Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ОПТИМІЗАЦІЯ МАРШРУТІВ І ЦІНОУТВОРЕННЯ У ЗАДАЧІ УПРАВЛІННЯ ВАНТАЖОПЕРЕВЕЗЕННЯМИ
(тема)

Виконав:
здобувач 2 року навчання,
групи ІНФм-24-2
Поляков В. Д.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Науковий керівник проф. Машталір С. В.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Полякову Владиславу Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Оптимізація маршрутів і ціноутворення у задачі управління вантажоперевезеннями

затверджена наказом університету від 14 листопада 2025 року № 1045Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 19 листопада 2025 р.
3. Вихідні дані до роботи літературні джерела та наукові праці, що висвітлюють сучасні методи оптимізації логістичних процесів, розрахунку вартості перевезень та побудови маршрутів, технічна документація та специфікації Google Maps Platform.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз сучасних підходів до розрахунку вартості логістичних перевезень та методів оптимізації транспортних процесів.
2. Дослідження літературних джерел і технічної документації щодо використання зовнішніх картографічних сервісів для побудови маршрутів та визначення відстаней.
3. Аналіз існуючих інформаційних систем у сфері транспортної логістики та визначення їхніх функціональних особливостей.
4. Проектування структури даних для логістичної системи та моделювання взаємозв'язків між водіями, траками, диспетчерами та вантажними рейсами.
5. Розробка програмного застосунку.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність автоматизації в логістиці, об'єкт та мету дослідження, вхідні дані та мета дослідження, аналіз існуючих рішень, архітектура системи, формули розрахунків, демонстрація додання вантажів, демонстрація прорахування статистики в окремих категоріях, демонстрація загальної статистики, перспективи та апробацію роботи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	29.09.2025	
2	Аналіз завдання, підбір літератури	30.09.25-05.10.25	
3	Аналіз літератури з досліджуваної проблеми	06.10.25-12.10.25	
4	Особливості методів прорахування вартості вантажу	13.10.25-22.10.25	
5	Дослідження методів прорахування вартості вантажу	22.10.26-27.10.25	
6	Програмна реалізація	27.10.25-01.11.25	
7	Обґрунтування отриманих результатів	02.11.25-08.11.25	
8	Оформлення пояснювальної записки	09.11.25-13.11.25	
9	Перевірка на нормоконтроль	29.11.25-12.12.25	
10	Перевірка на плагіат	30.11.25-10.12.25	
11	Рецензування	01.12.25-10.12.25	
12	Підготовка презентації та доповіді	30.11.25-22.12.25	
13	Занесення роботи в електронний архів	30.11.25-22.12.25	
14	Попередній захист кваліфікаційної роботи	30.12.25-22.12.25	

Дата видачі завдання 29 вересня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф. Машталір С. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 80 с., 11 рис., 49 джерел.

ВАЛОВИЙ ДОХІД ПЕРЕВЕЗЕННЯ, ДОПОМОГА ВОДІЯ, КАРТОГРАФІЧНИЙ СЕРВІС, ЛОГІСТИЧНИЙ ВЕБЗАСТОСУНОК, ОПЛАТА ЗА МИЛЮ, РЕЛЯЦІЙНА БАЗА ДАНИХ, РОЗРАХУНОК МАРШРУТІВ, СЕРВЕРНИЙ ВЕБФРЕЙМВОРК, СИСТЕМА ОБЛІКУ РЕЙСІВ, СКАСУВАННЯ ВАНТАЖУ, ФІНАНСОВА АНАЛІТИКА, ЧАС ПРОСТОЮ.

Об’єктом дослідження є процеси управління логістичними перевезеннями та їх економічна оптимізація, а предметом методи й програмні засоби, що забезпечують розрахунок вартості перевезень і аналітику логістичних даних. Мета роботи полягає у створенні вебзастосунку, який автоматизує визначення вартості перевезення, аналізує ключові показники ефективності й будує оптимальні маршрути на основі даних картографічного сервісу Google Maps. У дослідженні застосовано математичні моделі розрахунку вартості рейсу з урахуванням відстані, тарифів, додаткових витрат та інші параметри, а також використано сучасні вебтехнології для створення інтерфейсу користувача.

Наукова новизна полягає у розробці інтегрованої системи, що поєднує розрахунок вартості, аналітику логістичних показників і побудову маршрутів в одному інтерфейсі, що підвищує точність планування та знижує витрати. Отримані результати можуть бути впроваджені в логістичних компаніях для оптимізації перевізного процесу, аналізу транспортних витрат і планування маршрутів, сприяючи цифровій трансформації логістичних операцій.

ABSTRACT

Explanatory note to the qualification work: 80 pages, 11 figures, 49 sources.

CARGO CANCELLATION, CARTOGRAPHIC SERVICE, DRIVER ASSISTANCE, FINANCIAL ANALYTICS, FREIGHT GROSS REVENUE, LOGISTICS WEB APPLICATION, MILEAGE-BASED PAYMENT, RELATIONAL DATABASE, ROUTE CALCULATION, RIDE MANAGEMENT SYSTEM, SERVER-SIDE WEB FRAMEWORK, IDLE TIME MONITORING.

The object of the research is the management processes of logistics transportation and their economic optimization, while the subject is the methods and software tools that enable transportation cost calculation and logistics data analytics.

The purpose of this work is to develop a web application that automates transportation cost calculation, analyzes key performance indicators, and builds optimal routes based on data from the Google Maps mapping service.

The study employs mathematical models for calculating the cost of a transportation route, taking into account distance, tariffs, additional expenses, and other parameters, as well as modern web technologies for creating a user interface. The scientific novelty lies in the development of an integrated system that combines cost calculation, logistics performance analytics, and route generation in a single interface, which improves planning accuracy and reduces operational expenses.

The obtained results can be implemented in logistics companies to optimize transportation processes, analyze transport costs, and plan routes, thereby contributing to the digital transformation of logistics operations.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (програмний інтерфейс додатків)

ETA – Estimated Time of Arrival (розрахунковий час прибуття)

GPS – Global Positioning System (глобальна система позиціонування)

JSON – JavaScript Object Notation (формат обміну даними)

REST – Representational State Transfer (архітектурний стиль вебсервісів)

SQL – Structured Query Language (мова запитів для баз даних)

UI – User Interface (інтерфейс користувача)

UX – User Experience (зручність взаємодії з застосунком)

DT – Detention Time (простій вантажівки на завантаженні чи розвантаженні)

TONU – Truck Ordered Not Used (компенсація за відміну завантаження)

ORM – Object–Relational Mapping (інструмент прив’язки об’єктів до таблиць БД)

ВСТУП

Логістика є ключовим елементом сучасної економіки, що забезпечує ефективне переміщення товарів, ресурсів та послуг між виробниками і споживачами. З розвитком електронної комерції, глобальних ланцюгів постачання та цифрових технологій вимоги до точності, швидкості та прозорості логістичних процесів постійно зростають. У таких умовах виникає необхідність в автоматизації процесів планування перевезень, розрахунку їхньої вартості та оптимізації маршрутів доставки.

Сучасні інформаційні системи у сфері транспортної логістики орієнтовані на інтеграцію аналітичних інструментів із картографічними сервісами. Одним із найбільш потужних інструментів для цього є Google Maps API, який дозволяє отримувати актуальні дані про маршрути, відстані та час у дорозі. Використання таких API дає змогу створювати вебзастосунки, що автоматично обробляють просторові дані та виконують аналітичні розрахунки, необхідні для прийняття ефективних управлінських рішень.

Актуальність роботи полягає у зростаючій потребі логістичних компаній у програмних рішеннях, які дозволяють швидко та точно розраховувати вартість перевезень, враховуючи реальні дорожні умови, тип транспорту, тарифи та додаткові логістичні фактори. У багатьох існуючих системах такі розрахунки виконуються вручну або з використанням спрощених моделей, що призводить до неточностей та зниження ефективності транспортного планування.

Стан сучасних досліджень у галузі цифрової логістики свідчить про тенденцію до поєднання аналітики, автоматизації та геоінформаційних технологій. Розробка вебзастосунків на основі API відкриває нові можливості для створення гнучких, масштабованих і користувацьких систем, що здатні обробляти великі обсяги даних у реальному часі. Особливо важливим є впровадження алгоритмів, які можуть не лише побудувати маршрут, але й оцінити його економічну доцільність.

Наукова задача, яку вирішує дана робота, полягає у розробці вебзастосунку, що забезпечує інтегроване обчислення вартості перевезення та аналіз логістичних показників на основі даних, отриманих із Google Maps API. Це дозволить підвищити рівень автоматизації логістичних процесів і надати користувачам зручний інструмент для прийняття рішень, орієнтованих на мінімізацію витрат і оптимізацію маршрутів.

Таким чином, створення вебзастосунку для логістики з розрахунку вартості перевезень та пошуку найкоротших маршрутів є актуальним завданням сучасної транспортної інформатики. Його реалізація сприятиме цифровій трансформації логістичних процесів, підвищенню ефективності управління транспортом і загальному зниженню витрат на перевезення.

Актуальність дослідження зумовлена постійним зростанням обсягів транспортних перевезень і потребою у підвищенні ефективності логістичних процесів. В умовах глобалізації ринків, зростання вимог клієнтів до швидкості та точності доставки, а також зростання вартості пального та транспортних послуг, питання оптимізації маршрутів і точного розрахунку вартості перевезення набувають особливого значення.

Традиційні методи планування перевезень здебільшого ґрунтуються на досвіді логістів або використанні базових географічних сервісів без інтегрованого аналітичного компоненту. Це призводить до нераціонального використання транспортних ресурсів, перевитрат коштів і часу, а також ускладнює прийняття рішень у динамічному середовищі перевезень.

Використання сучасних вебтехнологій та API-сервісів, зокрема Google Maps API, відкриває нові можливості для побудови систем, які можуть автоматично отримувати інформацію про маршрути, відстані, час у дорозі та дорожню ситуацію. Поєднання цих даних із математичними моделями розрахунку вартості перевезень дозволяє створити інтелектуальний інструмент для логістичного аналізу.

Особливу значущість дослідження визначає необхідність розробки універсального вебзастосунку, який би забезпечував автоматизацію двох

ключових етапів логістики економічного розрахунку та маршрутизації. Такий підхід сприяє підвищенню точності планування, зниженню витрат, підвищенню конкурентоспроможності транспортних компаній і загальній цифровій трансформації галузі.

Крім того, розробка подібного вебзастосунку має освітнє та науково–прикладне значення. Вона дозволяє дослідити взаємозв’язок між аналітичними моделями вартості перевезень і геоінформаційними технологіями, а також створити платформу для подальших розробок у сфері транспортної аналітики.

Отже, актуальність дослідження полягає у потребі створення ефективного, інтегрованого та автоматизованого інструменту для розрахунку вартості перевезень і аналізу логістичних даних, який базується на сучасних вебтехнологіях і реальних географічних даних.

1 ОГЛЯД ОСНОВНИХ МЕТОДІВ ТА ЗАСОБІВ РОЗРАХУНКУ ВАРТОСТІ ПЕРЕВЕЗЕНЬ І ПОБУДОВИ МАРШРУТІВ У ЛОГІСТИЦІ

1.1 Перспективи розвитку логістичних вебзастосунків і їхня роль на сучасному ринку

У сучасному світі логістика є однією з найдинамічніших і найприбутковіших галузей глобальної економіки. З розвитком електронної комерції, міжнародної торгівлі та транспортної інфраструктури, потреба в ефективному управлінні перевезеннями, оптимізації маршрутів і зниженні витрат на транспортування стрімко зростає. За останні роки обсяги вантажних перевезень у світі значно збільшилися, а конкуренція між компаніями, що надають логістичні послуги, змусила їх активно впроваджувати цифрові рішення.

Основним напрямом розвитку галузі є автоматизація процесів та аналітика даних. Вебзастосунки, здатні виконувати розрахунок вартості перевезення, аналізувати транспортні витрати та пропонувати найкоротші маршрути доставки, стають важливими інструментами для підвищення ефективності бізнесу. Вони дозволяють скорочувати час планування, мінімізувати людські помилки та підвищувати точність прогнозування витрат. Такий підхід особливо актуальний для великих компаній, що мають сотні транспортних засобів і потребують постійного контролю за логістичними процесами в режимі реального часу.

Програмне забезпечення, яке здатне аналізувати історію перевезень, відстежувати поведінку транспортних засобів, прогнозувати затримки та автоматично обчислювати собівартість маршрутів, має величезний попит. Крім того, такі системи сприяють екологічній сталості, адже дозволяють зменшити витрати пального та шкідливі викиди, що відповідає сучасним світовим трендам зеленої логістики.

Особливо перспективним ринком для подібних рішень є Сполучені Штати Америки. Американський логістичний сектор є одним із найбільших у світі він включає тисячі компаній, що займаються вантажними перевезеннями, складським зберіганням і дистрибуцією [1]. Попит на програмне забезпечення для управління перевезеннями у США щорічно зростає, оскільки підприємства прагнуть знизити витрати, автоматизувати документообіг і покращити ефективність роботи. Висока цифровізація бізнесу, розвинена інфраструктура, велика кількість стартапів у сфері логістичних технологій створюють сприятливі умови для впровадження нових вебзастосунків.

Окрім цього, американський ринок вирізняється високою конкуренцією, що стимулює постійний пошук інновацій. Користувачі активно приймають сучасні IT-рішення, якщо вони дійсно допомагають скоротити витрати, покращити аналітику або спростити управління процесами. Програмний продукт, який забезпечує швидкий розрахунок вартості перевезення, аналітичні звіти, а також пропонує ефективні маршрути доставки, може бути надзвичайно привабливим для логістичних компаній, транспортних брокерів і навіть індивідуальних перевізників.

Таким чином, створення вебзастосунку для логістики, який дозволяє проводити розрахунок вартості перевезень, здійснювати аналіз ефективності маршрутів і надавати користувачеві зручний інтерфейс для управління транспортними операціями, є не лише актуальним, а й комерційно вигідним напрямом. Такий продукт має значний потенціал не лише для внутрішнього ринку, але й для міжнародного насамперед американського, де попит на цифрові логістичні рішення стабільно зростає.

1.2 Формування вартості перевезення

Формування вартості перевезення є складним процесом, що охоплює широкий спектр економічних, технічних і організаційних факторів. Вартість

логістичної послуги залежить від сукупності змінних і постійних витрат, а також від зовнішніх умов, таких як стан інфраструктури, ціни на пальне, попит на перевезення та регіональні особливості [2-3]. У більшості випадків загальна ціна перевезення складається з базової собівартості, прибуткової націнки перевізника та можливих додаткових зборів, які виникають у процесі виконання рейсу.

Найвагомішу частку у структурі витрат займають змінні витрати. До них належать витрати на пальне, яке є головним елементом формування собівартості. Споживання пального прямо залежить від типу транспортного засобу, його вантажопідйомності, технічного стану та характеристик маршруту. На маршрутах із великою кількістю підйомів або у міських зонах, де часті зупинки, витрата пального зростає, що безпосередньо впливає на кінцеву ціну перевезення. Другою значною складовою є оплата праці водія, що може бути погодинною або залежати від пройденої відстані. У деяких випадках враховуються добові витрати, оплата простоїв, надбавки за нічні зміни чи складні погодні умови.

Додатковими змінними витратами є плата за користування платними дорогами, мостами або тунелями, оренда стоянок, а також витрати, пов’язані з отриманням спеціальних дозволів для перевезення негабаритних або небезпечних вантажів. Усі ці фактори можуть суттєво збільшити загальну вартість рейсу. Особливу роль відіграють так звані порожні пробіги, коли транспортний засіб повертається без вантажу. Такі поїздки не приносять прибутку, але створюють додаткові витрати на пальне, амортизацію і заробітну плату. Саме тому ефективне планування логістичних маршрутів і пошук зворотних вантажів є одним із найважливіших завдань при зменшенні витрат.

До постійних витрат належать амортизація транспорту, технічне обслуговування, страхування, податки, адміністративні витрати, витрати на диспетчерську службу та облік. Хоча вони не залежать безпосередньо від кількості перевезень, у розрахунках вартості їх розподіляють на певну кількість рейсів або кілометрів пробігу. У сукупності постійні витрати визначають мінімальний рівень доходу, за якого перевізнику вигідно працювати.

Визначення загальної вартості перевезення вимагає врахування не лише фінансових складових, але й аналітичних аспектів, таких як ефективність маршруту, завантаженість транспорту, часові обмеження та логістичні ризики. У сучасних умовах для цього активно застосовуються цифрові технології, що дозволяють автоматично розраховувати вартість на основі поточних цін на пальне, відстані, середньої швидкості руху, дорожніх зборів та інших показників.

Зменшення вартості перевезення можливе завдяки низці стратегічних та операційних заходів. Одним із найефективніших способів є оптимізація маршрутів, що дозволяє мінімізувати пробіг і скоротити витрати на пальне. Застосування спеціалізованих вебзастосунків, які розраховують найкоротший або найекономічніший маршрут з урахуванням реальної дорожньої ситуації, здатне знизити витрати на десятки відсотків. Значний вплив має технічний стан транспорту своєчасне обслуговування зменшує витрати на ремонт і споживання пального. Економічно доцільним є також впровадження систем моніторингу руху та стилю водіння, адже агресивна манера керування призводить до перевитрати пального і підвищеного зносу техніки.

Важливим напрямом зменшення витрат є ефективне використання вантажного простору та зниження кількості порожніх рейсів. Пошук зворотних вантажів або об’єднання замовлень кількох клієнтів у межах одного маршруту дозволяє підвищити рентабельність кожного перевезення. Застосування автоматизованих аналітичних систем допомагає прогнозувати попит, оптимізувати використання автопарку і підвищувати ефективність логістичної мережі в цілому.

Таким чином, формування вартості перевезення є комплексним процесом, який поєднує економічні розрахунки, технічну оцінку та логістичний аналіз. Ефективне управління цими факторами дозволяє не лише забезпечити конкурентоспроможну ціну на ринку, а й створити умови для стабільного розвитку транспортного бізнесу. Використання вебзастосунків для автоматичного розрахунку вартості перевезень і пошуку оптимальних маршрутів

стає невід’ємною частиною сучасної логістики, оскільки сприяє скороченню витрат, підвищенню точності планування та загальній прозорості операцій.

1.3 Значення пошуку найкоротшого маршруту у логістичних перевезеннях

У сучасній логістиці визначення найкоротшого та найефективнішого маршруту перевезення є одним із ключових факторів, що впливають на економічність, швидкість і надійність доставки вантажів. Від правильності побудови маршруту залежить не лише час виконання рейсу, але й рівень витрат на пальне, амортизацію транспорту, оплату праці водіїв і навіть безпеку самого перевезення. У світі, де кожна хвилина простою чи зайвий кілометр означає фінансові втрати, пошук оптимального шляху набуває стратегічного значення для всіх учасників логістичного процесу від водія до власника транспортної компанії.

Традиційно маршрути перевезень будувалися вручну або за допомогою простих інструментів паперових карт, попередніх маршрутних листів чи базових навігаційних систем. Проте з розвитком цифрових технологій планування шляху все частіше здійснюється за допомогою GPS-навігаторів і картографічних сервісів, які враховують дорожню мережу, обмеження руху, дозволені швидкості та навіть середню інтенсивність трафіку [4]. Незважаючи на зручність таких систем, вони не завжди здатні забезпечити ідеальне рішення. У реальних умовах на маршруті можуть виникати непередбачувані обставини дорожні ремонти, перекриття трас, аварії, погодні умови або зміни в законодавчих обмеженнях для вантажного транспорту. Навіть сучасні GPS-системи не завжди миттєво оновлюють ці дані, через що водії стикаються з ситуаціями, коли обраний маршрут виявляється непридатним або значно довшим, ніж очікувалося

Проблема ускладнюється тим, що при міжміських чи міжнародних перевезеннях маршрути можуть проходити через різні юрисдикції, де діють

власні правила, обмеження руху для певних типів вантажів або платні ділянки доріг. У таких випадках побудова найкоротшого маршруту потребує не лише врахування відстані, а й аналізу економічної доцільності: дешевший шлях за кілометражем може виявитися дорожчим через наявність численних платних відрізків або тривалих заторів. Тому сучасні логістичні системи дедалі частіше використовують аналітичні алгоритми, що комбінують картографічні дані з економічними параметрами [5-6]: вартістю пального, часом руху, очікуваними затримками, витратами на мита та інші збори.

Особливу роль відіграє динамічне планування маршруту, яке дозволяє в реальному часі змінювати шлях залежно від дорожньої ситуації. Це можливо завдяки інтеграції вебзастосунків із API-картографічних сервісів, що надають актуальні дані про трафік, перекриття та погодні умови. Такий підхід дає змогу водієві або диспетчеру миттєво реагувати на зміни, обираючи найвигідніший альтернативний шлях. Таким чином, завдяки інтелектуальним алгоритмам і постійному оновленню інформації логістичні компанії можуть суттєво скоротити час доставки та знизити витрати.

Важливість пошуку найкоротшого маршруту проявляється не лише в економічному аспекті, але й у підвищенні рівня сервісу. Замовники очікують максимальної точності виконання графіка доставки, і кожна затримка може вплинути на репутацію компанії. Раціонально спланований маршрут допомагає не лише вчасно доставити вантаж, а й уникнути ризиків, пов’язаних із перевантаженням певних ділянок, зношуванням транспортних засобів або підвищеним споживанням пального.

Тому побудова найкоротшого маршруту є не просто технічною задачею, а ключовим елементом управління логістичним процесом. В умовах високої конкуренції на ринку транспортних послуг саме здатність компанії швидко і точно визначати оптимальні шляхи руху забезпечує її ефективність і прибутковість. Використання сучасних вебзастосунків, що дозволяють враховувати всі змінні фактори маршруту та оновлювати дані в режимі реального часу, відкриває новий рівень автоматизації й аналітики у сфері

перевезень. Такий підхід не лише скорочує витрати, а й підвищує гнучкість і надійність усієї логістичної системи.

1.4 Використання вебтехнологій у сучасних логістичних системах

Сучасна логістика неможлива без застосування вебтехнологій, які забезпечують ефективну взаємодію між клієнтами, перевізниками, диспетчерами та аналітичними системами. Вебзастосунки дозволяють у режимі реального часу розраховувати вартість перевезень, відстежувати рух транспорту, оптимізувати маршрути й аналізувати ефективність роботи компанії.

Завдяки розвитку технологій Front-end та Back-end, з’явилась можливість створювати інтерактивні, швидкі та масштабовані рішення для логістики. На боці клієнта зазвичай використовуються сучасні фреймворки React, Angular, Vue.js, які дозволяють реалізувати зручний інтерфейс, візуалізувати карти та відображати результати розрахунків [7]. На стороні сервера часто застосовуються Node.js, Python або Java, що забезпечують обробку запитів, взаємодію з базою даних та інтеграцію з зовнішніми API.

Особливу роль у логістиці відіграє використання REST API для обміну даними між клієнтською та серверною частинами. Це дає змогу забезпечити швидку взаємодію, незалежність модулів та можливість інтеграції з іншими сервісами.

У контексті вебзастосунків для логістики важливе значення має візуалізація даних. Інтерфейс має не лише відображати розраховані маршрути, але й забезпечувати користувачу можливість аналізу наприклад, побачити довжину шляху, орієнтовну вартість перевезення, альтернативні маршрути чи попередні історії поїздок.

Крім цього, сучасні системи логістики інтегрують аналітичні модулі, які дозволяють відстежувати показники ефективності середній час доставки,

середню відстань, співвідношення прибутку до витрат. Ці дані можуть бути представлені у вигляді графіків, таблиць або інтерактивних панелей управління.

Таким чином, вебтехнології у логістиці є не лише інструментом для побудови користувацького інтерфейсу, а й основою для створення повноцінної екосистеми, що забезпечує цифровізацію бізнес-процесів, підвищення точності планування маршрутів і ефективне управління транспортними ресурсами.

1.5 Аналіз існуючих інформаційних систем у сфері логістики

Сучасні вебзастосунки для логістики дедалі частіше спираються на інтеграцію з зовнішніми прикладними інтерфейсами програмування, які надають доступ до готових функцій, баз даних або аналітичних сервісів. Такий підхід дозволяє значно скоротити час розробки, підвищити точність розрахунків і забезпечити доступ до актуальних географічних, транспортних та аналітичних даних.

У контексті логістичних систем найпоширенішими є картографічні та геолокаційні API, які надають можливість отримувати інформацію про відстані між пунктами, тривалість маршруту, стан трафіку, а також будувати карти для візуалізації результатів [8]. Серед найбільш популярних рішень виділяються Google Maps API, Google Distance Matrix API, OpenRouteService API, Mapbox API та Here Maps API.

Найбільш зручним та широко застосовуваним у логістичних вебзастосунках є Google Maps Platform, яка складається з набору сервісів для роботи з геолокаційними даними. Основними складовими, що використовуються у логістичних рішеннях, є:

– Maps JavaScript API забезпечує візуалізацію карти, побудову маршрутів, нанесення маркерів і відображення додаткових даних;

– Directions API дозволяє розраховувати найкоротші або найшвидші маршрути між кількома точками, з урахуванням дорожньої ситуації, заторів і типу транспорту;

– Distance Matrix API надає можливість отримати інформацію про відстань і час у дорозі між будь–якими двома або кількома локаціями, що є ключовим для розрахунку вартості перевезень;

– Geocoding API використовується для перетворення адреси на координати та навпаки, що полегшує пошук і відображення точок на карті.

Інтеграція таких API у вебзастосунок відбувається через HTTP-запити, які надсилаються на сервер Google з параметрами, що містять координати, тип транспорту, режим маршруту та інші характеристики. Отримані у відповідь дані у форматі JSON або XML обробляються на серверній частині застосунку і передаються на клієнтську частину для відображення користувачу.

Перевагою використання зовнішніх API є висока точність і актуальність даних, адже такі сервіси постійно оновлюються та враховують реальні умови дорожнього руху [9]. Крім того, розробник звільняється від необхідності самостійно реалізовувати складні алгоритми пошуку маршрутів, обчислення відстаней і обробки геоданих ці функції повністю делегуються зовнішньому сервісу.

З іншого боку, використання сторонніх API потребує врахування певних аспектів, таких як обмеження кількості запитів, вимоги до реєстрації API-ключів, питання вартості використання сервісу. Також необхідно забезпечити безпечне зберігання ключів API, щоб запобігти їх несанкціонованому використанню.

У логістичних вебзастосунках інтеграція з Google API дозволяє автоматизувати процеси, які раніше вимагали ручних розрахунків. Наприклад, під час створення заявки на перевезення система може автоматично:

– отримати координати пунктів відправлення та призначення через Geocoding API;

– розрахувати найкоротший або найшвидший маршрут через Directions API;

– визначити реальну відстань і орієнтовний час у дорозі за допомогою Distance Matrix API;

– на основі отриманих даних сформувати розрахунок вартості перевезення, враховуючи тарифи, паливо, додаткові послуги тощо;

– основну аналітику на інтерактивній карті.

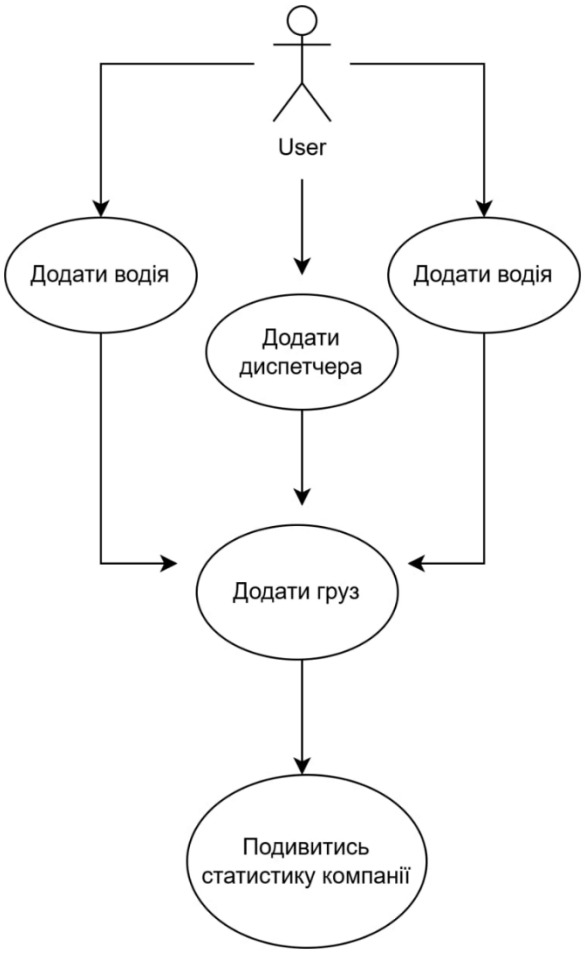

Рисунок 1.1 – Діаграма use case варіантів користуванням застосунку

Використання зовнішніх API у логістичних вебзастосунках не лише підвищує функціональність системи, а й відіграє ключову роль у цифровізації логістичних процесів. Завдяки можливості автоматично отримувати дані про міста, маршрути, відстані чи додаткову інформацію, застосунок стає значно точнішим, швидшим та зручнішим у роботі. Це дозволяє суттєво скоротити час, необхідний для прийняття рішень, зменшує кількість помилок під час

планування та розрахунків і забезпечує більш високий рівень сервісу для клієнтів, що в кінцевому підсумку підвищує ефективність діяльності компанії.

Для кращого розуміння структури (рис 1.1) та функціональних можливостей системи була розроблена діаграма прецедентів, яка відображає основні сценарії взаємодії користувача з вебзастосунком [10-11]. На ній головний актор користувач взаємодіє з системою, виконуючи такі типові дії, як додавання водія, трака, диспетчера, створення вантажу та перегляд щомісячної статистики. USE–case діаграма дозволяє візуально представити логіку роботи програмного забезпечення, визначити межі системи та узгодити функціональні вимоги ще на етапі проектування. Такий підхід забезпечує цілісність і продуманість архітектури, спрощує розробку інтерфейсу, структурує бізнес-логіку та мінімізує помилки, що можуть виникнути під час реалізації проекту.

У результаті діаграма прецедентів виступає важливим інструментом планування, який допомагає не лише описати функції, але й сформувати чітке бачення взаємодії користувача із системою, що є фундаментальним кроком для створення ефективного та надійного логістичного вебзастосунку.

1.6 Постановка задачі дослідження

Таким чином, розрахунок вартості перевезень та визначення найкоротших маршрутів є актуальним завданням у сучасній логістиці. З огляду на постійне зростання обсягів вантажоперевезень, підвищення цін на пальне та збільшення конкуренції між транспортними компаніями, особливої ваги набувають інструменти, які дозволяють ефективно планувати маршрути, оптимізувати витрати та здійснювати аналітику перевізного процесу. Прийнято рішення про розроблення вебзастосунку для логістики, який дозволить користувачам розраховувати вартість перевезення залежно від відстані, типу транспорту, ваги вантажу та інших факторів, а також автоматично визначати найкоротший або найекономічніший маршрут із використанням картографічного API від Google.

Об’єктом дослідження є маршрути перевезення вантажів у транспортній логістиці.

Метою дослідження є розробка вебзастосунку, що дозволяє виконувати автоматизований розрахунок вартості перевезення вантажу та визначати найкоротший маршрут із використанням сучасних інтернет-технологій і картографічних сервісів [12-13].

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз літературних джерел і сучасних підходів до розрахунку вартості перевезень у логістиці;

– дослідити методи побудови найкоротших маршрутів і принципи роботи навігаційних систем;

– проаналізувати можливості застосування картографічного API для побудови маршруту та розрахунку відстані;

– розробити математичну модель розрахунку вартості перевезення з урахуванням відстані, ваги вантажу, витрат пального та супутніх зборів;

– створити архітектуру вебзастосунку для логістики, що забезпечить інтеграцію з зовнішніми сервісами та зручний користувацький інтерфейс;

– реалізувати програмний прототип вебзастосунку з функціоналом розрахунку вартості перевезення і пошуку найкоротшого маршруту;

– провести тестування роботи застосунку на реальних прикладах маршрутів і здійснити аналіз отриманих результатів.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ ДЛЯ ЛОГІСТИКИ З ЗОЗРАХУНКУ ВАРТОСТІ ПЕРЕВЕЗЕНЬ ТА ПОШУКУ НАЙКОРОТШИХ МАРШРУТІВ

2.1 Вимоги до вебзастосунку

Застосунок для аналітики логістичних перевезень створюється з метою автоматизації процесів розрахунку маршрутів, відстаней між пунктами транспортування, вартості перевезень, а також збору статистичних даних для подальшої побудови графіків та аналізу ефективності роботи компанії. Основна ідея системи полягає у тому, щоб надати зручний інструмент для розрахунку перевезень за допомогою інтеграції з картографічними сервісами та забезпечити можливість формування докладної статистики для кожного учасника логістичного процесу компанії, диспетчера та водія.

Після відкриття вебзастосунку користувачу буде запропоновано ввести вихідні дані маршруту, поштові ZIP-коди пунктів відправлення, проміжних зупинок та кінцевого пункту доставки. Після введення необхідних даних система через інтеграцію з Google Maps API здійснює пошук маршруту, визначає найкоротший можливий шлях між усіма введеними пунктами та обчислює загальну довжину маршруту у милях або кілометрах [14]. Отримані дані будуть виведені на екран у зручному для користувача вигляді, а також відображені на інтерактивній карті, що дозволить візуально оцінити маршрут транспортування.

Після розрахунку відстані користувач може ввести ставку за милю або кілометр, яка відповідає поточним ринковим тарифам компанії. Після підтвердження введених даних програма автоматично обчислює орієнтовну вартість перевезення, що включає як витрати компанії, так і потенційний прибуток. Для більш точного аналізу система дозволяє закріпити за певним маршрутом водія, транспортний засіб, а також диспетчера, який відповідає за організацію конкретного перевезення. Такий підхід дозволяє не лише

розраховувати вартість кожного окремого рейсу, але й накопичувати аналітичні дані для подальшого узагальнення.

Після завершення розрахунків користувач має можливість зберегти дані про маршрут у базі, вказавши дату, номер траку, ім’я або код водія, а також інформацію про диспетчера. Збережена інформація використовується для формування статистики, де кожен виконаний рейс додається до загальної бази перевезень. Завдяки цьому, з часом система накопичує великий обсяг даних, який можна використовувати для подальшого аналізу визначення найефективніших водіїв, диспетчерів, а також для аналізу економічних показників компанії за певний період [15].

У застосунку передбачено можливість відображення статистики у вигляді таблиць і графіків. Наприклад, користувач може переглядати загальний прибуток компанії за місяць або рік, порівнювати середню вартість рейсів, аналізувати, який відсоток від загальної суми отримують диспетчери, а який водії. Для зручності аналізу інформація буде представлена у вигляді інтерактивних графіків, що дозволяють наочно порівнювати результати за різні часові проміжки або за різними параметрами.

Особливістю вебзастосунку є те, що він не має поділу на користувацьку та адміністративну частини. Усі розрахунки, введення даних, перегляд результатів і побудова графіків здійснюються в єдиному інтерфейсі. Це спрощує використання системи та робить її зручною навіть для невеликих транспортних компаній або приватних логістичних операторів, яким не потрібна багаторівнева система доступу. Користувач може вільно переходити між розділами, вводити нові маршрути, переглядати попередні результати або формувати звіти за будь–який обраний період.

Інтерфейс програми буде побудований з урахуванням максимальної простоти та інтуїтивності. У верхній частині вебсторінки передбачено навігаційне меню, яке дозволяє перейти до основних розділів: розрахунку маршруту, статистики, перегляду рейсів та довідкової інформації. Центральна частина сторінки містить інтерактивну форму для введення даних поля для ZIP–

кодів, введення ставки за милю та кнопки для обчислення маршруту. Після натискання кнопки розрахунку користувач бачить маршрут на карті, загальну довжину поїздки, орієнтовну вартість перевезення та розподіл доходу між компанією, диспетчером і водієм.

Окрім базових функцій, вебзастосунок також передбачає можливість пошуку рейсів за фільтрами. Наприклад, можна обрати конкретного водія або диспетчера, щоб переглянути лише його перевезення, або ж вивести статистику за певний проміжок часу [16]. Це дозволяє більш детально аналізувати ефективність окремих працівників і транспортних засобів. У майбутньому така система може бути доповнена можливістю експорту звітів у форматі PDF або Excel для подальшої обробки чи звітування перед керівництвом компанії.

Таким чином, вебзастосунок повинен забезпечувати зручний, швидкий і точний інструмент для проведення логістичних розрахунків і аналітики. Його головною перевагою є універсальність система підходить як для великих компаній з великим парком автомобілів, так і для малих підприємців, які прагнуть вести облік і аналіз власних перевезень. Завдяки інтеграції з Google Maps API, система гарантує актуальні маршрути, а завдяки розширеним функціям статистики дозволяє компанії бачити повну картину своєї діяльності та приймати ефективні управлінські рішення.

2.2 Основні функції системи

Основні функції вебзастосунку спрямовані на автоматизацію процесів, які зазвичай виконуються вручну під час організації логістичних перевезень. Система дозволяє вводити дані про маршрути, розраховувати відстань та орієнтовну вартість доставки, а також вести аналітику по виконаних рейсах, водіях і диспетчерах. Кожна функція взаємопов’язана з іншими модулями системи, що створює цілісний інструмент для аналітики перевезень.

Після запуску вебзастосунку користувачу доступна головна сторінка з інтерактивною формою введення даних. Вона містить поля для введення ZIP–кодів пункту відправлення, проміжних точок і пункту призначення. Цей етап є основним для подальших розрахунків, оскільки система на основі введених кодів визначає координати точок і передає їх до Google Maps API для побудови маршруту. Інтеграція з API дозволяє автоматично враховувати реальну дорожню мережу, наявні автошляхи, об’їзди та навіть тимчасові перекриття доріг.

Після отримання координат система виконує розрахунок найкоротшого маршруту між заданими пунктами [17-18]. Цей процес здійснюється автоматично, без потреби втручання користувача. Результатом є візуалізація маршруту на карті та обчислення загальної довжини поїздки в милях або кілометрах. Додатково користувач отримує змогу бачити альтернативні варіанти маршруту, якщо вони є доступними, що може бути корисним для вибору оптимального шляху залежно від часу чи стану доріг.

Коли відстань визначено, користувач може ввести базову ставку за милю або кілометр, яка використовується для обчислення вартості перевезення. Після натискання кнопки «Розрахувати», система визначає орієнтовну вартість рейсу та показує деталізований розподіл прибутку між усіма учасниками процесу: компанією, диспетчером та водієм. Цей підхід дозволяє одразу оцінити фінансову ефективність кожного маршруту, що є надзвичайно корисним для аналітики доходів і витрат підприємства.

Однією з важливих функцій системи є можливість збереження кожного розрахованого рейсу у внутрішній базі даних [19]. Після підтвердження користувачем усіх параметрів поїздки інформація зберігається з прив’язкою до конкретної дати, водія, транспортного засобу та диспетчера. Це дає змогу формувати індивідуальну статистику для кожного з них, а також загальну аналітику по компанії. У подальшому накопичені дані використовуються для побудови графіків, що відображають динаміку виконаних перевезень, доходів та кількості рейсів за обраний період.

Іншою ключовою функцією є перегляд статистики. У цьому розділі користувач може бачити узагальнені дані про всі виконані рейси. Інформація може відображатися у вигляді таблиць або інтерактивних графіків, які демонструють співвідношення доходів, витрат, середніх відстаней, кількості рейсів та продуктивності працівників. Наприклад, можна порівняти результати роботи різних водіїв або диспетчерів, визначити, хто з них має найвищий коефіцієнт виконання рейсів або найкращий середній прибуток. Такий підхід дозволяє керівництву компанії приймати обґрунтовані управлінські рішення на основі реальних даних [20-21].

Для зручності користувача вебзастосунок має функцію фільтрації даних. Це означає, що можна вивести лише ті рейси, які відповідають певним критеріям наприклад, за конкретним водієм, траком, диспетчером або часовим проміжком. Завдяки цьому аналітика стає більш гнучкою та орієнтованою на потреби користувача. Крім того, передбачено можливість сортування даних за прибутком, відстанню або датою виконання рейсу.

Ще однією важливою частиною системи є візуалізація маршрутів на карті. Користувач може не лише побачити маршрут нового перевезення, але й переглянути історичні маршрути, які вже були виконані. Це дозволяє оцінювати географічне покриття діяльності компанії та виявляти напрямки, де виконано найбільше перевезень. У майбутньому це може бути використано для побудови теплових карт транспортних потоків або планування нових логістичних маршрутів.

Таким чином, основні функції вебзастосунку об’єднують три головні напрями роботи: розрахунок маршрутів і вартості, ведення статистики перевезень і візуалізацію результатів у зручному форматі. Завдяки цьому вебзастосунок стає універсальним інструментом для щоденного використання у сфері логістики, забезпечуючи точність розрахунків, зручність аналізу та ефективність управління транспортними процесами.

2.3 Архітектура системи

Архітектура вебзастосунку побудована за клієнт-серверною моделлю, де користувацький інтерфейс забезпечує взаємодію користувача із системою, а серверна частина відповідає за виконання логічних обчислень, обробку запитів до бази даних та інтеграцію із зовнішніми сервісами. Такий підхід забезпечує гнучкість, масштабованість і надійність роботи системи, а також дозволяє без проблем доповнювати її новими функціями в майбутньому [22-23].

Користувацький інтерфейс реалізовано як вебсторінку, що містить інтерактивні елементи для введення даних про маршрут, ставки за милю, додаткові витрати, а також інформацію про водія, диспетчера та транспортний засіб. Після введення користувачем вихідних даних відбувається передача інформації на сервер, де обчислюється відстань між заданими пунктами за допомогою інтеграції з Google Maps API. Серверна частина отримує від Google Maps дані про відстань і час у дорозі, після чого використовує ці дані для розрахунку вартості перевезення.

В основі системи лежить модуль фінансових розрахунків, який є ключовим компонентом програми. Саме він виконує визначення вартості рейсу, розподіл прибутку між усіма сторонами, а також облік додаткових витрат. Розрахунок проводиться на основі таких базових параметрів:

Водій отримує фіксовану оплату за милю, яка коливається в межах 0,68–0,80 долара. Ця ставка є однією з найважливіших змінних у розрахунку, оскільки вона прямо впливає на собівартість перевезення. Диспетчер, у свою чергу, має фіксовану місячну ставку і додатковий бонус у розмірі 1% від загального гросу всіх перевезень тих траків, за які він відповідає. Таким чином, у системі формується логічний зв’язок між кількістю виконаних рейсів, загальною виручкою компанії та винагородою диспетчера.

Компанія отримує залишковий прибуток, який розраховується як різниця між сумою, сплаченою клієнтом за перевезення, та сумою всіх витрат. Окрім

основних ставок, у системі враховуються додаткові показники, які часто зустрічаються в логістиці США [24].

DT – оплата простою, яка нараховується у випадку затримки водія на місці завантаження або розвантаження. Її середнє значення складає близько 15 доларів за годину простою.

Layover – оплата за вимушене очікування протягом ночі через відсутність наступного завантаження, зазвичай становить 150 доларів.

Lumper – оплата за послуги розвантаження або навантаження, що може варіюватися залежно від складу або умов виконання робіт.

Driver Assist – допомога водія під час завантаження чи розвантаження, вартість якої визначається індивідуально менеджером і може коливатися від 50 до 200 доларів.

TONU – компенсація за випадок, коли вантаж було замовлено, але рейс скасовано. Середня виплата становить від 100 до 200 доларів.

Усі ці показники враховуються під час розрахунку кінцевої вартості рейсу. Наприклад, якщо під час виконання поїздки водій простоює декілька годин або залишається на місці ночівлі, система додає відповідну суму до загальних витрат і автоматично перераховує чистий прибуток компанії. Це дозволяє отримувати реалістичну картину фінансового стану кожного перевезення, враховуючи всі можливі фактори.

Отримані результати обчислень зберігаються у базі даних, яка містить кілька основних сутностей: маршрути, транспортні засоби, водії, диспетчери та фінансові операції. Кожен новий рейс створює запис із прив’язкою до відповідних осіб та їхніх параметрів. Завдяки цьому система може легко формувати узагальнену статистику, наприклад: сумарний дохід компанії за місяць, кількість виконаних рейсів, середній прибуток на милю, або прибутковість кожного траку [25]. Крім того, база дозволяє відстежувати, який диспетчер відповідав за певні рейси, і скільки відсотків прибутку він отримав.

Архітектурно система побудована таким чином, щоб розділити візуальну частину від логічної частини, яка здійснює обчислення та зберігання даних. Це

дозволяє легко оновлювати зовнішній вигляд застосунку або додавати нові функції без потреби змінювати математичну або логічну основу програми. Застосунок використовує REST API для обміну даними між клієнтською та серверною частинами, що забезпечує швидку реакцію на дії користувача і стабільність у роботі навіть при великій кількості запитів.

Таким чином, архітектура застосунку є чітко структурованою системою, у якій поєднуються аналітична точність, простота використання та гнучкість [26]. Інтеграція з Google Maps API дозволяє отримувати точні дані про маршрути, а розгалужена фінансова логіка проводити комплексні розрахунки вартості, прибутку та ефективності роботи компанії. Завдяки цьому застосунок виступає не лише інструментом для розрахунку перевезень, але й аналітичною платформою для прийняття управлінських рішень у сфері логістики.

2.4 Математична модель розрахунку вартості перевезень

Для забезпечення точності фінансових обчислень у вебзастосунку використовується математична модель, яка описує процес формування вартості кожного перевезення та розподіл прибутку між усіма учасниками [27-28]. Основна мета моделі полягає у визначенні загальної суми доходу, витрат компанії, чистого прибутку та індивідуальних виплат водіям і диспетчерам. Комп’ютерна модель фільтрації зображень

У процесі розрахунку використовуються такі змінні:

V – відстань маршруту в милях, отримана з Google Maps API;

M – ставка оплати за милю;

D– відсоток або фіксована ставка виплати водію;

P – відсоток диспетчера від загального гросу;

S – фіксована ставка диспетчера за місяць;

C – сума додаткових витрат;

G – загальний грос.

Розрахунок вартості перевезення починається з визначення гросу, тобто загального доходу від рейсу. Цей показник залежить від відстані маршруту та ставки за милю і обчислюється як добуток V на M. Таким чином, якщо вантаж проходить 1200 миль при ставці 2,5 долара за милю, то загальний грос становитиме 3000 доларів [29]. Це є базова величина, від якої надалі розраховуються всі інші фінансові показники.

Наступним етапом є обчислення виплати водію. Залежно від домовленостей із компанією, водій може отримувати оплату або за милю, або у вигляді відсотка від загального доходу. Якщо використовується фіксована ставка, то загальна виплата водію визначається як добуток V на D. Наприклад, якщо ставка становить 0,75 долара за милю, а відстань 1200 миль, то водій отримає 900 доларів. У випадку, якщо рейс передбачає додаткові компенсації, ці витрати додаються до суми C, яка враховується у фінальному розрахунку.

Винагорода диспетчера формується з двох частин фіксованої ставки S, яку він отримує щомісяця, та відсотка P від загального гросу всіх рейсів, за які він відповідає. Якщо диспетчер має ставку 900 доларів на місяць і отримує 1% від сукупного доходу у 120 000 доларів, то його місячна винагорода становитиме 2100 доларів [30-31]. Таким чином, диспетчер зацікавлений у збільшенні обсягу перевезень, адже його дохід прямо залежить від результативності роботи закріплених за ним траків.

Додаткові витрати враховують усі непередбачувані ситуації, які можуть вплинути на загальну собівартість рейсу. До них належать простої, добові за затримку, послуги вантажників, допомога водія під час розвантаження або скасування замовлення. Ці витрати підсумовуються у величину C, яка додається до витрат рейсу [32]. Наприклад, якщо водій простоював 3 години, мав Layover на 150 доларів і отримав компенсацію за допомогу під час розвантаження 100 доларів, то загальні додаткові витрати становитимуть 295 доларів.

Після цього визначається чистий прибуток компанії. Він обчислюється як різниця між загальним доходом і всіма витратами, які включають виплати водію, відсоток диспетчера та додаткові витрати. Формально це можна записати як:

$$N = G - (D \times V + (\frac{P}{100} \times G) + \text{C}) \quad (1.1)$$

Якщо грос становить 3000 доларів, виплата водію 900 доларів, диспетчер отримує 1%, а додаткові витрати становлять 295 доларів, то чистий прибуток компанії дорівнюватиме 1775 доларам. Саме цей показник використовується у системі для подальшого аналізу ефективності рейсів.

У межах аналітичного модуля вебзастосунку результати обчислень накопичуються, що дозволяє отримати зведену інформацію за місяць. Система автоматично підраховує сумарний грос, загальні витрати, суму виплат водіям і диспетчерам, а також чистий прибуток компанії [33-34]. На основі цих даних можна формувати графіки та звіти, які показують прибутковість кожного траку, диспетчера або маршруту, а також визначати середній прибуток на милю.

Розроблена математична модель дозволяє системі автоматично, швидко та точно розраховувати фінансові показники для кожного перевезення. Вона забезпечує прозорість усіх операцій, дозволяє оцінювати ефективність роботи водіїв і диспетчерів, а також надає менеджерам компаній необхідні інструменти для прийняття обґрунтованих управлінських рішень.

2.5 Обгрунтування архітектури front-end частини застосунку

Для розробки клієнтської частини застосунку, що забезпечує розрахунок вартості перевезень і пошук найкоротших маршрутів, доцільно використовувати сучасні фреймворки та бібліотеки, які спрощують створення динамічних, інтерактивних і продуктивних інтерфейсів користувача [35]. Основними інструментами, які можуть бути застосовані під час реалізації front-end частини, є «React.js», «Tailwind CSS» та «Google Maps JavaScript API».

React.js дозволяє структурувати застосунок у вигляді окремих компонентів, кожен із яких відповідає за свою частину інтерфейсу наприклад,

форму введення даних, відображення маршруту на карті або секцію статистики. Завдяки підходу компонентної архітектури можливо забезпечити високу гнучкість застосунку, його розширюваність і простоту обслуговування. Tailwind CSS використовується для швидкої побудови адаптивного та мінімалістичного дизайну без потреби створювати складні таблиці стилів вручну. Це дає змогу зосередитись на функціональності й логіці, водночас підтримуючи сучасний, чистий зовнішній вигляд системи.

Основою візуальної частини є інтеграція з Google Maps API, яка забезпечує отримання точної інформації про відстань між пунктами [36]. Користувачеві достатньо ввести ZIP-коди або адреси пункту завантаження та розвантаження, після чого система автоматично побудує маршрут, розрахує відстань та відобразить її на карті.

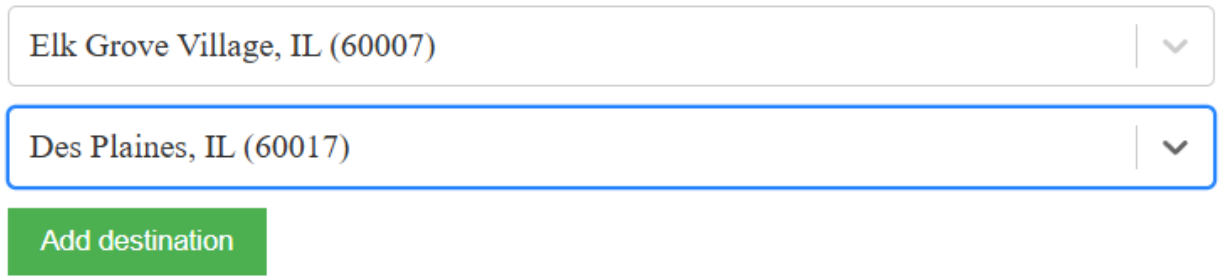

Рисунок 2.1 – Інтерфейс вибору локацій для створення транзиту без додаткових зупинок

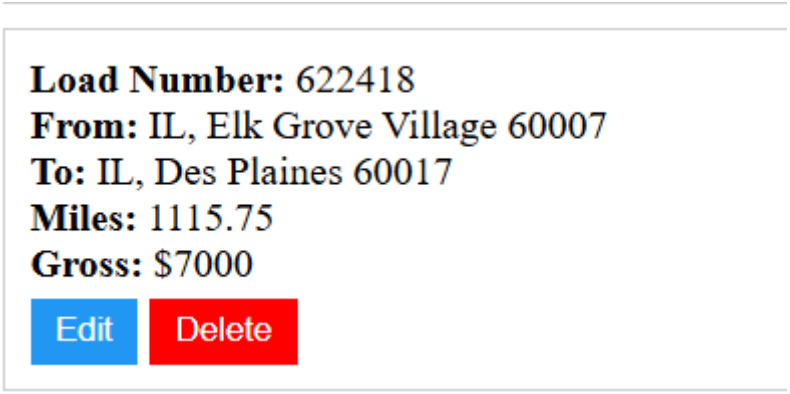

Рисунок 2.2 – Інтерфейс доданого вантажу

Усі дані обробляються у реальному часі, що забезпечує інтерактивність та оперативність розрахунків (рис. 2.1).

Інтерфейс застосунку матиме просту, логічну структуру (рис. 2.2). На головній сторінці користувач побачить форму введення маршрутних даних. Передбачено два основних поля для введення початкової та кінцевої точки маршруту. Після натискання кнопки «Розрахувати маршрут» система відправляє запит до Google Maps API, отримує оптимальний шлях і виводить його на карту, розташовану нижче форми введення [37]. Одночасно у правій частині сторінки з'являється панель результатів, де користувач бачить такі показники: загальну відстань маршруту, орієнтовну вартість перевезення, виплати водієві та диспетчеру, а також прибуток компанії.

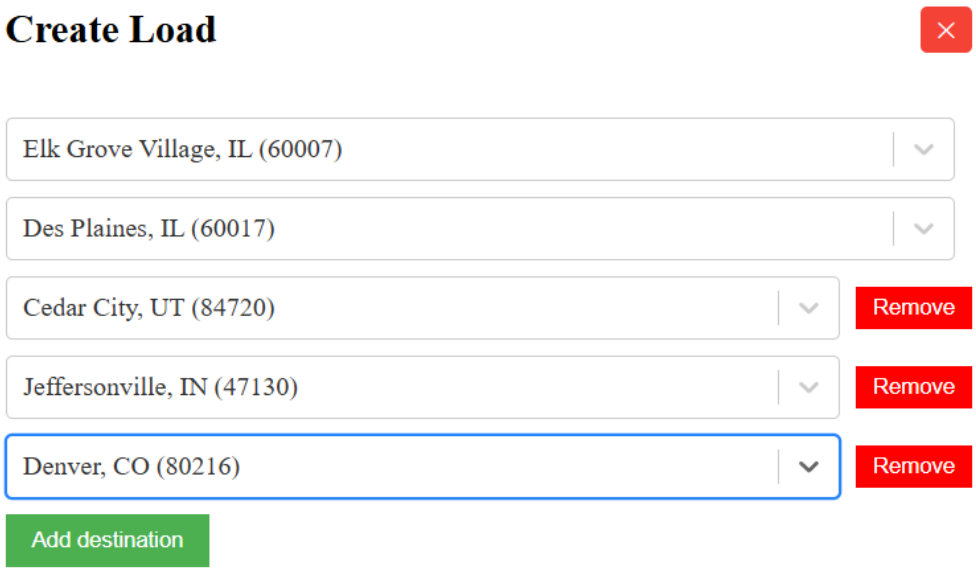

Рисунок 2.3 – Інтерфейс вибору локацій для створення транзиту з додатковими зупинками

Користувач може також додати проміжні точки маршруту, якщо перевезення включає кілька зупинок або завантажень (рис. 2.3). Після введення таких точок система автоматично оновлює маршрут, перераховує відстань і коригує фінансові результати. Всі оновлення відбуваються динамічно без перезавантаження сторінки, що створює зручність у користуванні.

Після розрахунку перевезення користувач може зберегти його результати у базу даних для подальшого аналізу (рис. 2.3). При цьому зберігається уся інформація відстань, кількість пунктів, виплати водію, прибуток диспетчера, загальний грос, а також ідентифікатор траку та водія, який виконував цей рейс. Надалі ці дані використовуються у модулі статистики.

Сторінка «Статистика» забезпечує можливість перегляду та аналізу виконаних перевезень. Тут користувач може обрати період і переглянути загальну кількість рейсів, сумарний прибуток компанії, середню ставку за милю, суму бонусів диспетчера та загальну ефективність усіх траків [38-39]. Для візуалізації використовується бібліотека Recharts, яка дозволяє будувати інтерактивні графіки лінійні, кругові або гістограми. Наприклад, на графіку може бути показано зміну доходу за тижнями або порівняння прибутковості різних траків.

Інтерфейс буде створено в мінімалістичному стилі з переважанням світлих відтінків і контрастних акцентів для важливих елементів. Верхня частина сторінки міститиме навігаційну панель із логотипом компанії, кнопками переходу до сторінок «Головна», «Статистика». У нижній частині сторінки буде футер з контактною інформацією, посиланням на документацію API та короткою довідкою щодо використання сервісу.

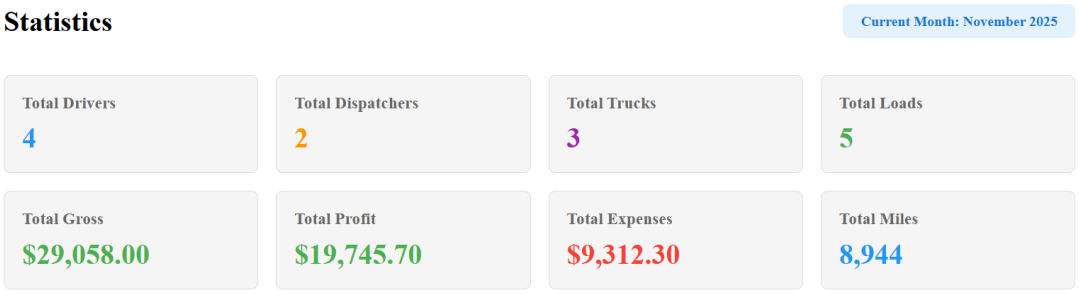

Рисунок 2.4 – Інтерфейс найважливішої статистики

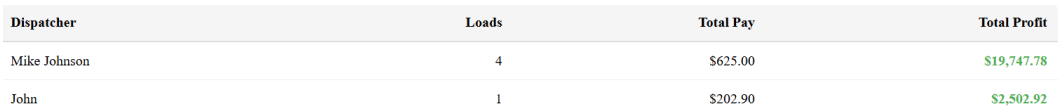

Рисунок 2.5 – Інтерфейс статистики диспетчерів

Driver	Loads	Total Miles	Total Pay	Total Profit
Alex	2	5,499	$3,574.15	$14,155.85
John Smith	2	1,755	$1,053.07	$3,086.93
Adam	1	1,690	$1,352.18	$2,502.92

Рисунок 2.6 – Інтерфейс статистики водіїв

У вебзастосунку буде реалізовано повноцінний модуль статистики, який забезпечуватиме детальний аналіз логістичних даних [40]. Система автоматично формуватиме статистику для водіїв, диспетчерів та транспортних засобів, а також надаватиме загальну агреговану статистику за весь період роботи компанії (рис. 2.4). Для водіїв програма показуватиме кількість виконаних рейсів (рис 2.6), сумарні милі, загальну виплату та чистий прибуток кожного водія, формуючи рейтинг найбільш ефективних працівників. Для диспетчерів відображатимуться дані щодо кількості закритих вантажів, отриманих комісій та загального фінансового внеску кожного диспетчера в роботу компанії (рис. 2.5).

Окремо система автоматично рахуватиме загальну статистику: валовий дохід, середній прибуток на рейс, середній gross, сумарні витрати на оплату водіїв і диспетчерів, а також інші ключові показники ефективності логістики. Цей модуль дозволить швидко оцінити поточний стан компанії, порівнювати результативність працівників та приймати більш обґрунтовані управлінські рішення на основі фактичних даних.

Рисунок 2.7 – Інтерфейс загальної статистки

Взаємодія користувача із застосунком відбуватиметься максимально інтуїтивно. Після введення маршрутних даних користувач одразу отримає візуальне підтвердження на карті з’являється лінія маршруту, а в панелі праворуч оновлюються всі фінансові показники [41-42]. Якщо користувач змінює будь–яку точку, система миттєво перераховує всі значення. Завдяки цьому застосунок

працює як інтерактивний калькулятор і водночас аналітичний інструмент для логістичних компаній (рис. 2.7).

Для підвищення ефективності роботи застосунок може зберігати історію останніх введених маршрутів, що дозволяє швидко повторювати типові перевезення без повторного введення даних.

Таким чином, архітектура front-end частини застосунку побудована за принципами модульності, інтерактивності та зручності користування. Використання React.js і «Google Maps» API дозволяє реалізувати динамічний інтерфейс, який не тільки забезпечує швидке введення та обробку даних, але й робить систему зрозумілою навіть для користувачів без технічної підготовки. Застосунок стане ефективним інструментом для аналітики, розрахунку перевезень і прийняття логістичних рішень у реальному часі.

2.6 Обгрунтування архітектури back-end частини застосунку

Back-end частина застосунку виконує ключову роль у забезпеченні всіх обчислювальних процесів, обробці даних, логіці бізнесу та взаємодії з базою даних. Саме вона відповідає за отримання запитів від користувача, проведення необхідних розрахунків, звернення до зовнішніх API та формування відповідей у зручному форматі [43]. Для розробки серверної частини найдоцільніше обрати «FastAPI», адже цей фреймворк забезпечує високу швидкість обробки запитів, просту структуру коду та підтримку асинхронних викликів. Завдяки «FastAPI» сервер може швидко опрацьовувати численні одночасні запити від користувачів, що є особливо важливим у випадку роботи з великою кількістю маршрутів, транспортних засобів та користувачів системи.

Основна логіка back-end полягає у прийманні вхідних даних від користувача, які містять поштові коди пунктів відправлення, транзиту та призначення. Після отримання цих даних сервер звертається до Google Maps API, який повертає відстань між заданими точками. Отримане значення обробляється на сервері, де проводяться математичні розрахунки на основі заданих формул,

що визначають загальний грос, виплати водію, диспетчеру та чистий прибуток компанії. Усі ці обчислення зберігаються в базі даних, що дозволяє формувати подальшу статистику, створювати графіки та порівнювати ефективність роботи різних учасників транспортного процесу.

Для зберігання даних у застосунку доцільно використовувати «PostgreSQL», як одну з найнадійніших реляційних систем управління базами даних. «PostgreSQL» дозволяє створювати чітко структуровані таблиці для кожного типу об’єктів водіїв, диспетчерів, траків, маршрутів та операцій. Наприклад, таблиця водіїв міститиме унікальний ідентифікатор, ім’я, кількість виконаних рейсів та суму заробітку [44]. Таблиця маршрутів включатиме дані про початкову та кінцеву точку, відстань, дату перевезення та всі розраховані фінансові показники. Це забезпечить можливість швидкого пошуку, фільтрації та агрегування інформації, що особливо важливо при створенні статистичних звітів або графіків.

Комунікація між front-end і back-end здійснюється через REST API. Користувацький інтерфейс надсилає HTTP-запити до певних end-points сервера, які виконують окремі функції. Наприклад, при введенні користувачем двох або кількох ZIP-кодів front-end надсилає запит /calculate-route, який активує відповідний метод на back-end. Сервер виконує виклик до Google Maps API, отримує відстань, проводить розрахунки та повертає результат у форматі JSON. Інший end–point, наприклад /driver-stats, може використовуватись для отримання детальної статистики по конкретному водію, включно з кількістю поїздок, пройденими милями, витратами та заробітком.

Для спрощення адміністрування та можливості перевірки результатів під час тестування можна також додати простий інтерфейс або API-документацію, автоматично згенеровану самим «FastAPI». Це дозволить переглядати доступні маршрути запитів, тестувати функціональність та перевіряти коректність роботи програми без необхідності створення додаткового клієнтського коду.

У майбутньому система може бути розширена модулем авторизації, який дозволить ідентифікувати користувачів та розподіляти доступ до різних рівнів

даних. Наприклад, менеджер може бачити фінансову статистику по всій компанії, тоді як водій матиме доступ лише до власних результатів. Також можливе підключення модулю аналітики, який автоматично формуватиме звіти про найефективніших водіїв або про середню вартість милі за певний період.

Для обробки складних розрахунків, таких як оптимізація маршруту або прогнозування прибутку за певних умов, back-end може використовувати бібліотеки на основі NumPy або Pandas [45]. Вони дозволять працювати з великими масивами даних, обчислювати середні показники, виявляти закономірності та будувати прогнози. Наприклад, «Pandas» може бути використана для аналізу минулих маршрутів та визначення найефективніших напрямків, а також для виявлення факторів, які впливають на витрати.

У результаті така архітектура back-end частини забезпечує високу швидкість, масштабованість та гнучкість системи. Вона дозволяє легко розширювати функціонал, додавати нові типи обчислень або звітів без необхідності змінювати основну структуру. Завдяки використанню «FastAPI», «PostgreSQL» і Google Maps API застосунок стає потужним інструментом для аналізу логістичних процесів, що здатен ефективно працювати навіть з великими обсягами даних і підтримувати потреби сучасної транспортної компанії.

2.7 Збереження та обробка даних у застосунку

Back-end Збереження даних є одним із ключових аспектів у роботі будь-якого сучасного застосунку, особливо коли йдеться про логістичну систему, яка має обробляти численні маршрути, транзакції, інформацію про водіїв, диспетчерів та транспортні засоби. Надійне зберігання забезпечує не лише стабільність роботи застосунку, але й точність фінансових розрахунків, коректність статистики та можливість формувати аналітичні звіти. Для створення ефективної системи збереження інформації необхідно обрати структуру, яка буде водночас продуктивною, безпечною та гнучкою у розширенні.

Основним варіантом, який найкраще підходить для даного застосунку, є використання реляційної бази даних «PostgreSQL». Ця система управління базами даних відома своєю надійністю, масштабованістю та підтримкою складних запитів [46]. «PostgreSQL» дозволяє створювати чітко структуровані таблиці, встановлювати між ними зв’язки та забезпечує підтримку транзакцій, що гарантує збереження цілісності даних навіть при одночасному доступі кількох користувачів. У структурі бази даних можна виділити кілька основних таблиць: таблиця водіїв, де зберігатимуться особисті дані, кількість рейсів, заробіток та загальна пройдена відстань; таблиця диспетчерів, яка міститиме інформацію про ставку, відсоток від гросу та кількість траків, закріплених за ними; таблиця траків, що описує кожен транспортний засіб, його власника та поточний стан; таблиця маршрутів, у якій зберігатиметься історія всіх перевезень, включаючи відстань, пункти відправлення та призначення, час виконання, суму гросу, а також усі додаткові витрати.

Важливою частиною є організація зв’язків між таблицями. Один водій може бути прив’язаний до одного траку, але один диспетчер може керувати кількома траками одночасно. Така структура дозволяє створити відношення типу «один-до-багатьох», що реалізується через зовнішні ключі. Це забезпечує гнучкість при формуванні статистики – наприклад, можна швидко отримати сумарний грос по всіх траках, закріплених за певним диспетчером, або розрахувати середню кількість миль, яку проїжджає водій за місяць.

Для обробки великої кількості обчислень, наприклад під час аналізу даних чи створення звітів, може бути реалізована кеш-пам’ять за допомогою «Redis» або внутрішнього механізму «FastAPI». Це дозволить тимчасово зберігати результати частих запитів, наприклад відстані між популярними пунктами чи середні витрати на маршрути певного типу. Таким чином, повторні запити користувача оброблятимуться значно швидше, знижуючи навантаження на основну базу даних.

Додатковим варіантом збереження інформації може бути використання хмарних сховищ – наприклад, «Amazon RDS» або «Google Cloud SQL». Такі

сервіси забезпечують високу доступність, автоматичне резервне копіювання та зручне масштабування при збільшенні кількості користувачів або обсягу даних. У разі розгортання застосунку в комерційних цілях, це дозволить уникнути ризику втрати інформації через технічні збої та забезпечить стабільну роботу навіть при великих навантаженнях.

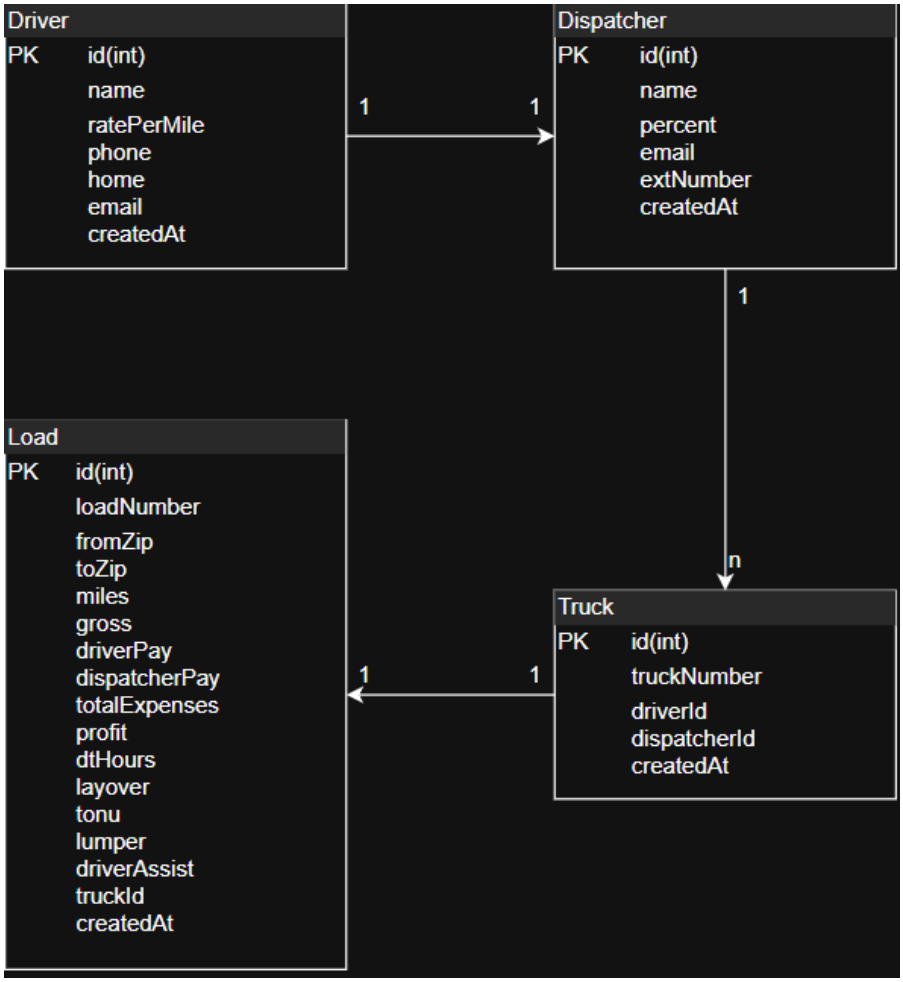

Рисунок 2.8 – UML-діаграма.

Особливу увагу слід приділити резервному копіюванню. У системі має бути передбачено автоматичне створення резервних копій бази даних з певною періодичністю, наприклад щодня або щотижня [47]. Це дозволить у разі збою або помилки швидко відновити всі критично важливі дані, не порушуючи цілісності інформації. Також бажано реалізувати журнал транзакцій, який зберігатиме історію змін у базі, щоб при необхідності можна було відстежити кожну операцію, проведену користувачем чи системою.

Для захисту даних під час передачі між клієнтською частиною та сервером використовується протокол HTTPS, що забезпечує шифрування трафіку. Крім того, back-end може мати внутрішню систему перевірки запитів, щоб уникнути несанкціонованого доступу до бази даних. Якщо в майбутньому буде реалізовано розділення ролей, то доцільно додати систему аутентифікації та авторизації, яка дозволить розмежувати доступ до певних елементів інформації.

Для більшої аналітичної гнучкості може бути створений окремий модуль архіву даних, куди переміщуватимуться старі маршрути та історія виплат. Це не лише зменшить навантаження на основну базу, але й дозволить швидко формувати звіти за попередні періоди наприклад, для аналізу прибутковості за рік або оцінки ефективності диспетчерів.

Отже, найоптимальнішою архітектурою збереження даних для цього застосунку є поєднання реляційної бази «PostgreSQL», кешування через «Redis» та, за потреби, резервного хмарного сховища. Такий підхід забезпечить баланс між швидкістю, безпекою та масштабованістю системи (рис. 2.8). Це створить надійну основу для подальшого розвитку програмного забезпечення, його інтеграції з іншими сервісами та розширення функціональних можливостей без втрати стабільності роботи.

3 КОМП'ЮТЕРНА СИСТЕМА ОБЛІКУ РЕЙСІВ ТА ФІНАНСОВОЇ АЛАЛІТИКИ ПЕРЕВЕЗЕНЬ

3.1 Обґрунтування вибору середовища програмної реалізації

Для розробки логістичного вебзастосунку було проаналізовано низку можливих технологій, які могли б бути застосовані як у серверній, так і у клієнтській частині системи. Під час вибору враховувалися такі критерії: продуктивність, масштабованість, простота розробки, наявність активної екосистеми, безпека, можливість інтеграції із зовнішніми сервісами та зручність довгострокової підтримки. Серед можливих варіантів серверних рішень розглядалися такі інструменти, як «Django та Flask», «Spring Boot», «FastAPI» і навіть serverless-платформи на базі «AWS Lambda». Незважаючи на широкий спектр можливостей цих технологій, остаточний вибір було зроблено на користь «Node.js» із використанням фреймворку «Express».

Для серверної частини логістичного вебзастосунку було обрано платформу «Node.js» у поєднанні з вебфреймворком «Express.js». На відміну від важчих корпоративних рішень, таких як «Spring Boot» чи «.NET Core», «Node.js» забезпечує значно швидший старт проекту, простішу конфігурацію та мінімальні вимоги до інфраструктури. Його подієва модель з неблокуючим ввод/виводом є особливо ефективною для вебзастосунків, які активно взаємодіють із мережею, зовнішніми API та базами даних. Оскільки логістична система інтенсивно використовує запити до «Google Maps API», виконує обробку великої кількості HTTP-запитів і постійно працює з даними про рейси, траки та користувачів, асинхронна природа «Node.js» ідеально підійшла для таких завдань.

«Express.js» було обрано як основний фреймворк завдяки його мінімалістичності та гнучкості. На відміну від фреймворків, що нав'язують жорстку структуру застосунку, «Express» дозволяє формувати архітектуру відповідно до специфічних потреб системи. Це було важливо, оскільки логістичний вебзастосунок містить власні бізнес-процеси: розрахунок вартості

рейсу, визначення винагороди водіїв і диспетчерів, роботу з витратами, а також механізми обліку всіх перевезень. «Express» не створює надлишкової обгортки навколо цих процесів і не обмежує розробника у виборі структури або способу реалізації «REST API».

Серед ключових переваг «Express.js» – простота визначення маршрутів, підтримка middleware, легкість інтеграції з будь-якими базами даних, відсутність зайвих абстракцій та широка екосистема модулів. У межах проєкту це дозволило швидко реалізувати повноцінний набір серверних функцій: додавання водіїв, диспетчерів і траків, створення рейсів, обробку розрахунків, збереження даних у базі та формування запитів для статистики. «Express» чудово поєднується з «Prisma», забезпечуючи чисту, зрозумілу та передбачувану взаємодію між API та базою даних.

Ще однією перевагою є легкість розширення. Завдяки гнучкості Express можна без ускладнень додавати нові ендпоїнти, інтегрувати додаткові сервіси, виконувати обробку помилок або створювати власні middleware для логування, авторизації чи аналітики. Це робить «Express» оптимальним вибором для логістичної системи, яка потребує можливості подальшого масштабування функціоналу.

Таким чином, вибір «Express.js» зумовлений поєднанням швидкодії, простоти, розширюваності та відсутності зайвих обмежень. Цей фреймворк став основою для створення гнучкого «REST API», який забезпечує ефективну взаємодію між клієнтською частиною, базою даних і зовнішніми сервісами, що є критично важливим для коректного функціонування логістичного вебзастосунку.

Для розробки логістичного вебзастосунку особливу увагу було приділено вибору формату та рушія бази даних, оскільки структура системи включає чітко визначені взаємозв’язки між сутностями, такими як водії, траки, диспетчери та рейси. Ці елементи логістичного процесу природно формують реляційну модель даних, у якій сутності мають стійкі зв’язки типу «один до одного» та «один до багатьох». Саме тому для зберігання та обробки інформації перевага була надана

реляційним системам управління базами даних, які гарантують цілісність даних, підтримку транзакцій та можливість виконання складних багаторівневих запитів. Документоорієнтовані бази даних, такі як «MongoDB» чи «Firestore», також розглядалися як потенційні варіанти, проте їхня слабкоструктурована природа не забезпечує такого рівня передбачуваності й строгості, що необхідно для логістичних моделей із жорсткими зв’язками між таблицями.

На етапі активної розробки було прийнято рішення використовувати «SQLite» як основну реляційну базу даних. «SQLite» є легкою, вбудованою файловою СУБД, яка не потребує окремого серверного процесу чи складної конфігурації, що робить її надзвичайно зручною для швидкого прототипування, тестування й роботи в локальному середовищі. Завдяки своїй простоті «SQLite» дозволяє розробнику повністю зосередитися на формуванні бізнес-логіки, не витрачаючи час на розгортання інфраструктури. Вона забезпечує підтримку повноцінного SQL-синтаксису, транзакційну цілісність та високу стабільність виконання запитів, що є важливими характеристиками для коректного моделювання реальної поведінки промислових реляційних систем.

Однією з ключових переваг «SQLite» є портативність: усі дані зберігаються в одному файлі, що значно спрощує роботу з тестовими наборами, створення резервних копій, відкат до попередніх станів і зміну структури бази даних. Такий підхід особливо корисний у проєктах, де необхідно регулярно коригувати схему або експериментувати з логістичними моделями. «SQLite» дозволяє швидко перевіряти логіку зв’язків між водіями, траками та рейсами, тестувати розрахунки оплати за милю, валового доходу, детеншну, лейоверу та інших елементів витрат, а також будувати коректні фінансові моделі всередині системи.

Поряд із «SQLite» розглядалися також альтернативи – «PostgreSQL» як високопродуктивна промислова «СУБД», «MySQL» та «MariaDB» як масштабовані серверні рішення. Хоча ці системи демонструють кращі можливості при роботі з великими обсягами даних і значною кількістю одночасних підключень, вони вимагають додаткових ресурсів, інсталяції та

налаштування, що ускладнює ранні етапи розробки. Система була спроєктована таким чином, щоб у будь-який момент можна було виконати прозору міграцію з «SQLite» на «PostgreSQL», коли застосунок буде готовий до промислового розгортання. «PostgreSQL» забезпечує розширену підтримку складних запитів, потужні механізми оптимізації, розвинену систему ролей і прав доступу, а також стабільність при високих навантаженнях, що робить її оптимальним майбутнім варіантом.

Отже, використання «SQLite» на етапі проєктування й тестування стало логічним і ефективним рішенням, оскільки ця «СУБД» поєднує простоту, швидкість налаштування, мінімальне використання ресурсів і повну сумісність із реляційною моделлю даних. Водночас архітектура системи залишається масштабованою та готовою до переходу на «PostgreSQL» у момент, коли це стане необхідним у рамках реального промислового застосування. Завдяки такому підходу вдалося створити стабільне середовище для розробки, забезпечити гнучкість у роботі з даними та зберегти можливість подальшого розширення системи без кардинальних змін у її структурі.

Для логістичного вебзастосунку важливим етапом став вибір інструментів для роботи з базою даних. Після порівняння таких «ORM», як «Sequelize», «TypeORM» та «Knex.js», оптимальним рішенням було визначено Prisma завдяки її структурованості, типізації та надійній системі міграцій. Prisma використовує декларативний опис моделі даних і генерує типізований клієнт, що суттєво зменшує кількість помилок при роботі з фінансовими розрахунками, де точність є критично важливою. Це робить її особливо корисною для логістики, де всі дані водії, траки, рейси пов’язані між собою та потребують гарантованої цілісності.

«Prisma» забезпечує просту й зрозумілу структуру схеми, дозволяючи описати зв’язки «водій-трак-рейси» в компактному декларативному форматі. Автоматичні міграції спрощують розвиток системи: будь-яка зміна моделі відображається у вигляді коректних SQL-скриптів, що забезпечує стабільність навіть під час активної розробки. Вбудована можливість отримувати вкладені

дані в одному запиті значно спрощує створення модулів статистики та фінансової аналітики.

Для етапу розробки була обрана «SQLite» – легка, автономна реляційна база даних, що не потребує окремого сервера та дозволяє зберігати всю структуру у вигляді одного файлу. Це робить її оптимальною для тестування логіки обліку рейсів, розрахунку гросу, виплат водіям і диспетчерам, а також для швидкої роботи з тестовими даними. «SQLite» забезпечує ACID-транзакційність і підтримку повноцінного «SQL», що дозволяє моделювати поведінку промислових СУБД.

Завдяки поєднанню «SQLite» і «Prisma» під час розробки вдалося створити повноцінну реляційну модель, протестувати фінансові алгоритми й одночасно зберегти можливість масштабування системи під реальне промислове навантаження. Таким чином, «Prisma» була обрана за поєднання простоти, надійності, типізації, гнучкості та зручності роботи з реляційними структурами, що робить її оптимальним рішенням для логістичного вебзастосунку.

Клієнтська частина логістичного вебзастосунку була реалізована з використанням «Reac»t, який було обрано серед можливих альтернатив, таких як «Angular» або «Vue.js». «React» забезпечує компонентний підхід, завдяки якому інтерфейс можна поділити на незалежні логічні блоки – сторінки з водіями, диспетчерами, траками, формами створення рейсів та аналітичними панелями. Така архітектура суттєво спрощує підтримку й розширення системи, що важливо у логістичних застосунках, де структура даних постійно змінюється. Крім того, «React» має одну з найбільших екосистем та спільнот, що дає змогу швидко інтегрувати бібліотеки для маршрутизації, форм, роботи з API та елементами інтерфейсу.

«Angular» і «Vue.js» також могли бути використані, проте «Angular» має вищий поріг входження та складнішу структуру, а «Vue.js» поступається «React» за кількістю інструментів і готових рішень. «React» став оптимальним вибором через гнучкість, стабільність і зручність роботи з великими динамічними списками даних, які характерні для логістичних систем.

Для збірки та запуску клієнтської частини було обрано «Vite» – сучасний інструмент, що забезпечує значно швидший запуск та оновлення застосунку під час розробки. На відміну від «Webpack» або «Parcel», «Vite» не виконує повну попередню збірку, а використовує нативне підключення модулів, що забезпечує миттєвий запуск дев-сервера та швидке оновлення змінених компонентів. Це особливо корисно під час роботи з формами, сторінками створення рейсів або таблицями, де дані змінюються майже постійно.

«Vite» також оптимізує продакшн-збірку, забезпечує мінімальний розмір фінального коду та швидке завантаження інтерфейсу. Завдяки поєднанню «React» та «Vite» вдалося створити інтерфейс, який працює швидко, підтримує сучасні стандарти «JavaScript», легко взаємодіє з «REST API», динамічно оновлює дані та забезпечує стабільний користувацький досвід у логістичному середовищі.

У процесі розробки логістичного вебзастосунку одним із ключових елементів інфраструктури стали сервіси «Google Maps API», які забезпечують точне та надійне отримання географічних даних, необхідних для побудови маршрутів, обчислення відстаней і визначення геолокації за ZIP-кодом. Розглядалися альтернативи, такі як «OpenStreetMap» у комбінації з «OSRM» або «GraphHopper», а також «Here Maps», однак саме «Google Maps API» продемонстрував найвищу стабільність, точність і якість покриття для території США – основної зони роботи логістичних компаній. Це стало важливим фактором, оскільки похибка навіть у декілька миль може призвести до неправильних розрахунків оплати, прибутку та планування завантаження.

У проєкті використовуються три основні сервіси: «Directions API», «Distance Matrix API» та «Geocoding API». «Directions API» забезпечує побудову оптимізованого маршруту між пунктами завантаження та доставки, враховуючи реальні дорожні умови, можливі обмеження руху та структуру транспортної мережі. «Distance Matrix API» дає можливість швидко отримати точну кількість миль між будь-якими точками, що є основою для розрахунку платежів: оплати за милю, валового доходу, планування рентабельності та W2-нарахувань.

«Geocoding API» використовується для перетворення ZIP-кодів у координати й назад, що дозволяє системі приймати ввід як у вигляді міста, так і у вигляді поштового індексу. Це значно спрощує роботу користувача і забезпечує гнучкість при заповненні інформації про рейс.

Перевагою «Google Maps API» є не лише точність даних, а й стабільність роботи, висока швидкість відповіді, надійність транспортних даних та добре структурована документація. Для логістичного програмного забезпечення особливо важливо, що дані «Google» регулярно оновлюються, що гарантує коректні розрахунки навіть у випадках зміни дорожніх маршрутів, ремонту доріг або перекриттів. Крім того, «Google Maps API» має прозору систему тарифікації, що дозволяє точно прорахувати витрати на сторонні сервіси та масштабувати застосунок у майбутньому без ризику непередбачуваних витрат.

Використання «Google Maps API» дає можливість повністю автоматизувати логістичні обчислення та прибрати необхідність ручного пошуку відстаней за сторонніми сервісами. Це скорочує час на оформлення рейсу та мінімізує ризик людської помилки. API працює через стандартні HTTP-запити, тому легко інтегрується з «Express.js» на сервері та може масштабуватись у більш складні системи, якщо компанія плануватиме впровадження автоматичної оптимізації завантаження, мульти-стоп маршрутів чи GPS-трекінгу в реальному часі.

Загалом використання «Google Maps API» стало одним із ключових рішень, яке дозволило створити точний, надійний і гнучкий механізм розрахунку логістичних параметрів. Сервіси Google забезпечують високий рівень точності даних, підтримують широкі можливості інтеграції та дозволяють проєкту розвиватися у напрямках автоматизації планування маршрутів, розрахунку витрат і впровадження аналітики на основі реальних геолокаційних даних.

У результаті обраний стек технологій – «Express», «Prisma», реляційна база даних, «React», «Vite» та картографічний сервіс «Google Maps» – було визначено як оптимальне рішення для створення продуктивного, розширюваного й надійного логістичного вебзастосунку. Він відповідає вимогам системи,

забезпечує простоту майбутньої підтримки, дозволяє легко інтегрувати нові модулі і масштабувати проєкт відповідно до потреб користувачів.

3.2 Обґрунтування вибору середовища програмної реалізації

Модуль керування водіями є одним із ключових компонентів логістичного вебзастосунку, оскільки забезпечує зберігання та обробку даних про водіїв, їх тарифні ставки, контактну інформацію та взаємозв’язки з транспортними засобами. Реалізація цього модуля складається з двох частин: серверної логіки, відповідальної за роботу з базою даних через Prisma ORM, та клієнтської частини, реалізованої у вигляді React-компонентів, що забезпечують взаємодію користувача зі системою. Обидві частини побудовані таким чином, щоб забезпечити прозорість логіки, надійність роботи та можливість подальшого масштабування.

На серверному рівні модуль реалізований у вигляді контролера, що опрацьовує стандартні CRUD-операції: створення, отримання, оновлення та видалення водіїв. Контролер використовує Prisma Client для виконання типізованих запитів до бази даних, що виключає помилки, пов’язані з неправильними типами даних або некоректними параметрами. Обробка помилок реалізована централізовано, що дозволяє контролеру повертати передбачувані JSON-відповіді. Нижче наведено приклад серверної логіки обробки запитів до водіїв, який демонструє типову взаємодію клієнтської частини з REST API, включаючи використання Prisma для створення нового водія, оновлення його параметрів або видалення запису з бази даних:

Лістинг 3.1 Контролер для роботи з водіями:

```
export const getDrivers = async (req, res) => {
  try {
    const drivers = await prisma.driver.findMany();
```

```
    res.json(drivers);
  } catch (error) {
    console.error("Error fetching drivers:", error);
    res.status(500).json({ error: "Failed to fetch drivers" });
  }
};

export const createDriver = async (req, res) => {
  try {
    const { name, ratePerMile, phone, home, email } = req.body;

    const newDriver = await prisma.driver.create({
      data: { name, ratePerMile: parseFloat(ratePerMile), phone, home, email },
    });

    res.json(newDriver);
  } catch (error) {
    console.error("Error creating driver:", error);
    res.status(500).json({ error: "Failed to create driver" });
  }
};

export const deleteDriver = async (req, res) => {
  try {
    const { id } = req.params;

    await prisma.driver.delete({
      where: { id: Number(id) },
    });
```

Клієнтська частина модуля реалізована у вигляді окремої сторінки, де відображається список водіїв та доступні інструменти для керування ними. Компонент DriversPage завантажує дані з сервера, відтворює їх у вигляді таблиці або карток та забезпечує основну взаємодію з модальними вікнами, що використовуються для створення або редагування водіїв. Завдяки використанню React компонент автоматично оновлює інтерфейс після виконання будь-якої операції – створення, зміни або видалення запису – що робить роботу з даними зручною та інтуїтивною. Клієнтська логіка містить асинхронні запити через axios, локальний стан для зберігання отриманих даних і механізм ефективного перерендерингу без додаткових перевантажень.

Лістинг 3.2 Підключення маршруту для водіїв:

```
import {
  getDrivers,
  createDriver,
  deleteDriver,
} from "../controllers/driverController.js";

const router = express.Router();

router.get("/", getDrivers);
router.post("/", createDriver);
router.delete("/:id", deleteDriver);
```

Модуль також містить окремий компонент модального вікна, який дозволяє вводити дані нового водія або редагувати існуючого. Цей компонент автоматично підставляє значення під час редагування, виконує мінімальну валідацію даних і відправляє їх на сервер для збереження. Модальне вікно забезпечує відокремленість логіки введення даних від основної структури сторінки, що покращує читабельність коду.

Рендеринг списку водіїв базується на проходженні масиву даних та відображенні кожного водія у вигляді окремої картки з кнопками редагування та видалення. Такий підхід дозволяє підтримувати просту та гнучку структуру інтерфейсу і швидко адаптувати її під додаткові вимоги, наприклад, виведення детальної інформації, пов’язаної з рейсами або виплатами. Логіка рендера також розташована у файлі DriversPage.jsx і може бути використана як приклад компонування інтерфейсу для інших сутностей застосунку.

Модуль керування диспетчерами є невід’ємною частиною логістичного вебзастосунку, оскільки диспетчер у транспортній компанії відіграє ключову роль в управлінні парком траків, плануванні рейсів та контролі їх виконання. Саме тому система повинна забезпечувати зручний спосіб створення, редагування, перегляду та видалення записів про диспетчерів, а також зберігати інформацію про їх контактні дані, відсоток комісії та закріплені траки. Архітектура модуля побудована за тим самим принципом, що й інші частини системи: серверна логіка реалізована через Express та Prisma, а клієнтська частина написана з використанням React.

Серверна частина складається з контролера, сервісного шару та маршрутизатора. Контролер відповідає за отримання HTTP-запитів, валідацію даних та передачу управління до сервісу, який вже напряму взаємодіє з базою даних через Prisma Client. Такий розподіл забезпечує кращу підтримуваність і дозволяє легко розширювати функціонал. У листингу 3.5 наведено фрагмент контролера, який реалізує створення, отримання, редагування та видалення диспетчера. Контролер виконує попередню перевірку даних, обробляє помилки та повертає уніфіковані JSON-відповіді, що дозволяє стабільно інтегрувати модуль з клієнтською частиною.

Сервісний шар міститься у файлі dispatcherService.js і реалізує взаємодію з реляційною базою даних. Він інкапсулює всі Prisma-запити, забезпечуючи роботу з такими сутностями, як створення диспетчера, отримання повного списку, пошук за ідентифікатором, редагування та видалення. Сервіс також при отриманні диспетчера додає пов’язану інформацію про траки, що дозволяє

формувати цілісну картину навантаження на кожного диспетчера і робить модуль придатним для подальшого розширення аналітичним функціоналом. Наведений нижче листинг демонструє структуру логіки сервісного рівня, яка є достатньо гнучкою для масштабування та впровадження нових параметрів.

Лістинг 3.6 Сервіс роботи з диспетчерами:

```
const { PrismaClient } = require('@prisma/client');
const prisma = new PrismaClient();

module.exports = {
  async getAll() {
    return prisma.dispatcher.findMany({
      include: { trucks: true }
    });
  },

  async getById(id) {
    return prisma.dispatcher.findUnique({
      where: { id: Number(id) },
      include: { trucks: true }
    });
  },

  async create(data) {
    return prisma.dispatcher.create({
      data: {
        name: data.name,
        percent: Number(data.percent),
        email: data.email,
        extNumber: data.extNumber
```

```
    }
  });
},

async update(id, data) {
  return prisma.dispatcher.update({
    where: { id: Number(id) },
    data: {
      name: data.name,
      percent: Number(data.percent),
      email: data.email,
      extNumber: data.extNumber
    }
  });
},

async remove(id) {
  return prisma.dispatcher.delete({
    where: { id: Number(id) }
  });
}
};
```

Маршрутизатор забезпечує зв’язок між URL-адресами API та відповідними методами контролера. Він визначає стандартні REST-маршрути для роботи з диспетчерами: створення, отримання всіх записів, отримання конкретного диспетчера, оновлення та видалення. Такий підхід підтримує зрозумілу структуру та полегшує документування API.

Лістинг 3.7 Маршрутизатор модуля:

```
const express = require('express');
const router = express.Router();
const controller = require('../controllers/dispatcherController');

router.get('/', controller.getAll);
router.get('/:id', controller.getById);
router.post('/', controller.create);
router.put('/:id', controller.update);
router.delete('/:id', controller.remove);
```

Клієнтська частина модуля реалізована на React і забезпечує інтуїтивно зрозумілий інтерфейс для керування диспетчерами. Компонент Dispatchers завантажує дані із серверної частини, відображає список диспетчерів і дозволяє відкривати модальні вікна для їх створення або редагування. Особливістю реалізації є використання окремого універсального модального компонента, який можна повторно застосувати в інших модулях системи. Це дозволяє зменшити дублювання коду і підтримувати єдиний стиль усіх форм введення даних. Модальне вікно для диспетчерів підтримує введення і зміни таких даних, як ім’я, комісійний відсоток, електронна адреса та внутрішній номер.

Крім того, сторінка відображає дату створення кожного диспетчера, що полегшує аудит даних і дозволяє контролювати історію змін. Дані оновлюються автоматично після кожної операції, що робить інтерфейс динамічним і зручним для реальної роботи диспетчера або менеджера транспортної компанії. Нижче наведено листинг клієнтської частини, включаючи модальне вікно та логіку CRUD-операцій.

Лістинг 3.8 Маршрутизатор модуля:

```
function Dispatchers() {
  const [dispatchers, setDispatchers] = useState([]);
  const [modalOpen, setModalOpen] = useState(false);
```

```
const [name, setName] = useState('');
const [percent, setPercent] = useState('');
const [email, setEmail] = useState('');
const [extNumber, setExtNumber] = useState('');
const loadDispatchers = async () => {
  const res = await axios.get('http://localhost:3000/api/dispatchers');
  setDispatchers(res.data);
};
const createDispatcher = async () => {
  await axios.post('http://localhost:3000/api/dispatchers', {
    name,
    percent,
    email,
    extNumber
  });
  setModalOpen(false);
  loadDispatchers();
};
useEffect(() => {
  loadDispatchers();
}, []);
return (
  <div>
    <h2>Dispatchers</h2>

    <button onClick={() => setModalOpen(true)}>Add dispatcher</button>

    {dispatchers.map((d) => (
      <div key={d.id} style={{ border: '1px solid #ccc', padding: 10, margin: 5
}}>
```

```
            <strong>{d.name}</strong> ({d.percent}%)
            <br />
            Email: {d.email || '–'}
            <br />
            Ext: {d.extNumber || '–'}
            <br />
            Trucks assigned: {d.trucks.length}
          </div>
        ))}
        {modalOpen && (
          <div className="modal">
            <h3>Create dispatcher</h3>
            <input placeholder="Name" onChange={(e) =>
setName(e.target.value)} />
            <input placeholder="Percent" onChange={(e) =>
setPercent(e.target.value)} />
            <input placeholder="Email" onChange={(e) =>
setEmail(e.target.value)} />
            <input placeholder="Ext number" onChange={(e) =>
setExtNumber(e.target.value)} />
            <button onClick={createDispatcher}>Save</button>
          </div>
        )}
      </div>
    );
  }
```

Модуль керування диспетчерами повністю реалізує необхідний життєвий цикл роботи з даними, забезпечуючи створення, збереження, редагування, видалення та відображення повної інформації про диспетчерів транспортної

компанії. Завдяки використанню Express, Prisma і React модуль має чітку архітектуру, високу надійність, чисту структуру коду та можливість розширення у напрямку детальної фінансової аналітики кожного диспетчера, включно з підрахунком їхнього гросу, середніх показників і статистики за періодами.

Модуль керування траками відіграє важливу роль у логістичному вебзастосунку, оскільки саме трак є базовою одиницею транспортної системи, до якої прив'язуються водії, диспетчери та всі виконані рейси. Коректне ведення обліку траків забезпечує прозорість логістичних процесів, точність фінансових розрахунків і можливість формувати аналітичні звіти за будь-який період. Завдяки чіткій структурі реляційної моделі сутність «Truck» поєднує кілька логістичних аспектів: інформацію про номер трака, водія, диспетчера та всі пов'язані з ним вантажі (рейси). Саме тому цей модуль є центральним елементом системи.

Серверна частина модуля реалізована за архітектурою, аналогічною до попередніх модулів: контролер отримує HTTP-запити, сервіс взаємодіє з базою даних через Prisma, а маршрутизатор зв'язує зовнішні запити з методами контролера. Контролер відповідає за обробку CRUD-операцій: створення нового трака, отримання загального списку, перегляд інформації про конкретний трак, редагування та видалення. Важливою деталлю є те, що при отриманні одного трака система повертає також пов'язані дані – водія, диспетчера та всі вантажі. Це дозволяє у майбутньому формувати розширену статистику, наприклад обсяги фрахту, відстань, що проїхав трак, або середню прибутковість.

На сервісному рівні реалізовано роботу з реляційною БД. Сервіс виконує всі запити Prisma: створення нового запису, оновлення, пошук за ID, видалення, а також включення пов'язаних сутностей за допомогою include. Це гарантує, що будь-який запит до трака повертає повну інформацію про нього, що особливо важливо для побудови фінансових звітів та дашбордів.

Лістинг 3.11 Маршрутизатор траків:

```
router.get('/', controller.getAll);
```

```
router.get('/:id', controller.getById);
router.post('/', controller.create);
router.put('/:id', controller.update);
router.delete('/:id', controller.remove);
```

У клієнтській частині модуль керування траками реалізований через React-компонент, який завантажує список траків із серверної частини та надає можливість додавати новий трак через форму. При створенні нового трака користувач вибирає водія (кожен водій може мати тільки один трак), а також диспетчера – система гарантує цілісність даних через перевірку на серверній стороні. Інтерфейс відображає номер трака, ім’я водія, ім’я диспетчера та дату створення, що забезпечує зручність перегляду та швидкий доступ до інформації.

Лістинг 3.12 Клієнтський модуль:

```
function Trucks() {
  const [trucks, setTrucks] = useState([]);
  const [modalOpen, setModalOpen] = useState(false);

  const [truckNumber, setTruckNumber] = useState('');
  const [driverId, setDriverId] = useState('');
  const [dispatcherId, setDispatcherId] = useState('');
  const [drivers, setDrivers] = useState([]);
  const [dispatchers, setDispatchers] = useState([]);
  const loadData = async () => {
    const t = await axios.get('http://localhost:3000/api/trucks');
    const d = await axios.get('http://localhost:3000/api/drivers');
    const disp = await axios.get('http://localhost:3000/api/dispatchers');
    setTrucks(t.data);
    setDrivers(d.data);
    setDispatchers(disp.data);
```

```
  };
  const createTruck = async () => {
    await axios.post('http://localhost:3000/api/trucks', {
      truckNumber,
      driverId,
      dispatcherId
    });
    setModalOpen(false);
    loadData();
  };
  useEffect(() => {
    loadData();
  }, []);
  return (
    <div>
      <h2>Trucks</h2>
      <button onClick={() => setModalOpen(true)}>Add truck</button>
      {trucks.map((t) => (
        <div key={t.id} style={{ border: '1px solid #ccc', padding: 10, margin: 5
}}>
          <strong>Truck #{t.truckNumber}</strong>
          <br />
          Driver: {t.driver?.name || '–'}
          <br />
          Dispatcher: {t.dispatcher?.name || '–'}
          <br />
          Loads: {t.loads?.length || 0}
        </div>
      ))}
      {modalOpen && (
```

```
            <div className="modal">
              <h3>Create truck</h3>
              <input placeholder="Truck number" onChange={(e) =>
setTruckNumber(e.target.value)} />
              <select onChange={(e) => setDriverId(e.target.value)}>
                <option>Select driver</option>
                {drivers.map((d) => (
                  <option key={d.id} value={d.id}>{d.name}</option>
                ))}
              </select>
              <select onChange={(e) => setDispatcherId(e.target.value)}>
                <option>Select dispatcher</option>
                {dispatchers.map((d) => (
                  <option key={d.id} value={d.id}>{d.name}</option>
                ))}
              </select>
              <button onClick={createTruck}>Save</button>
            </div>
          )}
        </div>
      );
    }
```

Таким чином, модуль керування траками забезпечує повноцінний життєвий цикл управління транспортом у логістичній компанії. Він інтегрує водіїв, диспетчерів і всі рейси, що дозволяє будувати повну картину діяльності кожного транспортного засобу. Правильна реалізація цього модуля є основою точних розрахунків та подальшої фінансової аналітики, що робить його одним із найважливіших компонентів у структурі вебзастосунку.

Модуль керування рейсами є центральним елементом логістичного вебзастосунку, оскільки саме рейс є основною одиницею, на основі якої здійснюється фінансовий розрахунок, формуються показники ефективності та ведеться загальна статистика роботи транспортної компанії. Рейс містить широкий набір параметрів: маршрут, пробіг у милях, валовий дохід, оплату водію, комісію диспетчера, додаткові витрати та фінальний прибуток. Саме тому модуль має складнішу структуру порівняно з іншими сутностями системи.

Серверна частина модуля складається з контролера, сервісного шару та маршрутизатора. Контролер відповідає за приймання HTTP-запитів, валідацію вхідних даних та виклик відповідних методів сервісу. Сервіс, у свою чергу, взаємодіє з базою даних через Prisma Client, забезпечуючи створення рейсу, його редагування, видалення та отримання повної інформації, включно з прив'язкою до конкретного трака. Використання Prisma дозволяє гарантувати, що всі зв’язки між сутностями залишаються цілісними, а операції з даними виконуються коректно та надійно.

Особливістю даного модуля є те, що при створенні рейсу частина параметрів може бути розрахована автоматично. Оскільки в системі вже реалізовано Google Maps API, у майбутньому модуль може отримувати відстань між ZIP-кодами без участі користувача. Фінансові значення, такі як виплати водіям або комісії диспетчера, також можуть обчислюватися автоматично на основі тарифів, закріплених за відповідним водієм чи диспетчером, що відкриває можливість розширення модуля у напрямку повноцінного автоматичного розрахунку вартості кожного рейсу.

Лістинг 3.14 Маршрутизатор рейсів:

```
router.get('/', controller.getAll);
router.get('/:id', controller.getById);
router.post('/', controller.create);
router.delete('/:id', controller.remove);
```

Клієнтська частина модуля реалізована у вигляді сторінки, яка дозволяє переглядати список усіх рейсів, створювати нові записи та видаляти існуючі. Інтерфейс використовує React та асинхронний select-компонент для пошуку міст за ZIP-кодами через сторонній сервіс Zippopotam.us. Такий підхід дозволяє користувачу вводити назву міста або ZIP-код і одразу отримувати підказки, що значно покращує зручність користування системою.

Форма створення рейсу включає поля для ZIP-кодів, милей та валового доходу, а також автоматично прив’язує рейс до конкретного трака. Після створення запису список рейсів оновлюється, а система відображає всі ключові параметри рейсу у компактному вигляді. Така реалізація дозволяє легко працювати з великими масивами даних, формувати майбутні аналітичні звіти та інтегрувати модуль з іншими частинами системи.

Загалом модуль керування рейсами є найбільш інформаційно насиченим, оскільки поєднує географічні дані, фінансові параметри, структурні зв’язки та механізми інтеграції із зовнішніми сервісами. Його функціональність створює основу фінансової аналітики, дає можливість будувати детальні статистичні звіти та визначати прибутковість кожного трака, диспетчера чи всієї компанії в цілому. Саме тому модуль «Loads» є ключовим для розрахунків і є невід’ємною складовою повноцінної транспортної інформаційної системи.

Модуль фінансових розрахунків є ключовим елементом логістичного вебзастосунку, оскільки він забезпечує автоматичне визначення основних економічних показників кожного рейсу. Всі логістичні компанії, незалежно від масштабу, здійснюють подібні розрахунки: визначення оплати водія, комісії диспетчера, витрат на простій, додаткових платежів за навантаження/розвантаження та загального прибутку компанії. Тому автоматизація цього процесу дозволяє уникнути помилок, прискорює обробку замовлень та суттєво полегшує подальшу побудову фінансової статистики.

Основою модуля є функція calculateLoad, яка отримує вхідні параметри рейсу – пробіг, валовий дохід (gross), тариф водія за милю, відсоток диспетчера, а також значення додаткових витрат. Усі дані попередньо перетворюються на

числовий формат для уникнення похибок та непередбачених сценаріїв. Після цього функція обчислює ключові фінансові показники.

Лістинг 3.15 Функція фінансових розрахунків:

```
function calculateLoad({
  miles = 0,
  gross = 0,
  driverRate = 0,
  dispatcherPercent = 0,
  dtHours = 0,
  layover = 0,
  tonu = 0,
  driverAssist = 0,
  lumper = 0,
}) {
  const milesNum = Number(miles) || 0;
  const grossNum = Number(gross) || 0;
  const driverRateNum = Number(driverRate) || 0;
  const dispatcherPercentNum = Number(dispatcherPercent) || 0;
  const dtHoursNum = Number(dtHours) || 0;
  const layoverNum = Number(layover) || 0;
  const tonuNum = Number(tonu) || 0;
  const driverAssistNum = Number(driverAssist) || 0;
  const lumperNum = Number(lumper) || 0;

  const driverPay = milesNum * driverRateNum;
  const dispatcherPay = (dispatcherPercentNum / 100) * grossNum;

  const dtPay = dtHoursNum * 15;
```

```
    const additionalCosts = dtPay + layoverNum + tonuNum + driverAssistNum
+ lumperNum;

    const totalExpenses = driverPay + dispatcherPay + additionalCosts;
    const profit = grossNum - totalExpenses;

    return {
      miles: milesNum,
      gross: grossNum,
      driverPay,
      dispatcherPay,
      dtPay,
      layover: layoverNum,
      tonu: tonuNum,
      driverAssist: driverAssistNum,
      lumper: lumperNum,
      additionalCosts,
      totalExpenses,
      profit,
    };
  }
```

Отримані значення повертаються на фронтенд і можуть бути відображені у модулі статистики або використані для збереження фінансової інформації у базі даних. Модуль розроблений таким чином, щоб його можна було легко розширити, наприклад, додати паливні витрати, штрафи за затримку, коефіцієнти залежно від регіону чи типу вантажу тощо.

З боку серверної частини модуль фінансових розрахунків інтегрується з маршрутизатором /calculate-load, який приймає дані від клієнта, викликає Google Maps API для визначення загальної відстані маршруту та передає ці параметри

до функції calculateLoad. Це забезпечує повністю автоматизований процес розрахунку рейсу: користувач вводить ZIP-коди маршрутів, а система самостійно визначає милі й виконує всі розрахунки без ручного втручання.

Таким чином, модуль фінансових розрахунків виконує роль «механізму бізнес-логіки» у логістичній системі. Він дозволяє уникнути людських помилок, забезпечує прозорість і повторюваність розрахунків та створює основу для детального фінансового аналізу діяльності водіїв, траків і диспетчерів. Це один з найбільш критично важливих компонентів усього вебзастосунку, оскільки саме від коректності цих розрахунків залежить прибутковість компанії.

3.3 Інструкція користувача

Розроблений логістичний вебзастосунок призначений для ведення обліку водіїв, диспетчерів, транспортних засобів та вантажів, а також для здійснення автоматичних фінансових розрахунків і формування статистики діяльності компанії. Після запуску застосунку користувач потрапляє до головного інтерфейсу, який складається з бокової панелі навігації та робочої області, де відображається необхідний функціонал. Система не потребує авторизації чи реєстрації, що дозволяє одразу перейти до роботи без додаткових налаштувань. Усі основні дії виконуються у відповідних розділах меню, де користувач може керувати даними або переглядати створені записи.

Розділ для роботи з водіями дає можливість додавати нових співробітників, указуючи їх ім’я, ставку за милю та інші контактні дані. Після створення водія він одразу відображається у списку нижче, а користувач може за потреби видалити запис одним кліком. Аналогічним чином працює і модуль диспетчерів, де додаються їхні імена, відсоток від валового доходу, а також необов’язкові атрибути на кшталт електронної пошти чи номеру внутрішньої лінії. Система відображає всіх створених диспетчерів у вигляді структурованого списку, з якого легко видаляти непотрібні записи.

У розділі управління траками користувач може створювати транспортні засоби, прив’язуючи кожен із них до відповідного водія та диспетчера. Для цього достатньо ввести номер траку та вибрати наявних співробітників із випадних списків. Щойно трак створено, він з’являється у загальному переліку, де можна переглянути, кому саме він закріплений. Така структура спрощує організацію автопарку та дозволяє уникнути дублювання інформації.

Розділ вантажів є центральним компонентом системи, оскільки саме тут створюються рейси та виконуються всі розрахунки, включно з визначенням пробігу, валового доходу і додаткових витрат. Під час створення вантажу користувач вводить пункти відправлення та призначення з можливістю автопошуку, значення пробігу або використовує дані, отримані автоматично через Google Maps API, а також вказує валовий дохід і усі додаткові фінансові параметри. Після заповнення форми система миттєво розраховує оплату водія, комісію диспетчера, загальні витрати та прибуток компанії. Створений вантаж додається до бази та відображається у списку разом із можливістю його видалення.

Важливою частиною застосунку є модуль статистики, який дозволяє аналізувати діяльність у зручному агрегованому форматі. Система формує дані для трьох основних категорій: по диспетчерах, по траках та загальну статистику компанії. Для диспетчерів відображається сумарний валовий дохід, кількість виконаних вантажів, розмір комісії та середній прибуток. Для траків система показує загальні пробіги, кількість рейсів, заробіток водія, валовий дохід і чистий прибуток по кожному транспортному засобу. У загальній статистиці користувач може побачити інформацію щодо всіх рейсів разом: кількість перевезень, сукупний прибуток, загальні витрати та інші важливі параметри. Усі дані оновлюються автоматично після додавання або видалення вантажів.

Застосунок передбачає інтуїтивно зрозуміле повідомлення про помилки: якщо користувач залишає певні поля незаповненими або вводить некоректні значення, система попереджає про це та блокує створення некоректного запису, що забезпечує цілісність бази даних і правильність фінансових обчислень. Після

завершення роботи користувачу не потрібно виконувати додаткових дій: усі дані автоматично зберігаються у локальному файлі SQLite, і система готова до подальшого використання при наступному запуску.

3.4 Використання статистики та її роль в управлінні логістичними перевезеннями

Модуль статистики, інтегрований у логістичний вебзастосунок, відіграє ключову роль у прийнятті управлінських рішень і забезпечує всебічний аналітичний огляд роботи компанії. На відміну від традиційних таблиць або ручного підрахунку результатів, система автоматично збирає всі дані з рейсів, водіїв, диспетчерів і траків, узагальнюючи їх у вигляді зручних аналітичних показників.

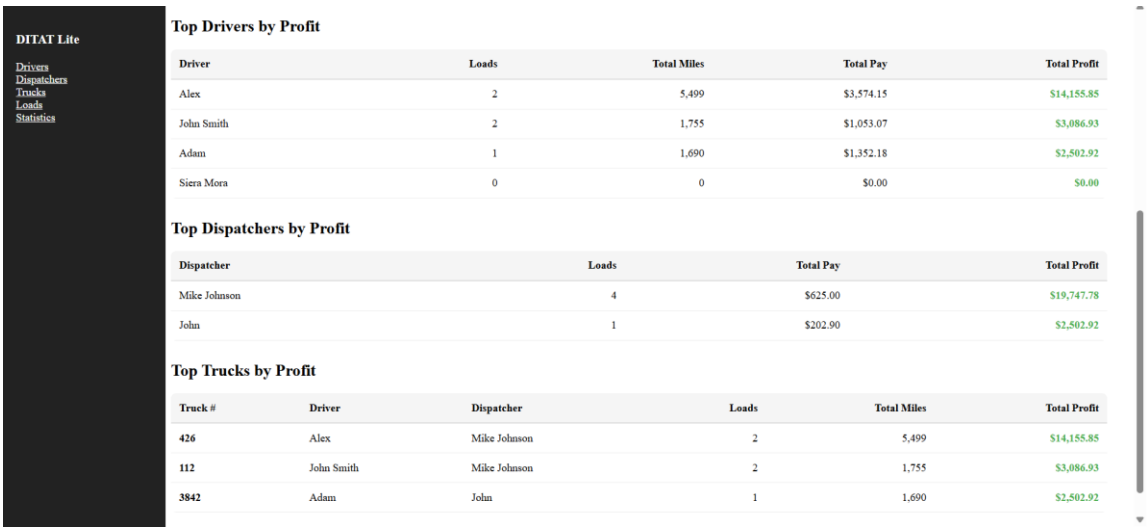

Рисунок 3.1 – Статистика по водіям, тракам і диспетчерам

Це дозволяє оперативно оцінювати ефективність кожного співробітника та транспортного засобу, виявляти слабкі місця в роботі автопарку та прогнозувати фінансові результати.

Статистика по диспетчерах надає можливість аналізувати продуктивність кожного з них на основі валового доходу, кількості відпрацьованих рейсів і

величини комісійних виплат(рис. 3.1). Такий підхід дозволяє визначити, який диспетчер працює найбільш ефективно, які клієнти або напрямки дають найкращий результат, а також наскільки збалансоване навантаження між співробітниками. Система автоматично підсумовує всі показники, що мінімізує ризик помилки та значно прискорює фінансові розрахунки, які в реальних компаніях часто проводяться вручну або в Excel.

Не менш важливим є аналіз статистики по траках, який дає змогу контролювати ефективність використання кожного транспортного засобу. У системі враховуються такі параметри, як загальний пробіг, кількість виконаних вантажів, загальний прибуток, витрати та виплати водієві. Це дає змогу виявляти надмірні витрати, оптимізувати завантаженість автопарку і планувати технічне обслуговування. Порівняння траків між собою дозволяє швидко зрозуміти, який транспорт приносить максимальну вигоду, а який працює нижче очікуваної ефективності.

Загальна статистика компанії формує повну картину всіх логістичних операцій за будь-який вибраний період. Система підсумовує валовий дохід, витрати, кількість перевезень, сумарний пробіг, загальний прибуток та інші фінансові показники. Це дозволяє оцінити рентабельність роботи компанії в цілому і швидко робити висновки щодо успішності поточного місяця чи кварталу. Завдяки автоматизації підрахунків користувач отримує точні результати без ручного введення формул і ризику арифметичних помилок.

Рисунок 3.2 – Головна та фінансова статистика

Використання статистики у вебзастосунку створює цінність для керівництва й аналітиків логістичної компанії, оскільки дозволяє бачити не лише окремі дані, а й взаємозв’язки між ними (рис. 3.2). Наприклад, можна визначити, як продуктивність диспетчера впливає на роботу конкретного траку, або як зміна ставки водія позначається на загальному прибутку за місяць. Така інформація є основою для ухвалення стратегічних рішень – від оптимізації маршрутів та скорочення витрат до перерозподілу автопарку та розробки нової системи мотивації.

У підсумку, модуль статистики виступає не просто інструментом для відображення числових даних, а повноцінною аналітичною платформою, що забезпечує глибоке розуміння бізнес-процесів. Його використання підвищує прозорість роботи компанії, покращує якість управління логістичними операціями та сприяє зростанню ефективності перевізного процесу.

3.5 Перспективи вебзастосунку для подальших досліджень

У розробленому логістичному вебзастосунку реалізовано базовий, але функціонально завершений набір можливостей: облік водіїв, диспетчерів, траків, рейсів, автоматичні розрахунки та детальна статистика. Проте потенціал розвитку системи значно ширший. Подальше вдосконалення може охоплювати як технічні, так і аналітичні аспекти, що відкриє нові можливості для автоматизації логістичних процесів, підвищить точність планування перевезень і зробить систему придатною для промислового використання у великих транспортних компаніях.

Одним із напрямів розвитку є інтеграція повноцінних картографічних інструментів Google Maps або інших сервісів, що дозволило б не лише отримувати відстані між ZIP-кодами, а й будувати оптимальні маршрути, враховувати затори, поточний стан доріг, ремонтні роботи та альтернативні

шляхи. Додавання автоматичного перерахунку маршруту в разі непередбачених обставин зробило б систему ближчою до реальних умов роботи логістичних компаній.

Перспективним є також впровадження багаторівневої статистики, яка включала б прогнозування фінансових показників на основі історичних даних. Використання методів машинного навчання (наприклад, регресійних моделей або моделей часових рядів) могло б забезпечити можливість передбачати майбутній грос, завантаженість траків, витрати чи прибутковість окремих напрямків. Це дало б змогу менеджерам приймати рішення не лише на основі минулих результатів, а й з урахуванням потенційних тенденцій розвитку.

Подальший розвиток застосунку може включати й розширення функціональності обліку витрат. Наразі система враховує основні види платежів - DT, layover, tonu, lumper, driver assist. У майбутньому додавання детального обліку витрат на пальне, штрафів, технічне обслуговування, платних доріг або митних платежів дозволило б побудувати повну фінансову модель роботи автопарку. Це зробило б аналітику значно точнішою та кориснішою для довгострокового планування.

З технічного боку система може бути адаптована до роботи з більш продуктивними серверними технологіями. Перехід від SQLite до PostgreSQL відкриває можливість масштабування, одночасної роботи великої кількості користувачів, складніших транзакцій та швидшого опрацювання великих обсягів даних. Додатково можна реалізувати систему ролей і доступів до адміністратора, менеджера, диспетчера, водія, що перетворить застосунок на багатокористувацьку корпоративну платформу.

Важливою перспективою розвитку є впровадження мобільної версії або PWA-застосунку, що дозволить водіям отримувати дані про рейс, надсилати оновлення статусу, прикріплювати документи, фотографії POD або BOL. Це покращить комунікацію між диспетчерами та водіями і скоротить час обробки інформації.

Ще одним напрямом удосконалення є автоматизація фінансових операцій. Інтеграція з сервісами виставлення інвойсів, системами електронних платежів або бухгалтерськими платформами дала б змогу автоматично генерувати звіти, інвойси, відомості по зарплаті та фінансові документи компанії. Це значно зменшить навантаження на офісних працівників і мінімізує ризик людської помилки.

Таким чином, розроблений вебзастосунок має великий потенціал подальшого розвитку й може стати основою для комплексної логістичної системи. Його модульна архітектура дозволяє розширювати функціональність без суттєвої перебудови існуючої структури. Перспективи досліджень охоплюють удосконалення маршрутних алгоритмів, глибшу аналітику, використання ШІ, масштабування інфраструктури та створення повноцінної корпоративної платформи для логістичних операцій. Усе це забезпечує довгострокову життєздатність проєкту та робить його корисним інструментом для реальних транспортних компаній [48-49].

ВИСНОВКИ

Таким чином, у кваліфікаційній роботі було досліджено процеси управління логістичними перевезеннями та розроблено вебзастосунок, який автоматизує розрахунок вартості рейсів, облік водіїв, диспетчерів, траків, а також формування детальної статистики діяльності транспортної компанії. У ході роботи вирішено такі основні завдання:

– проведено аналіз літературних джерел та існуючих інформаційних систем у сфері логістики, що дало змогу визначити недоліки традиційних методів ручного обліку перевезень та обґрунтувати необхідність цифровізації цих процесів;

– досліджено сучасні підходи до визначення відстаней і побудови маршрутів з використанням зовнішніх API, зокрема картографічного сервісу Google Maps, що дозволило інтегрувати в систему механізм розрахунку відстаней між пунктами відправлення та призначення;

– сформовано математичну модель обчислення вартості логістичного рейсу, яка враховує пробіг, валовий дохід, оплату водієві, комісію диспетчера, додаткові витрати (DT, layover, TONU, lumper, driver assist), а також кінцевий прибуток компанії;

– розроблено структуру реляційної бази даних і реалізовано її у вигляді моделі Prisma на основі SQLite, що забезпечило стабільне зберігання інформації про всі ключові сутності – водіїв, диспетчерів, траки та вантажі;

– створено серверну частину застосунку з використанням Express.js та Prisma ORM, що забезпечило якісну взаємодію з базою даних, а також надійність виконання CRUD-операцій;

– реалізовано клієнтську частину на основі React та Vite, спроєктовано зручний інтерфейс користувача для роботи з даними, створення рейсів, введення маршрутів та перегляду результатів розрахунків;

– розроблено модуль статистики, який автоматично аналізує діяльність за різними напрямами – по диспетчерах, траках та загальну статистику компанії

– що підвищило аналітичні можливості системи та дозволило проводити глибшу оцінку ефективності логістичних операцій.

У результаті виконання роботи створено повнофункціональний логістичний вебзастосунок, який поєднує інструменти розрахунку вартості перевезень, автоматизації фінансових операцій, побудови маршрутів та формування розширеної статистики. Система дозволяє суттєво скоротити час обробки логістичних даних, уникнути помилок ручного підрахунку та підвищити прозорість роботи компанії.

Наукова новизна роботи полягає в інтеграції зовнішнього картографічного сервісу, реляційного моделювання та автоматизованих фінансових алгоритмів у межах єдиного вебінтерфейсу, що забезпечує комплексну підтримку логістичного процесу. Отримані результати можуть бути застосовані транспортними компаніями для оптимізації маршрутів, зниження витрат, удосконалення системи обліку та ухвалення більш обґрунтованих управлінських рішень.

Перспективи подальших досліджень включають розширення функціональності системи за допомогою прогнозного аналізу, інтеграції з мобільними додатками для водіїв, переходу на промислові СУБД, а також впровадження інтелектуальних алгоритмів оптимізації маршрутів і планування логістичних процесів.

Результати роботи апробовано у вигляді 2 тез доповідей під час Міжнародної науково-практичної конференції science, technology and culture: integration and prospects, XXIX Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mukta, M. S. H., Ahmed, J., Raiaan, M. A. K., Fahad, N. M., Islam, M. N., Imtiaz, N., ... & Azam, S. (2024). Behavior based group recommendation from social media dataset by using deep learning and topic modeling. SN Computer Science, 5(6), 712.

2. Кобилін, О., Вечірська, І., & Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 63-76.

3. Roberts, E. C. (2025). Musical Remembering in the Age of Social Media (Doctoral dissertation, The University of Western Ontario (Canada)).

4. Mashtalir, S. V., & Lendel, D. P. (2025). AI-Generated video evaluation by fragment processing. Herald of Advanced Information Technology, 8(3), 316-325.

5. Schedl, M., Knees, P., McFee, B., & Bogdanov, D. (2021). Music recommendation systems: Techniques, use cases, and challenges. In Recommender systems handbook (pp. 927-971). New York, NY: Springer US.

6. Xiao, Y., Ruan, H., Zhao, X., Jin, P., Tian, L., Wei, Z., ... & Liu, L. (2024). An efficient bi-modal fusion framework for music emotion recognition. IEEE Transactions on Affective Computing.

7. Mashtalir, S. V., & Kovtunenko, A. R. (2025). Improved segmentation model to identify object instances based on textual prompts. Вісник сучасних інформаційних технологій, 8(1), 54-66.

8. Seufitelli, D. B. (2023). Understanding musical success beyond hit songs: characterization and analyses of musical careers.

9. Mashtalir[1], S. V., & Nikolenko, O. V. (2025). Tree-Based Classification of the Technical. Advances in Computer Science for Engineering and Education VII: Volume 1, 242, 339.

10. Bilonoh, B., Bodyanskiy, Y., Kolchygin, B., & Mashtalir, S. (2021, May). Tunable activation functions for deep neural networks. In International Scientific

Conference “Intellectual Systems of Decision Making and Problem of Computational Intelligence” (pp. 624-633). Cham: Springer International Publishing.

11. Mashtalir, S. V., & Lendel, D. P. (2024). Moving object shape detection by fragment processing. Herald of Advanced Information Technology, 7(4), 414-423.

12. Bodyanskiy, Y. V., Tyshchenko, O. K., & Mashtalir, S. V. (2019, May). Fuzzy clustering high-dimensional data using information weighting. In International Conference on Artificial Intelligence and Soft Computing (pp. 385-395). Cham: Springer International Publishing.

13. Mashtalir, S. V., & Nikolenko, O. V. (2024, October). Top-K Hierarchical Classification for Precision in Automotive Technical Data Analysis. In 2024 IEEE 19th International Conference on Computer Science and Information Technologies (CSIT) (pp. 01-04). IEEE.

14. Mashtalir, S. V., & Nikolenko, O. V. (2024). Optimizing hierarchical classifiers with parameter tuning and confidence scoring. Вісник сучасних інформаційних технологій, 7(3), 231-242.

15. Mashtalir, S. V., & Nikolenko, O. V. (2024, April). Tree-Based Classification of the Technical Ukrainian Texts. In International Conference on Computer Science, Engineering and Education Applications (pp. 339-348).

16. Mashtalir, S. V., & Nikolenko, O. V. (2023). Data preprocessing and tokenization techniques for technical Ukrainian texts. Applied Aspects of Information Technology, 6 (3), 318-326.

17. Mashtalir, S., & Nikolenko, O. (2024). Hierarchical classification of ukrainian technical texts using tree-based models: applications in the automotive industry.

18. Challapalli, S. S. N., Kaushik, P., Suman, S., Shivahare, B. D., Bibhu, V., & Gupta, A. D. (2021, November). Web Development and performance comparison of Web Development Technologies in Node. js and Python. In *2021 International Conference on Technological Advancements and Innovations (ICTAI)* (pp. 303-307). IEEE.

19. Vayadande, K., Purohit, S., Rathod, C., Rathod, M., Rathi, P., & Rathi, P. (2024). Web Development and Performance Comparison of Web Development Frameworks: A Review Paper. *Grenze International Journal of Engineering & Technology (GIJET)*, *10*.

20. Sharma, S., Singh, P., Sain, J., Shrivastava, V., & Pandey, A. (2024, March). Modern Backend Development Technologies: A Comparative Review and Case Study. In *International Conference on Emerging Trends in Expert Applications & Security* (pp. 139-151). Singapore: Springer Nature Singapore.

21. Lazuardy, M. F. S., & Anggraini, D. (2022). Modern front end web architectures with react. js and next. js. *Research Journal of Advanced Engineering and Science*, *7*(1), 132-141.

22. Li, Y., Liu, K., Satapathy, R., Wang, S., & Cambria, E. (2024). Recent developments in recommender systems: A survey. *IEEE Computational Intelligence Magazine*, *19*(2), 78-95.

23. Zhou, H., & et al. (2023). A Comprehensive Survey of Recommender Systems: Integration with Deep Learning. Applied Sciences, 13(20), 11378

24. Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*(1), 59.

25. Nguyen, T. T., Quoc Viet Hung, N., Nguyen, T. T., Huynh, T. T., Nguyen, T. T., Weidlich, M., & Yin, H. (2024). Manipulating recommender systems: A survey of poisoning attacks and countermeasures. *ACM Computing Surveys*, *57*(1), 1-39.

26. Ge, Y., Liu, S., Fu, Z., Tan, J., Li, Z., Xu, S., ... & Zhang, Y. (2024). A survey on trustworthy recommender systems. *ACM Transactions on Recommender Systems*, *3*(2), 1-68.

27. Aljunid, M. F., Manjaiah, D. H., Hooshmand, M. K., Ali, W. A., Shetty, A. M., & Alzoubah, S. Q. (2025). A collaborative filtering recommender systems: Survey. *Neurocomputing*, *617*, 128718.

28. Lopez-Avila, A., & Du, J. (2025). A Survey on Large Language Models in Multimodal Recommender Systems. *arXiv preprint arXiv:2505.09777*.

29. Salminen, J., Mustak, M., Jung, S. G., Makkonen, H., & Jansen, B. J. (2025). Decoding deception in the online marketplace: enhancing fake review detection with psycholinguistics and transformer models. *Journal of Marketing Analytics*, 1-18.

30. Novoa-Paradela, D., Fontenla-Romero, O., & Guijarro-Berdiñas, B. (2024). Explained anomaly detection in text reviews. *Engineering Applications of Artificial Intelligence*, *133*, 108065.

31. Gupta, R., Jindal, V., & Kashyap, I. (2024). Recent state-of-the-art of fake review detection: a comprehensive review. *The Knowledge Engineering Review*, *39*, e8.

32. Sun, P., Bi, W., Zhang, Y., Wang, Q., Kou, F., Lu, T., & Chen, J. (2024). Fake Review Detection Model Based on Comment Content and Review Behavior. *Electronics*, *13*(21), 4322.

33. Hasan, E., Rahman, M., Ding, C., Huang, J. X., & Raza, S. (2025). Based recommender systems: a survey of approaches, challenges and future perspectives. *ACM Computing Surveys*, *58*(1), 1-41.

34. Wang, B. (2023). *Towards trustworthy large language models* (Doctoral dissertation, University of Illinois at Urbana-Champaign).

35. Ibrahim, O. A. S., Younis, E. M., Mohamed, E. A., & Ismail, W. N. (2025). Revisiting recommender systems: an investigative survey. *Neural Computing and Applications*, *37*(4), 2145-2173.

36. Кобилін, О., Вечірська, І., & Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 63-76.

37. Шафроненко, А., Бодянський, Є., & Плісс, І. (2022). Нечіткі методи інтелектуального аналізу даних.

38. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5-12.

39. Mashtalir, S. V., & Nikolenko, O. V. (2025). From classification to taxonomy: Automated structuring of vehicle repair names in multilingual corpora. Вісник сучасних інформаційних технологій, 8(2), 151-163.

40. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5-12.

41. Tvoroshenko, I., & Gorokhovatskyi, V. (2022). The Application of Hybrid Intelligence Systems for Dynamic Data Analysis.

42. Bohdan N., Tvoroshenko I., Gorokhovatskyi V., and Kobylin O. (2025) Development of a hybrid method to enhance context memory for a chatbot application based on large language models, *International Journal of Academic Information Systems Research*, 9(10), pp. 7-18.

43. Suprun A., Tvoroshenko I., Gorokhovatskyi V., and Yakovleva O. (2025) Development and research of a method for the combined use of large language models for text generation, *International Journal of Academic and Applied Research*, 9(10), pp. 249-263.

44. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). Image classification method modification based on model of logic processing of bit description weights vector. Telecommunications and Radio Engineering, 79(1).

45. Huang, C., Huang, H., Yu, T., Xie, K., Wu, J., Zhang, S., ... & Yao, L. (2025). A Survey of Foundation Model-Powered Recommender Systems: From Feature-Based, Generative to Agentic Paradigms. *arXiv preprint arXiv:2504.16420*.

46. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5-12.

47. Ng, A., & Mehrotra, R. (2020). Investigating the impact of audio states and transitions for track sequencing in music streaming sessions. In Proceedings of the 14th ACM Conference on Recommender Systems (pp. 697–702).

48. Поляков В.Д., Машталір С.В. (2025). Розробка вебзастосунку для логістики з розрахунку вартості перевезень та пошуку найкоротших маршрутів.

XXIX міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті» (pp. 122-123).

49. Поляков В.Д., Машталір С.В. (2025). Розробка інформаційної системи для логістики з автоматизацією розрахунку вартості, пошуком найкоротших маршрутів та аналітикою прибутковості перевезень. Міжнародна науково-практична конференція science, technology and culture: integration and prospects (pp. 54-56).