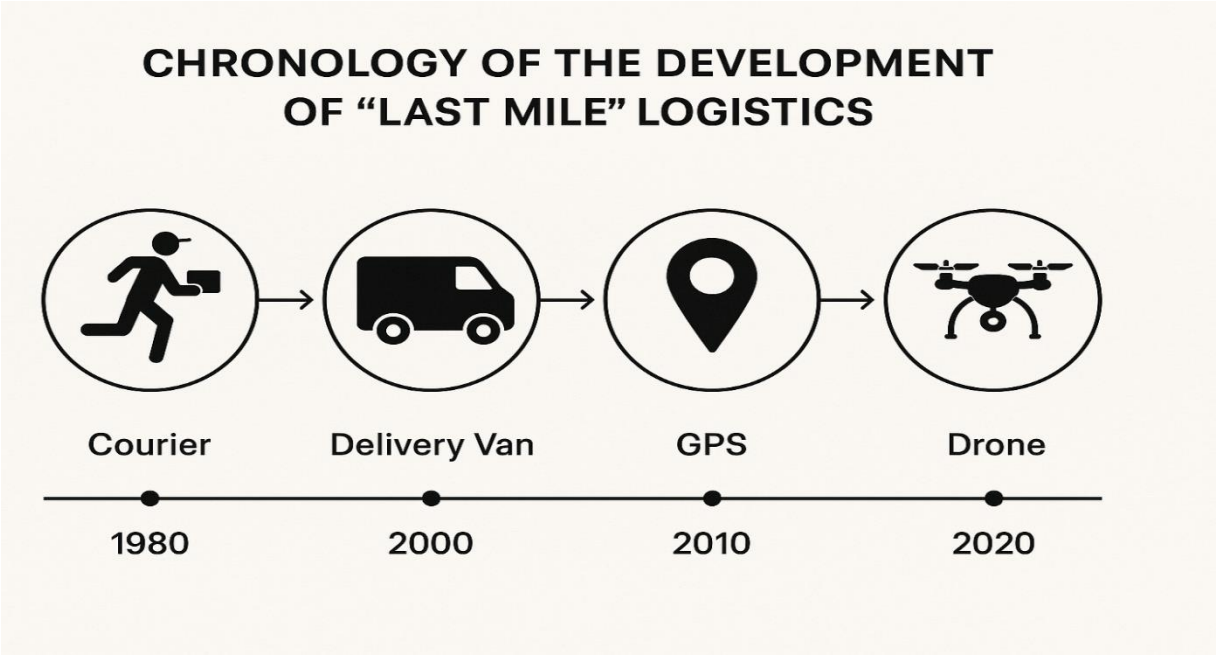

Рисунок 1.2 – Хронологія розвитку логістики «останньої милі»

Сучасна логістика «останньої милі» стала невід'ємною частиною глобальної економіки через зміну споживацьких звичок. Люди звикли замовляти товари онлайн, від одягу до продуктів, і очікують, що доставка буде швидкою, зручною та прозорою. За даними Forrester, у 2023 році 20% глобальних роздрібних продажів припадало на онлайн-канали, ця частка продовжує зростати [11]. Опитування Sendcloud показало, що більшість покупців вважають точний час прибуття, відомий як ETA вирішальним фактором при виборі магазину, а 52% готові відмовитися від компанії, якщо доставка їх не влаштувала [12]. Приклад інтерфейсу для відстеження доставки можна побачити на рисунку 1.3.

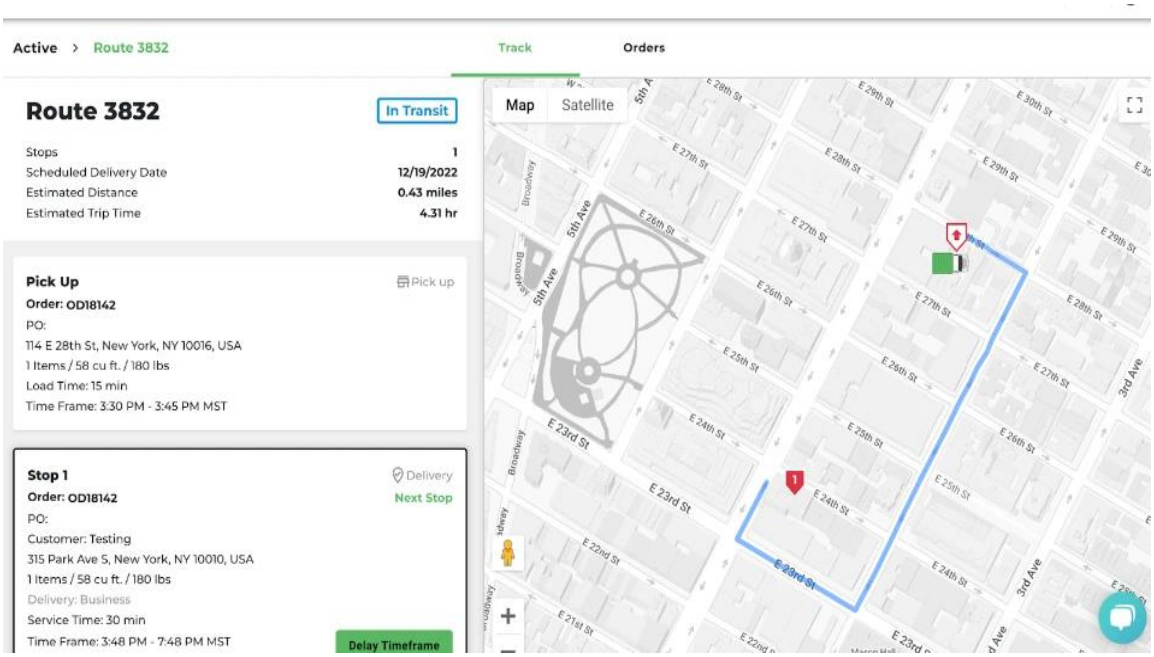

Рисунок 1.3 – Приклад інтерфейсу відстеження доставки

Логістика «останньої милі» постійно функціонує у динамічному середовищі, через що виникають непередбачувані фактори. У містах доставку ускладнює обмежена кількість паркувальних місць, щільний трафік та затори. Наприклад, у центрі Києва кур'єру може знадобитися 10 або 15 хвилин лише для пошуку місця для паркування. Дорожні аварії чи роботи можуть значно затримувати кур'єрів, що збільшує час доставки.

Карту міського трафіку з позначеними зонами заторів можна побачити на рисунку 1.4.

Індивідуальна доставка до кожного клієнта є значно дорожчою порівняно з масовою поставкою, адже витрати на пальне і час кур’єрів становлять значну частину бюджету. Зростання кількості доставок також сприяє збільшенню викидів вуглецю, що погано впливає на екосистему. Це створює тиск на компанії по впровадженню рішень цієї проблеми, таких як електромобілі, велосипеди чи оптимізовані маршрути, які можуть скоротити споживання палива.

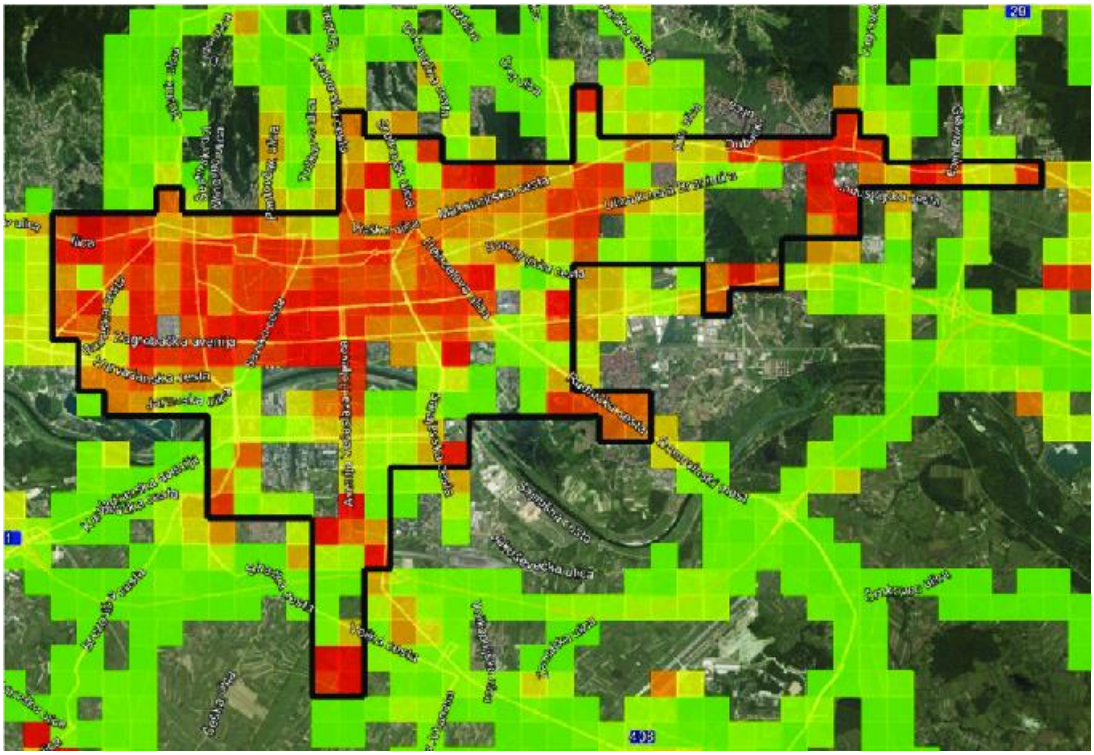

Рисунок 1.4 – Карта міського трафіку з позначеними зонами заторів

В Україні ці глобальні виклики доповнені місцевими особливостями. Якість дорожньої інфраструктури нерівномірна: у великих містах, як Київ чи Львів, дороги краще ніж в менших населених пунктах. Економічна нестабільність і зростання цін на пальне змушують бізнеси, особливо середні і малі, максимально оптимізувати свої витрати [13].

Повномасштабна війна додала ще більше проблем. Тепер кур’єри змушені об’їжджати небезпечні зони, адаптуватися до повітряних тривог та справлятися з перебоями в інфраструктурі. Наприклад, у містах поблизу фронту доставка може затримуватися через повітряну тривогу. Ці умови роблять питання оптимізації критично важливим для компаній.

Для подолання цих викликів компанії впроваджують нові рішення. Алгоритми штучного інтелекту та машинного навчання здатні прогнозувати маршрути, передбачати час доставки, аналізувати історичні дані про трафік, погоду чи поведінку клієнтів. Наприклад, система здатна визначити найшвидший шлях, уникаючи заторів. Використання електромобілів, пішої доставки або велосипедів зменшує викиди вуглецю, що відповідає вимогам екологічної відповідальності. Хоча дрони та автономні транспортні засоби ще знаходяться у тестуванні, у майбутньому вони обіцяють значно зменшити витрати. Ці рішення демонструють наскільки «логістика останньої милі» вимагає комплексного підходу який буде поєднувати в собі різні фактори.

Практичні приклади показують, як компанії адаптуються до цих викликів. Amazon створив власну логістичну мережу з розподільчими центрами та флотом транспортних засобів, що дозволяє доставляти товари протягом одного дня. В Україні, в умовах війни, Нова Пошта демонструє унікальну стійкість та адаптивність, активно використовуючи мобільні відділення на базі вантажівок, щоб забезпечувати доставку навіть у найскладніших прифронтових та деокупованих регіонах, де стаціонарна інфраструктура пошкоджена або відсутня. Ці мобільні точки виконують не лише логістичну, а й важливу соціальну функцію, стаючи символом повернення до життя та присутності держави. Це рішення, разом із розгалуженою мережею поштоматів, яка пропонує безпечний безконтактний сервіс в умовах повітряних тривог, підкреслює здатність компанії швидко реагувати на екстремальні виклики (рисунок 1.5).

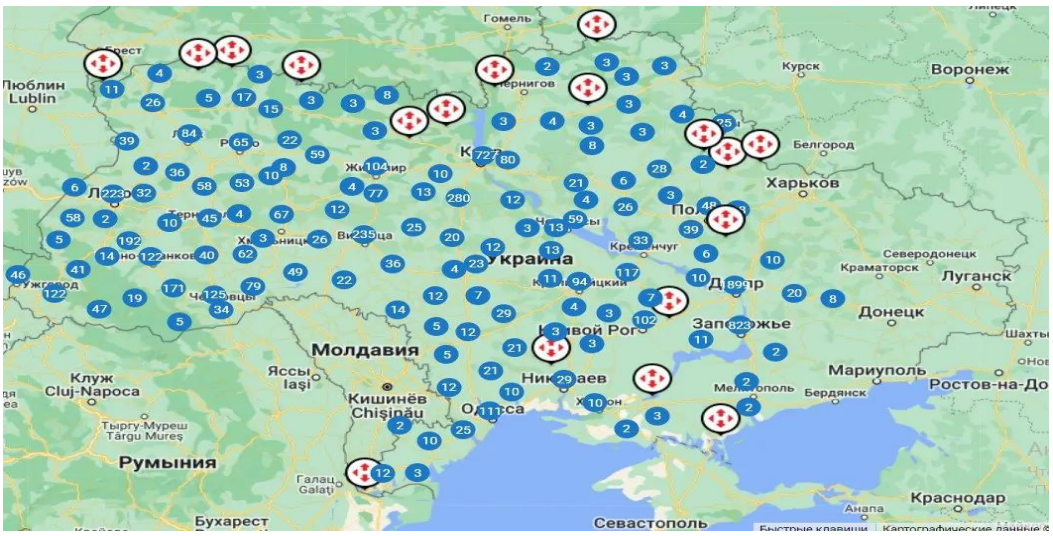

Рисунок 1.5 – Схема відділень Нової Пошти

Логістика «останньої милі» відіграє ключову роль у сучасній економіці. Оптимізація цього етапу дозволяє скоротити витрати, збільшити прибуток, підвищити швидкість доставки та зменшити екологічний вплив. В Україні, де війна та економічні обмеження створюють додаткові труднощі, розробка гнучких та доступних рішень є надзвичайно актуальною. Впровадження технологій, таких як штучний інтелект і картографічні сервіси може допомогти в подоланні цих викликів.

1.2 Поняття інтелектуальної маршрутизації та її роль в оптимізації

Логістика «останньої милі», як було зазначено вище, стикається з численними викликами. Задля подолання цих проблем сучасні компанії все частіше звертаються до інтелектуальної маршрутизації. Інтелектуальна маршрутизація дозволяє показувати не лише коротший шлях, а й враховувати безліч інших факторів, такі як затори, тип транспорту, погодні умови та інші.

На відміну від традиційних методів, які покладаються лише на статистичні дані або досвід диспетчерів, інтелектуальні системи здатні обробляти десятки змінних одночасно, роблячи це швидко і точно. Вони

здатні аналізувати історичні дані, знаходити неочевидні закономірності та адаптуватися до мінливих умов нашого світу. Наприклад, якщо кур’єр буде доставляти їжу в центрі великого міста, то система може передбачити, що в обідню пору певна вулиця може бути більш перевантаженою і знайти інший оптимальний шлях.

Одним із найголовніших завдань інтелектуальної маршрутизації є точне прогнозування часу доставки, відомого як ETA. Для цього система повинна аналізувати безліч параметрів: відстань, погодні умови, час доби, день тижня, рейтинг кур’єру та історичні дані щодо того скільки зазвичай займає підготовка замовлення в конкретному закладі.

Не менш важливою функцією сучасних логістичних систем є динамічна оптимізація маршрутів. Хоча звичайні картографічні сервіси, такі як Google Maps, чудово пропонують найшвидші маршрути для однієї поїздки, вони не враховують усієї складності комерційних перевезень. Інтелектуальна маршрутизація йде значно далі, перетворюючи логістику на високоточну науку.

Завдяки їй система може враховувати численні специфічні потреби бізнесу в режимі реального часу. Наприклад, вона може уникати платних доріг задля економії коштів, враховувати часові вікна доставки, бажані для клієнтів, об'єм та вагу вантажу, а також місткість транспортного засобу.

У складних сценаріях, коли один кур’єр доставляє декілька замовлень система використовує потужні оптимізаційні алгоритми, як-от алгоритм найближчого сусіда чи генетичні алгоритми. Вони прораховують тисячі й мільйони комбінацій, щоб визначити найкращу послідовність відвідування точок. Це дозволяє не тільки суттєво зменшити загальний час у дорозі і витрати на паливо, але й підвищити продуктивність кур'єрів та рівень задоволеності клієнтів, які отримують свої замовлення швидше. Порівняння традиційного та інтелектуального маршрутів наведено на рисунку 1.6.

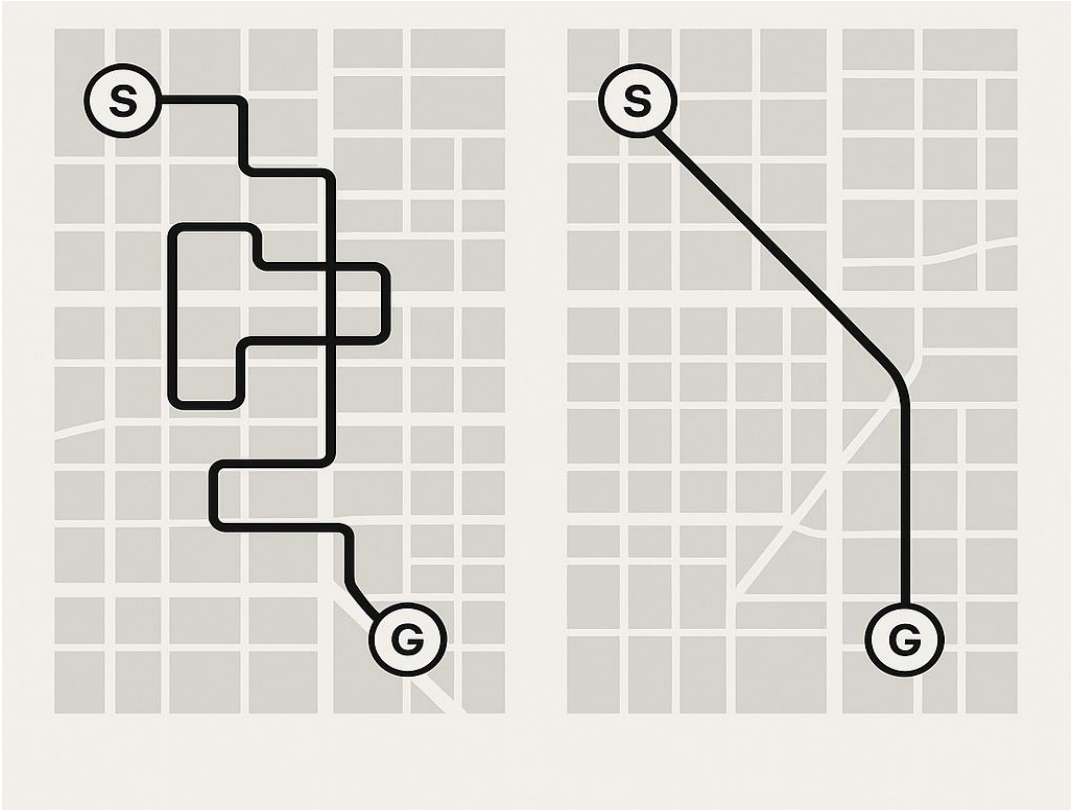

Рисунок 1.6 – Порівняння традиційного та інтелектуального маршрутів

Інтелектуальна маршрутизація також докорінно змінює процес призначення кур’єрів, що є критично важливим для швидкості та якості сервісу. Покладатися на диспетчера для ручного розподілення десятків чи сотень замовлень між кур’єрами – це повільний, неефективний процес, схильний до людських помилок, особливо в години пікового навантаження. Такий підхід не дозволяє врахувати всі змінні в реальному часі.

Натомість, інтелектуальна система автоматично підбирає найкращого виконавця для кожного замовлення за лічені секунди. Наприклад, якщо кур’єр щойно завершив доставку неподалік від ресторану, де готове нове замовлення, система миттєво запропонує це замовлення саме йому. Більше того, вона може об'єднати кілька замовлень для одного кур'єра, якщо вони прямують в одному напрямку. Це не тільки мінімізує час очікування для клієнта, а й оптимізує заробіток кур’єра. Приклад інтерфейсу призначення кур’єрів наведено на рисунку 1.7.

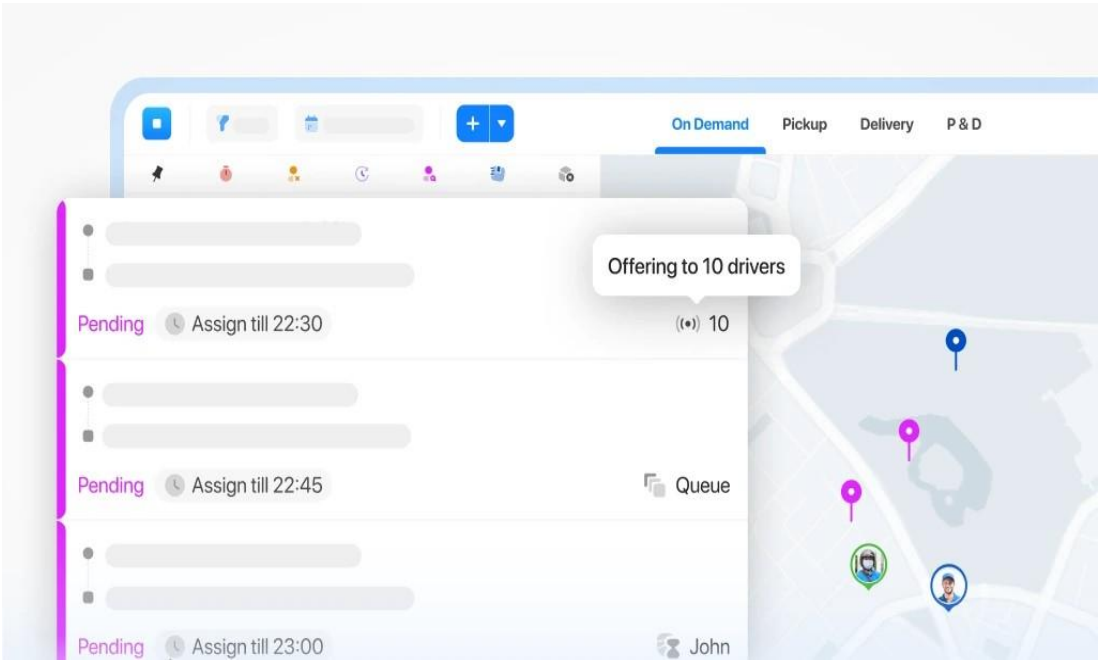

Рисунок 1.7 – Приклад інтерфейсу призначення кур’єрів

Однією з найбільших переваг інтелектуальної маршрутизації є її здатність адаптуватися до змін у реальному часі. Система може миттєво отримати оновлені дані про трафік через API картографічних сервісів, перерахувати маршрут і скоригувати прогноз ETA. Якщо ж затримка занадто велика, то система може передати замовлення іншому кур’єру і повідомити клієнта. Така гнучкість є критично важливою, особливо в Україні, де війна постійно додає виклики.

Також інтелектуальна маршрутизація приносить відчутні переваги для бізнесу. По-перше, вона скорочує витрати: оптимізовані маршрути зменшують витрати на паливо, а ефективне призначення кур’єрів дозволяє обробляти більше замовлень за той же час. Дослідження показують, що такі системи можуть значно скоротити час доставки і витрати. По-друге, точні прогнози ETA і швидка доставка підвищують задоволеність клієнтів. Інфографіка економії від інтелектуальної маршрутизації показана на рисунку 1.8.

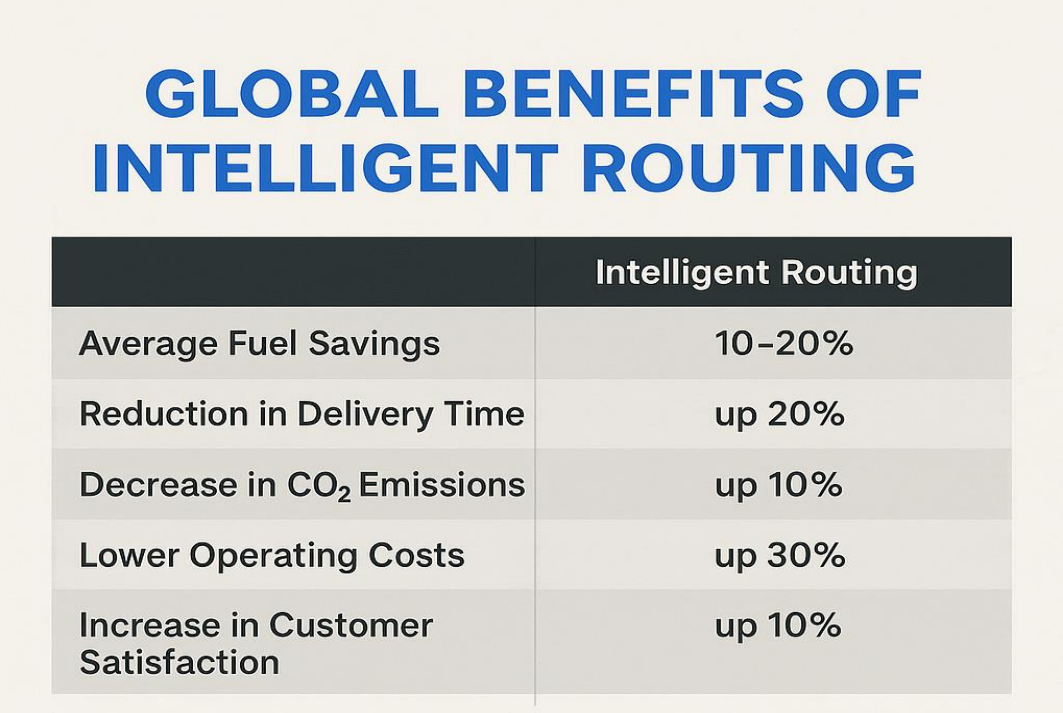

Рисунок 1.8 – Інфографіка економії від інтелектуальної маршрутизації

Проте інтелектуальна маршрутизація має свої недоліки. По-перше, вона потребує якісних і актуальних даних [14]. Якщо інформація про трафік чи погоду застаріла, то прогнози будуть менш точними. По-друге, інтеграція з існуючими бізнес процесами може бути складною, особливо для малого бізнесу. В Україні додаткові труднощі створює брак даних про стан доріг у зоні бойових дій.

Практичні приклади демонструють як інтелектуальна маршрутизація змінює сучасну логістику. Amazon використовує власні алгоритми для оптимізації маршрутів у своїй системі доставки. В Україні Нова Пошта починає впроваджувати елементи інтелектуальної маршрутизації. Такі реальні приклади демонструють актуальність та велику роль інтелектуальної маршрутизації в логістиці.

1.3 Огляд існуючих рішень та технологій для оптимізації маршрутів

На базовому рівні будь-якої системи оптимізації маршрутів лежать

картографічні сервіси. Вони надають карти, дані про трафік і інструменти для побудови маршрутів. Лідером тут, безумовно, є Google Maps Platform, який є золотим стандартом навігації. Google пропонує точні карти, інформацію про трафік у реальному часі та потужні API: Directions API, Routes API, Route Optimization API. Я обрав Google Maps API для реалізацію свого проекту саме через надійність його даних та потужність його API. Однак Google Maps API не є безкоштовним, через що для великої кількості запитів треба ретельно планувати бюджет.

Важливою альтернативою платним сервісам Google є OpenStreetMap (OSM) – глобальний, безкоштовний і відкритий картографічний проєкт. Його головна перевага полягає в тому, що дані створюються та оновлюються величезною спільнотою волонтерів, що робить їх неймовірно гнучкими і повністю безкоштовними для будь-якого використання. Для роботи з даними OSM існують потужні інструменти з відкритим кодом: OSRM (Open Source Routing Machine) і GraphHopper швидко будують маршрути, а Nominatim і Photon ефективно справляються з геокодингом (перетворенням адреси в географічні координати).

Було розглянуто OSM для проєкту, адже повна відсутність плати за API є значною економією коштів. Однак при детальному аналізі було виявлено кілька критичних проблем. По-перше, дані про трафік у реальному часі в OSM значно поступаються актуальності Google. Завдяки мільйонам активних користувачів, Google Maps практично миттєво фіксує затори, аварії та перекриття доріг, що є вирішальним для точного прогнозування часу доставки.

По-друге, якість самих мап та алгоритмів маршрутизації OSM, на жаль, гірше працює саме в межах України, особливо в невеликих містах та сільській місцевості. Це призводить до побудови не завжди оптимальних маршрутів. Порівняння можливостей картографічних API наведено на рисунку 1.9.

Comparison of Cartographic APIs

	Google Maps API	OSRM	Mapbox	HERE Maps
Traffic Accuracy	+	+	+	+
Cost	−	+	−	−
Customization	−	+	+	+
Geocoding Support	+	+	+	+

Рисунок 1.9 – Порівняння можливостей картографічних API

На наступному рівні технологічного стеку для логістики знаходяться спеціалізовані хмарні платформи, ще відомі як SaaS (Software as a Service). Це готовий інструментарій, створений для вирішення конкретних завдань бізнесу.Такі сервіси, як Route4Me, OptimoRoute, Routific вже пропонують готові рішення для планування маршрутів. Вони мають веб-інтерфейси, у яких менеджери можуть вводити адреси, планувати маршрути та відстежувати кур’єрів. Ці платформи дуже зручні, адже не вимагають знань програмування – усе працює вже з «коробки». Але тут також є мінуси: підписка може коштувати більше сотні доларів на місяць, що для малого бізнесу в Україні може бути непосильно. До того ж можливості кастомізації дуже обмежені. Якщо бізнес хоче додати щось унікальне, наприклад, власну модель машинного навчання для прогнозування часу доставки, такі платформи обмежують цю можливість. Приклад інтерфейсу SaaS-платформи Route4Me для маршрутизації наведено на рисунку 1.10.

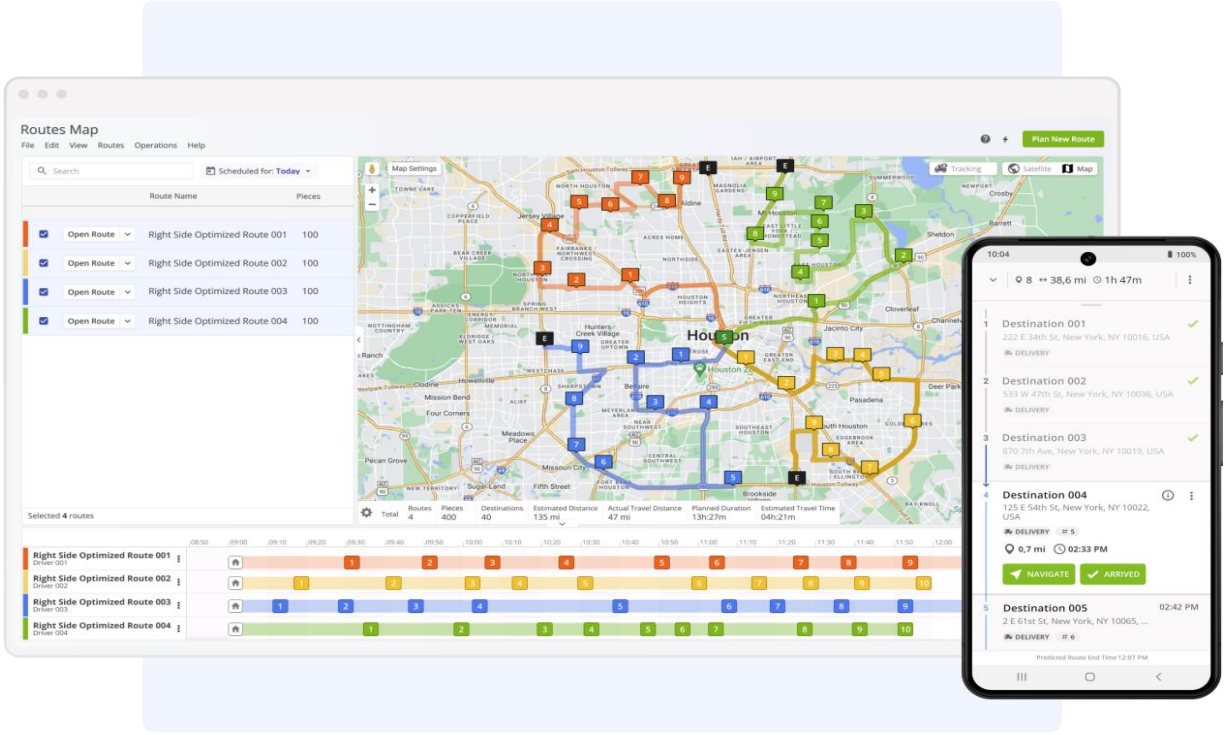

Рисунок 1.10 – Інтерфейс SaaS-платформи Route4Me для маршрутизації

Для великих міжнародних корпорацій зі складною, багатоетапною логістикою існують комплексні системи управління транспортом (TMS), серед яких лідерами є SAP Transportation Management та Oracle Transportation Management. Ці потужні платформи є справжніми «операційними центрами», що охоплюють абсолютно все: від стратегічного планування маршрутів та консолідації вантажів до управління власним автопарком, складами, митними процедурами та фінансами. Однак для малого чи середнього бізнесу такі системи є надмірними – занадто дорогими і складними.

Через це на ринку з'явилася гнучка альтернатива – хмарні SaaS-рішення. Вони працюють за моделлю підписки, не вимагають великих початкових інвестицій і дозволяють компаніям платити лише за ті функції, які їм потрібні. Якщо розглядати контекст України, то класичні дорогі TMS використовуються дуже рідко навіть великими компаніями. Приклад інтерфейсу TMS наведено на рисунку 1.11.

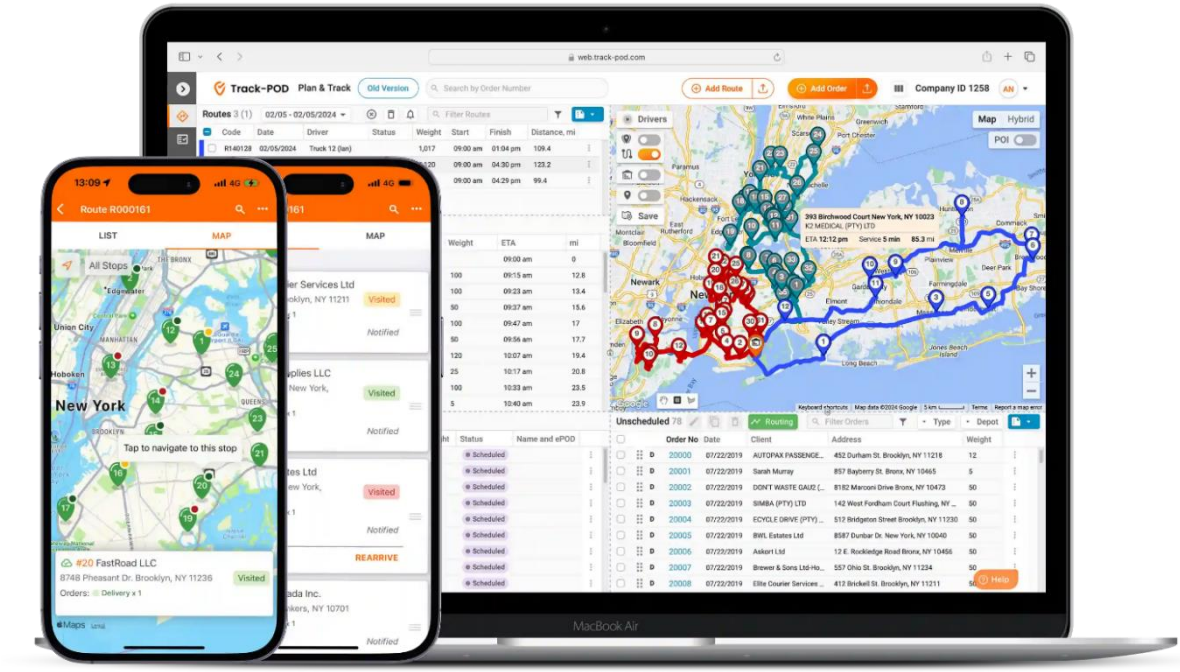

Рисунок 1.11 – Приклад інтерфейсу TMS

Окрім готових платформ також існують бібліотеки та фреймворки для компаній, які хочуть створювати власні рішення. У сфері машинного навчання стандартом є Python із бібліотеками Scikit-learn для базових моделей, XGBoost і LightGBM для прогнозування на табличних даних, а також TensorFlow і PyTorch для глибокого навчання. Наприклад, саме XGBoost часто використовується для прогнозування ETA, адже він дуже ефективно обробляє дані про трафік, відстань та замовлення. Для оптимізації маршрутів використовуються такі популярні інструменти, як Google OR-Tools або jsprit.

Як можна побачити, порівняння всіх цих рішень показує, що універсального інструменту не існує. Google Maps виграє в точності, але дорого коштує. OSM навпаки економить кошти, але втрачає точність і дані про трафік. TMS платформи потужні, але надто складні і дорогі. Бібліотеки дають свободу, але потребують технічних навичок або спеціалістів.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

У даному розділі детально розглядаються теоретичні основи та математичний апарат, що лежать в основі методів оптимізації логістичних процесів, прогнозування з використанням машинного навчання та застосування геоінформаційних технологій.

2.1 Основи оптимізації логістичних процесів

Логістика – це галузь, де потрібно знаходити найкращі рішення за наявність обмежених ресурсів: часу, транспорту чи грошей. Одним з ключових завдань є планування маршрутів для того, щоб зробити доставку швидкою та економною. Для виконання цього завдання використовуються різні методи дослідження операцій, математичного моделювання та теорії графів.

2.1.1 Формалізація та класифікація задач маршрутизації транспорту (Vehicle Routing Problems – VRP)

Задача Маршрутизації Транспортних Засобів (VRP) є однією з найбільш вивчених та водночас найскладніших задач комбінаторної оптимізації, яка лежить в основі логістики. VRP може розглядатися як кілька об’єднаних разом задач комівояжера (TSP) [15]. Задача полягає в тому, що потрібно знайти оптимальні маршрути для доставки товарів або надання послуг групою транспортних засобів (ТЗ), що базуються в одному або кількох депо. Метою є побудова такого набору маршрутів, який буде мінімізувати витрати, такі як загальна відстань або час у дорозі.

Формально VRP можна описати наступним чином: є граф із вершинами, які представляють собою депо та клієнтів. Ребрами будуть

дороги між ними. Кожне ребро має свою вагу, наприклад, відстань. Треба знайти маршрути, які мінімізують сумарну вагу. Це можна записати як:

$$\min Z = \sum_{k \in K} \sum_{i \in V} \sum_{j \in V, j \neq 1} c_{ij} x_{ijk}, \tag{2.1}$$

де c_{ij} – вага (наприклад, відстань);

x_{ijk} – 1, якщо транспортний засіб k їде по ребру, і 0 в іншому випадку.

Мінімізація відбувається за дотримання низки обмежень. По-перше, кожен клієнт має бути відвіданий рівно одним транспортним засобом:

$$\sum_{k \in K} \sum_{j \in V, j \neq 1} x_{ijk} = 1 \qquad \forall i \in C. \tag{2.2}$$

По-друге, кожен транспортний засіб, що прибуває до клієнта, повинен його покинути:

$$\sum_{j \in V, j \neq 1} x_{jik} - \sum_{j \in V, j \neq i} x_{ijk} = 0 \quad \forall i \in C, \forall k \in K. \tag{2.3}$$

По-третє, кожен маршрут має починатися і закінчуватися в депо (центральному складі). Це фундаментальне правило, адже саме з депо вранці виїжджають завантажені автомобілі, і саме туди вони повинні повернутися в кінці робочого дня. Крім того, існують обмеження, що запобігають утворенню підмаршрутів, які не включають депо. Це важлива технічна умова, яка гарантує, що рішення не міститиме абсурдних «ізольованих» кілець, де кур'єр обслуговує кількох клієнтів, але ніколи не повертається на базу. Графове представлення задачі VRP можна побачити на рисунку 2.1.

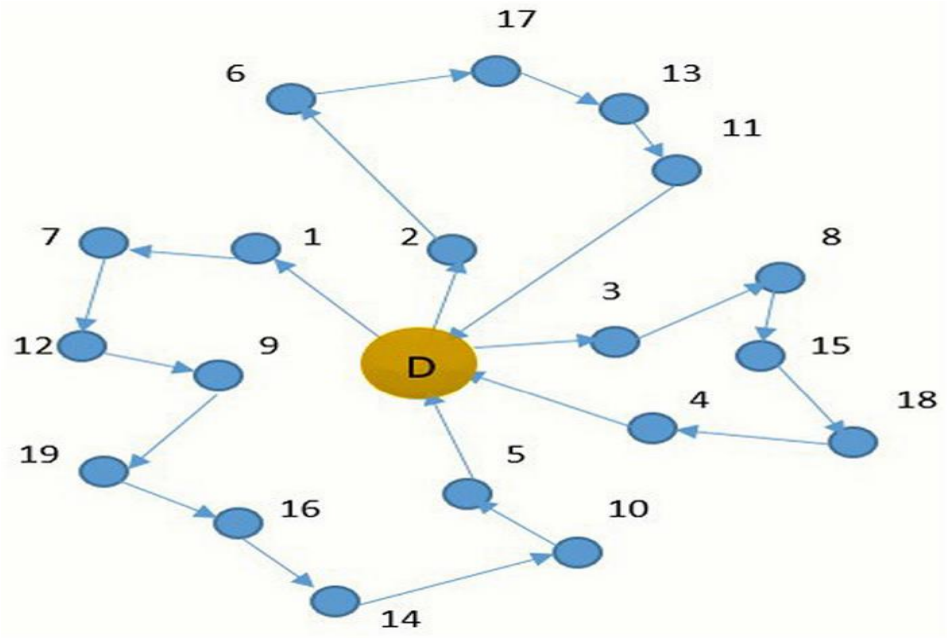

Рисунок 2.1 – Графове представлення задачі VRP

Задача маршрутизації транспорту (VRP) є NP-складною, тобто час знаходження точного, оптимального рішення для великої кількості клієнтів зростає експоненційно [16]. Це означає, що для 10 клієнтів кількість можливих маршрутів буде 10! = 3628800, а перебрати всі можливі маршрути, наприклад, для 57 клієнтів майже неможливо. NP-складність змушує шукати компроміс між якістю рішення та часом.

NP-складність змушує бізнес і розробників шукати компроміс між якістю рішення та часом, витраченим на його пошук. Точні методи, як-от метод гілок та меж чи лінійне програмування, хоч і гарантують знаходження найкращого рішення, займають неприйнятно багато часу і є ефективними лише для дуже малих задач. Тому в реальних сценаріях використовують евристики та метаевристики [17].

Прикладом простого евристичного методу є алгоритм найближчого сусіда. Ілюстрацію алгоритму найближчого сусіда можна побачити на рисунку 2.2.

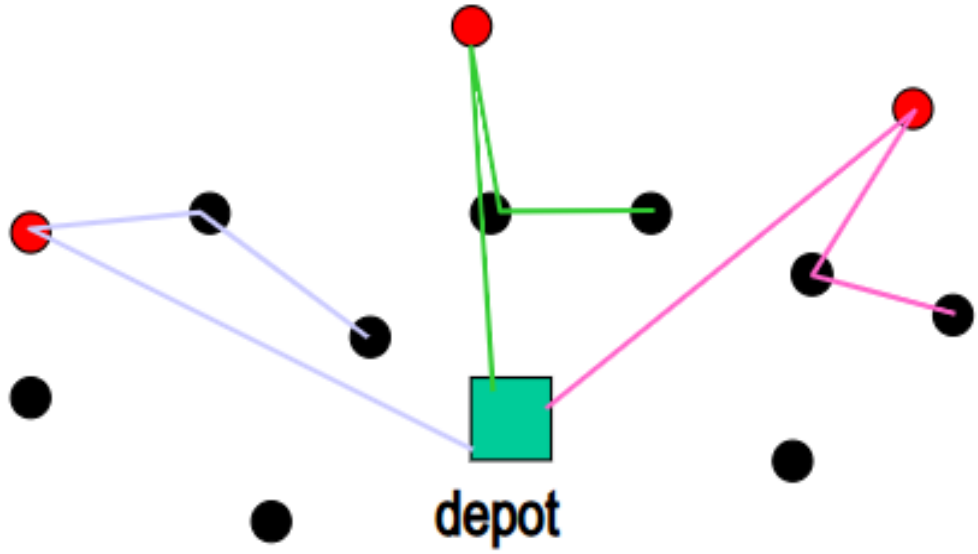

Рисунок 2.2 – Алгоритм найближчого сусіда

Цей метод швидкий, але може створювати неоптимальні маршрути, наприклад, змушуючи кур'єра їздити між віддаленими районами. Метаевристики, такі як генетичні алгоритми, табу-пошук, імітація відпалу чи мурашині алгоритми здатні знаходити кращі рішення. Наприклад, генетичний алгоритм створює популяцію маршрутів, потім комбінує частини маршрутів і змінює порядок клієнтів, щоб знайти оптимальний.

Реальні логістичні сценарії часто вимагають врахування додаткових умов, що призводить до різних варіацій VRP. Наприклад, CVRP (Capacitated VRP) вводить обмеження на вантажопідйомність кожного ТЗ, вимагаючи, щоб сумарний попит клієнтів на маршруті не перевищував місткості автомобіля.

$$\sum_{i \in C} d_i y_{ik} \leq Q_k \quad \forall k \in K, \tag{2.3}$$

де d_i – попит клієнта i;

y_{ik} – бінарна змінна, яка вказує, чи обслуговується клієнт і транспортним засобом k;

Q_k – вантажопідйомність транспортного засобу k.

VRPTW (VRP with Time Windows) додає обмеження на час прибуття до клієнта, тобто вимагаючи обслуговування в межах заданого часового проміжку [18]. MDVRP (Multi-Depot VRP) працює з кількома депо, а DVRP (Dynamic VRP) дозволяє додавати замовлення в процесі доставки. Ці варіації демонструють реальні сценарії, але й сильно ускладнюють задачу. Ієрархія задач VRP показана на рисунку 2.3.

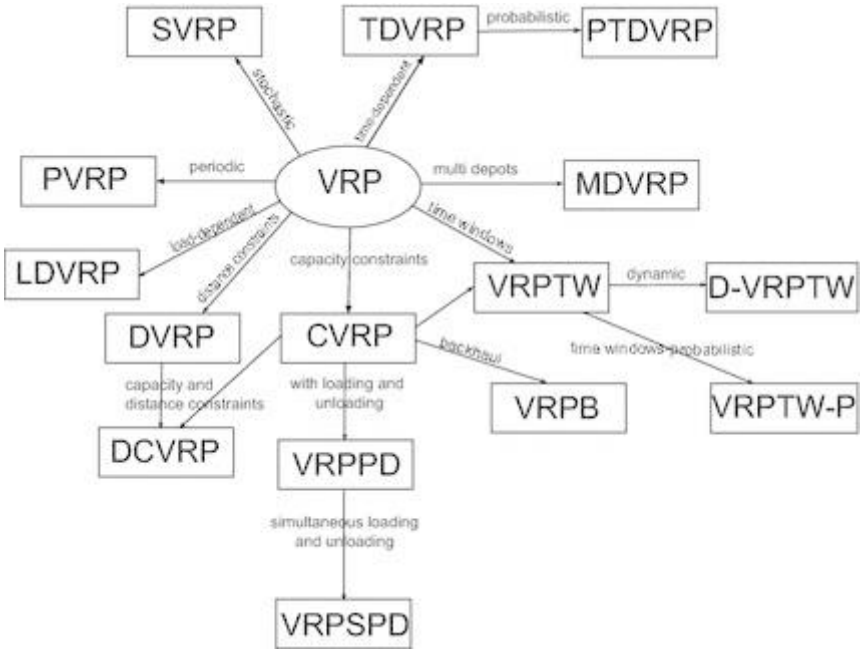

Рисунок 2.3 – Ієрархія задач VRP

2.1.2 Методи розв’язання задач VRP

Розв’язання VRP потребує розумних стратегій для того, щоб знайти хороший шлях швидко. Для цього існують три основні підходи: точні, евристичні та метаевристичні методи, кожен із яких має свої сильні та слабкі сторони.

Точні методи гарантують оптимальне рішення, але потребують значних обчислень і затрат у часі. Наприклад, метод гілок та меж (Branch-and-Bound) розбиває задачу на підзадачі, оцінює їхні межі і відсікає

неоптимальні гілки [19]. Інший метод – динамічне програмування, який розв'язує задачу шляхом розбиття на менші підзадачі й збереження їхніх рішень [20]. Але ці методи є ефективними лише для малих задач (до 20-30 клієнтів). Для реальних сценаріїв, як доставка в місті з сотнями замовлень, вони надто повільні через NP-складність.

Евристичні методи знаходять хороші, але не обов'язково оптимальні рішення за короткий час. Вони є простими і швидкими, що робить їх популярними у реальних системах. Наприклад, алгоритм Кларка-Райта є більш ефективним для CVRP. Він починає з маршрутів до кожного клієнта окремо, а потім об'єднує їх, максимізуючи економію відстаней:

$$s_{ij} = d_{0i} + d_{0j} - d_{ij}, \tag{2.4}$$

де s_{ij} – економія від об'єднання клієнтів і та j;

d_{0i} та d_{0j} – відстань від депо до і та j відповідно;

d_{ij} – відстань між клієнтами і та j.

Алгоритм об'єднує пари з найбільшою економією дотримуючись обмежень, як місткість. Евристики добре працюють для швидких рішень, але можуть застрягти в локальних оптимумах, пропускаючи кращі маршрути.

Метаевристичні методи балансують між швидкістю та якістю, досліджуючи простір рішень глибше, ніж евристики. Вони імітують природні чи фізичні процеси. Наприклад, генетичний алгоритм працює як еволюція:

- створити початкову популяцію маршрутів;
- оцінити їхню загальну відстань;
- схрестити кращі маршрути, комбінуючи їхні частини;
- додати «мутації» (змінити порядок клієнтів);
- повторити до знаходження оптимального рішення.

Імітація відпалу імітує охолодження металу: дозволяє тимчасово приймати гірші рішення, для того щоб уникнути локальних оптимумів, із поступовим зменшенням ймовірності таких кроків. Табу-пошук зберігає список заборонених рішень, щоб не повертатися до вже перевірених маршрутів. Ці методи ефективні для складних задач, таких як VRPTW чи DVRP, але потребують ретельного налаштування параметрів та більше часу, ніж евристики. Ілюстрацію методів вирішення VRP можна побачити на рисунку 2.4.

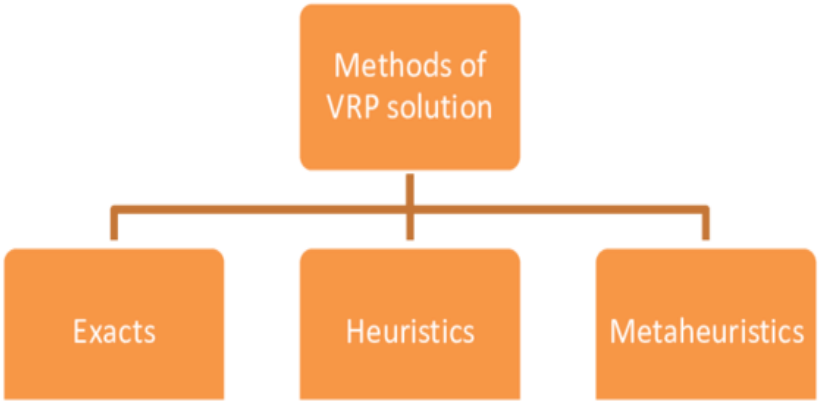

Рисунок 2.4 – Методи вирішення VRP

2.1.3 Ключові показники ефективності (KPIs) в логістиці доставки

Ключові показники ефективності (KPIs) – це метрики, які дозволяють оцінити наскільки добре працює логістична система [21]. Вони допомагають зрозуміти, чи економить бізнес ресурси, чи задоволені клієнти і чи можна покращити процеси. У логістиці останньої милі ці показники поділяються на фінансові, клієнтоорієнтовані та операційні.

Фінансові показники зосереджені на витратах. Середня вартість однієї доставки включає в себе витрати на паливо, зарплату кур’єру, плату за API чи навіть штрафи за запізнення. Наприклад, у містах України, де ціни на пальне значно зросли, скорочення цієї вартості є пріоритетом. Операційна

рентабельність показує, чи приносить доставка прибуток після всіх витрат. Ще одним показником є витрати на кілометр пробігу, що допомагає оцінити ефективність маршруту.

Операційні KPIs вимірюють продуктивність процесів. Середній час доставки – це період від готовності замовлення до його вручення. Для їжі стандартом є 30 – 45 хвилин, адже ніхто не хоче їсти холодну страву. Загальний добовий чи середній пробіг на доставку демонструють, наскільки оптимальними є маршрути. Утилізація ресурсів відображає, як використовується час і транспорт: якщо кур'єр третину зміни чекає замовлення – це простій. Прямим індикатором продуктивності є кількість доставок за зміну чи на один транспорт.

Водночас клієнтоорієнтовані KPIs оцінюють якість сервісу, що безпосередньо впливає на лояльність та утримання клієнтів. Своєчасні доставки – це відсоток замовлень, доставлених у межах обіцяного часу. Задоволеність клієнтів вимірюють через опитування, а Net Promoter Score (NPS) показує ймовірність, з якою клієнт порекомендує сервіс друзям. Точність прогнозу часу прибуття (ETA) є ключовою, адже значна розбіжність між обіцяним і реальним часом доставки руйнує довіру до сервісу.

Ці KPIs є взаємопов'язаними, але також можуть конфліктувати між собою, створюючи складну задачу для оптимізації. Наприклад, найшвидший маршрут для термінового замовлення може виявитись платним або неенергоефективним, що підвищує витрати. Або ж доставка в точний час, зручний для клієнта, може змусити кур'єра довго чекати, що знижує його утилізацію та кількість виконаних замовлень за день. Якісна система оптимізації має знаходити динамічний баланс між цими метриками, дозволяючи бізнесу гнучко керувати пріоритетами. Основні KPIs у логістиці продемонстровані на рисунку 2.5.

Рисунок 2.5 – Основні KPIs у логістиці

2.2 Застосування машинного навчання для прогнозування в логістичних системах

Машинне навчання стало ключовим інструментом у логістиці, що дозволяє прогнозувати час доставки чи попит з дуже високою точністю. Логістична можна уявити у вигляді складного механізму, де ML діє як інтелектуальний центр. Саме здатність прогнозувати тривалість логістичних операцій є фундаментальною для ефективного планування та управління.

2.2.1 Роль прогнозування в оптимізації логістики

Прогнозування є невід’ємною частиною ефективної логістики, особливо для доставки останньої милі, де швидкість і точність є вирішальними факторами. Воно дозволяє передбачити ключові параметри, такі як ETA, попит чи ймовірність затримок, що знижує невизначеність та покращує клієнтський досвід.

Прогнозування ETA є критично важливим для забезпечення

своєчасної доставки, що є одним із ключових показників ефективності, як було описано раніше [22]. Точний ETA дозволяє клієнтам планувати свій час, також зменшуючи ризики невдалих доставок, коли клієнт відсутній. Для компанії це буде означати більш ефективне використання кур'єрів і транспорту. У контексті України, де повітряні тривоги чи пошкоджені дороги можуть порушувати графік, точний ETA стає ще ціннішим.

Прогнозування попиту допомагає оптимізувати ресурси. Передбачення кількості замовлень у певний день або годину дозволяють компаніям підготувати необхідну кількість кур'єрів і транспорту. Це знижує ризик надлишку ресурсів, який веде до марних витрат. У нестабільних умовах, таких як війна в Україні, попит може коливатися непередбачувано, через що точні прогнози стають дуже важливими для компаній.

Математично прогнозування в логістиці часто є задачею регресії, де ціль – передбачити числове значення, наприклад, час доставки чи кількість замовлень.

$$y = f(X) + \varepsilon, \tag{2.5}$$

де y – цільова змінна (наприклад ETA);

X – вектор ознак (відстань, трафік, погода);

f – модель прогнозування;

ε – похибка.

Для затримок може застосовуватися класифікація, де модель визначає, чи буде доставка вчасною або ні (бінарний результат).

2.2.2 Огляд методів машинного навчання для задач регресії

Задачі прогнозування в логістиці, такі як передбачення ETA чи попиту, зазвичай вирішуються за допомогою методів регресії, що моделюють числові значення на основі набору ознак. Ми проаналізуємо три

основні методи: лінійну регресію, дерева рішень і методи опорних векторів (SVR) та порівняємо їх переваги та недоліки в контексті логістичних задач.

Лінійна регресія є найпростішою регресією, яка припускає лінійну залежність між ознаками та цільовою змінною. Така модель має вигляд:

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n, \tag{2.6}$$

де y – прогноз (наприклад ETA);

x_i – ознаки (відстань, трафік);

w_i – ваги;

w_0 – вільний член.

Ваги визначаються шляхом мінімізації середньоквадратричної помилки:

$$min \sum_{i=1}^{m} (y_i - \hat{y}_i)^2. \tag{2.7}$$

Лінійна регресія швидка, проста в реалізації та доволі легко інтерпретується. Коефіцієнт w_1 показує як зміна x_1 впливає на y. Однак цей метод має суттєві недоліки, особливо в логістиці. Він дуже погано справляється з нелінійними залежностями, які є типовими для цієї сфери. Наприклад, залежність часу доставки від часу доби не є лінійною – вона різко зростає в години пік. Лінійна модель не здатна вловити такі закономірності.

До того ж, лінійна регресія є чутливою до викидів (аномальних даних). Один випадок доставки, що зайняв аномально багато часу через ДТП, може суттєво викривити модель і погіршити всі інші прогнози. Ілюстрація лінійної регресії показана на рисунку 2.6.

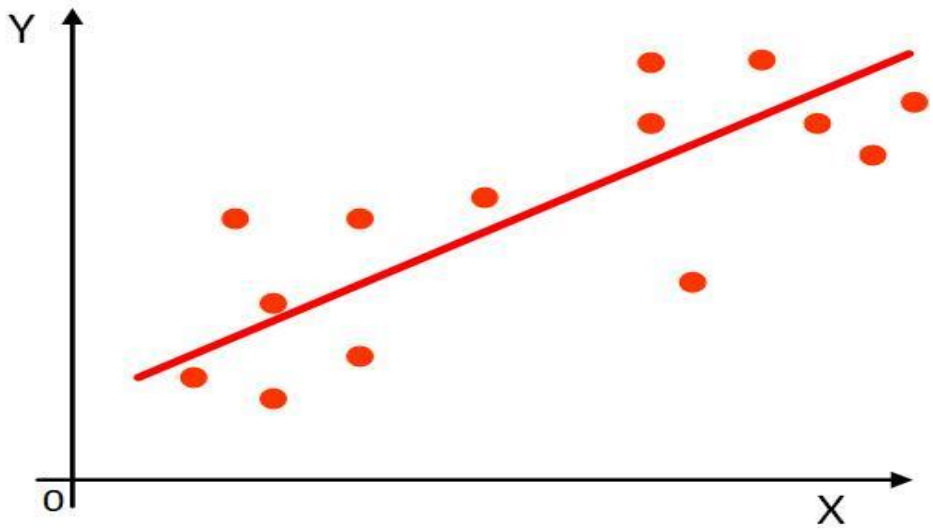

Рисунок 2.6 – Лінійна регресія

Дерева рішень моделюють залежності використовуючи ієрархічну структуру, де кожен вузол перевіряє умову, а листя містять прогнози. Цей алгоритм розбиває дані на підгрупи і в кожній мінімізує дисперсію:

$$min \sum_{i \in R} (y_i - \hat{y}_R)^2, \tag{2.8}$$

де R – підгрупа;

$\hat{y}_R$ – середнє значення в підгрупі.

Дерева рішень здатні моделювати нелінійні залежності та обробляти категоріальні ознаки (наприклад, тип транспорту) без додаткових перетворень. Наприклад, дерево може визначити, що в дощовий день ETA зростає на 20% для відстаней понад 5 км. Переваги дерева включають в себе інтуїтивність і гнучкість, але також є і недоліки. Дерева схильні до перенавчання: глибоке дерево може запам’ятати дані, а не узагальнити їх, через що точність буде великою лише на навчальній виборці. Також вони є нестабільними: навіть невеликі зміни в даних можуть змінити структуру дерева. У логістиці дерева рішень є корисними для базових прогнозів, але

їхня точність часто є нижчою, ніж у ансамблевих методів. Структуру дерева рішень можна побачити на рисунку 2.7.

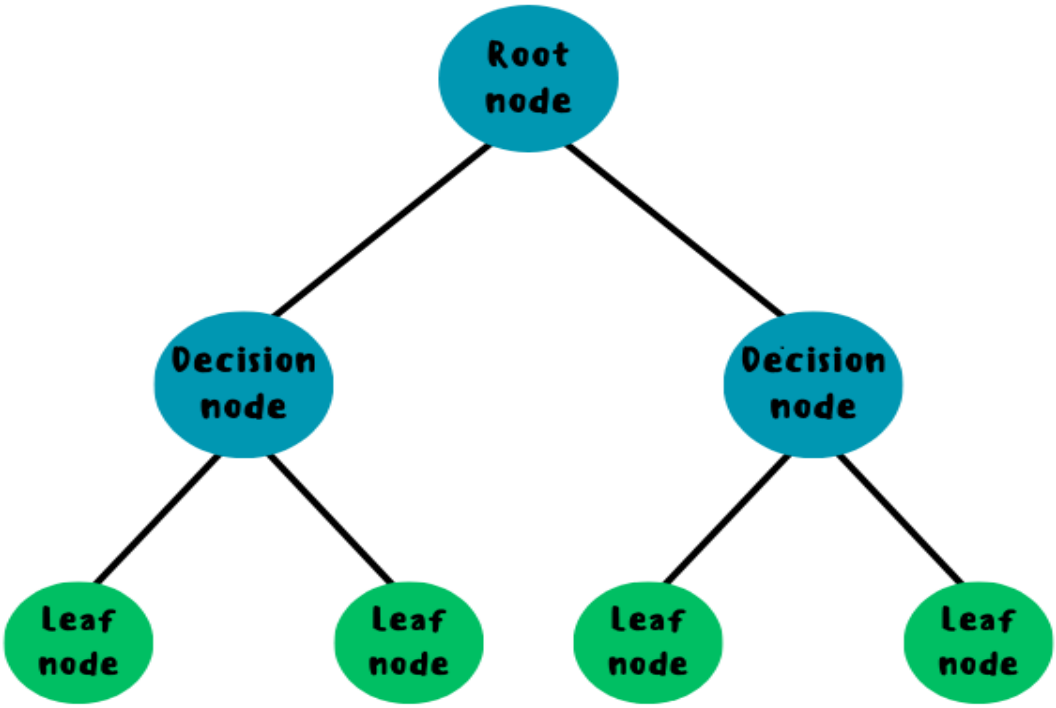

Рисунок 2.7 – Структура дерева рішень

Метод опорних векторів шукає функцію, яка передбачає значення з мінімальною помилкою, зберігаючи прогнози в межах «трубки» шириною ε.

$$\min \frac{1}{2}|w|^2 + C\sum_{i=1}^{m}\xi_i, \tag{2.9}$$

де w – ваги;

ξ_i – похибки;

C – параметр регуляризації.

SVR використовує ядрові функції, щоб моделювати нелінійні

залежності. Перевагами SVR є стійкість до викидів і висока точність на малих наборах даних. Наприклад, SVR може точно передбачити ETA для невеликого міста з обмеженим трафіком. Однак метод повільний і вимагає ретельного налаштування параметрів. У логістиці набори даних часто є великими і динамічними, через що SVR поступається швидшим методам.

2.2.3 Ансамблеві методи: градієнтний бустинг

Ансамблеві методи, зокрема градієнтний бустинг, є одним з найбільш ефективних інструментів для прогнозування в логістиці завдяки його здатності поєднувати кілька слабких моделей у сильну, яка перевершує окремі методи. Градієнтний бустинг працює шляхом послідовного навчання дерев рішень, де кожне нове дерево фокусується на зменшенні помилок попередніх. Математично модель представлена як сума дерев:

$$F(x) = \sum_{m=1}^{M} \gamma_m h_m(x), \tag{2.10}$$

де F(x) – прогноз;

$h_m(x)$ – m-те дерево;

γ_m – вага дерева;

M – кількість дерев.

На кожній ітерації алгоритм мінімізує функцію втрат, наприклад, середньоквадратичну помилку:

$$L = \sum_{i=1}^{n} (y_i - F(x_i))^2. \tag{2.12}$$

Нове дерево додається для коригування градієнта втрат:

$$h_m(x) = \frac{\partial L}{\partial F(x)}. \quad (2.13)$$

Цей процес дозволяє моделі поступово вдосконалювати свої прогнози, враховуючи складні нелінійні залежності, які є типовими для логістичних задач.

XGBoost (Extreme Gradient Boosting) – це вдосконалена реалізація градієнтного бустингу, яка включає в себе регуляризацію для запобігання перенавчанню [2.7]:

$$L = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{m=1}^{M} \Omega(h_m), \quad (2.14)$$

де l – функція втрат;

Ω – штраф за складність дерева, який залежить від кількості листків і їхніх ваг.

XGBoost має наступні ключові особливості:

– регуляризація: зменшує ризик перенавчання, що важливо для шумних даних;

– обробка пропусків: автоматично визначає оптимальний шлях для пропущених значень, що корисно при неповних даних;

– швидкість: оптимізований для паралельних обчислень, що дозволяє обробляти великі набори даних;

– гнучкість: підтримує різні функції втрат, що дозволяє прогнозувати ETA, попит та інше.

LightGBM (Light Gradient Boosting Machine) представляє собою ще швидшу альтернативу завдяки технікам Gradient-based One-Side Sampling (GOSS) і Exclusive Feature Bundling (EFB). GOSS фокусується на зразках із великими помилками, зменшуючи обсяг обчислень, а EFB об’єднує рідкісні ознаки, що зменшує розмір даних. LightGBM є особливо

ефективним для великих наборів даних, що є типовим для логістики, де можуть бути мільйони записів про доставку.

Переваги градієнтного бустингу для логістики включають високу точність завдяки моделюванню нелінійних залежностей, ефективну роботу з табличними даними, стійкість до шуму та гнучкість до різних задач. Недоліки – потреба в ретельному налаштуванні параметрів і значні обчислювальні ресурси. Проте в логістиці, де точність прогнозів прямо впливає на фінансові та клієнтські KPIs, ці методи виправдовують витрати.

У порівнянні з іншими методами регресії, розглянутими раніше, градієнтний бустинг перевершує лінійну регресію зі здатністю обробляти нелінійні залежності, дерева рішень – зі стабільністю, а SVR – зі швидкістю на великих даних. У логістиці, де дані часто містять шум і складні залежності, XGBoost і LightGBM є оптимальними для прогнозування. Порівняння XGBoost та LightGBM показано на рисунку 2.8.

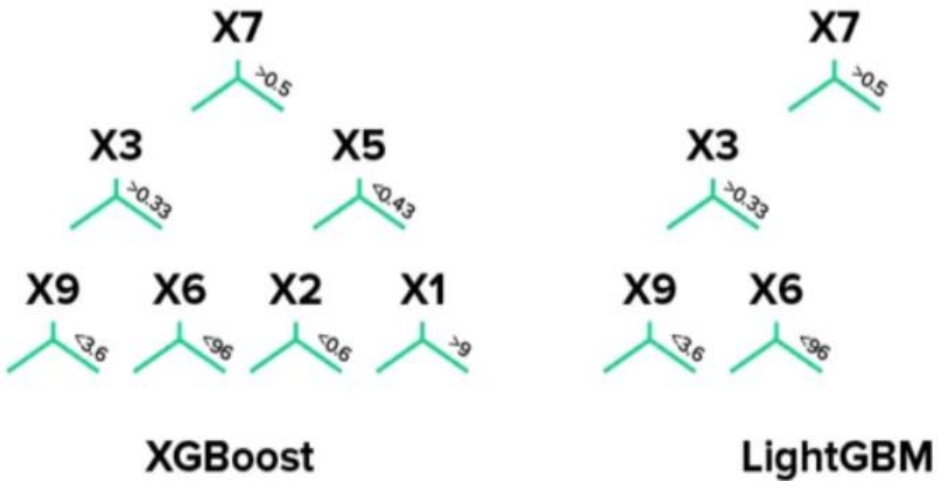

Рисунок 2.8 – Порівняння XGBoost та LightGBM

2.2.4 Інженерія ознак та підготовка даних для ML-моделей

Ефективність ML-моделей залежить від якості даних, які вони отримують. Інженерія ознак і підготовка даних є основою без якої модель

не буде надійною [23]. Ці процеси включають в себе очищення даних, обробку пропусків, створення нових ознак, кодування категоріальних змінних і масштабування числових.

Очищення даних є першим етапом, спрямованим на усунення помилок, дублікатів і викидів. Наприклад, запис про доставку на 100 км за 10 хвилин є аномалією, яку потрібно виправити або видалити. У логістиці дані часто містять шум через неточні GPS-координати, помилки введення чи збої в системах відстеження. Ретельне очищення забезпечує отримання моделлю достовірної інформації, що зменшує ризик спотворення прогнозів.

Обробка пропусків необхідна, оскільки логістичні дані можуть бути неповними. Пропуски можуть бути заповнені середніми значеннями, медіанами або прогнозами за допомогою алгоритмів, які знаходять схожі записи для заповнення. Деякі моделі, як XGBoost, автоматично обробляють пропуски, але попереднє заповнення покращує точність.

Створення нових ознак (feature engineering) є ключовим для підвищення якості моделі. У логістиці типові ознаки поділяються на декілька категорій:

– географічні: відстань між точками, щільність трафіку, тип дороги, координати пунктів доставки;

– часові: година, день тижня, місяць, святкові періоди;

– операційні: тип замовлення, час підготовки, тип транспорту.

Кодування категоріальних ознак необхідне, оскільки ML-моделі працюють із числовими даними. Наприклад, тип транспорту є категоріальним. Метод One-Hot Encoding створює бінарні стовпці для кожної категорії, що дозволяє моделі обробляти категорії. Однак це збільшує розмір даних, що може уповільнити обчислення. Label Encoding присвоює числові значення, але може ввести штучний порядок, що небажано для моделей, як лінійна регресія. У логістиці One-Hot Encoding

частіше використовується для ознак, через його нейтральність. Ілюстрація One Hot Encoding показана на рисунку 2.9.

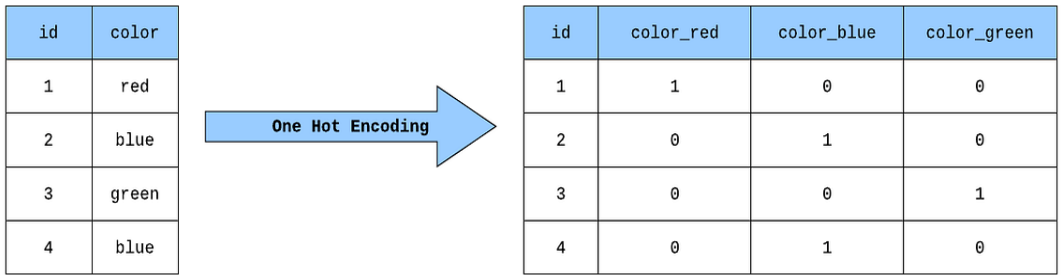

Рисунок 2.9 – One Hot Encoding

Масштабування числових ознак забезпечує однаковий вплив усіх змінних. Наприклад, відстань 0-100 км і трафік 0-1 мають різні масштаби, що може спотворити модель. Стандартне масштабування нормалізує дані:

$$x_{scaled} = \frac{x - \mu}{\sigma}, \tag{2.15}$$

де μ – середнє,

σ – стандартне відхилення.

Процес підготовки даних можна описати як послідовність етапів. Спочатку дані очищаються від помилок і дублікатів, щоб усунути аномалії. Далі пропуски заповнюються відповідними значеннями. Потім створюються нові ознаки. Категоріальні ознаки кодуються за допомогою One-Hot або Label Encoding залежно від моделі. І нарешті числові ознаки масштабуються для уніфікації їх впливу.

2.3 Геоінформаційні системи в інтелектуальній маршрутизації

Геоінформаційні системи (ГІС) та технології API картографічних

сервісів є фундаментом інтелектуальної маршрутизації, що забезпечує точні дані про географію, дороги та трафік. Вони дозволяють логістичним системам планувати оптимальні маршрути, прогнозувати час доставки та адаптуватися до змін у реальному часі.

2.3.1 Технології геокодингу

Геокодинг є процесом перетворення текстових даних у географічні координати широти і довготи, що є необхідним для точного визначення місцезнаходження точок [24]. Без геокодингу неможливо побудувати маршрут чи спрогнозувати ETA, оскільки системи потребують числових координат для обчислень. Зворотній геокодинг, навпаки, перетворює координати у адресу, що може бути корисним для відстеження кур'єрів.

Процес геокодингу починається з введення текстової адреси, наприклад, «вул. Хрещатик, 1, Київ, Україна». Система звертається до бази даних і зіставляє адресу з координатами.

Точність геокодингу критично впливає на маршрутизацію. Неточні координати можуть призвести до помилок у маршрутах, наприклад, відправлення кур'єра до неправильної точки, що збільшує час доставки. У логістиці навіть невелика похибка у 100 метрів може спричинити значні затримки, особливо в умовах щільного міського трафіку.

Популярні API для геокодингу включають Google Geocoding API, Nominatim і Mapbox Geocoding API. Google Geocoding API є високоточним, підтримує глобальні адреси та різні мови, повертаючи координати та форматовану адресу. Однак він є платним, із вартістю за кожен запит. Nominatim, що базується на OSM є безкоштовним, але має обмеження на частоту запитів і меншу точність для регіонів України. Mapbox Geocoding API пропонує комерційну альтернативу з підтримкою глобальних адрес і кастомізації, але є менш поширеним.

Геокодинг інтегрується з API маршрутизації для створення безперервного процесу планування. Наприклад, система спочатку геокодує адреси клієнтів, а потім використовує ці координати для обчислення оптимального маршруту. Теоретично геокодинг можна розглядати як задачу пошуку в базі даних. Ілюстрацію зворотнього геокодингу можна побачити на рисунку 2.10.

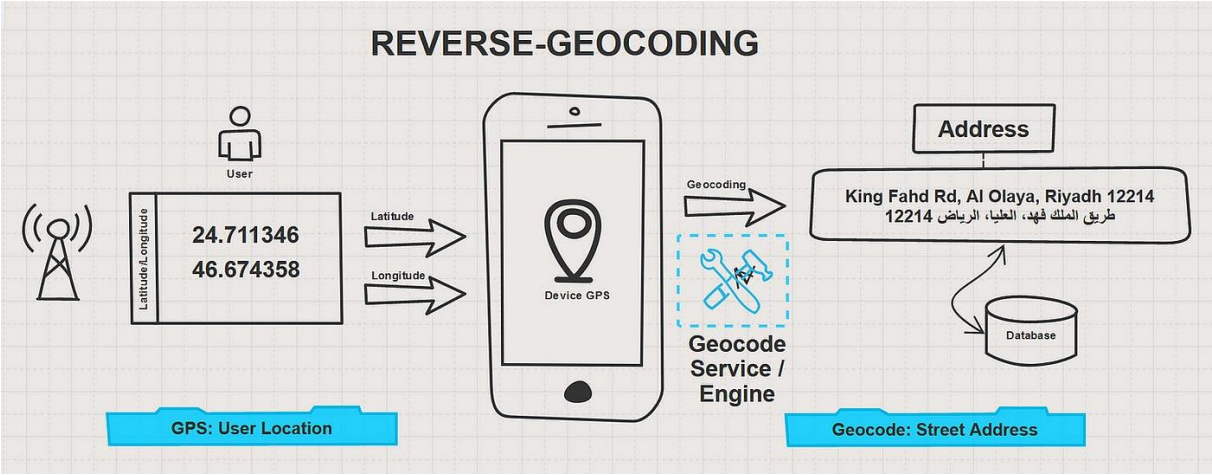

Рисунок 2.10 – Процес зворотнього геокодингу

2.3.2 API для побудови маршрутів та отримання даних про дорожній рух

API для побудови маршрутів, такі як Google Directions API і Routes API, є ключовими для інтелектуальної маршрутизації, оскільки вони обчислюють оптимальні шляхи. Ці API забезпечують логістичні системи деталізованою інформацією, необхідною для прогнозування ETA.

Google Directions API надає покрокові напрямки між двома або кількома точками. Запит до API включає початкову точку, кінцеву точку, режим транспорту і додаткові параметри. Відповідь API є JSON-об’єктом, що містить кроки маршруту, загальну відстань, прогнозований час у дорозі і альтернативні маршрути.

Google Routes API є більше просунутим інструментом, призначеним для складних сценаріїв, таких як маршрути з кількома точками. Він дозволяє оптимізувати порядок відвідування точок для мінімізації часу і відстані, враховувати тип транспортного засобу, надавати екологічні маршрути, які мінімізують витрати пального і обчислювати плату за проїзд платними дорогами. Routes API повертає деталізовані дані, включаючи ETA, відстань, кроки маршруту та альтернативні шляхи, що робить його ідеальним для логістики з кількома доставками. Ілюстрація використання Google Routes API показана на рисунку 2.11.

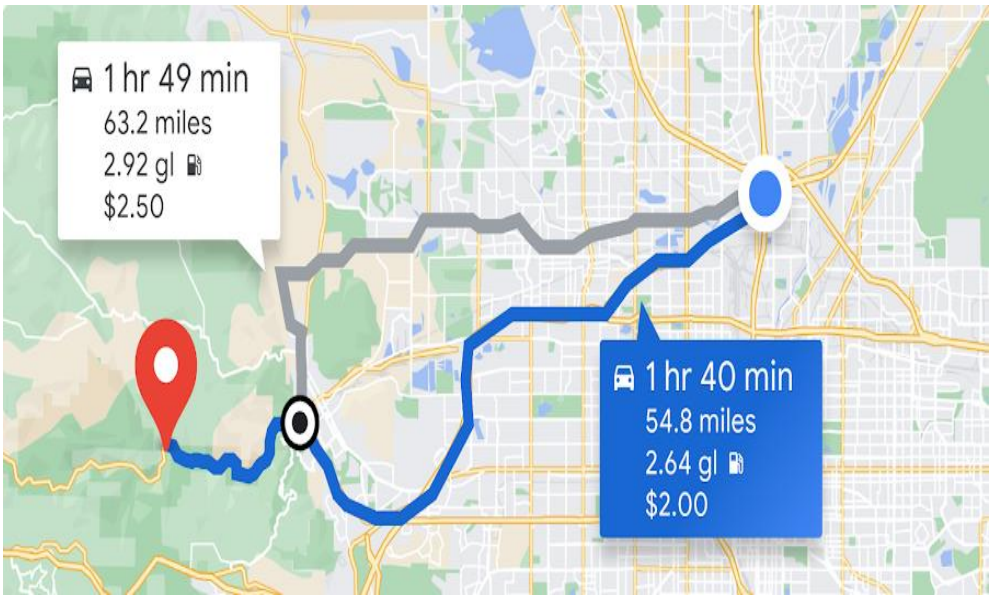

Рисунок 2.11 – Приклад використання Google Routes API

2.3.3 Принципи взаємодії з API картографічних сервісів

Взаємодія з API картографічних сервісів базується на принципах клієнт-серверної архітектури, де клієнт надсилає HTTP-запити до сервера, а сервер повертає структуровані дані. Цей процес є основою для отримання картографічної інформації, такої як координати, маршрути чи дані про трафік. Це вимагає розуміння технічних аспектів, таких як формати даних, автентифікація та обробка відповідей.

Архітектура REST API є стандартом для картографічних сервісів. REST (Representational State Transfer) використовує HTTP-методи, переважно GET і POST, для взаємодії. Наприклад, GET-запит до Google Directions API має вигляд:

```
GET https://maps.googleapis.com/maps/api/directions/json?origin=Kyiv&destination=Lviv&key=YOUR_API_KEY
```

Тут origin і destination – параметри запиту, а key – API-ключ для автентифікації. Запити можуть включати додаткові параметри, як режим транспорту чи вибір альтернативних доріг. POST-запити використовуються для складних ситуацій, де обсяг даних перевищує GET ліміт.

Формат даних у відповідях зазвичай JSON, який представляє інформацію у вигляді ключ-значення. JSON є дуже зручним для парсингу та підтримується багатьма мовами програмування. Альтернативний формат – XML, але він менш поширений через складність обробки.

Автентифікація забезпечується через API-ключі, які додаються до кожного запиту. Ключ ідентифікує користувача і обмежує кількість запитів, щоб запобігти зловживанню. Наприклад, Google Maps API вимагає унікального ключа, створеного в Google Clound Console. Без ключа запит буде відхилений, що забезпечує безпеку і контроль доступу.

У логістиці взаємодія з API є безперервним процесом. Система може періодично надсилати запити до Routes API для оновлення маршрутів на основі нових даних про трафік, що дозволяє адаптуватися до змін. Однак часті запити до платних API, як Google Maps, потребують обережності і ретельного управління бюджетом.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Цей розділ присвячено технічним аспектам розробки системи оптимізації логістичних процесів. Було розглянуто архітектурні рішення, обраний стек технологій, структура компонентів і принцип їх взаємодії. Метою розділу є надання чіткого уявлення щодо технічної реалізації, пояснити вибір технологій та показати їх взаємодію між собою для вирішення завдання оптимізації логістики.

3.1 Загальна архітектура системи

Архітектура програмного забезпечення являє собою фундамент, який визначає наскільки система буде стійкою, гнучкою і здатною витримувати навантаження. Для системи оптимізації логістичних процесів було обрано архітектуру, яка забезпечує ефективний розподіл функцій, масштабованість і надійність. Система враховує специфіку логістичних задач, таких як прогнозування часу доставки і управління маршрутами.

3.1.1 Огляд архітектури

Система побудована на основі трирівневої клієнт-серверної архітектури, що є класичним підходом для створення веб-додатків. Цей вибір був зумовлений потребою чітко розмежувати відповідальність між компонентами, що спрощує розробку та підтримку. Трирівнева архітектура складається з рівня представлення, рівня застосунку та рівня даних, кожен з яких виконує свої унікальні функції.

Рівень представлення відповідає за взаємодію з користувачем – менеджерами логістики та кур’єрами. Він реалізований як веб-додаток з використанням бібліотеки React, яка дозволяє створювати динамічні користувацькі інтерфейси. Для менеджерів інтерфейс надає можливості

управління замовленнями, призначення кур’єрів та перегляд аналітики, а для кур’єрів – доступ до списку доставок, маршрутів і статусів. React забезпечує компонентний підхід, де кожен елемент інтерфейсу являє собою окремий модуль. Веб-додаток взаємодіє з сервером завдяки REST API, надсилаючи HTTP-запити та отримуючи відповіді в форматі JSON.

Рівень застосунку, або бізнес логіки, є серцем системи. Він реалізований у вигляді серверної частини на мові програмування Java з використанням фреймворку Spring Boot. Бекенд спроектовано як монолітний додаток, але з чіткою модульною структурою. Модульність досягається шляхом поділу функціональності на окремі компоненти. Контролери обробляють HTTP-запити від клієнта, сервіси реалізують бізнес логіку, а репозиторії забезпечують доступ до бази даних. Наприклад, сервіс управління замовленнями перевіряє доступність кур’єрів, тоді як сервіс безпеки перевіряє права доступу. Такий поділ дозволяє легко модифікувати окремі частини без впливу на всю систему. Бекенд також взаємодіє з модулем машинного навчання, викликаючи Python-скрипт для прогнозування ETA, що додає інтелектуальності системі.

Рівень даних відповідає за зберігання інформації, такої як замовлення, дані користувачів, маршрути та статуси доставок. Для цього обрано PostgreSQL – реляційну базу даних, яка забезпечує надійність і підтримку складних запитів. База даних структурована за принципами нормалізації, щоб уникнути дублювання даних і забезпечити швидкий доступ. Наприклад, таблиця замовлень пов’язана із таблицею кур’єрів через зовнішні ключі, що дозволяє ефективно отримувати інформацію про доставки. Spring Data JPA спрощує взаємодію з базою, дозволяючи працювати з даними як з об’єктами Java, і автоматично генерує SQL-запити для стандартних операцій. Архітектурна схема системи зображена на рисунку 3.1.

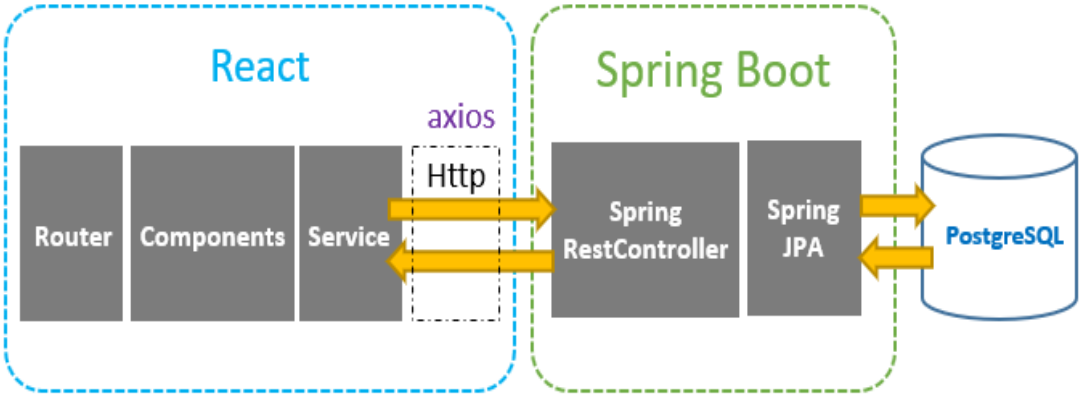

Рисунок 3.1 – Архітектурна схема системи

Окремим компонентом є модуль машинного навчання, який прогнозує ETA на основі даних про замовлення, трафік і погодні умови. Цей модуль розроблено на Python і викликається бекендом через системний виклик, що передає вхідні дані і повертає прогноз. Відокремлення ML-модуля від основного бекенду дозволяє незалежно розвивати обидві частини, хоча це й вимагає додаткових зусиль для інтеграції. У майбутньому архітектуру можна замінити на мікросервісну, де ML-модуль міг би працювати як окремий сервіс, підвищуючи продуктивність. Потоки даних у системі показані на рисунку 3.2.

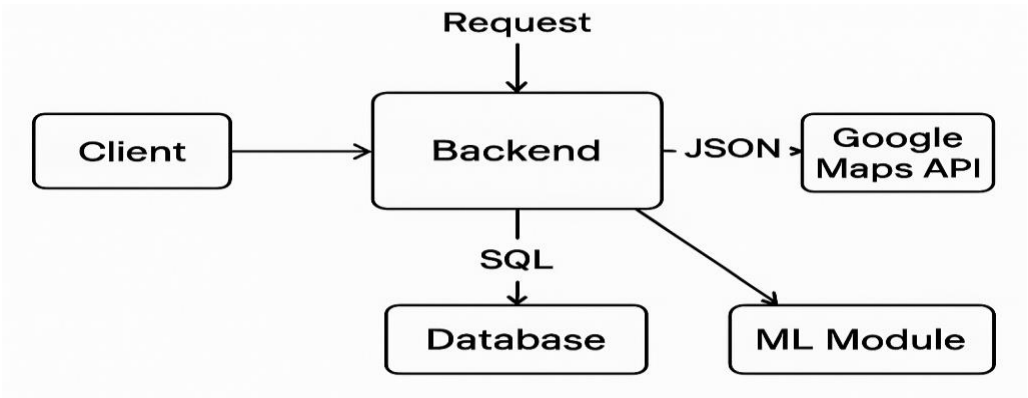

Рисунок 3.2 – Потоки даних у системі

3.1.2 Використані технології та інструменти

Для створення системи використано сучасний технологічний стек, який поєднує продуктивність, надійність і простоту розробки. Кожен інструмент обрано з урахуванням вимог до логістичної системи, таких як обробка великої кількості запитів, забезпечення безпеки та інтеграція із картографічними сервісами.

Серверна частина розроблена на мові Java, яка відома своєю стабільністю, строгою типізацією та широкою екосистемою. Java ідеально підходить для корпоративних додатків, оскільки забезпечує високу продуктивність і підтримку паралельних обчислень [25]. Фреймворк Spring Boot обрано за його здатність спрощувати конфігурацію серверів, надаючи вбудований Tomcat і готові модулі для REST API, безпеки та роботи з даними. Spring Security із механізмом JSON Web Token (JWT) захищає систему від несанкціонованого доступу, генеруючи токени для автентифікації користувачів. Наприклад, менеджер логістики отримує токен після входу, який перевіряється при кожному запиті до захищених ендпоінтів. Spring Data JPA разом із Hibernate абстрагує роботу із базою даних, дозволяючи писати запити на рівні об’єктів, а не SQL. Управління залежностями та збірка проекту здійснюються завдяки Apache Maven, який автоматизує процеси компіляції та тестування. Альтернативою був Python з його фреймворком Flask, але він менш підходить для складних монолітних систем через слабку типізацію та меншу кількість вбудованих інструментів. Також, Spring Boot має велику спільноту, що дуже полегшує пошук рішень для проблем, які з’являються під час розробки.

Клієнтська частина реалізована як веб-додаток на основі React. React використовує JavaScript, що забезпечує гнучкість і швидкість розробки. Для оформлення інтерфейсу обрано бібліотеку Material-UI, яка пропонує готові компоненти, адаптовані до сучасних стандартів дизайну. Взаємодія з бекендом здійснюється через HTTP-запити завдяки бібліотеці Axios, яка

спрощує обробку асинхронних запитів і помилок. Альтернативою був Angular, але він складніший у навчанні та потребує більше часу на розробку.

Модуль машинного навчання розроблено на Python, оскільки ця мова є стандартом у сфері Data Science. Бібліотеки Pandas і NumPy використовуються для обробки даних, наприклад, для очищення пропусків чи створення нових ознак. Scikit-learn забезпечує інструменти для попередньої обробки і оцінки моделей. Алгоритм XGBoost обрано для прогнозування ETA завдяки його ефективності на табличних даних, що підтверджується дослідженнями. Збереження моделі здійснюється через Joblib, що дозволяє швидко завантажувати її для прогнозів.

Для зберігання даних обрано PostgreSQL, яка підтримує транзакції, складні запити та забезпечує цілісність даних. Це критично для логістики, де втрата чи дублювання даних може призвести до фінансових збитків.

Картографічні функції, такі як відображення маршрутів і маркерів, реалізовані через Google Maps Platform, зокрема завдяки Maps JavaScript API для інтерактивних карт і Directions API для побудови маршрутів. Ці API надають точні дані про відстань та трафік, що доповнюють власну ML-модель.

Розробка велася в IntelliJ IDEA Ultimate для бекенду, WebStorm для фронтенду, PyCharm для Python-скриптів. Тестування API проводилось через Postman. Ці інструменти забезпечили ефективну та просту організацію роботи.

3.2 Проектування та реалізація бази даних

База даних є серцем будь-якої інформаційної системи. У системі оптимізації логістичних процесів база даних забезпечує надійне зберігання, управління та доступ до інформації.

3.2.1 Концептуальна модель даних

Концептуальне проектування – це перший крок, який допомагає зрозуміти, які саме дані потрібні системі. На цьому етапі було визначено основні сутності, що відображають об’єкти логістичної предметної області, їхні атрибути та взаємозв’язки.

Однією з центральних сутностей є користувач, який представляє всіх учасників системи: менеджерів, кур’єрів і, потенційно, адміністраторів системи. Користувач має унікальний ідентифікатор, ім’я, прізвище, захешований пароль для безпеки, унікальну електрону пошту та номер телефону. Додатково було передбачено поле секретного коду для менеджерів, яке пов’язує менеджерів з ресторанами. Кожен користувач пов’язаний із роллю, яка визначає його права, наприклад, «ROLE_MANAGER» для менеджерів або «ROLE_COURIER» для кур’єрів.

Іншою важливою сутністю є ресторан, адже система орієнтована на доставку із закладів харчування. Ресторан характеризується унікальним ідентифікатором, назвою, адресою, географічними координатами (широта і довгота), кодом для призначення менеджеру. Геокоординати дозволяють точно визначати розташування, що є критичним для маршрутизації.

Сутність замовлення об’єднує в собі інформацію про доставку. Воно включає ідентифікатор, ресторан, кур’єра, статус замовлення, дати створення та доставки, адресу й координати доставки, дані клієнта, ідентифікатор менеджера, загальну вартість і тип замовлення. Поле для пропозиції замовлення кур’єру дозволяє відстежувати, кому воно було запропоновано.

Позиція замовлення деталізує , які товари чи страви входять до замовлення. Вона містить ідентифікатор, посилання на замовлення, назву продукту, кількість, ціну за одиницю та загальну вартість позиції. Ця сутність необхідна для точного розрахунку вартості та аналізу попиту.

Сутність статусу кур’єра відображає поточний стан кур’єра, включаючи його рейтинг, тип транспорту і його статус. Ця сутність пов’язана з користувачем, забезпечуючи однозначну відповідність.

Взаємозв’язки між сутностями визначають їхню залежність. Кожен користувач має лише одну роль, але одна роль може застосовуватися до багатьох користувачів. Ресторан пов’язаний із одним менеджером через ідентифікатор. Замовлення належить одному ресторану, може бути призначине лише одному кур’єру, створене одним менеджером і містити кілька позицій.

3.2.2 Логічна модель даних

Логічна модель деталізує концептуальну, додаючи специфіку атрибутів і типів зв’язків. Вона враховує вимоги до нормалізації, щоб уникнути дублювання даних і забезпечити їхню цілісність.

Сутність користувачів включає унікальний ідентифікатор, ім’я, прізвище, захешований пароль, унікальну електрону пошту та унікальний номер телефону, але він не є обов’язковим, секретний код і посилання на роль. Роль має ідентифікатор і унікальну назву. Зв’язок між користувачами та ролями – один-до-багатьох.

Ресторани мають ідентифікатор, унікальну назву, адресу, координати, код для менеджера. Зв’язок із користувачами (менеджерами) є нуль або один, адже ресторан може бути без менеджера на етапі створення.

Нормалізація до третьої нормальної форми була ключовою. Наприклад, перенесення ролей в окрему таблицю усуває залежність від неключових атрибутів, а зв’язки через зовнішні ключі запобігають дублюванню. Нормалізація знижує ризик аномалій при вставці чи оновленні, що є критичним для логістики.

ER-діаграма була дуже корисною під час розробки, хоча її створення і вимагало часу. ER-діаграма системи зображена на рисунку 3.3.

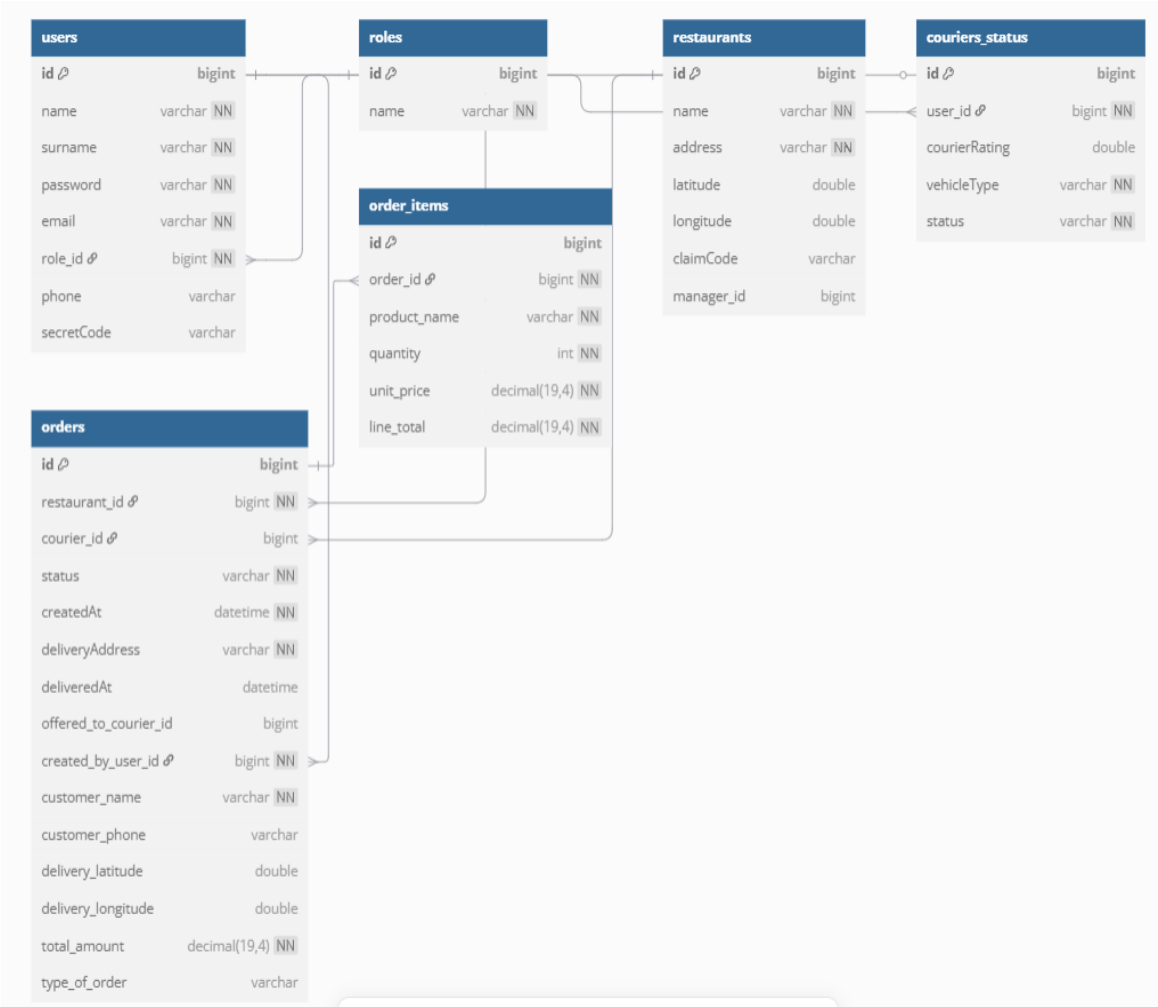

Рисунок 3.3 – ER-діаграма системи.

3.3 Реалізація серверної частини

3.3.1 Загальна структура backend частини додатку

Бекенд було розроблено на мові Java із фреймворком Spring Boot, що забезпечує надійність і модульність, як було описано раніше. Додаток є монолітним, але структурованим за шаровою архітектурою, що розділяє відповідальність між компонентами. Модульність досягається шляхом чіткого поділу на пакети, що спрощує розробку і підтримку системи.

Пакет для JPA-сутностей містить класи, що відображають таблиці бази даних. JPA-анотації, як @Entity, @ManyToOne, @OneToMany,

забезпечують мапінг на PostgreSQL, а методи, як recalculateTotalAmount, автоматизують розрахунки. Цей підхід дуже зручний, адже дозволяє працювати з даними як з об’єктами.

Пакет репозиторіїв визначає інтерфейси, що наслідують JpaRepository, для операцій із базою даних. Наприклад, OrderRepository підтримує CRUD і кастомні запити, як пошук замовлень за статусом. Використання JpaSpecificationExecutor додає гнучку фільтрацію, що корисно, наприклад, для головної сторінки менеджеру.

Сервісний пакет відповідає за інкапсулювання бізнес-логіки. OrderService обробляє створення замовлень, призначення кур’єрів, тоді як UserService керує користувачами та автентифікацією. Кожен сервіс ізольований, що значно спрощує тестування.

Пакет контролерів є вхідною точкою для системи. Він реалізує REST-ендпоінти, які приймають HTTP-запити від фронтенду і повертають JSON-відповіді. Контролери відповідають за валідацію вхідних даних та делегування бізнес-логіки сервісному шару. Наприклад, OrderController обробляє створення та оновлення замовлень, використовуючи DTO для передачі даних. DTO-пакет містить об’єкти, як OrderCreateRequest, що забезпечують чіткий контракт між фронтендом і бекендом.

За безпеку відповідає пакет безпеки (security). Він включає конфігурацію Spring Security, яка визначає правила доступу до ендпоінтів. Центральним елементом є кастомний фільтр JWT, який перехоплює кожен запит, перевіряє валідність токена за допомогою JwtTokenProvider і, в разі успіху, аутентифікує користувача в системі. Конфігураційний пакет (config) містить налаштування для всього застосунку, зокрема правила CORS для дозволу крос-доменних запитів від фронтенду, а також інші параметри, як-от шлях до Python-скрипта для ML-моделі.Структура проекту показана на рисунку 3.4.

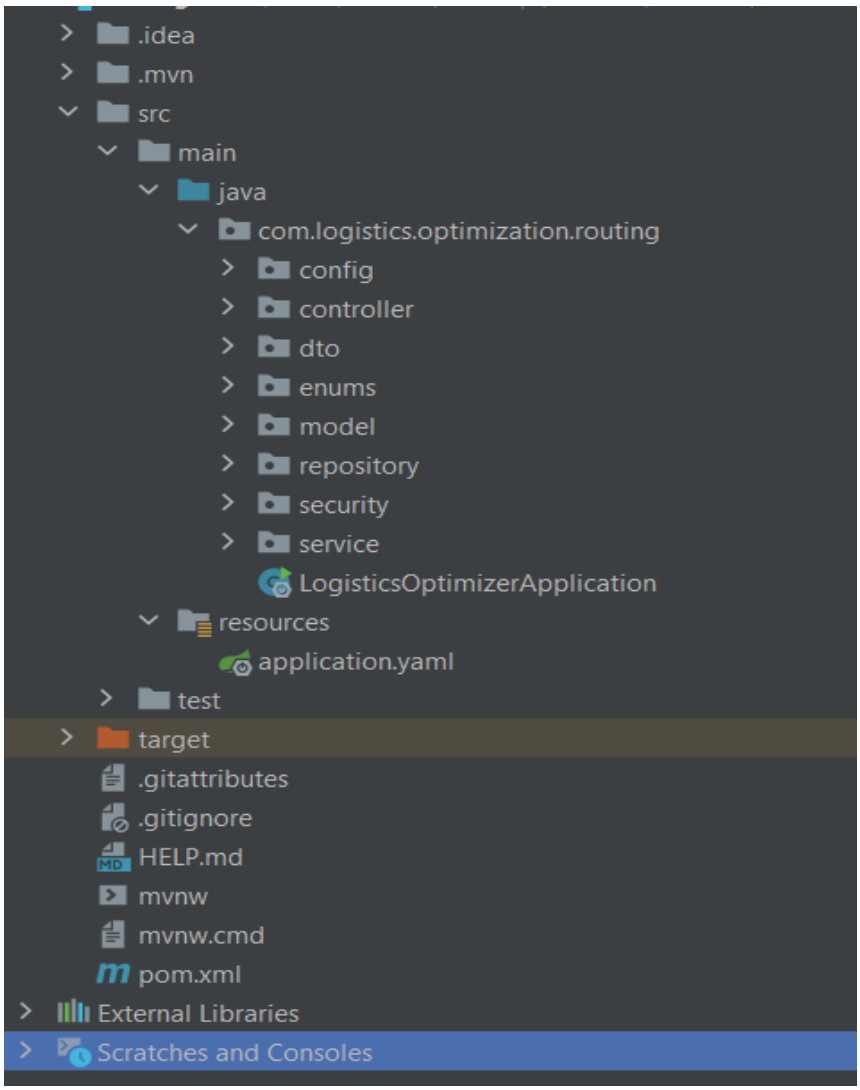

Рисунок 3.4 – Структура проекту

3.3.2 Опис ключових API ендпоінтів

RESTful API є своєрідним мостом між фронтендом і бекендом. Усі ендпоінти мають префікс /v1, що показує версію системи, та повертають JSON-відповіді.

Ендпоінти автентифікації /v1/auth/** відповідають за вхід і реєстрацію. POST /v1/auth/login приймає email і пароль, повертаючи JWT-токен і роль. POST /v1/auth/register створюють користувачів із ролями кур'єра або менеджера, повертаючи статус 201 Created. Ці ендпоінти є публічними, що дає доступ усім користувачам.

Ендпоінти управління замовленнями /v1/orders є основним. POST /v1/orders створює замовлення, приймаючи DTO із даними про клієнта, адресу, позиції та тип. Цей метод використовує анотацію @PreAuthorize для обмеження доступу. GET /v1/orders повертає пагінований список із фільтрами (статус, ресторан), що є корисним для менеджерів. GET /v1/orders/{id} надає деталі замовлення, включаючи позиції та ETA. PATCH /v1/orders/{id}/status оновлює статус замовлення, наприклад, з PENDING на ON_DELIVERY. POST /v1/orders/{id}/offer, /accept, /reject керують пропозиціями кур'єрам. Наприклад, менеджер пропонує замовлення кур'єру через offer, а кур'єр приймає його через accept.

Ендпоінти для кур'єрів /v1/courier-status/ дозволяють оновлювати статус, транспорт, а також переглядати поточний статус. Менеджери отримують список доступних кур'єрів через GET /v1/manager/couriers/available, що підтримують пагінацію.

Головні сторінки надають зведені дані. GET /v1/manager/dashboard повертає статистику для менеджера, кількість замовлень, активних кур'єрів. GET /v1/courier/dashboard повертає статистику для кур'єра , призначені замовлення та його заробіток.

Ендпоінт прогнозування ETA /v1/eta/predict приймає JSON із ознаками (відстань, час доби) і повертає прогноз у хвилинах, викликаючи Python-скрипт. Скрипт завантажує навчену модель, робить прогноз і повертає результат, який Java-додаток віддає клієнту у вигляді прогнозованого часу доставки в хвилинах.

Час приготування розраховується в OrderService без окремого ендпоінта, Це внутрішня логіка, яка не потребує прямого доступу ззовні. Розрахунок відбувається на основі простих, але ефективних евристик, що залежать від типу та складу замовлення. Схеми API-ендпоінтів можна побачити на рисунках 3.5 та 3.6.

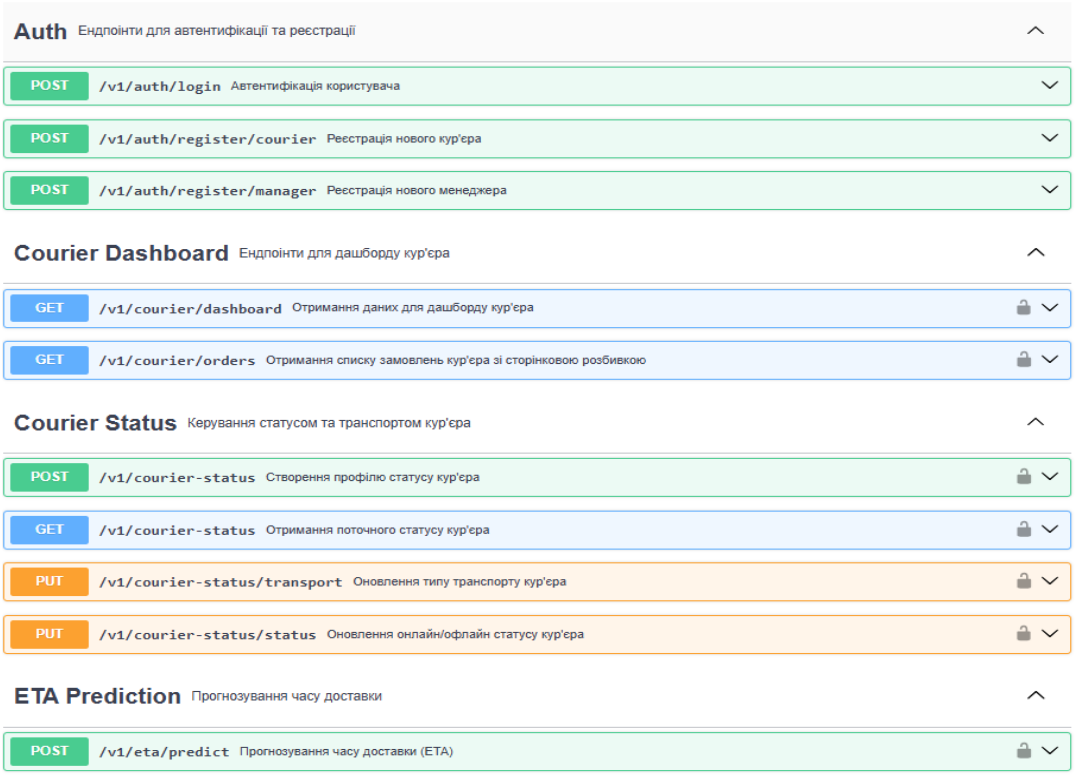

Рисунок 3.5 – Схема API-ендпоінтів

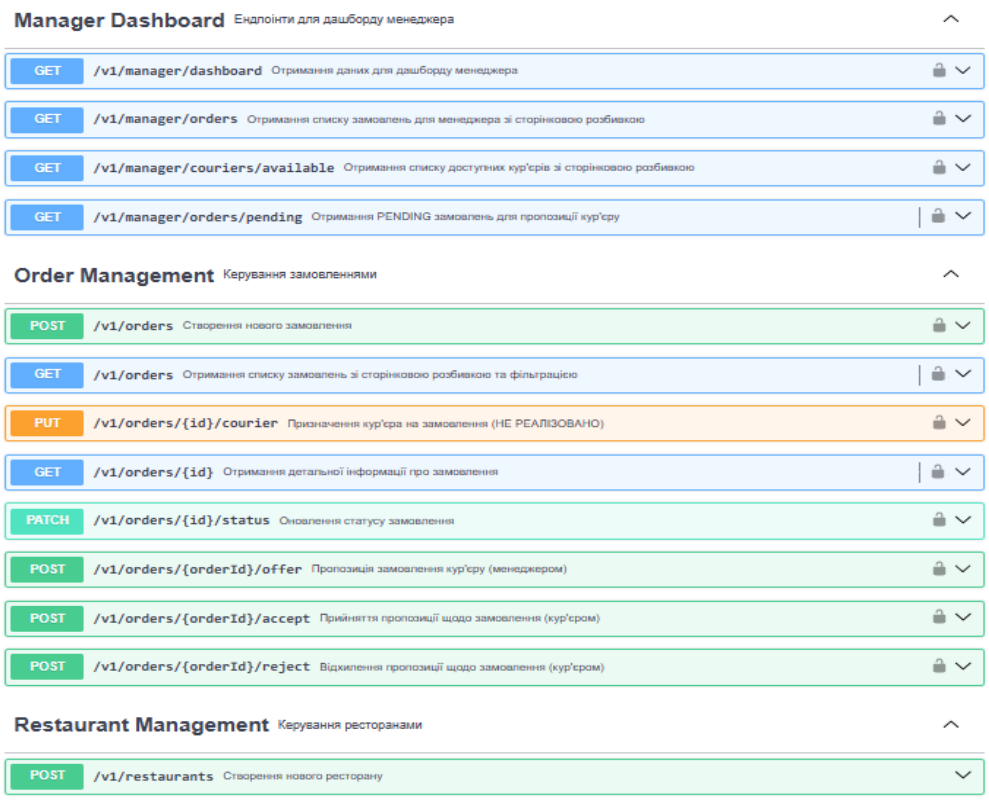

Рисунок 3.6 – Схема API-ендпоінтів (2 частина)

3.3.3 Реалізація бізнес логіки та безпеки застосунку

Бізнес логіка, яка реалізована в сервісному шарі, обробляє всі операції в системі, від створення замовлення до призначення кур'єра.

OrderService керує замовленнями. Метод createOrder валідує вчихідні дані, створює об'єкт Order, додає позиції через addOrderItem, розраховує totalAmount і зберігає в базі даних через orderRepository. Анотація @Transactional забезпечує цілісність: якщо збереження позиції провалиться, то все відкотиться. Метод getManagerOrdersPaginated використовує JPA Specifications для фільтрації, наприклад, за статусом PENDING, що дуже зручно для менеджерів із десятками замовлень щогодини.

CourierStatusService оновлює статуси кур'єрів, транспорт, а також повертає список доступних кур'єрів. Наприклад, метод getAvailableCouriersPaginated використовує Pageable для пагінації, що оптимізує запит при великій кількості кур'єрів.

UserService реалізує UserDetailsService для Spring Security, завантажуючи користувачів за email. Метод loadUserByUsername повертає UserDetails із роллю, що дозволяє перевіряти права. Реєстрація через createNewCourier хешує пароль і присвоює роль ROLE_COURIER.

Безпека застосунку реалізована через Spring Security і JWT, забезпечуючи захист від несанкціонованого доступу. Клас Security Config налаштовує правила доступу. CSRF відключено, оскільки JWT робить систему stateless. CORS дозволяє запити з фронтенду (http://localhost:3000), підтримуючи методи GET, POST, PUT. Правила авторизації визначають, що /v1/auth публічний, /v1/manager доступний для ROLE_MANAGER, а /v1/orders вимагає автентифікації.

JwtTokenProvider генерує і валідує токени, використовуючи HS256 і секретний ключ. Токен містить email, час видачі та термін дії. Ролі не включаються в токен, а завантажуються з бази через UserService, що

забезпечує актуальність прав. JwtAuthenticationFilter перехоплює запити, перевіряє токен у заголовку Authorization, валідує його та встановлює UserDetails у SecurityContextHolder. Якщо токен недійсний, то повертається 401 помилка.

3.4 Реалізація клієнтської частини

3.4.1 Загальна структура frontend частини додатку

Фронтенд було розроблено у вигляді односторінкового додатку на React, що забезпечує швидку навігацію без перевантаження сторінок. SPA (Single Page Application) підхід дозволяє оптимізувати досвід користувача, адже дає змогу швидко перемикатися між розділами. Це є дуже актуальним, особливо для кур'єрів, які працюють у реальному часі.

Структура проекту організована за функціональним принципом, що спрощує розробку та підтримку. Папка src/pages містить компоненти для сторінок, кожна з яких відповідає окремому екрану Наприклад, LoginPage.js реалізує вхід, а ManagerDashboardPage.js – головну сторінку менеджера. Це дозволяє чітко розділити функціонал, хоча й вимагає більше сил для організації маршрутів.

Папка src/components містить перевикористовувані UI-елементи, як AppLayout.js для макету сторінки з хедером та слайдером, або OrderTable.js для макету таблиць замовлень. Такі компоненти зменшують дублювання в коді та прискорюють розробку, подібно до шаблонів у конструкторі. Також такій підхід спрощує підтримку коду, адже замість того, щоб змінювати елемент на кожній сторінці де він використовувався, можна внести зміни лише до одного файлу.

Папка src/services включає в себе apiClient.js, де налаштовано Axios для HTTP-запитів, і api.js із функціями для виклику ендпоінтів.

Кореневий компонент App.js налаштовує маршрути через React Router. Він визначає шляхи, як /login чи /manager/dashboard, і пов’язує їх з компонентами. Використання BrowserRouter забезпечує чисті URL, що виглядає професійно. Компонент App.js наведен нижче в лістингу 3.1 та структура фронтенд додатку показана на рисунку 3.7.

Лістинг 3.1 – Вигляд компоненту App.js

```
function App() {
return (
<Router>
<Routes>
<Route path="/login" element={<LoginPage />} />
<Route path="/register/courier"
element={<RegisterCourierPage />} />
<Route path="/register/manager"
element={<RegisterManagerPage />} />
<Route path="/courier/dashboard"
element={<CourierDashboardPage />} />
<Route path="/courier/delivery/:orderId"
element={<CourierDeliveryPage />} />
<Route path="/courier/deliveries"
element={<CourierOrdersListPage />} />
<Route path="/manager/dashboard"
element={<ManagerDashboardPage />} />
<Route path="/manager/orders/create"
element={<CreateOrderPage />} />
<Route path="/manager/orders"
element={<ManagerOrdersListPage />} />
<Route path="/manager/couriers"
element={<ManagerCouriersListPage />} />
<Route path="/orders/:orderId" element={<OrderDetailPage
/>} />
</Routes>
</Router>
```

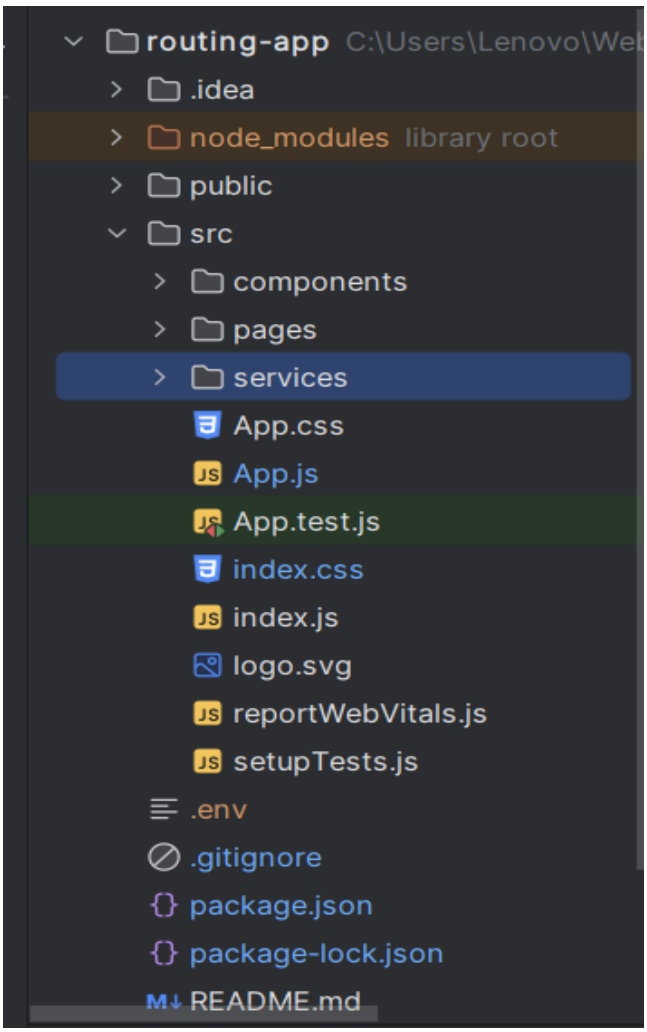

Рисунок 3.7 – Структура фронтенд додатку

3.4.2 Опис ключових компонентів та сторінок

Сторінка входу LoginPage.js дозволяє автентифікуватися, приймаючи email і пароль через форму. MUI-компоненти, як TextField і Button, створюють чистий дизайн. Після успішного входу токен зберігається в localStorage і користувач перенаправляється на головну сторінку. Сторінки реєстрації мають поля для імені, телефону та, для менеджера, коду ресторану.

Головна сторінка менеджеру ManagerDashboardPage.js – являє собою своєрідний центр управляння. Він відображає KPI (кількість замовлень, активні кур’єри), таблиці останніх замовлень OrderTable.js, кур’єрів

CourierTable.js і повідомлення. Кнопка «Create order» веде до CreateOrderPage.js, а «Offer order» відкриває модальне вікно OfferOrderModal.js. Дані завантажуються через getManagerDashboard із api.js, а MUI-компоненти, як Grid чи Paper забезпечують адаптивний дизайн. Головну сторінку менеджера можна побачити на рисунку 3.9.

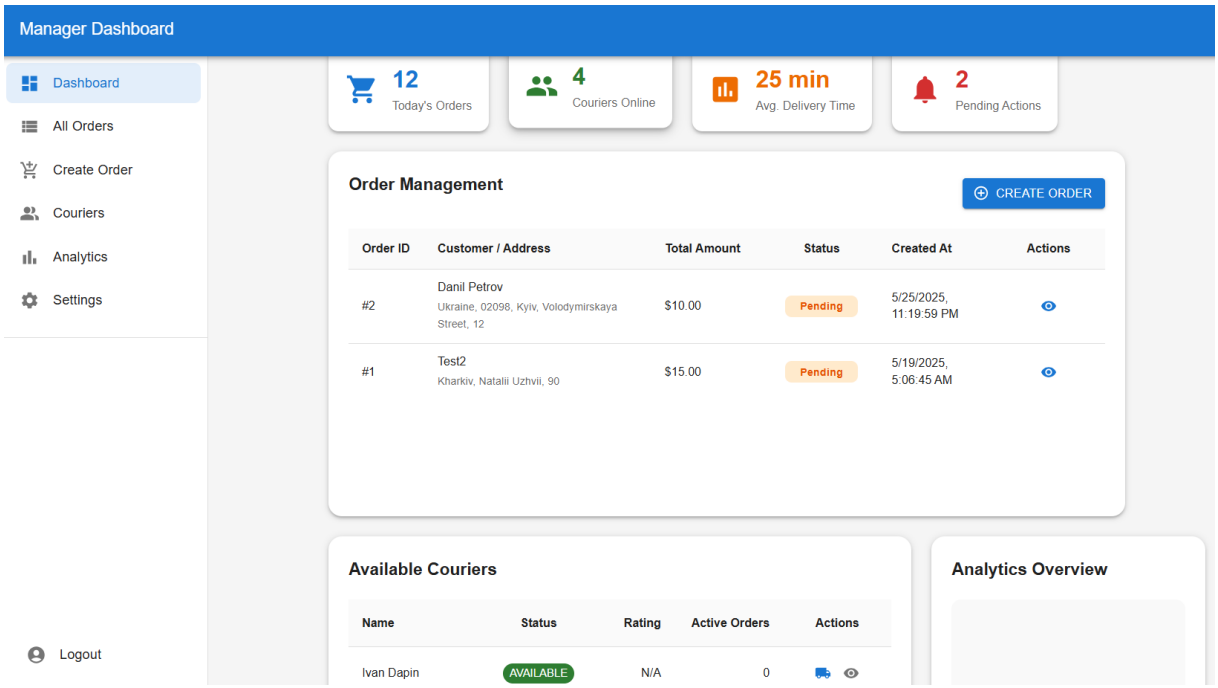

Рисунок 3.9 – Головна сторінка менеджеру

Головна сторінка кур’єра є його основним робочим інструментом, розробленим на React для забезпечення інтерактивності та швидкого оновлення даних у реальному часі. Ця сторінка спроєктована так, щоб надавати всю ключову інформацію в зручному та лаконічному вигляді. Таблиця OrderTable.js відображає запропоновані замовлення з кнопками «Accept/Reject». Статистика заробітку за тиждень реалізована через InfoBlock.js. Дані беруться через getCourierDashboard. Головна сторінка кур’єра показана на рисунку 3.10.

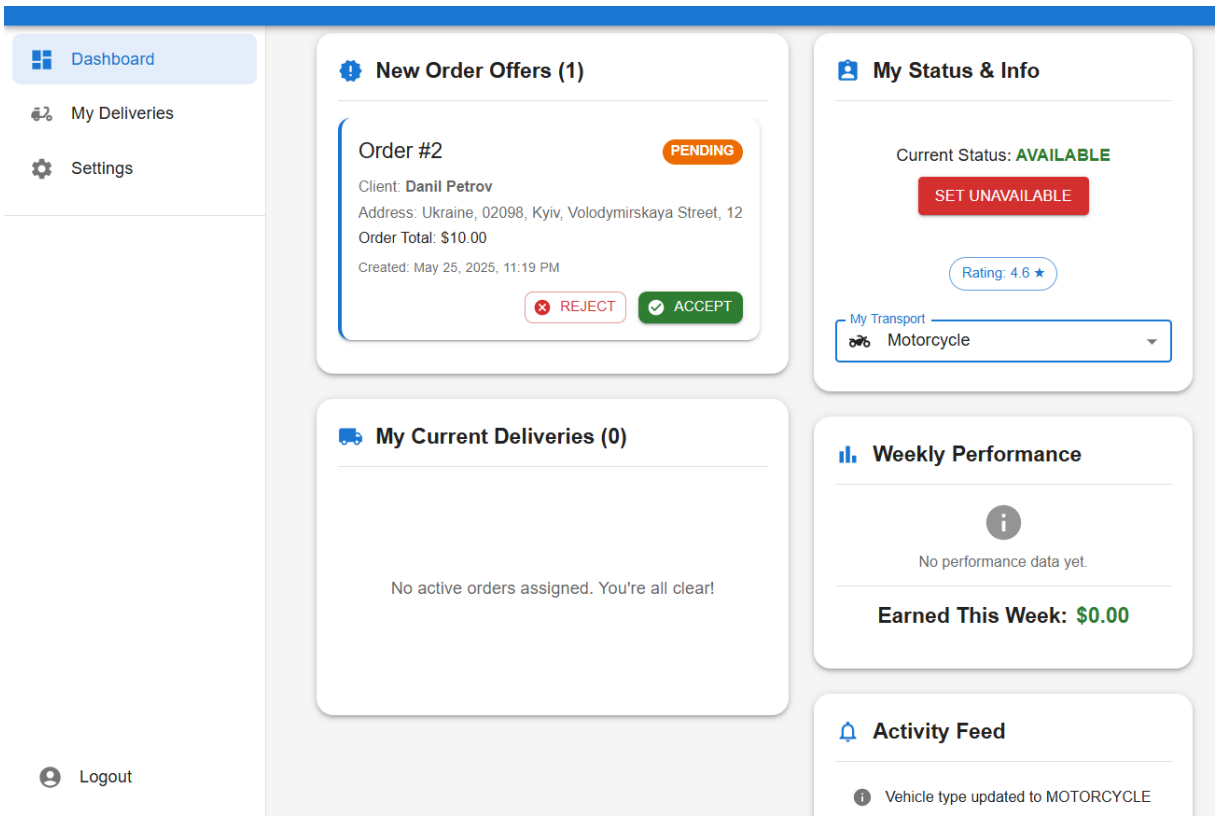

Рисунок 3.10 – Головна сторінка кур’єра

Сторінка створення замовлення – це форма для менеджера, де вводяться дані клієнта, адреса, тип замовлення і позиції. Позиції додаються динамічно, а загальна сума обчислюється одразу після додавання позиції. Запит на створення надсилається через CreateNewOrder.

Сторінка списку замовлень показує пагіновану таблицю OrderTable.js із пошуками і фільтрами. Пагінація через MUI TablePagination дозволяє переглядати 10 замовлень за один раз. Дані беруться через getManagerOrders, а перехід до деталей замовлення (OrderDetailPage) відбувається за кліком на відповідну іконку.

Сторінка доступних кур’єрів відображає CouriersTable.js із пагінацією і пошуком. Кнопка «Offer» відкриває модальне вікно OfferOrderModal.js, де менеджер обирає замовлення і пропонує його вільному кур’єру через offerOrderToCourier.

Сторінка деталей замовлення показує ID, статус, клієнта, адресу, позиції, суму і час приготування. MUI-компоненти, як Table і Chip, форматують дані. Частина коду сторінки Order Detail Page представлена у лістингу 3.2, а її дизайн на рисунку 3.11.

Лістинг 3.2 – Частина коду Order Detail Page

```
const OrderDetailPageContent = () => {
    const { orderId } = useParams();
    const navigate = useNavigate();
    const theme = useTheme();
    const [orderDetails, setOrderDetails] =
useState(null);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);
    const fetchOrderDetails = useCallback(async () => {
        if (!orderId) {
            setError("Order ID is missing from URL.");
            setLoading(false);
            return;
        }
        setLoading(true);
        setError(null);
        try {
            const data = await getOrderDetails(orderId);
            console.log('OrderDetailPage: Fetched Order
Details:', data);
            if (data) {
                setOrderDetails({
                    ...data,
                    orderItems: data.orderItems || [],
                    totalAmount: data.totalAmount != null
? parseFloat(data.totalAmount) : null
```

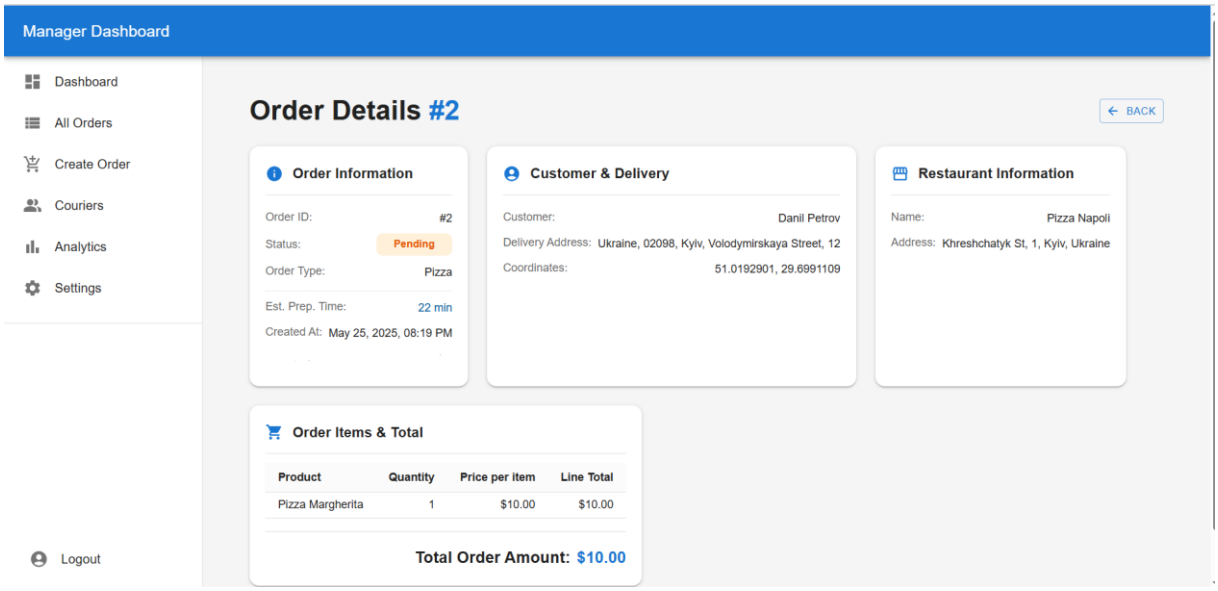

Рисунок 3.11 – Сторінка Order Detail

Сторінка доставки кур’єра має двоколонковий макет: ліворуч – список замовлень, праворуч – деталі і карта Google Maps. Кнопки «Start delivery» чи «Finish delivery» оновлюють статус замовлення через updateOrderStatus. Карта показує маршрут від ресторану до клієнта. Сторінка доставки кур’єра показана на рисунку 3.12.

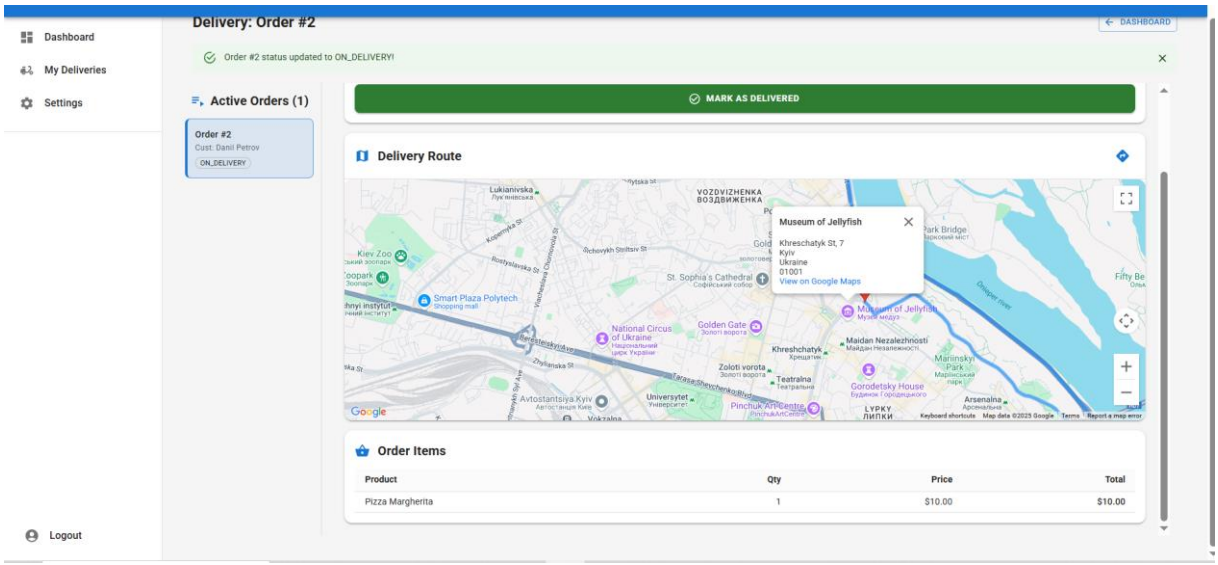

Рисунок 3.12 – Сторінка доставки кур’єра

3.4.3 Взаємодія з API бекенду та інтеграція з Google Maps

Взаємодія з бекендом реалізована через Axios у сервісному шарі. Файл apiClient.js налаштовує Axios із базовим URL http://localhost:8080/v1 і додає JWT-токен до заголовка Authorization. Токен береться з localStorage, що забезпечує автентифікацію. Код для apiClient.js наведений у лістингу 3.3.

Лістинг 3.3 – Код apiClient.js

```
import axios from 'axios';

const apiClient = axios.create({
    baseURL: 'http://localhost:8080/v1',
});

apiClient.interceptors.request.use(
    (config) => {
        const token = localStorage.getItem('token');
        if (token) {
            config.headers.Authorization = `Bearer
${token}`;
        }
        return config;
    },
    (error) => Promise.reject(error)
);
apiClient.interceptors.response.use(
    (response) => response,
    (error) => {
        console.error('API Error:', error.response ||
error.message);
        return Promise.reject(error);
    }
);
export default apiClient;
```

Файл api.js містить функції для ендпоінтів, такі як getOrderDetails чи createNewOrder. Вони формують запити, обробляють відповіді та логують помилки. Наприклад, getManagerOrders підтримує пагінацію через параметри page, size, а обробка помилок дозволяє показувати Alert у MUI.

Інтеграція з Google Maps досягається завдяки використанню бібліотеки @react-google-maps/api. Хук useJsApiLoader завантажує Google Maps API з ключем із .env (REACT_APP_MAPS_API_KEY). Код CourierDeliveryPage.js наведено у лістингу 3.4.

Лістинг 3.4 – Фрагмент коду CourierDeliveryPage.js

```
function CourierDeliveryPage({ orderId }) {
    const calculateRoute = useCallback(() => {
        if (!isLoaded || !restaurantCoords || 
!deliveryCoords) return;
        const directionsService = new 
window.google.maps.DirectionsService();
        directionsService.route(
            {
                origin: restaurantCoords,
                destination: deliveryCoords,
                travelMode: 'DRIVING',
            },
            (result, status) => {
                if (status === 'OK') {
                    setDirectionsResponse(result);
                } else {
                    console.error('Directions API Error:', 
status);
                }
            }
        );
    }, [isLoaded, restaurantCoords, deliveryCoords]);
}
```

Компонент GoogleMap відображає карту, MarkerF позначає ресторан і адресу доставки, а DirectionsRender будує маршрут. Координати беруться з orderDetails, мемоізовані через useMemo. Автоматичне масштабування (fitBounds) забезпечує повне відображення маршруту.

3.5 Розробка та інтеграція моделі інтелектуальної маршрутизації

Інтелектуальна маршрутизація є невід'ємною частиною системи доставки, адже вона оптимізує логістичні процеси, підвищує ефективність і покращує досвід користувачів. У цьому розділі буде описано процес розробки моделі прогнозування часу доставки, інтеграцію в систему та аналіз ефективності.

3.5.1 Постановка задачі моделі і обрання датасету

Задача полягає в прогнозуванні часу доставки, тобто тривалості від моменту, коли кур'єр забирає замовлення в ресторані, до його вручення клієнту. Цільова змінна, Time_taken(min), вимірюється в хвилинах і є числовою, що робить задачу регресією. Точний прогноз дозволяє менеджерам ефективно розподіляти замовлення, кур'єрам – планувати маршрути, а клієнтам – знати, коли чекати їжу.

Для навчання моделі було використано датасет Food Delivery Dataset із Kaggle, що містить понад 45 тисяч записів про доставки. У ньому зафіксовано деталі кожної поїздки. Ці дані включають в себе такі важливі ідентифікатори як замовлення та кур'єра, вік і рейтинг кур'єра, географічні координати ресторану і клієнта, точний час розміщення замовлення, погодні умови, щільність трафіку, тип замовлення, транспорт, а також фактори типу фестивалі чи тип міста.

3.5.2 Попередня обробка даних та інженерія ознак

Якість прогнозування будь-якої моделі, в першу чергу, залежить від якості даних. Саме тому багато уваги та значну кількість зусиль було приділено їх підготовці. Вихідний набір даних містить числові і категоріальні ознаки, а також пропуски, що вимагало ретельної обробки. Обробка даних включала в себе очищення, створення нових ознак і масштабування числових ознак.

Очищення почалося з обробки пропусків, оскільки більшість алгоритмів машинного навчання не можуть ефективно працювати з порожніми даними. Для числових ознак, як Delivery_person_Age чи Delivery_person_Ratings, які мали пропущенні значення. було обрано стратегію заповнення їх медіаною. Медіана, що є значенням, яке знаходиться посередині впорядкованого ряду всіх значень ознаки, розрахувалася на основі тренувальної частини даних. Перевага медіани над середнім арифметичним полягає в її меншій чутливості до викидів, що дозволяє краще зберегти початковий розподіл цієї ознаки. Для категоріальних ознак, як Wheather_conditions чи Road_traffic_density, пропуски заповнювались модою (найчастішим значенням), наприклад, «Sunny» для погоди. Особливу увагу було приділено саме ознакам, що фіксують час: Time_Ordered та Time_Ordered_Picked. Оскільки ці дані є критично важливими для подальших розрахунків, було прийнято рішення видаляти ті рядки, де ці значення відсутні або некоректні.

Крім того, якщо в категоріальних ознаках зустрічалися текстові позначення пропусків, вони замінювались на спеціальну категорію «Unknown», що дозволяє моделі враховувати такі випадки як окрему категорію. Такий підхід є більш досконалим, ніж просте заповнення модою, оскільки він дає моделі можливість самостійно визначити, чи є сам факт відсутності даних значущою ознакою. Код для обробки пропусків наведено у лістингу 3.5.

Лістинг 3.5 – Код для обробки пропусків

```
col = 'Delivery_person_Age'
train_df[col] = pd.to_numeric(train_df[col],
errors='coerce')
median_val = train_df[col].median()
train_df[col].fillna(median_val, inplace=True)

col = 'Weatherconditions'
mode_val = train_df[col].mode()[0]
train_df[col].fillna(mode_val, inplace=True)

col_time = 'Time_Orderd'
train_df[col_time] =
train_df[col_time].astype(str).str.strip().replace('NaN',
np.nan)
train_df.dropna(subset=[col_time], inplace=True)
```

Після обробки пропусків наступним кроком була стандартизація та приведення даних до єдиного формату. Категоріальні дані пройшли процес стандартизації: наприклад, з ознаки Wheatherconditions було видалено зайве слово «conditions», залишаючи тільки суть погодних умов. Для ознаки Road_traffic_density було забезпечено уніфіковане представлення категорій, наприклад, Jam або Low, шляхом видалення зайвих пробілів. Деякі числові ознаки, як multiple_ deliveries, спочатку були представлені у вигляді тексту і були конвертовані в числовий формат. Стовпець з датою замовлення Order_Date був переведений у спеціальний формат datetime, що дозволяє легко витягувати з нього компоненти дати. Стовпці з часом замовлення Time_Ordered та часу отримання замовлення кур’єром Time_Ordered_picked були оброблені для виокремлення годин та хвилин, що потім використовувалися для створення нових, більш деталізованих числових ознак. Цільова змінна Time_taken(min) також була очищена від текстового суфіксу «(min)» та конвертована в числовий формат, що є необхідним для

задачі регресії. Код для стандартизації та конвертації даних наведений у лістингу 3.6.

Лістинг 3.6 – Код для стандартизації та конвертації даних

```
    if 'conditions' in col_weather:
    df[col_weather] =
df[col_weather].str.replace(r'\\s*conditions\\s*', '',
regex=True).str.strip()

    train_df['multiple_deliveries'] =
pd.to_numeric(train_df['multiple_deliveries'],
errors='coerce')

    df['Order_Date'] = pd.to_datetime(df['Order_Date'],
format='%d-%m-%Y', errors='coerce')

    time_col = 'Time_Orderd'
    time_parts = df[time_col].str.split(':', expand=True)
    hours = pd.to_numeric(time_parts[0],
errors='coerce').fillna(0)
    minutes = pd.to_numeric(time_parts[1],
errors='coerce').fillna(0)
    df[time_col + '_minutes'] = hours * 60 + minutes

    y_val = y_val.astype(str)
    y_val = y_val.str.replace(r'\\(min\\)', '', regex=True)
    y_val = pd.to_numeric(y_val, errors='coerce')
```

Етап інженерії ознак полягав у створенні нових, потенційно більш інформативних ознак на основі вже наявних даних, що може суттєво підвищити якість моделі. Так, було розраховано відстань (distance_km) між рестораном та адресою клієнта за допомогою формули Гаверсинуса. Ця формула дозволяє обчислити відстань між двома точками на сфері за їх географічними координатами, і ця відстань є одним з основних факторів, що

прямо впливає на час доставки. Було також створено низку часових ознак: Time_Ordered_minutes та Time_Order_picked_minutes (час розміщення замовлення та час його отримання кур’єром, переведені у загальну кількість хвилин від початку доби), prep_time_minutes (час на підготовку замовлення рестораном, розрахований як різниця між попередніми двома), order_day_of_week та is_weekend. Ці ознаки дозволяють моделі враховувати добові та тижневі цикли активності, наприклад, пікові години завантаженості як ресторанів, так і дорожньої мережі. Після створення цих нових, більш змістовних ознак, оригінальні стовпці з координатами та часом були видалені. Це робиться для уникнення мультиколінеарності – ситуації, коли ознаки сильно корелюють між собою, що може негативно вплинути на стабільність моделі. Код для інженерії ознак наведений у лістингу 3.7.

Лістинг 3.7 – Код для інженерії ознак

```
    train_df['distance_km'] =
haversine(train_df['Restaurant_latitude'],
train_df['Restaurant_longitude'],
    train_df['Delivery_location_latitude'],
train_df['Delivery_location_longitude'])
    df_time['order_hour'] = (df_time['Time_Orderd_minutes'] //
60).astype(int)
    df_time['prep_time_minutes'] =
df_time['Time_Order_picked_minutes'] -
df_time['Time_Orderd_minutes']
    df_time['order_day_of_week'] =
df_time['Order_Date'].dt.dayofweek
    df_time['is_weekend'] =
df_time['order_day_of_week'].apply(lambda x: 1 if x >= 5 else
0)
    cols_to_drop = ['ID', ..., 'Restaurant_latitude', ...,
'Time_Orderd_minutes', ...]
    train_df_processed = train_df.drop(columns=[col for col in
cols_to_drop if col in train_df.columns])
```

Фінальним етапом підготовки даних було кодування категоріальних ознак та масштабування числових. Категоріальні ознаки, такі як погодні умови або тип міста, були перетворені на числовий формат за допомогою метода One-Hot Encoding. Цей метод створює для кожної унікальної категорії в ознаці новий бінарний стовпець, що містить значення 0 або 1. Це є необхідним кроком, оскільки більшість моделей, включаючи XGBoost, не можуть безпосередньо працювати з текстовими даними. Числові ознаки, в свою чергу, пройшли процедуру масштабування за допомогою StandardScaler. Цей метод перетворює дані таким чином, щоб кожна числова ознака мала середнє значення 0 і стандартне відхилення 1. Таке перетворення допомагає збалансувати вплив різних ознак, особливо якщо вони мають суттєво відмінні діапазони значень (наприклад, вік кур'єра порівняно з відстанню), і може покращити процес навчання моделі. Код для кодування та масштабування ознак у лістингу 3.8.

Лістинг 3.8 – Код для кодування та масштабування ознак

```
    categorical_features = X.select_dtypes(include=['object',
'category']).columns.tolist()
    numerical_features =
X.select_dtypes(include=np.number).columns.tolist()

    numerical_transformer = Pipeline(steps=[
        ('imputer', SimpleImputer(strategy='median')),
        ('scaler', StandardScaler())
     ])

    categorical_transformer = Pipeline(steps=[
        ('imputer', SimpleImputer(strategy='most_frequent')),
        ('onehot', OneHotEncoder(handle_unknown='ignore',
sparse_output=False))
     ])
```

3.5.3 Вибір, навчання, та валідація моделі прогнозування часу доставки

Для вирішення задачі регресійного аналізу, а саме прогнозування часу доставки, було обрано алгоритм XGBoost, який зарекомендував себе як ефективний інструмент для роботи зі структурованими даними. Цей вибір обґрунтований високою точністю XGBoost, що досягається завдяки реалізації концепції градієнтного бустингу – послідовного будівництва ансамблю дерев рішень. Важливими перевагами алгоритму є також вбудовані механізми регуляризації (L1 та L2) для запобігання перенавчанню, висока швидкість обчислень та можливість оцінки важливості ознак. У порівнянні з іншими розглянутими альтернативами, такими як Random Forest, XGBoost продемонстрував кращу точність та більшу стійкість до перенавчання. Простіші моделі, наприклад, лінійна регресія, не розглядалися як основні через нездатність адекватно врахувати нелінійні залежності в даних, тоді як складніші, як нейронні мережі, вимагали б значно більших обсягів даних та обчислювальних ресурсів. Таким чином, XGBoost став оптимальним вибором, що забезпечує баланс між простотою реалізації, обчислювальною ефективністю та високою якістю прогнозування.

Процес навчання та валідації моделі є ключовим для забезпечення її надійності та здатності до узагальнення нових, раніше не бачених даних. На першому етапі весь доступний набір даних було розділено на дві частини: тренувальну частину, що склала 80% від загального обсягу, та валідаційну вибірку, на яку припало решта 20%. Тренувальна вибірка, X_train для ознак та Y_train для цільової змінної, використовувалася безпосередньо для навчання моделі, тобто для побудови дерев рішень та оптимізації їх параметрів. Валідаційна вибірка залишалася недоторканою під час навчання і слугувала для незалежної оцінки якості навчання моделі.

Для структурування процесу попередньої обробки даних та навчання було використано механізм конвеєрів (Pipeline) з бібліотеки scikit-learn. Створений пайплайн послідовно виконував наступні кроки: стандартизацію числових ознак та перетворення категоріальних ознак у числовий формат. Після цих перетворень, підготовлені дані подавалися на вхід регресора XGBoost. Такий підхід забезпечує коректне та послідовне застосування всіх етапів обробки як до тренувальних, так і до валідаційних даних, запобігаючи витоку даних з валідаційної вибірки на етапі навчання.

Підбір оптимальних гіперпараметрів є критично важливими етапом для досягнення високої точності прогнозування та забезпечення моделі до узагальнення. Початкове навчання моделі проводилося на тренувальному наборі з використанням стандартних значень гіперпараметрів, запропонованих бібліотекою XGBoost (наприклад, швидкість навчання learning_rate=0.1 або 0.3, максимальна глибина дерев max_depth=6). Зміна гіперпараметрів суттєво впливає на поведінку моделі, наприклад, попередні експерименти зі збільшенням максимальної глибини дерев до 10 показували ознаки перенавчання: коефіцієнт детермінації R^2на тренувальній вибірці сягав високих значень близько 0.95, в той час як на валідаційній вибірці він був значно нижчим близько 0.8. Це вказувало на те, що модель ставала надто складною, запам’ятовуючи особливості тренувальних даних, включаючи шум і втрачала здатність ефективно узагальнювати на нових, раніше не бачених даних.

Для більш систематичного та об’єктивного підбору оптимальних гіперпараметрів було застосовано метод пошуку по сітці GridSearchCV. В рамках цього пошуку тестувалися різні комбінації максимальної глибини дерева max_depth зі значенням (3,5,6,7,8) та кількості дерев в ансамблі n_estimators (50,100,150,200) при фіксованій швидкості навчання learning_rate=0.1. Як цільову метрику для оптимізації було обрано коефіцієнт детермінації R^2. Використання grid_search наведено у лістингу 3.9.

Лістинг 3.9 – Код використання grid_search

```
xgb_model_for_grid = xgb.XGBRegressor(
    objective='reg:squarederror',
    learning_rate=0.1,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    n_jobs=-1
)

full_pipeline_for_grid = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', xgb_model_for_grid)
])

param_grid = {
    'regressor__max_depth': [3, 5, 6, 7, 8],
    'regressor__n_estimators': [50, 100, 150, 200]
}

grid_search = GridSearchCV(
    estimator=full_pipeline_for_grid,
    param_grid=param_grid,
    cv=3,
    scoring='r2',
    verbose=1,
    n_jobs=-1
)
```

Результати GridSearchCV показали, що найкращою комбінацією є max_depth=8 та n_estimators=50. При цих параметрах середній R^2 на крос-валідаційних фолдах тренувальної вибірки склав 0.8363. Візуальний аналіз теплової карти підтвердив ці висновки, наочно продемонструвавши, що область найвищих значень R^2 концентрується при max_depth рівному 7

або 8, та n_estimators в діапазоні від 50 до 100, причому подальше збільшення кількості дерев понад 50 для max_depth=8 не призводило до суттєвого покращення на крос-валідації. Теплова карта показана на рисунку 3.13.

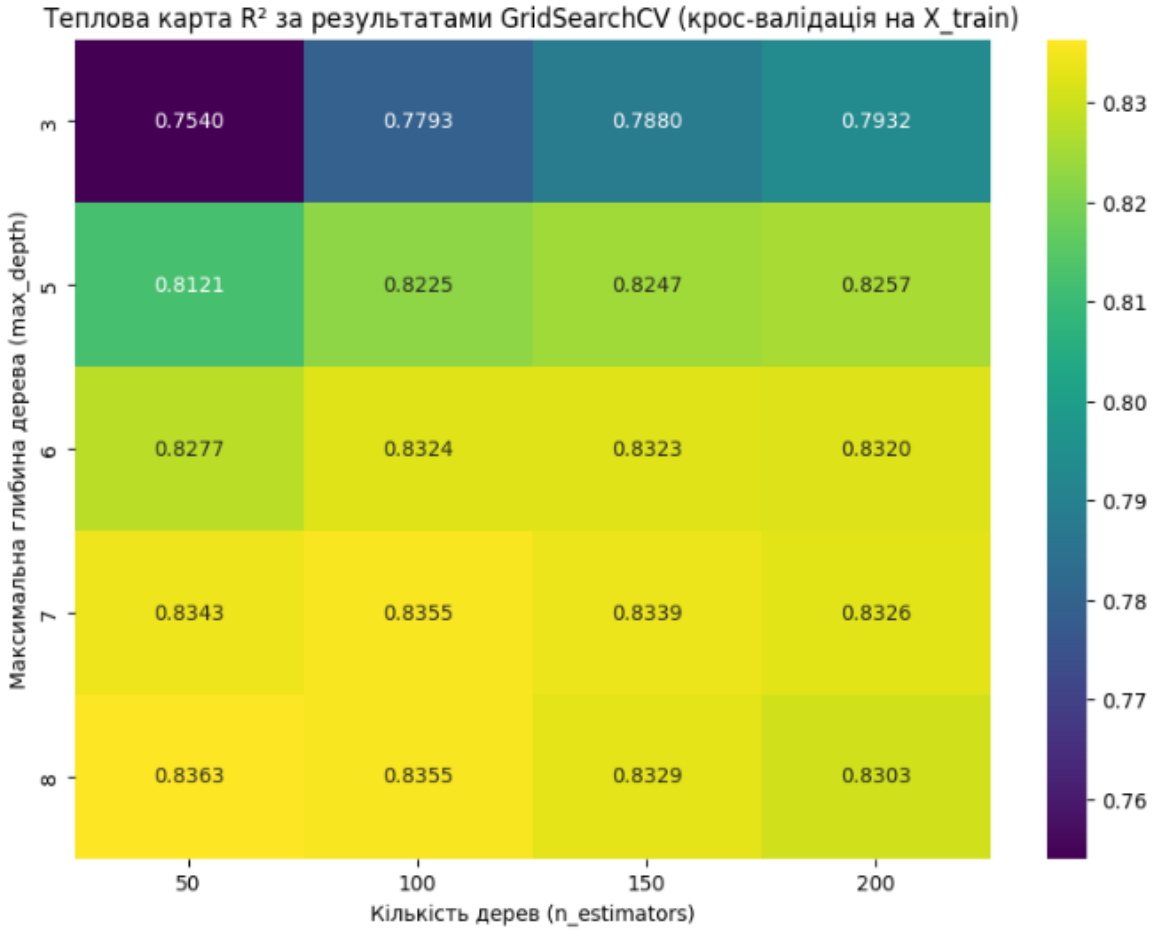

Рисунок 3.13 – Теплова карта

Після визначення оптимальних гіперпараметрів, фінальна модель XGBoost була навчена на всьому тренувальному наборі з використанням саме цієї конфігурації: max_depth=8, n_estimators=50 та learning_rate=0.1. Потім ефективність навченої моделі оцінювалась на валідаційній вибірці. На цьому етапі модель досягла коефіцієнта детермінації R^2 0.8420 та середньої абсолютної помилки (MAE) 2.97 хвилини. Ці показники виявилися кращими ніж у моделі зі стандартними

гіперпараметрами (R^2=0.8280, MAE=3.1 хвилини), що підтверджує ефективність проведеного підбору гіперпараметрів.

3.5.4 Оцінка ефективності та результатів моделі прогнозування часу доставки

Після успішного навчання та оптимізації гіперпараметрів моделі, було проведено оцінку її ефективності Для кількісної оцінки якості прогнозів використовувалися стандартні метрики регресії: середня абсолютна помилка (MAE), середньоквадратична помилка (MSE), корінь із середньоквадратичної помилки (RMSE) та коефіцієнт детермінації (R^2). Оптимізована модель XGBoost досягла наступних показників: MAE склала 2.9757 хвилини, MSE – 13.86, RMSE – 3.72, а коефіцієнт детермінації R^2 – 0.8420.

Для більш об'єктивної оцінки значущості цих результатів було проведено порівняння з простою базовою моделлю, яка для всіх прогнозів використовувала середнє значення часу доставки, розраховане на тренувальній виборці. Базова модель продемонструвала значно нижчу точність: MAE 7.62 хвилини, RMSE 9.366 хвилини та R^2 -0.0001.

Аналіз метрик показує, що оптимізована модель XGBoost є високоефективною. Середня абсолютна помилка близько 2.98 хвилини означає, що прогнози моделі в середньому відхиляються від фактичного часу доставки менш ніж на три хвилини, що є дуже хорошим результатом. Значення RMSE, хоч і дещо вище за MAE через властивість цієї метрики сильніше реагувати на більші помилки, але також демонструє високу точність. Коефіцієнт R^2 на рівні 0.8420 є підтвердженням того, що розроблена модель здатна пояснити близько 84.2% всієї варіативності часу доставки на основі використаних ознак. Це свідчить про глибоке розуміння моделлю факторів, що впливають на час доставки, таких як погодні умови, щільність трафіку чи відстань.

Для ще більш глибокого розуміння поведінки моделі та характеру її помилок було проведено візуальний аналіз. Графік співвідношення фактичних та передбачених значень часу доставки демонструє, що точки прогнозів переважно групуються вздовж діагональної лінії, яка представляє ідеальне співпадіння. Це візуально підтверджує тісну кореляцію між прогнозами моделі та реальними даними. Наявний розкид точок навколо діагоналі ілюструє величину помилок для окремих спостережень. Характерні вертикальні скупчення точок, що спостерігаються на графіку, ймовірно, пов’язані з дискретністю або округленням значень цільової змінної в оригінальному наборі даних. Це не є недоліком самої моделі, а скоріше відображенням специфіки даних. Графік співвідношення фактичних та передбачених значень часу доставки показано на рисунку 3.14.

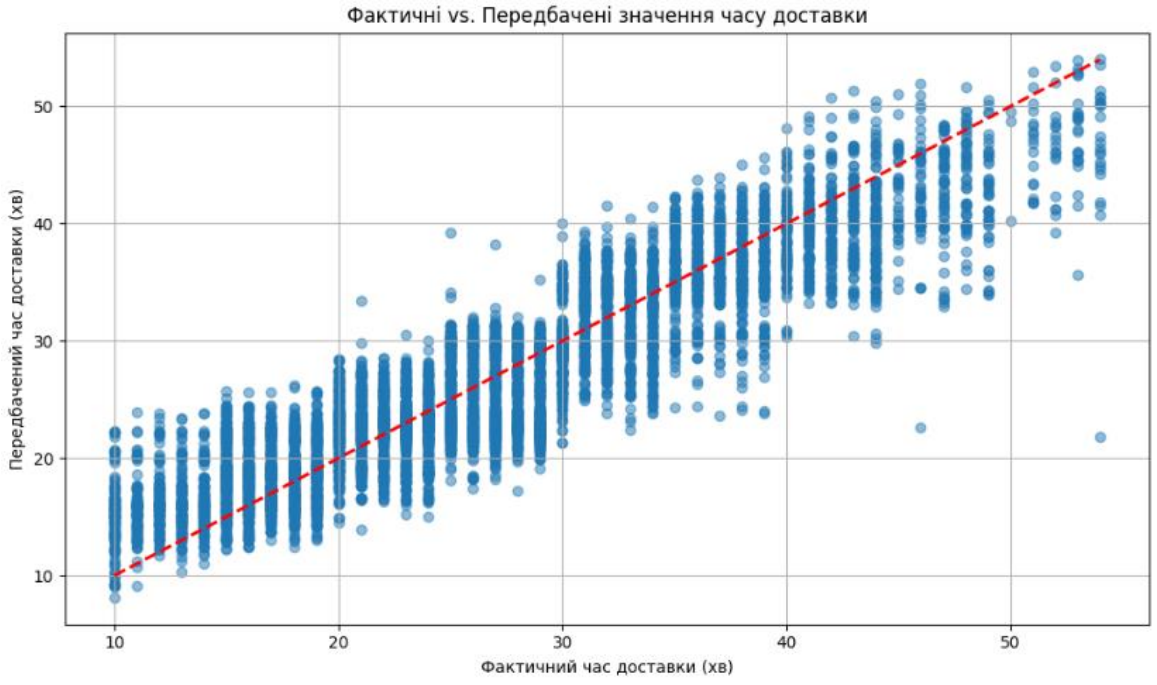

Рисунок 3.14 – Графік співвідношення фактичних та передбачених значень часу доставки

Аналіз графіка залишків, де по осі ординат відкладено різницю між фактичним та передбаченим часом, підтверджує відстустність явних систематичних помилок у моделі, оскільки залишки загалом центровані навколо нульової лінії. Спостережувані кластери та структури в розподілі залишків, як і на попередньому графіку, також можуть бути пояснені дискретністю фактичних значень часу. Важливо, що дисперсія залишків виглядає відносно стабільною по всьому діапазону передбачених значень, тобто помилка не зростає систематично зі збільшенням прогнозованого часу доставки. Графік залишків показано на рисунку 3.15.

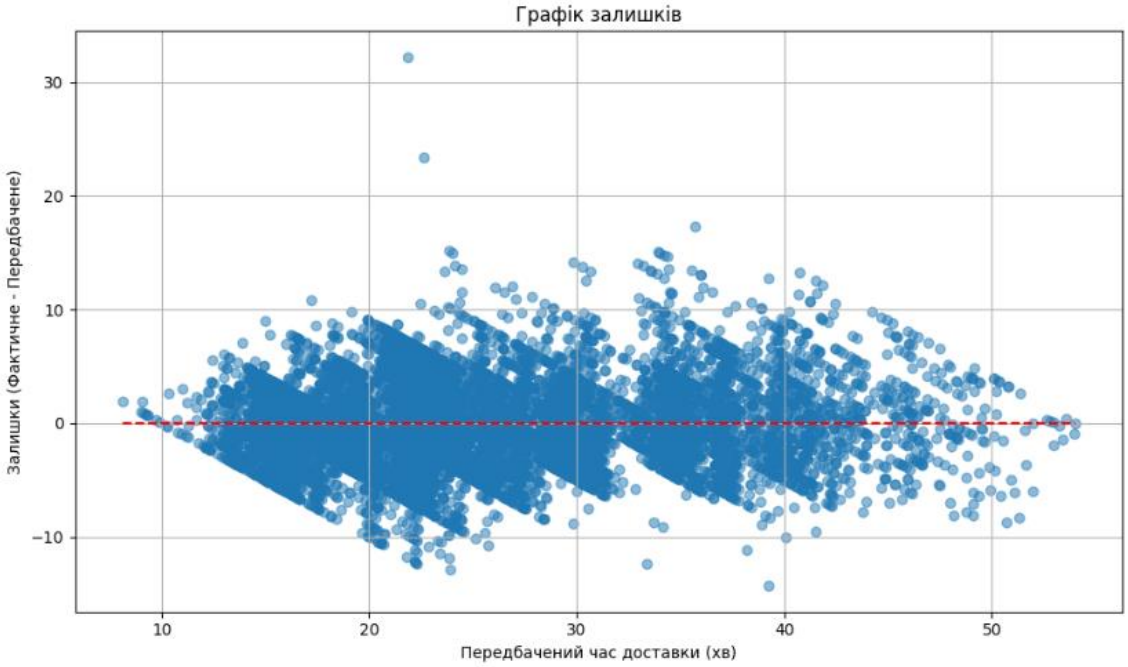

Рисунок 3.15 – Графік залишків

Гістограма розподілу залишків додатково підтверджує ці висновки: вона має чітко виражений пік поблизу нуля, що означає, що більшість помилок моделі є дуже малими. Водночас, спостерігається легка правостороння асиметрія у бік позитивних залишків. Це вказує на те, що хоча великі помилки рідкісні, модель іноді схильна недооцінювати час

доставки, тобто фактичний час виявляється більший за прогнозований. Гістограма розподілу залишків показана на рисунку 3.16.

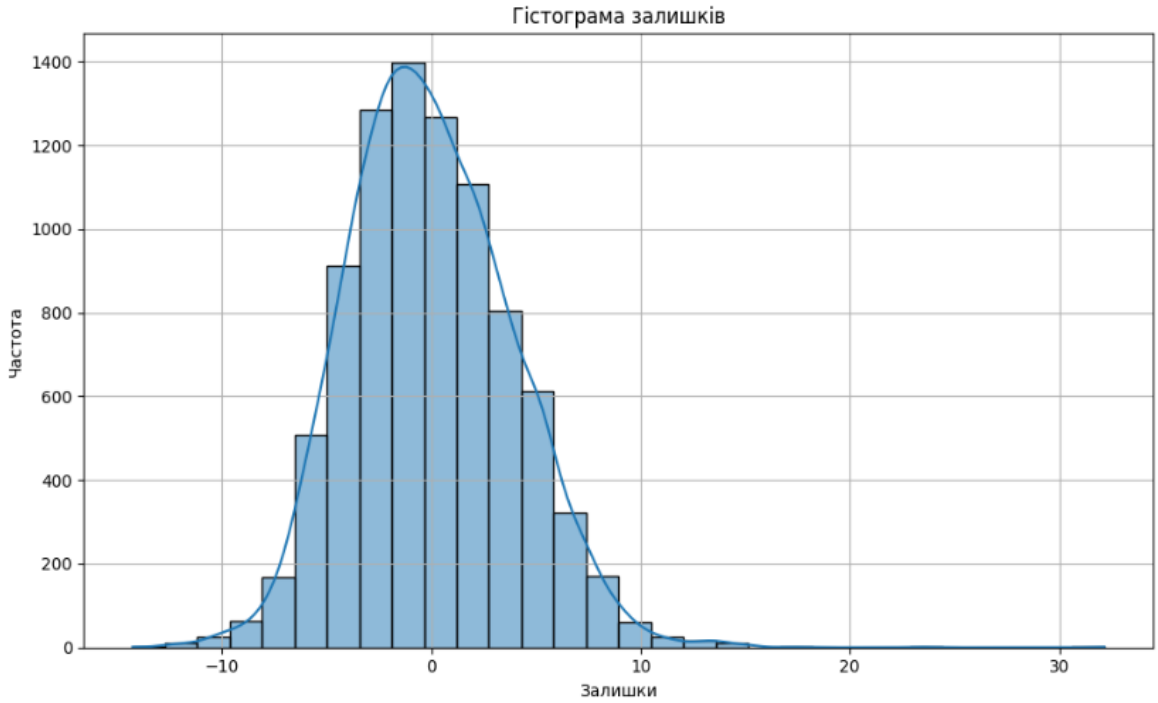

Рисунок 3.16 – Гістограма розподілу залишків

Графік важливості ознак дозволяє визначити ключові фактори, що найбільше впливають на прогнози моделі. Серед найбільш значущих ознак виділяються Road_traffic_density_Jam (наявність заторів), Delivery_person_Ratings (рейтинг кур'єра) та distance_km (відстань доставки). Логічно, що затори суттєво затримують доставку, а рейтинг кур'єра може відображати його досвід, ефективність та знання міста. Відстань є фундаментальним фактором. Також, важливий вплив мають погодні умови, час на підготовку замовлення та інші часові характеристики, що узгоджуються з реальними умовами функціонування служби доставки. Графік важливості ознак можна побачити на рисунку 3.17.

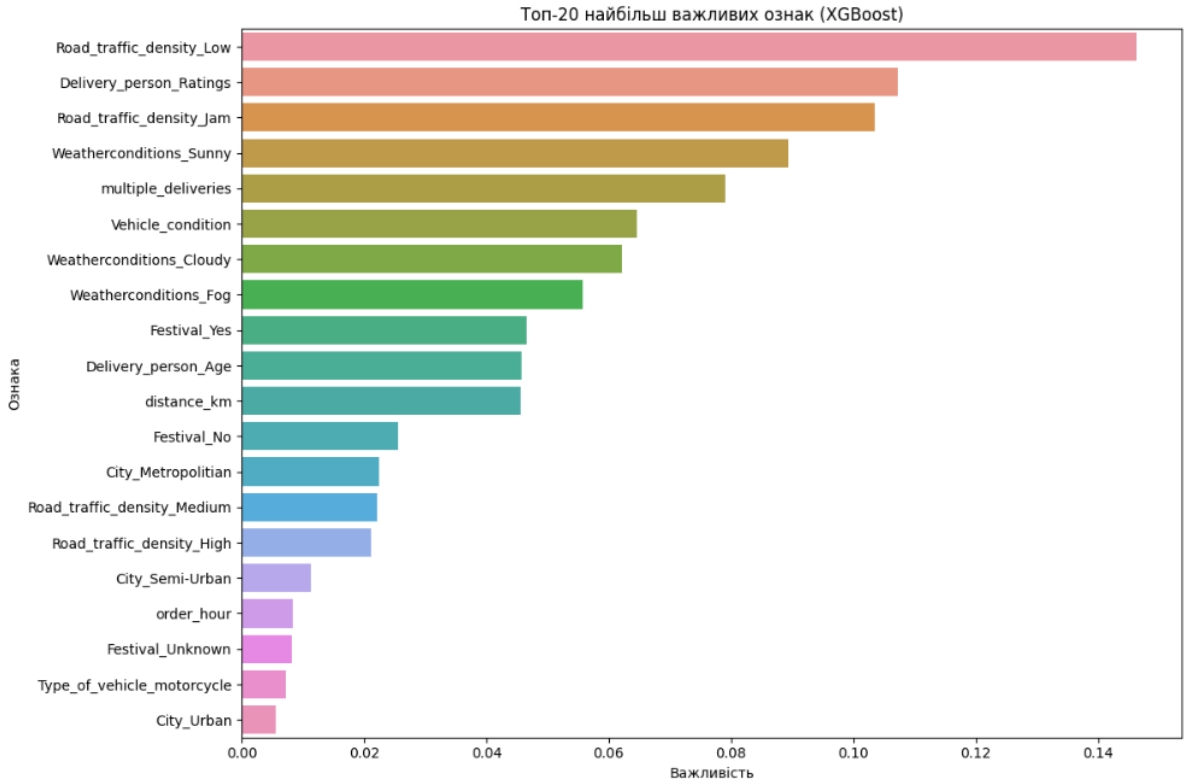

Рисунок 3.17 – Графік важливості ознак

Візуальне порівняння показує, що модель загалом добре відтворює діапазон та основну тенденцію розподілу реальних даних. Це позитивний знак, який свідчить про те, що прогнози не є хаотичними, а їхній центр та розкид близькі до фактичних. Модель успішно вловила загальну закономірність процесу доставки.

Але помітно, що модель дещо рідше прогнозує дуже короткий час доставки (менше 10-12 хвилин) порівняно з їх фактичною частотою. Це може бути пов'язано з кількома факторами. Можливо, в даних недостатньо прикладів таких швидких доставок, через що модель вважає їх аномаліями. Або ж моделі бракує специфічних ознак, які вказують на можливість надшвидкого виконання замовлення (наприклад, кур'єр уже перебуває в закладі). Графік порівняння розподілів фактичних та передбачених значень часу можна побачити на рисунку 3.18.

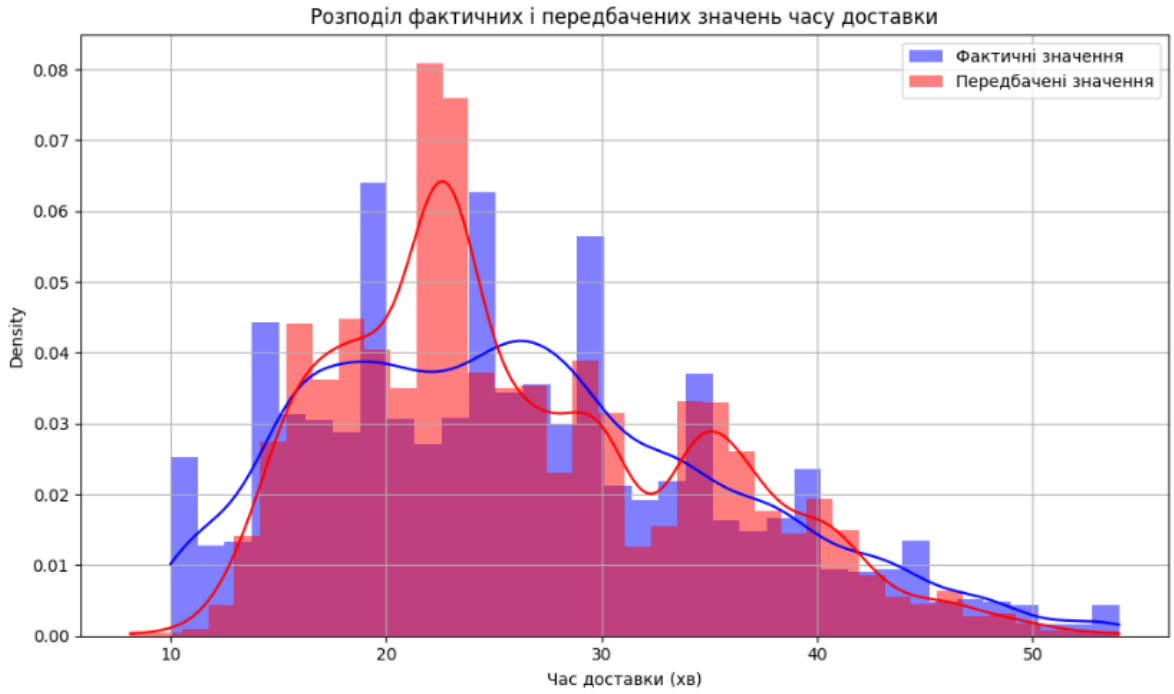

Рисунок 3.18 – Графік порівняння розподілів фактичних та передбачених значень часу

Обмеження розробленої моделі переважно пов’язані зі дискретністю цільової змінної, що впливає на інтерпретацію діагностичних графіків залишків та розподілів. Також виявлена легка схильність до недооцінки часу доставки в рідкісних випадках. Для корекції цього можна розглянути використання специфічних вагових функцій втрат, які сильніше штрафують за недооцінку, проте це може призвести до погіршення інших показників. У промислових системах для подальшого покращення було би доцільно збирати більш точні дані про час доставки, наприклад, з точністю до секунд, а не округлені до хвилини.

Розроблена модель є потужним інструментом для прогнозування часу доставки, що враховує комплекс взаємопов’язаних факторів, а також демонструє високу точність і пояснювальну здатність.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було повністю реалізовано поставлене завдання: розроблено комплексну інформаційну систему для оптимізації логістичних процесів. Створений програмний комплекс автоматизує управління замовленнями, здійснює прогнозування часу доставки та забезпечує інтелектуальну маршрутизацію. Розроблена модель прогнозування очікуваного часу доставки демонструє значні покращення точності порівняно з базовим підходом прогнозування середнього значення. Якісні показники включають зручний інтерфейс, швидку взаємодію через REST API та інтеграцію з Google Maps для побудови оптимальних маршрутів, що дозволяє системі ефективно функціонувати навіть в умовах складної дорожньої обстановки.

Порівняння із наявними рішеннями підтверджує конкурентоспроможність розробленої системи. Вітчизняні системи, такі як Raketa або Menu.ua, а також служби доставки локальних ресторанів, як мережі «Пузата Хата», переважно покладаються на ручне планування або базові евристичні алгоритми, що оцінюють час доставки за середньою відстанню чи історичними даними. Raketa, наприклад, використовує власний додаток для маршрутизації, але відсутність відкритої інформації про застосування машинного навчання свідчить про обмежену точність прогнозування ETA. На відміну від цих систем, а також від складних і дорогих закритих платформ світових аналогів, таких як Uber Eats, розроблена система відрізняється потенційною доступністю для малого й середнього бізнесу та відкритою архітектурою, що дозволяє інтегрувати додаткові сервіси, як CRM-системи. Модель прогнозування ETA, побудована на алгоритмі XGBoost, наближається до комерційних стандартів, але є економічно вигіднішим рішенням для українських компаній.

Перспективи покращення системи включають в себе розширення датасету шляхом збору даних із реальних служб доставки, що підвищить точність прогнозу. Застосування складніших алгоритмів, як нейронні мережі, може додатково зменшити помилку прогнозування. Інтеграція з технологіями реального часу і перехід до мікросервісної архітектури забезпечать масштабування для великих обсягів даних. Ці можливості роблять систему не лише конкурентоспроможною, але й перспективною для розвитку на українському ринку, де попит на ефективну логістику невпинно зростає.

Основним науково-практичним результатом є створення комплексної інформаційної системи, що поєднує серверне ядро, інтерактивний інтерфейс та модель прогнозування ETA. Аналіз важливості ознак моделі XGBoost може слугувати основою для подальших досліджень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) eCommerce - Worldwide | Statista Market Forecast. *Statista*. URL: https://www.statista.com/outlook/emo/ecommerce/worldwide (date of access: 13.05.2025).

2) The Future of E-Commerce in 2025: Trends and Opportunities for Growth – E-Commerce Update. *E-Commerce Update – Um Patrimônio do Comércio Eletrônico*. URL: https://ecommerceupdate.org/en/artigos/o-futuro-do-e-commerce-em-2025-tendencias-e-oportunidades-para-crescer/ (date of access: 13.05.2025).

3) Online retail experience and customer satisfaction: the mediating role of last mile delivery / Y. Vakulenko et al. *The International Review of Retail, Distribution and Consumer Research*. 2019. Vol. 29, no. 3. P. 306–320. URL: https://doi.org/10.1080/09593969.2019.1598466 (date of access: 13.05.2025).

4) Ключові проблеми у сфері транспорту та логістики під час війни в Україні –. *logist.today*. URL: https://logist.today/osoboe_mnenie-uk/2024-06-18/klyuchevye-problemy-v-sfere-transporta-i-logistiki-v-period-vojny-v-ukraine-2/ (дата звернення: 13.05.2025).

5) Industrial vehicle routing problem: a case study - Journal of Shipping and Trade. *SpringerOpen*. URL: https://jshippingandtrade.springeropen.com/articles/10.1186/s41072-022-00108-7 (date of access: 13.05.2025).

6) Supply-Chain Optimization: Levers For Rapid EBITDA. *Oliver Wyman*. URL: https://www.oliverwyman.com/our-expertise/insights/2018/may/supply-chain-optimization--levers-for-rapid-ebitda.html (date of access: 13.05.2025).

7) Gevaers R., Voorde E., Vanelslander T. Characteristics and Typology of Last-mile Logistics from an Innovation Perspective in an Urban

Context. *City Distribution and Urban Freight Transport: Multiple Perspectives*. 2011. P. 56–71. URL: https://doi.org/10.4337/9780857932754.00009.

8) The Last Mile Problem: Challenges and Innovative Solutions. *Vivatechnology*. URL: https://vivatechnology.com/news/the-last-mile-problem-challenges-and-innovative-solutions (date of access: 14.05.2025).

9) A History Of GPS Vehicle Tracking and Fleet Telematics. *Track Your Truck*. URL: https://www.trackyourtruck.com/blog/history-gps-vehicle-tracking-and-fleet-telematics/ (date of access: 14.05.2025).

10) The Future of Logistics: How Autonomous Vehicles and Drones are Changing the Game - Operations Council. *Operations Council*. URL: https://operationscouncil.org/the-future-of-logistics-how-autonomous-vehicles-and-drones-are-changing-the-game/ (date of access: 14.05.2025).

11) Global Retail E-Commerce Sales Will Reach $6.8 Trillion By 2028. *Forrester*. URL: https://www.forrester.com/blogs/global-retail-e-commerce-sales-will-reach-6-8-trillion-by-2028/ (date of access: 14.05.2025).

12) European E-commerce Delivery Compass 2021/2022 | Sendcloud. *Sendcloud*. URL: https://www.sendcloud.com/en_uk/whitepapers/ecommerce-delivery-compass-2021/ (date of access: 14.05.2025).

13) Пальне дорожчає: Що очікувати від цін на бензин у 2025 році. *Місто Кия - Ваш гід по Києву: афіша, заклади та цікаві місця*. URL: https://mistokyia.ua/business/palne-dorozhchaie-shcho-ochikuvaty-vid-tsin (дата звернення: 15.05.2025).

14) Samborska V. Scaling up: how increasing inputs has made artificial intelligence more capable. *Our World in Data*. URL: https://ourworldindata.org/scaling-up-ai (date of access: 15.05.2025).

15) Widuch J. Current and emerging formulations and models of real-life rich vehicle routing problems. *Smart delivery systems*. 2020. P. 1–35. URL: https://doi.org/10.1016/b978-0-12-815715-2.00006-3(date of access: 07.06.2025).

16) On a road to optimal fleet routing algorithms: a gentle introduction to the state-of-the-art / P. Gora et al. *Smart delivery systems*. 2020. P. 37–92. URL: https://doi.org/10.1016/b978-0-12-815715-2.00014-2 (date of access: 07.06.2025).

17) Mahmudy W. F., Widodo A. W., Haikal A. H. Challenges and opportunities for applying meta-heuristic methods in vehicle routing problems: a review. *Mechanical engineering, science and technology international conference*. Basel Switzerland, 2024. URL: https://doi.org/10.3390/engproc2024063012 (date of access: 07.06.2025).

18) Nghia N., Nguyen M. H., Linh N. Solving practical vehicle routing problem with time windows (VRPTW) – A case study of ICD tien son (vietnam). 2016.

19) GeeksforGeeks. Branch and bound algorithm - geeksforgeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/branch-and-bound-algorithm/ (date of access: 07.06.2025).

20) Dynamic programming explained: efficient optimization. *awork: Work management for agency, consulting & tech*. URL: https://www.awork.com/glossary/dynamic-programming#:~:text=Dynamic%20programming%20is%20an%20approach,to%20save%20time%20and%20resources. (date of access: 07.06.2025).

21) *Modeliks | Business planning software*. URL: https://www.modeliks.com/industries/professional-services/last-mile-delivery-services-kpis-dashboard (дата звернення: 07.06.2025).

22) GeeksforGeeks. Implementation of XGBoost (eXtreme Gradient Boosting) - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/implementation-of-xgboost-extreme-gradient-boosting/ (date of access: 07.06.2025).

23) Hassan N. Feature engineering: unlocking ML model optimization. *Medium*. URL: https://medium.com/@nay1228/feature-

engineering-unlocking-ml-model-optimization-a892c51bd952 (date of access: 07.06.2025).

24) What is geocoding - geocoding definition. *Caliper - Mapping Software, GIS, and Transportation Software*. URL: https://www.caliper.com/glossary/what-is-geocoding.htm#:~:text=Geocoding%20is%20the%20process%20of,visualize%20and%20analyze%20spatial%20data. (date of access: 07.06.2025).

25) Saeed L. 10 reasons why java is the future of enterprise app development in 2025. *Blog | Payara*. URL: https://blog.payara.fish/10-compelling-reasons-why-java-is-the-future-of-enterprise-app-development-in-2025 (date of access: 07.06.2025).